ÓBUDAI EGYETEM
Neumann János Informatikai kar
Mérnök informatikus BSc

Vizuális, interaktív programozás oktató rendszer moduláris megvalósítása

Projektmunka dokumentáció

Ráncsik Áron

2020. május 17.

Kivonat

Your abstract goes here... ...

Tartalomjegyzék

1. Bevezetés **2**

2. Módszertan **3**
2.1. Vizuális programozási környezet . . . 3
2.1.1. Alternatív megoldások vizuális programozásra . . . 3
Scratch . . . 3
Snap . . . 4
GP . . . 4
Alice . . . 5
Egyéb kész megoldások . . . 5
Blockly . . . 6
MakeCode (PXT) . . . 6
Választott vizuális programozási környezet . . . 7
2.2. Da Vinci . . . 7
Robot Operating System . . . 7
Da Vinci Research Kit . . . 7
iRob Surgical Subtask Automation Framework . . . 8
2.3. Kiterjesztett valóság . . . 8
2.4. Kézvezérlés . . . 8
2.5. Gépi látás . . . 8
2.6. Kamera mozgás becslés . . . 8
2.6.1. Markerek keresése . . . 8

3. Megvalósítás **9**
3.1. Da Vinci webes elérhetősége . . . 9

Irodalomjegyzék **11**

Ábrák jegyzéke **13**

fejezet 1

Bevezetés

Az oktatásban manapság elterjedt[18] gyakorlat az oktatott anyag játékos megfogalmazása, idegen szóval a [5]*gamification*. A módszer lényege, hogy a megtanítani kívánt ismeretet egyszerű, nyers forma helyett, játékos formában tesszük emészthetővé a tanulók számára. Dolgozatomban szeretném a tanulni vágyok elé tárni a programozást, olyan formában, hogy az ne okozzon nehézséget, illetve a tanulás emléke inkább egy élmény legyen, egy unalmas óra helyett.

Az alábbi célokat tűztem ki. Legyen lehetőség kézvezérléssel végezni a programozást. Programozás egyik motivációja , hogy a felhasználók által készített különböző programkódok, ágensek egymás ellen tudjanak versenyezni. A megmérettetést akár már létező, ismert számítógépes játékokhoz csatlakozva, vagy saját egyedi játékban tehetik meg. Egy másik tervezett funkció, hogy az írt kódok összecsapását pedig kiterjesztett valóságon keresztül szeretném követni. További cél, hogy legyen lehetőség az egyetemen elérhető DaVinci roboton is könnyen magas-szintű programkódot futtatni.

Röviden összefoglalva, a következő irodalomkutatást készítettem a fent megfogalmazott célokra. Első lépésként, a kitűzött célokat megvalósítsa érdekében, a 2.1.1 részben a vizuális programozási környezet elkészítésének lehetőségeit 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 2.2 részben azt vizsgálom, milyen lehetőségek vannak a Da Vinci robot egyszerű maga-szintű programozására. A 2.3 szekcióban a kiterjesztett valóság megvalósításának lehetőségeit részletezem. A Módszertan utolsó szekciójában 2.4 a kézvezérlés funkció megvalósításához szükséges ismereteket igyekszem analizálni.

fejezet 2

Módszertan

Itt a korában megfogalmazott célok elérése érdekében végzett kutatómunkát részletezem.

2.1. Vizuális programozási környezet

2.1.1. Alternatív megoldások vizuális programozásra

Mivel a dolgozatban részben vizuális programozást is szeretnék biztosítani, ezért megvizsgálom milyen konkurens megoldások léteznek erre a célra. Az alábbiakban bemutatok és elemzek több kutatást, korábbi kész alkalmazásokat és programkönyvtárakat melyek mindegyike a vizuális blokk alapú programozást nyújtja.

Scratch ♦ Nagyon népszerű, oktatásban méltán közkedvelten használt teljes környezet [12, 17, 19], 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 [7], 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 [9]. 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. május 17.) béta állapotban van.
- Monolitikus, bővítésre nézve többnyire zárt rendszer.
- A objektum orientáltság különböző megkötésekkel érhető el.

