Izrada RPG igre u otvorenom svijetu primjenom Unreal razvojnog okruženja

Grubiša, Mattia

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

MATTIA GRUBIŠA

IZRADA RPG IGRE U OTVORENOM SVIJETU PRIMJENOM UNREAL RAZVOJNOG OKRUŽENJA

Završni rad

Pula, rujan, 2023. godine

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

MATTIA GRUBIŠA

IZRADA RPG IGRE U OTVORENOM SVIJETU PRIMJENOM UNREAL RAZVOJNOG OKRUŽENJA

Završni rad

JMBAG: 0303092171, redoviti student
Studijski smjer: Računarstvo

Predmet: Programiranje
Znanstveno područje: Tehničke znanosti
Znanstveno polje: Računarstvo
Znanstvena grana: Programsko inženjerstvo
Mentor: izv. prof. dr. sc. Tihomir Orehovački

Pula, rujan, 2023. godine

IZJAVA

o akademskoj čestitosti

Ja, dolje potpisani Mattia Grubiša, kandidat za prvostupnika računarstva ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

U Puli, rujan, 2023. godine

IZJAVA

o korištenju autorskog djela

Ja, Mattia Grubiša dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom:

„Izrada RPG igre u otvorenom svijetu primjenom Unreal okruženja“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, rujan, 2023. godine

Sadržaj

1. Uvod

Velike igre u današnje doba zahtijevaju sve više prostora na računalu, veću snagu računala te veću rezoluciju i brzinu osvježavanja ekrana kako bi se igrači što više upustili u drugu dimenziju, dimenziju igara. Takve se igre većinom klasificiraju kao AAA Games igre velikog budžeta i zahtjevnosti kojima je potrebno godinama da se razviju. Za razvoj takvih igra koriste se razvojna okruženja kao „temelji“ koji ubrzavaju razvoj takvih igara. Okruženja za razvoj računalnih igara najviše koriste C++, Java i C# programske jezike, mnoge kompanije imaju razvijen svoje okruženje neke od poznatih kompanija koje proizvode takve igre su CD Projekt Red, Ubisoft, Bethesda, Rockstar Games i mnoge druge.

Glavi smisao ovog završnog rada je upoznavanje Unreal okruženja tijekom izrade RPG (eng. Role – Role Playing Game) igre. Upoznavanje elemenata koji se koriste pri razvoju video igara kao što je stvaranje svijeta, animacije, likova, objekata, programiranje, materijali, efekti i mnogi drugi. RPG igra se često povezuje sa time da se igrač uživi u lika s kojim on igra te istražuje i upoznaje novi otvoreni svijet. Neke od značajki s kojim se povezuje RPG otvorenog svijeta su priča, istraživanje, upuštanje, napredovanje te nakraju igre veliki šef kojem se protagonist suprotstavlja. Neke od potrebnih stvari su kreativnost, vještina i razumijevanje ne samo programa već i igrača koji će igrati igru što oni žele.

Odd Chase igra napravljena je u Unreal Engine programu, korištena verzija programa je 5.2. Glavni koncepti ove igre bi bili istraživanje ručno sastavljenog fantazi otvorenog svijet, dizajnirani neprijatelji i razgovor sa likovima za danji razvoj osnovne mehanike koje se mogu pronaći u skoro svakoj RPG igri kao što je borbeni sustav i sustav štete. Elementi korišteni u igri mogu se pronaći kao besplatna sredstva u Quixel Bridge ili online. Zbog veličine projekta koja je 50 gb, nije moguće postaviti link na github, već se pregled video igre i njezinih elemenata unutar Unreal Engine može pronaći na Youtube – u link - u: https://youtu.be/1WQICN6HPR4

2. Analiza igara koje su služile kao inspiracija

Inspiracija za izradu su služile mnoge igre avanturističkog tipa kao što su „Witcher 3 Wild Hunt“, „The Eldor Scroll 5: Skyrim“, „Elden Ring“ … Što čini te igre posebnima je veliki i bogati otvoreni svijet u kojem se može svatko zabaviti i izgubiti bezbrojno sati kako bi istražili svaki kutak i pronašli sve zanimljivosti koje su programeri postavili. Neki od glavnih mehanizama u takvim igrama su sistem borbe, prostor za spremanje stvari, prostor za napredovanje te priča koja je napravljena u obliku zadataka gdje se dobivaju u razgovoru sa NPC (eng. Non – playable character) ljudima. Glavni zadatak igre je proći glavni priču, uz glavnu priču nalaze se sporedne manje priče koje nisu obavezne ali u kojima se može nešto dobiti te olakšati prolaz igre. Glavni resursi u igri su izdržljivost većinom označeno zelenom bojom, zdravlje crvenom bojom i mana plavom bojom na trakama koje se nalaze na ekranu. Svaki od resursa može, a i ne mora biti povezanom sa nekim tipom igre koje igrač odabere tokom igre kao npr. ratnik više koristi izdržljivost dok čarobnjak manu. Neki od ostalih resursa su bodovi iskustva korišteni za povećanje razine igrača, novci za kupovinu, sirovine za izradu itd. Korisnička sučelja su minimalna te se ne prikazuju dok igrača ne uđe u bitku, na korisničkom sučelju se nalaze glavni resursi uz mapu i npr. strijele ili odabrano oružje/čaroliju. Jedna od najbitnijih stvari je sama atmosfera i dinamika igre. Neke igre polako upuštaju u sam štih igre dok druge brzo ubace u preživljavanje. Svako područje u otvorenom svijetu ima svoju atmosferu koja stvara dodatan ugođaj mnoge od navedenih elemenat prikazani su na slici 1.

Slika 1. Prikaz sučelja RPG igara lijevo „Elden Ring“, desno „The Elder Scroll V: Skyrim“

Odd chase igra uzima elemente iz različitih igra ponekih i koje nisu RPG igre. Glavni resursi se ne vraćaju uobičajenim načinom kao što su napici ili čarolije već se pronaći u svijetu što dodaje neki element taktičnosti i rizika prije sljedećeg susreta, također nije svaka bitka opcionalna neke su potrebne za prolaz dalje.

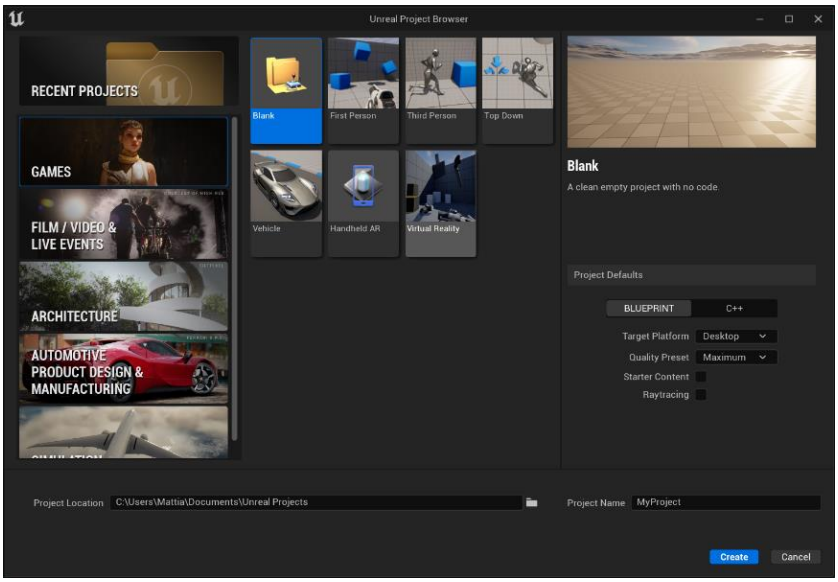
3. Unreal Engine

Unreal Engine je program otvorenog izvora korišten za stvaranje mnogih stvari uz video igre kao što su simulacije, filmovi, automobilskoj industriji, arhitektura i mnoge druge. Napravljen je od strane Epic Games prvi puta prikazan 1998., napisan je u C++ programskom jeziku te pruža podršku za mnoge platforme, računala, mobiteli, konzole, virtualna stvarnost… Najnovija inačica Unreal Engine je 5.2., sa pretpregledom 5.3. Sa brzim prelaskom na Unreal Engine 5 u kojem su pridoneseni mnogi novi elementi od kojih su mnogi još u eksperimentalnoj fazi sam program u kojem je mnogo elemenata nije stabilno. Veliki dio dokumentacije je namijenjen za Unreal Engine 4, nije ažuriran te u nekim situacijama i nedovoljan.

3.1. Unreal Engine Project Browser

Ulaskom u Project Browser nude se šablone za pokretanje projekata prikazuje se na slici 2. Šablone su temelji napravljeni za brži početak izrade. Fokus u ovom projektu je RPG igra te se odabire ta šablona Third Person. Ako se odabere Blank šablona za početak potpuno čistog projekta uvijek se kasnije mogu dodati predefinirane šablone. Također se pružaju opcije Blueprint ili C++ što određuje na koji način će se početku zadanom pisati funkcije. Blueprints je vizualni sustav za programiranje u Unreal Enginu na temelju sučelja čvorova za stvaranje elemenata. Može se odabrati za koju platformu želimo razvijati igru, kvaliteta kakvom će se prikazivati, početne stvari koje se i poslije mogu dobaviti i raytracing[1]. Unaša se lokacija gdje će se projekt nalaziti i ime projekta, za samo ime projekta dobra praksa je unijeti ime bez razmaka i bilo kojih znakove kako poslije ne bi došlo do poteškoća.

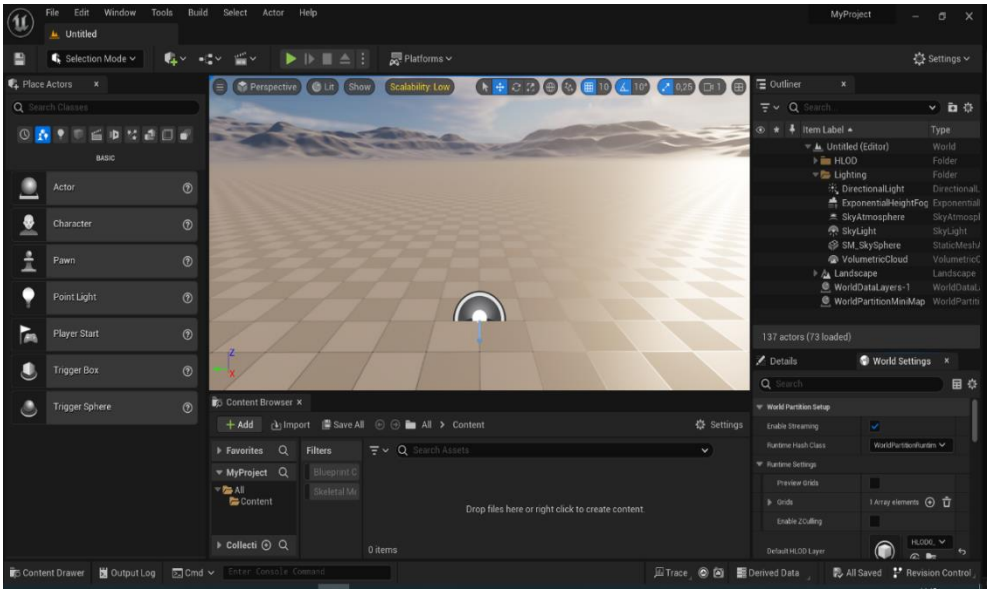
[1] Raytracing – praćene svjetlosnih zraka, njihovo osvjetljenje, ambijent koji stvaraju i refleksiju



Slika 2. Prikaz Unreal Project Browser

3.2. Unreal Editor

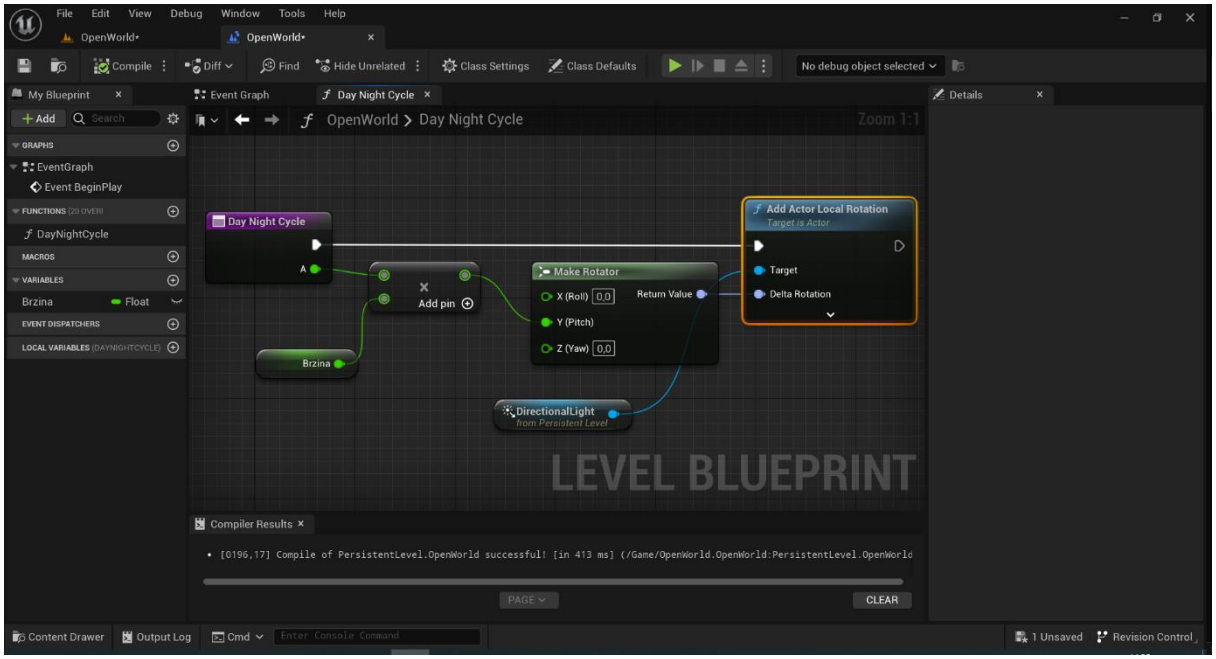
Otvaranje Unreal Editora po prvi put može biti pomalo zastrašujuće sa mnogo prozora svugdje kao što prikazuje slika 3 novo otvorene empty šablone. Sa lijeve strane se nalazi Place Actors ploča koja služi za dodavanje elemenata u prozor za prikaz. Prozor za prikaz je u sredini na kojem se radi kako će igra izgledati. Ispod prozora za prikaz je Content Browser ploča iz koje se mogu dovući elementi koji su naknadno dodati toga ćemo se dotakni kasnije. Sa desne strane Outliner prozor na kojem se prikazuje sve što je dodano u prozoru za prikaz i ispod toga Details prozor gdje se dobivaju detalji odabranog elementa i World Setting za namještaje postavki svijeta.



Slika 3. Prikaz Unreal Engine Editor

3.3. Unreal Engine Blueprints Visual Scripting

Blueprints Visual Scripting je jedan od načina kako se stvari mogu programirati u Unreal okruženju. Vrlo je jednostavan i intuitivan za koristiti. Neki od važnijih stvari za znati su Event Begin Play i Event Tick. Event Begin Play označava što će se dogoditi čim igra započne, Event Tick što će se dogoditi u svakoj frame[2] – u igre. Za dodavanje elemenata u Blueprints dovoljno je desni klik, upisati ime i pronaći funkciju koju želite i dodati. Spajanje je jednostavno samo povući bijele linije iz jedne u drugu iglu za izvršenje. Sa desne strane se nalaze detalji odabrane funkcije ili varijable. Sa lijeve strane sve moguće funkcije, varijable i grafovi. Za korištenje Blueprints Visual Scripting potrebno je ići u event graph u elementu u kojem želimo programirati, prikazano na slici 4.



