Izrada igre obrane tornjevima u programskom alatu Unity

Alagić, Aldin

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Aldin Alagić

IZRADA IGRE OBRANE TORNJEVIMA U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Aldin Alagić

Matični broj: 44924–16R

Studij: Informacijski sustavi

IZRADA IGRE OBRANE TORNJEVIMA U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Mentor:

Dr. sc. Mladen Konecki

Varaždin, rujan 2019.

Aldin Alagić

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom radu je detaljno prikazan kompletan proces razvoja jednostavne računalne igre koja se ubraja u žanr igara koji nosi naziv „obrana tornjevima“ (eng. *Tower defense*). Igra je razvijena u višeplatformskom programskom alatu „Unity“. Općenito, rad sadrži opis osnovnih elemenata i koraka u razvojnom procesu koji su popraćeni isječcima programskog koda i vizualnim prikazima različitih dijelova igre, kao i modela koji su u njoj korišteni. Rad je strukturiran na način da se započinje s kratkim opisom žanra „obrana tornjevima“, a potom se prelazi na opis najpoznatijih predstavnika navedenog žanra. Nakon toga slijedi opis alata korištenih u izradi igre. Preostali i najvažniji dio rada, fokusiran je na opis same igre i njenog procesa razvoja. U tom dijelu se detaljno opisuju algoritmi i mehanika na kojoj igra počiva te način na koji su oni realizirani u pomoću Unityja i programskog jezika C#. Cijeli programski kod napisan je u razvojnom okruženju „Visual Studio“. Također, prilikom dizajna pojedinih grafičkih elemenata korišteni su alati za grafički dizajn „Adobe Photoshop“ i „Adobe Illustrator“.

Ključne riječi: Unity; C#; računalna igra; modeliranje; algoritam; skripta; objekt;

Sadržaj

1. Uvod ….....1
2. Opis žanra „obrana tornjevima“.....2
2.1. Povijesni razvoj žanra 3
2.2. Najpoznatiji predstavnici žanra 4
2.2.1. Defense Grid: The Awakening 4
2.2.2. PixelJunk Monsters 4
2.2.3. Plants vs. Zombies 5
3. Opis korištenih alata 6
3.1. Unity …..…..6
3.2. Microsoft Visual Studio 8
3.3. Adobe Photoshop 10
3.4. Adobe Illustrator.....11
4. O igri „Legends of Inanis“.....12
4.2. Radnja …..…..12
4.2. Scenariji 13
4.2.1.Slučajevi korištenja.....15
5. Razvoj igre „Legends of Inanis“.....17
5.1. Sučelja ..…..18
5.2. Izbornik 20
5.3. Razina … 23
5.4. Kamera 23
5.5. Neprijatelji.....25
5.6. Valovi neprijatelja.....27
5.7. Tornjevi 28
6. Zaključak …..33
Popis literature 34
Popis slika ……..…..38

1. Uvod

Od nastanka prve računalne igre „OXO“, koju je razvio Alexander Douglas 1952. godine, pa sve do danas, svijet računalnih igara doživio je čitav niz drastičnih promjena. Među najvećim promjenama ubrajaju se promjene u sklopovlju potrebnom za pokretanje igara, uređaja koji se koriste pri igranju, alata i tehnika za razvoj igara i sve veća raznovrsnost u igrama [1]. Ta raznovrsnost ogleda se u nastanku velikog broja žanrova u proteklih 60 godina. Među najpopularnijim žanrovima računalnih igara današnjice ističu se [2]:

- akcijske igre,
- avanturističke igre,
- igre igranja uloga (eng. *Role-playing games*),
- simulacijske igre,
- sportske igre,
- strategijske igre,
- trkaće (simulacije vožnje) igre i
- edukacijske igre.

Od ostalih načina kategoriziranja računalnih igara ističe se podjela na osnovu perspektive ili gledišta koju igrač posjeduje unutar igre [2]:

- igre u prvom licu,
- igre u trećem licu,
- izometrične igre,
- platformske igre,
- igre klizanja u stranu (eng. *Side-Scrolling games*) i
- igre odozgo prema dolje (eng. *Top-down games*).

U zadnje vrijeme sve popularnije su strategijske igre, koje zahtijevaju osmišljavanje dugoročne strategije za izgradnju vojske ili sakupljanja sirovina kako bi se u konačnici postigao određeni cilj, primjerice, porazila protivnička vojska. Sve to čini ove igre zahtjevnim, ali i veoma zanimljivim i privlačnim.

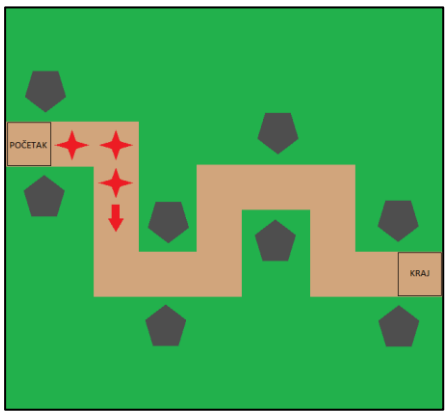
Igra koja je razvijena u sklopu ovog završnog rada spada u igre „obrane tornjevima“ koje predstavljaju podvrstu strategijskih računalnih igara. Ono što čini igre iz žanra „obrana tornjevima“ specifičnim je njihova jednostavnost, u smislu da ne zahtijevaju posebnu vještinu i iskustvo od igrača, već naprotiv, one su većinom lagane i može ih igrati bilo tko. Način igranja i pravila su veoma jednostavna i brzo se usvajaju, ali za potpuno savladavanje svih koncepata i strategija potrebno je dosta vremena i truda [3].

2. Opis žanra „obrana tornjevima“

Kao što je već spomenuto „obrana tornjevima“ (OT) je podvrsta strategijskog žanra računalnih igara. Elementi drugih vrsta strategija koji su vidljivi kod ove vrste igara je izvođenje u stvarnom vremenu, što je značajka strategije u stvarnom vremenu (eng. *Real-time strategy*), ili igranje u potezima, što je značajka strategije na poteze (eng. *Turn-based strategy*).

Igra se sastoji od mape s točno određenim putem koji se proteže kroz cijelu mapu. Cilj igre je spriječiti neprijatelja da dođe do kraja navedenog puta izgradnjom obrambenih građevina ili tornjeva uz put kojim se neprijatelj kreće [4]. Neprijatelji napadaju u valovima koji najčešće nastupaju u točno određenim vremenskim intervalima. Svaki val se razlikuje po broju, jačini ili tipu neprijatelja koji se u njemu nalaze, s tim da valovi postupno jačaju i postaju sve veći izazov za igrača [3]. U slučaju da pojedini neprijatelj uspije doći do kraja puta, igrač gubi život ili određeni dio životne energije. Prilikom borbe, igrač dobiva određenu količinu novca za svakog poraženog protivnika. Igraču preostaje da dobro razmisli o strategiji koju će koristiti prilikom trošenja skupljenog novca na kupovinu, pozicioniranje i nadogradnju tornjeva [4]. Tornjevi se razlikuju po jačini projektila, dometu i specijalnim sposobnostima. S druge strane neprijatelji se razlikuju po jačini, brzini kretanja, otpornosti na određene napade i veličini potencijalne nagrade u slučaju njihovog poraza [4]. U prvim igrama ove vrste igrač je imao izometričnu ili odozgo prema dolje perspektivu, dok modernije inačice su većinom 3D vrste [3].

Osim navedenih značajki, modernije inačice ove igre sadrže i određene novine: mogućnost da tornjevi budu napadnuti i popravljeni, kretanje neprijatelja različitim putevima (ne postoji točno određeni put kojim se neprijatelj kreće), izgradnja prepreka na putu, izgradnja građevina koje proizvode novac i sl. Ovo sve predstavljaju dodatke igri koje su proizvođači koristili kako bi učinili igru zanimljivijom.



Slika 1: Općeniti prikaz igre „Obrane tornjevima“

Slika 1 sadrži prikaz jedne uobičajene igre OT. Dakle, postoji određeni put kojim se neprijatelji kreću i on je označen smeđom bojom. Uz put se nalazi prostor za izgradnju tornjeva i on je označen zelenom bojom. Neprijatelji koji se kreću putem predstavljeni su s crvenom bojom, dok su tornjevi prikazani sa sivom bojom. Cilj neprijatelja je doći do kraja puta, dok je zadatak igrača da ih zaustavi u njihovom naumu.

2.1. Povijesni razvoj žanra

Ovaj žanr vuče korijene iz arkadne igre „Rampart“, koju je proizveo Atari 1990. godine. Cilj ove igre bio je obraniti dvorac od neprijatelja izgradnjom i popravljanjem obrambenih građevina [3]. Kao što je i slučaj kod današnjih igara ovog žanra, neprijatelj je napadao dvorac u nizu valova. Ovaj strategijski klasik stekao je veliku popularnost, što je za posljedicu imalo njegovo izdavanje na brojne druge platforme, kao što su Playstation 2, Xbox i Gamecube [4].

Međutim, mnogi smatraju da prva istinska OT igra bila realizirana u okviru igre „Warcraft III: The Frozen Throne“ [3]. Ova igra je imala skrivenu razinu s mini igrom koja je sadržavala brojne značajke modernih OT igara. Neke od tih značajki su: napad neprijatelja u valovima, predefinirani put, izgradnja i pozicioniranje tornjeva uz taj put, sakupljanje novca ili bodova ubijanjem neprijatelja. Popularnosti žanra, dodatno su doprinijeli korisnici sa svojim vlastitim OT kartama ili razinama za popularne igre, kao što su „Starcraft“, „Age of Empires II“ i „Warcraft III“ [5]. Ove karte i razine su stekle toliku popularnost da su ih i sami proizvođači počeli uključivati u svoje igre [5].

Žanr dostiže vrhunac popularnosti naglom pojavom velikog broja web igrica izrađenih pomoću Adobe Flasha. U tom periodu nastaju i brojne web OT igre, od kojih se najviše istaknula igra „Desktop Tower Defense“ iz 2007. godine [4]. Igra je stekla veliku popularnost, što dokazuje činjenica da je u prva četiri mjeseca igrana preko 15 milijuna puta. Sve to kulminiralo je u osvajanje „Gleemax nagrade za strategijsko igranje“ (eng. *Gleemax Award for Strategic Gameplay*) [6]. U narednom periodu nastalo je veliki broj različitih igara s potpuno novim temama i značajkama. Nastajale su igre smještene u srednjem vijeku, modernom dobu pa čak i u budućnosti.

Igre ovog žanra nastavile su biti popularne i nakon pojave igara za mobilne uređaje. Najznačajniji za napredak ovog žanra bili su iPhone uređaji, koji su prvi pokazali tvrtkama da je moguće igrati omiljene igre na mobilnom uređaju [7]. Nešto kasnije OT igre postaju popularne i na Android uređajima. Prve popularnije mobilne OT igre bile su „Fieldrunners“ i „Anomaly: Warzone Earth“ [6]. Danas, OT igre se redovno pojavljuju na vrhu ljestvica najpopularnijih igara na Google Play i App Store.

2.2. Najpoznatiji predstavnici žanra

Vremenom se pojavilo zaista veliki broj igrica ovog žanra. Zbog sličnosti korijenskih pravila i načina igranja, proizvođači su morali stalno smišljati nove dodatke i unapređenja klasičnog OT stila. Na taj način nastao je veliki broj različitih OT igara. Više o najpoznatijim i najutjecajnijim igrama ovog žanra slijed u narednom poglavlju.

2.2.1. Defense Grid: The Awakening

Riječ je o OT klasiku, kojeg je proizveo „Hidden Path Entertainment“ 2008. godine. Igra (Slika 2) je izdana na Microsoft Windows i Xbox 360 platformama [8]. Cilj igre je obraniti *napojne* jezgre (eng. *Power cores*) od vanzemaljaca koji napadaju u valovima. Igra se sastoji od 20 razina, a u svakoj razini neprijatelj napada u više valova [9]. U igri postoji 11 vrsta neprijatelja, od kojih svaki posjeduje posebne osobine i mogućnosti [9]. Neprijatelji se ne kreću linearnim putem, već uvijek biraju najkraći mogući put do pojedine jezgre [8]. Igrač ima na raspolaganju 10 vrsta tornjeva s različitim mogućnostima, s tim da je svaki toranj moguće nadograditi i do dva puta [9]. Tornjeve je moguće graditi samo na određenim platformama. Tornjevi se grade pomoću bodova koji se stječu uništavanjem neprijatelja. U slučaju da neprijatelj uspije doći do jedne od jezgri, on je počinje nositi prema izlazu [8]. Ako je uništen neprijatelj koji prenosi jezgru, ona se polako vraća na početnu poziciju. Igrač je izgubio ako neprijatelji odnesu sve jezgre.

