Treniranje funkcionalne umjetne inteligencije i razrada NPC sustava za potrebe razvoja FPS računalne igre

Martinović, Marijan

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

MARIJAN MARTINOVIĆ

TRENIRANJE FUNKCIONALNE UMJETNE INTELIGENCIJE I RAZRADA NPC SUSTAVA ZA POTREBE RAZVOJA FPS RAČUNALNE IGRE

DIPLOMSKI RAD

Zagreb, 2023.

Sveučilište u Zagrebu
Grafički fakultet

MARIJAN MARTINOVIĆ

TRENIRANJE FUNKCIONALNE UMJETNE INTELIGENCIJE I RAZRADA NPC SUSTAVA ZA POTREBE RAZVOJA FPS RAČUNALNE IGRE

DIPLOMSKI RAD

Mentor:
izv.prof.dr.sc. Tibor Skala

Student:
Marijan Martinović

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET
Getaldićeva 2
Zagreb, 13. 9. 2023.

Temeljem podnijetog zahtjeva za prijavu teme diplomskog rada izdaje se

R J E Š E N J E

kojim se studentu/ici Marijanu Martinoviću, JMBAG 0269103024, sukladno čl. 5. st. 5. Pravilnika o izradi i obrani diplomskog rada od 13.02.2012. godine, odobrava izrada diplomskog rada, pod naslovom: Treniranje funkcionalne umjetne inteligencije i razrada NPC sustava za potrebe razvoja FPS računalne igre, pod mentorstvom izv. prof. dr. sc. Tibora Skale.

Sukladno čl. 9. st. 1. Pravilnika o izradi i obrani diplomskog rada od 13.02.2012. godine, Povjerenstvo za nastavu, završne i diplomske ispite predložilo je ispitno Povjerenstvo kako slijedi:

1. doc. dr. sc. Rudolf Maja, predsjednik/ica
2. izv. prof. dr. sc. Skala Tibor, mentor/ica
3. doc. dr. sc. Stanić Loknar Nikolina, član/ica

REPUBLIKA HRVATSKA
GRAFIČKI FAKULTET ZAGREB
SVEUČILIŠTE U ZAGREBU

Dekan:
Prof. dr. sc. Klaudio Pap

SAŽETAK

Od kraja 90-ih godina prošlog stoljeća, industrija video igara raste sa svakom godinom. Video igra kao produkt može biti rezultat individue, malog tima do zajedničkog rada desetke možda i stotine različitih stručnjaka. Uz današnju razinu povezanosti preko interneta i nove poslovne paradigme od strane velikih kompanija koje proizvode programe za razvoj video igara, znanje i resursi su veoma dostupni za sve koji su voljni okušati se u jednom ili više aspekata razvoja videoigara.
Umjetna inteligencija je ključna komponenta u velikom broju modernih videoigra. Uz novi besplatni Unreal Engine 5 od strane Epic Gamesa, ovaj rad nadograđuje osnovni paket za pucačinu iz prvog lica koja dolazi s programom. Dodavanjem brojnih komponenata osnovni paket proširuje s karakterima koji mogu samostalno surađivati međusobno i s igračem. Prvi dio rada prolazi kroz skromne početke žanra pucačine iz prvog lica, te prati napredak sustava za umjetnu inteligenciju kroz nadolazeće naslove. Kroz važnije naslove prolazi kroz nove sustave, njihovu logiku i implementaciju unutar video igrice te razmatramo napredak tih sustava.

Drugi dio rada, projekt, razgrađuje nove komponente dodane u paket i objašnjava njihovu logiku i ulogu unutar cijelog sustava video igrice. Od važnijih komponenata u projekt dodaje novi prošireni okoliš s navigacijskom mrežom, četiri suigrača s umjetnom inteligencijom koja reagira i sudjeluje u borbi te sustav zdravlja i ponovnog stvaranja za igrača i suigrače. Svaki od novih komponenata ima svoju ulogu, svoj kod i logiku izvedbe te rad prolazi kroz njih.

Ključne riječi: Videoigre, Razvoj, Umjetna inteligencija, Unreal Engine

SUMMARY

From the end of the 90s of the previous century the videogame industry has a steady annual growth. Video game as a product can be the result of an individual, a small team or joint effort from tenths if not hundreds of different experts. With todays level of connection through the Internet and new buisness paradigms from big companies which produce programs for development of video games, knowledge and resources are easily available for anyone willing to try their hand at one or more aspects of video game development.

Artificial intelligence is a key component in a large number of modern video games. With new free Unreal Engine 5 from Epic Games, this thesis upgrades the first person template project which arrives standard with the program. By adding multiple components the standard package is expanded with characters able to interact with themselves and with the player without additional input. First part of the thesis goes over humble beginnings of the first person shooter genre, and follows the advancements of the systems for artificial intelligence through future titles. Through the important titles it goes through new systems, their logic and their implementation within the video game and observe the improvements of those systems.

Second part of the thesis, the project, breaks down the new components added into the package and explains their logic and role within the whole system of a video game. Of the more important components it adds new expanded playing enviroment with the matching navigation mesh, four characters with artificial intelligence which reacts and takes part in combat and a health and respawn system for the player and the characters. Each of the new components has his own role, code and logic of execution, and the thesis goes through them.

Keywords: Videogames, Development, Artificial Intelligence, Unreal Engine

1. UVOD

Pojam umjetna inteligencija je u današnjem društvu veoma širok i opširan pojam koji se pojavljuje u mnogo različitih tehnologija i usluga. Često je krivo shvaćen kao prijetnja ljudskom društvu i postojanju te je zamišljena kao mračna budućnost nekog distopijskog društva. Dok je takva okolina veoma privlačna za zbivanje nekog znanstveno fantastičnog filma, sama logika i principi na kojima se umjetna inteligencija zasniva čvrsto prati svoja ograničenja.

Inteligencija koja je nama oslikana kao veliki zlikovac budućnosti se više zasniva na samostalnoj inteligenciji digitalnog porijekla , a takvi pothvati su još daleko van ljudske sposobnosti. Dapače sama umjetna inteligencija može doseći toliko visoku razinu iluzije samostalnosti i inteligencije da nije iznenađujuće zašto bi ljudi pomislili da su digitalna bića sa svojom slobodnom voljom sljedeći korak u digitalnoj revoluciji 21. stoljeća.

Usprkos svemu tome, današnja umjetna inteligencija je još uvijek samo to, umjetna, stvara nam privid inteligencije, dok se zapravo se zasniva na kompleksnoj logici i visokoj matematici. Dok je većini ljudi ta razina kalkulacija van našeg laičkog poimanja, još uvijek je zasnovana na granama znanosti koji su ljudi otkrili i savladali.

Postoji mnogo vrsta umjetne inteligencije, od ciljanih reklama za proizvode koje algoritam vjeruje da kupac želi kupiti, do izbacivanja aktualnih vijesti baziranih na broju pregleda, u ovom radu dat je prikaz umjetne inteligencije u video igricama, kako stvaraju iluziju ljudskih protagonista, neprijatelja, prepreka i realnog okoliša.

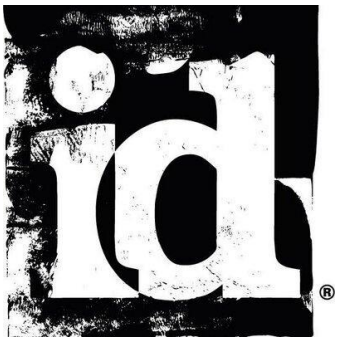
2. TEORIJSKI DIO

2.1. UMJETNA INTELIGENCIJA U STARIJIM IGRICAMA

Umjetna inteligencija je često asocirana s video igrama, i ključni je dio u stvaranju dinamične razine kompleksnosti proizvoda tako da zadovolji široki spektar korisnika. Veliki i osnovni je dio mnogih video igrica, ali za ovaj rad ne zalazi u stare arkadne igre, već se fokusira na popularne igrice kroz nedavnu povijest i rast popularnosti video igara kroz 1990-e do današnjeg doba. Doba gdje su osobna računala postala dostupna široj populaciji te razvoj industrije video igara počinje svoj dan danas neprestani rast.

2.1.1.DOOM, FSM I POČETAK FPS ŽANRA

2.1.1.1 ID SOFTWARE



Slika 1. id Software logo.

Izvor: https://en.wikipedia.org/wiki/Id_Software

Id Software [1] je američki studio za video igrice iz Richardsona, Texas. Osnovan je 1991. godine s četiri člana: programeri John Carmack i John Romero, dizajner igrica Tom Hall i umjetnik Adrian Carmack. Id Software je razvio svoj vlastiti 3d engine (hrv. Jezgra igre s tri dimenzije) koji dan danas ima razne inačice i uporabe u industriji video igara. Jezgra igre je glavni pokretač igrice, te je zaslužna za izvedbu svih njezinih dijelova. Animacije, teksture, zvukovi i ostali komponenti igre se prikazuju kroz jezgru. Id engine je razvio John Carmack uz pomoć John Romera te je uz igrice poput Wolfenstein 3D iz 1992., Doom 1993. i Quake iz 1996. doveo žanr FPS(First Person Shooter)(hrv. Pucačina iz prvog lica) igrica u popularnost.

Slika 2. id Software originalni tim krajem 1992.

Izvor: https://gamerant.com/id-software-developer-history-doom-wolfenstein-quake-microsoft/

Gledajući kako se rad bazira na umjetnoj inteligenciji u 3D okružju, rane igrice id Softwarea su idealna početka točka.

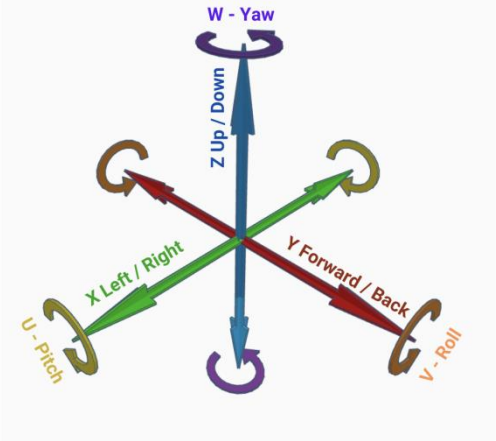
Id engine je 1992. napravio revoluciju u industriji video igara s tehnologijama koji se nisu činile moguće. Kompleksne, granajuće razine u 3D okružju koji su uz sadrže veliki broj neprijatelja koji svi imaju svoje osobne statistike i načine ponašanja oviseći o okolini ili igraču, tj posjeduju umjetnu inteligenciju.

Slika 3. Wolfenstein 3d iz 1992. i Doom iz 1993.

Izvori: https://store.steampowered.com/app/2270/Wolfenstein_3D/

https://store.steampowered.com/app/2280/DOOM_1993/

Dok su Wolfenstein 3D i rana Doom franšiza sebe reklamirali kao 3D video igrice, istina je da id engine koji je korišten u tim igricama, nije koristio „pravi“ 3d. Naime prava 3D tehnologija još nije bila spremna za svakodnevnu uporabu na osobnim računalima. Id engine je koristio puno trikova da simulira 3d okoliš, ali još uvijek nije koristio visinu kao dimenziju u igrici. Moguće je definirati razinu slobode koju okoliš podržava [13]. Najveći stupanj slobode je šest stupnjeva, tri osi i tri rotacije oko te tri osi.



Slika 4. Šest stupnjeva slobode.

Izvor: https://industrial-ia.com/what-are-the-6-degrees-of-freedom-6dof-explained/

Oboje Wolfenstein i Doom ne dopuštaju igraču da promjeni kut gledanja gore-donje, samo u lijevo-desno, dakle dopušta kretnju po naprijed- nazad osi, te po lijevo-desno osi, uz to dopuštena je rotacije po osi visine, što nam dopušta gledanje lijevo-desno. Dok visina nije uopće prisutna u Wolfenstein 3D, visina u Doom-u postoji, ona nije os nad kojom igrač ima kontrolu, te obje igre imaju isti stupanj slobode od tri stupnja.

- Pomicanje naprijed/nazad (Y os)
- Pomicanje desno/lijevo (X os)
- Rotacija oko visine (Rotacija oko W osi, Yaw)

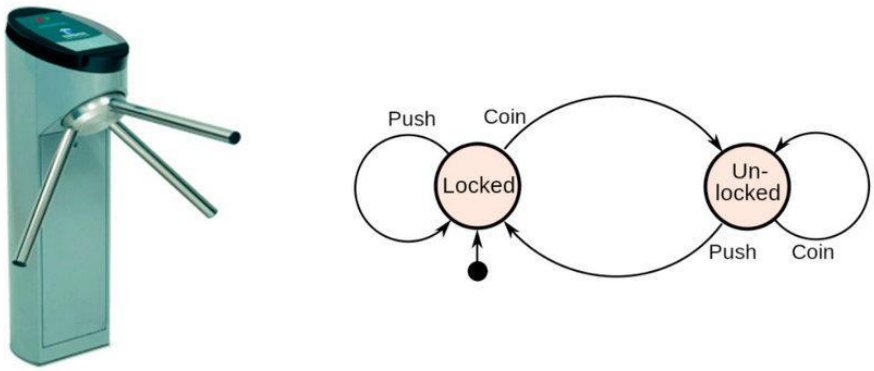
S tim limitacijama pravi 3D okoliš nije simuliran. Quake s novim id engineom 1996. dopušta pomicanje po sve tri osi i rotaciju po sve tri osi.

Prva igrica u 3d okolišu od Id-a, Wolfenstein 3D, sadrži granajuće labirinte kao svoje razine, neprijatelji patroliraju okolišem i reagiraju na igrača jednom kad ga uoče, razlika između Wolfenstein-a i Doom-a je da u Wolfenstein-u cijela igrica stoji na jednoj ravnini, nema promjena u vertikalnoj visini. Kroz cijelu video igricu nema stuba ili uzvišenog ili nižeg terena. Uz to svi neprijatelji funkcioniraju kao jedan tim, praktični je protagonist sam protiv svih. Doom i Wolfenstein oboje koriste istu bazu umjetne inteligencije neprijatelja, te Doom donosi broj sitnih detalja koji produbljuju iluziju stvarnog i slojevitog okoliša. Kroz Doom možemo pronaći razine s različitim visinama terena i bolju interakciju između neprijateljima. Može se reći da Doom duguje svoju popularnost Wolfenstein-u, koji je zapalio iskru zanimanja za njihove igrice, tako da Doom može uživati u punom intenzitetu FPS revolucije.

Umjetna inteligencija u tim igricama koristi Finite State Machine(Automat s konačnim brojem stanja) te se ista logika često koristi i danas u mnogim video igricama.

2.1.1.2 AUTOMAT S KONAČNIM BROJEM STANJA(FSM)

Automat s konačnim brojem stanja [2] (eng. Finite State Machine), skraćeno na FSM, se sastoji od više konačnih stanja koji se mijenjaju ovisno o zadovoljenim kriterijima. Dobar primjer je automat za okretište.

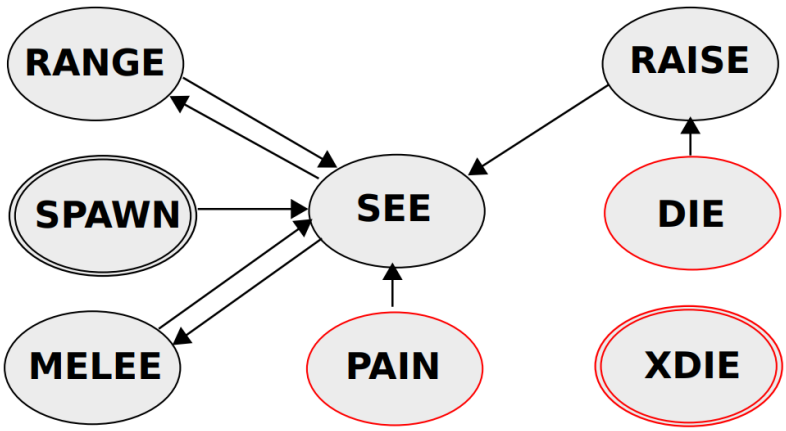


Slika 5. Okretište.

Izvor: https://www.researchgate.net/figure/Left-A-turnstile-machine-by-Wikimedia-Commons-user-Sebasgui-GFDL-Right-State_fig1_321449423

Ovaj automat ima dva stanja, otključano ili zaključano. Automat možemo okrenuti, te ako nema kovanice, ostaje u zaključanom stanju, ali ako je novčić prisutan, automat prijelazi u otključano stanje, te nakon toga ponovo ulazi u zaključano. Ako po istoj ovoj logici funkcionira umjetna inteligencija svih neprijatelja u igri, u toj logici igrač ima ulogu novčića te pogoni mijenjanje između stanja.

Ovako izgleda dijagram središnje logike neprijatelja iz Doom video igrice.



Slika 6. FSM dijagram umjetne inteligencije unutar Doom-a.

Izvor: https://www.gamedeveloper.com/blogs/the-ai-of-doom-1993

Svaki od stanja ima svoju logiku uz pomoć koje ulazi ili izlazi iz tih stanja. [3]

Prvo svi neprijatelji se pojavljuju u „Spawn” (hrv. Stvaranje) stanju, što znači da su učitani u mapu i stoje u pasivnom stanju, te čekaju poticaj od igrača da se to stanje promjeni, dakle da vide igrača ili ga čuju.

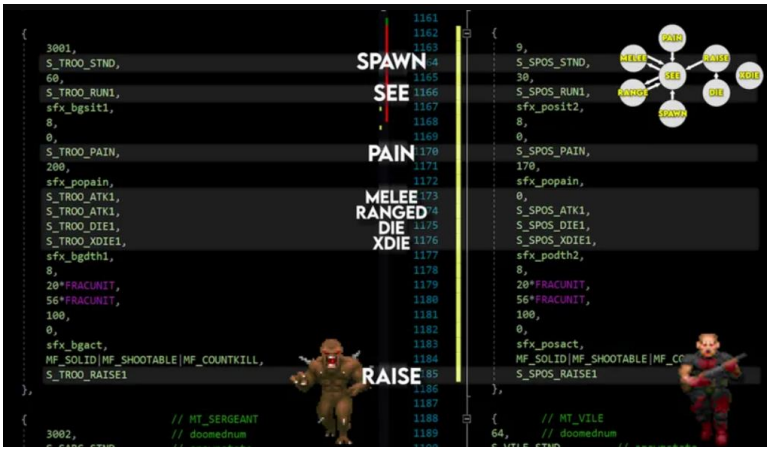
Za promjenu stanja u „See“(hrv. Vidi) stanje, neprijatelj mora imati direktni pogled na igrača, što znači da igrač treba biti u prednjem radijusu od 180 stupnjeva čudovišta. Drugi način je da čudovišta „čuju“ igrača. Dok zvuči čudno, neprijatelji zapravo vide zvuk u Doom-u. Svaki put kada igrač napravi zvuk uz pomoć oružja, igrač je označen kao izvor zvuka, koji neprijatelji mogu napasti ako su u mogućnosti da čuju taj zvuk. Mapa je podijeljena na sektore, ako je neprijatelj u sektoru u kojem je izvor zvuka, ili u susjednom sektoru koji nije ograđen protiv zvuka, igrač postaje meta. Jednom kada je kriterij zadovoljen, ulaze u „See“ stanje, iz kojeg sami mogu promijeniti u stanje napada, dakle „Range“(hrv. Domet) ili „Melee“(hrv. Borba prsa o prsa) zaviseći o vrsti neprijatelja i njegovom arsenalu napada.

Promjena iz „See“ stanja u napad, također povlači informacije za određene grafike koje pripadaju tom napadu, odigra grafike za napad određeni broj sličica, pušta zvuk napada uz to, te kalkulira je li napad pogodio ili promašio.

Dok u dva stanja napada mogu sami ulaziti i izlaziti, tj vidjeti igrača u „See“ stanju, napasti ga u „Range“ ili „Melee“ stanju, te se opet sami vratiti u „See“ stanje. Za stanje „Pain“(hrv. Bol) ulaze samo uz dodatni poticaj od strane igrača, kada ga igrač ozljedi. „Pain“ stanje je šansa za prekidanje napada ili pokreta u nakon kojeg se vraća nazad u „See“ stanje, svaki od neprijatelja ima različitu šansu da se to stanje uključi.

Šansa za „Pain“ stanje je definirana u odvojenoj datoteci u kojoj su definirane sve specifikacije neprijatelja, kao način napada, broj healtha(hrv. Zdravlja), veličina u okolišu etc. Šansa sa „Pain“ stanje je definirana s brojem od 0 do 256, koja nam daje postotak za uključivanje tog stanja jednom kada se neprijatelja ozljedi te se zove „Painchance“(hrv. Šansa za bol).

Na primjer „Imp“ ima 200/256 ili 79.3% šanse za „Pain“ stanje svaki put kada primi ozljedu od igrača, dok „Shotguy“ ima 170/256 ili 67.3% za to isto stanje. Dok se dodatak ovakvog stanja čini nepotrebnim na prvi površni pogled, dodatak ove funkcije dodaje pri načinima na koje igrač može pristupiti različitim situacijama i oružjima.

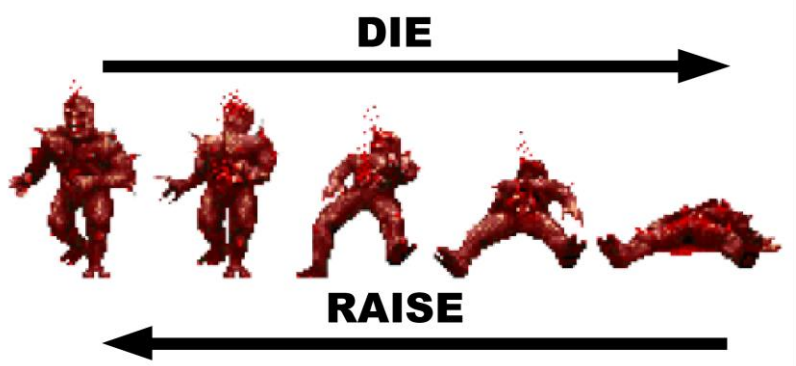


Slika 7. Usporedba „Imp“ i „Shotguy“ specifikacija.

Izvor: https://www.gamedeveloper.com/blogs/the-ai-of-doom-1993

Svaki neprijatelj također ima „Die“(hrv. Umri) i „Xdie“(hrv. Xumri) stanje, „Die“ je stanje obične smrti jednom kada mu broj zdravlja dođe do 0, neprijatelj ulazi u to stanje, igra animaciju smrti i pušta zvuk smrti. Dok „Xdie“ stanje je uključeno jedno ako neprijatelj primi ozljedu koja je dupla od vrijednosti njegovog maksimalnog zdravlja, što uzrokuje puštanjem animacije za eksploziju neprijatelja. Dok neki veći neprijatelji nemaju prilike da uključe to stanje, radi velikog broja zdravlja koji imaju, manji neprijatelji imaju veću šansu za to, koji imaju samo 20, 30 zdravlja.

Također može se vidjeti da oba stanja smrti prolaze kroz „Raise“(hrv. Ustani) stanje. Ta funkcija je uvedena radi neprijatelja koji ima mogućnost oživiti neprijatelja, a to učini tako da animaciju smrti iz „Die“ funkcije odigra unazad.



Slika 8. Primjer „Die“ i „Raise“ funkcije na „Imp“-u.

Dok svi neprijatelji imaju istu internu logiku, svaki neprijatelj ima drugačiji način razmišljanja na koji prilazi igraču i kako se ponaša, sve te informacije se drže u odvojenoj tekstualnoj datoteci koja drži sve specifikacije neprijatelja. Ovaj pristup pomaže igri da proda igraču ideju inteligencije među neprijateljima. Usprkos tome još jedan veliki igrač u tom procesu je sam okoliš i njegov dizajn.

Određeni neprijatelji također imaju opciju međusobne borbe, ako neprijatelja pogodi drugi neprijatelj prije nego što ga ozlijedi igrač, ti neprijatelji se počnu boriti međusobno, što je jedan od načina da se pokaže osobnost određenih vrsta neprijatelja. Iako na prvu ne služi prvotnoj svrsi neprijatelja da izazove igrača, sitnice ove prirode čine puno da prodaju iluziju živućeg svijeta unutar videoigre.

Uz to dizajneri koriste „Okidač“(eng. Trigger) entitete u dizajnu levela da naprave zasjede za igrača. Okidač dopušta dizajnerima da stvore jedinstvene scenarije za igrača, gdje puštaju neprijatelje na igrača u specifičnom trenutku.

Sličnu situaciju također stvaraju s neprijateljima koje označe kao „Gluhe“(eng. Deaf), uz pomoć ove funkcije neprijatelj ne napada radi pojave zvuka u okolišu, te ga napada jedino ako ga vidi uz pomoć „See“ funkcije tj. ako igrač prođe ispred njega. Dok se igrač često nađe u situaciji gdje je naviknut na napad od neprijatelja radi buke koje je uzročio, neprijatelj koji je označen kao gluh sada daje privid napada iz zasjede.

