ŠOLSKI CENTER CELJE

Srednja šola za strojništvo, mehatroniko in medije

Raziskovalna naloga

ROBOTSKI SISTEMI V OBOGATENI RESNIČNOSTI

Avtorji:
Gal HABJAN, M-4.c
Jakob ČOPER, M-4.c
Uroš UHAN, M-4.c

Mentor:
mag. Matej VEBER, univ. dipl. inž.

Celje, februar 2022

Mentor Matej Veber v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom ROBOTSKI SISTEMI V OBOGATENI RESNIČNOSTI, katere avtorji je/so Gal Habjan, Jakob Čoper in Uroš Uhan :

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, __15.4.2022 žig šole Podpis mentorja

Podpis odgovorne osebe

*

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

POVZETEK

Za to raziskovalno nalogo smo se odločili, ker menimo, da se po svetu vedno bolj širi elektronika, računalništvo in postaja vse bolj avtomatizirano ter preprostejše za uporabo. V veliko podjetjih se pojavlja poglobljena tehnologija (tj. tehnologija, ki zamenja ali poveča fizični svet), ki jo podpirajo AR (obogatena resničnost), MR (mešana resničnost) in VR (virtualna resničnost), metavesolje, 3D-zvok, računalniško prepoznavanje gibov, prostorsko zaznavanje, prepoznavanje govora, haptika, droni, 360 VR kamere in drugo. Tema raziskovalne naloge temelji na teh idejah. S pomočjo obogatene resničnosti smo tako razvili navideznega robota in raziskali uporabnost AR tehnologije v poučevanju robotskih sistemov. Tako smo v okvirju raziskovalne naloge razvili AR aplikacijo in jo testirali na eksperimentalni skupini dijakov. Omenjeno smo podkrepili s EEG merilniki in anketnimi vprašalniki. Z EEG merilnikom smo preverili kognitivne aktivnosti posameznika pri uporabi AR tehnologije, ter analizirali njihovo motivacijo za uporabo sodobnih tehnologij z anketnimi vprašalniki. Dognanja smo zapisali v zaključku.

Ključne besede: poglobljena tehnologija, obogatena resničnost, virtualna resničnost, mešana resničnost

SUMMARY

We chose the topic of our research project because we believe that electronics, computing are becoming more and more widespread around the world, and everything is becoming more automated and easy to use. Many companies are developing immersive technology (ie technology that replaces or enhances the physical world), supported by AR (augmented reality), MR (mixed reality) and VR (virtual reality), meta universe, 3D sound, computer motion recognition, spatial perception, speech recognition, haptics, drones, omnidirectional cameras... The theme of our research paper is based on these ideas. With the help of augmented reality, we have developed a virtual robot and research the usefulness of AR technology in the teaching of robotic systems. We developed an AR application and tested it on an experimental group of students. This was supported by EEG devices and survey questionnaires. Using the EEG meter, we checked an individual's cognitive activity while using AR technology and analyzed their motivation to use modern technologies with survey questionnaires. We wrote down the findings in the conclusion.

Key words: immersive tehnology, augmented reality, virtual reality, mixed reality

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL IN GRAFOV

ZAHVALA

Iskreno bi se radi zahvalili našemu mentorju, gospodu Mateju Vebru, ki nas je odlično popeljal do končane raziskovalne naloge in projekta. Pri veliko zapletih nam je pomagal in dajal rešitve problemov. Brez njega zagotovo ne bi naredili tako uspešne naloge.

Zahvaljujemo se tudi naši profesorici slovenščine, gospe Brigiti Renner, ki je našo nalogo jezikovno pregledala.

1 UVOD

Namen raziskovalne naloge je bil, da bi naredili učenje robotike zanimivejše, kot je samo gledanje seminarjev preko spleta. V obogateni resničnosti bi za izobraževalne namene prikazali model robota, ki bi lahko bil uporabljen kot navodilo, kako se robot uporablja v resničnem svetu.

Po pridobljeni ideji raziskovalne naloge smo si postavili cilje, ki smo jih želeli doseči. Naši cilji so bili:

- Prikaz modela s pomočjo »Image Target«, ki ga lahko premikamo z virtualnimi gumbi.
- Uporaba aplikacije kot navodilo uporabe s pomočjo prikaza informacijskih oken.
- Predvajanje videoposnetka kot prikaz delovanja robota.

Za izbrane cilje smo si postavili primerna raziskovalna vprašanja in hipoteze. Nato smo v programu Unity ustvarili svoj projekt. Ker smo potrebovali model navideznega robota, smo ga poiskali na spletu in ga naložili v Unity. Napisali smo še program, ki mu bo robot sledil. Vse to smo potem naložili na Android in preverili delovanje.

1.1 HIPOTEZE

Na začetku raziskovanja smo si postavili naslednje hipoteze:

1. Izdelano aplikacijo bomo lahko naložili na AR očala Vuzix Blade.
2. AR robot bo deloval kot realni robot.
3. Aplikacijo je relativno preprosto ustvariti in uporabljati v programskem okolju Unity.
4. Aplikacijo lahko uporabimo kot navodilo za delo s strojem-robotom.
5. Uporaba sodobnih tehnologij motivira uporabnike.

1.2 METODE RAZISKOVANJA

Najprej smo si ogledali podobne projekte, kjer uporabljajo tehnologijo AR in programsko okolje Unity. Tako smo ustvarili svojo AR aplikacijo, ki smo jo nadgrajevali skozi celotno nalogo. Podatke o umirjenosti in pozornosti smo dobili z meritvami možganski valov, s pomočjo ankete pa smo preverili motivacijo sodobnih AR tehnologij.

2 TEORIJA

Obogatena resničnost ali razširjena resničnost (AR) je interaktivna izkušnja okolja resničnega sveta, kjer so predmeti resničnega sveta izboljšani z računalniško ustvarjenimi informacijami. AR lahko definiramo kot sistem, ki vključuje tri osnovne značilnosti:

- kombinacijo resničnega in virtualnega sveta,
- interakcijo v realnem času,
- natančno 3D-registracijo virtualnih in resničnih objektov.

Prekrivne senzorične informacije so lahko konstruktivne (dodatek naravnemu okolju) ali destruktivne (maskiranje naravnega okolja). Izkušnja je brezhibno prepletena s fizičnim svetom, tako da je zaznana kot poglobljeni vidik resničnega okolja. Na ta način obogatena resničnost spremeni trenutno dojemanje realnega okolja, medtem ko virtualna resničnost popolnoma nadomesti uporabnikovo realno okolje s simuliranim. Razširjena resničnost je povezana z dvema večinoma sopomenskima izrazoma:

- mešana resničnost,
- računalniško posredovana resničnost.

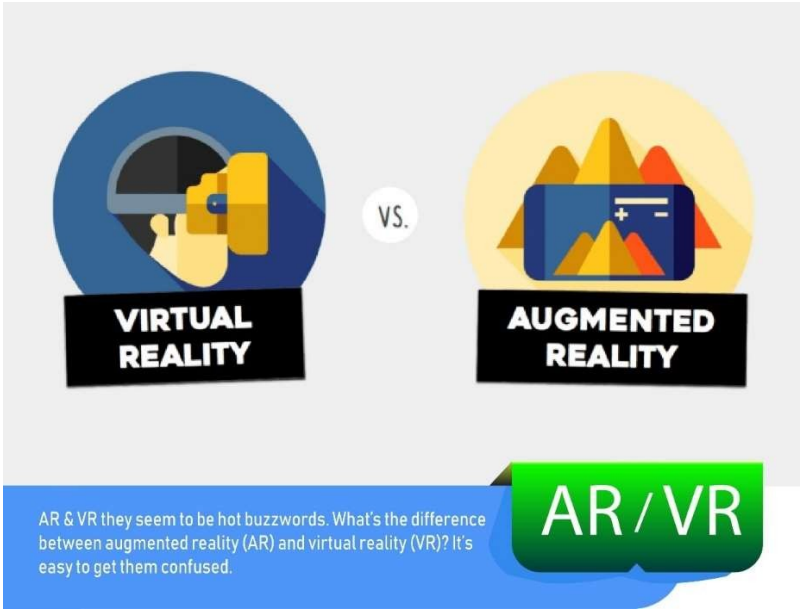
Primarna naloga razširjene resničnosti je zlivati komponente digitalnega sveta s človekovim dojemanjem resničnega sveta. Prve funkcionalne sisteme AR, ki so uporabnikom zagotavljali poglobljeno izkušnjo mešane resničnosti, so izumili v zgodnjih devetdesetih letih prejšnjega stoletja. Zasnovani so bili na podlagi sistema Virtual Fixtures, ki so ga razvili leta 1992 v laboratoriju Armstrong. Kasneje so se aplikacije AR razširile na komercialne panoge, kot so izobraževanje, komunikacija, medicina in zabava.

Obogatena resničnost se uporablja za izboljšanje naravnih okolij ali situacij in ponuja zaznavno obogateno izkušnjo. S pomočjo naprednih tehnologij AR (dodajanje računalniškega vida, vgradnja AR kamer v aplikacije za pametne telefone in prepoznavanje predmetov) postanejo informacije v realnem svetu uporabnika interaktivne in digitalno manipulirane. Informacije o okolju in njegovih predmetih se prekrivajo z resničnim svetom. Obogatena resničnost ima velik potencial tudi pri zbiranju in izmenjavi tihega znanja. Tehnike povečanja se običajno izvajajo v realnem času in pomenskih kontekstih z elementi okolja. Poglobljene zaznavne informacije se včasih kombinirajo z dodatnimi informacijami, kot so rezultati v videu športnega dogodka v

živo. To združuje prednosti tehnologije razširjene resničnosti in tehnologije »heads up display« (HUD).

2.1 PRIMERJAVA Z NAVIDEZNO RESNIČNOSTJO

V navidezni resničnosti (VR) uporabnikovo dojemanje resničnosti v celoti temelji na virtualnih informacijah. V razširjeni resničnosti (AR) pa so poleg informacij, zbranih iz resničnega življenja, uporabniku na voljo še dodatne računalniško ustvarjene informacije, ki izboljšajo njegovo dojemanje resničnosti. V arhitekturi se lahko na primer VR uporablja za ustvarjanje simulacije sprehoda skozi notranjost nove stavbe, AR pa za prikaz struktur in sistemov stavbe, zasnovanih na pogled iz resničnega življenja. Drug primer je uporaba uporabniških aplikacij. Nekatere aplikacije AR uporabnikom omogočajo uporabo digitalnih predmetov v resničnih okoljih. Podjetjem pa omogoča, da naprave za razširjeno resničnost uporabljajo kot način za predogled izdelkov v resničnem svetu.



Slika 1: Primerjava AR, VR

(Vir: OceanWP [17])

2.2 PROGRAMSKA OPREMA

Ključno merilo sistemov AR je, kako realistično povezujejo razširitve z resničnim svetom. Programska oprema mora izhajati iz koordinat realnega sveta, ki so neodvisne od kamere. Ta postopek se imenuje registracija slike in uporablja različne metode računalniškega vida, ki so večinoma povezane s sledenjem videoposnetkov.

Te metode so običajno sestavljene iz dveh delov. V prvi fazi je treba na slikah kamere zaznati interesne točke, referenčne oznake ali optični tok. V tem koraku se lahko uporabijo metode za zaznavanje značilnosti, kot so zaznavanje vogalov, zaznavanje madežev, zaznavanje robov ali pragov ter druge metode obdelave slik. V drugi fazi se iz podatkov, pridobljenih v prvi fazi, obnovi koordinatni sistem realnega sveta. Nekatere metode predpostavljajo, da so v prizoru prisotni predmeti z znano geometrijo (ali referenčni označevalci). V nekaterih od teh primerov je treba 3D-strukturo prizora izračunati vnaprej. Če informacije o geometriji prizora niso na voljo, se uporabijo metode strukture iz gibanja, kot je prilagajanje snopov. Matematične metode, uporabljene v drugi fazi, vključujejo projektivno (epipolarno) geometrijo, geometrijsko algebro, prikaz rotacije z eksponentno karto, Kalmanove filtre in filtre delcev ter nelinearno optimizacijo.

V razširjeni resničnosti razlikujemo med dvema različnima načinoma sledenja, znanima kot sledenje z označevalnikom in sledenje brez označevalnika. Označevalniki so vizualni znaki, ki sprožijo prikaz virtualnih informacij. Kamera tako prepozna geometrijo na podlagi določenih točk na risbi. Pri t. i. takojšnjem sledenju oziroma sledenju brez oznak namesto prepoznave geometrij uporabnik predmet v pogledu kamere po možnosti postavi v vodoravno ravnino. Uporablja senzorje v mobilnih napravah za natančno zaznavanje realnega okolja, kot so lokacije zidov in križišč.

ARML (Augmented Reality Markup Language) je podatkovni standard, ki ga je razvil Open Geospatial Consortium (OGC) in je sestavljen iz gramatike XML (Extensible Markup Language) za opis lokacije in videza virtualnih predmetov v prizoru ter povezav ECMAScript, ki omogočajo dinamičen dostop do lastnosti virtualnih predmetov.

Da bi omogočili hiter razvoj aplikacij za obogateno resničnost, so se pojavile nekatere aplikacije za razvoj programske opreme, kot sta Lens Studio, podjetja Snapchat, in Spark AR, podjetja Facebook, ter kompleti za razvoj programske opreme (SDK), podjetij Apple in Google.

Pri izvajanju razširjene resničnosti v potrošniških izdelkih je treba upoštevati zasnovo aplikacij in s tem povezane omejitve tehnološke platforme. Ker so sistemi AR močno odvisni od uporabnikovega vživljanja in interakcije med uporabnikom ter sistemom, lahko oblikovanje olajša sprejetje virtualnosti. Pri večini sistemov za obogateno resničnost je mogoče upoštevati podobno smernico za načrtovanje. V nadaljevanju je navedenih nekaj vidikov za oblikovanje aplikacij za obogateno resničnost.

2.3 OBLIKOVANJE OKOLJA/KONTEKSTA

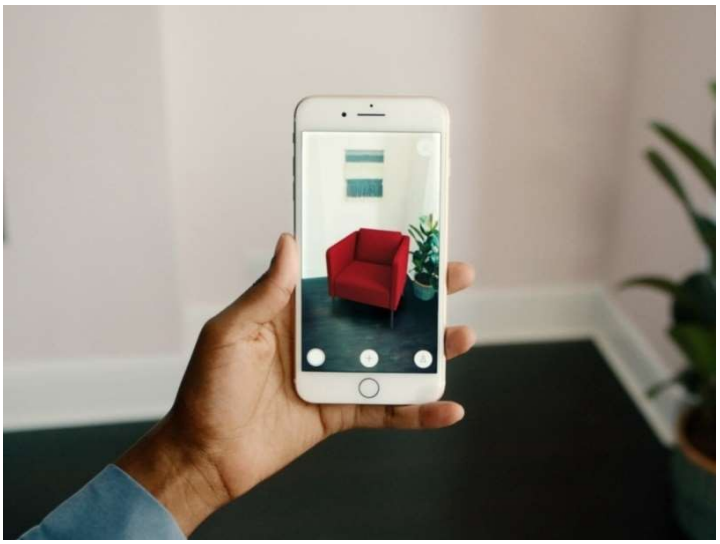
Oblikovanje konteksta se osredotoča na fizično okolje končnega uporabnika, prostor in dostopnost, ki lahko igrajo pomembno vlogo pri uporabi sistema AR. Oblikovalci se morajo zavedati možnih fizičnih scenarijev, v katerih se lahko znajde končni uporabnik, na primer:

- javni, pri katerih uporabnik za interakcijo s programsko opremo uporablja celotno telo;
- osebni, pri katerih uporabnik uporablja pametni telefon v javnem prostoru;
- intimni, pri katerih uporabnik sedi ob namizju in se ne premika;
- zasebni, pri kateri ima uporabnik na sebi nosljivo napravo.

Z oceno vsakega fizičnega scenarija se je mogoče izogniti morebitnim varnostnim tveganjem in uvesti spremembe, ki bodo izboljšale potopitev končnega uporabnika. Oblikovalci UX bodo morali opredeliti uporabniške poti za ustrezne fizične scenarije in določiti, kako se vmesnik odziva na vsakega od njih.

Pri sistemih AR je treba upoštevati tudi prostorske in okoliške elemente, ki spreminjajo učinkovitost tehnologije AR. Elementa okolja, kot sta osvetlitev in zvok, lahko senzorju naprave AR onemogočita zaznavanje potrebnih podatkov, kar uniči potopitev končnega uporabnika.

Drug vidik vključuje oblikovanje funkcionalnosti sistema in njegovo zmožnost prilagajanja uporabnikovim željam. Čeprav so pri osnovnem oblikovanju aplikacij orodja za dostopnost običajna, je treba pri oblikovanju časovno omejenih pozivov (za preprečevanje nenamernih operacij), zvočnih signalov in celotnega časa delovanja upoštevati nekatere vidike. Pomembno je opozoriti, da lahko v nekaterih primerih funkcionalnost aplikacije ovira uporabnika. Pri aplikacijah, ki se uporabljajo za vožnjo, je na primer treba zmanjšati število interakcij z uporabnikom in namesto tega uporabiti zvočne signale.



Slika 2: AR okolje

(Vir: Steantycip [1])

2.4 OBLIKOVANJE INTERAKCIJ

V tehnologiji obogatene resničnosti se oblikovanje interakcije osredotoča na uporabnikovo sodelovanje s končnim izdelkom, da bi izboljšali splošno uporabniško izkušnjo in užitek. Namen oblikovanja interakcije je, da se z organizacijo predstavljenih informacij izognemo odtujitvi ali zmedi uporabnika. Ker je interakcija z uporabnikom odvisna od njegovega vnosa, morajo oblikovalci poskrbeti, da so sistemske kontrole lažje razumljive in dostopne. Običajna tehnika za izboljšanje uporabnosti aplikacij razširjene resničnosti je odkrivanje pogosto dostopnih področij na zaslonu na dotik naprave in oblikovanje aplikacije, ki ustreza tem področjem nadzora. Prav tako je pomembno strukturirati zemljevide uporabniške poti in tok predstavljenih informacij, ki zmanjšujejo splošno kognitivno obremenitev sistema in močno izboljšujejo učno krivuljo aplikacije.

Pri oblikovanju interakcij je pomembno, da razvijalci uporabijo tehnologijo obogatene resničnosti, ki dopolnjuje funkcijo ali namen sistema. Uporaba zanimivih filtrov AR in zasnova edinstvene platforme za deljenje lahko aplikacija Snapchat uporabnikom omogoča, da povečajo svoje socialne interakcije v aplikaciji. Pri drugih aplikacijah, ki od uporabnikov zahtevajo, da razumejo ostrino in namen, lahko oblikovalci uporabijo mrežico ali žarek, ki ga oddaja naprava. Poleg tega se razvijalcem razširjene resničnosti lahko zdi primerno, da se digitalni elementi skalirajo ali odzivajo na smer kamere in kontekst zaznanih predmetov.

Tehnologija razširjene resničnosti omogoča uporabo uvedbe 3D-prostora. To pomeni, da lahko uporabnik v eni aplikaciji AR dostopa do več kopij 2D-vmesnikov.

2.5 VIZUALNO OBLIKOVANJE

Na splošno je vizualna zasnova videz aplikacije, ki se razvija in pritegne uporabnika. Za izboljšanje elementov grafičnega vmesnika in interakcije z uporabnikom lahko razvijalci uporabijo vizualne namige, s katerimi uporabnika obvestijo, kateri elementi uporabniškega vmesnika so namenjeni interakciji in kako naj z njimi komunicira. Ker je navigacija v aplikaciji AR lahko težavna in se zdi frustrirajoča, lahko oblikovanje vizualnih namigov naredi interakcije naravnejše.

V nekaterih aplikacijah razširjene resničnosti, ki kot interaktivno površino uporabljajo 2D-napravo, se 2D-nadzorno okolje ne prenese dobro v 3D-prostor, zato uporabniki ne želijo raziskovati okolice. Da bi rešili to težavo, bi morali oblikovalci uporabiti vizualne namige, ki uporabnikom pomagajo in jih spodbujajo k raziskovanju okolice.

Pri razvoju aplikacij za navidezno resničnost je treba upoštevati dva glavna predmeta v AR: 3D-volumetrične predmete, s katerimi se manipulira in ki realistično delujejo s svetlobo in senco, ter animirane medijske podobe, kot so slike in videoposnetki, ki so večinoma tradicionalni 2D-mediji, prikazani v novem kontekstu za razširjeno resničnost. Ko se virtualni predmeti projicirajo v realno okolje, je za oblikovalce aplikacij razširjene resničnosti izziv zagotoviti popolnoma brezhibno integracijo glede na realno okolje, zlasti pri 2D-predmetih. Oblikovalci lahko predmetom dodajo težo, uporabijo globinske zemljevide in izberejo različne lastnosti materialov, ki poudarjajo prisotnost predmeta v resničnem svetu. Druga vizualna zasnova, ki jo je mogoče uporabiti, je uporaba različnih tehnik osvetlitve ali metanja senc za izboljšanje splošne presoje globine. Pogosta tehnika osvetlitve je na primer postavitev vira svetlobe nad glavo v položaj ob 12. uri, s čimer se ustvarijo sence na navideznih predmetih.

2.6 IZOBRAŽEVANJE

V izobraževalnih okoljih se AR uporablja kot dopolnitev standardnega učnega načrta. Besedilo, grafika, video in zvok se lahko prekrivajo z učenčevim okoljem v realnem času. Učbeniki, kartice in drugo izobraževalno gradivo za branje lahko vsebujejo vgrajene »označevalce« ali sprožilce, ki ob skeniranju z napravo AR učencu posredujejo dodatne informacije v večpredstavnostni obliki. Na 7. mednarodni konferenci Virtual, Augmented and Mixed Reality: 7th International Conference 2015 so bila kot primer razširjene resničnosti, ki lahko nadomesti fizično učilnico, omenjena očala Google Glass. Tehnologije AR učencem pomagajo pri avtentičnem raziskovanju resničnega sveta. Virtualni predmeti, kot so besedila, videoposnetki in slike, pa so dopolnilni elementi, s katerimi učenci raziskujejo realno okolje.

Z razvojem AR lahko učenci interaktivno sodelujejo in pristneje uporabljajo znanje. Učenci lahko postanejo aktivni učenci, ki ne ostanejo pasivni prejemniki, temveč lahko sodelujejo s svojim učnim okoljem. Računalniško ustvarjene simulacije zgodovinskih dogodkov jim omogočajo raziskovanje in učenje podrobnosti o vsakem pomembnem območju kraja dogodka.

V visokošolskem izobraževanju Construct3D, sistem Studierstube, študentom omogoča učenje konceptov strojništva, matematike ali geometrije. Kemijske aplikacije AR učencem omogočajo vizualizacijo in interakcijo s prostorsko strukturo molekule s pomočjo označevalnega predmeta, ki ga držijo v roki. Drugi so uporabili brezplačno aplikacijo HP Reveal za ustvarjanje beležnic AR za preučevanje mehanizmov organske kemije ali za ustvarjanje virtualnih predstavitev uporabe laboratorijskih instrumentov. Študenti anatomije si lahko v treh dimenzijah predstavijo različne sisteme človeškega telesa. Uporaba AR kot orodja za učenje anatomskih struktur dokazano povečuje znanje učencev in zagotavlja notranje koristi, kot sta večja vključenost in potopitev učencev.

