JavaScript könyvtárak programozott rajzolás alapú tanulás támogatásához

Visnovitz Márton[1], Horváth Győző[2]

{[1]visnovitz.marton, [2]horvath.gyozo}@inf.elte.hu
ELTE Eötvös Loránd Tudományegyetem, Informatikai Kar

Absztrakt. Több kutatás foglalkozott korábban a JavaScript programozási nyelv oktatásban való használhatóságával, annak előnyeivel, hátrányaival. Egy korábbi kutatásunkban ennek a kérdéskörnek egy szűkebb területét, a böngészőben történő programozott, Canvas API alapú grafikát, és az arra épített konstrukcionista módszertan lehetőségeit vizsgáltuk. Kutatásaink és tapasztalataink feltárták a platform és a technológia több olyan hiányosságát, melyek javításával még hatékonyabb oktatási, tanulási eszközt kaphatunk. Ennek a cikknek a célja ezen problémák, illetve az ezen problémákat orvosló JavaScript eszközök bemutatása.

Kulcsszavak: JavaScript, programozott grafika, animáció, szimuláció, játékkészítés, Canvas API

1. Bevezetés

Az internet elterjedésével egyre több tanítási környezet jelent meg webes alkalmazásként a böngészőben, amelyeknek egy része a programozásoktatást tűzte ki célul ezen a viszonylag új platformon [1]. Kezdetben adta magát, hogy ezt a célt a böngészőben natívan futó programozási nyelv, a *JavaScript* segítségével érjék el, hiszen ehhez semmilyen plusz erőforrásra nem volt szükség sem a kliens, sem a szerver oldalon. Nem kellett a felhasználónak kiegészítőket telepítenie a böngészőjében, és nem kellett komoly futtató infrastruktúrát kialakítania a fejlesztőknek a szerveroldalon. Nem véletlen tehát, hogy az eleinte megjelenő webes programozásoktató alkalmazások jelentős része JavaScripten keresztül ismertette meg az alapokat a tanulókkal.

E programozási nyelvi választást segítette az is, hogy a JavaScript tervezéséből adódóan egy könnyen használható nyelv, amely kellően rugalmas a használatát illetően, és sokféle programozási paradigmát támogat, így az oktatási anyagok készítői könnyen a saját elképzeléseikhez tudták idomítani a tanítandó koncepciókat és azok formáit. Az alap nyelvi elemek és szolgáltatások fölé viszonylag egyszerűen lehetett egy plusz absztrakciós réteget kialakítani, így egy olyan eszközkészlet definiálása, amely a feladatok megoldásához szükséges, nem kívánt nagy erőfeszítést a fejlesztők részéről.

A JavaScript programozási nyelv nemcsak a webes oktatóplatformokon van jelen mint a programozásoktatás eszköze, hanem a felsőoktatásban is megjelent a bevezető programozástanításban [2,3,4]. Léteznek már egészen fiatal korosztálynak szóló könyvek is, melyek a JavaScriptet használják a programozás bevezetésére [5], de az amerikai Stanford egyetem bevezető programozási kurzusa[1] is a JavaScript nyelvet használja, kiegészítve egy tanulási célokat támogató absztrakciós réteggel.

Az idő előrehaladtával, és a webes platform fejlődésével egyre több klasszikus környezet is meg tudott jelenni a weben [6], így ma már számos példát találunk böngészőben futó teknőcgrafikára (pl. Python with turtle[2] a Repl.it[3]-en[4]), blokk-alapú programozásra (pl. Scratch[5]), hagyományos oktatási

[1] https://web.stanford.edu/class/cs101/
[2] https://docs.python.org/3/library/turtle.html
[3] https://repl.it/
[4] https://turtleacademy.com/
[5] https://scratch.mit.edu/

célú és ipari felhasználású nyelvek használatára (pl. Repl.it), vizuális programozásra, eseményvezérelt felhasználói felületek kialakítására (pl. Code.org).

