Razvoj računalne igre "Essentia“

Cerovac, Matija

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:603860

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni Diplomski studij informatike

Matija Cerovac

Razvoj računalne igre „Essentia“

Diplomski rad

Mentor: izv. prof. dr. sc. Miran Pobar

Rijeka, lipanj 2024.

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija

Rijeka, 2.5.2024.

Zadatak za diplomski rad

Pristupnik/ica: Matija Cerovac

Naziv diplomskog rada: Razvoj računalne igre Essentia

Naziv diplomskog rada na eng. jeziku: Development of the computer game „Essentia"

Sadržaj zadatka: Proučiti i opisati faze razvoja računalnih igara. Prikazati dokument dizajna vlastite računalne igre i detaljnije prikazati proces razvoja scenarija. Opisati implementaciju igre s naglaskom na izradi asseta za igru te umjetne inteligencije likova. Opisati korištene alate i tehnologije.

Mentor:
Izv. prof. dr. sc. Miran Pobar

Voditeljica za diplomske radove:
Doc. dr. sc. Lucia Načinović Prskalo

Komentor/ica:

Zadatak preuzet: 2.5.2024.

(potpis pristupnika/ce)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR1524020061400006966

Sažetak

U ovom diplomskom radu bit će objašnjen i demonstriran potpun proces razvoja videoigre imena „Essentia“. Igra je rađena u Unreal Engine-u 5, točnije u verziji 5.1. Tijekom ovog rada dotaknuti ćemo se svakog aspekta razvoja igre, s tim da će veći fokus biti na dijelovima na kojima sam osobno radio. Ti dijelovi uključuju pisanje scenarija, online marketing, stvaranje 3D modela u programu Blender te programiranje umjetne inteligencije unutar samog Unreal Engine-a korištenjem sustava vizualnog skriptiranja pod nazivom blueprints.

Ključne riječi: Videoigra, Essentia, Unreal Engine, Blender, Umjetna Inteligencija, 3D Objekti, 3D Mape, Blueprints

Sadržaj

1. Uvod

Videoigre su danas dostupne u velikim količinama i velikom broju ljudi. One su gotovo zamijenile televiziju kao glavni izvor zabave i odmora kod većine ljudi (PR newswire, 2024). Razlog tome je postojanje puno različitih žanrova videoigara koji se međusobno isprepliću te omogućuju da svaka osoba pronađe neki tip igre koji joj odgovara. Sama videoigra ili računalna igra je elektronička igra koja uključuje interakciju s korisničkim sučeljem ili uređajem za unos (kao što je joystick, kontroler, tipkovnica ili uređaj za otkrivanje pokreta) za generiranje vizualne povratne informacije s uređaja za prikaz, koji se najčešće prikazuje u video formatu na televizoru, monitoru računala, ravnom zaslonu ili zaslonu osjetljivom na dodir na ručnim uređajima ili slušalicama za virtualnu stvarnost. Većina modernih videoigara su audiovizualne, sa audio komplementom koji se isporučuje preko zvučnika ili slušalica, a ponekad i s drugim vrstama senzorskih povratnih informacija (npr. haptička tehnologija koja pruža taktilne senzacije). Neke videoigre također dopuštaju ulaze mikrofona i web kamere za razgovor i prijenos uživo u igri. Ovako veliki raspon interakcije sa videoigrama omogućio je industriji videoigara da zadovolji potrebe gotovo svakog ukusa koji bi igrači potencijalno mogli imati.

Industrija videoigara pripada tercijarnom i kvartarnom sektoru industrije zabave te je specijalizirana za razvoj, marketing, distribuciju, unovčavanje i povratne informacije potrošača o videoigrama. Industrija obuhvaća desetke radnih disciplina i tisuće poslova diljem svijeta. Industrija videoigara je kroz godine izrasla iz niše u mainstream. To se može vidjeti iz podataka da su u 2022. godini videoigre generirale zaradu u iznosu od 182,9 milijardi USD u globalnoj prodaji, (Wijman, 2023) što ih čini jednom od najprofitabilnijih industrija na svijetu, barem što se tiče industrije zabave. S obzirom na ovo nije čudno da se sve više poslodavaca i velikih kompanija želi pridružiti industriji videoigara. U drugu ruku, razvoj videoigara kao profesija je također postala i uvelike dostupna bilo kojoj osobi sa pristupom internetu.

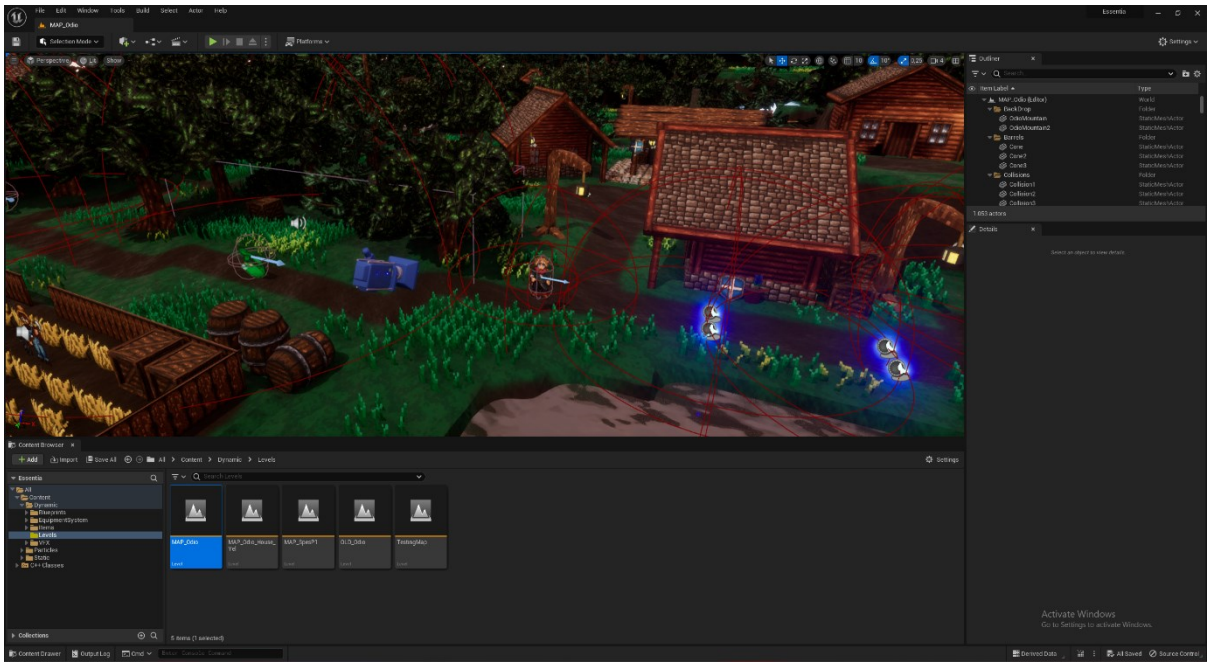
Uzeći u obzir velike količine tutorijala i foruma dostupnih na starnicama poput YouTube-a, u današnje doba svatko ima priliku napraviti videoigru te ju također i monetizirati te samostalno napraviti karijeru u razvoju videoigara. Te karijere mogu ostati u manjim sferama razvoja indie igara ili mogu ciljati na stvaranje velikih AAA studija. Indie igra, skraćeno za nezavisnu videoigru, je videoigra koju su kreirali pojedinci ili manji razvojni timovi bez financijske i tehničke podrške velikog izdavača igara, za razliku od većine "AAA" („triple-A“) igara. Zbog svoje nezavisnosti i slobode razvoja, indie igre često su usredotočene na inovacije, eksperimentalno igranje i preuzimanje rizika koji se obično ne dopuštaju u AAA igrama. U industriji videoigara, AAA je neformalna klasifikacija koja se koristi za klasifikaciju videoigara koje proizvode ili distribuiraju izdavači srednje veličine ili veći izdavači, koji obično imaju veći proračun za razvoj i marketing.

Upravo u sferu indie igre spada videoigra Essentia o kojoj će biti riječ u ostatku ovog diplomskog rada. Iako na tržištu već postoje tisuće i tisuće različitih videoigri, samo tržište još nije zasićeno, dapače ono neprestano raste. Sve više država daje poticaj novim studijima i projektima koji se bave razvojem videoigara, to se može vidjeti upravo na primjeru europske unije koja je sama u 2024. godini već raspisala značajan broj natječaja za poticaj za razvoj video igara (European Games Developer Fundation, 2024). U nastavku ćemo se dotaknuti razvoja videoigara općenito, proći ćemo kroz tri bitne faze razvoja videoigara, objasniti korištene tehnologije te ćemo zatim proći kroz općenite informacije o videoigri Essentia te planu njezinog razvoja i podjeli rada među članovima

tima. Naposlijetku ćemo proći kroz procese stvaranja i implementacije 3D modela i umjetne inteligencije specifično u igri Essentia.

2. Faze razvoja videoigara

Razvoj videoigara (ponekad skraćeno na gamedev) je proces stvaranja videoigara. To je multidisciplinarna praksa koja uključuje programiranje, dizajn, umjetnost, audio, korisničko sučelje i pisanje. Svaka od njih može biti sastavljena od više specijaliziranih vještina; umjetnost uključuje 3D modeliranje objekata, modeliranje likova, animaciju, vizualne efekte i tako dalje. Na Slika 1 je prikazan primjer rada na igri Essentia unutar programa Unreal Engine 5. Razvoj je podržan upravljanjem projektima, proizvodnjom i osiguranjem kvalitete. Timovi se mogu sastojati od više stotina ljudi, nekoliko desetaka ljudi ili čak jedne osobe.



Slika 1 Prikaz videoigre Essentia unutar Unreal Engine-a

Predprodukcija, produkcija i postprodukcija su ključne faze u razvoju videoigara. Svaka faza ima specifične zadatke i ciljeve koji doprinose konačnom proizvodu. Svaka od ovih faza je kritična za uspješan razvoj videoigre. Dobro planiranje i koordinacija između timova ključni su za stvaranje kvalitetnog i uspješnog proizvoda. Dalje slijedi detaljniji pregleda svake faze.

2.1. Predprodukcija

Predprodukcija je faza planiranja projekta gdje se fokusiramo na razvoj koncepta i ideje uz pisanje početnih dizajnerskih dokumenata. Dokumenti dizajna igre moraju biti precizni i opisni, jer oni služe kao temelj za daljnji razvoj igre. U ovoj fazi predprodukcije tim definira osnovne koncepte, mehanike igre, priču, likove i tehnološke potrebe te su svi ti koncepti zapisani upravo u tim dokumentima dizajna igre. Ono što je zapisano u tim dokumentima se može mijenjati i vrlo često se i mijenja, ali početan sklad i smjer koji daju za ostatak razvoja je jako bitan i ne bi se smio ignorirati. Ključni zadaci ove faze općenito uključuju:

1. **Razrada ideje:** Tim razvija osnovnu ideju igre, uključujući priču, svijet, likove i glavne mehanike igre.
2. **Dokumentacija dizajna igre:** Stvara se detaljna dokumentacija koja opisuje sve aspekte igre, često nazvana "dokument dizajna igre" (Game Design Document, GDD).

3. **Prototipiranje:** Razvijaju se osnovni prototipovi kako bi se testirale ključne mehanike igre i osiguralo da su zabavne i funkcionalne.
4. **Tehnička procjena:** Tim analizira tehničke zahtjeve i odlučuje o alatima, engineu (npr. Unity, Unreal Engine) i drugim tehnologijama koje će koristiti.
5. **Planiranje resursa:** Identificiraju se potrebni resursi, uključujući financije, ljudski resursi, softver i hardver.
6. **Izrada rasporeda:** Definira se vremenski okvir projekta, uključujući ključne prekretnice i datume isporuke.

2.2. Produkcija

Produkcija ili proizvodnja je glavna faza razvoja u kojoj se proizvode sredstva i izvorni kod za igru. Glavna produkcija obično se definira kao razdoblje u kojem projekt bude u potpunosti završen. Programeri pišu novi izvorni kod, umjetnici razvijaju elemente igre, kao što su spriteovi ili 3D modeli. Ovo je najduža i najintenzivnija faza gdje se obavlja većina posla. Ključni zadaci uključuju:

1. **Razvoj:** Programeri, dizajneri, umjetnici i drugi članovi tima rade na stvaranju igre prema specifikacijama iz predprodukcije.
2. **Umjetnost i animacija**: Umjetnici stvaraju 2D i 3D modele, teksture, animacije i druge vizualne elemente.
3. **Dizajn razina:** Dizajneri razina stvaraju različite razine i okruženja u igri, osiguravajući da su izazovne i zabavne.
4. **Zvuk i glazba:** Kompozitori i dizajneri zvuka stvaraju glazbu, zvučne efekte i dijaloge.
5. **Testiranje:** Kroz cijeli proces, testerima se daje igra kako bi identificirali i prijavili bugove, balansirali igru i osigurali da sve funkcionira kako treba.
6. **Iteracija:** Na temelju povratnih informacija, tim kontinuirano poboljšava i prilagođava igru.

2.3. Postprodukcija

Postprodukcija je završna faza razvoja igre te je njezina glavna svrha održavanje igre, što uglavnom uključuje ispravljanje grešaka, poliranje, optimizaciju i pripremu za njezino izdavanje. Unatoč naporima testera, većina igara još uvijek sadrži manje greške u trenutku njihovog izdavanja. Prvih nekoliko mjeseci tijekom faze postprodukcije obično se provodi u identificiranju i uklanjanju tih grešaka. Ključni zadaci uključuju:

1. **Poliranje:** Tim radi na uklanjanju preostalih bugova, optimizaciji performansi i poboljšanju svih aspekata igre.
2. **Testiranje kvalitete (QA):** Temeljito testiranje kako bi se osiguralo da igra nema kritičnih bugova i da je spremna za izdavanje.
3. **Priprema za izdavanje video igre:** Stvaranje marketinških materijala, priprema demo verzija, postavljanje na platforme za distribuciju (Steam, PlayStation Store, itd.).
4. **Podrška nakon izdavanja video igre:** Nakon izdavanja tim može raditi na zakrpama za ispravljanje bugova, dodavanju novih sadržaja (DLC) i pružanju podrške igračima.
5. **Analiza:** Prikupljanje povratnih informacija od igrača i analiza podataka kako bi se poboljšali budući projekti.

3. Dizajn i razvoj igre Essentia

U ovom djelu rada ćemo više pričati o dizajnu i realizaciji igre Essentia. To je igra koja je u razvoju sa ciljem puštanja na tržište i prodaje. Na Slika 2 Je prikazan izgled videoigre Essentia. Ona je videoigra akcijsko-avanturističkog žanra gdje igrač kontrolira lika imenom Yel (vidljiv na Slika 2). Igra je postavljena u svijetu fantazije gdje je magija stvarna te je svijet napučen mističnim biljkama i životinjama. Cilj igre i Yel-a je istražiti svijet i pronaći svoje mjesto u njemu, na tom putu suočiti će se sa raznolikim neprijateljima i drugim preprekama koje će morati proći. U tome će mu pomoći njegovi prijatelji koji će biti članovi družine, njih će sve igrač moći nadograđivati, ili sa iskusnim bodovima ili sa opremom koju nađe i kupi u svijetu te ovisno o razini interakcije sa drugim likovima moći će otključati različite zajedničke napade koji se dio originalnog „stacking“ sistema unutar igre. Nakon što igrač pobijedi zlog kralja demona koji terorizira svijet, igrač završava priču i pobjeđuje igru.

Slika 2 Prikaz igre Essentia

3.1. Game Design Document

Game design document (GDD) je temeljni pisani dokument koji detaljno opisuje sve aspekte video igre. Služi kao vodič za razvojni tim kroz cijeli proces stvaranja igre, od početne ideje do konačnog proizvoda. GDD uključuje različite elemente igre kao što su priča, likovi, svijet, mehanike igre, pravila, korisničko sučelje, umjetnički stil i audio dizajn. Također specificira tehničke zahtjeve i planove za implementaciju. Cilj GDD-a je osigurati da svi članovi tima imaju jasno razumijevanje igre i njezinih ciljeva, čime se olakšava komunikacija i koordinacija unutar tima. GDD se često ažurira tijekom razvoja igre kako bi odražavao promjene i poboljšanja. Dobro napisan i temeljit GDD je ključan za uspješan razvoj igre, jer pomaže u minimiziranju nesporazuma i zadržavanju fokusa na originalnoj viziji igre. U nastavku je prikazan GDD-a za videoigru Essentia.

Radni naslov

Essentia

Koncept u jednoj rečenici

Essentia je akcijsko avanturistički RPG sa fokusom na priču. Igrači će prolaziti kroz raličite razine, svaka sa drugačijim vizualnim stilom, gdje će ih dočekati različiti izazovi, zadaci i neprijatelji.

Žanr / žanrovi

Akcija, avantura, RPG

Ciljana publika

Publika koja voli staromodne RPG naslove, dob između 20 i 30 godina, također publika koja voli igre sa lijepim vizualima i pričom.

Jedinstveni elementi („selling points“)

3D svijet povezan sa 2D sprite likovima.

Doživljaj igranja i perspektiva igre

Igrač kontrolira glavnog lika Yel-a te može hodati raznolikim svijetom, istraživati ga i pronaći tajne prolaze i slično. Svijet je na prvu ruku miran i lijep, ali što više igrač igra to će se više suočiti sa različitim zadacima i neprijateljima koje mora nadvladati.

Vizualni i zvučni stil

Izgled je lagan i ugodan na oči, dizajn svijeta je pun fantazije dok su zvukovi realistični i ugodni igraču.

Tema

Fantasy, meta.

Igraći svijet

Svijet se sastoji od nekoliko različitih razina koje se mogu istražiti, one su vizualno drugačije jedna od druge, ali igrač uvijek ima pristup svakoj razini te se može vratiti na njih i istraživati ih svojim tempom. Naravno, nakon što prvo prijeđe glavnu misiju na jednoj razini, tek onda može ići na sljedeću, ali na prijašnju se uvijek može vratiti.

Monetizacija

Igra neće u sebi imati nikakvih dodatnih transakcija, nego će ju se samo trebati jednom početno kupiti.

Platforme, tehnologije i opseg

PC, Xbox One i Series X/S, PlayStation 4 i 5, Nintendo Switch, Apple i Android.

Ciljevi i napredovanje kroz igru

Igrač dobiva bodove iskustva koje može trošiti na posebne napade ili da pojača odnos sa pomagačima koji su sa njim u timu. Također može pronaći predmete u svijetu kojima može opremiti sebe ili druge članove tima kako bi ih ojačao ili pripremio za različite situacije. Cilj igre je pobijediti glavnog negativca te završiti priču.

Mehanike

Mehanika hoda, trčanja, izbjegavanja, borbe, korištenja posebnih sposobnosti, dijaloga, pokupljanja objekata, istraživanja, nadogradnje i napredovanja.

Podsustavi

Pronalaženje puta, umjetna inteligencija neprijatelja, prijateljski sistem, sustav dijaloga i sustav interaktivnog inventara.

Interaktivnost

Igrač može hodati svijetom ili trčati kako bi ga istražio, može imati razgovor sa druželjubivim NPC-evima, može pokupiti objekte sa poda ako je to dozvoljeno, bori se protiv neprijatelja, može nadograditi opremu ili kupovati stvari u trgovinama te može imati interakciju sa drugim članovima tima kako bi si poboljšao odnos sa njima i otključao nove zajedničke napade.

3.2. Korištene Tehnologije i alati

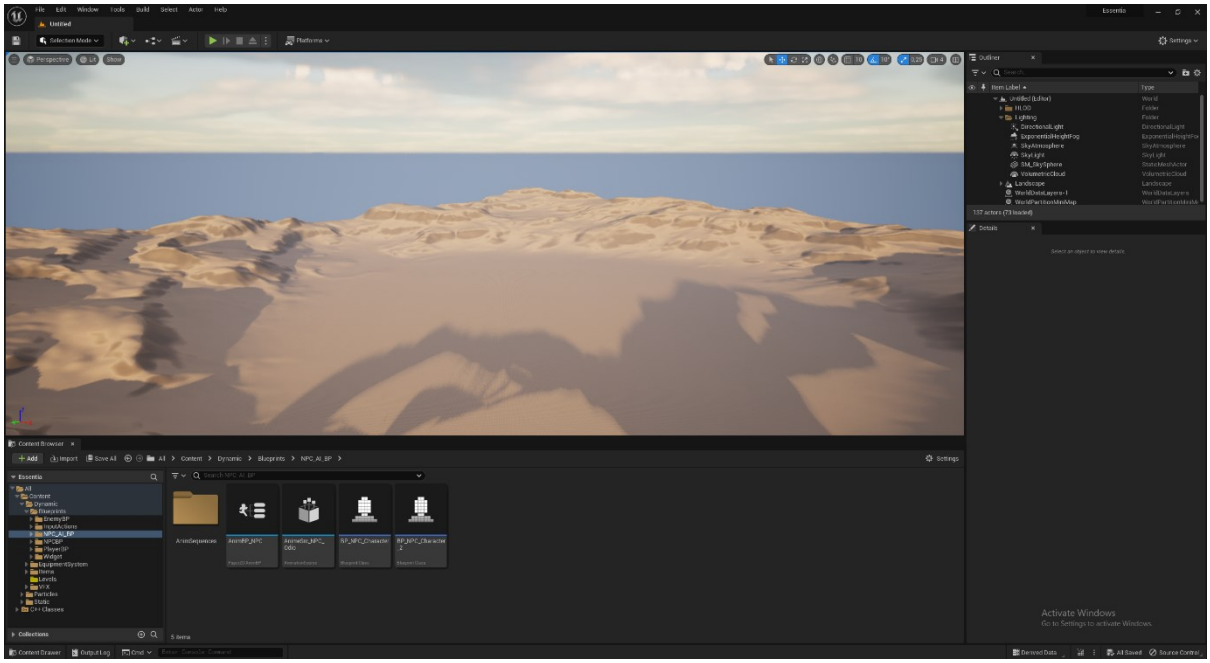
Kako je ovo produkcija indie igre tim nije imao nikakvog budžeta na raspolaganju. Iz tog razloga sve tehnologije koje su korištene su besplatne sa velikom količinom dostupnih tutorijala na internetu. Unreal Engine je upravo jedan od takvih tehnologija. Najnovija verzija Unreal Engine-a 5 sa sobom donosi najnovije tehnologije vezane uz kreiranje video igara te je također dostupan veliki broj tutorijala i foruma u slučaju da treba pomoć tijekom razvoja. Blender je sljedeći alat koji se koristio, u zadnjih nekoliko verzija Blender se transformirao u industrijski standard vezan za 3D modeliranje, što znači da on također donosi najnovije tehnologije sa sobom, također je besplatan za korištenje, što je bio i drugi razlog njegovog korištenja. Naposlijetku za kontrolu verzija projekta bio je korišten Git, kako je on također standard za kontrolu verzija projekata te je besplatan za korištenje, također se koristila platforma GitLab za primjenu Git-a unutar projekta, razlog tome je bio upoznatost i već postojeće iskustvo korištenja GitLab-a.

3.2.1. Unreal Engine 5

Osnovna korištena tehnologija za izradu igre Essentia je Unreal Engine 5. Unreal Engine (UE) je serija game engine-a koji je razvio Epic Games, prvi put prikazan u igri Unreal iz 1998. godine. Game engine je softverski okvir prvenstveno dizajniran za razvoj videoigara i općenito uključuje relevantne biblioteke i programe podrške. Prvobitno razvijen za PC pucačine u prvom licu, od tada se koristi u raznim žanrovima igara te su ga usvojile druge industrije, ponajviše filmska i televizijska industrija. Unreal Engine je napisan u C++ i ima visok stupanj prenosivosti, podržavajući širok raspon platformi za stolna računala, mobilne uređaje, konzole i virtualnu stvarnost (Epic Games, Inc., n.d.).

Najnovija generacija, Unreal Engine 5, izdana je u travnju 2022 te se izgled njegovog sučelja može vidjeti na Slika 3. Njegov izvorni kod dostupan je na GitHubu, a komercijalna upotreba dopuštena je

na temelju modela tantijema, pri čemu Epic naplaćuje 5% prihoda iznad 1 milijun USD, što se ne tiče za igre objavljene na Epic Games Store-u. (Epic Games) Epic je u game engine ugradio značajke stečenih tvrtki kao što je Quixel, čemu je pomogao prihod Fortnitea. Godine 2014. Guinnessova knjiga rekorda proglasila je Unreal Engine "najuspješnijim game engine-om za videoigre" na svijetu (Guiness world records, 2014).



Slika 3 Prikaz sučelja Unreal Engine-a 5

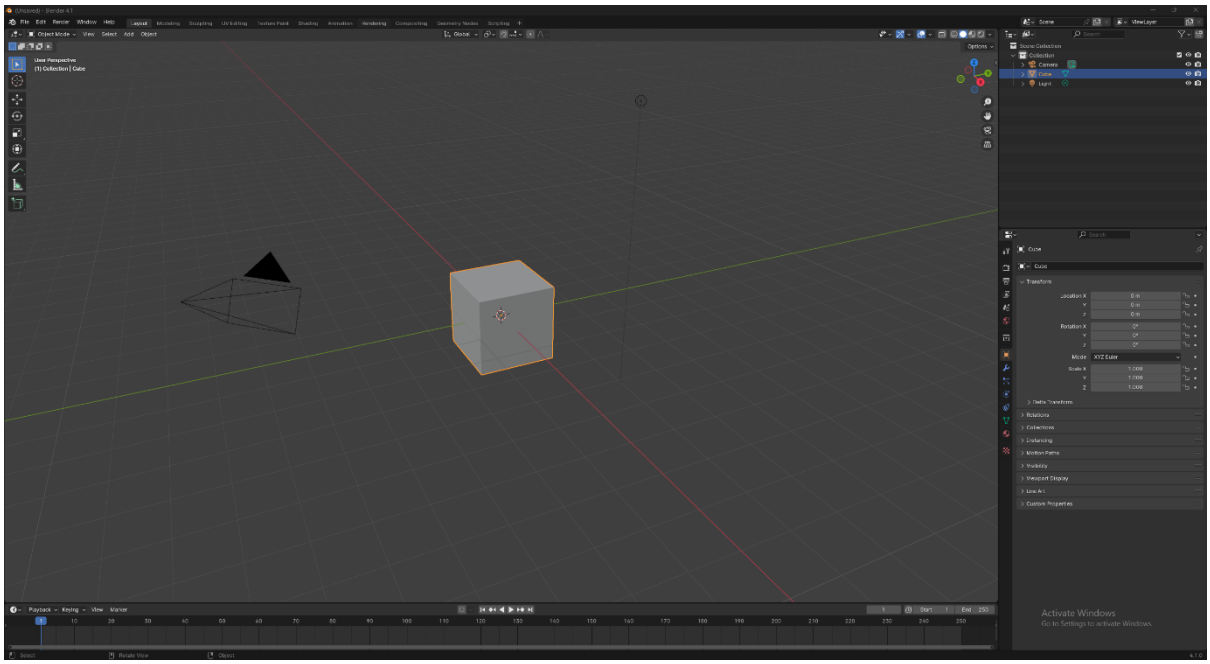
3.2.2. Blender

Za izradu svih 3D modela prvenstveno je korišten Blender. Blender je besplatni i open-source alat za 3D modeliranje, animaciju, renderiranje i postprodukciju, koji se koristi u raznim industrijama zabave. Omogućuje stvaranje i uređivanje 3D modela, animacija i rendera koristeći alate za modeliranje kao što su ekstrudiranje, rezanje, pomicanje, skaliranje i rotacija. Blender podržava razne tehnike animacije, uključujući ključne okvire, krivulje animacije, oblikovanje i rigging za kreiranje skeleta likova, dok render engine Cycles i Eevee omogućuju stvaranje fotorealističnih i stiliziranih rendera. Također, Blender nudi mogućnosti za simulacije tekućina, dima, vatre, krutih tijela, tkanina i čestica, omogućujući realistično prikazivanje fizičkih fenomena.

Ugrađeni node-based kompozitor služi za obradu i kombiniranje slojeva rendera i video materijala, dok osnovni nelinearni video editor (VSE) omogućuje uređivanje i montažu video zapisa. Alati za skulptiranje omogućuju detaljno modeliranje i oblikovanje površina modela pomoću različitih kistova, a podržava i proceduralno i ručno oslikavanje tekstura, te napredne mogućnosti stvaranja i uređivanja materijala.

