2D natjecateljska igra za više igrača u Unity-u

Ivanković, Vedran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

2D NATJECATELJSKA IGRA ZA VIŠE IGRAČA U UNITY-U

Završni rad

Vedran Ivanković

Osijek, 2018.

SADRŽAJ

1. UVOD

Razvoj računalnih igara nikad nije bio dostupniji široj publici nego što je danas. Pojavom besplatnih razvojnih alata za izradu igara (engl. Game engine) pojavila se i mogućnost izrađivanja vlastitih ideja na jednostavnije načine nego ikad prije. Game engine je sama jezgra svake igre. Određuje sve od grafičkog prikaza jednog okvira do svake matematičke operacije u igri. Izrada Game engine-a jako je kompleksan i skup projekt koji svaki studio izrađuje za potrebe svojih igara, te se kao takav često modificira i reciklira za nove projekte. Zbog razine kompleksnosti izrade ovakvog sučelja, razvoj računalnih igara dugo je vremena bio nedostupno područje sve do pojave komercijalnih game engine-a s grafičkim sučeljem kao što su Unity i Unreal Engine. Na ovaj je način omogućena kontrola game engine-ovog izvornog koda bez pisanja istog. Upravo zbog toga, komercijalni game engine-i postali su vrlo popularni s novim programerima jer uvelike ubrzavaju proces izrade i naglašavaju najvažnije obilježje svake igre – njen sadržaj.[1]

Tema ovoga završnog rada je igra za više igrača tipa pucača. Inspirirana je raznim igrama na domenama io koje su besplatne za igrati, jako popularne i nije ih potrebno instalirati; sve što je potrebno jest imati internet vezu i internet pretraživač. Ovaj projekt je izrađen za platformu Windows iako ga je moguće izraditi za WebGL player kako bi stvarno odgovarao pravom duhu .io igara[1].

Projekt je izrađen koristeći C# programski jezik budući da je on najraširenija program u upotrebi kada je u pitanju Unity. Ambijent, objekti i korisničko sučelje svi su izrađeni ili s gotovim objektima dostupnim u samom Unity Editoru ili su originalno izrađeni u programu Krita.

[1] io igre su serija igara za više igrača u stvarnom vremenu čije ime završava u domeni “io”. .io domena najviše razine (ccTLD) izvorno je kod zemlje britanskog teritorija, ali je populariziran za igre nekim od početnih pogodaka u žanru .io igara. U igri igrači zarađuju bodove poražavajući druge igrače, a najbolji su igrači navedeni na ljestvici. Najpoznatije .io igre su Agar.io i Slither.io.

2. RAZVOJNI ALATI

Za izradu ovog završnog rada koristili su se sljedeći alati: Unity 5.4.0 (Personal), Microsoft Visual Studio 2015 Community i Krita.

2.1. Unity 5.4.0

Unity je višeplatformska razvojna okolina za izradu igara (game engine) koju razvija Unity Technologies. Platforma je prvi puta najavljena i objavljena 2005. godine. Od 2018. podržava 27 različitih platformi za razvoj: Windows, iOS, Android, Mac, Linux i dr.[2] Danas je jedan od najrasprostranjenijih game engine-a na tržištu koji koriste svi, od malih nezavisnih studija do velikih poput Blizzard Entertainmenta s igrom Hearthstone: Heroes of Warcraft. Zbog velike baze korisnika postoji mnogo foruma na kojima se može brzo i lako pronaći odgovor na bilo koje pitanje.

Kroz povijest razvoja Unity je imao nekoliko glavnih verzija, a najnovija stabilna verzija je Unity 2018.3.5. Korisnicima se nude četiri vrste dozvola: Personal, Plus, Pro i Enterprise. Personal dozvola u potpunosti je besplatna te je svakome dostupna nakon izrade računa na stranici Unity-ja. Stoga, Personal dozvola korištena je pri izradi ovoga rada.

Širok je raspon platformi koje podržavaju Unity, što je i jedan od razloga zašto je toliko poznat i uspješan. Osim toga, program nije kompliciran za shvatiti, kako zbog jednostavnosti samog programa, tako i zbog velike količine tekstualnih i video sadržaja za pomoć pri korištenju, dostupnih na Unity stranici i općenito na Internetu.

Unity se može koristiti za stvaranje dvodimenzionalnih i trodimenzionalnih igara, kao i za simulacije.

Unity ima mnogo alata koji omogućavaju brzo uređivanje i razvoj. Urednik kompatibilan s Windows i Mac platformom, dizajnerski alati za animacije i još mnogo drugih, a moguće ga je i nadograditi s dodatnim alatima koje su drugi korisnici napravili. Glavni jezik za programiranje u Unity-ju je C# iako je sam napisan u C++-u. [3]

