Izrada videoigre pucanja za iOS u programskom alatu Unity

Taras, Bruno

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Bruno Taras

IZRADA VIDEOIGRE PUCANJA ZA iOS U PROGRAMSKOM ALATU UNITY

DIPLOMSKI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Bruno Taras

Matični broj: 44853/16–R

Studij: Poslovni sustavi

IZRADA VIDEOIGRE PUCANJA ZA iOS U PROGRAMSKOM ALATU UNITY

DIPLOMSKI RAD

Mentor/Mentorica:

Doc. Dr. sc. Mladen Konecki

Varaždin, ožujak 2023.

Bruno Taras

Izjava o izvornosti

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

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema rada je izrada akcijske pucačke mobilne igre za jednog igrača, koristeći alat Unity. Ovaj rad se bavim svim fazama izrade pucačke mobilne igre, te će sama pažnja biti usmjerena na inspiraciju za temu, funkcionalnosti te programski kod. Igra je namijenjena za jednog igrača, koristeći mobilni uređaj kojeg pokreće operativni sustav iOS. Igra se odvija u 3D perspektivi, kao i većina akcijskih pucačkih igara, pogled kamere je iz trećeg lica. Rad prolazi kroz širok spektar dijelova razvoja same igre, stoga neće prolaziti kroz cjelokupan proces izrade, već će se dati prednost elementima koji se odnose na samu mehaniku igre, njene funkcionalnosti te same implementacije istih. Rad je namijenjen osobama koji se zanimaju za izradu mobilnih igara.

Ključne riječi: igra; objekt; C#; Unity; izrada; pucanje; programiranje; animiranje; igrač; perspektiva; lik; protivnik; animacije;

Sadržaj

1. Uvod

Ovaj rad se bavi razvojem mobilne akcijske pucačke igre iz perspektive trećeg lica. Opisuje razvojna okruženja u kojima je sama igra izrađena, poput Unity alata, Visual Studio Code alata itd. Proces izrade ne bavi se izradom samih grafičkih modela poput okruženja, igrača, protivnika i sl., no bavi se izradom animacija, skripti, grafičkog sučelja te funkcionalnostima same igre. Kao što je već napomenuto igra je akcijskog žanra, napravljena po uzoru na razne akcijske mobilne igre s kojima sam se susretao, no rad će se u nastavku više baviti sličnim igrama i njihovim karakteristika.

Sama zainteresiranost i korištenje mobilnih igara u današnje vrijeme svakim danom sve više i više raste. Mobilne igre i aplikacije postale su prilika velikim korporacijama za privlačenje pozornosti samog korisnika. Također jedan od razloga koji bi mogao objasniti takvo ponašanje društva je sama dostupnost. Danas korisniku nije potrebno računalo, koje je na fiksnom mjestu, kako bi bio u mogućnosti igrati igre, već se korisnici okreću mobilnim igrama jer ih mogu igrati bilo kada na bilo kojem mjestu, dok se čeka u redu kod doktora, dok se korisnik duži period koristi javnim prijevozom, itd. Osobno kod mene je to jedan od glavnih razloga zašto sam se okrenuo mobilnim igrama, fleksibilnost i dostupnost u bilo koje vrijeme sa također velikim izborom za odabir igre željenog žanra.

Do sada se nisam imao prilike koristiti Unity alatom, te sam ovo prepoznao kao odličnu priliku za upoznati se sa istim, također donošenju odluke pripomogla je sama podrška zajednice koja koristi Unity alat.

Za izradu jednostavnih grafičkih modela poput gumbova, pozadina i sl. korišten je Adobe Photoshop alat. Također korištene su i web stranice koje omogućuju preuzimanje besplatnih asseta (modeli i sl.), kojima je uštađeno samo vrijeme izrade igre.

Kao Back-end developeru, sa radnim iskustvom većim od 1 godine, vjerujem da će mi izrada ovoga rada donijeti niz novih vještina kao što su primjena ideje, realizacija željenih funkcionalnosti te samo razmišljanje, uz to bavljenje samim grafičkim dijelom s čime se ne susrećem često.

2. Metode i tehnike rada

S obzirom na temu rada za početak je izrađen praktični dio, odnosno sama igra, te je završetkom iste i nakon njenog testiranja započet proces dokumentiranja. Izrada igre je realizirana postepenim dovođenjem iste do granice sa kojom ću biti osobno zadovoljan. U potpunosti su dokumentirane sve funkcionalnosti aplikacije koje autor smatra važnima.

Početak izrade igre fokusirao se na glavnu funkcionalnost same igre, a to je glavni zaslon. Na njemu se igrač ima mogućnost kretati sa svojim likom, te naravno sadrži funkcionalnosti poput korištenja oružja, skupljanja municije, itd. Nakon što je taj dio implementiran, proces izrade usmjerio se na izradu igre kako ju igrač doživljava kronološki, počevši sa glavnim izbornikom, odabirom načina igre, težine itd.

Glavni alat korišten za izradu igre je Unity, koji je temeljen na programskom jeziku C#. Unity zajednicu čini velik broj ljudi, čiji me doprinos privukao samom odabiru ovog alata pri izradi igre. Najvažniji grafički djelovi igre implementirane su od već gotovih grafičkih rješenja, a za izradu grafike koja je nedostajala korišten je Adobe Photoshop alat.

Za izradu programskog koda korišten je alat Visual Studio Code, a za njegovo pokretanje korišten je simulator u Unity alatu, te kasnije mobilni uređaj (iPhone XS). Također su za izradu praktičnog djela rada korišteni internetski izvori poput YouTube, StackOverflow, pisani izvori, službena Unity dokumentacija.

3. Alati

3.1. Unity

Unity je višeplatformski alat, razvijen u C++ programskom jeziku, korišten za razvoj video igara za različite platforme poput : Windows, macOS, Linux, Android, iOS itd. Alat je inicijalno izdan 2005. godine od strane tvrtke Unity Technologies te je do danas postao jedan od najpopularnijih alata u samoj industriji video igara. (Unity Technologies, bez dat)

Unity pruža kompletan skup alata potreban za razvoj igara, uključujući vizualni editor, Unity Scripting API (eng. Application Programming Interface), za čiju se komunikaciju koristi programski jezik C#, sama simulacija fizike (npr. pokreti igrača) te još mnogo toga. Navedeni skup alata omogućava developerima da lako i učinkovito izgrade složene i visokokvalitetne igre.

Što se tiče povijesti samog alata, Unity je od svog prvog prvog izdanja prošao niz značajnih proširenja i poboljšanja. Ranije verzije ovog alata bile su usmjerene prema developerima indie igara, , no kako je navedeni alat postajao sve popularniji, počeo je privlačiti veće tvrtke koje su se bavile razvojem igara. Unity je u posljednjih nekoliko godina proširio svoje mogućnosti prema razvoju proširenih i virtualnih stvarnosti što ga je učinilo univerzalnom platformom za širok raspon interaktivnih aplikacija.

Za izradu rada korištena je osobna besplatna odnosno osobna verzija ovog alata, koja je dovoljna za izradu ove igre s obzirom na njene zahtjeve. Postoje još plus, pro i enterprise verzije koje nude širok spektar mogućnosti, te su također namijenjene za profesionalnije svrhe. Dodatne mogućnosti koje ove licence nude vezane su uz dijagnostiku, analitiku, tehničku podršku, učenje „on-demand“ itd. (Unity Technologies, bez dat)

Neke od popluarnih mobilnih igara koje su izrađene pomoću Unity alata su Subway Surfers (prva igra koja je preuzeta više od milijardu puta na Google Play Storeu) i Fantasian (izdali kreatori Final Fanstasy serije, te su ovu kompleksnu igru uspješno optimizirali za mobilne uređaje izdajući verziju veličine od samo 4GB). Uz njih vrijedi spomenuti i tvrtku Tinytouchtales čije igre godišnje broje oko milijun preuzimanja u iOS i Android trgovinama (Unity Technologies, bez dat). Treba i napomenuti neke od popularnijih igara izrađenih za ostale platforme, a to su Rust (dostupne za macOS, Windows sustave te igraće konzole, u prva dva tjedna prodana u više od 150.000 primjeraka) (Wikipedia, 29.01.2023), Cities: Skylines (u prva 24 sata prodana u više od 250.000 primjeraka (Wikipedia, 29.01.2023) te Cuphead (u prva dva tjedna prodana u više od milijun primjeraka) (Wikipedia, 27.01.2023).

Unity se koristi i za ostale svrhe koje nisu povezane sa razvojem igara kao što su filmovi, konstrukcije, simulacije i slično. Kada govorimo o filmovima jedan od poznatijih projekata je ADAM: Episode 3 koji traje 9 minuta i izdan je 2017. godine, izradio ga je južnoafrički redatelj Neill Blomkamp. (We'll fix it in post, 08.07.2022.) Što se tiče simulacija rad „Production line simulation made with Unity and controlled by TwinCAT“, autora Jesse Reinikka, je prikazao kako se Unity može koristiti za izradu real-time aplikacija koje zahtjevaju snažno grafičko procesuiranje game enginea te prikazao sami proces izrade istoga.(Jesse Reinkka, 2019.) Također koristi se i u svrhu robotičkih testiranja ponuđenog rješenja što se tiče same izvedbe robota prije same implementacije u robota, a procjena izvedbe se odvija u smislu lokalizacije, planiranja pokreta i same kontrole. (Unity Technologies, bez dat) Također kod dizajniranja konstrukcija, industrije koriste real time 3D tehnologiju kako bi promjenili način operativnosti, kreiranja i dizajniranja zgrada. (Unity Technologies, bez dat)

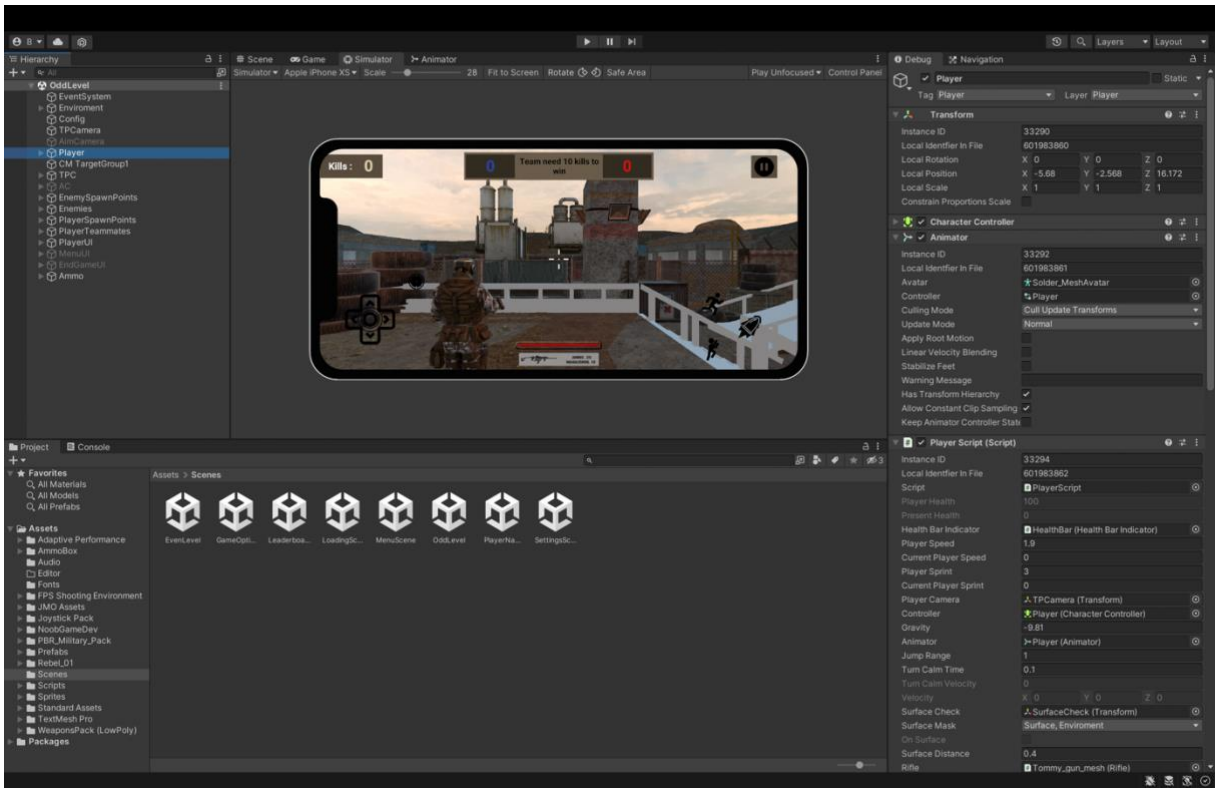
U zadnjih 6 godina broj višeplatformskih igara koje su objavljene je narastao za više od 200%. (Unity Technologies, bez dat) Tijekom 2021. godine zabilježeno je kako je skoro duplo više igara izrađeno pomoću Unitya nego tijekom 2020. godine, te je u razdoblju od 2021. do 2022. godine broj developera koji koristi Unity za izradu svojih rješenja narastao za 31%. Također tijekom 2021. godine, svi žanrovi su zabilježili rast prihoda od oglasa, no najviše je porastao u kartaškim, puzzle, avanturističkim i ležernim igrama, dok su najmanji rast prihoda zabilježii žanrovi poput akcijskih, RPG i arkadnih žanrova. (Unity Technologies, 13.03.2022.)

Neke od poznatijih tvrtki koje koriste Unity su : Amazon, Ubisoft, Cisco, Oracle itd. (Hired, bez dat) Unity je besplatan alat, koji ima veliku i aktivnu zajednicu, s podrškom za preko 20 platformi, što omogućava razvoj za više platformi bez korištenja različitih frameworka za svaku platformu posebno. Pruža pregršt značajki potrebnih za izradu 2D ili 3D igara. Uz sve navedeno sama dostupnost asset store olakšava pretragu potrebnih proširenja za izradu željene aplikacije, isto tako dostupnost pregršt online tutoriala olakšava korisnicima korištenje samog Unity alata. (Mind Inventory, 06.04.2022.)

U skladu s navedenim može se zaključiti kako je Unity imao i ima značajan utjecaj na razvoj industrije video igara, pružajući dostupan alat za developere svih razina iskustva. Potrebno je napomenuti kako Unity nije najbolji odabir alata za razvoj nekih aplikacija poput npr. mobilnog bankarstva.

3.1.1.Izgled i korisničko sučelje

Pri otvaranju projekta u Unity alatu prikazuje se sučelje koje se može vidjeti na slici 1. U centru samog sučelja prikazana je trenutna scena, a s lijeve strane se nalaze svi objekti koji se koriste na odabranoj sceni. Označavanjem jednog od objekata s desne strane u prozoru inspector se mogu vidjeti svojstva, postavke i komponente koje se koriste za odabrani objekt. Na donjoj polovici sučelja se nalazi file explorer pomoću kojeg možemo vidjeti sve datoteke sadržane u projektu, te ga koristimo u svrhu navigacije do željene datoteke u projektu, to može biti sama scena, asset, skripta itd. Dodatne funkcionalnosti Unity alata koji su korišteni no nisu prikazane na navedenoj slici će biti obrađene u nastavku.

