Izrada akcijske igre iz trećeg lica u programskom alatu Unreal Engine 4

Cmrk, Matija

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matija Cmrk

IZRADA AKCIJSKE IGRE IZ TREĆEG LICA U PROGRAMSKOM ALATU UNREAL ENGINE 4

DIPLOMSKI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matija Cmrk

Matični broj: 0016110689

Studij: ***Informacijsko i programsko inženjerstvo***

IZRADA AKCIJSKE IGRE IZ TREĆEG LICA U PROGRAMSKOM ALATU UNREAL ENGINE 4

DIPLOMSKI RAD

Mentor/Mentorica:

Doc. dr. sc. Konecki Mladen

Varaždin, srpanj 2021.

Matija Cmrk

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 rad se temelji na izradi akcijske igre iz trećeg lica u alatu Unreal Engine 4. Rad sadržava primjenu 3d modela na likove, dodjeljivanje različitih animacija likovima za radnje tipa trčanje, skakanje, napadanje... Za izradu logike igre koristit će se „blueprint“ način skriptiranja. Time će se postići povezivanje lika i animacija, prepoznavanje komandi koje će igrač unijeti te umjetna inteligencija igre. Pokazat će se kako se koristi alat za izradu nivoa, korištenjem 3d modela i postavljanja istih u nivo. Kako bi igrač mogao pogledati kontrole igre i pokrenuti sam igrivi dio igre pokazat će se izrada glavnog izbornika uz ekrane za pobjedu ili poraz.

Ključne riječi: akcija; unreal; igra; animacije; blueprint; nivo

Sadržaj

1. Uvod

Od malih nogu bio sam očaran video igrama. Način na koji nas mogu transportirati u novi svijet i pružiti nam nova iskustava koja u pravom životu nikad ne bi mogli iskusiti, od prikaza različitih čarobnih svijetova, letenje svemirom ili prikazivanje našeg svijeta uz neki zaokret oduvijek su me fascinirale. Tu fascinaciju je popratila i želja da pogledam „pod haubu“ te vidim kako dizajneri stvaraju te svjetove. Industrija video igara raste iz godine u godinu i time postoji sve više različitih načina izrade igara, neki od njih su: Godot Engine, Unity Engine, Unreal Engine 4 koji se koristio u ovom radu te mnogi drugi. U završnom radu koristio sam Unity Engine kako bi izradio „Beat 'em Up“ igru te mi se dopalo taj način izrade. Za diplomski rad želio sam isprobati nešto novo. Za Unreal Engine 4 odlučio sam se jer je tvrtka koja ga proizvodi „Epic“ 2017. godine izdala igru „Fortnite“ koja je postala svjetski mega hit te je tvrtki donijela veliki profit koji su djelomično uložili za proširenje „Unreal Enginea“ i smanjili naknade za korištenje alata drugim tvrtkama. Uz to alat uz klasično programiranje u C++ za stvaranje logike igre omogućava skriptiranjem blueprinata koji ne daju nešto slabije performanse od C++ koda ali daju mogućnost vizualizacije logike koda spajanjem elemenata koji bi u klasičnom programiranju bile varijable, funkcije i slično. Time se postiže laki uvod u „kodiranje“ za ljude koji nisu u tome vješti dok ljudi koji imaju predhodna znanja programiranja mogu dobiti novu dimenziju kako zamisliti logiku što može biti dobro iskustvo. U ovom radu koristit ću blueprinte jer za opseg ove igre blueprinti neće primjetno utjecati na performanse igre te će mi pružiti sve potrebne resurse za kreiranje te igre.

2. Akcijska igra iz trećeg lica

Akcijska igra iz trećeg lica je igra u kojem je kamera pozicionirana iza leđa igrivog karaktera te se njime upravlja. Cilj igre je koristiti igrivog karaktera tako da tako da se pobijede ostali neprijatelji na mapi. U kontrolu lika ulaze: kretanje u 4 smjera u različitim brzinama, napadanje različitim napadima, blokiranje neprijateljskih napada te izbjegavanje istih, a te akcije će trošiti energiju od lika koja će mu se vraćati kroz vrijeme. Igrač će također moći koristiti eliksire za povratak životne energije, te eliksire će moći pronaći na nekim lokacijama nivoa te će mu moći biti dodijeljeni nakon što pobijedi protivnika. Igra je inspirirana igrama poput: Devil May Cry Nier: Automata, Dark Souls i slične.

3. Pokretač igre

Pokretač igre je okruženje za razvoj softvera uglavnom za stvaranje video igara. Pokretač igre može biti nazvan različitim stvarima te varijacije za ovaj pojam u industriji su „arhitektura igre“, „framework igre“ ili „frame igre“. Glavne funkcionalnosti koje pokretač igara mora uključivati sastoji se od prikazivanja grafike 2D ili 3D, mehanizma simuliranja fizike, prikaz animacija, puštanje zvuka, simuliranje umjetne inteligencije,... [1]

3.1. Komponente pokretača igre

Pokretač igre se u najširem smislu ponaša kako bilo koji drugi IDE za razvoj software-a ali u ovom slučaju opremljen za razvoj video igara. Komponente koje su uključene u pokretač igre su:

- Mehanizam pokretanja
- Umjetna inteligencija
- Mehanizam fizike
- Mehanizam zvuka
- Mogućnosti umrežavanja

Najpopularniji pokretači video igara su: Unity, Unreal Engine, Amazon Lumberyard, Arkit, Gamemaker...

3.2. Unreal Engine

Pokretač Unreal Engine je napravljen od tvrtke Epic Games. Prvi put je predstavljen 1998. godine u igri pucanja iz prvog lica nazvanog Unreal. U početku je bio namijenjen samo za igre pucanja iz prvog lica no kako je njegova popularnost rasla tako se i njegovo korištenje povećalo na svakakve igre koje su rađene u 3D svijetu. Četvrta inačica pokretača objavljena je 2014. godine, a od 2015. godine može biti preuzeta besplatno uz uvjet da programeri igara plate 5% zarade igre. Nakon uspjeha igre Fortnite, ako programeri objave igru na „Epic Games Storeu“ oslobođeni su plaćanja tog postotka, te neće uzimati nikakvu zaradu prvih milijun dolara zarade, što čini Unreal Engine jako dobru opciju za manje programere igara. [2]

3.2.1. Blueprint Visual Scripting

Blueprint Visual Scripting u Unreal Engine pokretaču je cjelovit sustav skriptiranja/programiranja igara zasnovan na konceptu korištenja sučelja temeljenog na

nodeovima za stvaranje elemenata igara iz Unreal Editora. Koristi se za definiranje objektno orijentiranih klasa ili objekata. Ovaj je sustav izuzetno fleksibilan i moćan jer pruža mogućnost dizajnerima da koriste gotovo čitav niz koncepata i alata koji su općenito dostupni samo programerima. Stvari specifične za Blueprint dostupne su i u implementaciji C++. Unreal Engine-a omogućavaju programerima stvaranje osnovnih sustava koje dizajneri mogu proširiti. [3]

3.2.2. Tržnica paketa asseta

Epic Games nudi tržnicu asseta u kojoj se mogu kupiti različiti asseti, od osnovnih modela predmeta do kompleksnih logičnih sistema. Korištenje tih asseta je jako jednostavno, u biblioteci kupljenih asseta samo treba pronaći željeni asset te pritisnuti tipku „Add to Project“ te će taj asset paket biti sam dodan u projekt na kojem se radi. Uz to Epic Games svaki mjesec nudi besplatno nekoliko paketa asseta koji se inače plaćaju te time olakšava proces izrade programerima igara. [4]

3.2.3. Unreal Engine 5

Nova inačica pokretača bila je predstavljana i objavljena za testiranje 26. svibnja 2021. Predstavljeni su novi elementi i sustavi dodani u pokretač koji će znatno olakšati i poboljšati proces razvoja igara: sustav otvorenih svjetova, sustav osvjetljenja, zvučni sustav,... Prognozira se da biti peta inačica pokretača u potpunosti biti dostupna početkom 2022. godine. [5]

4. Proces stvaranja igre

U ovom će poglavlju biti opisan proces stvaranja igre iz trećeg lica. Igra će biti izrađena u verziji pokretača 4.26. Opis procesa bit će popraćen slikama povezanih blueprint skripta.

Kako bismo kreirali igru trećeg lica u procesu kreiranja treba odabrati novi projekt vrste „Third Person“ i „Blueprint“ verzija projekta te kliknuti na stvaranje projekta. Nakon što je podloga projekta stvorena, dobijemo generirani nivo trećeg lica te lika kojeg možemo kontrolirati te se kretati po svijetu.

Slika 1: Generirani nivo trećeg lica (Izvor: vlastita slika iz alata)

4.1. Kontrole

Kako bi kontrolirali karaktera trebaju se postaviti tipke davanja signala karakteru. To se može postaviti pod: „Settings -> Project Settings -> Input“.

Pod „Axis Mappings“ već su postavljene tipke kretanja te njih ne moramo mijenjati, postoje postavke više različitih tipki na različitim uređajima za jednake akcije no za ovu igru mi ćemo koristiti kombinaciju tipki koja je standard u igrama već godinama, a to je „WASD“ kombinacija.

Pod „Action Mappings“ postavit će se tipke koje će se koristiti za ostale akcije koje će karakter moći izvoditi:

- Skok: Space Bar

- Sprint: Left Shit
- Trčanje / hodanje: Caps Lock
- Čučanj: Left Ctrl
- Vađenje / spremanje oružja: 1
- Prva kombinacija napada: Left Mouse Button
- Druga kombinacija napada: Right Mouse Button
- Obrana: Q
- Fokus na neprijatelja: Middle Mouse Button
- Izbjegavanje: Left Alt
- Korištenje eliksira: Tab

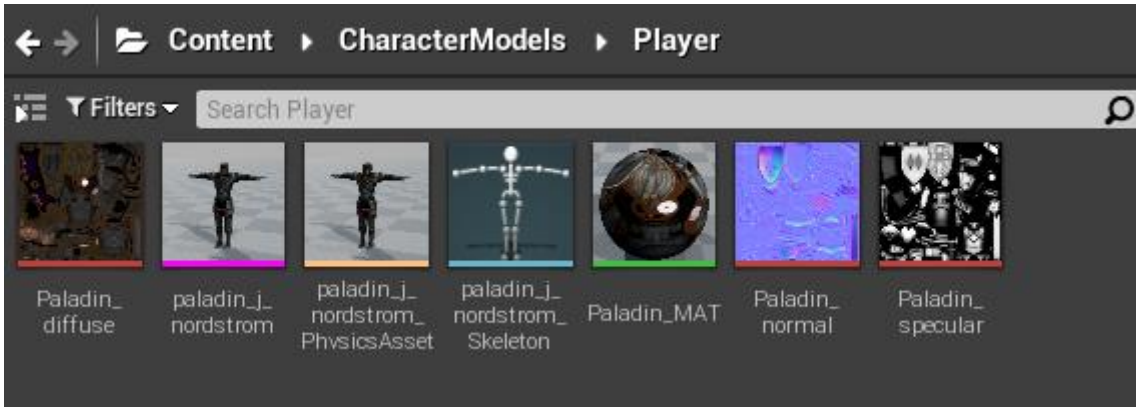
4.2. Implementacija animacija kretanja

4.2.1.Igrivi karakter

4.2.1.1. Ubacivanje likova i animacija u projekt

Da bi kreirali našeg igrivog karaktera potrebni su model karaktera te popratne animacije koje će taj karakter imati i izvoditi. Modele igrivog karaktera i neprijatelja kao i njihove animacije preuzet će se sa stranice Mixamo. Mixamo je web stranica tvrtke Adobe preko koje se mogu preuzeti različiti likovi i animacije. [6]

Kada se preuzmu karakteri i animacije potrebno ih je ubaciti u projekt. Prvo je potrebno prebaciti karaktera, dovoljno je samo povući preuzetu datoteku u željenu mapu u pokretaču te će si pokretač sam generirati potrebne reference za karaktera.



Slika 2: Ubačen lik u pokretač (Izvor: vlastita slika iz alata)

Nakon toga povučemo animacije te odaberemo skeleton lika kojeg smo prije ubacili u projekt kako bi pokretač mogao napraviti konekciju između animacije i lika. Valja reći da kada se stavljaju ostali likovi u projekt potrebno je na Mixamo stranici za njihov model preuzeti animacije prilagođene njima te njih ubaciti u projek, jer ako koristimo animacije koje su namijenjene drugom liku, animacije će izgledati potrgano zato što svaki lik na Mixamu ima različiti skeleton i to stvara probleme jer se neće moći povezati skeleton i animacija.

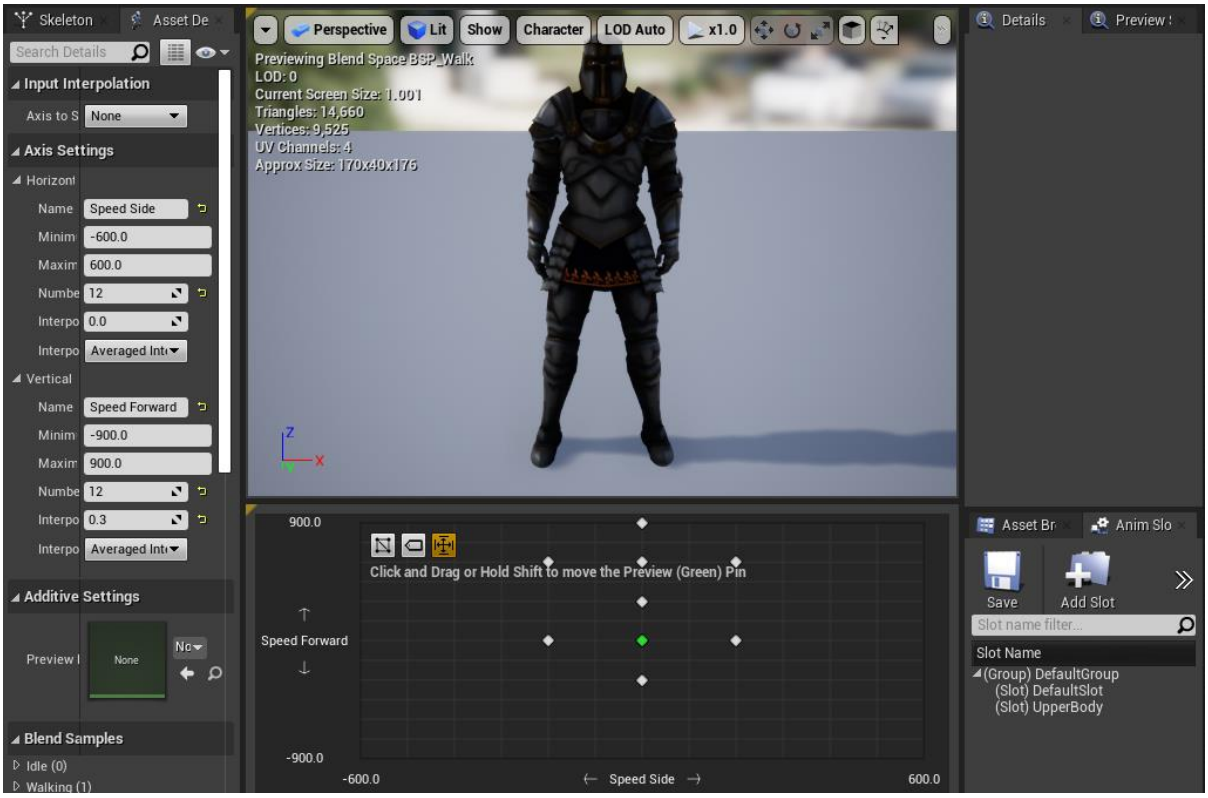
Slika 3: Ubačene animacije (Izvor: vlastita slika iz alata)

4.2.1.2. Izgled igrivog karaktera

Slika 4: Izgled igrivog karaktera (Izvor: vlastita slika iz alata)

4.2.1.3. Animation Blendspace

Nakon što su animacije ubačene u projekt možemo kreirati Animation Blendspace u kojem ćemo postaviti interakciju animacija i brzina kretanja lika kada je pritisnuta neka tipka za kretanje. Kreirat će se jedan Blendspace za uspravno kretanje i jedan za kretanje u čučnju. Kako bi postavili potrebne podatke postavit će maksimalne i minimalne brzine za horizontalno i vertikalno kretanje. Za horizontalno kretanje postavit će se maksimalna i minimalna brzina na 600 i -600 za uspravno kretanje a za vertikalno kretanje 900 i -900, razlog zašto imamo veće brojeve na vertikalnom kretanju je to što će igra imati opciju sprinta te se tako želi simulirati osjećaj sprinta. Nakon što su postavljeni minimalna i maksimalna brzina potrebno je ubaciti na grafu na kojoj brzini kretanja karaktera želimo pokrenuti koju animaciju.

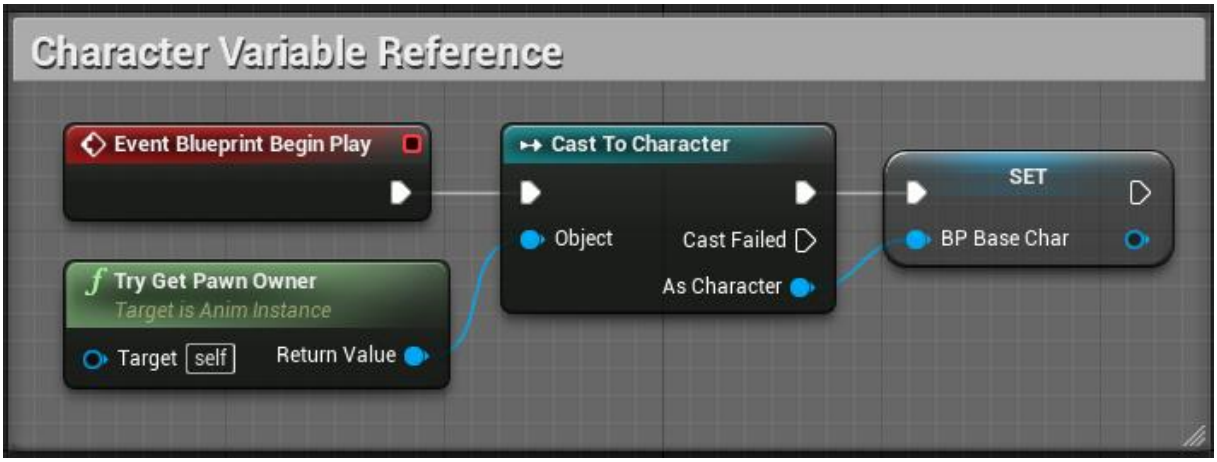


Slika 5: Animation Blendspace za kretanje (Izvor: vlastita slika iz alata)

4.2.1.4. Animation Blueprint

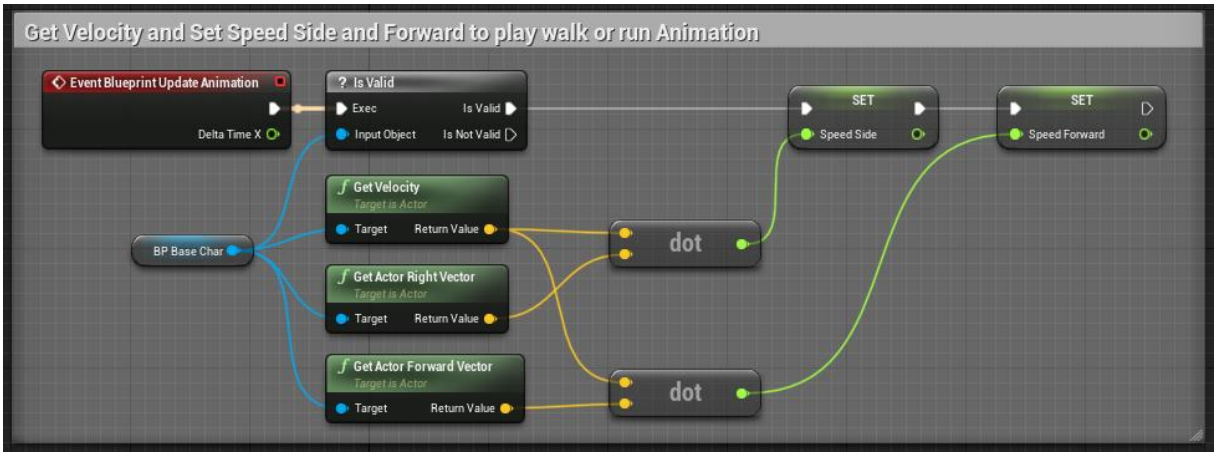
Nakon kreiranih Blendspace-a potrebno je kreirati Animation Blueprint kako bi se mogla postaviti logika povezana uz animacije. Uz Animation Blueprint potrebno je kreirati baznu Blueprint klasu koja će se zvati „BP_Base“ te će se koristiti za sve likove jer će imati logiku koja će biti povezana sa svim likovima u igri. Uz to treba kreirati Blueprint Interface preko kojeg će se prenositi vrijednosti varijabla u različitim Blueprintovima.

