SREDNJA ELEKTRO IN RAČUNALNIŠKA ŠOLA VELENJE

Trg mladosti 3, Velenje

MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

IZDELAVA DIDAKTIČNE IGRE V ŠTIRIH OKOLJIH

(Python, Blender, Unity, Swift)

Tematsko področje: Računalništvo

Avtorji:

Luka Lah, 1. letnik

Samo Pungaršek Pritržnik, 1. letnik

Žan Novak, 1. letnik

Mentorja:

Mag. Karmen Grabant

Nedeljko Grabant, dipl. inž.

Velenje, 2017

Raziskovalna naloga je bila opravljena na ŠC Velenje, Elektro in računalniška šola, 2016/2017.

Mentorja:

Mag. Karmen Grabant, uni. dipl. inž.

Nedeljko Grabant, dipl. inž.

Datum predstavitve: marec 2017

Ključna dokumentacijska informacija

ŠD ŠC Velenje, šolsko leto 2016/2017

KG Didaktična igra

AV LAH Luka, PRITRŽNIK Pungaršek Samo, NOVAK Žan

SA GRABANT, Karmen in GRABANT, Nedeljko

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2017

LI 2017

IN IZDELAVA DIDAKTIČNE IGRE V ŠTIRIH OKOLJIH

TD Raziskovalna naloga

OP VIII, 39 str., 0 tab.,0 graf.,50 slik, 4 pril., 50 vir

IJ SL

JI sl

AI

Želje po igrah vedno bolj naraščajo, saj se mladi in stari želijo vedno bolj zabavati. Večina današnjih iger je narejenih zelo kakovostno, vendar lahko večina iger tudi škodi oziroma niso v višjem pomenu človeku koristne. Zato smo se odločili, da bomo ustvarili didaktično igro za preverjanje osvojenega znanja srednješolcev. Odločili pa smo se tudi, da bomo vključili štiri razvijalna okolja, ker smo si želeli večjega izziva in ker smo želeli narediti primerjavo le-teh. Igra naj bi bila narejena v različnih razvojnih okoljih v Pythonu, Blenderju, Unityju, Swiftu in prav tako v različnih programskih jezikih: Blender – Python, Swift - XCode in Unity - C++. Objaviti smo jo želeli, da deluje na najbolj razširjene operacijske sisteme, kot so na primer Android, Windows ter Mac OS in iOS (igro za vse omenjene operacijske sisteme lahko najdemo na naši spletni strani: www.preizkusisestudio.wixsite.com/studio).

Key words documentation

ND ŠC Velenje, 2015/2017

CX Didactic game

AU LAH Luka, PRITRŽNIK Pungaršek Samo, NOVAK Žan

AA GRABANT, Karmen in GRABANT, Nedeljko

PP 3320 Velenje, SLO, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2017

PY 2017

TI CREATING DIDACTIC GAME IN FOUR DIFFERENT INVIRONMENTS

DT RESEARCH WORK

NO VIII, 39 pag., 0 col.,0 gra.,50 pic., 4 anne., 50 sour.

LA SL

AL sl/en

AB

Desires for games have been increasing very fast because young and old people want to have more fun. Most of the games are well made nowadays but they are also used for bad purposes or in the higher meaning they are just not useful. Because of that we have decided to make a didactic game for testing knowledge in high school population. But we have also decided to make this game in four different development environments in order to compare these with each other, and due to a bigger challenge, too. The game was supposed to be in different development environments Blender, Python, Unity, Swift as well as in different program languages: Blender – Python, Swift - XCode and Unity - C++. We want to upload the game on the most extended operating systems such as Android, Windows, Mac OS and iOS (our game for all operating systems can be found on our website:www.preizkusisestudio.wixsite.com/studio).

Seznam okrajšav

% – odstotek

3D – tridimenzionalno

angl. – prevod iz angleškega jezika

API - angl. Application Programming Interface (vmesnik za programiranje aplikacij)

ASCII – angl. American Standard Code for Information Interchange (ameriški standard za zapis črkovnih, številskih, posebnih, krmilnih znakov, med katerimi ni znakov za šumnike, preglase)

CC – angl. CreativeCommons (kreativna skupnost)

dipl. – diplomirani

ERŠ – Elektro in računalniška šola

HTML – angl. Hyper Text Markup Language (slovensko jezik za označevanje nadbesedila)

http – angl. hipertext transfer protocole (nadbesedilni prenosni protokol)

HTTP – Hyper Text Transfer Protocol (nadbesedilni prenosni protokol)

inž. – inženir

ISO – angl. Interational Standard Organisation (Mednarodna organizacija za standardizacijo)

npr. – na primer

OS – operacijski sistem

oz. – oziroma

SDK – angl. Software Development Kit, SDK or "devkit" (paket za razvoj programske opreme)

sl. – slovensko

spl. – splet

str. – stran

ŠCVelenje – Šolski center Velenje

t. i. – tako imenovani

URL – enotni naslov vira in enolični krajevnik (angleško Uniform Resource Locator) je naslov spletnih strani v svetovnem spletu

wiki – Wikipedia

www – angl. world wide web (svetovni splet)

Kazalo vsebine

Kazalo slik

1 UVOD

Do sedaj je bilo narejenih veliko različnih iger različnih zvrsti. Vendar pa je večna iger strelskih oziroma nam ne koristijo, razen zabavi. Želeli smo ustvariti didaktično igro, ob kateri in od katere bi se kaj naučili. Ta igra naj bi bila zabavna in primerna za srednješolsko populacijo.
Med izdelavo igre pa smo naleteli na razne probleme in težave. Nekatera programska okolja imajo nekatere slabosti in druga druge. Večinoma se okolij med seboj ne da primerjati, saj so vsa okolja približno enako zmogljiva.
Želeno igro pa smo želeli narediti za različne operacijske sisteme, kot so Android, Windows, macOS ...

1.1 Hipoteze

Pred raziskovanjem smo si zastavili naslednje hipoteze:

1. Predvidevamo, da brez začetnega znanja pred izdelavo raziskovalne naloge dijaki 1. letnika lahko izdelajo uporabno didaktično igro.
2. Izdelana igra naj bi bila primerna za spletno prodajo (npr. Trgovina Play, Steam, Apple Store).
3. Predvidevamo, da bo didaktična igra primerna za srednješolsko populacijo. Znanja, ki se preverjajo, so iz naslednjih predmetov: informatika, kemija in osnove elektrotehnike.
4. Predvidevamo, da bodo vsa SDK-okolja približno enako zmogljiva in uporabna.
5. Domnevamo, da je najbolj zmogljiv programski jezik C++.

1.1.1 Ideja za izdelavo igre

Za didaktično igro smo si zamislili, da bomo ustvarili kviz za srednješolce. Kviz naj bi vključeval različne teme. Za začetek smo si izbrali 3 teme za kviz, in te so:

- informatika in računalništvo,
- kemija in
- osnove računanja električnih veličin.

Odločili pa smo se tudi, da bo vsak naredil svojo igro v drugačnem SDK-okolju in da bomo ta okolja na koncu primerjali.

Okolja naj bi na koncu primerjali in na osnovi pridobljenih izkušenj se bomo odločili, katero orodje je najboljše, ter izpostavili njihove prednosti in slabosti.

Vse igre, ustvarjene v času raziskovanja, pa bodo na spletni strani, ki bo navedena med viri, prav tako bo na spletni strani tudi utemeljitev in izbira najboljšega programa ter raziskovalna naloga (URL-naslov spletne strani lahko najdete na koncu virov).

2 PREGLED STANJA TEHNIKE

Danes že veliko ljudi naredi igre v veliko različnih programih. Te igre so lahko zelo dobre ali pa samo narejene za zabavo. Zdi pa se nam, da ustvarjalci še nikoli prej niso bili dijaki 1. letnika srednje šole. Zato smo se mi prijavili s to raziskovalno nalogo.

V to raziskovalno nalogo smo se usmerili, ker nas je zanimalo 3D-modeliranje, vendar ker smo vsi oboževalci računalniških iger, smo se odločili za izdelavo didaktične igre, ki bi za spremembo koristila nam in ljudem okoli nas.

2.1 Predstavitev okolij in izdelava igre

Vsako okolje smo posebej predstavili in predstavili nekaj zgodovine le-tega.

2.2 Python

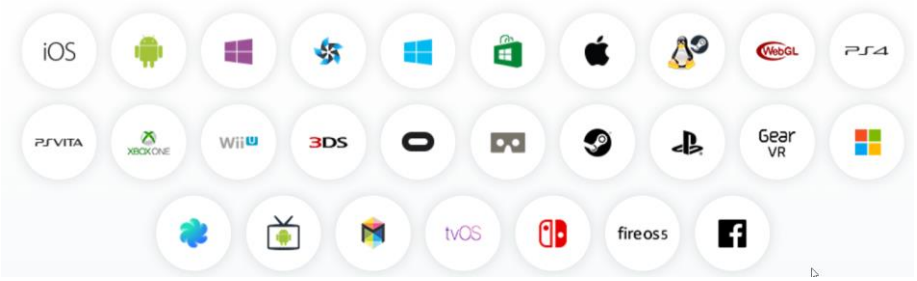
Python je programski jezik, ki ga je ustvaril Guido van Rossum leta 1989. Ime je dobil po angleški nadaljevanki Monty Python's Flying Circus.

Primeri sintakse Pythonovega programskega jezika so:

- seštevanje (1+2),
- odštevanje (2-1),
- množenje (2*1),
- deljenje (2/1),
- potenciranje (2**2),
- drugi koren iz a (math (sqrt(a))).

2.3 Unity

Unity je program za izdelovanje video-iger. Na začetku je bil izdelan samo za Nintendo. Čez čas so naredili module še za PC, Xbox one, android, iOS, macOS, Linux, PS4 (). Zdaj je Unity eden izmed najbolj razširjenih programov za izdelovanje iger. Zanj je značilno, da morajo ustvarjalci poznat C++, C# in JavaScript. Unity je bil izdan leta 2005. Leta 2014 je Unity dobil nagrado »Best engine« na podelitvi nagrad v Angliji. Zdaj Unity uporablja približno 1,3 milijonov ljudi po celem svetu.



Slika 1: Izvozi iz programa Unity

2.4 Kaj je Blender?

Blender je odprtokodno programsko orodje za grafično 3D-modeliranje, animiranje, komponiranje s post produkcijo, 3D-manipulacijo v realnem času ... V Blenderju je vgrajen programski jezik Python kot API, s katerim lahko uporabnik avtomatizira in dodatno razširi možnost edinstvenega programa. Z

novejšimi različicami se je Blender 2.5 močno približal svojim konkurentom, kot so Autodesk Maya, 3D-s Max, Cinema 4D … Sedanja različica Blenderja je 2.76b. Blender je lahko uporabljen na več platformah, kot so Windows, Linux, macOS. Blender uporablja OpenGl za izris uporabniškega vmesnika in za upodabljanje (slika 2).

Slika 2: Blender logo, [2]

Blender je ustvaril nizozemski animacijski studio. Glavni razvijalec programa je bil Ton Roosendaal. Yellova pesem z albuma Baby je Dance tako navdušila, da so dali programu ime Blender. Ko je Neo Geo kupilo drugo podjetje, sta Ton Roosendaal in Frank van Beek junija leta 1998 ustvarila Not a Number Technologies (NaN), da bi še naprej razvijala Blender. Leta 2002 je podjetje NaN bankrotiralo, zato je Roosendaal začel kampanjo »Osvobodimo Blender«. Septembra 2002 so uporabniki programa Blender zbrali dovolj denarja, da je Blender postal odprtokodni program.

1.1 XCode

XCode je popolno orodje za razvijalce Apple, ki ustvarjajo aplikacije za Mac, iPhone in iPad. XCode 4 je optimiziran za lažji in hitrejši razvoj aplikacij, saj so vsa orodja zbrana na enem mestu. Tako je možno razvijanje uporabniškega vmesnika, pisanje kode in razhroščevanje kode v enem samem oknu. XCodov IDE (integrirano razvojno okolje) analizira detajle projekta in tako odkrije napake bodisi logične ali napake v sintaksi.

XCode deluje na operacijskem sistemu macOS Sierra in vključuje XCode IDEinštrumente, simulator za iOS naprave (iPhone in iPad), najnovejše Mac OS X in iOS SDK-je (Software Development Kit, pribor za programiranje) ter stotine drugih zelo zmogljivih funkcij:

- Inovativna orodja, ki vam pomagajo pri ustvarjanju odličnih aplikacij.
- Poenostavljen vmesnik, ki je hiter in enostaven za uporabo.
- Profesionalni urejevalnik kode, ki vam pomaga, da je programer osredotočeni na kodo.
- Vgrajena Applova LLVM-tehnologija (infrastruktura prevajalnika napisana v C++ jeziku) poišče in popravi napake namesto nas.
- Inštrumenti za testiranje vizualnih zmogljivosti.

3 MATERIALI IN METODE DELA

V vsakem od programov Blender, Python, Unity in XCode smo ustvarili bolj kot ne enako didaktično igro, ki naj bi bila primerna za srednješolsko populacijo. Opisali smo tudi, kako smo ustvarili ikono ter kako smo vsako posamezno igro naredili.

3.1 Ustvarjanje ikone

Ikono smo ustvarili v odprtokodnem programu GIMP2, ker je bil naš prvotni namen imeti dva načina: časovno omejen način in standardnega. Ta načina delovanja smo želeli vključili v ikono igre. Odločili smo se, da je to bel vprašaj z rumeno ročno pobarvano strelo in barvo ozadja (html koda modrega ozadja ikone: #239dea - slika 3).

Slika 3: Ikona za igro, lasten vir

3.1.1 Kako smo izdelali ikono

V program GIMP2 smo dodali prozoren vprašaj in strelo, ki smo ju vzeli s spleta. Ikona je posebna tudi zato, ker je bila prej prozorna strela pobarvana na roke. Ker pa je ikona v obliki zaobljenega pravokotnika, smo morali v programu GIMP2 dodati alfa kanal, da je barvno ozadje postalo prozorno. Potem smo na vstavi izbrali zaobljen pravokotnik in pritisnili Ctrl | i, da smo obrnili izbor na zunanji del pravokotnika in ga izrezali. Sliko smo začeli risati v ločljivosti 1024x1024 p in jo izvozili kot sliko PNG (kasneje smo ji spremenili tudi velikost na 64x64 px).

3.2 Ustvarjanje kviza v Pythonu

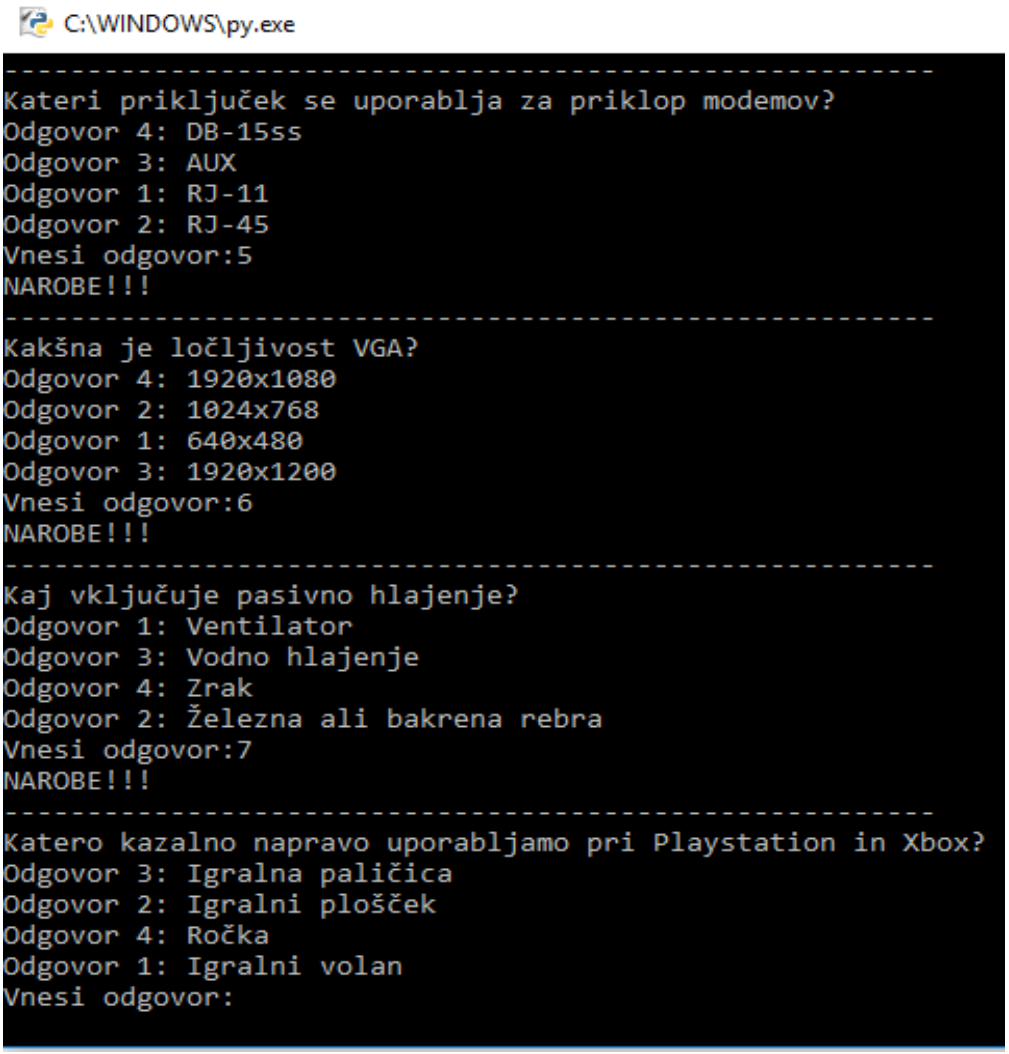
Ker program Blender vključuje kot API skriptni jezik Python, se nam je zdelo, da je bolje, če najprej naredimo kviz kar v Pythonu.

Kviz, ki smo ga naredili v Pythonu, smo želeli praktično uporabiti, zato smo dodali 45 vprašanj, iz katerih se jih 15 izbere naključno. Po 15 rešenih vprašanjih pa je kviz preveril stanje točk in podal oceno glede na odstotke doseženih točk, ki jih običajno uporabljajo v srednji šoli.

S tem smo se naučili uporabljati funkcijo Random in tako mogoče olajšali delo učitelju pri ocenjevanju. Ker je bila naša tema VIS (vzdrževanje informacijske strojne opreme), smo naredili temu primerna tudi vprašanja. Ker pa smo želeli, da bi dijaki pri reševanju kviza težje prepisovati smo si zamislili, da se tudi odgovori naključno pojavljajo.

Zamislili smo si tudi, da če igralec odgovori pravilno, se mu na zaslon izpiše beseda PRAVILNO!!!, če pa igralec odgovori narobe, se izpiše beseda NAROBE!!!

