ÓBUDAI EGYETEM
Neumann János Informatikai kar
Mérnök informatikus BSc

Vizuális, interaktív programozás oktató rendszer moduláris megvalósítása

Ráncsik Áron

2020. május 24.

Tartalomjegyzék

fejezet 1

Bevezetés

Először szeretném bemutatni azokat a célkitűzéseket, nem-funkcionális követelményeket, különböző elvárásaimat, melyek megvalósítását tervezem dolgozatomban.

Fő célom egy olyan rendszer elkészítése, melynek a segítségével, fiatal (10-16 éves) korosztály számára lehet, elérhetővé, élvezetessé tenni a programozást, algoritmizálást. Kedvcsináló módon, szeretném az algoritmizálás tanulásra rávezetni a fiatalokat. Egy szokványos tanóra kereteit jóval túlhaladó igényeket kielégítő rendszer elkészítését tervezem.

További fontos követelmények, célok ♦ A fő célon túl további ötleteim, terveim vannak, ezeket részletezem itt.

1. Nem egy létező programozási nyelv keretein belül szeretném az algoritmizálás bemutatását megtenni. Ezzel szembeni követelményem, egy grafikus felület, *GUI*, mellyel egyedi utasításokat teljesítő algoritmusokat lehet szerkeszteni, felhasználóbarát módon.

2. Cél, hogy ne szokványos (logikai) feladatok elvégzésére készüljenek a felhasználók által összerakott algoritmusok. Az előbb említett algoritmusokra, továbbiakban az ”ágens„ kifejezést használom. Az előbbi állításból kifolyólag, azt szeretném, hogy az ágensek egymás elleni versengés, megmérettetés és játék, céljából készüljenek.

   Egy példán keresztül szemléltetve, azt szeretném, hogy egy egymás ellen játszható játékon belül minden felhasználó, egy számára kijelölt ”játékost„ irányíthasson az ágensével, és így az ágensek és a felhasználók is versengjenek, küzdjenek egymással. Az ilyen eszközöket használó oktatási formát, játékos oktatásnak [39], vagy idegen szóval a *gamification*-nek [12] nevezik.

3. Az ágensek küzdelmét kiterjesztett valóságon keresztül is figyelemmel kísérhetővé szeretném tenni. Ennek a szemléltetésére is szeretnék példát írni. Tegyük fel, hogy az ágensek küzdelme egy kitüntetett környezetben, számítógépen történik. Azon túl, hogy az adott gép kijelzőjét kilehet vetíteni, a küzdelem egy telefon és annak kamerája segítségével az íróasztal lapjára is kiterjeszthető kell, hogy legyen. Ezzel további jó hangulatot lehet teremteni.

4. Az egymás ellen küzdő ágens programozáson túl szeretném, ha a programozást biztosító grafikus felület, a da Vinci robotkar könnyű programozására is használható legyen. Úgy gondolom, hogy a programozás oktatást egyik legszemléletesebb területe, a robot programozás. Továbbá, magasszintű program utasítások biztosításának a segítségével könnyen, érthetően lehet szemléltetni a robotika és orvosinformatika területének szépségeit is, és a programozáson túl az orvostudomány, robotika iránt is lehetséges érdeklődést felkelteni a fiatalokban.

A dolgozat áttekintése és kihívások bemutatása ♦ Most egy előretekintő és a dolgozatot valamelyest összefoglaló, vázlatot, szeretnék adni. A következő példa igyekszik bemutatni, hogy, hogyan gondoltam el, az egész rendszer használatát, működését, továbbá szeretném bemutatni, hogy a különböző kihívásokat az irodalom kutatás, mely részében tárgyalom részletesen.

A rendszer konkrét működését, felosztását, architektúráját a különböző alrészekben lehet megtekinteni itt: 3 Részletes munkaterv.

Vázlatos működés felhasználó szempontjából *

1. Egy felületen csatlakozik, megad egy egyedi azonosítót, nevét, esetleg regisztrálhat is.

2. Hozzáfér a grafikus programozást nyújtó szerkesztői felülethez. Azt, hogy milyen lehetőségek vannak, ilyen grafikus programozó felület készítésére a: 2.1.1 Alternatív megoldások vizuális programozásra résznél fogom tárgyalni.

3. Attól függően, hogy da Vinci vagy játék programozási módban vagyunk, a felhasználó különböző egyedi utasításokat illeszthet össze ágenssé, például da Vinci esetén "Fogd meg az 1. objektumot„ vagy játék módban "Lépj jobbra„ , illetve alapvető ciklus és elágazás, utasításokat is tud használni.

4. Van lehetősége helyi gépen történő tesztelésre. Mindenki futtathatja, tesztelheti saját eszközén is az elkészült ágensét.

5. Miután mindenki összerakta az ágenseit, az oktató ezeket távolról összegyűjtheti és egy kijelölt gépen elindíthatja. Így például játék módban a küzdelem, valamilyen nagyobb kijelzőn mindenki számára megtekinthető. Az előbbi lépések konkrét megvalósítási terve a későbbiekben bemutatott 3.5.2 Felhasználói kliens alaklámázás különböző alrészeiben található.

6. Da Vinci mód esetén az összegyűjtött ágensekből, oktatóknak van lehetősége egyet kiválasztva futtatnia a da Vinci roboton, vagy annak szimulációjában. A da Vinci robothoz történő kapcsolódási lehetőségek irodalomkutatása megtekinthető itt: 2.2 Da Vinci.

7. Akár, mindenki telefonján elérhető, vagy egy erre szolgáló eszközön, kiterjesztett valóságon keresztül is követhető az ágensek küzdelme. A kiterjesztett valósággal kapcsolatos irodalom kutatás megtekinthető itt: 2.3.1 Alternatív kiterjesztett valóság megoldások. Igyekeztem áttekinteni a kész megoldások listáját, továbbá utánajárni, a saját készítésű kiterjesztett valóság megvalósítás hátterének.

fejezet 2

Módszertan

A korában megfogalmazott célok elérése érdekében végzett, kutatómunkát részletezem itt. Az elkészült rendszert szeretném oktatási célokra is felhasználni, ezért a megvalósítás során, olyan programkönyvtárakat igyekszem használni, melyek forráskódja, működése szabadon elérhető. Amennyire lehetséges igyekszem egyszerűbb megvalósításokat alkalmazni, értem ez alatt azt, hogy inkább olyan megoldások használatát preferálom, ami "csak„ egy adott feladatra biztosít megoldást, de a használatához nem szükséges egy nagy, bonyolult rendszert használni. Az előbbi állításomból kifolyólag, például mellőzni igyekszem a különböző felhőszolgáltatások és zárt könyvtárak, valamint az ilyen komplett óriási kódbázisok, játékmotorok használatát.

2.1. Vizuális programozási környezet

2.1.1. Alternatív megoldások vizuális programozásra

Mivel a dolgozatban 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 programozásról szól.

Scratch

Nagyon népszerű, oktatásban méltán közkedvelten használt teljes környezet [31, 38, 41], mely vizuális programozást tesz lehetővé. Az alapvető célja, média alkalmazások mint például animáció, köszöntések, történet mesélés, zeneklip elkészítésének - folyamata közben megtanítani a programozást. Az egyedüli, intuitív tanulást 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, 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 lehet az alap működésre kiható szignifikáns módosításokat eszközölni.

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 [19], 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 kezdőbarát. *Url* utasításokkal webes csatlakozó felületen keresztül tud fogadó és küldőként is könnyen kommunikálni a külvilággal. Érdekességként felhő alapú gépi tanulás szolgáltatásokhoz csatlakoztatva akár „AI" fejlesztésre is alkalmas [23]. 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 24.) béta állapotban van.
- Monolitikus, bővítésre nézve többnyire zárt rendszer.
- A objektum orientáltság különböző megkötésekkel érhető el.

GP

Általános célú blokk alapú [35] [32] 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 környezetén belül futtatható, nem lehetséges egyedi környezetben futtatni.
– Az elkészült program nem, vagy csak nehezen alakítható valódi programkóddá.
– Ugyan van webes felület, de saját oldalba hivatalosan nem ágyazható.