U Animation Blueprintu kreiramo varijable „Speed Side“(Float), „Speed Forward“(Float), “BP Base Char“(Character), „Crouched“(Boolean). Da bi animacije radile trebamo postaviti igrača u varijablu “BP Base Char“ da bi imali referencu prema igraču. Iz „Event Blueprint Begin Play“ koji se pokrene na početku igre pokušamo dohvatiti „Pawn“ koji je naš igrivi karakter te njega pokušamo transformirati u Character i postaviti u varijablu.



Slika 6: Postavljanje igrača u varjiablu (Izvor: vlastita slika iz alata)

Kada se animacija promijeni želimo provjeriti je li “BP Base Char“ validna te ako je iz nje izvučemo brzinu i vektore kretanja te ih transformiramo i to spremimo u varijable „Speed Side“ i „Speed Forward“.

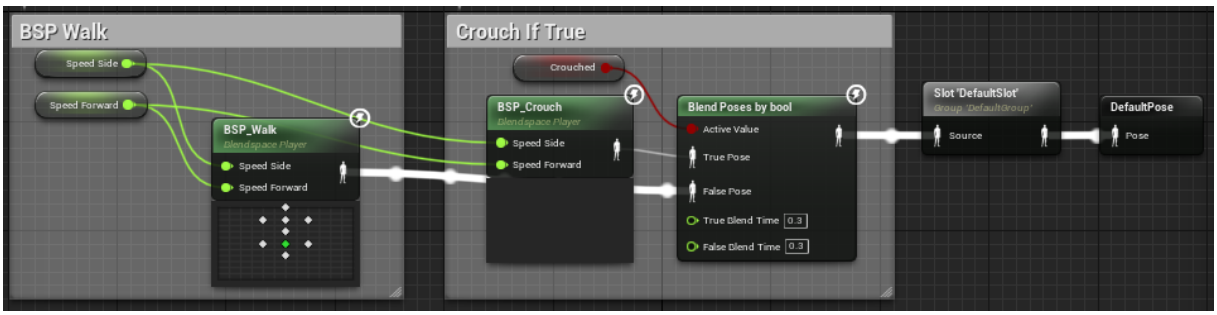


Slika 7: Postavljanje varijabla brzine (Izvor: vlastita slika iz alata)

Kako bi znali da li je karakter u čučnju pozovemo događaj iz sučelja za animacija u kojem se kreirala funkcija za čučanj koja ima jedan input „Enable“(Boolean) te njega spremimo u varijablu „Crouched“.

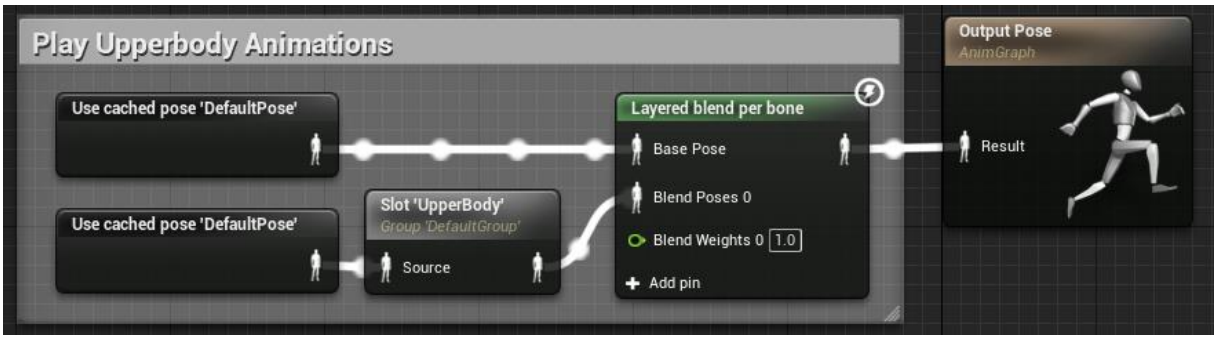
Slika 8: Provjera čučnja (Izvor: vlastita slika iz alata)

Pozivanje Blend Spacea se izvrši tako da iz varijabla Speed Side“ i „Speed Forward“ postavimo Blend Space za normalno kretanje i za kretanje u čučnju, nakon toga provjerimo je li karakter u stanju čučnja i prema tome proslijedimo odabrani Blend Space prema DefaltSlotu i prema DefaultPose.



Slika 9: Pozivanje Blend Spacea (Izvor: vlastita slika iz alata)

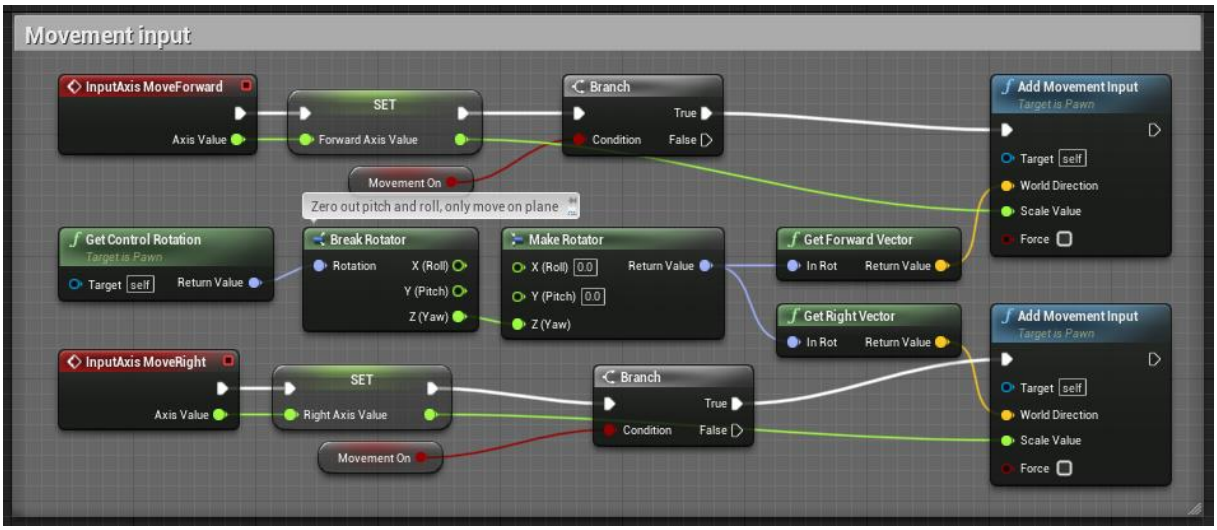
Za slučaj kada imamo animacije koje zahtijevaju da se gornji dio tijela ponaša na jedan način a donji dio na drugi (tipa izvlačenje mača kada se trči), postavimo Default Pose i Default Pose prebačen u UpperBody te njih stavimo u „Layerd blend per bode“ i to stavimo u output Pose. Tako se osiguralo da se neće dogoditi neprirodne animacije.



Slika 10: Postavljanje animacija za gornji dio tijela (Izvor: vlastita slika iz alata)

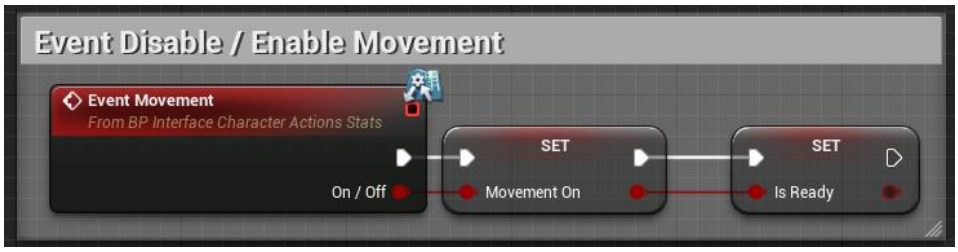
4.2.1.5. Bazična Blueprint klasa likova

U „BP_Base“ kreirat će se varijable „ForwardAxisValue“(Float), RightAxisValue“(Float), „MovementOn“(Boolean), „Dead“(Boolean), „IsReady“(Boolean), „Running“(Boolean), „Crouching?“(Boolean), „Sprinting“(Boolean), „Dodging“(Boolean) te ćemo kreirati logiku za čitanje unosa kontrola igrača. Desnim klikom miša možemo dodati „InputAxis MoveForward“ i „InputAxis MoveRight“ koji se aktiviraju kada kliknemo tipke za kretanje njih postavimo u variable „ForwardAxisValue“ i RightAxisValue“ zatim ako je dozvoljeno kretanje uz pomoć „Control Rotation“iz njegovog Z(Yaw) dobije se forward i right vector to se oni dodaju u MovementInput.



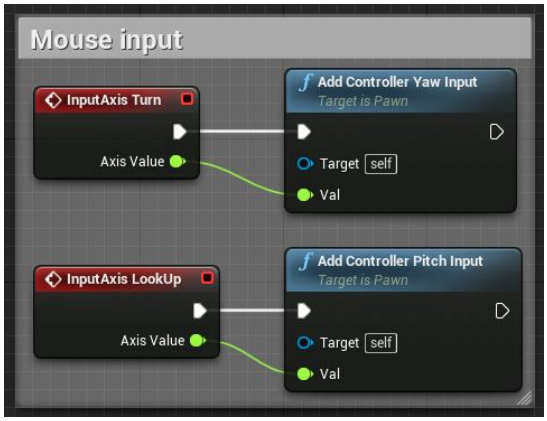
Slika 11: Blueprint logika čitanja tipki kretanja (Izvor: vlastita slika iz alata)

Kako u nekim akcije nebi trebalo biti mogućnosti kretanja (tipa napadanje), u sučelju napravimo metodu za event Movement s inputom „On / Off“(Boolean) ćemo preko njega postaviti varijablu „MovementOn“ na false ili true te nakon nje „IsReady“ s jednakom vrijednosti.



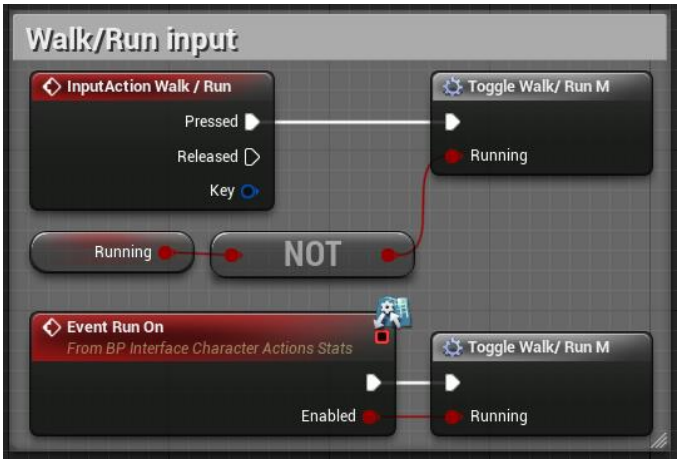
Slika 12: Omogućavanje kretanja (Izvor: vlastita slika iz alata)

Kako bi igra prepoznala miš trebamo pročitati InputAxis za okretanje i pogledavanje gore dolje te to dodati u Controller.



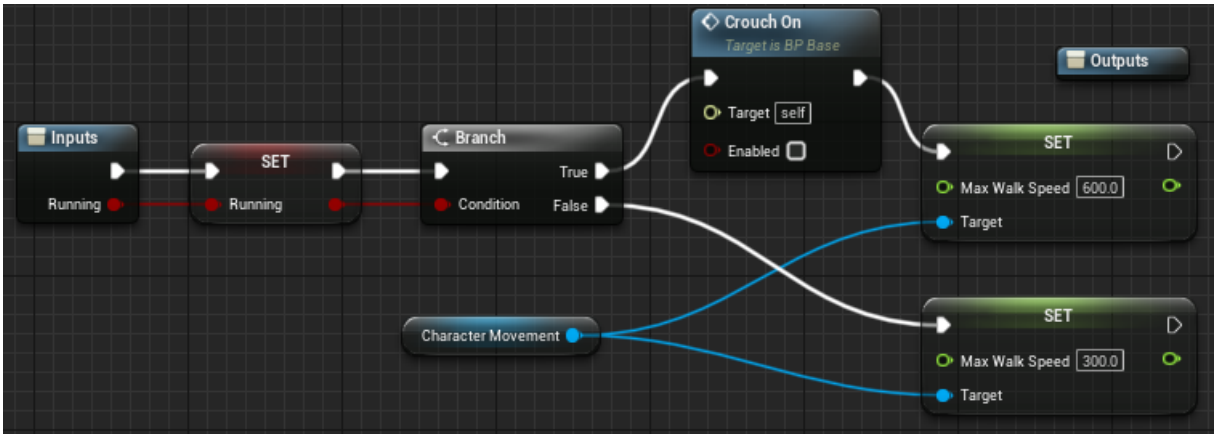
Slika 13: Čitanje kretanja miša (Izvor: vlastita slika iz alata)

Kako bi promijenili da karakter trči ili hoda uzimamo „InputAction Walk/Run“ koji je tipka Caps Lock te kada je prekinuta postavljamo prebacujemo varijablu „Running“ u suprotno stanje i prosljeđujemo u Macro koji smo kreirali da bi promijenili brzinu kretanja.



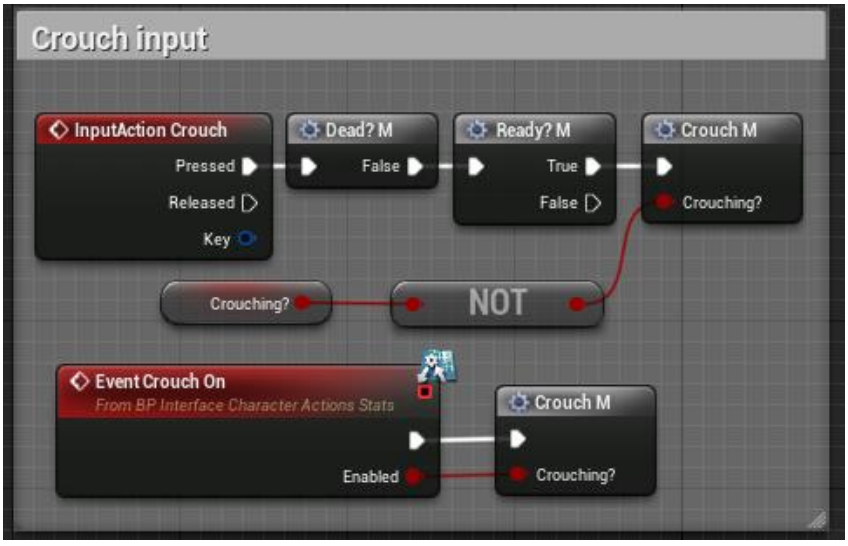
Slika 14: Logika pritiska tipke za trčanje/hodanje (Izvor: vlastita slika iz alata)

U macru kao ulaz imamo varijablu „Running“ te nju iznova postavljamo, ako je sada varijabla istina isključujemo čučanj ako je postavljen te postavimo brzinu kretanja na 600, ako je varijabla sada laž postavimo brzinu kretanja na 300.



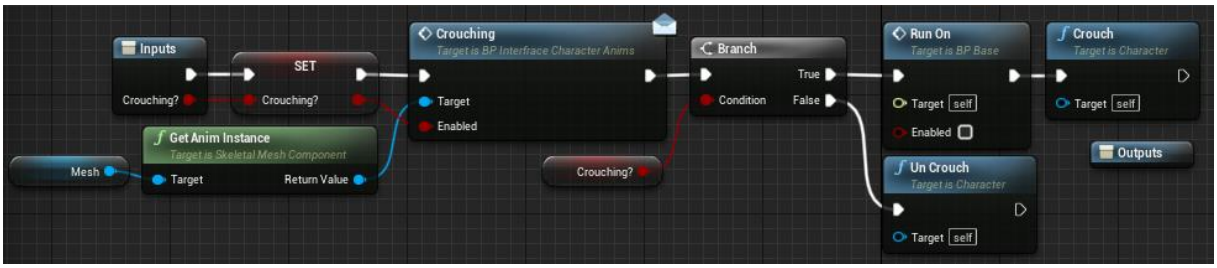
Slika 15: Macro postavljanja brzine trčanja / hodanja (Izvor: vlastita slika iz alata)

Da bi karaktera prebacili u stanje čučnja pozivamo „InputAction Crouch“ koja se nalazi na tipki Left Ctrl, nakon toga se provjerava da li je karakter mrtav i spreman preko zasebnih macroa (koji će biti objašnjeni kasnije) za pokrenuti tu akciju da se izbjegne izvođenje akcija u mrtvom stanju. Ako se zadovoljavaju ti uvjeti kao i za trčanje/ hodanje uzima se inverzija varijable, u ovom slučaju „Crouching“ te se prosljeđuje u macro za postavljanje čučnja. Event za čučanj poziva macro za čučanj kada je pozvan te mu prosljeđuje je li omogućen čučanj.



Slika 16: Logika pritiska tipke čučnja (Izvor: vlastita slika iz alata)

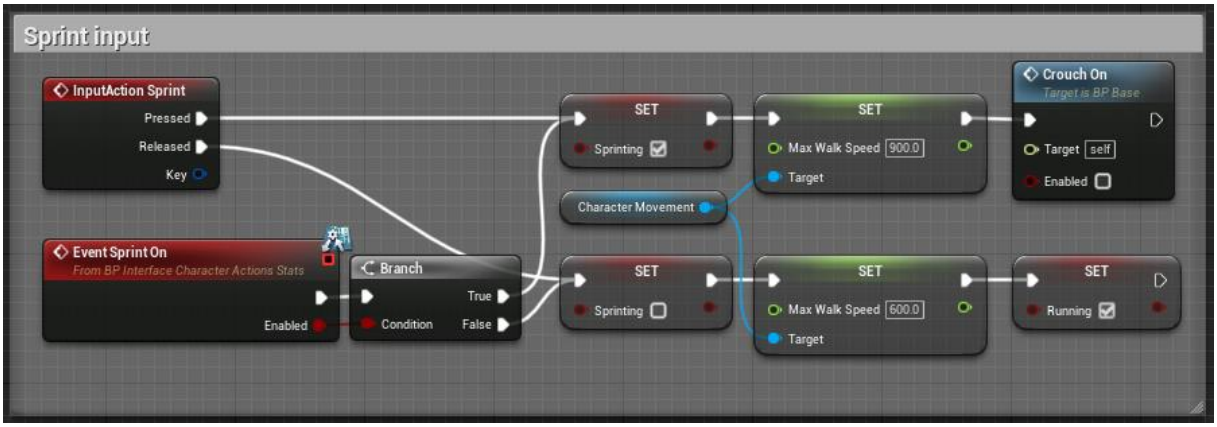
U macru čučnja imamo ulaz s varijablom „Crouching te nju postavljamo, nakon toga iz Mesha lika dobijemo Anim instancu koju postavljamo u funkciju Crouching iz sučelja namijenjenog da prenese Animation Blueprintu je li u stanju čučnja, nakon toga se provjerava je li lik u stanju čučnja, ako je, zove se Event Run kako bi se isključilo trčanje lika i smanjila brzina, te se postavi movement akcija Crouch, ako nije u stanju čučnja zove se movement akcija Un Crouch.