Slika 2: Igra "Defense Grid: The Awakening" [8]

2.2.2. PixelJunk Monsters

Neobična igra svjetlo dana ugledala je 2008. godine, kad je izdana od strane tvrtke „Q-Games“ za Playstation 3 [10]. Za razliku od ostalih OT igara tog perioda, „PixelJunk Monsters“ je bio prava novina. Naime, u igri (Slika 3) krećete se kroz šume jednog otoka i pritom nastojite obraniti svoje pleme [11]. Igra se sastoji od 21 razine, a moguće ih je igrati na tri različite težine [10]. Svaki neprijatelj koji uspije doći do baze oduzima život jednom pripadniku plemena. Igrač je izgubio u slučaju da pleme ostane bez svih pripadnika. Uz put kojim se neprijatelji kreću

nalaze se drveća koja služe kao pozicije za izgradnju tornjeva [11]. Postoji veliki broj tornjeva s različitim sposobnostima, kao što je usporavanje neprijatelja ledom i spaljivanje većeg broja neprijatelja istovremeno [10]. Igrač na raspolaganju ima novčiće i dragulje [11]. Novčiće dobiva uništavanjem neprijatelja, a koristi ih prilikom izgradnje tornjeva. S druge strane, dragulji se koriste za nadogradnju tornjeva ili istraživanje naprednijih tornjeva. Igrač upravlja likom kojim gradi i nadograđuje tornjeve na drveću u svojoj neposrednoj blizini.

Slika 3: Igra "PixelJunk Monsters" [11]

2.2.3.Plants vs. Zombies

Ovaj kulturni fenomen izdan je 2009. godine od strane „PopCap Games“ [12]. Srž igre (Slika 4) je pomalo smiješna, ali iznenađujuće zanimljiva, bitka između biljaka i zombija. Iako se na prvu sve ovo čini dosta čudnim, igra je zbog veoma dobrog dizajna i balansiranog sustava borbe stekla veliku popularnost [12]. Igraču je za obranu vrta stavljeno na raspolaganje čak 48 vrsta biljaka, što je daleko veći broj od bilo koje prijašnje OT igre [13]. S druge strane, igrač se nastoji obraniti od 26 različitih vrsta zombija [13]. Sve to zahtjeva dosta strateškog razmišljanja od strane igrača. Za razliku od klasičnih igara ovog žanra, neprijatelji u ovoj igri nemaju prethodno definirani put kretanja, već postoji šest redova kojima se zombiji mogu kretati [13]. Također, postoje značajne razlike u načinu „kupovine“ biljaka, gdje kod većine ostalih igara žanra tornjevi se grade na osnovu novca skupljenog uništavanjem neprijatelja, u „Plants vs. Zombies**“** igrač mora sakupljati sunčevu svijetlost ili saditi suncokrete [14].

Slika 4: Igra "Plants vs. Zombies" [12]

3. Opis korištenih alata

Tijekom razvoja igre korišteno je više različitih alata. Najvažniji alat prilikom razvoja igre bio je programski alat „Unity“ koji predstavlja njeno razvojno okruženje. Međutim, za potrebe razvoja logike i upravljanjem drugim složenijim aspektima igre, korišten je programski jezik C# koji je napisan unutar alata „Microsoft Visual Studio“. Razvoj igre zahtijevao je i određenu dozu grafičkog dizajna u rasterskoj i vektorskoj grafici, zbog čega su korišteni alati „Adobe Photoshop“ i „Adobe Illustrator“.

3.1. Unity

Unity, jedan od najkorištenijih alata u industriji video igara, rezultat je velikog napora i truda tri mlada programera. David Helgason, Joachim Ante i Nicholas Francis su nakon dugog i napornog rada 2005. godine izdali prvu inačicu alata odnosno inačicu 1.0 [15].

Unity 2.0 izdan je 2007. godine [16]. Ova inačica donijela je čak 50 novih dodataka, kao što je pokretač (eng. *Engine*) za detaljna 3D okruženja, dinamičke sjene u stvarnom vremenu, razne vrste osvjetljenja te rad sa zvukom [16]. Ovaj, u početku primitivan alat, do 2008. godine postao je veoma napredan i koristan [15]. Godine 2010. izdana je inačica 3.0, koja je imala još napredniju grafiku i mogućnosti [17]. Do 2012. godine Unity je koristilo preko 1,3 milijuna korisnika [18]. Iste godine izdana je i inačica 4.0. Ona je podržavala „DirectX 11“ i „Adobe Flash“ [19]. Kvaliteta ovog alata prepoznata je 2014. godine, kad mu je dodijeljena nagrada „Best Engine“ u Velikoj Britaniji na svečanosti „Develop Industry Excellence Awards“ [20]. Velik niz promjena donijela je peta inačica alata odnosno Unity 5.0. Najznačajnija poboljšanja bila su u vidu osvjetljenja i zvuka [21]. Još jedna novina bio je „Cinematic Image Effects“ [21]. Došlo je i do značajnog poboljšanja u performansi alata. Nakon ovog izdanja dolazi do promjene načina nazivanja novih izdanja alata, gdje budući nazivi sadrže godinu izdavanja. Sukladno s tim, sljedeće izdanje nazvano je Unity 2017, koje je sadržavalo novi pokretač za renderiranje grafike u stvarnom vremenu, gradnju svijeta (eng. *Worldbuilding*), prikaz informacija o performansi i analitici u stvarnom vremenu i niz drugih dodataka [22]. Najnovije izdanje alata je Unity 2018. Ovo izdanje sadrži još bolju podršku za stvaranje napredne grafike. Prateći trenutne trendove, alat je pruža i podršku za rad s strojnim učenjem putem svog „Imitation Learninga“ [23].

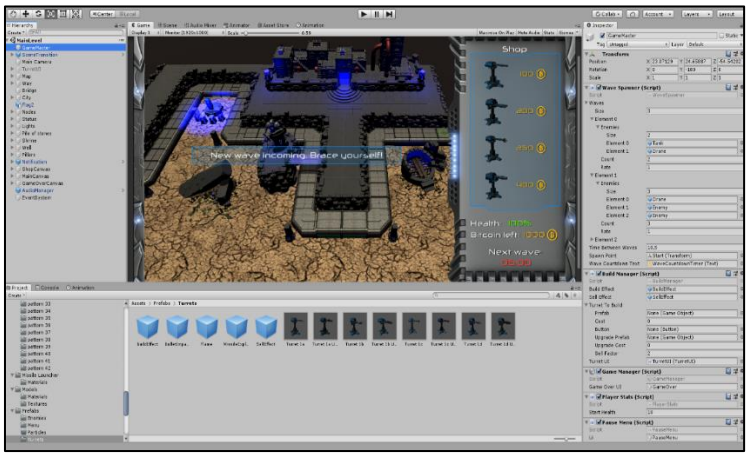
Danas „Unity Technology“, tvrtka koja stoji iza alata, zapošljava oko 300 osoba, a sam alat podržava razvoj igara za iOS, Android, Windows, Mac, Linux, PS3, Xbox 360, Wii U i Web tražilice [15].

Projekt izrađen u Unityu se sastoji od različitih scena. Svaka scena predstavlja dio igre, kao što je glavni izbornik ili pojedina razina. Alat korisniku daje „povuci i spusti“ (eng. *drag and drop*) okruženje za izradu igara [15]. Iako je moguće razviti igru bez pisanja i jedne linije programskog koda, većina igara zahtjeva određenu količinu programskog koda koji je organiziran u pojedine skripte. Prilikom pisanja programskog koda korisnik se može koristiti C#, JavaScript i Boo programskim jezicima [15]. Cijeli Unity funkcionira kao objektni sustav utemeljen na komponentama [15]. Dakle, svaki objekt igrice (eng. *Game object*), bilo da je to neki predmet, osoba ili osvjetljenje, može biti povezan s jednom ili više komponenti koje određuju njegovo ponašanje. U Unityu postoji velik broj ugrađenih komponenti, ali za naprednije mogućnosti potrebno je izraditi vlastite komponente. Takve komponente implementiraju se korištenjem skripti i programskog koda [24]. Unity posjeduje „Prefab“ sustav, koji omogućava izradu, izmjenu i spremanje objekata sa svim njegovim komponentama, vrijednostima i podobjektima [25]. Kako bi upravljanje i izmjena objekata unutar igre bilo što jednostavnije i brže moguće je koristiti „The Inspector Window“, prozorčić koji prikazuje sve detalje o odabranom objektu [26]. Prilikom razvoja igre u ovom alatu veoma su bitni i modeli koji će se u njoj koristiti. Modele je na nešto složeniji način moguće napraviti u samom alatu, ali zbog jednostavnosti većinom se koriste drugi specijalizirani alati kao što je „Blender“. Što se tiče animiranja objekata u igrici, Unity posjeduje opcije „Animation“ i „Animator“ koje pojednostavljuju izradu i upravljanje animacijama [27]. Također, prilikom izrade pojedinih grafičkih elemenata, korišteni su vanjski grafički uređivači kao što su „Adobe Photoshop“ i „Adobe Illustrator“. Kod izgleda pojedinih objekata u igrici, veoma su bitni materijali koji određuju boju objekata. U razvoju igre značajni su i zvukovi, kao što je pozadinska muzika ili zvuk određenih animacija.

Modeli, materijali i zvukovi se na veoma jednostavan način mogu uvesti u Unity te potom koristiti u razvoju igre. Veliki broj gotovih modela moguće je pronaći i preuzeti s trgovine „Unity Asset Store“. Međutim, treba napomenuti da se većina ovakvih modela plaća.

Na slici 5 imamo prikaz Unityevog uređivača u kojem je otvoren projekt igre koja se razvija u okviru ovog rada. S desne strane prozora nalazi se već spomenuti „Inspector“ koji sadrži detalje o odabranom objektu, u ovom slučaju to je „GameMaster“. Kao što vidimo, objekt ima „Transform“ komponentu te pet vlastitih komponenti odnosno skripti. Kod svake skripte moguće je vidjeti njene javne varijable, a kod nekih i kolekciju objekata skripte kao što je slučaj kod skripte „WaveSpawner“. S lijeve strane nalazi se popis objekata trenutno pokrenute scene. Iznad popisa nalaze se različiti alati za upravljanje kamerom i objektima. Na samom dnu prozora nalazi se pregled „imovine“ (eng. *assets*) projekta zajedno s pretraživačem. Ona predstavlja sve što koristimo u projektu. Dakle, svi uvezeni i izrađeni *prefabovi*, modeli, materijali, slike, efekti, zvukovi, animacije, skripte i sl. Ovdje možemo otvoriti i dodatne

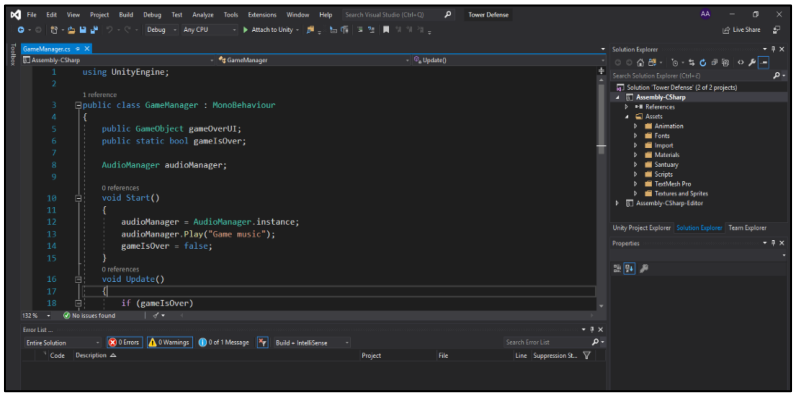
prozorčiće kao što je konzola za prikaz grešaka pri radu. U središtu prozora nalazi se trenutno otvorena scena te tipke za pokretanje, zaustavljanje i prijelaz na sljedeću scenu. Također, ovdje možemo otvarati dodatne prozorčiće za prikaz scene prilikom pokretanja, animacija, trgovine za „imovinu“ i sl.