Slika 3: AR, uporabljen za navodila

(Vir: Stroka [2])

2.7 INDUSTRIJSKA PROIZVODNJA

AR se uporablja kot nadomestilo papirnatih navodil z digitalnimi, ki se prekrijejo v vidno polje proizvodnega operaterja, kar zmanjša miselni napor, potreben za upravljanje. AR omogoča učinkovito vzdrževanje strojev, saj operaterjem omogoča neposreden dostop do zgodovine vzdrževanja stroja. Virtualni priročniki pomagajo proizvajalcem, da se prilagodijo hitro spreminjajočim se zasnovanim izdelkom, saj je digitalna navodila v primerjavi s fizičnimi priročniki lažje urejati in razširjati.

Digitalna navodila povečujejo varnost upravljavca, saj mu ni treba gledati na zaslon ali v priročnik izven delovnega območja, kar je lahko nevarno. Namesto tega so navodila prekrita z delovnim področjem. Uporaba AR lahko poveča občutek varnosti operaterjev pri delu v bližini industrijskih strojev z visoko obremenitvijo, saj jim daje dodatne informacije o stanju stroja in varnostnih funkcijah ter nevarnih območjih delovnega prostora.

Ena prvih aplikacij razširjene resničnosti je bila v zdravstvu, zlasti za načrtovanje, vadbo in usposabljanje pri kirurških posegih. Že leta 1992 je bil uradno izražen cilj izboljšanja človeške zmogljivosti med operacijo, ko so v laboratorijih ameriških zračnih sil gradili prve sisteme za obogateno resničnost. Od leta 2005 se za iskanje žil uporablja naprava, imenovana infrardeči iskalnik žil, ki snema podkožne žile, obdeluje in projicira sliko žil na kožo. AR kirurgom

omogoča spremljanje podatkov o pacientu, v slogu naglavnega zaslona pilota bojnega letala, in dostop do pacientovih slikovnih zapisov, vključno s funkcionalnimi videoposnetki, ter njihovo prekrivanje. Primeri vključujejo virtualni rentgenski pogled, ki temelji na predhodni tomografiji ali na slikah v realnem času iz ultrazvoka in sond za konfokalno mikroskopijo, vizualizacijo položaja tumorja v videu endoskopa ali tveganja izpostavljenosti sevanju zaradi rentgenskih slikovnih naprav. AR lahko izboljša ogled ploda v maternici matere. Podjetja Siemens, Karl Storz in IRCAD so razvili sistem za laparoskopsko kirurgijo jeter, ki za pregledovanje tumorjev in žil pod površjem uporablja razširjeno resničnost. AR se uporablja za zdravljenje fobije pred ščurki in za zmanjševanje strahu pred pajki. Bolnike, ki nosijo očala z razširjeno resničnostjo, je mogoče spomniti na jemanje zdravil. Obogatena resničnost je lahko zelo koristna na medicinskem področju. Z njim bi lahko zdravniku ali kirurgu posredovali ključne informacije, ne da bi mu bilo treba umakniti pogled z bolnika. 30. aprila 2015 je Microsoft napovedal Microsoft HoloLens, svoj prvi poskus razširjene resničnosti. Objektiv HoloLens je z leti napredoval in je sposoben projicirati holograme za slikovno vodeno kirurgijo na podlagi bližnje infrardeče fluorescence. Z razvojem razširjene resničnosti se ta vse bolj uporablja v zdravstvu. Obogatena resničnost in podobne računalniško podprte pripomočke uporabljajo za usposabljanje zdravstvenih delavcev. V zdravstvu se lahko razširjena resničnost uporablja za usmerjanje med diagnostičnimi in terapevtskimi posegi, na primer med operacijo. Magee in drugi na primer opisujejo uporabo razširjene resničnosti za medicinsko usposabljanje pri simulaciji nameščanja igle pod ultrazvočnim vodstvom. Nedavna študija je pokazala, da tehnologija AR izboljšuje laboratorijske spretnosti univerzitetnih študentov in jim pomaga oblikovati pozitiven odnos do fizikalnega laboratorijskega dela. Nedavno se je razširjena resničnost začela uporabljati v nevrokirurgiji, ki pred posegi zahteva veliko količino slik [3], [4], [5], [12], [13].

3 UNITY

3.1 MINIMALNE ZAHTEVE

Če želimo uporabljati Unity programsko opremo, jo moramo najprej naložiti na računalnik. Pri tem moramo paziti, da jo računalnik, na katerega želimo naložiti program, podpira. Tu seveda govorimo o strojni opremi. Podatke o minimalnih strojnih zahtevah, ki jih potrebujemo, lahko najdemo na spletni strani

Unity Editor system requirements

This section lists the minimum requirements to run the Unity Editor. Actual performance and rendering quality may vary depending on the complexity of your project.

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

Slika 4: Unity Editor sistemske zahteve

(Vir: osebni arhiv)

Desktop

Operating system	Windows	Universal Windows Platform	macOS	Linux
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11	Windows 10, Xbox One, HoloLens	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	x86, x64 architecture with SSE2 instruction set support.	x86, x64 architecture with SSE2 instruction set support, ARM, ARM64.	x64 architecture with SSE2.	x64 architecture with SSE2 instruction set support.
Graphics API	DX10, DX11, DX12 capable.	DX10, DX11, DX12 capable GPUs.	Metal capable Intel and AMD GPUs	OpenGL 3.2+, Vulkan capable.
Additional requirements	Hardware vendor officially supported drivers. For development: IL2CPP scripting backend requires Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK.	Hardware vendor officially supported drivers. For development: Windows 10 (64-bit), Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK.	Apple officially supported drivers. For development: IL2CPP scripting backend requires Xcode. Targeting Apple Silicon with IL2CPP scripting backend requires macOS Catalina 10.15.4 and Xcode 12.2 or newer.	Gnome desktop environment running on top of X11 windowing system Other configuration and user environment as provided stock with the supported distribution (such as Kernel or Compositor) Nvidia and AMD GPUs using Nvidia official proprietary graphics driver or AMD Mesa graphics driver.
	For all operating systems, the Unity Player is supported on workstations, laptop or tablet form factors, running without emulation, container or compatibility layer.			

Slika 5: Unity Editor sistemske zahteve

(Vir: osebni arhiv)

3.2 UNITY ID

Če želimo uporabljati Unity, je zelo priporočljivo ali celo pomembno, da si najprej ustvarimo Unity ID račun. Ustvarimo ga lahko tako, da obiščemo uradno spletno stran za Unity, in sicer na https://unity.com/. Ko se naloži, kliknemo na profilno ikono v desnem zgornjem kotu. Nato lahko kliknemo »Create a Unity ID«, vpišemo vse potrebne podatke in si ga ustvarimo.

Slika 6: Unity »Create a Unity ID«

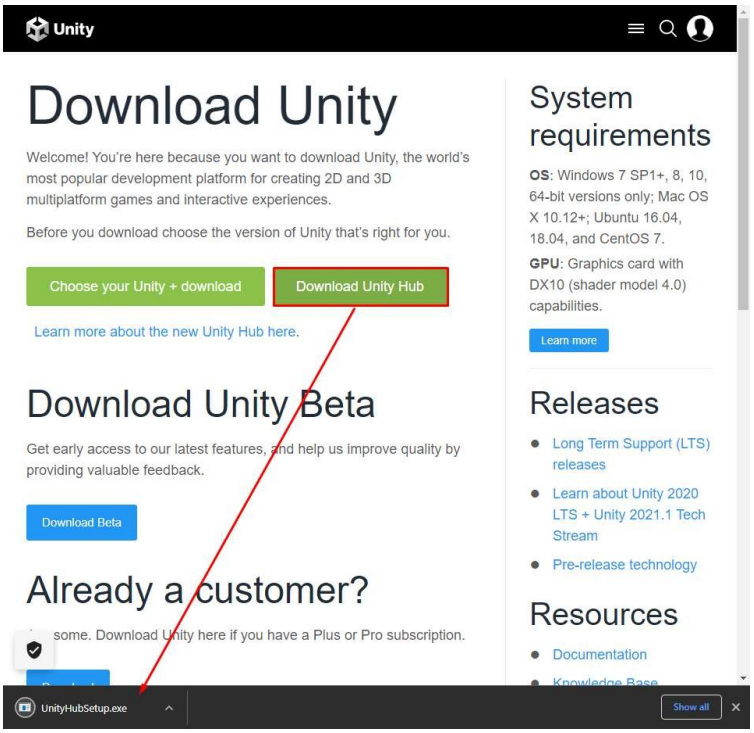
(Vir: osebni arhiv)

Če je Unity ID račun že ustvarjen, lahko namesto tega samo kliknemo »Sign in«. Če pa smo že prijavljeni, ta navodila preprosto preskočimo.

3.3 UNITY HUB

Je samostojna aplikacija, ki poenostavlja upravljanje z našimi licencami, verzijami Unity urejevalnika, projekti ... Omogoča preprosto ustvarjanje novih projektov in odpiranje že obstoječih. Seveda ima še veliko funkcij, ki nam lahko pomagajo, kot na primer vodene vaje za začetnike in Unity skupnost okno. V nadaljevanju bodo obravnavani koraki, ki jih zagotovo potrebujemo, da ustvarimo svoj AR projekt.

Če želimo naložiti Unity Hub, najprej obiščemo spletno stran https://unity3d.com/get-unity/download. Ko se naloži, kliknemo »Download Unity Hub«. Ko se UnityHubSetup.exe naloži, ga odpremo, preberemo ToS, izberemo mapo, kamor se bo naložil, in končamo.

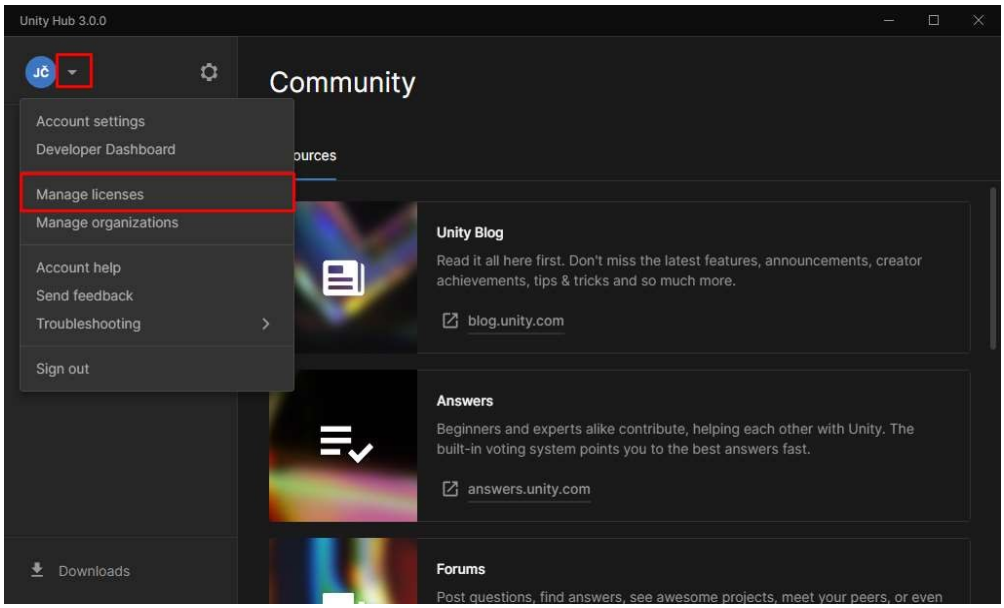


Slika 7: Nalaganje Unity

(Vir: osebni arhiv)

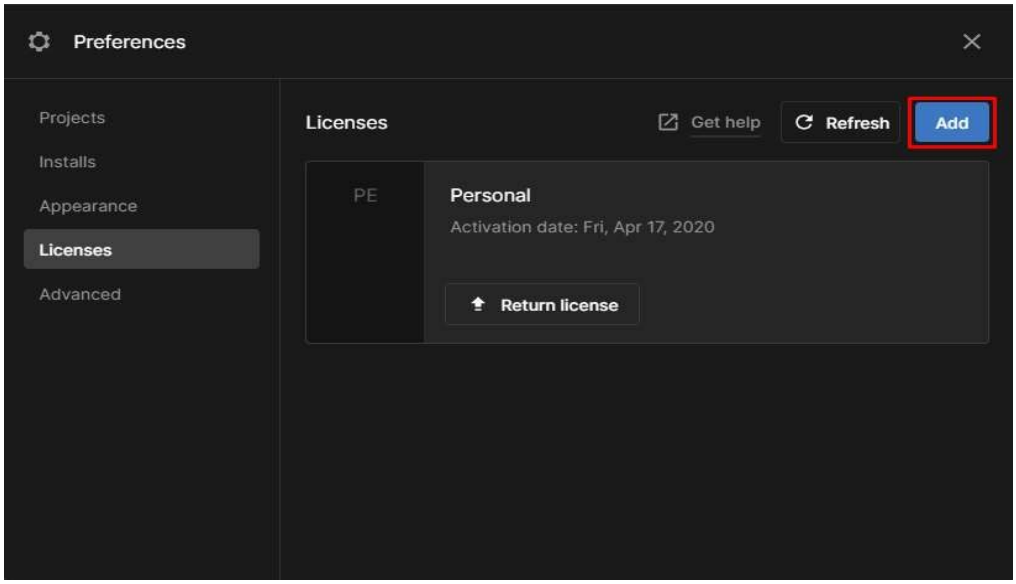
3.3.1 »License« okno

Preden lahko začnemo ustvarjati, potrebujemo dovoljenje Unity-ja. Za začetnike je najbolj priporočljiva osebna licenca, ki je brezplačna. Pridobimo jo tako, da po prijavi v Unity Hub s svojim Unity ID računom kliknemo na svojo profilno ikono, nato pa »Manage license«. To nas popelje do okna z naslovom »License«, kjer lahko upravljamo z licencami. Da dobimo osebno licenco, kliknemo »Add«, nato pa pravilno izberemo.



Slika 8: »Manage licenses«

(Vir: osebni arhiv)

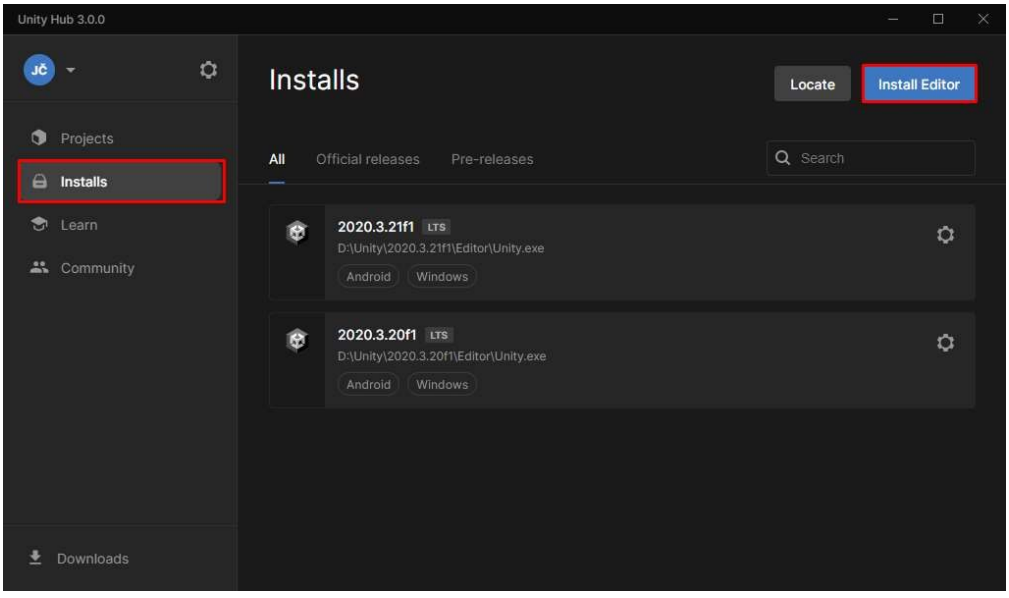


Slika 9: »Add«

(Vir: osebni arhiv)

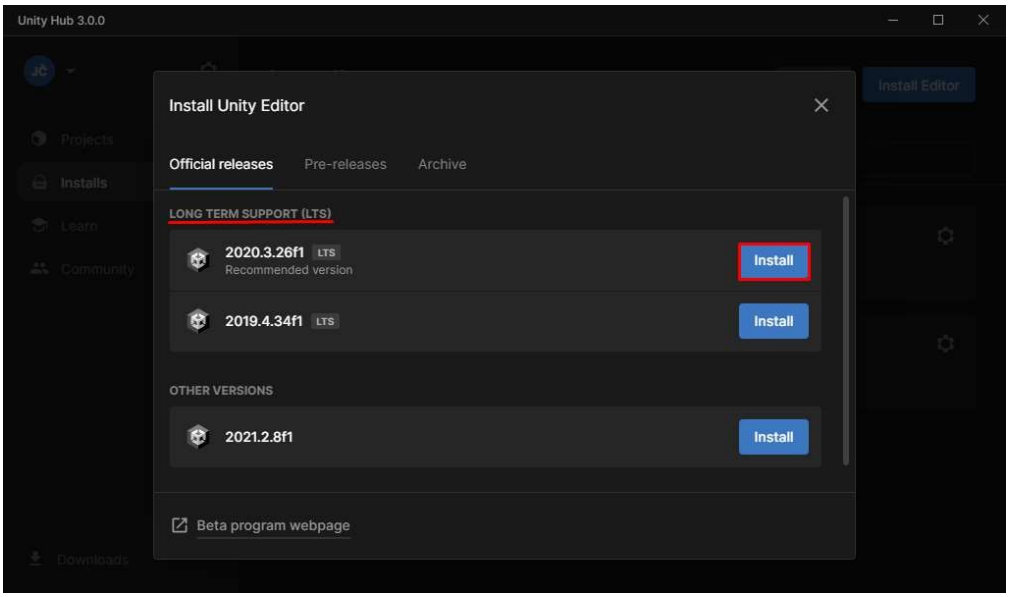
3.3.2 »Installs« okno

Tukaj lahko izberemo, katero verzijo Unity urejevalnika želimo naložiti. Vse, kar moramo narediti, je, da kliknemo »Install Editor« v desnem zgornjem kotu. Odpre se nam novo okno, v katerem lahko izbiramo med verzijami Unity urejevalnikov. Najpogosteje je »Recommended version« tudi najboljši, ker je najbolj stabilen in ima LTS, kar pomeni, da bo podpiran (s tem govorimo o popravljanju hroščev) še vsaj dve leti. Po izbrani verziji kliknemo »Install«, da se nam ponovno odpre novo okno, v katerem lahko izbiramo, ali bomo zraven naložili dokumentacijo, Visual Studio (urejevalnik za kodo), podporo za Android (če želimo ustvarjati projekte za telefon), podporo za Mac, Linux, drugačne jezike in podobne module. Ko obkljukamo vse, kar bomo potrebovali, lahko kliknemo »Install« in počakamo. Ob tem moramo biti potrpežljivi, saj se v tem celotnem postopku nalaganja Unity na računalnik katerakoli verzija Unity urejevalnika nalaga kar nekaj časa. Če želimo v prihodnosti dodati module, ni treba izbrisati naložene verzije in ponovno opraviti vse zgoraj predstavljene korake, ampak lahko samo kliknemo zobnik na desni zgornji strani kvadratka od verzije, nato pa »Add Modules«. V primeru, da želimo novejšo verzijo, pa je potrebno storiti vse znova. Pri tem je treba paziti, da ne zbrišemo stare verzije, ker je mogoče, da se nekateri projekti, ki smo jih ustvarili v tisti verziji Unity urejevalnika, ne bodo pravilno prilagodili novi verziji in bodo uničeni.



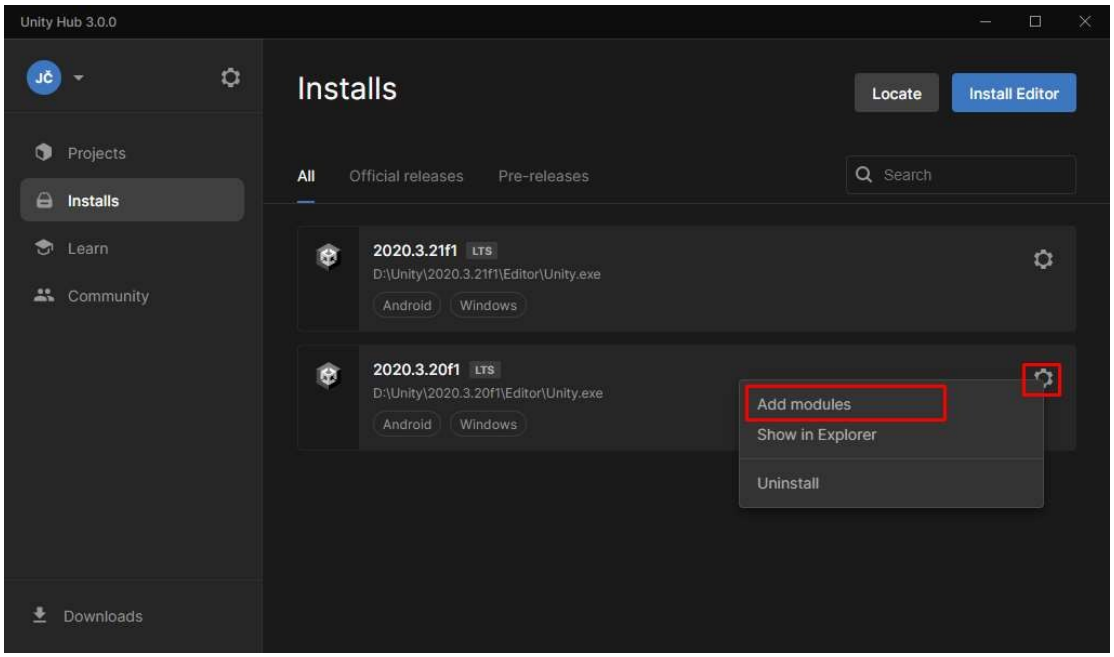
Slika 10: »Installs«

(Vir: osebni arhiv)



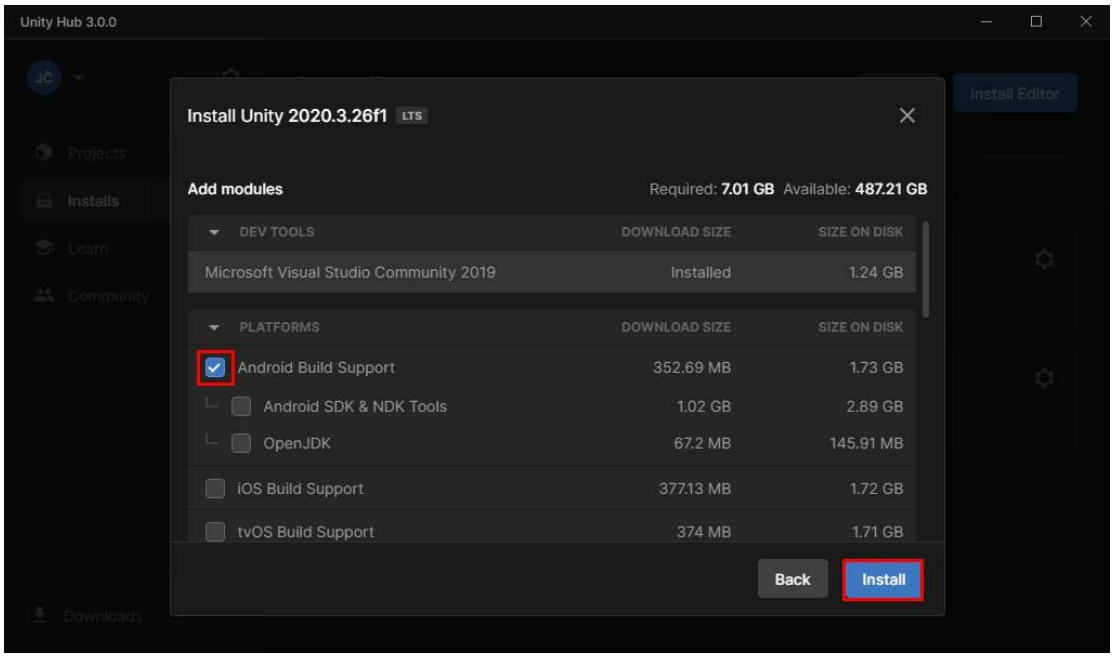
Slika 11: »Long term support«

(Vir: osebni arhiv)



Slika 12: »Add module«

(Vir: osebni arhiv)



Slika 13: »Android Build Support«

(Vir: osebni arhiv)

3.3.3 »Projects« okno

Šele ko naredimo že predstavljena koraka, lahko začnemo z ustvarjanjem projekta (o tem te Unity Hub tudi opozori). Projekte lahko ustvarimo tako, da v desnem zgornjem kotu kliknemo na moder gumb »NEW« (v novejših verzijah Unity Huba tudi »New project«), nato pa izberemo, kakšne vrste projekta bomo ustvarjali. Lahko izbiramo med 2D, 3D, HDRP (High Definition Render Pipeline) in URP (Universal Render Pipeline).

