SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

MARIN VUKASOVIĆ

RAZVOJ RAČUNALNE IGRE U JEZGRI GODOT ZA PRIMJENU NA MOBILNIM UREĐAJIMA

DIPLOMSKI RAD

Zagreb, 2020.

Sveučilište u Zagrebu
Grafički fakultet

MARIN VUKASOVIĆ

RAZVOJ RAČUNALNE IGRE U JEZGRI GODOT ZA PRIMJENU NA MOBILNIM UREĐAJIMA

DIPLOMSKI RAD

Mentor:
Doc.dr.sc. Tibor Skala

Student:
Marin Vukasović

Zagreb, 2020.

RJEŠENJE O ODOBRENJU TEME DIPLOMSKOG RADA

ZAHVALA

Prvenstveno ovim putem zahvaljujem se svojem mentoru doc.dr.sc. Tiboru Skali na pruženoj potpori tijekom izrade ovog rada. Zahvaljujem se na odličnoj i brzoj komunikaciji, riječima podrške te smjernicama i savjetima u cjelokupnoj izradi koje su izrazito olakšale izradu ovog rada.

Nadalje, zahvaljujem se svojim roditeljima na podršci tijekom trajanja studija. Zahvaljujem se što su me neiscrpno motivirali i polagali vjeru u mene.

Zahvalio bih se svojoj sestri, njezinom mužu te njihovoj djeci na pruženim savjetima i pruženim povratnim informacijama tijekom ispitivanja igre u ovom radu.

Također zahvalio bih se svim svojim prijateljima na pruženoj podršci i pozitivnim kritikama na igru koje su me usmjeravale u izradi.

Naposljetku, zahvalio bih se Grafičkom fakultetu, svim kolegama studentima i profesorima. Zahvalio bih se na pruženim iskustvima, mogućnostima, znanju, veselim i tužnim trenutcima koji su ostavili duboki utisak tijekom ovog razdoblja mojeg života. Taj utisak, uvjeren sam, pomogao mi je da se izgradim kao bolja osoba te pomoći će mi u nošenju sa svim izazovima i nedaćama u budućnosti.

Hvala Vam!

SAŽETAK

Ovaj rad bavit će se razvojem računalne igre u jezgri Godot. Za razvoj odabrana je igra pod nazivom Kraljevska igra iz Ura. Ovo je jednostavna igra za dvoje igrača, a smatra se najstarijom stolnom igrom na svijetu. Ta igra će se igrati na ploči pomoću dvije boje figurica. Za ostvarivanje poteza bacati će se četiri kockice. U Godotu biti će izrađena ta igra koja će se naposljetku izvesti u obliku aplikacije za mobilnu platformu Android. Unutar ovog rada proći će se kroz sve korake tijekom postupka izrade igre. Prvo će biti izrađen programski kôd tijekom postupka izrade igre. Putem tog kôda definirati će se sve funkcionalnosti igre. Zatim biti će izrađeni 3D modeli ploče i figurica unutar programa Blender. Nadalje modelima će se izraditi teksture u programu ArmorPaint. Potom biti će izrađene animacije pomicanja figurica. Zatim biti će izrađeni zvučni efekti i pozadinska glazba u programu Audacity. Naposljetku biti će izrađeno korisničko sučelje te ikona za mobilnu aplikaciju u programu InkScape. Primarni cilj ovog rada je ispitati mogućnosti Godot jezgre tijekom postupka izrade mobilne aplikacije. Sekundarni cilj ovog rada je ispitati mogućnosti ostalih besplatnih programa.

KLJUČNE RIJEČI: Godot jezgra, Kraljevska igra iz Ura, otvoreni kôd, mobilna aplikacija, Android, Blender, ArmorPaint

ABSTRACT

This paper deals with the development of a computer game in the Godot game engine. A game called the Royal Game of Ur was chosen for development. This is a simple two-player game and is considered to be the oldest board game in the world. Usually this game is played on a board using two colored figurines. Four dice will be rolled to make a move. This game will be made in Godot engine and later on it will be exported as an application for Android mobile platform. Within this paper all the steps during the game creation process will be explained. First step in the game creation process will be to create the program code. Through this code all functionalities of the game will be defined. Then 3D models of board and figurines will be made within the Blender program. Furthermore, the models will be textured in ArmorPaint. Animations of moving the figurines will also be made in Godot. Sound effects and background music are going to be created in Audacity. Finally, user interface will be created and the icon for the mobile application will be created in InkScape. The primary goal of this paper is to examine the capabilities of the Godot engine during the mobile application development process. The secondary goal of this paper is to examine the quality of other free programs.

KEY WORDS: Godot engine, Royal Game of Ur, open-source code, mobile application, Android, Blender, ArmorPaint

SADRŽAJ

1. UVOD 1
2. TEORIJSKI DIO 3
2.1. Programi otvorenog kôda 3
2.2. Godot jezgra igre 4
2.3. Kraljevska igra iz Ura 6
2.4. Dodatne korištene aplikacije u ovom radu 9
2.4.1. Blender 9
2.4.2. ArmorPaint 10
2.4.3. Audacity 12
2.4.4. InkScape 13
2.4.5. PureRef 14
3. PRAKTIČNA IZRADA VIDEO IGRE 16
3.1. Programski kôd 16
3.1.1. Godotov pristup (filozofija) u izradi aplikacija 16
3.1.2. Funkcionalnost pojedinih elemenata igre 18
3.1.3. Programski kôd i postavke glavne scene 30
3.2. 3D modeli 49
3.2.1. Prva varijanta 3D modela 49
3.2.2. Druga varijanta 3D modela 57
3.3. Animacija pomicanja figurica 67
3.4. Zvučni efekti i pozadinska glazba 69
3.4.1. Izrada zvučnih efekata 69
3.4.2. Pozadinska glazba 70
3.5. Korisničko sučelje 71

3.5.1. Izrada sučelja za bacanje kockica 71
3.6. Izrada ikone za mobilnu aplikaciju 73
3.7. Izvoz aplikacije na mobilni uređaj 75
4. REZULTAT I RASPRAVA 78
4.1. Izgled i funkcionalnost konačno dobivene igre 78
4.2. Prednosti i nedostatci Godot jezgre 81
4.3. Mogućnosti pri daljnjem razvoju aplikacije 83
5. ZAKLJUČAK 84
6. POPIS SLIKA 85
7. LITERATURA 90
8. POPIS MANJE POZNATIH RIJEČI I AKRONIMA 92
9. PRILOG 94

1. UVOD

U modernom društvu današnjice, koje sve više oblikuju dosezi informacijske tehnologije i napretka, industrija digitalnih igara zauzela je vrhovnu kategoriju medijske industrije. Pritom ostvarujući nezamislive prihode i stječući sve veću popularnost među korisnicima svih dobnih skupina. Eksponencijalnom rastu proizvodnje i potražnje računalnih igara svakako ide u prilog sve veća javna dostupnost besplatnih programa otvorenog kôda, a ti programi nastali su kao rezultat suradnje više ljudi ili skupina, bilo samostalno u izradi ili financijskim donacijama.

Tema ovog rada je razvoj računalne igre za primjenu na mobilnim uređajima. Razvoj igre odraditi će se u Godot jezgri igre, a ovaj program služi za izradu 2D i 3D igara te dostupan je pod *MIT* licencom. Osnovni princip ovog programa je na pristupačan način pružiti potrebite alate za izradu video igara svim korisnicima koji imaju tendencije prema izradi video igara, ali nemaju dovoljno iskustva i znanja kako bi započeli takvo nešto.

Cilj ovog rada je putem izrađene aplikacije ispitati mogućnosti Godot programa, zatim definirati njegovu primjenjivost i u konačnici ispitati funkcionalnost pri izradi igre za mobilne uređaje.

U teorijskom dijelu rada prezentirat će se besplatni programi otvorenog kôda: Godot, Blender, ArmorPaint, Audacity i InkScape koji će biti korišteni prilikom izrade računalne igre. Igra će se izraditi po uzoru na Kraljevsku igru iz Ura. Ovo je jednostavna stolna igra, a smatra se kao 5. najstarija stolna igra na svijetu koja datira iz vremena 2500 g. pr. Kr.

U praktičnom dijelu rada cjelovito će se opisati postupak izrade igre. Pomoću Godota izraditi će se programski i funkcionalni dio aplikacije te korisničko sučelje. U Blenderu i ArmorPaintu izraditi će se 3D modeli, odnosno vizualni dio aplikacije. Dok putem Audacity programa izraditi će se zvučni dio aplikacije. Naposljetku izraditi će se ikona za aplikaciju u InkScape programu. Nakon svih ovih koraka napraviti će se izvoz aplikacije iz Godota na mobilni uređaj.

U završnom dijelu rada prezentirat će se konačan rezultat. Opisati će se izgled i funkcionalnost gotove mobilne aplikacije. Nadalje usporedit će se prednosti i nedostaci Godot programa, a zatim navesti će se mogućnosti pri daljnjem razvoju aplikacije.

2. TEORIJSKI DIO

2.1. Programi otvorenog kôda

"Softver otvorenog kôda je generički je naziv za software čiji je izvorni kôd i/ili nacrt (dizajn) dostupan javnosti na uvid, korištenje, izmjene i daljnje promjene"[1] Programi otvorenog kôda koriste softvere čiji je izvorni kôd dostupan svim korisnicima, može se slobodno skidati sa interneta te mijenjati izvorni kôd s ciljem izrade dostupnih i razumljivih programa te razvoja i unaprijeđenja aplikacija i naobrazbe. Iako je kao open source (slobodan) softwer dostupan svima, razvojem računalnih programa javila se i potreba pravne zaštite programa prema propisima o autorskim pravima. „Ideje i načela na kojima se zasniva bilo koji element računalnog programa, uključujući i one na kojima se zasnivaju sučelja, nisu zaštićeni autorskim pravom".[2] Posljedično, javlja se veliki broj licenci sa ciljem definiranja i obaveza, ne samo autora već i korisnika softvera otvorenog tipa.

„Računalni program koristi se na temelju licencije kojom nositelj prava daje ovlaštenom korisniku odobrenje za korištenje i određuje naknadu. No Direktiva o zakonskoj zaštiti računalnih programa propisuje iznimke od isključivih prava autora računalnog programa koje su sadržane i u hrvatskom zakonu."[3] Autor zadržava pravo izvornosti kod daljnje distribucije kôda čime mora biti vidljivo tko je autor prve verzije programa, dok je korisnik obvezan sačuvati informaciju o autoru programa.

Softver otvorenog kôda (Open-source software (OSS)) „generički je naziv za software čiji je izvorni kôd i/ili nacrt (dizajn) dostupan javnosti na uvid, korištenje, izmjene i daljnje promjene (primjeri: Firefox web preglednik, MediaWiki softver, Joomla, Linux, Android) kad se eng. termin *open source*

[1] Otvoreni kod, na web stranici http://otvorenikod.weebly.com/ (pristupljeno: 10. rujna 2020.)
[2] Usp. HORVAT, Aleksandra
Knjižnice i autorsko pravo / Aleksandra Horvat, Daniela Živković. - Zagreb : Hrvatska sveučilišna naklada, 2009. - 189 str. ; 24 cm. - 89. str.
[3] Usp. HORVAT, Aleksandra
Knjižnice i autorsko pravo / Aleksandra Horvat, Daniela Živković. - Zagreb : Hrvatska sveučilišna naklada, 2009. - 189 str. ; 24 cm. - 90. str.

prevodi kao otvoreni izvor, onda se odnosi na bilo kakvu informaciju, bilo pisanu ili audiovizualnu, koja je slobodno dostupna javnosti”[4].

Kako su za programe otvorenog kôda potrebni znatno niži troškovi marketinga, njegova dostupnost je znatno veća nego kod nabave vlasničkih softvera. Preferencije tvrtki koje koriste ovakve slobodne softvere idu u korak sa tehnološkim razvojem što rezultira dobrom promocijom tvrtki i brzu i jeftinu proizvodnju kvalitetnog i pouzdanog softvera. Softver otvorenog kôda, zbog svoje dostupnosti, pospješuje brzi razvoj tehnologije i inovacija kao rezultat suradničkog odnosa brojnih programera. Tvrtke bazirane na inovacijama sve se manje usmjeravaju na prodaju vlasničkih softvera čime se sve više gubi njihova potreba. Koji su razlozi preferencije open source softvera? Četiri su najčešća razloga:

- Niža cijena
- Sigurnost
- Nema dobavljača
- Bolja kvaliteta[5]

Kao i u svakom segmentu razvoja modernog kapitalističkog društva, uspjeh određenog proizvoda ovisi o ponudi i potražnji i samom profitu koji on donosi. Ovisno o vrsti poslovne djelatnosti, uvođenjem softvera otvorenog kôda dugoročno se stvara profit kroz uštede i fleksibilnost prilagodbe softvera određenim zadacima u realnom vremenu.

2.2. Godot jezgra igre

Godot jezgra igre besplatni je program otvorenog kôda za izradu 2D i 3D igara. Program je dostupan pod *MIT* licencom. Tvorci programa su Juan Linietsky i Ariel Manzur, a njegovu izradu započeli su 2007. godine. Sedam godina kasnije, 2014. godine kôd jezgre postaje javno dostupan.

[4] Software, na web stranici http://otvorenikod.weebly.com/ (pristupljeno: 10. rujna 2020.)
[5] CMSWIRE, Bad Economy Is Good for Open Source (2009.g), autor: Irina Guseva, na web stranici: http://www.cmswire.com/cms/web-cms/bad-economy-is-good-for-open-source004187.php (pristupljeno: 10. rujna 2020.).

Slika 1. Prikaz logotipa Godot programa
Izvor: https://upload.wikimedia.org/wikipedia/commons/6/6a/Godot_icon.svg

Pomoću ovog programa može se na vrlo jednostavan način izraditi 2D ili 3D igra. Cijeli program koncipiran je s idejom da bude što pristupačniji korisnicima koji imaju slabo znanje iz područja programiranja ili općenito područja razvoja video igara. Osnovni objekti s kojima se može izrađivati igra u Godotu su: čvor, scena i skripta [5]. Čvor može biti bilo koji objekt u 2D ili 3D prostoru. Scena je objedinjeni skup tih objekata. Nadalje putem skripte dodjeljivati će se objektima i scenama određene funkcionalnosti.

Godotov interni programski jezik naziva se *GDScript*, odnosno Godot skripta. Sintaksa ovog programskog jezika relativno je jednostavna. Novi korisnici ne bi trebali imati problema s usvajanjem sintakse jer je osmišljena tako da naredbe se pišu logičkim slijedom. Zbog takve sintakse slijed definiranja naredbi unutar kôda na intuitivan način je posložen. Na primjer: dohvaćanje čvora unutar skripte te izvođenje pojedine funkcije na tom čvoru može se izvesti unutar jedne linije kôda. Ta linija kôda biti će postavljena na način: 1. dohvati čvor, 2. izvedi naredbu na dohvaćenom čvoru. Također program podržava izvođenje kôda koji je napisan u *C++* i *C#* programskim jezicima.

Čvorovi su vrlo moćan alat unutar Godota. Zadani čvorovi u programu definirani su na način tako da je korisniku pri prvom pokretanju programa odmah dostupan skup objekata pomoću kojih se može izraditi skoro pa svaki element unutar 2D ili 3D igre.

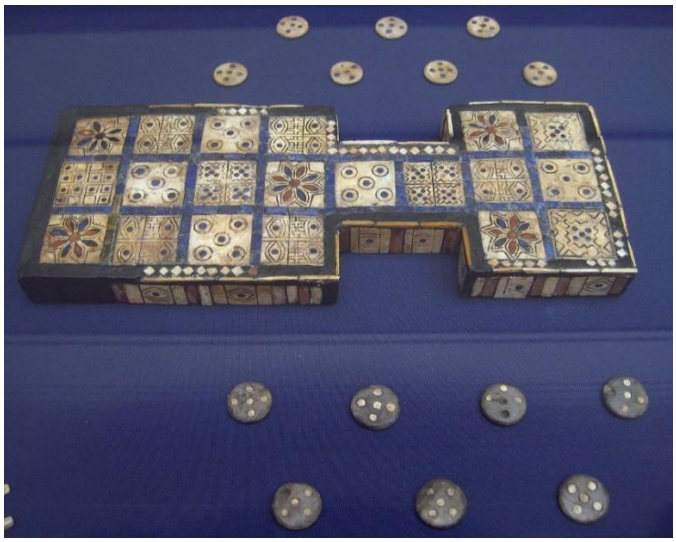
Nadalje, proces izvoza aplikacije vrlo je jednostavan i intuitivan – u nekoliko poteza može se automatski pokrenuti izvoz aplikacije za više platformi odjednom. Godot podržava izvoz na sljedeće platforme: *Android, iOS, HTML5, MacOSX, UWP, Windows* i *Linux*.

Godotova datoteka u kojoj se nalazi program zauzima vrlo malo prostora u pohrani (svega 30-ak megabajta). Također prije pokretanja samog programa nije potrebno izvršiti nekakvu instalaciju. Preuzeta datoteka automatski je spremna za pokretanje. Ovo je izrazito praktično jer je zbog male veličine datoteke olakšan pristup te omogućena je veća prenosivost razvojnog okruženja na širi spektar računala.

Za potrebe ovog rada kompletna igra za mobilni uređaj biti će izrađena u Godotu. Prvo će se definirati glavni elementi unutar igre. Zatim napraviti će se zasebne scene koje će predstavljati te glavne elemente igre. Svakoj od tih scena dodijeliti će se skripta koja će sadržavati potrebne funkcionalnosti za zadanu scenu. Naposljetku izraditi će se korisničko sučelje te materijali za naknadno uvezene 3D modele.

2.3. Kraljevska igra iz Ura

Kraljevska igra iz Ura je jednostavna stolna igra za dva igrača. Starost te igre procjenjuje se na 2500. godinu pr. Kr. Zbog ovoga igra se smatra 5. najstarijom stolnom igrom na svijetu [3]. Pločica zajedno s figuricama otkrivena je između 1922. i 1934. godine. Otkrio ju je Sir Leonard Woolley prilikom arheoloških iskapanja Kraljevskog groblja u Uru. Lokacija ovog groblja nalazi se danas u južnom Iraku. Upute za igru pronađene su na babilonskoj glinenoj pločici, a starost te pločice datira u 2. st. pr. Kr. Tekstualni sadržaj te glinene pločice odgonetnuo je kustos Britanskog muzeja, Irving Finkel.



Slika 2. Prikaz replike pločice koja je pronađena tijekom iskapanja

Izvor:
https://commons.wikimedia.org/wiki/File:British_Museum_Royal_Game_of_Ur.jpg#/media/File:British_Museum_Royal_Game_of_Ur.jpg

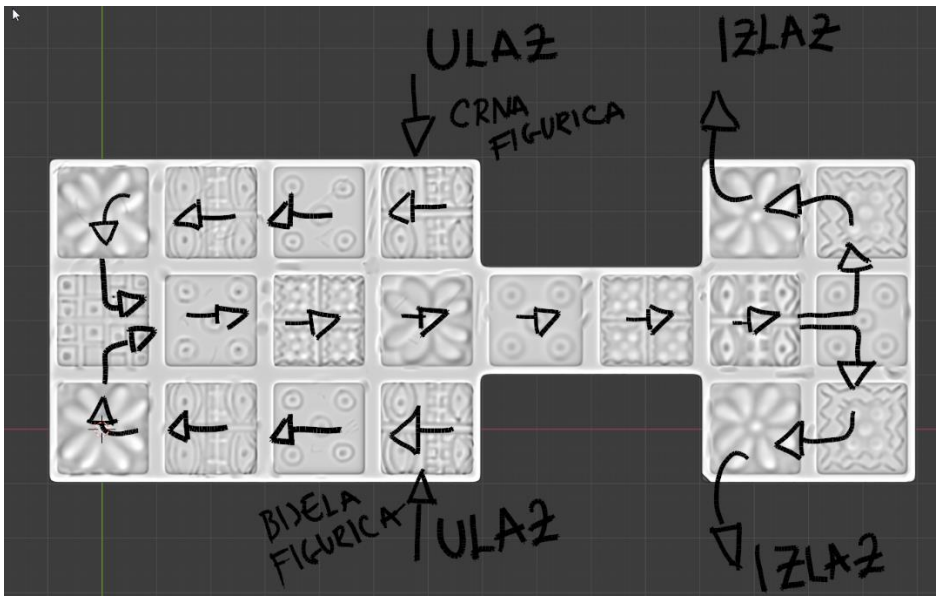
Igra se odvija na pločici s ucrtanim poljima. Po tim poljima pomiču se bijele i crne figurice. Kako bi igrači ostvarili potez figuricom, moraju baciti 4 kockice. Te kockice imaju oblik piramide te nakon bacanja daju vrijednost 0 ili 1.

Slika 3. Prikaz kockica za bacanje

Izvor: https://cf.geekdo-images.com/imagepage/img/wrr5hLLiMEHdTHU1BJ0kpDi4WTw=/fit-in/900x600/filters:no_upscale():strip_icc()/pic1503607.jpg

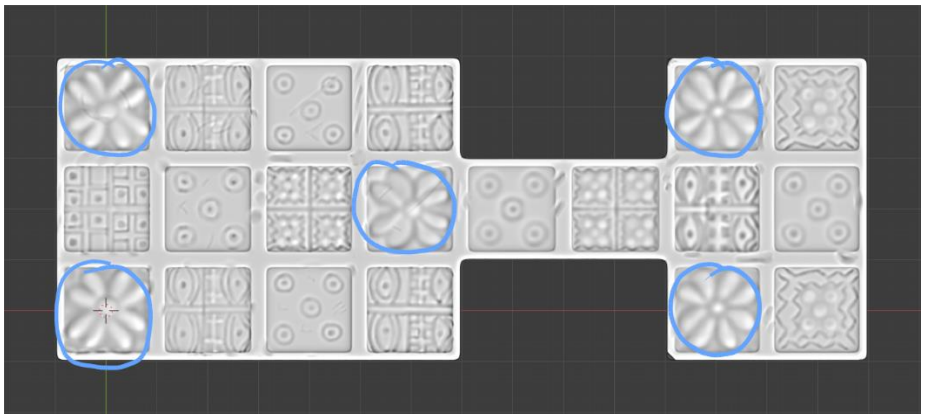
Pločica na sebi sadrži sveukupno 20 polja. Prvih četiri polja za svakog igrača su početna polja. Pomoću ovih polja figurice će ulaziti u igru. Na središnjih osam polja na pločici odvijati će se borba između figurica. Jedino na tih osam polja jedna figurica moći će izbaciti drugu figuricu iz igre. Zadnja dva polja su završna polja za oba igrača. S tih polja figurice će izlaziti van s ploče.

Cilj igre je izaći sa svih 5 ili 7 figurica izvan igre prije protivničkog igrača. Smjer kretanja figurica za oba igrača može se vidjeti na slici.



Slika 4. Prikaz smjera kretanja figurica na ploči

Na ploči se također nalazi 5 polja s oznakom cvijeta. Ta polja još nose naziv rozeta polje. Ukoliko igrač stane s figuricom na to polje, ostvariti će mogućnost za dodatno bacanje kocke. Postoji puno različitih pravila za vrstu puta kojim će se kretati figurice na ploči, ali za igru koja će se izrađivati u ovom radu koristiti će se klasična pravila sa po 6 figurica za svakog igrača.



Slika 5. Prikaz ploče s označenim rozeta poljima

2.4. Dodatne korištene aplikacije u ovom radu

2.4.1. Blender

Blender je besplatni programski paket otvorenog kôda. Ovaj program poznat je prvenstveno kao program za izradu 3D modela, ali ima puno širi spektar primjene. Pomoću ovog programa mogu se: izrađivati 3D animacije, iscrtavati 3D scene, obrađivati video zapisi, izrađivati pokretne grafike, itd. Zbog svih tih silnih mogućnosti koje Blender nudi, a i zbog nepostojeće cijene programa, Blender iz godine u godinu postaje sve popularniji. Tome svjedoči i činjenica da je prošle godine američka kompanija *Epic Games* donirala Blender fundaciji 1.2 milijuna američkih dolara [1]. Na temelju ove činjenice može se vidjeti kako je Blenderova budućnost sve samo ne neizvjesna, što je itekako bitno s obzirom da se radi o besplatnom programu. Nadalje bitno je napomenuti da puno malih i srednjih po veličini studija za izradu video igara koristi Blender za svoje komercijalne potrebe. Ovo dodatno podiže Blenderovu popularnost u svijetu.

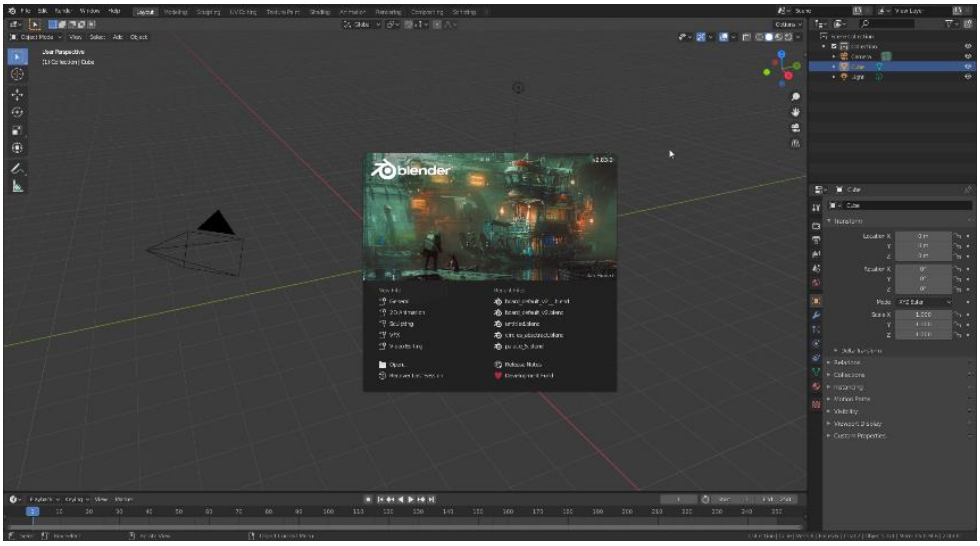
Slika 6. Prikaz logotipa Blender programa

Izvor: https://download.blender.org/branding/community/blender_community_badge_orange.png

Izradu Blendera započeo je 1994. godine Nizozemac Ton Roosendaal, a 1998. godine izašla je prva službena verzija programa. Ta prva verzija bila je isključivo namijenjena za *Linux* računalnu platformu. Kasnije Tonu se pridružuje nekoliko volontera unutar Nizozemske te osnovana je nova kompanija *Not a Number* putem koje je Ton Roosendaal pokušao prodavati Blender program. No, nakon četiri godine njegova kompanija doživaljava krah i investitori povlače svoje investicije iz kompanije. Blender program iduću prekretnicu doživljava 2002. godine kada je osnovana Blender fundacija. Ovo je ustvari bila neprofitna organizacija koju je osnovao Ton Roosendaal. Putem te organizacije Ton je

pokušavao pronaći način da i dalje omogući razvoj Blender programa. Došao je na ideju da Blenderov programski kôd učini dostupnim cijelom svijetu, tj. htio je Blender pretvoriti u program otvorenog kôda. Putem kampanje *Slobodni Blender* (eng. *Free Blender*), Ton je uspio ostvariti dogovor s prijašnjim investitorima. Blender fundaciji donirano je 100 000€ te Blender je postao program otvorenog kôda [2]. Od onda do danas Blender fundacija narasla je na 15-ak zaposlenika koji rade na razvoju i testiranju Blender programa.

U ovom radu Blender će biti korišten za izradu 3D modela figurica i ploče. Izrađivati će se dvije varijante modela. U prvoj varijanti, modeli figurice i ploče imati će vrlo jednostavnu topologiju. Za drugu varijantu figurice raditi će se model s velikim brojem poligona te pomoću tog modela izrađivati će se tekstura sa zapisom o smjeru normala 3D modela. Na kraju, izvesti će se iz programa prva varijanta modela i druga varijanta modela s normal teksturom. Modeli i teksture biti će prosljeđene u ArmorPaint program.



Slika 7. Prikaz korisničkog sučelja u Blender programu

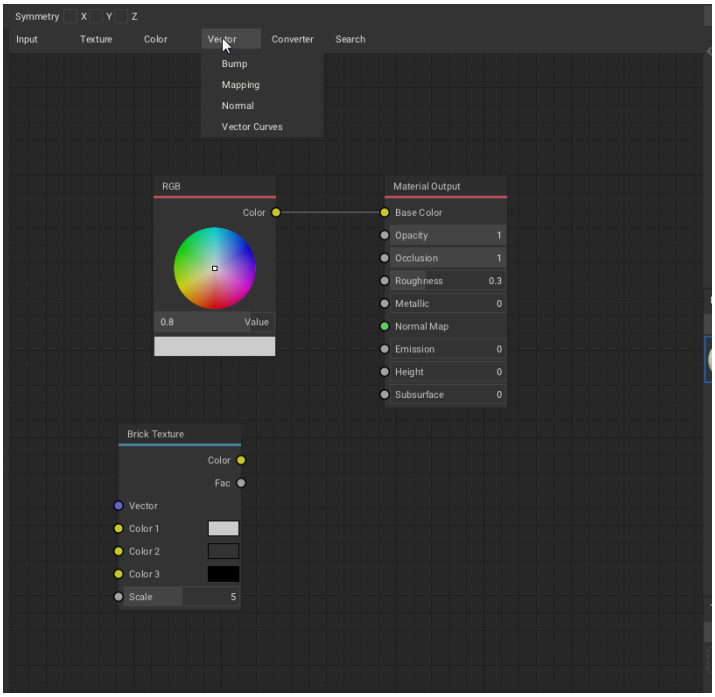
2.4.2. ArmorPaint

ArmorPaint program je besplatni program otvorenog kôda. Pomoću ovog programa mogu se putem *PBR* metode izrađivati teksture za 3D modele. Tvorac ovog programa je Lubos Lenco, programer iz Slovačke.