GP ♦ Általános célú blokk alapú [14] [13] 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 [3]. 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 szkriptnyelvvel 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 [8] 2D játék készítő, vizuális és saját szkriptnyelvvel is rendelkezik. A teljes verzió pénzbe kerül.
- Stencyl [11] 2D Játék készítő alkalmazás melyben vizuálisan lehet programozni.

A továbbiakban kész alkalmazások helyett, vizuális programozásra készült programkönyvtárak bemutatásával fogom folytatni az választható megoldások listáját.

Blockly ♦ Google által fejlesztett, vizuális programozást támogató program könyvtár [2] [15]. 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 MakeCode (PXT) környezet is Blockly-t használ a vizuális megjelenítésre.

Előnyei *
- Szabadon elérhető, nyílt forráskódú.
- Kezdők számára is egyszerűen érthető felület.
- Érthető, naprakész dokumentáció áll rendelkezésre.
- Program könyvtár, könnyen bővíthető egyedi utasításokkal
- A program futtatási környezetére nincs megkötés.
- Nemzetközi. Több mint 40 (köztük magyar) nyelven elérhető.
- Bármilyen futtatási környezetbe könnyen integrálható.
- Rengeteg programkód generálható.

Hátrányai *
- Csak egy vizuális programozást támogató „drag & dropp" programkönyvtár.
- Viszonylag régi technológiákkal készült, nehézkes modern környezetben használni.

MakeCode (PXT) ♦ Microsoft által fejlesztett, összetett, vizuális programozói környezet támogató, nyílt forráskódú programkönyvtár [20]. A Blockly könyvtárat felhasználja a blokk alapú vizuális programozás biztosítására. A blokk alapú programozáson túl biztosít lehetőséget a szöveges programozásra is, egy beépített weboldalon futamítható fejlesztő környezetben. Abban különbözik a korábban alternatív lehetőségektől, hogy egyben nyújt egy teljes fejlesztő környezetet, amihez könnyen hozzá lehet csatolni egy saját *target* modult egyedi utasításokkal. [6] Kutatás részletesen bemutatja, hogyan lehet használni beágyazott rendszerekhez. A fejlesztő környezethez alacsony szintű mikrokontrollerhez való fordítót csatlakoztatva.

Előnyei *
- Szabadon elérhető, nyílt forráskódú.
- Kezdők számára is egyszerűen érthető felület.
- Érthető, naprakész dokumentáció áll rendelkezésre.
- Program könyvtár, könnyen bővíthető egyedi utasításokkal
- A program futtatási környezetére nincs erős megkötés.

Hátrányai *
- Integrálása korlátozott, alkalmazkodni kell a felépítéséhez, ettől eltérni nem lehetséges könnyen.
- Béta verzióban van a fejlesztés.

Választott vizuális programozási környezet

Kipróbáltam, teszteltem a felsorolt megoldásokat továbbá a korábbi elemzés összevetése után a Blockly környezet használata mellet döntöttem. A dolgozat igényeit nem lehet kész megoldásokkal pl. Scratch, Snap stb. kielégíteni, mert túlságosan zárt működésük nem teszik lehetővé, hogy egyedei játékhoz lehessen programozni ezekben az alkalmazásokban. A Blockly kellőképpen módosítható és könnyen beilleszthető, vizuális programozást biztosító programkönyvtár, mely megfelel a feladat részéről támasztott elvárásoknak.

2.2. Da Vinci

Az célok között szerepelt a Da Vinci robot egyszerű vezérlése. Az elkövetkező részben, igyekszem körbejárni a lehetőségeket és azok előnyeit hátrányait. Az egyetemen egy 2010-es első generációs da Vinci classic rendszerhez van lehetőség hozzáférni. Ezért ehhez készült az alkalmazás. A továbbiakban mindig erről a modellről lesz szó.

Robot Operating System ♦ Annak érdekében, hogy a fejlesztés során, ne kelljen folyamatosan a fizikai Da Vinci rendszerhez hozzáférnem, egy olyan környezetre van szükségem ami képes biztosítani a Da Vinci robotot virtuálisan. Ugyan sok rendszer elérhető csak felsorolás szintjén néhány mely robot vezérlő környezetet biztosít azonban az egyik legkorábbi és legnépszerűbb lehetőség a Robot Operating System [16] továbbiakban csak ROS. Dolgozatomban több szempont miatt is a legjobb választásnak a ROS rendszert bizonyult. Attól a hátrányától eltekintve, hogy nem multiplatform azaz csak Ubuntu GNU/Linux és annak is csak bizonyos verzióihoz elérhető rengeteg előnnyel rendelkezik. Sok robot vezérléssel kapcsolatos alapfunkciót tartalmaz, amit így nem kell újból megírni, biztosít egy jól elszeparált keretet külön a kommunikációra és az üzleti logikára.