Pri projektih lahko tudi izberemo, katero verzijo Unity urejevalnika bomo uporabili, da projekt odpremo. Tu moramo biti pazljivi na potencialne okvare pri spreminjanju iz ene verzije v drugo. Največkrat celoten projekt ustvarimo in zaključimo v eni verziji Unity urejevalnika in je med ustvarjanjem (ki lahko traja tudi več let) ne spreminjamo.

3.4 UNITY EDITOR

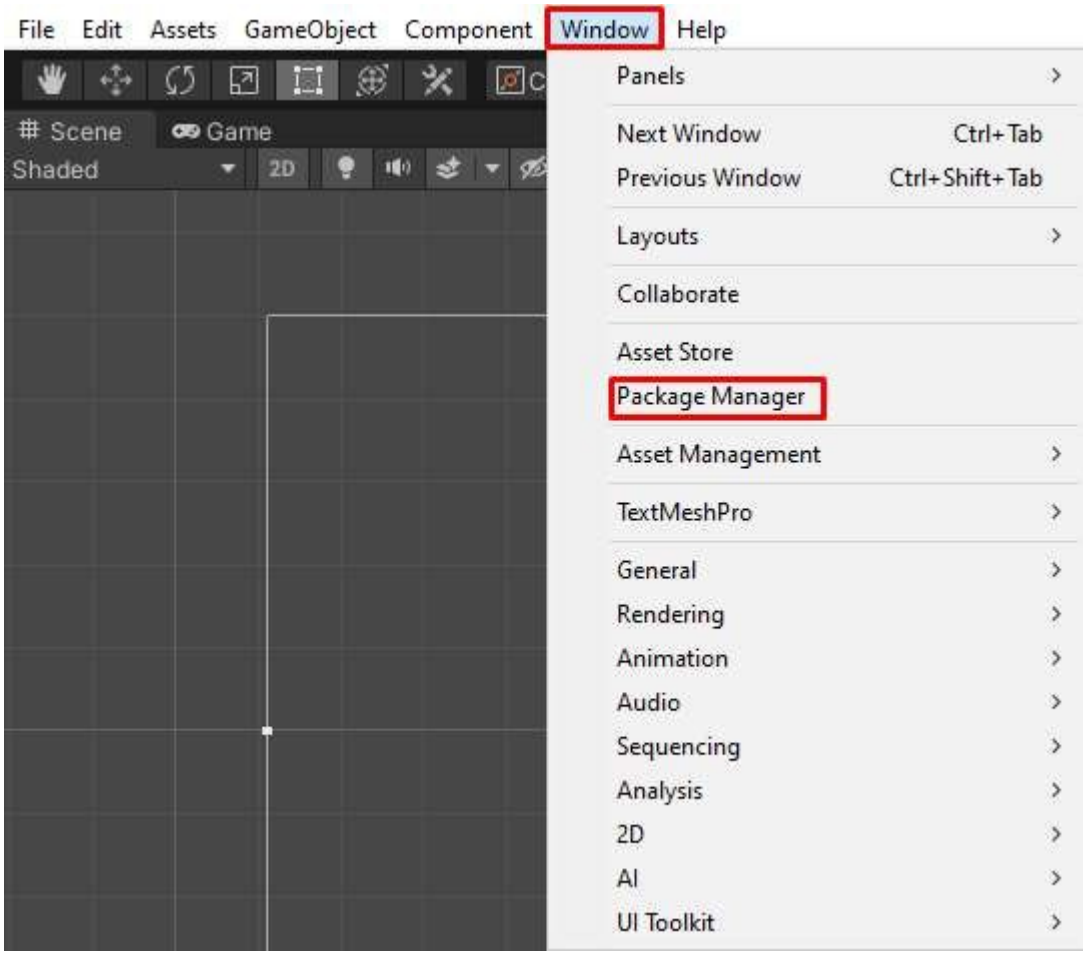
Ko ustvarimo svoj prvi projekt, se le-ta odpre v programu/aplikaciji Unity urejevalnik (Unity Editor). V tem vidimo več različnih oken, vsakega s svojo nalogo:

- Project: tu najdemo vse skripte, modele, materiale (ki jih bomo dali na modele v sceni), prefabe...
- Scene: glavno okno, v katero bomo dali vse objekte.
- Game: prikaže, kar bomo videl v igrici ali programu, ko kliknemo »Start«.
- Console: prikaže vse napake in morda kakšno besedilo, če imamo v katerem programu to vključeno.
- Hierarchy: prikaže vse, kar se nahaja v sceni, tudi nevidne stvari.
- Inspector: tu se nahaja še največja funkcija nasploh. V tem oknu nastavljamo pozicije in rotacije objektov, nanje dajemo skripte, trkalnike, toga telesa ...
- Lighting: v tem se lahko ukvarjamo z lučmi v sceni (svetlost, odboji, sence ...).
- Animator in Animation: vse povezano z animacijo stvari (na primer: kako bo človek izgledal, ko bo hodil). To je bolj uporabljeno kot olepšava celotnega projekta na koncu.

Seveda obstaja še veliko več takšnih oken, ki si jih lahko odpremo za oblikovanje svojega projekta, a zgoraj našteta so najpomembnejša in največkrat uporabljena. Če želimo odpreti dodatna okna, lahko v zgornjem levem kotu urejevalnika najdemo Window in kliknemo na okno, ki ga potrebujemo. To bomo seveda tudi storili, ker je eno od oken, ki smo jih uporabili v našem projektu, tudi Package Manager.

3.5 PACKAGE MANAGER (UPRAVITELJ PAKETOV)

Se uporablja za ogled paketov, ki so na voljo za namestitev ali so že nameščeni v našem projektu. Poleg tega lahko s tem oknom namestimo, odstranimo ali posodobimo pakete za vsak projekt na naši napravi.



Slika 14: »Package Manager«

(Vir: osebni arhiv)

Paketi so uporaben način za deljenje in ponovno uporabljanje projektov ter zbirke sredstev. Standardna Unity sredstva in stvari na Unity trgovini s sredstvi (Asset store) so zapakirana v paketih. To so zbirke datotek in podatkov iz Unity projektov, ki so stisnjena v eno datoteko, podobno kot neke vrste ZIP datoteka.

3.6 VUFORIA PAKET

Upravitelj paketov bomo uporabili, da v naš projekt namestimo paket, ki nam bo olajšal programiranje in upravljanje z obogateno resničnostjo, imenovan Vuforia. Vuforia Engine je najbolj razširjena platforma za razvoj AR (Augmented Reality = obogatena resničnost), s podporo za večino telefonov, tablic in očal. Razvijalci lahko preprosto dodajo napredne funkcije računalniškega vida v aplikacije za Android, iOS in UWP, da ustvarijo izkušnje AR, ki realistično komunicirajo s predmeti in okoljem. Vsebuje najnovejšo tehnično dokumentacijo za ustvarjanje aplikacij 10 SDK. Nenehno se posodablja z vodniki in informacijami o najnovejših funkcijah in storitvah, ki jih ponuja. Poleg tega pa je brezplačna.

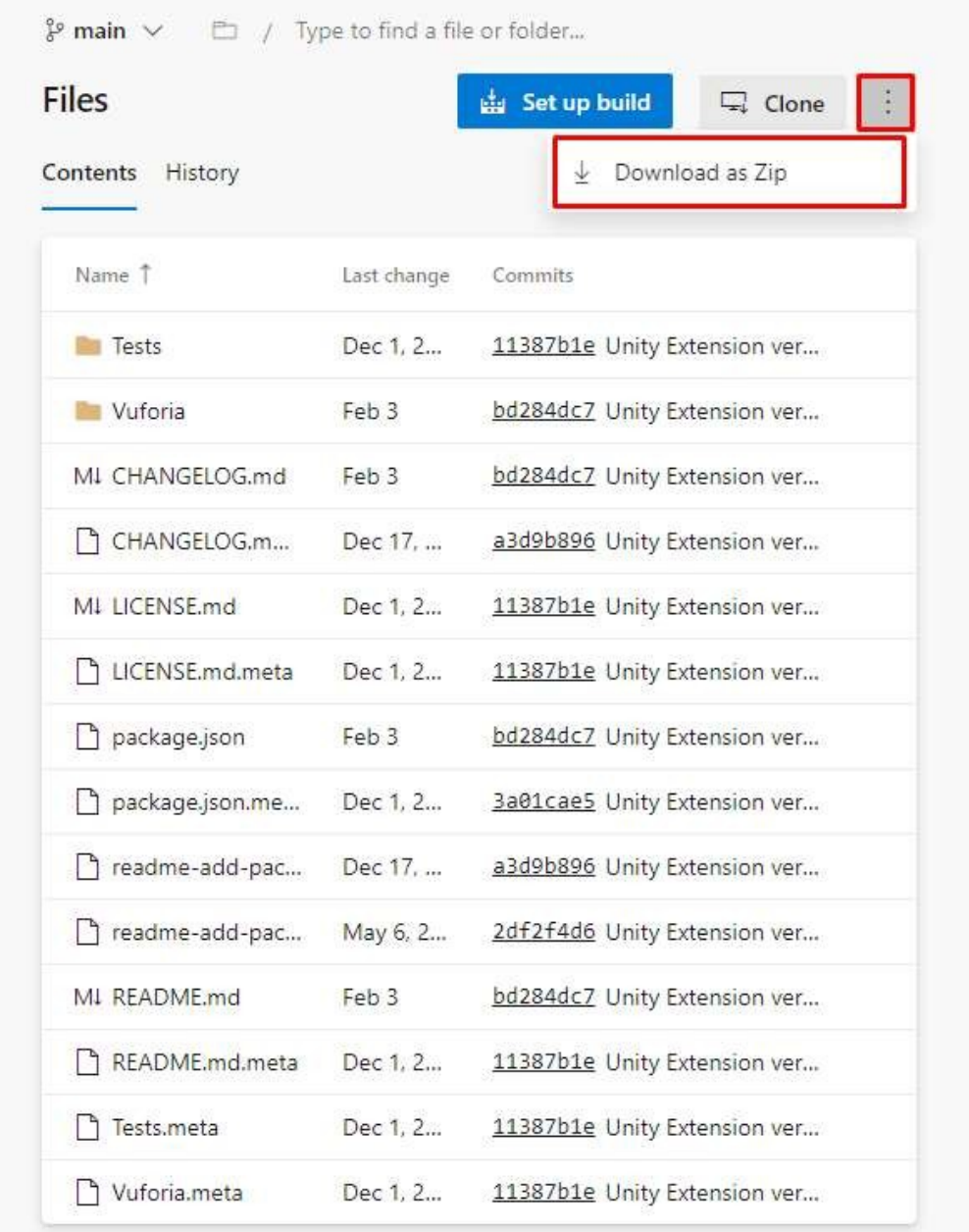
3.6.1 Naložitev Vuforie v naš projekt

Vuforia Engine lahko v Unity namestimo na tri načine:

- dodajanje paketa preko urejevalnega skripta,
- dodajanje odvisnosti v upravitelja paketov ali v manifest,
- prenos paketa in ročno dodajanje.

Za prvi dve možnosti se moramo najprej prepričati, da imamo na računalniku nameščen Git Client in da je dodan v sistemsko spremenljivko okolja PATH. To je potrebno, da Unity razreši paket Vuforia Engine Unity. Ker tega nimamo, izberemo tretji način.

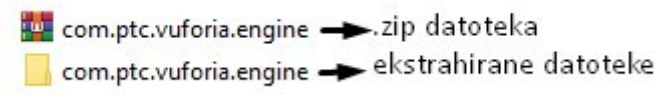
Najprej moramo naložiti Vuforia Engine paket kot .zip datoteko, kar lahko najdemo na spletni strani git-repo.developer.vuforia.com.



Slika 15: »Download as Zip«

(Vir: osebni arhiv)

Naložena datoteka se bo imenovala »*com.ptc.vuforia.engine.zip*«, iz katere moramo najprej ekstrahirati datoteke.

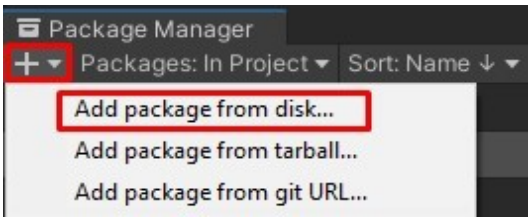


Slika 16: Datoteke

(Vir: osebni arhiv)

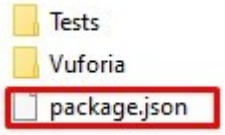
Ko to storimo, moramo odpreti projekt, v katerega želimo namestiti Vuforia Engine. V Unity urejevalniku najdemo okno Package Manager s prej omenjenim postopkom, torej Window->Package Manager.

Kliknemo »+« ikono in izberemo »Add package from disk...«, najdemo naložen paket in izberemo »*package.json*«.



Slika 17: »Add package from disk«

(Vir: osebni arhiv)



Slika 18: »package.json«

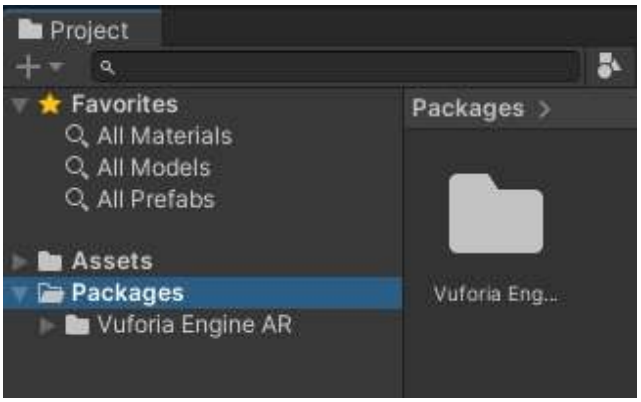
(Vir: osebni arhiv)

Če je namestitev uspela, bi na koncu morali v urejevalniku paketov videti:

Slika 19: »Vuforia Engine AR«

(Vir: osebni arhiv)

V Project oknu v Unity urejevalniku pa:

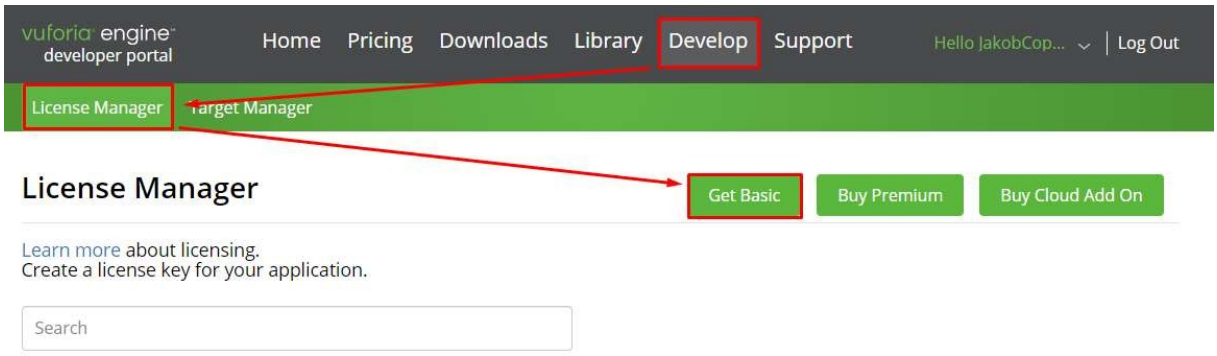


Slika 20: »Project« okno

(Vir: osebni arhiv)

Sedaj lahko pričnemo upravljati z Vuforio v Unity okolju. Za projekt AR robota bomo potrebovali tarčo, ki jo bo program prepoznal, in nanjo postavil navideznega robota. To tarčo moramo najprej ustvariti na bazi podatkov.

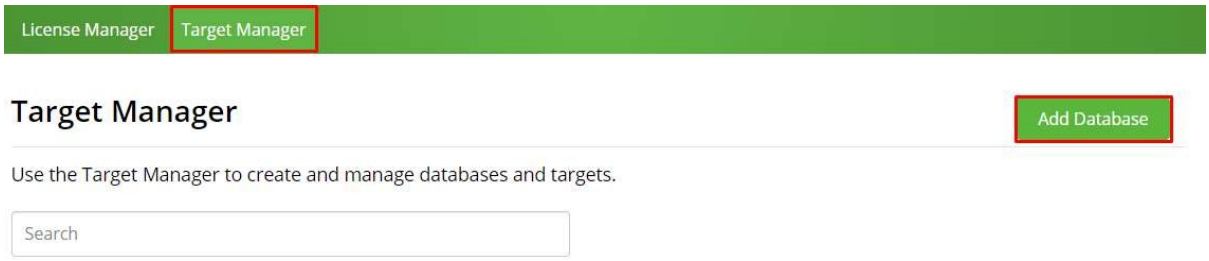
Če želimo svojo bazo podatkov, si moramo ustvariti račun. To lahko ustvarimo tako, da odpremo spletno stran https://developer.vuforia.com/vui/auth/register, vpišemo vse ustrezne podatke in se prijavimo. Nato lahko obiščemo spletno stran https://developer.vuforia.com/target-manager. Tudi tukaj bomo od ustvarjalca potrebovali dovoljenje ali licenco. Pridobimo jo tako, da na License Manager oknu izberemo pravilno vrsto, ki naj bo Basic, saj je brezplačna.



Slika 21: »Licence Manager«

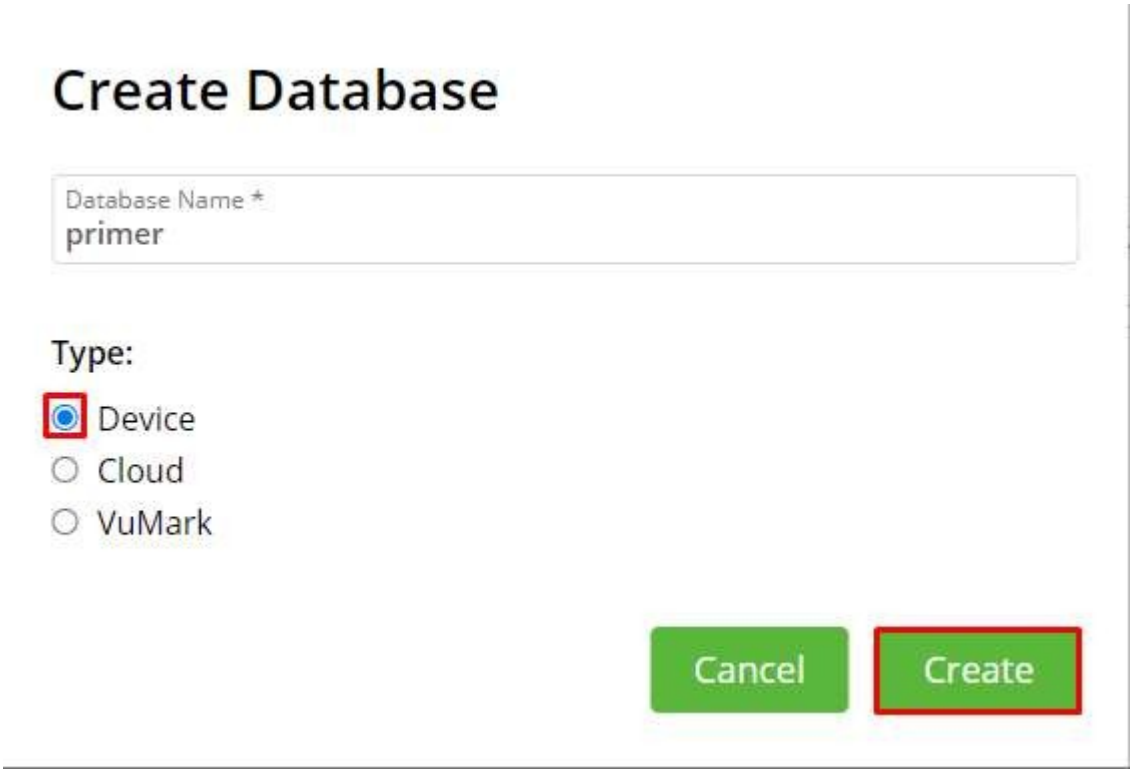
(Vir: osebni arhiv)

Po tem lahko izberemo okno »Target Manager«, kjer ustvarimo bazo podatkov tako, da kliknemo »Add Database«, vpišemo primerno ime in v pojavnem oknu poleg tega izberemo še »Device«, ki je brezplačen.



Slika 22: »Target manager«

(Vir: osebni arhiv)



Slika 23: Ustvarjanje podatkovne baze

(Vir: osebni arhiv)

V bazi podatkov lahko nato shranjujemo naše tarče in jih spreminjamo na čisto preprost način. Ko odpremo bazo, je prazna. Vanjo bomo dali slikovne tarče, na katere bomo potem lahko postavili navideznega robota. To storimo tako, da kliknemo gumb »Add Target«, izberemo pravilno vrsto tarče, podamo sliko, ki jo bo morala naprava prepoznati skozi kamero, poleg tega pa podamo še ime tarče in mere. Mere naj bodo v metrih v resničnem življenju. Torej, če je slikovna tarča široka 50 centimetrov, potem v kvadratek vpišemo »0.5«. Višino lahko računalnik izračuna sam.

Slika 24: »Add target«

(Vir: osebni arhiv)

Če je slika dobro prepoznavna in edinstvena, bo dobila večje število zvezdic v Rating stolpcu.

Slika 25: Ocena

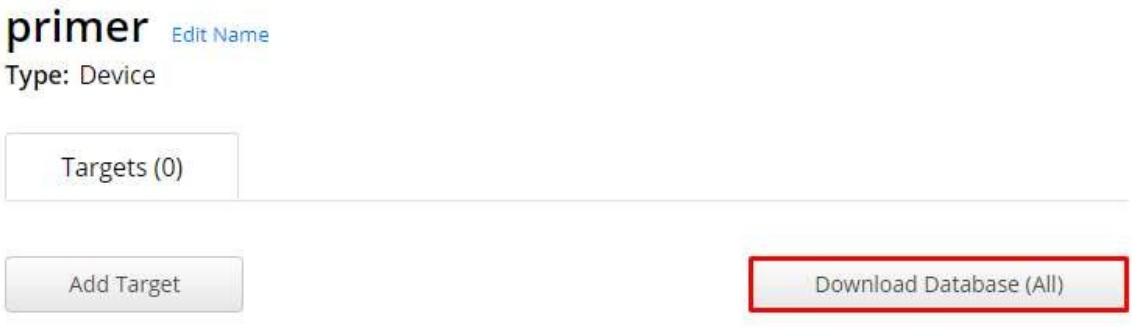
(Vir: osebni arhiv)

Prepoznavanje slik deluje na tak način, da si računalnik izbere določeno število točk na sliki, ki jo ima v bazi, in potem te točke primerja s tistimi v realni sliki.