Slika 5: Sučelje Unity uređivača

3.2. Microsoft Visual Studio

Kao što je već prethodno spomenuto, Unity podržava programski jezik C#. Pisanje programskog koda za potrebe Unity projekta u prošlosti se vršilo unutar njegove integrirane razvojne okoline „MonoDevelop“ [28]. Međutim, od 2018. godine ona više nije podržana. Trenutno, najkorištenije okruženje za pisanje C# programskog koda u ovu svrhu je „Microsoft Visual Studio“ (Slika 6). Ovo okruženje ima dobru podršku za Unity razvojnu platformu te „UnityEngine“ aplikacijsko programsko sučelje za skripte (eng. *Scripting API*).



Slika 6: Sučelje Microsoft Visual Studija

Kako bi mogla normalno funkcionirati, svaka skripta mora imati uključen imenski prostor (eng. *namespace*) „UnityEngine“ koji sadrži osnovne klase za rad s Unityom. Skripta svoju

vezu s projektom ostvaruje kroz klasu koja nasljeđuje ugrađenu klasu `MonoBehaviour` iz imenskog prostora `UnityEngine` [24]. Ova klasa predstavlja nužnu osnovu za izrađivanje komponente, koja će kasnije određivati ponašanje jednog od objekata unutar projekta. Svaki put kada se izradi nova skripta unutar Unitya, generira se istoimena klasa koja sadrži sljedeći kod:

```
using UnityEngine;
using System.Collections;
public class PrimjerKlasa : MonoBehaviour
{
    void Start()
    {
    }

    void Update()
    {
    }
}
```

Kao što vidimo na početku prikazanog koda uključen je prethodno spomenuti imenski prostor `UnityEngine`. Uz njega je uključen `System.Collections` koji je potreban za rad s kolekcijama kao što su polja, liste i redovi. U ostatku koda imamo generiranu klasu `PrimjerKlasa`, koja nasljeđuje klasu `MonoBehaviour`. Klasa sadrži funkcije `Start()` i `Update()` koje su naslijeđene iz klase `MonoBehaviour`. Funkcija `Start()` se poziva se samo jednom i to na samom početku izvršavanja skripte, odnosno u trenutku kada se omogući njeno izvršavanje [24]. Ova funkcija najčešće se koristi za inicijalizaciju varijabli i druge slične radnje. Ovdje je bitno napomenuti da se u ovim klasama ne koriste konstruktori iz klasičnog programiranja, jer njihovu zadaću obavlja sam uređivač [24]. Iako je moguće koristiti konstruktore to se ne preporučuje jer može prouzrokovati velike probleme u radu projekta. Funkcija `Update()` poziva se kontinuirano nakon svakog kadra (eng. *Frame*) , s tim da se prvi poziv dešava tek nakon poziva funkcije `Start()` [24]. Ova funkcija idealna je za smještanje programskog koda koji se izvodi tijekom cijelog rada igre, kao što je kretanje objekta, izvršavanje određenih akcija, provjera unosa od strane korisnika i sl.

`MonoBehaviour` sadrži niz drugih funkcija koje se pozivaju pod određenim uvjetima i one imaju veoma bitnu ulogu kod razvoja igre. Najznačajnije takve funkcije koje su korištene i u razvoju igre u okviru ovog rada su: `Awake()`, `StartCoroutine()`, `OnEnable()`, `OnDisable()`, `OnMouseDown()`, `OnMouseEnter()`, `Instantiate()`, `Destroy()` i `GetComponent()` [29]. Funkcija `Awake()` predstavlja funkciju koja se poziva prva, čak i prije funkcije `Start()`. Ona se poziva prilikom inicijalizacije objekta na koji je skripta povezana, neovisno o tome je li sama skripta aktivna ili ne [30]. Značajna je i funkcija `StartCoroutine()`, koja pokreće izvršavanje određene korutine (eng. *Coroutine*). Korutine,

za razliku od običnih funkcija, mogu se izvoditi tijekom više kadrova jer imaju mogućnost pauziranja i nastavljanja svog izvođenja u sljedećem kadru [31]. Funkcije `OnEnable()` i `OnDisable()` pozivaju se prilikom omogućavanja odnosno onemogućavanja objekta. Funkcije `OnMouseDown()` i `OnMouseEnter()` pozivaju se prilikom klika i ulaska pokazivača u prostor objekta, točnije u objektov „GUIElement“ ili „Collider“. Funkcija `Instantiate()` koristi se za inicijalizaciju objekta igre, dok se funkcija `Destroy()` koristi za *uništavanje* objekta igre.

Klasa sadrži i svojstva koja imaju značajnu ulogu u razvoju igre. Od tih svojstava prilikom razvoja igre korištena su svojstva: „gameObject“, „transform“, „tag“ i „name“ [29]. Za referenciranje povezanog objekta koristi se svojstvo „gameObject“, dok za referenciranje oznake i naziva objekta koristimo svojstva „tag“ i „name“. Svojstvo „transform“ koristi se za referenciranje objektovog „Transforma“ koji se koristi za rad s pozicijom, rotacijom i razmjerom objekta.

Od ostalih imenskih prostora koji su bitni za rad s Unityjem korišteni su imenski prostori `UnityEngine.SceneManagement`, `UnityEngine.UI` i `UnityEngine.Audio`. Imenski prostor `UnityEngine.SceneManagement` koristi se prilikom rada sa scenama, odnosno za kreiranje, *uništavanje*, poziv, spajanje i učitavanje scena. Prilikom rada s elementima grafičkog sučelja, kao što su platna, tekstovi, tipke i sl., potrebno je uključiti imenski prostor `UnityEngine.UI`. Klase i funkcije za rad sa zvukovima i tonovima sadržane su u imenskom prostoru `UnityEngine.Audio`.

Veoma bitna i praktična mogućnost koju pruža Unity je mogućnost pregleda i izmjene vrijednosti varijabli unutar uređivača, s tim da one prvo moraju biti deklarirane kao javne varijable u programskom kodu putem „public“ modifikatora pristupa. Ova mogućnost često je korištena tijekom cijelog razvoja igre u okviru ovog rada, pogotovo kod testiranja i prilagođavanja pojedinih aspekata igre kao što je količina novca, životna energija, svojstva neprijatelja i neprijateljskih valova. Također, omogućila je veoma jednostavno i brzo referenciranje ostalih objekata u igri. Često je korišten atribut `System.Serializable` koji omogućava kreiranje i upravljanje nad kolekcijom objekata određene klase iz uređivača. Primjeri korištenja svih spomenutih stavki bit će navedeni i dodatno objašnjeni u kasnijim poglavljima rada.

3.3. Adobe Photoshop

Prilikom razvoja igre u Unityju važnu ulogu imaju različiti grafički elementi, kao što su slike, ikone i materijali. Sam alat ima minimalne mogućnosti u smislu grafičkog dizajna navedenih

elemenata. Zbog toga se često koriste vanjski grafički uređivači. Jedan takav alat je „Adobe Photoshop“. Riječ je o grafičkom uređivaču za rad s rasterskom grafikom koji je u svom prvom izdanju svijetlo dana ugledao 1988. godine [32]. Autori alata su Thomas i John Knoll, a prava za distribuciju pripadaju tvrtki „Adobe Systems Inc.“ [32]. Ovaj alat smatra se daleko najpoznatijim i najkorištenijim alatom za rad s rasterskom grafikom. Posljednje izdanje alata je „Photoshop CC 2019“. U okviru ovog rada, alat je korišten prilikom izrade i izmjene materijala koje koriste različiti objekti u igri, poput tornjeva i neprijatelja.

3.4. Adobe Illustrator

Rad s određenim grafičkim elementima zahtijevao je korištenje dodatnih grafičkih uređivača. Tako je za rad s vektorskom grafikom korišten „Adobe Illustrator“, veoma popularni grafički uređivač koji je izdan 1987. godine za Mac i 1989. godine za Windows [33]. Ono što je „Adobe Photoshop“ za svijet rasterske grafike, „Adobe Illustrator“ je u svijetu vektorske grafike. Alat je trenutno spada među aplikacije „Adobe Creative Clouda“. Ovaj alat je korišten u radu s različitim ikonama i slikama, čije je očuvanje kvalitete bilo iznimno važno.

4. O igri „Legends of Inanis“

Cilj rada bio je razviti igru koja sadrži većinu značajki karakterističnih za ovaj žanr, ali opet učiniti ju na neki način posebnom i zanimljivom. Isto tako, igra bi trebala biti jednostavna i pristupačna za igranje osobama koje nemaju posebno iskustvo u igrama, s tim da savladavanje cijele igre ipak zahtijeva dosta truda i razmišljanja. Kako bi se postigli zadani ciljevi, bilo je potrebno pronaći pravu mjeru zahtjevnosti neprijateljskih valova i jačine obrambenih građevina. Također, igra je na engleskom jeziku. Na taj način igra bi bila pristupačna široj publici.

Cilj igre je obraniti ulazna vrata od neprijatelja koji se kreću u valovima koji periodički nastupaju. Valovi se međusobno razlikuju po broju i vrsti neprijatelja te po brzini njihovog pojavljivanja. Valovi se kreću unaprijed definiranim putem. U igri postoje četiri vrste neprijatelja koji se razlikuju po brzini kretanja i veličini životne energije. Svaki neprijatelj koji uspije doći do kraja puta smanjuje energiju vrata. Ako energija padne na nulu, igrač je izgubio. Igrač pobjeđuje ako preživi sve neprijateljske valove. Tornjevi se kupuju s novcem, točnije kriptovalutom „Bitcoin“, koja se dobiva uništavanjem neprijatelja. Igraču je na raspolaganje stavljeno četiri vrste tornjeva, odnosno obrambenih građevina s različitim vrstama naoružanja. Tornjevi se kupuju i pozicioniraju na povišeni prostor koji se pruža uz stranu puta kojim se neprijatelji kreću. Tornjeve je moguće, uz odgovarajuću cijenu, nadograditi kako bi postali još veća prijetnja neprijateljima. Na taj način pojačavaju se njihove sposobnosti, kao što je jačina projektila, brzina pucanja i domet. Također, tornjevi se mogu prodati kako bi se vratio dio uloženog novca. Igra je nazvana „Legends of Inanis“, razlog tomu bit će dodatno objašnjen u idućem poglavlju rada.

4.1. Radnja

Teško je odabrati izreku koja bi najbolje opisala prilike koje su zahvatile čovječanstvo u drugoj polovini 21. stoljeća. Bilo da je to Einsteinova „ne znam s kojim će se oružjem ratovati u trećem svjetskom ratu, ali u četvrtom svjetskom ratu ratovat će se grančicama i kamenjem“ ili Platonova „samo mrtvi su vidjeli kraj rata“, jedno je sigurno, ratovi su se nastavili, čak i onda kad su doveli čovječanstvo do ruba egzistencije. Godina je 2089. nekoć ogromna brojka od 8 milijardi ljudi nastanjenih u više od 130 država i još mnogo više gradova, svedena je na svega 500 000 i jedan grad, Zion, posljednje utočište čovječanstva. Zadnjeg značajnijeg tehnološkog napretka sjećaju se samo najstariji stanovnici Ziona, jer svaki pomisao o visoko naprednoj i prosperitetnoj budućnosti izgubljena je u ruševinama koje su zamijenile prelijepe gradove

nekoć preplavljene s bezbroj ljudskih duša. Sve što je ostalo je zastarjela tehnologija, građevine i, naravno za neke i dalje najvažnije, oružje s konca prve polovine 21. stoljeća.