Umjetna inteligencija je oduvijek bila to, umjetna. Napravljena s iznimnom brigom za interakcije između igrača, okoliša i drugih neprijatelja, ona je spoj više sustava koji zajedno rade uz pomoć logike da uvjere igrača da svijet i okoliš u kojem se nalazi je popunjen s pametnim i svjesnim bićima. Gledajući ograničenu tehnologiju iz 1993. razina kompleksnosti

i optimizacije enginea koji je id Software tim uspio proizvesti je uistinu impresivna i stoji kao jedan od legendi i začetnika FPS žanra s opravdanim razlogom.

2.1.2. HALF LIFE I EVOLUCIJA FSM SUSTAVA

Dok je id Software začetnik kvalitetnih 3D okoliša u svijetu videoigara s Wolfenstein-om, Doom-om i Quake-om. S izlaskom Half Life igrice od novonastalog Valve studija Gabe Newella i Mike Harringtona, 3D okoliši dobivaju novu dubinu i razinu kompleksnosti što se tiče umjetne inteligencije, dizajna i imerzije u svijet video igrice. Half Life [7] koristi nadograđeni Quake id engine iz 1996. te radi visokih ambicija projekta oko 70% koda je bilo promijenjeno i tako je ova verzija Quake id enginea nazvana GoldSrc koji kasnije raste u Source i Source 2 engine na kojima se zasnivaju sve igrice Valve Softwarea. Dok je umjetna inteligencija u Quake-u iz 1996. impresivna s Half-Life-om vidimo novi pomak u unutrašnjoj logici koja se koristi za simulaciju umjetne inteligencije.

Dok prijašnje igrice koriste samo jednu grupu neprijatelja koji zajedno ide protiv igrača, u Half Life-u imamo više grupa koja svaka funkcionira po svojoj logici i drugačije reagira na igrača. Gledajući kako svaki od grupa je različita vrsta stvorenja, tako i načini na koji se ponašaju trebaju biti odvojeni i specifični za njihovu grupu, Valve je uveo sustav „rasporeda akcija“ za svakog pojedinačnog entiteta u svijetu.

Svaki pojedinačni entitet koji sadrži umjetnu inteligenciju i s kojim igrač ima interakciju nazivamo NPC što je skraćeno od Non Player Character(hrv. Karakter koji nije igrač). Način na koji se sve verzije NPC-a razlikuju između sebe je uz pomoć posebnih bit-ova koji drže informacije o svakom pojedinom NPC-u. Bit je 32-člani integer koji se koristi u više navrata, svaki NPC ima više bitova koji zapisuju informacije uz pomoć kojih izvodi akcije u svijetu. Bit može biti „sposobni“, „zvučni“, „memorijski“ i „kondicijski“, dakle jedan bit se koristi za kondicije koje ovise o trenutnoj situaciji, jedan se bavi zvukovima koji treba pustiti, jedan pamti stvari koje su se dogodile i jedan se bavi sposobnostima NPC-a trenutno u svijetu.

Uz pomoć ovog sustava, svaki NPC ima veliki broj izbora akcija koji je jedinstven za njih specifično, također je lagan za memoriju računala. Na primjeru bita za „kondicije“ možemo vidjeti koliko jedinstvenih akcija jedan bit može pozvati.

Constant	Value	What it means to the monster
bits_COND_NO_AMMO_LOADED	(1 << 0)	My firearm is empty.
bits_COND_SEE_HATE	(1 << 1)	I see an enemy that I hate (more on this later).
bits_COND_SEE_FEAR	(1 << 2)	I see an enemy that I am afraid of (more on this later).
bits_COND_SEE_DISLIKE	(1 << 3)	I see an enemy that I dislike (more on this later).
bits_COND_SEE_ENEMY	(1 << 4)	I see an enemy.
bits_COND_ENEMY_OCCLUDED	(1 << 5)	Enemy is occluded by the world.

bits_COND_SMELL_FOOD	(1 << 6)	I smell food.
bits_COND_ENEMY_TOOFAR	(1 << 7)	My enemy is too far.

Primjer: „bits_COND“ podražaja.

Izvor: https://twhl.info/wiki/page/VERC%3A_Half-Life_AI%2C_Schedules_and_Tasks

Ovo je samo dio „bits_COND“ vrijednosti, ukupno ima 32 broja koji variraju između 0 i 1, dok 0 stoji za „Krivo“(eng. False) , 1 stoji za „Istina“(eng. True). Te dopušta jednom integeru da komunicira 32 različita stanja.

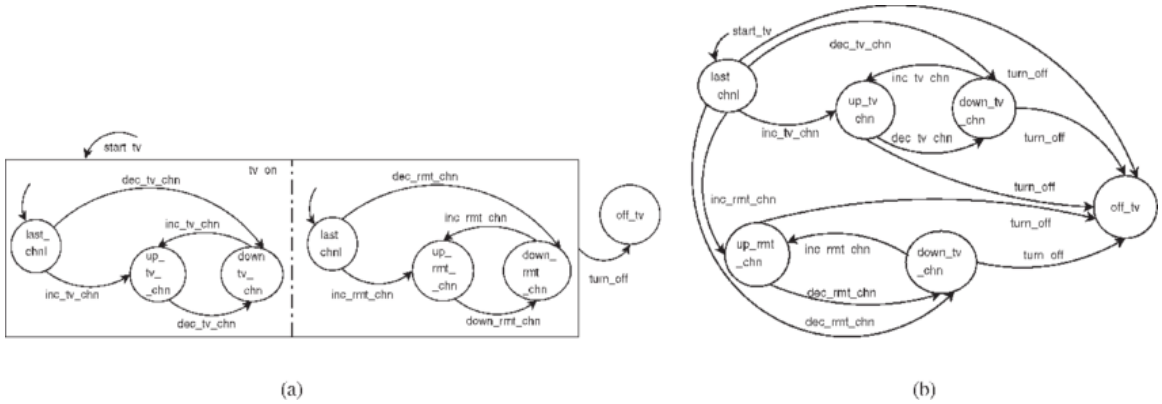
Uz pomoć ovih stanja, svaki od neprijatelja može odlučiti svoj sljedeći raspored puno specifičnije, ako je kondicija za „bits_COND_NO_AMMO_LOADED“(hrv. Bits_kond_nema_municije), neprijatelj si slaže raspored za obnovu municije, ako je uz to također primio ozljedu „bits_COND_HEAVY_DAMAGE“(eng. Bits_kond_teška_šteta), prvo si stvara raspored za bijeg u zaklon, te nakon toga, raspored za obnovu municije. S implementacijom ovih sustava sada imamo tri sustava koja rade zajedno, jedan koji označava grupu kojem neprijatelj pripada i koja je njegova uloga u toj grupi, 32-bitni integeri koji označavaju status svakog neprijatelja, te FSM sustav koji slaže osnovnu logiku i raspored akcija za svaki trenutak svakog neprijatelja. Samih konačnih zadataka koje raspored ispunjava ima preko 80, a specifičnih rasporeda preko 40, raspoređeni između vrsta i grupa neprijatelja.

Dok svaki od rasporeda također ima i kondicije u kojima se trenutni raspored zapiše kao propali, te entitet piše novi raspored uzimajući u obzir najnovije kondicije, sami zadaci se ne mogu spajati, npr. neprijatelj ne može pucati i kretati se u isto vrijeme. Svaki od zadataka je pojedinačna funkcija, koja se izvršava jedna po jedna, ako se raspored izvrši u potpunosti, stvara se novi raspored opet s novim kondicijama.

Naravno kao i prije veliki dio igra i sam okoliš unutar igrice, neprijatelji su naučeni da pronalaze zaklon iza zidova, kutija, prepreka. Dok su neprijatelji u Doom-u koristili okoliš bez puno interakcije, sada neprijatelji koriste okoliš kao alat. Bježe od igrača iza prepreka, bacaju granate preko prepreke gdje se nalazi igrač, mijenjaju položaj zaviseći o trenutnom položaju, i još puno toga. Sve interakcije koje dolaze preko podražaja u 32 bitnom integeru.

Uzimajući korak natrag i gledajući interakciju između svih ovih različitih sustava, možemo vidjeti evoluciju iz Doom-ove jednostavne logike. Dok logika ostaje ista, sustav je produbljen s mnogo različitih kondicija i grupa što omogućava veći broj ishoda u svakom okršaju s umjetnom inteligencijom.

Jedan od problema s FSM sustavom je njegova čitljivost, ako se broj ponašanja počne povećavati, logika sustava lagano postane komplicirana za pratiti, veliki broj ulaza i izlaza u stanja brzo postaje problem. Djelomično rješenje ovog problema možemo vidjeti u Batman Arkham serijalu, u novom Doom-u iz 2016. i Doom Eternal iz 2020. , koji svi koristi hijerarhični FSM. Hijerarhični FSM je jednostavno rješenje za problem čitljivosti. Umjesto jedne velike mreže stanja koji svi funkcioniraju međusobno, sustav je rastavljen u manje grupe, gdje se svaka fokusira na određeno ponašanje, sa samo par stanja koja idu između grupa.



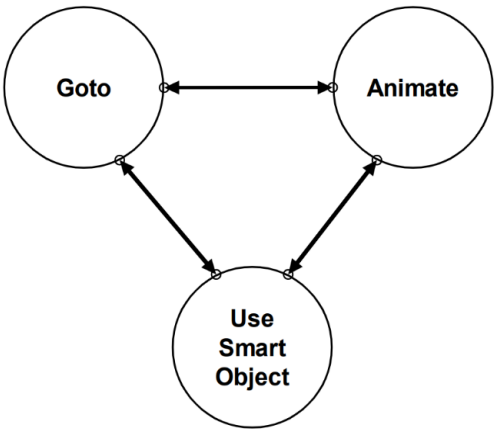
Slika 9. Slika (a)-hijerarhični (b)-obični FSM

Izvor: https://www.researchgate.net/figure/State-space-abstraction-a-hierarchical-FSM-and-b-equivalent-flat-FSM_fig6_3226043

Naravno same grupe također mogu brzo narasti i postati nečitljive i kompliciranе, usprkos tome važno je spomenuti da FSM sustav još uvijek korišten u industriji. Usprkos rijetkoj uporabi ovog sustava u današnjoj industriji oboje Arkham serijal i Doom serijal su hvaljeni za svoj AI, usprkos na pogled arhaičnom sustavu koji se danas ne koristi radi novijih tehnologija.

2.1.3. F.E.A.R I GOAP

Igrica F.E.A.R.(First Encounter Assault Recon) iz 2005. napravljena od strane Monolith studija je često hvaljena da posjeduje najbolju umjetnu inteligenciju među igricama iz ranih 2000-ih i kasnije. Gledajući njihovu osnovu logiku, sustav koristi FSM sa samo 3 stanja.



Slika 10. FSM sustav unutar F.E.A.R.

Izvor: Orkin, Jeff. (2006) Three States and a Plan: The A.I. of F.E.A.R. Monolith Productions / M.I.T. Media Lab, Cognitive Machines Group Stranica 2

Usprkos tome neprijatelji koriste GOAP(Goal Oriented Action Planning sustav) [2]. Dok postoje samo tri stanja u FSM-u, stanje „Goto“(hrv. Idi do) za pomicanje iz jedne lokacije na drugu, stanje „Animate“(hrv. Animiraj) za igranje animacije za potrebnu akciju, i „UseSmartObject“(hrv. Koristi pametni objekt) što nam znači interakcija sa svijetom okolo njih.

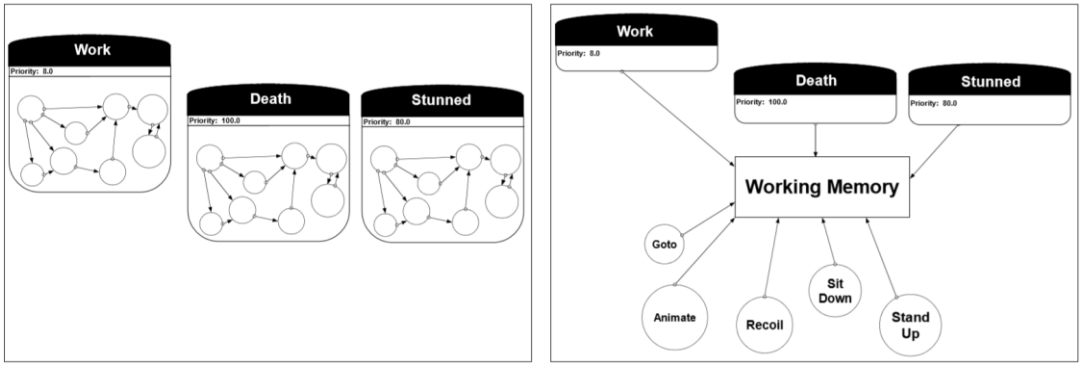
Na prvu ruku sustav FSM-a iz Half Life-a se čini isti kao GOAP sustav u FEAR-u, ali velika razlika stoji u njihovom načinu izvedbe svojih rasporeda, tj planova. Dok u Half Life-u FSM sustav implementira raspored za svakog individualnog neprijatelja, te on tada pokušava svoj raspored izvršiti do kraja, u GOAP sustavu prioriteta je suprotan. Npr, ako NPC u F.E.A.R-u ima zadatak „KillEnemy“(hrv. Ubi neprijatelja), taj je zadatak završni kondicional koji NPC želi izvršiti, te sve akcije između prvog stanja i završnog je promjenjivo i konstantno mijenjajuće da se ispuni završni kondicional, dok u Half Life-u raspored neprijateljskog ponašanja poprilično krut, NPC u F.E.A.R.-u ima početno stanje i cilj (eng. Goal) te koristi veliki broj definiranih akcija sa specifičnim kondicijama da taj cilj ispuni.

Ukratko, FSM je proceduralan, govori NPC-u procedure ili slijed akcija koje mora izvršiti, dok je GOAP deklarativan, daje NPC-u zadatak i nudi mu slobodu kako da taj zadatak izvrši. Glavna stavka ovog sustava planiranja je ponuditi više načina izvedbe zadatka. Sama logika iza ovog sustava planiranja dolazi iz STRIPS sustava, što stoji za STandford Research Institute Problem Solver. Sustav je razvijen 1970-ih na Standford Sveučilištu, sustav se sastoji od ciljeva i akcija, gdje su ciljevi stanje do kojeg želimo da svijet dođe a akcije su definirane s kondicijama i situacijama.

„Akcija se može izvršiti samo ako su kondicije zadovoljene, a svaka akcija mijenja trenutačno stanje svijeta.“ [11]

GOAP sustav je razvio Jeff Orkin ranih 2000.-ih te je uspješno implementiran u video igricu F.E.A.R. iz 2005. i njegove nastavke.

GOAP sustav ima sve dijelove FSM sustava, ali ih korisni na drugačiji način. FSM sustav funkcionira na predodređenim kondicijama koje se trebaju zadovoljiti da NPC promjeni svoje stanje, te prolazi kroz svoja stanja linearno, dok oboje FSM i GOAP koristi grafove, razlika je da su grafovi u GOAP-u odvojeni.

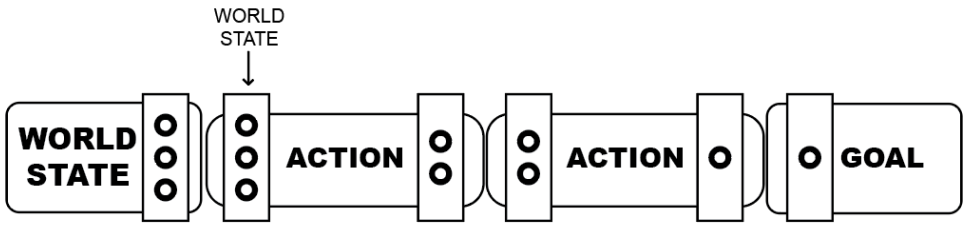


Slika 11. Graf FSM sustava i graf GOAP sustav

Izvor: Orkin, Jeff. (2006) Three States and a Plan: The A.I. of F.E.A.R. Monolith Productions / M.I.T. Media Lab, Cognitive Machines Group Stranica 8

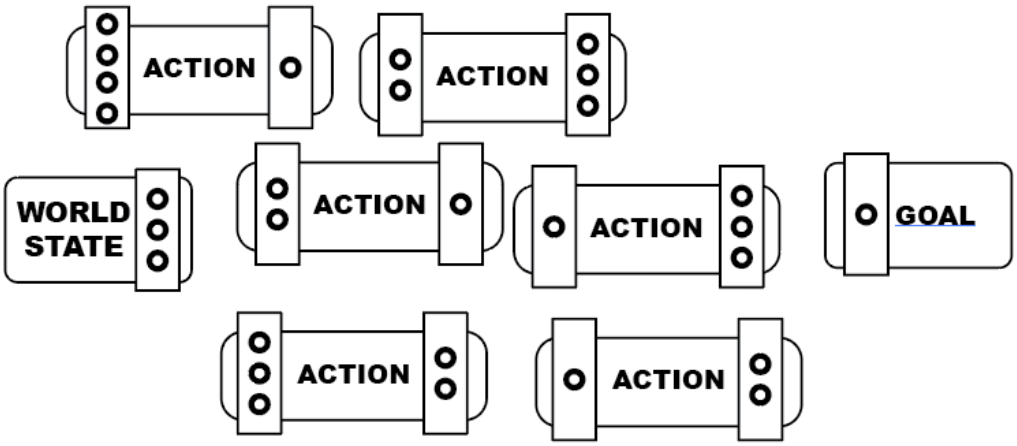
FSM ima određenu akciju da zadovolji cilj, GOAP može stvoriti više načina da izvrši isti cilj. Na grafovima vidimo da su akcije i ciljevi odvojeni u GOAP-u, te da zajedno prolaze kroz radnu memoriju. Velika prednost ovog sustava je puno lakša implementacija novih akcija i ciljeva.

Ali kako točno ovaj sustav bira svoje akcije? Uz pomoć kondicija i rezultata, svaka akcija koju GOAP sustav bira treba ispunjen kondicional stanja svijeta, te s izvršavanjem te akcije, to stanje se treba promijeniti, tako da druga akcija može biti odabrana, ovakvo slaganje akcija možemo vizualizirati kao slaganje domina.



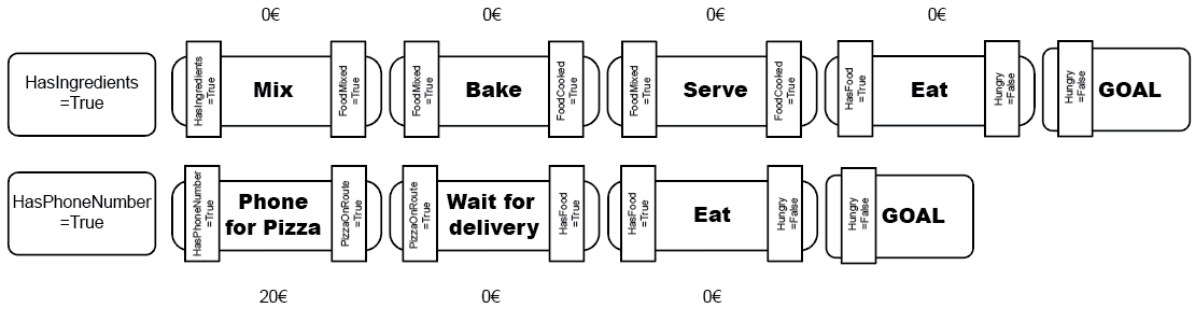
Slika 12. Slaganje akcija u GOAP-u

Uz pomoć ovakvog sustava možemo ubacivati veliki broj akcija između kojih onda sustav bira svoj raspored i slijed akcija.



Slika 13. Veliki broj akcija u GOAP sustavu

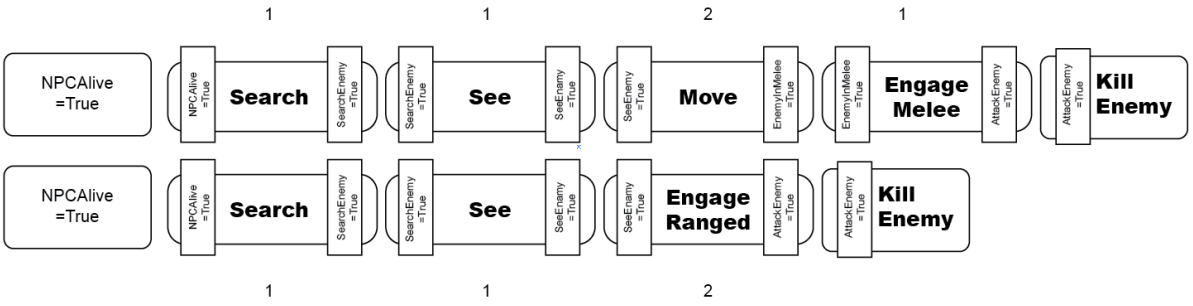
Veliki broj mogućih rasporeda postavlja novo pitanje, kako izabrati najbolji raspored? Često u svakoj situaciji postoji broj mogućih rasporeda, svi koji izvršavaju trenutni cilj, uz dodavanje cijena na akcije sustav uvijek bira najjeftiniju opciju. Kao primjer često se koristi situacija s hranom. Stanje svijeta je oboje „HasIngredients=True“ i „HasPhoneNumber=True“, što znači da su oba plana akcija validna, da pomognemo AI-u biranje između više rasporeda, uz same akcije vežemo i cijene akcija.



Slika 14. Cijene akcija unutar GOAP sustava

Sama cijena može biti promjenjiva, recimo da cijena u gornjem primjeru nije novčana već vremenska, sada opcija za spremanje hrane je skuplja, te se bira opcija s naručivanjem.

Cijene akcija se mijenjaju u svakom trenutku oviseći o parametrima svijeta oko njih. Recimo da NPC ima cilj „Kill Enemy“, akcije koje ima na raspolaganju su „Engage Ranged“ i „Engage Melee“, ako je igrač udaljen od NPC-a, akcija „Engage Ranged“ ima nižu cijenu nego „Engage Melee“, jer za pravednu izvedbu „Engage Melee“ NPC treba provesti i akciju „Move“ da uđe unutar dometa melee napada. Ako pri okršaju se igrač nađe blizu NPC-a sada akcija „Engage Melee“ ima nižu cijenu od „Engage Ranged“ jer su parametri zadovoljeni.



Slika 15. Cijene GOAP sustava u kontekstu video igrice

U primjeru možemo vidjeti kako zbog cijene akcije „Move“, raspored akcija za „Engange Ranged“ je niži od „Engange Melee“, ali kada bi isti scenario ponovili bez potrebe za akciju „Move“, prvi raspored bi imao nižu cijenu.

Ključna razlika između FSM-a i GOAP-a je da GOAP dopušta inteligenciji da sama složi raspored akcija koje će izvršiti da stigne do cilja, dok ako bi isti cilj bio implementiran u FSM strukturu, raspored akcija je već definiran. Možemo vidjeti da je razvoj umjetne inteligencije ide prema što više mogućih ishoda do kojih inteligencija može doći, sa što manje zatvorenih i strogo definiranih sustava.

2.1.4. HALO I INOVACIJA BEHAVIORAL TREES-A

Halo[6] je prvi put izašao 2001. godine za PC i Xbox, njegov nastavak, Halo 2 iz 2004. je već utemeljio serijal kao jedan od divova u industriji video igara. Inovacije poput limita od dva oružja, velika osnova na priči, regenerirajuće zdravlje i vizualno prepoznatljivi neprijatelji su tek od nekih pojedinosti zašto je Halo serijal poznat. Kupljen od strane Microsofta i postavljen kao ekskluzivan za njihovu Xbox konzolu, predstavljan je kao pionir uspješne multiplayer komponente na konzolama. Dok je to postigao radi dolaska stabilne i jeftine internetske veze za kućanstva, uz jeftinu i moćnu konzolu za igrice, uspješnost kampanje za jednog igrača često je akreditiran glavnom neprijatelju u serijalu, savezu vanzemaljskih rasa s različitim karakteristikama.

Velika prednost ovog antagonista je broj drastično različitih rasa protiv kojih se igrač bori. Bungie studio je svakoj rasi ovog saveza dodao različite osobnosti, načine ponašanja i reagiranja na okoliš, situacije i igrača. Gledajući kako se igrač susreće s više različitih rasa pri svakom okršaju, veoma brzo dolaze do izražaja osobnosti različitih neprijatelja.

Svaki od neprijatelja ima svoj način ponašanja, što igraču komunicira u koju se vrstu okršaja upušta u svakom danom trenutku. Uzimajući u obzir kako neprijatelji imaju i način ponašanja međusobno, daje svakom susretu veliki broj mogućih ishoda između neprijatelja i igrača.

Kritična komponenta Halo-ovog dizajna je njegovi okoliš, kojiječesto ogroman. Kao rezultat velika je pozornost na susretima, intenzivni periodi gdje igrač nailazi više neprijatelja koje treba poraziti. Pri ovakvim susretima oboje igrač i neprijatelji su postavljeni u lokacije povoljne za oboje, gdje se sukob može razviti na zanimljive načine.

