Alati otvorenog koda za izradu platformske 3D igre

Katić, Josip

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO MATEMATIČKI FAKULTET

ALATI OTVORENOG KODA ZA IZRADU PLTAFORMSKE 3D IGRE

Josip Katić

Rujan 2020

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu

Prirodoslovno-Matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

ALATI OTVORENOG KODA ZA IZRADU PLATFORMSKE 3D IGRE

Josip Katić

SAŽETAK

Jeli moguće alatima otvorenog koda postići rezultat komercijalne razine? Provjera korištenjem softvera 3D računalne grafike i *game engine-a*, pružaju li nam sve potrebne mogućnosti da bi to postigli? Korištenje osnovnih funkcija modeliranja i animiranja kroz softver za računalnu grafiku, te povezivanje logike koristeći *game engine* da bi postigli funkcionalnu računalnu igru. Kao konačni proizvod ispunjavamo ideju postizanja računalne igre predodređenog tipa, te kao zaključak dobivamo pretežno pozitivan odgovor, ali u određenim područjima smatramo da postoji mogućnost napretka.

Ključne riječi: 3D softver računalne grafike, *game engine,* računalna igra

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 36 stranica, 48 grafičkih prikaza i 7 literaturnih navoda. Izvornik je na hrvatskom jeziku

Mentor:

> **Dr.sc. Hrvoje Kalinić,** izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Ocjenjivači:

> **Dr.sc. Hrvoje Kalinić,** izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu
>
> **Dr.sc. Jelena Nakić,** poslijedoktorand Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu
>
> **Dr.sc. Divna Krpan,** viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad prihvaćen: Rujan 2020

Basic documentation card

Thesis

University of Split

Faculty of Science

Department of Computer science

Ruđera Boškovića 33, 21000 Split, Croatia

OPEN SOURCE TOOLS IN 3D PLATFORM GAME DEVELOPMENT

Josip Katić

ABSTRACT

The question of achieving commercial level results with open source tools. To do this we are using 3D computer graphics software and a game engine. Do they offer all the functions to achieve the desired results? Through basic functions of modeling and animating with 3D computer graphics software, as well as using a game engine for logical functions and connections we are aiming for a functional video game. As a final result we are succeeding in the idea of a functional video game of predetermined type and as a conclusion we get a predominantly positive answer, but of course in certain areas there is still a possibility of progress.

Key words: 3D computer graphics software, game engine, video game

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 36 pages, 48 figures i 7 references.

Original language: Croatian

Mentor:

Hrvoje Kalinić, Ph.D. Professor of Faculty of Science, University of Split

Ocjenjivači:

Hrvoje Kalinić, Ph.D. Professor of Faculty of Science, University of Split

Jelena Nakić, Ph.D. Professor of Faculty of Science, University of Split

Divna Krpan, Ph.D. Professor of Faculty of Science, University of Split

Rad prihvaćen: Rujan 2020

Sadržaj

Uvod

Računalne igre su jedan od načina zabave koji postoji već duži niz godina, te kroz vrijeme je zauzimao različite oblike. Od početnih varijacija konzola, velikog perioda arkadnih igri pa sve do današnjeg vrhunca raznolikosti u kojem računalne igre pronalazimo na ogromnom broju uređaja. Zbog neprestanog rasta interesa u temu računalnih igara, sve više se povećala i razina te iste industrije. Dolazimo do perioda kada je industrija računalnih igara jedna od najunosnijih, te imamo sve veći broj ljudi koji žele to iskoristiti izradom i prodajom programa koji pružaju programerima i kreatorima računalnih igara njihovu lakšu izradu. Primjerice to su programi kao 3D softveri za modeliranje i animacije i najbitnije *game engnie-i*, programi koje u većini slučajeva prati i određena novčana cijena.

Kroz ovaj projekt ideja je bila korištenjem takvih programa, ali onih koji su dostupni svima, vidjeti jeli moguće postići željene rezultate. Uz to što su alati koje koristimo u potpunosti besplatni, također su i tipa otvorenog koda gdje članovi zajednice mogu kreirati dodatne funkcije i dijelove za korištenje.

1. Ciljevi i motivacija

Slika 3 Nintendo

Slika 2 Activision Blizzard

Slika 1 Call of Duty

Trenutačno stanje industrije računalnih igara je usporedivo sa svim ostalim većim zabavnim industrijama. Iako se industrije primjerice filma ili glazbe smatraju glamuroznijima, industrija računalnih igara se nalazi na usporedivoj razini uspjeha. Naravno kada govorimo o uspjesima takvih razina pretežno mislimo na velike kompanije kao što su Nintendo ili Activision Blizzard koje posjeduju vlasništvo nad određenim računalnim igrama i serijalima računalnih igara koje u nekim slučajevima nose novčanu vrijednost od nekoliko milijardi dolara (npr. Call of Duty serijal). Uz tu razinu prihoda te kompanije si mogu priuštiti raznolike resurse i alate, od najboljih opcija najosnovnijih alata kao što su softveri 3D računalne grafike i *game engine-a* pa do nekih specifičnih alata i sredstava. Za razliku od kompanija koje se nalaze na vrhu industrije računalnih igara, također se susrećemo sa kreatorima koji se razvojem računalnih igara bave samostalno ili u manjim timovima. To su pretežno *developeri* koji su tek u početku ulaska u industriju računalnih igara ili jednostavno oni koji nisu povezani sa nikakvim firmama te samostalno financiraju i izdaju igre. Igre koji dobivamo od tih *developera* svrstavamo unutar *indie* kategorije jer dolazi od neovisnih (eng. *independent*) *developera*. Kada se radi o takvom tipu *developera* u većini slučajeva se ne koriste najbolja moguća sredstva i alati koji postoje, već samo ono što nam je dostupno. Naravno razlog tomu je što kao neovisni *developeri* nismo u mogućnosti financirati sva moguća i željena sredstva pa se snalazimo sa onim što nam je dostupno. Rješenje tomu, kada je moguće, su besplatni softveri.

Problematika koju nailazimo kod korištenja besplatnih softvera u većini slučajeva je što ti softveri budu znatno slabiji od svojih odgovarajućih komercijalnih verzija. Naravno takva situacija je i očekivana s obzirom da programeri ili možda čak i firme koje ih proizvode nisu sposobne uložiti potpuno radno vrijeme i znatne količine financija za proizvodnju tih softvera, osim u situacijama kada se koriste tipom financiranja iz donacija korisnika. U tim situacijama *developeri* dobivaju određenu slobodu za izradu softvera, ali i dalje se nalaze u situaciji koja nosi svoju određenu nesigurnost.

U djelomičnom smislu kao rješenje tog problema možemo pronaći pri korištenju programa koji su također i otvorenog koda. Kada *developeri* dopuštaju da softveri na kojemu rade bude otvorenog koda sami sebi olakšavaju posao njegove izrade tako da dopuštaju svim njegovim korisnicima i članovima zajednice da uz njih mogu raditi na razvoju tog softvera. Time povećavaju broj „*developera*“ na neograničen broj, a sebi smanjuju količinu potrebnog vremena za rad. *Developeri* o čijem se softveru radi, tada trebaju od svih ponuđenih dodataka vidjeti koji zapravo žele uključiti u glavni program i imati ga kao službeni dio softvera u novom izdanju. Time postižemo, tj. stvaramo mogućnost da softver koji je besplatan, dostupan svima i još uz to dopušta uređivanje i prilagođavanje vlastitim potrebama bude na razini ostalih softvera koji su postali standardi unutar industrije računalnih igara.

Naravno bez obzira što možemo takvo nešto postići, te stvoriti softver visoke razine koji je u isto vrijeme besplatan i otvorenog koda, moramo uvjeriti i korisnike, tj. *developere*, da smo zapravo i uspjeli u tome. Problem koji tu nailazimo je nekakav mentalitet gomile, naročito kada se radi o mladim *developerima*, točnije rečeno onima koji tek počinju sa zanimanjem *developera* računalnih igara. Takvi korisnici kada su u potrazi za svojim prvim softverom koji bi koristili, prvotno traže koji su se koristili za izradu popularnih računanih igara sa kojima su upoznati što su u većini slučajeva igre napravljene od strane kompanija koji ne koriste softvere otvorenog koda. Pri spoznaji da se velika većina računalnih igara proizvodi koristeći iste alate, često se može doći do zaključka da ostali izbori, naročito oni otvorenog koda zbog svoje manje popularnosti, su nevaljani. Možda možemo gledati i na način da će već iskusni *developeri* prestati sa korištenjem trenutnih softvera koji koriste da bi ih zamijenili sa onima koji su besplatni i otvorenog koda, ali u većini slučajeva se događa da je taj proces neisplativ zbog potrebe ponovne edukacije. Svaki zasebni softver, bez obzira koliko bio sličan, ima svoj način funkcioniranja i rada koji se treba naučiti što predstavlja veliki gubitak vremena i time i novca.

Jedan od ciljeva ovog projekta izborom besplatnih softvera otvorenog koda je da pokušamo utvrditi postoji li valjani razlog izbjegavanja alata otvorenog koda ili se uistinu radi samo o mentalitetu gomile, odnosno jesu li ti softveri zapravo na ciljanoj razini kvalitete i snage koja nam je potrebna?

1.1. 3D Platformer

Da bi mogli testirati teoriju o kvaliteti softvera otvorenog koda moramo definirati poligon na kojem ćemo to raditi. Naravno to radimo tako da napravimo nekakva vrsta računalne igre te kroz samu izradu i konačno korištenje konačne računalne igre vidimo rezultate.

Za vrstu računalne igre smo se trebali odlučiti za nešto što može pokazati pretežno sve osnovne funkcije koje očekujemo da ćemo pronaći u nekoj računanoj igri. Funkcije na koje mislimo su naravno vezane za kontrolu likom kojeg dajemo igraču, akcije predodređene za ostale elemente igre koji trebaju reagirati ili ne s obzirom na kontakt sa igračem i obrnuto, te odnos svih elemenata, naročito igrača, sa terenom. Od mogućih ideja odabrali smo žanr 3D platformske računalne igre jer ispunjava sve osnovne funkcije, te uz to je vrsta računalne igre koja je izvediva za napraviti bez pretjerivanja u određenim segmentima.

Primjeri popularnih 3D platformskih igara sa kojima možemo usporediti ciljanu ideju su Super Mario, Spyro the Dragon i Crash Bandicoot. U svim od tih navedenih primjera pronalazimo standardne elemente platformskih računalnih igri kao što su kontrole nad glavnim likom, susretanje sa protivnicima, skupljivi objekti, te naravno *platforming* koji se pojavljuje kroz razine svih igara. Naravno sve od navedenih primjera igara pokrivaju i funkcije koje zahtijevamo i od vlastite računalne igre

2. Dostupni alati na tržištu

Da bi mogli izraditi željeni projekt, tj. računalnu igru, potrebno nam je pronaći sve potrebne alate. Već znamo da alati koje tražimo pripadaju kategorijama 3D softvera za modeliranje i *game engine*, te da bi našli najbolji izbor i onaj koji najviše se slaže sa našim uvjetima trebali smo usporediti nekoliko izbora. Osim softvera koji su besplatni i otvorenog koda, zbog razloga same usporedbe, gledali smo i alate koji ne pružaju otvoren kod, te one koji su djelomično besplatni ili nose fiksnu cijenu.