Slika 8. Prikaz logotipa ArmorPaint programa

Izvor: https://armorpaint.org/img/Logo.png

U ovom programu mogu se ručno crtati materijali po 3D modelu, a putem sustava slojeva mogu se razgraničavati područja na modelu između više različitih materijala. ArmorPaint nastao je kao besplatna alternativa Substance Painter[6] programu. Razvojni se programer tijekom procesa dizajniranja sučelja jako referirao na izgled Blenderovog sučelja. Korisničko sučelje (eng. *User Interface*) ArmorPainta u puno stvari podsjeća na Blender - kratice za naredbe unutar programa gotovo su identične Blenderovim kraticama, a sustav čvorova za izradu materijala direktno je preuzet iz Blendera. Zbog ovih sličnosti s Blenderom odabran je ovaj program za potrebe ovog rada.



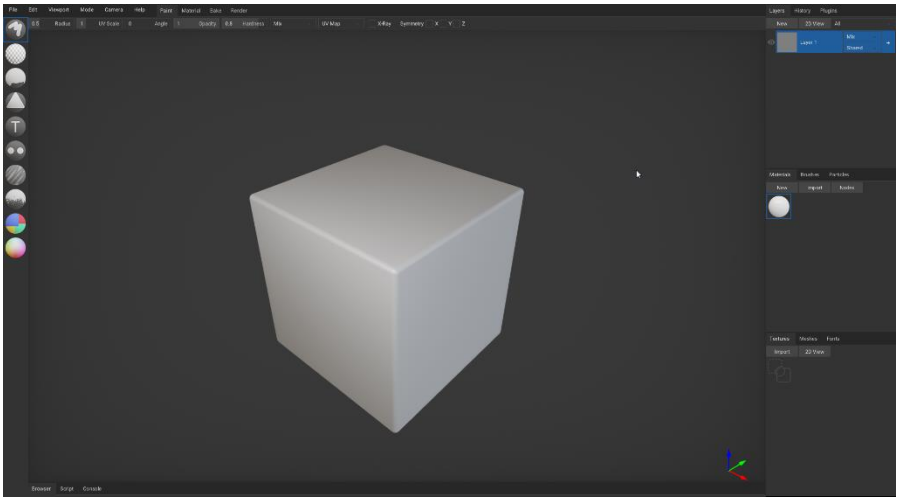
Slika 9. Prikaz sustava čvorova za izradu materijala u ArmorPaintu

[6] *Substance Painter* – plaćeni program za teksturiranje 3D modela; ovaj program smatra se industrijskim standardom

S obzirom da je ArmorPaint još uvijek u razvoju, odnosno nije još službeno izašao program, prilikom korištenja programa može se očekivati pojava grešaka. S obzirom na cijenu programa, ovaj nedostatak je zanemaren.

Teksture unutar ovog programa izrađuju se putem *PBR* metode. Fizikalno-definirano iscrtavanje je način na koji će se definirati teksture za prikazivanje na sceni. U prošlosti je bila popularna klasična metoda, tzv. *Diffuse/Specular* metoda [4]. Ovom metodom prilikom izrade nisu se uzimala u obzir fizikalna svojstva materijala koji se izrađuju. Zbog ovoga bilo je često vrlo teško postići uvjerljiv izgled materijala. Pomoću nove *PBR* metode mogu se postići izrazito uvjerljivi materijali na relativno jednostavan način.

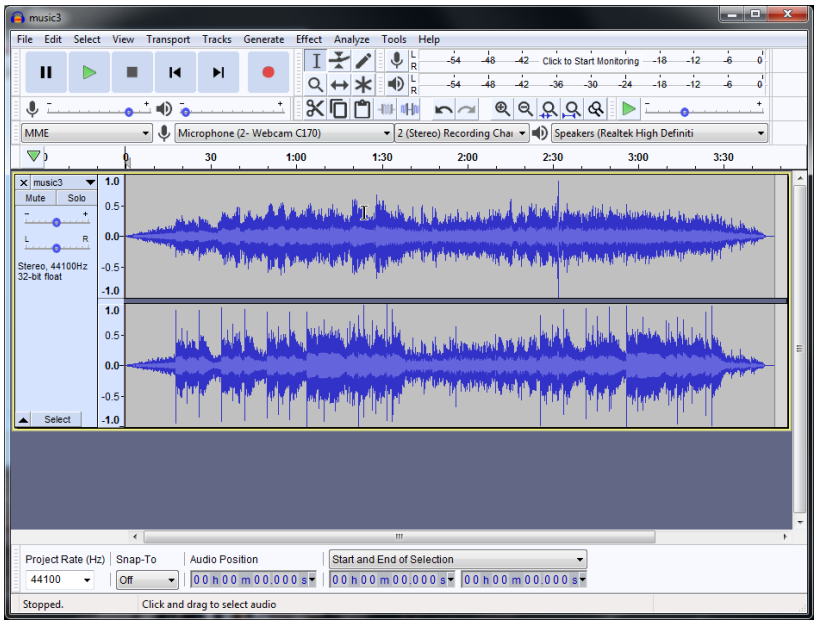
U ArmorPaint programu izrađivati će se teksture za drugu varijantu modela ploče i figurica. Nakon gotove izrade, teksture će biti izvezene iz programa te uvezene u Godot jezgru.



Slika 10. Prikaz korisničkog sučelja ArmorPainta

2.4.3. Audacity

Audacity je besplatan program otvorenog kôda koji služi za uređivanje zvukovnih datoteka. Program je nastao 1999. godine, a osmislili su ga Dominic Mazzoni i Roger Dannenberg. Program podržava sljedeće formate glazbenih datoteka: *.wav, .aiff, .ogg* i *.mp3*.

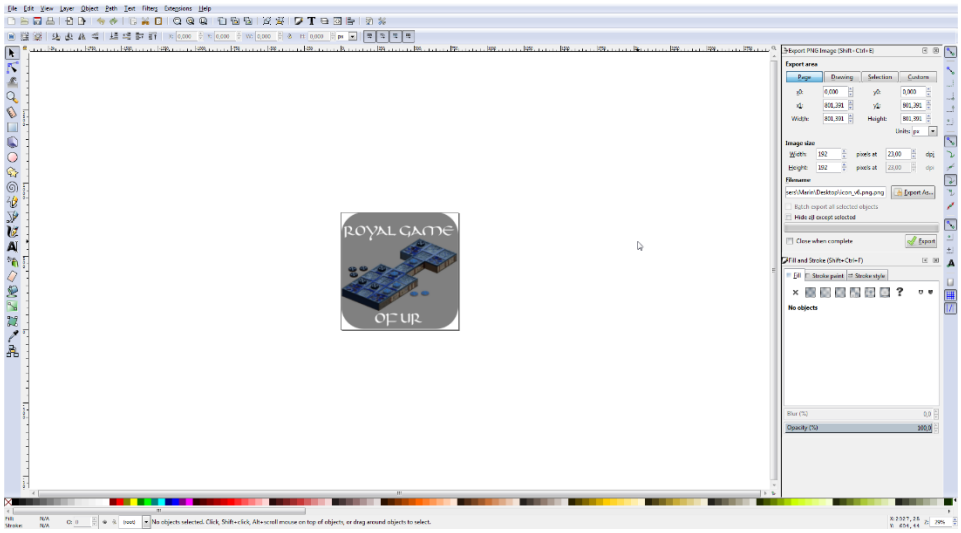


Slika 11. Prikaz korisničkog sučelja Audacity programa

Pomoću ovog programa izrađivati će se zvučni efekti i pozadinska glazba za igru. Prilikom izrade zvučnih efekata, pomoću programa Audacity datotekama će se dodjeljivati različiti zvučni efekti pomoću kojih će se dobivati različite varijante zvukova. Nakon izrade zvukova, datoteke će se izvoziti iz programa u *.ogg* i *.wav* formatima.

2.4.4. InkScape

InkScape je program namijenjen za izradu vektorske grafike. Besplatan je i ima otvoreni kôd. Ovaj program može se smatrati besplatnom alternativom u odnosu na Adobeov Illustrator.

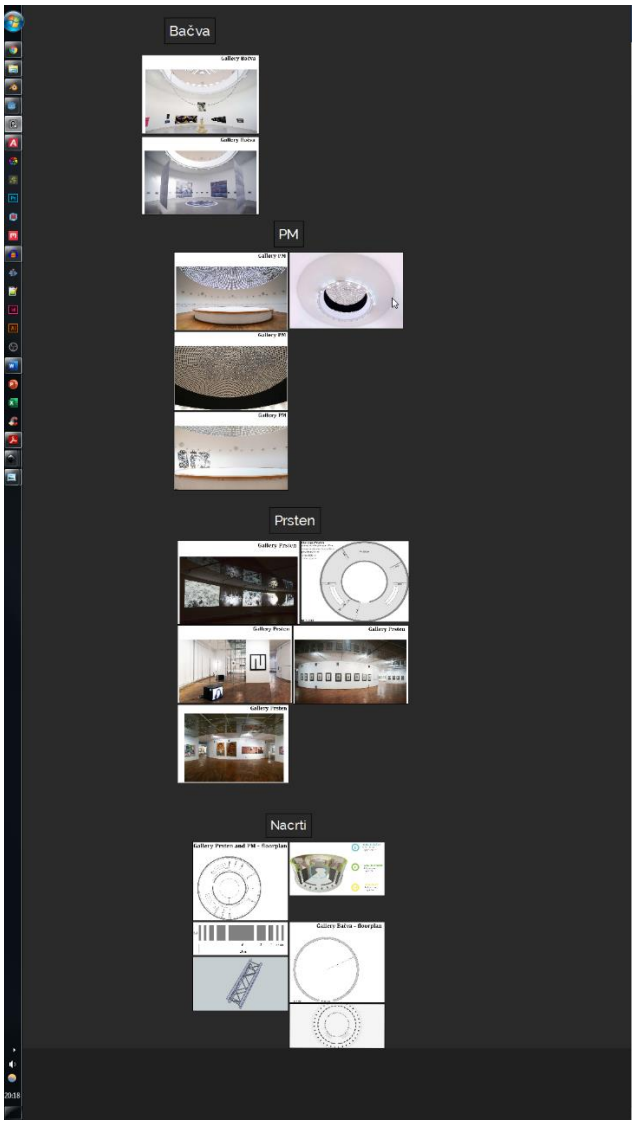


Slika 12. Prikaz korisničkog sučelja u InkScape programu

InkScape omogućava korisniku izradu i manipulaciju različitih vektorskih oblika. Za izradu jednostavnijih grafika, mogućnosti programa su i više nego dovoljne. Ukoliko se u program unese kompleksnija grafika, program počinje zastajkivati što jako ometa cjelokupni rad u programu. Za izradu ikone koja će imati dimenzije 192 x 192 piksela ne mogu se očekivati prevelika opterećenja programa i stoga InkScape je sasvim idealan program za to.

2.4.5. PureRef

PureRef je program za postavljanje i prikaz referentnih slika. Program nije otvorenog kôda, ali je zato besplatan. Ovaj program postao je izuzetno popularan zbog svoje nepostojeće cijene i mogućnosti organizacije referentnih slika. Velika prednost ovog programa je ta da se može bilo koja slika (čak i preuzeta s interneta) pomoću naredbi *Kopiraj* i *Zalijepi* (eng. *Copy&Paste*) vrlo jednostavno postaviti u program. Također jedna od vrlo korisnih mogućnosti je normalizacija referentnih slika. Prilikom unosa više slika u program, slike će biti različitih dimenzija. PureRef program omogućava na vrlo brz način normalizaciju dimenzija svih slika - svim slikama postaviti će se iste dimenzije, bez obzira na rezoluciju slike. Još jedna dobra mogućnost je ta da PureRef sve slike tretira na jednak način, tj. program neće zastajkivati, bez obzira na rezoluciju i dimenzije unesenih slika.



Slika 13. Prikaz PureRef programa s postavljenim referentnim slikama

Ovaj program biti će korišten tijekom modeliranja ploče i figurica. Unutar programa biti će postavljene referentne slike na koje će se referencirati tijekom izrade detalja u Blender i tijekom izrade boja na teksturama u ArmorPaint programu.

3. PRAKTIČNA IZRADA VIDEO IGRE

3.1. Programski kôd

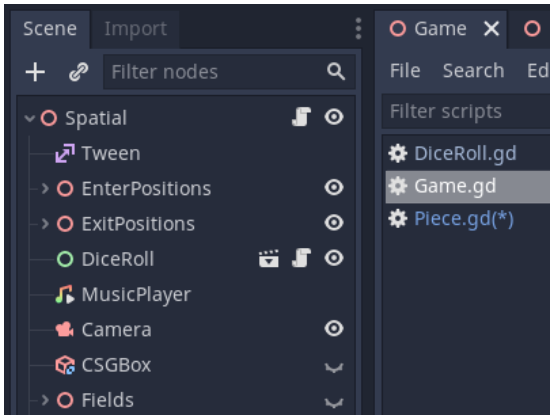
3.1.1. Godotov pristup (filozofija) u izradi aplikacija

Prije početka razvoja bilo kakve video igre u Godot jezgri bitno je obratiti pozornost na Godotov specifičan pristup, odnosno specifičan način u organizaciji pojedinih elemenata unutar igre. Taj Godotov pristup njegovi razvojni programeri nazvali su sustav čvorova (eng. *Node System*[7]). Skup tih istih čvorova nazvali su scena (eng. *Scene*). Ovakav sustav organizacije nalaže da svi elementi unutar igre imaju nekakvu srodstvenu poveznicu [5]. Ta poveznica manifestira se na idući način: jedan čvor koji se nalazi na višoj poziciji u hijerarhiji smatrati će se glavnim čvorom, odnosno roditeljem, dok ostali čvorovi koji se nalaze niže u hijerarhiji smatrati će se djecom tog glavnog čvora. Ovakva srodstvena veza između svih čvorova omogućava glavnoj sceni, tj. roditelju pristup svim varijablama i funkcijama, odnosno funkcionalnostima, koje su sadržane u čvorovima djeci[8]. Osobi koja izrađuje igru omogućeno je putem ovakve razvojne filozofije da najosnovnije funkcionalnosti pojedinih elemenata napravi i zadrži u tom samom elementu (čvoru). Kasnije po potrebi taj određeni skup čvorova, tj. scenu moguće je dodati (instancirati) na glavnu scenu gdje će se odvijati igra.

U Godot jezgri postoje tri najosnovnije jedinice koje čine neki element unutar igre. To su: čvor (eng. *Node*), scena (eng. *Scene*) i skripta (eng. *Script*). Čvor je bilo koji element koji se nalazi u igri, a to može biti na primjer: kamera, 2D tekstura, obujmica za sudare, 3D mreža modela (eng. *mesh*[9]), rasvjetno tijelo itd. Scena je obično skup čvorova u 2D ili 3D prostoru. Skripta je programski kôd pomoću kojega se dobivaju određene funkcionalnosti. Te

[7] Sustav čvorova (eng. *Node System*) – ovo je sustav unutar Godot programa pomoću kojeg se izvodi organizacija svih elemenata sadržanih unutar aplikacije
[8] Odnos između objekata roditelj-dijete (eng. *Parent-child object relation*) – u OOP ovo je metoda pomoću koje se definiraju poveznice među objektima; na objektu-roditelju definirane su određene funkcionalnosti koje objekti-djeca nasljeđuju; objekt-roditelj ne može naslijediti funkcionalnosti objekta-djeteta
[9] Žičana struktura 3D modela (eng. *Mesh*) – u računalnoj grafici 3D modeli definiraju se kao skup točaka s točno-preciziranim pozicijama u 3D prostoru; te točke zatim tvore plohe (poligone) pomoću kojih se dobiva cjelokupni oblik 3D modela

funkcionalnosti mogu se odvijati: 1. unutar samih čvora (npr. figurica mijenja boju), 2. u njihovoj međusobnoj interakciji (npr. izbacivanje figurica na ploči) ili 3. unutar glavne scene (npr. mjerač za preostali broj figurica u igri). Važno je opet napomenuti da svaka scena, odnosno skup čvorova može se po završetku njezine izrade dodati na glavnu scenu. Drugim riječima pojedini element igre može se napraviti kao zasebna scena. Dijelovi tog elementa bit će izrađeni pomoću čvorova, njegova funkcionalnost bit će sadržana u skripti te na kraju biti će pohranjen u obliku scene. Kasnije se ta scena može dodati (instancirati) u obliku čvora na glavnu scenu gdje će se odvijati igra.



Slika 14. Prikaz scene, sustava čvorova i skripte unutar Godot programa

Instanciranje[10] u objektno-orijentiranom programiranju (skraćeno OOP[11]) je način na koji se može stvoriti objekt dok je programski kôd već u izvedbi (eng. *runtime*[12]). Pomoću instanciranja može se, u slučaju Godot jezgre, npr. stvoriti mnoštvo objekata na zaslonu mobitela. Svi ti objekti imaju iste funkcionalnosti i izgled, no zbog činjenice da su instancirani na zaslon, u memoriji uređaja pohranjen je samo jedan objekt [5]. Ovakav pristup pomoću instanciranja objekata uvelike smanjuje opterećenje uređaja prilikom izvođenja programskog kôda. Uzimajući ovo u obzir instanciranje je idealno za mobilnu platformu zbog ograničene procesorske snage mobilnih uređaja danas.

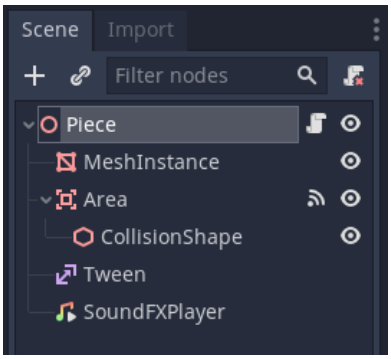
[10] Instanciranje (eng. *Instancing*) – u OOP ovo je način na koji se kreiraju objekti određene klase
[11] Objektno-orijentirano programiranje (eng. *Object-oriented programming*) – pristup u programiranju; putem ovakvog pristupa programski kod projektira se kao skup objekata koji između sebe vrše određenu komunikaciju
[12] Vrijeme izvođenja (eng. *Runtime*) – u programiranju vrijeme izvođenja definira se kao vrijeme tijekom kojega izvorni programski kod izvodi se na centralnoj procesorskoj jedinici

Na primjeru Kraljevske igre iz Ura jasno postaje vidljivo da prilikom izrade igre kada se ovakva Godotova razvojna filozofija uzme u obzir, elementi igre sačinjeni su od triju osnovnih dijelova: jedne figurice, jednog polja na ploči te jedne kockice. Ta tri dijela (djeca) su onda sadržana u glavnom (roditelj), četvrtom dijelu gdje se odvija cjelokupna igra. Vođeni ovakvim promišljanjima pokušalo se pojedine elemente unutar igre izraditi na zaseban način.

3.1.2. Funkcionalnost pojedinih elemenata igre

3.1.2.1. Figurica

Za glavnu figuricu napravljena je zasebna scena koja je pohranjena s nazivom *Piece_master.tscn*[13].



Slika 15. Izgled sustava čvorova za glavnu figuricu

Pomoću sustava čvorova glavnoj figurici određene su iduće funkcionalnosti:

- *Piece (3D Prostorni)* – glavni čvor
  - *Instanca Mreže* – figurica ima svoj prikaz u 3D prostoru
  - *Površina* – figurica će moći registrirati unos dodirom unutar svog volumena
  - *Tween* – figurica će se moći animirati u 3D prostoru
  - *SoundFXPlayer* – figurica će prilikom interakcije reproducirati zvuk

Desnim klikom miša na glavni čvor i odabirom opcije *Uglavi skriptu* (eng. *Attach Script*) dodana je skripta, odnosno programski kôd koji će sadržavati

[13] *.tscn* - sufiks za datoteku koju će Godot program prepoznati kao scenu

varijable i funkcije vezane za funkcionalnosti figurice. Skripta je nazvana *Piece.gd*[14].

```
1  extends Spatial
2
3  signal piece_pressed
4  signal piece_released
5
6  onready var white_default_MAT = preload("res://Materials/white_default_MAT.tres")
7  onready var white_default_v2_MAT = preload("res://Materials/white_default_v2_MAT.tres")
8  onready var white_inactive_MAT = preload("res://Materials/white_inactive_MAT.tres")
9  onready var white_inactive_v2_MAT = preload("res://Materials/white_inactive_v2_MAT.tres")
10 onready var black_default_MAT = preload("res://Materials/black_default_MAT.tres")
11 onready var black_default_v2_MAT = preload("res://Materials/black_default_v2_MAT.tres")
12 onready var black_inactive_MAT = preload("res://Materials/black_inactive_MAT.tres")
13 onready var black_inactive_v2_MAT = preload("res://Materials/black_inactive_v2_MAT.tres")
14 onready var piece_move_elig_MAT = preload("res://Materials/piece_move_eligible_MAT.tres")
15 onready var piece_move_inelig_MAT = preload("res://Materials/piece_move_ineligible_MAT.tres")
16
17 onready var mesh_simple = preload("res://Models/default_piece.obj")
18 onready var mesh_complex = preload("res://Models/default_piece_v2.obj")
19
20 onready var MovePieceTween = get_node("Tween")
21
22 onready var move_piece_sound1 = preload("res://SoundFX/piece_move_orig.wav")
23 onready var move_piece_sound2 = preload("res://SoundFX/piece_move_slow.wav")
24 onready var move_piece_sound3 = preload("res://SoundFX/piece_move_fast.wav")
25 onready var remove_piece_sound = preload("res://SoundFX/piece_removed.wav")
26 onready var select_piece_sound = preload("res://SoundFX/piece_selected.wav")
27 onready var playable_sound_FX = [move_piece_sound1, move_piece_sound2, move_piece_sound3]
28
29 onready var simple_mesh = true
30 onready var currentPos = -1
31 onready var currentColor = "white"
32 onready var isInGame = true
33 onready var touchCounter = 0
34 onready var isMoveEligible = false
```

Isječak kôda 1. Prikaz definiranih varijabli za glavnu figuricu

Pri vrhu skripte nalazi se *extend Spatial* što znači da već definiranu klasu *3D Prostor* unutar Godot jezgre želi se proširiti dodatnim funkcionalnostima. Definirana su dva signala unutar kôda: *piece_pressed* i *piece_released*.

U ovom slučaju definirani su nazivi dva signala koji će se emitirati kada se označi figurica i kada se odznači figurica, a taj signal osluškivat će glavna scena igre. Nadalje, definirano je deset varijabli u kojima su pohranjeni materijali za figurice. U Godot jezgri bilo kakav vanjski resurs (npr. teksturu,

[14] *.gd* – sufiks za datoteku koju će Godot prepoznati kao skriptu

zvuk, materijal) najbolje je pohraniti u varijablu [5] jer je onda zadani resurs puno lakše referencirati unutar kôda.

Naredba *onready* koristi se kada se želi napraviti inicijalizacija varijabli nakon što je igra pokrenuta i čvorovi na sceni su učitani [5]. Pomoću naredbe *preload* zadani materijali pohranjuju se u memoriju čime dostupni su odmah tijekom izvođenja igre te ih nije potrebno naknadno učitavati.

Prvih deset definiranih varijabli u sebi pohranjuju materijale za bijele i crne figurice. U idućoj varijabli *MovePieceTween* definirano je da pozivanjem ove varijable se želi dohvatiti *Tween* čvor iz sustava čvorova. U idućih 6 varijabli definirani su zvučni zapisi koji će se reproducirati prilikom pomicanja, izbacivanja i selektiranja figurica. Putem iduće definirane varijable *simple_mesh* pratiti će se koja varijanta modela figurice je trenutno postavljena u igri. Pomoću iduće varijable *currentPos* definirana je trenutna pozicija figurice u igri – ova vrijednost kretati će se od -1 do 14 za svaku figuricu. Varijabla *currentColor* označava boju figurice. Nadalje varijablom *isInGame* označava se je li odabrana figurica još uvijek u igri. Pomoću varijable *touchCounter* pokušalo se ograničiti broj dodira na figuricu. I u konačnici, zadnje definirana varijabla *isMoveEligible* u sebi pohranjuje vrijednost koja ukazuje je li potez figurice moguć ili nemoguć nakon bacanja kockica.

```
37 func set_piece_white_active():
38     $Area.show()
39     if simple_mesh == true:
40         $MeshInstance.set_material_override(white_default_MAT)
41     else:
42         $MeshInstance.set_material_override(white_default_v2_MAT)
43
44 func set_piece_white_inactive():
45     $Area.hide()
46     if simple_mesh == true:
47         $MeshInstance.set_material_override(white_inactive_MAT)
48     else:
49         $MeshInstance.set_material_override(white_inactive_v2_MAT)
```

Isječak kôda 2. Prikaz kôda za aktivni i neaktivni materijal bijele figurice

U prvoj funkciji *set_piece_white_active():* bijela figurica postaviti će se u aktivni status. Definirano je da se na poziv ove funkcije uključi obujmica koja registrira korisnikov dodir.

```
func set_piece_black_active():
    $Area.show()
    if simple_mesh == true:
        $MeshInstance.set_material_override(black_default_MAT)
    else:
        $MeshInstance.set_material_override(black_default_v2_MAT)

func set_piece_black_inactive():
    $Area.hide()
    if simple_mesh == true:
        $MeshInstance.set_material_override(black_inactive_MAT)
    else:
        $MeshInstance.set_material_override(black_inactive_v2_MAT)
```

Isječak kôda 3. Prikaz kôda za aktivni i neaktivni materijal crne figurice

Iduće dvije funkcije *set_piece_black_active():* i *set_piece_black_inactive():* rade identičnu stvar kao i prethodne dvije funkcije, no u ovome slučaju za crne figurice.

```
func set_piece_move_eligible():
    $MeshInstance.set_material_override(piece_move_elig_MAT)

func set_piece_move_ineligible():
    $MeshInstance.set_material_override(piece_move_inelig_MAT)
```

Isječak kôda 4. Prikaz kôda za pravovaljani i nepravovaljani materijal figurice

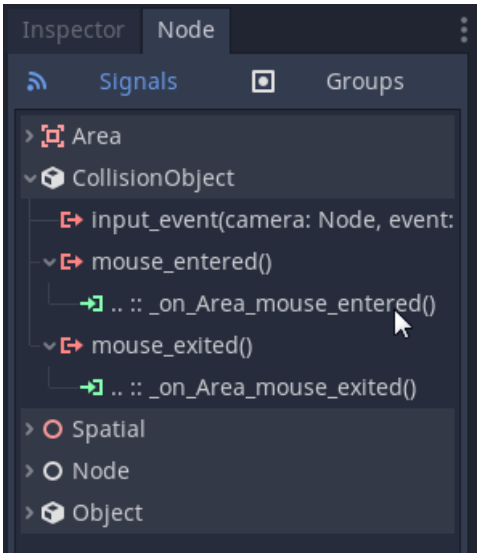
Sljedeće dvije funkcije *set_piece_move_eligible():* i *set_piece_move_ineligible():* mijenjaju materijal na figurici u zelenu boju ako je potez s figuricom moguć ili u crvenu boju ako potez nije moguć.

```
func _on_Area_mouse_entered():
    touchCounter += 1
    if touchCounter == 1:
        emit_signal("piece_pressed")
        play_select_piece_sound()
    elif touchCounter >= 2:
        touchCounter = 0
        emit_signal("piece_released")

func _on_Area_mouse_exited():
    emit_signal("piece_released")
    touchCounter = 0
    if currentColor == "white":
        set_piece_white_active()
    else:
        set_piece_black_active()
```

Isječak kôda 5. Prikaz funkcija koje će se pozivati na emitiranje signala figurice

Funkcijama *_on_Area_mouse_entered():* i *_on_Area_mouse_ exited():* definirano je što će se dogoditi ukoliko pokazivač miša uđe unutar volumena obujmice, odnosno ukoliko korisnik dotakne figuricu. Ove signale nije potrebno definirati na početku skripte jer su automatski postavljeni.



Slika 16. Prikaz uglavljenih signala unutar skripte za figuricu

Događaje koji se aktiviraju pomoću pokazivača miša Godot će automatski prilagoditi na događaje za dodir na mobilnim uređajima [5].

```
78 func play_move_piece_sound():
79     $SoundFXPlayer.set_stream(playable_sound_FX[randi()%3])
80     $SoundFXPlayer.play()
81
82 func play_remove_piece_sound():
83     $SoundFXPlayer.set_stream(remove_piece_sound)
84     $SoundFXPlayer.play()
85
86 func play_select_piece_sound():
87     $SoundFXPlayer.set_stream(select_piece_sound)
88     $SoundFXPlayer.play()
```

Isječak kôda 6. Prikaz funkcija za reprodukciju zvučnih efekata figurice

U iduće tri funkcije definirani su načini reprodukcije pojedinih zvukova prilikom interakcije s figuricama.

```
105 func change_mesh_simple():
106     $MeshInstance.set_mesh(mesh_simple)
107     $MeshInstance.set_scale(Vector3(0.85, 0.85, 0.85))
108     simple_mesh = true
109
110 func change_mesh_complex():
111     $MeshInstance.set_mesh(mesh_complex)
112     $MeshInstance.set_scale(Vector3(0.6, 0.6, 0.6))
113     simple_mesh = false
```

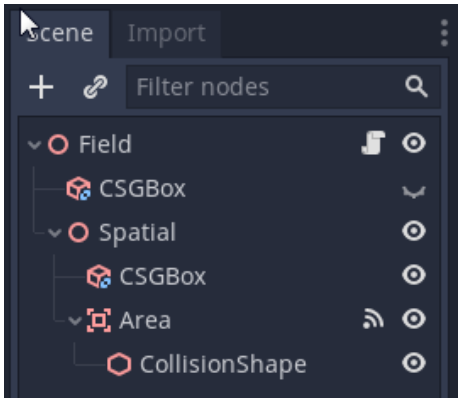
Isječak kôda 7. Prikaz funkcija za izmjenu dvije varijante modela

U zadnje dvije funkcije programske skripte definirane su naredbe za promjenu modela figurice iz prve u drugu varijantu.

Glavnoj figurici sada su dodijeljene sve potrebne funkcionalnosti te programski dio figurice je gotov.

3.1.2.2. Polja na ploči

Za glavno polje napravljena je zasebna scena koja je pohranjena s nazivom *Field_master.tscn*.