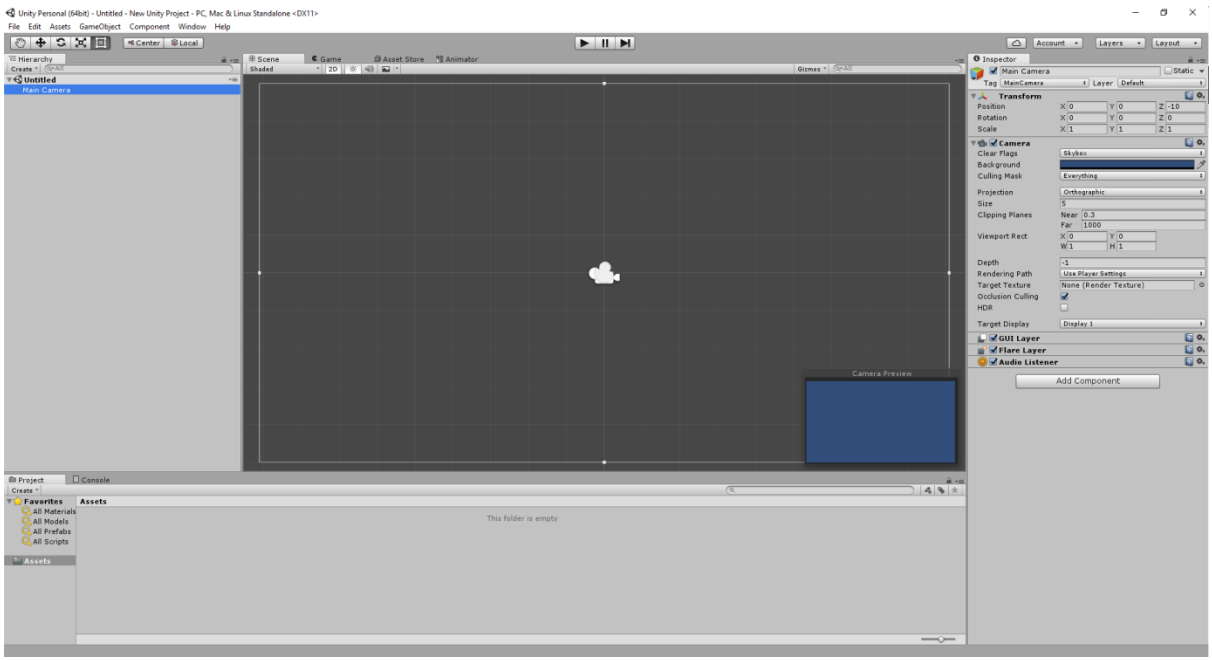
Nakon pokretanja Unity-ja, potrebno je prijaviti se na vlastiti Unity račun, čime započinje rad u programu. Tada je potrebno označiti radi se o novom ili već postojećem projektu. Ako se radi o novom projektu, potrebno je navesti ime projekta, mjesto

pohranjivanja, pripada li nekoj organizaciji, radi li se o 2D ili 3D projektu, itd. Nakon odabira projekta pojavljuje se Unity razvojno okruženje, s mogućnosti prilagođavanja sučelja.

Unity nudi *Scene View* mogućnost kojom se postavljaju objekti poput ambijentalnih elemenata (npr. drveće, grmlje, kamenje, potok), likova, itd. Navedene je objekte moguće pomicati, prilagoditi, okretati, povećati i smanjiti koristeći alate za transformaciju.

Velika prednost Unity-ja je mogućnost igranja igre unutar razvojnog okruženja, što uvelike ubrzava razvoj zbog mogućnosti testiranja igre tijekom razvoja.

Zbog velike popularnosti i učestalosti korištenja Unity-a među velikim i uspješnim studijima koji se bave razvojem igara, zbog činjenice da je glavni jezik za programiranje u Unity-ju C#, zbog jednostavnosti korištenja i svih ostalih navedenih prednosti i mogućnosti, za ovaj rad je korištenje Unity-ja bilo neupitno.



Sl. 2.1: Izgled Unity sučelja

2.2. Unity Networking (UNet)

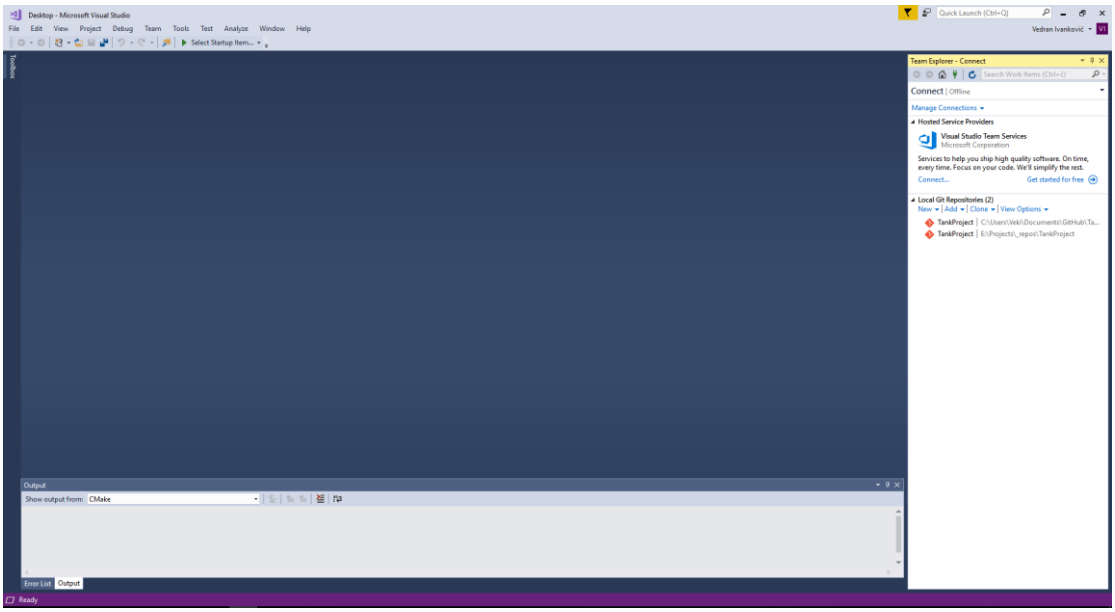
Unity Networking jedan je od servisa dostupnih unutar samog Unity-ja. Da bi se koristio potrebno je imati Unity ID i omogućiti multiplayer na stranici korisničkog računa za projekt i povezati ga unutar samog Unity Editora preko Collaboration prozora. UNet nudi mogućnost korištenja vrlo jednostavnog „high level“ API-ja koji omogućava kontrolu umreženog stanja, slanje i primanje umreženih poruka. Koristeći UNet može se stvoriti poslužitelj - klijent vrsta

igre u kojoj je poslužitelj ujedno i klijent koji igra. Jedan od najvažnijih elemenata UNeta je njegov „matchmaking“ servis koji olakšava pronalaženje igara preko interneta i spajanje igrača jednih s drugima preko relejskih servera kako sami igrači ne bi morali namještati postavke za svoj vatrozid da bi mogli igrati jedni s drugima.