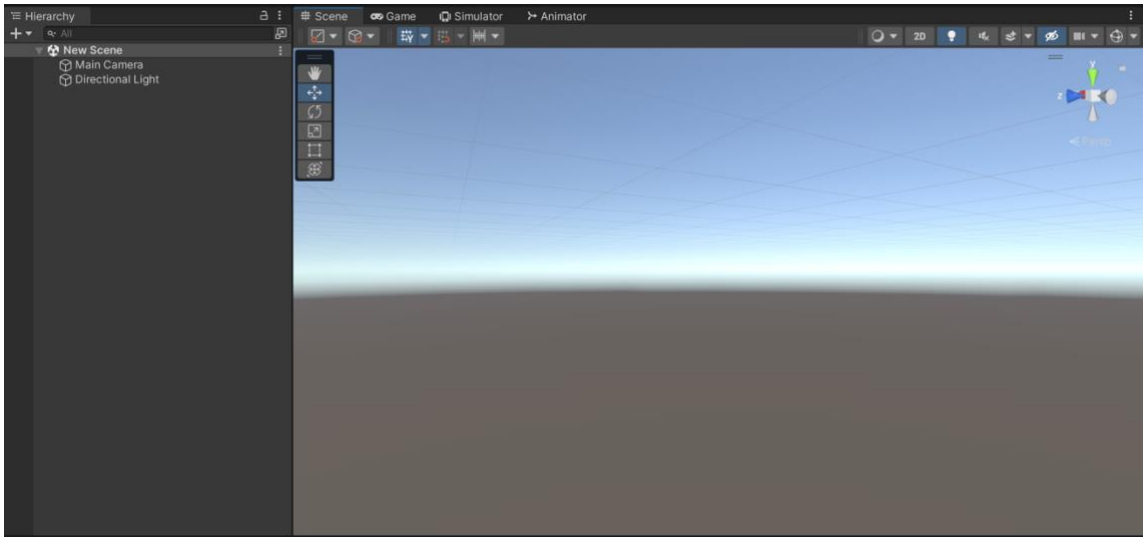


Slika 1 : Sučelje Unity Alata (Izvor: Vlastita produkcija)

3.1.1.1. Scene

Scena je mjesto gdje korisnik radi sa svojim sadržajem. Scena može sadržavati cijelu ili dio igre, za izradu jednostavne igre pojedinim korisnicima je dovoljno koristiti jednu scenu, dok za kompleksnije igre sa više razina, u kojima se razlikuju okruženja, likovi, prepreke, je

najbolje koristiti više scena. Broj scena po projektu nije ograničen. Prilikom kreiranja novog projekta, Unity alat otvara primjer scene koja sadrži kameru i svjetlo. (Unity Technologies, 03.02.2023.)



Slika 2 : Scene prozor (Izvor: Vlastita Produkcija)

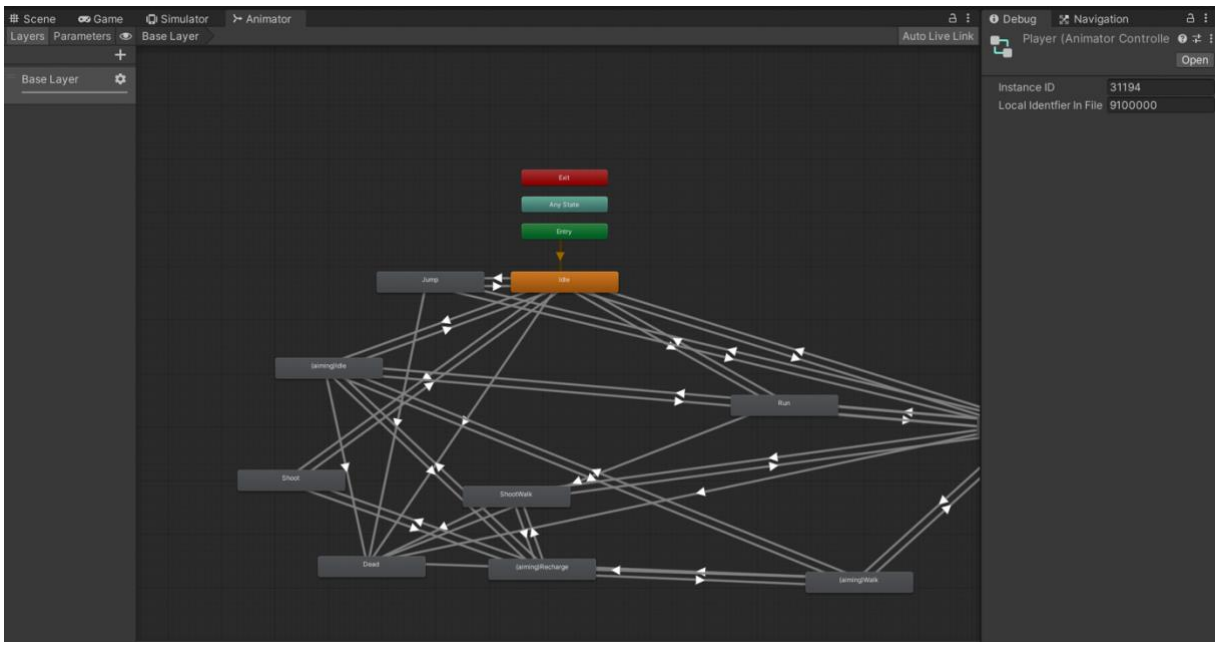
3.1.1.2. Simulator

Simulator prikazuje korisnikovu aplikaciju na simuliranom mobilnom uređaju, korisnik može odabrati željeni uređaj, kao što je u ovom projektu odabran Apple iPhone XS. Koristan je za povratnu informaciju kako igra izgleda na zaslonu, rezoluciji i orjentaciji samog uređaja.

Slika 3 : Simulator prozor (Izvor: Vlastita Produkcija)

3.1.1.3. Animator

Jedna od važnijih funkcionalnosti Unity alata je Animator. Animator nam služi za izradu animacija u samoj igri te je u njemu prikazan Animation Controller. Animation Controller je Unity asset koji je zadužen za logiku animiranja željenog GameObject-a. Unutar Animation Controllera nalaze se stanja (eng. State) i podstanja (eng. Sub-state) koji su povezani putem prijelaza (eng. Transitions). Stanja su zapravo prikaz animacijskih isječaka u samom Animatoru, dok prijelazi usmjeravaju tijek animacije iz jednog stanja u drugo stanje. Dok prijelazi orkestriraju logikom animacije, uvjeti kojima upravljaju parametri, koje je korisnik definirao, definiraju kada stanje animacije treba prijeći iz trenutnog u sljedeće stanje. Vrijednosti definiranih parametara mijenjaju se s obzirom na događaje (eng. Event) definirane u skriptama.



Slika 4 : Animator prozor (Izvor: Vlastita Produkcija)

Ako je u igri kretnja lika definirana unosima s tipkovnicem tada će te ulazne vrijednosti okidati (eng. Trigger) definirane događaje u skripti te na temelju toga mijenjati vrijednosti definiranih parametara te će proizvoditi animaciju povezani sa likom na temelju uvjeta postavljenih u samom Animation Controlleru.

3.1.1.4. Console

Console prozor prikazuje pogreške, upozorenja i druge poruke koje generira Unity. Za korisnika je ovaj prozor od velike koristi jer mu koristi za samo debugiranje igre, također korisnik može kreirati vlastite poruke (kao što je prikazano na slici) pomoću klase Debug. Sve što se ispiše u ovome prozoru također je zapisano i u log datoteku.

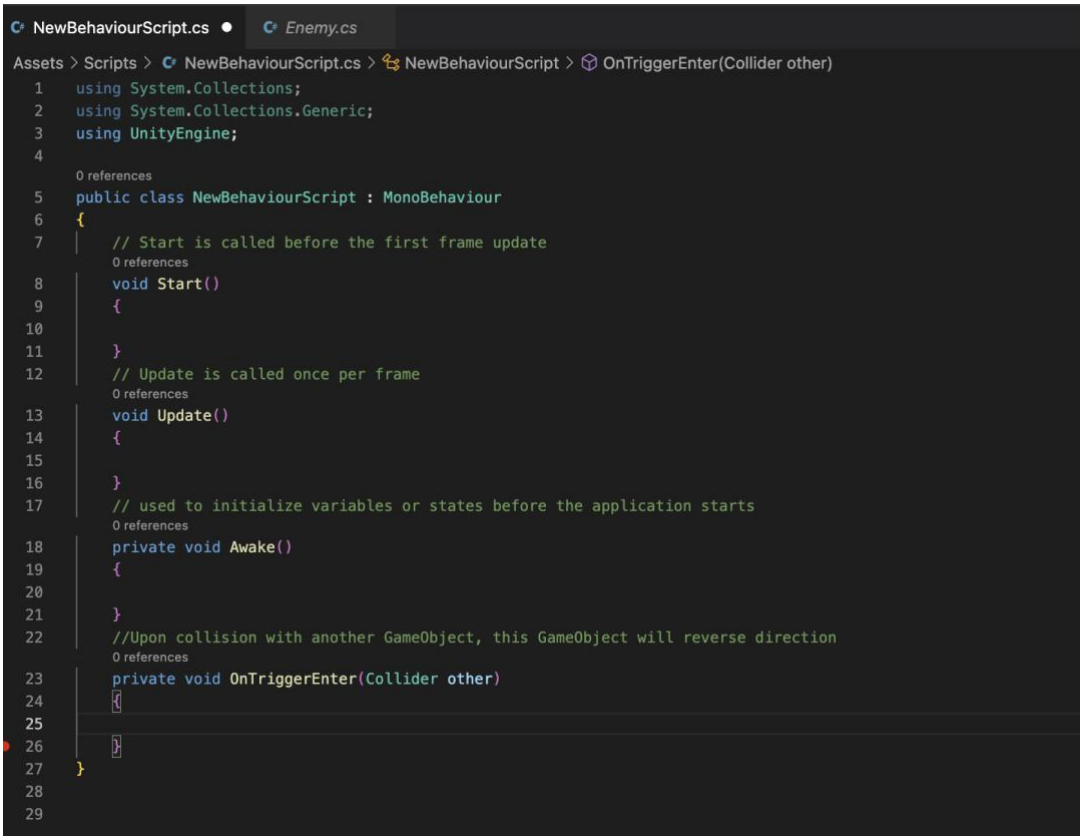
Slika 5 : Console prozor (Izvor: Vlastita Produkcija)

3.1.1.5. GameObject i Skripte

GameObject je osnovna klasa za sve objekte koje Unity može referencirati. Svaka javna varijabla koju korisnik definira u željenoj skripti, a proizlazi iz Object-a, prikazuje se u inspectoru kao drop target, odnosno inspector omogućuje korisniku postavljanje vrijednosti iz samog korisničkog sučelja. UnityEngine.Object je osnovna klasa za sve built-in Unity objekte.

Ponašanje GameObject-a kontroliraju komponente kojima je taj GameObject dodijeljen. Kako bi korisnik mogao u cijelosti implementirati svoje ideje oko igre, potrebna mu je skripta. Skripte u Unity-u omogućuju „okidanje“ željenih događaja u igri te mijenjanje svojstva komponenata u željenom trentuku.

Programski jezik koji se koristi tijekom definiranja skripti je, kao što je već napomenuto C#. Dvoklikom na određenu skriptu u file exploreru otvara se željeni alat za pisanje programskog koda. Zadani alat je Visual Studio, ali korisnik ima mogućnost definiranja željenog alata (Unity -> Preferences -> External Tools -> External Script Editor). Za izradu ove igre, alat za pisanje skripti koji je odabran je Visual Studio Code.



Slika 6 : Prikaz osnovnih metoda (Izvor: Vlastita Produkcija)

Kada je korisniku potrebna dodatna funkcionalnost, on kreira novu skriptu koja nasljeđuje MonoBehaviour klasu te nakon što su nove metode koje implementiraju željene funkcionalnosti definirane, korisnik ima mogućnost dodjeljivanja skripte određenom objektu kao komponentu. Skripta se može dodijeliti na neograničen broj objekata, što će se također moći primjetiti i u ovome projektu. (Unity Technologies, bez dat)

Start metoda poziva se neposredno prije prvog poziva Update metode. Start se poziva samo jednom u životnom vijeku (eng. Lifecycle) skripte. Start metoda neće se pozvati ukoliko u trenutku incijalizacije objekt, kojemu je skripta dodijeljenja, nije omogućen (eng. Enabled). Ovu metdou najčešće koristimo za inicijalizaciju određenih varijabli. (Unity Technologies, bez dat)

Ako je MonoBehaviour omogućen, Update metoda se poziva prilikom osvježenja vizualnog okvira (eng. Frame). Ovu metodu najčešće koristimo za provjeru određenih parametara, odnosno u nju upisujemo uvjete po kojima postavljamo vrijednosti određenim parametrima, kao primjer se može uzeti kretanje lika u igri, prilikom pritiska na određenu tipku za kretnju, komponenta se ponaša u skladu sa definiranim uvjetima. (Unity Technologies, bez dat)

Prilikom pokretanja scene, kod svih aktivnih objekata kojima je pridružena skripta u obliku komponente, poziva se metoda Awake. Ova metoda, kao i start, poziva se samo jednom prije bilo koje druge metode u životnom ciklusu, te također objekt kojemu je skripta sa ovom metodom dodijeljena mora biti aktivan kako bi se ista pozvala. (Unity Technologies, bez dat)

