# Izrada 3D akcijske videoigre iz trećeg lica u programskom alatu Unreal Engine 5

**Romić, Borna**

**Master's thesis / Diplomski rad**

**2023**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:211:650791

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**V A R A Ž D I N**

**Borna Romić**

# IZRADA 3D AKCIJSKE VIDEOIGRE IZ TREĆEG LICA U PROGRAMSKOM ALATU UNREAL ENGINE 5

**DIPLOMSKI RAD**

**Varaždin, 2023.**

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**V A R A Ž D I N**

**Borna Romić**

**Matični broj: 45857/07–R**

**Studij: Baze podataka i baze znanja**

# IZRADA 3D AKCIJSKE VIDEOIGRE IZ TREĆEG LICA U PROGRAMSKOM ALATU UNREAL ENGINE 5

**DIPLOMSKI RAD**

**Mentor:**

Doc. dr. sc. Mladen Konecki

**Varaždin, lipanj 2023.**

*Borna Romić*

**Izjava o izvornosti**

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

*Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi*

## Sažetak

Ovaj projekt uključuje stvaranje 3D akcijske video igre pomoću softverskog alata Unreal Engine 5, koristeći nacrte za razvoj. Igra je osmišljena iz perspektive trećeg lica, pružajući igračima iskustvo slično „Souls-like“ žanru videoigara. Napredne značajke i opsežan skup alata Unreal Engine 5 iskorišteni su za razvoj videoigre pritom poštujući sva pravila Objektno Orijentiranog Programiranja. Sveukupno, ovaj je pothvat pokazao snagu i svestranost Unreal Enginea 5 u oživljavanju uzbudljive 3D akcijske igre.

Cilj diplomskog rada je postaviti kvalitetne temelje za daljnju izradu videoigre kroz implementaciju svih najvažnijih mehanika za glavnog lika i za neprijatelje (Umjetnu Inteligenciju).

Cilj videoigre je pobijediti sve glavne neprijatelje kako bi se došlo do konačne razine.



**Ključne riječi:** Unreal Engine 5, nacrt *(eng. Blueprint)*, videoigra, animacija, Objektno Orijentirano Programiranje, model *(eng. Mesh)*, komponente

# Sadržaj

# 1. Uvod

Videoigra je posebna vrsta softvera koja uključuje interakciju s korisničkim sučeljem i ulaznim uređajem kako bi generirala vizualne povratne informacije s uređaja za prikaz, najčešće prikazanog u video formatu na televizoru, monitoru računala, ekranu mobitela i sl.

Primarna svrha videoigara je pružiti stimulaciju mozga kako bi se osoba oslobodila od stresa te uživala u samom činu igranja videoigre, međutim, prednosti igranja videoigara uključuju i poboljšanu sposobnost koncentracije, kreativnost, pamćenje, učenje jezika i timski rad. Videoigre mogu olakšati učenje obrazovnih sadržaja i razvoj kognitivnih vještina [1].

Industrija videoigara jedna je od najbrže rastućih industrija, godine 2022. prihod (u milijardama USD) bio je $220.79 te je predviđen rast do $583.69 u godini 2030 [2]. Takav rast u industriji ne prolazi ispod radara te se sve više radnih mjesta otvara za game developer pozicije. Velik poticaj u rastu industrije videoigara je izlazak najmoćnijeg video game engine Unreal Engine 5.

Najnovija verzija Unreal Engine 5 omogućuje stvaranje impresivnijih i realističnijih igara nego ikad prije.

- Nanite i Lumen omogućuju vrlo detaljne i složene scene i nude dinamičnu i realističnu rasvjetu u stvarnom vremenu bez žrtvovanja performansi.
- Chaos Physics, MetaSounds i Improved Animation pružaju mogućnost stvaranja svjetova s realističnom i interaktivnom fizikom, bogatim zvukovima i uvjerljivim animacijama s izvanrednim proceduralnim mogućnostima.
- World Partition omogućuje izgradnju velikih svjetova u velikom timu gotovo bez međusobnog blokiranja.

[3]

Činjenica da je industrija videoigara jedna od brže rastućih industrija, ali i osobnog užitka i brojnih videoigara koje sam odigrao i sam napravio su samo jedni od brojnih faktora zašto sam uzeo temu za diplomski rad upravo izradu videoigre i to upravo u programskom alatu Unreal Engine 5, koji zahvaljujući revolucionarnim tehnologijama koje nudi, omogućuje svakom početniku da s određenom količinom iskustva napravi videoigru točno kakvu je zamislio.

Glavni cilj ovog diplomskog rada je prikazati uporabu programskog alata, postupak izrade videoigre, tijek i način razmišljanja tijekom programiranja videoigre u programskom alatu Unreal Engine 5 i pritom poštujući pravila objektno orijentiranog programiranja (OOP).

# 2. Metode i tehnike rada

Kao što je prethodno spomenuto programski alat koji će se koristiti u ovom diplomskog radu je Unreal Engine 5.

Programiranje u Unreal Engine 5 može se odvijati korištenjem C++ programskog jezika, u kojem je ujedno i napisan, za izradu vlastitih skripti koje se pokreću u samom pogonu igre. Alternativa, koja će se koristiti u sklopu ovog diplomskog rada je programiranje korištenjem nacrta (*eng. blueprint)*. Nacrti su vrlo moćni gotovi blokovi koda koji se dodaju objektima za interakciju te svakom novom iteracijom programskog alata Unreal Engine, nacrti se sve više plasiraju kao jednako dobar odabir u programiranju videoigre u programskom alatu kao i C++ programski jezik. Velika prednost korištenja nacrta je upravo jednostavnost njegove uporabe koja omogućuje početnicima programskog alata brži razvoj (učenje, skriptiranje, debugiranje) te korištenje složene C++ kodove, koje je Epic Games napravio za nas [4].

„Nacrt, odnosno, Blueprint Visual Scripting sustav u Unreal Engineu je cjelovit sustav skriptiranja igranja zasnovan na konceptu upotrebe sučelja temeljenog na čvorovima za stvaranje elemenata igranja iz Unreal Editora. Kao i kod mnogih uobičajenih skriptnih jezika, koristi se za definiranje objektno orijentiranih (OO) klasa ili objekata u stroju.“ [4]

Ovaj sustav je izuzetno fleksibilan i moćan jer pruža mogućnost dizajnerima da koriste gotovo čitav niz koncepata i alata koji su općenito dostupni samo programerima te je svojim jednostavnim i intuitivnim dizajnom idealan odabir za početnike koji žele programirati svoju prvu videoigru.

Pri izradi videoigre veliki utjecaj i pomoć oko učenja svih potrebnih materijala za izradu igre imali su tečaji s Teachable-a [4], Udemy-a [5] te brojni Youtube videi.

Kao sustav za verzioniranje koda koristio se GitHub server do određene granice, a lokalno nakon početka uporabe modela (zbog veličine memorijskog prostora) te GitKraken, vrlo intuitivan Git GUI.

# 3. Razrada teme

Prije same izrade videoigre, slično kao i kod drugih pothvata oko novih projekata, potrebno je definirati konačan izgled naše videoigre te korake kroz koje ćemo prolaziti kako bi taj cilj realizirali. Kroz faze razvoja moguće je da se pojedini detalji izmjene, ali konačan cilj bi trebao ostati isti kako bi se izbjeglo nepotrebno implementiranje funkcionalnosti koje se neće koristiti i time trošili resurse poput vremena i novaca.

Tehnički aspekt videoigre bit će s velikim fokusom na OOP, odnosno SOLID principe. Poštujući SOLID principe ostvariti ćemo vrlo stabilnu videoigru s neograničenim mogućnostima proširivanja i dodavanja novih funkcionalnosti.

## 3.1. Inspiracija za videoigru

Kao najveća inspiracija za izradu videoigre koristio se „Souls-like“ žanr video-igre, koji je nastao nakon rasta popularnosti From Softwareovih Souls igara:

- Demon's Souls
- Dark Souls
- Dark Souls 2
- Dark Souls 3
- Bloodborne
- Sekiro
- Elden Ring

Elementi inspiracije iz „Souls-like“ žanra, koji će biti prisutni u videoigri ovog diplomskog rada su brza akcijska borba, koja će omogućavati uporabu više vrste oklopa i oružja svaka sa svojim stilom borbe, više vrsta neprijatelja i glavnih neprijatelja te unikatne razine.

## 3.2. Vizija i cilj videoigre

Prvi prvom pokretanju videoigre igrač će započeti s jednom vrstom oružja, a to je mač u jednoj ruci i štit u drugoj. Igrač će imati dostupna 3 napitka s kojima će si moći vratiti živote, napitci se ponovno pune ukoliko igrač umre te se ponovno oživi ili ako pređe na sljedeću razinu.

Prvi sukob s neprijateljem biti će već nakon prvih par koraka u zoni u kojoj se nalazi. Nakon što igrač porazi sve neprijatelje na kraju zone čekati će ga glavni neprijatelj koji nakon svoje smrti stvara portal kojim će igrač preći na sljedeću razinu.

Sljedeća zona predstavljat će veću količinu neprijatelja te nove vrste oružja koje će igrač moći koristiti kako bi nadvladao prepreke, ukoliko se igrač odluči istraživati van predefiniranog puta otkriti će oklop koji će moći pokupiti i time si olakšati ostatak igre.

Nakon pobjede glavnog neprijatelja druge zone igrač prelazi na konačnu treću zonu koja će služiti kao predstavljanje više vrsta oklopa, oružja i neprijatelja jednostavno i brzo stvorenih zahvaljujući poštivanju SOLID principa.

Sam proces videoigre bit će težak i zahtjevan, ali će kroz više iteracija igranja igrač usavršit svoju kontrolu lika te svladati prepreke koje su mu prezentirane.

## 3.3. Funkcionalnosti videoigre

Funkcionalnosti videoigre koje su implementirane u sklopu ovog diplomskog rada su:

1. Osposobljen sustav za verzioniranje koda
2. Kontrola glavnog lika
3. Uzimanje predmeta s poda i opremanje glavnog lika
4. Osposobljavanje glavnih mehanika
5. Osposobljavanje neprijatelja i njegove umjetne inteligencije
6. Kreiranje vanjskog svijeta za videoigru
7. „Targeting system“
8. Interakcija igrača i neprijatelja sa svijetom
9. Odabir modela glavnog lika/neprijatelja i ostalih predmeta
10. Implementacija zvuka i animacije za sve vrste kretanja, borbe
11. Glavni izbornik

Implementacijom ovih funkcionalnosti ostvariti će se pravi osjećaj videoigre te ćemo omogućiti lako nadovezivanje novih funkcionalnosti na već postojeće.

**Napomena:** proces opisan u diplomskom radu ne prati točan redoslijed kod kreiranja videoigre iz razloga što je pri izradi videoigre bilo potrebno više puta vraćati se u određene nacrte i nadopunjavati ih novim funkcionalnostima.

Kao prvi korak izrade videoigre započinjemo s kreiranjem novog „Third person“ projekta u programskom alatu Unreal Engine 5.

### 3.3.1.Osposobljen sustav za verzioniranje koda

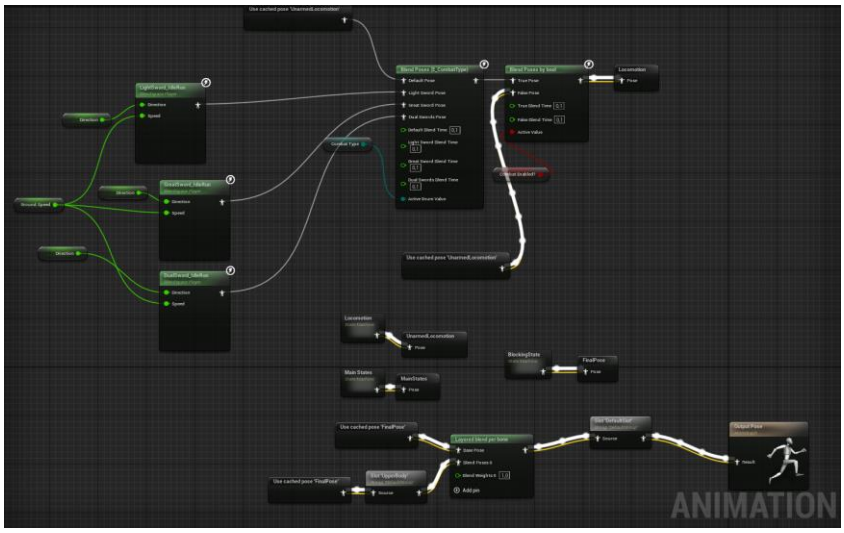
Jedna od najboljih praksa kod izrade projekata je projekt staviti na sustav za verzioniranje koda, kako bi imali lakšu kontrolu nad kodom te implementacijom zasebnih funkcionalnosti, testiranju tih funkcionalnosti i spajanju funkcionalnosti u glavni program. Time si omogućujemo efikasno održavanje sigurnosnih kopija s detaljnim pregledom izmjena u samom kodu. Ukoliko budemo nezadovoljni s donesenim izmjenama, uporabom sustava za verzioniranje možemo vrlo lako izbrisati donesene promjene.

Unutar programa pomoću „Source Control“ gumba na jednostavan način spajamo projekt na GitHub repozitorij te uporabom GitKrakena radimo commitove i pushove na naše Git grane.

### 3.3.2.Kontrola glavnog lika

Unreal Engine 5 pri otvaranju novog „Third person“ projekta već ima predefinirane kontrole za kretanje glavnog lika kao što su kretanje, skakanje i kontrola kamere preko „Enhanced Input Local Player Subsystem“ te osnovne funkcionalnosti kretanja imamo implementirane na samom početku u sklopu nacrta, kojeg smo nazvali, BP_CombatPlayerCharacter. Osim kontrole lika imamo predefinirani „Skeletal Mesh“ odnosno zadani model lika, koji će se koristiti sve dok ne implementiramo sve funkcionalnosti koje smo zadali, željeni model lika igrača i neprijatelja budemo primjenjivali u zadnjim koracima izrade videoigre.

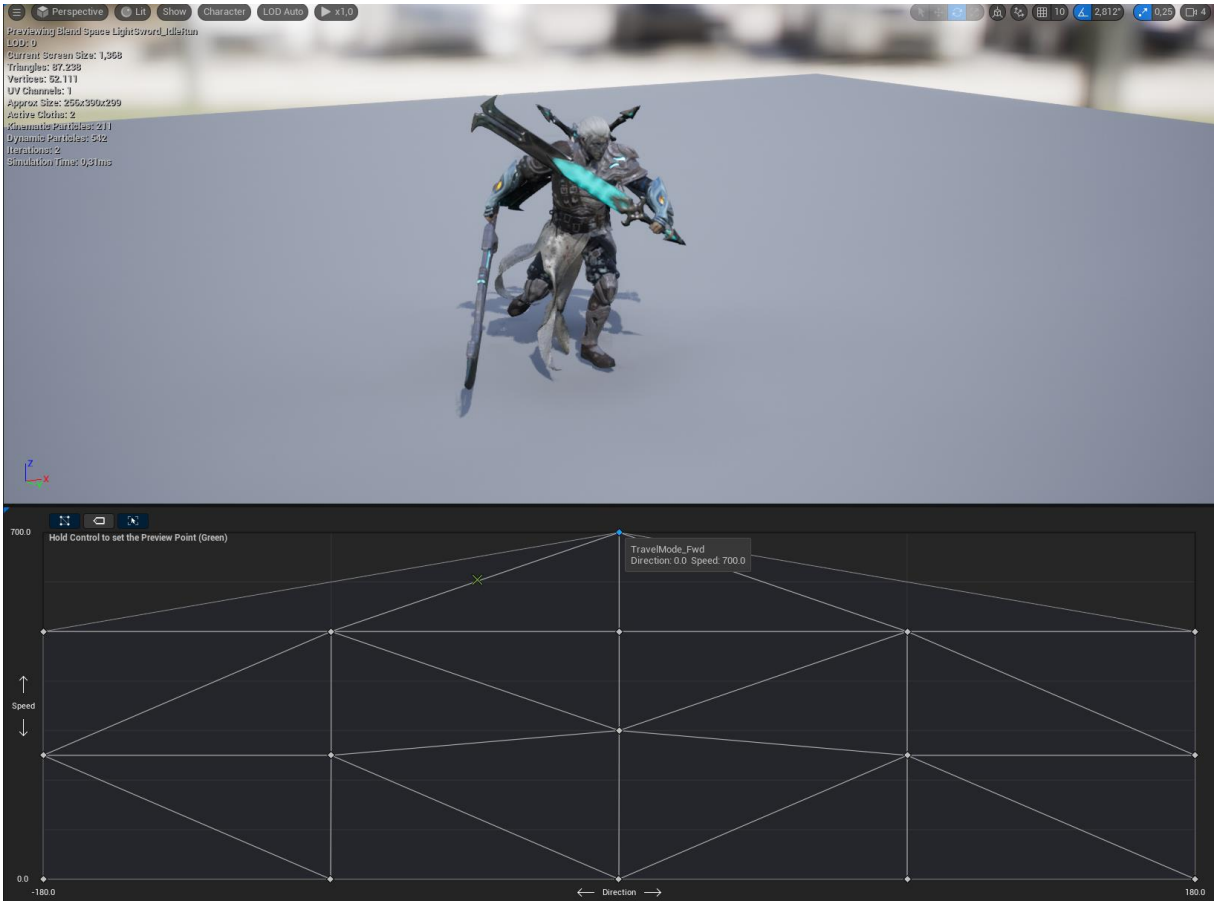
Kako bi upotpunili kontrolu glavnog lika potrebno je postaviti pravila u „Locomotion“ animacijskom nacrtu.




Slika 1: Locomotion

Unutar samog Locomotiona osposobljavamo kretanje likova te pomoću „BlendSpace“ postavljamo način kretanja ovisno o brzini kojom se kreću.

Kako bi pronašli modele i animacije koje želimo koristiti u svojoj videoigri Epic Games omogućuje nam pronalaženje raznoraznih modela i animacija preko Unreal Engine Marketplace stranice. Svi modeli koje koristim u videoigri su besplatni ili su preuzeti za vrijeme dok su se dijelili za besplatno.