2.3. Microsoft Visual Studio 2015

Microsoft Visual Studio integrirano je razvojno okruženje (engl. Integrated development environment – IDE) tvrtke Microsoft, sa širokom namjenom. Koristi se za razvoj računalnih programa, mrežnih mjesta, mrežnih aplikacija, mrežnih usluga i mobilnih aplikacija. Visual Studio koristi Microsoftove platforme za razvoj softvera kao što su Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Microsoft Visual Studio sadrži uređivač koda koji podržava IntelliSense[2] što uvelike olakšava proces pisanja koda i program za pronalaženje pogrešaka (engl. debugger). Sve ovo izuzetno nadopunjuje Unity u procesu razvoja, uvelike olakšavajući pisanje koda. Podržava razne programske jezike (njih 36), u određenoj mjeri, pod uvjetom da postoji servis za njih. Od ugrađenih jezika po postavljenom podržava C, C++ i C++/CLI, VB.NET, C# i F#.[4]

Microsoft Visual Studio integriran je s Unity-jem, C# kao programski jezik odlično je pokriven s Visual Studiom, prilikom korištenja Community inačice program je besplatan, itd. – sve navedeno razlozi su odabira Microsoft VS-a kao razvojnog okruženja za ovaj rad.



Sl. 2.2: Izgled Microsoft Visual Studio sučelja

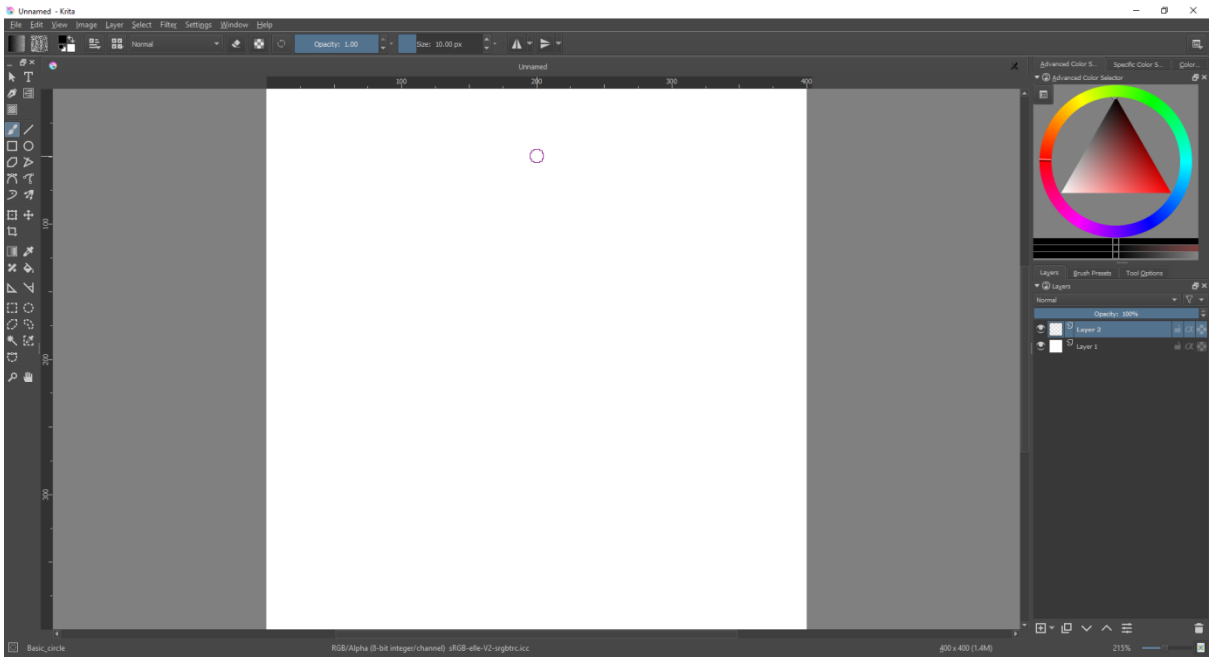
[2] IntelliSense – automatsko predviđanje i završavanje dijelova koda

2.4. Krita

Krita je besplatan program otvorenog koda za crtanje i slikanje. Služi kao alat dizajniran za digitalne umjetnike, ilustratore, umjetnike za teksturu i VFX industriju. Program je nastao 2005. godine i od tada se razvija. Program koristi Qt vizualni alat. Sadrži korisničko sučelje s niskom razinom ometanja, podršku za upravljanje bojama, nerazorne slojeve, podršku za vektorske radove i promjenjive profile prilagodbe. Može se koristiti na Linux-u, Microsoft Windows-u i MacOS-u.

Glavne značajke programa su: intuitivno korisničko sučelje koje, kako je već spomenuto, ne odvraća pažnju korisnika, a paneli se mogu premjestiti i prilagoditi za određeni tijek rada; stabilizatori kistova koji pomažu u ravnom crtanju (omogućuju smanjenje pokreta nastalih drhtavom rukom); „pop-up“ paleta koja se može koristiti za brzo biranje boja i kistova; itd.