Slika 26: Primer

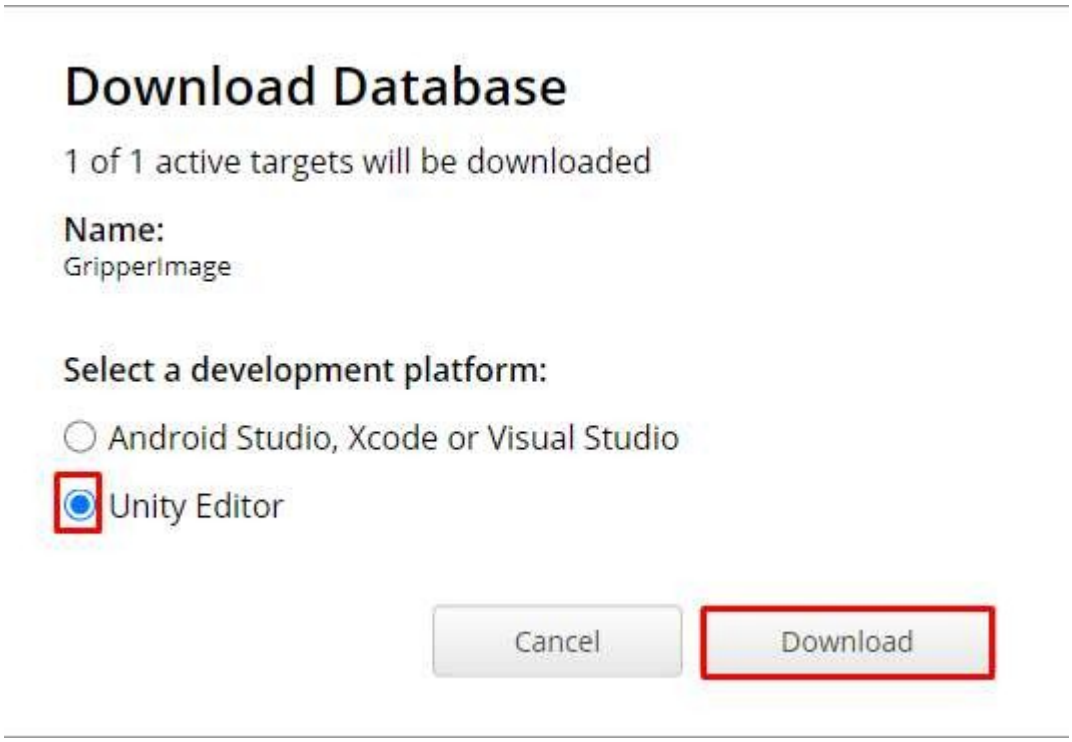
(Vir: osebni arhiv)

Ko imamo bazo in slikovne tarče, moramo te dobiti še v Unity. To storimo tako, da v naši bazi najdemo gumb »Download Database (All)«, nato pa v pojavnem oknu izberemo »Unity Editor«.



Slika 27: Nalaganje podatkovne baze

(Vir: osebni arhiv)



Slika 28: Izbira možnosti »Unity Editor«

(Vir: osebni arhiv)

Datoteko, ki jo pridobimo, lahko samo povlečemo v Project okno v Unity urejevalniku. Tako pridobimo svoje slikovne tarče v projekt [19], [21], [22].

4 PISANJE PROGRAMOV

V programu Unity je potrebno pisanje programov za delovanje aplikacije. Ti scripti se pišejo v programskem jeziku C#. Pisali smo jih v programu Visual Studio, ki ga Unity avtomatično odpre.

4.1 C SHARP PROGRAMSKI JEZIK

C sharp ali C# je programski jezik za splošno uporabo. C# se zgleduje po drugih programskih jezikih, kot so C++ in Java. Jezik je bil načrtovan z namenom, da bo čim bolj preprost in modern. Uporablja najboljše značilnosti drugih jezikov. Tako je omogočena večja optimizacija kode. C# je eden izmed najbolj priljubljenih programskih jezikov in tudi prva izbira za razvijanje iger. Je edini jezik, s katerim lahko programiramo v Unity.

Pri pisanju programa je potrebno vedeti določeno skladnjo in sintakso. V C# je potrebno konec stavka označiti s podpičjem. Stavki so združeni skupaj z zavitimi oklepaji. Spremenljivke dobijo vrednost preko enačaja, enakost spremenljivk pa preverjamo z dvojnim enačajem. Oglati oklepaji se uporabljajo pri tabelah (angl. array), pri njihovi deklaraciji in pridobivanju vrednosti na določenem indeksu.

V programu je potrebno definirati tip spremenljivk in jim nastaviti neko vrednost. Poznamo več tipov spremenljivk. Zapisali bomo tiste, ki smo jih uporabljali.

Tabela: Spremenljivke

(Vir: Wikipedija [20])

Tip	Obseg vrednosti	Velikost v pomnilniku
Int	Cela števila med –2.147.483.648 in 2.147.483.648	4 bajti
Bool	Vrednosti True in False	2 bajta
Float	Realna števila	4 bajti
string	Besede ali besedne zveze	//
GameObject	Predmet v Unity Editorju	//

4.2 VISUAL STUDIO

Visual Studio je bil narejen leta 1997. Razvili so ga v podjetju Microsoft. Visual Studio je integrirano razvojno okolje (IDE), ki je namenjeno za razvoj programov za Windows, spletne strani, spletnih aplikacij … Podpira različne programske jezike, med njimi tudi C#. Dostopen je kot brezplačni program in je idealen za razvijanje iger ali AR aplikacij v Unity Enginu.

4.3 PISANJE PROGRAMOV

Ko se Visual Studio odpre, dobimo predlogo, ki nam pomaga pri pisanju programa. Na začetku nam že napiše »using«, kar pomeni, da program pridobi razred iz tega »Namespaca«, kot na primer:

»using UnityEngine;«

To nam pridobi vse razrede, ki jih največkrat uporabljamo v Unity, na primer: gameObject …

Nato zapišemo ime tega programa, kar je v Visual Studio narejeno samodejno.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class testProgram : MonoBehaviour
{
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
}
```

Slika 29: Primer programa

(Vir: osebni arhiv)

Najprej je zapisan »public class«. To pomeni, da je ta program dostopen drugje, kot na primer v Unity urejevalniku ali v drugih programih.

Ime programa si izmislimo sami. Nastavljeno je samodejno, ko naredimo program v Unity urejevalniku.

»MonoBehaviour« je osnovni razred, iz katerega izhajajo vsi Unity programi.

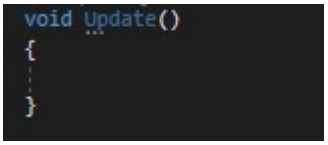
Nato sledijo zaviti oklepaji, ki označujejo vse, kar je v tem razredu. Vse, kar je izven tega, ne bo upoštevano.

Sledita dve največkrat uporabljeni funkciji, ki jih uporabljamo v Unity. »Void Start()« in »void Update()«.

Slika 30: Start funkcija

(Vir: osebni arhiv)

Programska funkcija »void Start« je sestavljena iz dveh delov. Prvi del je »void«, kar pomeni, da nam funkcija ne poda nobene uporabne vrednosti, kot so na primer »integers«. »Start« je že določena funkcija v »Unity Enginu« in pomeni, da se koda med zavitima oklepajema izvrši le enkrat.



Slika 31: »Update« funkcija

(Vir: osebni arhiv)

»Void Update()« je podobno kot »void Start()«, ampak se koda med zavitima oklepajema izvrši večkrat v zanki.

4.3.1 Osnove programiranja za Unity Engine

Z uporabo using UnityEngine dobimo dostop razredov, ki so največkrat uporabljeni v Unity Enginu. To nam omogoča dostop do funkcij, kot sta »void Start« in »void Update« ter nove spremenljivke, kot so: »GameObject«.

4.3.2 Spremenljivke

V C# moramo definirati spremenljivke (int, float, GameObject..) med začetkom razreda in funkcije »Start«.

```
public GameObject predmet;
private int celoStevilo = 10;
```

Slika 32: Primer spremenljivk

(Vir: osebni arhiv)

Definiranje spremenljivk je sestavljeno iz treh ali štirih delov, najprej zapišemo »public« ali »private«. »Public« nam omogoča, da dostopamo do te spremenljivke v drugih programih ali v Unity urejevalniku, medtem ko je »private« dostopen le v tem programu. Za tem je tip spremenljivke, ki se uporabljajo za shranjevanje informacij. Za tipom sledi ime spremenljivke, ki je lahko katerakoli beseda (mora biti le ena). Če ima beseda velike in majhne črke, bo v Unity urejevalniku zapisanih več besed. Pri definiranju spremenljivk je možno tudi zapisati njeno vrednost, kar pa ni potrebno, saj lahko to informacijo nastavimo v Unity Editor ali pa v programu. Če ne nastavimo te informacije, bo vzelo privzeto vrednost, na primer »0« ali »false«.

4.3.3 Funkcije

Pri pisanju programa je uporabno pisanje funkcij, saj nam omogočajo enostavnejši in preglednejši program. Če nam funkcija ne daje neke uporabne informacije, kot je »bool« ali »int«, uporabimo »void«.

```
public void ImeFunkcije()
{
    celoStevilo = 9;
}
0 references
private int DrugaFunkcija(int i)
{
    return i++;
}
```

Slika 33: Primer funkcij

(Vir: osebni arhiv)

Podobno kot pri definiranju spremenljivk uporabljamo »public« in »private«. »Public« nam omogoča, da dostopamo do te funkcije v drugih programih ali Unity urejevalniku v primeru pritiska na gumb. Ime funkcije je lahko karkoli. Na koncu mora imeti oklepaj, v katerem je lahko zapisan dejavnik te funkcije. To je odvisno od uporabe funkcij.

Funkcije uporabljamo v »void Start« ali »Update«.

```
void Update()
{
    ImeFunkcije();
    celoStevilo = DrugaFunkcija(10);

}
```

Slika 34: Primer uporabe funkcij

(Vir: osebni arhiv)

V tem primeru bo v zanki izvršilo obe funkciji, prva spremeni vrednost spremenljivke »celoStevilo« v 9, druga bo vzela vrednost 10 in ji prištela 1 ter to vrednost izpisala, kar se zapiše v spremenljivko »celoStevilo«.

4.4 PROGRAM ZA PREMIKANJE OSI Z VIRTUALNIMI GUMBI

Ta program smo napisali za krmiljenje navideznega robota v obogateni resničnosti s pomočjo virtualnih gumbov. Program naj bi preveril, če je bil virtualni gumb pritisnjen ali izpuščen, kar bo krmililo osi robota. Ta program lahko krmili tri osi, zato ga kopiramo v Unity urejevalniku dvakrat, saj uporabljamo dva »Image Targeta«, vsak s šestimi gumbi. Dva gumba na os za krmiljenje v dveh smereh.

V programu smo uporabili vse, kar smo zapisali v prejšnjem poglavju. Uporabljali smo »using UnityEngine«, saj smo potrebovali razrede, ki niso privzeti v C#. Najprej smo dodali »using Vuforia;«, saj potrebujemo razrede, ki niso dodani v UnityEngine. Nato smo definirali spremenljivke, ki so »public«, saj jih želimo spreminjati v Unity Editor. Definirali smo »GameObject«, torej dele robota, ki jih želimo vrteti. To so tri osi, zato smo jih poimenovali »Axis1« … Potem definiramo virtualne gumbe, ki jih nameravamo uporabiti za premikanje robota. Uporabili smo besedo »VirtualButtonBehaviour«, ki ga omogoča Vuforia. To naredimo za vsak virtualni gumb. Sledi definiranje vektorjev, ki so »public«, saj jim nastavimo vrednost v Unity Editor. »Vector3« je vektor v treh smereh x, y, z in mogoča premik ali rotacijo. Poimenoval smo jih »rotA1«, »rotA2« in tako naprej, saj predstavljajo rotacijo prve, druge in tretje osi. Zadnja spremenljivka je »directionA1«. Uporabili smo »private«, saj te spremenljivke ne potrebujemo nikjer drugje. Spremenljivka je tip »int« oziroma »integer«, kar pomeni celo število. V tem primeru bo vrednost med –1, 0 ali 1, kar predstavlja smer, v katero se premika. Na primer: če je vrednost »directionA1« enaka 1, se bo vrtela os 1 v levo smer, če bo vrednost –1, se bo vrtela v desno, če bo vrednost 0, bo mirovala.

Po definiranju spremenljivk sledi funkcija »Start()«. Tu smo definirali virtualne gumbe glede na to, ali se aktivira njihova funkcija glede na pritisk ali izpust gumba.

```
VbAxis1L.RegisterOnButtonPressed(rotateA1L);
VbAxis1R.RegisterOnButtonPressed(rotateA1R);
VbAxis1L.RegisterOnButtonReleased(stopA1L);
VbAxis1R.RegisterOnButtonReleased(stopA1R);
```

Slika 35: Definiranje virtualnih gumbov

(Vir: osebni arhiv)

Najprej smo zapisali spremenljivko, ki smo jo definirali na začetku, nato sledi funkcija »RegisterOnButtonPressed()«, ki nam omogoča, da se bo izvršila funkcija »rotateA1L«, ki smo jo napisali drugje.


```
VbAxis1L.RegisterOnButtonPressed(rotateA1L);
VbAxis1R.RegisterOnButtonPressed(rotateA1R);
VbAxis1L.RegisterOnButtonReleased(stopA1L);
VbAxis1R.RegisterOnButtonReleased(stopA1R);
```


Slika 36: Definiranje virtualnih gumbov

(Vir: osebni arhiv)

Podobno storimo pri definiranju funkcije pri izpustu gumba. Uporabili smo isto spremenljivko, saj definiramo za isti gumb. Spremeni pa se funkcija v »RegisterOnButtonReleased()« in ime »stopA1L« funkcije, ki smo jo napisali drugje.

To smo ponavljali, dokler ni imel vsak gumb funkcije za pritisk na gumb in za izpust gumba.

Sledi pisanje teh funkcij, ki smo jih definirali v »Start()«.

```
public void rotateA1L(VirtualButtonBehaviour vb)
{
    directionA1 = 1;
    Debug.Log("pressed");
}
```

Slika 37: Funkcija na virtualnem gumbu

(Vir: osebni arhiv)

V »Startu« smo definirali, kdaj se ta funkcija izvrši. Torej ko bo virtualni gumb 1 pritisnjen, se bo vrednost spremenljivke »directionA1« nastavila na 1 in bo v konzoli izpisalo "pressed". To smo storili za vsako funkcijo, torej 12-krat. Spremeni se le »direction« glede na smer, v katero želimo, da se premika.

Na koncu dodamo še funkcijo »Update()«, kjer se preračuna kot tega predmeta.

```
void Update()
{
    Axis1.transform.Rotate(directionA1 * rotA1 * Time.deltaTime);
    Axis2.transform.Rotate(directionA2 * rotA2 * Time.deltaTime);
    Axis3.transform.Rotate(directionA3 * rotA3 * Time.deltaTime);
}
```

Slika 38: »Update« funkcija za premikanje osi

(Vir: osebni arhiv)

Najprej smo napisali ime »GameObjecta«, ki mu želimo spremeniti rotacijo. To storimo s funkcijo »GameObject.transform.Rotate()«. To je funkcija v Unity Engine, ki nam omogoča premik predmeta v virtualnem svetu oziroma njegovo rotacijo. Z »directionA1« določimo smer, v katero se vrti, »rotA1« je vektor, ki določa, za koliko se bo zavrtela os v eni sekundi. Da zagotovimo, da se zavrti enakomerno in pride do določene vrednosti v eni sekundi, uporabimo »Time.deltaTime«. To nam poda čas v sekundah med zadnjo sličico in trenutno, omogoča pa nam realni čas in enakomerno premikanje ne glede na število sličic. [15]

4.5 PROGRAM ZA PREMIKANJE OSI BREZ VIRTUALNIH GUMBOV

Če želimo premakniti model robota brez virtualnih gumbov, so za to potrebni UI elementi, kot so drsniki. Drsnik je element, ki ga omogoča Unity. To je UI element, s katerim lahko spreminjamo vrednost med dvema določenima številoma (na primer od –180 do 180). Torej lahko s šestimi drsniki spreminjamo rotacijo vseh osi. Ti drsniki bodo ob začetku programa skriti, ampak jih bo možno uporabljati, če tako določimo v nastavitvah.

Ob začetku pisanja programa je potrebno vstaviti »using UnityEngine.UI«, kar nam omogoča dostop do UI funkcij v Unity. To so na primer drsniki, besedilo … Potem je treba definirati spremenljivke. Najprej smo definirali spremenljivke tipa »GameObject«, ki smo jih poimenovali »Axis«. Nato smo definirali zbirko decimalnih števil in jih poimenovali »rot«. Na koncu smo definirali tudi zbirko drsnikov. Tip spremenljivke se imenuje »Slider« in ga omogoča »UnityEngine.UI«.

V tem primeru je celoten program napisan v funkciji »Start()«. Najprej izberemo drsnik, s katerega želimo brati vrednost, na primer: »sliders[0]«. Uporabimo funkcijo »onValueChanged.AddListener((v) =>«, kar pomeni, da bo drsnik pošiljal informacijo o njegovi vrednosti vsakič, ko se le-ta spremeni. Ob spremembi vrednosti se izvrši funkcija in si zapiše vrednost drsnika. Ta vrednost je nastavljena med –180 do 180, saj smo tako nastavili v Unity urejevalniku. Nato spremenimo rotacijo predmeta »Axis1« tako, da nastavimo

»transform.localEulerAngles«, enako »new Vector3(0,0, rot[0])«. Vektor je enako nič na oseh x in y, saj se os 1 ne vrti okrog njih, vendar se vrti okoli osi z, zato je nastavljena »rot[]« namesto osi z. Vsi predmeti se ne vrtijo okoli iste osi, zato je treba paziti, kam postavimo spremenljivko »rot[]«.

Rotacija Eulerjevih kotov z enoto stopinje je relativna na »parent objectove« rotacije. Eulerjevi koti imajo tridimenzionalno rotacijo, saj se rotirajo okoli osi x, y in z.

»Vector3« je tridimenzionalni vektor, ki mu podamo smer in vrednost. Torej, v katero smer se bo vrtel oziroma premikal in kako hitro [9].

```
void Start()
{
    sliders[0].onValueChanged.AddListener((v) =>
    {
        rot[0] = v;
        Axis1.transform.localEulerAngles = new Vector3(0, 0, rot[0]);
    });
    sliders[1].onValueChanged.AddListener((v) =>
    {
        rot[1] = v;
        Axis2.transform.localEulerAngles = new Vector3(0, rot[1], 0);
    });
    sliders[2].onValueChanged.AddListener((v) =>
    {
        rot[2] = v;
        Axis3.transform.localEulerAngles = new Vector3(0, rot[2], 0);
    });
    sliders[3].onValueChanged.AddListener((v) =>
    {
        rot[3] = v;
        Axis4.transform.localEulerAngles = new Vector3(rot[3], 0, 0);
    });
    sliders[4].onValueChanged.AddListener((v) =>
    {
        rot[4] = v;
        Axis5.transform.localEulerAngles = new Vector3(0, rot[4], 0);
    });
    sliders[5].onValueChanged.AddListener((v) =>
    {
        rot[5] = v;
        Axis6.transform.localEulerAngles = new Vector3(rot[5], 0, 0);
    });

}
```

Slika 39: Definiranje drsnikov v funkciji »Start«

(Vir: osebni arhiv)

4.6 PROGRAM ZA ODPIRANJE IN SPREMINJANJE NASTAVITEV

Pri uporabi programa je treba dostopati do različnih gumbov, kot so gumb za ročno premikanje robota z drsniki, gumb za premik na glavni meni, določitev in shranjevanje pozicije osi robota in zakasnitev slik. Vseh teh gumbov ne moremo imeti na zaslonu, saj na njem ni dovolj prostora, zato uporabimo posebni prostor, ki smo ga imenovali nastavitve. S pritiskom na gumb preklopi na nastavitve, ob ponovnem pritisku na isti gumb pa jih izklopi.

V programu smo najprej definirali spremenljivki »set« in »manual«, ki sta tip »GameObject« in sta »public«. Spremenljivka set predstavlja prazen »GameObject«, pod katerim so vsi gumbi, ki se morajo prikazati ob pritisku na gumb. V programu definiramo dve funkciji.

```
public void toggle_settings(bool val)
{
    set.SetActive(val);
}
```

Slika 40: Funkcija za »Toggle«

(Vir: osebni arhiv)

Najprej smo izbrali ime in nastavili tip spremenljivke »val«, ki je »bool«, kar pomeni, da ima lahko le dve vrednosti (»true«, »false«). Ob pritisku na gumb se bo vrednost »val« spremenila iz »false« na »true«, kar bo aktiviralo »GameObject set«. Ob ponovnem pritisku se bo set deaktiviral. Enako naredimo za preklop ročnih kontrol.

»Empty GameObject« je predmet v Unity Editor, ki nima oblike. Po navadi nanj damo določene programe ali pa ga uporabljamo kot »parent object«. V tem primeru smo ga uporabili kot »parent object«, saj je lažje deaktivirati en predmet kot pa vse posebej [9].

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
Unity Script (3 asset references) | 0 references
public class settings : MonoBehaviour
{
    public GameObject set;
    public GameObject manual;

    0 references
    public void toggle_settings(bool val)
    {
        set.SetActive(val);
    }
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
    0 references
    public void toggle_manual(bool val)
    {
        manual.SetActive(val);
    }
}
```

Slika 41: Celoten program za nastavitve

(Vir: osebni arhiv)

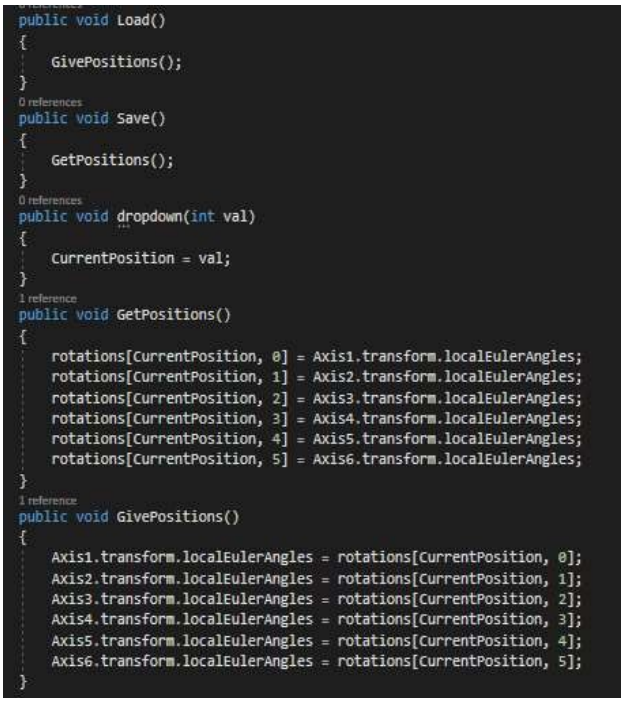
4.7 PROGRAM ZA SHRANJEVANJE IN PRIDOBIVANJE POZICIJ ROBOTA

Ta program nam omogoča shranjevati do deset pozicij robota, ki jih lahko shranimo v nastavitvah in kasneje ponovno pridobimo.