Blender također podržava Python skriptiranje za automatizaciju zadataka, stvaranje dodataka i prilagodbu radnog okruženja. Zahvaljujući svojoj fleksibilnosti, mogućnostima i aktivnoj zajednici koja doprinosi razvoju i dijeljenju resursa, Blender je popularan alat među profesionalcima i hobistima. Osim toga, budući da je open-source, besplatan je za korištenje, što ga čini pristupačnim širokom spektru korisnika (Blender, n.d.). Sama verzija Blender-a nije bitna jer se sam program Blender ne povezuje sa Unreal Engine-om, već se samo koriste izvozne datoteke iz njega. Stoga se

konstanto koristila najnovija verzija, koja je za vrijeme pisanja ovog diplomskog rada Blender 4.1, prikaz njegovog radnog sučelja se može vidjeti na Slika 4.



Slika 4 Prikaz sučelja Blender-a 4.1

Upotreba Blendera:

1. **Videoigre:** Kreiranje 3D modela, likova, animacija i scena za igre.
2. **Animirani filmovi:** Proizvodnja kratkih i dugometražnih animiranih filmova.
3. **Vizualni efekti:** Izrada vizualnih efekata za filmove, televiziju i reklame.
4. **Arhitektura:** Stvaranje 3D modela i vizualizacija arhitektonskih projekata.
5. **Dizajn proizvoda:** Modeliranje i vizualizacija novih proizvoda u fazi dizajna.

3.2.3. Git i Gitlab

Za suradnju u timu i kontrolu verzija korišten je Git i platforma Gitlab. Git je distribuirani sustav kontrole verzija koji prati verzije datoteka. To je besplatan softver otvorenog koda koji se dijeli samo pod licencom GPL-2.0. Često se koristi za kontrolu izvornog koda od strane programera koji zajednički razvijaju softver. Ova funkcionalnost je veoma korisna jer je omogućila da više programera radi na projektu u isto vrijeme (Torvalds, 2005).

Ključne značajke Gita uključuju:

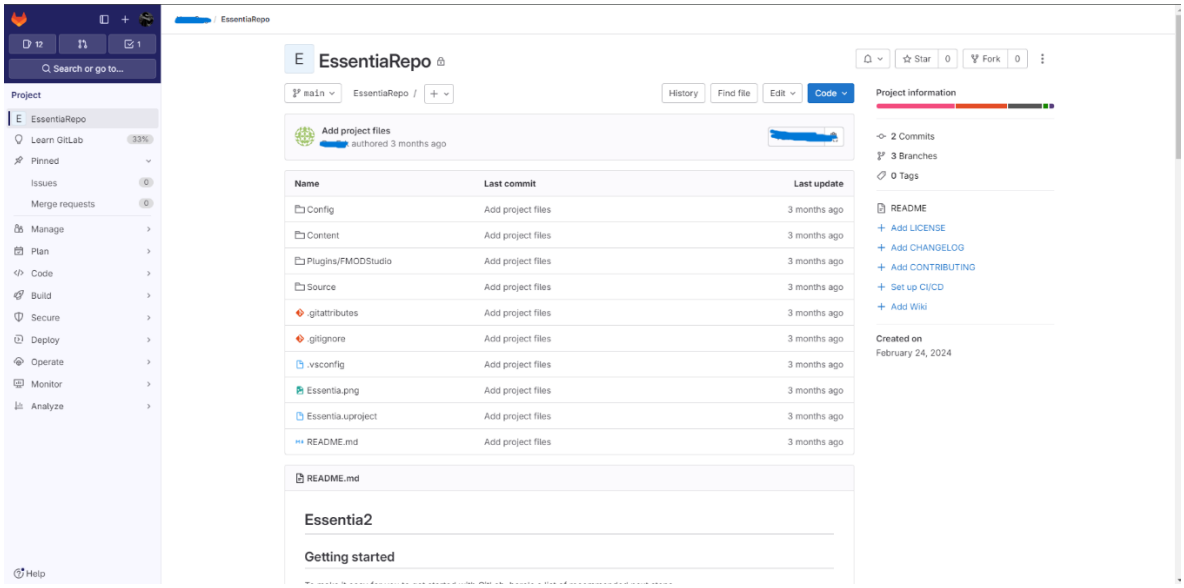
1. **Distribuirani sustav**: Svaki korisnik ima potpunu kopiju cjelokupne povijesti verzija, što omogućuje rad bez potrebe za stalnom vezom na centralni poslužitelj.
2. **Grananje i spajanje:** Git olakšava rad s različitim granama koda, što omogućuje paralelni razvoj novih značajki, popravke grešaka i eksperimentiranje bez utjecaja na glavni kod. Spajanje grana je učinkovito i često bezbolno.
3. **Praćenje promjena:** Git precizno prati sve promjene u kodu, omogućujući pregled i upravljanje poviješću projekta. Svaka promjena je označena jedinstvenim hashom, što omogućuje preciznu identifikaciju.

4. **Kolaboracija:** Git podržava suradnju među više programera, omogućujući im istovremeni rad na istim datotekama i lako upravljanje sukobima u kodu.
5. **Repozitoriji:** Kôd se pohranjuje u repozitorije, koji mogu biti lokalni na računalu korisnika ili udaljeni na poslužiteljima poput GitHub, GitLab ili Bitbucket, olakšavajući dijeljenje i suradnju.
6. **Izvedba:** Git je dizajniran za brzinu i učinkovitost, omogućujući brze operacije poput komitovanja, pregledavanja povijesti i grananja.

Danas je Git de facto standardni sustav kontrole verzija. To je najpopularniji distribuirani sustav kontrole verzija, s gotovo 95% programera koji ga prijavljuju kao svoj primarni sustav kontrole verzija od 2022. godine (Stack Overflow, 2022). To je najčešće korišten alat za upravljanje izvornim kodom među profesionalnim programerima. Postoje razne ponude usluga Git repozitorija, uključujući GitHub, SourceForge, Bitbucket i GitLab, te naš tim upravo odlučio na korištenje GItLab-a.

GitLab je web-bazirana DevOps platforma koja pruža alate za upravljanje projektima, kontinuiranu integraciju i isporuku (CI/CD), te druge DevOps prakse, izgrađena na temeljima Gita. Omogućava timovima da surađuju na razvoju softvera, upravljaju repozitorijima koda i automatiziraju tijekove rada od planiranja do produkcije (DevPro Journal Staff, 2021). GitLab je dao sve što se moglo tražiti od pouzdane web aplikacije za zajednički rad na projektu.

Nakon što sto se napravila inicijalna verzija projekta unutar Unreal Engine-a 5, također se i stvorio repozitorij na GitLab-u, zatim se prenijela prva verzija projekta na novostvoreni repozitorij. Ovo je sad omogućilo da se dodaju suradnici te uspostavi kolaboracija. Svi suradnici imaju pristup svim datotekama te im mogu pristupiti i mijenjati kada god žele. Kako bi se izbjegle greške koje bi nastale kada bi svi članovi istovremeno radili na istom projektu, iskoristila se značajka Git-a koja omogućuje grananje te kasnije spajanje. Većinu vremena bile su stvorene tri grane koje su omogućile programerima te dizajneru zvuka da rade na funkcionalnostima i implementiraju zvukove, 3D asset-e i druge stvari u projekt. Kada bi jedna funkcionalnost bila dovršena, grana na kojoj je bila rađena bi bila spojena sa glavnom granom naziva „master“ kako bise provjerilo da li dolazi do nekih grešaka sa ostalim funkcionalnostima. Dodatne grane za druge funkcionalnosti bi zatim bile napravljene iz nove master grane, to osigurava da se dodatne funkcionalnosti stalno rade na najnovijoj verziji projekta kako bi se odmah znalo da li ima grešaka u kompatibilnosti između implementacija funkcionalnosti. Bitno je napomenuti da GitLab također ima dobar sustav zaštite kao što su skeniranje ranjivosti, praćenje tajni, upravljanje pristupom i auditiranje, pomažući u osiguravanju sigurnosti koda i procesa razvoja, koji štiti privatan kod i projekte što sprječava njegovo curenje i rizik da ga netko ukrade. Na Slika 5 je upravo prikazan primjer gdje se vidi repozitorij sa datotekama vezanih uz projekt Essentia. Naposlijetku valja spomenuti da Git treba biti instaliran lokalno na računalu kako bi se koristile komande za korištenje Git-ove funkcionalnosti.



Slika 5 Prikaz sučelja GitLab-a

3.3. Plan Razvoja

Ovaj projekt je veoma širok, stoga je bilo potrebno pronaći adekvatne ljude za razvoj igre te napraviti realističan plan razvoja. U prvoj fazi predprodukcije bilo je potrebno razviti početnu ideju, stil izgleda te generalnu priču igre. Ovo je trajalo nekoliko mjeseci. U tom vremenu se također definirala veličina tima te potrebni talenti za uspješan razvoj igre. U završnom djelu predprodukcije se odlučilo kako će dalje teći razvoj videoigre.

Odlučilo se da će se unutar prve godine razviti demo verziju koja će u sebi sadržati prvih pola sata do sat vremena igre te će unutar nje većina sistema biti programirana. Ovo je faza produkcije u kojoj se i trenutno razvoj igre nalazi. Nakon toga biti će napravljena Kickstarter stranica. Kickstarter je stranica koja omogućuje ljudima da doniraju monetarnu pomoć projektima koje žele podržati. Ostatak razvoja bi bio podržan donacijama preko Kickstarter-a, sve dok se ne bi pronašla neka izdavačka kuća ili bi se odlučili na samostalno objavljivanje igre. Određeno je da ostatak razvoja traje još dodatnih godinu dana kako bi se popravljale greške, unapređivali postojeći sistemi te gradio ostatak svijeta.

Nakon završetka razvoja igra bi bila puštena na tržište gdje bi se tim nastavio brinuti o njoj i objavljivao ažuriranja ovisno ako se pronađu neke greške ili ako bi se nešto moglo unaprijediti. Ovisno o popularnosti igre naknadno bi se objavljivale besplatne kozmetike za likove i njihovu opremu kako bi se igra održala što duže na životu.

3.4. Podjela rada

Kako je već bilo napomenuto ovo je velik projekt te se on ne bi mogao ostvariti sa samo dva člana. Stoga je rano odlučeno da će tim trebati minimalno 5 članova sa mogućnosti proširenja.

Prvi član je glavni programer koji se bavio programiranjem većeg djela sistema videoigre Essentia te također služi kao kreativni redatelj igre pošto je ovo originalno bila njegova ideja. Uz to također sudjeluje u teksturiranju modela i pisanju objava na društvenim mrežama.

Ja sam osobno zadužen za različite aspekte razvoja igre. Prvotno kao dodatni programer koji se bavi razvojem različitih sistema, zatim kao 3D artist gdje stvaram različite objekte koji naseljavaju virtualni svijet i same 3D mape kojima se igrač kreće. Također sam zadužen za pisanje scenarija te smišljanje kampanja za digitalni marketing i stvaranje sadržaja za njega (primjerice montaža video uredaka).

Sljedeća članica je zadužena za crtanje i umjetnost. Ona radi 2D spriteove te ih ujedno i animira. Uz to njezin zadatak je da stvori koncepte vezane za okoliš, likove, neprijatelje itd.

Za zvuk je zadužen sljedeći član. On sve zvuke sam stvara i miksa ovisno po potrebi. Nakon što kreira zvukove jedan od dva programera ih implementira u samu igru.

Naposlijetku, što se tiče glazbe odlučili se za originalnu glazbu te se s toga unajmio posljednji član tima koji stvara originalnu glazbu za igru Essentia. One je samo privremeno dio tima te sudjeluje jedino kada treba napraviti novu originalnu glazbu za nove dijelove igre.

Svaki član tima je bitan za razvoj cijele igre. Također, suradnja je bitna kako bi bila ostvarena cjelovita vizija. Iz tog razloga se redovito održavaju sastanci gdje članovi diskutiraju o smjeru igre te daju ideje.

4. Realizacija igre

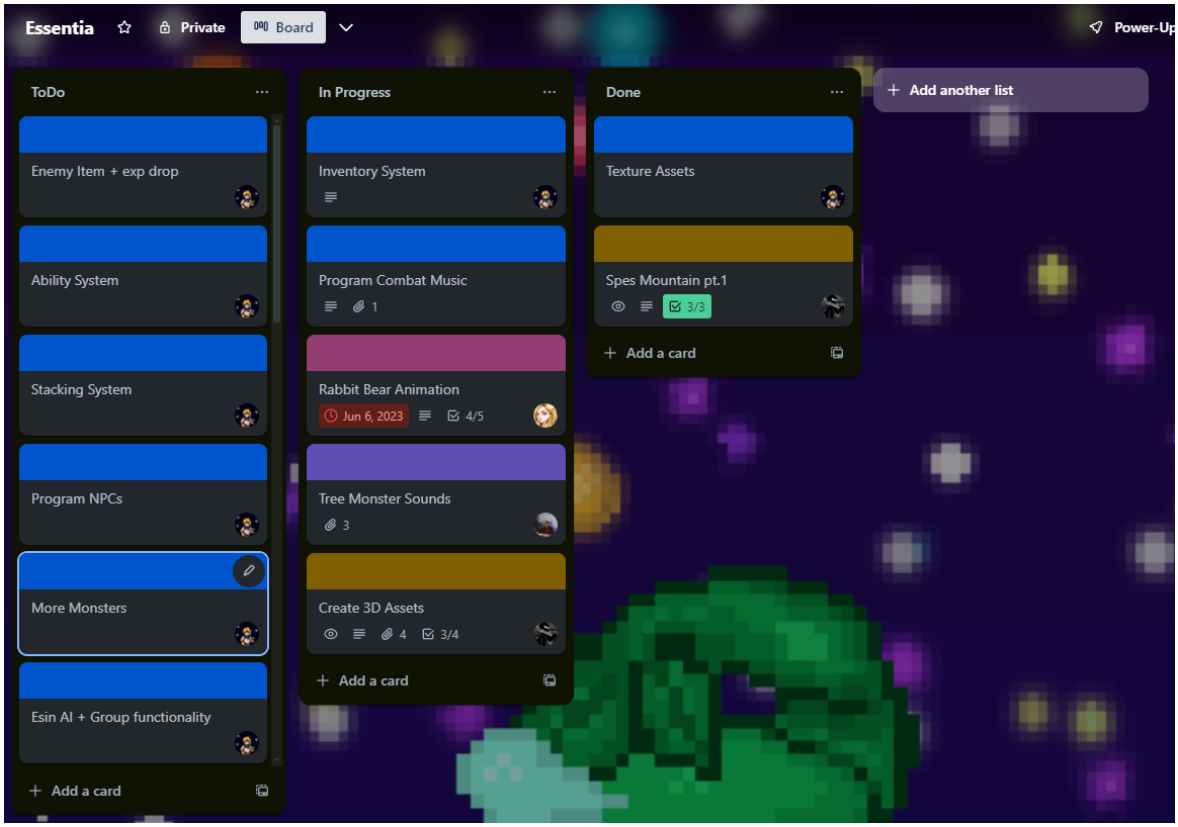
U ovom djelu ovog diplomskog rada ćemo detaljnije proći kroz konkretan razvoj videoigre Essentia. U početku ćemo se ukratko dotaknuti poslova i dijelova razvoja kojima se ja osobno nisam bavio, a zatim ćemo prijeći na dijelove za koje sam ja osobno bio zadužen.

4.1. Organizacija rada

Kod realizacije igre koristimo agilni pristup u stvaranju aplikacija. Agilna metoda razvoja videoigara je iterativni i inkrementalni pristup koji omogućuje fleksibilnost i brzu prilagodbu promjenama tijekom razvoja. Temelji se na kratkim ciklusima, nazvanim iteracijama ili sprintovima, koji obično traju od dva do četiri tjedna, a svaki ciklus donosi novi dio funkcionalnosti igre. Ključni elementi agilne metode uključuju Scrum i Kanban. Scrum timovi su mali i sastoje se od članova s različitim vještinama potrebnim za razvoj igre, dok se sav posao organizira kroz product backlog, sprint planning, daily stand-ups, sprint review i sprint retrospective sastanke. Kanban koristi vizualne ploče za prikaz tijeka rada i ograničava broj zadataka u određenim fazama kako bi se osigurala učinkovitost. Prednosti agilne metode uključuju brzu prilagodbu promjenama, povećanu vidljivost i transparentnost, poboljšanu suradnju, više povratnih informacija i bolji fokus na korisnika. Međutim, izazovi uključuju potrebu za disciplinom i organizacijom, potencijalnu fragmentaciju posla i rizik od preopterećenja timova. Uz pravilnu implementaciju i prilagodbu specifičnostima tima i projekta, agilna metoda može značajno poboljšati efikasnost i kvalitetu procesa razvoja videoigara, omogućujući brže prilagodbe i bolje zadovoljavanje potreba igrača.

Korištenje te metode omogućuje timu veću fleksibilnost i minimizira čekanje drugih članova tima. No čekanja još uvijek postoje, primjerice kako bi član tima zadužen za stvaranje zvuka mogao napraviti zvukove prvo treba čekati da član zadužen za dizajniranje dizajnira lika. U većini slučajeva to nije potrebno. Svaki član tima dobije svoje zadatke te ih samostalno izvrše, za to se koristio alat za upravljanje projektima zvan Trello. Na njega zaduženi član napiše ostalim članovima zadatke i definira rokove te nakon što članovi izvrše svoj zadatke ih stave iz TODO u DONE sekciju kako bi se znalo da je jedan zadatak dovršen, isječak Trello listi se može vidjeti na Slika 6. Na kraju se rezultati zadataka spoje na radnu verziju projekta. Tu dolazimo do korištenja Git-a. On omogućuje programerima da rade na različitim funkcionalnostima u isto vrijeme te ih na kraju samo spoje u finalni proizvod. Razlog zašto je specifično odabran GitLab je zbog znanja oko njegovog korištenja. Da se umjesto njega koristila neka alternativa poput GitHub-a, rezultat bi bio isti. Treba samo napomenuti da kako bi GitLab radio sa Unreal Engine-om treba izgenerirati poseban file unutar radnog direktorija projekta kako bi Unreal Engine korektno radio sa Git-om.

U nastavku ovog diplomskog rada ćemo proći kroz zadatke za koje sam osobno bio zadužen.

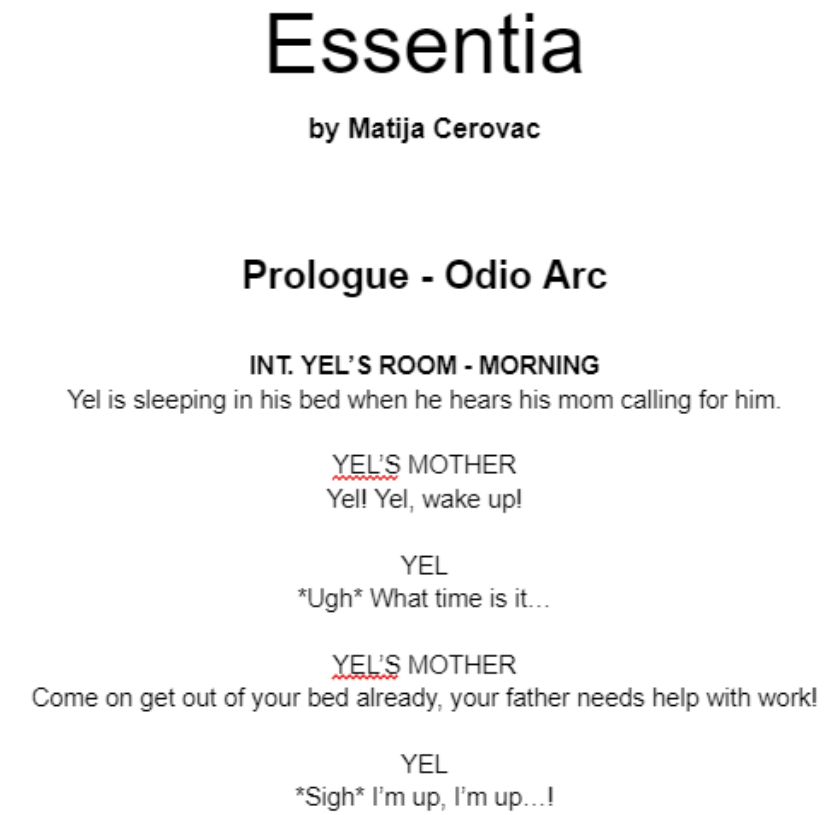


Slika 6 Prikaz Trello alata

4.2. Pisanje scenarija i digitalni marketing

4.2.1. Scenarij

Pisanje scenarija za videoigre je složen proces koji zahtijeva kreativnost, tehničke vještine i duboko razumijevanje interaktivnosti. Na Slika 7 se može vidjeti isječak sa početka scenarija za videoigru Essentia.



Essentia

by Matija Cerovac

Prologue - Odio Arc

INT. YEL'S ROOM - MORNING

Yel is sleeping in his bed when he hears his mom calling for him.

YEL'S MOTHER
Yel! Yel, wake up!

YEL
Ugh What time is it...

YEL'S MOTHER
Come on get out of your bed already, your father needs help with work!

YEL
Sigh I'm up, I'm up...!

Slika 7 Isječak scenarija za videoigru Essentia

Nekoliko dobrih praksi koje mogu pomoći u stvaranju uvjerljivog i uspješnog scenarija za videoigru koje sam koristio u pisanju scenarija za videoigru Essentia su sljedeći (Schell, 2014.):

1. **Poznavanje žanra:** Razumijevanje žanra igre pomaže u oblikovanju tona, stila i strukture priče. Essentia je akcijsko avanturistički RPG stoga se mogao primijeniti tradicionalni način pisanja priče sa 3 čina bez da bi stajao u jakom kontrastu sa ostatkom igre.
2. **Upoznavanje s gameplayem:** Priča treba biti u skladu s mehanikama igre. Pisac mora znati kako igra funkcionira kako bi integrirao priču s gameplay-om. Ovo je ostvareno u dva djela. Dok istražuje svijet, igrač ima mogućnosti pričati sa likovima unutar igre kako bi saznao više o priči i svijetu igre. Drugi dio je uobičajeni način gdje na početku nove misije, ili ponekad usred borbi, krenu takozvane cutscene koje guraju priču naprijed i služe kao glavni izvor bitnih informacija za igrača. U videoigri Essentia cutscene su zamišljene slično kao i dijalozi sa likovima u svijetu, bez ikakvih dodatnih animacija ili kutova gledanja.
3. **Glavni zaplet i podzapleti:** Razvijanje glavnog zapleta igre i nekoliko podzapleta kako bi priča bila bogata i složena. Ovo se direktno veže za već spomenute činove. Ovaj način pisanja činova je većinom korišten u filmovima, no kako priče u videoigrama većinom traju duže nego filmovi, to zahtjeva da ima više od jednog zapleta. Naravno, glavni dio priče treba imati i svoj glavni zaplet, ali pametno je i također staviti više manjih podzapleta koji upotpunjuju priču te stavljaju u fokus više drugih likova.
4. **Tijek radnje:** Nadovezajući se na točku broj 3, također treba definirati jasnu početnu točku, ključne trenutke (plot points) i završetak. Ako priča ima previše podzapleta, dolazi do rizika

da će se izgubiti glavna srž priče. Zato je bitno nikad ne zaboraviti dati fokus glavnom zapletu priče te ga inkorporirati u podzaplete da funkcioniraju zajedno, a ne jedan protiv drugog.

5. **Dubina likova:** Potrebno je stvoriti likove s jasnim motivacijama, osobnostima i prošlošću. Likovi trebaju biti uvjerljivi i slojeviti. Ova filozofija je bila korištena kod pisanja svaka od četiri glavna lika, ali i kod većine glavnih negativaca, kako bi oni imali veći utjecaj u priči i ostali u sjećanju igrača i nakon igranja igre.
6. **Razvoj likova kroz radnju:** Omogućiti da se likovi razvijaju tijekom igre, što će ih učiniti zanimljivijima igračima. Ovo se nadovezuje na prijašnju točku broj 6 gdje se dubina likova otkriva kroz tijek igre kako igrač prolazi kroz igru i saznaje više o likovima Essentia-e te vidi kako se mijenjaju kao likovi zbog događaja u priči.
7. **Prirodni dijalog:** Dijalog treba biti prirodan i odgovarati osobnosti likova, no ne smije se zaboraviti na žanr igre, kako dijalog ne bi naškodio konačnom stilu. U video igri Essentia to je postignuto tako da likovi ne koriste moderni žargon i ne rade reference na modernu kulturu, već pričaju u kontekstu svijeta fantazije.
8. **Detaljan svijet:** Poželjno je stvoriti bogat i detaljan svijet s vlastitom poviješću, kulturom i pravilima. To se najlakše ostvari pomoću dijaloga gdje likovi mogu spomenuti neki dio svijeta ili dio povijesti koji igraču više govori o svijetu video igre. Dobra je praksa pokazati te detalje kroz ponašanje likova, a ne nužno kroz čisti dijalog.
9. **Priča kroz okoliš:** Dobra je praksa ispričati dio priče kroz okoliš, objekte i vizualne elemente igre. To se može ostvariti tako da se ostave ruševine koje su dio prošlosti svijeta ili možda čak i oštećenu opremu koja je ostala od prijašnje bitke na ovom terenu, što signalizira igraču da je trenutačno mjesto opasno. U početnim dijelovima Essentia-e se to može vidjeti u obliku razrušenih objekata poput mostova i kuća na putu do borbe sa glavnim negativcem početnog dijela igre.
10. **Emocionalni trenuci:** Prijeko je potrebno uključiti trenutke koji će izazvati emocionalne reakcije kod igrača, bilo kroz priču, likove ili situacije. Na samom početku priče Essentie je već vidljiv jedan od takvih trenutaka. Nakon uvodnog dijela gdje igrač nauči mehanike igre, glavni lik Yel nakon emotivnog razgovora sa svojim najboljim prijateljem Esinom odluči napustiti rodno selo te putovati svijetom.
11. **Relatabilnost:** Povezivanje priče s temama i situacijama koje su prepoznatljive i relevantne igračima. Essentia je igra koja se odvija u nepoznatom svijetu, sa nepozantim pravilima i poviješću. To ne znači da osjećaji i osobni put koji prođu likovi nisu isti onima koji ljudi imaju i danas. Dapače, glavna tematika igre je pronalazak samog sebe i poante života glavnog lika.
12. **Kontinuitet priče:** Treba se osigurati da priča teče logično i bez prekida. Svaki igrač ima svoj vlastiti tijek igranja, stoga je važno osigurati da se priča ne raspadne dok igrač radi neplanirane stvari. To je osigurano tako da se dijelovi mape i druge mape općenito zaključane ako igrač nije došao do dijela priče gdje bi ih trebao prvi put vidjeti.
13. **Konzistentnost likova i svijeta:** Potrebno je paziti na konzistentnost u ponašanju likova i pravilima svijeta igre. To je generalno postignuto pazeći na stil pisanja i stil govora koji svaki lik ima. Na to je bitno paziti kako svaki lik ima svoju osobnost, poseban stil govora, poštapalice, ciljeve i drugo.
14. **Testiranje priče:** Testiranje priče s različitim igračima, članovima tima ili testerima kako bi se dobile povratne informacije. Za ovu su točku dobro došli sastanci na kojima su se prezentirale ideje i pitalo za mišljenja vezanih uz priču, također su se pitali i bliski prijatelji o mišljenjima vezanih za priču te su se svi komentari uvažili i diskutirali.

15. **Revizija i prilagodba:** Potrebno je biti spreman prilagoditi i poboljšati priču na temelju povratnih informacija. Bitno je također konstantno tražiti i nova mišljenja, čak i nakon revizija. Kod razvoja priče za Essentia-u dogodilo se nekoliko puta da se napravilo toliko revizija da su one jednostavno prekomplicirale male detalje. Stoga je bitno znati koje revizije uzeti u potpunosti, a koje djelomično, a koje u potpunosti odbaciti.
16. **Komunikacija s dizajnerima i programerima**: Blisko surađivanje s ostatkom tima kako bi priča bila dobro integrirana u igru. Priča ne igra samo ulogu u dijalogu, već primjerice izgled terena se mijenja ovisno u kojem dijelu priče je trenutno igrač. Neke posebne vještine su zaključane iza specifičnih dijelova u priči stoga će igrač njima imati pristup tek nakon što prođe te dijelove.
17. **Fleksibilnost:** Uvijek je potrebno biti spreman na promjene i prilagodbe tijekom razvoja igre. Sama priča Essentie je promijenjena više puta, te promijene su ponekad i izašle od ideje dizajnera koji bi predstavio novi dizajn lika što bi onda inspiriralo nove dijelove u glavnoj ili nekoj sporednoj priči. Ova točka je i najbitnija jer kroz konstantnu adaptaciju priča zaživi i bude dovršena.

