Analiza primjene modela strojnog učenja u videoigri The Last of Us

Glavač, Jakov

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Jakov Glavač

ANALIZA PRIMJENE MODELA STROJNOG UČENJA U VIDEOIGRI THE LAST OF US

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Jakov Glavač

Matični broj: 0016152889 (49362)

Studij: Informacijske tehnologije i digitalizacija poslovanja

ANALIZA PRIMJENE MODELA STROJNOG UČENJA U VIDEOIGRI THE LAST OF US

ZAVRŠNI RAD

Mentor:

doc. dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2024.

Jakov Glavač

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Tema ovog završnog rada je analiza metoda umjetne inteligencije koje su primijenjene kod neprijatelja u svjetski poznatoj videoigri *The Last of Us*. Rad se sastoji od dva dijela: teorijskog i praktičnog. U teorijskom dijelu detaljno su opisane i analizirane metode umjetne inteligencije koje se često koriste u modernim videoigrama, s posebnim naglaskom na akcijske igre. Razmatran je model konačni automat i model stablo ponašanja, pri čemu su objašnjene prednosti i nedostaci svakog modela u kontekstu razvoja neigrajućih likova. Također, detaljno je opisano kako su ove metode primijenjene kod zaraženih neprijatelja u videoigri *The Last of Us*, a pritom se nastojalo razumjeti odluke dizajnerskog i programerskog tima kompanije *Naughty Dog*. Praktični dio rada obuhvaća implementaciju odabranih metoda umjetne inteligencije i mehanika videoigre *The Last of Us* u razvojnom okruženju Unreal Engine 5. Za praktični dio rada izrađen je projekt i opisan je postupak izrade kraće razine, kako bi se praktično demonstrirao način na koji umjetna inteligencija može funkcionirati u akcijskoj videoigri. Cilj praktičnog dijela je simulirati ponašanje neprijatelja, to jest neigrajućih likova koristeći alate i funkcionalnosti koje razvojno okruženje Unreal Engine 5 nudi.

Ključne riječi: umjetna inteligencija; strojno učenje; konačni automat; stablo ponašanja; videoigre; neigrajući likovi; The Last of Us

Sadržaj

1. Uvod

U 21. stoljeću umjetna inteligencija (engl. *Artificial intelligence - AI*) ima široku primjenu u različitim područjima. Jedno od područja primjene jesu svakako moderne računalne videoigre koje zauzimaju veliki udio na tržištu. Jedna zanimljivost jest kako je industrija videoigara prerasla, odnosno postala jača od glazbene i filmske industrije zajedno, što je svakako pokazatelj kako u današnje vrijeme videoigre postaju sve popularnije, a industrija videoigara sve jača.

U videoigri umjetna inteligencija može biti primijenjena u različite svrhe. Jedna od svrha jest upravo i primjena umjetne inteligencije kod neigrajućih likova (engl. *Non-player character - NPC*) čime se i bavim u ovom radu. Važno je napomenuti kako utjecaj umjetne inteligencije u razvoju modernih videoigara neprekidno raste. Umjetna inteligencija uglavnom je zadužena za upravljanje ponašanjem neigrajućih likova, no pomaže stvoriti i dinamične scenarije koji igraču omogućuju dublju povezanost s likovima i svijetom u kojem se nalazi. Proučavanje metoda umjetne inteligencije koje su primijenjene u videoigri *The Last of Us* svakako može pružiti neke uvide u mogućnosti koje umjetna inteligencija trenutno pruža u razvoju videoigara, kao i potencijalna poboljšanja za razvoj budućih videoigara.

Tema ovog završnog rada je analiza metoda umjetne inteligencije te na koji način je umjetna inteligencija primijenjena kod neigrajućih likova koji predstavljaju ljudske neprijatelje u vrlo poznatoj videoigri *The Last of Us*. Videoigra *The Last of Us* je ostvarila veliki uspjeh u industriji videoigara, ne samo zbog priče i impresivne grafike, već i zbog sofisticirane primjene metoda umjetne inteligencije koje doprinose boljem iskustvu igranja. Završni rad sastavljen je od dva dijela. Prvi dio rada je teoretske prirode, to jest opis same videoigre i modela umjetne inteligencije koji su primijenjeni kod neprijatelja. Zatim, drugi dio rada je praktični dio koji obuhvaća pojednostavljenu implementaciju odabranih metoda umjetne inteligencije i mehanika videoigre u razvojnom okruženju Unreal Engine 5.

Videoigra *The Last of Us* je ustvari franšiza koja se sastoji od nekoliko dijelova. Za sada su izdana dva nastavka poznate videoigre, a prvi dio službeno je objavljen još 2013. godine za platformu *Playstation 3*. Videoigra me se dojmila već nakon što sam je prvi puta odigrao, upravo zbog zanimljive priče koja se razvija tijekom igranja, a dojmila me se također i mehanika igranja same videoigre. Iz tog razloga, odlučio sam se kako ću napraviti analizu metoda umjetne inteligencije upravo za tu videoigru, koja mi je još onda kada sam je prvi puta odigrao prirasla srcu. Također, svakako me zanima područje razvoja videoigara, kao i područje programiranja te pomoću ovog rada nastojim i produbiti svoje znanje upravo u tim područjima.

2. Metode i tehnike rada

Što se tiče same implementacije ovog dokumenta, za izradu i uređivanje koristio sam servis *Overleaf* i programski jezik LaTeX, a za uspješnije referenciranje izvora alatom *Zotero*.

S obzirom da je prvi dio rada teoretski uvod u temu, prvo je bilo potrebno istražiti dostupnu literaturu. Kao glavni izvor literature odabrao sam besplatne verzije poglavlja iz knjige *Game AI Pro Volume 2* koju je uredio Steven Rabin, a iz same knjige odabrao sam trideset i treće poglavlje čiji je autor Mark Botta. Iz tog poglavlja mogao sam saznati iz prve ruke sve informacije relevantne za temu ovog rada. Naravno koristio sam se i nekim popratnim izvorima kako bih podrobnije istražio temu i uspješnije izradio završni rad.

Za praktični dio rada odlučio sam se na pojednostavljenu implementaciju modela umjetne inteligencije koji su predstavljeni u ovom radu kod neigrajućih likova. Pojednostavljenu iz razloga jer je igra *The Last of Us* ogroman projekt iza kojeg stoji ogroman tim i cijela kompanija koja je zaslužna za uspješnost igre, no za cilj imam pokazati kako je danas uz pomoć dobrih alata moguće izraditi nešto relativno slično stvarnom projektu. Za razvojno okruženje u kojem ću implementirati praktični dio ovog rada odabrao sam Unreal Engine 5.

Danas na tržištu postoji nekoliko solidnih izbora razvojnih okruženja za razvoj videoigara. Neki od njih su Unity, Godot ili spomenuti Unreal Engine 5. Svaki od njih ima neke svoje prednosti i nedostatke.

Alat Unity imao sam već prilike isprobati, tako da poznajem njegove prednosti, ali svjestan sam i nekih nedostataka tog alata. Kako bih proširio svoje postojeće znanje u području razvoja videoigara, želim isprobati i neke druge alate za razvoj, zbog čega sam se odlučio kako je ovaj završni rad idealna prilika za to.

Unreal Engine 5 je najnovija verzija alata Unreal Engine razvijenog od strane *Epic Games* kompanije. Predstavljen je 2020. godine, a pušten u javnost 2021. godine. Unreal Engine 5 donio je nekoliko ključnih inovacija u područje razvoja modernih videoigara koje pomažu kreiranju realističnijih i impresivnijih iskustava. Dulje vrijeme alat Unity je bio najpopularniji alat za razvoj videoigara koji ima vrlo široku zajednicu razvijača, no polako Unreal Engine preuzima tržište, zbog opsežnih mogućnosti i inovacija koje alat nudi.

3. Videoigra The Last of Us

The Last of Us je serija videoigara, odnosno franšiza u dva nastavka koju je kreirao studio *Naughty Dog*, a objavio *Sony Interactive Entertainment* [1]. *The Last of Us* je post-apokaliptična akcijsko-avanturistička videoigra u trećem licu za jednog igrača [1]. Prvi dio objavljen je 2013. godine za platformu *Playstation 3*. Sama radnja prati avanturu u kojoj protagonist Joel i deuteragonist Ellie putuju kroz post-apokaliptični svijet Sjedinjenih Američkih Država, kako bi pronašli lijek za svijet koji je zahvatila pandemija gljivične infekcije [1]. Na putu susreću se s raznim preprekama i raznim zaraženim neprijateljima koje je infekcija pretvorila u smrtonosna čudovišta [1]. Videoigra je ostvarila veliki uspjeh zbog vrlo emocionalne priče, napredne grafike i detaljno razrađenih likova.

Franšiza se proširila i izvan svijeta videoigara [1]. Nastali su razni stripovi, figurice, natpisi na temu videoigre, a snimljena je i istoimena televizijska dramska serija emitirana na platformi *HBO Max* [1]. Tako je zapravo franšiza postala nešto puno više od same videoigre, što bi se slobodno moglo nazvati i kulturnim fenomenom.

U ovom radu naglasak je na analizi primijenjenih modela za razvoj umjetne inteligencije kod neprijatelja u videoigri *The Last of Us* koja je objavljena 2013. godine. U kontekstu razvoja sustava umjetne inteligencije i implementacije raznih naprednih tehnologija za razvoj modernih akcijskih videoigara, korištenje modela strojnog učenja (engl. *Machine learning - ML*) može otvoriti nove mogućnosti za evoluciju i napredak serijala. S obzirom da nije službeno potvrđeno da je strojno učenje stvarno korišteno prilikom razvoja videoigre *The Last of Us* i da nije bilo moguće pronaći konkretne podatke o primjeni modela strojnog učenja zbog ograničene dostupnosti literature, tema rada je proširena i na analizu metoda umjetne inteligencije općenito u kontekstu razvoja modernih videoigara.

Strojno učenje može se iskoristiti za dinamično prilagođavanje ponašanja neprijatelja tokom igranja, omogućujući neprijateljima da uče iz interakcija s igračem i razvijaju nove taktike u borbi protiv igrača, čime bi modeli umjetne inteligencije neprijatelja koji su analizirani u ovom radu postali izazovniji za samog igrača. Jedan praktični primjer gdje se strojno učenje može iskoristiti jest ukoliko se igrač prilikom interakcije s neprijateljom ponaša agresivno, umjetna inteligencija koja upravlja neprijateljima će to naučiti te će se prilikom budućih interakcija s igračem ponašati agresivnije. Nadalje, tehnike strojnog učenja mogle bi poboljšati proceduralnu generaciju animacija likova, čineći tako pokrete neigrajućih likova još prirodnijim i realističnijim.

Osim toga, strojno učenje moglo bi unaprijediti personalizaciju iskustva igrača, gdje bi se dinamički prilagođavala težina igre (engl. *Dynamic difficulty adjustment*) i reakcije neigrajućih likova na temelju analize ponašanja samog igrača. Na taj način može se stvoriti vrlo dinamično iskustvo igranja koje odgovara individualnom stilu pojedinog igrača. U kontekstu složenog narativnog dizajna, strojno učenje također ima potencijal prilagoditi tijek priče na temelju donošenja odluka igrača, čime bi se stvorile još dublje i personaliziranije priče, što bi u konačnici omogućilo igraču još dublju povezanost sa samim likovima. Implementacija ovih naprednih tehnologija i modela strojnog učenja mogla bi dodatno podići standard u industriji razvoja modernih videoigara.