V programu moramo najprej napisati »using TMPro;«, kar pomeni »TextMeshPro«. To nam omogoča uporaba funkcij in razredov za obdelovanje besedila, ki niso v Unity Engine. Na začetku programa definiramo spremenljivke. Prva spremenljivka je dvodimenzionalni »array«, tipa »Vector3«, z imenom »rotations«. Večdimenzionalne »array« definiramo tako, da zapišemo vejice med oglatimi oklepaji. To pomeni, da lahko shrani več stvari. Na primer, če imamo definirano spremenljivko z velikostjo [10,6], si lahko zapomni 10 informacij 6-krat, torej skupaj 60 informacij. V tem primeru predstavlja prva številka število pozicije, ki si jo želimo zapolniti, in druga število osi. Torej z eno spremenljivko dosežemo, da vseh 6 osi zapolni 10 pozicij. Nato definiramo vse osi in celo število, ki določa, katero pozicijo želimo.

V programu definiramo dve funkciji, ki sta izvršeni ob pritisku na gumb. Obe funkciji izvršita funkcijo »GetPosition()« ali »GivePosition()«. Funkcija »GetPosition« pridobi vse pozicije in jih zapiše v »array rotations«. Kot prvo število »array« uporabimo celo število »CurrentPosition«, ki jo dobimo s spustnim menijem, ki deluje podobno gumbu za preklop, saj vsakič, ko se spremeni vrednost, zapiše le-to. Nato je zapisan indeks osi. Vrednost kotov dobimo s funkcijo »Axis1.transform.localEulerAngles;«. V funkciji »GivePositions« naredimo obratno, in sicer določimo, da je vrednost »Axis1.transform.localEulerAngles« enaka »rotations«, ki smo ga določili v prejšnji funkciji.

Torej v spustnem meniju določimo eno izmed mest, kjer bomo to shranjevali. Nato pritisnemo gumb za shranjevanje in lahko spreminjamo rotacije osi, ampak ob pritisku na gumb za vrnitev v mesto, kjer smo shranili. Tako si je mogoče zapolniti 10 mest, vendar je to možno povečati, če bi tako želeli [9].



Slika 42: Program za shranjevanje pozicij

(Vir: osebni arhiv)

4.8 PROGRAM ZA OPRAVLJANJE MENIJA

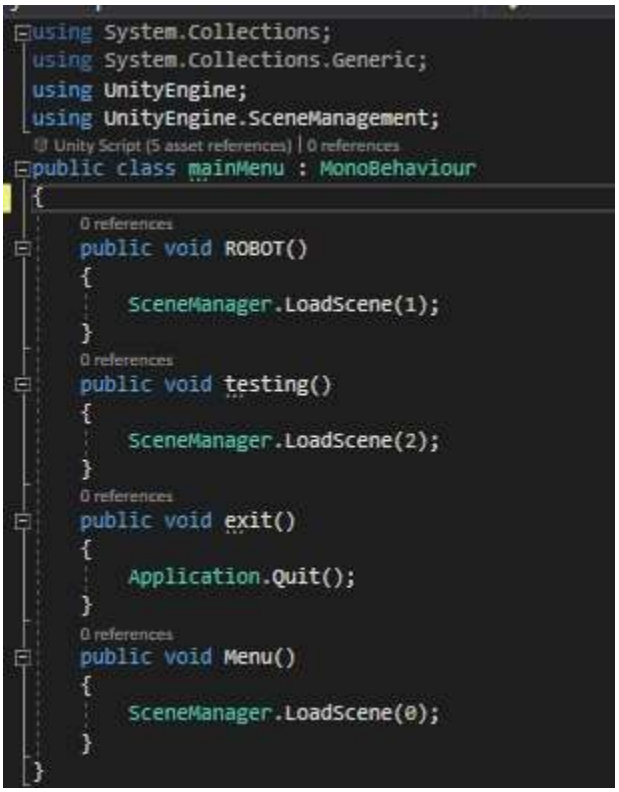
Ker ima ta aplikacija več scen, mora imeti tudi mesto, s katerega se ostale scene povezujejo. To imenujemo »meni«. Na »meni« smo nastavili tri gumbe. Prvi gumb nas prestavi v prvo sceno, kjer imamo model robota, drugi gumb nas prestavi v drugo sceno, kjer lahko postavljamo po resničnem svetu AR modele, ki so namenjeni robotski celici. Zadnji gumb zapre aplikacijo.

Pri pisanju programa je potrebno vstaviti »using UnityEngine.SceneManagement«. V tem programu sta funkciji »Start()« in »Update()« nepotrebni, saj se nič ne sme zgoditi ob začetku programa in med programom, če ni nič pritisnjeno. Nato smo definirali funkcije, ki morajo biti izvršene ob pritisku na določen gumb. Napisane funkcije nas prestavijo iz menija v obe sceni ali iz scene nazaj na meni. Če želimo, lahko tudi zapre program.

Prvo smo imenovali »ROBOT()« in nas prestavi v prvo sceno.

Druga se imenuje »testing()«, saj nismo vedeli, kaj bo v tej sceni med pisanjem programa. Enako smo storili za funkcijo »Meni()«. Nazadnje smo definirali tudi funkcijo »exit()«.

Pri opravljanju s scenami moramo vedeti, da se štetje začne pri 0, kot pri »array-jih«.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
Unity Script (5 asset references) | 0 references
public class mainMenu : MonoBehaviour
{
    0 references
    public void ROBOT()
    {
        SceneManager.LoadScene(1);
    }
    0 references
    public void testing()
    {
        SceneManager.LoadScene(2);
    }
    0 references
    public void exit()
    {
        Application.Quit();
    }
    0 references
    public void Menu()
    {
        SceneManager.LoadScene(0);
    }
}
```

Slika 43: Program s funkcijami za spreminjanje scen

(Vir: osebni arhiv)

4.9 PROGRAM ZA ZAJEMANJE ZASLONA

Če želimo shraniti slike zaslona z gumbom, kot da bi slikali s kamero na telefonu, bi lahko preprosto naredili posnetek zaslona, vendar je to nepotrebno in manj uporabno, saj bi ob tem opazili tudi UI elemente, ki so v našem programu skriti ob slikanju. Za zajemanje slik smo uporabili »Native Gallery Plugin«, ki smo ga naložili iz Unity »asset store«. Ta paket nam omogoča preprosto slikanje zaslona in shranjevanje teh slik v galerijo telefona. Ne dopusti pa nam skrivanje UI elementov in zakasnitev slike, zato smo morali to sami napisati.

Začeli smo z definiranjem spremenljivk za indeks slike »shotIndex«. To nam omogoča poimenovanje slik in oštevilčenje, saj ne želimo, da imajo isto ime. Nato smo definirali »GameObject canvas«, kjer so vsi UI elementi. Torej, če nastavimo »canvas« na neaktivno stanje, bodo vsi UI elementi tudi neaktivni. Nato definiramo »bool« vrednost, imenovano »IsDelay«, in celo število »delay«. Če je »IsDelay« na »true«, potem pogleda vrednost »delay« in zakasni za to vrednost. »Delay« nastavljamo s tipko preklop, vrednost »delay« pa nastavljamo s spustnim menijem.

V programu najprej definiramo funkcije, ki nam spremenijo vrednost »delay« in »IsDelay« glede na tipko in spustni meni, ter funkcijo »TakeAPicture()«. Ta funkcija najprej deaktivira »canvas« in spremeni »shotIndex« za eno vrednost navzgor. Nato preveri z if stavkom, če je »delay« vključen. Če je, se začne »COROUTINE(“wait”)«, če ni, potem se začne »COROUTINE(“TakeScreenshotAndSave”)«.

Pri tem uporabljamo »StartCoroutine« in »IEnumerator«. To uporabljamo, če moramo počakati na nek proces, vendar ne želimo, da se ustavi cel program. V funkciji »wait« smo uporabili funkcijo »Debug.log(delay)«, ki izpiše vrednost zamika v konzolo. Nato s funkcijo »yield return new WaitForSeconds(delay)« zaustavimo izvršitev celotne funkcije za število sekund, ki smo jih izbrali na spustnem meniju. Po čakanju se izvrši funkcija »TakeScreenshotAndSave()«, ki bi se izvršila takoj, ko bi imeli izključen »delay«.

Na začetku funkcije počakamo do konca sličice. V funkciji se naredi 2D-tekstura, ki jo imenujemo »ss«. To teksturo nastavimo tako, da je enaka velikosti zaslona, nakar na njo prilepimo vse piksle. Nato uporabimo funkcijo »SaveImageToGallery()«, ki je omogočena z »NativeGallery«. V funkciji nastavimo, katero teksturo uporabimo, torej »ss«, ime galerije, v katero shrani sliko, če galerije ni, jo bo ustvaril, in ime slike. Za ime smo izbrali »Screenshot (številka)«. Tako bodo vse slike imele enako ime, ampak drug indeks posnetka, ki se spremeni

ob vsakem posnetku. Slika se shrani v .png formatu. Na koncu funkcije se tekstura »ss« zbriše in ponovno aktiviramo »canvas« [8].

```
public int size = 2;
private int shotIndex;

public GameObject canvas;
public bool IsDelay;

public int delay;
0 references
public void dropdown(int val)
{
    delay = val+1;
}
0 references
public void toggle_delay(bool val)
{
    IsDelay = val;
}
```

Slika 44: Definiranje zakasnitve

(Vir: osebni arhiv)

```
public void TakeAPicture()
{
    canvas.SetActive(false);

    shotIndex++;
    if (IsDelay == true)
    {
        StartCoroutine("wait");
    }else
    {
        StartCoroutine("TakeScreenshotAndSave");
    }
}
0 references
IEnumerator wait()
{
    Debug.Log(delay);
    yield return new WaitForSeconds(delay);
    StartCoroutine("TakeScreenshotAndSave");
}
0 references
private IEnumerator TakeScreenshotAndSave()
{
    yield return new WaitForEndOfFrame();

    Texture2D ss = new Texture2D(Screen.width, Screen.height, TextureFormat.RGB24, false);
    ss.ReadPixels(new Rect(0, 0, Screen.width, Screen.height), 0, 0);
    ss.Apply();

    // Save the screenshot to Gallery/Photos
    NativeGallery.Permission permission = NativeGallery.SaveImageToGallery(ss, "Unreal Robotics", $"Screenshot{shotIndex}.png",
    Debug.Log("Permission result: " + permission);

    // To avoid memory leaks
    Destroy(ss);
    canvas.SetActive(true);
}
```

Slika 45: Program za shranjevanje slik

(Vir: osebni arhiv)

4.10 PROGRAM ZA DOLOČANJE, KATERI PREDMET POSTAVIMO V AR

V obogateni resničnosti je možno videti predmete v 3D, zato smo naredili tudi model robotske celice, kjer lahko posamezne dele postavljamo posebej. To smo naredili s pomočjo Vuforije, saj ima možnost prepoznavanja tal in postavitev predmetov na njih. Ob pritisku na zaslon se pojavi »GameObject« z vsemi modeli. Če ne želimo imeti vseh predmetov na enem mestu, moramo pred postavitvijo deaktivirati vse modele, razen tistega, ki ga želimo videti. Na začetku mora biti to prvi po vrsti, nato pa vzamemo tistega, ki ga nastavimo s spustnim menijem.

V programu definiramo »array« tipa »GameObject« in ga imenujemo »objects«. Nato v »Start()« funkciji deaktiviramo vse predmete s pomočjo »for« zanke in aktiviramo prvi predmet. V funkciji »objectSelection«, ki dobi vrednost s pomočjo spustnega menija, naredimo enako kot v »Startu«, ampak aktiviramo določen predmet.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (2 asset references) | 0 references
public class switchObjectAndDelete : MonoBehaviour
{
    public GameObject[] objects;
    public int avatar;



    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        for (int i = 0; i < objects.Length; i++)
        {
            objects[i].SetActive(false);
        }
        objects[0].SetActive(true);
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
    0 references
    public void objectSelection(int val)
    {

                for(int i=0; i < objects.Length; i++)
                {
                    objects[i].SetActive(false);
                }
                objects[val].SetActive(true);
        avatar = val;

    }

}
```

Slika 46: Program za spreminjanje 3D-modelov

(Vir: osebni arhiv)

4.11 PROGRAM ZA ROTACIJO PREDMETOV

Ob postavitvi predmetov na tla se morajo le-ti obrniti proti kameri, saj želimo sami spreminjati rotacijo predmeta. Ta program smo tudi uporabili pri rotaciji informacijskih oken. Čeprav sta ti dve svari zelo podobni, smo ju ločili z »bool« spremenljivko »faceCamera« in dvema »if« funkcijama. Če je nastavljena na »false«, potem bo gledal proti kameri le takrat, ko ga prvič vstavimo, torej to uporabljamo pri »ground detection« in robotski celici. Če pa je nastavljen na »true«, potem se uporablja za konstantno rotacijo informativnih oken.

V programu najprej definiramo spremenljivke »cam« tipa »GameObject«. Vanj v Unity Editor vstavimo AR kamero. Nato definiramo »Vector3« in možnosti spremembe rotacije, ki jo je možno spremeniti v urejevalniku. Nazadnje še »bool« vrednost »faceCamera«, ki mora biti vklopljena v urejevalniku in nam omogoča, da predmet vedno gleda proti kameri. Pri obeh delih smo uporabili »rotationChange«. Uporabili smo funkcijo »Awake()«, ki je podobna »Start()«, ampak se izvrši vedno po aktivaciji predmeta. V funkciji »Awake« smo spremenili vrednost rotacije s »transform.localEulerAngles«. X- in Z-vrednost ostane enaka, spremeni se le rotacija okoli osi y. Ampak se izvrši samo, če je spremenljivka »faceCamera« nastavljena na »false«. V tem primeru nam »rotationChange« premakne rotacijo osi y za določeno vrednost. To je uporabno, če je model robota v urejevalniku obrnjen narobe.

```
void Awake()
{
    if (!faceCamera)
    {
        transform.localEulerAngles = new Vector3(0, rotate.y + rotationChange, 0);
    }

}
```

Slika 47: »Awake« funkcija

(Vir: osebni arhiv)

V »Update()« funkciji smo spremenili rotacijo osi x glede na trenutno rotacijo kamere in model robota. Ti dve vrednosti dobimo s funkcijo »GameObject.transform.localEulerAngles.y«. Tukaj nam »rotationChange« spremeni rotacijo okoli osi y. V našem primeru morajo imeti rotacijo 90, ki jo nastavimo v Editor-ju [9].

```
void Update()
{
    rotate.y = cam.transform.localEulerAngles.y;

    robotrot.y = robot.transform.localEulerAngles.y;
    if (faceCamera)
    {
        transform.localEulerAngles = new Vector3(-rotate.y+ robotrot.y, rotationChange, 0);
    }
}
```

Slika 48: Program za obračanje informativnih oken

(Vir: osebni arhiv)

4.12 PROGRAM ZA BRISANJE PREDMETOV V AR

Če postavimo predmet na napačno mesto ali pa jih imamo preveč, jih je potrebno zbrisati. Za to smo naredili dva gumba. Prvi izbriše vse predmete, drugi pa le zadnjega.

V programu definiramo »array« tipa »GameObject«, z imenom »characters«. V »Update()« funkciji določimo vrednosti »array«. Te predmete dobimo s pomočjo funkcije »FindGAmeObjectWithTag("thing")«. To poišče vse predmete, ki imajo takšno ime. To je originalni »GameObject«, ki ima vse predmete na njem in vse kopije, ki jih naredimo s pritiskom na gumb. Nato ustvarimo dve funkciji. Prva izbriše vse, torej uporabimo »for« zanko, ki začne pri 1, saj ne želimo, da izbriše prvi predmet z indeksom 0. Druga izračuna indeks zadnjega postavljenega predmeta. Ta račun je število predmetov v »array characters« minus 1. Če je rezultat 0, funkcija ne naredi ničesar, če je rezultat več, potem zbriše predmet s tem indeksom.

```
public GameObject[] characters;
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{

}

// Update is called once per frame
Unity Message | 0 references
void Update()
{
    characters = GameObject.FindGameObjectsWithTag("thing");
}

0 references
public void deleteAll()
{
    for (int i = 1; i < characters.Length; i++) {
        Destroy(characters[i]);
}
}
0 references
public void deleteLast()
{
    int last = characters.Length - 1;
    if (last == 0)
    {
        return;
    }
    else
    {
        Destroy(characters[last]);
    }

}
```

Slika 49: Program za brisanje 3D-modelov

(Vir: osebni arhiv)

4.13 PROGRAM ZA PRIKAZOVANJE INFORMACIJSKIH OKEN

Če želimo dodati informacijska okna ob robotu, moramo za njihovo prikazovanje napisati program. Ob pritisku na virtualni gumb se bodo okna povečala, ob ponovnem pritisku pa bodo izginila.

Najprej moramo definirati spremenljivke za informacijska okna »SectionInfo« in »Vector3«, ki ga imenujemo »wantedScale«. To predstavlja faktor njihove velikosti in ga nastavimo na vrednost 0,0,0 s funkcijo »Vector3.zero«. Definiramo tudi virtualni gumb, ki ga imenujemo »Play«, hitrost, s katero se bodo povečala, in »bool« vrednost »isOn«, ki jo spreminjamo ob pritisku na gumb.

V funkciji »Start« najprej definiramo gumb »Play« s spremenljivko »P«.

Slika 50: Definicije spremenljivk in virtualnega gumba

(Vir: osebni arhiv)

V funkciji »Update« najprej preverimo vrednost »isOn«. Če je 1, potem aktivira »Informacijska okna« in nastavi »wantedScale« na 1,1,1, kar pomeni, da so v normalni velikosti. Nato s funkcijo »Vector3.Lerp« nastavimo velikost tako, da se s časom veča. Če je vrednost »isOn« enaka 0, potem nastavimo, da je »wantedScale« enak 0,0,0. Z enako funkcijo pomanjšamo okna in po 0,3 s deaktiviramo.

```
if (isOn)
{
    SectionInfo.SetActive(true);
    wantedScale = Vector3.one;
    SectionInfo.transform.localScale = Vector3.Lerp(SectionInfo.transform.localScale,
}
else
{
    wantedScale = Vector3.zero;
    SectionInfo.transform.localScale = Vector3.Lerp(SectionInfo.transform.localScale,
    StartCoroutine("wait");


}
```

Slika 51: Sprememba velikosti oken

(Vir: osebni arhiv)

V funkciji, ki jo program izvrši ob pritisku na virtualni gumb, se vrednost »isOn« spremeni v nasprotno vrednost, na primer 1 v 0. Nato izpiše to vrednost v konzolo. Naredili smo tudi »IEnumerator«, ki počaka 3 s, preden deaktivira okna [6], [8], [9].

```
public void P(VirtualButtonBehaviour vb)
{

    isOn = !isOn;,,
    Debug.Log("info show" + isOn);

}

0 references
IEnumerator wait()
{
    yield return new WaitForSeconds(0.3f);
    SectionInfo.SetActive(false);
}
```

Slika 52: Funkcija virtualnega gumba

(Vir: osebni arhiv)

5 NASTAVITVE V UNITY EDITOR

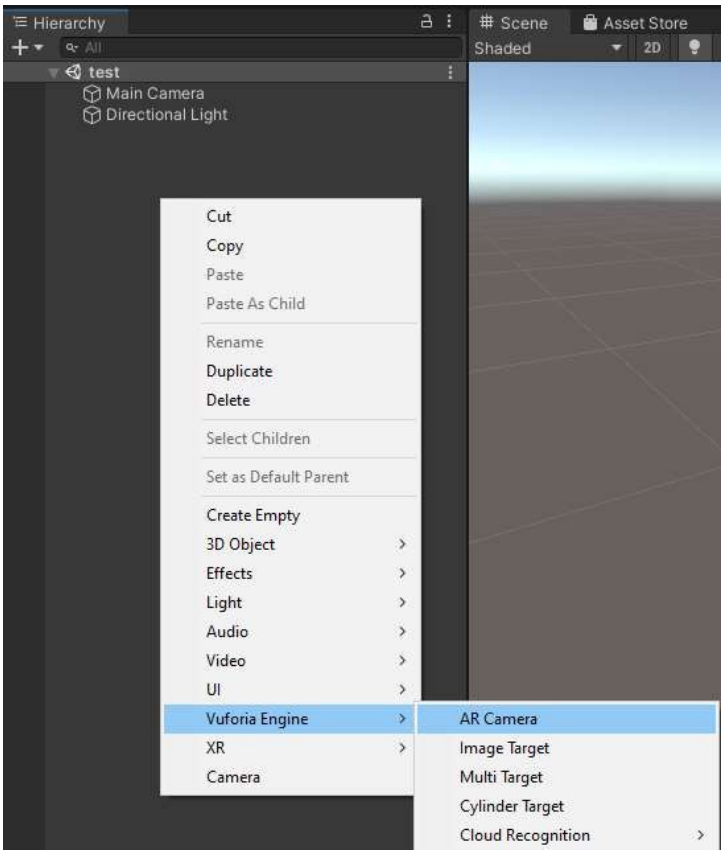
5.1 UNITY EDITOR

S pisanjem programov omogočamo premikanje ali delovanje predmetov v aplikaciji. Da pa lahko program nadzira predmet, moramo v Unity urejevalniku pravilno nastaviti predmet in program. Možno je nastaviti spremenljivke, ki so zapisane kot »public«. V urejevalniku je pomembno nastaviti predmete na pravilno mesto, jim dati barvo, nanj dati program in nastaviti spremenljivke. Vse, kar ni že vnaprej postavljeno v programu.

5.2 IMAGE TARGET ROBOTA

5.2.1 Kamera

Po nastavitvi Unity in Vuforie smo dodali v sceno AR kamero. To kamero omogoča Vuforia. Torej moramo klikniti desni klik na »Hierarchy« okno, nato pa še na »Vuforia Engine« in »AR Camera«. Ko ustvarimo sceno, se nam avtomatsko pojavi »Main Camera« in »Directional Light«, zato moramo izbrisati »Main Camera«. Po vstavitvi kamere je treba nastaviti licenco. To storimo s klikom na »AR Camera« in »Open Vuforia Engine configuration«. Tja smo kopirali licenco in ostale želene nastavitve.

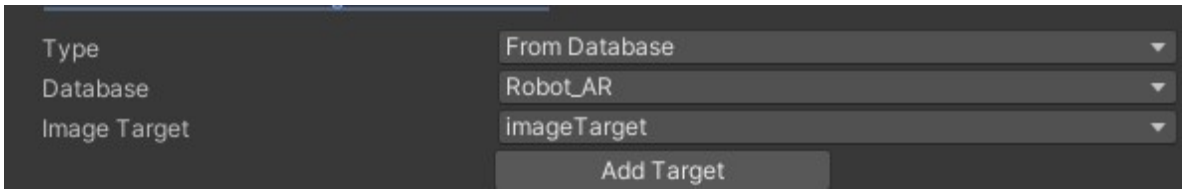


Slika 53: »Hierarchy« okno

(Vir: osebni arhiv)

5.2.2 Image Target

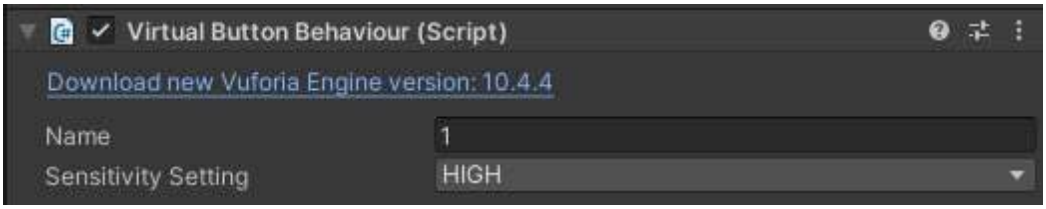
Podobno kot pri vstavljanju kamere smo tudi tukaj vstavili »Image Target« na isti način. Po vstavitvi smo kliknili na »Image Target« in nastavili sliko. Najprej smo izbrali, od kod naj bo slika izbrana, torej iz podatkovne baze, imenovane »Robot_AR«, in izbrali sliko, imenovano »imageTarget«.