2.1. 3D softver za računalnu grafiku

Trodimenzionalna računalna grafika temelji se na mnogim istim algoritmima kao dvodimenzionalni računalni vektor u modelu okvira i dvodimenzionalni računalni raster u konačnoj prikazanoj slici. U računalnim grafičkim aplikacijama, 2D programi mogu koristiti 3D metode kako bi proizveli slične učinke (na primjer, osvjetljenje), a 3D može koristiti 2D metode prikazivanja.

Trodimenzionalna grafika se često naziva 3D modeli. Osim prikazane slike, model se nalazi u grafičkoj datoteci podataka. Međutim, postoje razlike: trodimenzionalni model je matematički prikaz trodimenzionalnog objekta. Tehnički nije grafika dok se ne prikaže. Može se vizualno prikazati kao dvodimenzionalna slika kroz proces koji se zove 3D renderiranje, ili se koristi u ne-grafičkim računalnim simulacijama i izračunima. [1]

Izrada 3D računalne grafike podijeljena je u tri glavne faze:

1. 3D modeliranje – proces formiranja računalnog modela u oblik objekta
2. Raspored i Animacije – pozicioniranje i kretanja objekta unutar scene
3. 3D renderiranje – računalni izračuni koji prema pozicioniranju svjetla, vrsti površine i ostalih svojstava generiraju sliku

Slika 4 Renderiranje 3D modela prostora u svrhu prodaje stambene ustanove

[1] 3D Grafika – Što je to?[https://hrv.kagutech.com/4034945-three-dimensional-graphics-what-is-it#menu-1]

Česta primjena 3D računale grafike u današnjem vremenu je u filmovima u obliku CGI-a(*computer-generated-imagery*).

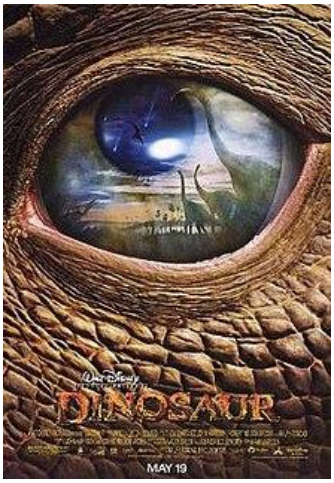
Softveri 3D računalne grafike sa kojima se susrećemo u svrhu pronalaska onoga koji ćemo koristiti unutar ovog projekta su: Maya[2], Houdini[3], Onshape[4] i Blender[5]

2.1.1.Maya

Pretežno se smatra kao standard u industriji računalne grafike, uglavnom zbog ne usporedive količine alata i funkcija. Sve ukupni program je dosta kompliciran za korištenje, ali zbog opširnosti korišten je u većim kompanijama kao Pixar, ILM i DNEG.

Slika 5 Maya

Maya je softver namijenjen generiranju 3D objekata koji se koriste za film, televiziju, računalne igre i arhitekturu. Korisnici definiraju virtualno radno okruženje, tj. scenu, da bi mogli raditi sa i uređivati elemente određenog projekta. Scene možemo spremati u većem broju formata od kojih je zadani .mb (Maya D). Maya koristi arhitekturu grafa čvorova, te svaki element u sceni je baziran na čvoru kod kojeg svaki od tih čvorova posjeduje vlastite atribute i svojstva uređivanja. Kao rezultat toga vizualna reprezentacija scene je u potpunosti zasnovana na mreži međusobno povezanih čvorova koji ovisi o toku informacija koji odašilju jedni drugima. Da bi mogli vidjeti ove mreže pružena nam je opcija grafa ovisnosti i direktnog acikličkog grafa.



Slika 6 Dinosaur - film na kojem se koristila Maya

Operacijski sustavi na kojima možemo koristiti Mayu su: Windows 7 i 10, Apple macOS 10.11.x i poviše, Linux Red Hat Enterprise, Linux 7.3 i 7.5, Linux CentOS 7.3 i 7.5.

Cijena korištenja je 222£/mjesečno.

2.1.2.Houdini

Široko se koristi u VFX industriji za kreiranje raznih 3D prizora. Kao i Maya koristi se čvorovima za upravljanje elementima koje stvaramo i animiramo, dodatno uz klasični način upravljanja sa poligonima. Zbog ovoga Houdini može biti zahtjevniji za korištenje, ali zato postoji verzija software-a za studente i hobiste preko koje se korisnici mogu naučiti prije kupovanja i komercijalnog korištenja.

Slika 7 Houdini

[2] Maya[https://www.autodesk.com/products/maya/overview?support=ADVANCED&plc=MAYA&term=1-YEAR&quantity=1]
[3] Houdini[https://www.sidefx.com/products/houdini/]
[4] Onshape[https://www.onshape.com/]
[5] Blender[https://www.blender.org/]

Houdini Pokriva sva glavna područja 3D produkcije uključujući:

- Modeliranje – svi standardni geometrijski entiteti uključujući poligone, krivulje i meta kugle
- Animiranje – animacija putem ključnih okvira i moguće korištenje hvatanja pokreta
- Čestice
- Dinamika – dinamika krutih tijela, tekućina, žica, simulacije tkanina i gomila
- Osvjetljenje
- Renderiranje – koristi domaći *engine* za renderiranje

Houdini ima otvoreno okruženje i podržava veći broj API-a. Python se pretežno koristi kao jezik za skriptiranje paketa te je ideja da bude zamjena originalnom jeziku za skriptiranje koji je nalikovao na CSell, Hscript.

Operacijski sustavi na kojima možemo koristiti Houdini su: Windows 7 SP1 i poviše, macOS 10.11 i poviše, Linux Ubuntu 14.04+, Linux Debian 8.0+, Linux CentOS 7+, Linux Open SUSE 13.2+, Linux mint 17.3+, Linux Fedora 21+.

Za cijenu korištenja imamo više opcija ovisno o kojoj se verziji radi: 4495$/godina (Houdini FX), 1995$/godina (Core), 269$/godina (Indie).

2.1.3. Onshape

Onshape je softver za koji se smatra da je za moderne agilne timove za dizajn i planiran je kao 3D softver za modeliranje koji je u potpunosti zasnovan na sustavu oblaka. Onshape je prvi i jedini u potpunosti zasnovan na oblaku 3D CAD sistem koji dopušta komuniciranje tima. Može se koristiti putem bilo kojeg web pretraživača, tableta ili pametnog telefona. Smatra se kao idealno rješenje za suradnje otvorenog koda.[6]

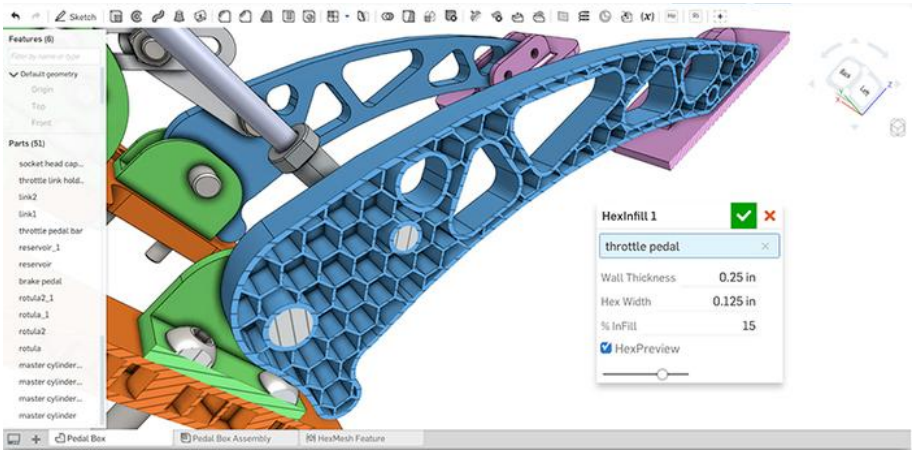
Slika 8 Onshape

Kreiran je na način da ga možemo koristiti bilo kada i bilo gdje u smislu da je namijenjen ekskluzivno za današnje moderne dizajnerske timove u pokretu. Prednosti koje pruža naprema većine je što je u potpunosti zasnovan na oblaku te nema potrebu za ikakvim preuzimanjima i instalacijama. Naravno zbog vrste rada vezanog za oblak također i renderiranje koje nam pruža je povezano za oblakom.

Platforma na kojima ga možemo pokrenuti su: Windows, Mac, Linux, Chromebook, iOS i android.

Cijena softvera u ovom slučaju je nepostojeća jer se radi o besplatnom programu otvorenog koda.

[6] The Best 8 Free and Open Source 3D Modeling Software[https://www.goodfirms.co/blog/best-free-open-source-3d-modeling-software]



Slika 9 Onshape sučelje

2.1.4. Blender

Softver za 3D računalnu grafiku koji se koristi za stvaranje animiranih filmova, vizualnih efekata, umjetnosti, modela za 3D ispisivanje, interaktivnih 3D aplikacija, virtualne stvarnosti i računalnih igrica. Smatra se kao jedan od najboljih izbora za korištenje kod početnika. Uz to već ima jako velik broj korisnika od kojih se također veliki broj posvećuje poboljšanju softvera izradom dodataka.

Slika 10 Blender

Slika 11 Renderiranje koristeći Blender

Pruža nam znatnu količinu funkcija kao što su modeliranje, simulacije, animiranje, renderiranje, te mnoge druge. U modeliranju ima podršku za veći broj geometrijskih osnovnih oblika, uključujući mreže poligona, krivulje, meta kugle kao i još neke primjere. Uz osnovno modeliranje također nam je pružena opcija kiparskog načina rada. Kroz način simulacija možemo simulirati kretanja tekućina, plinova, čestica prašine, kose kao i naravno krutih tijela. U animacijama se koriste ključni okviri, a pružen su nam brojne akcije da bi nam bio olakšan rad s njima. Primjer jedne od akcija je inverzija animacije.

Operacijski sustavi na kojem možemo koristiti Blender su: Windows Vista i poviše, macOS 10.12 i poviše, Linux.

Naravno kako se radi o besplatnom softveru otvorenog koda cijena je nepostojeća.

2.2. Game Engine

Game engine ili game okvir/okruženje je razvojno softversko okruženje dizajnirano za *developere* računalnih igara. *Game engine-i* omogućuju *developerima* izradu računalnih igrica za konzole, osobna računala, pametne telefone i sve slične uređaje.

Osnovne funkcije koje uobičajeno pronalazimo kod *game engine-a* su renderer, odnosno *engine* za renderiranje grafike, za 2D ili 3D grafiku, *engine* koji se brine za zakone fizike, sučelje za programiranje, animacije, zvuk, umjetnu inteligenciju i još neke određene funkcije. Nešto novija funkcija koja se počela pojavljivati kod većeg broja *game engine-a* je vizualno programiranje. Bez potrebe poznavanja ikakvog programiranja i programskih jezika *developerima* je omogućeno stvaranje aplikacija i računalnih igrica. Ovakav princip rada pretežno postoji da bi se početnicima *developerima* olakšala izrada prvih programa, te da bi se isto tako olakšala i ubrzala izrada određenih jednostavnijih programa.

