INFODIDACT'2022

15. Informatika Szakmódszertani Konferencia

Előadáskötet

2022

Szerkesztette: Dr. Szlávi Péter, Dr. Zsakó László
Megjelenés: 2023. január

ISBN 978-615-80608-6-8

Bírálók

Abonyi-Tóth Andor, ELTE Informatikai Kar

Bakonyi Viktória, ELTE Informatikai Kar

Bernát Péter, ELTE Informatikai Kar

Csernoch Mária, Debreceni Egyetem Informatikai Kar

Erdősné Németh Ágnes, ELTE Informatikai Kar

Holló Csaba, Szegedi Tudományegyetem Természettudományi és Informatikai Kar

Horváth Sándor, Sapientia Erdélyi Magyar Tudományegyetem

Kelemen András, SZTE Juhász Gyula Pedagógusképző Kar

Király Sándor, Eszterházy Károly Katolikus Egyetem Informatikai Kar

Kiss Attila, Nemzeti Adatvédelmi és Információszabadság Hatóság

Koczka Ferenc, Eszterházy Károly Katolikus Egyetem Informatikai Kar

Kovács Lehel, Sapientia Erdélyi Magyar Tudományegyetem

Lénárd András, ELTE Tanító- és Óvóképtő Kar

Menyhárt László, ELTE Informatikai Kar

Nikházy László, ELTE Informatikai Kar

Obonya Juraj, Tomas Bata University in Zlín

Osztián Erika, Sapientia Erdélyi Magyar Tudományegyetem

Piláth Károly, ELTE Trefort Ágoston Gyakorló Gimnázium

Pluhár Zsuzsa, ELTE Informatikai Kar

Prantner Csilla, Eszterházy Károly Katolikus Egyetem Informatikai Kar

Psenák Ildikó, Trnava University in Trnava

Szabó Tibor, Constantine the Philosopher University in Nitra

Törley Gábor, ELTE Informatikai Kar

Vincellér Zoltán, ELTE Informatikai Kar

Virányi Anita, ELTE Bárczi Gusztáv Gyógypedagógiai Kar

Tartalom

A CANVAS LMS használata az ELTE képzéseiben – oktatók támogatása, tapasztalatok

Dr. Abonyi-Tóth Andor

abonyita@inf.elte.hu
ELTE IK, ELTE OTO

Absztrakt. A Canvas LMS használatára a 2016/2017-es tanévtől van lehetőség az ELTE képzéseiben. A rendszer bevezetését megelőzően, illetve azzal párhuzamosan szükség volt különböző támogatási formák (pl. módszertani képzések, segédanyagok) kidolgozására, illetve egyedi fejlesztésekre, amelyekkel a rendszer összekapcsolásra kerülhetett a Neptun tanulmányi rendszerrel. Cikkemben bemutatom, hogy az ELTE Oktatási Igazgatóság Oktatásfejlesztési és Tehetséggondozási Osztálya (ELTE OTO) milyen e-learning támogatást biztosít az oktatók számára, valamint összegzem a Canvas LMS használatára vonatkozó adatokat és visszajelzéseket.

Kulcsszavak: e-learning, Canvas LMS, módszertani támogatás, statisztika

1. A Canvas LMS bevezetése és új funkciókkal való bővítése

A Canvas LMS rendszert 2011-ben indította útjára az Instructure[1] cég. A rendszer két konstrukcióban használható. Az intézmények regisztrálhatnak a központilag telepített és adminisztrált szolgáltatásra, amelyért előfizetési díjat kell fizetniük a felhasználók száma alapján. A másik lehetőség, hogy a rendszer Opensource változatát[2] használja az adott intézmény, de ebben az esetben saját hatáskörben kell azt telepítenie, illetve adminisztrálnia, amelyhez megfelelő szaktudással rendelkező rendszergazdákra, illetve portál adminisztrátorokra van szükség. A Canvas LMS rendszerre jellemző, hogy az USA-ban rendkívül népszerű, a piaci részesedése 30% feletti [1].

Az LMS rendszerek felsőoktatási intézményekben történő bevezetése számos kihívást tartogat. Ilyen például az is, hogy az LMS rendszereket hogyan lehet más digitális platformokba integrálni [1]. Az ELTE a 2016/2017-es tanév II. szemeszterében tette lehetővé először a Canvas LMS nyílt forráskódú változatának használatát az oktatók és hallgatók számára, párhuzamosan a Moodle rendszer használatával. Kezdetben a kurzusokat az adminisztrátorok manuálisan hozták létre az oktatók által kitöltött űrlapok alapján. 2017. januárjától (az ELTE belső fejlesztésének köszönhetően) azonban már lehetővé vált, hogy az e-learning kurzusok létrehozását a Neptun tanulmányi rendszerben közvetlenül kezdeményezhessék az oktatók, amelynek eredményeképpen a kurzusokhoz a hallgatók és oktatók is automatikusan hozzáférést kaptak. 2017. szeptemberétől (szintén belső fejlesztésnek köszönhetően) elérhetővé vált, hogy a Canvas (illetve Moodle) LMS rendszerekben összevontan lehessen kurzusokat létrehozni egy közös csoportkód megadásával. A megoldás lényege, hogy az azonos csoportkódba tartozó kurzusok résztvevői ugyanazt a kurzusfelületet látják, azonban az oktatók differenciáltan adhatnak ki feladatokat az egyes csoportok számára, eltérő beadási határ-

[1] Instructure | Educational Software Development
https://www.instructure.com/ (utoljára megtekintve: 2022. 10.20.)

[2] GitHub – instructure/canvas-lms: The open LMS by Instructure, Inc.
https://github.com/instructure/canvas-lms (utoljára megtekintve: 2022. 10.20.)

időkkel, illetve tartalommal. Így például nem szükséges minden gyakorlatvezetőnek saját kurzusában kiírni a beadandó feladatokat, azokat minden csoport azonos felületen töltheti fel. A beadandó feladatok javítását segíti, hogy a gyakorlatvezetők leszűrhetik a beadott feladatokat a saját csoportjaik szerint.

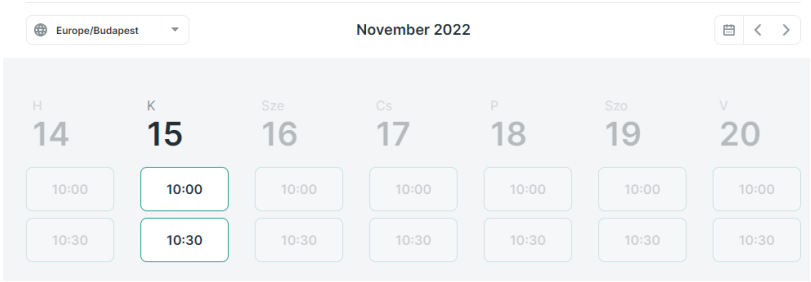
2. Az oktatók technikai, módszertani és anyagi támogatásának formái

Az oktatók technikai és módszertani támogatása többféle módon történik. Egyrészt e-learning ügyfélszolgálat segíti az oktatókat, másrészt rendszeresen indulnak képzések, illetve kerülnek publikálásra cikkek, segédanyagok, videók, amelyek mind-mind segítik az oktatói munkát. Az e-learninggel kapcsolatos aktualitásokat havi hírlevél foglalja össze. Emellett rendszeresen kiírásra kerülnek tananyagfejlesztési pályázatok is. A következőkben ezen támogatási formákat részletesebben is bemutatjuk.

2.1. E-learning ügyfélszolgálat, szakértői konzultáció

Az ELTE Oktatási Igazgatóság Oktatásfejlesztési és Tehetséggondozási Osztálya[3] dedikált email címet biztosít az e-learninggel kapcsolatosan kérdések fogadására, illetve a problémák jelzésére.

Az osztály munkáját e-learning szakértők is segítik, akik jellemzően az ELTE főállású oktatói, illetve munkatársai. A szakértők minden hétre meghirdetnek fogadóórákat, amelyekre az oktatók online időfoglalási rendszer segítségével jelentkezhetnek fel. Jómagam kezdetektől fogva ellátok ilyen szakértői feladatot.



1. ábra: Részlet az automatizált időpontfoglalási rendszerből

Az e-learning szakértő feladata, hogy segítsen az oktatóknak a korszerű digitális pedagógiai-módszertani ismereteket integrálni a (felső)oktatási gyakorlatba, tisztázni a hozzá forduló oktatók igényeit, megtalálva az online tanulási környezetben azokat a megoldásokat, amelyek segítik céljai elérésében, folyamatában látni és láttatni egy kurzus felépítését, szerkezetét, átgondolni előnyeit, lehetőségeit, hátrányait vagy korlátait, lehetséges kimeneteleit egy-egy módszer alkalmazásának, feltárni azokat a területeket és forrásokat, amelyeken és amelyek segítségével közelebb kerülhet pedagógiai céljai megvalósításához. [2]

A folyamat során két egyenrangú szakember dolgozik együtt, akik más-más területen rendelkeznek szaktudással, de együtt gondolkodnak arról, hogyan lehet a tanulási-tanítási folyamatot eredményesebbé tenni az online térben. Az oktató hozza a szaktárgyával kapcsolatos tanítási tapasztalatait, a tanulócsoportjával kapcsolatos ismereteit, a konzultáns a digitális pedagógia szakértője, egyaránt alkalmaz általános pedagógiai-andragógiai és technológiai tudáselemeket, és ezeket egyesítve az elvek

[3] https://www.elte.hu/elearning/munkatarsak

és módszerek olyan új összességét tudják használni, ami a felmerült problémák adekvát megoldása lehet. A konzultáns és oktató egyaránt tanul, hiszen minden konzultáció egyedi kérdéseket vet fel, és olyan megoldásoknak kell születni, amiket az oktató elég személyre szabottnak tart ahhoz, hogy a gyakorlatba is átvigyen. Akár több alkalommal is találkozik a konzultáns és az oktató, hogy megbeszéljék az oktató előrehaladását. Kritikus pont, hogy az oktató is elköteleződjön az aktív megoldáskeresésben, gyakorlata felülvizsgálatában és változtatásában. [2]

2.2. Módszertani cikkek

E-learning portálunkon[4] közel 100 cikk érhető el publikusan. Ezek egy része általános módszertani elveket, tippeket mutat be (pl. Hogyan készítsük fel a hallgatókat a vizsgáztatásra?; Egyetemes tervezés az oktatásban (UDL); Podcastkészítés – Mi kell egy jó oktatási podcasthez?).

2. ábra: Módszertani cikkek az ELTE e-learning portálon

A cikkek másik kategóriájába tartoznak azok, amelyek konkrétan egy adott alkalmazásra, LMS rendszerre koncentrálnak, és abban elérhető, haladó kurzusszervezési és értékelési lehetőségeket [3] és egyéb funkciókat mutatnak be példákon keresztük (pl. Előrehaladási napló exportálása Canvasból; Valós idejű és aszinkron interaktív tanulás a Nearpoddal).

A cikkeket az ELTE e-learning szakértői készítik, havonta átlagosan 4-5 új cikk kerül publikálásra.

2.3. Képzések, önjáró (self paced) kurzusok

Az oktatók számára különböző jellegű, témájú képzések állnak rendelkezésre. A képzések egy része konkrét LMS rendszerekkel foglalkozik (pl. Canvas kezdő/haladó, Moodle kezdő/haladó kurzusok), azonban elérhetőek oktatásmódszertani képzések is (pl. Online kurzusok módszertana). A képzések jelenléti, illetve online formában is zajlanak.

2021-ben önjáró (self-paced) kurzusokkal is bővítettük a képzési portfoliót. Ezen online kurzusokra az oktatók szabadon jelentkezhetnek, és saját időbeosztásuknak megfelelően végezhetik el. A kurzusok a Canvas LMS rendszerben érhetőek el.

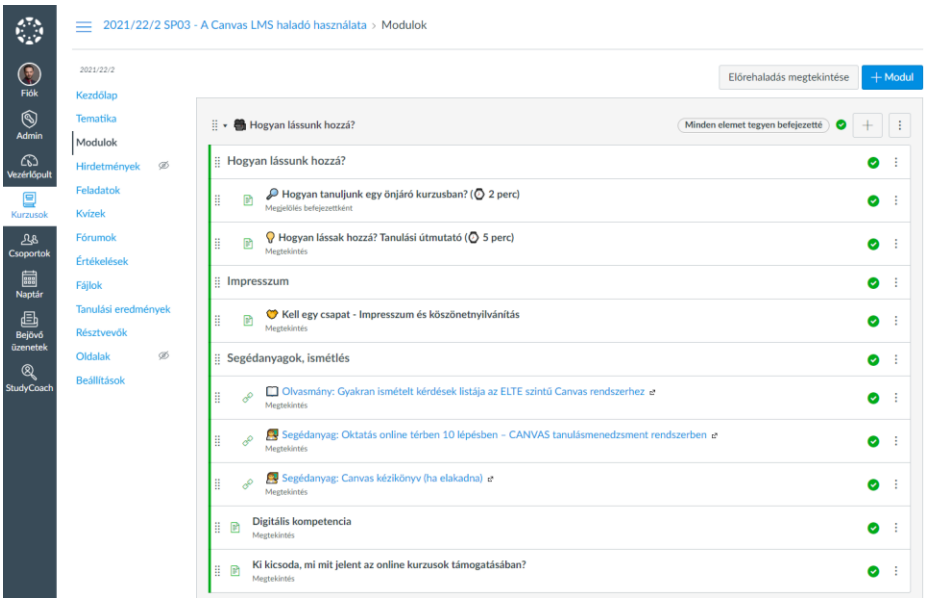
Amennyiben a résztvevők teljesítik a kurzus követelményeit, névre szóló tanúsítványt tölthetnek le a rendszerből, amely automatikus kerül legenerálásra. Ezen funkciót már magunk fejlesztettük ki annak érdekében, hogy a tanúsítványok kiadása minél kevesebb adminisztrációval járjon.

Jelenleg az alábbi témákban indulnak önjáró kurzusaink:

- A Canvas LMS haladó használata
- A tükrözött osztályterem lehetőségei a felsőoktatási gyakorlatban
- Bevezetés a grafikai tervezés alapjaiba
- Canvas: értékelési rendszer tervezése
- Canvas: Kvízek és kérdésbankok

[4] https://www.elte.hu/elearning

- Csoportmunka támogatása online
- Figyelemfelkeltő és hallgatói tevékenységet ösztönző előadások tervezése és megvalósítása
- Grafikai alapok: Betűk és színek
- Interaktív prezentációk a Wooclap alkalmazással
- Online kurzustartalom készítése



3. ábra: Példa egy önjáró kurzus felépítésére

2.4. Miniképzések egy kávé mellett (drop in sessions)

A képzési palettát színesítik azon miniképzések[5] is (15-30 percben), amelyek egy kisebb témára koncentrálnak, kötetlenebb beszélgetést tesznek lehetővé, akár egy kora délutáni kávézás közben. Ezen képzések online valósulnak meg, azokról felvétel is készül, így utólag is megtekinthetőek.

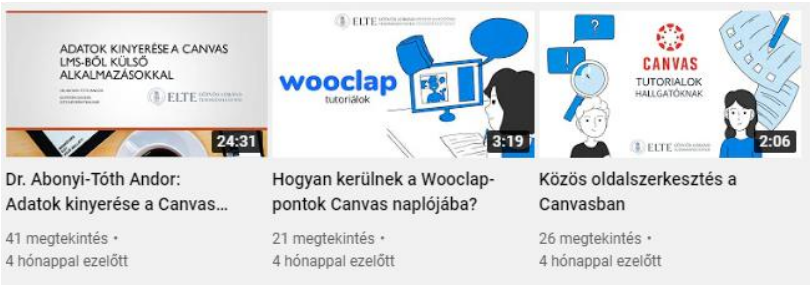
Néhány cím a közelmúlt képzéseiből: Számolásos feladatok elhelyezése kvízekben; Adatok kinyerése a Canvas LMS-ből külső alkalmazásokkal; Hogyan használjuk az értékelőtáblát a digitálisan beadott hallgatói munkák értékelésénél Moodle, Canvas és Teams kurzusokban?

2.5. Oktatóvideók

A cikkekben és online képzésekben számos magyarázó videó került elhelyezésre. Ezek egy önálló Youtube csatornán[6] is publikálásra kerültek. A videók között rövid (2-3 perces), egy adott problémára koncentráló felvételek, illetve hosszabb, átfogóbb témákat érintő felvételek is elérhetőek.

[5] https://www.elte.hu/elearning/eloadasok

[6] https://www.youtube.com/channel/UC9l892if9b6Hol3QKgslqyg



4. ábra: Részlet az oktatóvideók oldalról

2.6. Segédanyagok, dokumentációk, gyakori kérdések

Az ELTE e-learning portál Kézikönyvek oldalán[7] az ELTE képzéseiben használt rendszerekhez (Canvas, Moodle, Webex, Panopto, NeoLms, Microsoft Teams) készített kézikönyvek, segédanyagok, képernyővideók, illetve gyakran ismételt kérdések érhetőek el.

Ezek segítik az oktatókat (és a hallgatókat is) a rendszerek önálló megismerésében, illetve egyes problémák önálló megoldásában. Kifejezetten a hallgatók számára készített segédanyagok is helyet kaptak az oldalon, amelyekben csak azon funkciókra koncentrálunk, amelyek hallgatói jogosultsággal érhetőek el.

5. ábra: Segédanyagok a használt keretrendszerekhez

2.7. Hírlevél

Minden hónap elején az oktatói listára kiküldésre kerül egy Oktatásmódszertani hírlevél[8], amely tartalmazza az aktuális híreket, az elérhető képzéseket, illetve az újonnan publikált cikkek rövid tartalmi összefoglalóit.

Emellett az oktatással kapcsolatos podcastek ajánlóit, illetve a munkatársakkal készített interjúkat is olvashatnak az oktatók.

2.8. Pályázatok, támogatások

ELTE Oktatási és Képzési Tanácsa a 2018/2019-es tanévtől kezdődően minden évben pályázatot ír ki az egyetemi e-learning rendszerekben fejlesztett és megvalósult kurzusok oktatói számára. A beérkező pályázatokat az egyetemi e-learning szakértők közül felkért E-learning Pályázati Bizottság értékeli megadott szempontok alapján, amelyek előre ismertek a pályázók számára.

Az értékelőlapon a következő kategóriákba tartozó irányelvek találhatóak: kurzusáttekintés és bevezető; tanulási célok; mérés és értékelés; forrásanyagok; hallgatói interakció; kurzustechnológia; a

[7] https://www.elte.hu/elearning/segedanyagok

[8] Oktatásmódszertani hírlevél. Felelős szerkesztő: dr. Visnovitz Ferenc osztályvezető, Szerkesztő: dr. Virányi Anita

hallgatók támogatása; hozzáférhetőség. A kategóriákban összesen 33 irányelvet találunk, amelyek teljesítési fokától függően kapnak pontszámot az egyes kurzusok. Ilyen irányelvek például a következők:

- A kurzusfelület tartalmazza a technikai elvárások minimum szintjét, a hallgatói tudásminimumot, tisztázza a kötelező előfeltételeket.
- A kurzus tanulási céljai tagoltak, meghatározottak a modul / lecke szintjén.
- Az online kurzus során felhasznált minden forrásanyag megfelelő módon idézett.
- A kurzusban érzékelhető az oktatói jelenlét, az aktivitás és a hallgatókkal való kommunikáció.
- A kurzusfelület a vizuális és hanganyagokkal egyenértékű tartalmakat tartalmaz.

Az elbírálás során előnyben részesülnek azon pályázatok, amelyek elősegítik az ELTE szombathelyi képzéseinek integrációját, mintatantervben kötelezőként vagy kötelezően választhatóként szerepel a tárgy; a kurzus tartalmának egészét lefedik; akadálymentesítettek, nagyobb hallgatói létszámmal rendelkeznek, fejlesztői az egyetemi e-learning rendszerekkel kapcsolatos képzéseken, valamint oktatásmódszertani képzéseken való részvételt tanúsító oklevéllel rendelkeznek.

2.9. Konferenciák

A jó gyakorlatokat igyekszünk megismertetni oktatóinkkal konferenciák keretében is. Hagyományteremtő céllal, 2022. júniusában szerveztük meg először a Felsőoktatás felsőfokon konferenciát[9], amely lehetőséget adott arra, hogy az oktatók bemutathassák eredményeiket, megismerhessék egymás munkáit, tapasztalatokat szerezhessenek, illetve oszthassanak meg egymással.

A konferencia több szekciója is a modern pedagógia módszerekkel, innovációkkal, jó gyakorlatokkal kapcsolatos előadásoknak adott helyet.

3. Nyílt kurzusok

Az ELTE képzéseiben a Canvas rendszer két különálló példányát használjuk. A *canvas.elte.hu* címen elérhető rendszer a formális oktatás támogatására, míg *mooc.elte.hu* portál a nyitott kurzusok, belső képzések, egyetemek közti együttműködésben létrejött kurzusok lebonyolítására szolgál. Utóbbi portálon mind az oktatók, mind a kurzus résztvevők meghívása szabadon történhet (nincs Neptun kurzushoz kötve), sőt, akár a kurzusra történő önálló jelentkezés is beállítható.

Emiatt olyan segédletek kidolgozása is szükségessé vált, amely külsős résztvevők számára foglalják össze a rendszer használatát, a regisztráció módját, a felület alapbeállításainak módosítási lehetőségeit.

Az elmúlt évek során az ELTE számos nyílt kurzust indított[10]. Az online végezhető, 6 hetes nyílt kurzusokra bárki jelentkezhetett érdeklődésének megfelelően. Ezen kurzusok akár ezernél is több résztvevővel indultak, így mind technikailag, mind módszertanilag kihívás elé állították a kurzusok szerzőit és facilitátorait.

Néhány cím az elindított kurzusokból: Nyílt kurzusok tervezése; A pszichológia tudománya; Karrier 4.0 – az álláskeresés lépései; Gyógypedagógiáról mindenkinek; Túlélőkészlet kisgyermekes szülőknek; Felkészülés az egyetemi tanulmányokra.

[9] https://www.elte.hu/felsooktatas-felsofokon

[10] https://www.elte.hu/mooc

4. A Canvas LMS használatának tapasztalatai

A következőkben a Canvas LMS használatára vonatkozó statisztikákat mutatunk be.

4.1. Canvas kurzusok száma

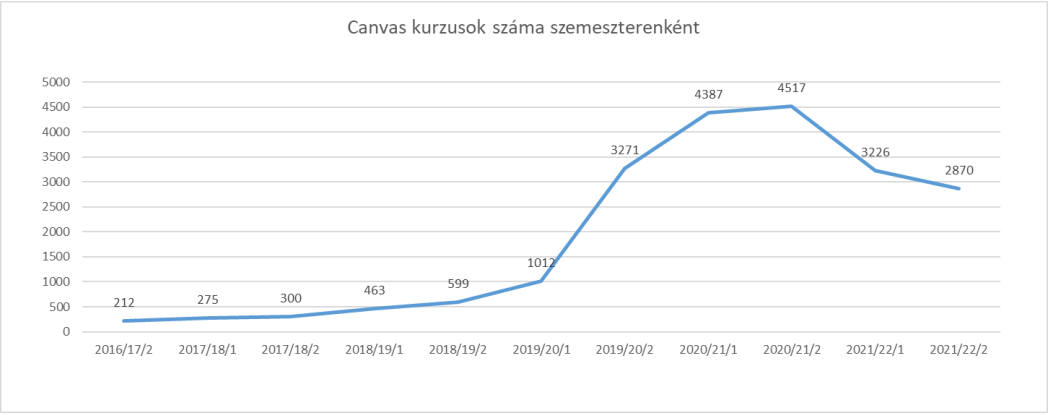
A Canvas rendszerben indított kurzusok száma dinamikusan nőtt a félévek során. Az első félév tapasztalatait korábbi cikkünkben ismertettük [4], a következőkben a 11 lezárt félévre vonatkozó összesített adatokat tesszük közzé.

Az 5. ábrán látható, hogy a koronavírus okozta pandémia miatti online oktatás bevezetésekor (2019/20/2) drasztikusan megnőtt a rendszerben indított kurzusok száma. Ekkor az előző félévhez képest 223%-os növekedés volt tapasztalható.

Az európai egyetemek 95%-a ebben az időszakban átállt a digitális oktatásra, az e-learninggel foglalkozó szervezeti egységek szerepe és feladata felértékelődött, sőt, egyes egyetemeken új szervezeti egységek jöttek létre annak érdekében, hogy az online oktatással kapcsolatos kihívásokat kezeljék [5].

Ebben az időszakban természetesen nem csak a Canvas rendszerben indított kurzusok száma nőtt meg az ELTE-n, az oktatók más, a szinkron, online oktatást lehetővé tevő megoldások (pl. Microsoft Teams) felé fordultak.

A pandémia nem csak a felsőoktatásban eszközölt változásokat, hanem a felsőoktatással kapcsolatos kutatásokban is, amely megnyilvánul a publikációs tér változásában, vagy a „szürke irodalom" felértékelődésében, amely gyorsabban reagált a különleges helyzetre. Az oktatók számára kiemelten fontossá vált például a jó gyakorlatok megosztása [6].



6. ábra: Canvas kurzusok száma szemeszterenként

Egyetemünkön a hibrid oktatás során (2020/21/1) tovább nőtt a kurzusok száma, amely elérte a 4517-es értéket. A jelenléti oktatás újbóli bevezetésekor (2021/22/1) a kurzusok számában ugyan visszaesés tapasztalható (-29%), de korántsem csökkent a kurzusok száma olyan arányban, mint amennyivel a pandémia miatt korábban megnőtt. Vagyis sok oktató továbbra is online kurzussal támogatja a jelenléti óráit.

Tanév	2016/17	2017/18		2018/19		2019/20		2020/21		2021/22	
Félév	2	1	2	1	2	1	2	1	2	1	2
Kurzusok	212	275	300	463	599	1012	3271	4387	4517	3226	2870
Változás (%)	0%	30%	9%	54%	29%	69%	223%	34%	3%	-29%	-11%

1. táblázat: Publikált státuszban lévő Canvas kurzusok számának változása a megelőző félévhez képest

4.2. LMS használat az egyetemi karok szerint

A Canvas LMS-ben, a 2016/17/2 - 2021/22/2 időszakban indított kurzusok számát vizsgálva a legtöbb kurzust a Pedagógiai és Pszichológiai Kar (PPK) indította (5520 kurzus), amelyet a Bölcsészettudományi Kar (BTK) követ 4957 kurzussal. Ezt követi az Informatikai Kar (IK, 3697 kurzus) és a Természettudományi Kar (TTK, 3132 kurzus).

Vannak olyan egyetemi intézetek, illetve karok, amelyek inkább a Moodle LMS használatát részesítik előnyben, ilyen például az Állam- és Jogtudományi Kar (ÁJTK), Bárczi Gusztáv Gyógypedagógiai Kar (BGGYK), Gazdaságtudományi Kar (GTK). A Társadalomtudományi Kar (TÁTK) pedig a CooSpace rendszert használja, azonban ez hamarosan kivezetésre kerül. Arra is van példa, hogy egy kar (Tanító- és Óvóképző Kar) vegyesen, közel azonos számban indít kurzusokat a Canvas/Moodle rendszerekben egy adott félévben.

Az utolsó lezárt szemesztert (2021/22/2) vizsgálva elmondható, hogy a Canvas LMS-ben már több kurzust indítottak az oktatók ELTE szinten, mint a Moodle keretrendszerben (2. táblázat).

Karok / Intézetek	Canvas kurzusok száma	Moodle kurzusok száma	Canvas kurzusok aránya
ÁJTK	132	704	16%
BDPK-SEK	4	47	8%
BGGYK	57	506	10%
BTK	628	394	61%
GTK	10	468	2%
IK	631	70	90%
PPK	811	105	89%
TÁTK	5	8	38%
TKK	10	7	59%
TÓK	148	158	48%
TTK	434	191	69%
Összesen	**2870**	**2658**	**52%**

2. táblázat: Canvas kurzusok aránya a karok/intézetek szerint a 2021/22/2 szemeszterben

4.3. Kurzusok nyelv szerint

A Canvas LMS-ben az a nyelv lesz a kurzushoz társítva, amely a tanrendi felterjesztésben szerepel. Ha pl. a kurzus angol nyelvűként van jelen a Neptun tanulmányi rendszerben, akkor a kurzus felülete angol nyelven fog megjelenni a hallgatóknak.

Elmondható, hogy nagy többségben (82%) magyar nyelvű kurzusokat indítottak az oktatók, amelyet az angol nyelvű kurzusok követnek (15%). A maradék 3%-on további nyelvek osztoznak (pl. német, dán, holland stb.)

4.4. Kurzustartalom vizsgálata

A 3. táblázatban láthatjuk félévenkénti bontásban, hogy az egyes kurzusok átlagosan mennyi feladatot, fórumot, kvízt stb. tartalmaztak, illetve feltüntettük azt is, hogy mi volt a maximális érték, illetve szórás. Az adatok is rámutatnak arra, hogy az egyes kurzusok esetén nagyon eltérő tevékenységekre helyezik a hangsúlyt az oktatók.

A **feladatok** kategóriába azon feladatok tartoznak, amelyeket akár papíron, online, vagy külső eszközben oldanak meg a hallgatók, és az arra kapott pontszámok az LMS rendszerben kerülnek adminisztrálásra. A feladatok akár gyakorló feladatok is lehetnek, amelyeknél a megszerzett pont nem számít bele a végső jegybe. A statisztikában látszik, hogy van olyan kurzus, amelyben 344 feladat került kiírásra. Ez természetesen nem azt jelenti, hogy egy adott hallgatónak ennyi feladatot kell megoldania, ezen kiemelkedő számok jellemzően összevont kurzusokhoz tartoznak, amelyekben az egyes csoportoknak esetenként külön-külön kerülnek kiírásra feladatok (röpzh, zh, gyakorló feladat stb.). Sőt a jelenlét vezetése is történhet feladatkiírásként, amelynél az oktatók adminisztrálhatják a jelenlétet, illetve hiányzásokat.

A **fórumok** lehetővé teszik, hogy a hallgatók és oktatók aszinkron módon kommunikálhassanak egymással. Jellemző, hogy az oktatók a kurzus egészével, a követelményekkel kapcsolatos fórumokat hoznak létre, illetve, hogy az egyes modulokhoz fórumok kapcsolódnak, amelyekben a modulhoz kapcsolódó kérdéseket lehet feltenni. A fórumok azonban feladatok kitűzésére is alkalmasak. A fórumbejegyzéseket akár értékelni is lehet, azokhoz pontszámot lehet társítani. Amikor szükség van arra, hogy a hallgatók láthassák egymás beadandóit (pl. az általuk által tartott kiselőadások diáit), a fórum jó megoldás lehet a feladatok begyűjtésére. A statisztikában látszik, hogy volt olyan kurzusunk, amelyben 1786 **fórumbejegyzés** született a félév során. Ezen kurzusban 83 hallgató és 5 oktató vett részt. A kurzus lehetővé tette a hallgatóknak, hogy bemutatkozzanak, reflexiókat, prereflexiókat osszanak meg egymással az egyes modulokkal kapcsolatban, illetve ötleteket osszanak meg egymással a tanultak alkalmazására.

Tanév		**2016 /17**	**2017 /18**		**2018 /19**		**2019 /20**		**2020 /21**		**2021 /22**	
Félév		2	1	2	1	2	1	2	1	2	1	2
Feladatok	Átlag	7	8	7	6	6	9	9	7	7	9	10
	Max.	91	76	55	55	45	295	299	329	337	344	246
	Szórás	15	15	10	7	8	14	14	15	14	14	16
Fórumok	Átlag	3	3	4	3	3	2	4	2	2	2	2
	Max.	29	30	159	96	95	99	84	68	55	130	68
	Szórás	5	5	14	8	8	5	6	5	4	6	5
Fórumbejegyzések	Átlag	7	10	11	7	5	5	11	5	4	6	4
	Max.	257	276	366	301	380	294	1357	1094	1076	1786	405
	Szórás	29	38	44	27	26	23	47	33	29	47	21
Kvízek	Átlag	2	3	1	1	1	1	3	3	3	4	5
	Max.	60	43	26	25	30	80	147	284	173	264	319
	Szórás	9	9	4	3	3	4	7	8	9	9	13

Kvíz-kérdések	Átlag	31	43	27	14	21	16	33	31	30	38	40
	Max.	895	975	600	700	615	1300	4779	2883	1652	1802	10629
	Szórás	110	126	88	48	64	72	119	98	84	91	216
Oldalak	Átlag	5	4	3	2	3	3	3	3	3	4	5
	Max.	168	46	46	38	76	125	154	175	212	214	215
	Szórás	15	10	7	5	8	9	9	10	10	15	15
Modulok	Átlag	5	4	4	3	4	4	4	3	3	4	4
	Max.	20	27	27	25	30	29	69	67	51	52	46
	Szórás	6	6	5	4	5	5	5	5	5	5	5

3. táblázat: Kurzustartalmakra vonatkozó statisztikai adatok félévenkénti bontásban

A **kvízek** létrehozása szintén népszerű az oktatók között, hiszen segítik a zh-k, vizsgák lebonyolítását, automatikus kiértékelését, de önellenőrzés céljából is lehet kvízeket készíteni. A kvízek egy alcsoportját jelentik a **felmérések**, amelyek segítségével visszajelzést kérhetnek az oktatók a hallgatóktól. Ezesetben a kérdéseknél nincs helyes válasz meghatározva, a hallgatók azért kapnak pontot, hogy válaszoltak a kérdésekre. A statisztikában kiemelkedik egy olyan kurzusunk, amely több, mint 10.000 kvízkérdést tartalmaz. Ezen kurzus éppen szerves kémiával foglalkozik, azonban a nyelvtanítással foglalkozó kurzusok esetén is jellemző a kvízkérdések átlagot jóval meghaladó száma.

Az **oldalak** fontos építőkövei a keretrendszerben elhelyezett tananyagoknak. Az oldalakat a Canvas rendszer beépített szövegszerkesztő felületén lehet tartalommal feltölteni, illetve formázni. Természetesen lehetőség van képek, multimédiás elemek beágyazására, táblázatok, képletek elhelyezésére. Olyan oldalak is létrehozhatóak, amelyeket közösen szerkeszthetnek az oktatók és hallgatók. Nem minden oktató dolgozza ki oldal szintjén a tananyagot. Alternatíva lehet egy PDF jegyzet (vagy annak fejezetenként feldarabolt változatának) fájlként való elhelyezése a modulban. Találhatunk azonban olyan kurzust is, ahol az oktató rengeteg idő befektetéssel a rendszeren belül hozta létre a tananyagot, amely több, mint 210 oldalból áll.

A kurzus összetevőt **modulokba** szervezhetjük. A modulokhoz társíthatóak követelmények. Például beállítható, hogy egy modult akkor teljesített egy hallgató, ha minden oldalt megtekintett, minden feladatot beadott, és a záró kvízen legalább 50%-os eredményt ért el. A moduloknak előfeltételt is beállíthatunk, vagyis lehet, hogy egy modul tartalma csak akkor lesz elérhető a hallgatók számára, ha egy előkövetelményként beállított modult sikerrel teljesítettek. Jellemzően azon kurzusokban találkozunk kiemelkedően nagy modul számokkal, amelyeket úgy hoztak létre, hogy mikrotanulásra alkalmasak legyenek. Ebben az esetben egy-egy modul kb. 15 percnyi időráfordítással elvégezhető lehet.

5. Összefoglalás

Az ELTE Oktatási Igazgatóság Oktatásfejlesztési és Tehetséggondozási Osztálya a Canvas LMS bevezetése többféle módon támogatja az oktatókat abban, hogy korszerű, hatékony, haladó pedagógiai és értékelési módszereket használó e-learning kurzusokat hozhassanak létre, illetve továbbképezhessék magukat az LMS rendszerek használata, valamint más oktatásmódszertani témák területén.

A koronavírus miatt bevezetett online oktatásnak köszönhetően az LMS rendszerekben ugrásszerűen megnőtt az e-learning kurzusok száma, amely a jelenléti oktatás bevezetése után kisebb mértékben csökkent.

A kurzusok tartalmát vizsgálva elmondhatjuk, hogy az oktatók igen eltérő tevékenységekre helyezik a hangsúlyt, amely kimutatható a kurzusokban aktívan használt fórumok, kvízek, feladatok, oldalak és modulok számában.

Jelenleg elég nagy eltérés mutatkozik atekintetben, hogy melyik kar/intézet melyik LMS megoldást preferálja a képzések során, azonban az utolsó lezárt félév adatai alapján elmondható, hogy a Canvas LMS népszerűbb választás az oktatók körében, mint a Moodle LMS.

Irodalom

1. Imed Bouchrika: *51 LMS Statistics: 2021/2022 Data, Trends & Predictions*. https://research.com/education/lms-statistics (utoljára megtekintve: 2022.10.20.)
2. ELTE Oktatási Igazgatóság Oktatásfejlesztési és Tehetséggondozási Osztály: *Miben segít az e-learning szakértő?* https://www.elte.hu/elearning/konzultaciok (utoljára megtekintve: 2022.10.20.)
3. Abonyi-Tóth Andor: *Haladó kurzusszervezési és értékelési lehetőségek a Canvas LMS rendszerben* In: InfoDidact 2017 (Budapest: Webdidaktika Alapítvány, 2017.) – ISBN 978-615-80608-1-3 (online), http://konferenciak.inf.elte.hu/infodidact/InfoDidact17/Manuscripts/ATA.pdf (utoljára megtekintve: 2022.10.31.)
4. Abonyi-Tóth, Andor, Tóth-Mózer, Szilvia*: A Canvas LMS használatának tapasztalatai az ELTE képzéseiben* In: Agria Média 2017. Eger, Líceum Kiadó. pp. 49-57. https://doi.org/10.17048/AM.2018.49
5. Dringó-Horváth, Ida és T.Nagy, Judit és Weber, Andrea (2021) *Felsőoktatásban oktatók digitális kompetenciáinak fejlesztési lehetőségei.* EDUCATIO, 30 (3). pp. 496-507. ISSN 1216-3384 (print); 1419-8827 (online) https://doi.org/10.1556/2063.30.2021.3.9
6. Kopp, Erika ; Saád, Judit: *A pandémia első hulláma a felsőoktatás-kutatások tükrében - szakirodalmi áttekintés NEVELÉSTUDOMÁNY: OKTATÁS KUTATÁS INNOVÁCIÓ : 3 pp. 7-22. , 16 p. (2021) https://doi.org/10.21549/NTNY.34.2021.3.1*

Új kihívások pandémia után

Bakonyi Viktória[1], Illés Zoltán[2], Pšenáková Ildikó[3]

{[1]hbv, [2]illes}@inf.elte.hu, [3]ildiko.psenakova@gmail.com
ELTE IK, Trnava University in Trnava

Absztrakt. Az online oktatás idején új eszközökkel, módszerekkel ismerkedtek meg mind az oktatók, mind pedig a hallgatók. Mindannyian örültünk, amikor végre visszatérhettünk "az iskolapadokba", de hamar rá kellett jönnünk, már semmi se lesz ugyanolyan, mint előtte. Megkérdeztük a hallgatóink véleményét, mit tartottak jónak, folytatandónak a vészhelyzeti oktatás idejéből, milyen igényeik vannak az elkövetkezendőkben. Ezek a visszajelzések mindenképpen segíthetik az oktatásfejlesztéssel kapcsolatos terveinket.

Kulcsszavak: oktatás, interaktivitás, kérdőív, hallgatói igények

1. Bevezető

Az ókori Görögországban a diákok a professzorok körül ültek, alaposan megvitattak egy-egy felmerülő kérdést. A történelem későbbi szakaszában az egyetemi oktatás lényege gyakran a résztvevők közötti vita volt. Napjainkban a legjobb angol egyetemeken tutor rendszert alkalmaznak, amelyben a személyes kapcsolatnak és a vitának nagy szerepe van. De mi történik a standard oktatásban? A 21. században a mindennapi életben is radikálisan gyors változás következett be az internet okozta információs forradalomnak köszönhetően.

Ez mindenre, így az oktatásra is hatással volt, ezért a professzorok olyan váratlan problémákkal szembesültek, amelyek korábban soha nem fordultak elő [1][2][3]. A régi, klasszikus módszerek már nem működtek úgy, mint korábban. A diákok kevésbé aktívak, és néha úgy tűnik, hogy teljesen unatkoznak. Ezt a nehéz oktatási helyzetet még nehezebbé tette a Covid-19 és a távoktatás, amely mindent a feje tetejére állított. Két év után visszatértünk az iskolába, de meg kellett értenünk, hogy nem lehet ott folytatni a munkát, ahol a vészhelyzet előtt abbahagytuk.

2. Covid előtt, személyes oktatás

A mai diákok digitális bennszülöttek, akiknek agyi aktivitása a napi több órás böngészőhasználat miatt megváltozott [4]. Hozzászoktak a színes, párhuzamos multimédia folyamhoz, amelyet az állandóan körülöttük levő eszközök szolgáltatnak számukra - ezért egy előadás, ahol egyetlen információ forrás az előadó, már nem annyira érdekes számukra. Ráadásul tudják, hogyan lehet gyorsan válaszokat kapni a netről, ezért gyakran mondják, hogy "időpocsékolás a tanulás, mindent megtudok egy kattintással".

Ellentétben azzal, amit elsőre gondolnánk, ez a párhuzamos információáramlás és a köztük levő gyors váltások, inkább ártanak a tanulás hatékonyságának. A jelenséget hiperfigyelemnek nevezzük [5], Kathrine Hayes nyomán. A több eszközt párhuzamosan használó személynek váltogatnia kell az ingerek között, és ezt csak az emberek 2%-a tudja megfelelően kezelni - a többieknél csökken a hatékonyság [6].

E. Dale, az amerikai pedagógus egy jól ismert grafikont készített, amely a különböző passzív és aktív tanulási környezetekben történő tanulás hatékonyságát mutatja. Bebizonyította, hogy az aktivitás növeli a tanulás hatékonyságát, lásd az 1. ábrát.

Az elmúlt évtizedekben a pedagógusok megpróbálták ezt az eredményt felhasználni, és újabb és újabb módszerek jelentek meg, amelyek a diákok aktivizálására összpontosítanak - például új interaktív tananyagok kifejlesztése a hagyományos dokumentációk helyett, csoportmunka, fordított osztályterem vagy Classroom Response Systems (CRS) [7] [8] [9] [10] [11] használata a tanórákon stb.

A gyakorlatban azt tapasztaltuk, hogy a klasszikus módszerek, a professzor és a diákok közötti beszélgetés nem működött tovább. A diákok nem akarták felemelni a kezüket, hogy kérdezzenek - esetleg a többiek előtt szégyellték a kérdéseiket. Eközben azt is felismertük, hogy a megszokott kommunikációs módjuk a csevegés [12].

1. ábra: E. Dale: (https://bit.ly/3zBNm2a)

1. ábra: E. Dale: (https://bit.ly/3zBNm2a)

Néhány évvel ezelőtt úgy döntöttünk, hogy a tanítási gyakorlatunkban CRS-t (Classroom Response System) használunk, hogy növeljük az előadások interaktivitását és az oktatás szolgálatába állítsunk még egy információ forrást, ne csak az előadó mondanivalója legyen a fókuszban, hanem a párbeszéd is nagyobb szerepet kapjon. Első lépésként néhány kész szoftvert vizsgáltunk meg. Ezek többnyire jól megtervezett, professzionális megoldások, de az ingyenes változatok használatánál korlátokkal, nehézségekkel találkoztunk. [13][14][15]. Végül is úgy döntöttünk, hogy egy saját rendszert valósítunk meg.

Az alkalmazásunk (E-Lection) egy BYOD (Bring Your Own Device), kétirányú, valós idejű webes alkalmazás lett, amely az egyetemi szabványos hitelesítési eljáráson alapul (nincsenek vicces azonosító nevek, nincsenek kétértelmű nyelvi kérdések stb., amivel sok esetben meg kell barátkozni az oktatónak).

Az E-Lection C# nyelven íródott, Web Forms sablon használatával, a valós idejű funkciókhoz pedig SignalR-t használtunk. Kétirányúnak nevezhető, mert miközben a tanár elküldhette a kvízkérdéseket az összes csatlakozott diák eszközére, amelyekre a diákok válaszolhattak az eszközeiken, de a másik irány is működik, tehát ők is kérdezhetik a tanárt vagy jelezhetik, hogy nem értenek valamit. (Lásd a 2. ábrát. hogyan néz ki az E-Lection.)

2. ábra: Az E-Lection rendszer

Készítettünk egy felmérést, még induláskor, hogy megkérdezzük a diákok véleményét erről a lehetőségről. Van-e igényük arra, hogy bevezetésre kerüljön egy ilyen eszköz.

A kérdőív:
Készítésének ideje: 2017 szeptember
Kitöltötte: 214 magyar diák
Korosztály: 20-25 éves
Körülményeik: Vidéki és nagyobb városokból is, vegyesen.
A kitöltés nem volt kötelező.
Anonimitás biztosítása: Google Forms
Elérhető: https://forms.gle/gCP7ySVnfSePY6rg8

Az egyik kérdés a következő volt:

Hasznosnak tartanád-e, ha az előadások során kétirányú kommunikáció lenne az előadó és a hallgatóság között? (1-5. osztályzat).

Osztályzat	Százalék
1	6.3%
2	10.7%
3	25.6%
4	31.7%
5	25.6%

1. táblázat: A kommunikáció hasznosságáról hallgatói vélemény 2017

A koncepció működött, a hozzáadott interaktivitás hatékonyságot eredményezett, és mérésekkel bebizonyítottuk, hogy a hallgatói eredmények jobbak lettek [20].

3. Covid-19 időszak – online oktatás

Kurzusainkat próbaidőszak vagy vészhelyzeti tervezés nélkül kellett egy hét alatt átvinnünk a virtuális térbe a Covid-19 megjelenésekor. Sokakat foglalkoztatott, hogy mi a helyzet az oktatással, amely általában személyes kapcsolatokon alapul [16][17][18]?

Karunkon a dékán döntése az volt, hogy az LCMS rendszerünk, a Canvas használata mellett a valós idejű virtuális tantermi rendszer (VCS), a Microsoft Teams szinkron online tanítási módszerrel és videofelvétellel folytatjuk a menetrendet a megkezdett órarend szerint. A döntés alapja az volt, hogy meg akartuk tartani a személyes kapcsolatot a hallgatóinkkal, amennyire csak a technika lehetővé teszi. Mindannyian tudjuk, hogy a teljesen aszinkron tanítás, mint a MOOC kurzusok, magas lemorzsolódással működnek.

Az első sürgősségi félév végén (és az őszi félév kezdetén) egy felmérés segítségével kikértük a hallgatók véleményét az online kurzusokról [19][20][21][22][23]. Tudni szerettük volna, min kell finomhangolnunk a hatékonyabb oktatás érdekében.

A kérdőív:

Készítésének ideje: 2020 május

Kitöltötte: 473 magyar diák

Korosztály: 20-25 éves

Körülményeik: Vidéki és nagyobb városokból is, vegyesen.

A kitöltés nem volt kötelező.

Anonimitás biztosítása: Google Forms

Elérhető: https://forms.gle/BH16yj61Dixbu2AVA

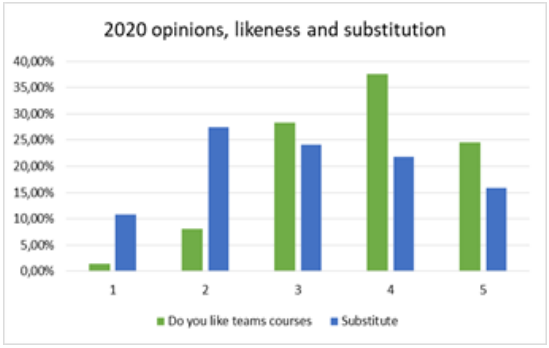
A következőkben a kérdőív következő 3 kérdésre összpontosítunk:

Szereti az online Teams-es órákat? (1. osztályzat (egyáltalán nem) -5. osztályzat (nagyon))

Mit gondol, egy Teams online óra helyettesíthet egy klasszikus élő órát, ahol Ön személyesen ott van? (1. fokozat (egyáltalán nem) -5. fokozat (teljesen))

*Mit nem szeret bennük? (*szabad szöveg*)*

A legtöbbjüknek tetszettek az online órák, de kevésbé volt pozitív arra a kérdésre a válasz, hogy mennyire lehet teljes egészében helyettesíteni a klasszikus órákat online órákkal lásd a 3. ábrát.



3. ábra: Tetszés és helyettesíthetőség, 2020

Természetesen megkérdeztük azt is, hogy mit nem szeretnek az online órákban. Egy-egy ilyen probléma feltárása segíthet elkerülni az olyan módszereket, amelyeket a diákok nem ked-

velnek. Mivel szabad szöveg volt, némi adatfeldolgozást kellett végeznünk. (Az összes eredményt elemeztük és publikáltuk is [19][20].

Az eredmény többek között azt bizonyította, hogy *saját meglátásuk szerint is, több interaktivitásra, több személyességre van szükségük*, ami a virtuális tantermek használatával elveszni látszott. Ezeket a tulajdonságokat tartjuk fontosnak a hatékony oktatásban mi is, az oktatói oldalról. Bár hiányolták az interaktivitást, és az online kurzusokat kevésbé személyesnek tartották, az egész online tanítási időszakban azt tapasztaltuk, hogy a diákok kevésbé aktívak, nem szeretik bekapcsolni a mikrofonjukat, a webkamerájukat, vagy akár megosztani a képernyőjüket. Leginkább a chat üzenetek vagy a like-ok használatát preferálták megfigyeléseink szerint.

4. Covid után – hogyan folytassuk?

Egy erős Covid-hullám után, 2022. március 1-jétől folytattuk a személyes tanítást, bár továbbra is streameltük az órákat, és az esetleges betegségek miatt felvételeket is készítettünk. Az oktatók észrevették, hogy valami megváltozott - ismét [24][25][26][27]. Igen, mind tudjuk ezt, egy dolog állandó, a változás – nem lehet egyszerűen a régi, Covid előtt bevált oktatási formákkal, visszatérni a régi módszerekhez.

Sok diák inkább otthon maradt kényelmesen, és felvételeket és streameket nézett, lemondott a természetes kommunikációról. Sokan közülük egyáltalán nem akartak bejönni az egyetemre - a felső határig hiányoztak, és az oktatói unszolások ellenére sem akarnak aktívan részt venni a vitákban. Ahogy észrevettük, az interaktivitásra való hajlandóság alacsony maradt. Ismét készítettünk egy anonim felmérést, hogy jobban megértsük az új helyzetet és ne csak a szubjektív megfigyeléseinkre támaszkodjunk.

A kérdőív:

Készítésének ideje: 2022 március-április, közvetlenül a távoktatási időszak után

Kitöltötte: 169 magyar diák

Korosztály: 20-25 éves

Körülményeik: Vidéki és nagyobb városokból is, vegyesen.

A kitöltés nem volt kötelező.

Anonimitás biztosítása: Google Forms

Elérhető: https://forms.gle/J5syCkqYXFYmuY148

Ebből a kérdőívből most négy olyan kérdésre szeretnénk összpontosítani, amelyekkel megállapíthatjuk, hogyan változtak a tanulók igényei.

1. *Milyen tanítási formát választana?* (legördülő lista) A lehetséges válaszok a következők voltak: hagyományos, részben online, online előadás, hibrid, teljesen online.
2. *Az előadás vagy a gyakorlat során hatékonyabb az online óra?* (legördülő lista) A lehetséges válaszok a következők voltak: Egyik sem, Előadás, Gyakorlat, Mindkettő. Lásd 4. ábra)
3. *Az előadás jellegű órák közül melyiket tartja a leghatékonyabbnak?* Legördülő lista) Választható válaszok: előre felvett anyag, élő előadás ppt-vel, demókkal, élő előadás csak ppt-vel, élő előadás demókkal.
4. *Hasznosnak tartja-e az előadó és a hallgatók közötti párbeszédet az előadás alatt?* (1-5-re vonatkozóan)

Lássuk sorra a kérdéseket:

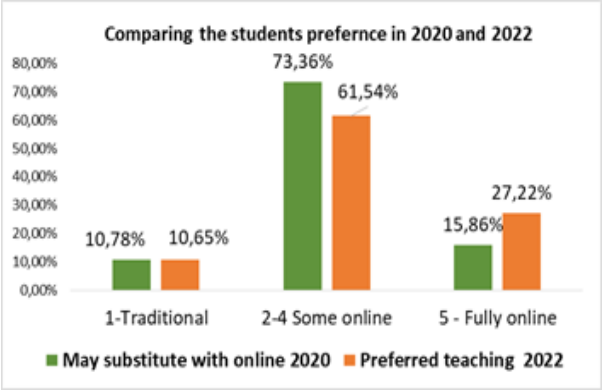
4.1. Milyen tanítási formát választana?

Két évnyi vészhelyzetben történő tanulás után, teljes online és hibrid oktatási modell használatáról is van a hallgatóknak tapasztalaton alapuló véleményük. (A hibrid mód azt jelentette, hogy a diákok fele az osztályban van, a másik fele pedig online, majd a következő héten cserélődtek a szerepek).

Adatokat gyűjtöttünk arról, hogy mit gondolnak a hagyományos oktatás esetleges helyettesítéséről az online oktatással. Míg a klasszikus egyetemi stílushoz ragaszkodó hallgatók aránya változatlan maradt ~10%, addig a teljesen online oktatást preferálók aránya kétszer akkora lett, mint korábban (15%=>27%). A változások összehasonlítását lásd a 4. ábrán. Az elmúlt két év változásainak győztese a *teljesen online mód*. Miután kipróbálták, tetszik nekik, ragaszkodnak hozzá. (Nem biztos, hogy mindegyik képzés azonos eredményeket mutatna. A mi diákjaink a jövő programozó matematikusai, így az informatika iránti affinitásuk magasabb, mint általában a diákoké). Sok diák azonban közülük is a vegyes stílust részesíti előnyben lásd a 2. táblázatot

Tanulási mód	Százalék
Hagyományos	10.65%
Valamennyi online	16.57%
Előadás online	36.69%
Hibrid	8.28%
Teljesen online	27.22%

2. táblázat Preferált tanulási mód 2022

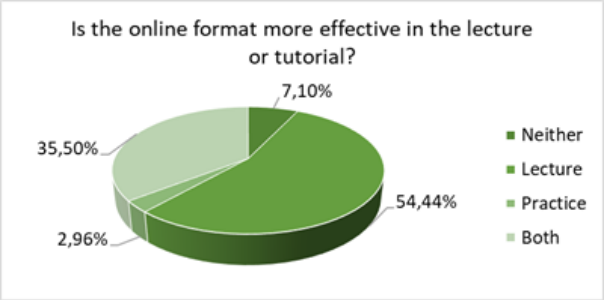


4. ábra: Preferált tanulási mód 2020-ban és 2022-ben

4.2. Az előadás vagy a gyakorlat során hatékonyabb-e az online óra?

Az online oktatás során természetesen mind az előadások, mind a gyakorlatok stílusa sokat változott a technikai környezet és a professzor által használt új módszerek függvényében. Tudtuk, hogy néhányan ragaszkodnak a klasszikus, többen inkább a teljesen online órához, de mi a diákok véleménye az online módszerekről az előadások, illetve a gyakorlatok alatt, lásd 5. ábra?

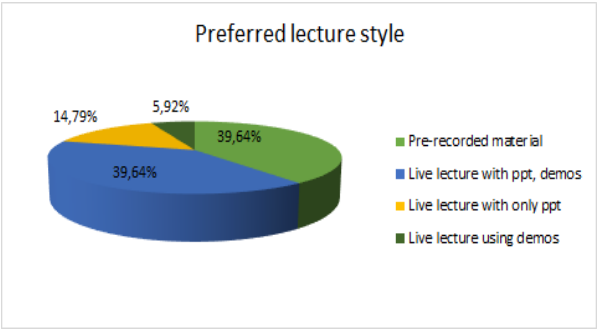
Az eredmény azt mutatja, hogy több mint 54%-uk szerint az online előadások hatékonyabbak, de a gyakorlatok esetében inkább a személyes tanítást kedvelik.



5. ábra: Előadás vagy gyakorlat

4.3. Az előadás jellegű órák közül melyiket tartja a leghatékonyabbnak?

Kíváncsiak voltunk, hogy a hallgatók milyen előadásmódot preferálnak, miután a Covid-19 időszak alatt párhuzamos élő előadásokat és felvett változatokat is hallgattak. A 6. ábra mutatja az eredményt, amely igencsak érdekes.

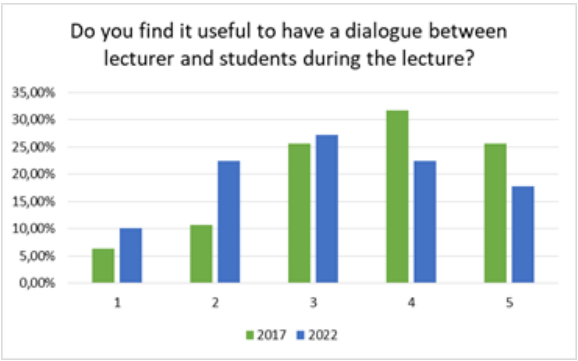


6. ábra: Preferált előadás

A hallgatók közel 40%-a az előre felvett videókat részesíti előnyben, ahol a valós idejű interakció, párbeszéd a professzorokkal egyáltalán nem lehetséges! A maradék 60% reményt ad arra, hogy valamilyen módon bevonhatók az előadásokba!

4.4. Hasznosnak tartja-e az előadó és a hallgatók közötti párbeszédet az előadás alatt?

Az igazat megvallva, igazából mindegy, hogy online vagy személyesen vagyunk-e ott, mindkét esetben interaktívabbá tehetjük az előadásokat akár hagyományos módon beszéddel érdekes kérdésfeltevésekkel vagy olyan technikai megoldásokkal, mint a CRS (pl. E-Lection). Az igazi kérdés az, hogy mit gondolnak az interakcióról, hajlandóak-e részt venni a vitákban vagy sem. A kérdéshez tartozó 5-ös osztályzat azt jelenti, hogy a diákok úgy érzik szükségük van az interakcióra.



7. ábra: Párbeszéd iránti igény 2017-ben és 2022-ben

Ahhoz, hogy összehasonlíthassuk a professzorokkal való kommunikációra való hajlandóságot és igényt az előadások alatt a vészhelyzet előtt és után, figyeljük meg a 7. ábrát. Jól látható, hogy az *interaktivitás* preferenciája csökkent az online időszak után. Visszautalva kiindulásukra, E. Dale tapasztalati kúpjára (lásd a 2. ábrát), az emlékezés hossza és a tudás mélysége összefügg a hallgatók aktivitásával.

Érték	**Hasznosnak tartja-e az előadó és a hallgatók közötti párbeszédet az előadás alatt 2022**
1	10.06%
2	22.49%
3	27.22%
4	22.49%
5	17.75%

3. táblázat Párbeszéd iránti igény 2022

Célunk, hogy visszanyerjük a tanulók aktív figyelmét. Szeretnénk őket ismét jobban bevonni a tanítási folyamatba. Ezért gondolkodunk módszereink és eszközeink finomításán, hogyan tudnánk a megváltozott tanulói szokások mellett is javítani az oktatás minőségét.

5. Összefoglalás

Régóta hiszünk az interaktivitásban, az elmélyült figyelem fontosságában és az aktív tanulási módszerekben. A Covid-19 előtt a CRS, az általunk fejlesztett E-Lection hatékony volt a klasszikus előadótermi előadások interaktívabbá tételében. Bebizonyítottuk, hogy használatával a hallgatók eredményei jobbak lettek. Ráadásul élvezték azt is, hogy egy egyedi megoldás készült a karon értük, az órák jobbítására.

Vészhelyzetben a személyes találkozók nem voltak lehetségesek, ezért az online virtuális osztályainkban MS Teams, az E-Lection és a Canvas alkalmazást használtuk. A különböző típusú alkalmazásokkal és módszerekkel tett kísérleteink ellenére azt vettük észre, hogy a diákok interaktivitása csökkent, amit a kérdőívünk is megerősített.

Most, miután visszatértünk a személyes tanításhoz, először azt reméltük, hogy mindent változatlanul folytathatunk attól a ponttól, ahol a vészhelyzet előtt befejeztünk, legfeljebb néhány elemet kell beemelnünk a gyakorlatunkba. Sajnos meg kellett értenünk, hogy már semmi sem ugyanaz, mint annak előtte. A hallgatók már nem akarnak visszatérni a klasszikus egyetemi oktatáshoz, az újonnan tanult és preferált eszközöket és lehetőségeket szeretnék továbbra is

használni. Sikeresebb vegyes tanítási módszereken kell dolgoznunk, az eszközök új keverékét kell kialakítani, hogy visszahozzuk a diákok aktivitását. A módszertani megújulást nem lehet elkerülni.

Az igazat megvallva, még nem minden kapott adatot elemeztünk ki. A munka befejezése után azt tervezzük, hogy alapjaiban újraírjuk az E-Lection rendszerünket – beemelve az újonnan megjelent új technológiai lehetőségeket és nyitva az új módszertani megoldások felé - hogy az új kihívásoknak is meg tudjunk felelni.

Köszönetnyilvánítás

A tanulmány a KEGA 012TTU-4/2021: „Integrácia využívania dištančných výučbových procesov a tvorby elektronických učebných materiálov do edukácie budúcich pedagógov.” (A távoktatási folyamatok alkalmazásának és az elektronikus tananyagok készítésének integrálása a leendő pedagógusok oktatásába) című projekt keretében készült.

Irodalom

1. Zitny, R. et al.: Education Using Mobile Technologies, ICETA 2016.11.24-25. Stary Smokovec IEEE, pp 115--120, ISBN:9781509046997
2. Z. Illés, Zoltán; H, Bakonyi Viktória; Jnr Z. Illés, “Supporting dynamic, bi-directional presentation management in real-time” In: Emil, Vatai (editor.) 11th Joint Conference on Mathematics and Computer Science, CEUR-WS.org, (2016) , 6 p.
3. H. Bakonyi, V., Illés Z.: Real-Time Tool Integration for Lectures, 15th IEEE International Conference on Emerging eLearning Technologies and Applications: ICETA 2017. Starý Smokovec, Slovakia, 2017.10.26-2017.10.27. Denver: IEEE Computer Society Press, 2017. pp. 31-36. ISBN:978-1-5386-3294-9
4. Marc Prensky: From On the Horizon, MCB University Press, Vol. 9 No. 5, October 2001, https://bit.ly/2YeKG7U
5. Hayes, N. Katherine, "Hyper and Deep Attention: The Generational Divide in Cognitive Moods. Profession (2007) 187-199pp
6. Travis Bradberry: Multitasking damaging your brain and career, new studies suggest https://www.forbes.com/sites/travisbradberry/2014/10/08/multitasking-damages-your-brain-andcareer-new-studies-suggest/2/#6088a80642ef
7. Jamila Shaaruddin, Maslawati Mohamad: Identifying the Effectiveness of Active Learning Strategies and Benefits in Curriculum and Pedagogy Course for Undergraduate TESL Students, Creative Education > Vol.8 No.14, November 2017 https://www.scirp.org/journal/PaperInformation.aspx?PaperID=80647
8. Opre, D. et al. (2022) ‘Supporting students’ active learning with a computer based tool’, Active Learning in Higher Education. doi: 10.1177/14697874221100465.
9. J. L. Brown: Quick, click: Student response systems evolve in higher ed, New student response systems offer increased versatility. University Business, November 2016 http://bit.ly/2fnJMRw
10. H. Dangel, C. Wang.: Student response systems in higher education: Moving beyond linear teaching and surface learning. Journal of Educational Technology Development and Exchange, 1(1), pp. 93-104. (2008) http://www.sicet.org/journals/jetde/jetde08/paper08.pdf
11. Mader S., Bry F. (2019) Audience Response Systems Reimagined. In: Herzog M., Kubincová Z., Han P., Temperini M. (eds) Advances in Web-Based Learning – ICWL 2019. ICWL 2019. Lecture Notes in Computer Science, vol 11841. Springer, Cham
12. Raymond Li: Communication preference and the effectiveness of clickers in an Asian university economics course, April 2020 Heliyon 6(4):e03847, DOI: 10.1016/j.heliyon.2020.e03847
13. Bakonyi, Viktoria; Illes, Zoltan;Verma Chaman: Key element in online education to activate students with real-time tools In: Institute of Electrical and Electronics Engineers 2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM) Conference: Dubai, United Emirates 2021.01.19. - 2021.01.21. Red Hook (NY): Curran Associates, pp 326-331 (2021)

14. Bakonyi, Viktoria; Illes, Zoltan; Verma, Chaman Towards the Real-Time analysis of Talks In: Ashok, K Chauhan; Gurinder, Singh International Conference on Computation, Automation and Knowledge Management Dubai, United Arab Emirates: Amity University, (2020) pp. 322-327. , 6 p.
15. Bakonyi, Viktoria;Illes, Zoltan;Verma Chaman: Analyzing the Students' Attitude Towards a Real-Time Classroom Response System In: Institute of Electrical and Electronics Engineers 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, (2020) pp 69-73 (2020), ISBN: 9781728140971
16. Kővári E., Bak G. (2021) University Students' Online Social Presence and Digital Competencies in the COVID-19 Virus Situation. In: Agrati L.S. et al. (eds) Bridges and Mediation in Higher Distance Education. HELMeTO 2020. Communications in Computer and Information Science, vol 1344. Springer, Cham. https://doi.org/10.1007/978-3-030-67435-9_13
17. Martin, F., Parker, M., Deale, D. (2012). Examining Interactivity in Synchronous Virtual Classrooms. International Review of Research in Open and Distributed Learning, 13 (3), 227–261. https://doi.org/10.19173/irrodl.v13i3.1174
18. Matt Bower: Virtual classroom pedagogy, Conference: Proceedings of the 39th SIGCSE
19. Technical Symposium on Computer Science Education, SIGCSE 2006, Houston, Texas, USA, March 3-5, 2006, DOI: 10.1145/1124706.1121390
20. Bakonyi, Viktória; Illés, Zoltán Real-time and digital solutions in education during emergency situation in Hungary In: Abonyi-Tóth, Andor; Stoffa, Veronika; Zsakó, László (szerk.) New Methods and Technologies in Education, Research and Practice: Proceedings of XXXIII. DidMatTech 2020 Conference Budapest, Magyarország: ELTE Informatikai Kar (2020) 507 p. pp. 231-240. , 10 p.
21. Bakonyi, Viktória; Illés, Zoltán: Real-time online courses during emergency situation in Hungary, 2020, ICETA 18th International Conference on Emerging eLearning Technologies and Applications November 12 Stary Smokovec, Slovakia, 2020
22. Bakonyi, V., Illés Z. REAL-TIME AND DIGITAL SOLUTIONS IN EDUCATION DURING EMERGENCY SITUATION IN HUNGARY, In: Abonyi-Tóth, Andor; Stoffa, Veronika; Zsakó, László (szerk.) New Methods and Technologies in Education, Research and Practice: Proceedings of XXXIII. DidMatTech 2020 Conference, Budapest, Hungary: ELTE Informatikai Kar (2020) 507 p. pp. 231-240. , 10 p.
23. Bakonyi, Viktória; Illés, Zoltán; Szabó, Tibor: Real-Time Interaction Tools in Virtual Classroom Systems, In: Singh, Pradeep Kumar; Singh, Yashwant; Chhabra, Jitender Kumar; Illés, Zoltán; Verma, Chaman (szerk.) Recent Innovations in Computing: Proceedings of ICRIC 2021, Volume 2 Singapore, Singapore: Springer Singapore (2022) pp. 625-636. Paper: Chapter 47, 12 p.
24. Viktoria, Bakonyi; Illés, Zoltán; Chaman, Verma: Real-time Education in Emergency Situation In: Institute of Electrical and, Electronics Engineers 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT) Piscataway (NJ), USA : IEEE (2021) pp. 1-6. , 6 p.
25. Mats Benner; Jonathan Grant; Mary O'Kane: The University Mission Before, During and After COVID, In book: Crisis Response in Higher Education, 2022, DOI: 10.1007/978-3-030-97837-2_1
26. Lauren Bialystok: Education after COVID, 2022, DOI: 10.7202/1088373ar https://www.researchgate.net/publication/360986155_Education_after_COVID
27. Manuel Mazzara; Petr Zhdanov; Mohammad Reza Bahrami; Ruslan Pletnev: Education After COVID-19, January 2022, DOI: 10.1007/978-981-16-9101-0_14, In book: Smart and Sustainable Technology for Resilient Cities and Communities https://www.researchgate.net/publication/358874947_Education_After_COVID-19
28. Oscar Koopman; Karen Joy; Koopman Karen: The Rise of the University without Classrooms after COVID-19, January 2021, In book: Re-thinking the Humanities Curriculum in the Time of COVID-19, Publisher: CSSALLPublishers (Pty) Ltd https://www.researchgate.net/publication/348234676_The_Rise_of_the_University_without_Classrooms_after_COVID-1

Négyzetrácsalapú akciójátékok programozása a Scratch-ben

Bernát Péter

bernatp@inf.elte.hu
ELTE IK

Absztrakt. A programozástanítás számos nagy témakörének egyike a játékfejlesztés. A bevezető programozástanítás során a számítógépes játékok sokféle műfaja közül gyakran esik a választás az akciójátékokra. Egy korábbi cikkemben három játéktípuson keresztül ismertettem az akciójátékok legfontosabb kellékeit és megvalósításukat a Scratch-ben. Jelen cikkemben az előzőre építve a négyzetrácsalapú akciójátékok – amelyekben a szereplők egy négyzetrács mezőin mozognak – megvalósításának a sajátosságait járom körül és értékelem továbbra is a Scratch programozási környezetben.

Kulcsszavak: Scratch, programozástanítás, játékfejlesztés, akciójátékok, négyzetrácsalapú játék

1. Bevezetés

Informatikatanárként a bevezető programozástanítás során számos olyan programozási terület közül választhatunk, amelyekkel a programozás alapfogalmai szemléletesen és motiváltan vezethetők be, és amelyeket a kezdőknek szánt programozási környezetek nagymértékben támogatnak [1]. Ilyen terület a teknőcgrafika, a robotika [2], az animációkészítés [3] és a játékfejlesztés is [4].

A játékfejlesztéssel kihasználható a tanulóknak a játékprogramok (működése) iránti érdeklődése, és ez a terület kézenfekvő folytatása az animációkészítésnek: a játékprogram objektumait továbbra is mozgatni és animálni szükséges, ugyanakkor a köztük, illetve a játékos és a játékprogram közti interakciók megvalósításához eseményvezérelt programozásra, az elágazások és a változók gyakori alkalmazására is szükség van [1].

A játékfejlesztésen belül gyakran esik a választás az akciójátékok műfajára, amelyek a kezdőknek szánt, objektumorientált és eseményvezérelt programozási környezetekben kismértékű absztrakcióval, praktikusan elkészíthetők. Ezekben a játékokban a főszereplővel egy leegyszerűsített absztrakt világban kell előre haladni, amelyben jellemzők az áthatolhatatlan falak, az előre haladást nehezítő ellenségek, és az előre jutáshoz szükséges tárgyak, amelyeket fel kell venni és máshol felhasználni. Jellemzően többpályásak, és folyamatosan nyilvántartják a játékos teljesítményét (például a teljesített pályák számát, a pontszámot, a felhasznált időt), és a játékos számára a cél a minél jobb teljesítmény elérése [5].

Korábbi cikkemben [4] a feladattípusorientált programozástanítási módszernek [6] megfelelően három egyre összetettebb feladattípuson – az akadálypályás, a labirintus- és a platformjátékokon – keresztül ismertettem az akciójátékoknak az imént felsorolt jellemzőinek a megvalósítási lehetőségeit a Scratch-ben. A bemutatott példaprogramok és a Scratch-ben alkalmazott megoldási módszerek a sajátjaim voltak.

Jelen cikkemben a négyzetrácsalapú akciójátékok – amelyekben a szereplők egy négyzetrács mezőin mozognak – elkészítésének a sajátosságait járom körül továbbra is a Scratch programozási környezetben, egy mintajáték megvalósításának a bemutatásán keresztül. A bemutatás meghatároz egyúttal egy lehetséges tanítási sorrendet. A megvalósítás ismertetését követően variációs lehetőségeket is megfogalmazok, amelyek további (önállóan megoldandó) feladatokként is felhasználhatók. Végül didaktikai szempontból értékelem a négyzetrácsalapú akciójátékok programozását.

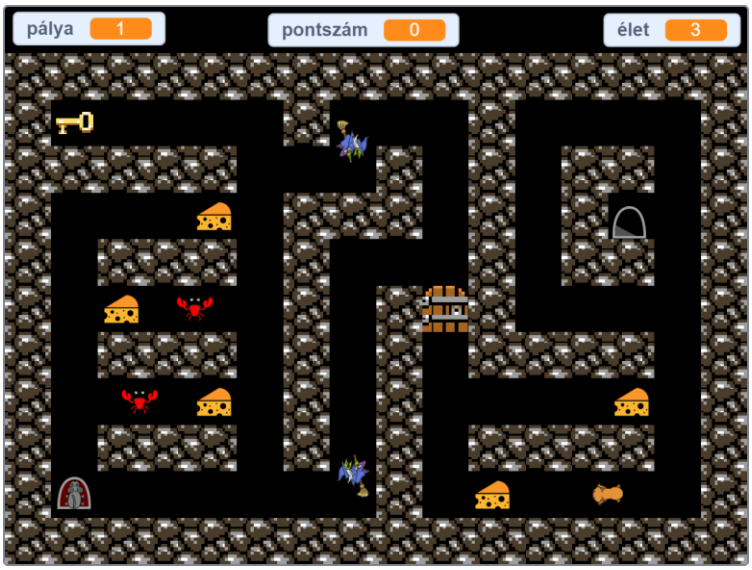
Az ebben a cikkben bemutatott mintaprogram és a Scratch-ben alkalmazott megoldási módszerek is a sajátjaim. Nem ismerek olyan irodalmat, amely a négyzetrácsalapú akciójátékok elkészítésével foglalkozna a Scratch-ben. A Scratch-re építő ismeretterjesztő könyvek egy részét a programozási alapfogalmak tematizálják, és csak egy-két nagyon egyszerű játék elkészítését mutatják be [7]. Pozitív példa Carol Vorderman könyve [8], amely kifejezetten a játékprogramozásról szól a Scratch-ben, de ez sem foglalkozik a négyzetrácsalapú játékok készítésével. Találtam más (alacsonyabb szintű, professzionális) programozási nyelvhez készült, a játékkészítésről szóló és a négyzetrácsalapú játékok készítésével is foglalkozó forrást [9], a benne foglaltakhoz képest azonban a Scratch sajátosságai miatt lényegesen eltérő megoldási módszereket kellett választanom.

2. A mintajáték bemutatása

A kétpályás (de további pályákkal bővíthető) játékban a piros színű *Bejárat* elől (a képernyőképen a bal alsó sarok környékén) induló és a nyílgombokkal irányítható szürke *Egérrel* kell összegyűjteni a pályán található összes *Sajtot,* majd távozni a szürke-fekete *Kijáraton* keresztül (amely a képernyőkép jobb felső sarkától nem messze található). Ennek a véghezvitelét a falakon kívül további akadályok nehezítik: a fából készült (a képernyőkép középső részén látható) *Ajtó* csak a *Kulcs* megszerzését követően nyitható, ha pedig az *Egér* a különböző szabályok szerint mozgó ellenségek valamelyikével ütközik, egy életet veszít és – ha még maradt élete – visszakerül a *Bejárathoz* (a *Bejáratra* az ellenségek nem léphetnek, ott az *Egér* az elindulásig biztonságban van).

Az ellenségek közül a *Rákok* folyamatosan vízszintesen haladnak, és akkor fordulnak vissza, ha falnak ütköznek. A *Boszorkányok* falkövető algoritmus szerint mozognak: mindig az irányukhoz képest balra eső falat követik. Végül a *Macska* (amely a képernyőkép jobb alsó sarkának környékén látható felülnézetben) az elágazásokban azt a (nem a háta mögötti) oldalszomszédos mezőt választja, amelyhez az *Egér* aktuálisan a legközelebb van. A játék a játékos egyes részeredményeit (például a *Sajtok* megszerzését) pontokkal is jutalmazza (1. ábra).

A kész játék és a programkód elérhető a következő címen: *https://scratch.mit.edu/projects/766757769*



1. ábra: A mintajáték első pályája.

3. A mintajáték megvalósítása

A következőkben bemutatom a játékprogram megvalósításának a fontosabb lépéseit az alulról felfelé építkezés elve szerinti sorrendben, amely az elképzelésem szerinti tanítási sorrendnek is megfelel. Az utolsó előtti lépésig a játék egypályás változata készül el, amelyben minden típusú pályaelemre található lesz példa. A több pálya megszervezésével a korábbi cikkemben már foglalkoztam, amelyet a jelenlegi mintajáték megvalósításának az utolsó lépésében használok fel. Az egypályás játék és a programkódja elérhető a következő címen: *https://scratch.mit.edu/projects/778788909*

3.1. Négyzetrácsos pálya rajzolása

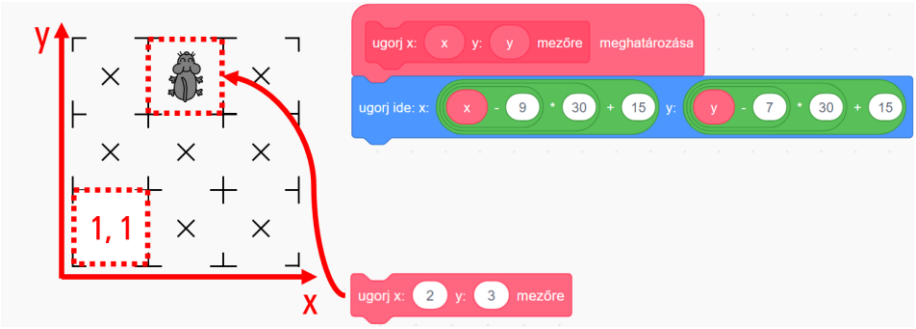
Az *Egér* mozgatásának a megvalósítása során elengedhetetlen lesz, hogy képesek legyünk ellenőrizni a falakkal történő átfedést. A Scratch-ben egy szereplő egy színárnyalat vagy egy másik szereplő érintését tudja érzékelni. A falak színárnyalatai azonban egyes szereplők jelmezeiben is előfordulhatnak, amely szereplőket nem szeretnénk tévesen falként érzékelni. Ezért a falakat egy a *Színpad* méretével megegyező méretű, (például) *Falak* nevű szereplő jelmezére kell rajzolnunk, és az *Egérnek* az ezzel a szereplővel történő átfedését kell majd vizsgálnunk.

A Scratch rajzolóablakában túl körülményes négyzetrácsalapú képet készíteni a négyzetekbe illő textúrákból („csempékből"), de léteznek kifejezetten erre a célra megalkotott könnyen használható és ingyenes rajzolóprogramok. Közülük a mintaprogramban a *Tiled*[1] nevűt használtam, a falak textúrájához pedig az *OpenGameArt*[2] oldaláról jutottam hozzá (szintén ingyenesen). A 480×360 képpontból álló *Színpadot* képzeletben 30×30 képpontos mezőkre osztottam fel, amelynek bizonyos mezőiben elhelyeztem a fal (30×30 képpontos) textúráját.

3.2. Az Egér elhelyezése a négyzetrácson

Az *Egeret* úgy szeretnénk majd mozgatni, hogy csak mezőközéppontról mezőközéppontra haladhasson. Ehhez azonban szükséges, hogy a játék elején már eleve valamely mező középpontjából induljon.

A Scratch beépített *ugorj ide x: y:* nevű parancsa a *Színpad* közepére illesztett koordinátarendszerben értelmezett x és y koordináta szerint pozícionálja a szereplőket. Körülményes lenne a számunkra állandóan kiszámolni egy-egy mező középpontjának a koordinátáit, ezért célszerű létrehoznunk azt az *ugorj x: x y: y mezőre* nevű eljárást, amely a beépített *ugorj ide x: y:* parancs segítségével a paraméterekben megadott *(x, y)* „mezőkoordinátájú" mező közepébe helyezi az eljárást meghívó szereplőt (2. ábra).



2. ábra: Az *Egér* (és a többi szereplő) elhelyezését segítő eljárás meghatározása, és egy példa a használatára.

[1] https://www.mapeditor.org/

[2] https://opengameart.org/

Az eljárás bemutatott megvalósításában az (1, 1) mezőkoordináta-pár a *Színpad* bal alsó sarkában található mezőt határozza meg, és a két mezőkoordináta jobbra, illetve felfelé haladva növekszik.

3.3. Az Egér irányítása a négyzetrácson

Az *Egeret* a játékosnak úgy kell tudnia irányítani, hogy azzal a négy oldalszomszédos mező közül a falakat (és a zárt *Ajtót*) nem tartalmazók középpontjára lehessen átsétálni. Azt szeretnénk, ha nem egyszerűen áthelyeződne az új mezőre, hanem (például) 3 képpontonként haladva jutna el oda úgy, hogy közben a mozgását ne lehessen befolyásolni, tehát például ne lehessen vele visszafordulni vagy kanyarodni.

A korábbi cikkemben láthattuk, hogy a játékos által irányítható szereplő mozgását egy olyan –például *mozdulj el* nevű – eljárással érdemes megvalósítani, amelyet a szereplő egy végtelen ciklusban (a szereplő úgynevezett játékciklusában) folyamatosan meghív.

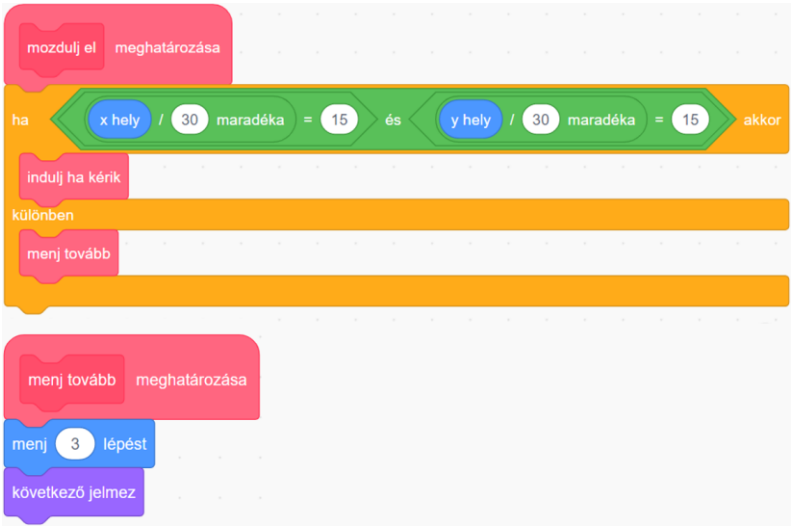
Az *Egér* tervezett működéséből következik, hogy eltérően kell viselkednie

a) egy mező középpontjában, illetve

b) azon kívül,

és hogy az utóbbi esetben a körülményektől függetlenül 3 képponttal kell az aktuális irányába mennie.

Hogy a két eset közül melyik áll fenn, az a szereplő két (eredeti értelemben vett) koordinátájából következtethető: pontosan akkor tartózkodik egy mező közepén, ha mindkét koordinátája 30-cal osztva 15-öt ad maradékul. Ennek megfelelően a *mozdul el* eljárás megvalósítását a 3. ábrán látható módon bonthatjuk fel.

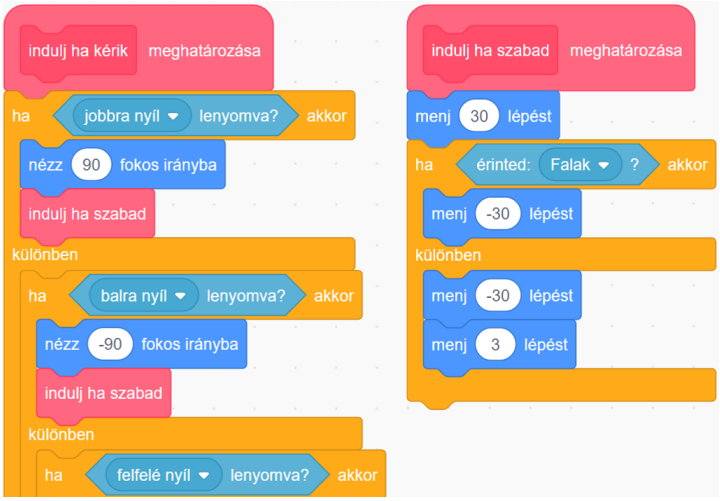


3. ábra: Az *Egér* irányítását megvalósító eljárás (fent),
és a két eset közül a másodikat megvalósító eljárás (lent).

Az *indulj ha kérik* eljárásnak kell gondoskodnia arról, hogy az *Egér* elinduljon valamelyik oldalszomszédos mező irányába 3 képponttal, ha a játékos ezt kezdeményezi, és a szóban forgó mező üres. Ha elindul, akkor a *mozdulj el* eljárás következő meghívásakor már nem mezőközéppontban lesz, ennek megfelelően a *menj tovább* eljárás fogja újabb 3 képponttal előre mozgatni. Ez az utóbbi eljárás pedig addig lesz újra és újra meghívva, ameddig az *Egér* a következő mező középpontjába meg nem érkezik. Onnan a játékos újra elindíthatja őt (ha szeretné) valamelyik megfelelő irányba.

Az *indulj ha kérik* eljárás egy lehetséges megvalósítása a 4. ábrán látható. Az *Egeret* a lenyomott nyílgombnak megfelelő irányba állítjuk (ha le van nyomva valamelyik nyílgomb), majd átadjuk a vezérlést az

indulj ha szabad eljárásnak. Ebben először átmozgatjuk a szereplőt az oldalszomszédos mezőre *(menj 30 lépést),* hogy azon elvégezhessük a *Falak* szereplővel történő átfedés ellenőrzését. Akár volt átfedés, akár nem, visszahelyezzük a szereplőt az eredetileg elfoglalt mezőre *(menj -30 lépést),* és ha nem történt átfedés, akkor elindítjuk 3 képponttal. Erre az oda-vissza helyezésre – amely olyan gyorsan történik, hogy a Scratch a képernyőn nem jeleníti meg[3] – azért van szükség, mert a szereplők csak az aktuális helyükön képesek érintést ellenőrizni.



4. ábra: Az *Egeret* az *indulj ha kérik* eljárás indítja el a játékos által meghatározott oldalszomszédos mezőre, de csak akkor, ha azon nincs fal[4].

3.4. A Kulcs és az Ajtó elhelyezése és működtetése

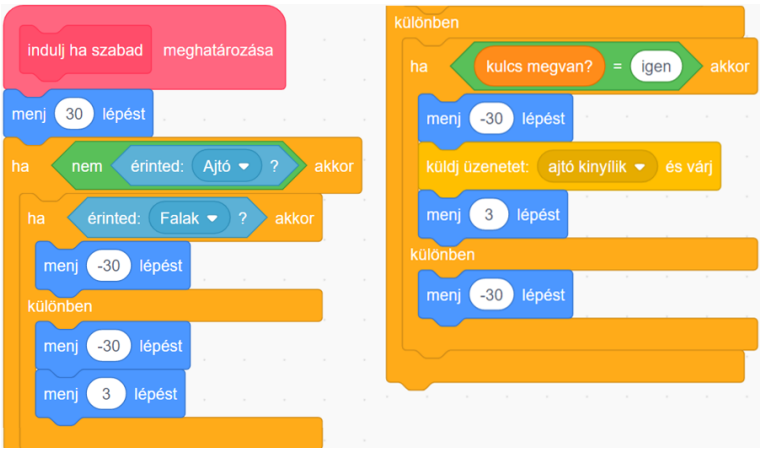
A *Kulcsot* és az *Ajtót* az *Egérhez* hasonlóan az *ugorj x: x y: y mezőre* segédeljárással helyezhetjük el a pályán.

A korábbi cikkemben találkozhattunk a kulccsal nyitható ajtó megvalósításával. A *Kulcs* működtetése a következőképpen történhet. Kezdetben egy (például) *kulcs megvan?* nevű változóban regisztráljuk (például a *nem* változóértékkel), hogy a játékos nem rendelkezik a kulccsal. A játék során pedig a főszereplő – jelenesetben az *Egér* – a saját játékciklusában folyamatosan ellenőrzi egy (például) *érzékeld a kulcsot* nevű eljárásban, hogy átfedésben van-e a *Kulccsal.* Ha igen, akkor üzenetküldéssel felszólítja a *Kulcsot* az eltűnésre, valamint a *kulcs megvan?* változóban annak a tárolására (például az *igen* változóértékkel), hogy a játékos már rendelkezik a kulccsal.

Az *Ajtó* a *Kulcs* megszerzéséig falként kell, hogy funkcionáljon, ezért a fallal való átfedés ellenőrzéséért felelős, korábban ismertetett *indulj ha szabad* eljárást kell kibővítenünk. Természetesen továbbra sem szabad az *Egeret* falat tartalmazó mezőre juttatni. Ha azonban a játékos által választott iránynak megfelelő oldalszomszédos mezőn az *Ajtó* található, akkor pontosan akkor kell azt a mezőt üresnek tekintenünk, ha a játékos már rendelkezik a kulccsal (és ebben az esetben egy üzenetküldéssel fel kell szólítanunk az *Ajtót* az eltűnésre) (5. ábra).

[3] Hasonlóan például ahhoz, ahogyan a teknőcgrafika teknőce „észrevétlenül" rajzol ki egy egyszerűbb alakzatot.

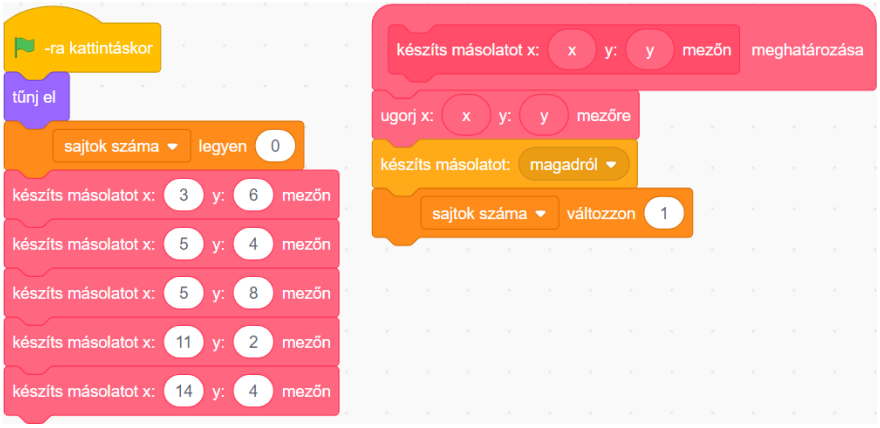
[4] A felfelé és a lefelé nyíl lenyomva tartásának a kezelése hasonlóan történhet.



5. ábra: Az *Egér indulj ha szabad* nevű eljárásának a kiegészítése az *Ajtó* kezelésével.

3.5. A Sajtok elhelyezése és működtetése

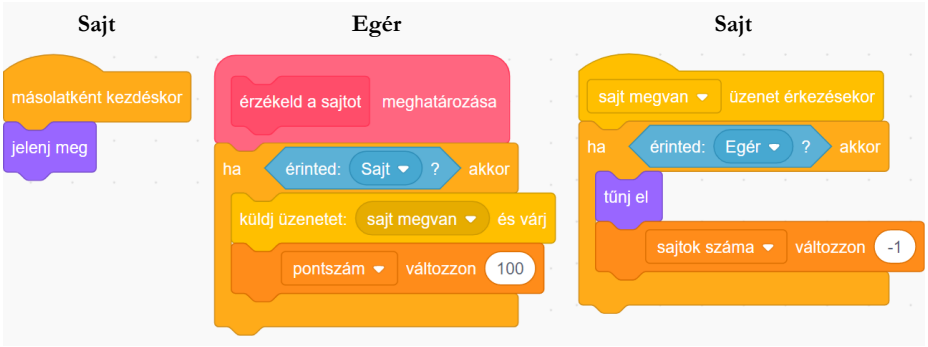
Mivel számos (de egyformán működő) sajtot szeretnénk elhelyezni a pályán, egyetlen *Sajt* szereplőből készíthetünk példányokat a Scratch másolatkészítés (klónozás) lehetőségével. Az eredeti példányát kezdetben láthatatlanná tesszük – ezáltal nem fog részt venni a játékban –, majd „pecsételőként” használjuk: a megfelelő helyeken egy-egy másolatot készítünk belőle, és a *sajtok száma* nevű változóban folyamatosan nyilvántartjuk a másolatok darabszámát (mert az beépített paranccsal nem lekérdezhető) (6. ábra).



6. ábra: A *Sajt* másolatainak az elhelyezése.

Az egyes példányoknak a létrejöttükkor egyetlen dolguk van, láthatóvá válni, hiszen a „pecsételő” minden tulajdonságát, így a pillanatnyi helyzetén túl a láthatatlanságát is megörökölték (7. ábra, balra).

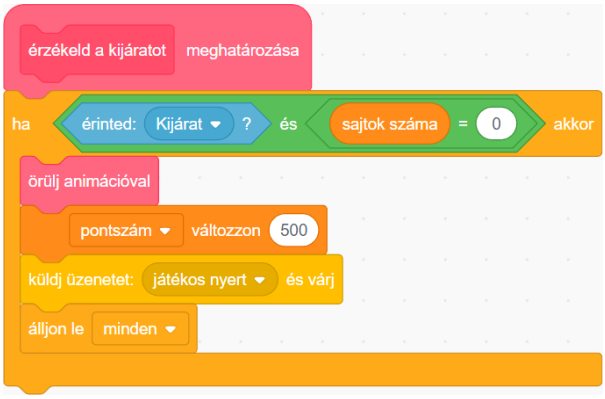
Az *Egér* a *Sajt* valamelyik másolatával történő ütközését is a végtelen ciklusában elhelyezett (például) *érzékeld a sajtot* nevű eljárásában ellenőrzi, amely szerint ha átfedésben van a *Sajt* valamelyik példányával, akkor a *sajt megvan* üzenet kiküldésével felszólítja a megfelelő példányt az eltűnésre (7. ábra, középen). Mivel azonban ezt az üzenetet minden példány megkapja, a példányoknak is ellenőrizniük kell, hogy ők vannak-e átfedésben az *Egérrel*. Ha igen, akkor (csak) a szóban forgó példány eltűnik, aki a *sajtok száma* változóban elkönyveli a találatot (7. ábra, jobbra).



7. ábra: A *Sajtok* működtetése.

3.6. A játék megnyerése

A szokásos módon elhelyezhetjük a *Kijáratot* a pályán, majd foglalkozhatunk avval, hogy ha az *Egér* eljutott a *Kijárathoz* úgy, hogy már minden *Sajtot* begyűjtött, akkor érjen véget a játék (valamilyen győzelmet kihirdető felirat megjelenítését követően). Az *Egér* játékciklusában meghívható a 8. ábrán látható *érzékeld a kijáratot* eljárás (a játékos győzelmét kihirdető felirat a *játékos nyert* üzenet érkezésekor jelenik meg).



8. ábra: A játék megnyerése.

3.7. Az ellenségek elhelyezése és mozgatása

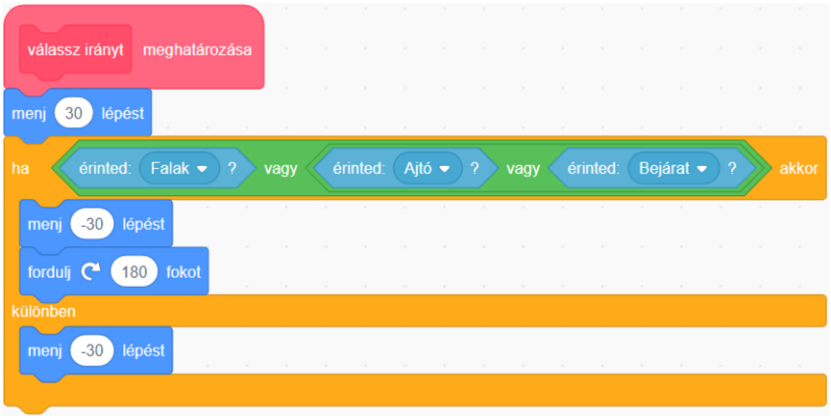
A játékot lényegesen izgalmasabbá tehetjük az ellenségekkel, amelyeket a *Sajtokhoz* hasonlóan egy-egy láthatatlanná tett prototípusuk (a *Rák*, a *Boszorkány* és a *Macska*) másolataiként hozhatjuk létre és helyezhetjük el a játék elején.

A másolatokat a létrejöttükkor meg kell jelenítenünk, majd ezután következhet a mozgatásuk megvalósítása egy végtelen ciklusban a 9. ábrán látható módon. Ha mezőközépponton állnak, akkor valamilyen logika szerint a négy lehetséges irány egyikébe fordulnak (*válassz irányt* eljárás), és mindenképpen (akár mezőközépponton tartózkodnak, akár nem) előre mennek 3 képpontot (*menj tovább* eljárás). A háromféle ellenség – egyre összetettebb – mozgása csak az irányválasztás logikájában fog különbözni.

9. ábra: Az ellenségeket mozgató eljárás.

3.7.1. A Rák mozgatása

A *Ráknak* az irányválasztás során ellenőriznie kell, hogy az iránya szerinti következő mező üres-e, hasonlóan ahhoz, ahogyan az *Egér* ellenőrizte ugyanezt: ideiglenesen rálép a következő mezőre, és ha az foglalt (mert fal van rajta, vagy az *Ajtó*, vagy a játékos számára védelmet biztosító *Bejárat*), akkor az eredeti mezőre visszakerülve ellentétes irányba fordul, ha pedig nem foglalt, akkor az irányának a módosítása nélkül ugrik vissza az eredeti mezőre (ahonnan majd a *menj tovább* eljárás elmozdítja) (10. ábra).

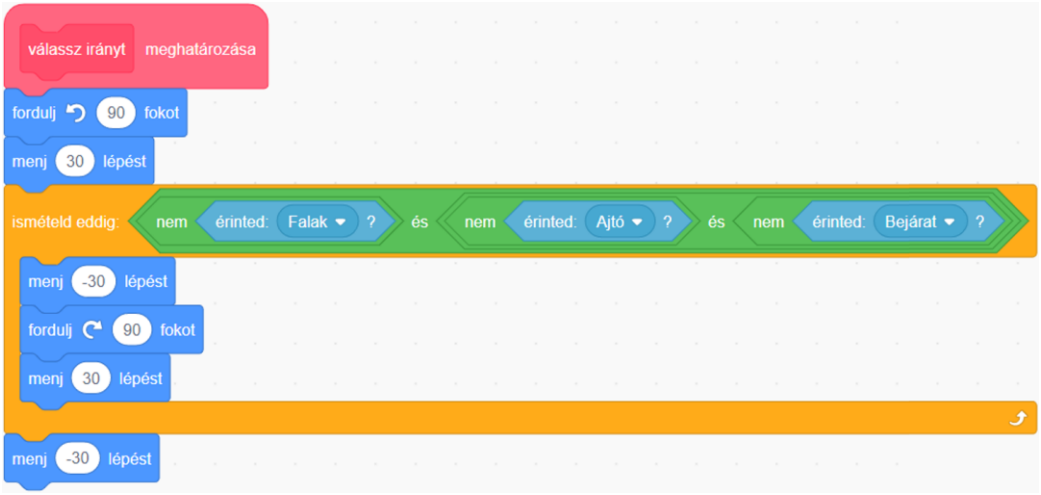


10. ábra: Ha a *Rák* nem léphet az iránya szerinti következő mezőre, visszafordul.

3.7.2. A Boszorkány mozgatása

Mint arról a játék bemutatásakor szó esett, a *Boszorkányokat* falkövető algoritmus szerint szeretnénk mozgatni: mindig az irányukhoz képest balra eső falat kell követniük. A lehetséges eseteket megvizsgálva rájöhetünk, hogy a *válassz irányt* eljárásban az irányukhoz képest balra, előre, jobbra, illetve hátra található mezők közül ebben a sorrendben az első üreset kell kiválasztaniuk.

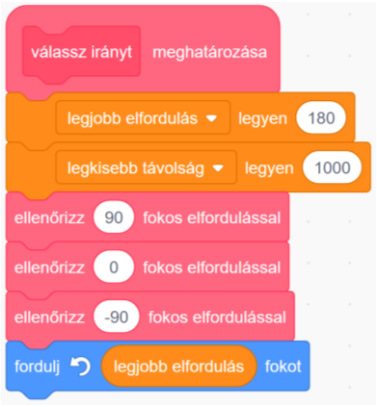
A *kiválasztás programozási tételt* [10] alkalmazhatjuk: először a balra eső mezőt vizsgáljuk meg, majd amíg foglalt az aktuálisan vizsgált szomszédos mező, addig vizsgáljuk sorban a következőt. Ehhez az *ismételd eddig* feltételes ciklust használhatjuk (amelyben a kilépés feltételét kell megfogalmazni) (11. ábra). A *Boszorkányok* háta mögötti mező (ahonnan az aktuális mezőre érkeztek) biztosan szabad, így biztosan találni fognak továbbhaladási irányt.



11. ábra: A *Boszorkány* az iránya szerinti balra, előre, jobbra és hátra található oldalszomszédos mezők közül (ebben a sorrendben) az első szabad mezőt választja.

3.7.3. A Macska mozgatása

A háromféle ellenség közül a macska lesz a „legintelligensebb" (és a legveszélyesebb), ugyanis figyelembe fogja venni az *Egér* aktuális helyét is: az elágazásokban azt a nem a háta mögötti szabad oldalszomszédos mezőt fogja választani, amelyhez az *Egér* a legközelebb van. A mozgását életszerűbbnek találtam úgy, hogy csak az elágazásokban változtathat az irányán (valamint a zsákutcákban, ahol visszafordul).

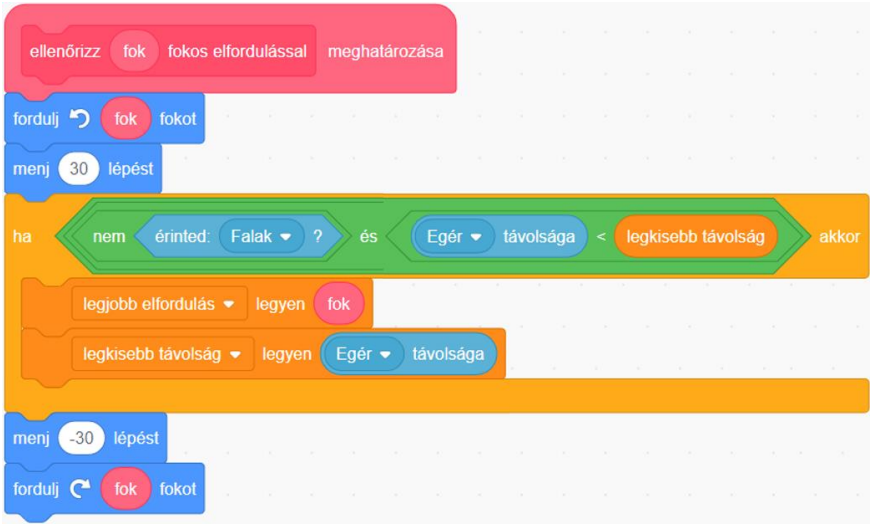


12. ábra: A *Macska* az irányához képest balra, előre és jobbra található mezőket ellenőrzi.

A *Boszorkánnyal* ellentétben a *Macskának* az irányához képest balra, előre és jobbra található mezőre is mindenképpen rá kell lépnie (például ebben a sorrendben), mert előfordulhat, hogy egy később ellenőrzött szomszédos üres mező közelebb van az *Egérhez* a már ellenőrzött üres mezőknél.

A feltételes minimumkiválasztás programozási tételt használhatjuk. Kezdetben a 180 fokos elfordulást (mint legrosszabb esetet) tekintjük a legkedvezőbb iránynak (*legjobb elfordulás* változó), és az 1000 képpontot (mint „végtelen" távolságot) az *Egértől* mérhető legkisebb lehetséges távolságnak (*legkisebb távolság* változó). Ha a három ellenőrzésre szánt oldalszomszédos mező valamelyike üres és közelebb

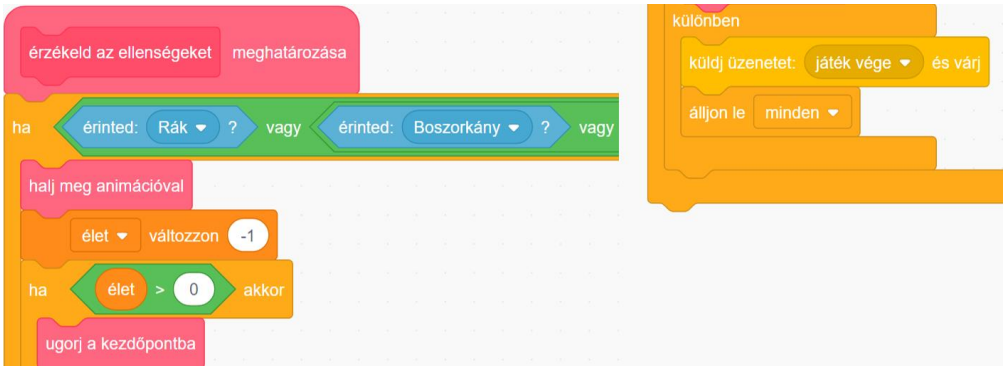
van az *Egérhez* az eddigi legkisebb távolságnál, akkor frissítjük a legkedvezőbb irányt és az *Egértől* mérhető legkisebb lehetséges távolságot. Végül a legkedvezőbb irányba fordítjuk a *Macskát* (12. és 13. ábra).



13. ábra: Ha a *Macska* egy adott irányban az *Egérhez* közelebb eső szabad mezőt talál, azt tekinti az addigi legjobb választásnak.[5]

3.8. Az ellenségekkel való ütközés ellenőrzése

Az *Egér* a valamely ellenséggel történő ütközést a végtelen ciklusában meghívott *érzékeld az ellenségeket* eljáráson belül ellenőrizheti, amelynek bekövetkezése esetén az *élet* változó értékét eggyel csökkentenie kell, valamint, ha még maradt a játékosnak élete, akkor vissza kell kerülnie a pálya kezdőpontjába (*ugorj a kezdőpontba*), különben viszont véget ér a játék (14. ábra).



14. ábra: Az ellenségekkel való ütközés ellenőrzése[6].

[5] A képről helyhiány miatt az elágazás feltételéből lemaradt az *Ajtóval* és a *Bejárattal* való átfedés ellenőrzése, amelyet a *Falakkal* való átfedés ellenőrzéséhez hasonlóan lehet elvégezni.

[6] A *Macskával* történő átfedés ellenőrzése hasonlóan történhet.

3.9. A többpályás változat létrehozása

A korábbi cikkemben foglalkoztam a több pálya kezelésének módszerével, amelynek a lényegén nem változtat az, hogy a most szóban forgó játék négyzetrácsalapú. A módszer lényege a következő.

Az *Egér* a zöld zászlóra kattintáskor (a játék indításakor) egy-egy üzenet kiküldésével kezdeményezi a játék, majd az első pálya inicializálását (15. ábra, balra).

A *játék előkészítése* üzenet érkezésekor azon változók beállítására kerül sor, amelyeknek csak a játék, és nem minden egyes pálya elején kell kezdőértéket adni: ilyen a *pályák számát* tartalmazó változó, a *pálya* változó (amely az aktuális pálya sorszámát tartalmazza), valamint az aktuális *pontszámot* és *életet* tartalmazó egy-egy változó (15. ábra, középen).

15. ábra: Az *Egér* fő eljárása (balra), valamint tevékenységei a *játék előkészítésekor* (középen) és a *pálya berendezésekor* (jobbra).

A *pálya berendezésekor* üzenetet pedig minden szereplő megkapja, és hatására a szereplők, így például az *Egér*, a pályának megfelelő kezdőpontjukba kerülnek (15. ábra, középen és jobbra), és a *Falak* szereplő is a *pálya* változó értékének megfelelő sorszámú jelmezét állítja be.

Lényeges továbbá, hogy a játékos a pálya megfejtésekor (amikor az *Egér* a *Kijáratnál* van, és már minden *Sajtot* megszerzett), már csak akkor nyeri meg egyúttal a teljes játékot, ha az aktuális *pálya* sorszáma megegyezik a *pályák számával*. Különben növelni kell eggyel a *pálya* változó értékét, és ki kell küldeni a *pálya berendezése* üzenetet, amelynek hatására berendezésre kerül a következő pálya, majd folytatódik az *Egér* játékciklusa, aminek köszönhetően a játékos hozzá is láthat a következő pálya megfejtéséhez.

4. Variációs lehetőségek

A bemutatott játéktípus számtalan variációs lehetőséget rejt magában a pálya és a szereplők kinézetének (és így a játék kerettörténetének és hangulatának) a megváltoztatásán túl.

Elkészíthetők például a játéktípusnak olyan variációi, amelyekben a főszereplő „kilép” a négyzetrácsból, és másféle „fizikai törvények” befolyásolják a mozgását, például hat rá a „gravitáció” (ilyen főszereplőirányítást megvalósítottam az előző cikkemben).

Vagy létrehozható egy a *Falak* szereplőhöz hasonló további (például *Akadályok* nevű) szereplő, amelynek a jelmezei az egyes pályák mozdulatlan ellenségeit tartalmazzák, és ennek a szereplőnek az érintése ugyanúgy életvesztéssel jár, mint a találkozás valamely mozgó ellenséggel.

A kulcs mellett vagy helyett szüksége lehet a játékosnak további tárgyak érintésére vagy megszerzésére a pálya bizonyos területeinek az eléréséhez, így például kapcsolók nyithatnak ajtókat.

Meg lehet változtatni a következő pályára lépés feltételét is: például legyen-e kijárat a pályán, vagy össze kelljen-e gyűjteni valamelyik szereplő összes példányát a továbbjutáshoz, vagy csak a pontszámot gyarapítsák?

A mozgó ellenségek mozgási szabályainak is sokféle variációja található ki (vagy leshető el valamilyen létező játékból). Például egy ellenség választhat véletlenszerűen a továbbhaladást biztosító mezők közül. Vagy mozgási útvonalak definiálhatók a pálya megfelelő pontjain elhelyezett, az ellenségeket 90 fokos balra vagy jobbra fordulásra felszólító, a játékos számára nem látható „irányjelző táblákkal" (ezzel a megoldással járőröztek például a *Wolfenstein 3D* című 1995-ös játék ellenséges katonái).

5. Értékelés

Cikkemben – a megelőzőre építve – bemutattam a négyzetrácsalapú akciójátékok legjellemzőbb összetevőinek a megvalósítási lehetőségeit a Scratch programozási környezetben. A „négyzetrácsalapúság" programozási szempontból elsősorban a szereplők elhelyezésének és mozgatásának a módját határozta meg, de ebben a játéktípusban foglalkoztam először a szereplők másolatainak (klónjainak) a kezelésével is, mert a sok mezőből álló pályák nagyon motiválják a sok másolat használatát.

A videójátékok világában az akciójátékok pályái nagyon gyakran négyzetrácsalapúak. Ennek az oka nyilvánvalóan az (amelyet a saját tapasztalataim is alátámasztanak), hogy könnyebb így megtervezni a pályák felépítését, és meghatározni a szereplők (például a főszereplő és az ellenségek) viselkedési szabályait. A szóban forgó játéktípus nagy előnyének tartom, hogy jól megfogalmazható és megvalósítható variációs lehetőségeket rejt magában, ezáltal teret biztosítva a tanulói kreativitásnak. Köszönhetően annak, hogy nagyon sok játék négyzetrácsalapú, számtalan ötlet meríthető (tanárként és diákként is) „valódi" játékok pályáiból és működéséből. Érdemesnek tartom megjegyezni azt is, hogy a szereplők (például az ellenségek) mozgatása kapcsolódhat robotikai típusfeladatokhoz, például az útvonalkövetéshez, a falkövetéshez vagy akár a labirintusok megfejtésének egyéb algoritmusaihoz is.

Gyakran állítják informatikatanárok is néhány tanórányi „Scratch-ezést" követően, hogy „nincs benne több". A korábbi és a jelenlegi cikkemmel célom volt annak a megmutatása is, hogy az egymásra épülő feladattípusok megfelelő összeállításával egy kezdőknek szánt programozási környezetben egy kezdőknek szánt programozási témakörön belül is sokféle programozási fogalom használatát igénylő és igen összetett programok fejlesztéséig lehet eljutni.

Irodalom

1. Bernát, P., Zsakó, L.: *Methods of teaching programming – strategy.* In: New methods and technologies in education and practice: Proceedings of XXX. DIDMATTECH 2017. pp 40-50.
2. Bernát, P.: *Robotika az általános iskolában és a RoboMind programozási környezet* In: Szlávi, Péter; Zsakó, László (szerk.) INFODIDACT 2015, Webdidaktika Alapítvány.
3. Bernát, P.: *Teaching introductory programming by creating animations with Scratch.* In: New Methods and Technologies in Education, Research and Practice 2020. ELTE Informatikai Kar. pp. 30-39.
4. Bernát, P.: *Feladattípusorientált játékfejlesztés a Scratch-ben.* In: INFODIDACT 2017. https://people.inf.elte.hu/szlavi/InfoDidact17/Manuscripts/BP.pdf (utoljára megtekintve: 2022.12.18.).
5. *Action game.* Wikipedia. https://en.wikipedia.org/wiki/Action_game (utoljára megtekintve: 2022.12.18.).
6. Szlávi, P., Zsakó, L.: *Methods of teaching programming.* In: Teaching Mathematics and Computer Science, 1. kötet, pp. 247-257.
7. McManus, S.: *Tanulj meg kódolni 10 lépésben!,* Babilon Kiadó (2017).
8. Vorderman, C.: *Computer Coding Games for Kids,* Dorling Kindersley, London (2015).
9. Egges, A.: *Swift Game Programming for Absolute Beginners,* Apress, New York (2015).

10. Szlávi, P., Zsakó, L.: *Módszeres programozás: Programozási tételek*. NJSZT, Budapest, 1999.

Milyen kérdéseket vet fel az oktatás területén a kvantumszámítógépek megjelenése?

Biró Csaba[1,2], Koczka Ferenc[1,3], Prantner Csilla[1]

{biro.csaba, koczka.ferenc, prantner.csilla}@uni-eszterhazy.hu
[1] Eszterházy Károly Katolikus Egyetem
[2] Eötvös Loránd Tudományegyetem
[3] Nemzeti Közszolgálati Egyetem

Absztrakt. A kvantumszámítógépek megjelenése várhatóan jelentős változásokat indukál az informatika területén. A teljesen újszerű elveken működő számítógépek korábban nem megoldható problémákra hozhatnak hatékony megoldást. Az Európai uniós és magyar törvénykezésben megjelentek már azok a rendeletek, amelyek ráerősítenek arra, hogy egyre közelebb vagyunk a kvantumszámítógépek időszakához. Úgy gondoljuk, hogy az oktatásnak és a tanároknak lépést kell tartaniuk a technológiai fejlődéssel, hogy ne érje váratlanul az informatika szektorban jövőben elhelyezkedő diákokat, legyen szó informatika tanárok, fizikusok vagy programozók képzéséről. Tanulmányunkban néhány külföldi példát mutatunk be, ahol a kvantuminformatika témakörei a képzések bizonyos szintjein már bekerültek az oktatásba.

Kulcsszavak: kvantuminformatika, kvantumfizika, általános iskola, középiskola, felsőoktatás, specializáció, STEM, oktatás.

1. Bevezetés

A kvantumszámítógépek kutatásairól szóló hírek begyűrűztek hétköznapi életünkbe, az online és nyomtatott sajtó tudományos rovatainak hasábjain mindennaposak a kvantuminformatikával kapcsolatos hírek, találgatások, spekulációk és a nagyvállalatok [11] fejlesztéseiről szóló tudósítások. Megjelentek a tengerentúlon az Európai Unióban és Magyarországon is a poszt-kvantum titkosítással kapcsolatos és a kvantumszámítógépek fejlesztésének támogatására irányuló jogszabályok és rendelkezések is [22, 23, 24, 41, 52, 63].

Ezek megszületése egyértelműen felszínre hozza a kvantumszámítógépekkel kapcsolatos érdeklődést, találgatást, adatbiztonsági és jelszóvédelmi kérdéseket. Egyre sürgetőbbé válik, hogy az emberek hiteles forrásokból szerezzenek ismereteket a kvantumszámítógépek működéséről, fizikai alapjairól, alkalmazási területeiről, jelenlegi fejlesztésekről, valamint a témával kapcsolatosan várható társadalmi és gazdasági hatásokról.

Fontossá vált néhány kérdés: pontosan mit is jelent a kvantumszámítógép, milyen fogalmak kapcsolódnak hozzá, mi a kvantumadatok tárolásának, továbbításának az alapegysége, ezt hogy értelmezzük, hogyan tároljuk és egyáltalán hogyan működnek a kvantumszámítógépek? Milyen fizikai elvek mentén lehet ezeket a gépeket megépíteni? Várhatóan felváltják-e a kvantumszámítógépek a mai számítógépeinket, hol lehet látni, kipróbálni, beszerezni ilyen újszerű elven működő gépet?

2. Rövid történeti áttekintés

A klasszikus értelemben vett kvantummechanika gyökerei az 1800-as évek elejére, empirikus fizikai-kémiai kutatásokra vezethetők vissza. A hőmérsékleti sugárzás és színképelemzés kapcsolatának vizsgálatakor fekete vonalakat találtak a színképben [7, 8, 54, 55]. Ez az anomália számos újabb és

újabb kérdés elé állította a kutatókat, amelyek megválaszolása közben az 1900-as évek elejére a kérdések, már a közben önálló tudománnyá váló kvantummechanika területéről érkeztek.

Az 1900-as évek elejére a spektrumokat helyesen leíró, de a klasszikus elméleti megközelítéseket használó kutatások sorra kudarcot vallottak. Planck új utat választott az atomi oszcillátorok fizikai tulajdonságainak értelmezéséhez a fenomenologikus termodinamika helyett az entrópiából indul ki [49, 50]. Az entrópiafüggvény és a termodinamikai valószínűség közötti kapcsolat már ismert volt, illetve az is, hogy egy izolált rendszer egyensúlyi állapotában veszi fel a maximális értékét. Feltevése az volt, hogy egyes termodinamikai valószínűséghez tartozó mikroállapotok száma megszámlálható, ez pedig csak abban az esetben lehetséges, amennyiben az energiát részekre (véges nagyságú energiaadagokra) osztjuk. Másképp megfogalmazva, a hőmérsékleti sugárzás (mechanikai oszcillátor) energiája nem folytonos, hanem kvantált. Kvantumhipotéziséről 1900. december 14-én Berlini Királyi Porosz Akadémia tudományos ülésén számolt be "Ueber das Gesetz der Energieverteilung im Normalspectrum" című előadásában, ezt a napot tekintjük a kvantumfizika születésnapjának.

Ez igazi paradigmaváltás volt, amelyet a tudományos irányvonalat meghatározó vezető fizikusokból álló közösség fenntartásokkal fogadott, és csak évekkel később ratifikált. A kezdeti bizonytalanság azonban nem szabott gátat az új irányzat kibontakozásának az 1900-as évek első három évtizedében évről-évre sorra dőltek meg a régi, és az újonnan született eredmények egyaránt. Az első évtized végére a kvantumfizikával párhuzamosan, egy új tudományág is napvilágot látott, a relativitáselmélet.

3. A hagyományos számítógépek fejlődésének határai

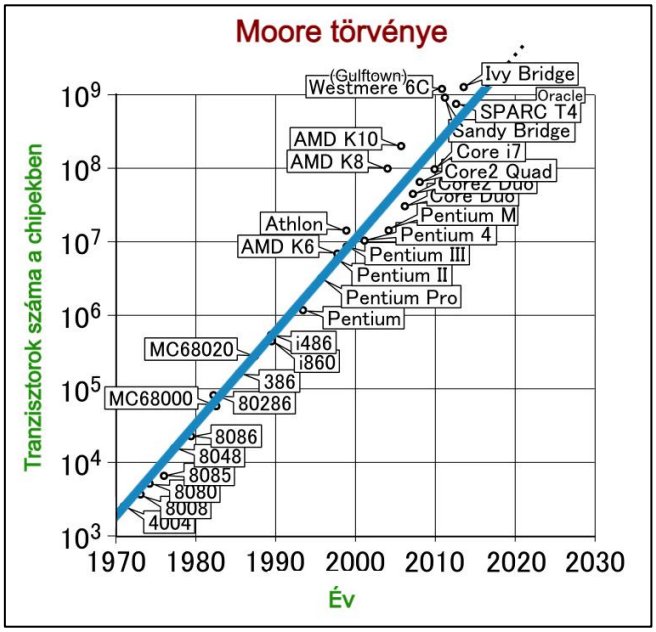
A kvantumszámítógépek építésére, mint új technológiára amiatt is van nagy igény, mert a jelenleg használatban lévő, Neumann-elvű számítógépek hamarosan elérik – sőt, néhány szempontból már el is érték − technológiai fejlődésük határát, erre a következő három grafikon által megjelenített tendenciák, világosan rámutatnak.

Moore-törvénye nagyon jól szemlélteti a klasszikus számítógépek rohamos mértékű, technológiai fejlődését. Valójában nem törvényről, hanem egy tapasztalati megfigyelésről van szó, miszerint az integrált áramkörökben lévő tranzisztorok száma 1,5 évente megkétszereződik. A megfigyelés Gordon E. Moore-nak, az Intel egyik alapítójának a nevéhez fűződik, aki az 1965-ben az *Electronics Magazine*-ban megjelent cikkében írt először az általa megfigyelt tendenciáról [27]. Habár Moore eredeti megfogalmazása nem pontosan úgy szólt, ahogyan az elterjedt, mindenesetre a processzorgyártók fejlesztési tempójára nagy hatással volt. Az eredeti megfogalmazás szerint a legalacsonyabb árú komponens összetettsége évenként nagyjából a kétszeresére nő és a jövőre tekintve ez a fejlődési ütem nem fog jelentősen változni. A chipfejlesztő cégek számára ez a megfigyelés tulajdonképpen egy teljesítendő célkitűzéssé vált, s a jövőre vonatkozóan is mintegy önmegvalósító jóslat működött, azaz teljes egészében prognosztizálta a gyártási menetet sok éven keresztül.

Olyannyira, hogy ennek megfelelően a tranzisztorok korát felváltotta az integrált áramkörök ideje, tehát a klasszikus működési elv megmaradt, csak más technológiai elemekkel oldották meg a kívánt fejlődési ütemet.