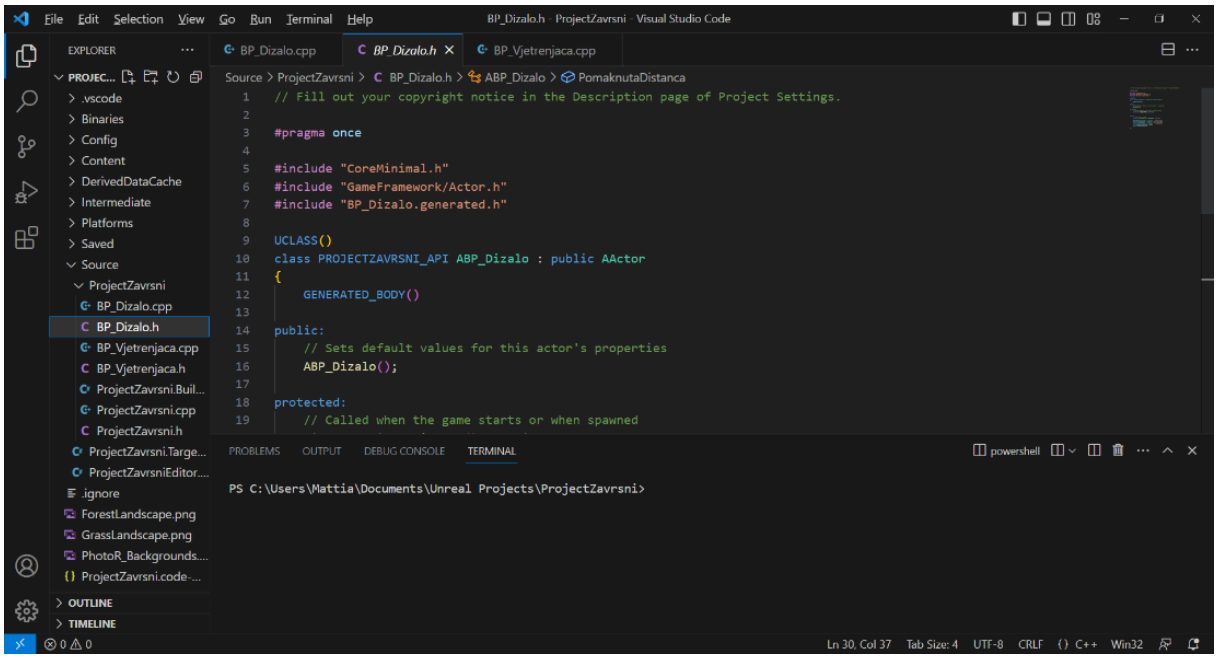
Slika 4. Prikaz Blueprints

3.4. Unreal Engine C++

Za programiranju s C++ programskim jezikom u Unreal Engine potrebno je par stvari. Za Unreal Engine 5 potreban je .NET 7.0 ili viši, Visual Studio koji će služiti za kompiliranje koda i integrirano razvojno okruženje kao što je Visual Studio Code. Na početku je potrebno skinuti i instalirati .NET 7.0 SDK paket koji će instalirati sve sa Microsoft stranice. Sljedeći korak je pronaći podržanu verziju Visual Studio, instalirati i uz to staviti da instalira paket razvijanje igra u C++ i zadnja instalacija je Visual Studio

[2] Frame – sličice koje vide igrači tijekom igre, referenca 30/60 FPS - sličica po sekundi

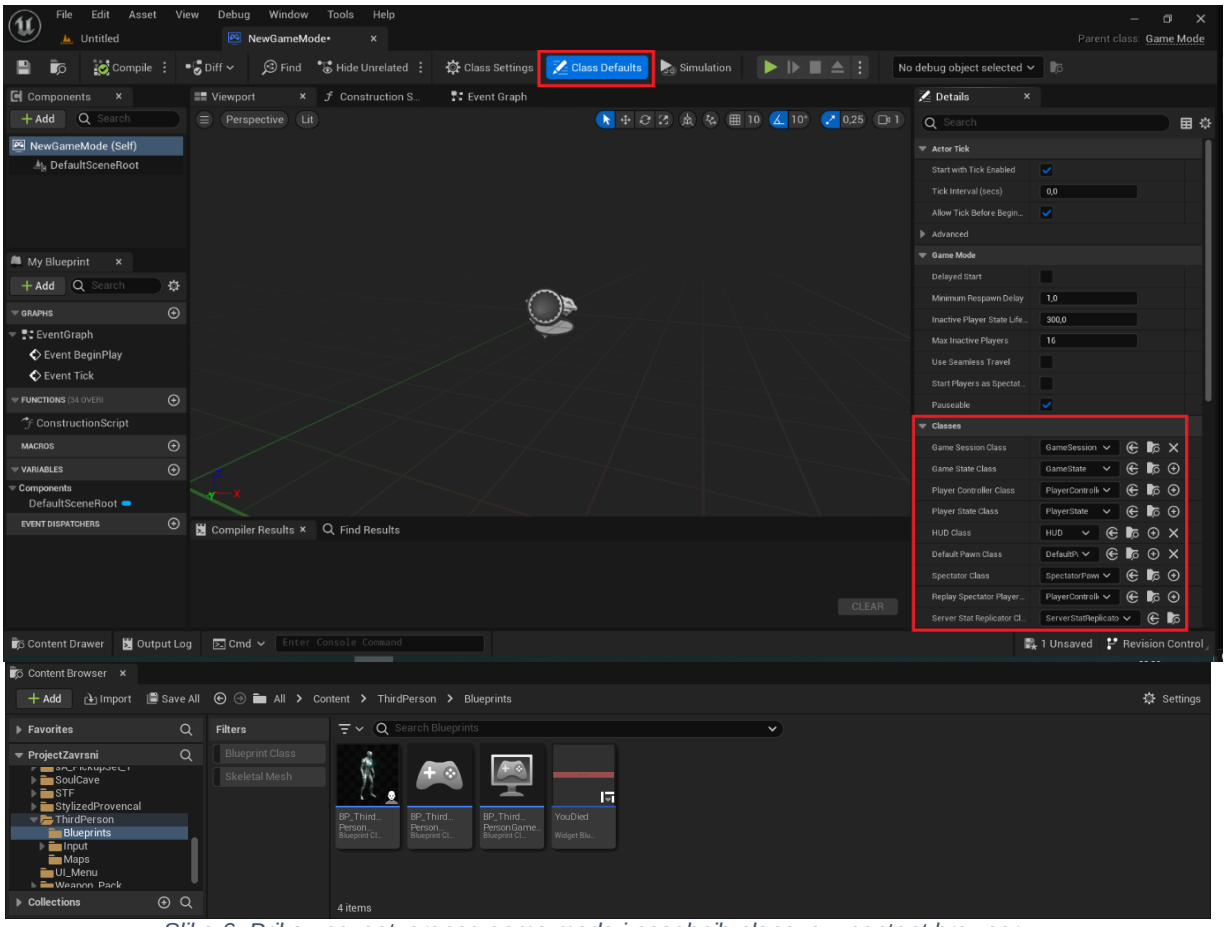
Code sa paketom koji dozvoljava korištenje C/C++, preporuka je instalirati Unreal Engine Snippets za lakše snalaženje u kodu. Nakon toga je potrebno otići u napravljeni projekt, onda u postavke projekta i pod editor označiti Visual Studio. U Unreal Engine Editoru sa desne strane kliknuti Add i dodati C++ klasu. Otvorit će se Visual Studio Code i zadnji korak je kliknuti Terminal na vrhu Run Build Task i onda ime projekta WIN64 Development Build. Sada je sve spremno za programiranje u C++ za Unreal okruženje. Na slici 5 je prikazana nova blueprint class unutar Visual Studio Code.



Slika 5. Prikaz Visual Studio Code spremnog za korištenje u Unreal

4. Izrada igre

Za sam početak izrade RPG igre potrebno je napraviti game mode u kojem ćemo određivati pravila igre, slika 6 prikazuje pripremljeni game mode za igru. Najbitnije postaviti Default Pawn Class koji predstavlja plan (eng. Blueprint) lika koji će se stvoriti u igri i Player Controller Class koji predstavlja prilagođen plan tipki koje će igrač koristiti.Hud Class predstavlja izgled što igrač vidi tokom cijele igre kao npr. u igrama s vatrenim oružjem ciljanik. Spectator Class u slučaju da nije dodan Default Pawn Class uzima se taj da se može kružiti po svijetu dok je u izradi. Jedna od prednosti postavljenog game mode sa desnim klikom bilo gdje se želi može se kliknuti Play From Here tamo gdje je desni klik postavljen će igra započeti.



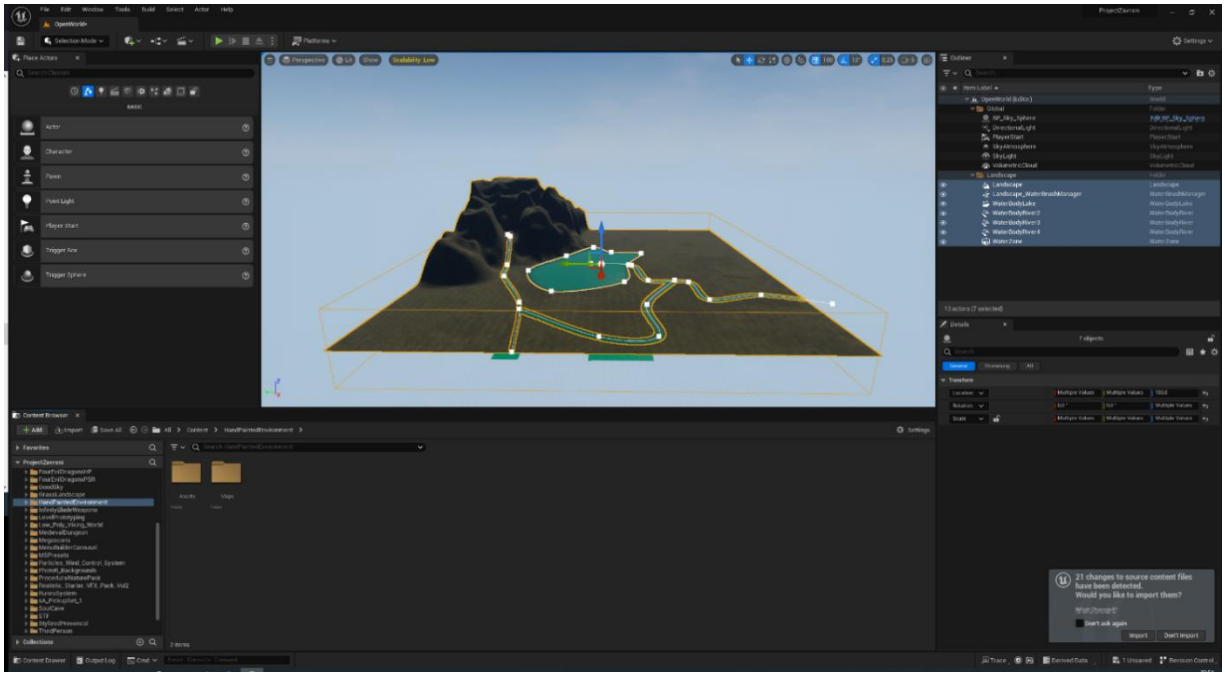
Slika 6. Prikaz novootvorenog game mode i posebnih planova u content browser - u

4.1 Landscape

Stvaranje terena ravnog terena je vrlo jednostavno otvaranjem Landscape mode gdje će već automatski biti otvoreno Manage unijeti veličinu svijetu i pritisnuti Create. Dobit će se ravan svijet s odabranom veličinom koja se pod Manage može mijenjati, nadopunjivati, smanjivati, povećavati itd. Sljedeći korak je obrada terena koja se radi u Sculpt prostoru. Za promjene se koristi kist na kojem se može mijenjati veličina, izgled, snaga, namjena i mnoge ostale. Lijevim klikom na mišu se povisuje teren, shift plus lijevi klik smanjuje. U Unreal 4 kako bi se napravile rijeke i jezera moralo se oblikovati teren, postaviti vodu samo kao element bez ikakvih funkcija, nadodati funkcije vode kao što su valovi i nadodati Post – processing volume[3] za jednostavni prikaz vode. U Unreal 5 postoji dodatak koji se jednostavno dodaje kada se ide u Edit pa Plugins pretraži se Water nadoda i ponovo pokrene uređivač. Pod Quickly Add Objects, otvoriti Select Actors prozor pretražiti Water i prikazat će sve

[3] Post – processing volume – kutija u kojoj kada se igrač nalazi mjenjat ambijent, u ovom slučaju kad igrač uđe u vodu ekran postaje tamniji i plaviji

moguće objekte od dodatka. Water dodatak je još u eksperimentalnoj fazi te zna biti nestabilan, ne raditi po zadanom, bacati grešku, nestajati i još neke od poteškoća. Kada sve proradi kako treba, za projekt trebalo je jezero i rijeke, potrebno je nadodati i dodatak će sam napraviti sve što se je u Unreal 4 moralo ručno. Sve postavke se mogu podešavati, objekta napravi samo s tri točke tako da se moraju nadodavati za bolji oblik, rijeka je napravljena koristeći spline metodu koja će se kasnije obraditi. Slika 7 prikazuje samo teren svijeta.

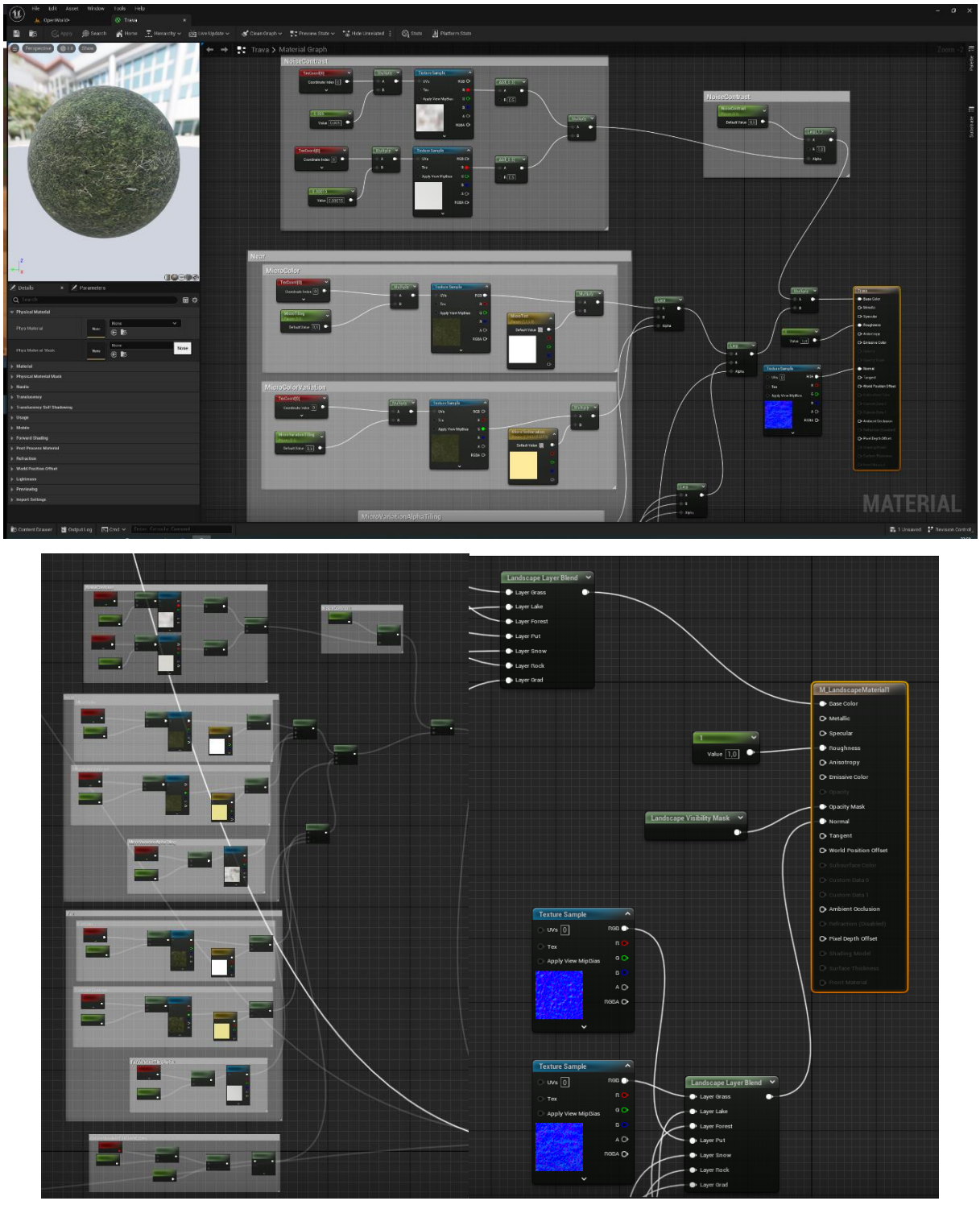


Slika 7. Prikaz terena sa vodom

4.2 Materijal

Jedini način na koji je moguće promijenit teksturu ili boju terena je nadodavanjem tekstura na materijal koji je korišten na elementu Landscape. Materijali se ne koriste samo na terenu već na svim objektima kojima je potrebno zadati neki izgled. Samo dodavanje materijala na teren daje boju terenu sa osnovnim popločenim izgledom. Za puno bolji izgled terena koriste se različite varijacije koje Micro i Macro koje ovisne o tome na kojoj se daljini igrač nalazi prikazuju teren, točnije da ne izgleda pločasto već glatko. Način na koji se to postiže je koristeći šum kojem se zadaje neki broj kolikoće se čisto vidjeti. Također se dodaje i neka boja za raznolikost. Kako bi teren izgledao još glađe i raznolikije pod Landscape u Paint postavkama dok se boja sa drugom teksturom preko prve može se odabrati tekstura šuma i jačina na kistu.