Alice

Interaktív 3D Animációs környezet [8]. 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 alkalmazás
– Adott 3D környezet
– Eseményvezérelt programozás tanítására is alkalmas

Hátrányai ♦
– Az elkészült program csak a saját környezetén belül futtatható, nem lehetséges egyedi 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, 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 [21] 2D játék készítő, vizuális és saját szkriptnyelvvel is rendelkezik. A teljes verzió pénzbe kerül.
– Stencyl [28] 2D Játék készítő alkalmazás melyben vizuálisan lehet programozni.

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

Blockly

Google által fejlesztett, vizuális programozást támogató programkönyvtár [6] [36]. A vizuális megjelenésért felelős, nem egy programozási nyelv, csak egy vizuális leíró nyelv, mely könnyedén exportálható vallós programkóddá. Több létező programnyelv 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.
- Programkönyvtár, 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 környezetet biztosító, nyílt forráskódú programkönyvtár [42]. A Blockly könyvtárat felhasználja blokkalapú vizuális programozás biztosítására. A blokkalapú programozáson túl biztosít lehetőséget a szöveges programozásra is, egy beépített weboldalon futamítható fejlesztő környezetben. Abban különbözik a korábbi alternatív lehetőségektől, hogy egyben nyújt egy teljes fejlesztő környezetet, amihez könnyen hozzá lehet csatolni egy saját *target* modult egyedi utasításokkal. A [13] kutatás részletesen bemutatja, hogyan lehet használni beágyazott rendszerekhez.

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.
- Bétav verzióban van a fejlesztés.

2.1.2. Választott vizuális programozási környezet

Kipróbáltam, teszteltem a legtöbb felsorolt megoldást, 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 egyedi környezethez lehessen programozni ezekben az alkalmazásokban. A Blockly kellőképpen módosítható és könnyen egyedi környezetbe illeszthető, vizuális programozást biztosító programkönyvtár, mely megfelel a feladat részéről támasztott elvárásoknak, továbbá kellőképpen egyszerűen használható.

2.2. Da Vinci

A célok között szerepelt a da Vinci robot egyszerű vezérlése. Az elkövetkező részben, igyekszem körbejárni a lehetőségeket és elemezni előnyeit hátrányait, melyek segítségével lehet kapcsolódni a da Vinci

rendszerhez. Az egyetemen egy 2010-ben bemutatott első generációs da Vinci classic rendszerhez van lehetőség hozzáférni, ezért ehhez készül az alkalmazás. A továbbiakban mindig erről a modellről lesz szó.

2.2.1. Robot Operating System

Annak érdekében, hogy a fejlesztés során, ne kelljen folyamatosan egy fizikai da Vinci rendszerrel dolgozni, egy olyan környezetre van szükség ami képes biztosítani da Vinci robotot virtuálisan. Több rendszer elérhető mely robot programozás vezérlő környezetet biztosít, csak felsorolás szintjén néhány [34, 33], azonban az egyik legkorábbi és legnépszerűbb lehetőség a Robot Operating System [37] továbbiakban ROS. A ROS órási és növekvő közösséggel rendelkezik, *de facto* standard az iparban és kutatásban. Dolgozatomban több szempont miatt is a legjobb választásnak a ROS rendszer bizonyult. Attól a hátrányától eltekintve, hogy nem multiplatform azaz csak Ubuntu GNU/Linux és annak is csak bizonyos verzióihoz támogatott hivatalosan, rengeteg előnnyel rendelkezik. Sok robotvezérléssel 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

2.2.2. Da Vinci Research Kit

A da Vinci Research Kit továbbiakban dVRK egy nyílt forráskódú rendszer, ami lehetővé teszi a Da Vinci robot karok programozását [24]. Gyakorlatilag ”firmware„-ként lehet rá tekinteni, ami képes meghajtani a robotkarokat. A ROS rendszerhez kapcsolódik, könnyen lehetséges az utasításait használni egy ROS modult biztosít használatra. A célkitűzéseimhez képest viszonylag alacsony-szintű da Vinci robotkarvezérlő utasítások kiadására alkalmas. Ennek a modulnak a segítségivel lehetséges a ROS rendszerrel da Vinci robotot vezérelni, továbbá virtuálisan szimulált robotkar létrehozása is lehetséges. mindenképp szükség van erre, amennyiben ROS rendszerhez szeretnénk kapcsolni da Vinci robotot.

2.2.3. iRob Surgical Subtask Automation Framework

Nagy D. Tamás és Haidegger Tamás által fejlesztett keretrendszer továbbiakban SAF [9]. 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 modulokat biztosít, a dVRK rendszerre épül. Használata során felépít egy hierarchikus modul rendszert, melynek a legmagasabb szintjén magas-szintű utasítások kiadása lehetséges. Számomra a céloknak tökéletesen megfelelő.

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

Hátrányai ♦
- Jelenleg fejlesztés alatt áll.

2.3. Kiterjesztett valóság

A célok között szerepelt egy olyan funkció, melynek a segítségével lehetséges kiterjesztett valóság segítésével követni az ágensek küzdelmét. Itt szeretném részletezni a hasonló megoldásokat, azokból levont következtetéseket. A [4] kutatásból idézve kiterjesztett valóságot a következő módon definiálhatjuk. Az a rendszer, ami a következőket teljesíti:
- Kombinálja a valós és virtuális objektumokat a valós környezetben.
- Azonnal fut, és valós időben.
- Egymáshoz rögzíti (igazítja) a valós és virtuális objektumokat.

Továbbikban az AR rövidítést fogom használni. Virtuális objektumokat szeretnénk elhelyezni a kamera képen úgy, hogy azok mozgása a kamera 3D mozgásához igazodjon.

Több létező megoldás van AR elkészítéséhez, ezen megoldások összehasonlításáról lesz itt szó. A legtöbb esetben nagyobb gyártók szolgáltatásaként lehetséges ilyen alkalmazásokat használni.

2.3.1. Alternatív kiterjesztett valóság megoldások

Unity

Unity Technologies által fejlesztet játékmotor. 2D és 3D számítógépes játékok elkészítésre alkalmas. Napjaink egyik, ha nem a legnépszerűbb játék fejlesztő motorja [17]. Rengeteg játékkészítéssel kapcsolatos funkciót biztosít. AR megvalósításához rendelkezik saját megoldással. Itt [25] egy példa látható arra, hogyan lehetséges a Unity beépített AR megoldását használni és milyen előnyök származnak ebből, illetve a Unity használatából. Az előbbi hivatkozott példában láthatjuk, hogy rengeteg kényelmi funkcióval rendelkezik, melyek használata, csupán kiterjesztett valóság rendszerekben, játékmotor nélkül, sokkal körülményesebb. További előnye, hogy a Unity rendszerrel készült alkalmazások az összes ma használatos platformra fordíthatóak.

Előnyei ♦
- Ingyenesen használható
- Érthető, naprakész, jó dokumentáció áll rendelkezésre
- Használata könnyű

Hátrányai ♦
- Nagyon nagy függőség
- Kifejezetten játék fejlesztésre van, rengeteg extra funkcióval rendelkezik
- Nem lehetséges csak modulként használni
- Feltétlen alkalmazkodni kell a használatához
- Nem nyílt a forráskódja

Konklúzió ♦ Úgy gondolom, hogy túlságosan nagy erőforrás és munka pazarlás lenne Unity-t csak kiterjesztett valóság elkészítésére használni. Egyáltalán nem vagy csak nagyon sok plusz munka árán lehetne alkalmazásba integrálni, inkább az alkalmazást kellene a Unity-be integrálni használata esetén.

Unreal Engine

Egy szoftver amely valósidejű 3D grafika készítésére készítettek [15]. Szűkebb értelemben játékmotornak is lehet nevezni. Szintén rendelkezik saját megoldással Virtuális valóság (VR), AR és kevert valóság (MR) számára is.

Konklúzió ♦ Többnyire ugyanazok igazak rá mint a Unity-re ezért külön nem tárgyalom. A fejlesztést alapjaiban meghatározná, a motor használatához kellene alkalmazkodni.

Mobil megoldások

iOS ARKit ♦ Az ARKit segítségével, Apple iOS platformra lehetséges AR alkalmazást készíteni [3]. A korábban említett különböző magas-szintű megoldások pl. Unity ezt a rendszert is használja, ha iOS platformra készítünk projektet. Egy AR alkalmazás számára szükséges összes funkciót megvalósítja, pl. a telefon kamera pozíció becslését, virtuális objektumok elhelyezését. Természetesen mint mobil megoldás, a telefon kameráját illetve szenzorrendrendszerét használja a működéshez.