Ove prakse mogu pomoći u stvaranju priče koja je ne samo zanimljiva i angažirajuća, već i dobro integrirana s gameplayem, što je ključno za uspjeh videoigre. Kao što je već bilo spomenuto glavni lik Essentia-e se zove Yel. Uz njega likovi koji će biti u njegovoj družini su njegov najbolji prijatelj Esin te Ceri i Tasla. Uz njih se u priči također pojavljuju mnogi sporedni likovi od kojih su najvažniji Emir Ordo koji im se često pridružuje na putovanjima. Trenutno, sveukupno se u scenariju nalazi 32 imenovana lika, ovo je broj koji se lako može povećati dodavanjem bitnih likova ili jednostavno imenovanjem sve više NPC-eva, kako svi NPC-evi nemaju imena unutar igre, već nekima pod imenom stoji samo „Villager“ (hrv. seljak) ili „Guard“ (hrv. stražar). Od svih njih najviše rečenica ima Yel, što je očekivano kako on je glavni lik te stupa u najviše interakcije sa ostalim likovima. Poslije njega najviše dijaloga ima Ceri kako je njezina osobnost da bude više pričljiva sa ostalima te nakon nje idu Esin pa Tasla, Emir Ordo ima najmanje od navedenih likova kako on je sporedni lik. Sam scenarij trenutno stoji na 112 stranica što se još uvijek može promijeniti dodavanjem novih dijelova ili sporednih misija unutar igre. Također, unutar scenarija je potrebno napisati interakcije koje igrač ima sa NPC-evima unutar igre koji progovore samo jednu do dvije rečenice, tako da ako se odluči proširiti broj NPC-eva u nekoj sceni to će zahtijevati i proširenje scenarija.

4.2.2. Digitalni Marketing

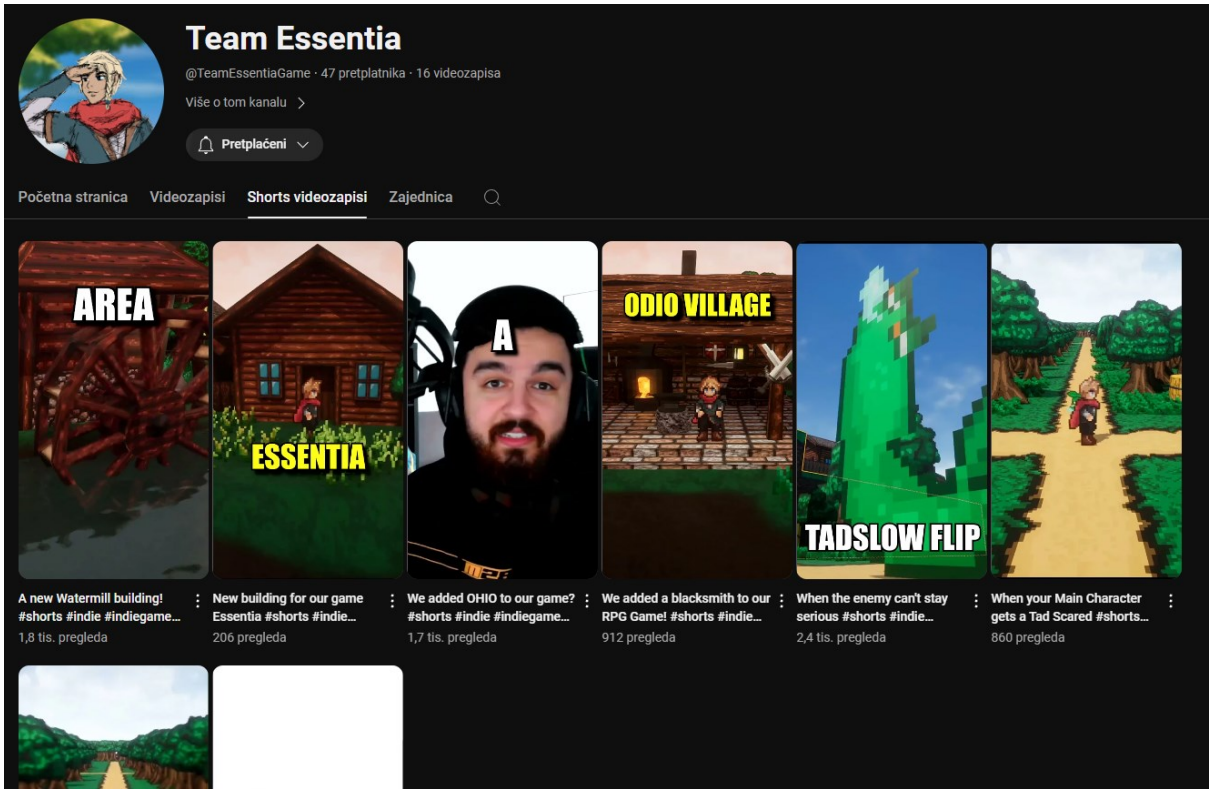
Strategija za digitalni marketing se može podijeliti u 3 faze. Prva faza je također i trenutna, dok su druga i treća u pripremi. Cilj prve faze je privući pažnju potencijalnih kupaca. Ovu fazu obilježavaju objave na X-u, YouTube-u i TikTok-u, prikaz YouTube kanala se može vidjeti na Slika 9, dok se prikaz TikTok profila može vidjeti na Slika 10. Ove objave su ili zabavnog ili informativnog sadržaja te se primjer tako jedne objave može vidjeti na Slika 8. Cilj je da barem dio ciljane publike sazna za igru i da se stvori osnovna publika koja će širiti daljnju riječ o igri. Ova faza je usko povezana sa fazom produkcije kako većina sadržaja dolazi iz objava vezanih uz razvoj igre.

Slika 8 Primjer objave na X-u

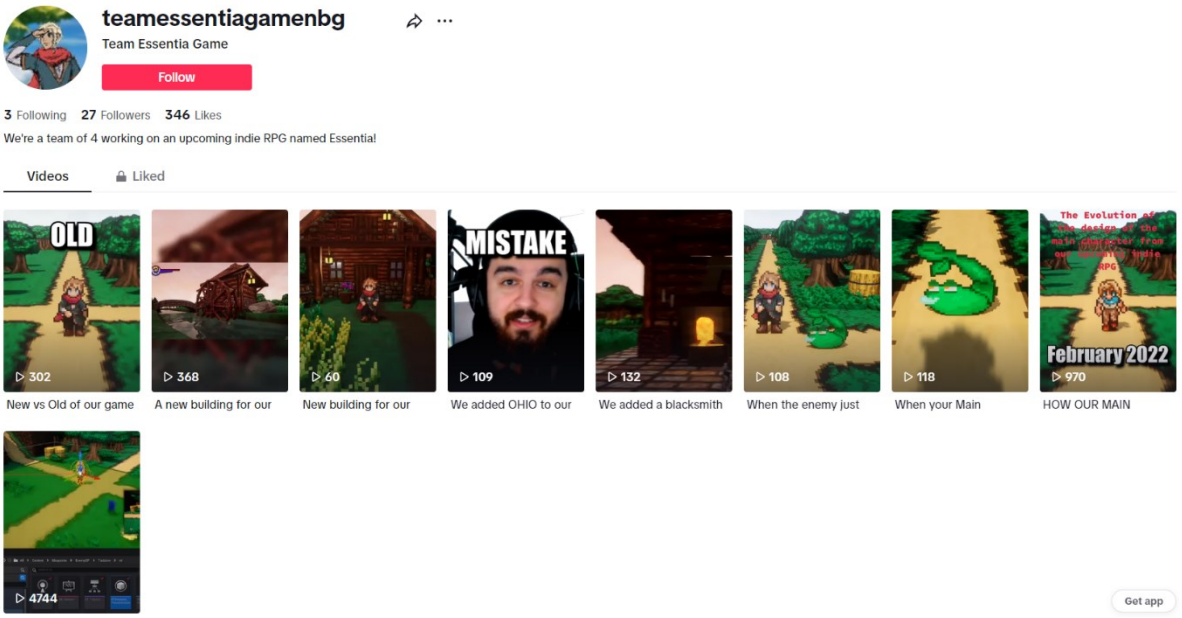
Sljedeća faza se odnosi na Kickstarter. Kako je bitno da se što više sadržaja pokaže tijekom kampanje za grupno financiranje tu sadržaj iz prve faze dolazi u igru te je također povezan sa fazom produkcije video igre. Doduše, taj sadržaj treba prilagoditi. Iz prve faze publika već može dobiti povjerenje da će se projekt razvijati te da će dobivati aktualne novosti vezane uz igru. Uz malo prilagođen sadržaj sličan onom iz prve faze, u drugoj fazi će se također napraviti najava koja će otkriti mogućnost igranja demo verzije igre. Uz to, također će biti napravljen video koji govori o povijesti ovog projekta kako bi publiku upoznao sa članovima tima i projektom općenito. Ova faza više-manje signalizira prijelaz iz ležernih tipova objava u više profesionalan stil.

To nas dovodi do posljednje faze. Ona će pratiti postprodukciju razvoja igre. Ovdje će sadržaj na svim platformama prijeći većinski u profesionalni oblik sa različitim najavama i novostima vezanima uz videoigru. Cilj je stvoriti profesionalnu i legitimnu sliku oko projekta tako da bude što uspješniji. Ova faza također signalizira i kraj Kickstarter kampanje kako je ona zamišljena da traje samo do konačnog izdavanja igre.

Naravno, kao i u svemu ova kampanja je fleksibilna i otvorena promjenama. Također postoji mogućnost ekspanzije na druge platforme ili možda čak i napuštanja originalnih. Sve ovisi o tome u kojem aspektu se uspije zainteresirati ciljana publika. Možda će neke vrste objave dobiti prednost naspram drugima, isto vrijedi i za različite društvene mreže.



Slika 9 Team Essentia YouTube kanal



Slika 10 Team Essentia TikTok profil

4.3. Realizacija Asseta

Asseti u videoigrama su ključni elementi koji čine vizualni, zvučni i interaktivni sadržaj igre, a uključuju grafiku, modele, teksture, zvukove, glazbu, animacije i skripte. Oni su bitni jer direktno utječu na doživljaj igrača, kvalitetu igre i njenu prezentaciju. Za razvoj igre Essentia bilo je potrebno nekoliko vrsta asseta. Prvotno, vizualni asseti kao što su 2D i 3D modeli, teksture i animacije koji oblikuju izgled svijeta i likova. Od njih sam ja osobno bio zadužen za izradu 3D modela dok su se za teksture i animacije pobrinuli drugi članovi tima. Dalje su bitni zvučni asseti, poput efekata i glazbe, oni dodaju emocionalni sloj i dinamiku u igranje, za njih je također bio zadužen drugi član tima koji je radio kompletno originalne zvukove i efekte. Interaktivni asseti, uključujući skripte i kod, omogućuju funkcionalnost i mehanike igre. Postoji nekoliko vrsta asset-a: statični asseti (poput objekata u okolišu), dinamični asseti (poput animiranih likova i predmeta), audio asseti (poput zvučnih efekata i glazbe) te tekstualni asseti (poput dijaloga i naracije). Oni su također zastupljeni unutar Essentia-e, ali njih igrač rijetko vidi. U njihovoj sam implementaciji i izradi također osobno sudjelovao. Kvalitetno izrađeni asseti ključni su za stvaranje uvjerljivog i imerzivnog svijeta, osiguravajući konzistentnost i vizualnu privlačnost igre. Proces realizacije asseta obuhvaća planiranje, dizajn, stvaranje i integraciju u igru, često zahtijevajući suradnju između umjetnika, dizajnera, programera i zvučnih inženjera. Bez pažljivo izrađenih i implementiranih asseta, igra bi izgubila na kvaliteti i privlačnosti, što naglašava njihovu važnost u cjelokupnom procesu razvoja videoigara.

4.3.1. 3D Objekti

U 3D računalnoj grafici, 3D modeliranje je proces razvoja matematičke koordinatne reprezentacije površine objekta (neživog ili živog) u tri dimenzije putem specijaliziranog softvera manipuliranjem rubova, vrhova i poligona u simuliranom 3D prostoru.

Trodimenzionalni (3D) modeli predstavljaju fizičko tijelo pomoću skupa točaka u 3D prostoru, povezanih različitim geometrijskim entitetima kao što su trokuti, linije, zakrivljene površine, itd. Budući da su skup podataka (točaka i drugih informacija), 3D modeli se mogu izraditi ručno, algoritamski (proceduralno modeliranje) ili skeniranjem. Njihove površine mogu se dalje definirati mapiranjem teksture.

3D objekti unutar igre Essentia imaju različite svrhe. Počinjemo od samih mapa na kojima se igrač kreće. U mnogim slučajevima, također i u ovom za mape ili prostore unutar kojih se igrač kreće se koriste 3D modeli većeg obujma koji se zatim popunjuju manjim objektima kako bi se ostvarila željena scena.

Potrebno je i također popuniti svijet sa manjim 3D objektima koji će dati finalni izgled i identitet igre. Ti objekti mogu biti različitih veličina i obujma, ali moraju biti manje od mapa kao bi stali na njih. Ti objekti mogu služiti za populaciju primjerice šuma, stoga je potrebno napraviti nekoliko različitih modela stabla te ih nasumično postaviti na područje gde će se nalaziti šuma. Kod šuma nije potrebno napraviti poseban model za svako drvo već je dovoljno promijeniti rotaciju, veličinu ili oboje, jer igrač neće primijetiti ista drva, osim ako mu to nije namjera. Takva implementacija šume je prisutna u videoigri Essentia. Osim toga, veći objekti mogu biti zgrade ili objekti poput mostova ili bala sijena koji upotpunjuju scenu i stvaraju određen ugođaj u igri.

Zadnji tip 3D objekata unutar videoigre Essentia su također i najmanji, ali imaju najveći utjecaj na igrača. To su 3D objekti poput oružja i dragulja koji se mogu sakupiti. Oružja su mogla biti

napravljena kao dio sprite animacija likova jer će svaki lik uvijek imati isti tip oružja, ali u igri će biti implementiran sustav kozmetike stoga je bitno da su oružja 3D modeli na koja se kasnije mogu staviti drugačije teksture ili zamijeniti model, ovisno što igrač bude htio. Što se tiče dragulja, oni su napravljeni u obliku 3D modela kako bi se bolje uklapali unutar svijeta i dizajna igre. Kada se oni pokupe doduše, u inventaru se prikazuju kao slike.

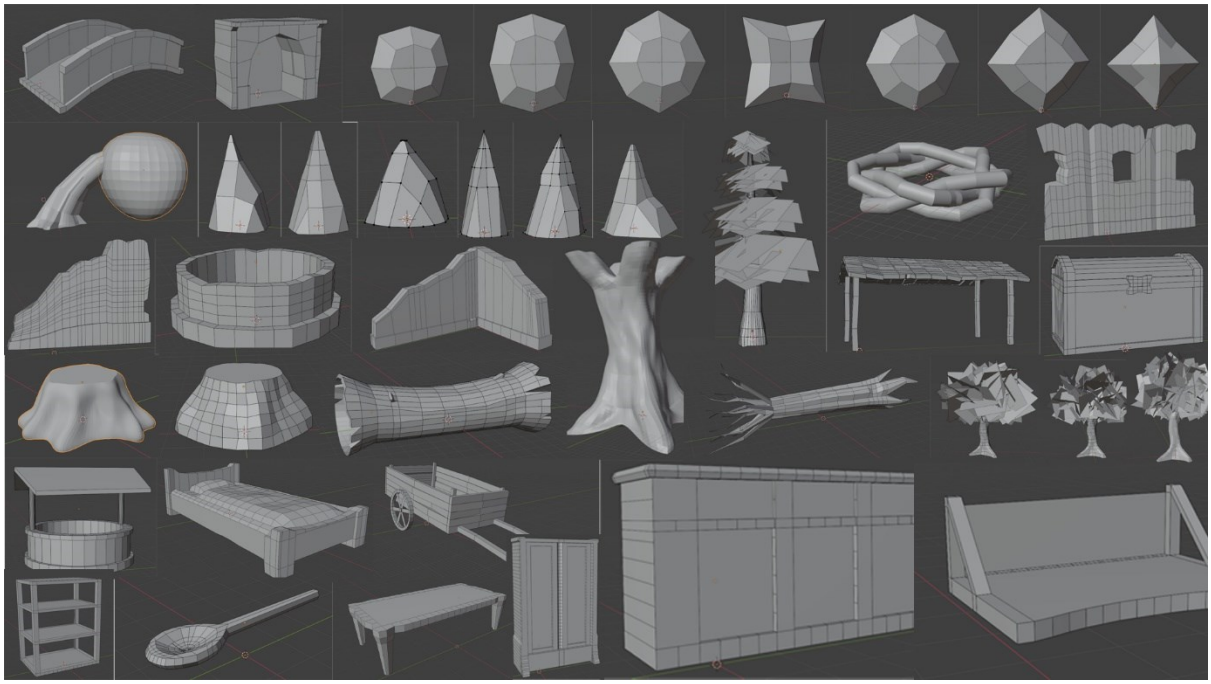
Mogu se još spomenuti mali 3D objekti poput žlica i tanjura koji su korišteni za interijere, ali oni imaju istu svrhu kao drugi tip, popunjavanje prostora. Također neki su elementi unutar igre mogli biti 3D modeli, ali su napravljeni u obliku 2D sprite tekstura iz razloga jer tako pomažu u upotpunjavanju vizualnog stila igre. Primjerice cvijeće, ono je moglo biti napravljeno sa 3D modelima, ali činjenica da su oni zapravo samo slike nacrtane u stilu sprite-a pomaže u spajanju 3D modela sa 2D sprite teksturama. Primjer sa cvijećem se može vidjeti na Slika 11.

Slika 11 Prikaz cvijeća unutar videoigre Essentia

Za uvodni dio video igre Essentia je bilo potrebno izraditi sljedeće elemente:

- 5 različitih drveća (tri uobičajena i dva bora)
- 6 različitih stalagmita
- 2 panja
- Jedno palo deblo
- 7 dragulja
- 4 različita tipa ruševina
- 9 različitih objekata koji se mogu pronaći u kućama (poput kamina, kreveta i ormara)
- Bunar
- Kolica
- Nastršnica
- Most
- I jedno uže

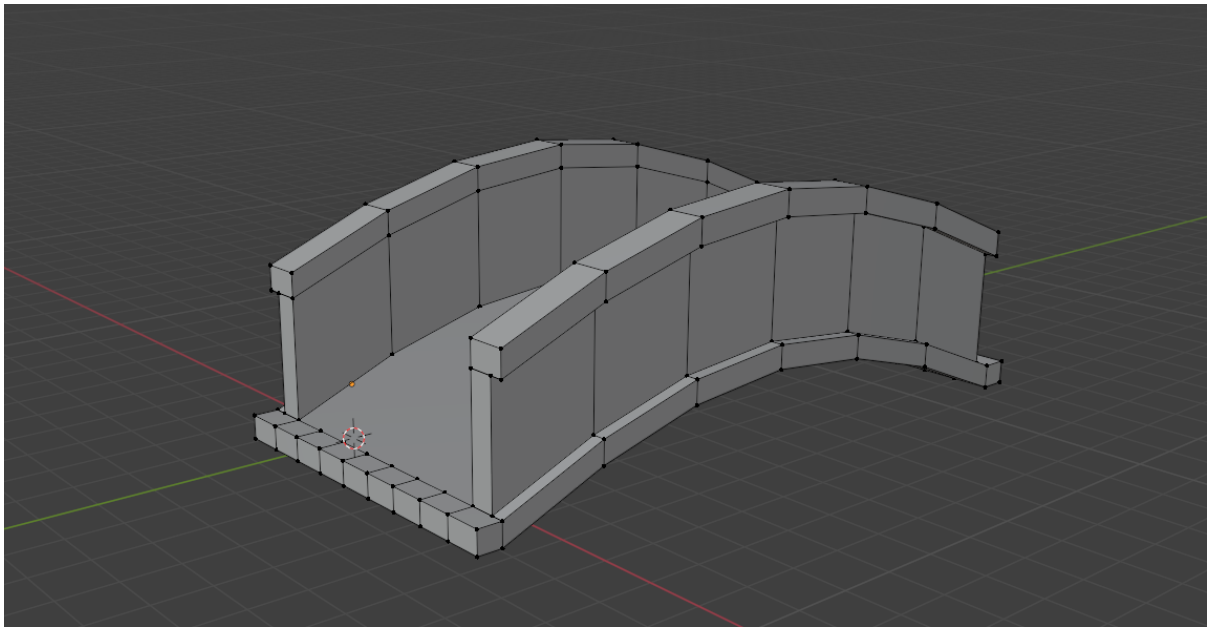
Većina ovih objekata ima ulogu popunjavanja scene i postavljanja atmosfere, ali neki imaju i utjecaj na igranje same igre. Primjerice, dragulji će biti objekti koji će igrač moći pronaći i pokupiti u svijetu te kada ih postavi u svom inventaru na lika, ti dragulji će mu dati pojačanja na određene statistike, tipa više zdravalja, jači udarci i drugo. Prikaz svih 3D modela koji su trenutnu u videoigri Essentia se mogu vidjeti na Slika 12.



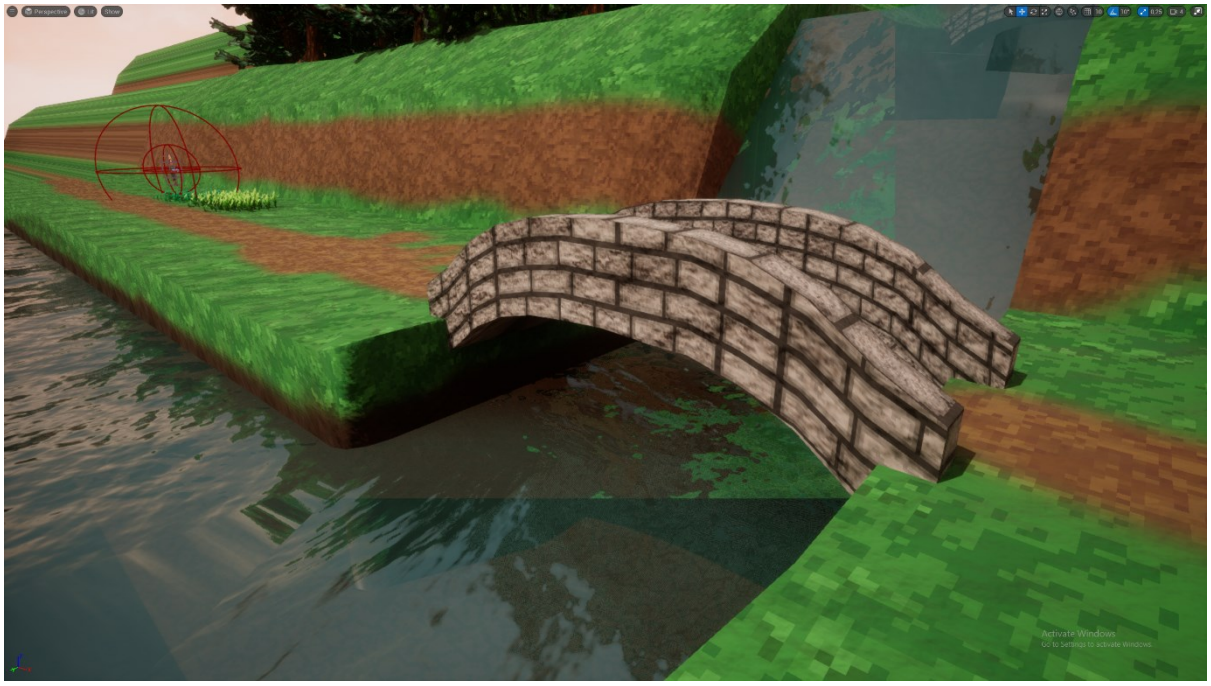
Slika 12 3D modeli izrađeni za video igru Essentia

4.3.2. Izrada 3D modela

Kao što je već bilo napomenuto, svi 3D modeli i mape su bili rađeni unutar programa Blender. Primjer tako jednog modela mosta se može vidjeti na Slika 13 te njegov konačan izgled u projektu na Slika 14. Počne se od jednostavnog modela kocke te se zatim prođe u „edit mode“ unutar blender pritiskom na tipku tab. Unutar Blender-a je svaki objekt definiran vrhovima koji kada se povežu čine konačni objekt, „edit mode“ dopušta korisniku da uređuje količinu, poziciju i odnos između tih vrhova, drugim riječima dopušta korisniku da modelira modele. Nakon što se uđe u „edit mode“ na početnu se kocku doda još vrhova, to stvara nove površine kojima se može dalje manipulirati. Cijela se kocka spljošti te izduži. Paralelne točke koje zajedno čine linije se jedna po jedna uzimaju i pomiču prema gore kako bi se dobila zaobljenost mosta. Nakon što se traženi oblik dobije potrebno je napraviti bočne držače. Osim točaka u „edit mode-u“ je moguće također i manipulirati linijama i površinama koje te točke zatvaraju. Stoga se odaberu površine koje su nastale na rubovima modela te se iz njih izvuče novi dio modela u visinu. Također, izvlačenjem površina na vrhu se dobije dubina i izgleda držača na vrhu. Te se na kraju uvijek mora paziti da ne dođe do grešaka kod generiranja dodatnih vrhova na nepoželjnim mjestima kako to može uništiti model te zakomplicirati proces teksturiranja ili implementacije u projekt.