Veliki prostori također omogućavaju i borbu s vozilima, dok su i druge igrice implementirale vozila u svoju igru, u Halo-u prisustvo vozila je samo još jedna varijabla s kojom neprijateljski AI surađuje. Razine s vozilima nisu odvojene, vozila nisu samo druga vrsta NPC-a koja je nama nepoznata, već neprijatelji ulaze u vozila i bore se s igračem bio on u vozilu ili van njega. Radi ovoga AI treba biti sposoban prilagoditi se kontekstu trenutne situacije.

Kao osnova ovog ponašanje, AI posjeduje brojna osnovna ponašanja:

- Idle – pasivni način ponašanja, često bez kontakta s igračem ili pri patroliranju okoliša
- Combat – borbeni način jednom kada je kontakt ostvaren, napada igrača, traži ga ili je u potjeri za njim
- Defensive – traži zaklon, pada na bolju, zaštićeniju poziciju, ili bježi glavom bez obzira ako se sve čini izgubljeno

Ova stanja ponašanja trebaju biti brza i reagirajuća ali također trebaju biti veoma fleksibilna u brojnim situacijama. AI treba biti sposoban mijenjati svoje ciljeve i primjereno odgovarati situaciji, uz sve to cijeli sustav treba biti neprimjetan od strane igrača. Dok je veliki broj elemenata zaslužan za ovako kompleksan sustav, ovdje možemo promotriti jedan od glavnih sustava zaslužan AI infrastrukturu, Behavioral Trees.

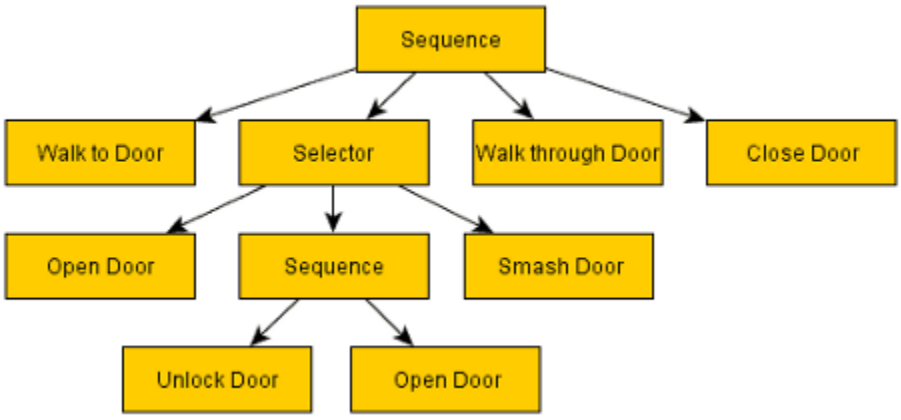
Stabla ponašanja(eng. Behavioral Tree) je najpopularniji način implementacije umjetne inteligencije u video igrice. Ovaj pristup dopušta razna vrste ponašanja iz jednog hijerarhijskog izvora, što dopušta veliku razinu čitljivosti i fleksibilnosti. Tehnika se pojavila

ranih 2000-ih u Halo igricama od Bungie softwarea te se koristi dan danas u istom serijalu i još mnogo drugih. Stabla ponašanja su postala toliko popularna i efikasna metoda da su implementirati u programe poput Unity-a i Unreal Engine-a. Stabla ponašanja dijele mnogo sličnosti s principom FSM-a, jer u prirodi oboje nude simbolički i apstraktni način identifikacije ponašanja koje želimo da naš NPC izvrši. Usprkos sličnosti, stabla ponašanja nude veći razinu čitljivosti i kontrole nad akcijama koje želimo implementirati.

Stabla ponašanja su složena u strukturu stabla, kako i ime implicira. Ova struktura nudi sustavu razinu hijerarhije što nudi veću broj opcija kroz istu akciju. Uz to nam daje veću kontrolu nad svakim individualnim izborom.

Kao i svako stablo, stablo ponašanja je sastavljeno od „grana“, uz pomoć tih grana se izbori množe u druge, manje grane ili u „listove“. Svi koji se naravno spajaju u glavnu središnju strukturu.

- „Leaf“ (hrv. List) završni čvor stabla, koje se ne grana dalje u više opcija.
- „Selector“(hrv. Odabirač) funkcija stabla koja služi kao „if“ deklaracija, bira prvu funkciju koja se može izvršiti. Često s ovom funkcijom dolazi broj granajućih opcija koje se zovu „children“(hrv. Djeca). Uz pomoć ove funkcije vrtimo opcije koje su nam ponuđene dok neka ne zadovolji sve kriterije koje treba ispuniti.
- „Sequence“(hrv. Sekvenca) funkcija koja nam diktira da izvršimo sve opcije u stablu, da umjesto prestanka na opciji koja nam zadovoljni kriterije i konstantnoj vrtnji iste, nakon što je ta izvršena(ili je propala) prelazimo na sljedeću opciju, i tako u krug.



Slika 16. Jednostavno stablo ponašanja.

Izvor: https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work

Slika iznad je prikaz jednog jednostavnog stabla, kroz koji možemo vidjeti kako sustav funkcionira na primjeru NPC-a koji prolazi kroz vrata.

Sve se grana iz prvotne sekvenca ćelije, te hijerarhični od gore prema dolje, i od lijevo prema desno, prolazi kroz sve funkcije.

Prva funkcija koja se izvršava je definitivno „Walk to Door“(hrv. Hodaj do vrata), prva je u hijerarhiji od lijevo prema desno. Dakle prva stvar koji NPC radi je prilazi vratima, nakon što je to izvršeno, zbog prvotne ćelije koja je sekvenca, izvršava se sljedeća akcija.

Sljedeća ćelija je „Selector“ ćelija što znači da ova grana stabla traži prvu funkciju koja se može izvršiti, te ostaje na njoj. Na ovom primjeru vidimo da je prva stvar koju NPC pokušava je „Open Door“, dakle pokušava otvoriti vrata, ako je ta funkcija zadovoljena, „Selector“ je našao svoju funkciju, i vraća se na prvotnu „Sequence“ granu. Naime ako su vrata zaključana i „Open Door“ čvor je neuspješan, „Selector se prebacuje na sljedeći čvor u svojoj hijerarhiji, što je „Sequence“ čvor s dvije opcije, „Unlock Door“ i „Open Door“.

Ovdje vidimo kako se oboje funkcije mogu koristiti u više navrata, naime radi „Sequence“ čvora, prvo se vrata pokušavaju otključati te onda nakon toga, „Open Door“ funkcija za otvoriti ista. Ako je NPC imao ključ za vrata i uspio uspješno otvoriti vrata, „Selector“ čvor našao svoju funkciju koja funkcionira, te prelazi na sljedeći čvor.

Ali ako su vrata ostala zaključana, te prijašnje dvije funkcije „Selektor“ čvora su označena kao neuspjesi, „Selector“ prelazi na sljedeću i zadnju granu „Smash Door“, te se vrata razbijaju.

Nakon uspješno izvršene „Selector“ čvora, prelazimo na završne dvije „Walk through Door“ i „Close Door“, gdje NPC prolazi kroz vrata i zatvara ih nakon sebe.

Gledajući unazad od prvotnog FSM sustava možemo vidjeti razvoj ovih tehnologija i njihov napredak u načinu na koji se prilazi umjetnoj inteligenciji, gdje je cilj automatizirati što veći broj odluka koje NPC može napraviti.

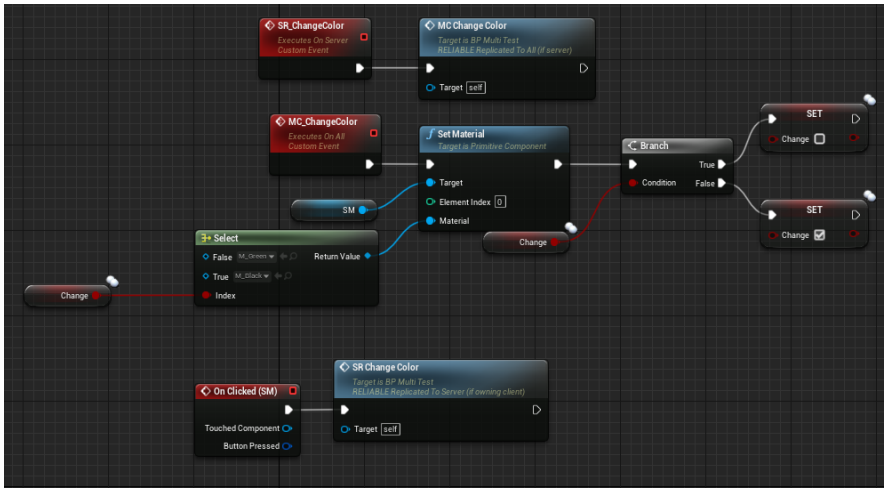
2.2. UNREAL ENGINE

Unreal Engine [8] je napravljen od strane Epic games. Preko 90% koda za Unreal Engine 1 je napisan od strane Tim Sweeney-a koji je ujedno i osnivač Epic Gamesa. Rad na Unreal Engineu počinje 1995. te prvi put izlazi na tržište s video igricom „Unreal“ u 1998. Ubrzo „Unreal“ postaje jedna od igrica s najboljom grafikom na tržištu te su među prvima zajedno sa id Software-om koji koriste potpuni 3D okoliš u svojim igrama. Radi svojih mnogih inovacija Unreal Engine je postao veoma popularan program za razvoj budućih video igara, te svaka nova inačica programa ima veliki broj naslova. Kroz neprestani razvoj Unreal Engine 5 je danas među najboljim i najkorištenijim programima za pogon video igara, te je besplatan za uporabu preko Epic Storea. Radi njegove kvalitete, lakoće pristupa i brojnoj dokumentaciji, praktični dio rada je napravljen u Unreal Engineu 5.

2.2.1. BLUEPRINTS

Godine 2014. na tržište izlazi Unreal Engine 4, te s njim dolazi nova inovacija u pristupačnosti programa. Blueprints [10], novi sustav za vizualno skriptiranje. Blueprints

sustav dopušta korisniku stvaranje koda uz pomoć sustava interaktivnih čvorova. Unreal Engine je još uvijek baziran na C++ kodu, te je korisniku dana opcija između Blueprints ili C++ kodiranja pri otvaranju svakog novog projekta, Blueprints sustav omogućava stvaranje koda korisnicima koji ne posjeduju ikakvo prijašnje znanje o kodiranju ili C++ jeziku.



Slika 17. Primjer Blueprints sistema za vizualno skriptiranje.

Izvor: https://stackoverflow.com/questions/68656875/trying-to-understand-unreal-engine-4-replication

Dok je Blueprints sustav izvrstan za približavanje mogućnosti Unreal Enginea publici koja nije vrsna u kodiranju, vizualni sustavi kodiranja sputavaju korisnike koji već znaju programirati. Ovaj slučaj nije isključiv za Blueprints sustav, već je standard pri svakom pokušaju da se programskom jeziku pridoda grafičko sučelje, usprkos tome Blueprints sustav je veliki razlog koji pridaje popularnosti i raširenosti Unreal Enginea.

Dok Unreal Engine 4 i Blueprints sustav dolazi na tržište 2014. drugi veliki razlog popularnosti Unreal Enginea dolazi 2015. godine kada Unreal Engine postaje besplatan za sve korisnike. Dok je prije pristup bio zatvoren iza dozvole za korištenje koja se treba platiti unaprijed, sada je Unreal Engine postao besplatan za sve korisnike te jedino zahtjeva plaćanje ako se izda video igrica na njemu. Promjena ove poslovne paradigme se možda čini neznačajna, ustvari je glavni razlog za ogroman rast Unreal Engine zajednice koja nudi veliku količinu smjernica, projekata te korisne dokumentacije. Prijašnje inačice Unreal Enginea zahtijevaju veliku financijsku predanost te nudi korisniku isključivo profesionalnu dokumentaciju, sada korisnik može pronaći pomoć u veoma razvrstanoj zajednici koja se specijalizira na veliki broj aspekata Unreal Enginea.

Uzimajući u obzir ove činjenice, jedna od velikih značajka ovog rada je demonstracija pristupačnosti jednom korisniku koji se prvi put susreće sa sistemima prisutnim u Unreal Engineu.

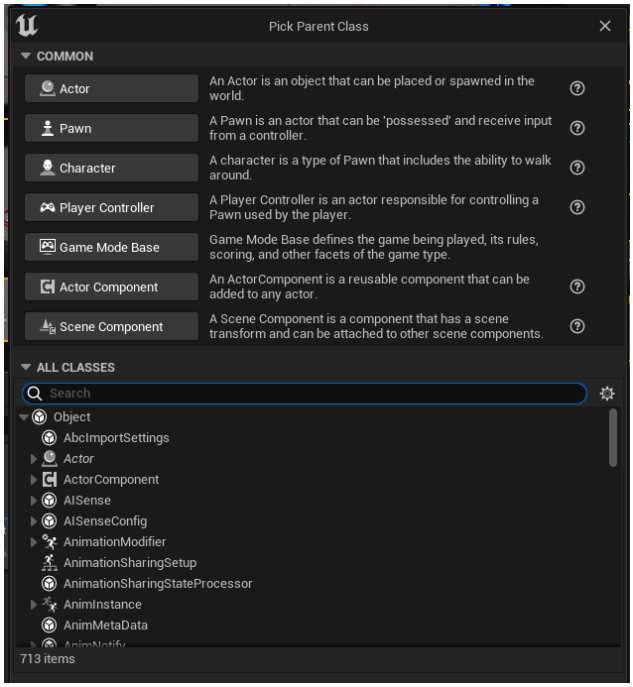
Bluprints sustav koristi čvorove za spajanje funkcija, tok koda možemo pratiti s linijom koja spaja čvorove sa strjelicom.

Slika 18: Linija izvedbe koda.

Spajajući više čvorova stvaramo “Događaj” (eng. Event) te pretećи njegovu izvedbu možemo pratiti slijed događaja. Svi kodovi počinju s čvorom s crvenim rubom, te je on pokretač samog slijeda. Kraj slijeda dolazi kada čvor ne može izvesti traženu radnju, ili prestanemo spajati čvorove. Svaki objekt koji sadrži u sebi blueprint kod se zove “Blueprint Actor”.

2.2.2. BLUEPRINT GLUMAC

Blueprint glumac[14] (eng. Blueprint actor) je svaki objekt koji je moguće staviti u okoliš projekta. Ovi objekti čine većinu dodatka ovog projekta.



Slika 19. Blueprint Actor izbornik.

Izbornik među češće korištenim glumcima nudi:

- Actor - (hrv. Glumac) objekt koji može biti postavljen ili stvoren u okoliš.
- Pawn - (hrv. Pijun) glumac koji može biti pod “opsjednut” i primati informacije od kontrolera.
- Character - (hrv. Karakter) vrsta pijun glumca koji se može kretati po okolišu.

- Player Controller - (hrv. Igračev kontroler) glumac koji igrač koristi za kontrolu nad karakterom.
- Game Mode Base - (hrv. Baza vrste igre) definira igru koja se igra, pravila, bodovanje, i ostale značajke.
- Actor Component - (hrv. Komponenta glumca) komponenta koja se može ponovno iskoristiti i dodati drugim glumcima.
- Scene Component - (hrv. Komponenta scene) komponenta koja sadrži transformaciju scene i može biti dodana drugim komponentama.

Svaki od navedenih glumaca ima svoju ulogu, projekt nadodaje većinom glumac objekte za postizanje svojeg cilja.

3. PAKTIČNI DIO

U razradi rad prolazi kroz dijelove projekta koji su promijenjeni ili dodani u standardni paket koji dolazi uz Unreal Engine 5.2. Cilj projekta je nadograditi standardni paket za pucačinu iz prvog lica u projekt koji sadrži NPC-eve čija umjetna inteligencija im dopušta konzistentnu međusobnu interakciju. Radi prirode sustava koji je zatvoreni sustav, nema jednostavnog načina za početi objašnjavati sve komponente, gledajući da prvi sustav također funkcionira sa zadnjim čije objašnjenje dolazi tek nakon broj stranica.

3.1. POČETNI PAKET

Paket koji projekt nadodaje se zove "First Person Shooter Template"[15] (hrv. Šablona za pucačinu iz prvog lica). Paket uključuje:

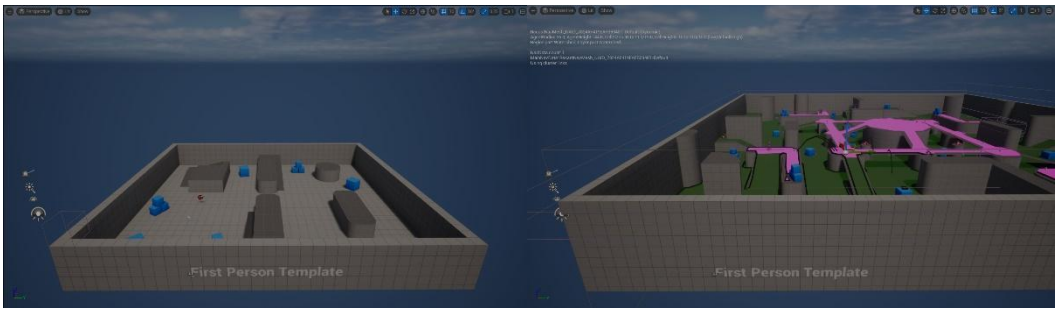
- Karakter u prvom licu nad kojim igrač može preuzeti kontrolu za kretanje i pucanje.
- Oružje koje se može pokupiti i ispaljivati projektile.
- Okoliš s osnovnom geografijom (rampe, platforme).
- Kocke koje reagiraju na podražaj igrača ili projektila.

Ovaj paket je dostupan u više inačica Unreal Engine-a, te postoje sitne razlike u kodu i funkcionalnosti, npr. u Unreal Engine 4 igrač se stvori sa oružjem, ne treba ga pokupiti, usprkos tome premisa osnovne šablone ostaje ista.

Projekt u osnovnu šablonu nadodaje veliki broj stvari. Glavne od njih bijući:

- Povećana mapa s više geografije (mapa je povećana da poveća broj smjerova i puteva koji umjetna inteligencija može izabrati).
- Navigacijska mreža s različitim prolaznim vrijednostima.
- Povećan broj kocki koje reagiraju na podražaj igrača i NPC-a.
- Četiri NPC-a koji patroliraju po mapi i ulaze u okršaje jedni sa drugima i s igračem.
- Umjetna inteligencija koja se bavi ponašanjem NPC-a.
- Environmental query system (hrv. Sustav za okolišno preusmjeravanje) koji određuje smjer i cilj kretanja NPC-a.
- Sustav za ponovno oživljavanje igrača i NPC-a.
- Sustav zdravlja za igrača i NPC-a.
- Sustav za brojanje bodova koje je igrač postigao u jednom životu (igra je završena jednom kada igrač postigne 10 bodova).

Uz sve ovo u projektu je također promijenjen veliki broj sitnica i vrijednosti koji ozbiljno mijenjaju način igre, sve promjene su navedene kroz daljnju razradu.



Slika 20. First Person Template prije i nakon.

Uz veliki broj nadodanih sustava, također postoji broj sistema koji nisu postignuti usprkos prvotnom planu da se uvedu u projekt, neki od njih bijući.

- Sustav za izračun metaka u oružju i sustav za ponovno punjenje (trenutno oružje ima beskonačno metaka).
- Dodatna oružja koja se mogu pokupiti s različitim načinima paljbe i količine štete koju uzrokuju.
- Paketi za zdravlje za NPC-ove (trenutno reagiraju samo na igrača).
- NPC s jasnim naznakama tima (trenutno svi dijele isti model).
- Lista bodova za sve sudionike (trenutno se broji samo igrač).

Unutar svakog glumca vidljiv je sav kod koji glumac koristi kao blueprint čvorovi. Razrada prolazi kroz slijed događaja koda i nudi dodatna objašnjenja.

Rad prvo prolazi kroz glavna tri komponenta umjetne inteligencije. “NPC_AI” glumac sadrži percepciju samog NPC-a te mu daje informacije o okolišu i lokaciji neprijatelja oviseći o podražajima koji su mu dostupni. Informacije dobivene s ovim glumcem se koriste u drugim komponentama, poput stabla ponašanja, koje se koristi za patrolu okoliša, napad i povlačenje s neprijateljem. Treća komponenta je sami NPC karakter koji sadrži oznaku tima i sustav za praćenje zdravlja.

Ova tri komponenta čine osnovicu umjetne inteligencije i projekta.

3.2. NPC_AI – SUSTAV ZA PERCEPCIJU

“NPC_AI” se bavi percepcijom[16] samog NPC-a, ovaj sustav određuje kako NPC dolazi do raznih vrijednosti koje se koriste unutar stabla ponašanja i interakcije sa igračem. Projekt koristi četiri sustava za percepciju:

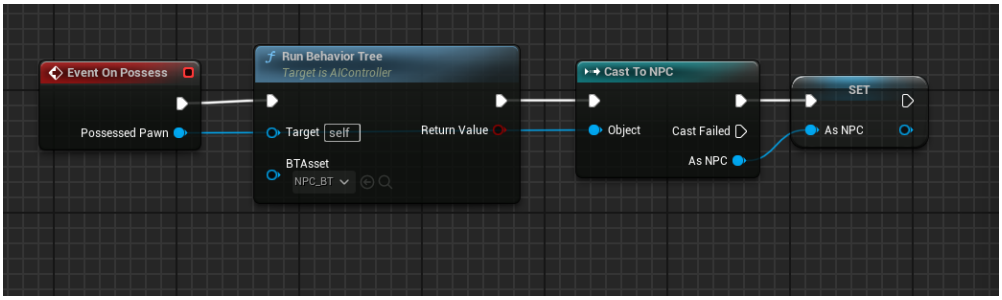
-Vid

-Sluh

-Štetu

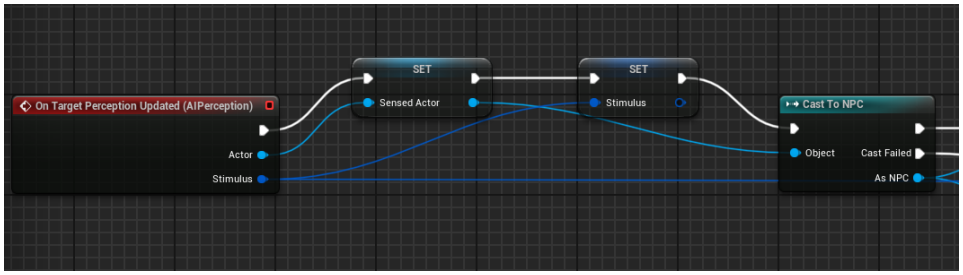
-Predviđanje

Ova komponenta je usko povezana sa stablom ponašanja, informacije koje nam percepcija pruža se koriste kao vrijednosti crne ploče unutar stabla, više o tome kasnije.



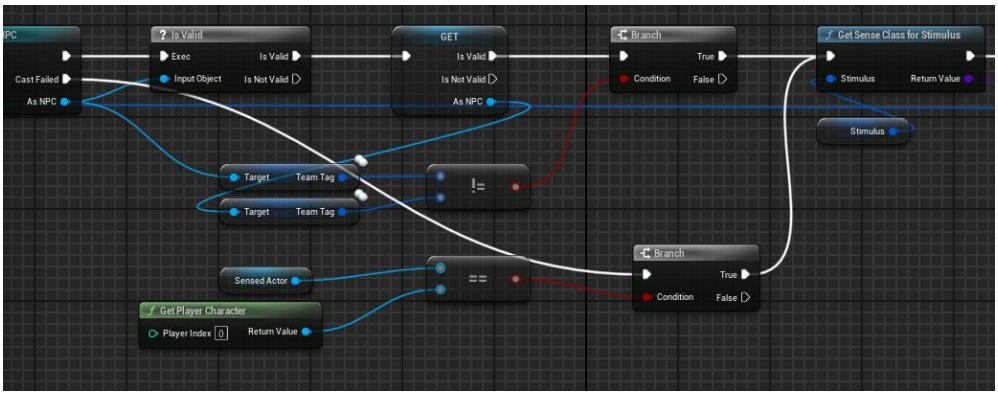
Slika 21: Slijeda događaja "Event on Possess" unutar NPC_AI.

Prvi slijed koji se poziva je "Event on Possess" (hrv. Događaj na posjedovanje) te on počne svaki put kad se novi NPC stvori u okoliš. Čvor "Run Behavior Tree" (hrv. Pokreni stablo ponašanja) aktivira stablo ponašanja na našem novom NPC-u, zatim u sljedeća dva čvora koristi "Cast to NPC" (hrv. Prebaci na NPC) čvor tako da se "NPC" može koristiti informacijama koje ovaj glumac dobavlja.



Slika 22: Prvi dio slijeda "On Target Perception Updated(AIPerception)".