Teksture za teren se mogu pronaći Quixel Bridge ili online, ako se preuzima sa Quixel Bridge dobra navika je preuzeti visoku rezoluciju zbog izgleda. Plave teksture ili Normal je ono što teksturi daje 3D oblik. Na slici 8 se prikazuje materijal editor u koje se prikazuje izgled materijala i kod kojim se sastavlja taj materija.



Slika 8. Prikaz Material Editor, varijacija na teksturi trave i dodavanje normala

4.3 Dodavanje elemenata uz C++

Nakon spajanja svega što je potrebno za razvoj u C++ stvara se C++ klasa koja se automatski otvori u razvojnom okruženju.

```h
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "BP_Dizalo.generated.h"

UCLASS()
class PROJECTZAVRSNI_API ABP_Dizalo : public AActor
{
    GENERATED_BODY()

public:
    // Sets default values for this actor's properties
    ABP_Dizalo();

protected:
    // Called when the game starts or when spawned
    virtual void BeginPlay() override;


public:
    // Called every frame
    virtual void Tick(float DeltaTime) override;

    UPROPERTY(EditAnywhere, Category = "Pomicanje")
    FVector BrzinaDizala = FVector (-300, 300, 600);
    UPROPERTY(EditAnywhere, Category = "Pomicanje")
    float PomaknutaDistanca = 20000;
    FVector PocetnaLokacija;

};
```

```cpp
void ABP_Dizalo::BeginPlay()
{
    Super::BeginPlay();
    PocetnaLokacija = GetActorLocation();
    SetActorLocation(PocetnaLokacija);
}

// Called every frame
void ABP_Dizalo::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    FVector TrenutnaLokacija = GetActorLocation();
    TrenutnaLokacija = TrenutnaLokacija + (BrzinaDizala * DeltaTime);
    SetActorLocation(TrenutnaLokacija);
    float Pomaknuto = FVector::Dist(PocetnaLokacija, TrenutnaLokacija);
    if(Pomaknuto > PomaknutaDistanca)
    {
        FVector Smjer = BrzinaDizala.GetSafeNormal();
        PocetnaLokacija = PocetnaLokacija + Smjer * PomaknutaDistanca;
        SetActorLocation(PocetnaLokacija);
        BrzinaDizala = -BrzinaDizala;

    }

}
```

Slika 9. Prikaz C++ koda za platformu u Visual Studio Code

Otvaraju se dvije datoteke jedna sa nastavkom .h, druga sa nastavkom .cpp kao što se vidi na slici 9. Na lijevoj strani se nalazi kod u datoteci .h u kojoj se ne ništa ne odvija već se samo definiraju varijable i njihova ograničenja. UPROPERTY stvara odjeljak, koji se prikazuje kada se gledaju detalji plana (eng. Blueprint). EditAynwhere označava da se može mijenjati unutar Unreal Editora i Visual Studio Code, a Category u koju skupinu će se varijabla svrstavati. FVector je varijabla u koju se spremaju 3 podatka. U varijabli BrzinaDizala ti podaci služe za određivanje gdje će se objekt nalaziti u 3D prostoru. Float tip podatka koji služi za korištenje brojeva s decimalnom točkom i varijable PomaknutaDistanca koja služi koliko će se objekt pomicati. Na desnoj strani je datoteka sa nastavkom.cpp u kojoj se stvari odvijaju. U funkciju BeginPlay() koja se pokreće na samom početku igre, koristi se funkcija GetActorLocation() kako bi se dobila lokacija objekta u prostoru u igri te se sprema u varijablu PocetnaLokacija koja je tipa FVector. Sljedi funkcija Tick() koje se odvija svake sličice u igri. U FVactor TrenutnaLokacija se postavlja lokacija objekta u

prostoru, nakon toga se TrenutnaLokacija koristi kako bi se pomicala platforma, na varijablu se dodaje BrzinaDizala koja je već definirana pomnožen sa DeltaTime. DeltaTime nam služi kako bi se platforma pomicala konstantno znači da ne ovisi koliko sličica u sekundi se odvija na ekranu. I se koristi SetActorLocation kako bi se postavila TrenutnaLokacija na lokaciju platforme, pošto se Tick() funkcija odvija svake sličice platforma se pokreće jer se cijelo vrijeme nadodaje. Sljedeće je Pomaknuto varijabla u kojoj se dobiva daljina između početne i trenutne lokacije objekta i koja nam služi u if izrazu u kojem moramo znati kada platforma dostigne neku lokaciju da se krene vraćat. GetSafeNormal() funkcija daje stvara kopiju vektora i vraća nul vektor ako je premali za pokretanje. Sada se koristi početna lokacija za pokretanje platforme i kada dostigne lokaciju PomaknutaDistanca kreće u povratnom smjeru samo okrećući varijablu BrzinaDizala.

```
public:
    // Called every frame
    virtual void Tick(float DeltaTime) override;
    UPROPERTY(EditAnywhere, Category = "Rotacija")
    FRotator BrzinaVjetrenjace = FRotator(30, 0, 0);
};
```

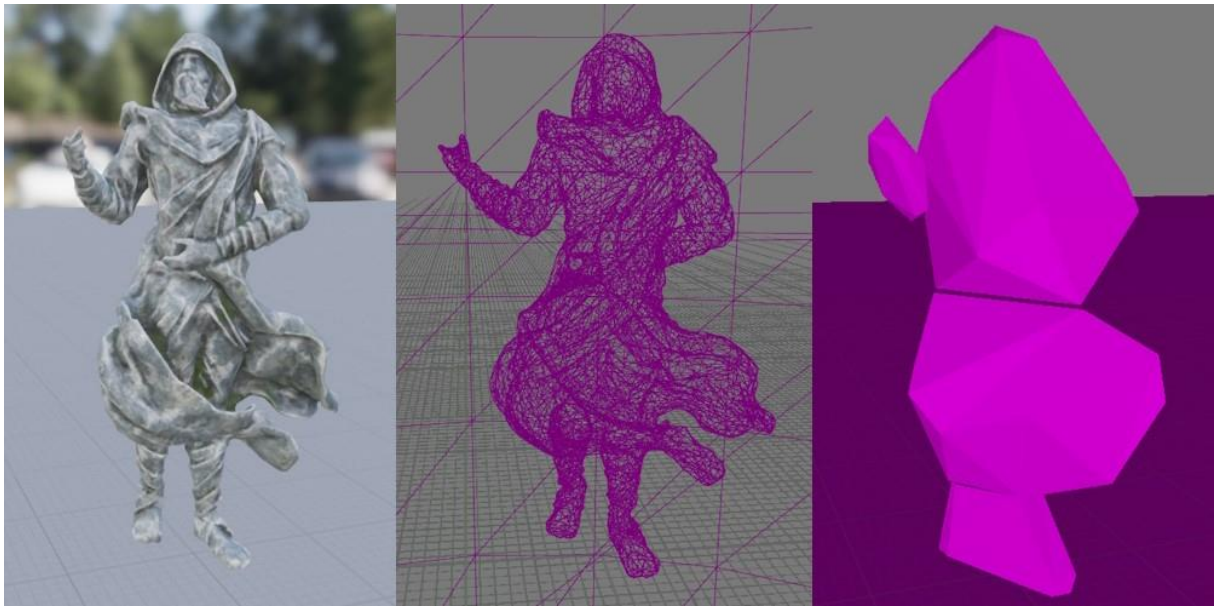
```
#include "BP_Vjetrenjaca.h"
#include "Math/Rotator.h"

// Sets default values
ABP_Vjetrenjaca::ABP_Vjetrenjaca()
{
    // Set this actor to call Tick() every frame.  You can t
    PrimaryActorTick.bCanEverTick = true;

}

// Called when the game starts or when spawned
void ABP_Vjetrenjaca::BeginPlay()
{
    Super::BeginPlay();
}

// Called every frame
void ABP_Vjetrenjaca::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    AddActorLocalRotation(BrzinaVjetrenjace * DeltaTime);
}
```

Slika 10. Prikaz koda za rotirajući objekt

Slika 10 prikazuje koda u .h i .cpp kada se stvori C++ klasa koja napravi dvije nove datoteke, u .h datoteku se stvara varijabla BrzinaVjetrenjace koja je tipa FRotator kao FVector prima 3 podatka X, Y, Z koji će kasnije koristit za smjer u kojem će se objekt okretati. Dodaje se biblioteka Math/Rotator.h u kojem je funkcija AddActorLocalRotation() i unutar funkcije je dovoljno postaviti varijablu koje predstavlja smjer okretanja pomnoženu sa DeltaTime kako bi bila neovisna.

4.4 Foliage mode i static mesh

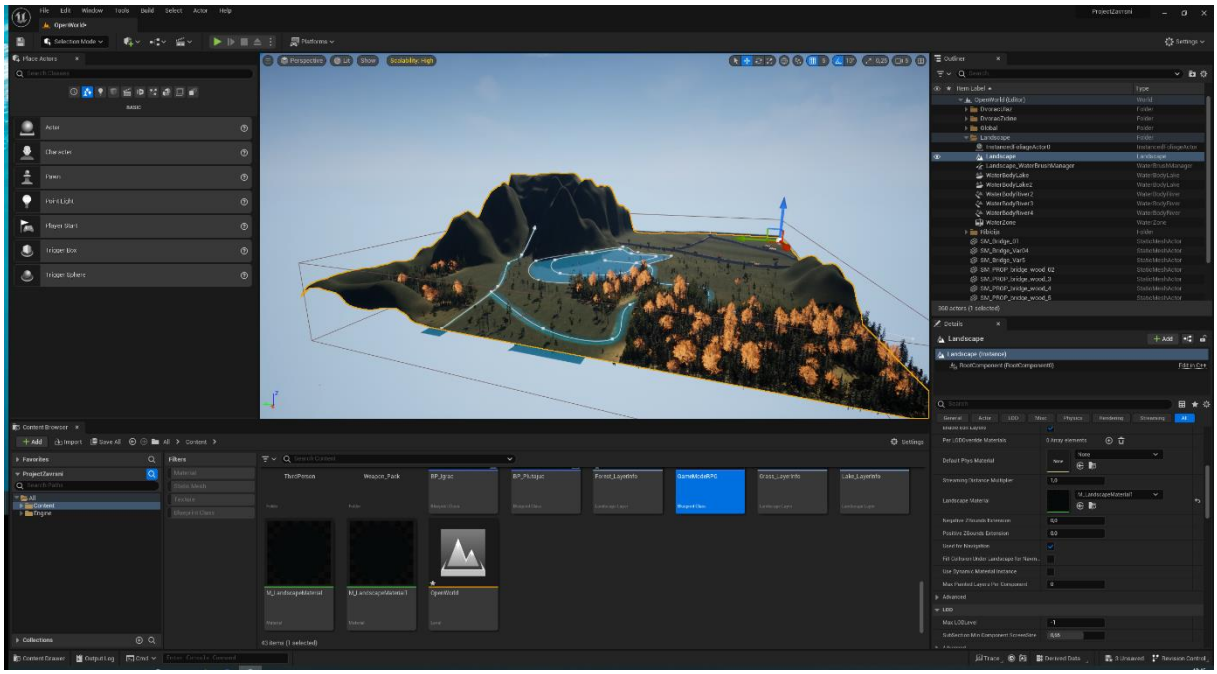
Static mesh su elementi u igri koji su sastavljeni od trokuta i imaju neki izgled kad su postavljeni u svijet, posebni su po tome što se ne mogu animirati već se animacija vrši nad njima. Većinom što više trokuta imaju elementi to će više prostora i video memorije zauzimati za prikaz. Kako bi element dobio izgled potreban je sam kostur elementa tj. trokuti i materijal koji prikazuje boju. Neke od važnijih stvari kod Static mesh su imaju li Collision znači da kad igrač dođe do objekta prolazi li samo kroz objekt ili se zaustavlja, na koju mobilnost je postavljen označava koliko memorije zauzima te je li Physics dozvoljen točnije kada se npr. igrač sudari s objektom hoće li ostati na mjestu ili će se pomaknuti ili može li igrač ručno pomicati objekt svi načini su prikazani na slici 11. Static mesh su preuzeti sa epic games store u vlastništvu epic games.



Slika 11. Prikaz lijevo static mesh kakvog igrač vidi, sredina trokuti koji čine element i desno collision do koga igrač može doći

Pošto se u šumi nalazi puno drva trave i sličnih stvari ručno postavljanje static

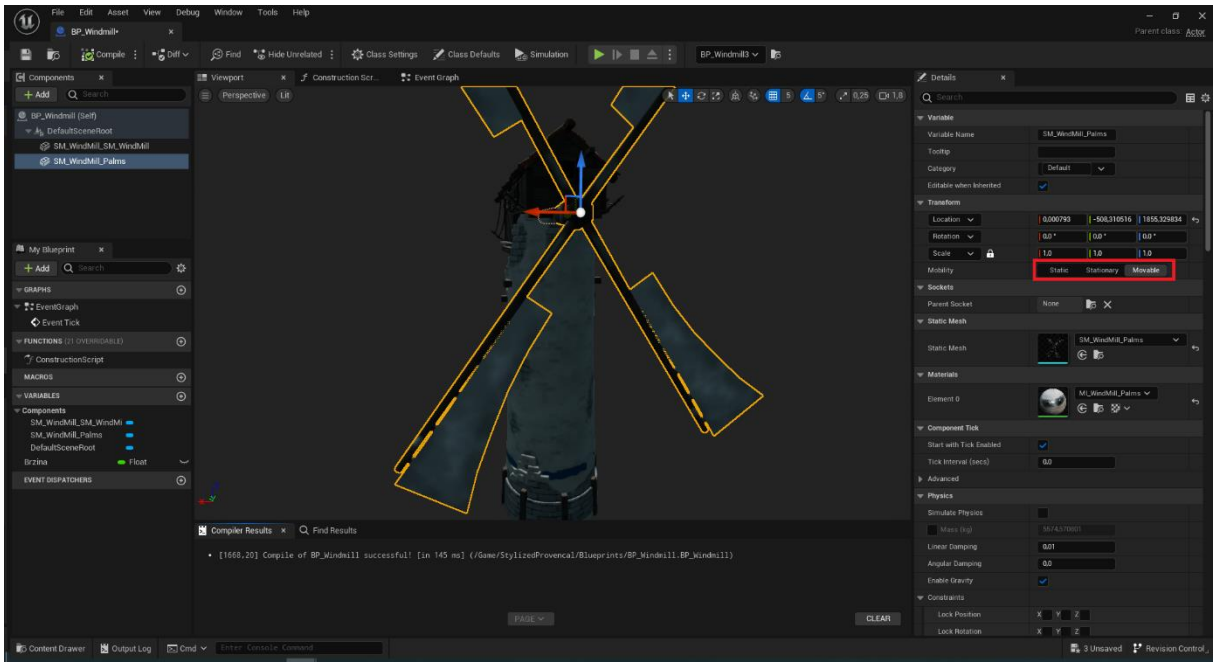
mesh trajalo bi u nedogled. Napravljen je alat s kojim se može odrediti koliko će se elemenata postaviti, njihova veličina i mnoge druge stvari. Za postavljanje mnogo elemenata koji su static mesh koristi se foliage mode. Odabire se foliage mode, otvori se content browser odaberu se elementi koji će se postavljati i nadodaju u prozor foliage mode. Kako bi se mogli uređivati detalji postavljenih elemenata pritisne se ikona u obliku olovke jer elementi koji se postave neće biti postavljeni zasebno već u InstancedFoliageActor plan. Slika 12 prikazuje izražene šumu korištenjem foliage mode.