Slika 17: Macro za postavljanje čučnja (Izvor: vlastita slika iz alata)

Da bi karakter mogao uči u sprint uzima se „InputAction Sprint“ koji je povezan s tipkom Shift. Ako je tipka pritisnuta, postavljamo varijablu Sprinting u stanje istine i postavimo brzinu kretanja na 900 te isključimo preko eventa čučanj ako je uključen. Ako je tipka puštena varijablu Sprinting postavimo u stanje laži te brzinu kretanja postavimo na 600 i kako te varijablu Running postavimo u stanje istine kako bi karakter nakon sprinta mogao normalno trčati ako prije pritiska tipke sprinta karakter nije trčao.

Za event Sprinta provjerava se je li sprint bio omogućen, te se prema njemu izvršava logika kao i za pritisak tipke.



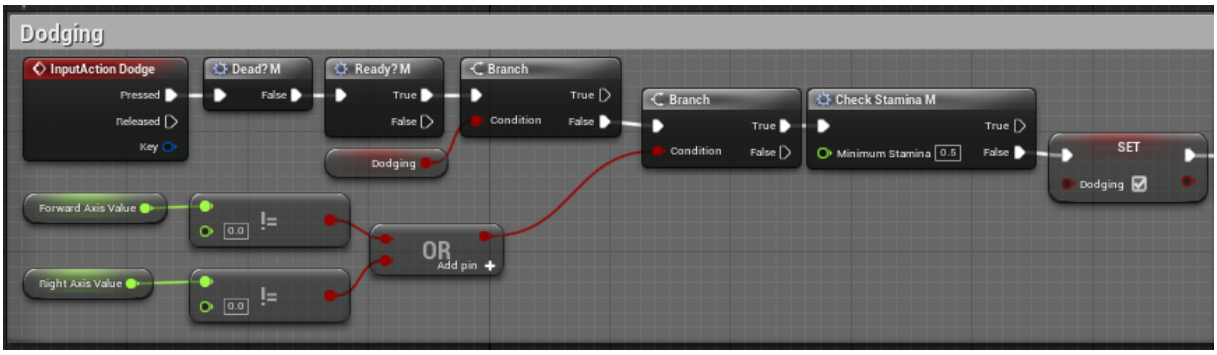
Slika 18: Logika Sprinta (Izvor: vlastita slika iz alata)

Kako bi se implementiralo skakanje u igri prvo treba iz animacije skakanja stvoriti Animation Montage. To se napravi tako da se na datoteku animacije klikne desni klik miša zatim Create -> Create AnimMontage.

Nakon što je AnimMontage kreiran u Blueprintu uzimamo „InputAction Jump“ koji je vezan za tipku Space provjeri se je li karakter mrtav i je li spreman iz razloga jer ne želimo da karakter može skočiti ako je mrtav ili ako radi neku drugu radnju. Ako je to zadovoljeno zove se Jump funkcija koja je postavljena po defaultu u Unreal Engineu, nakon toga se koristi „Play Anim Montage“ kako bi se izvršila animacija skakanja, pod Anim Montage moramo u padajućem izborniku odabrati koji Anim Montage želimo, u ovom slučaju za animaciju skakanja. Nakon što se izvršila animacija skakanja postavlja se varijabla za mogućnost kretanja na istinu u slučaju da je bila postavljena na laž ranije.

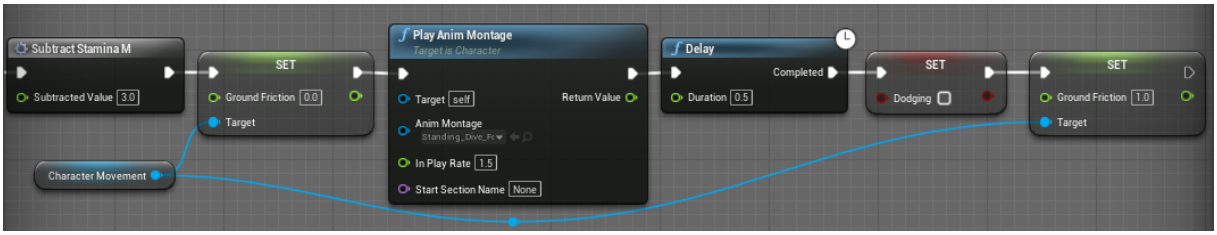
Slika 19: Logika Skakanja (Izvor: vlastita slika iz alata)

Logika animacije izbjegavanja najkompleksnija je animacija kretanja za implementiranje. Za početak treba uzeti „InputAction Dodge“ koji je vezan za tiplu Left Alt, nakon toga se već standardno provjerava je li karakter mrtav i je li spreman za radnju, ako je to zadovoljeno sljedeće se provjerava da li je lik već i procesu izbjegavanja preko varijable Dodging, ako nije sljedeće se provjerava da li igrač daje smjer u koji vi se htio kretati tako da se provjere varijable Forward i Right Axis Value, ako jedna od njih nije jednaka nuli možemo nastaviti. Sljedeće se mora provjeriti ima li igrač dovoljno energije da izvede izbjegavanje preko macroa za provjeravanje energije „Check Stamina M“ čija će logika biti objašnjena kasnije. Kao ulaz u makro postavi se minimalna potrebna energija s vrijednosti 0.5 tako da ako ima barem malo energije može izvršiti izbjegavanje te može ići u negativnu vrijednost, te će tako igrač dobiti efekt da na trenutak nema energije i neće vidjeti da se energija vraća, to je iz razloga da se spriječi konstantno ponavljanje akcije izbjegavanja. Nakon što je utvrđeno da je ima dovoljno energije za izbjegavanje postavlja se varijabla izbjegavanja u stanje istine



Slika 20: Provjera može li se izvesti izbjegavanje (Izvor: vlastita slika iz alata)

Nakon što su prošle sve provjere može li se izbjegnuti i postavljeno je da se lik nalazi u stanju izbjegavanja oduzme se energija liku s macrom „Subtract Stamina M“ u vrijednosti 3 čija će logika biti pojašnjena kasnije, zatim se postavi „Ground Friction“ na 0 tako da se dobije lijepo klizanje po površini za vrijeme izvođenja animacije izbjegavanja nakon toga se uz „Play Anim Montage“ izvrđi animacija izbjegavanja, postavi se „Delay“ od 0.5 sekundi koji je otprilike vrijeme izvođenja animacije no nije cijelo tako da lik ne bude u stanju izbjegavanja za cijelo vrijeme izvođenja animacije što će natjerati igrača da pripazi u kojem trenutku će krenuti izbjegnuti napad. Nakon što je delay završio postavimo varijablu izbjegavanja u stanje laž vratimo „GroundFriction“ na izvornu vrijednost.



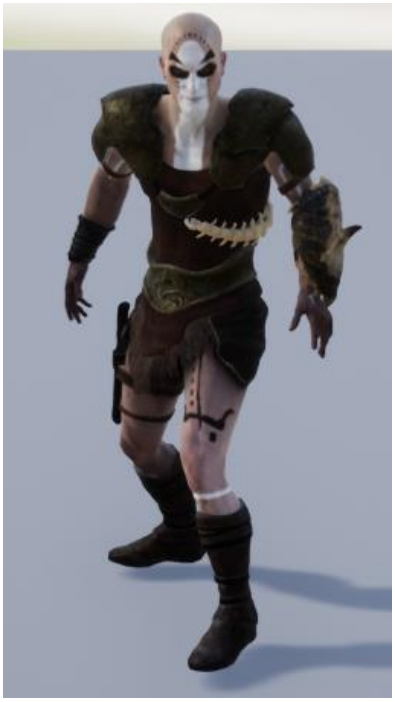
Slika 21: Izvođenje izbjegavanja (Izvor: vlastita slika iz alata)

4.2.2. Neprijatelji

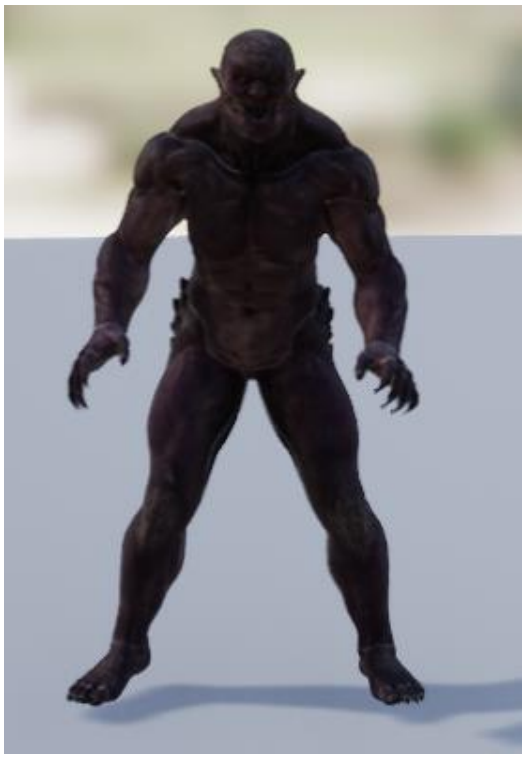
Proces postavljanja animacije za neprijatelje vrlo je sličan kao i za igrivog karaktera, no pošto su neprijatelji imaju manje mogućnosti koje će moći raditi u igri (hodanje,čučanj, skok, sprint) te pošto neće trčati dijagonalno već ravno prema zadanom cilju broj animacija je znatno manji.

4.2.2.1. Izgled neprijatelja

Slika 22: Izgled prvog neprijatelja (Izvor: vlastita slika iz alata)



Slika 23: Izgled drugog neprijatelja (Izvor: vlastita slika iz alata)



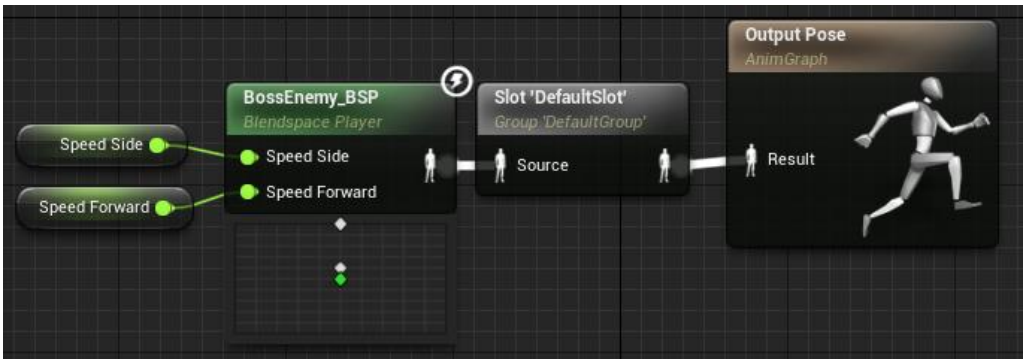
Slika 24: Izgled glavnog neprijatelja (Izvor: vlastita slika iz alata)

4.2.2.2. Animation Blendspace

Kreirat će se BlendSpace za neprijateljskog karaktera te postaviti maksimalna horizontalna i vertikalna brzina s minimalnim vrijednostima -500 i maksimalnim vrijednostima 500 tako da će neprijatelj biti nešto sporiji od našeg igrivog karaktera. Zatim se postave željene animacije na vrijednosti mreže.

4.2.2.3. Animation Blueprint

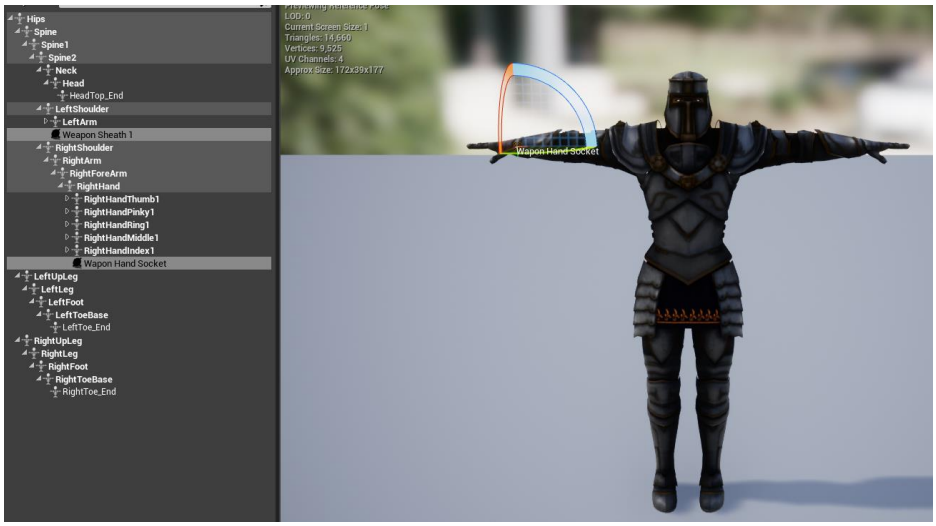
Zatim se kreira Animation Blueprint za neprijatelja te se postavi ista logika kao i za igrivog osim u slučaju pozivanja BlendSpace-a, pošto je logika animacija neprijatelja jednostavnija od igrivog karaktera, potrebno je samo uzeti varijable „Speed Side“ i „Speed Forward“ postaviti ih u BlendSpace uzeti „Default Slot“ i poslati u „Output Pose“.



Slika 25: Uključivanje BlendSpace-a neprijatelja u Animation Blueprint (Izvor: vlastita slika iz alata)

4.3. Implementacija borbe

Kako bi se implementiralo napadanje svih likova treba se postaviti dva nova „socketa“ na skeleton lika, socketi su specifična mjesta na skeletonu koja se mogu koristiti u različite svrhe u kodu. Jedan socket će se kreirati na lijevom ramenu lika dok će se drugi kreirati u desnoj ruci lika. Ti socketi će služiti kao poveznica lika s objektom mača, te će se time dobiti situacija u kojoj ako se ne koristi oružje, bit će na leđima odnosno na desnom ramenu lika, i time kako god se kreće lik, oružje će biti na tom mjestu. Jednako kao i na lijevom ramenu, na desnoj ruci će soket služiti kao poveznica s oružjem te koju god animaciju da lik izvodio mač će mu biti u desnoj ruci. Kada se kreiraju socketi, trebama stvoriti dva „Scene Componenta“ kao djecu Mesha te im dodijeliti nove sockete koji su stvoreni tako da postoji referenca prema njima.



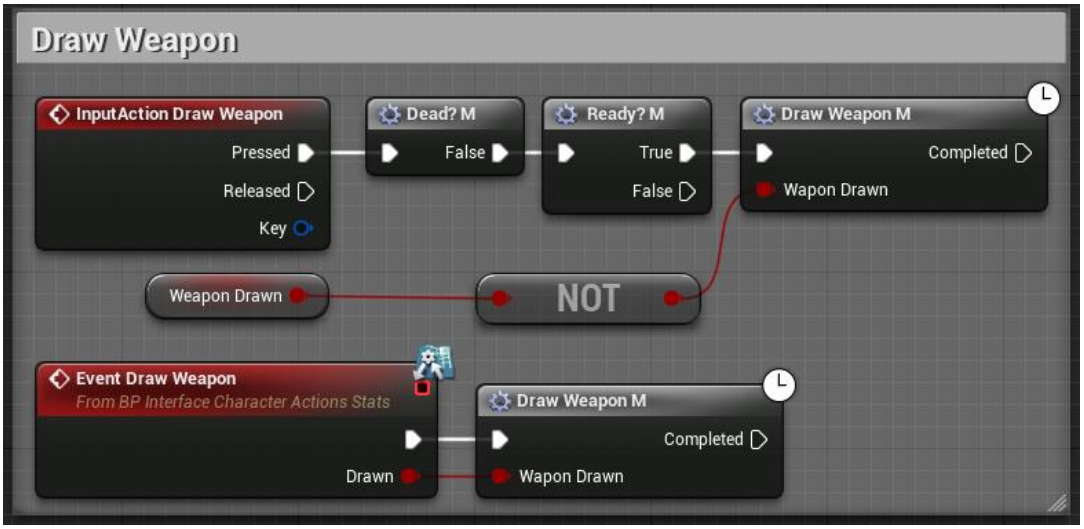
Slika 26: Pozicija Socketa na liku (Izvor: vlastita slika iz alata)

Nakon što postoje mjesta na kojima će biti postavljeno oružje, treba uvesti oružja u projekt, oružja koja će se koristiti u projektu bit će preuzete kao asset paket iz Epic Games Lunchera, ime tog paketa oružja je „Infinity Blade: Weapons“ [7]. Kako bi se paket dodao u igru, u Launcheru je potrebno pronaći paket u biblioteci te samo pritisnuti tipku „Add To Project“ i izabrati u koji projekt treba biti ubačen paket, nakon što je odabran projekt, paket će biti preuzet i pojavit će se u projektu.

Sada kada u projektu postoje oružja potrebno je kao dijete na dodani „Scene Component“ dodati kao dijete „Static Mesh Component“ kojemu će se kao postaviti željeno oružje kao „Static Mesh“ te će se postaviti kolizija kao „WorldStatic“ te će biti odabran „Overlap“ za sve opcije.

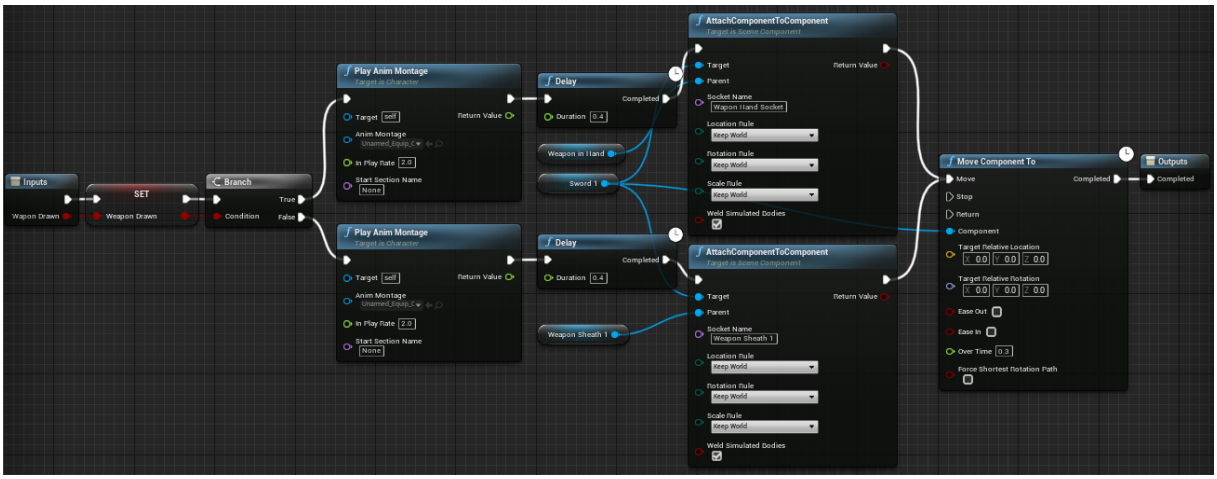
Uz to treba kreirati varijable koje će se koristiti u logici izvođenja animacija: „WeaponDrawn“(Boolean), „AttackCount“(Integer), „LockedOn“(Boolean), „LockedOnTo“(Actor), „PartyNumber“(Integer), „IsBlocking“(Boolean), „HitRange“(Float), „HitAnimation“(Instance Editable AnimMontage), „StaminaPoints“(Instance Editable Float – default 10), „StaminaPointsMax“(Instance Editable Float – default 10), „HealthPoints“(Instance Editable Float – default 10), „HpMax“(Instance Editable Float – default 10)

Kako bi simulirali pokrete izvođenja izvlačenja/spremanja oružja koristimo „InputAction WeaponDrawn“ koji je povezan na tipku 1, iz nje provjerimo je li lik mrtav i spreman izvršiti radnju, ako je zadovoljeno, uzimamo varijablu „WeaponDrawn“ te joj okrenemo vrijednost te pošaljemo u macro izvlačenja oružja „Draw Weapon M“. Isto i s eventom iz sučelja za izvlačenje oružja.



