SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1339

ANALIZA MREŽNOG PROMETA PRI INTERAKCIJI KORISNIKA S RAČUNALNO GENERIRANIM LIKOVIMA U 3D VIDEOIGRI

Dominik Maček

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1339

ANALIZA MREŽNOG PROMETA PRI INTERAKCIJI KORISNIKA S RAČUNALNO GENERIRANIM LIKOVIMA U 3D VIDEOIGRI

Dominik Maček

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 1339

Pristupnik:	**Dominik Maček (0036494293)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Telekomunikacije i informatika
Mentor:	prof. dr. sc. Maja Matijašević

Zadatak: **Analiza mrežnog prometa pri interakciji korisnika s računalno generiranim likovima u 3D videoigri**

Opis zadatka:

Vaš zadatak je proširiti raniju inačicu 3D interaktivne računalne igre "Wizard Wars" razvijene primjenom programskog okvira Unity unutar diplomskog projekta. Proširenje treba obuhvatiti uvođenje računalno-generiranih neprijateljskih likova, pri čemu treba osmisliti i razraditi njihovo ponašanje te ga programski izvesti primjenom neke od postojećih, ili vlastite strategije. Osmislite i u lokalnoj mreži provedite eksperiment u kojem ćete snimiti mrežni promet prilikom interakcije igrača i računalno-generiranog lika. Ako bude izvedivo, ponovite eksperiment s povezivanjem preko mreže širokog područja putem virtualne privatne mreže. Analizirajte odabrane značajke mrežnog prometa i ocijenite iskustvenu kvalitetu igranja uz različite uvjete u mreži. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 30. lipnja 2020.

Za Micka, i njeno beskonačno strpljenje.

Sadržaj

Uvod

Višekorisničke videoigre definiraju se kao videoigre u kojima može istovremeno sudjelovati više od jednog igrača, spojenih bilo lokalno ili preko Interneta. U ovom radu, kao temelj koristi se višekorisnička videoigra „Wizars wars“, prethodno izrađena u okviru kolegija Diplomski projekt. Korištena igra je zatim nadograđena dodavanjem neigračkih likova, implementiranih korištenjem umjetne inteligencije. Zatim je igra podvrgnuta je eksperimentima u virtualnoj privatnoj mreži kreiranoj pomoću LogMeIn Hamachi-ja. Predmet eksperimenata je analiza mrežnog prometa snimljenog za vrijeme igranja dva korisnika, pritom mijenjajući način povezivanja sa svojim usmjeriteljima, bilo bežično ili žično. Tijekom igre, mrežni TCP promet bilježi se pomoću analizatora paketa Wireshark, nakon čega se iz mrežnog prometa izdvajaju međudolazna vremena i duljine paketa. Posebna pažnja posvećena je statističkoj analizi međudolaznih vremena, nad čime se provodi identifikacija distribucije slučajne varijable. Cilj analize podataka je pridruživanje odgovarajuće statističke funkcije korištenjem jezika Python i modula Scipy.stats.

U ovom radu nalazi se 4 poglavlja. U prvom poglavlju je opisana višekorisnička video igra „Wizard wars“ koja je služila kao početna točka ovog rada. U drugom poglavlju je objašnjena ideja, dizajn i implementacija računalno generiranih protivničkih likova. U trećem poglavlju je opisan način izvođenja eksperimenta te mrežne konfiguracije pojedinih eksperimenata. Također je objašnjena napravljena analiza mrežnog prometa. U četvrtom poglavlju nalaze se dobiveni rezultati te diskusija o dobivenim rezultatima.

1. Prethodno razvijena igra „Wizard wars“

Kao početna točka diplomskog rada korištena je igra dobivena u sklopu kolegija Diplomski projekt. Razvijena je višekorisnička videoigra „Wizard wars“ kojoj je tema magična bitka između više korisnika povezanih lokalnom mrežom. Korisnik upravlja avatarom čarobnjaka koji se može proizvoljno kretati po površini unutar virtualnog svijeta. Korisnik svojim avatarom pomoću čarobnog štapa može stvarati čarolije koje mu služe kao napad na protivnike. Kako bi se branio od napada protivnika, korisnik može stvoriti magični štit koji ga brani od napada koji se sudare s magičnim štitom. Implementirane su funkcionalnosti praćenja zdravlja korisnika pomoću vizualne letvice koja se nalazi iznad korisnika (eng. *health bar*), povratak na početnu točku nakon umiranja (eng. *respawn*). Korisnici se međusobno razlikuju u virtualnom svijetu pomoću kapsule koja se nalazi iznad svakog korisnika te za svakog korisnika ima drugačiju nasumično odabranu boju.

Virtualni svijet podijeljen je na statičke i dinamičke elemente. Statički elementi se u virtualnom okruženju ne mijenjaju kroz vrijeme, u ovoj igri su to:

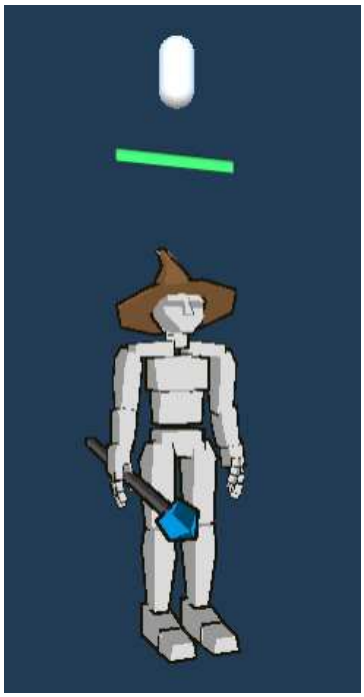
- stabla,
- tlo,
- grmovi,
- ograda koja ograđuje područje igre,
- tekstura pozadine,
- kapsula boje kojom se igrači međusobno razlikuju.

Dinamički elementi imaju funkcionalnosti zbog kojih se kroz vrijeme mijenjaju unutar virtualnog okruženja. U ovoj igri to su:

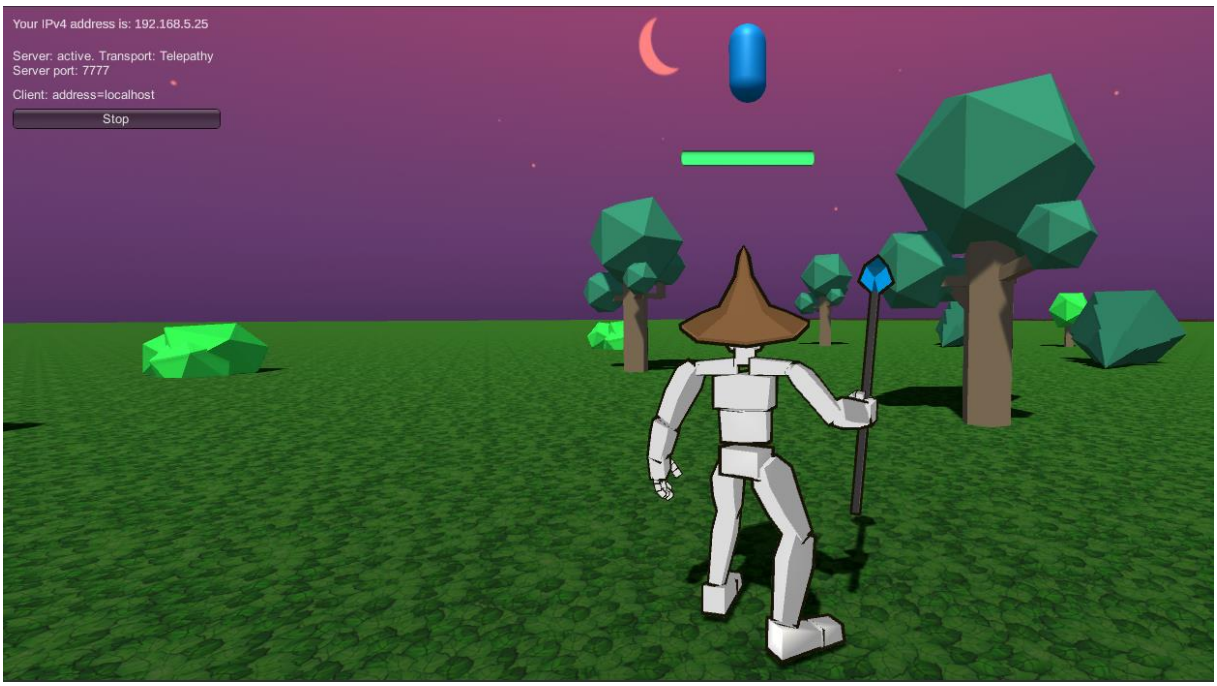
- čarolija napada,
- avatar igrača,
- letvica za praćenje zdravlja korisnika.

Svi statički elementi osim kapsule boje su preuzeti kao gotovi programski paketi. Avatar igrača te njegove animacije su preuzeti kao gotovi programski paketi. Humanoidni lik ima na sebi definiran kostur, zglobove, mišiće i teksture. Tekstura avatara se sastoji od materijala za tijelo i materijala za šešir. Tijelo avatara je bijele boje, šešir je smeđ. Avatar također sadrži štap, koji je povezan s tijelom i nalazi se u

njegovom koordinatnom sustavu. U nastavku je prikazan avatar čarobnjaka (Sl 1.1) te čarobnjak u virtualnom okruženju (Sl. 1.2)

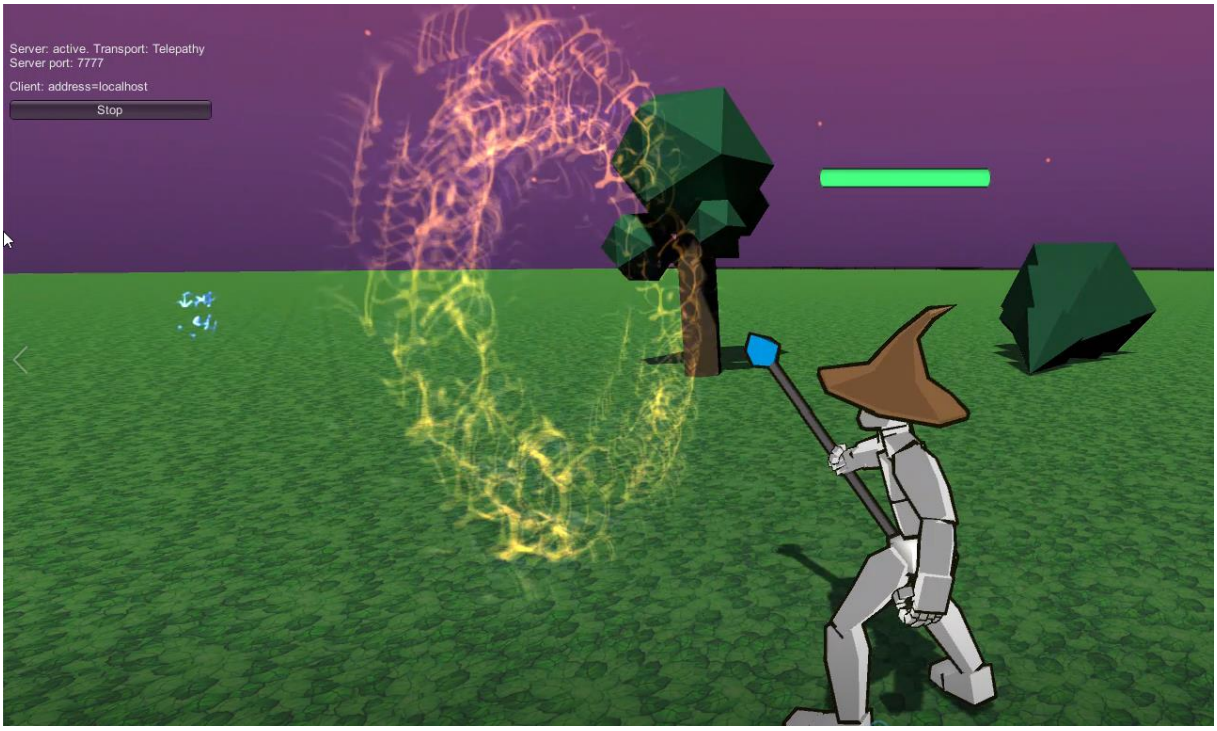


Sl. 1.1 Avatar čarobnjaka



Sl. 1.2 Igrač u virtualnom okruženju

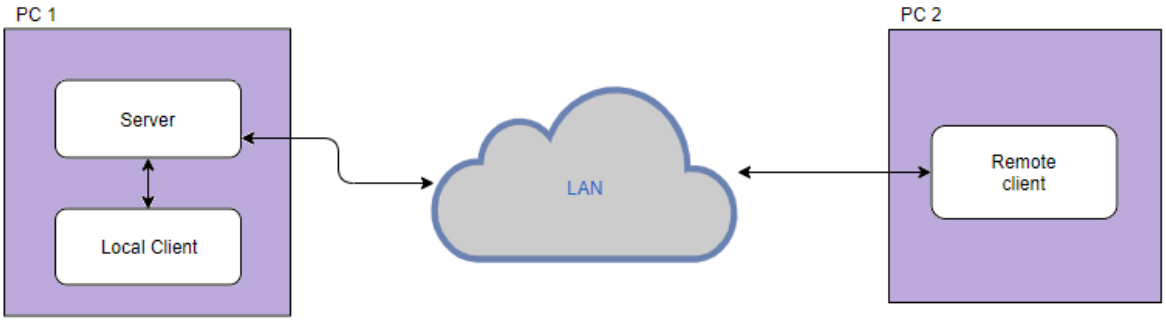
Igrač može stvoriti čaroliju i time napadati svoje protivnike. Nakon što se izvrši animacija napada, ispred štapa čarobnjaka se stvara čestični sustav (eng. *particle system*) oblika na sljedećoj slici (Sl. 1.3). Čestični sustav je preuzet kao zaseban paket koji nije povezan s avatarom čarobnjaka. Dodatno je napravljena funkcionalnost stvaranja čestičnog sustava te interakcije čestičnog sustava s virtualnim svijetom.



Sl. 1.3 Čarolija napada

Za implementaciju projekta korištena je platforma za razvoj igara Unity, verzije 2019.2.8f1 (64-bit). Za izradu skripta u Unity-ju korišten je programski jezik C# te IDE Visual Studio 2017. Za povezivanje više korisnika u isti virtualni svijet korišten je Mirror verzije 5.0.2, programska knjižnica koje pripada HL API alatima (High Level Application Programming Interface). Mirror nam omogućava povezivanje više korisnika u virtualni svijet koristeći arhitekturu klijent poslužitelj. Klijent i poslužitelj međusobno komuniciraju koristeći TCP protokol. TCP konekcija na klijentskoj strani koristi port 7777, a na poslužiteljskoj strani port 50927. Na jednom računalu se pokreće igra u ulozi poslužitelja na kojeg se zatim spajaju klijenti putem lokalne mreže. Također je moguće na istom računalu pokrenuti poslužiteljsku instancu igre i klijentsku instancu igre koja zatim direktno komunicira sa serverskom instancom, odnosno ne koristi

mrežno sučelje za komunikaciju. Na slici 1.4 prikazana je arhitektura videoigre kada su u virtualnom svijetu spojena dva korisnika, od kojih jedan ima ulogu poslužitelja i klijenta.