Slika 17. Prikaz sustava čvorova za glavno polje

Pomoću sustava čvorova, kao i prethodno kod izrade figurice, glavnom polju određene su sljedeće funkcionalnosti:

- Field (*3D Prostor*) – glavni čvor
  - *CSGKocka* – polje će imati svoj privremeni prikaz u 3D prostoru
  - *3D Prostor* – čvor koji je dodan radi lakšeg grupiranja ostalih čvorova
    - *CSGKocka* – polje će prikazivati buduću poziciju figurice

- *Površina* – polje će moći registrirati unos dodirom unutar svog volumena

Nadalje, za glavno polje dodana je nova skripta koja je pohranjena pod nazivom *Field.gd*. Skripta je uglavljena na čvor *Field* – taj čvor nalazi se najviše u hijerarhiji (eng. *root node*).

```
1  extends Spatial
2
3  signal field_pressed
4  signal field_released
5  signal field_wStart_pressed
6  signal field_bStart_pressed
7  signal field_battle_pressed
8  signal field_wFinish_pressed
9  signal field_bFinish_pressed
10 signal field_wExit_pressed
11 signal field_bExit_pressed
12
13 var isFieldTakenW = false
14 var isFieldTakenB = false
15
16 onready var fieldMoveValid = preload("res://Materials/field_eligible_MAT.tres")
17 onready var fieldMoveInvalid = preload("res://Materials/field_ineligible_MAT.tres")
18 onready var fieldDefaultMAT = preload("res://Materials/field_default_MAT.tres")
19 onready var fieldRosettaMAT = preload("res://Materials/field_rosetta_MAT.tres")
```

Isječak kôda 8. Prikaz definiranih varijabli za glavno polje

Na vrhu skripte nalazi se *extend Spatial* jer, kao i kod izrade figurice, pokušava se proširiti već predefinirana klasa. Definirano je 9 novih signala koji će se emitirati korisnikovim dodirom na zadano polje.

Nove dvije definirane varijable su: *isFieldTakenW* i *isFieldTakenB* – pomoću ovih dviju varijabli pratiti će se zauzetost pojedinog polja.

U iduće četiri varijable pohranjeni su materijali koji će se aplicirati na samo polje.

```
21 func _ready():
22     $Spatial.hide()
23     pass
```

Isječak kôda 9. Prikaz funkcije koja će se prva pozivati tijekom igre za glavno polje

U prvoj definiranoj funkciji za polje nalazi se funkcija *_ready():*. Ova funkcija zadana je funkcija unutar Godot programa. Njezina funkcionalnost je ta

da se uvijek pokreće i izvršava kada se pokrene cjelokupni program. Drugim riječima u ovu funkciju preporučljivo je stavljati one funkcije ili dijelove skripte za koje se želi da se izvrše prilikom prvog pokretanja same igre [5].

```
25 func field_move_valid():
26     $Spatial.show()
27     $Spatial/Area.show()
28     $Spatial/CSGBox.set_material(fieldMoveValid)
29
30 func field_move_invalid():
31     $Spatial.show()
32     $Spatial/Area.hide()
33     $Spatial/CSGBox.set_material(fieldMoveInvalid)
```

Isječak kôda 10. Prikaz funkcija za glavno polje kada je potez figuricom moguć, odnosno nije moguć

Iduće dvije funkcije u sebi sadrže naredbe za aktivaciju, odnosno deaktivaciju pojedinog polja.

```
35 func field_set_rosetta_mat():
36     $CSGBox.set_material(fieldRosettaMAT)
```

Isječak kôda 11. Prikaz funkcije za postavljanje plavog materijala na glavno polje

Treća funkcija *field_set_rosetta_mat():* postavlja plavi materijal na 3D model kocke – ova funkcija nema nikakvu svrhu unutar konačne igre već je korištena kao privremeni prikaz polja tijekom razvoja igre.

```
38 func field_hide():
39     $Spatial.hide()
```

Isječak kôda 12. Prikaz funkcije za isključivanje glavnog polja

Četvrtom funkcijom *field_hide():* definirana je naredba da se polje sakrije iz prikaza.

```
41 func _on_Area_mouse_entered():
42     field_hide()
43     emit_signal("field_pressed")
44     emit_signal("field_wStart_pressed")
45     emit_signal("field_bStart_pressed")
46     emit_signal("field_battle_pressed")
47     emit_signal("field_wFinish_pressed")
48     emit_signal("field_bFinish_pressed")
49     emit_signal("field_wExit_pressed")
50     emit_signal("field_bExit_pressed")
51
52 func _on_Area_mouse_exited():
53     field_hide()
54     emit_signal("field_released")
```

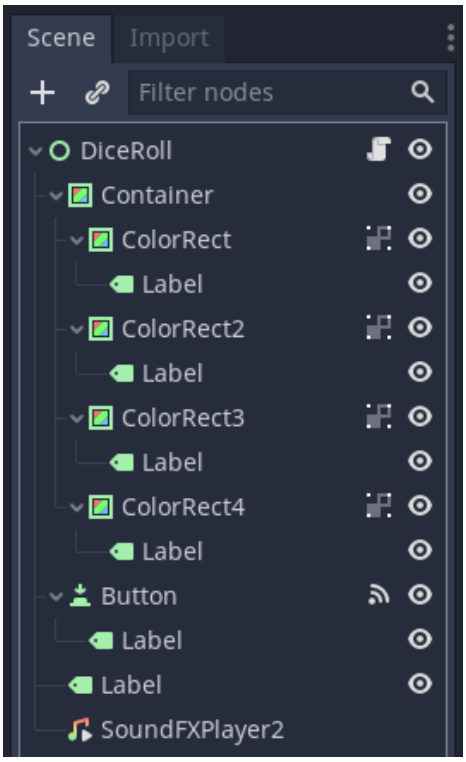
Isječak kôda 13. Prikaz funkcija koje će se emitirati pritiskom na polje

Peta funkcija *_on_Area_mouse_entered():* definira što će se događati ukoliko korisnik dotakne zadano polje.

Glavnom polju sada su dodijeljene sve potrebne funkcionalnosti te programski dio polja je gotov.

3.1.2.3. Bacanje kockica

Za dio igre u kojemu će se bacati kockice napravljena je nova scena i pohranjena je s imenom *Dice_roll.tscn*.



Slika 18. Prikaz sustava čvorova za bacanje kockica

Pomoću sustava čvorova sceni za bacanje kockica određene su iduće funkcionalnosti:

- *DiceRoll* – glavni čvor
  - *Spremnik* – spremnik za 4 čvora radi lakše organizacije
    - *Obojeni kvadrat*– prikaz dobivene vrijednosti za 1. kockicu
    - *Obojeni kvadrat 2* – prikaz dobivene vrijednosti za 2. kockicu
    - *Obojeni kvadrat 3* – prikaz dobivene vrijednosti za 3. kockicu
    - *Obojeni kvadrat 4* – prikaz dobivene vrijednosti za 4. kockicu
  - *Gumb* – gumb koji će pokretati bacanje kockica na korisnikov dodir, također prikazivati će se zbroj 4 dobivene vrijednosti kockica
  - *Oznaka* – ispisivati će se trenutni igrač na potezu tijekom igre

- *SoundFxPlayer2* – pritiskom na gumb reproducirati će se zvuk

Za scenu s bacanjem kockica dodana je skripta koja je pohranjena s nazivom *DiceRoll.gd*. Skripta je uglavljena na najviši čvor u hijerarhiji (*DiceRoll*).

```
1  extends Control
2
3  signal dice_roll_pressed(arg1)
4
5  onready var dice1 = get_node("Container/ColorRect/Label")
6  onready var dice2 = get_node("Container/ColorRect2/Label")
7  onready var dice3 = get_node("Container/ColorRect3/Label")
8  onready var dice4 = get_node("Container/ColorRect4/Label")
9  onready var dice_rolls = [dice1, dice2, dice3, dice4]
10 onready var dice_roll1 = preload("res://SoundFX/dice_rolled_default.wav")
11 onready var dice_roll2 = preload("res://SoundFX/dice_rolled_fast.wav")
12 onready var dice_roll3 = preload("res://SoundFX/dice_rolled_slow.wav")
13 onready var dice_soundFX = [dice_roll1, dice_roll2, dice_roll3]
```

Isječak kôda 14. Prikaz definiranih varijabli za scenu s bacanjem kockica

Pri vrhu skripte nalazi se *extends Control* jer nastoji se proširiti predifinirana klasa ovakve vrste s novim funkcionalnostima. Definiran je signal pod imenom *dice_roll_pressed(arg1)* i ovaj signal emitirati će se prilikom korisnikovog pritiska na gumb. Prve četiri varijable definirane su radi lakše referenciranja unutar kôda na polja s tekstom za 4 kockice. Peta varijabla definirana je kao polje radi lakšeg kasnijeg definiranja funkcije koja će polja s tekstom postavljati na vrijednost 0. Iduće tri varijable u sebi pohranjuju zvukove koji će se reproducirati prilikom pritiska na gumb za bacanje kockica. Zadnja varijabla u sebi pohranjuje ta tri zvuka unutar polja radi lakšeg kasnijeg nasumičnog odabira zvuka za reprodukciju.

```
15 func _ready():
16     set_dice_text_empty()
17     $".".hide()
18     disable_dice_roll()
```

Isječak kôda 15. Prikaz funkcije koja će se prva pozivati u skripti za bacanje kockica

Unutar *_ready():* funkcije definirano je da se onemogući funkcija bacanja kockica.

```gdscript
func set_dice_text_empty():
    for i in range(4):
        dice_rolls[i].set_text("0")
    $Label.set_text("Trenutno na potezu:\n\nIGRAC NIJE ODABRAN!")
```

Isječak kôda 16. Prikaz funkcije pomoću koje će se resetirati tekst s dobivenim vrijednostima kockica

Sljedeća funkcija *set_dice_text_empty():* u sebi kroz jednostavnu *for* petlju postavlja polja za kockice na vrijednost 0.

```gdscript
func roll_dice():
    var number_rolled = 0
    $Button/Label.set_text("Bacam\nkockice...")
    yield(get_tree().create_timer(.75), "timeout")
    for i in range(4):
        var one_dice_roll = randi()%2
        dice_rolls[i].set_text(str(one_dice_roll))
        number_rolled += one_dice_roll
    $Button/Label.set_text(str(number_rolled))
    emit_signal("dice_roll_pressed", number_rolled)
```

Isječak kôda 17. Prikaz funkcije za bacanje kockica

U funkciji *dice_roll():* definirane su naredbe pomoću kojih će se izvršavati bacanje četiriju kockica.

Nadalje u idućem retku funkcije izvršava se *for* petlja 4 puta za 4 kockice.

```gdscript
func enable_dice_roll():
    $Button.disabled = false
    $Button/Label.set_text("BACI\nKOCKICE")

func disable_dice_roll():
    $Button.disabled = true
```

Isječak kôda 18. Prikaz funkcija za uključivanje odnosno isključivanje gumba za bacanje kockica

Pomoću iduće dvije funkcije *enable_dice_roll()* i *disable_dice_roll()* uključiti će se, odnosno isključiti gumb za bacanje kockica.

```gdscript
func white_to_move():
    $Label.set_text("Trenutno na potezu:\n\nBIJELI IGRAC")

func black_to_move():
    $Label.set_text("Trenutno na potezu:\n\nCRNI IGRAC")
```

Isječak kôda 19. Prikaz funkcija za postavljanje teksta s trenutnim igračem na potezu

Funkcijama *white_to_move()* i *black_to_move()* definiran je tekst koji će se prikazati u prozoru s tekstom u donjem-desnom uglu ekrana zavisno o tome koji je trenutno igrač na potezu.

```
49 func _on_Button_pressed():
50     roll_dice()
51     play_dice_roll_sound()
52     disable_dice_roll()
```

Isječak kôda 20. Prikaz funkcije koja će se pozivati pritiskom na gumb

U predzadnjoj funkciji *_on_Button_pressed():* definirano je što će se dogoditi kada korisnik pritisne na gumb.

```
54 func play_dice_roll_sound():
55     $SoundFXPlayer2.volume_db = -7.5
56     $SoundFXPlayer2.set_stream(dice_soundFX[randi()%3])
57     $SoundFXPlayer2.play()
```

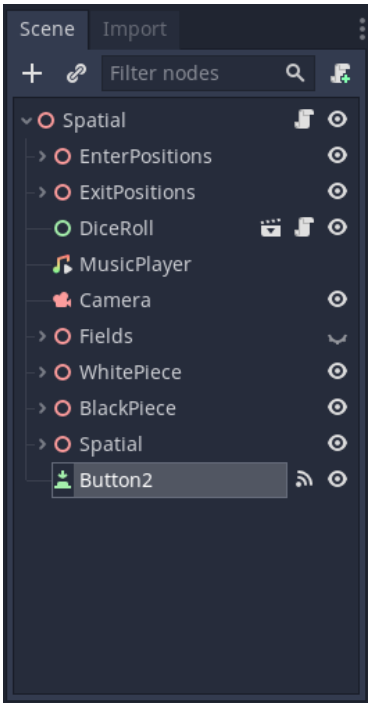
Isječak kôda 21. Prikaz funkcije za reprodukciju zvuka prilikom bacanja kockica

U zadnjoj funkciji *play_dice_roll_sound()* definiran je način na koji će se nasumično odabrati zvuk te zatim reproducirati.

Sceni za bacanje kockica sada su dodijeljene sve potrebne funkcionalnosti. Programski dio izrade bacanja kockica je gotov.

3.1.3. Programski kôd i postavke glavne scene

Za glavni dio igre, u kojemu će se instancirati do sada definirani elementi igre, napravljena je nova scena te pohranjena je s imenom *Game.tscn*.



Slika 19. Prikaz sustava čvorova za glavnu scenu igre

Za glavnu scenu pomoću sustava čvorova određene su sljedeće funkcionalnosti:

- *3D Prostor* – glavni čvor
  - *EnterPositions* – početne pozicije s kojih figurice ulaze na ploču
  - *ExitPositions* – završne pozicije s kojih figurice izlaze s ploče
  - *DiceRoll* – korisnik će moći izvoditi bacanje kockica
  - *MusicPlayer* – automatski će se reproducirati glazba u pozadini tijekom izvođenja igre
  - *Kamera* – korisnik će imati pregled 3D prostora tijekom izvođenja igre
  - *Fields* – korisnik će imati interakciju s pojedinim poljima na ploči
  - *WhitePiece* – korisnik će imati interakciju s bijelim figuricama
  - *BlackPiece* – korisnik će imati interakciju s crnim figuricama
  - *3D Prostor*– korisniku će biti prikazan model ploče u 3D prostoru
  - *Button2* – korisnik će moći mijenjati izgled figurica i ploče; također nakon gotove igre moći će pokrenuti novu igru

Za glavnu scenu igre dodana je skripta koja je uglavljena na najviši čvor u hijerarhiji (*3D Prostor*). Skripta je pohranjena s nazivom *Game.gd*.

```
1  extends Spatial
2
3  enum {WHITE_MOVE , BLACK_MOVE}
4  var state
5  var selectedPiece = 0
6  var playerPicked = false
7  var rolledNumber = 0
8  var bonusRoll = false
9  var white_pieces_playing = 6
10 var black_pieces_playing = 6
11 var materials_v2 = false
12 var piece_moved = false
13 var new_game = false
14
15 onready var board_simple = preload("res://Models/board_default_v3.obj")
16 onready var board_complex = preload("res://Models/ploca_druga_varijanta.obj")
17 onready var board_MAT1 = preload("res://Materials/board_red_MAT.tres")
18 onready var board_MAT2 = preload("res://Materials/board_regular_MAT.tres")
19 onready var board_MAT3 = preload("res://Materials/board_rosetta_MAT.tres")
20 onready var board_v2_MAT = preload("res://Materials/board_v2_MAT.tres")
21
22 onready var music1 = preload("res://Music/music1.ogg")
23 onready var music2 = preload("res://Music/music2.ogg")
24 onready var music3 = preload("res://Music/music3.ogg")
25 onready var music4 = preload("res://Music/music4.ogg")
26 onready var music5 = preload("res://Music/music5.ogg")
27 onready var player_wins = preload("res://SoundFX/player_wins_v2.wav")
28 onready var playableMusic = [music1, music2, music3, music4, music5]
```

Isječak kôda 22. Prikaz definiranih varijabli za glavnu scenu igre

Na početku skripte nalazi se *extends Spatial* jer proširuje se ova klasa novim funkcionalnostima. U prvoj varijabli definiran je enumerator. Ovakav tip varijable u Godotu služi za definiranje konstantnih (nepromjenjivih) vrijednosti. Sljedeća definirana varijabla je *selectedPiece* koja će u sebi privremeno pohraniti vrijednost *i*, odnosno informaciju o označenoj figurici. Iduća varijabla definirana je s nazivom *playerPicked* te pomoću ove varijable pratiti će se je li igrač koji započinje igru već odabran. Sljedeća definirana varijabla *rolledNumber* u sebi pohranjivati će dobiveni broj nakon bacanja kockica. Pomoću iduće varijable *bonusRoll* pratiti će se je li ostvaren uvjet za dodatno bacanje kockica. Putem iduće dvije varijable *white_pieces_playing* i *black_pieces_playing* pratiti će se preostali broj aktivnih figurica na ploči. Pomoću sljedeće varijable *materials_v2* pratiti će se koja varijanta modela

trenutno je postavljena na sceni. Nadalje iduća definirana varijabla *piece_moved* u sebi pohranjivati će uvjet ako je već odrađen potez figuricom. Naposljetku putem zadnje varijable *new_game* pratiti će se uvjet koji ukoliko je zadovoljen pokrenuti će se novi krug igre.

```
30 func _ready():
31     randomize()
32     rolledNumber = 0
33     initialize_fields()
34     yield(get_tree().create_timer(1.5), "timeout")
35     initialize_pieces()
36     initialize_music()
37     play_music()
38     yield(get_tree().create_timer(1), "timeout")
39     $DiceRoll.show()
40     yield(get_tree().create_timer(.5), "timeout")
41     initialize_dice()
42     print("\n= = = = = = = = = = = = = = = = = =")
43     print("Starting new game...")
```

Isječak kôda 23. Prikaz funkcije za glavnu scenu koja će se pozivati prva prilikom pokretanja igre

U prvoj funkciji *_ready():* vršiti će se radnje prilikom prvog pokretanja igre.

```
45 func initialize_dice():
46     $DiceRoll.connect("dice_roll_pressed", get_node("."), "_on_dice_roll_pressed")
47     show_dice_roll()
```

Isječak kôda 24. Prikaz funkcije za inicijalizaciju kockica

U funkciji *initialize_dice():* vršiti će se inicijalizacija scene bacanja kockica.

```
49 func initialize_music():
50     playableMusic.shuffle()
51     $MusicPlayer.set_stream(playableMusic.back())
```

Isječak kôda 25. Prikaz funkcije za inicijalizaciju pozadinske glazbe

Unutar funkcije *initialize_music():* vršiti će se inicijalizacija pozadinske glazbe.

```
53 func play_music():
54     $MusicPlayer.volume_db = -15
55     $MusicPlayer.play()
56     print("Currently playing:", playableMusic.back())
```

Isječak kôda 26. Prikaz funkcije za pokretanje reprodukcije glazbe

Funkcijom *play_music():* pokreće se reprodukcija pozadinske glazbe.

```
func show_dice_roll():
    $DiceRoll.enable_dice_roll()
```

Isječak kôda 27. Prikaz funkcije za aktiviranje bacanja kockica

Pomoću funkcije *show_dice_roll():* aktivira se funkcionalnost bacanje kockica u igri.

```
func pick_player():
    playerPicked = true
    var decide_player = randi()%2
    if decide_player == 0:
        state = WHITE_MOVE
        print("white player selected")
        $DiceRoll.white_to_move()
        white_piece_set_active()
    elif decide_player == 1:
        state = BLACK_MOVE
        print("black player selected")
        $DiceRoll.black_to_move()
        black_piece_set_active()
```

Isječak kôda 28. Prikaz funkcije za odabir igrača

Nadalje putem funkcije *pick_player():* izvršavati će se odabir igrača tijekom trajanja igre.

```
func update_field_status():
    for i in range(0, $Fields/WhiteStart.get_child_count()):
        $Fields/WhiteStart.get_child(i).isFieldTakenW = false
        $Fields/BlackStart.get_child(i).isFieldTakenB = false
    for i in range(0, $Fields/Battle.get_child_count()):
        $Fields/Battle.get_child(i).isFieldTakenW = false
        $Fields/Battle.get_child(i).isFieldTakenB = false
    for i in range(0, $Fields/WhiteFinish.get_child_count()):
        $Fields/WhiteFinish.get_child(i).isFieldTakenW = false
        $Fields/BlackFinish.get_child(i).isFieldTakenB = false
    for i in range(0, $WhitePiece.get_child_count()):
        var whiteCheck = $WhitePiece.get_child(i).currentPos
        if whiteCheck == -1:
            pass
        elif whiteCheck >= 0 and whiteCheck <= 3:
            $Fields/WhiteStart.get_child(whiteCheck).isFieldTakenW = true
        elif whiteCheck >= 4 and whiteCheck <= 11:
            $Fields/Battle.get_child(whiteCheck - 4).isFieldTakenW = true
        elif whiteCheck >= 12 and whiteCheck <= 13:
            $Fields/WhiteFinish.get_child(whiteCheck - 12).isFieldTakenW = true
        elif whiteCheck >= 14:
            pass
    for i in range(0, $BlackPiece.get_child_count()):
        var blackCheck = $BlackPiece.get_child(i).currentPos
        if blackCheck == -1:
            pass
        elif blackCheck >= 0 and blackCheck <= 3:
            $Fields/BlackStart.get_child(blackCheck).isFieldTakenB = true
        elif blackCheck >= 4 and blackCheck <= 11:
            $Fields/Battle.get_child(blackCheck - 4).isFieldTakenB = true
        elif blackCheck >= 12 and blackCheck <= 13:
            $Fields/BlackFinish.get_child(blackCheck - 12).isFieldTakenB = true
        elif blackCheck >= 14:
            pass
```

Isječak kôda 29. Prikaz funkcije za osvježavanje statusa polja

Pomoću funkcije *update_field_status():* raditi će se provjera zauzetosti pojedinih polja na ploči.

```
func white_piece_set_active():
    for i in range(0, $WhitePiece.get_child_count()):
        if $WhitePiece.get_child(i).isInGame == true:
            $WhitePiece.get_child(i).set_piece_white_active()
        else:
            pass

func white_piece_set_inactive():
    for i in range(0, $WhitePiece.get_child_count()):
        if $WhitePiece.get_child(i).isInGame == true:
            $WhitePiece.get_child(i).set_piece_white_inactive()
        else:
            pass
```

Isječak kôda 30. Prikaz funkcija za mijenjanje aktivnog statusa bijelih figurica

Iduća definirana funkcija *white_piece_set_active():* postavlja bijelu figuricu u aktivni status. Druga funkcija *white_piece_set_inactive():* postaviti će ju u neaktivni status.

```
124 func black_piece_set_active():
125     for i in range(0, $BlackPiece.get_child_count()):
126         if $BlackPiece.get_child(i).isInGame == true:
127             $BlackPiece.get_child(i).set_piece_black_active()
128         else:
129             pass
130
131 func black_piece_set_inactive():
132     for i in range(0, $BlackPiece.get_child_count()):
133         if $BlackPiece.get_child(i).isInGame == true:
134             $BlackPiece.get_child(i).set_piece_black_inactive()
135         else:
136             pass
```

Isječak kôda 31. Prikaz funkcija za mijenjanje aktivnog statusa crnih figurica

Nadalje iduće dvije funkcije *black_piece_set_active():* i *black_piece_set_inactive()* rade istu stvar kao i prijašnje dvije funkcije, ali za crne figurice.

```
138 func switch_player():
139     yield(get_tree().create_timer(.5), "timeout")
140     if state == WHITE_MOVE:
141         black_piece_set_active()
142         white_piece_set_inactive()
143         print("black player selected")
144         state = BLACK_MOVE
145         $DiceRoll.black_to_move()
146     elif state == BLACK_MOVE:
147         white_piece_set_active()
148         black_piece_set_inactive()
149         print("white player selected")
150         state = WHITE_MOVE
151         $DiceRoll.white_to_move()
```

Isječak kôda 32. Prikaz funkcije za promjenu igrača

Putem funkcije *switch_player():* mijenjati će se trenutni status u igri, odnosno mijenjati će se trenutni igrač na potezu.

```
func initialize_pieces():
    for i in range(0, $WhitePiece.get_child_count()):
        $WhitePiece.get_child(i).connect("piece_pressed", get_node("."), "_on_piece_pressed", [i])
        $WhitePiece.get_child(i).connect("piece_released", get_node("."), "_on_piece_released", [i])
        $BlackPiece.get_child(i).connect("piece_pressed", get_node("."), "_on_piece_pressed", [i])
        $BlackPiece.get_child(i).connect("piece_released", get_node("."), "_on_piece_released", [i])
    var random_piece_pick = randi()%2
    if random_piece_pick == 0:
        initialize_white_pieces()
        yield(get_tree().create_timer(.5), "timeout")
        initialize_black_pieces()
    elif random_piece_pick == 1:
        initialize_black_pieces()
        yield(get_tree().create_timer(.5), "timeout")
        initialize_white_pieces()
```

Isječak kôda 33. Prikaz funkcije za inicijalizaciju svih figurica

Iduća definirana funkcija je *initalize_pieces():*. Putem ove funkcije izvršiti će se inicijalizacija svih figurica.

```
func initialize_white_pieces():
    for i in range(0, $WhitePiece.get_child_count()):
        $WhitePiece.get_child(i).set_piece_white_inactive()
    for i in range(0, $WhitePiece.get_child_count()):
        $WhitePiece.get_child(i).animate_move_piece($WhitePiece.get_child(i), $EnterPositions/WhiteEnter.get_child(i))
        yield(get_tree().create_timer(rand_range(.1, .3)), "timeout")

func initialize_black_pieces():
    for i in range(0, $BlackPiece.get_child_count()):
        $BlackPiece.get_child(i).set_piece_black_inactive()
        $BlackPiece.get_child(i).currentColor = "black"
    for i in range(0, $BlackPiece.get_child_count()):
        $BlackPiece.get_child(i).animate_move_piece($BlackPiece.get_child(i), $EnterPositions/BlackEnter.get_child(i))
        yield(get_tree().create_timer(rand_range(.1, .3)), "timeout")
```

Isječak kôda 34. Prikaz funkcija za inicijalizaciju bijelih i crnih figurica

Funkcijom *initialize_white_pieces():* izvršavati će se inicijalizacija svih bijelih figurica.

```
func initialize_fields():
    for i in range(0, $Fields/WhiteStart.get_child_count()):
        $Fields/WhiteStart.get_child(i).connect("field_wStart_pressed", get_node("."), "_on_field_wStart_pressed", [i])
        $Fields/WhiteStart.get_child(i).connect("field_released", get_node("."), "_on_field_released", [i])
        $Fields/BlackStart.get_child(i).connect("field_bStart_pressed", get_node("."), "_on_field_bStart_pressed", [i])
        $Fields/BlackStart.get_child(i).connect("field_released", get_node("."), "_on_field_released", [i])
    for i in range(0, $Fields/Battle.get_child_count()):
        $Fields/Battle.get_child(i).connect("field_battle_pressed", get_node("."), "_on_field_battle_pressed", [i])
        $Fields/Battle.get_child(i).connect("field_released", get_node("."), "_on_field_released", [i])
    for i in range(0, $Fields/WhiteFinish.get_child_count()):
        $Fields/WhiteFinish.get_child(i).connect("field_wFinish_pressed", get_node("."), "_on_field_wFinish_pressed", [i])
        $Fields/WhiteFinish.get_child(i).connect("field_released", get_node("."), "_on_field_released", [i])
        $Fields/BlackFinish.get_child(i).connect("field_bFinish_pressed", get_node("."), "_on_field_bFinish_pressed", [i])
        $Fields/BlackFinish.get_child(i).connect("field_released", get_node("."), "_on_field_released", [i])
    $Fields/WhiteExit.get_child(0).connect("field_wExit_pressed", get_node("."), "_on_field_wExit_pressed")
    $Fields/BlackExit.get_child(0).connect("field_bExit_pressed", get_node("."), "_on_field_bExit_pressed")
```

Isječak kôda 35. Prikaz funkcije za inicijalizaciju polja

Putem funkcije *initialize_fields():* izvršavati će se inicijalizacija polja na ploči.

```
204 func _on_field_wStart_pressed(i):
205     white_piece_set_inactive()
206     piece_moved = true
207     yield(get_tree().create_timer(.5), "timeout")
208     var positionW = $WhitePiece.get_child(selectedPiece).currentPos
209     $WhitePiece.get_child(selectedPiece).animate_move_piece($WhitePiece.get_child(selectedPiece),
210         $Fields/WhiteStart.get_child(i))
211     yield(get_tree().create_timer(.5), "timeout")
212     $Fields/WhiteStart.get_child(i).isFieldTakenW = true
213     if i - rolledNumber >= 0:
214         $Fields/WhiteStart.get_child(i - rolledNumber).isFieldTakenW = false
215         $WhitePiece.get_child(selectedPiece).currentPos = i
216     else:
217         $WhitePiece.get_child(selectedPiece).currentPos = i
218     if i == 3:
219         bonusRoll = true
220     update_field_status()
221     yield(get_tree().create_timer(.5), "timeout")
222     $DiceRoll.enable_dice_roll()
```

Isječak kôda 36. Prikaz funkcije za prva četiri početna polja bijele figurice

Funkcija *_on_field_wStart_pressed(i):* je odaziv na signal koji je emitiran za jedno od prvih četiri početnih polja za bijelu figuricu.

```
224 func _on_field_bStart_pressed(i):
225     black_piece_set_inactive()
226     piece_moved = true
227     yield(get_tree().create_timer(.5), "timeout")
228     var positionB = $BlackPiece.get_child(selectedPiece).currentPos
229     $BlackPiece.get_child(selectedPiece).animate_move_piece($BlackPiece.get_child(selectedPiece),
230         $Fields/BlackStart.get_child(i))
231     yield(get_tree().create_timer(.5), "timeout")
232     $Fields/BlackStart.get_child(i).isFieldTakenB = true
233     if i - rolledNumber >= 0:
234         $Fields/BlackStart.get_child(i - rolledNumber).isFieldTakenB = false
235         $BlackPiece.get_child(selectedPiece).currentPos = i
236     else:
237         $BlackPiece.get_child(selectedPiece).currentPos = $Fields/BlackStart.get_child(i).get_index()
238 
239     if i == 3:
240         bonusRoll = true
241     yield(get_tree().create_timer(.5), "timeout")
242     $DiceRoll.enable_dice_roll()
```

