ÓBUDAI EGYETEM
Neumann János Informatikai kar
Mérnök informatikus BSc

Vizuális, interaktív programozás oktató rendszer moduláris megvalósítása

Projektmunka dokumentáció

Ráncsik Áron

2020. március 4.

Tartalomjegyzék

1. fejezet

Bevezetés

Különböző helyeken az oktatásban manapság elterjedt[13] gyakorlat az oktatott anyag játékos megfogalmazása, idegen szóval a [3]*gamification*. A módszer lényege, hogy a megtanítani kívánt ismeretet egyszerű nyers forma helyett, játékos formában tesszük emészthetővé a tanulók számára. Dolgozatomban a programozást szeretném a tanulni vágyok elé tárni, olyan formában, hogy az ne okozzon nehézséget, illetve a tanulás emléke inkább egy élmény legyen, szemben egy unalmas órával.

Cél egy olyan alkalmazás készítése melyben konkrét programozási nyelv használata nélkül lehetséges programozni. A programozást lehessen akár kézvezérléssel megvalósítani. A programozás motivációja pedig, hogy az egymás által készített különböző programkódok egymás ellen tudjanak versenyezni. A megmérettetést akár már létező, ismert számítógépes játékokhoz csatlakozva, vagy saját egyedi játékban tehetik meg. A írt kódok összecsapását pedig kiterjesztett valóságon keresztül szeretném követhetővé tenni.

Röviden összefoglalva, a következő rendszert készítettem a fent megfogalmazott célra. Első lépésként, hogy a kitűzött célt megvalósítsam, a 1.2.1 részben egy vizuális programozási környezet elkészítését részletezem, mely abban segít, hogy egy konkrét programozási nyelv tanulásának kezdeti nehézségein átugorva vizuális eszközökkel teszi elérhetővé a programozást, akár laikusok számára is. A Módszertan szekció 1.2.2 alrészben azt vizsgálom milyen lehetőségek vannak kiterjesztett valóságban való megjelenítésre. A bevezetés utolsó szekciójában 1.2.3 a kézvezérlés funkció megvalósításához szükséges ismereteket igyekszem analizálni.

1.1. Algoritmus elmélet

Az algoritmus definiálása nehéz feladat, erre nem vállalkoznék, az kijelenthető, hogy azokat a feladatokat lehet algoritmizálni melyek működése Turing gép segítségével megvalósítható. Olyan programot szeretnék készíteni, mely ebben a kategóriába tartozik.

1.2. Módszertan

A korában megfogalmazott cél elérése érdekében végzett kutatómunkát részletezi.

1.2.1. Vizuális programozási környezet

Több megoldás létezik, mely vizuális programozási lehetőséget tesz lehetővé, ezeknek az összehasonlításáról lesz szó ebben a fejezetben.

Kész megoldások vizuális programozásra

Scratch Nagyon népszerű, oktatásban méltán közkedvelten használt teljes környezet [8] [12] [14], mely vizuális programozást tesz lehetővé. Az alapvető célja olyan média alkalmazások mint pl.: animáció, köszöntések, történet mesélés, zeneklip elkészítésének folyamata közben megtanítani a programozást. Az egyedüli, intuitív tanulás is előnyben részesíti. Főleg kisebb 8-16 éves korosztály számára készült.

Az elkészült program alapértelmezetten egy saját „színpad" környezetben futtatható. Esemény vezérelt programozást is lehetővé tesz, ebben az esetben az egyszerre fellépő konkurens utasítások között esetleges versenyhelyzetet kezeli, de nem minden esetben úgy ahogy azt elsőre a használó gondolná. A programból *broadcast* utasításokkal, saját parancsok, szkriptek végrehajtása lehetséges. A *broadcast* utasítássokkal lehetséges a külvilággal történő kommunikáció is, de körülményes. Kiegészítőkkel bővíthető, de így sem nem lehet az alap működésre kiható szignifikáns módosításokat végrehajtani.