Zbog navedenih značajki te činjenice da je program besplatan, otvorenoga koda i jednostavan za korištenje, Krita je odabrana kao alat za kreiranje svih objekata u igri (ambijent, igrač, poster u glavnom izborniku).[5]

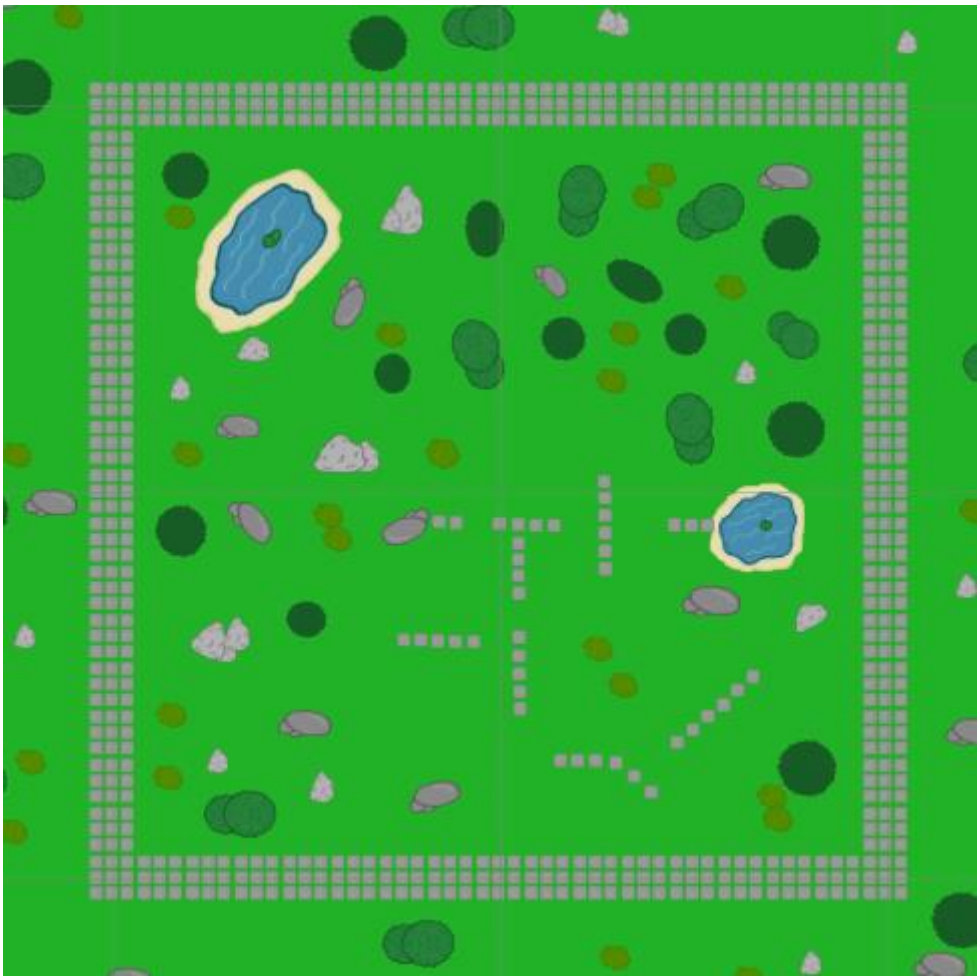


Sl. 2.3: Izgled Krita sučelja

3. RAZVOJ IGRE

3.1. Mehanika igre

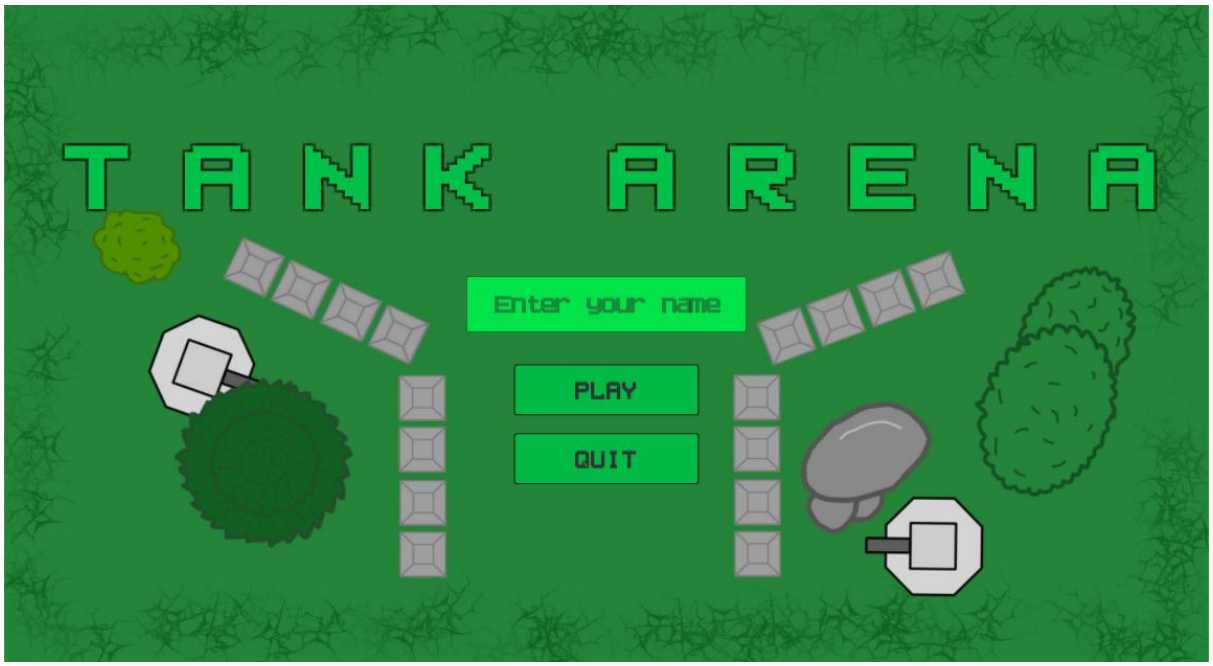
Igra je osmišljena prema konceptu popularnih igara na internetu na domenama koje završavaju s io. Na igru se gleda iz ptičje perspektive, jednostavnih je kontrola, zanimljivog sadržaja i grafike. Na početku igre, igrač se stvara nasumično na šumskoj mapi, gdje mora skupiti tri zlatne poluge kako bi pobijedio u igri. Zlatne se poluge mogu dobiti uništavanjem drugih igrača, a u slučaju da igrač sam biva uništen oživljava se brzo na novoj nasumično odabranoj poziciji s punim životnim bodovima. Municiju za pucanje ima u neograničenim količinama, jedino je ograničen s brzinom punjenja (2 sekunde). Sama mapa puna je prepreka za igrača, a neki su elementi i prepreka za projektile što daje igraču mogućnost strateškog skrivanja.