Sl. 1. 4 Arhitektura klijent poslužitelj

Na slici 1.5 prikazan je dio sučelja u igri s kojim igrač bira želi li pokrenuti samo poslužiteljsku instancu igre (Lan Server Only gumb), poslužiteljsku instancu igre kao i lokalni klijent (Lan Host gumb) ili želi pokrenuti klijenta s kojim se spaja na poslužiteljsku instancu unutar lokalne mreže unoseći IP adresu poslužitelja.

Sl. 1. 5 Dio sučelja kojim igrač bira način pokretanja igre

2. Proširenje postojeće igre

Postojeći sustav proširen je dodavanjem neigračkih likova (engl. *non-player character – NPC*), odnosno likova koje igrač ne kontrolira u igri te ti likovi napadaju avatara igrača. Točnije, proširenje je napravljeno u dva koraka:

- Kreiranje stacionarnog protivnika koji napada korisnika
- Kreiranje protivnika koji slijedi i napada korisnika

2.1 Stacionarni protivnik

Stacionarni protivnik je računalno generiran lik koji je postavljen na fiksnu točku unutar virtualnog svijeta. Protivnik nema mogućnost kretanja u horizontalnim smjerovima prostora, ali se može kretati u smjeru okomitom na tlo čime se postiže efekt lebdenja. Također može ponirati i skretati. Bitno svojstvo je da kada mu se igrač približi i uđe u područje interesa on prati kretanje igrača skretanjem te pokušava napasti istog. To radi tako da kreira čarolije koje se kreću u sceni te ako pogode avatar igrača, igrač gubi dio zdravlja. S obzirom na to da protivnika igrač može pogoditi, nakon prvog pogotka protivnik umire, ostavljajući iza sebe animaciju umiranja. Ponašanje napada stacionarnog protivnika opisano je sljedećim pseudokodom:

```
Dok nema igrača u području interesa, ponašaj se uobičajeno.

Kad u područje interesa uđe igrač, prati njegovo kretanje
te u vremenskim intervalima kreiraj čarolije napada u smjeru
igrača.
```

Izgled samog stacionarnog protivnika zamišljen je kao poluprozirna obojena sfera s dodatkom čarobnog štapića. Umiranje protivnika počinje skupljanjem a završava eksplozijom koja ne može nauditi igraču.

2.2 Protivnik koji slijedi i napada igrača

Ovaj protivnik je unaprjeđenje stacionarnog protivnika. Isto kao i stacionarni protivnik, ovaj protivnik se od početka nalazi na fiksnoj točki unutar virtualnog svijeta na kojoj lebdi te na njoj ostaje sve do ulaska igrača u područje interesa. U tom slučaju,

protivnik ima mogućnost kretanja u svim horizontalnim smjerovima. Također ima mogućnost poniranja i skretanja.

Njegova glavna značajka je mogućnost napada na igrača stvaranjem čarolije prilikom kretanja prema istom. Radi realističnosti stvaranja čarolije, protivnikovom čarobnom štapiću dodana je animacija zamaha štapićem. Protivnik ima svojstvo da nakon što ga igrač pogodi svojom čarolijom napada on nestaje ostavljajući za sobom animaciju umiranja.

Njegovo najbitnije svojstvo je mogućnost praćenja igrača prilikom ulaska igrača u njegovo područje promatranja. Ako se igrač udalji, protivnik ga prestaje napadati, ali ga i dalje slijedi. Kada se igrač udalji još više, protivnik dolazi na posljednju poziciju igrača koju je zapamtio i tamo čeka idućeg igrača.

Ponašanje protivnika koji slijedi i napada igrača definirano je sljedećim pseudokodom:

```
Dok nema igrača u području interesa, ponašaj se uobičajeno.
Kad u područje interesa uđe igrač:
    Slijedi igrača.
    Ovisno o udaljenosti od igrača, promijeni boju.
    Ako je udaljenost između protivnika i igrača ispod praga, kreiraj čarolije napada u smjeru igrača u određenim vremenskim intervalima.
Ako je igrač napustio područje interesa:
    Odi na posljednju zapamćenu lokaciju igrača prije izlaska iz područja interesa.
```

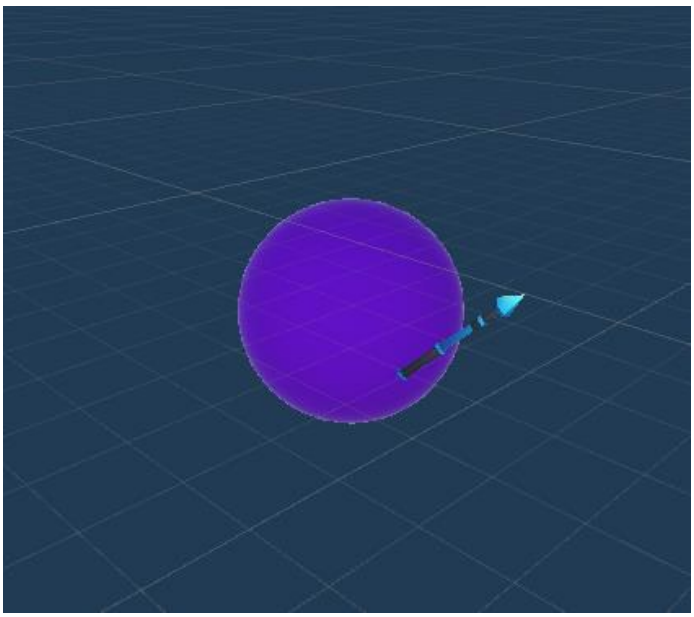
Kao i kod stacionarnog protivnika, izgled je zamišljen kao poluprozirna obojena sfera s dodatkom čarobnog štapića. Posebnost u izgledu ovog protivnika je što postepeno mijenja boju kako se približava igraču, do trenutka kada ga počne napadati. Umiranje je zamišljeno na isti način kao kod stacionarnog protivnika.

2.3 Programska izvedba

Programska izvedba nadogradnje postojeće digitalne videoigre „Wizard wars“ nastavljena je u Unity-u verzije 2019.2.8f1.

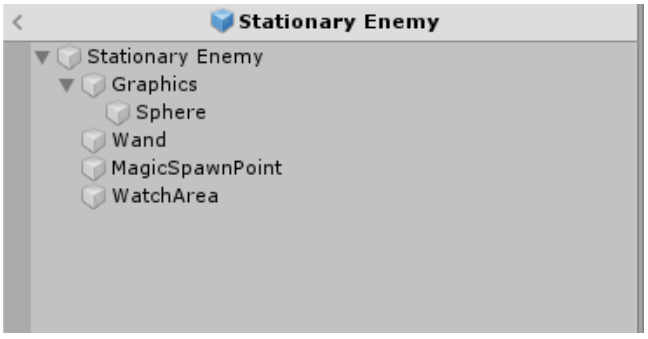
2.3.1 Kreiranje stacionarnog protivnika

Za izradu stacionarnog protivnika korišten je Unity te njegov editor. Izgled stacionarnog protivnika prikazan na slici 2.1.



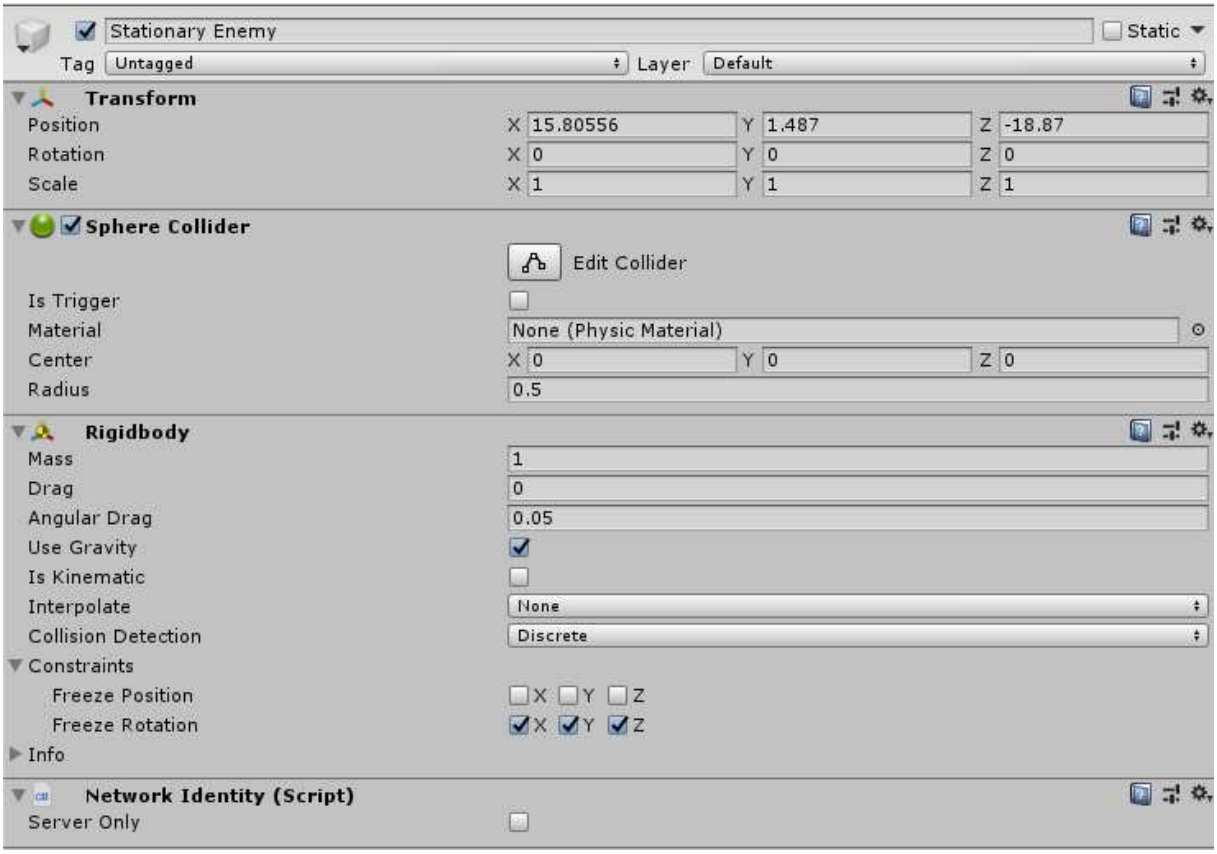
Sl. 2. 1. Izgled stacionarnog protivnika

Što se tiče hijerarhije objekata, protivnik je podijeljen na nekoliko objekata, svaki ima posebnu ulogu i obavlja jedan dio posla koji kao cjelina čine ponašanje i izgled protivnika. Ti objekti su: Stationary Enemy, Graphic, Sphere, Wand, MagicSpawnPoint te Watch Area. Na slici 2.2 je prikazana hijerarhija navedenih objekata.



Sl. 2. 2. Hijerarhija objekata Stacionarnog protivnika

Game objekt naziva „Stationary enemy“ ima ulogu interakcije sa svijetom što se tiče detekcije sudara i fizikalne simulacije unutar virtualnog okruženja. To postiže pomoću dvije komponente koje su već izrađene unutar Unity-a: Sphere collider i RigidBody. Sphere collider se koristi za detekciju sudara protivnika s okolinom u virtualnom svijetu. RigidBody nam koristi kako bi na protivnika mogla utjecati fizikalna simulacija virtualnog okruženja. Ovaj objekt također sadrži komponentu „Network identity“ iz paketa Mirror: Ta komponenta definira jedinstven identifikator svakog protivnika unutar mreže. Na slici 2.3 su prikazane postavke svake pojedine komponente na objektu.

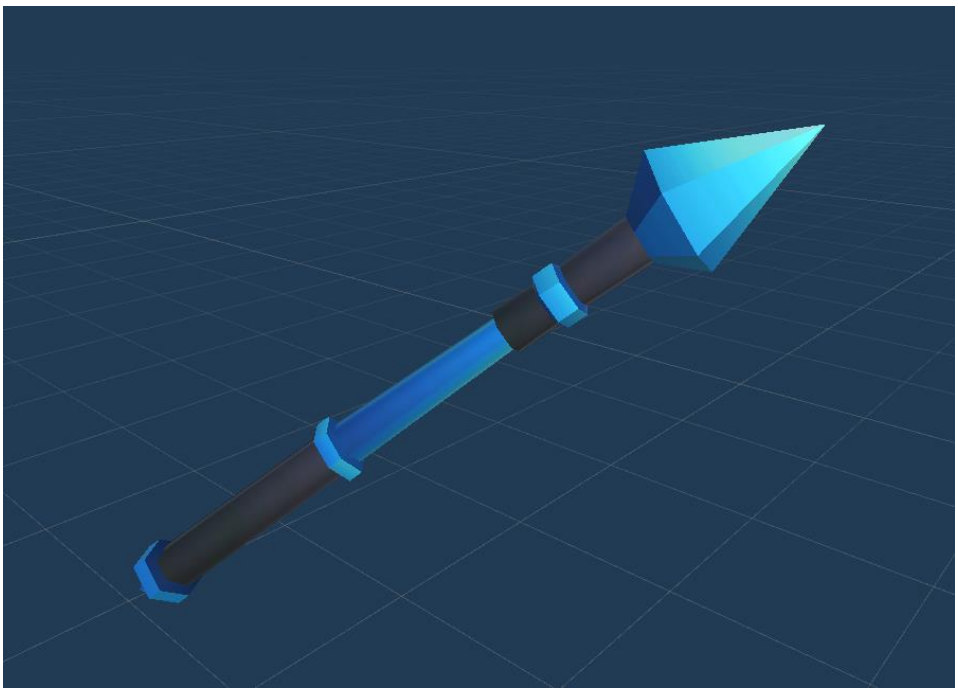


Sl. 2. 3. Komponente Stationary Enemy objekta

Graphics je roditeljski game objekt kojem je dijete Sphere, služi u svrhu lakše manipulacije hijerarhijom game objekata.