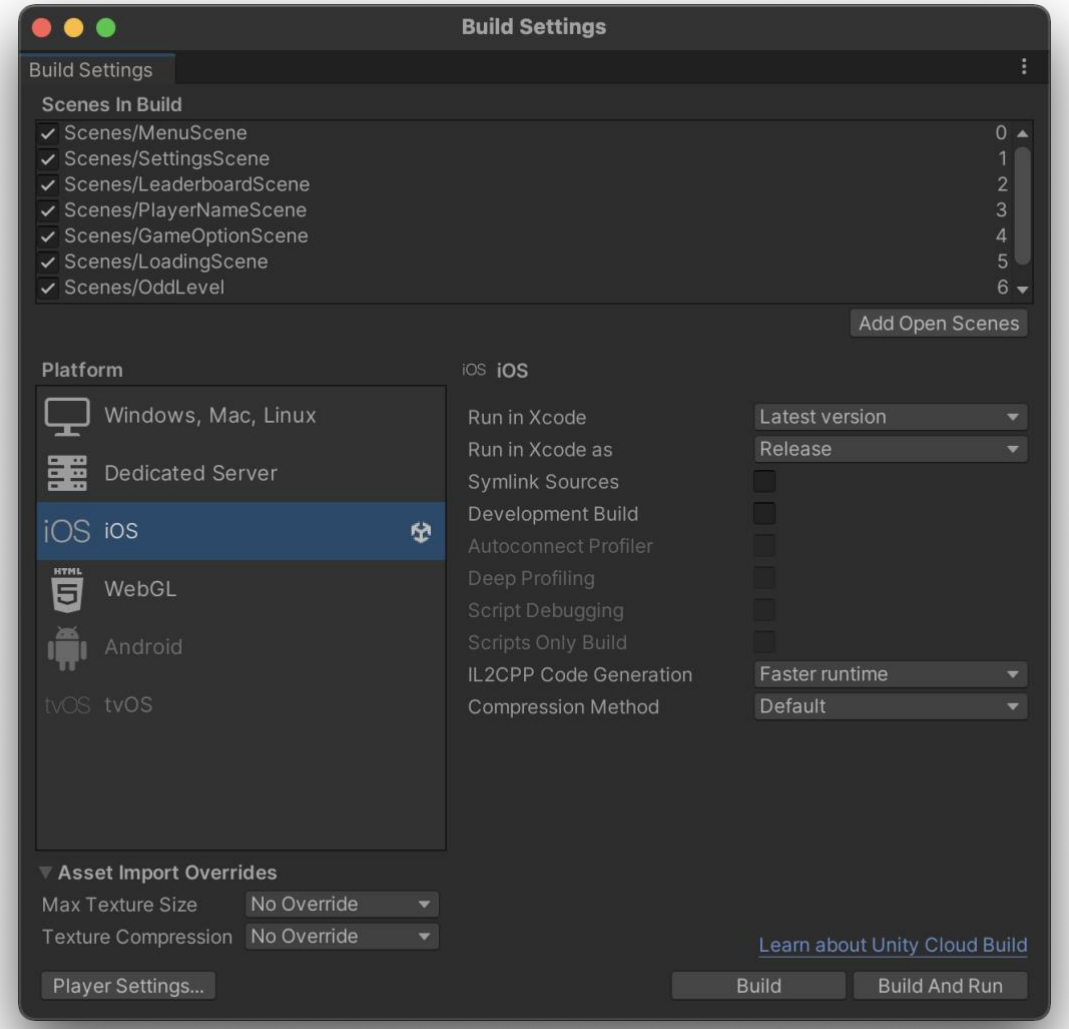
Kada se objekt „sudari“ sa drugim objektom poziva se metoda OnTriggerEnter. Kako bi se ova metoda uspješno pozvala, oba objekta moraju sadržavati komponentu Collider te najmanje jedan od objekata mora imati vrijednost isTrigger postavljenu na true.

3.1.1.6. Komponente

Komponenta (eng. Component) objekta jest bazična klasa za sve što se nalazi na objektu igre. Neke od osnovnih komponenata su: skripta (eng. Script), transformacija objekta (eng. Transform) , kolizija (eng. Collision) i kontroler igrača (eng. Character Controller).

3.1.1.7. Build

Prozor Build Settings omogućuje korisniku odabir platforme te prilagođavanje željenih postavki. U ovome prozoru korisnik ima mogućnost izvoza (eng. Export) izrađene igre u finalni proizvod. Na slici ispod vidi se struktura Build Settings prozora, na vrhu prozora nalaze se scene koje želimo uključiti tijekom izrade. U donjem lijevom djelu prozora korisnik ima mogućnost odabira platforme, dok na desnoj strani ima mogućnost podešavanja parametara koji su važni za sami izvoz. U ovome primjeru možemo vidjeti kako je željena platforma iOS sustav, za to nam je potreban instaliran Xcode alat koji je namijenjen za izradu iOS aplikacija. Nakon što je korisnik sve postavio može izvesti aplikaciju klikom na gumb „Build“ ili ju izvesti i odmah pokrenuti nakon što je izvedena klikom na gumb „Build And Run“.



Slika 7 : Build Settings prozor (Izvor: Vlastita Produkcija)

3.1.1.8. Unity Remote

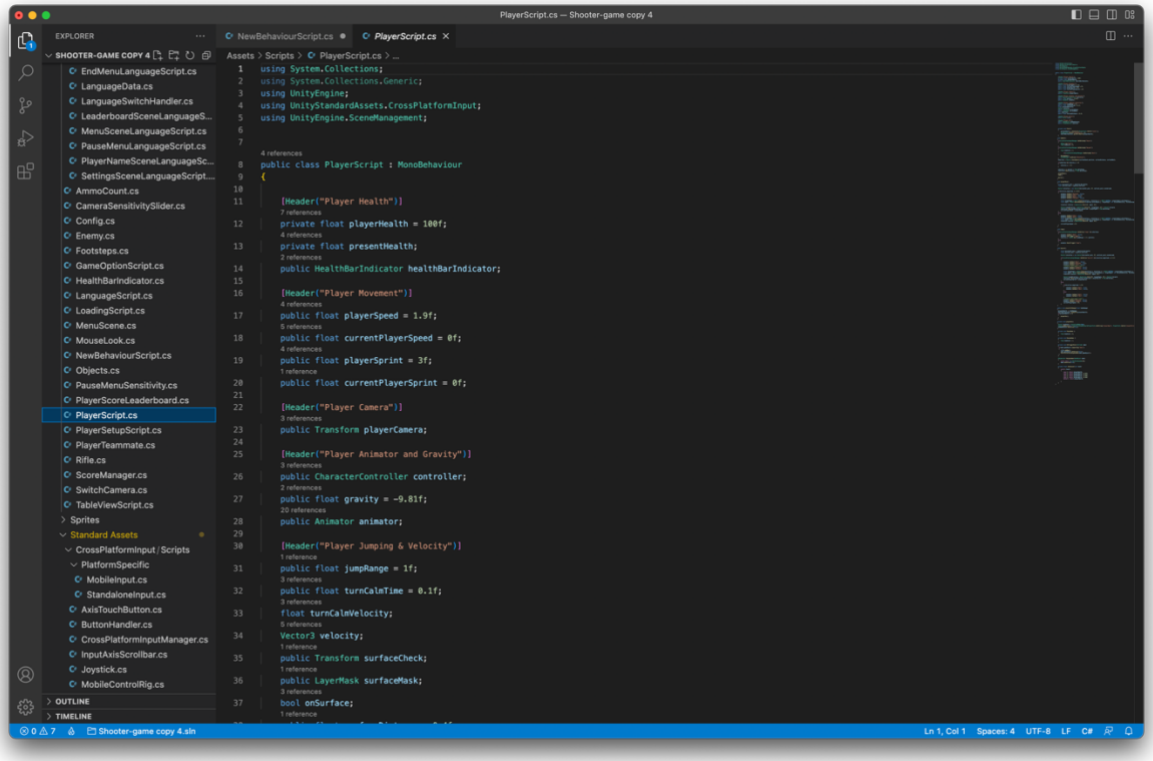
Unity Remote je aplikacija namijenjena za uređaje koje pokreću Android, iOS ili tvOS sustavi. Služi za povezivanje uređaja i Unity alata te tijekom pokretanja projekta u alatu prikazuje se scena i na povezanom uređaju, isto tako interakcije koje se odvijaju na uređaju se šalju nazad u alat. Ova aplikacija je korisna za testiranje projekta na željenim uređajima.

3.1.1.9. Unity Asset Store

Unity Asset Store je platforma za objavljivanje modela, animacija, audio datoteka kao i drugih asseta koji se mogu koristiti prilikom izrade Unity projekta. Za izradu ovog projekta također je korištena ova platforma, te će preuzeti asseti biti spomenuti u nastavku projekta. Na samoj platformi nalazi se pregršt besplatnih asseta kao i onih koje je potrebno platiti, koji znatno olakšavaju proces izrade igre.

3.2. Visual Studio Code

Ovaj alat je izdan od strane Microsofta, te je dostupan za macOS, Windows i Linux sustave. S obzirom kako ovaj alat ima bogat ekosustav ekstenzija, te također kako se radi o C# programskom jeziku, smatram kako je ovaj alat najbolje rješenje za pisanje programskog koda. Također jedna od prednosti ovog alata je provjera ispravnosti pisanja naredbi, formatiranje samog koda, te predlaganje dovršetka naredbi na temelju prvih upisanih znakova itd. (Microsoft, bez dat)



Slika 8 : sučelje Visual Studio Code

4. Pucačka igra iz perspektive trećeg lica (Third person shooter game)

Ovaj žanr karakterizira perspektiva kamere koja se razlikuje od perspektive samog lika igrača. Kamera je postavljena iza ili oko lika te ovakvom perspektivom korisnik može vidjeti tijelo lika što nije slučaj kada govorimo o pucačkim igrama iz perspektive prvog lica. Ovaj žanr je s vremenom postao jedan od najpopularnijih u game industriji.

4.1. Povijest

Razvoj računalnih igara započeo je u pojedinim svjetskim laboratorijima koji su se bavili računalima u kojima su igre razvijene kao predstavljanje mogućnosti novih aparatura. Prvom takvom igrom smatra se Tennis for two iz 1958. godine koja se igrala uz pomoć analognog računala te prikazivala na osciloskopu. Igra Pong predstavljena 1972. godine bila je nalik stolnomu tenisu, te je predstavlja jednu od prvih igara koje su se igrale na pristupačnim kućnim igraćim konzolama i na igraćim automatima. Tijekom 1980-ih pojavile su se i igre namijenjene osobnim računalima. (Enciklopedija, bez dat)

Slika 9 : "Tennis for two" (Izvor: https://en.wikipedia.org/wiki/Tennis_for_Two)

Prvo pojavljivanje igara iz perspektive trećeg lica dogodio se početkom 1980-ih godina, kada su arkadne igre počele koristiti 3D grafiku te su omogućili korisniku kontroliranje lika iz perspektive trećeg lica. Neke od prvih igara koje su se pojavlile na tržištu sa ovakvom perspektivom su Radar Scope (1979.), Atari's Tempest (1981.) i Tube Panic (1983.).

Međutim pucačke igre iz perspektive trećeg lica su tek 1990-ih godina počele razvijati svoju popularnost izdavanjem igara kao što su „Tomb Raider“ i „Resident Evil“, ove igre postavile su temelje za daljnji razvoj ovog žanra.

Slika 10 : Tomb Radier (lijevo) i Resident Evil (desno) (Izvor: Vlastita produkcija)

4.1.1.Primjeri

4.1.1.1. Grand Theft Auto 5

Grand Theft Auto 5 je akcijsko avanuristička igra koju je 2013. godine objavio Rockstar Games, to je sedmi nastavak Grand Theft Auto serije. Radnja je smještena u izmišljenoj državi San Andreas koja se nalazi u Južnoj Kaliforniji, a priča prati tri lika, umirovljenog pljačkaša Michaela De Santa, uličnog gangstera Franklina Clintona i trgovca drogom Trevora Phillipsa. Priča je usredotočena na pljačke, a mnoge misije uključuju pucanje i vožnje.

Igra se igra iz perspektive trećeg lica, a u igri je moguće kretanje pješice ili vozilom. Tijekom igre korisnik upravlja svim navedenim likovima, može mijenjati uloge između njih, tijekom i izvan misija. Grand Theft Auto Online omogućuje mrežni način igre za više igrača koji uključuje niz kooperativnih i natjecateljskih načina igre. (Wikipedia, bez dat) Igra se može igrati na svim igraćim konzolama te na Windows sustavima.

Slika 11 : GTA V Gameplay (Izvor: https://www.pinterest.com/pin/519954719453104686/)

4.1.1.2. Red Dead Redemption 2

Red Dead Redemption 2 je akcijsko avanturistička igra koju je 2018. godine objavio Rockstar Games. Radnja je smještena u izmišljenu rekreaciju američkog starog zapada 1899. godine. Priča se usredotočuje na život Arhura Morgana i njegovu poziciju u ozloglašenoj bandi Van der Linde, no također priča prati likove Dutch van der Linde, John Marston i Micaha Bella. Igra prati pad badne dok ih progone policajci, kolege iz bande i Pinkertonovi agenti.

Ova igra smještena je u svijet koji se sastoji od pet američkih regija koje korisnik može istraživati dok nastavlja sa pričom. Igra je dostupna za igraće konzole i Windows sustave. (Fandom, 03.12.2022.)

Slika 12 : RDR2 Gameplay (Izvor: https://www.polygon.com/2018/11/28/18113159/red-dead-redemption-2-rdr2-no-power-curve-problem-bad-gta5)

4.1.1.3. The Last Of Us

The Last Of Us je akcijsko avanturistička igra koju je 2013. godine objavio Sony Computer Entertainment. Korisnik upravlja likom imena Joel, krijumčara zaduženog za pratnju tinejdžerice Ellie kroz postakoliptične Sjedinjene Američke Države.

Igra se iz perspektive trećeg lica, a korisnici mogu koristiti vatreno i improvizirano oružje te također imaju sposobnost nevidljivosti za obranu od neprijateljski raspoloženih ljudi ili stvorenja koji su zaraženi mutiranom gljivicom. Ova igra također omogućuje mrežni način igre za više igrača koji uključuje niz kooperativnih i natjecateljskih načina igre. Ova igra namijenjena je za igranje na Playstation 3 i Playstation 4 konzolama.(Wikipedia, 01.02.2023)

Slika 13 : The Last of Us Gameplay (Izvor: https://www.thegamer.com/the-last-of-us-part-1-gameplay-reveal-combat-exploration/)

4.2. Karakteristike

Ovaj žanr karakterizira fokus na akciju i borbu, perspektivu kamere iz trećeg lica te niz alata (najčešće oružja) i sposobnosti koje korisnici trebaju koristiti kako bi porazili neprijatelja. Igre ovakvog žanra najčešće uključuju istraživanja, rješavanje zadataka, borbe te također veliko otvoreno okruženje kojim se korisnik odnosno igrač može kretati.

Osim navedenoga mnoge igre ovoga žanra imaju snažan naglasak na priču same igre koju čine složeni likovi i zapleti koji služe za pokretanje same radnje.

4.3. Osnovna ideja igre

Igra koja je izrađena za svrhu ovoga rada odvija se u 3D prostoru u kojem je igrač ograničen, u smislu kretanja, sa samim granicama mape. Kamera je u igri postavljena iz

perspektive trećeg lica. Glavni lik ove igre je vojnik koji se bori sa neprijateljima, te korisnik može birati želi li se boriti u timu protiv neprijatelja ili sam.

