Sprego avatárok a 3D térben

Dienes Nikolett[1], Gulácsi Ádám[2], Csernoch Mária[3]

[1]dienesniki@gmail.com, [2]adam.gulacsi@gmail.com,
[3]csernoch.maria@inf.unideb.hu
DEBRECENI EGYETEM, INFORMATIKAI KAR

Absztrakt. A kompetencia-alapú mérőeszközök egyértelműen bizonyítják, hogy a táblázatkezelés oktatása során jelenleg alkalmazott módszerek nem elég hatékonyak. Ezek a hagyományosnak tekinthető táblázatkezelési megközelítések az egyes szoftverek speciális funkcióira, jellemzőire, az eszközhasználatra fókuszálnak ahelyett, hogy az algoritmikus, számítógépes vagy absztrakt gondolkodás fejlesztését, valamint a számítógépes problémamegoldást helyeznék előtérbe.

Munkánk során egy olyan semi-unplugged eszközt hoztunk létre, amely programozási alaptételek Sprego-implementációját teszi elérhetővé 3D térben, demó és interaktív módban. A program alapkoncepciója szerint avatárokkal játszatjuk el az algoritmust, mellyel párhuzamosan futtatjuk a szöveges képletkiértékelőt. Ezzel célunk, hogy növeljük a diákok lelkesedését és hatékony támogatást tudjunk nyújtani a számítógépes gondolkodás, mint alapkészség fejlesztéséhez.

Kulcsszavak: Sprego, táblázatkezelés, összetett függvények, programozás, algoritmikus gondolkodás, 3D oktatószoftver

1. Motiváció

Célunk olyan módszertani megközelítések, eszközök fejlesztése, amelyekkel növelhető az informatikaoktatás hatékonysága, a tanulók számítógépes gondolkodásának, ezen belül is kiemelten az algoritmikus készség és a számítógépes problémamegoldás fejlesztése. Fontosnak tartjuk a koncepció alapú problémamegoldást [1], valamint a sémaépítést [2][3], amely technikák nagyban segíthetik a tanulói gyors és lassú gondolkodás megfelelő és megbízható alkalmazását, elősegítve ezzel a hatékony és problémamentes megoldások létrehozását, azok helyességének ellenőrzését [4][5][6].

Ezen felül növelni szeretnénk a diákok motivációját látványos és modern reprezentációkkal. Fontos szempont volt továbbá, hogy a való életből vett példákkal operáljunk, hiszen így a tanulók jobban bele tudják élni magukat a megteremtett helyzetekbe. Hosszú távú célunk pedig az, hogy egy olyan alkalmazást implementáljunk, amely egy egyedülálló szerepet tölthet be az informatika oktatásában. Ez a semi-unplugged megoldás ugyanis alkalmassá válhat arra, hogy egy hidat képezzen az unplugged (például ha matrjoska babákkal szemléltetjük az összetett függvényeket) [7][8] és a plugged-in (például ha táblázatkezelő feladatokat oldanak meg a tanulók a számítógépeknél) módszerek között.

2. Elméleti és gyakorlati háttér

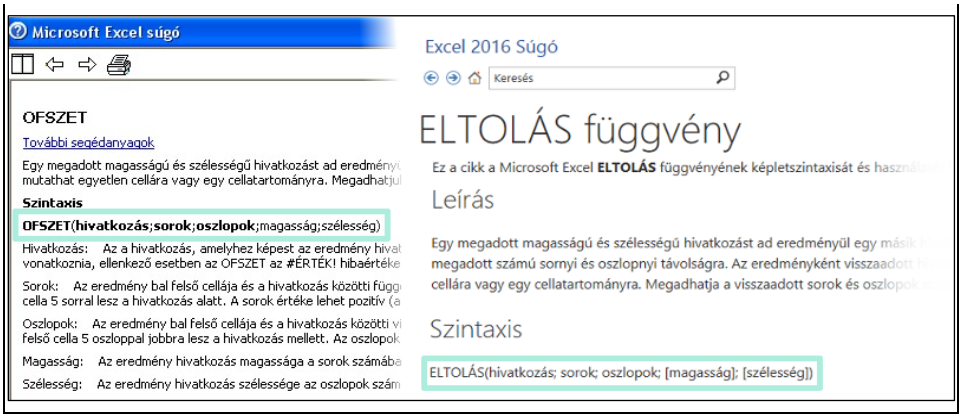
Kutatásunk során egy olyan 3D semi-unplugged szemléltetőeszközt hoztunk létre, amely várakozásainknak megfelelően hatékonyan fogja segíteni a diákok tanulási folyamatát a Sprego keretrendszerben, amely eltér a fentebb említett hagyományos megközelítéstől [9]. A szoftver jelenlegi verziójában két algoritmus vizualizációja érhető el, interaktív és demó módban. Az egyik egy erdei környezetben játszódó feltételes számlálás, a másik pedig egy városi környezetbe helyezett lineáris keresés.

2.1. Elméleti háttér – Sprego

Alkalmazásunk bemutatása előtt érdemes részletesen kifejtenünk azt, hogy mi is a Sprego módszer. Ez egy új, alternatív megközelítés, amely egy algoritmusalapú keretrendszert biztosít a táblázatkeze-

léssel kapcsolatos valódi problémamegoldáshoz [9]. A név a **Spre**adsheet és a Le**go** szavak kombinációjából alakult ki. Ahhoz, hogy megértsük a Sprego alapelveit, először részletesen meg kell vizsgálnunk a tradicionális megközelítésű táblázatkezelés buktatóit és nehézségeit.