Sljedeći dio sistema se čini kompliciran, ali većinom se bavi definiranjem identiteta i postavljanjem vrijednosti. Prvi dio koda postavlja vrijednosti za "Sensed Actor" i "Stimulus", što su oboje informacije koje su potrebne za događaj "On Target Perception Updated(AIPerception)" ovaj čvor se koristi za uzastopno postavljanje novih vrijednosti kako se one pojavljuju u okolišu.



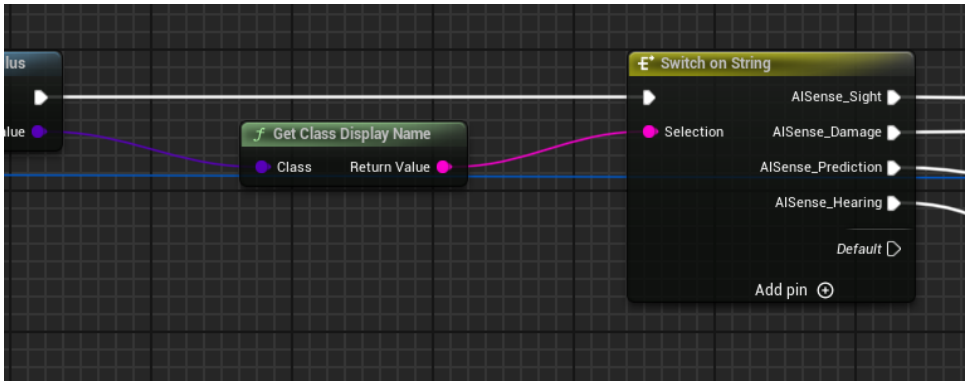
Slika 23: Drugi dio slijeda "On Target Perception Updated(AIPerception)".

Sljedeći čvor, “Cast to NPC” šalje informaciju o “Sensed Actor” u “NPC” sustav koji je dalje koristi. Čvor “Cast to” se koristi za slanje informacija jednog blueprint glumca do drugoga.

Nakon potvrde informacija o stanju “NPC” sistema sa “Is Valid” čvorom, kod nam prolazi kroz dvije provjere, prvo zahtjeva da ako je glumac kojeg je detektirao NPC, da je tim tog NPC-a drugačiji od tima NPC-a koji obavlja provjeru, druga opcija zahtijeva da je glumac kojeg je detektirao sam igrač, treća opcija ne postoji.

Kada NPC ima svoju metu, koja je ili neprijateljski NPC ili igrač, slijed se nastavlja dalje.

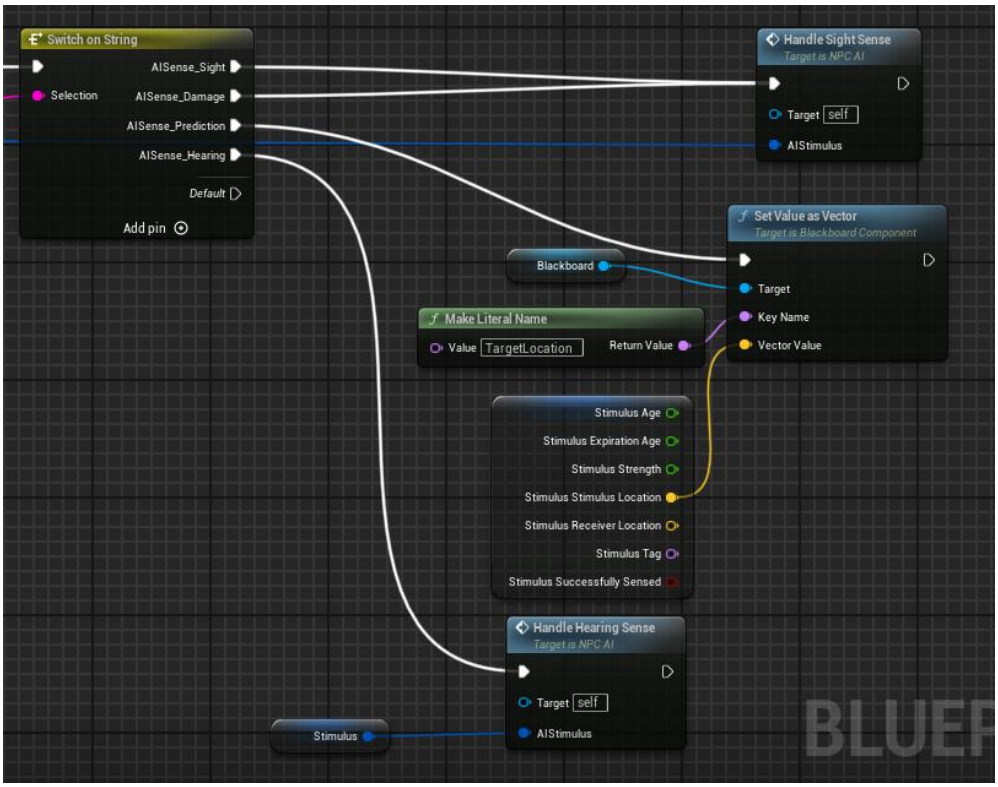
Nakon ove provjere slijed dolazi do čvora “Get Sense Class for Stimulus” (hrv. Dohvati klasu osjećaja od stimulansa).



Slika 24: Treći dio slijeda “On Target Perception Updated(AIPerception)”.

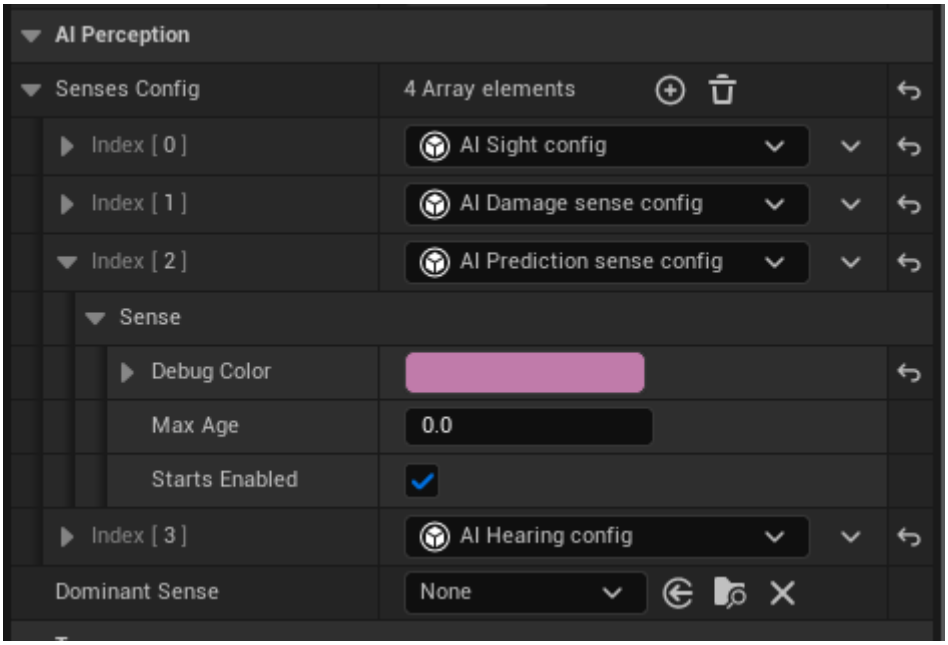
Uz pomoć čvora “Get Class Display Name” (hrv. Dohvati ime klase) informaciju o stimulansu mijenjamo u string informaciju. Nakon što je dohvaćena informaciji o stimulansu, uz pomoć “Switch on String”(hrv. Promjeni na string) čvora, preusmjerava sljedeću radnju ovisno o vrsti stimulansa. U ovom radu “AI_Perception” (hrv. AI percepcija) ima 4 vrste stimulansa na koje reagira.

- AISense_Sight(hrv. AI osjet na pogled)
- AISense_Damage(hrv. AI osjet na štetu)
- AISense_Prediction(hrv. AI osjet predviđanja)
- AISense_Hearing(hrv. AI osjet sluha)



Slika 25: Četvrti dio slijeda "On Target Perception Updated(AIPerception)".

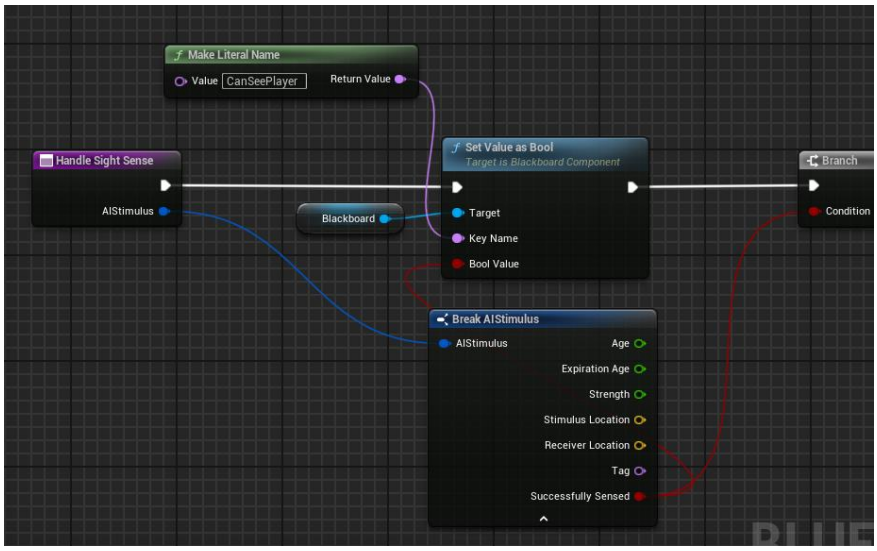
Dok osjeti "AISense_Sight" i "AISense_Damage" funkcioniraju jednako te se nastavljaju u čvor "HandleSightSense", "AISense_Prediction" i "AISense_Hearing" funkcioniraju po drugačijoj logici i postavljaju drugačije informacije.



Slika 26: AIPerception sadržaj.

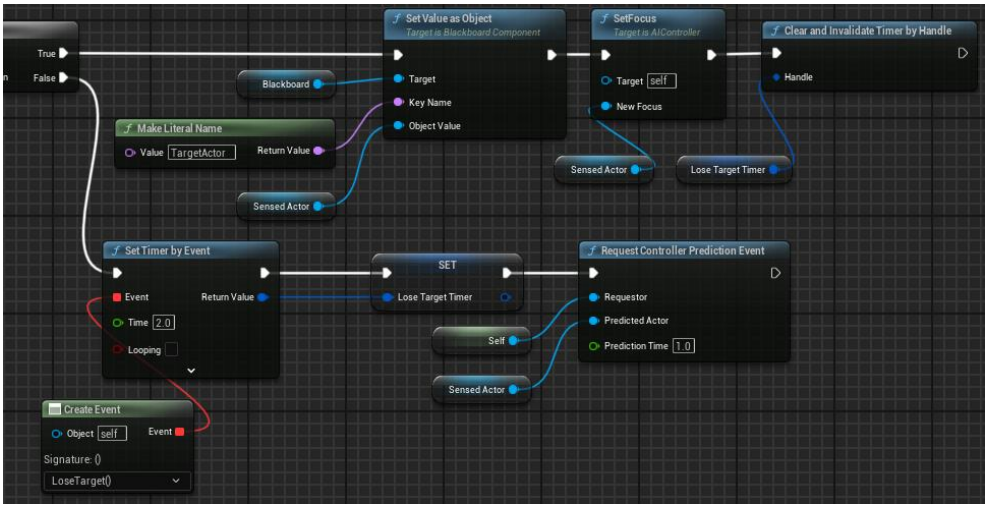
3.2.1. HANDLE SIGHT – SUSTAV PERCEPCIJE VIDA

Postavke za “AISense_Sight” i “AISense_Damage” su iste, ali također i malo kompliciranje, postavljaju više informacija i također trebaju imati opciju za gubitak linije vida.



Slika 27: Prvi dio slijeda “Handle Sight Sense”.

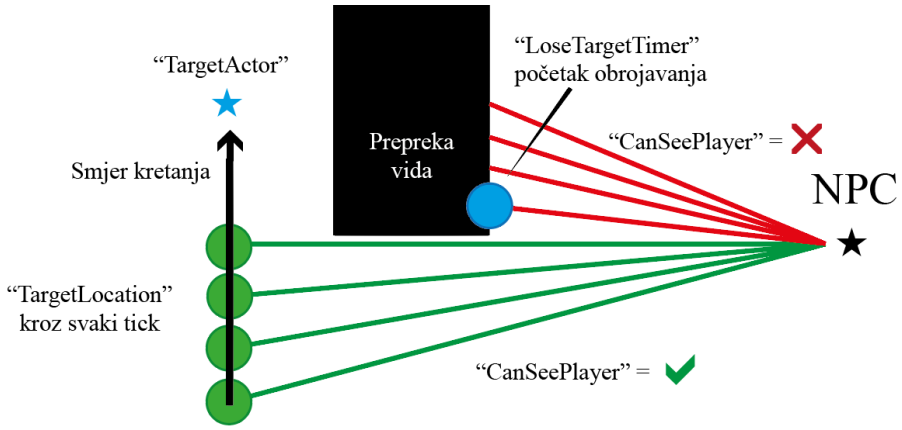
Prva stvar koja se događa je postavljanje Bool vrijednosti “CanSeePlayer” unutar crne ploče sa “Set Value as Bool” (hrv. Postavi vrijednost kao bool). Koristi funkciju “Break AI Stimulus”(hrv. Slomi AI osjet) jer nije potrebno imati sve informacije o podražaju, potrebna je samo binarna informacija koja nam dolazi preko boolean vrijednost. Nakon toga ulazi u “Branch” (hrv. Grana) što nam daje dva izbora radnje oviseći o boolean kondiciji koja je priključena u “Condition”(hrv. Kondicija).



Slika 28: Drugi dio slijeda “Handle Sight Sense”.

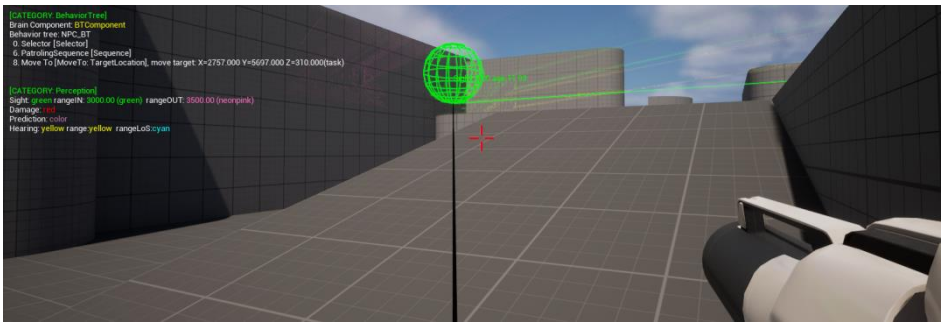
Ako je kondicija zadovoljena, i ima pozitivan stimulus, dakle ima pogled na metu, slijed nastavlja postavljati informacije. “Set Value as Object” (hrv. Postavi vrijednost kao objekt) postavlja glumca kojeg vidimo kao “TargetActor” unutar crne ploče. Nakon toga “Set Focus” postavlja glumca u osjetu kao fokus “AI Controller”-a, te zadnje uklanja nam mjerač vremena sa “Clear and Invalidate Timer by Handle” (hrv. Očisti i poništi mjerač vremena po ručci) mjerač je vrijednost koja je postavljena unutar slijeda ako nemamo stimulus.

Ako kondicija nije zadovoljena, i nema stimulus, dakle nema pogleda na metu, čvor do kojeg prvo slijed dolazi je “Set Timer by Event” (hrv. Postavi mjerač vremena po događaju) događaj koji postavlja ovaj mjerač je novi događaj koji smo stvorili pod imenom “LoseTarget”, mjerač ima traje 2 sekunde, dakle ako NPC nema pogled na metu 2 sekunde, meta je izgubljena. Nakon toga se postavlja mjerač pod imenom “Lose Target Timer” te dolazi do čvora “Request Controller Prediction Event” (hrv. Pozovi kontroler za predikciju) što znači nakon što NPC izgubi pogled na metu, pokušava predvidjeti gdje će biti za 1 sekundu u budućnosti. Vrijednosti za mjerač vremena i koliko daleko u budućnost će pokušati predvidjeti su promjenjive na samim čvorovima tih događaja.



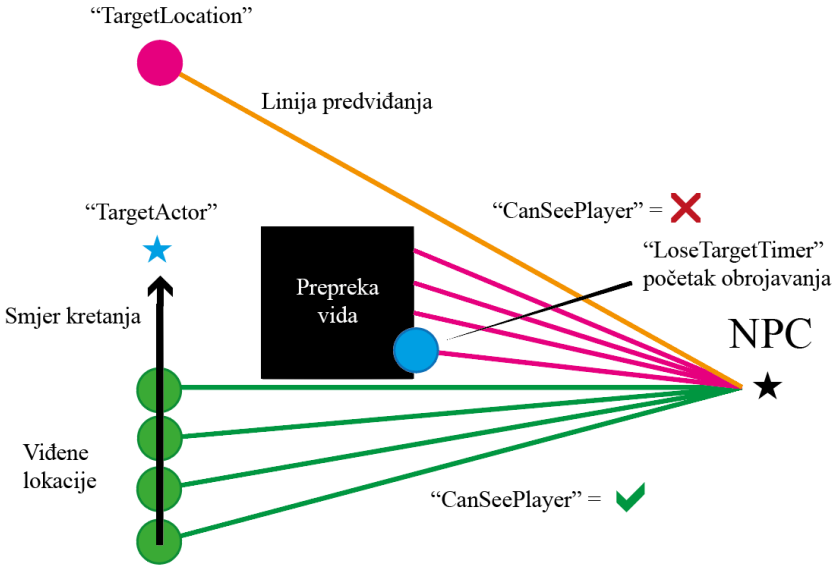
Slika 29: Primjer podražaja vida.

Objašnjenje za vizual percepcije se nalazi unutar glumca “EngageService” koji je također u dijelu sa stablom ponašanja.

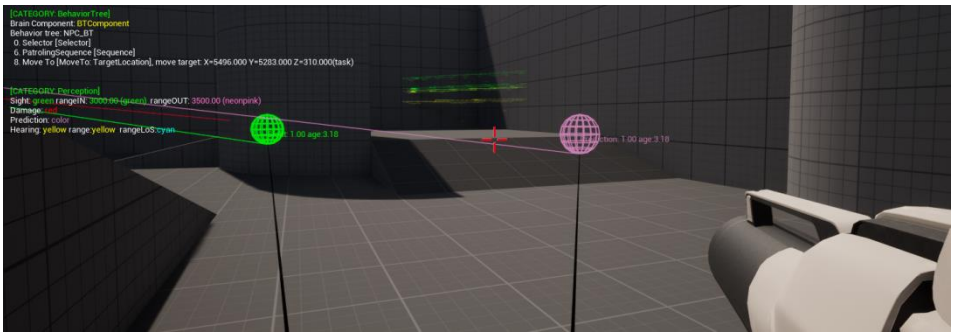


Slika 30: Primjer podražaja vida u okolišu.

“AISense_Prediction” je osjet koji kalkulira brzinu i smjer kretanja mete te predviđa gdje će se meta pojaviti za određeni broj sekundi nakon što se izgubi linija vida s metom. Većina ovog osjeta je već definirana unutar Unreal Engina-a te je postavljanje jednostavno, slijed se nastavlja u “Set Value as Vector”(hrv. Postavi vrijednost kao vektor), postavlja informaciju koju dobiva od stimulusa unutar komponente crne ploče te postavlja ime te informacije kao “TargetLocation”.



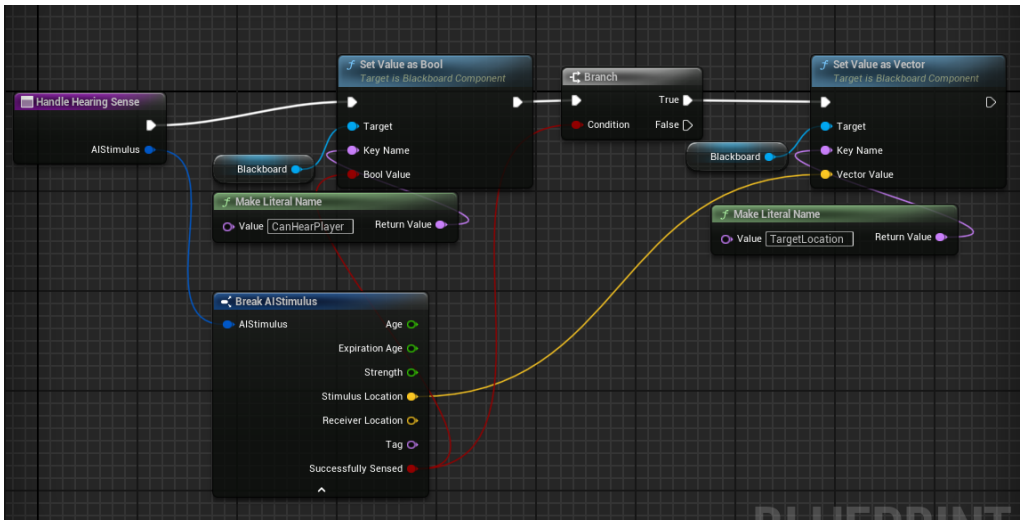
Slika 31: Primjer predviđanja kretanja mete.



Slika 32: Primjer predviđanja unutar okoliša.

3.2.2. HANDLE HEARING SENSE – SUSTAV PERCEPCIJE SLUHA

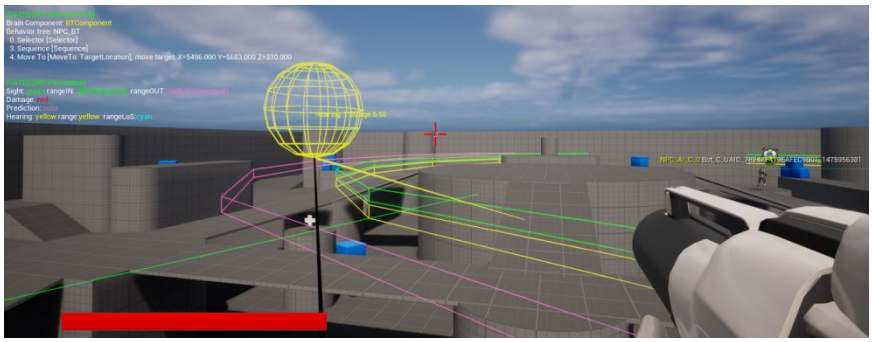
Osjet za zvuk također funkcionira drugačije, ali za razliku od njega on dolazi do izražaja jedino kada NPC nema pogled na igrača i koristi svoju sekvencu za patroliranje.



Slika 33: "Handle Hearing Sense".

Slično poput osjeta za pogleda, prvu stvar koju ovaj slijed postavlja je bool vrijednost od "CanHearPlayer" na komponenti crne ploče. Ako je kondicija zadovoljena, dakle stimulus je pozitivan, postavlja se druga vrijednost kao vektor, "TargetLocation", čiju vrijednost može povući iz "Break AIStimulus" čvora. Vektor vrijednosti sadrže koordinate unutar okoliša.

Slika 34: Primjer podražaja zvuka.



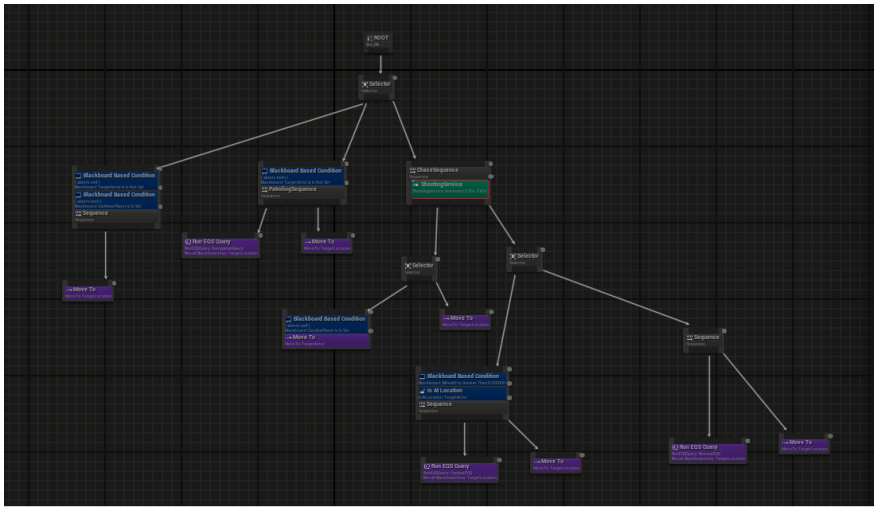
Slika 35: Primjer podražaja zvuka u okolišu.

3.3. STABLO PONAŠANJA I CRNA PLOČA

Stablo ponašanja [17](eng. Behavioral Tree) je jedan od ključnih komponenata za realizaciju umjetne inteligencije kod NPC-a. Stablo određuje glavnu logiku koja se izvršava sudeći po

kriterijima njemu zadanim. Logika stabla uvijek izvodi akcije od lijevo prema desno, te od gori prema dolje.