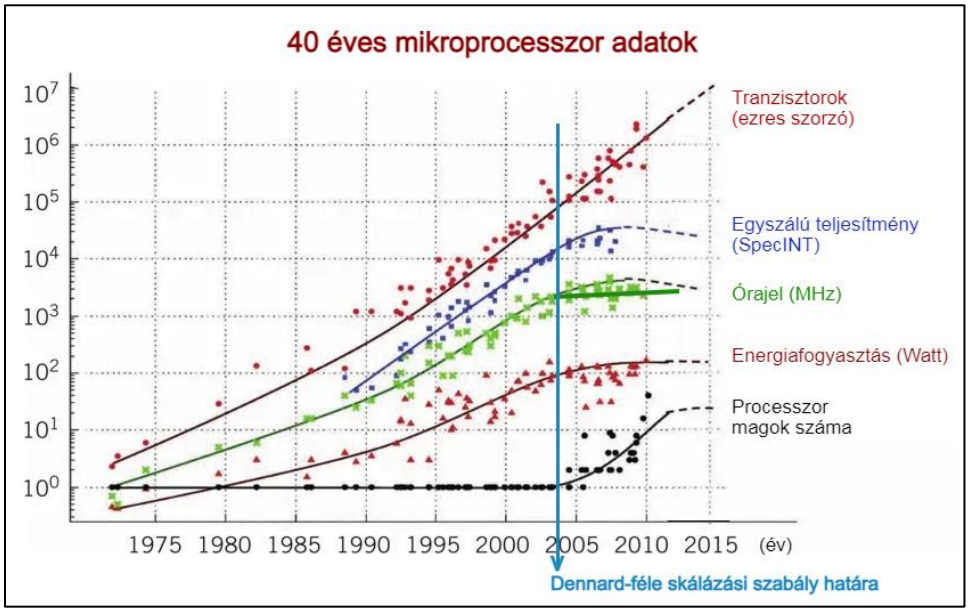
Az alábbi grafikonon [56] jól nyomon követhető az integrált áramkörök összetettségének növekedése, amely első ránézésre egy lineáris növekedést mutat, de ha jobban megnézzük, a függőleges tengely logaritmikus skálázású, így érzékelhető, hogy exponenciális növekedésről van szó. A vízszintes tengelyen az egyes processzorok kiadásának éve, a függőleges tengelyen pedig az egyes chipekben lévő tranzisztorok száma került ábrázolásra. Ez azt jelenti, hogy egyre kisebb tranzisztorokat építenek a számítógépchipekbe, ma a legmodernebb processzorok esetében 20x20-as szilíciumatomnyi területen fér el egy tranzisztor. Egyértelmű, hogy ez a fejlődés tarthatatlan, hiszen előbb-

utóbb a miniatürizálással elérjük az atomi léptéket és akkor a klasszikus típusú, Neumann-elven felépült számítógépek fejlődése e tekintetben eléri a határt.



1. ábra: Moore-törvény tendenciája. Szerkesztette Prantner Csilla [56] alapján.

Az alábbi extrapolációs grafikonon a mikroprocesszorok jellemzői láthatók az idő függvényében, ezek a következők: tranzisztorok száma, egyszálú teljesítmény, CPU-frekvencia, energiafogyasztás, magok száma [36]. 2004-nél a grafikonon látható egy jelzés, erre az évszámra teszik a Dennard-féle skálázási szabály megdőlését.

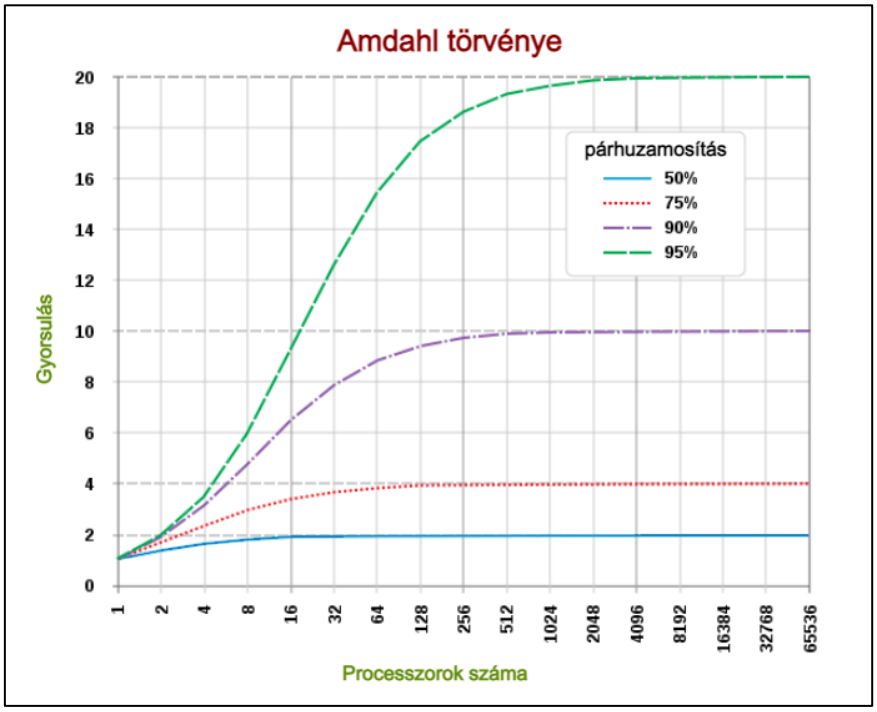


2. ábra: Dennard-féle skálázási szabály. Szerkesztette Prantner Csilla [36] alapján.

A szabályt Dennard skálázási törvényének is nevezik, amely azt mondja ki, hogy a tranzisztorok méretének csökkentésével a teljesítménysűrűség állandó marad, azaz hiába válnak kisebb méretűvé a tranzisztorok, azok ugyanolyan teljesítményre lesznek képesek. Ez a felfedezés előirányozta és biztosította a processzorok nagyiramú fejlesztését jó pár évre. A törvényt eredetileg Robert H. Dennard 1974-es cikkére alapozva MOSFET-ekre fogalmazták meg. [17].

A grafikon a jobb oldalán megjelenő tényezők közül a középsőt vizsgáljuk, ahol az órajel változásának üteme található. 2004-től már lineáris függvényt látunk, azaz 2004 óta 3-4 Hz órajelesek a processzorok. Az utóbbi években tehát nem tudták már az órajelsebességet növelni, márpedig, mivel a számítógépek az órajel ütemére végzik el az utastásokat, ezek elvégzési sebessége sem növelhető. Ez is a jelenlegi, klasszikus számítógépek fejlődésének határát mutatja.

A harmadik diagram Amdahl törvényét szemlélteti, amely a párhuzamos programozással kapcsolatos. Elviekben a párhuzamos programozás lehetőséget ad arra, hogy gyorsítsunk az algoritmusok lefutásán, hiszen egyidőben több processzor tud dolgozni ugyanazon a problémán.



3. ábra: Amdahl törvénye. Szerkesztette Prantner Csilla [17] alapján.

Az IBM-nél is dolgozó, Gene Amdahl 1967-ben az *AFIPS konferencián* beszélt arról, hogy egy párhuzamos feldolgozású programban a viszonylag kevés, egymás után végrehajtandó utasítás, korlátozó tényezőt jelent a program gyorsulására, így több processzor hozzáadásával a program nem feltétlenül fut sokkal gyorsabban. A szakember egy pontos képletet is megfogalmazott megállapítására vonatkozóan [1]. Az alapvető probléma az, hogy kevés olyan feladat van, ami valóban jól párhuzamosítható, azaz hogy az egyes programszálak oly annyira elkülöníthetőek egymástól, hogy azok nem épülnek egymás számításaira, részeredményeire, vagyis egyik programrész futására nincs függésben és nincs befolyással sem egy másikra. A párhuzamosítható problémákra egy jó példa a térképelemzés, amikor például egy nagy területet ugyanakkora méretű, kisebb területekre osztunk fel, és párhuzamosan történik ezek elemzése, például mely részeken található víz vagy erdős terület stb. Az összegzéshez ebben az esetben is össze kell várni az egyes elemzett területekről kapott eredményeket. A lényeg tehát az, hogy kevés jól párhuzamosítható probléma létezik. Az alábbi diagram jól szemlélteti azt, hogy a bizonyos százalékos arányban párhuzamosítható algoritmusok esetében, mennyit jelent programfutási sebességet illetően a processzormagok számának növelése [66].

Ha feltesszük, hogy olyan problémát oldunk meg, ami 95%-ban párhuzamosítható (zöld szaggatott vonal, csak nagyon speciális esetekben valósulhat meg), akkor is 2048 processzormagnál magasabb darabszám nem hoz már javulást. Amennyiben 50%-osan tudunk már problémákat párhuzamosítani (kék folytonos vonal, ami ritkaság), akkor már 16 magos processzornál nagyobb sem jelent nagyobb gyorsaságot. Tehát efölött a processzormagok gyarapítása inkább reklámfogás, mintsem valóban jól kihasználható lehetőség, hiszen az alapvető határ a programok logikai szinten történő jó párhuzamosításában van.

A bemutatott diagramok mindegyike arra világít rá, hogy elértük a Neumann-elvű számítógépek határait, és egy merőben más alapokon nyugvó technológiában érdemes gondolkodnunk, a külön-

böző fizikai megoldásokon alapuló kvantumszámítógépek ígéretesek e szempontból, amelyek igazi paradigmaváltást jelentenek

4. Kvantumszámítógépek fejlesztése, felhasználási területei

4.1. Felhasználási területei

Az elmúlt évtizedben áttörésnek lehettünk szemtanúi, egyre-másra jelentek meg a kvantumszámítógépek fejlődésével kapcsolatos hírek, melyek hatása különböző területeken lesz érzékelhető. A fejlesztésekbe hatalmas összegeket invesztálnak a vezető nagyhatalmak mellett olyan országok is, mint Ausztrália vagy Svájc. A fejlesztések célja négy fő területre osztható.

Az elsődleges, egyúttal a legszélesebb körben ismert ilyen a mai számítási teljesítmény megsokszorozására irányul, erre alapozva a mai szuperszámítógépeket a jövőben kvantumgépek válthatják le. Az így elérhető számítási teljesítmény számos területen lenne alkalmazható az időjárás előrejelzés hatékonyságának javításától a tudományos kutatásokig [59].

Szintén népszerű alkalmazási terület az anyag- és gyógyszerkutatás, általában minden olyan kutatási terület, amelyben a nagy számítási teljesítményt gyakorlati kivitelezés és tesztelési folyamat kiváltására vagy szűkítésére lehetne alkalmazni [59].

A kvantuminternet a mai hálózati technológiák területén jelenthet ugrást, melynek alapja az összefonódott kvantumbitek kapcsolatán alapuló kommunikációs lehetőség. Ez a szupergyors internetkapcsolat megvalósításán túl egyértelművé teszi az átvitt adatok bizalmasságának sérülését is.

A negyedik fejlesztési irány célja maga a kvantumszámítógép tökéletesítése, egy ilyen gép megépítése még napjainkban is számos problémát vet fel.

4.2. Fizikai megoldási lehetőségek

Egy kvantumszámítógép fizikai működésének pontos megértése kvantumfizikai alapismeretek nélkül meglehetősen nehézkes, ezért esetenként még annak programozói sem ismerik teljes mélységében. Ilyen gépet számos, különböző fizikai jelenség alapján lehet építeni, működésük egyaránt alapulhat a fény polarizációján [31, 46], egy atommag spinjén [6, 37] vagy az elektronok pozícióján.

Mivel a legtöbb gép működése abszolút nulla fok körüli hőmérsékleten, legfeljebb néhány másodpercig tartható fenn, jelenleg is számos kutatás irányul további, kedvezőbb tulajdonságú, jobban használható fizikai alap megtalálására.

4.3. Alapvető fogalmak

A *quantum* latin eredetű szó, amely mennyiséget jelent, többes száma a quanta. A szót közvetlenül a latinból Max Planck vezette be a fizikába, 1900-ban, a "létező mennyiség minimális mennyisége"[1] fogalmával; Einstein megerősítette, 1905. A kvantumelmélet 1912-ből való, a kvantummechanika pedig 1922-ből [71]. Ez a lehető legkisebb egység a fizikában, amellyel egy mérhető egység értéke növelhető [53]. Általában az atomi vagy a szubatomi részecskék, például az elektronok, a neutrínók vagy a fotonok tulajdonságaira alkalmazzák [44].

A gép működésének alapja az ún. *qubit*, mely a gép elemi egysége, és leginkább a hagyományos számítógépek bitjéhez hasonlítható, attól azonban jelentősen eltér. Míg egy bitnek kétféle (nulla vagy egy) értéke lehet, addig egy qubit ezeket és ezek között bármilyen értéket felvehet, akár egyidőben is, ez az ún. *szuperpozíció*. Szuperpozícióban a kvantumrészecskék tehát az összes lehetséges állapotnak a kombinációjában vannak ugyanabban az időpillanatban, csak más-más valószínűséggel, ez teszi

[1] Megfogalmazás angolul az Online etimológia szótárban: "minimum amount of a quantity which can exist;".

lehetővé, hogy egy kvantumgép egyszerre nagyon sok értékkel tud számolni, egyes források szerint egy 30 qubites gép a mai leggyorsabb szuperszámítógépek teljesítményével lenne összevethető. A bináris pozíció és a szuperpozíció közötti különbséget például egy pénzérmével jól lehet szemléltetni. A bináris állapotok (fej vagy írás) könnyen érthető. A szuperpozíció viszont ahhoz hasonló, mint amikor a pénzérmét feldobjuk és az folyamatosan pörög. Ekkor bármely érték lehet a végeredmény, minden időpillanatban bizonyos százalékkal (amplitúdóval) veszi fel a lehetséges értékeket, tehát bizonyos százalékkal a fej és bizonyos százalékkal az írás értéket. Egy másik példával élve, ha mondjuk 0-9 egész számok közül keressük egy számítás végeredményét, mindegyik érték egyidőben bizonyos százalékos bizonyossággal lehet a megoldás. A végeredmény az lesz, amely tartósan hozza a legnagyobb valószínűséget, a százalékos arány tehát ott csúcsosodik ki.

Nehézség a kvantuminformatikában, hogy a kvantumrészecskék változnak, ingadoznak folyamatosan, egészen addig, amíg meg nem mérjük őket. A qubitek rendkívül érzékenyek tehát, ennek pedig egyik leghátrányosabb következménye, hogy azok állapotát csupán egyszer lehet kiolvasni, ezt követően az azokban tárolt érték elvész. A kvantumrészecskéknek a szuperpozíció mellett van egy másik, nagyon érdekes tulajdonságuk, ez pedig az *összefonódás*. Amikor a qubitek összefonódnak, akkor egyetlen rendszert alkotnak, és hatással vannak egymásra. Az egyik qubitből származó mérések alapján következtetéseket vonhatunk le a másikra vonatkozóan és viszont. Ha több qubitet adunk hozzá összefonódással egy rendszerhez, akkor a számítógép hatványozottan több információt tud kiszámítani, és bonyolultabb, több változós problémákat is meg tud oldani. Emiatt jó olyan számításokra ez a technológia, ahol sok tényezőt/hatást/összetevőt szükséges egyidejűleg figyelembe venni és ezek egymásra való hatását is figyelembe venni. Az összefonódással korrelálni lehet az egyes kvantumrészecskék mérési eredményeit és a kvantuminformatikában ezt a tulajdonságot kitűnően lehet kamatoztatni.

A *kvantuminterferencia* a qubitnek, a szuperpozíciós tulajdonságuk okán kialakult viselkedése, amellyel az egyik vagy másik értékkel való egybeesés valószínűsége befolyásolható. A kvantumszámtógépek létrehozásának egyik legnagyobb kihívás, hogy az interferenciát a lehető legnagyobb mértékben csökkentsék a pontosabb eredmények érdekében [43].

E cél elérésére többféle technológiát is alkalmaznak, mindegyik esetében a cél a kvantumrészecskék stabilizációjának elérése úgy, hogy manipulálják azok szerkezetét, ezt elérhető például hűtéssel vagy azzal, hogy olyan kémiai vegyületekkel veszik körbe a kvantumrészecskéket, amelyek védik őket a külső interferenciától.

A *kvantumszámítógépeknél* a kvantumfizika speciális viselkedési formáit használja ki az informatikai számításoknál úgy, mint például a szuperpozíciót, az összefonódást és a kvantuminterferenciát. Mindezek új elgondolásokat hoznak a hagyományos algoritmusokhoz és programozási módszerekhez képest

5. Kvantumalgoritmusok

A kvantumgépek elméleti kutatásában már a 80-as években komoly eredményeket értek el. David Deutsch megalkotta a logikai kapuk kvantumgépre adaptált változatait [18], 1994-ben pedig Peter Shor publikálta az egyik leghíresebb algoritmust [60], amely a napjainkban alkalmazott rendszerek adatbiztonságát alapjaiban képes megrendíteni.

5.1. Legfontosabb kvantumalgoritmusok

A Deutsch-Jozsa algoritmus [16] volt az első olyan kvantumalgoritmus, amely azon túl, hogy rávilágított a kvantumalgoritmusokban rejlő lehetőségekre, képes volt egy bit-paritás függvény klasszikus algoritmusoknál gyorsabban történő meghatározására. Az ismeretlen logikai függvény bementként egyetlen bitet vár, és egyetlen bitet ad vissza úgy, hogy vagy minden bemeneti bit páros, vagy min-

den bemeneti bit páratlan. Az algoritmus segítségével egyetlen kvantummméréssel meghatározható az ismeretlen függvény jellege.

A Bernstein-Vazirani algoritmus [21] az előzőtől (Deutsch-Jozsa algoritmus) összetettebb problémák, függvények vizsgálatára ad lehetőséget. Az f függvény a következő: f(x) = a_1x_1 + a_2x_2 + ... + a_n*x_n + b, ahol x az n-bit hosszú bemenet, a_i az x_i-re vonatkozó súly, és b egy konstans. Az algoritmus segítségével egyetlen kvantummméréssel meghatározható az a_i és b értéke.

Grover-algoritmus [30] rendezetlen sorozatok esetében egy optimalizált keresési algoritmus, amely 2^(n/2)-szer gyorsabb, mint a klasszikus keresési eljárások.

Shor-algoritmus [60] tulajdonképpen a faktorizáció optimális algoritmusa.

Simon-algoritmus[15] bit-paritás meghatározására használt algoritmus, amely a klasszikus algoritmusoknál jóval alacsonyabb energiafogyasztással és rövidebb idő alatt képes megoldani a feladatot.

Harrow-Hassidim-Lloyd (HHL) algoritmus [32]: a lineáris egyenletrendszerek megoldására alkalmazott algoritmus, amely kisebb memóriaigény mellett, sokkal gyorsabb, mint a hasonló feladatra használt klasszikus algoritmusok.

A VQE-algoritmus [14] a kvantummechanika alapján alkalmazott hibrid algoritmus, amelynek segítségével optimális egyensúlyi állapotokat lehet meghatározni.

5.2. Részletesebben a Shor-algoritmusról

A jelenleg széles körben elterjedt titkosítási algoritmusok jelentős részben a prímtényezős felbontás[2] jelentette matematikai problémán alapulnak. A védelmet az eljárás során keletkező hatalmas prímszámok szorzata nyújtja, melynek ismeretében az azt alkotó prímszámok kiszámítása a hagyományos számítástechnikai környezetben gyakorlatilag kilátástalan.

Shor kvantum algoritmusa ezt a problémát oldja meg: egy szám prímtényezős felbontását végzi el rendkívül nagy sebességgel, így alkalmazása a prímtényezős felbontáson alapuló algoritmusokkal védett informatikai rendszerek egy részét védtelenné teszi.[3] Bár egyes esetekben a működési paraméterek, pl. kulcsméretek módosításával maga az algoritmus használható marad, a gyakorlatban ezeket a módosításokat gyakran csak az egyes rendszerek fejlesztői képesek elvégezni. A ma legelterjedtebb kriptoalgoritmusok védelmi képességeit a poszt-kvantum[4] világban az alábbi táblázat írja le.

[2] Egy szám prím, amennyiben 1-en és önmagán kívül nincs más osztója. Prímszám pl. a 17. A prímtényezős felbontás egy olyan matematikai művelet, mely egy számot prímszámok szorzatára alakít, pl. a 42 prímtényezős felbontása 2x3x7.

[3] Shor algoritmusa már egy hatqubites kvantumgépen működtethető, ilyen gép már évek óta rendelkezésre áll.

[4] A poszt-kvantum azt az informatikai korszakot jelenti, melyben a kvantumszámítógépek alkalmazásának lehetőségével már számolni kell.

Algoritmus	Alkalmazási terület	Fenntarthatóság
AES	Titkosítás	nagyobb kulccsal biztonságos
SHA-2, SHA-3	Lenyomatképzés	Hosszabb kimenet szükséges
RSA	Digitális aláírás, kulcs egyeztetés	Nem biztonságos
ECDSA, ECDH	Digitális aláírás, kulcscsere	Nem biztonságos
DSA	Digitális aláírás, kulcscsere	Nem biztonságos

3. táblázat: A titkosítási algoritmusok alkalmazhatósága a poszt-kvantum világban. Szerkesztette Koczka Ferenc.

A Shor algoritmus csak egy példa a kvantumgép paradigmaváltó hatására, olyan algoritmusokat kell találni és a jelenlegi rendszerekbe is beépíteni, amelyek képesek ellenállni kvantumgép hatalmas számítási teljesítményének. Az Amerikai Nemzeti Szabványügyi Hivatal (NIST) Számítógépbiztonsági Központja által folytatott kutatás harmadik fázisában több kvantumbiztos algoritmus is publikálásra került, melyek megoldást nyújtanak a hash lenyomatok képzéstől az elektronikus aláírásig. A problémakör kezelését a magyar jogalkotás is megkezdte, ennek eredményeként a 2013 évi L. törvénybe[5] is bekerült a poszt-kvantum algoritmusok alkalmazására való felkészülés követelménye az állami és önkormányzati szervek számára. Bár számos szervezet, így a gazdasági vállalkozások és az oktatási intézmények sem tartoznak a törvény hatálya alá, a tulajdonosi-fenntartói köröknek a jövőben számolniuk kell a rendszereik esetleges kitettségével.

A jelenlegi fejlesztések ellenére kevéssé valószínű, hogy a kvantumgépek ki tudnák váltani a ma használatos számítógépeket. A technikai nehézségeken túl alkalmazási körük rendkívül szűk, a hétköznapi és a gazdasági élet szoftvereinek futtatására nem képesek. A rendkívül felgyorsult fejlesztési folyamat azonban nyilvánvalóvá teszi, a technológia megjelenését a programozók, a fizikusok képzésében és az általános oktatásban is.

6. Oktatás

Nem csak a híreket figyelő emberek számára lényeges tájékozódni ezekről az újszerű számítógépekről, hanem célszerű elgondolkodnunk azon, hogy miként lenne érdemes behozni a témakört az oktatás meghatározott szintjeire és szakterületeire annak érdekében, hogy a kvantumszámítógépek megjelenésére és az általuk generált változásokra a társadalmunk fel legyen készítve. Legyen elég szakember a kvantumgépek működtetésére, az általuk esetlegesen okozott károk megelőzésére, kezelésére, a kvantum-adatbiztonság megőrzésére és a kvantumgépek támogatásával végezhető kutatások, fejlesztések véghezvitelére. A kvantumszámítógépek építése kapcsán például a fizikus-, az adatvédelem, a kutatások és fejlesztések kapcsán a programozóképzés erősen érintett.

[5] Ez a törvény az állami és önkormányzati szervek elektronikus információbiztonságát foglalja keretekbe, az ehhez kapcsolódó végrehajtási rendelet a 41/2015 BM Rendelet, mely a törvénnyel kapcsolatos konkrét teendőket definiálja.

Gondolni kell a szakemberképzések előkészítésére is, emiatt néhány fogalmat talán már a közoktatás felsőbb évfolyamain érdemes bevezetni. Fontos, hogy a fiatal korosztály tanulói ne legyenek magukra hagyva kérdésekkel a fejükben, hanem a tanárok által előszűrve, rendszerbe szedve és emészthető formába öntve, hiteles forrásból tájékozódhassanak e terület érdemi tartalmáról és tisztában legyenek az összefüggésekkel.

6.1. El vagyunk késve?

Függetlenül, hogy jelenleg hol tart a kvantuminformatika, az oktatás területén, elkéstünk már? A kérdésbe kódolt állítás túl erős lenne, de abban az esetben igaz, ha az elmúlt 15 évben megfigyelhető fejlődési trend ugyanilyen intenzitással folytatódik tovább e területen. Természetesen tudjuk, hogy a kvantuminformatika nem 15, hanem közel 70 éves múltra tekint vissza, a 2000-es évek közepén viszont valamilyen jelentős előrelépésnek kellett történnie. Hogy pontosan mi volt ez, azt nem célunk, és nem is tisztünk fejtegetni. Fogalmazzunk úgy, hogy valami olyan eseménysorozat, ami azt indukálta, hogy napjainkban már akkora a verseny a különböző technológiai óriások [11] között ezen a területen, hogy szinte hetente-havonta jelennek meg cikkek, hírek az újabb fejlesztésekről és eredményekről. Vizsgáljuk meg az okokat, miért van az a félelmünk, hogy valamiről már lekéstünk!

Egyik oka, amit már korábban is említettünk, hogy az informatika ezen új területének fejlődését tekintve az utóbbi 15 évben exponenciális robbanásnak lehettünk szemtanúi. Ismerve a klasszikus oktatási rendszereket, legalább egy évtized, mire az általános iskolától kezdve, egészen a felsőoktatásig, az életkori sajátosságoknak, és a célcsoportnak megfelelő tananyagok − alsóbb évfolyamokon a témával való érintettség, alapozás − megjelennek, illetve beépülnek a tantervekbe. Ami a tananyagba való beépítést még tovább nehezíti, az a következő ok: az informatikai ismereteken kívül ez a terület, magas szintű kémiai, kvantumfizikai és matematikai ismereteket, illetve az eddigiektől eltérően egy sokkal komplexebb problémamegoldási képességet igényel [3]. Problémának tartjuk azt, hogy a felsőoktatás utóbbi években végzett informatikatanárai nem lettek felkészítve ennek az új területnek a fogadására és a jövő nemzedéknek való továbbadására.

Hogy hol tart most a kvantumszámítástechnika? Napjainkban számos kvantumszámítógép [34, 67], illetve szimulátor [13] elérhető, kipróbálható, illetve programozható [35]. A különböző elven működő kvantumszámítógépek szemléltetésére rengeteg online kurzus [12, 20, 65], tutorial [42, 51, 67], illetve animáció érhető el.

6.2. Általános iskola

A digitális átalakulással az informatika az élet szinte minden területét átszövi, a gyerekek egyre gyakrabban találkoznak olyan fogalmakkal, mint a beágyazott rendszerek, a big data, az IoT, a mesterséges intelligencia vagy a kvantuminformatika. Néhány gondolatébresztő kérdést szeretnénk feltenni a korosztályra vonatkozóan:

- Meg kell-e jelennie az általános iskolában a kvantuminformatikai ismereteknek alapozásának?
- Életszerű, hogy az általános iskolában beszéljünk erről a területről?
- Mit jelent ennek a korosztálynak a kvantuminformatika?

A média már napi szinten tesz említést valamilyen színezettel a kvantuminformatikáról. Sajnos elég gyakran negatív aspektusból mutatja be úgy, mint egy, az egész életünket megváltoztató, illetve befolyásoló technológiát, amelynek segítségével a kiberbűnözők képesek akár a bankszámlákat védő számítógépes rendszereket feltörni. Az ilyen típusú, negatív tartalmú hírek belső feszültséget kelthetnek és kérdéseket indukálhatnak a gyerekekben, amelyekre megnyugtató és kimerítő válaszokat várnak. Úgy gondoljuk fel kell készíteni az informatikatanárokat az ilyen jellegű kérdésekre, fontos, hogy tájékozottak legyenek és szakszerű válaszokat tudjanak adni ezekre. Nem csak a negatív érzelmű kérdésekre kell válaszokat adniuk, hanem azt is el kell tudniuk mondani, hogy a kvantumszámítógép új lehetőségeket teremt például a biztonságos adatátvitelre, új utakat nyit a szimulációk terüle-

tén, forradalmasíthatja a gyógyszerkutatást, új anyagok fejlesztését, pontosabbá és hosszabb átvon érvényessé teheti a meteorológiai előrejelzéseket stb. A kvantumtechnológia tehát egyszerre lehetőség és kockázat a társadalom számára [10, 58].

Az informatikaoktatás elsődleges célja, hogy a digitális világban történő eligazodáshoz a megfelelő alapokat megteremtse. Az informatika oktatással kapcsolatos kutatásokban egyetértés van abban, hogy a rövidtávon használatos technológiák ismertetése helyett, nagyobb hangsúlyt kell fektetni az alapfogalmak, az algoritmusok, illetve az alapelvek megismertetésére. [2, 48, 64].

Úgy gondoljuk, hogy napjaink informatikatanárának a kvantuminformatika alapjaival tisztában kell lenniük, azért, hogy a közoktatásban tanuló gyerekeket megfelelően tudják tájékoztatni e terület fejlődési irányairól; elkerülhetetlen tehát, hogy az informatikatanárok egyetemi képzésének része legyen.

Azt is elkerülhetetlennek tartjuk, hogy idővel az általános iskolai tananyag része legyen érintőlegesen a felső tagozaton a kvantuminformatika. Ez a korosztály már nap mint nap szembesül a „kvantum" kifejezéssel a médiából, nem csak a kvantumszámítógéppel, hanem például a kvantummobilokkal, a kvantumtelevíziókkal stb. Számukra el kellene mondani azt, hogy egyáltalán mit takar a kvantum szó, milyen fizikai jelenségekre épülhetnek kvantumszámítógépek, segíteni lehet őket abban, hogy el tudják képzelni a működésüket, megértsék ezek célját, megismerjék a kvantumbit fogalmát, érzékeltetni tudjuk velük ennek mértékét és az adattárolásban, számításokban rejlő lehetőségeit, még akkor is, ha egy szakterület felnőtt tanárának sem feltétlenül könnyű ezek megértése.

Mégis fontosnak érezzük, hogy a fogalmakkal találkozzanak és tudják, hogy mire utal a kvantum szó. Sőt, cél volna az is, hogy képesek legyenek felismerni, megszűrni a reklámokban megjelenő tartalmakat, látni, hogy a reklámozott termékekben sokszor nem valódi kvantumszámítógépek vannak, hanem csupán egyfajta „kvantumos" jellemzővel rendelkező készülékekről beszélhetünk. Mint ahogyan az a porszívó sem rendelkezik mesterséges intelligenciával, amely tanulási folyamaton nem esik át, csak infravörös szenzort használva tud az akadály előtt pár centivel megállni. Gondolkodni kell megtanítani a gyerekeket ezekről a dolgokról.

6.3. Középiskola

Míg a 10-14 éves korosztály számára csak az alapfogalmak ismertetésére van lehetőség, addig a középiskolás korosztály számára már egy szinttel mélyebb, elméleti háttérrel megalapozott ismeretkör nyújtható. Az elméleti alapokkal lefektetett tudás átadásának elengedhetetlen feltétele a megfelelő matematikai, illetve kvantumfizikai háttér. Számos tanulmány megerősíti, hogy a kvantuminformatika bevezetése az általános- és középiskolás tananyagba nem csak lehetséges, de szükséges is [3, 25, 33, 45, 47, 57, 58, 67].

Stadermann és munkatársai [61] különböző nézőpontok szerint elemezték 15 ország kvantumfizika tantervét és tanmenetét. A középiskolai szintű kvantumfizika tárgyak nemrég jelentek csak meg a nemzeti tantervekben, beépítésük közel sem volt zökkenőmentes, gyerekcipőben járnak [3]. A fontosabb tartalmi elemek célja, hogy a tanulók betekintést nyerjenek a modern fizikába és alkalmazásaiba, valamit képessé tegye őket e tudományág természetének és aspektusainak megvitatására. A vizsgált országok tanterveinek közös elemei: diszkrét atomi energiaszintek, a fény és az anyag közötti kölcsönhatások, hullám-részecske dualitás, de Broglie-elmélet, Planck-hullámhossz, műszaki alkalmazások, Heisenberg-féle határozatlansági elv, és a kvantumfizika természete is helyet kapnak [40]. A kihívást jelentő részeket, mint például a kvantumfizika interpretációit, illetve ismeretelméleti aspektusait csak néhány országban tanítják. Általános tapasztalat, hogy az elkészített tantervek még gyerekcipőben járnak, így nem feltétlenül a legjobb tantervek. A jelenlegiek zöme megegyezik egy egyszerűsített egyetemi kvantummechanika kurzus elemeivel. A tantervi innovációk időigényesek, a nemzeti szabályok kidolgozása és módosítása pedig egy összetett, bonyolult folyamat [25]. Olyan

elemek [5, 44, 38, 70], amelyek elősegítenék a megértést, mint pl. a kvantumfizika filozófiai vonatkozásai, a kvantumösszefonódás és alkalmazásai, csak a norvég és a német tantervekben szerepelnek.

Anastasia Perry és munkatársai [47] egy tíz fejezetből álló tananyagot készítettek 15-18 éves középiskolások számára, amelyet egy ötnapos kurzus keretében ismertettek velük. A kurzus célja a középiskola és az egyetem közötti kapcsolatteremtés volt. A tananyagot a kvantuminformatika kulcsfontosságú alapfogalmai köré szervezték, melyben nevezetesen a következőkről volt szó: szuperpozíció, kvantummérés és összefonódás. A tananyagban az alapfogalmaktól kezdve a kvantumkapukon és a kvantumalgoritmusokon keresztül eljutnak egészen kvantum-alapú teleportálásig. Fontos megjegyezni, hogy nem feltételezik, az elektromosság, a mágnesesség és a hullámok magas szintű ismeretét, és számítógépes programozási tapasztalat sem szükséges. A kurzus elvégzése előtt és után is kérték a résztvevő diákoktól, hogy soroljanak fel minél több olyan kvantummechanikai, kvantuminformatikai fogalmat, amely eszükbe jut. Azt tapasztalták, hogy a diákok azon túl, hogy jelentősen javítottak a kurzus elején írt felméréshez képest, olyan motivációt kaptak, amely a visszajelzések alapján a fizika és az informatika irányába tereli őket [33].

Pashaei és munkatársai egy olyan online elérhető tananyagot készítettek, amely segítségével hatékonyan bevezethető a kvantuminformatika alapkoncepciói már általános és középiskolai szinten. Azon túl, hogy egy jól használható online tananyagot készítettek, meg is kongatták a vészharangot Kanadában. A kvantummechanika fogalmai nem részei a mindennapi életnek, holott ezek megismerése a tanulókra pozitív hatással lehet motivációs szempontból. Kiemelik, hogy a kvantumfizika és a kvantuminformatika az oktatás korai szakaszában történő bevezetése egy olyan társadalom kialakulásához járulhat hozzá, amely megérti a tudomány fontosságát, illetve hozzá tud járulni a fejlődéséhez, és ezzel az élvonalba emeli azt [45].

A kvantumszámítástechnika területe már kiforrott, és a diákok számára elérhető. Angara és munkatársai kvantuminformatikai workshopok eredményeiről számoltak be. Elsődlegesen olyan diákoknak tartottak rövid workshopokat, akiknek a kvantuminformatikával kapcsolatban semmilyen előismeretük és tapasztalatuk nem volt. A programozásalapú megközelítést választották, a Qiskit-en [51] keresztül vezették be a diákokat az IBM Q Experience világába. Tapasztalataik alapján megállapították, hogy a kvantuminformatikai fogalmak középiskolás diákok számára is érthetők, feldolgozhatók. Ők is kiemelik, hogy tapasztalataik szerint a kvantuminformatikával történő korai megismerkedés fejleszti a diákok problémamegoldó képességét, bővíti és hozzájárul az egyetemre történő belépés előtt megszerzendő kompetenciákhoz [3].

7. Konklúzió

Összefoglalva nagyon jó kérdés tehát az, hogy kell-e egyáltalán komolyan foglalkozni ezzel a területtel? Állíthatjuk, hogy hamarosan már része lehet az oktatásnak, akár az általános iskolás oktatásnak érintőlegesen, és a középfokú oktatásnak a fogalmak bevezetésének és a szimulációs kvantumgépek használatának szintjén, hiszen külföldön már látunk erre is jó pár példát?

Jelenlegi ismereteink alapján egyelőre az jelezhető előre, hogy a kvantumszámítógépek nem fogják leváltani a hagyományos, Neumann-elvű számítógépeket, de az elmúlt másfél évtized eredményei [4, 9, 63], illetve a tech-óriások [11] és a nemzeti kormányok jövőbeli tervei [26, 29] alapján szinte biztosan állítható, hogy az elkövetkezendő tíz évben ez egy már szabad szemmel is jól látható szegmense lesz az informatikának.

Nyilvánvaló, hogy mint minden irányzatnak, ennek is vannak olyan követői, illetve kutatói, akik jelentős fenntartásokkal kezelik az eredményeket és a jövőképük közel sem tekinthető bizakodónak. Véleményük szerint lehetetlen általános célú, akár csak a jelenlegi számítási kapacitás eléréséhez szükséges qubittel rendelkező, elfogadható hibahatáron belül dolgozó kvantumszámítógépet készíteni [39].

Mindezek tükrében azt tudjuk mondani, hogy érződik, és érthető az oktatás területén a kísérletezési kedv és a bizonytalanság egyaránt. De úgy látjuk, ha a kvantumszámítógépek megrekednek a jelenlegi szinten, azaz csak jól specifikált részfeladatok és valódi véletlenszámok generálására lesznek alkalmasak, már akkor is bevethetők a problémamegoldó képesség fejlesztésére, a látókör bővítésére, az időkomplexitás fogalmának mélyítésére, a fizika népszerűsítésére vagy a kvantumfizika megértésének elősegítésére [19].

8. Irodalom

1. Amdahl, G. M.: *Validity of the single-processor approach to achieving large scale computing capabilities*. In AFIPS Conference Proceedings vol. 30 (Atlantic City, N.J., Apr. 18-20). AFIPS Press, Reston, Va., (1967) 483–485
2. Andreas Schwill. 1997. Computer science education based on fundamental ideas. In Information Technology. Springer, 285–291.
3. Angara, P. P., Stege, U., & MacLean, A. (2020, October). Quantum Computing for High-School Students An Experience Report. In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE. (2020) 323–329
4. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., ... & Martinis, J. M.: *Quantum supremacy using a programmable superconducting processor*. Nature, 574(7779), (2019) 505–510.
5. B. C. Grau, How to teach basic quantum mechanics to computer scientists and electrical engineers, IEEE Trans. Ed. 47, 220 (2004).
6. Bauman NP., Liu H., Bylaska EJ. (et all):*Toward quantum computing for high-energy excited states in molecular systems: quantum phase estimations of core-level states* In Journal of Chemical Theory and Computation (2021) 201–210
7. Berg, H.: *Historical roots of bioelectrochemistry* Experientia 36 (1980) 1247–1249
8. Berg, H., Richter K., Ritter J W.: *Entdeckungen zur Elektrochemie, Bioelektrochemie und Photochemie* In Ostwalds Klassiker der Exakten Wissenschaften, Bd 271. Leipzig, Harri Deutsch; 2., Aufl. edition (1997)
9. Boixo, S., Isakov, S. V., Smelyanskiy, V. N., Babbush, R., Ding, N., Jiang, Z., ... & Neven, H.: *Characterizing quantum supremacy in near-term devices*. In Nature Physics, 14(6), . (2018) 595–600
10. Cao, Y., Romero, J., & Aspuru-Guzik, A. (2018). Potential of quantum computing for drug discovery. IBM Journal of Research and Development, 62(6), 6-1.
11. Chakraborty M.: *Top 10 Quantum Computing Companies to Watch Out in 2021* In Analytics Insight (2021) https://www.analyticsinsight.net/top-10-quantum-computing-companies-to-watch-out-in-2021/ (utoljára megtekintve: 2022.11.11.)
12. Class Centra Portal: *Quantum Computing Courses*. https://www.classcentral.com/subject/quantum-computing (utoljára megtekintve: 2022.11.20.)
13. Dargan J.: *Top 63 Quantum Computer Simulators For 2022*. The Quantum Insider Portal (2022) https://thequantuminsider.com/2022/06/14/top-63-quantum-computer-simulators-for-2022/ (utoljára megtekintve: 2022.11.20.)
14. Colless, J. I., Ramasesh, V. V., Dahlen, D., Blok, M. S., Kimchi-Schwartz, M. E., McClean, J. R., ... & Siddiqi, I. (2018). Computation of molecular spectra on a quantum processor with an error-resilient algorithm. Physical Review X, 8(1), 011021.
15. Daniel R. Simon (1997) "On the Power of Quantum Computation" SIAM Journal on Computing, 26(5), 1474–1483,
16. David Deutsch and Richard Jozsa (1992). "Rapid solutions of problems by quantum computation". Proceedings of the Royal Society of London A. 439: 553–558.
17. Dennard, Robert H.; Gaensslen, Fritz; Yu, Hwa-Nien; Rideout, Leo; Bassous, Ernest; LeBlanc, Andre: *Design of ion-implanted MOSFET's with very small physical dimensions*. IEEE Journal of Solid-State Circuits. SC-9 (5) (1974) 256–268

18. Deutsch, David: *Quantum theory, the Church–Turing principle and the universal quantum computer.* Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences 400.1818 (1985) 97–117
19. Dyakonov, M.: *When will useful quantum computers be constructed? Not in the foreseeable future, this physicist argues. Here's why: The case against: Quantum computing.* Ieee Spectrum, 56(3), (2019) 24–29
20. Edx Portal: https://www.edx.org/learn/quantum-computing (utoljára megtekintve: 2022.11.20.)
21. Ethan Bernstein and Umesh Vazirani (1997) "Quantum Complexity Theory" SIAM Journal on Computing, Vol. 26, No. 5: 1411-1473
22. Európai Parlament*: A digitális évtizedre vonatkozó uniós kiberbiztonsági stratégia (2022/C 67/08)* (2022) https://eur-lex.europa.eu/legal-content/HU/TXT/HTML/?uri=OJ:C:2022:067:FULL&from=EN (utoljára megtekintve: 2022.11.11.)
23. Európai Parlament: *Az Európai Parlament 2022. május 3-i állásfoglalása a digitális korban a mesterséges intelligenciáról* (2022) https://www.europarl.europa.eu/doceo/document/TA-9-2022-0140_HU.html (utoljára megtekintve: 2022.11.11.)
24. Európai Tanács: *A Tanács (EU) 2022/576 Rendelete*, I. melléklet (2022) https://eur-lex.europa.eu/legal-content/HU/TXT/HTML/?uri=OJ:L:2022:111:FULL&from=HU (utoljára megtekintve: 2022.11.11.)
25. Fullan Michael: *Change Forces: Probing the Depth of Educational Reform* (Falmer Press, London, UK, 1993).
26. Gidney, C., & Ekerå, M.: *How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*. Quantum, 5, 433. (2021)
27. Gordon E. Moore: *Cramming more components onto integrated circuits*. Electronics, Volume 38, Number 8, April 19 (1965) https://hasler.ece.gatech.edu/Published_papers/Technology_overview/gordon_moore_1965_article.pdf (2013) (utoljára megtekintve: 2022.11.11.)
28. Gesche Pospiech: *Teaching the EPR paradox at high school?*, Phys. Educ. 34, 311 (1999).
29. Gouzien, E., & Sangouard, N.: *Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory*. Physical Review Letters, 127(14), 140503 (2021)
30. Grover, L. K. (1996, July). A fast quantum mechanical algorithm for database search. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (pp. 212-219).
31. Harrison J., Sellars MJ., Manson NB.: *Measurement of the optically induced spin polarisation of N-V centres in diamond* In Diamond and Related Materials, Volume 15, Issues 4–8, April–August (2006) 586–588
32. Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. Physical review letters, 103(15), 150502.
33. Hughes, C., Isaacson, J., Turner, J., Perry, A., & Sun, R. (2022). Teaching quantum computing to high school students. The Physics Teacher, 60(3), 187-189.
34. IBM Quantum Portal: *Real quantum computers.* https://quantum-computing.ibm.com/ (utoljára megtekintve: 2022.11.20.)
35. IBM Quantum Portal: *IBM Quantum Composer.* https://quantum-computing.ibm.com/composer/docs/iqx/ (utoljára megtekintve: 2022.11.20.)
36. Jim X. Chen: *The Evolution of Computing: AlphaGo.* Computing in Science & Engineering Volume: 18, Issue: 4, July-Aug. (2016) https://ieeexplore.ieee.org/abstract/document/7499782 (utoljára megtekintve: 2022.11.11.)
37. Kloeffel C., Loss D.: *Prospects for spin-based quantum computing in quantum dots* In Annual Review of Condensed Matter Physics, Vol. 4:51-81 (Volume publication date April 2013) (2013)
38. Kohnle, A., Bozhinova, I., Browne, D., Everitt, M., Fomins, A., Kok, P., ... & Swinbank, E.. A new introductory quantum mechanics curriculum. European Journal of physics, 35(1), 015001 (2013).
39. Landauer, R.: *Irreversibility and heat generation in the computing process*. IBM journal of research and development, 5(3), (1961) 183–191

40. Lobato, T., & Greca, I. M. (2005). Quantum Theory contents insertion in High School curricula. Ciência & Educação (Bauru), 11, 119-132.
41. Magyarország Kormánya: *2013. évi L. törvény az állami és önkormányzati szervek elektronikus információbiztonságáról* https://net.jogtar.hu/jogszabaly?docid=a1300050.tv (2013) (utoljára megtekintve: 2022.11.11.)
42. Microsoft Portal: *Tutorial: Explore quantum entanglement with Q#.* https://learn.microsoft.com/en-us/azure/quantum/tutorial-qdk-explore-entanglement?pivots=ide-azure-portal (utoljára megtekintve: 2022.11.20.)
43. Microsoft Portal: *Introduction to quantum computing.* https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-quantum-computing/#introduction (utoljára megtekintve: 2022.12.20.)
44. Nayak, T., & Dash, T. (2012). A comparative study on quantum pushdown automata, turing machine and quantum turing machine. International Journal of Computer Science and Information Technologies, 3(1), 2932-2935. (angolból idézve: „Quantum is a discrete quantity of energy proportional in magnitude to the frequency of radiation it represents."
45. Pashaei, P., Amiri, H., Haenel, R., Lopes, P. L., & Chrostowski, L. (2020, October). Educational Resources for Promoting Talent in Quantum Computing. In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE) (pp. 317-322). IEEE.
46. Perez-Garcia B., Francis J., McLaren M. (et. all): *Quantum computation with classical light: The Deutsch Algorithm* In Physics Letters A, Volume 379, Issues 28–29, 28 August (2015)
47. Perry, A., Sun, R., Hughes, C., Isaacson, J., & Turner, J. (2019). Quantum computing as a high school module. arXiv preprint arXiv:1905.00282.
48. Peter J Denning. 2004. Great principles in computing curricula. In Proceedings of the 35th SIGCSE technical symposium on Computer science education. 336–341.
49. Planck, M. Ueber das princip der vermehrung der entropie. Annalen der Physik, 267(6), (1887) 189–203
50. Planck, M. *The theory of heat radiation. Entropie* 144(190), 164. (1900)
51. Qiskit Portal: *Tutorials.* https://qiskit.org/documentation/tutorials.html (utoljára megtekintve: 2022.11.20.)
52. Redaktor: *Hatályosak az állami és önkormányzati szervek elektronikus információbiztonságáról szóló 2013. évi L. törvény kvantumtitkosításra vonatkozó szakaszai*. EGOV Közigazgatás és Informatika (2022) https://hirlevel.egov.hu/2022/07/10/hatalyosak-az-allami-es-onkormanyzati-szervek-elektronikus-informaciobiztonsagarol-szolo-2013-evi-l-torveny-kvantumtitkositasra-vonatkozo-szakaszai/ (utoljára megtekintve: 2022.11.11.)
53. Reddy, P. P. (2020). Quantum Generators: A Formulation of Computational Models of Multiplication. Google Scholar.
54. Ritter J. W.: *Entdeckungen zur Elektrochemie, Bioelektrochemie und Photochemie* Leipzig, 1986, 135 p.,figuras. Encuadernacion original. Nuevo.
55. Ritter J. W.: *Key texts of Johann Wilhelm Ritter* on the science and art of nature (Vol. 16). Brill. (2010) 1776–1810
56. Robert X. Cringely: *Breaking Moore's Law.* BetaNews. https://betanews.com/2013/10/15/breaking-moores-law/ (utoljára megtekintve: 2022.11.11.)
57. Satanassi, S., Fantini, P., Spada, R., & Levrini, O. (2021, May). Quantum Computing for high school: an approach to interdisciplinary in STEM for teaching. In Journal of Physics: Conference Series (Vol. 1929, No. 1, p. 012053). IOP Publishing.
58. Seegerer, S., Michaeli, T., & Romeike, R. (2021, October). Quantum computing as a topic in computer science education. In The 16th Workshop in Primary and Secondary Computing Education (pp. 1-6).
59. Sejuti Dast: *Top Applications Of Quantum Computing Everyone Should Know About.* (2020), https://analyticsindiamag.com/top-applications-of-quantum-computing-everyone-should-know-about/ (utoljára megtekintve: 2022.11.11.)

60. Shor, P.W.: *Algorithms for quantum computation: discrete logarithms and factoring*. Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc (1994) 124–134
61. Stadermann, H. K. E., van den Berg, E., & Goedhart, M. J. (2019). Analysis of secondary school quantum physics curricula of 15 different countries: Different perspectives on a challenging topic. Physical Review Physics Education Research, 15(1), 010130.
62. Terhal, B. M.: *Quantum supremacy, here we come* Nature Physics, 14(6), (2018) 530–531
63. The White House – National Quantum Coordination Office: *Quantum Frontiers Report on Community Input tot he Nation's Strategy for Quantum Inormation Science* (2020) https://www.quantum.gov/wp-content/uploads/2020/10/QuantumFrontiers.pdf (utoljára megtekintve: 2022.11.11.)
64. Tim Bell, Paul Tymann, and Amiram Yehudai. 2011. The Big Ideas of K-12 Computer Science Education
65. Udemy Portal: *The Complete Quantum Computing Course* https://www.udemy.com/course/quantum-computers (utoljára megtekintve: 2022.11.20.)
66. Wikipedia commons: Amdahl'sLAw. https://commons.wikimedia.org/wiki/File:AmdahlsLaw.svg (utoljára megtekintve: 2022.11.11.)
67. Wootton, J. R., Harkins, F., Bronn, N. T., Vazquez, A. C., Phan, A., & Asfaw, A. T. (2021, October). Teaching quantum computing with an interactive textbook. In 2021 IEEE International Conference on Quantum Computing and Engineering (QCE) (pp. 385-391). IEEE.
68. Xanadu Portal: *Quantum computational advantage on Xanadu Cloud*. https://www.xanadu.ai/ (utoljára megtekintve: 2022.11.20.)
69. Zurich Instruments Portal: *Qubit Control*. https://www.zhinst.com/ (utoljára megtekintve: 2022.11.20.)
70. Dür Wolfgang, and Heusler Stefan: *Visualization of the invisible: The qubit as key to quantum physics*, Phys. Teach. 52, 489 (2014).
71. Online Etymologie Dictonary: *Quantum* notion. https://www.etymonline.com/ (utoljára megtekintve: 2022.12.20.)

Támogató

A kutatást az Innovációs és Technológiai Minisztérium és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatta a Kvantuminformatika Nemzeti Laboratórium keretében.

Játékkal tanulni, tanulással játszani: tapasztalatok az első középiskolai élményinformatikai táborunkról

Dienes-Gulácsi Nikolett[1], Gulácsi Ádám[2]

{[1]dienes.nikolett, [2]gulacsi.adam}@diosgyorigimnazium.hu
Diósgyőri Gimnázium, Miskolc

Absztrakt. Pályakezdő pedagógusként kifejezetten hamar szembesülni lehet az oktatás egyik nehézségével napjainkban. Nevezetesen, hogy az internet világában felnövekvő generáció motiválása az osztályteremben korántsem egy egyszerű feladat.

Egy olyan egyhetes tábort szerveztünk iskolánk tanulóinak, ahol élményekben gazdag, számukra eddig teljesen ismeretlen közegben fejleszthették informatikai ismereteiket. A résztvevők videójáték fejlesztés, játék tervezés (game design), unplugged feladatok, 3D modellezés és nyomtatás, drón API programozás és multimédiás tartalomgyártás témájú foglalkozásokon szerezhettek új tapasztalatokat.

A tábor célja elsősorban a diákok motiválása volt. Fontos rávilágítani arra, hogy ha valaki jó informatikai alapokkal rendelkezik, a tudomány és az élet más-más területein is hasznosíthatja azt. A digitális kultúra órákon fejlesztett számítógépes gondolkodásukat, algoritmizáló készségüket kivalóan alkalmazták a látványosabb, élménydúsabb témakörökben is a táborban.

Kulcsszavak: játékfejlesztés, játéktervezés, nyári tábor, programozás, algoritmikus gondolkodás

1. Motiváció

A diákok az informatika órákon gyakran motiválatlanok, ami jelentős hatással van a teljesítményükre is [14]. Sőt, a motiváció hiánya olyan tanulóknál is megjelenik, akik egyébként tehetségesek és érdeklődnek az informatika tudománya iránt. Az egyik fő probléma a teljesítmény alapú célorientált hozzáállás. Az ilyen beállítottságú diákok fő motivációját a jeles osztályzatok megszerzése, valamint a minél sikeresebb érettségi vizsga teljesítése adja [14]. Az effajta motiváció negatív hatással van a kognitív teljesítményükre és gyakran a Kahneman által körülírt lassú gondolkodást alkalmazzák olyan helyzetekben is, amikor gyors gondolkodás lenne indokolt [4]. A számítógépes problémamegoldás kontextusában ez azt jelenti, hogy a feladatok megoldása során felületi megközelítésű metakognitív módszereket alkalmaznak [16].

Szintén probléma a középiskolás tanulók esetében, hogy nagyon eltérő tudással érkeznek a gimnáziumi osztályokba. Az iskolákból hozott osztályzataik és a saját bevallásuk szerinti tudásszintjük sokszor nem tükrözi a valóságot [23]. Ez utóbbi a felsőoktatásban is megjelenik [17].

Érdekes módon a 2020-ban bevezetett új NAT már a 3–4. évfolyamra vonatkozóan olyan átfogó célokat fogalmaz meg, amelyek a gimnáziumi évek alatt is relevánsak:

„*1. elmélyülten dolgozik digitális környezetben, önellenőrzést végez;*

2. megvizsgálja és értékeli az általa vagy társai által alkalmazott, létrehozott, megvalósított eljárásokat;

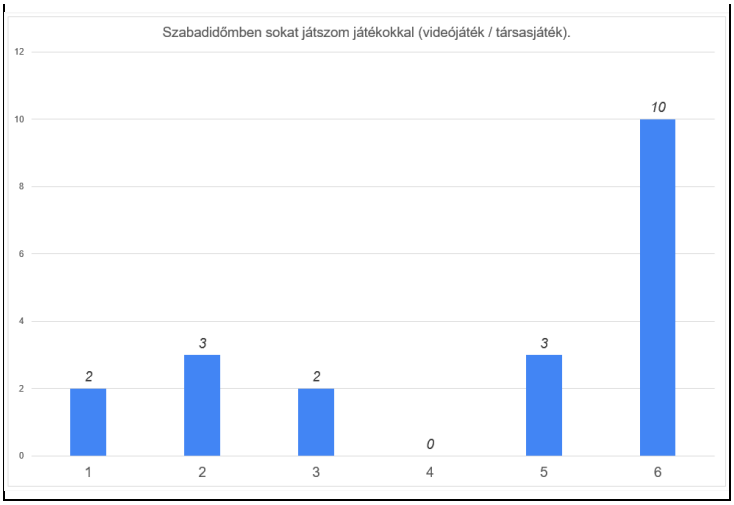
3. társaival együttműködve online és offline környezetben egyaránt megold különböző feladatokat, ötleteit, véleményét megfogalmazza, részt vesz a közös álláspont kialakításában;

4. kiválasztja az általa ismert informatikai eszközök és alkalmazások közül azokat, melyek az adott probléma megoldásához szükségesek;

5. eredményétől függően módosítja a problémamegoldás folyamatában az adott, egyszerű tevékenységsorokat." [13/141. o.]

A célunk egy olyan tábor szervezése volt, ami többek között a NAT által (is) megfogalmazott célok elérésében segíti a diákokat. Ezen felül a másik fő szempont az informatika tudományában tehetséges és érdeklődő tanulók motiválása. A jártasság alapú célorientált hozzáállásra fókuszáltunk, azaz az érettségi és a jeles osztályzatok helyett olyan célokat tűztünk ki a diákoknak, mint például „képes leszel egy játékot fejleszteni", vagy „jobban megismered a számítógépek működését" [14].

A tábor központi elemének a játékot választottuk. Munkánk során tapasztaljuk, hogy a tanulók rendszerint kapcsolatot vélnek felfedezni a videójátékok és az informatika tudásuk között. Logikájuk szerint „*én sokat játszom videójátékokkal, ezért jó vagyok informatikából*". Természetesen a kettő között az esetek többségében semmilyen összefüggés nincsen. Ettől függetlenül tény, hogy az érdeklődő tanulók jellemzően szeretnek játszani. A diákok játékok iránti szeretetét alátámasztja a tábor első napján kitöltött kérdőív is (1. ábra). Az elképzelésünk, hogy ebből a lelkesedésből motivációt kovácsolunk, aztán informatikát tanítunk és számítógépes gondolkodást fejlesztünk a videójátékok fejlesztésén (game development), illetve a játék tervezés (game design) témakörökön keresztül.



1. ábra: a tanulók 65%-a jelentős időt fordít játékokra (1=egyáltalán nem jellemző, 6=teljesen jellemző).

2. Tábor feladatai, felépítése

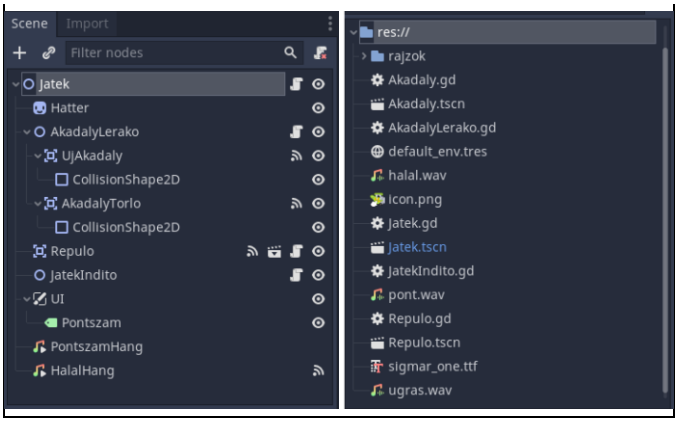
A tábor 5 nap alatt került megrendezésre. Minden nap 9:00-től 16:00-ig voltak 60 perces foglalkozások, egy-egy ebédszünettel. A beosztás tekintetében két lehetőségünk volt: vagy egy területre koncentrálunk egy nap és csak abban a témában tartunk órákat, vagy minden nap több témát érintünk, ami miatt jóval kevesebb idő jut naponta egy témára. Mi az utóbbit választottuk, abban bízva, hogy a diákok figyelme, érdeklődése jobban kitart, ha változatosabbak a programok. Ezzel a megoldással az egyes informatikai témakörök közötti tudástranszfer-elemek aktiválására is lehetőségünk nyílt.

2.1. Játfékfejlesztés

A tábor egyik központi témaköre a játékfejlesztés volt. Sajnálatos módon a középiskolákban ez nem egy kifejezetten népszerű módszer az informatika oktatására. Ezt alátámasztja a tény, hogy kevés olyan irodalom, kutatási anyag elérhető, amely a játékfejlesztés középiskolai alkalmazásainak lehetőségét taglalja [11]. Bizakodásra ad okot azonban, hogy az utóbbi néhány évben egyre több ilyen kutatás jelenik meg. Viszont ezeknek jelentős része nem a szöveg alapú programozással, hanem a vizuális, blokk-alapú nyelvekkel foglalkozik [5][18].

Ettől függetlenül számos előnye van a játékfejlesztés iskolai alkalmazásának. Az egyik az interaktivitás. A tábor ideje alatt két kisebb projektet készítettünk el. Mindkét feladat elején megmutattuk a végleges verzió demóját, mielőtt elkezdtük a munkát. A tanulók rendkívül motiváltak lettek a végeredmények láttán. A hagyományos szövegalapú programozáshoz tartozó ingerszegényebb integrált fejlesztői környezetekkel szemben az audiovizuális megjelenés és az interaktivitás jóval motiválóbb hatást gyakorolt a diákokra. Egy másik nagy előnye, hogy könnyű hosszabb távú célokat kitűzni a diákoknak, hiszen egy egyszerűbb játék elkészítése is komplex feladat. Ez is pozitív hatást gyakorol a tanulók motivációjára [14].

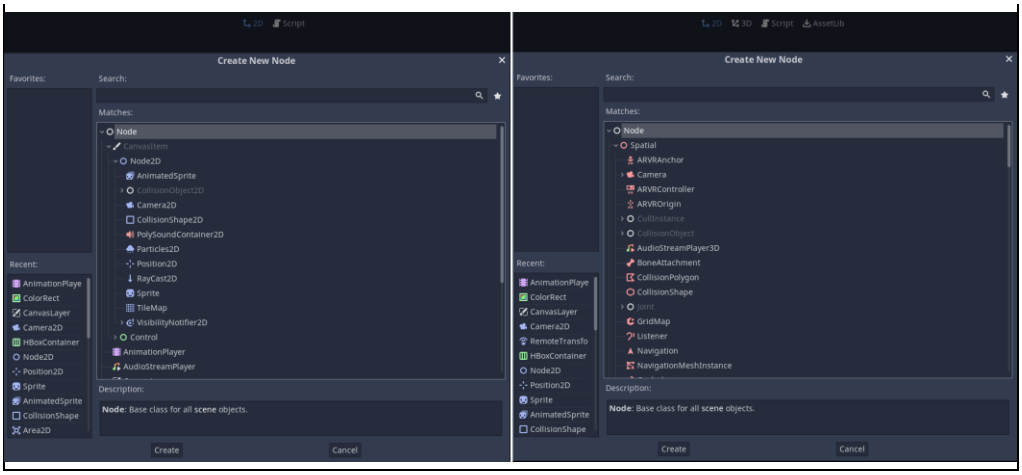
Egy kis kitérővel érdemes megemlíteni, hogy munkánk során rendszeresen tapasztalunk egy érdekes jelenséget. A diákoknak egyre nagyobb nehézséget jelent megérteni, hogy miért van szükség a számítógépes munka során a helyes fájlnevek megválasztására és a fájlok könyvtárakba történő rendszerezésére [15]. Ennek oka véleményünk szerint az egyre felhasználóbarátabb telefonos operációs rendszerekben keresendő. Vegyük például a fényképek elkészítését és megtekintését. A legnépszerűbb rendszerek Galéria alkalmazásai képesek dátum, helyszínek vagy akár emberek alapján is sorba rendezni, rendszerezni a képeket anélkül, hogy a felhasználónak bármit kellene csinálnia. Ez a mindennapi életben hasznos és kényelmes, de sajnos a számítógépes munka során ez sokszor komoly gondokat okoz. A játékfejlesztés két okból is alkalmas ennek a tanítására. Egyrészt a játékban létrehozott objektumokat is muszáj beszédesen elnevezni és rendszerezni, ha nem akarunk elveszni a játék fejlesztése során. Másrészt ez a játék objektumain kívül ugyanúgy igaz a projekthez tartozó állományokra is. Így két helyen is gyakoroltathatjuk a diákokkal a helyes fájlelnevezést és az állományok rendszerezését (2. ábra).



2. ábra: beszédes objektumnevek (bal),
és helyesen elnevezett fájlok (jobb).

A játékfejlesztéshez az ingyenes és nyílt forráskódú Godot Engine játékmotort használtuk [6]. Ez a program minden tekintetben tökéletes iskolai használatra: a legutóbbi stabil verzió fájlmérete 72 MB, ezért könnyen és gyorsan beszerezhető. Telepíteni sem szükséges, így nincs probléma a

rendszergazdai jogosultságokkal. A kis fájlméret mellett nagyon alacsony hardverigényekkel rendelkezik, így a korszerűtlenebb, régi gépeken is gond nélkül fut. Ezen kívül a játékok programozhatók több nyelven is. Az alapértelmezett választás a GDScript, ami a saját interpreteres nyelve a motornak. Ez a nyelv a leginkább kezdőbarát választás, hiszen nagyon hasonlít a Python nyelvre. Ezen kívül lehetőség van a kódolásra C, C++ és C# nyelveken is [6]. Végül egy kifejezetten oktatáshoz hasznos beállítás, hogy elrejthetjük a szerkesztő bizonyos funkcióit. A táborban például a teljes 3D részt kikapcsoltuk. Ez rengeteget segít a diákoknak, hogy a felhasználói felület ne tűnjön elsőre túl ijesztőnek (3. ábra).



3. ábra: kezdőknek szánt profil, csak a 2D elemekkel (bal),
valamint az alapértelmezett profil, benne minden 3D, VR eszközzel is (jobb).

2.1.1. Ismerkedés az alapokkal

Az első elkészített „játék” egy bevezető feladat volt, aminek a célja, hogy a diákok megismerkedjenek a játékmotor sajátosságaival. Erre azért volt szükség, mert minden felsorolt előnye ellenére a játékfejlesztés egy komplex feladat. A ma rendelkezésre álló játékmotorok nagyban megkönnyítik a helyzetet, de ez természetesen többlet feladattal is jár. A tanulóknak ugyanis meg kell ismerkedniük a játékmotor felépítésével, működési elvével is.

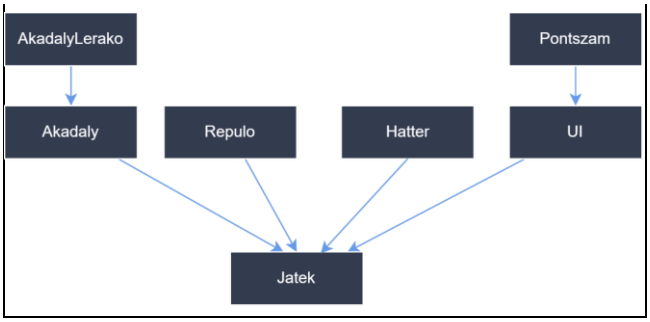
A példa működése nagyon egyszerű: a diákok előre leraknak szilárd falakat a világba, bal kattintásra pedig „bedobnak” egy labdát. A labdákra hat a gravitáció, a falak pedig megállítják a labdákat. A projekt tökéletes volt arra, hogy a tanulók megismerkedjenek a Godot Engine működésével. Ezt a feladatot a játékmotor dokumentációja ihlette, ahol található egy nagyon hasonló program, a példányosítás (instancing) fejezetben [6]. Természetesen a legtöbb diák kihasználta a lehetőséget, és tesztelte, hogy a rendszer hány labdát bír megjeleníteni mielőtt megadja magát (4. ábra). A program forráskódja elérhető, szabadon felhasználható bármilyen célra [8].

4. ábra: a diákok teletöltik labdákkal a „medencéjüket”.

2.1.2. Zuhanó repülés játék

A játékfejlesztés modul fő feladata egy Flappy Bird játék elkészítése volt. A játékot alkalmasnak találtuk kezdőprojektnek, hiszen a szabályrendszere elég egyszerű. Ennek ellenére viszont egy szórakoztató, akár mobiltelefonnal is kompatibilis, annak idején kifejezetten népszerű játékról van szó [21]. A választás utólag jó döntésnek bizonyult, a gyerekek élvezték a feladatot, elmondásaik alapján ez volt az egyik kedvenc részük a táborban. Egyébiránt a Harvard egyetem szabadon elérhető *CS50: Introduction to Game Development* nevű játékfejlesztés kurzusában is az első játékok között szerepel [2].

A Godot Engine előnyei közé tartozik még, hogy a működési filozófiájának köszönhetően könnyedén meg lehet tervezni a játék felépítését. Ha egyszerű ábrákkal leírjuk a legfontosabb objektumokat és azok kapcsolatát, gyakorlatilag meg is terveztük a játék felépítését. A diákok percek alatt össze tudtak rakni egy-egy ilyen tervet (5. ábra).



5. ábra: fontosabb objektumok a Zuhanó repülés játékban.

A játék fejlesztése során a tanulók megismerkedtek a változókkal, elágazásokkal és a ciklusokkal is. A részfeladatokat, fejlesztési fázisokat úgy választottuk meg, hogy a kognitív túlterhelés elkerülése érdekében egy foglalkozás során mindig csak egy új programozási eszköz kerüljön bevezetésre. Fontos volt ezen kívül, hogy a megtanult eszközök használatát gyakoroltassuk is velük a fejlesztés során. Ha délután már fáradtabbak voltak a tanulók, akkor a rajzok, képek beállításával és játéktesztteléssel dolgoztak (6. ábra).

6. ábra: a repülő objektum beállítása Godot Engine-ben.

A diákok által elkészített játékok elérhetők, böngészőprogramból is kipróbálhatók az iskola itch.io oldalán keresztül [3]. A játék forráskódja elérhető, szabadon felhasználható bármilyen célra [9].

2.1.3. Hangrögzítés, hangszerkesztés

A tábor során terveztünk egy multimédiás foglalkozást is. Biztosítottunk a tanulóknak stúdiómikrofont és keverőpultot, így kipróbálhatták a hangrögzítést, hangszerkesztést és a hangkeverést is. Természetesen itt is konkrét feladat elé állítottuk őket. A Zuhanó repülés játékhoz kellett hangeffekteket készíteniük.

Tanulónként legfeljebb 3 hangeffektet kellett elkészíteni: az ugrás hangját, a pontszerzés hangját, valamint a repülő „felrobbanásának" a hangját. Bíztattuk a kreativitásra a diákokat, és nem okoztak csalódást: volt, aki kulcsot, kulacsot vagy épp a cipőjét használta a hangok rögzítéséhez, de legtöbben persze a saját hangszálaikra hagyatkoztak. A rögzített hangokat az ingyenes és nyílt forráskódú Audacity nevű szoftverben rögzítettük, szerkesztettük [1].

Érdekes tapasztalat, hogy a diákok maguktól nem igazán gondolnak rá, hogy az előzetesen megszerzett számítógépes ismereteiket számukra eddig ismeretlen szoftverekben is kipróbálják, kamatoztassák. Ez abban nyilvánult meg, hogy kezdetben nehezen boldogultak az Audacity felületén. Viszont meglepő módon semmi problémájuk nem volt a szoftver kezelésével miután megjegyeztük, hogy az általuk ismert másolás, kivágás, beillesztés, törlés, kijelölés műveletek ebben a programban is ugyanúgy működnek.

Voltak olyan félénk diákok is, akik ódzkodtak a mikrofon használatától vagy éppen nem szerették volna visszahallani a saját hangjukat. Nekik biztosítottunk arra lehetőséget, hogy a Labchirp nevű, szintén ingyenes és nyílt forráskódú programmal generálhassanak nekik tetsző hangeffekteket [12].

2.2. Játéktervezés

Egy másik oszlopos témaköre a tábornak a játék tervezés (game design). Ez egy nehezen körbehatárolható terület, hiszen nemcsak játékok tervezésével foglalkozik, hanem „játékosok tapasztalatainak megalkotásával" is [10].

A foglalkozások során három témakört jártunk körbe a diákokkal:

- az érzelmek és a játékok kapcsolata,
- a játékélmény,

– a játékok célrendszere.

Ezekben nagy segítségünkre volt az oktatóknak ingyenesen felhasználható, Riot Games által kiadott játéktervezői kurzus, az *Urf Academy* [19]. Ennek a tananyagnak a moduljaiból indultunk ki, végül kisebb-nagyobb módosításokkal használtuk.

A játéktervezéses foglalkozások során kizárólag csoportmunkában dolgoztak a tanulók és szinte kivétel nélkül a felfedezéses tanulás módszerére hagyatkoztunk. A tanulói csoportokat tudatosan alakítottuk ki, minden csapatot egy-egy színt képviselt. Az első feladatuk az volt, hogy gyűjtsék ösz-sze, majd csoportosítsák, hogy a játék (bármilyen játék) milyen érzelmeket válthat ki a játékosokból (7. ábra).

7. ábra: érzelmek és játék kapcsolata.

A következőkben arról beszélgettünk, hogy mit jelent az, hogy egy játék szórakoztató (fun). A *szórakoztató* fogalom tervezői szemmel nézve haszontalan kifejezés, hiszen nem elég konkrét. Emiatt érdemes megismerkedni a Hunicke, LeBlanc & Zubek féle csoportosítással, mely szerint egy játék 8 féle módon lehet szórakoztató [20]. Miután ezekre mindenki tudott saját példákat mondani, áttértünk a játékok céljaira.

A diákok 4 fős csoportokban dolgoztak. A két összetolt asztal 4 oldalának a közepére mindenki felragasztott egy kaput. Ezután papírfocilabdát kellett hajtogatniuk, majd kezdődhetett a játék. A játékszabályok egyszerűek:

– A játék úgy kezdődik, hogy minden labda az asztal közepén van.
– Ha bármelyik labda leesik, utána az asztal közepére kerül vissza.
– A játék CÉLJA, hogy a labdáddal a másik játékos labdáit a saját kapujukba juttasd (meg is kell állnia ott).
– Minden játékosnak 3 lépése van egy körben.
– A labdát pöckölni, vagy suhintani lehet.
– A játékos köre véget ér, ha bármelyik labda leesik az asztalról.

Az első játéktesztelés célja, hogy a tanulók megismerjék a kiindulási állapotot, teljesen megértsék a játékszabályokat (8. ábra).

8. ábra: papírfoci játék.

Ezután következett az igazán érdekes része a feladatnak. A tanulók kitaláltak különböző célokat a játékhoz, a szabályok változtatása nélkül. Ezekkel kísérleteztek, tesztelték, hogy a cél megváltoztatásával hogyan változik a játékélmény. Kerestük a legszórakoztatóbb célokat, megbeszéltük, hogy mitől jó vagy rossz egy cél a papírfoci játék esetében.

Érdekességként érdemes megemlíteni, hogy az *Urf Academy* oktatóknak szóló leírásában részletezi, hogy az Egyesült Államok oktatási rendszerében megnevezett fejlesztendő attitűdök, alapkészségek közül melyikeket fedi le a képzés. Ilyen például a kooperációs képesség, íráskészség, beszédkészség, önkifejezés fejlesztése, egy szakterület területspecifikus szókincsének magabiztos használata vagy akár komplex valós problémák megoldásának megtervezése, a feladat több, kisebb problémára bontásával [19]. Ezek nagyon hasznos 21. századi képességek bármelyik diáknak, függetlenül attól is, hogy milyen munkakörben fognak a későbbiekben elhelyezkedni.

2.2.1. Társasjátékozás

A játéktervezést összekötöttük társasjátékozással is. Ennek a szórakozáson és a kikapcsolódáson felül az volt a célja, hogy tervezői szemmel is vizsgálják meg az egyes játékokat. Minden csapat rotációban próbálta ki a stílusukban nagyon különböző játékokat, majd ki kellett tölteniük egy elemző lapot a társasjátékokra vonatkozóan.

Ezzel lehetőséget biztosítottunk a diákoknak arra, hogy kamatoztassák a játéktervezés foglalkozásokon megszerzett ismereteiket. Képesek voltak elemezni a játékokat, eldöntötték közösen, hogy mi teszi szórakoztatóvá vagy épp frusztrálóvá. Ezen felül írhattak arra vonatkozóan ötleteket, hogy hogyan lehetne változtatni, esetleg javítani a játékon. Itt főleg a célok megváltoztatására helyeztük a hangsúlyt, hiszen arról tanultak a legtöbbet.

2.3. Unplugged informatika

A táborban minden nap 6 órányi foglalkozáson vettek részt a tanulók. Az egészségükre is gondolnunk kellett: nem ülhettek minden nap hat órán keresztül egy kényelmetlen székben és nem is tölthettek ebből az időből négy órát a képernyő előtt.

Az unplugged (számítógép használata nélküli) informatikai foglalkozások bevezetésével erre nem is volt szükség. A *CS Unplugged* programjai közül válogattunk témákat. Ezek olyan feladatok, tananyagok, amelyek kézzel fogható játékokon keresztül mutatnak be különböző számítógéptudományi témákat [22].

A tanulók változó informatikai háttérrel rendelkeztek, ezért olyan feladatokat választottunk, amelyek bevezető témának alkalmasak, de mindenki számára biztosítanak új ismereteket:

- bináris számrendszer, előjel nélküli egész számok ábrázolása,
- bevezetés a digitális képábrázolásba, faxgép működése,
- tömörítési eljárások, veszteséges és veszteségmentes tömörítés.

A feladatokat „magyarosítottuk”, illetve kisebb-nagyobb módosításokkal, extra gyakorló példákkal kiegészítve használtuk őket [22]. Mindegyik témát diszkusszióval zártuk, megbeszéltük, hogy a mindennapi életünkben hol és hogyan fedezhetők fel ezek az informatikai jelenségek.

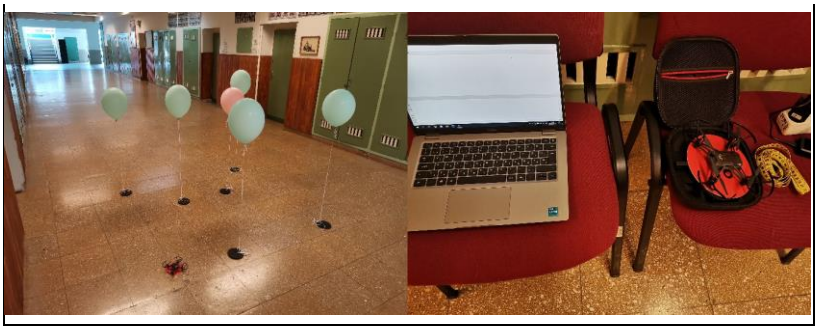
A diákok nagyon élvezték, hogy egy játékfejlesztés foglalkozás után kimehettek az udvarra és a nyári napsütésben „piknikezhettek” a feladatok megoldása közben (9. ábra).

9. ábra: ismerkedés a tömörítési algoritmusokkal a szabadban.

2.3.1. Drón programozás

Az unplugged foglalkozásokat egy meglepetés feladattal zártuk. A diákoknak csak annyit árultunk el a feladattal kapcsolatban, hogy pármunkáról lesz szó. Az iskolánkban rendelkezésre állt egy Python nyelven programozható drón.

A feladat két részből állt. Először a tanulók kiválasztottak egy pályakártyát a pakliból. A pálya tanulmányozása után a kezükbe vettek egy mérőszalagot és megépítették az akadálypályát úgy, hogy az alakzat minden sarkára elhelyeztek egy lesúlyozott héliumos lufit (10. ábra).



10. ábra: megépített akadálypálya lufikkal (bal), laptop, drón és mérőszalag (jobb).

A pálya megépítése után tértek át a programozásra. Az egyikük megfordította a pályakártyát, aminek a hátulján megtalálható a drón utasításkészlete. A másik diák gépelte a forráskódot. A párosnak együtt kellett megterveznie a program algoritmusát, a „kártyás” ember pedig segítette a gépelő társát a szintaxissal. Érdemes megjegyezni, hogy a drónhoz alapértelmezetten tartozó SDK parancsértelmezőként működik, azaz egyesével lehet a drónnak utasításokat küldeni hálózaton keresztül.

Ezért írtunk egy csomagoló API-t, amely segítségével előre megírható a drón teljes útvonalának programkódja. A program szekvenciálisan, másodpercenként küldi át az utasításokat. Így egy LOGO-szerű program írásával lehetett programozni a drón mozgását. A kék lufik közé becsempésztünk egy rózsaszínt is. A diákoknak kiadtuk az utasítást, hogy amikor a jármű a rózsaszín lufi felé ér, akkor szaltóznia kell egyet.

A tanulók személyes beszámolója alapján ez a meglepetés feladat volt az egyik kedvencük a táborban. A pályák és a csomagoló API szabadon felhasználható és ingyenesen elérhető a feltüntetett hivatkozáson keresztül [7].

2.4. 3D modellezés és nyomtatás

Végül a tábor utolsó témaköre következett, a 3D modellezés és a 3D nyomtatás. A tervezéshez a böngészőből elérhető, kezdőbarát és teljesen ingyenes Tinkercad programot használtuk [24]. A tanulóknak lehetősége volt a matematika órákon halmazok és geometria témakörökben megszerzett tudásukat kamatoztatni.

Az első feladat egy olyan kulcstartó tervezése volt, amely formája megegyezik az iskola címerével. A diákok nagy lelkesedéssel, motivációval vetették bele magukat ebbe a témakörbe is (11. ábra).

11. ábra: 3D modellezés szárnypróbálgatások.

Az első modell elkészítése jóval gyorsabban ment a tervezetthez képest. Így maradt idő a foglalkozáson arra, hogy mindenki megtervezze a saját kulcstartóját, amit ki is nyomtathat a tábor ideje alatt. A tanulók kreatívan oldották meg a feladatot, nagyon szép modellek, kulcstartók születtek (12. ábra).

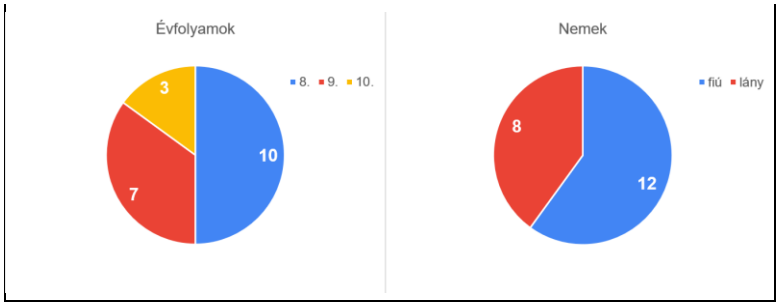
12. ábra: az egyik diák által készített baglyos kulcstartó.

3. Tapasztalatok, következtetések

Összességében elmondható, hogy a tábor elérte a célját. A diákok személyes beszámolója alapján jól érezték magukat, sok újat tanultak és gyorsan eltelt nekik a hét. Tanári oldalról azért érezhető volt, hogy sok foglalkozást szerveztünk nekik, az utolsó napokban már fáradtak voltak, valamint egyre nehezebben tudtak koncentrálni a délutáni órákban. Fontos megjegyezni azt is, hogy pályafutásunk

során először szerveztünk ilyen tábort, így a végeredményhez hozzátartozik, hogy nekünk is bőven van miben fejlődni, tapasztalatot gyűjteni.

A táborban összesen 20 tanuló vett részt, akiknek évfolyam- és nem szerinti eloszlását mutatja az alábbi ábra (13. ábra).



13. ábra: a tanulók aránya évfolyam (bal) és nem szerint (jobb).

A résztvevőkkel kitöltettünk egy névtelen kérdőívet a tábor első napjának első foglalkozásán, illetve lezárásként a foglalkozások befejeztével a hét legvégén is.

Mindenképpen érdemes összehasonlítani két kérdést, ahol a diákok saját szavaikkal rövid szöveges választ adtak:

- Azért jelentkeztem a táborba, mert... (tábor elején)
- Leginkább az tetszett a táborban, hogy... (tábor végén)

Az első kérdésre 10 válaszadó indokolt azzal, hogy „szeretem / érdekel az informatika", 3 konkrétan a programozást emelte ki, 3 a 3D nyomtatást nevezte meg, 3 másik pedig egyéb indokokat sorolt fel.

Ezzel szemben a tábor utáni kérdésben 10 válaszadónak a játékfejlesztés volt a kedvence, 8-an pedig a jó hangulatot, a társaságot és a kreatív, változatos programokat emelték ki.

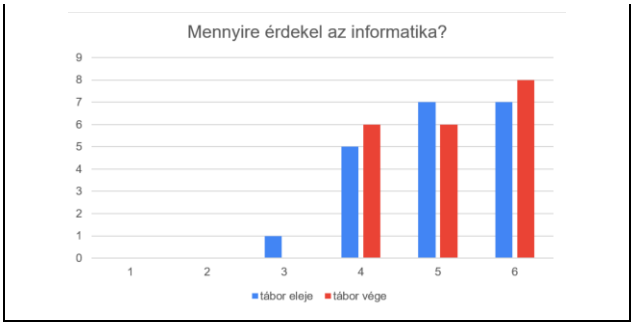
Ebből azt a következtetést vonhatjuk le, hogy a programterv előzetes ismertetése ellenére a diákoknak nem volt elképzelésük arról, hogy milyen programokra, konkrét feladatokra számíthatnak a foglalkozások nevei alapján. Örömmel vettük, hogy a tábor végére képesek voltak egyértelműen kiemelni a számukra legfontosabb előnyöket. A tény, hogy a válaszok fele a játékfejlesztést emelte ki, mindenképpen bizakodásra ad okot a játékfejlesztés témakör oktatási felhasználásra.

A tábor elsődleges célja a diákok motiválása volt az informatika tanulásához. Tekintve, hogy pályakezdő pedagógusként először, előzetes tapasztalatok nélkül szerveztünk ilyen programot, nem célunk bizonyítani, hogy ez a módszer hatékony a motiváció növelésére. A kérdőíveket a saját tapasztalataink rendszerezésére, az eredmények könnyebb értelmezése érdekében készítettük. Ez nagyban elősegíteni a jövőbeli munkánkat.

A tábor elején és végén is megkérdeztük tőlük a következőket:

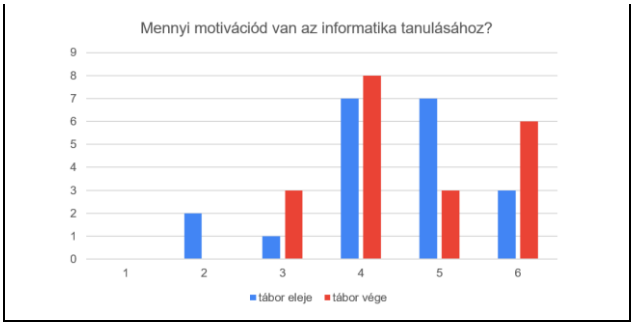
- Mennyire érdekel az informatika? (1–6 Likert-skála)
- Mennyi motivációd van az informatika tanulásához? (1–6 Likert-skála)

Az első kérdésben minimális eltérés mutatkozott a tábor előtt és után, a diákok többségét kifejezetten érdekli az informatika (14. ábra).



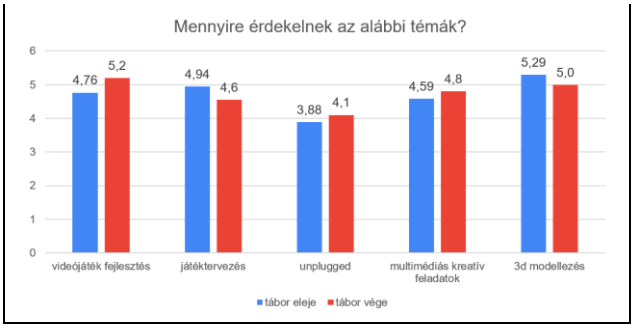
14. ábra: első kérdésre adott válaszok
a tábor elején (kék) és végén (piros).

A motivációra vonatkozó kérdésben viszont már jelentősebb eltérés mutatkozik a két kitöltés között (15. ábra). Pozitív irányú elmozdulások jelennek meg, a korábban ötös értékelést adó diákok közül a tábor befejeztével sokan a hatos választ („rendkívül motivált vagyok") jelölték meg. A másik végletet nézve pedig eltűntek a kettes értékelések, amelyek már erősen a másik véglet felé húztak („szinte nincs motivációm, nagyon unalmasak az órák").



15. ábra: második kérdésre adott válaszok
a tábor elején (kék) és végén (piros).

Érdekes összehasonlítási alap, hogy a diákok saját bevallásuk szerint mennyire érdeklődnek egy-egy témakör iránt. A tábor elején és végén is megkértük őket, hogy értékeljék 1–6-ig a témákat. Ezeknek az átlagos pontszámait hasonlítottuk össze (16. ábra).



16. ábra: átlagos pontszámai az egyes témaköröknek
a tábor elején (kék) és végén (piros).

Ez a diagram is alátámasztja a tanulók válaszait a tábor végén, miszerint a játékfejlesztés volt a kedvenc témakörük. Az átlagpontszámokban a legnagyobb pozitív növekedés ennél a témánál figyelhető meg és a tábor végén összességében is a játékfejlesztése a legmagasabb értékelés. Ebből arra következtettünk, hogy az előzetes elvárásaikat is felülmúlta ez a témakör. Érdekes módon a 3D modellezés iránti érdeklődésük csökkent, pedig a tábor elején a pontszám is rendkívül magas volt, illetve a szöveges kérdésben is többen megemlítették ezt a témát.

A legnagyobb csökkenés a játéktervezés témakörben figyelhető meg, 3 tanuló is megemlítette szöveges válaszaiban, hogy unalmasnak találták. A saját észrevételünk alapján a papírfocira vonatkozó feladatot már élvezték a csoportok, így valószínűleg a bevezető foglalkozásokat lenne érdemes átdolgozni, élvezetesebbé tenni számukra. Ha több idő jutott volna játéktervezésre, valószínűleg gyökeresen más véleményük lenne róla, hiszen a kurzus végén minden csoportnak egy saját társasjátékot kell terveznie és tesztelnie. Összességében ez a visszajelzés nagyon hasznos volt, a jövőben mindenképpen újra fogjuk gondolni ezt a témakört.

A legalacsonyabb értékelést mind a két méréskor az unplugged foglalkozások kapták. Ez érdekes módon nem tükrözi a valóságot. 1 válaszadó ugyan kiemelte, hogy szerinte az unplugged feladatok unalmasak voltak, de a tábor ideje alatt a tanulók nagyon élvezték, hogy tantermen kívüli programokon is részt vehettek. Két különböző kérdésre összesen 7 olyan válasz is érkezett, amely kevesellte kinti programokat. Ezek alapján elmondhatjuk, hogy igénylik a diákok azt, hogy egy informatika táborban sem egész nap a számítógép előtt üljenek. A személyes meglátásunk is az, hogy élvezték a kinti unplugged foglalkozásokat és a kötetlenebb beszélgetéseket.

Mindent egybevetve elmondhatjuk, hogy a diákok jó élményekkel, új tudással gazdagodva zárták az egyhetes élményinformatikai táborunkat. A szervezéssel és tervezéssel járó rengeteg kihívás ellenére pedagógusként is értékes tapasztalatokra, élményekre tehettünk szert. Bízunk benne, hogy a résztvevő diákok informatika tantárgy iránti motivációjára valóban pozitív hatást gyakorolt a tábori részvétel.

Irodalom

1. Audacity: *Free, open source, cross-platform audio software for multi-track recording and editing*. https://www.audacityteam.org/. (utoljára megtekintve: 2022. 10.)
2. C. Ogden, D. J. Malan: *CS50's Introduction to Game Development, Flappy Bird*. (2018). https://cs50.harvard.edu/games/2018/weeks/1/. (utoljára megtekintve: 2022. 10.)
3. Diósgyőri Gimnázium: *DIN 2022 játékok*. (2022). https://diosgyori.itch.io/. (utoljára megtekintve: 2022. 10.)
4. D. Kahneman: *Thinking, Fast and Slow*. New York: Farrar, Straus; Giroux. (2011).
5. G. Csapó: *Placing Event-Action-based Visual Programming in the Process of Computer Science Education*. APH. 16 (2), 35-57. (2019).
6. Godot Engine: *Documentation*. (2022). https://docs.godotengine.org/en/stable/. (utoljára megtekintve: 2022. 10.)
7. Gulácsi Ádám: *Github – drón csomagoló API forráskódja*. https://github.com/guladam/dji_tello_edu_py. (utoljára megtekintve: 2022. 11.)
8. Gulácsi Ádám: *Github – játékfejlesztés bevezető projekt forráskódja*. (2022). https://github.com/guladam/din_2022_labda_godot. (utoljára megtekintve: 2022. 10.)
9. Gulácsi Ádám: *Github – játékfejlesztés zuhanó repülés projekt forráskódja*. (2022). https://github.com/guladam/din_2022_zuhano_repules_godot. (utoljára megtekintve: 2022. 10.)
10. J. Schnell: *The Art of Game Design: A Book of Lenses*. Taylor & Francis Group. (2019).
11. K. M. L. Cooper, W. Scacchi: *Computer Games and Software Engineering*. Taylor & Francis Group. (2020).

12. Labchirp: *A Free and versatile sound effects generator*. http://labbed.net/software/labchirp/. (utoljára megtekintve: 2022. 10.)
13. Magyar Közlöny: *2020 évi 17. szám*. (2020). https://magyarkozlony.hu/dokumentumok/3288b6548a740b9c8daf918a399a0bed1985db0f/letoltes. (utoljára megtekintve: 2022. 10.)
14. M. K. Alderman. *Motivation for achievement: Possibilities for teaching and learning*, Routledge. (2013).
15. M. Csernoch: *The Stepchild of Informatics Education: File Management*. Academia Letters 2021 1-4 (2021).
16. M. Csernoch: *Thinking Fast and Slow in Computer Problem Solving*. Journal of Software Engineering and Applications, (2017) 10(1). http://file.scirp.org/pdf/JSEA_2017012315324696.pdf. (utoljára megtekintve: 2022. 10.)
17. M. Csernoch, P. Biró, J. Máth, K. Abari: *Testing algorithmic skills in traditional and non-traditional programming environments*. Informatics in Education. 14 (2), 175-197, 2015.
18. P. Domokos, M. Széll, V. Takács: *Blocklino: a graphical language for Arduino*. In: 8th IEEE International Conference on Cognitive Infocommunications: CogInfoCom 2017 : Proceedings : September 11-14, 2017 Debrecen, Hungary, IEEE Computer Society, Piscataway, 45-50 (2017).
19. Riot Games: *Urf Academy*. (2022). https://www.riotgames.com/en/urf-academy/curriculum-guide. (utoljára megtekintve: 2022. 10.)
20. R. Hunicke, M. LeBlanc & R. Zubek. *MDA: A formal approach to game design and game research*. Proceedings of the AAAI Workshop on Challenges in Game AI (Vol. 4, No. 1, p. 1722). (2004).
21. R. Syngel: *The rise and fall of Flappy Bird*. (2014). https://www.wired.co.uk/article/flappy-bird. (utoljára megtekintve: 2022. 10.)
22. T. Bell, I. Witten: *CS Unplugged: An enrichment and extension programme for primary-aged students*. (2015). https://www.csunplugged.org/en/. (utoljára megtekintve: 2022. 10.)
23. T. Nagy, M. Csernoch, P. Bíró: *The Comparison of Students' Self-Assessment, Gender, and Programming-Oriented Spreadsheet Skills*. Education Sciences. 11 (10), 1-29, 2021.
24. Tinkercad: *3D digitális tervek létrehozása online CAD-del*. https://www.tinkercad.com/. (utoljára megtekintve: 2022. 10.)

Alternatívák a LEGO EV3 helyettesítésére, az újgenerációs Mindstorms és a Stem:Bit készletek használata az oktatásban

Gaál Bence[1], Solymos Dóra[2]

{[1]gaalbence, [2]solymos.dora}@inf.elte.hu
ELTE IK

Absztrakt. Aki foglalkozott már robotika oktatásával, biztosan találkozott a LEGO Mindstorms EV3 modelljével, amely 2013 óta meghatározó szerepet tölt be ezen a területen. Ezen robotok gyártása azonban hivatalosan is megszűnt és támogatásuk is megszűnik a közeljövőben. Cikkünkben az új generációs LEGO Mindstorms Robot Inventor bemutatása mellett egy olcsóbb alternatívát is be kívánunk mutatni a Stem:Bit képében, amelyeket az EV3 helyettesítésére ajánlunk. Az eszközök összehasonlításának fontos eleme, hogy megvizsgáljuk azt is, milyen funkciókat biztosítanak az adott eszközök az EV3-hoz viszonyítva és melyiknek milyen előnyei és hátrányai vannak a robotika oktatásában.

Kulcsszavak: robotika, programozás, Mindstorms EV3, Robot Inventor, Stem:Bit

1. Bevezetés

2020 szeptemberében bevezetésre került NAT 2020-hoz illeszkedő kerettantervekben már az általános iskola harmadik osztályában [1] megjelennek a robotika alapfogalmai. Alsó tagozatban általában a népszerű padlórobotokkal kezdik a gyerekek a robotika megismerését, azonban felsőbe átlépve már komolyabb, nagyobb tudással és több programozási lehetőséggel rendelkező robotokat ismerhetnek meg [2]. Ebbe a csoportba tartozik a LEGO Education által fejlesztett LEGO Mindstorms EV3 is, amely éppen a NAT 2020 bevezetésével egy időben került kivezetésre a forgalomból. Az eszköz oktatásban betöltött szerepéről és felhasználhatóságáról számos cikkben olvashatunk [3][4][5] azonban, ha valaki, most szeretné beszerezni, akkor sajnos alternatívát kell keresnie, mert ezt már nem forgalmazzák az üzletekben.

Ehhez szeretnénk segítséget nyújtani két eszköz bemutatásával. Az első eszköz az EV3 utódja, amelynek a hivatalos neve LEGO Mindstorms Robot Inventor, magyarul pedig a Robot Feltaláló elnevezést kapta, azonban a köznyelvben EV4-ként is hivatkoznak rá. A másik eszköz a Stem:Bit, ami egy LEGO kompatibilis elemeket tartalmazó, BBC Micro:bit eszközzel vezérelhető robot.

Cikkünkben bemutatjuk a változásokat és a feladatok felhasználásához szükséges módosításokat. Tapasztalataink alapján mindkét eszköz használható az oktatás során, a vizsgált feladatok kisebb módosításokkal megoldhatók, így a korábbi verzióhoz készült tananyagok használhatóvá válnak.

2. A LEGO Mindstorms Robot Inventor (51515) bemutatása

A legújabb LEGO Mindstorms eszköz a Robot Inventor (51515) 2020-ban jelent meg, leváltva elődjét, az EV3-t. Az új eszköz papírdobozban érkezett, mely az 1. ábrán látható. A doboz 47,5 cm széles, 37 cm magas és 6,7 cm vastag, melyet felnyitva rengeteg LEGO elemet találunk, valamint egy matricákat tartalmazó kis ívet és egy kis füzetet, amiben megtalálható az elemlista. A doboz kinyitása egyszerűsödött az EV3 otthoni (31313) változatához képest, ugyanis cipősdobozhoz hasonlóan

egymásba illeszthetőek a részei, azonban, ennél jobb megoldás volt az EV3 oktatási (45544) verziójának doboza. Ugyanis, az oktatási verzióban volt belső rendszerező, ebben azonban nem találunk ilyet.

Ajánlott korcsoport	10 éves kortól
Programozási nyelv típusa:	blokkos/Python
Építőelemek száma	949
Motorok száma	4
Érzékelők típusa	távolság-, fény- és színérzékelő, iránytű, giroszkóp
Egyéb tartozékok	-
Jelenlegi ár	129 990 Ft[1]

1. táblázat: A LEGO Mindstorms Robot Inventor 51515 általános ismertetője

7. ábra: A LEGO Mindstorms Robot Inventor (51515) készlet doboza[2].

A dobozban összesen 949 elem található, ami összehasonlításképpen, az otthoni verzió elemszámának majdnem 1,5-szerese, míg az oktatási verziónak a kétszerese. Az elemek között megtaláljuk az EV3 készletekben található elemek jelentős részét, újdonság viszont a több nagyméretű elem a dobozban, köztük egy tábla szerű elem[3], amely alkalmas arra, hogy rá, illetve bele illesszünk érzékelőket, motorokat és egyéb elemeket. (A legjobban a BBC Micro:bittel használható próbapanelekhez lehetne hasonlítani az új LEGO táblát.) Az EV3-ban található fém golyót kivették a készletből, ami autószerű felépítéseknél okoz(hat) problémákat. Az elemek megjelenése modernizálódott, színesebb lett, emiatt inkább a LEGO Education SPIKE™ Prime (45678) szetthez hasonlítható.

2.1. Modellek

A fejlesztők 5 modellt terveztek a készlethez, melyek építési útmutatóit a programozáshoz használt alkalmazásban találhatjuk meg. (Erről a későbbiekben írunk részletesebben.) A modellek az 1. ábra

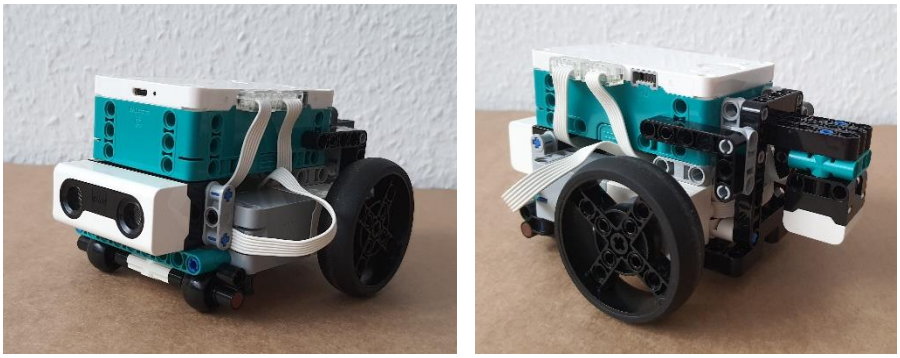
[1] Forrás: https://www.lego.com/hu-hu/product/robot-inventor-51515

[2] Forrás: https://www.lego.com/hu-hu/product/robot-inventor-51515

[3] Megtekinthető a következő honlapon: https://www.brickowl.com/catalog/lego-dark-turquoise-panel-11-x-19-39369

bal oldalán is láthatóak. Neveik balról jobbra a következők: M.V.P, GELO, BLAST, CHARLIE és TRICKY. A modellek nagy részéhez tartozik módosítási lehetőség építési útmutatóval. Például, a TRICKY alapmodellt át lehet úgy építeni, hogy tudjon kosárlabdázni (egy emelő karral mozgatja a labdát), focizni (lábszerű karokkal elrúg egy labdát), bowlingozni (forgatva kilő egy labdát) vagy rajzolni (felemelhető toll állványt lehet rátenni).

Az eszköz teszteléséhez többféle modellt is kipróbáltunk, azonban oktatási célra a készítők által ajánlott TRICKY felépítés tűnt a legpraktikusabbnak (2. ábra). A modellt kis mértékben módosítottuk, a díszitő elemeket nem helyeztük fel az esetleges későbbi módosítások miatt, ami hosszabb távon jó döntésnek bizonyult. Az alapmodellbe az útmutató szerint csak a távolságérzékelő kerül beépítésre, azonban mi szerettünk volna minél több érzékelőt kipróbálni, így a robot hátsó részére illesztettük a színérzékelőt, majd a nyomásérzékelőt (3. ábra).



2. ábra: A TRICKY nevű modell látható a képen, a ráadásként felszerelt színérzékelővel[4].

2.2. Motorok és érzékelők

A motorok számát tekintve növekedés történt, ugyanis négy egyforma méretű, úgymond nagy motort kaptunk a korábbi két nagy és egy közepeshez képest. Ezen kívül sokkal kisebbek lettek és több illesztő felülettel rendelkeznek, mint a korábbi Mindstorms motorok. Ezzel a megjelenéssel és működéssel a SPIKE szettben találkozhattunk korábban más színben. A motorok rendszere is frissült, ugyanis időről-időre az alkalmazás szól, hogy megjelent egy új verzió és frissíteni kellene őket, amivel korábban nem találkozhattunk.

Az érzékelők jelentősen megváltoztak a megjelenésükben, ezek is a SPIKE szett elemeire hasonlítanak, mint a motorok. A csomagban egy ultrahangos távolságérzékelőt találunk, mint az EV3 oktatási verziójában, azonban ez kisebb hatótávolsággal rendelkezik, csupán 200 cm-re lát el. Az eszköz pontosságát tekintve tapasztalataim alapján javulás fedezhető fel. További extraként elhelyeztek az érzékelő körül négy darab lámpát, amelyeket külön-külön programozhatunk.

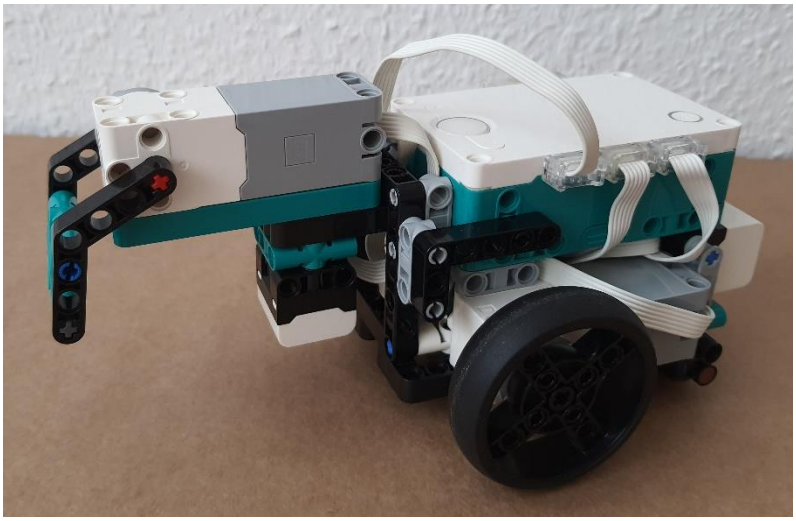
A fény- és színérzékelő nyolc színt és a semleges színt tudja megkülönböztetni. A korábbi listából kivették a barnát, helyette bekerült a rózsaszín, és kiegészítették a szett cián kék színével. Tehát a következő színeket érzékeli az eszköz: fekete, rózsaszín, kék, cián, zöld, citromsárga, piros, fehér és semleges. Az EV3-ban megismert fényvisszaverődés itt is használható, sőt, mindkét változata (*reflected light* és *ambient light*) elérhető a programozó felületen. Az érzékelő ezeken kívül képes megmérni, hogy az általa látott szín milyen mértékben tartalmaz pirosat, kéket vagy zöldet. Tehát, le tudnánk kérdezni egy szín RGB kódját, amit a színérzékelős feladatokhoz fel tudnánk használni. Így tudnánk olyan pályát vagy nyomkövető színes csíkokat nyomtatni, amit nagyobb valószínűséggel érzékel az

[4] Forrás: Solymos Dóra saját fotói

eszköz, mint korábbi változata. Azonban a színérzékelő továbbra is gyenge pontja az eszköznek, ugyanis a saját LEGO elemeiből sem ismeri fel minden esetben a zöldet, a kéket és az új ciánt, melyek az EV3-nál is problémás színek voltak. Bizonyos szögben, bizonyos magasságban nagyobb az esély a sikerre ezeknél a színeknél is, azonban, ha biztosra szeretnénk menni, inkább maradjunk a fekete, fehér, piros vagy sárga színeknél.

Az iránytű és a giroszkóp beépítésre került a Hub nevet viselő vezérlőbe. Ezeknek köszönhetően lehet például erőteljes érintésre megjeleníteni egy-egy smiley-t a Hubon, vagy a BBC Micro:bithez hasonlóan a különböző irányú mozgatásokra beállítani különböző funkciókat.

Az EV3-hoz képest hiányérzetünk lehet a szenzorokat tekintve, ugyanis a szettben nincs nyomásérzékelő. Azonban a SPIKE-ban van, ami kompatibilis ezzel a szettel, így ha be tudjuk szerezni, akkor könnyedén tudjuk használni. Ha szeretnénk valamivel helyettesíteni, akkor a meglévő elemeinkből kreativitásunkat felhasználva építhetünk valamit, ami elfogadható egy nyomásérzékelőnek. Az interneten[5] a legtöbben motor felhasználásával készítenek nyomásérzékelőt, mi is ebből inspirálódtunk és a 3. ábrán látható érzékelőt készítettük el, ami a színérzékelő felett kapott helyet a tesztautón. Ez az összeállítás működőképes, helyettesítheti a nyomásérzékelőt, de pontosságát tekintve gyenge.



3. ábra: A tesztautó nyomásérzékelővel kiegészített változata[6].

2.3. A vezérlő, vagyis a Hub

Megjelenésében jelentős változást tapasztalhatunk az elődhöz képest. Az EV3-on megtalálható kijelző eltűnt, így számos eddig megszokott műveletet nem vagy máshol tudunk elérni. A Hubon egy nagy bekapcsoló, két kisebb programozható, és egy bluetooth gombot találunk. A bekapcsoló gomb körül lévő LED lámpa programozható, mint elődjénél, csak a színek megváltoztak és nőtt a számuk. A kijelző, mint rajzoló felület bizonyos szempontból egyszerűsödött és inkább a Micro:bit ledjeihez hasonlóan vált programozhatóvá, tehát koordinátánként tudjuk őket kapcsolni és állítani a

[5] https://www.eurobricks.com/forum/index.php?/forums/topic/183443-fix-no-force-sensortouch-sensor-in-51515-robot-inventor/

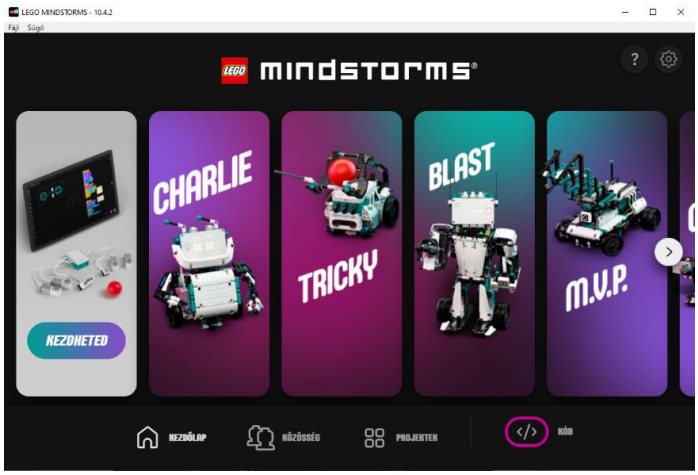
[6] Forrás: Solymos Dóra saját fotója

fényerősségüket. Továbbá, meg tudunk jeleníteni rajta animációkat és a képernyőnél hosszabb szövegeket, amiket fényszalaghoz hasonló módon jelenít meg. Ezen kívül még hang lejátszására is alkalmas.

A vezérlőn hat darab port található, amelyek az eszköz hosszanti oldalán helyezkednek el egyenlő arányban és A–F betűkkel azonosíthatók. A portok között nincs különbség, bármelyikbe csatlakoztatható motor és érzékelő is. Ezen kívül még találunk egy micro USB csatlakozási lehetőséget is a Hubon. Ezen keresztül, a csomagban található micro USB – USB kábel segítségével csatlakozhatunk a számítógéphez. A kábel nemcsak töltésre alkalmas, hanem adatátvitelre is.

2.4. A hivatalos programozói felület bemutatása

A Robot Inventor programozásához egy alkalmazást kell letöltenünk, ami elérhető a Microsoft Store-ban, a Play Áruházban és az App Store-ban is. Az alkalmazás neve magyarul LEGO® MINDSTORMS® Feltaláló, angolul pedig LEGO® MINDSTORMS® Robot Inventor. Több nyelv közül lehet választani, de először a telepített eszköz alapértelmezett nyelvén fog elindulni az alkalmazás. Az app megnyitásakor a 4. ábrán látható felület fog megjelenni, ami az app kezdőlapja. A *Kezdheted* gombot választva az eszköz használatához kapunk segítséget az alapoktól indulva. A modellek képeire kattintva pedig elérhetjük az építési útmutatókat, majd az építés után, kapunk egy kódot is, amivel kipróbálhatjuk a felépített robotunkat. A *Közösség* lapra kattintva elérjük a felhasználók által készített modellek építési útmutatóit. A felület hátránya, hogy csak akkor tudjuk megnézni bármelyik útmutatót, ha letöltjük az összeset. A főmenü harmadik gombja a *Projektek*, amivel a korábbi kódjainkat tudjuk előkeresni. Itt egy helyen találjuk meg a blokkos és a Pythonban megírt kódjainkat is. A kezdőlap utolsó gombja a *Kód*, amivel a tényleges programozói felületre tudunk átlépni.



4. ábra: A programozói felület kezdőlapja[7].

Az EV3-t a hivatalos szoftverjében ikon alapú blokkokkal tudtuk programozni, ahol vízszintesen kellett elhelyezni egymás mellett a blokkokat. Később MicroPythonban is lehetett programozni, de szöveg alapú blokkokkal is programozhattuk a Makecode felületén, ahol függőlegesen kellett egymásba illeszteni a blokkokat és egy szimulátor is rendelkezésünkre állt. Az új Mindstorms blokkos

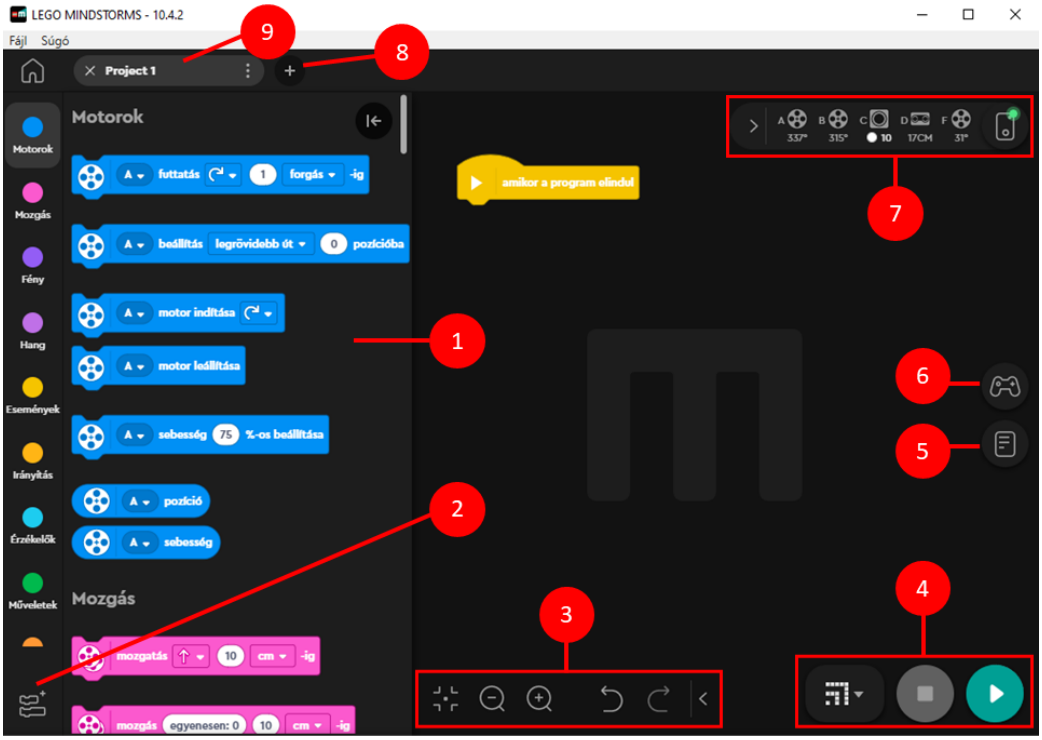
[7] Forrás: Solymos Dóra saját fotója

felületén szöveg alapú blokkokat használhatunk, amiket függőlegesen kell egymásba illeszteni. A környezet maga Scratch alapú, így a kinézet és néhány blokk ismerős lehet. Ezen kívül programozhatjuk az eszközt Python nyelven is a MicroPython segítségével. Ehhez készítettek egy dokumentációt is, amely angol nyelven itt érhető el: https://lego.github.io/MINDSTORMS-Robot-Inventor-hub-API/index.html

A programozói felület (5. ábra) letisztult, csak a programozáshoz szükséges gombok láthatóak, azonban ezek közül is elrejthetünk néhányat, ha éppen nem kellenek. A felület részei a következők:

1. Blokkpaletta
2. Bővítmények hozzáadása
3. Nagyítás, visszavonás és ismétlés
4. Letöltés/Streaming, leállítás és lejátszás
5. Súgó és figyelő
6. Távirányító
7. Hub kapcsolat
8. Új projekt létrehozása
9. Aktuális projekt

A felületen elrejthető az 1. számmal jelölt blokkpaletta, a 3. számú nagyítás, visszavonás és ismétlés, valamint a 7. számú Hub kapcsolat.



5. ábra: A blokkprogramozáshoz használható felület kezdőlapja[8]

[8] Forrás: Solymos Dóra saját fotója

A Bővítmények hozzáadása gombra kattintva felugrik egy ablak, ahol számos bővítmény közül választhatunk, melyek a következők: modellblokkok, több motor, több mozgás, zene, időjárás, LEGO powered up, további érzékelők. Ezen kívül kísérleti státuszban vannak, de ettől függetlenül kipróbálhatók a következő bővítmények: gépi tanulás, XBOX ONE kontroller, DUALSHOCK 4 kontroller és Hub-Hub kommunikáció.

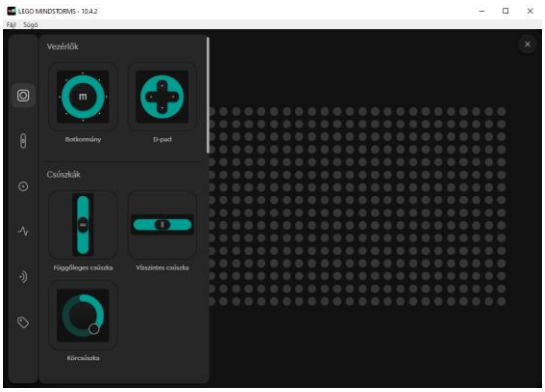
Az 5. ábrán 4-es számmal jelölt gombokkal van lehetőségünk a program letöltésére és bluetooth kapcsolaton keresztüli futtatására. Az utóbbi mód a *Streaming* mód kiválasztásával érhető el, amivel letöltés nélkül, azonnal futtatjuk a programunkat. A letöltési mód annyiban változott a korábbi verzióhoz képest, hogy itt maximalizálták a letölthető programok számát, mégpedig 19-ben, és a letöltés előtt kiválaszthatjuk, hogy hányadik helyre szeretnénk letölteni a programot. A letöltés után rögtön futtatja a Hub a programot, azonban, ha az eszközről szeretnénk elindítani azt, akkor ki kell keresnünk a sorszámát a Hubon a balra-jobbra gombokkal és a bekapcsoló gombbal kell elindítanunk.

Az EV3-nál magunk adhattunk nevet a programnak, és az alapján kellett megkeresnünk az eddigi összes letöltött fájl között. Az EV3-nál a fájlok karbantartása nehézkes volt, hiszen egyesével kellett törölni a fájlokat, azonban itt nagymértékben leegyszerűsödött. Az alkalmazásból könnyedén, egy gomb megnyomására törölhetjük a szükségtelen fájlokat.

A Robot Inventoron lévő két módot úgy tudjuk megkülönböztetni az eszközön, hogy más-más mintában kapcsolódnak fel a lámpák, ami a letöltésnél a letöltési hely sorszáma, míg a streamnél egy wifi jel.

A felületen (5. ábra 4-es számú gombcsoport) a mód választó gomb mellett találjuk a Leállítás és a Lejátszás gombokat. Ezek az első programok elkészítésénél nem tűnnek annyira fontosnak, azonban kicsit több próbálkozás után már elengedhetetlenek. Ugyanis az eszközön nincs olyan gomb, amivel le lehetne állítani az aktuálisan futó programot. (Ez az EV3-nál a képernyő bal alsó sarkában elhelyezkedő szürke gomb volt.) A program a blokkok végrehajtása után nem lép ki a programból, mint elődje, hanem tovább fut, még akkor is, ha már egyébként nincs mit tennie. Ekkor a Lejátszás gombra kattintva újra el tudjuk indítani a programot, míg a Leállítás gombbal tudjuk befejezni a futtatást. Másik módszer a programból való kilépésre az, hogy betesszük a kódunk végére az *állj és kilépés a programból* nevű narancssárga blokkot. Ekkor nincs szükség arra, hogy a Leállítás gombbal befejezzük a futtatást.

Az 5. ábrán a 6-os számú, kontroller alakú gombbal, egy kis táblát tudunk előhívni (6. ábra), amire különböző irányításhoz kapcsolódó gombokat, csúszkákat, érzékelőket helyezhetünk el, majd formázhatjuk őket és nevet adhatunk nekik. Ezt követően be is programozhatjuk őket és a futtatás során távirányítóként használhatjuk a felületet. Ezzel a funkcióval készíthetünk egy olyan programot, amely az EV3-hoz volt elérhető alkalmazás formájában és az eszköz telefonról történő irányítására lehetett használni.



6. ábra: A távirányító készítésére szolgáló felület[9]

A felületen található súgó egyfajta tudástárként is szolgál, hiszen a blokkok leírásán kívül videókat, útmutatókat és kódrészleteket is kapunk az egyes funkciók használatához. Azonban, a magyar felületen a súgó nem minden esetben érthető, ugyanis a fordítás néhol értelmetlen vagy nem azt adja vissza, mint például az angol nyelvű változata.

A 7-es számú, Hub kapcsolat nevű gombra kattintva tudunk csatlakozni az eszközhöz. Csatlakozás után itt tudjuk módosítani az eszköz nevét, kezelni tudjuk a Hubon lévő programokat, megváltoztathatjuk a motorok és szenzorok módját, illetve frissíthetjük a motorokat. Ezen kívül itt nézhetjük meg a beépített szenzorok aktuális értékeit is. A programozás során a Hub gombja mellett egy kis sávban láthatjuk, hogy melyik portba milyen eszköz van csatlakoztatva, sőt az aktuális értékét is kiírja a szenzoroknak. (A motoroknál megjeleníthető az aktuális elfordulási szög vagy sebesség.)

2.4.1. Blokkcsoportok bemutatása

A felület maga könnyen használható, különösen, ha már van blokkos környezetben tapasztalatunk, azonban az EV3-hoz képest bizonyos feladatokat máshogy kell megoldanunk. A motornál külön blokkban kapott helyet szinte minden olyan beállítás, ami eddig egy blokkban volt, például a motorok nevei, sebessége, irányítás módja (*degree, rotation, seconds*). A *Motorok* nevű (kék) blokkoknál minden egyes blokknál beállíthatjuk a használni kívánt motort, azonban a sebességet, irányt és a mozgás hosszát már külön blokkokban kell megadnunk. Az EV3-ban *rotation*nek nevezett módot magyarra forgásként fordították le, ami azért nem szerencsés, mert közvetlenül alatta találjuk a fok (korábbi *degree*) módot, amivel a gyerekek könnyen összekeverhetik.

Ha egy blokkban szeretnénk beállítani a motor mozgásának hosszát és a sebességét, akkor a bővítmények közül aktiválnunk kell a *Több motor* címűt, majd a blokkcsoportok legalján fogjuk megtalálni az erre alkalmas blokkokat. Ezek között találhatunk olyan blokkokat, amivel a motort elindíthatjuk általunk megadott értékű sebességgel vagy teljesítménnyel. A két blokk ugyanúgy néz ki, csupán az utolsó kifejezésben különböznek, a működésükben viszont nem találtunk különbséget. Ez a jelenség fennáll a *Több mozgás* című bővítménynél is.

A *Mozgás* című blokkok között a korábbi, két motort egyszerre működtető blokkokhoz hasonló blokkokat találunk. Azonban, itt csak az EV3-nál megismert *Move Steering* blokkhoz hasonló, iránymeghatározással működő blokkokat láthatjuk. Ha különböző sebességgel szeretnénk elindítani a két motorunkat, akkor ahhoz aktiválnunk kell a *Több mozgás* című bővítményt. Itt találhatunk elég összetett blokkokat is, hiszen egy-egy blokkban akár 2-3 értéket is módosíthatunk.