2. A grafikus programozás lehetőségei JavaScript nyelven

A mai világban a hagyományos, matematikai problémákon keresztül bevezetett programozás mellett nagyon nagy szerepet kapnak a különféle vizuális környezetek a programozás oktatásában. Egyre több programozást oktató rendszer, tananyag vezeti be a programozás fogalmait valamilyen programozott grafika, animáció vagy játékkészítés révén. Ezen programozásoktatási stratégiák hatékonyan támogatják a programozás alapfogalmainak elsajátítását, miközben a vizuális aspektus révén nagy teret engednek a kreativitásnak és magasan tartják a tanulók motivációját [7,8]. Sok környezet a JavaScript programozási nyelvet használja az ilyen jellegű grafikus programok készítéséhez [9]. A böngésző, a HTML nyelv és a JavaScript több, remek eszköztárral rendelkezik grafikák készítéséhez.

Raszteres grafikákat a böngésző Canvas API-ján keresztül készíthetünk alacsony szintű, de intuitív műveletek segítségével. A HTML nyelv `canvas` (vászon) eleme[6] egy olyan programozható rajzvásznat biztosít, melyen JavaScript segítésével tudunk úgynevezett „útvonalakat" definiálni vonalak és egyszerű alakzatok (téglalap, kör(ív), ellipszis) segítségével. Ezen útvonalaknak tudjuk adott stílusú vonallal megrajzolni a pályáját, valamint kitölthetjük őket tetszés szerinti színnel. Ez a módszer egy *imperatív* stílusú, programozott grafika készítését teszi lehetővé. Ez a fajta imperatív, programozott stílus közel áll a klasszikus, algoritmikus programozási ismeretekhez, lehetővé teszi, hogy annak ismereteit könnyen alkalmazzuk/tanítsuk grafikák készítésén keresztül.

Egy másik lehetőség a böngészőben a vektoros grafikák készítése. Ehhez a HTML nyelv beágyazott SVG támogatása ad kiváló eszközt. Amennyiben a tanuló korábban már találkozott a HTML nyelvvel a szintaxis ismerős lesz számára, csak a különböző alakzatok neveit, illetve a beállításokat szolgáló attribútumokat kell megismerni. Ezzel a *deklaratív* megközelítéssel könnyen összeköthetővé válik a böngészőben történő statikus ábrák készítése más ismeretkörökkel, például weboldalak készítésével a HTML nyelv révén vagy a (vektor)grafikai ismeretekkel az SVG formátum és a vektorgrafika elmélete révén. Ilyen formán ez a fajta megközelítés inkább az alkalmazói ismeretek oktatásához köthető.

Habár mindkét megközelítésnek vannak előnyei oktatási szempontból, mégis inkább a Canvas API alapú megoldások a népszerűek a webes oktatási platformok tananyagaiban.

2.1. Animációk, játékok készítése a böngészőben

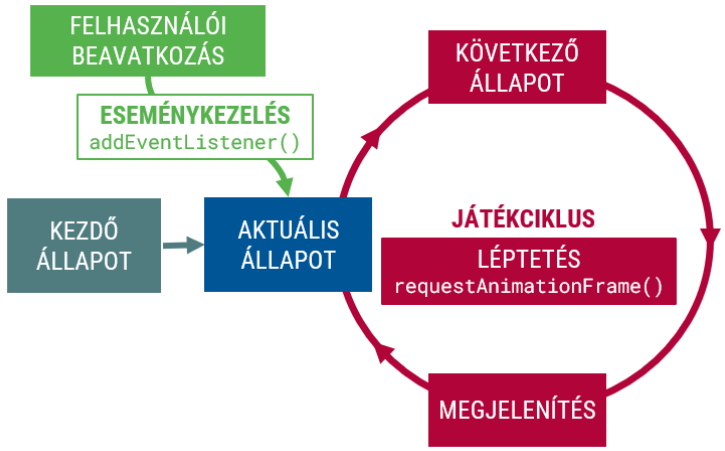
A böngésző azon túl, hogy lehetőséget biztosít statikus ábrák programozott készítésére a Canvas API-n keresztül, illetve vektorgrafikus ábrák készítésére az SVG dokumentumok beágyazása révén, rendelkezik egy olyan programozási eszköztárral is, mellyel könnyedén készíthetők animációk, szimulációk és játékok is natív eszközök segítségével [10].