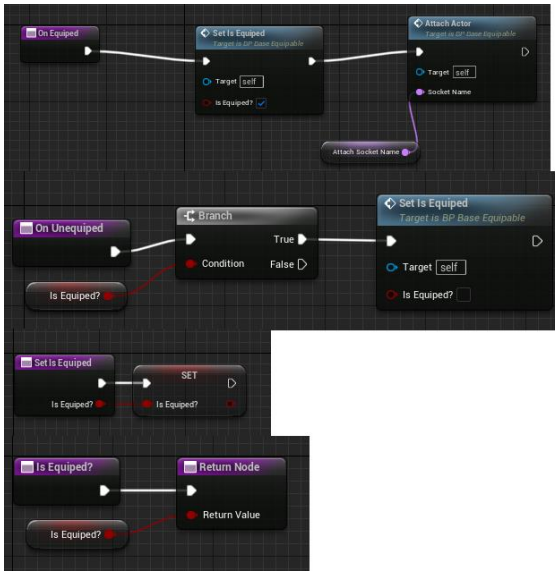



Slika 2: BlendSpace za Light Sword

Kako bi se igrač kretao što fluidnije potrebno je razne vrste animacija upotrijebiti te definirati vektore kretanja kako bi se napravile tranzicije u animaciji kretanja.

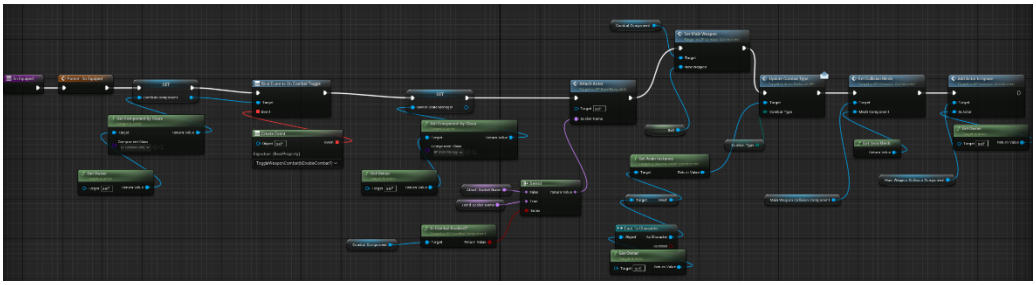
### 3.3.3.Uzimanje predmeta s poda i opremanje glavnog lika

Kako bi uspostavili odnos BP_CombatPlayerCharacter sa ostalim predmetima potrebno je prvo te predmete stvoriti. Budući da smo definirali kako želimo da igrač može pokupiti više vrsta predmeta poput oružja, oklopa, napitaka i slično. Stvoriti ćemo novi nacrt tip „Actor“, koji će nam služiti upravo kao temelj za sve druge vrste predmeta kako bi omogućili interakciju igrača sa svim drugim predmetima. Nacrt koji stvaramo u ovu svrhu zove se BP_BaseEquipable te unutar njega definiramo šest funkcija: GetItemMesh, AttachActor, OnEquiped, OnUnequiped, SetIsEquiped, IsEquiped? kako bi poštivali pravilo enkapsulacije i dohvatili i postavili privatne varijable deklarirane u sklopu nacrta. Pravilo enkapsulacije tvrdi da „Enkapsulacija je način ograničavanja izravnog pristupa nekim komponentama objekta, tako da korisnici ne mogu pristupiti vrijednostima stanja za sve varijable određenog objekta. Enkapsulacija se može koristiti za skrivanje i podatkovnih članova i podatkovnih funkcija ili metoda povezanih s instanciranom klasom ili objektom“ [7].




Slika 3: BP_BaseEquipable

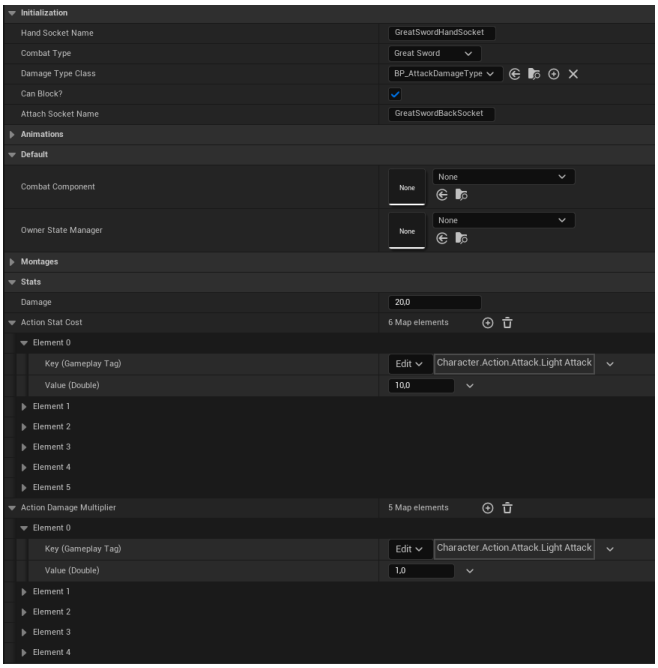
Nadalje, kako bi poštivali „Open Closed Principle“ radimo child class od „BP_BaseEquipable“ te ga nazivamo „BP_BaseWeapon“ koji ćemo koristiti za stavljanje različitih vrsta oružja u određeno mjesto na našem glavnom liku. Funkcija zadužena za dohvaćanje oružja koje je igrač pokupio je „OnEquiped“ te pomoću nje provjeravamo vrstu oružja, u koji socket se sprema, postavljamo koliziju predmeta te stanje u kojem se igrač nalazi.



Slika 4: OnEquiped (BP_BaseWeapon)

Kreiranjem enumeratora E_CombatType definiramo 3 vrste oružja: Light Sword, Great Sword, Dual Swords. Time dalje omogućujemo različito ponašanje lika s obzirom na oružje koje je odabrao.

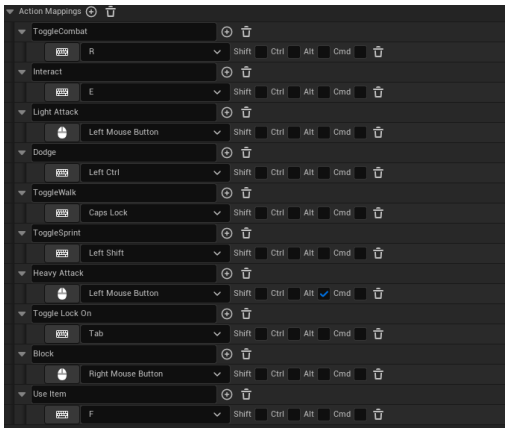
Kako bi napravili različite vrste oružja radimo child class od „BP_BaseWeapon“ za svako oružje koje želimo imati, u slučaju videoigre ovog diplomskog rada to su: BP_SwordAndShield, BP_GreatSword i BP_DualToughSwords. Unutar tih nacrta pomoću naslijeđenih varijabli jednostavno dodjeljujemo socket u koji se sprema, koja je vrsta oružja, animacije za sve vrste napada, štetu koju nanosi i računanje jačinu štete s obzirom na napad koji izvodi te koliko je potrebno resursa da igrač odigra određeni napad.




Slika 5: Postavljanje BP_Greatsword

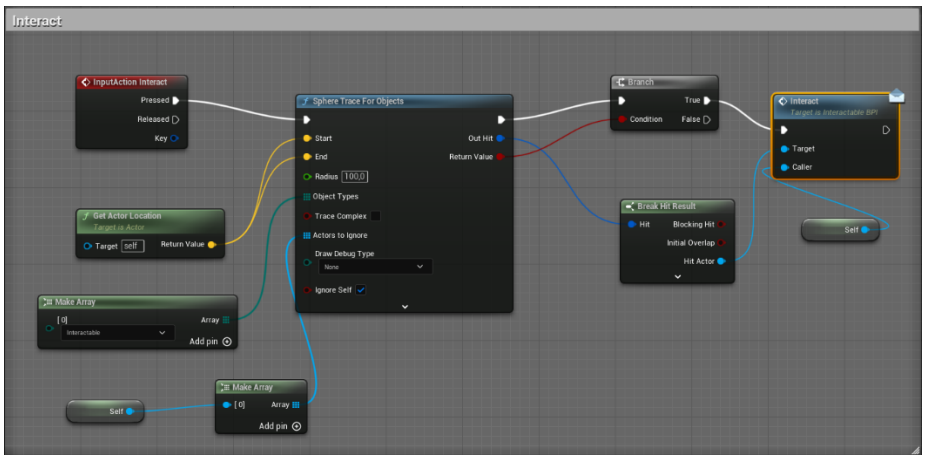
Sljedeće što je potrebno implementirati je funkcionalnost preuzimanja predmeta, a to ćemo postići s nacrtom „BP_PickupActor“ te ćemo napraviti novi interface „Interactable_BPI“ s funkcijom „Interact“ koju ćemo dalje pozivati u „BP_PickupActor“ i „BP_CombatPlayerCharacter“ kako bi ovisno o predmetu koji stupa u interakciju s likom proslijedili predmet koji je odabran.

Kako bi uopće postavili gumb za „Interakciju“ potrebno je u „Project Settings“ otići u sekciju „Input“ te postaviti željene inpute. Input, odnosno kontrole korištene u ovoj videoigri su sljedeće:



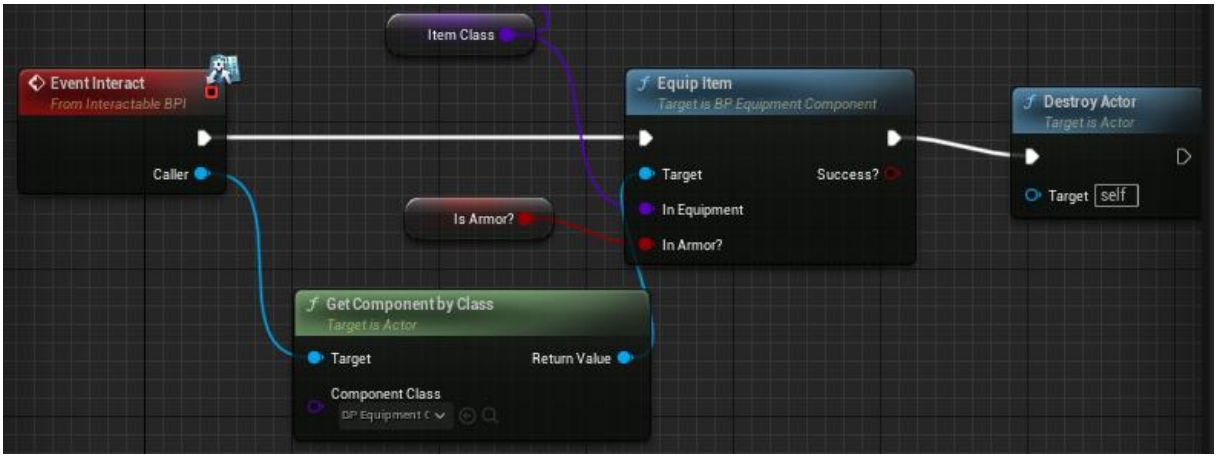

Slika 6: Inputi

Zatim u „BP_CombatPlayerCharacteru“ dodajemo novi događaj zvat „InputAction Interact“ kako bi definirali logiku koja se poziva pri pritisku tipke za „Interact“.



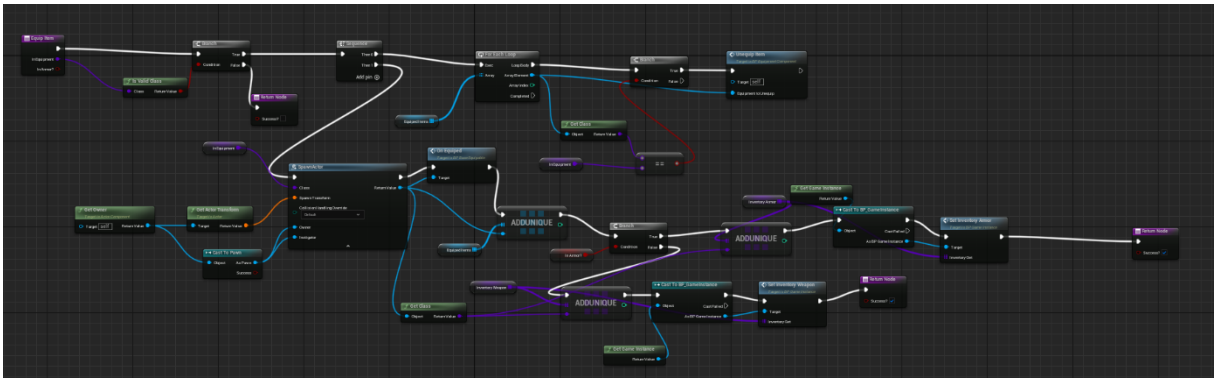

Slika 7: InputAction Interact

Ukoliko pronađemo putem ugradbene funkcije „Sphere Trace For Objects“ predmete koji su „Interactable“ pozivamo za svakog od njih funkciju „Interact“ koja se zahvaljujući interfaceu poziva i u „BP_PickupActor“.




Slika 8: EventInteract

EventInteract dalje poziva vrlo kompleksnu funkciju „Equip Item“, koja obavezno provjerava je li klasa proslijeđena validna te ukoliko je provjerava imamo li već na igraču tu određenu vrstu oružja obučenu, ako da poziva „Unequip Item“, zatim stvara oružje na liku pomoću „On Equiped“ funkcije i sprema predmet koji je pokupio u varijablu „Inventory Weapon“ ili „Inventory Armor“. Te dvije varijable spremamo u GameInstance, a ne u samog „BP_CombatPlayerCharacter“ iz razloga što bi te varijable izgubili u slučaju kada nam igrač umre odnosno kada koristimo „DestroyActor“ funkciju.



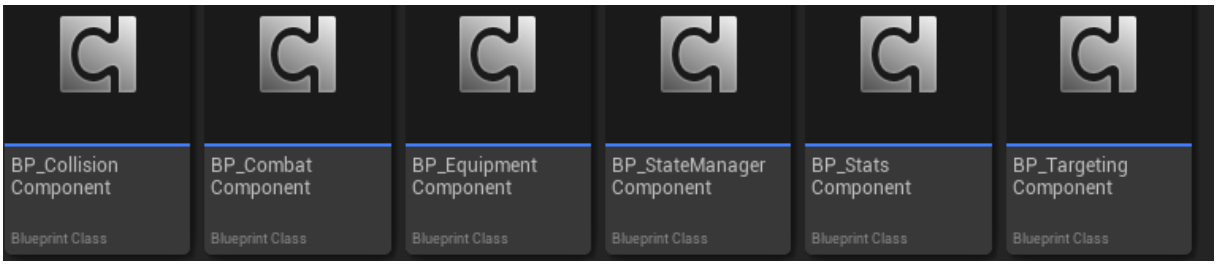
Slika 9: Equip Item

## 3.3.4.Osposobljavanje glavnih mehanika

Sljedeća funkcionalnost koju ćemo pokriti je pokretanje napada, uzrokovanje štete te ažuriranje stanja glavnog lika.

### 3.3.4.1. Komponente

Kako bi omogućili pokretanje napada našeg glavnog lika i ažuriranje njegovog stanja vrlo je dobro razdvojiti te funkcionalnosti na komponente stoga će u ovom radu koristiti se nekoliko komponenata kako bi te funkcionalnosti kasnije mogli prenijeti i na neprijatelje.




Slika 10: Komponente

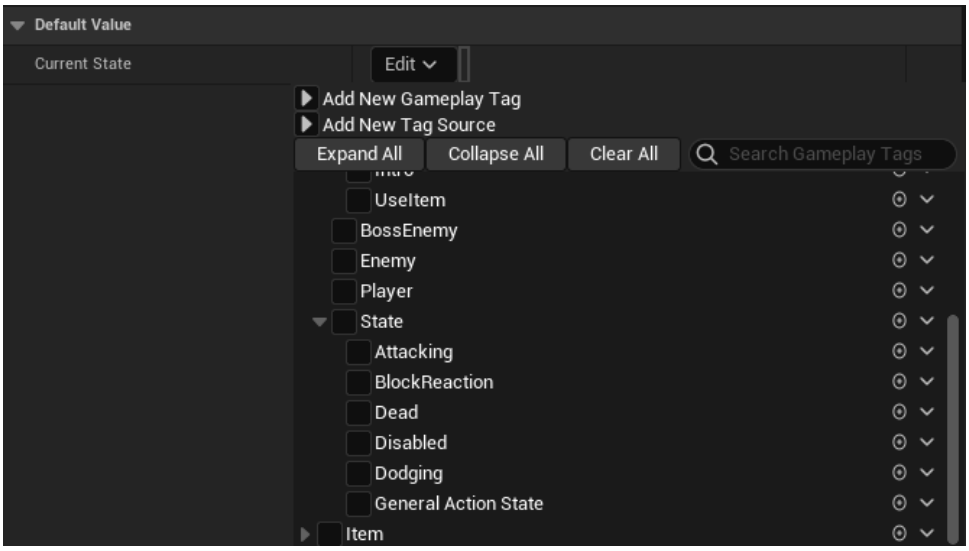
Komponente nam služe kao direktno nasljeđivanje funkcija i varijabla iz komponenta u nacrt „Actor“ klase gdje se pozovu, u trenutnom primjeru kod glavnog lika „BP_PlayerCombatCharacter“.

Komponente implementirane su:

- BP_CollisionComponent – postavljaju koliziju, koje su „Actor“ klase pogođene, koga ignorirati
- BP_CombatComponent – postavlja glavno oružje, resetira napad, provjerava blokiranje štete
- BP_EquipmentComponent – oblači i miče oružje, štitove i napitke
- BP_StateManagerComponent – upravlja stanjima „Actor“ klase
- BP_StatsComponent – postavlja životne bodove (*eng. Health Points)*, izdržljivost (*eng. Stamina*) te ostalih varijabli definirani u enumeratoru „E_Stats“
- BP_TargetingComponent – pronalazi metu, rotira lika u smjeru mete

### 3.3.4.2. Stanja glavnog lika

„BP_StateManagerComponent“ nam služi kao direktan pristup informacijama o stanju u kojem se lik nalazi. Npr. stanje napada, stanje kotrljanja, stanje konzumiranja itd, preko vrlo moćne varijable zvane „Gameplay Tag“, koja nam omogućuje vrlo jednostavno postavljanje i dohvaćanje stanja, vrste napada i mnogo drugih.