I onda kada su napaćeni ljudi Ziona mislili da stvari ne mogu postati gore, pojava vatrenih bljeskova jedne sasvim mirne i obične noći pokazat će im koliko su zapravo bili u krivu. Vatrene bljeskove proizvele su strane letjelice koje su velikom brzinom prodirale slojeve zemljine atmosfere. I tako, posljednji od velikih gradova čovječanstva, našao se oči u oči s novom prijetnjom, možda i svojom posljednjom. Sve što su znali o svom neprijatelju je kratko upozorenje koje je emitirano na svim radio frekvencijama i jeziku poznatom ljudima: „Mi smo Scev'ean, pratili smo vas, znamo vas i presudili smo vam. Došao je kraj vama i vašim iskvarenim običajima, predajte se i dočekajte ga s barem malo dostojanstva“.

Scev'eani predstavljaju jednu od najstarijih i najmoćnijih rasa u galaksiji. Riječ je o kibernetičkim organizmima koji su vođeni samo jednim ciljem, osigurati svoj opstanak pod svaku cijenu, čak i ako to znači istrebljenje bezbroj drugih rasa, koje oni smatraju nižim i iskvarenim. Scev'eani nastoje promatrajući, učeći i pokoravajući druge rase ispraviti svoje nedostatke i riješiti neke od temeljnih problema koji im stoje na putu do besmrtnosti. Iz ovog i proizlaze riječi koje su utkane u srž njihove kulture: „Naučiti. Usavršiti. Opstati“. Zbog svoje građe, ali i bezobzirnosti prema životu drugih, poznati su širom galaksije pod nazivom „Inanis“, što u prijevodu znači: „oni bez duše“.

Sudbina cijelog čovječanstva na vašim je ramenima…

4.2. Scenariji

Kako bi se što jasnije predstavila igra koja se želi izraditi u sklopu ovog rada i mogućnosti koje bi igrač trebao imati unutar nje, osmišljena su dva scenarija, koja obuhvaćaju cijeli smisao igre. Ovi scenariji prikazuju najvažnije tokove događaja kroz koje igrač prolazi prilikom pokretanja i igranja ove igre.

1. **Scenarij:** Promjena postavki igre
   Učesnik: Ind3Z (Igrač)
   Početni uvjeti: Igrač je uspješno pokrenuo igru i nalazi se u njenom glavnom izborniku
   Tok događaja:
   Igrač „Ind3Z“ želi pokrenuti novu igru, ali prvo želi provjeriti jesu li sve postavke igre u skladu s njegovim potrebama. Odabire opciju „Settings“, što u prijevodu znači postavke. Otvara mu se izbornik s četiri opcije:
   1. Gameplay – Opcija s postavkama vezanim za težinu i način igranja
   2. Video – Opcija s grafičkim postavkama

3. Audio – Opcija s zvučnim postavkama
4. Back – Opcija koja vraća igrača na početni izbornik

Igrač „Ind3Z“ ima već dosta prethodnog iskustva u igrama ovog žanra pa želi učiniti igru zahtjevnijom kako bi mu razine bile što izazovnije, a samim tim i zanimljivije. Iz tog razloga igrač odabire opciju „Gameplay“. Otvara mu se novi izbornik s različitim opcijama vezanim za početne postavke u igri, kao što su težina igre, količina novca, životna energija neprijatelja i sl. Korisnik postavlja najveću moguću težinu, što ujedno prilagođava i ostale postavke izbornika. Ipak, igrač smatra da je početna količina novca preniska pa ju malo povećava. Igrač sprema postavke i vraća se u prethodni izbornik pomoću tipke „Close“, što u prijevodu znači zatvaranje.

Prilikom izmjene prethodnih postavki, igrač je primijetio da trenutna kvaliteta grafike nije zadovoljavajuća. Odabire opciju „Video“, koja otvara novi izbornik s postavkama za rezoluciju, kvalitetu grafike i igranje u punom zaslonu (eng. *Fullscreen*). Igrač povećava kvalitetu i uključuje opciju „Fullscreen“. Igrač sprema postavke i vraća se u prethodni izbornik pomoću tipke „Close“, što u prijevodu znači zatvaranje.

Također, smatra da je trenutna razina pozadinskog zvuka preglasna i želi to ispraviti. Odabire opciju „Audio“, koja otvara novi izbornik koji sadrži postavke za glavni zvuk, muziku i zvučne efekte. Odlučuje smanjiti glasnoću muzike, a povećati glasnoću zvučnih efekata. Igrač sprema postavke i vraća se u prethodni izbornik pomoću tipke „Close“. Igrač je sada potpuno zadovoljan postavkama igre te odabire opciju „Back“, koja ga vraća na početni izbornik.

Izlazni uvjeti: Promjene su uspješno pohranjene i igrač se vratio u glavni izbornik

2. **Scenarij:** Igranje početne razine
   Učesnik: Ind3Z (Igrač)
   Početni uvjeti: Igrač je uspješno pokrenuo igru i nalazi se u njenom glavnom izborniku
   Tok događaja:

Igrač „Ind3Z“ želi početi s igranjem pa odabire opciju „New Game“, što u prijevodu znači nova igra. Zaslon se zamračuje, a zatim se prikazuje početna razina igre s kartom na kojoj se nalazi put kojim će se neprijatelji kretati te prostor na koji igrač može postavljati tornjeve. Na desnoj strani zaslona nalaze se različite opcije i informacije. Između ostalog, tu se nalazi prikaz trgovine koja sadrži različite tornjeve koje igrač može kupiti. Ispod trgovine nalaze se informacije o trenutnom stanju igre, kao što je trenutna količina novca, stanje ulaznih vrata i vrijeme do početka novog

napada. Igrač u bilo kojem trenutku može na pritisak tipke pauzirati igru i pozvati izbornik koji sadrži opcije poput nastavljanja s igranjem ili povratka na početni izbornik. Također, izbornik sadrži informacije o trenutnoj igri, kao što je broj preživjelih neprijateljskih valova, broj uništenih neprijatelja i sl.

Isteklo je vrijeme i započinje prvi napad odnosno neprijateljski val. Igrač odabire jedan od tornjeva koje može kupiti s trenutnom količinom novca odnosno „Bitcoina“. Nakon pronalaska idealne pozicije, igrač postavlja odabrani toranj. Količina novca se smanjuje, a toranj se priprema za nadolazeće neprijatelje. Neprijatelji potom ulaze u domet tornja, koji započinje s paljbom. Neprijatelji su uništeni, a količina novca se povećala za odgovarajuću količinu. Igrač je preživio prvi val neprijatelja.

Iz iskustva igrač zna da će sljedeći val neprijatelja biti još jači te procjenjuje da će mu za uspješnu obranu biti potreban jači toranj. On se potom odlučuje na nadogradnju postojećeg tornja. Vrijeme je isteklo i neprijatelji se opet počinju pojavljivati, ali sada u većem broju. Igrač pomoću ojačanog tornja ne uspijeva uništiti cijeli val i nekoliko neprijatelja se uspije provući pored tornja. Ti neprijatelji dolaze do kraja puta, gdje se nalaze ulazna vrata u grad Zion. Oni znatno oštećuju ulazna vrata žrtvujući se. Igrač s nešto manje uspjeha preživljava još jedan val neprijatelja.

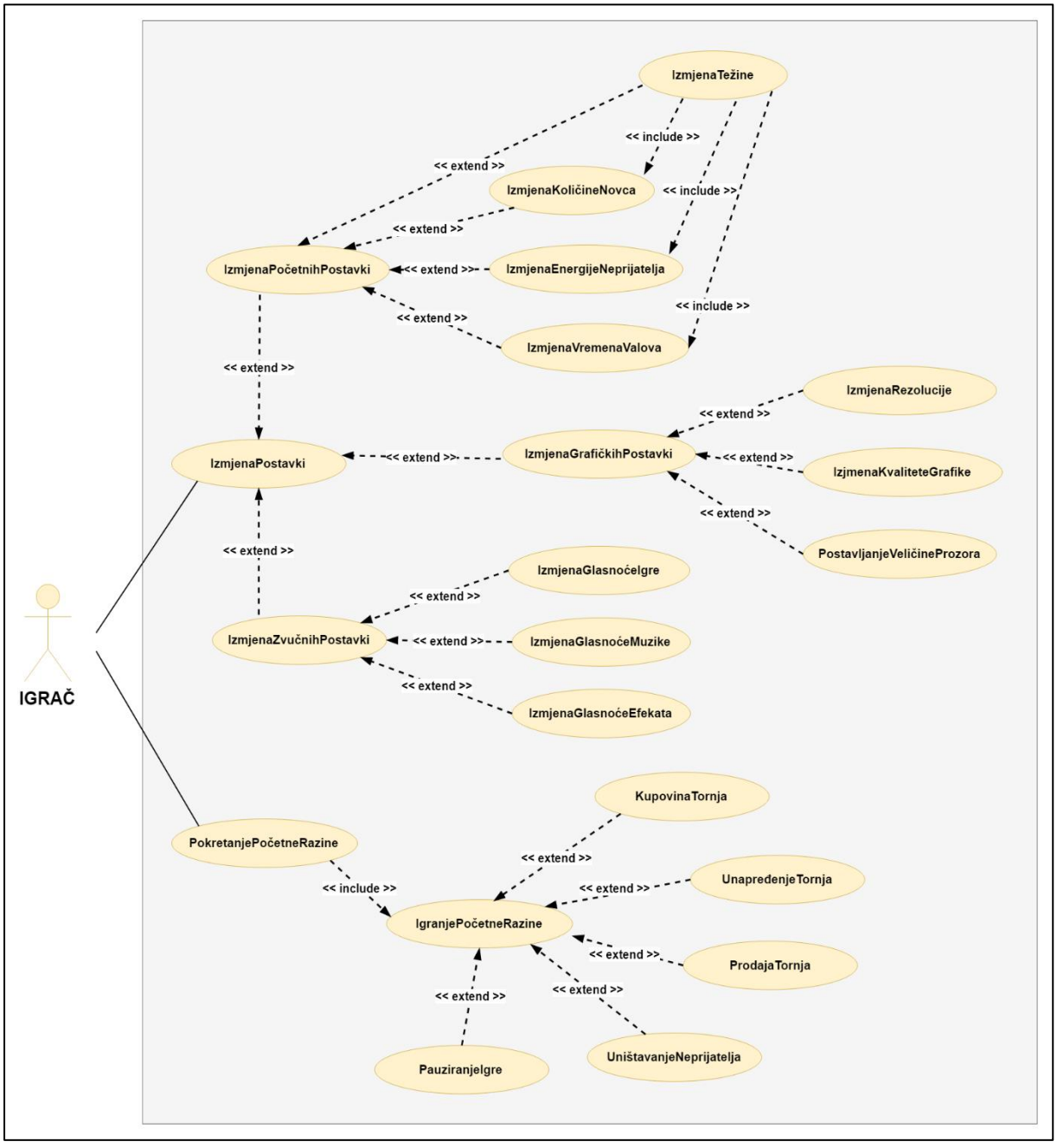
Igrač želi sada ojačati svoj obrambeni sustav s još jačim tornjem, ali nema dovoljno novca da ga kupi. Zbog tog razloga igrač prodaje prethodno postavljeni toranj, ali na njegovo iznenađenje količina novca opet nije dovoljna za kupovinu novog željenog tornja. Naime, igrač nije uračunao da je prodajna cijena tornjeva niža. Neprijatelji iskorištavaju igračev propust i dolaze do ulaznih vrate te ih potom uništavaju. Zaslon se zamračuje i prikazuje se obavijest o porazu igrača i izbornik koji sadrži statističke informacije o odigranoj igri i različite opcije. Igrač može iz tog izbornika započeti novu igru ili vratiti se na početni izbornik.

Izlazni uvjeti: Igrač je završio s igranjem određene razine i vratio se u glavni izbornik

4.2.1. Slučajevi korištenja

Opisani scenariji dobro prikazuju smisao igre i sve mogućnosti koje igrač ima na raspolaganju prilikom igranja. Spomenute scenarije moguće je na formalnije prikazati korištenjem UML dijagrama ponašanja. Kako bi se stekao pregledniji prikaz mogućnosti igrača, izrađen je UML dijagram slučajeva korištenja (eng. *Use Case diagram*). Ova vrste UML dijagrama opisuje što naš sustav radi. Slučaj korištenja je akcija koju pojedini sustav može izvoditi u međudjelovanju s učesnicima, a označavamo ga s elipsom. Učesnik je skup podudarajućih uloga koje korisnici imaju prilikom interakcije sa jednim od slučajeva korištenja.