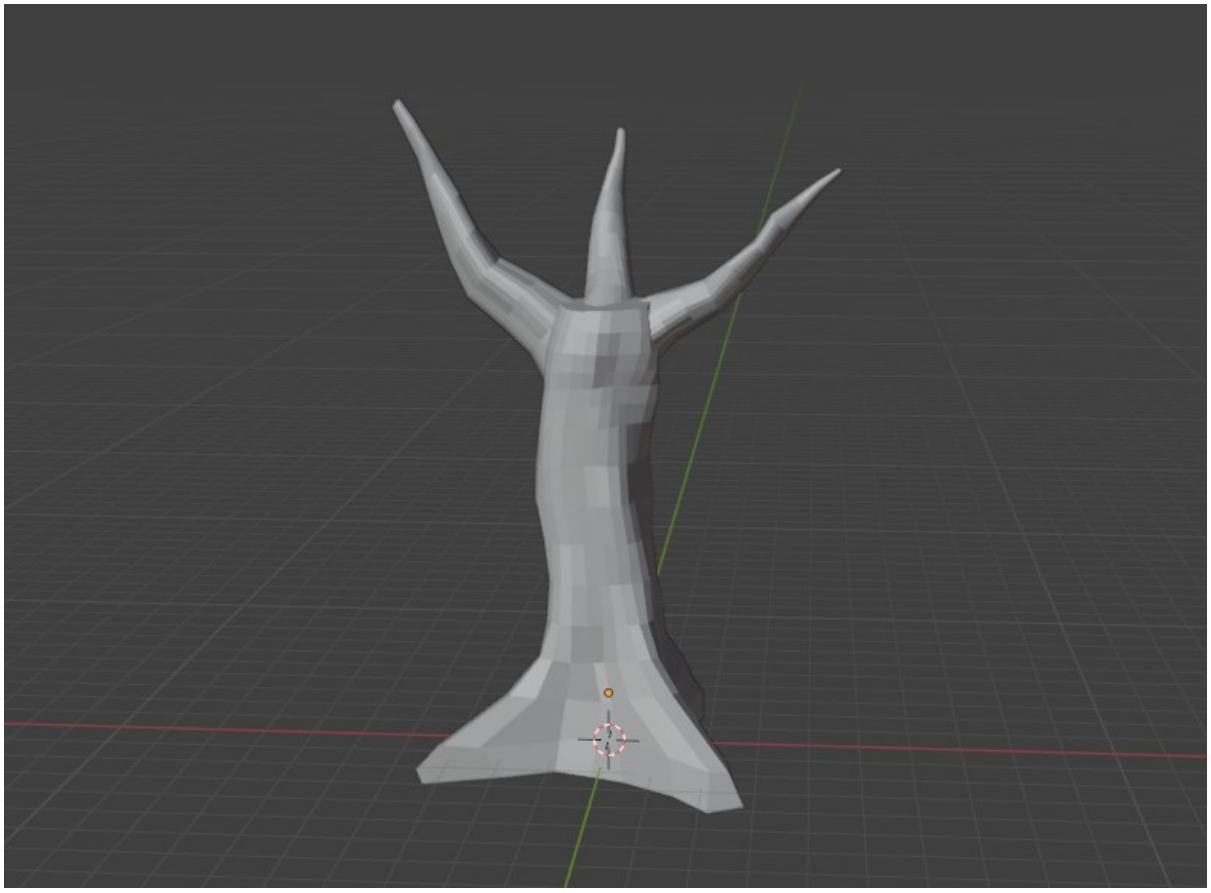
Slika 13 3D model mosta



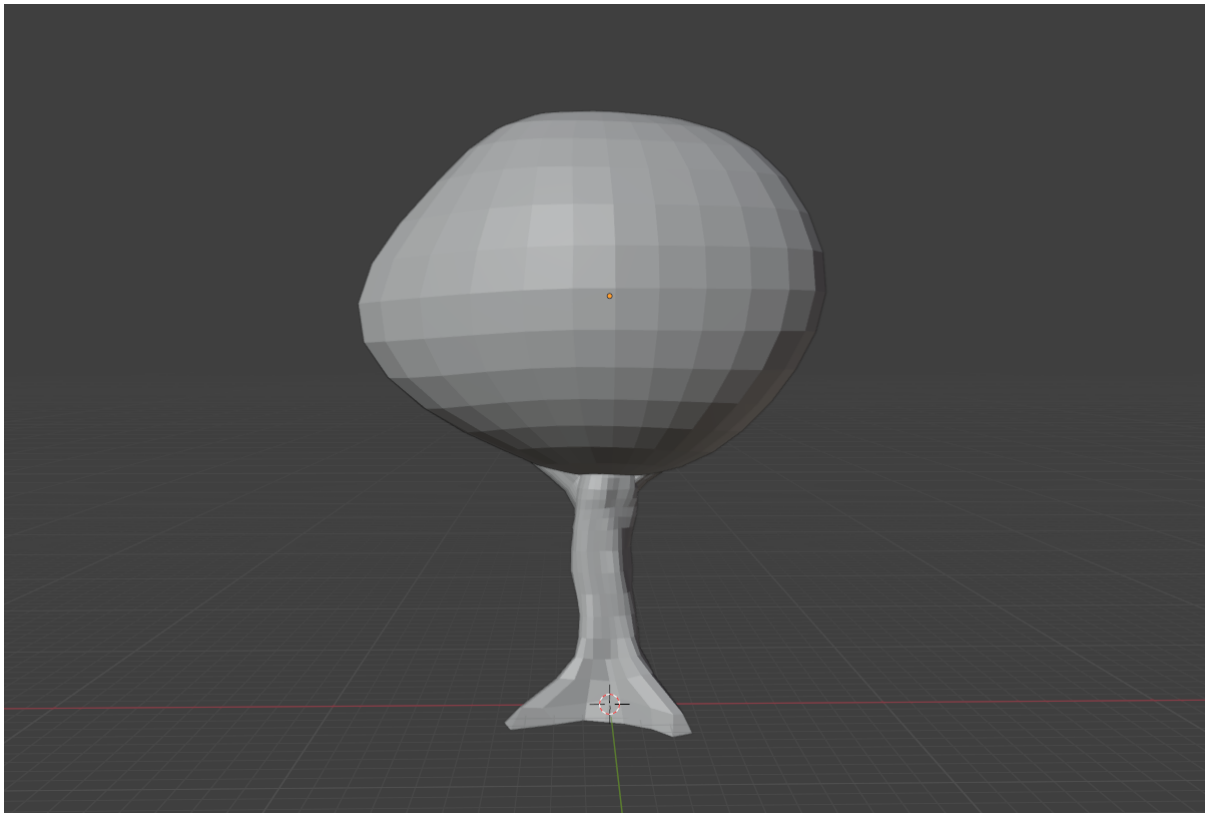
Slika 14 3D model mosta implementiran u projekt

Kod nekih objekta je još korištena i funkcionalnost skulptiranja, primjerice kod stvaranja drveća kako bi se postigao obli izgled stabla i dao hrapav izgled modelu. Unutar Blender-a se prvo napravi grubi izgled stabla, postupak je sličan kao i kod stvaranja mosta. Počne se od jednostavne kocke te se dodavanjem novih točaka unutar „edit mode-a“ omogućuje da se kocka zaobli. Dalje izvlačenjem površina se dobije visina stabla, također je moguće izvući površine pod kutom, to daje nepravilan i prirodan oblik stablu. Na dnu je također bitno izvući nekoliko dijelova koji će predstavljati korijene stabla. Zatim se uđe u novu karticu unutar Blender-a gdje se nalazi funkcionalnost skulptiranja. Ova funkcionalnost tretira modelu kao da je napravljen od gline te brojnim alatima korisnik može oblikovati model, tipa urezati liniju, izgladiti ga, saviti ga, ispuhati ga i drugo. Za stvaranje stabla je

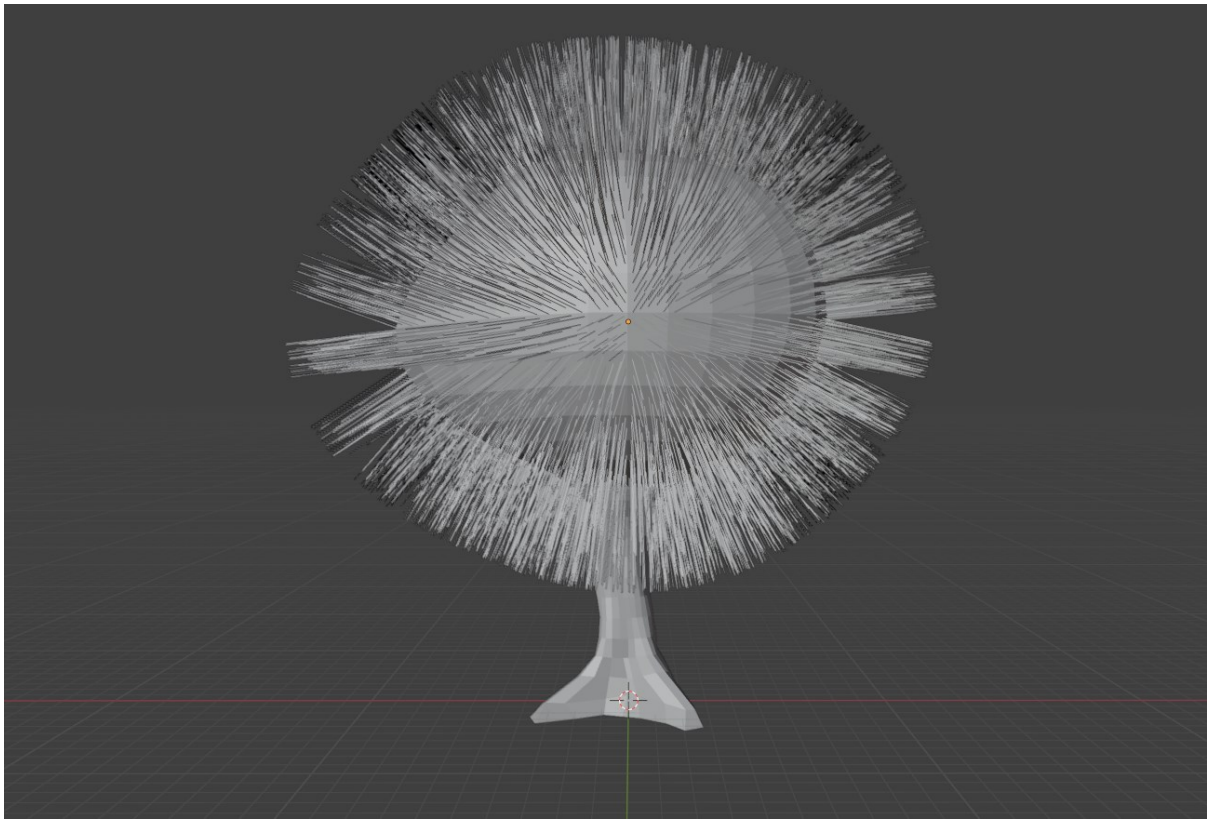
bilo bitno koristiti alat za urezivanje crta. Promjenom parametra alata, poput debljine i dubine se precizno oblikovao izgled stabla. Ovdje je bitno napomenuti da se najbolji rezultati postignu što više vrhova neki model ima, odnosno što je detaljniji. Dalje se uzmu alati za napuhivanje i ispuhivanje sa niskim parametrom kako efekt ne bi zahvatio cijelo stablo te se pomoću njih oblikuju korijeni te se njihov utjecaj protegne preko cijelog stabla kako bi se ostvarila kohezija oko modela. Na kraju se uzme hrapav alat koji daje modelu izgled kore, konačan izgled debla drveta se može vidjeti na Slika 15. Kod drveća je također i korišten sustav čestica kako bi stvorila krošnja. Ovo je bitno jer kad bi se individualno išla raditi krošnja, proces bi trajao dugo te bi također kod implementacije bio zahtjevan za računala. Prvotno se doda 3D model UV sfere koji je već dostupan unutar Blender-a te se on oblikuje u željen izgled krošnje, ova UV sfera će služiti kao baza za krošnju na koju će se staviti čestice dlaka, ovaj dio postupka se može vidjeti na Slika 16. Blender-ov sustav čestica je veoma napredan te se inače može koristiti da program sam izgenerira veliki broj sitnih objekata, tipa kosa, prašina, trava i drugo. Ali ako se ovaj sustav primjeni na drvo on može izgenerirati velik broj čestica kose na drvetu, što se može vidjeti na Slika 17, te dlake se u postavkama zatim zamijene sa lišćem i na brz i efikasan način se dobije krošnja drveta. Unutar ovog procesa se mogu također definirati grupe čestica odnosno dlaka te pomoću njih Blender na uzoru na grupe pored stvara grupe lišća, ova tehnika uvelike smanjuje na težini i kompleksnosti modela što štedi resurse računala. To je bitno jer jedan model poput drveta će se puno puta prikazati na sceni u jednom vremena, stoga da igra ne bi imala problema je potrebno činiti ovakve modele što manje kompleksnima. Primjer tako jedno napravljenog stabla se može vidjeti na Slika 18 te njegova implementacija u projekt se može vidjeti na Slika 19.



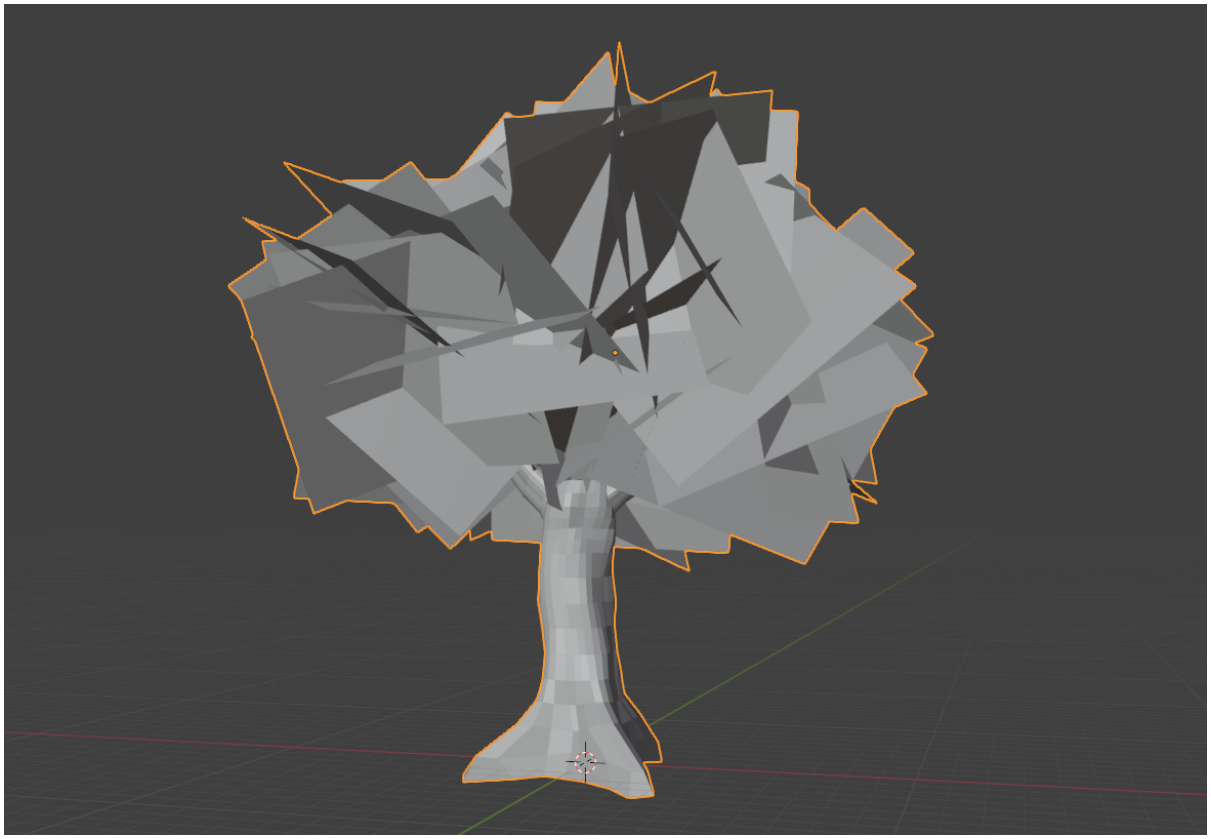
Slika 15 Prikaz stabla bez krošnje



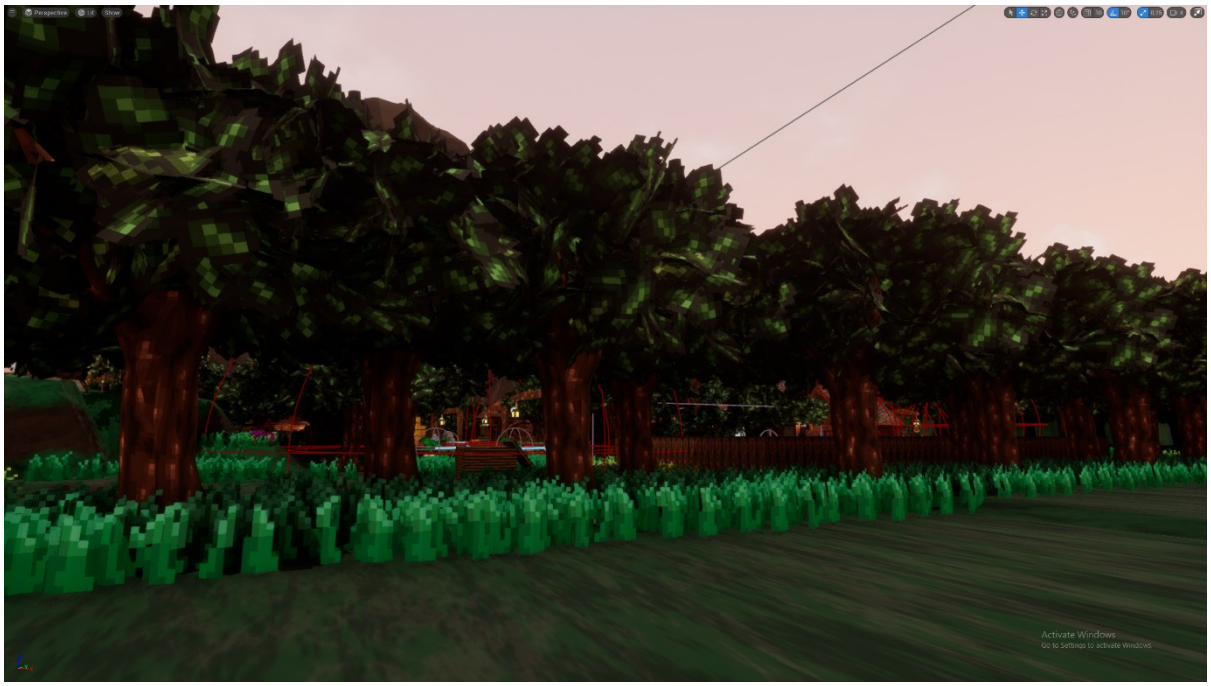
Slika 16 Prikaz stabla sa bazom krošnje



Slika 17 Prikaz stabla sa česticama kose



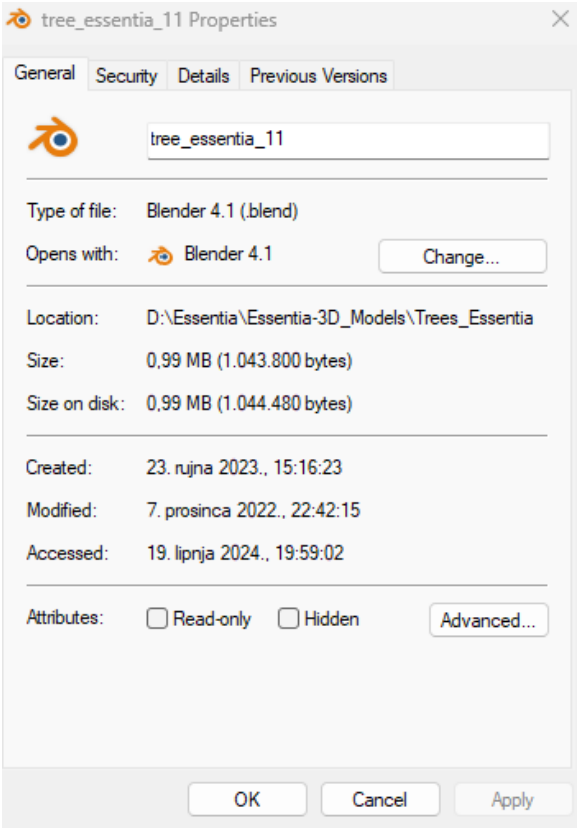
Slika 18 3D model stabla



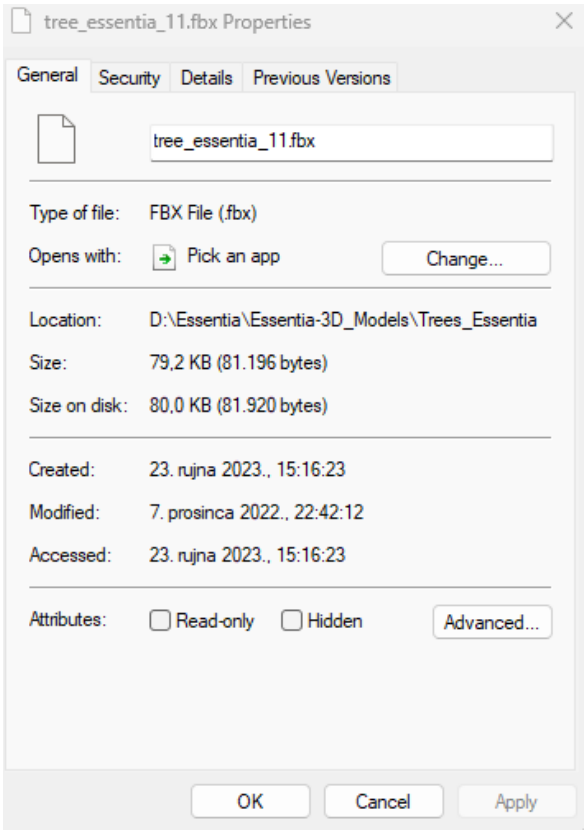
Slika 19 Implementirano drveće u projekt

To nas dovodi to optimizacija modela. Modeli u Blender-u koriste vrhove kako bi povezali vektore između samih sebi te tako stvorili modele koji se prikazuju na ekranu. Što više ima vrhova u modelu

to je model više detaljan. To je glavna razlika između low-poly (modeli male kvalitete) i high-poly (model visoke kvalitete) modela. Svi modeli uključujući i mape su morali koristiti što manje vrhova, to ne znači nužno da su trebali biti manje kvalitete već samo da su se višak vrhova maknuo. Razlog tome je zato što će uvijek biti veliki broj različitih modela na ekranu koje računalo mora konstantno procesirati, tako da se računalu olakša procesiranje i dobiju što veće performanse, važno je da su modeli što manje zahtjevni. Još jedna dobra nuspojava ovoga je da će individualne datoteke manje težiti te će i u konačnici i sama igra sveukupno manje težiti. Naposlijetku je bitno modele eksportati u pravilnom formatu koji zadržava kvalitetu, kompatibilan je sa Unreal Engine-om te ne zauzima previše mjesta. U slučaju videoigre Essentia se koristio za .fbx format, kako on zadovoljava sve uvjete. Usporedba između veličina datoteka kod modela jednog stabla se može vidjeti na slikama Slika 20 i Slika 21



Slika 20 Primjer težine datoteke stabla (u origianlnoj .blend datoteci)

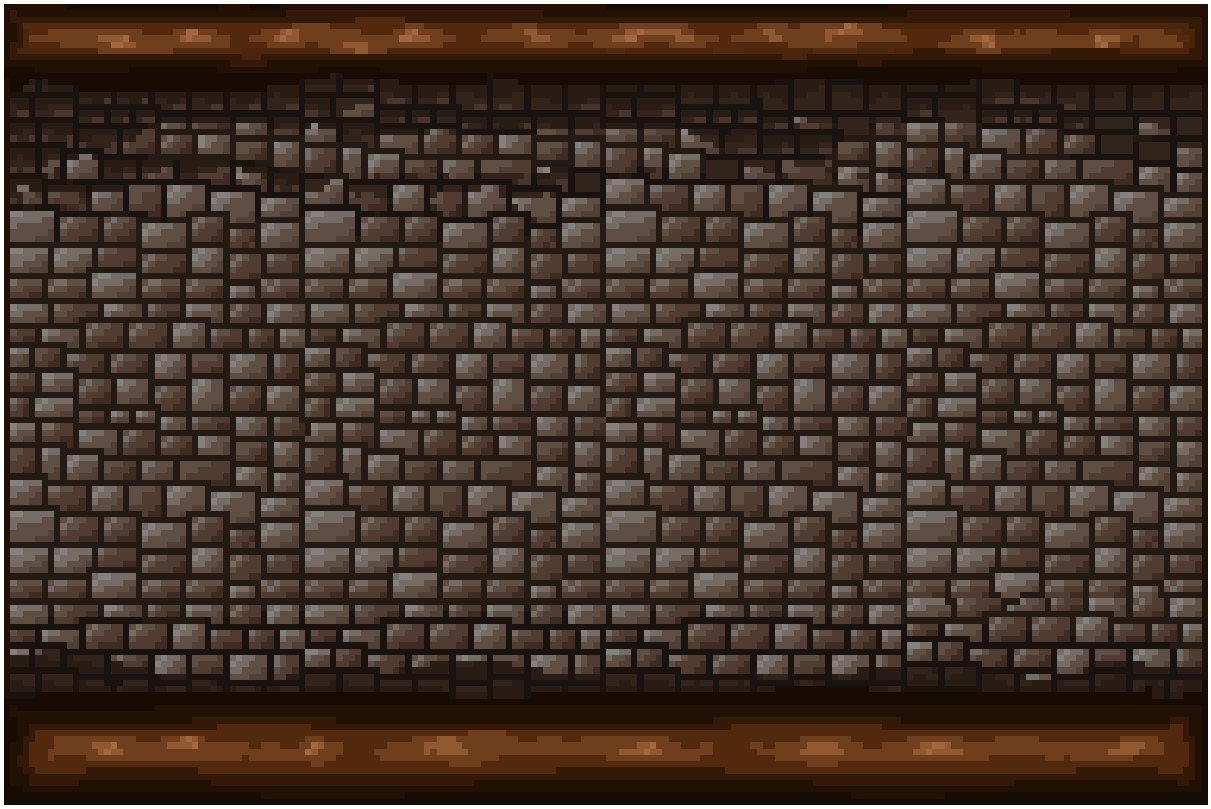


Slika 21 Primjer težine datoteke stabla (u .fbx formatu)

Trenutno u projektu Essentia se nalazi 41 izrađenih 3D modela. Oni su različite veličine i težine, ali principi korišteni u njihovom modeliranju su bili isti kao i prije navedeni. Negdje, doduše, se koristio isti bazični model koji se kasnije samo mijenjao kako bi se dobili drugačiji oblici. To se može vidjeti na primjeru dragulja i stalagmita. Ostali su svi modeli rađeni od nule.

4.3.3. Implementacija 3D modela u projekt

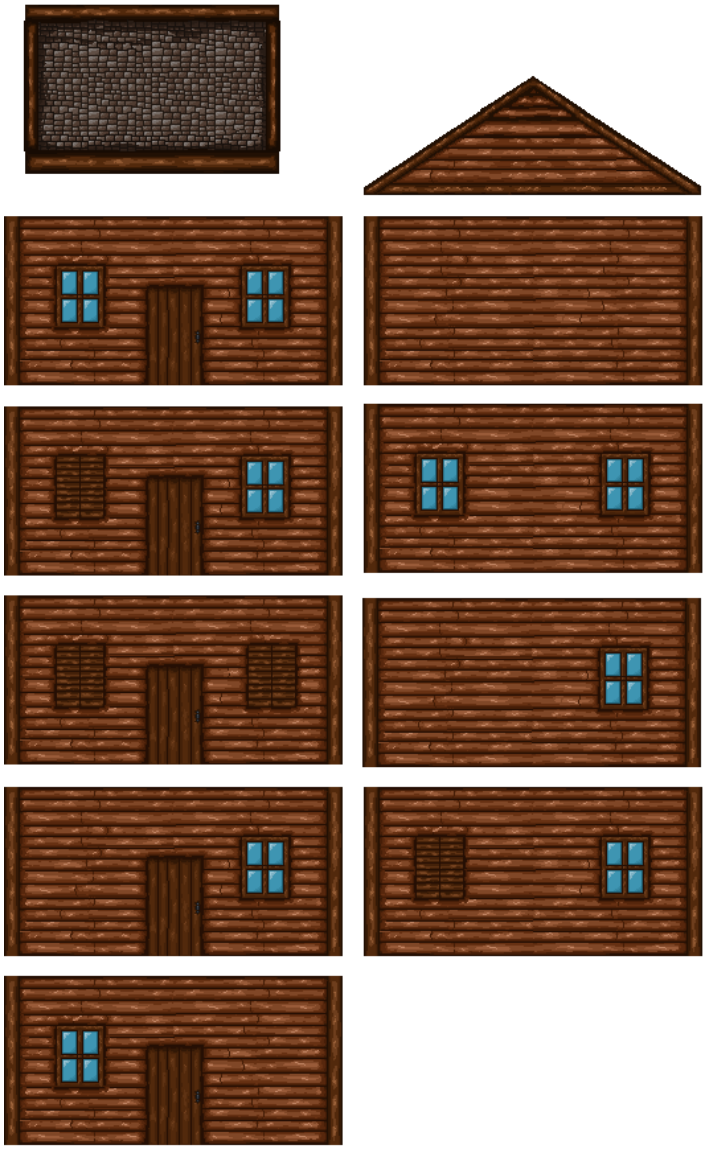
Sama implementacija modela u projekt unutar Unreal Engine-a ne zahtjeva posebne tehnike. Modeli se ubace u projekt tako da se samo povuku unutar odabrane mape u projektu te kako bi se pojavili na ekranu se samo povuku na scenu. Zatim se oni dalje mogu urediti, nijhova pozicija, loakcija, veličina itd. Ovdje je važno napomenuti da su se teksture dodavale unutar Unreal Engine-a, a ne u Blender-u. Razlog tome je jednostavnost i ujednačenost kod rađenja tekstura. Teksture u video igrama su digitalne slike koje se primjenjuju na površine 3D modela kako bi im se dodali detalji i vizualni realizam. One su ključni elementi u stvaranju vizualno privlačnog i uvjerljivog svijeta igre, te značajno doprinose estetskoj kvaliteti i atmosferi igre. Teksture su rađene u programu Aseprite, specijaliziranom za izradu 8 bitnih sprite tekstura.