[9] Forrás: Solymos Dóra saját fotója

A *Fény* (lila) blokkok között találjuk a Hub „képernyőjét" használó blokkokat. Újdonság, hogy lehet animációkat lejátszani, amihez rendelkezésünkre áll egy könyvtár, ahol a modellekhez készült animációkat találjuk. Azonban készíthetünk mi magunk is animációt. Ezen kívül megjeleníthetünk egy-egy képet is a kijelzőn, ehhez viszont nem áll rendelkezésünkre könyvtár. Továbbá, megjeleníthetünk szöveget is, amiben előrelépés történt, hiszen nem csak a képernyő méretével megegyező hosszúságú szöveget lehet megjeleníteni, ugyanis a szöveget fényszalagként jeleníti meg az eszköz. Ez a funkció a micro:bitnél megismert szöveg megjelenítésre hasonlít, mint ahogy a koordinátánkénti pont megjelenítés is. Ezt itt a *képpontok beállítása 1-en, 1-on 100%-ra* blokkal tudjuk elérni. Ezen kívül, ebbe a csoportba került a bekapcsoló gomb körüli ledek programozására szolgáló blokk is, ill. az újdonságként megjelenő, a távolságérzékelő körüli lámpákat állító blokk is. Itt kapott helyet az eszköz tájolását beállító két blokk is.

A *Hang* blokkcsoportban az EV3-ból megismert blokkokkal találkozhatunk. Újdonságként jelent meg, hogy hangot nemcsak a Hubon tudunk lejátszani, hanem a programozó eszközön is. Emiatt a hangokat két csoportra osztották. A *Hangok a Hubon* című részben azok a hangok szerepelnek, amiket a Hubon tudunk lejátszani, míg a *Könyvtár* részben azok, amiket a programozó eszközön. Új funkcióként szerkeszthetjük a könyvtárban szereplő hangokat és egyszerűbbé vált a hangfelvétel készítés is. A zongora hangok lejátszására szolgáló blokkok nevei megváltoztak, angolul a *play beep* kezdetű blokkokkal, míg magyarul a *sípszó lejátszása* kezdetű blokkokal tudunk ilyen hangokat lejátszani. Az EV3-ban ezt a *Play Note* mód választásával tudtuk megtenni.

Az *Események* csoportban találhatunk olyan indító blokkokat, amelyekben egy-egy érzékelő hatására, a Hub mozgatására vagy gombnyomásra kezdhetjük a programunkat. Ezen kívül, megtaláljuk a Scratchből ismert futtatást indító blokkokat is.

Az *Érzékelők* című blokkok között találhatunk a színek érzékeléséhez szükséges blokkokat, ill. a fényvisszaverődéshez is, azonban ebben a részben csak a korábbi *reflected light* módban működő blokkokat érjük el, amit *tükröző fényként* fordítottak le. Ha az *ambient light* módot szeretnénk használni, akkor a *További érzékelők* csoport aktiválása után, a *Környezeti fény* című blokkokkal tudjuk azt megtenni. Az érzékelők között új funkcióként jelent meg a gesztusok érzékelése és a tájolás, ami szintén a Micro:bithez hasonló módon működik. A *További érzékelők* között találhatjuk a vörös, kék és zöld színek lekérdezésére szolgáló blokkot, ami a színértéket adja eredményül egy 0 és 255 közötti számként. Továbbá, itt találjuk a Spike szetthez tartozó nyomásérzékelő blokkjait is.

2.4.2. Milyen feladatokat oldhatunk meg a Robot Inventorral?

A Robot Inventorhoz angolul megjelent néhány könyv, azonban magyar tananyag még nem érhető el, így az EV3-hoz készült tananyagok közül a [6]-ben szereplő feladatok közül válogattunk néhányat, amit igyekeztünk a Robot Inventor blokkos környezetében megoldani. A tesztünk során a feladatok jelentős részét meg lehetett oldani az eszközzel, azonban bizonyos esetekben csak a feladatok átfogalmazásával tudtuk ezt elérni.

A színes ledek használatánál olyan problémába ütköztünk, hogy a ledet nem tudjuk kikapcsolni, ill. nem tartozik villogó funkció a blokkhoz, így az ehhez kapcsolódó feladatokat módosítanunk kellett. A kikapcsolást a fekete szín beállításával oldottuk meg, míg a villogást ciklus és várakozó blokkok kombinációjával.

Az érintésérzékelő hiánya miatt az ehhez kapcsolódó feladatokat egy motorral kellett megoldanunk. Ez a csere, a feladatok teljes átfogalmazását igényelte és néhány kihagyását. Ugyanis, így a *pressed, released* és *bumped* funkciók tesztelése értelmetlenné vált, hiszen nem egy gombunk van, hanem egy karunk, ami a kis változásokra nagyon érzékeny, viszont, ha el akarjuk fordítani, akkor azt csak kézzel tudjuk megtenni. Tesztünk alapján, ha túl gyorsan megy neki egy tárgynak a robot, akkor nem fordul el a nyomásérzékelő motorja, de az eszköz felborul. Ha lassan megy a robot, akkor érin-

tésérzékelőként használható, viszont ekkor minimális mozgás is elég az érzékeléshez. Ez látható a 7. ábrán látható kódban.

7. ábra: A következő feladatot megvalósító program kódja látható a képen: az érintésérzékelőként működő motor ha „benyomódik", akkor mosolygó fej jelenik meg a kijelzőn[10].

A tananyagban szereplő párhuzamos program készítésével kapcsolatos feladatokat ([6] 27. oldal 1–4. feladatok) meg lehet oldani ebben a környezetben is, azonban nem lesznek párhuzamos programok, hanem csak egyszálúak. Ezen kívül, nem lehet megoldani vagy csak a feladat jelentős mértékű egyszerűsítésével a házikó kirajzolásával kapcsolatos feladatot ([6] 45. oldal 5. feladat). A kijelző 5 × 5 pixeles mérete eléggé lekorlátozza a megjeleníthető képeket. Továbbá, a vonlakövetési feladatot ([6] 33. oldal 3. feladat) a szín- és fényérzékelő pontatlansága miatt egy matematikai képlet megadásával tudjuk csak megoldani. A képletről és annak működéséről ebből a videóból lehet többet megtudni: https://youtu.be/Z8Cv60f75LI.

A hangok között nem szerepelnek a színek nevei, azonban mivel elég egyszerűvé vált a hangfelvétel készítése, valamint akár telefonról is programozható az eszköz, így a gyerekek felvehetik magyarul a színek neveit a telefonjukkal közvetlenül az appban, amiket fel is használhatnak a feladat megoldásánál. Így pici hangszerkesztést is tanulhatnak, továbbá, saját nyelvükön tesztelhetik a *Hangos színek* című feladat ([6] 36. oldal 3. feladat) működését, ami németül tanuló gyerekek esetében előnyt jelenthet.

Az említésre nem kerülő feladatoknál különösebb problémát nem tapasztaltunk, a rendelkezésünkre álló blokkokkal meg tudtuk oldani őket.

[10] Forrás: Solymos Dóra saját fotója

3. A Stem:Bit bemutatása

3.1. Az eszköz specifikációja [7][8]

Ajánlott korcsoport	5. osztálytól felfele
Programozási nyelv típusa:	blokk alapú/Python/JavaScript/Scratch
Építőelemek száma	302
Motorok száma	2
Érzékelők típusa	akadályérzékelő, vonalkövető, ultrahang csatlakoztatási lehetőség
Egyéb tartozékok	rezgőmotor, 2 × 10 mm RGB LED, 3× kicsi RGB LED, infravörös vevő egység, távirányító
Ár (micro:bittel)	117,95€/csomag

2. táblázat: A Stem:Bit általános ismertetője

8. ábra: A Stem:Bit csomag[11]

A Stem:bit csomag két fő részre bontható, az egyik az építőelemek, a másik pedig maga a kiterjesztést szolgáló expansion board. Ehhez tudjuk csatlakoztatni a micro:bitet, amely vezérelni fogja a robotunkat. A táblázatban (2. táblázat) látható egyéb tartozékok és érzékelők mind integrálva vannak erre a boardra, amely az építés során esetlegesen korlátozhatja a kreativitást. A csomag építőelemei teljes mértékben kompatibilisek a LEGO termékekkel. Ugyan az EV3 csomagja csak öt hivatalos építési lehetőséget tartalmaz, de kapunk hitelesített felhasználói útmutatókat is. A tananyagok tekintetében is az EV3 javára billen a mérleg, ugyanis sajnos nem érhető el kidolgozott tananyag a Stem:Bithez.

[11] Forrás: https://shop.sb-components.co.uk/products/stem-bit-the-programmable-blocks-kit-for-micro-bit?variant=31064155455571

A micro:bit általi vezérlés lehetővé teszi azt, hogy egy széles életkori spektrumon használható eszköz legyen a Stem:Bit. Emiatt az eszköz rendelkezik minden olyan előnyös tulajdonsággal is, amely a micro:bitre jellemző, mint például a költséghatékonyság, a széleskörben történt alkalmazás és sikeresség, valamint a kiterjeszthetőség elég magas volumene[9]. A micro:bitről számos leírás és cikk elérhető már, ezért a cikkben külön nem foglalkoztunk azzal, hogy mint vezérlőegységet bemutassuk azt.

3.2. Modellek

9. ábra: Az IR Car modellje

A csomagból kilenc hivatalos modellt van lehetőség megépíteni, amelyhez a kézikönyv segítséget is nyújt az építési útmutatók által. Ezen modellek megépítése során nem ütköztünk nehézségekbe. Fontos megjegyezni, hogy vannak csak olyan modellek, amelyek mozgásra nem képesek. A leginkább EV3 feladatok megoldására képes modell az IR Car (9. ábra[12]). A nyomvonallal és objektumok elkerülésével kapcsolatos feladatok terén hasonlóan szerepel a két robot, azonban fontos figyelembe venni, hogy nem tudunk egy darab Stem:Bittel olyan robotot építeni, amely mozog és képes objektumokat is mozgatni, tehát ez egy nagy hátrány lehet. Az akadályok érzékelése és elkerülése terén mindkét robot hasonlóan teljesít akárcsak a vonalkövetésben. A WRO verseny szabályzata miatt versenyzésre ez a csomag nem alkalmas, hiszen csak és kizárólag LEGO robotokkal lehet ebben részt venni[10].

Lehetőségünk van két lánctalpas jármű elkészítésére is, amelyek jó alapok lehetnek a terepi mérőjárművek készítésére, amit tudunk távirányítóval is működtetni. A micro:bit nyújtotta komplex szenzorcsomag pedig lehetőséget ad arra, hogy a digitális kultúra órákon kívül, komolyabban számításba lehessen venni az eszközt a természettudományok oktatásában is.

A fentebb említett eszközökön túl, tudunk egy darut, egy csúzlit, egy tuk-tuk kocsit, egy terepen mászni képes hatlábú robotot, egy úgynevezett robot autót (vonalkövetésre nem alkalmas), és egy tárgyak megfogására alkalmas eszközt építeni a csomagból.

3.3. Motorok és érzékelők

A Stem:Bit csomagban összesen két darab motort kapunk. Nem csak a darabszámban marad el a csomag kvalitása az EV3-hoz képest, de a motorok fajtáját tekintve itt nem szervomotorokat ka-

[12] Forrás: https://shop.sb-components.co.uk/products/stem-bit-the-programmable-blocks-kit-for-micro-bit?variant=31064155455571

punk. Emiatt a teknőcgrafika padlón való megvalósításánál nehézségekbe ütközhetünk, mivel így csak a motorok működésének idejét és sebességét adhatjuk meg, utóbbit pedig nagyban befolyásolni fogja az akkumulátor töltöttsége. A kiterjeszthetősége szempontjából azonban három helyet kapunk szervomotorok számára, így maximálisan 5 motort tudunk csatlakoztatni az extension boardra.

Szenzorok terén egy rendkívül sokrétű eszközről beszélhetünk. Ennek egyik oka az, hogy a micro:bit által nyújtott szenzorkészletet is használhatjuk a munka folyamán, így kapunk: gyorsulásmérőt, iránytűt, fényérzékelőt, hőmérőt, mikrofont és érintő szenzort (utóbbi kettőt csak v2-es micro:bit esetén). Ezeken felül pedig a board is el van látva egy infravörös akadályérzékelő és egy vonalkövető szenzorral, valamint lehetőség van ultrahang szenzor csatlakoztatására is. Az akadály elkerülő és a vonalkövető a tesztek során viszonylag kevés alkalommal hibázott, azt lehet mondani, hogy a szenzorok működése megfelelő.

A színérzékelő hiánya miatt szintén nem fog tudni minden EV3 által végzett feladatot megvalósítani, például nem lesz képes az eszköz a színeket megkülönböztetni. Ezen felül van egy infravörös vevőegység is az eszközön, ami lehetővé teszi, hogy irányítsuk a robotot a csomaghoz kapott távirányító segítségével is. Kiterjeszthetőség szempontjából egy jóval komplexebb eszközt kapunk, ha a szenzorokat nézzük. A csomag építőszámát tekintve alulmarad a LEGO termékekhez képest, de a kompatibilitás miatt, itt is lehetőség van a bővítésre.

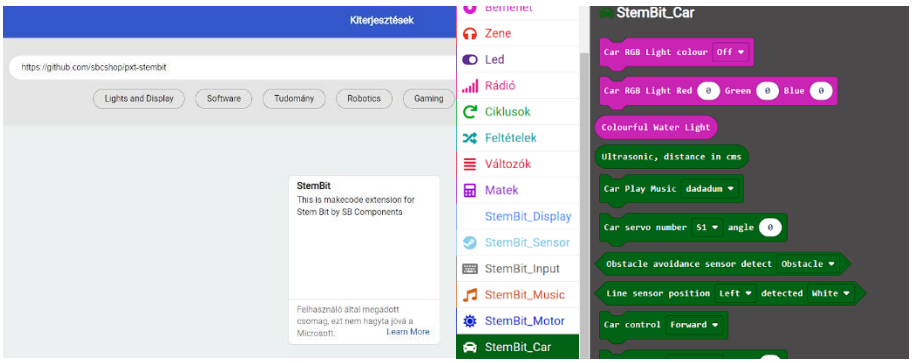
Amit még érdemes megemlíteni és hasznos lehet a gyerekek számára egy ilyen foglalkozáson, az a kimeneti perifériák. Itt kapunk egy rezgőmotort, valamint 3 kicsi RGB led izzót, amit programozhatunk, valamint további két 10 mm-es ledet, amely a board elején helyezkedik el, mintha egy autónak a fényszórója lenne. Amennyiben a micro:bit v2-es verzióját használjuk, úgy egy hangszóróval is bővül a kimeneti eszközök tárháza.

3.4. A hivatalos programozói felület bemutatása

A micro:bit elterjedése miatt, a hivatalos felület részletes bemutatása helyett, az adott eszközhöz kapcsolódó új blokkok és azok integrálására szeretnénk fókuszálni a makecode felületén. A blokk alapú nyelv tökéletes lehet arra, hogy bármilyen életkorban használjuk az eszközt és azzal bevezessük a programozást vagy nagyobb motivációt gyakoroljunk a diákokra. A blokkprogramozás hazánkban már a harmadik osztályban megjelenik [1], de van olyan nemzetközi kutatás is, amely bizonyítja, hogy a blokkprogramozásnak számos előnye van akár középiskolában is [11], hiszen egy könnyen olvasható, egyszerű programozási megoldásról beszélhetünk, így remek lehetőséget biztosít a programozás megszerettetéséhez, a motiváció növelésére. A board irányításáért felelős blokkok nem csak felhasználásra elérhetőek, hanem akár lehetőségünk van azok szerkesztésére is, hiszen a Stem:Bithez tartozó programkódok és források elérhetőek egy GitHub repositoryban[13]. A blokkok elnevezései egyértelműek és minimális angol nyelvtudással könnyen megérthető, hogy az egyes blokkokat mire is fogjuk tudni hasznosítani.

Az új blokkokat a kiterjesztésekre kattintva tudjuk elérni. Itt szükséges a repository linkjének beillesztése, ami után már ki tudjuk választani az eszközt (13. ábra).

[13] https://github.com/sbcshop/pxt-stembit



10. ábra: A kiterjesztés csatolása és az új blokkok[14]

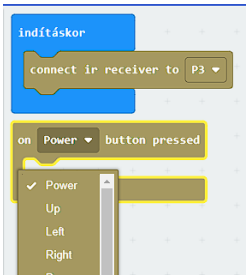
A kiterjesztés hozzáadása után megkapjuk a kódokat, amelyek angol nyelvűek, ezért kisgyermekek esetében ez nehézséget okozhat és szükség lehet tanári magyarázatra. A blokkok felépítésükben nem térnek el a már megszokott micro:bit környezettől, ezért pozitívum lehet az, hogy egy ismerős környezettel találják magukat szemben azok a gyerekek, akik már korábban programoztak micro:bitet vagy használtak blokkos nyelveket.

A felületről egy rövid ismertető megtalálható a csomaggal érkező kézikönyvben, a már korábban említett kilenc építési útmutatón túl, valamint 1-1 képet a szükséges kódsorról, amellyel működésre lehet bírni az eszközt, illetve az útmutatók elején egy listát leírással azokról a blokkokról, amelyet használni fogunk.

A korábban említett távirányítós vezérléshez is szükségünk van a kiterjesztésre és ahogy az Inventor esetében, itt is nekünk kell a különböző funkciókat beprogramoznunk. Erről a következő fejezetben lesz szó bővebben.

3.4.1. Blokkcsoportok bemutatása[15]

Ebben az alfejezetben főként azokra a blokkokra fogjuk a hangsúlyt fektetni, amelyek nem igényelnek egyéb kiegészítőt és a csomagban lévő eszközöket tudjuk velük irányítani. Ilyen blokkcsoportok a távirányítóhoz és az „autó” vezérlő blokkok.



11. ábra: A távirányítót vezérlő blokkok

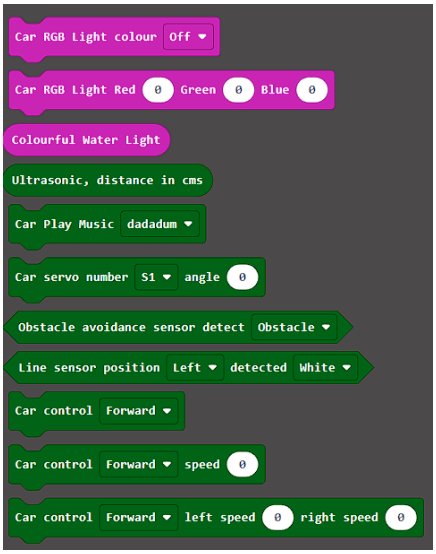
Az első bemutatni kívánt blokkcsoportunk a távirányítóhoz köthető. Itt mindösszesen kettő új blokk szerepel, amik közül az egyikkel gyakorlatilag aktiváljuk a vevőegységet. Tudni kell, hogy a vevő egység a P3-as pinhez van kötve, ezért nekünk is azt kell beállítani a programozás folyamán. A

[14] Forrás: Gaál Bence saját képe

[15] Képek forrása: Gaál Bence saját képei

másik blokknál pedig lehetőségünk van megadni, hogy melyik gomb megnyomásakor mi történjen. Ennek működési elve hasonló a micro:biten lévő gombok vezérléséhez, tehát adott gomb lenyomásakor a blokk belsejében lévő kód fog lefutni (11. ábra).

A *StemBit_Car* blokkcsoport már jóval több elemből épül fel, amelyeket három alcsoportra oszthatunk. Az első a képzeletbeli autónk fényszóróit hivatott vezérelni, ezek bíborvörös színnel vannak jelölve. Ezekkel előre beállított, vagy tetszésünk szerint létrehozott (rgb kód által) színeket tudunk megjeleníteni az eszköz két nagy ledjén. A második csoportba az érzékelőket soroltuk, ugyanis itt van lehetőségünk lekérdezni azt, hogy van-e objektum az akadályérzékelő szenzor előtt, hogy a vonalkövető érzékel-e vagy sem fehér/fekete vonalat, illetve az ultrahangos távolságmérő értékét centiméterben. A harmadik egységgel vezérelhetjük a motorok forgását, akár külön külön is, így lehetőségünk van a kanyarodásra. Itt kapott még helyet egy előre beépített zenéket lejátszó blokk, valamint lehetőség nyílik itt is állítani azt, hogy hány fokot forduljanak a servomotorok, amit az ultrahangos érzékelőhöz hasonlóan külsőleg kell csatlakoztatni az eszközhöz (12.ábra).



12. ábra: A motorvezérlő és érzékelő blokkok

A *StemBit_Music* blokkban tudjuk bekapcsolni a rezgőmotort, míg a *StemBit_Motor* blokkok segítségével a külsőleg csatlakoztatott servomotorok sebeségét, illetve azok fordulati fokát állíthatjuk be. Az *input* blokkok külsőleg csatlakoztatott eszközök kezelésére, míg a *sensor* blokkcsoport az eszközön található infra mellett, a mikrofon és az ultrahang szenzor kezeléseire alkalmas. A *neopixel* blokkok segítségével a boardon található 3 ledet tudjuk működésre bírni. Erre és a nagyobb ledek vezérlésére szintén lehetőségünk van a display blokkok használatával. Ezen blokkok tanórai használatát nem feltétlen javasoljuk a programozás bevezetésénél. Ennek egyik oka, hogy ezek a blokkok komplexitásukat nézve nem feltétlen ajánlottak kezdők számára, illetve az adott eszköz ki és bekapcsolása helyett a pineket vezérelve tudjuk a különböző beállításokat elvégezni.

3.4.2. Milyen feladatokat oldhatunk meg a Stem:Bittel?

A vonalkövető feladatok alapvetően megvalósíthatók az eszközzel, akár csak az objektumok kikerüléséhez kapcsolódó problémák is megoldhatók vele. A párhuzamos programok, pedig az Inventorhoz hasonlóan oldhatók meg ebben a környezetben. A feladatok megvalósításában a szenzorok fajtái, illetve a motorok típusai korlátozzák az eszközt, ahogy ezt a korábbi fejezetekben kifejtettük.

Szintén felmerült itt és a kijelző mérete így a rajzolós feladatok egyszerűsítésre szolgálnak, hiszen a micro:biten is csak 5x5-ös led panel található.

Versenykörnyezetben történő használatban pedig nem lehet összehasonlítani sem a korábban tárgyalt Inventorral, sem pedig az EV3 robottal, mivel a WRO Robo Mission és Robo Sports versenyeken csak és kizárólag a LEGO termékeit használhatják a résztvevők [10].

Főbb felhasználására inkább az osztálytermi órákat javasoljuk, ahol a programozás érdekesebbé tételében, illetve a micro:bit kibővítésében tudja az eszköz igazán megállni a helyét. A blokkos környezetnek hála pedig akár már kisiskoláskorban is használható, ahogy a tesztelés folyamán mi is hasznosítottuk ezt a tulajdonságát, negyedik osztályban mint figyelemfelkeltő és motivációt erősítő eszköz használtuk, míg ötödik osztályban az informatikaóra mellett a természettudomány tantárgyban játszott szerepet mint modellező eszköz.

Számos olyan érdekes projektet lehet vele megvalósítani, ami nem csak a digitális kultúra órán történő használatra korlátozódik. Lánctalpas kialakítása miatt hatásos lehet egy kutatórobot modellezésére, amely méréseket végez, majd adatokat továbbít egy másik eszközre. A micro:bitek segítségével akár olyan járműveket is építhetünk, amelyeket egy kézre erősített másik micro:bittel kommunikálva irányíthatóvá válnak a kézmozdulataink által.

Az eszköz bővíthetőségének komplexitása előnyös lehet a nagyobb korosztályok számára, hiszen a szenzorcsatlakozások és azok pinekkel történő vezérlése az elektronikai és villamossági területekhez kapcsolódó készségeket is erősíteni tudja. Ennek hátránya lehet az, hogy időigényesebb a bővítés, mint egy moduláris robot esetében.

4. Összegzés

Az EV3 kivezetése miatt szükséges, hogy alternatívák után nézzünk. Nyilvánvalóan az egyes robotkészletek felépítése és ára nagyban befolyásolja a választást, de érdemes lehet egyéb szempontok szerint is átgondolni, hogy melyiket is akarjuk használni. A fentebb taglalt két robotcsomag specifikációit (3. táblázat) nézve egy jó lehetőség lehet az EV3 kiváltására, de egyik sem tudja teljeskörűen helyettesíteni azt, így vegyük figyelembe az egyéb tényezőket is.

Szerettünk volna a közismert és már bejáratott LEGO márka mellett egy kevésbé ismert olcsóbb alternatívát is nyújtani. Az oktatásban való hasznosítása mindkét eszköznek nagyban függ attól, hogy milyen céllal is visszük be azt a tanterembe. Ugyan a LEGO Robot Inventor egy drágább lehetőséget biztosít, de mindenféleképpen egy jobban támogatott eszközről beszélhetünk és a márkának megfelelően egy magasminőségű eszközt kapunk. Hátránya, hogy a megvalósítások nem feltétlenül tükrözik azt, hogy ez az eszköz az EV3 utódja lenne. Erősen érződik, hogy a tervezés folyamán nagy behatást gyakorolt az eszközre a Spike termékcsalád, amely a közeljövőben teljes mértékben átveszi a szerepet a LEGO oktatási robot szortimentjében és a gyártó csak erre a termékre fog fókuszálni.

A táblázatból látszik, hogy a Stem:Bit fele annyi építőelemből áll, azonban szenzoros kiterjeszthetőségét nézve sokkal nagyobb szabadságot biztosít. A kevesebb motorszám és a kevesebb építőelem akadályt jelenthet a komplexebb robotok építésében, és mint korábban említettük, gátat szabhat a kreatív alkotásban. A robot ára viszont kevesebb, mint a fele az Inventornak és az oktatásban ez is egy meghatározó tényező.

Ennek ellenére úgy gondoljuk, hogy az osztálytermi feladatokhoz, a programozás megszerettetésére egy kifejezetten jó eszközt kapunk. Érdekes, látványos mérőeszközöket, távirányítású autókat hozhatunk velük létre és jó megoldás lehet különböző interdiszciplináris STEM projektek költséghatékony megvalósítására.

Összeségében tehát, mindkét robot kifejezetten jó oktatási célokra, csak azon belül más-más területet vettek vele célba. Az Inventor egy nagyobb támogatottsággal és szélesebb körben ismert eszköz, amely alkalmazható akár nemzetközi versenyeken is, míg az olcsóbb alternatívájú Stem:Bit a

sokoldalú micro:bitet kiaknázva nyújt teljes élményt minamellett, hogy mindkét eszköz lehetőséget biztosít a tanmenetek által előírt robotikaoktatás lefedésére.

	Robot Inventor (51515)	**Stem:Bit**	**EV3 Home (31313)**	**EV3 Education (45544)**
motorok száma	4	2	2 + 1	2 + 1
motorcsatlakoztatási lehetőségek (portok) száma	6	2 + 3 szervo	4	4
szenzorok száma	2 (beépített) + 2 (moduláris)	6 (beépített)	3 (moduláris)	5 (moduláris)
szenzorcsatlakoztatási lehetősége	6	beépített+4	4	4
vezérlőegység	Hub	micro:bit	Brick/Tégla	Brick/Tégla
áramellátás	akkumulátor	akkumulátor	6 db AA elem	akkumulátor
elemszám összesen	949	307	601	541
ár	(359,99 €[16])	117,95 €[17]	legutolsó ismert ár:[18] (570 €)	legutolsó ismert ár:[19] (745 €)

3. táblázat: A specifikációk összehasonlítása.

Irodalom

1. Kerettanterv az általános iskola 1–4. évfolyama számára- Digitális kultúra 3–4. évfolyam https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_A.docx (utoljára megtekintve: 2022.10.20.)
2. Kerettanterv az általános iskola 5–8. évfolyama számára- Digitális kultúra 5–8. évfolyam https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_F.docx (utoljára megtekintve: 2022.10.20.)
3. Çam, E. & Kıyıcı, M. (2022). The impact of robotics assisted programming education on academic success, problem solving skills and motivation . Journal of Educational Technology and Online Learning , 5 (1) , 47-65 . DOI: 10.31681/jetol.1028825
4. Bradley S. Barker & John Ansorge (2007) Robotics as Means to Increase Achievement Scores in an Informal Learning Environment, Journal of Research on Technology in Education, 39:3, 229-243, DOI: 10.1080/15391523.2007.10782481
5. Solymos Dóra, „LEGO robotok felhasználási lehetőségei az oktatásban," *InfoDidact 2019,* pp. 265-275, 2019.

[16] Forrás: https://www.lego.com/en-de/product/robot-inventor-51515

[17] Hazai forgalmazótól nem kapható, ezért csak euróban tudtuk megadni az árát.

[18] Forrás: https://kockashop.hu/lego/mindstorms-ev3-31313

[19] Forrás: https://kockashop.hu/lego/ev3-lego-mindstorms-education-bazis-szett-oktatasi-robotkeszlet-45544

6. Barbalics Dóra Krisztina, Solymos Dóra, Lego Mindstorms EV3 robotok programozása, ELTE Informatikai Kar, Budapest, 2018. Elérhető: http://tet.inf.elte.hu/tetkucko/wp-content/uploads/2018/12/legomindstorms_szakkorianyag.pdf (utoljára megtekintve: 2022.11.5.)
7. *A Stem:Bit hivatalos értékesítőjének honlapja* https://shop.sb-components.co.uk/products/stem-bit-the-programmable-blocks-kit-for-micro-bit?variant=31064155455571 (utoljára megtekintve: 2022.10.20.)
8. *A Stem:Bit kickstarter honlapja* https://www.kickstarter.com/projects/sb-gajendra/stem-bit/description (utoljára megtekintve: 2022.10.20.)
9. Gaál, B. (2022). Robotics-Enhanced Natural Science in Primary Schools. In: Bollin, A., Futschek, G. (eds) Informatics in Schools. A Step Beyond Digital Education. ISSEP 2022. Lecture Notes in Computer Science, vol 13488. Springer, Cham. https://doi.org/10.1007/978-3-031-15851-3_8 (utoljára megtekintve: 2022.11.08.)
10. General Rules WRO-2022. https://wro-association.org/competition/2022-season/ (utoljára megtekintve: 2022.11.08.)
11. Weintrop, David & Wilensky, Uri. (2015). To Block or not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. https://doi.org/10.1145/2771839.2771860 (utoljára megtekintve: 2022.11.08.)

Attiny13A mikrovezérlő használata C++ szintaktikák oktatásában

Kiss Ádám[1], Kelemen András[2]

[1] kiss.adam@med.u-szeged.hu;
SZTE SZAOK Élettani Intézet
[2] kelemen@jgypk.szte.hu
SZTE JGYPK Informatika Alkalmazásai Tanszék
Szegedi Radnóti Miklós Kísérleti Gimnázium

Absztrakt. Számos tanterv létezik arra, hogy egyszerű algoritmusokkal, egyszerű nyelven, hogy lehet programozási alapokat tanítani. Ezeknek hátránya, hogy vagy a nyelv nem emelhető át teljes mértékben a későbbi tanulmányokba, vagy kevés olyan élményt nyújtanak, aminek a nagyságát vagy okát értik a tanulók. Ebben a dolgozatban olyan eszközöket próbálunk bemutatni, melyek algoritmikus gondolkodás nélkül képesek nyelvi elemeket tanítani. A megszerzett tudással később már az algoritmusok működése tehető középpontba, illetve a nyelvi eszközkészlet élményekre épülő alapjainak bővítése.

Kulcsszavak: C++, élményalapú tanulás, beágyazott rendszerek

Mottó

„Mondd el és elfelejtem; mutasd meg és megjegyzem; engedd, hogy csináljam és megértem."

Konfuciusz

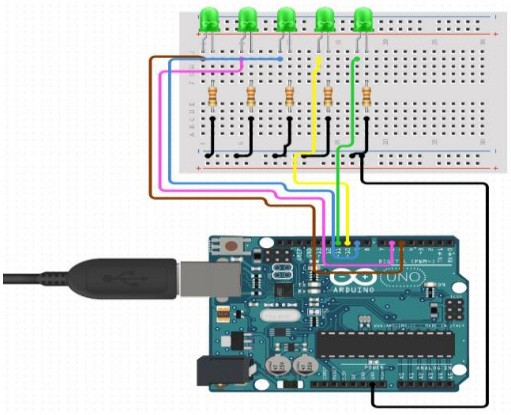
1. Bevezetés

A napjainkban zajló negyedik ipari forradalom erőteljes digitalizációs folyamatokat indított el a gazdasági életben és a társadalomban. Ezen folyamatokra reagálva 2020 szeptemberében bevezetésre került az új NAT, melynek kerettanterveiben az elődjénél hangsúlyosabban szerepel a robotika, valamint az algoritmizálási, programozási ismeretek oktatása [1]. A közoktatásban alacsonyabb évfolyamokon kerül sorra robotika (Lego, Microbit), a blokk programozás (Scratch), valamint a Logo programozási nyelv oktatása. Ezek az eszközök magas szintű vizualitásukkal önálló munkára serkentve a diákot pozitív élménnyé teszik a programozás tanulását. Napjainkban a szoftverek döntő többségét valamilyen szöveges programozási nyelven (C/C++, C#, Java, Python) készítik és a tapasztalatok alapján mind a közoktatásban, mind a felsőoktatásban a kezdők számára a szöveges programozási nyelveken történő programozási ismeretek elsajátítása nagyon nehéz folyamat, annak ellenére, hogy a tanári magyarázatokon, szemléltetéseken kívül több kiváló magyar nyelvű szakirodalom is rendelkezésre áll [2,3,4,5] Ebben a folyamatban első lépésként a tanulónak az adott programozási nyelv legalapvetőbb kulcs-szavait és szintaktikáját kell elsajátítaniuk, melyre építve jöhetnek az alapvető algoritmusok és az algoritmikus gondolkodási mód elsajátítása. Szintén tapasztalati tény, hogy ebben a tanulási-tanítási szakaszban motiváció felkeltése és folyamatos fenntartása nagyon nehéz feladat egyrészt a rendelkezésre álló idő és az előírt tananyag mennyisége közötti ellentmondás, valamint a diákok eltérő tanulási sebessége miatt. További fusztráló tény a diákok számára, hogy programjuk futásának eredménye az operációs rendszer parancssorában jelenik meg és nem a számítógépen kívül valami látványos felületen vagy jelenségben pl. egy robot mozgásában, vagy LED-ek villogásában.

Ebben a cikkben a fent említett problémák áthidalására mutatunk egy lehetséges utat, mely ugyan azt az élményt nyújtja, mint a fent említett robotok vagy a bélyeg gépek programozása, de sokkal olcsóbb hardverrel megvalósítva. Ráadásul ez módszer a programozási ismerteken túl áramkör tervezési és építési ismerteket is adhat az ez iránt érdeklődők számára.

2. Előzmények az élményalapú programozásban

A magyar közoktatásban az alsóbb évfolyamokon már sok éve használnak grafikus programozási nyelveket (Logo, Scratch) a kezdő lépések megtételére a programozásban. A Logo nyelvet 1967 -ben alkotta meg két matematikus és informatikus Wally Feurzeig és Seymour Paper. Ez a nyelv a **Lisp** programozási nyelv könnyített, egyszerűsített változata. Többféle implementációja létezik. Magyarországon a közoktatásban az Imagine Logo terjedt el, melyben egy teknőcöt utasításokkal lehet mozgatni a rajzfelületen. Az utasításokat lépésenként is kipróbálhatjuk, de eljárásokba is szervezhetjük. Az eredmény a teknőc által rajzolt ábra formájában azonnal megjelenik. A Scratch a blokk programozási nyelvek családjába tartozó objektumorientált, interpretált, dinamikus és vizuális programozási nyelv, amelyet elsősorban a programozással ismerkedő gyerekek számára fejleszt az MIT. A Scratch-ben a színpadon alakzatokat és szereplőket helyezhetünk el, a szereplőket mozgathatjuk, eseményeket rendelhetünk hozzájuk.



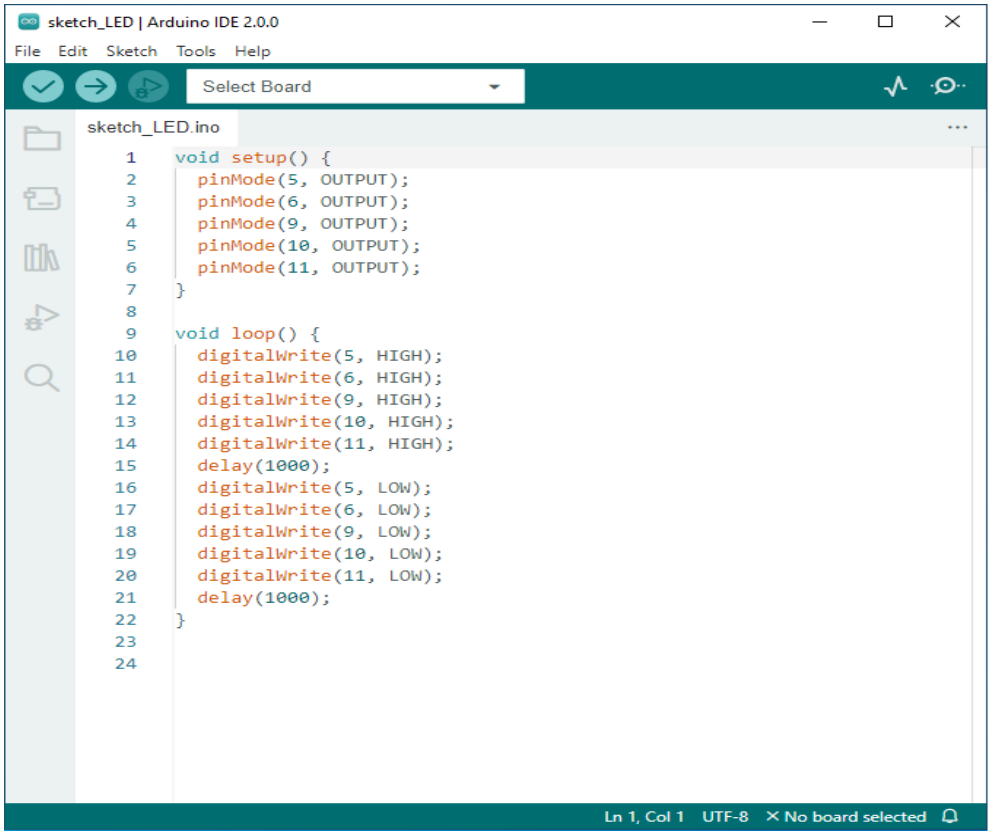
1. ábra: Öt LED bekötése egy Arduino-hoz
(Forrás: Circuit IO)

A LEGO cég 2006 augusztusában mutatta be az első olyan robotépítő készletét [6], mellyel könnyen lehetett játék robotokat építeni és programozni. A robot központi eleme a „tégla", melybe motorokat és érzékelőket lehet csatlakoztatni. A készletben elegendő mennyiségű passzív alkatrész (tartóelemek, fogaskerekek stb.) is található, melyekkel sokféle robot építhető. A robot programozásához a LEGO saját blokk programozási környezetet ad. Ez a környezet a legújabb verziójában (Robot Inventor) a Scratch-hez hasonló programozói felülettel rendelkezik. Ezzel az innovációval a programozás oktatása kimozdult a számítógép zárt dobozából és valódi élménnyé tette azt, valamint elindította a gyerekbarát hobbielektronika forradalmát.

A LEGO innovációjával párhuzamosan indult az Arduino[7] projekt, melynek célja egy teljes, mégis bővíthető beágyazott fejlesztőkörnyezet biztosítása, amibe a hardverek is beletartoznak az elektronikával hobbi szinten foglakozók számára. A fejlesztőkártyába beépített mikrovezérlő programozását egy magas szintű könyvtárral elfedték. Az Arduinoval való foglalkozáson először egy áramkört kell építeni. Ennek tervezésére és szemléltetésére a Circuit IO nevű webes alkalmazás az egyik legszemléletesebb. Itt a kiválasztott alkatrészekhez a szoftver rögtön egy bekötési ábrát rendel

(1. ábra), melynek segítségével az eszköz könnyen összerakható. Az eszköz programozása a szabadon letölthető Arduino IDE-vel egy a C++-hoz nagyon hasonló nyelven viszonylag egyszerűen megoldható.

Az Arduino programok forráskódja két részből áll. Az inicializáló rész (setup) rész egyszer fut le a bootoláskor. Az ún. loop rész folyamatosan ismétlődve lefut (2. ábra).



2. ábra: Az 1. ábrához tartozó LED-ket villogtató program kódja

3. Célkitűzés

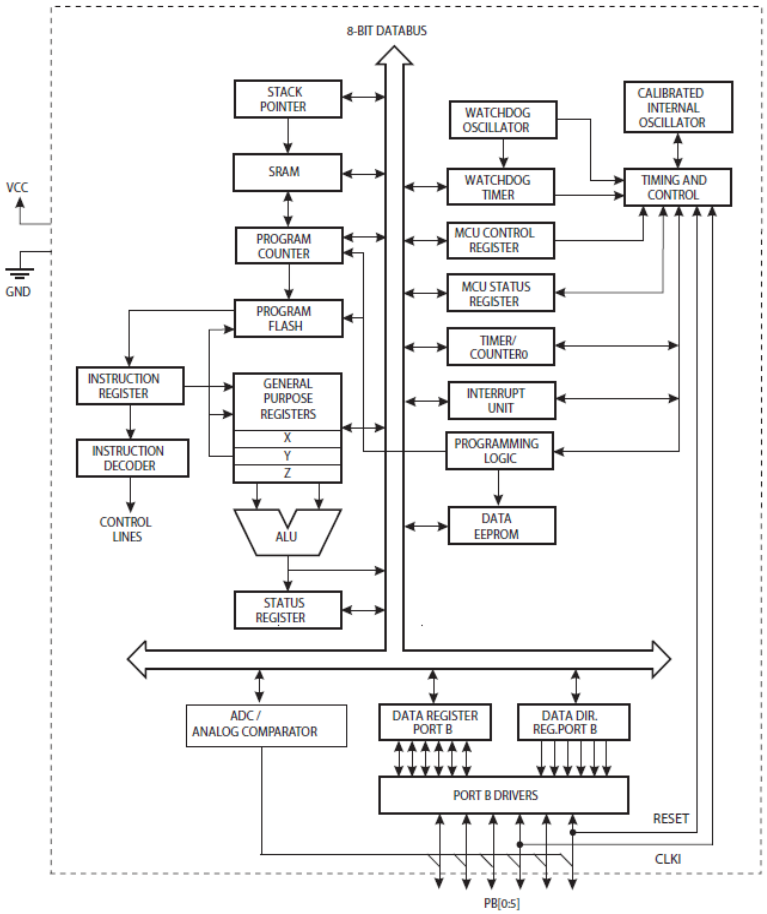
A korábban ismertetett eszközök árban, komplexitásban és újra felhasználhatóság tekintetében legtöbbször nem kielégítőek. Célunk egy olyan eszköz készítése volt, ami megfizethető, valamint transzferálható tudást biztosít, illetve élménypedagógia szempontjából versenyképes a korábban bemutatott Arduino és LEGO termékekkel. Az elkészített eszközt egy nyolc fős csoportban mutattuk be. Célunk volt az élményalapú pedagógia elveit követő demonstráció az utolsó alkalmakon.

4. Alkalmazott módszer

A kitűzött cél megvalósításához egy Arduino szerű, azonban annál kifizethetőbb, valamint kevesebb hardverismeretet igénylő eszköz tervezését, megépítését, illetve annak programozás oktatásban való kipróbálását tűztük ki.

4.1. Áramköri terv

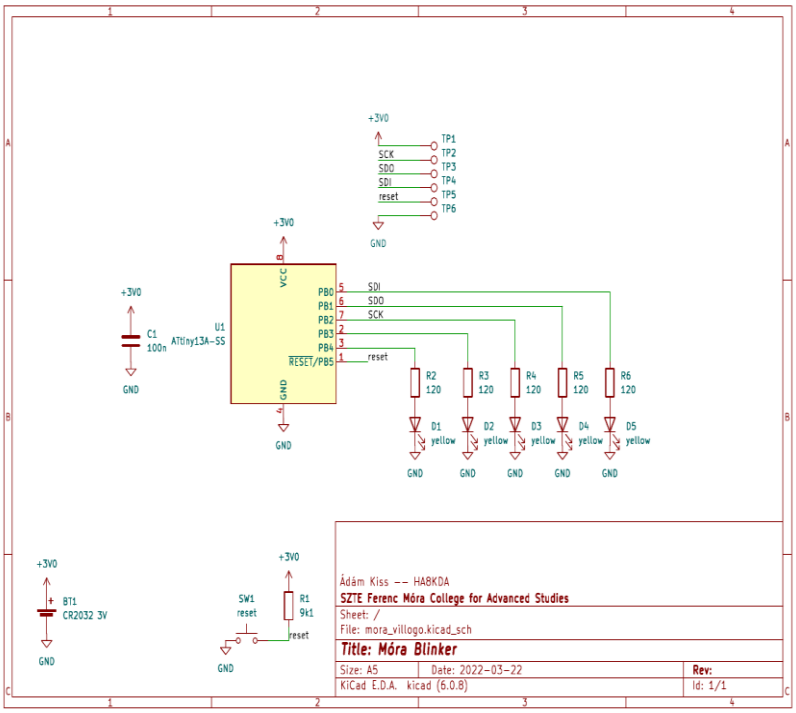
Munkánkban a korábban felsorolt lehetőséget szerettük volna céltudatosan bővíteni. A LEGO robot térbeli mozgása helyett egyszerű vizuális kimenettel láttuk el az eszközt. Az ehhez szükséges áramkör kapcsolási rajzára egy Attiny13A mikrovezérlő (3. ábra) került, ami 2022 tavaszán a legolcsóbb helyben beszerezhető mikrovezérlő volt. A 64 byte operatívmemória, és az 1 kByte flash memóriát elegendőnek ítéltük a célhoz, azaz néhány LED adott utasítássorozat alapján történő kapcsolásához Ezen kívül a szükséges programozó csatlakozók, a LED-ek, illetve egy CR2032-es elemfoglalat kapott helyet a kapcsolási rajzon. A rögzítő furat és a telepes üzem lehetővé teszik, hogy a diákok később díszként használják az elkészült áramkörüket.



3. ábra: Az Attiny13 mikrovezérlő blokkvázlata

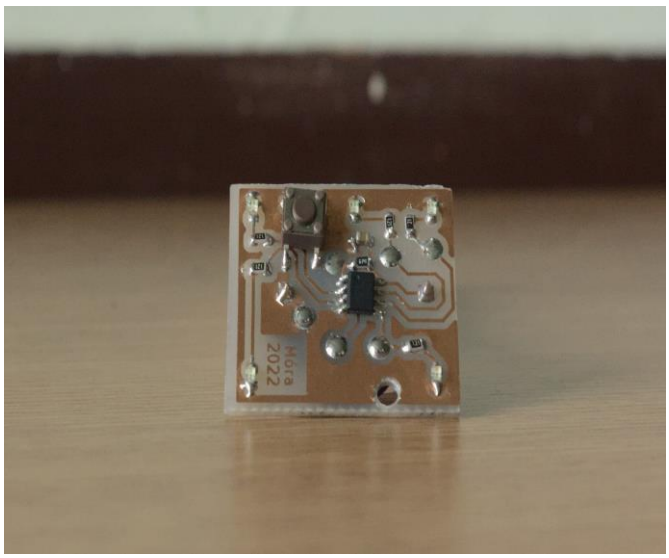
4.2. Megvalósított áramkör

A módszert az SZTE Móra Ferenc Szakkollégiumban próbáltuk ki egy szabadon választható szakkollégiumi kurzus keretében. Az áramköri lapokat az önként jelentkező diákok saját maguknak forrasztották össze a 4. ábrán látható kapcsolási rajz alapján, ami egy élvezetes és különleges élményt adott nekik.



4. ábra: Az elkészült áramkör kapcsolási rajza

Az egyik elkészült példány az 5. ábrán látható.



5. ábra: Egy diák elkészült forrasztása

4.3. Keretrendszer

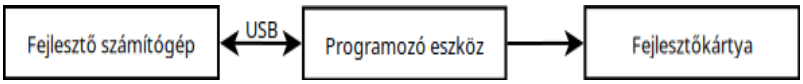
Az oktatási módszer alapgondolata, hogy a nyelvet oktassuk először, és csak később annak használati lehetőségeit. Ennek érdekében a hardver-közeli lépéseket egy keretrendszer biztosítja, maguk a diákok egyszerű kizárólag animációkat írnak egy függvény formájában. A keretrendszert a C++20[4]-as szabványok szerint, illetve annak lehetőségeit használva írtuk. A beépített rutinok egy öt csatornás szoftveres PWM alapú fényerőszabályzást valósítanak meg. A 4. ábra kapcsolási rajzán látható újraindítás gomb megnyomásával a következő animációt tölti be a szoftver.

A diákok a programjukban egy tömbben írnak át számértékeket, melynek eredményeként a számértékkel arányosan változik az ahhoz rendelt LED fényereje. A keretrendszer lehetőséget biztosít mind alacsonyabb szintű, számmal történő, és kvázi-magasszintű asszociatív elérésre is a fényerőket leíró tömbben. Utóbbi fordításidőben kiértékelődik, így nem pazarol erőforrásokat. A keretrendszer tartalmaz továbbá egy késleltető rutint is, melynek segítségével időzíteni is lehet a megjeleníteni kívánt változtatások érvényre jutását.

Magához a program megírásához egyszerű gcc fordítóra van szükség, és GNU Make fordítókörnyezetre. Ezt a párosítást a legtöbb mai fejlesztőkörnyezet támogatja (KDevelop [https://www.kdevelop.org/], Visual Studio Code [https://code.visualstudio.com/], Code::Blocks [https://www.codeblocks.org/] stb.). Egyszerű függvényként kell implementálnia minden diáknak a saját animációját.

5. A fejlesztés folyamatának bemutatása

A fejlesztés folyamán a gazdagépen írjuk a kódot, és egy programozó eszköz segítségével töltjük rá a mikrokontrollerre a 6. ábrán látható módon [8].



6. ábra: Az elkészült program feltöltéséhez szükséges eszközök

A 7. ábrán látható az általunk használt Avr MKII programozó eszköz.

7. ábra: Egy Avr MKII programozó
(Forrás: Microchip)

5.1. A fejlesztőkörnyezet telepítése a gazdagépre

1. Telepítsük az AVR-GCC-t például innen: https://blog.zakkemble.net/avr-gcc-builds/
2. Telepítsünk egy fejlesztőkörnyezetet például innen: https://www.codeblocks.org/
3. Szükséges telepíteni a használt programozó (például 7. ábra) driverét is

5.2. Saját animáció létrehozása

A student mappában hozzunk létre egy hpp fájlt. Ennek az első sorai így nézzenek ki:

```
#pragma once
#include"leds.hpp"
#include"timing.hpp"
```

Definiáljunk egy void(void) típusú függvényt. A LED-ek fényerejét a mappában lévő többi példa alapján lehet állítani egy tömb megfelelő értékeinek átírásával. (Pl.: leds[0]=16) A legmagasabb fényerő értéket a leds.hpp fájl tartalmazza, ami 16. A 0 érték a legkisebb fényerő. A delay_ms függvény segítségével adott millisecundumnyi időkésleltetést vihetünk a vezérlésünkbe.

A main.cpp fájl 25. sorában található tömbbe helyezzük el a saját animációnkat is. Megfelelő animációk közé illesztve bekapcsolás után megtaláljuk majd a sorban a sajátunkat. Amennyiben elfogy a programmemória, úgy néhány animációt törölni kell. Példa az animációk sorrendjét leíró tömbre:

```
static const avr::eeprom_array anim [[gnu::section(".eeprom")]] =
{fade, onoff<500>, rotate<1000>, powerdown, alma, bence, powerdown,
dominik<100>, korte, powerdown};
```

Sikeres fordítás után a használt programozó hardver utasításai alapján égethető a kód a mikrovezérlőbe. Ehhez szükséges a programozó vezetékeinek bekötése is. A bekötési pontok megtalálhatóak a kapcsolási rajzon.

5.3. Tanterv

Az oktatás során az alábbi sorrendben, heti rendszerességgel tartottunk 45 perces alkalmakat:

- Így működik a számítógép (Utasítás-végrehajtási minták)
- Függvény prológusok (ABI-k bemutatása)
- Összetett adattípusok és osztályok memóriaképe
- Fejlesztőkörnyezetek és a GNU Make rendszer
- Fényerőszabályzás alapjai (Impulzusszélesség-moduláció)
- Hogyan tervezzünk animációkat, hogyan valósítsuk meg őket?
- Forrasztás (két alkalom)
- Önálló kódírás

A különböző alkalmak alatt az alapvető programozási tételek [9, 10] (bejárás, elágazás, feltételformálás stb.) is bemutatásra kerültek, gyakorolni csak az utolsó foglalkozásokon volt lehetősége a résztvevőknek. Az utolsó két téma interakciót követelt meg a résztvevők oldaláról, ugyanis ekkor saját kezűleg építették meg a saját eszközeiket. Ekkor már differenciáltan foglalkoztunk az egyes diákokkal, hiszen a különböző készségeik és logikájuk eltértek egymástól.

5.4. Diákmunkák

Az alábbi példát egy pszichológus hallgató írta az órán mutatott módszerek alapján. Teljes mértékben saját maga valósította meg a kódjának első változatát, aholis szekvenciálisan írt utasítások írták

le az állapotok változását. Miután látta, hogy működik, már asszertíven lehetett neki for ciklust mutatni, ami egy egyszerűsítést hozott a kódjába, így az adekvátabb lett.

```
void korte(){
          for(uint8_t i = min_light; i<=max_light; i+=2){
            leds[BOTTOM_LEFT] = i;
            leds[BOTTOM_RIGHT] = i;
            leds[TOP_LEFT] = max_light - i;
            leds[TOP_RIGHT] = max_light - i;

            delay_ms(1000);
          }
}
```

Ezen kívül számos egyéb kód született, minden résztvevő legalább egy animációt írt. A diákok kisebb részben az órán mutatott példák alapján, nagyobb részt a saját maguk által kitalált animáció kódolása közben kapott egyéni segítség alapján készítették el az első változatokat. Ezt követően kis csoportokban egymást segítve tanultak új és adekvátabb nyelvi kifejezéseket.

Az elkészült programokat egy tömbben kell elhelyezni, melyek ezen sorrendben jelentek meg magán a hardveren.

5.5. Gyártás

Az eszköz esetleges gyártatása nemzetközi vagy hazai cégekkel is elvégeztethető. Ilyen például az Eurocircuits Kft. vagy a JLCPCB. A függelékben szereplő tervek alapján így rendelhető a hardver. A programozáshoz egy Atmel MKII vagy STK500 programozóra van szükség, amit külön kell vásárolni. A szoftver fordításához GNU környezetre van szükség.

6. Összegzés

Az értékelést a hallgatók személyes, szubjektív benyomása alapján végeztük. A próbakurzuson résztvevő hallgatók túlnyomó többsége tanult már korábban programozást, illetve egyetlen programozni nem tudó, pszichológus egyetemi BA hallgató is jelentkezett az alkalmakra, aki érdeklődve csinálta végig a kurzust. Külön öröm volt neki, hogy az elkészült áramkört hazavihette, ugyanis az alacsony anyagköltségek ezt megengedték, emellett megtanulta az alapvető programozási nyelvi elemeket, illetve az ezek használatához szükséges alapvető gondolkodásmódot. A korábban programozni tanuló résztvevők (7 fő) is tapasztaltak újdonságérzést, valamint a speciális körülmények (virtuális megjelenítés helyett kézzel fogható eredmény) számukra is élvezetessé tették az alkalmakat.

Összegzésként a módszer további vizsgálatra szorul, a kezdeti eredmények biztatóak. A nyelv ismeretével a diákok már tudnak később tisztán mások algoritmusának megértésére koncentrálni. A konkrét példa a modern C++ szabvány miatt összetett vagy bonyolult adatstruktúrák implementálását is megengedik. Hátránya az áramkörnek például egy Arduinoval szemben, hogy ehhez szükséges egy külső programozó áramkör, illetve a LEGO robotokkal szemben, hogy a használt anyagok és a csomagolás hiánya nem teszik kisgyermekbaráttá. Előnye a nagyságrenddel alacsonyabb ár és közvetlen oktatásra kész szoftver.

Meglátásunk, hogy a módszert szélesebb körben is érdemes megvizsgálni. Az észrevételek összegzését az alábbi táblázat tartalmazza:

	LEGO	**Arduino**	**Ez a fejlesztés**
Inicializálás	Nem igényel inicializálást	Hardverismeretet igénylő inicializáló rutinok	Nem igényel a beépítettem kívüli inicializálást

	LEGO	**Arduino**	**Ez a fejlesztés**
Transzferálható tudás	Nem ültethetők át nyelvi elemek	A megismert nyelvi elemek későbbi munkákban használhatóak	A megismert nyelvi elemek későbbi munkákban használhatóak
Előkészület	Bonyolult hardveres előkészületek	Bonyolult hardveres előkészületek	Egy programozó beszerzése szükséges
Plusz költség	Kb. 150 000 HUF	Kb. 15 000 HUF	Kb 1500 HUF

Irodalom

1. Nemzeti Kerettantervek a Digitális kultúra tantárgy számára https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat
2. Tóth Bertalan: Programozzunk C++ nyelven! Az ANSI C++ tankönyve, ComputerBooks, 2003
3. Juhász Tibor, Kiss Zsolt: Programozási ismeretek, Műszaki Könyvkiadó, 2011, 2015
4. Tamás Ferenc: Visual C# alapismeretek felhasználói szemmel, ISBN 978-963-89484-2-7, Informatika-Számítástechnika Tanárok Egyesülete Budapest 2017
5. Magyary Gyula: Emelt szintű informatika érettségi 2. Python lépésről lépésre, ISBN 978 963 551 028 3, Metropolis Média Group Kft. 2020
6. Lego. NXT Mindstorms. url: https://web.archive.org/web/20060911025537/ http://www.lego.com/eng/info/default.asp?page=pressdetail&contentid=21257&countrycode=2057&yearcode=2006&archive=true. (Utolsó megtekintés: 19.11.2022).
7. Michael Shiloh Massimo Banzi. Getting Started with Arduino. 3. kiad. Make: Community, 2014.
8. Kelemen András: Beágyazott rendszerek programozása C nyelven. [Online educational package (e-learning lesson/topic)] (https://eta.bibl.u-szeged.hu/864/), SZTE Elektronikus Tananyag Archívum, 2011
9. Ivor Horton, Peter Van Weert: Beginning C++20: From Novice to Professional 6th ed. Edition, Apress, 2020
10. Láposi Zoltán, Kiss Tibor: C# programozási segédlet és munkafüzet, Műszaki könyvkiadó Budapest 2019 ISBN 978-963-275-124-5

Függelék

A projekt forráskódja és a részt vevő diákok munkái megtekinthetők online is az alábbi linken: https://github.com/kissadamfkut/mora_blinker

Az áramkör megépítéséhez részletesebb információ a Readme file tartalmaz.

Megjelenés a nemzetközi médiában:
https://hackaday.com/2022/05/14/electronics-and-c-education-with-an-attiny13/

A rendszerszintű gondolkodás fejlesztésének oktatásmódszertani lehetőségei az informatika oktatásban

Korom Szilárd[1], Illés Zoltán[2]

{[1]korom.szilard,[2]illes}@inf.elte.hu
ELTE IK

Absztrakt. Az informatika oktatása során a fejlesztendő kompetenciák között legtöbbször az algoritmikus gondolkodást, a problémamegoldást vagy az alkalmazói képességet említik. A rendszerszintű gondolkodás, mint informatikai kompetencia fejlesztésének lehetőségei ritkán kerülnek elő, pedig annak fontosságáról, jelenlétéről és elkerülhetetlenségéről már számos tanulmány készült. Az alábbi cikkben azt kívánjuk bemutatni, hogy a gyakorlati oktatás során egy tananyagot hogyan lehet úgy felépíteni, hogy az alkalmas legyen a kompetencia fejlesztésére.

Kulcsszavak: rendszerszintű gondolkodás, rendszergondolkodás, informatikai kompetenciák, kompetenciafejlesztés

1. Bevezetés

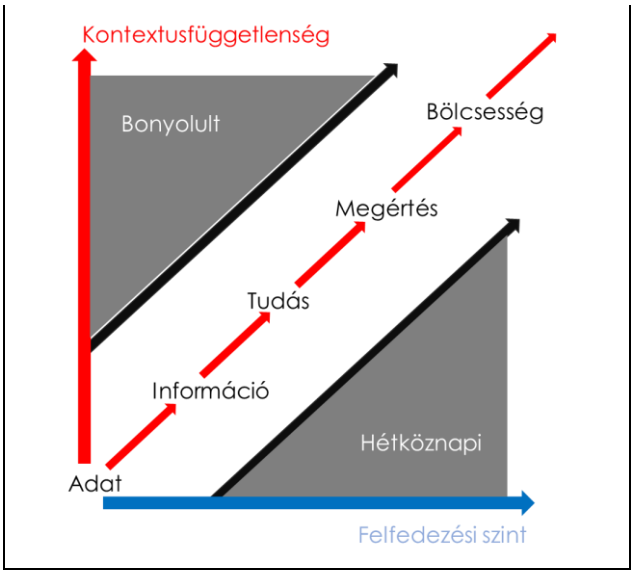
A rendszerszintű gondolkodás egy olyan konkrét kompetencia, mely az tanításmódszertani kutatásokban ritkán jelenik meg, bár már régóta tudjuk, hogy az informatikai kompetenciák egyike [1]. Mindemellett professzionális környezetben egyre részletesebben foglalkoznak vele. A szoftvertechnológiai, szoftverfejlesztés jellegű kutatásokban egyre gyakrabban említik nem csak, mint kompetenciát, de mint képességet, szemléletmódot is. Úgy tűnik ez elengedhetetlen nem csak az informatikával foglalkozó szakember számára, de szinte minden területen, ahol alapvetően komplex rendszerek működtetése, fenntartása a cél, márpedig egyre bonyolultabb rendszerek vannak az élet szinte minden területén, így a rendszergondolkodókra is egyre nagyobb szükség van [2].

A rendszerszintű gondolkodást már számos tanulmányban definiálták valamilyen módon, melyeket Arnold és Wade egyesített [2]. A definíciót lefordítottuk egy korábbi tanulmányunkban [3]:

> *„A rendszerszintű gondolkodás olyan szinergikus analitikus készségek csoportja, melyek javítják egy rendszer azonosításának és megértésének képességét, megjósolják viselkedésüket és módosítják azokat, a kívánt hatások elérése érdekében. Ezek a készségek rendszerként működnek együtt."*

A definíció önmagában kevés a számunkra. Ahhoz, hogy a rendszerszintű gondolkodás fejlesztéséhez alkalmas tananyagokat tudjunk készíteni, szükség lesz egy modellre, mely a rendszerszintű gondolkodást keretbe foglalja, mely útvonalat ad, így ellenőrizni tudjuk, mikor éri el a tanuló a kívánt szintet. A rendszerszintű gondolkodás fejlesztéséhez a DIKUW modellt érdemes párosítani [4]. Ez egy betűszó, ahol a betűk jelentései:

- *Data*, vagyis adat
- *Information*, vagyis információ
- *Knowledge*, vagyis tudás
- *Understanding*, vagyis megértés
- *Wisdom*, vagyis bölcsesség



1. ábra: A DIKUW (Adat; Információ; Tudás; Megértés; Bölcsesség) hierarchia [5]

A 1. ábra egy reprezentációja, hogy a tanuló egy új tananyagelemmel milyen utat járhat be. Például az adatbázis kezelésnél az „adat" azt jelenti, hogy a diák már hallott az Access programról, hallott kulcsszavakat (pl.: tábla, adatbázis), de ezek mögött semmiféle valós megértés, vagy jelentés nincsen. A végcél pedig nyilvánvalóan az, hogy a hallgató képes legyen felismerni problémák, rendszerek esetében pontosan hogyan, milyen adatbázis érdemes használni, tehát létezik egyfajta kompozíciós és dekompozíciós képesség is például. Természetesen a konkrét rendszer definiálja, pontosan milyen képességre lehet szükség. A képességek teljes listáját Arnold és Wade meghatározta a következő fő kérdések mentén:

- Hogyan közelítjük meg a rendszereket és a rendszerszintű problémákat?
- Mi a rendszer, mi van benne, és mi van rajta kívül?
- Hogyan szerveződik a rendszer tartalma?
- Hogyan hatnak egymásra a szervezet, az elemek, azok tulajdonságai és más tényezők, hogy létrehozzák a viselkedést? Mit tehetünk, hogy megváltoztassuk ezt a viselkedést?

Összességében tehát az oktató képes felmérni a rendszerszintű gondolkodás szintjét, ám ami még fontosabb, az 1. ábra, valamint Arnold és Wade listájának segítségével képesek vagyunk tananyagot készíteni, mely alkalmas a képesség fejlesztésére különböző korosztályokban, különböző témakörökön keresztül. A továbbiakban ezeket a lehetőségeket kívánjuk bemutatni részletes példák segítségével.

2. Kutatásmódszertan

A rendszerszintű gondolkodás fejlesztésére alkalmas tananyagokat dolgoztunk majd próbáltunk ki különböző tanulói csoportokkal. A kutatás lényege elsősorban a megvalósíthatóság és a tapasztalatok begyűjtése, rendszerezése volt. Természetesen akkor lenne teljes a kutatás, ha mérni tudnánk, hogy a módszertanunkkal ténylegesen tudtuk fejleszteni a rendszerszintű gondolkodást, illetve azt is,

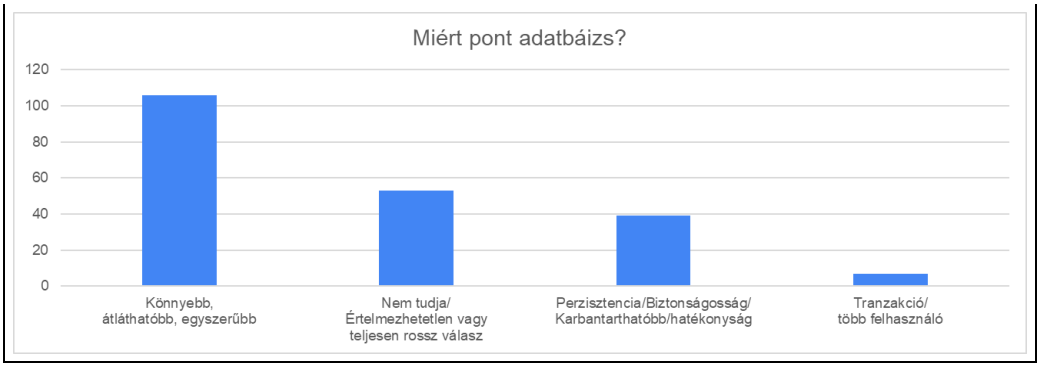
hogy más módszertannal nem tudnánk ilyen hatékonyan fejleszteni azt. Sajnos ezt még nem tudjuk megtenni, mert ezen kompetencia szintjének mérésére nem létezik alkalmas, kvantitatív módszer. Miután elkészülünk egy ilyen rendszerrel, ezeket az igen fontos méréseket el kívánjuk majd végezni.

Az előzőeken felül mélyinterjúkat készítettünk szenior, tapasztalt programozókkal és szoftvertervezőkkel annak érdekében, hogy feltérképezzük, pontosan mire van szüksége egy szoftverfejlesztőnek a munkája során, vagyis az egyetemeknek mire kell felkészíteniük a hallgatókat a programtervező informatikus képzésen.

3. A kompetencia szerepe

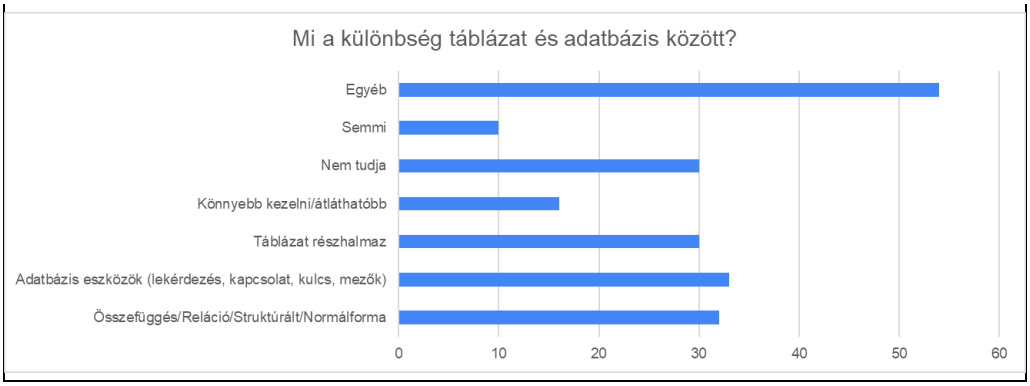
Jelenleg az informatika oktatása leginkább az alkalmazói rendszerekről, az algoritmikus gondolkodásról, illetve a digitális kultúra tantárgy bevezetése óta különböző digitális kompetenciák kialakításáról szól (mint az online kommunikáció, publikálás stb.) [6, 7]. Általánosságban elmondható, hogy a programozás oktatása és az algoritmikus képesség gyakran egyenlőnek van tekintve, vagyis a programozás oktatása az algoritmizálás kompetenciájának fejlesztésére szolgál, vagy azt tekinti fő feladatának. Jól mutatja ezt, hogy a kerettantervben is „Algoritmizálás, formális programozási nyelv használata" szerepel [7]. Egyre több esetben a módszertan a problémamegoldásra helyezi a fő fókuszt [8], azaz a programozást konkrét problémák megoldásán keresztül értelmezi. A fő probléma az, hogy a komponensek működése, a részproblémák megoldása általában nem okozza egy rendszer működését, hiszen sokszor a rendszer struktúrája és nem pedig az egyedi összetevők állapotai határozzák meg a működést [1]. Hiába tanul meg a diák vagy hallgató egyedi komponenseket, ha azokat nem tudja különböző informatikai kontextusokban értelmezni. A tanulás folyamata során kiemelkedő jelentősége van, hogy az új elem illeszkedjen a meglévő tudásbázishoz, beilleszkedjen az informatikáról kialakított tudásgráfba. A kontextus és a rendszerek elhagyásával a hallgatónak nincsen esélye erre, mert ekkor az új tudáselem egy izolált pont lesz a gráfban. A leggyakrabban ekkor a hallgató arra panaszkodik, hogy nem tudja mire jó ez az új tudás, hol és hogyan lehet alkalmazni vagy egyáltalán mi az értelme. Ha azonban már hozzá tudja kötni valamihez, akkor a motivációja is emelkedik, hiszen az új tudáselem egy korábbi problémára vagy egyszerűen egy lyukas foltra kínál megoldást.

Így fordulhat elő például az, hogy bár a diákok tanulják az Access-t középiskolában, mégsem értik mi a különbség egy relációs adatbázis és egy táblázat között [9]. Elsőéves hallgatók a 2. ábrán látható válaszokat adták arra a kérdésre, hogy általában miért adatbázisban tároljuk az adatokat és nem fájlokban. Az ábrán jól látszik, hogy a hallgatók több mint háromnegyedének gyakorlatilag fogalma sincs a helyes válaszról.



2. ábra: A „miért adatbázist használunk táblázat helyett" kérdésre adott válaszok arányai [9]

A 3. ábrán jól látható, hogy az Access oktatása középiskolában olyannyira sikeresen épít a táblázatkezelésre, hogy a hallgatók kétharmadának nem igazán sikerül értően megkülönböztetni a kettőt.



3 ábra: A „mi a különbség a táblázat és adatbázis között" kérdésre adott válaszok arányai [9]

Ebben a helyzetben a diákok valószínűleg meg tudnak oldani Access feladatokat, képesek felismerni egy adatbázist, sőt, összetett feladatokat is meg tudnak oldani, mégsem illesztették bele az informatikáról alkotott nagy képbe. A következő fejezetekben konkrét tananyagelemekkel kapcsolatban kívánjuk bemutatni ezt a problémát, valamint példákat kívánunk adni arra, hogyan lehet a rendszerszemléletet átadni.

4. Módszertani lehetőségek és példák a kompetencia fejlesztésére

4.1. Szövegszerkesztés – Word

A rendszerszintű gondolkodás fejlesztése a Word oktatása során azt jelenti, hogy nem az az egyetlen cél, hogy szövegszerkesztési problémákat oldjunk meg (pl.: hogyan lehet megoldani, hogy egyik oldal fektetett legyen, míg a másik állított, hogyan lehet önéletrajzot készíteni, meglévő sablonokat felhasználni), sem pedig, hogy növeljük az alkalmazói képességet, hanem hogy tanuljunk valami általánosat a szövegekről, az információ ábrázolásáról, formázásáról. Vegyék észre a diákok a kapcsolatot a Word és egy csevegő alkalmazás beviteli mezője, vagy egy fejlesztői környezet (IDE) között. A fontos fogalmak ezzel a megközelítéssel tehát sokkal inkább a gyűjtőfogalmak, mint az alakzatok, karakterszintű formázások, bekezdésszintű formázások, szakaszszintű formázások stb. A lényeg tehát, hogy amikor a diák elkészít például egy önéletrajzot, akkor ne a Word sajátosságiból induljon ki, hanem általános elképzelésekből, melyekhez egy, de nem az egyetlen eszköz a Word.

4.1.1. Óravázlat

A következőkben egy konkrét órapéldát kívánunk mutatni arra, hogy hogyan lehet a rendszerszintű gondolkodás kompetenciáját fejleszteni a szövegszerkesztésen keresztül.

Az óra a 8. évfolyam diákjai számára készült, mint a szövegszerkesztést bevezető óra. Ehhez mérten épít a következő alapismeretekre:

- Alapvető karakterformázások.
- Alapvető bekezdésformázások.
- Képek beágyazása a dokumentumba
- Bekezdés/oldal szegélyezés

Az 1. táblázat annak az órának a tervét mutatja, melyet élesben is megvalósítottunk, s mely véleményünk szerint a rendszerszintű gondolkodás fejlesztését tűzi ki célul.

Idő	Tananyag	A tevékenység célja	Munkaformák és módszerek	Megjegyzés
0–2 perc	A terem elfoglalása	A diákok bejönnek, elfoglalják a helyüket, felkészülnek az órára.	-	
2-5 perc	Köszönés, névsor, bevezetés	Bemutatkozás, jelenlévők ellenőrzése. Az előttünk álló téma megnevezése, rövid bemutatása	Oktató által irányított	
5-13 perc	Rövid bemutató	A téma felvezetése, gondolatébresztés. Mi mindenre használjuk a szövegszerkesztést?	Frontális	A mellékelt PowerPoint bemutató alapján.
13-15 perc	Feladat kiadása	A szöveg szerkezetének megértése	Frontális	A feladat: Az ismert műveleteket próbáljuk meg valamilyen szempont alapján kategorizálni. Mik azok, amik egy csoportba tartoznak?
15-20 perc	Feladatmegoldás	A téma önálló feldolgozása	Páros munka	
20-26 perc	A diákok válaszainak feldolgozása	Az alapvető fogalmak bemutatása, a szerkezet megismerése.	Ellenőrzés. Használandó eszköz: Word-SmartArt	Amennyiben a diákok által felállított kategóriák nem felelnek meg a valóságnak, a tanár finoman korrigáljon. Cél, hogy a szekció végére elkészüljön egy olyan struktúra, mely jól bemutatja az alapvető fogalmakat.
26-27 perc	Az ábra lerajzolása a füzetbe	Az elmélet rögzítése, hogy később visszakereshető legyen	Egyéni	A táblázat után látható a várható ábra
27-29 perc	Az önálló feladat kiadása, közben gépek bekapcsolása	Felkészülés az önálló feladatmegoldásra	Frontális	A feladat: Mindenki készítsen el egy dokumentumot a mellékelt nyers szöveg alapján[1]. Formázza úgy, ahogyan az szerinte a lehető legigényesebb, olvashatóbb. Az elkészült művekből a legigényesebbeket, a következő óra elején elemezni fogjuk.

[1] Maga a konkrét szöveg nem annyira lényeges, bár nem árt, ha témakör érdekli a diákokat.

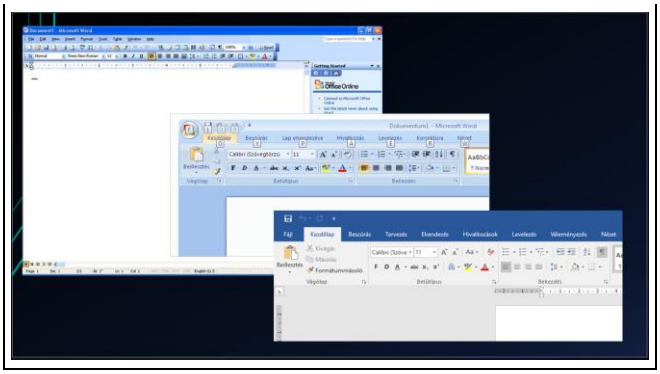
29-44 perc	Önálló feladat megoldása	Az eddig tanultak gyakorlati felhasználása, azaz egy olyan szemléletre való kitekintés, ahol a szövegszerkesztést nem instrukciók alapján csináljuk, hanem felhasználjuk a megszerzett tudást.	A diákok egyénileg dolgoznak. Az oktató járkál, ellenőrzi a munkavégzés folyamatát és segít, ha kell.	Az elkészített munkáknak a következő órán lesz nagy szerepe, hiszen ezekből néhányat közösen fogunk elemezni.

1. táblázat: Szövegszerkesztéses óravázlat a rendszerszintű gondolkodás fejlesztésére

A bevezető órának a lényege, hogy a meglévő alapvető információkra építve a program általános felhasználhatóságára és céljára helyezze a hangsúlyt. A diákok fő feladata a funkciók csoportosítása, s nem pedig pl. konkrét funkciók megtanulása különböző példákon, problémákon keresztül. Az oktató feladata a kontextus bemutatása és az összegző munka terelgetése és kiegészítése. Ehhez mérten az első óra legelején lévő igen rövid bevezetőhöz készült bemutató is összehasonlító, összegző diákat tartalmaz általánosságban a szövegszerkesztésről. A cikkben két példát kívánunk bemutatni a hatból.

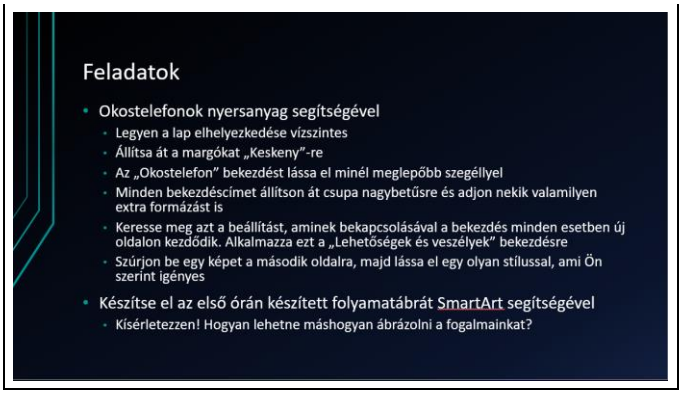
4. ábra: Az a dia a szövegszerkesztés órájának bevezető bemutatójából, mely a szövegek általános felhasználását mutatja be.

A 4. ábrán látható diának az a lényege, hogy rámutasson arra, hogy a szövegszerkesztés messze túlmutat a Wordon, de még a standard szöveges dokumentumokon is (pl.: videókban, logókban, forráskódban).



5. ábra: A szövegszerkesztő felületének mellékességét szemléltető dia

A 5. ábra a Word felhasználói felületének evolúcióját mutatja be, mellyel az oktató érzékeltetheti mennyire nem az a fontos, hogy a diákok azt tanulják meg, mi milyen beállítás hol van, hogyan hívják.



6. ábra: A szövegszerkesztő óra önálló feladatai igen vegyesek

Az óratervezetben, így a 6. ábrán az is megjelenik, hogy a szövegszerkesztésnél (rendszerszemlélettel) nincsen különbség a karakterszintű formázások és a szakaszszintű formázások között, hiszen ezek csupán 1-1 lehetőségek egy konkrét alkalmazásban. Ez a gyakorlatban azt jelenti, hogy ezeket a lehetőségeket egyáltalán nem szükséges külön-külön oktatni. Az óra tartása közben is azt tapasztaltuk, hogy a diákoknak egyáltalán nem okozott gondot a 6. ábrán látható feladatok vegyessége, ahogyan az sem, hogy eddig ismeretlen eszközöket kellett használniuk.

4.2. Táblázatkezelés – Excel

A rendszerszintű gondolkodás fejlesztése az Excel tanítása során azt jelenti, hogy valami általános szemléletet igyekszünk kialakítani a diákokban az adatok rendszerezéséről, értelmezéséről, feldolgozásáról és ábrázolásáról. Ha például a diákok kapnak egy mintafájlt válogatott és rendszerezett adatokról, amit be kell olvasniuk a táblázatkezelőbe (ahogyan az például az érettségin is lenni szokott [10], s ehhez mérten a gyakorló órákon is), akkor ezt a szemléletet nem kapják meg, hiszen a rendszerszintű gondolkodás szerinti legfontosabb feladatot nem ők végzik el. Az adatok összegyűjtése és rendszerezése a végeredmény szempontjából úgy, hogy a feldolgozáshoz szükséges függvényeket, lekérdezéseket hatékonyan meg lehessen csinálni maga a rendszerszintű gondolkodás (amit például egy tanár csinál egy dolgozat összeállításánál).

4.2.1. Óravázlat

A következő rendszerszintű gondolkodás fejlesztésére alkalmas példaóra kiválóan megvilágítja az imént említett problémát. Az óra a 9. évfolyam diákjai számára készült, mint a táblázatkezelést bevezető óra. Ehhez mérten épít a következő alapismeretekre:

- Alapvető formázások, cellaformázás
- Cellahivatkozások
- Alapvető, egyszerű függvények (pl.: Átlag, Darabteli, Min, Max)
- Egyszerű diagrammok

A 2. táblázat az óra vázlatát tartalmazza.

Idő	Tananyag	A tevékenység célja	Munkaformák és módszerek	Megjegyzés
0–2 perc	A terem elfoglalása.	A diákok bejönnek, elfoglalják a helyüket, felkészülnek az órára.	-	A gépet már most bekapcsolhatják
2-20 perc	Bevezetés	Bevezetés. A témakör szerepének áttekintése, ismétlés.	Oktató által irányított beszélgetés.	A következő kérdések alapján: - Mit tanulunk éppen? - Mi ennek a szerepe az informatikában? Miért fontos? - Miben más ez, mint egy normál táblázat? - Mik a táblázatkezelés sajátosságai? - Mik azok a függvények, elágazások?
20-30 perc	Egy témakör táblázatos absztrahálása.	A „táblázatos logika” elmélyítése, megközelítések összehasonlítása, esetleges félreértések tisztázása.	Páros munka.	5 percet kapnak a diákok, hogy a következő témakörben a szerintük leglogikusabb táblázatot összerakják mintaadatokkal: Iskola, osztály, diákok
30-40 perc	Milyen funkciókat várunk el.	Az elkövetkezendő függvények megtanulására ő maguknak legyen igénye.	Oktató által irányított beszélgetés.	Egy konkrét adathalmazból kiindulva/általánosítva fogalmazzuk meg, milyen (további)függvényeket várunk el a programtól. A konkrét adathalmaz mellékelve.
40-45 perc	Órazárás, gépek kikapcsolása, összefoglalás.	Az átbeszéltek alapján összefoglalás, az említett fogalmak elmélyítése.	Oktató által irányított beszélgetés.	

2. táblázat: Táblázatkezelés óravázlat a rendszerszintű gondolkodás fejlesztésére

Hasonlóan a szövegszerkesztős példához, ez az óra is 3 nagy részre osztható:

- Az eddig megtanult alapok rendszerezése, kontextusba helyezése
- Egy rendszer jellegű probléma megoldása
- Reflexió, melynek a végére a diákok a DIKUW modell szerinti „megértés" szintű kérdésekre válaszolnak

A táblázatkezelés esetében a rendszerprobléma az adatok „jól" szervezése volt. A 3. táblázat egy, a diákok által készített táblázatokból készült (a neveket kicseréltük, hiszen a diákok a sajátjaikkal töltötték fel). Ez a strukturális alapelv az 5 kiscsoportból két helyen megjelent.

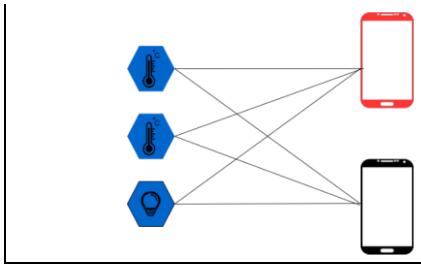
Névsor	**9.a**	**9.b**	**10.a**	**10.b**	**11.a**	**11.b**	**12.a**	**12.b**	**Összes**	**Átlag**
	Horváth Anna	Gábor Anna	Kiss Kincső	Tóth Ádám	Nagy Bence	Belső Donát	Orbán Etel	Pintér Emese		
	Kovács Mária	Pécsi Gábor	Varga József			Farkas Szonja	Vass Napsugár	Bakos Barna		
	Napos Ádám		Szabó Emese							
	Nagy Míra									
Létszám	4	2	3	1	1	2	2	2	20	2.125

3. táblázat: A diákok által készített táblázat

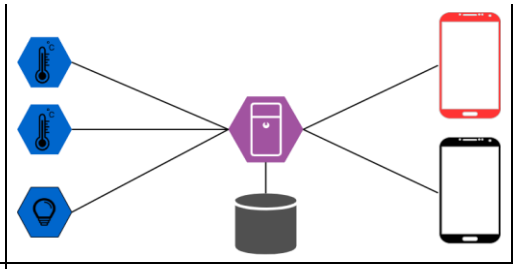
A táblázat felosztása nem teljesen rossz, hiszen csupán annyi instrukciót kaptak, hogy „Iskola, osztály, diákok" (bár élőben említésre került, hogy az nincsen rögzítve, pontosan milyen adatokat akarunk tárolni). A táblázat képes befogadni névsor adatokat osztályonként, de gyakorlatilag semmi másra nem képes. A legnagyobb probléma a táblázattal az, hogy nem bővíthető, további diák-, osztály-, iskolaszintű tulajdonságokkal nem kiegészíthető. Persze mivel nem relációs adatbázisban vagyunk, nem feltétlenül szükséges ilyen megkötéseket tennünk. A lényeg itt sokkal inkább az, hogy a diákok számára nem magától értetődő, hogy a feladatokban miért szinte mindig a normálformált adatstruktúra látható. Mivel az óra közben természetesen vetődött fel ez a helyzet, ráadásul „jó" megoldás is született, ezért értelmesen tudtuk összehasonlítani, sőt, a diákok tudták levonni a következtetés mikor melyik jó.

4.3. Informatikai rendszerek

Egy korábbi cikkünkben arra mutattunk példát, hogy komplex informatikai- vagy szoftverrendszerek eljátszhatók a diákokkal tantermi környezetben gyakorlatilag eszköz és előismeret nélkül [11]. Az ilyen játékok lényege, hogy a diákok valós tapasztalatot szerezzenek, játékosan fedezzenek fel megoldásokat, szembesüljenek buktatókkal [12]. A játék során a diákoknak 1-1 algoritmus, alkalmazás hardver vagy felhasználó szerepét kell felvenniük és közreműködve kell szimulálniuk egy informatikai rendszer működését. Ehhez csupán annyi szükséges, hogy az oktató kitalálja a csoport számára legérdekesebb, de az eszközökhöz és helyhez mérten leginkább megvalósítható informatikai rendszert. A fentebb említett cikkben mi egy okosotthon projektet választottunk, melynek lehetséges architektúrái a 7. és 8. ábrán láthatók.



7. ábra: Okosotthon architektúra szerver nélküli változata



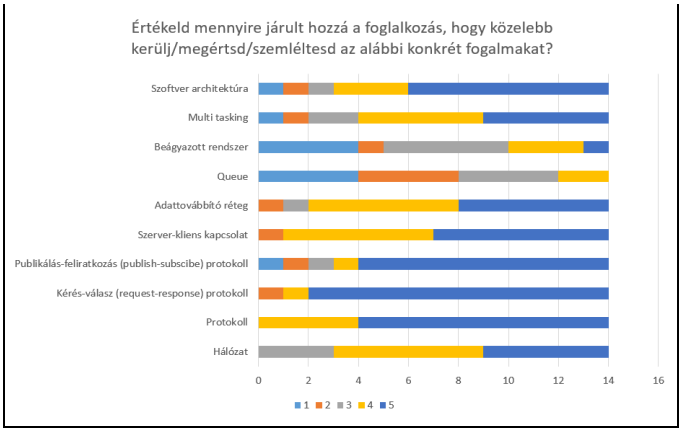
8. ábra: Okosotthon architektúra szerveres változata

A játékot kipróbáltuk első féléves informatikatanár szakos hallgatók között. A tapasztalatok nagyon pozitívak voltak. A hallgatóknak tetszett a játék, mely a 9. és 10. ábrán látható visszajelzésekből készült diagrammokon is látható. Aktívan részt vettek benne és tényleges szimulációját tudták adni egy valós informatikai rendszer működésének. Nekünk, mint oktatóknak csupán a terelgetés volt a szerepünk valamint az informatikai analógiák hangsúlyozása.

9. ábra: A „hogy érezted magad” kérdésre adott válaszok arányai

10. ábra: A hallgatók által adott értékelések a foglalkozásra

A 11. ábrán látható a visszajelzésekről készített diagram. Ez alapján elmondható, hogy ezen konkrét informatikai rendszer által érintett legfontosabb fogalmakhoz tartozó megértést sikerült elmélyíteni a hallgatókban



11. ábra: A hallgatók által adott értékelés az egyes informatikai fogalmak szemléletének sikerességéről

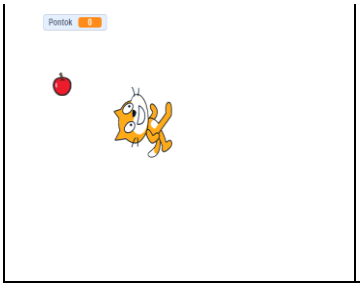
4.4 Programozás

A programozás esetében a rendszerszintű gondolkodás fejlesztése nagyon tág határok között mozog, hiszen a programok, szoftverrendszerek nagyon sokféle rendszert alkothatnak és gyorsan bonyolódnak. Oktatási környezetben bizonyos rendszerek szimulációja igen nehéz. Például azt bemutatni, hogy a shell script programozás hogyan illeszkedik az informatika nagy egész rendszerébe, pontosan milyen helyzetekben hasznos vagy elengedhetetlen nagyon nehéz. Tantermi körülmények között sem a konténerizált világ, sem a szoftverrendszerek funkcionális, automatikus tesztelése, sem pedig az operációs rendszerek működtetése nem igazán megvalósítható. Teljesen más rendszerről van szó, ha a beágyazott vagy alacsonyszintű programozásról, a webfejlesztésről, vagy magasszintű nyelvekről van szó. A rendszerszintű gondolkodás fejlesztése szoftverrendszereken, fejlesztésen keresztül tehát nagyban függ az oktató céljaitól, a csoport összetételétől, érdeklődési körétől és életkori sajátosságaitól. Ez persze inkább „feature" mint sem „bug", hiszen az oktató által kellően skálázható, modularizálható, spirális tananyag készíthető.

Mivel a szoftverrendszerek gyorsan bonyolódnak, ráadásul a technológiai „stack" (verem) is hamar nagyra duzzadhat, az oktatónak alaposan meg kell válogatnia, milyen rendszert visz be a diákok számára. Ahogyan azt a korábbi fejezetekben is említettük, a rendszerszintű gondolkodás fejlesztése nem azt jelenti, hogy bármilyen bonyolultságú, nehézségű rendszer bevihető a diákok számára. A nehézség pont abban rejlik, hogy a feladat maga csak annyira legyen bonyolult, amit a diákok javarészt saját erőből, minimális oktatói segítséggel/terelgetéssel képesek részeire bontani, a meglévő elemeket felismerni, vagy a rendszer működését módosítani, kiegészíteni. A szoftverrendszerekkel tehát az a probléma, hogy nehéz olyan kézzel fogható, könnyen érthető példát találni, mely könnyen skálázható, alakítható az oktató aktuális igényeinek megfelelően.

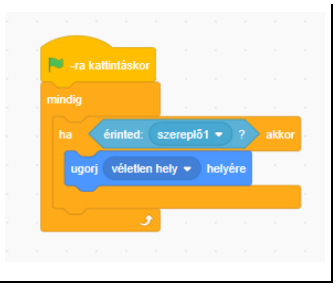
4.4.1. Scratch példa

Meg kell jegyezni azonban, hogy nem csak szoftverrendszerekkel lehet a rendszerszintű gondolkodást fejleszteni. A Scratch[2] környezetben is lehet készíteni olyan játékokat, mely rendszerproblémákat hoznak elő, pedig maga a felület nehezen nevezhető komplex rendszernek. Ilyenek azok a játékok, ahol több szereplő van interakcióban egymással, reagálnak valahogyan egymás viselkedésére, hiszen könnyen „race condition", vagyis versenyhelyzet állhat fenn, ami az egyik leggyakoribb rendszerprobléma a szoftverfejlesztés világában. Ezt könnyen demonstrálja a 12 - 14.ábrán látható példa, ahol akkor kapunk pontot, amikor az egyik szereplőnk elkapja a másikat, ami az elkapásra úgy reagál, hogy egy véletlen helyen megjelenik.



12. ábra: A Scratch játék kinézete.

13. ábra: A macska kódja (irányító gombokat kezelő kódjain kívüli)



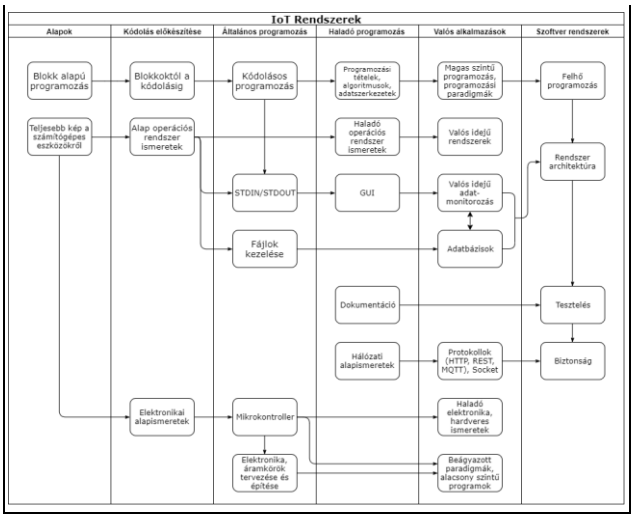
14. ábra: Az alma kódja

[2] https://scratch.mit.edu

Ez a megoldás problémás, hiszen nem determinisztikus a viselkedés, habár külön-külön a komponensek algoritmikusan és szemantikailag is helyesek. A kimenet attól függ, hogy egymáshoz képest a két szereplő mikor ér el a feltételben szereplő utasítás végrehajtására. Ha a macska kódja nagyon gyors, akkor akár több pontot is kaphatunk, mire az alma másik helyre ugrik, de ha az alma kódja nagyon gyors, akkor hamarabb elugrik, minthogy a macska érzékelte volna az érintkezést, vagyis egyetlen pontot sem kapunk. Végső soron tehát az ilyen helyzetek eredménye attól függ, hogy a párhuzamos folyamatok mikor és mennyi processzoridőt kapnak, vagyis az ütemezőtől és a rendszer aktuális állapotától

4.4.2. Szoftverrendszerek

Ahogyan korábbi cikkeinkben is javasoltuk, a beágyazott és valós idejű, vagyis „IoT" rendszerek kiválóan alkalmasak a kompetencia fejlesztésére [13]. A 15. ábra jól bemutatja mennyire könnyen alakíthatóak ezek a példák.



15. ábra: IoT rendszerekkel oktatható informatikai modulok.

A 15. ábra azt mutatja be, hogy az IoT rendszereken keresztül folytatott oktatás milyen komponenseket érinthet, mikre helyezheti az oktató a hangsúlyt.

Mi az alábbi példát próbáltuk ki mesterképzéses hallgatókkal az Eötvös Lóránd Tudományegyetemen. A csoport létszáma 8 fő volt. Okosotthon projekteket szerettünk volna építeni a hallgatókkal Raspberry Pi eszközökkel. Itt a hangsúly a valós idejű rendszereken volt.

Óra	Leírás
1.	**IoT bevezetés** Témakör bevezetése, elméleti háttér áttekintése. Egy minta program bemutatása, mely egy valós életbeli IoT projektet demonstrál **Hardverelemek összeépítése** A hardverek összeépítése, összekötése, élőben kipróbálása mintakódokkal. • Hogyan kommunikáljunk Sense Hat modullal (szenzor adatok begyűjtése, LED mátrixra kirajzolás) • Hogyan vezéreljük a kamera modult? tkinter Python GUI csomag alap funkcióinak kipróbálása
2.	**MQTT protokoll** MQTT broker installálása, konfigurálása és helyi hálózati kommunikációra használása. A publish-subscribe filozófia megismerése, események és eljárások használata az előző órai példák felhasználásával.
3.	**Felhő szolgáltatások** Firebase nem-relációs valós idejű adatbázis konfigurálása és integrálása egy okos otthon szimulációs projektbe.
4-6.	**IoT Lab** Önálló, valóság közeli rendszerek tervezése és implementálása két fős csapatokban.

4. táblázat: Mesterképzéses tananyag: IoT rendszerek oktatása

A 4. táblázat bemutatja a 6 órás blokk általános tervezetét, melyből jól látszik, hogy a modul egy lehetséges végleges projekt demonstrációjával kezdődik. A következő órákon pedig 1-1 önmagában értéket képviselő komponenssel foglalkozunk, melyeket a hallgatók rögtön tudják azonosítani, hogy hova tartoznak. Az utolsó órák pedig biztosítják, hogy biztosan képesek legyenek a megtanultakat együtt alkalmazni.

4.5. Egyéb módszertani fogások

Idáig konkrét példákon keresztül próbáltuk bemutatni a rendszerszintű gondolkodás fejlesztésének lehetőségeit. Vannak azonban általános lehetőségeink is, amit szinte mindenhol lehet alkalmazni. Erre már készültek tanulmányok, amik a gondolattérkép, visszajelző hurkok, ok-okozati térképek használatainak hasznosságát hangsúlyozzák ki [14]. A lényeg tehát, hogy gyakorlatilag minden tananyagelemről elmondható, hogy beilleszthető valamiféle kontextusba, hozzáfűzhető egy más meglévő tudáshoz vagy egymáshoz. Ha ezt a munkát a diákok végzik el az oktató segítségével, akkor a tudás megértéssé válik, hiszen annak értelmezését az ábrák elkészítésével a diákok elvégzik.

5. Visszajelzés az iparból

Készítettünk mélyinterjút 7 tapasztalt programfejlesztővel. A beszélgetések előtt direkt nem említettük, hogy a rendszerszintű gondolkodás lesz a téma, hogy ne befolyásoljuk őket. Az interjú a következő pontok mentén zajlott:

- Milyen olyan komplex/emlékezetes problémák jutnak eszedbe, amelyeket meg kellett oldanod, mint fejlesztő?
- Mire volt szükséges, hogy megold ezeket a problémákat?
  - Kompetencia
  - Szakmai tudás
  - Egyetemi tantárgy
  - Egyéb
- Fontosnak tartod az egyetem elvégzését?
- Hogyan lehetne az egyetemi képzést jobbá tenni?

A beszélgetések teljes kifejtése nem fér ezen cikk kereteibe (és szeretnénk is még bővíteni az interjúalanyok számát), azonban ami számunka jelen esetben releváns, hogy a hétből minden alany említette a rendszerben való gondolkodás fontosságát. Volt olyan fejlesztő, aki az algoritmikus gondolkodást nem is említette (habár annak fontossága megkérdőjelezhetetlen), tehát elmondható, hogy a rendszerszintű gondolkodás a programozói életben a legfontosabb kompetenciák közé tartozik. A megkérdezettek hangsúlyozták az egyetem fontosságát, visszatérő kritika viszont az volt, hogy csak később derült ki számukra az, hogy 1-1 tantárgy milyen fontos is valójában. Ahogyan ezt korábban is említettük a kontextus, a rendszerek elhagyására, nem hatékony bemutatására utalhat.

6. Összefoglalás

A cikkben megpróbáltuk a szövegszerkesztésen, táblázatkezelésen és programozásoktatáson, valamint általánosan használható eszközökön keresztül bemutatni, milyen lehetőségeink vannak a különböző korosztályok esetében a rendszerszintű gondolkodás fejlesztésére. Hogy az általános elveket alátámasszuk, konkrét tananyagpéldákat készítettünk és próbáltunk ki élesben a nyolcadik, kilencedik évfolyam diákjaival, valamint az egyetemi mesterképzés hallgatóival. A teszteléseink a megvalósíthatóságot és a szükségességet bizonyították. Annak igazolása, hogy a módszertan tényleg hatékony még hátravan, hiszen nem áll rendelkezésre alkalmas mérési módszer. Az azonban kijelenthető, hogy a rendszerszintű gondolkodás kompetenciáját lehet fejleszteni általános iskolától az egyetemig, az informatikán belül minden témakörön keresztül.

A rendszerszintű gondolkodás fejlesztését előtérbe helyező tananyagoknak szükségük van némi alapvető előismeretre, amiből lehet általánosítani. A nagy előnyük viszont az, hogy amennyiben a diákoknak sikerül a rendszerszemléletet átadni, sokkal sikeresebben fognak új funkcionalitásokat, addig nem ismert eszközöket felhasználni (ahogyan például a Word feladatokat sikeresen megoldották azok oktatása nélkül), az új anyagot régebbiekkel összekötni, így egy teljesebb képet tudnak alkotni az informatika tudomány egészéről, a konkrét eszköz felhasználhatóságáról, előnyeiről és hátrányairól.

Alapvető előismereteken túl szükség van egy problémára. A probléma vagy egy meglévő rendszerben jelentkező rendszerprobléma, vagy egy elkészítendő komplex megoldás. A komplex alapvetően egymásra ható, egymással kommunikáló, többkomponensű rendszert jelent (ahogy a példákban említettük, ez lehet akár egy Scatch játék is), mely dinamikusan választható, alakítható az aktuális korosztálynak és célnak megfelelően.

Programozók, programtervezők visszajelzéseiből, valamint korábbi kutatásokból tudjuk, hogy a rendszerszintű gondolkodás a legfontosabb kompetenciák közé tartozik, melynek fejlesztése jelenleg nem történik meg kellően hatékonyan. Mivel a kompetencia meglévő tananyagokkal is fejleszthető, pusztán módszertani fogásokkal (vagyis nem feltétlenül szükséges külön modult szánnunk rá), ezért

érdemes a különböző témaköröknél meggondolni, hogy az adott elem alkalmas-e, azon keresztül hatékonyan tudjuk-e fejleszteni a kompetenciát.

Köszönetnyilvánítás

A KULTURÁLIS ÉS INNOVÁCIÓS MINISZTÉRIUM ÚNKP-22-3 KÓDSZÁMÚ ÚJ NEMZETI KIVÁLÓSÁG PROGRAMJÁNAK A NEMZETI KUTATÁSI, FEJLESZTÉSI ÉS INNOVÁCIÓS ALAPBÓL FINANSZÍROZOTT SZAKMAI TÁMOGATÁSÁVAL KÉSZÜLT.

Irodalom

1. L. Zsakó és P. Szlávi: *Informatikai kompetenciák: Algoritmikus gondolkodás* In InfoDidact 2010, Szombathely (2010).
2. R. D. Arnold and J. P. Wade: *A Definition of Systems Thinking: A Systems Approach,* Procedia Computer Science, (2015) 669-678
3. S. Korom: *Architektúrális gondolkodás fejlesztése valós idejű rendszerekkel* In. InfoDidact 2018, Zamárdi (2018)
4. S. Korom., Z. Illés*: Systemic Thinking in Programming Education*. In Recent Innovations in Computing. Lecture Notes in Electrical Engineering, vol 855. Springer, Singapore (2022)
5. B. Castellani, F. Hafferty, C. Schimpf:. *E-Social Science from a Systems Perspective: Applying the SACS Toolkit*. Special Issue: Proceedings from the Urbino-Conference. 7, (2009) 89-106.
6. *Kerettanterv az általános iskola 5–8. évfolyama számára* https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_F.docx (utoljára megtekintve: 2022.11.19)
7. *Kerettanterv a gimnáziumok 9–12. évfolyama számára* https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_K.docx (utoljára megtekintve: 2022.11.19)
8. M. Newman: *A Pilot Systematic Review and Meta-Analysis on the Effectiveness of Problem-Based Learning* (2003)
9. S. Korom and Z. Illés: *Competence Improvements with IoT systems*, Central-European Journal of New Technologies in Research, Education and Practice, vol. 3, no. 1, (2020)
10. *Központi írásbeli feladatsorok, javítási-értékelési útmutatók*, Oktatási hivatal https://www.oktatas.hu/kozneveles/erettsegi/feladatsorok (utoljára megtekintve: 2022.11.19)
11. S. Korom és Z. Illés: *Offline situational game for teaching IT systems*, Central-European Journal of New Technologies in Research, Education and Practice, vol. 4, no. 1, (2022) 25-37
12. T. C. Bell, I. H. Witten, M. Fellows: *Computer Science Unplugged . . . off-line activities and games for all ages* (1998)
13. S. Korom: *IOT Systems in Education*, In Marek, Smid; René, Bílik; Veronika, Stoffová (eds) XXXII. Didmattech 2019, Trnava, Szlovákia : Trnava University in Trnava Faculty of Education, (2019)
14. Leyla Acaroglu: *Tools for Systems Thinkers: Systems Mapping,* Online: https://medium.com/disruptive-design/tools-for-systems-thinkers-systems-mapping-2db5cf30ab3a (utoljára megtekintve: 2022.11.19)

Videók az oktatás szolgálatában

Kovácsné dr. Pusztai Kinga

kinga@inf.elte.hu
ELTE IK

Absztrakt. Az egyre digitálisabb világunkban szükséges, hogy az oktatásunk is átalakuljon. Olyan új pedagógiai módszereket kell keresnünk, amelyek segítségével meg tudjuk szólítani a mai „facebook nemzedéket" is, azaz a Z illetve alfa generációk tagjait. Továbbá érdemes az elmúlt évekből a pandémia miatt bekövetkezett újításaink pozitív eredményeit beépíteni a tanításunkba.

Előadásom elsősorban a digitális történetmesélés módszeréhez kapcsolódik. A módszer rövid jellemzésén túl bemutatom az újításaimat, a diákok véleményét a változtatásokról, illetve a jövőbeni terveimet. A bemutatott példák egy része a közoktatáshoz kapcsolódik, másik része pedig a felsőoktatáshoz kötődik.

Kulcsszavak: videó, digitális történetmesélés, tükrözött osztályterem, informatikatanítás, számítógépes gondolkodás

1. Bevezetés

A rohamosan változó világunkban a néhány éve még bevált pedagógiai eszközök mára már elavulttá váltak, nem tudjuk azokat olyan hatékonyan alkalmazni, mint régen. Helyébe olyan új eszközöket kell keresni, amellyel a mai diákokat, a Z illetve az alfa generáció tagjait is eredményesen meg tudjuk szólítani. Őket a szakirodalom csak „digitális bennszülött" [1] illetve a „Facebook nemzedék" [2] névvel illetik, ők már úgy nőttek fel, hogy gyermekkoruktól elérhető volt az internet. Számukra magától értetődő a személyes kommunikációs eszközök használata, okostelefonnal kelnek és fekszenek, mindig elérhetőek és folyamatosan kapcsolatban vannak egymással az online térben. Könnyen kezelik az információk gyors áramlását, tevékenységeiket gyakran váltogatják multitasking során. Így, a hagyományos, frontális eszközökkel nehéz lekötni a figyelmüket. A vizuális megjelenítést részesítik előnyben, szemben a hosszú, tagolatlan szövegekkel.

Természetesen a generációs elméletekkel szemben számos kritika is napvilágot látott. Egyesek szerint a szociális és más természetes különbségek erősebbek az életkor szerinti besorolásnál. Számos kutató tagadja a digitális bevándorló/bennszülött felosztást. Közülük Kirschner és Bruyckere [3] elsősorban azt az elképzelést támadják, hogy a digitális bennszülöttek képesek a multitasking tevékenységre. Ollé [4] szerint pedig nemcsak a digitális bennszülöttek sajátosságai a fent leírt jegyek.

Összességében elmondható, hogy az egyének közötti különbségek jelentősebbek és sokszínűbbek, mint a generációk közötti különbségek, azonban a diákok tanulási szokásaik változtak. Ez a probléma jól látszik egy korábbi kutatásomban, ahol az egyetemista diákok tanulási szokásait vizsgáltam online kérdőív segítségével. Több féléven keresztül összesen 128 diákot tudtam megszólítani, akiknek a 69%-a prefelálta a tutorial videók megnézését és csak 16%-uk választotta a szöveges jegyzetet, illetve 15%-uk a prezentációt (1. ábra).

1. ábra: Hallgatói vélemény a videó alkalmazásáról. [Saját szerkesztés]

További probléma lehet, hogy az elmúlt néhány év nagy változást hozott az oktatás területén is. A COVID-19 világjárvány megalkotta az oktatás egy új formáját, az online oktatást, ami a pedagógus társadalomra egy óriási terhet tett. Sok digitális tananyag elkészült, mikor visszaálltunk a hagyományos oktatásra. Vajon mi lesz ezeknek a segédanyagoknak a sorsa? Tudjuk-e ezeket a személyes oktatásban is hatékonyan alkalmazni?

Mindezekre a problémákra egy lehetséges megoldást jelenthet számunkra a videók oktatásban történő alkalmazása, mely a digitális történetmesélés, illetve a tükrözött osztályterem módszeréhez is kötődhet. Cikkemben a módszerek rövid ismertetésén túl arra mutatok példát, hogy hogyan lehet a közoktatásban, illetve felsőoktatásban alkalmazni azokat.

2. Digitális történetmesélés bemutatása

A digitális történetmesélés [5] (digital storytelling, továbbiakban DST) **egy olyan új tanulásszervezési eljárás, melyben a hagyományos történetmesélés ötvöződik a digitális eszközhasználattal.** Lényege, hogy a tanulók nem öncélúan alkalmazzák a digitális eszközöket, hanem egyedi elbeszéléseket, sajátos multimédia alkotásokat hoznak létre, melyek felkeltik tanulótársaik figyelmét, lelkesedését és kommunikációt generálnak a feldolgozott témában a tanulóközösségen belül. Többszörösen bizonyított a DST tanulói motivációra [6, 7] és teljesítményre [8] gyakorolt pozitív hatása, fejleszti a tanulók problémamegoldó képességét, az önálló tanulás képességét, illetve a kritikai gondolkodás kialakulását is.

A digitális történetmesélés a diákok körében is nagyon népszerű, mivel olyan tevékenységeket használ, melyeket a diákok a kortárskapcsolataikban amúgy is csinálnak: képeket, videókat, történeteket osztanak meg egymással. A módszer alkalmazása során azonban ezen tevékenységek kiegészülnek egy önálló tanulási, gondolkodási fázissal, illetve egymás munkáinak kritikus, konstruktív értelmezésével.

A DST tanórai felhasználása esetében a módszer lépéseit a következő öt nagyobb szakaszban érdemes definiálni [9].

1. A diákok először megírják a digitális történetük magját adó szöveget. (Ezt megírhatják akár papíron, akár szövegszerkesztővel.) A feldolgozni kívánt témához kapcsolódó, meglévő ismereteiket, élményeiket kiegészíthetik különböző forrásokból válogatott adatokkal. A források felkutatásában, az adatok, információk szelekciójában segítséget nyújthat a facilitátor pedagógus, a kinyert adatok szintetizálása, szöveggé formálása azonban már a tanuló feladata. A szövegalkotási folyamatot végigkíséri a konzultáció lehetősége.
2. A második szakaszban a diákok felolvassák a megírt szövegüket, azaz létrehoznak egy hangfájlt.
3. A harmadik szakaszban a diákok megtervezik és elkészítik a szöveghez szükséges képi elemeket. Ezek lehetnek saját készítésű fotók, illusztrációk, digitalizálhatnak papíron lévő, régi fényképe-

ket, dokumentumokat is. A képanyagban megjelenhetnek interneten talált, szabadon felhasználható felvételek is. A tanulók figyelmét fel kell hívni arra, hogy a képi és szöveges forrásokra hivatkozzanak digitális történetük végén.

4. A negyedik szakasz a vágás, amikor egy tetszőlegesen választott videószerkesztő szoftver segítségével (például: Microsoft Movie Maker, Sony Vegas, illetve az okoseszközökkel is használható online vágóprogram-alkalmazások: WeVideo és Power Director) a tanulók összeállítják digitális történetüket.
5. Az utolsó fázis pedig az elkészült alkotások levetítése, megvitatása és értékelése.

Az egyes szakaszok lehetőséget adnak a tanulóknak kreativitásuk kibontakoztatására, továbbá a kooperatív munkára, mely egyre jobban előtérbe kerül a valós életben.

Az eljárás alkalmazhatósága kézenfekvő minden olyan tantárgyi tartalom tematizálása esetében, melyben létjogosultsága van a személyes elbeszélések megjelenésének [10]. Kérdés azonban, hogy hogyan vonható be a DST a természettudományos tantárgyak módszertanába. Lanszki szerint [11] DST-vel nemcsak egyéni történetek artikulációja valósítható meg, hanem tematikus tartalomfeldolgozás is. Természettudományos tantárgyak esetében feltételezhető, hogy – a diákok életkorából fakadóan – kevés egyéni élethelyzetet feltáró digitális történet születik. Szükségszerű tehát a digitális történetmesélés definícióját tágan értelmeznünk: nemcsak az egyéni élettörténeteket soroljuk a digitális történet osztályába, hanem **a módszer lépéseinek segítségével létrehozott, narrált audiovizuális prezentációkat is**. Így értelmezték a DST-t a Houstoni Egyetem tanárai is, és ezt az értelmezést alkalmaztam én is az óráimon.

3. A módszer megjelenése az oktatásban

3.1. A DST megjelenése külföldön

Számos cikk szól a digitális történetmesélés módszer sikeres alkalmazásáról egyetemi képzésekben elsősorban bölcsészettudományi [12] vagy idegennyelv tanítás [13] területén. A Houstoni Egyetemen egy kurzust [14] is létrehoztak a tanárok számára, ahol a digitális történetmesélés módszer használatát tanítják.

Az informatika tudományokban is történtek sikeres kísérletek a digitális történetmesélés alkalmazásáról annak ellenére, hogy ezen a területen e módszert nehezebb alkalmazni. Amy Csizmar Dalal (Carleton College) például sikeresen alkalmazta a digitális történetmesélést egy programozás alapjai (CS 0) kurzusban [15]. Dalal véleménye szerint a módszer alkalmazása elősegíti a hallgatók informatikai tudományokba való vonzását.

A helsinki egyetemen (University of Helsinki, Faculty of Science, Department of Computer Science) a Computer Science kurzusban pozitív eredménnyel alkalmazták a digitális történetmesélést. A módszer hatására a diákok 82%-a tette le a vizsgát, míg az előző évfolyamon a hagyományos módszernél ez csak a diákok 50%-nak sikerült. Az is elmondható, hogy a diákok motiváltabbak voltak [16].

3.2. A DST megjelenése a közoktatásban

Videó készítése a középiskolában számos módon előfordulhat. Talán legkézenfekvőbb alkalmazási terület, amikor más tantárgyakkal összekapcsoljuk azt. Erre számos példát látunk a szakirodalomban (pl. [10]). Például egy-egy irodalomórára egy költő vagy író életébe képzelve alkothatják meg a diákok a digitális történetüket, amely az általános definícióba is beleillik. Az ilyen digitális történetek megszületésének egyik, talán legnagyobb nehézsége általában a másik szakos tanár, aki nem ért annyira a videó szerkesztéséhez, hogy bevállalna egy ilyen innovatív ötletet. Éppen ezért nagyon fontos, hogy mi legyünk a kezdeményezők, mi nyissunk a másik tárgy tanára felé. Természetesen az

informatikatanárnak általában van egy másik szakja is, ahonnan bőven lehet témát választani, de fontos lenne azokkal a tárgyakkal is foglalkozni, amiket az informatika tanárok nem szoktak másik szaknak választani (pl. irodalom, történelem).

Egy másik kézenfekvő terület lehet, ha az iskola projekthetének célja egy videó elkészítése. Az ötlet és a szituáció hasonló, mint az előző helyzetben.

Azonban ezeknél az ötleteknél sokkal izgalmasabb, hogy az informatika (vagy digitális kultúra) tárgyon belül hogyan tudjuk eredményesen használni a videókat. Itt is több lehetőségünk van. Egy új téma bevezetésénél használhatunk videót motivációs céllal, de egy-egy kiegészítő anyagot is kiadhatunk diákoknak szorgalmi feladatként. Azonban talán legeredményesebben az ismétlésnél lehet alkalmazni. Ehhez kötődik a következő kutatásom is.

Egy rangos középiskola 12. osztályának informatika fakultációján tarthattam órákat a 2019/2020-as tanévben. Az év nagyobbik felében a diákok programozást tanultak, a maradék időben pedig (2. félévben) az emelt szintű érettségire készültek. Mivel az idő egyre rövidebb volt, de az ismétlendő anyag elég nagy, a következő ötletet vetettem be: szorgalmi feladatként a diákok vállalhattak egy témakört, ahol összegyűjtik az érettségihez fontos ismereteket és ezt videóban osztják meg társaikkal. Az elkészült videót ötössel jutalmaztam.

A videókészítésnél a csoportmunkát preferáltam, de lehetett egyéni munkában is dolgozni. A videó elkészítésének tervezett ideje 3 hét volt, minden hétre kitűztem egy elérendő célt. (Például videószerkesztő kiválasztása, feladatok megoldása, videó megtervezése és elkészítése.) A honlapomon több videószerkesztő programot és annak használatát osztottam meg a diákokkal, illetve igény szerint biztosítottam nekik konzultációs lehetőséget. (Arra is volt lehetőségük, hogy egy általam nem bemutatott videószerkesztő programot ajánljanak a társaiknak.) Mivel egy diáknak nagyon nehéz kitalálni, hogy melyek a fontos és kevésbé fontos ismeretek, a feladatot konkretizáltam: egy korábbi emelt szintű érettségi feladatait adtam ki. Kértem, hogy minden feladatot külön videóban készítsenek el. Ezt azért tartottam fontosnak, mert ez által sokkal könnyebben kezelhető a társai számára. Arra gondoltam, hogy a videó feldolgozása úgy történne, hogy a többi diák először megpróbálja megoldani a videó feladatát, majd ellenőrzésként vagy ötletszerzésként megnézi a videó megoldását. (Természetesen a diákok ellenőrzésképpen használhatják az interneten található hivatalos megoldást is, azonban az nem tartalmazza a megoldás menetét, csak a végeredményt.) A videó segítségével így mindenki a saját tempójában haladhatna, még az óra kereteihez sem kötöttek. Továbbá a videó segítségével az ismeret tartóssá válik, nem marad le az anyagról, ha véletlenül egy pillanatra kikapcsol, vagy nem jegyzetel megfelelően.

A kezdeti lelkesedés óriási volt, minden diák akart videót készíteni, így minden témakört sikeresen ki tudtam adni. Megalakultak a csoportok, melyek egy-egy témakört dolgoznak fel. (A csoportok általában 2 főből álltak, egy-két 3 fős csoport is alakult, de sajnos voltak, akik egyénileg akartak dolgozni.) Elkezdődött a munka, azonban már az első héten elkezdődött a csúszás is, több csoport a kitűzött célt nem teljesítette. (Ez persze adódhatott abból, hogy a videókészítés szorgalmi munka volt, azaz nem a tanóra idejében, hanem az otthoni időben kellett elkészíteniük. De a 12. évben már más tanárok is sok otthoni munkát adtak, többen közülük például a nyelvvizsgájukat is ekkor tették le.)

Egy videó készült el a kitűzött időre, és ezután jött a COVID első hulláma, és az oktatás online formába ment át. Az elkészített videó ez által sokkal hasznosabb lett, mint ahogy azt kezdetben elterveztem. A többi videó sajnos nem készült el. (Számítottam, hogy a videók egy része nem fog elkészülni, de azt gondoltam, hogy egynél több kész videó lesz.)

Ennek a problémának a feloldására javaslom, hogy a videó készítése ne a 12. évfolyam anyaga legyen. (Én azért választottam a 12. évfolyamot, mert csak erre volt lehetőségem.) A 11. évfolyamon a diákok a középszintű érettségire készülnek, itt is már csak fakultációban, azaz nem kötelező jelleggel tanulják a diákok az informatikát, illetve ekkor még sokkal inkább belefér az idejükbe a videó

készítés. Egy további ötlet lehet, ha még korábbra tesszük a videók elkészítését, nem ismétlésként alkalmazzuk, hanem összefoglalásként akkor, amikor tanulják az adott témakört. Ennek előnye, hogy korábban még inkább belefér a diákok idejébe, hátránya viszont az, hogy még kevésbé jártasak a videókészítésbe.

A videó értékelése sem úgy történt, ahogyan előre elterveztem. Eredetileg arra gondoltam, hogy az elkészült videókat felhasználjuk, majd az utolsó órán az összes videó megnézése után közösen értékeljük azokat. Ehelyett azonban egyénileg küldték vissza a videó értékelését.

Az elkészült videó sajnos egy nagyon hosszú videó lett, nem követte azt a kérésemet, hogy minden feladatot külön videó tartalmazzon. (Ezt a hibát azonban könnyen tudtuk javítani.) Tartalmi szempontból meg voltam elégedve a videóval, minden fontos információt jól tartalmazott. Formai szempontból azonban elmaradt az egyetemi hallgatói videók színvonalától, egy kicsit monoton volt, kimaradtak azok a látványos elemek (effektek, animációk), amelyek segítenek a diákoknak a figyelmük fenntartásában. Ez persze abból is adódott, hogy a téma túl konkrét volt, kevés kreatív lehetőséget rejtett magában. Összességében a diáktársaik hasznosnak találták a videót.

3.3. A digitális történetmesélés megjelenése a felsőoktatásban

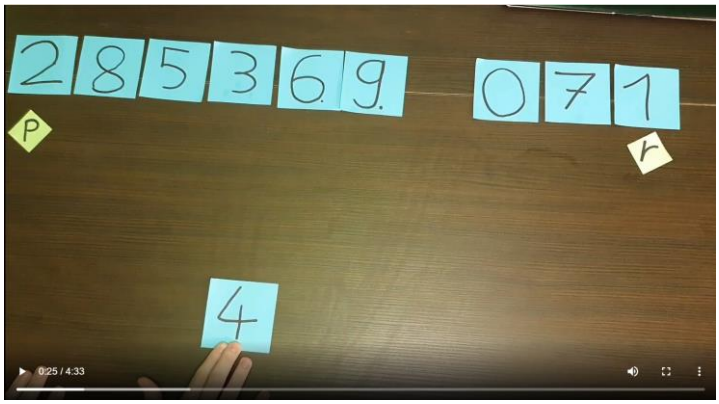
Az egyetemen végzett kutatásaimat 6 féléven keresztül alkalmaztam. Első lépésként a gyakorlati óráimhoz kiegészítő segédanyagokat tettem elérhetővé a hallgatók számára a honlapomon[1], melyek egy-egy algoritmus működését, illetve adatszerkezetet magyaráznak el vagy szemléltetnek. (Ezek általában youtube videók, vagy a www.algoanim.ide.sk honlapról származó animációk, vizualizációk [17] voltak, melyekből néhányat kicseréltem a hallgatói videókra.