Az első ilyen hátrány magából a táblázatkezelő szoftvercsaládokból ered, mivel a különböző programcsaládok csak a legelemibb szinten kompatibilisek. Továbbá egy programcsaládon belül is előfordulnak kompatibilitási problémák az egyes verziók között. Megtörténhet ugyanis az, hogy a táblázatkezelő függvényeket kicserélik vagy átnevezik. Erre egy példa az OFFSET() függvény átnevezése a Microsoft Excel különböző verzióiban (1. ábra).



1. ábra: Az OFFSET() függvény neve, szintaxisa és működése a Microsoft Excel 2003 (bal) és 2016 verziójában (jobb).

A második ilyen probléma az argumentumok sorrendjének következetlensége a hasonló célú függvények között. Ezt szemlélteti például az, hogy a SZUMHA() függvény argumentumai teljesen más sorrendben következnek, mint a SZUMHATÖBB() függvényé, ami növeli a szemantikailag inkorrekt formulák építésének kockázatát. [9][10].

A következő nehézség a mértéktelen mennyiségű probléma-specifikus függvényből ered. Ezek memorizálása a speciális funkciójukkal és argumentumaikkal együtt nagyon megterhelő feladat nemcsak a diákok, hanem a tapasztaltabb felhasználók számára is. Gyakori kérdés táblázatkezelői környezetben a lineáris keresés, ahol egy konkrét értéket egy vektorban, vagy egy mátrixban keresünk. A felhasználó találhat számos megoldást a problémájára, KERES(), FKERES(), VKERES() függvényeket, vagy az INDEX(HOL.VAN()) összetett függvényt alkalmazva [11][12][13][14][15]. Az első három függvénynek azonban vannak olyan limitációi, amelyek tovább korlátozzák a használatukat az összetett függvénnyel szemben (1. táblázat).

Függvény(ek)	Keresési vektor	Keresési vektor rendezettsége
KERES()	sorban és oszlopban	növekvő
VKERES()	csak sorban	növekvő / nem rendezett
FKERES()	csak oszlopban	növekvő / nem rendezett
INDEX(HOL.VAN())	sorban és oszlopban	növekvő (1, alapértelmezett) / nem rendezett (0) / csökkenő (-1)

1. táblázat: A KERES(), VKERES(), FKERES() és az INDEX(HOL.VAN()) függvények összehasonlítása.

A túl sok speciális függvény alkalmazása és tanítása nemcsak szükségtelen, hanem kevésbé hatékony is, ugyanis egy átlagos felhasználó körülbelül 12 függvényt használ normál problémamegoldó

keretek között [16]. Ezen felül nem elhanyagolható tény az sem, hogy leghatékonyabban úgy fejlesztheti a diákok számítógépes gondolkodását, ha fokozatosan építjük fel az instrukciók halmazát [17].

A Sprego egy olyan alternatíva, ami sikeres megoldást jelent a korábban említett nehézségekre. A módszer alapelve, hogy csupán 12 alapvető, általános célú táblázatkezelő függvényre van szükségünk, amelyeket mint Lego elemeket összekombinálva tudunk problémákat megoldani [9]. A 12 függvény további három csoportba sorolható, csoportonként 4-4 függvénnyel. Ez a rendszerezés lehetőséget biztosít egyfajta szemantikai és tudásszint szerinti csoportosításra (2. táblázat). Fontos megjegyeznünk, hogy ez a 12 függvényt tartalmazó halmaz nyitott, műveltségi területtől és a problémától függően bővíthető. Az egyetlen megkötés, hogy egyszerű, könnyen értelmezhető függvényeket használhatunk a bővítéshez, mint például: KICSI(), NAGY(), MEDIÁN() [9].

Sprego szöveg	Sprego szám	Sprego pro
BAL()	MIN()	HA()
JOBB()	MAX()	INDEX()
HOSSZ()	SZUM()	HOL.VAN()
SZÖVEG.KERES()	ÁTLAG()	HIBÁS()

2. táblázat: A 12 alapvető Sprego függvény és azok csoportosítása.

A 12 függvény megtalálható az összes táblázatkezelő szoftverben már a legelső verzióktól kezdve úgy, hogy az elnevezésük és az argumentumaik is konzisztensek az első megjelenésük óta. A Sprego felhasználónak mély megértéssel kell rendelkeznie ezen függvényeknek a működésével kapcsolatban szintaktikai és szemantikai szempontból is, hogy feladatmegoldáshoz tudja használni azokat. Ez azonban sokkal kevésbé nehéz feladat a mértéktelen mennyiségű probléma-specifikus függvény memorizálásához képest [18].

A Sprego módszer alkalmazásakor elengedhetetlen az összetett függvények és a tömbképletek használata. Az összetett függvények használatának egyik legfőbb előnye, hogy fejleszti a diákok algoritmikus és számítógépes gondolkodását, valamint tágítja a matematikai tudásukat a függvények, többváltozós függvények, tömbök, azaz n dimenziós vektorok témaköreiben [6][9].

Az eljárás másik nagy előnye, hogy új lehetőségeket nyit az unplugged eszközökkel való tanításban [7][8]. Erre egy szemléletes példa a matrjoska babák használata az osztálytermekben. A babák ugyanis tökéletesen demonstrálják az összetett függvények működését és azt, hogy ez a fajta problémamegoldás mennyire hasonlít a szöveg-alapú magas szintű programozási nyelvekhez (2. ábra).