Između ovih koncepata postoje veze tipa uključi (eng. *include*) i proširi (eng. *extend*). Veza tipa uključi označava da glavni slučaj korištenja sadrži ponašanje slučaja korištenja kojeg *uključuje*. Veza tipa proširi označava da se ponašanje glavnog slučaja korištenja proširuje ponašanjem drugog slučaja korištenja. Dijagram je prikazan na slici 7. Kao što vidimo dijagram opisuje što naša aplikacija radi. Postoji jedan učesnik i to je igrač. Glavni slučajevi korištenja su „IzmjenaPostavki“ i „PokretanjePočetneRazine“. Ovi slučajevi korištenja prošireni su s nizom slučajeva korištenja koje smo već prethodno objasnili kroz opis mogućih scenarija.



Slika 7: Dijagram slučajeva korištenja

5. Razvoj igre „Legends of Inanis“

Prvi korak u razvoju igre „Legends of Inanis“ bio je učitati potrebne modele, materijale, slika, fontova i zvukove. Kao što je već prethodno navedeno, većina ovog moguće je izraditi unutar uređivača, ali to zahtijeva jako mnogo vremena i vještine. Zato se koriste drugi alati koji donekle olakšavaju taj proces. Za izradu pojedinih modela, materijala i slika korišteni su spomenuti „Adobe Photoshop“, „Adobe Illustrator“ i sam uređivač. Međutim većina ostalih elemenata preuzeto je s „Unity Asset Storea“:

- Extreme Sci-Fi LITE (Autor: Aquarius Max)
- Free Sound FX (Autor: Soundbits)
- Modern GUI Skin (Autor: 3D.Rina)
- Sanctuary (Autor: Purple Jump)
- Sci-fi Interface Frames (Autor: CGY - Yemelyan K.)
- Sci-Fi Platforms – 3D Models (Autor: Sevastian Marevoy)
- SciFi Simple Gun Turret Set (Autor: Tarko 3D)
- Tower Defense Template (Autor: Unity Technologies)
- Unity Samples: UI (Autor: Unity Technologies)
- Yughues Free Ground Materials (Autor: Nobiax / Yughues)
- Yughues Free Metal Materials (Autor: Nobiax / Yughues)

Kombinacija sadržaja navedenih pakete, zvukova preuzetih sa stranice „freesound.org“ i samostalnog rada korištena je kako bi se realizirala ciljana igra. Veoma bitan dio procesa razvoja igre bio je i programski kod koji se odvija u njenoj pozadini. Veliku pomoć prilikom programiranja pružili su video uredci od autora Brackeys [34]. Cijeli programski kod koji se krije iza igre organiziran je u sljedeće skripte, koje su zapravo klase:

- ApplicationManager – Omogućava pokretanje razine ili gašenje igre iz izbornika.
- AudioManager – Koristi se za upravljanje zvukovima unutar igre.
- BuildManager – Omogućava gradnju i prodaju tornjeva.
- Bullet – Realizira cijelu logiku iza pojedinog projektila.
- CameraController – Koristi se za upravljanje kamerom.
- DestroyOnExit – Zadužena za pokretanje animacije i uništenja neprijatelja.
- Difficulty – Koristi se pri postavljanju težine igre.
- Enemy – Realizira cijelu logiku iza pojedinog neprijatelja.
- EnemyMovement – Omogućava kretanje neprijatelja po definiranom putu.
- GameManager – Provjerava stanje i završava igru u slučaju poraza igrača.

- GameOver – Realizira logiku vezanu uz „GameOver“ izborik.
- HealthUI – Upravlja s vizualnim prikazom stanja energije na korisničkom sučelju.
- MoneyUI – Upravlja s vizualnim prikazom stanja novca na korisničkom sučelju.
- Node – Realizira cijelu logiku iza pojedinog mjesta za postavljanje tornja.
- Notification – Omogućava prikaz notifikacija na korisničkom sučelju.
- PanelManager – Realizira cijelu logiku iza početnog izbornika.
- PauseMenu – Realizira logiku vezanu uz „PauseMenu“ izbornik.
- PlayerStats – Upravlja s ostvarenim rezultatom igrača.
- SceneTransition – Omogućava tranzicije između scena.
- Settings – Omogućava izmjenu postavki igre.
- Shop – Realizira cijelu logiku iza trgovine.
- Sound – Omogućava definiranje različitih zvukova koji će se koristiti u igri.
- StatisticsUI – Upravlja s vizualnim prikazom statistike igrača.
- TiltWindow – Omogućava promjenu nagiba prozora u početnom izborniku.
- Turret – Realizira cijelu logiku iza pojedinog tornja.
- TurretBlueprint – Omogućava postojanje različitih vrsta i osobina tornjeva.
- TurretUI – Upravlja vizualnim prikazom opcija vezanih uz pojedini toranj.
- Wave – Realizira cijelu logiku iza pojedinog vala neprijatelja.
- WaveSpawner – Omogućava stvaranje neprijateljskih valova.

5.1. Sučelja

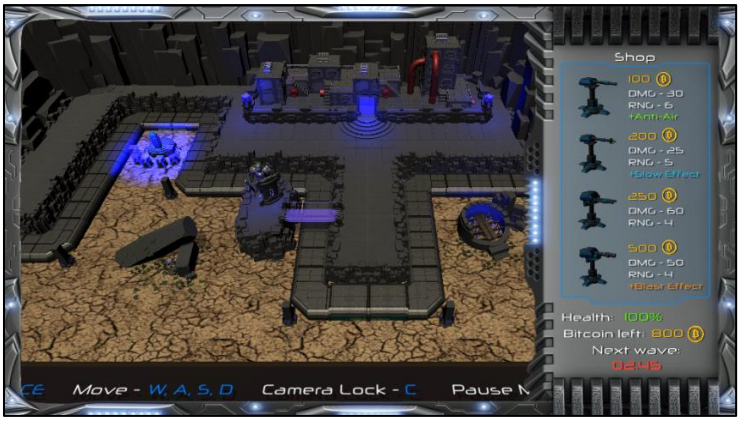
Igra se sastoji od dvije scene s pripadnim sučeljima. Prilikom pokretanja igre otvara se njen početni izbornik koji čini njenu prvu scenu. Sučelje izbornika prikazano je na slici 8. Kao što vidimo igrač ima na raspolaganju niz opcija. Igrač može započeti novu igru, odabrati razinu koju želi igrati, izmijeniti postavke i napustiti igru.

Slika 8. Početni izbornik igre

Odabirom opcije postavke, otvara se novi izbornik (Slika 9) u kojem može odabrati koju vrstu postavki želi mijenjati. Igrač može izmijeniti početne postavke, grafiku te zvuk igre. Kod početnih postavki moguće je promijeniti težinu igre, početnu količinu novca, životnu energiju neprijatelja i vremenski razmak između valova neprijatelja. Promjena težine igre uzrokuje i promjenu ostalih početnih postavki.

Slika 9. Izbornik za postavke igre

Nakon što igrač pokrene novu igru. Pokreće se sljedeća scena, odnosno početna razina igre. Sučelje igre prikazano je na slici 10. Kao što vidimo na slici je prikazana početna razina s okvirom koji sadrži različite informacije i opcije za igrača. Moguće je vidjeti trenutnu količinu novca koju igrač posjeduje, stanje ulaznih vrata te vrijeme do sljedećeg vala. Također, ovdje se nalazi i trgovina s tornjevima koje igrač može kupiti. Prilikom svih važnijih događaja igraču se prikazuju obavijesti. U ovom slučaju obavještava se da je započeo novi neprijateljski val. Ispred razine nalaze se i uputstva za kontrole vezane za igru.



Slika 10. Sučelje igre prilikom igranja

5.2. Izbornik

Prilikom izrade početnog izbornika korišten je predložak izbornika iz paketa „Unity Samples: UI“. Izbornik je prilagođen igri, tako da su pojedine opcije izbačene ili dodane. Također, potpuno su izmijenjene dostupne postavke koje igrač može mijenjati. To je zahtijevalo izradu vlastitih tipki i polja za unos, kao što je padajući izbornik. Sve ove opcije nisu imale nikakvu pozadinsku logiku, već je to posebno programirano. Izbornik nije imao nikakvih zvukova, već je dodatno implementirana pozadinska muzika i zvučni efekti prilikom kliktanja i prelaženja iznad opcija izbornika. Također, zvučni efekti dodani su prilikom korištenja klizača. Izmjena postavki sada sadrži popratnu informaciju o trenutnoj vrijednosti postavke.

Najbitniju ulogu kod izbornika imaju tri objekta: „ApplicationManager“, „MenuManager“ i „Settings“. Prvi je zadužen za pozadinsku muziku i izlazak iz igre. Drugi objekt je odgovoran za cijelu logiku iza izbornika. On omogućava otvaranje, zatvaranje, pomjeranje izbornika i pokretanje nove igre. Treći objekt omogućava izmjenu postavki igre. Programski kod koji je dodatno implementiran nalazi se u skriptama „ApplicationManager“ i „PanelManager“ i „Settings“.

Skripta: ApplicationManager

Isječak koda:

```
⋮
8.  private void Awake()
9.  {
10.   audioManager = AudioManager.instance;
11. }
12.
13. void Start()
14. {
15.    audioManager.Stop("Game music");
16.    audioManager.Play("Menu music");
17. }
18.
19. public void Quit()
20. {
21.   #if UNITY_EDITOR
22.     UnityEditor.EditorApplication.isPlaying = false;
23.   #else
24.     Application.Quit();
25.   #endif
26. }
```

Na trećoj liniji programskog koda deklarira se varijabla `audioManager` tipa `AudioManager`, koja se potom unutar funkcije `Start()` instancira. Ova varijabla se na 12. i 13. liniji programskog koda koristi za zaustavljanje prethodne muzike te puštanje pozadinske muzike za izbornik. Na ovaj način i ostale skripte upravljaju zvukovima. Klasa sadrži i funkciju

`Quit()` koja omogućava gašenje igre prilikom odabira opcije za izlaz iz igre. Ova funkcija koristi pretprocesorsku direktivu, koja omogućava da se dio programskog koda izvršava samo na određenim platformama [35]. U ovom slučaju koristi se direktiva `UNITY_EDITOR`, što znači da se taj dio koda izvodi samo unutar uređivača.

Skripta: PanelManager

Isječak koda:

```
⋮
28. public void OnEnable()
29. {
30.   m_OpenParameterId = Animator.StringToHash(k_OpenTransitionName);
31.   if (initiallyOpen == null)
32.     return;
33.   OpenPanel(initiallyOpen);
34. }
35.
36. public void OpenPanel(Animator anim)
37. {
38.   if (m_Open == anim)
39.   return;
40.   anim.gameObject.SetActive(true);
41.   var newPreviouslySelected = EventSystem.current.curentSelectedGameObj;
42.   anim.transform.SetAsLastSibling();
43.   CloseCurrent();
44.   m_PreviouslySelected = newPreviouslySelected;
45.   m_Open = anim;
46.   m_Open.SetBool(m_OpenParameterId, true);
47.   GameObject go = FindFirstEnabledSelectable(anim.gameObject);
48.   SetSelected(go);
49. }
⋮
87. public void CloseCurrent()
88. {
89.   if (m_Open == null)
90.     return;
91.   m_Open.SetBool(m_OpenParameterId, false);
92.   SetSelected(m_PreviouslySelected);
93.   StartCoroutine(DisablePanelDeleyed(m_Open));
94.   m_Open = null;
95. }
```

Skripta sadrži funkciju `OnEnable()` koja se poziva prilikom aktiviranja objekta "MenuManager". Unutar ove funkcije dohvaća se *id* parametra na osnovu kojeg će se pozivati animacija otvaranja i zatvaranja pojedinih izbornika. Također, na liniji 31. provjerava se da li je dodijeljen animator. U slučaju da jeste funkcija se završava, a u protivnom otvara se odgovarajući izbornik. Funkcija `OpenPanel()` iz linija 35.-48. obavlja otvaranje pojedinog izbornika. Na početku funkcije provjerava se je li argument isti kao animator koji se trenutno koristi pri otvaranju, ako jest funkcija završava. U slučaju da nije aktivira se objekt novog animatora, što je određeni izbornik. Nakon toga dolazi do osvježavanja zadnje odabrane opcije

izbornika u linijama 40. i 43. U liniji 42. zatvara se prethodni izbornik. U liniji 44. osvježava se vrijednost varijable `m_Open` i onda se pomoću parametra njenog animatora pokreće animacija i otvaranje novog izbornika. Funkcija `CloseCurrent()` mijenja vrijednost tog parametra, što dovodi do zatvaranja pripadajućeg izbornika.