Következő lépésként szorgalmi házi feladat keretében buzdítottam hallgatóimat, hogy az általam mutatott videók mintájára csoportmunkában vagy egyénileg készítsék el saját alkotásaikat. Célom az volt, hogy az elkészült videók az algoritmusoknak pontosan azt a variációját szemléltessék, ami az előadáson vagy gyakorlaton elhangzott, azaz a számonkérésnél tökéletesen felhasználható legyen. Ez azért nagyon fontos, mert a világhálón jónéhány változata fut egy-egy algoritmusnak, ezért többször előfordult már, hogy egy hallgató egy feladat megoldásakor azért nem kapta meg a maximális pontszámot, mert nem az előadáson elhangzott változatot szemléltette. (pl. Quicksortnál a pivot elem választása.)

A videókészítéshez segítségül konzultációs lehetőséget biztosítottam a hallgatók számára, melyet a hallgatók kb. egy harmada vett igénybe. A kezdeti lelkesedés minden félévben nagy volt, többen még a konzultációra is eljöttek, de minden félévben a vállalások egy része nem valósult meg. Előfordult olyan is, hogy csoportosan vállaltak el egy témát, de végül csak egy hallgató készítette el az ígért videót. Összesen 9 videó és 5 prezentáció készült el. (Ebből a legkorábbi, ami egyben a legtöbbet megnézett videóból egy képernyőképet láthatunk a 2. ábrán. A kép a videó 25. másodpercében készült, ahol a sorozatunk még rendezetlen, de a pivot elemet már véletlenszerűen kiválasztottuk, ez a 4-es lett.)

[1] 1. félév: https://people.inf.elte.hu/kinga/algoritmusok1/seged.htm

2. félév: https://people.inf.elte.hu/kinga/algoritmusok2/seged.htm



2. ábra: Képernyőkép a hallgatói videóból.

Az egyes félévekben a videókat különbözőképpen pontoztam. Általában egy videóval maximálisan 20 pontot lehetett szerezni. (Azaz ezzel a feladattal a maximálisan kapható szorgalmi pont megszerezhető volt.) Ha a diákok közösen készítettek egy videót, akkor a 20 ponton osztozkodtak. Egyik félévben azonban csak 10 pontot ajánlottam fel egy videó elkészítéséért, de ez kevésnek bizonyult. Ebben a félévben egy videó sem készült, az ok pedig az volt, hogy egy igényesen elkészített videó nagyon sok időt vesz el.

Egy videó elkészítésénél a csoportmunka módszerét preferáltam, hiszen a csoportmunkának számos előnye van. Kezdetben nem is akartam elfogadni egyéni alkotásokat, de a COVID19 járvány miatt a személyes órákat felváltotta az online oktatás, a hallgatók sokkal leterheltebbek lettek, illetve a közös munka is megnehezült. Az első néhány kísérleti félévben csak egyéni videó készült. Később, amikor már az oktatás is visszatért a hagyományos személyes órákhoz, egyre több – általában két főből álló – csoport alakult. A csoportoknál továbbá kértem még egy „projektnapló" vezetést, ami tartalmazza, hogy az egyes részfeladatot ki végezte el. Ebből kiderült, hogy a hallgatók kiválóan megoldották a szervezési feladatokat, általában egyformán vállaltak feladatokat és oldották meg azokat.

A videók minőségével általában meg voltam elégedve. Tartalmi szempontból kértem a hallgatóktól, hogy az algoritmusok órán elhangzott változatához készüljenek rövid (kb. 5 perces) videók, melyeknek a legfontosabb elemük az algoritmus működésének szemléltetése legyen, de örülök, ha tartalmaz algoritmust, illetve műveleti időt is. Bár a videók egy része valóban csak az algoritmusok szemléltetésére koncentrált, kihagyva a másik két opcionális részt, az ajánlott időlimit általában be lett tartva, (a videók átlagos ideje 4,95 perc lett,) illetve hibát csak egyetlen egy videónál találtam, ami aztán javítva is lett. Formai szempontból azonban színvonalas munkák készültek, melyek tartalmaztak olyan látványos elemeket (effekteket, animációkat), amelyek segítenek a hallgatóknak a figyelmük fenntartásában. A videók nagyobbik része valamilyen videószerkesztő programmal készült, a kisebbik része pedig a PowerPoint animált prezentáció videósított változata volt.

Kíváncsi voltam a hallgatók véleményére is, amelyet online kérdőívek segítségével vizsgáltam.

Első kérdésem a videókkal támogatott oktatásról szólt (3. ábra). A „Mennyire találja jó ötletnek, hogy az egyes algoritmusok magyarázatához a honlapomon elérhetővé tegyek tutorial videókat?" kérdésre a hallgatók 3 és 5 között átlagosan 4,77-ra értékelték 0,52-os szórással, illetve a leggyakoribb jegy (82%) az 5-ös volt. A hallgatók véleményezhették azt az ötletet is, hogy plusz pontokért hallgatói videók készüljenek. Ezek közül néhány így hangzik: *„jó ötlet, bevonja a hallgatót az oktatásba, fontosnak érzi, hogy felkészüljön részletesen. A videó elkészítése közben részletesen megérti az algoritmus működését.", „…a videók is segítenék a megértést, talán gyorsítaná is a megértés folyamatát".*

3. ábra: Hallgatói vélemény a tutorial videók használatáról.
[Saját szerkesztés]

Egy másik kérdőívben arra voltam kíváncsi, hogy a hallgatók hogyan értékelik a hallgatótársuk videóit. A kérdőívet 22 hallgató töltötte ki, az összes videót 3 és 5 közötti jeggyel átlagosan 4,55-ra értékelték (4. ábra) és egy hallgató kivételével (93%) mindenkit segített a tanulásban. A legtöbbet megnézett videó témája a Quick Sort volt, amiből láthattunk egy képernyőképet a 2. ábrán.

4. ábra: Hallgatói vélemény a társuk által készített videóról.
[Saját szerkesztés]

A videó elkészítése után a hallgatók véleményére is kíváncsi voltam a videókészítéssel kapcsolatban. Bár online kérdőívet használtam, de a kérdések fele itt kifejtős volt, melyre a következő válaszok születtek:

- „Mi motiválta a videó elkészítésében?"
  - Megértette az anyagot, de néhány társa nem.
  - Segíteni szeretne a társainak a megértésben. *(„Nekem ilyen esetben sokat segítenek a vizuális videó anyagok, amik lépésről lépésre levezetik a folyamatot és úgy gondoltam ez sokat segítene a csoporttársaimnak is.")*
  - Fontos, hogy megfelelő mennyiségű magyar anyag legyen elérhető, de algoritmusok tárgyból magyar nyelven olykor csak igen nehezen érthető szakirodalmak találhatóak.
  - Plusz pontot lehet vele szerezni.
  - Régóta készít videókat és szereti is a videókészítést.
- A videó elkészítésének ideje a tervezett idővel általában összhangban volt, egy hallgató foglalkozott csak kicsivel több időt a tervezettnél. Az okok vizsgálatánál csak két hallgatói véleményt emelek ki, mert ezek kellően tükrözik, hogy miért kellett, illetve nem kellett több idő a tervezettnél. Ezek a következők:

- *„Többször újrakezdtem / átírtam a diákat, mert menet közben jöttem rá, hogy valamit sokkal jobban vagy éppen másként kéne megjeleníteni ahhoz, hogy érthető(bb) legyen."*
- *„Talán az lehet az oka hogy fejben már több százszor elkészítettem, lejátszottam hogy hogyan kéne megvalósítani"*

- Egy hallgatónak volt nehézsége az algoritmus és az animáció összekapcsolásában. A többiek úgy nyilatkoztak, hogy nem voltak nem várt nehézségeik. Egyik hallgató meg is magyarázta az okát, mely így hangzik: *„szabadnak éreztem a megvalósítást ami megkönnyítette a dolgomat"*.
- A videókészítés egy ötös skálán átlagosan 4-re segítette-e a hallgatókat a tananyag elsajátításában. Ennek oka az, hogy olyan hallgatók kezdtek a videók elkészítésébe, akik már értették az anyagot.
- A videókészítésben egy hallgató kivételével mindenkinek voltak már előismeretei.
- A videókészítésről lehetett személyes véleményt is írni, ezek közül néhány érdekes:
  - *„Szerintem roppant hasznos, mert az, aki csinálja, jobban elmerül a témában, az, aki pedig esetleg le van maradva a tárgyból, könnyebben megérti, mint szakirodalomból."*
  - *„Nagyon jó ötletnek tartom, hogy egy ilyen rövid feladattal tudunk plusz pontokat elérni illetve ezzel együtt hozzá tudunk járulni a következő évfolyamok oktatásához is. Egyfajta Win-Win játszma :)".*

4. Kutatási terveim

Ebben a félévben újabb kutatásokat folytatok. Jó lenne, ha a két féléves tantárgy minden anyagához készülhetne jó minőségű videó, de ehhez még sok idő kellene. Viszont az online oktatás hozadékaként a félév minden órája megvan felvételen. A múlt félévek tapasztalatai alapján úgy gondolom, hogy a felvételek megosztásával azt érném el, hogy a hallgatók többet hiányoznának az óráimról, hiszen úgy is meg tudják nézni a felvételeket. Azonban a személyes órai jelenlétet nagyon fontosnak tartom, nemcsak azért, mert a hallgató ezáltal hatékonyabban tanul, hanem azért is, mert az oktató minőségibb órát tud tartani, ha van hallgatósága. Viszont a felvételeket megvágva, egy órából több kis videót létrehozva sokkal hasznosabban tudom felhasználni azokat. Ezen túlmenően a vágások következtében a videók jobb minőségűek lesznek. Ezek a videók nem csak a tananyag elsajátításban segítenek, hanem felhasználhatóak lesznek a diákoknak a saját videójuk elkészítésében is. Azt gondolom, hogy ennek segítségével a videókészítés ideje csökken, a módja esetleg egyszerűsödik, amivel talán több hallgatót tudok megmozgatni, így reményeim szerint több hallgatói videó fog elkészülni egy félév alatt. Tervem, hogy ezeket a felvételeket a hallgatóknak kiadom 10 pont értékben. (Az óráimon a diákok jegyek helyett pontokat gyűjtenek. Amennyiben elérték a zárthelyiken a kötelező minimumot, a két zárthelyi összpontszámát maximum 20 szorgalmi ponttal javíthatják. Szemeszter végén a pontok alapján kapják a gyakorlati jegyüket, 40 ponttól elégséges, majd 20 pontonként kapnak jobb jegyet.) Egy óra megvágása könnyűnek hangzik, de ahhoz, hogy jól elvégezzük a feladatot, többször is meg kell nézni az órai felvételeket, amely az óra anyagának észrevétlen elsajátítását eredményezi.

További kutatási tervem közé tartozik, a módszerek bővítése. A videók alkalmazása nemcsak a DST módszeréhez kötődik, hanem a tükrözött osztályterem (angolul flipped classroom, vagy flipped learning, magyarul "fordított tanulás" néven is ismert) módszeréhez is. Ez a modell a hagyományos oktatás egyfajta megfordítása, ami a hagyományos módon zajlik az osztályteremben, az most otthon történik, ami pedig otthon történik, az pedig az osztályteremben [18]. Olyan tanulásszervezési megoldás, ahol a hallgatók otthon tekinthetik meg az oktató által előre elkészített videót és egyénileg tanulmányozzák a tananyaghoz kapcsolódó ajánlott segédanyagokat, online forrásokat [19].

A módszer alkalmazására kiváló lehetőség lehet az előadás átalakítása. Véleményem szerint a személyes előadásokat felváltaná az online videó megnézése. (Ennek időpontja szabadon választható

lenne.) A kötelező jelenlétet pedig egy kvíz 70%-os kitöltése váltaná fel. Az előadások idejében az oktató konzultációt tart a hallgatóknak, melyen nem kötelező a részvétel.

5. Összefoglalás

Az elmúlt néhány év változásai az oktatásra is hatással vannak. A néhány éve még jól bevált módszerek mára már elavulttá váltak, helyükbe olyan új ötleteket kellene találnunk, amely jól illeszkedik a mai diákok gondolkodási módjához. Továbbá a pandémia miatt bevezetett online oktatásnak előnyei is voltak, amelyeket érdemes lenne megtartani, beleépíteni a tanításunkba. Erre ad egy lehetőséget a videók használata, amely többek között a digitális történetmesélés módszeréhez kötődik. Cikkemben a módszer rövid ismertetésén túl mutattam néhány olyan – közoktatásban vagy felsőoktatásban – kipróbált példát a videók alkalmazására, ahol a tutorial videókat a diákok készítették.

A videók oktatásban betöltött szerepe több szempontból is jelentős. Egyrészt használata közel áll a mai diákok szemléletéhez, így a tananyagba történő beépítése segíti a tanulás folyamatát. Másrészt egy minőségi videó tartós, több éven keresztül használható. Végül, de nem utolsó sorban az önálló tanulást is támogatja, mivel a diákok a videót a számukra legalkalmasabb időben tudják megnézni, többszöri megnézéssel elkerülhető a lemaradás, gyorsított visszanézéssel pedig megoldható az ismétlés, vagy a tudásuk frissítése.

A cikkemben bemutatott hallgatói vélemények is azt bizonyítják, hogy az oktatásnak fontos nyitnia a videók felé.

Irodalomjegyzék

1. M. Prensky, „Digitális bennszülöttek, digitális bevándorlók,” *On the Horizon,* %1. kötet9., %1. szám5., p. 6, október 2001..

2. D. Gelencsér, „Generációk különbségei : X, Y, Z és alfa az iskolában,” Tantrend, 2018. [Online]. http://tantrend.hu/hir/generaciok-kulonbsegei-x-y-z-es-alfa-az-iskolaban. [Hozzáférés dátuma: 19. 10. 2018.].

3. A. P. Kirschner és P. D. Bruyckere, „The myths of the digital native and the multitasker,” *Teaching and Teacher Education 67.,* p. 135–142, 2017.

4. J. Ollé, „A spekulatív jellegű digitális nemzedékelméletek kritikai áttekintése,” 2014. [Online]. https://www.slideshare.net/ollejanos/a-spekulativ-jellegu-digitalis-nemzedekelme-letek-kritikai-attekintese. [Hozzáférés dátuma: 10 10 2021].

5. A. Lanszki és A. Papp-Danka, „Digitális történetmesélés alkalmazása természetu-dományos témájú tantárgyi tartalmak feldolgozásában,” *Neveléstudomány,* %1. kötet2, pp. 26-44., 2017.

6. S. J. Abdolmanafi-Rokni és M. Qarajeh, „Digital Storytelling in EFL classrooms: The effect on the oral performance,” *International Journal of Language and Linguistics. 4. ,* p. 252−257., 2014.

7. C. Y. Ya-Ting és I. W. Wan-Chi, „Digital storytelling for enhancing student academic achievement, critical thinking, and learning motivation: A year-long experimental study,” *Computers & Education. 2.,* p. 339–352., 2012.

8. N. Smeda, E. Dakich és N. Sharda, „The effectiveness of digital storytelling in the classrooms:

a comprehensive study," *Smart Learning Environments,* %1. kötet1, %1. szám6, 2014.

9. H. C. Barrett, „How to Create Simple Digital Stories," 2009. [Online]. http://electronicportfolios.com/digistory/howto.html. [Hozzáférés dátuma: 02. 04. 2021.].

10. A. Lanszki, „Digitális történetmesélés és tanulói tartalom(re)konstrukció," *Új Pedagó-giai Szemle,* %1. kötet66. , %1. szám3/-4., pp. 82-88., 2016.

11. A. Lanszki, „A tanulói aktivitás szerepe a digitális történetmesélésben," D. Lévai és A. PappDanka, szerk., Budapest, Eötvös Kiadó, 2015, pp. 79-92..

12. C. Macalester, „Digital Story Samples In the Liberal Arts," 2021. [Online]. https://dwlibrary.macalester.edu/digitalstorytelling/overview/samples/. [Hozzáférés dátuma: 06. 10. 2021].

13. N. Parsazadeh, P.-Y. Cheng, T.-T. Wu és Y.-M. Huang, „Integrating Computational Thinking Concept Into Digital Storytelling to Improve Learners' Motivation and Performance," *Journal of Educational Computing Research,* %1. kötet59, %1. szám3, pp. 470-495., 2021.

14. R. R. Bernard és S. G. McNeil, „Powerful Tools for Teaching and Learning: Digital Storytelling," Coursera, 2021. [Online]. https://www.coursera.org/learn/digital-storytelling. [Hozzáférés dátuma: 06. 10. 2021].

15. A. Csizmar Dalal, „Digital Storytelling as a Gateway to Computer Science," *Journal of the Research Center for Educational Technology (RCET),* %1. kötet4, %1. szám2, pp. 124-137., 2009.

16. A. Korhonen és M. Vivitsou, „Digital Storytelling and Group Work: Integrating the Narrative Approach into a Higher Education Computer Science Course," *Proceedings of ACM ITiCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'19),* pp. 140-146, 02. 07. 2019.

17. L. Vegh és J. Udvaros, „Possibilities of Creating Interactive 2D Animations for Education Using HTML5 Canvas JavaScript Libraries," *eLearning and Software for Education,* %1. kötet2, pp. 269-274, 2020.

18. J. Bergmann és A. Sams, Flip Your Classroom: Reach Every Student in Every Class Every Day, United States of America: International Society for Technology in Education, ISTE/ASCD, 2012, p. 124.

19. D. Lévai, „Tükrözött osztályterem," Tempus Közalapítvány, 2019.

Alapvető számolási készségek tesztelése papíron és táblázatkezelőben

László Vilmos Csongor[1], Nagy Keve[2], Csernoch Mária[3]

[1]lvcs@dioszegi-refi.hu, {[2]nagy.keve, [3]csernoch.maria}@inf.unideb.hu
[1] Diószegi Kis István Református Két Tanítási Nyelvű Általános Iskola és Alapfokú Művészeti Iskola, Berettyóújfalu, [2,3]Debreceni Egyetem Informatikai Kar

Absztrakt. Az általános iskola 2–7. osztályos tanulóinak alapvető számolási készségét vizsgáltuk hagyományos papíralapú és táblázatkezelői környezetben. A papíralapú tesztelés feltételeit először 1971-ben Dr. Nagy József dolgozta ki és publikálta a „Az elemi számolási készségek mérése" című könyvében. A dolgozat egy rendkívül aktuális témát dolgoz fel, mivel széles körben elterjedt az az állítás, hogy nincs szükség a digitális bennszülöttek számítógépes kompetenciájának fejlesztésére. A mindennapi gyakorlat azonban cáfolja ezeket az állításokat. Jelen tanulmányban azt vizsgáltuk, hogy egy hagyományos műveleti sebességet mérő felmérés eredményét hogyan befolyásolja egy általános célú szoftver használata. A kutatás során öt hipotézis vizsgálatára került sor. A tanulmány kitér arra is, hogy hogyan fejleszthető a tanulók digitális kompetenciája, számítógépes gondolkodása annak érdekében, hogy ezen képességek, készségek hiánya ne akadályozza a valódi problémamegoldást.

Kulcsszavak: számolási készségek, hagyományos tesztelés, táblázatkezelői tesztelés

1. Bevezetés

A 21. század által kínált lehetőségek (a számítógépek mindenki számára elérhető volta, a digitalizáció, valamint az IKT környezetbe való mindennapi aktív cselekvésünk) arra ösztönözte a kutatást végző csoportot, hogy a korábbi, Nagy József és munkatársai [1] által kidolgozott alapvető számolási készségeket mérő módszert továbbgondolja és a mai modern technológiák alkalmazásával, valamint új hipotézisek felállításával újravizsgálja.

1.1. Az iskola és környezetének bemutatása

Az alapvető számolási készségek tesztelését az Észak-Alföldi régióban elhelyezkedő Berettyóújfalui Kistérség központjában, Berettyóújfaluban lévő Diószegi Kis István Református Két Tanítási Nyelvű Általános Iskola és Alapfokú Művészeti Iskolában végeztük el. Az intézmény a Tiszántúli Református Egyházkerület fenntartásában működik 18 tanulócsoporttal.

A kutatás helyszínéül szolgáló általános iskola a mérés időpontjában 420 tanuló oktatását látta el 1–8. évfolyamon. A mérésben részt vevő tanulók száma a 2–7. évfolyamon 302 tanuló volt. Az általános iskolába 18 településről járnak tanulók. A környező településekről bejáró „vidéki" tanulók aránya a vizsgált időszakban 38%. Az intézményben 54 fő pedagógus dolgozik, hat szakmai munkaközösséget alkotva.

A kutatást érintően két szakmai munkaközösséget emelnék ki. A Reál Munkaközösség, amelyhez a matematika, fizika és a kémia tantárgy tartozik, és amelynek matematikát tanító tagjai végezték el a papíralapú mérést, valamint a Digitális és Innovációs Munkaközösséget, amely a digitális mérés lebonyolításában, az adatok rögzítésében nyújtott nagy segítséget.

1.2. A korábbi tesztelések, mérések bemutatása

A több mint négy évtizeddel ezelőtt elvégzett országos méréssorozat [1] már akkor felhívta a figyelmet arra, hogy a köznevelés területén szükségesek olyan mérések, amelyek a megfelelő elemzése és az azokból levont következtetések nagyban hozzájárulhatnak az oktatás adott területein a minőség és az eredmények javulásához. Ennek a törekvésnek napjainkban az országos kompetenciamérés feleltethető meg leginkább.

Jelen kutatás tárgya igyekszik leszűkíteni a vizsgált területet a legegyszerűbb, általában mindenkitől elvárhatóan ismert négy alapművelet területén végzett vizsgálatra. A diagnosztikus mérés-értékelés eszközeinek készítésekor a kiindulópont a vizsgálandó terület tartalmi és strukturális elemzése volt.

A Nagy József [1] által kidolgozott mérés elemzése során az elemi számolási készségekkel kapcsolatban megállapítható, hogy az összeadás, szorzás, kivonás és bennfoglalás műveleteit a tanulóknak a begyakoroltság maximális szintjén kell ismerni. Nagy [1] elemi számolási készségekkel kapcsolatos vizsgálata azt mutatja, hogy olyan módszert (módszereket) kell alkalmazni, amely lehetővé teszi a maximális begyakoroltság, illetve ennek színvonalának mérését. Ennek megfelelően, a mérőlapok összeállításánál figyelembe vették a mérendő műveletek fajtáit, a tervezett mérés maximális időkeretét. Végül a mérőlapok tartalmát művelettípusonként 75 db műveletben és a mérés összidőtartamát tekintve 10 percben határozták meg úgy, hogy művelettípusonként 1 perc alatt a tanuló hány feladatot tud helyesen megoldani. A teljesítmény mérése során a teljesítmény tempóját és a teljesítmény minőségét vették figyelembe. Vizsgálták annak kérdését, hogy bizonyos elemi műveletek könnyebbek, mások pedig nehezebbek lehetnek a tanulók számára (pl.: 2+2, 8+7).

A négy évtizeddel ezelőtt indított országos mérés teszi lehetővé azt, hogy az adott közösségekben végzett mérések eredményei értelmezhetőek legyenek. Ezek viszonya mutatja meg, hogy az adott közösségben végzett mérések alapján a tanuló eredménye hol helyezkedik el az országos átlaghoz viszonyítva, illetve a maximálisan elérhető teljesítményhez képest milyen színvonalon áll.

A mért eredmények osztályzattá való átalakításának kérdése abban a tekintetben merült fel, hogy a tanulók, illetve azok szülei számára egy, a magyar oktatási rendszeren belül már mindenki számára ismert, jól értelmezhető módon kerüljön meghatározásra a tanuló teljesítménye. Így összevethetővé válik a tanítási órákon szerzett érdemjegyek esetleges szubjektív megállapítása, egy az országos mérésen alapuló tényleges teljesítménnyel.

2. Elméleti háttér

2.1. Digitális bennszülöttek

2001-ben Prensky [2] állítása alapján megérkeztek a digitális bennszülöttek, és ők azok a gyerekek, akiknek nincs szükségük formális oktatásra a digitális tanulmányok, az informatika területén. 2006-ban Wing [3] megfogalmazta, hogy a 3R (Reading, wRiting, and aRithmetic) mellett a számítógépes gondolkodásnak kellene a negyedik alapvető képességnek lennie mindenki számára. E két meghatározást figyelembe véve arra következtethetünk, hogy a gyerekek digitális eszközökkel és készségekkel születnek; következésképpen Wing csak összefoglalta a bizonyítékokat. Az éremnek azonban két oldala van. A Prensky által a gyermekekhez állítólagosan társított készségeket és képességeket vizsgálva megállapítható, hogy a gyermekek digitális bennszülötteknek születnek, de a további állításokat bizonyítani kell, ami még nem történt meg. 2017-ben Kirschner és De Bruyckere [4] szilárd bizonyítékot szolgáltattak arra, hogy egy korszakba születni csak az eszközökhöz való hozzáférést teszi lehetővé, de a készségekhez, képességekhez nem. A készségeket, képességeket nem születéskor kapjuk; ezeket megtanuljuk, elsajátítjuk képzésben, iskolákban, szakemberek, jól képzett tanárok által nyújtott segítséggel [5] [6] [7]. A naiv tudás tévhitekhez is vezethet [8], valamint a kevéssé hatékony problémamegoldó megközelítések alkalmazásához és elterjedéséhez. [9]–[14].

2.2. Mathability

A mathability [15] definíció szerint a kognitív infokommunikáció (CogInfoCom) egyik ága [16] [17], amely kereteken belül a létező rendszerek, technológiák és a problémamegoldás kapcsolatát vizsgálják. Ezen eredmények alapján két kategóriát határoztunk meg: a magas és alacsony mathability rendszerek [15].

Azokat a problémamegoldó megközelítéseket tekintjük alacsony matematikájúnak, ahol a problémák megoldására egy rendszer által biztosított meglévő funkciókat és módszereket – úgynevezett eszközöket – alkalmazunk. Itt a hangsúly az eszközökön van. Egy másik lehetőség, amikor a rendszer meglévő eszközei alapján új programokat és funkciókat (algoritmusokat) fejlesztenek ki újszerű problémák megoldására. Ebben az esetben a hangsúly a problémán van, és az eszközökre csak az algoritmusok végrehajtásához van szükség [18]–[20]. A mathability fogalma teljes összhangban van Wolfram megközelítésével, ahol az innovációról és az evidenciáról (létező és elérhető eszközök) van szó [21] [22]. A magas mathability az innovációvezérelt eszközhasználatot jelenti, míg az alacsony mathability az eszközvezérelt innovációt.

2.3. Alapkészségek és képességek

Régóta vizsgált terület, hogy az alapvető készségek hogyan befolyásolják a problémamegoldást az iskolai gyakorlatban. A vizsgálatok tárgya a tanulók kognitív fejlődése [5] [23] [24] [25] és az, hogy milyen szerepet játszik a gyors és a lassú gondolkodás a valós problémamegoldásban [25]–[27], továbbá, hogy a valós tartalmak hogyan növelhetik a diákok motivációját és eredményességét [28] [29]. Széles körben elfogadott, hogy az alapvető készségek kulcsfontosságúak a további fejlődéshez bármely tudományban, így a számolási készségek kifejezetten a matematika tanulmányok során, és implicit módon bármely tantárgyhoz kapcsolódó tanulási folyamatban.

Az 1971-ben indított vizsgálatsorozatot elsősorban Nagy végezte és adminisztrálta [1], majd később a követői folytatták ezt a munkát [30]–[35]. E vizsgálatok elsődleges célja az volt, hogy mérjék a tanulók 3R alapkészségeit az egymást követő tanévekben, és visszajelzést adjanak kognitív fejlődésükről maguknak a tanulóknak, iskoláiknak és a tanulók szüleinek.

2.4. Informatikai alapismeretek

Más tudományokhoz hasonlóan egyértelmű, hogy az algoritmikus gondolkodás fejlesztéséhez alapvető informatikai ismeretekre van szükség. Ez magába kell, hogy foglalja mind a hardver, mind a szoftver eszközöket is. Amíg a diákok nem képesek az alapvető perifériás eszközök és a „felhasználóbarát" irodai szoftvereszközök hatékony kezelésére, addig nem tudnak a valódi problémákra összpontosítani, amelyeket a számítógépekkel kell megoldani.

Jelen tanulmány azt vizsgálja, hogy a tanulók hogyan kezelik a táblázatkezelő környezetre átdolgozott, az alapvető számolási készségeket mérő klasszikus papíralapú tesztet. Prensky megállapításai alapján [2] feltételezhetjük, hogy nincs különbség a papír- és a táblázatkezelős tesztelési módszer eredményei között. Kirschner és De Bruyckere megállapításai azonban [4] arra utalnak, hogy még a digitális bennszülötteknek is szükségük van megfelelő oktatásra az alapvető készségek fejlesztéséhez.

Továbbá azt is meg akartuk vizsgálni, hogy a papíralapú tesztelés helyettesíthető-e táblázatkezelésen alapuló teszteléssel anélkül, hogy a tanulók számolási készségeinek eredményeire hatással lenne az alkalmazott eszköz.

2.5. Hipotézisek

A következő hipotéziseket a digitális bennszülöttekhez rendelt készségek, képességek és a kutatócsoportunk által bevezetett tesztek módosítása alapján fogalmaztuk meg.

H1 Nincs különbség a papíralapú és a táblázatos tesztek eredményei között.

H2 A papíralapú tesztelés helyettesíthető táblázatalapú teszteléssel anélkül, hogy az eredménye-

ket befolyásolná az adatfelvétel módja.

H3 A táblázatos tesztelés értékelése gyorsabb és megbízhatóbb, mint a papíralapú.

H4 Van különbség az összeadással és a kivonással megadott pótlás feladatok eredményei között.

H5 Van különbség a válaszok száma és a válaszok száma között az első szakadási pontig (kihagyott feladat).

3. Módszertan

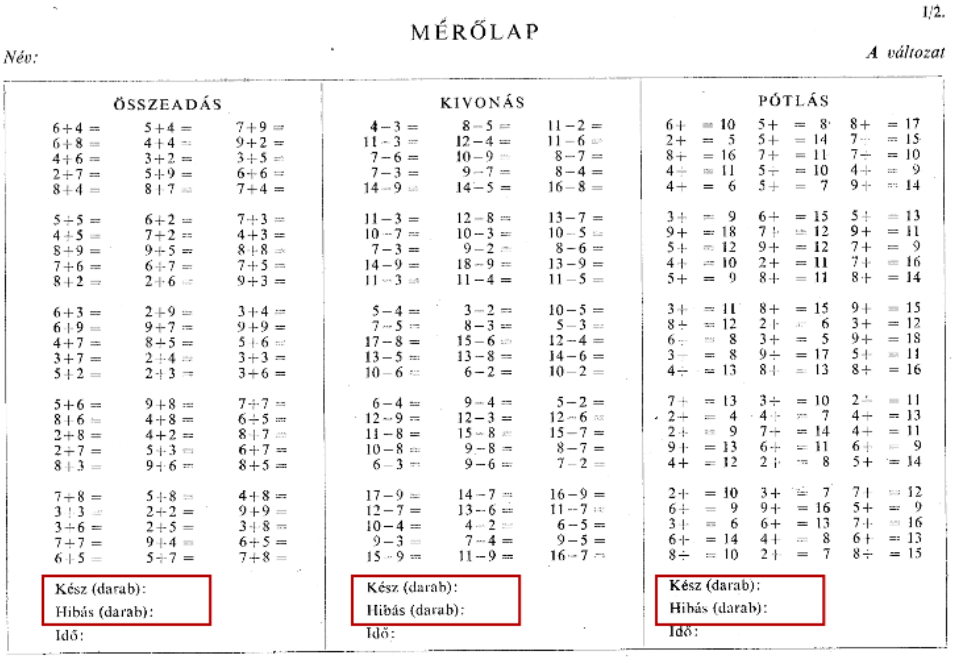
3.1. Tesztelések

3.1.1. Eredeti papíralapú tesztelés

Nagy eredeti vizsgálati módszere szerint [1] és követőinek kisebb módosításával – elsősorban [31] és más iskolák [36] [37] –, kutatócsoportunk az alábbi megkötéseknek megfelelően állított össze egy tesztsorozatot.

Nagy [1] öt aritmetikai művelet tesztelését javasolta: összeadás, kivonás, pótlás összeadással, szorzás és osztás. Mindkét operandus (összeadás és a szorzás esetében) vagy az egyik operandus és az eredmény (a kivonás, a pótlás és az osztás esetében) a [2, 9] intervallumban van. Nagy [1] gondosan megválasztotta az operandusokat, lehetővé téve a lassabb és a gyorsabb tanulók számára is a sikerélményt a tesztek elején.

Minden egyes művelet 75 feladatból áll, amelyek egy adott felépítésben vannak megszervezve. A 75 feladat három oszlopba van rendezve, következésképpen minden oszlopban 25 feladat található. A jobb olvashatóság érdekében öt soros csoportokra vannak osztva, 3 × 5 × 5 csoportot alkotva (1. ábra és 2. ábra).



MÉRŐLAP

1/2.

Név:

A változat

ÖSSZEADÁS

6+4 =	5+4 =	7+9 =
6+8 =	4+4 =	9+2 =
4+6 =	3+2 =	3+5 =
2+7 =	5+9 =	6+6 =
8+4 =	8+7 =	7+4 =
5+5 =	6+2 =	7+3 =
4+5 =	7+2 =	4+3 =
8+9 =	9+5 =	8+8 =
7+6 =	6+7 =	7+5 =
8+2 =	2+6 =	9+3 =
6+3 =	2+9 =	3+4 =
6+9 =	9+7 =	9+9 =
4+7 =	8+5 =	5+6 =
3+7 =	2+4 =	3+3 =
5+2 =	2+3 =	3+6 =
5+6 =	9+8 =	7+7 =
8+6 =	4+8 =	6+5 =
2+8 =	4+2 =	8+7 =
2+7 =	5+3 =	6+7 =
8+3 =	9+6 =	8+5 =
7+8 =	5+8 =	4+8 =
3+3 =	2+2 =	9+9 =
3+6 =	2+5 =	3+8 =
7+7 =	9+4 =	6+5 =
6+5 =	5+7 =	7+8 =

Kész (darab):
Hibás (darab):
Idő:

KIVONÁS

4−3 =	8−5 =	11−2 =
11−3 =	12−4 =	11−6 =
7−6 =	10−9 =	8−7 =
7−3 =	9−7 =	8−4 =
14−9 =	14−5 =	16−8 =
11−3 =	12−8 =	13−7 =
10−7 =	10−3 =	10−5 =
7−3 =	9−2 =	8−6 =
14−9 =	18−9 =	13−9 =
11−3 =	11−4 =	11−5 =
5−4 =	3−2 =	10−5 =
7−5 =	8−3 =	5−3 =
17−8 =	15−6 =	12−4 =
13−5 =	13−8 =	14−6 =
10−6 =	6−2 =	10−2 =
6−4 =	9−4 =	5−2 =
12−9 =	12−3 =	12−6 =
11−8 =	15−8 =	15−7 =
10−8 =	9−8 =	8−7 =
6−3 =	9−6 =	7−2 =
17−9 =	14−7 =	16−9 =
12−7 =	13−6 =	11−7 =
10−4 =	4−2 =	6−5 =
9−3 =	7−4 =	9−5 =
15−9 =	11−9 =	16−7 =

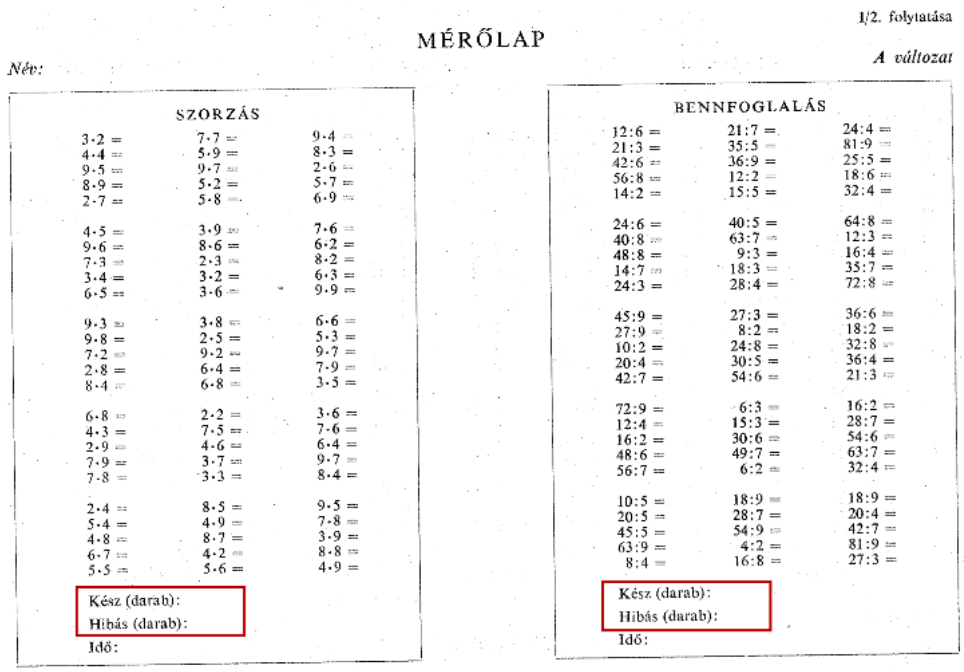
Kész (darab):
Hibás (darab):
Idő:

PÓTLÁS

6+ = 10	5+ = 8	8+ = 17
2+ = 5	5+ = 14	7+ = 15
8+ = 16	7+ = 11	7+ = 10
4+ = 11	5+ = 10	4+ = 9
4+ = 6	5+ = 7	9+ = 14
3+ = 9	6+ = 15	5+ = 13
9+ = 18	7+ = 12	9+ = 11
5+ = 12	9+ = 12	7+ = 9
4+ = 10	2+ = 11	7+ = 16
5+ = 9	8+ = 11	8+ = 14
3+ = 11	8+ = 15	9+ = 15
8+ = 12	2+ = 6	3+ = 12
6+ = 8	3+ = 5	9+ = 18
3+ = 8	9+ = 17	5+ = 11
4+ = 13	8+ = 13	8+ = 16
7+ = 13	3+ = 10	2+ = 11
2+ = 4	4+ = 7	4+ = 13
2+ = 9	7+ = 14	4+ = 11
9+ = 13	6+ = 11	6+ = 9
4+ = 12	2+ = 8	5+ = 14
2+ = 10	3+ = 7	7+ = 12
6+ = 9	9+ = 16	5+ = 9
3+ = 6	6+ = 13	7+ = 16
6+ = 14	4+ = 8	6+ = 13
8+ = 10	2+ = 7	8+ = 15

Kész (darab):
Hibás (darab):
Idő:

1. ábra: Nagy eredeti tesztlapjainak három művelete 0: összeadás, kivonás, kiegészítés összeadással.


MÉRŐLAP

1/2. folytatása

A változat

Név:

SZORZÁS

3·2 =	7·7 =	9·4 =
4·4 =	5·9 =	8·3 =
9·5 =	9·7 =	2·6 =
8·9 =	5·2 =	5·7 =
2·7 =	5·8 =	6·9 =
4·5 =	3·9 =	7·6 =
9·6 =	8·6 =	6·2 =
7·3 =	2·3 =	8·2 =
3·4 =	3·2 =	6·3 =
6·5 =	3·6 =	9·9 =
9·3 =	3·8 =	6·6 =
9·8 =	2·5 =	5·3 =
7·2 =	9·2 =	9·7 =
2·8 =	6·4 =	7·9 =
8·4 =	6·8 =	3·5 =
6·8 =	2·2 =	3·6 =
4·3 =	7·5 =	7·6 =
2·9 =	4·6 =	6·4 =
7·9 =	3·7 =	9·7 =
7·8 =	3·3 =	8·4 =
2·4 =	8·5 =	9·5 =
5·4 =	4·9 =	7·8 =
4·8 =	8·7 =	3·9 =
6·7 =	4·2 =	8·8 =
5·5 =	5·6 =	4·9 =

Kész (darab):
Hibás (darab):
Idő:

BENNFOGLALÁS

12:6 =	21:7 =	24:4 =
21:3 =	35:5 =	81:9 =
42:6 =	36:9 =	25:5 =
56:8 =	12:2 =	18:6 =
14:2 =	15:5 =	32:4 =
24:6 =	40:5 =	64:8 =
40:8 =	63:7 =	12:3 =
48:8 =	9:3 =	16:4 =
14:7 =	18:3 =	35:7 =
24:3 =	28:4 =	72:8 =
45:9 =	27:3 =	36:6 =
27:9 =	8:2 =	18:2 =
10:2 =	24:8 =	32:8 =
20:4 =	30:5 =	36:4 =
42:7 =	54:6 =	21:3 =
72:9 =	6:3 =	16:2 =
12:4 =	15:3 =	28:7 =
16:2 =	30:6 =	54:6 =
48:6 =	49:7 =	63:7 =
56:7 =	6:2 =	32:4 =
10:5 =	18:9 =	18:9 =
20:5 =	28:7 =	20:4 =
45:5 =	54:9 =	42:7 =
63:9 =	4:2 =	81:9 =
8:4 =	16:8 =	27:3 =

Kész (darab):
Hibás (darab):
Idő:

2. ábra: Nagy eredeti tesztlapjainak két művelete 0: szorzás és osztás.

Az eredeti feladatleírásnak megfelelően [1] a tanulóknak a függőleges feldolgozást kellett követniük: a bal felső sarokban kezdték, lefelé haladtak, majd a második oszlopban folytatták, végül a harmadik oszlopban dolgoztak.

A módosított változatban egy vízszintes feldolgozási sorrendet kell követni (3. ábra). A változtatás oka nem nyomon követhető, kutatócsoportunk nem talált megbízható forrást. A forgalomban lévő, nem publikált módosított változatok a „folk pedagogy" áldozatai, amelyet Lister világosan definiált [38]. A tesztelés és a kiértékelés menetében bekövetkezett, nem publikált módosítások a követelményeket és az értékelési folyamatot tisztázatlanná teszik.

Addition		
3+6=	2+3=	5+2=
3+3=	2+4=	3+7=
5+6=	8+5=	4+7=
9+9=	9+7=	6+9=
3+4=	2+9=	6+3=
…	…	…

3. ábra: Kindrusz módosított utasítása, amelyet az iskoláknak osztottak ki.

Az eredeti és a módosított vizsgálati utasítások között egy további változásra derült fény. A tesztekhez rendelt idő megváltozott. A módosított változatban minden egyes műveletre 60 másodperc áll rendelkezésre, és két művelet között 15–20 másodperces szünet engedélyezett.

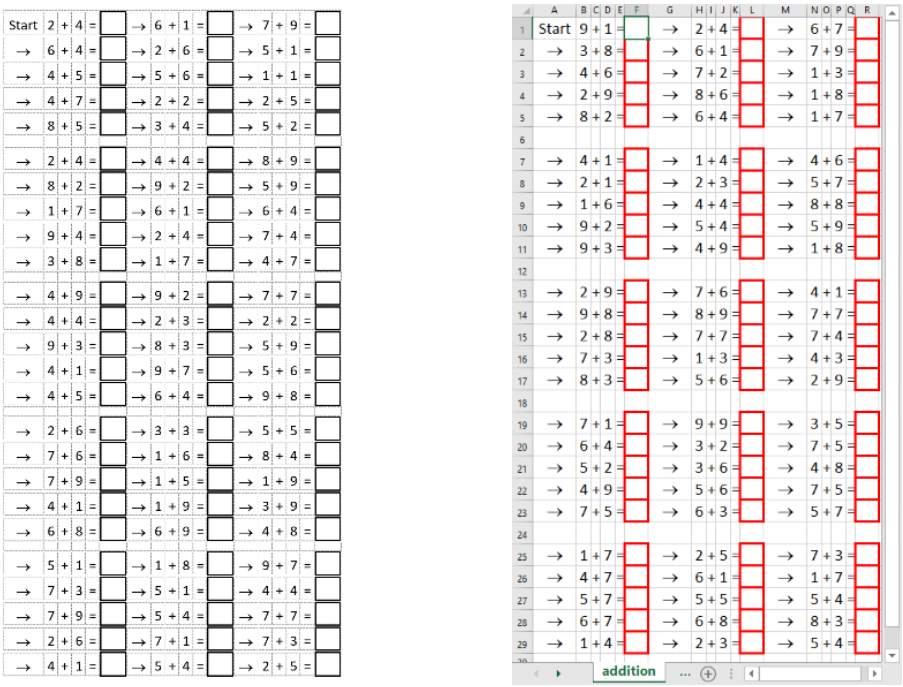
3.1.2. A papíralapú és a táblázatos tesztelés előkészítése

A tesztelési körülményeket figyelembe véve elsődleges célunk az volt, hogy a lehető legjobban kövessük a korábban elvégzett papíralapú teszteket. Ugyanakkor változtatásokat alkalmaztunk, mivel azt akartuk, hogy

- a lehető legnagyobb mértékben automatizáljuk mind az előkészítési, mind az értékelési folyamatokat,
- vízszintes nyilak segítsék a tanulókat a feldolgozási sorrend követésében,
- jelöljük ki vastag szegéllyel a kitöltendő cellákat,
- a pótlás műveletet nemcsak összeadással, hanem kivonással is tesztelni tudjuk,
- rögzíti tudjuk az összes rendelkezésre álló adatot, és
- a két tesztelési környezet a lehető legjobban hasonlítsuk egymáshoz.

E feltételek szerint mind a papíralapú, mind a táblázatos tesztek azonos feltételek mellett készültek (4. ábra). A számokat véletlenszerűen választottuk ki az [1, 9] intervallumban. Ez a megoldás lehetővé tette, hogy könnyebb és a nehezebb feladatokat is megjelenjenek a teszten belül bárhol, így a legelején is. Minden művelethez két operanduskészletet készítettünk: egyet a papír-, egyet pedig a táblázatkezelős teszteléshez. Ezzel a módszerrel minimalizáltuk a két teszt redundanciáját és interferenciáját.

A véletlenszám-generálás után a hat műveleti táblázatot egy MS Word körlevél dokumentumba konvertáltuk, amelyhez a diákok hivatalos adatbázisa szolgált címzettlistaként. Végül az egyesített dokumentumokat kinyomtattuk, a fejlécben a művelettel, a diákok nevével és osztályával. Az egyesítés kimenete egy hatoldalas, kétoldalasan kinyomtatott dokumentum volt, tehát diákonként három darab papírlap. A tesztelést osztályonként végeztük, Nagy [1] és Kindrusz [31] adminisztrációs utasításai szerint.



4. ábra: A módosított tesztlapok papíron és táblázatkezelőben (pótlás).

A táblázatkezelésen alapuló tesztelés során egy hat munkalapból álló munkafüzetet hoztunk létre. A kitöltendő cellákat a papírhoz hasonlóan szegéllyel és kiegészítő színnel emeltük ki. Továbbá csak azok a cellák voltak írhatók, amelyeket ki kellett tölteni, míg a többit a cellák formázásával és a lapok védelmével védtük. Következésképpen az összeadás munkalapon az első kitölthető cella az F1. A cellák között a legegyszerűbben és kényelmesebben a jobbra mutató nyíl segítségével navigálhatunk.

A mintatáblázat elkészítése után az iskola rendszergazdája elkészítette a személyre szabott táblázatkezelő munkafüzeteket, és feltöltötte azokat a tesztelésre előkészített számítógépekre. Ebben az esetben a név és az osztály szolgált azonosítóként.

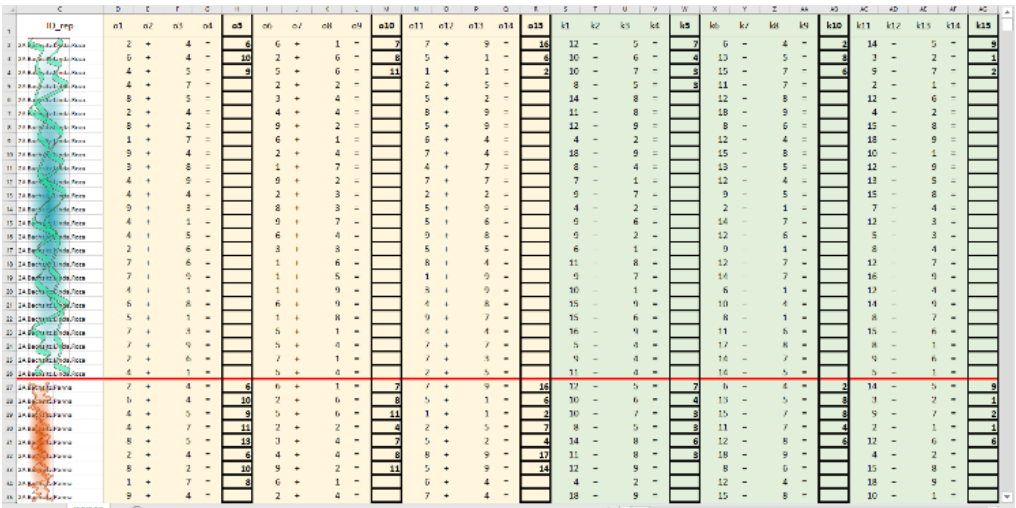
3.1.3. A vizsgálati eredmények rögzítése és feldolgozása

Mind a papíralapú, mind a táblázatos tesztek eredményeit értékelő táblázatokban rögzítettük. Az értékelő táblázatok mindkét teszt esetében a tanulók által kitöltött táblázatok alapján készültek. A papíralapú eredményeket az iskola pedagógiai asszisztensei gépelték, míg az Excel-alapú teszteket programokkal és scriptekkel alakították át.

A papíralapú tesztek esetében az eredeti hat lapot egy lapra alakították át, ahol a műveletek egymás mellett, egymás utáni sorrendben vannak. Ez az ábrázolás balról jobbra haladva követi a papíralapú és a táblázatos tesztelés során elvégzett műveletek sorrendjét. A gépelési terhek könnyítése érdekében ez a forma megtartja az eredeti 3 × 5 × 5 elrendezést. Az értékelő táblázat összesen 90 + 3 adatmezőből (tanulói azonosító, megjegyzés, rekord azonosító, 90 eredmény mező), 7550 rekordból és 1 mezőnevek sorból áll. Az adatrögzítő asszisztensek munkájának további segítésére egy speciálisan szerkesztett és formázott táblázatot hoztunk létre (5. ábra),

- a táblázatot egy színkóddal egészítettük ki, amely elkülöníti a különböző műveleteket (sárga: o1–o15 mezők az összeadáshoz, zöld: k1–k15 mezők a kivonáshoz stb.),
- a tanulókat piros vonallal választottuk el, és
- a tanulók azonosítóját (név és osztály) megismételtük a megfelelő rekordokban (ID_rep mező, D2:D26 az első diák esetében).

Az értékelő táblázat ilyen kialakításával minden diákhoz 25 rekordot rendeltünk. Az adatrögzítést követően, az adatok kétszeres ellenőrzését az iskolai pedagógiai asszisztensek végezték.



5. ábra: A papír alapú eredmények rögzítő lapja: összeadás (sárga), kivonás (zöld).

A tanulók adatainak rögzítését követően a 25 soros blokkokat rekordokká alakították át, minden tanulóhoz egy rekordot hoztunk létre. Ez az új táblázat 450 adatmezőből és az azonosítók oszlopából, valamint 302 rekordból áll, a tesztben részt vevő diákok számának megfelelően.

A táblázatalapú teszteket az iskolai hálózatról töltötte le a rendszergazda, a scripteket és a segédprogramokat pedig a kutatócsoport egyik tagja tervezte és kódolta. Az adatrögzítés szempontjából legfontosabb script az, amely egy táblázattá alakította a különálló táblázatkezelői munkafüzeteket, a munkafüzeteken belül a lapokat, a lapokon belül pedig a $3 \times 5 \times 5$ elrendezési struktúrát. A végleges táblázat szerkezete megegyezik a papíralapú elrendezéssel, $1 + 450$ mezővel és $1 + 302$ rekorddal.

4. Kiértékelés

4.1. Tesztek értékelése

A számolási készségek mérésének értékelési folyamatát tekintve erős vita folyik. Nagy [1] és Kindrusz [31] szerint a tanulók értékelhetik a tesztlapokat. Nyilvánvaló azonban, hogy a tanulók értékelése nem feltétlenül megbízható. Továbbá, csak a próbálkozások számát és a rossz válaszok számát rögzítették (Piros téglalapok a 1. ábra és 2. ábra). Abban azonban nincs egyetértés, hogy a próbálkozások számát hogyan kell kiszámítani. A vita forrása a teszt utasításában keresendő, ahol a végrehajtási sorrend szigorúan meg van adva. Az iskolák egy része az utasítás alapján számolta ki a tanulók eredményeit, vagyis azokat az eredményeket figyelmen kívül hagyták, amelyek az első üres cella után következnek. Ezzel szemben Nagy [1] és Kindrusz [31] szerint minden választ számolni kell, függetlenül a végrehajtási sorrendtől, nem adva értelmet a szigorú végrehajtási sorrendnek. A végrehajtási sorrend figyelmen kívül hagyásával a tanulóknak lehetőségük van a leginkább tetsző feladatok kiválasztására, ami az eredményeket összehasonlíthatatlanná teszi. Egy iskolán belül mindkét módszer elfogadható lehet. Ez az anomália azonban megbízhatatlanná teszi az iskolák összehasonlítását.

Jelen vizsgálatban a tanulók összes rendelkezésre álló adatát rögzítettük, így különböző értékelési folyamatok és statisztikai elemzések végezhetők. A rögzített adatok alapján mind a papíralapú, mind a táblázatkezelős tesztelés esetében meg tudjuk adni a tanulók eredményeit, amelyek a következőkből állnak:

- válaszok,
- válaszok az első üres cellákig (szakadási pont),
- helyes válaszok,
- helyes válaszok az első üres cellákig,
- helytelen válaszok,
- helytelen válaszok az első üres cellákig.

A rögzített eredmények alapján további statisztikai elemzések végezhetők. Mivel a tanulmány elsődleges célja a papíralapú és a táblázatkezelésen alapuló vizsgálati módszerek összehasonlítása, a statisztikai elemzések a két módszer közötti hasonlóságokra és különbségekre összpontosítottak.

4.2. Statisztikai elemzés

A statisztikai elemzést mind iskolai szinten, mind a teszteléshez beállított paraméterekkel összhangban végeztük el. A korábban közzétett eredményeknek megfelelően elsősorban leíró elemzést végeztünk, ahol az átlagokat és a szórásokat számoltuk ki és hasonlítottuk össze.

A további elemzésekhez az eredmények eloszlását a Kolmogorov-Smirnov-teszttel vizsgálták. Megállapítottuk, hogy az eredmények nem normális eloszlásúak, figyelembe véve mind a papíralapú, mind a táblázatos tesztelést és az összes választ az első szakadásig adott válaszokkal szemben.

Ezek alapján további elemzéseket végeztünk annak megállapítására, hogy hol találtunk különbségeket a számolási készségekben. Az elemzéseket a Mann-Whitney-teszt segítségével végeztük el.

5. Minta

A tanulók alapvető számolási készségeit vizsgáló tesztelésre egy olyan iskolában került sor, amely a régió hivatalos tesztsorozatában való részvétel nélkül, december első két hetében végezte el azt. Mind a tanárok, mind a tanulók jól ismerték a papíralapú tesztelési stílust, és nagy tapasztalatokkal rendelkeztek mind a kitöltés, mind az értékelés folyamatában. Következésképpen, a táblázatkezelői eredményekkel való összehasonlíthatóság miatti kisebb változások nem okoztak problémát. Az adatok rögzítését azonban, mint már említettük, szigorúan az iskolai asszisztensek végezték.

	Évfolyam						
	2	3	4	5	6	7	Összesen
papír	65	53	30	42	41	71	302
Excel	65	53	30	42	41	71	302

1. táblázat: A tesztelésben részt vevő diákok száma.

Az iskola összes 2–7. osztályos tanulóját teszteltük papíron és táblázatkezelőben (MS Excel) (1. táblázat) egyaránt. A 2. osztályban négy műveletet, négy lapot – összeadás, kivonás, pótlás összeadással, pótlás kivonással – kellett kitölteni. Az összes többi évfolyamon hat lapot kellett kitölteni, az említett négyet, valamint a szorzást és az osztást. Fontos továbbá megjegyezni, hogy az iskola minden tanulója 1. osztálytól tanul informatikát, heti 1 órában. A Nemzeti alaptanterv szerint [39], a 7. osztályos tanulók tanulnak táblázatkezelést, a többi évfolyam egyikének sincs táblázatkezelési tapasztalata. Az összes többi évfolyam azonban oktatóprogramokat használ, ahol a beviteli eszközök használatát lehet gyakorolni.

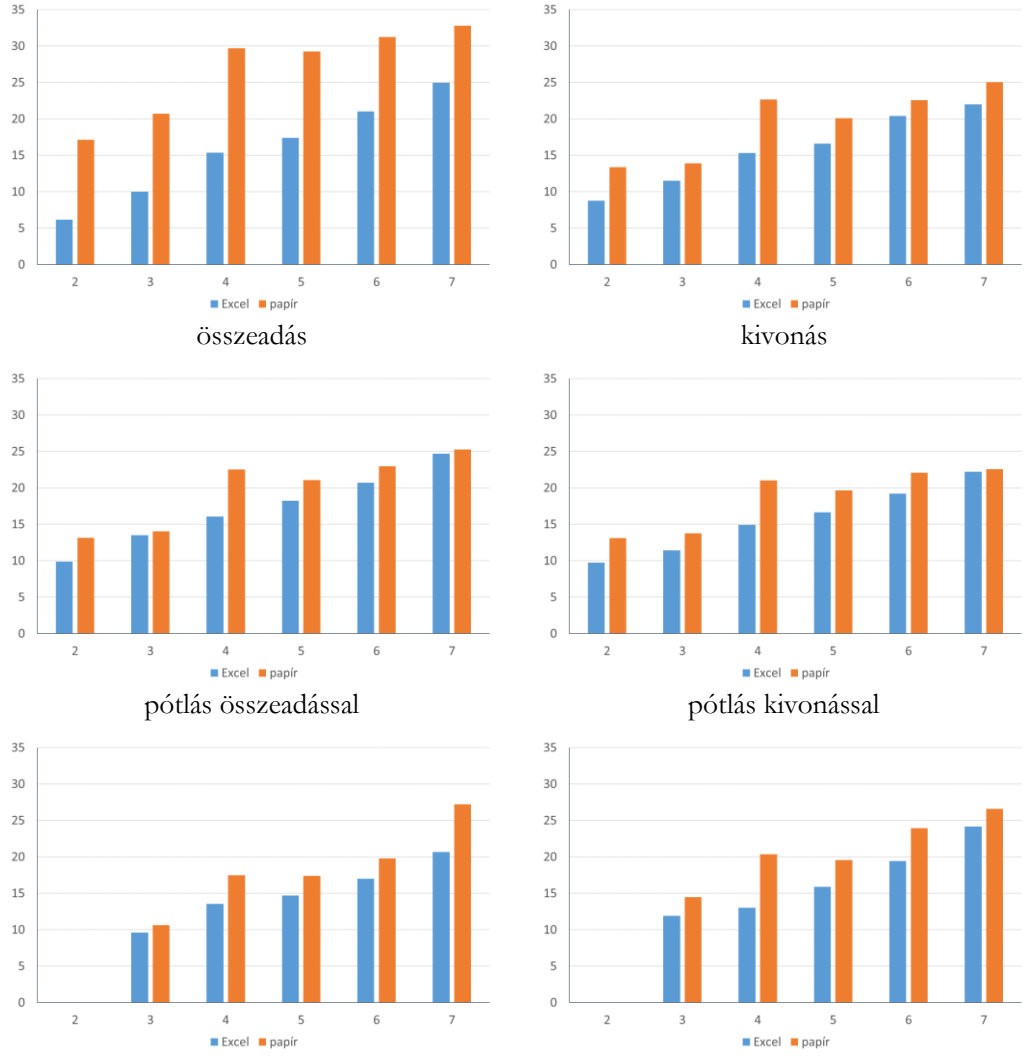
A papíralapú tesztelésre a matematikaórákon került sor, míg a táblázatos tesztelésre az iskola számítógéptermében, az informatikaórákon. A számítógépteremben munkaállomások vannak, melyek beviteli eszközként 105 billentyűs magyar billentyűzettel és egérrel vannak felszerelve.

6. Eredmények

6.1. Papír- és táblázatalapú mérések összehasonlítása

A vizsgálat elsődleges célja a tanulók eredményeinek összehasonlítása a papíralapú és a táblázatkezelős tesztekben. Arra kerestük a választ, hogy az eszközválasztásnak hatása van-e a tanulói teljesítményekre.

A grafikonok egyértelműen mutatják (6. ábra), hogy a papíralapú eredmények magasabbak, mint a táblázatkezelői dokumentum eredményei. A további következtetések levonása érdekében ezeket az eredményeket összehasonlítottuk és tovább elemeztük. Az is figyelemre méltó, hogy a 4. osztályosok az összeadásban, a kivonásban és a pótlási műveletekben ugyanolyan magasan teljesítenek, mint a felsőbb osztályok. Annak ellenére, hogy a 4. osztályosok viszonylag jobban teljesítenek matematikából, a digitális készségek terén nem tűnnek jobbnak. Ebben a tekintetben a 3. osztály tűnik sikeresebbnek a többieknél.



6. ábra: A gépelt válaszok átlagos száma a papíralapú (narancs) és az táblázatalapú (kék) tesztekben.

Figyelembe véve a követendő feldolgozási sorrendet és a válaszok értékelését Nagy [1] és később Kindrusz [31] szerint, kiszámítottuk mind a válaszok számát, mind az első szakadásig terjedő válaszok számát, azaz minden választ az első üres celláig értékeltünk.

Az iskolai szinten mind a papíralapú, mind a táblázatkezelős válaszokat összegyűjtve csak egy művelet (pótlás kivonással) esetében találtak szignifikáns különbséget ($p=0,023$), míg a többi feladatban a különbség nem szignifikáns (összeadás, kivonás, pótlás összeadással, szorzás, osztás: $p=0,427$; 0,340; 0,394; 0,282; 0,278). A statisztikai elemzés azt mutatta, hogy sem osztály-, sem eszközszinten – papír vagy táblázatkezelő – nincs szignifikáns különbség egyik műveletnél sem a szakadási pontokat és az összteljesítményt vizsgálva.

A helyes válaszokat tekintve a minta hasonló. Iskolai szinten a pótlás kivonással esetében van különbség ($p=0,038$), míg a többi művelet esetében nincs szignifikáns különbség. Az értékelést

osztályokra és eszközökre lebontva nem találtunk szignifikáns különbségeket. A H5 hipotézis részben elvetésre került: a tanulók általában követik az utasítás sorrendjét.

6.2. A műveletek összehasonlítása

A statisztikai elemzés következő szakaszában összehasonlítottuk a papír- és a táblázatalapú eredményeket. Megvizsgáltuk továbbá, hogy van-e szignifikáns eltérés a két mérési eredmény között.

	p MW	átlag excel	átlag papír	std excel	std papír
összeadás	0,000	14,89	25,92	9,307	0,000
kivonás	0,000	14,93	18,82	7,875	0,000
pótlás összeadással	0,001	16,66	19,04	8,205	0,001
pótlás kivonással	0,000	14,73	17,05	7,913	0,000
szorzás	0,146	14,80	14,31	6,858	0,146
osztás	0,108	17,19	15,73	8,171	0,108

2. táblázat: A diákok válaszainak átlaga és szórása az első szakadásig.

Iskolai szinten szignifikáns különbségek mutatkoztak a papíralapú és a táblázatkezelős tesztek válaszainak száma között az összeadás, a kivonás és a pótlás műveletek esetében, míg a szorzás és az osztás esetében nincs különbség (2. táblázat). Általánosságban elmondható, hogy különbség van a két tesztelés között, így a H1 hipotézist elvetjük.

Hasonló eredmények születtek akkor is, amikor a helyes válaszokat az első szakadásig és az utolsó válaszig számoltuk (a szakadási pontokat figyelmen kívül hagytuk).

	összeadás	kivonás	pótlás összeadással	pótlás kivonással	szorzás	osztás
2	0,000	0,000	0,000	0,002		
3	0,000	0,006	0,213	0,007	0,058	0,040
4	0,000	0,002	0,001	0,023	0,114	0,007
5	0,000	0,020	0,024	0,038	0,118	0,021
6	0,000	0,363	0,190	0,042	0,244	0,019
7	0,000	0,057	0,819	0,992	0,000	0,354

3. táblázat: Szignifikancia vizsgálata a Mann–Whitney próbával évfolyamok és műveletek szerint.

A tesztelést végző tanárokkal készített interjúk szerint a tanulóknak gondot okozott a válaszok begépelése és a feldolgozási sorrend követése. Nem tudták, hogyan kell kitölteni egy táblázat celláját, és hogyan kell navigálni a táblázatban az irányt jelző nyilak és a védett vs. nem védett cellák ellenére. Ahogy azonban a diákok haladtak előre a műveletekben, előbb-utóbb rájöttek (vagy egyedül, vagy osztálytársaik segítségével, amit a tanárok nem tudtak megakadályozni), hogyan kell hatékonyan

használni a billentyűzetet. Ez megmagyarázza, hogy a papíralapú és a táblázatkezelős tesztelés közötti különbség csökken, mire a tanulók eljutnak a szorzás és osztás műveleteihez.

Az osztályokra lebontott eredmények egyértelműen azt mutatják, hogy a 3. osztály kivételével az alsós évfolyamok tanulói nehezen kezelik a táblázatkezelői felületet és a billentyűzetet, annak ellenére, hogy az informatikát már az 1. osztályban elkezdték tanulni. A 6. és 7. osztályban még az összeadás is meglepte őket, de a többi műveletnél valamennyire felzárkóztak (3. táblázat). Ezek az eredmények azt mutatják, hogy az alapvető digitális készségek szilárd ismerete nélkül a papíralapú tesztelés nem váltható fel a táblázatkezeléssel; következésképpen a H2 hipotézist elvetjük.

6.3. A pótlási feladatok összehasonlítása

Nagy [1] eredeti tesztje szerint és a közösségben keringő kalózverziók szerint soha nem vizsgálták, hogy van-e különbség a két pótlási művelet között vagy sem. Statisztikai elemzésünk azt mutatta, hogy iskolai szinten szignifikáns különbség van a két feladat eredményei között ($p=0{,}000$), tehát a H4 hipotézist elfogadjuk. Ez az eredmény arra utal, hogy nem vonhatunk le következtetéseket kizárólag az összeadási művelettel történő pótlásra támaszkodva.

6.4. Időtényezők

A felkészülési és értékelési folyamatot figyelembe véve az összehasonlítás azt mutatta, hogy a táblázatalapú tesztelés megbízhatóbb és kevésbé időigényes, mint a hagyományos, papíralapú tesztelés. A korrektség kedvéért meg kell említeni, hogy a legelső teszt kiértékelése némi többletidőt igényelt a táblázatokban rögzített eredmények kiértékelését segítő programok/szkriptek összeállításához és kódolásához. A tesztek megismétlésekor erre a többletidőre már nincs szükség. A papíralapú eredmények rögzítése és ellenőrzése két pedagógiai asszisztens esetében körülbelül két hetet vett igénybe. Ezeket az eredményeket figyelembe véve a H3 hipotézis elfogadható.

Összegzés

A 2–7. osztályos tanulók alapvető problémamegoldási képességét/stratégiáit vizsgáltuk hagyományos papíralapú és táblázatkezelői környezetben. A papíralapú tesztelés feltételeit 1971-ben Nagy [1] fogalmazta meg. A tesztelés feltételei – szándékos kisebb és nem szándékos, szájhagyomány útján terjedő változtatásoktól eltekintve – elég stabilak voltak a táblázatkezelői változat elkészítéséhez.

A két teszt összehasonlítása azt mutatja, hogy a diákok nem rendelkeznek a billentyűzet és a táblázatkezelői táblázat kezeléséhez szükséges alapvető készségekkel és képességekkel, annak ellenére, hogy digitális bennszülöttek, és az első osztálytól kezdve informatikát tanulnak. Ugyanakkor az is kiderült, hogy ezen a rövid teszten belül a diákok eredményei a műveletek mentén haladva javultak. Ezek az eredmények azt jelzik, hogy jelenleg a papíralapú tesztelés nem helyettesíthető számítógépes alapúval, mivel a diákok valós tudását akadályozhatja a digitális készségek, képességek hiánya. Továbbá az is egyértelmű, hogy az alapvető digitális készségek fejlesztésére összpontosítva ezek a digitális anyanyelvű gyerekek gyorsan felzárkóznak.

Az is egyértelmű a teszteredmények összehasonlítása alapján, hogy az alapvető digitális készségek fejlesztése döntő szerepet játszik a gyors és lassú gondolkodás aktiválásában. A digitális eszközök kezelésének olyan magas szintet kell elérnie, mint a hagyományos eszközök (pl. toll, ceruza, nyomtatott könyvek stb.) kezelésének, hogy a gyors gondolkodás aktiválódjon, és teret engedjen más gyors gondolkodási műveleteknek (pl. alapvető számtani műveletek) vagy a lassú gondolkodásnak a valódi életből vett problémamegoldáshoz. Amíg a digitális eszközök használata akadályozza a magasabb szintű gondolkodást és problémamegoldást, addig a számítógépek és a digitális eszközök nem integrálhatók más iskolai tantárgyakba és tudományokba hatékonyan.

Irodalom

1. J. Nagy, "Az elemi számolási készségek mérése," Tankönyvkiadó, Budapest, 1971.
2. M. Prensky "Digital Natives, Digital Immigrants," From on the Horizon (MCB University Press, Vol. 9 No. 5, October 2001), Utolsó megtekintés: 2018. 03. 21. http://www.marcprensky.com/writing/Prensky%20-%20Digital%20Natives,%20Digital%20Immigrants%20-%20ParPRET.pdf.
3. J. Wing, "Computational Thinking," in Communications of the ACM, vol. 49, no. 3, pp. 33-35, 2006. doi: 10.1145/1118178.1118215.
4. P. Kirschner, and P. De Bruyckere, "The myths of the digital native and the multi-tasker," in Teaching and Teacher Education. 67 2017, pp. 135-142.
5. J. Hattie, "Visible Learning for Teachers: Maximizing Impact on Learning," Routledge, 2012.
6. M. Csernoch, "Thinking Fast and Slow in Computer Problem Solving", in Journal of Software Engineering and Applications, 2017, vol. 10, no. 1.
7. W. Ng "Can we teach digital natives digital literacy?," Computers & Education, 2012/11/01/;59(3):1065-1078. 2012 https://www.sciencedirect.com/science/article/abs/pii/S0360131512001005?via%3Dihub, Hozzáférés: Január 21, 2022.
8. B. Champagne, R. F. Gunstone és L. E Klopfer, "Naive knowledge and science learning," in Research in Science and Technological Education, vol. 1, no. 2, 1983, pp. 173-183.
9. R. R. Panko: What We Know About Spreadsheet Errors, Journal of End User Computing. Special issue on Scaling Up End User Development, 2008, (10) 2, pp. 15-21.
10. R. R. Panko & S. Aurigemma: Revising the Panko-Halverson taxonomy of spreadsheet errors, Decision Support Systems, 2010, (49) 2, pp.235-244.
11. R. R. Panko: The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad: Proceedings of the 46th Hawaii International Conference on System Sciences, Maui, Hawaii, 2013, pp. 1-10.
12. M. Ben-Ari, "Bricolage Forever!", in PPIG 1999. 11th Annual Workshop. 5–7 January 1999. Computer-Based Learning Unit, University of Leeds, UK. Utolsó megtekintés: 2018. 03. 21. http://www.ppig.org/papers/11th-benari.pdf.
13. M. Ben-Ari és T. Yeshno, "Conceptual models of software artifacts," in Interacting with Computers, vol. 18, no. 6, pp. 1336-1350, 2006. doi: 10.1016/j.intcom.2006.03.005.
14. M. Csernoch, P. Biró, J. Máth, & K. Abari, "Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments," in Informatics in Education: an international journal vol. 14, no. 2, 2015, pp. 175-197.
15. P. Baranyi & A. Gilányi, "Mathability: International Conference on Cognitive Infocommunications (CogInfoCom), 2013, pp. 555-558, doi: 10.1109/CogInfoCom.2013.6719309.
16. P. Baranyi, A. Csapo A., & G. Sallai: Cognitive Infocommunications (CogInfoCom), Springer International Publishing Switzerland, 2015. p.191. (978-3-319- 19607-7 http://www.springer.com/us/book/9783319196077#aboutBook
17. P. Baranyi & A. Csapo: Definition and Synergies of Cognitive Infocommunication, Acta Polytechnica Hungarica, 2012. Vol 9, of No 1. pp. 67-83. (ISSN 1785-8860)
18. K. Chmielewska és A. Gilanyi, "A matematizálhatóság oktatási kontextusa", Acta Polytechnica Hungarica, vol. 15, no. 5, pp. 223-237, 2018.
19. K. Chmielwska és A. Gilányi, " Educational context of mathability," in 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, 2019.
20. P. Biró & M. Csernoch, "The mathability of computer problem solving approaches," in 2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2015, pp. 111-114. doi: 10.1109/CogInfoCom.2015.7390574.

21. C. Wolfram, "Evidence: Let's promote not stifle innovation in education," 2015. Utolsó megtekintés: 2018. 03. 21. http://www.conradwolfram.com/home/2015/5/21/role-of-evidence-in-education-innovation.
22. C. Wolfram, "The Math(s) Fix: Wolfram Media, Inc., 2020.
23. ACER, "Trends in International Mathematics and Science Study," Utolsó megtekintés: 2018. 03. 21. https://research.acer.edu.au/timss/.
24. OECD, " Programme for International Student Assessment," Utolsó megtekintés: 2018. 03. 21. https://www.oecd.org/pisa/
25. J. Sweller, P. Ayres és S. Kalyuga, "Cognitive Load Theory," Berlin: Springer, 2011.
26. D. Kahneman, "Thinking, Fast and Slow", New York: Farrar, Straus; Giroux, 2011.
27. M. Roter, "Managing People for Improvement", Adaptiveness and Superior Results. McGraw-Hill Education, 2009.
28. P. Mishra és M. J. Koehler, "Technological Pedagogical Content Knowledge: A framework for teacher knowledge," in Teachers College Record, vol. 108, no. 6, 2006, pp. 1017-1054.
29. C. Angeli és N. Valanides, "Technological Pedagogical Content Knowledge: Exploring, Developing, and Assessing TPCK,", 2015. Springer, 2015.
30. Gy. Ágoston, J. Nagy és S. Orosz, "Mérési módszerek a pedagógiában", magyarul: "Mérési módszerek a pedagógiában", Tankönyvkiadó, Budapest, 1974.
31. P. Kindrusz, "Mérési módszerek a tanulók egyéni és differenciált fejlesztésében. Research Study," in Hungarian: "Mérési módszerek a tanulók egyéni és differenciált fejlesztésében Kutatási zárótanulmány," Országos Közoktatási Szakértői Konferencia, Hajdúszoboszló, 2018.
32. T. Vidákovics, "Diagnosztikus pedagógiai mérés", magyarul: "Diagnosztikus pedagógiai értékelés", Akadémiai Kiadó, Budapest, 1990.
33. B. Csapó, "A korábban megszerzett tudás mérése és elfogadása", magyarul: "Az előzetesen megszerzett tudás mérése és elismerése", Nemzeti Felnőttképzési Intézet, Budapest, 2005.
34. B Csapó, "Diagnosztikus mérés és egyéni, differenciált fejlesztési folyamat", magyarul: Diagnosztikus mérések és egyéni differenciált fejlesztések," Országos Közoktatási Szakértői Konferencia, Hajdúszoboszló, 2016.
35. Á. Ranschburg, "Az iskolák értékelési-mérési gyakorlata és a kompetenciák," Új Pedagógiai Szemle, 2004. március, Elérhető online, Hozzáférés: Január 21, 2022. https://ofi.oh.gov.hu/tudastar/iskolak-ertekelesi oldalról.
36. MezőMatek, "Műveleti sebesség,". Utolsó megtekintés: 2018. 03. 21. https://sites.google.com/a/mezosuli.hu/mezomatek/alapmuveleti-sebesseg oldalról.
37. érdCenter, "Matek a kicsiknek" Utolsó megtekintés: 2018. 03. 21. http://www.erdcenter.hu/pub/ec/jatek/matekkicsi.html oldalról.
38. R. Lister, "After the gold rush: Toward sustainable scholarship in computing," in ACE '08: Proceedings of the tenth conference on Australasian computing education. 78. kötet, 2008, pp. 3-17.
39. OFI, "Nemzeti alaptanterv 2012", 110/2012. (VI. 4.) Korm. rendelet a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról, Utolsó megtekintés: 2018. 03. 21. http://ofi.hu/sites/default/files/attachments/mk_nat_20121.pdf

Doodling és algoritmika

Osztián Pálma Rozália[1], Kátai Zoltán[2], Osztián Erika[3]

{[1]osztian.palma, [2]katai_zoltan, [3]osztian}@ms.sapientia.ro
Debreceni Egyetem, Sapientia EMTE

Absztrakt. A jegyzetkészítés vagy a piszkozat használatának fontossága számos tudományterületen megmutatkozik. Legyen szó matematikáról, irodalomról, történelemről vagy bármi más szakterületről a jegyzetek használata és elkészítése sokszor elengedhetetlen folyamata a tanulásnak. A tanulás mellett, az úgynevezett „doodling" (doodle) jelenség jelentősen hozzájárul a diákok problémamegoldó képességének fejlesztéséhez, és a feladatok megértéséhez is.

A különféle szakterületek közül az informatika és a számítógépes gondolkodás sem képeznek kivételt. Előzetes kutatások során bebizonyosodott, hogy a doodling nem csupán a problémamegoldó képességekre összpontosít, hanem kiemelt hangsúlyt helyez, sőt részese a feladat megértésének és elemzésének, főként kezdők esetén.

Dolgozatunk célja, hogy vizsgáljuk a doodling számítógépes gondolkodásra gyakorolt hatását három különböző algoritmus vizualizáció esetén. Továbbá, kíváncsiak vagyunk arra is, hogy miként befolyásolja a megértést a résztvevők előzetes programozási tapasztalata.

Kulcsszavak: doodling, problémamegoldó képesség, algoritmus, vázlat, nyomon követés, piszkozat

1. Bevezető

1.1. Algoritmika

Különféle algoritmikai feladatokkal akár már egészen kiskorban találkozhatunk. Minden kihívást jelentő matematikai-, fizikai-, kémiai–, informatikai- és hasonló szakterületekhez tartozó kérdésben fellelhetőek algoritmikai vonatkozások. De talán nem is kell szakterületekről, iskoláról és oktatásról beszélnünk, hiszen a mindennapi élet során felmerülő események és teendők is sokszor azonosíthatók különféle algoritmusokkal. Ezek közül a legismertebbek közé tartoznak a keresési és rendezési algoritmusok, melyekkel mindenki találkozhat élete során, akár tudatában van, akár nem. A rendező és kereső algoritmusok szemléltetéséhez és megtanításához, az adott korosztály függvényében, különféle módszereket alkalmaznak a pedagógusok.

1.2. Vizualizációk

A Sapientia Erdélyi Magyar Tudományegyetemen működő AlgoRythmics kutatócsoportunk egyik legjellegzetesebb kutatási témája a különféle algoritmusvizualizációkkal való oktatás [1], [2], [3], [4]. A módszer különlegessége abban rejlik, hogy a különféle kereső-, és rendező algoritmusok emberi mozgás által, a tánc nyelvén kerülnek szemléltetésre. Mostanáig, összesen 10 tánckoreográfiával szemléltetett algoritmus látott napvilágot, melyek három különböző táncstílus révén kerültek bemutatásra (néptánc, flamenco, ballet). A videók repertoárját idén az előzőekhez hasonló, mégis új stílust képviselő vizualizációval sikerült bővíteni. A tánc helyét ezúttal a leegyszerűsített emberi mozgás, pontosabban szólva a színészi játék vette át. Mindez lehetőséget adott arra, hogy további kutatásokat végezzünk az emberi mozgás által szemléltetett vizualizációk hatékonyságának felmérése kapcsán.

1.3. Doodling

A vizualizációk mellett még számos tényező befolyásolhatja az algoritmusok megértését. Több kutatás is amellett érvel, hogy a különféle programozási feladatok kihívást jelentenek a diákok számára [5,6]. Sokszor az informatikai alapfeladatok ismerete és megértése, akár egy év programozás tanítás után sem bizonyul elegendőnek, főként kezdők esetén [7,8]. Ennek egyik oka a diákok problémamegoldó képességének hiánya lehet és az, hogy nem osztják részfeladataira az adott problémát [9]. A probléma részfeladatokra való elosztása és megértése nagymértékben elősegíthető a különféle papír alapú levezetési stratégiák által [5-7,10]. A „doodling" egyik típusa a vázlatkészítési stratégiáknak, mely jelentősen elősegíti egy algoritmus lépéseinek levezetését, valamint egy adott kérdéshez tartozó helyes válasz meghatározását.

Jelen tanulmány három központi témája közé az algoritmusok, a vizualizáció típusok és a firkálás („doodling") tartozik. Az algoritmusok tekintetében két rendező stratégián (beszúró-, shell rendezés) mértük a hallgatók teljesítményét, melyet három különböző vizualizációval (animáció, tánc és színészi játék) szemléltettünk. Továbbá, a mérés során a résztvevőknek lehetőségük volt piszkozatlapot használni, melyre lejegyezhették a feladathoz kapcsolódó megoldásmenetet. Kíváncsiak voltunk, hogy a hallgatók hogyan teljesítenek a két algoritmus esetén, és arra is, hogy az eredményeket miként befolyásolja az előzetes programozási tapasztalat, a vizualizációk típusa, valamint a piszkozat használata.

2. Szakirodalmi áttekintő

2.1. A vázlatkészítéstől a doodle-ig

Az algoritmusok megértésének elősegítése érdekében a programozási feladatok megoldását a hallgatók gyakran különféle reprezentációkkal egészítik ki, amelyek úgynevezett annotációként szolgálnak és igen hasznosnak bizonyulnak [10]. Ezeknek a technikáknak három nagy csoportját különböztethetjük meg:

- **sketching** (vázlat: a programozó által meghatározott programállapotról vagy bármely más számítási folyamatról készített írásbeli vizualizáció leírása [7]);
- **tracing** (nyomon követés: egy algoritmus végrehajtási folyamatának a teljes vagy részleges emulálása, utánzása [11]) és;
- **doodling** (firkálás: diagramok és megjegyzések, amelyeket általában tapasztalt programozók írnak vagy rajzolnak, amikor egy algoritmus működésének meghatározásával szembesülnek [5]).

A feladatok megértése és megoldása szoros összefüggésben van a diákok problémamegoldó képességével. Ennek a képességnek az öt alapvető lépését a McCracken kutatócsoport [9] is meghatározta: (1) a probléma elvonatkoztatása a leírásától, (2) al-problémák generálása, (3) az al-problémák al-megoldásokká való átalakítása, (4) újra összeállítás vagy rekompozíció és (5) értékelés és iterálás. Ugyancsak az ITiCSE 2004-es „McCracken kutatócsoport" munkájához fűződik a doodling kérdésköréhez tartozó egyik legjelentősebb kutatás, akik szerint a rutinnak számító programozási feladatok végrehajtásának gyenge ismerete hatással van a hallgatók problémamegoldó képességének és programozási ismeretének hiányosságaira. Ilyen rutinnak nevezhető feladatnak számít a **nyomon követés** (tracing) is, amely nagyon sok hallgatónak, főként a kezdőknek ismeretlen [9]. Hasonló következtetéshez jutott Fitzgerald és kutatótársai is [11], akik szerint a problémamegoldó képességnek a hiánya szoros kapcsolatban áll a hallgatók bizonytalan vagy éppenséggel kevés tudásával. Kutatásában ő is megfogalmazza azt, hogy mindez fejleszthető, ha külön figyelmet fordítunk a doodlingre vonatkozó stratégiákra. Ő említi meg többek között a „gondolkodj hangosan" ("think aloud") elvet, amely kapcsán arra próbálja ösztönözni a hallgatókat, hogy egy probléma vagy feladat megoldása

esetén legyenek tudatában annak, hogy melyik lépés következik, vagy miért cselekedtek úgy, ahogy ("What are you thinking?, Why did you do that?").

Figyelembe véve a diákok előzetes programozási tapasztalatait minden esetben találkozhatunk hiányosságokkal, mégis érzékelhető egy látványos szakadék a kezdők és a haladók gondolkodása között. Több kutató is megfogalmazza azt, hogy míg a haladó programozási tapasztalattal rendelkező diákok részfeladataira osztanak egy-egy nagyobb problémát, a kezdők ritkán, vagy egyáltalán nem alkalmazzák ezt a stratégiát és teljes egészében próbálják megoldani a feladatot [9]. Mindez kihatással van arra is, hogy nem alakul ki bennük az a rutin, hogy az egyszerűtől a bonyolult fele haladjanak. Lister és kutatótársai a sakkal szemlélteti és helyezi párhuzamba ezt a jelenséget [12], ahol arról számolnak be, hogy haladó és kezdő játékosok két külön módszert alkalmaznak a sakkfigurák elhelyezkedésének memorizálására. Chase és Simon [13] kutatása rávilágít arra, hogy a kezdők olyan módszert alkalmaznak, amellyel minden sakkfigura helyzetét külön-külön próbálják megjegyezni, elszigetelten a többitől (izolált gondolkodás), míg ezzel szemben a haladók a támadó és védekező lehetőségek alapján próbálják felidézni a sakkfigurák pozícióit (absztrakt gondolkodás). Mindez az algoritmikai feladatok megoldásában is tükröződik, ahol kezdők esetén különösen érzékelhető a „nem látja a fától az erdőt" jelenség [12]. Lister és kutatótársai azt is kihangsúlyozzák, hogy ennek elősegítésére érdemes figyelmet fordítani egy koherens struktúra kidolgozására, melyhez a feladat lépésről lépésre való nyomon követése jelentősen hozzá tud járulni.

2.2. A „vak számítógéppel" való azonosulás

Egy másik oka a problémamegoldó képesség hiányának lehet az, hogy a diákok nem tudnak azonosulni a „vak számítógéppel" [14]. Cunningham és kutatótársai úgy tartják, hogy a vázlatkészítés olyan, mint egy elosztott megismerés, és az, ami a vázlatot (a lapon végzett jelek, a folyamat, amely során ezek változnak) és a vázlat készítőjét (a diák) összeköti nem más, mint a diák munkájának megismerése, hogy egy közös válasz szülessen egy adott feladatra [7]. Kutatásukban azt is megfogalmazzák, hogy ez a nyomon követés elősegítheti, hogy a „gép" működése nyilvánvalóvá váljon. Több kutató is azonosul ezzel az elvvel, miszerint a hallgatónak meg kell ismernie, kell azonosulnia a „vak számítógép" [14], a „képzeletbeli számítógép" (notional machine) [15,16] működésével, hiszen ezáltal elősegíthető egy mentális modell kialakítása [7]. Mindezt, a kód olvasáshoz, -íráshoz, -hibajavításhoz kapcsolódó feladatok megismerése és lépésről lépésre való követése jelentősen elő tudják segíteni. Néhány kutató úgy hivatkozik a „képzeletbeli számítógéppel" való azonosulásra, mint „emberi fordító" (human compiler) vagyis az az aktív résztvevője a folyamatnak, aki elemzi, és lépésről lépésre kiértékeli a részeredményeket [14].

2.3. Interaktivitás

Azt gondolnánk, hogy vázlatot készíteni, „firkálni", egy önálló munkafolyamat, amely nem tartalmaz interaktív elemeket, sokszor mégis azt tapasztalhatjuk, hogy az interaktivitás egy további pozitív hozadéka lehet a doodlingnek. Feltevődik a kérdés azonban, hogy mitől interaktív egy vázlat elkészítése. Kirsh az olvasást hozza fel példának a téma kapcsán, amelyről úgy vélekedik, mint önmagában egy nem interaktív folyamat, mely mégis aktívan hathat az olvasóra. Olvasni, aláhúzni, vagy összefoglalni egy szöveget már igencsak interaktív lehet hiszen kialakul egy oda-vissza kapcsolat az olvasó és az olvasott szöveg között [10]. Éppen ezért a rendkívüli szemléltetések nagymértékben hozzájárulhatnak a megértéshez és a tanulási folyamat érdekesebbé tételéhez. Mivel egy feladat nyomon követése az a teljes folyamat, amely magával vonhatja a doodling jelenséget [17] azt is elmondhatjuk, hogy ennek alkalmazása az oktatásba sokkal vonzóbbá teheti a tananyag ismertetését és magát a tanulást is, hiszen közvetlen módon vonja be a diákokat az oktatási folyamatba [7].

2.4. Kérdés- és doodling típusok

Gyakran feltevődik azonban a kérdés, hogy miért jó a doodling, miért nem elégséges az, ha csak ülünk és gondolkodunk a helyes megoldáson. A doodling, a gondolatok papírra való leírása nagy-

mértékben függ a feladat típusától is. Gyakran tapasztalhatjuk, hogy amennyiben a megértés automatikus, nem foglalkozunk a megoldásmenet leírásával, azonban komplexebb kérdések esetén szinte minden esetben benne van a zsigereinkben, hogy papírt és ceruzát ragadjunk [10]. Mindez nem meglepő, hiszen az emberi gondolkodás egy „kognitív operációs rendszernek" tekinthető amely formákra, állapotokra és struktúrákra van tervezve [18].

Az a tény, hogy egy hallgató hogyan dolgoz fel egy problémát vagy hogyan közelíti meg az adott feladatot, nagymértékben befolyásolhatja a doodle típust, amit választ. Ennek kapcsán a Leeds kutatócsoport 12 doodle kategóriát határozott meg: váltakozó válasz (alternate answer), üres (blank page), számítás (computation), idegen nyomok, jelzések (extraneous marks), számvetés (keeping tally), szám (number), furcsa nyom (odd trace), pozíció (position), (szinkronizált nyomkövetés) synchronized trace, (nyomkövetés) trace, (aláhúzás) underlined, kizáró (ruled out) [9].

A kiválasztott doodle típust tehát nagymértékben befolyásolja a kérdések típusa is. Ahogy azt McCartney és kutatótársai is megfogalmazzák tanulmányukban, adott kérdések esetén bizonyos megjegyzések hatékonyabbak, mint mások, de bármilyen problémáról is legyen szó a megjegyzések jelenléte mindenképp jobb, mint az, ha egyáltalán nincsenek jelen annotációk. Ugyancsak ők tapasztalták azt, hogy kód-olvasással kapcsolatos problémák esetén a konkrét kóddal (fixed-code) kapcsolatos kérdéseknél, amelyek egy adott kódrészlet eredményét várták megoldásul a diákok többet „firkáltak", mint a kód-vázlatokkal (skeleton-code) kapcsolatos kérdések során [19].

Az említett szakirodalmi kutatások rávilágítanak arra, hogy egy algoritmus stratégia megértése nagymértékben függ a hallgatók problémamegoldó képességétől, mely szoros összefüggésben van azzal, hogy a hallgatók milyen mértékben követik nyomon papíron is az algoritmus lépéseit. Jelen kutatásunkban arra voltunk kíváncsiak, hogy hogyan teljesítenek a hallgatók két rendező algoritmus esetén, figyelembe véve a vizualizációk típusát, a doodling mértékét és a résztvevők előzetes programozási tapasztalatát is.

3. Kutatási kérdések

Az előzetes szakirodalmi kutatások alapján azt feltételeztük, hogy az algoritmusok típusa és a feladatok nehézsége befolyásolja a hallgatók doodlingre való hajlamát. Továbbá, úgy gondoltuk, hogy eltérő lehet az erre való hajlam és annak minősége kezdő és haladó programozási tapasztalattal rendelkező résztvevők esetén.

Ezek kapcsán a következő kutatási kérdéseket fogalmaztuk meg:

- Hogyan befolyásolja az algoritmusok típusa és a kérdések nehézségi szintje a hallgatók doodling technikáit?
- Milyen hatással van a hallgatók előzetes programozási tapasztalata a doodlingre?
- Mi az, ami befolyásolja a hallgatók „firkálásra" való hajlamát?

4. Módszertan

A kísérlet megvalósítására a 2022/2023-as tanév első félévének regisztrációs (előkészítő) hetén, a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi karán került sor. A felmérésen összesen 229 elsőéves egyetemi hallgató vett részt. A kutatás egy előzetes kérdőívet, két algoritmus bemutatását és ezekhez tartozó utótesztet foglalt magába.

4.1. Résztvevők

A kísérleten összesen 239 hallgató vett részt, melyek közül 12 nem töltötte ki megfelelően mindkét kérdőívet vagy nem vett részt a kísérlet második fázisán, így a végső eredmények meghatározásához

227 (30% lány) hallgató válaszait dolgoztuk fel. A résztvevők csoportját a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi Kar elsőéves hallgatói alkották, a következő egyetemi szakokról: Automatika és alkalmazott informatika, Fordító és tolmács, Gépészmérnöki, Informatika, Kertészmérnöki, Kommunikáció és közkapcsolatok, Közegészségügyi szolgáltatások és politikák, Mechatronika, Számítástechnika, Tájépítészet és Távközlés.

A résztvevőket az előzetes kérdőív alapján három kategóriába soroltuk az előzetes programozási tapasztalatot illetően: **nincs előzetes programozási tapasztalat** (PT_0: egyáltalán nem tanult programozást a középiskolás évek során), **alap programozási tapasztalattal** (PT_{1-3}: 1, 2 vagy 3 évet tanult programozást a középiskolás évek során, természettudomány osztály diákjai; heti 1-2 programozás óra; vagy osztály változtatás esete), és **magas programozási tapasztalattal** (PT_4: 4 évet tanult programozást a középiskolás évek során, matematika informatika osztály diákjai; heti 5-7 programozás óra) rendelkező diákok.

A kísérlet során három csoportot határoztunk meg a vizualizáció típusa függvényben: Animáció, Tánc, Színészi játék. A három csoportba véletlenszerűen osztottuk el a hallgatókat úgy, hogy a fent említett programozási kategóriák mindegyikéből egyforma arányba kerüljenek résztvevők.

4.2. Kutatási eszközök

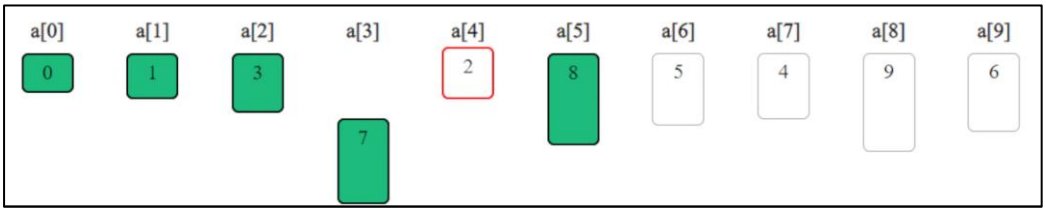
A kísérlet három nagy részre osztható fel: egy előzetes kérdőív, tanulási fázis és egy utóteszt. A kutatás során kitöltendő előzetes kérdőív és utóteszt is Google kérdőív segítségével került megvalósításra, míg a tanulási fázis videó szemléltetésével.

4.2.1. Előteszt

Az előzetes kérdőív összesen 10 kérdésből állt: 1 személyes adatok feldolgozására vonatkozó kérdés (GDPR), 2 demográfiai adatok feldolgozására vonatkozó kérdés, 2 szakterületre vonatkozó kérdés, 4 érettségi eredményekre vonatkozó kérdés és 1 előzetes programozási tapasztalatra vonatkozó kérdés (Hány évet tanultál programozást középiskolában?).

4.2.2. Tanulási fázis

A tanulási fázis két algoritmus bemutatását foglalta magába. A bemutatás videók vetítése segítségével valósult meg. Minden vizualizáció típushoz (animáció, tánc, színészi játék) algoritmusonként (beszúró-, shell rendezés) tartozott egy felvezető videó (tanár által szemléltetett bemutatás), egy algoritmus stratégiát bemutató videó, és végül egy záró videó (tanár által szemléltetett bemutatás). A második algoritmus jellegénél fogva, mivel egy nehezebb rendezési stratégiát mutatott be még kiegészült egy segítségként szolgáló videóval is.



1. ábra: Algoritmus reprezentáció animációval

2. ábra: Algoritmus reprezentáció tánccal

3. ábra: Algoritmus reprezentáció színészi játékkal

4.2.3. Utóteszt

Az utóteszt kérdései 4 felvonásra voltak osztva melyre összesen 18 pontot lehetett összegyűjteni:

Az első felvonás 1 demográfiai adatokra vonatkozó kérdést és egy szubjektív véleményhez tartozó kérdést („Te melyik vizualizációval tanulnál legszívesebben?") foglalt magába.

A második felvonás az első rendezési algoritmusra, a beszúró rendezésre vonatkozó kérdéseket tartalmazta: 2 alapműveletre, 3 belső műveletre és 4 algoritmus bonyolultságra vonatkozó kérdést.

A harmadik felvonás a második rendező algoritmushoz, a shell rendezéshez tartozó kérdéseket tartalmazta. Az előző algoritmushoz hasonlóan ebben az esetben is 2 alapműveletre, 3 belső műveletre és 4 algoritmus bonyolultsági kérdésre vártuk a hallgatók válaszát.

Végezetül, a negyedik felvonás néhány záró kérdést tartalmazott, amelyek során a résztvevők szubjektív véleményére voltunk kíváncsiak az algoritmus megértésével, az ábrázolásmód érdekességével, valamint a tanulási élménnyel kapcsolatosan. Ezek a kérdések nem számítottak bele a végső pontozásba.

A felvonások és az azokhoz tartozó kérdések részletes leírását és típusát az alábbi táblázat szemlélteti.

Kérdés száma	Kérdés leírása	Kérdés típusa
Első felvonás kérdései		
1	Név	nyílt
2	Melyik vizualizációval tanulnál a legszívesebben? (animáció, tánc, színészi játék)	egyválaszos
Második felvonás kérdései (beszúró rendezés)		
A következő 7 elemű számsorozaton: 1, 19, 7, 8, 12, 11, 9		
1	Melyik két szám kerül először összehasonlításra?	nyílt
2	Melyik két szám kerül először kicserélésre?	nyílt
3	Milyen művelet következik a 19-es és 12-es számok összehasonlítása után?	Egyválaszos
4	Melyik két szám kerül összehasonlításra a 19-es és 11-es számok összehasonlítása után?	nyílt
5	Melyik két szám kerül összehasonlításra a 11-es és 9-es számok összehasonlítása után?	nyílt
Algoritmus bonyolultsági kérdések		
6	Ha már eredetileg szigorúan növekvő sorrendben van egy 7 elemű számsorozat, hány összehasonlításra kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
7	Ha már eredetileg szigorúan növekvő sorrendben van egy 7 elemű számsorozat, hány cserére kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
8	Ha már eredetileg szigorúan csökkenő sorrendben van egy 7 elemű számsorozat, hány összehasonlításra kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
9	Ha már eredetileg szigorúan csökkenő sorrendben van egy 7 elemű számsorozat, hány cserére kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt

Harmadik felvonás kérdései (shell rendezés)		
A következő 7 elemű számsorozaton: 1, 19, 7, 8, 12, 11, 9		
1	Melyik két szám kerül először összehasonlításra?	nyílt
2	Melyik két szám kerül először kicserélésre?	nyílt
3	Milyen művelet következik a 7-es és a 11-es számok összehasonlítása után?	egyválaszos
4	Melyik két szám kerül összehasonlításra a 8-as és 9-es számok összehasonlítása után?	nyílt
5	Melyik két szám kerül összehasonlításra a 7-es és 8-as számok összehasonlítása után?	nyílt
Algoritmus bonyolultsági kérdések		
6-9	A beszúró rendezéshez hasonlóan 4 algoritmusra vonatkozó bonyolultsági kérdés.	nyílt kérdések
Negyedik felvonás kérdései (szubjektív vélemények)		
1	Milyen mértékben járult hozzá az ábrázolási mód az algoritmus megértéséhez?	skála
2	Mennyire kötötte le az ábrázolási mód a figyelmedet?	skála
3	Milyen mértékben járult hozzá az ábrázolási mód a tanulási élményhez?	skála

1. táblázat: Utóteszt kérdései

4.3. A kutatás menete

A kutatás menetét tekintve az előzetes kérdőív kitöltésére a felmérés előtti nap került sor. A résztvevőknek 8 óra állt rendelkezésükre kitölteni a kérdőívet. Ezt követően a beírt adatoknak megfelelően a diákokat három, a programozási előismereteknek megfelelően kiegyensúlyozott csoportba osztottuk. Másnap a résztvevők ennek a beosztásnak megfelelően helyezkedtek el az egyetem három különböző előadótermében, ahol egy-egy oktató fogadta őket. A három terem három különböző vizualizációval tanult: animáció, tánc, színészi játék.

A kísérlet mindhárom teremben egyforma módon zajlott. A résztvevőknek két algoritmus vizualizációt kellett megtekinteniük, a típusnak (animáció, tánc, színészi játék) megfelelő módon. Mindkét algoritmus stratégia bemutatása kétszer került vetítésre. Minden algoritmusvizualizáció egy bevezető videóval kezdődött, és egy záró videóval végződött, mely minden csoport esetén egyforma volt. Az utóteszt különböző felvonásainak kérdéseire a diákoknak megszakításokkal kellett válaszolniuk. Minden felvonás, az adott videó megtekintése után került megválaszolásra. A shell rendezési stratégia esetén minden csoport megtekinthetett egy kiegészítő videót is, lévén egy nehezebb rendezési stratégiáról szó.

A kutatás menetének részletes beosztását az alábbi táblázat szemlélteti:

Cs_1: Animáció	**Cs_2: Tánc**	**Cs_3: Színészi játék**
Utóteszt - Első felvonás kérdéseinek megválaszolása		
Bevezető videó - beszúró rendezés		
Beszúró rendezés videójának megtekintése **animációval** X2	Beszúró rendezés videójának megtekintése **tánccal** X2	Beszúró rendezés videójának megtekintése **színészi játékkal** X2
Utóteszt - Második felvonás kérdéseinek megválaszolása (beszúró rendezés)		
Bevezető videó - shell rendezés		
Shell rendezés videójának megtekintése **animációval** X2	Shell rendezés videójának megtekintése **tánccal** X2	Shell rendezés videójának megtekintése **színészi játékkal** X2
Kiegészítő videó - shell rendezés		
Utóteszt - III. felvonás kérdéseinek megválaszolása (shell rendezés)		
A kísérlet záró videója		
Utóteszt - IV. felvonás kérdéseinek megválaszolása (szubjektív vélemények összegyűjtése)		

2. táblázat: A kutatás menete

A kísérlet elején a résztvevők tudtára adtuk, hogy a kérdések során bármikor kérhetnek piszkozat lapokat, vagyis ezeket nem adtuk oda mindenkinek a kérdőív kitöltése előtt, csak abban az időpillanatban amikor igényelte az illető személy.

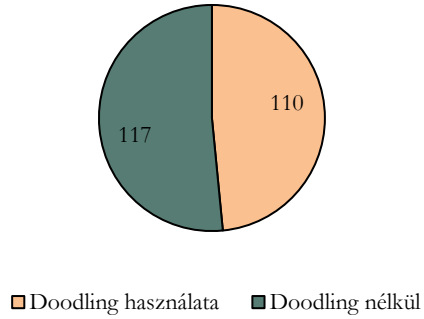
5. Eredmények

A kísérlet során több szempontot is vizsgáltunk, ami a résztvevők algoritmikus gondolkodását és problémamegoldó képességét illeti. Ebben a tanulmányban a piszkozat használatára, vagyis a doodling hatására fókuszáltunk, melyet a következő szempontok szerint vizsgáltunk: a résztvevők előzetes programozási tapasztalata, neme, a vizualizáció típusa és az alkalmazott doodling technikák.

5.1. A doodling hatása a teljesítményre

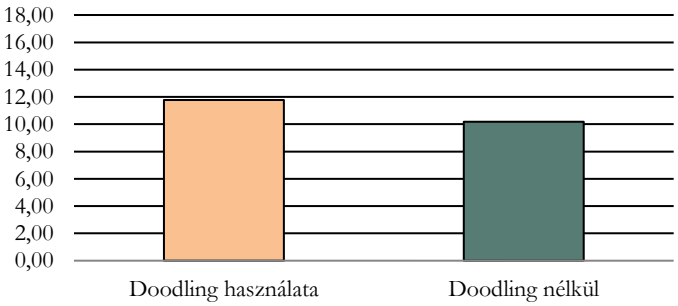
Elsőként kíváncsiak voltunk arra, hogy miként befolyásolja a hallgatók eredményét, vagyis végső pontszámát az, hogy használnak piszkozatot vagy sem. Amint azt már korábban is említettük, a kísérlet során teljesen fakultatív dolognak számított a piszkozatlap kérése. Ezeket csak azon hallgatóknak osztottuk ki, akik igényelték.

Ahogy azt a 4. ábra is szemlélteti, a résztvevők szinte fele-fele arányban kértek piszkozatot (110 hallgató) vagy anélkül dolgoztak (117 hallgató).



4. ábra: Piszkozatok használatának eloszlása

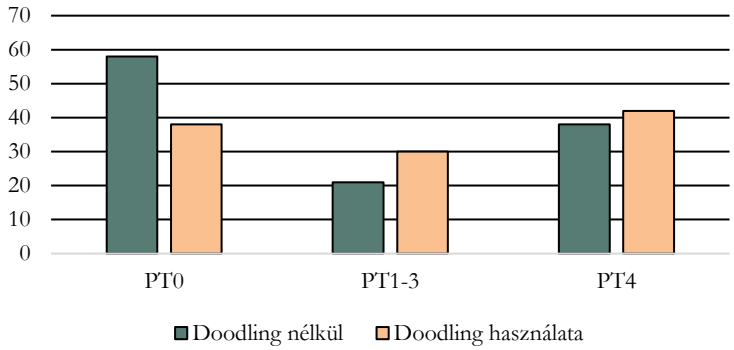
Az eredmények feldolgozását követően a két csoport látszólag szinte teljesen egyformán teljesített (ábra). Azok a résztvevők, akik piszkozatot használtak a maximális 18 pontból átlagosan 11.77 pontot gyűjtöttek össze, míg azok, akik piszkozat nélkül, átlagosan 10.18 pontot (5. ábra). Páros T próba alkalmazása után arra a következtetésre jutottunk, hogy a piszkozatot igénylő hallgatók szignifikánsan jobban teljesítettek, mint a piszkozat nélküliek ($p=0<0.005$). Ez a jelenség várható volt, hiszen számos kutató arról számol be, hogy bár a piszkozatra leírt megjegyzések típusa eltérő lehet, egy dolog bizonyos: a doodling alapjáraton elősegíti a helyes válasz meghatározását [5,19]. Hasonlóképpen, további kutatócsoportok [7,9] is kihangsúlyozzák azt a jelenséget kód olvasási problémák kapcsán, hogy azok, akik különféle nyomkövetési technikákat alkalmaznak, mint például a doodling, jobb teljesítményt érnek el.



5. ábra: Doodling hatása az eredményekre

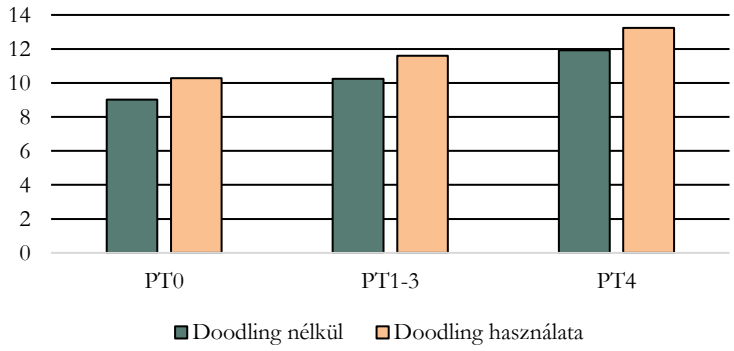
5.2. A doodling és az előzetes programozási tapasztalat közötti összefüggés

A továbbiakban kíváncsiak voltunk arra is, hogy milyen mértékben befolyásolja a résztvevők előzetes programozási tapasztalata a doodlingre való hajlamot. Az eredmények feldolgozását követően azt tapasztaltuk, hogy a több programozási tapasztalattal rendelkező hallgatók kategóriája a piszkozatok igénylési arányát tekintve felülmúlta a kezdőkét (PT_0: 96 diákból 38, $PT_{1\text{-}3}$: 51 diákból 30, PT_4: 80 diákból 42 igényelt piszkozatot). Érdekes módon csak a PT_0 kategóriára volt igaz az a jelenség, hogy többen nem kértek piszkozatot, mint ahányan kértek (6. ábra).



6. ábra: Doodling használata programozási tapasztalat szerint

A kérdésekre adott válaszok feldolgozását követően itt is várható volt az eredmény: azon diákok, akik piszkozatot igényeltek minden esetben (PT_0, $PT_{1\text{-}3}$ és PT_4 esetén is) jobban teljesítettek, mint azon társaik, akik nem (**7. ábra**). Mi több, a kezdő (P_0) és haladó (PT_4) programozási tapasztalattal rendelkezők kategóriájában is szignifikáns különbséghez jutottunk (P_0: $p=0.04<0.05$; P_4: $p=0.03<0.05$).



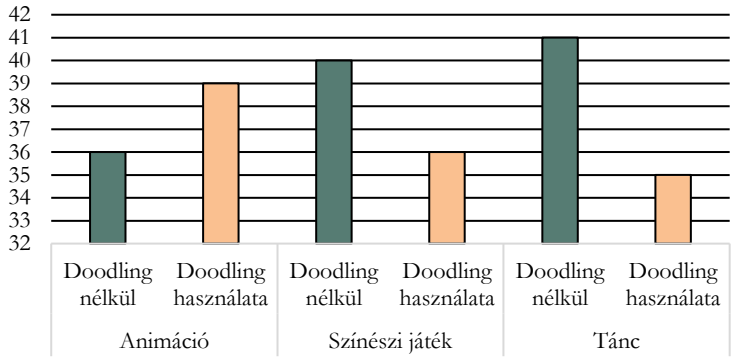
7. ábra: A doodling hatása az eredményekre programozási tapasztalat szerint

Eredményeink arra engedtek következtetni, hogy a haladók sokkal nagyobb valószínűséggel kezdenek el piszkozatot használni, mint a kezdők, melyet Whalley és kutatótársai [5] is megfogalmaztak. Ennek a jelenségnek Perkins és kollégái szerint akár több oka is lehet, amely a doodlinghez kapcsolható, mint például az, hogy sok hallgató nem érti miért is lehet hasznos számára az ilyen jellegű nyomon követése a feladatnak, vagy egyszerűen nem magabiztosak abban, hogy helyesen tudnának jegyzetelni [20]. Úgy véljük ez a bizonytalanság a mi esetünkben is hatással lehetett a kezdők piszkozat igénylési hajlamára.

5.3. A vizualizáció típusának hatása a doodlingre

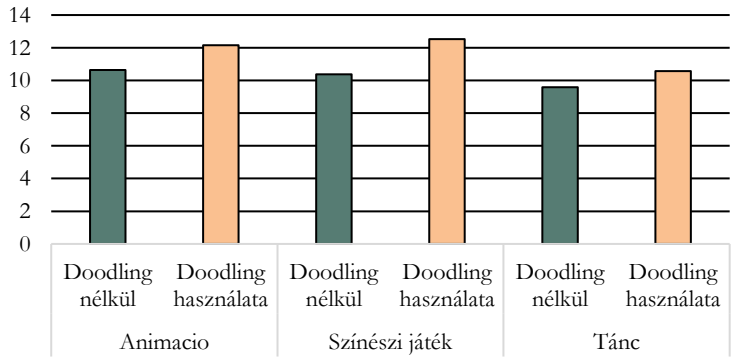
Arra is kíváncsiak voltunk, hogy miként befolyásolja az adott vizualizáció típusa azt, hogy a diákok használnak-e piszkozatot vagy sem. Hasonlóképpen, először összesítettük, hogy hány piszkozat volt igényelve külön-külön a három csoportban (8. ábra). Az animációval tanuló hallgatók közül a 75 hallgatóból összesen 39-en, a színészi játék esetén a 76 hallgatóból 36-on, míg a tánccal bemutatott algoritmusvizualizáció esetén 76 hallgatóból 35-ön alkalmaztak doodlinget a feladatmegoldás során. Érdekes módon, csak az animációs csoportra volt igaz az a kijelentés, hogy többen használtak pisz-

kozatot, mint sem. Ennek oka az is lehet, hogy az animáció absztrakt jellege miatt jobban ösztönzi a diákokat abban, hogy piszkozatot ragadjanak és nyomon kövessék az algoritmus lépéseit. Ehhez hasonló okot fogalmazott meg Cunningham és munkatársai is, akik szerint az ábrák, formák több tantárgy esetén is, mint például matematika, fizika, kémia, magukkal vonják és ösztönzik a diákokat a piszkozatírásra [7].



8. ábra: Doodling használata az algoritmusvizualizáció típusa szerint

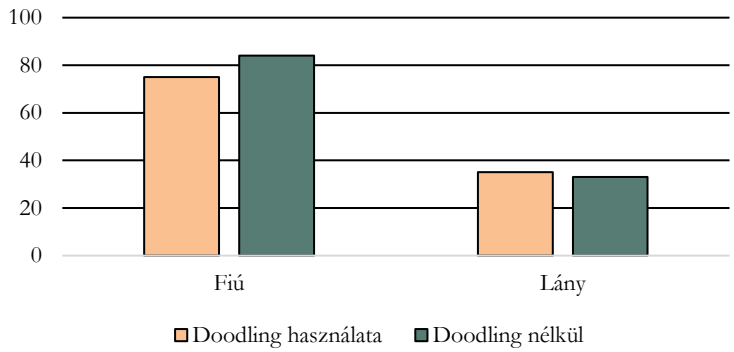
A résztvevők pontszámát figyelembe véve a következőképpen alakultak az átlagok: animációs csoportban (doodling használata: 12.15, doodling nélkül: 10.63), színészi játék csoportban (doodling használata: 12.52, doodling nélkül: 10.37) és tánccal bemutatott vizualizáció csoportban (doodling használata: 10.57, doodling nélkül: 9.58). Várható módon, mindhárom csoport esetén a piszkozatot használt hallgatók átlagos teljesítménye jobb volt, mint azon diákoké, akik nem használtak piszkozatot. Az animáció és színészi játék csoportokban a doodling szignifikánsan magasabb átlageredményekhez (animáció: $p=0.04<0.05$, színészi játék: $p=0.002<0.005$) vezetett (9. ábra).



9. ábra: Doodling hatása az eredményekre az algoritmusvizualizáció típusa szerint

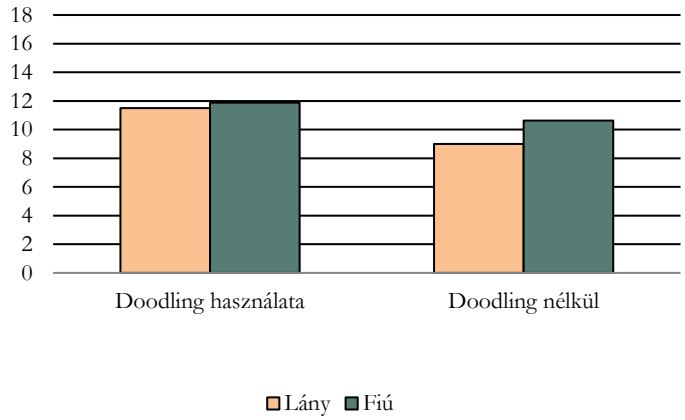
5.4. Nemek

Bár a diákok nemek szerinti eloszlását tekintve jelentősen kevesebb lány vett részt a kísérleten (30%) mégis kíváncsiak voltunk, hogy milyen mértékben jellemző a lányokra, illetve a fiúkra az, hogy doodlinget alkalmazzanak a feladatmegoldás során. A piszkozatok számát tekintve szinte fele-fele arányban kértek piszkozatot úgy a lányok (doodling: 51%), mint a fiúk (doodling: 47%).



10. ábra: Doodling használata nemek szerint

A pontszámokat tekintve (11. ábra) a doodlinget alkalmazó lányok 11.51pontot szereztek átlagosan, mellyel szignifikánsan jobban teljesítettek ($p=0.003<0.005$), mint azok a lányok, akik nem használtak piszkozatot (9 pont). A fiúk esetén is hasonló eredményhez jutottunk, hiszen a doodlinget alkalmazó fiúk (11.89 pont) szignifikánsan jobban teljesítettek ($p=0.01<0.05$), mint társaik (10.64 pont). Az előző eredményeink és a szakirodalmi kutatások alapján mindez várható volt, ezért azt is szerettük volna megvizsgálni, hogy milyen különbségekhez vezet az, ha összemérjük a fiúk és lányok eredményeit. Bár külön elemezve a fiúk és lányok eredményeit szignifikáns különbségekhez jutottunk, összehasonlítva ezeket lényegesen eltérő különbség csak a piszkozat nélküli kategóriákban volt észlelhető ($p=0.01<0.05$), míg doodlinget alkalmazó lányok nem maradtak le lényegesen az ugyancsak doodlinget alkalmazó fiúktól ($p=0.29>0.05$).

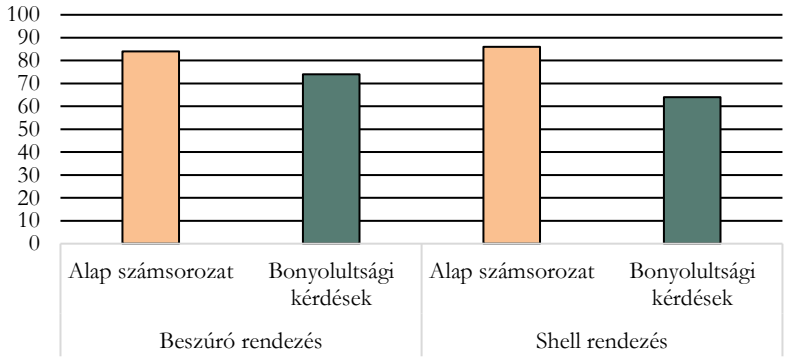


11. ábra: Doodling hatása az eredményekre nemek szerint

Mindez arra tud ösztönözni, hogy bár nyilvánvalónak bizonyul a piszkozat hasznossága, ennek kettős pozitívuma is lehet, ha a nemek eloszlását tekintjük. A lányoknak kifejezetten segítségére lehet az, ha levezetik az algoritmusok lépéseit, mely által csökkenthető a gyakran észlelhető szakadék fiúk és lányok teljesítménye között.

5.5. A doodling és a kérdéstípusok közötti összefüggés

Végül, de nem utolsó sorban szerettük volna közelebbről is megvizsgálni a piszkozatokat és elemezni a hallgatók által írt megjegyzéseket és firkákat algoritmusok, valamint kérdések szerint is (12. ábra). A kérdések összetételét tekintve mindkét algoritmus (beszúró- és shell rendezés) esetén két kategóriát vettünk figyelembe: (1) alap számsorozattal kapcsolatos kérdések, (2) algoritmus bonyolultsági kérdések. A doodling számát tekintve a 227 hallgatóból összesen 44 hallgató használta minden feladat esetén a piszkozatot. Külön-külön a két algoritmus esetén pedig összesen 158 (beszúró rendezés: alap számsorozat - 84, bonyolultsági kérdések - 74), illetve 150 (shell rendezés: alap számsorozat - 86, bonyolultsági kérdések - 64) hallgató használt piszkozatot.



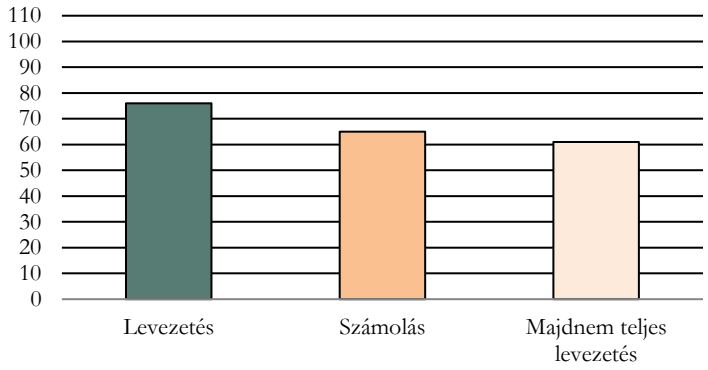
12. ábra: Doodling használata kérdéstípusok szerint

Bár Kirsh és kutatótársai úgy tapasztalták, hogy a komplex problémák nagyobb eséllyel vonzzák a piszkozat használatát [10], ez a jelenség nálunk nem igazolódott be. Érdekes módon, a beszúró rendezés kapcsán többen firkálták le a megoldásmenetet, mint a shell rendezés esetén, amely egy jellegében nehezebb algoritmusnak számít. Ennek ellenére sem az alap számsorozattal kapcsolatos feladatok, sem az algoritmus bonyolultsági kérdések esetén nem volt szignifikáns különbség a két algoritmusra kapott pontszámok között (alap számsorozat: $p=0.32>0.05$, bonyolultsági kérdések: $p=0.08>0.05$, összességében: $p=0.22>0.05$).

Annak oka, hogy kevesebb doodling volt a shell algoritmus során az is lehet, hogy a hallgatók nem értették az algoritmust és ez a tudáshiány, bizonytalanság ahhoz vezetett, hogy nem is kezdték el az algoritmus lépéseit papíron ábrázolni [19]. Az általunk mért eredmények bizonyos mértékben harmonizálnak McCartney és társai következtetésivel, akik kutatásukban megfogalmazták, hogy a piszkozat hiányának a hallgatók bizonytalansága mellett számos más okozója is lehet, mint például az, hogy a hallgatók nem rendelkeznek elegendő ismerettel a probléma megoldásához, vagy gyenge a résztvevők problémamegoldó képessége [19].

A fent említett jelenség az alap számsorozattal és bonyolultsági kérdések kapcsán is érzékelhető volt. Bár a bonyolultsági kérdések jelentősen több ismeretet és jó problémamegoldó képességet igényelnek, a piszkozathasználat ebben az esetben is háttérbe szorult, míg az alap feladatok esetén nem. Ennek egy másik oka a kérdés típusából adódó jelleg is lehet. Ahogy azt McCartney és kutatótársai is megfogalmazták a kérdések milyensége jelentősen befolyásolhatja azt, hogy a hallgató hogyan teljesít, illetve azt is, hogy milyen megjegyzéseket, firkákat jegyez le a papírra [19]. Hasonlóképpen, Whalley és társai is kihangsúlyozták tanulmányukban azt, hogy bizonyos kérdéstípusok, mint például a "fixed-code" jellegű kérdések jobban előídézik a doodle használatát [5]. Ehhez hasonlóan, a mi kísérletünk során is több megjegyzés és firka érkezett az alap számsorozatra vonatkozó kérdésekre ("fixed-sequence"), mely azonosítható a "fixed-code" típusú feladatokkal.

Ezt a jelenséget a 13. ábra is megerősíti, mely segítségével a levezetések és számolások közötti összefüggéseket szerettünk volna szemléltetni. A levezetések (76 hallgató) rendszerint az alap számsorozatra, míg a számolások az algoritmus bonyolultsági kérdésekre (65), vagyis az általánosításra vonatkozó feladatoknál voltak hasznosak.