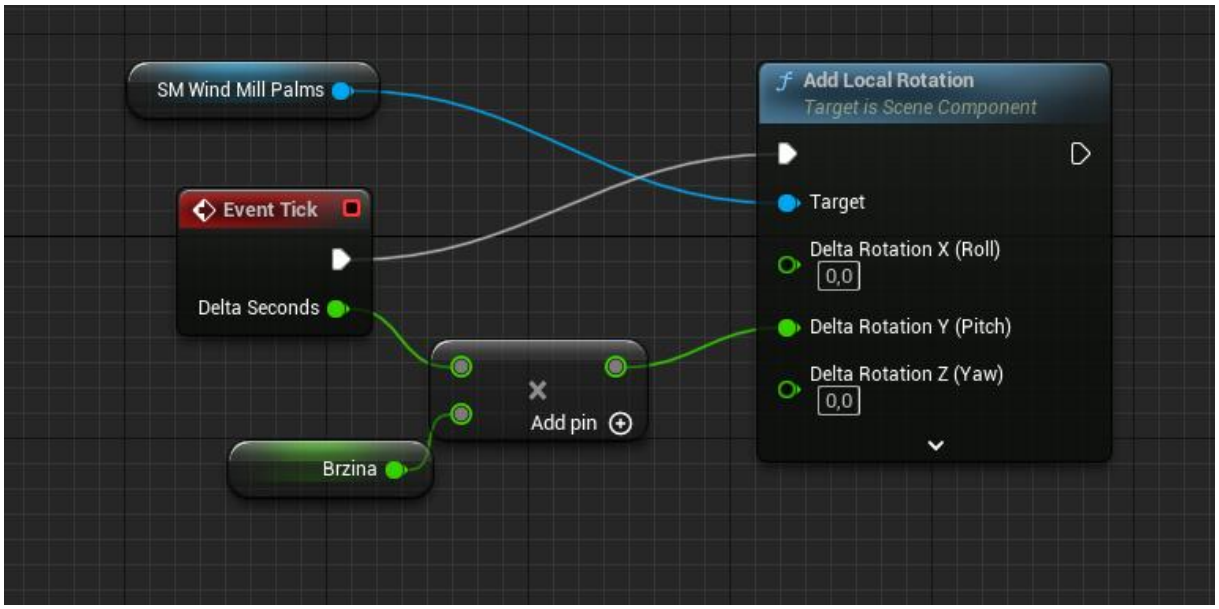
Slika 12. Prikaz šume upotrebom foliage mode

4.5 Blueprints

Blueprints ili planovi su kao klase u objektno orijentiranom programiranju. U Unreal Engine služe kao pristup Blueprint Visual Scripting programskom jeziku. Slike 13. i 14. prikazuju blueprints i event graph za vjetrenjaču.



Slika 13. Prikaz static mesh - a u Unreal Blueprints



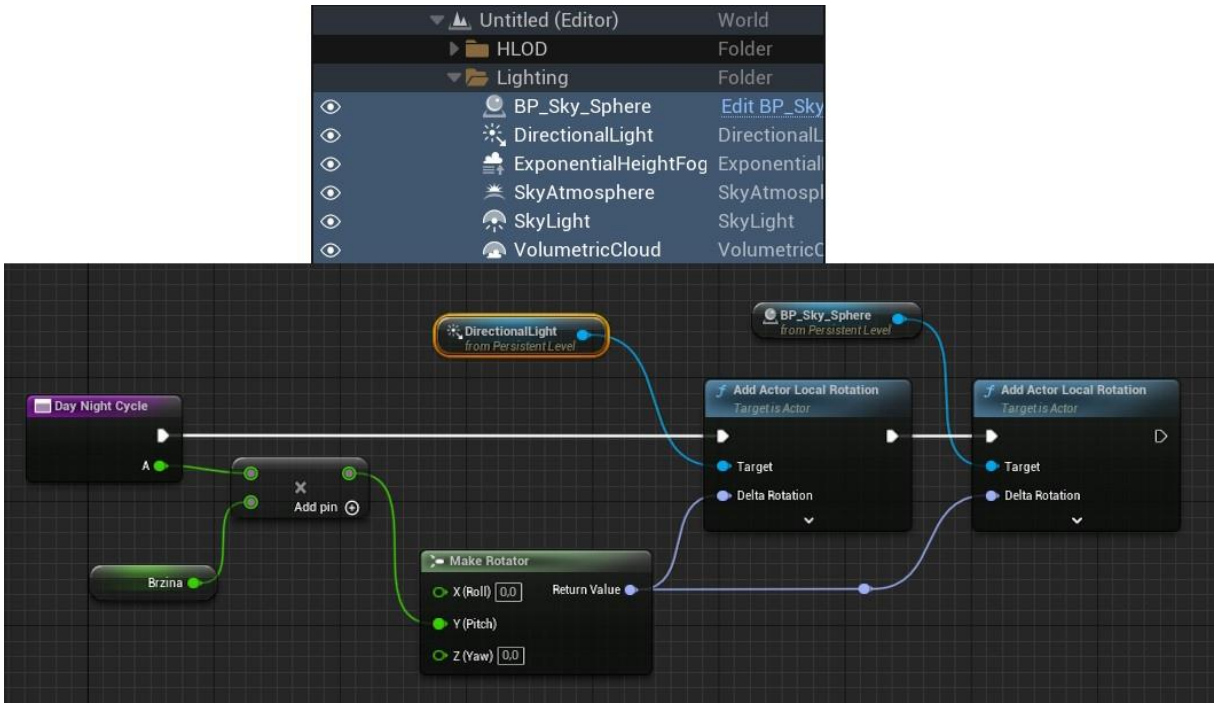
Slika 14. Prikaz koda u Unreal Blueprints

Na slikama iznad može se vidjeti plan za vjetrenjaču u kojem su dva elementa spojena zajedno te element koji se treba pomicati postavljen na Movable. Ispod toga je kod za okretanje tog elementa isti kao u C++. Prozor Add Local Rotation dodaje

dobivenu rotaciju na element. Brzina koja je tipa float je pomnožena sa delta time spojena u Delta Rotation Y, dobivena rotacija. Target je element na koji se rotacija dodaje.

4.6 Nebo

U content browser – u kliknuti na setting i kliknuti na Show Engine Content koji će prikazati stvari koje su već u Unreal Engin – u. Pronaći i dovući BP_Sky_sphere koji služi za reprezentaciju sunca, nepisano pravilo je blueprint – e imenovati sa BP_ na početku, to se pravilo primjenjuje na druge elemente. U actor panel, pod lighting dovući directional light koji služi kao osvijetljene cijelog svijeta. Kod detalja od BP_Sky_sphere postaviti na Movable i Directional Light Actor postaviti directional light koji je povučen u svijet. Kako je svijet osvijetljen se namješta pod direction light, boja svijetla, intenzitet itd. Neki od elemenata koji pridonose boljem izgledu neba su SkyAtmosphere, SkyLight i VolumetricCloud kao što prikazuje slika 15.



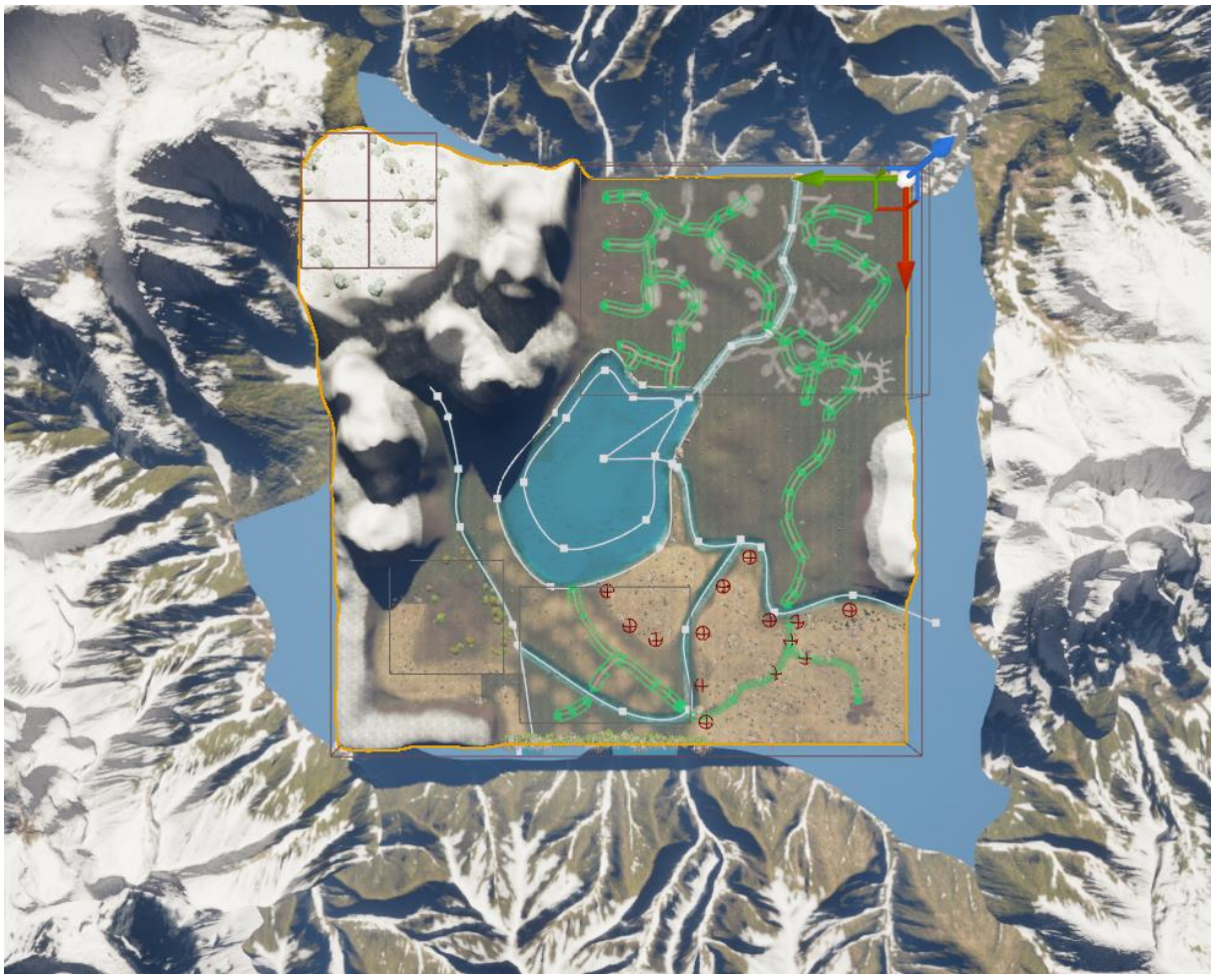
Slika 15. Prikaz elemenata za stvaranje ciklus dana i kod

Napravi se funkcija imenovana Day Night Cycle u kojoj se DeltaTime (A) pomnoženo sa varijablom Brzina koja određuje koliko će dan ili noć trajati. Pomnoženo se spaja sa rotatorom kojeg prosljeđujemo u Add Actor Local Rotation koji pomiče elemente

Directional light i BP_Sky_Sphere te stvara efekt pomicanja sunca. Kako bi bio ciklus noći i dana priključak Sun Location Calculator mora biti nadodan.

4.7 Put i efekti

Putevi su napravljeni pomoću spline pod manage u landscape mode. Nadodaje se novi sloj koji će biti namijenjen putevima te se kreće ocrtavanje puteva nadodavajući nove točke koristeći ctrl plus desni klik. Nakon što je put ocrtan dodaje se static mesh koji će se koristit za izgled put te nakon toga se dodaju materijali za boju. Efekti su blueprints koji su nadodani za bolji ambijent kao što je padanje snijega na planini, padanje lišća u šumi, leptiri kraj trave ribe u vodi i ostali. Planine oko svijeta su postavljene kad igrač pogleda u daljinu vidi da postoji još stvari za otkrivanje inače ne postoji ništa. Exponential Hight Fog nadodaje mističnosti i daljini tih planina tako što stvara magla u odabranoj visini kao što se prikazuje na slici 16.



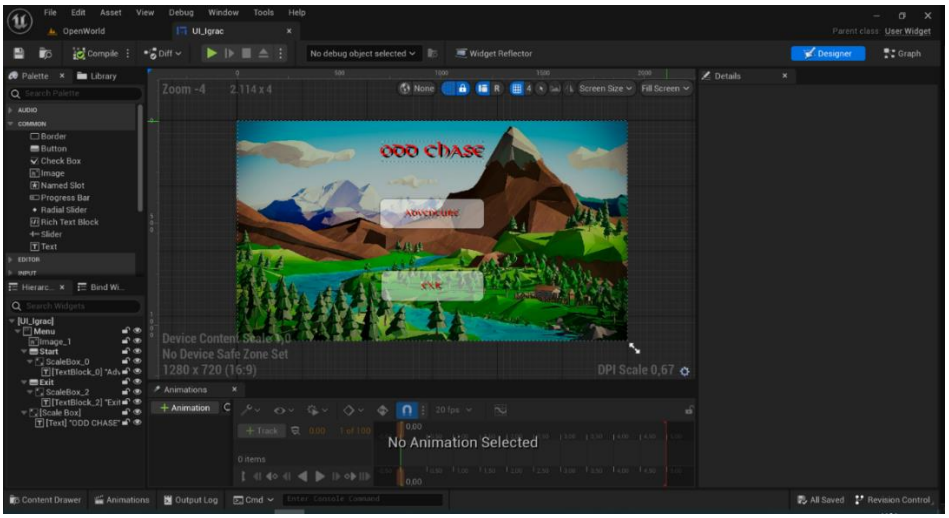
Slika 16. Prikaz puteva i efekata u svijetu

5. Widget blueprint

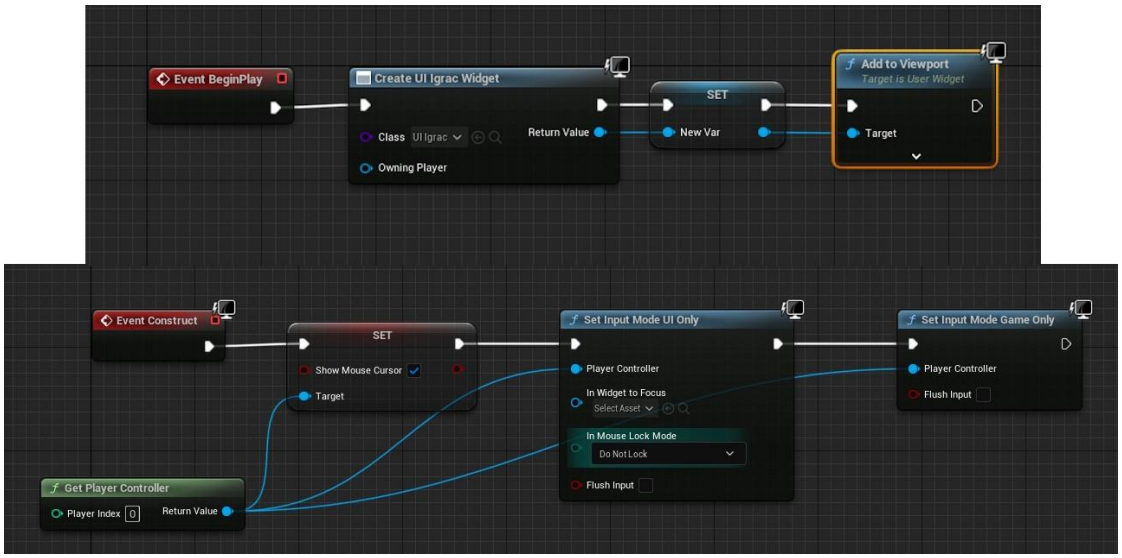
Heads – up display (HUD) je ono što igraču pokazuje statuse kao što su mapa, resursi, zadaci i ostalo. Glavni izbornik je izbornik koji se igraču pokazuje čim otvori igru. Obje stvari se rade sa widget blueprints. Widget blueprint se stvara sa desnim klikom u content browseru pod user interface.