Game engine-i sa kojima se susrećemo u svrhu pronalaska onoga koji ćemo koristiti unutar ovog projekta su: Unreal, Unity, CryEngine i Godot.

2.2.1.Unreal

Jedan od najpopularnijih *game engine-a* trenutno u svijetu, te također nosi titulu najuspješnijeg prema Guinness-u. U usporedbi s drugim *engine-ima* možemo reći da je najbolji kada nam je cilj izrada veće, sofisticiranije igrice za koju treba snažan 3D *game engine*. Naravno zbog snage samog *engine-a* očekivano je da i sami proizvod bude relativno veći, te za njegovo pokretanje potreban je i jači uređaj. Originalno je bio proizveden (1998. godine) u svrhu izrade FPS igara, ali od tada je bio korišten u raznim drugim žanrovima.

Slika 12 Unreal Engine

Pisan je koristeći C++, te osim očitog pružanja skriptiranja podržav animacije, renderiranje, osvjetljenje, korištenje materijala, simulacije, efekte i brojne slične funkcije.

Jezik koji se koristi za skriptiranje je C++.

Platforme za koje možemo stvarati su: Windows, Mac, Linux, iOS, Android, Playstation, Xbox, itd.

Za pitanje cijene Unreal nema standardni način plaćanja već dopušta besplatno korištenje, ali u slučaju da proizvodu koji smo napravili njegovim korištenjem donese zaradu od 3000$ po tromjesečju, plaćamo iznos od 5% ukupnih prihoda.

Slika 13 Killing Floor - igra napravljena koristeći Unreal Engine

2.2.2.Unity

Uz Unreal najkorišteniji *game engine*. Dok je Unreal popularniji kod PC i igrica za konzole, Unitiy je popularniji kod mobilnih igrica te je postao već i glavni izbor za njihovu izradu. Trenutno, 34% od top 1000 besplatnih mobilnih igrica su napravljene koristeći Unity.

Slika 14 Unity Engine

Također je među vodećima u pitanju VR tržišta, korišten je za izradu 90% (procjena) igrica za Samsung Gear VR i 53% za Oculus Rift.

Slika 15 Oculus Rift

Unity dopušta korisniku kreiranje računalnih igri u 2D prostoru kao i u 3D, te *engine* pruža primarni API za skriptiranje u C#. Prije nego se koristio C# kao glavni programski jezik, koristio se Boo i verzija JavaScript-a koja se nazivala UnityScript.

Unutar 2D igrica, dopušteno nam je unošenje slika te korištenje naprednog 2D renderiranja. Za 3D, koristimo se specifikacijom kompresije tekstura, mini mapama, upravljanje postavkama za svaku platformu koju *game engine* podržava, mapiranje refleksija, dinamičke sjene koristeći mapiranje sjena, pretvaranje renderiranja u teksture i efekte za popuni ekran koji dolaze nakon procesiranja.

Jezik za skriptiranje je C#.

Platforme za koje možemo stvarati su: Windows, Mac, iOS, Android, Playstation, Xbox, Windows Phone, itd.

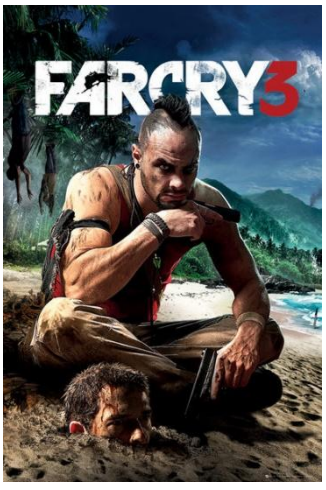
Cijena ovisi o vrsti korištenja. U slučaju da se radi o ne komercijalnom korištenju onda je *game engine* u potpunosti besplatan, ali u suprotnome plaćamo mjesečnu naknadu koja počinje od 35$.

2.2.3. CryEngine

Game engine otvorenog koda sa velikim naglaskom na grafiku i zvuk. Veliki broj alata sa kojim dolazi je orijentiran na ta dva područja, te primjer jednog alata je Fmod. Jedan od najboljih audio alata u području izrade igrica.

Slika 16 CryEngine

CryEngine je napravljen od strane Njemačkih *developera* računalnih igara Crytek zbog vlastitih potreba za *game engine-om.* Primarni razlog kreacije je bila izrada računalne igre Far Cry, te naravno od tada je konstantno obnavljan d bi podržavao sve potrebne novije funkcije i konzole.



Slika 17 Far Cry 3

Neke od funkcija koje posjeduje su: uređivač materijala, sustav čestica u stvarnome vremenu, funkcija za stvaranje vozila, u potpunosti fleksibilni sustav za dan i noć, prirodno osvjetljenje i dinamičke sjene, AI sustav za uređivanje, uređivač animacija lica i brojne druge.

Jezici u kojima imamo opciju programiranja su: C++, Lua i C#.

Platforme za koje možemo stvarati su: iOS, Android, Windows, Linux, Playstation, Xbox i Wii.

Cijena kompletnog *game engine-a* je nepostojeća, tj. CryEngine je besplanat, ali cijena za učlanjivanje počinje od 50$ mjesečno.

2.2.4.Godot

Godot je napredni, multi-platformni 2d i 3d open source game engine. Pruža veliki broj alata sa vizualnim editorom, jednostavnost instalacije i korištenja, te manju zahtjevnost od prethodno navedenih game engine-a. Za pokretanje i korištenje ne trebamo imati toliko jak uređaj kao primjerice za Unreal, te gotovi programi su drastično manje veličine.

Slika 18 Godot Engine

Godot cilja na pružanje popuno integriranog okruženja za *development* računalnih igara. Dopušta izradu svih dijelova igre osim onih vezanih za kreaciju modela ili slika, zvuka i slično. Arhitektura *engine-a* je osnovana na konceptu stabla čvorova, koji su organizirani unutar scena koje možemo koristiti bezbroj puta, te ih možemo primjenjivati i nasljeđivati. Svi resursi, uključujući kod i grafičke elemente, su spremljeni unutra datotečnog sustava računala, a ne u bazama. Ovo rješenje spremanja se koristi zbog lakše suradnje između timova *developera*.

Jezici u kojima možemo programirati su GDScript(velika sličnost Python-u), C#, C++.

Platforme za koje možemo stvarati su: Windows, Mac, Linux, iOS, Android, BlackBerry, HTML5, PlayStation, Nintendo, itd.

Cijena za bilo kakvo korištenje Godot-a je besplatna, te je softver u potpunosti tipa otvorenog koda.

2.3. Odluka

Kod izbora softvera 3D računalne grafike uvjete koje gledamo da treba ispuniti su primarno vezani za modeliranje i animacije. Da bi mogli postići željene rezultate potrebna nam je određena sloboda pri modeliranju objekata koje ćemo koristiti unutar računalne igre i da nakon što smo postigli željeni rezultat taj model možemo animirati na željene načine.

Također i kod izbora za *game engine* trebamo gledati na određene uvjete. Primarni uvjet nam je kompatibilnost *game engine-a* sa vrstom podataka koji stvaramo unutar softvera ta 3D modeliranje, te naravno da se radi o *engine-u* koji podržava izradu 3D računalne igre.

Naravno zajednički uvjet koji trebamo ispuniti za softvere je da se radi o softverima otvorenog koda.

Kao konačnu odluku za softver 3D računalne grafike biramo Blender, dok za *game engine* se odlučujemo za Godot.

Kod izbora koji smo promatrali za izbor softvera 3D računalne grafike imali smo samo dva softvera koji su pružali otvoreni kod. To su naravno Blender i Onshape, dok su Maya i

Houdini softveri koji nam ne pružaju otvoren kod te uz to dolaze uz znatnu novčanu cijenu za korištenje. Nakon toga nam preostaje odluka između Onshape-a i Blender-a. U oba slučaja radi se o besplatnim, ali softverima koji se nalaze na profesionalnoj razini no njihova razlika je u namjeni. Iako u oba slučaja s nam pružene opcije modeliranja elemenata i animiranje, Onshape te funkcije koristi u tehničke svrhe kao što je kreiranje modela za CAD. Zbog takve orijentacije rada ne može nam omogućiti valjane funkcije da bi izradili modele namijenjene za računalnu igru.

Blender kao izbor nam pruža sve osnovne potrebne alate za izradu projekta i još više. Unutar sustava nam pruža veliku slobodu kreiranja i manipulacije modelima, te naknadno njihove animacije. Zbog činjenice da se radi o softveru otvorenog koda sa velikom zajednicom omogućeno nam je korištenje velikog broja priključaka koji nam mogu olakšati testiranje i kreiranje igre. Naravno dio razloga izbora Blendera je i prethodno poznavanje softvera, što smanjuje krivulju učenja naprema softverima sa kojima se tek prvi put susrećemo.

Od izbora koje smo promatrali za *game engine* bili su nam ponuđena dva koja spadaju u kategoriju softvera otvorenog koda. Unreal i Untiy, kao softveri pretežno komercijalne namjere ne pružaju uvid u kod softvera, iako Unity pruža softver ML agenata kao softver otvorenog koda. Na izbor nam preostaju CryEngine, te Godot. Iako se radi o dva jako dobra *game engine-a* kod kojih za oba možemo reći da su dobra izbor jer nam pružaju sve potrebne elemente i alate, za Godot se odlučujemo zbog ponuđenih jezika skriptiranja i zbog jednostavnijeg korištenja.

Godot kao *game engine* nam pruža sve potrebne funkcije kao i kompatibilnost s Blender-om. Na izbor korištenja imamo alate namijenjene kreaciji jednostavnih modela, upravljanju postojećih i kreiranju novih animacija, te sučelje za stvaranje i 2D elemenata uz 3D elemente što nam može olakšati izradu dodatnih funkcija računalne igre.

3. Slični projekti

Da bi mogli bolje vizualizirati ideju računalne igre kakvu želimo postići usporedit ćemo je sa već postojećim projektima, neke od popularnijih što smo već i spomenuli u prethodnom poglavlju (Super Mario, Spyro the Dragon i Crash Bandicoot). No osim što ćemo malo opširnije proći kroz neke od tih popularnijih igara, valjano je spomenuti i slične projekte koji su napravljeni pomoću *game engine-a* koji ćemo i mi koristiti. Ti projekti su demo računalne igre pružene od *developera* Godot *game engine-a* pod nazivima Platformer 3D i Godot Shooter Demo.

3.1. Crash Bandicoot

Jedan u nizu platformera koji smo odabrali kao primjer zbog popularnosti. Zanima nas mogu li se postići slične performanse, efekti i mogućnosti.

Crash Bandicoot je serijal računalnih igara koji je originalno namijenjen za konzolu PlayStation.

Slika 19 Crash Bandicoot

Radi se o vrsti platformske igre sa svim klasičnim elementima. Cilj igre je prolazak određenim redoslijedom kroz razine igre da bi nam se dopustio pristup sljedećim, višim razinama. Kroz svaku od tih razina susrećemo se sa protivnicima koji nas pokušavaju spriječiti u rješavanju te razine, te skupljanju svih potrebnih elemenata. Glavni element koji postoji unutar igre su kristali koji određuju uspješnost rješavanja pojedine razine, te još među ostalim elementima imamo i skupljivo voće koje nam služi za povećanje broja života.