2. ábra: Unplugged tanítási módszerek, matrjoska baba.

Megállapítható továbbá az is, hogy a módszer tökéletesen alkalmas kisiskolás korban az alkalmazói szoftverek és a programozás témakör lefedésére is, hiszen fejleszti az algoritmikus készséget és a számítógépes gondolkodást. Ezek az átfedések azért bírnak nagy jelentőséggel, mert így megvalósíthatóvá válik a kerettantervben sokszor megemlített tudástranszfer. Ezáltal a gyerekek könnyebben tudnak befogadni egy-egy új témakört, valamint kevesebb nehézséggel abszolválják a felmerülő problémákat, feladatokat. További előnye a módszernek, hogy egyetlen programozási eszközzel a kerettanterv két nagy témaköre – táblázatkezelés és programozás – is lefedhető általános iskolában és a középiskolai bevezető szakaszban, valamint teret enged a további informatikai témakörök közötti tudástranszfer megvalósításához [19].

2.2. Sprego algoritmusok 2D-s vizualizációja

Szoftverünk tervezése során egy korábbi kutatás eredményeként elkészült alkalmazásból indultunk ki [20]. A kutatócsoport egy több platformot is támogató, 2D vizualizációs applikációt készített el a különböző Sprego algoritmusok reprezentálásának céljából. Az alkalmazás két gyakran felmerülő probléma algoritmusainak ábrázolását implementálta, az egyik a feltételes számlálás, a {=SZUM(HA())} tömbképlettel, a másik pedig a lineáris keresés, az =INDEX(HOL.VAN()) összetett függvénnyel.

Az animációk fő célja, hogy a diákok jobban megértsék az algoritmusok és a Sprego függvények működését, amelyet az algoritmusok műveleteinek lépésről lépésre történő bemutatásával tettek szemléletessé.

A szoftver a való életből hozott példákkal operál, ahol az avatárok szerepét színes matrjoska babák töltik be. Az oktatásban történő felhasználás szempontjából fontos, hogy az alkalmazás a multi-platform elérhetőségnek köszönhetően támogatja az Android, Windows, Macintosh és Linux felületeken történő lejátszást is [20][21][22].

3. Kutatásunk

A legmeghatározóbb feladatunk az volt, hogy a már létező 2D-s vizualizáció koncepcióját átültessük 3D környezetbe. Ez a transzformációs folyamat két fő részre osztotta a munkafolyamatot: az első rész a grafikai feladatok kivitelezése, ami magában foglalja a 3D modellek, textúrák és UI elemek tervezését és elkészítését az új környezet igényeinek megfelelően, a második részt pedig a programozási feladatok teljesítése képezi.

A munkafolyamat megtervezésekor fontosnak tartottuk, hogy megtartsuk a 2D applikáció fő aspektusait, például a matrjoska babákat, a való életből vett példákat és a multi-platform elérhetőséget. Ezen felül új funkciókkal is bővítettük alkalmazásunkat kihasználva a 3D adta lehetőségeket (3. ábra).

3. ábra: A 2D (bal) és a 3D (jobb) környezet összehasonlítása.

3.1. Fejlesztői környezetünk

Szoftverünket a Unity3D fejlesztőkörnyezetben implementáltuk [23]. A Unity3D egy játékfejlesztő engine, amely használható 2D és 3D applikációk létrehozásához is.

Ebben a környezetben a fejlesztéshez két fő eszköztár áll rendelkezésre: a beépített eszközök, valamint a programozó API, amelyen keresztül a projektben megtalálható objektumok működését tudjuk módosítani C# szkriptek segítségével. A beépített eszközök sokszínűsége és széleskörű felhasználhatósága nagyban megkönnyítette a kezdeti prototípusok előállítását. A C# szkriptekkel tetszőlegesen testreszabható az objektumok viselkedése a rendszerben. Ez a kettős eszközrendszer biztosította az optimális munkafolyamatot a projekt fejlesztési ideje alatt.

Fontos érv volt még a Unity3D engine mellett, hogy non-profit használat esetén teljesen ingyenesen alkalmazható. Ezen felül elmondható, hogy egy nagyon jól dokumentált rendszerről van szó [24]. Rendkívül népszerű, segítőkész és aktív fejlesztői közösséggel rendelkezik [25]. Ezek mind hasznos segítségnek bizonyultak a fejlesztés folyamán, és nagyban megkönnyítette a tanulási folyamatot, valamint a felmerülő problémák, bugok kiküszöbölését és megoldását.