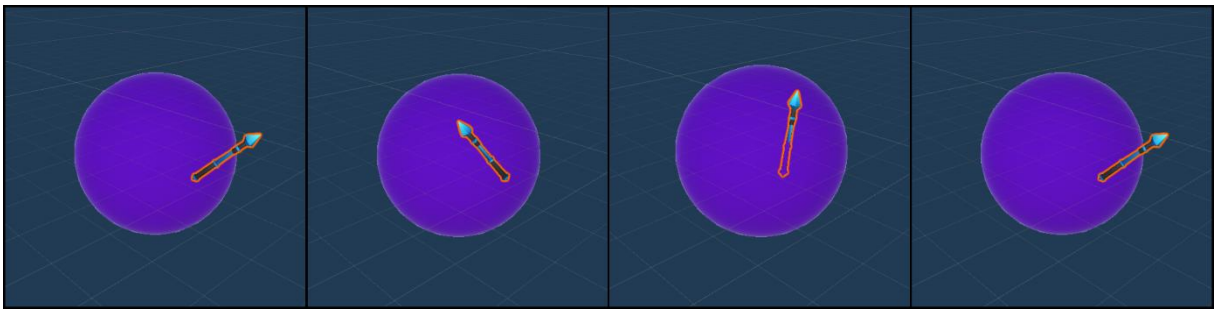
Sphere je game objekt koji se nalazi u lokalnom koordinatnog sustavu Graphics game objekta te je njegova svrha prikazivanje i iscrtavanje izgleda protivnika u sceni. Komponenta kojom se to postiže je Mesh renderer, Unity-jeva komponenta kojom se materijal i teksture game objekta iscrtavaju u virtualnoj sceni.

Sljedeća komponenta u hijerarhiji je objekt Wand, koji predstavlja čarobni štapić protivnika, prikazano na slici 2.4.



Sl. 2. 4. Čarobni štapić protivnika

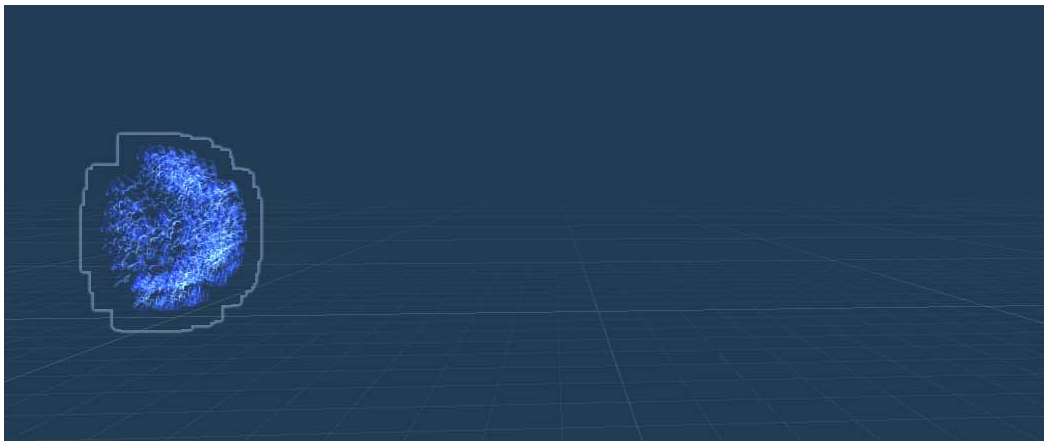
Štapić je specifičan po tome, što se ispred njegovog vrha nalazi prazan game objekt Magic Spawn Point iz kojeg se stvara protivnikova čarolija napada. Također, kako bi se objekt doimao što realističnije, pomoću Unity-jevog prozora animatora kreiran je isječak animacije. Animacija simulira ljudski pokret zamaha čarobnim štapićem prikazane na slici 2.5.



Sl. 2. 5. Animacija pokreta zamaha čarobnim štapićem

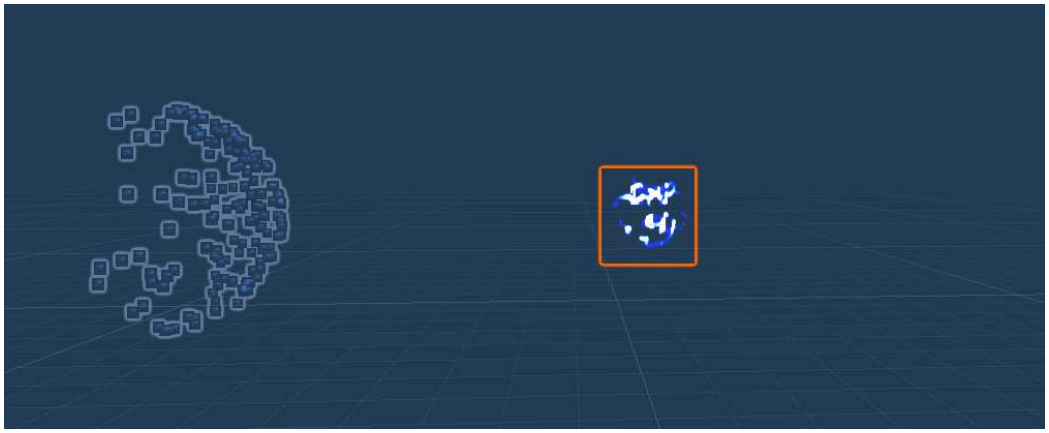
Čarolija napada nastala je kao čestični sustav preuzet iz diplomskog projekta. Nakon izvršenja cijele animacije prikazane na slici 2.6 se stvara čarobni projektil osmišljen kao čestični sustav. Čestični sustav sastoji se od polusfere koja služi kao popratni efekt

stvaranja čarobnog projektila koji zatim putuje kroz scenu u svrhu napada igrača. Čestični sustav je dorađen tako da je iz njega uklonjen vatreni obruč (vidljiv na slici 1.3) te je kao produkt dobivena čarolija koja započinje stvaranjem plavičaste polusfere iz koje zatim izlazi čarobni projektil. Izvođenje čarolije kroz vrijeme možete vidjeti na slikama 2.6, 2.7 i 2.8.



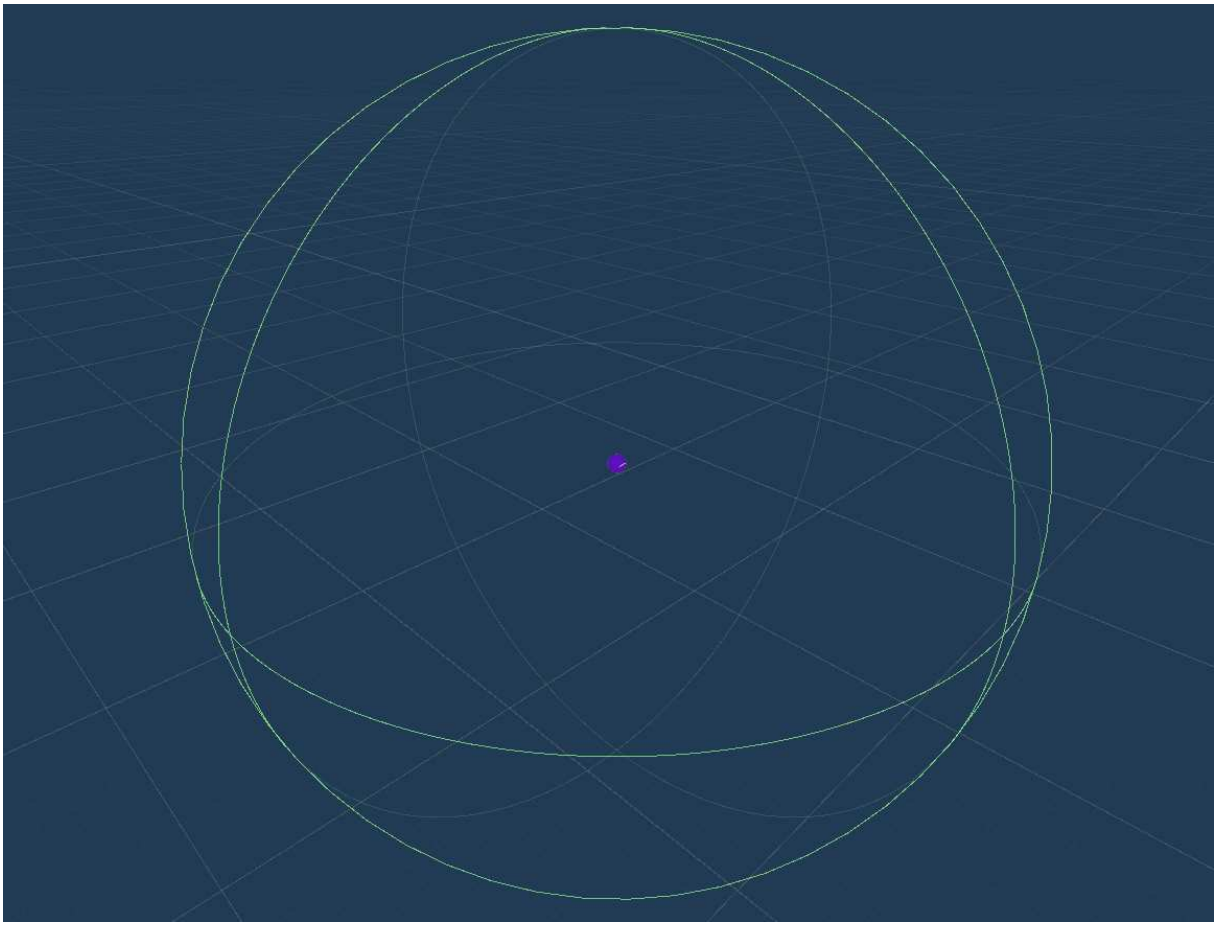
Sl. 2. 6. Početak stvaranja čarolije

Sl. 2. 7. Početak stvaranja čarobnog projektila



Sl. 2. 8. Kretanje čarobnog projektila i nestajanje čarobne polusfere

Posljednji objekt u hijerarhiji je WatchArea odnosno područje interesa protivnika. Radi se o prozirnoj sferi kojoj je središte u ishodištu lokalnog koordinatnog sustava Stationary Enemy-ja, vidljivo na slici 2.9.



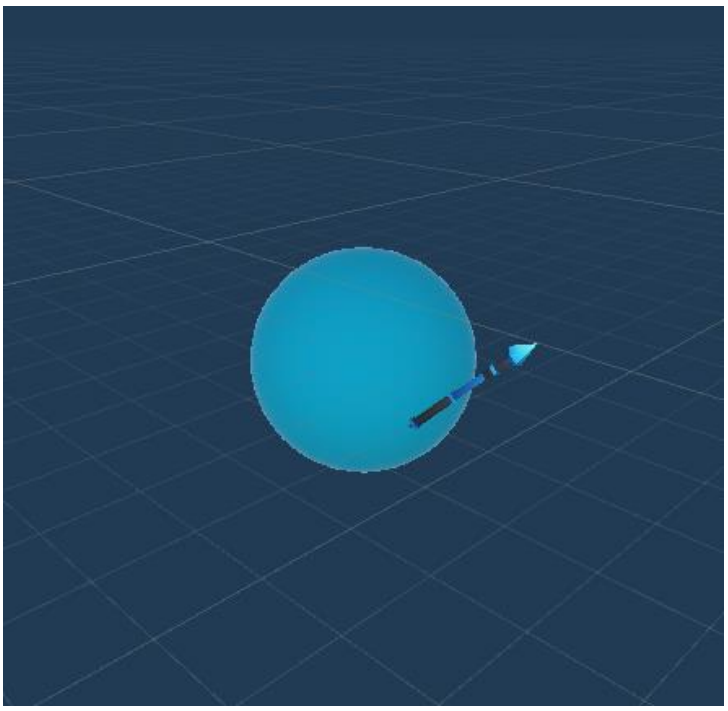
Sl. 2. 9. Udaljeni prikaz protivnika u središtu te transparentne sfere koja predstavlja područje interesa

Objekt sadrži komponentu Collider kojom se registrira igračev ulazak i ostanak unutar sfere. Pomoću skripte Enemy_Behaviour implementirano je odnos i ponašanje između igrača i protivnika.

2.3.2 Kreiranje protivnika koji slijedi i napada igrača

Za izradu protivnika koji slijedi i napada korisnika također je korišten Unity te njegov editor. Protivnik koji slijedi i napada korisnika je unaprjeđenje stacionarnog protivnika. Izgled protivnika koji slijedi protivnika je isti kao i kod stacionarnog, samo nije ljubičaste nego svijetlo plave boje te kako se protivnik približava igraču, mijenja

boju iz plave u crvenu do trenutka kada ga počinje napadati. Izgled protivnika koji slijedi i napada igrača prikazan je na slici 2.10.



Sl. 2. 10. Izgled protivnika koji slijedi i napada igrača

Hijerarhija objekata je gotovo ista kao i kod stacionarnog protivnika, uz dodatak DeathParticles objekta, što je vidljivo na slici 2.11.

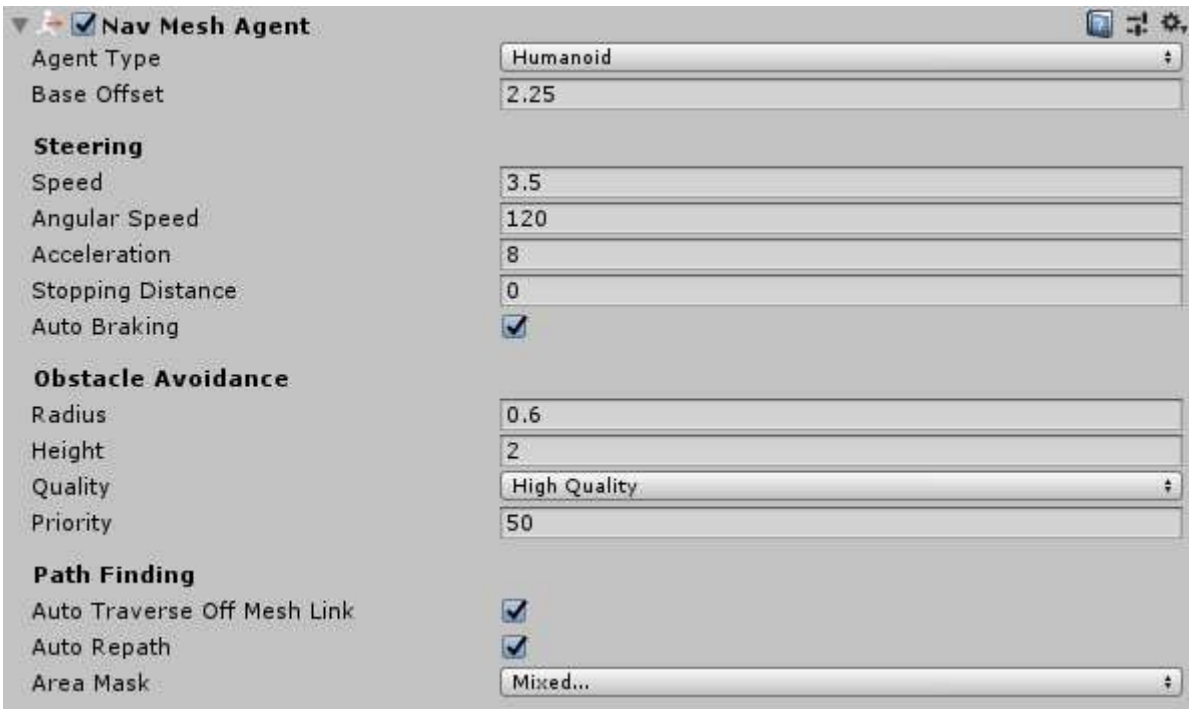
Sl. 2. 11. Hijerarhija protivnika koji slijedi i napada igrača

Following Enemy objekt ima istu ulogu kao i Stationary Enemy kod stacionarnog protivnika. To je vršni objekt na kojem se nalaze komponente već preuzete iz Unity-a: Sphere collider i RigidBody, a služe za detekciju sudara protivnika s okolinom te kako bi fizikalna simulacija virtualnog svijeta mogla djelovati na protivnika. Također sadrži i Nav Mesh Agent komponentu, pomoću koje se protivnik kreće u sceni prema igraču.