Slika 54: Izbira slike iz podatkovne baze

(Vir: osebni arhiv)

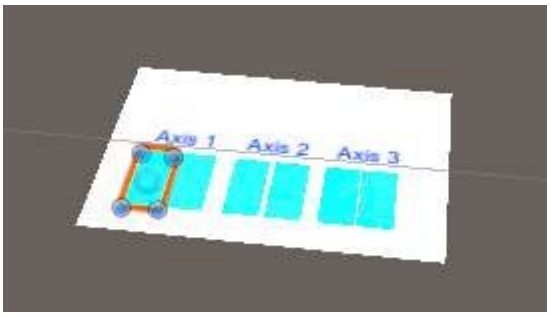
Na enak način smo vstavili vse »Image Target«, s katerimi lahko nadziramo premikanje robota. Na njih moramo vstaviti virtualne gumbe, ki jim damo ime in jim nastavimo občutljivost.



Slika 55: Nastavitev imena gumba

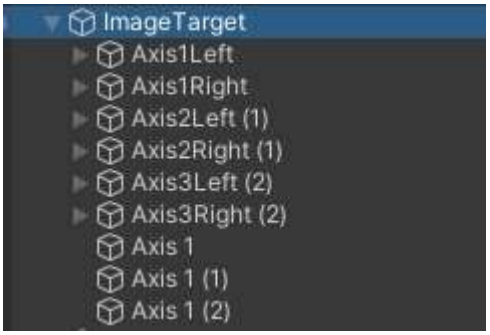
(Vir: osebni arhiv)

To smo naredili šestkrat za vsak »Image target«, saj potrebujemo 12 gumbov za premikanje.



Slika 56: Postavitev virtualnih gumbov

(Vir: osebni arhiv)

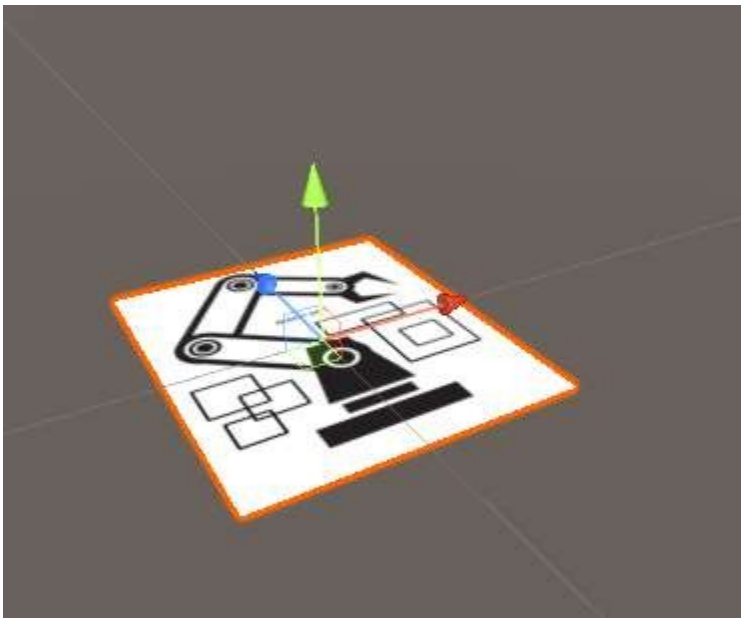


Slika 57: Vsi gumbi na »Image Target«

(Vir: osebni arhiv)

5.2.3 Vstavitev Modela

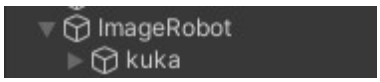
Slika 58 prikazuje trenutni izgled »Image Target«.



Slika 58: »Image Target« od robota

(Vir: osebni arhiv)

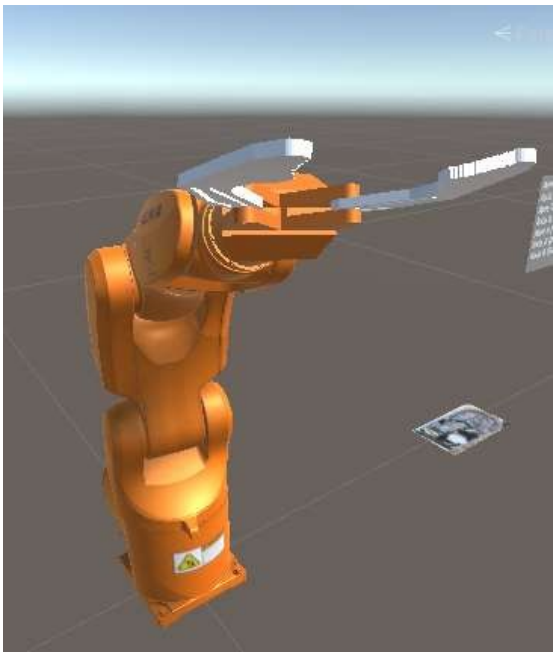
Zdaj je potrebno dodati model, ki se bo pojavil ob pritisku na gumb. Model smo dobili iz spletne strani, kjer smo lahko naložili brezplačne modele. Model, ki smo ga izbrali, smo nastavili kot »child object« predmeta »ImageRobot« (tipa »Image Target«).



Slika 59: »Child Object«

(Vir: osebni arhiv)

Po nastavitvi modela bo model prekrival celoten »Image Target«. Nato smo vsakemu delu robota nastavili »Tool handle position«, kar pove, v kateri točki se bo predmet rotiral. To nastavimo iz sredine na pivot, kar nam omogoča rotacijo predmetov na pravilnem mestu. V nekaterih primerih to ni mogoče, zato moramo narediti nov prazen »GameObject« in nastaviti ta predmet kot »child object« praznega predmeta.

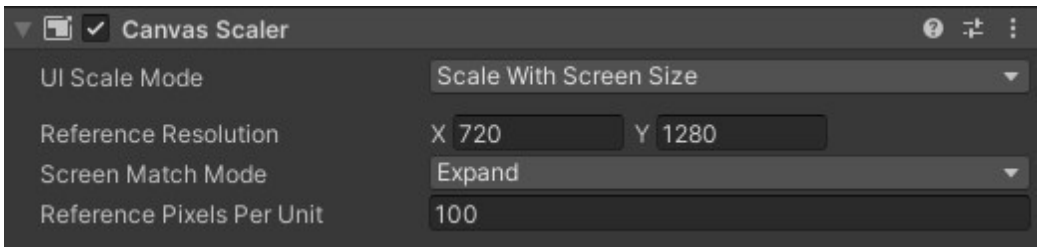


Slika 60: Model robota

(Vir: osebni arhiv)

5.2.4 UI komponente

Za uporabo programa je potrebno nastaviti razne gumbe. Ti gumbi nam omogočajo slikati, spremeniti nastavitve ali nadzirati robota brez virtualnih gumbov. Najprej smo vstavili v sceno »Canvas« in »EventSystem«. To nam omogoča uporabo gumbov in drugih UI komponent. Na »kanvasu« je potrebno nastaviti velikost UI komponent glede na velikost zaslona.

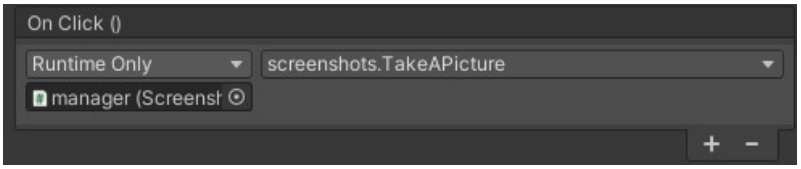


Slika 61: Nastavitev »kanvasa«

(Vir: osebni arhiv)

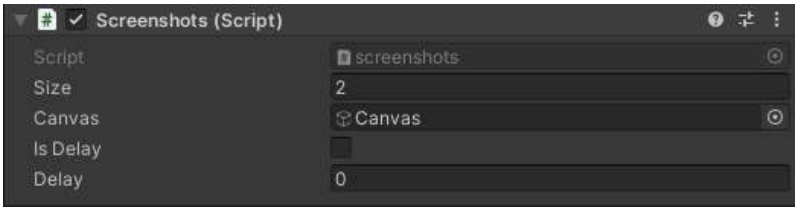
Po vstavitvi gumbov je potrebno nastaviti njihovo funkcijo. Najprej povlečemo predmet iz »Hierarchy okna«, ki ima na sebi program, ki vsebuje funkcijo, ki jo ta gumb izvrši. V tem primeru je gumb za slikanje in potrebuje funkcijo za zajemanje slik. Torej smo dali na

»Manager« program za zajemanje slik. Nastavili smo le »kanvas GameObject«, saj je vse ostalo že določeno v programu. Na enak način smo nastavili tudi ostale gumbe za nastavitve, zamik in spustne menije.



Slika 62: Nastavitev gumbov

(Vir: osebni arhiv)



Slika 63: Nastavitve programa za zajemanje slik

(Vir: osebni arhiv)

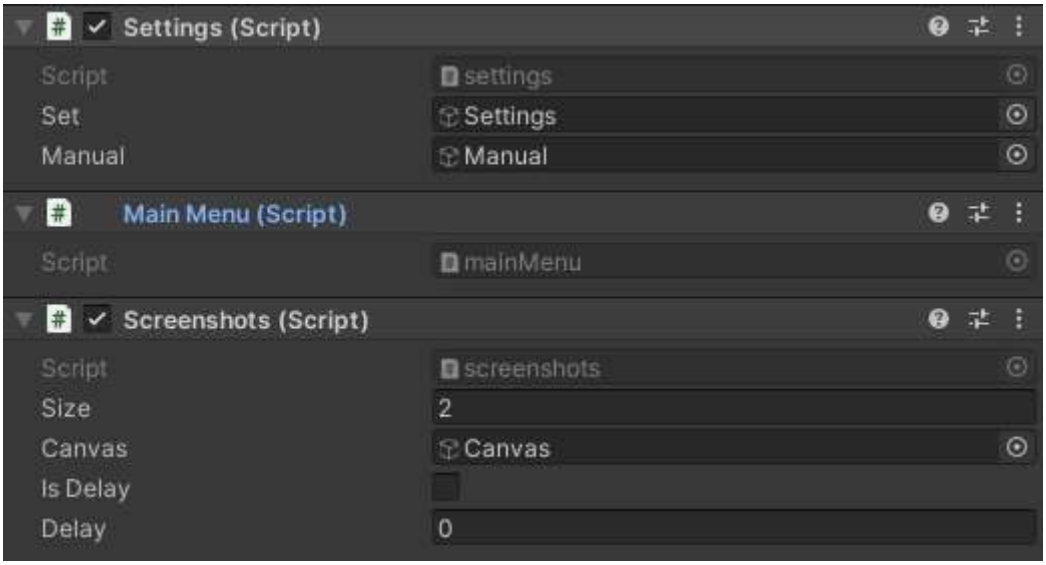
Slika 64: Nastavitve v aplikaciji

(Vir: osebni arhiv)

5.2.5 Nastavitev programov

Ko vstavimo program v sceno, ga moramo najprej postaviti na nek predmet. V nekaterih primerih lahko ta program nastavimo na prazna predmeta, ki smo ju imenovali »manager« in »save and load«.

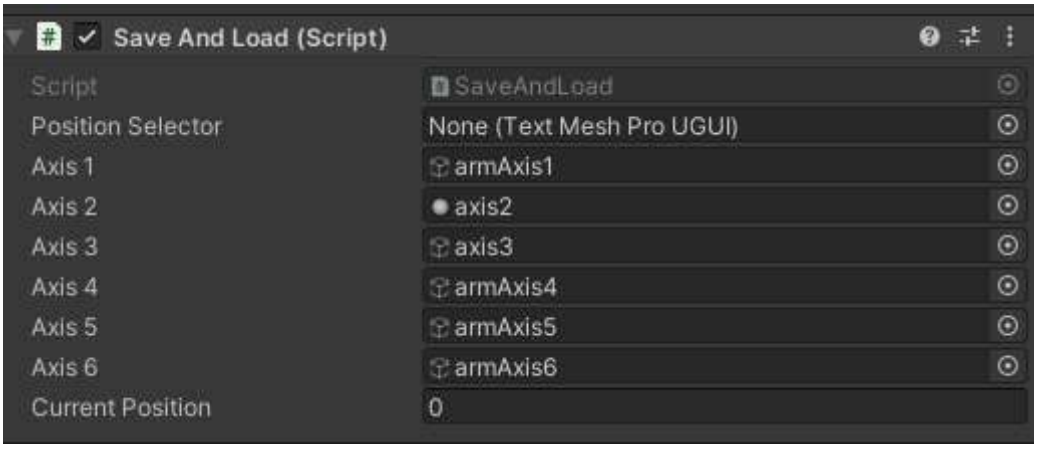
Na »manager« smo nastavili programe za nastavitve, glavni meni in slike. Program za glavni meni nima spremenljivk, zato ni treba narediti ničesar, ampak je treba pri programu za nastavitve in slike potegniti predmete za nastavitve in ročno premikanje osi robota.



Slika 65: Nastavitve programov za nastavitve

(Vir: osebni arhiv)

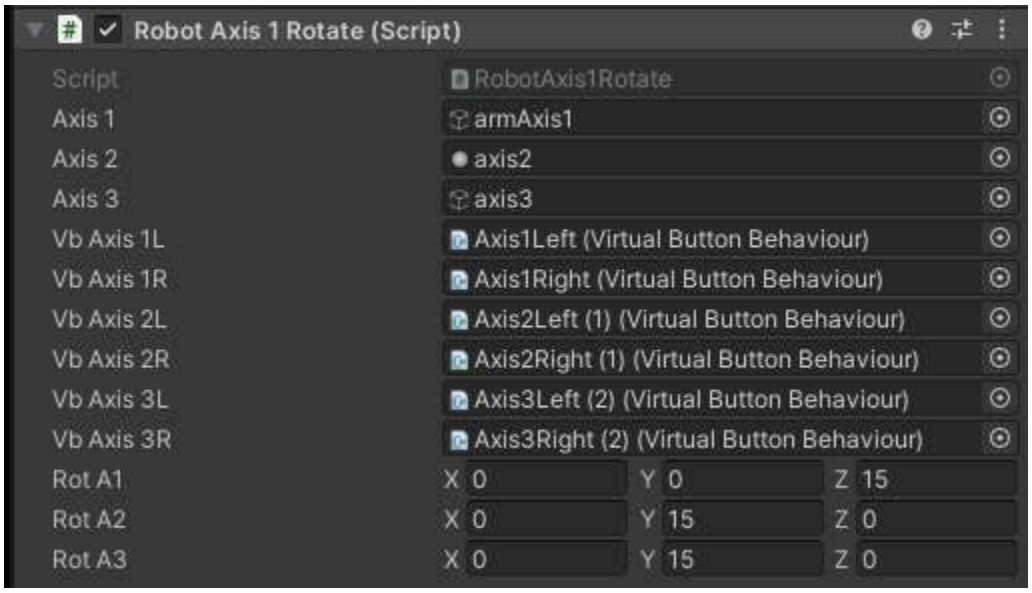
Na praznem predmetu »save and load« je treba vsak del robota dati pod pravilno os, saj želimo, da si zapomni in spreminja rotacije teh delov.



Slika 66: Nastavitev programa za shranjevanje pozicij robota

(Vir: osebni arhiv)

Za premikanje z virtualnimi gumbi smo naredili podobno, kot pri prejšnjih programih, vendar smo nastavili program na »ImageTarget« z virtualnimi gumbi. Tu smo najprej nastavili tri dele robota, saj moramo spremeniti njihove osi in vseh šest virtualnih gumbov, ki so na »ImageTargetu«. Na koncu smo nastavili še rotacijo vseh osi, saj se vsi deli ne vrtijo okoli iste osi. Na primer: 1. del se bo vrtel okoli z-osi, 2. in 3. pa okoli y-osi.



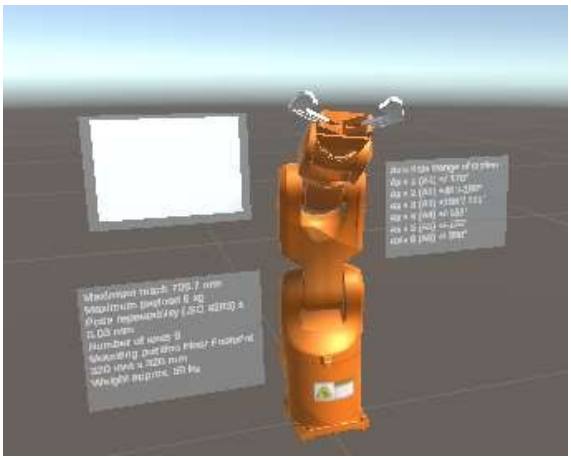
Slika 67: Nastavitev programa za premikanje robota z virtualnimi gumbi

(Vir: osebni arhiv)

5.2.6 Informacijska okna

Dodali smo informacijska okna, saj raziskujemo obogateno resničnost v robotiki in izobraževanju.

Najprej smo dodali 3D-ravnine in jih naredili prosojne, nanje smo dodali besedila in video. Ker lahko ta model gledamo iz več različnih smeri in ker je »ImageTarget« lahko obrnjen, smo napisali program, ki ga nastavimo na vsakega od oken.



Slika 68: Postavitev informacijskih oken

(Vir: osebni arhiv)

V programu vstavimo kamero in »ImageTarget« ter spremenimo »rotationChange« na 90 in nastavimo »faceCamera« na 1 [6], [8], [14], [15].

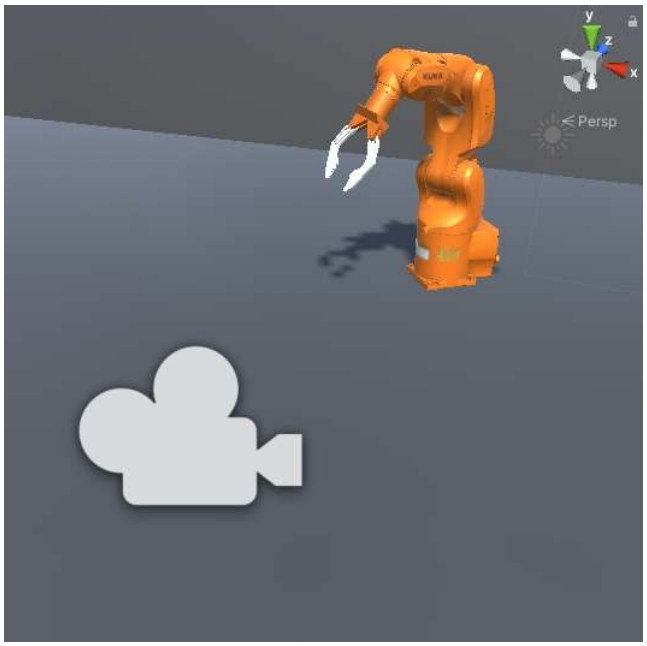
Slika 69: Nastavitve za program, ki rotira predmete

(Vir: osebni arhiv)

5.3 MENI

5.3.1 Ozadje

Za ozadje imam dve ravnini, ki sta pobarvani sivo. Ena služi za tla, na kateri je postavljen robot, druga je postavljena pod kotom 90 stopinj. Postavljen je model, ki ga uporabljamo pri »ImageTargetu«. Kamero smo orientirali tako, da je usmerjena proti robotu.

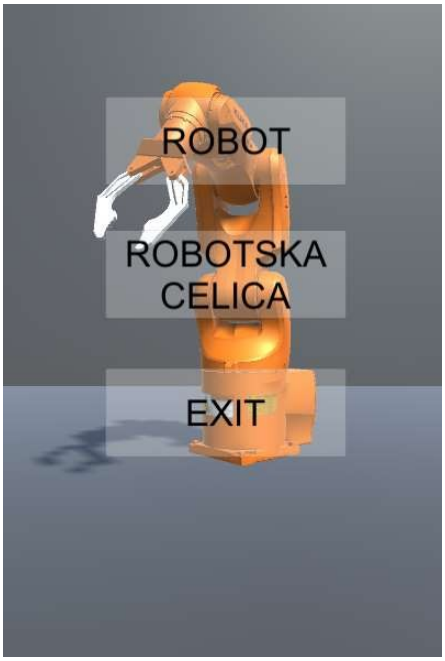


Slika 70: Model za ozadje

(Vir: osebni arhiv)

5.3.2 Gumbi

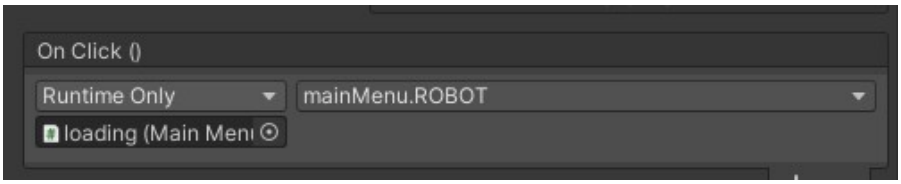
V »kanvas« smo dodali tri gumbe. Prvi gumb nas premakne v drugo sceno, kjer imamo »Image Target« robota, drug gumb nas premakne v sceno tri, kjer imamo »ground detection« za robotsko celico. Zadnji gumb nam zapre aplikacijo.



Slika 71: Gumbi za opravljanje menija

(Vir: osebni arhiv)

Glede programov smo najprej naredili prazen predmet, na katerega smo dali program za nadziranje menija. Ta predmet smo imenovali »loading«. Na vsak gumb določimo, katera funkcija se naj izvrši ob pritisku. To naredimo tako, da vstavimo predmet, na katerem je program s funkcijo, ki jo želimo izvršiti, in jo izberemo.



Slika 72: Nastavitve za gumb

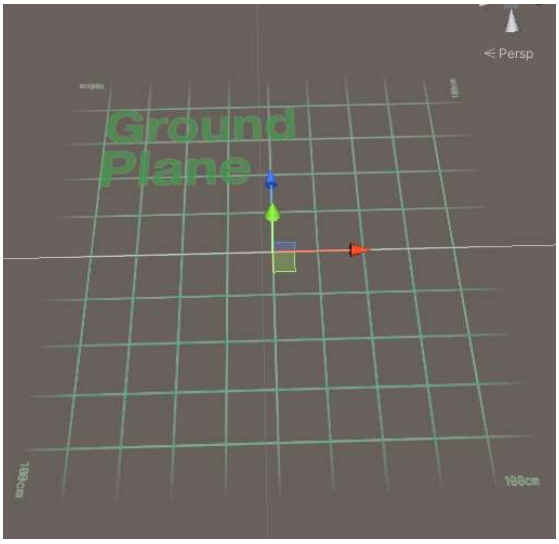
(Vir: osebni arhiv)

5.4 ROBOTSKA CELICA MODEL

5.1.1 Modeli in »Ground Plane«

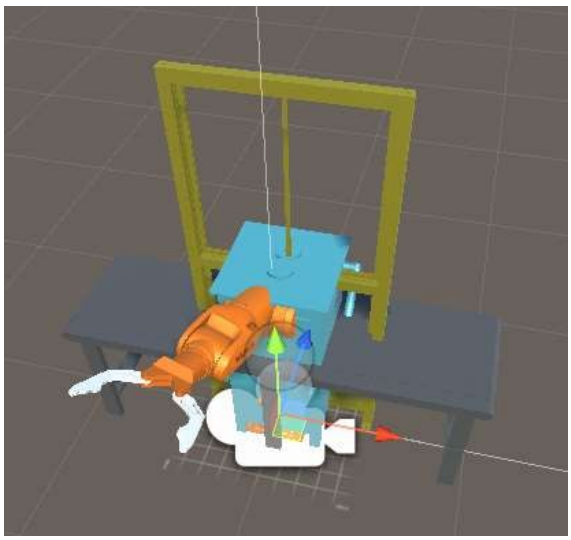
V tej sceni s pomočjo zaznavanja tal vstavljamo AR modele v resnični svet. Tukaj smo vstavili enake AR komponente kot pri »Image target« sceni, dodali pa smo še dve. To sta »Plane Finder« in »Ground Plane Stage«.