Isječak kôda 37. Prikaz funkcije za prva četiri početna polja crne figurice

Iduća funkcija definira istu stvar, samo za crne figurice.

```
244 func _on_field_battle_pressed(i):
245     var backwardsField = i - rolledNumber
246     piece_moved = true
247     if state == WHITE_MOVE:
248         white_piece_set_inactive()
249         yield(get_tree().create_timer(.5), "timeout")
250         $WhitePiece.get_child(selectedPiece).animate_move_piece($WhitePiece.get_child(selectedPiece),
251             $Fields/Battle.get_child(i))
252         $Fields/Battle.get_child(i).isFieldTakenW = true
253         if $Fields/Battle.get_child(i).isFieldTakenB == true:
254             for j in range(6):
255                 if $BlackPiece.get_child(j).currentPos == (i + 4):
256                     $BlackPiece.get_child(j).animate_move_piece($BlackPiece.get_child(j),
257                         $EnterPositions/BlackEnter.get_child(j))
258                     $BlackPiece.get_child(j).currentPos = -1
259                     $Fields/Battle.get_child(i).isFieldTakenB = false
260                     $BlackPiece.get_child(j).play_remove_piece_sound()
261                 else:
262                     pass
263         else:
264             pass
265         if backwardsField < 0:
266             $Fields/WhiteStart.get_child(abs(backwardsField + 4)).isFieldTakenW = false
267         else:
268             $Fields/Battle.get_child(backwardsField).isFieldTakenW = false
269         $WhitePiece.get_child(selectedPiece).currentPos = i + 4
270         if i == 3:
271             bonusRoll = true
272         yield(get_tree().create_timer(.5), "timeout")
273         $DiceRoll.enable_dice_roll()
```

Isječak kôda 38. Prikaz funkcije za osam središnjih polja

```
275     elif state == BLACK_MOVE:
276         black_piece_set_inactive()
277         yield(get_tree().create_timer(.5), "timeout")
278         $BlackPiece.get_child(selectedPiece).animate_move_piece($BlackPiece.get_child(selectedPiece),
279             $Fields/Battle.get_child(i))
280         $Fields/Battle.get_child(i).isFieldTakenB = true
281         if $Fields/Battle.get_child(i).isFieldTakenW == true:
282             for j in range(6):
283                 if $WhitePiece.get_child(j).currentPos == (i + 4):
284                     $WhitePiece.get_child(j).animate_move_piece($WhitePiece.get_child(j),
285                         $EnterPositions/WhiteEnter.get_child(j))
286                     $WhitePiece.get_child(j).currentPos = -1
287                     $Fields/Battle.get_child(i).isFieldTakenW = false
288                     $WhitePiece.get_child(j).play_remove_piece_sound()
289                 else:
290                     pass
291         else:
292             pass
293         if backwardsField < 0:
294             $Fields/BlackStart.get_child(abs(backwardsField + 4)).isFieldTakenB = false
295         else:
296             $Fields/Battle.get_child(backwardsField).isFieldTakenB = false
297         $BlackPiece.get_child(selectedPiece).currentPos = i + 4
298         if i == 3:
299             bonusRoll = true
300         yield(get_tree().create_timer(.5), "timeout")
301         $DiceRoll.enable_dice_roll()
```

Isječak kôda 39. Prikaz drugog dijela funkcije za središnja polja

Funkcijom *_on_field_battle_pressed(i):* definiran je niz radnji koje će se izvršavati kada korisnik dotakne jedno od osam središnjih polja na ploči.

```
func _on_field_wFinish_pressed(i):
    white_piece_set_inactive()
    piece_moved = true
    var positionW = $WhitePiece.get_child(selectedPiece).currentPos
    yield(get_tree().create_timer(.5), "timeout")
    $WhitePiece.get_child(selectedPiece).animate_move_piece($WhitePiece.get_child(selectedPiece),
        $Fields/WhiteFinish.get_child(i))
    $Fields/WhiteFinish.get_child(i).isFieldTakenW = true
    if positionW < 12:
        $Fields/Battle.get_child(positionW - 4).isFieldTakenW = false
    else:
        $Fields/WhiteFinish.get_child(positionW - 4).isFieldTakenW = false
    $WhitePiece.get_child(selectedPiece).currentPos = i + 12
    if i == 1:
        bonusRoll = true
    yield(get_tree().create_timer(.5), "timeout")
    $DiceRoll.enable_dice_roll()
```

Isječak kôda 40. Prikaz funkcije za dva završna polja bijelih figurica

_on_field_wFinish_pressed(i): funkcija definira radnje koje će se izvršavati kada korisnik dotakne jedno od dva završna polja bijelih figurica.

```
func _on_field_bFinish_pressed(i):
    black_piece_set_inactive()
    piece_moved = true
    var positionB = $BlackPiece.get_child(selectedPiece).currentPos
    yield(get_tree().create_timer(.5), "timeout")
    $BlackPiece.get_child(selectedPiece).animate_move_piece($BlackPiece.get_child(selectedPiece),
        $Fields/BlackFinish.get_child(i))
    $Fields/BlackFinish.get_child(i).isFieldTakenB = true
    if positionB < 12:
        $Fields/Battle.get_child(positionB - 4).isFieldTakenB = false
    else:
        $Fields/BlackFinish.get_child(0).isFieldTakenB = false
    $BlackPiece.get_child(selectedPiece).currentPos = i + 12
    if i == 1:
        bonusRoll = true
    yield(get_tree().create_timer(.5), "timeout")
    $DiceRoll.enable_dice_roll()
```

Isječak kôda 41. Prikaz funkcije za dva završna polja crnih figurica

Funkcija *_on_field_bFinish_pressed(i):* definira iste radnje kao i prethodna funkcija, samo u ovome slučaju za crnu polja.

```gdscript
func _on_field_wExit_pressed():
	white_piece_set_inactive()
	piece_moved = true
	var positionW = $WhitePiece.get_child(selectedPiece).currentPos
	if positionW < 12:
		$Fields/Battle.get_child(positionW - 4).isFieldTakenW = false
	else:
		$Fields/WhiteFinish.get_child(positionW - 12).isFieldTakenW = false
	yield(get_tree().create_timer(.5), "timeout")
	$WhitePiece.get_child(selectedPiece).animate_move_piece($WhitePiece.get_child(selectedPiece),
		$ExitPositions/WhiteExit.get_child(selectedPiece))
	$WhitePiece.get_child(selectedPiece).currentPos = 14
	$WhitePiece.get_child(selectedPiece).isInGame = false
	white_pieces_playing -= 1
	if white_pieces_playing == 0:
		end_game()
	else:
		pass
	$DiceRoll.enable_dice_roll()
```

Isječak kôda 42. Prikaz funkcije za izlazno polje bijelih figurica

Funkcija *_on_field_wExit_pressed():* definira radnje koje će se poduzimati kada korisnik dotakne polje za izlaz bijele figurice s ploče.

```gdscript
func _on_field_bExit_pressed():
	black_piece_set_inactive()
	piece_moved = true
	var positionB = $BlackPiece.get_child(selectedPiece).currentPos
	if positionB < 12:
		$Fields/Battle.get_child(positionB - 4).isFieldTakenB = false
	else:
		$Fields/BlackFinish.get_child(positionB - 12).isFieldTakenB = false
	yield(get_tree().create_timer(.5), "timeout")
	$BlackPiece.get_child(selectedPiece).animate_move_piece($BlackPiece.get_child(selectedPiece),
		$ExitPositions/BlackExit.get_child(selectedPiece))
	$BlackPiece.get_child(selectedPiece).currentPos = 14
	$BlackPiece.get_child(selectedPiece).isInGame = false
	black_pieces_playing -= 1
	if black_pieces_playing == 0:
		end_game()
	else:
		pass
	$DiceRoll.enable_dice_roll()
```

Isječak kôda 43. Prikaz funkcije za izlazno polje crnih figurica

Funkcija *_on_field_bExit_pressed():* raditi će sve identično kao i prethodna funkcija, samo u ovom slučaju za crne figurice.

```
func _on_piece_pressed(i):
    selectedPiece = i
    if state == WHITE_MOVE:
        var futurePositionW = $WhitePiece.get_child(i).currentPos + rolledNumber
        if futurePositionW >= 0 and futurePositionW <= 3:
            if $Fields/WhiteStart.get_child(futurePositionW).isFieldTakenW == false:
                $Fields/WhiteStart.get_child(futurePositionW).field_move_valid()
                $WhitePiece.get_child(i).set_piece_move_eligible()
            else:
                $Fields/WhiteStart.get_child(futurePositionW).field_move_invalid()
                $WhitePiece.get_child(i).set_piece_move_ineligible()
        elif futurePositionW >= 4 and futurePositionW <= 11:
            if ($Fields/Battle.get_child(futurePositionW - 4).get_index() == 3 and
                $Fields/Battle.get_child(futurePositionW - 4).isFieldTakenW == true):
                $Fields/Battle.get_child(futurePositionW - 4).field_move_invalid()
                $WhitePiece.get_child(i).set_piece_move_ineligible()
            elif ($Fields/Battle.get_child(futurePositionW - 4).get_index() == 3 and
                $Fields/Battle.get_child(futurePositionW - 4).isFieldTakenB == true):
                $Fields/Battle.get_child(futurePositionW - 4).field_move_invalid()
                $WhitePiece.get_child(i).set_piece_move_ineligible()
            elif ($Fields/Battle.get_child(futurePositionW - 4).isFieldTakenW == false or
                $Fields/Battle.get_child(futurePositionW - 4).isFieldTakenB == true):
                $Fields/Battle.get_child(futurePositionW - 4).field_move_valid()
                $WhitePiece.get_child(i).set_piece_move_eligible()
            else:
                $Fields/Battle.get_child(futurePositionW - 4).field_move_invalid()
                $WhitePiece.get_child(i).set_piece_move_ineligible()
        elif futurePositionW > 11 and futurePositionW < 14:
            if $Fields/WhiteFinish.get_child(futurePositionW - 12).isFieldTakenW == false:
                $Fields/WhiteFinish.get_child(futurePositionW - 12).field_move_valid()
                $WhitePiece.get_child(i).set_piece_move_eligible()
            else:
                $Fields/WhiteFinish.get_child(futurePositionW - 12).field_move_invalid()
                $WhitePiece.get_child(i).set_piece_move_ineligible()
        elif futurePositionW == 14:
            $Fields/WhiteExit.get_child(0).field_move_valid()
            $WhitePiece.get_child(i).set_piece_move_eligible()
        elif futurePositionW > 14:
            $Fields/WhiteExit.get_child(0).field_move_invalid()
            $WhitePiece.get_child(i).set_piece_move_ineligible()
```

Isječak kôda 44. Prikaz funkcije koja će se pokretati prilikom dodira figurice

```
elif state == BLACK_MOVE:
    var futurePositionB = $BlackPiece.get_child(i).currentPos + rolledNumber
    if futurePositionB >= 0 and futurePositionB < 4:
        if $Fields/BlackStart.get_child(futurePositionB).isFieldTakenB == false:
            $Fields/BlackStart.get_child(futurePositionB).field_move_valid()
            $BlackPiece.get_child(i).set_piece_move_eligible()
        else:
            $Fields/BlackStart.get_child(futurePositionB).field_move_invalid()
            $BlackPiece.get_child(i).set_piece_move_ineligible()
    elif futurePositionB > 3 and futurePositionB < 12:
        if ($Fields/Battle.get_child(futurePositionB - 4).get_index() == 3 and
            $Fields/Battle.get_child(futurePositionB - 4).isFieldTakenW == true):
            $Fields/Battle.get_child(futurePositionB - 4).field_move_invalid()
            $BlackPiece.get_child(i).set_piece_move_ineligible()
        elif ($Fields/Battle.get_child(futurePositionB - 4).get_index() == 3 and
            $Fields/Battle.get_child(futurePositionB - 4).isFieldTakenB == true):
            $Fields/Battle.get_child(futurePositionB - 4).field_move_invalid()
            $BlackPiece.get_child(i).set_piece_move_ineligible()
        elif ($Fields/Battle.get_child(futurePositionB - 4).isFieldTakenB == false or
            $Fields/Battle.get_child(futurePositionB - 4).isFieldTakenW == true):
            $Fields/Battle.get_child(futurePositionB - 4).field_move_valid()
            $BlackPiece.get_child(i).set_piece_move_eligible()
        else:
            $Fields/Battle.get_child(futurePositionB - 4).field_move_invalid()
            $BlackPiece.get_child(i).set_piece_move_ineligible()
    elif futurePositionB > 11 and futurePositionB < 14:
        if $Fields/BlackFinish.get_child(futurePositionB - 12).isFieldTakenB == false:
            $Fields/BlackFinish.get_child(futurePositionB - 12).field_move_valid()
            $BlackPiece.get_child(i).set_piece_move_eligible()
        else:
            $Fields/BlackFinish.get_child(futurePositionB - 12).field_move_invalid()
            $BlackPiece.get_child(i).set_piece_move_ineligible()
    elif futurePositionB == 14:
        $Fields/BlackExit.get_child(0).field_move_valid()
        $BlackPiece.get_child(i).set_piece_move_eligible()
    elif futurePositionB > 14:
        $Fields/BlackExit.get_child(0).field_move_invalid()
        $BlackPiece.get_child(i).set_piece_move_ineligible()
```

Isječak kôda 45. Prikaz drugog dijela funkcije za dodir figurice

Putem funkcije *_on_piece_pressed(i):* biti će definiran niz radnji koje će se izvršavati kada zadana figurica bude bila označena.

```
func _on_piece_released(i):
    if state == WHITE_MOVE:
        var futurePosition = $WhitePiece.get_child(i).currentPos + rolledNumber
        if futurePosition >= 0 and futurePosition < 4:
            $Fields/WhiteStart.get_child(futurePosition).field_hide()
        elif futurePosition > 3 and futurePosition < 12:
            $Fields/Battle.get_child(futurePosition - 4).field_hide()
        elif futurePosition > 11 and futurePosition < 14:
            $Fields/WhiteFinish.get_child(futurePosition - 12).field_hide()
        elif futurePosition == 14:
            $Fields/WhiteExit.get_child(0).field_hide()
        elif futurePosition > 14:
            $Fields/WhiteExit.get_child(0).field_hide()
    elif state == BLACK_MOVE:
        var futurePosition = $BlackPiece.get_child(i).currentPos + rolledNumber
        if futurePosition >= 0 and futurePosition < 4:
            $Fields/BlackStart.get_child(futurePosition).field_hide()
        elif futurePosition > 3 and futurePosition < 12:
            $Fields/Battle.get_child(futurePosition - 4).field_hide()
        elif futurePosition > 11 and futurePosition < 14:
            $Fields/BlackFinish.get_child(futurePosition - 12).field_hide()
        elif futurePosition == 14:
            $Fields/BlackExit.get_child(0).field_hide()
        elif futurePosition > 14:
            $Fields/BlackExit.get_child(0).field_hide()
```

Isječak kôda 46. Prikaz funkcije za emitirani signal *piece_released*

Putem funkcije *_on_piece_released(i):* biti će definiran niz radnji koje će se izvršavati kada zadana figurica više ne bude bila označena.

```
func debug_field_status():
    print("\n")
    print("= = = = = = = = = = = = = = = =")
    print("White Start Fields Status:")
    for i in range(0, $Fields/WhiteStart.get_child_count()):
        print("White Start ", i, ": ", $Fields/WhiteStart.get_child(i).isFieldTakenW)
    print("Black Start Fields Status:")
    for i in range(0, $Fields/BlackStart.get_child_count()):
        print("Black Start ", i, ": ", $Fields/BlackStart.get_child(i).isFieldTakenB)
    print("Battle Fields Status:")
    for i in range(0, $Fields/Battle.get_child_count()):
        print("Battle White ", i, ": ", $Fields/Battle.get_child(i).isFieldTakenW)
        print("Battle Black ", i, ": ", $Fields/Battle.get_child(i).isFieldTakenB)
    print("White Finish Fields Status:")
    for i in range(0, $Fields/WhiteFinish.get_child_count()):
        print("White Finish ", i, ": ", $Fields/WhiteFinish.get_child(i).isFieldTakenW)
    print("Black Finish Fields Status:")
    for i in range(0, $Fields/BlackFinish.get_child_count()):
        print("Black Finish", i, ": ", $Fields/BlackFinish.get_child(i).isFieldTakenB)
    print("\n")
```

Isječak kôda 47. Prikaz funkcije za ispis statusa polja u konzolu

Funkcijom *debug_field_status():* ispisivati će se status svakog polja unutar konzole u Godotu.

```gdscript
func debug_piece_status():
    print("\n")
    print("White Piece Status:")
    for i in range(0, $WhitePiece.get_child_count()):
        print("White Piece Current Position :", $WhitePiece.get_child(i).currentPos)
    print("Black Piece Status")
    for i in range(0, $BlackPiece.get_child_count()):
        print("Black Piece Current Position :", $BlackPiece.get_child(i).currentPos)
    print("\n")
```

Isječak kôda 48. Prikaz funkcije za ispis statusa figurica u konzolu

Pomoću funkcije *debug_piece_status():* unutar konzole Godot programa ispisivati će se pozicije svake figurice.

```gdscript
func _on_dice_roll_pressed(number_rolled):
    yield(get_tree().create_timer(.5), "timeout")
    piece_moved = false
    rolledNumber = number_rolled
    if bonusRoll == true:
        bonusRoll = false
        if rolledNumber == 0:
            print("Bonus Roll! Rolled number is zero")
            $DiceRoll/Label.set_text("Dobiveni broj je nula!")
            yield(get_tree().create_timer(2), "timeout")
            $DiceRoll/Label.set_text("")
            if playerPicked == false:
                print("Bonus Roll! Rolled number was zero and player wasn't picked yet")
                $DiceRoll.enable_dice_roll()
            elif playerPicked == true:
                print("Bonus Roll! Rolled number was zero and player was already picked")
                $DiceRoll.enable_dice_roll()
        elif rolledNumber > 0:
            print("Bonus Roll! Rolled number was greater than zero")
            if state == WHITE_MOVE:
                print("Bonus Roll! Rolled number was greater than zero and it is white's turn")
                white_piece_set_active()
            elif state == BLACK_MOVE:
                print("Bonus Roll! Rolled number was greater than zero and it is black's turn")
                black_piece_set_active()
    else:
        if rolledNumber == 0:
            print("Rolled number is zero")
            $DiceRoll/Label.set_text("Dobiveni broj je 0!")
            yield(get_tree().create_timer(3), "timeout")
            $DiceRoll/Label.set_text("")
            if playerPicked == false:
                print("Rolled number was zero and player hasn't been picked yet")
                $DiceRoll.enable_dice_roll()
            elif playerPicked == true:
                print("Rolled number was zero and player is already picked")
                $DiceRoll.enable_dice_roll()
        elif rolledNumber > 0:
            print("Rolled number is greater than zero")
            if playerPicked == false:
                print("Rolled number is greater than zero and player hasn't been picked yet")
                pick_player()
            elif playerPicked == true:
                print("Rolled number is greater than zero and need to switch players")
                switch_player()
```

Isječak kôda 49. Prikaz funkcije za bacanje kockica

Funkcijom *_on_dice_roll_pressed(number_rolled):* provjeravati će se nizovi uvjeta nakon svakog bacanja kockica.

```
779 func start_new_game():
780     white_pieces_playing = 6
781     black_pieces_playing = 6
782     playerPicked = false
783     rolledNumber = 0
784     bonusRoll = false
785     initialize_music()
786     play_music()
787     $Button2.set_modulate(Color(1, 1, 1, 0.35))
788     $DiceRoll.enable_dice_roll()
789     for i in range(4):
790         $Fields/WhiteStart.get_child(i).isFieldTakenW = false
791         $Fields/BlackStart.get_child(i).isFieldTakenB = false
792     for i in range(8):
793         $Fields/Battle.get_child(i).isFieldTakenW = false
794         $Fields/Battle.get_child(i).isFieldTakenB = false
795     for i in range(2):
796         $Fields/WhiteFinish.get_child(i).isFieldTakenW = false
797         $Fields/BlackFinish.get_child(i).isFieldTakenB = false
798     for i in range(6):
799         $WhitePiece.get_child(i).currentPos = -1
800         $WhitePiece.get_child(i).isInGame = true
801         $WhitePiece.get_child(i).translate(Vector3(-14, 0, 0))
802         $BlackPiece.get_child(i).currentPos = -1
803         $BlackPiece.get_child(i).isInGame = true
804         $BlackPiece.get_child(i).translate(Vector3(-14, 0, 0))
805     yield(get_tree().create_timer(1), "timeout")
806     var random_piece_pick = randi()%2
807     if random_piece_pick == 0:
808         initialize_white_pieces()
809         yield(get_tree().create_timer(.5), "timeout")
810         initialize_black_pieces()
811     elif random_piece_pick == 1:
812         initialize_black_pieces()
813         yield(get_tree().create_timer(.5), "timeout")
814         initialize_white_pieces()
```

Isječak kôda 50. Prikaz funkcije za pokretanje nove igre

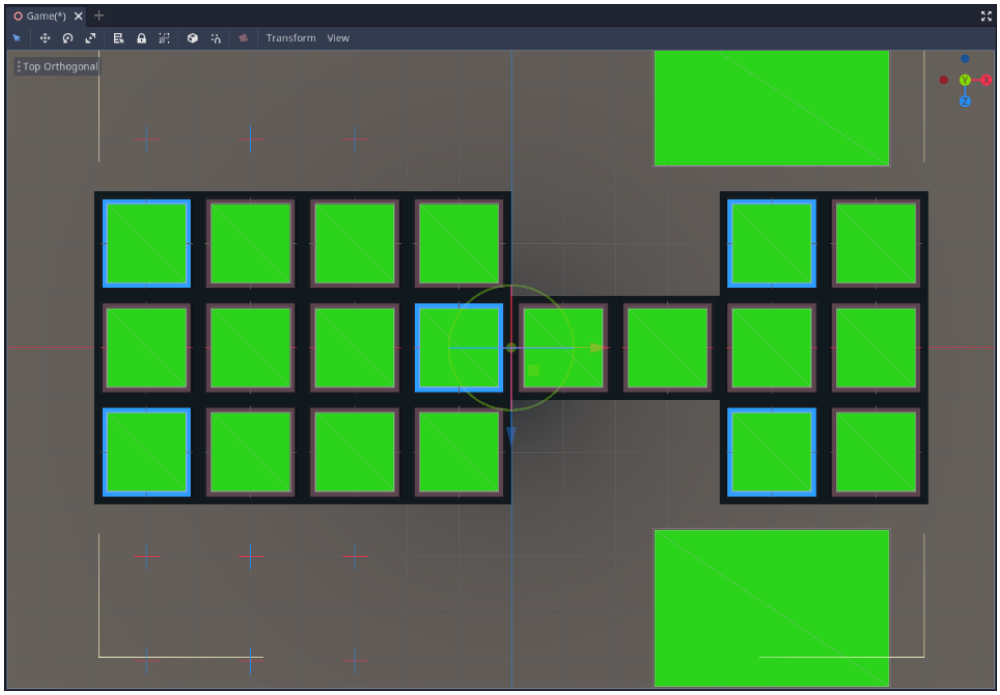
Pomoću funkcije *start_new_game():* definiran je niz radnji koje će se izvršavati prilikom pokretanja novog kruga igre.

```
816 func end_game():
817     white_piece_set_inactive()
818     black_piece_set_inactive()
819     $DiceRoll.disable_dice_roll()
820     $MusicPlayer.stop()
821     $MusicPlayer.set_stream(player_wins)
822     $MusicPlayer.volume_db = -7
823     $MusicPlayer.play()
824     new_game = true
825     if white_pieces_playing == 0:
826         $DiceRoll/Button/Label.set_text("Bijeli je\npobjednik!")
827     elif black_pieces_playing == 0:
828         $DiceRoll/Button/Label.set_text("Crni je\npobjednik!")
829     yield(get_tree().create_timer(3), "timeout")
830     $Button2.set_text("Nova igra?")
831     $Button2.set_modulate(Color(1, 1, 1, 1))
```

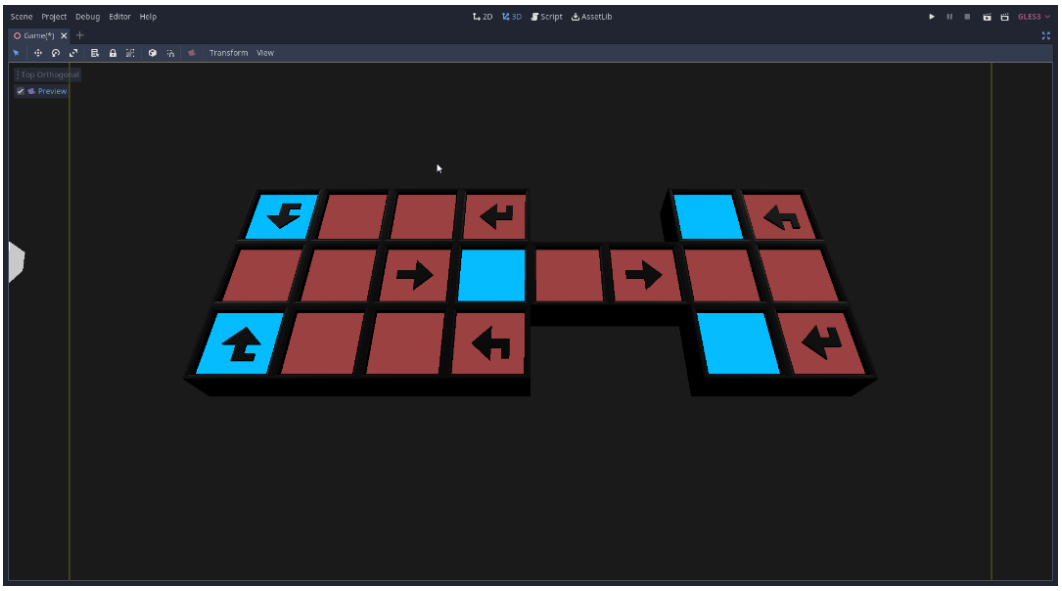
Isječak kôda 51. Prikaz funkcije koja će se pokretati na kraju kruga igre

U zadnjoj funkciji *end_game():* definirane su radnje koje će biti izvršene kada jedan krug igre bude gotov.

S ovom zadnjom funkcijom postavljene su sve funkcije i omogućene su sve funkcionalnosti unutar igre. Iduće potrebno je postaviti pozicije čvorova na sceni te odrediti glavne postavke za aplikaciju.

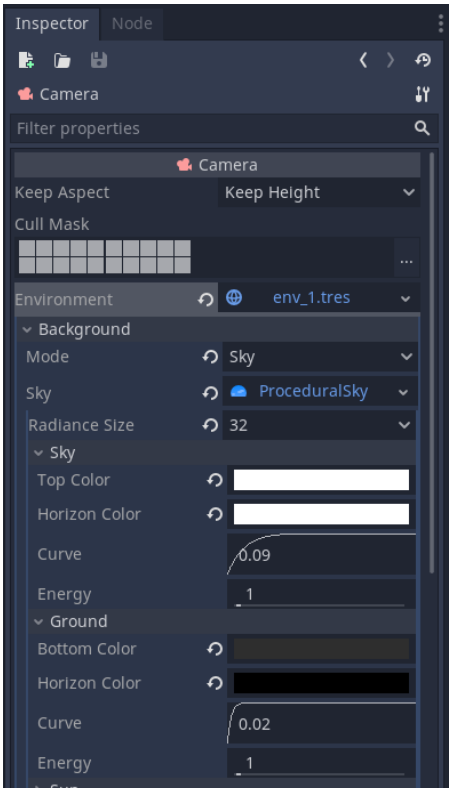


Slika 20. Prikaz postavljenih čvorova na glavnoj sceni

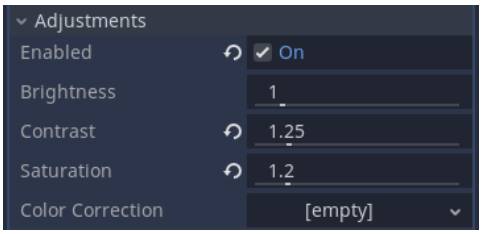


Slika 21. Prikaz pogleda koji će imati kamera tijekom igre

Nadalje određene su postavke okoline za kameru (eng. *Environment*).

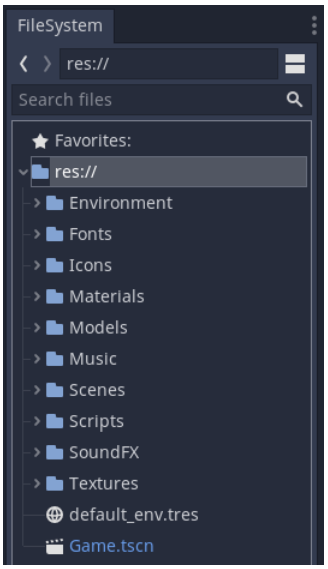


Slika 22. Prikaz izbornika s postavkama za okolinu



Slika 23. Prikaz izbornika s opcijama za svjetlinu, kontrast i zasićenost boje na sceni

U sustavu s datotekama (eng. *FileSystem*) izrađene su mape te pridijeljeni su odgovarajući nazivi ovisno o vrsti datoteke koju će pojedina mapa pohranjivati. Izgled organiziranih mapa može se vidjeti na slici. Ovime je završena izrada programskog dijela za glavnu scenu igre.