5.1 Glavni izbornik

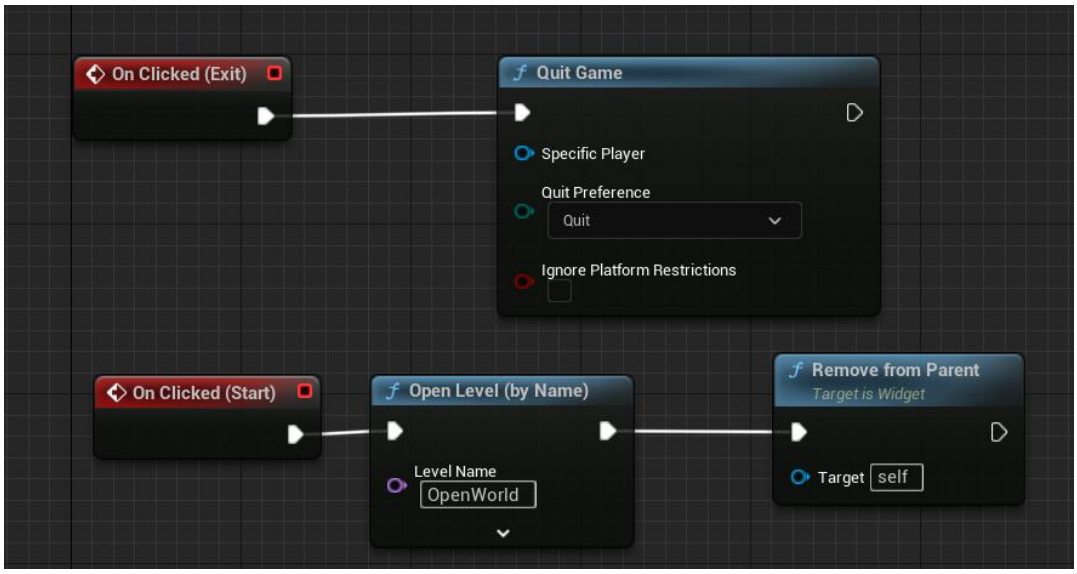
Početkom rada u widget blueprint editor – u najvažnije je postaviti canvas na kojem će se postavljati sve stvari koje će se nalaziti na zaslonu. Canvas se postavlja na rezoluciju ekrana na koju će se prikazivati kao što je normalni ekran 1920 x 1080 te postavljaju se anchor na njega. Anchor je jednostavan način koji se postavlja na elemente i služi kad neko ima veći ili manju rezoluciju ekrana da se elementi na canvas – u skaliraju. Anchor se postavlja na svaki element, slika, gumb, tekst, kutija… Nakon što se postavio canvas sa gornje lijeve strane se dodaju elementi i postavljaju kako će izbornik izgledati. Za svaki element se posebno postavke namještaju. Text je jedini element na kojeg skaliranje s anchor ne djeluje pa je potrebno dodati scale box s kojim djeluje. S desne strane u details prozoru se namještaju postavke, u postavci Is Variable je potrebno staviti kvačicu u kutiju do nje kako bi se omogućile radnje na elementu, kad se ide do dna prozora u skupini events se može dodati funkcija na element koja se može programirati u event graph – u. Donji prozor služi za dodavanje animacija na bilo koji element. Dodaje se animacija pod + Animation nakon toga se dodaje traka pod + Track, odabire se element određuje vrijeme i koja postavlja se željena animacija. Slika 17. predstavlja izraženi glavni izbornik.



Slika 17. Prikaz glavnog izbornika u widget blueprint editor – a



Slika 18. Prikaz koda za postavljanje widget - a



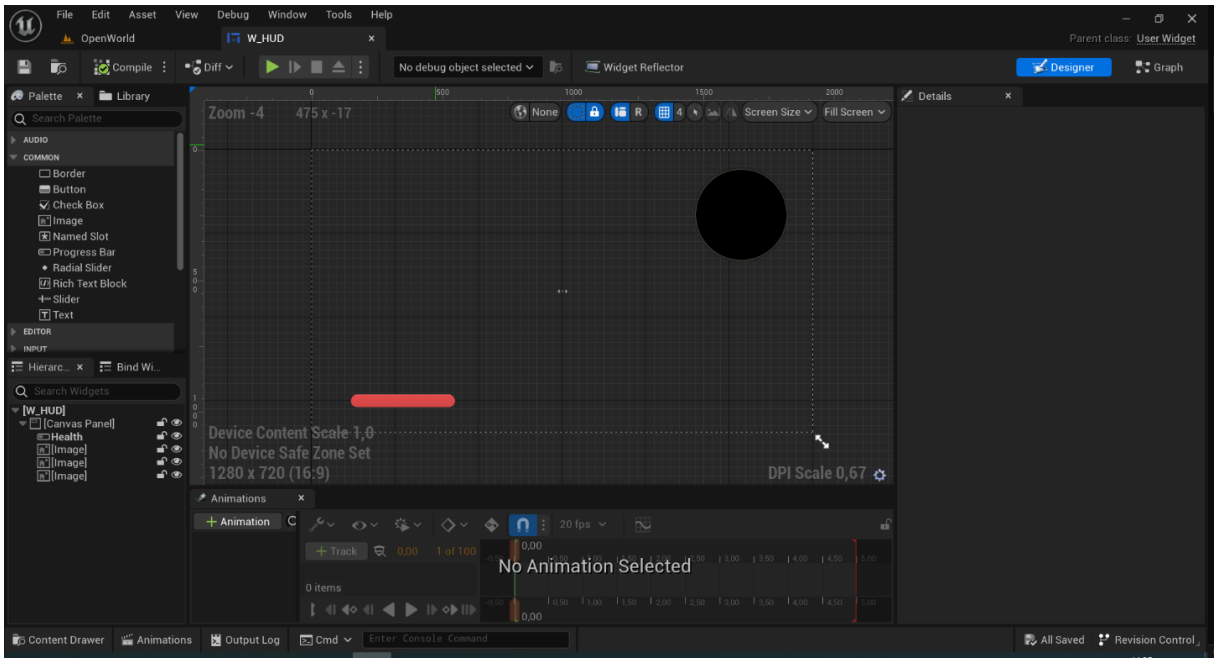
Slika 19. Prikaz koda u event graph za glavni izbornik

Slika 18. prikazuje kod za tipke u glavnom izborniku, postavlja se u level blueprint i mora se postaviti za svaki widget blueprint da bi se pojavio na zaslonu. Create UI Igrac Widget stvara widget i na ljubičastom pin – u se odabire koji widget stvara, nadalje se promovira Return Value promovira u varijablu tako nastaje SET prozor koji služi kao referenca na widget kad se nešto želi uređivati u widgetu nazove se SET koja je varijabla imena New Var u ovom slučaju i uređuje i nakraju Add To Viewport dodaje widget na zaslon igrača. U srednjem djelu se nalaz kod koji poziva miš SET prozor do njega dolazi Get Player Controller koji daje funkcije miša lijevi i desni klik, nakon što je miš prikazan Set Input Mode UI Only znači da igrač dok je u

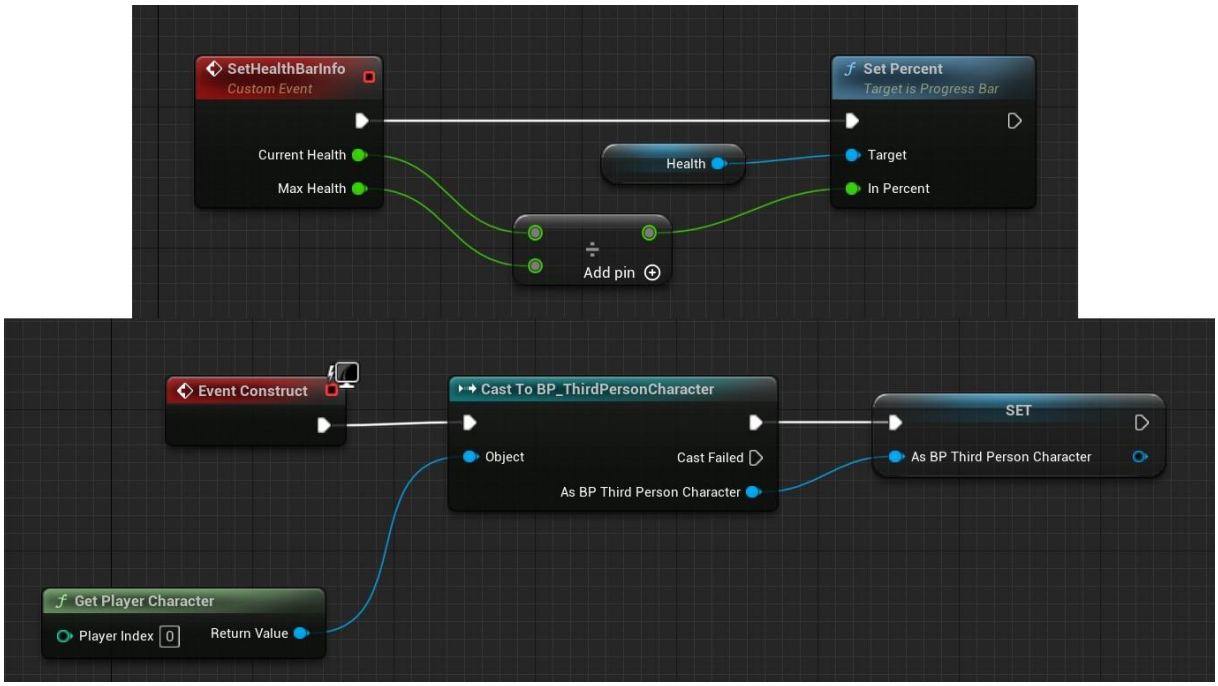
izborniku ne može koristit druge tipke i pomicati se i kada se sve to odvije i nestane izbornik Set Input Mode Game Only vraća igraču kontrole te se može pomicati. Slika 19. prikazuje funkcije gumba na izborniku kada se gumb pritisne On Clicked, ovisno o gumbu (Exit) ide u funkciju Quit Game koja zatvara igru, (Start) ide u Open Level koji otvara level tj. otvoreni svijetu u kojem igrač igra, na ljubičastom pinu se određuje koji level se otvara, mora biti točno napisano jer je osjetljivo na velika i mala slova, i na kraju Remove from Parent koji ukloni izbornik da se ne pojavi kada se otvori level. Font se može pronaći na stranici DaFont, a slike za HUD pretragom na google.

5.2 HUD i damage system

Minimalistički napravljen HUD sa mini map u gornjem desnom kutu canvas – a, ciljnik u sredini, i život u donjem lijevom kutu kao što prikazuje slika 20. Život je namješten kao traka za napredak (eng. Progress Bar) trenutno napunjena do vrha, a mini map kao slika sa svim pripadajućim anchor – ima.

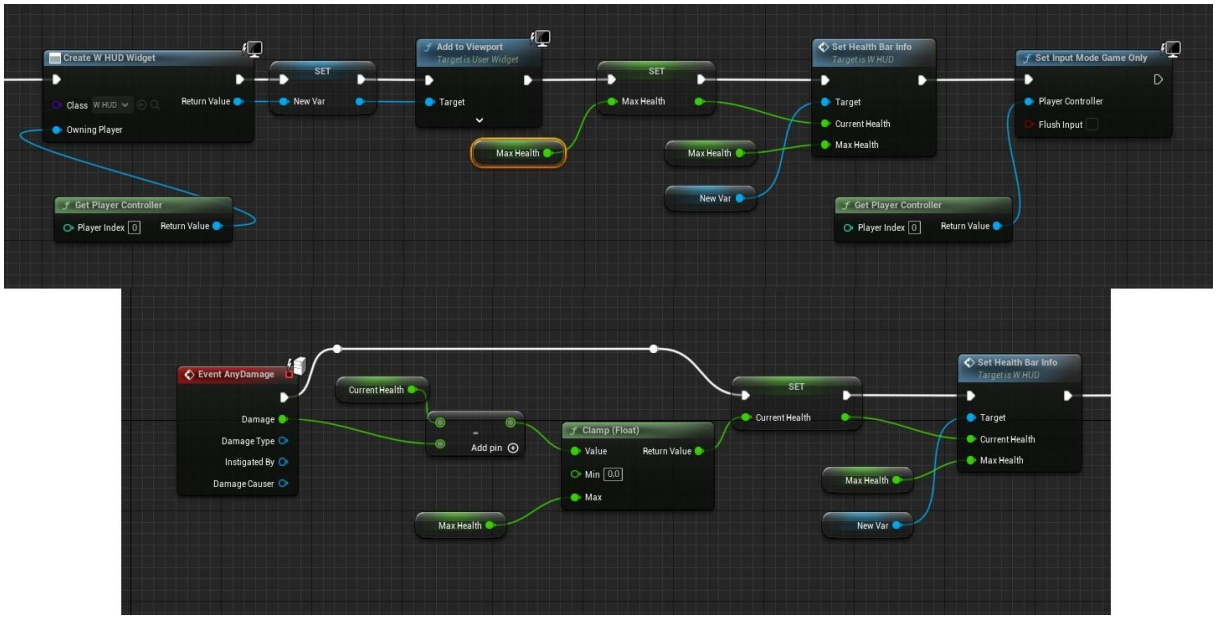


Slika 20. Prikaz HUD igrača



Slika 21. Prikaz koda u grafu HUD za igrača

Donji dio koda stvara objekt koji spaja blueprint lika s kojim igrač igra s widgetom služi da se život može dinamično izmjenjivati tijekom igre. Gornji dio koda stvara se prilagođen događaj sa kojega se dobivaju varijable Current Health i Max Health koje se dijele da bi se dobio postotak koliko života igrač preostalo, nakon toga se u funkciju Set Percent koja namješta postotak varijable Health prosljeđuje dobiveni rezultat. Prilagođeni događaj koristi varijable koje se nalaze u BP_ThirdPersonCharacter lik s kojim igrač igra, mogao se dosegnuti zbog donjeg dijela koda.

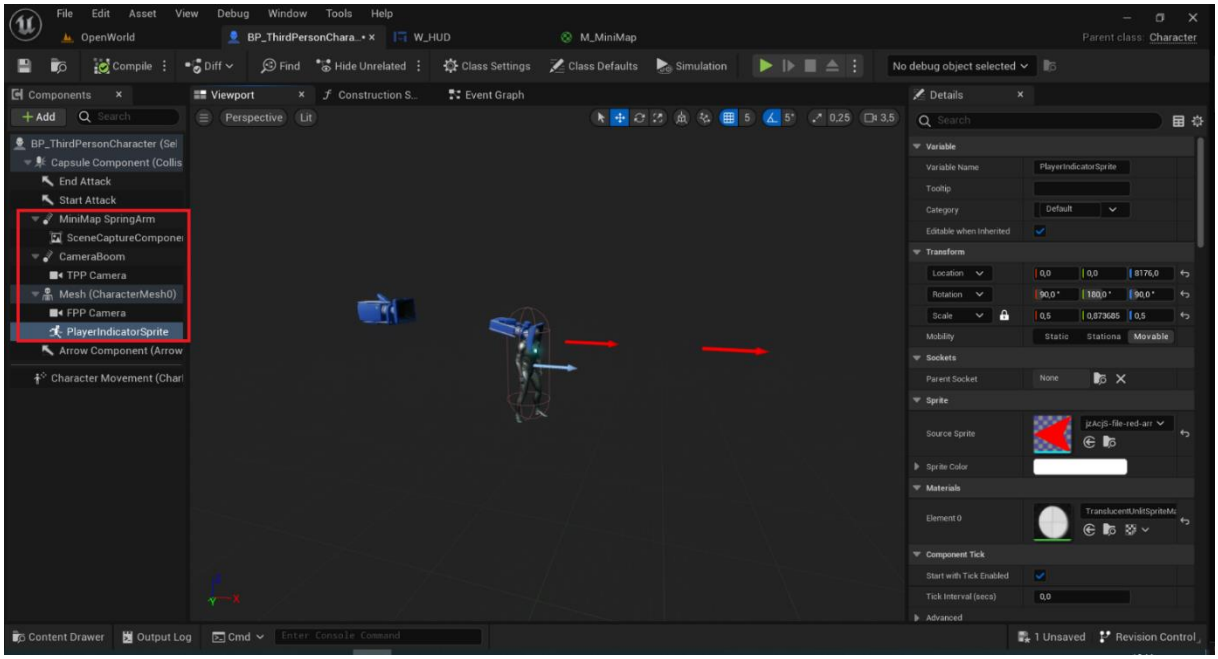


Slika 22. Prikaz koda za HUD igrača i damage system

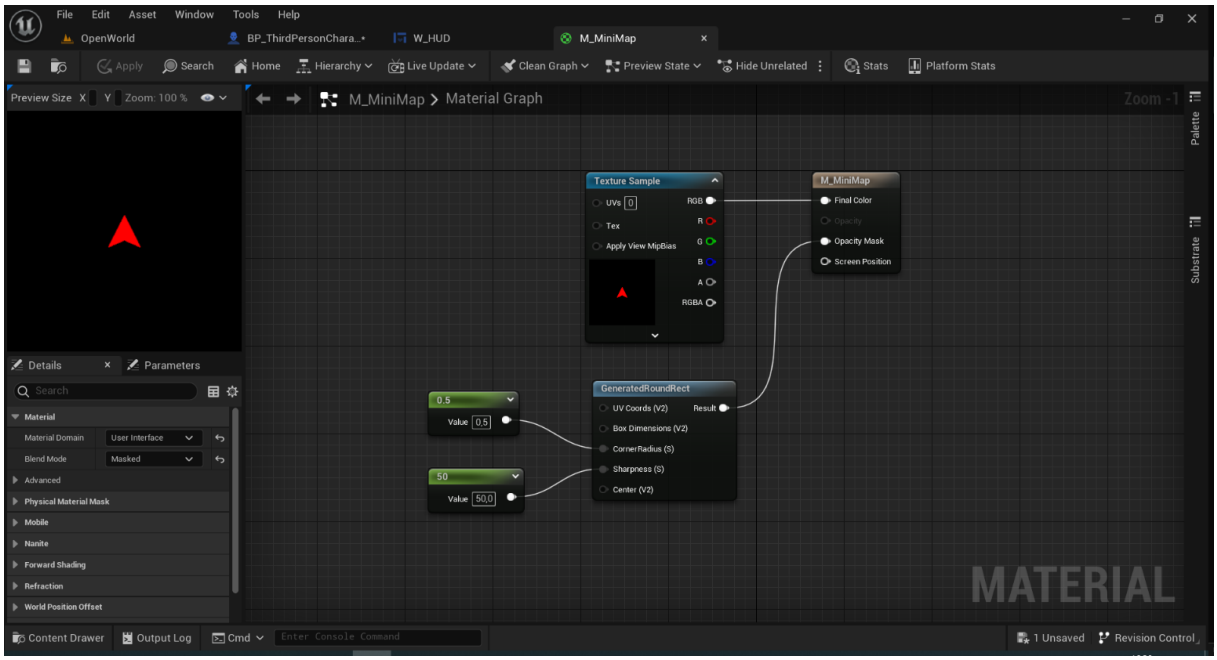
Na slici 22. gornji dio koda stvara widget i postavlja igrač život do vrha da se prikazuje napunjen do vrha sa funkcijom Set Health Bar Info koja prima varijable Max Health kao maksimalni život koji može imati i New Var koja je u ovom slučaju varijabla za widget. Max Health se postavlja kao Current Health koji je trenutni život i tako se postiže maksimalni život na početku i na kraju Set Input Game Only, da igrač ne može ručno mjenjati što se nalazi na ekranu već samo lika. Event AnyDamage je funkcija unutar Unreal Engine – a koja služi za sistem štete, radi tako što od trenutnog života oduzima neka dobivena šteta, funkcija Clamp postavlja minimalne i maksimalne vrijednosti preko koji se ne može prijeći i nadalje se postavlja se trenutni život prosljeđuje i postavlja tako da se Health Bar u igri dinamično izmjenjuje.