Slika 11: Gameplay Tag

Funkcionalnosti pomoću kojih dobivamo i postavljamo stanja su:

- SetState
- GetCurrentState
- ResetState
- IsCurrentStateEqualToAny?
- Set Current Action
- GetCurrentAction
- IsCurrentActionEqualToAny?

Pomoću ovih funkcija održavamo pravilo enkapsulacije i nasljeđivanja varijabli po pravilima OOP (Objektno Orijentiranog Programiranja).

### 3.3.4.3. Heads-up display

Za postavljanje UI koristimo vrlo moćan nacrt, koji nam nudi Unreal Engine, a to je Widget. Widget postavljamo po želji te ga vrlo jednostavno možemo ažurirati u stvarnom vremenu po potrebi.

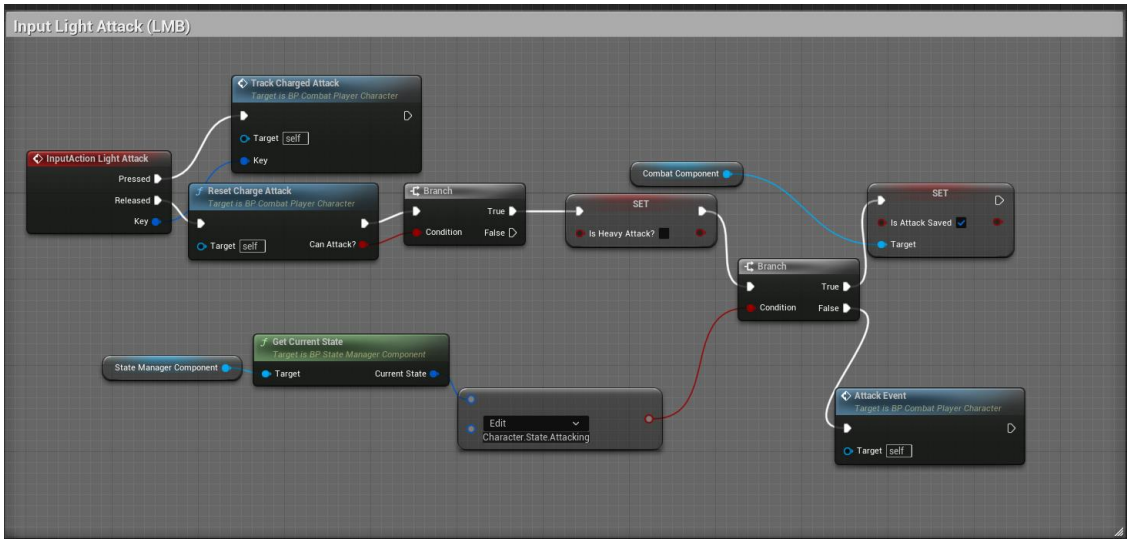
Slika 12: Player HUD

Player HUD nam omogućuje prikaz vlastitih životnih bodova, izdržljivosti, broj napitaka te životni bodovi neprijatelja (običnih neprijatelja iznad njihovog modela, glavnih neprijatelja kao sa slike 12.).

### 3.3.4.4. Pokretanje napada

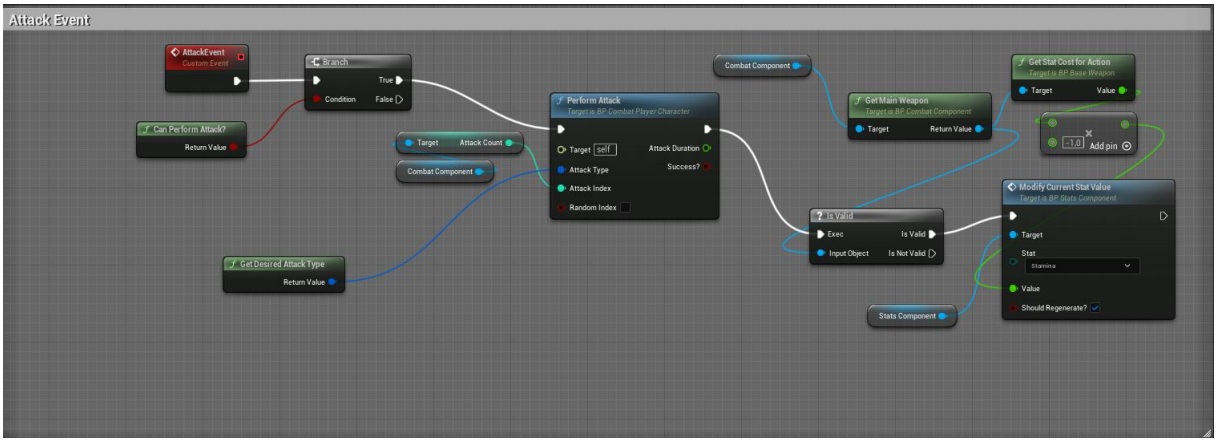
Kako smo već definirali sve potrebne inpute u „Project Settings“ sada je potrebno razraditi logiku koja se poziva pri odabiru određenih inputa, tako ćemo morati osposobiti što se desi kada igrač pokrene napad npr. lijevim klikom miša te ukoliko se odluči kotrljati s CTRL gumbom i ostali.

Napad lijevim klikom miša želimo nadovezati na lanac napada te ukoliko igrač odluči držati lijevi klik miša poziva se „Charged Attack“ vrsta napada.



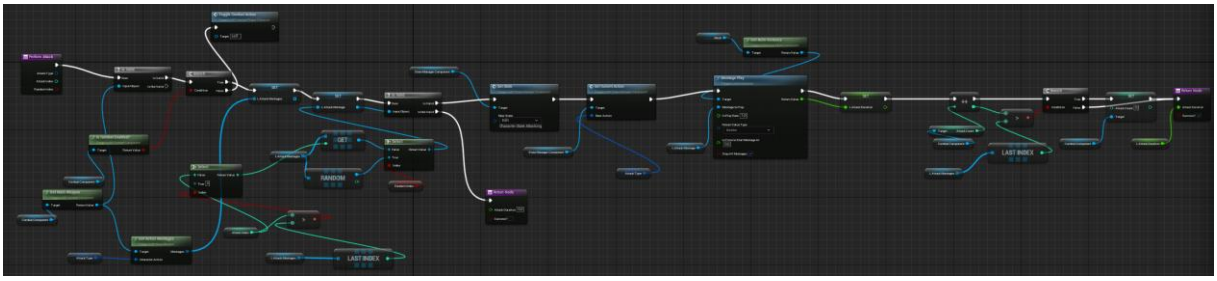

Slika 13: InputAction Light Attack

U samom inputu za „Light Attack“ potrebno je odmah provjeriti drži li igrač lijevu tipku preko događaja „Track Charged Attack“ te ukoliko nije „Charged Attack“ pozivamo događaj „Attack Event“.



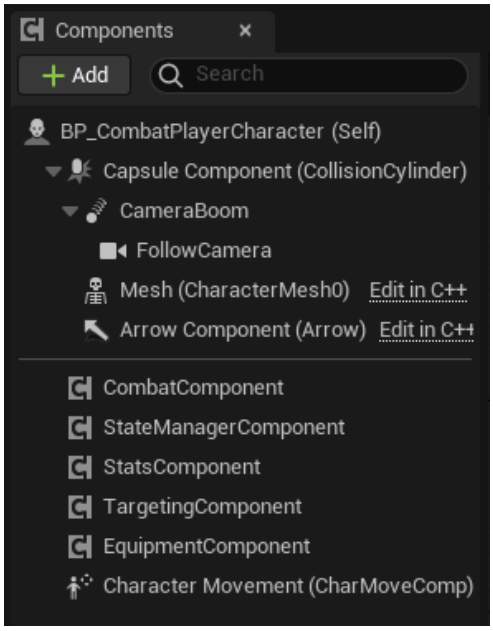

Slika 14: Attack Event

„Attack Event“ je događaj koji se poziva na kraju svakog napada te on provodi napad preko „Perform Attack“ funkcije i ažurira stanje igrača preko „Modify Current Stat Value“ funkcije.



Slika 15: Perform Attack

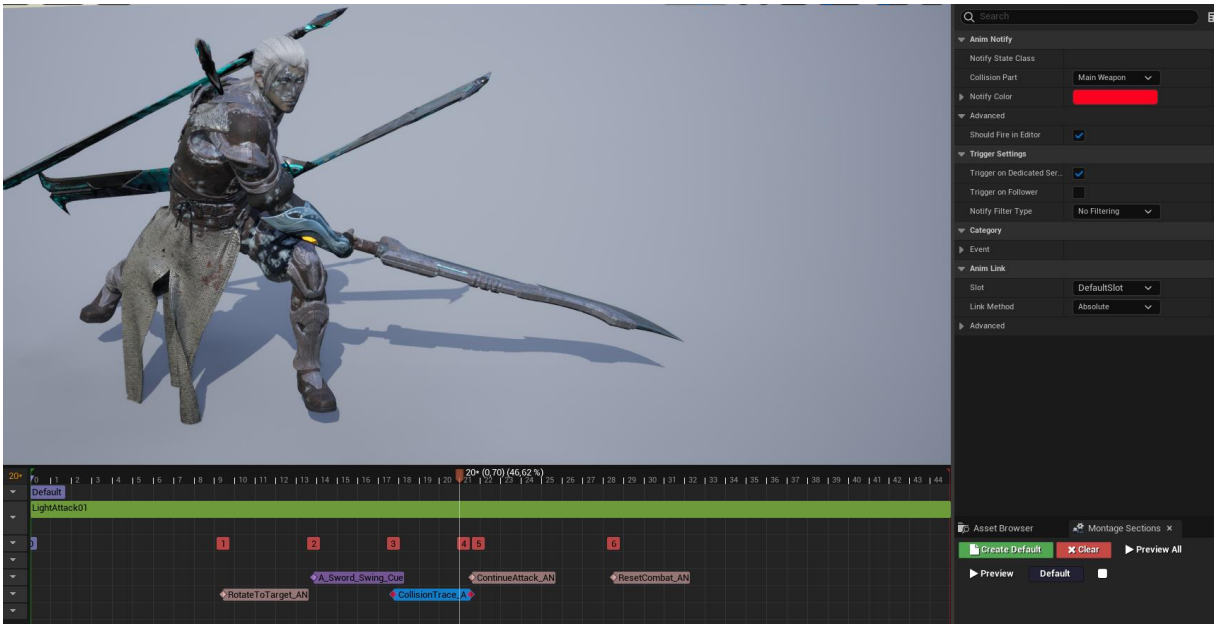
„Perform Attack“ funkcija provjerava s obzirom na „Main Weapon“ koji dobivamo iz „BP_CombatComponent“ komponente, koja je slične funkcionalnosti kao „BP_BaseEquipable“ te nam služi za komunikaciju igrača s funkcijama i varijablama u „BP_CombatComponent“ komponenti.




Slika 16: Komponente u "BP_CombatPlayerCharacter"

„Perform Attack“ funkcija nadalje dohvaća sve animacije za napade te postavlja stanje glavnog lika u „Attacking“ pomoću našeg „State Manager Component“, nakon toga pokreće animaciju pomoću „Montage Play“ funkcije.

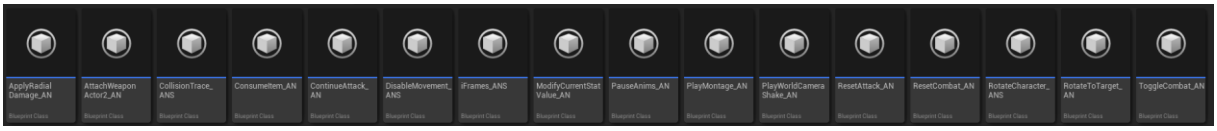
### 3.3.4.5. Animation Notify




Slika 17: Animacija Light Attack 01

Uslijed animacije pomoću „Animation Notify“ imamo opciju u određenom trenutku animacije okinuti funkcije koje definiramo unutar samih „Animation Notify“.

Implementirani Animation Notify:

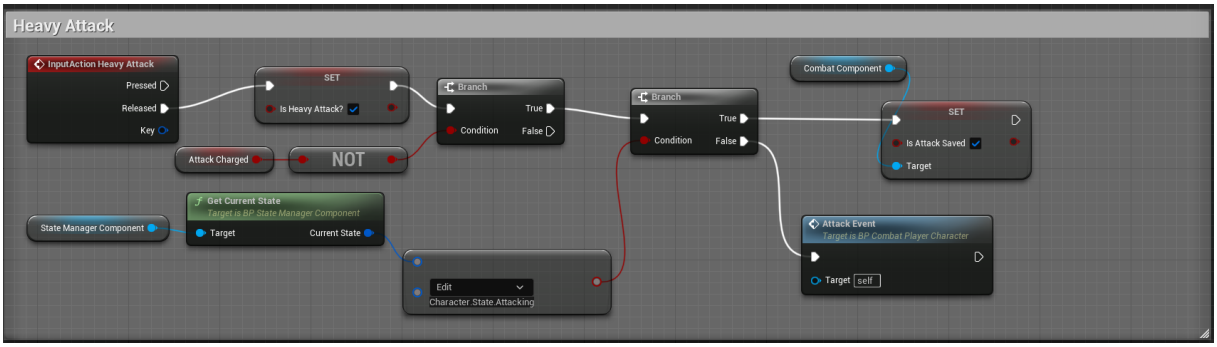



Slika 18: Animation Notifies

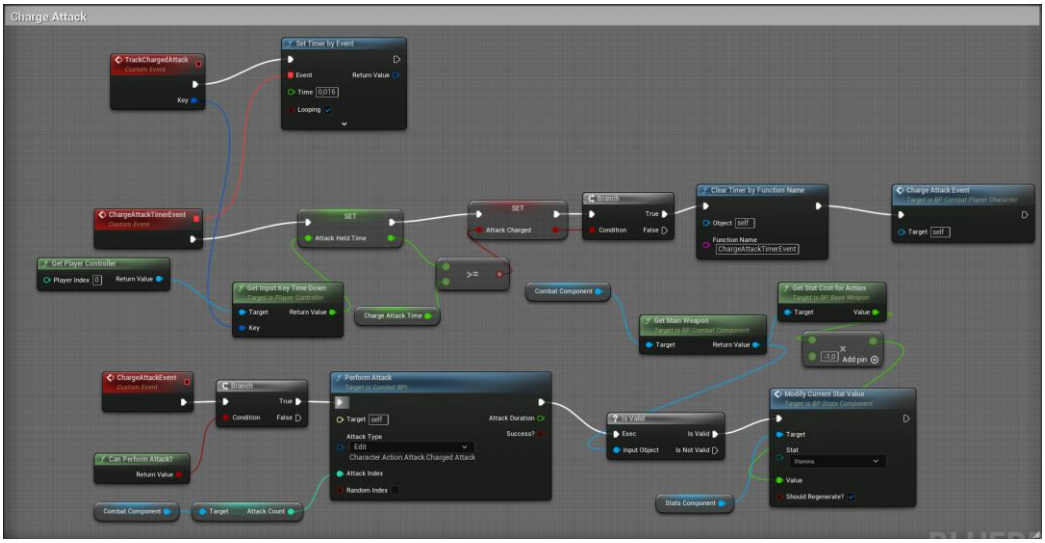
Animation Notify koji nam je potreban za interakciju igrača s neprijateljem je „CollisionTrace_AN“, upravo on aktivira koliziju oružja te okida daljnje događaje koji neprijatelju nanose štetu.

### 3.3.4.6. Ostale vrste napada

Osim Light Attack igrač ima opciju napraviti dodatne vrste napada poput:

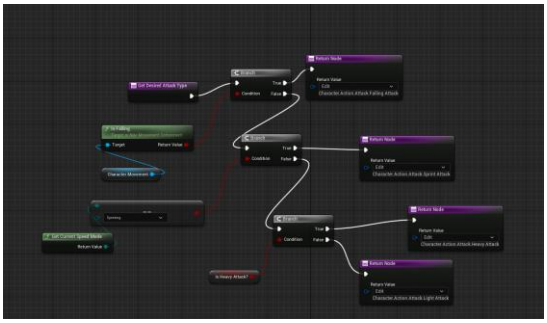



Slika 19: Heavy Attack




Slika 20: Charge Attack

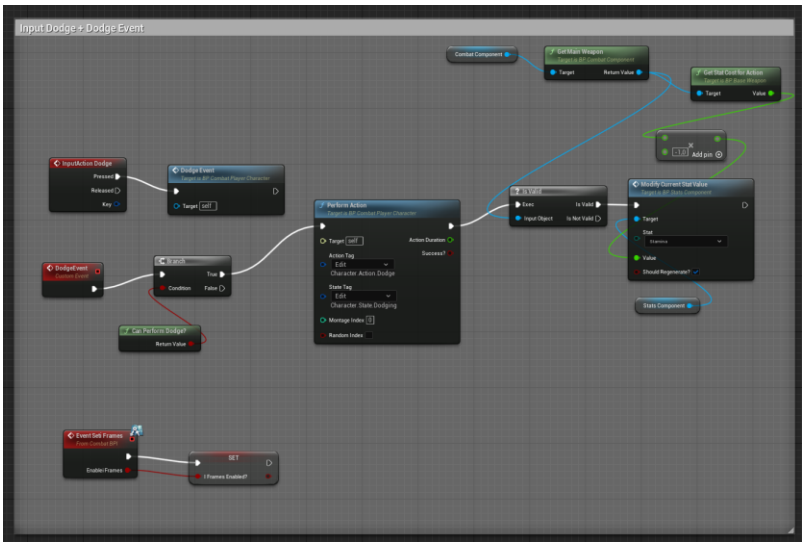
Te ostale vrste napada poput „Falling Attack“ i „Sprinting Attack“.