ARCore ♦ Hasonlóan az ARKite-hez, az ARCore a modern Android operációs rendszerrel rendelkező telefonokon elérhető kiterjesztett valóságot biztosító API [2].

Konklúzió ♦ Az előbb említett megoldások használatához Andoid vagy iOS alkalmazás készítése szükséges. A legtöbb játékmotor is ezeket a megoldásokat használja mobil platformokon. Ezeknek a rendszereknek a teljes működése rejtett, egy API-t biztosítanak melyen keresztül lehetséges használni, ugyan az API-n keresztül minden kínált funkció elérhető ,a működéshez ezen túl nem lehet hozzáférni. Ezek a megoldások jól működnek, de egyedi igényeket nem lehetséges velük kiszolgálni. A működésüket nem lehet megismerni, továbbá zárt forráskódú rendszerek. Ezeket figyelembe véve, számomra nem megfelelőek.

Egyéb kész megoldások

Itt olyan megoldásokat említek, melyeket kész alkalmazásként vagy könyvtárként lehetséges használni. Felsorolás szintjén a Vuforia [44], Wikitude [46]. Ezekről általánosan elmondható, hogy használatuk részben vagy teljesen nem ingyenes. Igyekeznek saját komplett felhőszolgáltatást kényszeríteni a használathoz. Forráskódjuk ezeknek is zárt. Használatukhoz a biztosított API-t kell használni egyedi megoldások megvalósítása ezekkel is körülményes. A felsorolt hátrányok miatt számomra alapvetően nem alkalmasak használatra. Az EasyAR [14] és Kudan [27] rendszerek használata bizonyos feltételek mellet ingyenes, de forrásuk továbbra is zárt. Többnyire ezekre is igazak a fizetősekre felsorolt hátrányok.

Egyedi megoldás

Egy jó lehetőség lehet, saját kiterjesztett valóság megvalósítása, ingyenes program könyvtárak segítségével. Azt értem az ilyen megoldás alatt, hogy a készítés során alacsonyabb-szintű gépi látás szintjén, egy saját kiterjesztett valóság szolgáltatást építünk. Egy ilyen megoldásnak több nehezebb feladatot kell tudni megoldania. Ilyen esetben a kiterjesztett valóság elkészítse a feladat, nem pedig egy AR szolgáltatás használata. Elkészítése a korábbiaknál nagyobb kihívást jelent. Az elérhető "kész„ alkalmazások, könyvtárak komplett fejlesztő csapatok által, több év óta folyamatosan fejlesztés alatt vannak. Így teljesítmény szempontjából ezekkel szemben felvenni a versenyt szerintem lehetetlen, viszont modularitás és megérthetőség szempontjából úgy gondolom, hogy van értelme megpróbálni. A különböző megvalósítási lehetőségek megértése és adott feladathoz történő választása továbbra is egy előny, mert nem feltétlen kell általános megoldást készíteni.

Az egyik legnépszerűbb és legjobban elterjedt gépi látással kapcsolatos programkönyvtár az eredetileg Intel által fejlesztett, majd később nyílt forráskódú közösség által, a mai napig is folytatott OpenCV [7].

2.3.2. Választott módszer

Számomra fontos volt, hogy az AR rendszer átlátható kegyen, nem csak egy használható, de a működését megismerhető rendszert szeretnék használni. Korábban már többször említettem, hogy különböző játékmotorok használatát elvetettem, mert túlságosan alkalmazkodni kellene hozzá.

A felsorolt (többnyire) zárt könyvtárak, használata sem biztosítja a megfelelő szabadságot, továbbá előnytelennek találtam azt, hogy forrásuk zárt.

	Játékmotorok	(Részben) fizetős könyvtárak	Egyedi megoldás
Forráskód	Zárt	Zárt	Nyílt
AR Sebesség	Jól optimalizált	Jól optimalizált	A fejlesztő feladata az optimalizáció
Egyedi igények	A maga játékmotor kínál rengeteg lehetőséget, hozzáköt a motorhoz	Saját ökoszisztémát nyújt, vagy korlátozottak a lehetőségek	Teljes mértékben lehetőségünk van egyedi igények megvalósításra
Ár	Bizonyos feltételek mellet ingyenes	Bizonyos feltételek mellet ingyenes	Ingyenes

2.1. ábra. Főbb AR megközelítési lehetőségek összehasonlítása

A különböző megoldásokat összehasonlítva, a saját AR megoldás megvalósítása mellet döntöttem. Fő indokom, egy olyan rendszer készítése, melynek a működése megérthető ennélfogva, amennyire lehetséges nem támaszkodik fekete-dobozokra.

2.4. Saját AR készítéshez szükséges irodalomkutatás

Annak érdekében, hogy saját AR rendszert készítsek, igyekeztem utána járni a lehetséges módszereknek, fogalmaknak. Alapvetően abból indultam ki, hogy egy átlagos telefon kameraképeivel működjön a rendszer, így reális minőségű képekkel tudok dolgozni.

2.4.1. Kamera kalibráció

Kalibrált képen a valóságban egyenes szakaszok a képeken is így fognak megjelenni. Egy kamera rendelkezik külső és belső tulajdonságokkal.

Külső tulajdonság ♦ Így nevezzük, a 3D pozíció és orientáció együttesét. Az előbb említett összetett tulajdonság különösebb bizonyítás nélkül belátható, hogy egy transzformációs mátrix segítségével leírható.

Belső tulajdonság ♦ Összetett tulajdonság mátrixban tárolva. A kamera középpontjának koordinátáit, a pixelek méreteit tartalmazza. A kamerák képei nem tökéletesek. Tangenciális torzulás lehetséges az objektív miatt, ezt hívják halszem hatásnak is, ennek a korrigálását teszi lehetővé ezen tulajdonságok pontos meghatározása.

A kamera külső és belső paramétereinek megismerése érdekében a kamerakalibráció egy szükséges lépés a 3D gépi látáshoz. Itt [47] láthatunk, egy napjainkban is elterjedt módszer működését.

A bemutatott kalibráció előnyei ♦
- Nem igényel különösebben drága felszelést.
- A folyamat könnyen kivitelezhető a kamera, mintázat szabad kézzel tartható
- Pontosabb mint, az automatikus kalibrációs módszerek.

2.4.2. Képpontok egymásnak megfeleltetése

Egy digitális képen alapvetően színek, fényességértékek sokaságaként tárolunk, ebből szeretnénk egy adott pontot megnevezni. A következő megközelítéseket vizsgáltam meg ennek érdekében. Egyszerűbb megközelítés, a klasszikusnak számító Multiple View Geometry című könyvben is találunk [18, ó. 140.], melyben példaképp RANSAC [11] algoritmus segítségével, közelít optimális képpont megfeleltetést. Legegyszerűbben szerintem ezen a példán érthető meg.

Jellemző pontok keresése ♦ Az úgy nevezett, *wide baseline stereo* megközelítésbe tartozik. Bizonyos jellemzők *feature* alapján lehetséges jellemezni képen egy pontot. Az a cél, hogy az adott karakterisztika, tulajdonság amivel jellemeztünk egy pontot, folyamatosan minél tovább, megmaradjon az egymás utáni képeken. Így az adott ”egyedi„ pont, több képen is beazonosítható. A jellemzőt előállító algoritmust [feature descriptor]-nak hívja szakirodalom. Erről részletesebben fogok foglalkozni a következő bekezdésben.

Jellemző leírók ♦ Egy képből jellegzetes, többnyire nagy lokális kontrasztú pontokat jelölünk ki. Ezeket a pontokat fényviszony változása mellet, illetve átméretezett és átforgatott képen is szeretnénk megtalálni, ez utóbbit hívják méretezés és forgatás invarianciának.

	SIFT [29]	SURF [5]	ORB [40]
Ingyenes?	Nem	Nem	Igen
Méret invarián?	Igen	Igen	Igen
Forgatás invariáns?	Igen	Igen	Nem
Elterjedtség	Mára kevésbé népszerű	Népszerű	Népszerű
Egyéb tulajdonság	Képes a pontról 8 irányban gradiens jellemzők alapján beazonosítást végezni. Minden talált ponthoz egy orientációt is meghatároz.	A SIFT gyorsított változat., Kicsit pontatlanabb, de sokkal gyorsabb. Jól párhuzamosítható algoritmus. Közelítő módszerekkel számol, a gyorsaság édenkében	Ingyenes alternatíva a SIFT és SURF algoritmusokra. Több módszer kombinációjaként született meg.