Sl. 3.1: Izgled mape u igri

3.2. Glavni izbornik

Glavni izbornik ujedno je i prva scena u igri. U sceni se nalazi objekt NetworkManager koji je glavni objekt (sa svojim priključnim komponentama) za umrežavanje igrača. Na njemu se nalazi polje za unos imena, gumb za otvaranje prozora izbornika za postavljanje igre i gumb za izlaženje iz igre. Za navigiranje kroz izbornik napravljena je skripta *MenuNavigator.cs* u kojoj se nalaze javne metode povezane na gumbove i polje za unos u izborniku.



Sl. 3.2: Izgled glavnog izbornika

Izbornik za postavljanje igre sastoji se od dijela za postavljanje igre i dijela za pronalazak i priključivanje u igru. Za postavljanje igre potrebno je u polje za unos unijeti ime sobe i pritisnuti gumb Create Game. Radnju i kreiranje sobe omogućava skripta *HostGame.cs* u kojoj se nalazi referenca na NetworkManager objekt i pritiskom na gumb Create Game poziva metodu *CreateRoom()* s naredbom *matchmaker.CreateMatch* koja stvara igru na matchmakeru, što dalje omogućava pronalazak igre i priključivanje na mreži. Za priključivanje u igru, dovoljno je pritisnuti gumb na listi Join Game, koji se mogu osvježiti po potrebi u slučaju da nema ni jedna igra dostupna na listi. Ove funkcije omogućava skripta *JoinGame.cs* sa svojim metodama. Za osvježavanje liste koristi se naredba *matchMaker.ListMatches()* koja u slučaju uspješnog pronalaska igre, poziva metodu *OnMatchList()* koja stvara UI element s potrebnim podacima za priključivanje u tu igru s naredbom *matchMaker.JoinMatch()*. Uz funkcionalnosti izbornika nalazi se i naputak kako

igrati igru, a izlazak iz izbornika za postavljanje igre i povratak na glavni izbornik omogućava gumb Main Menu.

Sl. 3.3: Izgled izbornika za postavljanje igre

3.3. Igra

Ulaskom u igru igrač se stvara nasumično na mapi i odmah počinje igrati. Za sva stvaranja objekata na mreži zadužen je Network Manager objekt koji sadrži informacije o umreženim objektima, u slučaju ovog projekta, to su objekt igrača i objekti projektila. Glavni objekt za logiku igre je GameManager objekt sa svojom priključnom skriptom. U njemu se nalaze informacije poput: koliko bodova igrač treba skupiti da bi završio igru, je li igra gotova te ime pobjednika; metoda za provjeru uvjeta pobjede i metoda za pozivanje kraja igre koja se šalje svim klijentima kada igrač skupi dovoljno bodova. Kada je igra gotova, igraču se pojavljuje novi prozor gdje mu javlja ime pobjednika i gumb za izlazak iz igre koji poziva metodu *LeaveGame()* s naredbom *matchMaker.DropConnection*() ako je igrač klijent i *StopHost()* ako je poslužitelj. Iz igre je moguće izaći i prije njenog kraja pritiskom Escape gumba na tipkovnici koji otvara novi izbornik gdje se nalazi gumb Disconnect.

3.4. Kamera

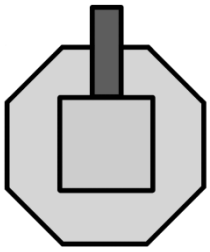
Kamera je važan objekt u hijerarhiji Unity-a jer pokazuje sve ostalo na sceni igraču. Kao i svaki drugi objekt u Unity-ju, ima transformacijsku komponentu te se može pomicati, a

mogu se i dodati druge komponente iz urednika, kao i skripte. Može biti perspektivna (koristi se u 3D igrama jer dosljedno prikazuje udaljenost objekata) ili ortografska (koristi se za 2D igre gdje udaljenost između objekata nije važna).

U ovom projektu koristi se ortografska kamera, a praćenje igrača omogućava skripta *FollowTarget.cs*. *FollowTarget* sadrži dvije varijable i funkciju naslijeđene iz *MonoBehaviour* klase.[6] Sve klase u Unity-u po pretpostavljenom nasljeđuju iz klase *MonoBehaviour*. Varijabla *target* je javna tipa GameObject (referira se na objekt u Unity uredniku) koja sadrži objekt koji će se slijediti na sceni i podešava se putem *PlayerSetup.cs* skripte (više u idućem poglavlju). Varijabla *offset* je tipa Vector3 koja sadrži udaljenost kamere od igrača (iako je ortografska ako se nalazi na istoj z koordinati kao i igrač, igrač neće biti vidljiv). Funkcija *Update()* poziva se jednom po okviru i u njoj ažuriramo vrijednost transformacije kamere kako bi pratila svoju metu u svakom okviru scene.

3.5. Igrač

Igrač je jedan od umreženih objekata u ovoj igri. Kako bi se definirao objekt igrača potrebno je napisati mnogo skripti i funkcionalnosti. Za potrebe ovog projekta napisane su skripte *Player.cs, PlayerSetup.cs, PlayerController.cs, PlayerWeapon.cs* i *PlayerScore.cs*, osim njih koriste se još komponente iz UnityNetworking API: *NetworkIdentity, Network Transform i Network Transform Child*. Network identity označava je li objekt od servera ili lokalnog igrača, Network Transform ažurira poziciju igrača u igri preko cijele mreže, a Network Transform Child ažurira poziciju objekta u hijerarhiji originalnog objekta (u slučaju da je objekt kompleksniji i sastoji se od više djece objekata), konkretno služi za ažuriranje rotacije topa igrača na mreži.