Graphics i Sphere objekti služe za iscrtavanje i izgled protivnika unutar virtualnog okruženja.

Wand objekt predstavlja čarobni štapić protivnika. Magic spawn point je prazan game objekt iz čijeg središta nastaje čarolija kojom protivnik pokušava nauditi igraču.

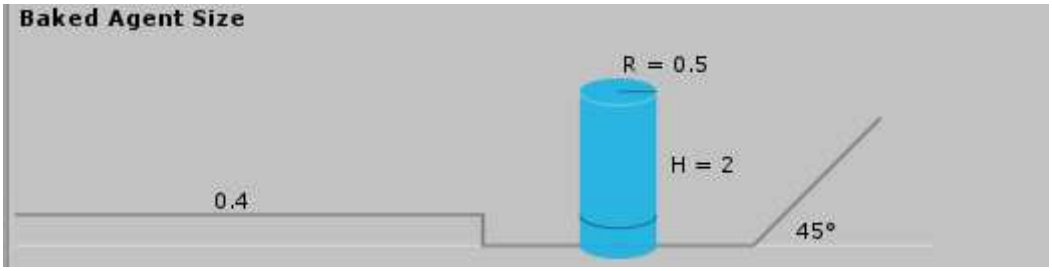
Watch area je objekt koji je zadužen za područje interesa protivnika. To je prozirna sfera koja na sebi ima komponentu Collider pomoću koje registrira ulazak korisnika u područje interesa i skriptu Following_Enemy_Behaviour kojom je definirano ponašanje protivnika. Kretanje ovog protivnika je glavna značajka i unaprjeđenje naspram stacionarnog protivnika. Kako bi se protivnik znao kretati u sceni potrebno je protivniku dati komponentu koja će upravljati njegovim kretanjem, a to je upravo Nav Mesh Agent komponenta spomenuta ranije. Postavke Nav Mash Agent komponente prikazane su na slici 2.12.



Sl. 2. 12. Postavke Nav Mesh Agent komponente

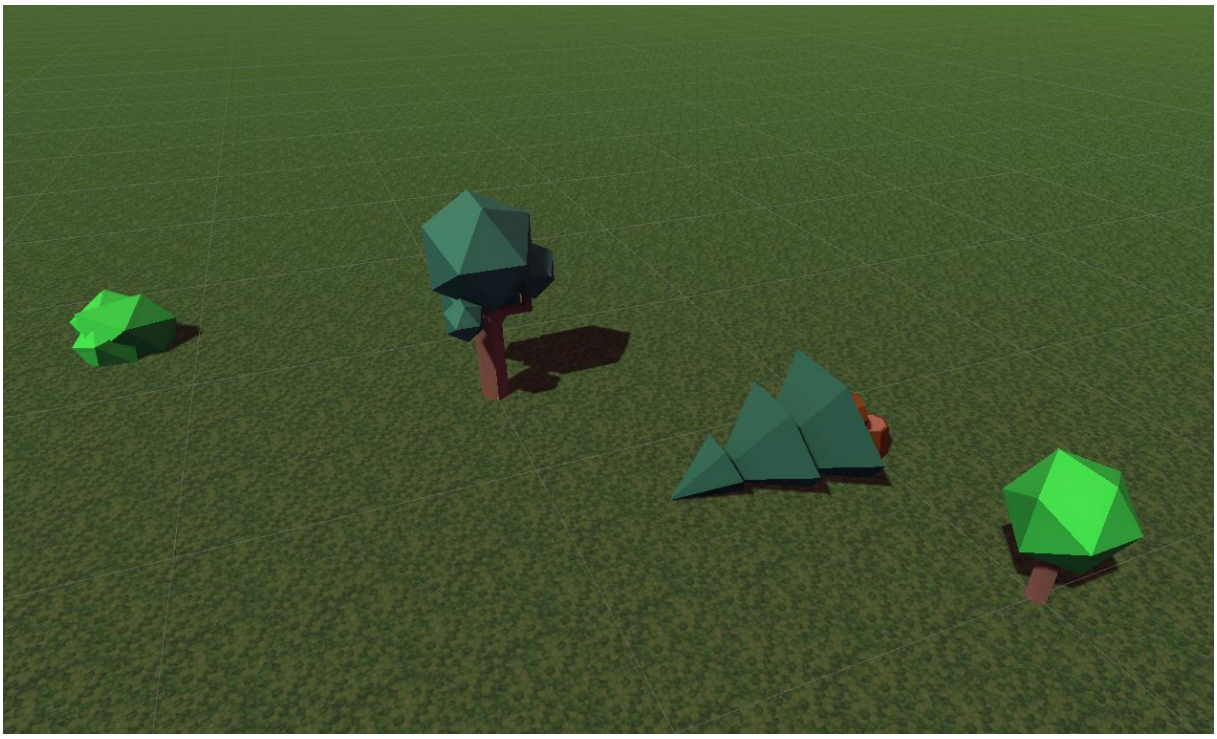
Nav Mash Agent komponenta se bavi pronalaskom puta u virtualnom okruženju između dvije točke. Početna točka je trenutna pozicija protivnika u globalnom koordinatnom sustavu virtualne scene, a krajnja točka je trenutna pozicija igrača kojem se protivnik približava. Put se izračunava korištenjem A* algoritma traženja puta u grafu. Za svaku scenu se mora napraviti graf, koji predstavlja sve površine kojima se

Nav Mash Agent, odnosno protivnik, može kretati. Taj graf se može dobiti koristeći postojeće Unity-eve alate. Kako bi Unity znao izračunati graf kretanja u sceni, potrebno je postaviti parametre agenta koji će se kretati u sceni. Ti parametri su prikazani na slici 2.13.

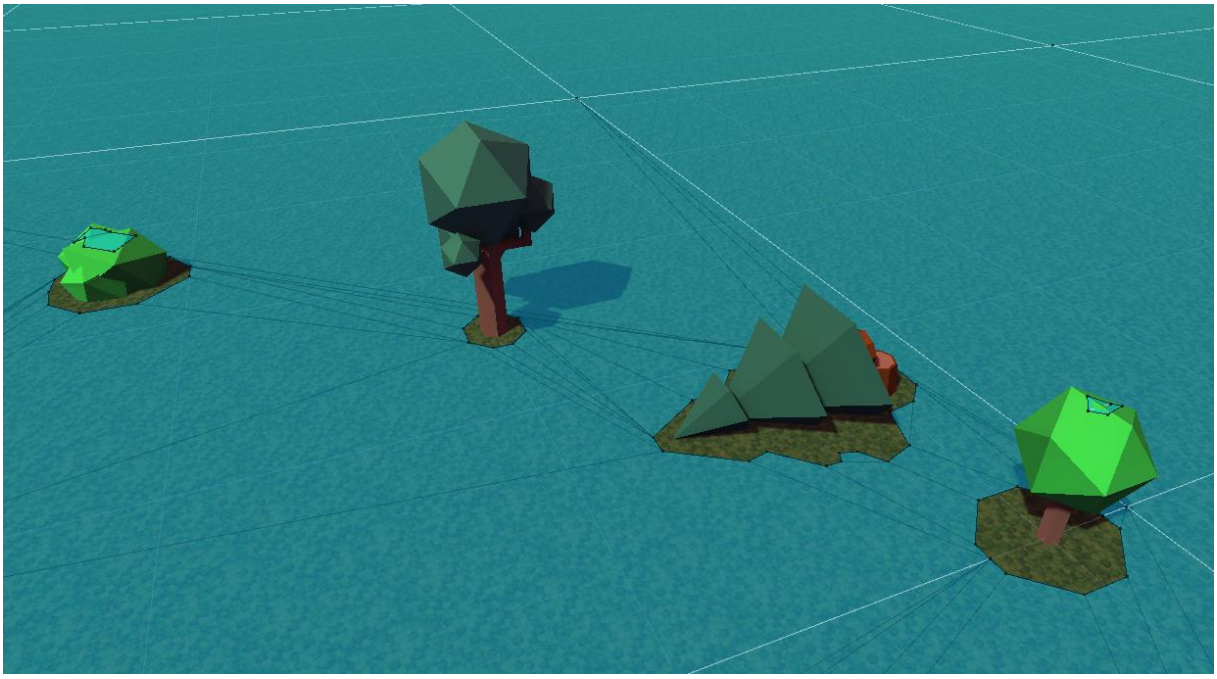


Sl. 2. 13. Parametri agenta za kreiranje grafa kretanja

Vrijednost parametra R sa slike definira polumjer agenta, odnosno minimalan prostor koji je potreban između objekata u sceni kako bi agent mogao proći između njih. Vrijednost parametra H sa slike definira visinu agenta, odnosno donja granica visine ispod koje agent može proći bez kolizije sa scenom i objektom ispod kojeg agent želi proći. Postavljanjem parametra nagiba (engl. *Slope*) definiramo najveću moguću padinu po kojoj se agent može kretati. Postavljanjem parametra visine stepenice (engl. *Step height*) definiramo najveću moguću uzvisinu koju agent može prijeći. Definiranjem ovih parametara Unity može izgraditi graf kretanja agenta za promatranu scenu. Na slici 2.14 i 2.15 prikazana je scena s i bez grafa kretanja, koji je napravljen koristeći Unity i parametre definirane na slici 2.13.



Sl. 2. 14. Dio scene prije izračuna grafa kretanja



Sl. 2. 15. Dio scene nakon izračuna grafa kretanja

Na slikama je prikazana izgrađena mreža poligona plavkaste boje, koja predstavlja graf unutar kojega se naš agent može kretati. Sa slike se mogu vidjeti primjeri područja oko objekata u sceni kojima se definiraju granice do kojih agent može doći.

2.3.3 Skripta koja upravlja kretanjem protivnika

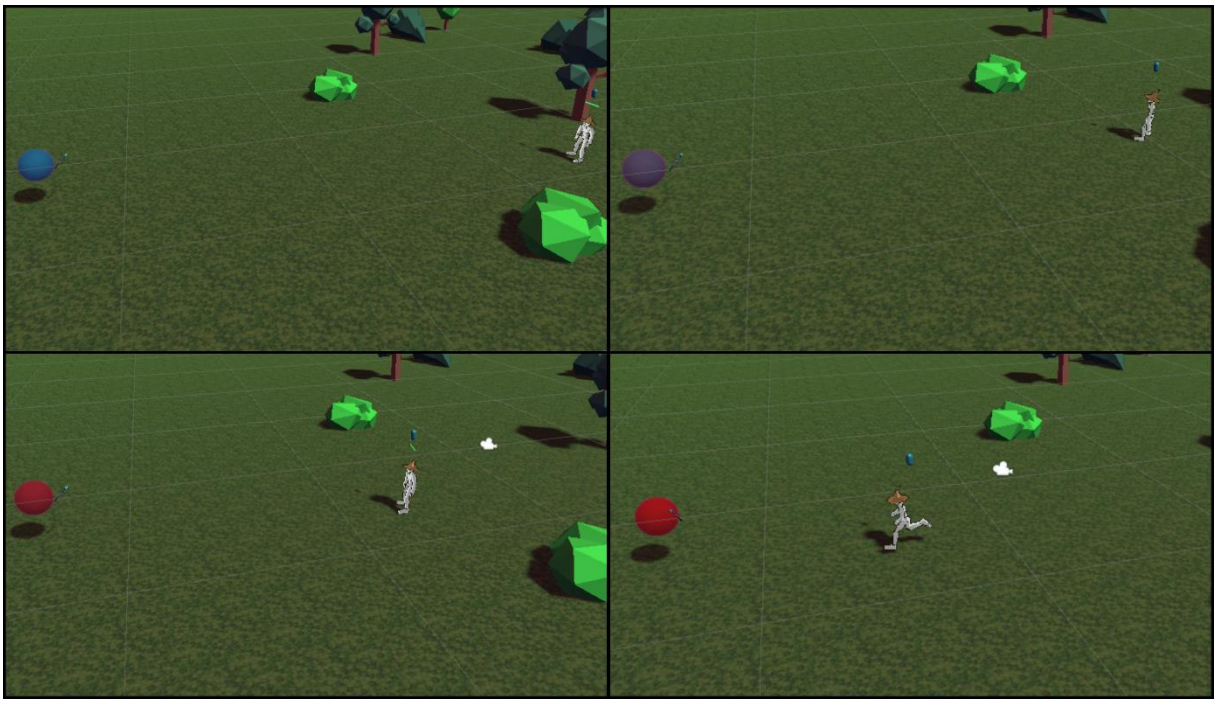
Implementacija kretanja napravljena je skriptom Following_Enemy_Behaviour koja se nalazi na WatchArea objektu. Na slici 2.16 prikazan je isječak koda kojim se agent kreće prema igraču te mijenja boju iz plave u crvenu kako mu se približava.

```
// Update is called once per frame
0 references
void Update()
{
    if (isLocalPlayer) return;
    if (player != null && canAttack)
    {
        Vector3 lookat = player.transform.position;
        Vector3.ProjectOnPlane(lookat, Vector3.forward);
        lookat = lookat + new Vector3(0, 1.4f, 0);
        enemy.transform.LookAt(lookat);
        agent.destination = player.transform.position;

        //Get distance between those two Objects
        var distanceApart = Vector3.Distance(this.transform.position, player.transform.position);

        //Convert 15 and 30.7 distance range to 0f and 1f range
        var lerp = MapValue(distanceApart, 15f, MAX_DISTANCE, 0f, 1f);

        //Lerp Color between near and far color
        Color lerpColor = Color.Lerp(near, far, lerp);
        ChangeColor(lerpColor);
    }
    if(!canAttack)
    {
        agent.destination = enemy.transform.position;
    }
}
```

Sl. 2. 16. Update metoda skripte Following_Enemy_Behaviour

Unutar Update metode, koja se poziva za svaki izračun slike (engl. *frame*), prvo provjeravamo poziva li se ova metoda na serveru pomoću varijable isLocalPlayer. Ta varijabla je false ako se metoda poziva sa serverske instance, a ako se poziva s klijentske instance, odnosno s računala klijenta, onda je ona true. Zatim provjeravamo postoji li igrač kojeg protivnik prati i može li protivnik uopće napadati. Provjera može li protivnik napadati se koristi kako protivnik ne bi mogao napadati korisnika dok se odvija animacija umiranja. Zatim uzimamo vektor pozicije igrača, projiciramo ga na ravninu tla te ga translatiramo za 1.4f u smjeru pozitivne vertikalne osi. Tada protivnika rotiramo tako da gleda prema tom vektoru. Jednom kada je protivnik rotiran, Nav Mesh Agent-u dajemo krajnju točku njegovog puta prema kojem se želi kretati, u ovom slučaju je to vektor pozicije igrača kojeg protivnik prati. Time je postignuta željena funkcionalnost