4. Neprijatelji u videoigri The Last of Us

Neprijatelji (engl. *Enemy*) su jedan od najvažnijih izazova s kojima se igrač suočava tijekom igre. Što se tiče mehanika borbe, igrač se u videoigri može protiv neprijatelja boriti pomoću vatrenog oružja, improviziranog oružja koje je moguće izraditi iz raznih dijelova ili zaskočiti neprijatelja šuljanjem (engl. *Stealth*) [1]. Igrač tijekom igre ima kontrolu nad glavnim likom Joelom, a kontrolu nad ljudskim neprijateljima, deuteragonistom Ellie i ostalim likovima ima umjetna inteligencija [1]. Najvažnija vrsta, to jest glavni tip neprijatelja u igri jesu zaraženi (engl. *Infected*), a postoje i nezaraženi (engl. *Non-infected*) ljudski neprijatelji [2]. Mehanika borbe protiv obje vrste neprijatelja je poprilično slična, no postoje razlike u kretnjama, ponašanjima i mogućnostima koje pojedina vrsta neprijatelja posjeduje.

4.1. Zaraženi neprijatelji (Infected)

Zaraženi neprijatelji u kontekstu videoigre *The Last of Us* zapravo su ljudi koji su zahvaćeni od strane gljivične infekcije zvane *Cordyceps* koja napada ljudski mozak, a prolaskom vremena događa se i fizička transformacija koja ih pretvara u užasne oblike koji podsjećaju na hodajuća čudovišta [2]. Rezultat infekcije su vrlo agresivna stvorenja koja napadaju plijen kako bi se ugrizom žrtve infekcija mogla proširiti [2]. Kao što je već spomenuto zaraženi su glavna grupa neprijatelja u videoigri protiv kojih se igrač mora boriti, a ti će neprijatelji u svakom trenutku napasti nezaražene ljude ili životinje koje se pojave u blizini [2]. Nakon što infekcija zahvati žrtvu, infekcija se nastavlja razvijati kroz nekoliko faza tijekom kojih se žrtvi mijenja fizički izgled, osjetila i motoričke sposobnosti [2]. U nastavku poglavlja svaki tip zaraženog neprijatelja je detaljnije pojašnjen.

Ukratko, u prvom dijelu videoigre postoje četiri vrste zaraženih neprijatelja koje predstavljaju životni ciklus infekcije. Prva i najčešća faza poznata je pod nazivom *Runner*, a njih definira brzina, nespretni napadi i napadi u grupama [3]. Zatim slijedi faza *Stalker* tijekom koje se neprijatelj skriva u mraku i napada svoj plijen iz zasjede [3]. Tijekom treće faze *Clicker* već su vidljive i velike fizičke promjene kod neprijatelja [3]. Ova vrsta neprijatelja je slijepa, što im je omogućilo razvoj eholokacije pomoću koje se kreću okolinom [3]. Posljednja i vrlo rijetka faza jest *Bloater* [3]. Ova vrsta neprijatelja poprilično se sporo kreće, no posjeduje iznimnu izdržljivost i snagu [3]. Kako bi se sve te karakteristike raznih vrsta neprijatelja preglednije prikazale, izrađena je tablica 1.

Tablica 1: Prikaz karakteristika zaraženih neprijatelja prema vrsti [3, str. 408]

Vrsta	Runner	Stalker	Clicker	Bloater
Brzina	Brza	Brza	Srednja	Spora
Vid	Ograničen	Ograničen	Slijep	Slijep
Rijetkost	Često	Rijetko	Rijetko	Vrlo rijetko
Borba	Napadi u grupama	Zasjede u mraku	Ograničena ranjivost u bliskoj borbi	Oklopljeni, napadaju iz daljine, neranjivi na borbu iz blizine

Važno je spomenuti kako su ovi tipovi neprijatelja poredani kronološki s lijeva na desno, kako se ujedno igrač tijekom igre s njima i susreće. Analogno tome, igrač se ujedno najlakše bori protiv prve vrste neprijatelja, a najteže protiv četvrte vrste neprijatelja. U nastavku dokumenta analizirano je nekoliko modela umjetne inteligencije koje su razvojni programeri studija *Naughty Dog* primijenili nad zaraženim neprijateljima i na koji način su pristupili njihovu razvoju.

4.2. Primijenjeni sustav umjetne inteligencije

Razvoj sustava umjetne inteligencije zahtijevao je od razvojnih programera velik broj iteracija i eksperimentiranja [3]. Kako bi to postigli programeri su se odlučili za modularnu arhitekturu sustava umjetne inteligencije koja im je omogućila lakše dodavanje, brisanje ili izvršavanje promjena u kodu tijekom razvoja videoigre [3]. Također, takav pristup olakšao je i kreiranje novih likova koji se različito ponašaju u odnosu na okolinu, dok je kod ostao "čist" i pregledan [3]. Modularna arhitektura je programerima bila vrlo važna iz razloga jer je jednostavan kod bio ključan za brzo provođenje iteracija i eksperimentiranje s novim idejama, likovima itd. [3].

Sustav umjetne inteligencije koji je primijenjen u videoigri izgrađen je na temelju ideje da postoji logika donošenja odluka na visokoj razini (*Skills*) koja odlučuje što bi neprijatelj trebao učiniti i mogućnosti neprijatelja na nižoj razini (*Behaviors*) koje implementiraju te odluke [3]. Takvo razdvajanje u sustavu omogućilo je da se kod likova odnosno neprijatelja koji posjeduju logiku na visokoj razini ponovno iskoriste neke od mogućnosti na nižoj razini (*Behaviors*) [3]. Jedan od ključnih aspekata kojeg su programeri primijenili kako bi održali kod čistim i jednostavnim jest da se nije koristilo referiranje na tipove neprijatelja u samome kodu, već su se umjesto toga specificirali skupovi karakteristika koji definiraju pojedinu vrstu neprijatelja [3]. Botta također navodi [3] kako zapravo svi tipovi zaraženih neprijatelja dijele jednu C++ klasu, a razlikuju se jedino prema skupu vještina i vrijednostima varijabli unutar njihovih datoteka.

Nakon opisa arhitekture i dizajna samog sustava, valja odgovoriti i na pitanje kako je sustav u konačnici implementiran. Dakle, umjetna inteligencija za svaku specifičnu vrstu neprijatelja implementirana je kao set vještina (*Skills*) koji zatim poziva odnosno aktivira hijerarhiju

ponašanja neprijatelja (*Behaviors*) [3]. Set vještina i hijerarhija ponašanja neprijatelja kako opisuje Botta [3] implementirani su kao konačni automat (engl. *Finite-state machines - FSM*), što svakako znači da je jedan od modela umjetne inteligencije koji su razvojni programeri primijenili u videoigri upravo konačni automat. Vještine su zapravo bile prilagođene za specifične tipove neprijatelja, dok su ponašanja dosta često bila ponovno iskorištena [3]. Primjerice, nezaraženi neprijatelji i zaraženi neprijatelji ne dijele nikakve vještine, ali ipak dijele većinu ponašanja [3].

Nadalje, svaki tip neprijatelja sadrži i listu vještina koje su poredane prema prioritetu [3]. Razvojni programeri su testirali svaku vještinu zasebno prema prioritetu krenuvši od visokog prema nižem, kako bi uspješno utvrdili izvršavaju li se ispravno [3]. Također, ključno je da vještina koja se nalazi zadnja na listi bude uvijek važeća, kako ne bi došlo do nekontroliranog ponašanja neprijatelja ili potencijalnih grešaka prilikom igranja [3]. Vještina se izvodi tako dugo dok ne završi ili dok je prekinuta od strane vještine koja ima veći prioritet [3]. U svrhu prikaza liste vještina poredanih prema prioritetu, izrađena je tablica 2.

Tablica 2: Prikaz prioritetnih vještina za zaražene tipove neprijatelja [3, str. 412]

Runner	Stalker	Clicker	Bloater
U plamenu	U plamenu	U plamenu	U plamenu
Potjera	Zasjeda	Potjera	Bacanje
Pretraživanje	Spavanje	Pretraživanje	Potjera
Slijeđenje	Lutanje	Slijeđenje	Pretraživanje
Spavanje		Spavanje	Slijeđenje
Lutanje		Lutanje	Spavanje
			Lutanje

Vještina "Pretraživanje" postaje istinita, odnosno počinje se izvršavati kada zaraženi neprijatelj izgubi trag igrača tijekom potjere [3]. Ideja koja stoji iza ove vještine jest da se igrača natjera na kretanje okolinom, dok neprijatelj pretražuje obližnja skrovišta [3]. Vještina pretraživanja okoline implementirana je pomoću specijalnog ponašanja *infected-canvass* što je omogućilo razvojnim programerima izradu nepredvidivih i instinktivnih kretnji zaraženih neprijatelja, dok traje potraga za igračem. Detaljan opis načina na koji specijalno ponašanje *infected-canvass* funkcionira, nalazi se unutar sekcije 4.3.3. Infected-canvass.

Sljedeća vještina koju valja razmotriti je "Potjera". Većina zaraženih neprijatelja posjeduje dosta trivijalne borbene vještine [3]. U suštini, kada zaraženi "osjeti" prisutnost igrača, nastoji što prije stići do njega te ga odmah i napasti [3]. To je implementirano upravo pomoću vještine "Potjera". Kada neprijatelj prvi puta uoči igrača, pokreće se animacija i reproducira se specifičan zvuk koji daje igraču do znanja da je uočen od strane neprijatelja, a sama animacija orijentira neprijatelja prema smjeru igraču [3]. Vještina "Potjera" također poziva ponašanje *move-to* koje pruža sučelje za navigacijski sistem i ponašanje ujedno osigurava kretanje neprijatelja prema lokaciji igrača [3]. Kako se igrač ili neki drugi entitet kreće okolinom, tako *move-to* ponašanje neprekidno ažurira putanje i ostavlja roditeljskom ponašanju da nadzire uspješnost ili neuspješnost izvršavanja [3]. Također, za vrijeme potjere neprijatelj se nasumično zaustavlja

kako bi se reorijentirao, što omogućuje igraču da se sakrije ili pripremi za napad [3].

S obzirom da zaraženi neprijatelji nemaju razvijen način komunikacije, na podražaje u okolini reagiraju samo instinktivno koristeći svoja osjetila. Analogno tome, kako navodi Botta [3], tijekom razvoja i testiranja ukoliko je jedan zaraženi neprijatelj osjetio prisutnost nekog entiteta ili igrača i krenuo u potjeru, drugi zaraženi koji su se nalazili u blizini ostali su nesvjesni igračeve prisutnosti. Međutim, u praksi to je učinilo neprijatelje slabije osjetljivima na okolinu, zbog čega je razvijena upravo vještina "Slijeđenje" [3]. Vještina slijeđenja omogućuje jednom neprijatelju da slijedi drugog neprijatelja koji juri za nekim entitetom. Neprijatelj koji slijedi drugog neprijatelja zapravo ne dobiva informacije o entitetu za kojim se potjera izvršava, već samo nagonski slijedi drugog neprijatelja [3]. Posljedično, može se dogoditi da jedan neprijatelj kada krene u potjeru, usput aktivira i ostale neprijatelje na putu do igrača, no na lokaciju prvi stiže neprijatelj koji je osjetio podražaj i prisutnost igrača [3].

Vještina "Zasjeda" specifična je za jednu vrstu zaraženih neprijatelja *Stalker*. Vještina "Zasjeda" zapravo zamijenjuje vještinu "Potjera" koju koriste svi ostali tipovi zaraženih neprijatelja [3]. Kako navodi Botta [3], ova vještina je nastala tijekom kasnijeg razvoja, kada je tim želio uvesti neke nove raznolikosti kod neprijatelja. Tim je eksperimentirao sa raznim tipovima neprijatelja koji napadaju u mraku i pružaju igraču dobar izazov. Nakon nekoliko iteracija, tim se odlučio na taktiku da se igrač kreće u mraku od zaklona do zaklona, pružajući igraču kratke uvide neprijatelja koji se brzo kreće i sakriva [3]. Shodno tome, vrsta neprijatelja *Stalker* ustvari čeka u zaklonu tako dugo dok se igrač ne približi, a zatim napada i ponovno se vraća u zaklon [3]. Takav jednostavan koncept dodatno je povećao osjećaj straha i neizvjesnosti tokom igranja, što utječe na igrača na način da u susretima s ovom vrstom neprijatelja, postane defenzivniji i prema svakom kutu pristupa oprezno [3].