5.3 Mini map i izmjena perspektive

Mini map je karta koja prikazuje gdje se igrač nalazi u prostoru trenutno. Nalazi se u HUD igrača, a dobivena je pomoći materijala i lika igrača. Postoji 3 kamere na igraču, jedna služi za kartu, dvije za perspektivu, jedna za pogled iz prvog lica druga iz trećeg lica.

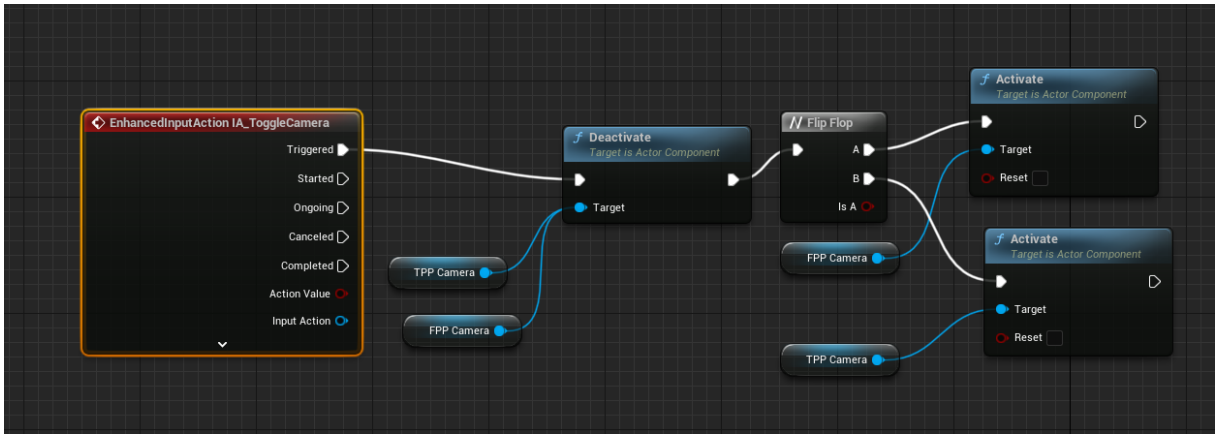


Slika 23. Prikaz lika sa kamerama

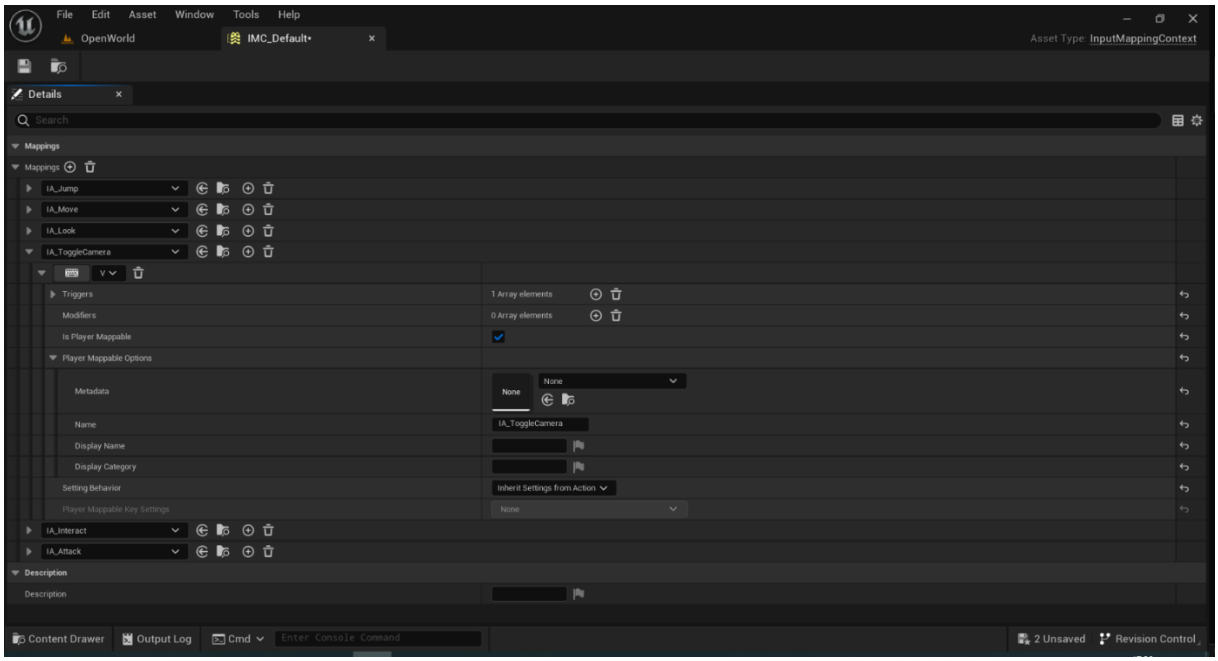


Slika 24. Prikaz materijala za kameru

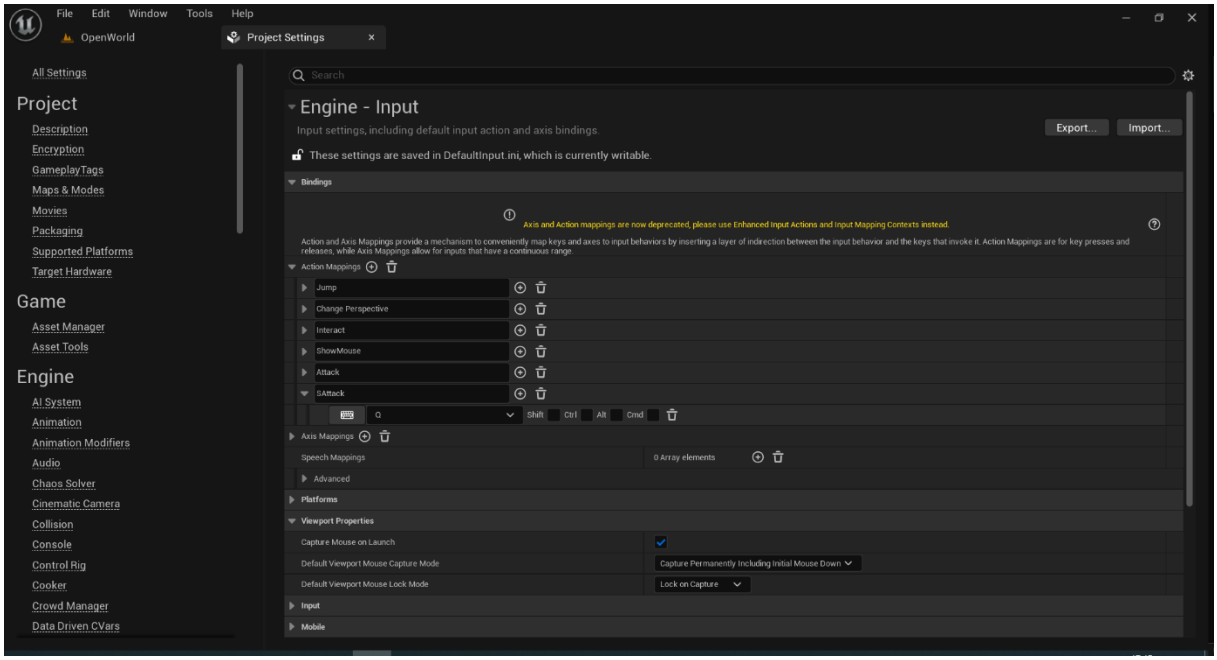
Slika 23. prikazuje igrača s postavljenim kamerama iza igrača je treće lice, unutra igrača prvo lice i za kartu kamera se nalazi na nekoj visini točno iznad igrača. Važnije postavke kako će se karta prikazivati nalaze u kategoriji Projection to su ortogrphic i perspectiv, označavajući hoće li karta biti statična uvijek ista visina ili će se mijenjati. Slika 24. prikazuje materijal za kartu koji je ikona trokuta bez ičega oko njega i funkcija koja pretvara sliku u krug. Materijal se dodaje na sliku u widget – u i još jedna slika kruga iznad slike kao okvir karte. Da bi se pojavio trokut na karti koristi se PlayerIndicatorSprite na kojem je postavljen trokut kao tekstura i stavljen ispred kamere koja snima direktno iznad igrača.



Slika 25. Prikaz koda za izmjenu kamere



Slika 26. Prikaz postupka dodavanja novih funkcija tipki



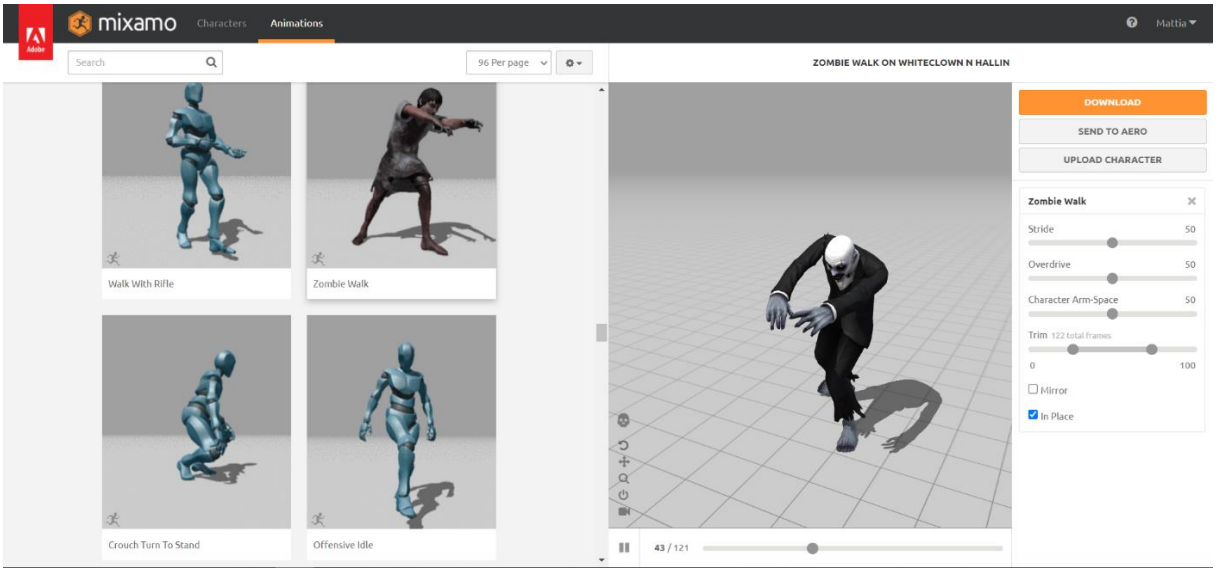
Slika 27. Prikaz dodavanja tipki za korištenje

Za dodavanje funkcije InputAction Change Perspective od Unreal Engine 5.3 verzije bit će samo jedan način za dodavanje koje tipke rade, do Unreal Engine 5.2 postoji tri način. Prvi način je u blueprint editor – u desnim klikom pretražiti tipku koja se želi dodati i dodati ju, drugi način je otići u Project Setting, pronaći Input ispod Engine, i pod Action Mappings dodati tipku kao što prikazuje slika 27. Treći način i jedini koji će se moći koristit nadalje je pronaći gdje se lik nalazi u content browser – u

i pod Inputs lika otvoriti IMC_Default koji je prikazan na slici 24. Kraj Mappings kliknuti na plus ikonu koja dodaje novi tipku, odabrati tipku koja će se koristit, kliknuti na ikonu plusa kraj Triggers odabrati kada će se tipka koristit kada je pritisnuta ili nakon pritiska, označiti Is Player Mappable sa kvačicom, dodati ime za tipku to ime će se koristit za pozvati tipku kao funkciju tijekom kodiranja radnje tipke i na kraju spremiti kao što se prikazuje na slici 26. Kod za promjenu perspektive sa slike 25. djeluje na način da se poziva funkcija EnhancedInputAction IA_Toggle Camera Toggle Camera je kako je nazvana funkcija za promjene perspektive u IMC_Default. Kada se pritisne tipka „V“ jedna kamera se deaktivira dok se druga kamera aktivira. TPP Camera i FPP Camera su varijable koje predstavljaju kamere postavljene u liku.

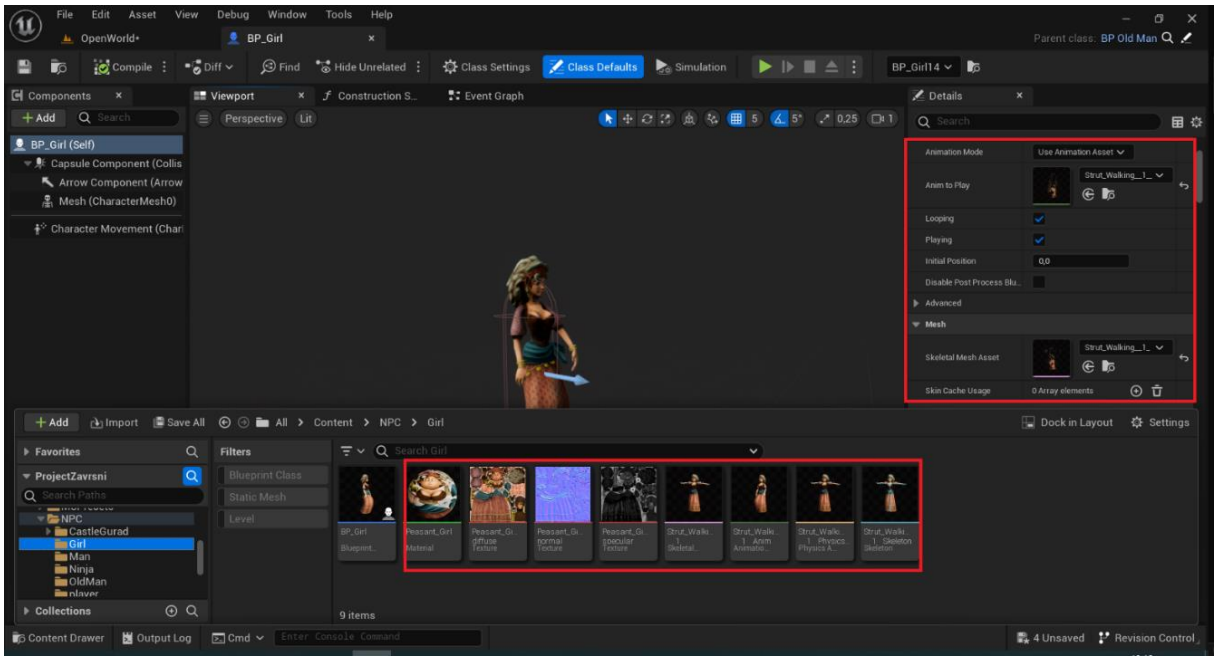
6. Likovi

NPC skraćeno za non – player character su likovi koji se nalaze u svijetu i sa kojima se ne može upravljati. Većinom služe za dodavanje „života“[4] u igru i pomoć pri ispričavanju priče, pod NPC spadaju svi pomični likovi u svijetu. U ovom projektu za likove se koristila online stranica Mixamo sa koje se mogu besplatno preuzeti animacije i izgled likova kao što se prikazuje na slici 28.