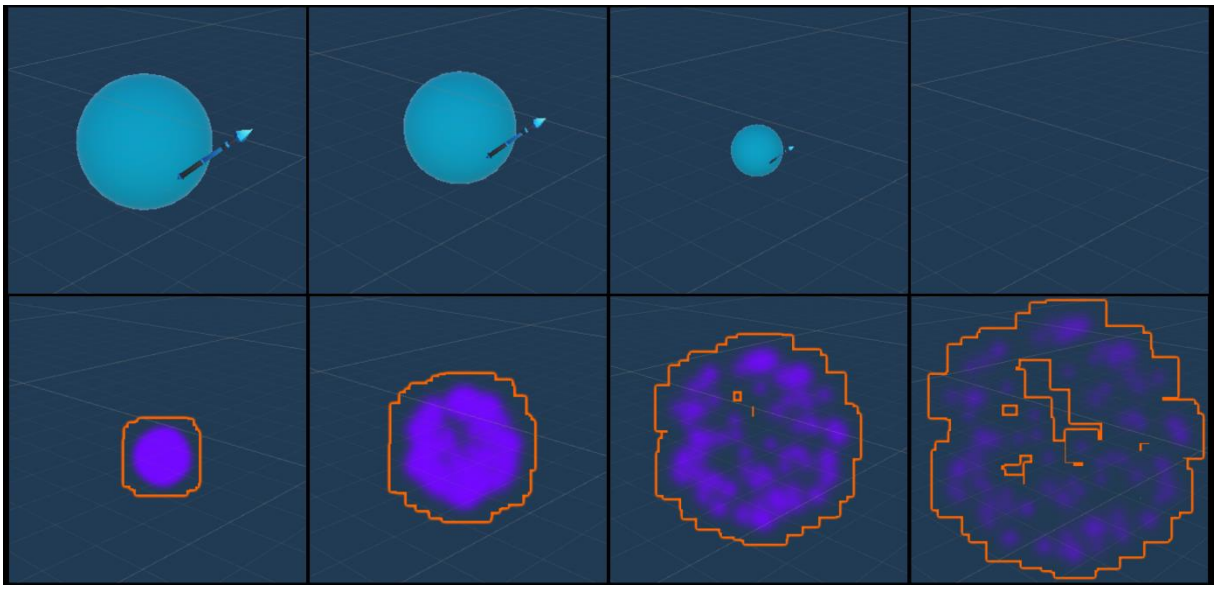
kretanja protivnika, te je potrebno ovisno o udaljenosti igrača i protivnika promijeniti boju između crvene i plave. Prvo se izračuna udaljenost između igrača i protivnika. Zatim se udaljenost skalira na 0 i 1, gdje 0 predstavlja udaljenost gdje protivnik poprima crvenu boju, a 1 udaljenost gdje je svijetlo plave boje. Potom koristimo interpolaciju kako bi dobili boju protivnika ovisno u trenutnoj udaljenosti između igrača i protivnika. Na kraju mijenjamo boju protivnika i javljamo promjenu svim klijentima u mreži. Promjena boje protivnika kako se isti približava igraču vidljiva je na slici 2.17.



Sl. 2. 17. Promjena boje protivnika u ovisnosti o udaljenosti od igrača

2.3.4 Skripta umiranja protivnika

Implementacija umiranja protivnika napravljena je pomoću skripte Enemy_Death. Unutar te skripte se provjerava je li protivnik pogođen čarolijom. Budući da protivnik ima samo jedan život, već pri prvom pogotku od neke čarolije on umire. Nakon detekcije sudara čarolije i protivnika, pokreće se animacija umiranja. Ta animacija je skupljanje (engl. *Shrink*) protivnika u jednu točku. Nakon te animacije se pokreće čestični sustav koji daje realistični prikaz eksplozije čestica iz jedne točke. Spajanjem animacije i čestičnog sustava napravljen je uvjerljiv prikaz protivnikove smrti. Smrt protivnika prikazana je na slici 2.18.



Sl. 2. 18. Umiranje protivnika

3. Analiza mrežnog prometa

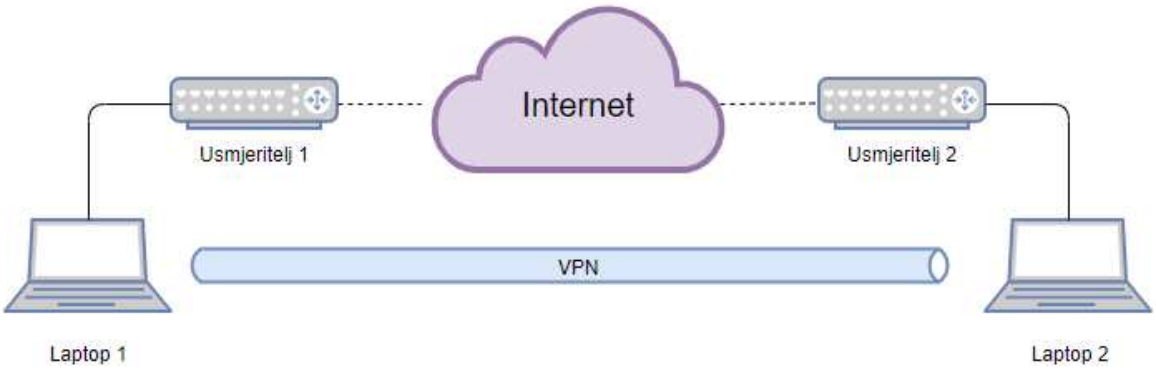
Analiza mrežnog prometa je proces presretanja, snimanja i analize obrazaca mrežnog prometa. Česti parametri analize su veličine dolaznih paketa te njihova međudolazna vremena. Postavlja se pitanje postoji li neka matematička pravilnost u veličinama paketa i u međudolaznim vremenima paketa koja se može pronaći statističkom obradom podataka, odnosno prikupljenog mrežnog prometa. Alati za analizu mrežnog prometa mogu se podijeliti na programske ili sklopovske. Sklopovski alati ugrađeni su u telekomunikacijsku opremu, dok su programska rješenja šire namjene, a sposobna su analizirati mnogo više protokola.

U ovom poglavlju napravljeni su eksperimenti za 3 različite mrežne konfiguracije u kojima se vrši interakcija s računalno generiranim likovima tijekom kojih se snima mrežni promet koristeći alat Wireshark. Iz prikupljenog mrežnog prometa se promatraju vrijednosti duljina paketa te vremenske oznake paketa. Iz dobivenih vremenskih oznaka obradom podataka su dobivena međudolazna vremena paketa. Na posljetku je nad obrađenim podacima napravljena mrežna analiza za svaku konfiguraciju kako bi se probala pronaći matematička pravilnost u prikupljenom prometu.

3.1 Konfiguracije provedenih eksperimenata

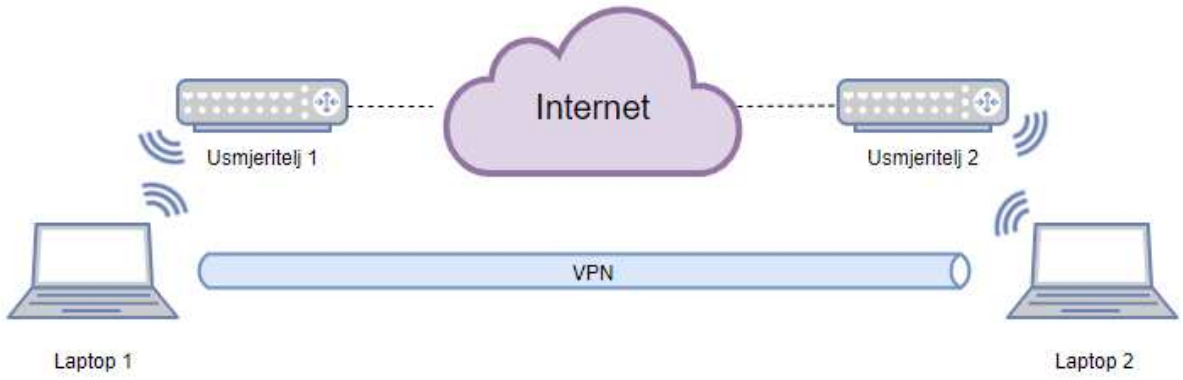
Za prikupljanje podataka osmišljene su tri različite konfiguracije mrežne topologije:

1. K1, konfiguraciju u kojoj su oba korisnika na mrežno sučelje spojeni žicom



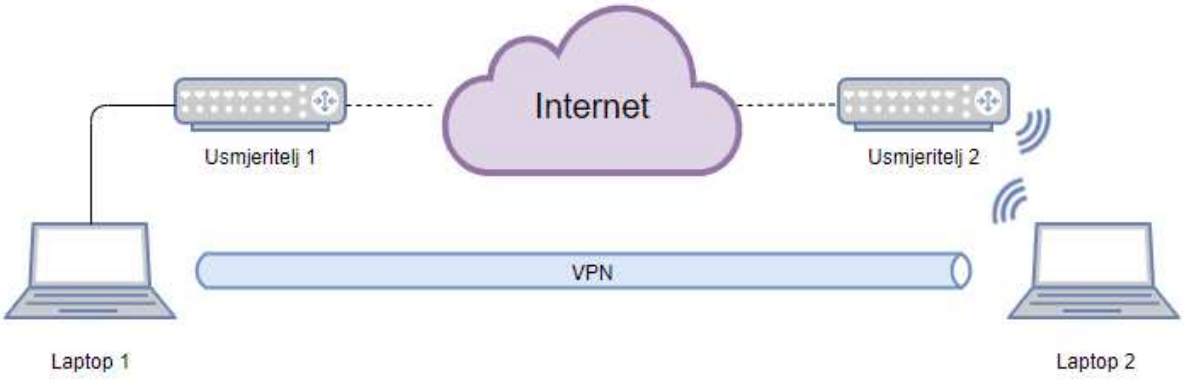
Sl. 3. 1. Mrežna topologija gdje su oba korisnika spojena žicom

2. K2, konfiguraciju u kojoj su oba korisnika na mrežno sučelje spojeni bežično



Sl. 3. 2.Mrežna topologija gdje su oba korisnika spojena bežično

3. K3, konfiguracija u kojoj je jedan korisnik na mrežno sučelje spojen žicom, a drugi bežično



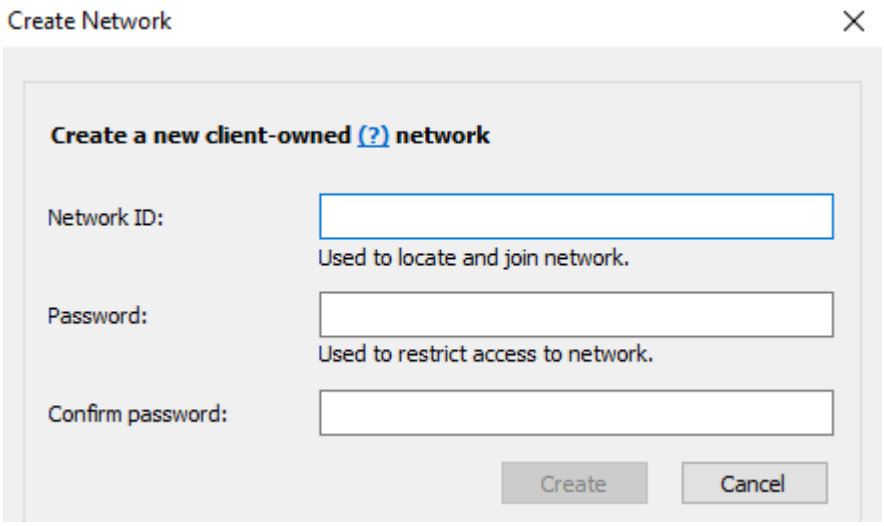
Sl. 3. 3. Mrežna topologija gdje je jedan korisnik spojen žicom, a drugi bežično

Budući da je igra implementirana za rad u lokalnoj mreži, a računala se nalaze u različitim privatnim mrežama koje su povezane internetom, za uspješnu komunikaciju se koristi aplikacija LogMeIn hamachi, koja daje mogućnost izgradnje virtualne privatne mreže preko interneta. Slike mrežnih topologija nalaze se na slikama 3.1, 3.2 i 3.3.

3.2 Kreiranje virtualne privatne mreže koristeći LogMeIn Hamachi

Budući da su korisnici spojeni preko interneta, kako bi komunikacija između računala bila izvediva i uspješna koristi se aplikacija LogMeIn Hamachi koja omogućava kreiranje virtualnih privatnih mreža. Pomoću Hamachi-ja se putem interneta uspostavlja veza između računala koja oponaša vezu koji bi postojala kada bi ta računala bila spojena u lokalnu mrežu (engl. *Local Area Network – LAN*). Time Hamachi ostvaruje vezu između računala koja su zaštićena NAT (engl. *Network Address Translation*) poslužiteljem i vatrozidom (engl. *Firewall*).

Kreiranje virtualne privatne mreže unutar aplikacije LogmeIn Hamachi započinje jedan korisnik koji napravi virtualnu mrežu koja ima jedinstveno ime i lozinku, prikazano na slici 3.4.