Slika 20 Crash Bandicoot model

Jedini lik sa kojim kao igrač možemo upravljati je glavni lik u igri, tj. Crash Bandicoot. Da bi mogli proći sve razine koje postoje glavom liku su dane određene akcije i kontrole. Osim osnovnih sposobnosti kretanja, tj. hodanja, imamo sposobnosti skoka i napada koje služimo da bi mogli efikasno preći preko platformskih zadataka ili koristiti protiv neprijatelja.

Prikaz igre, tj. kamera, je fiksni. Funkcionira na način da u slučajevima kada su razine dizajnirane u 3D pogledu, pretežno dizajn tunela, gledamo u leđa glavnog lika. Također se pojavljuju razine kod kojih gledamo lika u bučnu stranu, kamera tada ostaje isto pozicionirana ali dizajn razine nas ograničava na način da se pretežno samo možemo koristiti bočnim kretanjem.

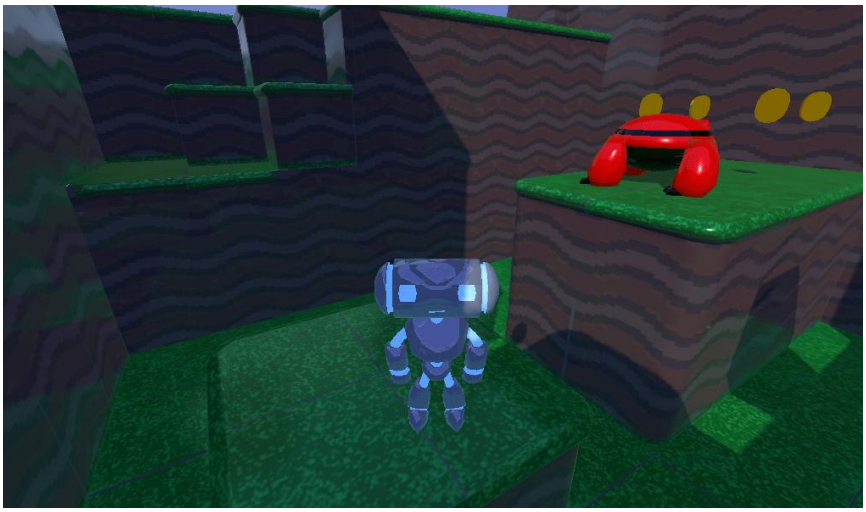
3.2. Godot Platformer 3D

Godot PLatformer 3D je demo[7] primjer računalne igre koji pronalazimo na istoj platformi, te prema njihovom primjeru gledamo možemo li i mi postići slične rezultate.

Pošto se radi o demo izlošku igra nema ugrađenu posebnu priču već se zasniva na prikazu mogućnosti *game engine-a*. Unutar demo igre osim glavnog lika s kojim upravljamo pronalazimo određeni broj neprijatelja te elemenata za skupljanje koji su u obliku novčića.

Akcije koje su pružene glavnom liku su akcije za hodanje, skok te pucanje da bi mogao napasti protivnike.

Za razliku od kamere kod Crash Bandicoot primjera, ovdje imamo situaciju gdje se kamera konstantno pomiče. Točnije rečeno pozicija kamere ovisi o poziciji glavnog lika i njegovom smjeru kretanja. Ima zadane vrijednosti maksimalne i minimalne udaljenosti koliko se smije udaljiti od lika igrača i okreta koje mora pratiti, ali ostatak prepuštamo relativnom položaju naprema modela igrača.



Slika 21 Godot Platformer 3D Demo

3.3. Godot Shooter Demo

Godot Shooter Demo[8], za razliku od prethodnog primjera, nije platformer vrste već se radi o TPS vrsti igre. Shooter demo je također samo primjer mogućnosti Godot *game engine-a*, u smislu da se ne radi o računalnoj igri sa razrađenim razinama i pričom, ali je znatno razvijeniji prikaz od Platformer 3D demo igre u vizualnom smislu. Ovakav prikaz demo igre grafički možemo usporediti sa nekim puno naprednijim potpunim igrama. Također kao i kod prethodnog primjera osim glavnog lika i okruženja nailazimo i na protivnike, ali u ovom slučaju nam se ne pojavljuju elementi za skupljanje što je i za očekivati s obzirom da se ne radi o igri platformer vrste.

Kontrole koje su nam pružene su ponovno osnovne vrste. Dopušteno nam je kretanje u smislu hodanja, skok te naravno akcija pucanja jer se radi o demo igri vrste TPS.

[7] Godot Platformer 3D[https://godotengine.org/asset-library/asset/125]
[8] Godot Shooter Demo[https://github.com/godotengine/tps-demo]

Način funkcioniranja kamere je sličan onome kod 3D Platformer igre u smislu praćenja, ali zbog dodatnog dijela koji zahtjeva TPS igra imamo i mogućnost kontrole kamere pomoću miša. Zbog dodatne veće kontrole akcije pucanja dodana je i kontrola okretanja kamere oko igračeve osi.

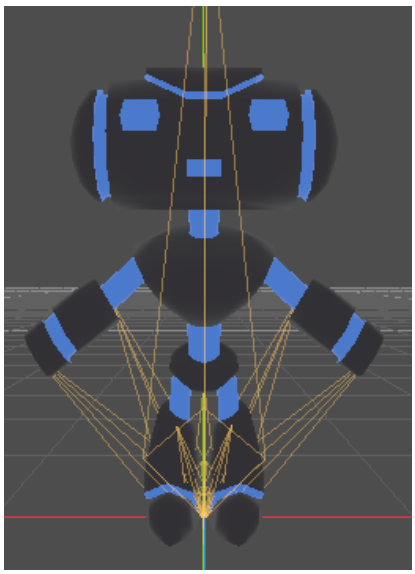
Slika 22 Godot Shooter Demo

4. Ideja konačnog proizvoda

Nakon što smo odredili globalnu ideju o tome što želimo napraviti kao primjer projekta računalne igre (3D platformska računalna igra) i koje alate želimo koristiti za njenu izradu (Blender, Godot), promatramo što sve trebamo uključiti unutar konačnog proizvoda da bi ispunio željene uvjete, te jesu li odabrani alati u mogućnosti to učiniti.

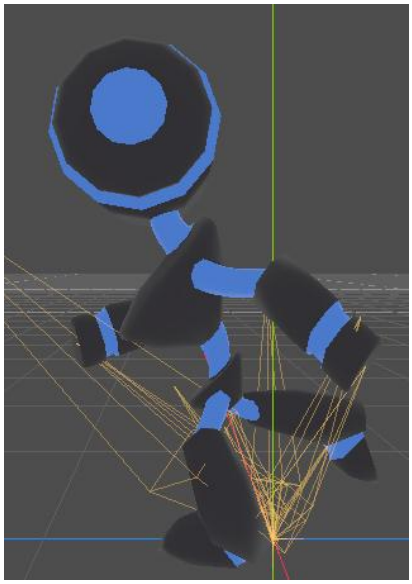
Prvotno se orijentiramo na dio projekta vezan za softver 3D računalne grafike, jer ako želimo stvoriti računalnu igru unutar bilo kojeg *game engine-a* potrebni su nam svi modeli koji će se nalaziti unutar igre.

Kao prvi model možemo odabrati model glavnog lika, tj. igrača. Želja nam je stvoriti model koji će imati sve osnovne karakteristike ljudskog tijela. Pod to mislimo na osnovni izgled, ruke, noge, glava i trup. Na takav oblik glavnog lika ciljamo zbog same praktičnosti i da bi pojednostavnili razumijevanje modela glavnog lika igraču. Primjerice iako se u brojnim igrama ne koristi izričito čovjek kao glavni lik, model koji predstavlja glavnog lika ipak u velikom broju slučajeva ima osnovni čovjekoliki oblik. Primjer tomu su Crash Bandicoot i Bugs Bunny. Nakon što kreiramo taj model potrebne su nam i animacije. Animacijama se služimo da bi igrač imao točan prikaz onoga što se događa. Tako primjerice su nam potrebne animacije vezane za hodanje, skok i po ovisno o akcijama koje koristimo unutar igre i animacija za napad.



Slika 23 Primjer izgleda glavnog lika – Godot Demo

Isti proces koji koristimo za model glavnog lika trebamo koristiti i za ostale modele koji ćemo koristiti, naravno sa razlikom u izgledu modela i vrstom animacija. Za modele neprijatelja ideja nam je stvoriti dva izgleda. Jedan model ćemo koristiti u stilu neprijatelja zemljane vrste, tj. kreće se po tlu, dok drugi zamišljamo kao zračnog neprijatelj, kretanje naravno neovisno o tlu. Pošto planiramo napraviti modele u obliku nekakve vrste vozila kod kojih su pretežno sve strane jednake, za animacije nemamo potrebe raditi više vrsta već samo osnovnu animaciju koja će biti konstantno aktivna. Slično tome ćemo stvoriti i animacije za modele elemenata za skupljanje i prijelaza iz jedne scene u drugu. Pošto ti elementi nemaju potrebu kretanja potrebe su im jedino animacije postojanja.

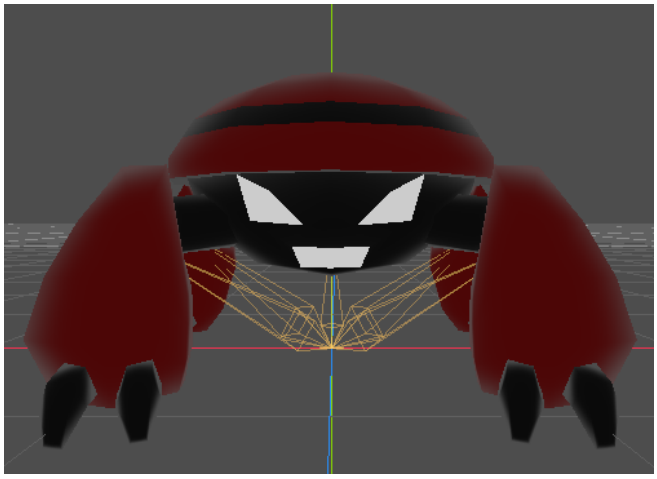


Slika 24 Primjer animacije kretanja glavnog lika - Godot Demo

Nakon što odradimo cijeli proces vezan za kreaciju modela, prelazimo na *game engine*.

Ideja koju trebamo omogućiti unutar *game engine-a* se nadovezuje sa prethodno opisanom idejom modela. Razlog tomu je što nam modeli i njihove animacije opisuju što želimo

postići korištenjem logike *game engine-a*. Animacije koje smo načinili za model glavnog igrača trebamo povezati sa ispravnim akcijama, animacija hoda treba odgovarati akciji hoda i tako i za sve ostale. Naravno osim toga putem *game engine-a* također određujemo međusobne odnose pojedinačnih modela, kako će koji model reagirati u doticaju s igračem i obrnuto.