Előnyei *
- Szabadon elérhető, nyílt forráskódú
- Kiforrót
- Népszerű
- Moduláris
- Kutatási célokhoz megfelelő
- Érthető, naprakész, jó dokumentáció áll rendelkezésre

Hátrányai *
- Nem multiplatform, csak Ubuntu GNU/Linux és annak is csak bizonyos verzióihoz elérhető

Da Vinci Research Kit

A Da Vinci Research Kit [10] továbbiakban DVRK egy nyílt forráskódú rendszer ami lehetővé teszi a Da Vinci robot karok programozását. A ROS rendszerhez kapcsolódik, könnyen lehetséges az utasításait használni egy ROS modult node-ot biztosít használatra. A célkitűzéseimhez képest viszonylag alacsony-szintű Da Vinci robot kar vezérlő utasítások kiadására alkalmas.

iRob Surgical Subtask Automation Framework

Nagy Tamás és Haidegger Tamas által fejlesztett keretrendszer [4]. Alapvetően arra a célra készült, hogy részleges automatizáció segítségével levegye a sebészekről a kognitív terhelést. Részfolyamatok automatizálását teszi lehetővé a keretrendszer. A ROS rendszerben elérhető node-okat biztosít, a DVRK rendszerre épül. Felépít egy hierarchikus node rendszert melynek a legmagasabb szintjén magas-szintű utasítások kiadása lehetséges. Számomra a céloknak tökéletesen megfelelő.

Előnyei *
- Szabadon elérhető, nyílt forráskódú
- Moduláris
- Kutatási célokhoz megfelelő
- Érthető, naprakész, jó dokumentáció áll rendelkezésre

Hátrányai *
- Jelenleg fejlesztés alatt áll.

2.3. Kiterjesztett valóság

A [1] kutatásból idézve kiterjesztett valóságot a következő módon definiálhatjuk: Kiterjesztett valóság az a rendszer, ami a következőket teljesíti:
- Kombinálja a valós és virtuális objektumokat a valós környezetben.
- Azonnal fut, és valós időben.
- Egymáshoz rögzíti (igazítja)

Virtuális objektumokat szeretnénk elhelyezni a kamera képen úgy, hogy azok mozgása a kamera 3D mozgásához igazodjon.

Több létező megoldás van kiterjesztett valóság elkészítéséhez, ezen megoldások összehasonlításáról lesz itt szó.