Da bi bilo vsako vprašanje bolj vidno, pa smo med vprašanji dodali tudi več črtkanih črt, ki ločujejo vprašanja (slika 4).



Slika 4: Primer izvajanja kviza, lasten vir

Da smo naredili takšen kviz, smo morali napisati 350 vrstic kode, kar sploh ni zastrašujoča številka, saj je potrebno v kodo vstaviti tudi vprašanja, odgovore in kateri odgovori so dejansko pravilni.

Sledi nekaj primerov kode, za izvedbo prejšnjega kviza.

Naslednja koda prikazuje, kako smo v program vnesli funkcijo random, kako smo naredili tabele, ki jih je prepoznalo kot vprašanja, podvprašanja … in kako smo zapisali vprašanja ter na kakšen način so zapisana (koda 1). V program pa smo vnesli tudi čas, kot je razvidno že iz slike. Vnesli pa smo ga zaradi morebitnega posodabljanja igrice na časovno omejena vprašanja.

kviz 3.0.py - G:\A-raziskovalna 2017\A-raziskovalna word\1- KVIZ\VIS\PYTHON\kviz v pythonu\kviz 3.0.py (3.5.2)

File Edit Format Run Options Window Help

```
import random
import time
st_vprasanj, st_odgovorov = 44, 4.
st_podvprasanj= 15
vprasanja = {}
rezultati = {}
odgovori = {}
vprasanja_num = {}
odgovori_num = {}



vprasanja[0] = "Kaj od naštetega je pribor za čiscenje?"
vprasanja[1] = "Po kakšnem imenu prepoznamo paralelni vmesnik?"
vprasanja[2] = "Koliko pinov ima USB tip A?"
vprasanja[3] = "Koliko pinov ima USB tip B?"
vprasanja[4] = "Kako z drugim imenom imenujemo priključek FireWire?"
vprasanja[5] = "Kakšna je hitrost priključka eSATA? "
vprasanja[6] = "Besedilni del na tipkovnici se drugače imenuje:"
vprasanja[7] = "Kakšno tipkovnico imamo v večini danes glede na postavitev tipk?"
vprasanja[8] = "Kaj od naštetega je komunikacijski protokol?"
vprasanja[9] = "Katera vrsta zaslonov uporablja katodno cev?"
vprasanja[10] = "Kaj je optična prepoznava znakov?"
vprasanja[11] = "Kolikšna je barvna globina na prvih računalnikih?"
vprasanja[12] = "Kakšno je drugo ime za video kamero?"
vprasanja[13] = "Katero kazalno napravo uporabljamo pri Playstation in Xbox?"
vprasanja[14] = "Katera je najhitreje delujoča zgoščenka?"
vprasanja[15] = "Česa spletna kamera navadno nima?"
vprasanja[16] = "Iz kod izvira QR koda?"
vprasanja[17] = "Kateri barvni model imajo tiskalniki?"
vprasanja[18] = "Za koliko je vijačnica obrnjena pri LCD zaslonu če je TN?"
vprasanja[19] = "Za koliko je vijačnica obrnjena pri LCD zaslonu če je DSTN?"
vprasanja[20] = "Kakšna je ločljivost VGA?"
vprasanja[21] = "Kakšna je ločljivost XGA?"
vprasanja[22] = "Kakšna je ločljivost WUXGA?"
vprasanja[23] = "Kakšno je navadno razmerje zaslona?"
vprasanja[24] = "Kakšna je navadna postavitev zvočnikov?"
vprasanja[25] = "Kako hiter je USB 1.1?"
vprasanja[26] = "Kako hiter je USB 3.1"
vprasanja[27] = "Kaj je CNC stroj?"
vprasanja[28] = "Kaj uporabljajo matrični tiskalniki?"
vprasanja[29] = "Kaj uporablja laserski tiskalnik?"
vprasanja[30] = "Kakšen barvni model ima plazma zaslon?"
vprasanja[31] = "Kolikšna je običajna starostna doba plazma zaslonov?"
vprasanja[32] = "Kaj od naštetega lahko pri zaslonih spremenimo s konfiguracijo?"
vprasanja[33] = "Katera je največja cifra pri osmiškem sestavu?"
vprasanja[34] = "Katera je največja cifra pri dvajsetiškem sestavu?"
vprasanja[35] = "Katera je največja cifra pri dvaintridesetiškem sestavu?"
vprasanja[36] = "Kaj vključuje pasivno hlajenje?"
vprasanja[37] = "Kaj je miška?"
vprasanja[38] = "Kaj je hitrejše zvok ali svetloba?"
vprasanja[39] = "Kateri tip USB ne spada zraven?"
vprasanja[40] = "Za kaj se uporablja VGA priključek?"
vprasanja[41] = "Za kaj se uporablja AUX priključek?"
vprasanja[42] = "Kako je drugače imenovan TRS priključek?"
vprasanja[43] = "Kateri priključek se uporablja za priklop modemov?"
vprasanja[44] = "Katera ni osnovna operacija miške?"


rezultati[0] = "3"
rezultati[1] = "2"
```

Koda 1: Prvi del Python kode prejšnje igre, lasten vir

V naslednjem delu koda prikazuje, kako in na kakšen način smo zapisali odgovore in kako izbrali, kateri je pravilen (koda 2).

kviz 3.0.py - G:\A-raziskovalna 2017\A-raziskovalna word\1- KVIZ\VIS\PYTHON\kviz v pythonu\kviz 3.0.py (3.5.2)

File Edit Format Run Options Window Help

```
rezultati[0] = "3"
rezultati[1] = "2"
rezultati[2] = "4"
rezultati[3] = "1"
rezultati[4] = "4"
rezultati[5] = "2"
rezultati[6] = "3"
rezultati[7] = "1"
rezultati[8] = "3"
rezultati[9] = "2"
rezultati[10] = "1"
rezultati[11] = "1"
rezultati[12] = "2"
rezultati[13] = "2"
rezultati[14] = "3"
rezultati[15] = "1"
rezultati[16] = "1"
rezultati[17] = "2"
rezultati[18] = "1"
rezultati[19] = "3"
rezultati[20] = "1"
rezultati[21] = "2"
rezultati[22] = "3"
rezultati[23] = "1"
rezultati[24] = "2"
rezultati[25] = "1"
rezultati[26] = "3"
rezultati[27] = "2"
rezultati[28] = "1"
rezultati[29] = "2"
rezultati[30] = "1"
rezultati[31] = "2"
rezultati[32] = "2"
rezultati[33] = "1"
rezultati[34] = "1"
rezultati[35] = "3"
rezultati[36] = "2"
rezultati[37] = "1"
rezultati[38] = "2"
rezultati[39] = "4"
rezultati[40] = "1"
rezultati[41] = "2"
rezultati[42] = "1"
rezultati[43] = "1"
rezultati[44] = "4"


odgovori[0,1] = "Odgovor 1: Ploscati izvijac"
odgovori[0,2] = "Odgovor 2: Torex izvijac "
odgovori[0,3] = "Odgovor 3: Stisnjen zrak "
odgovori[0,4] = "Odgovor 4: Zgoscenka"

odgovori[1,1] = "Odgovor 1: CTG"
odgovori[1,2] = "Odgovor 2: LPT"
odgovori[1,3] = "Odgovor 3: PV"
odgovori[1,4] = "Odgovor 4: PAP"

odgovori[2,1] = "Odgovor 1: 25"
odgovori[2,2] = "Odgovor 2: 3"
```

Koda 2: Drugi del kode, lasten vir

Iz tretjega dela kode je razvidno, kako smo naredili izgled vprašanja (koda 3). Razvidno je tudi, kako smo določili ocene in izpis sporočila po izbranem odgovoru. Naključno izbiro angl. random funkcijo smo uporabili znotraj vprašanja, da se odgovori naključno pojavijo.

Ker smo kviz naredili tako preprosto, lahko podatke, kot so vprašanja in odgovore, kadarkoli zamenjamo.

kviz 3.0.py - G:\A-raziskovalna 2017\A-raziskovalna word\1- KVIZ\VIS\PYTHON\kviz v pythonu\kviz 3.0

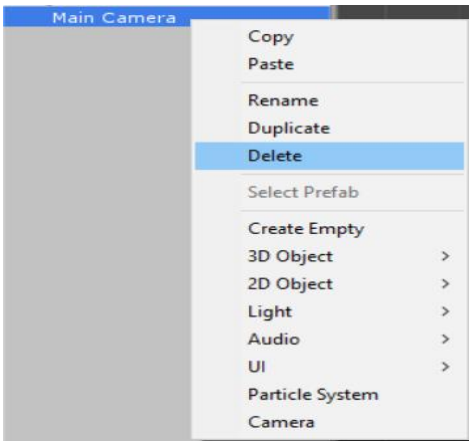
File Edit Format Run Options Window Help

```
odgovori[40,2] = "Odgovor 2: Za zvok"
odgovori[40,3] = "Odgovor 3: Za internet"
odgovori[40,4] = "Odgovor 4: Za Bluetooth"

odgovori[41,1] = "Odgovor 1: Za video"
odgovori[41,2] = "Odgovor 2: Za zvok"
odgovori[41,3] = "Odgovor 3: Za internet"
odgovori[41,4] = "Odgovor 4: Za Bluetooth"

odgovori[42,1] = "Odgovor 1: AUX"
odgovori[42,2] = "Odgovor 2: MIDI"
odgovori[42,3] = "Odgovor 3: USB"
odgovori[42,4] = "Odgovor 4: LTP"

odgovori[43,1] = "Odgovor 1: RJ-11"
odgovori[43,2] = "Odgovor 2: RJ-45"
odgovori[43,3] = "Odgovor 3: AUX"
odgovori[43,4] = "Odgovor 4: DB-15ss"

odgovori[44,1] = "Odgovor 1: Klik"
odgovori[44,2] = "Odgovor 2: Dvojni klik"
odgovori[44,3] = "Odgovor 3: Premikanje"
odgovori[44,4] = "Odgovor 4: Pristanek"



vprasanja_num = random.sample(range(0,44),44)

a=0
b=0
for index in range(st_vprasanj):
    vprasanje = random.randint(0,44)
    odgovori_num = random.sample(range(0, 4), 4)
    print("------------------------------------------------------------")
    print(vprasanja[vprasanja_num[index]])
    print(odgovori[vprasanja_num[index],odgovori_num[0]+1])
    print(odgovori[vprasanja_num[index],odgovori_num[1]+1])
    print(odgovori[vprasanja_num[index],odgovori_num[2]+1])
    print(odgovori[vprasanja_num[index],odgovori_num[3]+1])

    odg = input("Vnesi odgovor:")
    if odg==rezultati[vprasanja_num[index]]:
        print("PRAVILNO !!!")
        a = a + 1
    else:
        print("NAROBE!!!")
    b = b + 1
    if b == st_podvprasanj:
        break
if a<7 :
    print("NZD (1)")
elif a<9 :
    print("ZD (2)")
elif a<12 :
    print("DB (3)")
elif a<14 :
    print("PDB (4)")
else:
    print("ODL (5)")
input()
```

Koda 3: Tretji del kode, lasten vir

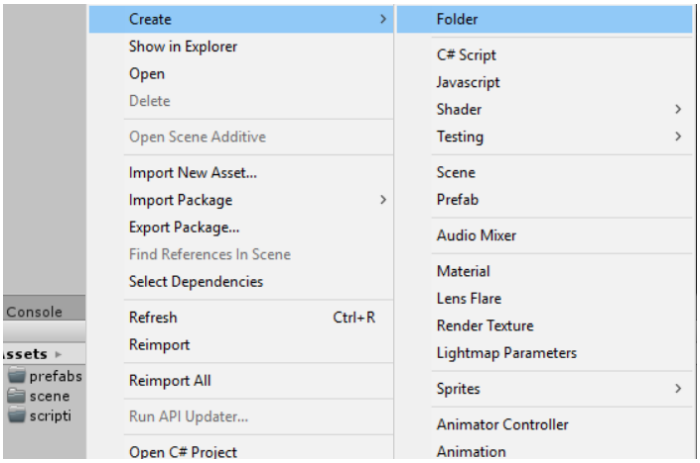
3.3 Ustvarjanje igre v okolju Unity

Na začetku smo najprej vnesli osnovne podatke o projektu. Nato pa smo izbrisali kamero (slika 5).



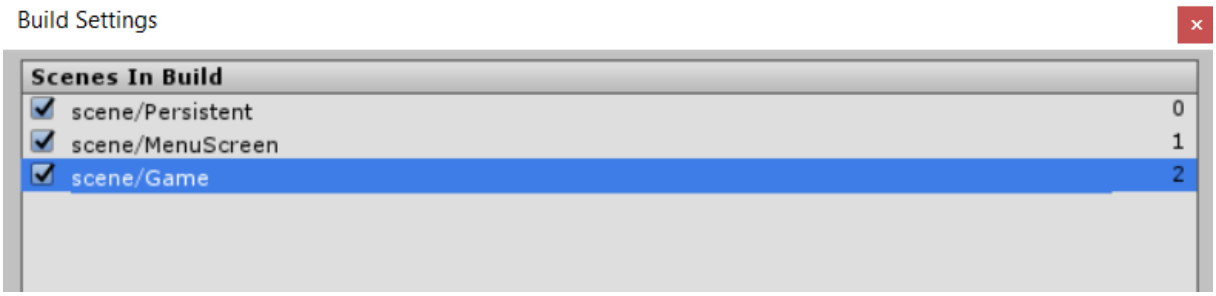
Slika 5: Izbris kamere, lasten vir

Potem smo ustvarili tri mape, po imenu Scripti, Prefabs in Scene (slika 6).



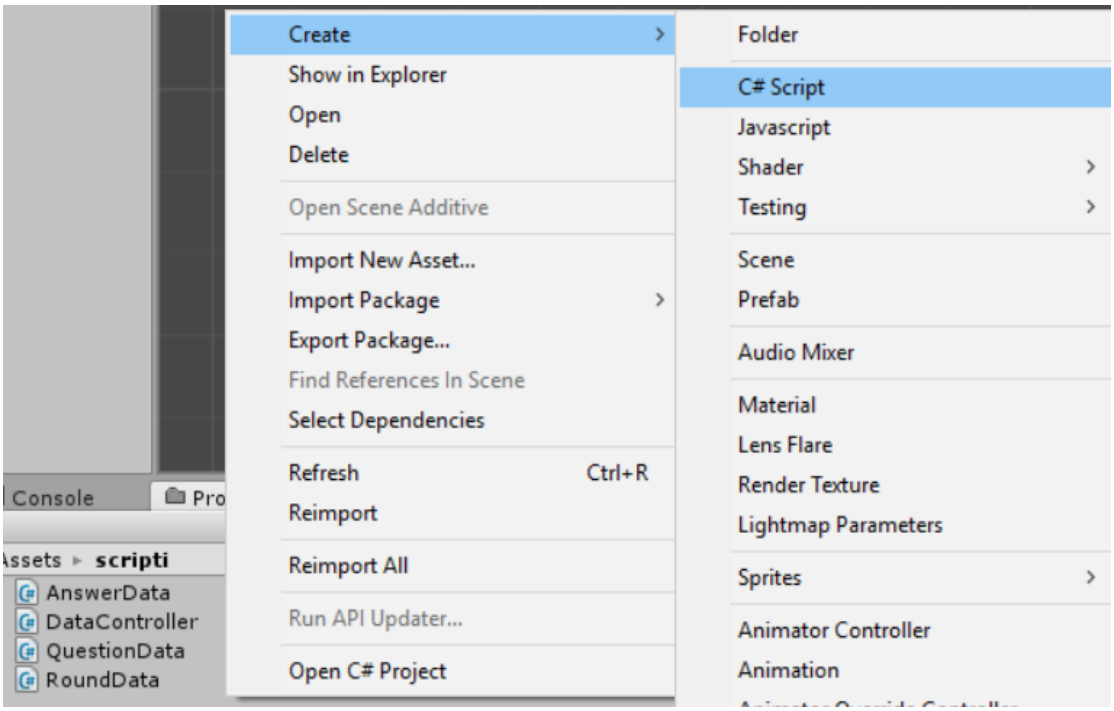
Slika 6: Ustvarjanje map, lasten vir

Zatem smo ustvarili tri scene z imeni: Persistent, MenuScreen in Game, da bi le-te lažje prepoznali (slika 7).



Slika 7: Izdelava scen, lasten vir

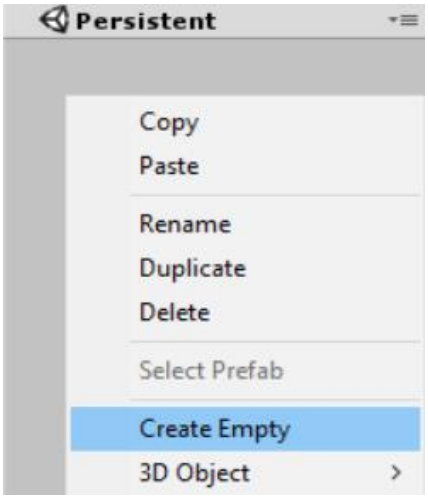
Nato smo nadaljevali s pisanjem kode C# Script (slika 8).



Slika 8: Izdelava programskih kod, lasten vir

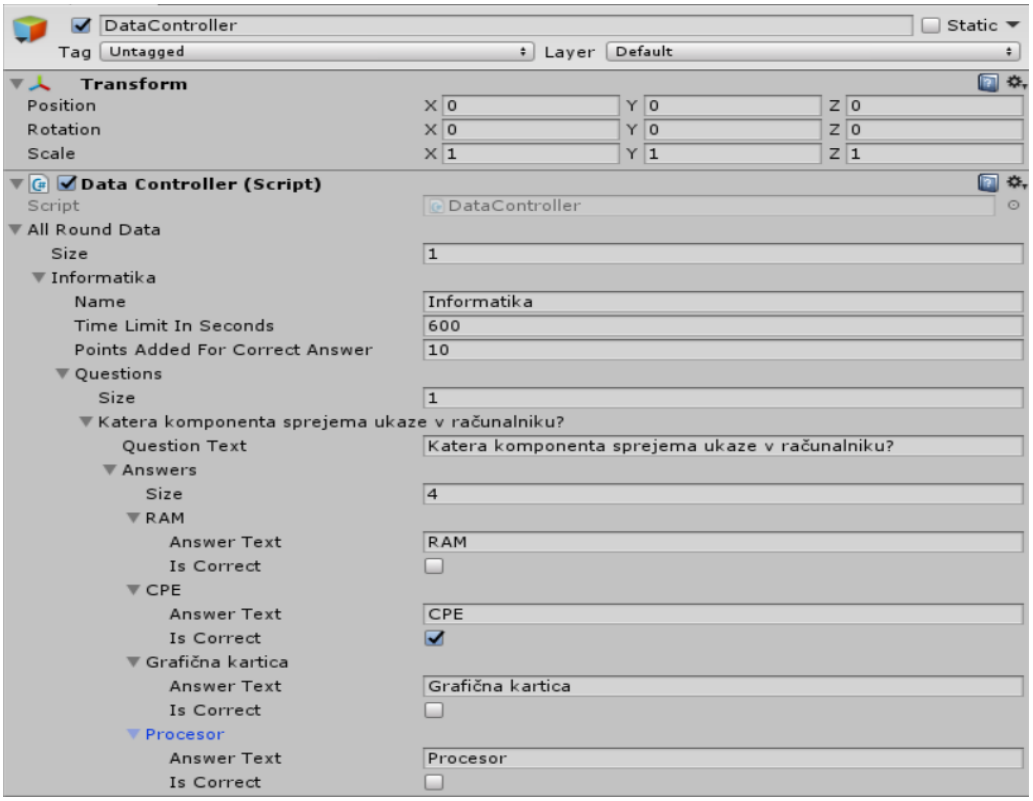
Vse kode povezuje ena koda z imenom »DataController«.