Az animációk, szimulációk készítésének alapja az ábra folyamatos változása. Ez a fajta folyamatos változás megvalósítható a már meglévő elemek mozgatásával (pl. egy SVG grafika esetén) vagy az ábra gyors és folyamatos újrarajzolásával. Ez utóbbi módszert jellemzően a Canvas alapú grafika esetén használjuk. A grafika ilyen jellegű folyamatos változtatására a böngésző biztosítja a fejlesztő számára a `requestAnimationFrame` függvényt[7], mely az adott számítógép erőforrásainak megfelelően, lehetőség szerint a képernyő képfrissítési gyakoriságához igazítva képes egy függvény periodikus meghívására, és ebben az újrarajzolást elvégezve az ábra folyamatos változtatására.

[6] https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement

[7] https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame

Animációk, szimulációk és játékok készítésekor, mint minden más webes alkalmazás esetén, a megjelenés mellett a másik két fontos komponens: az állapot tárolása, illetve az események kezelése [11]. Az adatok (állapot) tárolása és kezelése a megjelenéstől (grafikától) teljesen függetlenül, alapvető nyelvi elemek (változók, függvények) segítségével megvalósítható. Az események kezelésére egyszerű megoldást biztosít a böngésző mint programozási platform az `addEventListener` függvény/metódus révén, hiszen az eseménykezelés minden böngészőben futó JavaScript programnak alapvető részét képezi. Ezen elemek segítségével könnyen felépíthető egy animációs-/játékciklus, mely működtetni képes a programunkat. Minden ciklusléptésben módosítjuk az állapotteret, ami alapján ugyanebben a lépésben megtörténik a kirajzolás. Az eseménykezelőkben lefutó logika pedig ugyancsak az állapotteret módosítja, amely hatása a kirajzolás során azonnal meg is jelenhet.



1. ábra: Az animációs-/játékciklus sematikus ábrája [12]

2.2. Canvas API használatának előnyei és hiányosságai

A böngésző és a Canvas API egy kiváló eszköz programozott grafikák, animációk és játékok készítésére. A webes platform ezen fajta sokoldalúsága lehetőséget nyújt a feladattípus-orientált programozásoktatás több stratégiájának alkalmazásához [7,8] a böngészőn belül a JavaScript programozási nyelv segítségével. A platform különböző stratégiák alkalmazása révén akár a bevezető programozástól kezdve is alkalmas lehet a programozás oktatására [2]. Ezt erősítik meg az ELTE Informatikai Karának nyári táborában szerzett tapasztalataink is. A programozás Canvas API alapú tanítására a konstrukcionista neveléselmélet elveit alkalmazva kialakítható egy projektközpontú, élményalapú tanulási környezet [10]. Ebben a tanulási környezetben nem szükséges semmilyen programozási előismeret és egy programozási nyelv segítségével lehetőségünk van az alapoktól akár a komplexebb játékok készítéséig eljutni, út közben számos programozási koncepciót megismerve.

A környezet használatának további előnye, hogy nem igényel semmilyen speciális szoftvert, csupán egy böngészőprogramra és egy szövegszerkesztőre (akár jegyzettömb) van szükség a munkához. Mivel a böngészőben natív technológiákat használunk, ezért akár webes szerkesztőkkel is könnyen integrálható a technológia. Akár teljeskörű online integrált fejlesztőkörnyezetekben (pl. Repl.it) is könnyen lehetséges a Canvas alapú programozás oktatása.