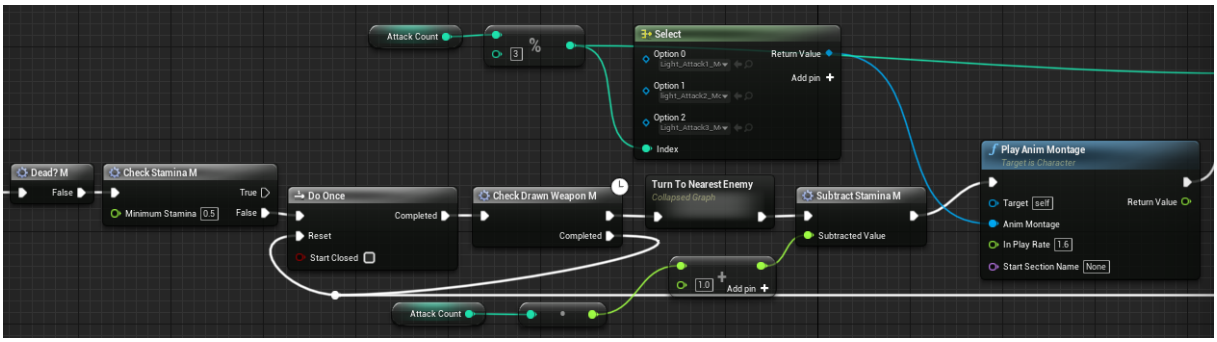
Slika 27: Logika Izvlačenja/spremanja oružja (Izvor: vlastita slika iz alata)

Macro izvlačenja oružja ima jedan ulaz a to je boolean vrijednost, ta vrijednost se sprema u varijablu „WeaponDrawn“ te ovisno o njoj imam pokretanje iste animacije sa delayom od 0.4 sekunde, no je li varijabla u stanju istine ili laži zove se „AttachComponentToComponent“. Time ćemo prebaciti mač iz jednog socketa u drugi, u situaciji istine kao parent postavi se socket ruke s targetom objektom oružja koji je na liku, u slučaju laži parent je socket lijevog ramena. Nakon toga oba „AttachComponentToComponent“ će se spojiti u „Move Component To“ kojemu ćemo za component postaviti objekt oružja te će iz ovog nodea ići na izlaz macroa.



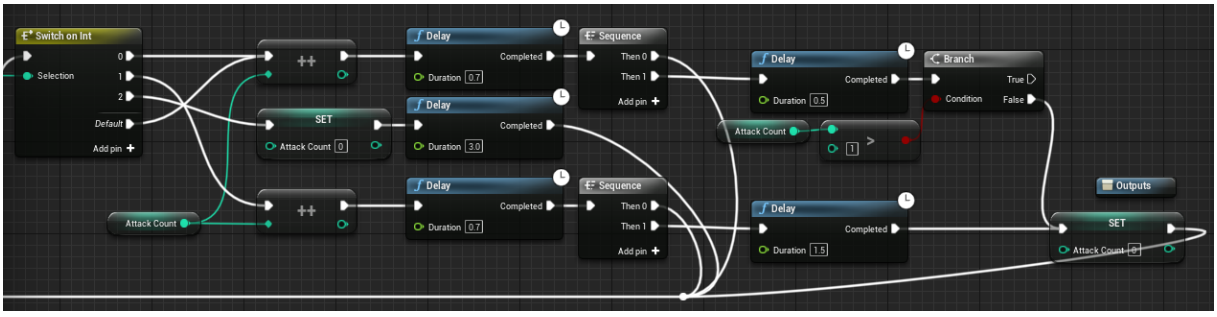
Slika 28: Macro izvlačenja oružja (Izvor: vlastita slika iz alata)

Kako bi se postigla mogućnost napadanja trebamo uzeti „InputAction Light Attack“ koji je vezan uz lijevu tipku miša, zatim se provjeri je li karakter mrtav, ako nije provjeri se ima li dovoljno energije za napad preko marcroa „Check Stamina M“ s minimalnom vrijednosti 0.5, zatim odradimo jednom provjeru je li izvučeno oružje preko macroa „Check Drawn Weapon M“ taj macro je jednostavan, koristi varijablu „Weapon Drawn“ te ako je postavljana na istinu ništa se ne dogodi, ako nije istinita pokreće macro „Draw Weapon M“ da izvuče mač i postavlja varijablu na istinitu vrijednost te vraća informaciju da je završio s izvlačenjem oružja. Ako je vratio povratnu vrijednost da je izvukao oružje vraća se ponovo na resetira se kod. Ako je prošao ispitivanje je li oružje izvučeno izvršava kod za okretanje prema najbližem neprijatelju koji će biti pojašnjen kasnije, zatim preko macroa „Subtract Stamina M“ smanjuje energiju na varijabli „StaminaPoints“ ovisno koji je broj napada uvećan za 1. Nakon smanjenja energije pokreće animaciju napada koju dobije ovisno koji je trenutni modulo 3 od broja napada te se koristi select s tri opcije koje od kojih svaka ima jednu animaciju napada.



Slika 29: Logika provjere može li se napadati i pokretanje animacije (Izvor: vlastita slika iz alata)

Nakon što se izvršila animacija napada treba se povećati brojač napada. Za to se koristi „Switch on Int“ koji ovisno o modulo koji se koristio za određivanje animacije pokreće različite radnje. Ako je modulo 0 broj napada se povećava za 1 te se nakon toga pričeka 0.7 sekundi da bi se mogao dati ponovni unos jer se resetira „Do Once“ zatim nakon pola sekunde (time simbolizira vrijeme kojim se može raditi kombinacija napada) provjerava je li broj napada veći od 1 te ako je vraća ga na 0. Ako je modulo vrijednosti 1 postavlja broj napada se povećava za 1 čeka se 0.7 sekundi te se resetira „Do Once“ te se čeka još 1.5 sekunde i broj napada padne na 0. Ako je modulo vrijednosti 2 broj napada se smanjuje na 0 čeka se 1.5 sekunde i resetira se „Do Once“.

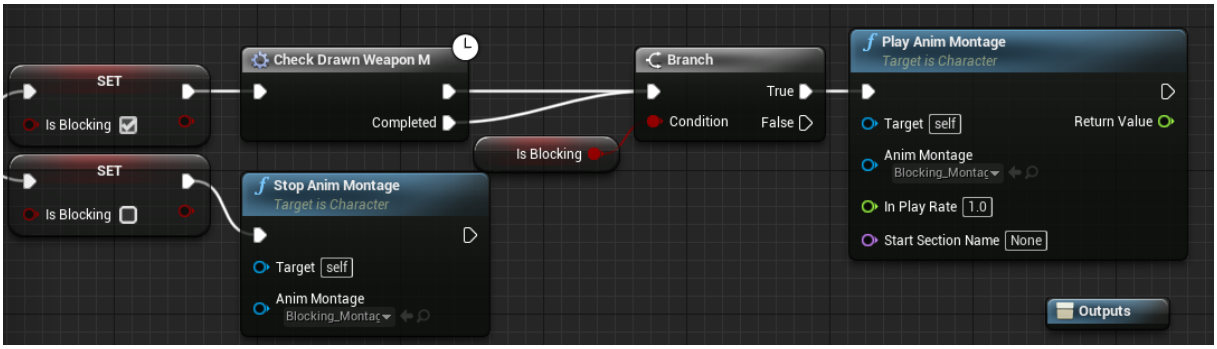


Slika 30: Logika brojača izvršenih napada u kombinaciji (Izvor: vlastita slika iz alata)

Za napade povezane s desnom tipkom miša kod je jednaki kao i za napade lijeve tipke miša s razlikom da započinje s „InputAction Heavy Attack“ i Select ima različite animacije.

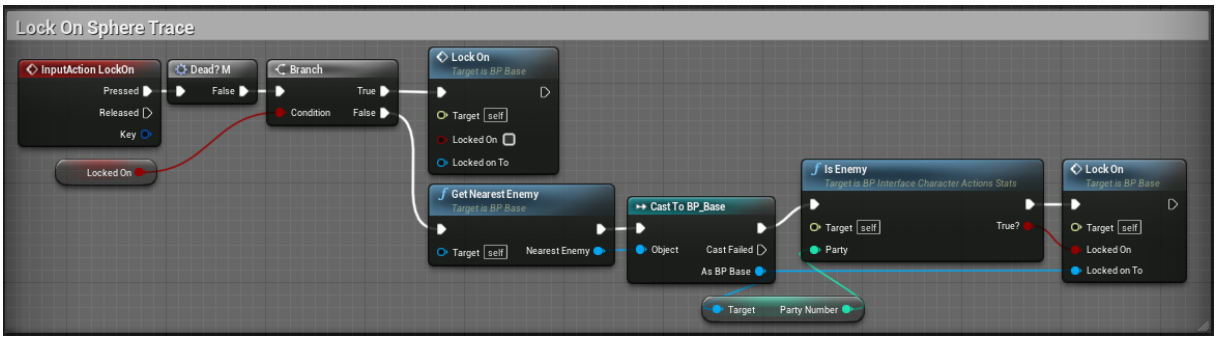
Kako bi se karakter mogao braniti od napada, za „InputAction Block“ koji je vezan za tipku Q, kada je tipka pritisnuta provjeri se je li karakter mrtav i je li spreman, ako je zadovolji uvjete postavlja varijablu „IsBlocking“ istinitu vrijednost, provjeri preco macroa „Check Drawn Weapon M“ je li oružje izvučeno te ako nije gas izvlači, nakon toga ako je još uvijek istinito

„IsBlocking“ pokreće se animacija za blokiranje. Ako se tipka prestane držati „IsBlocking“ se postavlja na laž i prestaje se izvršavati animacija.



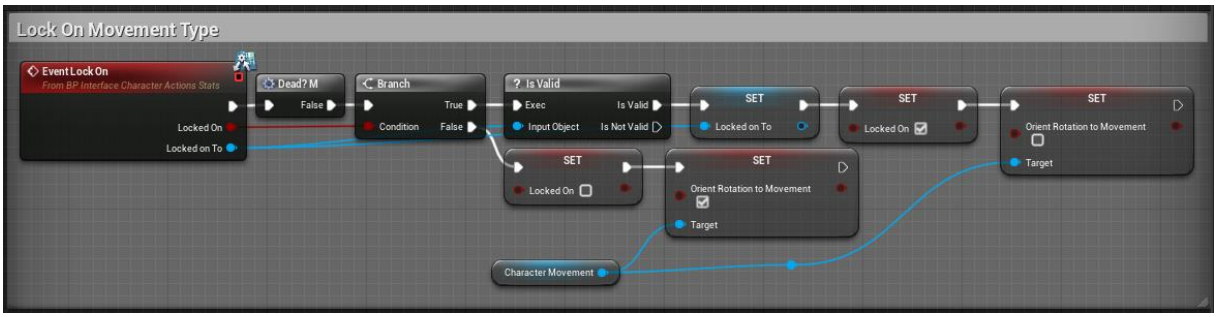
Slika 31: Logika izvršavanja obrane (Izvor: vlastita slika iz alata)

Bitna mehanika u akcijskim igrama je mogućnost fokusiranja na jednog neprijatelja. Kada se pritisne „InputAction LockOn“ koja je vezana na srednju tipku miša, prvo se provjeri je li karakter mrtav, ako nije provjeri se je li „LockedOn“ varijabla istinita, ako je zove Event LockOn sa željom da se poništi fokusiranje. Ako „LockedOn“ nije istina funkcija „Get Nearest Enemy“ (više o funkciji kasnije) daje najbližeg neprijatelja te se on pretvori u objekt „BP_Base“ i iz njega provjeri „Party Number“ igrač ima jedan, neprijatelji drugi, te se preko njega provjeri je li najbliži lik neprijatelj, te se zove event LockOn i postavlja se istina ili laž ovisno je li neprijatelj.



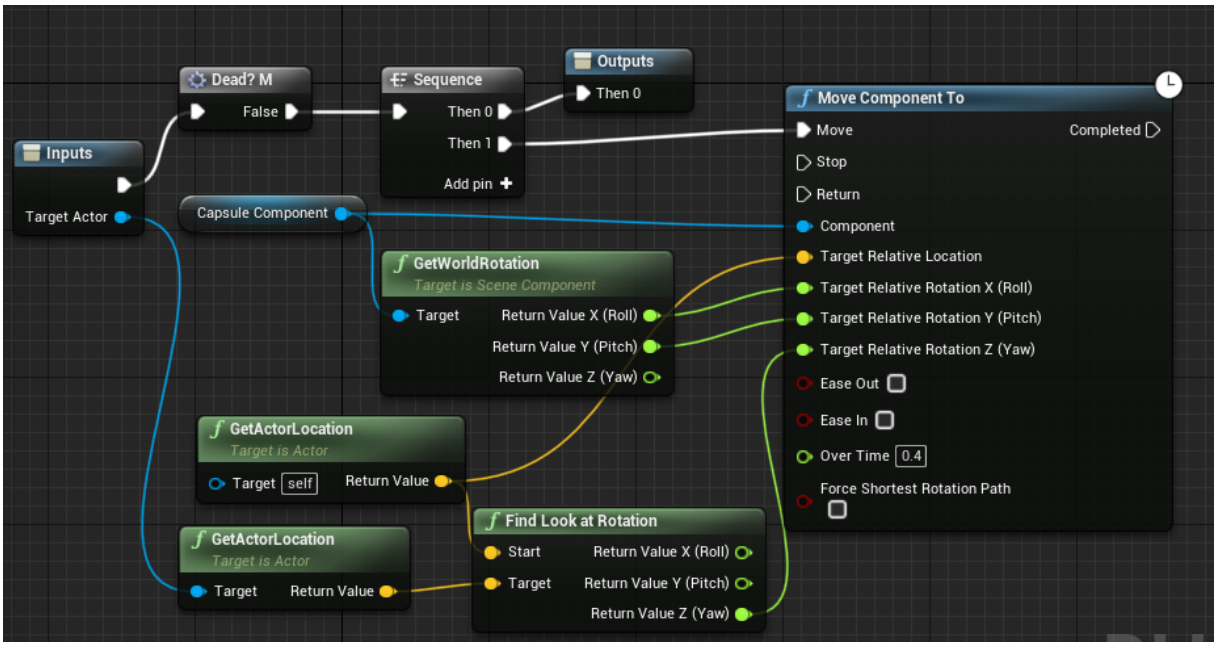
Slika 32: Logika uključivanja fokusiranja na neprijatelja (Izvor: vlastita slika iz alata)

Kada je pozvan event LockOn provjeri se je li lik mrtav, ako nije provjeri se je želi li se fokusirati na neprijatelja ili ne, ako se želi i actor na kojeg se želi fokusirati je validan postavi se varijabla „LockedOnTo“ s tim actorom, „LockedOn“ se postavi na istinu i isključi se orijentacija rotacije prema kretanju, ako se ne želi fokusirati „LockedOn“ se postavi na laž i postavi se orijentacija prema kretanju.



Slika 33: Event fokusiranja (Izvor: vlastita slika iz alata)

Da bi se postiglo okretanje prema drugom liku napravljen je macro „Turn To Actor M“. Prvo provjerava je li lik mrtav, ako nije radi sekvencu, prvo pošalje signal prema izlazu, drugo zove funkciju „Move Component To“ koja uzima „Capsule Component“ lika koji je u suštini cijeli lik, uzima njegovu rotaciju za X i Y koji se proslijede u funkciju, te uzima lokaciju neprijatelja i prema svojoj lokaciji izračuna Z rotaciju te koja se također proslijedi u funkciju.



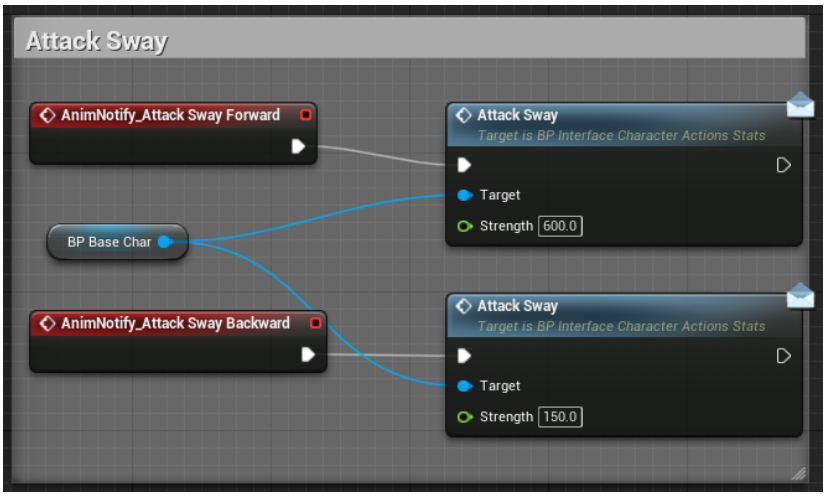
Slika 34: Mactro „Turn to Actor M“ (Izvor: vlastita slika iz alata)

Kod napadanja, treba zaustaviti mogućnost kretanja. To će se postići tako da se na „AnimMontageu“ za zasebne animacije postavi „notifier“ koji je će u određenom trenutku animacije poslati signal da se izvrši neki dio koda. Na animaciju će se sada postaviti 2 notifiera jedan za prekid kretanja te jedan koji će omogućiti kretanje. Nakon što su postavljeni notifieri, u „AnimationBlueprintu“ će se koristiti „AnimNotify_EnableMovement“ i „AnimNotify_DisableMovement“ koji će se upaliti na trenutak kada je postavljen notifier u

animaciji. Nakon toga se zove funkcija iz sučelja kojoj se prosljeđuje „Character“ za kojeg želimo da se to izvrši.

Slika 35: Zaustavljanje kretanja uz pomoć notifiera (Izvor: vlastita slika iz alata)

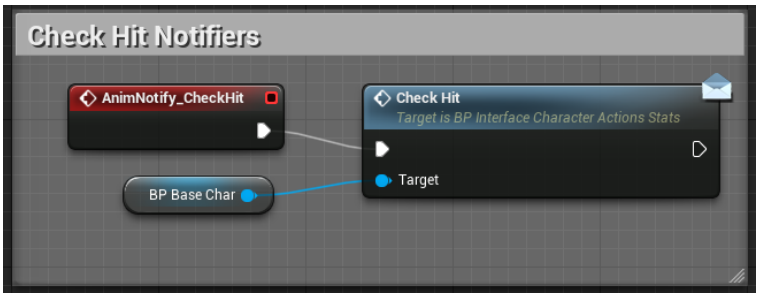
Da bi napadanje imalo dobar osjećaj, u napad će biti dodan lagani impuls koji će lagano pomaknuti lika prema naprijed u jednom trenutku napada pa nešto manje prema nazad kada na kraju napada. Za to će se opet koristiti notify sistem. U „Animation Blueprintu“ pozove se notify, te se proslijedi u event sučelja „Character“ za kojeg se to želi izvršiti sa snagom impulsa za kojeg će se pomaknuti.