Potlej smo ustvarili prazen objekt »game object« po imenu »DataController«, vanj smo dodali tudi kodo »DataController« (slika 9).



Slika 9: Izdelava objekta DataController, lasten vir

Za tem smo dodali vprašanja (slika 10).

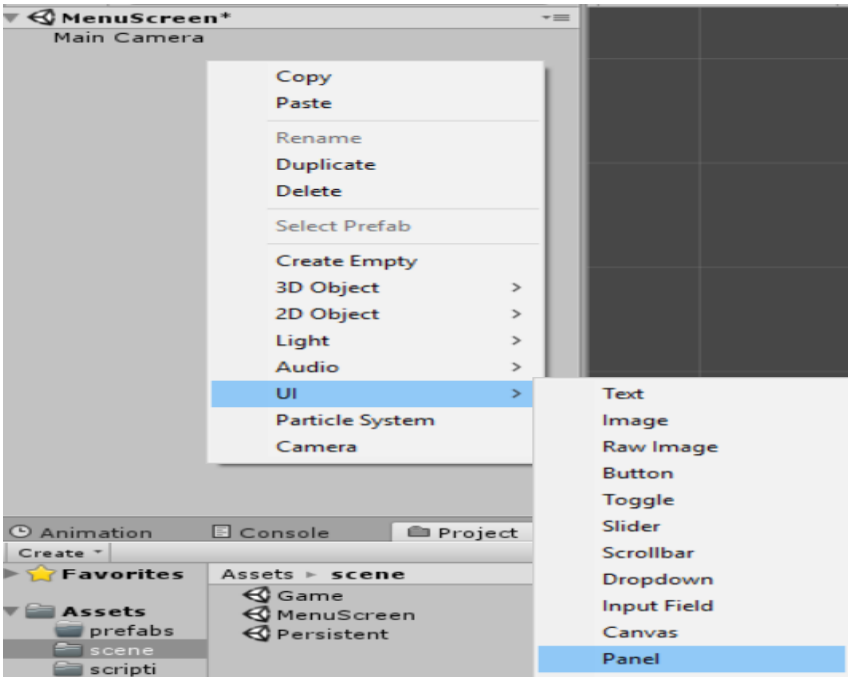


Slika 10: Tukaj vidimo eno izmed vprašanj, lasten vir

3.3.1 Meni v okolju Unity

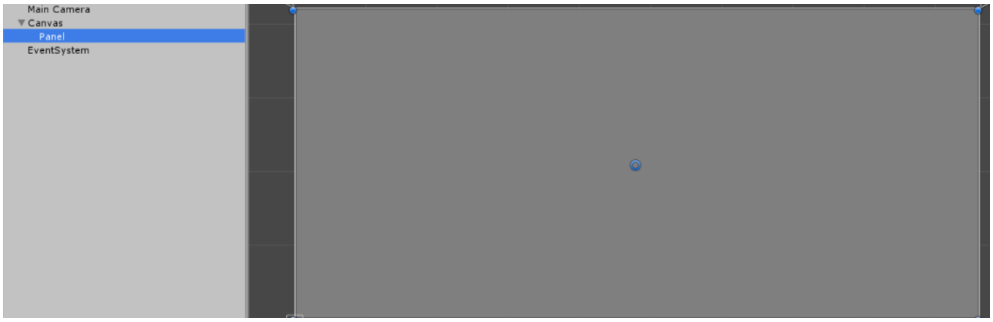
Sedaj začnemo izdelovati naš meni igre.

Najprej ustvarimo panel kot uporabniški vmesnik, zato da lahko začnemo vstavljati gumbe in besedilo na sceno (slika 11).



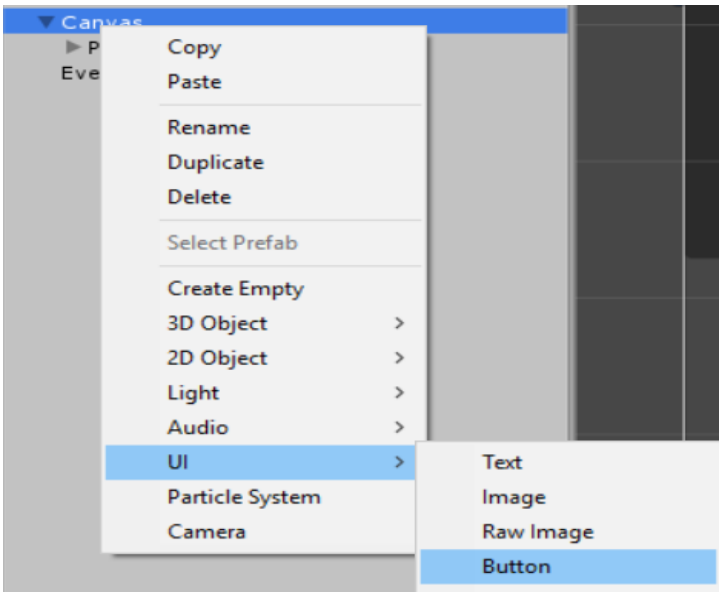
Slika 11: Izdelava panela, lasten vir

Desno na naslednji sliki je viden panel (slika 12).



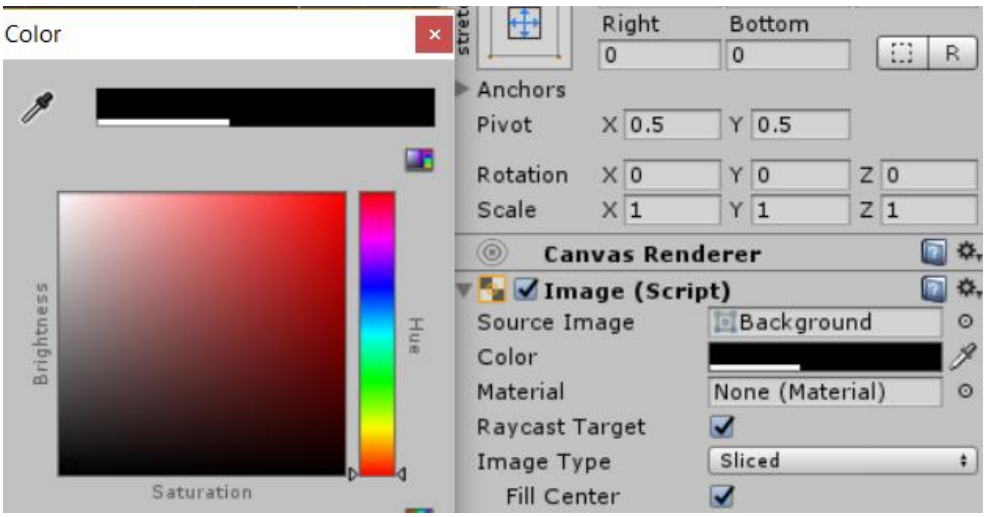
Slika 12: Panel, lasten vir

Nato dodamo gumb, imenovan začetni gumb t. i. »StartButton«, ki nam omogoča začetek igre (slika 13).



Slika 13: Gumb "Start button", lasten vir

S klikom v polje Color spremenimo barvo gumba na rdečo, saj je tako gumb bolje viden igralcu (*slika 14*).



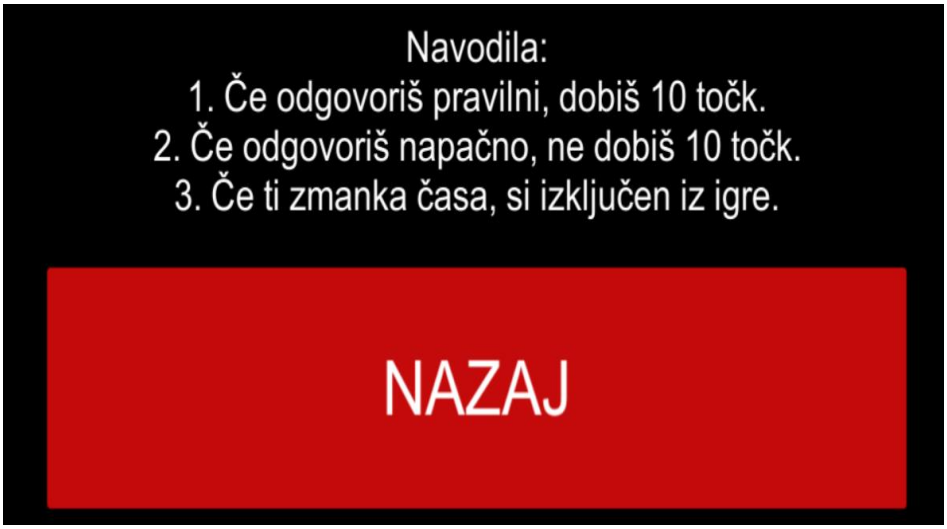
Slika 14: Spreminjanje barve gumba, lasten vir

Za tem naredimo še en gumb imenovan »navodila«. Ko smo končali s tem, pa lahko začnemo dodajati še eno kodo (MenuSceneController) h GameObject-u. Nato pa smo dodali besedilo k panel-u. V besedilu je pisalo Preizkusi.se Studio. Izgled končane scene je viden na naslednji sliki (slika 15).

Slika 15: Slika začetnega zaslona igre, lasten vir

3.3.2 Navodila za igro kot scena igre

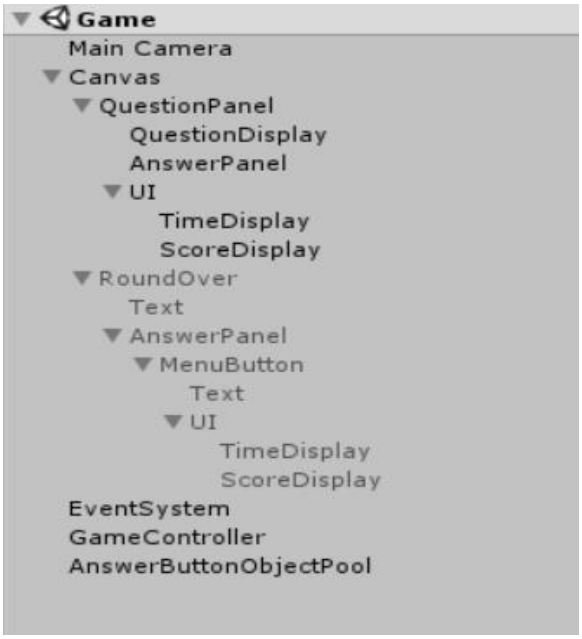
Zatem naredimo novo sceno (»navodila«). V to sceno najprej dodamo panel. V panel pa smo dodali besedilo. Nato smo dodali še gumb, ki igralca usmeri nazaj na glavni meni. Da bi to preusmerjanje delovalo, smo ustvarili novo kodo z imenom »UIController«. Končana scena je razvidna iz slike spodaj (slika 16).



Slika 16: Navodila, lasten vir

3.3.3 Glavna scena

Najprej smo vstavili panel z imenom »QuestionPanel«. Nato pa smo v panel vstavili besedilo »Question«, še en manjši panel »QuestionDisplay« in še en panel »AnswerPanel«. V »AnswerPanel« smo vstavili gumb »AnswerButton«. Nato smo vstavili še en panel »RoundOver«. Vanj smo vstavili gumb, ki te po končani igri preusmeri nazaj na glavni meni. Orisni pogled na uporabniški vmesnik vidimo v naslednji sliki (slika 17).



Slika 17: Angl. "Outliner" za scene, lasten vir

Na prejšnji sliki lahko vidimo še besedilo za štetje točk (ime ScoreDisplay) in besedilo, na katerem se odšteva čas (ime TimeDisplay). Končana scena je razvidna iz slike spodaj (slika 18).

Slika 18: Izgled vprašanja z odštevanjem časa in zbiranjem točk, lasten vir

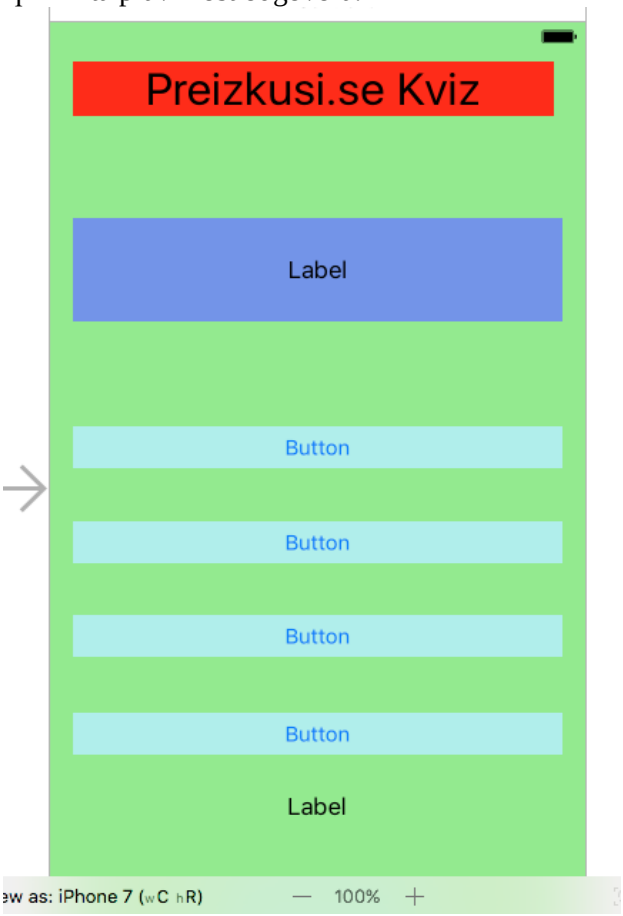
3.4 Kaj potrebujem za razvoj aplikacij v okolju XCode?

Za razvoj aplikacij v XCode potrebujemo Mac računalnik.
Pri izbiri med Apple računalniki je praktično vseeno, saj so priporočene zahteve programa XCode takšne, da lahko program zaženemo brez problemov iz Mac Mini-ja, Mac Pro ali MacBook Aira. Seveda je priporočljivo, da se uporablja čim boljši računalnik, da se poveča zmogljivost in zmanjša čakalni čas (slika 19).

Slika 19: Podprti Apple računalniki, ki lahko poganjajo XCode, vir: www.apple.com/mac/compare/, 5. 2. 2017

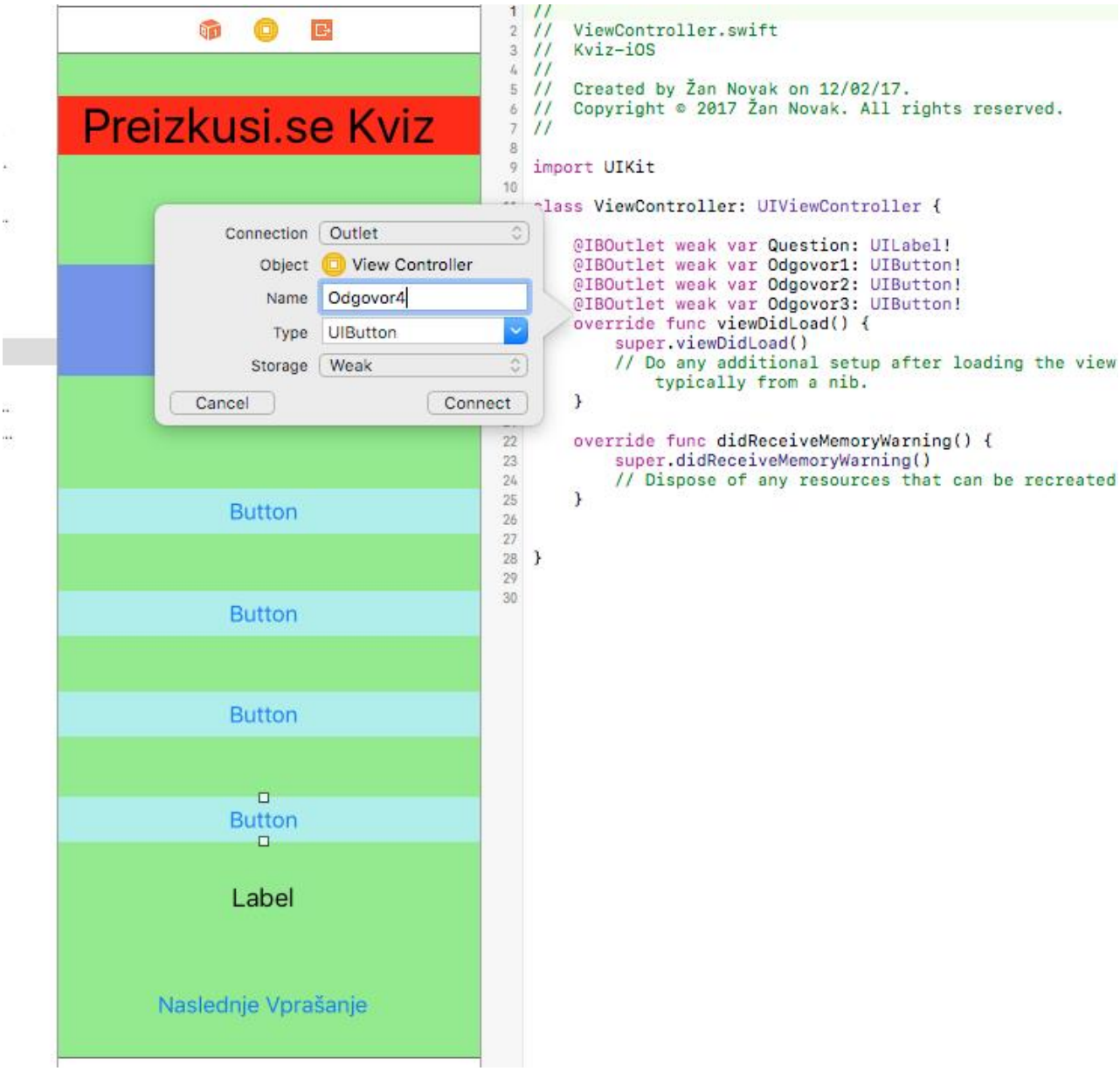
3.4.1 Izdelava igre v okoljuXCode

Za izdelavo aplikacije smo v okolju XCode izbrali Single view aplication z razlogom, da bo igra imela samo en pogled (kviz), nato smo potrdili pogoje izdelave aplikacije. Pred začetkom je bilo potrebno narediti tudi Apple Developer ID, kar je v bistvu navaden Apple ID z dodatkom za digitalno podpisovanje aplikacije (sledenje aplikacije po internetu, na računalniku ...). Ko nam Apple potrdi razvojni račun, imamo »zeleno luč«, da izdelamo aplikacijo.
Ko se nam odpre in naloži orodje XCode, smo v njem izbrali Main.storyboard. Tam smo nato naredili uporabniški vmesnik (angl. UI -slika 20). Najprej smo naredili ogrodje aplikacije: izberemo barve, kakšen bo izgled aplikacije ... Dodali smo napise angl. label, eden za naslov in za šepet za gumbe. Ozadje naslova aplikacije smo pobarvali rdeče. Na napis z oznako label se prikaže vprašanje. Pod tem imamo štiri gumbe, na katerih se izpisujejo vprašanja. In nato še dva napisa angl. label, ki nam sporočita pravilnost odgovora.