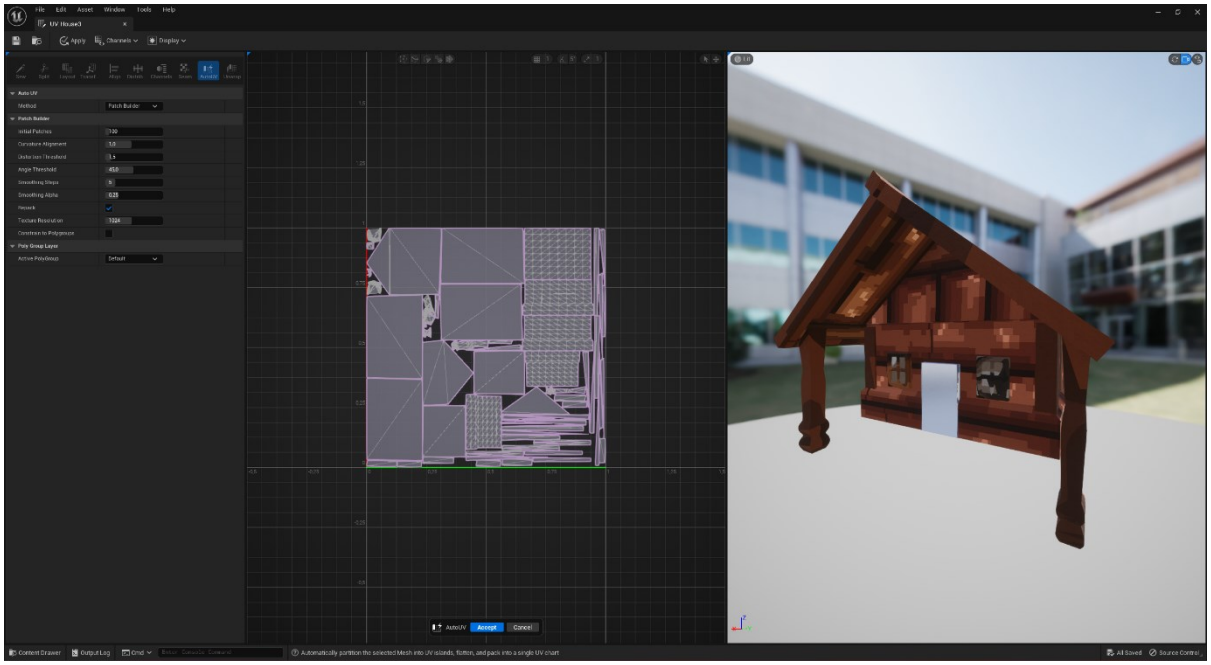
Slika 22 Primjer teksture rađene u Aseprite-u

Sve teksture imaju dimenziju od 48x48 piksela te kada ih se exporta iz programa Aseprite oni su u obliku slike proizvoljnog formata, tipa PNG. Primejr jedne takve teksture se može vidjeti na Slika 22. Dalje za njihovu primjenu na modele je korištena funkcionalnost UV map editora unutar Unreal Engine-a, iako se mogla koristiti i ista funkcionalnost unutar Blender-a. UV mape predstavljaju koordinate koje povezuju trodimenzionalnu površinu modela s dvodimenzionalnom teksturom, omogućujući pravilno apliciranje tekstura na 3D geometriju. Bitno je napomenuti kada se importaju teksture unutar Unreal Engine-a da se primjeni postavka „apply 2D texture settings“ na sve teksture. To makne efekt zamučenosti sa tekstura niske rezolucije, također ako neke teksture imaju efekt prozirnosti na njihovim alpha kanalima (kanal za prikaz boja na računalima koji kontrolira prozirnost) ova postavka ih također primjeni unutar projekta, kod Essentia-e primjerice je to bilo bitno postaviti kod tekstura za lišće. Nakon toga potrebno je primijeniti teksture na modele. Prvotno je potrebno izrezati UV mapu modela i teskture te ručno postaviti teksture da njezini dijelovi pašu i adekvatni su za dijelova modela na koje se stavljaju. UV mape predstavljaju metodu za projiciranje 2D tekstura na 3D modele, omogućujući preciznu kontrolu nad time kako se te teksture "omotavaju" oko modela. UV mape rade tako da transformiraju 3D površinske koordinate modela u 2D prostor teksture. Oznaka "UV" dolazi od osi koje se koriste u ovom 2D prostoru, za razliku od X, Y i Z osi koje se koriste u 3D prostoru. Prilikom izrade UV mapa, svaki poligon 3D modela mapira se na odgovarajuću lokaciju unutar 2D teksture. Ovaj proces uključuje "razmotavanje" 3D modela u 2D prostor, sličan otvaranju kutije u ravninu. Ovaj zadatak često zahtijeva ručno prilagođavanje kako bi se osiguralo da tekstura odgovara precizno svim površinama modela bez izobličenja. Jedan od glavnih izazova kod UV mapiranja je izbjegavanje šavova, područja gdje se tekstura razdvaja i može pokazivati neusklađenosti kada se ponovo omota oko modela. Dizajneri moraju pažljivo postaviti šavove na mjestima gdje će biti manje vidljivi ili ih maskirati teksturama. UV mape se koriste u kombinaciji s

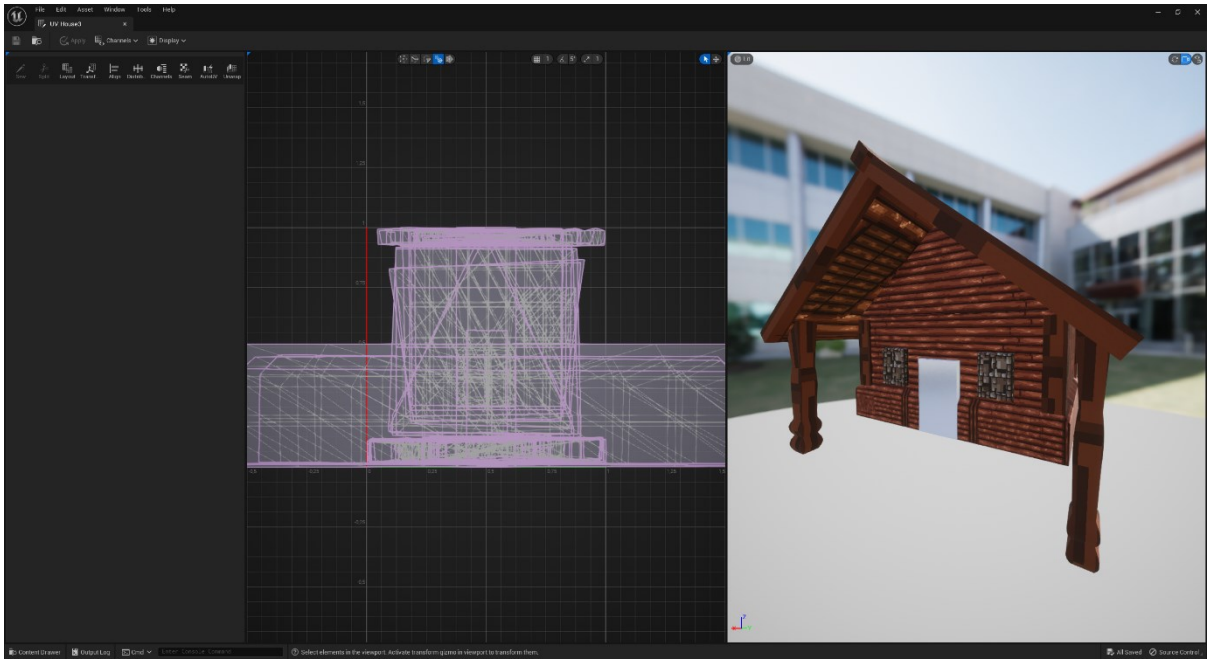
teksturama za definiranje različitih aspekata izgleda modela, uključujući boju (difuznu teksturu), refleksiju (specular mapu), reljef (normalnu mapu) i druge. Ove teksture se zatim koriste u shaderima unutar 3D rendera ili igračkog motora kako bi se postigli vizualno bogati i realistični modeli (Joe, 2024). UV mapiranje se obavlja u UV Editoru, koji je inače dio već postojećih alata poput Blender-a i Unreal Engine-a te unutar njega korisnici mogu ručno prilagoditi pozicije UV koordinata. Ovaj alat omogućuje precizno postavljanje tekstura na model, što je ključno za izradu kvalitetnih 3D modela koji izgledaju realno u različitim svjetlosnim uvjetima i perspektivama. UV mapama se dalje može manipulirati, primjerice mijenjati dimenzije, što je bitno napraviti jer neće svaka UV mapa pasati svakom 3D modelu, već svaku UV mapu treba prilagoditi da paše 3D modelu na kojemu se. Kada se postave željene dimenzije na UV mapu se stavi tekstura, ali kod kompliciranih objekta nije dovoljno samo staviti teksturu na UV mapu jer će tekstura biti nasumično postavljena na model. Stoga je potrebno teksturu razrezati unutar uređivača te precizno odrediti na kojem će dijelu UV mape se nalaziti koji dio teksture što će se prevesti u to da će specifični dijelovi teksture biti prikazani na njima određenim dijelovima na modelu Primjerice tekstura za model kuće, koja se može vidjeti na Slika 23, sadrži u jednom komadu i zid i stup na kutovima kuće. Kada bi se takva tekstura primijenila kakva je rezultat bi bio iskrivljen te ne bi pasao na modelu, kao što se može vidjeti na primjeru na Slika 24 te se isti popravljeni model može vidjeti na Slika 25.



Slika 23 Prikaz teksture korištene za drvenu kuću



Slika 24 Loše teksturiran model kuće unutar uređivača UV mapa



Slika 25 Dobro teksturiran model kuće unutar uređivača UV mapa

Stoga se uzme samo dio teksture koji predstavlja drveni zid, izreže se i primjeni na zid na 3D modelu te se isto napravi i sa stupovima i drugim dijelovima teksture. Ovaj postupak se treba ponoviti za svaki novi model koji se želi teksturirati. Uz to, također i koristimo princip ponavljanja tekstura. To je mogućnost gdje se može podesiti veličina tekstura te ovisno o njihovoj veličini tekstura će se više ili manje puta ponavljati jedna do druge na istom modelu. To omogućuje da se ponovno koriste iste teksture na drugačijim modelima bez potrebe da se crtaju iznova kako bi se prilagodile novom modelu te također omogućuje da različiti dijelovi istog modela sa istom teksturom izgledaju drugačije. Nakon što se podese veličine tekstura da imaju smisla model je gotov te uspješno teksturiran. Primjer teksturiranog i implementiranog modela drvene kuće se može vidjeti na Slika 26.

Naposljetku, osim teksturiranja potrebno je i postaviti collidere, to su svojstva koja omogućuju da se različiti elementi sudaraju unutar igre. Primjerice kod drveća to će služiti da igrač i NPC-evi ne mogu proći kroz modele, dok kod mosta to će biti korišteno da igrač i NPC-evi mogu prelaziti preko njega. Nakon svih ovih postupaka objekt je uspješno implementiran u projekt te se može proizvoljno staviti na scenu, bitno je napomenuti da se isti objekt može postaviti na scenu koliko god puta programer to želi.

Slika 26 Prikaz teksturiranog 3D modela kuće

4.3.4. 3D mape u videoigrama

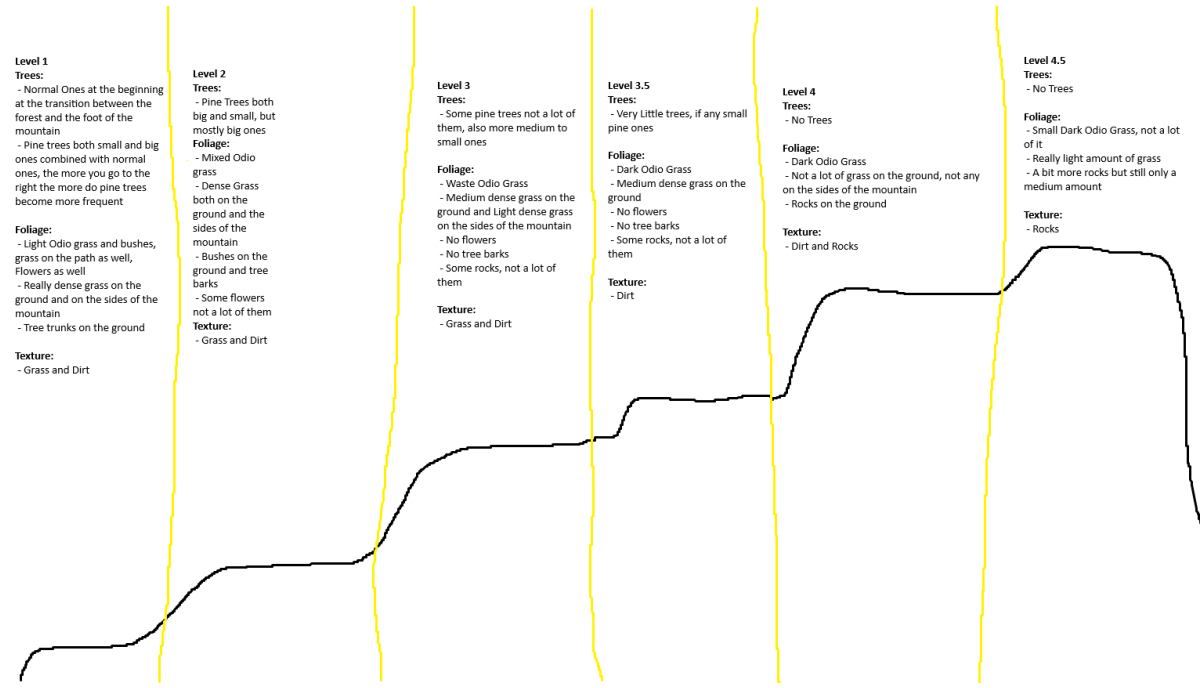
U videoigrama, razina (u nekim starijim igrama naziva se i karta, misija, pozornica, staza ili runda) je svaki prostor dostupan igraču tijekom ispunjavanja cilja. Razine u video igrama obično imaju progresivno rastuće poteškoće unutar razine, kako bi privukle igrače s različitim razinama vještina. Svaka razina može predstavljati nove koncepte i izazove za održavanje visokog interesa igrača.

U igrama s linearnim napredovanjem, razine su područja većeg svijeta. Igre također mogu sadržavati međusobno povezane razine, koje predstavljaju lokacije. Iako je izazov u igri često poraziti neku vrstu lika, razine su ponekad dizajnirane s izazovom kretanja, kao što je pronalazak pravog puta, oblik poligona s preprekama i slično. Igrači moraju procijeniti udaljenost između platformi ili izbočina i sigurno skočiti između njih kako bi došli do sljedećeg područja. Ove zagonetke mogu usporiti zamah za igrače brzih akcijskih igara.

Dizajn razina ili dizajn okruženja, je disciplina razvoja igara koja uključuje izradu razina videoigara — lokaliteta, faza ili misija. To se obično radi pomoću uređivača razina, softvera dizajniranog za izgradnju razina; međutim, neke igre imaju ugrađene alate za uređivanje razina.

U slučaju videoigre Essentia mape su bile rađene unutar programa Blender te su korištene iste funkcionalnosti kao i kod razvoja 3D modela. Principi kod njihovog stvaranja su doduše različiti. Kako su 3D mape tlo kojima se igrač kreće, one su ključne u stvaranju imerzivnih iskustava. One su također glavna vodilja koja govori igraču kako da se kreću kroz svijet.

Stoga je prvi korak u stvaranju uvjerljivih 3D mapa njihova konceptualizacija. Ovo zvuči kao da bi mogao biti dio faze predprodukcije, ali je zapravo dio faze produkcije jer se individualne mape stvaraju tijekom razvoja igre te u puno slučaja izgled sljedeće ovisi o izgledu prijašnje mape. Stoga je bilo potrebno prvo napisati što se traži od svake mape te njezin planirani izgled. Za primjer će se uzeti mapa planine Spes koja se može vidjeti na Slika 28. Prvotno se napravio grubi nacrt izgleda mape na koji su se napisali traženi detalji. Kako je u pitanju planina, ona se podijelila na različite razine te su se za svaku razinu definirali traženi detalji poput vrste vegetacije njene gustoće, oblik teksture i drugo. Ovdje je također bio definiran i grub generalni put kojim se igrač mora kretati kako bi došao do cilja. Točno definirani detalji za svaki dio mape se može vidjeti na Slika 27. Vidi se da je planina podijeljena na nekoliko razina (eng. levels), te svaka razina predstavlja „platformu“ na kojoj će zbog nekog razloga igrač morati stati. Svaka razina ima točno definirano koje teksture mora koristiti, koje modele te koja vegetacija može biti prisutna. Vidljivo je da što se igrač uspinje više u visinu da vegetacija postaje sve oskudnija te tlo prelazi iz trave u šljunak. Ovo je sve korisno definirati kako bi se olakšao proces modeliranja i implementacije mapa u projekt.



Slika 27 Detalji izrade mape planine Spes

Optimizacija je ključna faza u stvaranju 3D mapa kako bi igre mogle glatko raditi na različitim hardverskim platformama. Dizajneri koriste tehnike poput smanjenja broja poligona, prilagodljivih razina detalja (LOD) i korištenja tekstura niske rezolucije na udaljenim objektima kako bi smanjili

zahtjeve za računalne resurse. Također, primjenjuju se napredne metode renderiranja poput occlusion cullinga i mipmappinga kako bi se postigla ravnoteža između vizualne kvalitete i performansi.

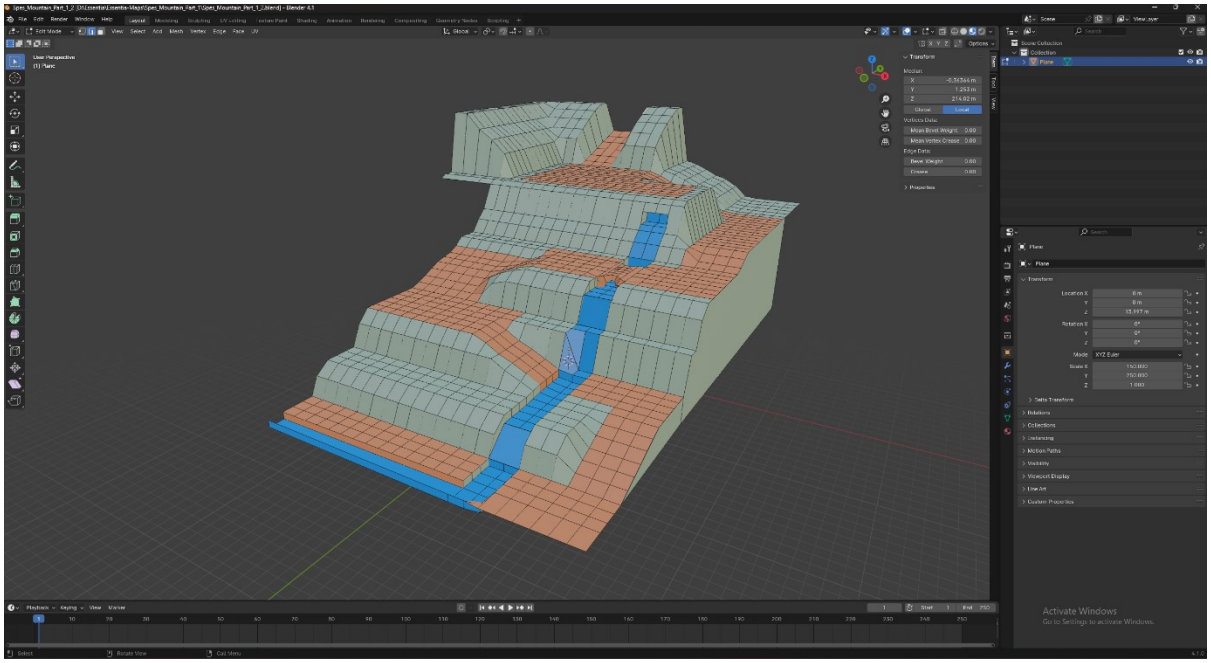
Interaktivnost je još jedan važan aspekt 3D mapa u videoigrama. Mape često sadrže dinamične elemente poput vrata koja se mogu otvoriti, skrivenih prolaza, pokretnih platformi i drugih interaktivnih objekata koji doprinose gameplayu. Fizičke simulacije, kao što su gravitacija, sudari i destrukcija okruženja, također igraju važnu ulogu u stvaranju uvjerljivih i dinamičnih svjetova. Sve tri prikazane mape videoigre Essentia sadrže takve elemente. Tok rijeke, ne dodaje samo osjećaj dinamičnosti svijetu već i služi kao vizualna poveznica između dvije mape kako se ona prostire između njih. Tu su također prisutne i fizičke simulacije padanja lišća kako bi se upotpunio svijet. Naposlijetku, također su implementirane prepreke koje sprječavaju igrača na određenim dijelovima da idu na sljedeći dio mape te tek nakon što odradi neku misiju se može nastaviti kretati njom.

4.3.5. Izrada 3D Mapa za Videoigru Essentia

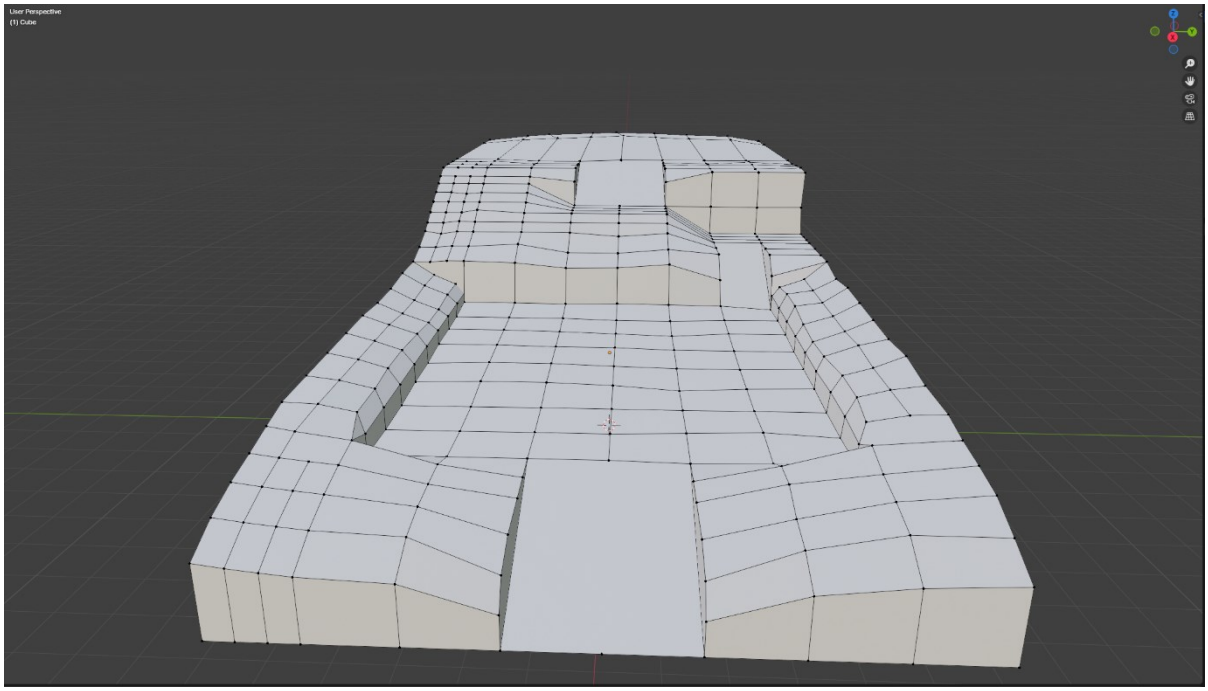
Kao što je već bilo napomenuto, tehnike korištene kod izrade 3D modela također su korištene i kod izrade mapa. To također uključuje i tehnike optimizacije. Osim toga kod mapa se treba paziti na još jednu bitnu stvar, raspodjelu puteva. Kako su mape doslovni temelj na kojem stoji cijeli svijet videoigre, jako je bitan način na koji su one izmodelirane. Drugačiji tip igre također zahtjeva i drugačiji tip mape. Na primjeru mape koja se može vidjeti na Slika 28 je bilo rečeno da mapa mora sadržavati 4 razina visine sa njihovim prijelazima. Ova mapa predstavlja planinu te osim povećanja visina također je bilo napomenuto kako vegetacija mora postajati sve više i više oskudna te teksture moraju prijeći iz travnatih u pretežito zemljaste te zatim u kamenite. Ovo igraču daje dojam da se uspinje na velike visine. Također je bilo potreno definirati jasan put napretka što je ostvareno sa dizajnom samog hodajućeg puta koji je dostupan igraču. Također se treba ostaviti i dovoljno prostora za istraživanje igraču ako želi pronaći tajne mape na kojoj se nalazi. Primjer na Slika 29 prikazuje sljedeću mapu. Dobro je napomenuti da je dizajn druge mape prilagođen da izgleda kao da je prirodno napredovanje od prijašnje. Ova sljedeća mapa prikazuje vrh planine te je također i mjesto gdje će se igrač suočiti sa glavnim neprijateljem ove razine, to zahtijeva drugačiji dizajn mape koji sada ima više otvorenog prostora, ali također i mogućnost daljnjeg napredovanja kako je to zahtijevano od strane priče. Ovo je filozofija oko dizajna mapa koje imaju svrhu vođenja igrača. Na Slika 30 je prikazan drugi tip mape koji je više otvorenog tipa gdje igrač mora pronaći vlastiti put te više proizvoljno može istraživati mapu. Kod takvih tipova mapa bitno je više da mapa i postavljeni objekti imaju smisla za svijet u kojem se nalaze. Kao što je bilo napomenuto u prijašnjem poglavlju, optimizacija je također ključna faza izrade 3D mapa. Stoga je bilo potrebno smanjiti što veći broj nepotrebnih vrhova i dijelova koji neće biti vidljivi te su također kao i kod 3D modela, mape exportane iz Blender-a u fbx formatu kako bi zauzimale što manje prostora i izgleda što veće kvalitete.

Za početni dio Essentia-e je bilo potrebno izraditi 4 mape:

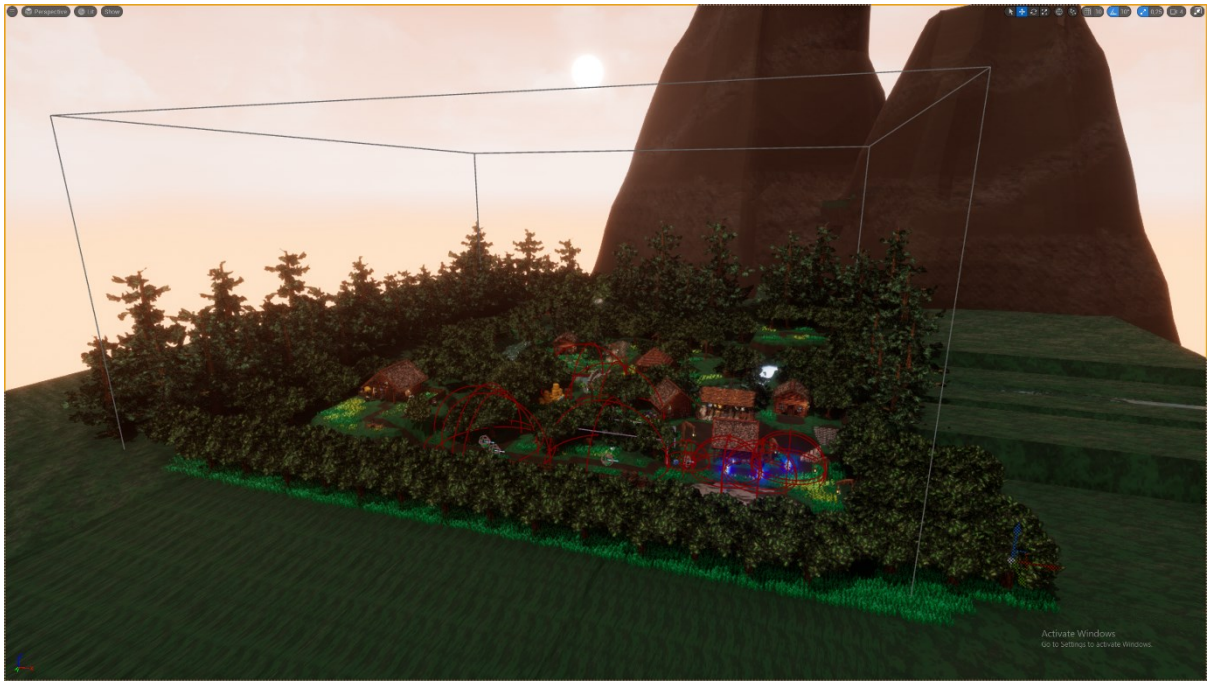
- Prva mapa predstavljala kuću glavnog lika Yel-a, u kojoj se igrač probudi kada započne igru.
- Druga je mapa otvorenog tipa koja predstavlja rodno selo glavnog lika.
- Treća je podnožje i većinski dio planine do čijeg vrha igrač pokušava doći u početnom dijelu igre.
- Četvrta i zadnja mapa predstavlja vrh planine te ona simbolizira kraj početnog dijela igre.