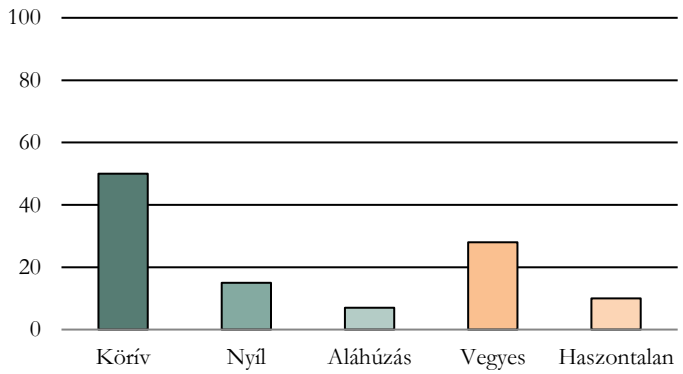


13. ábra: Levezetések és számolások

Érdekességképp, az is megemlítendő, hogy a 110 piszkozatot használó hallgató közül több, mint fele (61 hallgató) vezette le teljesen, vagy majdnem teljes egészében a két algoritmus lépéseit a piszkozatlapra.

A McCartney kutatócsoporthoz hasonlóan [19], mi is meghatároztunk 5 doodling kategóriát, melyet vizsgáltunk a résztvevők piszkozatlapjain: (1) körív: rendszerint két szám összehasonlításának/cseréjének szemléltetésére, (2) nyíl: rendszerint az algoritmus egy új állapotának meghatározására, (3) aláhúzás: rendszerint a számsorozatban szereplő aktív értékek szemléltetésére, (4) vegyes: az előzőek közül több együttes használata, (5) zavaros: rendezetlen, következetlen piszkozat, kihúzásokkal, összefüggéstelen jelekkel (14. ábra).

Érdekes módon, bár a résztvevők többségének a különböző algoritmus lépéseit szemléltető jelek ismeretlenek voltak, előszeretettel használták a körív (50 hallgató) jelet két elem összehasonlításakor/cseréjekor. Az aláhúzás (7 hallgató), lévén egy ritkább jelölési forma, mely inkább azok számára volt ismert, akik tanultak már programozást jelentősen kevesebb piszkozatlapon volt látható, míg a nyíl jelzés 15 hallgató piszkozatán. Emellett, összesen 28 résztvevő használta vegyesen a fent említett jeleket, míg 10 hallgató piszkozata zavarosnak, következetlennek bizonyult. Fontos megemlíteni, hogy jelen kísérlet során a hallgatóknak igényelni kellett a piszkozatot. Ennek értelmében, összesen 117 hallgató nem is használt doodlinget a felmérés során, melyet asszociálhatunk az úgynevezett üres megjegyzésekkel.



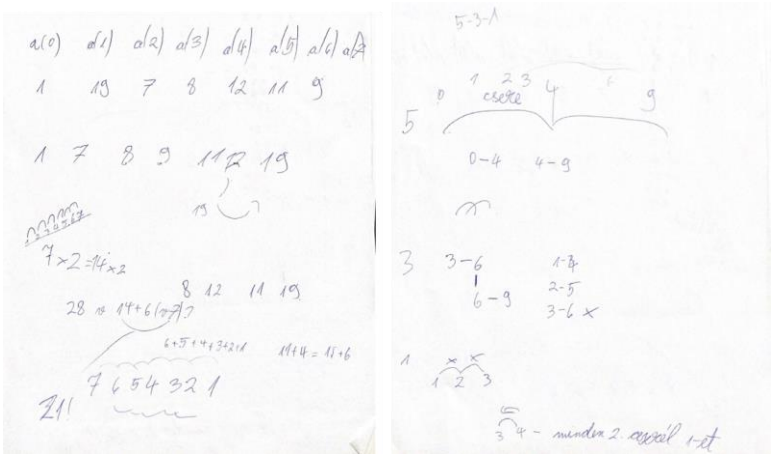
14. ábra: Jelek használata a piszkozaton

5.6. Különleges - Haszontalan piszkozatok

Ebben a részben szeretnénk néhány számunkra különlegesnek, illetve zavarosnak minősített piszkozatot szemléltetni. Fontos megjegyezni, hogy a 110 piszkozat közül több is különlegesnek nevezhető, ezek közül azonban csak néhányat fogunk bemutatni.

Az alábbi doodling technikákat azért tartottuk különlegesnek, mert felhasznált jelzések (nyíl, körív stb) mellett, fellelhetőek további rendszerezettségre és következetességre utaló jelek is.

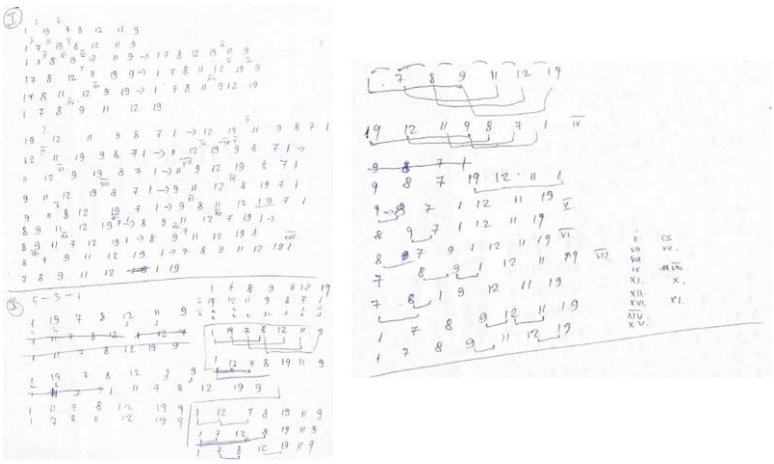
Az 1. piszkozat esetén (15. ábra) a hallgató szemléltette a tömb (a[0], a[1].. a[9]) adatszerkezetet is annak ellenére, hogy 0 év programozási tapasztalattal rendelkezett. Emellett, tömör megjegyzéseket is írt a lapjára. Ennek ellenére érdekes módon összesen 3 pontot kapott a tesztre.



15. ábra: 1. piszkozat - PT_0, pontszám: 3

A 2. piszkozat (**16. ábra**) az arab számok mellett római számokat is használt a két algoritmus elkülönítésére, valamint az algoritmus lépéseinek nyomon követésére is. Az is megfigyelhető, hogy az összehasonlítások és cserék nyilak, valamint körívek felhasználásával egyaránt szemléltetve vannak.

Hasonlóképpen, ez a hallgató is 0 programozási tapasztalattal rendelkezett a teszt kitöltésekor, ennek ellenére 13 pontot sikerült összegyűjtenie a teszten.

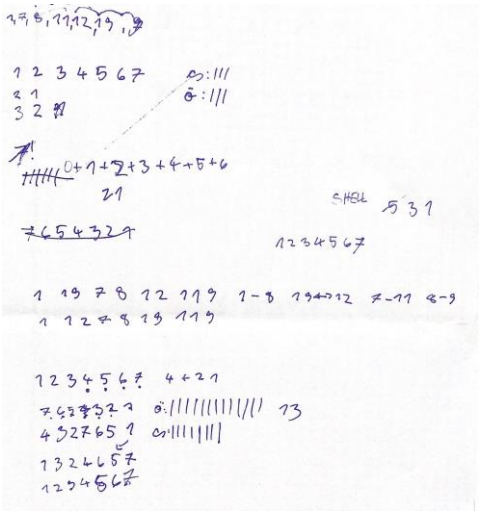


16. ábra: 2. piszkozat - PT_0, pontszám: 13

A 3. piszkozat (17. ábra) az egyik leghosszabbnak nyilvánított piszkozat volt, ahol részletesen le volt vezetve az algoritmus minden lépése. A piszkozat különlegessége abban is megmutatkozik, hogy egyszeres, illetve többszörös aláhúzásokkal vannak jelölve a számsorozatban szereplő értékek és a megjegyzések sem maradnak el. A piszkozat tulajdonosa 2 év előzetes programozási tapasztalattal rendelkezett a felmérés idején, és 16 pontot kapott a tesztre, amely igen jó eredménynek számít a maximálisan megszerezhető 18 pontból.

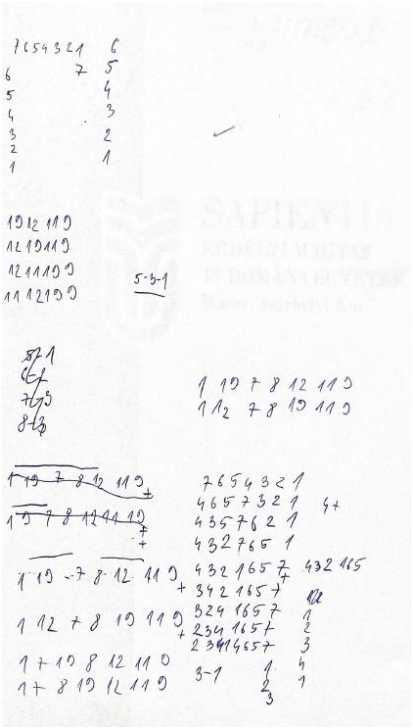
17. ábra: 3. piszkozat – $PT_{1\text{-}3}$, pontszám: 16

A 4. piszkozat (18. ábra) az előző háromhoz hasonlóan kevés, pontosabban 0 előzetes programozási tapasztalattal rendelkező hallgató tulajdona, aki a teszten ennek ellenére 15 pontot gyűjtött össze. A tömör doodling ellenére, sok helyes fázisa fedezhető fel az algoritmusnak, mely minden bizonnyal segítette a hallgatót a válaszadásban.

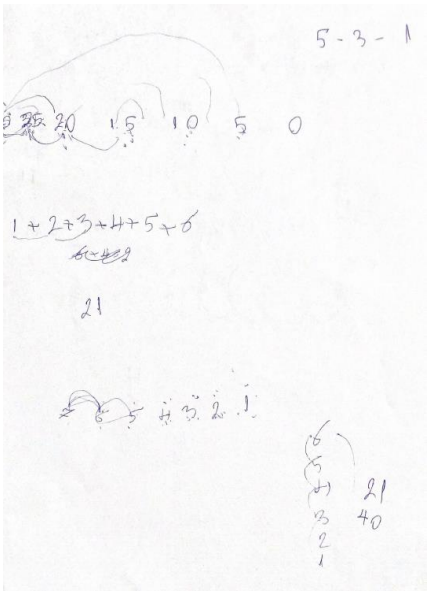


18. ábra: 3. piszkozat - PT_0, pontszám: 15

A zavaros vagy nem követhető piszkozatok kapcsán három példát szeretnénk szemléltetni. Az első két piszkozat mindegyike egy-egy 4 év előzetes programozási tapasztalattal rendelkező hallgató tulajdona (19. és 20. ábra). Érdekes módon, a piszkozat alapján kevés algoritmus stratégiájára vonatkozó rész fedezhető fel, a megoldásmenet is nehezebben követhető, sok esetben a számsorozat sem megfelelő. Ennek ellenére a hallgatók összesen 12, illetve 14 pontot szereztek a teszten.

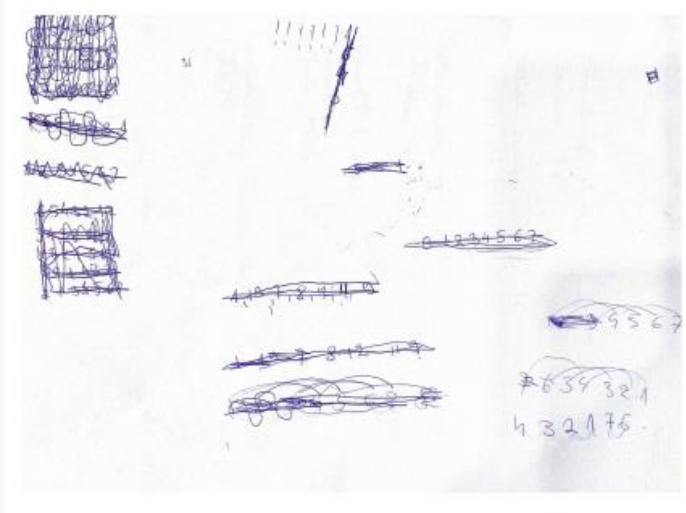


19. ábra: 1. piszkozat – PT_4, pontszám: 12



20. ábra: 2. piszkozat – PT_4, pontszám: 14

A harmadik zavarosnak nevezhető piszkozat (21. ábra) tulajdonosa ismeretlen, így az előzetes programozási tapasztalat mértékét és a pontszámot nem volt lehetőségünk meghatározni. Az azonban bizonyos, hogy a résztvevőnek minden próbálkozása ellenére nem sikerült megoldáshoz jutnia, vagy legalábbis a piszkozat szintjén nem volt látható az algoritmus lépéseinek nyomon követése.



21. ábra: 3. piszkozat – ismeretlen adatok

Bár a fent bemutatott piszkozatok csak töredékét képezik az összképnek, érdekesnek mondható az a tény, hogy sok 0 programozási tapasztalattal rendelkező hallgató készített rendszerezett piszkozatot, míg a magas programozási tapasztalattal rendelkező résztvevők közül sokan felületesen, zavarosan vetik papírra ötleteiket.

A pontszámok ennek ellenére nem feltétlen azonosíthatók a gondosan és rendszerezetten elkészített piszkozatokhoz. Sok esetben a felületes, zavaros és kódolt piszkozatok tulajdonosai szereztek nagyobb pontszámot.

6. Limitációk

A kísérlet lehetőséget biztosított arra, hogy felmérjük és megvizsgáljuk a résztvevők doodling technikáit különböző szempontok szerint. Kutatásunk egyik limitációja, hogy a kiosztott piszkozatlap nem volt feladatonként külön szegmensekre osztva. Ennek segítségével sokkal pontosabban és egyszerűbben tudtuk volna összesíteni a hallgatók eredményeit. Ugyancsak ehhez kapcsolódhat az a limitáció, miszerint a hallgatók utótesztje Google kérdőív keretén belül volt megvalósítva. Minden bizonnyal, egy papír alapú kérdőív újabb befolyásoló tényező lehet, ami a résztvevők piszkozatírási hajlamát illeti.

7. Összegzés

Jelen tanulmányban azt vizsgáltuk meg, hogy milyen a hallgatók doodling technikájának minősége, illetve milyen tényezők befolyásolják a piszkozatírásra való hajlamot.

Az eredmények és az előzetes szakirodalmi kutatások is egyértelműen arra utalnak, hogy a piszkozat önmagában jobb eredményekhez vezetnek. Legyen nemekről, előzetes programozási tapasztalatról vagy épp vizualizáció típusokról szó, azon diákok, akik használnak piszkozatot, összességében jobb eredményt érnek, mint azok, akik egyáltalán nem használnak.

A hallgatók doodlingre való hajlamát számos tényező befolyásolhatja. Az egyik ilyen tényező lehet a hallgatók előzetes programozási tapasztalata. Eredményeink arra engedtek következtetni, hogy a haladó programozási tapasztalattal rendelkező diákok nagyobb valószínűséggel ragadnak papírt és ceruzát annak érdekében, hogy nyomon kövessék az algoritmus lépéseit. Ennek egyik oka a kezdő programozási tapasztalattal rendelkező hallgatók bizonytalansága, vagy akár a probléma megoldó képesség hiánya is lehet.

A kísérlet során arra is választ kaptunk, hogy a feladatok és kérdések milyensége is nagy hatással lehet a doodlingre való hajlamra. Az utóteszt kérdései alapján azt tapasztaltuk, hogy az alapfeladatoknak számító, levezetést igénylő feladatok jobban igénylik a piszkozat használatát, mint a jellegüknél fogva elvont kategóriába tartozó kérdések.

Az ábrázolás tekintetében, arra utalnak az eredmények, hogy a leginkább az animációval bemutatott algoritmus vizualizáció idézi elő a piszkozatlap használatát. Ennek oka az animációban megtalálható ábrák, formák jelenléte és annak absztrakt jellege lehet, mely sokkal inkább társítható a szabad kézzel elkészített firkákhoz, mint a másik kép vizualizáció.

A tanulmány számos érdekes jelenségre is rávilágított, mint például arra, hogy sok résztvevő esetén az összehasonlítások és cserék műveletét szinte tudat alatt is a körívekkel azonosítják. Ennek kapcsán is érzékelhető, hogy a diákok mondhatni automatikusan kialakítják önmagukban piszkozatírási technikáikat, ez azonban nagyon sokszor felületesnek bizonyul.

Jelen dolgozat arra is felhívta figyelmünket, hogy fontos a hallgatók doodling technikáira is fókuszálni, hiszen ez nagymértékben befolyásolhatja a továbbiakban a programozás oktatásban való előrehaladásukat. Véleményünk szerint ezek a technikák kellőképpen fejleszthetők, ha a mindennapi

informatika tanórákba helyet keresünk a debuggolás, nyomkövetés, program olvasás, program felismerés és hasonló jellegű feladatoknak.

8. Köszönetnyilvánítás

A jelen munkát Magyarország Collegium Talentum programja támogatta. Köszönjük továbbá a Kutatási Programok Intézetének támogatását is.

9. Irodalom

1. Kátai, Z., & Osztián, E. (2021). Improving AlgoRythmics Teaching-Learning Environment by Asking Questions. International Journal of Instruction, 14(2), 27-44.
2. Kátai, Z. (2021). Algorythmics. Technologically and artistically enhanced computer science education.
3. Nagy, E. J., Osztián, P. R., Cosma, C., Kátai, Z., & Osztián, E. (2019, June). Looking for the Optimal Interactivity Level in the AlgoRythmics Learning Environment. In EdMedia+ Innovate Learning (pp. 106-114). Association for the Advancement of Computing in Education (AACE).
4. Osztián, P. R., Kátai, Z., & Osztián, E. (2020, October). Algorithm Visualization Environments: Degree of interactivity as an influence on student-learning. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.
5. Whalley, J., Prasad, C., & Kumar, P. A. (2007, January). Decoding doodles: novice programmers and their annotations. In ACM International Conference Proceeding Series (Vol. 239, pp. 171-178).
6. Hertz, M., & Jump, M. (2013, March). Trace-based teaching in early programming courses. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 561-566).
7. Cunningham, K., Blanchard, S., Ericson, B., & Guzdial, M. (2017, August). Using tracing and sketching to solve programming problems: replicating and extending an analysis of what students draw. In Proceedings of the 2017 ACM Conference on international computing education research (pp. 164-172).
8. Xie, B., Nelson, G. L., & Ko, A. J. (2018, February). An explicit strategy to scaffold novice program tracing. In Proceedings of the 49th ACM technical symposium on computer science education (pp. 344-349).
9. Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. ACM SIGCSE Bulletin, 36(4), 119-150.
10. Kirsh, D. (2010). Thinking with external representations. AI & society, 25(4), 441-454.
11. Fitzgerald, S., Simon, B., & Thomas, L. (2005, October). Strategies that students use to trace code: an analysis based in grounded theory. In Proceedings of the first international workshop on Computing education research (pp. 69-80).
12. Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. ACM SIGCSE Bulletin, 38(3), 118-122.
13. Chase, W. G., & Simon, H. A. (1973). Perception in chess. Cognitive psychology, 4(1), 55-81.
14. Katai, Z. (2014, June). Selective hiding for improved algorithmic visualization. In Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 33-38).
15. Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. ACM Transactions on Computing Education (TOCE), 13(4), 1-64.
16. Du Boulay, B. (1986). Some difficulties of learning to program. Journal of Educational Computing Research, 2(1), 57-73.
17. Fitzgerald, S., Simon, B., & Thomas, L. (2005, October). Strategies that students use to trace code: an analysis based in grounded theory. In Proceedings of the first international workshop on Computing education research (pp. 69-80).
18. Giere, R. N. (2004). How models are used to represent reality. Philosophy of science, 71(5), 742-752.

19. McCartney, R., Moström, J. E., Sanders, K., & Seppälä, O. (2004). Questions, Annotations, and Institutions: observations from a study of novice programmers. In the Fourth Finnish/Baltic Sea Conference on Computer Science Education, October 1–3, 2004 in Koli, Finland (pp. 11-19). Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science, FINLAND.
20. Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. Journal of Educational Computing Research, 2(1), 37-55.

A ProgCont rendszer jelene és jövője

Pánovics János[1], Kádek Tamás[2], Biró Piroska[3]

{panovics.janos[1], kadek.tamas[2], biro.piroska[3]}@inf.unideb.hu
Debreceni Egyetem, Informatikai Kar[1,2,3]
Sapientia Erdélyi Magyar Tudományegyetem, Csíkszeredai Kar[3]

Absztrakt. A ProgCont rendszert 2011 óta használjuk és fejlesztjük a Debreceni Egyetem Informatikai Karán. A rendszer legfontosabb szolgáltatása a programozási feladatokra beküldött forráskódok automatikus értékelése. Az első pár évben még csak programozói versenyek lebonyolítására használtuk, majd folyamatosan integráltuk az oktatásba, alkalmassá téve számonkérések lebonyolítására és gyakorló feladatsorok közzétételére. Az utóbbi pár évben felértékelődött az online oktatástámogató eszközök szerepe, és ennek megfelelően átalakult a ProgCont rendszer szolgáltatási palettája is. Cikkünkben rövid áttekintést nyújtunk ezen szolgáltatásokról, melyek egyben a későbbi fejlesztési irányt is megszabják.

Kulcsszavak: ProgCont rendszer, automatikus kiértékelés, programozás, szakkör alkalmazás

1. Bevezetés

Az elmúlt három évben az oktatási rendszer számos új kihívással találta szembe magát. A pandémia hirtelen átállást kényszerített ki a jelenléti oktatásról az online eszközök irányába. A Debreceni Egyetem Informatikai Karán a programozási nyelvek oktatásában, elsősorban számonkérésekre és versenyfeladatok kiértékelésére használt ProgCont rendszer alkalmasnak tűnt a távolléti oktatás kihívásaira való gyors reagálásra [1], [2], [3].

A rendszer előnye, hogy a Bíró és Mester szoftverekhez hasonlóan [4] a különböző forrásnyelveken beküldött feladatok megoldását automatikusan online értékeli. Így a laborgyakorlatokon kiválthatja a szemináriumvezető munkájának egy részét azzal, hogy ítéletet mond az elkészített feladatmegoldások felett.

Természetesen ez nem tudja helyettesíteni a hibákat tartalmazó megoldások problémáinak felderítésében játszott oktatói szerepet. A rendszer jellegéből adódóan a hibás megoldások értékelése igencsak szűkszavú volt, a visszautasításának okát mindössze az alábbi kategóriákra bontottuk:

- **Szintaktikai hiba**: azt jelzi, hogy a feltöltött forrásszöveg szintaktikailag hibás. Tekintettel arra, hogy a benyújtott megoldás ebben az esetben nem hajtható végre, a fordítóprogram hibaüzenetén kívül más információ nem áll rendelkezésre (1. táblázat).
- **Szemantikai hiba**: azt jelzi, hogy hogy a program hibaüzenettel megállt (E-Run), nem fejeződött be időben (E-Tme), vagy nem a megfelelő kimenetet produkálta (E-Res, E-Pre). A beküldött megoldást tipikusan több tesztesettel teszteljük, így a visszajelzés futtatásról futtatásra változhat (2. táblázat).

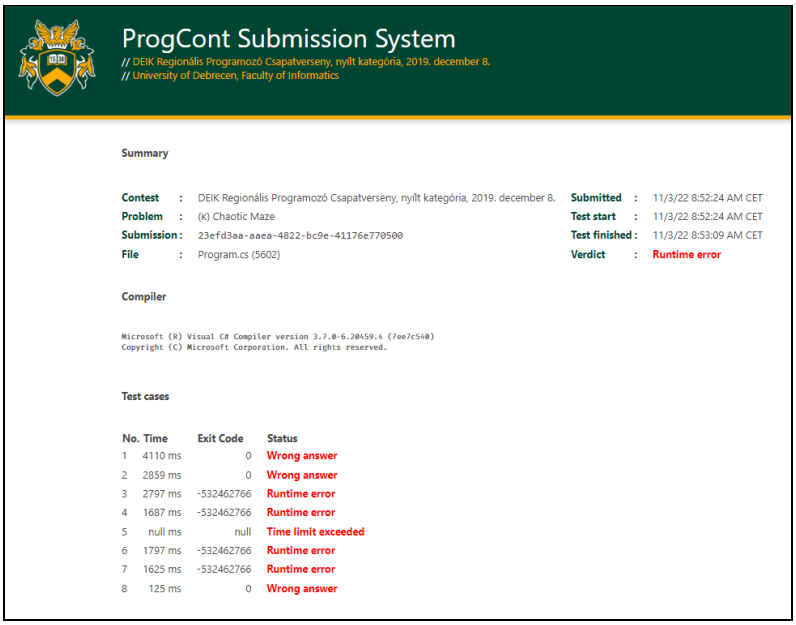
Visszajelzés	Magyarázat
E-Cmp	Fordítási hiba: A beadvány szintaktikailag hibás. Nem tudjuk lefordítani és lefuttatni a benyújtott programot, így nem tudjuk kiértékelni a teszteseteket rajta.

1. táblázat: Szintaktikai hibák

Visszajelzés	Magyarázat
E-Run	Futtatási hiba: A program végrehajtása meghiúsult, pl. hibaüzenettel megszakadt.
E-Tme	Időkorlát-túllépés: A program végrehajtása a megadott időkorláton belül nem fejeződött be, ezért a futása megszakításra került.
E-Res	Rossz válasz: A program futtatásának kimenete eltér az elvárt eredménytől.
E-Pre	Prezentációs (formátum-) hiba: A program futtatásának kimenete eltér az elvárt eredménytől, de az eltérés csak a szóköz karakterekből adódik.

2. táblázat: Szemantikai hibák

Annak érdekében, hogy a visszajelzések minél több információt nyújtsanak a hibakeresés során, kihasználva a rendszer azon tulajdonságát, hogy egy-egy megoldás ellenőrzéseséhez több tesztesetet is lefuttat, kézenfekvőnek tűnt az értékelést tesztesetenkénti bontásban is megadni.

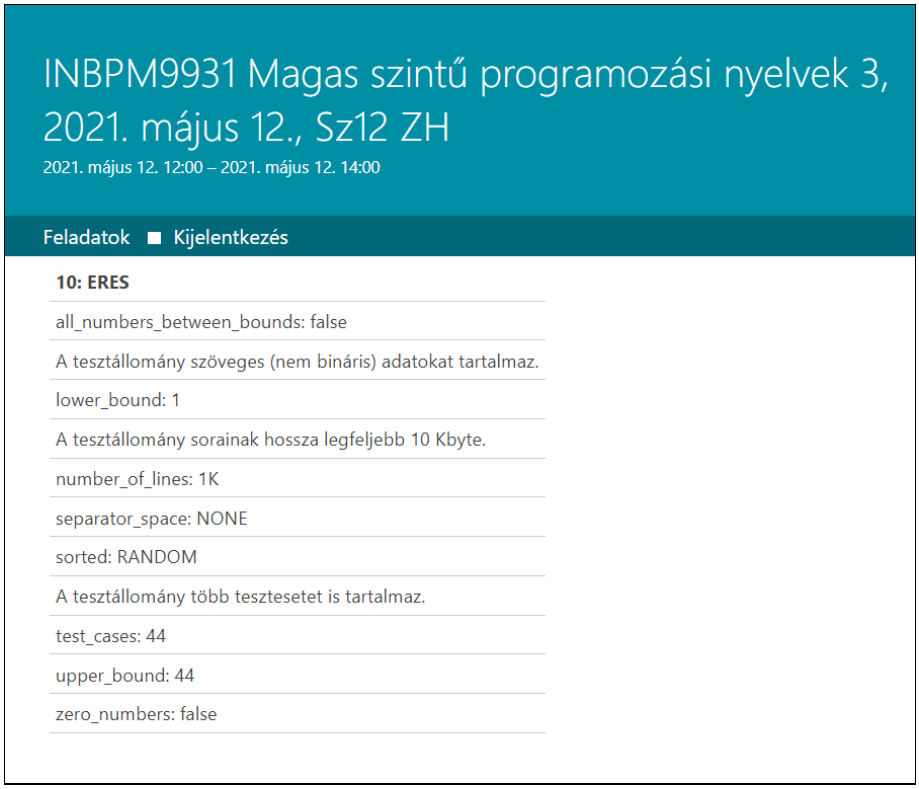


1. ábra: Megoldás értékelése tesztesetenként

Az 1. ábrán egy olyan beküldött megoldás látható, mely mind a nyolc teszteseten elbukott, három alkalommal hibás választ adott, négyszer futási hibával állt meg, és egy teszteset esetében túllépte a megengedett időkeretet és ezért leállításra került. A ProgCont rendszer ez esetben a futási hiba (E-Run) értékelést adja (mert ez a legmagasabb prioritású hiba). A tesztesetenkénti eredményből azonban már az is látszik, hogy még azon esetekben is, amikor sikerült lefutnia a programnak, hibás eredményt produkál.

Természetesen az, hogy milyen jellegű hibák fordultak elő, a teszteset tartalmának ismerete nélkül továbbra sem segíti hatékonyan a hiba felderítését. A tesztesetek nyilvánosságra hozatala ugyanakkor kerülendő, mert ennek eredményeképp olyan programok születhetnének, amelyek csak és kizárólag a tesztesetekre adnak helyes megoldást.

Ezen ellentmondás feloldásaként vezettük be a teszteset-annotációs módszert, amely az egyes tesztesetek tartalmi különbségeit igyekszik kiemelni a konkrét teszteset felfedése nélkül [5], [6], [7].



2. ábra: Megoldás értékelése annotációkkal

A 2. ábrán egy beküldött megoldás értékelésének részlete látható, mely szerint a 10-es számú teszteseten hibás kimenetet állított elő az alkalmazás. A teszteset tartalma ugyan továbbra sem elérhető, az azonban látszik, hogy milyen tesztadatok kerültek a 10-es számú tesztesetbe. Összehasonlítva a többi tesztesettel, így már sok esetben beazonosítható, hogy melyik részfeladat megoldása okozta a kihívást.

2. Teszteset-annotációk

A probléma érzékeltetésére nézzük meg a következő feladatot, mely a *Magas szintű programozási nyelvek 1* vizsgához tartozó feladatsor tagja volt, majd a vizsga után gyakorló feladatként jelent meg a ProgCont rendszerben.

FELADAT — Időpontok[1]

Írjon programot, amely a standard bemenetről állományvégjelig (EOF) „HH.MM" alakú, 24 órás intervallumon értelmezett, érvényes időpont adatokat olvas! A program írja a standard kimenetre az időpont 12 órás intervallumon értelmezett angol megfelelőjét úgy, hogy a 0–12 óra közötti időpontokat „am" utótaggal, a 12–24 óra közöttieket pedig „pm" utótaggal látja el, arra is ügyelve, hogy az angol időpontok között nem szerepelhet 0 órás időpont. A 10-nél kisebb órákat egy számjeggyel, a perceket viszont mindig két számjeggyel kell megadni. Például: 0.02-ből 12.02am, 11.58-ból 11.58am, 12.32-ből 12.32pm, 13.29-ből 1.29pm, 22.17-ből 10.17pm lesz.

2.1. Értékelés annotációk nélkül

Erre a feladatra már több mint 1300 beküldés érkezett. Kezdetben két teszteset volt a feladathoz. Az egyik egy rövid minta volt, amely a feladat leírásában is szerepel. A második teszteset az összes lehetséges bemenetet tartalmazta, összesen 1 440 sorból állt, amelyek mindegyike egy-egy feladatot jelentett. Az idők rendezetlenül jelentek meg a tesztfájlban.

Hasonló problémákat már korábban is sokféleképpen elemeztünk. A beküldött megoldások 30%-a oldotta meg helyesen a feladatot és 13%-a a két teszteset közül csak az egyikben működött helyesen. Mivel a második teszteset tartalmazta az összes lehetséges bemenetet, nem nehéz kitalálni, hogy ezek a programok ezen a második teszteseten buktak meg.

2.1. Értékelés annotációkkal

Az eredeti visszajelzésekből a hiba okára vonatkozó következtetéseket nem lehet levonni, ugyanakkor a beküldött megoldások bizonyos részfeladatokat (bizonyos időintervallumokba eső értékek konverzióját) helyesen oldották meg. Ezért 9 új tesztesettel bővítettük a feladatot, melyeket az alábbi annotációkkal láttunk el.

Alapvetően kétféle annotáció különböztethető meg: olyan, amelyik nem használja ki a feladat jellegzetességeit, hanem általános problémákra hívja fel a figyelmet (feladatfüggetlen annotációk, 3. táblázat), és olyan, amely kifejezetten a feladatleírásból fakadó különféle sajátosságokra koncentrál (feladatspecifikus annotációk, 4. táblázat).

[1] https://progcont.hu/progcont/100029/?pid=200502

Annotációk	Leírás
`empty input = true`	a teszteset egy üres bemeneti fájlból áll
`input type = text`	a tesztesetben szöveges inputot kell feldolgozni
`line length = max 10`	a bemeneti sorok hossza legfeljebb 10 karakter lehet
`lines = none`	a teszteset nem tartalmaz feladatokat
`lines = more`	a teszteset egynél több feladatot tartalmaz

3. táblázat: Feladatfüggetlen annotációk

Általában a szélsőséges input adat (pl. az üres bemenet) kihívások elé állíthatja a feladatmegoldót.

Annotációk	Leírás
`hour = 0`	a teszteset csak olyan feladatokat tartalmaz, amelyekben az egyes időpontok óraértéke 0; azaz a helyes kimenet 12.MM am.
`hour = 1-11`	a teszteset csak olyan feladatokat tartalmaz, amelyekben az egyes időpontok óraértéke 1–11; azaz a helyes kimenet 1–11.MM am.
`hour = 12`	a teszteset csak olyan feladatokat tartalmaz, amelyekben az egyes időpontok óraértéke 12; azaz a helyes kimenet 12.MM pm.
`hour = 13-23`	a teszteset csak olyan feladatokat tartalmaz, amelyekben az egyes időpontok óraértéke 13–23, azaz a helyes kimenet 1–11.MM pm.
`minute = 10-59`	a teszteset csak olyan feladatokat tartalmaz, amelyekben a perc két számjegyű.
`minute = 0-9`	a teszteset csak olyan feladatokat tartalmaz, amelyekben a perc egyjegyű; ezért a helyes kimenet perc része 0-val kezdődik.

4. táblázat: Feladatspecifikus annotációk

Ezek segítségével megállapíthattuk, hogy a részben jó megoldások esetében többségében a 0 óra és az egy számjegyű perc értékek megjelenítése okozta a legnagyobb kihívást.

Ami ennél is fontosabb, hogy az annotációk segítségével a részben jó megoldást készítő diákok be tudták azonosítani azt a részfeladatot, melyet még nem teljesítettek sikeresen, és ez nagyban megkönnyítette a hibakeresést, ami így önállóan is elvégezhető volt.

3. Szakkör alkalmazás

A ProgCont rendszer fejlesztésében fontos feladat a hozzáférhetőség javítása. Nemcsak házon belüli számonkéréseken és versenyeken szeretnénk használni a szoftvert, hanem nemzetközi versenyek lebonyolításában, nyári programozó szakkörökön, és úgy általában programozási feladatok automatikus kiértékelésében akár házon kívül is [8].

A rendszerben lévő feladatok majdnem teljes egészében a külvilág számára is korlátozások nélkül elérhetők a **`progcont.hu`** weboldalon, ahol korábbi versenyeink, számonkéréseink, gyakorló feladatsoraink nagy számban állnak az érdeklődők rendelkezésére (47 versenyfeladatsor, 253 zárthelyi dolgozat, összesen 1795 feladat).

A ProgCont rendszer ezen feladatokra fogadja a megoldásokat, melyeket mindenféle regisztráció nélkül, folyamatosan bővülő programozásinyelv-kínálat mellett értékel (jelenleg C, C++, C#, Java, Pascal, Python és Racket nyelvű megoldásokat tud tesztelni).

A szakkör alkalmazás keretein belül megvalósult legfontosabb fejlesztés a ProgCont rendszerben a szoftver modularizálása volt. Ennek keretén belül alapjaiban gondoltuk újra, hogy az automatikus értékelő szolgáltatás miként alakítható át úgy, hogy számos különböző felhasználási esetben egyaránt használható legyen. Végeredményként egy rugalmas ProgCont API került kidolgozásra [9], mely egységes keretbe foglalja a különféle kiértékelési feladatokat. A szakkör alkalmazás valójában nem egyetlen alkalmazás, hanem alkalmazáskomponensek gyűjteménye (diák felület, zsűri weblap, adminisztrátori oldalak), melyeket egymáshoz kapcsolva kínálnak egyre több és egyre inkább testre szabható lehetőséget a ProgCont erősségeinek kihasználására.

4. Összefoglalás

A pandémia okozta kényszerek új lendületet adtak az online oktatástámogató eszközök fejlesztésének. Felértékelődtek az önálló felkészülést támogató szoftverek. Ugyanakkor az is látszik, hogy ahhoz, hogy a ProgCont rendszer ezen a téren is sikeres legyen, további fejlesztésekre van szükség. Az utóbbi két év tapasztalataiból kiindulva, két fő fejlesztési irány rajzolódik ki.

Egyfelől fejleszteni kell a rendszer visszajelzéseit annak érdekében, hogy a legalább részben jó megoldások esetében iránymutatást adjon a lehetséges hibák okainak felderítéséhez.

Másfelől ki kell terjeszteni a rendszer elérhetőségét, kihasználva a moduláris felépítésben rejlő azon lehetőséget, hogy a különböző felhasználói körök számára testre szabott felületeket tud kínálni.

Irodalom

1. P. Biró, T. Kádek: *Automatic evaluation of programming tasks at the University of Debrecen*, in INTED2020 Proceedings, 2–4 March, 2020, pp. 3522–3527, DOI:10.21125/inted.2020.0994.
2. P. Biró, T. Kádek: *How can the ProgCont system help with programming education during a pandemic*? In 15th International Conference on Economics and Business: Challenges in the Carpathian Basin : Global Challenges - Local Answers. Interdependencies or Slobalisation? Cluj-Napoca, Editura Risoprint (2021), pp. 179-191.
3. P. Biró, T. Kádek: *The Mathability of Computer Problem Solving with ProgCont*. Acta Polytechnica Hungarica 2022, 19(1) pp. 77-91. https://doi.org/10.12700/APH.19.1.2022.19.6
4. G. Horváth, G. Horváth, L. Zsakó: *A bíró és a mester – az online értékelés szerepe a programozás oktatásában*, Mérési és értékelési módszerek az oktatásban és a pedagógusképzésben Szerk. Károly Krisztina és Homonnay Zoltán, Diszciplínák tanítása – a tanítás diszciplínái 5, ELTE Eötvös Kiadó, Budapest, 2017, pp. 89–103.
5. P. Balázs, P. Biró, T. Kádek, M. Kósa, J. Pánovics: *Mathability and exploring mathematical skills of programmers with exercises' annotations*. In: Nikodem, Jan; Klempous, Ryszard (eds.) 12th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2021): Proceedings IEEE 2021, pp. 347-350.
6. P. Biró, M. Kósa, J. Pánovics, T. Kádek: *Test case annotation of programming tasks in the ProgCont system for differential education and strengthening stand-alone preparation opportunities*. In EDULEARN21 Proceedings 2021, (pp. 5433-5441). http://doi.org/10.21125/edulearn.2021.1109
7. P. Biró, T. Kádek, M. Kósa, J. Pánovics: *A New Method to Increase Feedback for Programming Tasks During Automatic Evaluation*. Acta Polytechnica Hungarica 2022, 19(9) pp. 103-116. https://doi.org/10.12700/APH.19.9.2022.9.6
8. T. Kádek, P. Biró: *A távolléti oktatás hatásai a ProgCont rendszerre*. In XXI. Energetika-Elektrotechnika – ENELKO és XXX. Számítástechnika és Oktatás – SzámOkt Multi-konferencia 2020, (pp. 104–109).
9. T. Kádek, P. Biró: *A ProgCont API: programozási feladatok megoldásainak újszerű kiértékelése*. In ENELKO 2019 SzámOkt 2019 (pp. 191–195).

BitHÓDítás interaktív megvalósítás tapasztalatai

Pluhár Zsuzsa

pluharzs@inf.elte.hu
ELTE IK

Absztrakt. A nemzetközi Bebras kezdeményezés magyar megvalósulása már több, mint tíz éve HÓDít az oktatásban. Nem csupán a verseny, de a kiterjesztett számítógép nélküli aktivitások és az algoritmikus gondolkodás mérésére, fejlesztésére használt, kiemelt feladatok felhasználásai, megjelenései is. A 2021-es évben a kezdeményezés alapját jelentő verseny egy új környezetbe került, mely a 2022-es évben interaktív feladatokkal is kiegészülhetett. Jelen tanulmány ennek a két évnek az eredményeit és tapasztalatait foglalja össze, vizsgálva az új környezet által nyújtott lehetőségeket és módosításokat.

Kulcsszavak: informatikai gondolkodás, verseny, hód, motiváció, Bebras

1. Informatikai gondolkodás és a Hód

Az elmúlt években az informatikai gondolkodás (computational thinking, CT), illetve elemei minden polgár alapvető készségeként, képességeként jelenik meg [1,2,3]. Sok kutató különböző módon definiálja az informatikai gondolkodást és a benne foglalt készségeket, összetevőket és főbb fogalmakat. Többen újabb és újabb megközelítéseket alakítanak ki, valamint tanulási tartalmat és értékelési eszközöket hoznak létre.

Az egyik erőteljes irányzat azokat a kezdeményezéseket foglalja magába, melyek számítógép nélküli aktivitásokkal, előismeretek nélküli elvárásokkal valósítják meg az informatikai gondolkodás fejlesztését, vizsgálatát. Ilyen kezdeményezések pl. a legkisebbeknek szóló Computer Science Unplugged [4], az idősebbeket megszólító Computer Science For Fun [5] vagy a legszélesebb életkori és dimenziókat összefoglaló Bebras [6, 7].

A nemzetközi Bebras kezdeményezés az informatikát és az informatikai gondolkodási folyamatokat népszerűsíti rövid feladatok megoldásával. Számos országban formális és informális iskolai rendezvényként is megjelenik [8,9,10]. A kezdeményezés sarkalatos pontja a megjelenő, nemzetközi csapat által elkészített feladatok, melyek az informatikai koncepciókra épülnek, a technológián túli ismeretek megértését segítik elő, rövidek, motiválóak, és néhány perc alatt megválaszolhatóak.

A magyar megvalósulás 2010 óta követhető nyomon az alábbi célkitűzésekkel [10]:

- az érdeklődés felkeltése és fenntartása az informatika iránt;
- az informatikával kapcsolatos félelmek, negatív érzések feloldása;
- az informatika sokszínűségének, felhasználási lehetőségeinek és területeinek megmutatása.

Mindezt úgy, hogy informatikai előképzettséget nem igényel.

Célcsoportként nem csupán a diákokat nevezhetjük meg, de ugyanilyen fontosak számunkra a közoktatásban résztvevő, illetve oda hamarosan bekerülő tanárok is. Az ő számukra célunk támogatást adni az egyes informatikai témakörök motiváló probléma-felvetéseihez, példa alapú megoldásaihoz, valamint az informatika sokszínűségének és integrálásának lehetőségeihez.

A kezdeményezés alapját jelentő verseny mellett számos kutatási és fejlesztési projekt valósul meg: számítógéptől elszakított tevékenységek, mérések és fejlesztési koncepciók akár egyetemi előkészítő csoportokban is [10,11,12].

2. BitHÓDítás új környezetben

2.1. ViLLE

2021-ben a magyar kutatócsoport csatlakozott egy digitális tanulási környezethez, amelyet a finn Turku Egyetem Tanuláselemzési Központjában fejlesztettek ki [13,14,15]. A ViLLE automatikus értékelést végez, és azonnali visszajelzést ad a diákoknak, valamint átfogó tanulási elemzést a tanárnak.

A ViLLE rendszer támogatja az anonim fiókokat, rengeteg feladattípust, az automatikus értékelést, az időkorlátokat, a hallgatók teljesítményének nyomon követését és még sok mást.

A Bebras kezdeményezés, és a HÓDítsd meg a biteket verseny megvalósításához a keretrendszer magában foglalja a feladatok létrehozását, a korosztályonkénti kurzusokba szervezését és a Kutatási Rendszerbe való bekapcsolását.

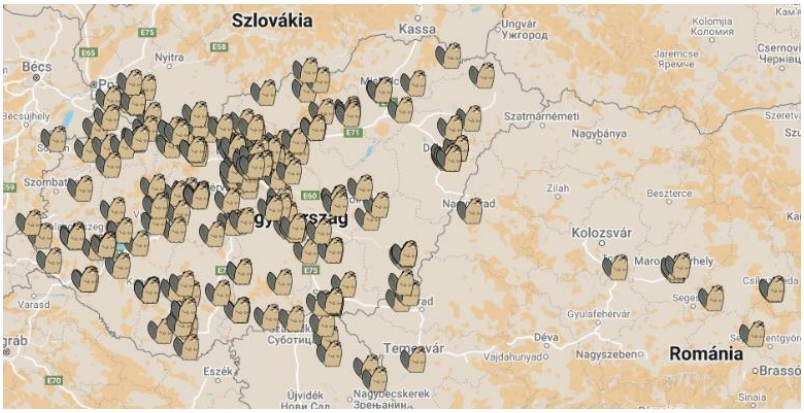
2.2. e-hód 2021

Egy 40 iskola (1351 diák) részvételével megszervezett októberi pilot tesztet követően 2021 novemberében a ViLLE rendszerben szerveztük meg a versenyt [16].

2.2.1. A verseny

A versenyt a korábbi évekhez hasonlóan folytattuk le: a feladatok nem voltak interaktívak; a tanulók négy lehetőség közül választhattak egy választ. A feladatok sorrendje azonos volt (könnyű, közepes és nehéz), a válaszlehetőségek sorrendje pedig véletlenszerű.

Több mint 30 ezer diák (N=33467, 305 iskolából) vett részt. A legtöbb iskola a fővárosból és a nyugati régiókból (lásd 1. ábra).



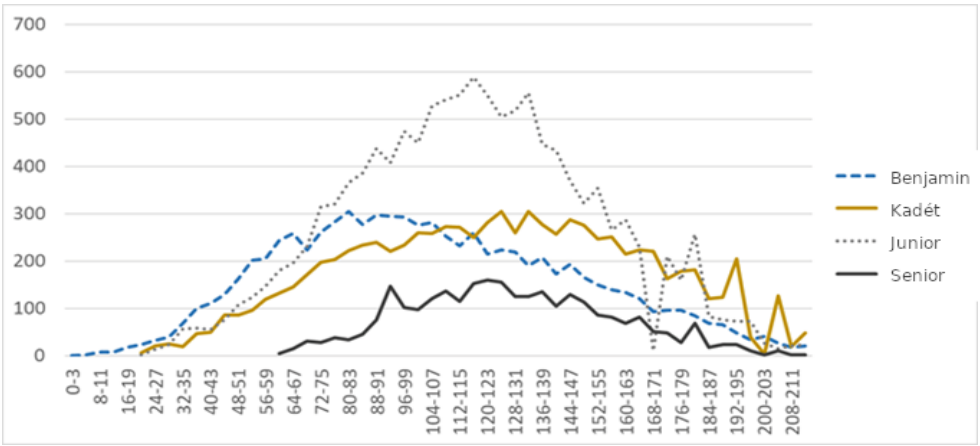
1. ábra: A 2021-es e-Hód verseny résztvevő iskolái térképen.

A legnagyobb létszámnövekedést a kishód korosztályában (9-10 évesek) tapasztalhattuk, 36%-os növekedéssel 2020-hoz képest. A legtöbb résztvevő (12491 diák) a junior korosztályban (15-16 évesek) volt.

A legfiatalabbaknak ezúttal 12 (nehézségi szintenként négy) feladatuk volt, és összesen 144 pontot érhettek el. A másik négy korcsoportban szokás szerint 18 (nehézségi szintenként hat) feladat megoldásával összesen 216 pontot érhettek el.

2.2.2. Eredmények

A feladatok nehézségét tekintve az elsődleges elemzések azt mutatták, hogy a szokásosnak megfelelően sikerült a besorolás – a diákok a vártnak megfelelően teljesítettek a feladatok megoldása során. A 2. ábra az egyes csoportok tanulóinak számát mutatja korcsoportonként az adott összpontszámokat tekintve (kivéve a kishódokat az alacsonyabb összpontszám miatt).



2. ábra: 2021-es bitHÓDítás résztvevőinek pontszám eloszlása.

Az egyes korcsoportok pontszámainak szórására vonatkozóan nem találtunk szignifikáns különbséget a lányok és a fiúk pontszámai között a két legfiatalabb korcsoportban. A feladatokban eltöltött idő viszont szignifikánsan magasabb volt a lányok esetében.

A három legidősebb csoport pontszáma szignifikáns különbséget mutatott a fiúk és a lányok között. Itt viszont nem volt szignifikáns különbség a fiúk és a lányok feladatokban eltöltött ideje között.

A varianciaanalízis a 2021-ben elért pontszámokra és a feladatokban eltöltött időre vonatkozóan szignifikáns eltérést eredményezett az egyes korcsoportok között. A legidősebb korosztály tanulói szignifikánsan alacsonyabb pontszámot értek el átlagosan, és ők töltötték a leghosszabb időt feladatokkal. A kadétok tanulói lényegesen kevesebb időt töltöttek el az egyes feladatok megoldásával, és szignifikánsan magasabb átlagpontszámot értek el. A pontos elemzések, tesztek eredményei a [16]-ban olvashatóak.

A versenyt követően egy kérdőívet küldtünk ki a tanároknak, mellyel elsősorban a rendszer használatával kapcsolatos tapasztalataikra, meglátásaikra voltunk kíváncsiak.

Általánosan elmondható, hogy a rendszert mind a tanárok, mind a diákok megbízhatóbbnak és gyorsabbnak értékelték. Az egyetlen nehézséget a regisztráció csoportonkénti szétválasztása jelentette a több, mint 10 csoportot koordináló tanároknak.

2.3. e-hód 2022

2.3.1. A verseny

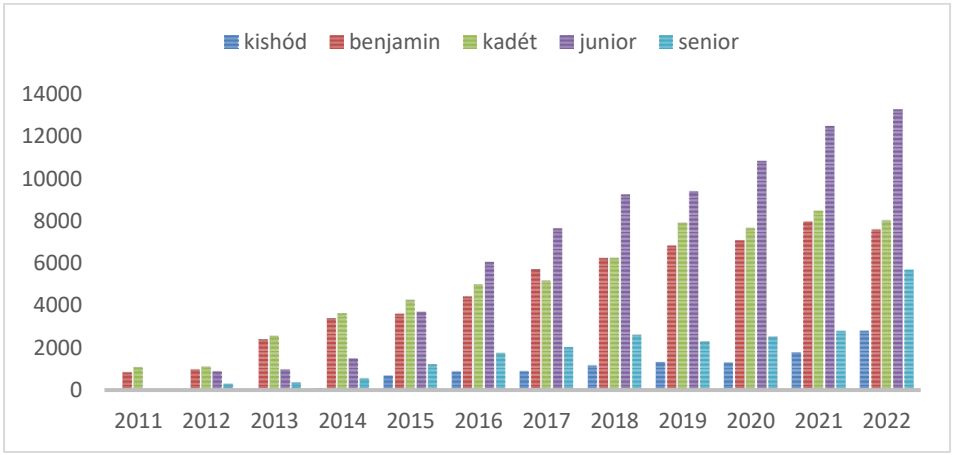
A 2021-es verseny tapasztalatai és visszajelzései alapján a ViLLE környezet regisztrációs folyamatait igyekeztük könnyebbé és zökkenőmentesebbé tenni. Köszönet ezért a finn fejlesztőknek.

A feladatokban annyi változás történt, hogy 2022-ben először a versenyek történetében interaktív feladatokkal is találkozhattak a versenyzők.

Az interaktív feladatok között nem csupán a megfelelő helyre való kattintás, de az adott színnel való kitöltés, a drag&drop aktivitások, és párosítás is szerepet kapott.

2.3.2. Eredmények

A 2022-es versenyen 283 iskola 37351 diákja vett részt. Ez összességében kicsit magasabb, mint 11%-os növekedést jelentett az előző évhez képest.

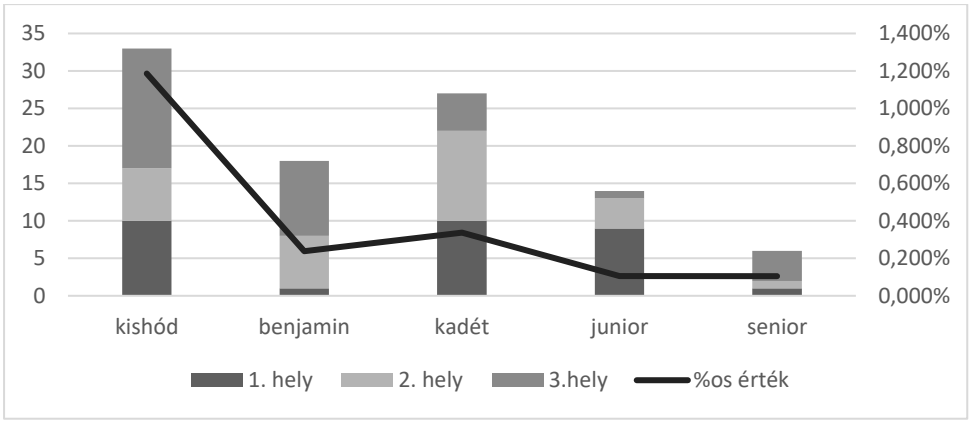


3. ábra: A bitHÓDítás résztvevőinek életkori eloszlása 2011-2022 között.

Arányaiban a legnagyobb növekedés a senior kategóriában történt, ahol a résztvevők száma megduplázódott. Ez a növekedés a „felnővő" generációnak is köszönhető, azaz a korábbi évek fiatalabb résztvevői átlépnek az idősebb korosztályba.

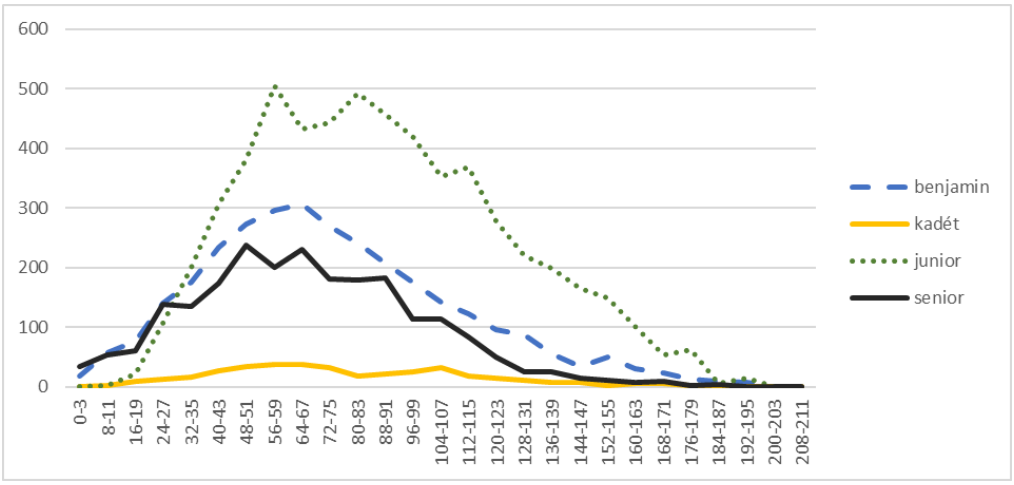
Az iskolák átlagosan 134 diákot neveztek, a legmagasabb létszám 742 főt jelentett.

Az 1-3. helyezettek száma az előző évekhez hasonlóan alakult, a kishód kategóriában, a „csalogató" feladatokkal lett a legmagasabb, míg a legidősebb kategóriában a legalacsonyabb. Arányaiban – a résztvevők számát tekintve a kishódok esetében több, mint 1% az 1-3. helyezettek száma, de a többi kategóriában kiegyensúlyozott: a kadét (0,337%) és a benjamin (0,237%) valamivel magasabb, mint a két legidősebb életkori kategóriában (0,106%).



4. ábra: A 2022-es év 1-3. helyezettjeinek száma és aránya életkori kategóriánként.

Az átlagpontszámokat tekintve a 2021-es évhez hasonlóan alakultak a görbék némi eltolódást mutatva az alacsonyabb pontszámok irányába (5. ábra).



5. ábra: 2022-es bitHÓDítás résztvevőinek pontszám eloszlása.

A feladatok interaktivitásával könnyebbnek várhattuk a teljesítést a gondolkodási folyamatokat tekintve. A feladatok megoldási módja, hogy nem tippelhető a megoldás a megadott válaszlehetőségek közül, a tipikusan rossz válasz nem zárható ki – azonban pont azt mutatja, hogy a feladatok megoldása jobban tükrözheti a diákok gondolkodási erősségét.

3. Konklúzió

A feladatok további elemzése mindenképpen szükséges. Következő tervezett vizsgálataink célja annak megállapítása, hogy az egyes feladatok nehézségét, megoldhatóságát mennyire befolyásolta az új feladattípusok, az interaktivitás megjelenése.

4. Köszönetnyilvánítás

A verseny és a hód kiterjesztések nem tudnának ilyen sikeresen helyt állni az ELTE IK tanárszakos hallgatóinak közreműködése nélkül. További köszönet illeti azokat a lelkes pedagógusokat, akik diákjaik részvételét lehetővé teszik, valamint a nemzetközi Bebras csapatot, akik évről évre töretlenül munkálkodnak azon, hogy érdekesebbnél érdekesebb feladványok kerülhessenek terítékre.

Irodalom

1. Wing, J. M. (2006): Computational thinking. In: Communications of the ACM, 49(3), 33-35.
2. Arfé, B., Vardanega, T., & Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. Computers and Education, 148, 1–16. https://doi.org/10.1016/j.compedu.2020.103807
3. Palts, T., & Pedaste, M. (2020). A Model for Developing Computational Thinking Skills. Informatics in Education, 19, 113–128, doi:10.15388/INFEDU.2020.06.
4. Bell, T., Witten, I. H., Fellows, M. (2010): Computer Science Unplugged. http://csunplugged.org/books (utoljára megtekintve: 2016. 10. 25.)
5. Computer Science For Fun http://www.cs4fn.org/ (utoljára megtekintve: 2022. 11. 19.)

6. Dagiene, V. (2006): Information technology contests – introduction to computer science in an attractive way, Informatics in Education, 5. 1.s., 37–46.
7. Cartelli, A., Dagiene, A., Futschek, G. (2010): Bebras Contest and Digital Competence Assessment: Analysis of Frameworks. International Journal of Digital Literacy and Digital Competence. Január-Március. 24-39.
8. Dagienė, V., & Stupurienė, G. (2016). Bebras - a sustainable community building model for the concept based learning of informatics and computational thinking. Informatics in education, 15(1), 25-44.
9. Pluhár, Zs., Gellér, B.: International Informatic Challenge in Hungary. In: Teaching and Learning in a Digital World: Proceedings of the 20th International Conference on Interactive Collaborative Learning. Berlin, Germany: Springer, (2018). pp. 425-435
10. Pluhár, Zs. (2020) Informatikai gondolkodás fejlesztésének dimenziói, In: Zsakó, László; Szlávi, Péter (szerk.) InfoDidact 2020, Budapest, Magyarország : Webdidaktika Alapítvány (2020) Cikk: 14
11. Pluhár, Zs., Torma, H., Törley, G. (2018) Hallgatói teljesítményértékelés az algoritmikus gondolkodás tükrében, In: Szlávi, Péter; Zsakó, László (szerk.) InfoDidact 2018, Budapest, Magyarország : Webdidaktika Alapítvány (2018)
12. Pluhár, Zs. Torma, H. (2019) Introduction to Computational Thinking for university students, In: Pozdniakov, Sergei N.; Dagienė, Valentina (szerk.) Informatics in Schools. New Ideas in School Informatics Cham, Svájc : Springer International Publishing, pp. 200-209.
13. Laakso, M.-J., Kaila E., & Rajala T. (2018). ViLLE - Collaborative Education Tool: De-signing and Utilizing an Exercise-Based Learning Environment. Education and Information Technologies, 01/2018
14. 27. Kurvinen, E., Dagienė, V., & Laakso, M.-J. (2018). The impact and effectiveness of tech-nology enhanced mathematics learning. Constructionism 2018: Constructionism, computational thinking and educational innovation: conference proceedings (eds. V. Dagienė, E. Jasutė). Vilnius University, 351-363.
15. 28. Kurvinen, E., Kaila, E., Laakso, M., & Salakoski, T. (2020). Long Term Effects on Tech-nology Enhanced Learning: The Use of Weekly Digital Lessons in Mathematics. Informatics in Education, 19(1), 51-75. doi:10.15388/infedu.2020.04
16. Pluhár, Zs., Kaarto, H., Parviainen, M., Garcha, S., Shah, V., Dagienė, V., Laakso, M. (2022) Bebras Challenge in a Learning Analytics Enriched Environment: Hungarian and Indian Cases Lecture Notes in Computer Science 13488 pp. 40-53.

Az online oktatás „tízparancsolata"

Pšenáková Ildikó[1], Pšenák Peter[2]

[1]ildiko.psenakova@truni.sk, [2]psenak9@uniba.sk
Trnava University in Trnava, Comenius University in Bratislava

Absztrakt. Még ha a tanulmány címében, lehet kicsit viccesen, parancsolatoknak nevezzük, de inkább jótanácsoknak szántuk, melyekkel segíteni szeretnénk elsősorban a leendő pedagógusoknak, de a jelenleg már gyakorló pedagógusok is találhatnak benne hasznos gondolatokat. A „parancsokat" a járványhelyzet alatti táv- illetve online oktatás alatt szerzet saját tapasztalataink alapján szedtünk össze. Ezeket szeretnénk megosztani olvasóinkkal, remélve azt, hogy segítségükre lesznek, ha hasonló helyzetekben lesznek kénytelenek oktatni.

Kulcsszavak: távoktatás, online tanítás, tapasztalatok, jótanácsok

1. Bevezetés

A távoktatás és az online tanulás a koronavírus okozta pandémia előtt szinte csak a felsőoktatásban volt jelen, ahol a diákok többsége tanulni akartak és nem azért tanult, mert iskolába járni és tanulni kötelező. A kialakult járványhelyzet azonban számos változást idézett elő az iparban, a mezőgazdaságban, a kultúrában és az oktatásban is. A vírus terjedésének megelőzése vagy lassítása érdekében bezárták az éttermeket, gyárakat, üzleteket és be kellett zárni az iskolák kapuit is. Az általános és középiskoláknak is át kellett állniuk a távoktatásra, ami hatalmas és szokatlan változást jelentett.

A gyakorló pedagógusok, különösen az alsó tagozatos oktatásban, nem voltak kellőképpen felkészülve a távoktatásra, illetve az online tanításra, ezért a járvány okozta helyzet meglepte őket. Nem kizárt, hogy ilyen vagy hasonló, esetleg még rosszabb helyzet (háború) ismét kialakul, ezért szükséges, hogy a jelenlegi és leendő pedagógus hallgatókat erre felkészítsük. Ha a leendő pedagógusok megismerkednek a távoktatási formákkal, felkészülnek az online térben végzett munkára, szinte biztosak lehetünk benne, hogy szükség esetén megbirkóznak a távoktatás kihívásaival.

2. A távoktatás néhány formája

Jelenleg a távoktatás megvalósítására számos technikai eszköz (személyi számítógép, táblagép, mikrofon, mobiltelefon, interaktív tábla, projektor stb.) és az információs és kommunikációs technológiák különféle formái (internet, WiFi, mobilhálózatok – 3G, 4G, 5G stb.) állnak rendelkezésre. Ezek szövegek, képek és hangok segítségével is biztosítják a kapcsolatot a tanár és a diákok között. A pedagógusok a tanítás során biztosítják a diákok számára a tananyagokat, amelyek különböző weboldalakon érhetnek el, de a diákok önállóan is kereshetnek webes adatokat és kiegészíthetik ismereteiket.

A kommunikációs eszközök és lehetőségek sokrétűsége okozza, hogy a távoktatásban különböző megközelítések és eljárások kerülnek alkalmazásra, amelyeket a pedagógusok a számukra legmegfelelőbb, illetve a rendelkezésre álló eszközök lehetőségei alapján választanak ki. Ez az oka annak, hogy a gyakorlatban a távoktatás különféle formái alakultak ki, amelyeket gyakran kombinálnak az oktatás során, és néha még a tanár sem veszi észre, hogy a távoktatás különböző formáit használja.

A gyakorlatban jelenleg használatos távoktatási formák ábrázolására az 1. ábrát hoztuk létre. Ezzel arra akartunk utalni, hogy az egyes formák egymáshoz kapcsolódnak, bizonyos jellemzőikben

kiegészítik egymást, és bizonyos gyakorlatokban átfedik egymást, vannak közös és eltérő jellemzőik, mindegyik valamiben a legjobb, valamiben rosszabb, ezért nehéz hierarchiát kialakítani közöttük. Lényegében összekapcsolják, és a végén egy egészet alkotnak – a távoktatást.

1. ábra: A távoktatás formái

Nem szeretnénk a tanulmányban aprólékosabban foglalkozni az összes típussal, de azért említettük meg őket, hogy éreztessük a távoktatás különböző létező formáit, melyeket a leendő pedagógusoknak jó lenne ismerni, hogy szükség esetén ki tudják választani a nekik megfelelőt. Ha a leendő pedagógusok tantervébe sikerül beiktatni olyan tantárgyat, mely a távoktatással, annak formáival, lehetőségeivel, előnyeivel, hátrányaival foglalkozik, és megtanítjuk őket mikor és hol lehet, illetve szükséges használni, akkor további tudást és képességet adunk a tarsolyukba. Ezzel lehetőséget is kapnak arra, hogy a távoktatás formáit nem csak járványhelyzetben, de más helyzetekben is tudják használni.

3. Online oktatás

A távoktatás egyik leggyorsabban fejlődő módszere az online oktatás. Gyorsabban terjed azokban az országokban (Amerikai Egyesült Államok, Szingapúr, Japán, Kanada, Ausztrália, Új-Zéland, Európa nagy része), ahol a nagy sebességű szélessávú internet-hozzáférés dominál, az ország szinte minden iskolája és nagyszámú háztartása kapcsolódik hozzá, és a lakosság nagy százaléka rendelkezik a szükséges képességekkel. [1]

Az online oktatás előnye abban rejlik, hogy képes:

- többcsatornás oktatás biztosítására, beleértve a nyomtatott tartalmat, hangot, képet és videót,
- több formátum biztosítására a szöveges, hang- és video kommunikációhoz valós időben,
- együttműködés biztosítására a kollégákkal a világ minden tájáról,
- a „bárhol" tanulás biztosítására, feltéve, hogy a diákok hozzáférnek az internethez.

Az online oktatás legegyszerűbb jellemzése lehet az lenne, hogy egy olyan távoktatási formáról van szó, amelyben a résztvevők térben különböző helyeken tartózkodnak, de a tanítás egyszerre történik, olyan eszközökön keresztül, amelyek lehetővé teszik a résztvevők kép- és hangkapcsolatát.

Az online oktatás részhalmazának tekinthetjük például a:

- *szinkron tanulást* – amikor minden diák egy időben és gyakran ugyanazon a helyen tanul együtt, de a tanár egy másik helyen van. A tanár és a diákok digitális összekötésére a videokonferencia vagy a telekonferencia szolgál.
- *hibrid tanulást* – vegyes oktatási rendszer, amelyben néhány diák személyesen (alapvetően offline) van jelen, míg más diákok online kapcsolódnak ugyanahhoz az órához.
- meghatározott időpontban tartott *online kurzusokat* – a diákok előre megadott időpontban egy adott virtuális helyet látogathatnak meg. Ez lehetővé teszi a diákok számára, hogy a világ bármely pontjáról kapcsolódjanak, tanuljanak és kommunikáljanak egy oktatóval vagy egymással.

  Az online oktatás azonban rejtett pénzügyi költségekkel is jár a diákok számára. Ez elsősorban azokat érinti, akik addig az iskola által biztosított technikai eszközöket használták. Ha online órákon szeretnének részt venni, akkor megfelelő technikai eszközökre van szükségük, amelyek biztosítására anyagi forrásokat kell fordítani. Ezek a költségek például magukban foglalják:

- a személyi számítógép vásárlását,
- az internetkapcsolat biztosítását,
- webkamera, fejhallgató és mikrofon vásárlását, ha azok nem részei a számítógépnek,
- a közszolgáltatások költségeit (pl. villany, internet szolgáltatás díja...).

Sajnos, Szlovákiában is még vannak olyan lakossági csoportok, amelyek nem engedhetik meg maguknak ezt a kiadást, így nem vehetnek részt online órákon vagy más távoktatási formákon.

Az online oktatással kapcsolatos problémákat az alacsonyabb hálózati kapcsolati sebesség is okozhat. A gyenge (lassú) internetkapcsolat megnehezítheti a videók vagy a valós idejű konferenciák követését.

Az online oktatás iránti igény azonban világszerte folyamatosan nő. Ennek oka az új technológiák kölcsönhatása, az internethasználat globális növekedése, valamint az internetet és az információs és kommunikációs technológiákat kezelni képes, képzett munkaerő iránti növekvő kereslet. A szakértők még 2018-ban úgy becsülték, hogy az online oktatás jó úton halad, hogy 2025-re az oktatás fő módszerévé váljon. Akkor még nem tudhatták, hogy becslésüket a világjárvány miatt erősen túlbecsülték. [2]

4 „Tízparancsolat” vagy „jó tanácsok”

A Nagyszombati Egyetemen és a Comenius Egyetemen az oktatás négy szemeszteren keresztül online formában folyt. Ezalatt az idő alatt a szerzők sok tapasztalatot szereztek és kutatói munkát is végeztek a távoktatás és a hagyományos oktatás eredményeinek összehasonlítására. A gyakorlati tapasztalatok alapján és a folytatott munka tartalmi és kutatási eredményeiből érdekes következtetések születtek, amelyek alapján több ajánlást készítettünk a pedagógiai gyakorlat számára. Ezek segíthetik a leendő pedagógusokat a távoktatási forma elsajátításában, de a jelenleg gyakorló pedagógusok is meríthetnek belőlük ihletet, ötletet. Ezeket a következő pontokban foglaltuk össze:

1. Ne használj egyszerre több különböző, online tanulást támogató környezetet!

Indokok: Ha a tanár egyszerre több felületet használ (MS Teams, Skype, Facebook, Whatsapp...), akkor a diáknak váltogatnia kell a környezetek között, amivel időt veszít. Előfordulhat az is, hogy megfeledkezik valamelyik platformról és elmulasztja a fontos információkat. A tanárnak is egyszerűbb az órák adminisztrálása egyetlen programfelületről.

2. Ha lehetséges, rögzítsd a tanórákat!

Indokok: Ha egy előadásról vagy az egész óráról videófelvétel készül, akkor azt közzé lehet tenni, és azok a diákok, akik nem tudtak részt venni az online órán, lejátszhatják azt offline módban, amikor

nekik kényelmes. A tanárnak is előnyös, mert nem kell megismételnie az előadást egy másik diákcsoportnak, nekik el tudja indítani, miközben biztos lehet benne, hogy minden csoportban ugyanazt fogja elmondani, és nem felejt el semmit.

3. Készítsd az online előadást úgy, hogy rögzíthesd, majd terjeszthesd!

Indokok: Az indokok hasonlóak, mint a 2. pontban. Az online óra elkészítésekor célszerű prezentációkat (Power Point, Canva, Google Slides) használni, amelyek növelik a tanítás áttekinthetőségét és a felvételek tartalmának érthetőségét is.

4. Tartsd be az elektronikus tananyagok megfelelő helyes elkészítésének alapelveit!

Indokok: A távoktatás nem valósítható meg megfelelő elektronikus tananyagok használata nélkül. Csak a megfelelően elkészített tananyag teljesíti a pedagógus által neki szánt célját. A tananyagban optimális egyensúlyt kell kialakítani a vizuális észlelés és a grafikus és szöveges információk között. A vizuális támogatás nélküli sűrű szöveges dokumentumok olvashatatlanná válnak. A formák, színek és kontrasztok vizuális hatása nélkül a képernyőn megjelenített információ unalmas és motiválatlan. A diákok biztosan nem néznek meg sokszor olyan rossz minőségű videófelvételt, amelyen eltűnik a hang, vagy ugrál a kép. [3]

5. Használd ki az aszinkron tanítás előnyeit!

Indokok: Sok pedagógus nem vette/veszi észre az online tér adta lehetőségeket. Ez elsősorban a tanítási idők és a diákok tanulási idejének különbsége. Ez a különbség különösen az egyetemeken jól használható, ahol a tanár úgy alakíthatja a munkarendjét, hogy jobban megfeleljen neki és a diákoknak is.

6. Az oktatás időpontjának/idejének tervezéséhez használd az online naptárat!

Indokok: A naptár funkciót szinte minden online oktatást támogató környezet biztosítja. A naptár a csoport (osztály) minden tagja számára látható. A naptárban lehetőség van találkozási időpontok, értesítések, feltételek, változások, konzultációs időpontok stb. beállítására.

7. Hozz létre áttekinthető mappastruktúrát a fájlok, dokumentumok és felvételek tárolásához!

Indokok: A mappák áttekintést biztosítanak az állományok között. A tanárnak egyszerűbb figyelni arra, hogy egyes dokumentumok, tananyagok ne kerüljenek tárolásra többször. Egy áttekinthető struktúrában a diákoknak is könnyebb orientálódni, mert pontosan tudják mit és hol kell keresni.

8. Használj videó hívásokat!

Indokok: Jobb, ha az online óra alatt a résztvevők nem csak nézik a megosztott képernyőt és hallják a tanár hangját, hanem látják is őt. Természetesen ez nem mindig megfelelő az óra teljes időtartamára. A tanár egy videó hívás segítségével egyszerűen és gyorsan (valós időben) adminisztrálhatja az óra menetét. Ha lehetséges, a videó hívások közben javasoljuk, hogy használja a képernyő háttérváltóját. Ezzel elkerülhető, hogy a diákok kritizálják (vagy dicsérjék) a pedagógus munkahelyét vagy otthonát. Talán még ennél is fontosabb, hogy nem kell „eltakarítani" a „munkahelyi rendetlenséget".

9. Gondolj arra, hogy „az ördög soha nem alszik" és „az internet néha összeomlik"!

Indokok: Mindegy, hogy otthonról vagy munkahelyről (irodából, tanteremből) dolgozik a pedagógus, előfordulhat áramszünet, vagy megszakad az internetkapcsolat. Ha ez valamelyik diák részéről történik, akkor nem akkora nagy a probléma, sokszor a többiek észre sem veszik, a baj akkor van, ha ez a tanár oldalán van. Megfelelő megoldás a mobilhálózat használata lehet, amely legalább arra szolgál, hogy a pedagógus értesítse az online óra többi résztvevőjét a problémáról.

10. Legyél te is „YouTuber"!

Indokok: A tanárnak tisztában kell lennie azzal, hogy a diákok lehet egész nap a számítógép előtt ülnek és a képernyőt nézik. A tanárok váltják egymást, a monitoron változnak az előadások képei/prezentációi, valaki felolvassa a szöveget, jó esetben saját szavaival magyarázza. Fárasztó. Akkor

miért váljon egy tanárból „YouTuber”? Azért, hogy egy YouTuberhoz hasonlóan, akár napokig a képernyő előtt tarthassa a diákjait, gyakran szünet nélkül. Ehhez érdekes módon kell „fűszereznie” az órái menetét. [4]

5. Befejezés

Nyilvánvaló, hogy a járványhelyzet alatt gyakorolt távoktatási formákkal szerzett tapasztalatok hasznosak voltak és hasznosak maradtak a résztvevők mindkét oldalának, azaz pedagógusoknak és diákoknak is egyaránt. Kiderült, hogy az online oktatás (távoktatás) lehet hatékony és sok diák számára még kényelmesebb is.

A cikkben megfogalmazott ajánlásainkat elsősorban a leendő és gyakorló pedagógusoknak szántuk, mivel ők azok, akik részt vesznek az online oktatás sikeres megvalósításában. Ők azok, akik tudásukat és tapasztalataikat a jövő nemzedékének nevelésében tudják kamatoztatni, hogy diákjaik iskolai végzettsége a lehető legmagasabb szintet érje el, függetlenül attól, hogy milyen képzésben részesültek.

Köszönetnyilvánítás

A tanulmány a KEGA 012TTU-4/2021: „Integrácia využívania dištančných výučbových procesov a tvorby elektronických učebných materiálov do edukácie budúcich pedagógov.” (A távoktatási folyamatok alkalmazásának és az elektronikus tananyagok készítésének integrálása a leendő pedagógusok oktatásába) című projekt keretében készült.

Irodalom

1. Burns, M.: *Distance Education for Teacher Training: Modes, Models, and Methods*. Education Development Center, Inc. 2011.
   https://www.edc.org/sites/default/files/uploads/Distance-Education-Teacher-Training.pdf (utoljára megtekintve: 2022.10.22.)
2. Palvia, S., Aeron, P., Gupta, P., Mahapatra, D., Parida, R., Rosner, R., Sindhi, S.: *Online Education: Worldwide Status, Challenges, Trends, and Implications*, In: Journal of Global Information Technology Management, 21 (2018) 4, 233-241, DOI: 10.1080/1097198X.2018.1542262
3. Pšenáková, I.: *Tvorba didaktických interaktívnych materiálov a kritériá hodnotenia ich kvality*. Trnava: Pedagogická fakulta Trnavskej univerzity v Trnave, 2021. online, 50 s.
   https://pdf.truni.sk/veda-vyskum?e-kniznica#monografie
4. Pšenáková, I., Pšenák, P., Kováč, U.: *Skúsenosti a poznatky z on-line vzdelávania počas pandémie covid-19 = Experience and knowledge from online education during the covid-19 pandemic*. In: Proceedings of 33. DidMatTech 2020 Conference. Budapest: Eötvös Loránd University, Trnavská univerzita v Trnave, 2020. [online] 110-118.

Elektronikus tananyag és tananyagelemek használata az általános iskola alsó tagozatán

Pšenáková Ildikó[1], Szabó Tibor[2]

[1]ildiko.psenakova@truni.sk, [2]tszabo@ukf.sk
Trnava University in Trnava, Constantine the Philosopher University in Nitra

Absztrakt. A tanulmány az elektronikus tananyag tervezésével, létrehozásával és használatával foglalkozik, különös tekintettel az általános iskola alsó tagozaton tanulók számára. Tartalmában kitér az elektronikus tananyagelemek helyes készítésének alapelveire, ismerteti a jó tananyag jellemzőit és bemutat néhány alkalmazást, amelyek az interaktív feladatok készítésében lehetnek segítségre a pedagógusoknak. Ezen kívül az ismertetett alkalmazásokhoz az alsó tagozatos tanulók számára készített konkrét példákkal is szolgál.

Kulcsszavak: elektronikus tananyag, interaktív feladat, tervezés, használat, alsó tagozat, tanuló

1. Bevezetés

A mai világ az élet minden területén nagyon gyorsan változik, és a technikai fejlődés folyamatosan halad előre. A digitalizáció és a modern technológiák az emberi élet különböző területeire jutnak el, amiből nem kivétel az iskolarendszer sem. Az iskola alkalmazkodik ezekhez a változásokhoz, hogy megfeleljen a modern társadalomnak és feltételeket biztosítson a gyermekek minőségi oktatására.

Az iskola korszerűsítése alatt a különböző digitális technológiákat értjük, amelyek az oktatási folyamat javítását segítik elő. A modern műszaki berendezések, technikák, oktatási formák és módszerek segítségével, megfelelő használatával megújulhat az oktatás. Fontos azonban, hogy a pedagógusok elsajátítsák ezen eszközök használatának módszertanát, ellenkező esetben csak az iskolákban kihasználatlanul álló drága felszerelések, ill. szoftverek lesznek, amelyek nem segítik a tanulási folyamatot.

Az oktatás során a pedagógusok különböző módszerek és eszközök segítségével próbálják a tananyagot a tanulóknak átadni. Ezek közé sorolhatjuk az elektronikus (digitális) tananyagot is.

2. Elektronikus (digitális) tananyag

Az elektronikus (digitális) tananyag fogalmát az irodalom sokféleképen definiálja. Legegyszerűbb megfogalmazása talán az lehetne, hogy „*a digitális tananyag - elektronikus (digitális) formátumban tárolt adatok halmaza, amely alkalmas valamilyen információ - tudás átadására.*" [1]

A könyvtári gyakorlatban használatos az elektronikus dokumentum fogalom, amely „*számítógéppel kezelhető, digitálisan kódolt dokumentum, amely vagy valamely fizikai hordozón jelenik meg, és használatához számítógéphez illesztett, vagy annak részét képező periféria (pl. CD-ROM lejátszó, lemezmeghajtó) szükséges, vagy hálózati úton érhető el (pl. távoli hozzáférésű adatbázis, elektronikus hirdetőtábla, elektronikus időszaki kiadvány, webterület)*". [2]

Az elektronikus dokumentum ilyen meghatározása kiegészítheti az elektronikus tananyag fogalmát, mivel pontosítja az elektronikus (digitális) formátumban tárolt adatok halmazának, beleértve az elektronikus tananyagot is, lehetséges helyét (pl. CD-ROM, webterület stb.).

Nádasi szerint „*a „digitális tananyag" fogalmát általánosságban nehéz definiálni, mivel az oktatástechnológia legdinamikusabban fejlődő területe. Legegyszerűbb megközelítésben digitális tananyag lehet minden elektronikus, ma már szinte kizárólag digitális formátumban tárolt és elérhető szellemi alkotás, amely alkalmas valamilyen tudás, információ átadására, közvetítésére. Az online tananyag az internetes változat*." [3]

Köztudott, hogy a több érzékszervre is ható oktatási segédlet jobb tanulói eredményekhez, és a tananyag tartósabb elsajátításához vezet. Ezért előnyös, ha az elektronikus tananyag interaktív elemeket is tartalmaz. Akkor már interaktív segédeszközzé válik, ami „*olyan didaktikai eszköz, amely különféle dokumentumformátumokat (pl. szöveg, táblázat, animáció, kép, hang, videó stb.) integrál, a valóságot közvetíti vagy imitálja, segíti annak szemléletesebbé tételét vagy megkönnyíti a tanítást.*" [4]

Az elektronikus tananyagelemek kihasználásának egyik alapfeltétele, hogy a tanárok és a tanulók egyaránt tudják használni legalább a számítógép alap funkcióit. Ma már ez szinte természetessé vált, és a tanulóknak, mint digitális bennszülötteknek, nem okoz gondot az elektronikus tananyag használata. Megjelenhetnek azonban problémák is, mint például a hardver és szoftver hiánya. Sok iskola egyszerűen nem rendelkezik a megfelelő technikával, vagy nem jut mindenki számítógéphez (illetve más szükséges eszközhöz), esetleg nincs internet hozzáférés. Ha az iskolában vannak is számítógépek, gyakran elavultak, vagy nem rendelkeznek megfelelő szoftverrel, esetleg kompatibilitás problémák lépnek fel. Ennek ellenére a pedagógusok a tantermi oktatásban is egyre gyakrabban használják az elektronikus anyagokat, amelyek nélkül az elmúlt járványhelyzetben a tanítást nehezen lehetett volna megoldani.

2.1. Elektronikus tananyagelemek és interaktív feladatok készítése

Saját elektronikus tananyag készítése nehéz és kihívásokkal teli folyamat a pedagógusnak. Különösen a fiatalabb iskoláskorú gyermekeknek szánt anyagok fejlesztése nem egyszerű. Az elektronikus gyakorlatoknak vonzóknak, megnyerőknek, lebilincselőknek kell lenniük, de ugyanakkor egyszerűeknek és megérthetőknek a tanulók számára. Ezen kívül sok lehetőséget kell biztosítaniuk a tanulónak az aktív önálló alkotómunkára.

Ahhoz, hogy az elektronikus tananyagot a tanár el tudja helyesen készíteni, elég sok számítástechnikai, ill. informatikai tudásra van szüksége. Ezekkel azonban nem minden tanár rendelkezik. A jó tananyag elkészítéséhez nemcsak szakmai-, didaktikai- tudás, számítógépes jártasság és kreativitás, de idő is szükséges. Ilyen szakember, akiben mindezek a képességek megvannak, pedig kevés van. Éppen ezért az elektronikus tananyag elkészítésére az igazán megfelelő forma a csoportmunka. [1] Erre azonban nem mindig van lehetőség, ezért a pedagógusok gyakran használnak az elektronikus tananyag létrehozására irányuló szoftvereket, melyek segítségével egyszerűbbé válik a munka.

Sok tanár él a saját tananyag elkészítésének lehetőségével. Bár időigényes és fárasztó munka, számos előnnyel jár. Az egyik, hogy saját elképzeléseik és aktuális igényeik alapján készíthetik el azokat. További előnye, hogy a tananyagokat nem csak egyszer használhatják fel, hanem bármikor át lehet ezeket alakítani, igény szerint bővíteni a tartalmukat, módosítani, frissíteni.

2.2. Az elektronikus tananyagelemek helyes készítésének néhány alapelve

Ha egy tanár olyan teljes értékű elektronikus tananyagot szeretne készíteni a tanulók számára, amely alkalmas az adott évfolyam tanterve szerinti tanításra, akkor az anyag elemei elkészítésekor több alapelvet is szem előtt kell tartania:

1. *Célok* – első lépés a célok megfogalmazása, ennek elmaradása komoly hibának számít. Világos konkrét célok nélkül a tanárok gyakran elkalandozhatnak a tanítási témától. A tartalommal csak a célok definiálása után érdemes foglalkozni. Tűzzünk ki konkrét célt (tantárgy, tanterv, célcsoport szerint), amit az elektronikus tananyag segítségével szeretnénk a tanulók számára elérni. Az alapfokú oktatói-nevelői folyamatban résztvevő gyermekek célcsoportjait az adott korosztály

specifikumai szerint szükséges figyelembe venni. Minél pontosabban határoljuk be a célközönséget, annál biztosabbak lehetünk, hogy ki tudjuk igényeiket elégíteni. [1]

2. *Tartalom* - a kiválasztására több tényező is hat. Döntő tényezőként a képzettség áll. Hasonlóan, mint a tankönyv írójának, az elektronikus tananyag alkotójának is alkalmazkodni kell a tanulók tudás szintjéhez. A kiválasztott tartalomnak egységet kellene alkotni, hogy a tanulók megértsék. A nyelv szintén ide sorolható, mivel helytelen használni trágár szavakat és szlenget. A tananyagban felhasznált információknak meg kell felelni az egyes tantárgyak szellemi beállítottságának, szerkezeti és logikai rendezettségének és helyes beillesztésének az egyes tantárgyakba.
3. *Ellenőrzés* - célszerű-e a kiválasztott témakört/tartalmát elektronikusan feldolgozni. Ellenőrizni, hogy informatikai segédeszközökkel támogatott oktatással (elektronikus tananyag segítségével) lehetséges-e a megfogalmazott célt elérni, hogy megtudják-e a tanulók szerezni a szükséges ismereteket. A pedagógusnak meg kell fontolnia, hogy a kiválasztott tananyag megfelelő-e az elektronikus formában való prezentálásra. Felesleges például a tanulóknak olyan kísérleteket nézni videó formájában, amelyeket maguk is biztonságosan végrehajthatnak az osztályban vagy laborban. De teljesen jogos olyan folyamatokat elektronikusan prezentálni, melyeket iskolai környezetben nem lehet, esetleg nem ajánlatos végrehajtani. [5]
4. *Tervezés* - az elektronikus tananyag elemei készítésének megkezdése előtt a tanárnak el kell gondolkodnia azon, hogy az egész óra alatt, vagy csak bizonyos részeknél (motiváció, főrész, következtetés) kívánja-e az informatikai eszközöket (számítógép, projektor, interaktív tábla stb.) használni.
5. *Megfelelőség* – fontos figyelembe venni a tanulók képességeit, készségeit és érdeklődését az elektronikus anyagok tervezése és elkészítése során. Az elkészített anyagoknak fel kell hívniuk a tanulók figyelmét.
6. *Szerkesztés* – a tananyaghoz szükséges segédanyagokat célszerű előre elkészíteni és megszerkeszteni. Ezek lehetnek:
   - Képek, fényképek (megfelelő méretben)
   - Hangok, animációk, videók
   - Grafikonok, diagramok
   - Dokumentumok (az általuk lefedett tananyag teljesítéséhez)

   A segédanyagok elmentése a számítógépre meggyorsíthatja a tananyag szerkesztését. Az előkészített és szerkesztett képek, hangok, videók, beillesztése növeli a tananyag érthetőségét, a meglévő ismeretek bővítését és módosítását.
7. *Szöveg* – a szöveg megfelelő elhelyezése a képernyőn, illetve kivetítő vásznon vagy az interaktív tábla felületén, fontos tényező. A megfelelő betűméret (hogy a tanuló az iskolapadból is el tudja olvasni a szöveget), szín (a betűszín nem keveredhet össze a háttérrel; a színek használata szórakoztatóbbá és vidámabbá teszi a tananyagot), betűtípus (az éppen olvasni tanulók nem ismernek sok betűtípust és nem tudják elolvasni a szöveget). Ezeket a tényezőket figyelembe kell tartani a szöveg szerkesztésénél. [6]
8. *Interaktivitás* - ahhoz, hogy az elektronikus tananyagok érdekesek és vonzóak legyenek a tanulók számára ma már sokszor elvárt tényező, hogy interaktívak is legyenek.
9. *Hivatkozások* –sokat segít a tanulóknak a navigációs hivatkozások beillesztése az elektronikus tananyagba. Ezek utalhatnak különböző weboldalakra, ahol a tananyagot kiegészítő dokumentumokat, mellékletek találnak, amelyek segítik a tananyag megértését, rögzítését.
10. *Ellenőrzés* – utolsó lépésként az elkészült elektronikus tananyag minden részét „illik" letesztelni. Ellenőrizni kell az összes szöveget (elírások, helyesírási hibák javítása), grafikus elemeket (kép,

grafikon, táblázat), linkeket (működésük) stb. Lényegében az elkészült „végeredmény" összes működőképes részeinek végső ellenőrzése. [7]

Igaz, hogy az elektronikus tananyagok készítése időigényes tevékenység. Ez jelentős visszatartó erő lehet egyes tanárok számára. De nézzük a dolog másik oldalát. Az elkészített tananyagelemek elmenthetőek a számítógép memóriájába, és szükség szerint tovább és többször használható. A tananyag könnyen módosítható, igazítható, kiegészíthető, ami minimális időt vesz igénybe. Ha kezdetben a tanároknak több erőfeszítést és időt is kellet fordítaniuk az tananyag elkészítésére, de a későbbi használat során ezt kamatoztatni tudják, mert csak kinyitják az anyagot, és felhasználhatják a tanítás során.

2.3. Milyen is legyen egy jó elektronikus tananyag?

- *Átlátható*: a tananyag struktúrája legyen azonnal átlátható. Ez elérhető, ha kisebb részekre osztva jelenik meg a képernyőn. A látható szöveg alkalmazkodjon a képernyők méretéhez, és ne legyen túl zsúfolt.
- *Átjárható*: az elhelyezett navigációs linkek működjenek, ne legyenek félreérthető elnevezésű és működésű menüpontok, tanulás közben lehetséges legyen visszalépni.
- *Felhasználó barát*: egyszerű kezelés, hogy a tanuló intuitívan tudja használni, fontos az olvashatóság (betűtípusok, méretek, egyszerű háttér).
- *Hatékony*: hatékonyságot emelő vizuális elemek (kép, animáció, hang) használata, könnyű szövegolvasás, jó tanulhatóság.
- *Karbantartott*: gyakori, hogy az elektronikus tananyag az internet segítségével weboldalakon is megjelenik. Ilyen esetben a legmodernebb, legújabb informatikai megoldásokat érdemes alkalmazni. Ezek lehetőséget adnak a tananyag mindennapos, sőt minden perces javítására, bővítésére, törlésére, de biztosítani tudják a tanár-tanuló, tanuló-tanuló kommunikációt is. [1]

Ez csak néhány tulajdonság a sok közül, melyek hatással lehetnek a digitális tananyag minőségének megítélésére. Azonban a grafikai elemek, színharmónia, alkalmazott technológia és átláthatóság azok az elemek, amelyek meghatározzák egy elektronikus tananyag összképének megítélését. A digitális tananyagok kiválóan alkalmazhatók a mindennapi tanításban és az órákra való felkészülésben is. A tanuló új típusú tanulási környezetet talál, és a tananyag egy új hatékony információ átadási módon jut el hozzá.

Az elektronikus tananyag helyes megtervezése, létrehozása és felhasználása az oktatásban kétségtelenül megnövelheti az oktatás hatékonyságát és eredményességét. Ezért a pedagógusnak kell eldöntenie, melyik tantárgyban, témakörben akarja, vagy tudja használni az elektronikus tartalmakat.

3. Szoftverkörnyezetek elektronikus tananyagelemek és interaktív faladatok készítéséhez

Az elektronikus tananyagelemek készítésére különféle szoftver alkalmazások léteznek. Egyesek ingyenesek, úgynevezett freeware alkalmazások, mások különböző pénzösszeg ellenében érhetőek el. Olyan alkalmazások is léteznek, amelyek ingyenes használatot kínálnak a felhasználónak, de csak alap, korlátozott formában, a teljes verziót a felhasználó csak bizonyos pénzösszegért kaphatja meg. Léteznek online - webes alkalmazások, vagy számítógépre letölthetőek – offline alkalmazások.

Ha a pedagógus online alkalmazásokat használ az elektronikus tananyag készítéséhez, akkor egyértelmű, hogy internetkapcsolattal kell rendelkeznie. Sok ilyen alkalmazáshoz felhasználói fiók létrehozása szükséges, ami nagyon egyszerű és nem jelent gondot a tanítóknak. Sokszor számos videó és leírás is segít a regisztrációban és az alkalmazás megismerésében (mit kínál az alkalmazás, hogyan kell vele dolgozni). Nagy előnye az online alkalmazások használatának, hogy a létre-

hozott tananyagokat a weboldal hivatkozásán keresztül egyszerűen meg lehet osztani. A tananyagok megosztásának ez az egyszerű elve nagyon hasznos lehet az iskolákban és jól használható az általános iskola alsó tagozatán is. A tanár e-mailben elküldi a tanulóknak a linket, amelyen megtalálhatják az anyagot a gyakorláshoz, és a tanulók otthon is használják. Az alkalmazások sokféle sablont, különböző típusú gyakorlatokat biztosítsanak a felhasználónak (pl. keresztrejtvény, kiegészítés, párosítás stb.), amelyekbe a tanár beépítheti a tananyagot, de lehetőség van arra is, hogy saját ötleteit, kreativitását is kamatoztassa a készítésnél.

Az alábbiakban interaktív feladatok készítésére szolgáló szoftverek néhány példáját mutatjuk be. Az ilyen és hasonló program csomagok segítségével elkészített feladatok egyszerűen beépíthetőek az elektronikus tananyagba tananyagelem vagy tananyagsegédletként, de önmagukban is felhasználhatóak a tanórákon vagy otthoni gyakorló feladatokként.

3.1. Learning Apps

A Learning Apps egy érdekes online alkalmazás saját elektronikus interaktív gyakorlatok készítéséhez (rendezés, vetélkedők, párosítás, kép leírása, szavak kitalálása, pexeso stb.), amely a *http://learningapps.org* címen ingyenesen érhető el. Használatához saját fiókra van szükség, amely létrehozásához csak a felhasználónevet, az e-mail címet és a jelszót szükséges megadni. Később hozzáadható az avatar, iskola adatai, webhelye, és kiválasztható, hogy a felhasználó e-mail címe nyilvános legyen-e mások számára. Mivel Web 2.0-s eszközről van szó, az elkészített gyakorlatok különféle internetkapcsolattal rendelkező eszközökön – számítógépen, táblagépen, okostelefonon – futtathatók.

A Learning Apps jelenleg 22 nyelvet támogat, közöttük a magyart (1. ábra) és a csehet is, de a szlovákot nem, ezért Szlovákiában kevés pedagógus használja.

1. ábra: A Learning Apps kezdőlapja.

A tanár közel 40 különböző típusú tankocka közül választhat a Learning Apps-ben, amelyeket saját tartalommal tölthet meg, vagy más felhasználók már elkészített feladatait is megkeresheti, amelyeket azonnal felhasználhat. A feladattípusok hat tematikus kategóriába sorolhatók: választási lehetőségek, illesztők, szekvenciák, gépelők, többjátékos feladatok és különféle segédeszközök. A gyakorlatokba multimediális elemeket is beépíthetők, például képek, videók vagy hangfelvételek. A létrehozott tevékenységek mappákba rendezhetők, másolhatók, áthelyezhetők és természetesen bármikor szerkeszthetők. [8]

Egyes tanárok és tanulók a Learning Apps hátrányának tekinthetik azt, hogy a gyakorlatok megoldása közben nem mutatja a részeredményeket, az elért pontszám csak a gyakorlatok végén je-

lenik meg. A feladat kitöltése közben azonban megjelennek a helyes válaszok, így a tanulók látják, hol hibáztak. Az említett hátrány azonban előnyt is jelenthet a tanulásban, mivel megoldás közben a tanulót nem demotiválja a kudarc.

Példánk az általános iskola 4-ik osztályos tanulóinak készült az informatika órára. A témakör tartalma az e-mail (2., 3. ábra). A feladatban párosítani kell az össze tartozó szavakat, illetve fogalmakat. Például a kukac (zavináč) szót a @ -hoz.

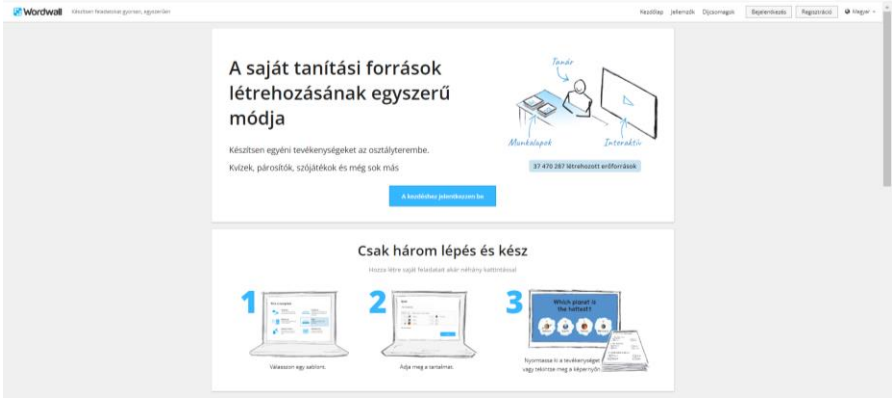
A feladat megtalálható a https://learningapps.org/watch?v=pyay0unb222 címen.

2. ábra: Feladat.

3. ábra: Feladat megoldása.

3.2. Wordwall

A Wordwall egy online alkalmazás saját tanítási források létrehozásához. Az alkalmazás a *https://wordwall.net* címen érhető el, magyar nyelven is (4. ábra). A program fizetős (pl. standard verzió 5 €/hónap), de alapvető ingyenes verziót is kínál, amely regisztrációhoz kötött. Ebben természetesen bizonyos funkciók és a készíthető feladatok száma korlátozott. A programban a felhasználó könnyen létrehozhat interaktív gyakorlatot az általa kínált sablonok segítségével. Az elkészített gyakorlatok bármilyen webböngészővel rendelkező eszközön használhatók, pl. számítógép, interaktív tábla, táblagép, okostelefon.



4. ábra: A Wordwall kezdőlapja.

A Wordwall különböző előre programozott interaktív gyakorlat típusokat kínál az oktatóknak, amelyekbe csak a tananyag tartalmát kell beilleszteni a tanulók aktuális igényeinek megfelelően. A tanár létrehozhat egy tanulókból vagy más tanárokból álló csoportot (vagy osztályt), akikkel megoszthatja az összes általa létrehozott tananyagot. Ugyanakkor a program értékelési eszközei lehetővé teszik számára, hogy ellenőrizze a feladatok elvégzését, a gyakorlatok és feladatok kidolgozásának mértékét, illetve a kidolgozásukhoz szükséges időt. Mindezek az adatok lehetővé teszik a tanár számára, hogy áttekintést kapjon az egyes tanulók tanulmányi eredményeiről vagy

szükségleteiről és általános tanulási előrehaladásáról, így a Wordwall fontos diagnosztikai eszközzé is válik.

Mint már említettük, a Wordwall lehetővé teszi, hogy a tanár a tanulók tanulási igényeihez igazított feladatokat készítsen. Összesen 33 féle interaktív gyakorlat áll a pedagógus rendelkezésére, amelyek közül 18 az ingyenes változatban érhető el. Ezek közé tartoznak például: Egyezés, Kvíz, Szerencsekerék, Kártyaosztó, Doboznyitó, Csoportosító, Párosító, Hiányzó szó, Egyező párok, Anagramma, Feloldó, Játékos kvíz, Diagram, Igaz vagy hamis, Üss a vakondra, Labirintus, Lufi pukkasztó, Repülőgép. [9]

Példánk az általános iskola 3-ik osztályos tanulóinak készült szintén az informatika órára. A témakör tartalma a billentyűzet (5. ábra). A feladat a labirintus sablonban készült. A tanulóknak a képernyő alján megjelenő kérdésre úgy kel válaszolni, hogy eljuttatják kis figurájukat a labirintuson keresztül a helyes válaszhoz. Közben vigyázniuk kell nehogy elnyeljék figurájukat a szörnyecskék. A feladat a https://wordwall.net/sk/resource/31958898/počítačový-labyrint címen található.

5. ábra: Labirintus (WordWall).

3.3. HotPotatoes

A Hot Potatoes szoftvercsomag freeware, tehát szabadon hozzáférhető, és bármilyen célra vagy projektre felhasználható. A program megtalálható a *http://hotpot.uvic.ca/* weboldalon, és jelenleg a 7. verzió tölthető le. Hat részből áll, amelyek segítségével interaktív feladatok készíthetők (6. ábra):

1. *JCloze* – lyukas szöveges feladat készítése. A szövegben a tanár hézagokat (lyukakat) helyez el, amelyeket a tanulóknak ki kell tölteni. Minden réshez egy vagy több helyes válasz is adható. Kitöltés közben a tanuló a megfelelő gomb megnyomásával segítséget kérhet, és a helyes válaszból egyszerre egy-egy betű jelenik meg.
2. *JMatch* – párosítási feladat készítése. A képernyő bal oldalán egy állandó elemekből álló lista van (ezek lehetnek képek vagy szövegek), a jobb oldalon pedig a rendezetlen válaszok találhatók, amelyeket a bal oldali lista elemeihez kell helyesen illeszteni. Az ilyen típusú feladatok használhatók például a szókincs gyakorlására, a képeken látható tárgyak megnevezésére stb.
3. *JQuiz* – felelet választós vagy rövid válaszos feladat készítése. Négy különböző típusú kérdés hozható létre: feleletválasztós, rövid válaszos, többválasztós és vegyes. A különböző feladattípu-

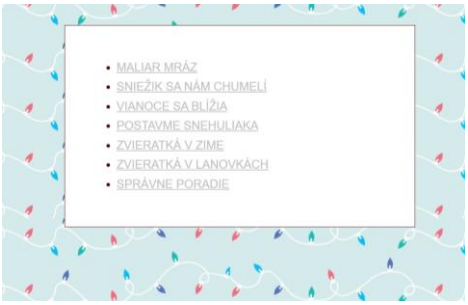
sok önmagukban is használhatók, vagy kombinálhatók egyetlen gyakorlaton belül. Mind a helyes, mind a helytelen/hibás válaszokra konkrét visszajelzés adható.

4. *JCross* – keresztrejtvény készítése, amely online kitölthető. Gyakorlatilag tetszőleges méretű rácsot lehet használni, és a létrehozás nagyon egyszerű, mivel a keresztrejtvényt egy automatikus készítő segítségével jön létre.
5. *JMix* – összekevert mondat készítése. A tanuló feladata, hogy a rendezetlen mondatokat kibogozza, a matematikai példák kifejezéseit vagy eredményeit helyes sorrendbe rendezze. A feladattól függően több helyes válasz is adható. Az alapmondatban írásjelek is használhatók.
6. *The Masher* – feladatok összefűzése indexelt egységbe. Ez azt jelenti, hogy a létrehozott fájlokat (melyek a különböző típusú gyakorlatokat tartalmazzák) automatikusan egy sorozatba köti, amit .html formátumú fájlban ment el. Ez a formátum lehetővé teszi a feladatok weben való használatát is.

6. ábra: A Hot Potatoes kezdőlapja.

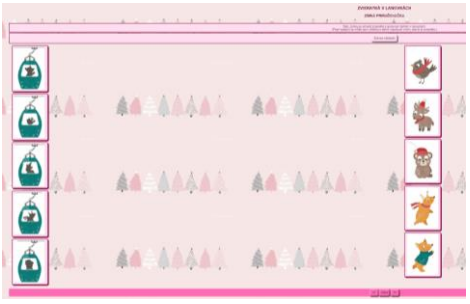
A feladatok eredményét a program automatikusan százalékos formában értékeli ki. Ha a tanuló segítséget használt, a százalékok megfelelő arányban levonásra kerülnek.

Példánk az általános iskola 2. osztályos tanulóinak készült és a karácsony témáját dolgozza fel.



The Masher

JClose



JMatch

JCross

JQuiz

JMix

7. ábra: Példák a Hot Potatoes programban készült feladatokból (készítette: Pechová Monika)

4. Befejezés

Manapság az általános iskolákban egyre több gyermek küzd tanulási problémákkal. Ennek leggyakoribb okai a koncentrálóképesség hiánya, figyelemzavarok, fejlődési tanulási zavarok, a gyerekek szociális környezete, a kudarctól való félelem és még sok más. Ezek leküzdésében nagyon sok segítséget nyújthatnak a tanárok, akiknek a pedagógiai mesterségén múlik, hogy minden gyermek tanulás iránti érdeklődését felkeltsék. Az elektronikus interaktív tananyagok használatával nemcsak az új ismeretek elsajátításának hatékonyabb módja érhető el, hanem a játékos és szórakoztató tanulási mód örömének megidézése is. Fő feladatuk a tudás rögzítése és jobb memorizálása. Az interaktív gyakorlatok segítségével a tanulási folyamat szemléletesebbé, vonzóbbá válik a tanulók számára, mint a könyvekkel való tanulás szokásos módja. Fontosnak tartjuk azt is, hogy a gyakorló, ill. a leendő pedagógusok is képesek legyenek készíteni és helyesen használni elektronikus tananyagokat.

Köszönetnyilvánítás

A tanulmány a KEGA 012TTU-4/2021: „*Integrácia využívania dištančných výučbových procesov a tvorby elektronických učebných materiálov do edukácie budúcich pedagógov.*” (A távoktatási folyamatok alkalmazásának és az elektronikus tananyagok készítésének integrálása a leendő pedagógusok oktatásába) és a KEGA 011UKF-4/2022: „*Inovatívne vzdelávacie materiály s dôrazom na výchovu zdraviu a environmentálnu výchovu žiakov 3. a 4. ročníka ZŠ*” (Innovatív oktatási anyagok, különös tekintettel az egészségi és környezeti nevelésre az általános iskola 3. és 4. évfolyamának tanulói számára) című projektek keretein belül készült.

Irodalom

5. Pšenáková, I.: *A digitális tananyag*. In: Képességfejlesztés digitális tananyaggal. (2010) Debrecen: Kocka kör, 10.
6. Berke B.: *Az elektronikus dokumentumok és könyvtári használatuk*. In: Könyvtári Figyelő, 47 (2001) 2. 275.
7. *Nádasi, A.: Online tananyaok az oktatásban.* http://digitall.uni-eger.hu/tananyagok/learn//04_online_tananyagok_az_oktatasban_nadasi_andras/index.html (utoljára megtekintve: 2022.10.28.)
8. Dostál, J.: *Interaktivní tabule ve výuce*. In: Journal of Technology and Information Education. 2009. https://jtie.upol.cz/pdfs/jti/2009/03/02.pdf (utoljára megtekintve: 2022.10.20.)
9. Pšenáková, I., Mészárosová, Z., Papp, L.: *Médiapedagógia*. Univerzita Konštantína Filozofa, Nitra. 193 p. (2009)
10. Pšenáková, I.: *Tvorba interaktívnych aplikácií*. Trnava: Typi Universitatis Tyrnaviensis, (2019)
11. Brečka, P., Koprda, Š.: *Metodika tvorby elektronických výučbových materiálov pre interaktívne tabule.* In: Trendy ve vzdělávaní: informační technologie a technické vzdělávaní. Monografie z medzinárodní konference, 2012, Olomouc: GEVAK, p. 713-720.
12. Farárik, P.: *Tvorba interaktívnych cvičení v Learning Apps s dôrazom na geografiu*. Metodicko-pedagogické centrum. (2014) https://lepsiageografia.sk/wp-content/uploads/2019/04/far%C3%A1rik-learning-apps.pdf (utoljára megtekintve: 2022.10.20.)
13. *WordWall. Egyedi tervek.* https://wordwall.net/hu/price-plans (utoljára megtekintve: 2022.10.25.)

A digitális tanrend tapasztalatai Magyarországon a Covid 19 ideje alatt

Rumbus Anikó[1], Szlávi Anna[2]

[1] rumbus@inf.elte.hu, [2] anna.szlavi@ntnu.no
[1] Eötvös Lóránd Tudományegyetem Informatika Kar,
[2] Norwegian University of Science and Technology

Absztrakt. 2020. március 15-ével egyik pillanatról a másikra valamennyi oktatási intézménynek digitális tanrendre kellett áttérni a pandémiás helyzet miatt. A már régebb óta krízis helyzetben lévő pedagógus társadalom sem eszközökkel, sem szoftverekkel, sem módszerekkel nem volt ellátva. A tanároknak maguknak kellett megtanulniuk az alkalmazások használatát, illetve az online tanítási módszereket. A legtöbb intézményben belső konzultációk, továbbképzések zajlottak. A tanárok egymást tanították az új módszerek, platformok használatára. Rengeteg Facebook csoport alakult egymás megsegítésére. Az amúgy is leterhelt pedagógusok még több feladatot kellett ellássanak. Hirtelen autodidakta módon el kellett nekik kezdeniük egy újfajta világban felkészülni tanórákra. Az átállás nem ment zökkenőmentesen, amelynek számos oka volt. A tanárokat nem látták el kellőképpen eszközökkel (laptop, tablet stb.), online tanítási módszertárral. A diákok is teljesen új helyzetbe kerültek. Legtöbbjüknél technikai (eszköz, internet stb.) gondok nehezítették a tananyaggal való haladást, tanórákon való részvételt. Alsó tagozaton, az eszközök használata is komoly nehézségeket okozott. Kutatásunkban arra is kerestük a választ, hogyan vélekednek a tanárok a digitális tanrend körülményeivel kapcsolatban. Volt-e pozitív hozadéka az online oktatásnak? Alakítottak-e ki új módszereket a tananyag átadásában, számonkérésénél?

Kulcsszavak: Covid19, digitális tanrend, online alkalmazás, online tananyag, módszertan, oktatási reform

1. Bevezető

2019 végén kitört a COVID19 nevű koronavírus járvány, amely Kína után Európát és a világ többi részét is elérte 2020 elejére. A világjárvány komolysága miatt a legtöbb ország korlátozni kényszerült a személyes kontaktust, ami az oktatásra is kiterjedt. Egyre több ország tért át a digitális oktatásra, ami sok esetben kihívást jelentett a társadalomnak. Több technikai cég kedvezményes, esetenként ingyenes hozzáférést adott a digitális oktatást segítő szoftvereihez, többek között a norvég fejlesztésű Kahoot! is (Bernát – Szlávi, 2022). Emellett, digitális oktatást segítő projektek is elindultak 2020-ban, például a DIGIVID nevű, osztrák, német és norvég partnereket egyesítő Erasmus+ konzorcium (DIGIVIDget(turgaz.at)). A projekt célja egy olyan platform létrehozása, amely a DigComp 2.1 es DigCompEdu keretrendszert követve, online oktatási stratégiákat és kompetenciákat ad a tanároknak. A platform kifejlesztése és tesztelése még mindig tart, így a tényleges digitális tanrend ideje alatt nem lehetett segítség.

Magyarországon 2020. március 15-ével valamennyi hazai oktatási intézménynek digitális tanrendre kellett áttérni a pandémiás helyzet miatt szinte egyik pillanatról a másikra. Az első hetek azzal teltek a tanárok közt, hogy a rengeteg felmerült kérdésre válaszokat találjanak. A már régebb óta krízis helyzetben lévő pedagógus társadalom sem eszközökkel, sem szoftverekkel, sem módszerekkel nem volt ellátva. (Papp-Danka, 2014) A tanároknak maguknak kellett megtanulniuk az online alkalmazások használatát, illetve az online tanítási módszereket felkutatni, kidolgozni. A legtöbb

intézményben belső konzultációk, továbbképzések keretein belül próbáltak megoldásokat találni. A pedagógusok egymást tanították az új módszerek, platformok használatára. Számos Facebook csoport is alakult egymás segítésére.

Az amúgy is leterhelt pedagógusoknak most még jóval több feladatot kellett ellátniuk egy teljesen új területen, az internet világában. Hirtelen kellett váltaniuk a jelenlétből az online térre. A legtöbben autodidakta módon kezdtek el felkészülni tanóráikra.

Az átállás egyáltalán nem ment zökkenőmentesen, amelynek számos oka volt. A tanárokat nem látták el megfelelő számú eszközökkel (laptop, tablet stb.), és komoly gondot okozott az online oktatási módszerek ismeretének hiánya is. (Lévai, 2014) A diákok is egy teljesen új helyzetbe kerültek. Legtöbbjüknél technikai (eszköz, internet stb.) gondok nehezítették a tananyaggal való haladást, tanórákon való részvételt. (N. Kollár, 2021)

A hátrányos helyzetű tanulók esetében nem is sikerült online megoldást találni. Nekik a legtöbb esetben az iskolatitkárok papír alapon kiadták, vagy kipostázták a tanulni valókat, melyet önállóan kellett feldolgozniuk, megoldaniuk, majd visszajuttatni tanáraiknak. Alsó tagozaton további súlyos problémát jelentett az eszközök használata is. Legtöbbjük még semmilyen informatikai eszközzel nem volt kapcsolatban, nem használt okos eszközt tanuláshoz. A legtöbb szülő a digitális tanrend ideje alatt újra iskolás lehetett. Nagyon sokszor fordult elő, hogy nem a diák, hanem a szülők tudását mértük az online oktatásban.

Nagy problémát jelent Magyarországon a jelenlegi pedagógus hiány is, mely főként a reál területeket érinti. A tanár kollégáknak túlórákat kell vállalniuk, melyre nem kapnak fizetést. A pedagógusfizetések sajnos nem vonzóak a fiatal pályakezdőknek, így még munkába állás előtt el is hagyják a tanári pályát. A túlterheltség az oktatás minőségének romlásához vezethet. Jelenleg nagy a nyugtalanság a Magyarországi tanárság körében. (Vass, 2022) Nem csak a pedagógusok, a szülők is elégedetlenek a jelenlegi helyzettel.

Kutatásunkban arra is kerestük a választ, hogyan vélekednek a tanárok a digitális tanrend körülményeivel kapcsolatban. Volt-e pozitív hozadéka az online oktatásnak? Dolgoztak-e ki új, digitális tanítási módszereket a tananyag átadásához? Hogyan tudták megoldani a számonkérést és értékelést az online térben?

2. A kutatás módszertana

Kutatásunkhoz egy kérdőívet készítettünk, amelyet a közoktatásban dolgozó pedagógusok körében terjesztettünk. A kitöltésére 2022. februárjától 2022. áprilisáig volt lehetőség. A megosztás levelezőlistákon, valamint közösségi médián keresztül történt és elküldtük a Kar gyakorlóiskoláiba is.

Kérdőívünk 25 kérdést tartalmaz, amelyet 3 szekcióba osztottunk. A kitöltéshez körülbelül 5-8 perc volt szükséges. Az első szakaszban általános adatokra kérdeztünk rá. Az egyik kérdésünk arra irányult, melyik generációhoz tartoznak, hogy a kapott válaszokból megtudjuk, volt-e nehézségbeli különbség a digitális tanrendre való átálláskor az egyes generációk között. Egy másik kérdésünk a kitöltő nemére vonatkozott. További kérdések a lakóhely és a munkahely típusára vonatkozott. Találunk-e összefüggéseket vagy eltéréseket az egyes településtípusokon tapasztaltak között. Kíváncsiak voltunk, mely tantárgyakat oktatják, és mely korosztályban tanítanak. Milyen tapasztalatokat vonhatunk le az egyes korosztályokban, vannak-e eltérések a tantárgyak közötti online oktatási módszerekben, lehetőségekben.

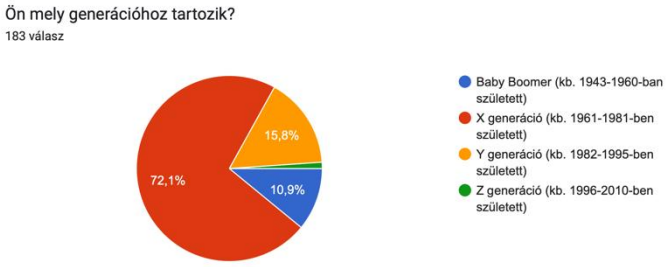
A második szekcióban a kitöltő pedagógusok általános tapasztalataira voltunk kíváncsiak. Menynyire ment könnyen/nehezen az online oktatásra való átállás, mi okozott problémát az átállás során. További kérdések voltak, hogy kitől kaptak támogatást, kihez tudtak segítségért fordulni. Megkér-

deztük továbbá, hogy honnan tartották a tanórákat, valamint milyen forrásból származó eszközökön tudtak dolgozni. Biztosított-e megfelelő eszközöket az oktatási intézmény, vagy sem. Milyen alternatívákkal tudták megoldani az eszközhiányt. A használt internet szolgáltatás minőségével mennyire voltak megelégedve, sikerült-e minőségében is megfelelő órákat tartani az online térben. Kíváncsiak voltunk milyen típusú órákat tartottak. Megtalálták-e a tudás átadásához szükséges megfelelő alkalmazásokat, vagy inkább az előre elkészített prezentációkból tanítottak. Esetleg digitális tananyagokat küldtek ki önálló feldolgozásra. Egy másik fontos kérdés, hogyan tudták a diákokat aktivizálni az online órák ideje alatt. Milyen gyakorisággal használtak online alkalmazásokat, és azok megtalálása, használatának elsajátítása okozott-e problémákat.

A harmadik szakaszban pedig a tanulást segítő alkalmazásokkal kapcsolatban szerzett tapasztalataikról kérdeztünk. Kíváncsiak voltunk, mely alkalmazásokat használták. Megkérdeztük, hogy milyen módszerekkel oldották meg a számonkérést, és ahhoz mely alkalmazásokat tudták használni, valamint mennyire voltak megelégedve a választott számonkérési módjaikkal. Tudtak-e szóbeli feleleteket tartani, vagy csak írásban számoltak be tudásukról. Az utolsó három kérdésre kifejtős válaszokat kértünk. Részletezzék, hogy vannak-e olyan módszerek, alkalmazások, amelyeket megtartottak a jelenléti oktatásban is, és indokolják meg, hogy miért ezeket választották további munkájukhoz.

3. Eredmények

Az űrlapot 182-en töltötték ki. A kitöltők 70%-a nő, 30% férfi, ami nem tér el az országos átlagtól. Sajnálatos dolog, hogy a pedagógusok közt ilyen kevés férfi. A generációs megoszlások azt mutatják, hogy a kitöltők 72%-a az X generáció, 15%-a az Y generáció és 10%-uk a Boomer generáció tagjai. Hazánkban a pedagógus társadalmat az elöregedés fenyegeti, melyet ezek a visszajelzések is alátámasztanak.



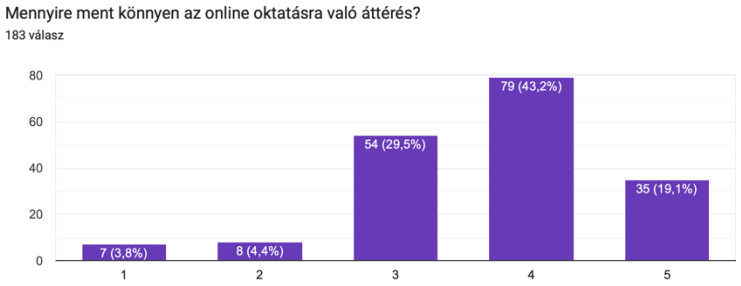
1. ábra: Életkori megoszlás

A település megoszlásban vezető helyen 23%-kal a nagyvárosi kitöltők szerepelnek, majd a fővárosiak 22,4%-kal. Őket követik a kisvárosi válaszadók 21,3%-kal, majd a középvárosi kitöltők 17,5%-kal, utánuk a községek kitöltői 10,9%-kal, és végül a legkevesebb választ 4,9%-ban a falvakból kaptuk. A tanított korosztályokat tekintve a következő eredmények születtek. A kitöltők 59%-a középiskolában, 43,7%-a általános iskola felső tagozatán és 22,4%-a általános iskola alsó tagozatán tanít. A százalékos összegzésben a 100%-tól való eltérés abból adódik, hogy vannak olyan pedagógusok, akik több korosztályban is tanítanak. Például általános iskola felső tagozatán és középiskolában is, vagy általános iskola alsó és felső tagozatán is. A tanított tárgyak száma meghaladja a harmin-

cat. A legtöbben közülük informatikát 47,5%, matematikát 35,5%, vagy magyar nyelv és irodalmat 19,1% oktatnak.

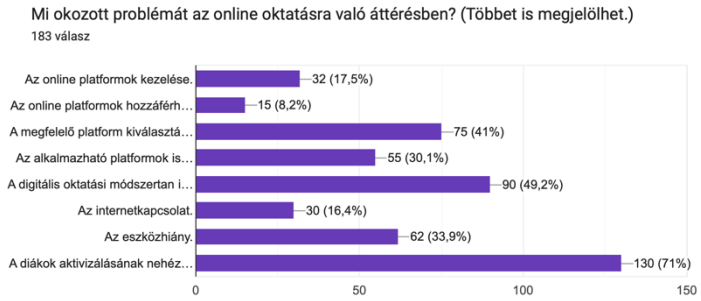
A kérdőív kitöltésével az alábbi eredményekre jutottunk. A válaszadó tanárok többsége, 60,7%-uk otthonról tanított. Az ezt követő helyen az általában otthonról, néha az intézményből tanítók állnak 25,7%-kal. Őket követik az általában iskolából, néha otthonról dolgozók 6,6%-kal, majd a fele-fele arányban otthonról és iskolából oktatók 6,6%-kal. A kitöltőknek mindössze 1,1%-a tanított kizárólag a munkahelyéről. A tanórák megtartásához, felkészüléshez a legtöbben saját, és a munkáltató által biztosított eszközökön dolgoztak 53,6%. Csak saját eszközökön 31,1%-uk oktatott, és csupán 15,3%-uknak biztosította teljes mértékben a munkáltató az eszközöket, ami már egy régebbi problémaforrás, hiszen elvárás a pedagógusok felé, hogy használják az IKT eszközöket, és a digitális megoldásokat az oktatásban, ám ennek teljesítéshez nincsen elegendő eszköz biztosítva.