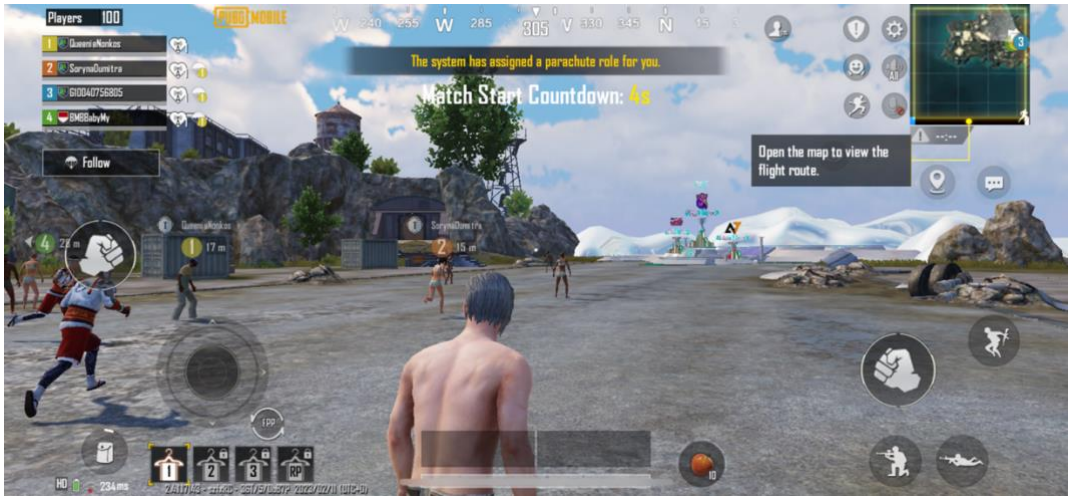
Cilj ove igre je proći svih 5 razina (eng. Level), u slučaju da budemo pobijeđeni od strane neprijatelja ili glavni lik umre, igra se prekida te korisnik ima mogućnost ponovnog igranja. Kako bi igrač uspješno prošao određenu razinu, njegov tim (koji čini glavni lik i njegovi suigrači ovisno o odabranom načinu igre) mora imati potreban broj smrti (eng. Kill) neprijatelja, isto tako ako neprijatelji ubiju glavnog lika ili prije dođu do definiranog broja smrti igračevog tima, igra za igrača prestaje. Igrač ima mogućnost odabira tri načina igre (Team, Single, Unlimited), te također tri opcije težine same igre (Easy, Medium, Hard) koji će biti detaljnije obrađeni u poglavlju koje će obrađivati samu mehaniku igre.

Tijekom igre igrač ima mogućnost skupljanja municije na mapi, te prilikom završetka same igre, rezultat igrača (broj smrti neprijatelja uzrokovane igračem) će biti spremljen na ljestvici rezultata, kojoj se može pristupi iz glavnog izbornika. Za igranje ove igre potreban je mobilni uređaj kojeg pokreće iOS operativni sustav.

4.3.1. Slični primjeri

4.3.1.1. PUBG MOBILE

PUBG Mobile je jedna od popularnijih mobilnih igara koja je dio Battle Royale žanra. Igra započinje sa 100 igrača koji dolaze na otok padobranom, te je cilj igre ostati posljednji. Igrači imaju mogućnost traženja oružja i opreme, eliminiranje drugih igrača kako bi napredovali. Igra broji više od milijun preuzimanja na App Storeu i Google Playu. (Apple App Store, bez dat)



Slika 14 : PUBG Mobile Gameplay (Izvor: Vlastita produkcija)

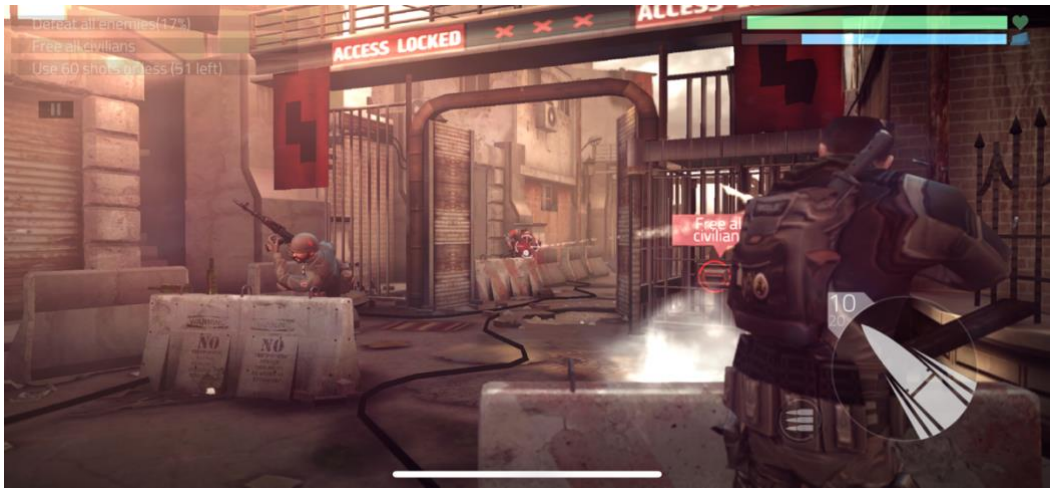
4.3.1.2. Free Fire

Free Fire je mobilna pucačina koju je 2017. godine objavila Garena. Ova igra je također dio Battle Royal žanra, no u ovome primjeru igra započinje sa 50 igrača, no cilj je ostao isti, kako bi pobjedio igrač mora ostati posljednji na mapi. Igra omogućuje igračima da igraju sami ili u timu. Ova je igra privukla veliki broj igrača te je postala jednom od najpopularnijih mobilnih igara u svijetu. (Apple App Store, bez dat)

Slika 15 : Free Fire Gameplay (Izvor: Vlastita produkcija)

4.3.1.3. Cover Fire: Gun Shooting Games

Cover Fire : Gun Shooting Games je akcijska pucačina u kkojoj igrači preuzimaju ulogu vojnika koji se bori protiv neprijateljskih snaga. Igra sadrži scenarije borbe i oružja, izazivajući igrače da koriste svoje taktičke vještine i strateško razmišljanje kako bi uspjeli. Igra je dostupna na App Storeu i Google Play Storeu. (Apple App Store, bez dat)



Slika 16 : Cover Fire Gameplay (Izvor: Vlastita produkcija)

4.4. Inspiracija / Motivacija

U prethodnom poglavlju spomenuti su slični primjeri s obzirom na osnovnu ideju igre te na te primjere možemo gledati kao na glavnu inspiraciju ovoga rada. Kroz godine sam se susreo najviše sa žanrovima poput FPS (eng. First person shooter) i TPS (eng. Third person shooter) stoga i ne čudi kako sam se odlučio za izradu igre ovog žanra. Prva pucačka igra koju sam imao prilike igrati je Return to Castle Wolfenstein, te sam se kroz godine više fokusirao na serije Call Of Duty, Counter Strike i Wolfenstein igara, no više sam vremena proveo igrajući mobilne igre, poput Call Of Duty Mobile, Fortnite, PUBG.

Navedene igre poslužile su mi kao primjeri pri izradi korisničkog sučelja dok sam modele igrača i okruženja odlučio preuzeti iz razloga što smatram da bi uložio puno vremena u izradu istih. Kretanje igrača po okruženju, ograničeno je granicama mape, stoga se da zaključiti kako su me ponajviše inspirirale igre poput Call Of Duty (koristeći multiplayer načinu rada) i Counter Strike. Povremenu pomoć pronašao sam u pojedinim vodičima i forumima, ali njih ne smatram inspiracijom za ovaj rad jer sam u tada već znao što i kako želim implementirati, te mi je navedena pomoć služila za dobro usmjeravanje samog procesa rada.

Slika 17 : Glavni dio igre (Izvor: Vlastita produkcija)

5. Mehanike igre

U ovom dijelu rada opisane su mehanike, pravila i ideje koje su ključne za rad i poznavanje igre.

5.1. Igrač

Kao što je već spomenuto igrača predstavlja vojnik, te se njime upravlja pomoću korisničkog sučelja prikazanog na zaslonu tijekom same igre. Njegovim kretanjima se upravlja pomoću kontrolera koji se nalazi u donjem lijevom kutu zaslona (pogledaj sliku 2.). Također uz kontroler se nalazi i tipka za ciljanje, a u donjem desnom kutu zaslona tipke za skakanje, sprint i pucanje iz oružja. U gornjem desnom kutu se nalazi tipka za pauzu. Neke od važnijih osobina igrača koje je potrebno spomenuti su : zdravlje (eng. Health), brzina (eng. Speed), skakanje (eng. Jump).

Brzina i skakanje se odnose na kretnje igrača, brzina definira brzinu kretanja igrača, promjenjiva je pritiskom na tipku sprint, no nakon što igrač pusti sprint tipku, brzina igrača se postavlja na početnu vrijednost, dok samo skakanje definira položaj igrača po y-osi, također, položaj igrača u odnosu na y-os promjenjiv je samo pritiskom na tipku za skok te se nakon toga položaj igrača u odnosu na y-os postavlja na početnu vrijednost. Zdravlje označava koliko još štete igrač može primiti prije kraja igre, a smanjuje se primanjem štete od neprijatelja. Početna vrijednost zdravlja definira se s obzirom na razinu u kojoj se igrač trenutno nalazi, ova značajka biti će opisana u nastavku rada. Kada je vrijednost zdravlja jednaka 0 igrač umre te igra počinje iznova.

Slika 18 : Igrač i korisničko sučelje (Izvor: Vlastita produkcija)

5.1.1. Puška

Puška je dio igrača, no također ima vlastite osobine, a neke od važnijih su: šteta koju puška nanosi (eng. Damage), domet hica puške (eng. Range), spremište za streljivo (eng. Mag) i ponovno punjenje (eng. Reloading). Početna vrijednost štete koju puška nanosi određuje se s obzirom na težinu igre koju je igrač odabrao, ova značajka biti će opisana u nastavku. Domet se odnosi na maksimalnu udaljenost do koje metak može doći, ako je neprijatelj unutar dometa, igrač mu hicom nanosi štetu, ova vrijednost se ne mijenja tijekom igre. Ponovno punjenje je sposobnost punjenja puške municijom nakon što je igrač ostao bez iste, vrijednost početnog broja spremišta za streljiva je fiksna, no može se mijenjati tako što igrač ima mogućnost skupljanja municije tijekom same igre.

5.2. Suigrač / Neprijatelj

Suigrač (eng. Teammate) i Neprijatelj (eng. Enemy) imaju iste osobine, te će u ovome djelu one biti opisane. Neprijatelja je uvijek četiri, neovisno o razini u kojoj se igrač nalazi, dok je suigrača uvijek tri, kako bi zajedno sa igračem činili tim od četiri, no ukoliko igrač odabere način igre „Single“ suigrači se neće pojavljivati.

Neke od važnijih osobina ovih likova su : zdravlje (eng. Health), šteta (eng. Damage), brzina (eng. Speed), vrijeme između pucanja (eng. Time between shooting), vidno polje (eng. Vision radius), domet pucanja (eng. Vision radius) i ponovno rođenje (eng. Respawn). Vrijednost zdravlja suigrača i neprijatelja inicijalno je postavljena na vrijednost 100, te se ona smanjuje s obzirom na štetu koju nanosi neprijatelj odnosno siugrač. Šteta koju suigrač nanosi je također fiksna postavljena vrijednost koja se ne mijenja, dok kod neprijatelja to nije slučaj, njihova vrijednost se postavlja s obzirom na razinu u kojoj se igrač nalazi, ova značajka će biti detaljnije obrađena u nastavku.

Vrijednost brzine je inicijalno postavljena te ona nema mogućnost promjene ni kod suigrača, a ni kod neprijatelja. Glavna zadaća vremena između pucanja je ograničiti pucanje suigrača i neprijatelja netom nakon što su zapucali. Vidno polje je zapravo radijus u kojem suigrač ili neprijatelj može vidjeti odnosno primjetiti neprijatelja odnosno suigrača, te nakon što je neprijatelj primjećen suigrač odnosno neprijatelj se počinje kretati prema njemu. Domet pucanja se odnosi na maksimalnu udaljenost do koje metak može doći, ako je neprijatelj unutar dometa, suigrač odnosno neprijatelj mu hicom može nanijeti štetu.

Osobina koju imaju i suigrač i neprijatelj, a ne postoji kod igrača, je sposobnost ponovnog rođenja (eng. Respawn), kada suigrač odnosno neprijatelj umre on se nakon nekog vremena ponovno pojavi, na definiranoj lokaciji mape, sa svim zadanim vrijednostima. Suigrači

i neprijatelji se nasumično kreću po mapi ili prema lokaciji neprijatelja te u trenutku dolaska do željene lokacije na kojoj se neprijatelj nalazi u vidnom polju i dometu pucanja, suigrač odnosno neprijatelj se prestaje kretati te započinje pucati.

Slika 19 : Prikaz suigrača (lijevo) i neprijatelja (desno)

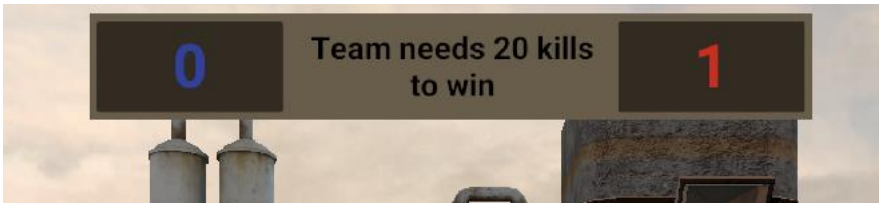
5.3. Objekti

Željenim objektima na mapi se može dodijeliti osobina zdravlja (eng. Health), te bi nakon što se ista dodjeli igrač imao mogućnost uništavanja tog objekta, incijalna vrijednost je postavljena na 100.

Slika 20 : Primjer objekta (Izvor: Vlastita produkcija)

5.4. Score Manager

Zadaća Score Managera je sadržavanje podataka o trenutnom rezultatu (broju ubojstava pojedinog tima), te na temelju rezultata određuje daljnji tijek igre. U slučaju da neprijatelji prvi dostignu zadani rezultat, igra za igrača završava, a u slučaju da igračev tim ili sam igrač dostigne zadani rezultat tijek igre vodi na iduću razinu te u slučaju da se radi o posljednjoj razini igra završava te se igrač smatra pobjednikom. Rezultat igrača ostaje zabilježen nakon igre, te će se o ovoj značajki govoriti više u nastavku.



Slika 21 : Score Manager (Izvor: Vlastita produkcija)

5.5. Kamera

Kamera gleda na igrača iz perspektive trećega lica, te se mijenja u slučaju kada igrač pritisne tipku za ciljanje, tada se kamera približi igraču kako bi igrač lakše ciljao neprijatelja.

5.6. Jezik

Korisnik ima mogućnost odabira između dva jezika, a to su hrvatski i engleski, te se odabirom jednoga, tekst na korisničkom sučelju mijenja u željeni jezik. Više o samoj implementaciji istoga biti će obrađeno u nastavku rada.

5.7. Način igre

Prije pokretanja prve razine, igrač ima mogućnost odabira težine i načina igre. Težine igre između koji igrač može birati su : lagano (eng. Easy), srednje (eng. Medium) i teško (eng. Hard), dok što se tiče načina igre izbor je sljedeći: Timski (eng. Team), Sam (eng. Single) i neograničen (eng. Unlimited). Svaki od ovih izbora ima utjecaj na neke od inicijalnih vrijednosti koje se postavljaju prilikom pokretanja igre, a logika postavljanja vrijednosti je prikazana u tablicama u nastavku.