Slika 36: Lagani pomak u napadanju preko notifiera (Izvor: vlastita slika iz alata)

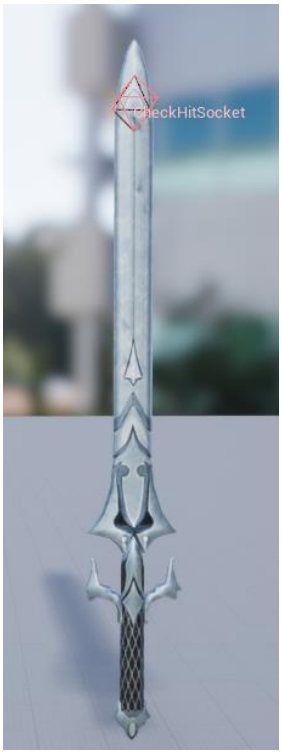
Zatim se u baznom blueprintu pozove taj event te se u funkciju „Add Impulse“ proslijedi „Character movement“ koji je kretanje lika i snaga impulsa pomnožena vektorom prema naprijed koji dobijemo iz „Capsule Componenta“ lika. Te će to pomaknuti lika za vrijeme animacije.

Sada kada su riješene animacije, kako bi borba funkcionirala treba napraviti koliziju između oružja i igrača/neprijatelja. Za to će se također koristiti notifier sistem. U određenom trenutku animacije postavit će se notifier za provjeru kolizije. U „Animation Blueprintu“ pozove se notifier za provjeru kolizije koji će proslijediti „Character“ event za provjeru kolizije.



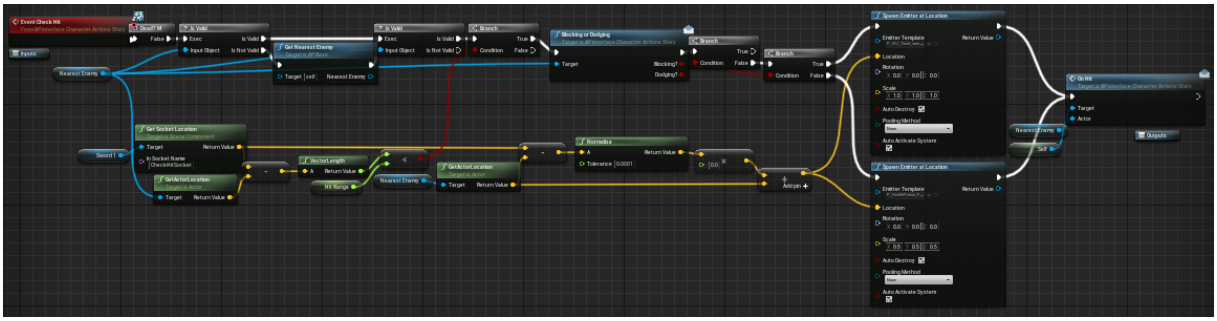
Slika 37: Notify za provjeru kolizije napada (Izvor: vlastita slika iz alata)

Sada se na mesh mača postavi socket koji će služiti za provjeru dogodi li se kolizija.



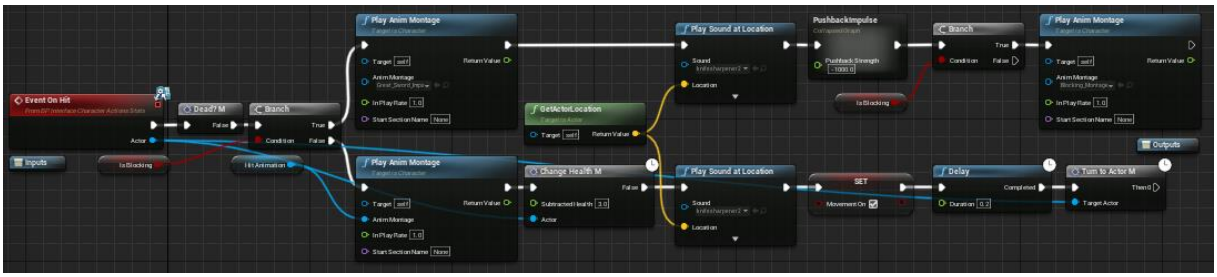
Slika 38: Socket za provjeru kolizije na oružju (Izvor: vlastita slika iz alata)

U baznom blueprintu kada je pozvan event za koliziju, prvo se provjeri je li karakter mrtav jer se ne želi pogađati mrtve neprijatelje, zatim se provjeri je li najbliži neprijatelj validan, ako nije postavi se preko funkcije „Get Nearest Enemy“ zatim se ponovno provjeri je li najbliži neprijatelj validan, ako je, provjeri se iz lokacije socketa mača i lokacije neprijatelja je li vektor razmaka u dovoljno udaljen da ga se udari, zatim se zove funkcija iz sučelja „Blocking or Dodging“ iz koje se dobije boolean vrijednosti da li se lik brani ili da li je u procesu izbjegavanja. Provjeri se prvo da li lik izbjegava, ako ne, provjeri se da li se brani. Ako se brani generira se efekt preko funkcije „Spawn Emitter at Location“ kako bi se dobi znak je li se lik obranio, ako se nije obranio preko iste funkcije generira se drugačiji efekt. Efekti Infinity Blade: Effects [8] su preuzeti u projekt iz kao i oružja preko Epic Games Lunchera. Nakon što je efekt generiran, poziva se event „OnHit“.



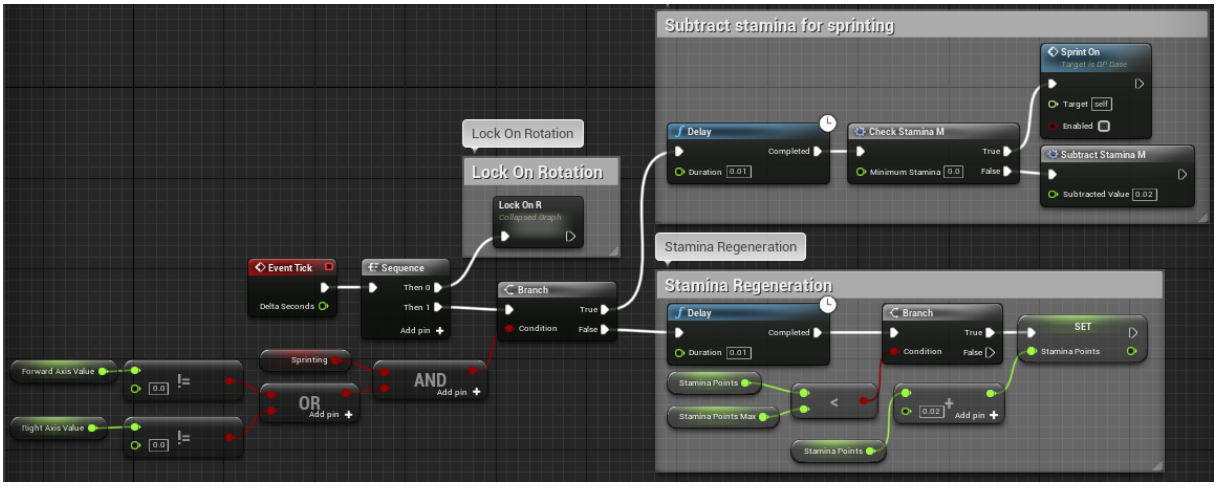
Slika 39: Kod provjere kolizije napada (Izvor: vlastita slika iz alata)

Kada je pozvan event „OnHit“ provjeri se je li lik mrtav, ako nije, provjeri se da li se brani, ako se brani pokreće se animacija pogođenog lika koji se brani te odmah nakon toga se pokrene zvuk da na lokaciji pogođenog lika, kako bi se signalizirala pogođenost, nakon toga se doda impuls koji će pomaknuti lika unatrag, te ako se još uvijek brani pokreće se animacija blokiranja. Ako se ne brani pokreče se animacija pogođenog lika iz varijable „HitAnimation“ koja je postavljana drugačija za svakog lika, nakon što se pokrene animacija pokreće se macro za smanjenja života (bit će objašnjen kasnije), nakon smanjenog života pokreće se zvuk za signalizaciju pogađanja, nakon se varijabla kretanja postavlja na istinito iz razloga ako je lik pogođen u animaciji napadanja prije nego što se ponovo vratila mogućnost kretanja, te se pričekaju 0.2 sekunde i zove se macro „Turn to Actor M“ za okretanje prema liku koji je nanio štetu.



Slika 40: Kod kada je postignuta kolizija (Izvor: vlastita slika iz alata)

Kako bi postojao sistem korištenja energije treba postaviti mogućnost regeneracije energije. Event Tick je event u pokretaču koji se izvršava svaki frameu u igri, nakon event ticka pozvat će se sequence, prvo će se izvršiti kod fokusiranja tako da će se svaki frame u igri postaviti lokacija neprijatelja ako je lik u stanju fokusiranja na neprijatelja. Sljedeće će se u sekvenci provjeriti da li igrač sprinta, ako sprinta provjerit će se preko macroa „Check Stamina M“ da li ima energije za sprint te ako nema isključit će se sprint ako ima energije zvat će se macro za smanjenje energije „Subtract Stamina M“ koji će smanjiti energiju liku u svakom svakih 0.01 sekundu za 0.02. Ako igrač ne sprinta, provjeri se da li je energija spremljena u varijabli „StaminaPoints“ manja od maksimalne energije koju vidimo u varijabli „StaminaPointsMax“, ako je, energija se povećava svakih 0.01 sekundu u varijabli „Stamina Points“.

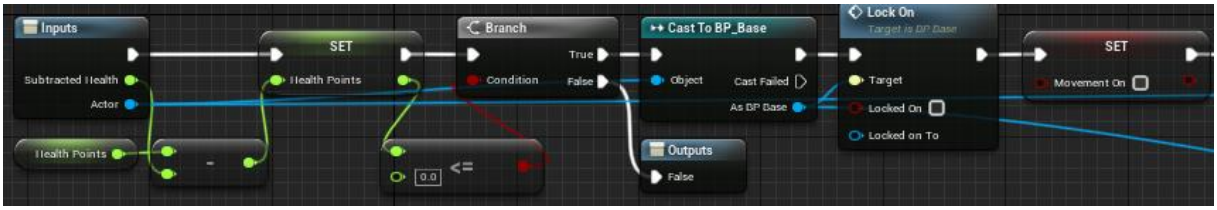


Slika 41: Kod sustava regeneracije energije (Izvor: vlastita slika iz alata)

„Change Health M“ je najbitniji macro u blueprintu jer se u njemu smanjuje život karaktera kada je to potrebno, te se izvršava procedura ako se život smanji na 0.

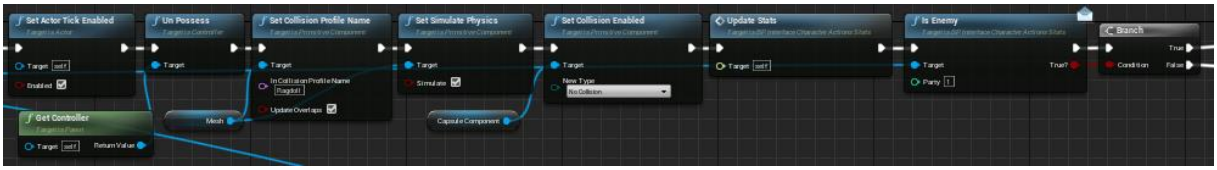
Na početku macroa, od života igrača u varijabli „Health Points“ oduzima se broj života koji je proslijeđen u macro, zatim se provjerava je li život na 0 ili manje, ako nije izlazi se iz macroa,

ako je, Actor se pretvara u BP_Base, isključuje se fokus na mrtvog lika i isključuje se kretanje lika.



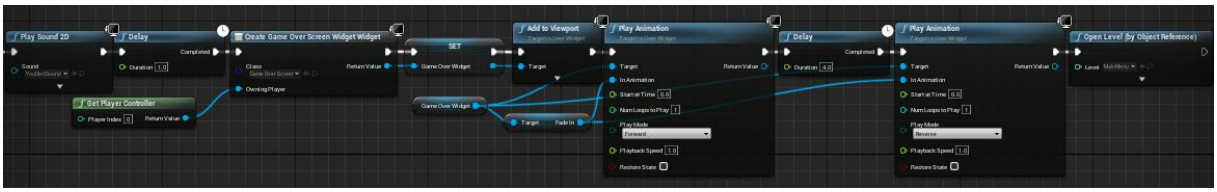
Slika 42: Prvi dio"Check Health M" macroa (Izvor: vlastita slika iz alata)

Zatim se omogući actor tick funkcijom „Set Actor Tick Enabled“, isključe se kontroliranje lika funkcijom „Un Possess“, funkcijom „Set Collision Profile Name“ pretvara se u lutku da više ne može stajati, zatim se funkcijom „Set Simulate Physics“ simulira fizika da karakter padne na pod, zatim se funkcijom „Set Collision Enabled“ isključuje kolizija karaktera, nakon toga se zove funkcija sučelja „Update Stats“ kako bi se promijenila traka života, pa se funkcijom „Is Enemy“ s ulazom party provjerava je li karakter igrač ili neprijatelj (1 simbolizira igrača), te se ulazi u granu ovisno je li istina ili laž.



Slika 43: Drugi Dio "Check Health M" macroa (Izvor: vlastita slika iz alata)

Ako je igrač umro pokrene se zvuk smrti [9] funkcijom „Play Sound 2D“ pričeka se 1 sekunda, kreira se widget koji će biti ekran smrti funkcijom „Create Widget“ i postavi se u varijablu „GameOverWidget“ i dodaje se u ekran funkcijom „Add to ViewPort“, iz varijable se uzima kreirana animacija widgeta ublježđivanja u ekran te se pokreće da bi se polako pojavio widget smrti u ekran, pričeka se 4 sekunde da se widget pokaže do kraja, zatim se ista animacija pokreće ali u suprotnom smjeru tako da polako nestane iz ekrana i funkcijom „Open Level (by Object Reference)“ se prebacuje na glavni izbornik igre.



Slika 44: Kod "Check Health M" macroa - smrt igrača (Izvor: vlastita slika iz alata)

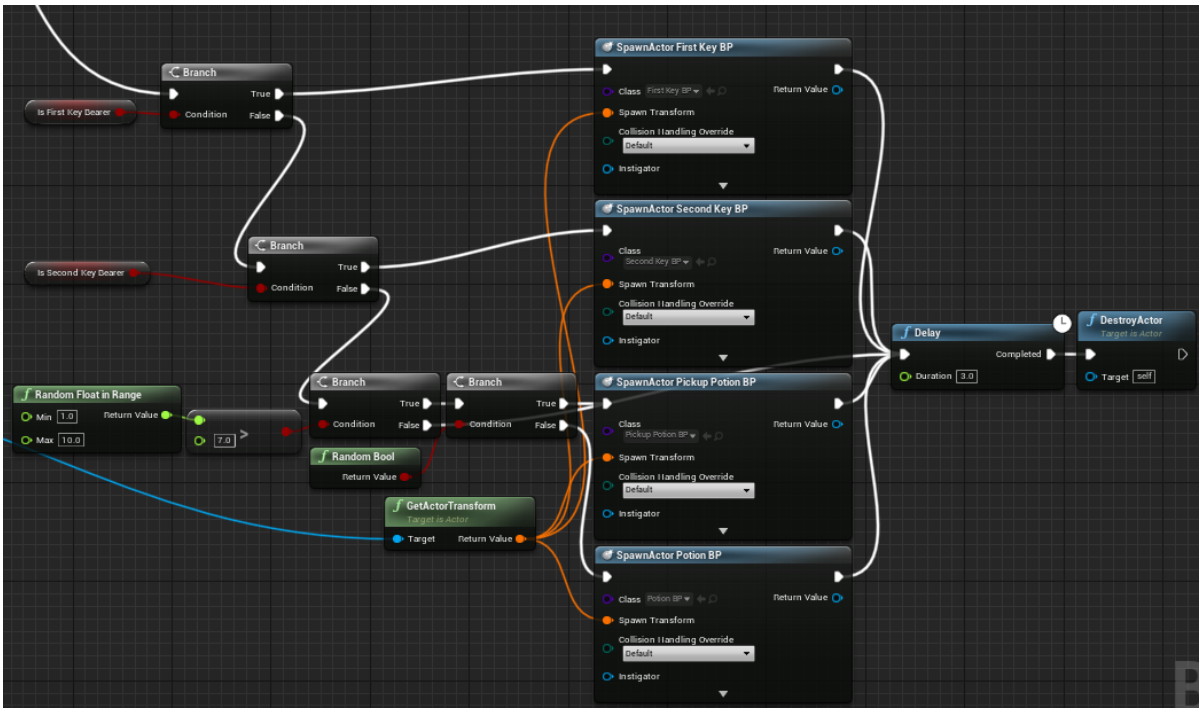
Ako je umro neprijatelj, provjerava se je li taj neprijatelj bio glavni neprijatelj, ako je proces je isti kao i za smrt igrača, uz razliku što se pokreće se zvuk pobjede [10] i widget namijenjen za pobjedu.

Slika 45: Kod "Check Health M" macroa - pobjeda igrača (Izvor: vlastita slika iz alata)

Ako je ubijen neprijatelj koji nosi prvi ključ, generirat će se objekt prvog ključa funkcijom „Spawn Actor“, te će se pričekati 3 sekunde i uništiti tijelo neprijatelja funkcijom „DestroyActor“.

Ako je ubijen neprijatelj koji nosi drugi ključ, generirat će se objekt drugog ključa funkcijom „Spawn Actor“, te će se pričekati 3 sekunde i uništiti tijelo neprijatelja funkcijom „DestroyActor“.

Ako je ubijen normalan neprijatelj, generira se broj između 1 i 10 te ako je taj broj veći od 7 generira se elixir, postoji 50% šanse da se generira eliksir koji se automatski aktivira i 50% šanse da se generira eliksir za kasnije korištenje. Eliksiri se generiraju funkcijom „Spawn Actor“. Ako je generiran eliksir ili ne, pričeka se 3 sekunde te se uništi tijelo neprijatelja funkcijom „DestroyActor“.



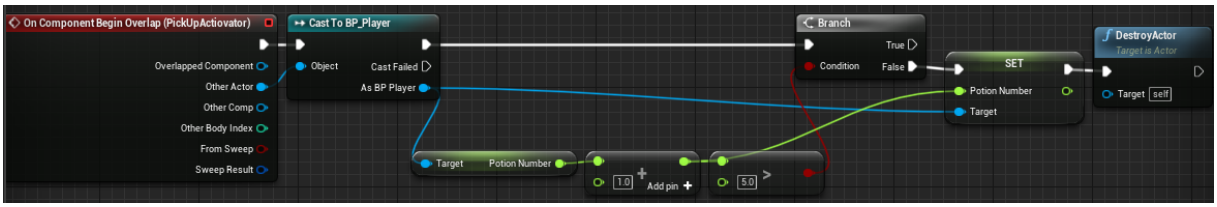
Slika 46: Kod "Check Health M" macroa -ubijen nositelj prvog ključa, drugog ključa i običan neprijatelj (Izvor: vlastita slika iz alata)

4.4. Eliksir

Kako bi eliksiri bili kreirani potrebni modeli koji bi bili prikazani u igri, Multistory Dungeons [11] koje će biti preuzete iz Epic Games Lunchera kao i prošli paketi ima modele eliksira koji će biti korišteni. Treba kreirati dvije blueprint klase za dvije vrste eliksira koje će biti u igri. U klasi se u „static mesh“ dodaje model koji se želi koristiti. Kada je to napravljenouz to treba se kao dijete „static mesha“ dodati „capsule collision“ kako mogla postojati interakcija između igrača i objekta eliksira.