2.2. ábra. Jellemző leírók összehasonlítása

Optikai-folyam ♦ *Optical Flow* (OF), *Small deformation* kategóriába lehet sorolni, mely alap feltételezésében eltér a korábban említett *wide baseline stereo* megközelítéstől. Az alap feltételezés, hogy nagyon kicsi két képkocka között a különbség, továbbá egy adott sugarú pixel környezet, nem feltétlen ugyanolyan, de hasonló mozgást végez. A teljes mozgás *smooth*, sima, nincsenek olyan pixelek amik a környezetüktől nagyon külön utat járnak. A modell ellentétben a jellemzőpont keresésessel, *dense* sűrű ábrázolására alkalmasak. Kötelező megemlítenem a Lucas-Kanade [30] és Horn-Schunck [20] egyszerre, igen korán 1981-ben bemutatott kutatását. Horn módszere a az összes pixelhez mozgás vektort rendel, ezért a korai időszakban nem volt valósidejű, de mára már annak lehet tekinteni. Kanade megoldása

viszont, könnyebben megérthető. Ma már alap algoritmusnak, annyira közkedvelt és alapvető, hogy videokártyák minta kódja között is lehet találkozni a különböző implementációkkal.

2.4.3. Kamera mozgásának rekonstrukciója

Miután ismerjük a kamera tulajdonságait a képekkel torzítás mentesen tudunk a továbbiakban dolgozni. A kamera mozgásának meghatározása a kamerával készült képekből. A videó feldolgozása során, szeretnénk minden képhez, egy merev-test mozgást meghatározni, melyre igaz, hogy a virtuális objektumra történő alkalmazás esetén a tanformáció után az objektum a kamera képen, megfelelő szögbe és pozícióba kerül az objektum. Az első kép esetén természetesen külön kell dönteni a virtuális objektum elhelyezkedéséről.

Simultaneous Localisation and Mapping

Meg kell különböztetni az ismert és ismeretlen *scene* "jelenet„-ben történő kamera *tracking* "követés„-t. Ismeretlen jelenet esetén, egy lehetséges kidolgozott megközelítés a "Simultaneous Localisation and Mapping„ (SLAM). Módszer arra, hogy a kialakított modell, *structure* és kapott kép információk alapján, lehetőleg minél pontosabban szeretné a kamera, merev-test mozgását meghatározni. Az egyik nagy előrehaladás a 2003-ban bemutatott sokat hivatkozott valósidejű megvalósítás [10]. A megoldás egy darab kamerát használt, *monocular* ezt SLAM-nek hívják. Az ötlet a Kálmán szűrőn alapszik [45], mely a egy optimális becslő algoritmus. Egy modell állapotából és tapasztalat által szerzett zajos állapotból igyekszik optimális állapotot meghatározni. Ennek egy továbbfejlesztett megoldására latunk példát itt [26] mely a modellt magasabb szintű részekkel bővítette ki. A SLAM módszerekről elmondható, hogy legtöbb esetben a jellemző képpontok keresését és azok egymásnak való megfeleltetésének módszereket használnak, de optikai-folyam módszer alapján is működhetnek. Robotika területén, gyakran alkalmazzák, legtöbb specifikus programkönyvtár több variánsát támogatja.

2.4.4. Saját AR, jelölő keresésén alapuló módszerrel

Az eddigiektől egészen eltérően megközelítés. A kamera mozgásának meghatározása helyett, egy előre ismert *marker*, "jelölő„ keresése a cél.

Egy előre kiszámolt hasítótáblázatból keres jelölőket a képeken. Fontos, hogy a jelölők konkrét fizikai méretének az ismeret is szükséges. Bináris négyzet alakú jelölőket keres. A [16] munkában részletesen leírják a jelölők detektálásának működését. Nagy vonalakban a következőképpen működik.

1. Szűrést véges négyzetnek tűnő alakzatokra, ezeket a jelölteknek hívjuk. Itt érdemes megemlíteni, hogy a keresett négyzetek tetszőlegesen elforgatva lehetnek a képeken.

2. Miután a jelöltek megvannak a tartalmukat bizonyos szabályossági tesztnek, idegenszóval a kijelölt négyzet belső tartalom kodifikációját kell ellenőrizni. Magyarán ellenőrizzük, hogy megfelel-e bizonyos szabályoknak, pl. mivel a jelölők mind fekete sávval vannak körbe véve, ezért lehet vizsgálni hogy a konkáv tulajdonságnak megfelelnek-e fekete sávok, vagy egyéb komplexebb tulajdonságot.

3. Az elkészült modul, élő kamera folyamon próbál keresni egy előre megadott jelölő listákból minél többet, majd megjeleníti a 3D-s elhelyezkedését a talált jelölőknek.

egy Jelző beazonosításhoz szükség van kalibrált kamerára, magyarán ismerni kell a kamera és objektív torzulás mátrixokra, mert a torzítást ki kell korrigálni a keresés elött. Ezt a feladatot a OpenCV Aruco könyvtár segítségével oldottam meg. A jelölő detektálást a könyvtár végzi. Az eredmény 6 jelölő alapú póz meghatározás ábrán is látható, a módszer valós idejű gyors,

Összehasonlítás a korábbi ötlettel ♦ A megvalósítása sokkal kisebb számítás igénnyel rendelkezik.

	AR kamera pozíció rekonstrukcióval	AR jelölő kereséssel
Kamera kalibráció	Szükséges	Szükséges
Erőforrásigény	Magas	Közepes
Használatához szükséges eszközök	Kamera	Kamera, jelölő
Komplexitás	Bonyolult	Kevésbé bonyolult
Előnyök, hátrányok	Jól tűri a hibákat, viszont jelölő nélkül nem használható.	Robusztusabb rendszer, előfordulhat drift jelenség.

2.3. ábra. Saját AR alapvető ötleteinek összehasonlítása

2.4.5. Saját AR Konklúzió

Az előbb tárgyalt elméleti módszerek, legtöbbjének legalább alapvető ismeretére, de adott esetben a teljes működésével tisztában kell lenni, saját AR készítése során. Előreláthatólag nem lesz könnyű feladat megvalósítani. Főleg a futásidőre történő optimalizáció szempontjából sokkal rosszabb helyzetből indul, mint nagy gyártók által elkészült alternatívák. Mérlegelve a feladat nagyságát és értelmét, arra döntésre jutottam, hogy ezek ellenére is saját AR rendszer elkészítésébe kezdek. Úgy gondolom, hogy megéri ezen a területen kutatni, és egy egyszerűbb, de teljesen igényeknek megfelelő AR alkalmazást készíteni, ha más nem a elkészítés során szerzett új ismeretek miatt is.

fejezet 3

Részletes munkaterv

3.1. Nevezéktan

Fiatal ♦ Azok a felhasználók, akik a célcsoportjai ennek az alkalmazásnak.

Oktató ♦ Azok a felhasználók, akik a fiatalokat koordinálják, segítik.

3.2. Felosztás

A dolgozat méretére való tekintettel, a következő fogalmak bevezetésével felosztom a dolgozatot.

Rendszer ♦ ”Rendszer„-ként fogok hivatkozni a teljes elkészült megoldásra.

Alkalmazások ♦ A rendszeren belül külön ”alkalmazásokra„ osztom a különálló, megoldandó feladatokat, melyek egymással Ethernet hálózaton vagy interneten kommunikálnak. A következő alkalmazásokra és a hozzájuk tartozó feladatokra bontottam a rendszert.

1. Felhasználói kliens alkalmazás.

   Fő feladata, biztosítani:

   - Grafikus programozási felületet
   - Oktatók számára szükséges kezelőfelületet
   - Közdelem megjelenítését

   További feladat:

   - Oktatói gépről, játék állapot adatok küldése a kiterjesztett valóság megjelenítő számára.
   - Oktatói gépről, kapcsolat kiépítése és utasítások küldése a ROS WebSocket szerviz alkalmazás felé.

2. Felhasználói szerver alkalmazás.

   Fő feladatai:

   - Autentikáció
   - Fiatalok ágenseinek tárolása
   - Az ágensek eljuttatása az oktatói gépre.