Slika 28 Prikaz prvog dijela mape planine unutar Blender-a



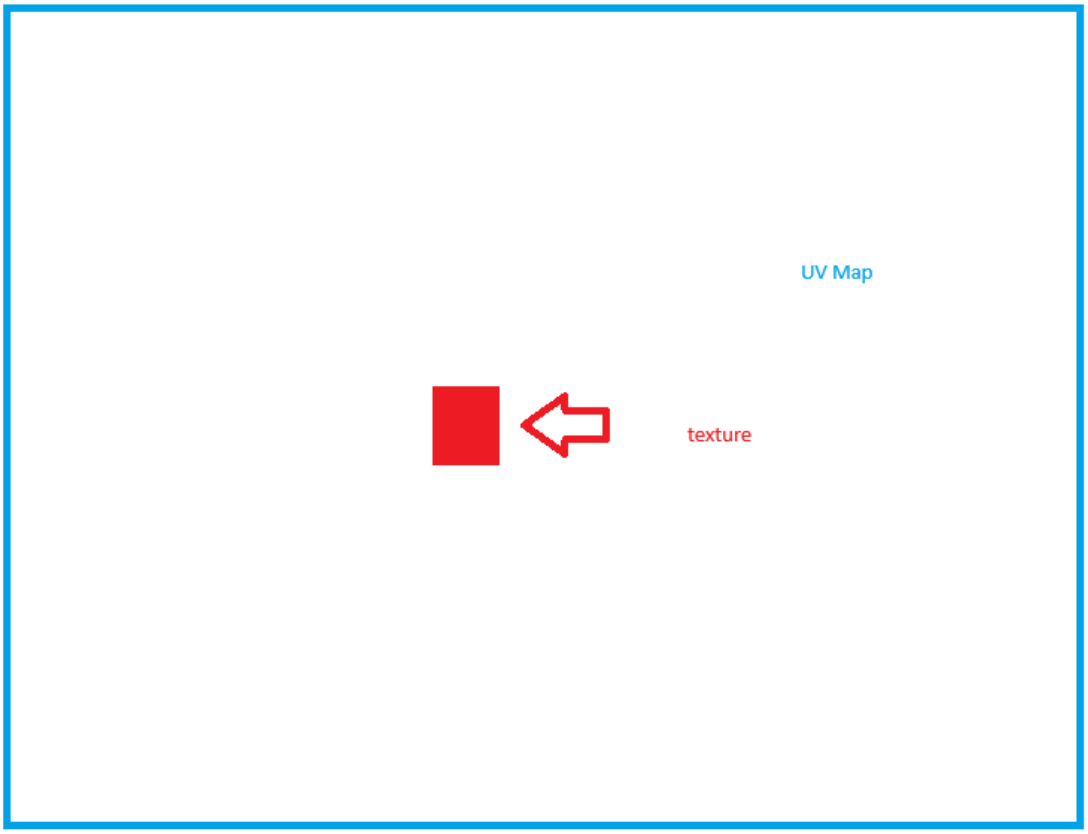
Slika 29 Prikaz drugog dijela mape planine unutar Blender-a



Slika 30 Prikaz mape sela koja je otvorenijeg tipa

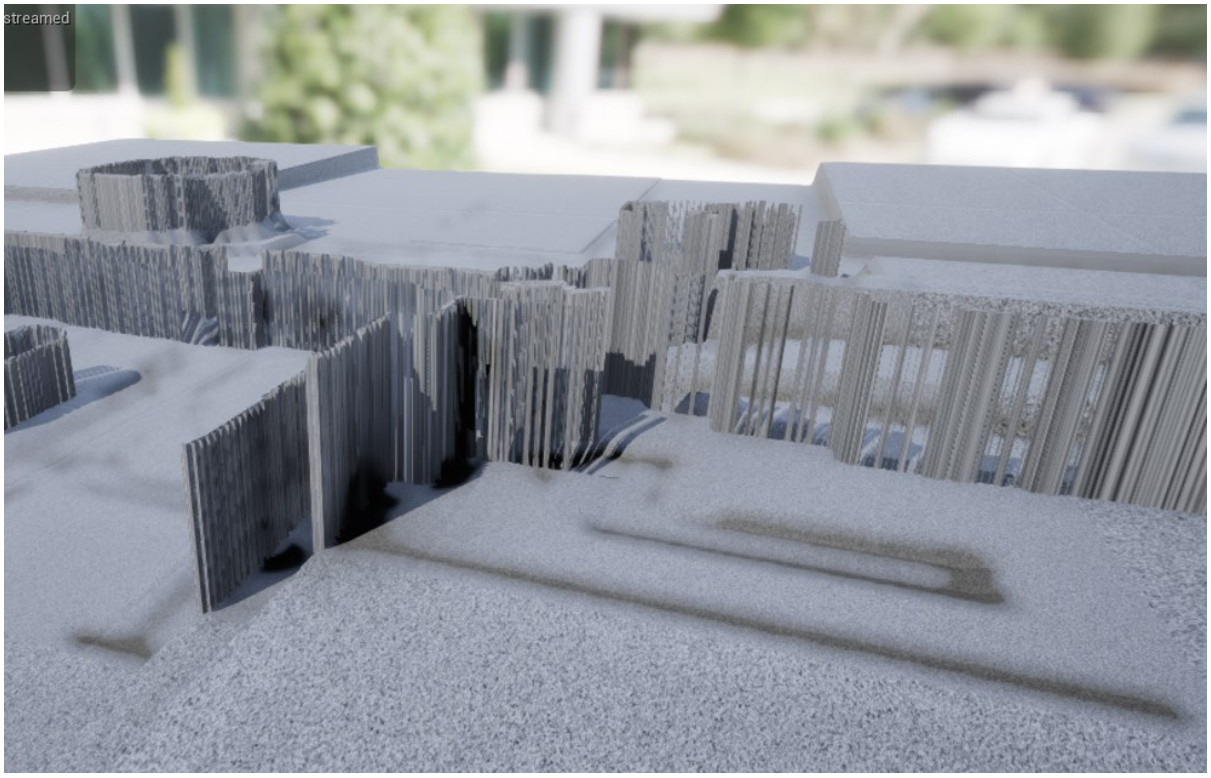
4.3.6. Implementacija 3D Mapa u Projekt

Priča implementacije 3D mapa u projekt je ista kao i kod implementacije 3D modela. Doduše, ima razlika kod implementacija tekstura. Proces je većinom isti kao i kod teksturiranja 3D modela, ali kada se rade UV mape potrebno je postaviti veličinu UV mapu jako velikom dok je veličinu teksture potrebni postaviti malom. Vizualizacija toga se može vidjeti na Slika 31.



Slika 31 Vizualizacija razlike veličine UV mape i teksture kod 3D mapa

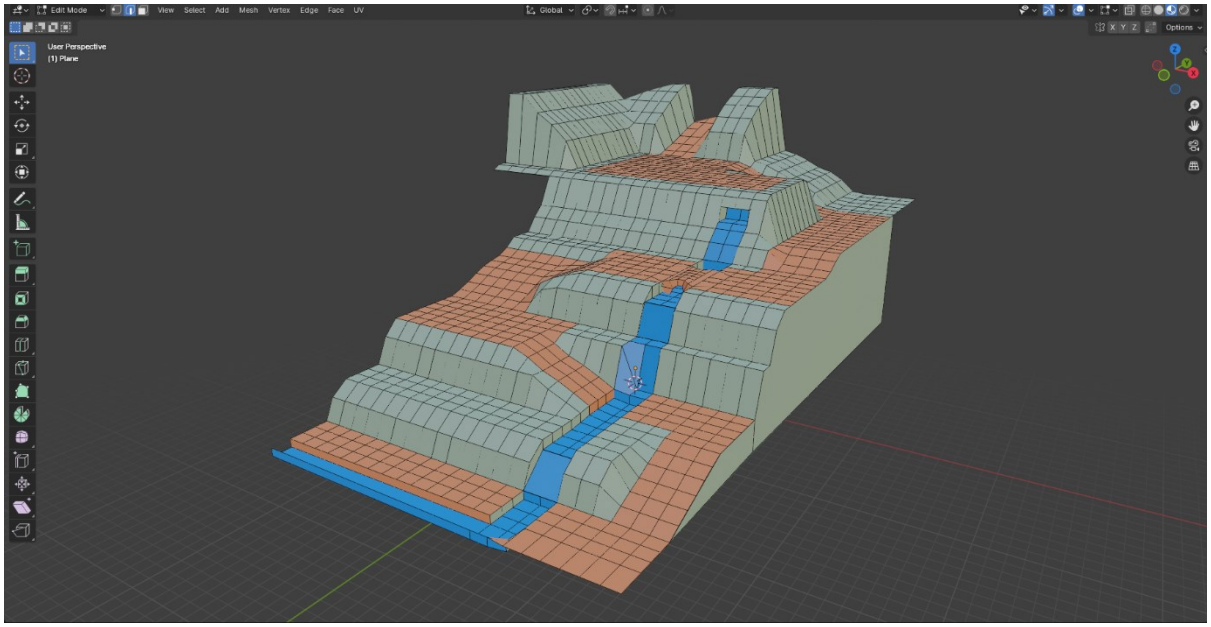
Kod implementacija 3D mapa unutar Unreal Engine-a je dodatno potrebno staviti polje ili grid koji govori Unreal Engine-u na kojim je sve područjima dopušteno igračima i NPC-evima da se kreću. Osim toga kako igrač ne bi slučajno pao sa ruba mapa ili visokih dijelova mape, bilo je potrebno unutar Unreal Engine-a upaliti postavku koja se zove „use complex collisions as simple“. Ova postavka postavlja granice koje su nevidljive igraču i programeru, ali unutar Unreal Engine-a, ovisno o izgledu mape, postavi te granice. Vizualizacija tih granica je vidljiva na Slika 32.



Slika 32 Vizualizacija granica za pad sa mapa unutar Unreal Engine-a 5

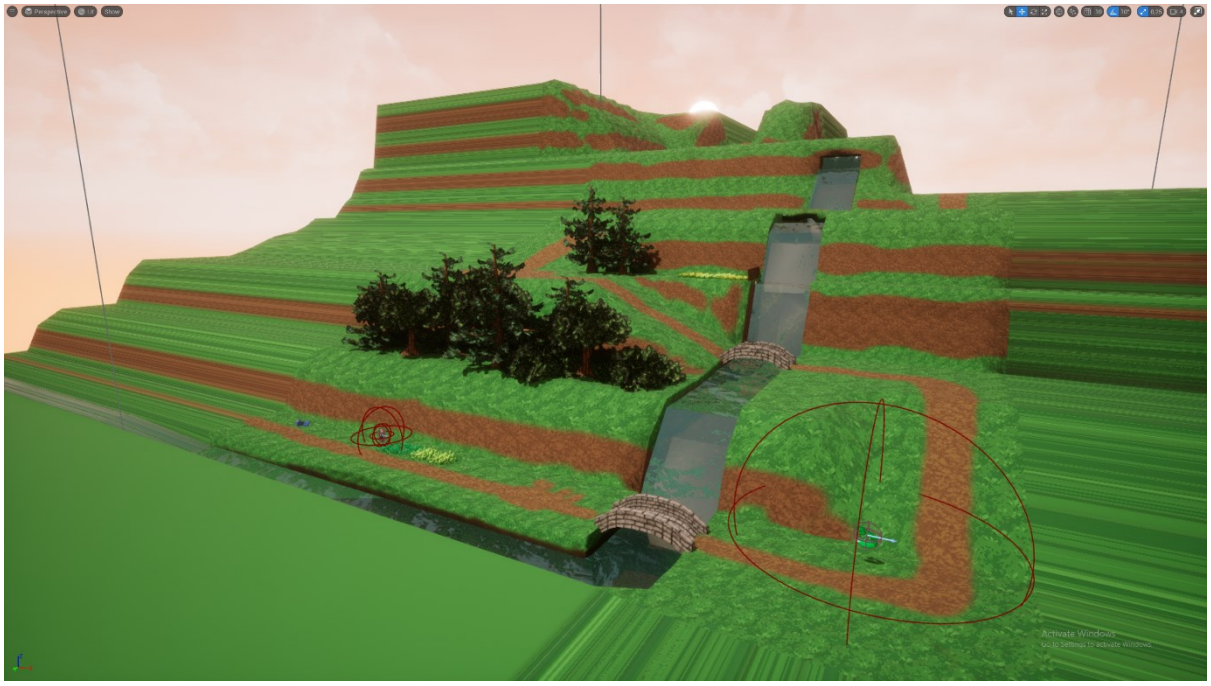
Dalje je potrebno popuniti mape sa 3D objektima kako bi se stvorila potpuna scena te dao dojam živućeg svijeta. Jedan trik koji olakšava optimizaciju je da se ne napravi velika mapa koja zauzima cijelo vidno polje igrača tijekom igranja, već se, primjerice za potrebe ovog projekta napravilo nešto što bi se moglo zvati samo dio mape te se onda populirao ekran drvećem i drugim asset-ima tako da igrač dokle god da se kreće kroz svijet ne može vidjeti rubove mapa. To je vidljivo na Slika 30 gdje igrač, zbog postave kamere nikad neće vidjet dalje od ruba šume. Što se tiče filozofije postavljanja 3D modela u scenu, potrebno je modele postaviti tako da prate logiku mape. Također je bitno da modeli stvaraju konkretan volumen, primjerice kod stvaranja šume nije dobro postaviti drveća na velikom razmaku ako bi šuma na sceni trebala biti gusta. Kod mapa koje vode igrača kroz svijet, kao u primjeru prve mape planine, bitno je da se ne zamuti put koji igrač treba pratiti postavljanjem previše objekta na njega, isto vrijedi i za tajne dijelove mape, oni trebaju biti skriveni, ali svejedno dostupni igraču. Naposlijetku, dobra je praksa staviti 3D modele koji upotpunjuju povijest svijeta ili daju veći kontekst priči na to adekvatna mjesta, primjerice postavljen stariji most koji se raspada i ostatci zgrada kako bi se dao dojam da su ljudi prije mnogo godine živjeli na ovom području. Na Slika 33, Slika 34 i Slika 35 se vidi postepen prijelaz od ne implementirane mape planine u potpuno implementiranu i popunjenu mapu unutar Unreal Engine-a 5.

Također ćemo još jednom ukratko proći kroz proces implementacije 3D mapa u projekt. Prvotno se mapa izradi unutar Blender-a te označe svi potrebni dijelovi i putevi kao što je vidljivo na Slika 33.



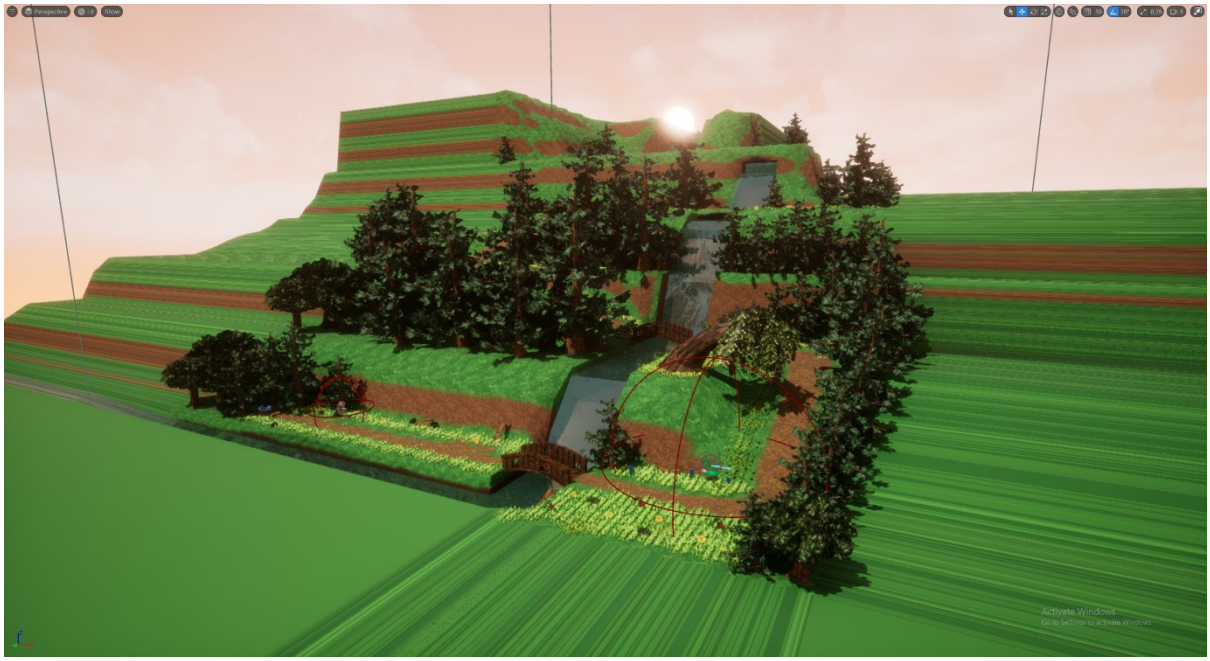
Slika 33 Prikaz ne implementirane mape planine unutar Blender-a

Zatim se ta mapa unese unutar Unreal Engine-a gdje se na njoj primjene sve potrebne postavke te se ona teksturira, ponovna napomena teksturiranje se moglo izvršiti i unutar Blender-a, vidljivo na Slika 34.



Slika 34 Prikaz u djelomično implementirane mape unutar Unreal Engine-a 5

Naposlijetku se dodaju 3D modeli kako bio se upotpunila scena i pomoglo voditi igrača kroz mapu, kao što je vidljivo na Slika 35.



Slika 35 Prikaz u potpunosti implementirane mape planine unutar Unreal Engine-a 5

4.4. Realizacija umjetne inteligencije

Realizacija umjetne inteligencije zahtjeva znanje programiranja te rad sa blueprint sustavom unutar Unreal Engine-a, isti se efekt također mogao postignuti pisanjem koda u C++ programskom jeziku, ali ova varijanta olakšava stvari i čini kod malo preglednijim. Blueprints sustav unutar Unreal Engine-a 5 je vizualni skriptni alat koji omogućava programerima i dizajnerima da kreiraju igru bez potrebe za pisanjem tradicionalnog koda. Ovaj sustav koristi node-based (čvorni) pristup za definiranje logike igre, što olakšava razumijevanje i modifikaciju koda, posebno za one koji nemaju iskustva s programiranjem.

4.4.1. Umjetna inteligencija u igrama

Umjetna inteligencija (UI) u videoigrama igra ključnu ulogu u stvaranju dinamičnih, izazovnih i interaktivnih iskustava za igrače. UI omogućuje računalno kontroliranim likovima (NPC-ima) i elementima igre da se ponašaju na realističan i inteligentan način, pridonoseći dubljem uranjanju i složenijem gameplay-u. Non Player Character (NPC) je bilo koji lik u igri koji nije pod kontrolom igrača. U videoigrama to obično znači lik kojim upravlja računalo, koji ima unaprijed određen skup ponašanja koji će potencijalno utjecati na igranje, ali neće nužno biti proizvod prave umjetne inteligencije (Yannakakis, 2018).

Essentia je akcjska igra sa velikim fokusom na priču. Stoga je bilo potrebno implementirati dvije varijante iste metode umjetne inteligencije ovisno o vrsti likova, koje se dalje mogu granati na podvrste. Jedna će se varijanta odnositi na dobre likove, a druga na loše likove. Prva vrsta se odnosi na druželjubive NPC-eve. To su likovi unutar igre koji hodaju po sceni i generalno služe kako bi upotpunili svijet i dali mu života. Igrač sa ovim NPC-evima može komunicirati i razgovarati, ali ih ne može napadati. Razina i dubina komunikacije se mijenja od lika do lika. Većina likova će reći samo nekoliko rečenica koje se ponavljaju, dok bitniji likovi, primjerice oni koji daju misije i zadatke igračima će imati više dijaloga. Iako je Essentia tipa igre RPG, ona neće u sebi imati mogućnost odabira različitih odgovora i drugačijih razgovora koji mijenjaju stanje svijeta i priče, kako je fokus više na linearnoj priči koja ima definiran konkretan početak i kraj. Drugi tip umjetne inteligencije, ili onaj korišten za loše NPC-eve se odnosi na negativce. Njihova svrha je napadati igrača te ga pokušati spriječiti da nastavi sa napretkom kroz igru. Oni se mogu podijeliti na jednostavne neprijatelje, kojih ima više i slabiji su te na glavne negativce ili glavne šefove koju su puno moćniji od jednostavnih i treba koristiti efektivne strategije kako bise oni porazili. Negativci također imaju mogućnost komunikacije sa igračem, ali u manjoj sferi te oni također patroliraju svijetom, jedino što kada primjete igrača ga počnu napadati. Glavni šefovi su nešto drugačiji kako su oni više povezani uz priču i misije te imaju više dijaloga od običnih neprijatelja te se kroz igru ponekad znaju pojaviti u skriptiranim segmentima u kojima ne moraju napadati igrača.

U videoigrama, umjetna inteligencija (UI) može se podijeliti na različite tipove, svaki s različitim karakteristikama i svrhom. Evo nekoliko tipova UI-a koji se često koriste:

1. Skriptirana UI (Scripted AI):Skriptirana UI je najosnovniji oblik umjetne inteligencije u videoigrama. Ova vrsta UI-a koristi unaprijed definirane skripte ili skupove pravila kako bi regulirala ponašanje računalno kontroliranih likova (NPC-ova) ili neprijatelja. Primjerice, NPC-ovi mogu slijediti jednostavne putanje ili izvoditi osnovne akcije poput napada na igrača kad su u blizini (Lou, 2017) (Codex, 2023).

2. Reaktivna UI (Reactive AI): Reaktivna UI reagira na trenutne uvjete u igri i ponaša se na temelju trenutnog stanja. Ova vrsta UI-a koristi skup pravila ili heuristika za donošenje brzih odluka o tome kako reagirati na igračeve akcije ili promjene u okruženju. Na primjer, neprijatelj može reagirati na napad igrača pokušavajući se obraniti ili pobjeći (Urwin, 2023).
3. Lokalna UI (Local AI): Lokalna UI fokusira se na ponašanje neprijatelja ili NPC-ova unutar njihovog neposrednog okruženja. Ova vrsta UI-a omogućuje NPC-ovima da donose odluke i akcije koje su relevantne za njihov trenutni prostor djelovanja, poput reagiranja na igračevo prisustvo ili prepoznavanja okolnih opasnosti (Lou, 2017).
4. Sistemska UI (Systemic AI): Sistemska UI omogućuje NPC-ovima ili neprijateljima da interakciju s različitim dijelovima igre ili sustava. Ova vrsta UI-a može uključivati složenije algoritme koji omogućuju NPC-ovima da surađuju, komuniciraju ili se nadopunjuju jedni s drugima u skladu s ciljevima igre (Codex, 2023).
5. Adaptivna UI (Adaptive AI): Adaptivna UI koristi tehnike strojnog učenja ili evolucijskih algoritama kako bi NPC-ovi ili neprijatelji učili i prilagođavali se tijekom vremena na temelju iskustava igrača. Ova vrsta UI-a može optimizirati strategije, prilagoditi težinu igre ili poboljšati učinkovitost NPC-ova na temelju povratnih informacija iz stvarnog vremena (Urwin, 2023).

Svaki od ovih tipova umjetne inteligencije ima svoje prednosti i ograničenja te se može koristiti pojedinačno ili u kombinaciji kako bi se postigla željena razina kompleksnosti i dinamičnosti u videoigrama. Kombinacija različitih vrsta UI-a može stvoriti bogatije i autentičnije iskustvo za igrače, pružajući im izazovne protivnike i nepredvidljive situacije unutar igračkog svijeta. Za potrebe našeg projekta mi smo se odlučili za korištenje lokalnog i skriptiranog tipa umjetne inteligencije. Razlog tome je prvotno zbog jednostavnosti implementacije obiju vrsta, a drugo, niej bilo potrebe za drugim tipovima, odnosno ne bi ih imali gdje smisleno iskoristiti. Skriptirani tip AI-a je bio korišten kod stvaranja druželjubivih NPC-eva, kako oni ne trebaju vršiti neke kompleksne poslove, dok je lokalni tip AI-a bio korišten u stvaranju neprijatelja jer ovaj tip omogućuje da neprijatelji reagiraju na igračeve akcije.

4.4.3. Metode implementacije umjetne inteligencije

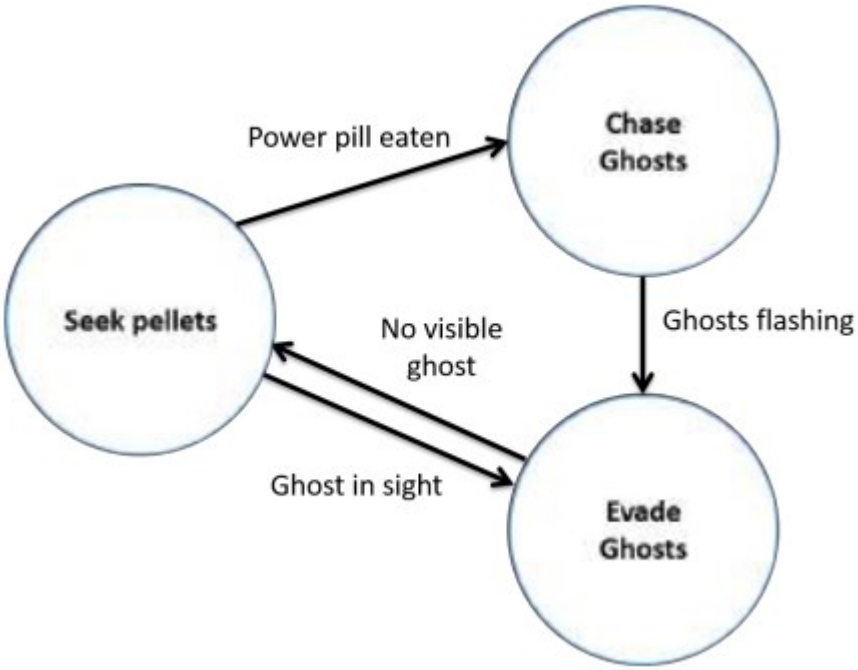
Klasične metode implementacije umjetne inteligencije (UI) u videoigrama obuhvaćaju različite tehnike i algoritme koji omogućuju računalno kontroliranim likovima (NPC-ovima) ili neprijateljima da donose odluke i ponašaju se na inteligentan način (Millington, 2009). Evo nekoliko klasičnih metoda:

1. **Konačni automat (Finite State Machine - FSM)**
2. **Stablo ponašanja**
3. ***A* algoritam***
4. **Minimax algoritam**
5. **Genetski algoritmi**
6. **Napredne tehnike strojnog učenja**

Klasične metode implementacije UI-a u videoigrama pružaju različite alate i tehnike za modeliranje inteligentnog ponašanja NPC-ova, prilagođavajući se specifičnim zahtjevima igre i vrsti gameplay-a. Kombinacijom ovih metoda može se postići željena razina kompleksnosti i realističnosti u interakciji igrača s virtualnim svijetom te naravno ne treba se koristiti samo jedna metoda u jednom projektu.

Od spomenutih metoda unutar projekta razvoja videoigre Essentia se koristila metoda konačnih automata i stablo ponašanja.

Konačni automat je jedna od najosnovnijih i najčešće korištenih metoda za implementaciju AI-a u video igrama. Ova tehnika koristi definirane stanje i prijelaze između njih kako bi modelirala ponašanje likova. Svako stanje predstavlja određeno ponašanje ili aktivnost (npr. napad, obrana, patrole), a prijelazi između stanja se događaju na temelju određenih uvjeta ili događaja u igri (npr. igračeva blizina ili promjena u okruženju). On se zasniva na matematičkom modelu računanja. To je apstraktni stroj koji može biti u točno jednom od konačnog broja stanja u bilo kojem trenutku. FSM se može promijeniti iz jednog stanja u drugo kao odgovor na neke ulaze; promjena iz jednog stanja u drugo naziva se prijelaz. FSM je definiran popisom svojih stanja, svojim početnim stanjem i ulazima koji pokreću svaki prijelaz. Ova tehnika daje mogućnost radnji jednostavnih algoritama koji će vršiti različite zadatke, ali također omogućuje da arhitektura umjetne inteligencije ima sličnu strukturu između različitih tipova NPC-eva i neprijatelja, što olakšava razvoj i uštedi na vremenu. Primjer jednog jednostavnog konačnog automata korišten za kontroliranje lika Ms Pac-Man iz istoimene video igre se može vidjeti na Slika 36.