Sljedeća vještina koja je isto tako tipična samo za jednu vrstu neprijatelja je "Bacanje", a pojavljuje se kod vrste neprijatelja pod nazivom *Bloater*. *Bloater* je ustvari jedina vrsta neprijatelja koja posjeduje mogućnost projektilnog napada [3]. Kako bi izvršio napad *Bloater* otkida dijelove gljivičnih izraslina sa svojeg zaštitnog oklopa, a zatim te komadiće baca u smjeru kretanja igrača, stvarajući tako oblak pun čestica koji usporava igraća i na kraće vrijeme smanjuje vidno polje [3]. Igrač mora uništiti slojeve zaštitnog oklopa kako bi pobijedio ovu vrstu neprijatelja, a vještina "Bacanje" tijekom borbe reproducira animacije koje odabiru razne slojeve zaštitnog oklopa tako dugo dok igrač ne uništi cijeli oklop, a time *Bloater* i umire [3].

Vještina koja ima najviši prioritet je vještina "U plamenu" te ujedno ona smije prekinuti izvršavanje bilo koje druge vještine [3]. Za razliku od drugih vještina koje posjeduju zaraženi neprijatelji, vještina "U plamenu" ne pokazuje namjere neprijatelja [3]. Kod ove vještine cilj nije bio pokazati svjesne odluke i namjere kod neprijatelja, već na neki način dozvoliti neprijatelju adekvatnu reakciju na nastalu situaciju i učiniti njegovo ponašanje uvjerljivim i zanimljivim [3]. Vještina "U plamenu" radi na način da poziva specijalno ponašanje *infected-canvass* unutar manjeg područja sa određenim skupom animacija, omogućavajući tako neprijatelju vrlo kaotično kretanje uokolo, bez zalijetanja u razne prepreke i zidove [3].

Na dnu svakog stupca unutar tablice 2 nalazi se vještina "Lutanje" koja ima najniži prioritet, što je jednako za sve vrste zaraženih neprijatelja. Vještina "Lutanje" izvršava se kada

se ne izvršava niti jedna druga vještina [3]. Primjerice, kada neprijatelj ne primjećuje nikakve podražaje u okolini, on će se kretati okolinom nasumično ili u predvidljivim uzorcima [3]. Izazov za tim bio je kreirati lutanja neprijatelja okolinom na način da igrač bude prisiljen brzo donositi taktičke odluke o napadima, šuljanju ili distrakciji neprijatelja [3]. Dizajneri mogu odrediti hoće li se neprijatelj kretati fiksnom ili nasumičnom rutom, a svaki tip kretnje ima svoje koristi [3]. Fiksna ruta je dosta predvidljiva što omogućava igraču lakše šuljanje okolinom kako ne bi uznemirio neprijatelja, dok je nasumična ruta korisna za pokrivanje većeg područja okoline na nepredvidiv način, pružajući tako igraču drugačiju vrstu izazova [3]. U praksi, fiksne rute su iskorištene kod prvog susreta između igrača i neprijatelja, a nasumične rute nakon izlaska iz borbe između igrača i neprijatelja, kada zapravo neprijatelj završi daleko od originalne rute kojom se kretao [3].

Ostaje još jedna vještina koju valja objasniti, a to je vještina "Spavanje". Dizajneri mogu konfigurirati zaražene neprijatelje tako da tijekom neutralnog stanja koriste vještinu "Spavanje" umjesto vještine "Lutanje" [3]. Ovo neutralno stanje, odnosno vještina "Spavanje" zapravo umanjuje osjetila zaraženih neprijatelja [3], olakšavajući tako igraču šuljanje okolinom, zaobilaženje neprijatelja i sl. Kada se neprijatelj nalazi u stanju spavanja, on može reagirati na promjene u okolini na nekoliko načina [3]. Ukoliko se igrač malo brže pomiče u blizini neprijatelja, on će se kratko trznuti, što igraču daje do znanja kako mora biti oprezniji prilikom kretanja [3]. S druge strane ukoliko igrač proizvodi glasnije zvukove kretnjama okolinom, neprijatelj će se probuditi, odnosno prestaje se izvršavati vještina "Spavanje" i ujedno se aktivira specijalno ponašanje *infected-canvass* pomoću kojega se neprijatelj kreće do mjesta gdje se dogodila promjena u okolini prije nego li se vrati u stanje spavanja [3]. Međutim, ukoliko igrač nastavi proizvoditi glasne zvukove, neprijatelj će se naglo probuditi i prijeći na vještinu "Potjera" kako bi uhvatio igrača [3]. U slučaju da igrač uspije pobjeći neprijatelju, neprijatelj će prijeći na vještinu "Lutanje" umjesto vraćanja na stanje spavanja [3].

Vještine su za svaki tip neprijatelja poredane prema prioritetu, zbog toga što prioritet određuje koja vještina smije prekinuti drugu vještinu [3]. Tako primjerice vještina "Potjera" smije prekinuti vještinu "Pretraživanje", no opet "Potjera" može biti prekinuta od strane vještine "U plamenu" itd. [3]. Kao što je prethodno spomenuto, jedna od vještina mora uvijek biti važeća, kako ne bi došlo do nekontroliranog ponašanja neprijatelja. U tablici 2 je vidljivo kako je to upravo vještina "Lutanje" koja predstavlja tzv. *fallback* stanje [3] koje je jednako za sve tipove neprijatelja. Dakle, ukoliko zaraženi neprijatelj nije zauzet izvršavanjem neke od vještina, on će nastaviti lutati okolinom [3].

4.3. Primijenjeni modeli umjetne inteligencije

Nakon objašnjenja arhitekture na temelju koje su programeri sagradili umjetnu inteligenciju neprijatelja u videoigri *The Last of Us*, vrijeme je i za dublju analizu samih modela koje su odlučili primijeniti za njihov razvoj. Kao što je spomenuto u prethodnom poglavlju, jedan od modela iskorišten za razvoj umjetne inteligencije kod neprijatelja jest konačni automat (engl. *Finite-state machine - FSM*) pomoću kojega su implementirane vještine i ponašanja samih neprijatelja. Zatim još jedan model koji je primijenjen kako bi se kreirala složenija i hijerarhijski

strukturirana ponašanja samih neprijatelja jesu Stabla ponašanja (engl. *Behavior Trees - BT*).

Vezano uz prethodno spomenutu vještinu "Potraga" koju zaraženi neprijatelji posjeduju, veže se posebno ponašanje *infected-canvass* koje se aktivira kada zaraženi neprijatelj kreće pretraživati okolinu nakon percipiranog zvuka [3], kako bi pronašao izvor gdje se zvuk dogodio u okolini s ciljem pronalaska žrtve, odnosno igrača.

4.3.1. Konačni automat (*Finite-state machine*)

Konačni automat ili automat konačnih stanja je diskretni matematički model koji se već dugo koristi u raznim područjima poput računarstva, elektrotehnike, matematike itd. [4]. Sastoji se od konačnog broja stanja, prijelaza između definiranih stanja i akcija koje se izvršavaju [4]. Preciznije, konačni automat definira uvjete koji moraju biti zadovoljeni kako bi se stanje automata promijenilo [4]. Što se tiče stanja, stanje zapravo određuje kako se automat treba ponašati.

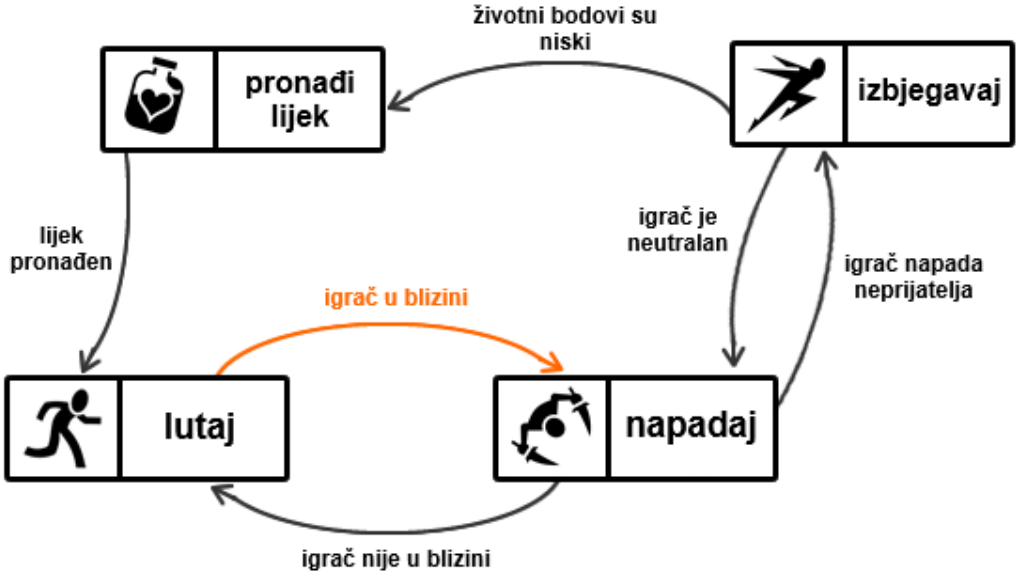
Konačni automat ima široku primjenu u raznim područjima, no u ovom radu naglasak je na primjeni dotičnog modela za programiranje umjetne inteligencije kod neigrajućih neprijatelja u videoigri. Neke od značajnijih videoigri u kojima je kroz povijest primijenjen model konačnog automata za kreiranje umjetne inteligencije su primjerice *Half-Life* [5] i *Pac-Man* [6], no danas se i dalje model dosta često koristi prilikom razvoja modernih videoigara.

Ključno je za spomenuti kako konačni automat može biti samo u jednom stanju u isto vrijeme, dok se tranzicija u drugo stanje može izvršiti samo ako je odgovarajući uvjet zadovoljen [6]. U suštini, konačni automat se sastoji od tri ključne komponente [6]:

- konačni broj stanja koja sadrže informacije o zadatku
- broj tranzicija između stanja koje označava promjenu stanja i opisane su uvjetom koji treba biti ispunjen
- skup akcija koje se trebaju slijediti unutar svakog stanja

U kontekstu videoigara konačni automat se dosta često koristi za organizaciju i reprezentaciju toka izvršavanja, što je svakako korisno za implementaciju umjetne inteligencije u videoigrama [7]. Jednostavije rečeno, konačni automat zapravo se može zamisliti kao "mozak" ili "pamet" kod raznih neigrajućih likova, s obzirom da stanja konačnog automata predstavljaju akcije koje će neigrajući lik izvršavati [7].

Model konačnog automata može se prikazati, odnosno vizualizirati pomoću grafičkog prikaza [7]. Vizualizacija pomoću grafičkog prikaza ima svoje prednosti, jer olakšava razumijevanje toka izvršavanja stanja, tranzicija između stanja i uvjeta koji su postavljeni kako bi se tranzicije izvršile. U nastavku nalazi se jedan jednostavan primjer konačnog automata s grafičkim prikazom na slici 1.



Slika 1: Grafički prikaz jednostavnog primjera konačnog automata, prema uzoru na [7]

Na grafičkom prikazu čvorovi, to jest u ovom slučaju pravokutnici predstavljaju stanja, dok su tranzicije ilustrirane pomoću strelica [7]. Svaka strelica sadrži oznaku koja predstavlja uvjet koji mora biti zadovoljen kako bi se tranzicija u drugo stanje izvršila.