3. AR alkalmazás

   Fő feladata: Folyamatosan beérkező adatok 3D megjelenítése a kamera mozgásához igazítva.

   További feladat: kamerakalibráció elvégzése.

4. da Vinci ROS környezet. Azon ROS modulok összessége, melyek szükségesek a da Vinci, ROS keretrendszerbe történő csatlakoztatásához illetve a szimulációhoz.

5. ROS WebShocket szerviz alkalmazás

   Mediátor feladata van. Az oktatói számítógépről jövő utasítás, eljuttatja a da Vinci ROS környezet megfelelő moduljának, valamint a választ továbbítja az oktató gépre.

Modulok * Az alkalmazások belül, modulokból épülnek fel a részfeladatok megoldása mentén, egyszerű esetben egy osztály egy modult reprezentál, de vannak ennél összetettebb esetek is. Egy modul akár több másik modulra épülhet.

Layout * Főleg a rendszer *front-end* részénél van értelme ennek újabb absztrakt csoportnak a bevezetésére, több különböző modul összerakásával készíthetünk egy ilyen felületet, melyen keresztül egy adott komplex feladatot lehet ellátni, a modulok több *layout*-ban is újrahasználhatóak, így igyekszem elkerülni a duplikációkat.

3.3. Alkalmazásokhoz választott alapvető technológiák

A teljes fejlesztés alapvetően GNU/Linux operációs rendszer alól történik. Több éve használom, és minden szempontból jobb számomra mint a többi konkurens OS.

Első döntésem az volt, hogy a felhasználói alkalmazásokat, webes technológiák segítségével fogom elkészíteni. Böngészőből futtatható lesz az alkalmazás azon része, mely a felhasználókkal tartja a kapcsolatot. Így gyakorlatilag a platform különbséggel alkalmatosos problémák megszűnnek, manapság mindenhol fut a böngésző. A konkrét webes technológiát később részletezem.

Az AR alkalmazás C++ és OpenCV segítségével fog elkészülni. C++ a sebesség, optimalizáció és népszerűsége szempontjából egy jó választás. A nehézkes szintaktikája, viszont megnehezíti a használatát. Korábban már említettem az OpenCV pozitív tulajdonságait, gyakorlatilag *de facto* sztenderd gépi látással kapcsolatos programkönyvtár.

A da Vinci rendszerhez történő kapcsolathoz szintén C++ nyelvet fogok, ahsonló okok miatt elletve már választásom itt nem is lenne. Továbbá a ROS programkönyvtárat fogom még használni. A ROS ról is elmondható, hogy robotika területén *de facto* sztenderd. A ROS környezethez készült, dVRK és az ezen alapuló SAF könyvtárak használatára is szükségem lesz, a magas-szintű da Vinci utasítások kiadására.

3.4. Alkalmazások közötti kapcsolatok

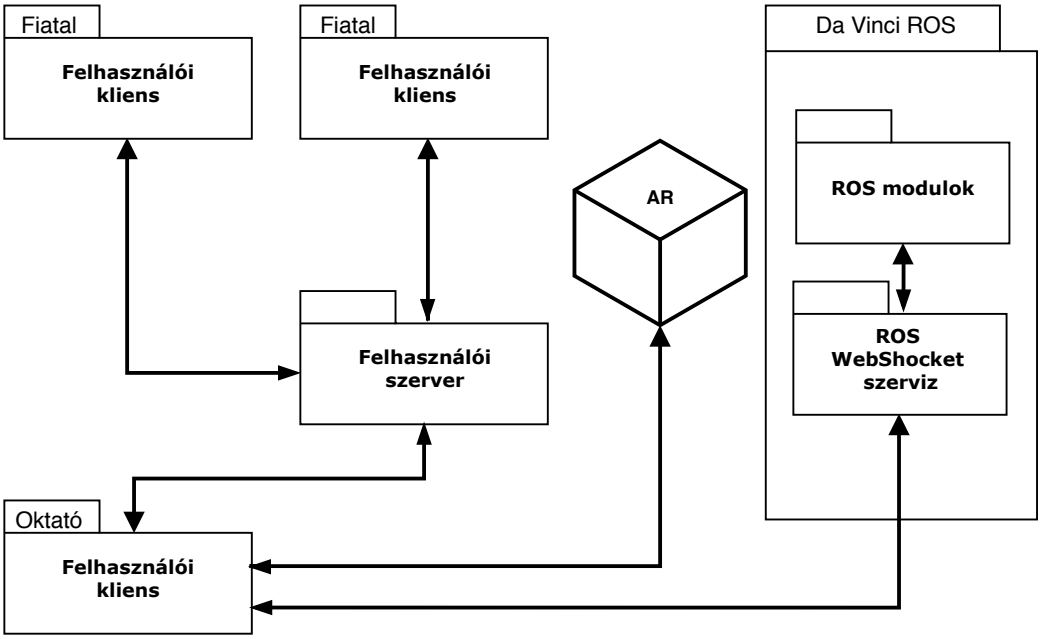
Korábban említésre került, hogy az alkalmazások Ethernet hálózaton, akár interneten keresztül kommunikálnak. A 3.4 ábrán, a rendszer felépítésének terve és alkalmazások között közötti kommunikációs kapcsolatok láthatóak.

3.5. Alkalmazások tervezett felépítése

Az egyes alkalmazások tervezésének részletei a következőkben találhatóak.

3.5.1. Felhasználói szerver alkalmazás

Korábban a webalkalmazás mellet döntöttem a felhasználói alkalmazásokkal kapcsolatban. Webes technológiákból rengeteg létezik, jobbnál-jobb újításokkal, azonban a felhasználói szerver nem annyira kulcsfontosságú része az alkalmazásnak. Gyakorlatilag bármilyen értelmes technológiával elkészülhetne, olyan kevés funkcióra lesz szükségem, hogy különösebb kutatás nélkül kiválasztottam az egyik



3.1. ábra. Alkalmazások közötti kapcsolatok

kedvencemet a NodeJS-t. Napjainkban továbbra is töretlenül népszerű technológia a NodeJS. Nem kifejezetten bonyolult webvállalkozásokhoz a használata kényelmes. Vertikálisan könnyen skálázható. Rengeteg elérhető programkönyvtár miatt pedig nagyon gyorsan lehetséges benne fejleszteni. Az objektum orientált paradigma kényelmes támogatása céljából TypeScript nyelven tervezek írni. Express és WebShocket.io és egyéb csomagok segítségével egy egyszerű, minimális szervert szeretnék készíteni.

Felhasználói szerver külső funkciók

- Regisztráció / Autentikáció
- Ágens mentése / betöltése
- Ágensek begyűjtése
- Begyűjtött ágensek kiküldése oktató felé
- Felhasználók online állapotának nyilvántartása.
- Összes online felhasználó lekérése

Fiatal felhasználók tárolt adatai

- Felhasználónév
- E-mail
- Sózott jelszó hash
- Ágens (több verzióban lehetséges tárolni)
- Iskola
- Online-e?
- Utoljára mikor volt online

Az egész szervert *production* környezetben majd egy később választott, valószínűleg Azure felhőszolgáltatásban fog futni. Egy felhőszolgáltató specifikus nem relációs adatbázisban kerülnek majd az adatok eltárolásra. Fejlesztési idő alatt MongoDB adatbázismotort fogok használni. A szerver megvalósítása közben szeretnék a REST megszorításoknak megfelelni.

3.5.2. Felhasználói kliens alaklámázás

Gyakorlatilag a *front-end* része a megvalósítani kívánt teljes rendszernek, minden felhasználó ezen keresztül használja a rendszert, kivéve a AR alkalmazást mely saját megjelenítéssel kell, hogy rendel-

kezzen. Statikus weboldal technológiával szeretném megvalósítani. A HTML oldalak, magasabb szintű kódból lesznek generálva. A kód generálásához, lehetséges a Github ajánlását a Jenkins-t használni, de terveim szerint a Vue.JS -re épülő Nuxt.JS keretrendszert fogom használni, mely szintén támogatja a statikus weboldal generálást. A Github Pages szolgáltatás ingyenesen támogatja az ilyen jellegű weboldalak kiszolgálását, így semmilyen plusz költséggel nem fog járni ennek a fenntartása. A kliens, aszinkron http hívásokkal vagy szintén aszinkron WebShocket hívásokkal, eseményekkel tud kommunikálni a szerverrel és környezetével is adott esetben.