Slika 36 Pojednostavljeni FSM primjer visoke razine za kontrolu Ms Pac-Man (Yannakakis, 2018).

Dalje u posebnoj blueprint datoteci je napravljeno stablo ponašanja. Stablo ponašanja je sustav stručnog znanja koji, slično FSM-u, modelira prijelaze između konačnog skupa zadataka (ili ponašanja). Snaga stabla ponašanja u usporedbi s konačnim automatima je njihova modularnost: ako su dobro dizajnirani, mogu dati složena ponašanja sastavljena od jednostavnih zadataka. Glavna razlika između stabla ponašanja i konačnih automata je da su sastavljeni od ponašanja, a ne od stanja. Kao i kod automata s konačnim stanjem, stabla ponašanja je lako dizajnirati, testirati i ispravljati, što ih je učinilo dominantnima na sceni razvoja igara. Stablo ponašanja koristi strukturu stabla s korijenskim čvorom i određenim brojem nadređenih i odgovarajućih podređenih čvorova koji predstavljaju ponašanja, primjer vidljiv na Slika 37. Prelaženje kroz stablo ponašanja počinje od

korijena. Zatim se aktivira izvršenje parova roditelj-dijete kao što je označeno u stablu. Dijete može roditelju vratiti sljedeće vrijednosti u unaprijed određenim vremenskim koracima (kvačicama): trčanje ako je ponašanje još uvijek aktivno, uspjeh ako je ponašanje dovršeno, neuspjeh ako ponašanje nije uspjelo. Stabla ponašanja se sastoje od tri vrste čvorova: niz, selektor i dekorater (Yannakakis, 2018).

- **Niz** (vidljiv kao plavi pravokutnik na Slika 37): ako ponašanje djeteta uspije, slijed se nastavlja i na kraju roditeljski čvor uspijeva ako su sva ponašanja djeteta uspješna; inače sekvenca ne uspijeva.
- **Selektor** (vidljiv kao crveni zaobljeni pravokutnik na Slika 37): postoje dvije glavne vrste selektorskih čvorova: selektori vjerojatnosti i selektori prioriteta. Kada se koristi selektor vjerojatnosti, ponašanja djeteta odabiru se na temelju vjerojatnosti roditelj-dijete koje je postavio BT dizajner. S druge strane, ako se koriste selektori prioriteta, ponašanja djece se poredaju na popisu i isprobavaju jedno za drugim. Bez obzira na korištenu vrstu selektora, ako je dijete uspješno, selektor je uspješan. Ako ponašanje djeteta ne uspije, odabire se sljedeće dijete u redoslijedu (u selektorima prioriteta) ili selektor ne uspije (u selektorima vjerojatnosti).
- **Dekorator** (vidljiv kao ljubičasti šesterokut na Slika 37): čvor dekoratera dodaje složenost i povećava kapacitet ponašanja jednog djeteta. Primjeri dekoratera uključuju broj pokretanja ponašanja djeteta ili vrijeme dano ponašanju djeteta da dovrši zadatak.

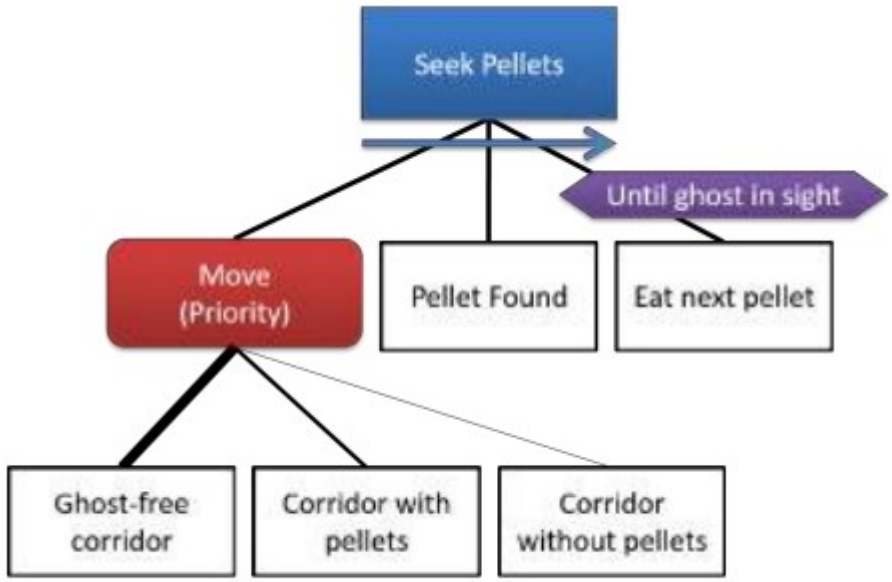


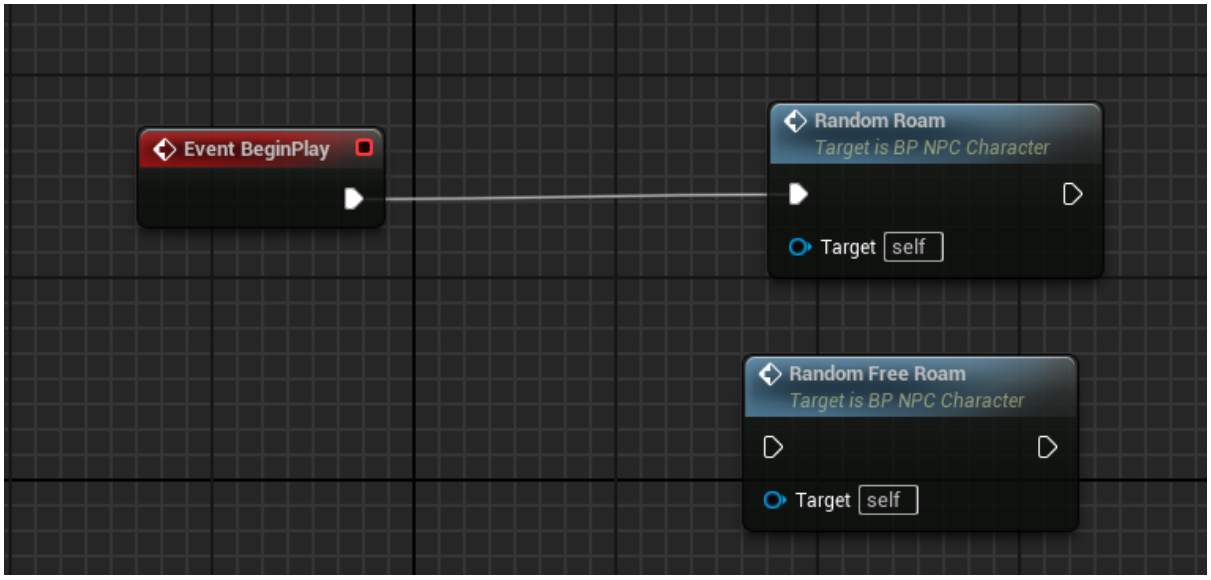
Fig. 2.3 A BT example for the *seek pellets* behavior of Ms Pac-Man.

Slika 37 Primjer stabla odlučivanja za ponašanje Ms Pac-Mana pri traženju kuglica (Yannakakis, 2018).

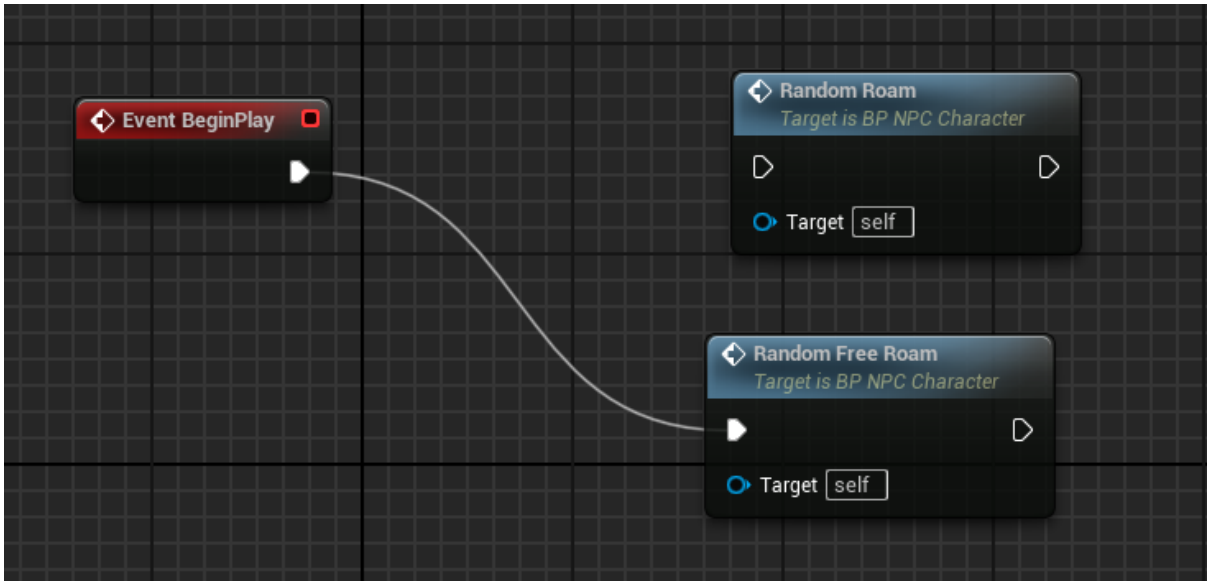
4.4.4. Implementacija Umjetne Inteligencije Druželjubivih NPC-eva

Implementacija umjetne inteligencije druželjubivih NPC-eva zahtjeva izradu nekoliko dijelova kako bi radila. Prvotno imamo dva tipična ponašanja NPC-a koje koristimo, prvo je da NPC-evi imaju zadanu putanju kojom se kreću, gdje idu od točke A do Točke B i tako patroliraju. Drugo ponašanje je da nasumično putuju po cijeloj mapi. Ovisno o potrebi koju NPC treba zadovoljiti može biti korišteno bilo koje ponašanje. Ponašanja su implementirana kao složena stanja konačnog automata unutar

istoe blueprint datoteke Biranje ponašanje se vrši tako da se pozove funkcija „Event BeginPlay“ koja poziva sve ostale funkcije koje su sa njom povezane, programer može odlučiti koje će dalje ponašanje odnosno složeno stanje koristiti tako da poveže ponašanje 1 ili ponašanje 2 sa početnom „Event BeginPlay“ funkcijom. Prikaz obaju ponašanja se može vidjeti na slikama Slika 38 i Slika 39. Oba ponašanja koriste metodu konačnih automata, gdje AI stalno promatra u kojem je stanju NPC te kakvi su ulazi iz okoliša koji utječu na njega, primjerice da li je igrač stupio u interakciju sa njime. Ovisno o ulazima se stanja NPC-eva ili neprijatelja primjereno mijenjaju.

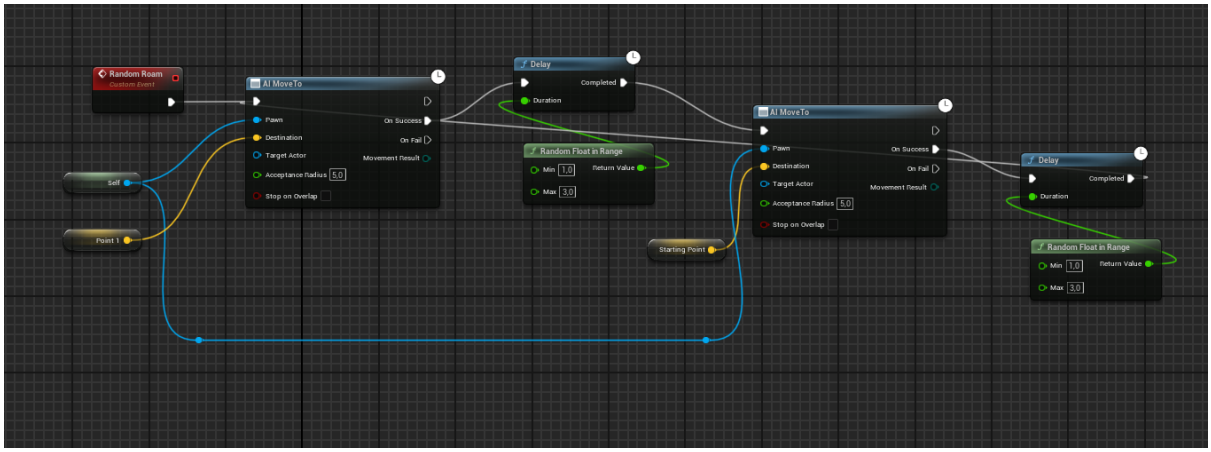


Slika 38 Umjetna inteligencija - ponašanje 1

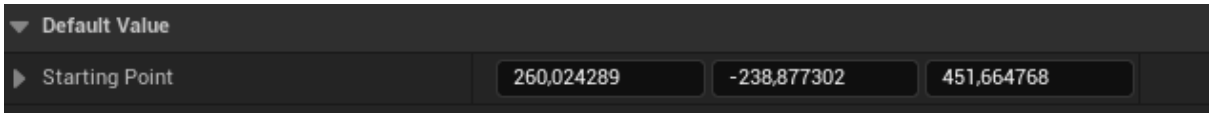


Slika 39 Umjetna inteligencija - ponašanje 2

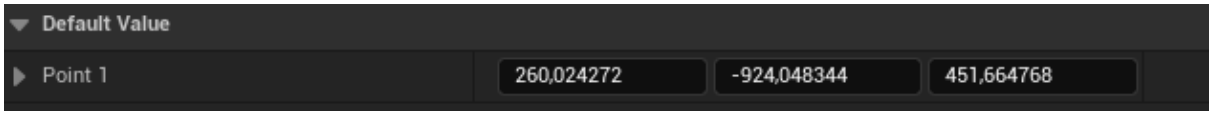
Logika iza ponašanja jedan je jednostavna, NPC ima dvije koordinate između kojih se miče. Kada dođe do točke jedan on se zadrži na njoj nasumično između jedne do tri sekunde zatim ide do sljedeće točke ili se vraća do prve ako su mu zadane samo dvije točke, primjer izgleda koordinati dviju takvih točki se nalazi na Slika 41 i Slika 42, dok prikaz blueprint klase koja to omogućuje se nalazi na Slika 40.



Slika 40 Blueprint klasa umjetna inteligencija - ponašanje 1

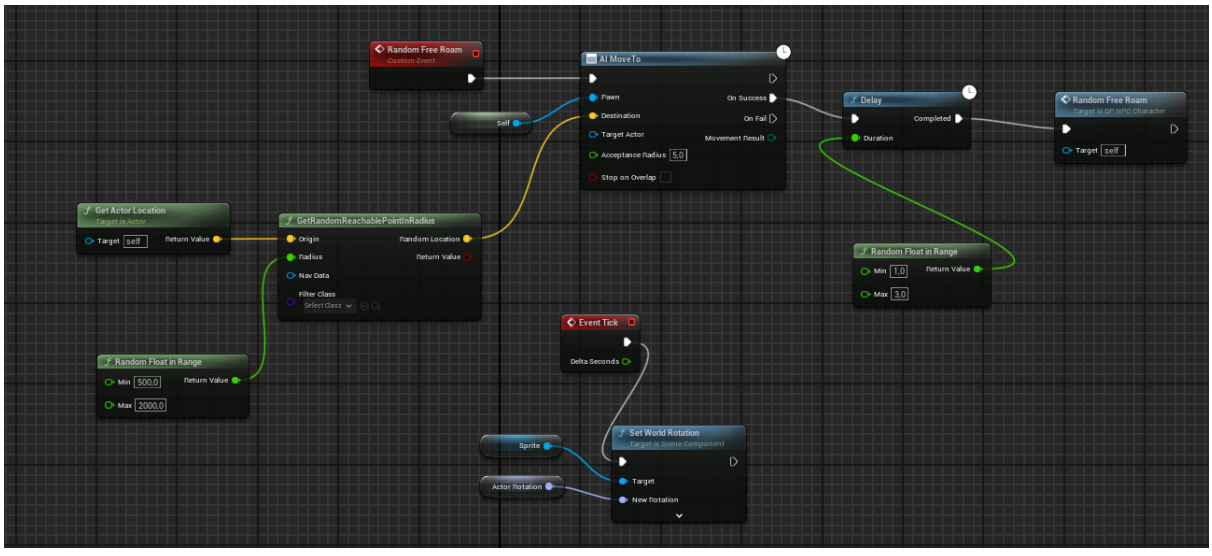


Slika 41 Primjer koordinata početne točke



Slika 42 Primjer koordinata konačne točke

Implementacija drugog ponašanjaje također jednostavna. NPC ima krug od određenog broja metara unutar kojega se nasumično kreće, primjerice ima krug od maksimalno 100m unutar kojega se može kretati, ajmo reći da se nasumično odabere broj 15, to znači da će se NPC kretati 15 metara u nasumično odabrano smjeru. Kada prijeđe tih 15 metara od opet čeka između jedne do tri sekunde te ponavlja proces. Brzina hodanja te radijus kretanja se može mijenjati u oba ponašanja proizvoljno kako to programer želi. Implementacija ovog ponašanja se može vidjeti na Slika 43.



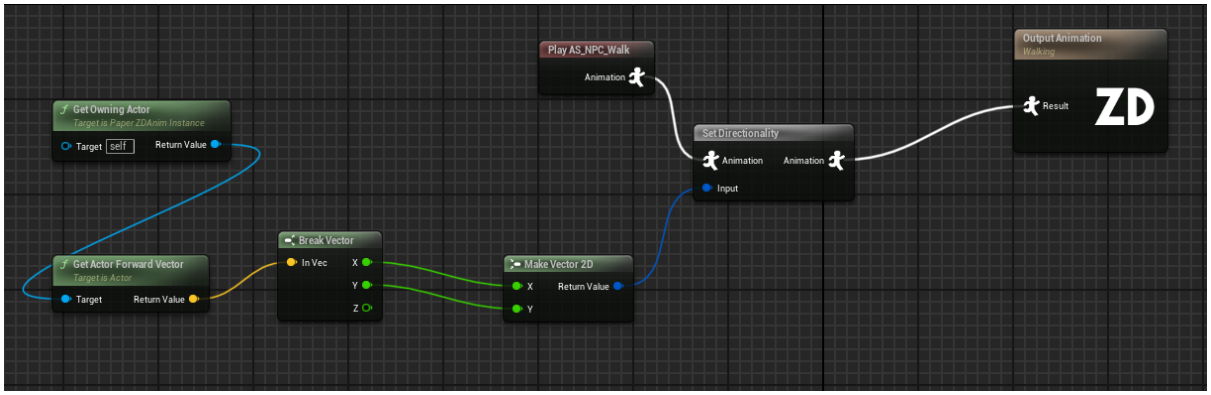
Slika 43 Blueprint klasa umjetna inteligencija - ponašanje 2

Uz logiku potrebno je također podesiti i animacije. Nakon što se animacije uvedu u projekt one se moraju raspakirati kako bi Unreal Engine znao u koliko se sličica po sekundi one kreću. Zatim se treba napraviti animation source unutar kojega se definira koja se animacija prikazuje u određenim smjerovima kada se NPC kreće, kao što se može vidjeti na primjerima sa Slika 44 i Slika 45.

Slika 44 Postavljanje smjerova animacija (1/2)

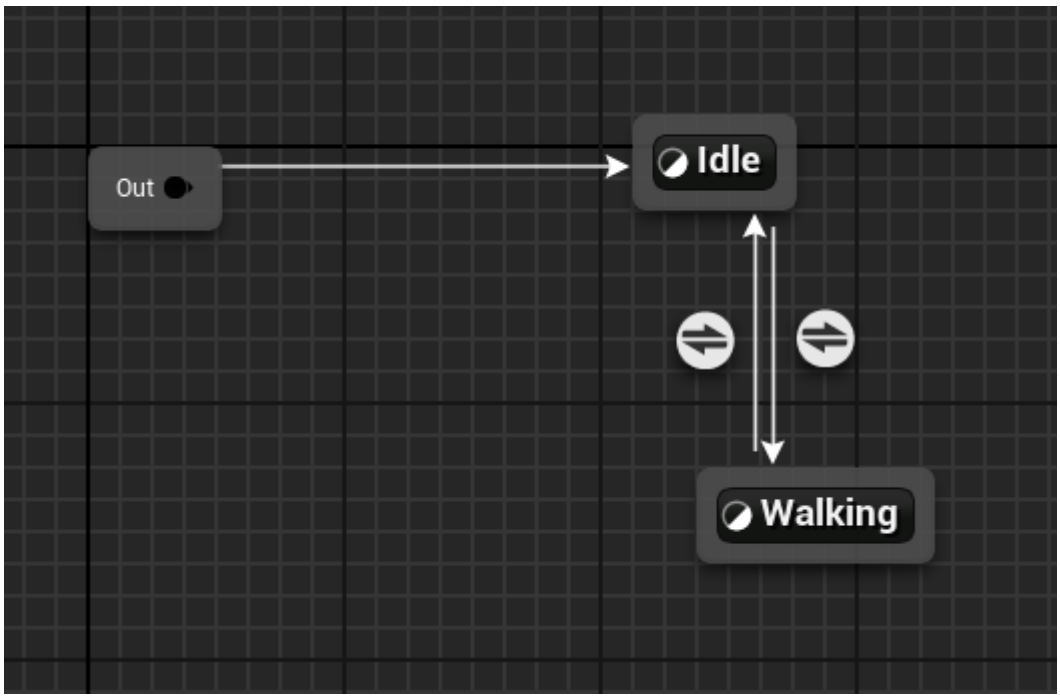
Slika 45 Postavljanje smjerova animacija (2/2)

Naposlijetku, potrebno je definirati prijelaze između animacija, to se radi unutar animation blueprintova. Unutar ovog projekta mi smo koristili PaperZD plugin za rađenje ovih animacija. PaperZD je plugin za Unreal Engine koji je osmišljen kako bi unaprijedio rad s 2D animacijama i sadržajem unutar Unreal Engine okruženja. Primjer jednog takvog PaperZD čvora se nalazi na Slika 46. Cilj mu je pojednostaviti proces kreiranja i upravljanja 2D animacijama, pružajući korisnicima dodatne alate i funkcionalnosti koje nisu standardno dostupne u osnovnom Unreal Engine-u.

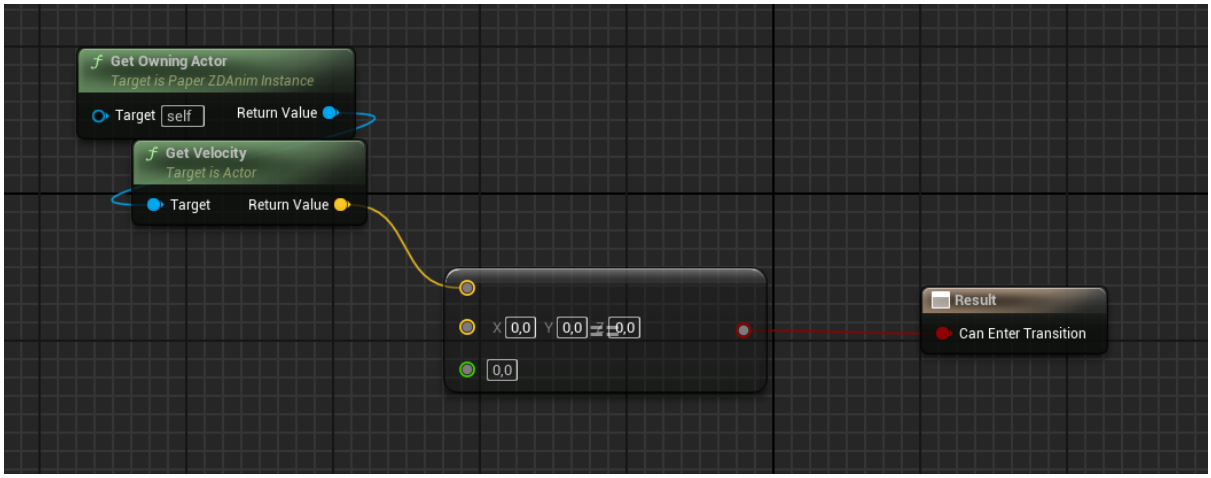


Slika 46 Prikaz PaperZD animacijskih čvorova

Nakon što se definira animation blueprint, unutar kojega se definira logika kada se animacije događaju i pokreću, treba napraviti stablo animacija, vidljivo na Slika 47, koje će kontrolirati prijelaze između svih animacija. Na strelicama koje pokazuju prijelaz iz jednog stanja u drugi se može dodati posebna logika koja provjerava, primjerice da li je NPC prestao kretati, ako da to znači da treba pokrenuti samo animaciju stajanja te ako se pokrene treba prikazati animaciju hodanja. Dublji pogled u izgled jednog takvog prijelaza se može vidjeti na Slika 48.



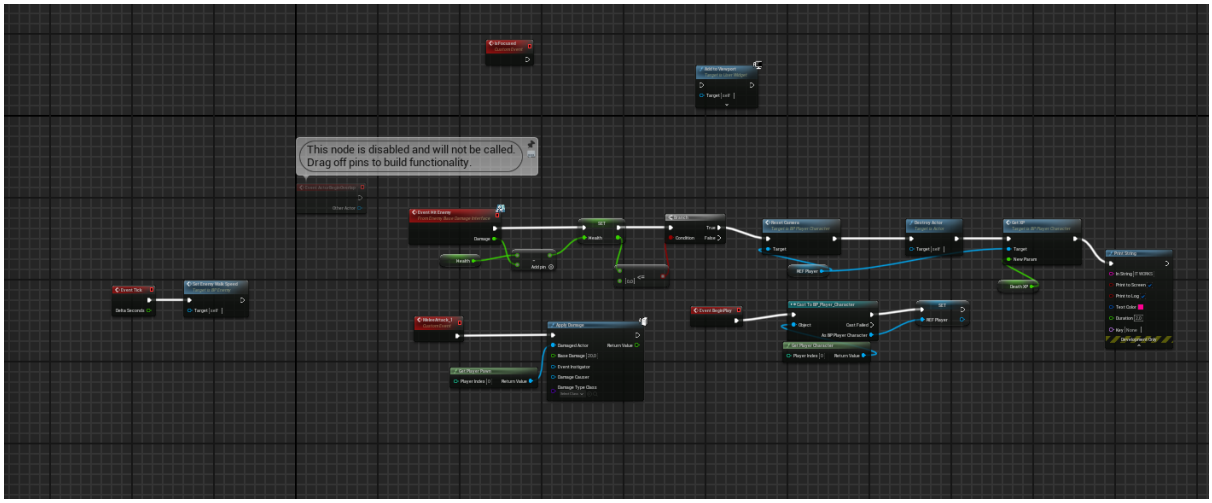
Slika 47 Stablo animacija



Slika 48 Uvjet prijelaza iz animacije hodanja u animaciju stajanja

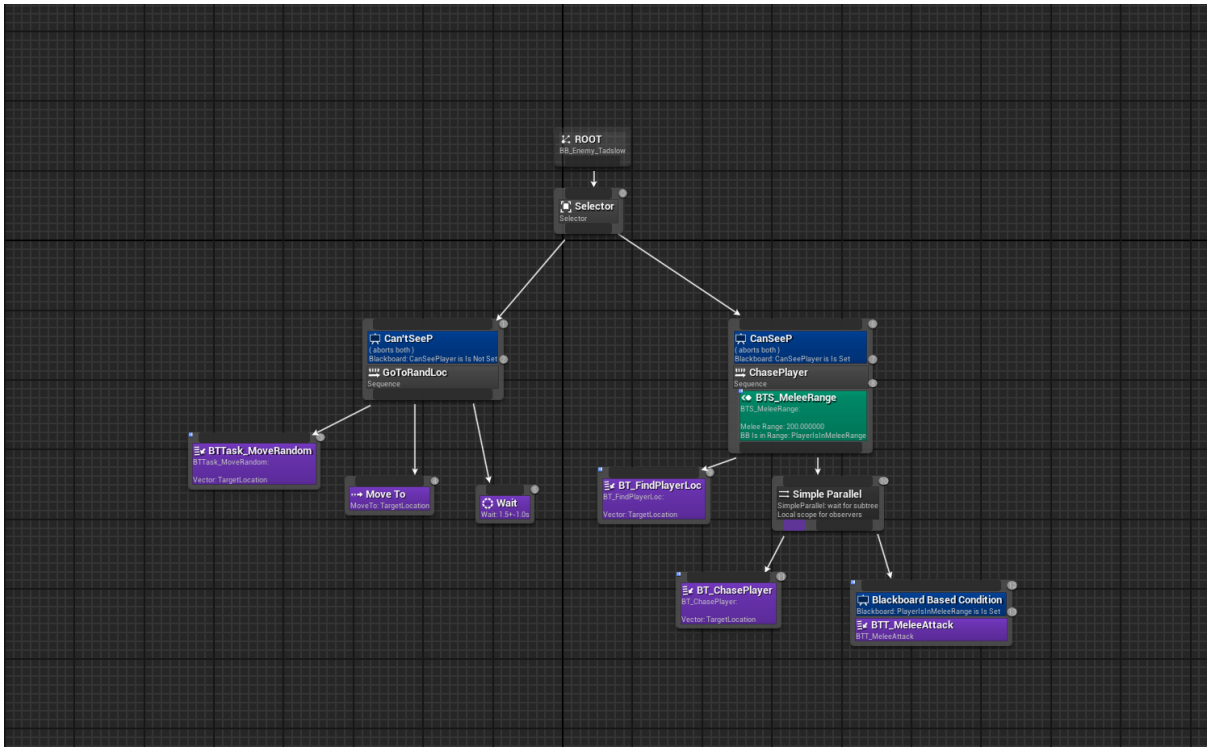
4.4.5. Implementacija umjetne inteligencije neprijatelja

Implementacija umjetne inteligencije neprijatelja je slična kao i kod druželjubivih NPC-eva. Što se tiče animacije postupak je identičan tako da ćemo preskočiti njegovo ponovno objašnjenje ovdje. Razlika se događa kod logike. Kako će neprijatelji raditi puno više stvari od druželjubivih NPC-eva to zahtjeva detaljniju i robustniju logiku. Neprijatelji prvo imaju općeniti blueprint koji provjerava osnovne podatke o neprijatelju, primjerice da li je živ ili ne itd. Opseg tako kompliciranijeg sustava se može vidjeti na Slika 49, također se mora uzeti u obzir da svaki od čvorova koji provjerava neko stanje ima dublju logiku unutar sebe.