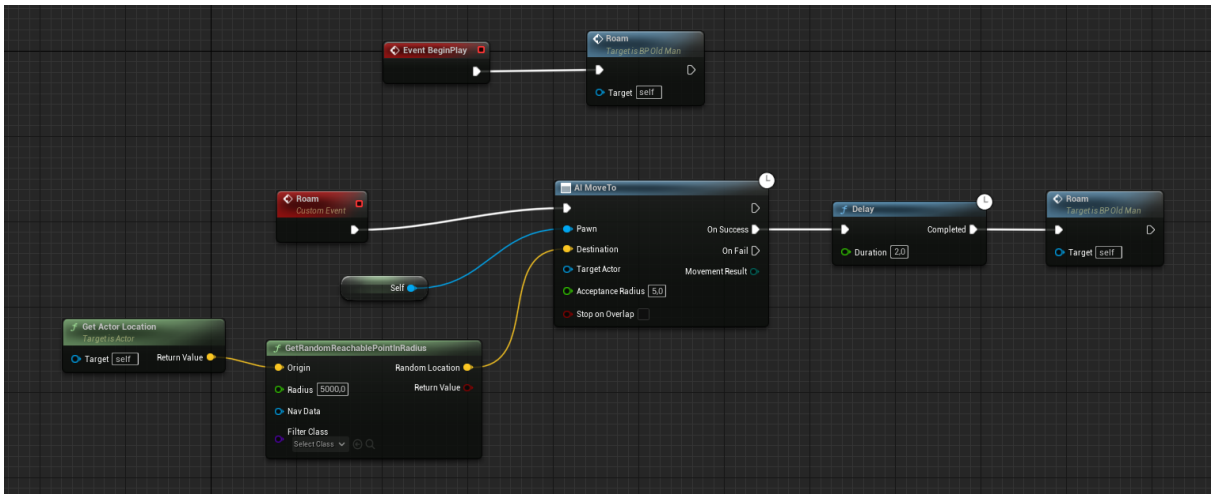
Slika 28. Prikaz Mixamo stranice

[4] „Života” – u ovom smislu misleći kako bi bila pustoš u gradu bez likova



Slika 29. Prikaz NPC blueprint -a

Za stvaranje NPC lika koji se slobodno kreće po svijetu potrebne su stvari koje se nalaze u content browser – u na slici 29. Stvara se novi blueprint tipa character, sa desne strane na details prozoru pod Animation mode odabire se Use Animation Asset i odabere se preuzeta animacija, znači da će odabrani lik konstanto biti u toj animaciji ako je označena looping opcija. Pod skeletal mesh se odabire kostur koji je potrebno koristiti. Sljedeće je potrebno napraviti kretanje NPC – a, to se radi u event graph. Napravi se novi prilagođeni event koji će biti spojen u funkciju AI Move To funkcija koja pomiče lik drugi node koji je spojen u tu funkciju je GetRandomReachablePointInRadius i u taj node ide Get Actor Location koji dobavlja lokaciju lika u svijetu. Funkcija GetRandomReachablePointInRadius služi za pravljenje radiusa po kojem se lik kreće. To je dovoljno da se lik konstantno kreće u svijetu bez zaustavljanja, kako bi izgledalo realnije dodan je delay od 2 sekunde znači da se lik zaustavi i ne radi ništa i nakon opet se kreće kao što prikazuje slika 30. Zadnja stavka u svijetu je potrebno dodati NavMeshBoundsVolume, volumen koji u svijetu pravi gdje je sve liku dozvoljeno kretanje. Dok je odabran NavMeshBoundsVolume pritiskom tipke „P“ prikazuje se zelenom bojom gdje sve lik može ići. Kako se ne bi za svakog lika bilo potrebno raditi ponovo blueprint i kod može se na već napravljeni blueprint desni klik i Create Child Blueprint Class, koji će napraviti novi blueprint sa svim elementim blueprinta iz kojeg je napravljen.



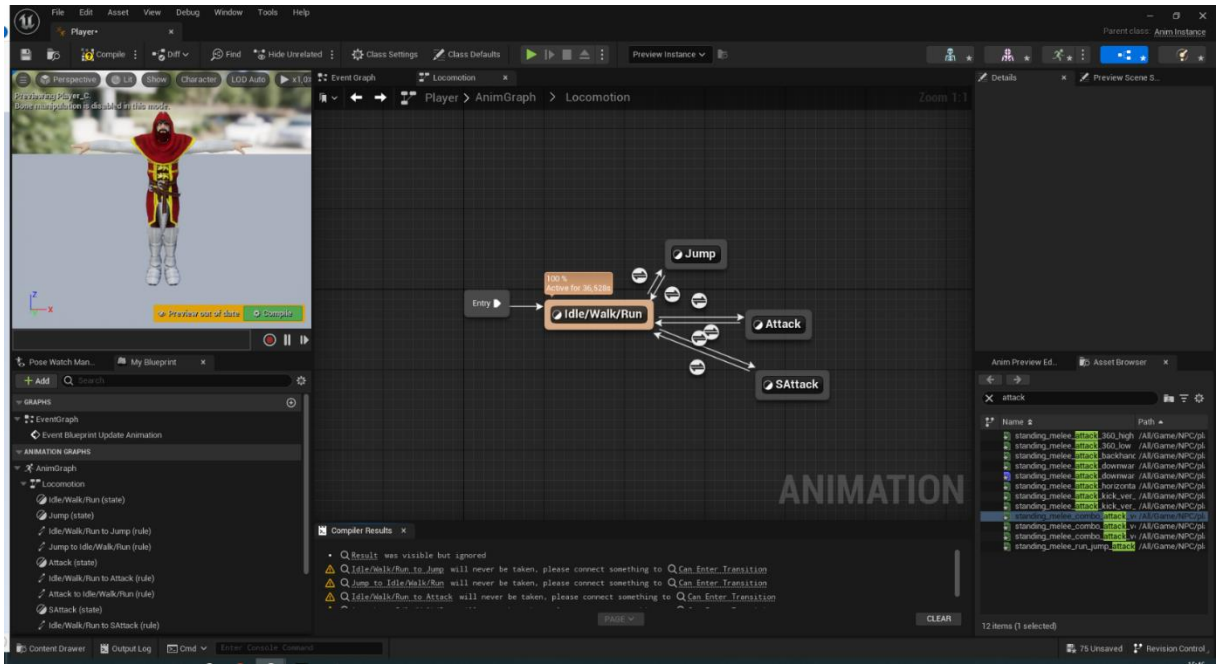
Slika 30. Prikaz koda za slobodno kretanje NPC - a u svijetu

6.1 Animacija

Kako bi se mogao napraviti sustav borbe, novi likovi s više pokreta i mnogo toga potrebne su animacije. Animacija borbe u ovom projekt je napravljena na način kada je potreban napad poziva se određena animacija. Animacija lika napravljen je novi blueprint u kojem je više animacija povezano.

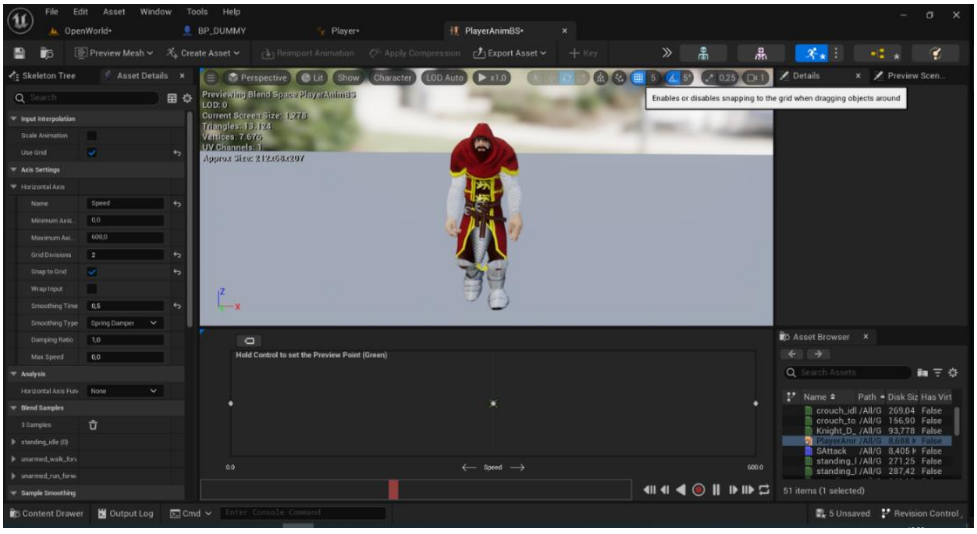
6.2 Animacija lika

Lik koji je prikazan i animacije za njega su preuzete s mixamo stranice. Desni klik u content browser – u i odabirom na animation blueprint otvara se animation editor u kojem su tri glavna prozora: event graph, animation graph i locomotion. Prvi koji će biti otvoren je locomotion u kojem se određuju akcije i koji je odgovoran u kojoj se animaciji se igrač nalazi. Na svakom node – u na kojem je neka animacija kada se otvori node u njega se postavljaju animacije, na linijama koje povezuju animacije kada se otvore se postavljaju varijable koje određuju je li lik u nekoj animaciji.



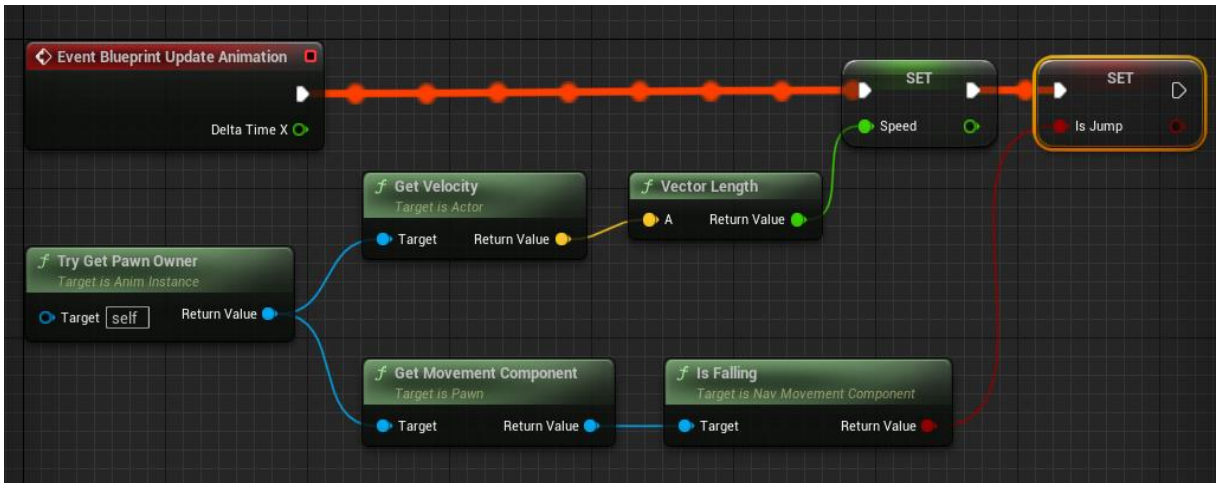
Slika 31. Prikazuje Locomotion prozor

Sljedeća stavka koja se radi su animacije koje su spojene skupa kao što je Idle/Walk/Run ili kad bi bio neki combo napad kao što prikazuje slika 31. Desnim klikom u content browser – u pod animation se napravi Blend Space 1 D kao što se prikazuje na slici 32. U ovom prostoru se prave jednosmjerne animacije, određuje se brzina animacije, pomičnost, statičnost i mnoge druge stvari. Animacija dok igrač stoji, hoda i trči je napravljena na način ovisno o brzini igrač u toj animaciji će se igrač nalaziti. Dakle dodane su tri točke pri početku igrač stoji, u sredini je u hodu i na kraju trči. Pošto se animacije za borbu pozivaju u kodu animation sequence koje su zelene boje moraju biti pretvorene u animation montage plave boje.



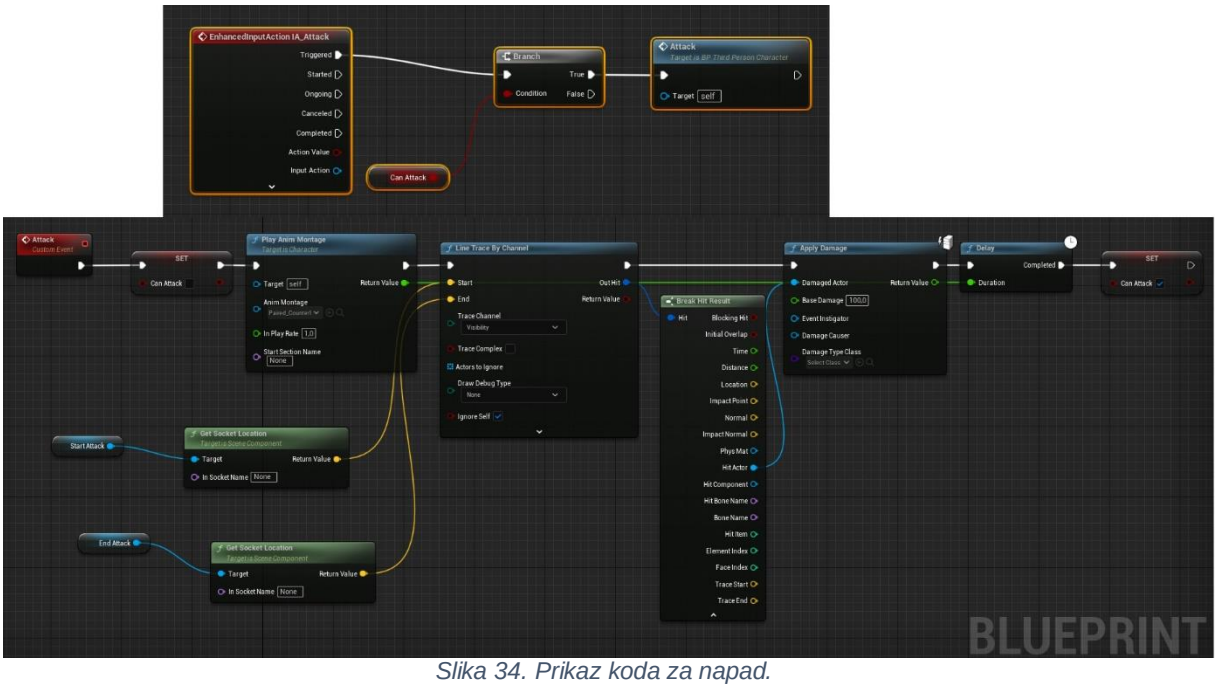
Slika 32. Prikaz Blend Space 1D

Kod koji se prikazuje na slici 33. postavljeno je tako da dobije vlasnik animacije, dobije se brzina pomicanja tog vlasnik pretvori u vektor i postavlja u varijablu Speed koja predstavlja koji će se dio animacije odvijati. Za skok se dobiva kretanje lika i pada li lik ako lik pada stavlja ga se u animaciju skoka.



Slika 33. Prikaz koda za animaciju lika

Za napad lika najprije se postavlja tipka ili tipke koje će se koristiti u IMC_Default, nakon toga se u event grafu poziva funkcija sa kojom se namješta što napad radi, slijedi branch ili if koji sa varijablom Can Attack koja opet služi kao limitator za igrača, može li napasti tek nakon što animacija napada završi, ako je točno poziva se prilagođen događaj Attack.