Vrijednost zdravlja igrača postavlja se s obzirom na razinu na kojoj se igrač nalazi, te je ona određena po sljedećim pravilima koji su vidljivi u tablici ispod.

Tablica 1 : Prikaz vrijednosti zdravlja igrača s obzirom na razinu

Zdravlje igrača (vrijednost)	Razina
100	1
95	2
95	3
90	4
90	5

(Izvor: Vlastita produkcija)

Vrijednost štete koju puška igrača nanosi postavlja se s obzirom na težinu igre koju je igrač odabrao te je ona određena po sljedećim pravilima koji su vidljivi u tablici ispod.

Tablica 2 : Prikaz štete koju igračeva puška nanosi s obzirom na težinu igre

Šteta koju puška nanosi (vrijednost)	Težina igre
10	Lagana
9	Srednje
8	Teška

(Izvor: Vlastita produkcija)

Vrijednost štete koju puška neprijatelja nanosi postavlja se s obzirom na razinu na kojoj se igrač nalazi, te je ona određena po sljedećim pravilima koji su vidljivi u tablici ispod.

Tablica 3 : Prikaz štete koju neprijateljeva puška nanosi s obzirom na razinu igre

Šteta koju neprijatelj nanosi (vrijednost)	Razina
5	1
5.5	2
5.75	3
6	4
6.25	5

(Izvor: Vlastita produkcija)

Vrijednost broj ubojstava potreban za odlazak na iduću razinu k postavlja se s obzirom na razinu na kojoj se igrač nalazi, te je ona određena po sljedećim pravilima koji su vidljivi u tablici ispod.

Tablica 4 : Prikaz potrebnih ubojstava tima za odlazak na iduću razinu s obzirom na razinu igre

Broj ubojstava potreban za odlazak na iduću razinu (vrijednost)	**Razina**
10	1
10	2
15	3
15	4
20	5

(Izvor: Vlastita produkcija)

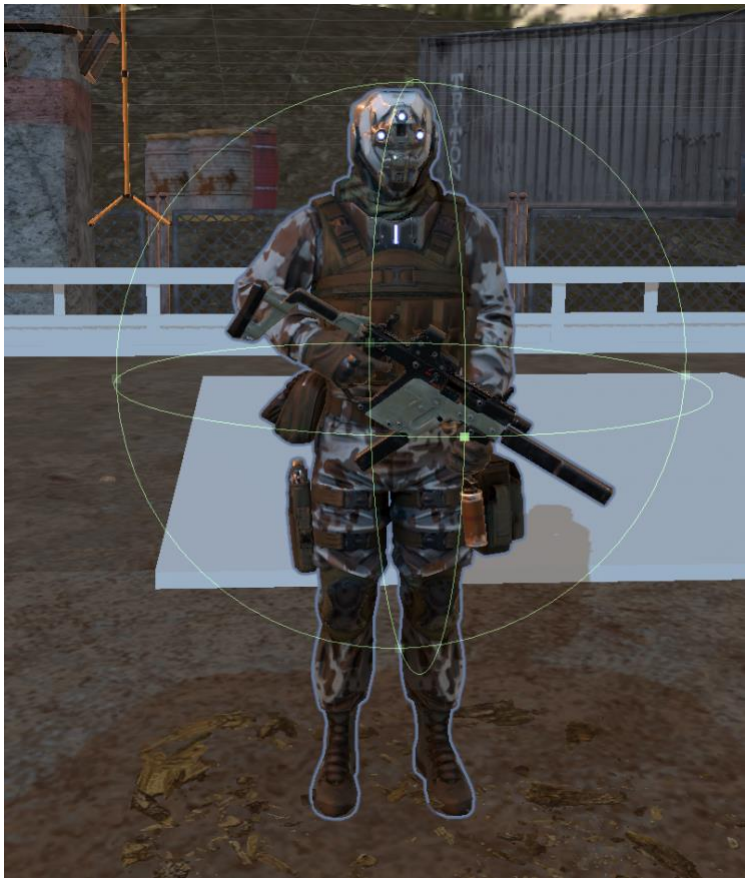
5.8. Okruženje

Za okruženje je korišteno već gotovo okruženje koje je preuzeto za Unity Asset Storea, okruženje je vidljivo na slici 22. Izgled okruženja predstavlja mapu na kojoj se igra odvija, te tu mapu nije moguće napustiti, ograničena je ogradom koju igrač ne može uništiti a ni preskočiti. Svakom objektu na mapi bilo je potrebno dodati Collider kako igrač i ostali likovi u igri ne bi imali mogućnost prolaska kroz iste, odnosno da se objekti ne preklapaju kao što je to i u pravom svijetu.

Slika 22 : Okruženje (Izvor: Vlastita produkcija)

Ostali objekti također imaju kolizije, primjer toga je sami igrač, jer u njegovom slučaju moramo znati kada stoji na tlu a kada je u zraku odnosno kada skače. Primjer collidera

igrača prikazan je na slici broj 23. Također i suigrači i neprijatelji imaju dodijeljene collidere, kao i ostali objekti koji će se obraditi u nastavku.



Slika 23 : Igrač i Collider (Izvor: Vlastita produkcija)

6. Tijek igre

U ovom djelu rada pojašnjen je tijek igre, što je potrebno raditi u fazama igre i kako je isto zamišljeno.

6.1. Početak

Kako bi igrač pokrenuo igru u glavnom izborniku koji se može vidjeti na slici 24., u ovom slučaju u kojem je odabran engleski jezik, odabire play, nakon toga igrač upisuje ime kojim želi igrati (vidi sliku 25.) te nakon što je upisao ime pojavljuje mu se tipka proceed (vidi sliku 26.). Nakon što je igrač upisao korisničko ime sljedeći korak je odabir težine i načina igre (vidi slike 27. i 28.).

Slika 24 : Glavni izbornik (Izvor: Vlastita produkcija)

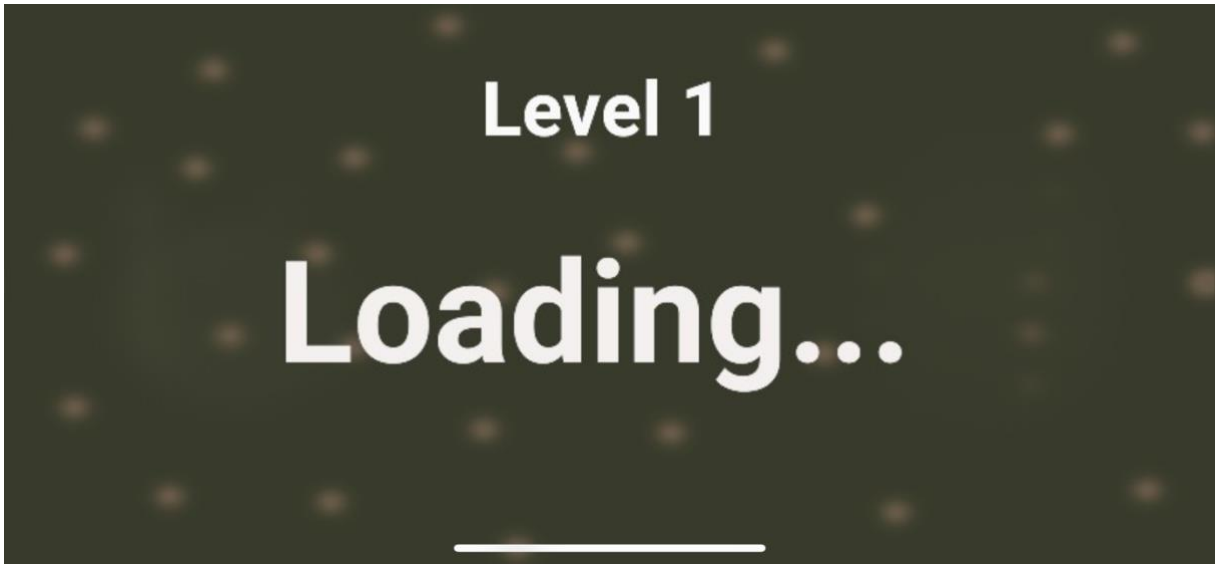
Slika 25 : Upis naziva igrača (Izvor: Vlastita produkcija)

Slika 26 : Upisani naziv igrača (Izvor: Vlastita produkcija)

Slika 27 : Odabir težine igre (Izvor: Vlastita produkcija)

Slika 28 : Odabir načina igre (Izvor: Vlastita produkcija)

Nakon što je sve konfigurirano, pritiskom na tipku procced pokreće se igra.



Slika 29 : Učitavanje igre (Izvor: Vlastita produkcija)

6.2. Sredina

Sredina ili glavni dio igre je ujedno i najduži dio igre, sastoji se pretežito od borbe igračevog i neprijateljskog tima. Za svaku razinu potrebno je i učitavanje same razine.

Slika 30 : Glavni dio igre (Izvor: Vlastita produkcija)

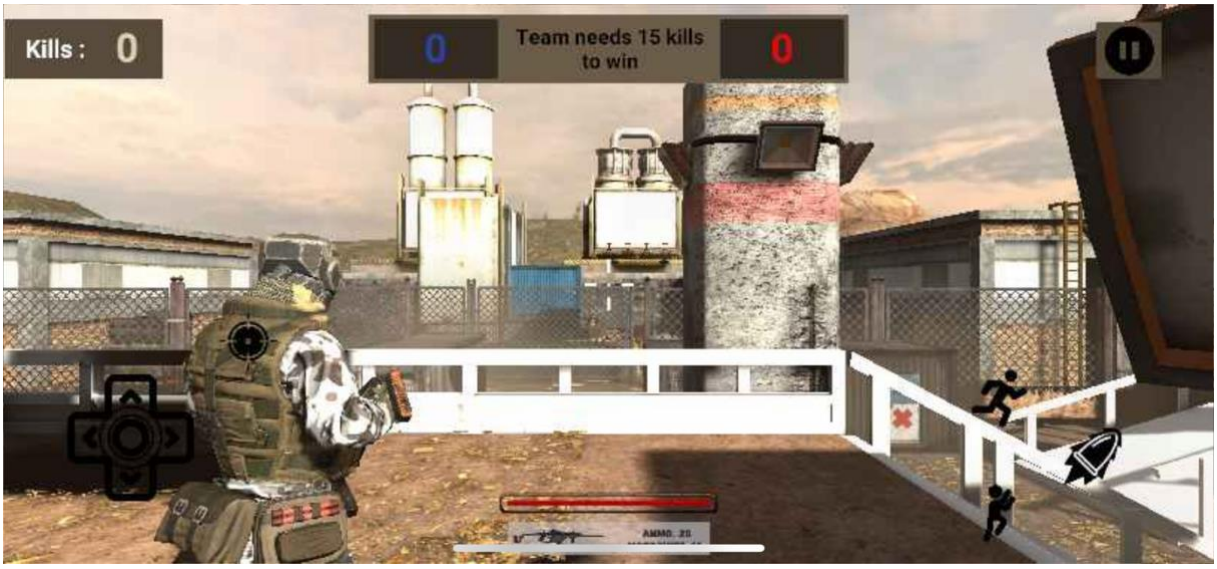
Na početku igre igrač se uvijek nalazi na istoj lokaciji mape. Igru je moguće pauzirati, no ne i spremiti. Kada igračev tim dosegne zadani broj ubojstava neprijatelja učitava se iduća razina.

Igrač ima mogućnost pucanja, trčanja, skakanja, ciljanja protivnika i sakupljanja dodatne municije (vidi slike 31. i 32.).

Slika 31 : Ciljanje (Izvor: Vlastita produkcija)

Slika 32 : Municija (Izvor: Vlastita produkcija)

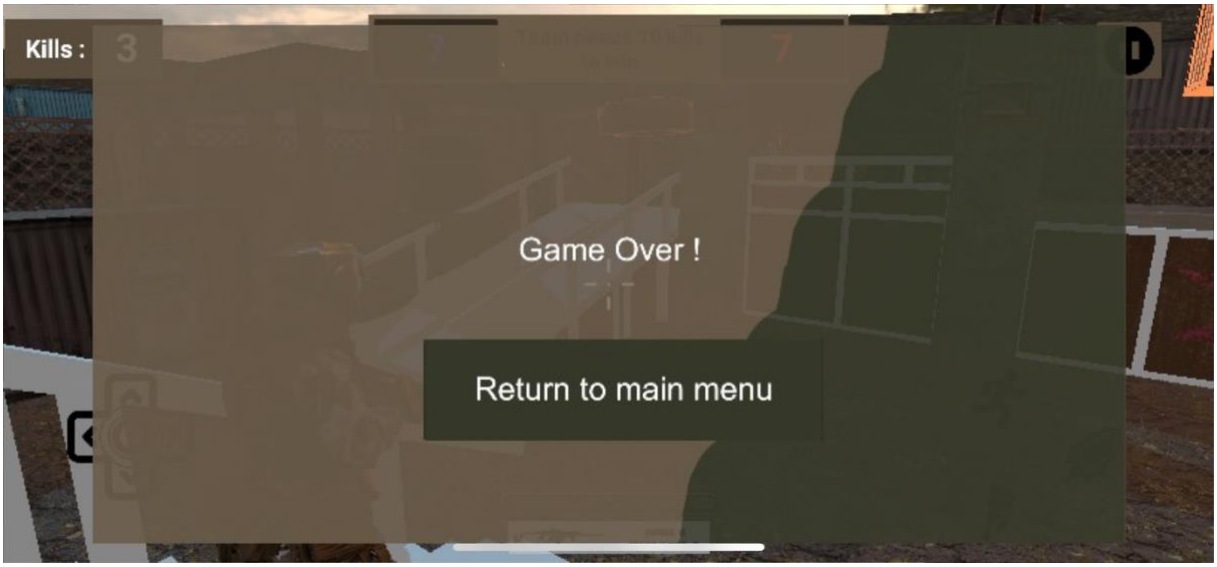
Razine neparnog rednog broja su tamnije prirode, dok su razine parnog rednog broja svjetlije prirode.



Slika 33 : Glavni dio igre svjetlije prirode (Izvor: Vlastita produkcija)

6.3. Kraj

Igra može završiti na nekoliko načina: kada igrač umre neovisno o razini, kada neprijateljski tim prvi dosegne zadani rezultat te kada igrač i njegov tim pobjede svih 5 razina igre. Ukoliko se dogodi jedan od prva spomenuta dva scenarija, otvara se izbornik u kojem je navedeno kako je igra završila te igrač ima jedan izbor, a to je povratak u glavni izbornik.



Slika 34 : Kraj igre (Izvor: Vlastita produkcija)

Ukoliko igrač i njegov tim pobjede svih 5 razina, na vrhu zaslona je ispisano kako je igračev tim pobijedio te se nakon 5 sekundi otvara glavni izbornik.