Sl. 3. 4. Kreiranje virtualne privatne mreže unutar aplikacije LogMeIn Hamachi

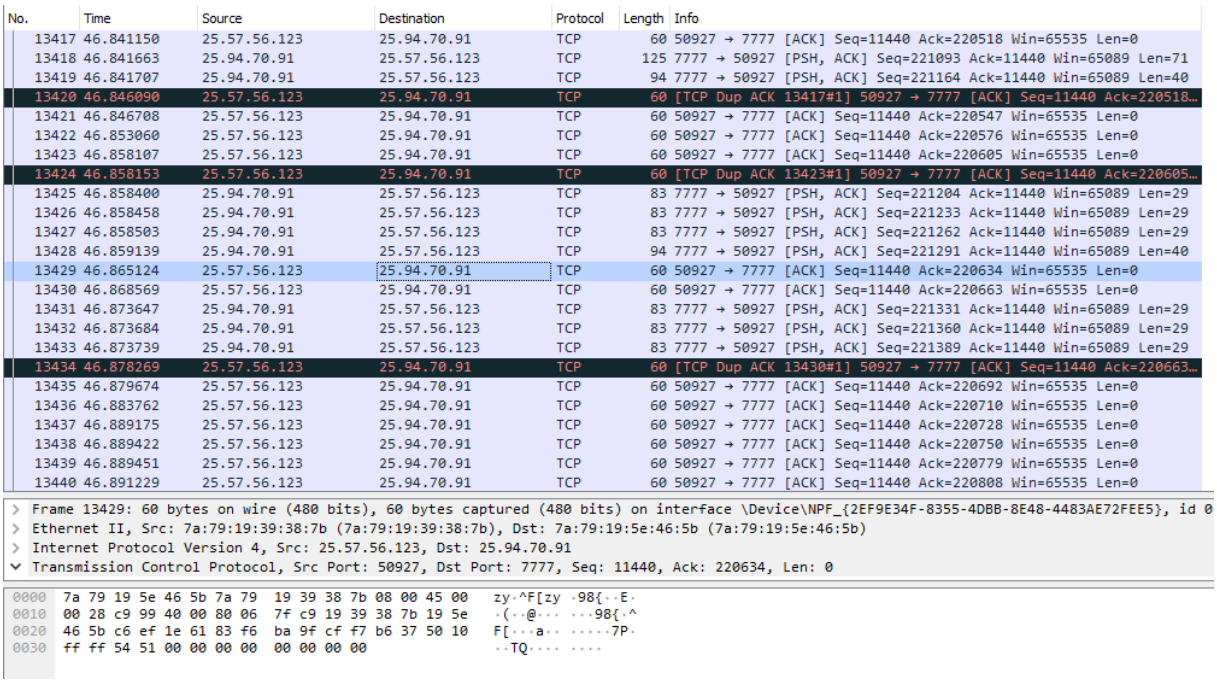
Nakon kreiranja virtualne privatne mreže, korisnik postaje prvi čvor unutar VPN-a (engl. *Virtual Private Network*). Drugi korisnik se zatim može spojiti na virtualnu privatnu mrežu koristeći jedinstveno ime i lozinku koju je osmislio prvi korisnik tijekom kreiranja VPN-a. Spajanjem drugog korisnika u VPN-u postoje dva čvora koja mogu međusobno komunicirati preko interneta. Nakon kreiranja privatne mreže mjerena je vrijednost vremena odziva (engl. *Round Trip Time – RTT*) koristeći naredbu ping operacijskog sustava Windows te je ustanovljeno da se vrijednost nalazi unutar intervala od 60 do 100 ms.

3.3 Wireshark

U ovom radu za snimanje mrežnog prometa koristi se analizator mrežnog prometa Wireshark. Wireshark je besplatni program otvorenog koda (engl. *open source*) koji se koristi za analiziranje snimljenog mrežnog prometa. Ovaj programski alat koristi programsku knjižnicu pcap (engl. *packet capture*) koja definira API (engl. *application programming interface – API*) te dohvat paketa s mrežnih sučelja različitih operacijskih sustava, u slučaju ovog rada Windows OS-a (engl. *Operating System*). Mogućnosti alata Wireshark su različite, a najbitnije, koje i njegov proizvođač ističe, su:

- hvatanje podatkovnih paketa s mrežnog sučelja,
- prikazivanje paketa s vrlo detaljnim informacijama o mrežnom protokolu,
- otvaranje i spremanje paketa,
- uvoz i izvoz podataka u druge slične programe,
- pretraga i filtriranje paketa po raznolikim kriterijima.

U radu je za snimanje i analizu mrežnog prometa korištena Wireshark verzija 3.2.4. Primjer sučelja s TCP prometom prikazan je na slici 3.5.



Sl. 3. 5. Wireshark sučelje

3.4 Postupak provođenja eksperimenta

Eksperiment je započet odgovarajućim spajanjem korisnika na mrežno sučelje. Ovisno o konfiguraciji eksperimenta koji se provodio, korisnici su bili spojeni ili žičano, UTP kabelom, ili bežično, preko WiFi-ja. Nakon spajanja na mrežno sučelje, bilo je potrebno pokrenuti aplikaciju LogMeIn Hamachi kojom se korisnici spajaju na virtualnu privatnu mrežu. Jednom kada su korisnici na istoj virtualnoj privatnoj mreži, oba korisnika pokreću snimanje mrežnog prometa, pritom filtrirajući mrežni promet kako bi bili uhvaćeni isključivo paketi poslani preko VPN-a. Zatim oba korisnika pokreću igru na svom računalu s time da jedan korisnik odabire da igru pokrene kao poslužitelj (engl. *Server*), dok se drugi igrač zatim spaja na serversku instancu igre koristeći virtualni IP drugog igrača unutar virtualne mreže.

Kad su oba igrača u virtualnom svijetu, obvezne su interakcije s računalno-generiranim protivnicima unutar virtualnog svijeta na 30 do 60 sekundi. Pod interakcijama korisnika s računalno-generiranim protivnicima se smatra ulazak u područje interesa, napadanje protivnika čarolijama, izlazak iz područja interesa protivnika. Također su dopuštene i druge interakcije u virtualnom okruženju, kao npr. napadanje drugog igrača, podizanje štita i sve akcije koje igrač može učiniti unutar virtualnog svijeta. Nakon što prođe 30 do 60 sekundi, korisnik koji je pokrenuo igru kao poslužitelj prekida instancu poslužitelja, čime igra završava. Nakon završetka igre zaustavlja se snimanje mrežnog prometa.

3.5 Priprema podataka

Za svaku konfiguraciju je proveden eksperiment 10 puta, odnosno za svaku konfiguraciju postoji 10 pcap datoteka snimljenog mrežnog prometa. Informacije o agregiranim skupovima podataka su dane u tablici 3.1.

Tablica 3. 1 Informacije o agregiranim skupovima podataka za pojedine konfiguracije

Konfiguracija	Ukupan broj paketa	Ukupno trajanje sjednica [s]
K1	126596	425
K2	155356	517
K3	148260	496

Snimljeni mrežni promet potrebno je urediti u svrhu lakšeg analiziranja. Uređivanje je odrađeno tako da se za svaku datoteku unutar eksperimenta snimljeni mrežni promet izvede (engl. *export*) u obliku csv (engl. *coma separated values*) datoteke te zatim pomoću uređivača csv datoteka izdvoje potrebni stupci, vremenske oznake paketa (engl. *timestamp*) te duljine paketa, svaki u svoju datoteku csv datoteku. Na posljetku za svaku konfiguraciju postoje dvije csv datoteke, u jednoj su agregirane sve duljine paketa, a u drugoj sve vremenske oznake svih eksperimenata za gledanu konfiguraciju.

Iz vremenskih oznaka paketa možemo dobiti međudolazno vrijeme paketa. Za promatrane pakete n i n+1, međudolazno vrijeme je vrijeme proteklo od dolaska n-tog paketa do dolaska n+1.-og paketa. Ta informacija se može dobiti oduzimanjem vremenske oznake n+1.-og paketa s vremenskom oznakom n-tog paketa. Dodatno je bilo potrebno obraditi podatke o međudolaznim vremenima prije statističke analize. Iz podataka su izdvojeni paketi koji imaju vrijednost 0, nastali zbog premale razlučivosti vremenske oznake, budući da nije moguće da se dva paketa šalju na mrežno sučelje u isto vrijeme. Također su uklonjene i ekstremne vrijednosti koje su bile u malom broju, koje bi inače pogrešno utjecale na daljnju statističku obradu podataka.

3.6 Skripta za obradu podataka

U svrhu statističke obrade podataka iskorištena je skripta napravljena na kolegiju „Teorija prometa“. Skripta je napravljena u programskom jeziku Python, verzije 2.7.14.

Skripta na početku izvođenja učitava podatke iz neke od csv datoteka spomenutih ranije. Nakon što su podaci učitani, skripta pokušava prilagoditi 50-ak kontinuiranih distribucija ulaznim podacima, npr. eksponencijalna, uniformna, gaussova. Prilagodba podacima se radi koristeći fit() metodu koja je definirana unutar scipy knjižice za svaku promatranu distribuciju. Metodi fit() je potrebno predati podatke, a rezultat su parametri distribucije za koju metoda pokušava pronaći globalnu procjenu najveće izglednosti (engl. *Maximum Likelihood Estimator – MLE*), ali ne garantira pronalazak globalne procjene najveće izglednosti, međutim, postoji mogućnost pronalaska lokalnog optimuma koji ne mora nužno biti globalni optimum. Statistika

definira MLE kao metodu procjene parametara distribucije maksimizirajući funkciju izglednosti, tako da su unutar pretpostavljenog statističkog modela dobiveni podaci najvjerojatniji.

Nakon izračuna parametara promatrane distribucije, radi se Kolmogorov-Smirnovljev test. Kolmogorov-Smirnovljev test se koristi za donošenje odluke potječe li neki uzorak podataka iz neke specifične kontinuirane distribucije te se temelji na dvije početne hipoteze:

1. Nulta hipoteza (H0): podaci dolaze iz specificirane distribucije
2. Alternativna hipoteza (H1): barem jedna vrijednost se ne poklapa sa specificiranom razdiobom

Test vraća dvije vrijednosti, D i p. P vrijednost je dokaz protiv nulte hipoteze. Što je p vrijednost bliže nuli, dokaz je čvršći da dobiveni uzorak ne potječe iz specificirane distribucije. Međutim, problem je što p vrijednost akumulira grešku, odnosno za uzorke s velikom količinom podataka p vrijednost će uvijek težiti prema nuli, čak i ako su podaci generirani iz promatrane distribucije. Iz tog razloga promatramo vrijednost D. D vrijednost je vrijednost između 0 i 1, te nam govori koliko dobro podaci pripadaju promatranoj distribuciji. Po teoremu Glivenko-Cantell, ako podaci dolaze iz promatrane distribucije, za beskonačno mnogo podataka D vrijednost će težiti nuli. Nakon prolaska po svih promatranim distribucija, uzimamo onu koja ima najmanju vrijednost D nakon Kolmogorov-Smirnovljevog testa.

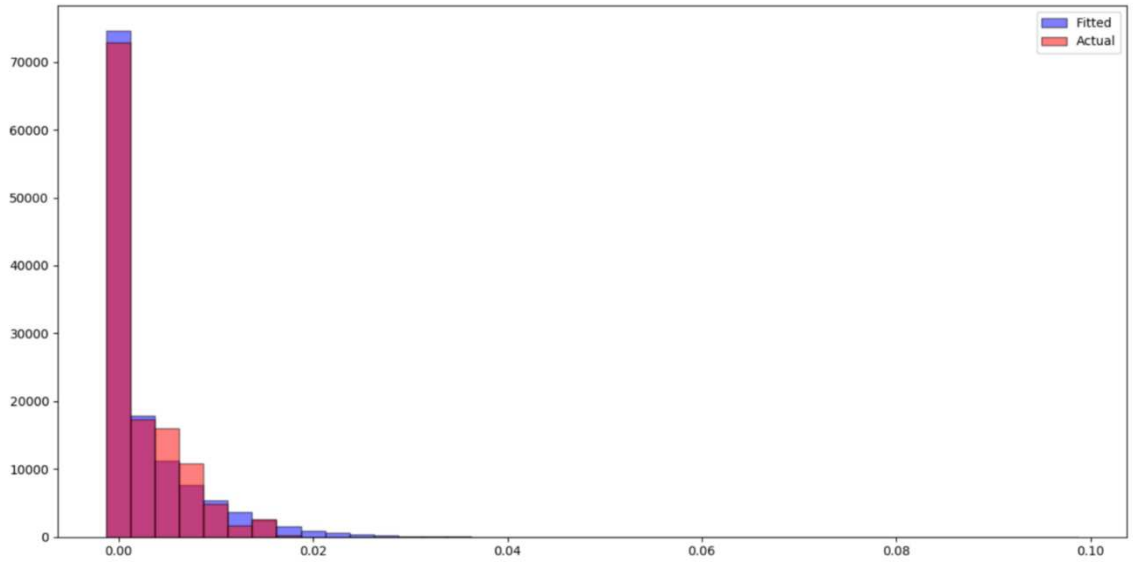
Nakon što smo odredili distribuciju koja najviše odgovara podacima, skripta iscrtava dva histograma na istom grafu. Jedan histogram predstavlja stvarne podatke koji su bili ulaz u skriptu, a drugi predstavlja histogram dobiven generiranjem jednake količine podataka koristeći dobivenu distribuciju s izračunatim parametrima.

4. Rezultati i diskusija

U nastavku su prikazani i diskutirani rezultati za svaku od pojedinih konfiguracija, gledano za međudolazna vremena paketa i duljine paketa. Prilikom izvođenja eksperimenata za različite konfiguracije korisnici nisu primijetili promjene u iskustvenoj kvaliteti.

4.1 Međudolazna vremena

Za konfiguraciju K1 rezultati izvođenja skripte su prikazani na slici 4.1 i 4.2.



Sl. 4. 1. Histogram stvarnih podataka i histogram podataka dobivene distribucije prve konfiguracije

```
Distribution name for time between packets: fatiguelife
Parameters: (5.606215944258489, -5.99125465866613e-07, 0.00018933160867256943)
```

Sl. 4. 2. Ime dobivene distribucije te njeni parametri za prvu konfiguraciju

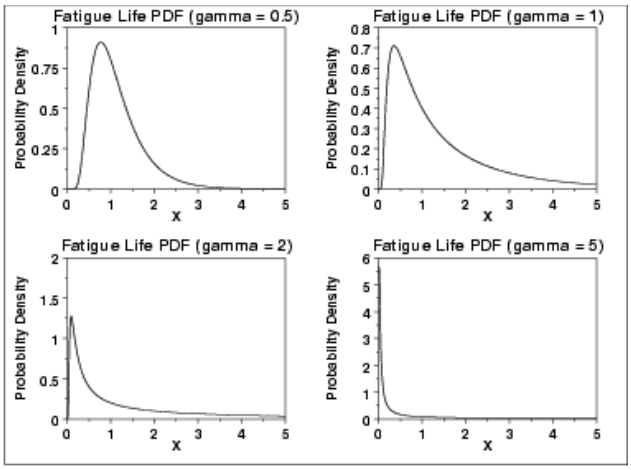
Kod histograma, na osi apscisa se nalazi vrijeme u sekundama (t[s]). Na osi ordinata se nalazi broj paketa u tom intervalu. Ime dobivene distribucije je fatiguelife, što u scipy biblioteci predstavlja Birnbaum-Saundersovu distribuciju. S histograma se može vidjeti poprilično dobro poklapanje, budući da je većina histograma obojena ljubičastom bojom, nastala preklapanjem plave boje generiranih podataka i crvene boje stvarnih podataka.