2.4. Kézvezérlés

2.5. Gépi látás

2.6. Kamera mozgás becslés

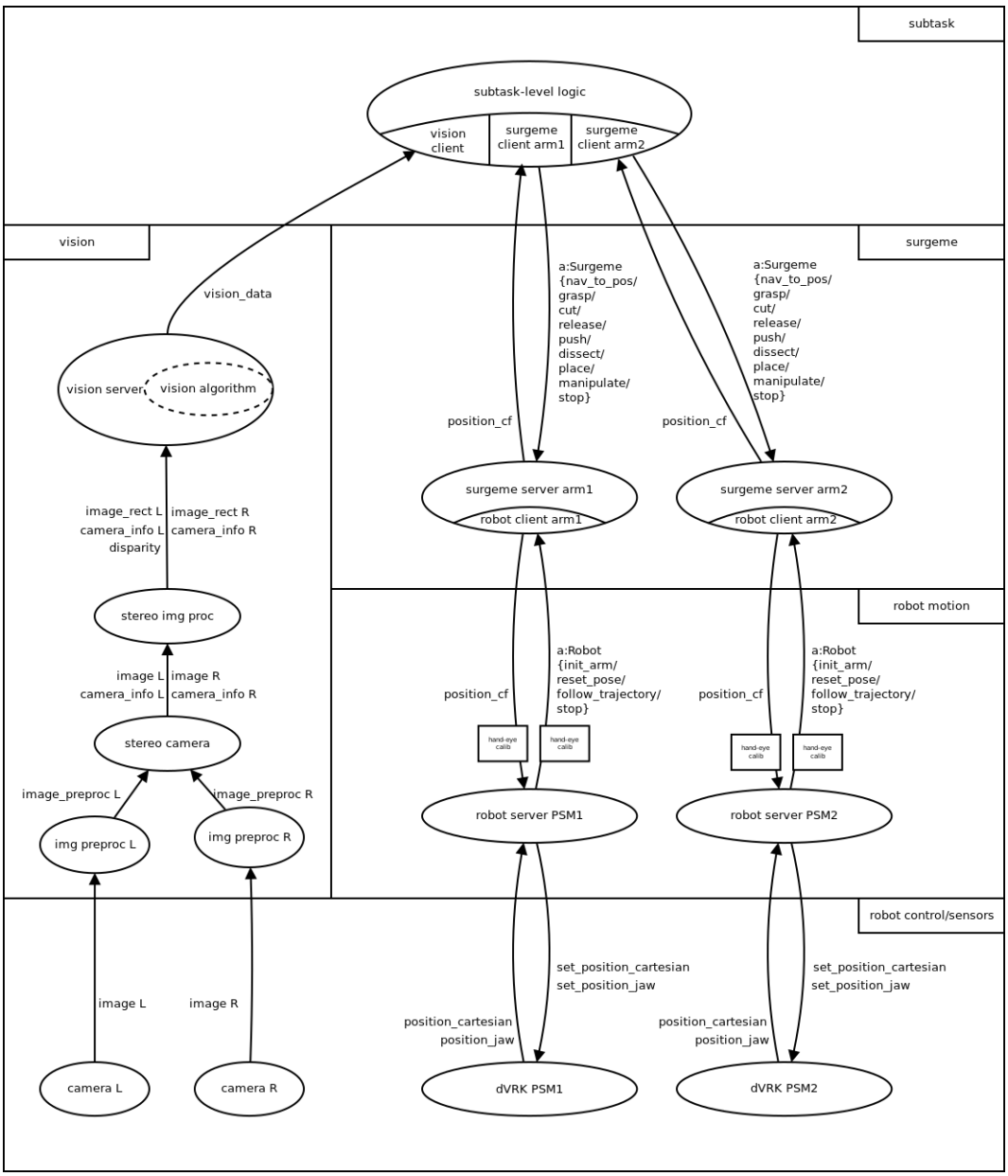
2.6.1. Markerek keresése

fejezet 3

Megvalósítás

3.1. Da Vinci webes elérhetősége

Először meg kellet ismerkednem a korábban részletezett 2.2, [4] keretrendszettszer és ROS architektúra használatával. Az irob_saf keretrendszer működését a 3.1 ábra szemlélteti.



3.1. ábra. A irob_saf keretrendszer működését szemléltei

Irodalomjegyzék

[1] Ronald Azuma és tsai. „Recent advances in augmented reality”. *IEEE computer graphics and applications* 21.6 (2001), 34–47. old.

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

[3] 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.

[4] Tamás D. Nagy és Tamás Haidegger. „A DVRK-based Framework for Surgical Subtask Automation”. *Acta Polytechnica Hungarica* 16.8 (2019), 61–78. old.

[5] 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`.

[6] James Devine és tsai. „MakeCode and CODAL: intuitive and efficient embedded systems programming for education”. *ACM SIGPLAN Notices* 53.6 (2018), 19–30. old.

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

[8] 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.

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

[10] Peter Kazanzides és tsai. „An open-source research kit for the da Vinci® Surgical System”. *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, 6434–6439. old.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] Morgan Quigley és tsai. „ROS: an open-source Robot Operating System". *ICRA workshop on open source software*. 3. köt. 3.2. Kobe, Japan. 2009, 5. old.

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

[18] 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.

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

[20] Pradeeka Seneviratne. „MakeCode Setup Fundamentals". *BBC micro: bit Recipes*. Springer, 2019, 1–28. old.

Ábrák jegyzéke

3.1. A irob_saf keretrendszer működését szemléltei 10