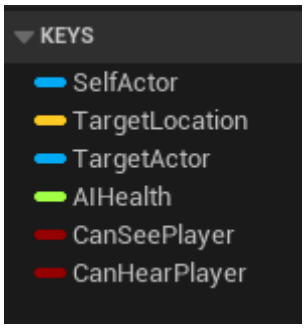
Stablo ponašanja ovog projekta izgleda ovako:



Slika 36. Stablo ponašanja korišteno u projektu.

Na stablu možemo vidjeti da grane ili listovi stabla imaju plave, ljubičaste ili zelene dodatke, ti dodaci se zovu “Ukrasi” (eng. Decorators) te se koriste za dodavanje konteksta, filtera ili kondicija za bolje izvršavanje akcija. Informacije koje se koriste u ukrasima se pohranjuju u Crnoj ploči (eng. Blackboard).

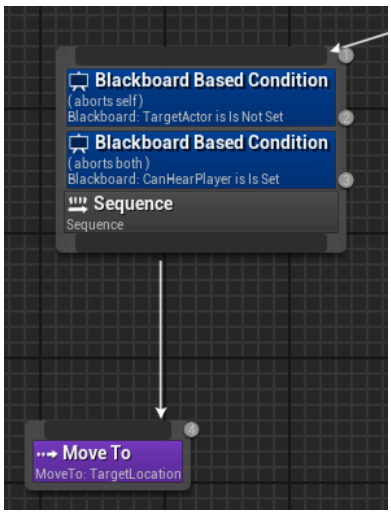
Crna ploča[18] se koristi za zapisivanje kratkotrajnih i lako izmjenjivih informacija koje pomažu pri izvršavanju umjetne inteligencije. Informacije zapisane u crnoj ploči se zovu Keys (hrv. Ključevi), te ih je moguće mijenjati i nadograđivati tijekom izvedbe programa. Svaki ključ može biti jedan od brojnih vrsta podataka, koji su označeni različitim bojama, crveni ključ stoji za boolean vrijednost što znači da ima samo vrijednosti “Istina” (eng. True) i “Krivo” (eng. False), ova vrijednost se koristi za izražavanje binarnih vrijednosti. Zelena je za “Plutajuću” (eng. Float) vrijednost što označava brojevnu informaciju, žuta označava vektorsku vrijednost dok je plava najčešće Blueprint vrijednost, dakle kao referencu drži metu s Blueprint vrijednosti.



Slika 37. Ključevi unutar crne ploče korišteni u projektu.

Kombinacija crne ploče unutar logike stabla ponašanja dopušta veliku kontrolu nad akcijama koje stablo određuje. Na slici možemo vidjeti četiri odvojene grane koje stablo posjeduje, i

gledajući logiku kojom stablo funkcionira, možemo vidjeti veliku većinu logike kojom jedan NPC funkcionira. Prolazeći sve grane od lijevo prema desno, te istražujući njihove ukrase možemo jednostavno dokučiti sustav prioriteta, kondicija i okidača koji funkcionira unutar grana stabla.



Slika 38. Prva grana unutar stabla.

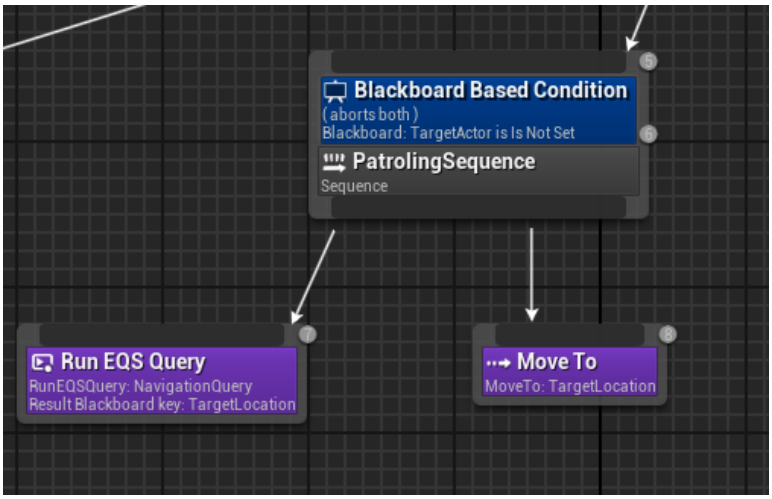
Prva grana koja ima se prva izvršava dolazi iz "Selector" (hrv. Birač) čvora, što znači da se ova grana vrti dok god ima funkciju za izvesti koja je dopuštena, dok "Sequence" (hrv. Slijed) čvor izvrši dopuštenu funkciju jednom te nakon toga prelazi na sljedeću funkciju.

Nakon toga dolazimo do čvora sekvence s dva ukrasa crne ploče(eng. Blackboard Based Condition) koji vodi u jednu „Move To" (hrv. Premjestiti se) funkciju. Prva kondicija koju ukras postavlja je „TargetActor is Not Set" što bi značilo da se ova grana može pokrenuti samo ako ključ crne ploče pod imenom „TargetActor"(hrv. GlumacMeta) nije postavljen, dakle da je njegova vrijednost „Not Set", nepostojeća. Dok druga kondicija traži „CanHearPlayer is Is Set" što prati istu logiku, ključ crne ploče pod imenom „CanHearPlayer"(hrv. MožešČutiIgrača) treba biti postavljen, treba imati vrijednost „Is Set".

Tek kada su obje kondicije valjane, funkcija „Move To" se izvršava s ciljem namještenim na „TargetLocation"(hrv. LokacijaMeta) vrijednost. Gledajući kako funkcija dolazi iz čvora sekvence, to znači da ova funkcija uvijek iznova prolazi ove dvije kondicije prije izvršavanja.

Također u plavim ukrasima su vidljive stavke „(aborts self)"(hrv. Abortiraj sebe) i „(aborts both)"(hrv. Abortiraj obje) što dopušta grani da mijenja vrijednosti ovih ključeva samostalno oviseći o trenutnoj situaciji. Ove opcije dopuštaju grani da prekine djelovanje ako naiđe na igrača i vrijednost "TargetActor" postane pozitivna, tj "Is Set". Bez ove mogućnosti grana bi se odigrala do kraja, sa "TargetLocation" postavljenim na izvor zvuka, umjesto na igrača.

Uzimajući u obzir koje se vrijednosti se koriste u ovoj grani i u kojim kondicijama, može se zaključiti da je ova grana za situaciju kada NPC ne vidi igrača, ali ga čuje, te radi toga mijenja lokaciju, dakle prva grana stabla isključivo koristi podražaj za zvuk.

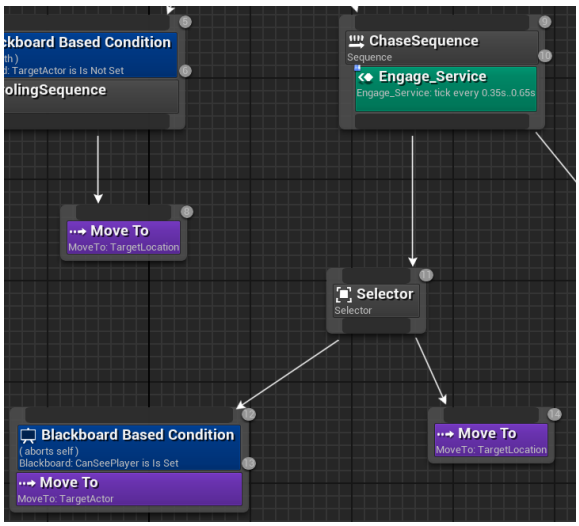


Slika 39. Druga grana unutar stabla.

Druga grana se bavi s patroliranjem, te se izvršava kada „TargetActor is Not Set“, dakle nemamo „TargetActor“ vrijednost. Zatim sekvenca izvršava funkciju „Run EQS Query“ što pokreće „Enviormental Query System“ (hrv. Sustav za okolišno preusmjeravanje), EQS koji je odabran ovdje se zove „Navigation Query“ (hrv. Sustav za navigaciju“) te od toga dobije rezultat koji je ključ zvan „TargetLocation“.

Sustave za okolišno preusmjeravanje rad prolazi nakon stabla ponašanja.

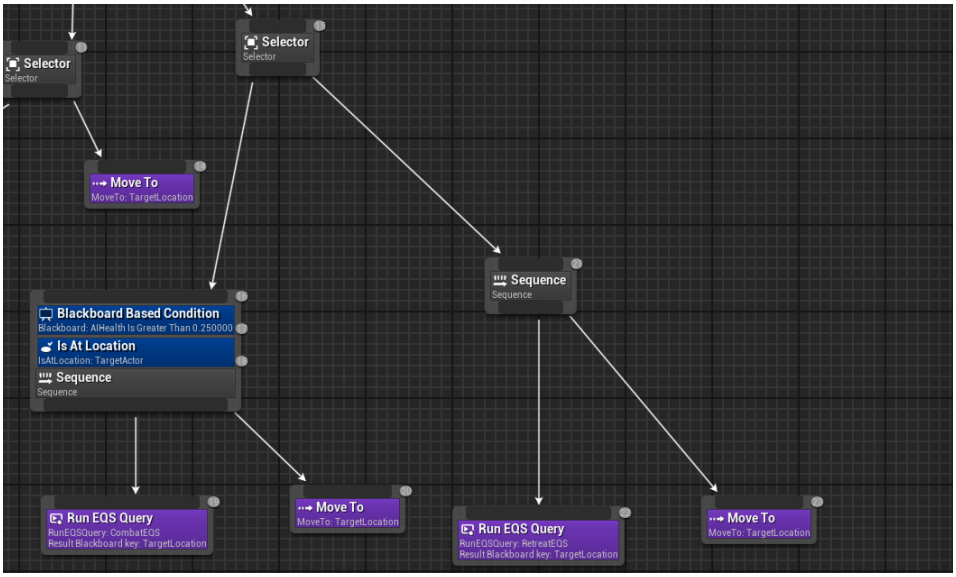
Nakon toga sekvenca prelazi na „Move To“ funkciju koja koristi tu „TargetLocation“ vrijednost. Ova grana dolazi druga od lijevo, i također sadrži opciju “Aborts both” što znači ako se podražaj “CanHearPlayer” promjeni na “Is Set”, stablo ponovno odlazi na prvu granu, te NPC provjerava izvor zvuka, ako se promjeni “TargetActor” na “Is Set”, stablo prijelazi na sljedeću granu.



Slika 40. Treća grana unutar stabla.

Zadnje dvije grane dolaze iz Sequence čvora “ChaseSequence” koji također sadrži “Engage_Service“ modifikator, o njemu malo kasnije. Jednom kada “ChaseSequence” uđe u selektor, ima dvije funkcije, prva se vrši jedino ako je vrijednost „CanSeePlayer is Is Set“,

dakle zahtjeva kondiciju „CanSeePlayer“ da ima vrijednost, zatim se izvršava „Move To“ funkcija sa „TargetActor“ ciljom. Uzimajući u obzir da je ovo selektor čvor, ova se akcija ponavlja dokle god je kondicija „CanSeePlayer is Is Set“ točna. Ako kondicija nije postignuta, selektor odlazi na drugu opciju, a to je „Move To“ funkcija s ciljem „TargetLocation“ koja je najčešće zadnja lokacija od “TargetActor” ili nova lokacija od percepcije za predviđanje.



Slika 41. Četvrta grana unutar stabla.

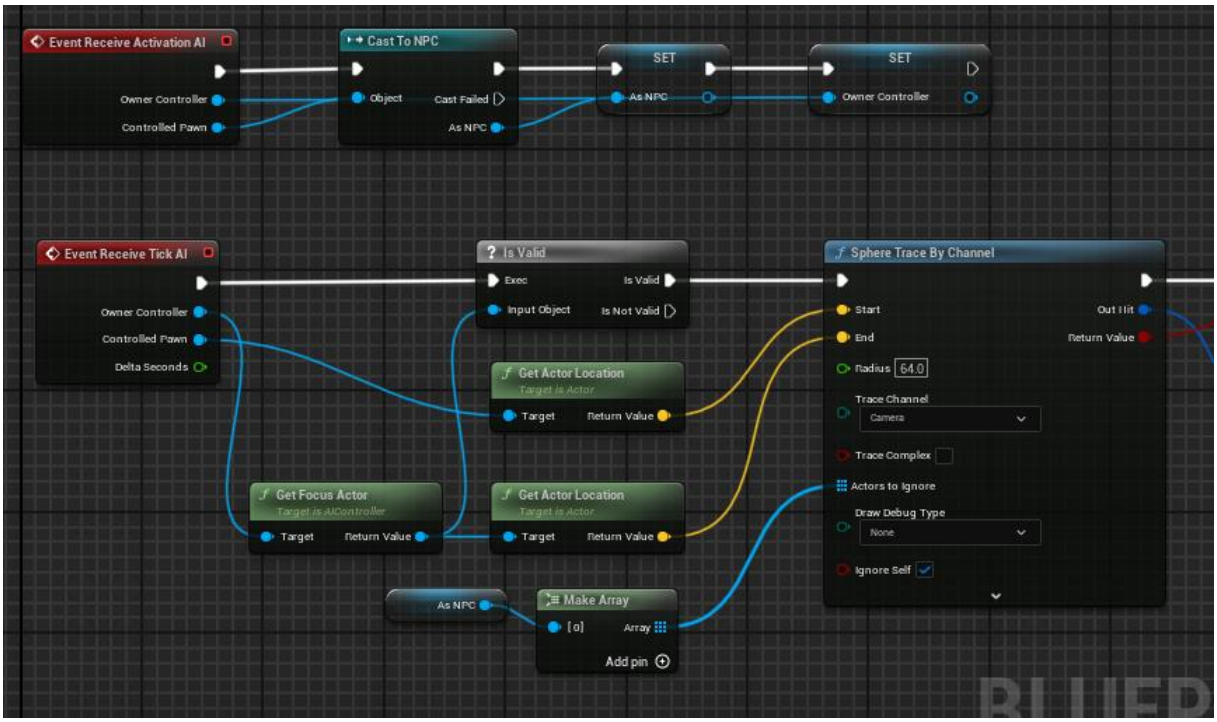
Četvrta grana ulazi u selektor čvor, te se odmah razdvaja u dvije različite grane. Prva grana odmah ima dva dekoratora. Prvi zahtjeva da je „AIHealth Is Grater Then 0.25“, dakle zahtjeva da je zdravlje od NPC-a veće od 25% i drugi provjerava „IsAtLocation: TargetActor“, provjerava lokaciju vrijednosti „TargetActor“. Nakon toga provodi sekvenca čvor u kojem izvršava „Run EQS Query“ sa „CombatEQS“ te dobiva novu „TargetLocation“ vrijednost. Nakon toga prelazi na „Move To“ funkciju u kojoj iskorištava tu „TargetLocation“ vrijednost.

Ako kondicija nije ispunjena, dakle ako „AIHealth Is Greater Then 0.25“ nije istinit, selektor čvor ide dalje, na zadnji čvor sekvence, koji prvo ulazi u funkciju „Run EQS Query“ sa „RetreatEQS“, dobiva novu „TargetLocation“ te nakon toga izvršava novu „Move To“ funkciju s novim „TargetLocation“ kao ciljem.

Više o “CombatEQS” i “RetreatEQS” nakon poglavlja o stablu ponašanja.

3.3.1. SERVIS ZA PALJBU

Služba[19](eng. Service) je sustav koji pridodaje čvorovima stabla ponašanja tako da se njihov zadatak odvija dokle god je taj čvor aktivan. Rad koristi “EngageService” u zadnjem čvoru stabla koji se bavi s okršajem s neprijateljem. Ovaj dodatak čvoru osigurava da NPC koristi događaje s točnim ciljevima.



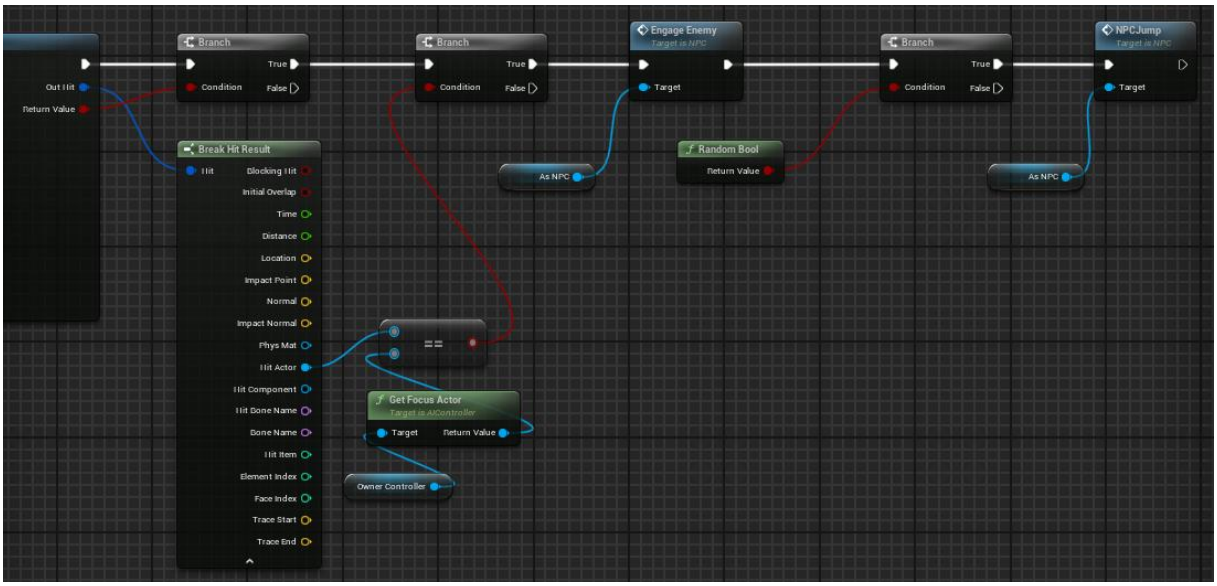
Slika 42. Početak koda unutar "EngageService"-a.

Prvi događaj je "EventReceiveActivationAI", dakle svaki put kada se NPC stvori u okolišu, ova služba mu šalje informacije, sa čvorom "Cast to NPC" možemo informacije koje ovaj sistem postavi, također iskoristiti u drugom sistemu, u ovom slučaju, "NPC" sustav.

Sljedeće "EventReceiveTickAI" se poziva na svaki "Tick"(hrv. Otkucaj) što se događa više puta u sekundi, znači da su informacije koje šalje veoma brze.

Dio događaja koji možemo ovdje vidjeti se većinski bavi s čvorom "Sphere Trace By Channel"(hrv. Sferična linija s kanalom) što nam je svojevrsni način testiranja linije vida, nakon što u njega postavimo nužne informacije poput "Start"(hrv. Početak) koji dolazi od "GetActorLocation"(hrv. DobaviLokacijuGlumca) a on sam se priključuje u "ControlledPawn"(hrv.KontroliraniPijun),te "End"(hrv. Kraj) koji također traži drugačiji "GetActorLocation" koji dobivamo iz "GetFocusActor"(hrv. DobaviGlumcaUFokusu) koji se spaja u "OwnerController"(hrv. VlasnikKontrolera). Možemo zaključi da nam ovo služi za početak i kraj linije koji čvor stvara. Liniju koju ovo ocrtava počinje od lokacije glumca koji kod izvršava, a završava na lokaciji glumca koji mu je u fokusu, tj meta.

Također "Actors to Ignore" (hrv. Glumci za Ignorirati) sadrže "Array"(hrv. Listu) na kojoj je samo NPC što osigurava da NPC ne cilja samog sebe.



Slika 43. Ostatak koda unutar "EngageService"-a.

Ostatak koda se sastoji većinom od "Branch"(hrv. Grana) čvorova koji se uz pomoć "Condition"(hrv. Kondicija) priključka koriste kao osigurači za provođenja akcija.

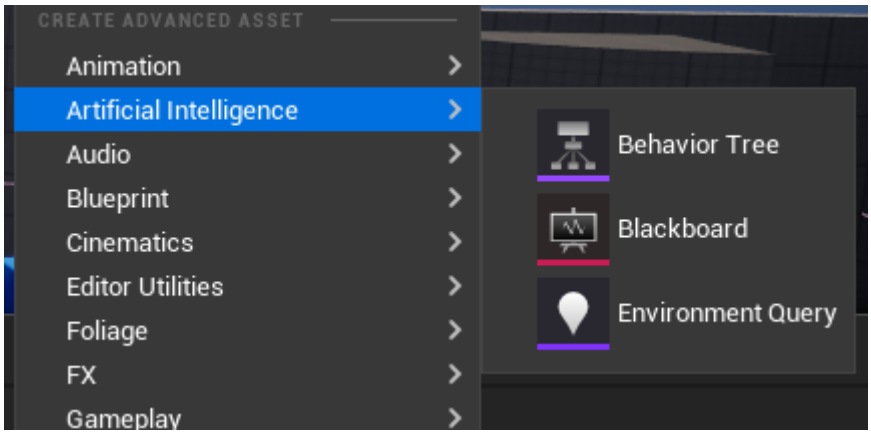
Prvi "Branch" čvor zahtjeva da prijašnji čvor ima metu, što možemo vidjeti s njegovim "Condition" priključkom u "Return Value" (hrv. Povratna Vrijednost) priključku.

Sljedeći čvor ima par koraka više, iz "Sphere Trace by Channel" možemo razdvojiti "Break Hit Results"(hrv. Slomi rezultate hitca) što nam daje sve informacije o rezultatu tog čvora, nama je potrebna informacija "HitActor"(hrv. PogođeniGlumac) nakon što tu informaciju spojimo u "==" čvor, što zahtijeva da su dvije informacije identične, dakle tražimo da je glumac pogođen također glumac koji je u fokusu, onda tek "Branch" čvor dopušta da izvedemo događaj koji se zove "EngageEnemy", o njemu više kada budemo pričali o kodu NPC-a.

Zadnji "Branch" ubacuje "RandomBool" (hrv. NasumičniBool) nasumičnu vrijednost "Istina" ili "Krivo" što uključuje događaj pod imenom "NPCJump" u nasumičnim intervalima što dopušta NPC-u skakanje dok je "EngageService" aktivan.

3.3.2. SUSTAV ZA OKOLIŠNO PREUSMJERAVANJE

Sustav za okolišno preusmjeravanje [20](eng. Environmental Query System) ili EQS skraćeno je jedan od alata za umjetnu inteligenciju unutar Unreal Engine-a 5 koji se koristi za prikupljanje podataka o okolišu. Uz pomoć generatora koji stvaraju točke interesa može provesti test koji govori koje su točke u okolišu najpovoljnije oviseći o pitanjima koja su postavljena. Zajedno sa stablom ponašanja i crnom pločom čini glavna tri alata za umjetnu inteligenciju unutar Unreal Engine-a.



Slika 44. Artificial Intelligence (hrv. Umjetna Inteligencija) padajući izbornik unutar Unreal Engine 5.

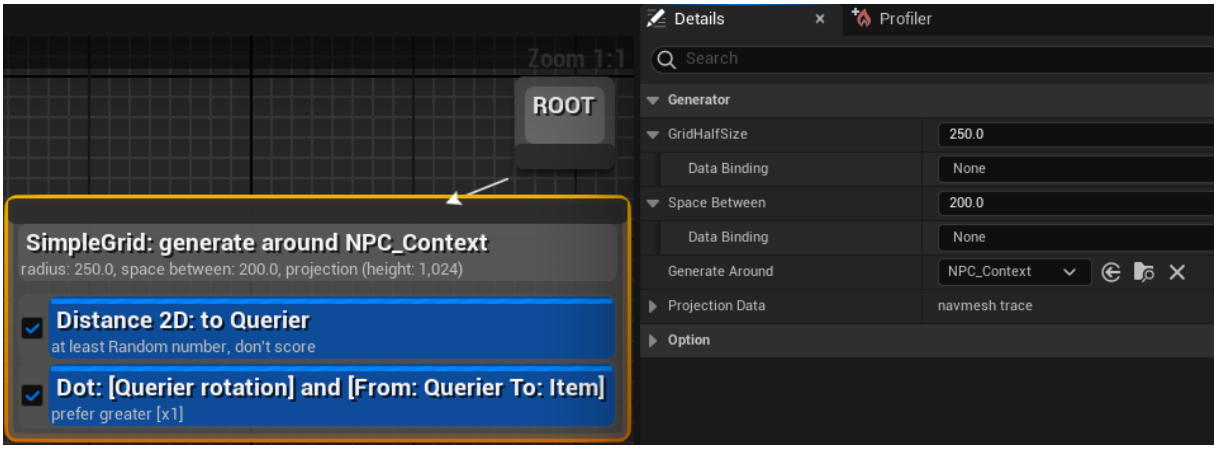
EQS se koristi za razne situacije, rad koristi tri različita EQS sistema, jedan za navigaciju po okolišu, drugi za borbu s neprijateljima, i treći za povlačenje jednom kada je zdravlje NPC-a otpalo ispod 25%.