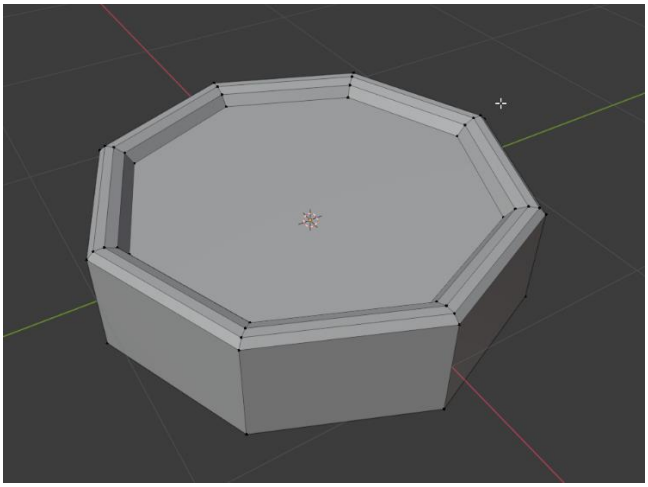


Slika 24. Prikaz organiziranih mapa u sustavu datoteka

3.2. 3D modeli

3.2.1. Prva varijanta 3D modela

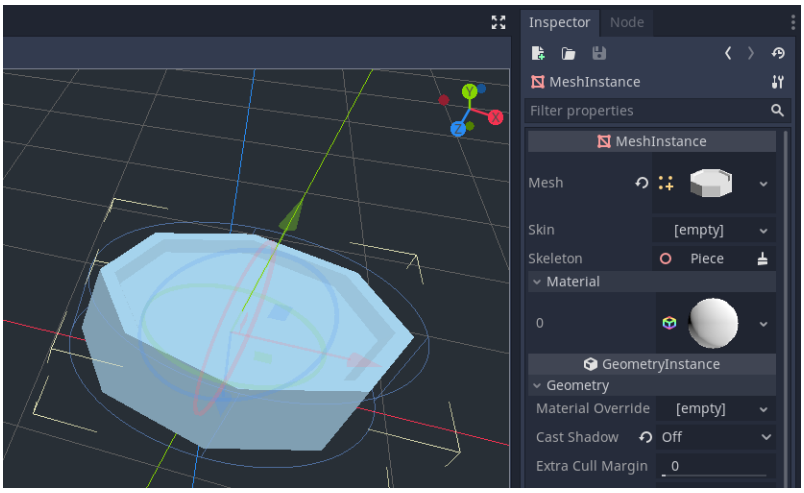
Prva varijanta 3D modela zamišljena je na način da izrađeni modeli budu sa što manjim brojem poligona te sa što jednostavnijim materijalima (po mogućnosti jedna boja, bez tekstura). Nastojalo se izraditi modele sa što jednostavnijom topologijom kako prilikom igranja igre korisnikov uređaj ne bi bio dodatno opterećen uslijed prikazivanja složenih 3D modela.



Slika 25. Prikaz zaglađenih gornjih rubova na figurici

Izrađen je model te dodijeljen mu je jedan materijal unutar Blendera. Nadalje model je izvezen iz programa putem *.obj*[15] formata. Nakon toga model je uvežen u Godot program.

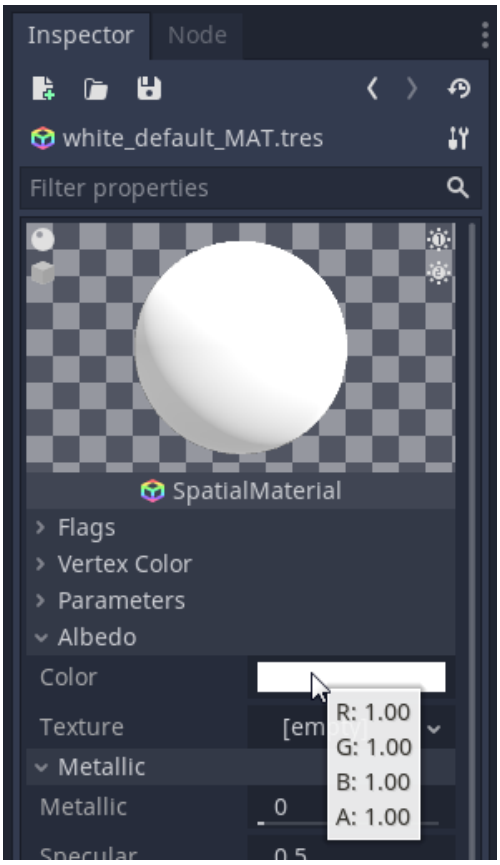
Unutar programa prepoznato je da uveženi model sadrži materijal na sebi. No dodatnu datoteku s postavkama materijala, koja je izvežena uz *.obj* format, Godot program nije uspio prepoznati.



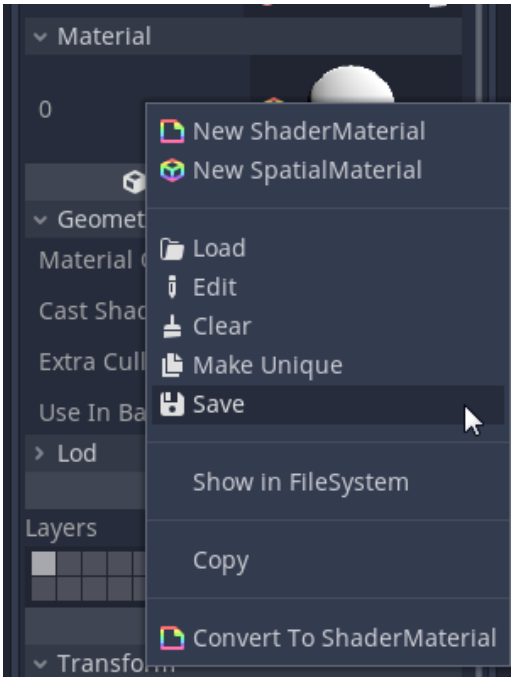
Slika 26. Prikaz prve varijante modela u Godotu

Nakon što je model unešen u Godot otvorena je scena za glavnu figuricu te izrađen je materijal.

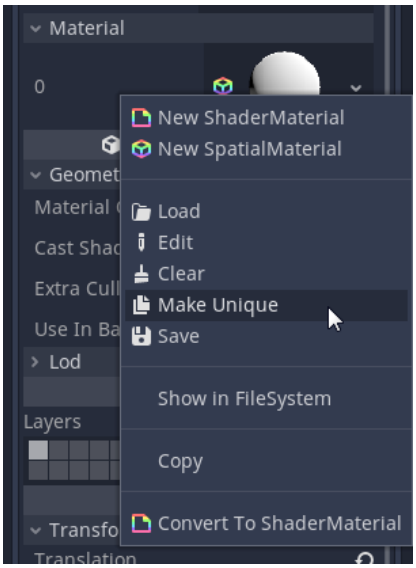
[15] *.obj* – sufiks za datoteku 3D modela



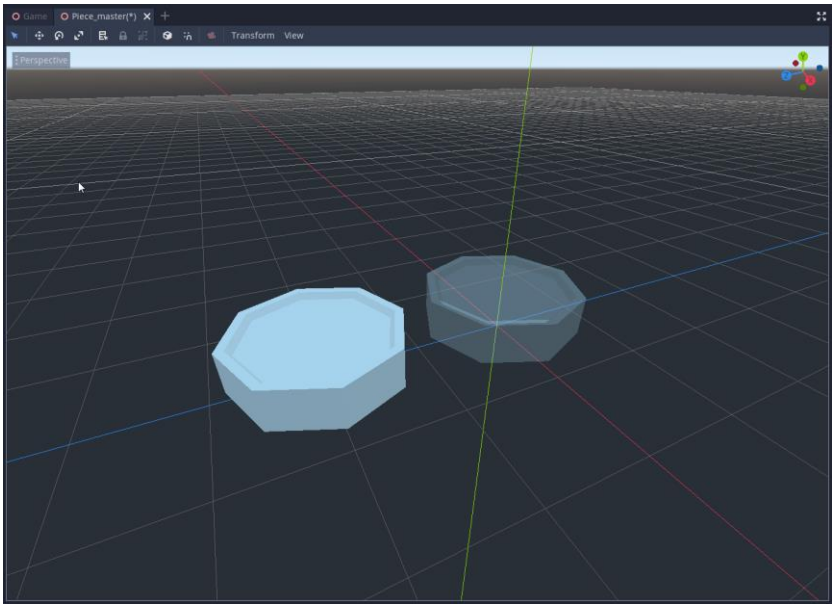
Slika 27. Prikaz postavki boje za bijelu figuricu



Slika 28. Prikaz izbornika i naredbe za pohranjivanje materijala u Godotu

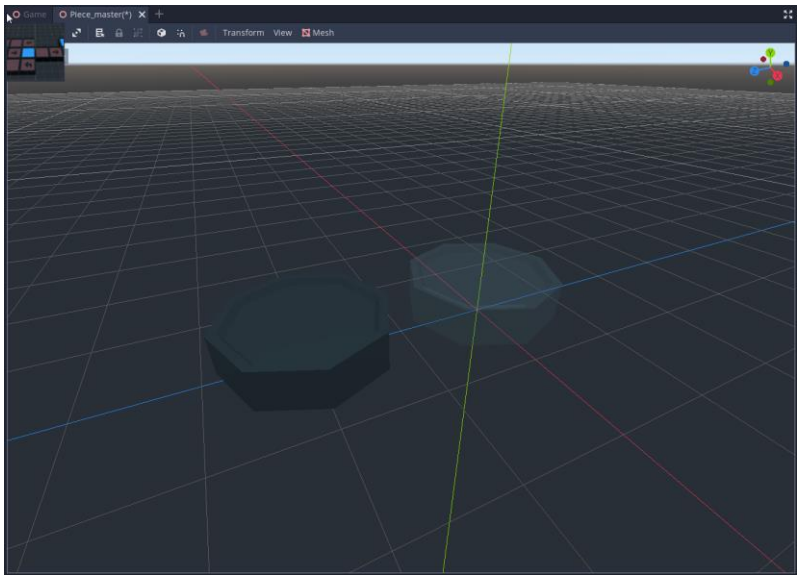


Slika 29. Prikaz izbornika i naredbe za dupliciranje trenutno postavljenog materijala u Godotu



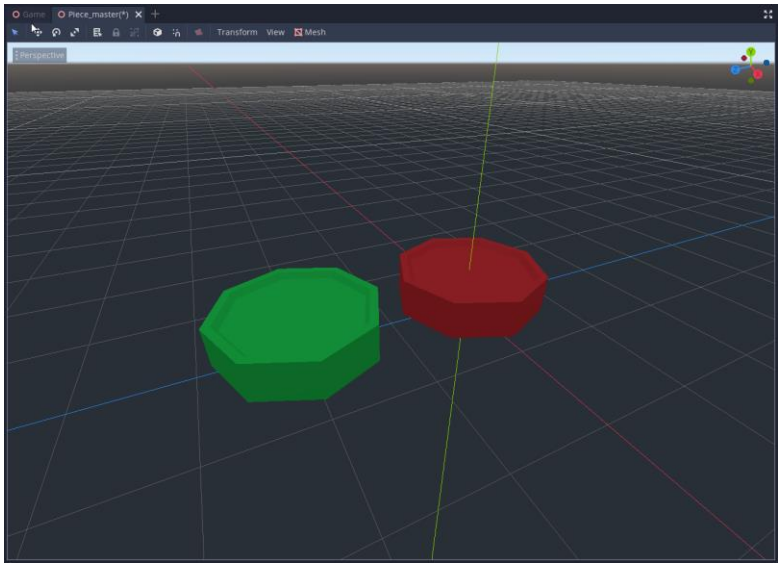
Slika 30. Prikaz materijala za bijelu figuricu

Ostala dva materijala za crnu figuricu napravljena su na potpuno identičan način.



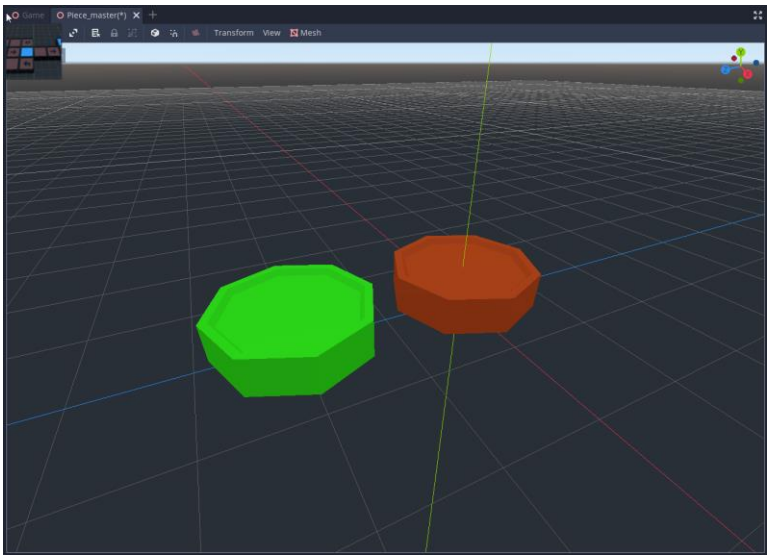
Slika 31. Prikaz materijala za crnu figuricu

Nadalje potrebno je postaviti materijale za figuricu kada će potez figuricom biti pravovaljan, odnosno nepravovaljan.



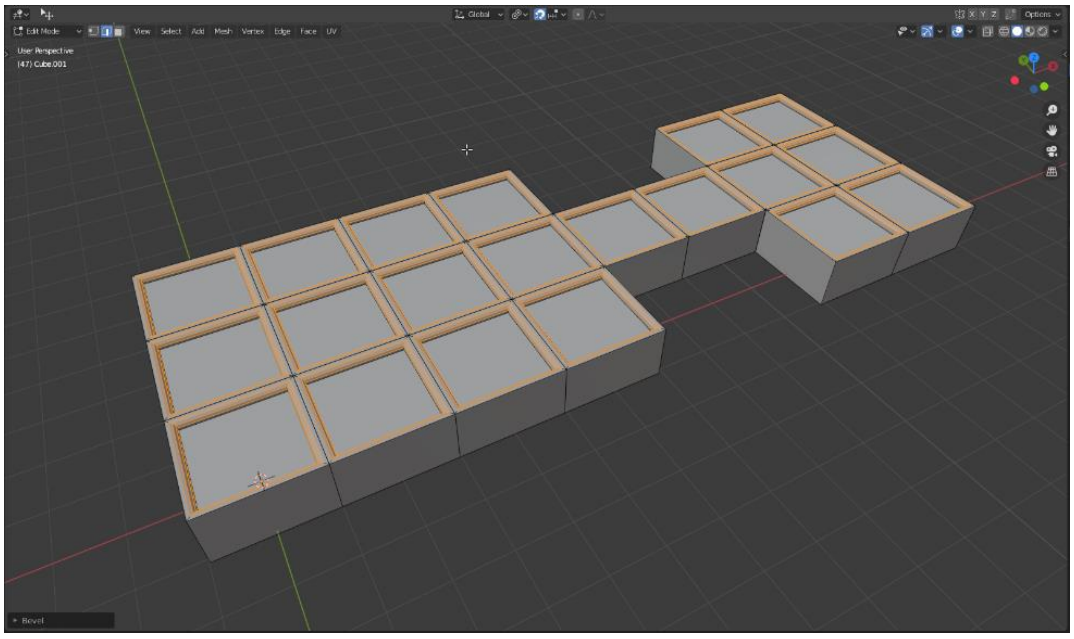
Slika 32. Prikaz materijala za mogući i nemogući potez figuricom

Prije nego što se krene s izradom ostalih 3D modela, potrebno je izraditi materijale za polja na ploči. Napravljena su dva nova materijala za polja.

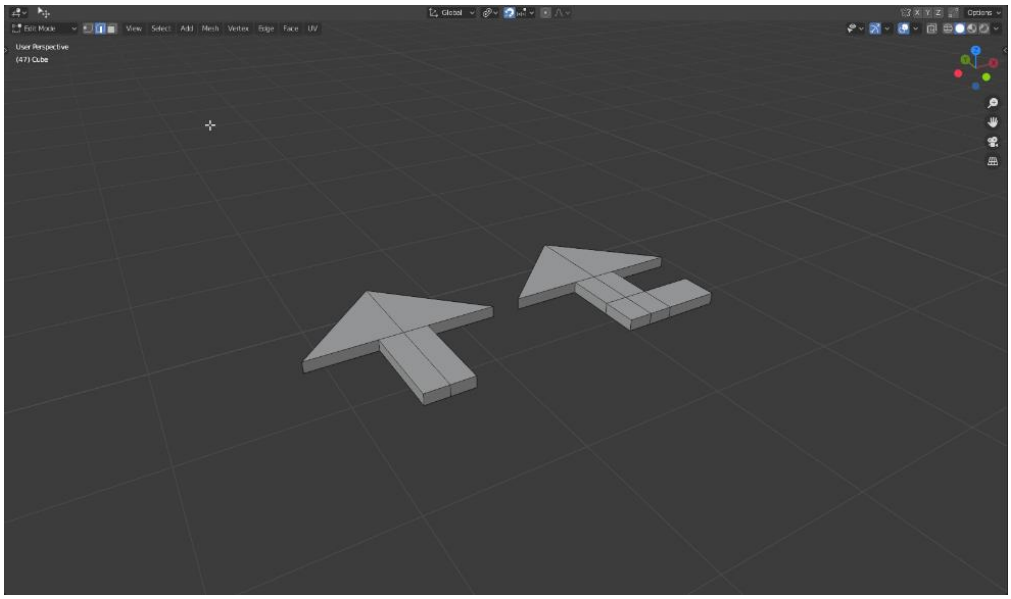


Slika 33. Prikaz materijala za mogući i nemogući potez na polju

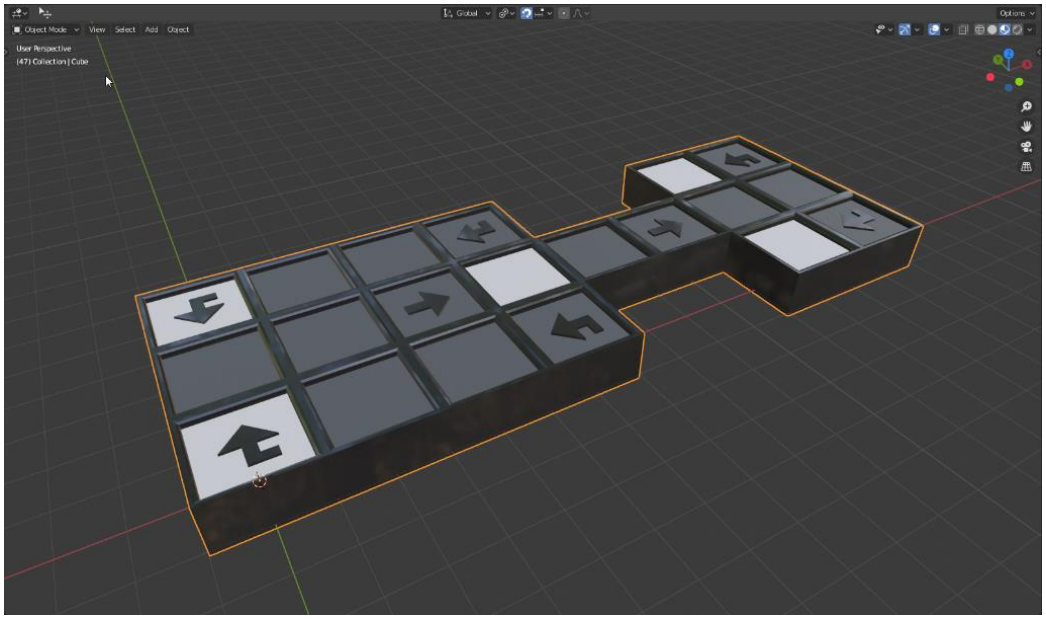
Sada je 3D model figurice za prvu varijantu gotov te napravljeni su materijali za figurice, ali i za polja. Dalje potrebno je izraditi 3D model ploče za prvu varijantu po kojoj će se figurice pomicati tijekom igre.



Slika 34. Prikaz modela ploče za prvu varijantu

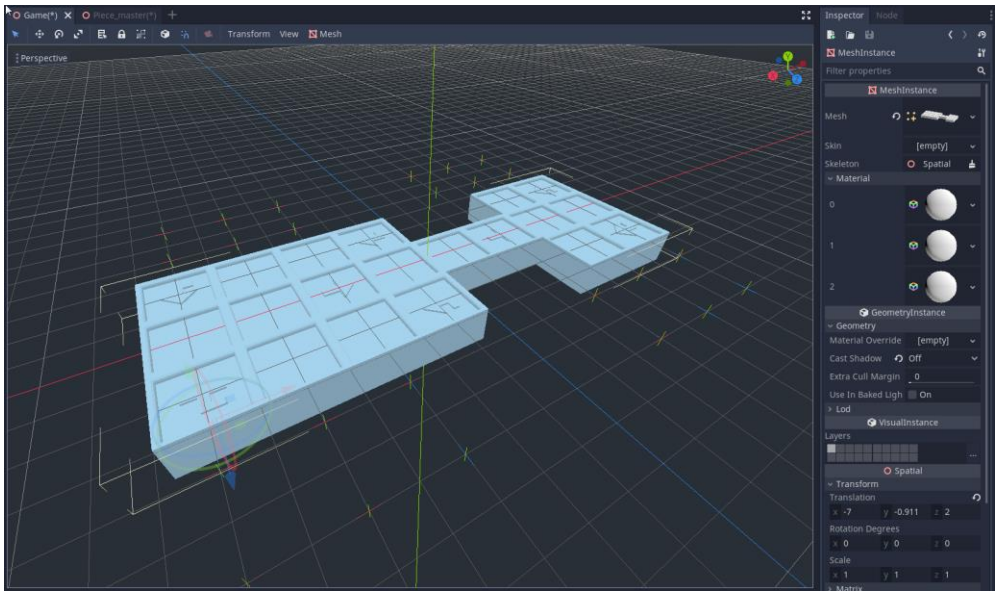


Slika 35. Prikaz modela strjelica u Blenderu

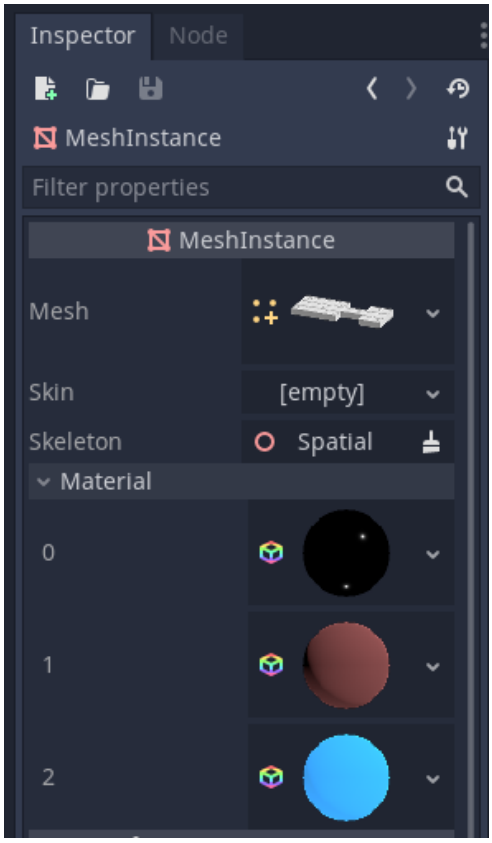


Slika 36. Prikaz gotovog modela ploče za prvu varijantu

Nakon što je 3D model ploče izrađen, izvezen je iz Blendera u *.obj* formatu. Zatim model je uvežen u Godot program te dodijeljeni su mu materijali.

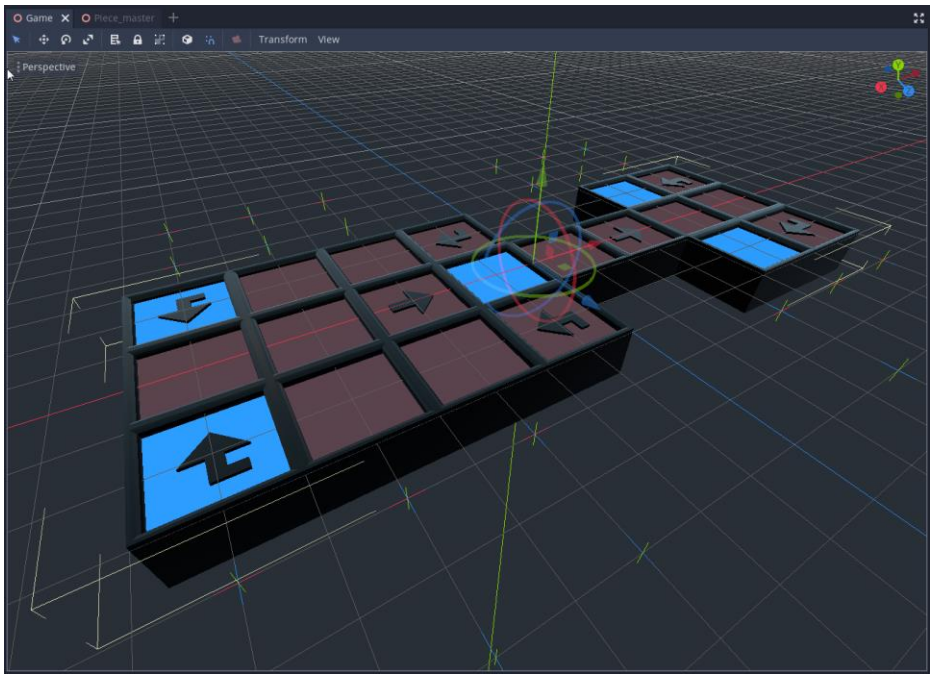


Slika 37. Prikaz modela ploče prve varijante unutar Godota



Slika 38. Prikaz postavki materijala za prvu varijantu ploče

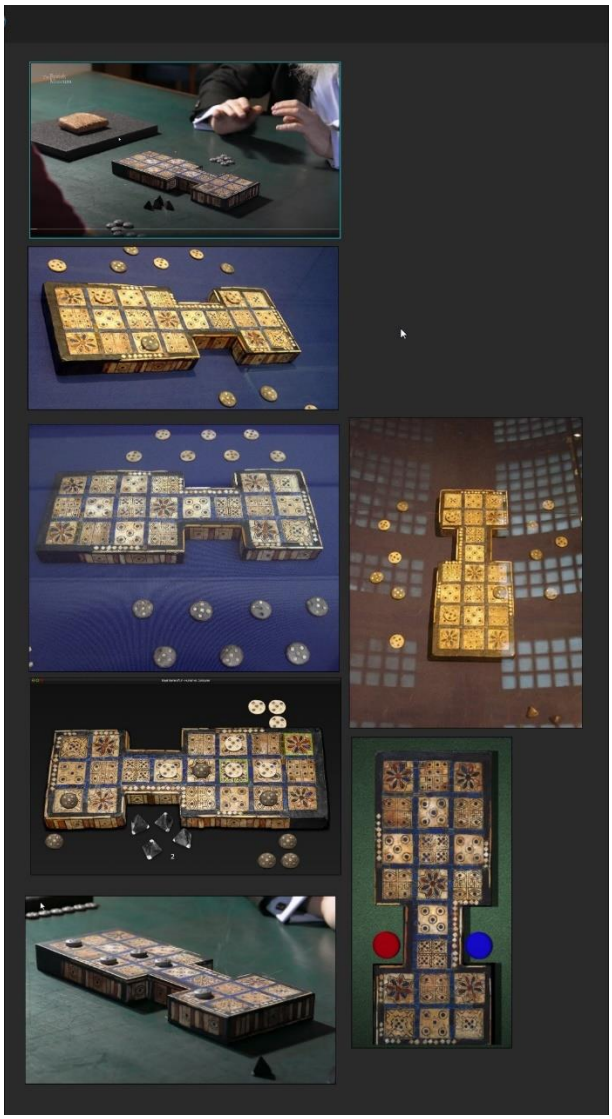
S ovime je završena izrada prve varijante modela za igru. Nadalje izrađivati će se druga varijanta 3D modela za igru.



Slika 39. Prikaz gotovog modela ploče za prvu varijantu unutar Godota

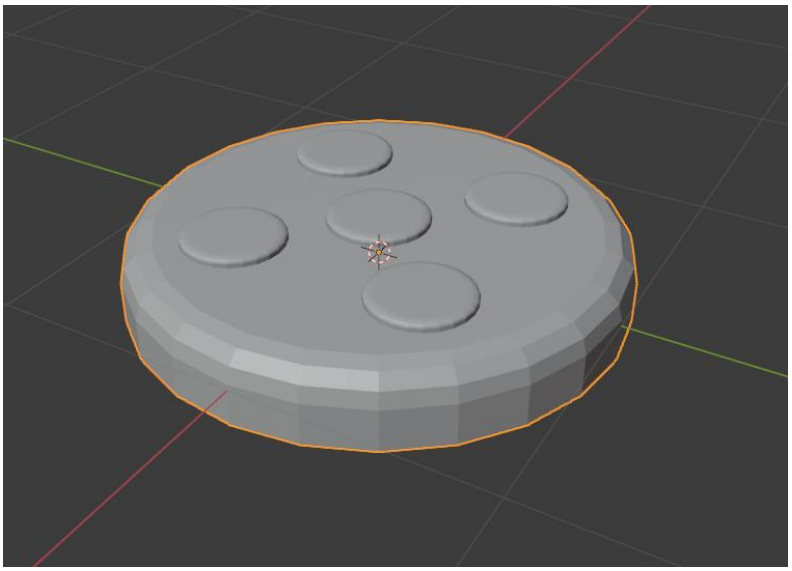
3.2.2. Druga varijanta 3D modela

Prije početka procesa modeliranja i teksturiranja druge varijante modela, uključen je PureRef program na drugom monitoru te postavljene su referentne slike na radnu površinu programa.