U ovom primjeru, konačni automat počinje u stanju `lutaj` koje ostaje aktivno sve dok uvjet za tranziciju u drugo stanje nije zadovoljen [7]. Kako bi se stanje promijenilo potrebno je zadovoljiti uvjet "igrač u blizini" kako bi se stanje `lutaj` promijenilo u stanje `napadaj` [7]. Također, kao što je vidljivo na slici 1 postoje i povratne veze između određenih stanja, iz razloga kako bi se stanja mogla međusobno izmjenjivati. Analogno tome, ako je trenutno aktivno stanje napadaj, a uvjet "igrač nije u blizini" je zadovoljen, izvršiti će se tranzicija natrag u stanje `lutaj`. Međutim, ukoliko je zadovoljen uvjet "igrač napada neprijatelja", u tom slučaju će se izvršiti tranzicija u jedno drugo stanje, `izbjegavaj`. Iz stanja `izbjegavaj` zatim je moguća povratna tranzicija u stanje `napadaj` ako je igrač neutralan, odnosno ne napada neprijatelja. Nadalje, tijekom stanja `izbjegavaj`, ako neprijatelj dostigne nisku razinu životnih bodova, događa se tranzicija u stanje `pronađi lijek` koje će se izvršavati tako dugo dok neprijatelj ne pronađe lijek. Kada neprijatelj pronađe lijek, stanje automata prelazi u početno stanje `lutaj`.

U videoigri *The Last of Us* svaka vještina neprijatelja zapravo predstavlja jedno stanje unutar konačnog automata [5]. Svaki tip zaraženog neprijatelja ima određena stanja poput slijeđenja, pretraživanja, potjere itd. Prijelaz ili tranzicija iz jednog u drugo stanje temelji se na specifičnim uvjetima koji moraju biti zadovoljeni kako bi se prijelaz mogao izvršiti [5]. U kontekstu videoigre neki od uvjeta koji moraju biti zadovoljeni kako bi se stanje promijenilo, zapravo se odnose na zvuk koji glumi okidač za promjenu stanja [5], a može ga proizvesti igrač ili neki drugi neigrajući lik.

Što se tiče zvuka, zvuk je u videoigri još jedan od ključnih aspekata pomoću kojega su programeri razvili neprijatelje. U videoigri dva tipa neprijatelja su slijepa (*Clicker* i *Bloater*), što

znači da se oni oslanjaju primarno na zvuk, odnosno sluh kako bi uočili i napali plijen [5]. Tako u pozadini zaraženi neprijatelji koriste senzor za zvuk kako bi percipirali okolinu i detektirali igrača koji se bori protiv njih [3].

Kada se zvuk dogodi u igri, generira se logični zvučni događaj (engl. *Logical sound event*) u okolini [3]. Zvučni događaj zatim se emitira unutar okruga, to jest radijusa koji je određen od strane dizajnera [5]. U slučaju zaraženih neprijatelja radijus se množi s podesivom vrijednošću za svaki tip neprijatelja [5]. Primjer toga jest svakako kretnja igrača [5]. Zaraženi neprijatelji su osjetljivi na razne zvukove i što se igrač brže kreće, radijus zvučnog događaja se proporcionalno povećava [5], što u konačnici neprijateljima omogućuje bržu reakciju na promjene u okolini.

Ovakav pristup omogućio je dizajnerima igre bolju kontrolu nad raznim parametrima i određivanju koji zvuk će se čuti na kojoj daljini i koji tipovi neprijatelja će ga moći čuti [3]. Također, tijekom testiranja dizajneri igre mogli su mijenjati razne parametre zvučnih događaja, bez da utječu na umjetnu inteligenciju [3].

Konačni automati su u principu vrlo jednostavni za dizajniranje, implementaciju, vizualizaciju ili otkivanje grešaka u kodu [6], što predstavlja neke od njegovih prednosti. Model konačnog automata se kroz dugi niz godina pokazao kao dobar izbor za razvoj umjetne inteligencije u modernim videoigrama [6], a vrlo vjerojatno će svoje mjesto pod suncem pronaći i u budućim videoigrama.

S druge strane, model naravno ima i neke nedostatke kao što je teže dizajniranje i kompleksnost u slučaju većih razmjera [6]. Isto tako, nakon što je razvoj automata dovršen i testiran, dosta često ne ostaje prostor za naknadna unapređenja ili promjene, što može rezultirati predvidivim ponašanjima umjetne inteligencije u videoigrama [6].

4.3.2. Stabla ponašanja (*Behavior Trees*)

Stablo ponašanja (engl. *Behavior Tree – BT*) matematički je model koji se kao i konačni automat koristi u raznim područjima poput informatike, robotike itd., a koristi se i u videoigrama za modeliranje i stvaranje kompleksnijeg, hijerarhijski strukturiranog ponašanja kod neigrajućih likova [6].

Glavna razlika između stabla ponašanja i konačnog automata jest da su stabla sastavljena od ponašanja (engl. *Behaviors*), a ne od stanja (engl. *States*) [6]. Kao i model konačnog automata, stablo ponašanja je relativno jednostavno za dizajniranje, testiranje ili otkrivanje grešaka u kodu [6]. Model stablo ponašanja postao je popularan u području razvoja videoigara nakon njegove uspješne implementacije u poznatim videoigrama *Halo 2* razvijene od strane kompanije *Microsoft Game Studios* 2004. godine i videoigre *Bioshock* od kompanije *2K Games* predstavljene 2007. godine [6].

Za razliku od modela konačnog automata ili nekih drugih modela koji se koriste za programiranje umjetne inteligencije u videoigrama, stablo ponašanja zapravo je stablo koje se sastoji od hijerarhijski strukturiranih čvorova (engl. *Nodes*) koji kontroliraju tok odlučivanja, odnosno donošenja odluka entiteta kojeg pogoni umjetna inteligencija [8]. Dakle, stablo se sastoji

od jednog korijenskog čvora (engl. *Root node*) [6], nekoliko roditeljskih čvorova (engl. *Parent nodes*) [6] i odgovarajućih podređenih čvorova djece (engl. *Child nodes*) koji predstavljaju sama ponašanja [6].

Tok odlučivanja započinje naravno u korijenskom čvoru i zatim se aktivira izvršavanje parova roditelj-dijete [6]. Dijete zatim može vratiti odgovarajuće statuse roditeljskom čvoru u unaprijed određenim vremenskim intervalima, tzv. tikovima (engl. *Ticks*) [6]. Status *run* vraća se ako je ponašanje uvijek aktivno, *success* se vraća ukoliko je ponašanje završeno, a *failure* ako je ponašanje bilo neuspješno [6]. Stabla ponašanja sastoje se od tri vrste čvorova: sekvenca (*sequence*), selektor (*selector*) i dekorator (*decorator*) [6].

Kod čvora sekvence, ako ponašanje djeteta uspije, sekvenca se nastavlja dalje i naposljetku roditeljski čvor uspijeva ako sva ponašanja djece uspiju [6]. U suprotnom, sekvenca ne uspijeva [6].

Selektor čvor je nešto kompleksniji. Postoje dvije vrste selektorskih čvorova: selektori vjerojatnosti i selektori prioriteta [6]. Što se tiče selektora vjerojatnosti, ponašanja djece odabiru se na temelju vjerojatnosti roditelj-dijete koje određuje dizajner stabla ponašanja [6]. No, ukoliko se koristi selektor prioriteta, ponašanja djece poredana su prema popisu i isprobavaju se redom [6]. Nadalje, neovisno o vrsti selektora, ako ponašanje djeteta uspije, selektor se smatra uspješnim [6]. S druge strane, ukoliko ponašanje djeteta ne uspije, odabire se sljedeće podređeno ponašanje (selektor prioriteta) ili selektor postaje neuspješan (selektor vjerojatnosti) [6].

Ostaje treća vrsta čvora, dekorator. Dekorator čvor ustvari dodaje složenost i unaprjeđuje kapacitet jednog podređenog ponašanja [6]. Primjeri dekoratora mogu uključivati broj ponavljanja ponašanja djeteta ili vremenski okvir unutar kojeg podređeno ponašanje mora biti izvršeno [6].

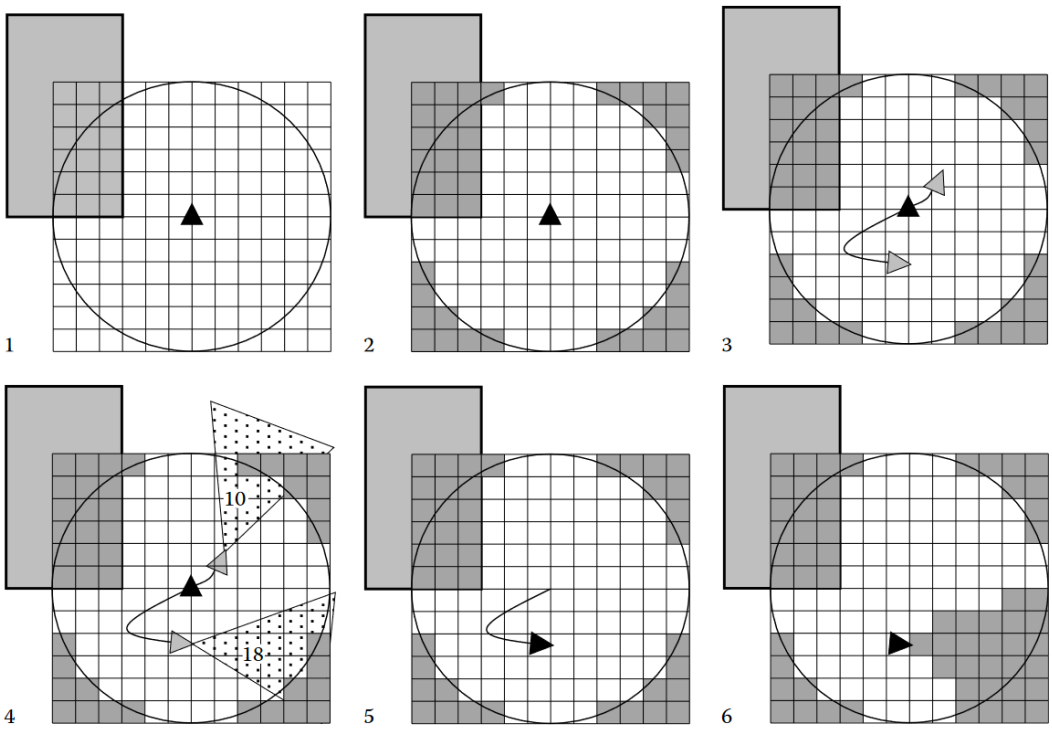
Model stablo ponašanja omogućio je razvojnim programerima da u videoigri *The Last of Us* definiraju razne sekvencijalne i paralelne zadatke koje zaraženi neprijatelji mogu izvršavati, kao što su koordinirani napadi u grupama ili složene potrage za igračem.

U usporedbi s modelom konačnog automata, model stabla ponašanja zapravo pruža slične prednosti i nedostatke za razvoj umjetne inteligencije kod neigrajućih likova. Prednosti su svakako jednostavno dizajniranje, lakša testiranja i lakše otkrivanje potencijalnih grešaka u kodu. No, isto tako nedostaci kao i kod modela konačnog automata leže u niskoj dinamičnosti i dosta slabom prostoru za naknadna unapređenja i promjene [6].

4.3.3. Infected-canvass

Vještina potrage se fokusira na posjećivanje lokacije izvora zvuka ili mjesta neke promjene koja se dogodila u okolini, no ujedno se aktivira i specijalno ponašanje *infected-canvass* [3] kako bi zaraženi pretražio okolinu, pronašao izvor zvuka ili igrača koji može proizvesti zvuk koracima ili brzim kretnjama. S obzirom da su zaraženi neprijatelji u videoigri zapravo stvorenja bez razuma, dizajnirani su tako da se okolinom kreću instinktivno i nepredvidivo [3]. Kako bi razvojni programeri postigli takve kretnje zaraženih neprijatelja okolinom, kreirali su upravo

infected-canvass. Radi se o specijalnom ponašanju koje je specifično za sve tipove zaraženih neprijatelja koje im omogućuje brze i nepredvidive okrete te opservaciju okoline [5]. Ponašanje je kao i svako drugo vezano uz skup animacija koje svaki tip neprijatelja posjeduje, a koristi animacije koje definiraju kretnje zaraženih okolinom [5]. Kako funkcionira *infected-canvass* prikazano je pomoću šest koraka na slici 2.