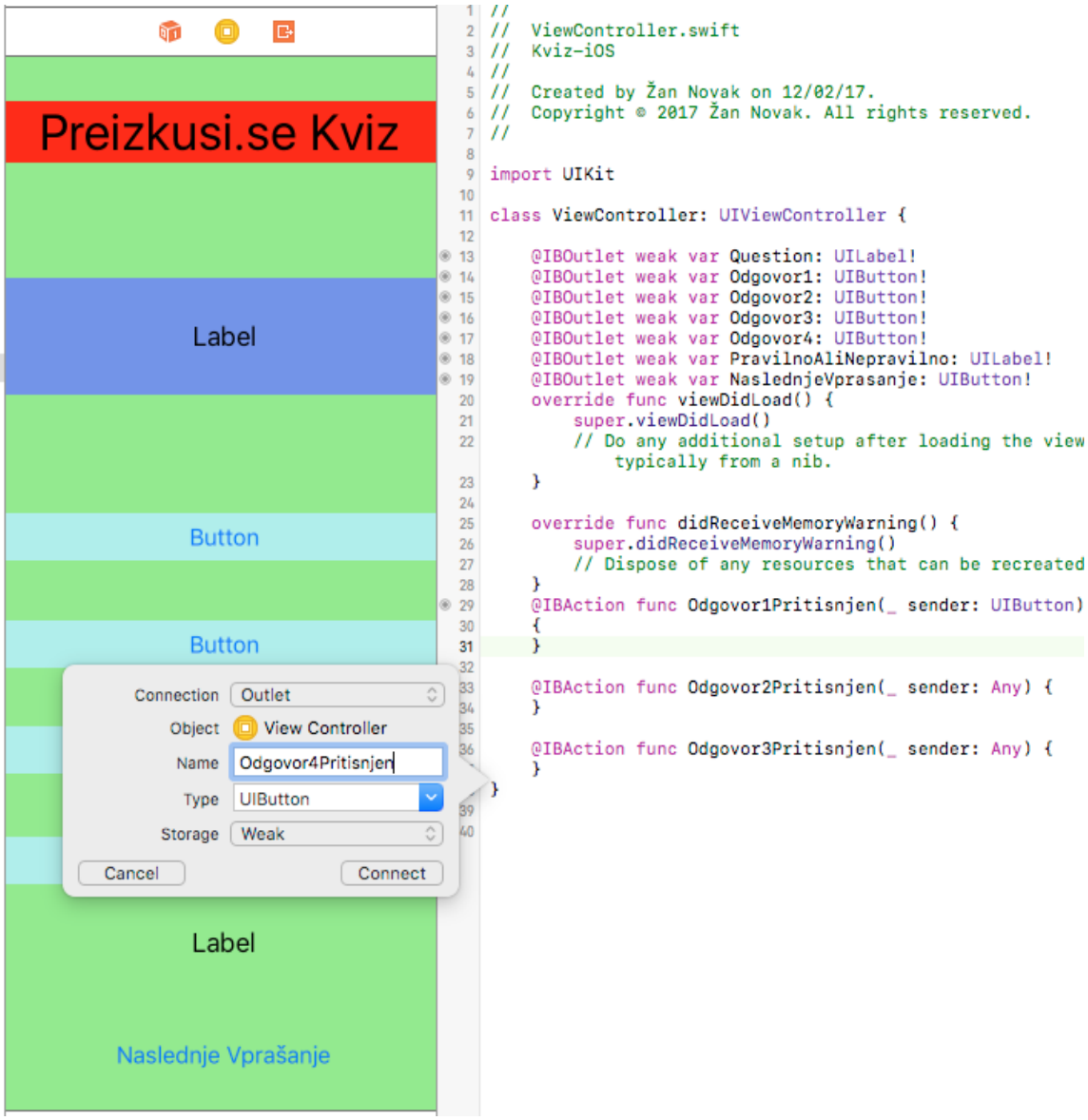
Slika 20: Uporabniški vmesnik za igro v XCode, osnova za 1 verzijo aplikacije, lasten vir

S pomočjo funkcije @IBOutlet (slika 21), samo z desnim klikom povali gumbe in napise (labele) v kodo in jih s tem naredimo aktivne, kar pomeni, da se nato pokažejo v kodi kot povezave in/ali hiperlinki.



Slika 21: Spreminjanje lastnosti uporabniškega vmesnika @IBOutlet, lasten vir

Posledično smo nastavili, kaj bodo naredili gumbi, ko bodo pritisnjeni, in kako se odzivajo na to akcijo. Potrebno je tudi nastaviti, kakšna animacija se bo predvajala na gumbu, ko bo ta pritisnjen (kaplja, strela ...) (slika 22).



Slika 22: Spreminjanje lastnosti @IBOutlet za pritisnjen gumb, lasten vir

S tem ko smo nastavili lastnosti gumba, ko je pritisnjen, smo odpri tudi možnosti za nadaljnjo izdelavo sistema za seštevanje točk in beleženja zgodovine pritiskov gumba (z namenom, da se ne ponavlja isti odgovor na istem mestu).

Ko smo pri pisanju kode končali s klicanjem gumbov in label v kodo, začnemo urejevat pogled t. i. viewcontroler.swift. V tem kontrolerju smo naredili večino kode, ki nato poganja aplikacijo. Zastavili smo spremeljivko funkcije za številko vprašanja. Nato smo naredili pogojne primere “case 0,1,2,3,4”. Ko naredimo kodo za “Case 0”, lahko kopiramo kodo in jo prilepimo pod “case 1,2,3,4,” nato smo spremenili pogoje v narekovajih in s tem končali s tem delom kode (slika 23).

```
override func viewDidLoad() {
    super.viewDidLoad()
    stevilkavprasanja = Int(arc4random_uniform(5))
    switch stevilkavprasanja{
    case 0:
        Vprasanje.text = "Kaj je BIT?"
        ODG1.setTitle("Byte", for: UIControlState.normal)
        ODG2.setTitle("8 Bytov", for: UIControlState.normal)
        ODG3.setTitle("Najmanjsa Osnovna Enota", for: UIControlState.normal)
        ODG4.setTitle("1/8 Zloga", for: UIControlState.normal)
        break
    case 1:
        Vprasanje.text = "Kaj je Informacija?"
        ODG1.setTitle("Podatek ", for: UIControlState.normal)
        ODG2.setTitle("Pojem Ali Predstava", for: UIControlState.normal)
        ODG3.setTitle("Predznanje", for: UIControlState.normal)
        ODG4.setTitle("Vrsta Kodnega Zapisa", for: UIControlState.normal)
        break
    case 2:
        Vprasanje.text = "Koliko Bytov ima Bit?"
        ODG1.setTitle("8", for: UIControlState.normal)
        ODG2.setTitle("1/8", for: UIControlState.normal)
        ODG3.setTitle("16", for: UIControlState.normal)
        ODG4.setTitle("64", for: UIControlState.normal)
        break
    case 3:
        Vprasanje.text = "Kaj je Podatek?"
        ODG1.setTitle("Byte", for: UIControlState.normal)
        ODG2.setTitle("Bit", for: UIControlState.normal)
        ODG3.setTitle("MegaByte", for: UIControlState.normal)
        ODG4.setTitle("Gigabyte", for: UIControlState.normal)
        break
    case 4:
        Vprasanje.text = "Katera od naštetih naprav je analogna?"
        ODG1.setTitle("Pečica", for: UIControlState.normal)
        ODG2.setTitle("El. Motor", for: UIControlState.normal)
        ODG3.setTitle("Računalnik", for: UIControlState.normal)
        ODG4.setTitle("Kavomat", for: UIControlState.normal)
        break
```

Slika 23: Primer kode, s katerim se programu prenesejo podatki, ko je gumb pritisnjen, lasten vir

Nato smo postavili še pogoje, kaj koda naredi, ko je gumb pritisnjen. Tako kot prej smo napisali za 1 pogoj in nato kopirali naprej, s tem da smo prilagodili spreminjanje pogojev. S tem smo osnovo aplikacije končali. Nato smo vstavili še ostala vprašanja (okoli 130) in se pripravili na izvoz aplikacije iz SDK-okolja XCode (slika 24).

```
        // Dispose of any resources that can be recreated.
    }

    @IBAction func ODG1P(_ sender: AnyObject) {
        if stevilkavprasanja == 0 {
            PRNP.text = "NAROBE!"
            NextQuestion.isHidden = false
        } else if stevilkavprasanja == 1 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 2 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 3 {
            PRNP.text = "Pravilno!"
        }else if stevilkavprasanja == 4 {
            PRNP.text = "NAROBE!"
        }
    }
    @IBAction func ODG2P(_ sender: AnyObject) {
        if stevilkavprasanja == 0 {
            PRNP.text = "NAROBE!"
            NextQuestion.isHidden = false
        } else if stevilkavprasanja == 1 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 2 {
            PRNP.text = "Pravilno!"
        }else if stevilkavprasanja == 3 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 4 {
            PRNP.text = "NAROBE!"
        }
    }
    @IBAction func ODG3P(_ sender: AnyObject) {
        if stevilkavprasanja == 0 {
            PRNP.text = "NAROBE!"
            NextQuestion.isHidden = false
        } else if stevilkavprasanja == 1 {
            PRNP.text = "Pravilno!"
        }else if stevilkavprasanja == 2 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 3 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 4 {
            PRNP.text = "NAROBE!"
        }
    }
    @IBAction func ODG4P(_ sender: AnyObject) {
        if stevilkavprasanja == 0 {
            PRNP.text = "NAROBE!"
            NextQuestion.isHidden = false
        } else if stevilkavprasanja == 1 {
            PRNP.text = "Pravilno!"
        }else if stevilkavprasanja == 2 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 3 {
            PRNP.text = "NAROBE!"
        }else if stevilkavprasanja == 4 {
            PRNP.text = "NAROBE!"
        }
    }
```

Slika 24: Programu ukažemo,katera dejanja mora narediti, ko se pritisne določen gumb. Če je pravilen, da izpiše Pravilno, lasten vir

Ko smo izvozili aplikacijo, smo jo najprej naložili na virtualen iPad in nato še na fizični iPad. Ker tej aplikaciji še nismo določili, kakšno naj ima ikono, se je na simulatorju in na fizičnem iPadu pokazala prazna ikona s predstavljeno mrežo, diagonalami in krogi (slika 25).

Slika 25: Aplikacija prazne ikone, lasten vir

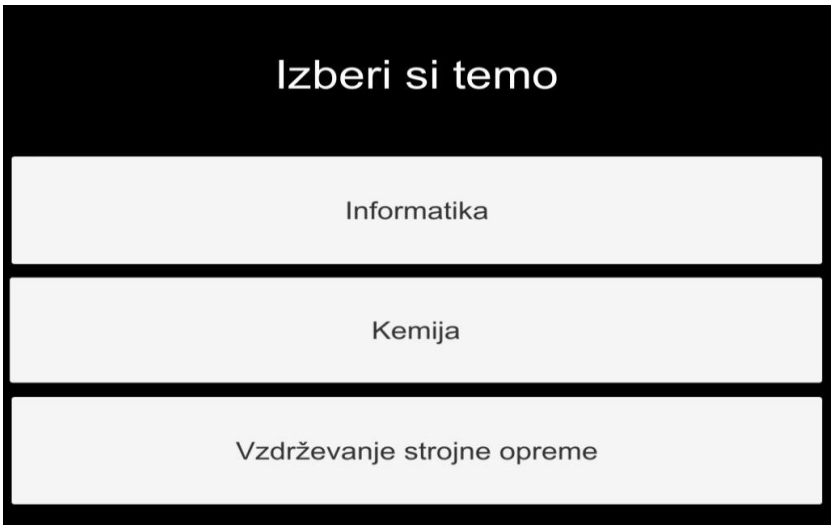
Ko smo naredili prototip aplikacije, smo se odločili, da bomo uporabili enak uporabniški vmesnik, kot smo ga zasnovali v Unity Game engine. Ko smo nehali usklajevati UI, dodajati vprašanja in funkcijo točkovanja, smo uporabili isto ikono na iOS-aplikaciji (slika 26).

Slika 26: Aplikacija z ikono, lasten vir

Po koncu usklajevanja UI v aplikaciji izgleda zelo podobna kot aplikacija Android.
Imamo napis »Preizkusi.se studio« in dva gumba za izhod in začetek igre (slika 27).

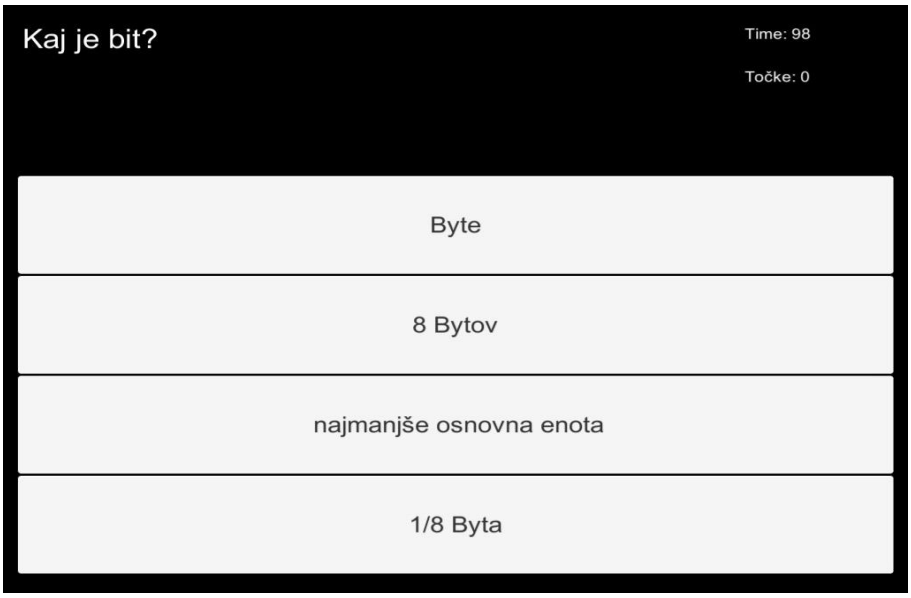
Slika 27: UI v posodobljeni aplikaciji, lasten vir

Ob pritisku na gumb Začni odpre okno menija za izbiro teme. V tem pogledu je napis »Izberi si temo« in 3 gumbi za izbor kategorije vprašanj (slika 28).



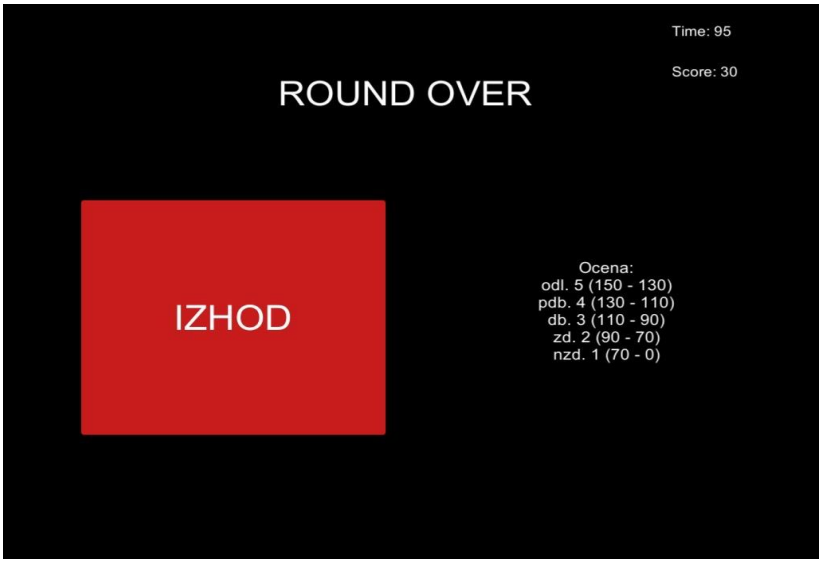
Slika 28: Uporabniški vmesnik pri izboru kategorij vprašanj, lasten vir

Uporabniški vmesnik ima med igranjem igre štiri gumbe z odgovori in tri napise. Prvi napis prikazuje vprašanje, druga odštevanje preostalega časa in tretji seštevek doseženih točk (slika 29).



Slika 29: Uporabniški vmesnik med igranjem igre, lasten vir

Ko končamo serijo 15 vprašanj, se nam izpiše rezultat, preostali čas, dosežena ocena in gumb za izhod (slika 30). Pri objavljanju aplikacije na Trgovino App Store smo imeli problem, in sicer nihče med nami nima Apple Developer ID, kar pomeni, da ni možno naložiti aplikacije za prosti prenos iz trgovine App Store. Poskusno je možno aplikacijo naložiti preko kabla in Mac računalnika, ki ima naložen XCode in Arhiv aplikacije.



Slika 30: Uporabniški vmesnik po končani igri, kjer se izpiše preostali čas in dosežene točke

3.5 Opis izdelave 3D-igre v Blenderju

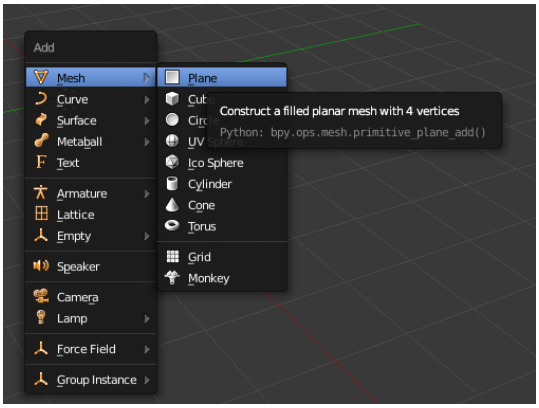
Sledi opis izdelave poizkusne 3D-igre in 2D-igre v Blenderju, ter osnovni ukazi.