Sl. 3.4: Objekt igrača

Upravljanje igraćim objektom omogućavaju skripte *PlayerController.cs* i *PlayerWeapon.cs*. *PlayerController.cs* je skripta koja u *Update()* metodi „sluša“ korisnikov unos za pokretanje objekta koristeći klasu *Input* iz UnityEngine biblioteke u kojoj se nalaze globalne varijable za unos te ga pokreće koristeći klasu Rigidbody2D mijenjajući njegovu komponentu velocity. Ista skripta služi za ciljanje tako da računa vektor između igraćeg objekta i trenutne lokacije miša na ekranu, te tu vrijednost šalje u skriptu *TurretMotor.cs* koji se nalazi na objektu djetetu koji onda primjenjuje tu rotaciju na top igraćeg objekta, koji potom dalje biva ažuriran na mreži. *PlayerWeapon.cs* je skripta s referencama na projektil, pozicijska vrijednost gdje se projektil treba instancirati i kut pod kojim se projektil treba instancirati. Kao i skripta *PlayerController.cs,* sluša korisnikov unos za pokušaj iniciranja paljbe. Iniciranje paljbe je vremenski ograničeno primjenom varijabli *fireRate* i *canFire,* kada se paljba inicira, skripta poziva naredbu *CmdFire()* na serveru koja instancira projektil naredbom *NetworkServer.Spawn()*, potom šalje naredbu svim klijentima metodom *RpcFire()* kako bi i oni instancirali taj projektil.

Na skripti *Player.cs* nalazi se vitalnost igrača (životni bodovi), metoda za smanjivanje životnih bodova i metoda za ponovno oživljavanje kada životni bodovi dosegnu nulu. Varijabla *maxHealth* označava maksimalni mogući broj životnih bodova, a *currentHealth* trenutni. CurrentHealth je sinkronizirajuća varijabla koja proizlazi iz Unity Networkinga, prilikom svake promjene, vrijednost se automatski ažurira preko svih klijenata. Postoji i javna varijabla tipa bool *IsRespawning* koja označava oživljava li se igrač trenutno i služi za gašenje igračevih kontroli dok se ponovno ne oživi. Da bi igrač izgubio trenutne životne bodove, potrebno je pozvati metodu *CmdTakeDamage(int amount)* koja zatim šalje svim klijentima preko *RpcTakeDamage(int amount).* Kada životni bodovi dosegnu nulu, igračev objekt se blokira, nestaje sa scene, a potom se oživljava pozivajući *Spawn()* metodu koristeći *Invoke()* naredbu nakon par sekundi.

PlayerScore.cs je skripta koja sadrži ime igrača i njegove trenutne bodove. Povezana je s GameManager.cs skriptom tako da poziva njegovu *CheckForVictory(int score, string playerName)* svaki put kada igrač osvoji bod i Network Managerom preko *PlayerName.cs* skripte koja joj dodjeljuje ime. Sadrži i metodu *GiveVictoryPoint()* koja se poziva kada igrač biva uništen i instancira objekt koji daje pobjednički bod. Network Manager se kao objekt ne uništava promjenom scena u igri te služi za prijenos korisničkih postavki sa igraćeg izbornika u igru.

PlayerSetup.cs je možda i najvažnija skripta od svih jer sadrži logiku koji je igrač lokalni, a koji udaljeni. Kako ne bi upravljali objektom drugog igrača u lokalnoj kopiji igre, PlayerSetup.cs provjerava odmah na početku koji je igrač lokalni, a koji ne, te za nelokalnog igrača onemogućuje skripte *PlayerWeapon.cs* i *PlayerController.cs.* Za lokalnog igrača podešava kameru da ga slijedi i instancira korisničko sučelje na kojem se nalazi trenutno stanje životnih bodova igrača i njegovi bodovi te ih povezuje sa pripadajućim komponentama gdje se nalaze ti podaci.

3.6. Projektili i pobjednički bodovi

Zbog kašnjenja između poslužitelja i klijenta, postoje dvije vrste projektila u igri. Poslužiteljski, koji zapravo oduzima životne bodove i klijentski koji se instancira odmah lokalno kako igrač ne bi imao kašnjenje između ispaljivanja projektila i pojave projektila. Lokalni projektil, koji sadrži skriptu *Dummy.cs,* ima sve jednake funkcionalnosti kao i poslužiteljski, jedino što on ne oduzima životne bodove. Na poslužiteljskom projektilu nalazi se skripta *Projectile.cs* u kojoj se nalazi metoda OnTriggerEnter2D(Collider2D) (naslijeđena iz MonoBehaviour klase) koja se poziva prilikom kolizije s drugim objektom. Ako je objekt u koliziji tipa Player, poziva metodu *CmdTakeDamage()* iz *Player.cs* komponente i tim putem primjenjuje promjenu životnih bodova. Ovim putem onemogućena je mogućnost varanja s klijentske strane jer se kolizija i oduzimanje životnih bodova događa samo na serveru, a potom sinkronizira na sve klijente.

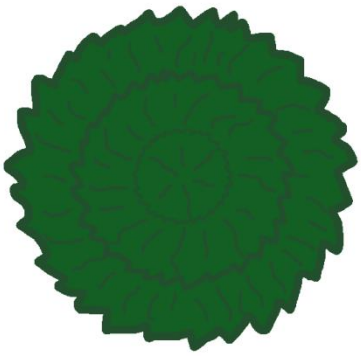
Sl. 3.5: Projektil

Kako bi igrač dobio pobjednički bod, mora uništiti drugog igrača i pokupiti zlato koje uništeni igrač ostavlja. Kada igrač primi fatalni udarac *Player.cs skripta* iz metode *RpcTakeDamage()* poziva naredbu *GiveVictoryPoint()* u *PlayerScore.cs* skripti koji instancira objekt sa skriptom *VictoryPoint.cs.* Kolizija s tim objektom poziva naredbu

GetVictoryPoint() u *PlayerScore.cs* skripti koja daje jedan bod score varijabli, a zatim provjerava u GameManageru je li uvjet za pobjedu postignut.