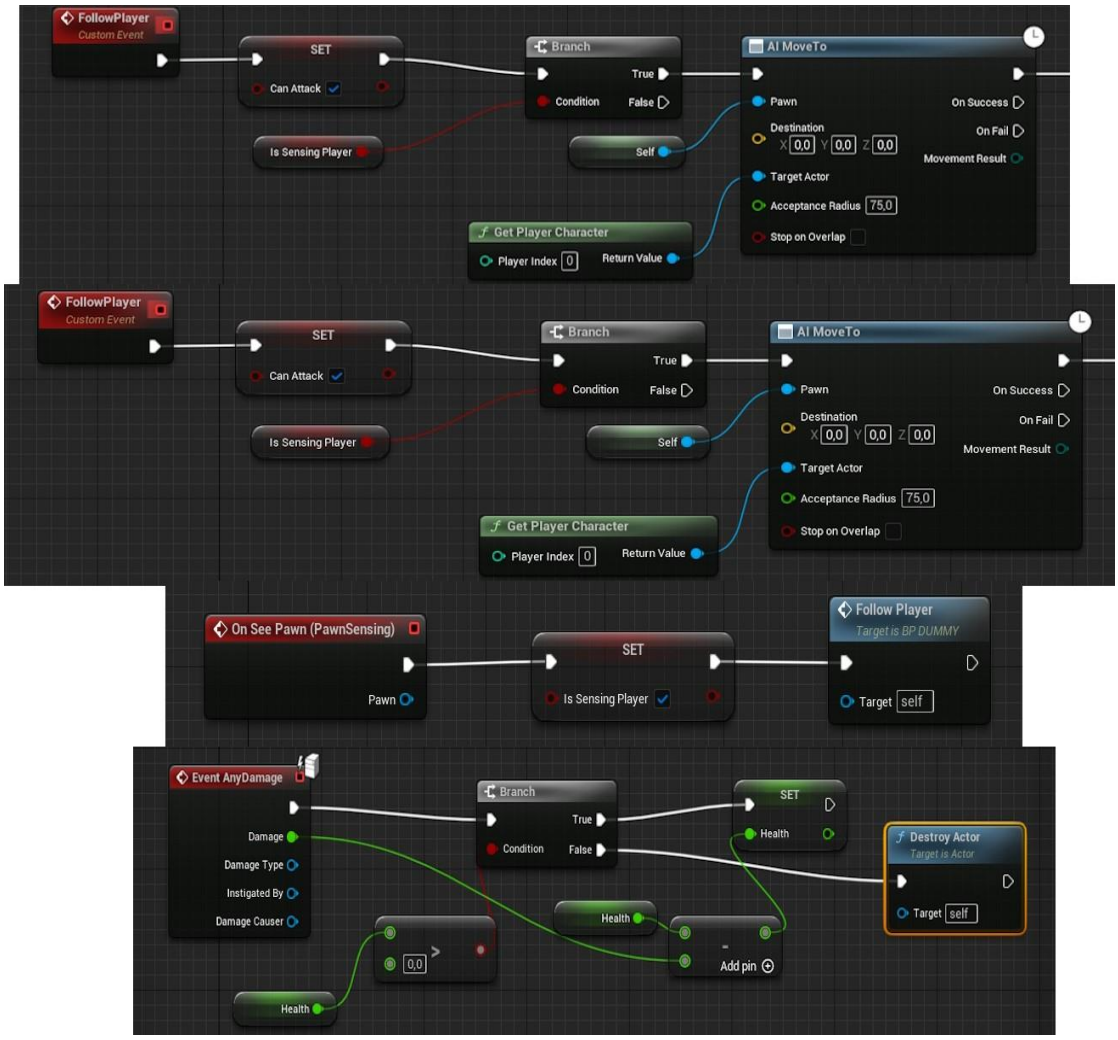


Slika 34. Prikaz koda za napad.

Na slici 34. u donjem djelu koda poziva se prilagođen događaj koji postavlja varijablu Can attack u netočno i funkcija Play Anim Montage pokreće montažu animacije. Line Trace By Channel stvara liniju ispred igrača po kojoj se saznaje je li linija pogodila nešto, ako je linija pogodila u funkciji Break Hit Resault se odabire što napravi sa pogođenom stvari, u kodu je označen Hit Actor kad je pogođen lik funkcija Apply Damage stvara štetu neprijatelju. Varijable Start Attack i End Attack su dobivene iz dvije strelice koje su postavljene ispred igrača u prozoru za prikaz i dobiva se njihova lokacija kako bi se moglo saznati je li nešto pogođenu između početka i kraja linija. Delay na koji se dobiva vrijeme trajanja napada i ne može drugi napad dok ne završi i postavljanje varijable u točno da može sljedeći napad.

6.3 Neprijatelj AI

Izrađuje se character blueprint na kojem se postavlja skeletal mesh kako će neprijatelj izgledati i kao animation asset animaciju hodanja. Dodaje se komponenta PawnSensing koja daje neprijatelju krug u koje on vidi, čuje i osjeća. Na ovom neprijatelju stavljeno je samo kada vidi igrača, na dnu details klikne se On See Pawn koji otvara event graph i postavlja funkciju, također se dodaje varijabla IsSensingPlayer koja je postavljena na true, kada neprijatelj vidi igrača odvija se prilagođen događaj Follow Player. Prilagođen događaj Follow Player ima varijablu Can Attack koja služi čim neprijatelj dođe do igrača ne dopušta animaciju da se odvije do kraja već pokreće iznova konstanto. Branch node služi kao if u programiranju ako neprijatelj vidi igrača kreće za njim. AI Move To pokreće neprijatelja i zaustavlja se na nekoj udaljenosti da napade. Nadalje se odvija animacija napada Can Attack se postavlja u netočno stanje i nakon doga postoji dogoda koji dopušta sljedeći napad tek kad prođe vrijeme animacije. Postavljeni sistem štete djeluje na način da se na neprijatelju zadaje neki broj života, taj broj života se oduzima sa primljenom štetom kada šteta dođe do nula neprijatelj nestaje kao što se vidi na slici 35.

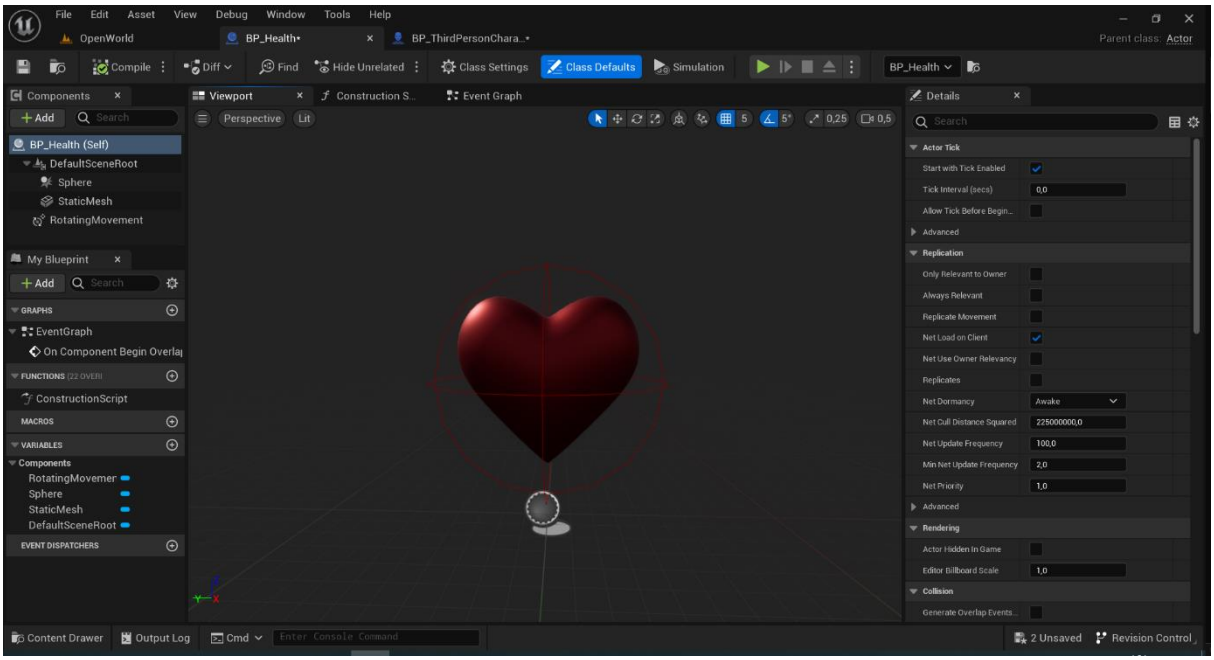


Slika 35. Prikaz koda za neprijatelja AI

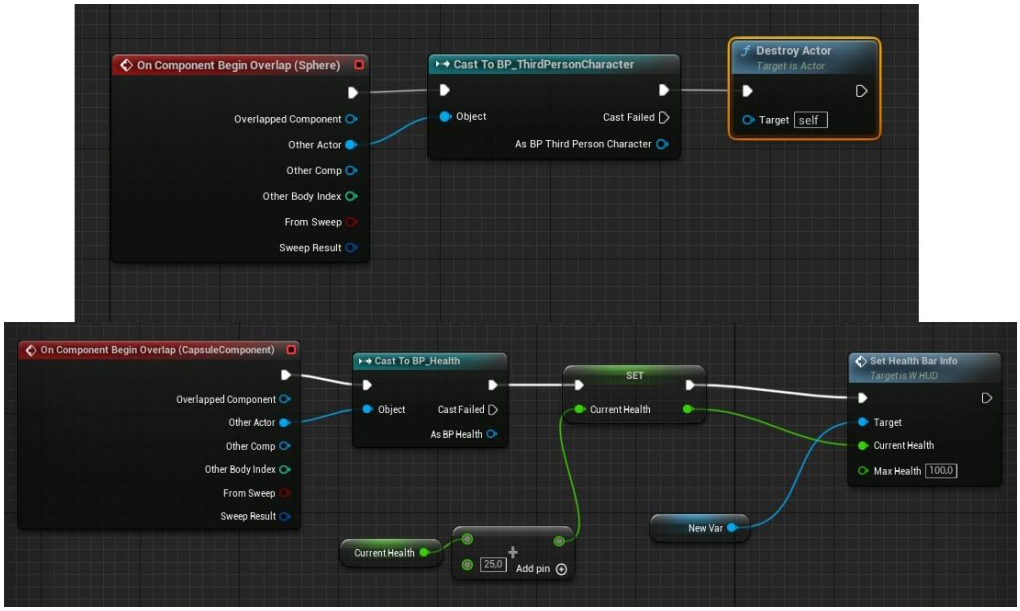
6.4 Život

Vraćanje života u ovoj igri je osmišljeno na način da se treba pronaći u svijetu srca koja vraćaju život. Napraviti actor blueprint u kojega se dodaje static mesh koji izgleda kao srce, napraviti crveni materijal za staviti na srce, materijal se radi na način poziva se funkcija param u kojoj se odabire boja srca i varijabla const koja se stavlja na jedan kako bi materijal imao sjaj. U blueprintu se postavlja static mesh sa materijalom, dodaje se komponenta RotatingMesh koja okreće srce i Sphere Collision koja djeluje kada se igrač sudari sa srcem, pod collision preset je obavezno postaviti OverlapAllDynamic kao što se prikazuje na slici 36.. Na dnu prozora sfere otvara se funkcija On Component Begin Overlap iz koje se podaci šalju na lika sa funkcijom Cast to BP_ThirdPersonCharacter, i kada se igrač sudario sa srcem uništava se. Funkcija On Component Begin Overlap (Capsule Component) kada se kapsula u kojoj se lik

nalazi sudari sa srce šalje podatke blueprint – u od srca postavlja varijablue Current Health za plus 25 i ažurira se stanje trake za napredak srca kao što se vidi na slici 37.



Slika 36. Prikaz blueprint - a za vraćanje života



Slika 37. Prikaz blueprint - a života i kod

7. Zaključak

Izrada videoigara je vrlo zahtjevan posao za jednu osobu, čim jedna osoba sama sve izrađuje i mora imati autorska prava potrebno je izrađivati static mesh, animacije, kod, widget, svijet, zvuk, priče i mnogo ostalih stvari. Nasreću postoji moćam alat kao što je Unreal Engine koji donekle olakšava programeru da ne mora postavljati i same temelje za igru. Tijekom izrade projekta pregledane su mnoge tehnike u razvijanju igara izgleda i navigacija u Unreal Engine – u, stvaranje svijeta, stvaranje materijala, stvaranje i dodavanje blueprint – a, efekti, stvaranje glavnog izbornika i HUD za igrača pomoću widget – a, stvaranje lika, animacija lika i neprijatelja, sistem bitke i štete i drugačiji način za vraćanje života. Naoko ono što igraču izgleda jednostavno za napraviti može imati mnogo linija koda ili povezanih node – ova kako bi funkcioniralo kako treba. Sam proces osmišljavanja igre, postavljanja stvari, stvaranja funkcija i akcija nije toliko kompliciran, dok proces poliranja da sve u jednom radi po zamišljenome je vrlo zahtjevan. Dok sam Unreal Engine 5 daleko od savršenog programa, ima veliki broj funkcija i opcija koje ubrzavaju, olakšavaju, daju ideje što se sve može implementirati u igri, opcije su beskonačne. Stvaranje elemenata i postavljanje koda u igri, ako se postavlja sa Blueprint Visual Scripting, je ekvivalenta objektno orijentiranog programiranja u C++ programskom jeziku na kojem se i Unreal Engine temelji.

Neke od prednosti Unreal Engine – a bi bile dopuštanje izrade igara visoke kvalitete, brz daljnji razvitak programa, naprede postavke za sve elemente, razvitak na raznim platformama i blueprint programski kod koji je na temelju C++ programskog jezika koji je lako pristupačan. Mane Unreal Engine – a su komplicirano korisničko sučelje, mnogo malih detalja kao što su collision ili physics koji utječu radi li kod ili ne, mnogobrojni padovi rada programa, pojava mnogobrojnih bug – ova, potrebno je snažno računalo za održavanje programa i mnogo memorije i nedostatak ili manjak ažuriranja dokumentacije. Jedna od glavnih prednosti Unreal Engine je lako stvaranje velikog otvorenog svijeta koji oduzima dah samo sa elementima koji su besplatni u Quixel Bridge ili Epic Games Store. Mnogo stvari se još nalaze u eksperimentalnoj fazi te se mogu čudno ponašati tijekom uređivanja i izrade igre. Neke od stvari koje bi se još trebale nadodati su inventory sustav s kojim bi igrač izrađivao stvari, sustav quest – ova,sustav napredovanja, sustav kupovanja i prodaje i animacij za plivanje, pametni neprijatelji i bolje animacije za bitku.

8. Literatura

1. Satheesh, PV. (2016.) Unreal Engine 4 Game Development Essentials. Birmingham. Packt Publishing Ltd. datum pristupa 1.8.2023

2. Tristem, B., TEAM, G. i ULIBARRI, S. (2023) Unreal Engine 5 C++ Developer: Learn C++ & Make Video Games. https://www.udemy.com/course/unrealcourse/ datum pristupa 1.8.2023

3. Unreal Engine 5 (2022.), Documentation https://docs.unrealengine.com/5.2/en-US/ datum pristupa 1.8.2023

4. Unreal Engine 5 (2022.), Understanding the Basics https://docs.unrealengine.com/5.2/en-US/understanding-the-basics-of-unreal-engine/ datum pristupa 1.8.2023

5. Unreal Engine 5 (2022.), Building Visual Worlds https://docs.unrealengine.com/5.2/en-US/designing-visuals-rendering-and-graphics-with-unreal-engine/ datum pristupa 5.8.2023

6. Unreal Engine 5 (2022.), Designing Visuals, Rendering, and Graphics https://docs.unrealengine.com/5.2/en-US/designing-visuals-rendering-and-graphics-with-unreal-engine/ datum pristupa 5.8.2023

7. Unreal Engine 5 (2022.), Programming and Scripting https://docs.unrealengine.com/5.2/en-US/unreal-engine-programming-and-scripting/ datum pristupa 20.8.2023

8. Unreal Engine 5 (2022.), Animating Characters and Objects https://docs.unrealengine.com/5.2/en-US/unreal-engine-programming-and-scripting/ datum pristupa 25.8.2023

Popis slika

Sažetak

Cilj završnog rada bio je upoznavanje sa metodama izrade igre, upoznavanje sa Unreal Engine alatom i izrada RPG igre otvorenog svijeta. Odd Chase RPG otvorenog svijeta ima većinu elemenata klasične RPG igre kao što je bitka, istraživanje i likovi. Veliki ručno sastavljeni otvoreni svijet u kojem se igrač može upustiti. Animacije i korisničko sučelje, sistem bitke i drukčiji sistem vraćanja života. Cilj igre je avantura, istraživanje i zabava u otvorenom svijetu. Upoznavanje sa blueprints visual scripting programskim jezikom i korištenje c++ programskog jezika sa razvoj.

Ključne riječi: Unreal Engine, RPG, Otvoreni svijet

Abstract

The goal of the bachelor thesis was to get acquainted with the methods of making a game, to get acquainted with the Unreal Engine tool and to make an open world RPG game. Odd Chase Open World RPG has most of the elements of a classic RPG game such as battle, exploration and characters. A large hand-crafted open world for the player to venture into. Animations and user interface, battle system and a different life recovery system.Goal of the game is adventure, exploration and having fun in open world. Getting to know the blueprints visual scripting programming language and using the c++ programming language with development.

Key words: Unreal Engine, RPG, Open world