Slika 21: Desired Attack Type

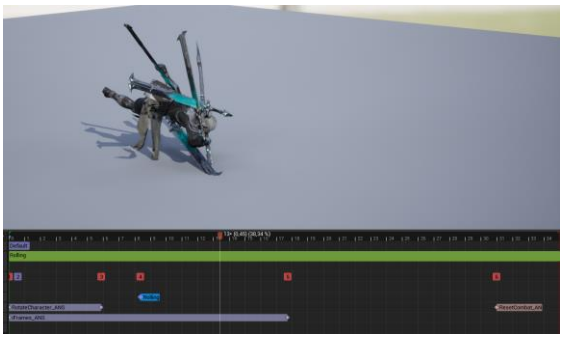
### 3.3.4.7. Kotrljanje

Kotrljanje jedna je od najistaknutijih funkcionalnosti kod „Souls-like“ videoigara te ona služi da se igrač kotrlja i uslijed animacije za kotrljanje ima određene trenutke neranjivosti kako bi izbjegao štetu.




Slika 22: InputAction Dodge

Ukoliko igrač pritisne tipku za „InputAction Dodge“ poziva se događaj „DodgeEvent“ te glavni lik započinje animaciju kotrljanja i oduzima određene bodove izdržljivosti od glavnog lika.

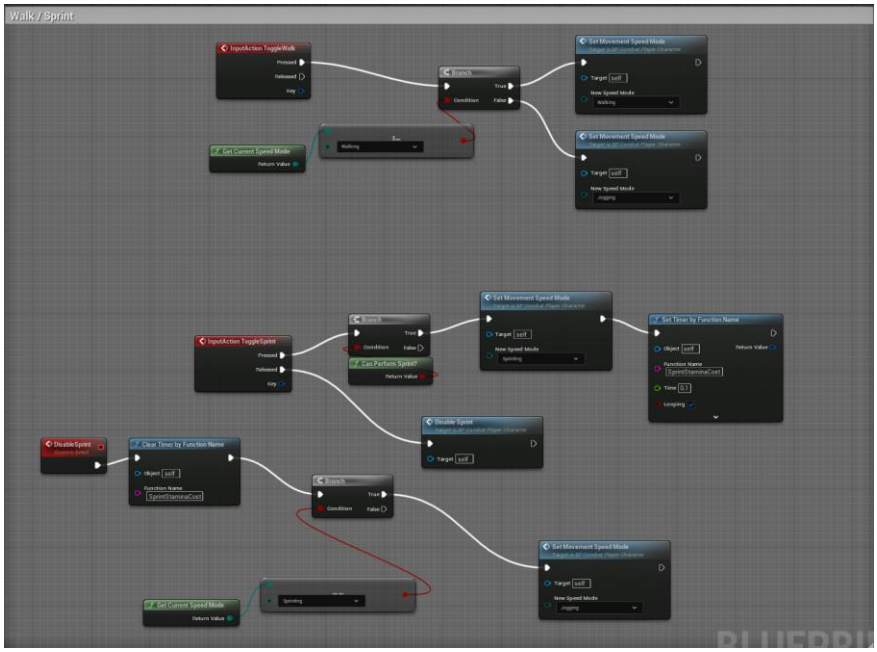


Slika 23: Animacija kotrljanja

Kao prethodne animacije napada, vidimo da kotrljanje poziva u određenim trenutcima „Animation Notify“, a to su „RotateCharacter_ANS“ koji nam omogućuje da rotiramo lika u drugim smjerovima te „iFrames_ANS“ koji nam za period svog trajanja daje trenutke neranjivosti.

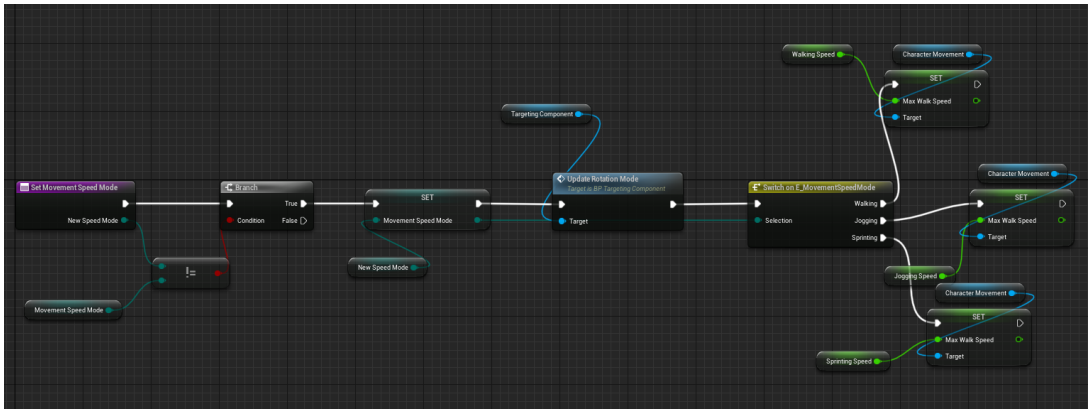
### 3.3.4.8. Trčanje

Kako bi napravili tranziciju iz hodanja u trčanje glavnog lika koristimo ponovo događaj koji se okida pritiskom gumba, a to je „InputAction ToggleWalk“ događaj.

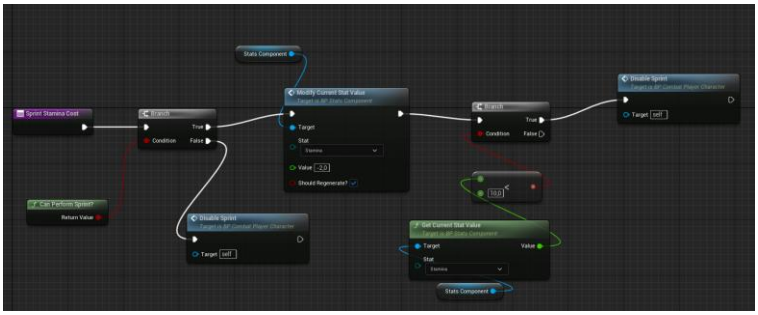



Slika 24: InputAction ToggleWalk

Tim događajem igraču se postavlja brzina njegovog kretanja na „Sprinting Speed“, ali se ujedno i troši njegova izdržljivost pomoću funkcije „SprintStaminaCost“ koja se okida svake 0,1 sekunde. Zahvaljujući postavljenom „Locomotion“ animacijskom nacrtu automatski se ažurira animacija kretanja glavnog lika ovisno o njegovoj brzini.



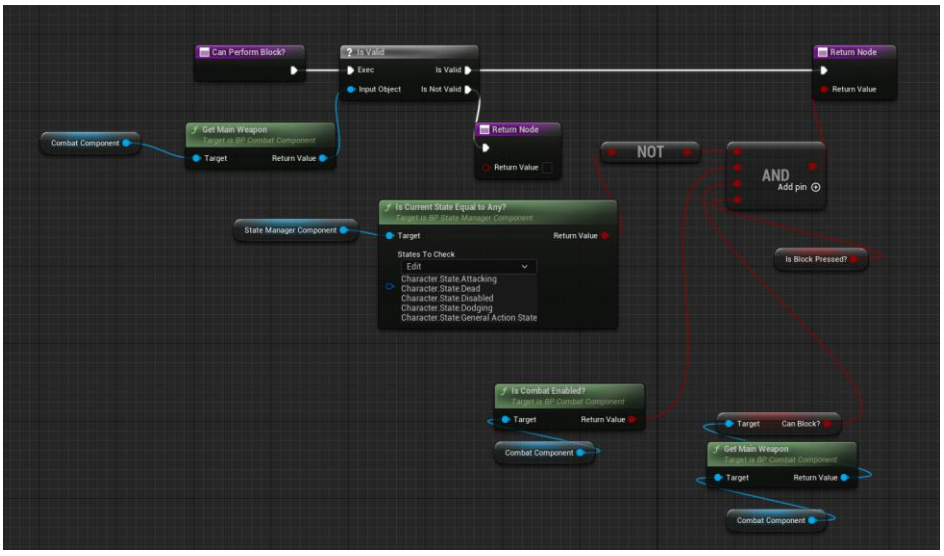
Slika 25: Set Movement Speed Mode



Slika 26: Sprint Stamina Cost

### 3.3.4.9. Blokiranje štete

Blokiranje štete je implementirano tako da svako oružje ima opciju za blokiranje štete, neka oružja blokiraju više štete, nego ostala. Ukoliko ne želimo da oružje koje imamo može blokirati dovoljno je samo ostaviti varijablu „CanBlock?“ na false.

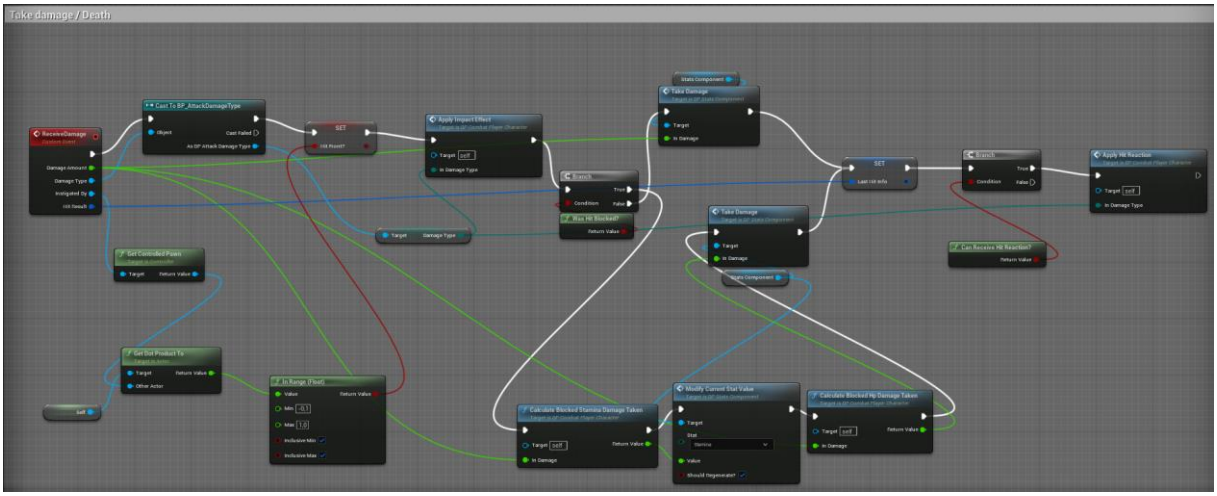



Slika 27: Can Perform Block?

Svako oružje ima svoju vrijednost varijable „ShieldDamage“, koju nasljeđuje od klase „BP_BaseWeapon“ te time određuje koliko štete uspijeva blokirati.

### 3.3.4.10. Primanje štete

Primanje štete je usko vezano uz blokiranje štete te prolazi kroz nekoliko provjera. Prvo dohvaćamo tip napada da znamo kakvu reakciju treba igrač odigrati, zatim određujemo s koje strane je igrač udaren kako bi proveli korektnu animaciju reakcije na napad. Nakon toga provjeravamo je li napad blokiran te ukoliko je kroz funkcije za kalkulaciju štete oduzimamo od izdržljivosti i životnih bodova ovisno o jačini napada te o jačini štita koji je blokirao štetu.

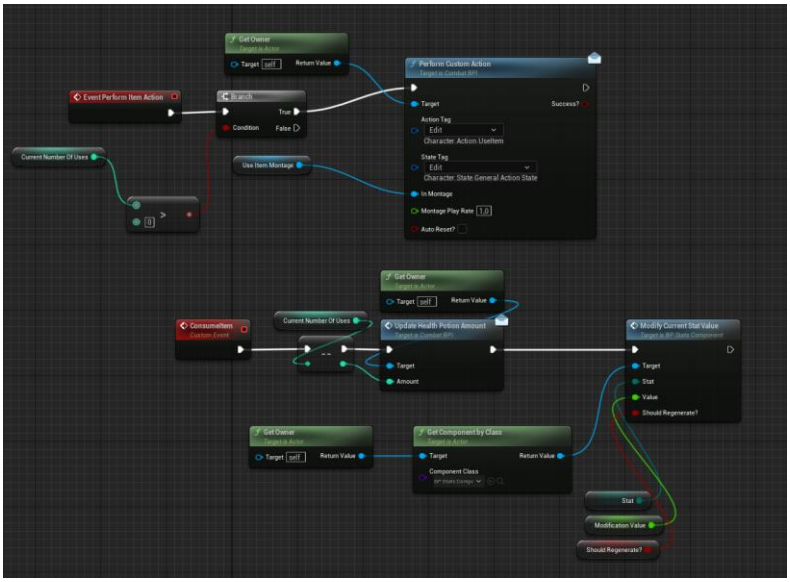



Slika 28: ReceiveDamage

### 3.3.4.11. Vračanje životnih bodova

Životne bodove glavnog lika vraćamo preko napitaka koje igrač može konzumirati. Napitci su ograničenih kapaciteta, ali se uvijek vrate na postavljenu vrijednost ukoliko igrač umre ili prijeđe na sljedeću razinu. Napitak je programiran kao „Actor“, koji nasljeđuje „BP_BaseEquipable“, slično kao oružja i oklopi.

Pozivanjem događaja za konzumiranje napitka provjeravamo može li igrač konzumirati napitak, postavljamo ga u željeno stanje te okidamo događaj ConsumeItem koji oduzima broj napitaka, ažurira „Player HUD“ te podiže igraču životne bodove.

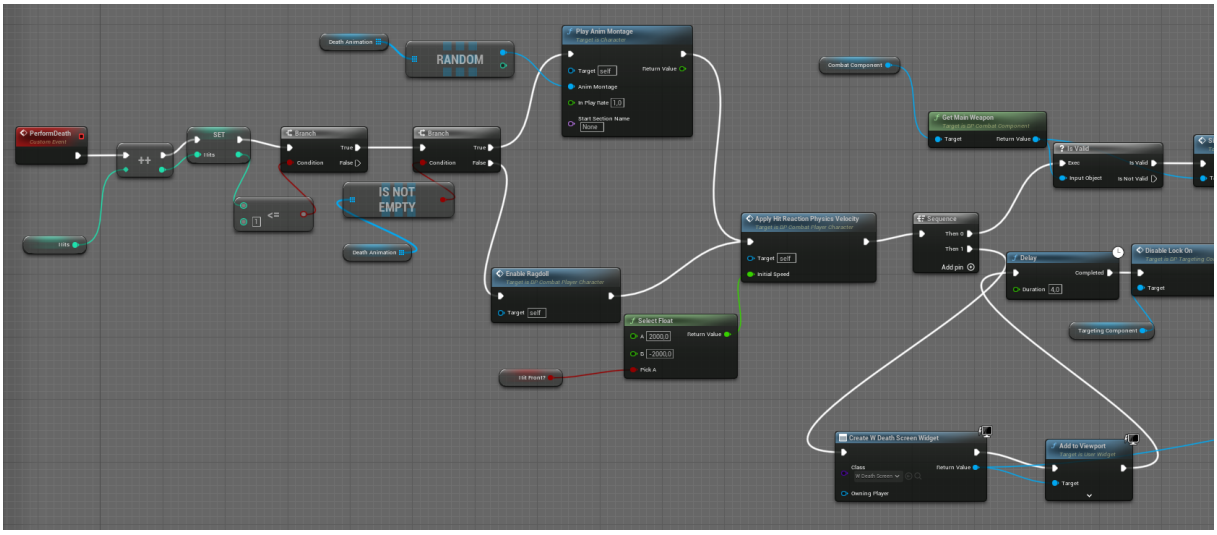


Slika 29: Event Perform Item Action

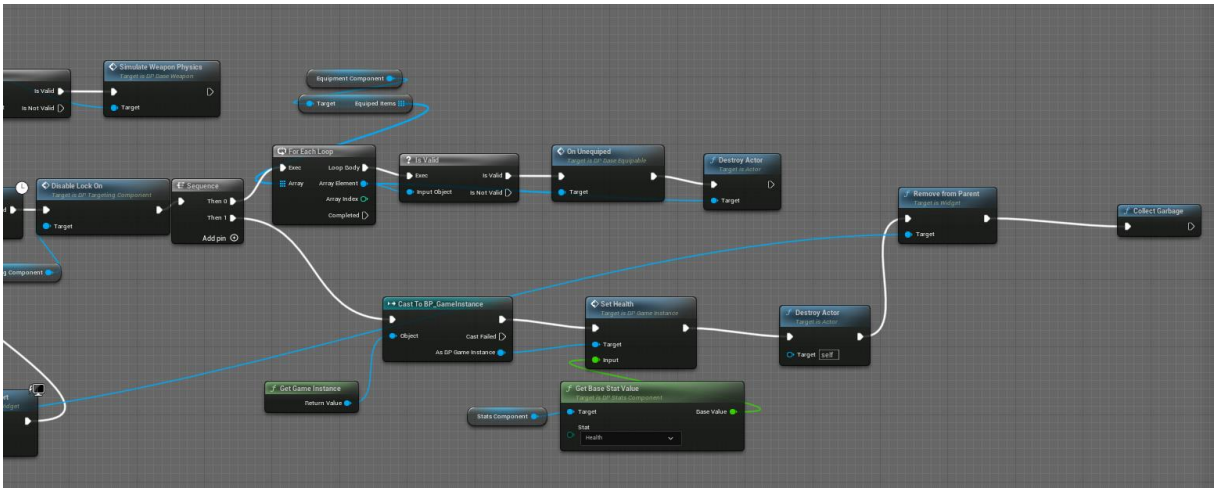
Događaj „Event Perform Item Action“, zahvaljujući poštivanju pravila OOP omogućuje da dodamo više vrsta napitaka, koja ne moraju nužno vraćati životne bodove.

### 3.3.4.12. Smrt

Smrt je proces koji se mora odviti kako kod glavnog lika tako i kod neprijatelja, a događa se kada životni bodovi padnu na nulu te se stanje lika promijeni u „Dead“ preko „BP_StateManagerComponent“. Kada se stanje lika postavi na „Dead“ potrebno je pronaći njegovu odgovarajuću animaciju za smrt te pokrenuti animaciju, ukoliko animacija nije prisutna, odnosno nije dodana, potrebno je uključiti „Ragdoll“ fiziku kako bi fizika riješila pad „Mesh“ komponente od lika, zatim ovisno o smjeru udarca lik nam treba odgovarajuće reagirati. Osim „Mesh“ komponente lika imamo još „Mesh“ komponente oružja i oklopa, oružje mora ispasti iz ruke te simulirati fiziku oružja nakon čega će se maknuti iz svijeta preko „Destroy Actor“ funkcije, ekran da je glavni lik umro se pojavljuje ukoliko je umro glavni lik. Nakon svega, uništavamo „Actor“ komponentu od lika.