4.1.1 Birnbaum-Saundersova distribucija

Birnbaum-Saundersova distribucija se primarno koristi za modeliranje vremena ispada u sustavima. Funkcija gustoće vjerojatnosti ove distribucije je sljedeća:

$$f(x) = \left(\frac{\sqrt{\frac{x-\mu}{\beta}} + \sqrt{\frac{\beta}{x-\mu}}}{2\gamma(x-\mu)} \right) \phi \left(\frac{\sqrt{\frac{x-\mu}{\beta}} - \sqrt{\frac{\beta}{x-\mu}}}{\gamma} \right) \qquad x > \mu; \gamma, \beta > 0$$

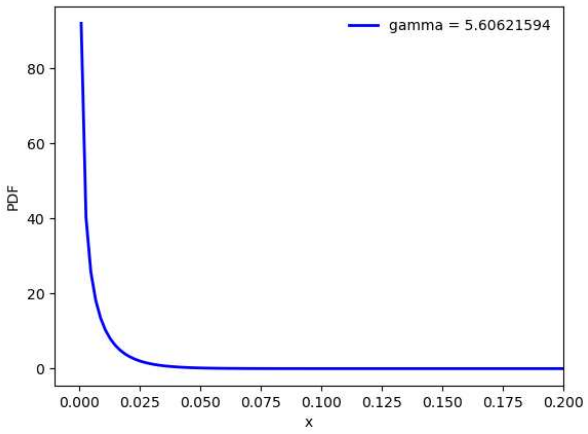
Parametar γ upravlja oblikom krivulje, dok su μ i β parametri razmjera i lokacije. Na slici 5.2 su redom prikazane vrijednosti parametra γ pa μ i β. Na slici 4.3. je prikaz promjene funkcije gustoće vjerojatnosti ovisno o parametru γ.



Sl. 4. 3. Funkcija gustoće vjerojatnosti Birnbaum-Saundersove distribucije u ovisnosti o parametru γ

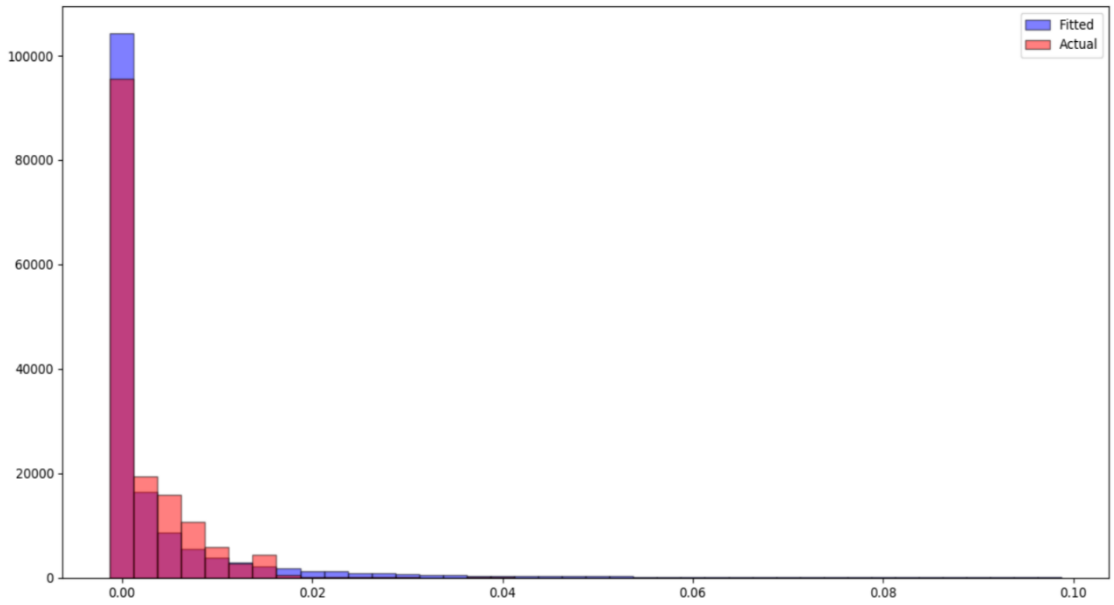
Teoretski gledano, distribucija je definirana da modelira ispade nastale zbog pukotina u nekom materijalu nastale zbog trošenja materijala. Nad materijalom se repetitivno izvodi nekakav stres, rad koji troši taj materijal. Za svaki ciklus stresa pukotina se povećava za određeni iznos dok ne dođe do kritične točke koja uzrokuje ispad. Usporedno s ispadima nastalim zbog pukotina u materijalu, može se poistovjetiti sljedeći primjer scenarija u telekomunikacijskoj mreži. Paketi koji se šalju kroz mrežu su u fizičkom sloju OSI sustava zapravo struja koja putuje bakrenom žicom. Ta struja troši materijal dok ne dođe do pogrešnog prijenosa bita, zbog čega je potrebna retransmisija paketa zbog koje se povećava međudolazno vrijeme paketa unutar TCP sjednice.

Na slici 4.4 možemo vidjeti funkciju gustoće vjerojatnosti s dobivenim parametrima. Slika prikazuje povezanost dobivene funkcije gustoće vjerojatnosti s histogramom prilagođene funkcije.



Sl. 4. 4. Funkcija gustoće vjerojatnosti Birnbaum-Saundersove distribucije dobivenih parametara

Za konfiguraciju K2 rezultati izvođenja skripte su prikazani na slici 4.5 i 4.6



Sl. 4. 5. Histogram stvarnih podataka i histogram podataka dobivene distribucije druge konfiguracije

```
Distribution name for time between packets: gengamma
Parameters: (1.0488652415261612, 0.42249460518830745, 9.999999999999997e-07, 0.0017395645942054313)
```

Sl. 4. 6. Ime dobivene distribucije te njeni parametri za drugu konfiguraciju

Kod histograma, na osi apscisa se nalazi vrijeme u sekundama (t[s]). Na osi ordinata se nalazi broj paketa u tom intervalu. Ime dobivene distribucije je gengamma, što u scipy biblioteci predstavlja generaliziranu gamma distribuciju.

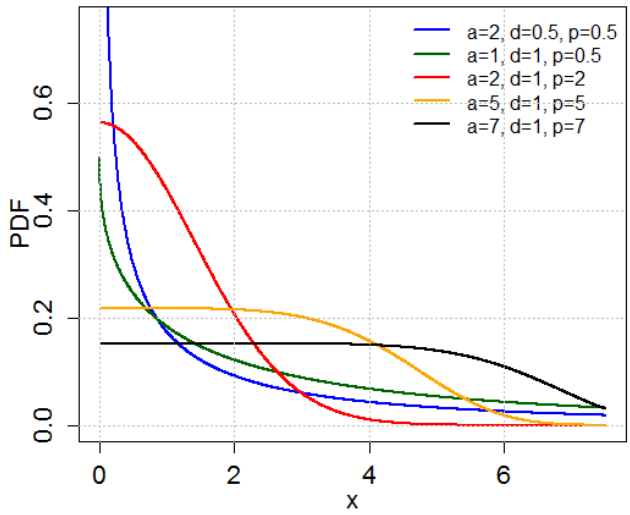
4.1.2 Generalizirana gamma distribucija

Generalizirana gamma distribucija je kontinuirana distribucija vjerojatnosti s tri parametra. Ova distribucija je zapravo generalizacija gamma distribucije s dva parametra. Iz ove distribucije nastaju druge, poznatije distribucije, kao npr. eksponencijalna, Weibullova te naravno gamma distribucija. I ova distribucija se u statistici koristi za analizu očekivanog vremena koje treba proći do nekog događaja, kao što je greška unutar nekog sustava ili smrt kod promatranja bioloških organizama.

Funkcija gustoće vjerojatnosti ove distribucije je sljedeća:

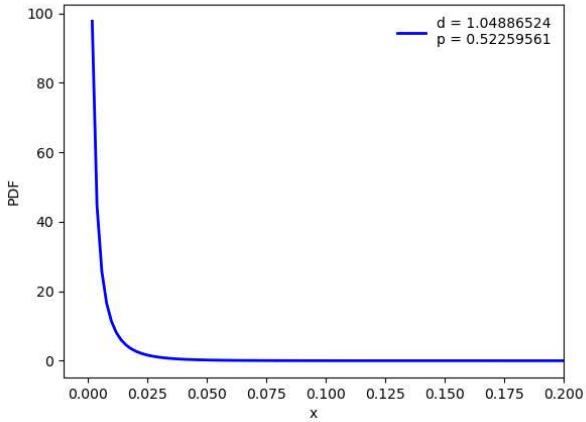
$$f(x;a,d,p) = \frac{(p/a^d)x^{d-1}e^{-(x/a)^p}}{\Gamma(d/p)}$$

Parametri d i p su strogo pozitivni parametri oblika funkcije. Parametar a je parametar razmjera. Γ(a) predstavlja gamma funkciju s parametrom a. Na slici 5.5 su redom prikazane vrijednosti parametara d, p, a te parametra lokacije koji nije prikazan u gornjoj formuli. Na slici 4.7 su prikazane različite funkcije gustoće razdiobe ove distribucije u ovisnosti o parametrima a, d i p.



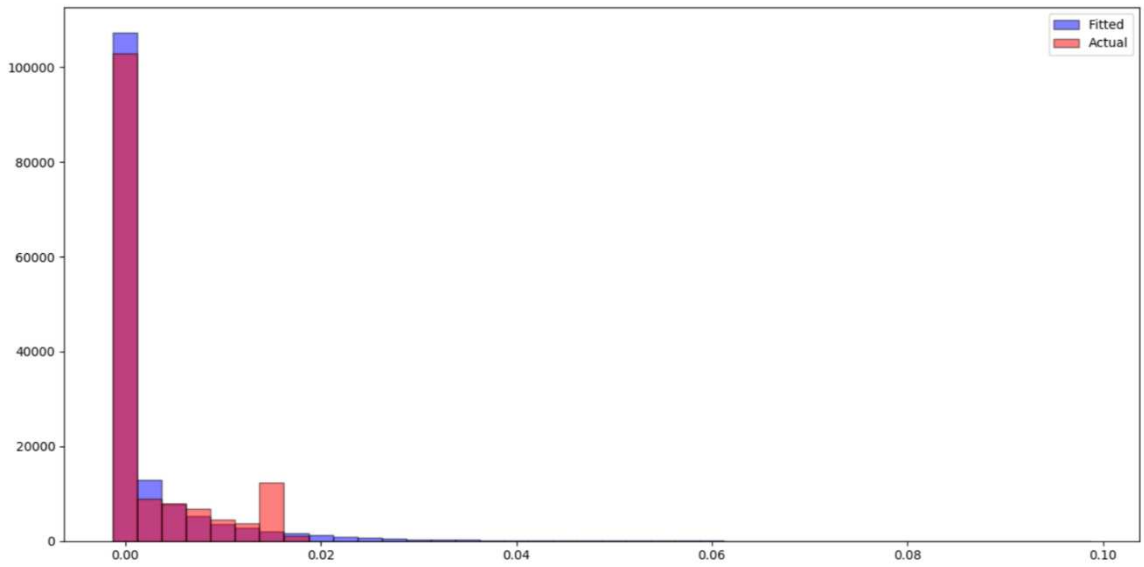
Sl. 4. 7. Funkcija gustoće vjerojatnosti generalizirane gamma distribucije u ovisnosti o parametrima a, d i p

Na slici 4.8 možemo vidjeti funkciju gustoće vjerojatnosti s dobivenim parametrima. Slika prikazuje povezanost dobivene funkcije gustoće vjerojatnosti s histogramom prilagođene funkcije.



Sl. 4. 8. Funkcija gustoće vjerojatnosti generalizirane gamma distribucije dobivenih parametara

Za konfiguraciju K3 rezultati su prikazani na slici 4.9 i 4.10.



Sl. 4. 9. Histogram stvarnih podataka i histogram podataka dobivene distribucije treće konfiguracije

```
Distribution name for time between packets: chi
Parameters: (0.32269215563379294, 9.999999999999997e-07, 0.01115050757066154)
```

Sl. 4. 10. Ime dobivene distribucije te njeni parametri za treću konfiguraciju

Kod histograma, na osi apscisa se nalazi vrijeme u sekundama (t[s]). Na osi ordinata se nalazi broj paketa u tom intervalu. Ime dobivene distribucije je chi, što u scipy biblioteci predstavlja chi distribuciju. S histograma se može vidjeti poprilično dobro poklapanje u gotovo svim intervalima međudolaznih vremena, osim u jednom intervalu gdje stvarni podaci imaju nagli skok u padajućem kontinuitetu količine podataka.

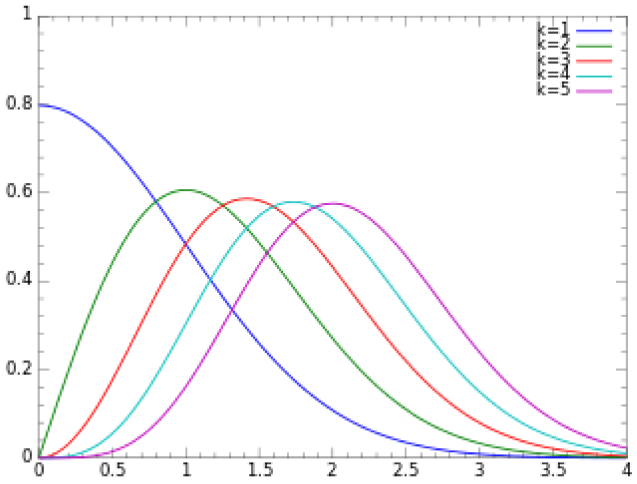
4.1.3 Chi distribucija

Chi distribucija je kontinuirana distribucija vjerojatnosti pozitivnih korijena sume kvadrata skupa nezavisnih slučajnih varijabli koje su generirane iz standardne normalne razdiobe. Drugim riječima, to je distribucija euklidskih udaljenosti slučajnih varijabli od njihovih izvornih vrijednosti. Chi distribucija ima jedan parametar, $k > 0$, koji definira broj stupnjeva slobode. Poznati primjeri chi distribucije su Rayleigh-ova distribucija ($k = 2$) i Maxwell-Boltzmann-ova distribucija brzina molekula u idealnom plinu ($k = 3$).

Funkcija gustoće vjerojatnosti ove distribucije je sljedeća:

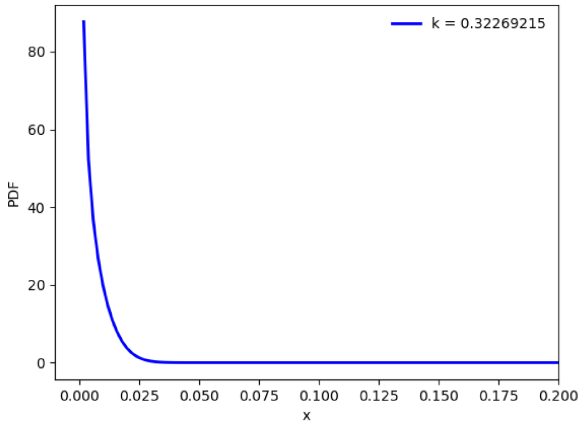
$$f(x,k) = \frac{1}{2^{k/2-1}\Gamma\left(k/2\right)} x^{k-1} \exp\left(-x^2/2\right)$$

gdje k predstavlja proizvoljan parametar, odnosno broj stupnjeva slobode ako govorimo o pozitivnom cijelom broju. Γ(k/2) predstavlja gamma funkciju s parametrom k. Na slici 5.8 prva vrijednost predstavlja parametar k, dok druge dvije vrijednosti predstavljaju parametar razmjera i parametar lokacije. Na slici 4.11 su prikazane funkcije gustoće razdiobe ove distribucije za različite vrijednosti parametra k.