3.5.1 Izdelava poizkusne 3D-igre v Blenderju

Igro v Blenderju lahko začnemo na več načinov, npr. najprej lahko ustvarimo meni, lahko tudi igro samo ali pa celo konec igre.

3.5.1.1 Ustvarjanje igre

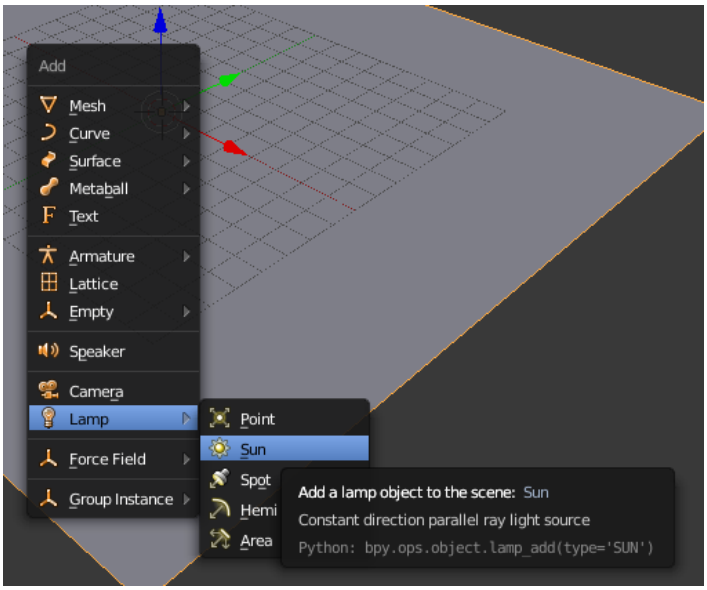
Za ustvarjenje igre je dobro, da najprej izbrišemo vse objekte in sami dodamo, kar potrebujemo. Ko začnemo z igro, najprej naredimo tla, tako da vstavimo (Shift + A| Mesh | Plane - slika 31).



Slika 31: Dodajanje tal, lasten vir

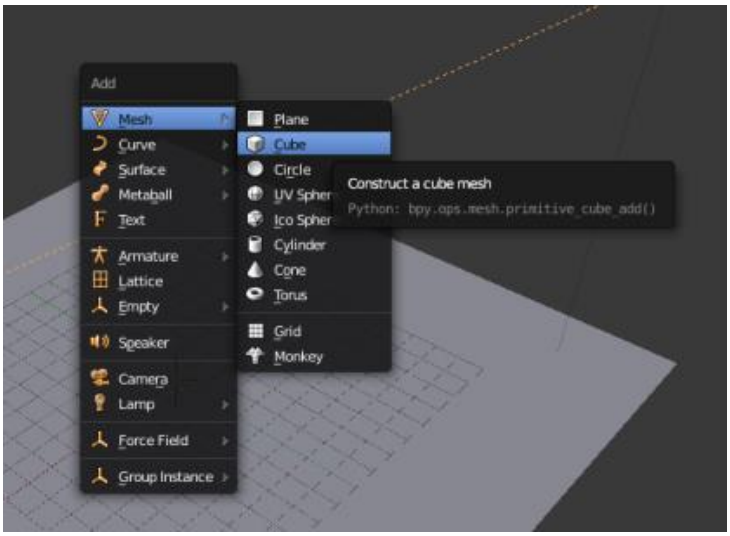
S tem dodamo ploskev v okolje. Z levim klikom lahko prestavljamo 3D-kazalec, ki določa, kje se bodo ustvarjali predmeti. S tipko G lahko predmet premikamo. Po kliku na črko G lahko izberemo tudi, po kateri osi ga želimo premakniti in lahko nato kliknemo na eno od črk Z, X ali Y. S tem se predmet avtomatsko poravnano premika po različnih oseh. S tipko R pa predmet zavrtimo po različnih oseh, lahko si pomagamo kot maloprej pojasnjeno s pritiskom na eno od črk Z, X ali Y.

Luč dodamo tako, da kliknemo (Shift + A | Lamp | Sun) in tako smo dodali kot vir svetlobe sonce (slika 32). Da bi izbrali, pod katerim kotom želimo, da padajo žarki, sonce zavrtimo s črko R. Če želimo videti v sceni sence, moramo v desnemu okvirčku Lastnosti klikniti na ikono Render (v obliki kamere) in izbrati način GLSL.



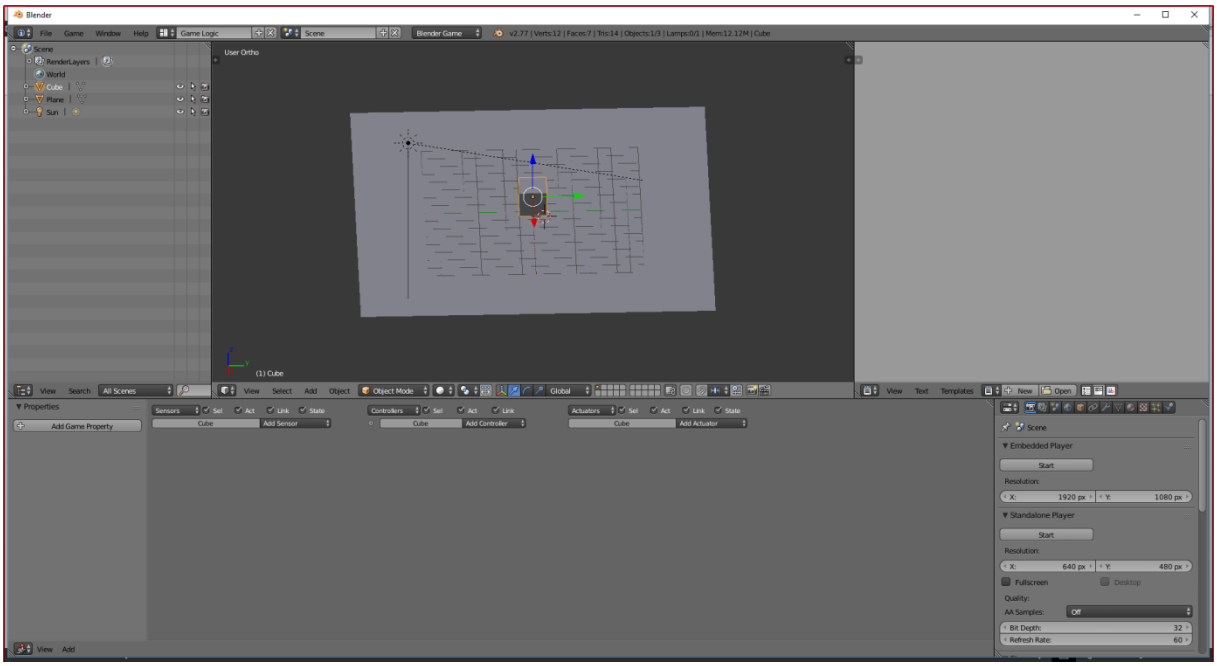
Slika 32: Dodajanje luči, lasten vir

Vse kar nam sedaj še manjka, je igralec, ki bo vseboval vse nastavitve za premikanje. Tega naredimo tako, da pritisnemo tipki (Shift + A| Mesh | Cube). V našem primeru smo kot osebek dodali kocko (slika 33). Kocko moramo nastaviti, da bo malo nad tlemi, saj bo drugače obtičala.



Slika 33: Dodajaje kocke, lasten vir

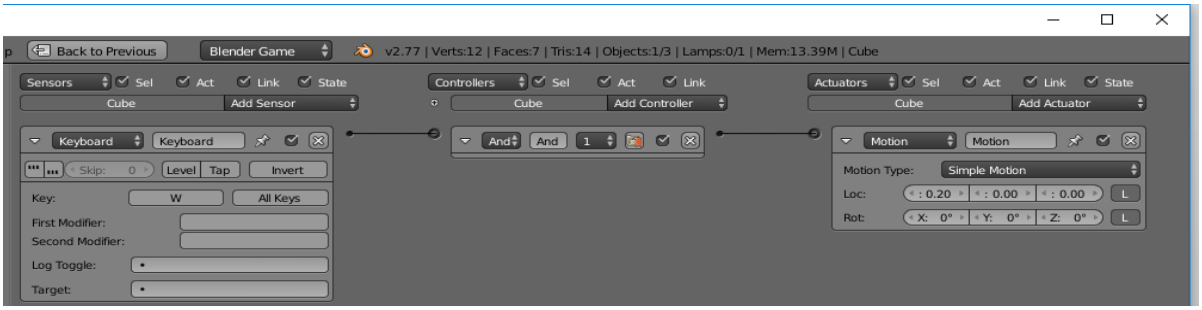
Sedaj pa preidimo k programiranju premikanja h kocki. Kocko lahko programiramo tako, da spremenimo okno Blender Render v Blender Game (v naslovni vrstici). Potem pa spremenimo način Default mode na Game logic (v naslovni vrstici). Ko kliknemo na Game logic, se v spodnjem delu okna prikaže prazen okvirček, v kateremu lahko dodajamo ukaze različnim stvarem. Kocki jih dodamo tako, da z desnim klikom kliknemo nanjo in začnemo dodajati ukaze logike igre (slika 34). Sedaj lahko začnemo dodajati ukaze kocki.



Slika 34: Dodajanje ukazov logike igre, lasten vir

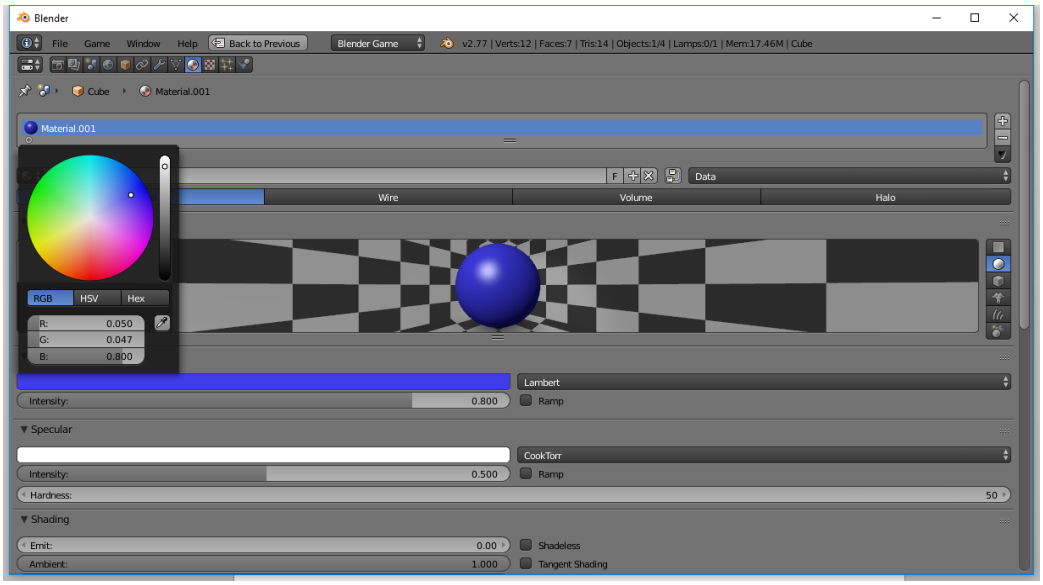
3.5.1.2 Dodajanje premikanja kocke

Če hočemo, da se kocka premika v 3D-prostoru naravnost, si moramo določiti, kaj je naravnost, npr. v našem primeru je to x-os. Potem pa dodamo Sensor | Keyboard in si izberemo, s katero tipko želimo predmet premikati po x-osi (naravnost). PriSensor | Keyboard kliknemo v polje Key in izberemo črko x. Potem pa dodamo še Actuator | Motion in določimo premikanje predmeta po x-osi. To je tako, kot bi povedali, če kliknemo črko (izbrana črka), se bo kocka začela premikati po x-osi z določeno hitrostjo. Da preverimo, da to res deluje, kliknemo črko P (play) in s tem lahko igramo našo igro. Ko pa kliknemo ESC-tipko, pa zapremo svojo igro in v prvem polju ob Loc določimo hitrost premikanja po x-osi. Podobno naredimo za ostale tipke premikanja npr. »w« za naprej, »y« za nazaj, »a« za zasuk v levo in tipka »s« za zasuk v desno. Da pa bo vse pravilno delovalo, moramo prvi in zadnji blok povezati s pomočjo kontrolnega bloka, ki ima nastavljeno funkcijo »In« oz. angl. »And« (slika 35).



Slika 35: Dodan en ukaz, lasten vir

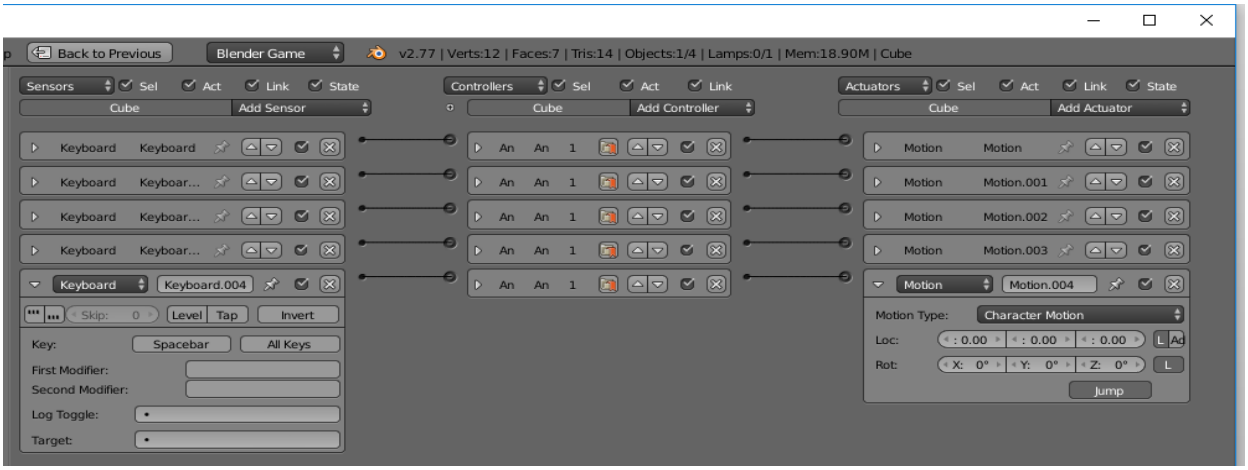
Ko so vsi ukazi za premikanje določeni, dodajmo še kamero, na kateri bo potekala igrica. Ker pa so tla in osebek enake barve, osebka ne bomo videli, zato njegovo barvo spremenimo. To naredimo tako, da v desnem okvirčku kliknemo na material in potem kliknemo na gumb New. S tem ustvarimo barvo in jo izberemo v barvnem krogu (slika 36).



Slika 36: Spreminjanje barve, lasten vir

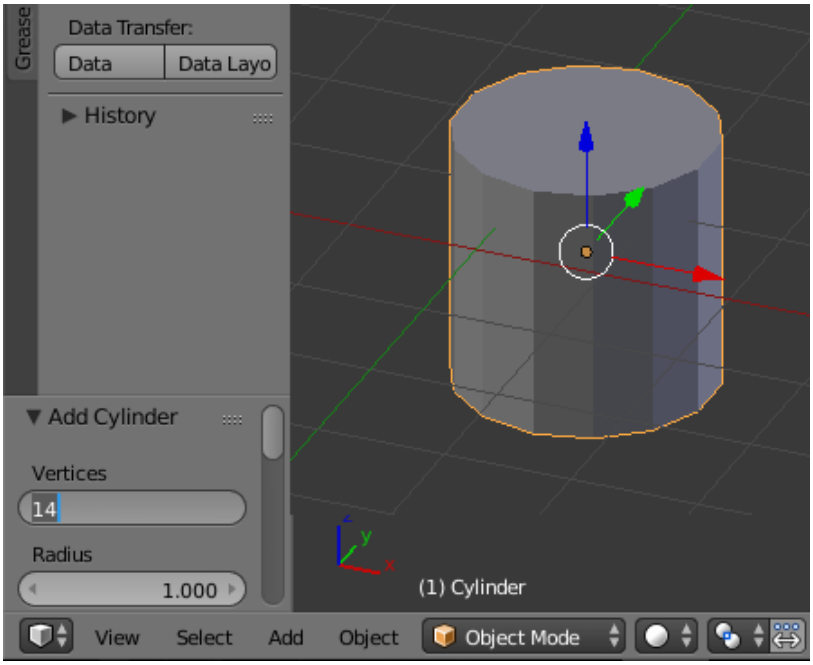
Sedaj kocki dodamo še zakone fizike (gravitacija) z desnim klikom na kocko in v desnem okvirčku Lastnosti na Physics ter namesto Static izberemo Character. S tem mu dodamo gravitacijsko silo in lahko izberemo hitrost padanja ter višino skoka.

Da bi osebku dodali ukaz skok, pogledamo spet v spodnji okvirček v Game logic. Tam spet dodamo senzor Keyboard in aktuator Motion. Pri senzorju Keyboard izberemo, s katero tipko želimo skočiti. Na aktuatorju motion izberemo Character motion in potem v istem okvirčku kliknemo na gumb Jump. Sedaj samo še oba povežemo s pomočjo kontrolnega bloka »In« oz. angl. »And« in dobimo ukaz skok (slika 37).



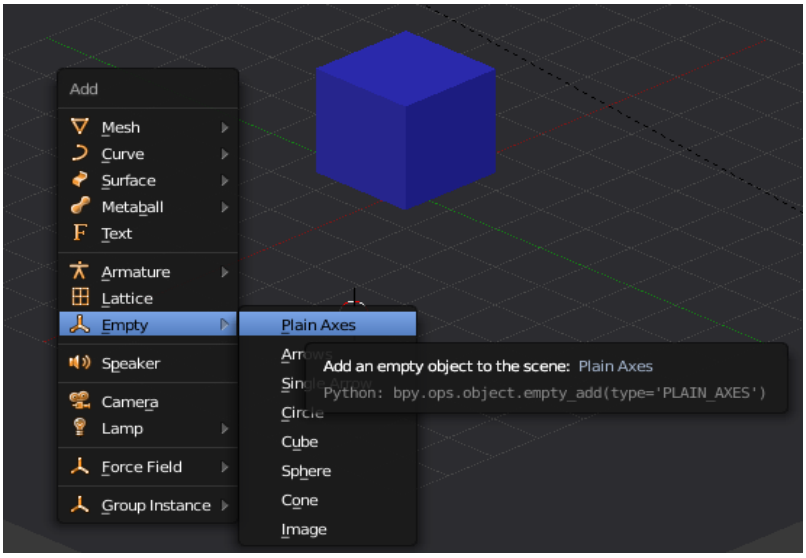
Slika 37: Ukazi premikanja, lasten vir

Potem naredimo objekt, ki ga bo naš osebek moral pobrati oziroma ga pridobiti. Za to moramo v scene 2 in klikniti na mali plus v naslovni vrstici, ki je levo zgoraj v bližini drugega okna. Plus mora biti znotraj okna, v katerem delamo. Sedaj izberemo želeni objekt in v levem oknu, kjer smo prej kliknili plus, se sedaj spodaj pojavijo možnosti, ki jih lahko spreminjamo (slika 40). Spremenimo število oglišč angl. Vertices, saj manj kot je oglišč, iz katerih je sestavljen objekt, hitreje bo kasneje igra delovala. Hitreje bo delovala zato, ker računalniku ne bo potrebno obdelati toliko podatkov glede na posamezna oglišča objekta (slika 38).