Slika 25 Primjer izgleda neprijatelja - Godot Demo

5. Konačni proizvod i proces izrade

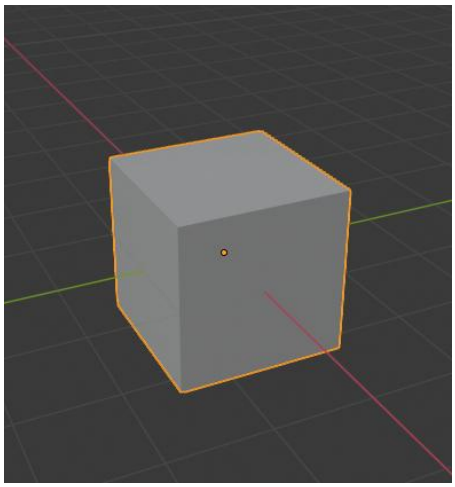
Pod konačni proizvod, tj. računalnu igricu, kao što smo i planirali dobivamo osnovnu platformsku 3D igru, a unutar same igre smo uzeli naziv Block-Head Jumper.

Kroz igru upravljamo karakterom koji ima mogućnosti slobodnog kretanja i skakanja, što su dovoljne akcije i sposobnosti da bi mogli uspješno izvršiti sve tri razine ove igre. Ideja ove računalne igre nam je da koristeći sposobnosti našeg karaktera uspješno pređemo sve priložene razine igre, te na putu od početne pozicije do krajnje pozicije skupljamo određene objekte koji se unutar igrice nazivaju fragmenti. Smatramo da smo uspješno izvršili trenutačnu razinu ako smo uspjeli doći od početne do krajnje lokacije bez umiranja, u suprotnome naš karakter se automatski vraća na početak te moramo ponovno pokušati proći tu razinu. Unutar igrice imamo dva načina na koji možemo uzrokovati gubljenje, tj. smrt. Prvi se slučaj uzrokuje kada ne uspješno odradimo platformski dio igrice te padnemo sa platforme i općenito sa cijelog područja te razine, dok drugi način je uzrokovan susretanjem, tj. kolizijom sa neprijateljem. Naime u slučaju kolizije našeg karaktera sa nekim od neprijatelja koji se nalaze unutar svake razine dolazi do gubljenja.

5.1. Modeli igre i njihova izrada

Svi modeli karaktera, bio to model kojeg koristi igrač, jedan od neprijatelja ili fragment koji skupljamo unutar razina, kao što je rečeno izrađen je koristeći 3D software Blender. Naravno kao što smo već spominjali osim izrade samih modela Blender smo također koristili i sa njihove animacije, te i postavljanje tekstura.

Da bi napravili svaki pojedinačni model svaki put smo započeli sa praznom, novom scenom i sa osnovnim geometrijskim elementom, kao što je kocka, koji već postoji unutar Blendera kao gotovi element. Za svaki od modela smo koristili veći broj primarnih elemenata te im prema želji mijenjali oblik mreže vrhova.

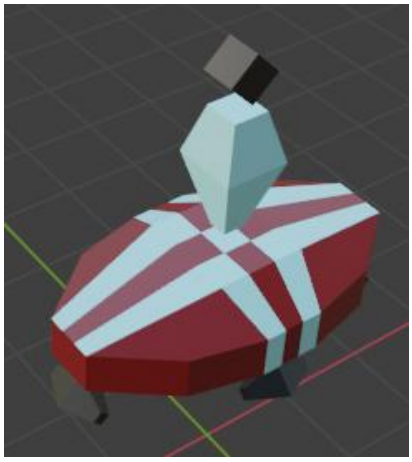


Slika 26 Osnovni element - Blender

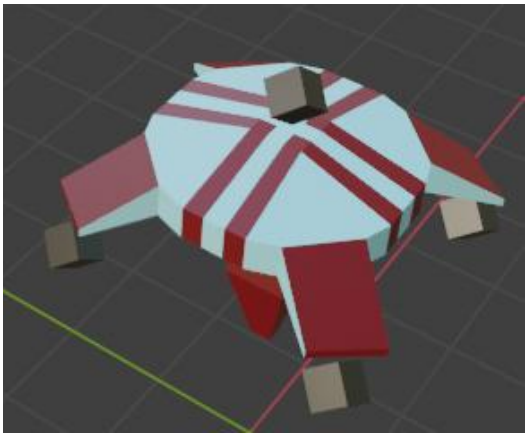
Slika 27 Model glavnog lika

Za izradu modela glavnog lika korišteno je 27 mreža vrhova koje smo oblikovali da bi svaka od njih dobila oblik onome dijelu tijela koje bi trebala predstavljati. Na taj način smo dobili oblik modela koji posjeduje osnovni izgled čovjeka. Stvorili smo pojedinačno elemente ruku, nogu, trupa i glave. Pošto smo željeli da određeni elementi budu jednaki, kao primjerice ruke i noge za koje nismo smjeli imati da je jedna ruka ili noga veća od druge, radili smo kopije već napravljene ruke ili noge te na njih primjenjivali metodu zrcaljenja. Kada imamo jednostavniju vrstu modela, nedostatak simetrije može biti upečatljiv što nam ne bi ulazilo u prilog.

Nakon izrade glavnog lika prelazimo na ostale modele, primarno modele neprijatelja. Za razliku od glavnog lika modeli neprijatelja su jednostavniji i koriste manje elemenat, 7 naprema 27 kojih smo imali kod modela glavnog lika, jer nam se pokazao da možemo postići sve potrebne funkcije i sa manjim brojkama. To nam je bilo izvedivo jer nismo pokušavali kreirati nekakav zahtjevniji oblik kao što je ljudsko tijelo već smo kreirali oblike apstraktnih vozila. Osim neprijatelja još nam preostaju modeli elemenata za sakupljanje, tj. fragmenti, koje ćemo također koristiti kao poveznice među razinama ili portali. Za njih smo odlučili koristiti najosnovnije modele. Sadrže samo 3 elementa, te osim uređivanja njihovog međusobnog odnosa, nismo izvršavali nikakve modulacije nad mrežom.



Slika 28 Neprijatelj tip 1 Model



Slika 29 Neprijatelj tip 2 Model

Nakon što smo odlučili da smo izvršili sve potrebne modele, prelazimo na postavljanje boja i tekstura. U ovom slučaju unutar Blender-a nisu korištene nikakve posebne teksture, već smo koristili svojstva materijala. Unutar Blender-a imamo korištenja materijala koja nam pomaže da bi za svaki element mogli stvarati nekakve, u ovom slučaju jednostavne, teksture ili bolje rečeno boje. Za model igrača i portala koristili smo se samo jednom bojom za bojanje svih zasebnih elemenata modela, ali kod bojanja modela neprijatelja korišteno je više boja i na način da smo unutar jedne mreže elementa postavili tih više boja. Iako imamo mogućnost bojanja na način da cijeli pojedinačni element obojimo od jednom, također možemo i birati koje točno plohe u mreži pojedinog elementa želimo obojiti kojom bojom ili koji materijal želimo staviti. Na taj način u ovom slučaju su postignuti modeli koji koriste veći broj boja.

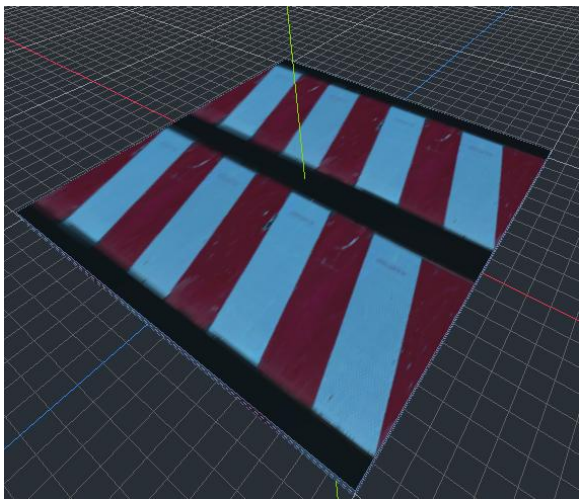
Da bi završili dio rada koji se nalazi unutar Blendera, potrebne su još i animacije. Animacije koje smo trebali izraditi su animacije postojanja, hodanja, trčanja i skoka/pada za model glavnog lika, te animacije postojanja za modele neprijatelja, fragmenta i portala. No prije nego što možemo početi sa samim animiranjem potrebno je odraditi namještanje ili montiranje kostura na predviđeni model. To je proces pri kojemu uzimamo kosti, koje su element unutar Blender softvera, te nakon konstruiranja kostura povezujemo pojedinačne kosti sa odgovarajućim dijelom u modelu. To nam omogućuje jednostavniju i veću kontrolu nad našim modelom pri postupku animacije, te zbog međusobne povezanosti kostiju ne trebamo se brinuti o neželjenim pomicanjima dijelova modela. S tim prelazimo na proces animiranja. Animacije kao i u bilo kojem drugom procesu animiranja činimo koristeći se ključnim okvirima/scenama. Koristeći se prethodno postavljenim kosturom namještamo model u željenu pozu, koju tada pretvaramo u ključni okvir na način da unutar vremenske linije u Blender-u pamtimo svojstva kostura. U željenom vremenu, tj. okviru, spremamo podatke o lokaciji, rotaciji i veličini svake kosti unutar kostura, te onda prelazimo na izradu sljedećeg ključnog okvira. Nakon što su izrađeni potrebni ključni okviri dobivamo animaciju na način što Blender izvodi povezivanje tih okvira tako da imamo kretanje svake kosti od početne željene pozicije pa sve do krajnje. U slučaju da smatramo da u animaciji prijelaz koji je Blender izveo nije u potpunosti primjeren i nije onakav kakav smo zamislili, unosimo još određeni broj ključnih okvira između već postojećih da bi animacija bila preciznija i ljepša.

Slika 31 Animacija trčanja za glavnog lika

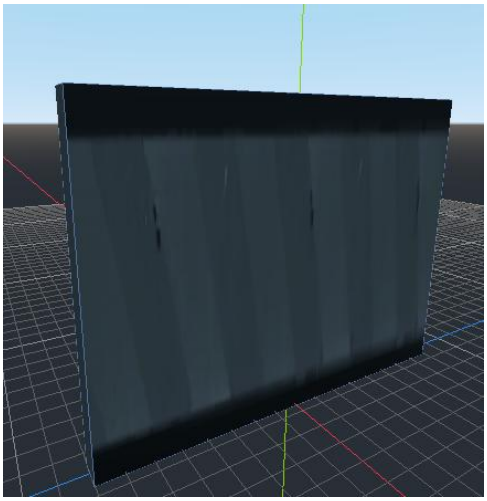
Slika 30 Animacija pada glavnog lika

Prije nego pređemo trebamo spomenuti da za izradu okruženja, tj. izradu razina, nismo koristili samo modele napravljene unutar Blender-a. Odnosno koristeći se Godot napravili smo osnovne elemente koje koristimo za sveopće okruženje. Koristeći se sličnim vrstama alata koji postoje unutar Blender-a izrađujemo potrebne modele, ali zbog puno veće ograničenosti tih alata koristimo ih za izradu elemenata osnovnog oblika. Kao i kod procesa koji smo imali u Blenderu potrebno nam je koristiti nekakvu teksturu ili boju, a u ovom slučaju umjesto da sami bojimo koristimo se već gotovim teksturama koje samo povezujemo sa pojedinim elementima te tako dobivamo željene rezultate. Elemente koje smo ovdje

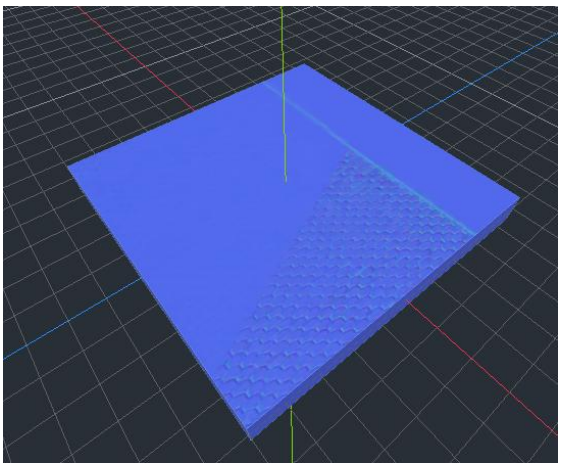
napravili koristimo za izradu tla, zidova i platformi. Nakon izrade okruženja u kojem ćemo se kretati napokon počinjemo dodavati i elemente koje smo napravili koristeći Blender. Kada smo uveli sve potrebne elemente javlja nam se i potreba izrade određene količine animacija za neke od njih, primjerice za platforme koje bi se trebale kretati i naravno za neprijatelje koji uz osnovnu animaciju koju smo napravili koristeći Blender također trebamo i koristiti animaciju da bi se kretali u trenutno stvorenom prostoru. Pošto nam nisu potrebne kompliciranе animacije Godot-ov sustav za animacije je više nego dovoljan pružajući animiranje promjena pozicije po osima x, y, z te također i rotacije po istim osima, a funkcionira naravno na isti način kao i kod Blendera tako da se koristimo ključnim okvirima.