Skripta: Settings

Isječak koda:

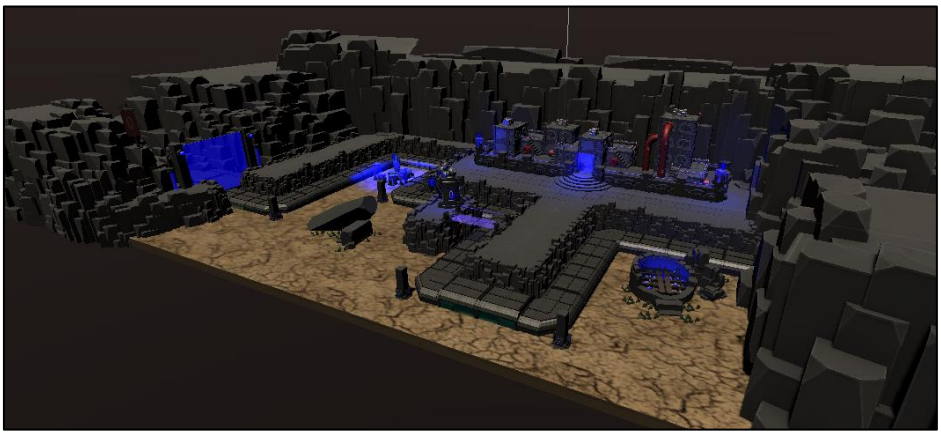
```
⋮
32. void Start()
33. {
34.   difficultySlider.value = difficulty;
35.   qualitySlider.value = quality;
36. }
37.
38. public void DifficultyChange (float value)
39. {
40.   difficulty = (int)value;
41.   difficultyText.text = difficulties[difficulty].title;
42.   startMoneySlider.value = difficulties[difficulty].money / 100;
43.   enemyHealthSlider.value=difficulties[difficulty].enemyHealthFactor*10;
44.   timeBetweenWavesSlider.value =
       difficulties[difficulty].timeBetweenWaves;
45. }
46.
47. public void MoneyChange(float value)
48. {
49.   startMoneyText.text = (value * 100).ToString();
50.   startMoney = (int)value * 100;
51. }
⋮
65. public void QualityChange(float value)
66. {
67.   quality = (int)value;
68.   qualityText.text = qualities[quality];
69.   QualitySettings.SetQualityLevel(quality);
70. }
71.
72. public void FullScreenChange(bool value)
73. {
74.   Screen.fullScreen = value;
75. }
76.
77. public void MusicVolumeChange(float value)
78. {
79.   audioMixer.SetFloat("VolumeMusic", value);
80. }
```

Funkcija `Start()` u skripti „Settings“ postavlja težinu igre i kvalitetu grafike na početne vrijednosti ili vrijednosti koje je igrač odabrao zadnji put. Sve postavke u igri imaju pripadajuće funkcije koje se pozivaju prilikom promjene njihove vrijednosti. Funkcija `DifficultyChange()` poziva se prilikom promjene vrijednosti klizača za težinu igre. Ona prvo sprema vrijednost klizača i osvježava tekst klizača, a nakon toga mijenja vrijednosti ostalih

klizača. Promjenom vrijednosti ostalih klizača okidaju se i njihove funkcije, kao što je funkcija `MoneyChange()` iz 47. linije koda. Nešto drugačija je `QualityChange()` funkcija koja pohranjuje novu vrijednost kvalitete grafike, osvježava tekst klizača, a zatim poziva funkciju `SetQualityLevel()`. Ova funkcija mijenja kvalitetu grafike igre. Na sličan način funkcionira i funkcija `FullScreenChange()`. Što se tiče promjene zvučnih postavki, to je realizirano korištenjem objekta klase `AudioMixer`. Ovom objektu je unutar uređivača pridružen „MusicMixer“ koji sadrži različite grupe zvukova, kao što su grupe za muziku i zvučne efekte. Svaka funkcija mijenja glasnoću odgovarajuće grupe zvukova.

5.3. Razina

Početna razina igre izrađena je kombinacijom velikog broja modela i materijala iz paketa „Sci-Fi Platforms – 3D Models“, „Sanctuary“, „Extreme Sci-Fi LITE“, „Yughues Free Ground Materials“ i „Yughues Free Metal Materials“. Razvoj razine započeo je s izradom puta kojim će se neprijatelji kretati. Dizajn ostataka razine prilagođen je obliku puta. Put prate uzvišenja, odnosno stijene, na koje će biti moguće postavljati tornjeve. Uz skoro cijeli rub razine nalaze se brojne planine koje sprječavaju igrača da vidi kraj razine. Izuzetak je prednji dio, u kojem će se prikazivati kontrole igre. Naposlijetku, dodani su brojni modeli s estetskom ulogom kako bi se uljepšala izgled razine i postigao pravi ugođaj za temu igre. Izgled početne razine prikazan je na slici 11.



Slika 11: Početna razina igre

5.4. Kamera

Značajnu ulogu u ovakvoj vrsti igre ima kamera. Naime, igrač mora imati dobar uvid u cijelu situaciju koja se odvija ispred njega kako bi donio što kvalitetnije strateške odluke. On u

svakom trenutku mora imati mogućnost pogleda na svaki dio razine. Iz tog razloga implementirane su brojni načini upravljanja kamerom. Kameru je moguće približiti kako bi se vidio samo jedan mali dio razine ili udaljiti kako bi se stekao uvid u cijelu razinu i onom što se na njoj dešava. Također, kameru je moguće na više načina pomjerati u različitim smjerovima. Jedan način je pomjeranje kamere korištenjem tipki kojim je svaki iskusniji igrač dobro upoznat: W, A, S i D. Drugi način je pomjeranje kamere korištenjem miša. Na ovaj način igrač još brže može reagirati na različita dešavanja. Kod ove opcije bilo je potrebno implementirati i način zaključavanja položaja kamere kako igrač slučajno ne bi napustio položaj kamere koji mu odgovara. Programski kod koji omogućava sve ovo sadržan je u skripti „CameraController“.

Skripta: CameraController

Isječak koda:

```
⋮
20. void Update()
21. {
⋮
28.   Vector3 position = transform.position;
29.
30.   if (Input.GetKeyDown(KeyCode.C))
31.     locked = !locked;
32.
33.   if (locked)
34.     return;
35.
36.   if (Input.GetKey("w") || Input.mousePosition.y >=
      Screen.height - panBorderThickness)
37.     position.z += panSpeed * Time.deltaTime;
⋮
47.   float scroll = Input.GetAxis("Mouse ScrollWheel");
48.   position.y -= scroll * 1000 * scrollSpeed * Time.deltaTime;
49.   position.x = Mathf.Clamp(position.x, minX, maxX);
50.   position.y = Mathf.Clamp(position.y, minY, maxY);
51.   position.z = Mathf.Clamp(position.z, minZ, maxZ);
52.   transform.position = position;
53. }
```

Skripta sadrži samo jednu funkciju i to je funkcija `Update()`. U funkciji se prvo provjerava da li je igra završila. U slučaju da jeste kamera se deaktivira, što onemogućuje daljini prikaz razine. U varijabli `position` pohranjuje se trenutna lokacija kamere kako bi se u ostatku koda mogla mijenjati. U ostatku koda nalaze se selekcije koje provjeravaju da li je pritisnuta jedna od tipki kojom se upravlja kamerom. U selekciji koja se nalazi u 36. liniji koda, provjerava se je li pritisnuta tipka „W“ ili nalazi li se položaj pokazivača u rubnom gornjem dijelu zaslona. Ovaj rubni dio zaslona dobiva se razlikom između visine zaslona i debljine okvira koju određujemo po volji. Sličan postupak se primjenjuje i za ostale smjerove pomjeranja kamere. U 47. i 48. liniji koda omogućava se približavanje i udaljavanje kamere. Varijabla `scroll` sadrži smjer

pomjeranja kotačića miša. Promjena kamere dobije se umnoškom te vrijednosti s brzinom klizanja kotačića i delta vremenom. Delta vrijeme sadrži proteklo vrijeme između kadrova i osigurava da vrijednost bude jednaka neovisno o broju kadrova, a samim tim i od postavki igre [36]. Nakon toga pomoću funkcije `Clamp()` osigurava se da kamera ne izađe iz područja koji smo za nju odredili, tj. da bude između definirane minimalne i maksimalne vrijednosti.

5.5. Neprijatelji

Igra sadrži četiri vrste neprijatelja i to su paukovi, dronovi, tenkovi i letjelice. Svaki od neprijatelja razlikuje se po svojim vanjskim karakteristikama, brzini kretanja, jačini i količini novca koju donosi. Neprijatelji su prikazani na slici 12. Za igru je bilo potrebno implementirati kretanje neprijatelja po definiranom putu. Implementirani su i brojni drugi detalji, kao što je animaciju prilikom kretanja i umiranja, prikaz trenutne životne energije, ranjavanje i napadanje. Programski kod koji realizira ove mogućnosti nalazi se u skriptama: „Enemy“ i „EnemyMovement“.

Slika 12. Vrste neprijatelja u igri

Skripta: Enemy

Isječak koda:

```
⋮
30. public void Attacked(float amount)
31. {
32.   health -= amount;
33.   healthBar.fillAmount = health / startHealth;
34.
35.   if (health <= 0 && !dead)
36.     Die();
37. }
⋮
51. void Die()
52. {
53.   dead = true;
54.   WaveSpawner.enemiesAlive--;
55.   PlayerStats.Money += value;
56.   PlayerStats.KillCount++;
57.   GameObject effectInstance =
```

```
      Instantiate(DeathEffect,transform.position,Quaternion.identity);
58.   Destroy(effectInstance, 4f);
59.   audioManager.PlayDelayed("Explosion", 0.2f);
60.   audioManager.Play("Enemy death")
61.   if (hasAnimation)
62.     anim.SetBool("isDying", true);
63.   else
64.     Destroy(gameObject);
65. }
```

Funkcija `Attacked()` poziva se svaki put kad je jedan od neprijatelja napadnut. Varijabla `health` sadrži životnu energiju neprijatelja koju funkcija smanjuje na 32. liniji koda. Funkcija potom osvježava prikaz životne energije koristeći `healthBar` objekt kojem je unutar uređivača pridružena grafika životne energije iznad neprijatelja. U slučaju da životna energija neprijatelja padne na nulu i da neprijatelj nije već uništen, poziva se funkcija `Die()`. Ova funkcija smanjuje broj neprijatelja vala, povećava novac igrača i broj uništenih neprijatelja. Nakon toga generira se efekt uništenja na poziciji i rotaciji neprijatelja. Za dobivanje i prilagodbu efekta položaju osa objekta koristi se `Quaternion.identity`. Zatim se u 64. liniji označava da je neprijatelj uništen. Ako neprijatelj ima pridruženu animaciju, ona se pokreće promjenom vrijednosti animatorovog parametra „isDying“. U slučaju da nema animaciju, neprijatelj se odmah uništava.