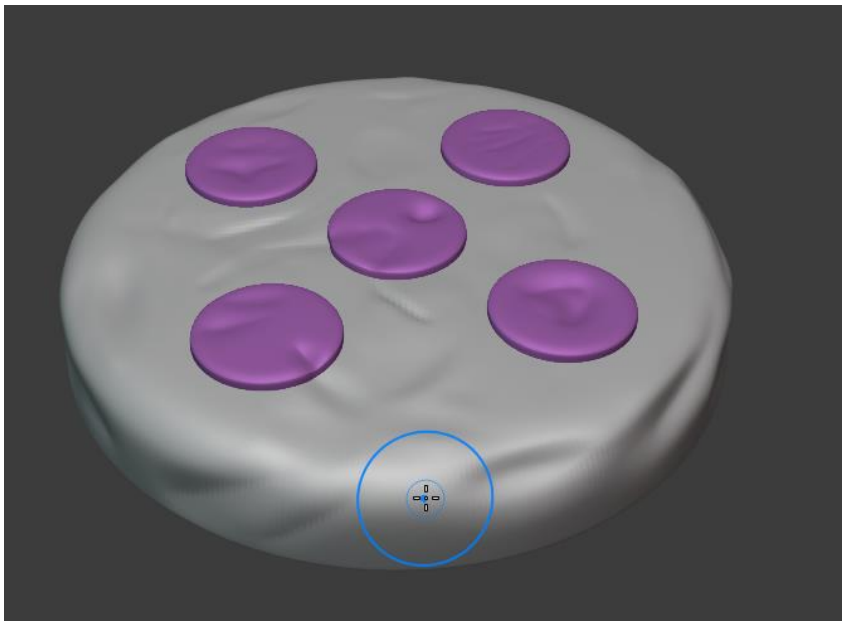


Slika 40. Snimka zaslona PureRef programa s postavljenim referentnim slikama

Za izradu druge varijante 3D modela korišteni su prijašnje izrađeni modeli kao predložak.

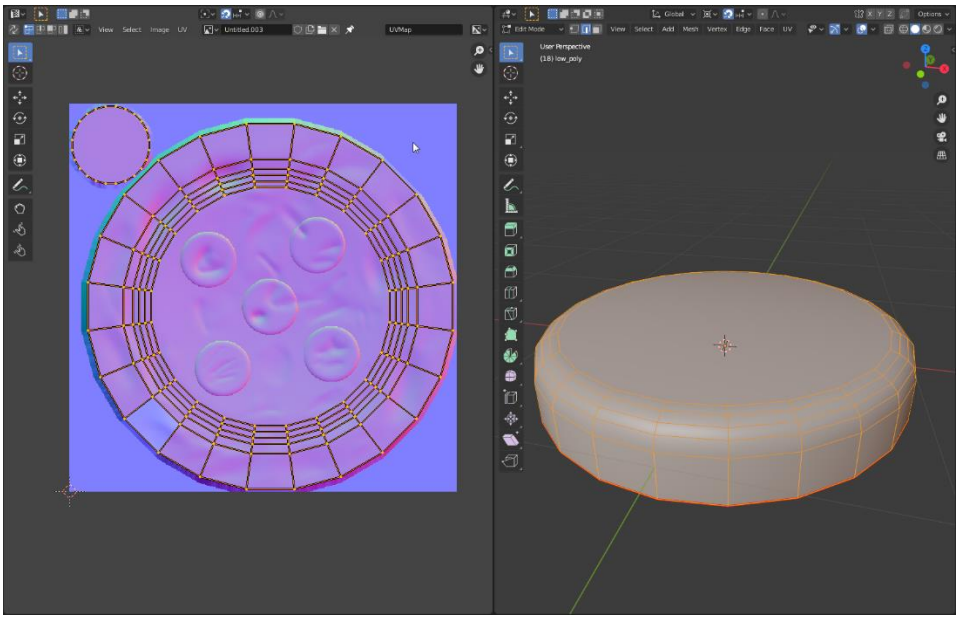


Slika 41. Prikaz modela figurice koji će biti korišten u igri



Slika 42. Prikaz modela figurice visoke razine detalja u Blenderu

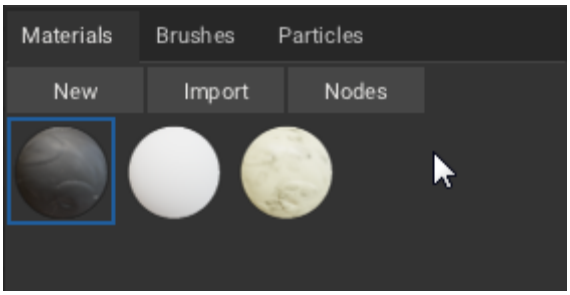
Nadalje izrađena je tekstura s normalama.



Slika 43. Prikaz rastvorenog modela figurice u Blenderu

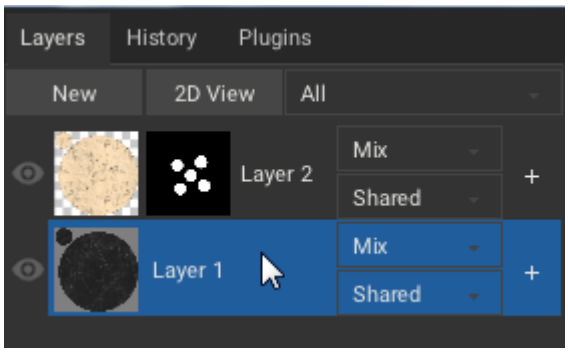
Tekstura je pohranjena s nazivom *piece_NORM*, jednostavni model izvezen je iz Blendera putem *.obj* formata s imenom *piece_druga_varijanta.obj*, a zatim uvezen je zajedno s teksturom u ArmorPaint.

Unutar ArmorPaint programa potrebno je izraditi teksture koje će biti korištene za izradu materijala u Godotu. Izrađeni su materijali.

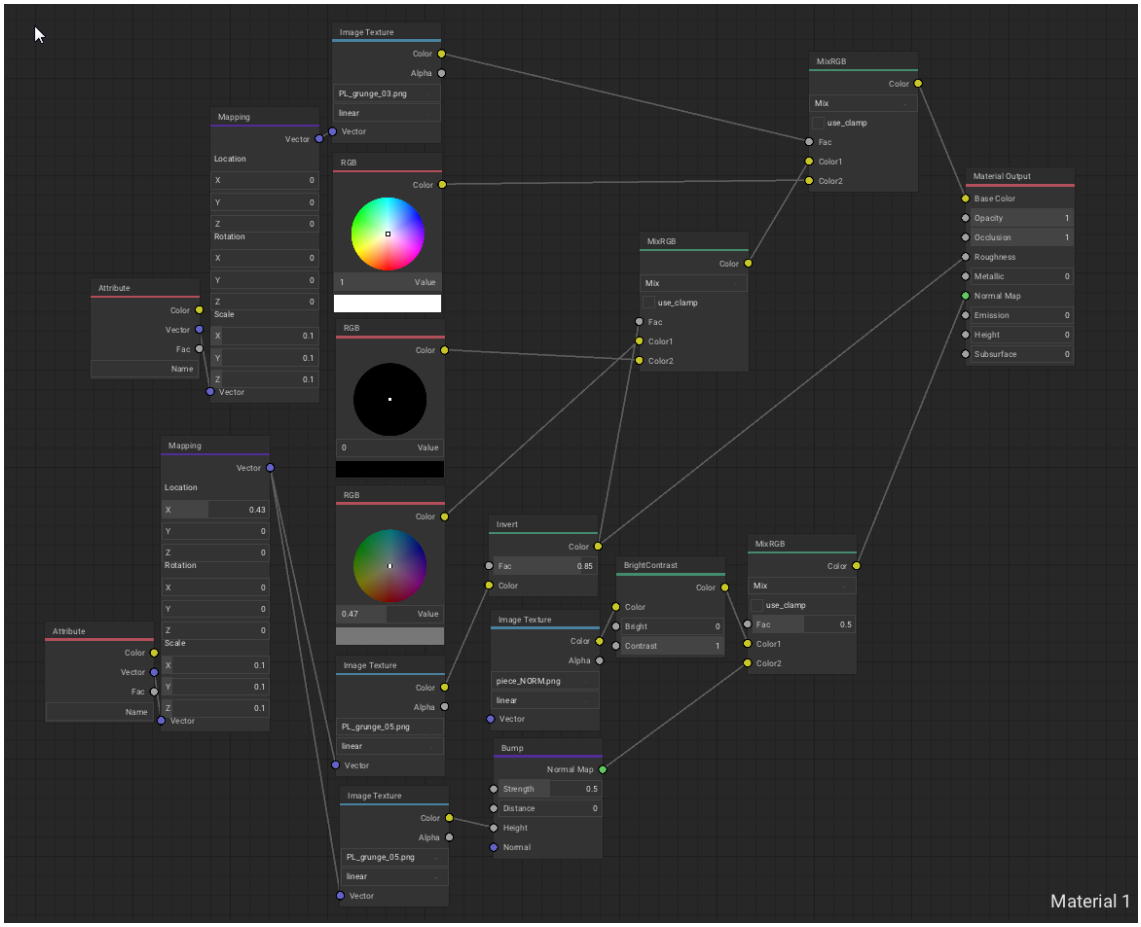


Slika 44. Prikaz materijala za izradu crne figurice u ArmorPaintu

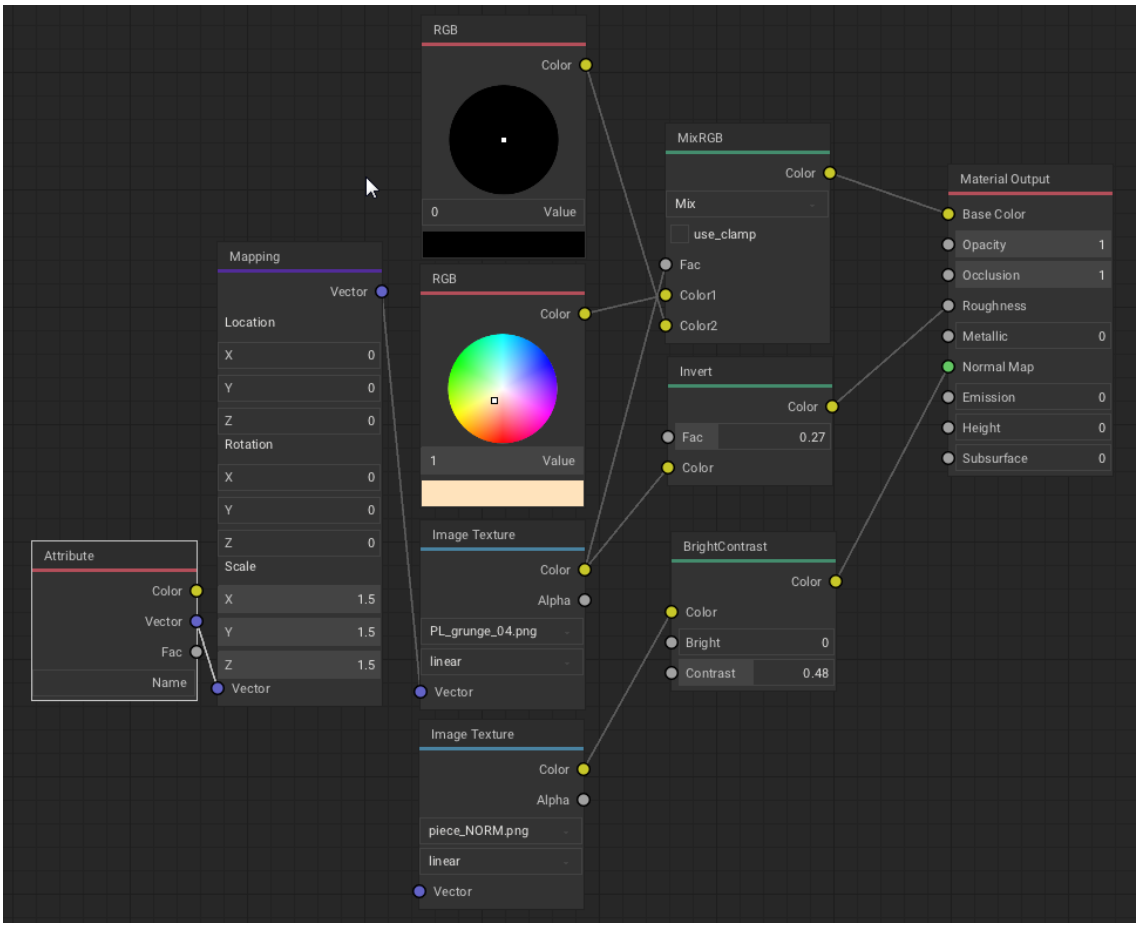
Nadalje dodan je novi sloj (eng. *Layer*) u programu te na tom sloju napravljena je crno-bijela maska. Putem maske biti će određena područja na modelu [13] na koja će biti apliciran drugi materijal na drugom sloju.



Slika 45. Prikaz slojeva i maske u ArmorPaintu



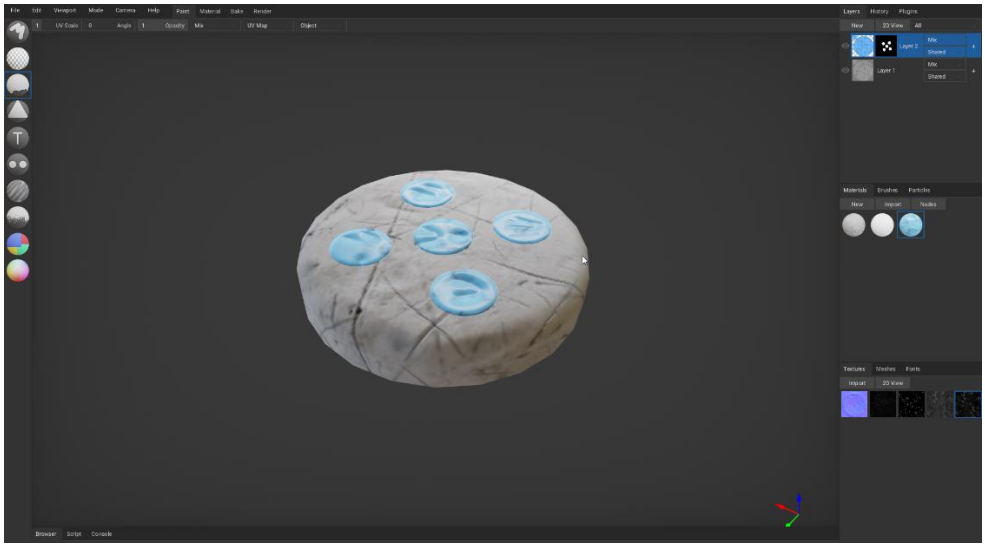
Slika 46. Prikaz postavki čvorova u ArmorPaintu za prvi materijal na crnoj figurici



Slika 47. Prikaz postavki čvorova u ArmorPaintu za drugi materijal na crnoj figurici

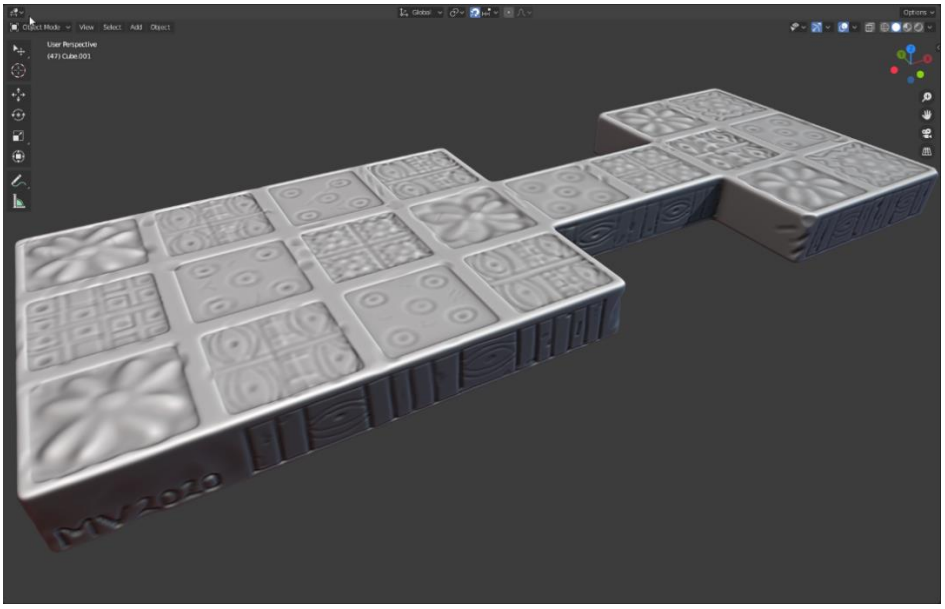
Slika 48. Prikaz gotovih materijala za crnu figuricu u ArmorPaintu

Za izradu tekstura bijele figurice korišteni su isti materijali, samo su izmijenjene postavke za boju unutar čvorova.



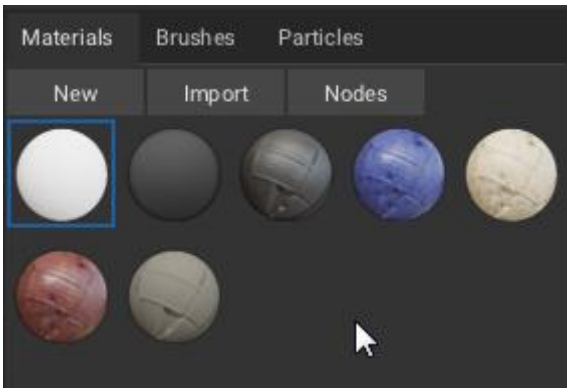
Slika 49. Prikaz gotovih materijala za bijelu figuricu u ArmorPaintu

Naposljetku napravljen je izvoz tekstura izvan programa za bijelu i crnu figuricu u rezoluciji 512x512 piksela. Nadalje potrebno je izraditi drugu varijantu ploče.

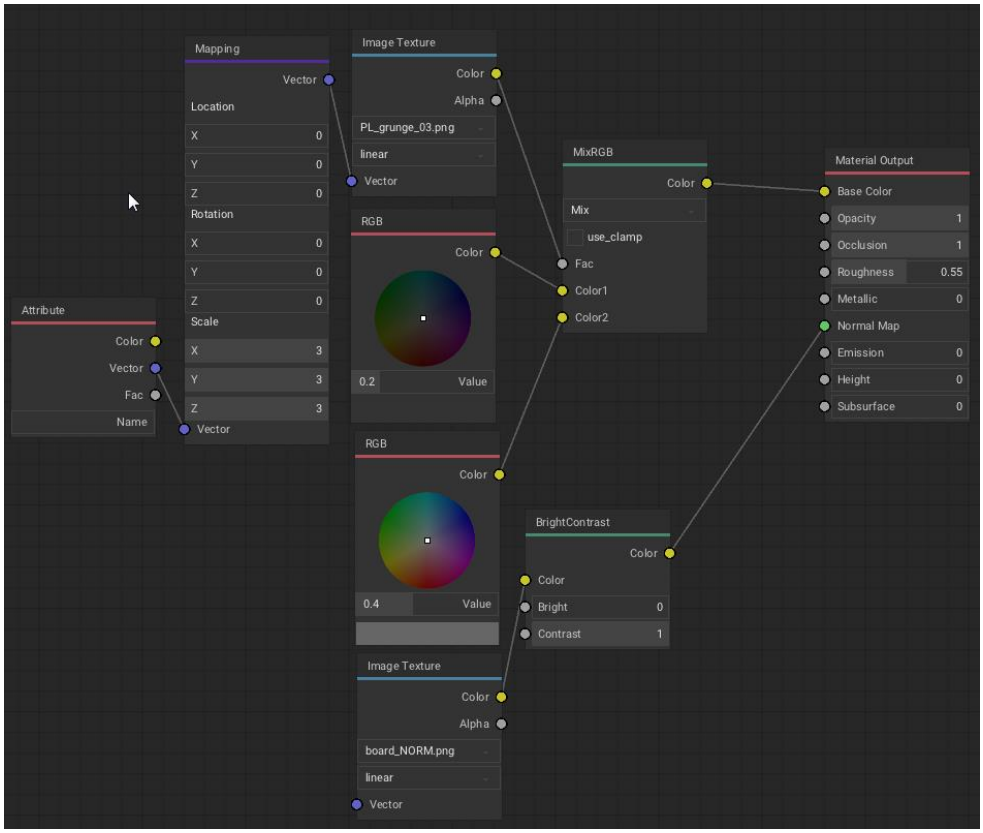


Slika 50. Prikaz modela ploče visoke razine detalja u Blenderu

Istom metodom kao i kod izrade figurice izrađena je tekstura s normalama. Prvi model i tekstura s normalama izvezeni su iz Blendera, a zatim su uvezeni u ArmorPaint. Unutar ArmorPainta izrađeno je 7 materijala.

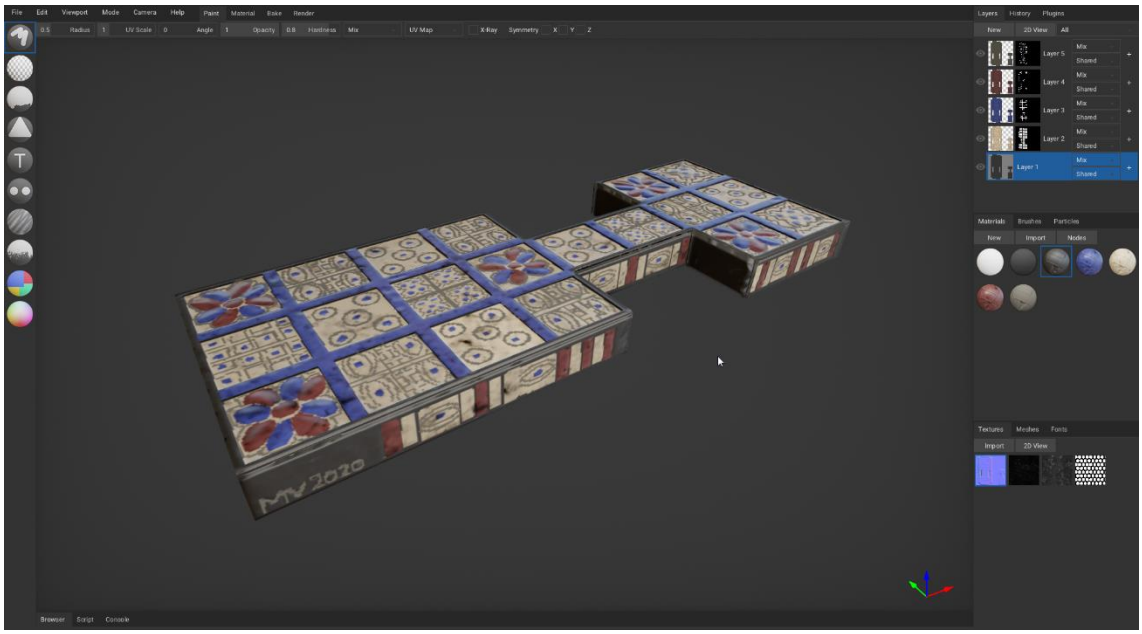


Slika 51. Prikaz postavljenih materijala za ploču u ArmorPaintu



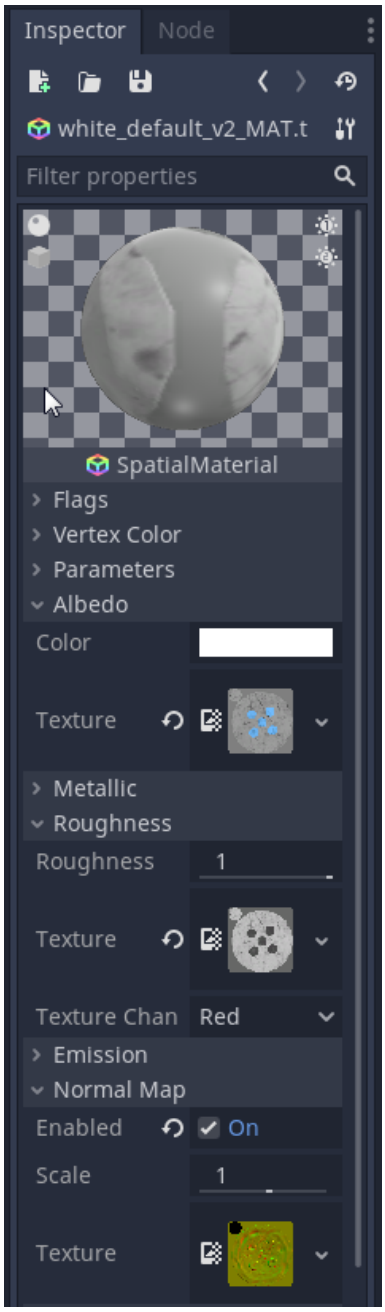
Slika 52. Prikaz postavki čvorova za prvi materijal ploče u ArmorPaintu

Nakon što su izrađeni svi materijali napravljeno je 5 slojeva. Na slojevima aplicirani su materijali i izrađene su maske. Ovime je postupak završen i teksture su izvezene iz programa u rezoluciji 512x512 piksela.

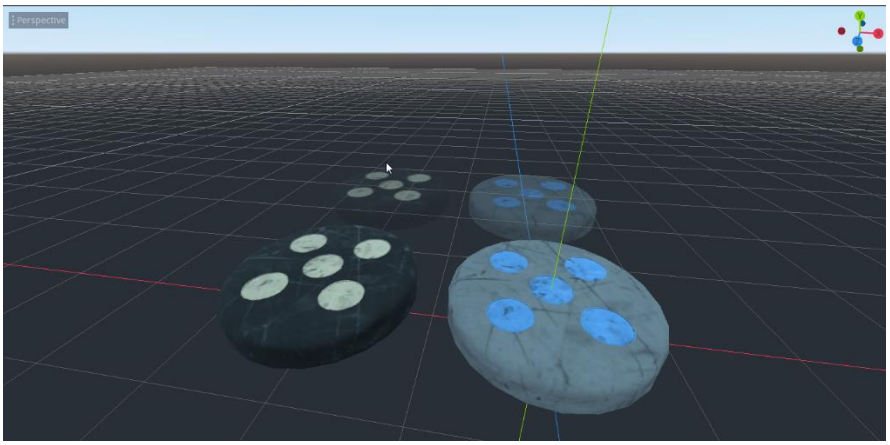


Slika 53. Prikaz gotovog modela ploče u ArmorPaintu

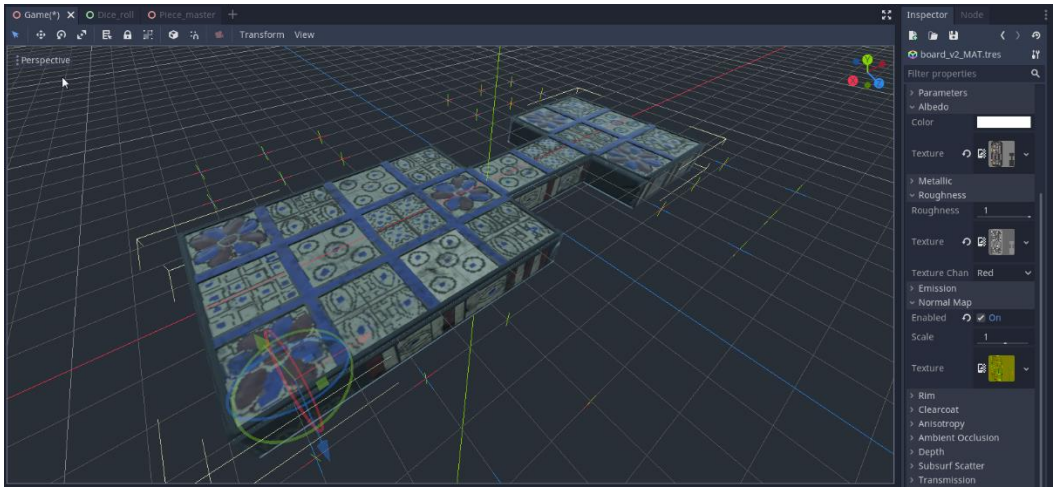
Modeli figurice i ploče zajedno s njihovim pripadajućim teksturama uvezeni su u Godot. Zatim otvorena je scena s glavnom figuricom i postavljeni su materijali.



Slika 54. Prikaz izbornika za postavke materijala u Godotu



Slika 55. Prikaz figurica s materijalima u Godotu



Slika 56. Prikaz modela ploče s materijalima unutar Godota

Nakon postavljanja materijala u Godotu, izrada druge varijante modela je gotova. Nadalje potrebno je izraditi animacije.

3.3. Animacija pomicanja figurica

Za izradu animacije pomicanja figurice na ploči korišten je *Tween* čvor unutar scene za glavnu figuricu. Otvorena je programska skripta za tu scenu. U skripti definirana je funkcija *animate_move_piece(selectedPiece, finalPos):*.

```
88 func animate_move_piece(selectedPiece, finalPos):
89     MovePieceTween.interpolate_property(selectedPiece, "translation", selectedPiece.translation,
90         finalPos.translation, 0.2, Tween.TRANS_SINE, Tween.EASE_OUT)
91     MovePieceTween.start()
92     play_move_piece_sound()
```

Isječak kôda 52. Prikaz funkcije za animiranje figurica

Pomoću ove funkcije vremenski će se interpolirati vrijednosti između trenutne lokacije figurice i buduće lokacije na kojoj će figurica završiti [5]. Bitno je napomenuti da prilikom pozivanja ove funkcije moraju se navesti dvije varijable. Te dvije varijable *selectedPiece* i *finalPos* predstavljati će prvi objekt koji se želi animirati i drugi objekt prema kojemu će se pomicati prvi objekt. U prvom retku funkcije putem naredbe *MovePieceTween* kôd se referencira na čvor te pomoću naredbe *.interpolate_property()* želi se interpolirati određena vrijednost. Prilikom pozivanja ove naredbe potrebno je definirati 7 parametara. Prvi parametar definirati će objekt na kojemu se želi interpolirati vrijednost. U ovom slučaju želi se interpolirati vrijednost na prvom objektu odnosno na figurici koja je trenutno označena. Zatim drugi parametar definirati će opciju koju se želi interpolirati. Pomoću *„translation“* navedeno je da se želi interpolirati translacija. Nadalje u trećem parametru potrebno je definirati početnu vrijednost s koje će započeti interpolacija. Putem naredbe *selectedPiece.translation* definirano je da za početnu vrijednost interpolacije biti će uzeta trenutna vrijednost za translaciju označene figurice. Iduće u četvrtom parametru potrebno je definirati završnu vrijednost prema kojoj će se izvršavati interpolacija. To će biti lokacija polja na ploči ili figurice koja je drugačije boje. U petom parametru potrebno je brojčano definirati vremensko trajanje interpolacije – ovdje je navedeno da će trajanje interpolacije će biti 0.5 sekundi. Putem šestog parametra potrebno je definirati način tranzicije u animaciji. Načini tranzicije su ustvari oblici krivulje [11] koje će pratiti interpolirana vrijednost tijekom postupka interpolacije. Kao parametar postavljeno je *Tween.TRANS_SINE* odnosno definirano je da za krivulju interpolacije biti će korištena sinus krivulja matematičke funkcije.Naposljetku u zadnjem, sedmom parametru odabrano je putem naredbe *Tween.EASE_OUT* koja od četiri varijante će [11] biti postavljena za tip krivulje interpolacije. Odabrana je druga varijanta. U zadnjem retku funkcije pomoću naredbe *MovePieceTween.start()* započeti će se izvođenje animacije unutar igre. Ovom zadnjom naredbom gotova je izrada animacija za figurice. Iduće potrebno je izraditi zvukove za igru.