3.2. Grafikai feladatok

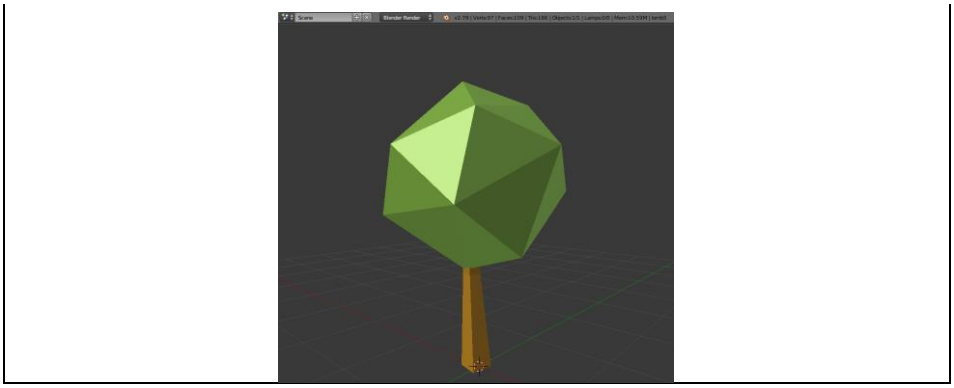
A fejlesztési munkálatok egyik része a grafikai kivitelezés volt. A 3D modellezéshez két különböző szoftvert használtunk fel: a MagicaVoxel-t kizárólag a városi jelenetben megtalálható házak építéséhez [26], az összes többi modell létrehozásához pedig a Blender szoftvert tartottuk a legalkalmasabbnak [27]. Továbbá szükség volt a projektben 2D grafikai elemekre is, ami a 3D modellek textúráit, illetve a UI elemeket jelenti. Ezeknek az elkészítéséhez a GIMP programot választottuk ki [28]. Általánosan elmondható mindegyikről, hogy ingyenes, megbízható, népszerű és jól dokumentált, ami lényegesen megkönnyítette a használatukat.

A 3D modellezési munkák első lépése egy absztrakt környezeti modell létrehozása volt. Ez azt jelentette, hogy szét kellett bontani a már adott 2D környezetet atomi elemekre. Az absztrakció eredményeként előállt egy egyértelmű lista azokról a 3D modellekről, amelyeket egyesével el kellett készíteni. Ez a lista egy rendkívül hatékony módja volt annak, hogy a modellezéssel kapcsolatos feladatainkat és a munkafolyamatot előre megtervezzük (3. táblázat).

Egy 3D alkalmazást nagyon sokféle különböző grafikai ábrázolási móddal el lehet készíteni. A mi szoftverünk esetében a low poly stílust találtuk a legmegfelelőbbnek több különböző okból kifolyólag is.

A hardver limitációja az egyik sarkalatos pont. Sok iskolában nem állnak rendelkezésre a legkorszerűbb hardverrel rendelkező számítógépek és egyéb IKT eszközök, így ezt mindenképpen figyelembe kellett vennünk. További hardverrel kapcsolatos korlátokat jelent az is, hogy a szoftvernek a korábbi terveinknek megfelelően mobil platformokon is elérhetőnek kell lennie. A low poly kétségtelenül egy hatékony stílus ebből a szempontból. Ugyanis az ilyen módon elkészített 3D modellek egy részletgazdag 3D modellhez képest jelentősen kevesebb poligonból épülnek fel. Kevesebb sokszög kevesebb számítást igényel a grafikus kártya részéről, ami hatékonyabb, gyorsabb végrehajtást és alacsonyabb rendszer követelményt igényel magától az applikációtól.

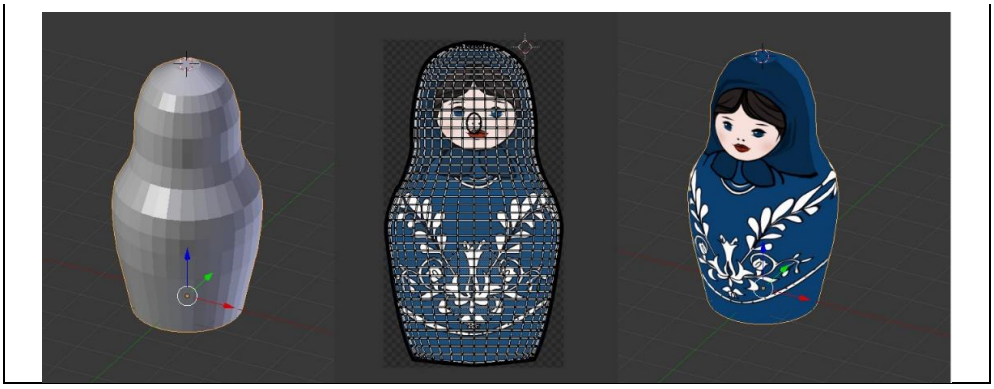
A második fontos a szempont a vizualizáció befogadásával kapcsolatos. Arra törekedtünk, hogy a videojátékok világából ismert játékosság érzetét alakítsunk ki a felhasználókban annak érdekében, hogy a Sprego tanulási folyamatának hatékonyságát növelni tudjuk. Mivel a low poly stílus mind a két elvárásunknak eleget tett, és ezen felül még rövidebb kivitelezési időt is igényel alternatíváinál, ezért végül a modelljeinket ilyen módon alkottuk meg (4. ábra).



4. ábra: Egy low poly fa 3D modellje Blender-ben.

A matrjoska babák és a település táblák modellezése egy teljesen más típusú munkafolyamatot és technikai megközelítést igényelt, hiszen textúrákat is kellett feszíteni a geometriai alakzatokra. A babák elkészítése komolyabb kihívást jelentett, így ezt a folyamatot fogjuk bemutatni.