A felhasználói kliens alaklámázás a következő fő összetett modulokból fog állni.

GameAdmin modul

Az oktató feladatinak ellátására szolgáló modul. Tipikusan ez az a modult mely biztosan az oktató számítógépnépnek böngészőjében fut. Ezen keresztül vezérelhető a rendszer viselkedése, állapota.

GameAdmin modul külső funkciók ♦

- Alkalmazáshoz csatlakozott online felhasználok kilistázása.
- Ágensek begyűjtése
- Ágensek vagy ágen betöltése a ”GameView„ modulba.
- Ágensek elindítása, egymás ellen történő párhuzamos futtatása
- Kiválasztott ágens elindítása, utasításainak da Vinci ROS környezetben történő elküldése céljából.
- Da Vinci és játék mód közötti váltás.
- Esetleges adminisztratív feladatok elvégzése
- ROS rendszer IP beállítása

GameView modul

Szintén az oktató által futtatott modul, mely nem feltétlen kell, hogy az oktató számítógépének böngészőjében fusson. Előfordulhat, hogy a nagyobb kijelzőre külön számítógép van csatlakoztatva, így azon szeretnénk futtatni ezt a modult. Ebbe a modulba töltjük az ágenseket, illetve itt van lehetőség azok futtatására. A ”GameAdmin„ modul betölti a Blockly utasításait. A Blockly utasítások JavaScript nyelvi utasításokra fordulnak ebben a modulban. . A különböző egyedi Blockly utasítások számára itt lehetséges egyedi JavaScript utasításokat definiálni. A JavaScript-re fordított ágens kódját ”JS-Interpreter„ segítségével futtatom [22]. A ”JS-Interpreter„ egy JavaScriptben írt JavaScript értelmező ”sandbox„ környezet, melyben biztonságosan van lehetőségem elválasztani az ágensek kódjait a böngésző valós JavaScript motorjától. Az ágensek kódja csak és kizárólag egyedileg definiált függvények segítségével tudnak kommunikálni a valós környezettel ezt a dokumentáció ”API hívásnak„ nevezi, jelen esetben a ”JS-Interpreter„-t könyvtárat használó fejlesztő feladata a hivatkozott ”API„ elkészítése. Az utasítások végrehajtása nem valós-időben történik, folyamatosan megvárja a utasítás végrehajtásának sikerességét.

GameView modul külső funkciók ♦

- Ágensek feltöltése
- Minden beérkezett ágens futtatása, következő ágensutasítások párhuzamos végrehajtása.
- Megjelenített állapot frissítése
- Játék újraindítása

DaVinciView modul

Szintén az oktató által futtatott modul, mely nem feltétlen kell, hogy az oktató számítógépének böngészőjében fusson. Nagy valószínűsége van annak, hogy a ROS rendszerhez történő csatlakozás külön számítógépről történik. Ekkor ezen a számítógépen kell elérni ezt a modult. Működése nagyon hasonló

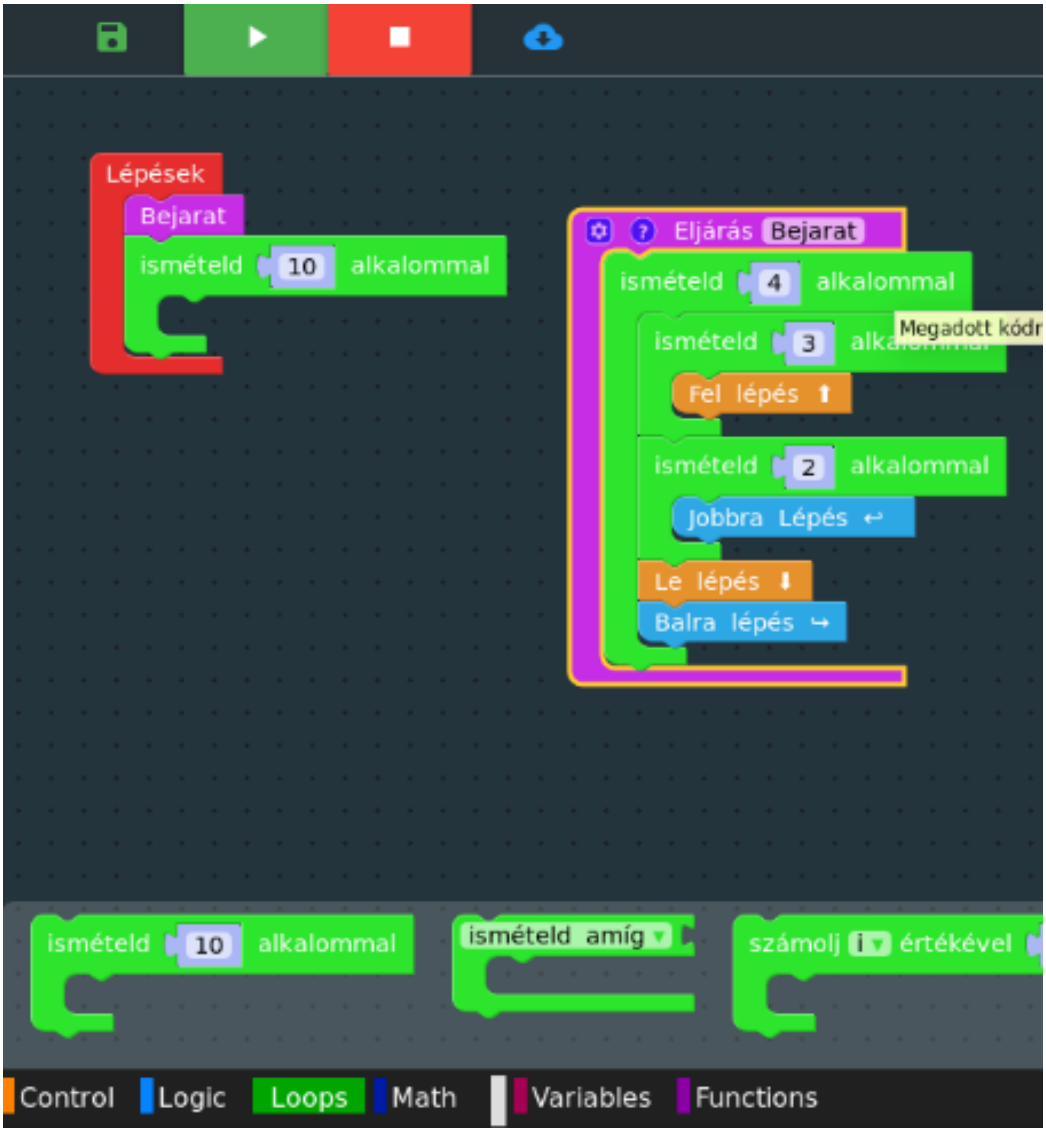
a ”GameView„-hoz a különbség annyi, hogy a futtatott utasításokat, megjelenítés helyett da Vinci ROS modul számára továbbítja.

DaVinciView modul külső funkciók ♦

- Utasítás küldés Da Vinci ROS alkalmazásnak
- Ágens kiválasztása
- Ágens futtatása, da Vinci utasítások utáni visszajelzések figyelembe vételével

BlocklyEditor modul

Célja, grafikus programozási környezet biztosítása. A korábban kiválasztott Blockly programkönyvtárat használja.



3.2. ábra. Vizuális tervet láthatunk a működésére.

BlocklyEditor modul külső funkciók ♦

3.5.3. AR alaklámázás

Az AR alaklámázás a következő fő összetett modulokból fog állni.

Kamera kalibráció modul

A kamera mátrixok meghatározására készül, több megfelelően rögzített kép lehet a bemenete, de élő módban is lehetséges használni, a kamerával folyamatosan új képek készítésével. A kimenete egy kamera mátrixokat leíró állomány.

Kamera SLAM tracker modul

Az egyik saját készítésű. Segítségével a kamera pozícióját lehet valós-időben hozzárendelni egy bemeneti képfolyamhoz.

Kamera marker tracker modul

Szinték saját készítésű. Segítségével a jelölő (*marker*) kamerához viszonyított relatív 3D pozícióját és orientációját lehetséges meghatározni.