Slika 32 Model tla



Slika 33 Model zida

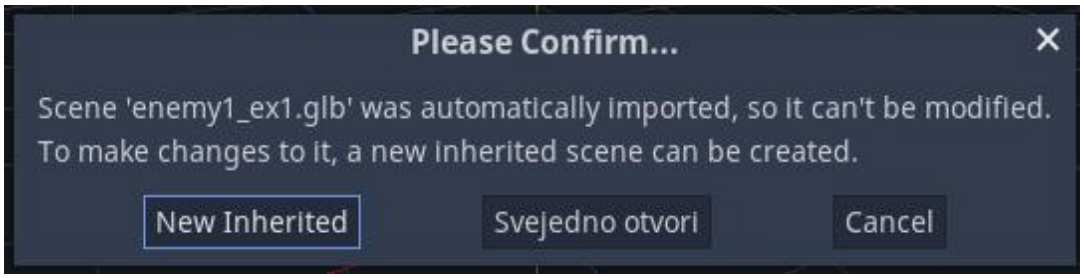


Slika 34 Model platforme

5.2. Logika igre i njenih elemenata

Nakon što smo završili sa dijelom igre vezanim za kreaciju potrebnih elemenata kao što su modeli glavnog lika ili neprijatelja, trebamo te modele uvesti u Godot te povezati logiku njihovog ponašanja.

Da bi sve elemente iz Blender-a mogli unijeti u Godot, prvo trebamo napraviti izvoz iz Blendera što činimo tako da te elemente pretvaramo vrstu datoteke čitljiv unutar Godot-a, u ovom slučaju e radi o glTF 2.0 vrsti datoteke. Nakon što smo dobili takvu vrstu datoteke prebacujemo ga u mapu u kojoj se nalazi naš projekt te ga unutar Godot-a pokrećemo na način da ga otvaramo u zasebnoj sceni gdje nad njim možemo izvoditi željene akcije. Naravno pri pokretanju dobivamo model koji smo izradili unutar Blendera sa svim svojstvima izgleda, tekstura i animacija.



Slika 35 Pokretanje Blender modela unutar Godot-a

Najbitniji dio izrade projekta za koji koristimo *game engine* je dio vezan za programiranje. Glavni razlog njegovog korištenja je da bi nam olakšao korištenje funkcija kao što su manipulacija scenama, određivanje kolizije, te upravljanje likom igrača. Razlog tomu je što svaki *game engine* sadrži gotove funkcije koje su namijenjene velikom dijelu čestih akcija i provjera.

Kretanje lika igrača omogućavamo na način da pri slanju ulaznog signala od strne igrača njegov model pomičemo po odgovarajućoj osi. Povećavanjem ili smanjivanjem vrijednosti osi x ili osi z model igrača se kreće u četiri osnovna smjera. Također uz osnovno kretanje trebamo i stvoriti opciju skoka u kojoj trebamo pomicati model igrača prema „gore“, u ovom slučaju tu se radi o osi y, pa nakon dostizanja određene visine prepustiti pad igrača gravitaciji. Naravno element gravitacije nam je konstantno prisutan na način da na model igrača po osi y djeluje negativnom vrijednosti, a u slučaju skoka taj efekt je ignoriran da bi se na jedan kratki period vremena usmjerenje igrača po osi y povećalo te bi on mogao skočiti.

```
if Input.is_action_pressed("move_forward"):
    direction += -cam.basis.z
    dir += -cam.basis.x
    walk = true
if Input.is_action_pressed("move_backward"):
    direction += cam.basis.z
    dir += cam.basis.x
    walk = true
if Input.is_action_pressed("move_right"):
    direction += cam.basis.x
    dir += -cam.basis.z
    walk = true
if Input.is_action_pressed("move_left"):
    direction += -cam.basis.x
    dir += cam.basis.z
    walk = true
```

```
if is_on_floor() and Input.is_action_pressed("jump"):
    g_direction.y = 30
```

Slika 36 Naredbe za upravljanje kretanjem i skokom glavnog lika

Uz logiku kretanja igrača također vežemo i logiku kamere, točnije rečeno njenog kretanja. Za to smo zapravo pronašli rješenje u prethodno spomenutom Godot demo projektu, Godot Platformer 3D, od kojeg uzimamo čitavu logiku kamere te je primjenjujemo na vlastitom projektu. Uz određenih kalibracija vezanih za udaljenosti kamere i brzine njenog kretanja, ponovno imamo kameru koja prati igrača te se okreće relativno po njegovoj putanji.

```
export var min_distance = 10
export var max_distance = 20
export var angle_v_adjust = 0.0
export var autoturn_ray_aperture = 25
export var autoturn_speed = 50
```

Slika 37 Varijable za određivanje udaljenosti i brzine kamere

Uz sve navedene akcije također treba povezati i animacije koje smo stvorili unutar Blender-a ili Godot-a. Takvo nešto nam ne stvara problem ako se radi o elementima koji imaju samo jednu animaciju, ali kod elementa glavnog igrača koji ih ima više trebamo postaviti dodatnu logiku. Koristeći se uvjetima koji provjeravaju u kojem je trenutno stanju akcije model igrača odlučujemo koja nam je animacija potrebna. U slučaju ne djelovanja i stanja mirovanja osnovna animacija će biti pokrenuta i to od aktivacije igre. No u suprotnom slučaju postavljamo vrijednost, naziv, animacije na željenu animaciju te pozivamo dio *game engine-a* koji je odgovoran za pokretanje animacija. Kada odredimo koja nam je animacija potrebna pozivamo se na funkciju za njenu aktivaciju unutar pokretača animacija, te nastavljamo sa kodom.

```
#animacije
var anim_to_play = "idle-loop"
if walk == true:
    anim_to_play = "walk-loop"
if run == true and walk == true:
    anim_to_play = "run-loop"
if !is_on_floor():
    anim_to_play = "fall-loop"

var t_anim = anim_player.get_current_animation()
if t_anim != anim_to_play:
    anim_player.play(anim_to_play)
```

Slika 38 Logika animacija za glavnog lika

Ostatak koda koji nam se pojavljuje unutar igre, koji nije direktno vezan za glavnog lika ili kameru, se odnosi na protivnike, elemente za skupljanje, portale te na izbornik. Za razliku od koda povezanim sa glavnim likom, ovdje se radi o drastično manjoj količini koji se i kod većine elemenata ponavlja. Najčešći od njih je vezan za logiku promjene scene. Primjerice kod korištenja portala kada se trebamo kretati iz glavne scene u jednu od željenih razina potrebno nam je pozvati metodu change_scene() te navesti lokaciju/scenu koju želimo pokrenuti.

```
func _on_Area_body_entered(body):
    get_tree().change_scene("res://Level1.tscn")
    pass # Replace with function body.
```

Slika 39 Logika promjene scene

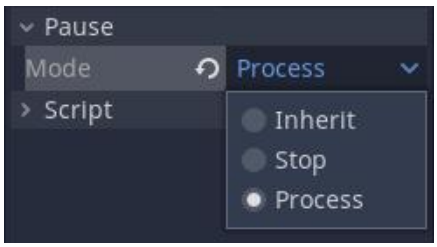
To funkcionira na način da pozivamo stablo koje sadrži sve scene unutar programa odmah pri pokretanju. Nakon što imamo pozvano stablo pružen nam je pristup svim scenama, te onda pozivamo funkciju change_scene() preko koje biramo jednu od postojećih scena iz stabla.

Takav princip rada se koristi i kod funkcija koje pronalazimo na botunima na izborniku, ali bitniji dio koda kod tog dijela je vezan za pauziranje programa i njegovo ponovno pokretanje. Što je zapravo bitno kod tog dijela je kada zovemo funkciju za pauziranje pauziramo cijeli program, pa za određene dijelove koje želimo tada koristiti moramo posebno namjestiti da se ne pauziraju u bilo kojem uvjetu. U slučaju kada to ne bih napravili mogli bismo završiti u slučaju kada bi program (igra) u potpunosti bio pauziram i ne bi ga mogli ponovno pokrenuti.

```
func _input(event):
    if event.is_action_pressed("ui_cancel"):
        get_tree().paused = true
        $ColorRect.visible = true
        Input.set_mouse_mode(!Input.MOUSE_MODE_CAPTURED)

func _on_Resume_pressed():
    get_tree().paused = false
    $ColorRect.visible = false
```

Slika 41 Logika privremenog zaustavljanja programa



Slika 40Svojstvo nasljeđivanja - Mode

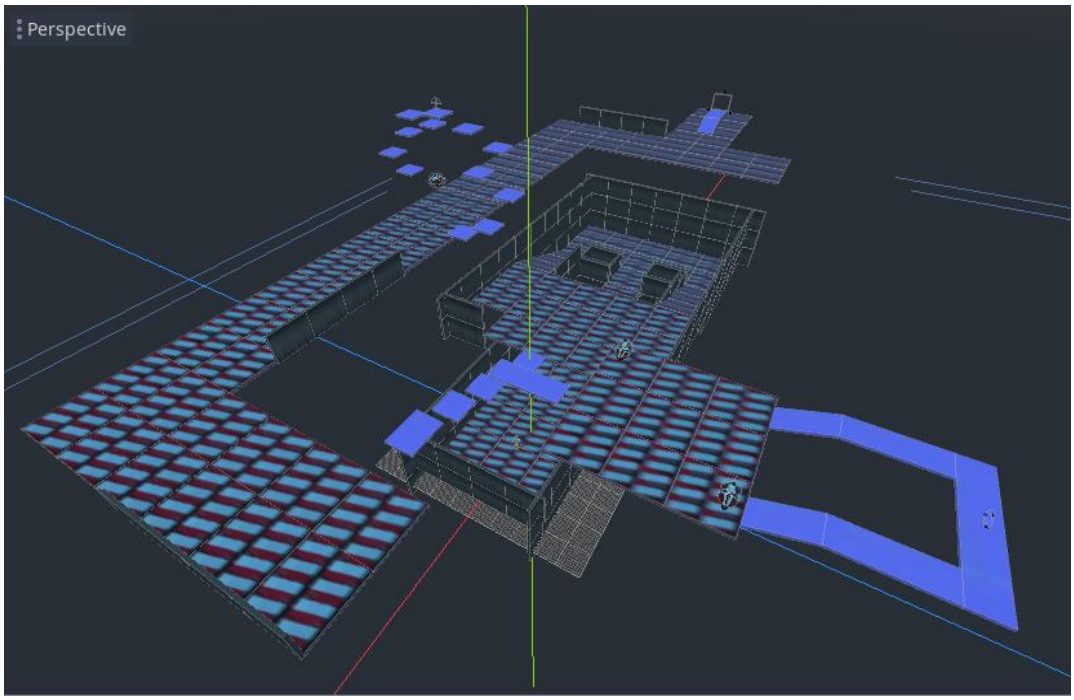
Također bitno je za spomenuti da scene koje su korištene za izradu izbornika nisu 3D, već se koristimo 2D scenama koje su onda povezane sa pripadajućim 3D scenama na kojima ih želimo pokrenuti. Izbornik je izrađen zasebno sa svim potrebnim elementima, te ubačen u odgovarajuću 3D scenu, ali je sakriven sve do trenutka kada ga aktiviramo. Jedina razlika među postojećim izbornicima unutra igre je startni izbornik koji funkcionira kao vlastita scena, te je prvo što nam se prikaže pri pokretanju programa.