Slika 2: Biranje animacije u *infected-canvass* ponašanju blizu prepreka [3, str. 414]

Prvi korak je logička mreža postavljena preko područja koje je pokriveno radijusom pretraživanja i ujedno je centrirana u odnosu na zaraženog [3]. Korak dva odnosi se na prepreke i područje izvan radijusa pretraživanja koji su označeni kao pretraženi, što znači da tako ostaju ćelije koje još neprijatelj treba provjeriti [3]. Zatim slijedi korak tri gdje pozvana vještina pruža ponašanju skup animacija koje su dostupne neprijatelju, dok programeri određuju lokaciju i orijentaciju koju će neprijatelj imati na kraju svake animacije [3]. Nadalje, četvrti korak ilustrira kako se broj nevidljivih ćelija u polju odnosno sektoru ispred lika broji za svaku animaciju. Sektor ne mora nužno odgovarati području pokrivnom stvarnim osjetilima [3]. Također, veći sektor pruža veću pokrivenost za bržu pretragu, dok manji sektor pruža detaljniju i temeljitiju pretragu [3].

Peti korak na ilustraciji vezan je uz animacije. Što se tiče animacija, animacije koje rezultiraju većim brojem nevidljivih ćelija su poželjnije, a animacije koje su se prethodno izvršile su manje poželjne [3]. Dakle, izvršava se najpoželjnija animacija koja pomiče neprijatelja [3]. Kada se animacija izvrši, ćelije u senzornom sektoru označene su kao "pregledane" što je prikazano na ilustraciji kao šesti korak i proces se ponavlja na novoj lokaciji od trećeg koraka [3].

Specijalno ponašanje *infected-canvass* nikada ne odabire eksplicitno smjer u kojem će

se neprijatelj kretati, već se temelji isključivo na dostupnim animacijama [3]. Takav pristup omogućio je jednostavniji kod i animatorima nije bilo ograničeno stvaranje animacija u fiksnom broju smjerova, što im je ujedno pružilo veću slobodu za stvaranje raznolikih izražajnih izvedbi [3]. Također, dizajneri su imali kontrolu nad radijusom pretraživanja i određivanjem koliko dugo će zaraženi neprijatelj pretraživati okolinu [3].

Vezano uz raznolikost, ovo specijalno ponašanje uključuje skupove animacija koje pomiču neprijatelja na blizinu ili na daljinu. U manjem području neprijatelj koristi skup animacija za blizinu poput brzih okreta kako bi pogledao u svim smjerovima na vrlo nepredvidljiv i nasumičan način [3]. U većim područjima koristi se skup animacija za daljinu koje se dodaju u kombinaciju, što rezultira kretanjem neprijatelja preko područja pretraživanja u kraćim i dužim naletima [3].

Tehnika *infected-canvass* pomogla je razvojnim programerima prilikom kreiranja nepredvidivih, ali opet organskih kretnji zaraženih neprijatelja okolinom [3]. Specijalno ponašanje *infected-canvass* iskorišteno je i za još nekoliko drugih vještina kod zaraženih neprijatelja, pored vještine pretraživanja okoline [3].

4.4. Zaključno o zaraženim neprijateljima

Zaraženi neprijatelji su u videoigri *The Last of Us* zapravo samo jedan manji dio cijelog svijeta unutar videoigre. Kroz prethodna poglavlja potrudio sam se analizirati i istražiti na koji način su razvojni programeri i ostatak razvojnog tima pristupili razvoju umjetne inteligencije koja stoji iza zaraženih neprijatelja i zašto su ih razvili na način na koji su ih razvili.

Zaraženi neprijatelji su u videoigri neigrajući likovi, odnosno neprijatelji protiv kojih se igrač mora boriti. Oni nisu nastali preko noći, već su rezultat dizajna, umjetnosti, animacije, zvuka i programiranja [3], što znači da su metode umjetne inteligencije koje stoje iza njihovog ponašanja samo jedan manji dio te cjeline koji je pomogao njihovu razvoju.

Razvoj zaraženih neprijatelja svakako je razvojnom timu predstavljao dosta zahtjevan posao, jer su nekako morali neprijatelje stopiti s pričom, okolinom i mnogim drugim faktorima [3]. Neprijatelji su na kraju ispali dostojni protivnici prema kojima igrač mora pristupiti pametno i s dozom poštovanja kako bi ih nadmudrio [3].

Vještine i ponašanja na kojima se temelji sustav umjetne inteligencije koji je primijenjen kod zaraženih neprijatelja, isto tako su samo jedan manji dio kompleksnog sustava umjetne inteligencije koji stoji iza zaraženih, no ključni su za donošenje odluka koje su potrebne da zaraženi neprijatelj reagira uvjerljivo i organski na poteze i kretnje igrača [3]. Održavanje koda jednostavnim i korištenje modularne arhitekture bilo je ključno kako navodi Botta [3]. Korištenje promjenjivih varijabli koje su bile neovisne o tipu zaraženog neprijatelja omogućile su da svi tipovi zaraženih dijele iste vještine i ponašanja [3], ali opet na okolinu reagiraju unikatno i interesantno, pružajući tako igraču pravi izazov.

5. Implementacija odabranih modela

Nakon odrađenog teoretskog dijela ovog dokumenta, vrijeme je i za izradu praktičnog kako bi se prikazalo na koji način je danas moguće uz adekvatne i besplatne alate izraditi relativno realistično i zanimljivo iskustvo igranja.

Odlučio sam kako ću za praktični dio završnog rada razviti jednu jednostavnu (engl. *Demo level*) razinu u alatu Unreal Engine 5. Razina će se sastojati od glavnog lika kojime će moći upravljati igrač, a također unutar razine biti će smještena dva tipa zaraženih neprijatelja kojima će upravljati umjetna inteligencija. Praktični dio ovog rada poslužiti će ustvari kao zanimljiva kraća prezentacija osnovne mehanike kretanja i mehanike borbe (*Stealth*) protiv zaraženih neprijatelja koja se koristi unutar stvarne videoigre.

5.1. Unreal Engine 5

Unreal Engine 5 je jedna vrlo moćna mašina, to jest alat koji se koristi za razvoj modernih videoigara. Polagano alat postaje sve poznatiji i raširenij te ujedno sve više razvojnih programera za videoigre odabire Unreal Engine kao platformu za razvoj. Unreal Engine 5 donosi puno inovacija te jedan vrlo zanimljiv i moderan pristup razvoju videoigara. Neke od naprednih tehnologija koje nudi Unreal Engine 5 su *Nanite*, *Lumen*, *MetaHuman Creator* itd. Pomoću tih inovativnih tehnologija i sa ponešto znanja iz područja razvoja videoigara moguće je kreirati vrlo impresivne kreacije.

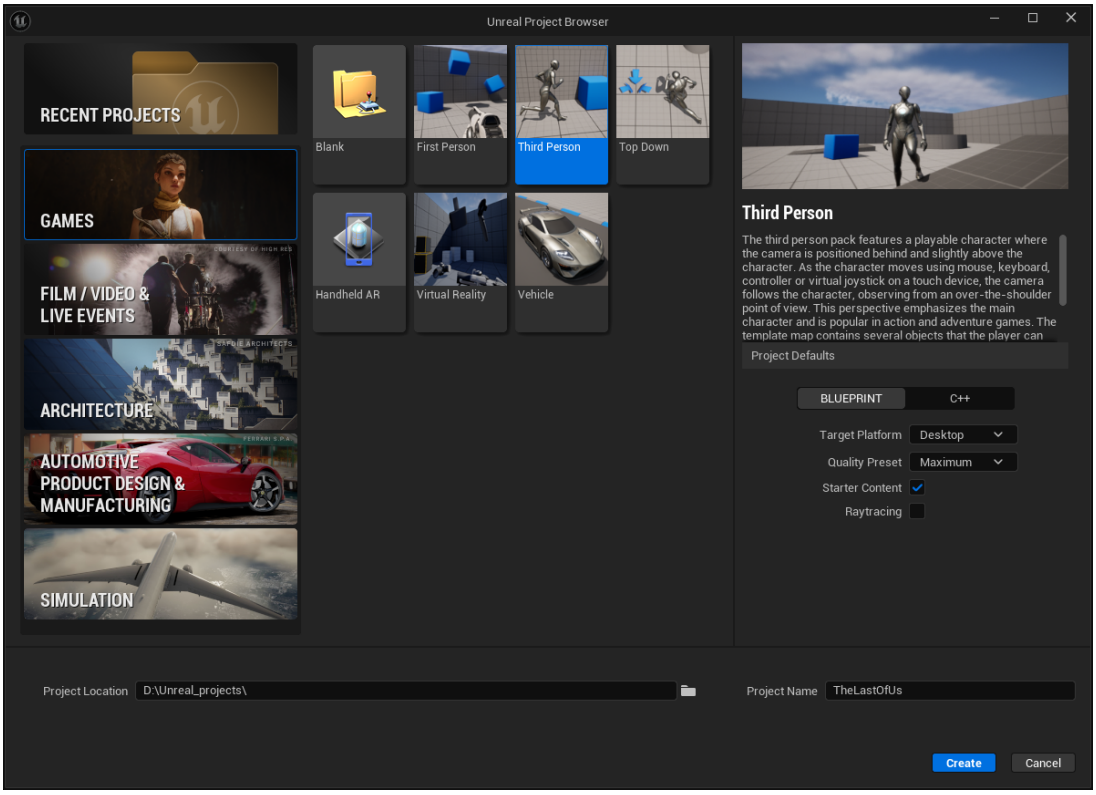
Valja spomenuti kako se Unreal Engine ne koristi samo za razvoj videoigara, već i za razvoj različitih vrsta vizualnih i interaktivnih multimedijskih sadržaja, uključujući simulacije, filmove i sl. Sve te napredne tehnologije koje su ugrađene u samu mašinu i mnoštvo funkcija koje mašina posjeduje, omogućavaju kreatorima da stvore sadržaj koji je vizualno zapanjujuć i tehnički impresivan.

Alat Unreal Engine 5 za programiranje logike videoigre koristi programski jezik C++. Prilikom kreiranja projekta alat nudi odabir između dva načina koji su vezani za kreiranje programske logike u projektu. Jedan način jest korištenjem nacrta (engl. *Blueprints*), dok je drugi način klasičnim pisanjem C++ koda. Unreal Engine upravo je po tome i specifičan, jer korištenjem nacrta olakšava i ubrzava izradu programske logike koja će se izvršavati u pozadini videoigre.

Tehnologijom nacrta Unreal Engine je unio inovaciju u područje razvoja videoigara, s obzirom da se korištenjem nacrta mogu postići vrlo impresivni rezultati, a ujedno nije potrebno prethodno poznavati programiranje, odnosno klasični način izrade programske logike. Naravno, poznavanje koncepata programiranja i programskog koda definitivno olakšava cijeli proces. Do sada se još nisam susreo s takvim načinom izrade programske logike te sam iz tog razloga odlučio kako ću kreirati projekt upravo sa opcijom korištenja nacrta, a siguran sam kako ću odabirom te opcije i nešto novo naučiti prilikom izrade ovog projekta.

5.2. Stvaranje projekta

Kako bi se započelo sa izradom praktičnog dijela, kreiran je projekt u alatu Unreal Engine 5. Prilikom kreiranja projekta potrebno je odabrati ciljanu platformu i predložak na temelju kojega će teći daljnji razvoj projekta. Tako je za predložak projekta odabrana opcija videoigre u trećem licu (engl. *Third Person*), s obzirom da je videoigra *The Last of Us* izrađena isto tako u trećem licu. Postupak stvaranja samog projekta vidljiv je na slici 3.