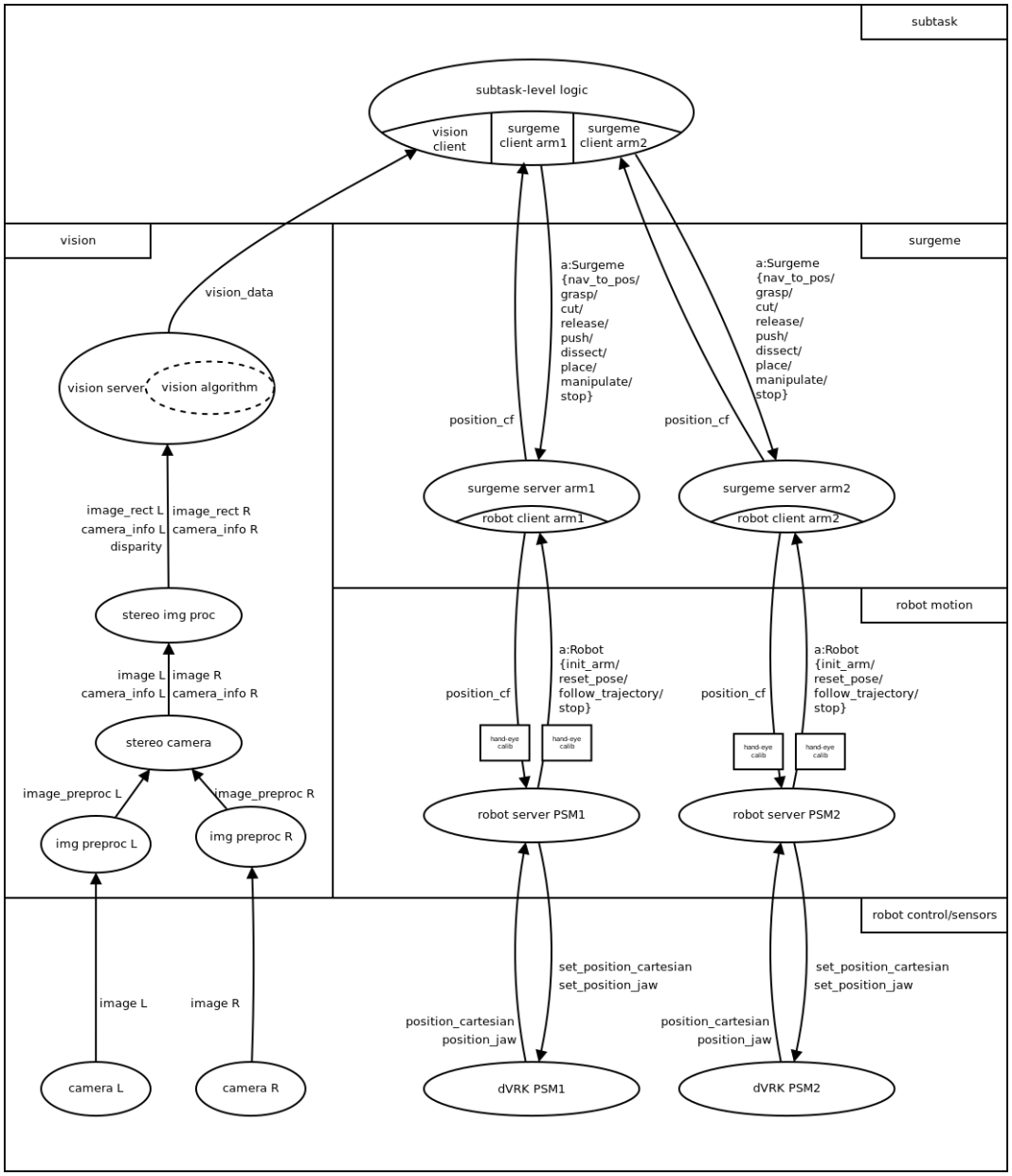
AR megjelenítő modul

Belső felépítése ♦ A Belső felépítése árbán az AR logika fő lépéseinek tervezett működése látható

3.5.4. Da Vinci ROS alkalmazás

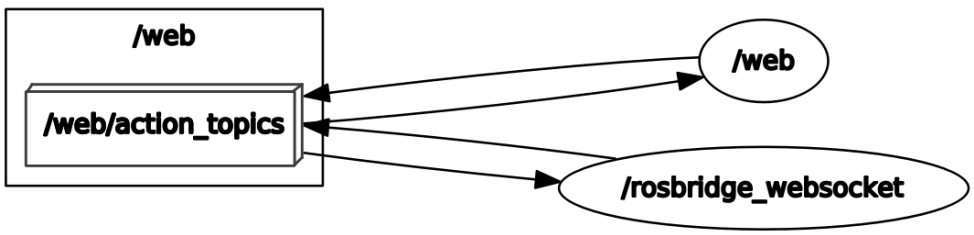
Da Vinci Web modul

A ROS környezetben sok modul foglal helyet melyek nem én készítettem hanem többnyire a SAF könyvtárból érkeznek melyet én használok az Web elnevezésű modul az egyik ROS modul amit én készítettem a következő módon kerül majd megvalósításra. Először meg kellet ismerkednem a ROS architektúra és a korábban részletezett 2.2.3 SAF keretrendszer használatával. A SAF felépítését a 3.5.4 ábra szemlélteti. A SAF ”subtask„ nevű legfelső moduljában található ”megfogás„ művelet külső meghívásának megvalósítását készítem el ezt a modult. A meglévő SAF kódból leágazva hozom létre a ROS web modult, amely ROS action kezelésére alkalmas. Az új modul a ”web„ témakörre van feliratkozva, innen vár goal definíciókat, amiket kiszolgálhat. ROS szolgáltatásként lehetséges elindítani a webes modult. Összefoglalva a következőképpen működik: a ROS web modul a következő részben ismertetett Standard ROS JavaScript könyvtár segítségével lehetséges megszólítani. A webes kérés ROS action formában érkezik. Amennyiben kérés érkezett a SAF iRob Surgical Subtask Automation Framework megfelelő modult hívja a kérésnek megfelelő utasítással. A mennyiben a kiadott utasítás sikeresen befejeződött ezt egy action alapértelmezett művelődésének megfelelően egy *callback* függvény segítségével visszajelzi a kérést kiadónak. A böngésző és a ROS közötti kommunikáció a Standard ROS JavaScript könyvtár működéséből következően WebShocket protokollon keresztül fog történik. A WebShocket protokol előnye a http-vel szemben, hogy kétirányú kapcsolat könnyű kialakítására alkalmas.



3.4. ábra. A SAF sematikus működése, felépítése [1]

A 3.5.4 ábrán láthatjuk a modul tervezett felépítését.