A speciális modellek elkészítésének az első lépése egy olyan geometriai alakzat elkészítése volt, ami alkalmas avatárként működni az applikáció keretein belül úgy, hogy közben hasonlít a babák eredeti, a való életben megtalálható kinézetére is. Ezek után a következő feladat az volt, hogy a babák 3D alakzatát kicsomagoljuk (unwrapping) a síkba. Ezt követően meg kellett rajzolni az összes különböző színű textúrát GIMP-ben. A fényes, részletgazdag és színes textúrák biztosították, hogy az avatárok kellőképpen kitűnjenek majd a környezetükből az alkalmazásban, ezzel is biztosítva a minél jobb felhasználói élményt a tanulók számára. Az alakzatok kicsomagolása és a textúrák elkészítése után össze kellett rendelni a síkra kifeszített formát és a textúrákat, hogy azok megfelelően jelenjenek meg. Ezt a bemutatott módszert UV mapping-nek nevezik (5. ábra).



5. ábra: Textúra ráfeszítése egy geometriai alakzatra UV mapping módszerrel.

3.3. Programozási feladatok

A programozási feladatokat a két különböző környezet (jelenet) felépítésével kezdtük, amihez először az elkészített 3D modelleket kellet beimportálni a projektbe. Az építési folyamat atomi elemek, úgynevezett Game Objec-tek hierarchiájának felépítésén keresztül valósult meg. Ezekhez az objektumokhoz lehet hozzárendelni például olyan tulajdonságokat, hogy hol helyezkednek el a világon belül (a koordináta-rendszerben), milyen 3D modell tartozik hozzájuk, hat-e rájuk gravitáció stb.

A környezet kezdeti felépítése után egy finomhangolási folyamat következett, amely alatt számos beállítást megváltoztattunk a jelenetre vonatkozóan, hogy a végeredmény megfeleljen a megjelenéssel kapcsolatos elképzeléseinknek. Ez többek között a jelenetek világításának és az objektumok árnyékainak megváltoztatását, finomhangolását jelentette.

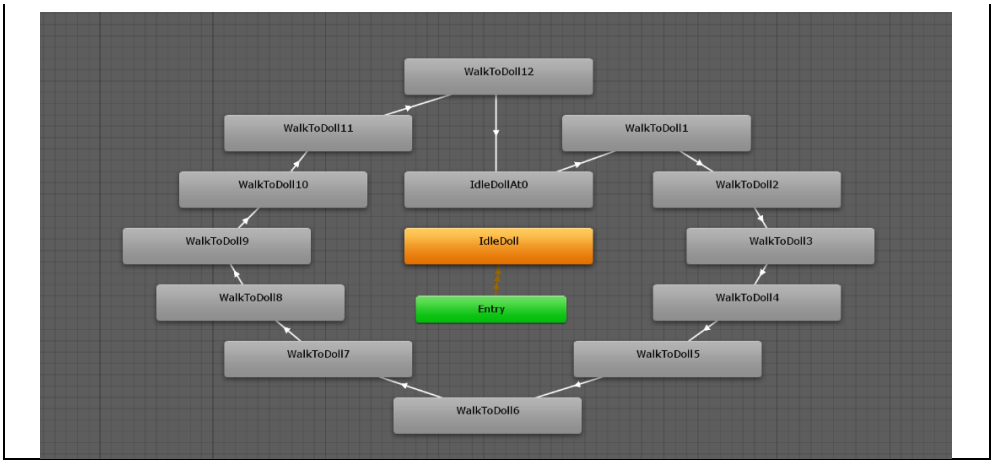
Ezeknek a munkálatoknak a kapcsán fontos megemlíteni a Unity-nek a Prefab nevű eszközét, ami az újrafelhasználható objektumok létrehozásáért felelős. Az ilyen speciális objektumok fő előnye, hogy amikor megváltoztatjuk a Prefab egyik példányának egy-egy attribútumát, akkor eldönthetjük, hogy ezek a változások vonatkozzanak-e az összes többi példányra is vagy sem.

A jelenetek felépítése után az animációk implementálása volt a következő programozási feladat. Az animációval kapcsolatos munkafolyamat szemléltetésére az erdős jelenet (feltételes számlálás algoritmus) elkészítésén keresztül mutatjuk be. Az algoritmus animációja hasonlóképp működik, mint a már említett 2D alkalmazásban [20]. A babák egy erdőben mozognak a tábortűz körül, és az algoritmus megszámolja, hogy a felhasználó által kiválasztott színű babából hány darab van összesen. Az animáció előállításáért felelős folyamat azonban meglehetősen különbözik a két alkalmazásban, a két fejlesztői környezet, a Construct és a Unity3D eltérő természete és célja miatt. Unity-ben két adott eszközre van szükség az animációk implementálásához: animációs fájlokra és a Unity Animator komponensre.

Az animációs fájlok segítségével lehet képkocka alapú animációkat definiálni az adott jelenetben megtalálható objektumokra vonatkozóan. Ez specifikusan ennél a jelenetnél azt jelentette, hogy a 12 babához tartozóan 24 darab animációs fájlt kellett létrehozni, ugyanis a babák a tűz körül egy nagyobb és egy kisebb kör körül is tudnak mozogni.

Ezek után C# szkripteket és a Unity Animator komponenst használtuk arra, hogy irányítani tudjuk az animáció működését. Ezt az Animator komponenst elképzelhetjük úgy, mint egy véges automatát, mert a kettő között hasonlóság fedezhető fel a következő tekintetben:

- az állapotok tükrözik az animált objektum jelenlegi helyzetét (az állapot általában megegyezik egy animációval, amit egy animációs fájlban definiálunk),
- vannak speciális, kitüntetett szerepű állapotok (például: az animáció belépési pontja),
- az (állapot)átmenetek definiálhatók úgy, hogy megadjuk, hogy melyik animációból melyik másik animációba léphet az objektum, és milyen feltételek teljesítésével.