Slika 35 : Kraj igre - pobjeda (Izvor: Vlastita produkcija)

6.4. Ostale opcije

Neke od ostalih značajki igre koje je važno spomenuti su: pauza, tablica rezultata i postavke. Ukoliko igrač odluči pauzirati igru otvara se izbornik u kojem je moguće podesiti osjetljivost kamere i zvuka, uz izbor nastavak igre ili povratka u glavni izbornik.

Slika 36 : Pauza - izbornik (Izvor: Vlastita produkcija)

Tablica rezultata prikazuje rezultate igrača, njegovo ime i broj ubojstava. Prilikom završetka igre rezultat igrača sprema se u ovu tablicu.

Slika 37 : Tablica rezultata (Izvor: Vlastita produkcija)

Izbornik postavki omogućava igraču podešavanje zvuka (on/off), jezika sučelja (engleski/hrvatski) i podešavanje osjetljivosti kamere.

Slika 38 : Postavke - engleski jezik (Izvor: Vlastita produkcija)

Slika 39 : Postavke - hrvatski jezik (Izvor: Vlastita produkcija)

7. Implementacija

Ovaj dio rada baviti će se programerskim i tehničkim aspektima projekta koje treba istaknuti. S obzirom na to kako je rad opširan, obrađeni su dijelovi koji se smatraju najvažnijima za ovaj projekt.

7.1. Assets

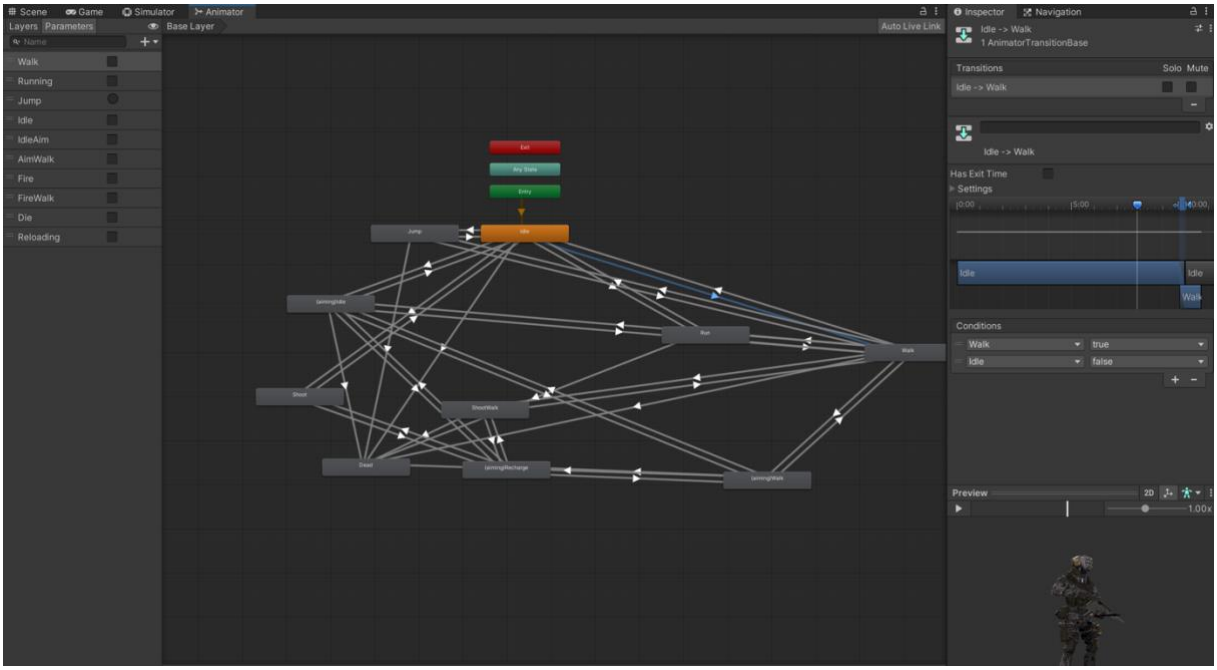
Imovina (eng. Asset) predstavlja svaku vrstu digitalne imovine koja se koristi prilikom izrade pojedine igre, to može biti 2D slika, 3D model, audio zapis, efekt ili nešto drugo. Dodavanje samog asseta u projekt je jednostavno, potrebno je asset postaviti unutar direktorija projekta ili pomoću samog Unity sučelja, tako da se unutar njega povuče željeni asset. Za ovaj projekt korišteno je nekoliko gotovih asseta, kako bi proces izrade igre bilo brži.

Tako su preuzeti asseti za okruženje (eng. Enviroment) u kojem se igra odvija, likove igrača, suigrača i neprijatelja, no animacije za iste su bile izrađene putem online alata, koji će biti naveden u literaturi rada. Zvukovi i efekti koji se događaju tijekom pucanja i pogađanja igrača su također uvedeni u projekt. Uz navedene preuzeti su i asseti za prikazivanje zdravlja igrača, municija koju igrač može pokupiti te oružja, izvori ovih asseta biti će navedeni u literaturi rada. Nakon što su svi potrebni asseti uvedeni u projekt, izrada može početi.

7.2. Animator

Animator je komponenta koja služi za kontrolu nad samim animacijama, nudi grafičko sučelje unutar zasebnog prozora. Omogućuje definiranje stanja (eng. State), te je moguće svakom stanju dodijeliti željenu animaciju. Prijelaz odnosno tranzicija između dva stanja ovisi o parametrima koje je korisnik definirao. Na temelju vrijednosti parametara, animator prelazi iz jednog stanja u drugo.

Na slici 40. prikazan je prozor animatora, s lijeve strane su definirani parametri o kojima će ovisiti pojedini prijelazi animacija, u ovome slučaju tipovi parametara koji se koriste za animiranje igrača su bool i trigger, samo je za skakanje parametar defniran kao trigger, dok su svi ostali tipa bool. S desne strane prikazan je inspector unutar kojeg se nalazi prikaz izgleda animacije (na dnu) te uvjeti koji se moraju ispuniti kako bi se dogodio prijelaz iz jednog stanja u željeno.



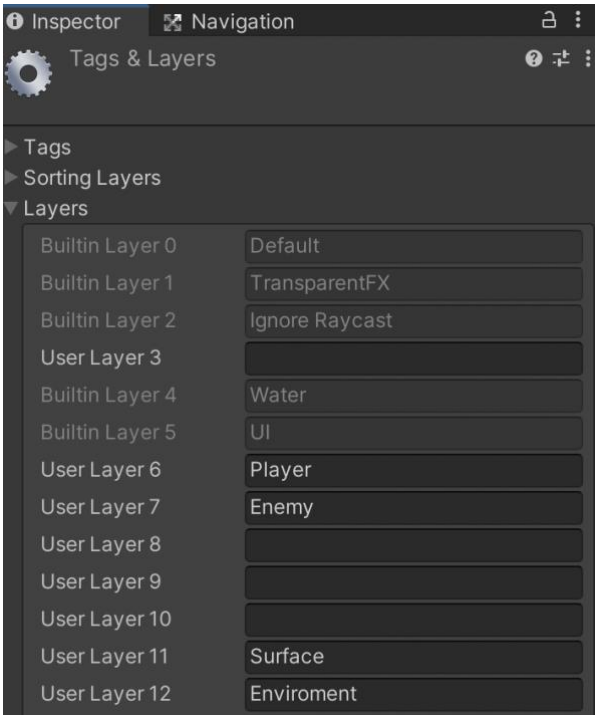
Slika 40 : Animator prozor (Izvor: Vlastita produkcija)

Na slici je prikazan animator koji se odnosi na glavnog igrača. Kako bi se dogodila tranzicija između stanja idle u stanje walk, parametar idle mora biti postavljen na false, dok parametar walk mora biti postavljen na true. Vrijednosti parametara mijenjaju se dinamički programskim putem te se time utječe na promjenu animacije objekta na kojem je animator dodijeljen.

Također animator nudi animiranje statičnih slika u prozoru naziva Animation, te u njemu korisnik ima mogućnost izrađivanja vlastitih animacija, no u ovome radu to nije korišteno, već su animacije preuzete gotove i uvedene u projekt.

7.3. Layers

Slojevima (eng. Layers) se omuogućava razlikovanje objekata u sceni. Definiraju se putem korisničkog sučelja u Unity alatu, a mogu se provjeravati putem samog koda, zapravo ovom značajkom se definira interakcija objekata u sceni, primjer korištenja će biti prikazan u nastavku. Na slici ispod prikazano je sučelje za dodavanje sloja, nalazi se na desnoj strani te je dovoljno upisati ime sloja kako bi on bio definiran. Nakon dodavanja sloja potrebno ga je pridružiti željenom objektu.



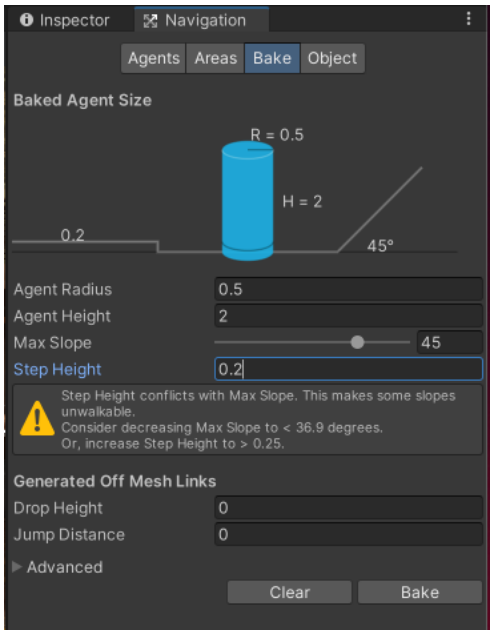
Slika 41 : Layer prozor (Izvor: Vlastita produkcija)

7.4. NavMeshAgent

NavMesh je definirana mreža (eng. Mesh) u Unity sceni, koja specificira navigacijska područja u trenutnom okruženju, uključujući područja gdje se likovi u igri mogu kretati, kao i prepreke. Koristi se za scenraije koji uključuju pronalaženje puta i navigaciju samim putem koja je kontrolirana umjetnom inteligencijom.

NavMeshAgent je komponenta koje služi likovima u igri za izbjegavanje jednih od drugih, kretanje prema zadanom cilju ili nekoj drugoj situaciji za koju je potrebno prostorno razmišljanje ili pronalaženje puta.

Kako bi se navedena komponenta mogla pravilno koristiti potrebno je napraviti „bake“ okruženja scene. Odabirom objekta nad kojim želimo obaviti navedenu operaciju, u ovom slučaju okruženja, na desnoj strani sučelja gdje se nalazi inspector ima još jedna kartica koja se zove navigation, odabirom nje možemo izvršiti „bake“.



Slika 42 : Bake prozor (Izvor: Vlastita produkcija)

Nakon što je „bake“ napravljen za željenu scenu dodaje se komponenta NavMeshAgent objektu kojemu se želi omogućiti kretanje okruženjem.

7.5. PlayerPrefs

U nastavku rada često će se naići na ovu klasu, PlayerPrefs je klasa koja omogućuje pohranjivanje vrijednosti tijekom igre. Pomoću PlayerPrefs mogu se pohraniti idući tipovi podataka: integer, float i string. Unity pohranjuje PlayerPrefs u lokalni registar bez ikakve enkripcijem, stoga ga nije preporučljivo koristiti za pohranu osjetljivih podataka.

7.6. Igrač

7.6.1.Kretanje

Metoda playerMove brine se o kretnjama igrača. Prve dvije linije dohvaćaju unos igrača, ove vrijednosti ovise o smjeru u kojem je sami joystick usmjereni te od tih vrijednosti definirano smjer kretanja igrača. Uvjet koji provjeravamo je želi li se igrač kretati, ako želi program ulazi u if blok koda. Nakon što je program ušao u if blok, izračunavamo kut za koji želimo zarotirati igrača, on ovisi i o kameri jer želimo da ona uvijek bude pozicionirana iza igrača, te se nakon toga metodom Euler klase Quaternion igrač rotira. CharacterController je komponenta koja se koristi za kretnje igrača, u njegovu metodu move prvi parametar označava smjer u kojem se želimo kretati, drugi parametar označava brzinu kretanja igrača, te je posljednji parametar vrijeme jer želimo da je kretnja ovisna o vremenu.

```
   void playerMove()
   {
    float horizontal_axis = joystick.Horizontal;
    float vertical_axis = joystick.Vertical;

    Vector3 direction = new Vector3(horizontal_axis, 0f,
vertical_axis).normalized;

    if(direction.magnitude >= 0.01f)
    {
        animator.SetBool("Running", false);
        animator.SetBool("Idle", false);
        animator.SetBool("Walk", true);
        animator.SetBool("IdleAim", false);

        float targetAngle = Mathf.Atan2(direction.x, direction.z) *
Mathf.Rad2Deg + playerCamera.eulerAngles.y;
        float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y,
targetAngle, ref turnCalmVelocity, turnCalmTime);

        transform.rotation = Quaternion.Euler(0f, angle, 0f);

        Vector3 moveDirection = Quaternion.Euler(0f, targetAngle, 0f) *
Vector3.forward;
        controller.Move(moveDirection.normalized * playerSpeed *
Time.deltaTime);
        currentPlayerSpeed = playerSpeed;
    }
    else
    {
        animator.SetBool("Idle", true);
        animator.SetBool("Walk", false);
        float targetAngle = Mathf.Atan2(direction.x, direction.z) *
Mathf.Rad2Deg + playerCamera.eulerAngles.y;
        float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y,
targetAngle, ref turnCalmVelocity, turnCalmTime);
        transform.rotation = Quaternion.Euler(0f, angle, 0f);
```

```
        currentPlayerSpeed = 0f;
    }
   }
```

Metoda sprint sastoji se od iste logike, no u tom slučaju brzina kretanja nije jednake vrijednosti već je veća.

7.6.2. Skakanje

Skakanje igrača implementirano je na način da ukoliko je igrač pritisnuo tipku za skakanje i nalazi se na tlu, podižemo ga po y-osi.

```
   void Jump()
   {
    if(CrossPlatformInputManager.GetButton("Jump") && onSurface)
    {
        animator.SetBool("Walk", false);
        animator.SetTrigger("Jump");
        velocity.y = Mathf.Sqrt(jumpRange * -2 * gravity);
    }
    else
    {
        animator.ResetTrigger("Jump");
    }
   }
```

7.6.3. Primanje štete

Ako igrača pogodi hitac neprijatelja smanjujemo mu vrijednost zdravlja. Ukoliko vrijednost zdravlja je jednako ili manje 0 pozivamo metodu playerDie i igra završava.

```
   public void playerHitDamage(float takeDamage)
   {
    presentHealth -= takeDamage;
    healthBarIndicator.setHealth(presentHealth);
    if(presentHealth <= 0)
    {
        playerDie();
    }
   }

   private void playerDie()
   {
    Cursor.lockState = CursorLockMode.None;
    Config.addRecord(new
PlayerScoreLeaderboard(PlayerPrefs.GetString("playerName"),
PlayerPrefs.GetInt("playerKills")));
    endGameCanvas.SetActive(true);
   }
```

Ako je igrač umro, njegov rezultat postavljamo u tablicu rezultata te prikazujemo zaslon na kojemu piše da je igra gotova.