»Ground Plane Stage« je predmet v obliki kvadrata, z merami 1 x 1 m. Nato dodamo vse 3D-modele, ki jih nameravamo uporabiti v robotski celici. Vsi modeli se prekrivajo, zato moramo uporabiti program, s katerim deaktiviramo vse modele, razen tistega, ki ga uporabljamo.



Slika 73: »Ground Plane«

(Vir: osebni arhiv)



Slika 74: 3D-modeli na »Ground Planu«

(Vir: osebni arhiv)

Naredili smo prazen predmet, ki smo ga imenovali »objects« in nanj postavili program »Switch Object And Delete«, ki nam omogoča, da postavljamo na tla le tisti 3D-model, ki ga želimo. V »array«, imenovan »Objects«, smo vstavili vse modele.

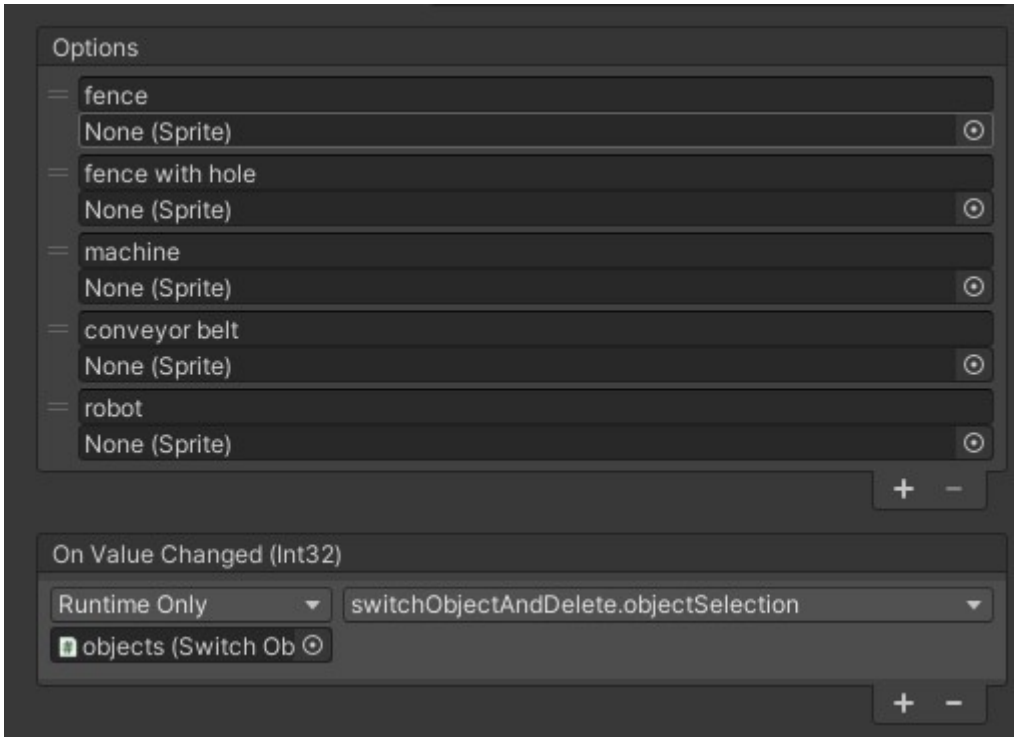
Slika 75: Nastavitev programa za spreminjanje 3D-modelov

(Vir: osebni arhiv)

Izbiro modela smo izvedli s pomočjo spustnega menija. Nastavili smo imena in jim dali enak indeks kot v »Switch Object And Delete« programu.

Slika 76: Spustni meni

(Vir: osebni arhiv)



Slika 77: Nastavitev spustnega menija

(Vir: osebni arhiv)

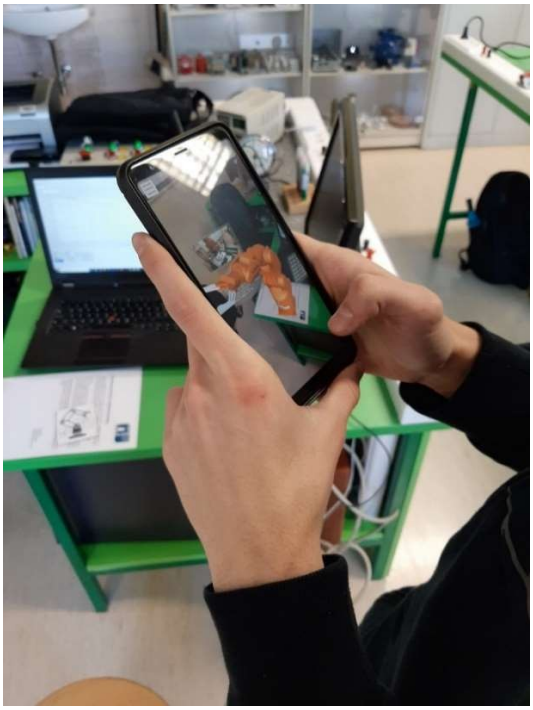
»Plane Finder« je predmet, ki ga doda »Vuforia Engine«. Avtomatsko najde tla in postavi nanj indikator. Ob pritisku na zaslon se bo pojavil model, ki smo ga izbrali. Nastavili smo »Duplicate Stage«, tako da je lahko hkrati aktivnih več različnih modelov.

Slika 78: »Plane Finder«

(Vir: osebni arhiv)

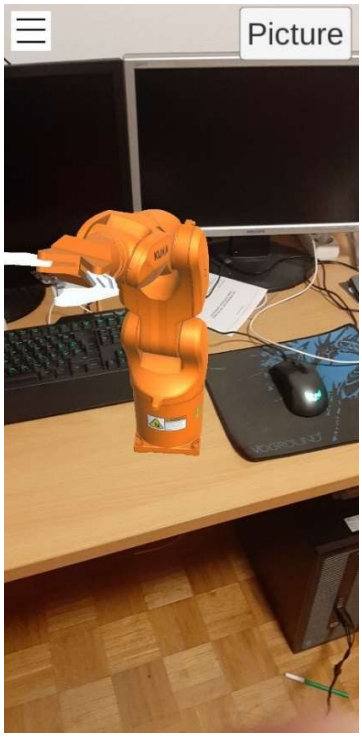
Če želimo izbrisati model, ki smo ga postavili, ali izbrisati vse modele, je treba pritisniti na gumbe. Na vsakega od gumbov smo nastavili funkcijo, ki jo mora izvršiti. Kot v prejšnji sceni smo vstavili gumb za zajemanje slik, vendar smo ga preoblikovali in ga premaknili na spodnji del zaslona.

5.5 SLIKE UPORABE APLIKACIJE



Slika 79: Uporaba AR aplikacije

(Vir: osebni arhiv)

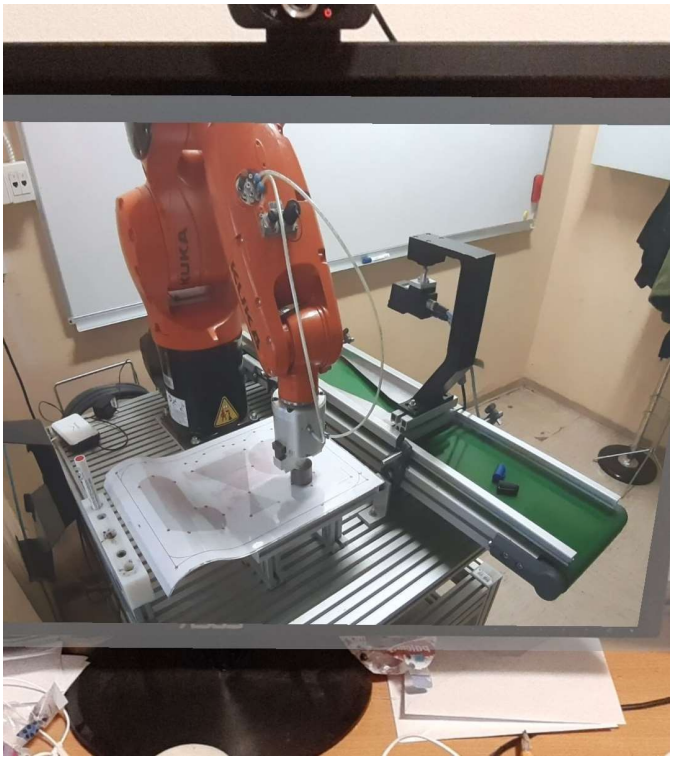


Slika 80: AR model ob zagonu aplikacije

(Vir: osebni arhiv)

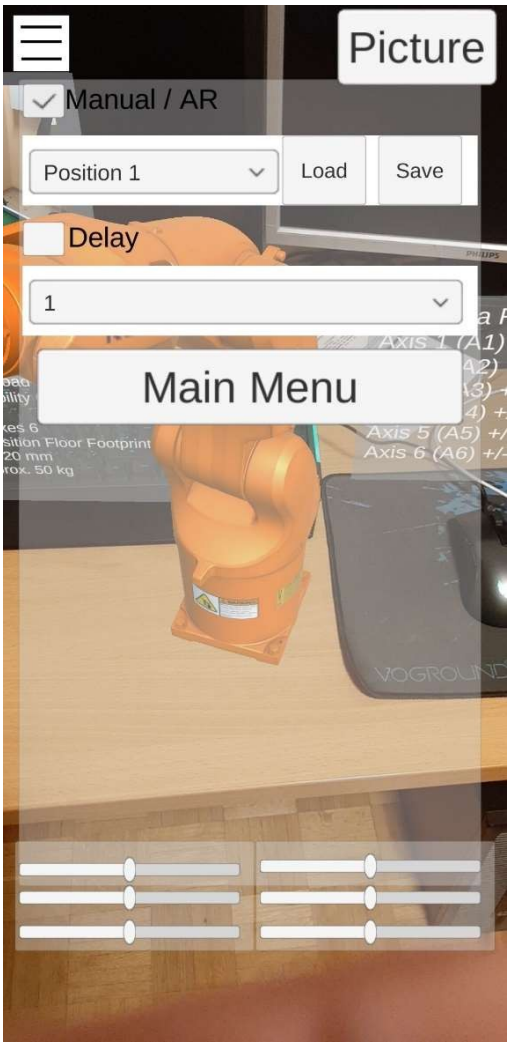
Slika 81: Informacijska okna

(Vir: osebni arhiv)



Slika 82: Video

(Vir: osebni arhiv)



Slika 83: Nastavitve in drsniki za premikanje robota

(Vir: osebni arhiv)

6 MERITEV POZORNOSTI IN SPROŠČENOSTI V POVEZAVI Z UPORABO AR OKOLJA

6.1 POZORNOST IN POTOPITEV (IMERZIJA)

Imerzija je tehnični izraz za potopitev v umetni svet. Opisuje učinek, ki ga povzroči situacija, okolje ali grafična predstavitev, zaradi katere se uporabnikova zavest umakne v ozadje, tako da se virtualno okolje dojema kot resničnost.

Poznamo dva tipa imerzije, in sicer mentalno in fizično.

Mentalna (miselna) imerzija nastopi, kadar se uporabnik vživi v določen izmišljen svet. Potopitev te vrste največkrat doživljamo med branjem ali med gledanjem filma, ko si v živo predstavljamo dogajanje in smo včasih žalostni ali razočarani, ko se kakšen zaplet ali razplet ne konča tako, kot smo želeli. Fizična ali telesna imerzija pa je samo dodatek mentalni, kjer se še s telesom vživimo v navidezno resničnost.

Oprema EEG postaja vse bolj prenosna in miniaturna, s preprosto pripravo pa je mogoče pridobiti natančne podatke o možganskih valovih in tako izboljšati uspešnost študentov. Elektroencefalogram (EEG) je široko uporabljena neinvazivna metoda za spremljanje možganov. Naprava EEG uporablja elektrode na čelu za zaznavanje električnih valov v možganih, elektronika pa za obdelavo signalov in prenos podatkov prek povezave Bluetooth v prenosni računalnik, tablični računalnik ali pametni telefon. Za pošiljanje podatkov v komunikacijski kanal uporablja signalni procesor in komunikacijsko enoto. EEG se uporablja na področju izobraževalnih raziskav za ocenjevanje in spremljanje učinkovitosti tradicionalnih in sodobnih metod poučevanja ali za razvoj sistemov, ki temeljijo na povratnih informacijah. EEG zaznava človeške možganske valove na podlagi izbranih značilnosti valov α, β, d in q.

Slika 84: Merjenje z »MindWave Mobile 2«

(Vir: osebni arhiv)

6.2 MERITVE POZORNOSTI IN POTOPITVE

Spreminjanje vala β v EEG je močno povezano s pozornostjo, vala α pa z meditacijo. Pozornost je vedenjski in kognitivni proces selektivnega osredotočanja na določen vidik informacije, ki je subjektiven ali objektiven, medtem ko se druge zaznavne informacije prezrejo. V nasprotju s pozornostjo je meditacija namerno in samo-regulativno usmerjanje pozornosti za sprostitev in umiritev uma. Meditacija predstavlja duševno stanje posameznika in se nanaša na zmanjšanje aktivnih duševnih procesov v možganih. To pomeni, da višja stopnja sproščenosti pomeni, da je posameznik bolj sproščen in manj pod stresom. Obstajajo različni kognitivni vidiki, ki jih je mogoče meriti z napravo EEG, kot so kontekst branja, vzorci predstavitve, interaktivno vedenje, izobraževalna zabava, e-učenje, pridobivanje motoričnih sposobnosti in spodbujanje uspešnosti.

Možganski valovi (EEG) so nihajoči električni impulzi v možganih, ki merijo le nekaj milijonink volta. Prek njih se med nevroni v naših možganih prenašajo posameznikovo vedenje, čustva in misli. Nastajajo zaradi sinhroniziranih električnih impulzov, ki so posledica komunikacije med skupinami nevronov. Naši možganski valovi imajo različne frekvence. Obstaja pet splošno priznanih možganskih valov. Klasična imena teh pasov EEG so alfa, beta, gama, delta in theta. Merijo se v ciklih na sekundo ali hercih (Hz).

- Delta možganski valovi (1–3 Hz) so najpočasnejši možganski valovi z največjo amplitudo. Najpogosteje jih doživljamo med spanjem. Na splošno so različne ravni delta možganskih valov povezane z razumom in dojemanjem.
- Možganski valovi theta (4–7 Hz) predstavljajo zasanjano, vesoljno stanje uma, povezano z duševno neučinkovitostjo. Pri zelo počasnem možganskem valovanju theta gre za zelo sproščeno stanje med budnostjo in spanjem.
- Možganski valovi alfa (8–12 Hz) so povezani s stanjem sproščenosti in predstavljajo možgane, ki v stanju pripravljenosti čakajo, da se na potrebo odzovejo.
- Možganski valovi beta (13–38 Hz) so povezani s stanjem duševne in intelektualne dejavnosti ter navzven usmerjene koncentracije in pozornosti.
- Možganski valovi gama (39–42 Hz) so najhitrejši možganski valovi. Gama valovi oblikujejo posameznikovo sposobnost zaznavanja in zavest.

[7], [11], [16], [18]

6.3 POSTOPEK MERITVE

V namen raziskovanja smo se odločili izmeriti pozornost in meditacijo v treh različnih točkah:

- video posnetku,
- sliki,
- uporabi AR aplikacije.

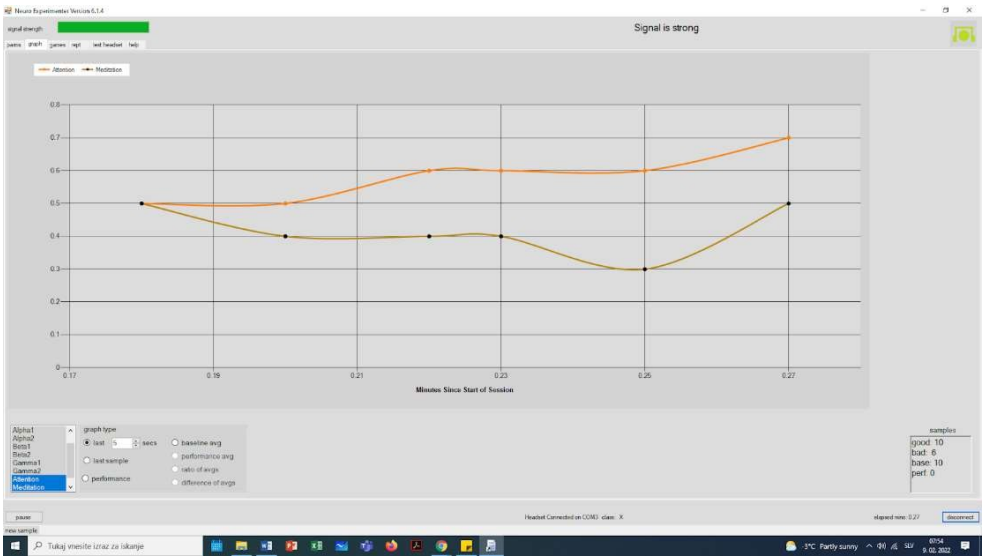
Meritve smo izvajali s pomočjo naglavnega merilca možganskih valov, pri kandidatih pa smo vse meritve izvajali po eno minuto za vsako fazo. Le-te smo izvajali s pomočjo naglavnega merilnika »NeuroSky MindWave Mobile 2«. Merilnik omogoča varno merjenje možganskih valov in jih preko Bluetooth povezave pošlje računalniku. Pri naši raziskavi sta nas najbolj zanimali ravni α in β valov, saj so ti povezani s pozornostjo in meditacijo.

Slika 85: »NeuroSky MindWave Mobile 2«

(Vir: NeuroSky store [10])

Kandidatom smo najprej na glavo nastavili »MindWave Mobile 2«, ga povezali na računalnik preko Bluetooth povezave in na programu »Neuro experimenter« začeli zapisovati meritve. Kandidatom smo nato za posamezen test razložili, kaj morajo storiti. Pri prvi fazi raziskave so si kandidati morali ogledati enominutni posnetek delovanja industrijskih robotov, v drugi fazi so si ogledali nekaj slik, v tretji, zadnji fazi, pa so s telefonom uporabljali našo AR aplikacijo.

Po vseh zaključenih meritvah za vse faze smo zbrali podatke in jih združili po fazah. Od vseh meritev sta nas zanimali samo povprečna raven pozornosti in meditacije. Seveda so bili podatki za vsakega kandidata nekoliko drugačni, vendar so imeli za vsako fazo podobna odstopanja.



Slika 86: Graf pozornosti in meditacije

(Vir: osebni arhiv)

6.4 TABELE MERITEV

Tabela 1: Meritve Meditacije in pozornosti

Vir: Osebni Vir

Osebe	Meritev pozornosti – ogled videa	Meritev meditacije – ogled videa	Meritev pozornosti – ogled slike	Meritev meditacije – ogled slike	Meritev pozornosti – AR aplikacija	Meritev meditacije – AR aplikacija
Oseba 1	52,2348	52,2043	71,375	46	49,3247	37,7532
Oseba 2	48,6895	54,5108	15,8308	66,3231	26,6729	58,9626
Oseba 3	25,5414	53,3702	59,2969	85,8281	55,5273	55,8182
Oseba 4	64,6515	60,4293	52,0495	63,4059	47,0667	58,9
Oseba 5	45,1198	59,3226	34,6154	60,9231	70,514	72,215
Oseba 6	55,6198	52,6405	53,8846	42,1923	21,1402	54,9252
Oseba 7	40,4783	53,9565	53,1224	58,3776	42,1407	52,237
Oseba 8	55,2573	41,1602	57,0694	48,8056	33,2883	63,2883
Povprečje	48,4490	53,4493	49,6555	58,9819	43,2093	56,7624

Ugotovili smo, da sta bili ravni pozornosti in meditacije pri ogledu videoposnetka približno enaki pri vseh kandidatih. Vendar je izmed dveh rahlo prevladala pozornost, kar pomeni, da so bili kandidati kljub sorazmerno visoki pozornosti na dogajanje v videoposnetku dokaj sproščeni.

Pri ogledu fotografij smo prišli do drugačnih ugotovitev. Ravni pozornosti in meditacije sta narasli, domnevno zaradi manjše količine podatkov, ki so jih možgani kandidatov morali obdelati. Pri vseh je bila najbolj opazna dokaj visoka raven meditacije, iz česar lahko sklepamo, da je gledanje slik manj stresno.

Pri zadnji fazi meritev, uporabi AR aplikacije, smo opazili drastično zmanjšanje pozornosti, pri čemer je raven meditacije ostala ista.

Sklepamo, da je razlog za manjšo pozornost pri uporabi AR aplikacije odvisna od tega, kako se neka oseba osredotoča na določene dele aplikacije. Zato je bila pozornost tudi večja pri slikah, saj so te statične, medtem ko je pri AR aplikaciji več delov, ki spreminjajo osredotočenje osebe.

7 ANKETA

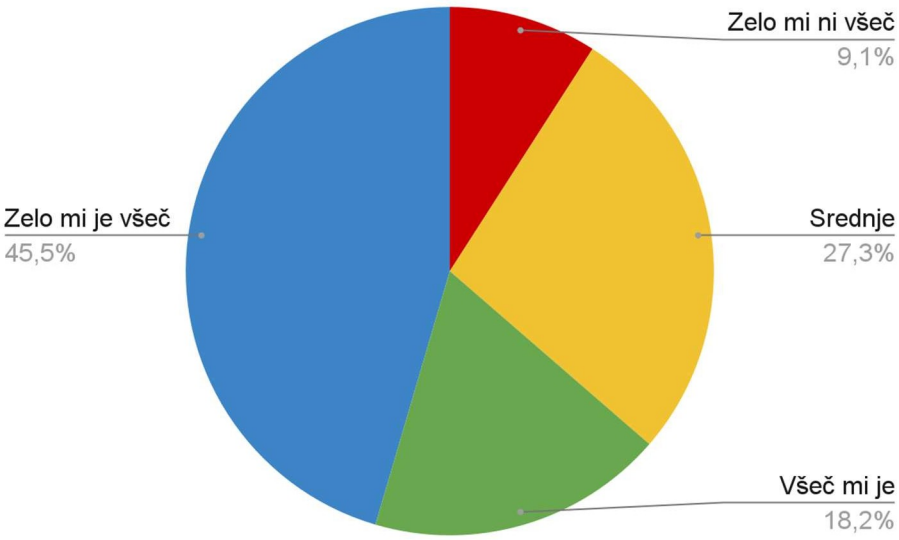
Ob pisanju raziskovalne naloge smo naredili dve anketi. Prva je bila splošna o AR tehnologiji, druga pa specificirana na robotiko in smo jo poslali le dijakom razreda M-4. c.

Pri prvi anketi smo prejeli 143 odgovorov. Večinoma jih je bilo iz starostne skupine do 20 let (70 %), 20 do 40 let jih je bilo najmanj (5,7 %), nad 40 let pa je bil preostali del (24,3 %). Večinoma jih je bilo iz naravoslovnih (39 %) in družbenih ved (29,1 %), medtem ko je 31,9 % odgovorilo, da nič od naštetega.