Előnyei

- Szabadon elérhető, nyílt forráskódú.
- Kész megoldásról beszélve, sokkal kevesebb munka segítségével lehet használni.
- Egyedi működés *broadcast* utasításokkal lehetséges.
- Az alap környezet többnyire magas szintű előre definiált utasításokban gazdag pl.: animációk
- Kezdők számára is egyszerűen érthető felület.
- Tapasztalatok alapján nagyon ismert az általános iskolások körében.
- Érthető, naprakész dokumentáció áll rendelkezésre

Hátrányai

- A *broadcast* utasításon kívül máshogy nem megoldható egyedi utasítás létrehozása.
- Az elkészült program csak a saját környezetén belül futtatható, nem lehetséges egyedi környezetben futtatni.
- Az elkészült program egyszerűen nem alakítható valódi programkóddá. (Van harmadik féltől származó kész megoldás)
- Monolitikus, bővítésre nézve többnyire zárt rendszer.
- Ugyan van webes felület, de saját oldalba nem ágyazható.

Snap! Új, még kevésbé ismert, teljes megoldás [4], segítségével vizuálisan van lehetőségünk programozni. A Scratch-el szemben több szabadságot biztosít a személyre szabhatóság tekintetében. Sokkal bonyolultabb problémák megoldására is alkalmas, az objektum orientált paradigmát is támogatja. A környezet kevésbé kezdő barát. *Url* utasításokkal webes csatlakozó felületen keresztül tud fogadó és küldőként is könnyen kommunikálni a külvilággal. A felhő alapú gépi tanulás szolgáltatásokhoz csatlakoztatva akár „AI" fejlesztésre is alkalmas [6]. A párhuzamos ciklusokat időosztásos módszerrel futtatja párhuzamosan „yield" utasítások segítségével, de erre is igaz. hogy mindezt igyekszik elrejteni a felhasználók elől és nem igényel különösebb beavatkozást az alapvető működéshez hozzá lehet szokni.

Előnyei

- Szabadon elérhető, nyílt forráskódú.
- Kész megoldásról beszélve, sokkal kevesebb munka segítségével lehet használni.
- Egyedi működés *url* utasításokkal lehetséges. Sokkal kényelmesebben mint Scratch esetén.
- Az alap környezet előre definiált utasításokban gazdag pl.: *sprite*-ok vezérlése.
- Bonyolultabb problémák megoldására is alkalmas.
- Érthető, naprakész dokumentáció áll rendelkezésre
- Van webes felülete, de saját oldalba nem ágyazható.
- Egyedi blokkok létrehozását támogatja.
- Objektum orientált paradigmát támogatja

Hátrányai

- Az elkészült program csak a saját környezetén belül futtatható, nem lehetséges egyedi környezetben futtatni.
- Az elkészült program nem alakítható valódi programkóddá.
- Nem biztosít felhasználható könyvárat, csak a projekt forrását felhasználva lehet egyedi alkalmazásokban használni.
- Jelenleg (2020. március 4.) béta állapotban van.
- Monolitikus, bővítésre nézve többnyire zárt rendszer.
- A objektum orientáltság különböző megkötésekkel érhető el.

GP Általános célú blokk alapú [10] [9] programozási nyelv. Egy valódi programozási nyelv mely blokk és kód alapú programozási lehetőséggel is rendelkezik. Sok alacsony szintű funkciók érhetőek el benne. Sokkal közelebb áll a valóságos programozáshoz a korábban említett lehetőségekhez képest, rendelkezik webes felülettel és a külvilággal internetes *get*, *put* utasítás lehetséges a kommunikáció egyedi rendszerekkel. Támogatja az objektum orientált paradigmát. Tartalmaz bonyolultabb adatszerkezeteket is.

Előnyei

- Kész megoldásról beszélve, sokkal kevesebb munka segítségével lehet használni.
- Egyedi működés *get, put* utasításokkal lehetséges. Sokkal kényelmesebben mint Scratch esetén.
- Az alap környezet többnyire alacsony szintű előre definiált utasításokban gazdag pl.: *set pixel* utasítás blokk.
- Bonyolultabb problémák megoldására is alkalmas.
- Moduláris módon készült.
- Érthető, naprakész dokumentáció áll rendelkezésre.

Hátrányai

- Az elkészült program csak a saját környezeten belül futtatható, nem lehetséges saját környezetben futtatni.
- Az elkészült program nem alakítható valódi programkóddá.
- Ugyan van webes felület, de saját oldalba nem ágyazható.

Alice Interaktív 3D Animációs környezet [2]. Az idézett kutatás által megfogalmazottan a célja egy 3D-s környezet interaktív fejlesztése melyben szabadon lehet felfedezni az elkészült alkotást. Főleg szkript alapú prototípus fejlesztő környezet.