7.6.4.Skupljanje municije

Igrač ima mogućnost skupljanja municije koja se nalazi na mapi, na objekt municije postavljena je komponenta zadužena za kolizije, te joj je vrijednost bool parametra isTrigger omogućena, naime kako bi se kolizija između dvaju objekata dogodila, barem jedan od njih mora imati ovaj parametar postavljen na vrijednost true.

```
private void OnTriggerEnter(Collider ammo)
 {
  if(ammo.gameObject.CompareTag("Ammo"))
  {
     rifle.addMag();
     ammo.gameObject.SetActive(false);
     StartCoroutine(RespawnAmmo(ammo.gameObject));
  }
 }

 IEnumerator RespawnAmmo(GameObject ammo)
 {
     yield return new WaitForSeconds(10);
     ammo.SetActive(true);
 }
```

Kada igrač dođe do municije, ona mu se dodaje i nestaje sa mape, nakon 10 sekundi municija se opet pojavljuje na istome mjestu.

7.6.5.Zdravlje

Prilikom učitavanja razine, igraču se postavlja vrijednost zdravlja pomoću ove metode, inicijalna vrijednost varijable *playerHealth* iznosi 100.

```
private float checkLevel(int level)
 {
     switch (level)
     {
         case 1: return playerHealth;
         case 2: return playerHealth * 0.95f;
         case 3: return playerHealth * 0.95f;
         case 4: return playerHealth * 0.90f;
         case 5: return playerHealth * 0.90f;
         default: return playerHealth;

     }
 }
```

7.6.6.Puška

Kao što je već navedeno u radu puška je dio igrača. Update metoda puške provjerava je li pritisnuta tipka za pucanje, te na temelju toga postavlja vrijednosti parametara koji su potrebni za animator. Važno je napomenuti da smo fireCharge i nextTimeToShoot varijablama, čije su inicijalne vrijednosti 15 odnosno 0, implementirali vrijeme između pucanja. U nastavku je isječak koda iz Update metode.

```
if(CrossPlatformInputManager.GetButton("Shoot") && Time.time >=
nextTimeToShoot)
        {
            animator.SetBool("Fire", true);
            animator.SetBool("Idle", false);

            nextTimeToShoot = Time.time + 1f/fireCharge;
            Shoot();
        }
```

7.6.6.1. Pucanje

Kada igrač pritisne tipku za pucanje poziva se metoda Shoot skripte koja je dodijeljena pušci. Ulaskom u metodu odmah se oduzima trenutno stanje municije za jedan, te se na temelju toga ažurira tekst koji prikazuje stanje municije.

Za provjeravanje je li igrač nešto pogodio koristi se Physics.Raycast() metoda, kojom zapravo provjeravamo je li igrač nešto pogodio. Pomoću ove metode stvara se virtualna zraka koja se proteže zadanim smjerom pravca sve dok se ne sudari sa objektom. U ovom primjeru stvara se virtualna zraka koja započinje od sredine kamere u smjeru prema naprijed, te objekt koji gađamo mora biti untar dometa puške koji je definiran.

Ukoliko je igrač pogodio objekt kojem je dodijeljena skripta Objects, oduzima se zdravlje navedenom objektu, a ukoliko se radi o neprijatelju, također mu se oduzima zdravlje, no ukoliko je igrač ubio neprijatelja (skripta Enemy, dodijeljena svakom neprijatelju kao komponenta), povećava mu se vrijednost ubojstava za 1.

```
void Shoot()
    {
        presentAmmunition--;

        AmmoCount.instance.UpdateAmmoText(presentAmmunition);
        AmmoCount.instance.UpdateMagText(mag);

        muzzleSpark.Play();
        audioSource.PlayOneShot(shootingSound);

        RaycastHit hitInfo;

        if(Physics.Raycast(cam.transform.position, cam.transform.forward,
out hitInfo, shootingRange))
        {
            Objects objects = hitInfo.transform.GetComponent<Objects>();

            Enemy enemy = hitInfo.transform.GetComponent<Enemy>();

            if(objects != null)
            {
                objects.objectHitDamage(giveDamage);
                GameObject woodGo = Instantiate(woodedEffect,
hitInfo.point, Quaternion.LookRotation(hitInfo.normal));
                Destroy(woodGo, 1f);
            }

            else if(enemy != null)
            {
                enemy.enemyHitDamage(giveDamage);
                if(!enemy.IsAlive())
                {
                    numberOfKills +=1;
                    PlayerPrefs.SetInt("playerKills", numberOfKills);
                    killText.text = numberOfKills + "";
                }
                GameObject goreGo = Instantiate(goreEffect, hitInfo.point,
Quaternion.LookRotation(hitInfo.normal));
                Destroy(goreGo, 1f);
            }
        }
    }
```

7.6.6.2. Punjenje municije

Metodom Reload puni se municija puške, kada igrač ostane bez nje, brzina kretanja igrača postavlja se na 0 te punjenje municije traje ovisno o postavljenoj vrijednost varijable reloading time te se nakon toga municija postavlja na maksimalan broj, a brzine se postavljaju na inicijalne vrijednosti.

```
IEnumerator Reload()
    {
        player.playerSpeed = 0f;
        player.playerSprint = 0f;
        setReloading = true;

        animator.SetBool("Reloading", true);
        audioSource.PlayOneShot(reloadingSound);
        yield return new WaitForSeconds(reloadingTime);

        animator.SetBool("Reloading", false);
        presentAmmunition = maximumAmmunition;
        player.playerSpeed = 1.9f;
        player.playerSprint = 3f;
        setReloading = false;
    }
```

Metodu u nastavku poziva igrač kada pokupi municiju na mapi.

```
public void addMag() {
        mag++;
        AmmoCount.instance.UpdateMagText(mag);
    }
```

7.6.6.3. Šteta

Na temelju odabrane težine igre, postavlja se vrijednost koliku štetu nanosi igračevo oružje.

```
private float checkDifficulty(int diff)
    {
        switch (diff)
        {
            case 0: return 10f;
            case 1: return 10f * 0.9f;
            case 2: return 10f * 0.8f;
            default: return 10f;
        }
    }
```

7.7. Neprijatelj

Kao što je već spomenuto u radu, suigrač i neprijatelj nisu toliko različiti što se tiče implementacije njihovog ponašanja, razlika je u Awake metodi u kojoj, kod neprijatelja, provjeravamo način i razinu igre.

```
private void Awake()
    {
        handleGameMode();
        enemyAgent = GetComponent<NavMeshAgent>();
        presentHealth = enemyHealth;
        giveDamage = checkLevel(PlayerPrefs.GetInt("Level"));
    }

private void Update()
    {
        playerInvisionRadius = Physics.CheckSphere(transform.position,
visionRadius, PlayerLayer);
        playerInShootingRadius = Physics.CheckSphere(transform.position,
shootingRadius, PlayerLayer);

        if(playerInvisionRadius && !playerInShootingRadius) Pursueplayer();
        if(playerInvisionRadius && playerInShootingRadius) ShootPlayer();
    }
```

Update metoda provjerava je li neprijatelj (u ovome slučaju igračev suigrač) u vidnom polju i je li u radijusu kojeg hitac može doseći te na temelju toga poziva jednu od dvije metode.

7.7.1. Praćenje igrača

Praćenje igrača ili suigrača implementirano je pomoću NavMeshAgent komponente, te se u ovom slučaju postavlja ili ažurira cilj prema kojemu neprijatelj treba ići.

```
private void Pursueplayer()
    {
        if(enemyAgent.SetDestination(playerBody.position))
        {
            anim.SetBool("Running", true);
            anim.SetBool("Shooting", false);
        }
        else
        {
            anim.SetBool("Running", false);
            anim.SetBool("Shooting", true);
        }
    }
```

7.7.2.Pucanje

Kada je protivnik (u ovom slučaju igrač ili suigrač) u vidnom polju i na udaljenosti na kojoj će ga hitac pogoditi poziva se Shoot metoda. Cilj kretanja postavlja se na trenutnu neprijateljevu poziciju jer se više ne treba kretati nego treba pucati na protivnički tim.

Logika pucanja slična je onoj koja je implementirana kod samog igrača, no tu se provjerava je li pogođen igrač ili njegov suigrač. Nakon što je neprijatelj pucao, postavlja se vrijednost varijable *previouslyShoot* na true te se nakon toga poziva metoda ActiveShooting sa odgodom čija je jedina zadaća postaviti spomenutu varijablu na vrijednost false čime se implementiralo vrijeme između pucanja.

```
private void ShootPlayer()
    {
        enemyAgent.SetDestination(transform.position);
        transform.LookAt(LookPoint);

        if(!previouslyShoot)
        {
            muzzleSpark.Play();
            audioSource.PlayOneShot(shootingSound);

            RaycastHit hit;

            if(Physics.Raycast(ShootingRayCastArea.transform.position,
ShootingRayCastArea.transform.forward, out hit, shootingRadius)) {
                Debug.Log("Shooting" + hit.transform.name);
                PlayerScript playerBody =
hit.transform.GetComponent<PlayerScript>();

                if(playerBody != null)
                {
                    playerBody.playerHitDamage(giveDamage);
                }

                PlayerTeammate playerBodyTm =
hit.transform.GetComponent<PlayerTeammate>();

                if(playerBodyTm != null)
                {
                    playerBodyTm.PlayerTeammateHitDamage(giveDamage);
                }

                anim.SetBool("Running", false);
                anim.SetBool("Shooting", true);
            }

            previouslyShoot = true;
            Invoke(nameof(ActiveShooting), timebtwShoot);
        }
    }
```

7.7.3.Primanje štete i ponovno rođenje

Primanje štete neprijatelja slično je implementirano kao i kod igrača, no za razliku od igrača neprijatelj ima sposobnost ponovnog rođenja.

```
public void enemyHitDamage(float takeDamage)
    {
     presentHealth -= takeDamage;
     if(!IsAlive())
     {
         StartCoroutine(Respawn());
     }
    }
```

Ukoliko je vrijednost zdravlja neprijatelja manja ili jednaka 0, poziva se Respawn metoda kojom se neprijatelj ponovno rađa, nakon 5 sekundi, na zadanoj lokaciji, u ovome slučaju na lokaciji Spawn objekta. Svaki suigrač i neprijatelj ima svoju vlastitu lokaciju ponovnog rođenja.

```
    IEnumerator Respawn()
    {
     enemyAgent.SetDestination(transform.position);
     enemySpeed = 0f;
     shootingRadius = 0f;
     visionRadius = 0f;
     playerInShootingRadius = false;
     playerInvisionRadius = false;
     anim.SetBool("Die", true);
     anim.SetBool("Running", false);
     anim.SetBool("Shooting", false);

     gameObject.GetComponent<CapsuleCollider>().enabled = false;
     scoreManager.kills += 1;

     yield return new WaitForSeconds(5f);

     gameObject.GetComponent<CapsuleCollider>().enabled = true;


     presentHealth += 120f;
     enemySpeed = 1f;
     shootingRadius = 20f;
     visionRadius = 50f;
     playerInvisionRadius = true;
     playerInShootingRadius = false;

     anim.SetBool("Die", false);
     anim.SetBool("Running", true);

     EnemyCharacter.transform.position = Spawn.transform.position;
     Pursueplayer();

    }
```

7.7.4.Šteta

Prilikom učitavanja razine, neprijatelju se postavlja vrijednost nanošenja štete pomoću ove metode.

```
   private float checkLevel(int level)
    {
        switch (level)
        {
            case 1: return 5f;
            case 2: return 5f * 1.1f;
            case 3: return 5f * 1.15f;
            case 4: return 5f * 1.20f;
            case 5: return 5f * 1.25f;
            default: return 5f;

        }
    }
```

7.7.5.Način igre

Svaki tim se sastoji od 4 igrača, jedan tim čine glavni igrač i 3 suigrača dok drugi tim čine 4 neprijatelja. Svakom suigraču i neprijatelju dodijeljena je točka gledišta (eng. Look point) od jednog od protivnika, ona služi kako bi se likovi pravilno okrenuli prema protivniku. Ukoliko je igrač odabrao način igre u kojemu želi igrati sam protiv protivničkog tima, ova metoda postavlja svim neprijateljima točku gledišta od igrača.

```
   private void handleGameMode()
    {
        if(PlayerPrefs.GetInt("Mode") == 1)
        {
            GameObject player = GameObject.Find("Player");
            GameObject lookPoint = GameObject.Find("/Player/Lookpoint");
            playerBody = player.transform;
            LookPoint = lookPoint.transform;

            GameObject PlayerTeammates =
GameObject.Find("PlayerTeammates");
            PlayerTeammates.SetActive(false);
        }
    }
```

7.8. Zvukovi i efekti

Unity nudi sustav čestica (eng. Particle System) pomoću kojeg se mogu simulirati tekućine, dim, oblaci, plamen i ostali efekti. Efekti za ovaj rad su također preuzeti i uvedeni u projekt. Za primjer efekti se koriste kod pucanja, u ovome slučaju sustav čestica se zove muzzle spark te on simulira bacanje iskri iz cijevi puške prilikom pucanja pozivom metode play.

```
muzzleSpark.Play();
```

Za reproduciranje audio zapisa koristi se komponenta AudioSource koja je pridodana objektu koji će proizvoditi zvuk u 3D okruženju. Za reprodukciju zvukova tijekom same igre potrebna je komponenta AudioListener. Ova komponenta najčešće se pridodaje kameri koja se koristi tijekom igre.

Korištenje ove značajke se također može pronaći u metodama koje su zaslužne za implementaciju pucanja, no ispod se nalazi metoda skripte koja je zadužena za proizvodnju zvukova tijekom kretanja te je dodijeljena kao komponenta svim likovima u igri.

```
private void Step()
{
 audioSource.PlayOneShot(clip);
}
```

7.9. Kamera

Polje koje je označeno na slici ispod predstavlja joystick kojim upravljamo kamerom.