4.4.1. Kasnije iskoristivi eliksir

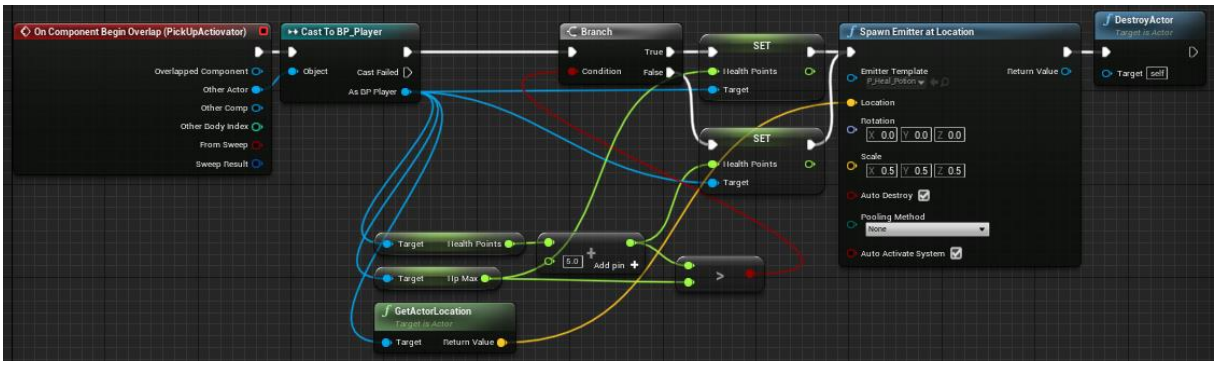
Kada dođe do dodira između igraća i eliksira pozvat će se „On Component Begin Overlap“, te će se od „Other Actora“ dobiti klasa BP_Player, provjerava se ima li igrač 5 eliksira, ako nema, broj njegovih eliksira se povećava za 1 i uništava se elixir na nivou.



Slika 47: Logika interakcije s kasnije iskoristivim eliksirom (Izvor: vlastita slika iz alata)

4.4.2. Odmah iskoristivi eliksir

Kada dođe do dodira između igraća i eliksira pozvat će se „On Component Begin Overlap“, te će se od „Other Actora“ dobiti klasa BP_Player, provjerava koji je trenutni život i maksimalan život igrača, trenutni život se poveća za 5 te ako je sada trenutni život veći od maksimalnog život mu postaje jednak maksimalnom životu. Nakon što se poveća život igrača generira se efekt koji će biti signal da je iskorišten elixir i uništava se elixir na nivou.



Slika 48: Logika interakcije s odmah iskoristivim eliksirom (Izvor: vlastita slika iz alata)

4.5. Nivo

Igra će imati dva nivoa. Prvi nivo će biti glavni izbornik gdje će igrač moći pokrenuti igru, pogledati kontrole i izaći iz igre. Drugi nivo će biti igrivi nivo na kojemu će igrač moći kontrolirati lika i kretati se po nivou i boriti se s neprijateljima. Font koji će se koristit će biti „Hemi Head“ [12].

4.5.1. Glavni izbornik

Da bi kreirali dizajn izbornika u svijet nivoa će se postaviti običan „plane“ objekt, na njega će se postaviti animacija stanja mirovanja lika, te će se u nivo dodati „pointlight“ koji će svijetliti u našeg lika, dodat će se dvije kamere (CineCameraActor) na različite pozicije, one će služiti u animacije izbornika. Te se dodaje objekt „AmbientSound“ koji će svirati pozadinsku

glazbu [13]. Kako bi mogli svirati glazbu treba tu glazbu pretvoriti u „Sound Cue“ to se radi tako da se klikne desni klik miša i „Create Cue“.

4.5.1.1. Widget početnog ekrana

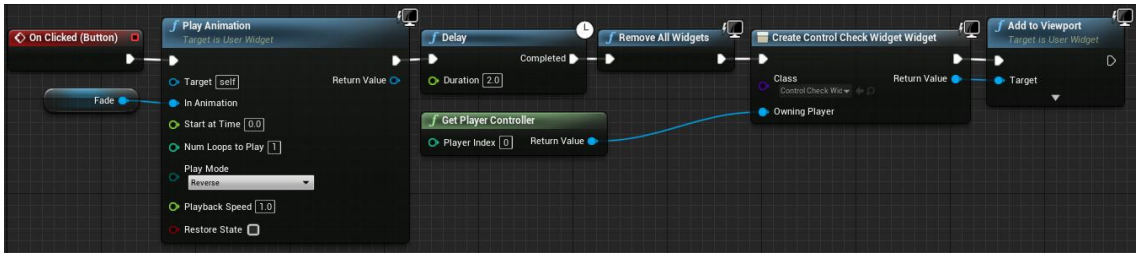
U ovom widgetu postavit će se naslov igre „Diplomski Rad“ kao teks i dodat će se tekst „Pritisni SPACE za Nastavak“.

Uz to će se kreirati dvije animacije, jedna će biti da widget lagano ublijedi u ekran, to će se postići tako da se selektira platno widgeta i doda se nova animacija „FadeIn“, animacija će biti dužine 1.5 sekunde na početku animacije postavi se „Render Opacity“ na 0 a na kraju na 1. Druga animacija će biti napravljena tako da se selektira drugi tekst te se doda animacija koja će trajati 3 sekunde, na početku će se u „Color and Opacity“ A vrijednost staviti na 0, na 1.5 sekunde bit će 1 i na 3 sekunde bit će 0 to će dati efekt titranja.

4.5.1.2. Widget Glavnog izbornika

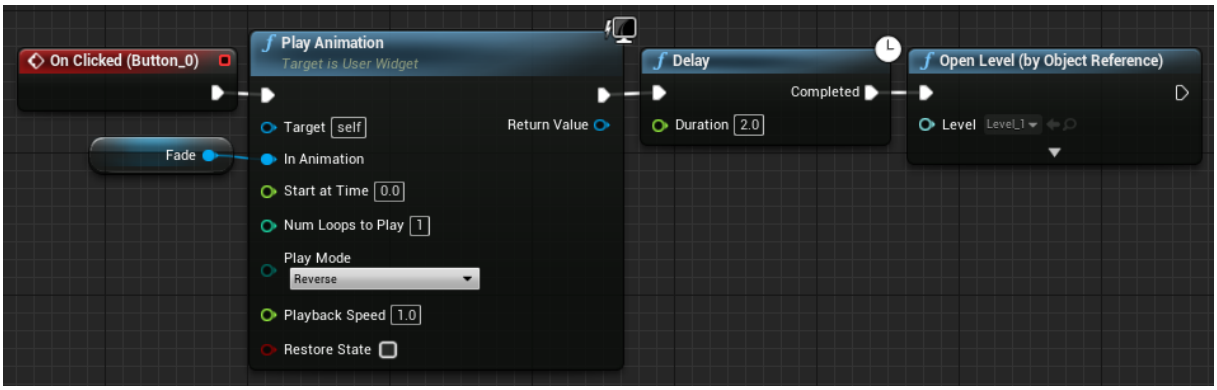
Ovaj widget imat će 3 tipke. Te animaciju „Fade“ koja će biti jednaka kako i animacija u prošlom widgetu.

Tipka „Provjeri Kontrole“ će na klik pokrenuti animaciju u obrnutom smjeru, pričekat će 2 sekunde da završi animacija, uništit će sve trenutne widgete, kreirati widget sa kontrolama i dodati ga na ekran.



Slika 49: Logika tipke "Provjeri Kontrole" (Izvor: vlastita slika iz alata)

Tipka „Pokreni Igru“ će na pritisak tipke pokrenuti animaciju u suprotnom smjeru, pričekati 2 sekunde i prebaciti igru na igrivi nivo.

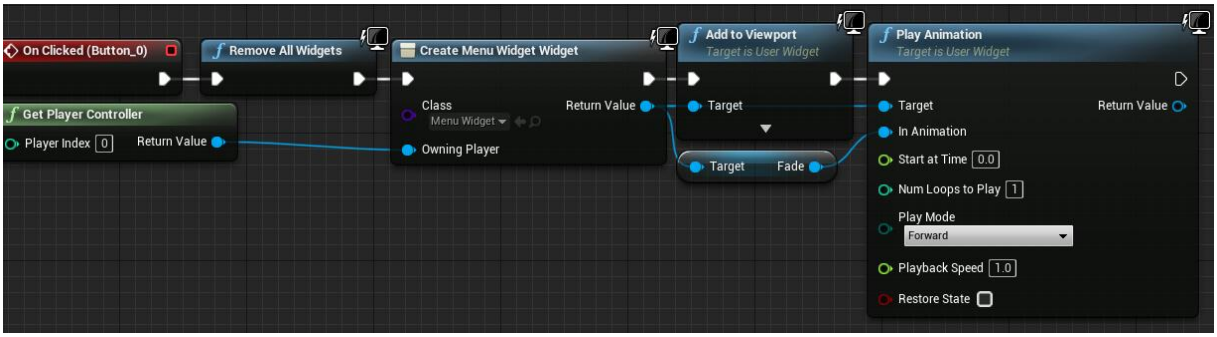


Slika 50: Logika tipke "Pokreni Igru" (Izvor: vlastita slika iz alata)

Tipka „Prekini igru“ će na pritisak tipke zvati funkciju „Quit Game“ koja gasi igru.

4.5.1.3. Widget Kontrola

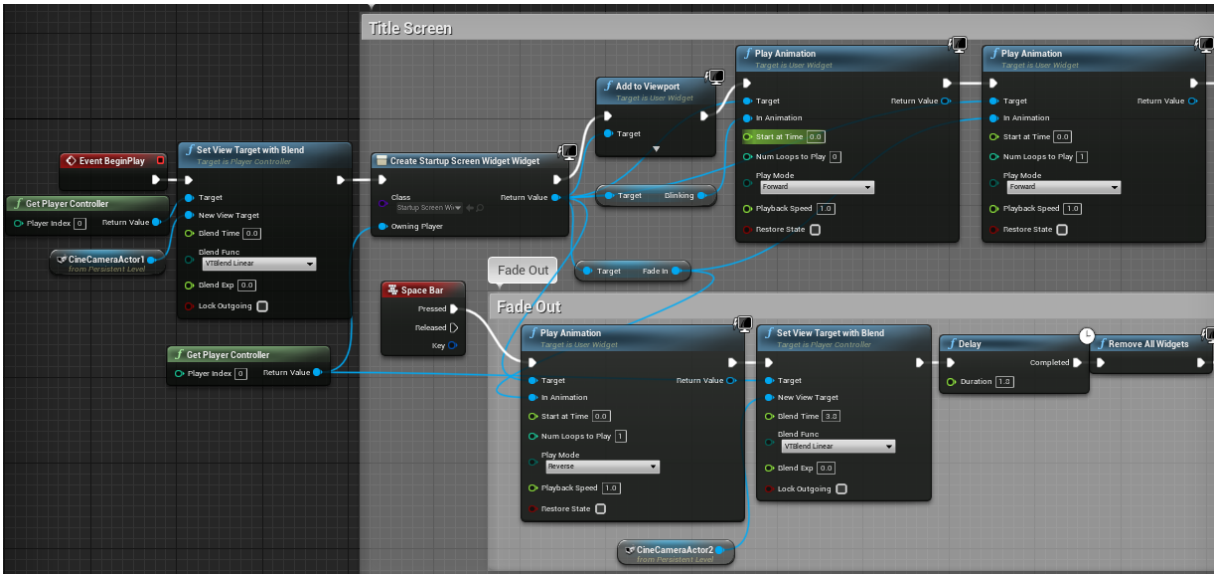
Ovaj widget će imati popis kontrola koje će biti dostupne u igri, te tipku „Back“ koja će na pritisak uništiti sve postojeće widgete, kreirati widget glavnog izbornika, dodati ga na ekran i iskoristiti njegovu fade animaciju da ga ublijedi na ekran.



Slika 51: Logika tipke "Back" (Izvor: vlastita slika iz alata)

4.5.1.4. Logika nivoa

Na „Event BeginPlay“ koji se pokreće kada se pokrene igra postavit će se funkcijom „Set View Target with Blend“ uz „Blend Time“ 0, preko „Player Controllera“ i prve kamere prizor na prvu kameru, zatim će se kreirati widget početnog ekrana te se dodati na ekran uz animacije treptanja i ublijeđivanja u ekran. Kada se pritisne tipka razmaknice, poziva se animacija ublijeđivanja ali u suprotnom smjeru, opet se poziva funkcija „Set View Target with Blend“ uz „Blend Time“ 3.8 i proslijeđenu drugu kameru , to će nam dati da kroz sljedeće 3.8 sekunde se prebacuje kontinuirano pogled iz jedne kamere na drugu. Zatim se pričeka 1.8 sekunde da prođe animacija i maknu se svi widgeti s ekrana. Nakon što je maknut widget kreira se novi widget glavnog izbornika i dodaje u ekran s animacijom ublijeđivanja.

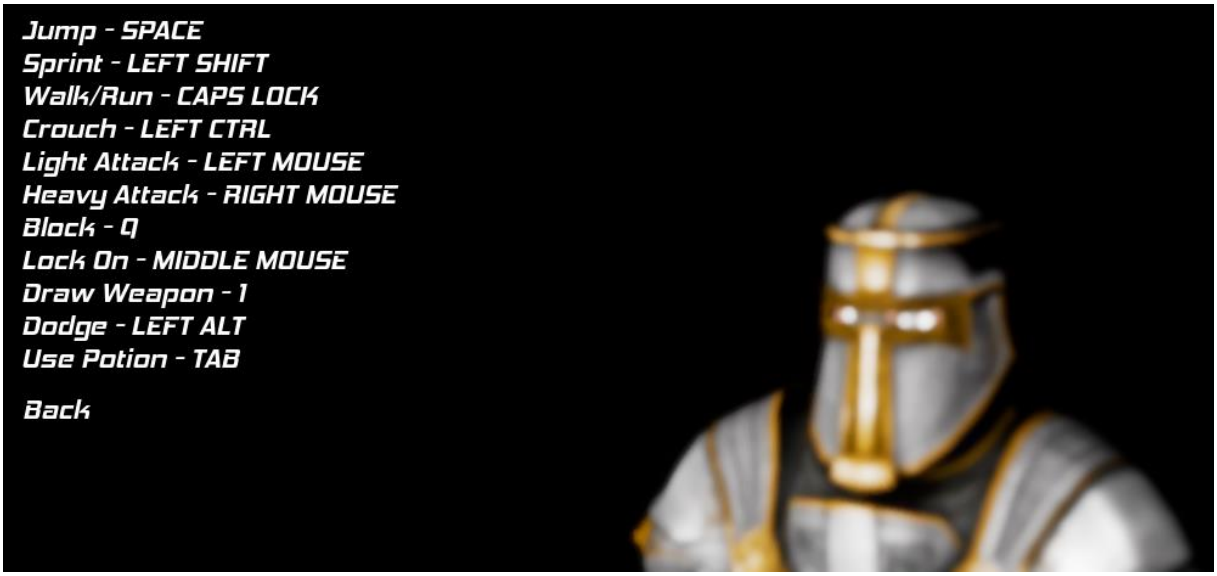


Slika 52: Pokretanje nivoa i logika početnog ekrana (Izvor: vlastita slika iz alata)

4.5.1.5. Izgled nivoa

Slika 53: Izgled početnog ekrana (Izvor: vlastita slika iz alata)

Slika 54: Izgled glavnog izbornika (Izvor: vlastita slika iz alata)

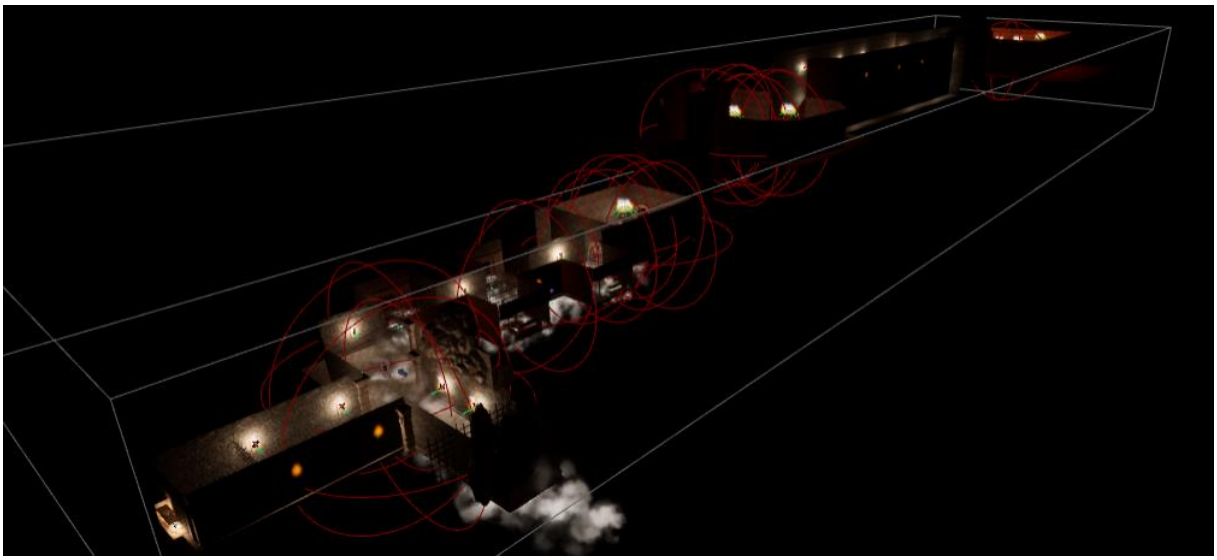


Slika 55: Izgled ekrana s pregledom kontrola (Izvor: vlastita slika iz alata)

4.5.2. Igrivi nivo

Igrivi bit će napravljen preko uz pomoć paketa Multistory Dungeons i Medieval Dungeon [14]. Nivo će imati nekoliko prostorija u kojima će se nalaziti neprijatelji koje će igrač morati pobijediti ako želi pobijediti. Na nivou će igrač moći pronaći nekoliko sakrivenih eliksira da mu pomognu riješiti nivo. Uz objekte koji će činiti nivo i dati mu izgled, nivo treba obuhvatiti s „NavMeshBoundsVolume“ koji će izračunati podlogu na kojoj će neprijatelji moći hodati.

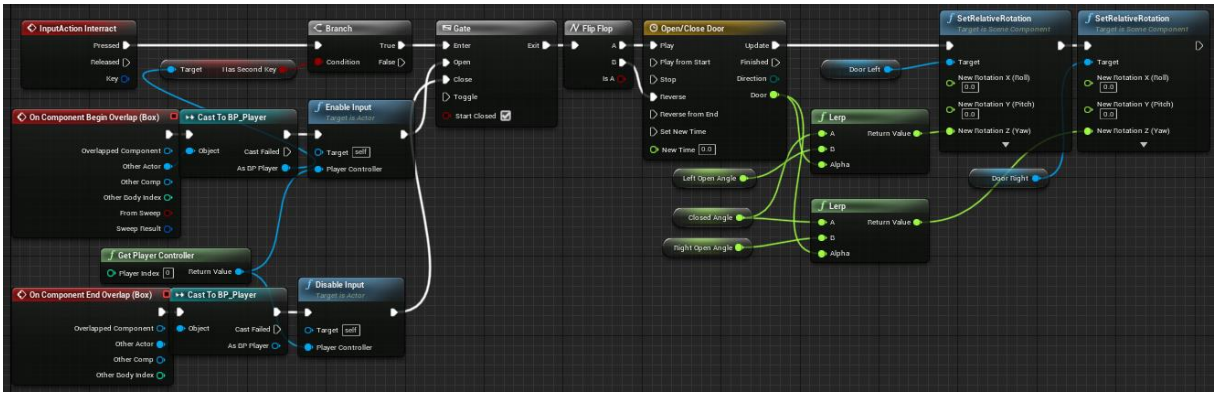
Na nivou neposredno prije nego se uđe u prostoriju s glavnim neprijateljem bit će postavljan jedan „Trigger box“ koji će pokrenuti glazbu za posljednju borbu [15].