A Canvas API alapú programozás hátrányai között megemlíthetjük, hogy a natív Canvas eszközkészlet viszonylag alacsony szintű, az alapvetően elérhető formák száma kicsi. Habár sok minden beállítható a környezetben, de ezek a beállítások sok apró műveleten keresztül érhetőek el. Animációkat és játékokat is könnyen készíthetünk ezzel a technológiával, de néhány funkció és haladóbb lehetőség

csak körülményesen valósítható meg ebben a környezetben. Ezek közé tartozik például, hogy ha precízen akarjuk kezelni a mozgó elemek sebességét, akkor nekünk kell követnünk és számolnunk az animáció/játék során az eltelt időt, illetve az is, hogy a vásznak létrehozása, illetve azok rajzolási kontextusához[8] való hozzáférés megkívánja minimális mértékben a HTML kód szerkesztését, illetve a JavaScript oldalról némi, a lényeghez közvetlenül nem tartozó kód írását is.

3. Külső könyvtárak programozott grafika támogatására

Az egyik első online programozás-oktatási platform, a Khan Academy[9] programozásba bevezető kurzusa[10] is grafikus elemek JavaScriptben történő programozásával épül fel. Ehhez egy oktatási célú Canvas könyvtárat, a Processing.js[11]-t használja. Számos más webes oktatási platform (pl. Udacity, Udemy, Coursera, Codacademy) szintén rendelkezik a csak nyelvi elemeket oktató kurzusok mellett Canvas API alapú tananyagokkal. Ezek egy része a natív utasításokat tanítja, de vannak olyanok is, amelyek (játék)keretrendszereket használnak.

A külső könyvtárak használata felvet egy újabb kérdést is: hogyan töltsük be a külső könyvtárat a programunkba. A JavaScript nyelv erre több lehetőséget is biztosít, de a leggyakoribb megoldás a HTML kódban külön `<script>` tagben megadni a külső könyvtár elérési útját. Ez lehet egy helyi fájl is, de gyakoribb, hogy valamilyen online forrásból (pl. CDN) töltjük be a függőséget. Ezt a megoldást használja a p5.js[12] könyvtár online homokozója[13] is, mely alapértelmezetten el is rejti a HTML fájlt, melyben ez a kódrészlet található, és csak a felhasználó által írt kódot helyezi fókuszba.

3.1. Magas szintű absztrakciót használó könyvtárak

A technológia előző fejezetben bemutatott hiányosságinak orvoslására sok platform használ valamilyen a Canvas API-re alapuló, oktatási célú vagy vizualizációs függvénykönyvtárat, mely magasabb absztrakciós szinten megfogalmazott eszközkészletet definiál. Ilyen függvénykönyvtár például a korábban említett Processing.js vagy annak továbbfejlesztése a p5.js. Ezek a könyvtárak a legtöbb esetben egy vastagabb, magas szintű absztrakciós réteget építenek a Canvas API felé, elfedve ezzel a böngészőben elérhető natív műveleteket, teljesen új eszközkészletet definiálva.

A Processing.js (illetve az azt felváltó p5.js) oktatási célból talán a legkiemelkedőbb Canvas API-re épülő függvénykönyvtár. Többek között éppen az hívta életre, hogy a programozás grafikus élményen keresztüli bevezetéséhez egy kellően egyszerű, jól használható alapot adjon, hogy a tanuló a feladatra és ne a környezet jellegzetességeire koncentrálhasson. Ennek megfelelően számos azonnal elérhető függvényt definiál, illetve vannak olyan speciális függvényei is, amelyek külön szolgáltatásokat biztosítanak. Így például, ha implementálunk egy `draw` nevű függvényt, akkor azt a rendszer egy animációs ciklusban hívogatja meg. Az oktatási profil mellett erős hangsúlyt kap a kreatív oldal is, így kulturális projektekben, illetve adatvizualizációk során is használják ezt a függvénykönyvtárat.