Slika 43 : Korisničko sučelje (Izvor: Vlastita produkcija)

Kretanje kamere prema gore ili prema dolje odnosno po y-osi je ograničeno što se može primjetiti na liniji koda na kojoj se koristi Mathf.Clamp metoda koja brine o tome da vrijednost y ne prelazi maksimalnu i mnimalnu granicu koja je zadana.

```
private void LateUpdate()
{
        currentX += floatingJoystick.Horizontal * sensitivity *
Time.deltaTime;
        currentY -= floatingJoystick.Vertical * sensitivity *
Time.deltaTime;

        currentY = Mathf.Clamp(currentY, YMin, YMax);

        Vector3 Direction = new Vector3(0, 0, -CameraDistance);

        Quaternion rotation = Quaternion.Euler(currentY, currentX, 0);

        transform.position = lookAt.position + rotation * Direction;

        transform.LookAt(lookAt.position);
}
```

Također skripta zadužena za upravljanjem kamere, sadrži metodu kojom se postavlja udaljenost kamere od igrača i njena osjetljivost prilikom upravljanja.

```
public void setCameraProps(float distance, float sensitivity)
{
        this.CameraDistance = distance;
        PlayerPrefs.SetFloat("CameraSensitivity", sensitivity);
}
```

U ovome projektu postoje dvije kamere, zadana kamera te kamera koja se aktivira ukoliko igrač pritisne tipku za ciljanje. Definirana je skripta koja se bavi ovim problemom, a u nastavku će biti prikazan blok koda koji je zadužen za aktiviranje kamere tijekom ciljanja.

```
if(CrossPlatformInputManager.GetButton("Aim‟)
{
            animator.SetBool("Idle", false);
            animator.SetBool("IdleAim", true);
            animator.SetBool("AimWalk", true);
            animator.SetBool("Walk", true);
            mouseLook.setCameraProps(1f, 200f);
            ThirdPersonCanvas.SetActive(false);
            AimCanvas.SetActive(true);
}
```

Kada igrač pritisne tipku za ciljanje, poziva se metoda setCameraProps te joj se postavlja udaljenost od igrača i osjetljivost kamere. Ovim blokom koda se zapravo kamera približi igraču, a ukoliko nije pritisnuta tipka za ciljanje, udaljenost kamere postavlja se na vrijednost 2.

7.10. Sučelje

Ovaj dio rada obrađuje pojedine implementacije koje su pomoćne prirode.

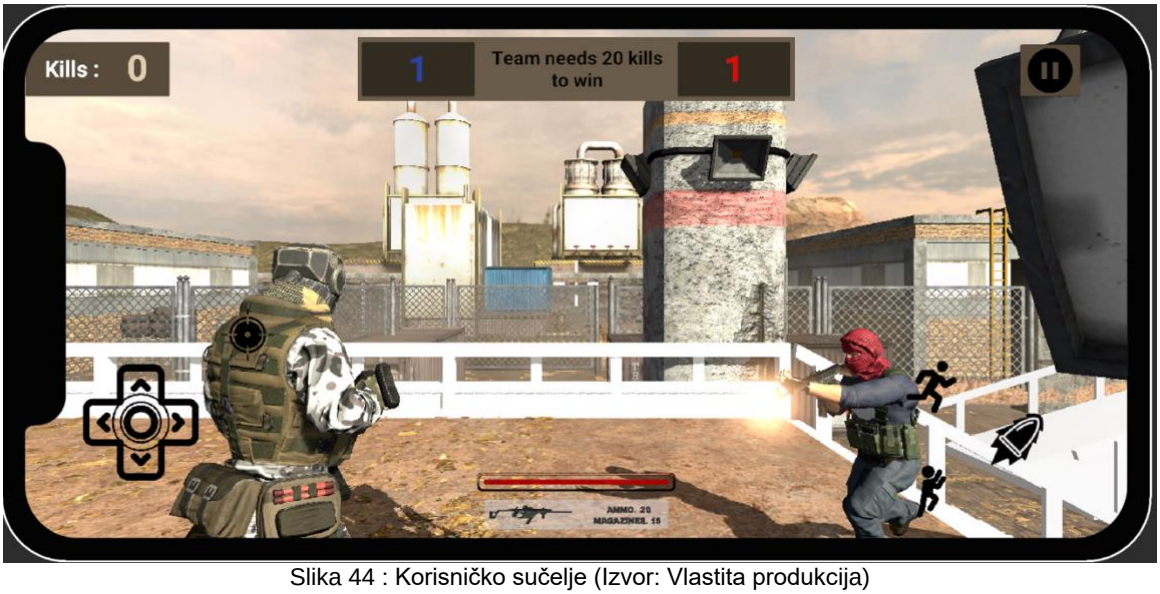
7.10.1. Korisničko sučelje (tijekom igre)

Korisničko sučelje koje je prikazano tijekom igre sadrži nekoliko elemenata kojima se dinamički mijenjaju vrijednosti tijekom same igre.

Metode u nastavku su zadužene za ažuriranje ispisa trenutnog stanja municije i trenutnog broja spremišta sa municijom koje igrač ima.

```
public void UpdateAmmoText(int presentAmmunition)
{
  ammunitionText.text = "Ammo. " + presentAmmunition;
}

public void UpdateMagText(int mag)
{
  magText.text = "Magazines. " + mag;
}
```



Slika 44 : Korisničko sučelje (Izvor: Vlastita produkcija)

Na isti način rade i metode zadužene za indikator trenutnog stanja zdravlja igrača.

```
public void setHealth(float health)
{
  healthBarSlider.value = health;
}
```

Skripta koja je zadužena za vođenje rezultata trenutne razine, ispisuje tekst koliko je ubojstava potrebno za završetak razine, te ispisuje trenutne rezultate timova. U nastavku je prvo prikazan dio koda zadužen za ispisivanje poruke između rezultata timova, dok je drugi dio koda primjer kada igračev tim skupi potreban broj ubojstava.

```
if(kills < neededKills && enemyKills < neededKills)
{
 MainText.text = string.Format("Team needs {0} kills to win", neededKills);
}


if(kills >= neededKills)
{
MainText.text = "Player Team Victory";
}
```

Također u ovoj skripti se nalazi metoda koja određuje koliko je ubojstava potrebno za prelazak na iduću razinu.

Ukoliko je igrač odabrao način unlimited, potreban broj ubojstava je postavljen na 1000 te i u slučaju da se on dostigne igra ne prelazi na iduću razinu već završava.

7.10.2. Tablica rezultata

U nastavku se nalazi petlja koja je zadužena za obrađivanje svih zabilježenih rezultata te ih ispisuje u tablicu.

```
foreach (PlayerScoreLeaderboard record in list)
{
   var item = Instantiate(tableRow);
   var no = item.transform.Find("No").GetComponent<Text>();
   var player = item.transform.Find("PlayerName").GetComponent<Text>();
   var score = item.transform.Find("Score").GetComponent<Text>();

   no.text = "" + ++i;
   player.text = record.getPlayerName();
   score.text = "" + record.getPlayerKills();

   item.transform.SetParent(content);
 }
```

7.10.3. Postavke

U postavkama igrač ima mogućnost gašenja zvuka na razini cijele igre, postavljanja jezika korisničkog sučelja na hrvatski ili engleski jezik te podešavanje osjetljivosti kamere.

Ukoliko igrač odabere da želi ugasiti zvuk izvršava se sljedeća linija koda

```
AudioListener.volume = 0;
```

U suprotnom ova vrijednost je postavljena na 1.

Prilikom pomicanja klizača (eng. Slider) vrijednost osjetljivosti kamere postavlja se na sljedeći način.

```
void Update()
{
  PlayerPrefs.SetFloat("CameraSensitivity", slider.value);
}
```

7.10.4. Jezik

Prilikom odabira željenog jezika izvršava se sljedeći blok koda.

```
if(CrossPlatformInputManager.GetButton("English"))
{
  PlayerPrefs.SetString("Language","English");
}
if(CrossPlatformInputManager.GetButton("Croatian"))
{
  PlayerPrefs.SetString("Language","Croatian");
}
```

U nastavku je prikazan dio klase koja zapravo predstavlja rječnik te se iz nje dohvaćaju vrijednosti za pojedini tekst na sučelju.

```
public static string PLAY_EN = "Play";
public static string PLAY_CRO = "Nova igra";
public static string EXIT_EN = "Exit";
public static string EXIT_CRO = "Izlaz";
public static string SETTINGS_EN = "Settings";
public static string SETTINGS_CRO = "Postavke";
```

Također svaka scena ima dodjeljenju skriptu koja brine o ispisivanju teksta u ispravnom jeziku, u nastavku je prikazan blok koda koji je zadužen za ispisivanje teksta u glavnom izborniku.

```
if(PlayerPrefs.GetString("Language") == "Croatian")
{
 play.text = LanguageData.PLAY_CRO;
 leaderboard.text = LanguageData.LEADERBOARD_CRO;
 settings.text = LanguageData.SETTINGS_CRO;
 exit.text = LanguageData.EXIT_CRO;
}
else
{
 play.text = LanguageData.PLAY_EN;
 leaderboard.text = LanguageData.LEADERBOARD_EN;
 settings.text = LanguageData.SETTINGS_EN;
 exit.text = LanguageData.EXIT_EN;
}
```

8. Zaključak

U ovom radu izrađena je akcijska pucačka igra iz perspektive trećeg lica u programskom alatu Unity. Uz Unity korišteni su i alati poput Adobe Photoshop, koji je služio za izrađivanje jednostavnih grafičkih elemenata potrebnih za ovu igru te su se oni više odnosili na korisničko sučelje same igre, i Visual Studio Code u kojem je razvijen programski kod za potrebe ove igre. Za Visual Studio Code bilo je potrebno instalirati nekoliko proširenja izdanih od Microsofta kako bi Intellisense ispravno radio.

Alat Unity, kroz ovaj rad, je prikazao kako je to napredan alat koji je uistinu specijaliziran za izradu video igara. Smatram da mi je višegodišnje iskustvo u programiranju uveliko olakšalo samo korištenje alata te isto tako i u donošenju odluka prilikom odlučivanja od načinu implementiranja pojedine značajke.

U radu se da primijetiti kako je ovaj žanr dobro zastupljen u industriji video igara, te kako su neki od najpopularnijih naslova ovog žanra. Iz tog razloga smatram da je izazov u današnje vrijeme napraviti igru ovoga žanra koja će konkurirati na tržištu. Također shvatio sam kako igru ne čini samo programski kod, već niz elemenata od ideje, grafičkih elemenata pa sve do finalnog proizvoda, ne mogu zamisliti koliki je broj ljudi i timova potreban za izradu igre poput Grand Theft Auto V.

Oduvijek sam želio izraditi igru, za završni sam također izradio igru, no ona je bila izrađena u React Native frameworku te to danas ne smatram tolikom pothvatom. Također da nisam preuzeo neke od važnijih grafičkih elemenata za izradu ovu igre, ne znam kako bi izgledao finalni proizvod te koje bi funkcionalnosti imao.

Ovaj rad smatram dovoljno dobrim izazovom za nekoga tko se želi baviti izradom video igara, ne samo zbog korištenja Unity alata, već zbog svih drugih aspekata koji su potrebni u samom procesu izrade video igre.

Smatram da industrija video igara ne može propasti, ona se iz dana u dan sve više razvija i raste na tržištu. Također danas postoji puno više natjecanja u igranju video igara nego što je to bio slučaj unazad nekoliko godina.

Zahvaljujem se mentoru Mladenu Koneckom što je pristao biti moj mentor i što mi je odobrio ovu temu.

Popis literature

Unity Technologies, Unity, Unity Technologies (verzija 2021.3.12f1) [alat] dostupno na :

https://unity.com/

Unity Technologies, Plans [web stranica]. Preuzeto 23.01.2023. dostupno na :

https://store.unity.com/compare-plans/

Unity Technologies, Solutions [web stranica]. Preuzeto 28.01.2023. dostupno na :

https://unity.com/solutions/

Unity Technologies, Gaming Report [web stranica]. Preuzeto 28.01.2023. dostupno na :

https://create.unity.com/gaming-report-2022

Unity Technologies, Gaming Report (zadnje editiranje 13.03.2022.) [web stranica]. Preuzeto 28.01.2023. dostupno na :

https://blog.unity.com/games/unity-gaming-report-2022-five-insights-on-the-gaming-industry-today

Unity Technologies, Dokumentacija (03.02.2023.) [web stranica]. Preuzeto 04.02.2023. dostupno na :

https://docs.unity3d.com/Manual/

Microsoft, Visual Studio Code, Microsoft [alat] dostupno na :

https://code.visualstudio.com/

Rust (zadnje editiranje 29.01.2023.) u Wikipedia [web stranica]. Preuzeto 31.01.2023. dostupno na : https://en.wikipedia.org/wiki/Rust_(video_game)

Cities: Skylines (zadnje editiranje 29.01.2023.) u Wikipedia [web stranica]. Preuzeto 31.01.2023. dostupno na : https://en.wikipedia.org/wiki/Cities:_Skylines

Cuphead (zadnje editiranje 27.01.2023.) u Wikipedia [web stranica]. Preuzeto 31.01.2023. dostupno na : https://en.wikipedia.org/wiki/Cuphead

We'll fix it in post (zadnje editiranje 08.07.2022.) [web stranica]. Preuzeto 31.01.2023. dostupno na : https://wellfixitinpost.com/adam-by-unity-demo-team-behind-the-scenes/

Jesse Reinkka (2019.) [istraživački rad]. Preuzeto 31.01.2023. dostupno na : https://www.theseus.fi/bitstream/handle/10024/168595/Reinikka_Jesse.pdf?sequence=2&isAllowed=y

Hired, Companies using Unity [web stranica]. Preuzeto 31.01.2023. dostupno na :

https://hired.com/companies/unity

Mind Inventory (zadnje editiranje 06.04.2022.) Preuzeto 31.01.2023. dostupno na :

https://www.mindinventory.com/blog/unity-3d-game-development/

Enciklopedija, Računalne igre [web stranica]. Preuzeto 01.02.2023. dostupno na :

https://www.enciklopedija.hr/natuknica.aspx?ID=68642

GTA V (zadnje editiranje 23.01.2023.) u Wikipedia [web stranica]. Preuzeto 01.02.2023. dostupno na : https://en.wikipedia.org/wiki/Grand_Theft_Auto_V

Fandom, Red Dead Redemption 2 (zadnje editiranje 03.12.2022) Preuzeto 01.02.2023. dostupno na : https://reddead.fandom.com/wiki/Red_Dead_Redemption_2?action=history

The Last of Us (zadnje editiranje 01.02.2023.) u Wikipedia [web stranica]. Preuzeto 01.02.2023. dostupno na : https://en.wikipedia.org/wiki/The_Last_of_Us

PUBG Mobile, Apple App Store [web stranica]. Preuzeto 02.02.2023. dostupno na : https://apps.apple.com/hr/app/pubg-mobile/id1330123889

Free Fire, Apple App Store [web stranica]. Preuzeto 02.02.2023. dostupno na : https://apps.apple.com/hr/app/free-fire/id1300146617

Cover Fire Gun : Shooting Games, Apple App Store [web stranica]. Preuzeto 02.02.2023. dostupno na : https://apps.apple.com/hr/app/cover-fire-gun-shooting-games/id1148931033

Popis slika

Popis tablica

Prilozi

https://assetstore.unity.com/packages/3d/characters/humanoids/pbr-military-pack-118245
https://assetstore.unity.com/packages/3d/characters/rebels-77626
https://assetstore.unity.com/packages/tools/utilities/health-system-for-dummies-215755
https://assetstore.unity.com/packages/3d/environments/fps-shooting-mobile-game-optimized-environment-213745
https://assetstore.unity.com/packages/3d/props/weapons/ammo-box-7701
https://assetstore.unity.com/packages/3d/props/guns/weapons-pack-realistic-lowpoly-200967
https://freesound.org/
https://www.mixamo.com