Slika 30: Perform Death #1



Slika 31: Perform Death #2

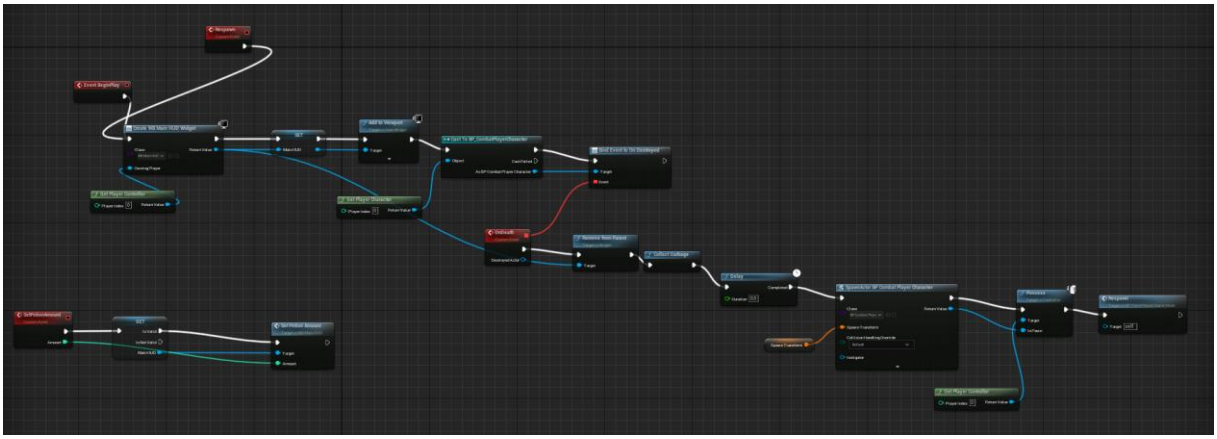
### 3.3.4.13. Oživljavanje glavnog lika

Kako bi oživili glavnog lika, a pritom sačuvali sve predmete koje je izgubio nakon „Destroy Actor“ funkcije, potrebno je spremiti ih u GameInstance.

GameInstance je objekt upravitelja visoke razine za instancu pokrenute igre. Pokreće se prilikom stvaranja igre i ne uništava dok se instanca igre ne ugasi [8]. U GameInstance spremamo varijable poput „InventoryWeapon“, „InventoryArmor“ i „Health“. Ukoliko se glavnom liku trajno poveća izdržljivost zbog nekog događaja, potrebno će biti i tu varijablu proslijediti u GameInstance kako ju ne bi izgubili pri ponovnom stvaranju lika.

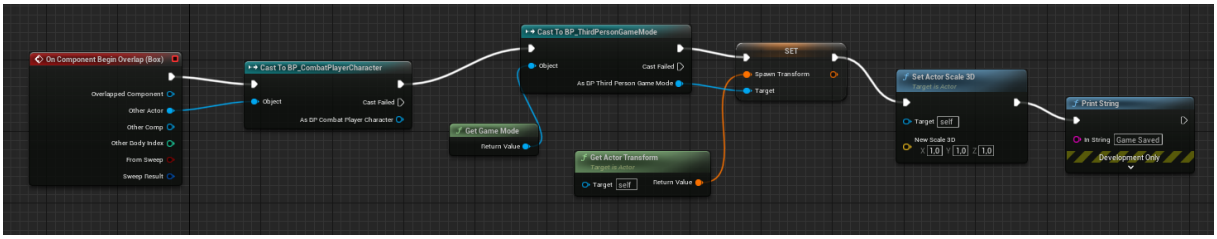
Nadalje kako bi se glavni lik oživio, potrebno je u nacrtu „BP_ThirdPersonGameMode“ razraditi logiku iza ponovnog postavljanja lika nakon što ga postavimo u svijet.

Pri prvom pokretanju igre postavljamo Main HUD Widget te postavljamo glavnog lika, ukoliko se desi događaj da je glavni lik odnosno „BP_CombatPlayerCharacter“ uništen, pozivamo događaj „OnDeath“ pomoću „Bind Event to On Destroyed“ kako bi maknuli sve Widgete te ponovno stvorili lika na zadnjem zabilježenom mjestu (Checkpoint) te pozivamo događaj „Respawn“.



Slika 32: BP_ThirdPersonGameMode

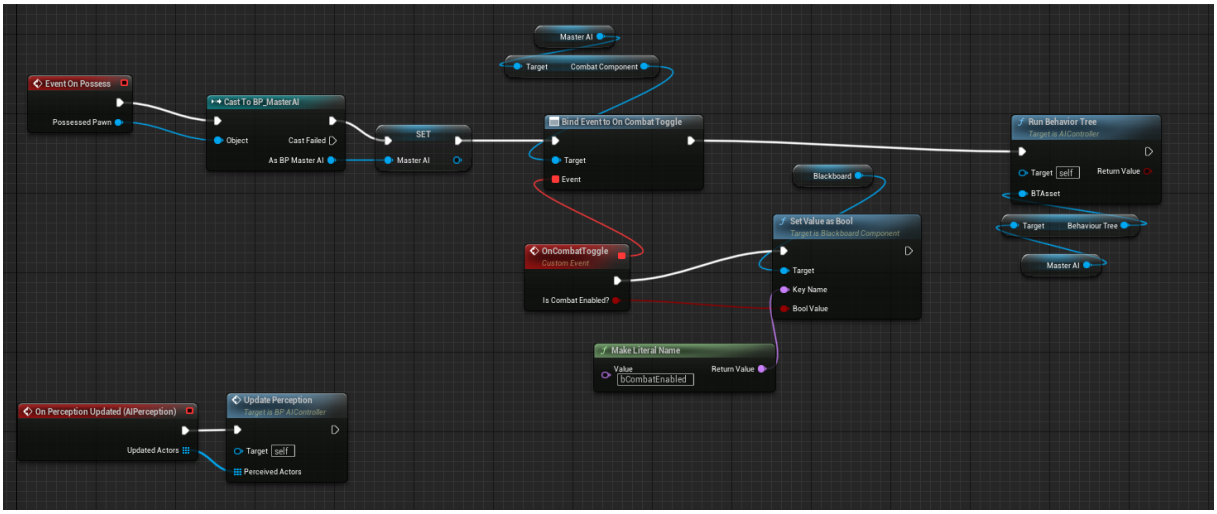
Kako bi zabilježili kontrolnu točku (*eng. Checkpoint*) koristimo „Actor“ komponentu koja nakon aktiviranja događaja „On Component Begin Overlap (Box)“ zabilježavamo lokaciju u varijablu „Spawn Transform“ kako bi u „BP_ThirdPersonGameMode“ u slučaju okidanja događaja „OnDeath“ imali zadnju lokaciju kontrolne točke kojoj je glavni lik pristupio.




Slika 33: BP_ RespawnPoint

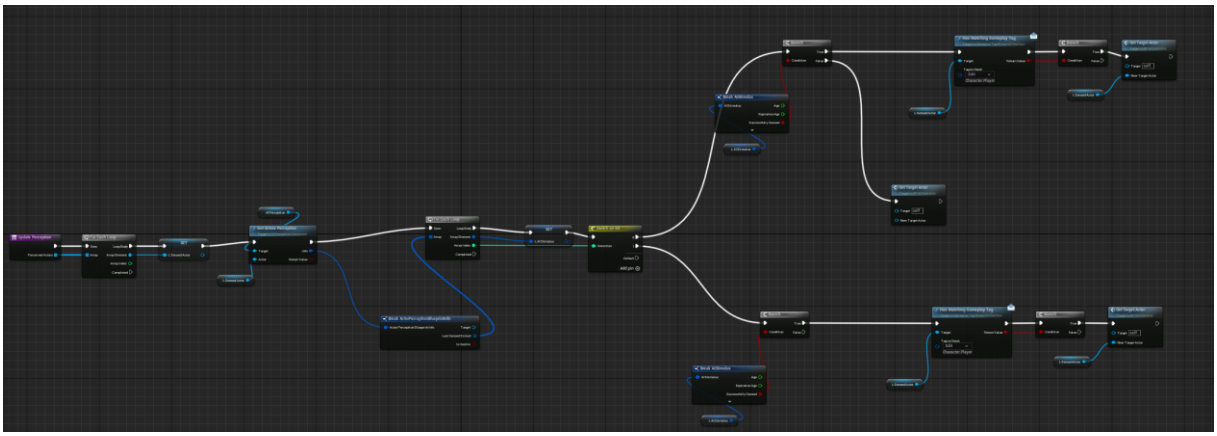
### 3.3.5.Osposobljavanje neprijatelja i njegove umjetne inteligencije

Neprijatelj će koristiti puno funkcionalnosti kao glavni lik, ali ih je dobro držati razdvojene stoga dupliciramo „BP_CombatPlayerCharacter“ i nazivamo ga „BP_MasterAI“ te izbacujemo sve funkcije i događaje koji neprijatelju neće trebati poput inputa, a ostavljamo događaje poput primanja štete. Sada imamo spreman nacrt „BP_MasterAI“, koji ima sposobnost razlikovanja vlastitih stanja, oružja i štitovi koja mu pripadaju i animacija. Nacrt „BP_MasterAI“ doduše ne može imati ikakvo ponašanje bez nekakve vrste kontrolera, stoga kreiramo „BP_AIController“, koji ima dužnost provođenja logike koje mu definiramo.




Slika 34: BP_AIController

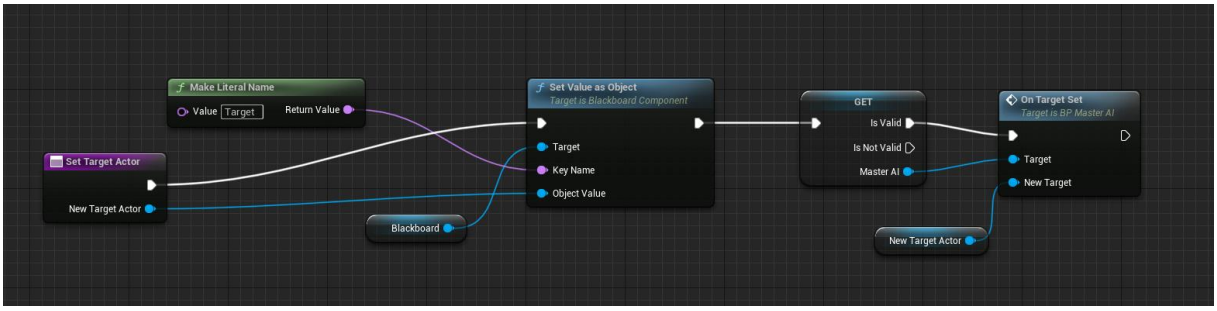
Dodajemo mu komponentu „AIPerception“ te ovisno o „Actor“ komponentama koje primijeti pozivamo funkciju „Update Perception“.



Slika 35: Update Perception

„Update Perception“ koristi dohvaćene „Actor“ te ovisno o vrsti „Actor“ postavlja ga kao novu metu ukoliko mu uđe u vidokrug definiran unutar „AIPerception“ postavka. Također ima mogućnost primjećivanja preko „AIStimulus“ kako bi postavio glavnog lika kao metu ukoliko glavni lik izazove taj podražaj putem, npr., napada.

Kada postavimo metu putem „Set Target Actor“ funkcije postavljamo naziv mete u našu „Blackboard“ komponentu unutar „Behaviour Tree“.

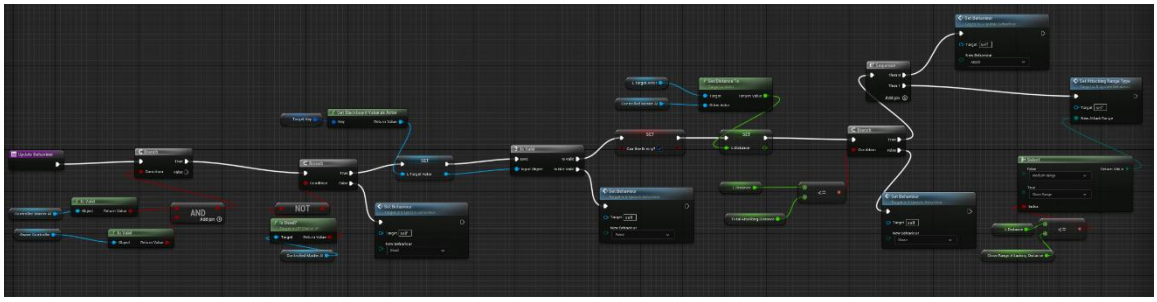



Slika 36: Set Target Actor

Sredstva „Behavior Trees“ u Unreal Engine 5 (Unreal Engine) mogu se koristiti za stvaranje umjetne inteligencije (AI) za likove koji nisu igrači u vašim projektima. Dok se sredstvo stabla ponašanja koristi za izvršavanje grana koje sadrže logiku, kako bi se odredilo koje se grane trebaju izvršiti, stablo ponašanja se oslanja na drugo sredstvo koje se zove „Blackboard“ koje služi kao "mozak" za stablo ponašanja [9].

„Blackboard“ sadrži nekoliko korisnički definiranih tipki koje sadrže informacije koje stablo ponašanja koristi za donošenje odluka. Na primjer, možete imati Booleov ključ pod nazivom Je li meta postavljena na koji se stablo ponašanja može pozvati da vidi je li se vrijednost promijenila. Ako je vrijednost istinita, mogla bi izvršiti grananje koje uzrokuje ganjanje igrača. Ako je netočan, mogao bi izvršiti drugu granu gdje se neprijatelj kreće nasumično po okolini.

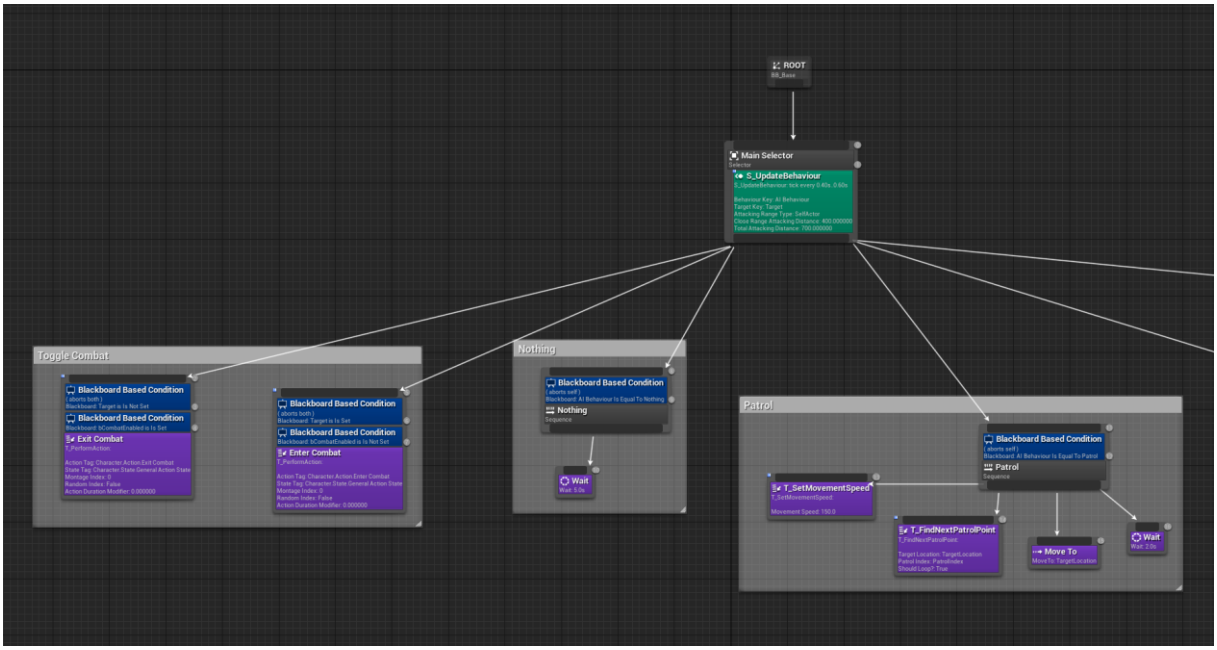
„Behaviour Tree“ nazvan u projektu „BT_MobEnemy“ postavljamo da svakih 0,4 sekunde poziva funkciju „Update Behaviour“.



Slika 37: Update Behaviour

„Update Behaviour“ nam provjerava je li AI uopće ispravan zajedno s njegovim kontrolerom, ukoliko je provjerava je li AI mrtav te ako nije postavlja si lokalnu vrijednost „Target Actor“. Ako je „Target Actor“ prazan, znači da meta nije pronađena te se AI postavlja u stanje „Patrole“ koja će uskoro biti detaljnije objašnjena. Ukoliko meta nije pronađena, ali tijekom patrole neprijatelj uoči metu računa udaljenog mete od sebe te kreće u napad. Ako je meta dovoljno blizu napad odmah započinje, ukoliko meta nije dovoljno blizu postavlja se u stanje „Chase“ odnosno ganjanje.