5.3. Razine igre

Da bi bilo koja računalna igra mogla imati smisla, treba imati cilj, koji smo prethodno već i naveli. A da bi se unutra ove igre ili bilo koje druge platformske igre taj cilj mogao ispuniti trebaju postojati određene razine na kojima ćemo ih izvoditi.

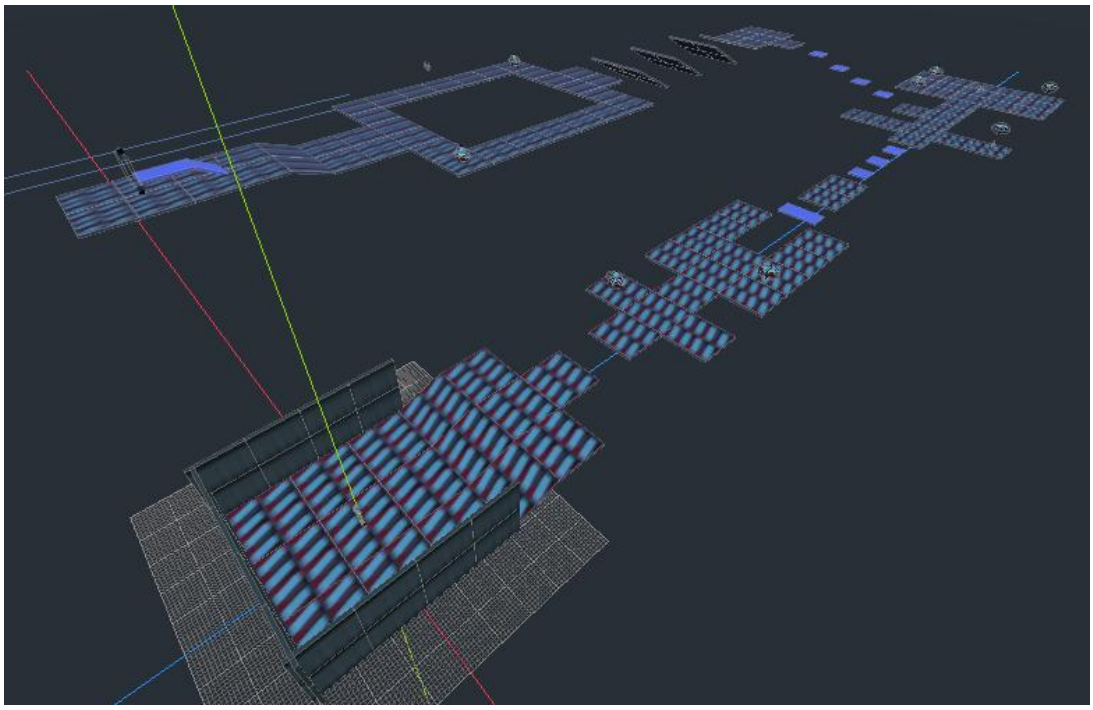
Unutar naše računalne igre postoje tri razine kroz koje možemo prolaziti. Sve od njih u osnovnom smislu su iste, na način da se susrećemo sa protivnicima i platformama, ali svaka od njih ima zasebne razlike.

Dizan prve razine je postavljen da bi bio najlakše izvediv. Na čitavoj prvoj razini ne susrećemo se pomičnim platformama, te broj protivnika je minimalan. Uz to veći dio prve razine je okružen zidovima koji onemogućuju pad i u dizajniran je u principu kruženja.



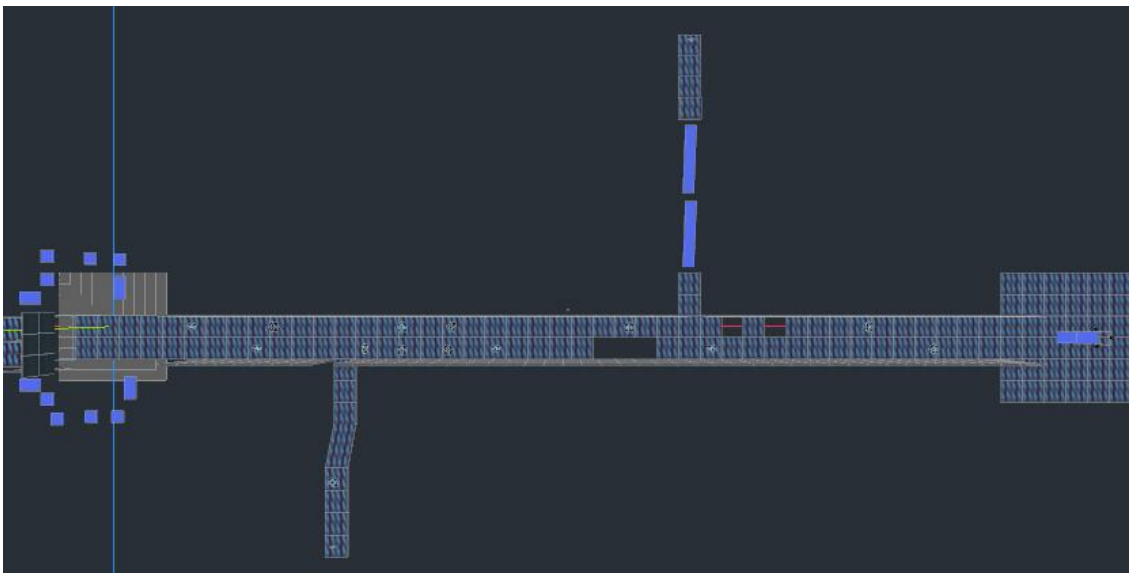
Slika 42 Prikaz Razine 1(Godot game engine)

Kod druge razine iako se također ne pojavljuju pokretne platforme, susrećemo se sa platformama otežane vrste za razliku od prve razine. Također kada prolazimo kroz drugu razinu prisiljeni smo se koristiti funkcijom trčanja da bi uspjeli izvršiti određene prijelaze. Naravno susrećemo se i sa povećanim brojem neprijatelja koji nam otežavaju prolaz do cilja i sam dizajn je u izgledu slova U.



Slika 43 Prikaz Razine 2(Godot game engine)

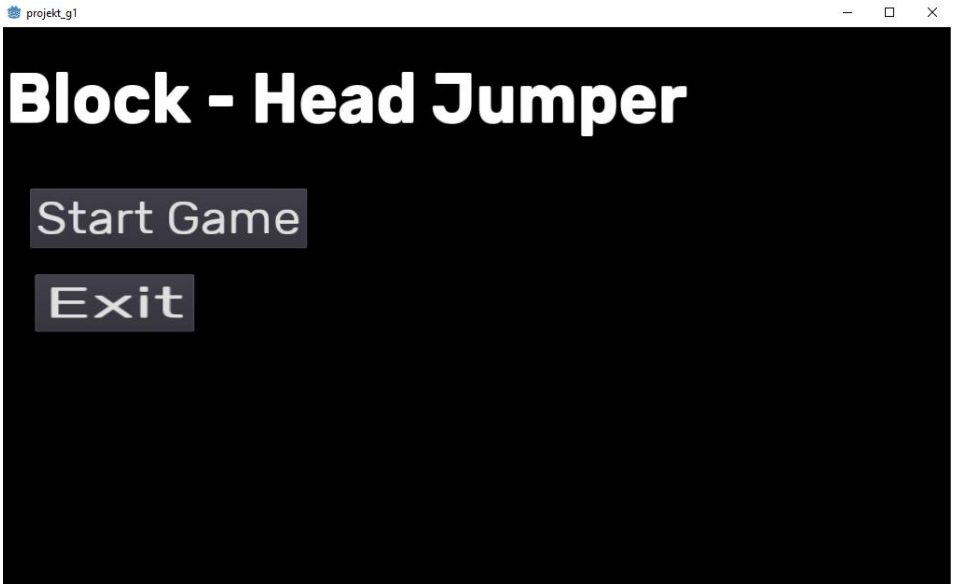
Treća razina za razliku od prethodne dvije ima najjednostavniji dizajn u kojemu imamo samo gibanje po pravcu. No bez obzira na to napokon nam se pojavljuju pomične platforme od više vrsta, te kroz tunel pravca kroz koji prolazimo se susrećemo sa znatno većim brojem neprijatelja.



Slika 44 Prikaz Razine 3(Godot game engine)

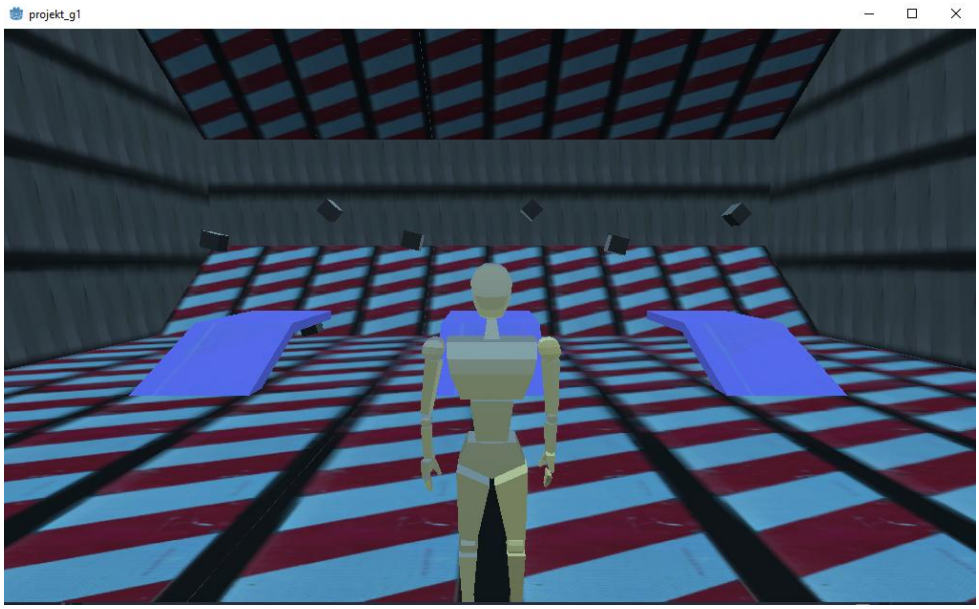
6. Upute korištenja

Pri pokretanju računalne igre početni prizor nam je početni izbornik na kojem se nalazu dva botuna, jedan od njih za gašenje igrice dok je drugi za pokretanje. Način izbora je jednostavnim klikom miša na jednog od njih. Ako izaberemo da ćemo nastaviti i pokrenemo igricu ulazimo u način rada u kojem nam je miš isključen jer nam nije potreban na korištenje.