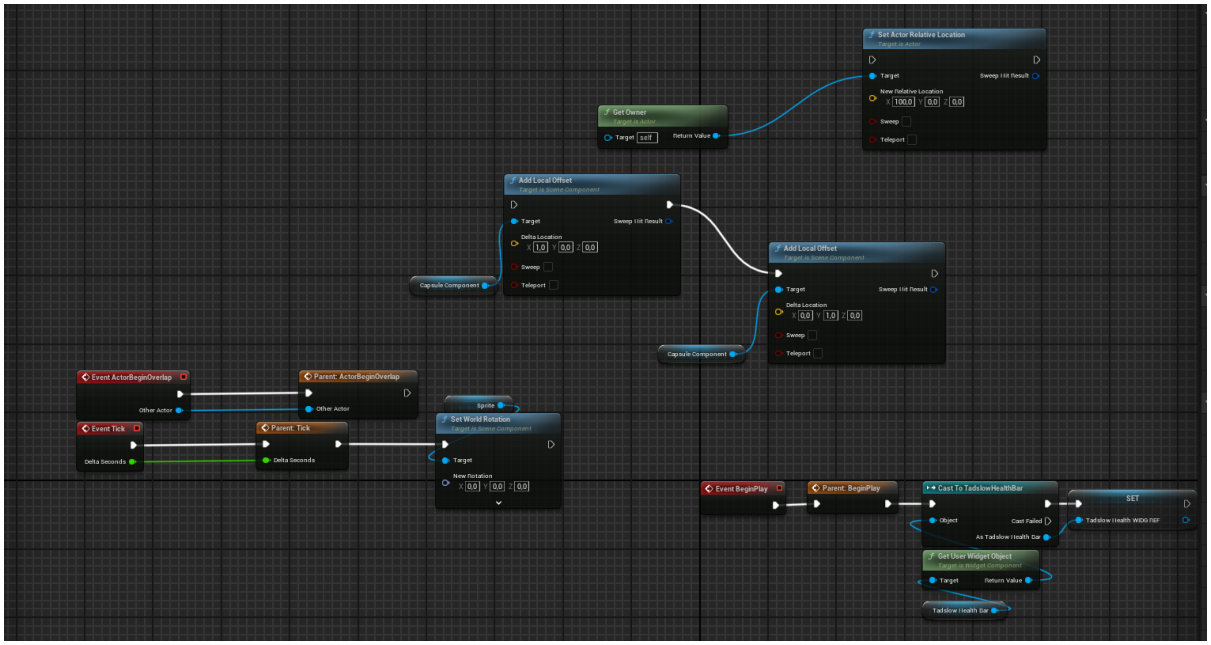
Slika 49 Opći blueprint neprijatelja

Dalje je potrebno napraviti posebnu blueprint datoteku unutar koje se nalazi stablo ponašanja. U suštini, to je logika koja ima strukturu stabla te govori što da neprijatelj radi ovisno u kojoj se situaciji nalazi, npr. da li samo patrolira ili možda napada igrača ili ga pokušava uhvatiti. To su prve dvije veće grane na stablu ponašanja vidljivom na Slika 50.



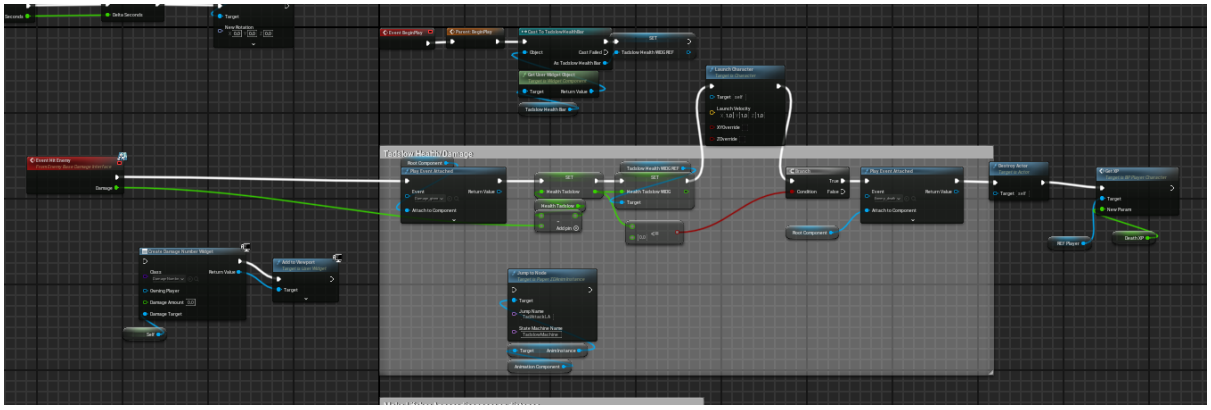
Slika 50 Stablo ponašanja neprijatelja

Zatim, ovisno što stablo ponašanja odluči, pozivaju se različite funkcije koje vrše prikladne akcije, npr. ako neprijatelj vidi igrača početi će ga hvatati te ako ga uhvati početi će ga napadati. Te dvije varijante grane su vidljive na Slika 50. Na istoj slici je na lijevoj grani vidljivo da dokle god da neprijatelj ne vidi igrača on će se kretati uokolo dok ga ne vidi. Na drugoj grani nakon što ga vidi i uhvati se pozovu još i dodatna stanja. Jedno od njih je da se pojavi traka zdravlja iznad glave neprijatelja, koja igraču govori koliko još mora oštetiti neprijatelja da ga porazi, ovo je stanje aktivno dokle god da neprijatelj nije poražen ili ako se igrač ne odvoji od neprijatelja i izađe iz borbe te se traka onda ponovno pojavi sa istom vrijednosti kao i kada je izašao iz borbe igrač, ako opet stupi u borbu sa istim neprijateljem. Drugo stanje je jednostavno stanje napadanja, koje se vrši sve dok je igrač u dometu neprijatelja i dok sam igrač nije poražen. Ove sve funkcije su također povezane sa animacijama gdje se prikladne animacije učitavaju ovisno koju akciju neprijatelj vrši. Nakon provjere se ponovno vraćamo u opći blueprint sa Slika 49 gdje ovisno što je odlučeno će se pozvati funkcija za to zadužena. Primjerice, nakon pozivanja da se stvori neprijatelj poziva se funkcija koja svake sekunde provjerava orijentaciju svijeta ovisno o poziciji kamere te prikladno okreće 2D sprite neprijatelja da je uvijek korektno okrenut prema kameri. Ovo je bitno kako su svi likovi u igri 2D sprite-ovi u 3D svijetu. Prikaz te funkcije se nalazi na Slika 51. Ova funkcija također služi kao baza za čitanje vlastite pozicije neprijatelja koja je bitna za usporedbu sa lokacijom igrača.



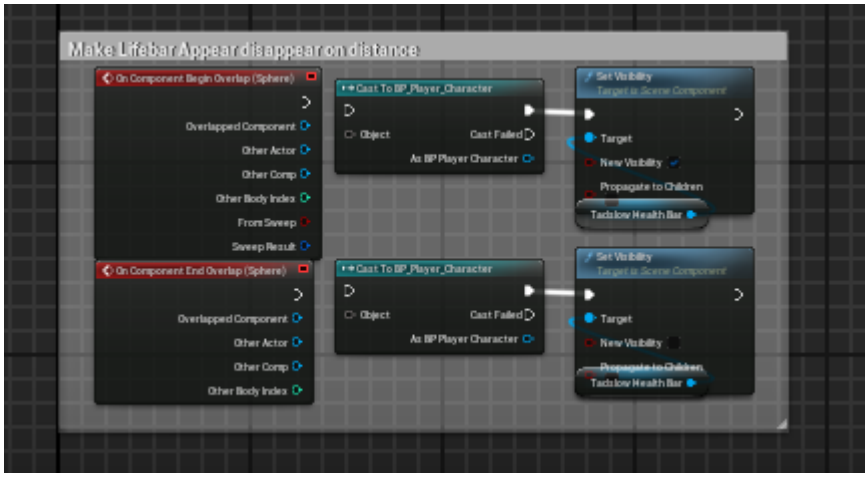
Slika 51 Blueprint pronalaska orijentacije i lokacije neprijatelja

Ako je igrač uočen te ako je u dometu, biti će pokrenut event koji provjerava zdravlje neprijatelja. Prvo se provjerava da li je neprijatelj napadnut, ako je ovisno o statistikama igrača se izvrši računica koja smanji zdravlje neprijatelja za prikladnu količinu te se njegovo novo zdravlje postavi na mjesto prijašnjeg zdravlja. Ovaj se proces iterira sve dok neprijatelj ne bude poražen te se zatim poziva funkcija koja uništava neprijatelja kako se više ne bi prikazivao na ekranu. Naposlijetku se pozove funkcija koja podari igrača sa određenim brojem bodova iskustva koji služe da se igrač ojača. Prikaz ove logike je vidljiv na Slika 52 te je i također dio koji se iterira sa primanjem štete označen sivom bojom.



Slika 52 Blueprint primanja štete i uništenja neprijatelja

Tijekom borbe je prikazana traka zdravlja iznad glave neprijatelja. Ova se logika izvršava cijelo vrijeme dok traje borba te je njena implementacija vidljiva na Slika 53. Ona se ponavlja i poziva svake sekunde tijekom koje je neprijatelj u borbi te prestaje sa radom ili kada igrač izađe iz borbe sa neprijateljem ili ako neprijatelj biva poražen. Također na Slika 54 je prikazan primjer neprijatelja kako hvataju igrača, dok je na Slika 55 prikazan primjer neprijatelja kako napada i stupa u borbu sa igračem.



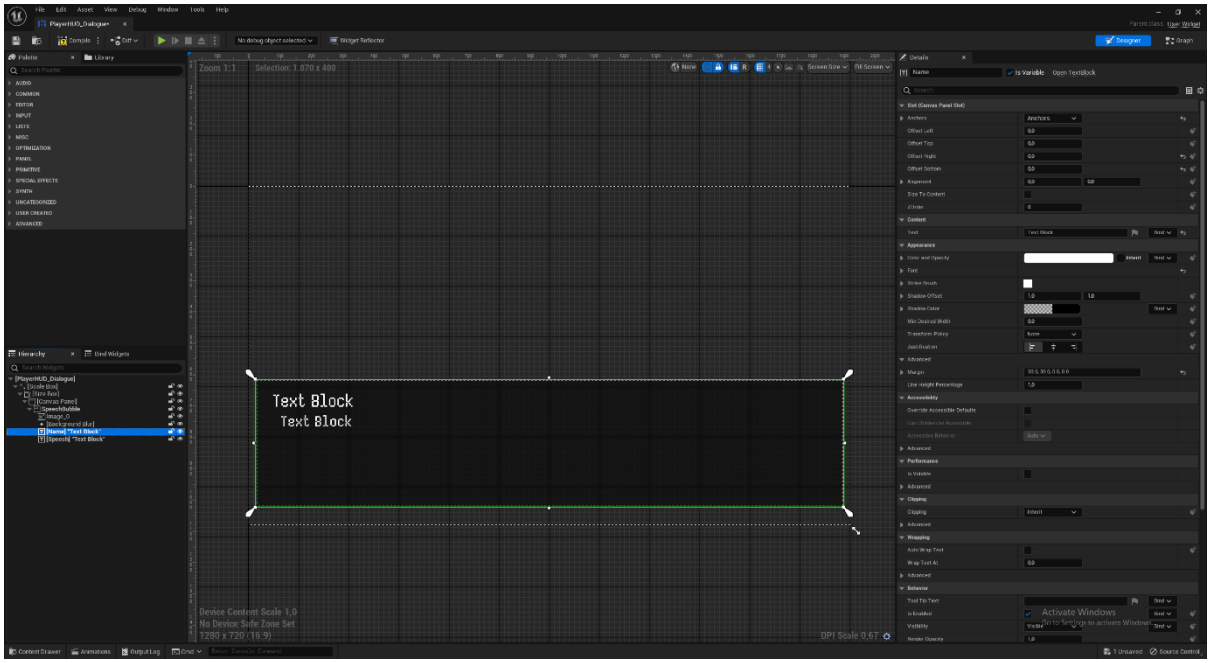
Slika 53 Blueprint ispisa trake zdravlja iznad neprijatelja

Slika 54 AI neprijatelja kako hvata igrača

Slika 55 AI neprijatelja kako napada igrača

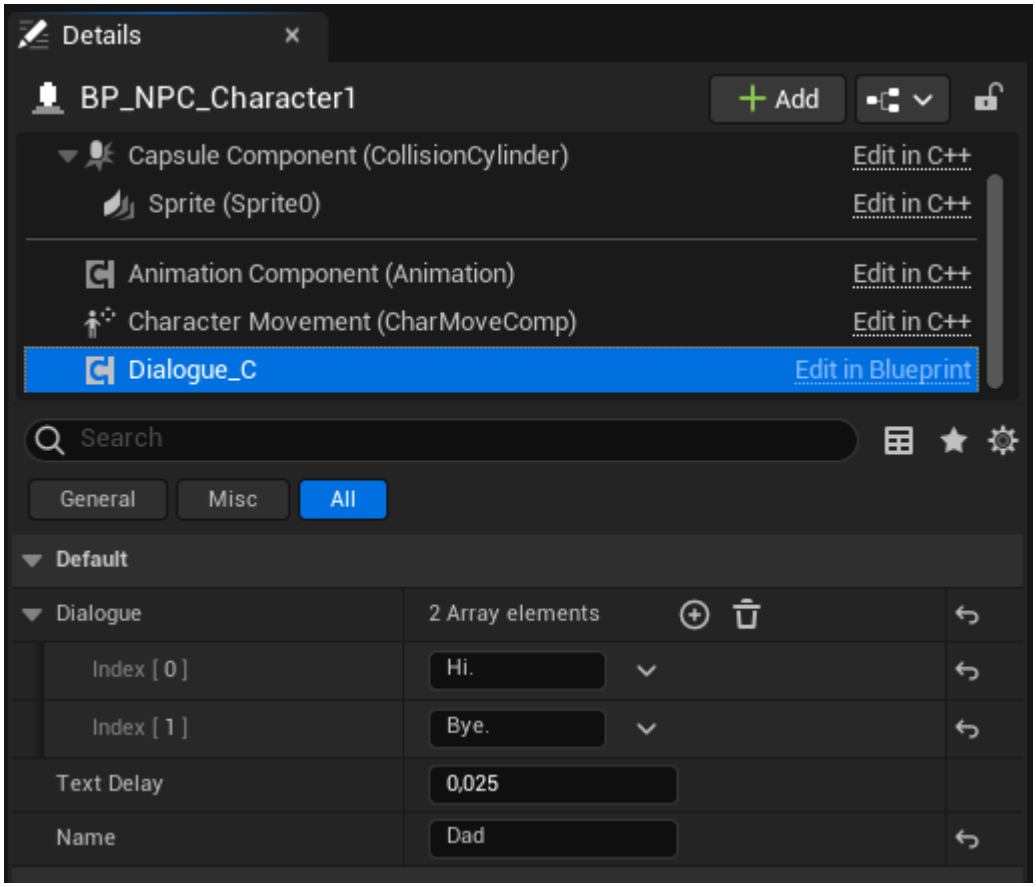
4.4.6. Implementacija dijaloga

Naposlijetku bilo je također potrebno implementirati sustav dijaloga na sve NPC-eve. To se ostvarilo tako što se prvo napravio widget. Unutar Unreal Engine-a, widget je osnovna komponenta za kreiranje korisničkog sučelja (UI). Widgeti omogućuju dizajnerima i programerima da kreiraju interaktivne elemente sučelja, kao što su gumbi, tekstualna polja, klizači, slike, i drugi UI elementi, koje igrači mogu vidjeti i s kojima mogu interagirati tijekom igre. Prikaz widgeta korištenog za okvir dijaloga je vidljiv na Slika 56.



Slika 56 Widget okvira dijaloga

Nakon što se napravio widget bilo je potrebno u već postojeću logiku igrača dodati dio da kad igrač pritisne određenu tipku da će se provjeriti da li objekt sa kojim igrač pokušava imati interakciju ima na sebi komponentu dijaloga, ako ima, dijalog će se pokrenuti, ako ne, ništa se neće dogoditi. Unutar Unreal Engine-a, komponenta je osnovni modularni element koji se može dodati objektima kako bi im se proširile funkcionalnosti. Komponente omogućuju programerima i dizajnerima da razdvoje različite aspekte ponašanja i osobina glumaca u manje, specijalizirane jedinice koje se mogu ponovno koristiti i kombinirati na različite načine. Prikaz komponente dijaloga postavljene na jednog NPC-a je vidljiva na Slika 57. Dobra stvar kod komponenti što se jako lagano može staviti na bilo koji objekt i napisati dijalog koji će reći, on funkcionira i na druželjubivim NPC-evima (primjer jednog takvog dijaloga na Slika 58), neprijateljima pa čak i običnim objektima poput bačve (prikaz takvog primejra na Slika 59). Programer sa lakoćom može napisati što će sve NPC ili objekt reći ćim doda komponentu na njega.



Slika 57 Komponenta dijaloga

Slika 58 Primjer dijaloga sa NPC-em

Slika 59 Primjer dijaloga sa objektom

5. Zaključak

Razvoj videoigara je težak posao te kada se videoigra napravi ništa ne garantira njezin uspjeh. Unatoč tome, sam proces je veoma kreativno ispunjujuć. Razvoj videoigara zahtjeva znanje iz raznih područja, ne nužno samo informatičkih. Primjerice, znanje iz područja arhitekture je poželjno ako se žele stvoriti realistične kuće, zgrade i slično. Ova znanja iz drugih zanimanja nisu potrebna, ali uveliko pomaže ako postoje. To nas dovodi do glavnog zaključka, a to je da za razvoj videoigara ne treba neko preveliko predznanje, već samo volja i spremnost na učenje. Danas ima puno samostalnih game developera koji vlastoručno naprave i izbace cijele igre i vrlo su uspješni u tome. Razvoj videoigara omogućuje ljudima da se izraze kako oni žele te put koji uzmu da dođu do konačnog cilja ne mora biti isti kao i kod svih ostalih. Nekima je dovoljno da budu samo dio tima i da rade na samo jednom aspektu videoigara, dok drugi žele što više biti angažirani u cjelokupan proces. Bitno je znati vlastite granice i osobne interese kako bi se te granice prešle preko učenja i osobnog razvoja kako bi se stvorile najbolje videoigre moguće.

6. Popis Slika

7. Literatura

Zackariasson, P. and Wilson, T.L. eds. (2012). The Video Game Industry: Formation, Present State, and Future. New York: Routledge.

The Gamer (2024). 'ESA: Gaming niche popular die mad gamers', [Mrežni pristup]. Dostupno: https://www.thegamer.com/esa-gaming-niche-popular-die-mad-gamers/ [Zadnji pokušaj pristupa 4.7.2024.].

PR Newswire (2024). 'Video games remain lifelong source of entertainment for 190.6 million Americans', [Mrežni pristup]. Dostupno: https://www.prnewswire.com/news-releases/video-games-remain-lifelong-source-of-entertainment-for-190-6-million-americans-302146098.html [Zadnji pokušaj pristupa 4.7.2024.].EGDF (2024). 'Summary of EU video games industry funding calls for 2024', [Mrežni pristup]. Dostupno: https://www.egdf.eu/summary-of-eu-video-games-industry-funding-calls-for-2024/ [Pokušaj pristupa 4.7.2024.].

Newzoo (2024). 'The latest games market size estimates and forecasts', [Mrežni pristup]. Dostupno: https://newzoo.com/resources/blog/the-latest-games-market-size-estimates-and-forecasts [Zadnji pokušaj pristupa 4.7.2024.].

Steinberg, S. (2007). The definitive Guide: Videogame Marketing and PR (1st ed.). iUniverse. ISBN 978-0-59543-371-1.

CB Insights (2024). 'Game engines growth expert intelligence', [Mrežni pristup]. Dostupno: https://www.cbinsights.com/research/game-engines-growth-expert-intelligence/ [Zadnji pokušaj pristupa 4.7.2024.].

Unreal Engine (2024). 'Unreal Engine End User License Agreement', [Mrežni pristup]. Dostupno: https://www.unrealengine.com/en-US/eula/unreal [Zadnji pokušaj pristupa 4.7.2024.].

Guinness World Records (2024). 'Most successful game engine', [Mrežni pristup]. Dostupno: https://www.guinnessworldrecords.com/world-records/most-successful-game-engine [Zadnji pokušaj pristupa 4.7.2024.].

Blender (2024). 'Corrective releases for version 4.1', [Mrežni pristup]. Dostupno: https://developer.blender.org/docs/release_notes/4.1/corrective_releases/ [Zadnji pokušaj pristupa 4.7.2024.].

GitHub (2024). 'Git commit e83c5163316f89bfbde7d9ab23ca2e25604af290', [Mrežni pristup]. Dostupno: https://github.com/git/git/commit/e83c5163316f89bfbde7d9ab23ca2e25604af290 [Zadnji pokušaj pristupa 4.7.2024.].

DevProJournal (2024). 'GitLab acquires UnReview to expand its open DevOps platform with machine learning capabilities', [Mrežni pristup]. Dostupno: https://www.devprojournal.com/news/gitlab-acquires-unreview-to-expand-its-open-devops-platform-with-machine-learning-capabilities-2/ [Zadnji pokušaj pristupa 4.7.2024.].

Concept Art Empire (2024). 'What is 3D modeling?', [Mrežni pristup]. Dostupno: https://conceptartempire.com/what-is-3d-modeling/ [Zadnji pokušaj pristupa 4.7.2024.].

Siemens (2024). ‘3D modeling technology’, [Mrežni pristup]. Dostupno: https://www.sw.siemens.com/en-US/technology/3d-modeling/ [Zadnji pokušaj pristupa 4.7.2024.].

Takeoff Pros (2024). ‘Guide to 3D modeling’, [Mrežni pristup]. Dostupno: https://www.takeoffpros.com/2020/04/27/guide-to-3d-modeling/ [Zadnji pokušaj pristupa 4.7.2024.].

Science Daily (2024). ‘3D computer graphics’, [Mrežni pristup]. Dostupno: https://www.sciencedaily.com/terms/3d_computer_graphics.htm [Zadnji pokušaj pristupa 4.7.2024.].

Oxland, K. (2004). Gameplay and design. Addison Wesley. ISBN 0-321-20467-0.

Google Books (2024). ‘Gameplay and design by Kevin Oxland’, [Mrežni pristup]. Dostupno: https://books.google.hr/books?id=kRMeBQAAQBAJ&redir_esc=y [Zadnji pokušaj pristupa 4.7.2024.].

Internet Archive (2024). ‘Next Generation Issue 015’, [Mrežni pristup]. Dostupno: https://archive.org/details/nextgen-issue-015/page/n39/mode/2up [Zadnji pokušaj pristupa 4.7.2024.].

Bartle, R. (2003). Designing Virtual Worlds. New Riders. ISBN 0-13-101816-7.

Wikipedia (2024). ‘Designing Virtual Worlds by Richard Bartle’, [Mrežni pristup]. Dostupno: https://en.wikipedia.org/wiki/Designing_Virtual_Worlds [Zadnji pokušaj pristupa 4.7.2024.].

O'Reilly (2024). ‘AI for game development’, [Mrežni pristup]. Dostupno: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch01.html [Zadnji pokušaj pristupa 4.7.2024.].

Medium (2024). ‘Implementation of artificial intelligence in gaming’, [Mrežni pristup]. Dostupno: https://medium.com/@tagx20/implementation-of-artificial-intelligence-in-gaming-c6a73ac3b415 [Zadnji pokušaj pristupa 4.7.2024.].

Built In (2024). ‘AI in games’, [Mrežni pristup]. Dostupno: https://builtin.com/artificial-intelligence/ai-games [Zadnji pokušaj pristupa 4.7.2024.].

Markovate (2024). ‘AI in gaming’, [Mrežni pristup]. Dostupno: https://markovate.com/blog/ai-in-gaming/ [Zadnji pokušaj pristupa 4.7.2024.].

Columbia Engineering (2024). ‘AI applications in video games’, [Mrežni pristup]. Dostupno: https://ai.engineering.columbia.edu/ai-applications/ai-video-games/ [Pokušaj pristupa 4.7.2024.].

Saagie (2024). ‘Artificial intelligence in video games’, [Mrežni pristup]. Dostupno: https://www.saagie.com/en/blog/artificial-intelligence-in-video-games/ [Zadnji pokušaj pristupa 4.7.2024.].

Brilliant (2024). ‘Finite state machines’, [Mrežni pristup]. Dostupno: https://brilliant.org/wiki/finite-state-machines/ [Zadnji pokušaj pristupa 4.7.2024.].

Schell, J. (2014). *The Art of Game Design: A Book of Lenses* (2nd ed.)

Yannakakis, G.N. i Togelius, J. (2018). *Artificial Intelligence and Games*. ISBN 978-1-4665-9867-6

Millington, I. i Funge J. (2009). Artificial Intelligence for Games (2nd ed.). ISBN 978-0-12-374731-0

Built In (2024). 'AI in Games', [Mrežni pristup]. Dostupno: https://builtin.com/artificial-intelligence/ai-games [Pokušaj pristupa 4.7.2024.].

Reintech (2024). 'Create Game AI with Python: Tutorial', [Mrežni pristup]. Dostupno: https://reintech.io/blog/create-game-ai-python-tutorial [Pokušaj pristupa 4.7.2024.].

Harvard University (2024). 'AI in Video Games: Toward Intelligent Game Design', [Mrežni pristup]. Dostupno: https://sitn.hms.harvard.edu/flash/2017/ai-video-games-toward-intelligent-game/ [Pokušaj pristupa 4.7.2024.].

BlendAnimate (2024). 'Introduction to UV Mapping in Blender for Complete Beginners', [Mrežni pristup]. Dostupno: https://blendanimate.com/introduction-to-uv-mapping-in-blender-for-complete-beginners/ [Pokušaj pristupa 4.7.2024.].