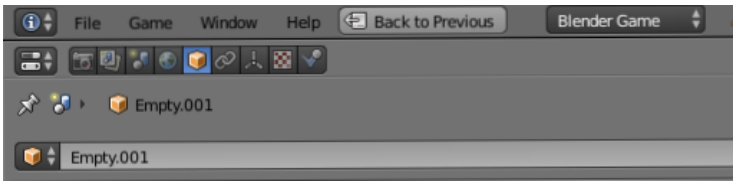
Slika 38: Spreminjanje oglišč angl. Vertices objekta, lasten vir

Sedaj pa dodajanje dejanskega objekta k igri. Ker smo objekt dodali v drugačen scene, le-ta ne bo viden v igri, zato ga moramo dodati v dejansko igro. Zato moramo v scene 1 dodati prazen objekt, najbolje da je to prazen koordinatni sistem (empty Plain Axes - slika 39).



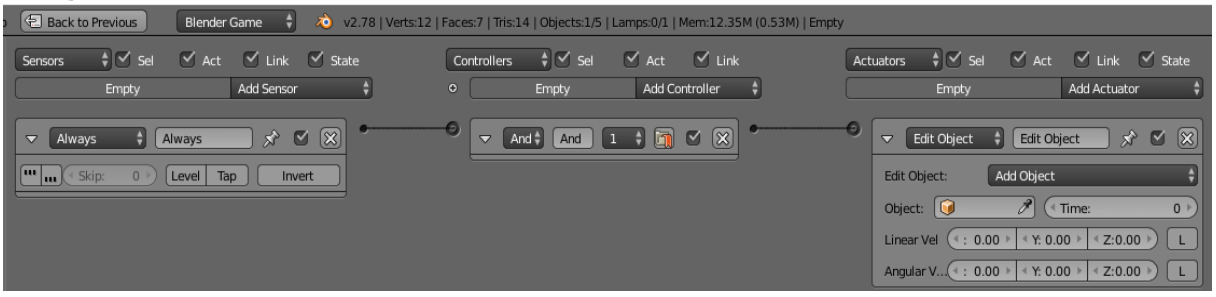
Slika 39: Dodajanje praznega koordinatnega sistema, lasten vir

Da ga dodamo v igro, ga moramo najprej preimenovati (slika 40).



Slika 40: Preimenovanje objekta Empty, lasten vir

Potem pa kliknemo na prazen koordinatni sistem in mu dodamo logiko. Dodamo mu tipalo Sensor | Always in Actuator | Edit Object in v aktuatorju Edit Object izberemo ime predmeta, ki ga želimo dodati objektu z imenom prazni, ki je prej bil Empty.001 (ime, ki smo ga si izbrali prej - slika 41).



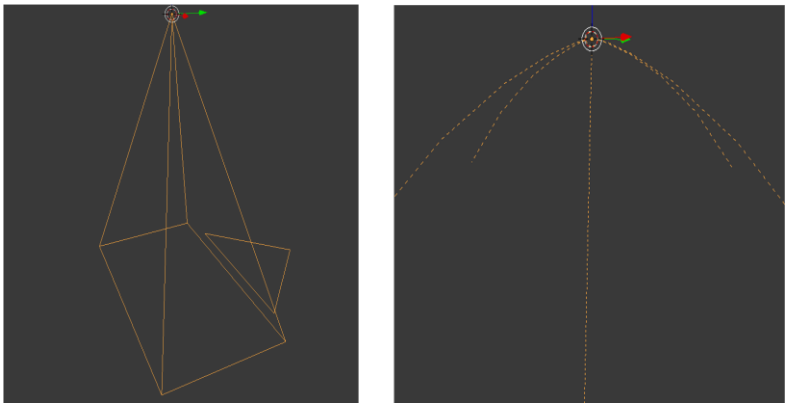
Slika 41: Dodajanje objekta k igri, lasten vir

Sedaj izberemo objekt v scene2 in mu dodamo logiko Sensor | Collision in Actuator | Edit Object in na aktuatorju Edit Object izberemo tip End Object (slika 42). To omogoča, da se objekt združi z osebkom.

Slika 42: Dodajanje učinka združevanja, lasten vir

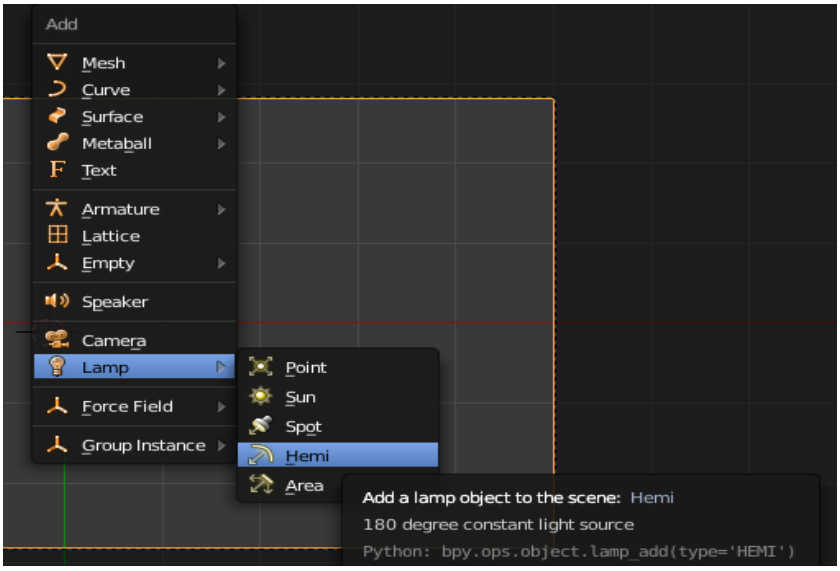
3.5.1.3 Ustvarjanje menija in konca igre

Za ustvarjanje menija je najbolje, da izbrišemo vse, kar je sedaj v novo odprtem Blender programu. Ko smo to storili, dodamo kamero in ji nastavimo pogled (slika 43).



Slika 43: Kamera in luč, lasten vir

Po tem dodamo luč, saj se drugače predmeti ne bodo videli (slika 44).



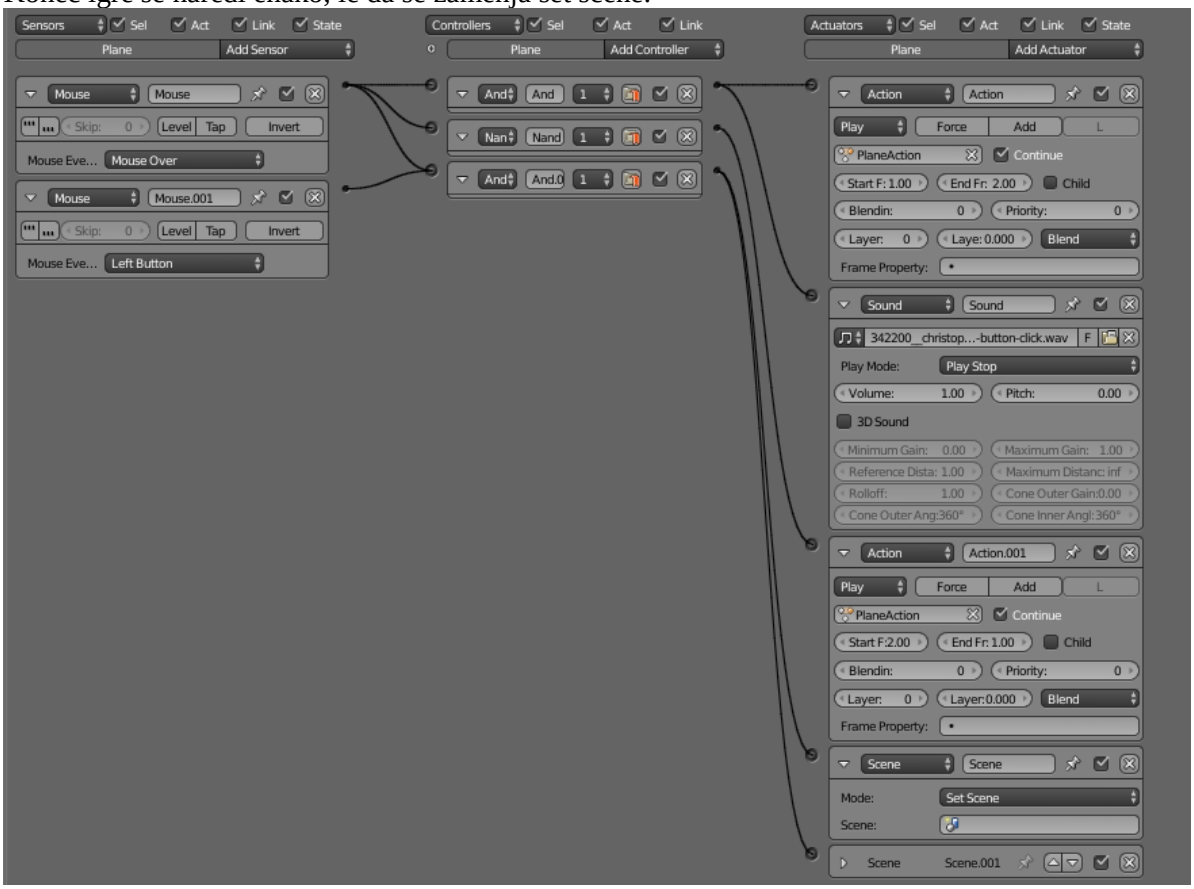
Slika 44: Dodajanje luči, lasten vir

Sedaj pa dodamo še plosko podlago (Mesh | Plane) in nanjo besedilo (angl. Text), na katerega napišemo funkcijo igranja npr. besedilo Igraj (slika 45). Objekte smo postavili enega na drugega tako, da se besedilo vidi, pod njim pa je ploskev.

Slika 45: Besedilo in ploskev, lasten vir

Sedaj kliknemo na ploskev in ji dodamo logiko (slika 46). Za animacijo si izberemo prvi okvir (ang. fps 1) in kliknemo črko I in izberemo LocScale, da zaklenemo velikost na okvirju 1. Potem pa še prestavimo na okvir 2 in jo malo povečamo ter spet kliknemo I ter LocScale. Sedaj smo pridobili enostavno animacijo, ki jo bomo uporabili v trenutku, ko bo miška prišla na predmet.

Pri senzorju ukaz Mouse over pomeni, da se bo senzor sprožil, ko bo miška prekrila objekt. Ukaz Mouse left button pomeni, da se bo senzor sprožil, ko bo izveden levi klik na miški. Ukaz Action pomeni animacija predmeta. Ukaz Sound pomeni zvok. Ukaz Scene pa nam pove, kam bomo prešli ob kliku na predmet. Ukaz Nand je obratna funkcija odAnd. Ukaz Play stop pomeni, da zvok traja tako dolgo, kot traja dejavnost.
Konec igre se naredi enako, le da se zamenja set scene.



Slika 46: Logika gumba v meniju (ploskve),lasten vir

3.5.2 Izdelava didaktične 2D-igre v Blenderju

Blender je odprtokodni program za ustvarjanje animacij in iger ter modeliranje. Postopek izdelave 2D-igre se ne razlikuje kaj dosti od postopka izdelave 3D-iger, vendar pa je potrebno spremeniti nekaj stvari. Npr. če smo začeli z izdelavo 2D-igre, moramo vse predmete pretvoriti v objekte Mesh. To pa je tudi slabost, ker se besedilo ne more več spreminjati. Slabost pa je tudi čas, ki ga potrebujemo za izdelavo 2D-igre, saj je Blender namenjen bolj animacijam in izdelavi 3D-iger kot izdelavi 2D-iger. Posledično potrebujemo tudi skoraj dvakrat več časa kot pa v kakšnem drugem orodju, kot je na primer Unity. Še ena slabost pa je, da so stvari na programu nekoliko pomešane in je do njih težje dostopati preko kod ter da za uporabo kod zahteva veliko znanja in izkušenj. Ima pa tudi prednosti, na primer, da lahko s povezovanjem hitreje naredimo logiko gumba in predmeta ali pa, da je datoteke zelo lahko izvoziti in dodati kode iz programa Python.
Prednosti:

- Za ta program ni potrebna posebno dobra strojna ali programska oprema.
- Hitrejše pisanje logike kot s grafičnim povezovanjem logičnih gradnikov v Blenderju.
- Odprtokodni program.
- Hitra izdelava animacij in 3D-iger.
- Niso nam znane omejitve z nobenimi orodji (v programu se lahko naredi skoraj vse).
- Hitri izvoz igre za Windows OS.

Slabosti:

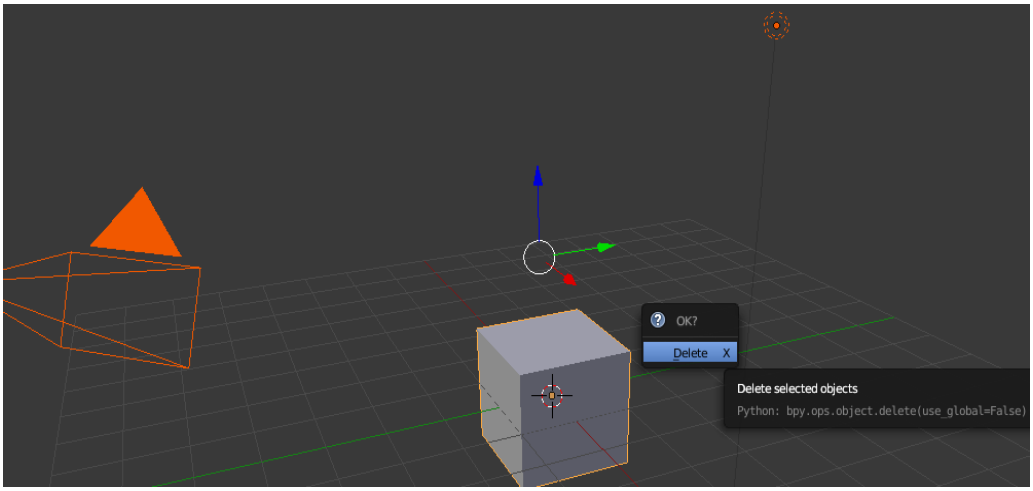
- Veliko zahtevanega znanja na področju kodiranja.
- Izvoz samo na en operacijski sistem.
- Rahlo pomešane stvari.
- »Računalniški hrošči«, ki povzročajo neželene stvari (izginevanje predmetov…).

3.5.3 Izdelava 2D-igre v Blenderju

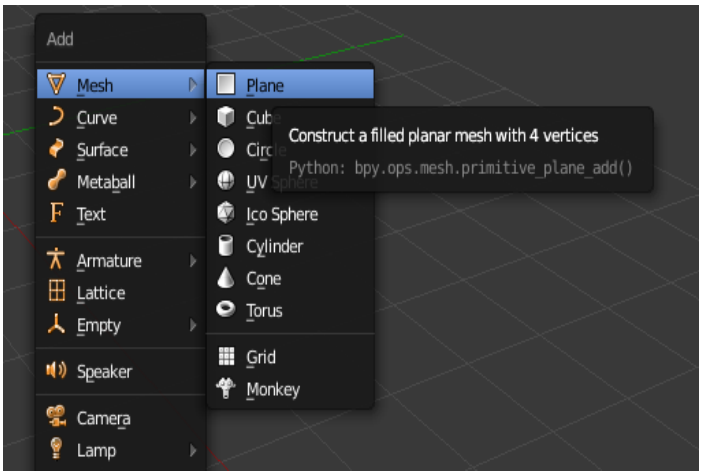
Sledi kratek opis izdelave 2D-igre v Blenderju.

3.5.3.1 Ustvarjanje menija

Za izdelavo menija moramo iz novo odprtega Blenderja izbrisati vse predmete (A | Delete). Sedaj pa začnemo vstavljati predmete, kot je na primer besedilo ali pa ravnine, katerim bomo kasneje dodali logiko (Shift + A). Najbolje pa je, da so vsi naši dodani predmeti izbrani iz vrste Mesh-a, razen seveda izjem, kot je kamera ali pa besedilo (slika 47 in slika 48).