Skripta: EnemyMovement

Isječak koda:

```
⋮
38. void Start()
39. {
40.   enemy = GetComponent<Enemy>();
41.   enemy.dead = false;
42.   waypoints = new Transform[way.transform.childCount];
43.   for (int i = 0; i < waypoints.Length; i++)
44.   {
45.     waypoints[i] = way.transform.GetChild(i);
46.   }
47.   target = waypoints[0];
48. }
49.
50. void Update()
51. {
52.   if (!enemy.dead)
53.   {
54.     Vector3 direction = target.position - transform.position;
55.     transform.Translate(direction.normalized * enemy.speed *
        Time.deltaTime, Space.World);
56.
57.     transform.forward = Vector3.Normalize(direction);
58.
59.     If(Vector3.Distance(transform.position,target.position)<= (0.4f))
60.       GetNextWaypoint();
61.   }
62.   enemy.speed = enemy.startSpeed;
```

```
63. }
```

Na 42. – 46. linijama koda dohvaćaju se sve putne točke (eng. *Waypoints*) puta kojim se neprijatelj treba kretati. Prvo se definira polje tipa `Transform` duljine broja točaka koju put posjeduje. Nakon toga polje se kroz petlju „napuni“ s pozicijom svake od točaka puta. U 47. liniji koda postavlja se meta kretanja neprijatelja na prvu točku puta. Unutar funkcije `Update()` realizira se kretanje neprijatelja po točkama puta. U slučaju da neprijatelj nije mrtav, dohvaća se smjer u kojem se treba kretati neprijatelj. Vektor smjera je dobiven razlikom pozicije mete i trenutne pozicije neprijatelja. Nakon toga slijedi pomjeranje neprijatelja pomoću funkcije `Translate()`. Ova funkcija pomjera neprijatelja na osnovu umnoška normaliziranog vektora smjera, brzine neprijatelja i delta vremena. Vektor smjera mora biti normaliziran kako ne bi utjecao na vrijednost umnoška. Posljednji argument funkcije `Space.World` označava da se neprijatelj kreće u odnosu na cijelo okruženje. Selekcijom iz 59. linije koda provjerava se je li neprijatelj blizu putnoj točki i u slučaju da jeste usmjerava ga na sljedeću putnu točku.

5.6. Valovi neprijatelja

Kako bi se realizirala potpuna funkcionalnost neprijatelja, bilo je potrebno implementirati način stvaranja valova neprijatelja. Valovi bi se trebali razlikovati u broju i vrsti neprijatelja. Programski kod koji omogućava ovu logiku nalazi se u skripti „WaveSpawner“.

Skripta: WaveSpawner

Isječak koda:

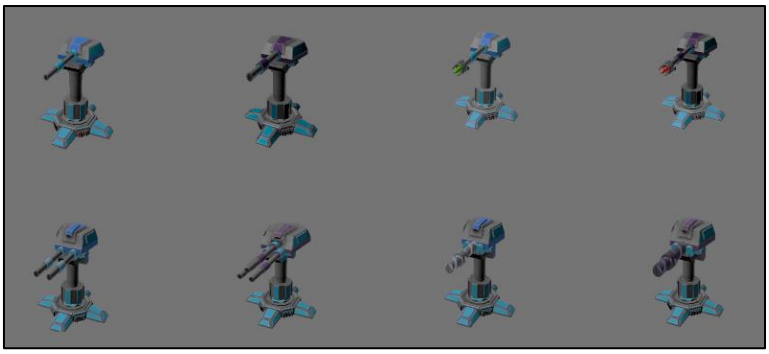
```
⋮
18. void Update()
19. {
20.   if (enemiesAlive > 0)
21.     return;
22.   if (countdown <= 0f)
23.   {
24.     StartCoroutine(SpawnWave());
25.     countdown = timeBetweenWaves;
26.     return;
27.   }
28.   countdown -= Time.deltaTime;
29.   countdown = Mathf.Clamp(countdown, 0f, Mathf.Infinity);
30.   waveCountdownText.text = string.Format("{0:00.00}", countdown);
31. }
32.
33. IEnumerator SpawnWave ()
34. {
35.   StartCoroutine(notification.Display("New wave is coming!", 5f));
36.   audioManager.Play("Wave incoming");
37.   PlayerStats.Rounds++;
38.   Wave wave = waves[waveIndex];
39.
```

```
40.   for (int i = 0; i < wave.count; i++)
41.   {
42.     SpawnEnemy(wave.enemies[i]);
43.     yield return new WaitForSeconds(1f / wave.rate);
44.   }
45.
46.   waveIndex++;
47.
48.   if (waveIndex == waves.Length)
49.   {
50.     gameManager.WinLevel();
51.     this.enabled = false;
52.   }
53. }
```

Na početku funkcije `Update()` provjerava se je li postoji još živih neprijatelja i u slučaju da ima prekida se daljnje njeno izvođenje. Zatim se provjerava je li isteklo vrijeme prije početka novog vala i u slučaju da jeste poziva se korutina `SpawnWave()` koja stvara novi val neprijatelja. Vrijeme se smanjuje korištenjem prethodno opisanog delta vremena. Pomoću funkcije `Clamp()` osigurava se da ono nikad neće postati negativno, dok funkcija `Format()` mijenja format ispisa vremena na zaslon. Kao što je već prethodno navedeno, funkcija `SpawnWave()` stvara novi val neprijatelja. Kako bi ispisala obavijest o nadolazećem napadu, funkcija koristi korutinu `Display()` sa sadržajem i duljinom obavijesti kao parametrima. Na 40. liniji koda petljom se instanciraju pojedini neprijatelji prethodno odabranog vala. Vremenski razmak između stvaranja neprijatelja realiziran je korištenjem funkcije `WaitForSeconds()`. Ova funkcija pauzira izvođenje na određeno vrijeme, u ovom slučaju to je vremenski razmak vala. Ako je instanciran posljednji val, poziva se funkcija `WinLevel()` koja aktivira prikaz pobjedničkog zaslona. Nakon toga skripta se deaktivira.

5.7. Tornjevi

Igra sadrži četiri vrste tornjeva od kojih se svaki može dodatno nadograditi. Sve tornjeve i nadogradnje moguće je vidjeti na slici 13. Tornjevi posjeduju različite sposobnosti, zvučne efekte i vrste projektila. Toranj s laserom kontinuirano usporava i oduzima energiju neprijatelju. Toranj s raketom ima projektile koji prouzrokuju eksploziju i oduzimaju energiju svim neprijateljima u blizini. Postoje tornjevi koji imaju više cijevi. Ove mogućnosti realizirane su sa skriptom „Turret“.



Slika 13: Vrste tornjeva u igri

Skripta: Turret

Isječak koda:

```
⋮
96.  void LockOnTarget()
97.  {
98.    Vector3 direction = target.position - transform.position;
99.    Quaternion lookRotation = Quaternion.LookRotation(direction);
100.   Vector3 rotation = Quaternion.Lerp(rotatingPart.rotation, lookRotation,
       Time.deltaTime * rotateSpeed).eulerAngles;
101.   rotatingPart.rotation = Quaternion.Euler(rotation.x, rotation.y, 0f);
102. }
103.
104. void UpdateTarget()
105. {
106.   GameObject[] enemies = GameObject.FindGameObjectsWithTag(enemyTag);
107.   float shortestDistance = Mathf.Infinity;
108.   GameObject nearestEnemy = null;
109.
110.   foreach (GameObject enemy in enemies)
111.   {
112.     float distanceToEnemy = Vector3.Distance(transform.position,
       enemy.transform.position);
113.
114.     if (distanceToEnemy < shortestDistance)
115.     {
116.       shortestDistance = distanceToEnemy;
117.       nearestEnemy = enemy;
118.     }
119.   }
120.
121.   if (nearestEnemy != null && shortestDistance <= range)
122.   {
123.     target = nearestEnemy.transform;
124.     enemy = nearestEnemy.GetComponent<Enemy>();
125.   }
126.   else
127.     target = null;
128. }
129.
130. void Shoot()
131. {
132.   if (useRocket)
133.     audioManager.Play("Rocket shot");
134.   else
```

```
135.     audioManager.Play("Regular shot");
136.
137.   foreach (var firePoint in firePoints)
138.   {
139.     if (firePoint != null)
140.     {
141.       GameObject bulletGO = Instantiate(bulletPrefab, firePoint.position,
           firePoint.rotation);
142.       Bullet bullet = bulletGO.GetComponent<Bullet>();
143.       if (bullet != null)
144.         bullet.Seek(target);
145.
146.     }
147.   }
148. }
```

Funkcija `LockOnTarget()` omogućava tornjevima da naciljaju označenog neprijatelja, tj. neprijatelja kojeg su izabrali za metu. U funkciji se prvo dohvati odgovarajući smjer u kojem se toranj treba usmjeriti, a zatim se izračuna potrebna rotacija tornja na temelju tog smjera u 99. liniji koda. Pomoću funkcije `Lerp()` se interpolira između trenutne rotacije tornja i prethodno dobivene rotacije. Dobivena vrijednost se u 101. liniji koristi za rotiranje tornja prema neprijatelju. Funkcija `UpdateTarget()` omogućava pronalazak i označavanje najbližeg neprijatelja u dometu tornja. Prvo se u polje `enemies` pohrane svi neprijatelji koji se trenutno nalaze u razini, a onda se petljom iz 110. linije koda pronađe najbliži neprijatelj. Blizina neprijatelj izračunava se pomoću funkcije `Distance()`, kojoj se prosljeđuje pozicija tornja i pojedinog neprijatelja. Nakon što je pronađen najbliži neprijatelj, provjerava se je li on u dometu tornja. U slučaju da je neprijatelj u dometu, on postaje nova meta tornja. Kada je neprijatelj pronađen i naciljan toranj započinje pucanje, koje je omogućeno funkcijom `Shoot()`. Ova funkcija instancira projektil na poziciji i rotaciji objekta `firePoint` koji se nalazi na izlazu cijevi tornja. Nakon instanciranja projektila koristi se funkcija `Seek()` koja pomjera metak od tornja prema označenom neprijatelju.

Veoma važna bila je mogućnost gradnje tornjeva, odnosno postavljanja tornjeva na za to predviđene pozicije. Također, bilo je potrebno omogućiti nadogradnju i prodaju tornjeva. Ove mogućnosti implementirane su skriptama „Node“ i „BuildManager“. Naziv „node“ označava poziciju na koju je moguće graditi pojedini toranj.

Skripta: Node

Isječak koda:

```
⋮
47. void BuildTurret(TurretBlueprint blueprint)
48. {
49.   if (PlayerStats.Money < blueprint.cost)
50.   {
51.     StartCoroutine(notification.Display("Not enough bitcoin!", 2f));
52.     return;
```

```
53.    }
54.    PlayerStats.Money -= blueprint.cost;
55.
56.    GameObject _turret = Instantiate(blueprint.prefab,
        GetBuildPosition(),Quaternion.identity);
57.    turret = _turret;
58.    turrentBlueprint = blueprint;
59.
60.    GameObject effect = Instantiate(buildManager.buildEffect,
        GetBuildPosition(),Quaternion.identity);
61.    Destroy(effect, 3f);
62. }
63.
64. public void UpgradeTurret()
65. {
66.    if (PlayerStats.Money < turrentBlueprint.upgradeCost)
67.    {
68.      StartCoroutine(notification.Display("Not enough bitcoin!", 2f));
69.      return;
70.    }
71.    audioManager.Play("Build sound");
72.    PlayerStats.Money -= turrentBlueprint.upgradeCost;
73.    Destroy(turret);
74.
75.    GameObject _turret = Instantiate(turrentBlueprint.upgradePrefab,
        GetBuildPosition(), Quaternion.identity);
76.    turret = _turret;
77.
78.    GameObject effect = Instantiate(buildManager.buildEffect,
        GetBuildPosition(), Quaternion.identity);
79.    Destroy(effect, 3f);
80.    isUpgraded = true;
81. }
82.
83. public void SellTurret()
84. {
85.    PlayerStats.Money += turrentBlueprint.GetSellPrice();
86.    audioManager.Play("Sell sound");
87.
88.    GameObject effect = Instantiate(buildManager.sellEffect,
        GetBuildPosition(), Quaternion.identity);
89.    Destroy(effect, 3f);
90.
91.    Destroy(turret);
92.    turrentBlueprint = null;
93.    isUpgraded = false;
94. }
```

Funkcija `BuildTurret()` poziva se prilikom gradnje tornja. Ona prima kao argument toranj koji je igrač odabrao i na početku provjerava ima li igrač dovoljno novca za gradnju odabranog tornja. U 56. liniji koda instancira se odabrani toranj na poziciji objekta odnosno *nodea*. Njegova pozicija dobiva se pozivanjem funkcije `GetBuildPosition()`. Postavljeni toranj pohranjen je u varijablu `turret`. U 60. liniji instancira se efekt gradnje tornja, koji se potom nakon četiri sekunde uništava. Funkcija `UpgradeTurret()` implementirana je na sličan način. Razlika je u tome što se postojeći toranj prvo uništi, a nakon toga se instancira

nadograđeni toranj. U slučaju da igrač odluči prodati toranj, poziva se funkcija `SellTurret()`. Ova funkcija poveća novac igrača ovisno o prodajnoj cijeni tornja, a nakon toga slijedi instanciranje efekta prodaje tornja te uništavanje tornja.