2018-tól a Processing.js-t felváltotta a p5.js nevű függvénykönyvtár, ami a nagy előd nyomdokain elindulva továbbra is az élményalapú programozást vallja magáénak. A böngésző rasztergrafikai képességein túllépve azonban számos egyéb izgalmas elemhez is magas szintű támogatást nyújt, így például DOM programozáshoz, hang- és videóelemek programozásához, 3D-s animációkhoz. Egy

[8] https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D
[9] https://www.khanacademy.org/
[10] https://www.khanacademy.org/computing/computer-programming/programming
[11] https://github.com/processing-js/processing-js
[12] https://p5js.org/
[13] https://editor.p5js.org/

sokkal szélesebb és a modern böngészők lehetőségeit sokkal jobban kihasználó palettát kínál fel a webes programozásoktatáshoz (és természetesen egyéb vizualizációkhoz).

Az általános célú, Canvas API-ra épülő függvénykönyvtárak mellett léteznek kifejezetten interaktív 2D animációk készítésére készített függvénykönyvtárak is, melyek hatékonyan használhatók az oktatásban is [13]. A magas szintű absztrakciót használó könyvtárak körében érdemes még megemlíteni az amerikai Stanford Egyetem bevezető CS101 kurzusán[14] használt „JavaScript változatot" is. Ez a könyvtár egy egészen magas szintű absztrakciót vezet be a JavaScript nyelv felett, mely gyakorlatilag a böngészőben megtalálható összes elemet elfedi. Ez a kiegészítése a JavaScriptnek kifejezetten ehhez a kurzushoz készült, célja, hogy a hallgatók ne a nyelvi, hanem az algoritmikus elemekre fókuszáljanak a programozás kezdő lépéseiben.

3.2. Az alacsony szintű absztrakció lehetőségei

Lehetséges azonban olyan függvénykönyvtár készítése is, mely fókuszáltan a platform és a technológia hiányosságainak kijavítását célozza meg, miközben minél nagyobb mértékben támaszkodik a platform nyújtotta *natív* lehetőségekre. Ebben az esetben nem kapunk nagyobb eszköztárat, csupán az alapból elérhető eszközök használatát tesszük kényelmesebbé, a komplexebb eszközöket magunknak kell felépíteni (pl. összetett alakzatok rajzolásához érdemes saját – paraméterezett – függvényeket létrehozni, melyek segítségével azok megrajzolása könnyebbé válik). Az eszköztár ilyen jellegű, saját magunk által történő bővítése is hozzájárulhat a tanulási folyamathoz, hasonlóan, mint Logo környezetben a saját eljárások bevezetése sorminták rajzolásához.

Kutatásunk során egy ilyen függvénykönyvtár megvalósíthatóságát vizsgáltuk, majd el is készítettük annak prototípusát. Oktatási tapasztalataink, illetve a táborainkban résztvevő diákok visszajelzése alapján meghatároztuk a két legfőbb hiányosságát a natívan elérhető eszközkészletnek, melynek javításával reményeink szerint minőségi javulás érhető el a tanulási/tanítási folyamatban.

3.2.1. Saját függvénykönyvtár megvalósítása

Saját függvénykönyvtárunk prototípusának megvalósításakor törekedtünk arra, hogy minél vékonyabb legyen az absztrakciós réteg a Canvas API felett amellett, hogy a legfontosabb hiányosságokat orvosoljuk. Ennek megfelelően két új absztrakció (osztály) került bevezetésre:

1. A `Canvas` osztály egyesíti, és egyetlen objektumon keresztül teszi elérhetővé a böngészőben a `<canvas>` HTML elem, illetve annak „2D" rajzolási kontextusának műveleteit. Ezen új osztály bevezetésével szükségtelenné válik a vászon (vagy vásznak) létrehozásakor a HTML kód szerkesztése, illetve a rajzolási kontextust elkérő JavaScript kód is (2. és 3. ábra).

```
<!-- HTML -->
<canvas></canvas>

// JavaScript
const canvas = document.querySelector("canvas");
const context = canvas.getContext("2d");
```