3.7. Ostali objekti i efekti

Ostali objekti u igri služe za estetiku same igre. To su drveće, kamenje, jezera, kameni blokovi i grmovi. Drveće i kamenje kao objekti ne sadrže ručno napisane skripte nego su sačinjene od gotovih komponenti koje se nalaze u Unity. To su komponente *Collider2D* i *Rigidbody2D*, oboje komponente iz MonBehaviour klase. Collider2D definira površinu koja se sudara s drugim objektima koji sadrže Collider2D svojstvo, a Rigidbody2D daje svojstvo fizike u igri (kao gravitacija i sile, iako gravitaciju u ovom projektu ne koristi). Objekti s komponentom Rigidbody2D „sudarat“ će se s projektilima, dok drugi neće.

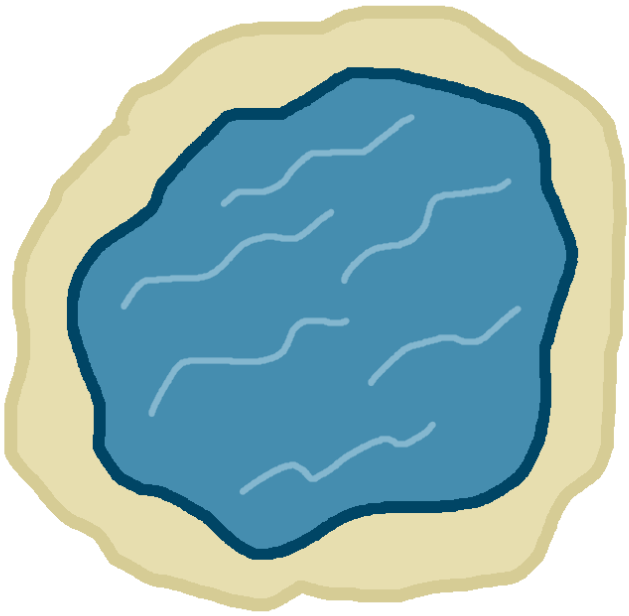


Sl. 3.6: Drvo

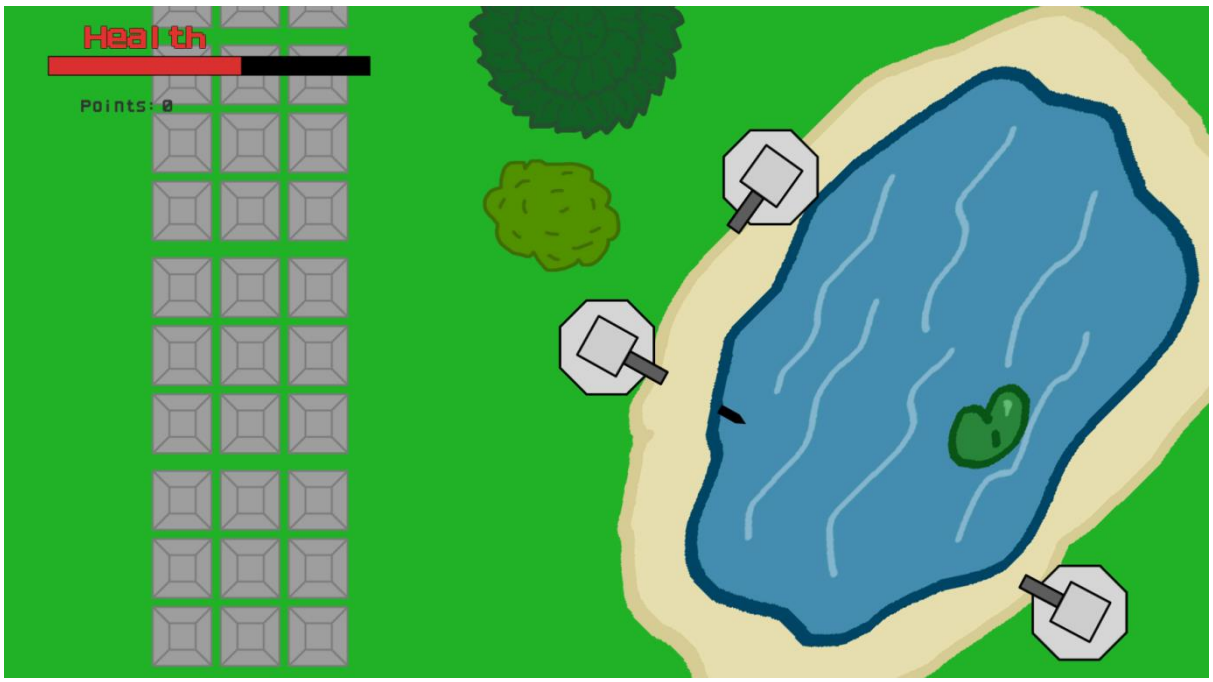
Grmovi u igri imaju efekt podrhtavanja kada se prelazi preko njih. Taj efekt lagano je postignut koristeći još jedan od dostupnih alata Unity Editora, Animator. Grm sadrži komponentu Collider2D, Animator i skriptu *Bush.cs.* Prilikom kolizije u skripti Bush.cs poziva se metoda OnTriggerEnter2D (Collider2D) te u Animatoru postavlja okidač IsColliding koji onda okida animaciju podrhtavanja. Animacije se lagano podešavaju u Editoru mijenjajući komponente na tom objektu (u ovom slučaju skalu ili veličinu samog objekta).