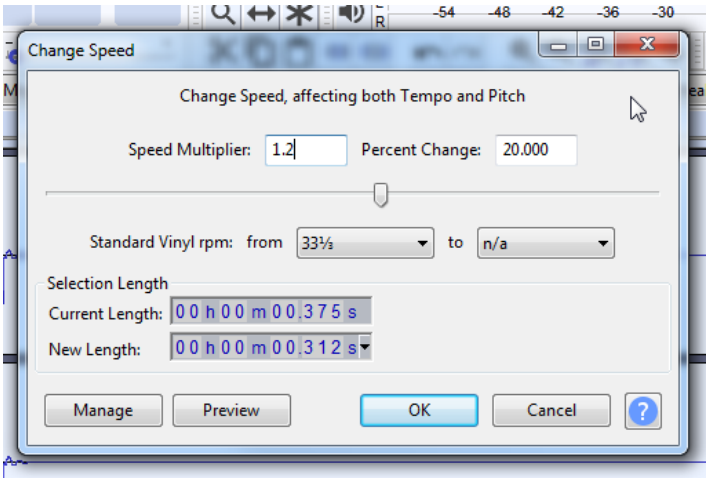
3.4. Zvučni efekti i pozadinska glazba

Datoteke sa zvučnim efektima i pozadinskom glazbom preuzete su s interneta. Putem Audacity programa na te datoteke dodani su jednostavni efekti kako bi se brzo i efektivno postigle različite varijacije istih zvukova. Potrebno je obratiti pozornost na format datoteke prilikom izvoza iz programa. Preporučljivo je koristiti datoteke formata *.wav* za kratke zvukove, odnosno kratke zvučne efekte unutar igre [5]. Nadalje preporučljivo je za dugačke glazbene datoteke koristiti format *.ogg* [5].

3.4.1. Izrada zvučnih efekata

Zvukovi koji će biti postavljeni kao zvučni efekti unutar igre preuzeti su s internetske stranice https://freesound.org/. Preuzet je po jedan zvučni efekt za micanje, izbacivanje i označavanje figurica. Nadalje preuzet je zvučni efekt za bacanje kockica i proglašavanje pobjednika na kraju kruga igre.

Prvo uključen je program Audacity, a zatim uvezen je zvučni efekt za pomicanje figurice. Zatim zvuk je izvezen iz programa u *.wav* formatu. Ovo će biti prva varijanta zvučnog efekta za pomicanje figurica. Nadalje pomoću naredbe *Promijeni Brzinu* (eng. *Change Speed*) zvučni zapis usporen je za 20% te izvezen je iz programa. Ovo će biti druga varijanta zvuka pomicanja figurice. Potom provedena je još jednom ista naredba, ali ovaj put zvuk je ubrzan za 20%. Zatim zvuk je izvezen iz programa kao treća varijanta zvuka pomicanja figurice.



Slika 57. Prikaz naredbe za usporavanje zvuka u Audacity programu

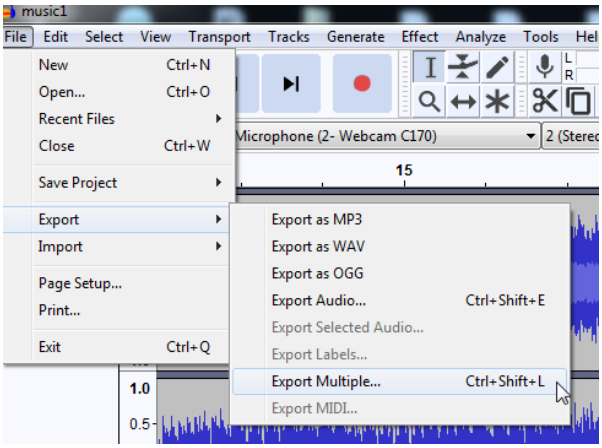
Ostali zvučni efekti izrađeni su na potpuno identičan način.

3.4.2. Pozadinska glazba

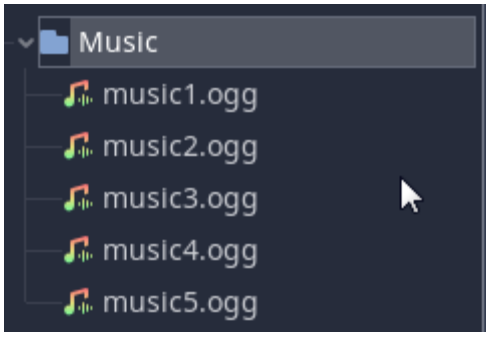
Za pozadinsku glazbu preuzeto je 5 pjesama s internetske stranice https://freemusicarchive.org/, pritom obraćena je posebna pozornost na vrstu autorskih prava za pojedinu pjesmu. Korištena je prethodno navedena internetska stranica jer ta stranica ne polaže autorska prava i preuzete zvučne datoteke mogu se koristiti i u komercijalnim projektima. Nazivi pjesama koje su preuzete: *Shipping Lanes, Organisms, Negentropy*, *Moonrise* i *Algorithms*, a izvođač je *Chad Crouch*. Žanr odabranih pjesama je elektronična glazba te ovaj stil vrlo dobro se uklopio u vizualni izgled igre.

Pokrenut je Audacity program te uvezeno je svih 5 pjesama. Zatim pjesme su izvezene putem naredbe *Izvezi više datoteka* (eng. *Export Multiple*). Kvaliteta je postavljena na vrijednost 3 – ovime se pokušao smanjiti prostor koji će zauzimati datoteka na prostoru za pohranu mobilnog uređaja. Pri tome pazilo se da kvaliteta ne bude preniska što bi dovelo do pojave zvučnog šuma prilikom reprodukcije glazbe. Datoteke su izvezene u obliku *.ogg*[16] formata s nazivima *music1*, *music2*, *music3*, *music4* i *music5*. U konačnici datoteke su uvežene u Godot program te postavljene su unutar *Music* mape u sustavu datoteka.

[16] *.ogg* – sufiks za glazbenu datoteku koja koristi *OGG Vorbis* vrstu zvučne kompresije



Slika 58. Prikaz izbornika i naredbe za izvoz više datoteka odjednom



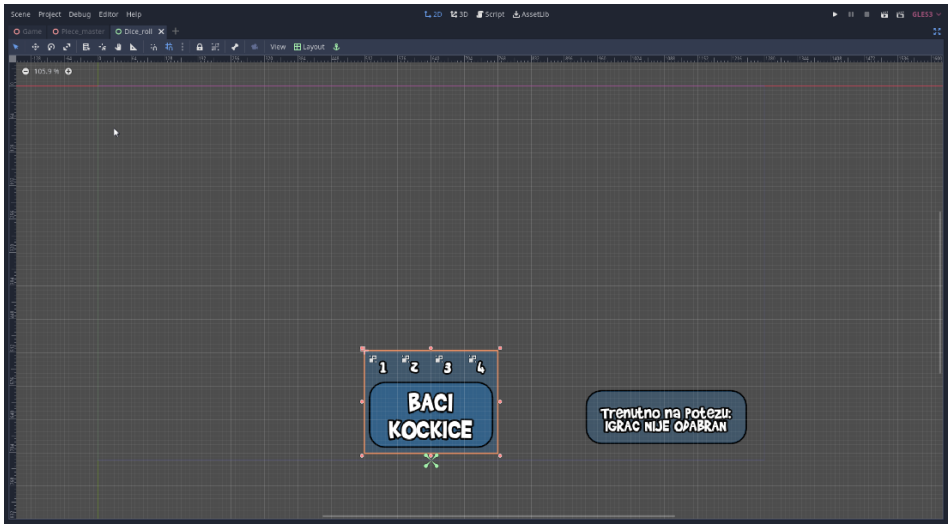
Slika 59. Prikaz mape sa glazbenim datotekama

3.5. Korisničko sučelje

3.5.1. Izrada sučelja za bacanje kockica

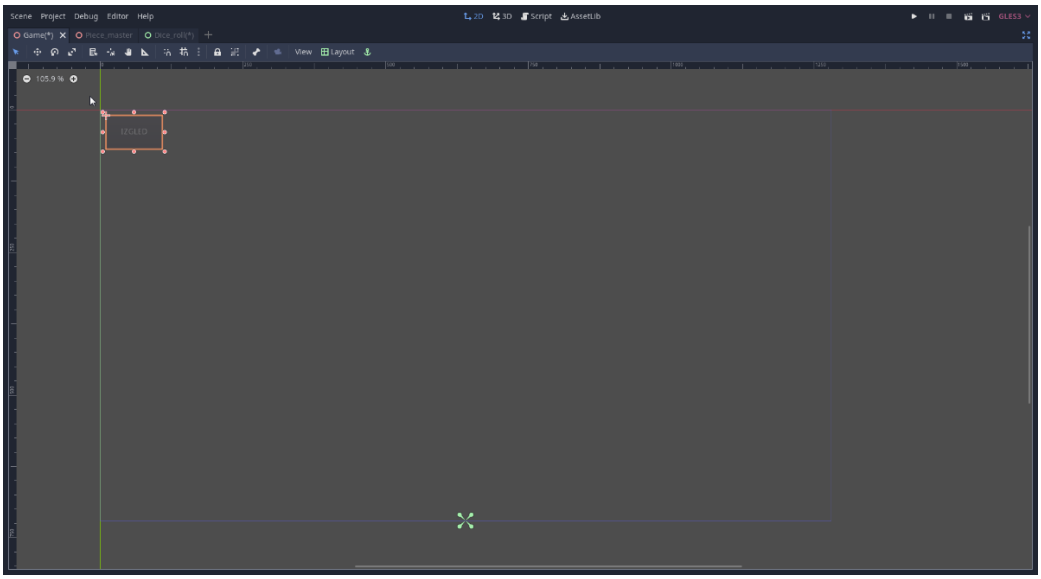
Kako bi se izradilo korisničko sučelje u igri potrebno je postaviti pozicije čvorova unutar scene za bacanje kockica. U Godotu otvorena je prvo scena za bacanje kockica. Zatim označen je čvor *Spremnik* te postavljena mu je pozicija pri dnu zaslona na sredini. Nadalje označen je čvor *Oznaka* te smješten je u donji-desni kut zaslona. Zatim postavljena su sidra (eng. *Anchor*) za čvorove. Pomoću sidra čvor će biti vezan za određeni dio ruba zaslona. Ukoliko dođe do promjene rezolucije zaslona, čvorovi će uvijek biti pozicionirani na zaslonu u odnosu na svoja sidra. Sidro za spremnik postavljeno je pri dnu zaslona, na sredini. Zatim sidro za oznaku postavljeno je u donji-desni kut. Iduće potrebno je postaviti font za tekst koji će se prikazivati na zaslonu. Na internetskoj stranici https://www.1001freefonts.com/ preuzeta su dva fonta: *BradyBunch* i *Chocolate Bar*. Brojkama iznad gumba za bacanje kockica dodijeljen je drugi font, a na

gumb i polje u donjem-desnom kutu dodijeljen je prvi font. Zatim gumbu i polju postavljen je obrub na blago-ovalni te boja obruba postavljena je na crnu boju. *RGBA* vrijednosti za gumb postavljeni su na: 0, 138, 255 i 50. Dok za spremnik i polje postavljene su vrijednosti: 0, 138, 255 i 40.



Slika 60. Prikaz postavljenih čvorova za korisničko sučelje

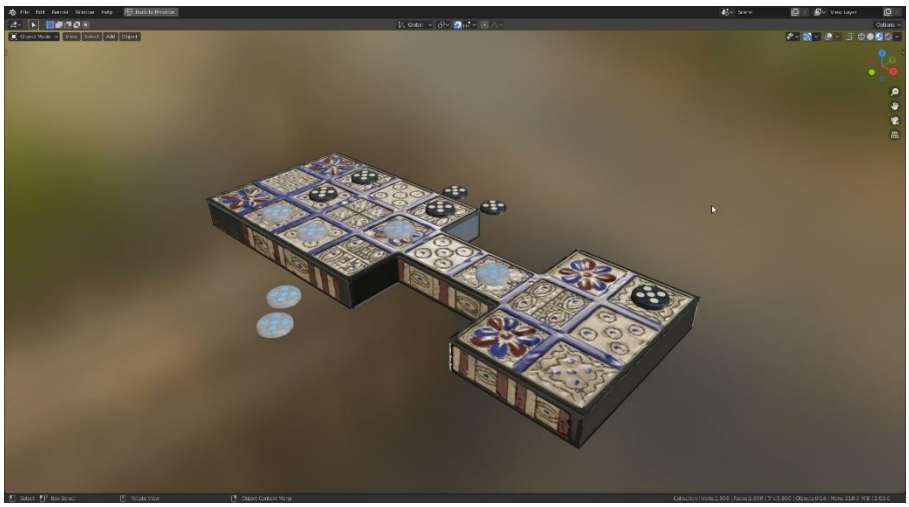
Iduće otvorena je glavna scena igre. Označen je gumb te pozicioniran je u gornji-lijevi kut zaslona. Sidro ovog gumba postavljeno pri sredini ekrana na dno. Nadalje izmijenjena je vrijednost opaciteta materijala na 90, a za boju teksta postavljena je obična bijela boja. Ovime definiran je izgled sučelja te izrada korisničkog sučelja je završena.



Slika 61. Prikaz postavljenog gumba na glavnoj sceni igre

3.6. Izrada ikone za mobilnu aplikaciju

Ikona za mobilnu aplikaciju izrađena je u InkScape programu. Prvo uključen je Blender program te na scenu su postavljeni modeli ploče i figurice zajedno s pripadajućim teksturama. Potom sceni dodjeljena je *HDR*[17] slika koja će osvjetljavati modele na sceni.



Slika 62. Prikaz postavljene scene prije postupka iscrtavanja

Nadalje, postavljena je kamera na scenu te usmjeren joj je pogled na modele. U postavkama za kameru pod sekcijom *Leća* (eng. *Lens*) za opciju *Tip* (eng. *Type*) postavljeno je *Ortografsko* (eng. *Ortographic*). Na ovaj način ukloniti će se distorzija slike uslijed perspektive kamere [12]. Dimenzije slike za iscrtavanje postavljene su na 1:1 kvadratni oblik te napravljeno je iscrtavanje slike u *Cyclesu*[18]. Gotova iscrtana slike izvezena je iz programa u dimenzijama 512 x 512 piksela te zatim slika je uvezena u InkScape program.

[17] Visoki dinamički raspon (eng. *High dynamic range*) – u fotografiji ovo je slika koja ima visok dinamički raspon tonova
[18] *Cycles* 3D iscrtavač (eng. *3D renderer*) – dio programa u Blenderu pomoću kojega se može iscrtavati slika na zaslonu računala

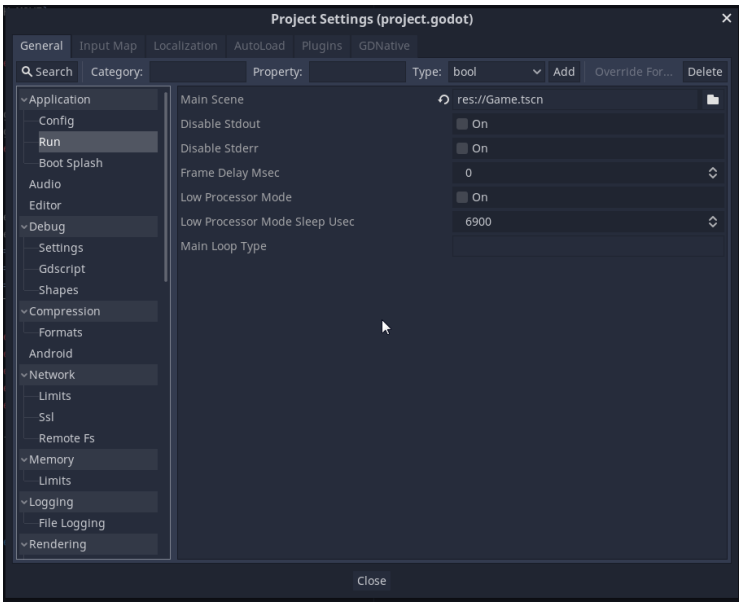
Slika 63. Prikaz iscrtane slike u Blenderu

Unutar InkScape programa napravljena su dva kvadratna oblika te zaglađeni su im vrhovi. Boja prvog kvadrata postavljena je na *RGBA* vrijednosti: 62, 62, 62 i 255. *RGBA* vrijednosti boje za drugi kvadrat postavljene su na: 150, 230, 179 i 255. Kvadrati su pozicionirani tako da se preklapaju jedan preko drugoga te zajedno tvore rub oko cijele ikone. U sredinu tih kvadrata skalirana i centrirana je iscrtana slika s modelima. Zatim na ikonu dodan je tekst *Royal Game of Ur* – tekstu je dodijeljen font pod nazivom *Carolingia(BigfooT)*. Naposljetku napravljen je izvoz slike u rezoluciji 192 x 192 piksela te gotova ikona uvezena je u Godot program. Ikona će biti postavljena unutar aplikacije u sljedećem koraku izrade.

Slika 64. Prikaz gotove ikone za aplikaciju

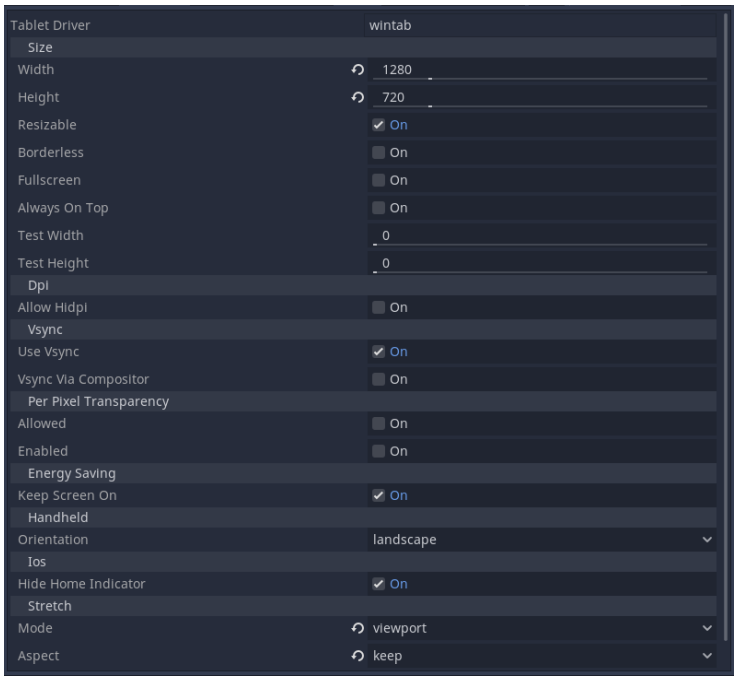
3.7. Izvoz aplikacije na mobilni uređaj

Prije izvoza mobilne aplikacije iz Godot programa potrebno je prvo odrediti postavke projekta. Na gornjoj alatnoj traci programa unutar izbornika *Projekt* (eng. *Project*) odabrana je opcija *Postavke Projekta...* (eng. *Project Settings...*). Unutar novootvorenog prozora u postavkama za *Aplikaciju* (eng. *Application*) označena je opcija *Pokretanje* (eng. *Run*). Zatim pod opciju *Glavna Scena* (eng. *Main Scene*) dodjeljena je datoteka *Game.tscn*. Ovime je postavljena glavna scena igre kao prva scena koja će se učitati prilikom pokretanja aplikacije.



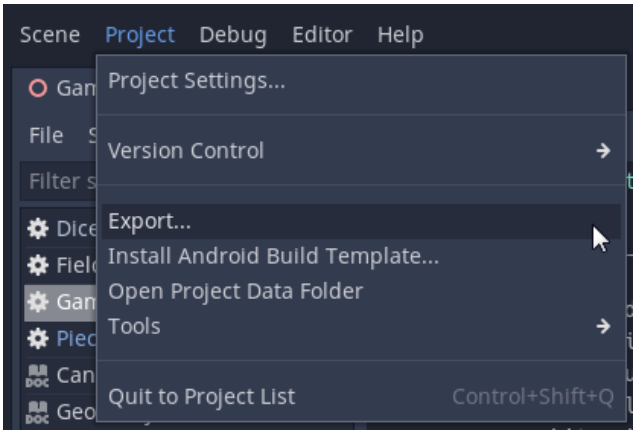
Slika 65. Prikaz postavki aplikacije

Nadalje pod postavkama za *Zaslon* (eng. *Display*) označena je opcija *Prozor* (eng. *Window*). Unutar ove opcije postavljena je rezolucija u kojoj će se odvijati igra. Za rezoluciju određena je vrijednost 1280 x 720 piksela. Iduće pod opcijom *Rastezanje* (eng. *Stretch*) stavka *Način* (eng. *Mode*) postavljena je na *Radni prozor* (eng. *Viewport*), a stavka *Omjer* (eng. *Aspect*) postavljena je na *Zadrži* (eng. *Keep*). Pomoću ovih opcija određeno je da aplikacija ima definiranu rezoluciju od koje neće odstupati. Ukoliko mobilni uređaj na kojemu se prikazuje igra bude imao različite proporcije ekrana, igra će zadržati svoj omjer, ali prilagoditi će se tim proporcijama ekrana [5].



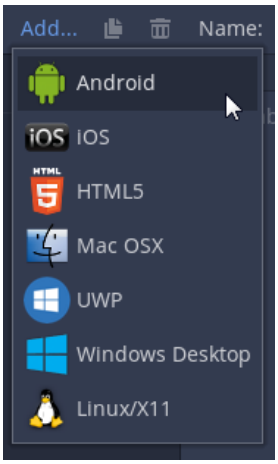
Slika 66. Prikaz postavki za zaslon uređaja

Nakon određenih postavki aplikacije igra je spremna za izvoz iz programa. Odlaskom na alatnu traku pod izbornikom *Projekt* odabrana je opcija *Izvoz*.



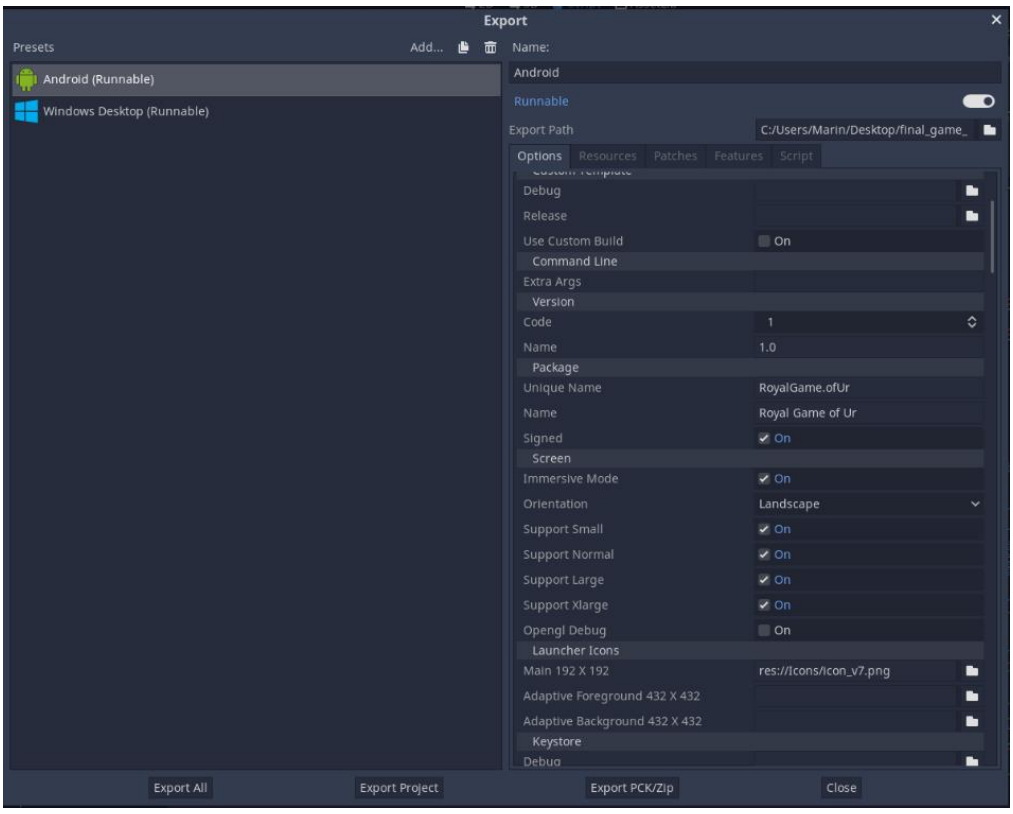
Slika 67. Prikaz izbornika za izvoz aplikacije

U novootvorenom prozoru potrebno je dodati profil s postavkama za izvoz aplikacije na Android platformu.



Slika 68. Prikaz mogućih platformi za izvoz u Godotu

U postavkama za izvoz na Android platformu potrebno je dodijeliti sliku ikone koja će se zatim dodijeliti izvezenoj aplikaciji. Pod opcijom *Ikone pokretača* (eng. *Launcher Icons*) potrebno je prvoj stavci dodijeliti ikonu za igru. Nakon što je ikona postavljena odabirom naredbe *Izvezi Projekt* (eng. *Export Project*) aplikacija će biti izvezena. Na kraju aplikaciju je potrebno prebaciti na mobilni uređaj te instalirati. Ovime je izrada mobilne aplikacije u potpunosti gotova.

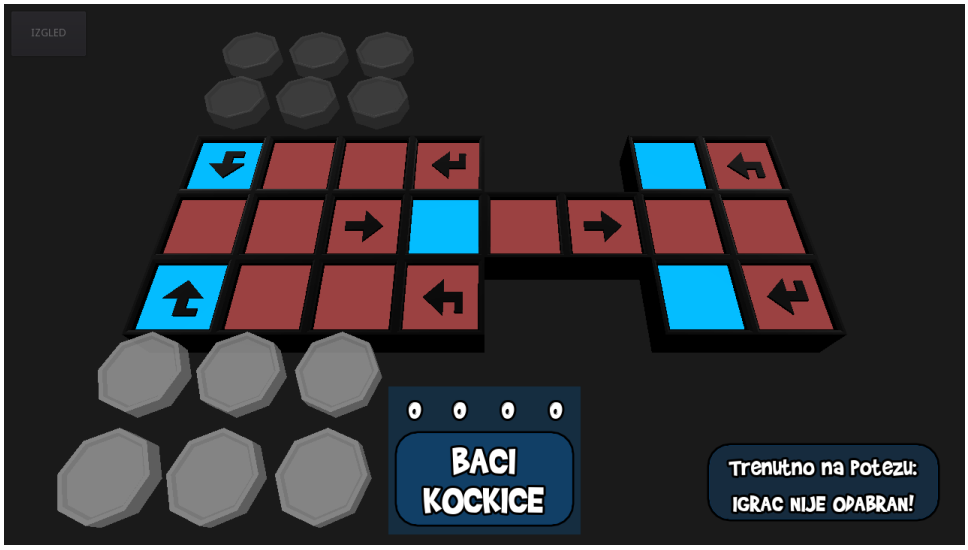


Slika 69. Prikaz postavki za izvoz aplikacije na Android platformu

4. REZULTAT I RASPRAVA

4.1. Izgled i funkcionalnost konačno dobivene igre

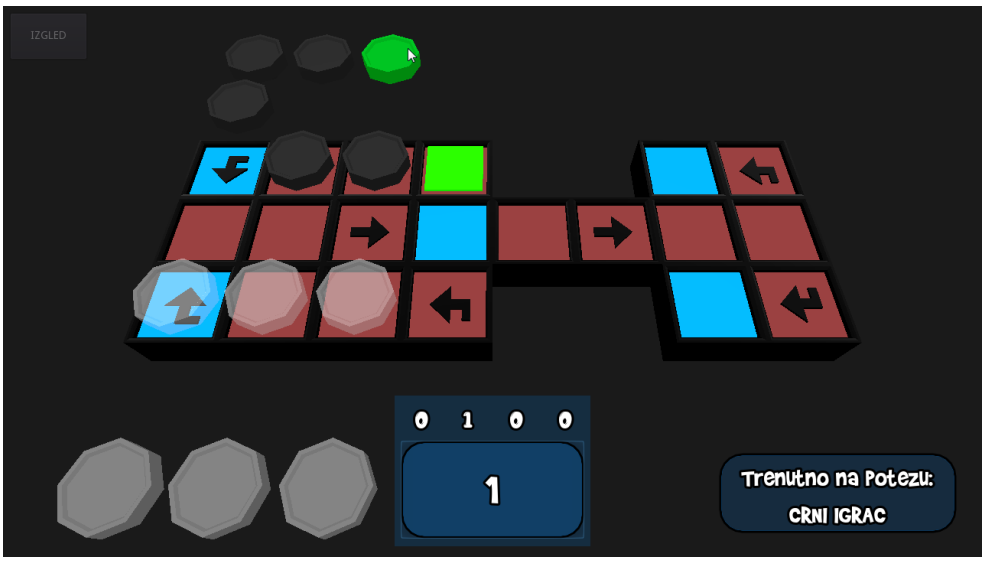
Izgled konačno dobivene igre vrlo je zadovoljavajući. Animacije iako imaju jednostavan izgled vrlo dobro se uklapaju u vizualni identitet cijele igre te na jasan način prikazuju pomicanje figurica. Izgled gumba za bacanje kockica solidno je izrađen. Gumb ima dobro postavljene dimenzije te nisu uočeni problemi prilikom interakcije s gumbom. Slova na gumbu za bacanje kockica dovoljne su veličine te jasna su i njihova čitljivost je vrlo dobra na ekranu pametnog telefona. Također slova na polju u donjem-desnom kutu ekrana prihvatljive su veličine i čitanje se stoga odvija nesmetano. Tekst na gumbu u gornjem-lijevom kutu ekrana je također vrlo dobro izrađen. Gumb na prvi pogled nije lako uočljiv te ne skreće pažnju tijekom igranja igre, ali nakon određenog vremena može se ipak primjetiti da na tom mjestu postoji gumb.