Az online oktatásra való áttérés könnyen sikerült 62,3%-uknak. Közepesen nehezen ment 29,5%-nak, ami egy könnyű átállásnak mondható. A legtöbb nehézséget a diákok aktivizálása okozta 71%-ban. Az ezt követő probléma a digitális oktatási módszerek hiánya 49,2%-ban jelentette, majd a megfelelő platformok kiválasztása 41%-ban. További problémákat mutatnak az eszközhiány 33,9%, a megfelelő platformok ismeretének hiánya 30,1%, az online platformok kezelése 17,5%, a megfelelő internet kapcsolat hiánya 16,4%, valamint az online platformok hozzáférhetősége 8,2%.



2. ábra: Áttérés nehézsége

Az online oktatás során a legnagyobb kihívást a diákok aktivizálása jelentette. A kitöltők 47,5%-a a közepesen nehezen megoldható válaszlehetőségre voksolt. Második helyen az oktatási módszerek hiánya szerepel, melyre 49,2%-uk szavazott. A probléma enyhítésére különféle csoportokat hoztak létre például Facebookon, és ott osztották meg tapasztalataikat egymás közt, egymást megsegítve. Segítséget a legtöbben kollégáktól 63,9%, az intézménytől 44,8%, internetes fórumokról 38,8%-uk kaptak. További lehetőségek voltak tutoriálokból való tanulás 24,6%, családból kapott 23,5%, valamint a diákoktól 14,2% kapott segítséget az alkalmazások kiválasztásához, használatához.



3. ábra: Kihívások az online oktatásban

Leggyakrabban a Google Drive, Meets, MS Form, Redmenta és Learning Apps alkalmazásokat használták a megkérdezettek. Informatika és matematika tanárok többsége pedig a Google Form, Redmenta, Kahoot és MS Form alkalmazták a tanórák megtartásához.

4. Összegzés

A kérdőívre kapott válaszokból kiderült, hogy többségük megtart néhány online alkalmazást a digitális tanrend lejárta után is. A pozitív tapasztalatokat, jó gyakorlatokat továbbra is megosztják egymás közt a létrehozott csoportokban.

A legtöbben a módszertani felkészítés hiányát, és a munkáltató által biztosított eszközök hiányát tartják a legnagyobb problémának. Ez a probléma azonban nem csak a digitális tanrend ideje alatt volt jelen, hanem már korábban is felmerült a jelenléti oktatás ideje alatt. A felkészüléshez és az adminisztrációhoz is szükségük van digitális eszközökre (laptop, táblagép, asztali számítógép stb.).

Komoly igény fogalmazódott meg olyan módszertani képzések iránt, melyekben az online oktatási alkalmazások tanórai és tanórán kívüli felhasználhatóságáról szólnak. A pedagógusoknak a diákok online aktivizálásával problémák voltak. A kutatás szerint némi korreláció fedezhető fel a diákok aktivitása és a pedagógusok gyakorlottsága között. Minél jobban tudták a tanárok az appokat használni, annál jobban tudták a diákokat is bevonni a tanórába.

Hazánkban a pedagógus társadalom generációs összetételéből adódóan, ahol kevésbé vannak jelen a fiatalabb Z generáció tanárai, különösen problémás helyzetben volt a digitális tanrendre történő átálláskor. Az idősebb generáció számára a digitális eszközök, online alkalmazások használatának megtanulása jóval több időt és energiát igényelt, sőt, voltak olyanok is, akik szinte tiltakoztak a használatuk ellen. Sokukban az IKT eszközök használata és a digitális oktatás fogalma egy és ugyanaz, annak ellenére, hogy ez nem teljesen igaz. Ők leragadtak azon a szinten, hogy a digitális oktatáshoz elég egy projektoron kivetített diasort alkalmazni, ám attól, hogy használunk egy kivetítőt, vagy elküldünk e-mailben egy pdf dokumentumot feldolgozásra, még nem beszélhetünk digitális oktatásról.

A digitális tanrendre való átállás során a kérdésekre adott válaszok alapján nem mutatható ki összefüggés a település fajtája és az áttérés nehézsége közt. Körülbelül azonos szinteket jelöltek a falvakban élő tanárok úgy, mint a fővárosban tanítók.

A válaszadó pedagógusok több, mint fele otthonról tartotta meg tanóráit, melynek egyik oka lehetett, hogy az intézményvezető ezt megengedte, másik oka pedig a megfelelő sávszélesség hiánya. Az otthonról dolgozó tanároknak több, mint 80%-a volt megelégedve az internet szolgáltatással, míg az iskolából tanítók körülbelül két harmada. Úgy gondoljuk, nagy szükség van a hazai oktatás fejlesztésére minden területen. Új módszerek fejlesztése vált szükségessé, melyben az online alkalmazásokon keresztül történő oktatási lehetőségekről, és a digitális eszközökkel történő problémamegoldásnak nagyobb szerepet kell kapniuk.

Az államnak jobban kellene támogatnia az intézményeket eszközökkel is. A válaszokból kiderült, hogy csak egy hetedüknek tudta az eszközöket teljes mértékben az intézmény biztosítani. Ne csak elvárás legyen a digitális adminisztráció és a digitális oktatás. Kőkorszaki körülmények között, és elavult eszközökkel a tanárok nem lesznek képesek a mai nemzedéket megtanítani korunk elvárásainak megfelelően. Figyelembe kell venni a technika rohamos fejlődése miatt megváltozott elvárásokat az oktatásban is. A diákokat a körülöttünk lévő világ feladatainak ellátására kell felkészítenünk. A tananyagokat a mai szakmák elvárásainak megfelelően kellene megreformálni. A digitális eszközök használatával az oktatásban is új dimenziókat nyithatunk meg.

Irodalom

1. Bernát, Péter; Szlávi, Anna. (2020). Lányok az Informatikában – Módszerek és Esettanulmányok a Nők Eredményesebb Bevonására. InfoDidact.
2. DIGIVID (2022). DIGIVIDget (tugraz.at)
3. N. Kollár, K. (2021). Az online oktatás tapasztalatai és gyakorlata a pedagógusok nézőpontjából, Iskolakultúra, 23-53.
4. Lévai, D. (2014). A pedagógus kompetenciái az online tanulási környezetben zajló tanulási-tanítási folyamat során, ELTE Pedagógiai és Pszichológiai Kar, ELTE Eötvös Kiadó, Budapest, Pest, Magyarország. https://www.eltereader.hu/media/2015/03/Levai_D_A-_pedagogus_kompetenciai.pdf (utoljára megtekintve: 2022.11.23.)
5. Papp-Danka, A. (2014). Az online tanulási környezettel támogatott oktatási formák tanulásmódszertanának vizsgálata, ELTE Pedagógiai és Pszichológiai Kar, ELTE Eötvös Kiadó, Budapest, Pest, Magyarország. https://www.eltereader.hu/media/2015/01/Papp_Danka_A_Online_tanulasi_READER.pdf (utoljára megtekintve: 2022.11.23.)
6. Válaszonline (2022). https://www.valaszonline.hu/2022/10/19/oktatas-tuntetes-horvathne-vass-ildiko-publi/ (utoljára megtekintve: 2022.11.23.)

Tehetség targeting

Sarmasági Pál

psarmasagi@inf.elte.hu
ELTE IK

Absztrakt. Minden diáknak joga van a képességének megfelelő szintű oktatáshoz, de ez sem a hazai, sem a nemzetközi gyakorlatban nem valósul meg. A közoktatás a Gauss görbe közepére koncentrál, az ettől való eltérés egyik irányban sem szerencsés, különösen az informatikában tehetséges diákok esetén. A megfelelő szintű oktatás, tehetséggondozás első lépése pedig az érintett diákok keresése, felismerése. Ennek a célnak a támogatására mutat a cikk egy az üzleti életben alkalmazott komplex megközelítést. A célcsoport meghatározása, annak dinamikus kezelése és a mindezekhez szükséges mérések egységes koncepcióba rendezése egy folyamatban lévő kísérlet elméleti hátterét képezi.

Kulcsszavak: Informatikai tehetség, formatív értékelés, algoritmikus gondolkodás, targeting

1. Bevezető

A 2020-ban bemutatott új nemzeti alaptantervben az informatika oktatás hangsúlyosabban szerepel. [1] A digitális kultúraként megjelent tantárgy oktatása során nagyobb figyelmet kap az informatika felhasználása a többi tantárgy tanulásában, a különböző tudományterületek támogatásában. Ezt erősen indokolja a digitális világ, ezen belül az informatika fejlődése, változása, ami a diákok szűkebb rétegét érdeklő és érintő számítástudományból az élet minden területét lefedő kulturális területté vált. A matematikához hasonlóan az informatika esetén is felerősödött annak segédtudomány jellege, ami minden állampolgártól elvárja az otthonos mozgást a digitális kultúrán belül. Ez már több, mint a digitális írástudás, az EU által néhány évenként frissített European Digital Competence Framework for Citizens vagy röviden DigComp 2.1. néven ismert dokumentum tanulási eredmények formájában írja le az uniós állampolgárok számára jelenleg relevánsnak tartott digitális kompetencia tartalmát. [2] A hazai alaptanterv ezen ajánlásokat követve növelte az óraszámot, és a segédtudomány jelleget erősítő tartalom keresés és kezelés mellett az algoritmikus gondolkodás és számítógépes problémamegoldást is hangsúlyosabban tartalmazza.

A megnövekedett óraszámok és a kibővített tananyagok kezelése során a diákoknak több lehetősége van egy-egy területen esetlegesen meglévő képességüket megmutatni, ezzel párhuzamosan a tanároknak pedig lehetőségük van újabb informatikában potenciálisan tehetséges diákokat felismerni. A tehetségkiválasztás és tehetséggondozás indoklására számos érv ismert, ezek közül a hazai szakirodalomban is népszerű holland pedagógus, Franz J. Mönks egyik állítását idézem: *„Mindent meg kell tenni, hogy minden tanuló, legyen az nagy tehetségű, vagy csekély adottságú, saját képességeinek megfelelően fejlődhessék."* [3/57.o.] Másszóval minden gyereknek joga van ahhoz, hogy képességének és érdeklődésének megfelelő oktatáshoz jusson, ami sajnos nem adatik meg minden diáknak sem a hazai, sem a nemzetközi gyakorlatban. Különösen igaz ez az informatikában tehetséges diákok esetén, akik néha túlszárnyalják tanáraikat egy-egy területen. A megfelelő szintű oktatás, tehetséggondozás első lépése pedig az érintett diákok keresése, felismerése. Ennek a célnak a támogatására mutat a cikk egy az üzleti életben alkalmazott megközelítést.

2. A tehetség terhe

A közvélemény szerint a tehetség egyaránt áldás a szülőnek és a gyereknek. A tapasztalat szerint azonban ez sok esetben inkább teher. [3] A társadalmi elvárásoknak megfelelően a közoktatás a Gauss görbe közepére van felkészítve, a sávtól való eltéréseket (legyen az negatív, vagy pozitív) a rendszer nem képes megfelelően kezelni. A szocializmus éveiben a tehetséggondozást az elitképzéssel azonosították, így hosszú időre elvetették, amikor pedig újra felismerték szükségességét, már a tanárok ellenállásába ütközött. [4] A XXI. században már szervezett keretek között, a Magyar Tehetségsegítő Szervezetek Szövetségének támogatásával próbál a hazai oktatás felzárkózni a nemzetközi szintre, de ezzel egyidőben az Egyesült Államokból elindult mozgalmak, amelyek az akadémiai és a politikai baloldal szimbiózisából alakultak ki, a lemaradók felzárkóztatására helyeznek nagyobb hangsúlyt a tehetséggondozással szemben. Európában mindig erősebb volt ez a megközelítés, így a tehetséges diákokat továbbra is sok esetben magukra hagyják, hiszen ők segítség nélkül is képesek a haladásra, könnyen elérik a tantervekben meghatározott képességek és ismeretek szintjét. [5]

A tehetséggel foglalkozó szakirodalom azonban a fenti gondolatmenettől jelentősen eltérő tapasztalatokról számol be, amit sok szülő és pedagógus megerősít, akik már találkoztak beazonosított tehetséggel. A nem képességüknek megfelelő szintű oktatásban részesülő tehetséges diákok könnyen motiválatlanná, lustává válnak, emiatt gyakran szemtelenek, az órát zavaró magatartási problémákkal küzdő diákok lesznek. Az iskolának és a pedagógusnak is érdeke, hogy a tehetséges diákok ne zavarják meg a tanítást, hanem együttműködők legyenek, miközben ezen diákoknak is biztosítani kell a lehetőséget, hogy képességeiknek és igényeiknek megfelelő oktatásban részesüljenek. [3]

A tehetségirodalomban az oktatás szempontjából elsősorban a tehetségígéretek érdekesek, hiszen valódi, megvalósult tehetségről elsősorban a felnőtt életben beszélhetünk, ott is már egy értékelhető időszakon keresztül fennálló tartós teljesítmény esetén. Angol nyelvterületen szokás megkülönböztetni a kettőt, magyar nyelvben Czeizel Endre szorgalmazta a két tehetségszint megkülönböztetését tehetségígéret, valamint megvalósult tehetség formában. [5] Akár megvalósult, akár ígéretes tehetségről beszélünk, számos tehetségmodell ismert, melyek közös pontjai a következőkben összegezhetők:

- kiemelkedő képességek (intelligencia, kreativitás)
- jó személyiség jellemzők (motiváltság, kitartás)
- jó környezeti körülmények

Természetesen az egyes összetevők részletezésében már nagyon sokféle álláspont megjelenhet. Érdekes azonban, hogy a közös pontok mindegyikére hatással van a közoktatás, az iskola és a diákot tanító tanárok. Optimális esetben kiemelt szerepe van a tanároknak a képességek fejlesztésében, a nevelőmunkájuk során a diák személyiségét is fejlesztik, így megfelelő környezetet biztosíthatnak a tehetséges diákok fejlesztéséhez, fejlődéséhez.

Általános vélemény, hogy a magas intelligencia – aminek méréséhez számos standardizált intelligencia teszt létezik – jól korrelál a tehetséggel. Érdekes tény, hogy az intelligencia tesztekben előforduló kérdések faktoranalízissel történt elemzése alapján különböztetünk meg hét fő tehetségterületet: nyelvi, zenei, matematikai-logikai, vizuális-téri, testi-mozgásos, interperszonális és intraperszonális. [5] Az informatika nem szerepel közöttük, és a digitális kompetenciák felől közelítve nehéz egyértelmű relációt felállítani az informatika egyes területei és az ismert hét tehetségterület között. Az algoritmikus gondolkodás és a rendszerszintű gondolkodás mögött feltételezhető a jó matematikai-logikai képesség, az alkotó képességhez hasznos a vizuális és térlátás fejlett szintje. Az infokommunikációs eszközök, alkalmazások használata sok diák esetén már teljesen automatikus, testi-mozgásos képességre épül, míg az információ keresés és kezelés kompetenciája legalább annyira épít a nyelvi képességekre, mint a matematikai-logikai területre.

Gardner az ismert hét tehetségterületet a 80-as években határozta meg, amikor az informatika még csak éppen beszivárgott a középiskolákba az első személyi számítógépeken keresztül. Nem volt tantárgy, amelyben szervezetten megjelent volna az informatika oktatás, és így a standardizált tesztek sem tartalmaztak olyan kérdéseket, amelyek segítségével a számítógépes gondolkodás kompetenciáit ellenőrizhették. Hasznos lenne újabb intelligencia teszteket fejleszteni, amelyek az elmúlt 40 év tanterv és tananyag változásait követve már a digitális kompetenciák összetevőit is lekérdeznék.

Abraham Maslow munkássága ismert a közgazdaságtudományokban, általában vizsgálta az emberek motivációt. Véleménye szerint az emberek tevékenységét, a tevékenységük mögötti motivációt a szükségleteik határozzák meg. Felállította a szükségletek hierarchiáját, amelynek legalján, mint a legelemibb emberi szükséglet, a fizikai, élettani szükségletek kielégítése áll (étel, ital, ruha, lakhely stb.) Ezt követi a biztonság, a valakihez tartozás és az önbecsülés, elismertség szükséglete. Amikor ezek teljesültek, akkor lehet továbblépni a magasabb szintű szükségletekre, a világ megértése, esztétika és harmónia, végül a legfelső szinten az önmegvalósítás. [6] Maslow szükséglet hierarchiáját nem validálták, sokan vitatják az alkalmazott módszertan miatt. Maslow annak idején kimondottan sikeres, tehetséges embereket vizsgálva állította fel a szükségletek „piramisát", tehát feltételezhetően motivált embereket vizsgált csak, ami arra figyelmeztet, hogy nem csak a sikeres embereket kell megkérdezni. [7] Az üzleti szférában ugyan lehet csak a sikerre fókuszálni, az oktatásban azonban fontos, hogy a kudarcot és annak okait is elemezzük, hogy csökkenteni tudjuk azok arányát.

3. Informatikai tehetségígéretek

Mivel a tehetségmodellek és az elérhető tehetség-tesztek nem fókuszálnak informatikai tehetségígéretekre, fontos kérdés, hogy kit tekinthetünk informatikában tehetségesnek? Aki már óvodás korában programozott? Amennyiben van ilyen gyermek, valószínűleg tehetséges, de ennél általánosabb, mérhetőbb szempontokra van szükség. Természetesen vannak szakmai ajánlások azzal kapcsolatban, hogy mely korcsoportban milyen készségekkel, képességekkel kell rendelkezni, és azon diákok esetén, akik a korcsoportjukhoz képest 2-3 évvel előrébb járnak, feltételezhető a potenciális tehetség. [3][5] A tehetségirodalom mellett számos kutatás [3][8] igazolta, hogy a gyermekek a szülői modellt követve már kisgyermekkorban többletismerettel rendelkezhetnek a szüleik által gyakorolt tevékenységekhez kötődő kompetenciák területén. Az informatika esetén napjainkban egyre több szülő ül otthonában is számítógép előtt, így a modellt követő gyermekek is mielőbb megismerkedhetnek a számítógépekkel. A számítógéppel való ismerkedés motivációját erősíti a számítógépeken megjelenő szórakoztatóipari tartalom, a játékok és a videók, amelyek már nagyobb választási szabadságot és interaktivitást biztosítanak, mint a televízió. Kérdés azonban, hogy a videó nézegetés és a játékok használata potenciális informatikai tehetségre utal, vagy csupán a szórakozáshoz szükséges lépések motorikus rögzüléséről beszélhetünk. Meg kell jegyezni ugyanakkor, hogy a számítógépes szórakozás sok gyermek esetén motivációt jelent a számítógép használat és a hozzá kapcsolódó egyéb ismeretek elsajátítására, így egyfajta belépőt jelenthet az informatikai tehetségígéretek közösségébe.

A szakirodalomban az informatikai tehetségek, illetve tehetségígéretek keresése és felismerése szempontjából számos megközelítés ismert, hogy milyen ismeretekkel kell rendelkeznie a diákoknak, mely területeken kell kiemelkedő képességgel rendelkezniük. Ionica-Ona „technológiai tehetségről" ír, akik *„Az informatika és műszaki ismeretek területén kiemelkedő teljesítményt nyújtanak, vagy kiváló gondolkodással, képességgel rendelkeznek, amit a mérések és értékelések rendszeresen igazolnak."* [9] Másik megközelítés az EU digitális kompetencia-definíciója, ami a készségek és kompetenciák hierarchikus rendszerét tartalmazza. [2] Ebben közel azonos hangsúllyal szerepel a programozói munkához szükséges algoritmikus gondolkodás, és az alkalmazói ismeretek [10][11]. A nemzetközi szakirodalomban elterjedt továbbá a számítógépes gondolkodás is, amely fogalmát hasonlóan nehéz meghatározni, mint a tehetség fogalmát, mivel ez az egyik legfiatalabb tudományterület és tudásterület, és fejlődése rendkívül gyors. [12] Kiemelhető, és tesztekkel mérhető azonban a számítógépes gondolkodásnak főbb

összetevői, így a feladatok részekre bontása, a mintafelismerés, az absztrakció, általánosítás, valamint az algoritmikus gondolkodás. Meg kell még említeni Nardelli 3 pillérre épülő meghatározását, melyek az algoritmus, a programozási nyelv és egy ágens vagy gép, amely képes a kapott utasításokat megvalósítani. [13] Ez a három pillér magában foglalja a digitális kompetenciákon belül fontos problémamegoldó készséget.

Az informatikai tehetségígéretek beazonosításához így nem tudunk egyetlen jól bevált tesztet alkalmazni, hanem a többféle megközelítéshez tartozó tesztek alkalmazásával nyerhetünk ismeretet az egyes diákok adott informatikai területen meglévő ismeretükről, kompetenciájukról.

4. Digitális kultúra oktatás

A Magyarországon jelenleg hatályos NAT2020 az EU kulcskompetenciáit figyelembe véve a digitális kultúra tantárgy mellett majdnem minden tantárgy tanulásához ajánlja az informatikai eszközök használatát, hogy a diákok a megfelelő digitális kompetencia szintet elérjék. [1] A matematika tantárgy keretei között az 1-4 osztály végére a matematikai képességeket is fejlesztő számítógépes játékok és programok ismeretét írja elő a matematikai problémák számítógépes megoldása mellett. A matematikai képességet fejlesztő játékok használata még a következő 10-14 éves korcsoport számára is tantervi előírás. A digitális kultúra tantárgy 3. osztályban indul, amikor elsődleges cél az eszközhasználat elsajátítása mellett a megfelelő attitűd kialakulása, amit alkotómunkával, rajzoló programok használatával is segítenek. Ezen túl a korcsoportnak megfelelő robot eszközök vezérlése során a kódolás alapjaival is megismerkedhetnek. Felsőtagozaton már az elterjedt irodai alkalmazások használata és a robotok programozásán keresztül az algoritmizálás folyamatának megismerése a cél. A középiskolába lépő diákok így az EU digitális kulcskompetencia által meghatározott főbb területeken már rendelkeznek akár mérhető ismeretekkel is.

Mivel nagyon sok diák a játékok felől ismerte meg az informatikát, sokan örömmel és motiváltan mélyültek el benne, ezt a lendületet az oktatásban is ki kell használni. Másrészt vannak olyan diákok, akik már elveszítették motivációjukat, vagy még fel sem ismerték. Az új digitális kultúra tantárgy más tantárgyak tanításának és tanulásának támogatásához is javasolja az informatikai eszközök használatát. Ennek segítségével más tárgyak elkötelezett tanulói esetén is kialakulhat egy motivált érdeklődés az informatikai feladatok irányába.

A tanórai keretek között természetesen törekedni kell arra, hogy minél több diák aktív résztvevője legyen az órának, mindenkinek olyan feladatot tudjon adni a tanár, ami érdeklődési körének és képességeinek megfelel, és képes felkelteni a kíváncsiságát. Az ilyen feladatmegoldások során van lehetősége a tanárnak megfigyelni az egyes diákok problémamegoldó készségét, feladatmegoldási képességét, ezeken keresztül gondolkodásukba is betekintést nyerni. Azokat a diákokat, akik vagy gyorsabban készülnek el az átlaghoz képest, vagy egyedibb, érdekesebb megoldásokat mutatnak fel, mint a többiek, érdemes meghívni szakköri foglalkozásra, amely során több lehetősége van egy tanárnak a potenciális tehetségek felismerésére és a tehetségek gondozására.

5. Statisztikai megközelítés

A tehetségfelismerésnek vannak ismert és kevésbé ismert nehézségei, ezek közül kiemelek néhányat:

- A létező tesztek segítenek felismerni a tehetséges diákokat, de egy teszt önmagában, vagy akár tesztek sorozata, nem feltétlenül elegendő a tehetséges diák felismerésére.
- A tehetséges diákok kiválogatásában pszichológiai vizsgálatok is segíthetnek, de hasonlóan a tesztekhez, a vizsgálatok is egy-egy időponthoz, alkalomhoz köthetők. Ez nem mindig elegendő a megfelelő képesség és teljesítmény felismerésére.

- A szaktanárok az előbbiekkel szemben folyamatosan megfigyelhetik diákjaikat, azonban többnyire olyan feladatokat kell kiadni a tanulóknak, amelyeket az átlagos képességű diákok is képesek megoldani, így ismételten csökken az esély a kiemelkedő képességű diákok felismerésére.
- Jó lehetőséget kínálnak a szakkörök és külön foglalkozások, ezekbe azonban érdeklődés és motiváció alapján jelentkeznek a diákok, ami ismételten csökkenti a potenciális tehetségek felismerésének a lehetőségét, hiszen gyakran kimaradnak a nem megfelelő önismerettel, önbizalommal rendelkező diákok.

A visszalépéses keresés algoritmusának alapján, a nehézségek kiküszöbölését félretéve, lépjünk vissza és keressünk egy másik lehetséges utat. Induljunk el onnan, hogy egy középiskolában hány diák lehet tehetséges? Ismert, hogy a középiskolák között is jelentős eltérések vannak, a legtöbb középiskola törekszik a minél jobb képességű diákokat felvenni, és képességek tekintetében homogén osztályokat létrehozni. [14] A matematikai statisztika alapján ezekben az esetekben is felrajzolható egy Gauss görbe, a homogén osztályokban ugyanúgy megkülönböztethetőek bizonyos területeken az osztályátlagnál jobb, és annál gyengébb tanulók is.

A különböző tehetségmodellek megalkotói szintén gyakran adtak közelítő becslést a modelljükhöz, hogy az adott korcsoport hány százaléka lehet tehetséges az adott modell alapján. Mivel a különböző szakemberek által megalkotott különböző modellek más-más tehetségdefinícióval és más-más jellemzőkkel dolgoztak nem meglepő, hogy a korcsoportban feltételezhetően tehetséges diákok arányát is különbözően becsülték. Az intelligencia teszteket tehetségmérésre használó Terman (1925) szerint a populáció felső 1%-a lehet tehetséges; Robinson (2005) szerint a felső 1–3%; Brody és Stanley (2005) a felső 3%-ot véli. Freeman (2005) a felső 5–10%-ot, Gagnè (2005) a felső 10%-ot. Gordon és Brigdlall (2005) szerint a populáció felső 15%-a tehetséges, végül Renzulli (2005) a populáció felső 15–20%-ában látja ezt. Az eltérő definícióknál míg Terman és Robinson specifikusabb, addig Renzulli inkább általánosabb modellt követett, így hasznosabbnak tűnik a diákok 15-20%-ra fókuszálni, mint sem 1%-ra. [5] Ez utóbbi nehezen értelmezhető olyan intézményben, ahol évfolyamonként kevesebb, mint 100 diák tanul.

A megfelelő arány kiválasztásában segíthet egy a közgazdaságtudományban alkalmazott szabály, amit Vilfredo Pareto alkotott meg 1906-ban. Megfigyelései alapján az egyenlőtlen vagyoneloszlásra alkotta meg a 80-20-as szabályt, miszerint a megtermelt javak 80%-a a társadalom 20%-hoz kerül. Joseph Juran (1940) hasonló megfigyelésre jutott a minőségügy területén amikor felismerte, a problémák 80%-át az elkövetett hibák 20%-a okozza. [15] Később a Pareto-elv teljesen általánossá vált a közgazdaságtanban és a menedzsmenttudományban. A forgalom 80%-t a vevők 20%-a hozza, illetve a hirdetések 20%-a hozza a forgalom 80%-át.

A Renzulli féle tehetségmodell és a Pareto-elv figyelembevételével érdemes a diákok 20%-át potenciális tehetségnek tekinteni. Mely diákokat soroljuk a 20%-ba és mely diákokat a 80%-ba, illetve ezek a választások mennyire végérvényesek kérdésében szintén egy a közgazdaságtanban alkalmazott módszert érdemes követni.

6. Targeting az iskolában

Az üzleti világban a marketing feladata feltérképezni és elkötelezetté tenni a potenciális vevőket, akik a célcsoportot képezik. A célcsoport meghatározása a célzás, angol szóval a targeting. A targeting során egy fontos szempont a Pareto-elv alkalmazása. Elsődleges feladat az ügyfeleknek azon 20%-nak a megtalálása, akik a forgalom 80%-át generálják. Az üzleti világban a legtöbb területen ez könnyen meghatározható a forgalmi adatokból. Az aktuális, illetve a megelőző időszak forgalmi adatai mellett célszerű ismerni a célcsoport potenciálját, hogy az ismert forgalom tovább növelhető-e? Ezen információk alapján módosítják marketing tevékenységüket, üzenetüket a vállalkozások, fokozzák, vagy módosítják, esetleg más ügyfélre csoportosítják erőforrásaikat.

Az oktatásban a differenciálást hasonló célból alkalmazzák, a különböző képességű és érdeklődésű diákoknak más-más feladattal lehet fenntartani az érdeklődésüket, sikerélményre serkenteni őket. A differenciálás alapját legtöbb esetben a tanár órai megfigyelései és a diákok eredményei képezik. A differenciálás hatékonyabb alkalmazásához hasznos információt szolgáltathat az üzleti életben alkalmazott targetálás, amely a lehetőségekre, az ügyfelek potenciáljára is figyel. A hasonlóságok és különbözőségek figyelembevétele mellett lehet kialakítani a modellt, amely alapján vizsgálható a targetálás alkalmazása az oktatásban.

A potenciál, az egyes diákok tehetségígéretének kérdése a közoktatásban általában ismeretlen változó. Az üzleti forgalommal párhuzamba állítható tanulói teljesítmény pedig ingadozhat. A pontosabb modellezéshez az üzleti élet egy szűkebb szeletét választhatjuk, ahol az iskolai környezethez hasonlóan szintén korlátozott információk állnak csak rendelkezésre. Ilyen iparág a gyógyszeripar, ahol a piaci viszonyok sajátosságai miatt szigorú szabályozás védi a forgalmat generáló, így a célcsoportot képező orvosok valós potenciálját. A vényköteles gyógyszerek szabadon nem reklámozhatók, a gyógyszertárak forgalmi adatai csak járási (korábban kistérségi) szinten elérhetők, így sem a konkrét betegszám, sem a felírások száma orvos szinten nem elérhető adat. A gyógyszeripari cégek munkatársainak így a felírást végző orvosok potenciálját indirekt módon kell megbecsülni. Az értékesítő képviselők feladata a területükön praktizáló orvosok megismerése rendszeres látogatások alkalmával, hasonlóan ahhoz, ahogy a tanárok is megismerik a rájuk bízott diákok képességeit, tulajdonságait a tanórák tevékenységei során. [16]

A célcsoport kialakítás, a targetálás két fő kategória alapján történhet, az egyes kategóriákon belül pedig több szempont is alkalmazható. A két fő kategória a potenciál, és a lojalitás (vagy elkötelezettség), mindkettő felismerhető és alkalmazható a középiskolai tehetséggondozásban.

6.1. A potenciál

A potenciál, esetünkben a tehetségígéret fogalma nehezen definiálható, de a különböző tehetségmodellekkel jól körülírták. Egyszerűen megfogalmazva azokat a diákokat tekintjük tehetségígéretnek, akik valamely képességterületen megfigyelhető, az átlagos szintet meghaladó teljesítményre képesek. A tehetségígéret felismeréséhez mérésekre van szükségünk és a mérési eredmények kielemzésére. Fontos hangsúlyozni a többesszámot, nagyon sok tehetségígéret a megismételhetetlen fontos mérés terhe alatt leblokkol és alulteljesít. [17] Másfelől, nem áll rendelkezésünkre olyan teszt, ami minden kétséget kizáróan kimutatja a tehetséget. Hasonlóan a gyógyszeriparhoz, több oldalról közelítve, különböző mérések és megközelítések összesített eredménye segít jó becslést adni egy adott orvos potenciáljára.

Az egyik fontos mérendő terület a digitális kompetenciák. A mérések során fontos, hogy ne csupán a szummatív értékelésre alkalmazott teszteket vegyük figyelembe, hanem akár havi szinten megíratott, formatív értékelésre használt teszteket is alkalmazzunk. Sok diák a megmérettetést stresszhelyzetként éli meg, és alulteljesít. A formatív jelleget erősítendő célszerű vetélkedő jellegű, játékos feladatokat, teszteket megíratni a diákokkal. Közismert és népszerű a Kahoot, mint környezet [18][19], érdekes és játékos feladatok találhatók az e-Hód (Bebras)[20] illetve a CS-Unplugged weboldalain [21]. A tesztek során figyelni kell arra, hogy legyenek egyszerűbb feladatok, amit minden diák meg tud oldani, hogy ne váljanak motiválatlanná az esetleges kudarc hatására. Ugyanakkor nehezebb feladatokat is el kell helyezni ezekben a tesztekben, amelyek helyes megoldása kimagasló képességet igényel. Ilyen feladatok találhatók az OKTV versenyek archívumában [22], illetve a nemzetközi versenyek weboldalain. Ezen mérések eredményei számszerűsíthetők, és összegezhetők.

A kompetenciák gyakori mérése mellett alkalmanként egyéb képességek mérése is hasznos lehet. Így például egy általános intelligencia teszt kitöltése jó forrás a tehetségkiválasztáshoz. Az általános intelligencia teszten mért magas intelligencia hányados ugyanis gyakran utal valamely területen kiemelkedő képességre, tehetségre. Ez természetesen ritkán utal informatikai tehetségre, de a digitális

kultúra tanórákon egy számítógépekkel felszerelt tanteremben van mód arra, hogy a diákokkal kitöltessünk egy online intelligencia tesztet.

A tehetségmodellek az általános intelligencia mellett fontos szerepet tulajdonítanak a kreativitásnak is a tehetség szempontjából. A kreativitás szintén nehezen mérhető, de vannak elérhető online tesztek. Másfelől a programozási feladatok sajátsága, hogy több jó megoldás létezik. Így a kreativitás mérésének egyik módja, hogy a digitális kultúra tanórák programozási feladatainak a megoldásait átbeszélve kiválaszthatók a kreatív ötletek, és az azokat felvető, megvalósító diákok.

A kreativitás és az intelligencia mérés eredménye is számszerűsíthető, így ezen eredmények a digitális kompetencia egyes összetevőit mérő tesztek során gyűjtött pontokhoz hozzáadhatók. Míg a tantárgyi tudást és kompetenciákat akár heti rendszerességgel mérhetünk, különösen formatív értékelés céljából, addig az általános intelligencia és a kreativitás mérése évente egyszer is elegendő. Természetesen ezen képességek is változnak a középiskolai tanulmányok során, de a tehetségígéret mérés szempontjából ezek a változások általában nem jelentősek, a különböző időpontban mért értékek csak kisebb különbséget mutatnak.

6.2. Az elköteleződés

A célcsoport meghatározás másik dimenziója, kategóriája az elköteleződésen alapul. Egy felismert és beazonosított tehetség esetén is előfordulhat, hogy a diákot nem érdekli az adott terület, nem óhajt vele foglalkozni. Jellemző eset, amikor a diák az informatika mellett matematikából is tehetséges, és inkább a matematikával foglalkozik. Sokan a számítógép használata során tanulnak meg angolul, játékprogramokból, és közülük lehetnek olyanok, akik az informatikát továbbra is csak egy segédeszköznek tartják, és az angol nyelvtanulás irányába kötelezik el magukat. A diákok környezete is csökkentheti az elköteleződést, akár a kortárs csoportok, akár a szülők, vagy egy másik szaktanár, aki a diákot más irányba tereli. Az elköteleződést csökkentheti a bizonytalanság, önbizalom hiány, vagy a személyiséghez kapcsolódó tulajdonságok, mint például a lustaság. Így a tehetségígéretet jelentő potenciál mellett fontos figyelni az elköteleződésre, annak mértékére, és annak indokaira is.

Az elköteleződés szintjét növeli a pozitív megerősítés, a versenyeken elért jó eredmények és a támogató légkör. Hatással van rá, illetve kölcsönhatásban áll a diák motivációjával. Ennek vizsgálatához használható a Maslow féle szükséglet piramis. Többgyermekes családokban élő diákok még jó anyagi háttér esetén is kézzelfogható javakkal jobban motiválhatók, miközben másokat az elismerés, a dicsőség motivál.

Az elköteleződés mértékének meghatározása nehezebb, mivel egyáltalán nem, vagy csak nehezen mérhető, inkább kvalitatív összetevői vannak, mint kvantitatívak. Leginkább a tanári megfigyelésre lehet támaszkodni, de vannak ismert, jól behatárolt kvalitatív jellemzők, amelyek mérhetők, és eredményük adatot szolgáltathat az elköteleződés mértékéről, vagy annak felismerési nehézségeiről.

A különböző képességek mellett egyre gyakrabban felmerül az interperszonális, szociális készségek, idegen szóval a soft-skill-ek vizsgálata. Középiskolában kiemelten fontos ezek közül a kommunikációs készségek, az együttműködési készség, és a csapatmunkában való aktív részvétel képessége. Ezek az interperszonális képességek egyrészt mérhetők, másrészt egyszerűen megfigyelhetők az órai munkák során. A képességekkel ellentétben az interperszonális képességek esetén gyakran pont az alulteljesítő diákok igényelnek figyelmet, mint lehetséges tehetségígéretek [23]. Informatikai tehetségek esetén ez különösen jellemző.

Érdemes ezek mellett egy egyszerűbb általános személyiségjellemzők mérését végző tesztet is kitöltetni a diákokkal. Ilyen például a William Martson által kidolgozott DISC teszt, ami az üzleti életben és az emberi erőforrás gazdálkodásban egyaránt elterjedt. Ez a teszt az extrovertált vagy introvertált, feladat vagy kapcsolat orientált személyiségtípus mellett a diákok motivációjában és a szükséges kommunikációs mód megtalálásában is segít [24]. Mivel a személyiség ezen jellemzői a

középiskola végére nagyjából kialakulnak, így ezt a tesztet elegendő lehet egyszer megíratni a diákokkal a középiskolai tanulmányaik során.

A formatív értékelés nem csupán a tantárgyi ismeretek mérésére és visszajelzésére alkalmas, hanem az egyes tárgyak megismeréséhez és alkalmazásához szükséges egyéb kompetenciák esetén is hasznos. A marketingben elterjedt SWOT elemzés mintájára középiskolai tanulók körében is végezhetünk az adott diák erősségeit, gyengeségeit, lehetőségeit és veszélyeit vizsgáló SWOT analízist. Előre meghatározott szempontrendszert érdemes alkalmazni, a diákoknak pedig határozottan dönteniük kell, hogy az adott terület erősség vagy gyengeség a belső tényezők esetén, illetve lehetőség vagy veszély a külső tényezők közül. Belső tényezőknek elsősorban kompetenciákat célszerű vizsgálni, köztük olyan képességeket is, mint az önálló tanulás vagy az időbeosztás képessége. A külső tényezők a környezet hatásait vizsgálják. Egyik előnye ezen elemzéseknek, hogy emlékezteti, informálja a diákokat a szükséges kompetenciákra, illetve az életükre és tanulmányaikra hatással lévő környezeti jellemzőkre. A belső tényezők a tanulási folyamat eredményeként fokozatosan javulhatnak, a külső tényezőkre bár nincs hatása a diákoknak, de felismerve a veszélyeket felkészülhetnek azok kivédésére. Egy ilyen vezérelt SWOT elemzést is érdemes félévente elvégeztetni a diákokkal. [25]

A DISC személyiségteszt például az adott személy motivációit segít feltárni, ennek ismeretében felismerhető, hogy a diák ténylegesen nem elkötelezett, motiválatlan, vagy csupán introvertált személyiségénél fogva nem tudja megfelelően kifejezni érdeklődését. A SWOT elemzés külső tényezői is segítenek képet kapni a diák motivációiról és elkötelezettségéről az alapján, hogy mit tekint lehetőségnek és mit veszélynek. Aki a tanórán kívüli foglalkozásokat és a versenyeket lehetőségnek tekinti, a számítógépes és videójátékokat pedig veszélynek, azok esetén feltételezhető az elköteleződés. Ezen információkat a tanórai megfigyeléssel együtt alkalmazva jól behatárolható az egyes diákok elkötelezettsége.

A potenciálhoz hasonlóan célszerű lenne az elkötelezettséget is számszerűsíteni, a fenti szempontrendszer ezt azonban csak limitáltan teszi lehetővé. Egy három szintű értékeléshez azonban elegendő adattal szolgálnak, amelyben a következő értékek lehetnek: elkötelezett vagy pozitív, nem meghatározható vagy semleges, elutasító vagy negatív.

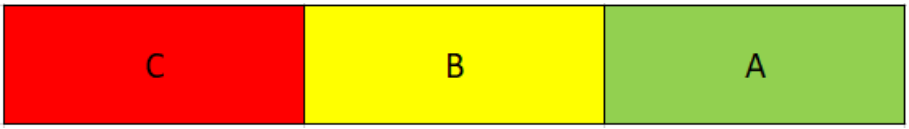
6.3. Besorolás

Az üzleti világban a célcsoport képzés (targetálás) utolsó lépese az ügyfelek szegmentálása, „osztályozása". Az ügyfelek besorolását, kategorizálását nagyon sok esetben egyetlen szempont, a potenciál alapján végzik el. Akinél nagy forgalom van, az fontos, akinél kicsi, az kevésbé, vagy egyáltalán nem számít. Itt fontos megjegyezni egy alapvető különbséget az üzleti világ és az oktatás között, az utóbbi esetben ugyanis minden diák számít, a besorolással a differenciáláshoz szeretnénk hasznos információt szolgáltatni. A modellnek használt gyógyszeriparban is jellemző a tisztán potenciál alapú megközelítés, de egyre általánosabb a két lépéses, két szempont alapján végzett szegmentáció, ahol a potenciál mellett az elköteleződést, lojalitást is vizsgálják. A két kategória egy derékszögű koordinátarendszer két tengelye, két dimenziója. A vízszintes tengelyen a potenciált, a függőleges tengelyen az elköteleződést jelöljük, az alkalmazott értékek alapján a koordinátarendszernek csak az első negyedét használjuk. A kétdimenziós besorolás esetében is első lépésként a potenciál alapján történik a felírást végző, így forgalmat generáló orvosok besorolása.

Az iskolai alkalmazás során a potenciál korábban bemutatott számítása alapján a diákok különböző összpontszámokat érnek el, miután a különböző tesztek, értékelések eredményeit összegezzük, vagy azok relatív, százalékos értékeit átlagoljuk. Minden tanulócsoport esetén lesz minimum és maximum érték, és ezek között oszlanak el az egyes diákok eredményei. Hasonlóan a közoktatásban gyakran alkalmazott normaorientált értékeléshez a minimum és maximum érték között ki kell jelölni néhány ponthatárt. Az üzleti világban elterjedt gyakorlat két-három határ kijelölésével három-négy szegmenst határoz meg. Ezek megkülönböztetésére az ABC betűit használjuk, így A, B és C, vagy A,

B, C és D csoportba sorolhatjuk egy tanulókör diákjait. A kategóriák száma a tanulócsoportok létszámától és homogenitásától is függ. Kis létszám, illetve homogén csoport esetén elegendő a három csoport, nagyobb létszám, vagy nagyon inhomogén csoport esetén alkalmazható a négy csoport.

A normaorientált értékeléstől eltérően nem az átlaghoz, vagy más referencia értékhez viszonyítva kell beállítani a ponthatárokat, hanem a csoportok létszámát kell figyelembe venni. Az első csoportba, a legmagasabb pontszámmal rendelkező diákokat soroljuk be, akik komoly tehetségígéretnek tekinthetők. Az 5. fejezetben leírtak alapján ebbe a tartományba a diákok kb. 20-25%-át soroljuk. Kis létszámú csoportok esetén ez növelhető, például egy 10 fős tanulócsoportból három diák is része lehet a csoportnak, vagy akár négy is, pontazonosság esetén. A középső tartományt kell a legszélesebbre tárni, amibe a diákok kb. 50%-át soroljuk. A további diákok, akik kevesebb pontszámmal rendelkeznek, a harmadik csoportba kerülnek, az első csoporttal közel azonos arányban, követve a Gauss-eloszlásból ismert arányokat.

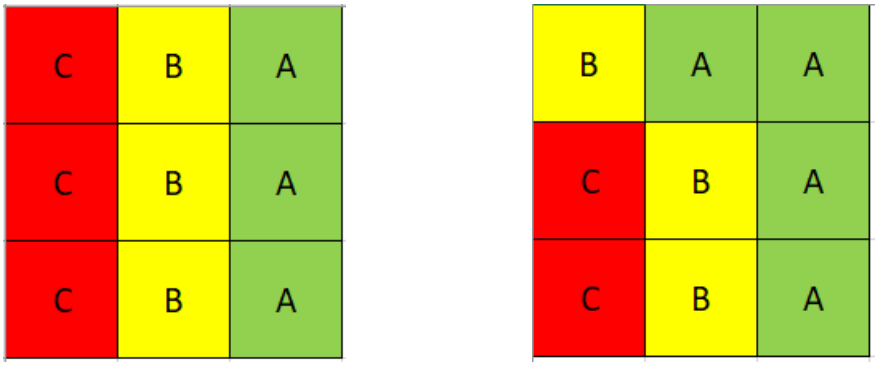


1. ábra: Potenciál skála

A potenciál alapú besorolást finomítja a második szempont, az elköteleződés bevonása a szegmentációba. Az üzleti életben a kétdimenziós besorolás kifinomultabb tervezésre, komolyabb stratégiára utal. Iskolai környezetben szinte elengedhetetlen figyelembe venni a diákok motivációját, szorgalmát. A gyakran csak teljesítmény alapon kalkulált potenciál alapú besorolás alkalmazása az üzleti életben is nagyon leszűkíti a célcsoportot, és nem alapoz az értékesítők képességeire. Ez rövid távon hozza ugyan az elvárt forgalmat, de hosszú távon versenyhátrányt eredményezhet. A közoktatásban szintén szűkíti a célcsoportot, mivel kirekeszti azokat a diákokat, akikben bár meg van a tehetség ígéret, de ezt lassabban, több munkával lehet felszínre hozni. A személyiségjegyek és az interperszonális képességek nehezen számszerűsíthetők, hiszen több különböző szempont szerint értékelik a diákokat, és inkább minőségi, mint sem mennyiségi információt hordoznak. Fontos azonban ezek segítségével kompenzálni a kompetencia tesztek által kalkulált potenciált, amennyiben ez indokolt. Ez természetesen a tanártól több figyelmet, komolyabb szakmai tudást és munkát igényel.

Az elköteleződésnél javasolt 3 fokozatú skála alapján 3 jól meghatározott csoport alakul ki. Így a potenciál által kialakított 3 csoporttal kombinálva a diákok összesen 9 csoportba sorolhatók. A második dimenzió célja, hogy a meglévő három csoport határait képlékenyebbé, könnyebben átjárhatóvá tegye. A tanulási folyamat egyik fő mozgatója a motiváció, ami jelentős része az elköteleződésnek. Egy kevesebb pontszámot elérő, de motivált diákkal gyakran komolyabb feladatokba lehet belekezdeni, mint egy jobb képességű, de kevésbé motivált diákkal. Épp ezért a két szempont szerinti besorolás a potenciál alapján alacsonyabb pontszámú, de motivált diákokat is a magasabb szegmensbe sorolja.

A második dimenzió bevezetésével természetesen megváltozhatnak a potenciál alapján kijelölt határok. Ilyenkor célszerű az eleve képlékenynek felvett határokat módosítani, hogy az A,B,C csoportba sorolt diákok aránya minél jobban megközelítse a 25-50-25 százalékos eloszlást.



2. ábra: Két dimenziós besorolás és annak korrigált változata

A mérések folyamatosak, ennek megfelelően az összpontszámok változnak, a harmadik csoportba sorolt diákok is átkerülhetnek az első csoportba és viszont. A középső csoportba sorolt diákok az iskolában zajló oktatási tevékenység, illetve az egyéni tanulás, érdeklődés és motiváció hatására az újabb besorolás készítésekor feljebb léphetnek, vagy ezek hiányában a harmadik kategóriába kerülhetnek.

A besorolás, a különböző szegmensekbe csoportosítás célja, hogy az adott méréseknek és attitűdnek megfelelő foglalkozásban részesüljenek a diákok a tanórákon. Ez egyrészt segíti a tanár munkáját, keretet ad a tantermi differenciálásnak, másrészt a diákok szempontjából is hasznos, mert olyan feladatokat kaphatnak, amelyek képességeikhez igazítva kihívást is jelentenek, ugyanakkor nem megoldhatatlanok, biztosítják a sikerélmény lehetőségét, ami elengedhetetlen a sikeres tanítási folyamathoz.

6.4. Időzítés

Az üzleti szférában a vállalkozások naptári időszakokhoz köthető értékesítési ciklusokban gondolkodnak. Jellemző a negyedéves értékesítési ciklus, de előfordul harmadéves, féléves és éves ciklus is. A vállalati stratégia függvényében gyakran minden értékesítési ciklusban felül kell vizsgálni az aktuális besorolásokat, de évente egyszer mindenképp. A közoktatásra a féléves ciklus a jellemző, így véleményem szerint félévente elegendő a besorolási értékeket felülvizsgálni, az egyes tanulócsoportok besorolásait újra elvégezni. A potenciál és elköteleződés szintje folyamatosan változik, ahogy a formatív értékelések, vagy az esetleges versenyek eredményei a tanár által vezetett táblázatba kerülnek. Az elköteleződés is változhat, ez azonban jellemzően ritkábban. A félév során így elegendő a pontszámokat felvezetni az attitűdben esetlegesen bekövetkező változásokkal együtt. Ezen információkat a legtöbb tanár egyébként is követi, figyeli, így nem jelent többlet terhelést a pedagógus számára. A besorolás elkészítése pedig az összegyűjtött adatok alapján félévente jelent csupán további feladatot, amelynek időigénye a fent ismertetett módszerrel általában nem hosszabb egy óránál.

7. Középiskolai első teszt

A tesztelésre kiválasztott tanulócsoport egy budapesti középiskola 11. évfolyamának informatika tagozatos osztálya, ahol a létszám 22 fő. A korábbi évfolyamokon más tanároktól tanulták az informatikát, így a diákok előzetes ismerete nélkül kezdtük meg a közös munkát.

A diákok a bemutatkozást követően, amelyben kitértek közelgő továbbtanulási céljaikra is, egy DISC tesztet töltöttek ki. Ez egy kiindulási alapot ad a tanár részére, hogy megismerje a diákok főbb viselkedési motivációit. [24]

A diákok korábbi tanulmányainak a felmérését egy Kahoot vetélkedőbe ágyazott, konkrét programozási ismereteket nem igénylő játékos teszt segítette. Ez egy diagnosztikai értékelése a diákoknak, ami segíti a potenciál meghatározását. A félév kezdetén nem érzékelhető az a nyomás, amit a jobb jegy megszerzésének kényszere okoz, a vetélkedős forma motiválja a diákokat a minél jobb helyezés elérésére, így a diákok játékként élik meg a tanár mérését. A vetélkedő környezetben elvégzett rendszeres mérést a diákok rendszeres játéknak élik meg, amit sokszor a tanóra csúcspontjaként élnek meg. A félév első két hónapjában a következő témakörökben írtak tesztet a diákok Kahoot környezetben:

- számítógépek működése, Neumann-elv
- algebrai műveletek és logikai kifejezések
- kettes számrendszer használata
- keresési és rendezési algoritmusok

Az első komolyabb megmérettetés pedig az e-hód versenyen való közös indulás volt november hónap második hetében. Az összegyűlt adatok már elegendőek az első besorolás elkészítéséhez. Először a kompetenciát mérő tesztek összesített értékelése alapján egy potenciál alapú besorolás készült. Az összesített értékelés során azonban figyelembe kell venni az esetleges torzító tényezőket. Különösen a rendszeres mérések esetén nem garantálható, hogy minden diák minden órán részt vegyen, így az esetleges hiányzást figyelembe kell venni. A különböző teszteken eltérő pontszámot lehet elérni, ez is aránytalanságot okozhat. A pontszámok helyett így célszerű relatív értékekkel, az egyes teszteken elért százalékok átlagával számolni. Az egyes tesztek esetén különbség lehet a nehézségi szintekben is, így azokat célszerű nagyobb súllyal beszámítani. Jelen vizsgálat során az e-Hód verseny eredményeit kétszeres súllyal számítottuk, a meg nem írt teszteket pedig kihagytuk a számításból, hogy ne húzzák le indokolatlanul az átlagot.

A táblázat tartalmazza a diákok monogramját, az első diagnosztikus értékelés eredményét, a további Kahoot tesztek, valamint az e-Hód verseny eredményét. A 10 hét alatt megírt 5 teszt eredménye már jó alapot ad arra, hogy folyamatos teljesítményt mérjen, ami szükséges a potenciál számításához. A 22 fős létszám esetén az A kategóriába 4-6 diák bevonása indokolt a Pareto-elv szerint. Az adott mérések eredményénél megfigyelhető, hogy a 4. és 5. diák között nagyobb pontkülönbség van, mint az 5. és 6. diák között. A pontszámok alsó tartományában az 5. és 6. között van nagyobb különbség, de a 4. és 5. is elkülönül. A potenciál alapú besorolást így el lehet készíteni 4 – 13 – 5 felosztásban, amit az elkötelezettség bevonásával még tovább lehet finomítani.

Diák	Neumann	Algebra	Bináris	Keresés	e-Hód	Átlag
HA	53%	100%	89%	87%	87%	84%
BD		85%		80%	82%	82%
ZA	63%	85%	0%	60%	98%	81%
KM			68%	93%	69%	80%
CM	58%	96%	79%		77%	77%
KD	48%	81%	58%	93%	93%	77%
VA	50%	100%	79%	67%	77%	75%
PM	48%	88%	95%	67%	71%	73%
GD	53%	96%	79%	67%	71%	73%
TB	50%	73%	79%	73%	71%	70%
GH	30%	96%	79%	73%	66%	68%

IB	58%	81%	84%	67%	55%	67%
PD	50%		68%	53%	81%	67%
LD	53%	69%	63%		74%	67%
CG	45%	92%	68%	73%	53%	64%
RA	65%		47%	87%	55%	62%
HR	55%	85%	68%	60%	50%	61%
DZ	48%	96%	53%	67%	47%	60%
BJ	40%	58%		67%	61%	57%
PS	45%	88%	74%	53%	31%	54%
PB	50%	73%		40%	49%	52%
MP	58%	62%	32%	40%	47%	47%

1. táblázat: Diákok potenciál alapú besorolása

A diákok tanórai aktivitása megfelelő, az informatika foglalkozásokhoz képest nagyobb csoportlétszám ellenére jól lehet haladni. A tanórai munka, a diákok motivációjának megfigyelése és követése fontos tanári feladat. Célszerű azonban legalább egy-egy apró jellel, plusz jellel és mínuszjellel, esetleg a semlegességet jelölő nullával rögzíteni is az egyes órákon megfigyelt aktivitást. Ez segíti az objektivitást, amit nagyon nehéz megvalósítani a tanári munka során. Általában a potenciál alapú besorolás alapjául szolgáló értékelőtáblázat legmagasabb pontjait elért diákok a tanórákon is látványosan aktívak. Két hónap elteltével 7 diák tartott 0 pontnál, mivel nem mutattak fel mérhető órai aktivitást. Részben az ő jobb megismerésük, objektívebb megítélésük érdekében fontos tesztekkel mért eredményeket felhasználni.

A DISC személyiségteszt alapján a csoport tagjainak a fele introvertált személyiséggel rendelkezik, nem látványos a motivációjuk, elvárják, hogy megszólítsa őket a tanár és kérdezzen tőlük. A tanórai megfigyelés során passzívnak vélt introvertált diákok kaptak egy korrekciós tényezőt, egy pontot az elkötelezettséghez. A csoport másik fele extrovertált, proaktívak, jelentkeznek és gyakran felszólítás nélkül is elmondják a véleményüket. Aki közülük passzív, vagy esetleg ellenséges a tanórákon, náluk maradt a 0 pont az elkötelezettségnél. Az informatikával foglalkozó személyekre általában jellemző az S és C tulajdonság. Ebben a matematika-informatika tagozatos csoportban ez is megfigyelhető, egyetlen olyan diák van, akinek DISC teszt által feltárt viselkedési jellemzőiben sem az S, sem a C tulajdonsága nem meghatározó.

A formatív értékelés részeként, a diákok önértékelési képességének fejlesztésére és az elköteleződés kvalitatív jellemzőinek mélyebb megismerésére október végén elkészítették a diákok saját maguk SWOT elemzését az előkészített sablon segítségével. [25] A SWOT teszt külső tényezői alapján további diákok kaptak plusz egy pontot, akik szerint a verseny, és a délutáni foglalkozások jó lehetőséget kínálnak. Ezek az irányított SWOT tesztek a tanárnak is fontos visszajelzést küldenek, rámutattak, hogy a csoportmunka és az együttműködés terén nagyon rosszul állnak, igénylik a szervezett csoportmunkát a tanórákon.

A DISC és SWOT elemzés segítségével a 7 korábban értékelhetetlen aktivitású diák közül 3 maradt a passzív, motiválatlan kategóriában. Másik három diák esetén normál aktivitást, egy diák esetén mind két teszt alapján korrekciós tényezőt lehetett alkalmazni, így az elkötelezett diákok közé került.

A potenciál alapú besorolást végül az elkötelezettség értékekkel finomíthatjuk. A kiindulás a pontszámok alapján számított potenciál 3 értéke. Amennyiben az elkötelezettség három értéket tartalmaz, a kétdimenziós besorolás egy 3x3-as táblázattal szemléltethető. Első körben az elkötelezettségtől függetlenül alakítjuk át a táblázatot, majd a táblázat felső sorában a legelkötelezettebb

diákokat egy kategóriával előrébb soroljuk a korábban kialakított potenciál skálán, ahol erre van lehetőség. (2. ábra)

A kétdimenziós besorolás alkalmazásával végül a 2. táblázatban látható besorolás készült a korrekciók végrehajtását követően. Látható, hogy 4 diák is az elkötelezettsége alapján került eggyel magasabb kategóriába, közülük hárman az A besorolási értékre.

A módosítást követően, amennyiben a potenciálhoz kialakított határokat változatlanul hagyjuk a 7 – 12 – 3 megoszlás alakul ki, ami eltér az elméletben megfogalmazott 25 – 50 – 25 százalékos értéktől. Viszont az oktatás során az a szerencsés, ha a tanár felfelé húzza, tereli a diákokat, így ez az eltérés didaktikai szempontból hasznos. A táblázat alsó negyedében azonban a könnyebb feladatok jobban motiválhatják a diákokat, így az arányokat is javítandó, a B kategória határát egy kicsit megemelve alakult ki a 7 – 11 – 4 diákot tartalmazó besorolás.

A táblázatot elemezve megfigyelhető egy érdekes, tanárok számára is kihívást jelentő diák, akit a KM monogram azonosít. A diák jó eredményeket ért el a különböző teszteken, elkötelezettsége azonban nulla, órai munkáját, illetve annak hiányát az eddigi tesztek nem magyarázzák. Esetében javasolt egy személyes beszélgetés, jövőbeli terveiről, érdeklődéséről, hogy azoknak megfelelő feladatokkal be lehessen vonni az aktívabb órai munkába, és jó informatikai képességei fejleszthetők legyenek.

Diák	Neumann	Algebra	Bináris	Keresés	e-Hód	Átlag	Korrekció
HA	53%	100%	89%	87%	87%	84%	2
BD		85%		80%	82%	82%	1
ZA	63%	85%	0%	60%	98%	81%	2
KM			68%	93%	69%	80%	0
CM	58%	96%	79%		77%	77%	2 ▲
KD	48%	81%	58%	93%	93%	77%	1
VA	50%	100%	79%	67%	77%	75%	2 ▲
PM	48%	88%	95%	67%	71%	73%	2 ▲
GD	53%	96%	79%	67%	71%	73%	1
TB	50%	73%	79%	73%	71%	70%	1
GH	30%	96%	79%	73%	66%	68%	0
IB	58%	81%	84%	67%	55%	67%	1
PD	50%		68%	53%	81%	67%	1
LD	53%	69%	63%		74%	67%	0
CG	45%	92%	68%	73%	53%	64%	1
RA	65%		47%	87%	55%	62%	1
HR	55%	85%	68%	60%	50%	61%	1
DZ	48%	96%	53%	67%	47%	60%	1
BJ	40%	58%		67%	61%	57%	1
PS	45%	88%	74%	53%	31%	54%	1
PB	50%	73%		40%	49%	52%	2 ▲
MP	58%	62%	32%	40%	47%	47%	1

2. táblázat: Diákok besorolása potenciál és elkötelezettség alapján

8. Összegzés

A targetálás és a célcsoport besorolása elsősorban egy kiegészítő eszköz, ami segíti a következő időszak oktatási tevékenységét, a tantermi differenciálást és a tehetséggondozást. Ha nem is teljesen személyre szabott, de homogénebb csoportokra igazított tanítást tesz lehetővé. A folyamatos mérés, mint rendszeres formatív értékelés nem csak a diákoknak, de a tanárnak is nagyon hasznos visszajelzéseket ad a diákok haladásáról, a nehezebben érthető tananyagokról.

Az A besorolású diákoknál célszerű alkalmazni a tehetséggondozásban elterjedt, informatika és digitális kultúra tantárgyban is alkalmazható gyorsítást és gazdagítást. [26] A tehetség fejlesztéséhez és felismeréséhez is gyakorlás, különböző nehézségű feladatok megoldása szükséges, mind egyénileg, mind csoportosan, például projekt munkában. Fontos azonban, hogy a diákok sikerélményben részesüljenek a feladatmegoldásaik során, aminek előfeltétele a differenciált, testre szabott feladatokkal szervezett tantermi munka. Ennek támogatására alkalmas az üzleti életből átvett targetálás és besorolás.

Irodalomjegyzék

1. Magyar Közlöny: *A Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról szóló 110/2012. (VI. 4.) Korm. rendelet módosításáról*, Budapest (2020)
2. DPMK: *Digitális munkarend a köznevelésben, módszertani ajánlások,* Budapest (2019) https://dpmk.hu/wp-content/uploads/2019/07/DigComp2.1_forditas_6_20200130.pdf (utoljára megtekintve: 2022. 11. 12)
3. Franz J. Mönks – Irene H. Ypenburg: *A nagyon tehetséges gyerekek*, Budapest (1998)
4. Tóth László – Sarka Ferenc: *A hazai tehetségsegítés története 1990-ig* In: A Tehetség kézikönyve. Budapest (2020) 23-43
5. Balogh László – Révész György: *Tehetségmodellek mint a fejlesztő programok kiindulási alapjai* In: A Tehetség kézikönyve. Budapest (2020) 44-94
6. A.H.Maslow: *A Theory of Human Motivation* (1943) https://www.researchhistory.org/2012/06/16/maslows-hierarchy-of-needs/ https://doi.org/10.1037/h0054346 (utoljára megtekintve: 2022.11.12)
7. Skultéty Viktor: *A humanisztikus pszichológia a vezetéstudományban* In: Tudományos közlemények 8. Általános vállalkozási Főiskola, Budapest (2003) 141-158
8. Steven D. Levitt – Stephen J. Dubner: *Lökonómia*, Budapest (2007)
9. A. Ionica-Ona: *Identification of students with talent in the technical do-mains.* In Studia Universitatis Babes-Bolyai – Psychologia-Pedagogia Vol. 58 Iss. 1, (2013) 83–92
10. European Unio: *Key competences for lifelong learning.* Luxembourg, (2019) https://op.europa.eu/en/publication-detail/-/publication/297a33c8-a1f3-11e9-9d01-01aa75ed71a1/language-en (utoljára megtekintve: 2022.08.28)
11. P. Szlávi, L. Zsakó: *Key concepts in informatics: Algorithm.* In: Acta Didactica Napocensia. Vol. 7 Iss. 1, (2014) 39–48 http://real.mtak.hu/39330/1/article_7_1_4.pdf (utoljára megtekintve: 2022.08.28)
12. H.Su: *Who are information and communication technology talents? A literature review.* In: Human Behavior and Emerging Technologies Volt. 2, Issue 3, July (2020) 288-297 https://doi.org/10.1002/hbe2.206

13. E. Nardelli: *Do We Really Need Computational Thinking?* In: Communications ACM. Vol. 62 Iss. 2, (2019) 32–35.
https://cacm.acm.org/magazines/2019/2/234348-do-we-really-need-computational-thinking/fulltext
https://doi.org/10.1145/3231587
(utoljára megtekintve: 2022.08.28)
14. Radnóti Katalin: *Milyen oktatási és értékelési módszereket alkalmaznak a pedagógusok?* In: Kerber Zoltán (Szerk.) Hidak a tantárgyak között. ISBN: 9636825726, Budapest (2005) 131-167
15. Farkas Beáta: *A közgazdasági gondolkodás rövid története*. ISBN: 978 963 454 742 6, Budapest (2021)
https://doi.org/10.1556/9789634547426
16. Csépe Andrea: *Marketing-kommunikáció a gyógyszeriparban*. In: Marketing & Menedzsment 2004.3. Pécs (2004) 36–44
17. Dávid Mária – Dávid Imre: *A tehetségígéretek keresése, azonosítása* In: A Tehetség kézikönyve. Budapest (2020) 95-153
18. Barsy Anna: *Tippek és trükkök az értékelésben* In: Károly Krisztina, Homonnay Zoltán (szerk.): *Mérési és értékelési módszerek az oktatásban és a pedagógusképzésben*; ISSN: 2416-2957 Budapest (2017) 113-124
19. Kahoot
https://kahoot.com/
(utoljára megtekintve: 2022.11.18)
20. Bebras
https://www.bebras.org/
(utoljára megtekintve: 2022.11.18)
21. CS Unplugges
https://csunplugged.org/en/
(utoljára megtekintve: 2022.11.18)
22. OKTV
https://www.oktatas.hu/pub_bin/dload/kozoktatas/tanulmanyi_versenyek/oktv/oktv2021_2022_1ford/info2_flap1f_oktv_2122.pdf
(utoljára megtekintve: 2022.11.18)
23. Olajos Tímea: *A tehetségígéretek fejlődésének általános jellemzői, kiemelten az alulteljesítő és speciális bánásmódot igénylő más tehetségesek fejlődésének sajátosságai.* In: A Tehetség kézikönyve. Budapest (2020) 186-209
24. P. Sarmasági: *DISC assessment usage in school talent management* In: Ciencia e Tecnica Vitivinicola 37. ISSN: 2416-3953. Lisboa (2022)
25. P. Sarmasági: *SWOT Assessment Usage in School Talent Management*. In: CEJNTREP.3.2.1355 Budapest (2021)
https://doi.org/10.36427/CEJNTREP.3.2.1355
26. Fülöp Márta: *Informatikai tehetség.* In: A Tehetség kézikönyve. Budapest (2020) 95-153

Valósidejű grafika a videojátékokban

Szabó Dávid[1], Dr. habil. Illés Zoltán[2]

{[1]sasasoft,[2]illes}@inf.elte.hu
ELTE IK

Absztrakt. Mit csinál egy játékfejlesztő? A játék szónak köszönhetően komolytalan a téma hangzata, de valóban így van? Hogyan kell játékokat készíteni? Olyan könnyű és egyszerű ez a feladat, mint a játékokkal játszani? Ezekre a kérdésekre szeretnénk választ adni ebben a cikkben, mely a videojátékok, mint valósidejű grafikus alkalmazások fejlesztésébe biztosít betekintést. Bemutatjuk a szükséges lépéseket, melyeket egy interaktív játékalkalmazás elvégez a képernyőn megjelenéshez, illetve bepillantást nyújtunk a valósághű 3D grafika előállításához használt technológiákba és algoritmusokba. Cikkünkben szemléltetjük, hogy a játékfejlesztés miért nem játék, miért egy komplex, nehéz ágazata a szoftverfejlesztésnek.

Kulcsszavak: valósidő, grafika, videojáték, shader, játék motor, grafikus motor

1. Bevezetés

A legtöbb videojáték a valóságnak egy szimulált és egyszerűsített változatát formálja meg. Szükség van egy virtuális világ (pálya) leírására, melyben különböző egymással interaktáló objektumok találhatók és jellemzően a felhasználó is befolyásolhatja, irányíthatja ezen objektumokat. A felhasználó felé kép-szekvencia és hanghatások segítségével mutatja be a virtuális világot. Mivel a felsorolt funkciókra szinte az összes játékban szükség van, ezért a videojáték ipar a fejlődése során a szimulációt végrehajtó és prezentáló algoritmusok jelentős részét újra felhasználható modulokba szervezte, ezeket nevezzük játék motoroknak.

1.1 Játékmotor

Egy játékmotor egy szoftverfejlesztői eszköz, melyet felhasználva nem szükséges egy új játék fejlesztéséhez a teljes szimulációs kódot újra implementálni. A motorban elérhető funkciókat módosítva és kiegészítve, mindössze a program egyedi részének elkészítése szükséges. Ezáltal egy általános játékmotor, mint a Unity3D [1], Unreal Engine [2] vagy bármely más, akár nyílt hozzáférésű, akár belső fejlesztésű motor tartalmaz különböző eszközöket, melyek gyakorian (vagy szinte mindig) szükségesek egy játék fejlesztésében.

Ilyen eszközök például a játékelemek menedzselése (Hogy néz ki? Hol van a térben? Hogy hat rá a fizika? Stb.), az erőforrások menedzselése (több gigabájtnyi kép és geometria adat dinamikus ki-és betöltése a memóriába), mesterséges intelligencia (gépi ellenfelek), audio motor (hangeffektek és zenék), grafikus motor (képi megjelenítés) és magát a szimulációt végző Game-Loop (periodikus reagálás a felhasználó interakcióra majd a hatásának prezentálása).

Ebben a cikkben a játékmotorok működését a bennük alrendszerként működő grafikus motor irányából megközelítve ismertetjük, mivel általában a játékmotorok architektúrájának megtervezésében jelentős szerepet játszik a grafikus megjelenítés.

1.2 Grafikus motor

A videojátékok és az átlagos grafikus felületű alkalmazások, mint például a Microsoft Word, Spotify és egyéb asztali programok megjelenítésében több közös vonás is felismerhető. A felsoroltak mind valósidejű grafikus alkalmazások, melyek a felhasználó interakcióira azonnal reagálnak és prezentál-

ják azt a felhasználó felé. Mikor egy gombra mutatunk az egérkurzorral elvárjuk, hogy a gomb felvillanjon, mikor kattintunk elvárjuk, hogy az alkalmazás reagáljon. Ugyanígy egy játékban mikor az "előre mozgás" billentyűjét lenyomjuk elvárjuk, hogy lássuk karakterünket elindulni a képernyőn.

Az azonnali reakció egy informális definíció, de a gyakorlatban vannak erősen definiált korlátok, melyek meghatározzák, hogy milyen gyorsan tud programunk reagálni a felhasználó mozdulataira. Ez a korlát a megjelenítőnk (monitor, televízió, képernyő stb.) frissítési rátája, a másodpercenként megjeleníthető képkockák száma. Grafikus alkalmazásainknak ezen megjelenítőket kell kiszolgálnia és a futás során folyamatosan a megfelelő számú képkockát kell előállítania az elvárt időzítéseknek és határidőknek megfelelően. [3]

1.2.1 Valósidejű grafika

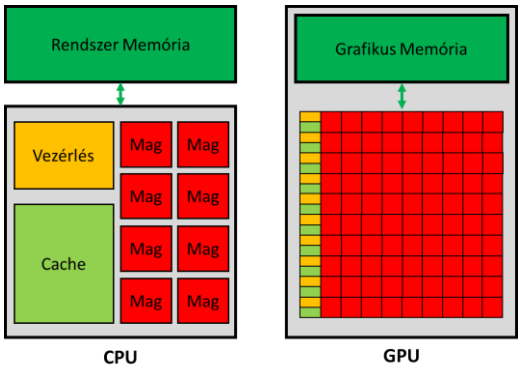
Egy mai átlagos monitor 60 hertz-es panellel rendelkezik, így 60 képkockát kell másodpercenként továbbítani az eszköz felé. Modernebb telefonok és televíziók 120 hertz-es, míg speciálisabb monitorok akár 144 vagy 240 hertz-es panelt használnak. Ez azt jelenti, hogy egy képkocka feldolgozására és előállítására nagyjából 16 milliszekundum számítási ideje van az alkalmazásunknak (1 másodperc / 60 hertz), melyet a program végrehajtása során folyamatosan és periodikusan be kell tartani, különben a grafika "szaggat", veszít a folyamatosságából és kizökkenti a felhasználót.

A manapság elterjedt televíziók 4k felbontásúak, tehát közel 8 millió képpontot (pixelt) jelenítenek meg egy képkockán (3840 pixel szélesség * 2160 pixel magasság). Egy grafikus alkalmazásnak az összes pixelre meg kell adnia egy színt, gyakorlatilag egy RGB byte struktúrát (a három színkomponensből kikeverhetők a megjelenítők által előállítható színek). Ebből kiszámítható, hogy körülbelül 25 Mb egy 4k képkocka eltárolásához szükséges memóriaigény (3 byte * ~8 millió pixel) és ilyen képkockából másodpercenkét 60-ra van szükség a megjelenítő kiszolgálásához, így egy grafikus alkalmazásnak nagyjából 1.5 Gb adatot kell előállítania másodpercenként (60 hertz * ~25 Mb).

A fejlesztők hamar felismerték, hogy egy CPU ekkora adatmennyiség párhuzamosított előállítására nem alkalmas, ezért megalkották a grafikai feladatok végrehajtására specializált GPU-t (Graphics Processing Unit).

2. CPU és GPU

Egy általános CPU 4-8 nagy teljesítményű processzormaggal rendelkezik, mely hosszú szekvenciába rendezett programok változatos utasításainak végrehajtására megfelelő, de egy 4k felbontású képkocka előállítása során 4-esével / 8-asával kellene feldolgoznia a ~8 millió pixelt. Egy GPU ezzel szemben alacsonyabb teljesítményű processzormagokat használ, de több tízezer ilyen mag érhető el egyetlen egységen. [4]



1. ábra: CPU és GPU architektúrájának összehasonlítása

2.1 GPU

Egy GPU nagyszerűen alkalmazható erősen párhuzamosított feladatokra és a grafika (a ~8 millió pixel feldolgozása), pontosan ilyen feladat. Ezen specializált egyszerű processzormagok nem előnyösek hosszú, változatos szekvenciális programok végrehajtására, de ez a grafikában ritkán szükséges. Mivel ekkora az architektúrális különbség a két processzortípus között, ezért eltérő a programozásuk módja is. Egy általános CPU-ra C-ben, C#-ban, Java-ban vagy hasonló programozási nyelvekben írt alkalmazást nem lehet egyszerűen GPU-ra lefordítani és ott futtatni. A GPU programozáshoz Grafikus API programfejlesztői könyvtárakat kell használnunk, melyek specifikációjukon keresztül elérhetővé teszik a GPU vezérlését és programozását. Ilyen könyvtárak az OpenGL [5], DirectX, Vulkan [6] és Metal, melyek C nyelvhez biztosítanak interfészeket. Manapság biztonsággal kijelenthetjük, hogy ha egy alkalmazás képes valamilyen grafikus felületet megjeleníteni a képernyőn (legyen szó, akár gombokból, feliratokból és egyéb komponensekből összerakott felületekről vagy vászonra rajzolt alakzatokról, formákról), akkor a háttérben a felsorolt Grafikus API-k egyikét használja a látvány prezentálásához.

2.2 Grafikus API programozás

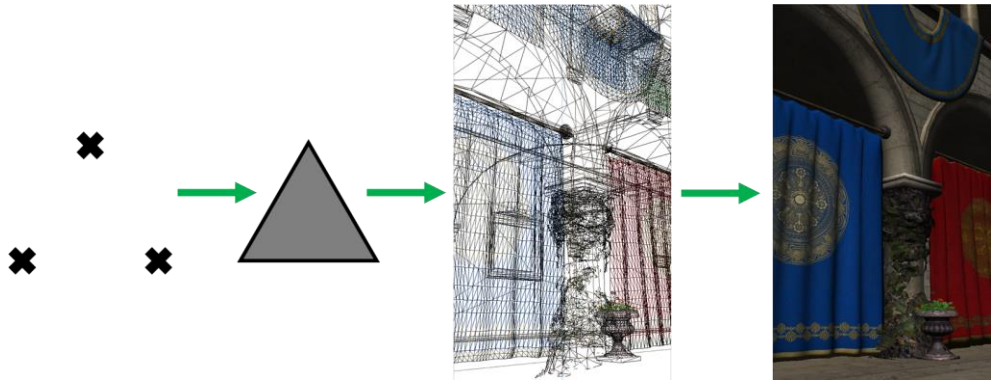
Grafikus programozás során megkülönböztetjük a feladatokat melyeket a CPU és a GPU végez. A két eszköz párhuzamosan dolgozik és végrehajtásuk adott pontjain szinkronizálják munkáik eredményét. A Grafikus API struktúráit és függvényeit használva a CPU-n kell a GPU-nak szánt feladatokat és parancsokat előkészíteni, melyhez jellemzően C vagy C++ nyelvet használnak. A parancsokat a GPU-nak átadva a GPU megkezdi a parancsok végrehajtását a CPU pedig folytathatja más feladatokkal (vagy akár a következő GPU parancsok előállításával is).

2.2.1 GPU parancsok

Ezek a parancsok jellemzően a GPU erőforrásaink konfigurálása és a rajzolási állapot beállítása. [7] A GPU állapot alapú rajzolással dolgozik. Hasonlóan, mint például a Logo teknőcgrafikája esetén, először be kell állítani rajzolási információkat (Logo-ban szín, tollvastagság, irány stb.), majd erre az állapotra ki kell adni a rajzolási parancsot (Logo-ban előre/hátra), mely elindítja a számítást és elkezdi a megfelelő pixelek színezését. Az állapothoz jellemzően különböző erőforrásokra van szükség a GPU-n, ezek közül a legtöbbet előre létre kell hozni és inicializálni kell, majd rajzolás előtt már csak az aktiválásuk szükséges. Szükség van buffer-ekre az adatok tárolásához a GPU memóriájában, textúrákra a kép-szerű adatokhoz, shader-ekre, melyek a GPU-n futó programok és algoritmusok és pipeline-ra (szerelőszalag), mely az ezek közötti kapcsolatot írja le az a következő rajzoláshoz. Összegezve: ahhoz, hogy a GPU-val „rajzoljunk" először be kell állítani a rajzolni kívánt geometria adatait, a rajzoláshoz használatos szerelőszalagot, majd ki kell adni a rajzolási parancsot.

2.2.2 Geometriák, modellek

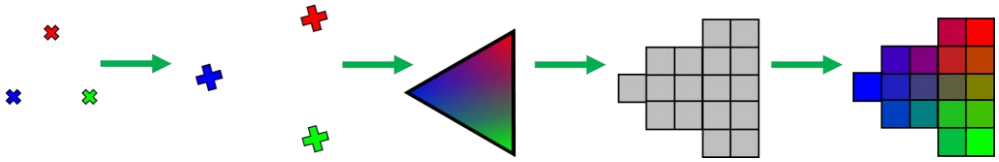
A Grafikus API-kkal jellemzően primitív geometriákat rajzolunk, pontokat, vonalakat és háromszögeket. Ha bármi komplexszebb alakzatra van szükség, akkor ezen primitívek összetételéből kell a geometriát felépíteni. A primitíveket a térbeli háromdimenziós csúcspontjaik felsorolásával adjuk meg (kétdimenziós esetben elég csupán az X és Y koordinátát használni a pozícióhoz, a Z-vel pedig a sorrendet adhatjuk meg).



2. ábra: Virtuális világ grafikájának összeállítása geometriák csúcspontjaiból. Balról jobbra: csúcspontok, háromszög, komplex geometriák háromszögekből felépítve, végső kép [8] fénymodellel megjelenítve

3. Grafikus Szerelőszalag

Egy rajzolási parancs kiadásakor a GPU a grafikus szerelőszalagot futtatja, mely a GPU-ba beprogramozott és beégetett algoritmusok sorozata. A szerelőszalag feldolgozza a beállított állapotot és ez alapján pixeleket színez ki a képernyőn. Feladatai, hogy a megadott geometria csúcspontjait transzformálja a térben, a csúcspontok összeszerkesztése komplexszebb alakzatokká, majd az alakzatok képernyőre vágása. Ezután meg kell állapítani, hogy melyik alakzat mely pixeleket fedi le, végül a lefedett pixeleket ki kell színezni.



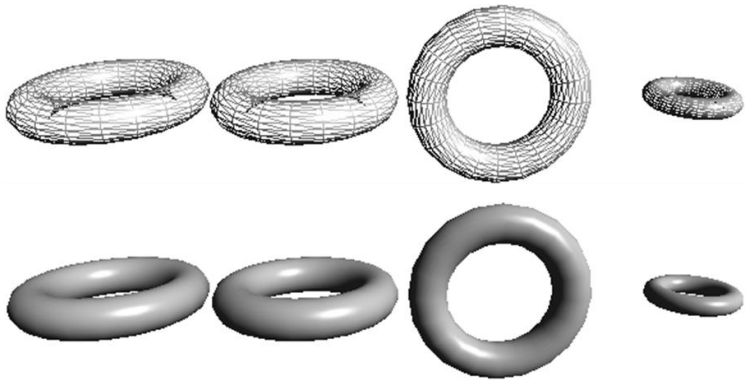
3. ábra: A Grafikus Szerelőszalag lépései. Balról jobbra: Csúcspontok (bemeneti adat), csúcspontok feldolgozása (transzformációk), geometria összeszerkesztése, lefedett pixelek meghatározása, pixelek színezése (kimeneti adat)

A szerelőszalag egyes lépései során gyakran előfordul, hogy nagy mennyiségű adatot kell feldolgozni. Amennyiben a teljes képernyőt lefedi egy geometria, akkor a pixelek színezését a teljes képernyőre le kell futtatni (tehát 4k megjelenítőn ~8 milliószor). Egy mai videojátékban átlagosan 1-3 millió csúcspont jelenik meg egyszerre a képernyőn, melyeket transzformálni kell majd összeszerkeszteni, így a szerelőszalag korai fázisai is jelentős adatmennyiséget dolgoznak fel. Míg egy mai modern CPU képtelen megbirkózni ekkora számossággal, a GPU-k előbb ismertetett architektúrája tökéletesen alkalmas, mert erre a feladatra lett tervezve.

3.1 Csúcspontok feldolgozása

A lineáris algebrában tanult vektor-mátrix szorzással transzformálhatjuk a geometriák csúcspontjait. A megfelelő transzformációs mátrixszal mozgathatjuk, forgathatjuk és méretezhetjük az alakzatokat. Inicializálásnál elég csak egyetlen egyszer identitás állapotban definiálnunk a geometriák csúcspontjait, később kirajzolásnál a transzformációkkal helyezhetjük el őket a virtuális világunkban, akár egyszerre több különböző helyen és orientációban is. [9] A kamera leírása, azaz a "virtuális térbeli szemünk" elhelyezése is transzformációkkal történik, ez határozza meg, hogy mit látunk a virtuális világunkból. A transzformációk alkalmazásához a geometriák minden csúcspontjának pozícióját

meg kell szorozni az aktuális mátrixszal, ezáltal a megfelelő helyre és orientációba transzformálva a teljes alakzatot.



4. ábra: Csúcspontok transzformációi. Balról jobbra: identitás (nincs transzformáció), eltolás, elforgatás, méretezés

3.2 Pixelek feldolgozása

A geometriák összeszerkesztése után a szerelőszalag kiszámítja, hogy ezek a képernyő mely pixeleit fedik le. Az alakzatok csúcspontjaihoz további információkat is megadhatunk, például szín, normálvektor (felfelé mutató irány az alakzat felszínén) stb., mely adatok súlyozottan átlagolva lesznek elérhetők a lefedett pixelek színezésekor. A színezés történhet egyszerűen, például az átlagolt adatokból megkapott szín módosítás nélküli felhasználásával. Jellemzően a pixelek színezése során szimuláljuk a fénymodellt, valamilyen algoritmus alapján.

3.2.1 Fénymodellek

Az emberi szem a fényeket érzékeli, a fényeknek köszönhetően látunk. Ebből kifolyólag, ha képesek vagyunk a fényeket fizikailag korrekten megjeleníteni, akkor az összes valóságban megtalálható vizuális hatást (pl. árnyékok, tükröződések, fénytörés, fényszóródás stb.) is sikeresen megjelenítettük. A fények szimulálását a pixelek szintjén végezzük, mivel ez a legkisebb egység melyet a megjelenítő prezentálhat, tehát ez a felhasználó eszközén elérhető maximális részletességi szint.

A fizikailag korrekt fények reprezentálásának egyenlete (rendering equation) [10], már 1986-ban be lett mutatva, melyet megvalósítva teljesen valósághű képet kaphatunk. Ez az egyenlet leírja, hogy egy adott felület miként reagál a beérkező fénysugarakra. Mivel az egyenlet komplex műveleteket használ (például integrálást), így a mai napig nem lehetséges számítógépen az egyenlet pontos implementálása, csupán közelíteni tudjuk azt. Minél pontosabb a közelítés annál valóságosabb a fény/felület interakció, de az approximációt kiszámító kód teljesítményigénye is ennek függvényében növekszik.

$$I(x,x') = g(x,x')\left[e(x,x') + \int_S p(x,x',x'')I(x',x'')dx''\right] \quad (1)$$

ahol:

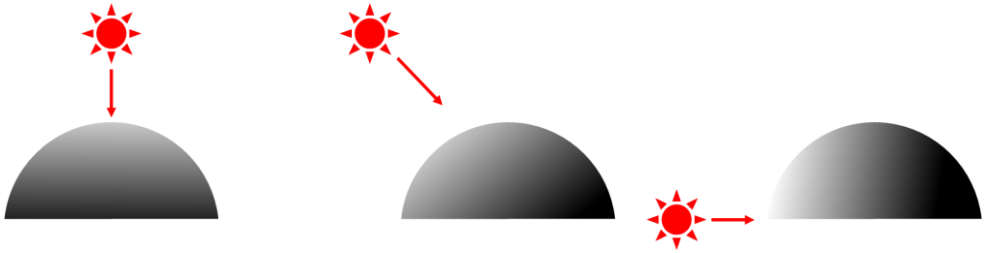
I(x, x'): x' beérkező irányból x kimenő irányba továbbított fény intenzitása

g(x, x'): geometria függvény

e(x, x'): x' irányból kibocsátott fény intenzitása x kimenő irányba

p(x, x', x''): x'' irányból a felületre beérkező és szétszóródó fény intenzitása x' kimenő irányba

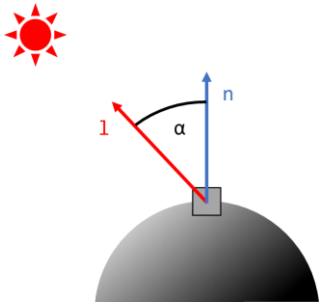
Az egyenlet egy adott felületi pontra (pixelre) írja le a virtuális világ fényforrásai és környező objektumai alapján a visszavert fényt intenzitását (színét). A felületi pontot az x vektor irányából nézzük (a „virtuális szemünkből" a felületi pontra mutató irányvektor) és egy fényforrás (pl. nap, fáklya, villanykörte stb.) fénysugarai x' vektor irányából érkeznek a felületi pontra. Az egyenlet kimenete a fényforrás fénysugarainak a felületi pontról a szemünkbe továbbított intenzitása (az adott szemszögből, milyen színűnek látjuk az objektumot azon a pixelen).



5. ábra: Fényforrás hatása egy objektum felületén.

3.2.2 Egyszerűsített fénymodell

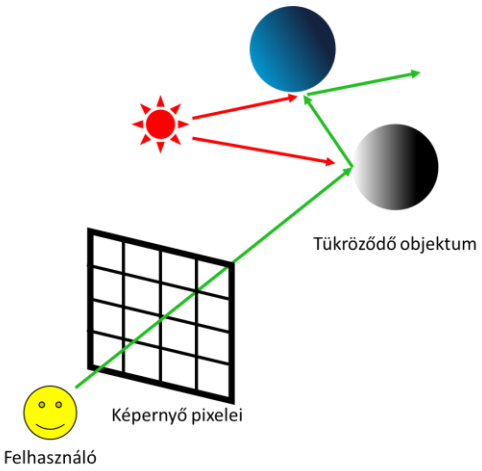
Általánosságban a fény szimulálásához egy pixelen szükségünk van a felületi normál vektorra, a fényforrás felé mutató irányvektorra, illetve a fény és a felület színére. A normálvektor és a fényforrás felé mutató vektor közötti szög határozza meg, hogy mennyire ideális a fény beesési szöge. Ha a két vektor párhuzamos, akkor 100%-ban megvilágítja a felületet a fény, ha merőleges, akkor egyáltalán nincs hatása a fénynek az adott pixelen. [11]



6. ábra: Egyszerűsített fénymodell kiszámításához szükséges adatok egy felületi ponton (pixelen). l: a fényforrás felé mutató irányvektor, n: a felületi pont normálvektora, α: l és n által bezárt szög (fény intenzitása a felületi ponton)

3.2.3 Fénymodell sugárkövetéssel

A videojátékok grafikus motorjainak számára az elmúlt években elérhetetlen volt a filmekben használt sokkal valósághűbb fény szimulációs metódus, a raytracing (sugárkövetés). [12] Ahogy a neve is sugallja, sugárkövetés esetén a fényforrások fénysugarainak útját követjük és szimuláljuk. Megvizsgáljuk, hogy mely felületi pontokra, mely fényforrások sugarai jutnak el (nem takarja egyéb objektum az utat a felületi pont és a fényforrás között), továbbá kiszámítjuk a felületen megtört és tükröződő sugarakat, melyeket tovább követve információkat kaphatunk a felületi pont tükröződéséről és belső fényvisszaverési tulajdonságairól.



7. ábra: Fénymodell szimulálása sugárkövetéssel

A modern GPU-k már képesek valós időben a teljes képernyő felbontás töredékén végezni a sugárkövetést és azt mesterséges intelligencia algoritmusokkal felnagyítva alkalmazni a teljes képre. Habár már az is nagyszerű eredmény, hogy erre 16 milliszekundum alatt képes a mai technológia, a filmekben látott teljes felbontású valósidejű sugárkövetésre még nem áll készen.

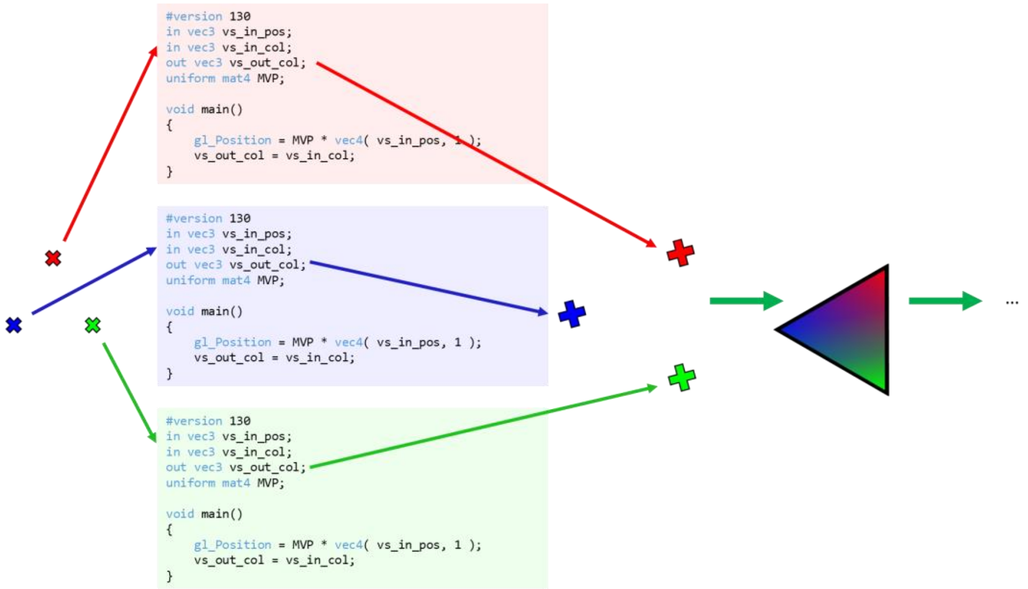
4. Shader

A szerelőszalag működésének jelentős része "be van égetve" a GPU architektúrájába. Speciális tranzisztorok és áramkörök találhatók minden GPU processzormagban, melyek kizárólag a szerelőszalag egyes feladatait képesek elvégezni, mindezt nagy sebességgel az általános számítást végző áramkörökhöz képes. A szerelőszalag egyes részei viszont programozhatók, saját algoritmust írhatunk, melyeket a GPU fog végrehajtani a szerelőszalag adott fázisaiban. Ezeket a kicsi programokat nevezzük shader-eknek (árnyaló, először csak a fény árnyékolás kiszámítására használták), melyeket a használt Grafikus API által meghatározott specializált C szerű nyelven kell elkészíteni. [13]

OpenGL esetén ez a nyelv a GLSL (OpenGL Shading Language) [14], melynek a Vulkan egy bináris változatát használja. DirectX shader-eket HLSL (High-Level Shading Lanugage), Metal shadereket pedig MSL (Metal Shading Language) nyelven kell készíteni. Az előbb bemutatott szerelőszalag két programozható fázisa a csúcspont feldolgozás és a pixel feldolgozás, melyekhez készíthetünk shader programokat.

4.1 Shader programozás koncepciója

Egy kirajzolás során ugyanazt a csúcspont shader programot használjuk a geometria összes csúcspontjára és ugyanazt a pixel shader programot az alakzatból keletkező összes pixelre. Így a shader programba írt main függvény többször is futtatva lesz egy kirajzolás során, csupán a program bemenete lesz a lefutások között más. A csúcspont transzformációs shader esetén minden csúcspontra ugyanaz a kód fut, melyben a pozíció vektort megszorozzuk a megadott mátrixszal, de a szerelőszalag mindig másik csúcspont pozícióját fogja a shader futtatásakor megadni, ezáltal az összes pontra futtatva a programot. A kirajzoláshoz használandó shader-eket, a GPU állapotában kell megadni a szerelőszalag beállításával.



8. ábra: Transzformációs csúcspont shader GLSL nyelven. Az ábrán a geometriánk három csúcspontból áll, ezért ugyanaz a csúcspont shader háromszor fog futni, mindhárom futtatás más csúcspont adatait kapja meg bemenetként és ezáltal más kimenetet is fognak generálni.

A shader-ek programozásának gondolatmenete eltér egy hagyományos C, C#, Java vagy hasonló nyelvekben írt szekvenciális programok fejlesztésétől. Meg kell barátkozni a koncepcióval, hogy ugyanaz a shader kód egyszerre többször fog futni, mindössze a bemeneti értékek lesznek mások (csúcspont feldolgozás esetén a geometria csúcspontok adatai, pixel feldolgozás esetén a lefedett pixelek meghatározása során előállított átlagolt csúcspont adatok). Ezen felül lehetnek további változóink, melynek értéke ugyanaz a geometria összes csúcspontjának és pixelének feldolgozása során. Ilyen értékek lehetnek a transzformációs mátrixok vagy a fényforrások információi. GLSL nyelvben az in, out és uniform kulcsszavakkal adhatjuk meg, hogy melyik változóink bemeneti, kimeneti, illetve az összes adathoz ugyanolyan értékkel bíró változók.

5. Modern videojáték motorok

Egy mai videojáték motor többnyire elrejti az előbbi alacsony szintű funkciókat a felhasználói elől, mivel ezek minden játék fejlesztéséhez szükséges metódusok. A shader-ek programozása a mai napig egy fontos lépése a grafikus effektek megjelenítésének, ezek továbbra is mind valamilyen szerelőszalagban futó egyedi shader-ekkel készülnek. A shader-ek ismétlődő részei, mint a transzformációk vagy a fények szimulációja beépített makrókként és függvényekként érhetők el, így ezen részek újra-implementálása nem szükséges, mindössze az egyedi effekt hozzáfűzése a feladat. Természetesen a hozzáadott kódnak kompatibilisnek kell lennie a motor elvárásaival és beépített algoritmusaival, különben ezeket is nekünk kell teljesen újra implementálnunk (feltéve, hogy a motor támogatja).

5.1 Grafika absztrakció a játékmotorokban

A motorok jellemzően megkülönböztetik a Mesh-t (modellek, geometriák) és Material-t (anyagtulajdonságok). [15] A Mesh-ek írják le, hogy milyen alakzatokat jelenítünk meg, mik a csúcspontok és milyen csúcspont adatok érhetők el. Itt lehet szó statikus objektumokról, mint épületek, domborza-

tok, fák stb. vagy akár dinamikusan formázható modellekről is, mint például a karakterek (hajló végtagok, forgó fej). Ezen Mesh-ek megjelenítéséhez használt shader-eket a Material-ok határozzák meg. A Material írja le, hogy milyen shader-eket használjon a szerelőszalag a kirajzoláshoz és azon shader-ek milyen paraméterekkel dolgozzanak (textúrák, színek, változó értékek stb.). Végül szükség van valamilyen Transzformációs komponensre, amely megadja, hogy egy játékbeli objektum hol van a virtuális térben. A motor ezen komponensből számítja ki a transzformációs mátrixokat, melyeket a Material-ban kiválasztott shader-nek ad át kirajzolás előtt.

5.1.1 Hagyományos grafikus motorok

Összegezve egy grafikus motor egy képkockájának megjelenítése elképzelhető a következő pszeudo algoritmussal. Az algoritmus több egymásba ágyazott foreach ciklusból épül fel, melyek a Material-okat, Mesh-eket és Objektum Transzformációkat sorolják fel. [16] Először végig iterálunk az összes jelenleg használt Material-on. Minden iterációban először a GPU állapotába beállítjuk az aktuális Material adatait, majd végig iterálunk az összes Mesh-en, amely használja az aktuális Material-t. Minden iterációban az aktuális Mesh-t beállítjuk a GPU állapotába, majd felsoroljuk az összes objektum Transzformációs komponenseit, amelyek a jelenlegi Mesh kirajzolásához a jelenlegi Material-t használják. Ha az objektum látható az adott pozícióban (nem mögöttünk van, nincs takarva), akkor beállítjuk a GPU állapotába a transzformációs mátrixot és kiadjuk az állapotra a rajzolási parancsot (ezen a ponton az állapotban a megfelelő Material, Mesh és Transzformáció aktív).

Ez a megközelítés univerzálisan képes több különböző geometriából, akár több példányt megjeleníteni a szükséges fény szimulációval és egyéb grafikus effektekkel. Több motor (pl. Unity3D, Godot [17]) még manapság is ezt a megközelítést alkalmazza, mivel megfelelően általános, valamint gyengébb teljesítményű vagy régebbi funkcionalitású GPU-kon is megfelelően fut. A Mesh-ek és Material-ok számának növelésével fokozatosan teljesítmény igényesebb lesz az algoritmus, míg el nem jutunk a küszöbig, ahol már a motor nem lesz képes a szükséges 60 képkockát előállítani másodpercenként.

5.1.2 GPU vezérelt grafikus motorok

Egyes játékmotorok (pl. Unreal Engine, id Tech) az újabb GPU driven rendering (GPU vezérelt megjelenítés) megközelítését alkalmazzák, melynek hátránya, hogy gyengébb és régebbi GPU-k nem támogatják, de a modernebb eszközökön sokkal jobb teljesítményt biztosít, mint a hagyományos algoritmus. GPU driven rendering során az összes Mesh és Material információt egy-egy nagy GPU buffer-be vonjuk össze és a GPU-n egyetlen speciális rajzolási utasítással, megfelelően indexelve és összepárosítva rajzoljuk ki a játékelemeket. [18] Ezzel a metódussal nincs szükség a CPU kódban a többszörösen egymásba ágyazott nagy iterációszámú ciklusokra, mert ezt is a GPU végzi a saját párhuzamosítási eszközeivel.

6. Konklúzió

Cikkünkben a videojátékok grafikus alrendszeréhez használt szoftverfejlesztési eszközök használatáról biztosítottunk egy átfogó képet. A videójátékok, mint interaktív valósidejű grafikus alkalmazások a GPU (grafikus processzor) felhasználásával érik el a megjelenítéshez szükséges teljesítményt. A GPU-k programozása nagyban eltér az általános CPU programozástól. Áttekintettük a grafikus szerelőszalag működését és shader programok fejlesztését, melyekkel transzformálhatjuk a játék modelljeit és megfelelő fényszimulációval jeleníthetjük meg azokat.

Az ismertetett metódusokat egy mai játékmotor tartalmazza magában és felhasználhatóvá teszi a játékfejlesztők számára, de ezek kiegészítéséhez szükséges a belső rendszer alapjainak áttekintése. A technológia folyamatos fejlődéséhez pedig elengedhetetlen a grafikus motorok programozásának

alapos ismerete, mivel a fejlesztés ezen rendszerek bővítésével vagy éppen teljesen új alapokra helyezésével jár.

Irodalom

1. Unity Technologies: *Unity*, https://unity.com/ (utoljára megtekintve: 2022.11.18.)
2. Epic Games, *Unreal Engine*, https://www.unrealengine.com/ (utoljára megtekintve: 2022.11.18.)
3. D. Szabó, Z. Illés, V. B. Heizlerné: *Real-Time functionality in Windows*, INFODIDACT (2018) 263-264
4. NVIDIA Corporation: *NVIDIA Fermi Architecture Whitepaper*, https://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf (utoljára megtekintve: 2022.11.18.)
5. D. Szabó, Z. Illés: *Real-time OpenGL graphics in modern C#*, DIDMATTECH (2019)
6. D. Szabó, Z. Illés: *Vulkan in C# for Multi-Platform Real-Time Graphics,* 18th International Conference on Emerging eLearning Technologies and Applications (2020) 687-692
7. D. Szabó, Z. Illés: *Real-time rendering with OpenGL and Vulkan in C#*, Recent Innovations in Computing: ICRIC (2021)
8. Jimmie Bergmann: *Sponza GitHub*, https://github.com/jimmiebergmann/Sponza (utoljára megtekintve: 2022.11.18.)
9. T. Akenine-Möller, E. Haines, N. Hoffman: Real-Time Rendering, Third Edition, Taylor & Francis Group (2008)
10. J. T. Kajiya: *The Rendering Equation*, SIGGRAPH (1986)
11. B. T. Phong: *Illumination for Computer Generated Pictures*, Communications of the ACM (1975)
12. T. Akenine-Möller, E. Haines: *Ray Tracing Gems*, Apress (2019)
13. Y. He, T. Foley, T. Hofstee, H. Long, K. Fatahalian: *Shader Components: Modular and High Performance Shader Development*, ACM Transactions on Graphics (2017)
14. John Kessenich, Dave Baldwin: *The OpenGL shading Language (Version 4.5)*, https://registry.khronos.org/OpenGL/specs/gl/GLSLangSpec.4.50.pdf (utoljára megtekintve: 2022.11.18.)
15. B. Karis: *Real Shading in Unreal Engine 4*, SIGGRAPH 2013 Course: Physically Based Shading in Theory and Practice (2013)
16. Sebastian Aaltonen, Tim Cooper, Natalya Tatarchuk: *YouTube: SIGGRAPH 2021 REAC: Unity Rendering Architecture*, https://www.youtube.com/watch?v=6LzcXPIWUbc (utoljára megtekintve: 2022.11.18.)
17. Juan Linietsky, Ariel Manzur: *Godot Engine*, https://godotengine.org/ (utoljára megtekintve: 2022.11.18.)
18. Brian Karis, Rune Stubbe, Graham Wihlidal: *YouTube: A Deep Dive into Nanite Virtualized Geometry*, https://www.youtube.com/watch?v=eviSykqSUUw (utoljára megtekintve: 2022.11.18.)

Teaching Python Considered Harmful? A strukturált programozás Python nyelven tanítása káros

Szalayné Tahy Zsuzsanna

sztzs@inf.elte.hu
ELTE IK

Absztrakt. A Python programozási nyelv alapjai a programozásban teljesen járatlanok számára könnyen tanulható, a bőségesen rendelkezésre álló nyelvi modulok sok területen jól használhatók. Ezek és hasonló érvek teszik a programozás alapjainak oktatásához divatossá a Python nyelvet. A 2020-as NAT kötelező tananyaga az algoritmizálás és programozás, így ez a téma is külön fejezetként szerepel az egyetlen állami tankönyvben, amely – követve a divatot – a Python nyelvet használja. A dokumentumok elemzése során megvizsgáljuk, hogy a pythonic kódolási stílus egyedisége mennyire segíti vagy gátolja az algoritmizáláshoz, strukturált programozáshoz szükséges fogalmak elsajátítását, a gondolkodási és kódolási kompetenciák fejlesztését.

Kulcsszavak: programozás oktatása, programozási nyelv választása, Python, strukturált programozás

1. Bevezetés

1968-ban „*Go-to statement considered harmful"* címen jelent meg Dijkstra levele [1], amelyben a strukturált programozást javasol a gépikódnál magasabb szintű programok írásához. A levél eredeti címe „*A Case against the GO TO statement"* arra utal és a levél tartalma azt fejti ki, hogy spagettikód egy összetett, nagyobb program esetén káros. A publikált, frappáns cím a gépikódnál magasabb szintű nyelvekre érvényes strukturált programozás [2] leggyakrabban idézett mondatává vált [3].

Az akkori, gépikódra optimalizált `goto` használatához hasonlóan, napjainkban az emberi kommunikációra optimalizált Python nyelv tanulása divatos. A divatot meghatározó előnye, hogy elsődlegesen objektumorientált; „mindenre" van kész modul; ingyenes. Ezek az előnyök érvényesülnek a mesterséges intelligencia programozása területén, az IoT fejlesztések során, a szoftver prototípusok készítésekor, más tudományágak informatikai támogatása során. Python nyelven a gyakran használt nyelvi eszközök használata könnyű (szinte angol nyelven írt egyszerű mondat); ezért a nyelv tanulásának első lépései sikerélményt adnak, motiválnak.

2. A Python nyelv szerepe az iparban és a felsőoktatásban

Az előnyöket érvényesítendő, a BME VIK ipari szereplőkkel közösen szervezett – duális – üzemmérnök-informatikus képzésén, a programozási alapismeretek első szemeszteres tantárgyban[1] (2018–2022 években) a Python nyelvet használják. Az oktatási szakértőkkel beszélgetve azt halljuk,

[1] BME-VIK Bprof képzés „A programozás alapja" tantárgy adatlapja: https://portal.vik.bme.hu/kepzes/targyak/VIEEBA01/ [Megtekintés: 2022.10.31]

hogy „az ipar a Python ismeretét várja el, ezért mindenhol Pythont tanítanak”, de az állítás ellenőrzése során nem egészen ezt tapasztaljuk: A 2019-es Craft Conference[2] kiállítóit kérdezve, a Python „kezdetnek jó az is” nyelv, a cégek termékei jellemzően C++, C#, Scala nyelven készültek.

Másik végletként figyelemre méltó az ELTE IK-n zajló képzés. Az elmúlt években volt, hogy a felsőfokú szakképzésen Python nyelven tanították a Programozási alapismereteket, néhány évig az *Imperatív programozás* tantárgy keretében a C nyelv pointere tárgyalása helyett Pythont tanultak a hallgatók, de a 2022-es tanévben a bevezető, *Programozás* tantárgyban a strukturált programozás alapjait C# nyelven[3], az *Imperatív programozás*t C nyelven tanítják[4]. Ezzel szemben a korábban IK-n is tanuló matematikusok ma már a TTK keretei között csak Python nyelvet használnak[5].

Az ELTE TTK-n tanított „*Bevezetés a tudományos programozásba gyakorlat*” tantárgy tematikájában feltűnő, hogy az objektumorientált paradigma mellett még utalás szintjén sem fordul elő sem a strukturált programozás, sem a programozás típusalgoritmusai (programozási tételek). A tantárgy fókuszában a Python matematikai moduljai állnak: numpy, matplotlib, pandas.

Kibővítve a vizsgálódást, az egyetemi képzésben a Python szerepéről online tájékozódhatunk. Magyarországi egyetemek körében:

- BME VIK BSc mérnökinformatikus képzésén C[6] nyelvvel kezdenek.
- Az Óbudai Egyetem NIK-en „Szoftverfejlesztés és -tervezés I” tantárgy[7] C# nyelvet használ.
- A Pázmány Péter Katolikus Egyetemen saját PLang[8] nyelven tanítják a strukturált programozást.
- A Szegedi Tudományegyetemen C nyelven implementálják a programozás alapjait.[9]
- A Corvinus Gazdaságinformatikus[10] képzésén a C# nyelvet preferálják.
- Semmelweis Egyetemen orvostanhallgatóknak az általános informatika képzés részeként Pythont tanítanak.[11]

---

[2] Craft Conference 2019 (szervező: Prezi és IBM Budapest Lab): https://craft-conf.com/2019 [Megtekintés 2022.10.31.]

[3] ELTE IK: *Programozás*. Programtervező informatikus nappali és esti tagozat részére: https://progalap.elte.hu/
Programtervező informatikus felsőoktatási szakképzés: https://progalapfsz.elte.hu/ [Megtekintés 2022.10.31.]

[4] ELTE IK: *Imperatív programozás*. http://gsd.web.elte.hu/imper/ [Megtekintés 2022.10.31.]

[5] ELTE TTK Matematika Intézet: *Bevezetés a tudományos programozásba gyakorlat.* https://www.math.elte.hu/kepzesek/bsc-alapkepzes/bsc-matematika-tantervi-halo-tervezo/#verboseHook [Megtekintés 2022.10.31.]

[6] BME VIK EET: *Programozás alapjai 1*. https://infoc.eet.bme.hu/utemterv/ [Megtekintés 2022.10.31.]

[7] ÓE NIK: *Szoftverfejelsztés és -tervezés I.* OE-NIK-AII, 2017 https://docplayer.hu/126600453-Szoftvertervezes-es-fejlesztes-i.html [Megtekintés 2022.10.31.]

[8] Pázmány Péter Katolikus Egyetem ITK: *Bevezetés a programozásba 1*. https://docplayer.hu/6664391-Ppke-itk-bevezetes-a-programozasba-1.html [Megtekintés 2022.10.31.]

[9] SzTE TTIK Informatikai Intézet: *Programozás alapjai.* http://www.inf.u-szeged.hu/kurzusleirasok/IB103.xml [Megtekintés 2022.10.31.]

[10] Budapesti Corvinus Egyetem Információrendszerek Tanszék: *Szoftvertechnológia I.* http://sirius.uni-corvinus.hu/root/web/web.nsf/do?open&lang=hu&page=oktatas [Megtekintés 2022.10.31.]

Nézzük néhány ismert külföldi egyetem képzési tájékoztatóját is:

- A Columbia[12] egyetemen hét hetes, heti kétórás tárgy keretében lehet Pythont tanulni.
- Az ETH Zürich tájékoztatójában[13] jellemzően a biológiával kapcsolatos képzéseken (42,5%) Pythont tanítanak, a képzések további 37,5%-án C++, 20%-ban Java a kezdő nyelv.
- A Harvardon David Malan nem programozóknak szóló, online is tanulható kurzusán[14] az első héten bemutatott nyelv a Scratch, majd C követi. Egy előadásban van szó a Python nyelvről.
- A MIT[15]-en azoknak tanítanak Pythont, akiknek nincs programozási előismerete.
- Cambridge egyetemén a felvételizőknek ajánlják[16], hogy valamilyen nyelvet tanuljanak, ami lehet a divatos Python.

Konklúzió:

Az ELTE TTK matematikus szakján nem tanítják a strukturált programozást, de vélhetően használják. Ahol a Python nyelvet használják a programozás tanítására, ott a programozás kvázi alkalmazói ismeret; az algoritmizálás absztrakciója, tipizálása nem tananyag. Az informatikus (Computer Science) képzésben kivétel nélkül a szakma alapjainak elsajátítása a strukturált programozás oktatását helyezi középpontba, a Python nyelv nem szakmai, hanem a holisztikus tanulási/tanítási módszer eszközeként kerül a tananyagba.

3. Programozás, mint az általános műveltség része

Főként a külföldi példákon érzékelhető, hogy a programozás egyetemi oktatása felzárkóztató jellegű, aminek során egy, ma már mindenki számára szükséges kompetencia megszerzését biztosítják. Ezeket a kompetenciákat találjuk meg a DigComp 2.2: The Digital Competence Framework for Citizens [4] elvárásai 3.4-es pontjaként. A témához képest tömör megfogalmazásokhoz készített tanulmány – Programming for All: Understanding the Nature of Programs [5] – a programozással kapcsolatos kompetenciákat nyelvfüggetlenül írja le.

A korábban ECDL, újabban Európán túl is elismert bizonyítványt adó ICDL vizsgák fejlesztése a DigComp kompetenciákhoz igazodik. Egyik új (2021-es) modul a *Kódolási ismeretek (Python)*. A modul tananyaga, a syllabus 1.0 [6], csak magyar nyelven található meg, bár ausztrál, román és szlovák tematikában is szerepel. A tananyag néhány részlete:

3.2.6 Összetett adattípus használata egy programban, például: tömb, lista, tuple (tároló).

---

[11] Szemmelweis Egyetem Egészségügyi Közszolgálati Kar: Bevezetés az információs technológiákba (2) – Programozás https://semmelweis.hu/dei/files/2020/09/Programozas_20220607.pdf [Megtekintés 2022.10.31.]

[12] Columbia University Computer Science Department: *COMS W3101-1 - Programming Languages. Python.* http://www.cs.columbia.edu/~bauer/cs3101-1/ [Megtekintés 2022.10.31.]

[13] ETH Zürich: *Programming for beginners.* https://ethz.ch/en/studies/bachelor/beginning-your-studies/subject-related-preparation/programming-beginners.html [Megtekintés 2022.10.31.]

[14] Harvard University: *CS50: Introduction to Computer Science.* https://online-learning.harvard.edu/course/cs50-introduction-computer-science [Megtekintés 2022.10.31.]

[15] Massachusetts Institute of Technology *Programming & Software Engineering.* http://catalog.mit.edu/subjects/6/ [Megtekintés 2022.10.31.]

[16] University of Cambridge, Department of Computer Science and Technology: *Admission FAQ.* https://www.cst.cam.ac.uk/admissions/undergraduate/faqs [Megtekintés 2022.10.31.]

4.2.2 Ciklus típusok: elöl tesztelő, hátul tesztelő és számláló (FOR, WHILE, REPEAT).

4.2.3 Ciklus használata egy programban.

4.4.4 Függvény írása és elnevezése egy programban.

4.5.1 Az esemény ismerete. Az esemény célja egy programban.

4.5.2 Eseménykezelők használata, pl.: egérkattintás, billentyű lenyomása, gombkattintás, időzítő.

A tananyag jelentős része a strukturált programozásról szól, amihez Pythont használ. A nyelv választásának oka, a népszerűsége. A holisztikus tanuláshoz – könnyen tanulható – a Python az elsőnek ajánlott nyelv. A cél és eszköz egymásmellé helyezését megfigyelhetjük a syllabusban:

- A folyamatábra és pszeudokód készítéséhez a tömb és rekord (adatstruktúra) fogalma szükséges, de Pythonban ezek nem léteznek. A 3.2.6 pont alapján a tömböt helyettesíti a lista, a rekord helyett lenne a tuple, ami alkalmas vegyes típusú adatok tárolására, de ez az adattípus Pythonban nem módosítható, ráadásul bár több nyelvben létezik ez az adattároló, nagyon eltérő a szerepük, tulajdonságaik. Kérdéses, hogy a „tároló" kiegészítés miért szükséges és valójában mi lesz a kapcsolat a pszeudokód és a Python között.
- Ugyancsak kérdéses, hogy ha az összetett adattípusok a lista és a tuple (3.2.6), akkor milyen adat lesz az, amire értelmezhető az egérkattintás (4.5.2). A tananyagból az OOP ismeretének részletezése hiányzik, pusztán az eseménykezelés utal arra, hogy a rajzon vázolt algoritmus és konzolos alkalmazások készítésén túl grafikus felületű alkalmazást is tudni kell készíteni.
- A 4.2.2 pontban megjelölt ciklus típusok a strukturált programozás vezérlési elemei, az eredeti szövegben csupa nagybetűvel kiemelt vezérlési kulcsszavak a Pascalra emlékeztetnek. A három típusból csak kettő adott a Python nyelvben. Vajon milyen működő programmal lehet kipróbálni a hátul tesztelő ciklust? Vajon a számláló ciklus absztrakciójában a ciklusváltozó növekedni fog vagy végig lépked egy tartomány elemein, az egymást követő számokon?

Az egymásmellé rendelt tartalmak alapján úgy tűnik, hogy a kódolási ismeretek megszerzésének két része van. Az egyikhez felfedező, kísérletező tanulás társul, aminek során Python nyelven kódolt alkalmazások készülnek. A másik egy elméleti tananyag, ahol rajzolni vagy írni kell számítógépen, de ennek kevés köze lesz a működő programhoz. Így az is kérdéses, hogy miért szükséges és az is, hogy mi ezekben (az algoritmus leíró módszerekben) a motiváló.

További kérdés, hogy mi az a tudáscsomag, aminek ismeretét a Kódolási ismeretek (Python) vizsgamodul méri, mi teremti meg az elemi adattípusok és az eseménykezelés között a kapcsolatot. A vizsgamodul esetleg mérheti egy robotnak, annak korlátozott számú részegységének a programozásában való jártasságot. Ebben az esetben a programozás alapfogalmait: a kódolást, az adattípusokat, a vezérlési struktúrákat egy robotra, egy eszköz vezérlésére vonatkoztatva kell ismerni.

Kétségtelen, hogy napjainkban nagyon fontos a robotok (és mesterséges intelligencia) programozásának ismerete. Valószínű, hogy az orvosi egyetem és az ETH Zürich biológia szakjain is többnyire az automatizált és mesterséges intelligenciával bíró eszközök kezeléséhez szükséges a programozás, az első évben, ezért ennek alapjait tanítják. Ugyanakkor a DigComp-ban leírt elvárások és egyes egyetemi képzések azt mutatják, hogy a digitalizáció a robotok (eszközök) programozásánál nem áll meg, ezért a problémamegoldási készség fejlesztését nem szabad leszűkíteni egy-egy robot vezérlésére. A digitális kompetenciáknak része a nagy rendszerek működésének értelmezése és nagy adathalmazokban tájékozódás is.

Egy gondolat erejéig visszautalva a bevezetésre: Dijkstra a `goto` programozásban való alkalmatlanságát akkor vetette fel, amikor már nagyobb, összetettebb problémákat kellett megoldani. [1][3] Ma is használják a vezérlésátadó utasításokat gépikódban és használják magasabb szintű nyelvekben is. A strukturált programozás elvei a tervezés során tiltják a vezérlés átadását, az átláthatatlan spagettikódok írását. A Python nyelv készítőinek is célja az átlátható, könnyen írható és olvasható kód,

amit nem a strukturált programozás szabványos lépéseivel, hanem az összetett problémák modulokra bontásával ér el. A robotika OOP alapú programjai jellemzően eseményhez rendelt egyszerű eljárások, az összetett rendszerek kicsi moduljai. A címkére ugrások helyett események vezérlik a program működését: egérrel kattintás, gomblenyomás, erős hang, időzítő `tick` eseménye…

Gonosz kérdés: A program érthetőségét mennyiben változtatja meg, ha sok `goto` helyett ugyanannyi függvénydefiníció van? A válasz kitérő: az egyes függvényeket jó programozók írták, tesztelték, ezért nem kell hibát keresni benne, jól működnek. Előregyártott megoldások bőséges kínálata megkönnyíti munkát [7] anélkül, hogy a részletekkel, a „nehéz matematikai absztrakcióval" foglalkoznunk kellene [8]. Ebből a megfontolásból következik, hogy nem fogunk belezavarodni a spagetti kódba, de nem is fogjuk tudni, hogy valójában hogyan működik a programunk.

Tényleg meg lehet tanulni úgy gondolkodni, mint egy „Computer Scientist" informatikus azzal, hogy Python nyelven programokat írunk?

Esetelemzés:

Egy csoportban C# nyelven tanítottam programozást, de az egyik diák inkább a barátjával – egyénileg – Python nyelven akart megtanulni programozni. Hagytam… Azt látták, hogy rengeteg forrás van mindenféle érdekes programok megírásának tanulásához. Egyik tanórán egy dátumok összehasonlítását is tartalmazó feladatot kellett megoldani. Felírtuk a megoldáshoz szükséges ciklust, elágazást és logikai formulát. A diák pedig hozta a sokkal rövidebb megoldását, amiben a Python három kész függvényét használta: „minek erőlködnek a többiek a megoldással, ez így érthetőbb és működik." Az ő kódja mindig rövidebb, frappánsabb volt, sokszor kombinálnia sem kellett, csak megkeresni a kész eszközt. Amikor a csoport haladó diákjai, a „menő programozók", a bmp fájl bináris olvasására és módosítására írtak programot, megváltozott a véleménye. A csoporttársai korábban tanult eszközöket (elágazás, ciklus, elemi adatok, tömb, osztály…) kombinálva módosítottak képeket, neki új eszköz megismerésére lett volna szüksége. A következő félévben már C#-ot tanult. A váltásnak lényegében az volt az oka, hogy hiába írt sok tucatnyi programot, nem tanult meg programozni, nem tanult meg úgy gondolkodni és kreatívan alkotni, mint a társai.

A korábban bemutatott példák alapján, azokon az egyetemi képzési helyeken, ahol a programozás általános problémamegoldási eszköz, a Python nyelv ismerete nem elvárt. Nem a Python nyelv az, amin a programozás alapjait, gondolkodásmódját megismerik, de konkrét területeken a nyelvet önállóan megtanulva, használják.

Konklúzió:

Úgy tűnik, a `goto`-hoz hasonlóan, a Python sok esetben jó, de az alapokat, a strukturált programozás általános elveinek elsajátítását nem támogatja. A megszerzett tudás nem transzferábilis.

4. Programozás oktatása a középiskolában

A nem informatikai egyetemi szakokon a bevezető programozás oktatás sokszor hiánypótló, felzárkóztató, kivételnek tűnik a fenti magyar példák közül az ELTE TTK programozás tantárgya. A DigComp, az ICDL minősítés is elsősorban a már felnőtt állampolgárok számára adnak iránymutatást, itt is fő cél a technológiai fejlődés követése, a lemaradók támogatása. A közoktatás, ezen belül a középiskola feladata azonban nem a jelenre, hanem a jövő szükségleteire való felkészítés. A Nemzeti Alaptanterv egy 12 éves képzési program, aminek a végére a végzéskor szükséges kompetenciákat kell biztosítani. A társadalmi-gazdasági problémát is okozó lemaradás hatására a 2020-as Nemzeti Alaptantervben a korábbinál hangsúlyosabban jelenik meg a programozás.

4.1. Hozott ismeretek

A 2020-ban bevezetett Nemzeti Alaptantervben a programozás, az algoritmizálással és robotikával ismerkedés már az óvodákban elkezdődik.

A Robot Turtles – magyar nyelvű kiadásban Robot Teknősök – társasjáték hagyományos, kartontáblás játék, nem kell hozzá elem sem. A játék során a gyerek tanul számlálni, a vezérlési struktúrák közül a szekvenciát és a modul kiemelését (több kártya helyettesítése eggyel) gyakorolja. Négyéves kortól, már óvodás korban ajánlott.

Az elemmel működő játékok között számos programozható játékot találunk, amelyeket kártyák helyett gombokkal, színes kockákkal, színes vonalakkal lehet vezérelni, így fejlesztik a programozási készségeket.

Robot Teknősök társasjáték[17]

Alsó tagozaton – a tantervkészítők elképzelése szerint – az egyre többet megértő játék robotok programozása során a gyerekek megismerkednek néhány fejlettebb vezérlési eszközzel.