3.3.2.1. NAVIGACIJSKO PREUSMJERAVANJE

Prvi sistem se zove “NavigationQuery”(hrv. NavigacijskiUpit) i njegov zadatak je priuštiti umjetnoj inteligenciji lokaciju za patrolu NPC-a, ovaj sistem je u uporabi kada NPC nema nikakvih podražaja, te samo patrolira po mapu dok ne naiđe na podražaj. Cilj sustava je dovesti sve karaktere u blizinu jedni drugoga, trenutno sustav ima 4 generatora koji stvaraju točke interesa, te se njihove vrijednosti mijenjaju oviseći o udaljenosti i rotaciji NPC-a od tih točaka.

Slika 45. Točke generatora i generatori po mapi.

Na slikama svaki generator stvara mrežu od 9 točaka, zajedno sa 4 generatora svaka točka ima vrijednost od 0 do 1, što nam pomaže pri izračunavanju najpovoljnije točke za kretanje.

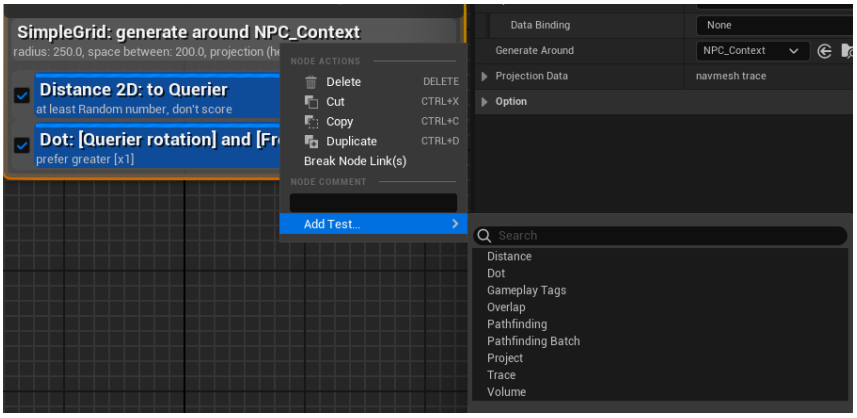


Slika 46. Postavke "NavigationQuery" EQS sistema.

Unutar samog EQS-a može se urediti njegove postavke, na desnoj strani sa "GridHalfSize" i "Space Between" se određuje koliko točaka se generira i koliko su udaljene jedna od druge, dok opcija "Generate Around" je postavljena na "NPC_Context" što je nova stavka koja je stvorena i postavljena na mjesta gdje želimo generirati točke.

Na lijevoj strani je određena logiku, prva stavka u sivoj boje "SimpleGrid: generate around NPC_Context" je sve što je postavljeno prije, na desnoj strani, s radijusom, prostorom između i visinom navedenom odmah ispod. Dvije plave stavke su testovi koje "NavigationQuery" provodi da odredi točke koje tražimo.

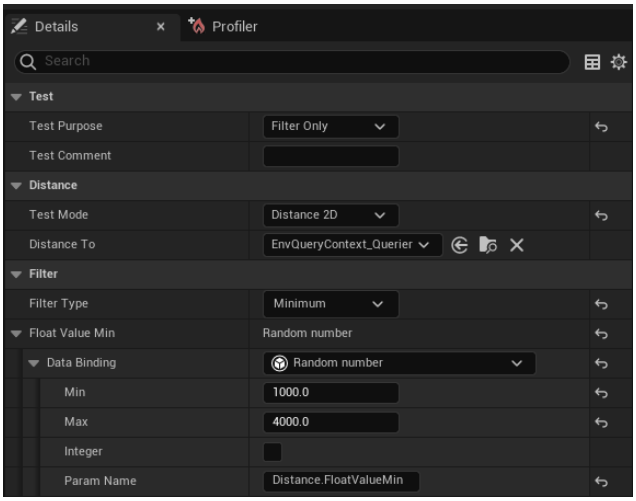
Testovi koji se provoditi su razni, ali nama treba samo "Distance" (hrv. Udaljenost) test i "Dot" (hrv. Točka) test.



Slika 47. Razne opcije testova u padajućem izborniku.

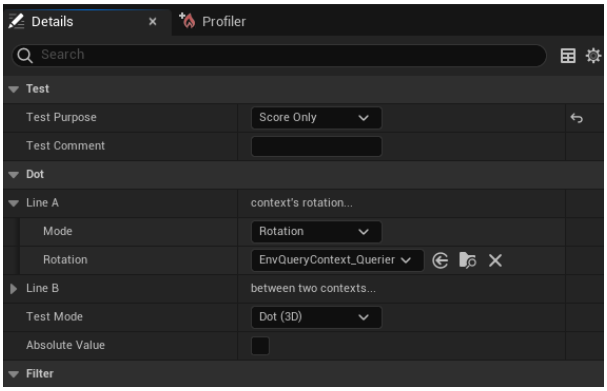
Prvi test za udaljenost je tu da definira koje točke želi da test odvoji, u opcijama je postavljeno da je "Test Mode" (hrv. Vrsta testa) 2D udaljenost, što znači da test ne gleda visinu kao parametar. Glavne stavke koje su postavljene pod "Filter"(hrv. Filter) opcijama i stavili Min i Max vrijednost od 1000 i 4000, što znači da test uzima samo točke koje su između te dvije vrijednosti, i druga stavka je u "Test Purpose"(hrv. Svrha testa) gdje postoje

dvije opcije “Score”(hrv. Ocijeni) i “Filter”(hrv. Filter). Ocijeni opcija daje rezultatu vrijednost od 0 do 1, dok Filter ocjenjuje samo sa 0 ili 1. Trenutno Filter opcija više odgovara jer želi izbaciti točke koje su preblizu ili predaleko.



Slika 48. Distance test opcije.

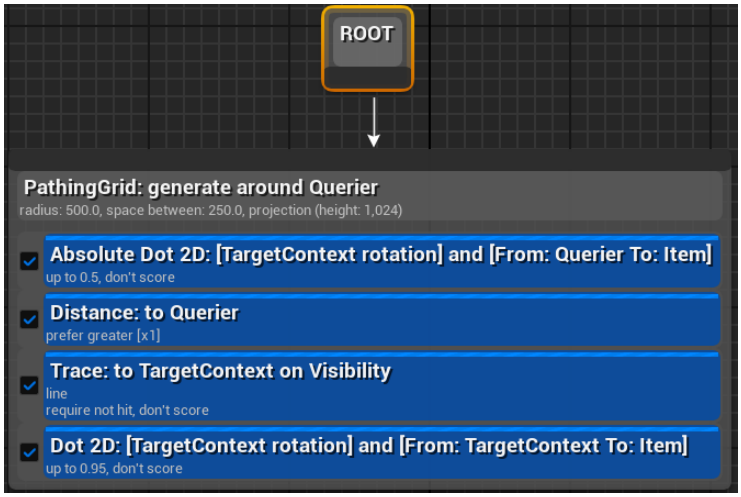
Drugi test, “Dot”jednostavno služi za rotaciju karaktera, te onemogućava NPC-u da se vrti između iste točke, to radi tako da postavimo opciju za “Score Only” te stavljamo “Mode”(hrv. Način rada) na “Rotation” (hrv. Rotacija) i postavljamo metu kao “EnvQueryContext_Querier” što označava našeg NPC-a. Uz pomoć ovoga točke koje su ispred imaju veću ocjenu nego točke koje bi zahtijevale rotaciju od NPC da krene prema njima.



Slika 49. Dot test opcije.

3.3.2.2. PREUSMJERAVANJE ZA BORBU

Slično sustavu za navigaciju zvanom "NavigationQuery" ima i sustav za borbu pod imenom "CombatQuery". Funkcionira na istim principima kao i navigacijsku sustav, samo traži druge točke za uporabu.



Slika 50. "CombatQuery" postavke.

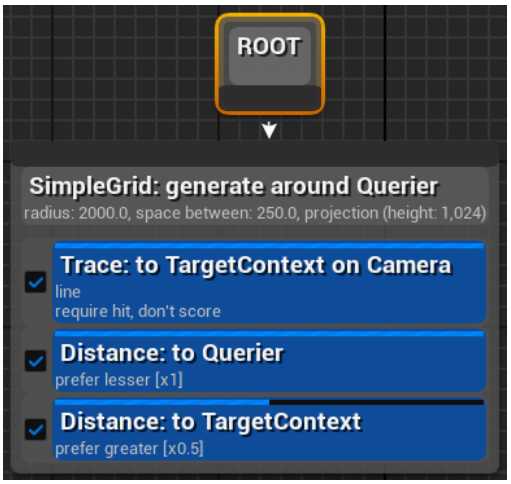
U postavkama može se vidjeti da sistem za borbu ima 4 testa kroz koja prolazi.

- Prvi test provjerava rotaciju NPC-a u usporedbi sa suigračem te bira točke lijevo ili desno od svojeg središta.
- Drugi test provjerava udaljenost točke igračem te bira što veći broj.
- Treći test provjerava ima li NPC liniju vida sa suigračem s nove točke.
- Četvrti test provjerava da točke koje biraju nisu ispred ili iza njega, već samo postrance.

Uz pomoć ovog sistema jednom kada NPC uđe u borbu on konstantno mijenja poziciju oviseći o ciljanju i pozicije suigrača.

3.3.2.3. PREUSMJERAVANJE ZA POVLAČANJE

Poput prijašnja dva sistema, u postavkama možemo vidjeti da sistem za povlačenje ima 3 testa koja provodi.



Slika 51. "CombatQuery" postavke.

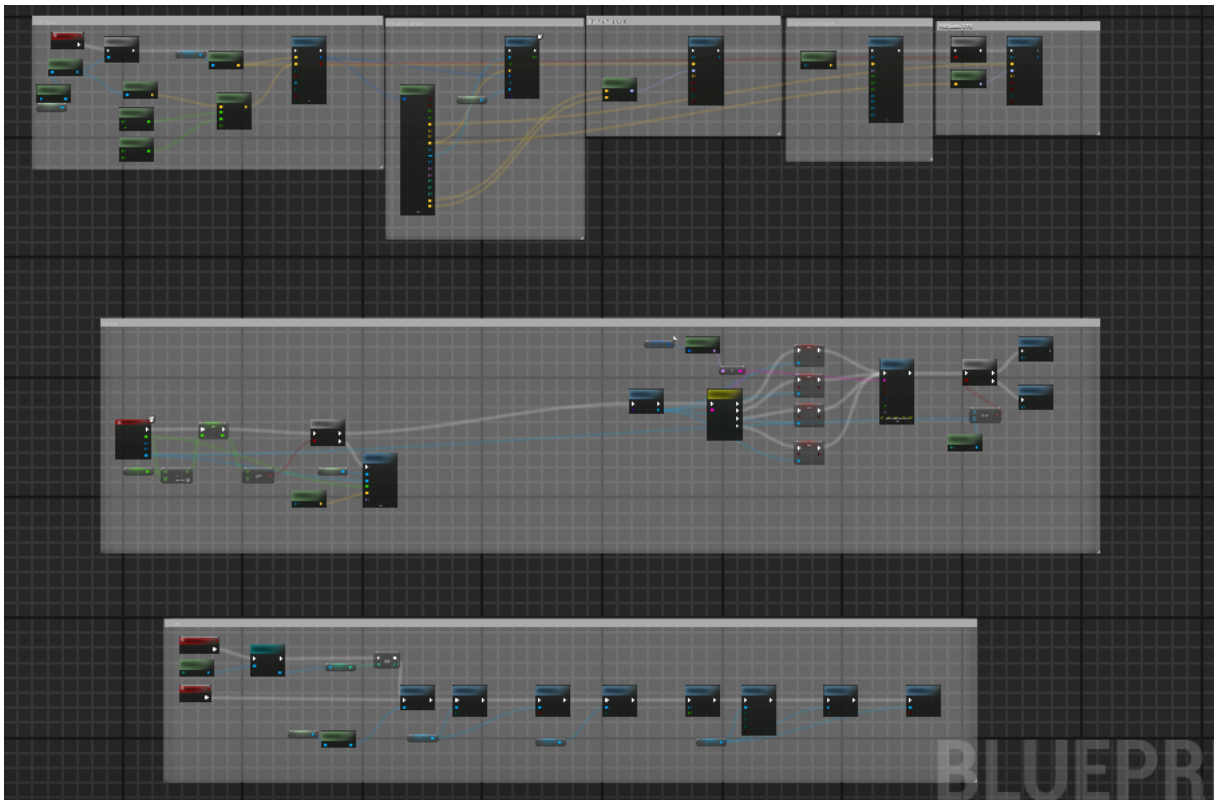
Prvo je važno primjetiti da ovaj sistem uzima najveći radijus od 2000, što znači da kalkulira veću površinu, uz pomoć svoja tri testa on prolazi:

- Prvi test provjerava "Line Trace"(hrv. trag linije), tj pokušava slomiti našu liniju vidu i pronaći zaklon.
- Drugi test provjerava koje su točke najbliže njemu te im daje veću ocjenu.
- Treći test provjerava koje su točke najudaljenije od suigrača te im daje veću ocjenu.

Uz pomoć ovog sistema, NPC se skriva i bježi od napadača. Zajedno sa stablom ponašanja, ovi sistemi stvaraju iluziju inteligencije unutar sustava.

3.4. NPC – KARAKTER S UMJETNOM INTELIGENCIJOM

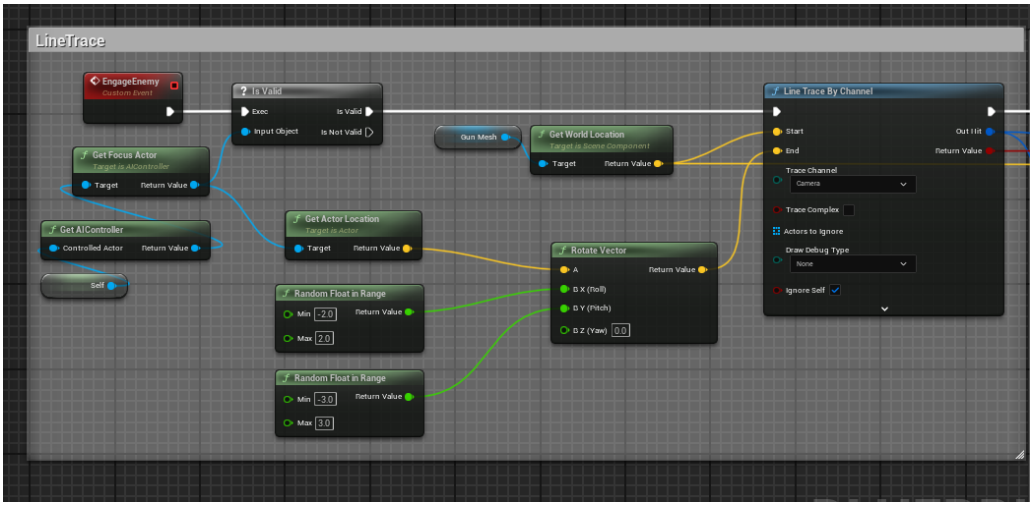
NPC je glavni dio projekta, unutar njegovog koda imamo više sustava koji obavljaju različite zadatke. Svaki od sustava je jednostavan za objasniti. Većina potrebnih informacija dolazi do glumca preko drugih komponenti, poput stabla ponašanja i sustava za percepciju. Ovom glumcu je također dodan model koji je vidljiv u okolišu, uz njega spojen je i sustav animacija. Sustav animaciju dopušta odigravanje animacija za navigaciju po okolišu. Također koristimo model puške koju koristi sam igrač. Modeli i animacije su dio standardnog paketa, ali nisu upotrijebljeni na drugim glumcima radi prirode njihove funkcije, koja je isključivo za primanje i slanje informacija.



Slika 52. Cijeli blueprint kod za NPC-a.

3.4.1. SUSTAV ZA PALJBU

Jedan od važnijih sustava je kod koji se bavi paljbom, čvor “Engage Enemy” (hrv. Napadni neprijatelja) sadrži tok događaja koji se odvijaju svaki put kada NPC napadne neprijatelja.



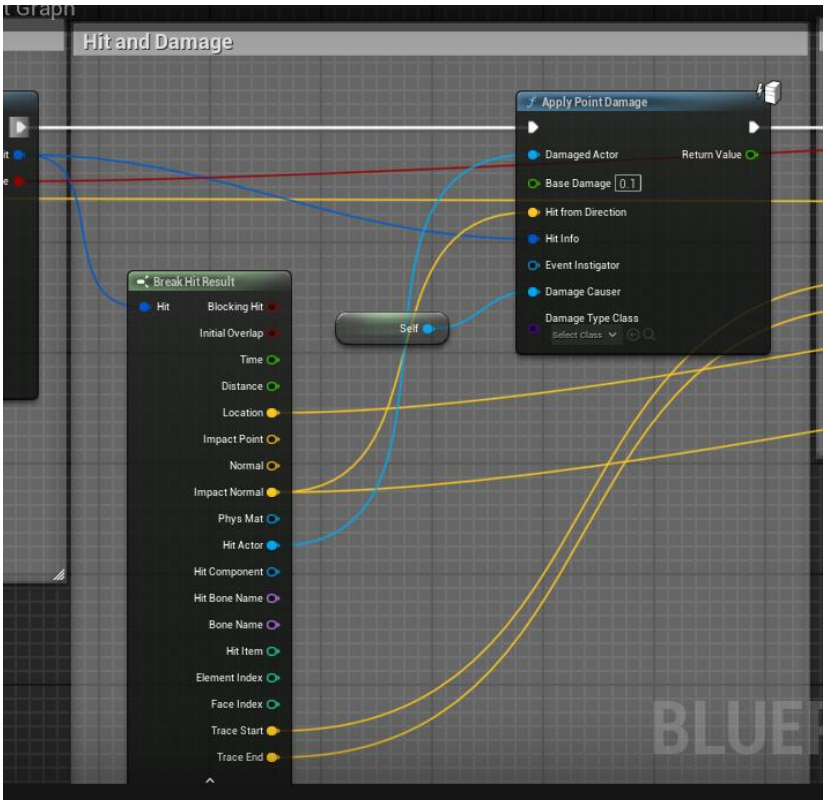
Slika 39. Prvi dio “Engage Enemy” slijeda događaja.

Prvi čvor na koji događaj nailazi je “Is Valid” (hrv. jest valjan), ovaj čvor provjerava da je objekt koji čvor poziva valjan i sadrži potrebne informacije, u ovom slučaju čvor provjerava “Get Focus Actor”, koji ima cilj na “Get AI Controller” koji kontrolira “Self” čvor, gledajući

kako je kod unutar NPC koda, “Self” poziva NPC glumca. Ako su sve informacija validne, kod nastavlja dalje, ako nije, ništa se ne događa.

Sljedeći čvor do kojeg se dolazi je “Line Trace by Channel”, funkcija čvor je poprilično slična poput istog čvora koji se koristi unutar “EngageService”-a, dok u onoj situaciji se čvor koristi za provjeru mete, te dolazi na kraju do “Engage Enemy” funkcije, u ovom slučaju čvor se koristi za oduzimanje bodova zdravlja. “Start” priključak se spaja sa “Get World Location” (hrv. Dohvati lokaciju u svijetu) kojoj je meta “Gun Mesh”(hrv. Mreža puške) dakle početak linije je model oružja koji NPC koristi. “End” priključak ima više čvorova, ova kombinacija se koristi da doda paljbi NPC-a šansu za promašaj, “Rotate Vector” (hrv. Rotiraj Vektor) pod vrijednost “A” uzima vektorsku lokaciju glumca te vrijednost “B” rastavljamo na više vrijednosti, u originalnom čvoru vrijednost “B” ima samo jedan priključak, vrijednost je rastavljena jer je potrebno utjecati samo na dvije vrijednosti što je dovoljno za dati nasumičnu paljbu.

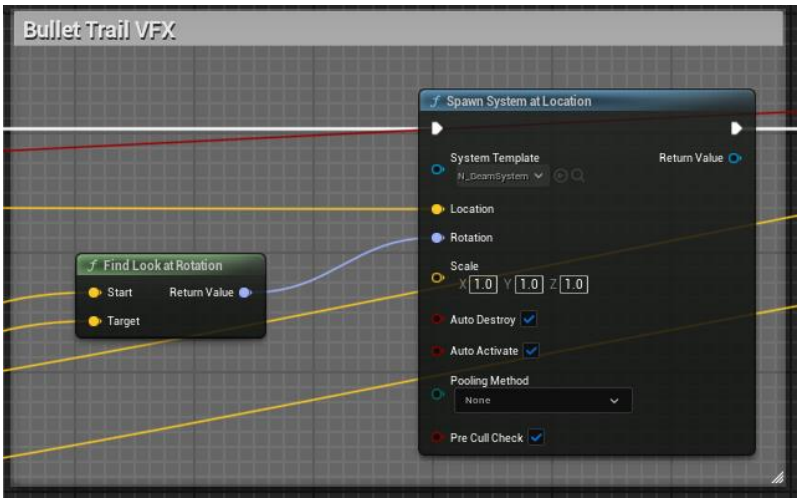
Za vrijednosti “B X (Roll)” i “B Y (Pitch)” je spojen “Random Float in Range” (hrv. Nasumični Plutajući unutar raspona) te dodane minimum i maksimum vrijednosti od -2, 2 i -3, 3 što utječe na završnu točku linije za te vrijednosti. Zbog postavljanja vrijednosti na min i max, osigurava da je svaka linija različita od prijašnje.



Slika 53: Drugi dio “Engage Enemy” slijeda događaja.

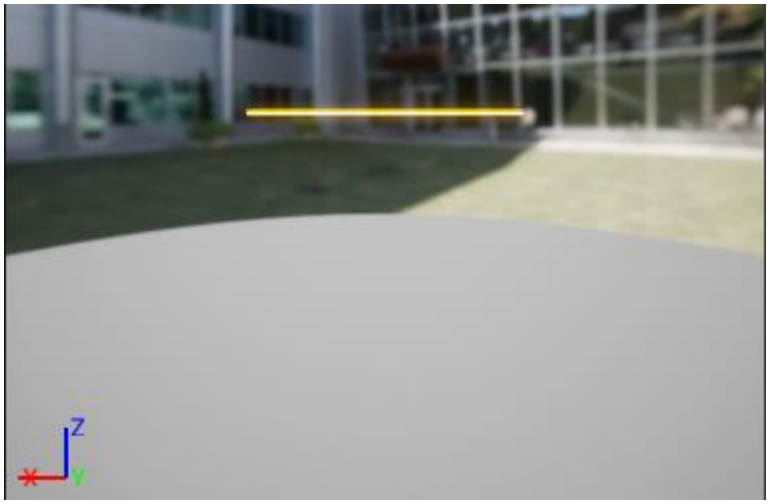
Sljedeći čvor aplicira štetu na neprijatelja, ”Apply Point Damage” (hrv. Apliciraj točku štete) te vrijednost “Base Damage” (hrv. Baza Štete) oduzima pogođenom 0.1 zdravlja. Za više informacija razdvajamo “Break Hit Results” iz “Line Trace by Channel” čvora, radi razdvojenih informacija, te vrijednosti za “Hit Actor” (hrv. Pogođeni Glumac) i “Impact Normal” (hrv. Normal Udarca) uzet iz rezultata te priključujemo u “Apply Point Damage” čvor. Rezultat prethodnog čvora također spajamo u “Hit Info” (hrv. Informacije o pogotku).

Priključak “Damage Causer” (hrv. Počinitelj Štete) je spojen zajedno sa “Self” čvorom, dakle samim NPC glumcom.



Slika 54: Treći dio “Engage Enemy” slijeda događaja.

Glavni dio koda je zasad sređen, zadnja dva dijela se bave sistemima za vizualno i auditivno poboljšavanje iskustva. Sljedeći čvor se zove “Spawn System at Location” (hrv. Stvori sistem na lokaciji) te se koristi za stvaranje vizualnih linija koje predstavljaju paljbu iz vatrenog oružja, ovaj dio je važan radi prenošenja informacija igraču koje je oružje trenutno u funkciji. Sistem koji ovaj čvor koristi je Unreal Engine sistem za vizualne efekte, Niagara sistem, projekt koristi modificiranu žutu liniju, original je plave boje, širi na početku i uži na kraju, dakle moguće je napraviti svoj vlastiti sustav po volji, radi potrebe projekta, jednostavna žuta linija prenosi sve potrebne informacije.