Slika 3: Prikaz početnog zaslona prilikom stvaranja projekta u alatu Unreal Engine 5

Nakon što se projekt uspješno kreira, otvara se novi pogled, to jest scena unutar koje su postavljene neke osnovne postavke. Također, scena se sastoji i od osnovne platforme s nekoliko objekata kako bi se moglo što lakše i brže krenuti sa prototipiranjem videoigre.

5.3. Izrada likova

Prvi korak prototipiranja bio je izrada likova koji će se biti na konačnoj sceni, preciznije izrada glavnog lika kojime će upravljati igrač. Likovi koji se pojavljuju unutar videoigre u principu su animirani trodimenzionalni modeli. Na internetu na platformi *Rig Models* uspio sam pronaći nekoliko besplatnih trodimenzionalnih modela koji će mi jako dobro poslužiti za izradu planirane scene.

S platforme *Rig Models* preuzet je jedan model koji će poslužiti za implementaciju glavnog lika i još dva modela koja će predstavljati dva tipa zaraženih neprijatelja koji se ujedno

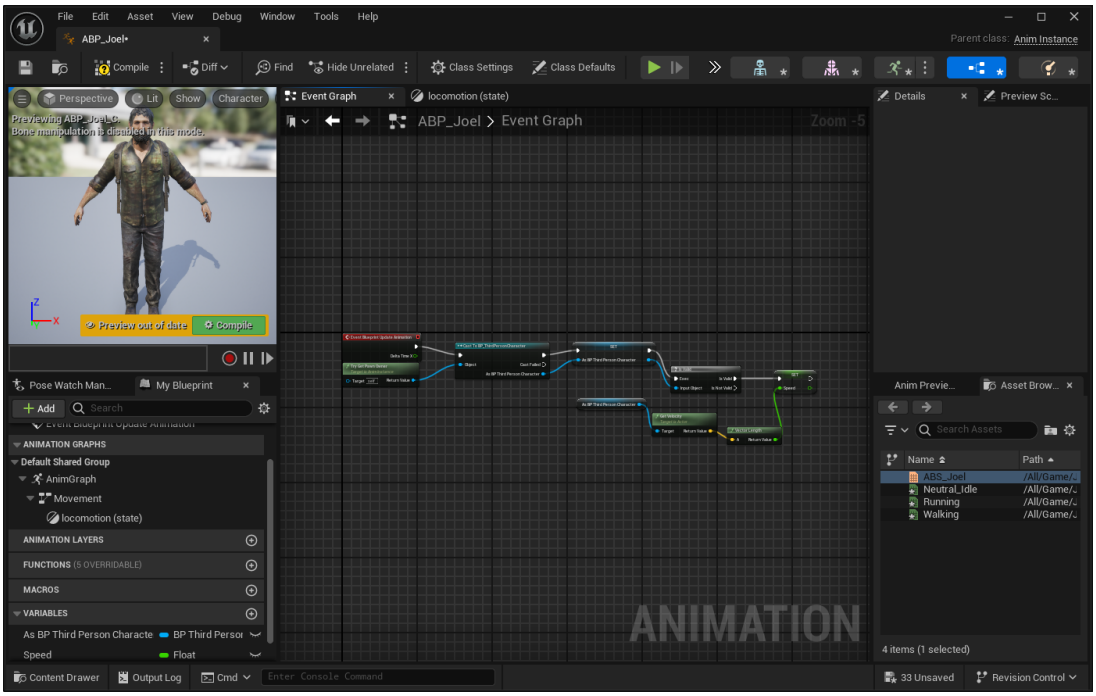
najčešće pojavljuju i u stvarnoj videoigri. Iskorišten je alat *Blender* kako bi se preuzeti modeli dodatno podesili, provjerilo prikazuju li se ispravno teksture i pripremili za uvoz u Unreal Engine.

Preuzete modele biti će potrebno animirati kako bi se pokreti modela mogli uskladiti s kretnjama likova na sceni. Postoji nekoliko besplatnih platformi na internetu gdje je moguće pronaći već unaprijed pripremljene animacije, koje je zatim potrebno povezati s odabranim modelom. Nakon malo duljeg istraživanja odlučeno je kako će se animacije za likove preuzeti s platforme *Mixamo* iza koje stoji tvrtka *Adobe*, gdje je uspješno pronađeno dosta raznolikih animacija koje će se dobro uklopiti u projekt. Nakon precizno odabranog seta animacija za svaki pojedini preuzeti model, animacije su uvezene u Unreal Engine kako bi se moglo krenuti sa izradom likova.

5.3.1. Izrada glavnog lika

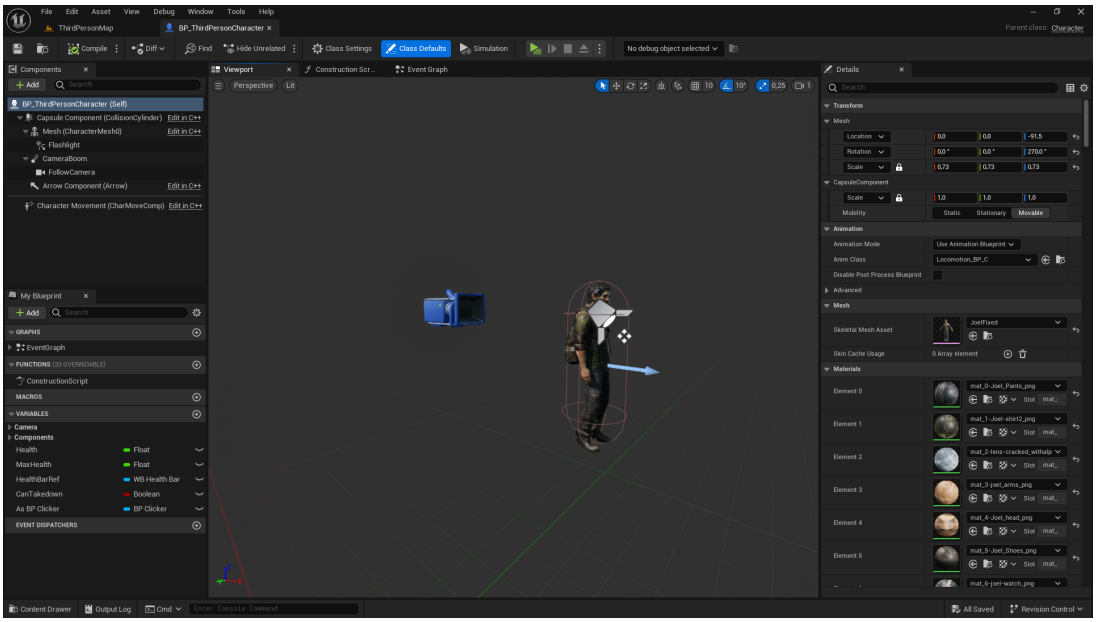
Kao što je već ranije spomenuto, glavnim likom će moći upravljati igrač. Kako bi se to postiglo na scenu se mora ubaciti objekt, odnosno lik koji će primati ulaze preko tipkovnice kako bi se on mogao kretati. Kreiranjem projekta iz odabranog predloška stvara se i nekoliko inicijalnih objekata koji se mogu dodatno razraditi i prilagoditi za konačnu scenu. Unreal Engine koristi tehniku nacrta (engl. *Blueprints*) koji olakšavaju i ubrzavaju sami tijek razvoja i prototipiranja. Unutar tih nacrta moguće je podesiti koliziju, razne varijable, postavke kretanja itd. za lik koji se razvija.

Nakon podešavanja osnovnih kretnji glavnog lika, bilo je vrijeme i za ugradnju animacija u odabrani model, kako bi se kretnje preuzetog modela podudarale sa kretnjama lika koji je smješten na scenu. Animacije koje će specifični model izvršavati u alatu Unreal Engine implementiraju se pomoću animacijskog nacrta (engl. *Animation Blueprint*), gdje se ovisno o postavljenim parametrima izvršavaju određene animacije na trodimenzionalnom modelu. Slika 4 prikazuje postupak izrade animacijskog nacrta za glavnog lika.



Slika 4: Prikaz izrade i podešavanja animacijskog nacrta glavnog lika

Unutar nacrta glavnog lika ugrađeno je još nekoliko dodatnih komponenti, a jedna komponenta je i sama kamera. S obzirom da je kamera kao komponenta ugrađena unutar nacrta glavnog lika, kamera će automatski pratiti lika odnosno igrača na sceni. Kamera je u nacrtu tako pozicionirana iza samog lika, tako da će ustvari pogled koji ima igrač biti u prostor koji se nalazi ispred glavnog lika, što je i slučaj kod videoigara koje su izrađene u trećem licu. Slika 5 prikazuje nacrt glavnog lika i poziciju same kamere. Kamera kao komponenta u alatu Unreal Engine nakon kreiranja projekta već posjeduje neke osnovne postavke koje su i više nego dovoljne za konačnu scenu, no odlučeno je kako će se dodatno podesiti parametri same kamere, kako bi se što više približilo izgledu i stilu stvarne videoigre.



Slika 5: Prikaz nacrta i osnovnih komponenti glavnog lika

Nakon provedenih testova i isprobavanja raznih parametara, postignut je željeni rezultat. Konačni pogled prikazan na slici 6 koji je bio zadovoljavajuć jest preko ramena glavnog lika u prostor koji se nalazi ispred samoga lika kako bi igrač mogao vidjeti gdje se potrebno kretati. Isto tako, pogled je moguće rotirati oko osi glavnog lika, što omogućava igraču istraživanje i pregledavanje okoline unutar koje se glavni lik nalazi u određenom trenutku.

Slika 6: Konačni izgled pogleda iz trećeg lica na glavnog lika i okolinu

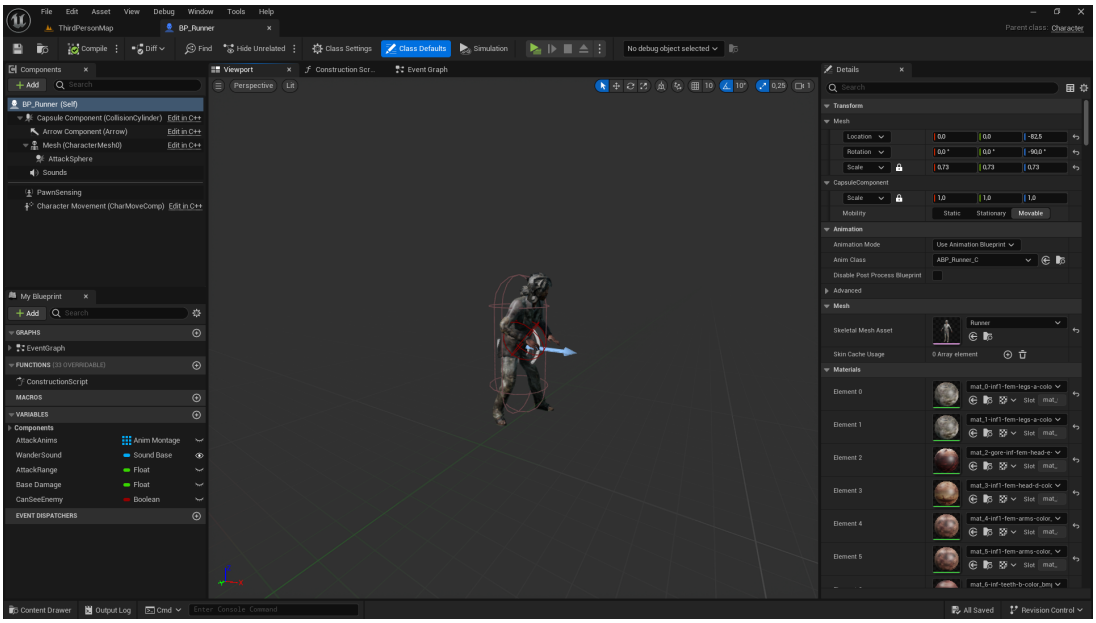
5.3.2. Izrada neprijatelja

Kada je završena izrada i testiranje glavnog lika, započeta je i izrada likova koji će predstavljati neprijatelje protiv kojih se igrač mora boriti. Odlučeno je kako će se izraditi dva

neprijateljska lika, odnosno dvije vrste neprijatelja koje se ujedno i najčešće pojavljuju unutar stvarne videoigre. Jedan neprijateljski lik će predstavljati tip neprijatelja *Runner*, a drugi lik tip neprijatelja *Clicker*. Neprijatelji će kao i glavni lik imati određene parametre poput brzine kretanja itd., no za razliku od glavnog lika, njima će upravljati umjetna inteligencija.