6. ábra: A matrjoska babák animálásáért felelős véges automata egy részlete.

Ezeknek az ismereteknek a tudatában egy olyan véges automatát terveztünk, amely képes irányítani a teljes animációs folyamatot az összes babára vonatkozóan (6. ábra). Ez a tervezési döntés később nagyon kifizetődőnek bizonyult, ugyanis így tökéletesen kontrollálható volt a teljes animáció. Ezen felül a teljes irányítás lehetővé tette azt is, hogy az animáció lejátszását bármikor szüneteltetni és folytatni tudjuk, ami elengedhetetlen volt az interaktív modulok beépítésekor a későbbiekben (7. ábra).

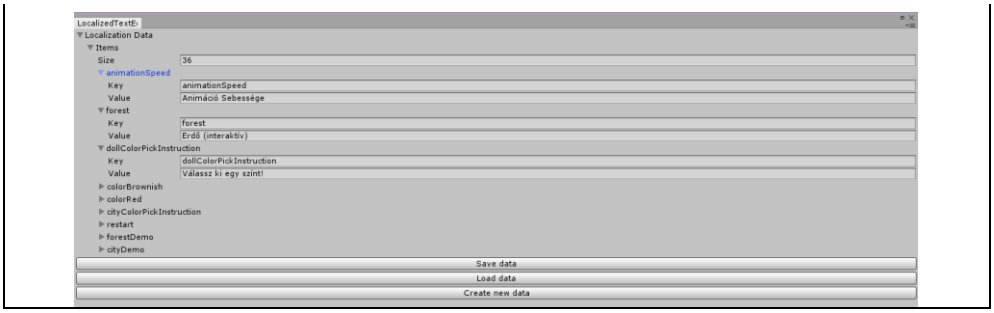
Az animációk előállítása után a fókusz a felhasználó-barát UI összeállítására irányult. A Unity3D egy kiterjeszthető UI rendszerrel rendelkezik, ami magában foglal számtalan előre legyártott ilyen elemet, például gombokat, csúszkákat, paneleket és bemeneti mezőket. Ezeknek nagy előnye, hogy a megjelenésük és a viselkedésük is teljesen testreszabható. Ennek következtében olyan 2D grafikai elemeket hoztunk létre GIMP-ben, amelyek beleillenek a prezentált környezetbe, ezzel is növelve az alkalmazás grafikai vonzerejét a tanulók számára (7. ábra).

A technikai részleteknél oda kellett figyelnünk arra is, hogy a program interaktív elemei és a felhasználói felület kompatibilisek legyenek a különböző bemeneti eszközökkel (egér, érintőképernyő). Továbbá implementáltunk egy extra világítást biztosító funkciót is, amely egy jelölőnégyzettel kapcsolható ki és be. Ennek oka az, hogy számos osztályteremben alkalmatlan körülmények merülhetnek fel a vetítéshez, például ha túl világos van és nincs lehetőség besötétíteni a termet. Ennek a funkciónak a beépítésével biztosítottuk azt, hogy az alkalmazásunk ilyen körülmények között is jól használható legyen az oktatásban.

7. ábra: Az erdős jelenet interaktív felhasználói felülete az alkalmazásban.

Fontos jellemzője a szoftverünknek, hogy a szöveges elemek támogatják a többnyelvű megjelenítést az alkalmazáson belül (az alkalmazás jelenleg a magyar és angol nyelven érthető el). Ehhez implementálni kellett egy lokalizációs rendszert, amelyben elérhetőek a megjeleníthető szövegek az összes támogatott nyelven. A rendszer tervezésénél az elsődleges szempontunk az volt, hogy a bővítés minél egyszerűbben megvalósítható legyen, amennyiben egy új szöveg vagy akár egy teljesen új nyelv bevezetésére kerülne a sor a jövőben. Ehhez a nyelvi szövegeket vezérlő rendszer JSON állományokat használ a kifejezések tárolására. A lefordítható kifejezések kulcs-érték párokként vannak eltárolva a JSON fájlokban, ahol a kulcs azonosítja a kifejezést az alkalmazáson belül, az érték pedig sztringként tárolja a kifejezést az adott nyelven. Annak érdekében, hogy az értékek könnyen szerkeszthetők és hozzáférhetők legyenek, fejlesztettünk egy saját Unity plugint, ami kezeli ezeknek a JSON állományoknak a betöltését, szerkesztését és a mentését (8. ábra).

Ezt követően implementáltunk a kezdőképernyőben egy legördülő listát, amely segítségével ki tudja választani a felhasználó az általa preferált nyelvet. A kiválasztást követően az alkalmazás futási időben betölti a fordításokat a JSON állományból és annak megfelelően egyből megváltoztatja a megjelenített szövegeket.



8. ábra: Többnyelvű kifejezések szerkesztése a saját Unity pluginunkban.

A programozási feladatok végső részét a felhasználói interaktivitás megvalósítása jelentette. Ezt két lépésben tudtuk megtenni. Először ki kellett találnunk azt, hogy hogyan is tudnánk egy olyan interaktivitást biztosítani a felhasználók számára, ami nem befolyásolja az algoritmus animációjának lejátszásának a sikerességét, de mégis érdekesebbé és hatékonyabbá teszi a folyamatot. A második lépésben pedig fel kellett térképeznünk azt, hogy egy ilyen megoldást hogyan tudunk integrálni a meglévő rendszerekbe.