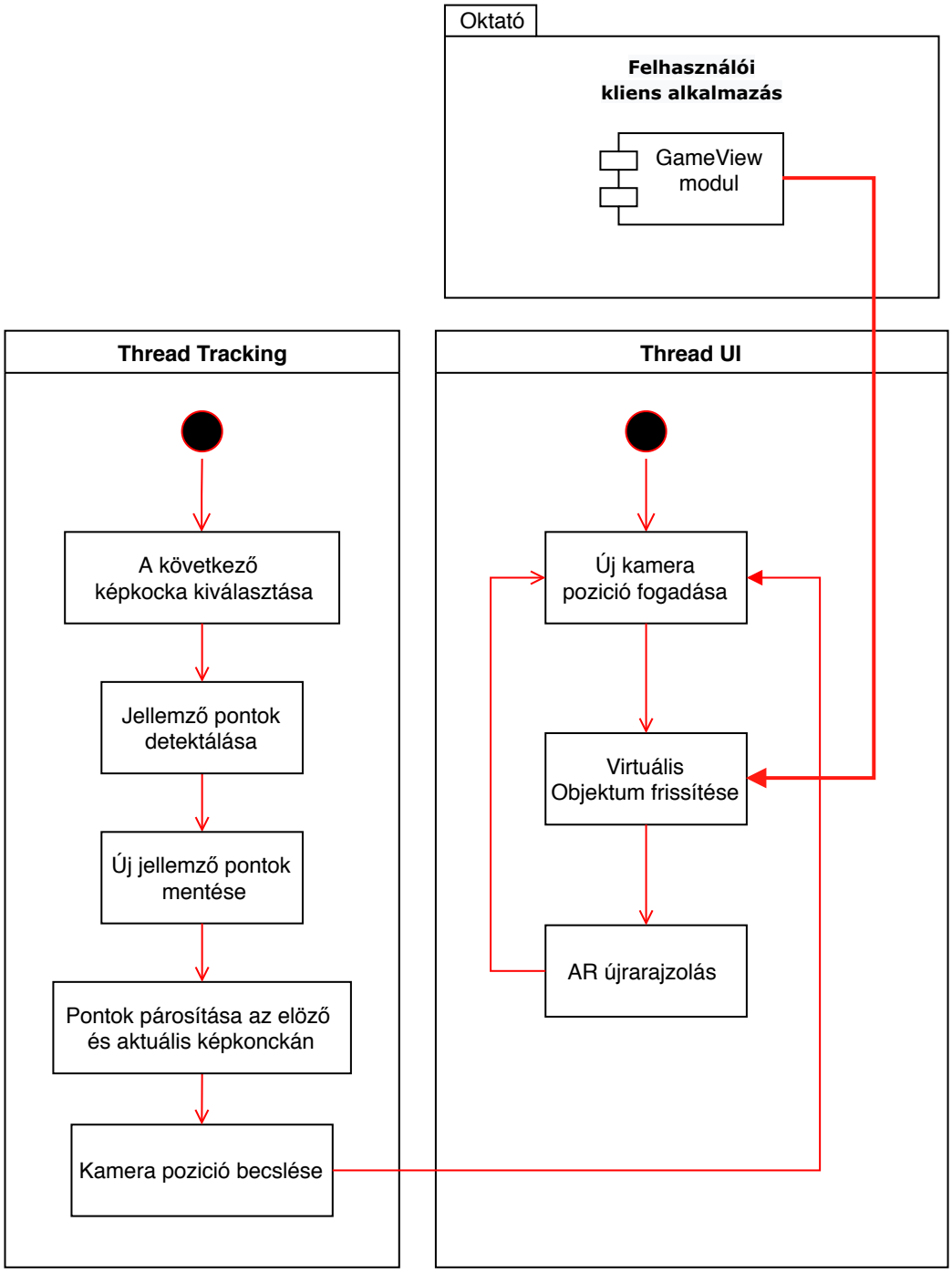


3.5. ábra. ROS web node ki/be meneteli csatlakozásai

Da Vinci WebShocket szerviz modul

Kiegészítő modul mely a konkrét csatlakozást teszi lehetővé a böngésző számára a ROS környezethez, ezt a modult nem nekem kell készítenem, csak használom a ROS környezet által nyújtott lehetőségeket. A modul működéséhez szorosan kapcsolódó programkönyvtár.

Standard ROS JavaScript könyvtár ♦ A Standard ROS JavaScript könyvtár továbbiakban roslibjs a Robot Web Tools része [43]. Alap JavaScript könyvtár, ami a ROS környezettel való böngészőből történő interakció kialakítására készítettek. A Robot Web Tools olyan eszközök fejlesztésének gyűjteménye melyek célkitűzése a ROS rendszer böngészőből történő használatához készítenek, napjainkban is folyamatosan egyre több funkciót lehetséges elérni benne. A könyvtár két részből áll. Biztosít egy JavaScript könyvtárat melyen keresztül kéréseket tudunk megfogalmazni a ROS szerver számára nevezzük ezt most JavaScript könyvtárnak. A Robot Web Tools legfontosabb része a *rosbridge* protokoll, ami a kiadott kérések fogadására van. A *rosbridge_server* egy ROS szolgáltatás amit el kell indítani, hogy továbbítsa a kéréseinket megfelelő ROS modul számára.



3.3. ábra. AR SLAM logika fő lépéseinek tervezett működése

fejezet 4

Megvalósítás

Irodalomjegyzék

[1] Abc-irobotics. *irob-saf*. [Online; accessed 18. May 2020]. 2020. máj. URL: https://github.com/ABC-iRobotics/irob-saf/blob/master/docs/irob-autosurg-blockdiagram.png.

[2] *ARCore API reference | Google Developers*. [Online; accessed 21. May 2020]. 2020. máj. 12. URL: https://developers.google.com/ar/reference.

[3] *ARKit | Apple Developer Documentation*. [Online; accessed 21. May 2020]. 2020. máj. 21. URL: https://developer.apple.com/documentation/arkit.

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

[5] Herbert Bay, Tinne Tuytelaars és Luc Van Gool. „Surf: Speeded up robust features". *European conference on computer vision*. Springer. 2006, 404–417. old.

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

[7] G. Bradski. „The OpenCV Library". *Dr. Dobb's Journal of Software Tools* (2000).

[8] 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.

[9] Tamás D. Nagy és Tamás Haidegger. „A DVRK-based Framework for Surgical Subtask Automation". *Acta Polytechnica Hungarica, Special Issue on Platforms for Medical Robotics Research* 16.8 (2019), 61–78. old.

[10] Andrew J Davison. „Real-time simultaneous localisation and mapping with a single camera". *null*. IEEE. 2003, 1403. old.

[11] Konstantinos G Derpanis. „Overview of the RANSAC Algorithm". *Image Rochester NY* 4.1 (2010), 2–3. old.

[12] 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.

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

[14] *EasyAR-Best engine for developing Augmented Reality*. [Online; accessed 22. May 2020]. 2020. máj. 15. URL: https://www.easyar.com.

[15] Epic Games. *Unreal Engine | The most powerful real-time 3D creation platform*. [Online; accessed 21. May 2020]. 2020. máj. 21. URL: https://www.unrealengine.com/en-US.

[16] Sergio Garrido-Jurado és tsai. „Automatic generation and detection of highly reliable fiducial markers under occlusion". *Pattern Recognition* 47.6 (2014), 2280–2292. old.

[17] John K Haas. „A history of the unity game engine". (2014).

[18] Richard Hartley és Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

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

[20] Berthold KP Horn és Brian G Schunck. „Determining optical flow". *Techniques and Applications of Image Understanding*. 281. köt. International Society for Optics és Photonics. 1981, 319–331. old.

[21] 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.

[22] *JS-Interpreter Documentation*. [Online; accessed 20. May 2020]. 2020. febr. URL: `https://neil.fraser.name/software/JS-Interpreter/docs.html`.

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

[24] 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.

[25] Sung Lae Kim és tsai. „Using Unity 3D to facilitate mobile augmented reality game development". *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE. 2014, 21–26. old.

[26] Georg Klein és David Murray. „Improving the agility of keyframe-based SLAM". *European conference on computer vision*. Springer. 2008, 802–815. old.

[27] *Kudan*. [Online; accessed 22. May 2020]. 2020. máj. 22. URL: `https://www.xlsoft.com/en/products/kudan`.

[28] 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.

[29] David G Lowe. „Distinctive image features from scale-invariant keypoints". *International journal of computer vision* 60.2 (2004), 91–110. old.

[30] Bruce D Lucas, Takeo Kanade és tsai. „An iterative image registration technique with an application to stereo vision". (1981).

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

[32] 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.

[33] Michael Montemerlo, Nicholas Roy és Sebastian Thrun. „Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit". *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. 3. köt. IEEE. 2003, 2436–2441. old.

[34] *MRPT – Empowering C++ development in robotics*. [Online; accessed 20. May 2020]. 2020. máj. URL: `https://www.mrpt.org`.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] Ethan Rublee és tsai. „ORB: An efficient alternative to SIFT or SURF". *2011 International conference on computer vision*. Ieee. 2011, 2564–2571. old.

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

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

[43] Russell Toris és tsai. „Robot web tools: Efficient messaging for cloud robotics". *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, 4530–4537. old.

[44] *Vuforia: Market-Leading Enterprise AR*. [Online; accessed 22. May 2020]. 2020. jan. 8. URL: `https://www.ptc.com/en/products/augmented-reality/vuforia`.

[45] Greg Welch, Gary Bishop és tsai. „An introduction to the Kalman filter". (1995).

[46] *Wikitude Augmented Reality: the World's Leading Cross-Platform AR SDK*. [Online; accessed 22. May 2020]. 2020. máj. 22. URL: `https://www.wikitude.com`.

[47] Zhengyou Zhang. „Flexible camera calibration by viewing a plane from unknown orientations". *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 1. köt. Ieee. 1999, 666–673. old.

Ábrák jegyzéke