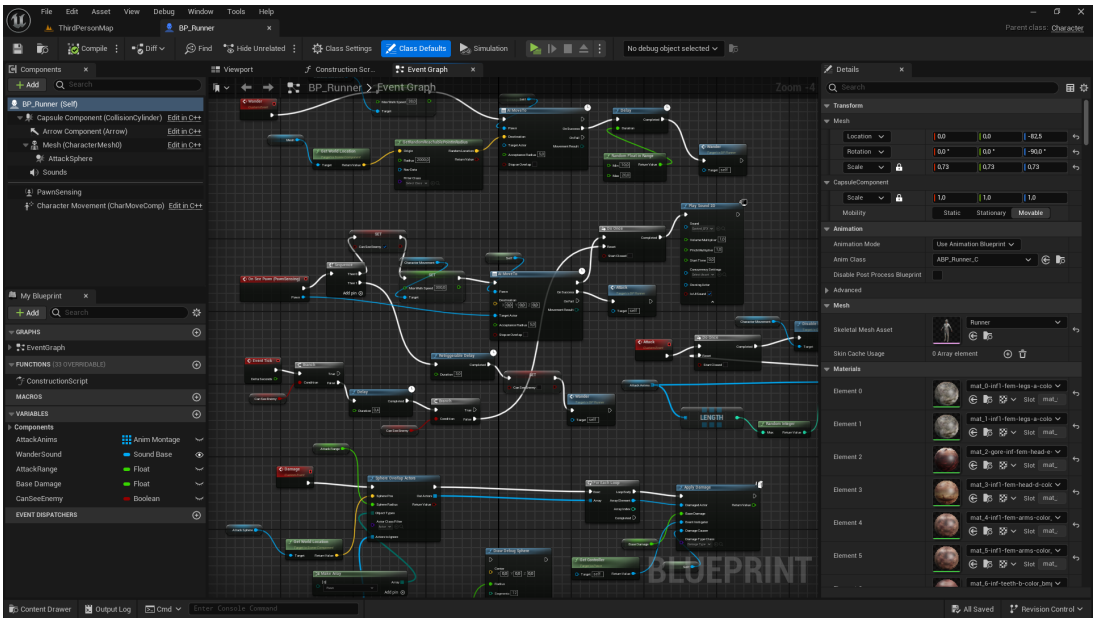
Prije izrade logike koja će upravljati neprijateljima, radilo se na ugradnji animacija u modele, kako za vrijeme izvršavanja videoigre oni ne bi bili statični, već da se animacije kretnji modela izvršavaju ispravno u odnosu na kretnje lika. Izradu tranzicija između animacija u alatu Unreal Engine omogućuje tehnologija *Animation Blendspace* koja omogućava stapanje i izradu glatkih prijelaza između animacija. Zatim je još bilo potrebno izraditi nacrt animacija (*Animation Blueprint*), to jest logiku koja će se pobrinuti za adekvatno izvršavanje animacija kod lika. Ovaj postupak ugradnje animacija za likove neprijatelja u principu je vrlo sličan postupku koji je proveden i kod glavnog lika.

Oba neprijateljska lika implementirana su kao nacrt *Blueprint class*. Na slici 7 prikazan je kreirani nacrt za izradu neprijatelja *Runner*. Nacrt neprijatelja kao i nacrt od glavnog lika ima ugrađene neke osnovne komponente poput kapsule za koliziju, *Mesh* komponente unutar koje se učitava određeni trodimenzionalni model koji se želi koristiti za lika itd.



Slika 7: Prikaz nacrta i osnovnih komponenti koje posjeduju neprijateljski likovi

Umjetna inteligencija koja stoji iza neigrajućih likova u alatu Unreal Engine razvija se preko pogleda *Event graph* koji je smješten unutar nacrta lika (*Blueprint class*). *Event graph* pogled nalazi se prikazan na slici 8 ispod. Ono što je specifično za alat Unreal Engine jest da se programiranje umjetne inteligencije može izvršavati dodavanjem čvorova u nacrt i njihovim međusobnim povezivanjem, što zapravo zamjenjuje klasično programiranje pisanjem koda. U pozadini Unreal Engine za programiranje koristi programski jezik C++, a sam nacrt zapravo je vizualna prezentacija logike koja se izvršava. Tijekom kreiranja projekta alat Unreal Engine nudi odabir načina razvoja korištenjem nacrta ili klasičnim pisanjem C++ koda.

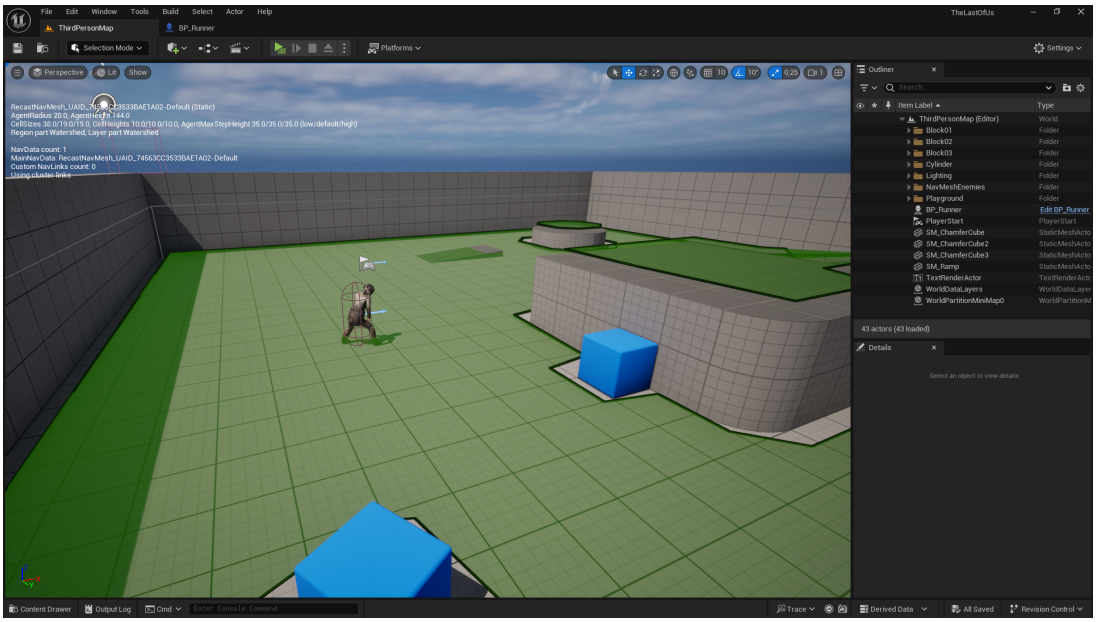


Slika 8: Izrada umjetne inteligencije koja će kontrolirati neigrajućeg lika

Najvažnija komponenta koja je dodana neprijateljima jest *Pawn Sensing*. U alatu Unreal Engine ta komponenta se koristi kako bi se neigrajućim likovima omogućila detekcija drugih likova, u ovom slučaju neprijatelj će moći detektirati glavnog lika, to jest igrača koji se kreće okolinom. Unutar te komponente mogu se podesiti razni parametri poput širine vidnog polja, interval ažuriranja vidnog polja, što utječe i na vrijeme koje je potrebno da neprijatelj detektira igrača i sl. Nakon nekoliko testova i isprobavanja postignut je dobar balans između parametara koji stvaraju zanimljivo i napeto iskustvo igranja.

Još jedan vrlo važan objekt unutar alata Unreal Engine koji je postavljen na scenu, a vezan je za neigrajuće likove jest *NavMesh Bounds Volume*. Postavljanje objekta na scenu prikazano je ispod na slici 9. Objekt *NavMesh Bounds Volume* omogućava definiranje područja unutar kojeg se neigrajući lik kojim upravlja umjetna inteligencija može kretati.

Objekt *NavMesh Bounds Volume* zapravo je nešto slično specijalnom ponašanju *infected-canvass* koje se koristi u stvarnoj videoigri *The Last of Us*, a isto tako je zaslužno za definiranje područja kojim se neigrajući lik može kretati. Naravno, *Naughty Dog* je razvio nešto puno sofisticiranije, no objekt *NavMesh Bounds Volume* će svakako vrlo dobro poslužiti za implementaciju osnovnih kretnji neigrajućih likova okolinom unutar ove *demo* razine.



Slika 9: Dodavanje objekta *NavMesh Bounds Volume* na scenu

5.4. Izrada mehanike borbe

Nakon uspješne izrade likova, bilo je vrijeme i za implementaciju interakcije, odnosno mogućnosti borbe između glavnog lika i neprijateljskih, odnosno neigrajućih likova.

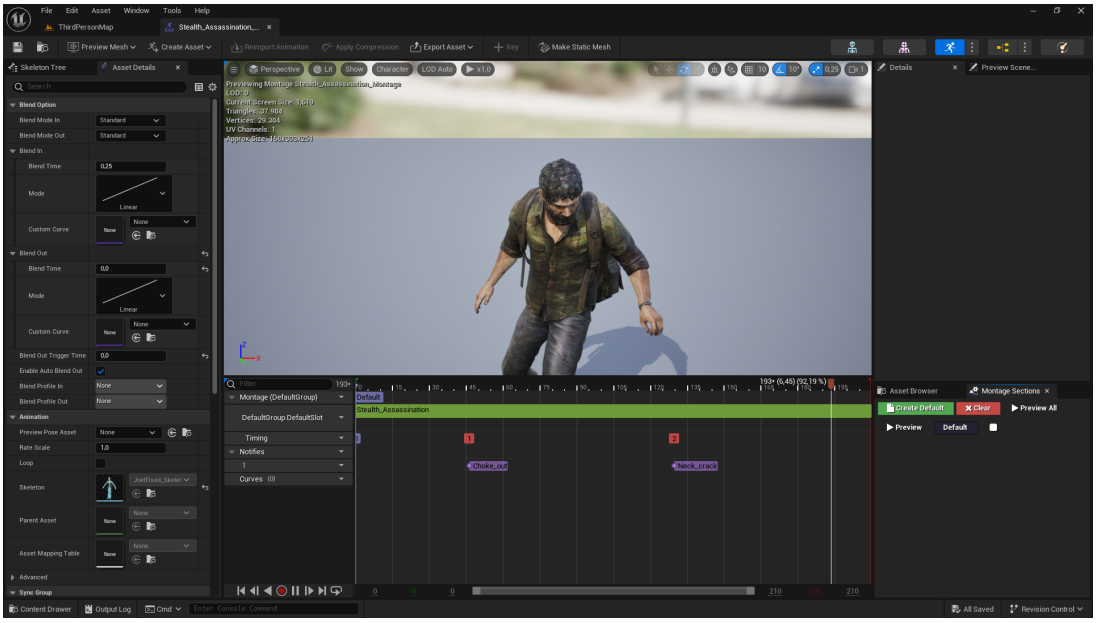
U videoigri *The Last of Us* postoji nekoliko mehanika borbe i načina na koje se igrač može boriti protiv neprijatelja. Jedna od tih mehanika borbe jest napad na neprijatelja šuljanjem (engl. *Stealth*), što je i odlučeno da će se imlementirati kao mehanika borbe unutar ovog praktičnog dijela rada, zbog veće napetosti i zanimljivosti.