Slika 56: Igrivi nivo (Izvor: vlastita slika iz alata)

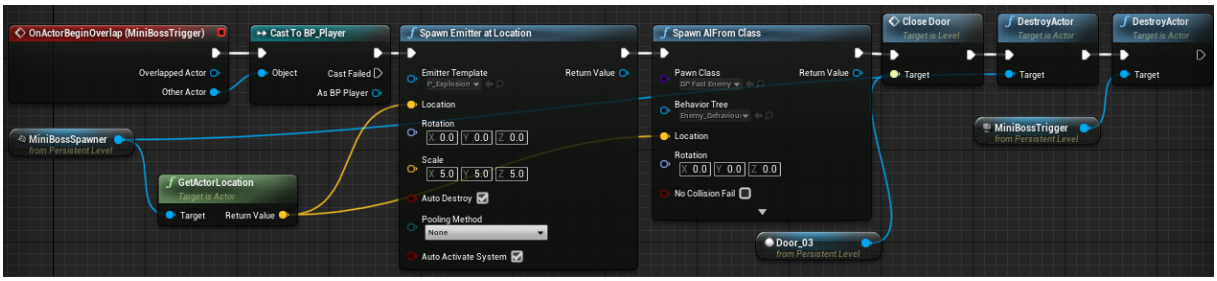
4.5.2.1. Animirana vrata

Blueprint vrata ima u sebi 2 različita krila koje se mogu otvoriti. Kada se pritisne tipka „Interract“ vezana za tipku E, i igrač se nalazi u „BoxCollisionu“ provjerava se da li igrač ima ključ te se odlazi timeline koji traje 3 sekunde i kroz te tri sekunde postavlja se nova relativna rotacija vrata.



Slika 57: Otvaranje i zatvaranje vrata (Izvor: vlastita slika iz alata)

Nakon što se otvore prva vrata odlazi se na drugi kat te kada se prođe kroz koridor i „Box Collision“ pozove se event koji stvoriti animaciju eksplozije i protivnika ispred igrača te će zatvoriti vrata iza igrača kako bi igrač bio zatvoren s neprijateljem i nebi mogao napredovati tako dugo dok ne pobijedi neprijatelja i ne dobije ključ za vrata.

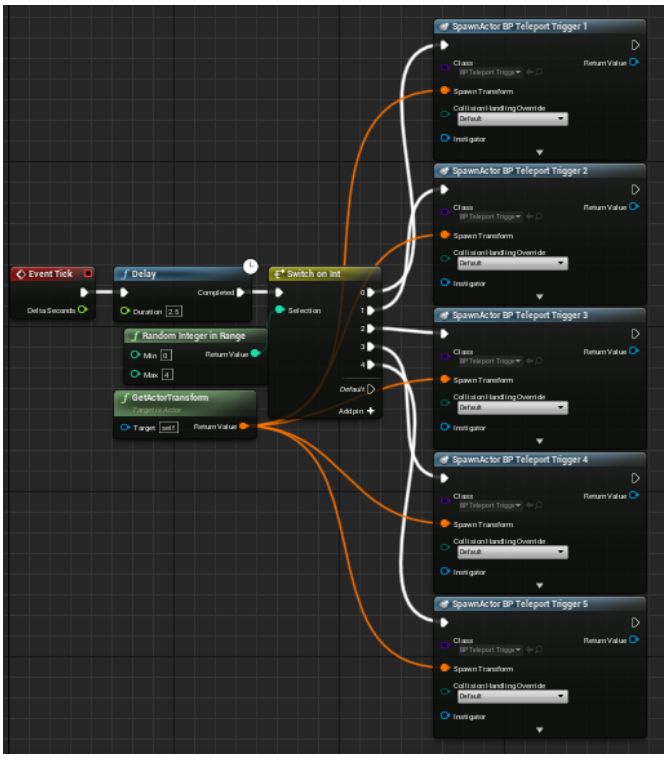


Slika 58: Stvaranje neprijatelja i zatvaranje vrata (Izvor: vlastita slika iz alata)

4.5.2.2. Platforme

Prije zadnjeg neprijatelja igrač će morati pokazati vještinu korištenja različitih kontrola kretanja. Igrač će morati ići na most na kojemu će prema njemu dolaziti različite platforme koje će morati preskočiti, uči u čučanj ili se izmaknuti kroz njih. Ako to ne uspije bit će teleportiran na početak mosta, pad s mosta također rezultira teleportacijom.

Kako bi različite platforme bile kreirane, postojat će blueprint objekt koji će biti zadužen za stvaranje platformi. Na „Event Tick“ će se prvo pričekati 2.5 sekunde te će se generirati nasumično broj između 0 i 4 te će se prema tom broju preko switch selekcije stvoriti jedna od 5 platformi. Uz to je pripremljen „BoxCollision“ koji će kada ga igrač dostigne isključiti generiranje platformi tako da se uništi objekt za generiranje.



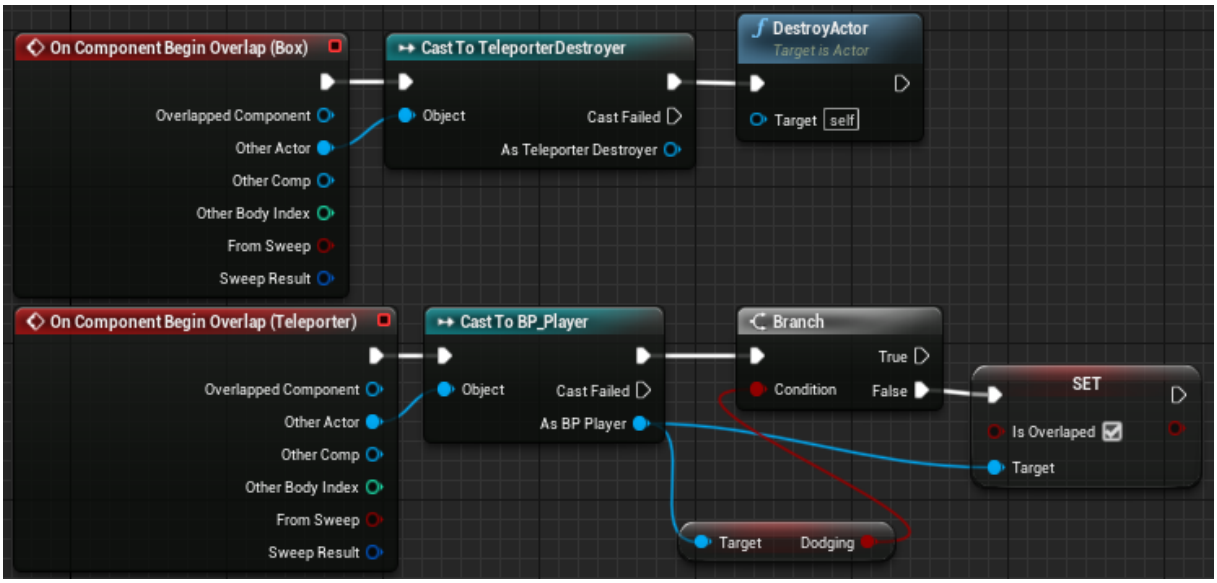
Slika 59: Generiranje platformi (Izvor: vlastita slika iz alata)

Sama platforma će u sebi imati logiku da na „EventPlay“ postavi početnu lokaciju, iz timeline-a dobije animaciju kamo treba putovati i koliko sekundi te se uz zbroj početne lokacije i dobivene direkcije postavi nova lokacija prema kojoj će platforma putovati.

Slika 60: Kretanje platforme (Izvor: vlastita slika iz alata)

Kako se nebi dogodilo opterećenje igre sa previše objekta platformi na početku mosta biti će postavljen objek koji ako platforma dođe u kontakt biti će uništena.

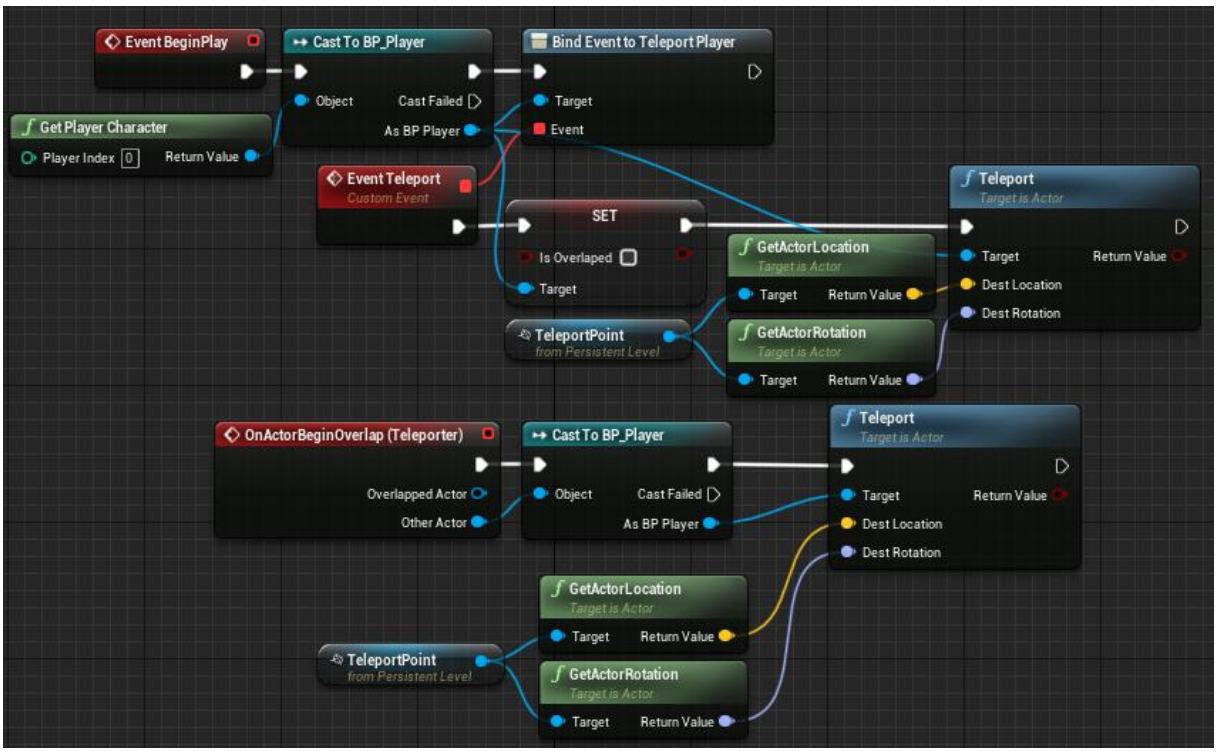
Ako platforma dodirne igrača postaviti će mu varijablu „IsOverlaped“ na istinu te će u klasi igrača biti pozvan „EventDispatcher“ koji će ga teleportirati na početak mape.



Slika 61: Uništenje platforme i postavljanje varijable igraču (Izvor: vlastita slika iz alata)

Sam nivo će na početku igre postavitipreko klase igrača „EventDispatcher“ koji će pokretati „Event Teleport“ kada igrača dodirne platforma. Taj event će postaviti igraču da varijablu „IsOverlaped“ na laž, te će teleportati igrača na dio mape gde je postavljen objekt

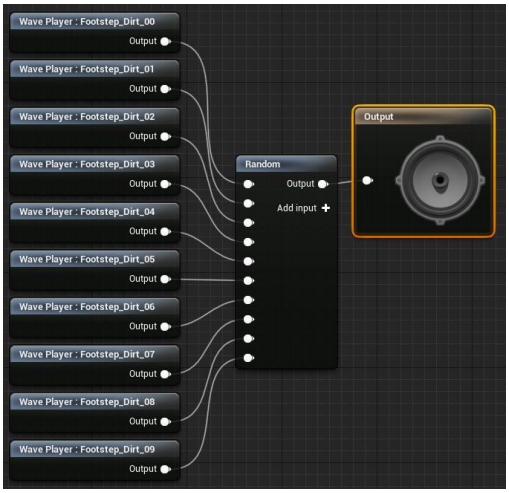
destinacije teleportiranja. Takodžer, kada igrač padne s mosta dodirnuti će „BoxCollision“ koji će ga teleportirati na destinaciju teleportiranja.



Slika 62: Teleportiranje igrača (Izvor: vlastita slika iz alata)

4.5.2.3. Zvukovi kretanja likova

Kako bi iskustvo igrača bilo bolje kreirat će se zvukovi koraka za likove. Za zvukove koraka [16] napravi se „sound cue“ koji će nasumično pokretati jedan od 10 različitih zvukova koraka kako bi se stvorio doživljaj da je svaki korak drugačiji.



Slika 63: Sound cue koraka (Izvor: vlastita slika iz alata)

Pozivanje zvukova bit će implementirano pomoću notifiera, na animaciji hodanja postavi se notify na trenutak kada noga dirne pod. U animation blueprintu se postavi se pozivanje eventa „Footestep Sound“ na notify koraka. U baznom blueprintu pozove se event te uz lokaciju karaktera se poziva funkcija „Play Sound at Location“ koja će pokrenuti „sound cue“ koraka.

4.5.2.4. Widgeti pobjede i poraza

Widgeti pobjede i poraza jednostavni su te jer imaju samo sliku na sebi i animaciju za ublijeđivanje. Pozivaju se u macru „Change Health M“.

Slika 64: Izgled ekrana poraza (Izvor: vlastita slika iz alata)

Slika 65: Izgled ekrana pobjede (Izvor: vlastita slika iz alata)

4.5.2.5. Widget igračevih statistika

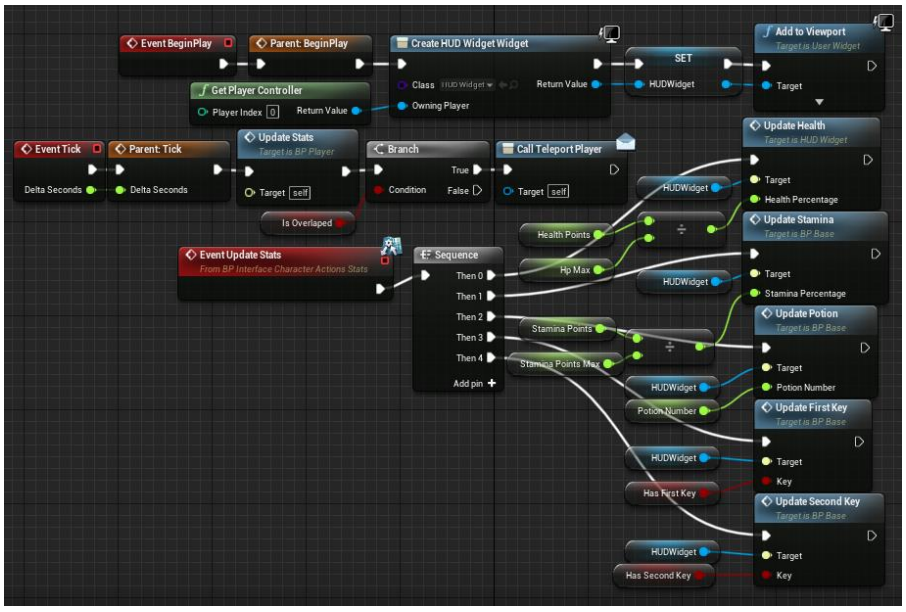
Kako bi igrač u svakom trenutku znao koliko mu je preostalo života i energije imat će na ekranu konstantno widget koji će mu u trakama simbolizirati kolko mu je još preostalo života, energije i eliksira. Životna traka, traka energije i broj eliksira bit će u gornjem lijevom kutu ekrana.

Slika 66: Widget igračevih statistika (Izvor: vlastita slika iz alata)

Kako bi widget konstantno mogao osvježivati statistiku igrača u widgetu pozivamo evente za osvježivanje te preko njih postavljamo vrijednosti koje su prikazane.

Slika 67: Logika osvježavanja statistika u widgetu (Izvor: vlastita slika iz alata)

Zatim u blueprintu igrača, koji je dijeti baznog blueprinta postavimo widget kod početka igre. Zatim svaki frame igre zovemo event „Update Stats“. Taj event će u sekvenci izračunati postotak života i pozvati događaj koji će u widgetu osvježiti život. Zatim će izračunati postotak energije i pozvati event koji će u widgetu osvježiti energiju. Zatim će pozvati event koji će u widgetu osvježiti broj eliksira. Zatim će pozvati event koji će widgetu pokazati da li da prikaže prvi ključ. Te se na kraju poziva event koji će pokazati drugi ključ ako ga igrač posjeduje.

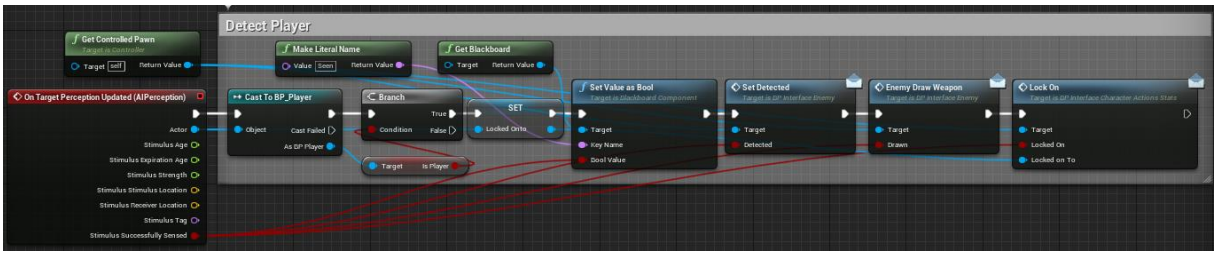


Slika 68: Prikaz i osvježavanje widgeta statistika od strane igrača (Izvor: vlastita slika iz alata)

4.6. Neprijatelj

4.6.1. Umjetna inteligencija neprijatelja

Kreira se blueprint klasu za AI, dodaje se „AIPerceptionComponent“ u klasu zatim se na event „BeginPlay“ pokrene „Behavior Tree“. Zatim kreiramo „On Target Perception Updated“ te iz njega i Controlled Pawna dobije se igrač, zatim ako je igrač se postavi na varijabla „Locked Onto“ koja simbolizira karaktera na kojeg se fokusira, i to se proslijedi u Blackboard, postavi se da neprijatelj pronašao igrača, postavi se mogućnost izvlačenja oružja te se postavlja da je neprijatelj fokusiran i na koga.

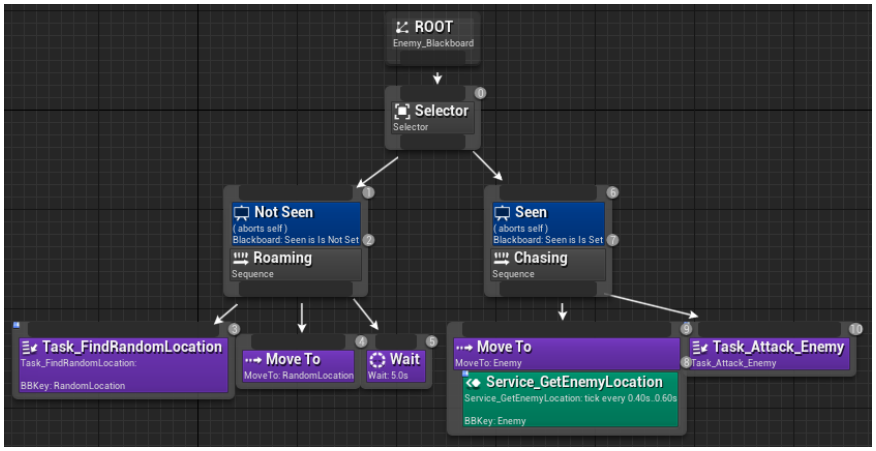


Slika 69: Neprijateljski AI blueprint (Izvor: vlastita slika iz alata)

U behavior tree-u iz roota uzmemo selector, ako je igrač nije viđen izvršit će se „Roaming“ sekvenca, ako je viđen izvršit će se „Chasing“ sekvenca.