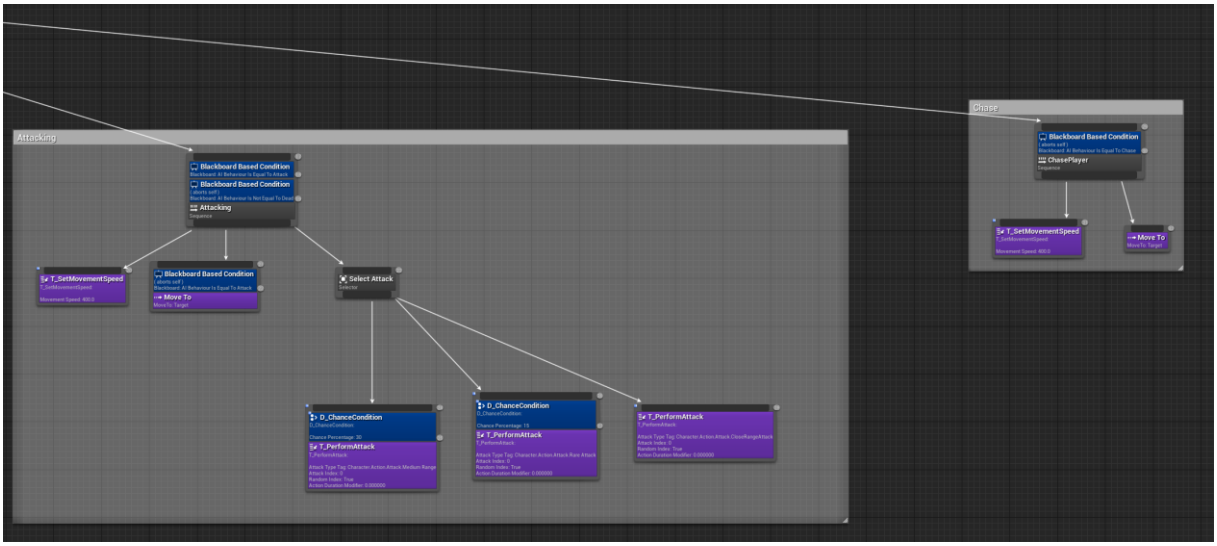
Unutar samog „BT_MobEnemy“ definiramo ponašanje koja se vrte slijedno od lijevo prema desno, gore do dolje.




Slika 38: BT_MobEnemy #1

Počevši od lijeve strane ukoliko meta nije postavljena, ali je stanje „CombatEnabled“ postavljeno na true započinje „Exit Combat“ gdje korektno ažurira svoje stanje i odigra animaciju za izlazak iz borbe. Ukoliko meta je postavljena, ali „CombatEnabled“ je false započinje „Enter Combat“ što također korektno ažurira stanje i odigra animaciju za ulazak u borbu.

Ukoliko je neprijatelj u stanju „Patrol“ onda postavljamo brzinu kretanja, pronalazi sljedeću točku za patrolu, kreće se prema tamo te pri dolasku čeka dvije sekunde.




Slika 39: BT_MobEnemy #2

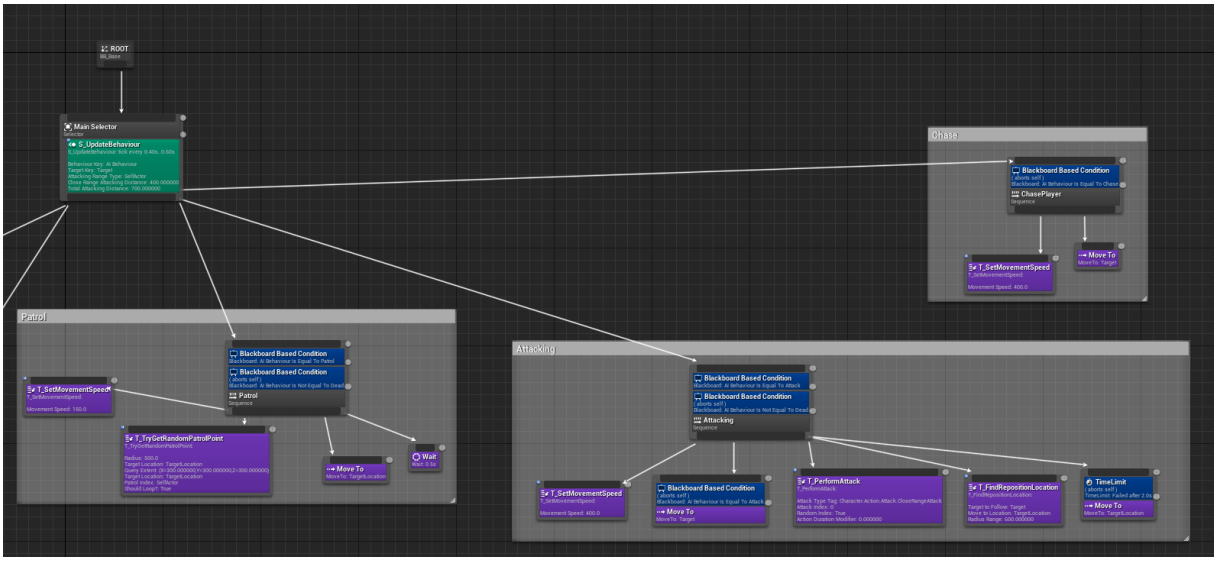
Ukoliko se nalazi u stanju napada te nije mrtav, brzina mu se ažurira i kreće se prema meti, ovisno o vrstama napada koje definiramo te šansama za njihovo okidanje bira između njih, sukladno tome ako je u stanju „Chase“ postavljamo mu brzinu kretanja i usmjeravamo meti.

Ostala dva primjera ponašanja neprijatelja su prisutna kod „Gruntling Enemy“ i kod glavnih neprijatelja.

### 3.3.5.1. Mob enemy

„BP_GruntlingEnemy“ jedan je primjer „child class“ od „BP_MobEnemy“ te je specifičan po tome što nema predefiniranu patrolu, neprijatelj se nasumično kreće po zoni unutar određenog radijusa te ukoliko pronađe metu kreće se prema meti, obavi jedan napad te ponovno traži nasumično mjesto kako bi se brzo odmaknuo od mete te ponovno vratio meti da obavi napad.

Sva logika za „BP_GruntlingEnemy“ definirana je u njegovom „Behaviour Tree“ „BT_GruntlingEnemy“.

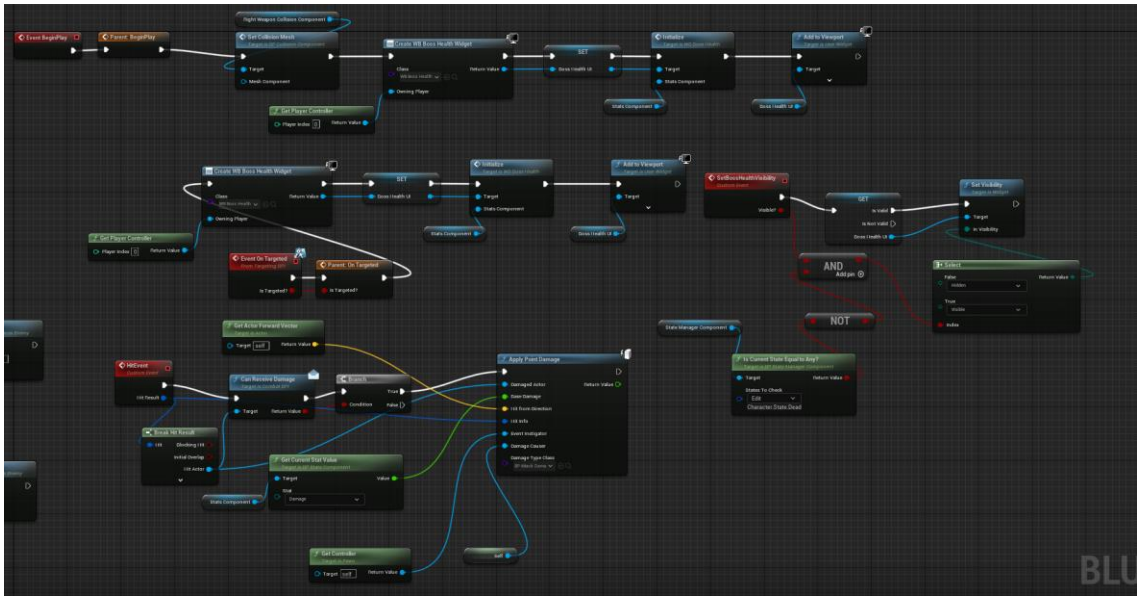



Slika 40: BT_GruntlingEnemy

### 3.3.5.2. Glavni neprijatelj

Glavni neprijatelj, za razliku od ostalih vrsta neprijatelja treba pružati veći izazov te ima vlastite životne bodove koji se prikazuju igraču na njegovom „Player HUD“.

Sva logika za glavnog neprijatelja razrađena je u „BP_BossEnemy“ kao „child class“ od prethodno spomenutog „BP_MasterAI“, time dobivamo mogućnost laganog dodavanja više vrste neprijatelja kao što ćemo prikazati u kasnijim sekcijama diplomskog rada.



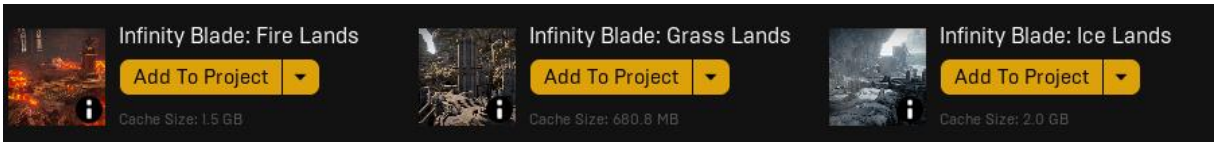
Slika 41: Boss Health Widget događaj

Pobjedom ovih vrsta neprijatelja stvaramo portal koji služi igraču kao tranzicija u sljedeći nivo putem „On Component Begin Overlap (Box)“ te funkcijom „Open Level by Name“.

Slika 42: Tranzicija u novi svijet

## 3.3.6. Kreiranje vanjskog svijeta za videoigru

Zahvaljujući Epic Gamesovom Unreal Engine Marketplaceu imamo velik odabir besplatnih imovina (*eng. Asset*) raznih zona koje možemo koristiti u sklopu videoigre. U sklopu ovog diplomskog rada koristit će se tri besplatne zone, a to su Infinity Blade: Grass Lands, Infinity Blade: Ice Lands, Infinity Blade: Fire Lands.




Slika 43: Infinity Blade besplatne zone

Svaku od zona potrebno je malo prilagoditi potrebama igre.

### 3.3.6.1. Infinity Blade: Grass Lands


Slika 44: Infinity Blade: Grass Lands

Postavljamo mjesta kojima se neprijatelji mogu kretati preko „Nav Mesh Bounds Volume“ (zeleno omeđeno) zatim postavljamo neprijatelje i radimo veliku plohu za borbu s glavnim neprijateljem. Osim toga potrebno je definirati mjesto gdje se igrač stvara preko „Player Start“ te postavljamo dvije vrste oružja koje može pokupiti na samom početku.

### 3.3.6.2. Infinity Blade: Ice Lands

Slika 45: Infinity Blade: Ice Lands

Proces postavljanja je isti kao u prethodnoj zoni, ali dodajemo posebno mjesto na koje igrač može doći ako dovoljno istražuje te će pronaći četiri vrste oklopa koje može obući i time ojačati svoju zaštitu od neprijateljskih napada.

### 3.3.6.3. Infinity Blade: Fire Lands

Slika 46: Infinity Blade: Fire Lands

Infinity Blade: Fire Lands zona služi kao završetak demonstracije diplomskog rada tako što prezentira više vrsta glavnih neprijatelja te nova oružja koja omogućuju pobjedu nad neprijateljima.

### 3.3.6.4. Implementacija Nanite tehnologije

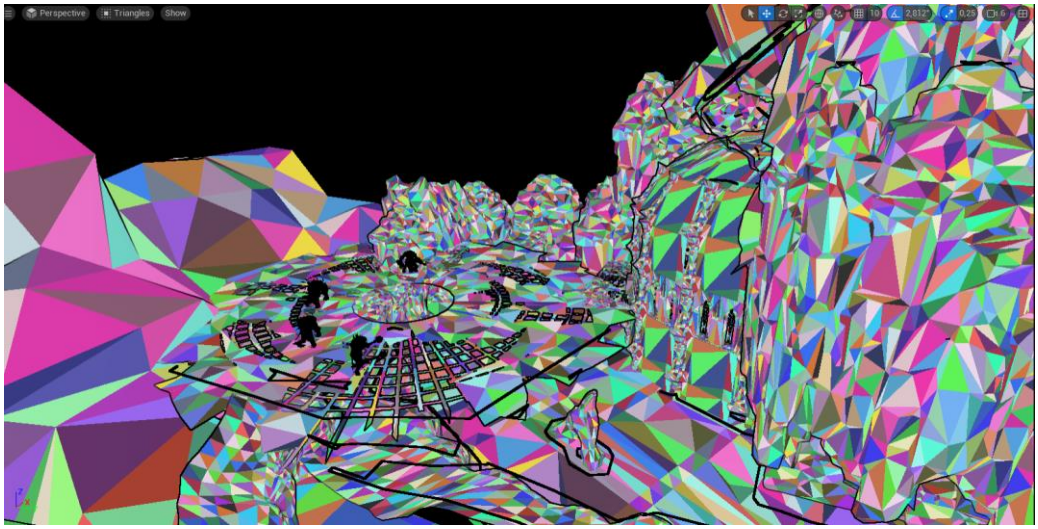
Nanite je novi i poboljšani virtualizirani geometrijski sustav uveden s Unreal Engine 5. Omogućuje interni format za uvezene mreže, stvarajući "Nanite mreže" za razliku od prethodnih statičkih mreža. Nova tehnologija renderiranja omogućuje Naniteu da bez napora renderira skalu piksela i visok poli broj. Sadrži nove metode renderiranja razine detalja (LOD) koje omogućuju odlaganje detalja i polibrojeva iz svake perspektive.

Prednosti Nanitea uključuju:

- Složenost geometrije, ukupni polibrojevi koji se mogu prikazati u stvarnom vremenu,
- Fotogrametrijska sredstva, mogućnost uvoza geometrija izravno u motor kako bi se umjetnicima omogućilo učinkovitiji rad s fotogrametrijskim sredstvima bez brige o višemilijunskim višebrojevima,
- Frame Budget, koji više nije ograničen policountima, pozivima za izvlačenje ili upotrebom mesh memorije,
- i Automatski LOD, gdje se razina detalja sada automatski postavlja nakon uvoza, čime se uklanja potreba za ručnim postavljanjem pojedinačnih LOD mreža.

[10]

Iako Infinity Blade zone su nešto starije te ne zahtijevaju preveliku optimiziranost, dobra je praksa uvijek osposobiti novu Nanite tehnologiju zato što će optimizacija i općenite performanse uvijek biti bolje.

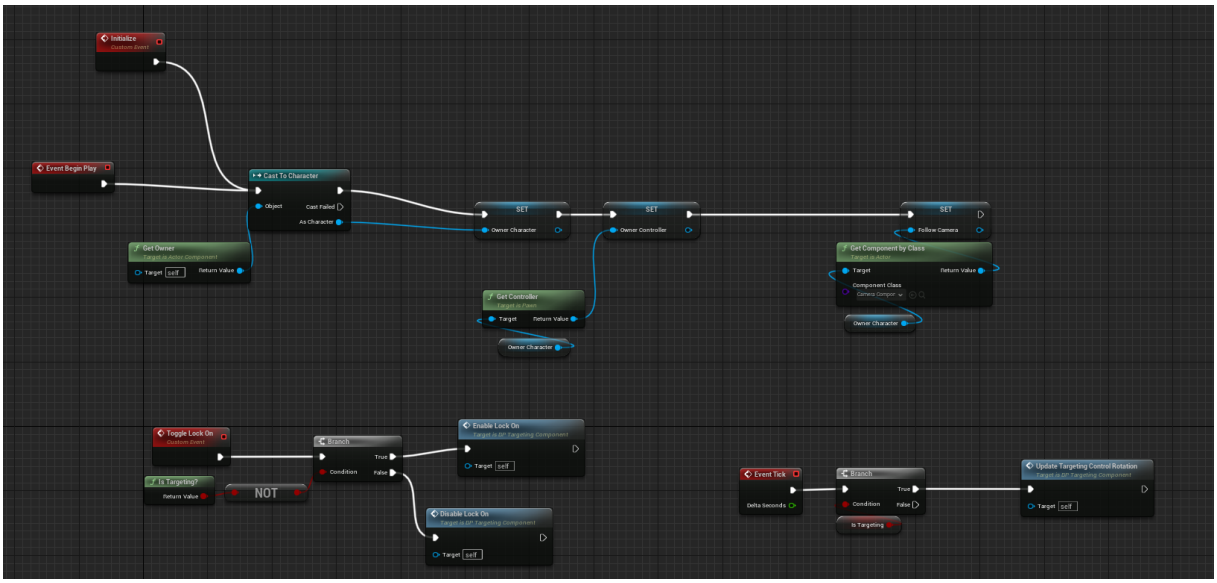


Slika 47: Nanite vizualizacija trokuta za Infinity Blade: Fire Lands

### 3.3.7.Targeting system

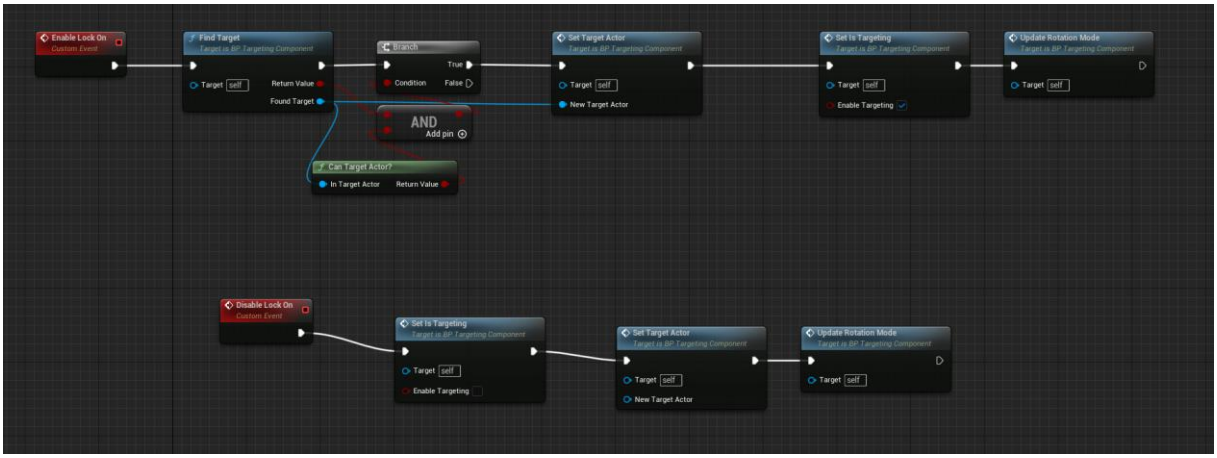
„Targeting System“ jedna od glavnih značajki „Souls-like“ igara koja omogućava lakšu kontrolu nad likom i snalaženje usred borbe. Svu logiku iza „Targeting System“ imamo definiranu u „BP_TargetingComponent“ komponenti.