Első lépésként azzal az ötlettel álltunk elő, hogy a felhasználónak interaktív kérdéseket teszünk fel az animáció lejátszása során. Ezeket a kérdéseket párbeszédablakok formájában prezentáljuk a tanulóknak, akik egyszerű nyomógombokkal tudnak válaszolni a kérdésre (7. ábra). Amíg az adott párbeszédablak aktív, addig az animáció lejátszása szünetel, így a diákok is érzik, hogy a program futására befolyással vannak. A nyomógombok azért bizonyultak szerencsés választásnak, mert így a szoftver minden támogatott platformmal megőrizte a kompatibilitását, a bemeneti eszköztől függetlenül. Itt fontos megemlítenünk, hogy le kellett kezelnünk azt az esetet is, amikor a tanuló helytelen választ ad az egyszerű eldöntendő kérdésekre. Ekkor feldobunk egy „Biztos vagy benne?" üzenetet a diáknak, majd visszatérünk az eredeti kérdésre, ahol a felhasználónak újra lehetősége nyílik a kérdés megválaszolására. Ez oktatási környezetben kifejezetten előnyös lehet, ugyanis ilyenkor a tanárnak jut ideje arra, hogy a csoportos coaching módszerével újra átbeszéljék az algoritmus működésének részleteit.

A második lépésben kellett megvizsgálnunk az interaktivitás megvalósításához szükséges technikai hátteret. Ehhez az animáció különböző állapotai közötti átmeneteket kellett megszakítanunk, azáltal hogy az átmenetek közé beillesztettünk egy új fázist. Az új fázis alatt kellett láthatóvá tennünk az aktuális eldöntendő kérdést figyelve arra, hogy az animáció működése mindaddig szüneteljen, amíg a felhasználó nem adott elégséges választ a kérdésünkre. Az implementációhoz az Animator komponenseket, illetve specifikusan erre a célra megírt C# szkripteket használtunk.

4. Összefoglalás

Az általunk megépített alkalmazás célja egy olyan semi-unplugged 3D vizualizációs oktatóprogram elkészítése volt, ami hatékony módon segíti a Sprego programozás tanulási folyamatát.

A Sprego módszer egy alternatív megközelítése a tradicionális táblázatkezelői oktatási folyamatnak. Fő jellemzője, hogy a feladatmegoldáshoz az algoritmusépítésre, az összetett függvények és a tömbképletek használatára támaszkodik. Tagadhatatlan előnye a Sprego módszertannak, hogy fejleszti

a tanulók számítógépes gondolkodását és algoritmikus készségét, ezáltal a diákok olyan tudáshoz jutnak hozzá, amely az informatika bármely területén hasznosítható.

A módszer tanításának hatékony támogatásához egy olyan látványos, színes low poly stílusú grafikai megoldást terveztünk, ami nemcsak felkelti a tanulók figyelmét, hanem hatékonyan működik limitált specifikációval rendelkező hardverek esetén is. Ezen felül biztosítjuk a különböző platformok közötti átjárhatóságot, hogy a szoftver minél szélesebb körben elérhető legyen (interaktív táblán, asztali számítógépen, okostelefonon). Továbbá beépítettük az interaktív lejátszás lehetőségét az egyes vizualizációkhoz kapcsolódóan.

Szoftverünk egy korábbi kutatás eredményeként elkészült programon alapszik, amelyben a szerzők egy olyan 2D vizualizációs applikációt készítettek, amely animációkon keresztül szemlélteti lépésről lépésre a különböző alapvető Sprego algoritmusokat [22]. Az alkalmazás legfontosabb jellemzői, hogy több platformon elérhető, matrjoska babákat használ avatárokként és a való életből vett, szemléletes példákkal operál.

A fő feladatunk az volt, hogy a feltételes számlálás és a lineáris keresés algoritmusának vizualizációját implementáljuk a már meglévő alkalmazásból kiindulva. Ehhez úgy kellett átültetnünk a környezetet, hogy a 2D megvalósítás fentebb említett fő előnyeit és aspektusait megtartjuk, miközben a 3D környezet által biztosított lehetőségek kihasználásával új funkciókat is implementálunk.

Az alkalmazás jövőjével kapcsolatban már vannak terveink és kitűzött céljaink. Az egyik, hogy szeretnénk új funkciókkal és új algoritmusok vizualizációjával bővíteni alkalmazásunkat. A nyelvi támogatás bővítését is tervezzük további fordítások hozzáadásával. Ezen túlmenően szeretnénk minél több különböző forrásból visszajelzéseket gyűjteni annak érdekében, hogy megfelelő adataink legyenek az alkalmazás használatával kapcsolatban. A kapott adatokat felhasználva tudnánk biztosítani, hogy a szoftver evolúciója a megfelelő irányba menjen végbe. Tervezzük továbbá a Google Play online felületre is publikálni munkánkat a könnyű elérhetőség biztosításának érdekében. A legfontosabb célunk pedig a szoftver minél szélesebb oktatási környezetben történő kipróbálása és hatékonyságának vizsgálata kontrollált keretek között az általános és középiskolákban, valamint a felsőoktatásban egyaránt.