2. ábra: A `<canvas>` elem és annak rajzolási kontextusának elérése natív eszközök segítségével

```
// JavaScript
const canvas = new Canvas();
```

3. ábra: A rajzvászon elérése a `Canvas` osztállyal

[14] https://web.stanford.edu/class/cs101/

Az új osztályon keresztül lehetőség van hozzáférni a vászon elem fontosabb tulajdonságaihoz (szélesség, magasság, stílusok), illetve a rajzolási kontextus műveleteihez is. Ez a megoldás amellett, hogy kényelmesebb használtot tesz lehetővé, biztosítja, hogy a natívan rendelkezésre álló teljes fejlesztői eszköztár továbbra is rendelkezésre áll. A Canvas objektum konstruktora ezen túl lehetőséget nyújt a vászon kezdő méreteinek beállítására is.

```
// A canvas elem tulajdonságának módosítása
canvas.width = 500;

// A rajzolási kontextus műveletének meghívása
canvas.fillRect(0, 0, 100, 100);
```

4. ábra: A `<canvas>` elem egy tulajdonságának és a rajzolási kontextus egy műveletének elérése a Canvas objektumon keresztül

A Canvas osztály megvalósítására a JavaScript nyelvben található Proxy típust használtuk fel. Ez teszi lehetővé, hogy a `Canvas` osztály automatikusan döntse el, hogy egy adott metódus vagy tulajdonság a Canvas HTML elemhez vagy a rajzolási kontextushoz tartozik-e. Ahhoz, hogy a HTML kódot ne kelljen szerkeszteni új vászon létrehozásakor, a konstruktor automatikusan létrehozza a szükséges HTML elemet és azt be is szúrja a dokumentumba.

2. A `Timer` osztály egyesíti a böngésző `requestAnimationFrame` és `setInterval` műveleteinek lehetőségeit és interfészt biztosít egy adott függvény adott időközönként történő futtatására úgy, hogy a háttérben a hatékony `requestAnimationFrame`-et használja. Lehetőséget ad automatikus (képernyő frissítési gyakoriságnak megfelelő) időközönkénti, illetve a felhasználó által megadott adott időközönkénti futtatásra is. Az időzítő leállítható és újraindítható, illetve van lehetőség az időzítő frissítési gyakoriságát állítani az `interval` tulajdonságon keresztül. A `Timer` osztály ezen túl automatikusan átadja paraméterként az előző futás óta eltelt időt a paraméterként megadott függvénynek (4. és 5. ábra), ezáltal könnyen készíthetünk realisztikus, a ténylegesen eltelt időtől függő mozgásokat.

```
// JavaScript
function update(dt) { /*...*/ }

let lastFrameTime = performance.now();
function next() {
  const currentTime = performance.now();
  const elapsedTime();
  update(dt);
  lastFrameTime = currentTime;
  requestAnimationFrame(next);
}
```

5. ábra: A képfrissítések között eltelt idő manuális követése natív eszközökkel

```
// JavaScript
function update(dt) { /*...*/ }

const timer = new Timer(update);
timer.start();
```

6. ábra: A képfrissítések között eltelt idő követése a `Timer` osztállyal

4. Összefoglalás, további célok

Elmondható, hogy a böngészőben történő grafikus programozás – hiányosságai ellenére – egy jó eszköz lehet a programozás oktatásában, mint ahogy azt a számos online platform gyakorlata, illetve saját oktatási tapasztalataink mutatják. A technológia hiányosságának kiküszöbölésére a leggyakoribb módszer valamilyen absztrakciós réteg használata, mely a natív eszközök fölött definiál saját eszköztárat. Ezen absztrakciós rétegeket jellemzően valamilyen külső programkönyvtár formájában valósítják meg. Az absztrakciós réteg „vastagságától" függően beszélhetünk magas szintű (vastag), illetve alacsony szintű (vékony) absztrakciós rétegekről. Ezek tulajdonságait az 1. táblázat foglalja össze.