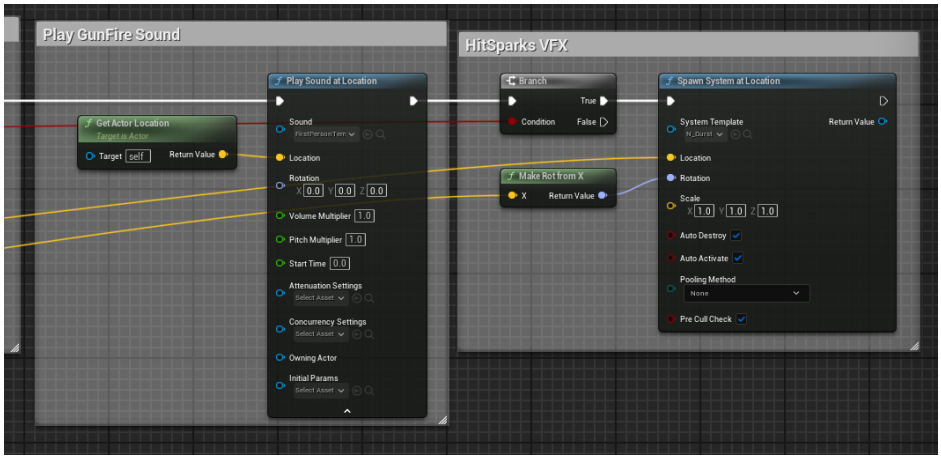


Slika 55: Žuta linija koja označuje paljbu oružja.

Pod priključak za lokaciju povezan je čvor “Get World Location” sa “Gun Mesh” priključkom, isti koji je korišten u čvoru za trag linije.

Priključak za rotaciju koristi čvor “Find Look at Rotation” (hrv. Pronađi pogled na rotaciju) te u njega je stavljen “Trace Start” (hrv. Trag start) i “Trace End” (hrv. Trag kraj) iz “Break Hit

Results" koji dolazi iz čvora za trag linije. S ovom kombinacijom sustava vizualna linija prati trag linije koji aplicira štetu.



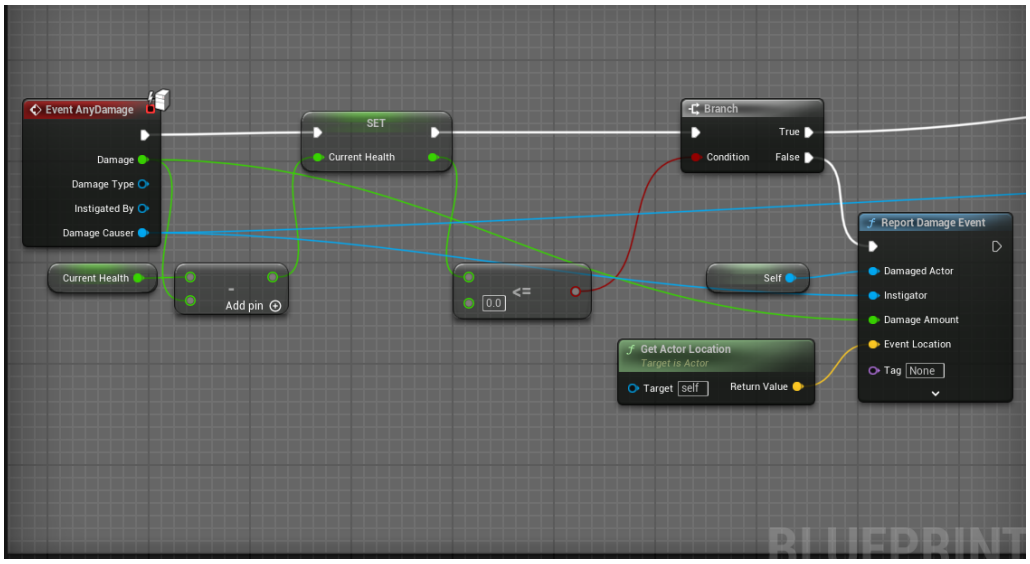
Slika 56: Četvrti dio "Engage Enemy" slijeda događaja.

Sljedeći čvor, "Play Sound at Location" (hrv. Igraj zvuk na lokaciji) nam pridodaje iskustvu tako da igra zvuk paljbe svaki put kada je paljba aktivna, čvor "Get Actor Location" spojen s priključkom za lokaciju osigurava izvor zvuka iz lokacije NPC-a.

Zadnji čvor "Spawn System at Location" ima kondiciju koja dolazi iz čvora za trag linije, dakle potrebno je da linija ima uspješan pogodak, dok to nije potrebno za prošla dva sustava. Čvor također koristi Niagara sustav te kao uzorak ima par iskrica koje skaču u vis, ovaj vizual označava hitac o površinu, objekt ili neprijatelja. Priključak za lokaciju spajamo s lokacijom iz "Break Hit Results", a rotaciju provlačimo kroz "Make Rot from X" (hrv. napravi rot od X) što nasumično mijenja X vrijednost vizuala, rotira ga tako da svaki hitac ima drugačiji smjer iskrica.

3.4.2. SUSTAV ZA ZDRAVLJE

Sljedeći sustav unutar NPC-a je sustav koji prati zdravlje i smrt[21]. Sustav koristi jednostavan sustav za praćenje razine zdravlja te zabilježava krivca smrti koji je NPC ili igrač.

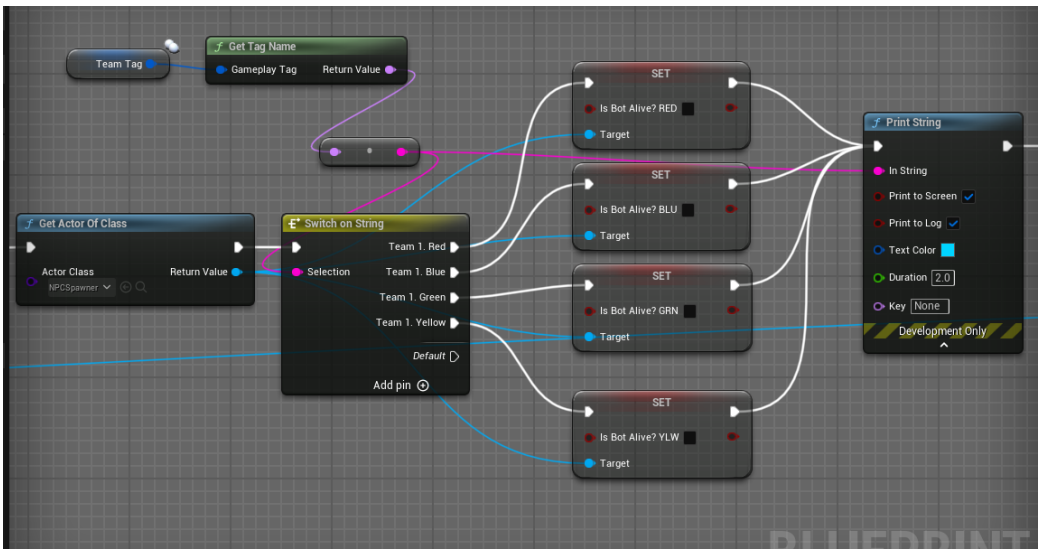


Slika 57: Prvi dio "Event AnyDamage" slijeda događaja.

Slijed događanja počinje sa "Event AnyDamage" (hrv. Događaj bilo koje štete) čvorom koji se koristi za registraciju štete. Čvor odmah prelazi u "Set" (hrv. Postavi) funkciju s ciljem na "Current Health" (hrv. Trenutačno zdravlje) dakle mijenja njegovu vrijednost. Njegova vrijednost je promijenjena za iznos priključka "Damage" (hrv. Šteta) koja se oduzima s funkcijom za oduzimanje "-" od prijašnje vrijednost "Current Health". Sustav je veoma jednostavan te nam konstantno zapisuje novu vrijednost pri svakom zvanju sustava.

Nakon zapisivanja nove vrijednost "Current Health" ,provodi se provjera sa "<=" funkcijom što zahtijeva da je vrijednost "Current Health" jednaka ili manja od nule.

Ako je vrijednost veća, i kondicija ne prolazi, znači da NPC još uvijek ima pozitivni broj u "Current Healthu" te se događaj štete zapisuje sa "Report Damage Event" (hrv. Zapiši događaj štete) sa svim potrebnim informacijama spojenim u njega. Ovaj čvor je veoma koristan za praćenje toka događaja i provjeravanje sustava.



Slika 58: Drugi dio "Event AnyDamage" slijeda događaja.

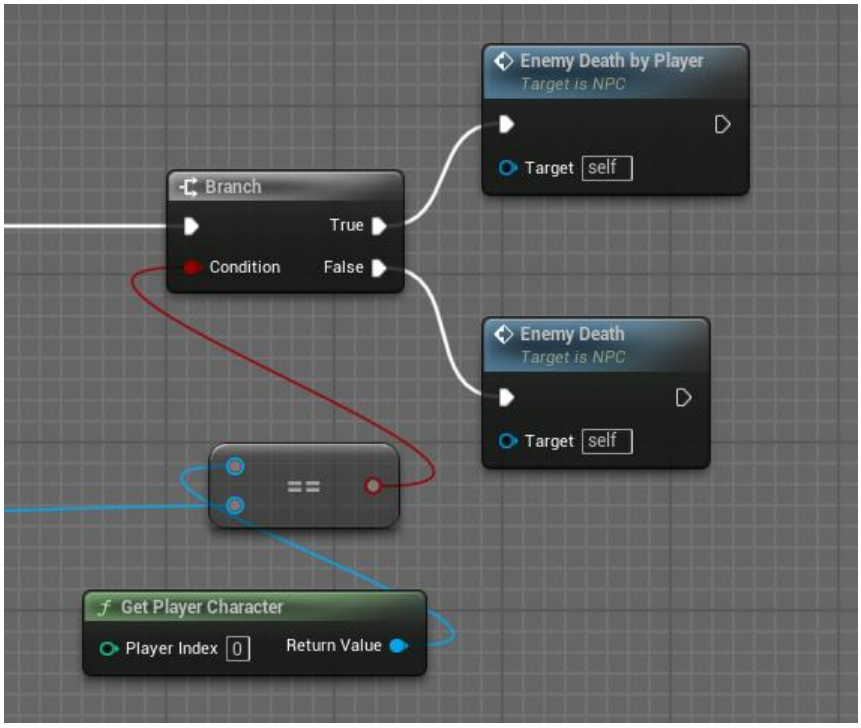
U suprotnom, ako “Current Health” padne na nula ili ispod nule, sustav vrši provjeru koji je točno NPC ubijen i od koga.

Prvi čvor na koji nailazi je “Get Actor of Class” (hrv. Dohvati glumca od klase) uz pomoć ove linije možemo u jedan sustav, ovom slučaju “NPC” sustav, dohvatiti vrijednosti iz drugog sustava, ovom slučaju “NPCSpawner” jer je to meta čvora. Također može mijenjati informacije koje su u tom sustavu.

Nakon što su informacije iz “NPCSpawner”-a dohvaćene, ulazimo u “Switch on String” (hrv. Promijeni po String-u) čvor koji se koristi kao filter te propušta informacije zaviseći o “Selection” (hrv. Selekcija) informaciji. Ovaj priključak dohvaća informacije o “Team Tag”-u kojeg dobiva preko “Get Tag Name” (hrv. Dohvati Ime Tima) te ta informacija prolazi kroz pretvarač koji mijenja informaciju u tekst, tj. String. Nakon toga ta informacija ulazi u čvor i on odlučuje koji okidač za koji tim propušta daljnji tok koda.

Recimo na primjer da je NPC-u crvenom timu pao “Current Health” ispod nule, čvor “Switch on String” dobiva informaciju o njegovom timu, te provodi slijed za “Team 1. Red”, ulazi u “Set” funkciju za informaciju koja je u “NPCSpawner” sustavu, te postavlja boolean vrijednost informacije “IsBotAlive? RED” (hrv. Je li Bot živ? CRVENI) na “Krivo”. Ova informacija se koristi u “NPCSpawner” sustavu za regulaciju statusa i broja NPC-a koji su trenutno živi.

Nakon toga ulazi u “Print String” čvor koji printa ime tima koji je ubijen.



Slika 59: Zadnji dio “Event AnyDamage” slijeda događaja.

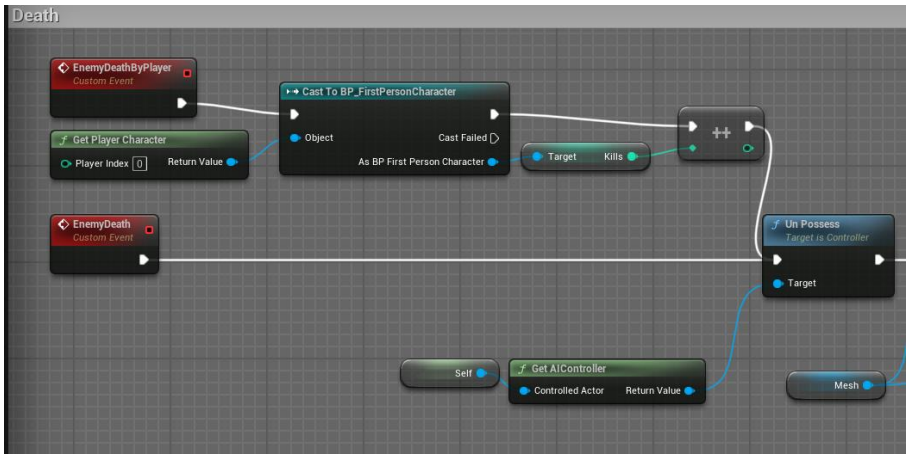
Nakon što je dokučeno koji je NPC ubijen, zadnji dio slijeda provjerava tko ga je ubio, trenutačno sustav razlikuje samo smrt od strane drugih NPC-a ili od strane igrača.

Pod “Branch” čvorom kao kondiciju imamo “==” funkciju, koja zahtijeva da je “Get Player Character” (hrv. Dohvati karakter igrača) jednak s priključkom kojeg uzimamo iz “Event AnyDamage”, “Damage Causer” (hrv. Počinitelj štete).

Ovo je veoma jednostavan način provjere je li počinitelj koji je snizio “CurrentHealth” ispod nule, također i igrač. Ako je kondicija istinita, slijed prelazi u “Enemy Death by Player” (hrv. Neprijateljska smrt od igrača), ako nije, onda prelazi u “Enemy Death”. Razlika između njih je objašnjena u sljedećem dijelu sustava.

3.4.3. SUSTAV U SLUČAJU SMRTI

Zadnji slijed događaja počinje u prijašnjem sustavu, te nema puno razlika između dva događaja “Enemy Death” i “Enemy Death by Player”.

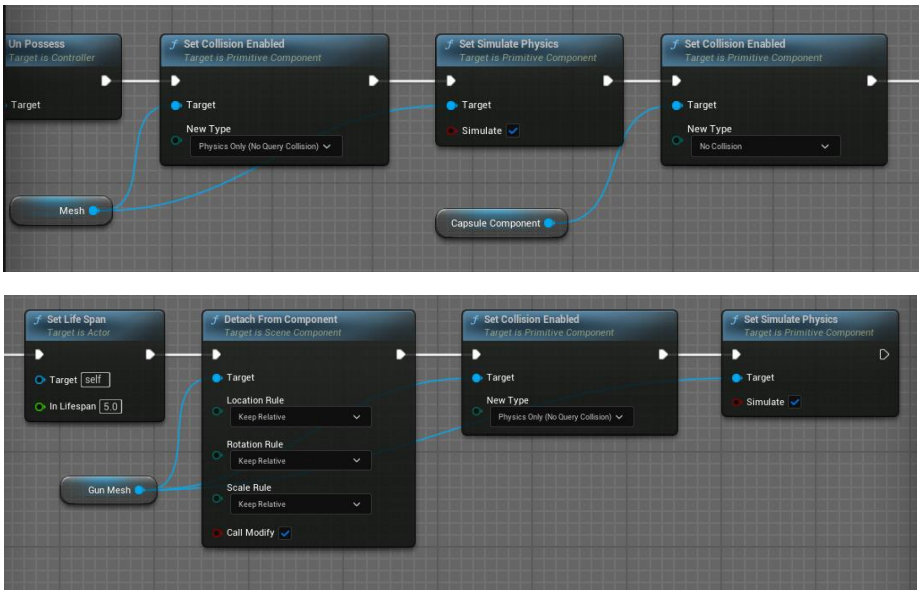


Slika 60: Prvi dio slijeda događaja za smrt NPC-a.

Iz navedenog je očito da je jedina razlika između dva događaja to što “Enemy Death by Player” ima par više koraka od “Enemy Death” te nakon toga zajedno vrše isti slijed.

Ako je smrt došla od strane igrača, slijed ulazi u “Cast to BP_FirstPersonCharacter” čvor (hrv. Pošalji u BP_FirstPersonCharacter) što nam omogućuje slanje informacija u drugi sistem, ali ne i dohvaćanje tih informacija, u ovom slučaju koristimo “++” čvor koji nam dodaje +1 vrijednost integer informaciji “Kills” koja je unutar “BP_FirstPersonCharacter” sistema. Ova informacija se koristi za praćenje broja uzrokovanih smrti od strane igrača.

Nakon toga oba događaja imaju isti slijed. Prvo koriste čvor “Un possess” (hrv. Oduzmi posjed) s metom na “GetAIController” što uklanja umjetnu inteligenciju iz modela NPC-a.



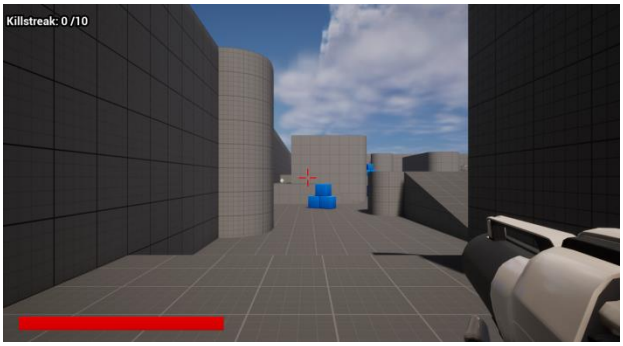
Slika 61: Ostatak slijeda događaja za smrt NPC-a.

Ostatak slijeda omogućava koliziju modela s okolinom i simulira fiziku na njegu, dopušta mu interakciju s drugim dinamičkim objektima.

Također odvaja model puške od modela tijela i dopušta mu istu razinu interakcije. Postavlja dužinu života ovih modela na 5 sekundi, nakon čega nestaju iz okoline.

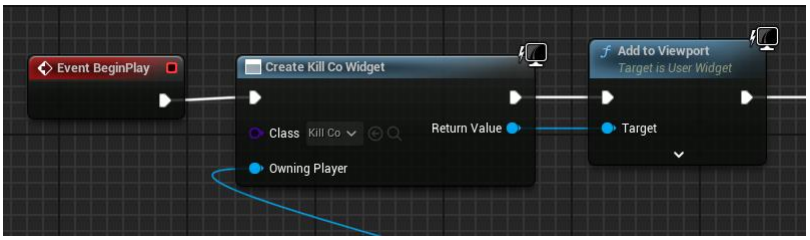
3.5. WIDGET

Widget [22]komponenta se koristi za prikaz informacija na igračevom sučelju, projekt sadrži widget komponente za zdravlje, broj bodova (“Killstreak”) i ciljnik na sredini ekrana.



Slika 62: Igračevo sučelje.

Za implementaciju widgeta u sučelje unutar komponente za igrača dodajemo čvor “Create Widget” (hrv. Stvori Widget) i za njim “Add to Viewport” (hrv. Dodaj na sučelje) koji spajamo unutar slijeda događaja za početak igre.

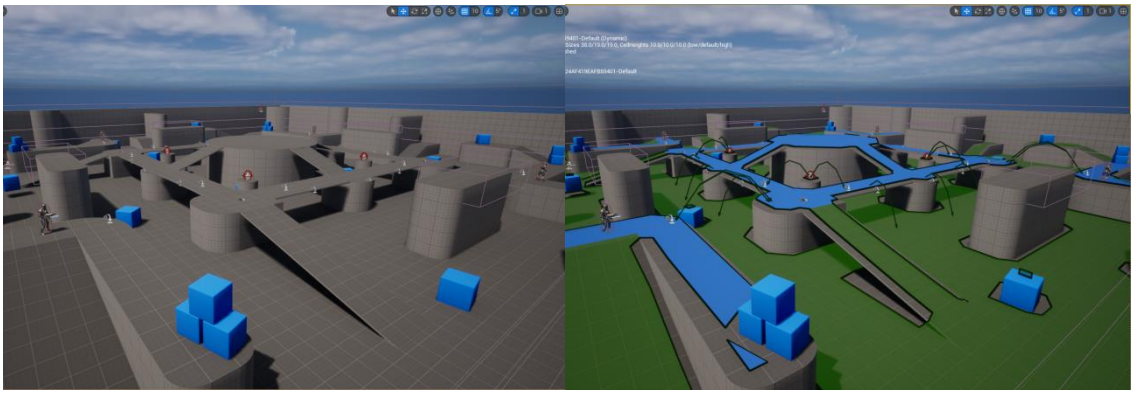


Slika 63: Widget za bodove s imenom "Kill Co".

Ostatak slijeda je standardan s projektom s kojim je došao.

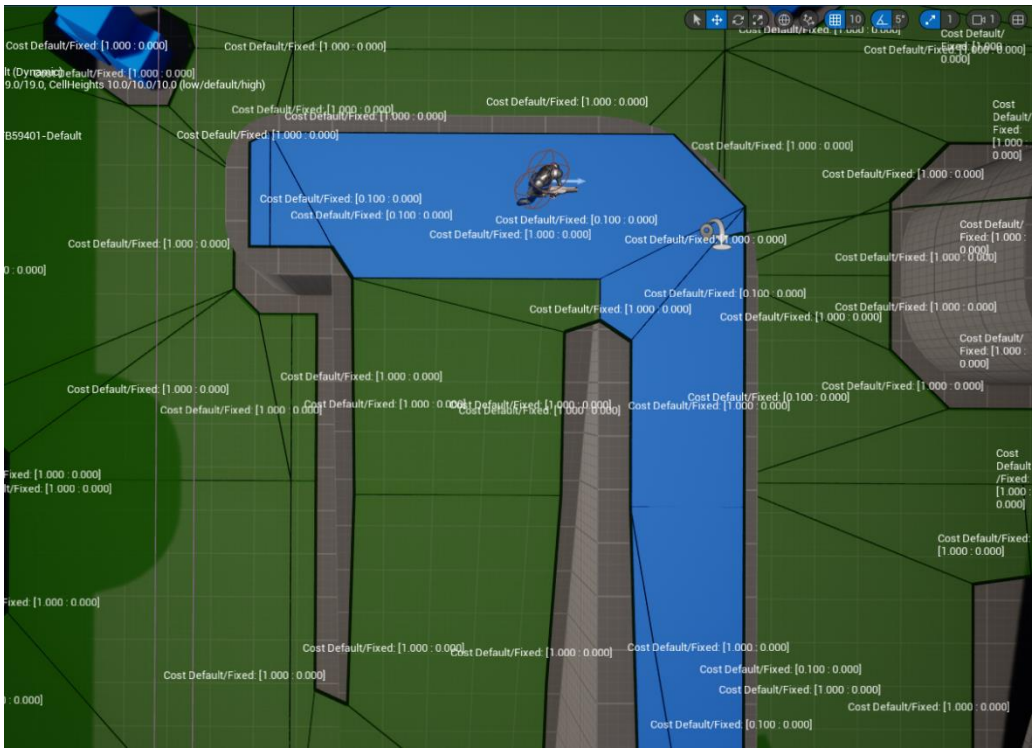
3.6. NAVIGACIJSKA MREŽA

Navigacijska mreža[23](eng. Navigation mesh) je komponenta koji stavljamo preko područja koje želimo učiniti prohodnim za karaktere s umjetnom inteligencijom. Unutar Unreal Engine-a 5 možemo vidjeti navigacijsku mrežu unutar razine s prečicom na tipki "P".



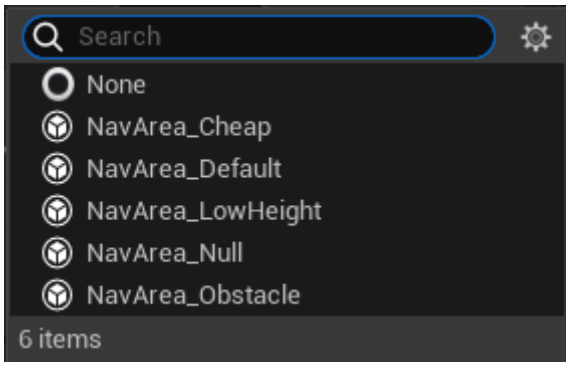
Slika 64. Vizual Navigacijske mreže.

Jednom kada je mreža razvučena preko područja, sama mreža razdvaja sva prohodna područja te ih dijeli na još manje dijelove, jednom kada je mreža dovoljno rastavljena, svaki od novih sekcija svoju cijenu, uz pomoć koje navigacijska mreža izračunava najpovoljnije rute kretanja između dvije točke.



Slika 65. Različite cijene navigacijskih sekcija.

Iz navedenog primjera možemo vidjeti kako zelene i plave sekcije imaju različitu cijenu. Dok zeleni sekcije imaju cijenu od 1.0, plave sekcije imaju cijenu od 0.1, što ih čini jeftinijim za kretanje. Nad prvotnu navigacijsku mrežu postavljena je mreža s modifikatorom, što čini cijene tih područja nižim. Ovo je jedna od više opcija koja se koristi za usmjeravanje umjetne inteligencije na područje koje želimo da više koristi. Postoji više modifikatora koji se koriste za različite svrhe.