Ötödik osztálytól a tankönyvi tananyagba [7] rejtve „Robotika, algoritmizálás, programozás" címen megjelenik a programozás. Ezen belül a pszeudokód, algoritmus, szekvencia, elágazás és ciklus fogalma. Programozási nyelvként a Scratch a leginkább ajánlott, de van utalás a magyar oktatásban nagy hagyományú Imagine Logo programra is. A programozás jellemzően a robotikával kapcsolatos feladatok megoldását jelenti, ami, ahol erre van mód akár 5. osztálytól kezdve micro:bit programozása is lehet. A tankönyv online „okostankönyv" verziójában [10] megoldási útmutató és működő program is megtalálható ehhez. A motivációt az elkészült játék adja, amelyet mintakövetéssel (megoldási útmutató) készít el a tanuló.

A felső tagozatos kerettanterv [11], a tankönyvi tananyagok (5.[7], 6.[12] és 7.[13] osztályra), ezek online okostankönyv feldolgozásai (5.[10], 6.[14] és 7.[15] osztályra) és feladatgyűjtemény[16] alapján felsőtagozat végére algoritmus, robotika és programozás témakörökben a diákoknak gyakorlati tapasztalatuk lesz az ICDL Kódolási ismeretek (Python) vizsgájának legtöbb témájában. A témakörök elnevezése is mutatja a hangsúlyáthelyezés irányát: 5. osztályban a „robotika" a cím első, később az utolsó eleme. Kiemelendő, hogy a programok nem karakteres – és így nem is Python –, hanem blokk alapú programozási nyelven készülnek. Az egyetemi bevezető/felzárkóztató programozás tantárgyak – az alapján, ahol elérhető a tananyag – az általános iskola végére elvárt ismeretektől csak a programozási nyelvben térnek el. Az ICDL modulhoz képest ezen túlmenően hiányoznak a tananyagból az elméleti definíciók.

A Python nyelv kezdő, motiváló nyelvként történő tanítását a felső tagozaton általában nem lehet kihasználni, mert a tanulók nem tudják kezelni a billentyűzetet. Ezért a kezdő nyelv egy blokk nyelv. A vizuális környezet, a vezérlési struktúrák és adatok értelmezést segítő jellegzetes alakú és színű blokkok a 4–14 évesek körében motiválók. Ebben a korban az elemek pakolása kevesebb frusztrációval jár, mint a billentyűk keresése, gépelési hibák javítása. (Frusztrációt okoz például a gyakori fókuszváltás, az, hogy a billentyűzetre le kell nézni.) Azonban haladóbb csoportokban a blokk-programozásról át lehet térni karakteres kódolásra. Erre alkalmas a Python nyelv, mivel Logo teknőc, micro:bit és Lego robot programozásához is van Python modul, így a már elkészített prog-

[17] RobotTurtles társasjáték kép forrása: kép forrása: https://www.thinkfun.com/products/robot-turtles/ [Megteintés: 2022.11.01]

ramok alapján elkészíthető a Blokk–Python szótár. A billentyűzet kezelés nehézségei is elviselhetőbbek, ha egy programon belül lehet váltogatni a programozási nyelveket, a blokk és kód nézetet.

Mivel a tantervet felmenő rendszerben vezetik be és a tankönyveket sem adják ki a bevezetés előtt, a 8. évfolyam programozás témaköre még nem ismert. Az alacsony (heti 1) óraszámra és a táblázatkezelés témakör súlyára tekintettel, elképzelhető, hogy robotikára, programozásra egyáltalán nem lesz idő 8. évfolyamon, a gondolkodási készségeket adatok kezelésén és függvények használatán keresztül fejlesztik. Ezt támasztja alá az is, hogy a tanterv 7-8. évfolyamos követelményeiben programozás témakörben javasolt tevékenység: „Típusalgoritmusok – összegzés, másolás, eldöntés, maximumkiválasztás – használatát igénylő programozási feladatok megoldása" inkább az algoritmus elvi vagy eljátszással történő megvalósítását sugallja, a blokk-programozás ilyen típusú feladatok megoldására nem optimális. Ehelyett motivációt jelenthet, hogy táblázatkezelésből a tananyagban megtaláljuk a típusalgoritmusokat megvalósító statisztikai függvények is.

4.2. Programozás oktatása a középiskolában, a tantervek alapján

A gimnáziumokra érvényes Digitális kultúra tantárgy kerettanterv [17] alkalmazandó a szakgimnáziumi képzésre is [18] A szakképző iskolákra külön, összesen 32 tanórában tanított Digitális kultúra tanterv [19] van. Ebben a programozásra 6 tanóra jut. Figyelemre méltó, hogy előzetes ismeretként elvárja egy fejlesztői környezet használatát, valamint, az is, hogy programot írni az összetett adattípusok és egyszerű algoritmusok értelmezése alapján kell; az alkotás, a kitalálás nem követelmény.

A hátrányos helyzetű középiskolás korú tanulók segítését célozza meg az Arany János Tehetséggondozó Program (AJTP). A program része a tanulók ECDL alapvizsgára történő felkészítése [20]. A programban heti kétórás tárgy az alap modulok vizsgáira készít fel [21], a három- [22] és négyórás [23] modul emellett az általános iskolai digitális kultúra tananyagnak felel meg, így a robotika, algoritmizálás és blokk programozás tanulására is van lehetőség.

Az új NAT bevezetése 2020-ban az 1. 5. és 9. évfolyamokon indult, jelenleg a középiskolákban digitális kultúra tantárgyat tanuló diákok nem tanultak digitális kultúrát az általános iskolában, így pillanatnyilag nem számolhatunk az általános iskolai digitális kultúra ismeretek meglétével. A 2012-es tantervben az Informatika óraszáma a 2020-as tanterv szerinti digitális kultúra óraszámának a fele volt.

A középiskolai tanterv erősen épít az általános iskolai követelményekre: „*A 8. évfolyam végére a tanulók a digitális írástudás alapjainak elsajátítását lezárták.*" [17 p. 4.] A tanterv bevezetője kiemelten foglalkozik a programozással: „*A programozás és algoritmizálás témaköreiben a tanulók új kihívással találkoznak. Míg korábban a blokkprogramozás segítségével gyakran közvetlenül vezéreltek eszközöket, most magasabb szintű absztrakciót igénylő feladatokat oldanak meg hagyományosnak nevezhető, azaz a programkód közvetlen beírását elváró fejlesztői környezetben.*" A robotika középiskolában nem digitális kultúra tananyag, valószínűleg a többi tantárgyban a robotika a tanulás egyik eszköze kellene legyen. Az algoritmizálás és programozás területén magasabb szintű absztrakció célul való kitűzése viszont azt jelenti, hogy az „alacsonyabb szintű absztrakció" meglétét elvárja. Tovább építkezik, ezért elmondhatjuk, hogy a 2020–2023-ban induló 9. évfolyam hátrányos helyzetű, számukra az AJTP-hoz hasonló kiegészítő képzés lenne szükséges a tantervi követelmények teljesítéséhez.

A felső tagozatos [11] és gimnáziumi tanterveket [17] összehasonlítva nagyon szoros egymásra épülés, a spirális (1-2 évente visszatérő ismétlő-fejlesztő témakörök) tantervi struktúra látható. Ezzel szemben feltűnő, hogy

- a 7–8. osztályos tanterv [11 p. 13] javasolt tevékenységei között szerepel a strukturált programozásra utaló típusalgoritmusok használata:

  „– *Vezérlőszerkezetek tudatos választását igénylő blokkprogramozási feladatok megoldása. Típusalgoritmusok – összegzés, másolás, eldöntés, maximumkiválasztás – használatát igénylő programozási feladatok megoldása*"

- de a 9–10. osztályos tanterv [17 p. 5] ezt elkerüli.
  „– *Típusok, változók és vezérlőszerkezetek (szekvencia, elágazás, ciklus) tudatos választását igénylő feladatok önálló megoldása, a választás indoklása*
  …
  – *Feladat megoldása során a fejlesztői környezet lehetőségeinek használata (pl. tesztelés)*
  – *Feladatmegoldás strukturálatlan algoritmussal és függvények, eljárások használatával.*"

Első olvasásra talán fel sem tűnik a „**strukturálatlan**" kifejezés. Lehetne elírás is, ha nem tudnánk, hogy informatikus körökben milyen viták folynak a `goto` a `break` és egyéb vezérlésátadási utasításokhasználatával kapcsolatban. A strukturált programozásban az algoritmus vezérlő szerkezetekből épül fel, ezek ismeretét és tudatos választását írja elő a 7–8. osztályos (és az 5–8. osztályos [11 p. 5]) tanterv. A micro:bit programozáshoz javasolt makecode[18] fejlesztőkörnyezetben a kimondottan strukturált programozást támogató blokkok között, a ciklusok csoportban megjelenik a „folytat" (angolul „continue") és a „kitörés" (angolul „break") blokk, de csak kontrolláltan, az adott ciklusra vonatkozva használható, így a teljes algoritmus nem lesz strukturálatlan, a szigorúan strukturált megoldás formális átírással (blokkok átszervezésével) előállítható. Informatikai, programozásmódszertani szempontból a strukturálatlan algoritmus alkalmazása káros, csak akkor fogadható el, ha a vezérlési struktúrák (elágazás, ciklus) nem állnak rendelkezésre.

Felmerülhet, hogy oktatási szempontból lehet pozitív hozadéka a strukturálatlanul kódolt megoldásoknak. Kérdéses, hogy a tananyagban hogyan jelenik meg a strukturálatlan algoritmus készítése, de az már most elmondható, hogy az általános iskolai tananyagra alapozva nincs létjogosultsága, az adott korosztálynak további javasolt tevékenységeihez nem kapcsolódik, a fejlesztési célok elérését nem támogatja. Elképzelhető, hogy egy diák strukturálatlan algoritmust készít, de ez legfeljebb része lehet a *Fogalmak* részben elvárt „tervezési folyamat"-nak, vagy tanulságos példája a tesztelésnek. A digitális kultúra tantárgyi követelmények teljesítése szempontjából egy strukturálatlan algoritmus programozásból annyit ér, mint szövegszerkesztés témakörben a szóközökkel középre igazított cím.

4.3. Programozás témakör megjelenése a Digitális kultúra tankönyvekben

A digitális kultúra tantárgyban az oktatásirányítás szándéka szerint az informatika oktatása (is?) új megközelítéssel szerepel. A tantárgy 3–11. évfolyamokon átívelő tantervét 2020-ban a 9. évfolyamon úgy vezették be, hogy a megelőző 6 év tananyagának áttekintésére, a tantervi követelmények fokozatos bevezetésére semmilyen központi terv nem volt. Tovább rontotta a helyzetet, hogy a tankönyvek is csak a már aktív évekre készültek el, így 2020-ban az 5. és 9. évfolyamra készült tankönyv. Ennek ellenére több témakörben a 9. évfolyam tananyaga épít a korábbi években tanultakra. Ez részben azért megalapozott, mert az adott témakör tanítását elvárták az Informatika tantárgyban is, másrészt azért lehetséges, mert az Informatikából elvárt minimum követelmények nagyjából megfelelnek a Digitális kultúra tantárgyban elvárt minimumnak. Praktikus megfogalmazással: lehet, hogy Informatikából a jeles Digitális kultúrából csak közepes, de aki Informatikából elérte az elégségest, annak ez a tudás Digitális kultúrából is elégséges.

Az Algoritmizálás, formális programozási nyelv használata [17] tantervi téma előzménye – hasonlóan a többi témakörhöz – az előző NAT felső tagozatos Informatika tantárgyában is megtalálható, az új tantervben ennek blokk nyelvvel kibővítése jelenik meg. Azonban a 2012-es kerettantervben az oktatásához nem volt óraszám, így jellemző, hogy egyáltalán nem tanulták a diákok, az átmeneti időszakban a Digitális kultúra tantárgyban 9. évfolyamon nem lehet az általános iskolai

[18] makecode: https://makecode.microbit.org/

ismeretek meglétében bízni. Sajnos, a problémára nem egy átmeneti tananyag elkészítése lett a megoldás, hanem a középiskolai tananyag függetlenítése az általános iskolában tanultaktól.

Amennyiben a következő évben megjelenő 8. évfolyamos digitális kultúra tankönyvben van utalás a blokk nyelvről Python nyelvre áttérésre és a gimnáziumi tananyag építene az általános iskolában tanultakra, akkor ésszerű folytatás lehetne programozási nyelvként – legalábbis kezdetben – a Python nyelv választása. Az 5–8. osztályos Digitális kultúra feladatgyűjteményben – sem a programozás fejezetben [16], sem máshol – nem szerepel a Python szó. Ezért a megelőző tanulmányok nem indokolják, hogy a gimnáziumban a Python nyelv legyen a programozás nyelve. A Python nyelv választásának másik lehetséges magyarázata: „a Python nyelv jó a kezdőknek, motiváló, hogy könnyen lehet benne programot írni" szintén nem állja meg a helyét, hiszen az első 4 évfolyam kivételével nem lesznek teljesen kezdők a diákok.

Az Algoritmizálás és (formális) programozási nyelv használata fejezetek 9. évfolyamon pdf [24] és okostankönyv formában [25] elvileg teljesen kezdőknek szól, a folytatás a 10. évfolyamon [26] [27] majd 11. évfolyam végére a [28] [29] tananyaggal lefedi a tantervi követelményeket.

Az okostankönyvek [25] [27] [29] statikus része megegyezik a pdf formátumban megjelent tankönyvek tartalmával (az elírások is), de az okostankönyvekben minden oldalon van 2-3 interaktív feladat is. Ezek sokszor a tananyagban szereplő új fogalmak gyakorlását szolgálják, máskor kiegészítik azt. A plusz feladatok és a szintén online megjelent feladatgyűjtemények [30][31] is nagyban segítik a tananyag elsajátítását, de egyben rámutatnak a tankönyv hiányosságaira is.

Ilyen kiegészítés a 9.-es okostankönyv [25] elő leckéjének (Mi az a programozás?) végén, a lecke utolsó tartalmi egysége (Kérdések) után található kipróbálható példa, ami a micro:bit programozásának Python nyelvű változatát mutatja be. Ha lenne visszautalás a korábbi évekre, akkor ezen a példán értelmezni lehetne a változó, a 10. évfolyamon tárgyalt eljárás és a 11. évfolyam végén érintett eseménykezelést is, alátámasztva, hogy jó döntés lehet a Python nyelvet választani az általános iskolai tanulmányok után is. Ha lenne…, de nincs.

Az okos(feladat)gyűjtemény két programozás fejezete pedig azt mutatja, hogy a téma feldolgozásának több módszere is lehet. Egyrészt a Python mellett megtalálható a C# nyelvű mintamegoldás is, másrészt az „Általános irány" [30] az adatok, vezérlési szerkezetek, típusalgoritmusok (programozási tételek) mentén csoportosítva jeleníti meg a feladatokat, míg a „Tehetséggondozó irány" [31] problémakörökre bontva, az egyes feladatokon belül – akár hosszú távú projektként – egyre több ismeretet igénylő részfeladatokat tartalmaz.

Konklúzió:

A rendelkezésre álló forrásokból látható, hogy a tankönyv csak egy példa a témakör feldolgozására, azonban ennek a feldolgozásnak biztosítania kell a tantervi követelmények teljesítését (az Oktatási Hivatal által akkreditált tankönyv). Az oktatási cél az algoritmizálás, a strukturált programozás alapjainak megtanítása, az ehhez választott implementálási eszköz a Python nyelv.

5. Python nyelven tanított algoritmizálás

A Digitális kultúra tantárgyhoz írt 9–11. évfolyamosoknak szóló tankönyvek és okostankönyvek Algoritmizálás és (formális) programozás fejezetei rögzített tanítási/tanulási útvonalat, tanítási tervet adnak. Mindegyik tankönyvhöz tartozik tanmenet, ami a tankönyvbéli fejezeteket adja meg egy-egy tanóra címének. A tankönyv megjelenésének formájából következően a témakör tárgyalására kötött volt a terjedelem, ezért több helyen rövidíteni kellett rajta. Ez eredményezte azt, hogy a tankönyv sokkal többet foglalkozik a Python nyelvvel, a programozás eszközének ismertetésével, mint az elméleti alapokkal, magyarázatokkal. A keletkezés körülményeitől függetlenül, a tanárok és diákok csak a végeredményt látják, ez alapján kell tanítaniuk, tanulniuk.

Ebben a fejezetben azt tekintjük át, hogy a tankönyv hogyan segíti a kerettantervben [17] előírt tevékenységek megvalósítását és a kompetenciafejlesztést.

5.1. Adattípusok

5.1.1. A karakter nem szöveg

Tanterv: „*ismeri a következő elemi adattípusok közötti különbségeket: egész, valós szám, karakter, szöveg, logikai;*" [17 p. 4]

A Python nyelv nem ismeri a karakter adattípust, pontosabban a szöveg adattípus karakterlánc, de a lánc eleme nem jelent külön típust, az is „lánc". Ez némiképp ellentmond a matematikában már tanult halmazelméleti fogalmaknak, ott egy egyszerű halmaz eleme nem halmaz.

A karakter a 11-es tananyagban kap némi értelmet [28 p. 171]: „*Most, hogy már úgy írunk és olvasunk fájlokat, mintha mindig is remekül ment volna, ki kell térnünk egy fontos részletre. A számítógép a karaktereket számkóddal tárolja, mind a memóriában, mind a fájlokban.*" Úgy tűnik, hogy a bájt, a bit, a bináris adattárolás fogalmát misztifikálni kell. Mivel minden adatot bitekkel tárolunk és a bitek véges mennyiségével kell leírnunk, ezért ezek a bináris jelek (binary digits) 1 és 0 értékekkel értelmezve kettes számrendszerben „számkód"-ot eredményeznek. Valójában egy jelsorozat nem eredményez semmiféle „számkód"-ot, a felhasználás módjától függ, hogy karakternek vagy számként, ezen belül előjeles vagy előjel nélkülinek értelmezi a program.

A karakter „számkód" értelmezésére azért tér ki a tankönyv, mert a Windows és a Python között alapértelmezésben nincs meg az összhang a karakterkódolások értelmezésében. A probléma a tankönyv szerint a Windows hibája, a háttérben meghúzódó adatmodellezési kérdéseket nem tekinti át, inkább elkerüli az ékezetes karakterek használatát a fájlokban. Nemhogy 10. osztály végére [17 p. 4], de a tananyag befejezésekor se lesz tiszta, hogy mi a különbség a szöveg és a karakter között, például, hogy a szöveg – mint karaktersorozat – lehet üres, de a karakter nem. A szöveges fájl beolvasása során épp csak említés szintjén megjelenik az 1 karakter beolvasása (read(1))[28 p. 172], ami sikertelen olvasás esetén lehet üres szöveg is. Sehol nem jelenik meg, hogy az adatoknak mérete lenne, hogy a memóriában tárolása valahány bájt lefoglalásával jár.

A karakter adattípus a kódtáblákon keresztül természetes kapcsolatot jelent az adatmennyiség fogalmának értelmezéséhez. Megmutatja a rögzített, nyolcbites tárolási mód korlátait, a programoknak a számítógép-architektúrával (pl. 64 bites processzor), az elektronikával, robotikával való kapcsolatát. A Python nyelv ennek megtapasztalásától elzárja a használóját, jelen esetben a tanulót.

5.1.2 Az adattípus fogalom kialakítása

A Python nyelv dinamikusan típusos. A változók típusát az értéke határozza meg. „*Leegyszerűbb, ha a változókra olyan dobozként gondolunk, amibe bármit tehetünk*" A változónak ez a fajta értelmezése elfedi a digitális világ alaptéziseit. Két dolgot nagyon pontosan kellene látnia a tanulónak: a „bármi"-nek fizikai mérete van, méghozzá nyolcszor annyi elemi tárolócellát foglal le, mint ahány bájtos, továbbá, minden tárolóhely véges, így a doboz mérete is véges. A dinamikus típusosság esetén vagy végtelen nagy méretű doboz kell, vagy nagyon korlátos a „bármi", vagy a változónak nem csak az értéke változik, hanem a doboza is. A változó típusának módosulását már az első bemutatott példa [24 p. 104] is kihasználja:

```
"
IDEI_ÉV = 2021
felhasználó_kora = input('Hány éves vagy? ')
print('Te most ', felhasználó_kora, ' éves vagy.')
felhasználó_kora = int(felhasználó_kora)
születési_év = IDEI_ÉV - felhasználó_kora
print('Ekkor születtél: ', születési_év, '.', sep='')
```

…Logikusnak tűnik egy `ilyen=input('És milyen', felhasználó_kora, 'évesnek lenni?')` *megoldás… A hibaüzenet ismét int-ről és str-ről beszél, és ekkor már gyanítjuk, hogy az lehet a baj, hogy a felhasználó_kora a 4. sor óta egész szám,…*"

Mennyiben segíti a változó fogalmának megértését az, hogy ugyanazt a változónevet használjuk különböző típusú adatokra? Másik adattípus, más memóriaméret. Hogyan segíti a programozás megértését, hogy az adattárolás és adatra hivatkozás pontos ismerete, a fogalom pontos megértése előtt „varázsolunk"?

„*A programozónak fontos a jó változónév, hogy ha holnapután előveszi a programját, még mindig el tudjon igazodni rajta. A változónevek választásával a program értelmezését segítjük.*" [24 p. 104] olvashatjuk a változónevek megadásának szabályai után. Vajon a beszédes névhez miért nem tartozik hozzá az is, hogy nem adom ugyanazt a nevet különböző típusú adatoknak?

A Python nyelv a dinamikus típusosságával támogatja, hogy kevesebbet kelljen gondolkodni a változónév kiválasztásán, de

- a hibás adattípus használata csak futtatáskor derül ki;
- a kód későbbi értelmezését megnehezíti a típus változékonysága;
- az adat, adattípus és változó fogalmak megértését felületessé teszi.

Ezért a dinamikus típusosság kihasználása a tanulás során káros, nem felel meg a tantervi céloknak, pusztán a Python nyelven beszélők – „mi így szeretjük" – közösségtudatának erősítését szolgálja.

Egy adattípust két dolog határoz meg: az adat tárolásának mérete és a rajta végezhető műveletek. Az adattípusnak (például karakternek) Pythonban – a dinamikusságot biztosító indirekció miatt – nincs értelmezhető mérete, ezért alkalmatlan az adattípus fogalmának tanítására.

5.1.3. Adatsorozatok

Tantervben 9-10. évfolyam: „*Az elemi adatok és sorozatok megkülönböztetése, kezelése és használata …Fogalmak: … sorozat …*" [17 p. 5]. 11. évfolyam: „*Az elemi és összetett adatok megkülönböztetése, kezelése és használata, Fogalmak: vektor*" [17 p. 16]

A sorozat és vektor két kifejezés, feltételezhetjük, hogy két különböző adatsorozatot jelent. A sorozat (sequence, range) összefügg a sorban léttel, egymásra következéssel, míg a vektor (array) a dimenzióval, aminek kétféle értelmezését használják: a vektor ahány dimenziós annyi adat, illetve annyi kiterjedtség. Számítógépes megvalósításra is többféle mód létezik. Kérdés, hogy melyeket célszerű tanítani, melyek azok, amelyeknek az ismerete hozzájárul a természeti és társadalmi környezet megértéséhez. A tankönyvben a Python nyelvben preferált lista lett az egyik adatsorozat: „*Ilyen például a Python lista adattípusa.*" Majd rögtön az egynemű adatok láncolásának bemutatása után egy specialitást is megemlít: „*Nem kell egyforma típusú adatokat tenni egy listába, azaz teljesen jó a csata = ['Isaszeg', 1849, 'április', 6, 'magyarok']*" [24 p. 120].

Láthatjuk azt is, hogy a lista mindenre jó csodaeszköz. Egyszerre valósítja meg a struktúrát és a láncolt listát. „*Akiben felmerül a kérdés, hogy „És hogyan tárolnám a tavaszi hadjárat összes csatáját egyetlen listában?", azt megnyugtatjuk, hogy lehet egy lista eleme egy másiknak. Aki ettől megijed, azt is megnyugtatjuk: ebben a könyvben nem foglalkozunk ilyen listákkal.*" [24 p. 120] Valóban, „ebben a könyvben" nem szerepel ilyen lista, de később annál inkább. Egy évvel később megjelenik a kétdimenziós adatszerkezet: „*A kétdimenziós adatszerkezetek használata nagyon gyakori – pont úgy, ahogy a táblázatoké a valóságban.*" Azonban a táblázat megvalósítása „listában listával" történik, így a fogalomzavar még egy elemmel bővül: a lista egyszerre láncolt lista, struktúraszerű és véges méretű tömbszerű (táblázat).

Valószínűleg eddig tart a „Pythonban könnyű megérteni a programozást" motiváló hatása. Az egyetemi programozásoktatás során a csak Python-t tanult hallgatókra jellemző, hogy nem értik, mi a tömb, mi a struktúra. Az ismereteik elégtelenségének felfedezése elkeseredést vagy lázadást vált ki.

Pythonban a lista „kézreálló", a használata könnyű. Bizonyos típusú feladatokra. A tankönyvből megtudjuk, hogy mire nem praktikus a lista: „*Kihívást jelentő feladatok, ahol nem segít rajtunk a* `count()`*, és mindenképp be kell járnunk a listát: a. Hány helyen előzi meg a hatos dobást ötös dobás? b. Hány helyen van egymás után két hatos?*" [24 p. 126]

A „kihívás", a feladattípus nehézségének az oka, hogy a listában az adatoknak nincs pozíciója, indexe. Nem mindegy, hogy az adatokat tároló „dobozok" egymáshoz illeszthetőek, mint az irattartó papusok vagy egy kartotéktároló szekrény fiókjaiként képzeljük el. Mi jelenti a kihívást: a másmilyen struktúra vagy a használt nyelvi eszközzel a struktúra implementálására?

Azok a feladatok jelentenek kihívást, ahol nem az adatsorozatban lévő adat (objektum) tulajdonságáról, állapotáról szól a kérdés, hanem a többi objektumhoz képest a helyéről. Megszemélyesítve: a „helyem a világban" megadható úgy is, hogy az én tulajdonságom a „gps-koordináta" értékem és úgy is, hogy a föld egy adott gps koordinátájú pontjára azt mondom, hogy „itt a helyem" Ez utóbbihoz szükséges egy térkép, ami a helyeket tárolja. Ennek digitalizált megfelelője a tömb (vektor, array, mátrix). A kétféle megközelítési módot a diákok számára is érthetően szemlélteti a pixelgrafikus és vektorgrafikus képmegjelenítés, mivel mindkét absztrakciós módszert tanulták már 10. évfolyamos korukra.

A láncolt lista és tömb összemosása kritikussá válik a kétdimenziós adatstruktúrák tárgyalása során: „*Mik azok a kétdimenziós adatszerkezetek? Az egyszerű listák egydimenziós adatszerkezetek – azaz csak hosszuk van, mint egy szakasznak a geometriában. Azonban a listákban elhelyezhetünk olyan elemeket is, amelyek saját maguk is listák. Így lesz az adatszerkezet kétdimenziós: van „szélessége" és „magassága."… A kétdimenziós adatszerkezetek használata nagyon gyakori – pont úgy, ahogy a táblázatoké a valóságban.*" [26 p. 81]

Itt a valós dimenzió fogalmát vetítjük a digitális adattárolásra. Ha a szakaszról van szó, azt digitálisan az egy dimenziós tömb (vektor) valósítja meg. A lista ebben az értelemben félegyenesnek, esetleg egyenesnek felel meg. Kétdimenzió esetén a téglalap alakú terület a tömb, míg a síknegyed a listában lista megfelelője. Aminek van kerete, – ez adja a szélességét és a magasságát, – az tömb jellegű. Aminek nincs határa, bővíthető, az a megfelelő dimenziók jellege alapján lehet akár listában lista is, de a tömb és lista keveréke is. A 10. évfolyamos diák számára ismert tömb a rögzített méretű táblázat. Például:

- A megjelenített kép egy 2D pixel tömb. Minden vektorgrafikusan tárolt képet végeredményben egy adott szélességű és magasságú pixel tömbben látunk.
- A táblázatkezelő munkalapja nagy, de véges méretű 2D tömb.
- A sakk, sudoku, keresztrejtvény… játékok.
- Ha tanult általános iskolában is digitális kultúrát, akkor ismeri a micro:bit 5×5-ös LED-mátrixát, de ha nem, akkor is számtalan jól érzékelhetően táblázatos elrendezésű LED-mátrixot lát a környezetében, például tömegközlekedési eszközök tájékoztató tábláit, reklám táblákat.

Ezekre az adatstruktúrákra könnyen tud kérdéseket és feladatokat megfogalmazni. Érti, hogy mit jelent a felette, az alatta, a jobbra, a balra, a „lóugrás", adott esetben a 90°-os elforgatás, tükrözés. A „listában lista" megvalósítása során a belső listák különbözőek lehetnek (Pythonban még az adattípusok is), így szemléletben a fel-le lépés és a jobbra-balra lépés közül csak az egyik értelmes. Míg tömb esetén a „sorfolytonos tárolás" logikus implementáció, a listában listára ez, és ezzel együtt a 2D táblázat értelmezése 1D vektorként is értelmetlen. A lista absztrakció a „dolgokat egymás után teszünk", sorozatképzési feladathoz köthető. A „listában lista" gyakorlati előfordulása például egy feladatlista, aminek minden tétele mellé odaírjuk, hogy milyen eszközök szükségesek hozzá. Egy ilyen struktúra megalkotása, konkrét eset elképzelése (fejben megalkotása) nehéz. Ezzel szemben a

2D tömb absztrakciója lényegében a létünk része: egy szem a világról 2D tömb képet érzékel. A szemidegek valószínűleg nem négyzetrácsban helyezkednek el, de nincs az információfeldolgozásban irány szerinti külső-belső rendelés. Másik oldalról közelítve: az egymás mellettiség és egymásfölöttiség is kapcsolat, míg a todo listában – ha egymás alá helyezem a „kis listákat", akkor ezek között nincs az egymás fölöttiek között kapcsolat.

A tömb listában listaként történő értelmezése ezek szerint egy programozói trükk, aminek megtanulását elvárjuk egy programozásban kezdőtől. Ezzel ellentmondunk a Python fejlesztők „hitvallásának" [32], azaz olyan elvek mentén tanítjuk a programozást, amit az éppen tanított nyelv fejlesztői károsnak tartanak:

(2.) „*Explicit is better than implicit.*" A kifejtett jobb, mint a hozzágondolt.

(3.) „*Simple is better than complex.*" Az egyszerű jobb, mint a bonyolult.

(5.) „*Flat is better than nested.*" A sima jobb, mint a beágyazott.

(7.) „*Readability counts.*" Az olvashatóság számít.

(8.) „*Special cases aren't special enough to break the rules.*" Nincs annyira speciális eset, hogy felülírja a szabályokat.

(17.) „*If the implementation is hard to explain, it's a bad idea.*" Rossz az ötlet, ha a megvalósítást nehéz elmagyarázni.

Bár a fenti kijelentések szakemberek számára irányadóak, oktatási és nevelési szempontból nem fogadható el, hogy a felhasználó és esetleg jövendő szakember képzésében más elveket közvetítsünk. Nem taníthatjuk diákjainknak, hogy a Föld lapos, ugyanígy, a programozáshoz hozzá tartozik a kétdimenziós intervallum digitális absztrakciója, mert amit a digitális világból a szemünkkel érzékelünk, az többnyire egy korlátos kétdimenziós tömbre van leképezve.

A fenti elveket figyelembe véve, a kétdimenziós tömb nem helyettesíthető „listában lista" struktúrával az oktatás során. Ugyanakkor, mivel alkalmazásával folyamatosan szembesül a diák, a 2D tömb megismerése a körülöttünk lévő világ értelmezéséhez szükséges. Ebből következően, vagy a Python nyelvben is meg kell valósítani a tömb adattípust, vagy más programozási nyelvet kell választani.

A programozók pontosan tudják, hogy a programozási nyelvek adott feladatok megoldására íródnak. A Python bizonyos feladatokra nagyon könnyen használható nyelv, amely elrejti a részleteket, ezzel segít a probléma megoldására koncentrálni. Pontosan ez, az általában hasznos a tulajdonsága teszi alkalmatlanná arra, hogy a diákok megismerjék a géphez egy absztrakciós szinttel közelebb álló adatstruktúrákat.

5.1.4. Rekord és osztály

A tantervben a rekord (vagy struct) adattípus nem szerepel. Adatbáziskezeléshez kapcsolódva találjuk meg a strukturált adattárolás ismeretét. [17 p. 12-13, 15-16; 21] Az osztály (class) ismerete sincs nevesítve, csupán az objektum fogalmán keresztül: fejlesztési feladat 9. évfolyamon: „*Az objektumorientált szemlélet megalapozása*" [17 p. 5] majd 11. évfolyamon szintén, valamint a fogalmak felsorolásánál: „*Objektumorientált szemlélet … eljárás, függvény, kódolás, objektumorientáltság, típusfeladatok, tesztelés*…" [17 p. 16].

A tanterv alapján nem szükséges a rekord, struktúra adatszerkezet ismerete, másrészt érteni kell az OOP alapvető megjelenési formáit, az objektum fogalmát, létrehozás, a tulajdonság és metódus fogalmakat. A fogalmak sorrendjéből következően vélhetően elvárt a típusfeladatok megoldása objektumok sorozatára is, de nem tananyag az öröklés. Ehhez képest vizsgáljuk a tankönyv releváns tartalmait.

A tankönyvekben több fejezetben szerepel az objektum kifejezés. Objektumokkal dolgozunk szövegszerkesztés és grafikák készítése során. A Python objektumorientált paradigmát követ, ezt hangsúlyozza a 9. évfolyamos tankönyvben a lista-objektum és a range-objektum kifejezés [2428 p. 123–124]. A range-objektum külön érdekessége, hogy ez lehetne egy vektor, de nem hivatkozunk rá változónévvel. Emellett a lista általában inkább adatsorozat. Úgy tűnik, adatsorozat-objektum. Ezzel a szemléletmóddal érünk az „Objektumok adatai kétdimenziós listában" fejezet példafeladatához: „*Egy osztály tanulóinak adatait tároljuk egy 2D-listában. Egy-egy kis lista egy-egy objektumról, azaz egy-egy diákról szól. A kis lista elemei, azaz az objektumainkról tárolt tulajdonságok: név, nem, kor és e-mail-cím*" [26 p. 84]

Egyrészt, a lista egy programozásban használt, bejárható objektum, aminek a tulajdonságairól egy szó sem esik. A 2D-lista vélhetően „objektumban objektum", a „kis lista" is objektum, ami egy objektumról, a diákról szól. A lista-objektum elemei a diák-objektum tulajdonságai – vagy a cím alapján az objektum adatai.

Innen egyenes út vezetne az osztály fogalmához, a példányosításhoz, azonban csak egy évvel később, lényegében kiegészítő ismeretként kerül elő a téma: „*De mi az az objektum?*" majd a meghatározás: „*Az objektumok egy-egy létező, azaz egy-egy személy, tárgy, fogalom számítógépes programban létrehozott reprezentációi, megvalósulásai. A létező lehet bármi: diák…*" [28 p. 202]

Egészen a 11. évfolyamos tananyag végéig az objektum a programozáson kívüli „létező bármi", értelmű, csak ekkor vált át arra, hogy az objektum egy programbéli reprezentáció. Tekintettel arra, hogy a tantervben nem hadászati témakörben, hanem az adatok és programozási nyelv használata témakörben szerepel az objektum szó, az objektumorientált szemlélet nem a célszemély tárgyiasításáról, hanem az objektum tulajdonságainak számítógépes reprezentációjáról kellene, hogy szóljon.

A Python nyelv elrejti az adatot a felhasználó elől, a tankönyvi feldolgozás ezt továbbviszi, amennyire lehet: az objektum belső struktúráját is elrejti. Ahelyett, hogy az egyes tulajdonságokhoz megnevezéseket (és adattípust) társítana a változó-érték fogalompár felhasználásával, kulcs–érték párokkal, objektum–objektum modellt épít csak azért, hogy könnyebben olvasható legyen a kód: „*Előrebocsátjuk, hogy a szótár adattípus nem szükséges abban az értelemben, hogy minden, amire a szótár típus használható, megoldható listák használatával is, de a szótárak használata olyan sok esetben teszi kényelmesebbé a munkánkat, hogy kifejezetten érdemes megismerkednünk vele.*" [26 p. 85]

A szótárra természetes példa egy angol-magyar szópár lista, de pont ez a példa alkalmatlan a szótár specifikálására, hiszen mindkét nyelven számos szinonima teszi pontossá egy fogalom megfelelő megnevezését e két nyelven, ami a kulcs ismétlődését jelentené. De az alfejezet címéből tudjuk, hogy vagy a fogalom, vagy a magyar és angol szó egy objektum: „*Objektumok szótárban*" [26 p. 86].

Ezt követi az „*Osztálytagok a szótárban*" [26 p. 87] fejezet. A címnek többféle értelmezése is lehetséges, de a folytatás nem tisztázza, hanem érthetetlenné teszi a fogalmakat.

„*A szótárakat nagyon gyakran használjuk arra, hogy egy-egy objektum tulajdonságait tároljuk egy szótárban. Az előző lecke utolsó feladatában, ahol az osztálytagok adatait – tulajdonságait – tároltuk, a tárolást listával oldottuk meg.*" [26 p. 87]

Ez alapján azt várnánk, hogy van egy feladat, amiben egy osztály tagjait, azaz a diákokat tároltuk listában, most diákok szótára lesz. A végére az lesz, de előbb diák tulajdonságaiból képez szótárat. Minden diáknak lesz 'nev', 'nem' és 'kor' kulcsú szótárbejegyzése a megfelelő értékekkel. Ebben a megközelítésben a 'név', 'nem' és 'kor' objektum, hiszen a kulcs és az érték is objektum. Ugyanakkor a class adattag fogalmához hasonlóan adja meg a diák tulajdonságait:

```
{'név': 'Noémi', 'nem': 'l', 'kor': 15}
```

Minden diákra elkészítve a szótárt, az osztály a diákok, azaz az osztálytagok listája, adatszerkezete szótárbejegyzésekből épített szótárak listája.

Úgy tűnik a 10. évfolyam (heti 1 órája) a bátorságról szól. Az egy évvel korábban ijesztő listában lista mellé jön a szótár, de nem eredeti értelmében, hanem struktúra (vagy class) helyett, ráadásul rögtön listában alkalmazva. De ez sem elég: „*Legyünk bátrabbak*…" és már jön is a szótárban szótár.

```
osztály = {'Noémi': {'nem': 'l', 'kor': 15},
           'Dezső': {'nem': 'f', 'kor': 17},…
```

Miért kell a szótár? Miért kell összekeverni az objektum, az objektum számítógépes reprezentációja, a tulajdonság, változó, adat, kulcs, érték fogalmakat? „*Ezért küzdöttünk!*" [26 p. 88]

```
print(osztály['Dezső']['kor'])
```

Sok bátorság, fogalmak zagyvasága árán eljutunk egy jól olvasható leíráshoz. Az eszköz – a szótár – nem tananyag, csak egy újabb Python objektum. Az objektum mibenlétéről nem tudunk meg semmit. Valójában érteni sem kell. Ezen a ponton a tanár és a diák is vagy a nyelvi eszköz megtanulását választja, vagy marad a listában lista.

Nem csak a programozás alapjainak megértését nehezíti (vagy gátolja) meg a szótár bevezetése, de a tantárgy más témáival való kapcsolódás szempontjából is káros a struktúra (vagy class) ilyen formán történő megkerülése. 10. évfolyamon – a tankönyvben a programozás után, de valójában valamikor a tanévben – az adatbázis-kezelés alapjainak megismerése is tananyag.

„*Képzeljünk el egy osztályt, a 10.g-t! Ott van a barna hajú Molnár Ricsi, aki 160 cm magas és 59 kg. Padszomszédja… Ezeket … nevezzük adatoknak. Egy-egy tanuló adatainak összességét… rekordnak hívjuk, a rekordok egyes adatait mezőnek, az adatok megnevezését pedig mezőnévnek.*" [26 p. 90].

A 10. évfolyamos tankönyv (84–88.) öt oldalán egy bevallottan kihagyható levezetéssel eljutunk odáig, hogy jól olvasható egy osztály tagjainak néhány tulajdonságára történő hivatkozás. Majd kétoldal gyakorlófeladatot követően, ugyanarra a feladatra (diákok tulajdonságai) egy, az eddig bemutatott gondolatmenethez, fogalom-rendszerhez és leírási módhoz semmilyen módon nem kapcsolódó fogalmak, definíciók sorakoznak. A digitális világ kettéhasadt: van a táblázatkezelés és adatbáziskezelés az adatokkal, táblákkal, és van a Python nyelven programozás, listában listával, szótárban szótárral. A kettő között nem látszik fogalmi kapcsolat. Ezzel a Python nyelven programozás és fogalomhálója elszigetelődik a többi témakörtől, ezért a tanultak a dolgozat után elfelejthetőek. A programozás ismeretekből annyi marad, hogy Pythonban írt kódot a diák.

5.1.5 OOP és osztály

A tanterv a 11. évfolyamra előírja a továbblépést a grafikus felületen futó programok készítése irányába, javasolt tevékenységként elvárja bizonyos objektumok létrehozását és tulajdonságainak módosítását:

„– *Az alapvető vezérlők használata: címke, nyomógomb, szövegmező, jelölőnégyzet, rádiógomb a felhasználói felület programozásában alkalmazói jellegű feladatok során (pl. megrendelés beviteli felülete)*

– *Alapvető grafikus vezérlőelemek létrehozása és használata a felhasználó felület programozásában*"

A „saját adatszerkezet", azaz a class definiálásához ezúttal is egy osztály (mostantól csoport) egyes adatai jelentik a példát, a példa megoldását először listában szótár struktúrával láthatjuk, de választható lenne a listában listák is. [28 p. 197–202.] Ezt követi az osztállyal kapcsolatos fogalmak tisztázása. Továbbra is a Python nyelv specialitásai elsőbbséget élveznek az általánosan használt szakmai terminológiával szemben: „*Úgy mondjuk, hogy ilyenkor történik meg az objektum inicializálása (magyarra előkészítésnek fordíthatnánk, de nem szokás fordítani). Néhány más programozási nyelvben az osztályok, objektumok hasonló szerepű függvényét konstruktornak nevezik, és ez az elnevezés olykor átszivárog a Python világába is.*" [28 p. 203]. – Ez az egyetlen említése a konstruktor kifejezésnek.

A konstruktorhoz hasonló sorsra jut a tulajdonság és adattag fogalom is: „*Az objektumokat ebben a leckében még csak arra használjuk, hogy a tárolt létezők bennünket érdeklő tulajdonságait tároljuk bennük. … Az*

ilyen tulajdonságokat ebben a könyvben jellemzőnek nevezzük (szokás még többek között a mező, az adattag és a változó elnevezést használni)." A „jellemző" valószínűleg az „adattag" megnevezést helyettesíti. Nem esik szó arról, hogy az OOP egyik alapelve az egységbe zárás, de felhívja a figyelmet arra, hogy ha az objektumnak nincs saját függvénye, akkor a szótár és lista elég.

Bár a tanterv csak a grafikus képernyőn tipikusan használt objektumok programozását várja el, a tankönyv alapján ehhez gyakorlatot kell szerezni tagfüggvények írásában, meg kell ismerkedni a kulcsszó-paraméteres függvényekkel és a modulkészítéssel is. Mindezt lényegében egy-egy mintán bemutatva, majd szintén példaként, azok, akik még értik, hogy miről van szó és eléggé érdeklődőek, lemásolhatják két grafikus alkalmazás kódját. A tantervben szereplő javasolt tevékenységek így legfeljebb kiegészítésként, a legérdeklődőbb, kitartó diákok számára lesz megvalósítható. [28 p. 219–230.] A grafikus felületen futó alkalmazások programozását tipikusan kész sablonokból, a programkód módosításával szokták tanítani. Itt ehelyett egy ennél jóval nehezebb, megtanulásra alkalmatlan módszert látunk, két grafikus felület elkészítését. A diák nem arra lát példát, hogy hogyan tudja más környezetben alkalmazni eddigi tudását, hanem arra, hogy hogyan lehet elkészíteni egy másik alkalmazási felületet.

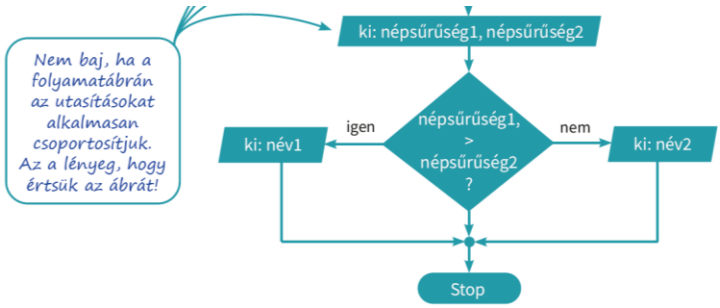
5.2. Vezérlési szerkezetek

Ahogy az adattípusok, úgy a vezérlés alapelemei és a típusalgoritmusok tárgyalása is a Python nyelvhez igazítva jelenik meg, előtérbe helyezve a nyelvi specialitásokat a tantervben elvárt ismeretekhez képest.

5.2.1. Elágazás

A Python nyelv specialitásai elsődlegesnek tekintése más programozási nyelvekhez, technikákhoz képest mutatkozik meg az elágazásban használható 'elif' összevonás kapcsán. Az egybeírt módot a mondatszerű leírás (pszeudokód) nyelvére is visszavetíti: „*Elsőként az algoritmus mondatszerű leírását módosítjuk. Az új, „különbenha" ágat megvalósító Python-utasítás az elif.*" [24 p. 110].

A kettőnél több ágú elágazás egybevonása folyamatábrán is megjelenik. Az egyik feladatban a folyamatábra alapján kell elkészíteni a kódot. Ennek részlete az alábbi ábra: [26 p. 47]



Az elágazás – a mintamegoldás alapján – nem hibás, hanem azt akarja szuggerálni, hogy legyen kiírás az egyenlőség esetére is.

5.2.2. Ciklus

A Python nyelv kétféle ciklust ismer, az elöltesztelős feltételes (while) ciklust és a bejáró (foreach) ciklust. A típusalgoritmusok tanításához kétféle ciklust használnak, az elöltesztelős feltételes (while) ciklust, és a számlálós (for) ciklust. A spagetti kód strukturált programmá „kibogozása" során célszerű volt a hátul tesztelő ciklust (do-while) is bevezetni, a belépéskor tesztelő és a tesztelés nélkül, automatikusan léptető ismétlés mellé. Kérdés, hogy a rendelkezésre álló nyelvi eszközök alkalmasake az elvárt kompetenciák elsajátítására. A tanterv alapján a ciklusok ismeretére épülnek a típusalgoritmusok, illetve a felhasználó által megadott adatok ellenőrzése is.

5.2.2.1. Léptető ciklusok

A bejáró és számlálós ciklus használata során az adatsorozat elemének elérésében van jelentős eltérés: a bejáró ciklus az adatsorozat objektum elemein iterál, míg a számlálós ciklus egy, az adatsorozattól akár független változót, a ciklusszámlálót módosítja. A típusalgoritmusokban ezt a ciklusszámlálót használjuk a tömb elemeinek elérésére mutatóként. A Python lista (illetve a range és később a set) adatsorozat objektum, amely egyes adattagjainak az elérése objektumon belül történik, ehhez igazodik a bejáró ciklus. A számlálós ciklust a range objektum helyettesíti. Azonban e kettő együtt sem biztosítja az adatsorozat elemeinek egyszerű, természetes elérését kívülről „rámutatással", hivatkozással. Ezért lesz kihívás azoknak a feladatoknak a megoldása, amelyekben kívülről több sorozatelemre kellene hivatkozni: egy részsorozat első és utolsó eleme, szomszédos elem vagy többdimenzióban a transzponált.

A tankönyv minden évben mutat olyan feladatot, ami az indexek elérésének körülményessége miatt kihívást jelent.

- 9. évfolyamon egymás után két 6-os dobás megszámlálása nehéz feladat [24 p. 126],
- 10. évfolyamon a visszafelé számlálás, utolsó jó érték keresése [26 p. 78]: „*A bejárandó indexeket előállító range() függvényt így paraméterezzük: range(len(bevételek)-1, -1, -1). A len()…ezért az első –1 … … hogy … A második –1 … azaz … A harmadik –1 … Ez a korrekt megoldás. Sok más nyelven ezt egy visszafelé haladó számlálós ciklussal valósítanánk meg – de Pythonban nincs ilyen.*".
- 11. évfolyamon a típusalgoritmusokkal kapcsolatban olvashatjuk: „*Ha nem az ott szereplő elemmel, hanem a belőle kiszámítható, meghatározható értékkel, vagy az elemek közül csak azokkal, amelyek megfelelnek valamilyen feltételnek, tehát nem mindegyikkel kell dolgoznunk, az egyszerűbb, függvényes formák mit sem érnek.*" [28 p. 160] Az egyszerűbb, függvényes formákban tisztán lehet használni a bejáró ciklust, ezért is készítették el a függvényt rá.

Vajon helyes választás a Python a típusalgoritmusok tanításához, ha az alapesetekre beépített függvény használatát ajánlja az algoritmus helyett, a módosított algoritmus kódolása viszont nehezebb, mint „sok más nyelvben"?

5.2.2.2. Feltételes ciklusok

Az elöltesztelős feltételes ciklus a Pythonban és a pszeudokódban azonos módon használható. Talán érdemes megemlíteni, hogy a feltétel mindkét esetben a ciklusba belépésre vonatkozik, míg vannak nyelvek, ahol az ismétlendő átugrására szól (például Scratch).

Hátultesztelős ciklus Pythonban nincs. Ciklus belsejében kialakuló feltételek ellenőrzésére a „biztosan belépjen" kezdőállapotot használja: „*A Python az ilyen esetekre tartogatja a nagybetűs semmit, azaz a None értéket. Ha a harmadik sort így írjuk át: kacat = None"* [24 p. 122]. Ez a megoldás filozófiájában más, mint a pszeudokódban és sok más nyelvben meglévő do-while, hátul tesztelő ciklus. A gondolkodásmód – és ebből következően a megoldási mód – nehezen általánosítható, mert feltételezi a minden más adattól különböző `None` adatot. Mivel Python nyelven nem gondoljuk egy változónév „dobozról", hogy fizikai megfelelője, azaz lefoglalt, biteket tartalmazó memóriaterülete van, ezért nem gond, hogy üres. Más nyelveken azonban problémát jelent az „éppen nem szükséges speciális érték" használata és mindenképpen más gondolkodást igényel ez a megoldás és a ciklusmag végén történő tesztelés. Emiatt a hátultesztelős ciklus működésének megértéséhez és használatának begyakorlásához nem alkalmas ez a kerülő megoldás.

A Python nyelv lehetőséget ad „belül tesztelő" ciklus szervezésre. „*A Python nyelv úgy segíti elő a break utasítás használatát, azaz a ciklusokból való idő előtti kilépést, hogy a ciklusoknak létezik else-záradékuk – ez meglehetősen ritka dolog a programozási nyelvek körében. Az else-záradék csak akkor fut le, ha a ciklus végigfutott, azaz nem léptünk ki belőle break-kel.*" A megoldás lényege egy, a szigorúan strukturált programozásban nem értelmezett utasítás: a vezérlés megszakítása. Ugyanakkor, ha az `if` és ezen belül a `break` közvetlenül a ciklus vége előtt van, akkor a működése megfeleltethető hátul tesztelő ciklus-

nak. Kérdéses, hogy így egyszerűbb-e, ahogy az is kérdéses, hogy a strukturált programozás céljának, a biztosan helyes program tervezésének megfelel-e, hogy a programozó a `break` után bármit beírhat.

Végül, a ciklus lefutását adatbevitel ellenőrzésekor kívülről is lehet tesztelni megszakítás kezelővel, a try-catch utasítás párral. Ekkor nem `break` utasítással szakad meg a ciklus, hanem kivétellel (exception), de a hatás lényegében hasonló.

Míg a strukturált programozás elől, illetve hátul tesztelő ciklusa a stabil programvezérlést segíti, a „belső" és „külső" tesztelés a programozási nyelv specialitásait használja ki. Bármelyik gondolkodásmód lehet helyes. Kérdés, hogy könnyen megtanítható-e egy speciális megoldásra alapozva a biztonságos megoldás, vagy előtérbe kellene helyezni a biztonságos megoldást, és ebből specializálni a nyelvben megtalálható megoldási lehetőségeket.

5.3. Típusalgoritmusok, programozási tételek

Az Algoritmizálás és programozási nyelv használata témakör algoritmizálás részében központi szerepe van a típusalgoritmusoknak. A típusalgoritmusok egyes feladattípusokra kidolgozott gondolkodási sémák, amelyeket programozási nyelven kódolva a számítógép elvégzi a megadott feladatot. A digitális kultúra Algoritmizálás és (formális) programozási nyelvek használata témakörének tárgyalásában az egyik legnagyobb hiba, hogy felajánlja a típusalgoritmusok (gondolkodási sémák) megtanulásának elkerülését. Minden típusalgoritmus magyarázata és a mintafeladatok megoldása is úgy kezdődik, hogy kell-e vagy nem kell a típusalgoritmust használni. Ezzel azt is sugallja, hogy a tantárgy teljesítéséhez nem szükséges az algoritmizálási készség. Hasonlattal élve: mintha a főző tanfolyamon azt tanítanák, hogy főzésnek nevezzük a konzerv vásárlását és megmelegítését: „*Persze van egyszerűbb megoldás is*" [26 p. 67-]. A Python nyelvet a könnyen tanulhatóság miatt választják programozás tanulására. Pontosabban azért, mert könnyen lehet vele programot készíteni. Ezért divatos, ez adja a motiváció alapját. Amikor a típusalgoritmusok tanulása a feladat, akkor nem a programkészítés a cél, hanem a számítógép programozásához szükséges gondolkodásmód fejlesztése. Ha nem várjuk el a tanulótól, hogy megértse és alkalmazza a típusalgoritmusokat, hanem a programozási nyelv alkalmazkodik hozzá, az olyan, mintha a csecsemővel gügyögve beszélnénk. Ennek fejlesztő hatása maximum nulla, sőt, a könnyen szerzett sikerélmény miatt a gondolkodás megerőltető lesz, ezért a fejlődés ellen hat, így negatív.

Egy példa erre a csere algoritmusa, amelyet algoritmizálás és blokk-programozás során gyakorolhatott a diák. A tankönyv az algoritmusra hivatkozás nélkül, a rendezés kapcsán a Python-os kettős értékadást mutatja [28 p. 190]:

```
lista[egyik], lista[másik] = lista[másik], lista[egyik]
```

Ez a formula azt sugallja, hogy a cseréléshez nem szükséges átmeneti tároló, lehet „levegőben, feldobva" cserélni adatokat. Fel sem merül a gondolat, hogy vagy az adatokat kell áthelyezni, vagy az adatokra történő hivatkozásokat kell megkeresni és módosítani.

5.3.1. Szemléletnek megfelelő típusalgoritmusok

A megszámlálás, másolás és kiválogatás algoritmusa könnyen megérthető, a típusalgoritmusok tanulása előtt is több példában szerepel. Ezekre a típusalgoritmusokra mondhatjuk azt, hogy úgy kell megírni az algoritmust, ahogy az ember is csinálná. A megszámlálás algoritmusa helyett használható `count()`, illetve lista hosszára a `len()`. Használatukat már egy évvel a típusalgoritmusok tanulása előtt javasolja a tankönyv [24 p. 96]. A másolás és kiválogatás algoritmusok haladó algoritmusok, mert sorozatot adnak eredményül. A természetes lépésenkénti megoldás helyett a map vagy a lista értelmezés (list comprehension) is használható. Valójában nem az algoritmus nehéz, hanem a helyette használható függvények paraméterezése. A map első paramétere egy függvénypointer, ami legfeljebb formálisan érthető, mivel az sem teljesen tisztázott, hogy az int egy objektum típusa vagy függ-

vény, ami után esetleg véletlenül lemaradtak a zárójelek [28 p. 174]. Ugyanitt, a listaértelmezés még ennél is összetettebb nyelvi varázslat, lényegében lambda kifejezésként adja meg a másolást és kigyűjtést. [28 p. 174].

5.3.2. Összegzés típusalgoritmus

Az összegzés algoritmusának függvénnyel történő helyettesítése újabb oktatásmódszertani problémát vet fel: nem igaz, hogy az ember úgy végzi el a műveletet, ahogy az algoritmus. Az ember iskolában tanulja a számok összeadását és több szám esetén egymás alá írva, helyiértékenként végzi el a műveletet, nem pedig kumulációval. Ezért azok a diákok is, akik táblázatkezelésben, adatbáziskezelésben már esetleg tanulták a `sum()` függvényt, megakadnak azon, hogy hogyan algoritmizálják a helyiértékes összeadást. Külön, megtanulandó és gyakorlandó a számítógépes összegzés algoritmusa.

5.3.3. Eldöntés, keresés, kiválasztás

Az eldöntést – és ebből következően a keresést és kiválasztást is – általában teljesen más módszerrel végzi az ember egy géphez képest. A lineáris és bináris keresés tanulás eredménye. Egy átlagos eldöntendő kérdésre az ember ránézésre válaszol. Van-e szőke ember a buszon? – nem kezdi el sorban megnézni, mert „szemmel látható" foltot vesz észre (vagy nem vesz észre). Az ember már kisgyermek kortól gyakorolja a képek összehasonlítását, a növények, tárgyak közötti különbségek felfedezését. A különbségeket, hibákat, adott részleteket szkenneléssel keresi, nem képpontonkénti összehasonlítással. Amikor listában a „van-e" kérdésre az `in` operátorral adunk választ, akkor nem gondolunk bele, hogy a gép másképp működik, mint az emberi agy [26 p. 59]:

```
if vizsgált in lista: print('Benne van.') else: print('Nincs benne.')
```

Az 'in' lebontása algoritmusra a legnehezebb a típusalgoritmusok tanulása során, mert a vizsgált adatokra „ránézésre" megállapítható az eredmény. Ráadásul általános, hogy az adott programozási nyelven vannak olyan logikai vizsgálatok, döntések, amelyek alacsonyabb szintű implementációban már eldöntés tétellel oldhatók csak meg. Például két szám összehasonlítása processzor szintjén az első eltérő bit megkeresést jelenti, az egyenlőség annak eldöntése, hogy minden bitjükben megegyeznek-e.

Azzal, hogy Pythonban az adatsorozat lista-objektum, erősítjük a sorozat vizsgálata helyett az objektum tulajdonság megközelítést, amihez a kívülről meghívott `in` operátort használjuk. Egy tömb sokkal nyitottabb, az egyes elemeit közvetlenül elérjük (nincs elrejtve a belsőjébe a szerkezete). Ezért sokkal jellemzőbb, hogy lineáris kereséssel adunk választ a kérdésre.

A kétféle megközelítés – mi a döntés és mi az, amihez az eldöntés típusalgoritmust kell alkalmazni – az elmúlt 3 évben az ELTE-s programozás csoportjaimban a hallgatók több, mint felének okozott problémát. Például a feltételes maximum-kiválasztást eldöntés tétel és maximum-kiválasztás tétel kombinációjának specifikálták.

A keresés és kiválasztás típusalgoritmusok operátorral és függvénnyel történő helyettesítése egy harmadik szempont szerint is hibás, ez pedig a hatékonyság. A tankönyvi példák közül tipikusan az indexre rákérdező – melyik elem rendelkezik az adott tulajdonsággal – feladatok megoldása rossz hatékonyságú. Például [26 p. 71]:

„*Természetesen ezúttal is létezik egyszerűbb megoldás – és ez is a két előző „egyszerűbb megoldás" összeépítése.*"

```
if 5 in bevételek:
print('Az ötpiculás fuvar sorszáma:', bevételek.index(5) + 1)
else: print('Nincs ötpiculás fuvar.')
```

A megoldásban az `in` operátor egy eldöntés típusalgoritmust rejt el, majd az `index()` a kiválasztás típusalgoritmusát használja. A kód futtatási ideje általában a vizsgálandó adatok számától négyzetesen függ. Ez az a kombináció, ami miatt például a már ismert INDEX(…HOL.VAN())

függvénykombináció tömeges használata lelassítja a táblázatkezelő újraszámítási eljárását. Így az egyszerűbb megoldás nem csak gondolkodásban jelent eltunyulást, lustaságot, hanem a program futását is lassítja.

Összefoglalva: az `in` operátor és `index()` függvény használata az eldöntés, keresés és kiválasztás típusalgoritmusok helyett

- nem teszi lehetővé az adatsorozat elemeinek közvetlen elérését;
- nem fejleszti az algoritmikus gondolkodást;
- nem teszi lehetővé a program idő (és ezzel erőforrás) hatékony felhasználását.

Emiatt ezeknek az algoritmusoknak a függvényekkel történő helyettesítése kiemelten káros.

5.3.4. Szélsőérték-kiválasztás

Az eldöntéshez hasonló problémákkal találkozhatunk a szélsőérték kiválasztása során is. Amikor egy alkalmazásban a `max()` függvényt használjuk, nem gondolunk bele a függvény működésének algoritmusába. Amikor mi magunk keressük a maximumot, akkor viszont nem hasonlítgatjuk össze egyenként az adathalmaz elemeit. Egy hatalmas tömegből úgy választunk maximumot, hogy kiválasztunk néhány, a környezeténél magasabb embert és ezeket hasonlítjuk össze, jó eséllyel nem sorban, hanem páronkénti kiejtéssel. A maximum kiválasztás algoritmusának megírása ezért más, nem a természetes gondolkodásmódot igényli.

Ennél a típusalgoritmusnál is nehézséget okoz, hogy a lista-objektum adatai nehezen érhetők el közvetlenül: „*Ötlet: vezessünk be egy változót, aminek az értéke a feladatban elképzelhető legkisebb eredmény…*” a következő feladatban: „*…a farkas legkisebb libáját tároló változó kezdőértékéül a 100-at választottuk, bátor döntés…*” A típusalgoritmust követően: „*Kihívást jelentő feladat: Módosítsuk úgy az előző programunkat, hogy biztosan helyesen adja meg a farkasnak jutó legkisebb liba tömegét!*” [26 p. 73-74].

A típusalgoritmus végiggondolásába Python nyelven nem fér bele, hogy az első adat legyen a kezdőérték. A további gyakorlófeladatok során a kezdőérték jellemzően 0. Ha nem az értéket kell visszaadni, akkor az erre használt változó `None` értékű.

A maximum-kiválasztás algoritmusa összetettebb feladatok esetén kifejtve is megtalálható, de mindig van mód az „egyszerűbb, függvényes” megoldásra. Például [28 p. 162]:

```
legtöbb = max(heti_hiányzások)
print('A(z) ', heti_hiányzások.index(legtöbb)+1, '. héten volt
          a legtöbb hiányzás.', sep='')
```

A megoldás itt nem a kezdőérték választása miatt pongyola, hanem azért, mert a függvényekkel nem egy, hanem két kiválasztási algoritmus szükséges, ami miatt nő a futási idő. A feladat függvényes megoldása táblázatkezelőben az INDEX(…HOL.VAN(MAX(),…)) függvénykompozíció, ott ez megfelelő megoldás, de tudni kell, hogy az újraszámítások miatti tömeges használata lelassíthatja a gép működését. Ezért érdemes megtanulni egy hatékonyabb megoldást, ami nem a függvények összetett alkalmazása, hanem közvetlen, hatékonyabb megoldást ad.

Az eldöntés típusalgoritmusának alkalmazása bonyolult keresési feltétel esetén nehezen kerülhető el, ezzel szemben a szélsőérték-kiválasztásának típusalgoritmusa – esetleg előzetes kiválogatással kombinálva – minden esetben helyettesíthető függvénnyel, ezzel a szélsőérték-kiválasztás algoritmusát teljesen elfedi.

A táblázatkezelés során használt függvények programozási nyelvben történő alkalmazása nem fejleszti a gondolkodási képességet, legfeljebb ismétli a már tanultakat. Mivel nem visz tovább annál, mint amit táblázatkezelőben 4 kattintással megkapunk, nem ad értelmet a programozási nyelven való megvalósításnak. Emiatt nem csak a gondolkodás fejlesztése marad el, hanem a motiváció is elkopik.

5.3.5. Összetett algoritmusok és függvényeik

Az összetett algoritmusok az egyszerű típusalgoritmusra épülnek. Kivéve, ha függvényeket alkalmazunk az egyszerű típusalgoritmus helyett, mert akkor az összeépítés nehézséget fog okozni. Ezért az összetett algoritmusok megértése és felépítése helyett a tankönyvben előtérbe kerülnek az ezeket megvalósító függvények. Ebből adódóan a speciális igények megvalósítása nem a tanultak módosított alkalmazása (a tudás fejlesztése), sokkal inkább internetes keresés és formális behelyettesítés.

„*Megjegyezzük, hogy a Python sort() és sorted() függvénye egyébiránt remek dolgokat tud, például a kedvencek kétdimenziós listát a kis listák utolsó eleme – a kor – alapján akár fordított irányba is képes rendezni. A szintaxis nem egyszerű, de az érdeklődők számára megmutatjuk:* [28 p. 192]

```
kedvencek.sort(key=lambda x: x[-1], reverse=True)
```

Vajon mit gondol az érdeklődő diák a „key=lambda…" kifejezésről? Volt diákom elbeszélése alapján: „sokat tanultam, de nem jutottam semeddig". Egy, adatbáziskezelésben néhány kattintással vagy egyszerű SQL utasítással megoldható problémához minden eddigi ismeret Python nyelven kevés. Ha egy rendezés megfordításához ennyi plusz ismeret kell, akkor mi kell egy játékprogram elkészítéséhez?

6. Összegzés

A Python világszerte elismert, az informatika több területén jól használható programozási nyelv. Objektumorientált, a memóriát dinamikusan kezeli. A gyakori feladatokra kész, a humán gondolkodáshoz alkalmazkodó megoldásokat tartalmaz, ezért Python nyelvet használva a programozásban járatlanoknak is sikerélményt ad az első programok elkészítése.

A Python első programozási nyelvként azoknak ajánlott, akiknek a gépelés nem jelent problémát, emiatt a felnőttképzésben, az egyetemi tanulmányok kezdetén jellemző a Python nyelvű programozásoktatás.

A közoktatásban 2020-ban bevezetett tanterv szerint a programozásoktatás az általános iskola alsó tagozatán kezdődik. Ekkor a Python még nem használható, ugyanakkor a különböző blokk alapú programozási környezetekben a programozás – kód begépelésén kívüli – elemei játékos formában tanulhatók. Egyes blokk programozási környezetekben kódnézet is rendelkezésre áll, ez alkalmat ad például a Python nyelvre való átállásra, a Python nyelvvel való ismerkedésre. Erre a nyelvváltásra még nincsenek gyakorlati tapasztalatok.

Hosszú távon a középiskolai programozásoktatás építhet az általános iskolás években tanult elméleti és blokk-programozás során tanult ismeretekre, ezért a középiskolás programozásból elvileg nem kezdő. Azonban jelenleg, a tanterv több ponton történt bevezetése miatt, vannak olyan évfolyamok, amelyeknél még nem számíthatunk az általános iskolában tanult programozási ismeretekre, ezért reális lehet Python nyelven tanítani a középiskolás tananyagot.

Összehasonlítva a tanfolyami, egyetemi előkészítő és közoktatási tantervet, azt tapasztaljuk, hogy a közoktatásban az általános iskola végére elérendő cél – a Python nyelv használatától eltekintve – megegyezik az ICDL és más, felnőtteknek szánt bevezető képzések céljával.

A középiskolai programozás tananyag ezen túllép: az adat absztrakció a virtuálisan tapasztalttól fizikai megvalósításhoz közelít, az adatok kezelésére a strukturált programozás elveit figyelembe vevő tervezés és kódolás jellemző. Ezzel betekintést nyújt a programozás (program tervezés, adatstruktúra tervezés, objektum tervezés) szakmákba. A tananyag érzékelhetően túllép a belépő szinten, a programkészítés felszíne alá nézve elméleti és gyakorlati ismeretek elmélyítésére és rendszerezésére törekszik.

A középiskolai Digitális kultúra tantárgy Algoritmizálás és (formális) programozási nyelv használata témakör 9–11. évfolyamos fejezetei a tananyagot Python nyelv használatával dolgozza fel. Pontosabban, a Python nyelvet és programozói gondolkodásmódot tanítja a kitűzött tananyag mentén. Talán a terjedelmi korlátoknak köszönhető, de jól látható, hogy a nyelv nem a tantervi cél megvalósításához használt eszköz, hanem fordítva, a cél a Python nyelv és gondolkodásmód használata, amihez a tantervhez kapcsolódó feladattípusok szolgálnak eszközül. A cél-eszköz kapcsolat módosításának indoka, hogy a Python nyelv használata motiválja a diákokat a programozásra.

A tananyag elemzése során azt vizsgáltuk, hogy a Python nyelv oktatása során megtaníthatók-e a tantervben előírt fogalmak, ismeretek, kompetenciák, illetve, ha megtaníthatók, akkor mi a motiváló tényező. Másképp: mennyire jó, mennyire használható a Python nyelv a tananyag elsajátítására. Az elemzés egyértelműen kimutatta, hogy a tantervi követelményekhez a Python nyelv nem tartalmazza az eszközöket. Ennek oka leginkább az, hogy a Python nyelvben a memóriában tárolt adathoz csak pointeren keresztül, azaz indirekt módon lehet hozzáférni:

- Python nyelven nincs karakter típus.
- Az elemi adattípusokat csak műveleteik jellemzik, nincs méretük, memóriában lefoglalt helyük.
- A változónév dinamikusan van az adathoz társítva, így az adat a változó deklarálásával nem absztrahálható.
- Nincs tömb (vektor) adatstruktúra. Mivel az adat sem érhető el közvetlenül, az adatsorozatra (intervallumra) optimalizált típusalgoritmusok adattípusa hiányzik. Ebből következően nincs egyszerű kétdimenziós „táblázat" adatszerkezet se, amiben az adatot sor és oszlop indexen keresztül lehet elérni.
- Nincs a tantervben elvárt típusalgoritmusokhoz számlálós ciklus a strukturált programozás ciklus-típusaiból csak az elöl tesztelő ciklus található meg az eredetivel azonos működéssel.

A tantervi tananyag elsajátításának további akadálya, hogy a Python nyelvet nem arra optimalizálták, hogy a programozó részletesen leírja, hogyan működjön a program, hanem arra, hogy a kész modulokból válasszon. Ezért a tantervben kitűzött kompetenciafejlesztés helyett a Python nyelv használata kerül előtérbe.

- Nincs rekord, struktúra. Helyette a szótár típust használja, aminek a tantárgy másik, adatbáziskezelés témakörével való kapcsolatát ezen a tudásszinten nem lehet bemutatni.
- Rekord implementációra másik megoldás lehetne az osztály adatstruktúra, de – úgy tűnik, – ez a nyelv fejlesztése, ezért szakember feladata, nem nyelvhasználónak való. Ebből következően haladó szintű ismeret a mező szelekció (más nyelvekben a pont operátor). A misztifikálás miatt nem tud kapcsolódni a tananyag többi témakörében és a programozás témakörön belül is gyakran emlegetett objektumokhoz. (Például: betű, kép, táblázat, ellipszis, pixel.)
- A lista mindenre használható, strukturálatlan rekordnak is. Emiatt a különböző adatstruktúrák absztrakciójára nem használható.
- A Python programozói szemlélet a típusalgoritmusok használata elé helyezi a tipikus statisztikai feladatokra kész függvényeket. A tankönyvben ezért a függvények is típusalgoritmusként jelennek meg. Ezzel megkerüli a tantervi előírások kompetenciafejlesztési céljait. A típusalgoritmusok helyett használt függvények zavart okoznak az absztrakciós készség fejlesztésében, mivel a „könnyű" feladatok algoritmizálása helyett függvényt alkalmazva a „nehéz" feladatok algoritmusának elkészítéséhez nincs alapozás.
- Az eldöntés, keresés, kiválasztás típusalgoritmusának operátorral helyettesítése zavart okoz az elvárt absztrakciós szint meghatározásában. Az objektumban „in" operátorral kapott eredmény tulajdonság, de az objektumba belépve sorozaton történő keresés. A tantervben leírt cél a magasabb absztrakciós szint (elemibb megvalósítás), aminek az a megoldás nem tesz eleget.

- Az összeg, szélsőérték-kiválasztás, eldöntés, keresés, kiválasztás – az egyszerű típusalgoritmusok 2/3-a –, ha látó ember végzi, akkor a szem 2D képalkotásból következően nem sorfeldolgozással, hanem szkenneléssel, folt érzékeléssel, memóriában tárolt kép rávetítéssel történik. Ezért a típusalgoritmus nem a természetes végrehajtás számítógépes megvalósítása, hanem egy számítógépre optimalizált, ezért absztrakciót igénylő, megtanulandó módszer. A függvény elfedi a feldolgozásbéli különbséget.

Amikor a tantervben kitűzött célok eléréséhez eszközöket rendelünk, alapfeltétel, hogy az eszköz rendelkezzen a célok eléréséhez szükséges funkciókkal. Amennyiben a konkrét eszköz valami miatt nem elérhető (például életveszélyes lenne), akkor a céloknak megfelelő szimulációval helyettesíthető a funkció. A fenti lista azt mutatja, hogy a Python nyelvből hiányoznak a tantervben megnevezett adattípusok és vezérlési struktúrák, a helyettesítésre használt eszközök más szemléletmódú megoldásokat nyújtanak.

A Python nyelv alkalmas a programozás iránti érdeklődés felkeltésére, de dinamikus adatkezelése elrejti az adatot. Az adatabsztrakció és a típusalgoritmusokat helyettesítő függvények tekintetében a táblázatkezelőkhöz hasonló.

Az általános iskolás korosztály programozásoktatásához a Python – több blokknyelvekkel együtt – megfelelő eszköz, de az erre épülő középiskolai (gimnáziumi) képzésben a kitűzött célok elérését gátolja, hogy a nyelv jellegéből adódóan a gyakorlat az elmélettel nincs összhangban.

Bibliográfia

1. Edsger W. DIJKSTRA: *A Case against the GO TO statement* (letter EDW 215) *Go-to statement considered harmful.* In Commun, ACM (11) (1968); 3, p. 147–148. Online archívum: https://www.cs.utexas.edu/users/EWD/ewd02xx/EWD215.PDF [Megtekintés: 2022.10.31.]
2. Edsger W. DIJKSTRA: *Notes on Structured Programming* In: E.W.D Archive the manuscript of Edsger W. Dijkstra, EWD249 https://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF [Megtekintés 2022.10.31.]
3. Edsger W. DIJKSTRA: *What led to "Notes on Structured Programming"*. Nuenen, 2001.06.10 In: E.W.D Archive the manuscript of Edsger W. Dijkstra, EWD1308 https://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1308.PDF [Megtekintés 2022.10.31.]
4. Vuorikari, R., Kluzer, S. and Punie, Y.: *DigComp 2.2: The Digital Competence Framework for Citizens - With new examples of knowledge, skills and attitudes*. EUR 31006 EN, Publications Office of the European Union, Luxembourg, 2022, ISBN 978-92-76-48882-8, doi:10.2760/115376, JRC128415. https://publications.jrc.ec.europa.eu/repository/bitstream/JRC128415/JRC128415_01.pdf [Megtekintés 2022.11.04.]
5. Brodnik et all: Programming for All: Understanding the Nature of Programs. (2021) https://arxiv.org/pdf/2111.04887.pdf [Megtekintés 2022.11.04.]
6. *ECDL Kódolási ismeretek (Python). Syllabus 1.0.* NJSZT, Budapest (2021) https://njszt.hu/hu/ecdl/modul/kodolasi-alapismeretek-python [Megtekintés 2022.10.31.]
7. *Why is Python So Polular?* Pulumi (2022) https://www.pulumi.com/why-is-python-so-popular/ [Megtekintés: 2022.11.06.]
8. David Beazley: *Foreword* In: P. Wentworth, J. Elkner, A.B. Downey, C. Meyers: *How to Think Like a Computer Scientist: Learning with Python 3 (RLE).* (2002) https://openbookproject.net/thinkcs/python/english3e/foreword.html [Megtekintés: 2022.11.06.]
9. *Robotika, algoritmizálás, programozás.* In: Digitális kultúra tankönyv 5., p. 7–34, Oktatási Hivatal (2020) ISBN: 978-615-81539-4-2. https://www.tankonyvkatalogus.hu/pdf/OH-DIG05TA__teljes.pdf [Megtekintés: 2022.11.01.]
10. *Robotika, algoritmizálás, programozás.* In: Digitális kultúra 5 (NAT 2020) okostankönyv. Oktatási Hivatal (2020)

https://www.nkp.hu/tankonyv/digitalis_kultura_5_nat2020/fejezet_01_fejezetnyito [Megtekintés: 2022.11.01.]

11. *Digitális kultúra 5–8. évfolyam.* In: Kerettanterv az általános iskolák 5–8. évfolyama számára, Oktatási Hivatal, (2020) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_F.docx [Megtekintés: 2022.11.02.]
12. *Algoritmizálás, programozás, robotika.* In: Digitális kultúra tankönyv 6., p. 35–60, Oktatási Hivatal (2020) ISBN: 978-615-6256-38-6. https://www.tankonyvkatalogus.hu/pdf/OH-DIG06TA__teljes.pdf [Megtekintés: 2022.11.01.]
13. *Algoritmizálás, programozás, robotika.* In: Digitális kultúra tankönyv 7. p. 39–60, Oktatási Hivatal (2022) ISBN: 978-663-436-290-6. https://www.tankonyvkatalogus.hu/pdf/OH-DIG05TA__teljes.pdf [Megtekintés: 2022.11.01.]
14. *Algoritmizálás, programozás, robotika.* In: Digitális kultúra 6 (NAT 2020) okostankönyv. Oktatási Hivatal (2021) https://www.nkp.hu/tankonyv/digitalis_kultura_6_nat2020/fejezet_03_fejezetnyito [Megtekintés: 2022.11.01.]
15. *Algoritmizálás, programozás, robotika.* In: Digitális kultúra 7 (NAT 2020) okostankönyv. Oktatási Hivatal (2022) https://www.nkp.hu/tankonyv/digitalis-kultura-7-nat2020/fejezet_04_fejezetnyito [Megtekintés: 2022.11.01.]
16. *Algoritmizálás és blokkprogramozás. Robotika.* In: Digitális kultúra 5–8. okosgyűjtemény. Oktatási Hivatal (2022) https://www.nkp.hu/tankonyv/digitalis_kultura_okosgyujtemeny_5-8/fejezet_01_fejezetnyito [Megtekintés: 2022.11.01.]
17. *Digitális kultúra 9–11. évfolyam.* In: Kerettanterv a gimnáziumok 9–12. évfolyama számára, Oktatási Hivatal, (2020) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_K.docx [Megtekintés: 2022.11.04.]
18. *Bevezetés.* In: Kerettanterv a szakgimnáziumok 9–12. évfolyama számára. Oktatási Hivatal (2020) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/szakkepzes/Bevezetes.docx [Megtekintés: 2022.11.04.]
19. *Digitális kultúra*. In: Közismereti kerettanterv a szakképző iskolák számára. Oktatási Hivatal (2020) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/digitalis.doc [Megtekintés: 2022.11.04.]
20. *A Hátrányos Helyzetű Tanulók Arany János Tehetséggondozó Programja* http://www.ajtp.hu/kozerdeku?csu=AAAVIXVC [Megtekintés: 2022.11.04.]
21. *Digitális kultúra* In: Arany János Kollégiumi Program 9/AJKP évfolyam Oktatási Hivatal (2021) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/AJKP/9_digitkult_ajkp_uj_VD.doc [Megtekintés: 2022.11.04.]
22. *Digitális kultúra (3 óra/hét) 9. évfolyam.* In: Arany János Tehetséggondozó Program kerettanterve, Oktatási hivatal (2021) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/AJTP/Digitalis_kultura_9AJTP_evfolyam_3_ora_2021.docx [Megtekintés: 2022.11.04.]
23. *Digitális kultúra (4 óra/hét) 9. évfolyam.* In: Arany János Tehetséggondozó Program kerettanterve, Oktatási Hivatal (2021) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/AJTP/Digitalis_kultura_9AJTP_evfolyam_4_ora_2021.docx [Megtekintés: 2022.11.04]
24. *Algoritmizálás és programozási nyelv használata.* In: Digitális kultúra 9. tankönyv, p. 93–128, Oktatási Hivatal (2020) https://www.tankonyvkatalogus.hu/pdf/OH-DIG09TA__teljes.pdf [Megtekintés: 2022.11.04.]

25. *Algoritmizálás és programozási nyelv használata.* In: Digitális kultúra 9. (NAT 2020) okostankönyv. Oktatási Hivatal (2021) https://www.nkp.hu/tankonyv/digitalis_kultura_9_nat2020/fejezet_05_fejezetnyito [Megtekintés: 2022.11.04.]
26. *Algoritmizálás és programozási nyelv használata.* In: Digitális kultúra 10. tankönyv, p. 41–90, Oktatási Hivatal (2020) https://www.tankonyvkatalogus.hu/pdf/OH-DIG10TA__teljes.pdf [Megtekintés: 2022.11.04]
27. *Algoritmizálás és programozási nyelv használata.* In: Digitális kultúra 10 (NAT 2020) okostankönyv. Oktatási Hivatal (2021) https://www.nkp.hu/tankonyv/digitalis_kultura_10_nat2020/fejezet_05_fejezetnyito [Megtekintés: 2022.11.04.]
28. *Algoritmizálás és formális programozási nyelv használata.* In: Digitális kultúra 11. tankönyv, p. 151–234, Oktatási Hivatal (2020) https://www.tankonyvkatalogus.hu/pdf/OH-DIG11TA__teljes.pdf [Megtekintés: 2022.11.04.]
29. *Algoritmizálás, formális programozási nyelv használata.* In: Digitális kultúra 11 (NAT 2020) okostankönyv. Oktatási Hivatal (2022) https://www.nkp.hu/tankonyv/digitalis-kultura-11-nat2020/fejezet_07_fejezetnyito [Megtekintés: 2022.11.04.]
30. *Algoritmizálás, programozás (Általános irány).* In: Digitális kultúra okosgyűjtemény 9-12. Oktatási Hivatal (2022) https://www.nkp.hu/tankonyv/digitalis-kultura-okosgyujtemeny-9-12/fejezet_01_fejezetnyito [Megtekintés: 2022.11.06.]
31. *Algoritmizálás, programozás (Tehetséggondozó irány)* In: Digitális kultúra okosgyűjtemény 9-12. Oktatási Hivatal (2022) https://www.nkp.hu/tankonyv/digitalis-kultura-okosgyujtemeny-9-12/fejezet_02_fejezetnyito [Megtekintés: 2022.11.06.]
32. Tim Peters: *The Zen of Python.* https://peps.python.org/pep-0020/ (2022.03.15.) [Megtekintve: 2022.11.08.]

Adatvédelem és információbiztonság oktatási kérdései a 2020-as NAT tükrében

Törley Gábor[1], Holló Csaba[2]

[1]gabor.torley@inf.elte.hu; ORCID: 0000-0002-0496-936
ELTE Eötvös Loránd Tudományegyetem Informatikai Kar
[2]chollo@inf.u-szeged.hu; ORCID: 0000-0003-0077-3153
SZTE TTIK Informatikai Intézet

Absztrakt. Nagy nemzetközi vizsgálatok alapján következtethetünk arra, hogy a 9-16 korosztályt egyre jobban érintik a digitális világ veszélyei. A legutóbbi PISA felmérés arra mutat rá, hogy a magyar diákok az OECD átlag alatt teljesítettek olvasásból/szövegértésből, amelyet az információbiztonság fontos faktorának tartunk, mert az információ helytelen értelmezése kockázatokat rejt, pl. adathalászat, álhírek, keresési eredmények értékelése stb. Cikkünkben a magyar tankönyvek adatvédelemmel és információbiztonsággal kapcsolatos becsült óraszámait vetjük össze a Digitális kultúra tantárgy más témaköreinek, egy konkrét angol tanmenet, valamint a korábbi 2012-es magyar kerettanterv óraszámaival és tartalmaival. Ezek alapján ajánlást teszünk a még nem kiadott 4. és 8. osztályos Digitális kultúra tankönyvek adatvédelemmel és információbiztonsággal kapcsolatos részeihez.

Kulcsszavak: adatvédelem, információbiztonság, tantervek.

1. Bevezetés

Az EU Kids Online 2020-ban publikált kérdőíves kutatása [1] 9-16 éves gyerekeket vizsgált. A felmérés eredménye szerint a 15-16 éves gyerekek szinte naponta használják az okostelefonjaikat, és ezt az eszközt preferálják a világhálón való szörfözésre. A fiatalabb, 9-11 éves korosztály fele annyit tölt a világhálón, mint az előbb említett idősebb társaik. A legtöbb vizsgált országban a gyerekek több, mint fele heti rendszerességgel látogatja a közösségi oldalakat, érdekes módon Franciaország, Málta és Németország kivétel ez alól, ott ritka, hogy a gyerekek rendszeresen látogatnák a közösségi oldalakat. A korhatárbesorolások ellenére nem elhanyagolható számú 9-11 éves gyermek fér hozzá a közösségi oldalakhoz nap, mint nap. A 12-16 éves gyermekek magas pontszámot értek el a böngészőhasználat és a közösségi hálózatok használata terén. Több országban alacsony szintet mértek az információkezelés terén, amely magában foglalja az információ kritikus feldolgozását. A világháló veszélyei közül leginkább a sexting és az új emberekkel való találkozás került megnevezésre. Az előbb említett veszélyek ellenére tudatosan és helyesen tudtak reagálni az olyan helyzetekre, amikor számukra bántó viselkedéssel találkoztak. Az évek múlásával egyre több adatvédelmi problémával találkoznak, pl. pénzfizetés online alkalmazáson, játékon keresztül. A gyermekek több, mint 80%-a a szüleitől, barátaitól és a tanáraitól kap tanácsolat az internet biztonságos használata terén.

A 2018-as PISA vizsgálatok eredménye [2] hasonló dolgokat mutatott: a magyar diákok az OECD átlag alatt teljesítettek olvasásból/szövegértésből. A magyar tanulók több, mint negyede a 2. szint alatt teljesített. Ezen a szinten a diákok meg tudják határozni a fő gondolatát egy mérsékelten hosszú szövegnek. Az értékelés és a reflektálás mindig is a szövegértés műveltségterület része volt. A digitális szövegértés korszakában az olvasók egyre több és több információval találkoznak, és képesnek kell lenniük arra, hogy el tudják dönteni, hogy mi az, ami megbízható és mi az, ami nem. Ez ugyancsak része az informatikai biztonságtudatosságnak. Ugyanis az alacsony szövegértéssel rendelkező tanulók tudatossága is alacsonyabb lesz.

2. Irodalmi áttekintés

A 2012-es NAT-hoz illeszkedő, informatika kerettanterv alapján négy[1] szervezet helyi informatika tanterv ajánlását vizsgáltuk meg korábbi munkánkban [3], hogy az óraszám hány százalékát szánja az információbiztonsággal, adatvédelemmel kapcsolatos témakörökre. A számszerű adatokat az 1. táblázat mutatja.

Név	Óraszám	Informatika óraszám (összes)	Százalékos arány
Mozaik	12	252	5%
KPSZT	9	180	5%
JOS	16	252	6%
NTK	10	180	6%

1. táblázat: Mintatantervek óraszámai

Az adatokból látszik, hogy a KPSZT-n és az NTK-n kívül mindenki felhasznált órákat a szabadon tervezhető órakeretből[2]. Elmondható tehát, hogy 2012-es tantervi javaslatok alapján átlagosan az informatika órák 5-6%-án tanulnak információbiztonságról és adatvédelemről az 5-12. évfolyamokban.

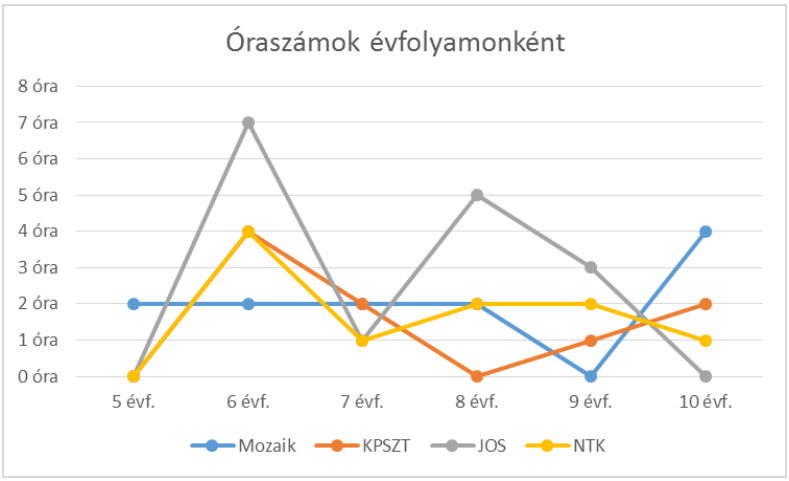
Az 1. ábrán látható diagram mutatja, hogy a kiadók többsége az 5-12. évfolyamok elején súlyozza jobban az adatvédelemmel és adatbiztonsággal kapcsolatos órákat, ez alól kivételt képez a Mozaik Kiadó.

Ugyanebben a munkánkban 22 angliai középiskolai tanárt kérdeztünk arról, hogy mely évfolyamon hány tanórát és milyen témákat tanítottak adatvédelem és adatbiztonság témakörből. A visszaküldött kérdőívek eredménye szerint ezekre a tanárok átlagosan 13,36 tanórát szántak összesen a fenti két téma oktatására a Key Stage[3] 3 és 4 fejlődési fokozatban, amely megfelel a magyar iskolák 5-10. évfolyamának. Az óraszámok eloszlásából következik, hogy a tanárok fele 15 vagy annál több órát használ fel adatvédelem és adatbiztonság tanítására a Key Stage 3 és 4 évei alatt, és az is elmondható, hogy az órák többsége a Key Stage 3 első két évére összpontosul.

[1] Mozaik Kiadó (Mozaik), Katolikus Pedagógiai Szervezési és Továbbképzési Intézet (KPSZT), Nemze-ti Tan-könyvkiadó (NTK), Jeldik Oktatási Stúdió (JOS)

[2] A 2012-es szabályzás szerint minimum 180 órát kellett szánni az informatika tantárgyra. Ahol ennél többet szántak rá, azt a szabadon felhasználható órakeretből tehették meg.

[3] magyarul: fejlődési fokozat



1. ábra. A kiadók ajánlott tantervi óraszámainak évfolyamonkénti megoszlása, NAT 2012.

A Eurodyce 2022-es jelentése ([4]) megerősíti, hogy a biztonságtudatosság a digitális alapműveltség része, valamint leírja, hogy a vizsgált 30 európai országban nagyon hasonlóak az elvárások ezen a téren. Magyarország azon országok között van, ahol az adatvédelem és információbiztonság témája a közoktatás minden szintjén megjelenik, ráadásul kötelező és külön tantárgyban.

3. Az aktuális Digitális kultúra tankönyvek adatvédelemmel és információbiztonsággal kapcsolatos tartalmai

Megvizsgáltuk, hogy a 2020-as NAT és kerettantervek alapján készült Digitális kultúra tankönyvek ([5,6,7,8,9,10,11]) milyen adatvédelemmel és információbiztonsággal kapcsolatos tartalmakat tárgyalnak, milyen mélységben, és mennyi időt javasolnak ezek tanítására. A ráfordítandó idők tekintetében természetesen csak becsléssel élhetünk, hiszen lehetnek kisebb eltérések az egyes tanárok által ténylegesen felhasznált idők között. Mivel az Oktatási Hivatal által közzétett mintatanmenetek [12] sok esetben általánosabb, és helyenként a tankönyvektől teljesen eltérő leírásokat tartalmaznak, ezért vizsgálatunkban a tankönyvekre támaszkodtunk. Viszont, a 4. és 8. osztályos tankönyvek még nem állnak rendelkezésre, ezért a becsült óraszámokban ezek jövőbeli tartalma nem szerepel.

Az adatvédelemmel és információbiztonsággal kapcsolatos tartalmakat három fő kategóriába soroltuk: a megszerzett információ megbízhatósága, a rendelkezésre álló adatokra vonatkozóan az adatvesztés és jogosulatlan hozzáférés megakadályozása, illetve az információk szándékos megosztásának lehetőségei és veszélyei.

3.1. A megszerzett információ megbízhatósága

Ebben a kategóriában olyan ismeretekkel foglalkozunk, melyek azt célozzák, hogy a felhasználó az általa talált tartalmak relevanciájáról, valódiságáról és igazságtartalmáról meg akarjon és meg is tudjon győződni. A kategóriában tárgyalt ismereteket (évfolyamonként az azokat magukba foglaló tankönyvbeli témakörökkel együtt), és az azokra fordítható körülbelüli időkeretet (tanórában számolva) az alábbi táblázatban összegezzük (a 4. és 8. osztályos tankönyvek tartalma jelenleg ismeretlen, ezért az azokhoz tartozó sorokba, és a hozzájuk tartozó óraszámokhoz kérdőjeleket tettünk):

Évfolyam	Témakör és tartalom	Tanóra
3.	**Információszerzés az e-világban** Mi az álhír? Milyen ellenőrzéseket lehet végezni az információ megbízhatóságával kapcsolatban?	1
3.	**Védekezés a digitális világ veszélyei ellen** Ellenőrzések végzése a beszélgetőtárs személyének kilétével kapcsolatban.	0,1
4.	?	?
5.	**e-Világ és online kommunikáció** Álprofilok, álhírek, és szempontok az információ megbízhatóságának ellenőrzésére. Felhasználók által létrehozott oldalak, Wikipédia.	0,3
6.	**Az e-világ és az online kommunikáció** Álhírek, megbízhatóság. Webáruházak és a vásárlással kapcsolatos információk megbízhatósági jellemzői.	0,2
7.	**Az e-világ és az online kommunikáció** Adathalászat céljából készített oldalak felismerése.	0,05
8.	?	?
9.	**Online kommunikáció** A hitelesség szempontjai.	0,05
10.	**Az információs társadalom, e-Világ** Álprofilok, álhírek, hoax, szűrőbuborék (véleménybuborék).	0,8
11.	**Online kommunikáció** Talált információk megbízhatóságának szempontjai.	0,4
11.	**Az információs társadalom, e-Világ** Deepfake.	0,05
	Összesen	**2,95 + ?**

2. táblázat: A megszerzett információ megbízhatóságának megbeszélésére fordítható idő becslése tanórában kifejezve

3.2. Az adatvesztés és jogosulatlan hozzáférés megakadályozása

Ebben a kategóriában az adatvesztés és jogosulatlan hozzáférés lehetőségeivel és azok megelőzésével foglalkozunk. Egy másik kategóriaként vizsgáljuk majd azokat az eseteket, amikor a tulajdonos az adatait szándékosan teszi közzé. Ebbe a kategóriába tartozó ismereteket (évfolyamonként az azokat magukba foglaló tankönyvbeli témakörökkel együtt), és az azokra fordítható körülbelüli időkereteket (tanórában számolva) az alábbi táblázatban összegezzük:

Évfolyam	Témakör	Tanóra
3.	**A digitális eszközök használata** Vírusveszély.	0,1

Évfolyam	Témakör	Tanóra
3.	**Védekezés a digitális világ veszélyei ellen** Személyes, illetve nem személyes adatok. Adathalászat.	1
4.	?	?
5.	**e-Világ és online kommunikáció** Adathalász jellegű e-mailek, illetve vírust, kémprogramot tartalmazó mellékleteik. Érzékeny, személyes adatok. Adatgyűjtési és -felhasználási engedélyek megadásának mérlegelése. Adatvédelmi beállítások használata. Jelszó készítése. Felhőszolgáltatásokban tárolt adatok jelszavas védelme, megosztás során beállítható jogosultságok.	0,8
6.	**Az e-világ és az online kommunikáció** Elektronikus ügyintézéshez kapcsolódó adatvédelmi, adatbiztonsági kérdések: belépési azonosító, profil óvása, jelszavak erőssége, cseréje, különböző oldalakon különböző jelszavak használata, illetve adatok sérülésének, megsemmisülésének megelőzése.	0,2
6.	**Digitális eszközök használata** A fényképezés etikája: hozzájárulás kérése a fénykép készítéséhez.	0,1
7.	**Az e-világ és az online kommunikáció** Hozzáférési, azonosítási adatok védelme, többfaktoros azonosítás, virtuális bankkártya használata, adathalászat. Az online kommunikációs szolgáltatást üzemeltető cégek mesterséges intelligenciája „belehallgathat" a beszélgetésekbe.	1
8.	?	?
9.	**Online kommunikáció** Elektronikus levelezés: kártékony csatolmányok, adathalászat, spamszűrő szolgáltatások. A böngésző adatvédelmi beállításai, előzmények, inkognitó ablak, https protokoll.	0,3
9.	**Az információs társadalom, e-Világ** Felhőszolgáltatások: kétfaktoros azonosítás, megosztási jogosultságok, az adatok elvesztésének kérdése.	0,2
9.	**A digitális eszközök használata** Rosszindulatú szoftverek és ezek elleni védekezés lehetőségei.	0,3
9.	**Mobiltechnológiai ismeretek** Okostelefonok biztonsága. Kártékony szoftverek, engedélyek. Elvesztés, képernyőzár, nyomkövetés. Biztonsági másolat. Jelszavak.	0,9
9.	**Multimédiás dokumentumok készítése** GDPR.	0,1

Évfolyam	Témakör	Tanóra
10.	**Online kommunikáció** Rosszindulatú programok és támadások elleni védekezés. Digitális lábnyom, tevékenységeink eltárolása. Sütik használata. Közösségi portálok adatvédelmi beállításai.	0,6
10.	**Az információs társadalom, e-Világ** Adathalászat. Személyes adat. GDPR. Különböző, biztonságos jelszavak használata. Többfaktoros azonosítás. Különböző oldalakon tett regisztrációk összekötésének veszélyei.	0,4
11.	**Az információs társadalom, e-Világ** Kriptográfia. Szimmetrikus és aszimmetrikus kulcsú rejtjelezés. Támadások. Hitelesítés, digitális aláírás. Hashfüggvény. Tanúsítványok. Adatvédelem böngészés közben: tanúsítványok, sütik, biztonsági beállítások.	3
	Összesen	**9 + ?**

3. táblázat: A adatvesztés és jogosulatlan hozzáférés megakadályozásának megbeszélésére fordítható idő becslése tanórában kifejezve

3.3. Az információk szándékos megosztásának lehetőségei és veszélyei

Ebbe a kategóriába olyan helyzetekre vonatkozó ismeretek tartoznak, amikor a tulajdonos az információkat szándékosan osztja meg, úgy, hogy nem foglalkozik a megosztás lehetséges veszélyivel. Feltevődik a kérdés, hogy miért tesz valaki ilyet? Természetesen lehetséges, hogy a tulajdonos nincs tisztában, vagy meggondolatlanságból nem foglalkozik ezekkel a veszélyekkel, de nem ritka az sem, hogy a valamilyen mértékben meglevő ismereteket is elhomályosítják pszichológiai tényezők. Az ilyen cselekedeteket sok esetben a hiúság, az ismerősök látszólagos felülmúlásának (vagy legalábbis az azokhoz történő felzárkózásnak) igénye, vagy éppen mások segítésének szándéka motiválja, és megvalósulási színterei többnyire a közösségi szolgáltatások. De az is viszonylag gyakori, hogy a tulajdonos nem gondolja bizonyos információkról, hogy azok visszaélésre alkalmasak lehetnek, vagy (esetlegesen indokolatlanul) megbízik a kommunikációs partnerében, és ezért oszt meg érzékeny vagy akár intim információkat is. A tipikus veszélyhelyzetek, lehetséges veszélyek minél jobb ismerete hozzájárulhat ahhoz, hogy az érintett személyek elővigyázatosabbak legyenek. Természetesen van átfedés a jogosulatlan hozzáférés eseteivel (például, amikor a tulajdonos olyan partnerrel oszt meg adatokat, aki másnak adja ki magát), és egyes adatvédelmi ismeretek (például biztonsági beállítások) ebben az esetben is szükségesek, de itt a fókusz a szándékos megosztáson, és az azzal kapcsolatos veszélyek tudatosításán van.

Ezen ismereteket (évfolyamonként az azokat magukba foglaló tankönyvbeli témakörökkel együtt), és az azokra fordítható körülbelüli időkeretet (tanórában számolva) az alábbi táblázatban összegezzük:

Évfolyam	Témakör	Tanóra
3.	**Védekezés a digitális világ veszélyei ellen** Olyan érzékeny személyes adatok megnevezése, melyeket fokozottan óvni szükséges a digitális kommunikáció során.	0,5
4.	?	?

Évfolyam	Témakör	Tanóra
5.	**e-Világ és online kommunikáció** Személyes adatok. Adatok, melyeket fokozottan óvni szükséges a digitális kommunikáció során. Az adat, mint fizetőeszköz. Digitális lábnyom, nyilvános tevékenységek, megosztott információk (feltöltött képek, videók, bejegyzések, lájkolások, médiafogyasztás stb.) és azok nem törölhető jellege. Álnév, valótlan adatok használata a kommunikációs partner által.	0,9
6.	**Az e-világ és az online kommunikáció** Internetes kommunikáció során ne osszunk meg visszaélésre alkalmas adatokat, és győződjünk meg a partner személyéről.	0,1
6.	**Digitális eszközök használata** Megosztás szempontjából veszélyes információk. Évek múlva is vállalható megosztások. A fényképezés etikája: hozzájárulás kérése a közzétételhez.	0,4
7.	**Az e-világ és az online kommunikáció** Személyes adataink, mások adatainak védelme. Levélküldéskor titkos másolat használata. Hamis személyiség, hamis weboldalak, adathalászat.	0,7
8.	?	?
9.	–	0
10.	**Online kommunikáció** Digitális lábnyom, nyilvános tevékenységek, megosztott információk (feltöltött képek, videók, bejegyzések, lájkolások, médiafogyasztás stb.), azok időszakos átnézése, törlése, évek múlva is vállalható megosztások.	0,7
11.	–	0
	Összesen	**3,3 + ?**

4. táblázat: Az információk szándékos megosztásának lehetőségeivel és veszélyeivel kapcsolatos ismeretek megbeszélésére fordítható idő tanórában kifejezve

3.4. A megbeszélt ismeretek és időkeretek elemzése

Amint korábbi cikkeinkben [13] kifejtettük, ezekben a témakörökben a konkrét ismeretek elsajátításán túl még fontosabb azok beépülése a tanulók mindennapi viselkedésébe, hogy azokat olyan körülmények között is alkalmazzák, amikor a tanár nem látja. Ehhez a tanulókat meg is kell győzni a tanultak alkalmazásának fontosságáról, aminek eszközei lehetnek konkrét veszélyhelyzetek és következmények bemutatása (akár például videó megtekintésével, saját tapasztalataik megosztásával) és azok megbeszélése.

A tankönyvekben leírt tartalmakat összegezve, úgy gondoljuk, hogy tartalmazzák azokat a legfontosabb ismereteket, melyekről ebben a témakörben szükséges beszélni. Természetesen előfordul néhány olyan ismeret, amelyekkel - akár a még készülő tankönyvekben - érdemes lenne ezeket kiegészíteni (mint például rendelkezés a szolgáltatások által tárolt személyes adatok halál utáni kezeléséről), de figyelembe véve, hogy az újdonságok folyamatos megjelenése miatt kisebb hiányosságok mindig lesznek, a tankönyvek kínálatát alapvetően jónak gondoljuk.

Az idő múlásával az ismeretek felejtődnek, a tanultak fontosságának érzése csökkenhet, továbbá a diákok a különböző életkorokban más képességekkel rendelkeznek, újabb helyzetekbe kerülhetnek, amikor az ismereteket használni kellene, ezért szükséges a tanultakra időnként visszatérni és azokat bővíteni, amit a tankönyvek többnyire meg is tesznek. Sok esetben azonban csak rövid emlékeztetőket találunk, amelyek nyilván hasznosak, de új tananyag esetén is a ráfordítható időkeret többnyire nem feltétlen elegendő a tanulók meggyőzésére a tanultak alkalmazásának szükségességéről, amit különböző életkorokban ismételten szükséges lehet megtenni. A szerzők az okostankönyvekben néhány esetben interaktív tesztekkel, videókkal segítik a tanultak elmélyítését, tehát a szándék látszik, azonban a ráfordítható időkeret ezt jelentősen korlátozza.

A digitális kultúra tantárgy információs társadalom, e-világ, online kommunikáció témakörökre fordítható időkeretein belül, a megtanítandó ismeretek mennyiségét figyelembe véve, az adatvédelemre, adatbiztonságra fordított idő nem mondható aránytalanul kevésnek, ugyanakkor a Digitális kultúra más területeit figyelembe véve már nem ez a helyzet. Az 5. táblázat tartalmazza (részben összegezve), hogy a Digitális kultúra tantárgy egyes témaköreire a kerettanterv szerint mennyi idő fordítható.

Témakör	Óraszám
Szövegszerkesztés, bemutatókészítés. Publikálás a világhálón.	63
Grafika, multimédia.	50
Algoritmizálás, programozás.	109
Táblázatkezelés. Adatbázis-kezelés.	61
Az információs társadalom, e-Világ, Online kommunikáció.	47
Digitális eszközök használata. Mobiltechnológia.	44
Összesen	**374**

5. táblázat: A Digitális kultúra tantárgy egyes témaköreire fordítható időkeretek

Úgy gondoljuk, hogy a közoktatásban kötelező jelleggel olyan kompetenciákat kellene fejleszteni, amikre mindenkinek szüksége van. Míg a digitális kultúra tantárgy egyes témaköreiben számos olyan tartalom – és ahhoz rendelt időkeret – van, ami csak néhány munkakörben lehet szükséges (például egyenletet tartalmazó vagy többhasábos szöveg szerkesztése, bonyolult műveleteket igénylő ábrák rajzolása és képek szerkesztése, adatbázisok létrehozása, az adathalmazokon matematikai számítások végzése és kimutatások készítése, webhelyek létrehozása), addig az információs társadalom, e-Világ, online kommunikáció témakörökben tanítandó kompetenciákra mindenkinek, a mindennapi életben is szüksége lenne, viszont ezek elmélyítésére a keretantervben aránytalanul kevés idő van előírva, így a tankönyvek szerzőinek is ehhez kellett alkalmazkodniuk. Ily módon, mivel a kevés ráfordított időt a 4. és 8. osztályos tankönyvek - a kerettantervi korlátok miatt - nem tudnák orvosolni, azoktól azt várjuk, hogy a kerettanterv által lehetővé tett időkeret arányos részének megfelelően segítsék az adatvédelem és adatbiztonság jellegű ismeretek mélyítését is.

4. Információbiztonság, adatvédelem az angol tantervekben

2014. szeptember 1-jétől új alaptanterv [14] lépett életbe Angliában.[4] Sajátossága, összehasonlítva a magyar tantervvel, csak célokat és témaköröket fogalmaz meg, valamint, hogy nem írja elő azt, hogy az egyes témakörökre mennyi órát kell szánni minimum. A szabályozás ezt az iskolákra bízza, azzal a kitétellel, hogy az intézményeknek elég időt kell szánniuk arra, hogy a tanterv elég széleskörű és kiegyensúlyozott legyen, és megfeleljen a törvény által előírt követelményeknek. [15]

A magyar tantervnél vizsgált évfolyamok (3-12.) az angol rendszerben az ún. Key Stage 2-nek (7-11 éves tanulók), Key Stage 3-nak (11-14 éves tanulók) és 4-nek (14-16 éves tanulók) felel meg. A magyar 3. évfolyamnak az angol 4. évfolyam felel meg.

Általános célként fogalmazza meg a tanterv, hogy a tanuló felelősségteljes, kompetens, magabiztos felhasználója legyen az információs és kommunikációs technikáknak (IKT).

A tanulóknak a következő célokat kell elérni az adatvédelem és információbiztonság terén az egyes fejlődési fokozatok végén:

Key stage 2

- A tanuló legyen képes értékelni az adatok és információk valóságtartalmát, valamint biztonságosan, felelősségteljesen és tisztelettel használja a technológiát, tudjon különbséget tenni a helyes és helytelen viselkedés között, ismerje azon módokat, ahogyan és akiktől segítséget tud kérni, illetve jelenteni tudja a nem kívánatos tartalmakat.

Key stage 3

- A tanuló ismerjen különböző felhasználási módokat, hogy miként tudja biztonságosan, tiszteletteljesen, felelősen használni a különböző technológiákat, beleértve saját online személyiségének védelmét, helytelen tartalom észrevételét, valamint a helytelen tartalom jelentését.

Key stage 4

- A tanuló értse meg, hogy a technológia változásai hogyan hatnak a biztonságra, beleértve olyan új módszerek ismeretét, amellyel meg tudja védeni a saját online magánéletét és személyes adatait, valamint észlelni és jelenteni tudja, ha ezek veszélybe kerülnek.

Az alábbi táblázatban egy konkrét iskolai tantervi ajánlás alapján vizsgáltuk meg, hogy a NAT 2020 által definiált fejlődési területek mely évfolyamon, és milyen óraszámban fordulnak elő. [16]

Évf. (magyar)	Évf. (angol)	NAT 2020 Fejlesztési feladatok és ismeretek	Angol tanterv tanóra
3.	4.	**Információszerzés az e-világban** Példák, tapasztalatok elemzése a hamis információkkal, azok felismerésével kapcsolatban	1
3.	4.	**Védekezés a digitális világ veszélyei ellen** A személyes adat fogalmának értelmezése	0,66

[4] Csak az angol alaptantervet tárgyaljuk, és nem az Egyesült Királyságét, mert a Királyság országainak joguk van saját alaptantervet készíteni

Évf. (magyar)	Évf. (angol)	NAT 2020 Fejlesztési feladatok és ismeretek	Angol tanterv tanóra
3.	4.	**Védekezés a digitális világ veszélyei ellen** A személyes adatok védelme	0,66
4.	5.	–	
5.	6.	**Online kommunikáció** Etikus és hatékony online kommunikáció a csoportmunka érdekében	1,33
6.	7.	**Az információs társadalom, e-Világ** Az informatikai eszközök használatának következményei a személyiségre és az egészségre vonatkozóan	0,2
7.	7.	**Az információs társadalom, e-Világ** Az elektronikus kommunikáció gyakorlatában felmerülő problémák megismerése, valamint az ezeket megelőző vagy ezekre reagáló, biztonságot szavatoló beállítások megismerése, használata.	2
7.	8.	–	
8.	9.	**Az információs társadalom, e-Világ** Az adatbiztonság és adatvédelem tudatos felhasználói magatartásának szabályai.	6
9.	10.	**Információs társadalom, e-Világ** Személyhez köthető információk és azok védelme	2
9.	10.	**Online kommunikáció** Az identitás kérdésének összetettebb problémái az online kommunikáció során	2
10.	10.	**Az információs társadalom, e-Világ** Az információhitelesség ellenőrzésének egyszerű módjai	1,5
10.	10.	**A digitális eszközök használata** Állományok kezelése és megosztása a felhőben	1
11.	10.	**A digitális eszközök használata** Digitális kártevők elleni védekezés	1
11.	10.	**Az információs társadalom, e-Világ** A személyes adatokkal kapcsolatos etikai szabályok és törvényi előírások	2,5

11.	10.	**A digitális eszközök használata** Tudatos felhasználói magatartás erősítése, a felelős eszközhasználat kialakítása, tudatosítása; etikus információkezelés	1
		Összesen	**23,85**

6. táblázat: A konkrét angol tanterv óraszámai és a NAT 2020. fejlesztési feladatai és ismeretei

Fontos megjegyeznünk, hogy az angliai 10-11. évfolyamoknál külön van tanterv az alapszintű és az emelt szintű érettségire. A fenti táblázatban az alapszintű érettségire felkészítő tantervet vettük figyelembe, ugyanis Magyarországon is általában fakultáción, szakkörön vagy külön felkészítőn vesz részt az a tanuló, aki emelt szintű érettségit kíván tenni informatikából / digitális kultúrából.

A vizsgált évfolyamokon, az angliai tantervi ajánlás 261 órát szán informatikára, ami kevesebb, mint a magyar, viszont Angliában az első évfolyamtól kezdve folyamatosan vannak informatika órák, tehát az általunk vizsgált évfolyamok előtt is. Angliában 1-11. évfolyamon, az alapszintű érettségire felkészítő tanterv összesen 369 órát szán informatikára, ami lényegében megegyezik a magyar óraszámmal.

A tantervek összehasonlítása azt is megmutatta, hogy hasonló témákat, területet tárgyal a két ország tanterve, időben vannak apróbb eltérések, vannak témák, amit az angol tanterv hamarabb vesz elő, mint a magyar, de ez nem jellemző.

Fontos különbség azonban, hogy az angol tanterv arányaiban majdnem kétszer annyi időt (9,1%) szán az adatvédelemmel és informatikai biztonsággal kapcsolatos témakörökre, mint a magyar (a vizsgált évfolyamoknál).

Másik fontos különbség a két ország tantervei között, hogy az angliai tanterv a 9. évfolyam végén egy 6 órás, valamint a 10. évfolyam elején egy 10 órás összefüggő blokkot szán a témakörre. Fontosnak tartjuk hangsúlyozni, hogy ahhoz, hogy tudatos viselkedés alakuljon ki a tanulókban, ahhoz megfelelő idő és fókuszálás szükséges. Úgy gondoljuk, hogy a NAT 2020 által megadott órakeretek, illetve az általunk vizsgált magyar tantervi javaslat alapján ez nehezen kivitelezhető, így hosszabb tematikus időszakok szervezését – mint pl. az Európai Kiberbiztonsági Hónap[5] – ajánljuk. Ez lehetőséget teremthet más informatikai kompetenciák használatára, valamint tantárgyközi kapcsolatok megvalósítására is (pl. az adatvédelem egyes témái kapcsolódhatnak az állampolgári ismeretek és a mozgóképkultúra és médiaismeret tárgyakhoz).

5. Összefoglalás

Munkánkban megvizsgáltuk a jelenlegi Digitális kultúra tankönyveket, hogy mely témákat érintik adatvédelem és információbiztonság témakörökből.

Elmondható, hogy a tankönyvek az összes szükséges témát érintik, viszont a tanórák arányát kevésnek gondoljuk az informatikára szánható össz-óraszámot tekintve. Vizsgálatunk azt mutatta ki, hogy az adatvédelemre és információbiztonságra szánt óraszám lényegében nem nőtt a 2012-es kerettantervhez képest, holott a tantárgyra fordítható kötelező óraszám nőtt.

[5] Az ENISA (Európai Uniós Kiberbiztonsági Ügynökség) által koordinált figyelemfelhívó kampány, amelynek célja a kiberbiztonsági tudatosság növelése, valamint a kibertérben megjelenő fenyegetések széles körben történő megismertetése.

Az informatikai biztonságtudatosságot, mint kompetenciát, olyan tudásnak tartjuk, amire bármelyik embernek szüksége van. Továbbá, a témakör kihat az informatika más területeire is, pl. képszerkesztésnél lehet beszélni az etikus képmegosztásról, hangsúlyozva, hogy milyen személyes adatokat tartalmazhat egy fénykép, a digitális eszközöknél pedig lehet beszélni a tárolt adatok védelméről, ahogyan ezek a tankönyvekben meg is jelennek.

Az érintett témaköröket illetően nem találtunk lényegi különbséget az angol és a magyar tantervek között, viszont az angol tanterv arányosan közel kétszer annyi órát használ az adatvédelem és információbiztonság témájára, mint a magyar, ráadásul az óraszám jelentős részét blokkosítva, ami jobban támogatja azt, hogy mélyebben hasson a tanulók viselkedésére. Ilyen fajta és mértékű fókuszt ajánlunk ahhoz, hogy egy átlagos diák valóban biztonságtudatosan tudja befejezni középiskolai tanulmányait.

Irodalom

1. Smahel, D., Machackova, H., Mascheroni, G., Dedkova, L., Staksrud, E., Ólafsson, K., Livingstone, S., and Hasebrink, U. (2020). EU Kids Online 2020: Survey results from 19 countries. EU Kids Online. https://doi.org/10.21953/lse.47fdeqj01ofo
2. OECD 2019. PISA 2018 Results (Volume I): What Students Know and Can Do, PISA, OECD Publishing, Paris https://doi.org/10.1787/5f07c754-en
3. Törley Gábor: Adatvédelem, adatbiztonság, biztonságtudatosság tanítása Angliában és Magyarországon, in. Szlávi Péter, Zsakó László (szerk.) INFODIDACT 2015. Konferencia helye, ideje: Zamárdi, Magyarország, 2015.11.26 Budapest: Webdidaktika Alapítvány, 2015. Paper 16. 10 p. ISBN 978-963-12-3892-1
4. European Commission / EACEA / Eurydice, 2022. Informatics education at school in Europe. Eurydice report. Luxembourg: Publications Office of the European Union.
5. Dr. Lénárd András, Sarbó Gyöngyi, Tarné Éder Marianna, Turzó-Sovák Nikolett: Digitális kultúra tankönyv 3. osztály, Oktatási Hivatal, 2021 (utoljára megtekintve: 2022.11.18.)
6. Lénárd András, Abonyi-Tóth Andor, Turzó-Sovák Nikolett, Varga Péter: Digitális kultúra tankönyv 5. osztály, Oktatási Hivatal, 2020. Okostankönyv. (utoljára megtekintve: 2022.11.18.)
7. Abonyi-Tóth Andor, Farkas Csaba, Turzó-Sovák Nikolett, Varga Péter: Digitális kultúra tankönyv 6. osztály, Oktatási Hivatal, 2020. Okostankönyv. (utoljára megtekintve: 2022.11.18.)
8. Abonyi-Tóth Andor, Farkas Csaba, Varga Péter: Digitális kultúra tankönyv 7. osztály, Oktatási Hivatal, 2021. Okostankönyv. (utoljára megtekintve: 2022.11.18.)
9. Varga Péter, Jeneiné Horváth Kinga, Reményi Zoltán, Farkas Csaba, Takács Imre, Siegler Gábor, Abonyi-Tóth Andor: Digitális kultúra tankönyv 9. osztály, Oktatási Hivatal, 2020. Okostankönyv. (utoljára megtekintve: 2022.11.18.)
10. Abonyi-Tóth Andor, Farkas Csaba, Jeneiné Horváth Kinga, Reményi Zoltán, Tóth Tamás, Varga Péter: Digitális kultúra tankönyv 10. osztály, Oktatási Hivatal, 2020. Okostankönyv. (utoljára megtekintve: 2022.11.18.)
11. Abonyi-Tóth Andor, Farkas Csaba, Fodor Zsolt, Jeneiné Horváth Kinga, Reményi Zoltán, Siegler Gábor, Varga Péter: Digitális kultúra tankönyv 11. osztály, Oktatási Hivatal, 2021. Okostankönyv. (utoljára megtekintve: 2022.11.18.)
12. Oktatási Hivatal: Mintatanmenetek https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat/mintatanmenetek (utoljára megtekintve 2022.11.18.)
13. Holló Csaba, Témák és ötletek az információkezelés tanítására , INFODIDACT 2021 (2021. 11. 18-19), 14. Informatika Szakmódszertani Konferencia, elektronikus kiadványa, 35-47 , Webdidaktika alapítvány, 2022 január, ISBN: 978-615-80608-5-1. (utoljára megtekintve: 2022.11.18.)

14. National curriculum in England: computing programmes of study, 2013., https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study (utoljára megtekintve: 2022.11.18.)
15. European Commission/EACEA/Eurydice (2015):Recommended Annual Instruction Time in Full-time Compulsory Education in Europe 2014/15. Eurydice – Facts and Figures. Luxembourg: Publications Office of the European Union.
16. National Centre for Computing Education (NCCE) – Teach Computing https://teachcomputing.org/curriculum (utoljára megtekintve: 2022.11.18.)

Robotépítés és robotprogramozás virtuális környezetben

Veronika Stoffová[1], Martin Zboran[2], Hana Hyksová[3]

[1]veronika.stoffa@gmail.com; [2]mazboran@gmail.com; [3]hanahy@seznam.

[1]Faculty of Education of Trnava University in Trnava (Slovakia) & ELTE IK

[2]Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava (Slovakia)

[3]Faculty of Education, Palacký University in Olomouc (Czech Republic)

Absztrakt. Az általános iskolát látogató gyerekek a játékos tanítási formát részesítik előnyben. Az alternatív oktatási stratégiák alapvető célja, hogy a tanuló ne csak passzív befogadója legyen az ismereteknek, hanem aktívan vegyen részt a tanulási folyamatban. A didaktikai játék során a tanulók spontán módon sajátítanak el bizonyos ismereteket, amelyek összhangban vannak az oktatási célokkal, anélkül, hogy tudatosítanák, hogy tanulnak. A programozás játékos oktatásának egyik lehetősége a programozható játékok és robotok programozása. Sajnos az iskolák nincsenek kellőképpen felszerelve programozható robotokkal vagy robotkészletekkel és nagyon nehéz egyforma tanulási feltételeket és környezetet biztosítani minden diáknak. Tapasztalataink azt igazolják, hogy a programozható játékok, valamint a robotkészletek megfelelően helyettesíthetők ezen eszközök emulátoraival és szimulátoraival. Ezek jelentős előnye az is, hogy használhatók a távoktatásban is. A szimulációs környezetben való programozás ugyanolyan szórakoztató, mint egy igazi robot programozása. A robotvezérlésre összeállított program vizualizált szimulációs kísérlettel is tesztelhető a virtuális térben.

Kulcsszavak: programozás, programozható játékok, robotkészletek, robotprogramozás, szimuláció, emuláció.

1. Bevezető

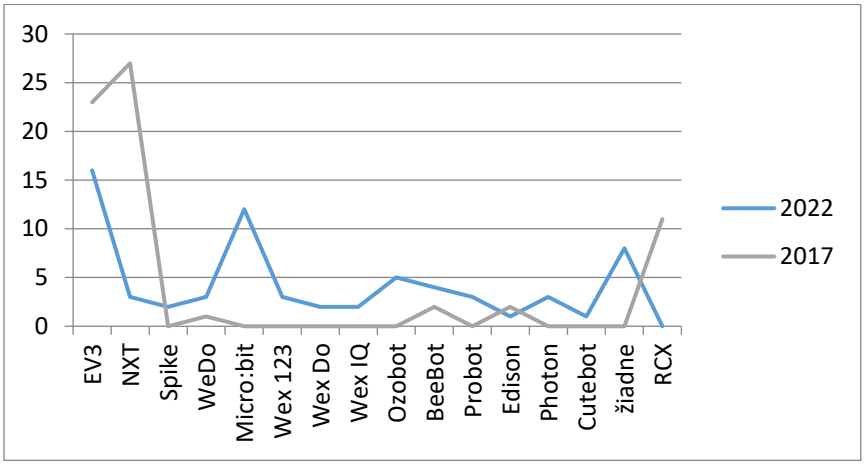
A szlovák általános iskolákban az elmúlt tanévekben a COVID-19 világjárvány miatt kombinált formában folyt a tanítás. A távoktatási formával lehetőség nyílt a tanítási órák csökkentésére, a tanulmányi terv egyes részeinek kihagyására. A gyakran kihagyott tantárgyrészek speciális felszereléssel rendelkező tantermekben és laborokban tartott gyakorlati órákhoz kötődtek, esetleg speciális eszközök vagy eszközök használatához, mint például laboratóriumi gyakorlatok és kísérletek, robotok építése, programozás, testnevelés stb. A távoktatás alatt az ilyen tanítási órák sokszor törölve voltak, vagy módosított formában voltak megvalósítva.