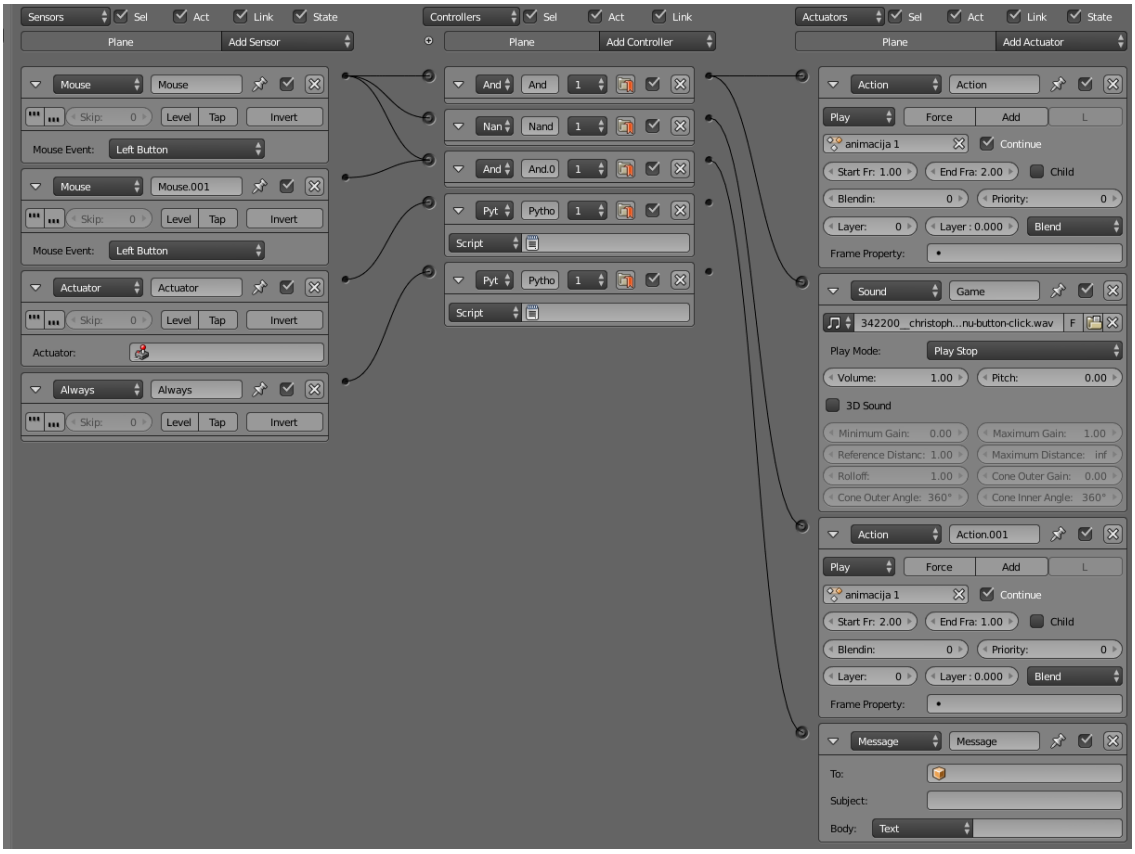


Slika 47: Brisanje predmetov, lasten vir



Slika 48: Ustvarjanje ravnine Plane, lasten vir

Ko smo ustvarili izgled menija, mu začnemo vstavljati logiko. To storimo tako, da zgoraj izberemo način Game engine in pa Game Logic zgoraj proti levemu kotu (v naslovni vrstici). Lotimo se z dodajanjem logike (slika 49). Senzorje in aktuatorje ter ukaze povežemo s kotrolerji, kot smo naredili že prej.



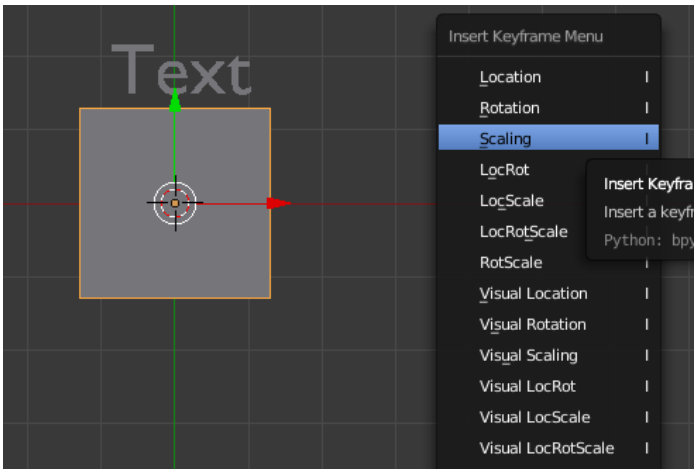
Slika 49: Logika gumba, lasten vir

Pojasnilo:

- Prvi stolpec so senzorji, ki se nam sprožijo in začnejo pošiljati ukaze po določeni stvari, za katero so namenjeni.
- Drugi stolpec so kontrolorji, ki nam povedo logiko, kako se bo program izvedel.

- Tretji stolpec pa so dejanski izvajalci oz. aktuatorji.
- S tem ko povežemo LeftMouse in MouseOver hkrati na eden kontroler tipa»And«, ki ukaz posreduje naprej samo, če kliknemo točno na izbrani objekt.
- Ker pa imamo v Python kodi zapisane končne ukaze, jih ni treba nikamor povezati.
- Zadnji senzor Always nam omogoči, da je koda v Pythonu aktivirana skozi celotno igro.
- Akcijski aktuator (Action Actuator) predmetu ustvari animacijo ob prekrivanju le-tega z kazalcem miške.
- Kontoler »Nand« nam izvede obratno funkcijo.

Sedaj lahko še vstavimo kamero ter sonce (Shift+ A | Light | Sun in Shift+ A| Camera). Za pogled skozi kamero kliknemo na tipko nič na numerični tipkovnici. Izberemo lahko več ravnine hkrati, če kliknemo in od prvega klika naprej držimo tipko Shift. Zdaj se prepričamo, če smo na prvi sličici, nato kliknemo tipko I | Scailing, da ustvarimo ključni okvir na prvem okvirju pri ustvarjanju animacije (angl. keyframe). Nato z desno puščico premaknemo sliko na okvir dva, kjer s tipko S malo povečamo velikosti ravnine. Ko smo dovolj povečali ravnino, zopet kliknemo tipko I | Scailing, da ustvarimo animacijo na drugem okvirju (angl. keyframe) (slika 50 in slika 51).



Slika 50: Dodajanje animacije ključnih okvirjev, lasten vir

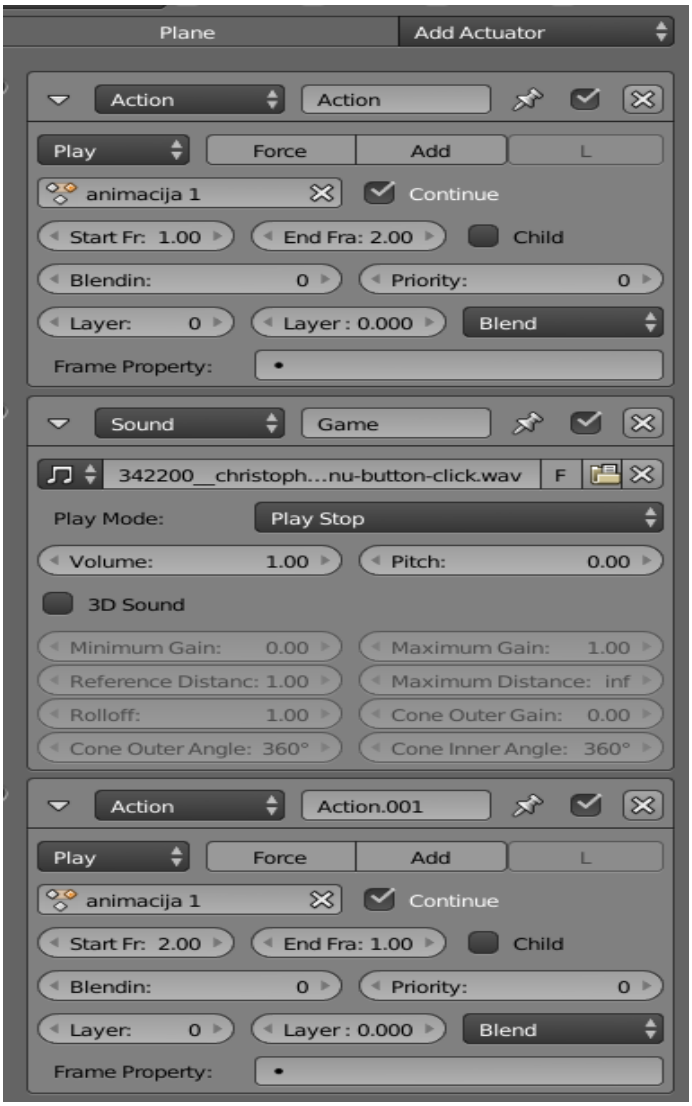
Slika 51: Animacije ključnega okvirja 2, lasten vir

Sedaj preimenujmo nastalo animacijo v animacija 1. Ko smo animacijo preimenovali, ji dodamo še logiko. V logiko dodamo tudi zvok s pomočjo aktuatorja Actuator sound (slika 52).

Razlaga:

- S prvim aktuatorjem Action poskrbimo za del animacije, ko z miško prekrijemo ravnino (se poveča).
- Z drugim aktuatorjemSound poskrbimo za zvok ob kliku miške in prekrivanju le-te z ravnino.
- S tretjim aktuatorjem Action pa poskrbimo za del animacije, ko miško umaknemo iz ravnine (se pomanjša).

S tem smo nekako zaključili z izdelavo menija.



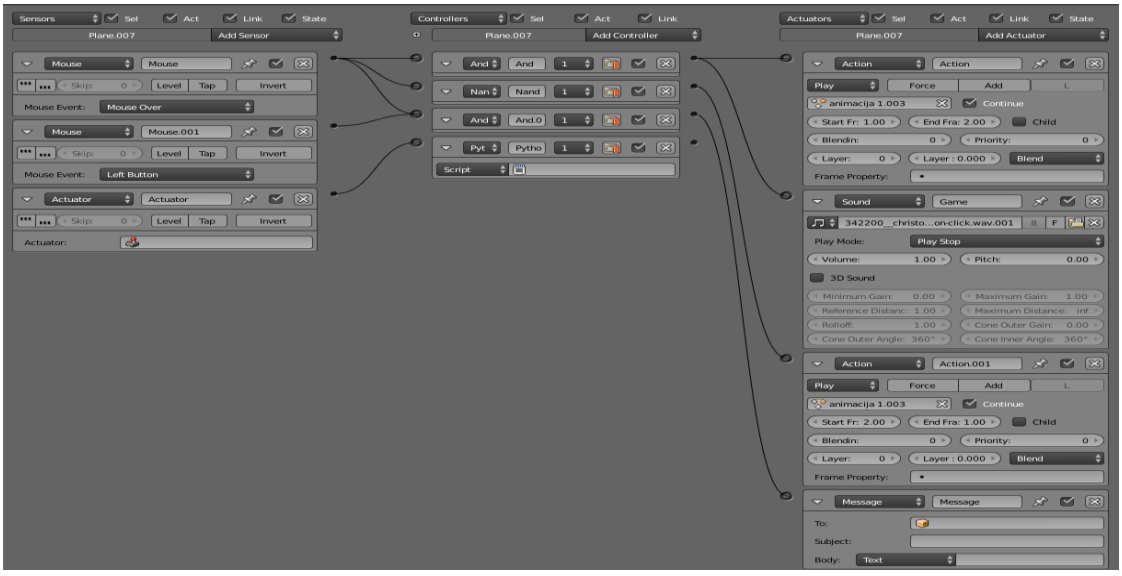
Slika 52: Angl. Actuators, lasten vir

3.5.4 Izdelava vsebine igre

Za izdelavo vsebine igre bomo morali ustvariti novo sceno, ki jo bomo izvozili po vseh vprašanjih in jo nato naknadno spreminjali. Dobro je, če prvo sceno poimenujemo meni in drugo vsebina 1. Sedaj naredimo podobno sceno, kot smo je prej, samo da bomo na to dodali več ravnin (toliko kot je število odgovorov v kvizu) in pa dodali besedilo, ki bo naše vprašanje (ne smemo pozabiti, da moramo tudi to sceno spremeniti pogon za igro oz. Game engine) (slika 53).

Slika 53: Izgled vprašanja, lasten vir

Vsak od gumbov na zgornji sliki mora imeti enako logiko kot gumb na meniju. Ker potrebujejo enako logiko, jim je potrebno ustvariti tudi animacije. Logika mora biti identična, le da ni ukazov v Python skripti, temveč je ta vezan na senzor Aways (slika 54).

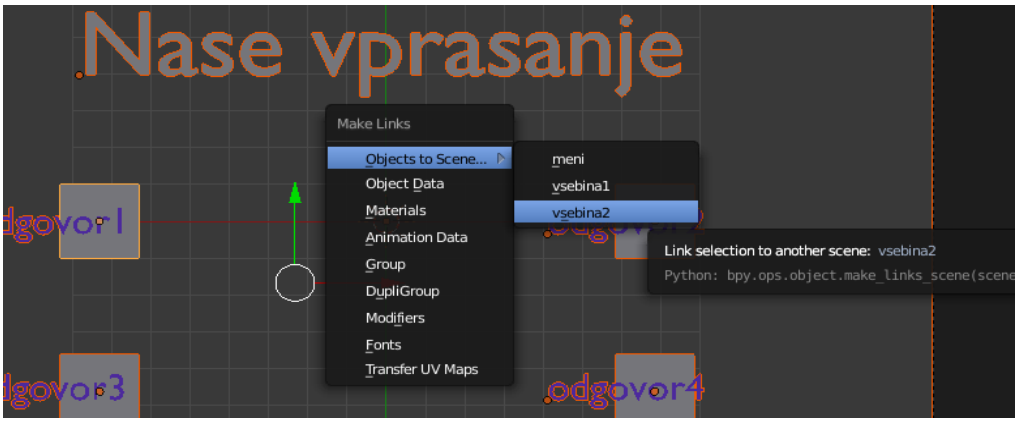


Slika 54: Logika odgovora, lasten vir

Sedaj moramo dodati še izbrano število odgovorov z besedili nad ravnino ozadje. Npr. naše vprašanje je Kako hitra je svetloba? Vprašanju dodamo štiri odgovore, na primer 1. odgovor: 20 km/h, 2. odgovor: 30 km/h, 3. odgovor: 2000000 km/h in 4. odgovor: 300000 km/s. Odgovorom moramo spremeniti še barvo, da bodo drugačne barve kot ravnina ozadja (slika 55).

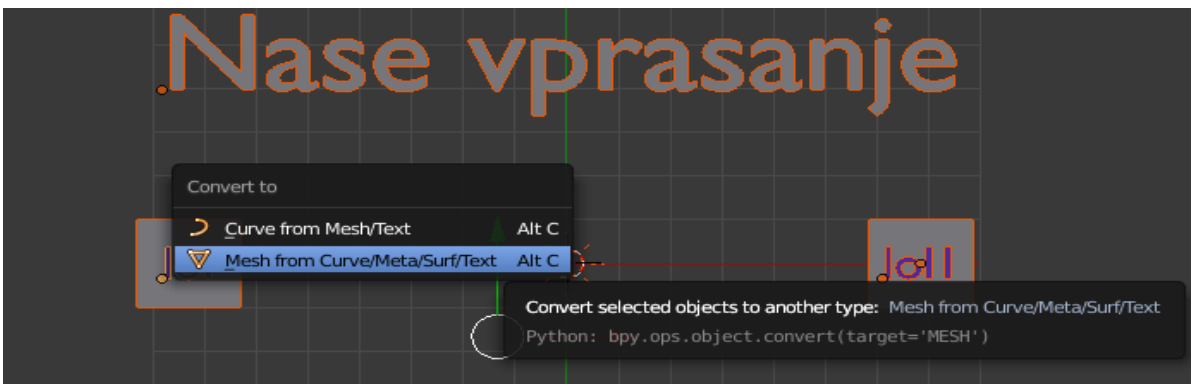
Slika 55: Dodana besedila, lasten vir

Zdaj moramo narediti samo še poljubno število scen (število vprašanj), kar je v tem primeru 45. Ko bomo imeli vse scene izdelane in poimenovane, lahko začnemo z delom. V sceni vsebina1 izberemo vse (tipka »A«) in jo povežemo na druge scene (Ctrl | L). Sedaj, ko smo povezali z vsemi scenami, začnemo spreminjati vprašanja in odgovore na drugi. To storimo tako, da najprej na sceni, ki jo bomo urejali, izberemo (tipka »A«) in potem kliknemo tipko U (Object and data). To nam poskrbi za to, da se z urejanjem nove scene ne bo spreminjala prva, ki smo jo povezali z drugimi (slika 56).



Slika 56: Povezovanje scen, lasten vir

Sedaj lahko na sceni, ki smo jo povezali, spremenimo vsebino besedila (tipka tabulator) in zaključimo tako, da vsak predmet na sceni spremenimo v Mesh (Alt +C | Mesh from Curve) (slika 57).



Slika 57: Pretvorba v Mesh, lasten vir

Zdaj posebej izberemo vsak odgovor in s pritisnjenim Shift kliknemo na ravnino za njim ter jih združimo v en objekt (Ctrl+ J).

Po takšnem postopku naredimo vsa vprašanja od 1 do 45. Ko smo končali z vsemi scenami, enako postorimo še na prvi sceni, ki se je povezala z drugimi scenami. Dobro je, če si igro sproti shranjujemo in naredimo več datotek, če se slučajno kje zmotimo, da lahko še vedno spremenimo besedila ali pa scene.

S tem smo končali našo vsebino, sedaj pa na Python skripte, ki jih moramo še napisati.

3.5.5 Python kode za delovanje igre v Blenderju

Kode v Pythonu morajo zagnati scene in potem naključno izbrati vprašanja ter jih prikazati. Kode pa morajo šteti tudi točke in določati dolžino igre (vse kode so v prilogi).

Najprej ustvarimo Python kodo, poimenovano »logika gumba.py«. (Dobro je, če si odpremo prostor desno od igre in v njemu odpremo urejevalnik besedil angl. text editor). Ko smo to postorili, kliknemo na »Templets« in potem Python ter game Logic simple. Sedaj se nam že odpre nekaj privzetega, kar moramo izbrisati. Notri pa napišemo kodo, ki je priložena v datoteki z imenom logika gumba).

LOGIKA GUMBA.docx

V datoteko Logiko gumba.py dodamo Python ukaze v igri (le napačne odgovore in meni) ter koda ukaze za senzorje in aktuatorje).

Potem ustvarimo novo kodo, poimenovano »pravilen gumb.py«.

PRAVILEN GUMB.docx

V to kodo dodamo le vse pravilne odgovore.

Zadnja nova koda, ki pa jo ustvarimo, je poimenovana »gamelogic_simple.py«.

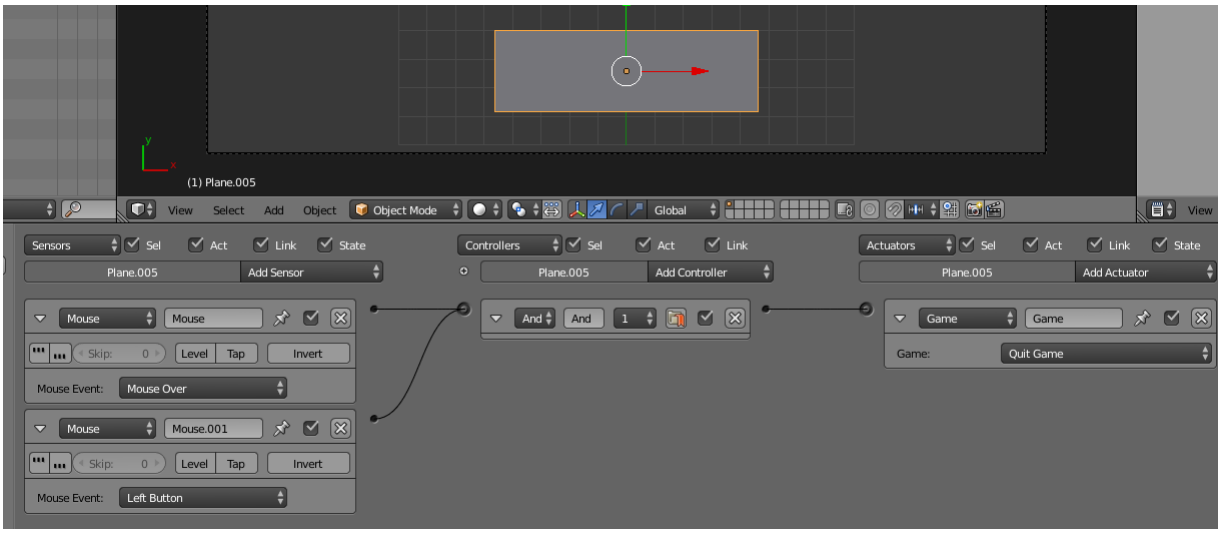
gamelogic_simple.docx

To kodo pa dodamo na Python ukaz v igri za menij in od senzorja Always.