Anketa nam prikaže, da več kot polovica udeležencev ne prepozna definicije obogatene resničnosti, saj je le 37 % odgovorilo pravilno, medtem ko je 44,9 % pravilno odgovorilo šele ob pogledu na sliko. 64,5 % udeležencev meni, da obogatena resničnost poveča pozornost na določeno stvar, 31,2 % pa meni, da se pozornost ne spremeni.

Pri drugi anketi smo prejeli 11 odgovorov, ki so bila večinoma v prid AR tehnologij v robotiki.

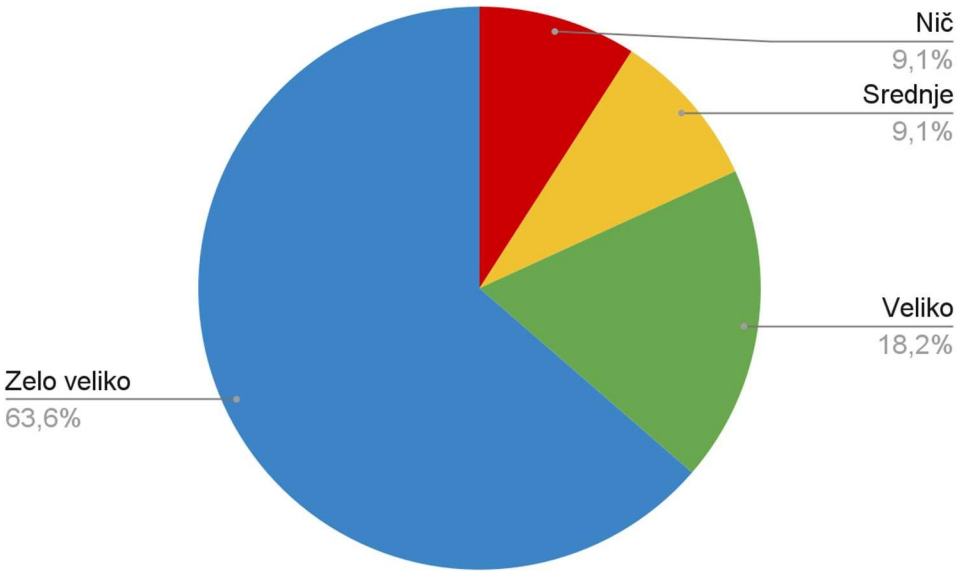
1. Kako Vam je všeč pouk robotike?



Graf 1: Vprašanje 1

Večina razreda M-4. c meni, da jim je všeč pouk robotike, medtem ko več kot četrtina nima posebnega mnenja.

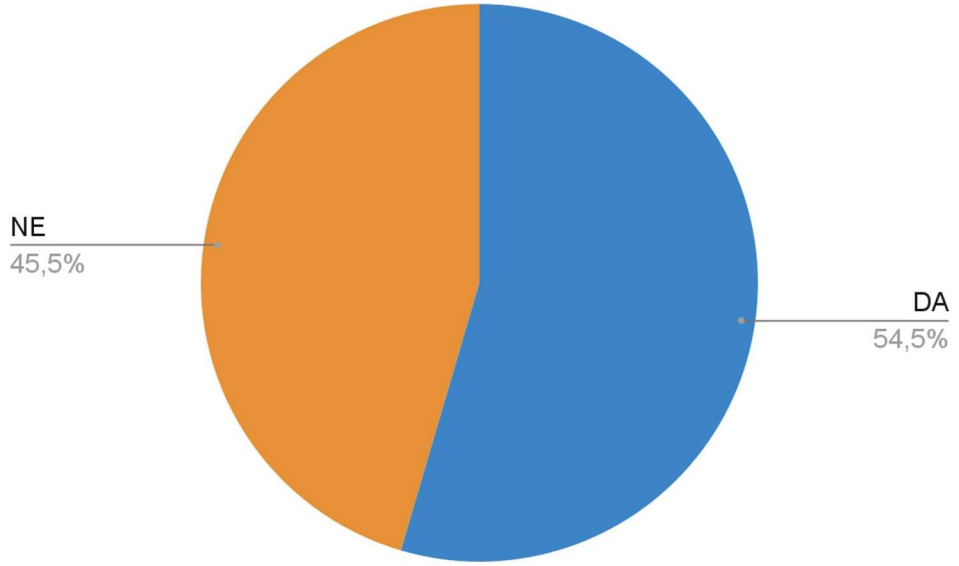
2. Koliko Vas zanima robotika?



Graf 2: Vprašanje 2

Več kot tri četrtine vseh dijakov meni, da jih zanima robotika, kar je več kot pri pouku robotike.

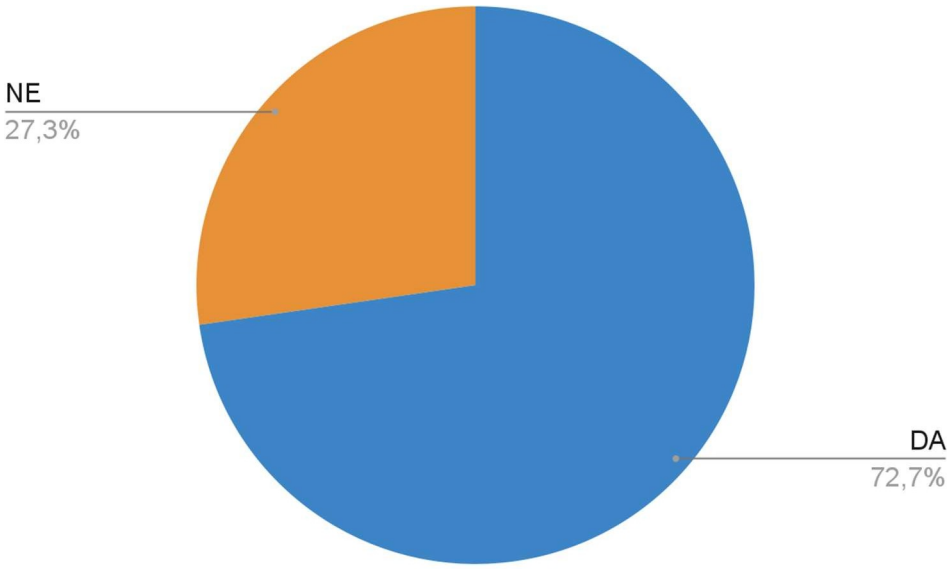
3. Ali ste že kdaj uporabljali AR?



Graf 3: Vprašanje 3

Večina vseh dijakov je že uporabljala AR, vendar le 54,5 %.

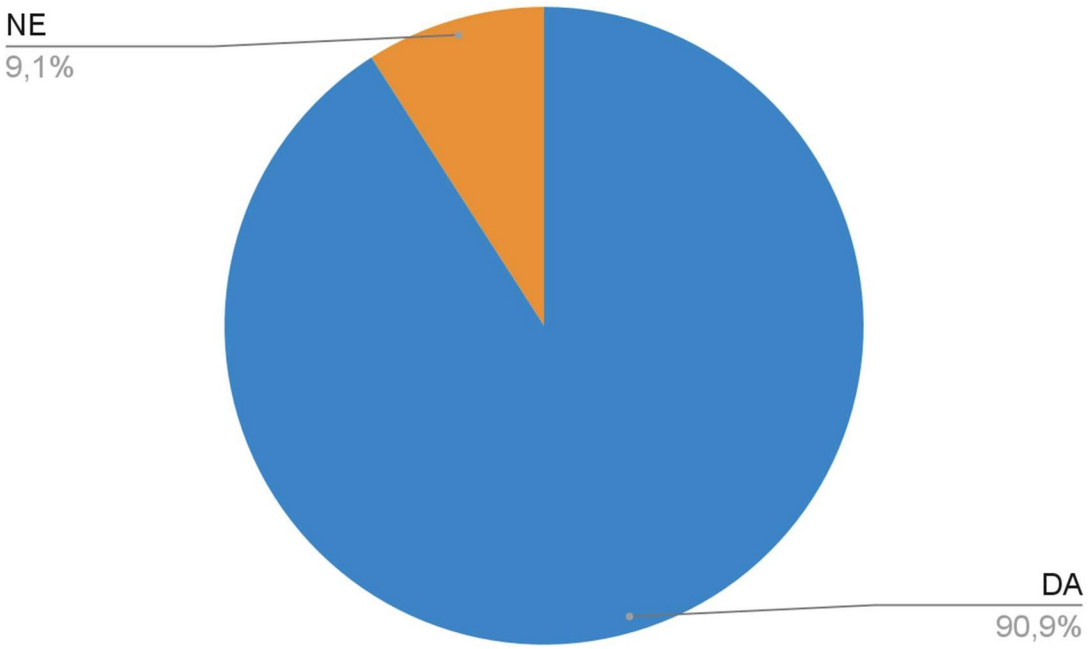
4. Ali ste že kdaj uporabljali VR?



Graf 4: Vprašanje 4

Izvemo, da imajo dijaki več izkušenj z navidezno kot pa z obogateno resničnostjo.

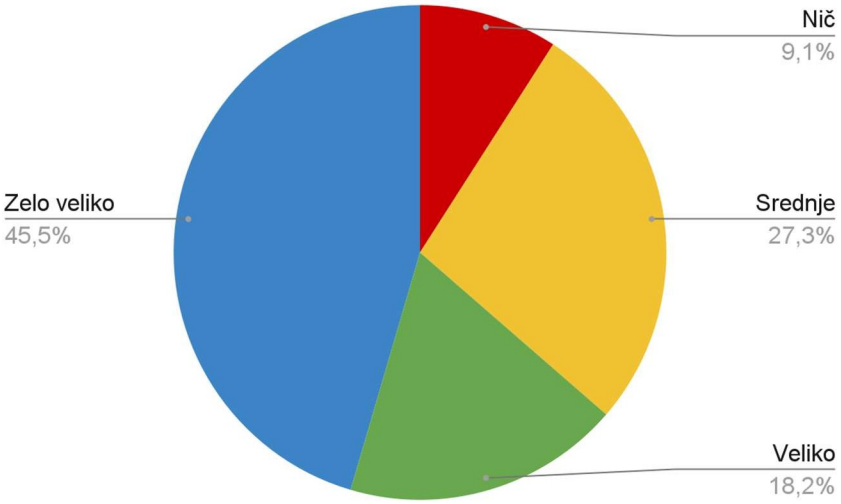
5. Ločite razliko med AR in VR?



Graf 5: Vprašanje 5

Večina dijakov loči razlike med AR in VR, kar je več kot v prvi, bolj splošni anketi.

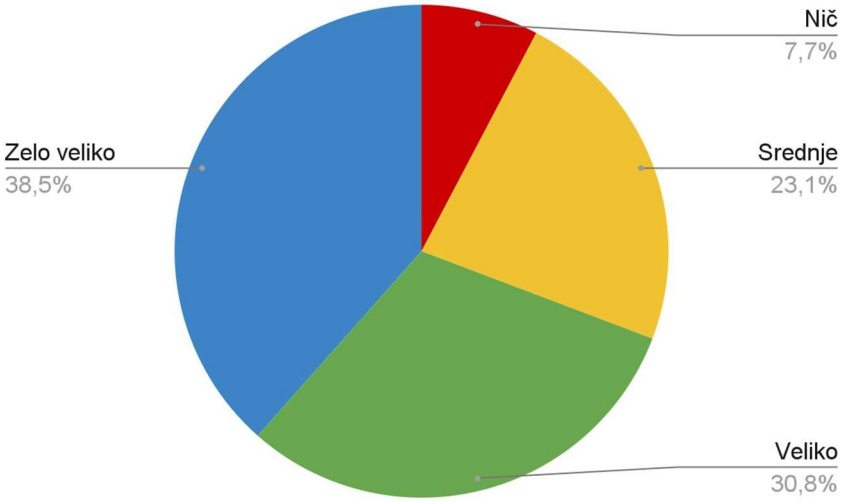
6. Bi Vas AR v robotiki bolj pritegnil k tej panogi?



Graf 6: Vprašanje 6

Večina razreda meni, da jih AR tehnologije bolj pritegnejo k robotiki.

7. Ali menite, da bi vas AR v povezavi z robotiko dodatno motivirala?



Graf 7: Vprašanje 7

Velika večina razreda meni, da jih AR bolj motivira pri pouku robotike.

Kot vidimo na odzivih, lahko sklepamo, da velika večina dijakov loči med AR in VR. Analize anket so pokazale, da bi z vključitvijo AR in VR tehnologij naredili pouk robotike zanimivejši in bi več dijakov pritegnili v to smer. Ankete kažejo na to, da so uporabniki bolj motivirani za učenje z uporabo sodobnih tehnologij.

8 UGOTOVITVE

Med raziskovanjem smo odgovorili na raziskovalna vprašanja, ki smo si jih zastavili na začetku. Pri raziskovanju smo imeli kar zapleteno nalogo, čeprav so se metode branja literature in anketiranja dijakov izkazale kot dobre.

Tabela 2: Hipoteze

Hipoteze	Stanje
Izdelano aplikacijo bomo lahko naložili na AR očala Vuzix Blade.	Ni potrjeno.
Aplikacijo lahko uporabimo kot navodilo za delo s strojem.	Potrjeno.
AR robot bo deloval na isti način kot realni robot.	Delno potrjeno.
Aplikacijo je relativno preprosto ustvariti in uporabljati v programskem okolju Unity.	Potrjeno.
Uporaba sodobnih tehnologij motivira uporabnike.	Potrjeno.

Večino tez smo potrdili, razen prve in tretje.

Analiza hipotez:

Izdelano aplikacijo bomo lahko naložili na AR očala Vuzix Blade.

Te hipoteze nismo mogli potrditi, saj so očala Vuzix blade neprimerna za uporabo AR aplikacije, saj niso imela ustrezno verzijo sistema Android. Zato smo se odločili za nadaljevanje naloge le na telefonih, ki imajo sistem android 7 ali več.

AR robot bo deloval na kot realni robot

Robot v AR se bo premikal tako, da se bodo deli vrteli okoli osi. Tako se lahko premika tudi robot, vendar ima ta več različnih načinov. Ta hipoteza je delno potrjena.

Aplikacijo je relativno preprosto ustvariti in uporabljati v programskem okolju Unity

Po ustvarjanju aplikacije smo potrdili, da je izdelava aplikacije relativno preprosta. Vendar smo imeli nekaj težav. Največja težava je bila nalaganje aplikacije na Android napravo. Pri tem smo imeli srečo, saj smo se v prostem času že srečali z njo. Pri tem je težava tudi v knjižnicah, ki bi jih Unity moral imeti nastavljeno avtomatsko, vendar smo nekatere datoteke morali zamenjati.

V veliki večini smo vse probleme, ki smo jih našli med ustvarjanjem aplikacije, hitro rešili s pomočjo dokumentacije in video posnetkov na YouTubu. Veliko nam je pomagalo tudi predznanje, ki smo ga dobili s programiranjem v Unity.

Glede uporabe je aplikacijo z lahkoto uporabljati, saj je potrebno le telefon usmeriti proti »Image Target« in se pokaže model.

Aplikacijo lahko uporabimo kot navodilo za delo s strojem-robotom

V aplikacijo smo dodali 3D-besedilo in video, ki se vidi, ne glede na to iz katerekoli smeri gledamo. Na besedilu so zapisane osnovne informacije o robotu. V videu vidimo primer uporabe.

Uporaba sodobnih tehnologij motivira uporabnike

To hipotezo smo potrdili z anketo, kjer so dijaki razreda M-4. c. odgovorili na vprašanje: »Ali menite, da bi vas AR v povezavi z robotiko dodatno motiviral?« Na vprašanje so večinoma odgovorili pozitivno. 81.9 % udeležencev je odgovorilo zelo motivirano ali veliko motivirano, medtem ko je le 9.1 % odgovorilo, da jih ne motivira.

Torej lahko sklepamo, da bi se več ljudi zanimalo za robotiko, če bi delali več s sodobnimi AR tehnologijami in bi zvišali zadovoljstvo delavcev.

9 MOŽNOSTI ZA NADALJEVANJE RAZISKOVANJA

Ob izdelovanju naloge smo ugotovili, da bi lahko raziskovanje v prihodnje nadgradili s sledečimi izboljšavami in nadgradnjami.

9.1 PRIJEMALKA NA ROBOTSKI ROKI

Možno bi bilo napisati program, ki odpira in zapira prijemalko. To bi lahko storili s pomočjo »local euler angles«, podobno kot pri programu za premikanje osi robota. Funkcijo za odpiranje in zapiranje robota je možno izvršiti preko UI gumba, kot pri »Premikanju osi brez virtualnih gumbov« ali z virtualnimi gumbi.

Prijemalo bi lahko še dodatno nadgradili s 3D-modeli, na katere bi lahko robotska roka vplivala. Kot na primer model preproste kocke, ki bi jo vstavljali v program s pomočjo »Plane Detection«. Kocka bi se nato prijela na prijemalko, če sta na istem mestu in pritisnemo gumb za prijem, in potem padla, ko pritisnemo gumb še enkrat.

9.2 IOS IN AR OČALA

Aplikacija, ki smo jo ustvarili, deluje na skoraj vseh android napravah, ki imajo android 7 ali več. Izjeme so tisti telefoni, ki jih »Vuforia« ali »AR Core« ne podpirata. Lahko bi povečali število telefonov, kjer deluje ta aplikacija, tako da bi naredili aplikacijo za IOS telefone. Problem pri tem je, da je izdelava Android aplikacije brezplačna, za IOS aplikacijo pa moramo biti registrirani kot Apple Developer, kar pa je potrebno plačati.

Aplikacijo bi lahko tudi naredili za AR očala, kot so Vuzix M400. Problem pri tem je cena očal. Najcenejša stanejo okoli 1000 €, vendar so neprimerna, saj nimajo Android OS, ki je večji ali enak 7. Očala z novejšimi android sistemi pa so po navadi 1 800 € ali več.

9.3 NADZOR AR MODELA S POMOČJO ROBOTA IN OBRATNO

V trenutnem stanju aplikacije lahko AR model premikamo le z gumbi ali drsniki, vendar bi lahko raziskali možnosti premikanja na isti način kot pravega robota, ampak bi zato potrebovali povezavo do robota. Razmišljali smo o možnosti, da bi se pri premikanju realnega robota

premikal tudi AR model ali pa druga možnost, da se pri premikanju AR modela premika tudi realni robot.

9.4 VKLJUČITEV UMETNE INTELIGENCE

Uporaba umetne inteligence v sodobnih učnih okoljih, kjer bi umetna inteligenca glede na zaznavo obraznih parametrov prilagajala AR vsebino.

10 ZAKLJUČEK

Tema naše raziskovalne naloge je v nas vzbudila veliko zanimanje, saj mladi o vplivu računalništva in robotike premalo razmišljamo.

Skozi raziskovanje smo se srečali s kar nekaj težavami, pri reševanju katerih smo izgubili dober delež časa. Ena od teh so bila očala Vuzix Blade, na katera smo poskusili naložiti naš program, a nam po veliko poskusih vseeno ni uspelo, ker se verzija Android na očalih ni ujemala s tisto v Unity okolju.

Ugotovili smo tudi, kako so različni ljudje zainteresirani za učenje na različne načine in kako deluje možgansko valovanje pri delanju le-tega.

Glede na analizo ankete se veliko ljudi zanima za AR, in to ne samo v robotiki. Veliko ljudi zanimajo tudi druge vrste realnosti, na primer virtualna realnost, in velika večina ljudi se je tudi že srečala z obema.

Pri samem raziskovanju nismo imeli preveč zapletene naloge, ker na spletu lahko najdeš veliko pomoči za programiranje in snov, povezane z raziskovalno nalogo, a smo za to morali malo globlje poiskati. Pri raziskovanju so se seveda tudi sproti našle nove informacije, razmišljanja in postavljala nova vprašanja.

Tehnologija se po svetu hitro širi in izboljšuje in kmalu bodo samo še naprave in roboti opravljali težka, fizična dela, medtem ko bodo ljudje skrbeli za servisiranje in ustvarjanje programov za vodenje le-teh, zato poskušamo narediti učenje programiranja čim lažje.

11 VIRI

[1] AR okolje [Fotografija na spletu], STEANTYCIP © 2022, pridobljeno 21. januar 2022 s https://steantycip.com.

[2] AR uporabljen za navodila [Fotografija na spletu], © Skupina stroka.si – vse pravice pridržane, pridobljeno 20. januar s https://www.stroka.si.

[3] *Augmented reality*, Interaction design foundation 2002, pridobljeno 24. februar 2022 s https://www.interaction-design.org/literature/topics/augmented-reality.

[4] *Augmented Reality*, Investopedia, pridobljeno 10. januar 2022 s https://www.investopedia.com/terms/a/augmented-reality.asp.

[5] *Augmented reality*, Wikipedija, pridobljeno 10. januar 2022 s https://en.wikipedia.org/wiki/Augmented_reality.

[6] *Augmented Reality Tutorial* | Gaze Interaction in Unity [Video datoteka], pridobljeno 10. januar 2022 s https://www.youtube.com/watch?v=OE66gtiF8QQ.

[7] *Brain Waves*, Science Direct, pridobljeno 24. februar 2022 s https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/brain-waves.

[8] Dokumentacija za Ienumerator, Copyright © 2022 Unity Technologies, pridobljeno 29. november 2021 s https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html.

[9] Dokumentacija za local euler angles, Copyright © 2022 Unity Technologies, pridobljeno 29. november 2021 s

https://docs.unity3d.com/ScriptReference/Transform-localEulerAngles.html.

[10] EEG Headset [Fotografija na spletu], © 2015 Neurosky, pridobljeno 12. februar 2022 s https://store.neurosky.com.

[11] *Generating Brain Waves*, the Power of Astrocytes, frontiers, pridobljeno 24. februar 2022 s https://www.frontiersin.org/articles/10.3389/fnins.2019.01125/full.

[12] Haynes, A.H. (2020). *What is Augmented Reality?* Dotdash Meredith, pridobljeno 23. februar 2022 s https://www.investopedia.com/terms/a/augmented-reality.asp.

[13] *How Augmented Reality Works*, pridobljeno 10. januar 2022 s https://computer.howstuffworks.com/augmented-reality.htm.

[14] *How to create an Augmented Reality App* [Video datoteka], pridobljeno 21. november 2021 s https://www.youtube.com/watch?v=MtiUx_szKbI.

[15] *How to create Virtual buttons with Vuforia AR & Unity3D* [Video datoteka], pridobljeno 26. november s https://www.youtube.com/watch?v=ElmzIq6stNI.

[16] *Intuitive Robot Programming Using Augmented Reality*, Science Direct, pridobljeno 23. februar 2022 s https://www.sciencedirect.com/science/article/pii/S2212827118300933.

[17] Primerjava AR-VR, Copyright OceanWP Theme by Nick, pridobljeno 10. december 2021 s https://azimstudio.com.

[18] *Review on Portable EEG Technology in Educational Research*, Science Direct, pridobljeno 23. februar 2022 s https://www.sciencedirect.com/science/article/abs/pii/S0747563217307173.

[19] Unity spletna stran, Copyright © 2022 Unity Technologies, pridobljeno 20. november 2021 s https://unity.com.

[20] Variable (computer science) Wikipedija, pridobljeno 10. januar 2022 s https://en.wikipedia.org/wiki/Variable_(computer_science)

[21] Vuforia Engine 9.0 Ground Plane Tutorial #1 [Video datoteka], pridobljeno 10. december 2021 s https://www.youtube.com/watch?v=oTgc9ozukyY.

[22] Vuforia spletna stran, © Copyright 2021 PTC, pridobljeno 22. november 2021 s https://developer.vuforia.com.