Skripta: BuildManager

Isječak koda:

```
⋮
48. public void SelectNode(Node node)
49. {
50.   if (node == selectedNode)
51.   {
52.     DeselectNode();
53.     return;
54.   }
55.
56.   selectedNode = node;
57.   turretToBuild = null;
58.   turretUI.SetTarget(node);
59. }
60.
61. public void DeselectNode()
62. {
63.   selectedNode = null;
64.   turretUI.Hide();
65. }
66.
67. public void SelectTurret(TurretBlueprint turret)
68. {
69.   turretToBuild = turret;
70.   DeselectNode();
71. }
72.
73. public void CheckTurretButton(TurretBlueprint turret)
74. {
75.   if (PlayerStats.Money < turret.cost)
76.     turret.button.interactable = false;
77.   else
78.     turret.button.interactable = true;
79. }
```

Kada igrač klikne na određeni *node* poziva se funkcija `SelectNode()`. Ova funkcija sprema odabranu poziciju za postavljanje tornja u varijablu `selectedNode`. Ako je igrač kliknuo na poziciju koja je već odabrana, onda se poziva funkcija `DeselectNode()`. Ova funkcija briše sadržaj varijable `selectedNode`. Pomoću funkcije `SelectTurret()` odabire se toranj kojeg igrač želi postaviti. Funkcijom `CheckTurretButton()` provjerava se ima li igrač dovoljno novca za gradnju odabranog tornja. U slučaju da igrač nema dovoljno novca, tipka tornja se deaktivira na 76. liniji koda.

6. Zaključak

Cilj ovog završnog rada bio je razviti jednostavnu i zanimljivu igru obrane tornjevima. Iz ovog poduhvata izvučeni su brojni zaključci. Prilikom razvoja igre ovog žanra korišteni su brojni alati od kojih je najvažniju ulogu imao programski alat Unity. Ovaj alat pokazao se veoma moćnim, ali i iznenađujuće jednostavnim za izradu igara. Većina funkcionalnosti pojednostavljena je do te razine da svaki početnik može u veoma kratkom periodu usvojiti sve osnovne koncepte. Ono što početniku može predstavljati određene probleme prilikom razvoja igre je pozadinska logika objekata igre. Naime, različita ponašanja objekata ostvaruju se pomoću skripti napisanih u jednom od već navedenih programskih jezika. Ovo zahtijeva određeno znanje o programiranju u jednom od ovih jezika, ali čak i onda potrebno je naučiti raditi s Unityjevim klasama i specifičnom mehanikom koje igre imaju. Također, razvijanje igara često zahtijeva i druge popratne vještine i znanja. Primjer je poznavanje osnova vektorske algebre i analitičke geometrije. Također, posjedovanje vještine rada u alatima za grafički dizajn pokazalo se veoma korisnim prilikom dizajna modela koji su korišteni u igri. Sve ovo pokazuje da je razvoj i najjednostavnijih igara složen proces koji zahtijeva dosta vremena i širok portfolio različitih znanja i vještina.

Popis literature

[1] M. Overmars, "A Brief History of Computer Games", str. 1-10, (30.01.2012.) [Na internetu]. Dostupno:https://www.stichtingspel.org/sites/default/files/history_of_games.pdf [pristupano 11.08.2019.].

[2] D. Arsenault, "Video Game Genre, Evolution and Innovation" *Eludamos. Journal for Computer Game Culture*, sve. 3, izd. 2, str. 151-153, 2009. [Na internetu]. Dostupno: ResearchGate, https://www.researchgate.net [pristupano 11.08.2019.].

[3] P. Avery, J. Togelius, R. P. Leeuwen, i E. Alistar, "Computational Intelligence and Tower Defence Games" *Zbornik radova konferencije: IEEE Congress of Evolutionary Computation 2011. (CEC)*, 2011. [Na internetu]. Dostupno: ResearchGate, https://www.researchgate.net/ [pristupano 11.08.2019.].

[4] PC Dreams Group, "History of Tower Defense Games", (08.05.2017.) [Na internetu]. Dostupno: https://pcdreams.com.sg/history-of-tower-defense-games/ [pristupano 12.08.2019.].

[5] D. Soos, "History of Tower Defense Games", (20.08.2012.) [Na internetu]. Dostupno: https://ezinearticles.com/?History-of-Tower-Defense-Games&id=7241055 [pristupano 12.08.2019.].

[6] "Tower Defense Games", (bez dat.) *u MediaWiki*. Dostupno: https://mediawiki.middlebury.edu/FMMC0282/Tower_Defense_Games#History [pristupano 12.08.2019.].

[7] J. Dalrymple, i D. Mark, "Understanding Tower Defense games", (30.03.2010.) [Na internetu]. Dostupno: https://www.loopinsight.com/2010/03/30/understanding-tower-defense-games [pristupano 12.08.2019.].

[8] E. Watson, "Defense Grid, the 'beer game' that changed a genre", (24.04.2018.) [Na internetu]. Dostupno: https://www.pcgamer.com/defense-grid-the-beer-game-that-changed-a-genre/ [pristupano 14.08.2019.].

[9] J. Ocampo, "Defense Grid: The Awakening Review", (11.05.2012.) [Na internetu]. Dostupno: https://www.ign.com/articles/2008/12/22/defense-grid-the-awakening-review-2 [pristupano 14.08.2019.].

[10] D. Cuthbert, "PixelJunk Monsters Set to Launch Next Week", (16.01.2008.) [Na internetu]. Dostupno: https://blog.us.playstation.com/2008/01/16/pixeljunk-monsters-set-to-launch-next-week/ [pristupano 15.08.2019.].

[11] R. Clements, "PixelJunk Monsters Review", (13.05.2012.) [Na internetu]. Dostupno: https://www.ign.com/articles/2008/01/29/pixeljunk-monsters-review [pristupano 15.08.2019.].

[12] D. Takahashi, "How George Fan created the wacky Plants vs. Zombies a decade ago", (10.05.2019.) [Na internetu]. Dostupno: https://venturebeat.com/2019/05/10/how-george-fan-created-the-wacky-plants-vs-zombies-a-decade-ago/ [pristupano 17.08.2019.].

[13] D. Hatfield, "Plants vs. Zombies Review", (11.05.2012.) [Na internetu]. Dostupno: https://www.ign.com/articles/2009/05/05/plants-vs-zombies-review-4 [pristupano 17.08.2019.].

[14] K. Stuart, "Plants vs Zombies", (18.02.2010.) [Na internetu]. Dostupno: https://www.theguardian.com/technology/gamesblog/2010/feb/18/games-apple [pristupano 17.08.2019.].

[15] J. Brodkin, "How Unity3D Became a Game-Development Beast", (03.06.2013.) [Na internetu]. Dostupno: https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast [pristupano 18.08.2019.].

[16] P. Cohen, "Unity 2.0 game engine now available", (10.10.2007.) [Na internetu]. Dostupno: https://www.macworld.com/article/1060484/unity.html [pristupano 19.08.2019.].

[17] D. Girard, "Unity 2.0 game engine now available", (28.09.2010.) [Na internetu]. Dostupno: https://arstechnica.com/information-technology/2010/09/unity-3-brings-very-expensive-dev-tools-at-a-very-low-price/ [pristupano 19.08.2019.].

[18] D. Takahashi, "Game developers, start your Unity 3D engines (interview)", (02.11.2012.) [Na internetu]. Dostupno: https://venturebeat.com/2012/11/02/game-developers-start-your-unity-3d-engines-interview/ [pristupano 20.08.2019.].

[19] D. Tach, "Unity 4.0 available for download today with DX 11 support and Linux preview", (14.11.2012.) [Na internetu]. Dostupno: https://www.polygon.com/2012/11/14/3645122/unity-4-0-available-download [pristupano 21.08.2019.].

[20] C. Clarke, "Unity wins 'Best Engine' at 2014 Develop Industry Excellence Awards", (07.10.2014.) [Na internetu]. Dostupno: https://www.gamasutra.com/view/pressreleases/220751/UNITY_WINS_ldquoBEST_ENGINErdquo_AT_2014_DEVELOP_INDUSTRYEXCELLENCE_AWARDS.php [pristupano 21.08.2019.].

[21] G. Kumparak, "Unity 5 Announced With Better Lighting, Better Audio, And 'Early' Support For Plugin-Free Browser Games", (18.03.2014.) [Na internetu]. Dostupno: https://techcrunch.com/2014/03/18/unity-5-announced-with-early-support-for-plugin-free-browser-games/?guccounter=1 [pristupano 21.08.2019.].

[22] Nanalyze, "Unity Technologies – The World's Leading Game Engine", (18.10.2017.) [Na internetu]. Dostupno: https://www.nanalyze.com/2017/10/unity-technologies-leading-game-engine [pristupano 22.08.2019.].

[23] J. Batchelor, "Unity 2018 detailed in GDC keynote", (20.03.2018.) [Na internetu]. Dostupno: https://www.gamesindustry.biz/articles/2018-03-20-unity-2018-detailed-in-gdc-keynote [pristupano 22.08.2019.].

[24] Unity Technologies, "Creating and Using Scripts" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html [pristupano 28.08.2019.].

[25] Unity Technologies, "Prefabs" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/Manual//Prefabs.html [pristupano 28.08.2019.].

[26] Unity Technologies, "The Inspector window" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/Manual/UsingTheInspector.html [pristupano 29.08.2019.].

[27] Unity Technologies, "Animation" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/Manual/AnimationSection.html [pristupano 29.08.2019.].

[28] L. Paczkowski, "Replacing MonoDevelop-Unity with Visual Studio Community starting in Unity 2018.1" *Unity Blog*, (05.01.2018.) [Na internetu]. Dostupno: https://blogs.unity3d.com/2018/01/05/discontinuing-support-for-monodevelop-unity-starting-in-unity-2018-1 [pristupano 01.09.2019.].

[29] Unity Technologies, "MonoBehaviour" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html [pristupano 02.09.2019.].

[30] Unity Technologies, "MonoBehaviour.Awake()" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html [pristupano 02.09.2019.].

[31] Unity Technologies, "Coroutines" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/Manual/Coroutines.html [pristupano 02.09.2019.].

[32] "Adobe Photoshop", (bez dat.) u Techopedia. Dostupno: https://www.techopedia.com/definition/32364/adobe-photoshop [pristupano 3.09.2019.].

[33] PC Magazine, "Adobe Illustrator", (bez dat.) [Na internetu]. Dostupno: https://www.pcmag.com/encyclopedia/term/37547/adobe-illustrator [pristupano 3.09.2019.].

[34] A. Thirslund, (15.01.2017.) "How to make a Tower Defense Game", *Youtube* [Video datoteke]. Dostupno:

https://www.youtube.com/playlist?list=PLPV2KyIb3jR4u5jX8za5iU1cqnQPmbzG0 [pristupano 20.08.2019.].

[35] Unity Technologies, "Platform dependent compilation" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/Manual/PlatformDependentCompilation.html [pristupano 08.09.2019.].

[36] Unity Technologies, "Time.deltaTime" *Unity User Manual*, (bez dat.) [Na internetu]. Dostupno: https://docs.unity3d.com/ScriptReference/Time-deltaTime.html [pristupano 09.09.2019.].

Popis slika

Slika 1: Općeniti prikaz igre „Obrane tornjevima“ 2
Slika 2: Igra "Defense Grid: The Awakening" 4
Slika 3: Igra "PixelJunk Monsters" 5
Slika 4: Igra "Plants vs. Zombies" 5
Slika 5: Sučelje Unity uređivača 8
Slika 6: Sučelje Microsoft Visual Studija 8
Slika 7: Dijagram slučajeva korištenja 16
Slika 8. Početni izbornik igre 18
Slika 9. Izbornik za postavke igre 19
Slika 10. Sučelje igre prilikom igranja 19
Slika 11: Početna razina igre 23
Slika 12. Vrste neprijatelja u igri 25
Slika 13: Vrste tornjeva u igri 29