Előnyei

- Kész munka elvileg kevesebb munka lehet átalakítani, felhasználni
- Adott 3D környezet
- Esemény vezérelt programozás tanítására is alkalmas

Hátrányai

- Az elkészült program csak a saját környezeten belül futtatható, nem lehetséges saját környezetben futtatni.
- Nem vizuális egy sajátos egyedi szkript nyelvvel rendelkezik.
- Az elkészült program nem alakítható valódi programkóddá.
- Nincs webes felülete, futtatható állományok telepítését igényli.
- Kissé régi, idejét múlt, már kevésbé támogatott.

Egyéb kész megoldások A fenti részben az általam, a dolgozatommal legjobban összehasonlítható szempontok alapján fontosnak tartott kész megoldásokat mutattam be. Természetesen rengeteg egyéb kész alkalmazás létezik mely vizuális programozási lehetőséget biztosít. Felsorolás szintjén itt leírok pár szerintem említésre méltó alkalmazást.

- Game Maker [5] alapvetően 2D saját szkript nyelvvel rendelkezik. A teljes verzió pénzbe kerül.
- Stencyl [7] Alapvetően játék készítő alkalmazás melyben vizuálisan lehet programozni.

Könyvtárak vizuális programozásra

Blockly Vizuális programozást támogató program könyvtár [1] [11]. A vizuális megjelenésért felelős, nem egy programozási nyelv, csak egy vizuális leíró nyelv, mely könnyedén exportálható vallós programozási kóddá. Több létező programozási nyelv kódot lehet a vizuális környezetből generálni, többek között: Javascript, Python, Lua, Dart nyelvű kódokat is. A Blockly könyvtárat sok kész alkalmazás használja a saját egyedi vizuális környezetének megvalósítására. Például a korábban említett Scratch, Snap! is használja vizuális könyvtárként, de az ezután részletezett PXT környezet is Blockly-t használ a vizuális megjelenítésre.

PXT Vizuális és szöveg alapú programozási környezet.

1.2.2. Kiterjesztett valóság

1.2.3. Kézvezérlés

1.3. Gépi látás

1.3.1. Kamera mozgás becslés

1.3.2. Markerek keresése

Irodalomjegyzék

[1] *Blockly | Google Developers*. [Online; accessed 4. Mar. 2020]. 2020. febr. URL: `https://developers.google.com/blockly`.

[2] Stephen Cooper, Wanda Dann és Randy Pausch. „Alice: a 3-D tool for introductory programming concepts". *Journal of computing sciences in colleges* 15.5 (2000), 107–116. old.

[3] Sebastian Deterding és tsai. „From game design elements to gamefulness". *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek 11*. ACM Press, 2011. DOI: `10.1145/2181037.2181040`.

[4] Brian Harvey és tsai. „Snap!(build your own blocks)". *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013, 759–759. old.

[5] Jennifer Jenson és Milena Droumeva. „Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study." *Electronic Journal of e-Learning* 14.2 (2016), 111–121. old.

[6] KM Kahn és tsai. „AI programming by children using snap! block programming in a developing country". (2018).

[7] Jiangjiang Liu és tsai. „Making games a" snap" with Stencyl: a summer computing workshop for K-12 teachers". *Proceedings of the 45th ACM technical symposium on Computer science education*. 2014, 169–174. old.

[8] John Maloney és tsai. „The scratch programming language and environment". *ACM Transactions on Computing Education (TOCE)* 10.4 (2010), 1–15. old.

[9] Jens Monig, Yoshiki Ohshima és John Maloney. „Blocks at your fingertips: Blurring the line between blocks and text in GP". *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. IEEE. 2015, 51–53. old.

[10] Yoshiki Ohshima, Jens Mönig és John Maloney. „A module system for a general-purpose blocks language". *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. IEEE. 2015, 39–44. old.

[11] Erik Pasternak, Rachel Fenichel és Andrew N Marshall. „Tips for creating a block language with blockly". *2017 IEEE Blocks and Beyond Workshop (B&B)*. IEEE. 2017, 21–24. old.

[12] Mitchel Resnick és tsai. „Scratch: programming for all". *Communications of the ACM* 52.11 (2009), 60–67. old.

[13] Marc Riar és tsai. „How game features give rise to altruism and collective action? Implications for cultivating cooperation by gamification". *Proceedings of the 53rd Hawaii International Conference on System Sciences*. 2020.

[14] *Scratch - Developers*. [Online; accessed 4. Mar. 2020]. 2019. jan. URL: `https://scratch.mit.edu/developers`.

Ábrák jegyzéke