Kada igrač odluči pokrenuti događaj „Toggle Lock On“ iz „BP_CombatPlayerCharacter“ putem odgovarajućeg inputa okidamo događaj „Enable Lock On“.



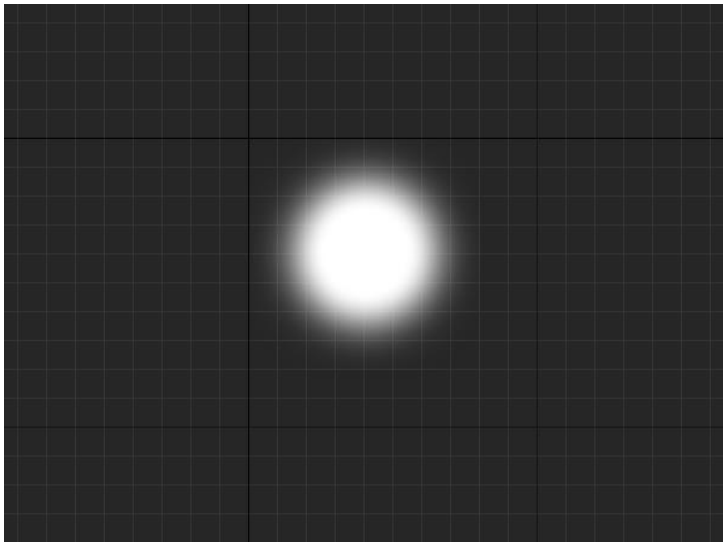

Slika 48: BP_TargetingComponent #1

Događaj „Enable Lock On“ pronalazi najbližu metu, postavlja ju kao „New Target Actor“ te obavlja rotaciju kamere prema neprijatelju putem „Update Rotation Mode“ funkcije.




Slika 49: BP_TargetingComponent #2

Kako bi igraču dali do znanja na kojeg neprijatelja je fokusiran, definiran je „WB_LockOn“ koji se pojavljuje na ekranu igrača kada neprijatelja događaj „Enable Lock On“ odabere.



Slika 50: "Lock On" Widget


Slika 51: „Lock On“ Widget na ekranu igrača

### 3.3.8.Interakcija igrača i neprijatelja sa svijetom

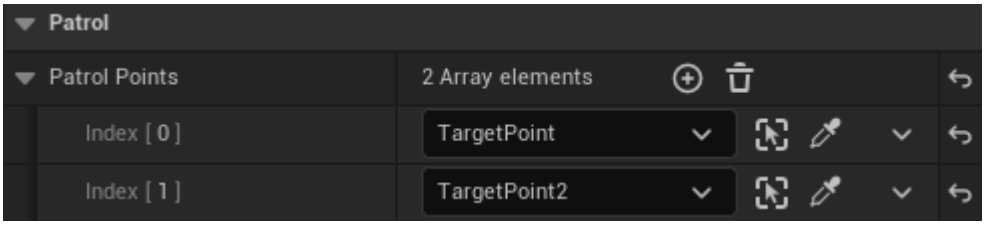
Unutar zona ovog diplomskog rada potrebno je bilo omeđiti velik dio zone kako igrač ne bi nepotrebno propao kroz zonu. To se postiglo postavljanjem „BlockingVolume“, odnosno prozirnih zidova koji onemogućuju prolaz igraču.


Slika 52: BlockingVolume

Ukoliko se desi da igrač i dalje uspije propasti kroz zonu pozivamo već postojeću „Kill Z“ funkciju koja uništava sve „Actor“ komponente ukoliko se nalaze na ili ispod vrijednost Z osi od -4000. Nakon toga zahvaljujući „Respawn“ funkcionalnosti oživljavamo lika na njegovoj zadnjoj kontrolnoj točki.

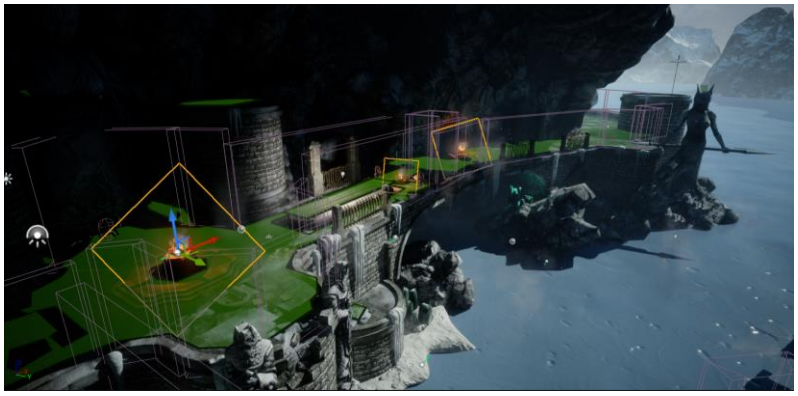
Kao prethodno spomenuto, neprijatelju omogućujemo kretnju preko patrolnih točaka te za svakog neprijatelja (izuzev onih koji imaju randomno hodanje) definiramo od koje do koje točke mora patrolirati.

Slika 53: Patrolne točke neprijatelja




Slika 54: Niz patrolnih točki

Budući da su zone bile proširene potrebno je također postaviti nove izvore svijetlosti koji se uklapaju s tematikom zone. Tako je u Infinity Blade: Ice Lands dodano 3 fenjera kao izvor svjetlosti tamo gdje je to bilo potrebno.



Slika 55: Dodavanje izvora svijetlosti

## 3.3.9.Odabir modela glavnog lika/neprijatelja i ostalih predmeta

Kako bi ova igra ostvarila ono što želi, a to je zabavan interaktivni svijet u kojem igrač kontrolira vlastitog lika te se bori s drugim neprijateljima potrebno je imati osjećaj težine modela. Kako bi to ostvarili Epic Games Unreal Engine Marketplace nudi nam niz besplatnih modela likova iz vrlo kvalitetne videoigre Paragon [11].

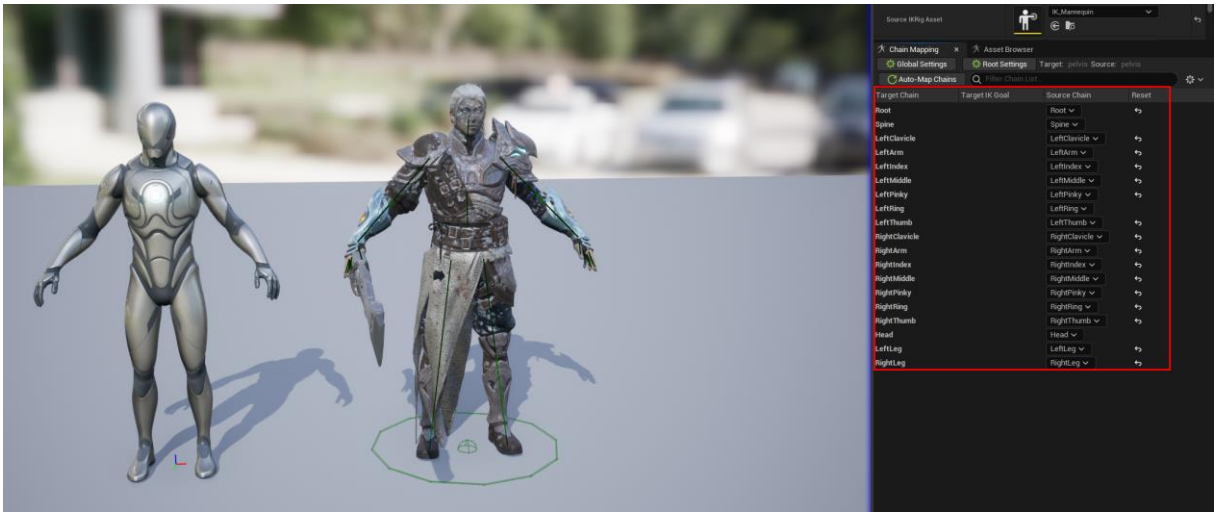
Kao mode glavnog lika koristiti će se lik „Kwang“ [12], za neprijatelje „Gideon“ [15] i „Gruntling“ [17] te za glavne neprijatelje „FrostGiantCaptain“ [17], „Rampage“ [14] i „Grux“ [16] te razna oružja iz „Infinity Blade: Weapons“ [20] i efekti iz „Infinity Blade: Effects“ [19].

### 3.3.9.1. Model glavnog lika

Kako bi mogli uporabiti model glavnog lika potrebno je prvo u programskom alatu Blender odvojiti model mača od modela samog lika te ih odvojeno uvesti u projekt.

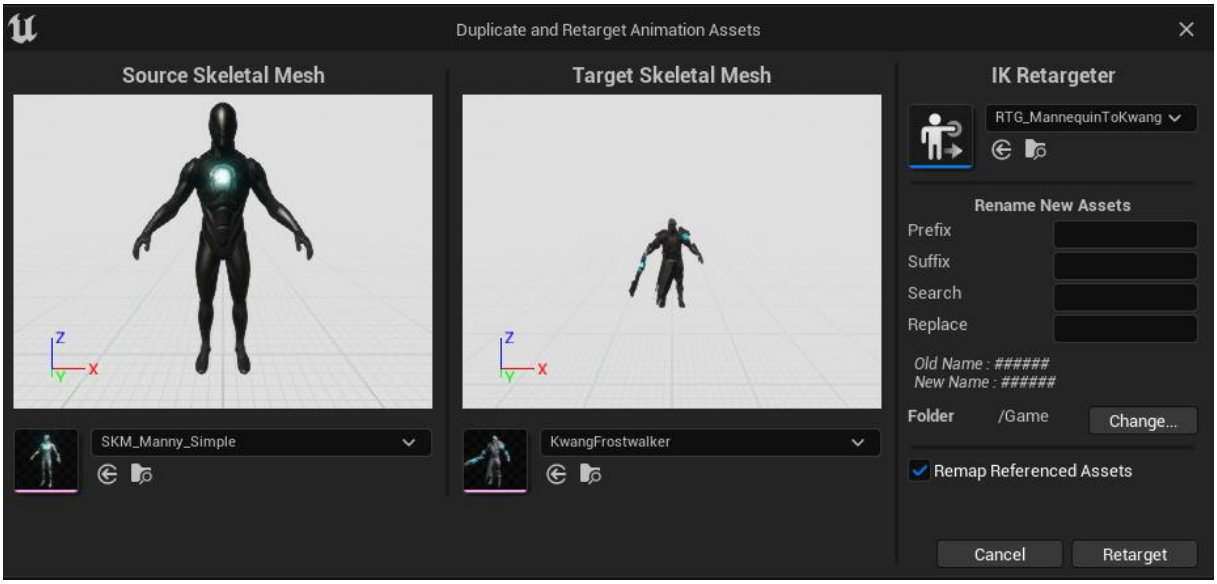
Slika 56: Model lika Kwang

Kako bi mogli koristiti sve animacije s testnog modela koji nudi Unreal Engine 5, potrebno je koristiti IK Rig Retargeter kako bi „Skeleta Mesh“ od testnog modela prenijeli na naš željeni model.



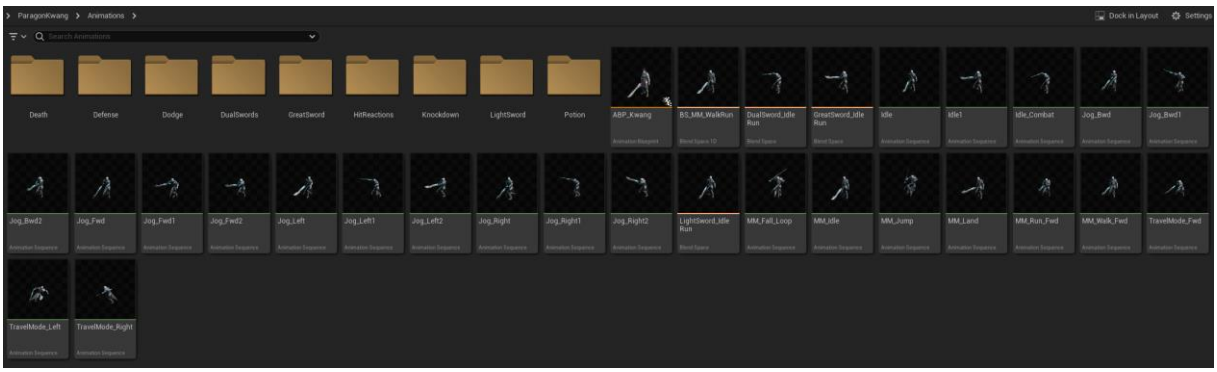
Slika 57: Kwang IK Rig Retargeter

Ovim putem spojili smo njihov „Skeletal Mesh“ te omogućili da koriste iste animacije, ali je za svaku animaciju potrebno ponovno postaviti za „Kwang“ model putem IK Rig Retargetera.




Slika 58: Retarget Animation Assets

Isti proces potrebno je primijeniti na sve animacije koje imamo i koristimo.



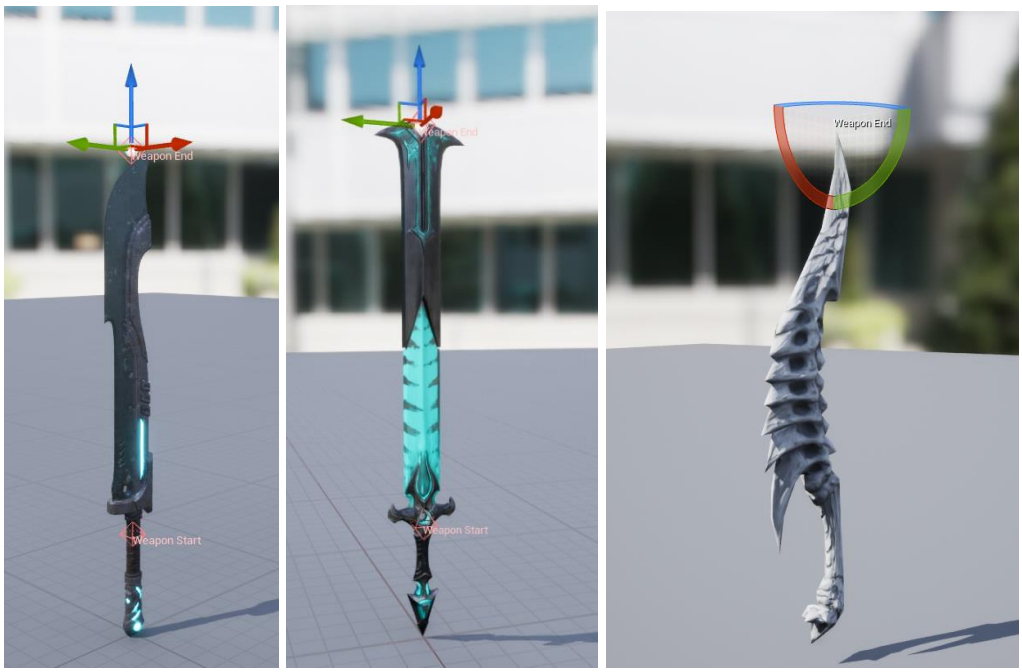

Slika 59: Kwang animacijski folder

Osim animacija potrebno je otići u kostur modela i postaviti sockete za oružja koja će koristit te gdje će ih postaviti.


Slika 60: Socketi u kosturu modela "Kwang"

### 3.3.9.2. Modeli oružja

Za svako oružje, bilo to od glavnog lika ili neprijatelja, potrebno je otići u „Socket Manager“ od modela i postaviti „Weapon Start“ i „Weapon End“ kako bi napravili oblik koji određuje koliziju. Postoji opcija korištenja modela kao oblik kolizije, ali takav pristup dosta utječe na performanse.




Slika 61: Modeli oružja - početak i kraj

### 3.3.9.3. Modeli neprijatelja

Najjednostavnije recikliranje animacija kako bi se napravio neprijatelj je neprijatelj vrsta „Skeleton“ koji koristi model „Gideon“. Proces postavljanja kostura i animacija identičan je kao za model glavnog lika „Kwang“ te se na lagan i brz način dobije vrsta neprijatelja slična glavnom liku te je također lako podesiva i proširiva.

Slika 62: Skeleton Mob model

Druga vrsta neprijatelja „Gruntling“ dolazi sa animacijama podešenim za njegov kostur te nema potrebe provoditi IK Retargeter proces, dovoljno je samo postaviti mu sockete i „Anim Notify“ u animacijama te dodijeliti oružje koje će koristiti.

Slika 63: Gruntling Mob model

Glavna vrsta neprijatelja slično kao „Gruntling“ neprijatelj ima svoje animacije, ali ujedno imaju i svoja oružja (šake, mačevi). Te je potrebno u samom kosturu modela definirati gdje oružje počinje, a gdje završava.

Slika 64: FrostGiantCaptain EnemyBoss model

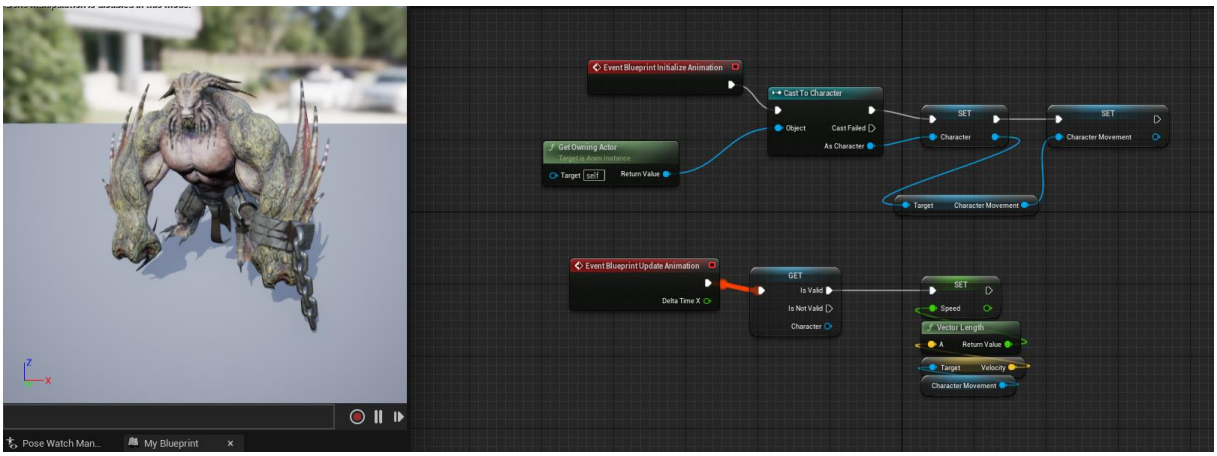
Istim procesom kreirali smo dodatnih pet vrsta glavnih neprijatelja:

Slika 65: Grux modeli neprijatelja

Slika 66: Rampage modeli neprijatelja

Kako bi se svaki od ovih neprijatelja ponašao drugačije, jednostavno je potrebno napraviti novi „Behaviour Tree“ i dodijeliti im u njihovom nacrtu.