Slika 45 Početni izbornik

Tada ulazimo u glavnu scenu i dobivamo kontrolu nad modelom igrača kojeg kontroliramo sa klasičnim izborom tipki. Kada se želimo normalno kretati koristimo se tipkama W, A, S i D kod kojih svaka od tipki odgovara određenom usmjerenju. W kao kretanje za unaprijed, A kretanje prema lijevo, S za okretanje i kretanje prema kameri, te D za kretanje prema desno. Naravno svi od tih smjerova su relativno određeni prema položaju kamere i modela lika. Uz to ako želimo pokrenuti akciju trčanja uz već navedene tipke potrebno je držati tipku Shift, a u slučaju skoka koristimo se tipkom Space.



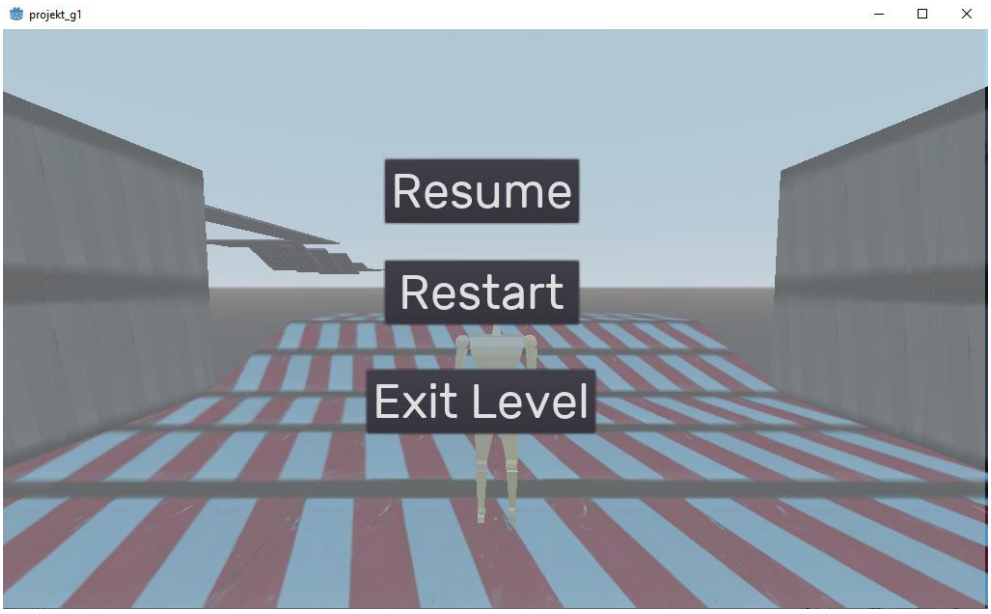
Slika 46 Početna scena

Za pokretanje pauze i otvaranje izbornika koristimo tipku Esc, te time također oslobađamo i miš jer ponovno koristimo izbornik sa botunia koje odabiremo klikom miša. Unutar igre ovisno gdje se nalazimo prikazuju nam se dvije vrste izbornika, oba različita od početnog.

Prvi izbornik nam se prikazuje ako se nalazimo unutar početne scene i samo nam nudi opcije izlaska i nastavljanja igre.

Slika 47 Izbornik početne scene

Dok drugi izbornik se pojavljuje kada se nalazimo u jednoj od ponuđenih razina. Za razliku od izbornika iz početne scene ponuđen nam je izbor vraćanja na početak trenutne razine u slučaju ako želimo je probati proći iz početka.



Slika 48 Izbornik za vrijeme pokrenute razine

7. Zaključak

Nakon izvoda cijelog projekta možemo postaviti dva pitanja. Jesmo li uspjeli ispuniti sve uvjete koje smo planirali i jeli zapravo ima smisla korištenje softvere otvorenog koda.

Glavi plan koji smo imali za ovaj projekt je bila izrada jednostavne 3D platformske računalne igre koristeći alate otvorenog koda. Da bi to napravili bilo nam je potrebno kreirati sve potrebne elemente, tj. sve modele koji će se pojavljivati unutar naše računalne igre. Naravno to su bili modeli glavnog igrača, neprijatelja i slično. Nakon toga krenuli smo sa sveukupnim povezivanjem elemenata unutar 3D okruženja uz programiranje logike da bi odredili kako bi se ti elementi trebali ponašati.

Kroz konačni rezultat možemo reći da se osnovna ideja projekta uspjela ispuniti, no naravno ne u potpunosti bez problema.

Sa strane 3D softvera za računalnu grafiku Blender, nisu se pojavljivali problemi. Softver se pokazao jako jednostavnim i sa velikom potporom zajednice jednostavnost rješavanja problema je trenutačna. Uz snagu i broj alata koji nosi sa sobom, možemo reći da se s razlogom počeo koristiti i u profesionalne svrhe, te zamjenjivati trenutačne standarde u svijetu softvera za 3D grafiku.

Sa strane *game engine-a* Godot, ne možemo reći da su postojali problemi, ali s usporedbom sa određenim *game engine-ima* koji postoje u području 3D računalnih igara postoji manje podataka. To se najviše osjetilo pri kreaciji kontrolora za glavnog igrača. Pošto se radi o kontroloru vrste gdje je kamera pozicionirana u vrsti treće osobe, što je znatno zahtjevnije od kontrolora za prvo lice, bilo je teže pronaći primjere osim onih koji su ponuđeni od strane Godot-a. Usporedbom sa *game engine-om* Unity, koji postoji puno duže vremena, gdje za takvo nešto već postoji ugrađeno rješenje. Naravno takvo nešto ne ograničava kvalitetu *game engine-a* Godot, ali pokazuje kako već jako dobar *game engine* koji ima otvoreni kod sadrži područja čijim napretkom može postati još bolji.

Konačno pitanje, koristiti softver otvorenog koda ili ne?

Za sve one koji tek počinju ulaziti u zanimanje *developmenta* računalnih igara softveri koji su besplatni, te posebice otvorenog koda, predstavljaju jedinstvenu priliku. Ne trebaju se brinuti za ograničenja koja nameću programi i firme koje proizvode profesionalne softvere, a i dalje mogu kreirati računalne igre kakve su zamislili.

Iako za pitanje besplatnih softvera otvorenog koda i dalje postoji stigma manje kvalitete, softveri kao što su Blender ili Godot to mogu zanijekati. Te naravno na pitanje postoji li nekakav napredni projekt napravljen koristeći se takvim alatima, ubrzo će biti da. Naravno dokaz tome je što softver kao što je Blender već počinje biti korišten kod ozbiljnijih firmi u svrhu izrada animacija, animiranih filmova ili CGI efekata za filmove. Iako neki od takvih softvera imaju prostor za napredak, imaju svijetlu budućnost u korištenju i preuzimanju tržišta. U kratko odgovor je da, ima smisla koristiti se besplatnim softverima otvorenog koda.

8. Literatura/Izvori

3D grafika - što je to? - Informacijska tehnologija – 2020 - https://hrv.kagutech.com/4034945-three-dimensional-graphics-what-is-it

3D computer graphics - https://en.wikipedia.org/wiki/3D_computer_graphics

The best 3D modelling software in 2020 - https://www.creativebloq.com/features/best-3d-modelling-software

The Best 8 Free and Open Source 3D Modeling Software - https://www.goodfirms.co/blog/best-free-open-source-3d-modeling-software

CC0 Textures - https://cc0textures.com/view?id=Sticker001

Why choose the Godot Game Engine over Unity or Unreal Engine - https://gamefromscratch.com/why-choose-the-godot-game-engine-over-unity-or-unreal-engine/

Game engine - https://en.wikipedia.org/wiki/Game_engine

Godot 3.0 3D Platformer Demo - https://www.youtube.com/watch?v=aRwCxMYSIk8

Godot Shooter Demo - https://www.youtube.com/watch?v=3sPFBmnfEEM

Crash Bandicoot - https://en.wikipedia.org/wiki/Crash_Bandicoot

Crash Bandicoot Ps1 Gameplay - https://www.youtube.com/watch?v=HWOApx8F0V4

9. Slike

10. Sažetak

Alati otvorenog koda za izradu platformske 3D igre

Jeli moguće alatima otvorenog koda postići rezultat komercijalne razine? Provjera korištenjem softvera 3D računalne grafike i *game engine-a*, pružaju li nam sve potrebne mogućnosti da bi to postigli? Korištenje osnovnih funkcija modeliranja i animiranja kroz softver za računalnu grafiku, te povezivanje logike koristeći *game engine* da bi postigli funkcionalnu računalnu igru. Kao konačni proizvod ispunjavamo ideju postizanja računalne igre predodređenog tipa, te kao zaključak dobivamo pretežno pozitivan odgovor, ali u određenim područjima smatramo da postoji mogućnost napretka.

Ključne riječi: 3D softver računalne grafike, *game engine,* računalna igra

11. Summary

Open source tools in 3D platform game development

The question of achieving commercial level results with open source tools. To do this we are using 3D computer graphics software and a game engine. Do they offer all the functions to achieve the desired results? Through basic functions of modeling and animating with 3D computer graphics software, as well as using a game engine for logical functions and connections we are aiming for a functional video game. As a final result we are succeeding in the idea of a functional video game of predetermined type and as a conclusion we get a predominantly positive answer, but of course in certain areas there is still a possibility of progress.

Key words: 3D computer graphics software, game engine, video game

12. Privitak

Instalacija potrebnih alata

Za izradu projekta trebali smo imati samo dva alata, Blender i Godot.

Instalacija Blender-a funkcionira kao i većine drugih softvera. Prvi korak nam je preko službene Blender web stranice preuzeti verziju Blender-a koja nam je potrebna, točnije rečeno instalacijsku datoteku, te kada je imamo na vlastitom računalu pokrenuti te odraditi sav proces instalacije kroz koji nas navodi. Verzija Blender-a koju smo u ovom projektu koristili je Blender 2.82.

Za razliku od Blender-a, Godot nema potrebnu instalaciju, ali kao i u prethodnom slučaju prvo što činimo je da pronalazimo na službenoj Godot web stranici verziju softvera koji nam je potreban. Nakon preuzimanja dobivamo .rar tip datoteke koji trebamo raspakirati, te nam se tu pojavljuje Godot datoteka preko kojeg ga automatski pokrećemo. Verzija korištea u svrhe ovog projekta je Godot 3.2.