Sl. 4. 11. Funkcija gustoće vjerojatnosti chi distribucije u ovisnosti o parametru k

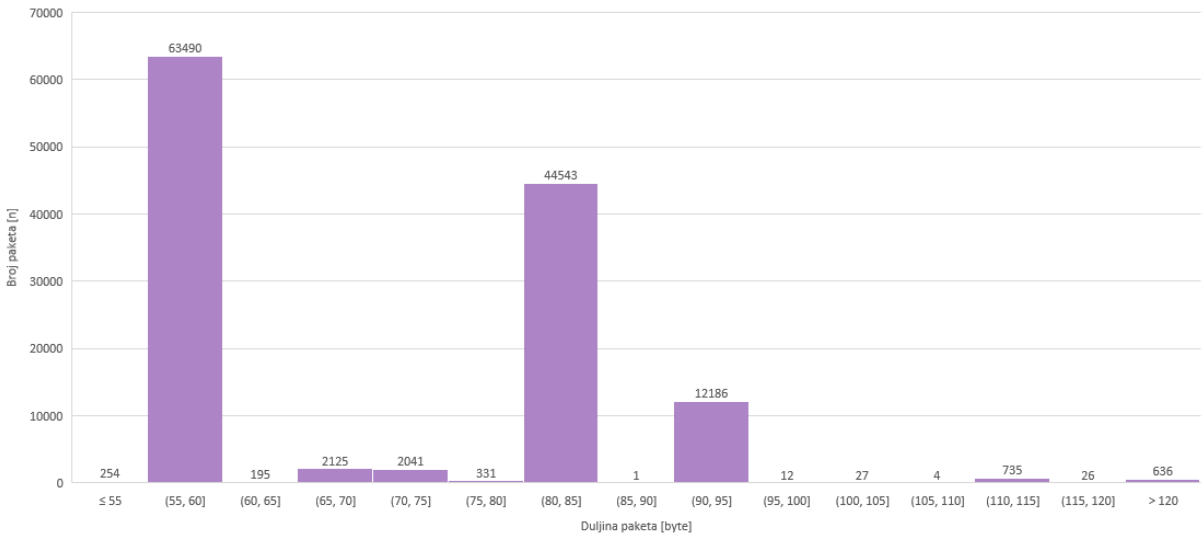
Na slici 4.12 možemo vidjeti funkciju gustoće vjerojatnosti s dobivenim parametrima. Slika prikazuje povezanost dobivene funkcije gustoće vjerojatnosti s histogramom prilagođene funkcije.



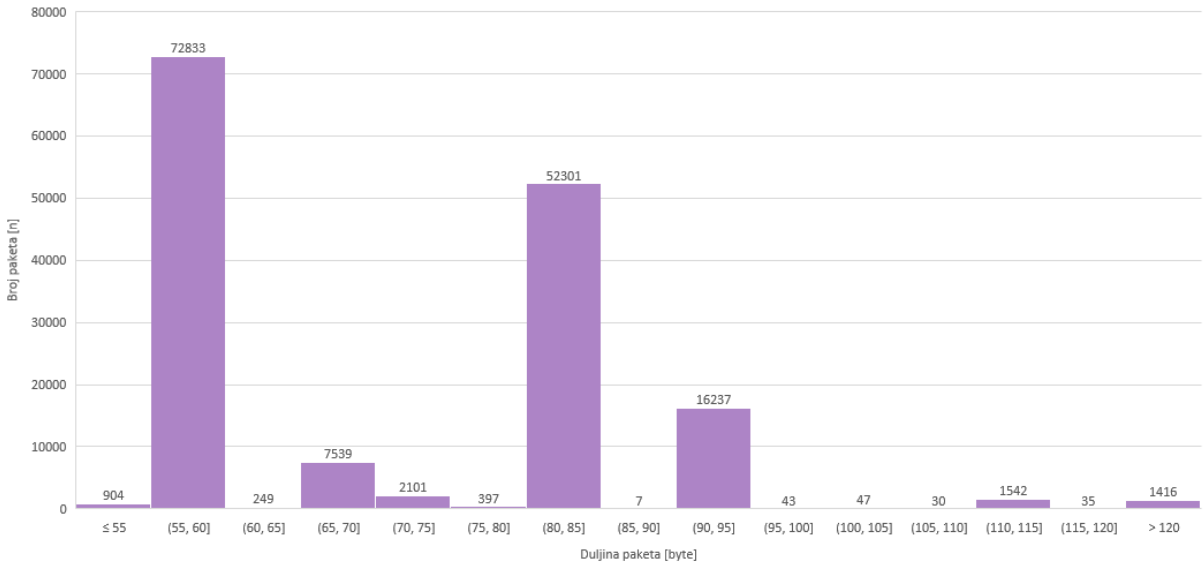
Sl. 4. 12. Funkcija gustoće chi distribucije dobivenih parametara

4.2 Duljine paketa

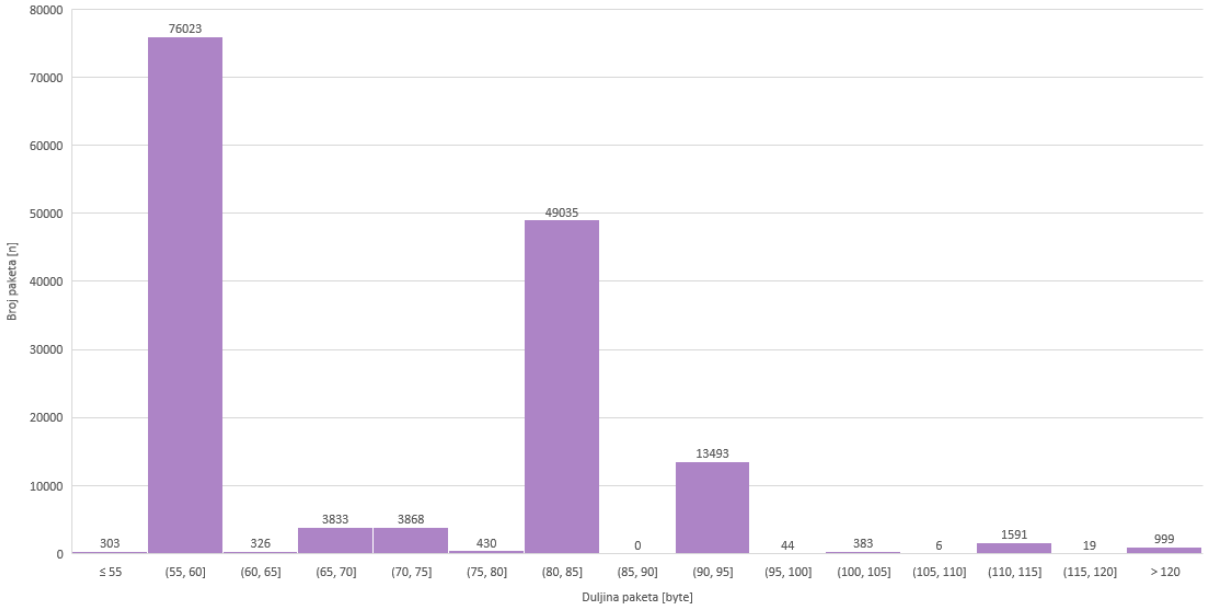
U nastavku su prikazani histogrami duljina paketa za sva tri eksperimenta. Na slici 5.10 prikazan je histogram za eksperiment u kojem su oba korisnika spojena žicom. Na slici 5.11 prikazan je histogram za eksperiment u kojem su oba korisnika spojena bežično. Na slici 5.12 je prikazan histogram za eksperiment u kojem je jedan korisnik spojen žicom, a drugi bežično.



Sl. 4. 13. Histogram duljine paketa za eksperiment gdje su oba korisnika spojena žicom



Sl. 4. 14. Histogram duljine paketa za eksperiment gdje su oba korisnika spojena bežično



Sl. 4. 15. Histogram duljine paketa za eksperiment gdje je jedan korisnik spojen žicom, a drugi bežično

Kod sva tri histograma vidimo sličnost, postoji velik broj paketa u određenim intervalima veličina paketa. Budući da su podaci poslanih paketa šifrirani, ne možemo sa sigurnošću utvrditi zašto se šalju ovakvi paketi. Međutim, možemo pretpostaviti da programski paket Mirror koji se koristi u ovom projektu u svrhu umreživanja korisnika u virtualni svijet šalje kontrolne i naredbene pakete u određenim vremenskim isječcima. Drugim riječima, ovaj programski paket ne čeka da se segment napuni do maksimalne veličine segmenta (engl. *Maximum Segment Size* – MSS) transportnog sloja, u ovom slučaju TCP, nego u određenim vremenskim intervalima šalje segmente. Kontrolni paketi služe kako bi se na klijentskim instancama sinkronizirali objekti i radnje koje se događaju i virtualnom okruženju. Naredbeni paketi služe kako bi klijentska instanca zatražila od serverske instance mogućnost neke radnje, kao npr. stvaranje čarolije.

Zaključak

Nadogradnjom umrežene virtualne igre „Wizard wars“ dobivena je interesantna, te pomalo izazovna višekorisnička računalna videoigra. Kreiranom agentu su osim funkcionalnosti praćenja i napada dodane i animacije kretanja i stvaranja čarolije, što je cjelokupno učinilo videoigru vizualno ugodnijom.

Nakon spajanja u privatnu virtualnu mrežu, nad proširenom videoigrom, pomoću dva korisnika i njihovih udaljenih računala, provedeni su eksperimenti za tri različite mrežne konfiguracije.

Za prvu konfiguraciju, dobivena je Birnbaum-Saundersova distribucija, za drugu generalizirana gamma distribucija, te za treću konfiguraciju Chi distribucija. Navedene distribucije vraćene su kao rješenje analize s njihovim parametrima razmjera, lokacije i oblika. Iz rješenja se može primijetiti kako su distribucije međusobno slične, te im je zajednički oblik pada sličan kao kod eksponencijalne funkcije. Promatranjem međudolaznih vremena za sve tri konfiguracije može se zaključiti da većina paketa ima međudolazna vremena niskih vrijednosti, te za vrijednosti većih međudolaznih vremena broj paketa strmo pada.

Što se tiče veličine paketa, uočeno je da se koristi mehanizam pri kojem se ne čeka da se paket napuni do maksimalne jedinice prijenosa, nego se šalje više manjih paketa, što također pridonosi kontinuiranoj komunikaciji.

Promjenom mrežnih konfiguracija nije dolazilo do degradacije iskustvene kvalitete prilikom igranja igre.

Literatura

[1] Maček D., Šimunović D., *Izrada višekorisničke video igre.* Diplomski projekt. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2020.

[2] Unity Technologies, Building a NavMesh, 5. ožujka 2020., *Unity Manual*, https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html, 10. travnja 2020.

[3] Unity Technologies, NavMeshAgent, 5. ožujka 2020., *Scripting API*, https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html, 10. travnja 2020.

[4] Grozaj, M. Primjena algoritama traženja puta u grafu pri razvoju jednostavne 3D videoigre. Završni rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2020.

[5] LogMeIn, Inc., LogMeIn Hamachi, 2017., *LogMeIn Hamachi Getting Started Guide*, https://documentation.logmein.com/documentation/EN/pdf/Hamachi/LogMeIn_Hamachi_GettingStarted.pdf, 7.5.2020.

[6] The Wireshark team, Introduction, 2017., *Wireshark User's Guide*, https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html, 8.5.2020.

[7] Stephanie Glen, Kolmogorov-Smirnov Goodnes of Fit Test, 5. srpnja 2020., *Statistics How To*, https://www.statisticshowto.com/kolmogorov-smirnov-test/, 15.5.2020.

[8] The SciPy community, Statistical functions (scipy.stats), 19. prosinca 2019, *SciPy v1.4.1 Reference Guide,* https://docs.scipy.org/doc/scipy/reference/stats.html, 2.6.2020.

[9] Leiva, V. The Birnbaum-Saunders Distribution: Chapter 1 Genesis of the Birnbaum-Saunders Distribution, Chapter 2 Characterizations of the Birnbaum-Saunders Distribution. Prvo izdanje. Sjedinjene Američke Države: Academic Press, 2015.

[10] 30. svibnja 2020., *Generalized gamma distribution.*

https://en.wikipedia.org/wiki/Generalized_gamma_distribution, 2.6.2020.

[11] 25. veljače 2020., Chi distribution,

https://en.wikipedia.org/wiki/Chi_distribution, 2.6.2020.

[12] Chen K-T, Huang P, Lei C-L. Game traffic analysis: An MMORPG perspective. Computer Networks. 50/2006, 2006, 3002-3023.

[13] Šimunović D., Analiza mrežnog prometa pri interakciji dvaju korisnika u višekorisničkoj 3D videoigri. Diplomski rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2020.

Sažetak

U ovom diplomskom radu višekorisnička umrežena videoigra „Wizard wars“ nadograđena je kreiranjem računalno-generiranih likova koristeći pokretačku platformu za igre Unity. Provedeni su eksperimenti s tri različite mrežne konfiguracije, pri čemu su računala krajnjih korisnika bila međusobno povezana putem putem virtualne privatne mreže kroz Internet. Mrežni promet tijekom igranja snimljen je s pomoću alata Wireshark. Iz snimljenog prometa izdvojena su međudolazna vremena i duljine paketa protokola Transmission Control Protocol (TCP) za daljnju analizu. Nadalje je provedena identifikacija distribucije slučajne varijable te su identificirane razdiobe za međudolazna vremena. Također su prikazani histogrami izračunatih distribucija i stvarnih podataka. Za sve tri mrežne konfiguracije uspoređeni su histogrami dobivenih duljina paketa.

Ključne riječi: višekorisnička videoigra, računalno-generirani likovi, analiza mrežnog prometa, distribucija slučajne varijable, međudolazno vrijeme, duljina paketa

Summary

In this master's thesis, a multiplayer videogame „Wizard wars“ based on Unity game engine has been upgraded by adding non-player characters. Experiments were conducted in three network configurations, in which the connection between the end-user computers was established through a virtual private network across the Internet. Network traffic was captured by using Wireshark. From the captured traffic, mean Transmission Control Protocol (TCP) packet length and interarrival time were extracted for further analysis. For the packet interarrival time, the random variable distribution was identified, and histograms were plotted that show the identified distribution against the actual data. For the packet length, histograms were compared for all three network configurations.

Key words: multiplayer videogame, non-player character, network traffic analysis, random variable distribution, interarrival time, packet length