Irodalom

1. Gy. Polya: *How To Solve It. A New Aspect of Mathematical Method.* (2nd Edition 1957), Princeton University Press, Princeton, New Jersey. (1954)
2. J. J. G. Merriënboer, J. Sweller: Cognitive Load Theory and Complex Learning: Recent Deve-lopments and Future Directions. *Educational Psychology Review*, 17(2), 147–177. DOI= http://doi.org/10.1007/s10648-005-3951-0. (2005)
3. R. R. Skemp: *A matematikatanulás pszichológiája.* Edge 2000 Kiadó, Budapest. (1975)
4. D. Kahneman: *Thinking, Fast and Slow*. New York: Farrar, Straus; Giroux. (2011)
5. R. Panko: The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad. *Proceedings of the 46th Hawaii International Conference on System Sciences*, January 7-10, 2013, Maui, Hawaii. (2013)
6. 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: 2017. 07.)
7. T. Bell, H. Newton: *Unplugging Computer Science.* Improving Computer Science Education, Routledge (2013).
8. P. Biró, M. Csernoch: *Unplugged tools for building algorithms with Sprego.* International Conference on Education and New Development, Lisbon, Portugal, (2017).
9. Csernoch Mária: *Programozás táblázatkezelő függvényekkel – Sprego.* Budapest, Műszaki Könyvkiadó (2014).
10. K. Sebestyén, G. Csapó: *Visualising Sprego Inequality Problems with 2D Representations.* The Turkish Online Journal of Educational Technology. Accepted (2018).

11. Microsoft: LOOKUP function (2018a). https://support.office.com/en-us/article/lookup-function-446d94af-663b-451d-8251-369d5e3864cb. (utoljára megtekintve: 2018. 08.)
12. Microsoft: HLOOKUP function (2018b). https://support.office.com/en-us/article/hlookup-function-a3034eec-b719-4ba3-bb65-e1ad662ed95f. (utoljára megtekintve: 2018. 08.)
13. Microsoft: VLOOKUP function (2018c). https://support.office.com/en-us/article/vlookup-function-0bbc8083-26fe-4963-8ab8-93a18ad188a1. (utoljára megtekintve: 2018. 08.)
14. Microsoft: INDEX function (2018d). https://support.office.com/en-us/article/index-function-a5dcf0dd-996d-40a4-a822-b56b061328bd. (utoljára megtekintve: 2018. 08.)
15. Microsoft: MATCH function (2018e). https://support.office.com/en-us/article/match-function-e8dffd45-c762-47d6-bf89-533f4a37673a. (utoljára megtekintve: 2018. 08.)
16. J. Walkenbach: *Microsoft Excel 2010 Bible*. Wiley Publishing, Inc. Indianapolis, (2010) 202
17. K. Freiermuth, J. Hromkovič, B. Steffen: *Creating and Testing Textbooks for Secondary Schools.* In: R. T. Mittermeir, M. M. Sysło (Eds.), Informatics Education - Supporting Computational Thinking Berlin Heidelberg, Germany: Springer (2008) 216–228 http://link.springer.com/chapter/10.1007/978-3-540-69924-8_20, pp. 219. (utoljára megtekintve: 2016. 01.)
18. M. Csernoch, P. Biró, K. Abari, J. Máth: *Programozásorientált táblázatkezelői függvények.* XIV. Országos Neveléstudományi Konferencia, Debrecen, Hungary, (2014).
19. OFI: *Informatika kerettanterv.* (2013) http://kerettanterv.ofi.hu/02_melleklet_5-8/2.2.15_informat_5-8.doc. (utoljára módosítva: 2018.08.)
20. G. Csapó, K. Sebestyén: *Educational Software for the Sprego Method.* The Turkish Online Journal of Educational Technology, INTE 2017 October, (2017) 986-999 http://www.tojet.net/special/2017_10_1.pdf. ISSN 2146-7242 (utoljára megtekintve: 2018. 07.)
21. G. Csapó: *Sprego Virtual Collaboration Space.* 8th IEEE International Conference on Cognitive Infocommunications, (2017) 137-142 http://ieeexplore.ieee.org/document/8268230/. ISBN 978-1-5386-1264-4. DOI=10.1109/CogInfoCom.2017.8268230 (utoljára megtekintve: 2018. 01.)
22. G. Csapó, K. Sebestyén: *Sprego – Spreadsheet Lego* (2018). https://play.google.com/store/apps/details?id=hu.sprego.oktatoprogram. (utoljára megtekintve: 2018. 05.)
23. Unity Technologies: *Unity - The world's leading content-creation engine* (2018). https://unity3d.com/unity. (utoljára megtekintve: 2018. 08.)
24. Unity Documentation: *Unity Scripting API* (2018). https://docs.unity3d.com/ScriptReference/index.html. (utoljára megtekintve: 2018. 08.)
25. Unity Forums: *Unity Forum topics* (2018). https://forum.unity.com/. (utoljára megtekintve: 2018. 08.)
26. MagicaVoxel: *Magicavoxel by ephtracy* (2018). https://ephtracy.github.io/. (utoljára megtekintve: 2018. 08.)
27. Blender Foundation: *Free and Open 3D Creation Software* (2018). https://www.blender.org/. (utoljára megtekintve: 2018. 08.)
28. GIMP: *About GIMP* (2018). https://www.gimp.org/about/. (utoljára megtekintve: 2018. 08.)