Osim ponašanja, kako bi se model uskladio s kretanjem potrebno je za svaku novu vrstu neprijatelja napraviti „Animation Blueprint“.

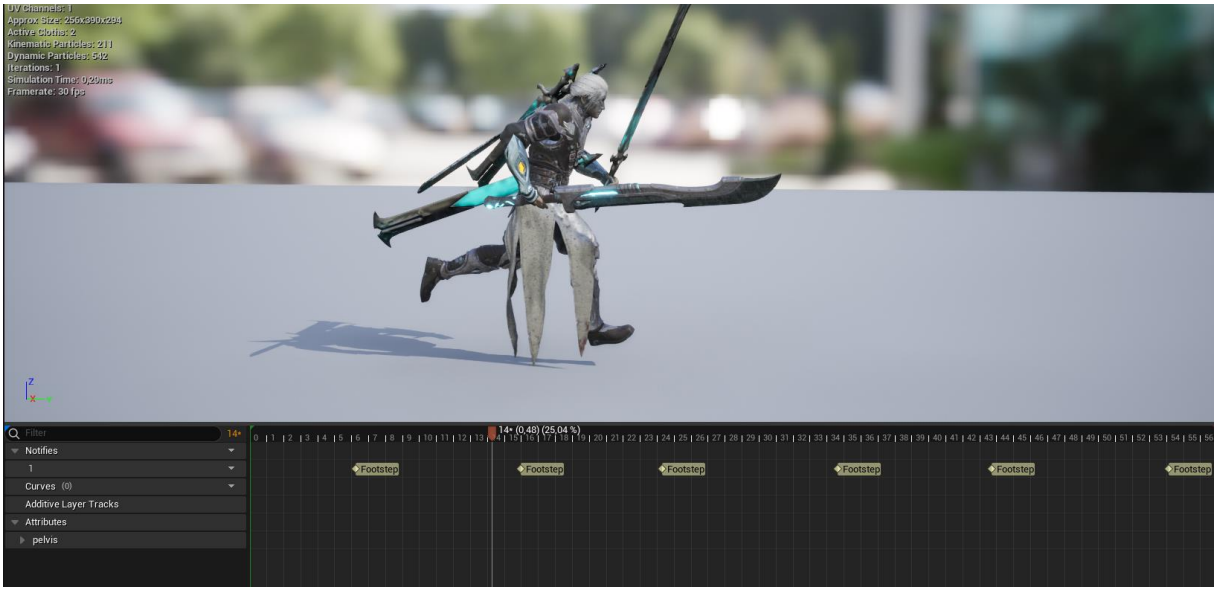



Slika 67: Rampage_ABP

### 3.3.10. Implementacija zvuka i efekta

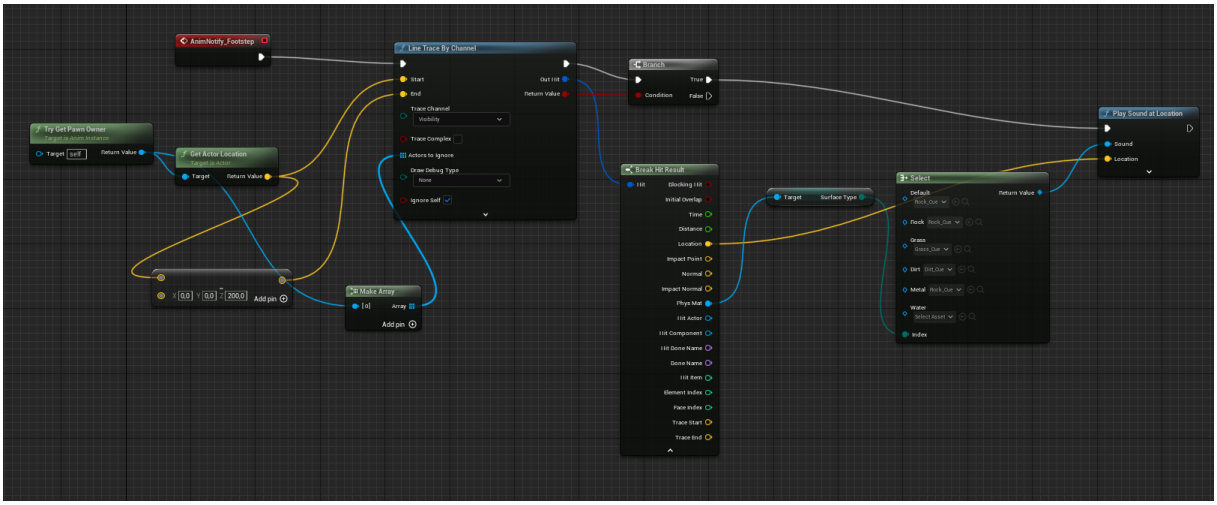
Implementacija zvukova i efekta primarno odvijamo pomoću animacija te u određenom trenutku animacije pokrećemo željene zvukove i efekte.

Kako bi simulirali zvuk koraka po specifičnom materijalu dodajemo „Anim Notify“ „Footstep“ kod svakog koraka kretanja animacija glavnog lika.



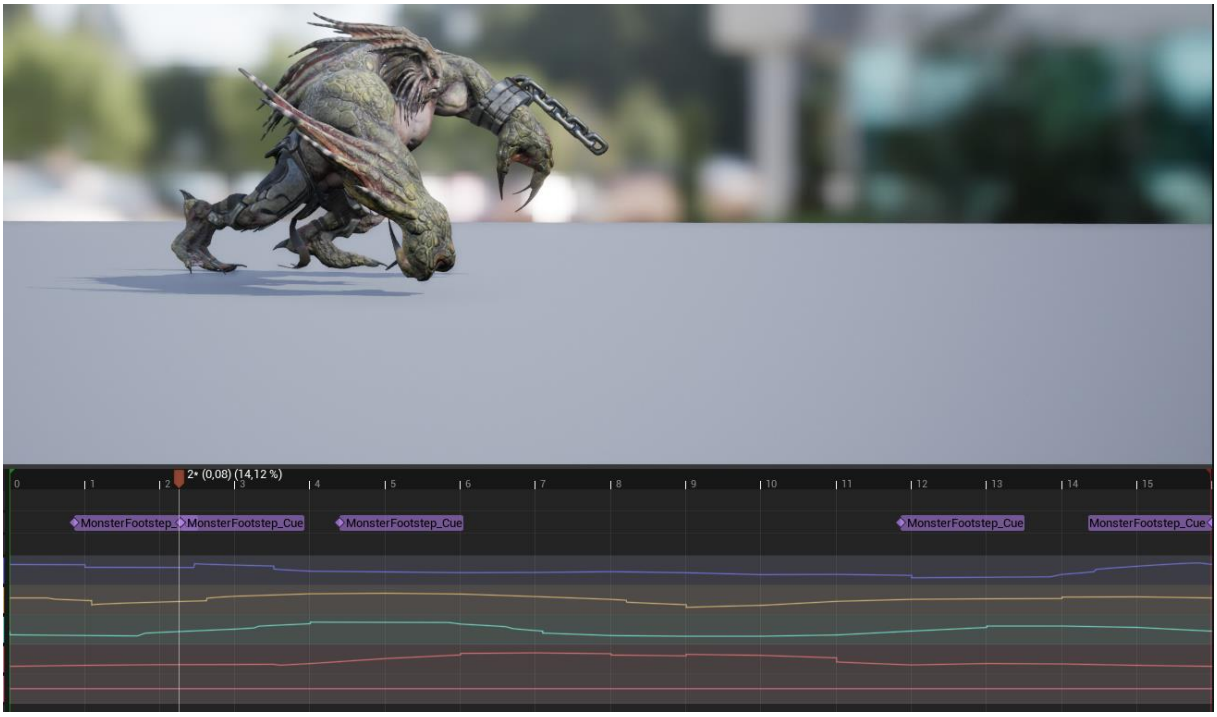

Slika 68: Footstep

„AnimNotify_Footstep“ događaj provjerava koji materijal se nalazi ispod stopala glavnog lika te ovisno o materijalu odabire odgovarajući zvuk, koji treba pokrenuti.



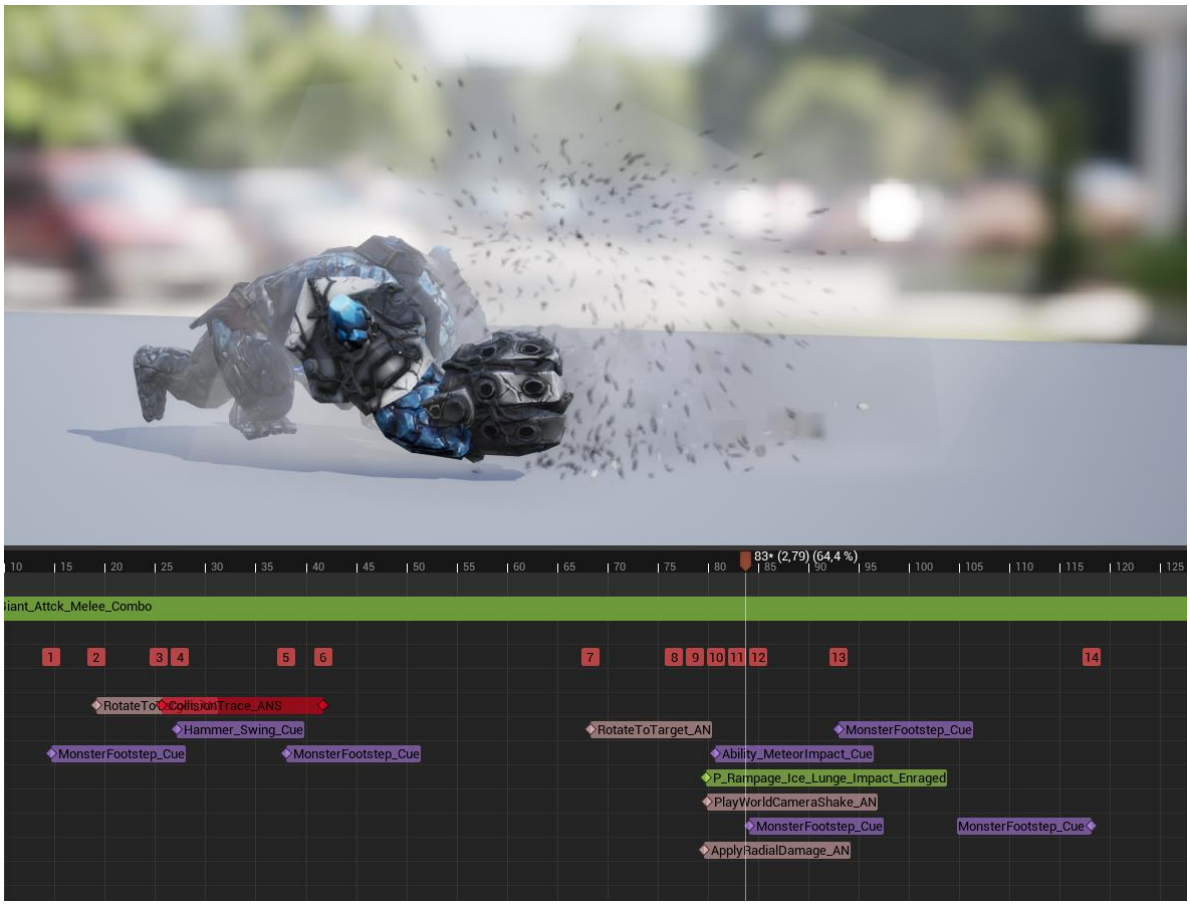

Slika 69: AnimNotify_Footstep

Za razliku od glavnog lika, kod neprijatelja jednostavno postavljamo „AnimNotify Play Sound“ te pozivamo zvuk da nasumičan zvuk za hod neprijatelja pozove u trenutku kad animacija dođe do njega.




Slika 70: AnimNotify MonsterFootstep_Cue

„AnimNotify“ posebno je koristan kada radimo napade neprijatelja koji najčešće zahtijevaju zvuk koraka, zvuk udarca, efekt udarca te „WorldCameraShake“, koji ovisno o postavljenim parametrima protrese kameru igrača kako bi dodatno simuliralo težinu udarca neprijatelja.




Slika 71: Primjer napada FrostGiantCaptain neprijatelja

Kako bi postavili gdje da se efekt udarca pojavi, potrebno je podesiti „Location Offset“.


Slika 72: Location Offset

### 3.3.11. Glavni izbornik

Glavni izbornik služi kao početno mjesto prije samog pokretanja igre, tu se igrač upoznaje s nazivom igre, uputama te može pročitati zasluge.


Slika 73: Main Menu

Pri prvom pokretanu otvara se „Main Menu“ widget te pritiskom na gumb „Insturctions“ otvara se „Instructions“ widget.


Slika 74: Instructions Widget

Pritiskom na gumb „Credits“ otvara se „Credits“ widget.


Slika 75: Credits Widget

Pritiskom na gumb „Quit“ izlazi se van iz igre, a pritiskom na gumb „Start“ započinje videoigra u prvoj razini, a to je razina „Grass Lands“.

# 4. Zaključak

Proces razvoja moje vlastite videoigre pomoću Unreal Enginea 5 bio je duboko iskustvo učenja, obilježeno mnoštvom tehničkih izazova i trijumfa. Napredne mogućnosti Unreal Engine 5 i opsežan skup alata omogućili su mi da ostvarim svoju kreativnu viziju i postavim kvalitetne temelje za daljnje proširivanje videoigre.

Tijekom ovog projekta stekao sam duboku zahvalnost za vrhunske značajke Unreal Engine 5 i njihov utjecaj na razvoj igre. Od korištenja dinamičke rasvjete i realističnih fizičkih simulacija do besprijekorne integracije audio i vizualnih efekata, ovaj se programski alat pokazao kao moćan alat za stvaranje impresivnih i vizualno zapanjujućih iskustava.

Poduzimanje ovog pothvata pojačalo je važnost ustrajnosti, prilagodljivosti i vještina rješavanja problema u suočavanju s tehničkim preprekama. Iterativna priroda razvoja igara naučila me vrijednosti iteracije i usavršavanja, kao i potrebi za učinkovitim tehnikama otklanjanja pogrešaka i optimizacije kako bi se osigurala optimalna izvedba.

Ovaj je projekt produbio moje razumijevanje zamršenog umijeća i pedantne izrade uključene u razvoj igre. Zamršenost stvaranja zanimljive mehanike igranja, zadivljujuće vizualne estetike i impresivnih zvučnih pejzaža dodatno su potaknuli moju znatiželju da istražim ogroman potencijal dizajna igara kao poslovnu priliku.

Dok završavam ovaj projekt, ponosan sam na tehnička dostignuća i znanje koje sam stekao. Ovo iskustvo učvrstilo je moju strast za razvojem igara i pojačalo moju želju da nastavim karijeru u ovom području koje se stalno razvija.

Kao zaključak mogu reći kako je prilika da razvijem vlastitu videoigru koristeći Unreal Engine 5 bila transformativno putovanje. Kroz svoje robusne značajke i inovativnu tehnologiju, Unreal Engine 5 mi je omogućio da ostvarim svoju kreativnu viziju i proširim svoju tehničku stručnost. Ovo iskustvo, ne samo da je ojačalo moje sposobnosti kao programera, već je i potaknulo moju ambiciju da istražim neograničene mogućnosti koje su pred nama u području razvoja videoigara.

# 5. Popis literature


[1] https://living.geico.com/home/technology/9-reasons-to-give-video-games-a-try/

[2] https://www.bankmycell.com/blog/video-game-industry-revenue

[3] https://bulldogjob.com/readme/unreal-engine-5

[4] https://docs.unrealengine.com/5.0/en-US/blueprints-visual-scripting-in-unreal-engine/

[5] Souls-like action rpg game with multiplayer. Available: https://devaddict.teachable.com/courses/souls-like-action-rpg-game-with-multiplayer/lectures/24976710. [Pokušaj pristupa 30. ožujak 2023.]

[6] Unreal Engine 5: Soulslike Melee Combat System https://www.udemy.com/course/unreal-engine-5-soulslike-combat/ [Pokušaj pristupa 30. ožujak 2023.]

[7] https://www.sumologic.com/glossary/encapsulation/

[8] https://docs.unrealengine.com/5.1/en-US/API/Runtime/Engine/Engine/UGameInstance/

[9] https://docs.unrealengine.com/5.0/en-US/behavior-trees-in-unreal-engine/

[10] https://cghero.com/glossary/what-is-nanite

[11] https://www.unrealengine.com/en-US/paragon

[12] https://www.unrealengine.com/marketplace/en-US/product/paragon-kwang

[13] https://www.unrealengine.com/marketplace/en-US/product/paragon-greystone

[14] https://www.unrealengine.com/marketplace/en-US/product/paragon-rampage

[15] https://www.unrealengine.com/marketplace/en-US/product/paragon-gideon

[16] https://www.unrealengine.com/marketplace/en-US/product/paragon-grux

[17] https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-enemies

[18] https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-plain-lands

[19] https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-effects

[20] https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-weapons

[21] https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-ice-lands

[22] https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-fire-lands

[23] Zvukovi: https://freesound.org/

# 6. Popis slika