Sl. 3.6: Grm

Još jedan objekt u igri sadrži animaciju a to je jezero. Na jezeru se nalazi list koji slobodno pluta. Sastoji se samo od komponente Collider2D i Animation. Po pretpostavljenom odmah od iniciranja objekta pokreće se animacija i stalno se vrti te nema potrebe za pisanje posebnih skripti kako bi se kontrolirala animacija.



Sl. 3.7: Jezero



Sl. 3.8: Prikaz igre u tijeku

4. ZAKLJUČAK

U ovom završnom radu prikazan je i pojašnjen cijeli proces razvoja Tank Arena igre koristeći Unity Engine u C# programskom jeziku. Izradom igre steknuto je vrijedno znanje objektno - orijentiranog programiranja, C# programskog jezika, osnove dizajna i umrežavanja.

Unity je odlična platforma za razvoj zbog široko dostupnih elemenata, uputa, dokumentacije kao i odličnih zajednica na forumima koje pokrivaju mnogo pitanja i odgovora za određene situacije, kao i ljude koji su spremni brzo odgovoriti na postavljeni upit na bilo koju temu vezanu uz dizajn igre i Unity Engine-a.

Unity Networking odličan je alat za početak učenja osnova umrežavanja i implementaciju jednostavnih koncepata za više igrača. Osim što nudi gotove komponente, nudi i mogućnost proširivanja i poboljšanja istih budući da kašnjenje između poslužitelja i klijenata zna biti veliko, što znatno utječe na kvalitetu same igre i korisničkog iskustva.

Kako bi projekt ušao u komercijalne vode i došao na jednu od domena io, bilo bi potrebno još nadograditi s dodatnim elementima kako bi se proširio sadržaj i korisničko iskustvo, tipa dodatne mape, određena pojačavanja igrača, mogućnost komunikacije, imena igrača u samoj igri i sl. Trebale bi se i dodatno optimizirati funkcije Unity Networkinga da bi se smanjio utjecaj kašnjenja između poslužitelja i klijenta te implementirati još određenih razina zaštite protiv varanja.

LITERATURA

[1] W. Goldstone, Unity Game Development Essentials, Packt Publishing Ltd., Birmingham, 2009.

[2] https://en.wikipedia.org/wiki/Unity_(game_engine), lipanj 2018.

[3] https://unity3d.com, lipanj 2018.

[4] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio, lipanj 2018.

[5] https://www.designbombs.com/adobe-photoshop-alternatives/, veljača 2019.

[6] https://docs.unity3d.com/ScriptReference/MonoBehaviour.html, lipanj 2018.

SAŽETAK

Rad opisuje postupak izrade 2D igre za više igrača (engl. multiplayer) u Unity Engine-u koristeći programski jezik C#. Glavni cilj igre Tank Arena je skupiti određeni broj bodova. To se može postići poražavanjem drugih igrača i skupljanjem zlatnika koji ostanu nakon što je drugi igrač jednom poražen. U igri može sudjelovati ukupno četiri igrača, svi jedni drugima protivnici. Na samom početku potrebno je upisati željeni nadimak te, ili kreirati novu „sobu“ kojoj će se drugi igrači pridružiti, ili odabrati već postojeću kreiranu od strane drugog igrača. Nakon toga započinje igra u kojoj se upravlja standardnim kontrolama za računalne igre (W, A, S, D), a projektile se ispucava lijevim klikom miša. Protivnika je potrebno pogoditi pet puta (od strane bilo kojeg igrača) kako bi on za sobom ostavio zlatnik. Zlatnik može pokupiti bilo tko, pa su spretnost i brzina također komponente potrebne prilikom igranja ove igre. Nakon što jedan od igrača pokupi tri zlatnika, tj. skupi tri boda, igra je završena.

Za izradu velike većine vizualnih sadržaja (drveće, grmlje, kamenje, itd.) korišten je program otvorenog koda za crtanje i slikanje Krita. Sav vizualni sadržaj je ili originalno autorovo djelo ili preuzeto s interneta s pravima za slobodnu upotrebu.

Ključne riječi: 2D, C#, Krita, multiplayer, Tank Arena, Unity

ABSTRACT

This paper describes the process of making a 2D multiplayer game in Unity Engine using the C# programming language. The main goal of the game Tank Arena is to collect a certain number of points. This can be achieved by defeating other players and collecting the gold coins remaining after the other player is defeated. A total of four players can participate in the game, all of them opponents. At the beginning, it is neccesary to enter the players nickname and to either create a new "room" for the other player's to join, or select an existing one created by another player. After that, the game begins. To move the tank around the map the standard controls for computer games are used (W, A, S, D), and the missiles are fired with the left mouse button. Opponents need to be hit five times (by any player) to leave a gold bar. The gold bar can be picked up by anyone, so skill and speed are also the components needed when playing this game. Once one of the players collects three gold bars, IE three points, the game is over.

To create the vast majority of the visual contents (trees, bushes, stones, etc.), an open code program for drawing and painting Krita was used. All visual content is either an original author's work or downloaded from the internet with rights for free use.

Keywords: 2D, C#, Krita, multiplayer, Tank Arena, Unity

ŽIVOTOPIS

Vedran Ivanković rođen je u Vinkovcima 03.01.1993. Završio je OŠ Josipa Kozaraca Vinkovci 2007. godine nakon čega upisuje Gimnaziju (jezičnu) Matije Antuna Reljkovića u Vinkovcima. Godine 2012. upisuje Elektrotehnički fakultet u Osijeku, stručni studij elektrotehnike, smjer informatika. Kroz studiranje i proučavanje sadržaja na internetu, te općenito zanimanje za igre i način na koji se razvijaju, počinje se zanimati za *Game Design* i u slobodno vrijeme na mreži uči kako koristiti Unity. Na taj je način i nastala ideja za izradu igre putem spomenute platforme.