Što se tiče neprijatelja, pored mogućnosti lutanja okolinom, neprijateljima je razvijena i mogućnost napada na glavnog lika, nakon što glavni lik bude detektiran pomoću prethodno dodane komponente *Pawn Sensing*. Kako bi se implementirala mogućnost napada, neprijatelju je dodana jedna nova komponenta koja je prezentirana kao manja sfera koja je ugrađena na ruku trodimenzionalnog modela. Dodana komponenta prikazana je crvenom bojom i vidljiva je na slici 7. Sfera ustvari tako predstavlja jedan manji krug pomoću kojega neprijatelj može nanijeti štetu igraču, to jest oduzeti igraču životne bodove. Analogno tome, glavnom liku je također dodana jedna sfera koja će predstavljati zamišljeni krug oko glavnog lika unutar kojeg on može primiti udarac od strane neigrajućeg lika.

Nakon uspješno podešenih komponenti izrađena je i programska logika koja će se izvršavati u pozadini prilikom borbe između likova.

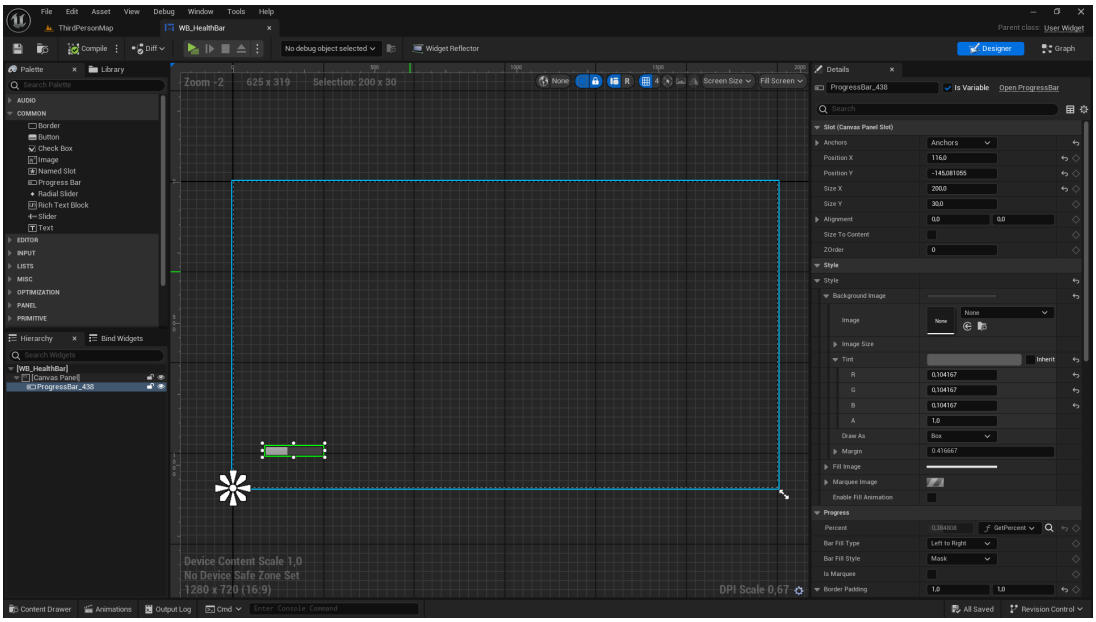
Unreal Engine omogućuje stvaranje tzv. animacijske montaže (engl. *Animation Montage*). Animacijska montaža se može stvoriti na temelju prethodno uvezenih animacija koje su specifične za određeni model. U svrhu implementacije mehanike borbe izrađeno je nekoliko animacijskih montaža za svaki pojedini lik. Ukratko objašnjeno, animacijska montaža služi za spajanje i kontrolu izvršavanja animacija unutar nacrta, odnosno programske logike samoga

lika. Primjerice, kada neprijatelj dostigne glavnog lika i pokuša izvršiti napad, u slučaju ove *demo* razine to je zamah neprijatelja rukom prema glavnom liku, izvršava se animacijska montaža koja osigurava da se animacije modela izvršavaju u skladu sa logikom u pozadini lika. Isto tako, mehanika napada neprijatelja šuljanjem implementirana je pomoću animacijske montaže. Prikaz jedne takve animacijske montaže nalazi se na slici 10.



Slika 10: Prikaz kreirane animacijske montaže

Kako bi se još dodatno produbilo iskustvo igranja ove *demo* razine, razvijena je i jednostavnija mehanika životnih bodova. Logika iza životnih bodova nije previše komplicirana, namiještena je tako da svaki udarac neprijateljeve šake koji se zatekne unutar užeg kruga glavnog lika, igraču oduzme životne bodove. Za realističnije iskustvo podešeno je tako da tri udarca neprijatelja mogu ubiti igrača ukoliko ih igrač ne izbjegne, nakon čega se igrača vraća na početak razine (engl. *Respawn*). Prikazivanje životnih bodova na ekranu kako bi igrač u svakom trenutku imao uvid u stanje životnih bodova, implementirano je pomoću nacrta korisničkog sučelja *Widget Blueprint*, kao što je i prikazano na slici 11.



Slika 11: Izrada grafičke prezentacije životnih bodova

6. Zaključak

Teoretska analiza modela konačni automat i modela stablo ponašanja koji su primijenjeni za razvoj umjetne inteligencije u videoigri *The Last of Us* pokazala je da oba modela nude značajne prednosti u izradi umjetne inteligencije za moderne videoigre, ali također da dolaze s određenim ograničenjima. Konačni automat, unatoč svojoj jednostavnosti i pouzdanosti, pokazuje nedostatke kada je riječ o izradi kompleksnijih sustava, posebno u kontekstu velikih projekata gdje može doći do problema s predvidljivošću ponašanja umjetne inteligencije. Analogno tome, stabla ponašanja pružaju slične prednosti, ali se suočavaju s ograničenjima u dinamičnosti i sposobnosti za naknadna unapređenja i promjene.

Analizom metode umjetne inteligencije primijenjene u videoigri *The Last of Us*, posebno kod zaraženih neprijatelja, jasno je da su ti likovi rezultat detaljnog i pažljivog razvoja. Umjetna inteligencija nije samo tehnička komponenta, već se organski povezuje s pričom, okolinom i drugim dizajnerskim elementima. Korištenje modularnog pristupa omogućilo je razvojnom timu jednostavno održavanje koda i prilagodljivost različitim tipovima neprijatelja, čineći njihove reakcije vjerodostojnima i izazovnima za igrača.

Iako su metode umjetne inteligencije samo jedan dio cijelog procesa razvoja zaraženih neprijatelja, one su ključne za stvaranje uvjerljivih, dinamičnih i izazovnih protivnika koji igraču pružaju pravi osjećaj napetosti i neizvjesnosti. Razvojni tim kompanije *Naughty Dog* uspješno je implementirao prethodno analizirane metode umjetne inteligencije kako bi igračima omogućio jedinstveno iskustvo unutar svijeta videoigre.

Unreal Engine 5 je vrlo moćan alat za razvoj modernih videoigara. Opisom i izradom praktičnog dijela ovog rada samo je zagrebena površina svih mogućnosti i funkcionalnosti koje alat nudi.

Alat Unreal Engine 5 dobro je osmišljen i razvijen od strane kompanije *Epic Games*. Alat je prvenstveno besplatan do određene razine zarade, što ga čini dostupnim svima, a potrebna je samo internetska veza. Doduše, alat zahtijeva dosta računalnih resursa, što je zapravo jedina mana ovog moćnog sustava. Unreal Engine 5, osim za razvoj videoigara, može se koristiti i u druge svrhe, poput izrade filmova, simulacija i mnogih drugih interaktivnih multimedijskih sadržaja. Zbog svojih širokih mogućnosti, sustav privlači širok spektar korisnika koji žele razvijati svoje kreacije upravo u tom alatu.

Nadalje, Unreal Engine 5 ima veliku i aktivnu zajednicu razvojnih programera koji dijele resurse i spremni su pomoći početnicima koji se tek upoznavaju sa samom platformom i mogućnostima alata. Također, kompanija *Epic Games* pruža opsežnu dokumentaciju i redovita ažuriranja alata, kao i besplatne resurse koji su namijenjeni za učenje i inspiraciju.

Izrada praktičnog dijela završnog rada pružila mi je izazov, s obzirom na to da još nisam koristio alat Unreal Engine 5, ali mogu reći kako sam zadovoljan konačnom kreacijom i kako mi je drago što sam odabrao upravo taj alat za razvoj praktičnog dijela. Imao sam priliku naučiti nešto novo za što sam siguran da će mi pomoći u daljnjem osobnom razvoju i razvoju buduće karijere.

Popis literature

[1] „The Last of Us," The Last of Us Wiki. [Na internetu]. (), adresa: https://thelastofus.fandom.com/wiki/The_Last_of_Us (pogledano 12. 7. 2024.).

[2] „Infected," The Last of Us Wiki. [Na internetu]. (), adresa: https://thelastofus.fandom.com/wiki/Infected (pogledano 14. 7. 2024.).

[3] M. Botta, „Infected AI in The Last of Us," *Game AI Pro Volume 2*, 1. izdanje, sv. 2, [Na internetu], A K Peters/CRC Press, 2015., str. 407–417. adresa: http://www.gameaipro.com/ (pogledano 20. 3. 2024.).

[4] „Konačni automat," Wikiwand. [Na internetu]. (), adresa: https://www.wikiwand.com/hr/Kona%C4%8Dni_automat (pogledano 19. 7. 2024.).

[5] T. Thompson. „Endure and Survive: the AI of The Last of Us." [Blog post]. (17. 6. 2020.), adresa: https://www.gamedeveloper.com/design/endure-and-survive-the-ai-of-the-last-of-us (pogledano 22. 7. 2024.).

[6] G. N. Yannakakis i J. Togelius, *Artificial Intelligence and Games*, 1. izdanje. Springer International Publishing, 2018.

[7] F. Bevilacqua. „Finite-state machines: Theory and implementation," Code Envato Tuts+. [Na internetu]. (24. 10. 2013.), adresa: https://code.tutsplus.com/finite-state-machines-theory-and-implementation--gamedev-11867t (pogledano 25. 8. 2024.).

[8] C. Simpson. „Behavior trees for AI: How they work." [Blog post]. (18. 7. 2014.), adresa: https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work (pogledano 23. 7. 2024.).

Popis slika

Popis tablica

Prilozi

1. Prilog 1

1.1. Preuzeti trodimenzionalni modeli

- *Rig Models*, "Joel 3D Model". [Na internetu], adresa: https://rigmodels.com/model.php?view=Joel-3d-model__WXL7HFRJ6939KC6MFQ0H23PNX
- *Rig Models*, "Zombie 3D Model". [Na internetu], adresa: https://rigmodels.com/model.php?view=Zombie-3d-model__603B9JX80N3R30TG5RKPYHM5I
- *Rig Models*, "Infected Zombie 3D Model". [Na internetu], adresa: https://rigmodels.com/model.php?view=Infected_Zombie-3d-model__PF1LRXWDSLHG859K8W5LMUL4F

1.2. Resursi za izradu zvukova

- "The Last of Us: Sounds of Infected [Runners, Stalkers, Clickers, Bloaters]". *Youtube*, adresa: https://www.youtube.com/watch?v=xhnf9sU146k&t=3101s
- "The Last of Us Part 2 - Joel Miller Voice Sounds". *Youtube*, adresa: https://www.youtube.com/watch?v=DLOdJ7NKHyI&t=1794s
- "The Last of Us: Death Sound Effect". *Youtube*, adresa: https://www.youtube.com/watch?v=ObMUZyq5pBI
- "THE LAST OF US | Mix Ambient Music". *Youtube*, adresa: https://www.youtube.com/watch?v=THlUmzjlvKg&t=7s