Magas szintű absztrakciós függvénykönyvtár	**Alacsony szintű absztrakciós függvénykönyvtár**
többnyire saját eszközkészlet	többnyire a platform natív eszközkészlete
több koncepcionális elem elrejtése	koncepcionális elemek elrejtése minimális
kifejezetten oktatási célú (is lehet)	az natív eszközkészlet alapvetően nem oktatási célú, de oktatásra is jól használható, a kiegészítések az oktatási célt szolgálják

1. táblázat: Magas- és alacsony szintű absztrakciót használó függvénykönyvtárak összehasonlítása

Az alacsony szintű absztrakciós réteget biztosító függvénykönyvtár-prototípus, melyet a kutatásunk keretében készítettünk igazolja, hogy lehetséges a Canvas API használatakor felmerülő legfontosabb problémákat orvosolni egy ultravékony programozási interfész segítségével. Természetesen ez a prototípus jelenlegi formájában még nem alkalmas oktatásban történő éles alkalmazásra, további fejlesztések szükségesek.

A könyvtár kódjának letisztázása és további fejlesztése mellett az oktatásban történő használathoz szükséges egy jól használható dokumentáció készítése, valamint oktatási segédanyagokra, tananyagokra is szükség van. A hosszabb távú tervek között szerepel további feladattípus-orientált tanítási stratégiák támogatásának kialakítása további alacsony szintű könyvtárak segítségével. Ilyen például az automataelvű teknőcgrafika (ld. Logo[15]) és ügynök-vezérelt szimulációk (ld. NetLogo[16]) támogatása.

Irodalom

1. Hováth, G., L. Menyhárt, *Oktatási környezetek vizsgálata a programozás tanításához*, (2014).
2. Horváth, G., L. Menyhárt, *Teaching introductory programming with JavaScript in higher education*, in *Proceedings of the 9th International Conference on Applied Informatics*, (2015), pp. 339–350, DOI: 10.14794/icai.9.2014.1.339.
3. Horváth, G., L. Menyhárt, L. Zsakó, *Viewpoints of Programming Didactics at a Web Game Implementation*, *DIDMATTECH 2016*, (2016).
4. Mercuri, R., N. Herrmann, J. Popyack, *Using HTML and JavaScript, in Introductory Programming Courses*, Philadelphia, PA, (1998).
5. Morgan, N., *JavaScript for Kids*. No Starch Press, (2014).
6. Horváth, G., *A web-based programming environment for introductory programming courses in higher education*, *ANNALES MATHEMATICAE ET INFORMATICAE*, vol. 48, pp. 23–32, (2018).

[15] https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html
[16] https://ccl.northwestern.edu/netlogo/

7. Bernát, P., L. Zsakó, *Methods of Teaching Programming - Strategy*, *XXXth DIDMATTECH 2017*, pp. 40–51, (2017).
8. Bernát, P., L. Zsakó, *Programozás tanítási módszerek – stratégia a kezdetekre*, in *INFODIDACT 2019*, (2019).
9. Wu, P., *Teaching Basic Game Programming Using JavaScript*, (2009).
10. Visnovitz, M., G. Horváth, *A Constructionist Approach to Learn Coding with Programming Canvases in the Web Browser*, *CONSTRUCTIONISM 2020*, pp. 1–8, (2020).
11. Visnovitz, M., G. Horváth, *The Web - A Platform for Creation*, *CONSTRUCTIONISM 2018*. Vilnius, (2018).
12. Horváth, G., M. Visnovitz, *A böngésző mint alkalmazásfejlesztési platform*. ELTE Informatikai Kar: Budapest, (2018).
13. Végh, L., J. Udvaros, *Possibilities of creating interactive 2d animations for education using html5 canvas javascript libraries*, in *eLearning and Software for Education Conference*, (2020), pp. 269–274, DOI: 10.12753/2066-026X-20-119.