Slika 70. Prikaz scene unutar igre prije početka kruga

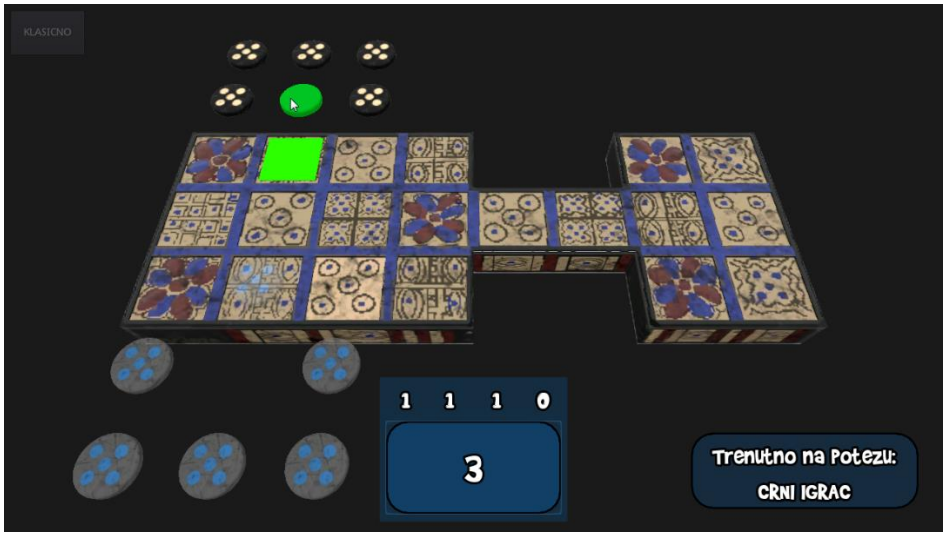
Nadalje, interakcija između korisnika i figurica te gumba za bacanje kockica vrlo je intuitivna. Interakcija u igri uglavnom se odvija u jednom ili dva dodira ekrana što je i više nego zadovoljavajuće. Unatoč tome uočeno je da nepostojanje početnih uputa (tutorijala) za nove je igrače poprilična mana pošto izgled same ploče i kretanje figurica nije u potpunosti razumljivo. Što se tiče izgleda modela, prva varijanta djeluje vrlo dobro. Modeli u ovoj varijanti prikazuju cjelokupnu igru na vrlo jednostavan i intuitivan način. Boje su

jednostavne i na modelima jasno daju do znanja koja boja je trenutno na potezu te gdje će se pomaknuti označena figurica.



Slika 71. Prikaz prve varijante modela u igri

S druge strane, druga varijanta modela iako ima vizualno bogatiji prikaz, u pojedinim dijelovima čini igru dosta neintuitivnom. Izrađene teksture za drugu varijantu modela ponekad previše odvlače pažnju tijekom trajanja igre te djeluju zbunjujuće zbog svoje razine detalja. Također teksture na pojedinim dijelovima ploče djeluju izrazito nekvalitetno izrađene. Pretpostavka je da rezolucija teksture 512 x 512 piksela je ipak premala rezolucija te kako bi se uklonila zrnatost tekstura bilo bi potrebno ponovno izraditi teksture u većoj rezoluciji.



Slika 72. Prikaz druge varijante modela u igri

Što se tiče funkcionalnog dijela aplikacije nisu uočene greške tijekom igranja igre. Odaziv označenih figurica je zadovoljavajući, a ovo također vrijedi i za polja. Funkcija pokretanja nove igre radi bez grešaka. Aplikacija se uglavnom odvija nesmetano, bez greški unutar kôda ili nekakvog zamrzavanja i rušenja programa. Jedini problem koji je uočen je da pritiskom na gumb za promjenu modela aplikacija će se, ukoliko je igra prvi put pokrenuta zamrznuti na djelić sekunde. Pretpostavka je da dolazi do preopterećenja mobilnog telefona prilikom učitavanja tekstura u memoriju. Ova pojava ne kvari cjelokupni dojam igre, ali ipak je uočena.

Zvukovi unutar igre dobro su odrađeni. Označavanje figurica dati će zvukom dobru povratnu informaciju korisniku. Također pomicanje figurica vrlo kratkim zvučnim signalom dati će do znanja korisniku da je figurica pomaknuta. Zvuk za izbacivanje figurica najbolje je izrađen od svih efekata. Zvuk na svojem završnom dijelu ima uzlazni ton te iz tog razloga ostavlja dobar dojam nakon izbacivanja figurice. Nadalje zvuk za bacanje kockica ima odmjereno trajanje te ostavlja dojam kako se u pozadini igre pripremaju kockice na bacanje. Zvuk koji se reproducira prilikom proglašenja pobjednika vrlo kvalitetno je odabran i izrađen. Trajanje tog zvuka je odmjereno i stvara pobjedonosni dojam. Zvučni efekti uglavnom vrlo dobro su izrađeni jer uklapaju se u nekakav minimalistički stil igre. Jedini uočeni nedostatak kod zvučnih efekata je nepostojanje zvučnog signala kada se dobije nula na bacanju kockica. Pozadinska glazba također djeluje minimalistički i vrlo dobro se uklapa uz stil zvučnih efekata. Glasnoća pozadinske glazbe je odmjerena. Glazba ne odvlači pažnju tijekom trajanje igre, ali dovoljno je glasna i igra u nijednom trenutku neće biti u potpunosti tiha.

Tijekom razvoja aplikacije za vizualni dio igre planirana je bila izrada puno više varijanti modela od trenutne dvije dostupne. Također planirano je bilo izraditi složenije animacije te vizualne efekte koji bi se prikazivali na zaslonu mobitela. Nadalje započeto je skladanje pozadinske glazbe za igru s mješavinom egipatskog i perzijskog stila. U konačnici došlo se do zaključka da vrijeme koje je potrebno za individualnu izradu svih tih zamišljenih stavki premašuje preostalo vrijeme do predaje ovog rada. S obzirom da je većina

vremena tijekom razvoja aplikacije utrošena na pisanje i prepravljanje programskog kôda, cjelokupni dojam gotove igre je više nego zadovoljavajući.

4.2. Prednosti i nedostatci Godot jezgre

Kao prednost Godot jezgre može se navesti izrazita jednostavnost korištenja programa. Uvoz modela u program poprilično je jednostavan. Datoteka modela može se mišem povući u program i biti će automatski učitana u program. Nadalje, izrada materijala za model je vrlo intuitivna. Izbornik s materijalima organiziran je na vrlo pristupačan i intuitivan način.

Programski jezik *GDScript* vrlo je jednostavan za korištenje. Uz nekakvo osnovno predznanje iz programiranja bilo je potrebno oko tjedan dana da se usvoji sintaksa jezika. Priručnik unutar Godota koji je dostupan na klik mišem, još jedan je veliki plus za program, a također je uvelike pripomogao u usvajanju sintakse. Nadalje kao velika prednost može se navesti aktivno ukazivanje na greške u kôdu prilikom pisanja programa. Ova funkcionalnost u puno je slučajeva brzo ukazala na grešku koja je nedugo zatim prepravljena i cijeli je proces pisanja kôda iz tog razloga bio brži. Deklariranje funkcija i omogućavanje signala također je vrlo jednostavno unutar skripte.

Izrada animacija unutar programa vrlo je jednostavna pomoću *Tween* čvora. Bilo je potrebno neko određeno vrijeme za shvaćanje koji točno argumenti su ustvari potrebni prilikom pozivanja ove naredbe. No, kada je shvaćen način funkcioniranja čvora primjena je postala jednostavna i brza. Nazivi samih čvorova vrlo su logični te način njihove organizacije na sceni omogućava da na vrlo brz način prolazi se kroz različite verzije tijekom razvoja aplikacije.

Dvije velike prednosti koje svakako treba navesti su cijena programa i veličina jezgre. Program je besplatan i uz sve što nudi za izradu aplikacije, ovo je više nego idealno. Nadalje, veličina jezgre je svega 30-ak megabajta - ovo je doista nevjerojatno. Ako se uzme za usporedbu recimo *Unreal* jezgra (25 gigabajta) dobiva se da je Godot jezgra po veličini datoteke oko 850 puta manja. Resursi za učenje programa lako su dostupni na internetu te ih ima jako puno. Godotov priručnik na internetu vrlo detaljno je napisan, a ukoliko se ne

nađe određeni pojam, pomoću internetske tražilice do tog pojma sigurno će se naići na nekom forumu, videozapisu, unutar izvornog kôda itd.

Kao zadnju prednost može se navesti da je cjelokupno sučelje programa posloženo na vrlo sličan način kao i sučelje unutar programa Blender. Za osobe koje već imaju iskustvo rada u Blenderu Godotovo sučelje izgledati će jako familijarno.

Kao nedostatak Godot jezgre može se navesti da *GDScript* je ponekad dosta nejasan. Za nekoga tko je upoznat s klasičnim načinom pisanja programskog kôda, pisanje skripte u Godotu moglo bi djelovati dosta zbunjujuće jer sintaksa se dosta razlikuje u odnosu na sintaksu klasičnih programskih jezika.

Na pojedinim dijelovima sučelje je dosta neintuitivno i neresponzivno. Prilikom odabira boje za materijal ukoliko se drži lijevi klik te mišem se povuče izvan polja s bojama, neće se više moći odabrati boja. Pokazivač za odabir boje ostat će pričvršćen na pokazivaču miša, neovisno drži li se lijevi klik ili ne. U puno slučajeva ova greška poremetila je postavljenu boju na materijalu, a da se to nije ni primijetilo. Često se pojavljivala iduća greška u programu: prilikom uvoza resursa u program, ukoliko prije nije izričito označena mapa u koju se želi uvesti resurs, resurs će biti uvezen u zadnju mapu koja je bila označena. Ovo znači da puno puta prilikom uvoza resurs je završio u pogrešnoj mapi te bilo ga je potrebno premjestiti u ciljanu mapu. Idući nedostatak je način brisanja resursa iz *Sustava Datoteka*. Dogodilo se dosta puta da je označen resurs te pritiskom na tipku *Delete* pokušalo ga se obrisati. No, resurs se nije obrisao, već je bio obrisan označeni objekt na sceni. Kako bi se obrisalo nešto iz *Sustava Datoteka* to se mora napraviti isključivo putem desnog klika mišem te odabirom opcije *Obriši*. Ovakva nelogičnost unutar jezgre dosta je usporavala rad tijekom razvoja igre.

Kao zadnji nedostatak Godot jezgre može se navesti problem koji je uočen da se pojavljuje na Android platformi nakon izvoza aplikacije. Unutar jezgre omogućene su dodatne opcije za prikaz različitih efekata na ekranu (npr. zamućenje slike, magla itd.). Ovi efekti bili su vidljivi na računalu, dok na

mobilnom uređaju nisu. Ne može sa sigurnošću reći da je ovo krivica Godota, ali na internetu nije se moglo naći objašnjenje ovog problema.

4.3. Mogućnosti pri daljnjem razvoju aplikacije

Utvrđeno je puno mogućnosti za budući razvoj ove aplikacije. U budućnosti aplikaciji bi se mogla dodati funkcionalnost igranja protiv kompjutera i taj kompjuter imao bi tri razine težine. Trenutno igra je napravljena za dva igrača, ali uvođenjem računalnog protivnika, korisnik bi bio u mogućnosti na vlastiti prohtjev odigrati krug igre, bez potrebe za pronalaženjem drugog igrača.

Iduća mogućnost mogla bi biti implementacija sustava dostignuća (eng. *achievment system*). Pomoću ovakvog sustava aplikacija bi bilježila niz različitih vrijednosti vezanih uz igru. Primjeri nekih vrijednosti koje bi se pratile: ukupan broj poteza od prvog pokretanja aplikacije, prosječno trajanje jednog kruga igre, mjerač za najmanji odnosno najveći broj poteza u krugu igre, itd. Zatim unutar aplikacije bile bi prikazane te vrijednosti na statistički način. Također igraču bi se dodijelilo dostignuće ovisno o praćenoj vrijednosti koja je prešla određeni prag. Implementiranjem sustava dostignuća kod korisnika bi se stvorila veća potreba za igranjem igre.

Nadalje za budući razvoj aplikacije moglo bi se implementirati kvalitetnije korisničko sučelje. Trenutno sučelje u igri vrlo je minimalistički napravljeno. Pretpostavka je da putem boljeg sučelja igra bi stvarala veće zadovoljstvo tijekom igranja. Putem izrade više vrsta stilova sučelja te zvučnih efekata za svaku vrstu stila, pretpostavka je da interakcija sa sučeljem bila bi puno privlačnija za korisnika.

Mogućnost implementacije različitih pravila igre – igra se može igrati s izmijenjenim smjerom kretanja figurica.

U konačnici zadnja pretpostavljena mogućnost je da se aplikacija prilagodi za način igranja s jednom rukom. Ovo bi bilo iznimno praktično, pogotovo za korisnike koji svakodnevno putuju gradskim prijevozom.

5. ZAKLJUČAK

Postupak izrade igre za mobilni uređaj bio je dosta zahtjevan, ali i zabavan proces. U relativno kratkom vremenskom roku napravljena je igra zadovoljavajuće kvalitete. Tijekom postupka izrade igre unutar Godota nije se naišlo na nekakve veće greške u programu koje bi bile u stanju obustaviti cijeli proces izrade igre. Uočene su neke manje greške, ali njihov utjecaj na cjelokupno iskustvo korištenja programa bilo je zanemarivo. Zaključeno je da Godot jezgra je iznimno kvalitetan i adekvatan program za izradu mobilne aplikacije.

Od ostalih aplikacija najviše problema uočeno je u ArmorPaintu. Postupak izrade tekstura na trenutke bio je iznimno frustrirajući proces. Preporuka je da se ovaj program ne koristi za ozbiljnije projekte bar dok ne izađe službena verzija.

6. POPIS SLIKA

Slika 1. Prikaz logotipa Godot programa Izvor: https://upload.wikimedia.org/wikipedia/commons/6/6a/Godot_icon.svg 5
Slika 2. Prikaz replike pločice koja je pronađena tijekom iskapanja Izvor: https://commons.wikimedia.org/wiki/File:British_Museum_Royal_Game_of_Ur.jpg#/media/File:British_Museum_Royal_Game_of_Ur.jpg 7
Slika 3. Prikaz kockica za bacanje Izvor: https://cf.geekdo-images.com/imagepage/img/wrr5hLLiMEHdTHU1BJ0kpDi4WTw=/fit-in/900x600/filters:no_upscale():strip_icc()/pic1503607.jpg 7
Slika 4. Prikaz smjera kretanja figurica na ploči 8
Slika 5. Prikaz ploče s označenim rozeta poljima 8
Slika 6. Prikaz logotipa Blender programa Izvor: https://download.blender.org/branding/community/blender_community_badge_orange.png 9
Slika 7. Prikaz korisničkog sučelja u Blender programu 10
Slika 8. Prikaz logotipa ArmorPaint programa Izvor: https://armorpaint.org/img/Logo.png 11
Slika 9. Prikaz sustava čvorova za izradu materijala u ArmorPaintu 11
Slika 10. Prikaz korisničkog sučelja ArmorPainta 12
Slika 11. Prikaz korisničkog sučelja Audacity programa 13
Slika 12. Prikaz korisničkog sučelja u InkScape programu 14
Slika 13. Prikaz PureRef programa s postavljenim referentnim slikama 15
Slika 14. Prikaz scene, sustava čvorova i skripte unutar Godot programa 17
Slika 15. Izgled sustava čvorova za glavnu figuricu 18
Slika 16. Prikaz uglavljenih signala unutar skripte za figuricu 22
Slika 17. Prikaz sustava čvorova za glavno polje 23
Slika 18. Prikaz sustava čvorova za bacanje kockica 27
Slika 19. Prikaz sustava čvorova za glavnu scenu igre 31
Slika 20. Prikaz postavljenih čvorova na glavnoj sceni 47
Slika 21. Prikaz pogleda koji će imati kamera tijekom igre 48
Slika 22. Prikaz izbornika s postavkama za okolinu 48

Slika 23. Prikaz izbornika s opcijama za svjetlinu, kontrast i zasićenost boje na sceni 49
Slika 24. Prikaz organiziranih mapa u sustavu datoteka 49
Slika 25. Prikaz zaglađenih gornjih rubova na figurici 50
Slika 26. Prikaz prve varijante modela u Godotu 50
Slika 27. Prikaz postavki boje za bijelu figuricu 51
Slika 28. Prikaz izbornika i naredbe za pohranjivanje materijala u Godotu 51
Slika 29. Prikaz izbornika i naredbe za dupliciranje trenutno postavljenog materijala u Godotu 52
Slika 30. Prikaz materijala za bijelu figuricu 52
Slika 31. Prikaz materijala za crnu figuricu 53
Slika 32. Prikaz materijala za mogući i nemogući potez figuricom 53
Slika 33. Prikaz materijala za mogući i nemogući potez na polju 54
Slika 34. Prikaz modela ploče za prvu varijantu 54
Slika 35. Prikaz modela strjelica u Blenderu 55
Slika 36. Prikaz gotovog modela ploče za prvu varijantu 55
Slika 37. Prikaz modela ploče prve varijante unutar Godota 56
Slika 38. Prikaz postavki materijala za prvu varijantu ploče 56
Slika 39. Prikaz gotovog modela ploče za prvu varijantu unutar Godota 57
Slika 40. Snimka zaslona PureRef programa s postavljenim referentnim slikama 58
Slika 41. Prikaz modela figurice koji će biti korišten u igri 59
Slika 42. Prikaz modela figurice visoke razine detalja u Blenderu 59
Slika 43. Prikaz rastvorenog modela figurice u Blenderu 60
Slika 44. Prikaz materijala za izradu crne figurice u ArmorPaintu 60
Slika 45. Prikaz slojeva i maske u ArmorPaintu 61
Slika 46. Prikaz postavki čvorova u ArmorPaintu za prvi materijal na crnoj figurici 61
Slika 47. Prikaz postavki čvorova u ArmorPaintu za drugi materijal na crnoj figurici 62
Slika 48. Prikaz gotovih materijala za crnu figuricu u ArmorPaintu 62
Slika 49. Prikaz gotovih materijala za bijelu figuricu u ArmorPaintu 63

Slika 50. Prikaz modela ploče visoke razine detalja u Blenderu 63
Slika 51. Prikaz postavljenih materijala za ploču u ArmorPaintu 64
Slika 52. Prikaz postavki čvorova za prvi materijal ploče u ArmorPaintu 64
Slika 53. Prikaz gotovog modela ploče u ArmorPaintu 65
Slika 54. Prikaz izbornika za postavke materijala u Godotu 66
Slika 55. Prikaz figurica s materijalima u Godotu 67
Slika 56. Prikaz modela ploče s materijalima unutar Godota 67
Slika 57. Prikaz naredbe za usporavanje zvuka u Audacity programu 70
Slika 58. Prikaz izbornika i naredbe za izvoz više datoteka odjednom 71
Slika 59. Prikaz mape sa glazbenim datotekama 71
Slika 60. Prikaz postavljenih čvorova za korisničko sučelje 72
Slika 61. Prikaz postavljenog gumba na glavnoj sceni igre 72
Slika 62. Prikaz postavljene scene prije postupka iscrtavanja 73
Slika 63. Prikaz iscrtane slike u Blenderu 74
Slika 64. Prikaz gotove ikone za aplikaciju 74
Slika 65. Prikaz postavki aplikacije 75
Slika 66. Prikaz postavki za zaslon uređaja 76
Slika 67. Prikaz izbornika za izvoz aplikacije 76
Slika 68. Prikaz mogućih platformi za izvoz u Godotu 77
Slika 69. Prikaz postavki za izvoz aplikacije na Android platformu 77
Slika 70. Prikaz scene unutar igre prije početka kruga 78
Slika 71. Prikaz prve varijante modela u igri 79
Slika 72. Prikaz druge varijante modela u igri 79

Isječak kôda 1. Prikaz definiranih varijabli za glavnu figuricu 19
Isječak kôda 2. Prikaz kôda za aktivni i neaktivni materijal bijele figurice 20
Isječak kôda 3. Prikaz kôda za aktivni i neaktivni materijal crne figurice 21
Isječak kôda 4. Prikaz kôda za pravovaljani i nepravovaljani materijal figurice 21
Isječak kôda 5. Prikaz funkcija koje će se pozivati na emitiranje signala figurice 21
Isječak kôda 6. Prikaz funkcija za reprodukciju zvučnih efekata figurice..... 22
Isječak kôda 7. Prikaz funkcija za izmjenu dvije varijante modela..... 23
Isječak kôda 8. Prikaz definiranih varijabli za glavno polje 24
Isječak kôda 9. Prikaz funkcije koja će se prva pozivati tijekom igre za glavno polje 24
Isječak kôda 10. Prikaz funkcija za glavno polje kada je potez figuricom moguć, odnosno nije moguć 25
Isječak kôda 11. Prikaz funkcije za postavljanje plavog materijala na glavno polje 25
Isječak kôda 12. Prikaz funkcije za isključivanje glavnog polja 25
Isječak kôda 13. Prikaz funkcija koje će se emitirati pritiskom na polje..... 26
Isječak kôda 14. Prikaz definiranih varijabli za scenu s bacanjem kockica 28
Isječak kôda 15. Prikaz funkcije koja će se prva pozivati u skripti za bacanje kockica 28
Isječak kôda 16. Prikaz funkcije pomoću koje će se resetirati tekst s dobivenim vrijednostima kockica 29
Isječak kôda 17. Prikaz funkcije za bacanje kockica 29
Isječak kôda 18. Prikaz funkcija za uključivanje odnosno isključivanje gumba za bacanje kockica..... 29
Isječak kôda 19. Prikaz funkcija za postavljanje teksta s trenutnim igračem na potezu 29
Isječak kôda 20. Prikaz funkcije koja će se pozivati pritiskom na gumb 30
Isječak kôda 21. Prikaz funkcije za reprodukciju zvuka prilikom bacanja kockica 30
Isječak kôda 22. Prikaz definiranih varijabli za glavnu scenu igre 32

Isječak kôda 23. Prikaz funkcije za glavnu scenu koja će se pozivati prva prilikom pokretanja igre 33
Isječak kôda 24. Prikaz funkcije za inicijalizaciju kockica 33
Isječak kôda 25. Prikaz funkcije za inicijalizaciju pozadinske glazbe 33
Isječak kôda 26. Prikaz funkcije za pokretanje reprodukcije glazbe 33
Isječak kôda 27. Prikaz funkcije za aktiviranje bacanja kockica 34
Isječak kôda 28. Prikaz funkcije za odabir igrača 34
Isječak kôda 29. Prikaz funkcije za osvježavanje statusa polja 35
Isječak kôda 30. Prikaz funkcija za mijenjanje aktivnog statusa bijelih figurica 35
Isječak kôda 31. Prikaz funkcija za mijenjanje aktivnog statusa crnih figurica . 36
Isječak kôda 32. Prikaz funkcije za promjenu igrača 36
Isječak kôda 33. Prikaz funkcije za inicijalizaciju svih figurica 37
Isječak kôda 34. Prikaz funkcija za inicijalizaciju bijelih i crnih figurica 37
Isječak kôda 35. Prikaz funkcije za inicijalizaciju polja 37
Isječak kôda 36. Prikaz funkcije za prva četiri početna polja bijele figurice 38
Isječak kôda 37. Prikaz funkcije za prva četiri početna polja crne figurice 38
Isječak kôda 38. Prikaz funkcije za osam središnjih polja 39
Isječak kôda 39. Prikaz drugog dijela funkcije za središnja polja 39
Isječak kôda 40. Prikaz funkcije za dva završna polja bijelih figurica 40
Isječak kôda 41. Prikaz funkcije za dva završna polja crnih figurica 40
Isječak kôda 42. Prikaz funkcije za izlazno polje bijelih figurica 41
Isječak kôda 43. Prikaz funkcije za izlazno polje crnih figurica 41
Isječak kôda 44. Prikaz funkcije koja će se pokretati prilikom dodira figurice ... 42
Isječak kôda 45. Prikaz drugog dijela funkcije za dodir figurice 43
Isječak kôda 46. Prikaz funkcije za emitirani signal piece_released 44
Isječak kôda 47. Prikaz funkcije za ispis statusa polja u konzolu 44
Isječak kôda 48. Prikaz funkcije za ispis statusa figurica u konzolu 45
Isječak kôda 49. Prikaz funkcije za bacanje kockica 45
Isječak kôda 50. Prikaz funkcije za pokretanje nove igre 46
Isječak kôda 51. Prikaz funkcije koja će se pokretati na kraju kruga igre 47
Isječak kôda 52. Prikaz funkcije za animiranje figurica 67

7. LITERATURA

1. *** https://www.blender.org/press/epic-games-supports-blender-foundation-with-1-2-million-epic-megagrant/
   Epic Games supports Blender foundation with $1.2 million Epic Megagrant. (2.8.2020.)
2. *** https://www.blender.org/foundation/history/
   Blender history. (14.8.2020.)
3. *** https://www.oldest.org/entertainment/board-games/
   8 Oldest Board Games in the World. (12.8.2020.)
4. *** https://academy.substance3d.com/courses/the-pbr-guide-part-2
   PBR Guide – part 2. (13.8.2020.)
5. ***https://docs.godotengine.org/en/stable/
   Godot priručnik. (1.8.2020.)
6. Bradfield C. (2018). **Godot Engine Game Development projects: Build five cross-platform 2D and 3D games with Godot 3.0**. Packt Publishing. Birmingham, UK. ISBN-13: 978-1788831505
7. Manzur A. (2018). **Godot Engine Game Development in 24 Hours, Sams Teach Yourself: The Official Guide to Godot 3.0**. Sams Publishing. Carmel, SAD. ISBN-13: 978-0134835099
8. ***http://www.cyningstan.com/game/151/royal-game-of-ur
   Cyningstan – traditional board games, Royal Game of Ur. (3.8.2020.)
9. ***https://www.ancient-origins.net/artifacts-other-artifacts/royal-game-ur-0011202
   Ancient Origins – The Enigmatic Ancient Royal Game of Ur – Will We Ever Understand It? (3.8.2020.)
10. ***https://www.youtube.com/watch?v=WZskjLq040I
    Tom Scott vs Irving Finkel: The Royal Game of Ur | PLAYTHROUGH | International Tabletop Day 2017. (3.8.2020.)
11. ***https://raw.githubusercontent.com/godotengine/godot-docs/master/img/tween_cheatsheet.png
    Godot Tweening CheatSheet. (10.8.2020.)

12. ***https://docs.blender.org/manual/en/latest/
Blender 2.9 priručnik za korištenje. (20.7.2020.)
13. ***https://armorpaint.org/manual.html
ArmorPaint priručnik za korištenje. (22.7.2020.)
14. ***https://manual.audacityteam.org/
Audacity priručnik za korištenje. (25.8.2020.)
15. ***https://inkscape-manuals.readthedocs.io/en/latest/index.html
InkScape priručnik za korištenje. (21.8.2020.)
16. Faust, K. (2018). **Pogon otvorenog kôda za izradu 2D i 3D igara Godot**. Diplomski rad. Rijeka: Sveučilište u Rijeci, Odjel za informatiku.
17. Gal, P. (2019). **Objektno orijentirano programiranje u PHP-u**. Završni rad. Varaždin: Sveučilište Sjever, Odjel za multimediju.
18. ***http://otvorenikod.weebly.com/
Otvoreni kôd. (14.8.2020.)
19. Horvat, A. (2009). **Knjižnice i autorsko pravo.** Aleksandra Horvat, Daniela Živković. Zagreb : Hrvatska sveučilišna naklada, 2009. - 189 str. ; 24 cm. - 89. str.
20. ***https://www.cmswire.com/cms/web-cms/bad-economy-is-good-for-open-source-004187.php
Bad Economy Is Good for Open Source. (17.8.2020.)

8. POPIS MANJE POZNATIH RIJEČI I AKRONIMA

Jezgra igre – vrsta programa koja spaja grafiku, modele, teksture i ostalo te pokreće cijelu igru

Skripta – datoteka koja sadrži programske naredbe

Platforma – u programiranju osnova na kojoj može počivati programski jezik ili uređaj

Inicijalizacija – u programiranju, vraćanje parametara ili procesa na zadane vrijednosti

Petlja – u programiranju, dio programa koji će se izvoditi s ponavljanjima

Varijabla – u programiranju, mjesto pohrane koje je sposobno primiti podatak, taj podatak se kasnije u programu može mijenjati

Obujmica – u programiranju, nevidljivi volumen koji može registrirati sudare

Objektno-orijentirano programiranje – pristup u programiranju računala, putem ove metode aplikacija projektira se kao skup objekata koji vrše komunikaciju između sebe

Klasa – tip podataka definiran od strane korisnika, pomoću tih podataka radi se modeliranje objekata sa sličnim svojstvima

Instanca – jedan primjerak klase

PBR - Fizikalno-definirano iscrtavanje (eng. *PBR* ili *Physically-based rendering*) je nova, trenutno najpopularnija metoda za izradu materijala na modelima; putem ove metode prilikom izrade materijala uzimaju se u obzir fizikalna svojstva tih materijala

randomize() naredba - programu će se dodijeliti sjeme (eng. seed) putem kojega će se dalje moći generirati nasumični brojevi [5] (eng. pseudo-random numbers)

Signali - posebne funkcije u Godot jezgri koje se izvršavaju i prosljeđuju vrijednost ukoliko dođe do emitiranja signala [5]. Kada se određeni signal emitira potrebno je imati dio unutar programa koji osluškuje emitirani signal te aktivira određene funkcije

yield() naredba - pomoću ove naredbe u Godotu može se kôd, koji je trenutno u izvođenju, staviti na privremeno čekanje. Ovisno o definiranom uvjetu unutar naredbe te nakon zadovoljenja tog istog uvjeta kôd će se nastaviti izvršavati [5]

9. PRILOG

Prilog 1 – optički medij – instalacijska datoteka (*.apk*) gotove igre za Android platformu

Prilog 2 – optički medij – aplikacija (*.exe*) gotove igre za Windows platformu

Prilog 3 – optički medij – Godot projekt igre s izvornim kôdom