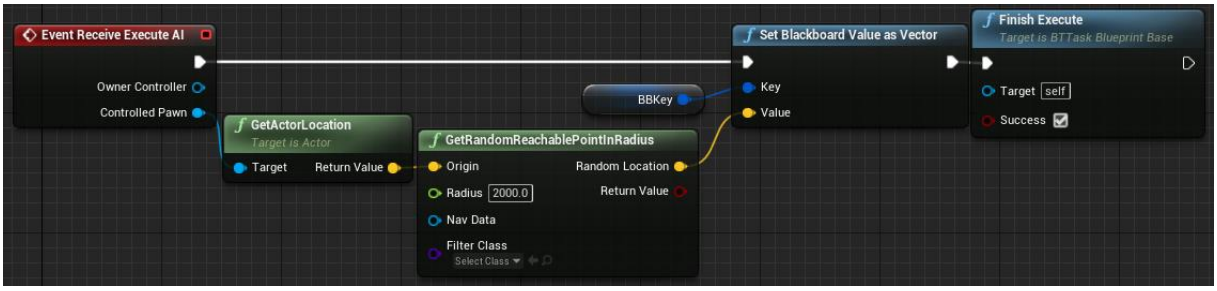
„Roaming“ sekvenca prvo izvrši task „FindRandomLocation“ zatim se pomakne na lokaciju koju je dobio iz taska, i nakon toga čeka 5 sekundi.

„Chasing“ sekvenca iz servisa „GetEnemyLocation“ dobije lokaciju neprijatelja te se pomakne prema njemu, i kada dođe do njega izvrši se task „Attack_Enemy“ preko kojeg napada neprijatelja.



Slika 70: Behavior tree neprijatelja (Izvor: vlastita slika iz alata)

Neprijatelj će na početku taska „GetEnemyLocation“ uzeti svoju lokaciju i iz nje generirati nasumičnu lokaciju u radiusu 2000 te će na Blackboard postaviti tu lokaciju i završiti task.

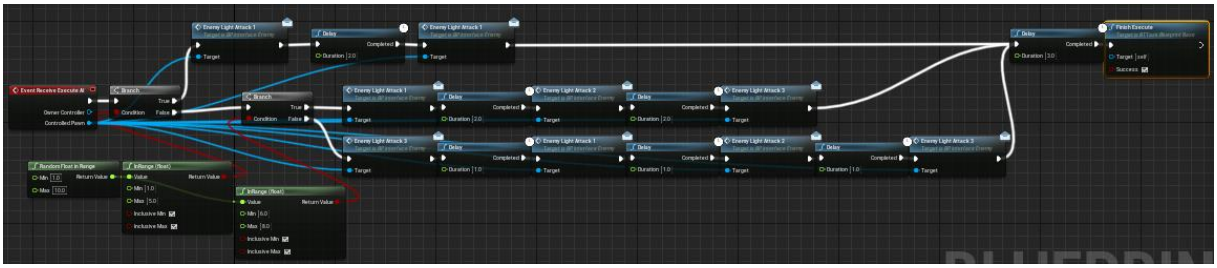


Slika 71: Task GetRandomLocation (Izvor: vlastita slika iz alata)

Servis „GetEnemyLocation“ će uzeti karaktera na kojeg je neprijatelj fokusiran te ga postaviti kao vrijednost u Blackboard.

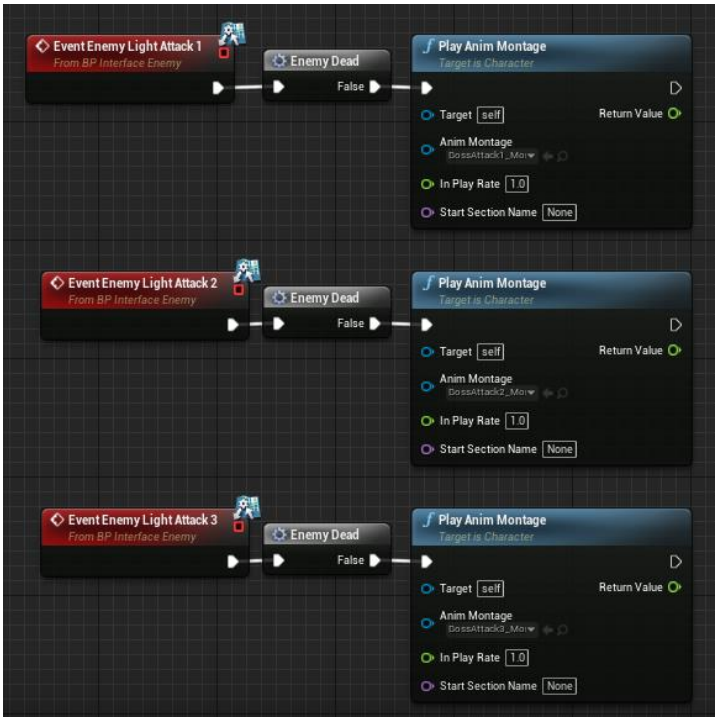
Slika 72: Servis GetEnemyLocation (Izvor: vlastita slika iz alata)

Task „Attack_Enemy“ će generirati random broj od 1 do 10, ako je broj između 1 i 5 pokreće se sekvenca od 2 eventa koji simboliziraju različite napade, ako je broj između 6 i 8 izvršava se sekvenca od 3 različita eventa napada, te zadnji slučaj je sekvenca od 4 eventa napada, te kada se to izvrši završava task.



Slika 73: Task Attack_Enemy (Izvor: vlastita slika iz alata)

U blueprint klasi neprijatelja koja je dijete bazne klase zovemo evente napada, te ako neprijatelj nije mrtav izvršavamo animacije napada.



Slika 74: Pozivanja animacija napada (Izvor: vlastita slika iz alata)

4.6.2.Život neprijatelja

Prvo se kreira običan widget kao i za prikazivanje statistike igrača no sada widget ima samo traku za život. U baznoj klasi neprijatelja kao i u baznoj klasi igrača svaki frame neprijatelj osvježava statistiku svog života.

Ne žele se vidjeti trake neprijatelja ako se igrač ne približi dovoljno blizu neprijatelja te će se dodati na neprijatelja mesh neprijatelja direktno taj widget da mu bude prikazan iznad glave i „Sphere Collision“ u koji ako se uđe će postaviti vidljivost widgeta na istinu a ako se izađe postavit će na laž.

Slika 75: Logika prikazivanja widgeta (Izvor: vlastita slika iz alata)

5. Pojmovnik

5.1. Actor

Actor je bilo koji objekt koji se može smjestiti u nivo, kao što je kamera, statična mreža ili mjesto početka igrača. Actori podržavaju 3D transformacije poput translacije, rotacije i skaliranja. Mogu se stvoriti uništiti pomoću koda (C ++ ili Blueprinta). [17]

5.2. Blueprint

Klasa Blueprint-a, često skraćena samo kao Blueprint, resurs je koji programerima igara omogućuje jednostavno dodavanje funkcionalnosti povrh postojećih klasa igre. Blueprinti se ne pišu kao kod, već se kreiraju vizualno u editoru, te se spremaju kao paket sadržaja. Oni u osnovi definiraju novu klasu ili tip actora koji se zatim mogu smjestiti na nivo kao instance koje se ponašaju kao bilo koja druga vrsta actora. [18]

5.3. Blend Space

Blend Space posebni su asseti koji se mogu uzorkovati u AnimGraphs koja omogućuju miješanje animacija na temelju vrijednosti dva ulaza. Jednostavno miješanje dvije animacije na temelju jednog ulaza može se izvesti pomoću jednog od standardnih Blend nodeova dostupnih u Animation Blueprints. Blend Space pruža način za složenije miješanje između više animacija na temelju više vrijednosti (ograničeno na dvije).

Cilj Blend Spaces je smanjiti potrebu za stvaranjem pojedinačnih, kodiranih čvorova za izvođenje miješanja na temelju određenih svojstava ili uvjeta. Omogućavajući animatoru ili programeru da navede ulaze, animacije i način na koji se ulazi kombiniraju između animacija, gotovo bilo koja vrsta miješanja može se izvesti pomoću generičkog Blend Space-a. [19]

5.4. Animation Blueprint

Animation Blueprint specijalizirani je nacrt koji kontrolira animaciju skeletnog mesha. Uređuju u programu Animation Blueprint Editor, gdje možete izvoditi miješanje animacije, izravno kontrolirati kosti kostura ili postavljati logiku koja će u konačnici definirati konačnu pozu animacije za Skeletal Mesh koja se koristi po kadru. [20]

5.5. Blueprint Interface

Blueprint Interface je zbirka jedne ili više funkcija - samo ime, bez implementacije - koja se može dodati ostalim Blueprintima. Bilo koji Blueprint kojem je dodan Blueprint Interface zajamčeno ima te funkcije. Funkcijama sučelja mogu se dati funkcionalnosti u svakom od nacrta koji su ga dodali. To je u osnovi poput koncepta sučelja u općenitom programiranju, koje omogućuje zajednički rad više različitih vrsta objekata i pristup zajedničkom sučelju. Jednostavno rečeno, Blueprint Interface omogućuju različitim Blueprintima međusobno dijeljenje i slanje podataka. [21]

Blueprint sučelja mogu stvoriti tvorci sadržaja putem uređivača na sličan način kao i drugi nacrti, ali oni imaju određena ograničenja jer ne mogu:

- Dodajte nove varijable
- Uredi grafikone
- Dodajte komponente

5.6. Node

Node su objekti poput događaja, poziva funkcije, operacija kontrole protoka, varijabli... koji se mogu koristiti u grafovima za definiranje funkcionalnosti određenog Blueprinta koji ga sadrži. [22]

5.7. Macro

Blueprint Macro je principu di koda koji je sažeti u neki node. Imaju ulaznu i izlaznu točku u node tunelima. Svaki tunel može imati bilo koji broj izvršavanja ili pinova podataka koji su vidljivi na čvoru makronaredbi kada se koriste u drugim nacrtima i grafikonima. [23]

5.8. Event

Eventi su nodeovi koji se pozivaju iz koda za igre kako bi se započelo izvršavanje pojedinačne mreže unutar EventGraph-a. Omogućuju blueprinta da izvrše niz radnji kao odgovor na određene događaje koji se događaju u igri (kada igra započne, kada se razina resetira ili kada igrač napravi štetu...). Događajima se može pristupiti u Blueprintima radi primjene nove funkcionalnosti ili nadjačavanja ili proširivanja zadane funkcionalnosti. Bilo koji broj događaja može se koristiti unutar jednog EventGraph-a. [24]

5.9. Animation Montage

Animation Montage (AnimMontage) omogućuju širok spektar animacijskih efekata, prvenstveno povezanih s izlaganjem kontrola animacije unutar skriptiranja Blueprint-a ili putem koda. Također se mogu koristiti za stvaranje širokog spektra animacijskih efekata, uključujući inteligentne petlje animacije, prebacivanje animacije temeljeno na logici... [25]

5.10. Widget

Sama komponenta widgeta 3D je primjerak nacrta widgeta s kojim se može komunicirati u svijetu igre. Widget komponente omogućuju manifestiranje 3D elemenata korisničkog sučelja stvorenih kroz Unreal Motion Graphics u svijetu igre. [26]

5.11. Sound Cue

Sound Cue je audio objekt koji obuhvaća složene zadatke dizajniranja zvuka u grafu nodea. Zvučni signali pružaju slobodu da se dinamički mijenjaju dijelove dizajna zvučnog efekta uređivanjem i modificiranjem zvučnih nodeova, stvarajući složene i zanimljive audio izlaze. [27]

5.12. Behavior Tree

Bahavior Tree može se koristiti za stvaranje umjetne inteligencije (AI) za likove koji nisu igrači. Behavior tree koristi za izvršavanje grana koje sadrže logiku, da bi se utvrdilo koje grane treba izvršiti, behaviour tree oslanja se na Blackboard koji služi kao "mozak". [28]

6. Zaključak

Razvoj video igara vjerojatno je najzahtjevniji način razvijanja informatičkih proizvoda. To je zato što spaja mnogo različitih područja informatike u jedno. Postoji dio koji je povezan s programiranjem, dio koji je povezan dizajnom statika, dio povezan dizajnom animacija, dio povezan dizajnom zvuka i mnoštvo drugih. Svaki od tih dijelova je zahtjevan sam po sebi, no izazov razvoja igara je povezati se te dijelove u jedan tako da su svi međusobno komplementarni te se osnažuju.

Drago mi je što sam odabrao Unreal Engine 4, jer je to razvojni alat koji u sebi ima sve potrebne alate za rad u svim disciplinama koji su potrebni da se razvije funkcionalna igra. Štoviše, najavom Unreal Engine 5, Epic games uvodi poboljšane alate kako naprimjer za animiranje koji su toliko razvijeni da su bolji i jednostavniji od drugih alata koji su specijalizirani samo za to područje. Time će se postići da se Unreal Engine postati sve više zastupljen u razvoju igara, i znanje rada u tom razvojnom alatu postat će vrlo vrijedno na tržištu rada.

Samo iskustvo razvoja igre u Unreal Engine 4 za mene je bilo jako interesantno. Blueprint način skriptiranja koda je u isto vrijeme jednostavan za pohvatati u smislu novog jezika i načina programiranja, a u isto vrijeme i malo zbunjujuć jer način rada na koji sam da sad programirao nije primjenjiv na blueprint skriptiranje. No kao i bilo koji programerski jezik, kroz korištenje je postao sve lakši i intuitivniji, također zbog dobre dokumentacije koja se nalazi na internetu.

Kao zaključne misli rekao bih da je razvoj igara vrlo zanimljiv proces, koji zahtjeva različita znanja i vještine, od kojih su mi neke zanimljivije od drugih. Iz tog razloga ako bih se nastavio baviti razvojem igara ne bi htio razvijati igre sam, već bi htio raditi u timu u kojem svaka osoba ima svoje zadatke koji se ponekada isprepliću.

Popis literature

[1] »What is Game Engine?,« Full Scale, 11 Siječanj 2021. [Mrežno]. Available: https://fullscale.io/blog/what-is-game-engine/. [Pokušaj pristupa 5 Srpanj 2021].

[2] Wikipedia, »Unreal Engine,« 6 Srpanj 2021. [Mrežno]. Available: https://en.wikipedia.org/wiki/Unreal_Engine. [Pokušaj pristupa 6 Srpanj 2021].

[3] »Unreal Engine 4.26 Documentation - Blueprint Visual Scripting,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/. [Pokušaj pristupa 6 Srpanj 2021].

[4] »Marketplace,« Epic Games, [Mrežno]. Available: https://www.unrealengine.com/marketplace. [Pokušaj pristupa 6 Srpanj 2021].

[5] »Unreal Engine 5 is now available in Early Access!,« Epic Games, 26 Svibanj 2021. [Mrežno]. Available: https://www.unrealengine.com/en-US/blog/unreal-engine-5-is-now-available-in-early-access. [Pokušaj pristupa 6 Srpanj 2021].

[6] »Mixamo,« Adobe, [Mrežno]. Available: https://www.mixamo.com/#/. [Pokušaj pristupa 5 Srpanj 2021].

[7] »Infinity Blade: Weapons,« Epic Games, 9 Rujan 2015. [Mrežno]. Available: https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-weapons. [Pokušaj pristupa 6 Srpanj 2021].

[8] »Infinity Blade: Effects,« Epic Games, 9 Rujan 2015. [Mrežno]. Available: https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-effects. [Pokušaj pristupa 6 Srpanj 2021].

[9] »Dark Souls - "You Died" Sound Effect,« 13 Travanj 2013. [Mrežno]. Available: https://www.youtube.com/watch?v=CpaT28qMfis. [Pokušaj pristupa 6 Srpanj 2021].

[10] »Dark Souls "You Defeated/Victory Achieved" Sound Effect,« 16 Lipanj 2016. [Mrežno]. Available: https://www.youtube.com/watch?v=bny63lsskhY. [Pokušaj pristupa 6 Srpanj 2021].

[11] M. Station, »Multistory Dungeons,« Epic Games, 18 Studeni 2015. [Mrežno]. Available: https://www.unrealengine.com/marketplace/en-US/product/top-down-multistory-dungeons. [Pokušaj pristupa 6 Srpanj 2021].

[12] »Hemi Head,« Tyoidermic Fonts, 23 Lipanj 2014. [Mrežno]. Available: https://www.dafont.com/hemi-head.font. [Pokušaj pristupa 6 Srpanj 2021].

[13] M. Cupelli, »DARK FANTASY MUSIC | Royalty Free,« 30 Lipanj 2020. [Mrežno]. Available: https://www.youtube.com/watch?v=djQrAY6YDJA. [Pokušaj pristupa 6 Srpanj 2021].

[14] I. Studio, »Medieval Dungeon,« 25 Siječanj 2019. [Mrežno]. Available: https://www.unrealengine.com/marketplace/en-US/product/a5b6a73fea5340bda9b8ac33d877c9e2. [Pokušaj pristupa 6 Srpanj 2021].

[15] »Taurus Demon - Dark Souls Soundtrack,« 29 Rujan 2011. [Mrežno]. Available: https://www.youtube.com/watch?v=ovjd22Rkhlk. [Pokušaj pristupa 6 Srpanj 2021].

[16] L. R. S. Factory, »Fantasy Sound Effects Library,« 13 Travanj 2015. [Mrežno]. Available: https://opengameart.org/content/fantasy-sound-effects-library. [Pokušaj pristupa 6 Srpanj 2021].

[17] »Unreal Engine 4.26 Documentation - Actors,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Actors/. [Pokušaj pristupa 5 Srpanj 2021].

[18] »Unreal Engine 4.26 Documentation - Blueprint Overview,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/Overview/. [Pokušaj pristupa 5 Srpanj 2021].

[19] »Unreal Engine 4.26 Documentation - Blend Spaces,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/SkeletalMeshAnimation/Blendspaces/. [Pokušaj pristupa 5 Srpanj 2021].

[20] »Unreal Engine 4.26 Documentation - Animation Blueprints,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/. [Pokušaj pristupa 5 Srpanj 2021].

[21] »Unreal Engine 4.26 Documentation - Blueprint Intefrace,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/Interface/. [Pokušaj pristupa 5 Srpanj 2021].

[22] »Unreal Engine 4.26 Documentation - Nodes,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Nodes/. [Pokušaj pristupa 5 Srpanj 2021].

[23] »Unreal Engine 4.26 Documentation - Macros,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Macros/. [Pokušaj pristupa 5 Srpanj 2021].

[24] »Unreal Engine 4.26 Documentation - Events,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Events/. [Pokušaj pristupa 5 Srpanj 2021].

[25] »Unreal Engine 4.26 Documentation - Animation Montage,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimMontage/. [Pokušaj pristupa 5 Srpanj 2021].

[26] »Unreal Engine 4.26 Documentation - Widget Components,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/Basics/Components/Widget/. [Pokušaj pristupa 6 Srpanj 2021].

[27] »Unreal Engine 4.26 Documentation - Sound Cue Reference,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/WorkingWithMedia/Audio/SoundCues/NodeReference/. [Pokušaj pristupa 6 Srpanj 2021].

[28] »Unreal Engine 4.26 Documentation - Behavior Trees,« Epic Games, [Mrežno]. Available: https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/ArtificialIntelligence/BehaviorTrees/. [Pokušaj pristupa 7 Srpanj 2021].

Popis slika