3.5.6 Štetje točk in konec igre

Za konec ustvarimo sceno z eno ravnino, ki bo zaključila igro. To sceno poimenujemo »konec«.

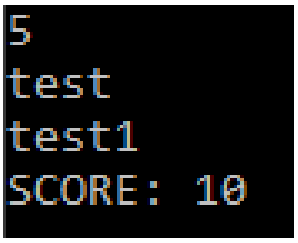
Preko napisanih kod se bo ta scena pojavila, ko bomo rešili 10 vprašanj (slika 58).



Slika 58: Logika gumba konca, lasten vir

Točke v igri se v našem primeru ne štejejo, vendar izpisujejo v glavnem oknu za ukaze Python. Če bi želeli imeti še eno prekrivno okno, kjer bi se v enem izmed kotov izpisovale točke, bi morali napisati še eno kodo, ki bi sprejemala sporočila od prejšnjih kod.

Torej, če bi poleg Blenderja odprli še glavno ukazno okno za program Python, bi videli, kako se nam izpisujejo točke (slika 59).



Slika 59: Štetje točk v glavnem ukaznem oknu Pythona, lasten vir

4 RAZPRAVA

Za vsako okolje posebej smo navedli njegove ključne prednostni in slabosti ter izpostavili, čemu je namenjeno.

4.1 Unity

Je orodje, ki je namenjeno pretežno izdelavi 2D- in 3D-iger. Za te ne zahteva veliko znanja. Namenjen je vsem vrstam programerjev od slabših do nekoliko boljših za izdelavo raznih iger in aplikacij.
Njegova prednost lahko postane tudi slabost, saj bomo za iskanje napak v njem pozneje porabili veliko časa. Za programiranje v tem programu potrebujemo vsaj srednje dobro strojno opremo. Pozitivno je tudi, da lahko igre, narejene v Unity, izvozimo za skoraj vse OS in strojno opremo. Je program, ki skoraj nima slabosti.
Zahteva poznavanje programskega jezika C++.
Za izdelavo dobre 3D-igre potrebujemo zelo veliko predznanja.

Presenetilo pa nas je tudi, koliko časa potrebuješ za izdelavo programa v tem jeziku (pozitivno presenečeni, saj je povprečni čas izdelave ene igre 7 dni).

4.2 Blender

Je odprtokodno programsko orodje za grafično 3D-modeliranje, animiranje, komponiranje s post produkcijo, 3D-manipulacijo v realnem času. V Blenderju je vgrajen programski jezik Python kot API, s katerim lahko uporabnik avtomatizira in dodatno razširi možnost edinstvenega programa. Izdelava 2D-igre pa je lahko kar izziv. Namenjen je vsem vrstam programerjev. Zanj ne potrebujemo dobre strojne opreme razen dobre grafične, saj je program dobro optimiziran in prejšnje verzije programa delujejo tudi na slabih računalnikih.
Ni namenjen samo izdelavi iger, zato z njim ne moremo narediti grafično najbolj zahtevne igre.
Velika slabost, s katero smo se srečali, so tako imenovani računalniški »hrošči«.
Velika slabost pa je tudi to, da so stvari zelo pomešane in jih je zelo težko klicati preko kod oziroma moramo biti zelo dobri na področju pisanja kod v Pythonu.
Presenetilo pa nas je tudi, koliko časa smo potrebovali za izdelavo programa v tem jeziku (negativno presenečeni, saj je čas izdelave ene igre trajal skoraj 2 meseca).
Izdelano igri izvozimo samo za tisti operacijski sistem, kot je nameščen Blender, za izvoz v drug operacijski sistem bi morali igro odpreti v drugem sistemu in jo tam izvoziti.
Čeprav je odprtokodni program, lahko v njem naredimo praktično vse.

4.3 Python

Je programski jezik, ki je namenjen pisanju programov in celo lažjih ter zahtevnih 2D-iger. Izdelava 2D-igre brez uporabniškega vmesnika je lahka, vendar pa je igra v besedilni obliki oziroma skoraj cela sestavljena iz besedila. Njegova največja slabost je naše omejeno znanje, saj omogoča izdelavo zahtevnih programov. Mi smo se lahko nekaj potrebnega znanja pridobili tudi na spletu. Ustvarjanje programov v tem programskem jeziku je zelo hitro in lahko ter je odvisno od stopnje uporabnikovega znanja.

Je program, ki praktično nima slabosti, saj je programski jezik osnova številnim SDK-okoljem.

Neposredna izdelava grafično zahtevne 3D-igre praktično ni mogoča.

Ima zelo veliko funkcij in v njem se da narediti skoraj vse.

Je odprtokoden programski jezik, okolje IDLE je brezplačno.

Brez dodatkov se ne da preprosto pretvarjati Python datoteke, ki tečejo v IDLEju v samostojne Windows programe, sicer ko se temu privadimo, delo kar steče.

1.1 XCode

Je Applovo programsko okolje in orodje, v katerem se lahko ustvarja programska oprema za iPhone, iPad, Mac, Apple Watch in Apple TV. Namenjen je dobrim programerjem za izdelavo raznih iger in

programov. Okolje vsebuje že nameščene skripne jezike: AppleScript, Perl, Python in Ruby. Njegova velika slabost pa je, da za ta programski jezik potrebujemo zelo drago strojno opremo, Applove računalnike Mac. Programski jezik Swift pa teče tudi v operacijskem sistemu Linux in v okolju XCode 8.2 (https://swift.org/). Slabost je tudi, da moramo za dodajanje naše aplikacije v kakršno koli trgovino plačati veliko denarja. Namenjen je samo razvijalcem za uporabnike sistemov Apple.

Ko igro ali aplikacijo enkrat naredimo, je ne moremo naložiti niti zase, saj ne podpira niti prenosov preko kabla.

Za izvoz programov je potrebno najeti licenco razvojnika Developer Enterprise Program za 299 USD/leto (za testiranje vsake od vrste naprav Apple Watch,iPad ali iPod touchpo 100 USD/leto). Ker teh možnosti nismo imeli, smo imeli nekaj težav z računalniškimi »hrošči«.

Presenetilo pa nas je tudi, koliko časa potrebuješ za izdelavo programa v tem jeziku (negativno presenečeni, saj je čas izdelave ene igre trajal skoraj 2 meseca).

4.4 Primerjava razvojnih okolij

Med seboj smo primerjali vsa štiri orodja in jih razvrstili glede na njihove slabosti ter prednosti.

4.4.1 Prvo mesto Unity 5 C++,

Na prvo mesto smo postavili SDK-okolje Unity 5 C++, ker:

- Skoraj nima pomanjkljivosti.
- Je dobro optimiziran.
- Zaradi kratkega časa za izdelavo naše igre.
- Je za osebno rabo zastonj, sicer je treba plačati.

4.4.2 Drugo mesto

Na drugo mesto smo postavili SDK-okolje Blender, ker:

- V njem lahko naredimo praktično vse.
- Ne potrebujemo vrhunskega znanja za izdelavo igre ali aplikacije.
- Odlična optimizacija programa.
- Je odprtokodni program.

4.4.3 Tretje mesto

Na tretje mesto smo postavili SDK-okolje Python, ker:

- V njem lahko naredimo veliko stvari.
- Najlažje je najti napake ali računalniške »hrošče«.
- V njem ne moremo ustvariti grafično zahtevne igre brez posebnih znanj.
- Namenjen je ustvarjanju programov.

4.4.4 Četrto mesto

Na četrto mesto smo postavili SDK-okolje XCode-Swift, ker:

- Potrebno je zelo veliko znanja o programiranju.
- Za delovanje programa potrebujemo zelo drag računalnik Apple (izjema je Linux).
- Narejene aplikacije ne moremo prenesti, da deluje skoraj niti zase.
- Aplikacije lahko izvozimo samo na drago strojno opremo (izjema je Linux).

5 Zaključek

Strelske video igre so najbolj razširjene med vsemi igrami. Zato moramo iskati rešitve, da bodo mladi bolj množično začeli igrati didaktične igre. Ena izmed rešitev je ustvarjanje novih in zabavnih didaktičnih iger. Ampak kako prepričati uporabnike, da bodo igrali didaktične igre? Odgovor je preprost. Ustvariti moramo zelo zabavno didaktično igro, ki bo vključevala različne dogodke, izhajajoč iz trenutnih situacij.

Tudi izdelava iger ni tako zahtevna, saj lahko večino podatkov pridobimo iz knjig, videov vodičev na YouTube in s spletnih strani. Igro, ki jo naredimo, lahko tudi objavimo na trgu, vendar to po navadi nekaj stane.

Igri pa lahko dodamo tudi različne teme in nivoje. Tako jo naredimo primerno za določeno populacijo. Lažja kot so vprašanja v didaktični igri, mlajši so lahko uporabniki. Če pa bodo vprašanja težja, se bo meja dvignila na srednjo populacijo. Tema igre je pomembna tudi zaradi zanimivosti. Bolj kot bo tema zanimiva, več uporabnikov ali igralcev si bo želelo igrati igro.

Ko pa ustvarjamo igro, moramo vedeti, s čim jo bomo ustvarili. Ali bo programska oprema zelo draga? Ali bo to programska oprema, ki je zastonj oziroma odprtokodna? Ali celo ustvarimo svoj programski jezik? Ugotovili smo, da so skoraj vsi programski jeziki in SDK-okolja skoraj enako zmogljivi. Na nekaterih lahko naredimo stvari, ki jih na drugih ne moremo in obratno. Delno je to odvisno tudi od predznanja in izkušenj programerjev oziroma nas. Zato je težko vrednotiti programske jezike ter SDK-okolja. Preprostejša kot je oblika uporabniškega vmesnika in zasnove igre, hitreje bomo lahko izdelali igro ter obratno. Bistvena razlika med programskimi jeziki ter SDK-okolji je njihov uporabniški vmesnik in posledično hitrost in kvaliteta dela.

Programi in SDK-okolja so namensko narejena za posamezni programski jezik. Vendar kateri programski jezik je najboljši? Najboljšega programskega jezika ni, saj vsi programski jeziki omogočajo skoraj enake stvari, vendar z njemu lastno kodo in sintakso.

Tako lahko prvo hipotezo potrdimo, saj ne potrebujemo začetnega znanja, ker se lahko veliko potrebnih znanj naučimo sami iz knjig, videov vodičev na YouTube, s spletnih strani ali od izkušenih programerjev.

Tudi drugo hipotezo lahko delno potrdimo, saj za spletno prodajo aplikacije ni nobenih posebnih pogojev. Delno pa to hipotezo zavrnemo, saj bi bilo za spletno prodajo potrebno plačati kar nekaj denarja.

Tretjo hipotezo lahko tudi potrdimo, saj je primerna za srednješolsko populacijo zaradi tematskih področij, ki smo jih izbrali.

Četrto hipotezo lahko potrdimo, saj so vsa SDK-okolja skoraj enako zmogljiva in se razlikujejo samo po uporabniškem vmesniku in sintaksi.

Zadnjo hipotezo pa zavračamo, saj C++ ni najboljši programski jezik, ker so programski jeziki, ki smo jih preizkusili, skoraj enako zmogljivi. To je odvisno tudi od predznanja programerja.

6 ZAHVALA

Raziskovalna naloga ne bi bila v takšni obliki, če nam pri nastajanju le-te ne bi pomagalo veliko ljudi. Zahvala je torej namenjena naslednjim:

- mentorjema mag. Karmen Grabant in Nedeljku Grabantu, dipl. inž., za pomoč, voljo, vztrajnost, njun prosti čas in spodbudo pri nastajanju raziskovalne naloge;
- našim staršem;
- Lidiji Šuster, prof., za lektoriranje povzetka;
- Mariji Klemenšek, prof., za lektoriranje naloge;
- Simoni Diklič, prof., za lektoriranje angleškega povzetka;
- učiteljem ERŠ in ravnatelju Simonu Konečniku, univ. dipl. inž., za vso podporo in spodbudo;
- razredničarki Marjetki Herodež, prof., za veliko razumevanje in vse opravičene ure;
- recenzentu raziskovalne naloge;
- komisiji Mladih raziskovalcev in koordinatorici gibanja Mladi raziskovalci Karmen Hudournik;
- Mateju Zormanu za pomoč pri programiranju v Pythonu;
- vsem neomenjenim, ki so kakorkoli pomagali pri izdelavi naloge.

7 Priloge

Na DVD-ju so priložene v mapah:

7.1 Vsi opisi oziroma postopki izdelave iger.

7.2 Lastno ustvarjene igre.

7.3 E-oblika raziskovalne naloge.

7.4 Python kode iz Blenderja.

7.5 Unity 5 koda.

7.6 XCode kode in app za iOS.

8 Viri in literatura

[1] Apple spletna stran, 29. 01. 2017

[2] Slika logo Blenderja, 29. 01. 2017

[3] Unity podatki, 29. 01. 2017

[4] Quiz Game - Intro and Setup, 29. 01. 2017

[5] Vstavljanje podatkov, 29. 01. 2017

[6] Ustvarjanje menija, 29. 01. 2017

[7] Oblikovanje igre, 29. 01. 2017

[8] Ustvarjanje gumbov, 29. 01. 2017

[9] Prikazovanje vprašanj, 29. 01. 2017

[10] Ustvarjanje kod, 29. 01. 2017

[11] Konec igre, 29. 01. 2017

[12] Primeri kod, 29. 01. 2017

[13] Vir glasbe, 30. 11. 2016

[14] Ustvarjanje menija Blender, 30. 11. 2016

[15] How to Make a Basic Game in Blender • Part 1, 16. 12. 2016

[16] How to Make a Basic Game in Blender • Part 2, 16. 12. 2016

[17] How to Make a Basic Game in Blender • Part 3, 16. 12. 2016

[18] How To Make A Basic Game In Blender • Part 4, 16. 12. 2016

[19] How To Make A Basic Game In Blender • Part 5, 16. 12. 2016

[20] Blender Game Engine Basics Tutorial #1, 16. 12. 2016

[21] Blender Game Engine Basics Tutorial #2, 16. 12. 2016

[22] Blender Game Engine Basics Tutorial #3, 16. 12. 2016

[23] Blender Game Engine Basics Tutorial #4, 16. 12. 2016

[24] Blender Game Engine Basics Tutorial #5, 16. 12. 2016

[25] Blender Game Engine Basics Tutorial #6, 16. 12. 2016

[26] Blender Game Engine Basics Tutorial #7, 16. 12. 2016

[27] Blender Game Engine Basics Tutorial #8, 16. 12. 2016

[28] Blender Game Engine Basics Tutorial #9, 16. 12. 2016

[29] Blender Game Engine Basics Tutorial #10, 16. 12. 2016

[30] First Person Movement, 16. 12. 2016

[31] Blender Game Engine Basics Tutorial #17, 16. 12. 2016

[32] Blender Game Basics Tutorial #18, 16. 12. 2016

[33] Blender Game Engine Basics Tutorial #19, 16. 12. 2016

[34] Blender Game Engine Basics Tutorial #20, 16. 12. 2016

[35] How to Make a Game Menu in the Blender, 16. 12. 2016

[36] How to Make an Advanced Menu in Blender, 16. 12. 2016

[37] Python - How To Create A Random Number Generator, 23. 12. 2016

[38] random numbers in Python, 23. 12. 2016

[39] Python Random Number Generator: the Random Module, 23. 12. 2016

[40] Blender Tutorial on Random Placement, 23. 12. 2016

[41] How to get random numbers in the Blender, 23. 12. 2016

[42] Random Enemy Generation Blender, 23. 12. 2016

[43] Using Radiosity In The BGE, 23. 12. 2016

[44] pomoč pri ustvarjanju s programom Blender, 27. 01. 2017

[45] Koda za prepoznavo miške v Blenderju, 27. 01. 2017

[46] Blender-Python API references, 27. 01. 2017

[47] Koda senzorja miške v Blenderju, 27. 01. 2017

[48] Game logic intruduction, 27. 01. 2017 (GLOBAL)

[49] SCA_MouseSensor, 27. 01. 2017

[50] KNJIGA: Dalai Felinto in Mike Pan, Game development in Blender, Cengage learning, ISBN-13: 978-1-4354-5662-9, 2013

[51] Vodič za izdelavo aplikacije v Swift 3,12. 2. 2017

[52] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[53] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[54] Vodič za izdelavo aplikacije v Swift 3, 12.2.2017

[55] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[55] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[56] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[57] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[58] Vodič za izdelavo aplikacije v Swift 3, 12. 2. 2017

[59]Vodič za izdelavo kviza v Swift 3, 12. 2. 1017

[60] Vodič za izdelavo Osnovnega kviza v Swift 3, 12. 2. 1017

Naša spletna stran: www.preizkusisestudio.wixsite.com/studio.

8.1 AVTORJI RAZISKOVALNE NALOGE

Luka Lah je dijak 1. letnika Elektro in računalniške šole (ERŠ) v Velenju. Za raziskovalno nalogo se je odločil, ker so ga zanimale lastnosti in funkcije Blenderja in Pythona. Zanima ga tudi programiranje s Python-om ter drugimi programskimi jeziki. Zelo je aktiven tudi na športnem področju, kjer je v atletiki v državni reprezentanci. V prihodnosti se želi ukvarjati s programiranjem ter izdelavo raznih aplikacij (slika 60).

Samo Pungaršek Pritržnik je dijak 1. letnika Elektro in računalniške šole (ERŠ) v Velenju. Za raziskovalno nalogo se je odločil, ker so ga zanimali različni načini ustvarjanja iger. Zanimajo ga tudi drugi programski jeziki. V prihodnosti se želi ukvarjati s programiranjem in z zagotavljanjem varnosti v računalniških sistemih.

Žan Novak je dijak 1. letnika Elektro in računalniške šole (ERŠ) v Velenju. Za raziskovalno nalogo se je odločil, ker ga zanimajo električne meritve in elektronska vezja. Zanima ga tudi programiranje uporabnih programov in aplikacij za izboljšanje kakovosti življenja. V prihodnosti se želi ukvarjati s programiranjem.

Slika 60: Mladi raziskovalci Žan Novak, Samo Pungaršek Pritržnik in Luka Lah (z leve proti desni)