Slika 66. Modifikatori navigacijske mreže.

- NavArea_Default (hrv. Standardno) područje je izvornih vrijednosti.
- NavArea_LowHeight (hrv. Niska Visina) područje dopušta prohod samo karakterima ispod određene visine ili ako hodaju u čučnju.
- NavArea_Null (hrv. Ništa) područje nema cijenu i nije prohodno.
- NavArea_Obstacle (hrv. Prepreka) područje je teže za korištenje i ima veću cijenu.

Uz ove modifikatore moguće je napraviti i svoje npr:

- NavArea_Cheap (hrv. Jeftino) čini područje jeftinijim za kretanje.

Ovaj modifikator je napravljen tako da svako područje nad kojom je ima postavljenu cijenu od 0.1 umjesto od 1, uz pomoć ovog malog dodatka osiguravamo da se naši NPC-evi se uvijek kreću po željenom dijelu mape.

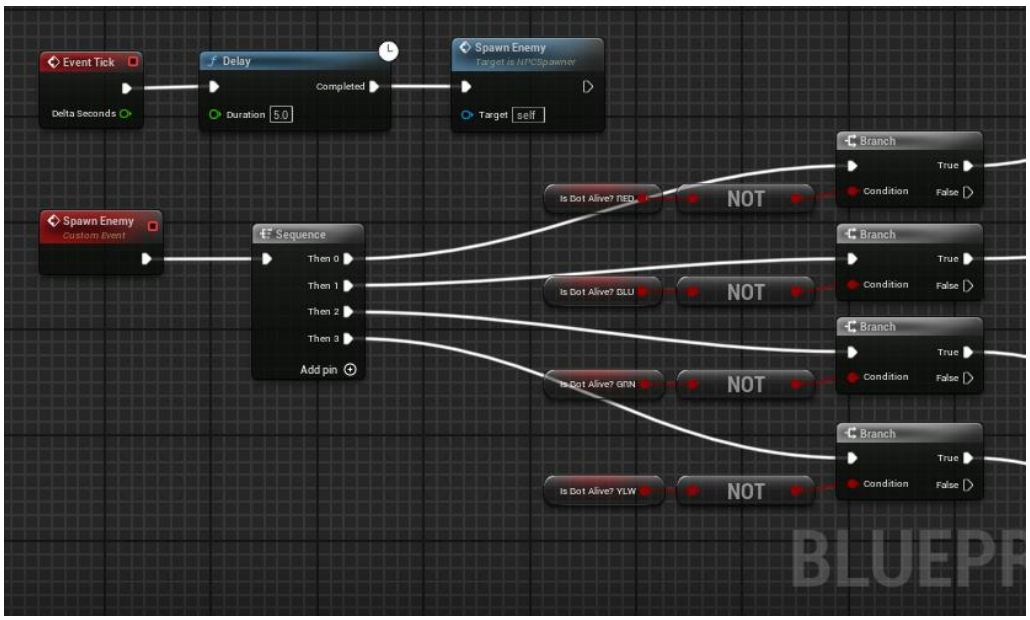
3.7. NPCSPAWNER – SUSTAV ZA PONOVNO STVARANJE

NPCSpawner(hrv. NPC Stvarač) je dodatak originalnom šablonu koji je napravljen za ovaj projekt. Ključna stavka ovog projekta je omogućiti uzastopnu borbu NPC-a, ovaj dodatak osigurava da jednom kada je NPC poražen, on se nakon brzog vremena ponovo stvori nazad u okoliš i nastavlja svoju zadatak ispočetka.

Trenutno su u projektu prisutna četiri tima svaki sa svojom bojom:

- Red (hrv. Crvena)
- Blue (hrv. Plava)
- Green (hrv. Zelena)
- Yellow (hrv. Žuta)

Svakom timu dodan je jedan NPC, s ovom raspodjelom timova, moguće je lagano odrediti koji NPC pripada kojem timu i njegov status života ili smrti.



Slika 67: Prvi dio koda "NPCSpawner"-a.

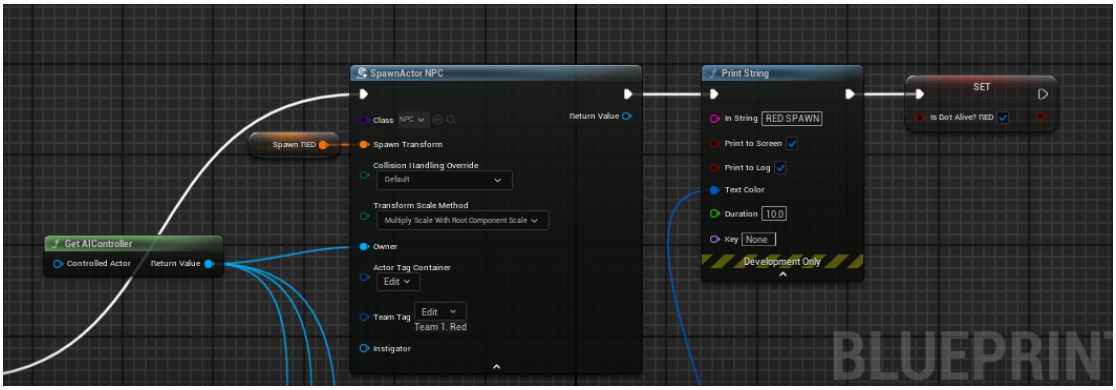
Kod počinje sa "EventTick" čvorom, te ulazi u "Delay" (hrv. Odgodi) čvor, koji ima vrijednost "Duration" (hrv. Trajanje) postavljeno na 5. Što znači da prolazi 5 tickova unutar enginea prije nego što se kod nastavlja, te tek onda pokreće "SpawnEnemy" (hrv. Stvori Neprijatelja) čvor.

Odmah ispod je vidljivo što sve ovaj kod sadrži. Prvi čvor je "Sequence" (hrv. Sekvenca), isto kao i unutar stabla ponašanja ovaj čvor vrti sve mogućnosti koje može izvesti. U ovom

primjeru svaki od opcija je moguće izvršiti do kraja, ili zapne na sljedećem čvoru. Četiri opcije su sve iste, samo s različitim booleanom kao kondicija za prolazak dalje, svaki od boolean opcija je provjera za status NPC-a tog tima.

Radi preglednosti rad prolazi samo kroz kod jednog tima.

Nakon "Sequence" čvora slijed dolazi do "Branch" koji za kondiciju ima boolean vrijednost "IsBotAlive?RED"(hrv. JeLiBotŽiv?CRVEN) koja prolazi kroz "NOT" (hrv. NIJE) čvor, što znači da nam kondicija zahtjeva da je ova vrijedna "False" (hrv. Krivo) da bi mogao proći dalje s provodom koda, ako je vrijednost "True" (hrv. Istina) ništa se ne događa i kod nastavlja sa sljedećom stavkom u "Sequence" čvoru.



Slika 68: Drugi dio koda "NPCSpawner"-a.

Ali ako je kondicija ispunjena, pokreće se čvor "SpawnActor NPC" (hrv. StvoriGlumca NPC) te unutar ovog čvora je namješten broj postavki. Prvo pod priključak "SpawnTransform" (hrv. Stvaranje Transformacija) dodana vektorska vrijednost koja sadrži koordinate mjesta gdje se NPC ponovno stvoraju. Svaki od timova ima svoju, različitu vrijednost koja je na suprotnim krajevima mape.

Nakon toga pod "Owner" (hrv. Vlasnik) spajamo čvor "Get AI Controller" (hrv. Dobavi AI Kontroler) što postavlja NPC-a pod kontrolu umjetne inteligencije koja je napravljena. Nakon toga pod "Team Tag" (hrv. Tim Oznaka) dodaje svakom timu njihov tim.

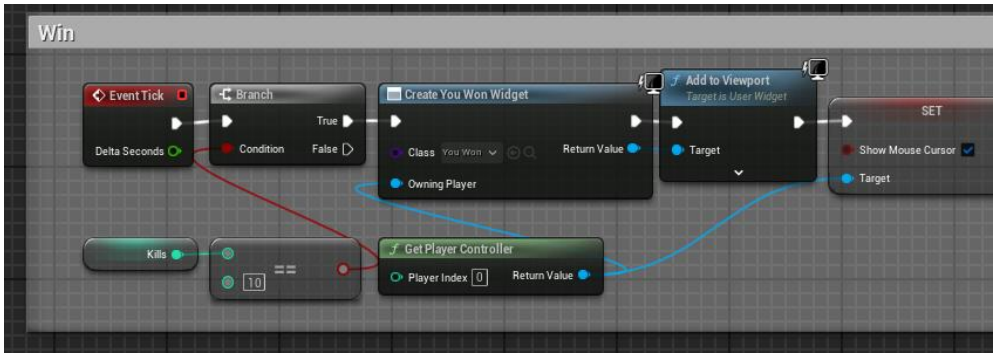
Nakon provedbe ovog čvora jednostavan "PrintString" (hrv. Printaj String) koji obavještava o ponovnom stvaranju NPC-a tog tima.

Na kraju što je veoma važno, boolean vrijednost koja je kao kondicija bila "Krivo" se sada postavlja na "Istina" i tako onemogućava konstantno stvaranje NPC-a istog tima.

3.8. IGRAČ IZ PRVOG LICA

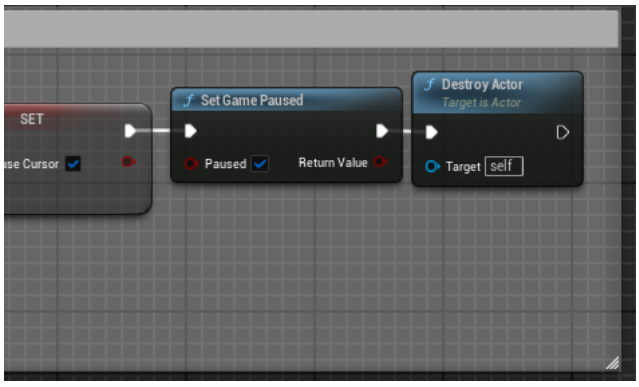
Važna komponenta za projekt je "BP_FirstPersonCharacter"(hrv. Blueprint Osoba iz prvog lica) ova komponenta je komponenta igrača, te igrač uzima kontrolu nad njim radi sudjelovanja u igri.

Ovoj komponenti dodan je sistem za praćenja broja smrti koje je uzrokovao koji je također i kondicija za pobjedu, sistem za ponovno stvaranje i par Widgeta, komponenti koji se koriste vizualnu prezentaciju informacija na igračevom ekranu.



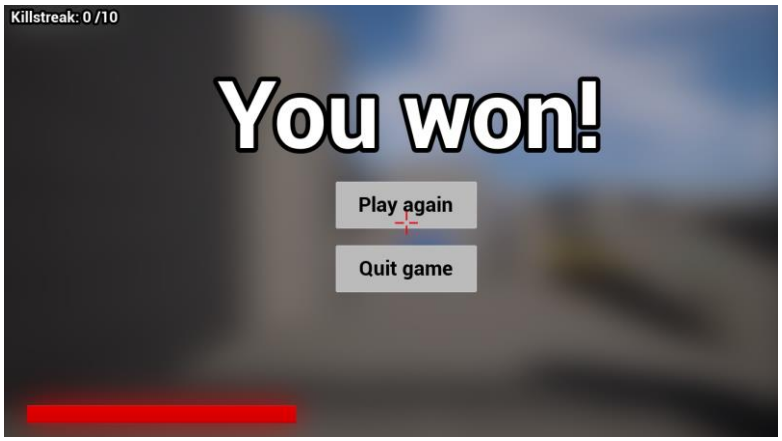
Slika 69: Početak slijeda brojača za bodove.

Brojač se sastoji od "Branch" čvora i funkcije za jednako "==", jednom kada integer "Kills" (hrv. Ubojstva) je jednak "10", kondicija je ispunjena i na ekran se stavlja Widget za pobjedu sa "Add to Viewport" (hrv. Dodaj na sučelje).



Slika 70: Kraj slijeda brojača za bodove.

Za kraj se igra pauzira i uništava se igračev glumac kojeg je kontrolirao.



Slika 71: Kraj igre.

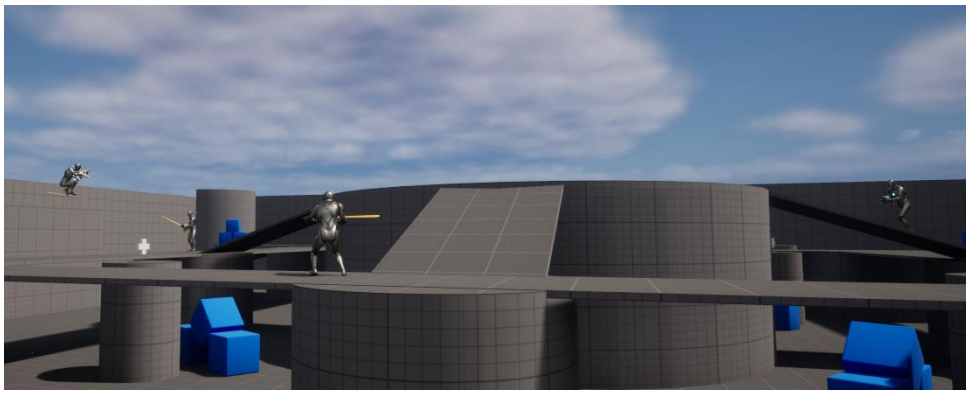
4. RASPRAVA I DISKUSIJA

Rad na kraju dodaje pošten broj novih sustava i komponenti, objašnjavane Blueprint koda preko tekstualnog medija nije idealan zadatak, ali je moguće. Svaki dio je rastavljen na osnovne funkcije koda i objašnjen. Jednom kada sav kod funkcionira zajedno, projekt može funkcionirati kako je zamišljeno. Tekstualni medij možda nije idealan za razradu Blueprint koda, ali za demonstraciju projekta je još lošiji, usprkos tome možemo zaključiti kako projekt funkcionira.

Projekt na početku stvara četiri NPC igrača s različitim timovima, te ih uz pomoć sekvence za patrolu i postavljenim navigacijskim mrežama sve vodi do istog područja gdje se potom uključuju u međusobnu borbu. Ovisno o podražajima koje dobivaju od "NPC_AI" glumca koji je ključan za percepciju, neki reagiraju na liniju vida, drugi na zvuk pucnjave ili primanje štete. Stablo ponašanja dodaje dodatne opcije u načinu borbe ovisno o broju zdravlja. Jednom kada je NPC poražen, ponovno se stvaraju u okoliš na zadanim lokacijama. Ove akcije mogu se događati u krug beskrajno, svaki put s različitim interakcijama. Jednom kada igrač odluči se pridružiti borbi, on ima mogućnost pobijediti tako da uspije zaraditi 10 uzastopnih bodova.

Promatrajući karaktere u borbi možemo okvirno prepoznati njihovo trenutno stanje oviseći o interakciji s drugim karakterima. Okoliš nudi veliku mogućnost različitih interakcije, razlike u vertikali dopuštaju borbu iz različitih kutova. Način borbe uključuje pucanje iz poluautomatskog oružja, te podražaj za svaki pojedini pucanj dolazi iz servisa za paljbu, a brzina njegove izvedbe je direktno povezana s brzinom otkucaja samog programa. Ako bi povećali brzinu otkucaja, povećavamo brzinu paljbe. Brzina paljbe nije jedina brojka za vrijeme ubijanja, uz to karakteri imaju nasumičnu min i max brojku što utječe na mjesto pogotka. Uz ove dvije varijable također imamo broj štete koju karakter prouzroči svaki put kada pogodi metu, trenutno je broj postavljen na "0.1" dok je ukupno zdravlje postavljeno na "1", što znači da karakter može primiti 10 pogodaka prije nego što bude poražen. Uz modifikaciju ove četiri varijable možemo lagano promijeniti vrijeme borbe.

Svaki dio interakcije moguće je modificirati do željenog učinka. Percepcije za vid, sluh, štetu mogu imati različitu veličinu zone podražaja. Percepcija za previđanje može pogađati lokaciju druže u budućnost. Stablo ponašanja može imati različite kondicije izvedbe. Vrijednosti postavljene kroz razradu su tek jedna od mogućnost koje nam se nude, te uz veliki broj iteracija dolazimo do rezultata koji nas najviše zadovoljava. Pri interakciji s igračem moguće je vidjeti nijanse između navigacije za borbu i za povlačenje, ali tek nakon određenog broja ponavljanja samog okršaja, što je dobra stvar kada dolazi do stvaranja iluzije inteligencije.



Slika 72: Prikaz borbe.

Rad je uspio pri demonstraciji fluidne i fleksibilne umjetne inteligencije unutar okoliša, svaka akcija ima svoj logički slijed događa zašto i kako se izvodi. Kod je teško pohvatati odjednom, gledajući kako sustav funkcionira s više različitih i odvojenih komponenti.

5. ZAKLJUČAK

Prvotna zamisao rada je bila napraviti kvalitetnu umjetnu inteligenciju, zatim je dodana i ideju uzastopnog okršaja između NPC-a. Radi prirode samog okršaja umjetne inteligencije protiv same sebe, broj detalja koji bi dolazili do izražaja u borbi s igračem sada više nisu očiti. Percepcija zvuka postaje veoma ne važan dio percepcije radi konstante pucnjave, opcija povlačenja pri niskom zdravlju također ne dolazi do izražaja radi neprestane borbe. Sustav za izbjegavanje također nije očit. Veliki broj problema dolazi iz borbe između NPC-a, jednostavno je prekaotična. Međusobne borbe je zanimljivo gledati, ali često preuzimaju važnost preko interakcije s igračem, što nije dobra stvar ako želimo prikazati umjetnu inteligenciju samom igraču. Dok borba s igračem može biti vođena na bilo kojem djelu mape, većina interakcija između NPC-a se događa u sredini gdje je postavljena niska cijena navigacije. Jeftina mreža postavljena preko tog područja obavlja svoju funkciju, ali zbog istog razloga, karakteri nemaju potrebu izaći iz tog prostora jednom kada ugledaju metu. Promjene u navigacijskoj mreži bi mogle promijeniti gdje se borba odvija, ali promjena same mape, bi vjerojatno bila bolje rješenje. Mapa s više asimetričnih ciljeva, umjesto jednog velikog centralnog bi pružila veću raznolikost u borbi.

Uz to, veliki dio iluzije umjetne inteligencije dolazi preko animacije, zvuka i suradnje s okolišem, dok NPC u projektu skače tijekom borbe i može navigirati po mapi, on ne sudjeluje s mapom, samo ju koristi za kretanje. Također vjerujem da bi bolje prikazao mogućnosti ove umjetne inteligencije s boljim fokusom na jednog NPC-a i njegovom interakcijom s igračem. S fokusom na jednim NPC-em svaka njegova akcija bi zahtijevala više pozornosti od igrača, za usporedbu od više isti karaktera koji svi rade istu stvar.

Priroda ovakvog projekta zahtjeva veliki broj iteracije mape i različitih postavi da bi rezultat pokazao svoju maksimalnu mogućnost. Gledajući kako je ovo treća ili četvrta iteracija projekta, mogućnosti koje su prikazane su zadovoljavajuće, ali ostavljaju još puno za poželjeti.

Rad na ovom projektu je trebao doći do završetka u jednom trenutku radi vremenskog ograničenja, bez takvog ograničenja vjerujem da je moguće dovesti sve dijelove ovog projekta na sasvim novu razinu.

Usprkos svemu projekt je dokaz velikog postignuća u dostupnosti programa i pomoćnog materijala za razvoj video igara. Nove poslovne paradigme Epic Games-a dopustili su Unreal Engine-u da postane dostupan svima koje zanima razvoj video igara. Uzimajući u obzir moje osobno iskustvo s ovakvim programima, koje je nikakvo, smatram da ovakav projekt ne bi bio moguć prije samo par godina.

6. LITERATURA

(1) Kushner, David (2003.) **Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture,** New York: Random House. ISBN 0-375-50524-5.
(2) Orkin, Jeff (2006.) **Three States and a Plan: The A.I. of F.E.A.R.** Monolith Productions / M.I.T. Media Lab, Cognitive Machines Group
(3) Aaronson, Scott (2011.) **Automata, Computability, and Complexity:** Lecture 3 MITopenCurse
(4) Sanglard, Fabian (2018.) **Game Engine Black Book: DOOM** Posljednji pristup 30.08.2023.
(5) Jagdale, Devang (2021.) **Finite State Machine in Game Development.** 384-390. 10.48175/IJARSCT-2062.
(6) https://modl.ai/halflife-halo/ Posljednji pristup 30.08.2023.
(7) https://twhl.info/wiki/page/VERC%3A_Half-Life_AI%2C_Schedules_and_Tasks Posljednji pristup 30.08.2023.
(8) https://docs.unrealengine.com/5.2/en-US/ Posljednji pristup 29.08.2023.
(9) https://en.wikipedia.org/wiki/Id_Software Posljednji pristup 29.08.2023.
(10) https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/ Posljednji pristup 30.08.2023.
(11) Russell, S. & Norvig, Peter. (2003). Artificial Intelligence, A Modern Approach. Second Edition.
(12) https://gamerant.com/id-software-developer-history-doom-wolfenstein-quake-microsoft/ Posljednji pristup 29.08.2023.
(13) https://industrial-ia.com/what-are-the-6-degrees-of-freedom-6dof-explained/ Posljednji pristup 30.08.2023.
(14) https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/MouseInterface/BlueprintActors/ Posljednji pristup 30.08.2023.
(15) https://docs.unrealengine.com/5.1/en-US/first-person-template-in-unreal-engine/ Posljednji pristup 30.08.2023.
(16) https://www.youtube.com/watch?v=bx7taRBjJgM Laley, Ryan: Unreal Engine 5 Tutorial - AI Part 3: Perception System. Posljednji pristup 30.08.2023.
(17) https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/ArtificialIntelligence/BehaviorTrees/BehaviorTreeQuickStart/ Posljednji pristup 30.08.2023.
(18) https://docs.unrealengine.com/5.0/en-US/BlueprintAPI/AI/UseBlackboard/ Posljednji pristup 30.08.2023.
(19) https://docs.unrealengine.com/5.2/en-US/unreal-engine-behavior-tree-node-reference-services/ Posljednji pristup 30.08.2023.
(20) https://www.youtube.com/watch?v=Uu0UK2uvjbQ Laley, Ryan: Unreal Engine 5 Tutorial - AI Part 11: Intro to EQS. Posljednji pristup 30.08.2023.
(21) https://www.youtube.com/watch?v=kZVIa2uWOiM Gorka Games: How to Make a Simple Health System in Unreal Engine 5. Posljednji pristup 30.08.2023.

(22) https://www.gamedeveloper.com/blogs/the-ai-of-doom-1993 Posljednji pristup 29.08.2023.
(23) https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/ArtificialIntelligence/NavigationSystem/ModifyingTheNavigationMesh/ModifyingtheNavigationSystem/ Posljednji pristup 30.08.2023.
(24) https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work Posljednji pristup 29.08.2023.
(25) https://medium.com/@mlbors/what-is-a-finite-state-machine-6d8dec727e2c#:~:text=Introduction,artificial%20intelligence%2C%20games%20or%20linguistics Posljednji pristup 29.08.2023.
(26) http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-045j-automata-computability-and-complexity-spring-2011 Posljednji pristup 30.08.2023.
(27) https://fabiensanglard.net/gebbdoom/ Posljednji pristup 30.08.2023.
(28) https://www.researchgate.net/publication/355518086_Finite_State_Machine_in_Game_Development Posljednji pristup 30.08.2023.
(29) https://web.archive.org/web/20061027224334/http://collective.valve-erc.com/index.php?doc=1018030771-90025600 Posljednji pristup 30.08.2023.

7. POPIS SLIKA

8. POPIS MANJE POZNATIH RIJEČI

Game engine – jezgra igra korištena za pokretanje video igara.

NPC(Non-Player Character) - karakter koji nije igrač, često kontroliran od strane umjetne inteligencije.

Integer – brojevna informacija korišten za programiranje.

String – tekstualna informacije korištena za programiranje.

Bool – binarna informacije korištena za programiranje.

Bit – mjera digitalne informacije, jedan bit je jedan član od 0 ili 1.

FPS(First person shooter) - vrsta igre gdje igrač ima kontroli nad likom iz prvog lica i koristi najčešće vatreno oružje.

FSM(Finite state machine) - automat s konačnim brojem stanja, sustav korišten unutar umjetne inteligencije za raspodjelu akcija.

GOAP(Goal oriented action planning) - planiranje akcija orijentirano na cilj, sustav korišten unutar umjetne inteligencije za raspodjelu akcija.

Stablo ponašanja - sustav korišten unutar umjetne inteligencije za raspodjelu akcija.

Widget – sustav unutar jezgre igre za prenošenje informacija na igračevo sučelje.