Az általános iskolai állami oktatási program szerint az algoritmikus és programozói gondolkodás fejlesztésére van szükség. Ezen cél elérésére az algoritmikus problémamegoldásra és programozásra összpontosító tematikus egységek vannak részletesen kidolgozva és a követelmények standardok formájában kifejezve [1].

A cikk szerzői a robot építésből robotok programozásából szerzett tapasztalataikat szeretnék megosztani az olvasóval, amely egyaránt sikeresen megvalósítható jelenléti és távoktatási formában, valós vagy virtuális robotok és robotkészletek segítségével. A szerzők a tanulás színesítésére és hatékonyságának növelésére korszerű oktatási eszközöket és digitális oktatási technológiát alkalmaztak, amelyek a távoktatásban is beváltak. Virtuális valóságot használtak, távoli laboratóriumokban dolgoztak, és laboratóriumi kísérleteket végeztek vizualizált szimulációs modellek és környezetek, valamint emulációs és szimulációs technikák segítségével. A szerzők beszámolnak az általános iskolai progra-

mozás oktatása során szerzett tapasztalataikról programozható játékok, robotok és mikrokontrollerek segítségével [2, 3, 4, 5, 6].

Kutatásaik azt mutatják, hogy az általános iskolák nincsenek megfelelően felszerelve robotjátékokkal és robotkészletekkel. Ezt mutatják az alábbi gráfba foglal 2017 és 2022-ben végzett kutatásaink eredményei. Még 2017-ben az LegoMindstorms EV3 és LegoMindstorms NXT vezetett, az utóbbi években a Micro:bit van az élen (1. gráf).



1. gráf: A szlovákiai általános iskolák felszereltsége robotjátékokkal és robotkészletekkel (százalékban kifejezve) [6].

2. A játékok programozásától a robotprogramozásig

A játék- és speciális szaküzletekben programozható játékok egész sora megtalálható, és az érdeklődőnek van miből választani bármely korosztály számára. Ezek közül csak néhányat mutatunk be, amelyeket gyakran programozható játékként terjesztenek különféle kiegészítőkkel általános iskolások számára. Ezen szórakoztató robotkombináció különböző korosztályok számára alkalmas, az óvodásoktól az általános iskolát látogató gyermekekig. Használatuk nem igényel programozási készségeket és tapasztalatokat. A tanulók játékos formában sajátítják el a parancsok végrehajtásának szimbolikus jelölését, egyfajta kódolást, miközben fejlesztik a logikai, algoritmikus és programozói gondolkodásukat [6, 7, 8, 9, 10, 11]. Némelyek közülük nemcsak egyszerű, igénytelen végrehajtandó parancssorok létrehozását teszik lehetővé, hanem hosszabb és összetettebb programok létrehozását is, amelyekbe procedúrákat, szenzorvezérlési eljárásokat lehet beintegrálni [12, 13, 14]. Például alakzatok rajzolása, vonal követése, akadályok elkerülése és így tovább. Az összetettebbek a bemenetek és kimenetek széles skálájával rendelkeznek egy olyan programhoz, amely a szerkesztés befejezése és a futás kiválasztása után azonnal végrehajtásra kerülnek.

Példaként említenénk a programozható méhecskét (Bee-Bot- és Blue-Botot), az intelligens programozható autót (Pro-Bot-ot) és ezekhez hasonló más programozható játékot.

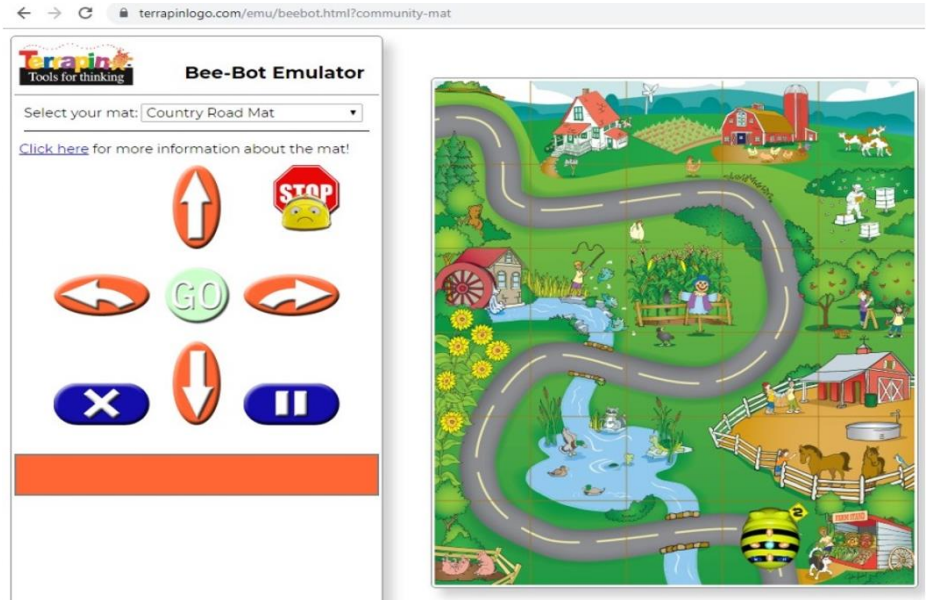
2.1. Bee-Bot

A Bee-Bot egy egyszerű, interaktív programozható játék, amellyel a tanulók játékosan sajátíthatnak el új ismereteket különböző területeken. Alkalmas betűk és számok azonosítására, síkban való tájékozódás gyakorlására stb. A "méhecske" irányítására "gomb" parancsok sorozatának létrehozásával a gyerekek fejlesztik a logikus gondolkodásukat, a térbeli tájékozódásukat és a játékos módon történő tervezési képességüket. Ezért a programozható méhecske fontos szerepet játszik a programozás

propedeutikában. A méhecske hátán lévő gombokkal jelzett utasítások használhatók a robotméhecske vezérlésére. (Lásd az 1. ábrát).

A Bee-Bot könnyen használható felhasználói felülettel rendelkezik. A memóriájában 40 utasításból álló sorozat tárolható. A gombok a következő parancsokat képviselik: előre/hátra mozgás, 90 fokkal balra/jobbra fordulás, adott időegység kivárása (kb. 1 s). A vezérlőgombokkal bevitt parancsok a GO gombbal hajthatók végre, amellyel elindítható a program végrehajtása a memóriában. Új algoritmus létrehozásához a Clear gombbal törölhetjük a parancsokat a memóriából. Lehetőség van négyzetrács (15 x 15) alapján játéktáblákat/területeket létrehozni/tervezni a méhek mozgatásához.

1. ábra: Két nézet a programozható méhecskére



2.ábra: Bee-Bot emulátor (a méhecske kiindulási helyzete).

3. ábra: Bee-Bot emulátor (a méhecske helyzete a parancssorozat végrehajtása után).

A négyzetekből álló területen a méhecske mozoghat és végrehajthatja a program utasításait. A tanulók saját alkalmazásokat készíthetnek a méhecske mozgatására, fejlesztve kreativitásukat és elképzelőerejüket. A méhecskék irányítását szolgáló programok létrehozásának sikere erős motiváló hatással van más érdekes feladatok megoldására, ami egyúttal segíti a logikai, algoritmikus és programozói gondolkodás alapjainak továbbfejlesztését és megszilárdítását. A Bee-Bot programozható játék előnye, hogy emulátorral is rendelkezik, ami felbecsülhetetlen segítséget jelent a távoktatásban és az olyan iskolákban, ahol nincs ilyen programozható játék. A Bee-Bot emulátor megfelelő helyettesítőnek tekinthető egy igazi játékszer pótlására [3, 12]. Egy egyszerű program végrehajtása az alapvető utasítások sorozatának gombparancsokkal történő bevitelén alapul (lásd 2. és 3. ábrát).

A méhecske hátán lévő gombokkal jelzett utasítások használhatók a robotméh vezérlésére. (Lásd 1. ábrán). A Bee-Bot könnyen használható felhasználói felülettel rendelkezik. A memóriájában 40 utasításból álló sorozat tárolható. A gombok a következő parancsokat képviselik: előre/hátra mozgás, 90 fokkal balra/jobbra fordulás, adott időegység kivárása (kb. 1 s).

A vezérlőgombokkal bevitt parancsok a GO gombbal hajthatók végre, amellyel elindítható a program végrehajtása a memóriában. Új algoritmus létrehozásához a Clear gombbal törölhetjük a parancsokat a memóriából. Lehetőség van négyzetrács (15 x 15) alapján játéktáblákat/területeket létrehozni/tervezni a méhek mozgatásához. A négyzetekből álló területen a méh mozoghat és végrehajthatja a programozó/játékos utasításait. A tanulók saját alkalmazásokat készíthetnek a méhek mozgatására, fejlesztve kreativitásukat és képzelőerejüket. A méhek irányítását szolgáló programok létrehozásának sikere erős motiváló hatással van más érdekes feladatok megoldására, ami egyúttal segíti a logikai, algoritmikus és programozói gondolkodás alapjainak továbbfejlesztését és megszilárdítását. A bee bot programozható játék előnye, hogy emulátorral is rendelkezik, ami felbecsülhetetlen segítséget jelent a távoktatásban és az olyan iskolákban, ahol nincs programozható játék. A Bee Bot emulátor megfelelő helyettesítőnek tekinthető egy igazi játék számára [3, 12]. Egy egyszerű program végrehajtása az alapvető utasítások sorozatának gombparancsokkal történő bevitelén alapul (lásd 2. és 3. ábrát).

2.2. Blue-Bot

A Blue-Bot nagyon hasonlít a Bee-Botra. A bekapcsolás és a GO gomb megnyomása után lépések sorozatát lehet beírni. A GO ismételt megnyomásával megkezdődik a beírt parancsok sorozatának végrehajtása. Ha a tanuló hibás programot ír, amely nem teljesíti a feladatot, lehetőség van további parancsok hozzáadására, a robot visszahelyezésére az eredeti pozícióra és az elkészült program újraindítására.

4. ábra: Blue-Bot felül- és alulnézetből

A robot nem rendelkezik kijelzővel, így a program nem szerkeszthető. Ha valamelyik tematikus szőnyeg van használva, amiben 15 x 15 cm-es négyzetrács található, a gyerekek különböző érdekes feladatokat hajthatnak végre a Blue-Bot segítségével.

Lásd a https://www.nextech.sk/a/Robotika---co-dokazu-roboty-Blue-Bot-a-Pro-Bot

2.3. Pro-Bot

A Pro-Bot robotot játékautó ruhába öltötték. Lehetővé teszi a változó centiméterben kifejezett mozgáshossz megadását, elforgatási szög megadását, az így megadott parancsok alapján tud haladni és jelölővel az megtett utat kirajzolni. A felhasználó programozhat ciklusokat és használhat eljárásokat is. A robotautó kijelzős, így a program szerkeszthető. (https://www.vyuka-vzdelanie.sk)

5. ábra: Pro-Bot – programozható játékautó tollal és toll nélkül, majd a további funkciók menüje

A Pro-Bot robot beépített érintésérzékelőkkel rendelkezik az első és a hátsó lökhárítóban, egy fény- és hangérzékelő van az első motorháztetőn. Az USB-n keresztül a robot nem csak tölthető, hanem a külön mellékelt Probotix szoftveren keresztül a PC-n is programozható.

A mozgást biztosító kerekek közötti tengelyben (a játékautó kerekei csak dekoratívak) egy tartóval ellátott lyuk található, amibe jelölőt lehet behelyezni és különböző formákat rajzolni. Alapértelmezés szerint a robot 25 cm-es lépésekben mozog, de a lépés hossza megadható. A parancs utáni paraméter a mozgás hosszát jelzi cm-ben. Programozhat ciklusokat is, ahol az első paraméter a szögletes zárójelben a parancssorozat ismétlődéseinek száma a ciklusban.

A fő program és két eljárás egy hat szirmú virág rajzolásához a következő 6. ábrán látható.

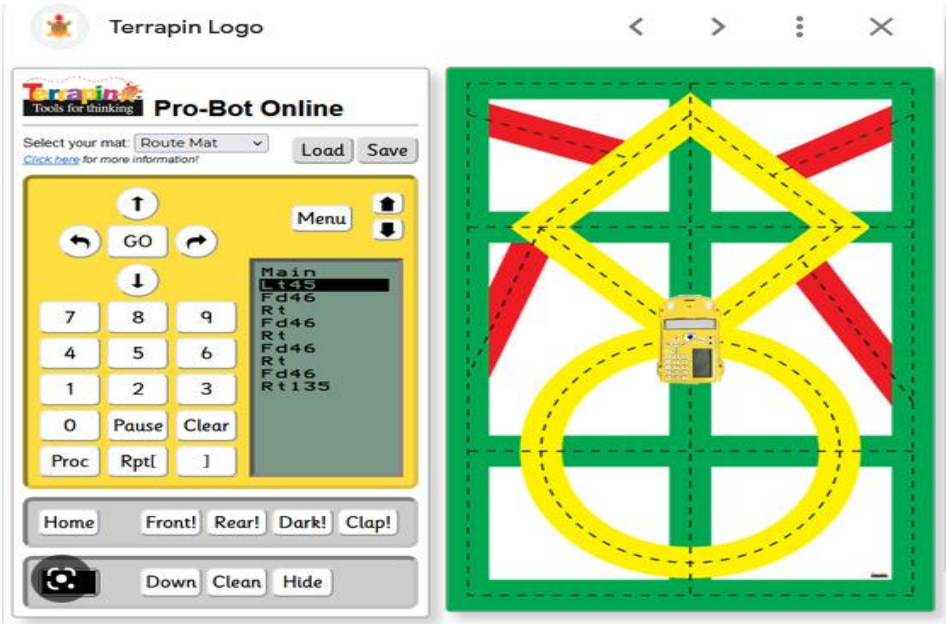
```
Main          Proc 1        Proc 2
Rpt 6 [       Proc 2        Rpt 10 [
Proc 1        Lt 120        Fd 1
]             Proc 2        Lt 6
              Lt 60         ]
```

6. ábra: Pro-Bot emulátor és környezete

A robot 32 felhasználó által meghatározott eljárás alkalmazását teszi lehetővé. Más eljárások az érzékelőktől érkező események kezelésére szolgálnak, így például a FRONT eljárásban programozható, hogy mit tegyen a robot, ha az első lökhárítójával akadályba ütközik. Az eljárási lehetőségek a https://www.nextech.sk/files/photo/2019-07/62946/293b08/Snimka-obrazovky-2019-07-17-o-20-00-32.png linken elérhető videófelvételben találhatók.

A Menü gomb hosszan tartó megnyomása után megjelenik a további funkciók menüje, amely az 5. ábra jobb oldalán látható.

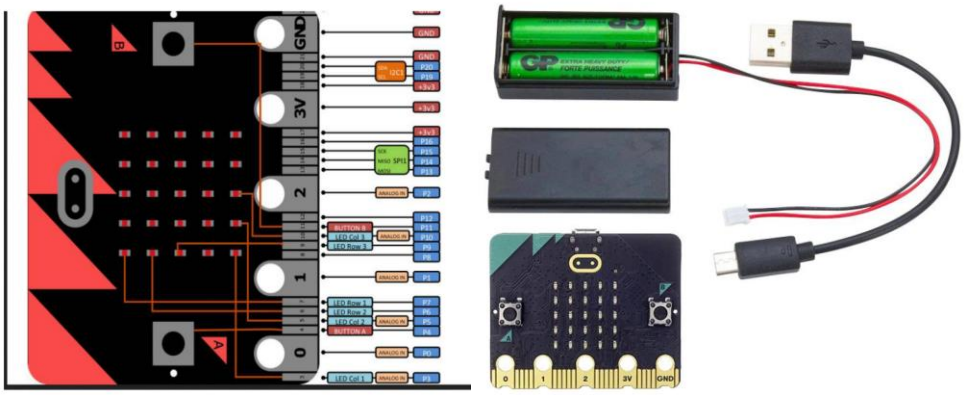
A Pro-Bot online virtuális környezetben is programozható a Pro-Bot emulátor segítségével [1, 16]. Így eszközhiány esetében és a távoktatásban is használható.



7. ábra: Pro-Bot emulátor és környezete

2.4. Micro:bit

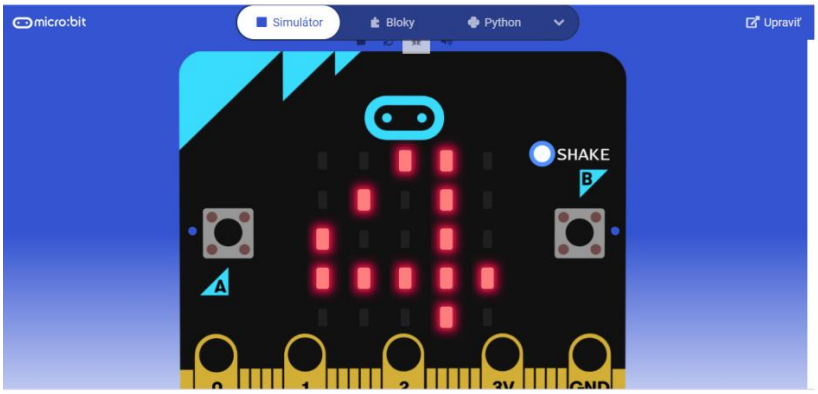
Az általános iskolákban egyre gyakrabban jelenik meg a micro:bit kit, ezért ezt a robotkészletet választottuk példaként a programozáshoz. Ebben a mikrokontrollerben több érzékelő is található, ami növeli a népszerűségét a felhasználók körében. Az érzékelők között megtalálható a nyomásérzékelő, a fényintenzitás-érzékelő, a gyorsulás- és dőlésérzékelő, a hőmérséklet-érzékelő és a mágneses térérzékelő (8. ábra).



8. ábra: A micro:bit készlet

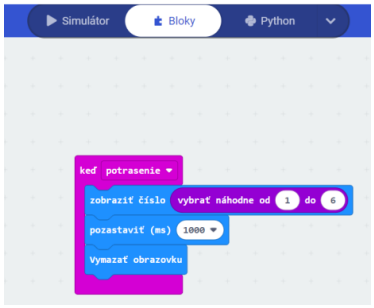
3. A micro:bit programozása

Példaként egy egyszerű feladatot választottunk. A gyorsulás- és dőlésérzékelő segítségével egy digitális dobókockát programozunk, amely ha eredményként a „dobás után" egy 1-6 intervallumba tartozó véletlen számot jelenít meg a LED képernyőn. A szám 1 másodperc múlva törlődik (9 ábra). (https://www.conrad.sk/p/micro-bit-mirco-bit-kit-microbit-v2-go-bundle-2308377?&vat=true&gclid=Cj0KCQiAveebBhD_ARIsAFaAvrEYqMRY3FNLGwzrS2Wkl07-3fP98frcS6-L-9_HCcEcWwTvUmTOogwaAtEFEALw_wcB)

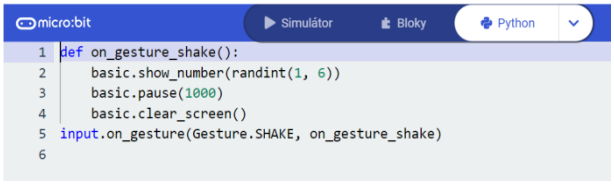


9. ábra: Digitális dobókocka a micro:.bit-tel szimulálva

Az INPUT blokkok menüjéből választottuk ki a „shake blokkot", a Basic blokkból mutasd a számot, hogy ne mindig ugyanaz a szám jelenjen meg, a Matematika blokkot használjuk - véletlenszerűen válasszunk 0-6 között. A megjelenített szám jobb megfigyelhetősége érdekében az Alapblokk képernyőről a szünetet és a törlést választottuk, ezzel biztosítva a LED panelen megjelenített szám törlését a beállított idő letelte után (9. ábra). A kód blokkból történő összeállítása során a blokkok egymásba illeszkednek, így összeállításuk intuitív és egyszerűbb, mint a Python nyelvű programozás esetén (10.-11 ábra), ahol szintaktikai hibákra is figyelnie kell a tanulónak.



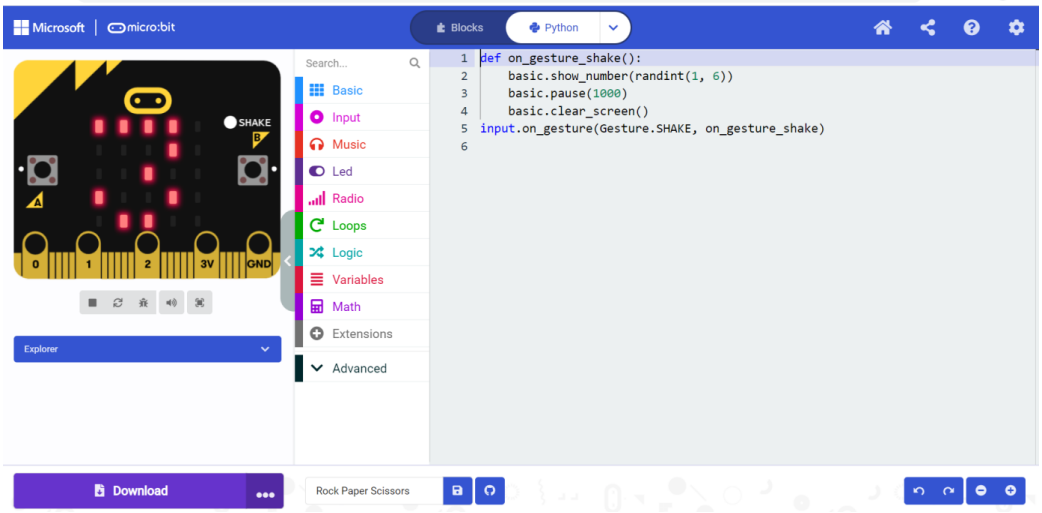
10. ábra: Dobókocka programozása blokkok segítségével



11. ábra: Dobókocka programozása Pyton programozási nyelven

További példák a dobójátékkocka programozásának különböző módjaira a következő képeken találhatók (10.-12. ábra). A "dobókocka" program egy másik eredménye blokkokból összeállított

program formájában a 9. ábrán látható. A 10. ábra ezen feladat megoldását a Python programozási nyelven mutatja be.



12. ábra: Dobókocka programozása interaktív Python programozási környezetben

4. Befejezés

Programozható játékok, robotok, mikrokontrollerek stb. az iskolai gyakorlatban bevált eszközök a programozás alapjainak elsajátítására - a logikai algoritmikus és programozói gondolkodás fejlesztésére. Kutatásunk eredményeinek megfelelően a gyakorlati programozásra orientált tantárgyak megjelennek az informatika oktatási programokban (nemcsak az informatika tanárképzésben, hanem az általános iskolák számára is) [1, 15]. E tantárgyak elvégzése keretében a leendő informatika tanárok gyakorlati tapasztalatot szereznek a diákok alapvető programozási készségek kialakításában és a gyerekek programozásra való tanításához. Új tankönyvek jelennek meg, amelyek támogatják a robotjátékokkal, robotokkal és mikrokontrollerekkel való munkát. Különféle programozási környezetek jöttek létre az interaktív programozáshoz. A leendő tanárok és a gyakorlatban dolgozó tanárok módszertani támogatást kapnak az egyetemektől, hogy ezeket az eszközöket bevezessék az informatika oktatásába. A pedagógusok élethosszig tartó gyakorlati oktatásának és átképzésének részeként különböző tanfolyamok és oktatóprogramok szervezésével segítik őket az első akadályok leküzdésében a tanítási folyamat hatékony, játékos és eredményes tanítási mód felé történő korszerűsítésében.

A robot építőelemek általános iskolai programozásra való alkalmazása nemcsak a programozható játékok és robotépítő építőelemek népszerűségét növeli, hanem magát a programozást és népszerűségét az általános iskolás gyerekek körében, valamint a középiskolai és főiskolai műszaki képzési területek iránti érdeklődést is támogatja. A virtuális eszközök, emulációk és szimulációk használata lehetővé tette a diákok számára, hogy a pandémia idején is dolgozhassanak online programozási eszközök és virtuális környezetek használatával távoktatás formájában.

KÖSZÖNETNYILVÁNÍTÁS

A tanulmány kidolgozását a KEGA 013TTU 4/2021 „Interaktív animációs és szimulációs modellek a mély tanuláshoz" (Interactive animation and simulation models for deep learning) szlovák nemzeti projekt és az ELTE IK Informatikatudományi Intézete támogatta.

Irodalom

1. Informatika - primárne vzdelávanie. https://www.statpedu.sk/files/articles/dokumenty/inovovany-statny-vzdelavaci-program/informatika_pv_2014.pdf, 2014 (utoljára megtekintve: 2022. 10. 31.)
2. Stoffová, V.: Computer Games as a Tool for Development of Algorithmic Thinking In: The European Proceedings of Social & Behavioural Sciences EpSBS, 2018 pp. xxx-xxx. http://dx.doi.org/ Corresponding Author: Selection and peer-review under responsibility of the Organizing Committee of the conference eISSN: 2357-1330
3. Czakóová, K., Stoffová V.: Hravá forma rozvíjania algoritmického myslenia na základnej škole = A Playful Form of Developing Algorithmic Thinking in Primary School /. In: Didinfo 2020 [electronic] : sborník konference : [medzinárodní konference o vyučování informatiky] / [bez zostavovateľa]. - Liberec : Technická univerzita v Liberci, 2020. - ISBN 978-80-7494-532-8. - ISSN 2454-051X. - online, S. 104-111
4. Stoffová, V.: Educational Computer Games in programming Teaching and Learning. In: New technologies and redesigning learning spaces: eLearning and Software for Education. Bucuresti : Carol 1 National Defence University, 2019. ISSN 2066-026X, CD-ROM, p. 39-45. WoS. DOI 10.12753/2066-026X-19-004 (2019)
5. Stoffová, V., Zboran, M.: My virtual School – I attend school virtually. In: *INTED2022 Proceedings 16th International Technology, Education and Development Conference* March 7th-8th *2022.* (Edited by: A. López Martínez, L. Gómez Chova, I. Candel Torres), [1. ed.] – Valencia : IATED Academy, 2022, pp. 5563-5569, ISBN: 978-84-09-37758-9 / ISSN: 2340-1079 doi: 10.21125/inted.2022
6. Stoffová, V., Zboran, M.,Gabaľová, V.: Iconic and block programming for teaching programming basic for primary school pupils. In:, *15th annual International Conference of Education, Research and Innovation*, Seville (in press)
7. Weintro, D., Wilensky, U.: Playing by Programming: Making Gameplay a Programming Activity. https://ccl.northwestern.edu/2016/playingbyprogramming.pdf 2016, (utoljára megtekintve: 2022.10.31.)
8. Végh, L., Takáč, O.: Online games to introducing computer programming to children. In: L. Gómez Chova – A. López Martínez – I. Candel Torres (eds.): INTED2021 Proceedings. 15th International Technology, Education and Development Conference, 8th-9th March, 2021. pp. 10007– 10015. ISBN: 978-84-09-27666-0, ISSN: 2340-1079. (WOS)
9. Végh, L., Takáč, O.: Mobile coding games to learn the basics of computer programming. In: L. Gómez Chova – A. López Martínez – I. Candel Torres (eds.): EDULEARN21 Proceedings. 13th International Conference on Education and Learning Technology, 5th-6th July, 2021. pp. 7791–7799. ISBN: 978-84-09-31267-2, ISSN: 2340-1117. (WOS)
10. Czakóová, K., Udvaros, J.: Applications and games for the development of algorithmic thinking in favor of experiential learning. In. EDULEARN21 : Proceedings of the 13th International Conference on Education and New Learning Technologies. DOI: 10.21125/edulearn.2021.1389, p. 6873-6879, Valencia : IATED Academy, 2021. ISBN 978-84-09-31267-2. ISSN 2340-1117.
11. Czaková, K.: Game-based programming in primary school informatics. In. INTED 2021 Proceedings of the 15th International Technology, Education and Development Conference. Valencia : IATED Academy, 2021.
12. Hyksová, H. Programování robotů na základní škole. (Robot programming at elementary school) In: *DIDINFO 2021*. Univerzita Mateja Bela, Banská Bystrica, 2021. s. 81-85. ISBN 978-80-557-1823-1. ISSN 2454-051X.

13. Udvaros, J., Czakóová, K.:. Developing of computational thinking using microcontrollers and simulations. In. EDULEARN21 : Proceedings of the 13th International Conference on Education and New Learning Technologies. DOI: 10.21125/edulearn.2021.1619. , p. 7945-7951, Valencia : IATED Academy, 2021. ISBN 978-84-09-31267-2. ISSN 2340-1117. (WOS)
14. Csóka, M., Czakóová, K.: Innovations in education through the application of raspberry pi devices and modern teaching strategies. In. INTED 2021 Proceedings of the 15th International Technology, Education and Development Conference. Valencia : IATED Academy, 2021Stoffová, V., Zboran, M.: Teaching construction and programming of robots in a distance form. In: Proceedings of the 15th International Technology, Education and Development Conference. Valencia : IATED Academy, 2021,, pp. 4911-4918, ISBN: 978-84-09-27666-0, ISSN: 2340-1079, doi: 10.21125/inted.2021.0991(WoS)
15. Szlávi, P., Zsakó, L.: *Informatics as particular field of education*. Teaching Mathematics and Computer Science, 3 (2005) 2. Institute of Mathematics University of Debrecen (2005) 283–294
16. https://www.google.com/search?sxsrf=ALiCzsYjsDf158_4rsWWazZec9H99-xrBw:1668906030168&source=univ&tbm=isch&q=probot+emulator&client=firefox-b-d&fir=wILwWfqqeLmzyM%252Cawbr68ttLvVkdM%252C_%253B10lFhUf338cL_M%252Cawbr68ttLvVkdM%252C_%253BtNN5FlC5JB24YM%252CnQa_WAdgWHp8RM%252C_%253B_8irarPMvD9aUM%252Cawbr68ttLvVkdM%252C_%253Bp3AqIfKtKmtUlM%252CiHTOfb0GgsZ4yM%252C_%253BoH31mREzgQrE3M%252C1m8koVnYBeKv9M%252C_%253B9AtYVIS18JvZcM%252CzOINC2nZv-cW1M%252C_%253BwcLm45DsuygN5M%252C9gZuuBKmbhLlCM%252C_%253Bm-MybORcUioe-M%252CW6RRXUSrmj_VbM%252C_%253Bor3JQ2r9mLKYsM%252C3AMEYzcU7SEDVM%252C_&usg=AI4_-kS7pKVCUqBimG8pPr57wevIBl2s_w&sa=X&ved=2ahUKEwizwt24x7v7AhWDQ_EDHVVnCFcQjJkEegQIJBAC&biw=1827&bih=905&dpr=1#imgrc=or3JQ2r9mLKYsM (utoljára megtekintve: 2022.10.31.)
17. https://www.conrad.sk/p/micro-bit-mirco-bit-kit-microbit-v2-go-bundle-2308377?&vat=true&gclid=Cj0KCQiAveebBhD_ARIsAFaAvrEYqMRY3FNLGwzrS2Wkl07-3fP98frcS6-L-9_HCcEcWwTvUmTOogwaAtEFEALw_wcB) (utoljára megtekintve: 2022.10.31.)

Agent JS – Ügynökvezérelt programozás JavaScripttel

Visnovitz Márton[1], Horváth Győző[2]

{[1]visnovitz.marton,[2]horvath.gyozo}@inf.elte.hu
ELTE IK

Absztrakt. A Logo programozási nyelv nagy hagyományokkal rendelkezik az informatika oktatásában Magyarországon, diákok generációit vezette be a programozás világába. A teknőcgrafika, illetve a grafikus programozás a mai napig nagyon népszerű módszerek a programozás alapjainak bemutatására. Idővel számos új eszköz, nyelv, környezet jelent meg, mely lehetővé tette az ilyen jellegű programok készítését (pl. Scratch, Python with Turtle), sőt a Logo nyelv egyik változatával, a NetLogo-val már nemcsak egy, hanem több teknőcből (ügynökből) álló rendszereket, komplex modelleket is készíthetünk. Ebben a cikkben egy fejlesztés alatt álló lehetőséget mutatunk be, mely lehetővé teszi, hogy JavaScript programozási nyelven, könnyedén készítsünk ilyen, akár több ügynökös programokat, modelleket. Ezáltal lehetővé válik, hogy a Python with Turtle rendszerhez hasonlóan, egy az iparban is jelentős, de az oktatásban is egyre inkább elterjedt nyelven tudjuk teknőcgrafikán, modellezésen keresztül programozni tanítani a diákokat.

Kulcsszavak: teknőcgrafika, ügynökvezérelt modellezés, JavaScript, Logo, NetLogo

Bevezetés, motiváció

A Logo programozási nyelv[1] 1967-es megjelenése óta hatalmas hatással volt a programozás oktatására. A nyelv, és a hozzá tartozó programozási környezetek egyszerű eszközökkel teszik lehetővé komplex grafikák készítését, a programozás számos eszközének, fogalmának megismerését [1, 2]. A teknőcgrafika mint témakör direkt módon volt jelen az előző, 2012-es Nemzeti Alaptantervben (NAT) [3], az új, 2020-as NAT-ban [4] a robotika témakör részeként jelenik meg. Magyarországon a mai napig népszerű módszer, rengeteg tanár alkalmazza az iskolában. A legutóbbi, 2022. évi Országos Grafikus Programozási Versenyen is közel 1000 diák indult el[2].

A Logo programozási nyelv és a teknőcgrafika alapja egy szereplő (teknőc, ügynök) számítógépes programmal való vezérlése, irányítása. Ennek a megközelítésnek az egyik lehetséges továbbfejlesztési iránya az úgynevezett többügynökös (többteknőcös) modellek programozása. Ezekben a programok-ban már nem csak egy, hanem több ügynököt vezérelhetünk, minden ügynök egy önálló szereplő a modellben. Az ilyen jellegű programok létrehozására is léteznek környezetek, az ügynökvezérelt modellezés oktatási felhasználásának egyik úttörője a NetLogo[3]. Megalkotója, Uri Wilensky, a Logo nyelv szellemiségében egy olyan környezetet alkotott, melyben „alacsony a belépési küszöb, de a határ a csillagos ég" (eredeti: "low threshold and no ceiling"), és ami remekül használható a programozás tanítására, a számítógépes gondolkodás fejlesztésére [5, 6, 7]. Az ügynökvezérelt modellek készítésének egyik nagy előnye, hogy segítségükkel nemcsak programozást taníthatunk, ha-

[1] https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html

[2] https://njszt.hu/hu/news/2022-02-08/1000-gyerek-indult-el-grafikus-programozas-versenyen

[3] https://ccl.northwestern.edu/netlogo/

nem változatos modellek segítségével kombinálhatjuk az informatikai ismeretek tanítását más tantárgyakkal is. A NetLogo környezethez készített modell-könyvtárban[4] találhatunk példákat többek között biológia, kémia, fizika, matematika és társadalomtudományok témakörökben is.

A teknőcgrafika, illetve a hozzá kapcsolódó robotika, modellezés/szimuláció, fontos szerepet töltenek be a programozás oktatásában. Ezen stratégiák lehetőséget biztosítanak, hogy a tanított korosztálynak megfelelő módszerekkel tudjuk tanítani az informatikával, számítástudománnyal kapcsolatos ismereteket [8]. Ezeken a területeken a Logo megjelenése óta rengeteg új lehetőség, eszköz, környezet látta meg a napvilágot, mint például a korábban említett NetLogo, a Scratch[5], mely blokkprogramozással tette lehetővé a szereplők irányítását, vagy a Python with Turtle[6] könyvtár, mely a népszerű Python programozási nyelven biztosít eszközöket ahhoz, hogy teknőcgrafikus programokat hozzunk létre. Egy érdekes új kezdeményezés továbbá az XLogo[7] webes környezet, mely különböző korcsoportoknak biztosít egyre bővülő eszköztárat teknőcgrafikus programok létrehozására: a legkisebb korosztályoknak blokkalapú programozással, az idősebbeknek pedig a Python with Turtle környezet segítségével.

Az új eszközöket illetően megfigyelhető trend a programozásoktatás világában a blokkalapú programozás, valamint a szkriptnyelvek használata. A Python mellett a másik szkriptnyelv, amivel gyakran találkozhatunk oktatási környezetekben a JavaScript (pl. CodeCombat[8], Grasshopper[9], Khan Academy[10]). Ez a szövegalapú programozási nyelv követi a blokk alapú kódolást a Micro:bit[11] esetében is. Több tanulmány is igazolja, hogy a JavaScript jó tulajdonságokkal rendelkezik a programozás tanításához [9, 10].

A Python mint (az oktatásban is) népszerű szkriptnyelv már rendelkezik teknőcgrafikai lehetőséggel, mellette a JavaScript nyelv egy új irányt jelenthet a programozott grafikában, ügynökvezérlet modellezésben. Ennek támogatására jött létre az Agent JS projekt.

Célok, filozófia

Az Agent JS projekt célja egy olyan JavaScript programkönyvtár készítése, ami lehetővé teszi ügynökvezérelt programok készítését JavaScript nyelven, különböző komplexitási szinteken. Segítségével könnyedén készíthetünk ábrákat teknőcgrafika segítségével a Logo nyelvhez hasonlóan, vagy többügynökös, illetve sejtautomata modelleket a NetLogo mintájára. Ehhez a könyvtárban hozzáférhető osztályokat kell kombinálnunk a JavaScript nyelv natív eszközeivel (pl. ciklusok, elágazások).

A könyvtár által biztosított eszköztár motivációját a Python with Turtle és a NetLogo környezetek adták. Célunk az, hogy az ezek által nyújtott lehetőségeket minél jobban lefedjük. Egy olyan könyvtárat szeretnénk létrehozni, ami sokféle lehetőséget kínál, de azok használatára nem kényszerít rá. Ennek megfelelően a könyvtár úgy került kialakításra, hogy lehetőség van csak a számunkra ép-

[4] https://ccl.northwestern.edu/netlogo/models/index.cgi

[5] https://scratch.mit.edu/

[6] https://docs.python.org/3/library/turtle.html

[7] https://xlogo.inf.ethz.ch/

[8] https://codecombat.com/

[9] https://grasshopper.app/

[10] https://www.khanacademy.org/

[11] https://microbit.org/

pen szükséges elemek betöltésére. Így például, ha egy egyszerű, „egyteknőcös” rajzot szeretnénk készíteni, akkor nincs szükségünk az időzítő osztály betöltésére.

A könyvtár számos olyan kisegítő lehetőséget is tartalmaz, amelyek megkönnyítik bizonyos programozási feladatok megoldását, de ezek használata is opcionális a tanítási céltól függően. Például, a könyvtár tartalmaz segédfüggvényeket a szögek *fok* és *radián* közötti átváltásához. Ezeket szabadon használhatjuk egy alacsonyabb évfolyamon, ahol a diákok még nem feltétlenül tanulták az erre vonatkozó matematikai ismereteket, de gimnáziumban a tanulók már maguk is implementálni tudják ezeket a függvényeket a JavaScript nyelv beépített eszközeivel.

Felépítés, technológia

A JavaScript programozási nyelv jó alapokat ad grafikus programkönyvtárak készítéséhez. A böngészőben elérhető Canvas API[12] révén rendelkezésre áll egy beépített eszköztár a programozott rajzolásra, segítségével rendkívül komplex grafikák is készíthetők natív eszközökkel. Használatában az egyetlen dolog, ami nehézséget jelent kezdők számára a HTML nyelv és az azt vezérlő JavaScript API közötti összefüggés megértése. A fejlesztés során igyekeztünk minél közelebb maradni a natív eszköztárhoz, de kiküszöbölni az azzal kapcsolatos nehézségeket. Ennek megfelelően az Agent JS könyvtár több vékonyabb és vastagabb absztrakciós réteget épít a böngésző natív programozási interfészei fölé.

A könyvtár a modern webes sztenderdeknek megfelelően JavaScript modulként[13] lett megvalósítva, és a GitHub[14] kódtár segítségével van publikálva. Betöltése URL-en keresztül lehetséges, de amennyiben letöltjük a forráskódot a helyi számítógépre, internethozzáférés nélkül is használható. Az egyes osztályok és segédfüggvények úgynevezett „named import”-ok[15] formájában férhetőek hozzá.

```
import { Model } from "https://vimtaai.github.io/agent/lib/index.js";
```

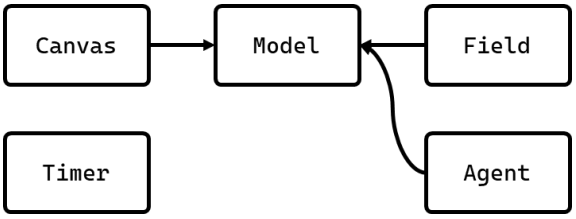
A könyvtár alapját a `Canvas` osztály [11] adja, mely a natív `HTMLCanvasElement` típuson (`<canvas>` HTML tag) alapszik, annak lehetőségeit kombinálja a Canvas API programozó interfészével, a `RenderingContext2D` osztállyal. Erre, és az időzítők használatát megkönnyítő `Timer` osztály [11] fölé épül a `Model` absztrakció. Ez az osztály reprezentálja az ügynökvezérelt modellünket. A `Model` osztály példányának létrehozásakor automatikusan létrejön a modell megjelenéséért felelős `Canvas` példány, illetve automatikusan feltöltődik a modell a sejtautomata-elvű szimulációk működéséhez szükséges `Field` objektumokkal is. Ezek a `Field` objektumok a NetLogo környezet „patch” típusú ügynökeinek az analógiái. A modellhez ezen kívül van lehetőségünk ügynököket („teknőcöket”) hozzáadni az `Agent` osztály példányainak létrehozásával. A könyvtárban definiált osztályok kapcsolatát az 1. ábra szemlélteti.

[12] https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

[13] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules

[14] https://github.com/

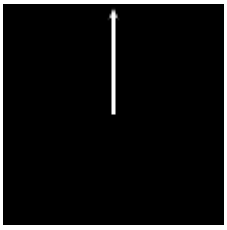
[15] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import#named_import



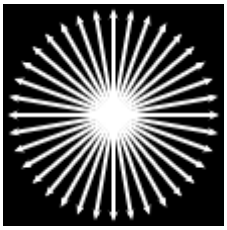
1. ábra: Az Agent JS könyvtár osztályai

A hozzáadott ügynökök számától függően készíthetünk sejtautomata-elvű modelleket (nincs mozgó ügynök, csak egyhelyben álló ügynökök – mezők – vannak), egyszerű teknőcgrafikát (1 mozgó ügynök) vagy többügynökös modellt (tetszőleges számú mozgó ügynök) . Ezek a lehetőségek tetszés szerint kombinálhatóak is. A könyvtár használatára példát a 2. ábra mutat.

```
const model = new Model();
const agent = new Agent({
  x: model.centerX,
  y: model.centerY
});

model.addAgent(agent);
agent.putPenDown();
agent.forward(10);
```



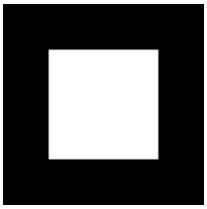
```
const model = new Model();

for (let i = 0; i < 36; i++) {
  const agent = new Agent({
    x: model.centerX,
    y: model.centerY,
    heading: i * 10
  });

  model.addAgent(agent);
  agent.putPenDown();
  agent.forward(10);
}
```



2. ábra: Példa együgynökös és sokügynökös rajzolásra

Habár a könyvtár fejlesztésének elsődleges fókusza a teknőcgrafika és az ügynökvezérelt programozás volt, a komponensei úgy lettek felépítve, hogy az alacsonyabb szintű absztrakciók külön is használhatóak legyenek. Ezek segítségével – teknőcök és a modellek nélkül – kizárólag a rajzolást és az animációt támogató osztályok (`Canvas` és `Timer`), valamint natív JavaScript függvények segítségével is van lehetőség ábrák, animációk vagy akár játékok készítésére (3. ábra) [12]. Ezek az alacsonyabb szintű absztrakciók korábbi tapasztalataink alapján úgy lettek kialakítva, hogy a natív JavaScript eszközöket csupán annyira „okosítsák fel", hogy csak a használat szempontjából kényelmetlen, a megértést nehezítő részek legyenek elfedve, de amennyire lehetséges, natív függvényhívásokkal legyenek programozhatóak [13].

```
const canvas = new Canvas();

canvas.fillStyle = "black";
canvas.fillRect(0, 0, 100,
100);
canvas.fillStyle = "white";
canvas.fillRect(25, 25, 50,
50);
```



3. ábra: Egyszerű alakzatok rajzolása az „ügynök” absztrakció nélkül, natív eszközökkel

A könyvtár az ügynökvezérelt programozáshoz szükséges absztrakciókon túl számos segédfüggvényt tartalmaz, melyek megkönnyíthetik a programjaink elkészítését. Ezek a függvények három kategóriába vannak sorolva: véletlenszám-generáló függvények, geometriai függvények, illetve típushelyesség-ellenőrző függvények. Ezek használata teljesen opcionális, működésüket a diákok maguk is leprogramozhatják, de a kezdőbbek, fiatalabbak számára segítséget nyújthatnak abban, hogy könnyebben készíthessenek minél látványosabb programokat.

A `Canvas` és a `Model` absztrakciókhoz kihasználtuk a JavaScript nyelv és a böngészők egy viszonylag új, és elterjedőben lévő lehetőségét, a webkomponensek[16] definiálását. Ezzel a módszerrel saját HTML elemeket tudunk létrehozni, vagy egy meglévő elem lehetőségeit bővíteni. Ilyen formán a `Canvas` típus a natív `<canvas>` elem kibővítésén, a `Model` osztály pedig egy egyedileg definiált HTML elemen alapul.

Habár az Agent JS könyvtár által biztosított osztályok használatához HTML ismeretekre nincs szükség, a webes, böngészőbeli környezet jellegéből adódóan mégis szükség van egy minimális HTML fájlra programjaink futtatásához. Ahhoz, hogy a böngészőben JavaScript kódot futtassunk, a forráskódunkat tartalmazó `.js` kiterjesztésű fájlt vagy külső szkriptfájlként kell hivatkozni, vagy közvetlenül a HTML fájlba kell írni a forráskódot `<script>` tag-ek közé. Ezen túl további HTML elemek hozzáadására nincs szükség, azok automatikusan beillesztésre kerülnek az oldalra amikor szükséges.

```
<script type="module">
// Ide írjuk a forráskódot
</script>
```

```
<!-- A program.js fájlba írjuk a forráskódot --
>
<script type="module"
src="program.js"></script>
```

Természetesen ettől függetlenül az oldalhoz van lehetőség további HTML elemeket beilleszteni amennyiben szeretnénk. Ezeket bemenetként vagy kimenetként használhatjuk a programunkban, hasonlóan a NetLogo környezet vezérlőihez. A programunk ilyen irányú kibővítése – azon túl, hogy a modellünk tudását bővíti – kiváló lehetőség a HTML nyelv, illetve a natív JavaScript programozás további részeinek megismerésére. Egyszerű eseménykezeléssel lehetőségünk nyílik a modellünket vezérlő gombokat, a modell paramétereit tartalmazó beviteli mezőket, vagy a modellünk állapotát kijelző mezőket létrehozni.

---

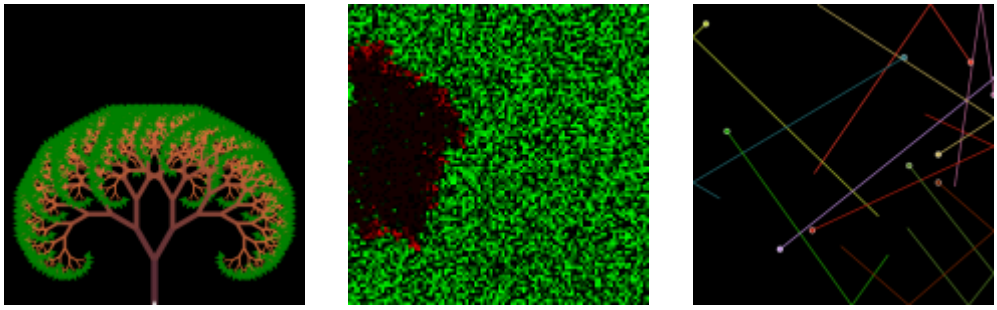
[16] https://developer.mozilla.org/en-US/docs/Web/Web_Components

Dokumentáció

Az Agent JS mindenki számára szabadon hozzáférhető, nyílt forráskódú szoftver. A forráskód[17] és a dokumentáció[18] bárki számára elérhető a GitHub online kódtárban, bővítésükhöz, fejlesztésükhöz bárki hozzájárulhat.

A dokumentáció tartalmazza az osztályok és segédfüggvények részletes leírását, illetve rövid összegzést a könyvtár használatáról. Ezen túl a dokumentáció részét képezi néhány valós, élő példa, melyek a könyvtárban rejlő lehetőségeket mutatják be (4. ábra). Az alábbi feladattípusok mindegyikéhez készültek példák[19]:

- mozaikrajzolás, rekurzív ábra rajzolása teknőccel (parketta minta, rekurzív fa)
- sejtautomata modell (élet játéka, erdőtűzmodell)
- többügynökös modell (labdák pattogása, naprendszermodell)



4. ábra: Különféle rajzok és modellek az Agent JS könyvtárral
(balról jobbra: rekurzív fa, erdőtűzmodell, labdák pattogása)

Ezeken a példákon keresztül láthatjuk, hogy hogyan oldhatók meg különböző jellegű és komplexitású feladatok az Agent JS segítségével. A feladatok megoldásra révén szerzett tapasztaltokat felhasználtuk a könyvtár továbbfejlesztésére.

Jövőbeli tervek

Habár az Agent JS könyvtár már jelenlegi formájában is alkalmas számos programozásoktatási stratégia alkalmazására, a jövőben további lehetőségekkel tervezzük fejleszteni. A fejlesztések elsődleges iránya, hogy a könyvtár alkalmas legyen a Python with Turtle és a NetLogo környezetek által nyújtott lehetőségeket minél jobban lefedni. Reményeink szerint, ha a könyvtárat elkezdik használni az iskolákban, akkor a tanárok tapasztalatai alapján további ötleteket kaphatunk arra vonatkozóan, hogy mivel lehetne a könyvtárat használhatóbbá, kényelmesebbé tenni.

Ahhoz, hogy minél kevesebb előkészülettel legyen lehetőség elkezdeni dolgozni a környezettel, a terveink között szerepel az is, hogy az Agent JS könyvtárat legyen lehetőség valamilyen online szerkesztőprogram segítségével, közvetlenül a böngészőből használni. Ezáltal külső fejlesztőkörnyezet és a HTML fájlok használatára nélkül is lehetne modelleket készíteni.

[17] https://github.com/vimtaai/agent

[18] https://vimtaai.github.io/agent/

[19] https://vimtaai.github.io/agent/examples

Amennyiben a könyvtárnak sikerül nagyobb népszerűségre szert tenni, szeretnénk kezdeményezni, hogy az Országos Grafikus Programozási Verseny választható környezetei között szerepeljen az Agent JS is.

Összefoglalás

A teknőcgrafika a mai napig népszerű módja a programozásoktatásnak, használatát több programozási nyelv és környezet is támogatja. Az Agent JS projekt azért jött létre, hogy JavaScript nyelven is legyen lehetőségünk teknőcgrafikus alkalmazásokat, ügynökvezérlet modelleket készíteni. Használatával a tanárok egy újabb eszközt kapnak ahhoz, hogy a tanulói igényekre szabott módszerekkel tanítsanak programozást, valamint programozási témaköröket, ismeretköröket kapcsoljanak össze.

Irodalom

1. Seymour Papert, Cynthia Solomon: *Twenty Things to Do with a Computer*, Artificial Intelligence Memo No. 248, (1971)
2. Seymour Papert: *Mindstorms. Children, Computers and Powerful Ideas*, Basic Books Inc., Harper Colophon Books, (1981)
3. *Nemzeti Alaptanterv 2012. (110/2012. (VI. 4.) Kormányrendelet)*, Magyar Közlöny 2012 66. sz. (2012) https://magyarkozlony.hu/dokumentumok/f8260c6149a4ab7ff14dea4fd427f10a7dc972f8/letoltes
4. *Nemzeti Alaptanterv 2020. (5/2020.(I. 31.) Kormányrendelet)*, Magyar Közlöny 2020 17. sz. (2020), https://magyarkozlony.hu/dokumentumok/3288b6548a740b9c8daf918a399a0bed1985db0f/letoltes
5. Seth Tisue, Uri Wilensky: *NetLogo: A Simple Environment for Modeling Complexity*. Proceedings of International conference on complex systems, Boston (2004) pp. 16-21.
6. Uri Wilensky: *Modeling nature's emergent patterns with multi-agent languages*. Proceedings of the Eurologo 2001 Conference, Linz (2001).
7. Bernát Péter: *Modelling and simulation in education and the NetLogo simulation environment*, Teaching Mathematics and Computer Science, 2014. Vol. 12, Num. 2, pp. 229-240.
8. Bernát Péter, Zsakó László: *Methods of teaching programming – strategy*, XXX. DIDMATTECH (2017), Trnava. pp. 40-50.
9. Visnovitz Márton: *Szöveges programozási nyelvek a közoktatásban*, Szakdolgozat
10. Horváth, Gy., Menyhárt, L. (2014). *Teaching introductory programming with JavaScript in higher education*, Proceedings of the 9th International Conference on Applied Informatics, Eger, Hungary. pp. 339-350.
11. Visnovitz Márton, Horváth Győző: *JavaScript könyvtárak programozott rajzolás alapú tanulás támogatásához,* InfoDidact 2020, (2020)
12. Horváth Győző, Menyhárt László, Zsakó László: *Viewpoints of programming didactics at a web game implementation,* XXIX. DIDMATTECH (2016) Budapest, pp. 79–88
13. Visnovitz Márton, Horváth Győző., *A Constructionist Approach to Learn Coding with Programming Canvases in the Web Browser*, *CONSTRUCTIONISM 2020*, (2020), pp. 1–8

BBC micro:bittel vezérelt kinematikai mérések kiértékelése MS Excel-bővítménnyel

Somogyi Anikó[1], Kelemen András[2], Mellár János[3], Mingesz Róbert[4]

{[1]somogyia, [3]mellar, [4]mingesz}@inf.u-szeged.hu;

[1,3,4] SZTE TTIK Műszaki Informatika Tanszék

[2]kelemen@jgypk.szte.hu

[2] SZTE JGYPK Informatika Alkalmazásai Tanszék

[1,2] Szegedi Radnóti Miklós Kísérleti Gimnázium

Absztrakt. Magyarországon a NAT 2020-as fizika kerettanterv előírja a mozgások - egyenes vonalú egyenletes mozgás, egyenes vonalú egyenletesen változó mozgás, egyenletes körmozgás, harmonikus mozgás, ingamozgás - kvantitatív jellemzését és vizsgálatát. A tanárok és a diákok számára javasolt az elérhető számítógépes eszközök és programok használata a mérések elvégzése és kiértékelése során. Az előadásban bemutatunk olyan mikrokontroller-vezérelt mérési kísérleteket, amelyek alkalmasak a mozgásokat leíró fizikai mennyiségek időfüggésének bemutatására. A mérési elrendezés micro:bitet, néhány optikai szenzort és 3D nyomtatott elemeket is tartalmaz. A micro:biten futó mérőszoftvert C++-ban (CODAL) fejlesztettük. A küldő mikrokontroller a mért adatokat vezeték nélküli kommunikáción keresztül juttatja a számítógéphez csatlakoztatott fogadóhoz, és egy általunk fejlesztett Excel-bővítmény szolgál az adatok soros kommunikáción történő fogadására és kiértékelésére. A kísérletezés nem csak a méréssel támogatott fizikaoktatást támogathatja, de a valódi mérés adatai az informatika órán is érdekes adalékot adhatnak a tananyaghoz.

Kulcsszavak: BBC micro:bit, mikrokontroller-vezérelt mérés, egyszerű mozgások, optikai szenzorok, vezeték nélküli kommunikáció, soros kommunikáció, Excel-bővítmény

1. Bevezetés

A mozgások tudományos igényességgel történő vizsgálata a 16. században vált kvantitatív mérésekre alapozott természettudománnyá. Galilei pontos összefüggést fogalmazott meg az idő, valamint a szabadon eső vagy a lejtő segítségével lelassított tárgy elmozdulása között [1]. Azt is felfedezte, hogy az inga periódusideje független a tömegtől [2]. Ezek az egyszerű mozgások képezik a középiskolai mechanika, azon belül a kinematika témakör alapját manapság is [3]. Kutatási módszerének hatása több száz év után is megfigyelhető: az ún. tudományos megismerés módszere továbbra is fontos része a magyarországi természettudományos oktatás tananyagának. Ennek egyik legfontosabb lépése a kísérleti adatgyűjtés és -elemzés, amely a 21. században nemcsak a hagyományos mérési technikákkal, hanem a műszaki informatika korszerű eszközeivel is megvalósítható. Magyarországon a NAT 2020-ban megjelent fizika kerettanterv az egyenes vonalú egyenletes mozgás, az egyenes vonalú egyenletesen gyorsuló mozgás, az egyenletes körmozgás, a harmonikus rezgőmozgás és a matematikai inga mozgásának ismeretét várja el a tanulóktól. A tanterv továbbá javasolja a rendelkezésre álló számítógépes eszközök és programok alkalmazását a mérések és kiértékelésük során [4]. Ez az újítás azért üdvözlendő, mert egyrészt egyfajta interdiszciplináris, azaz a számítástechnikát és természettudományt egységként kezelő oktatásra ad lehetőséget, másrészt pedig a tanulói aktivitásra.

Ezzel párhuzamosan a digitális kultúra oktatásában jelentős szerephez jut a robotika témaköre, amely természetesen magában foglalja a szenzoros méréseket [4]. Ilyen módon szintén a fizika és az

informatika egyfajta fúziója valósulhat meg az oktatásban. A digitális kultúra tankönyvekben a témakörhöz kapcsolódóan a nemzetközileg is igen népszerű és elismert, kifejezetten oktatási célokra tervezett BBC micro:bit eszköz [5–8] programozása kiemelt szerephez jut, amelynek egyik vélhető oka, hogy az esetleges ideiglenes eszközhiány részben átugorható a *MakeCode*, esetleg a *micro:bit Python Editor* szimulátora segítségével [9–13]. Számos segédletet és tanulmányt elérhet az érdeklődő tanár és diák az ELTE IK által megvalósított micro:bit botorkálás honlapján [14], melyek közül kiemelkedik a szakköri tananyaguk [15]. Ezek segítségével elsajátítható a micro:bit V2 beépített (fényerő-, hang-, gyorsulás-, hőmérséklet-, mágnesességérzékelő) szenzorainak alapvető használata, így segítséget nyújthatnak a fizikaoktatás szereplőinek is.

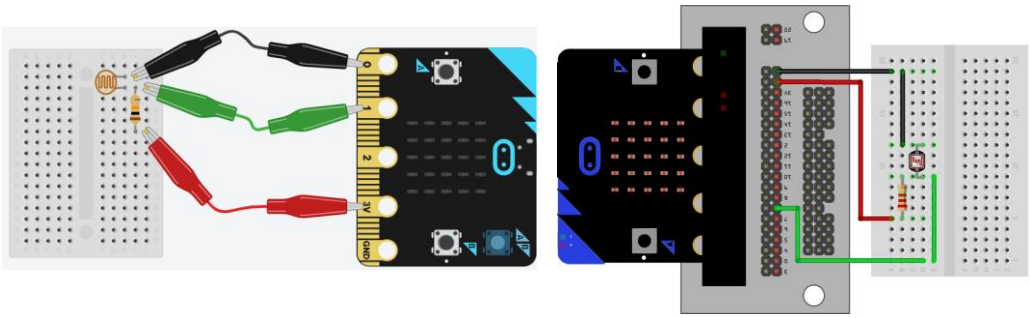
Az MTA-SZTE Műszaki Informatika Szakmódszertani Kutatócsoport (MISZAK), melynek munkájában e cikk szerzői is részt vesznek, többek között céljául tűzte ki, hogy olyan módszereket, eszközöket fejleszt, amelyek támogatják, hogy a tanulók számítógéppel segített mérések, illetve a könnyen elérhető mikrokontroller-alapú eszközök (Arduino, BBC micro:bit) programozása révén elsajátítsák a mérnöki gondolkodás alapjait [16]. Ezzel összhangban számos kísérletet is fejlesztettek és publikáltak, amelyek beilleszthetők a fentebb leírt kerettantervi elvárások közé [17–19]. A tantárgypedagógiai kutatócsoport honlapján mind a fizikát, mind pedig az informatikát oktató tanárok találhatnak magyar nyelvű segédleteket a műszaki informatikai tartalmak saját tantárgyaikba történő beemeléséhez [20].

A mért adatok felvétele mellett kiemelt fontosságú az adatok igényes kiértékelése. A Microsoft Office táblázatkezelő programja, az Excel jó ideje szerepel a fizikaoktatás módszertanát a számítógép bevonásával modernizálni kívánó tantárgypedagógiai kutatók által javasolt szoftverek között [3,21,22]. Számos publikáció született, amelyek bemutatták a BBC micro:bit, illetve az Arduino Excellel történő összekapcsolását számítógéppel segített mérőkísérletekben [23–28].

Ebben a tanulmányban először bemutatunk egy BBC micro:bitre épülő, 3D nyomtatott és otthon elkészíthető elemeket tartalmazó kísérleti elrendezést. Ezután egy C++ nyelven írt, micro:biten futó mérőszoftvert, továbbá a mért adatok PC-n történő fogadáshoz egy C# nyelven írt Excel-bővítményt. A kiértékelést az Excel beépített függvényei és bővítményei segítségével valósítjuk meg. Ez a hardver-szoftver együttes alkalmas lehet mikrokontroller-vezérelt mérések oktatásban történő alkalmazására, azon belül a kerettanterv által előírt kinematikai jelenségek számítógéppel segített, kvantitatív vizsgálatára.

2. A BBC micro:bitre épülő kísérleti elrendezés

A BBC micro:bit beépített szenzorai mellett csatlakoztathatunk a micro:bithez további szenzorokat. A micro:bit szélén lévő csatlakozókhoz első közelítésben krokodilcsipesszel lehet érzékelőket csatlakoztatni, de javasolt egy egyszerű tüskesorral (vagy esetleg forrpontos csatlakozóval rendelkező) kiegészítő panel, ún. *breakout board* [29] alkalmazása, melynek segítségével az összes kivezetett csatlakozóhoz könnyen és stabilan csatlakoztathatunk áramköri elemeket, így sokkal jobban ki lehet használni az eszközben rejlő lehetőségeket.

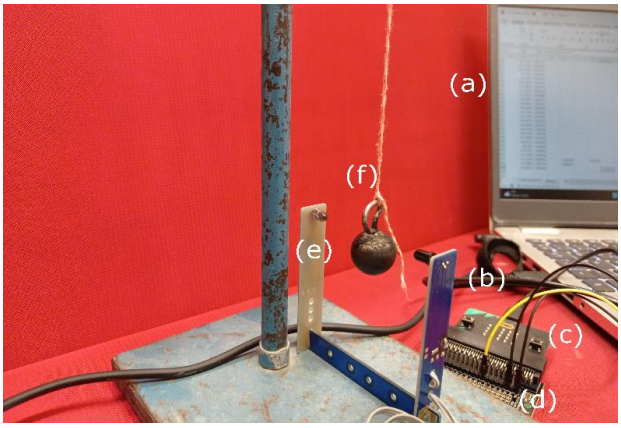


1. ábra: Próbapanelen sorba kötött fotoellenállás és ellenállás csatlakoztatása a micro:bit csatlakozósorához a) krokodilcsipeszekkel[1], illetve b) breakout board beiktatásával[2].

2.1. Transzmissziós fotokapuval kiegészített micro:bit

Egy egyszerű, a látható fénytartományban működő fotokapu vevőjeként szolgálhat akár a kijelző LED-mátrix is fényerősségmérő szenzorként konfigurálva [30], vagy egy egyszerű fotoellenállás is [31]. Korábbi munkánkban bemutattuk a saját készítésű fotokapu lehetőségét [27]. A kereskedelmi forgalomban általában infravörös tartományban működő LED-et és egy vele szemben álló fototranzisztort/fotodiódát tartalmazó áramkört alkalmaznak. Ezúttal egy ilyen, a MISZAK keretében fejlesztett fotokaput alkalmaztunk [17].

Ebben az esetben nincs szükség vezeték nélküli kommunikációra, így a micro:bitet USB-kábellel kapcsolhatjuk a PC-hez, amivel egyúttal a micro:bit tápellátását is biztosítjuk. A fotokapu referenciafeszültséget a micro:bit GND és 3V csatlakozóiról kaphat, míg a jelvezetéket pedig a micro:bit 2-es bemenetére csatlakoztathatjuk. Ezzel az egyszerű áramkörrel a periodikus mozgások (ingamozgás, körmozgás) periódusideje mérhető nagy pontossággal [32], de egy ejtőlétrával kiegészítve szabadesést is lehet vele vizsgálni [33]. A 2. ábrán látható elrendezést alkalmaztuk a fonálinga perióduside-jének mérésére.



2. ábra: Az inga lengésidejének méréséhez összeállított kísérleti elrendezés: a) számítógép b) USB-kábel c) micro:bit, d) breakout board e) fotokapu f) inga állványra rögzítve

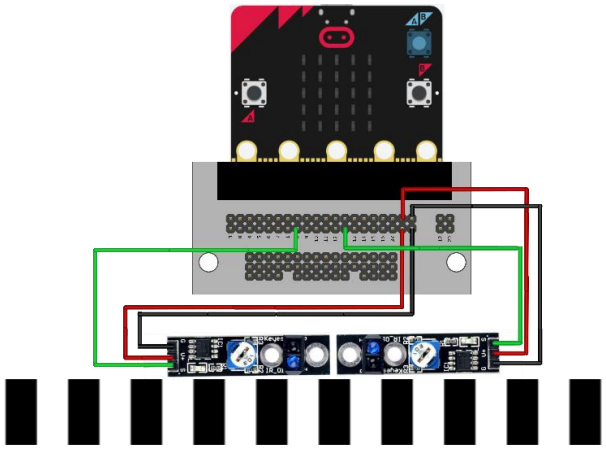
[1] Az ábra a Tinkercad szimulátorban készült.

[2] Az ábra a Fritzing alkalmazásban készült.

2.2. Reflexiós fotokapuval (vonalkövető szenzorral) kiegészített micro:bit

Egyenes vonalú mozgást végző test pozíciójának folyamatos detektálását megvalósíthatjuk a szögelfordulás mérésére alkalmazott kvadratúra-enkóder jeladókkal [34,35] analóg módon működő kísérleti elrendezéssel. Első lépésben a mozgás pályájára (esetünkben egy légpárnás sín[3] oldalfalának aljára, a légfúvó lyuksorok alá) ragaszthatunk egy fekete-fehér (IR elnyelő és visszaverő) matt felületeket váltogató öntapadós enkóderszalagot, ehhez A4-es nyomtatható tervet mellékeltünk. A lovason elhelyezett micro:bit valamely digitális bemenetére ezúttal reflexiós IR adó-vevőt (pl. TCRT5000-M vonalkövető szenzormodul) csatlakoztathatunk úgy, hogy a mozgás közben a szenzor a szalag fölött helyezkedjen el. A szenzor kimeneti jele így egy bináris négyszögjel lesz: a jelben a fekete-fehér mezők határainál (0,785 cm-enként) lesznek ugrások. Ezen események bekövetkezésekor az időbélyegző rögzítésével és a pozíció számlálójának léptetésével megvalósítható a hely és az idő együttes mérése.

Ha a mozgás során a mozgás iránya is változik, akkor egy lehetséges megoldás ennek észlelésére, ha két szenzort helyezünk el a szalag fölött úgy, hogy négyszögjelükben körülbelül egy negyed periódusnyi fáziseltolódás legyen, és így ún. kvadratúra jel jelenjen meg a kimenetükön (3. ábra). Hasonló elvű enkóderszalaggal ellátott sínt és szenzorokkal, kontrollerrel felszerelt kiskocsit tartalmazó taneszköz [37] kapható ugyan a kereskedelmi forgalomban [38], de az általunk bemutatott összeállítás töredékáron megvalósítható.



3. ábra: A TCRT5000-M modulokból és a BBC micro:bitből álló áramkör. A szalag beosztásához képest így elhelyezett modulok negyed periódus fáziskülönbséggel detektálnak négyszögjelet

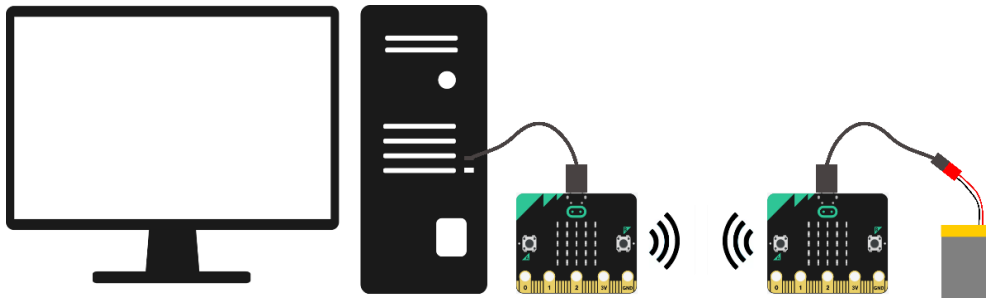
2.3. A vezeték nélküli adattovábbítás megvalósítása a micro:bit és a számítógép között

Mivel a légpárnás sínen vezeték nélküli mérőkísérletet végzünk el, természetesen adódik az igény, hogy egy egyszerű beágyazott rendszert megvalósítva gondoskodjunk a tárggyal (lovassal) együtt mozgó mikrokontroller által gyűjtött adatok tárolásáról vagy számítógépre történő elküldéséről, továbbá az eszköz tápellátásáról. Két micro:bit rádiós kommunikációjára alapozva mobilizálható lehet a mozgó micro:bit áramköre, és a valós idejű adatküldéssel [31,39,40]

[3] A taneszközként kapható légpárnás sínek helyett készülhet elérhető árú barkácsváltozat [36]. Kísérleteinket az SZTE Gyakorló Gimnáziumában, a SzeReTeD labor kézzel készült légpárnás sínén tudtuk tesztelni.

A micro:bit kártya vezérlését a Nordic Semiconductors márkájú NRF52833 (BLE) végzi [41], amely egyúttal beépített rádiófrekvenciás modulként a vezeték nélküli kommunikációt is lehetővé teszi az eszköz számára. A lovason elhelyezett micro:bitet adóként vezeték nélküli mérőrendszerként, míg a másikat fogadóként USB-n keresztül a PC-hez csatlakoztatva megvalósíthatjuk a vezeték nélküli adattovábbítást a mérőeszköz és a számítógép között [42].

A légpárnás sínen történő kísérletezéshez az adó tápellátását nem célszerű a nagy tömegű, 2 db sorba kötött, egyenként 1,5V-os AAA-s elemmel megvalósítani, míg a Thomson CR2032-es 3V-os gombelem kis árama miatt a rendszerünk (a szenzorok működése és a kommunikáció is) rendkívül bizonytalanul működött. A kis méretű, 190 mAh-s Li-Po tölthető (és így környezettudatosabb megoldást kínáló) akkumulátor (3,7-4,2V) az áramkör specifikációja alapján közvetlenül nem köthető be a tápcsatlakozóba. Azonban a mini USB-csatlakozó megfelelő feszültségregulátorral van ellátva, azon keresztül beköthető az akkumulátor tápfeszültsége, esetünkben egy megbontott mini USB-kábelre tüskesor forrasztásával kapott JST-USB adapter segítségével, így stabil és könnyű áramforrást biztosíthatunk a tárggyal együtt mozgó micro:bit számára. A kommunikációt és a tápellátást biztosító eszközök blokkvázlata a 4. ábrán látható.



4. ábra: Vezeték nélküli adattovábbítás blokkvázlata

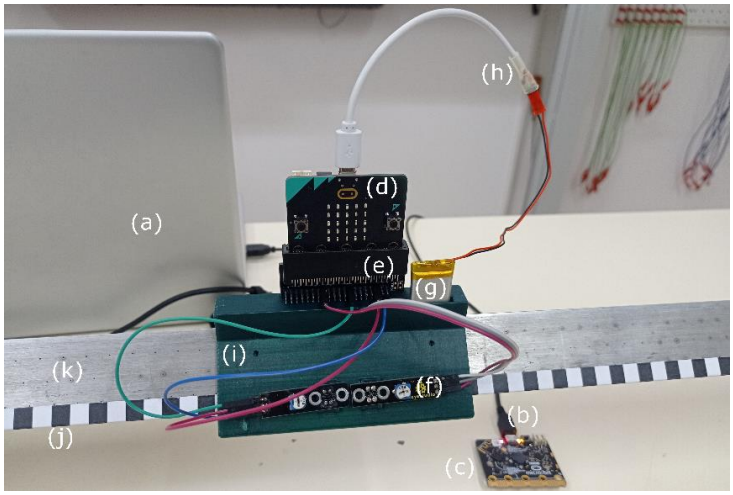
2.4. Légpárnás sínen mozgó 3D-nyomtatott lovas a micro:bit és kiegészítői számára

A 3D-nyomtatott eszközök egyre elterjedtebbek a természettudományos oktatásban is, többek között a fizikaszertárak is gyarapodhatnak saját tervezésű eszközökkel [43]. Barkácsolás helyett tervezhetünk (pl. Tinkercadben vagy Fusion 360-ban) olyan egyedi lovast, amely alkalmas a fenti áramkör stabil hordozására, azaz megfelelő helyet biztosít a micro:bit, az akkumulátora, és a szenzorok számára. Utóbbiak esetén különösen figyelni kell arra, hogy egyrészt az enkóderszalag fölött helyezkedjenek el, másrészt olyan távol legyenek a szenzorok, hogy ténylegesen kvadratúra jelet kapjunk.

Tervezésnél érdemes még néhány praktikus szempontot figyelembe venni. A légellenállást érdemes minél inkább csökkenteni, törekedni az áramvonalasságra, ezért érdemes lekerekíteni az éleket. A sínből kiáramló levegő el kell tudja emelni a tárgyat a sín falától, így a tömegét korlátozni, másrészt a megfelelő tömegeloszlást biztosítani kell, azaz ki kell tudni egyensúlyozni a lovast. A stabilitáshoz elég hosszúra érdemes tervezni a sínről lelógó oldalfalakat. Mivel a szenzorokat egy oldalra helyezve a lovasunk könnyen kibillenhet, ami az egyik oldalon megakadályozhatja a levegőáramlást, ezért a falakban tanácsos elhelyezni olyan belső furatokat, amelyekbe apró ólomgolyókat dobva kiegyensúlyozható a tárgy. A lovas gerince alatt található belső élnél továbbá érdemes egy kis (kb. 1 mm átmérőjű) furatot készíteni a levegőáramlás biztosítására, amivel jelentősen csökkenthető a

súrlódás. A diákjainkkal szakkörön Fusion 360-ban közösen tervezett és gondosan kikísérletezett lovas 3D-tervét mellékeltük.[4]

A teljes kísérleti elrendezés, amellyel az egyenes vonalú mozgások pozíció-idő függése vizsgálható, az 5. ábrán látható.



5. ábra: A légpárnás sínen történő mozgások pozíció-idő méréséhez összeállított kísérleti elrendezés: a) számítógép, b) USB-kábel, c) vezetéken csatlakozó micro:bit (vevő), d) vezeték nélküli micro:bit (adó) e) breakout board, f) reflexiós infravörös adó-vevő (A TCRT5000-M), g) Li-Po akkumulátor h) mini USB-átalakító, i) 3D-nyomtatott lovas j) enkóderszalag k) légpárnás sín

3. Szoftverfejlesztés a BBC micro:bitre épülő kinematikai kísérletek megvalósításához

A micro:bittel történő mérőkísérletekre jóval kevesebb példát látunk a hazai és nemzetközi szakirodalomban, fórumokon, szemben az Arduinóval. Ezért fontosnak tartottuk, hogy a kísérletekhez fejlesztett szoftver felépítését részletesen ismertessük, és az egyes kísérletek igényeinek megfelelően egyre komplexebbé váló programkódokat is mellékeljük. Az adatfogadásra számos ingyenes szoftveres lehetőség elérhető, mindazonáltal bemutatjuk az általunk fejlesztett Excel-bővítmény jelenlegi verziójának hasznos funkcióit.

3.1. A micro:bit programozása adatgyűjtőként

A 2. szakaszban ismertettük a kísérleti elrendezés felépítését, amelyből az is kiderült, hogy tulajdonképp időben változó szélességű négyszögjelek feldolgozása szükséges, hogy időt és pozíciót tudjunk meghatározni.

3.1.1. A micro:bit alapú mérőszoftver programozása különböző környezetekben

A micro:biteket a MakeCode felületen blokkprogramozással [44], illetve JavaScriptben [45] vagy Pythonban (MicroPython)[13] lehet programozni. Talán kevésbé köztudott hogy Arduino (C/C++)

[4] Mivel az enkóderszalagot mindenképp úgy kell felragasztani a saját vagy saját készítésű légpárnás sínünkre, hogy a levegőáramlást ne zavarják, így a modellünk esetleges felhasználásánál mindenképp módosítani kell az oldalfalak hosszát, a szenzorok helyét.

nyelven is elérhetők jól kidolgozott függvénykönyvtárak [46]. Mind a MakeCode, mind pedig a MicroPython C++ nyelven írt, DAL/CODAL(C++) absztrakciós rétegre épülnek, ami pedig az NRF5 SDK-ra [47,48]. Általában elmondható, hogy minél alacsonyabb szintű nyelven tudjuk programozni az eszközt, annál nagyobb időbeli pontosság érhető el [49]. A jelenlegi kísérletekhez a mérőszoftvert a micro:bit V2-t támogató CODAL-ban fejlesztettük C++ nyelven [50].[5]

3.1.2. Eseménykezelés és feladatütemezés CODAL-ban C++ nyelven

A CODAL C++ kódjában a MakeCode blokkprogramozásából ismerős Indításkor() és Állandóan() funkciói megjelennek a main() függvényben: az első felében egyszer lefuttatandó, beállítás jellegű parancsokat helyezünk el, pl. inicializáljuk a micro:bit osztály „uBit" elnevezésű példányát, míg a második részében egy végtelen ciklus helyezkedik el, amelyben az ismételt feladatokat végezhetjük el.

A CODAL-ban egyfajta többszálúságot valósíthatunk meg egyrészt az ütemezője révén, amely több feladat összehangolt működését teszi lehetővé, másrészt az üzenetbusz nevű eseménykezelő rendszer révén, amely lehetővé teszi reaktív kód írását. A listen() függvény meghívásával figyelhetünk egy konkrét hardverblokkban bekövetkező eseményt (pl. valamelyik gomb lenyomását vagy rádióüzenet érkezését), a mi kísérleteinkben főként az egyik digitális bemeneten (`MICROBIT_ID_IO_P2`) a digitális jelben történő egységugrásokat (`MICROBIT_PIN_EVT_FALL` vagy `MICROBIT_PIN_EVT_RISE`). Az üzenetbusz tájékoztatja a programot, és az ütemező a feladatok közé rendeli az általunk megírt eseménykezelő függvényt (`onFallingEdgeA` vagy `onRisingEdgeA`). Az ütemező beállítása nélkül minden esemény egyforma prioritással rendelkezik, azonban az ütemező meghívása után kiemelhetünk időkritikus eseményeket az `MESSAGE_BUS_LISTENER_IMMEDIATE` kulcsszóval.

Az esemény figyelését megvalósító kódrészlet:

```
uBit.messageBus.listen(MICROBIT_ID_IO_P2,      MICROBIT_PIN_EVT_FALL,
onFallingEdgeA, MESSAGE_BUS_LISTENER_IMMEDIATE)
```

3.1.3. Idő mérése az élvezérelt mérésben

Bár a CODAL is tartalmaz megvalósított timert, amellyel egyenlő időközönként mintavételezhetnénk a digitális jelet, csupán azok a pillanatok adnak értékes információt a test helyéről, amikor a digitális jelben ugrás történik. Ha ebben a pillanatban az eseménykezelő függvényünk rögzíti az időbélyegzőt, akkor a mozgás egy nagyon jól meghatározott eseményéhez (fekete-fehér mező közötti váltás) rendelünk hozzá időpontot mikroszekundum pontosan[6].

Az esemény kezelését megvalósító kódrészlet:

```
void onRisingEdgeA(MicroBitEvent evt)
{
    timeStampTemporary=system_timer_current_time_us();
}
```

Mivel az eseménykezelő függvényt érdemes amennyire csak lehet, rövid idő alatt végrehajtani (hiszen az ütemező időkritikusan kezeli), ezért a további adatfeldolgozás, továbbítás a main() függvény végtelen ciklusában fog zajlani. Érdemes az időbélyegzőt (64bites egész) volatile típusúnak deklarálni, hogy biztonságosan lehessen átadni a végtelen ciklusnak az értékét. A végtelen ciklus minden iterációjában megvizsgáljuk, hogy a jelenleg rendelkezésre álló időbélyegző és az előző ese-

[5] Telepítési útmutatót és mintakódokat találhatunk a Lancester University GitHub oldalán [51].

[6] Lehetőség van a milliszekundum pontos mérésre is.

mény bekövetkezésekor eltárolt időbélyegző eltér-e, mert ekkor tudhatjuk, hogy új esemény következett be. Itt persze további műveleteket végezhetünk a mért időadaton.[7] Akár soros porton, akár rádiós kommunikációban küldjük tovább az adatot a számítógépnek, a CODAL saját típusát kell használni, az ún. ManagedString()-et. Mivel a típuskonverziót a 64bites integerre közvetlenül nem valósították meg a környezetben, ezért a mikroszekundumban mért időt másodpercbe váltva tizedes törtként (az egészrészt és a törtrészt külön kezelve) konvertáltuk ManagedString típusúvá.

Az így megírt egyszerű kód már alkalmas arra, hogy például transzmissziós fotokapu segítségével ingamozgás vagy forgómozgás periódusidejét megmérjük. A mellékelt kódban (01) csak a felszálló ágra történik eseménykezelés, így a fényúton való áthaladáshoz egyetlen időbélyegzőt rendelünk.

3.1.4. Egyirányú mozgás pozíciójának mérése élek számlálásával

A reflexiós szenzor jelének feldolgozásakor az időbélyegző rögzítése mellett az eseményhez pozícióértéket is szeretnénk rendelni. Mivel az enkóderszalagon a fehér és fekete csíkok szélessége megegyezik, így célszerű mind a felszálló és leszálló ág eseményénél időpontot rögzíteni, és ezeket felváltva, ugyanazzal a feldolgozó rutinnal kezelhetjük. Egyirányú mozgás esetén minden új él a helykoordináta 0,785 cm-rel történő növekedését jelenti, ezért a végtelen ciklusban az időbélyegző feldolgozása és elküldése mellett a pozíció számlálójának inkrementálása is megtörténik. A mellékelt kód (02) a fent leírtak alapján alkalmas egyenes vonalú egyenletes és egyenletesen változó mozgás vizsgálatára a 2. szakaszban bemutatott légpárnás sínhez tervezett kísérleti elrendezésben.

3.1.5. Váltakozó irányú mozgás pozíciójának mérése kvadratúra jel vizsgálatával

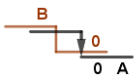
A váltakozó irányú mozgás aktuális pozícióját és a mozgás aktuális irányát a két szenzor által mért körülbelül ¼ periódussal eltolt négyszögjelek 4-féle állapotából olvashatjuk ki (1. táblázat). Amennyiben tudnánk biztosítani megfelelő geometriai pontossággal, hogy a jelek között pontosan ¼ periódus eltolás van, úgy mindkét szenzor (A és B) jelének mindkét típusú (felszálló és leszálló) éle által vezérelt eseményben lehetne időbélyegzőt rögzíteni, amelyek között eltelt időben biztosan lehetne tudni, hogy a pozíció pozitív vagy negatív irányban 0,3925 cm-rel változott. A 3D nyomtatás pontatlansága és a szenzormodulok mérete ilyen mértékű pontosságot nem tesz lehetővé.

Ezért kompromisszumként csak az A jel éleinél rögzítünk időbélyegzőt, közben megtett távolság pontossága (0,785 cm) a fekete-fehér csíkok nyomtatásától függ csupán. Az eseménykezelő függvényekben az időbélyegző rögzítésén túl mind az A, mind a B digitális jel aktuális értékét rögzíthetjük.

Az esemény kezelését megvalósító kódrészlet:

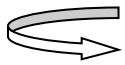
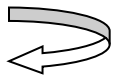
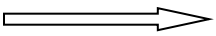
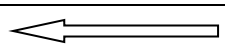
```
void onFallingEdgeA(MicroBitEvent evt)
{
    timeStampTemporary=system_timer_current_time_us();
    stateOfSensorATemporary=uBit.io.P2.getDigitalValue();
    stateOfSensorBTemporary=uBit.io.P8.getDigitalValue();
}
```

[7] Az itt vázolt algoritmus megtalálható egy korábbi munkánkban Arduino nyelven, megszakítást kezelő rutinnal megvalósítva [27].

Esemény az A jel ugrásakor	Él típusa	Esemény után az A jel értéke	Esemény után az B jel értéke	Mozgásirány és a hozzárendelt logikai érték
	Felszálló	1	1	A-t követi B→1
	Leszálló	0	0	A-t követi B→1
	Felszálló	1	0	B-t követi A →0
	Leszálló	0	1	B-t követi A→0

1. táblázat: a kvadratúra jelben bekövetkező ugrások, és a jel kiértékelésének logikai vázlata.

Az 1. táblázat alapján látható, hogy amennyiben az élek után a mért digitális értékek megegyezzenek, úgy az A jelet a B követi, ellenkező esetben éppen fordított a mozgásirány. A kódunkban a mozgásirányt egy bináris értékkel rögzítjük. A 2. táblázatban szemléltetjük a pozitív vagy negatív irányú pozícióléptetés feltételrendszerét. Mindössze az előző eseménynél (felszálló vagy leszálló élnél) rögzített mozgásiránnyal kell összevetni az aktuális eseményben megállapított mozgásirányt, és amennyiben változott, a megfelelő irányban léptetnünk kell a számlálót.

Mozgásszakasz	Mozgásirány az előző eseménynél	Mozgásirány a mostani eseménynél	A pozícióérték léptetése
	B-t követi A →0	A-t követi B→1	Nem változik
	A-t követi B→1	B-t követi A →0	Nem változik
	A-t követi B→1	A-t követi B→1	Pozíció növelése
	B-t követi A →0	B-t követi A →0	Pozíció csökkentése

2. táblázat: A pozíciószámlálás léptetésének logikai vázlata

3.1.6. Rádiós kommunikáció a két micro:bit között

A két micro:bit közötti rádiókapcsolatot a DAL/CODAL dokumentációja alapján [52] és a fejlesztők által biztosított adó-, illetve vevőoldali mintakódok [51] alapján valósítottuk meg.

Az adó szerepét betöltő micro:bit egyszer lefutó parancsai között engedélyezzük a rádiókapcsolatot (`uBit.radio.enable()`), majd a végtelen ciklusában az időbélyegző típuskonverziója és a pozíció megfelelő léptetése után az adatokat egymástól vesszővel elválasztva, sortöréssel ellátva ManagedString típusú üzenként elküldjük a vevőnek `(uBit.radio.datagram.send(message)` . A mellékelt mérőszoftver (03) a micro:biten futtatva alkalmas a légpárnás sínen oda-vissza mozgó lovas pozíció-idő értékpárok felvételére, és azok rádiókapcsolaton keresztüli továbbítására.

Mellékeltük a fogadó oldal kódját is (04), melyben a másik micro:biten szintén engedélyezzük a rádiós kapcsolatot, és a listen() függvény ezúttal a rádiós üzenet érkezésének eseményét figyeli.

3.1.7. Soros kommunikáció a micro:bit és a PC között

Adatérkezéskor egy pufferbe töltődik az adat, amelyet ManagedString típusú üzenetté konvertálva azt a soros porton keresztül a PC-re küldhetjük `(uBit.serial.send(message))`. Megjegyezzük, hogy a kommunikáció baud rate-je ebben az esetben 115200 bit/s [53].

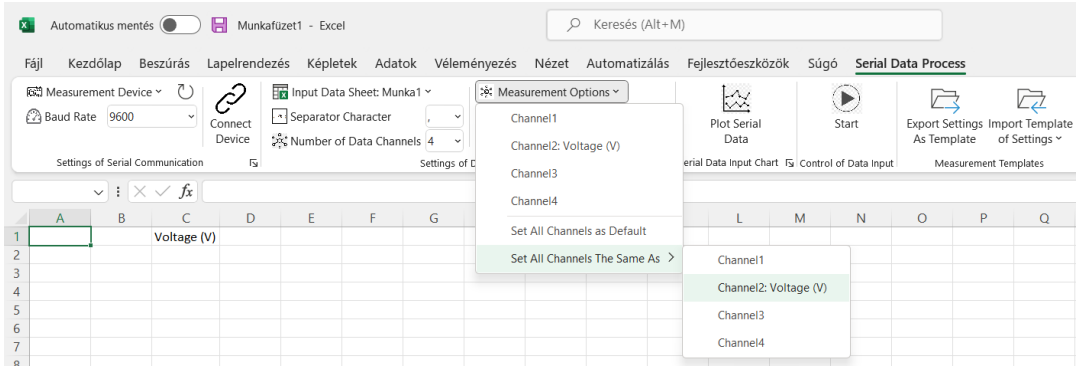
3.2. Adatfogadás PC-n

Az adatok számítógépen történő fogadásának többféle módja van, ezek közül a legegyszerűbb ha egy fejlesztőkörnyezet (Arduino IDE, Visual Studio Code) soros monitorát használjuk. Egy népszerű megoldás egy táblázatkezelő-bővítményt, például a Microsoft Data Streamer bővítményben [54] fogadni a soros porton érkező adatokat, mivel ott lehetőség van az adatok azonnali, helyben történő kiértékelésére. Korábbi munkáink során többször előfordult, hogy a Data Streamer bizonyos korlátai (pl. maximált a sorok száma) miatt Excel-makrót fejlesztettünk, hogy a bejövő adatokat megfelelően tudjuk feldolgozni [27,28]. A Visual Studio Tools for Office lehetőséget biztosít arra, hogy olyan bővítményt fejlesszünk, amely a Data Streamerhez hasonló adatfogadáson túl több funkcióval, és kevesebb korláttal rendelkezik, így a kísérleteink minél szélesebb körében tudjuk alkalmazni.

3.2.1. Excel-bővítmény fejlesztése VSTO eszközkészlettel C# nyelven

Az Microsoft többféle lehetőséget biztosít a fejlesztőknek, Excel-bővítmény létrehozására. Az egyik legelterjedtebb és legrégebbi opció a Visual Basic nyelven történő makrófejlesztés. A Visual Studio Tools for Office (VSTO) Windows operációs rendszer alapú offline bővítmények fejlesztési lehetőségét biztosítja a Visual Studio környezetben, .NET Framework keretrendszerben C# és Visual Basic nyelven egyaránt. Egy újabb lehetőség az online, platformfüggetlen Office Add-in típusú bővítmények JavaScript/TypeScript nyelven történő programozása.

Az adatfogadó bővítményünket a VSTO eszköztár segítségével C# nyelven fejlesztettük. A bővítmény front end oldala az Excelbe épülő Ribbon (szalag), amely tartalmazza az általunk megadott gombokat, dinamikus és statikus menüket, lenyíló listákat (combobox), szöveges mezőket, stb. A Ribbon tervezésére létezik ugyan vizuális tervező, de a haladó szintű funkciók eléréséhez (pl. az Excel beépített gombjainak átemelése vagy azok feladatainak felülírása) az XML leíró nyelvet használtuk. A backend oldalon C# nyelven a Ribbon classban adjuk meg azokat az eseménykezelő (callback) függvényeket, amelyeket a különböző gomblenyomások (pl. onAction, onPressed típusú események), menükezelés során fogunk meghívni. A bővítmény rugalmasságát a gombok, menük dinamikus kezelése teszi lehetővé (pl. getPressed, getLabel, getItemLabel stb.). A .NET keretrendszer lehetővé teszik, hogy például részletesebb beállítási opciókhoz felugró WPF-ablakokat készítsünk, vagy éppen figyelmeztető vagy hibaüzeneteket közvetítő WinForms párbeszédablakokat hozzunk létre.



7. ábra: A Serial Data Process bővítmény előlapja a szalagon (Ribbon)

3.2.2. Adatok fogadása soros portról

A .NET Framework komplett eszköztárat (SerialPort Class) biztosít a fejlesztett alkalmazások soros kommunikációjához [55]. Bővítményünkben a szalagon beállítható a BaudRate értéke egy lenyíló listából kiválasztott szabványos értékként vagy egy tetszőlegesen beírt értékként is. A szalag betöltésekor az alkalmazás a PC eszközkezelőjéből lekérdezi a rendelkezésre álló eszközök nevét, COM-kiosztását, és egy dinamikus menübe rendezi a felhasználó számára, aki így kiválaszthatja mérőeszközét. Egy eszköz későbbi csatlakoztatása során frissíthetjük ezt a dinamikus listát. A haladóbb beállítások (pl. Parity, Handshake, stb. a menü dialogButtonjára kattintva megjelenő WPF-ablakban jelennek meg.

A Connect gombra kattintva inicializálódik a SerialPort objektum, a felhasználó által megadott tulajdonságokkal, a Start gomb lenyomásakor pedig a DataReceived típusú eseménykezelőt engedélyezzük, amely elindítás után a soros portról érkező adatokat az Excel mezőibe írja soronként, mennyiségenként.

3.2.3. Az adatfogadás összehangolása a mérőszoftverrel

A következő blokkban olyan gombokat, jelölőnégyzeteket és menüket helyeztünk el, amelyek azt segítik elő, hogy a felhasználó össze tudja hangolni az általa írt mérőszoftver és az adatfogadó bővítmény beállításait. A már korábban említett BaudRate (esetünkben 115200 bit/s) összehangolásán túl megadhatjuk, hogy soronként hány darab, egymástól milyen karakterrel elválasztott adat érkezik, sőt azt is, hogy melyik mennyiség milyen típusú, mi a mértékegysége stb. A sorokon belül fogadott adatok (channel) számát nem korlátoztuk, a dinamikus menüben lehetőség van ezek egységes beállítására is (7. ábra), felugró WPF-ablakokban. A felhasználó által megadott beállításokat (pl. a légpárnás sínen végzett kísérletekben Channel1 másodpercben megadott időt, míg a Channel2 egységben megadott pozíciót tárol) a WPF-ablak bezárása után egy .xml fájlban tárolja el a program, és a program a későbbi működés során (pl. adatoszlopok fejlécének kitöltése, grafikus megjelenítéskor kiválasztható mennyiségek) ebből a konfigurációs fájlból olvassuk ki a felhasználó által megadott beállításokat. Ha a mérőszoftverben nem mérünk időt, az adatfogadás időpontjának rögzítését engedélyezhetjük a Get Software TimeStamp jelölőnégyzet segítségével, amelyet a szoftver később úgy kezel, mint a soros portról érkező adatok bármelyikét. Előfordulhat, hogy meg kell szakítanunk a mérést, és nem szeretnénk, ha az előző méréseink eredménye törlődne (mint pl. a DataStreamerben), ezért kipipálhatjuk a Start with appending Data jelölőnégyzetet, így a program a kommunikáció újraindításakor megvizsgálja az utolsó adattal rendelkező sor helyét, és onnan folytatja az adatok feltöltését.

3.2.4. Valós idejű grafikus megjelenítés

Az adatok soros porton történő fogadása lehetővé teszi, hogy egyfajta közel élő adatmegjelenítést valósítsunk meg. Erre a ScottPlot WPF eszköztárat [56] alkalmaztuk. Az alapelv egyfajta író-olvasó adattovábbítás megvalósítása, amelyet cirkuláris pufferrel valósítottunk meg. A DataReceived eseménykezelő által meghívott callback függvény írja a cirkuláris puffer tartalmát, míg a grafikus adatmegjelenítő olvassa azt, akár időkéséssel is.

3.2.5. Beállítások eltárolása sablonként

Egy kifejezetten „tanárbarát” funkciót építettünk be a bővítménybe, a beállítások sablonként történő elmentésének lehetőségét. Minden mikrokontroller, amit alkalmazunk, más és más kommunikációs beállításokat igényel, a bejövő üzenetek struktúrája, a feldolgozott fizikai mennyiségek típusa és mértékegysége más és más. Ugyanakkor ha egy kísérletnél megtaláltuk a legideálisabb beállításokat, célszerű azokat eltárolni, hogy az oktatás rohamtempójában az adott kísérlethez előre összeállított berendezés csatlakoztatásakor néhány kattintással visszaállíthatók legyenek korábbi beállításaink. Az Export Settings as Template gombra kattintva a bővítmény a program futása alatt a háttérben író-

dott .xml fájlt más néven menti el, míg az Import Settings as Template dinamikus menüben a bővítmény lekérdezi a háttérmappa tartalmát, feldolgozza a sablonfájlok címét, helyreállítva a felhasználó által eredetileg megadott (különleges karaktereket is tartalmazó) fájlneveket, és a menüben a megfelelő fájlnévre kattintáskor behívja a sablonban elmentett beállításokat.

3.4. A kinematikai kísérletek eredményeinek kiértékelése Excelben

A kísérleteket szakköri kereten belül légpárnás sínen realizáltuk, eredményeink nagyon ígéretesek, jó egyezést mutatnak a bevezetőben felsorolt mozgások hely-idő összefüggéseivel.

Az időmérés pontosságát mutatja, hogy 90 cm hosszú inga 65 teljes lengés (130 mért lengésidő) esetén átlagosan 1,8935 s-nek adódott, $6{,}72 \cdot 10^{-4}$ s szórással. (Ebből az értékből a nehézségi gyorsulás értéke 9,91 $\frac{m}{s^2}$-nek adódik.)

3.4.1. A légpárnás sínen végzett kísérletek mérési adatainak kiértékelése Excelben a Trendvonal segítségével

A pozíció-idő adatpárokat XY-grafikonokon jelenítettük meg Excelben. A Trendvonal elnevezésű opció lehetőséget kínál lineáris, logaritmikus, exponenciális, polinomiális, hatványfüggvény görbék illesztésére vagy mozgóátlagos simítás alkalmazására. Ezenkívül a görbék egyenletei és a statisztikai R^2 értékek is megjeleníthetők a grafikonon. A vizsgált mozgások közül az egyenes vonalú egyenletes és az egyenletesen változó mozgás hely-idő függése a Trendvonal segítségével elemezhető.

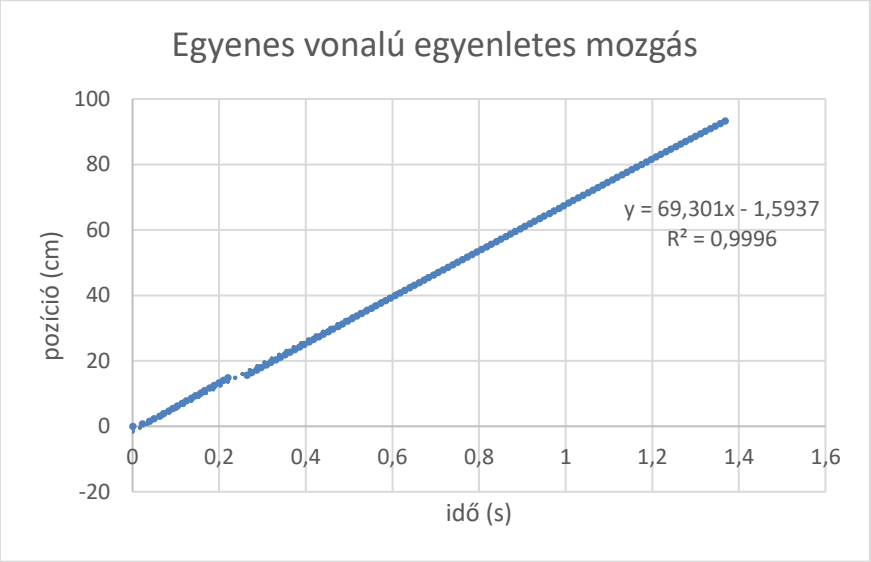
3.4.2. A mérési adatok kiértékelése az Excel beépített statisztikai függvényei segítségével

Az Excel biztosít néhány olyan, a legkisebb négyzetek módszerén alapuló függvényt, amelyek kimeneti értékei felhasználhatók a mozgások jellemzőinek meghatározására, mint például az egyenletes egyenes vonalú mozgás kezdőpozíciója, sebessége, az egyenletesen gyorsított mozgás kezdőpozíciója, kezdősebessége, gyorsulása [57,58]. A kiértékelés megkezdése előtt tanácsos a cellatartományokat dinamikusan elnevezni [59] (Képletek →Név megadása), pl. **x** a pozícióértékek, **t** az időbélyegértékek esetén. Ezután ezekre a Név jelölésű cellatartományokra hivatkozni lehet az Excel képletekben, ahogy az a 3. táblázatban látható.

Mozgás típusa	Képlet	Paraméter meghatározása az Excel függvényeivel	
Egyenes vonalú egyenletes mozgás	$\mathbf{x}=v\mathbf{t}+x_0$	v	MEREDEKSÉG(**x**;**t**)
		x_0	METSZ(**x**;**t**)
Egyenes vonalú egyenletesen változó mozgás	$\mathbf{x}=\frac{1}{2}a\mathbf{t}^2+v_0\mathbf{t}+x_0$	a	2*INDEX(LIN.ILL(**y**,x^{1,2}),1,1)
		v_0	INDEX(LIN.ILL(**y**,**x**^{1,2}),1,2)
		x_0	INDEX(LIN.ILL(**y**,**x**^{1,2}),1,3)

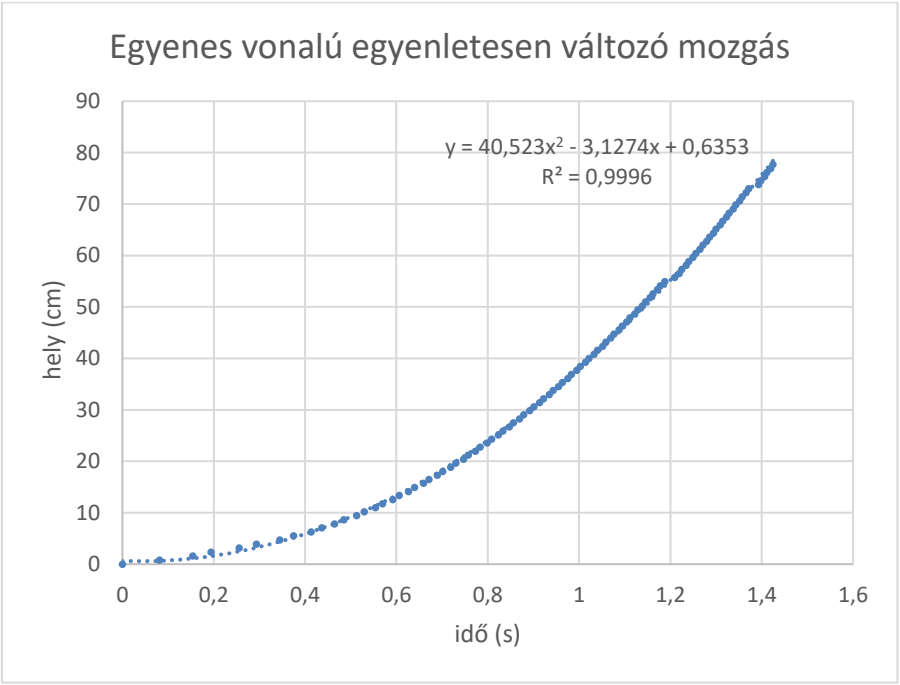
3. táblázat: Paraméterek, egyenletek és képletek az egyenes vonalú egyenletes és egyenletesen változó mozgások pozíció-idő adatainak elemzéséhez. A vastag betűk a megnevezett adattartományokra utalnak

Az illesztett lineáris függvény paramétereinek kinyeréséhez egyszerűen a MEREDEKSÉG() és METSZ() Excel-függvények használhatók az egyenletes mozgások paramétereinek kiértékelésére.



6. ábra: A légpárnás sínen egyenes vonalú egyenletes mozgást végző lovas pozíciója az idő függvényében

A légpárnás sín végére egy csigát rögzítettünk, átvetettünk rajta egy fonalat, amelynek egyik végét a lovashoz, másik végét pedig a nehezékhez rögzítettük. A 6. ábra szemlélteti, hogy a négyzetes úttörvénynek megfelelő eredményt kaptunk. A másodfokú függvény illesztésére a LIN.ILL() Excel-függvényt alkalmazhatjuk, amely eredményül a regresszió összes paraméterét és statisztikai jellemzőjét tartalmazó táblát generál. Ebből az adatsorból az INDEX() Excel-függvény segítségével kiválaszthatjuk az illesztett másodfokú polinomfüggvény megfelelő rendű tagjainak együtthatóit [57].



7. ábra: Az egyenes vonalú egyenletesen változó mozgást végző lovas pozíciója az idő függvényében.

Mindkét mérési eredménynél látható, hogy ha valamilyen kommunikációs/érzékelési hiba miatt egy-egy élen nem történik eseménykezelés és mérés, a pozíció léptetése késve következik be. Ezért további vizsgálatok szükségesek, amelyekkel megmutatható, hogy ezzel a mérési módszerrel mekkora sebességig tudunk mozgásokat megfelelő pontossággal vizsgálni légpárnás sínen, és hogyan fejleszthető a kommunikációs protokoll, hogy csökkentsük a kimaradt üzenetek előfordulásának valószínűségét.

3.6. Nemlineáris regresszió (legkisebb négyzetek módszere) az Excel Solver bővítményével

A kísérleti összeállításunkban két viszonylag laza spirálrugó közé fogattuk be a lovast, és kis mértékben kitérítettük az egyensúlyi helyzetéből. A kísérlet nem vezetett csillapítatlan rezgésre, viszonylag hamar csillapodott a rezgés. Ha a harmonikus rezgést sebességgel arányos erő csillapítja, a kitérése (pozíciója) az idő függvényében $\boldsymbol{y(t)} = \mathrm{a} \cdot e^{-bt} \cdot \sin\left(\frac{2\pi}{\mathrm{T}} \cdot \mathbf{t} + \varphi_0\right) + \mathrm{y}_0$ összefüggéssel adható meg [24]. Ilyen típusú függvényt a Trendvonal segítségével, illetve a LIN.ILL() beépített függvénnyel nem tudunk illeszteni. Az elmúlt években olyan útmutatók jelentek meg oktatási szaklapokban, amelyek lépésről lépésre mutatják be, hogyan lehet Excelben a legkisebb négyzetek regresszióit alkalmazni a nemlineáris görbék illesztésére [60].

Először az a b, T, φ0 és y0 paramétereket becsültük meg a mérési adatok segítségével, amennyire csak lehetséges volt. Megjegyezzük, hogy ez a becslési folyamat nagyon hasznos lehet a tanulók számára, mert segít megérteni az egyes paraméterek fizikai jelentését. Másodszor, ezekkel a paraméterekkel kiszámoltuk a fenti y(t) formulával a becsült értékeket, valamint a mért értéktől való eltérésüket. Ezután az eltérések négyzetösszegét meghatároztuk, és az Excel Solver bővítményével minimalizáltuk, így a becsült paraméterek optimalizált értékeit kaptuk. Ezeket a paramétereket felhasz-

nálva egy egyenletes, sűrű beosztású idősoron az y(t) formulával a trendvonalhoz hasonló görbeillesztés valósítható meg.

8. ábra: A csillapodó rezgőmozgás grafikonja

4. Összefoglalás

A tanulmányban körbejártunk egy hardveres és szoftveres összeállítást, amely a BBC micro:bit taneszközre építve egy komplex mérőrendszerként funkcionálhat a fizikaoktatás számára. A bemutatott kísérleti elrendezésben fontos szerepet kapott a 3D-nyomtatás. A mérőszoftver fejlesztésénél prezentáltuk a CODAL (C++) környezetben rejlő lehetőségeket. A kiértékelésnél a joggal népszerű Excel-funkciók mellett egy saját fejlesztésű Excel-bővítmény működésének alapjait ismertettük. Kísérleti eredményeink alátámasztják, hogy a megoldásaink helytállóak.

Úgy véljük, a fent bemutatott eszközök, megoldások az informatikatanári közösségnek is hasznosak lehetnek: fontos, hogy építsük a tantárgyak közötti kapcsolatokat, hiszen ezen tudományterületek oktatása együtt tud hozzájárulni ahhoz, hogy a diákok műszaki pályát válasszanak.

Köszönetnyilvánítás

A Kulturális és Innovációs Minisztérium ÚNKP-22-3 - SZTE-452 kódszámú Új Nemzeti Kiválóság Programjának a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból finanszírozott szakmai támogatásával készült.

A szerzők hálásan köszönik dr. Piláth Károly értékes hozzászólásait és előremutató kritikáit. A szerzők köszönetüket fejezik ki a Szegedi Tudományegetem SzeReTeD laboratóriumának vezetőjének, Csiszár Imrének, aki rendelkezésünkre bocsátotta a légpárnás sínt és a megfelelő helyszínt a mérésekhez. Köszönettel tartozunk továbbá dr. Kalmár György kollégánknak, aki számos hasznos ötlettel és jótanáccsal látott el a projekt alatt. Megköszönjük a Szegedi Radnóti Miklós Kísérleti Gimnázium Számítógépes Fizika szakkörös diákjainak, hogy a 3D-tervezésben, a kísérletek kivitelezésében aktívan segédkeztek.

Irodalom

1. S. Straulino, *Reconstruction of Galileo Galilei's experiment: the inclined plane*. Physics Education, (2008) 43. 316–321. http://www.fyysika.ee/vorgustik/wp-content/uploads/2011/11/Reconstruction-of-Galileo-Galilei.pdf (Utoljára megtekintve: 2021. 7. 27.).
2. Simonyi K., A fizika kultúrtörténete: A kezdetektől a huszadik század végéig, Akadémiai Kiadó, 2011.
3. Juhász A., Tasnádi P., Jenei P., Illy J., Wiener C., Főzy I., *A fizika tanítása a középiskolában I.* http://fiztan.phd.elte.hu/letolt/fizika_tanitasa_1.pdf.
4. *Kerettanterv a gimnáziumok 9–12. évfolyama számára*. https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat/kerettanterv_gimn_9_12_evf (Utoljára megtekintve: 2020. 6. 29.).
5. Y. Rogers, V. Shum, N. Marquardt, S. Lechelt, R. Johnson, H. Baker, M. Davies, *From the BBC micro to micro:bit and beyond: a British innovation*. Interactions, (2017) 24. 74–77. https://discovery.ucl.ac.uk/id/eprint/1546735/1/Rogers-Y_BBC%20Micro%20to%20micro%20paper_2017.pdf (Utoljára megtekintve: 2022. 11. 12.).
6. J. Austin, H. Baker, T. Ball, J. Devine, J. Finney, P. De Halleux, S. Hodges, M. Moskal, G. Stockdale, *The BBC micro:bit: from the U.K. to the world*. Communications of the ACM, (2020) 63. 62–69. https://doi.org/10.1145/3368856.
7. C. Andrews, *BBC micro:bit - a little bit too late? [IT education]*. Engineering & Technology, (2016) 11. 30–33. https://doi.org/10.1049/et.2016.0400.
8. S. Sentance, J. Waite, S. Hodges, E. MacLeod, L. Yeomans, *"Creating Cool Stuff": Pupils' Experience of the BBC micro:bit*, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, ACM, Seattle Washington USA, 2017 531–536. https://dl.acm.org/doi/10.1145/3017680.3017749 (Utoljára megtekintve: 2022. 11. 13.).
9. *Digitális kultúra 5*. Oktatási Hivatal, (2020) 7–24. https://www.tankonyvkatalogus.hu/pdf/OH-DIG05TA__teljes.pdf (Utoljára megtekintve: 2022. 11. 12.).
10. *Digitális kultúra 6*. Oktatási Hivatal, (2021) 47-51., 54-60. https://www.tankonyvkatalogus.hu/pdf/OH-DIG06TA__teljes.pdf (Utoljára megtekintve: 2022. 11. 12.).
11. *Digitális kultúra 7*. Oktatási Hivatal, (2022) 54-60. https://www.tankonyvkatalogus.hu/pdf/OH-DIG07TA__teljes.pdf (Utoljára megtekintve: 2022. 11. 12.).
12. *Microsoft MakeCode for micro:bit*. Microsoft MakeCode for Micro:Bit. https://makecode.microbit.org/ (Utoljára megtekintve: 2022. 11. 13.).
13. *micro:bit Python Editor*. https://python.microbit.org/v/3 (Utoljára megtekintve: 2022. 11. 13.).
14. *Segédanyagok – micro:bit botorkálás*. http://microbit.inf.elte.hu/segedanyagok/ (Utoljára megtekintve: 2022. 11. 12.).
15. A. Abonyi-Tóth, *Programozzunk micro:biteket!*. ELTE Informatikai Kar, (2017). http://microbit.inf.elte.hu/wp-content/uploads/2018/05/Programozzunk-microbiteket-2018.pdf (Utoljára megtekintve: 2022. 11. 12.).

16. MISZAK.
http://www.inf.u-szeged.hu/miszak/ (Utoljára megtekintve: 2019. 11. 4.).

17. Gingl Z., Mellár J., Szepe T., Makan G., Mingesz R., Vadai G., Kopasz K., *Universal Arduino-based experimenting system to support teaching of natural sciences*. Journal of Physics: Conference Series, (2019) 1287. 012052.
https://doi.org/10.1088/1742-6596/1287/1/012052.

18. Makan G., Antal D., Mingesz R., Gingl Z., Mellár J., Vadai G., Kopasz K., *Interdisciplinary Technical Exercises for Informatics Teacher Students*. Central-European Journal of New Technologies in Research, Education and Practice, (2019) 1. 21–34.
http://ojs.elte.hu/cejntrep/article/view/382 (Utoljára megtekintve: 2020. 4. 22.).

19. Gingl Z., Makan G., Mellar J., Vadai G., Mingesz R., *Phonocardiography and Photoplethysmography With Simple Arduino Setups to Support Interdisciplinary STEM Education*. IEEE Access, (2019) 7. 88970–88985.
https://ieeexplore.ieee.org/document/8754669/ (Utoljára megtekintve: 2019. 11. 4.).

20. *Arduino alkalmazása a fizika és a digitális kultúra oktatásában | MISZAK*.
http://www.inf.u-szeged.hu/miszak/arduino-alkalmazasa-a-fizika-es-az-informatika-oktatasaban/ (Utoljára megtekintve: 2020. 11. 2.).

21. Piláth K., STEM receptek fizikatanároknak, ELTE Fizika Doktori Iskola, Budapest, 2021.

22. Korom E., Radnóti K., Gondolkodtató természettudomány-tanítás: Fizika | MTA-SZTE Természettudomány Tanítása Kutatócsoport, Szeged: Mozaik Kiadó, 2020.
http://edu.u-szeged.hu/ttkcs/sites/default/files/konyvek/MS-9403_MTA_Fizika_online.pdf (Utoljára megtekintve: 2022. 11. 4.).

23. Teiermayer A., *Improving students' skills in physics and computer science using BBC Micro:bit*. Physics Education, (2019) 54. 065021.
http://fiztan.phd.elte.hu/english/student/bbc.pdf (Utoljára megtekintve: 2022. 11. 12.).

24. R.G. Rinaldi, A. Fauzi, *A complete damped harmonic oscillator using an Arduino and an Excel spreadsheet*. Physics Education, (2019) 55. 015024.
https://doi.org/10.1088/1361-6552/ab539d.

25. D. Nichols, *Arduino-Based Data Acquisition into Excel, LabVIEW, and MATLAB*. The Physics Teacher, (2017) 55. 226–227.
https://doi.org/10.1119/1.4978720.

26. Piláth K., *Adatgyűjtés Excelben – Fizika kísérletek*. https://pilath.wordpress.com/adatgyujtes-excelben/ (Utoljára megtekintve: 2019. 11. 5.).

27. Somogyi A., Kelemen A., Mellár J., Mingesz R., *Investigation of the Rotational Motion of the Fidget Spinner by Means of Technical Informatics*. Central-European Journal of New Technologies in Research, Education and Practice, (2021).
http://ojs.elte.hu/cejntrep/article/view/515 (Utoljára megtekintve: 2021. 3. 15.).

28. Somogyi A., Kelemen A., Mingesz R., *Motion tracking by an Arduino Due and Excel*, in: Proceedings of XXXIII. DidMatTech 2020 Conference. New Methods and Technologies in Education, Research and Practice, Eötvös Loránd University in Budapest, Trnava University in Trnava, .

29. *Kitronik Edge Connector Breakout Board for BBC micro:bit - Pre-built*. Kitronik Ltd.
https://kitronik.co.uk/products/5601b-edge-connector-breakout-board-for-bbc-microbit-pre-built (Utoljára megtekintve: 2022. 11. 20.).

30. P. Bernad, D. Šic, R. Repnik, D. Osrajnik, *Development of Measurement Systems with the BBC Micro:bit*, in: 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), 2021 853–858. https://doi.org/10.23919/MIPRO52101.2021.9596834.
31. Piláth, *Micro:bit Alapú Fénykapu Építése*. Fizika kísérletek, (2019). https://pilath.wordpress.com/microbit-alapu-fenykapu-epitese/ (Utoljára megtekintve: 2021. 8. 22.).
32. Somogyi A., Kelemen A., Mingesz R., *Low-cost high-resolution measurements of periodic motions with Arduino in physics teacher in-service education*. Journal of Physics: Conference Series, (2022) 2297. 012031. https://doi.org/10.1088/1742-6596/2297/1/012031.
33. Z. Gingl, R. Mingesz, J. Mellár, B. Lupsic, K. Kopasz, *Efficient Sound Card Based Experimention At Different Levels Of Natural Science Education*. ArXiv Preprint ArXiv:1108.1109, (2011).
34. *What is Rotary Encoder? Types, Principle, working in detail*. Circuit Schools. https://www.circuitschools.com/what-is-rotary-encoder-types-principle-working-in-detail/ (Utoljára megtekintve: 2022. 11. 20.).
35. *Basics of Rotary Encoders: Overview and New Technologies*. Machine Design, (2014). https://www.machinedesign.com/automation-iiot/sensors/article/21831757/basics-of-rotary-encoders-overview-and-new-technologies (Utoljára megtekintve: 2020. 6. 14.).
36. T.B. Greenslade, *An inexpensive air track*. The Physics Teacher, (1986) 24. 231–231.
37. *Dynamics Cart and Track System with Motion Encoder - Vernier*. https://www.vernier.com/product/dynamics-cart-and-track-system-with-motion-encoder/ (Utoljára megtekintve: 2022. 11. 20.).
38. *Reviews - IOPscience*. https://iopscience.iop.org/article/10.1088/0031-9120/51/6/066001/pdf (Utoljára megtekintve: 2022. 11. 20.).
39. *Micro:bitek Beszélgetnek*. Fizika kísérletek, (2019). https://pilath.wordpress.com/microbitek-beszelgetnek/ (Utoljára megtekintve: 2022. 11. 20.).
40. *Vezetéknélküli fizika*. Fizika kísérletek, (2020). https://pilath.wordpress.com/vezeteknelkuli-fizika/ (Utoljára megtekintve: 2022. 11. 20.).
41. *Hardware Overview - micro:bit V2*. https://tech.microbit.org/hardware/#nrf52-application-processor (Utoljára megtekintve: 2022. 11. 20.).
42. *Remote data collection*. Microsoft MakeCode. https://makecode.microbit.org/device/data-analysis/remote (Utoljára megtekintve: 2022. 11. 20.).
43. S. Ford, T. Minshall, *Invited review article: Where and how 3D printing is used in teaching and education*. Additive Manufacturing, (2019) 25. 131–150. https://doi.org/10.1016/j.addma.2018.10.028.
44. T. Ball, J. Protzenko, J. Bishop, M. Moskal, J. de Halleux, M. Braun, S. Hodges, C. Riley, *Microsoft touch develop and the BBC micro:bit*, in: Proceedings of the 38th International Conference on Software Engineering Companion, ACM, Austin Texas, 2016 637–640. https://doi.org/10.1145/2889160.2889179.
45. *JavaScript*. Microsoft MakeCode. https://makecode.microbit.org/javascript (Utoljára megtekintve: 2022. 11. 20.).

46. *Micro:bit with Arduino*. Adafruit Learning System. https://learn.adafruit.com/use-micro-bit-with-arduino/overview (Utoljára megtekintve: 2022. 11. 20.).
47. J. Devine, J. Finney, P. de Halleux, M. Moskal, T. Ball, S. Hodges, *MakeCode and CODAL: Intuitive and efficient embedded systems programming for education*. Journal of Systems Architecture, (2019) 98. 468–483. https://linkinghub.elsevier.com/retrieve/pii/S1383762118306088 (Utoljára megtekintve: 2022. 11. 13.).
48. *The micro:bit runtime DAL/CODAL*. https://tech.microbit.org/software/runtime/ (Utoljára megtekintve: 2022. 11. 20.).
49. *micro:bit (nRF51, Cortex-M0) GPIO toggling*. https://metebalci.com/blog/microbit-gpio-toggling/ (Utoljára megtekintve: 2022. 11. 20.).
50. *CODAL target for the micro:bit v2.x series of devices*. (2022). https://github.com/lancaster-university/codal-microbit-v2 (Utoljára megtekintve: 2022. 11. 20.).
51. *lancaster-university/microbit-v2-samples: CODAL build tools and sample programs for the micro:bit v2.x*. https://github.com/lancaster-university/microbit-v2-samples (Utoljára megtekintve: 2022. 11. 20.).
52. *radio - micro:bit runtime*. https://lancaster-university.github.io/microbit-docs/ubit/radio/ (Utoljára megtekintve: 2022. 11. 20.).
53. *serial - micro:bit runtime*. https://lancaster-university.github.io/microbit-docs/ubit/serial/ (Utoljára megtekintve: 2022. 11. 20.).
54. *Microsoft Data Streamer Add-in for Excel - Excel Data Streamer*. https://learn.microsoft.com/en-us/microsoft-365/education/data-streamer/ (Utoljára megtekintve: 2022. 11. 20.).
55. *SerialPort Class (System.IO.Ports) | Microsoft Learn*. https://learn.microsoft.com/en-us/dotnet/api/system.io.ports.serialport?view=dotnet-plat-ext-7.0 (Utoljára megtekintve: 2022. 11. 22.).
56. *ScottPlot*. https://ScottPlot.NET (Utoljára megtekintve: 2022. 11. 22.).
57. P. Peterlin, *Data analysis and graphing in an introductory physics laboratory: spreadsheet versus statistics suite*. European Journal of Physics, (2010) 31. 919–931. https://doi.org/10.1088/0143-0807/31/4/021.
58. *Guidelines and examples of array formulas*. https://support.microsoft.com/en-us/office/guidelines-and-examples-of-array-formulas-7d94a64e-3ff3-4686-9372-ecfd5caa57c7 (Utoljára megtekintve: 2021. 8. 15.).
59. *Excel Dynamic Named Ranges • My Online Training Hub*. My Online Training Hub, (2013). https://www.myonlinetraininghub.com/excel-dynamic-named-ranges (Utoljára megtekintve: 2021. 8. 28.).
60. A.M. Brown, *A step-by-step guide to non-linear regression analysis of experimental data using a Microsoft Excel spreadsheet*. Computer Methods and Programs in Biomedicine, (2001) 65. 191–200. https://doi.org/10.1016/S0169-2607(00)00124-3.