SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

**Unity - Tower defense**

*Ante Mihanović*

*Antonio Miličić*

*Bruno Domančić*

*Split, 2018/19*

Sadržaj

[1 Uvod 1](#_Toc533967779)

[2 Unity 2](#_Toc533967780)

[2.1 Više o Unity-u 2](#_Toc533967781)

[2.2 Instalacija 2](#_Toc533967782)

[2.3 Izrada novog projekta 6](#_Toc533967783)

[3 Izrada osnovnog nivoa 8](#_Toc533967784)

[3.1 Upoznavanje s radnom okolinom 8](#_Toc533967785)

[3.2 Izrada nivoa 11](#_Toc533967786)

[4 Protivnici 13](#_Toc533967787)

[4.1 Izvor i krajnje odredište za objekte protivnika 13](#_Toc533967788)

[4.2 Izrada puta 14](#_Toc533967789)

[4.3 Izrada protivnika 16](#_Toc533967790)

[4.4 Skripta za generiranje protivnika 18](#_Toc533967791)

[4.5 Skripta za obradu dolaska na odredište 20](#_Toc533967792)

[4.6 Obrada uništavanja protivnika 21](#_Toc533967793)

[5 Obrambene utvrde 22](#_Toc533967794)

[5.1 Atributi Utvrda 23](#_Toc533967795)

[5.2 Ciljanje 25](#_Toc533967796)

[5.2.1 Pucanje 27](#_Toc533967797)

[5.3 Bullet.cs 30](#_Toc533967798)

[5.3.1 Update 30](#_Toc533967799)

[5.3.2 HitTarget 31](#_Toc533967800)

[5.4 Postavljanje utvrda 33](#_Toc533967801)

[5.4.1 Interakcija s korisnikom 33](#_Toc533967802)

[5.4.2 Gradnja utvrda 34](#_Toc533967803)

[5.4.3 Kupovina utvrda 35](#_Toc533967804)

[6 Izbornici 37](#_Toc533967805)

[6.1 Glavni izbornik 37](#_Toc533967806)

[6.2 Izbornik razine 38](#_Toc533967807)

[6.2.1 Otključavanje i pobijeđivanje razina 39](#_Toc533967808)

[6.3 „Game Over“ izbornik 41](#_Toc533967809)

[6.4 Izbornik pauze 44](#_Toc533967810)

[7 Izrada korisničkog sučelja 46](#_Toc533967811)

[7.1 Kamera 46](#_Toc533967812)

[7.2 Trgovina utvrda 48](#_Toc533967813)

[7.3 Sučelje utvrde 50](#_Toc533967814)

[7.4 Interaktivna podloga 53](#_Toc533967815)

[7.5 Statusni tekst 54](#_Toc533967816)

[7.5.1 Vrijeme 55](#_Toc533967817)

[7.5.2 Novac 56](#_Toc533967818)

[7.5.3 Cijena utvrda 56](#_Toc533967819)

[7.5.4 Životi igrača 57](#_Toc533967820)

[7.5.5 Životi objekta protivnika 57](#_Toc533967821)

[7.6 Prijelaz između scena 58](#_Toc533967822)

[8 Vodič za igru 60](#_Toc533967823)

[9 Literatura 63](#_Toc533967824)

[10 Popis kratica i reference 64](#_Toc533967825)

# Uvod

Zadatak ovog projekta je napraviti video igru tipa „tower defense“, koristeći Unity programsko okružje za izgradnju igara.

U sklopu toga, dan je opis instalacije Unity-a, upoznavanje s njegovim mogućnostima, izrada same igre te kratki vodič kroz igru.

Kontekst izrade video igre može se proširiti naknadno na bilo koju drugu video igru u istom okružju, koristeći C# jezik za programiranje skripti za pojedine funkcionalnosti.   
Stoga se ova igra, te njegova dokumentacija mogu se koristiti za učenje osnova Unity-a te programiranja u istom, koristeći C# jezik.

# Unity

U ovom poglavlju pozabavit ćemo se Unity-jem, njegovim mogućnostima te instalacijom

## Više o Unity-u

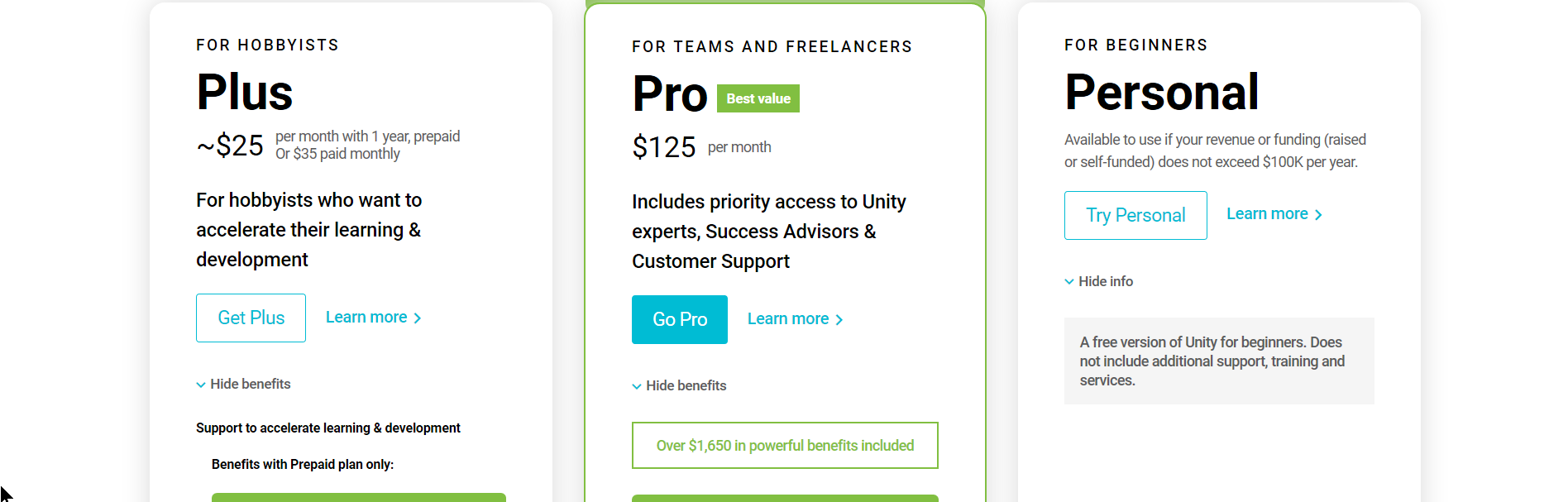
Unity je „GameEngine“ koji se generalno koristi za izradu video igara, bilo 2D ili 3D pristupom. Kodiranje je moguće obaviti koristeći C# jezik ili JavaScript, iako u novim verzijama je JavaScript izbačen iz uporabe.

Osim izrade video igara, Unity se može koristiti za simulacije, poput testova sudara… No osnovna ideja Unity-a je i dalje korištenje komponentno baziranog pristupa, koje omogućava jednostavniju i bržu izradu novih stvari koristeći postojeće objekte. Tako Unity dolazi s više prethodno napravljenih animacija, čestica, te predložaka za izradu projekata.

## Instalacija

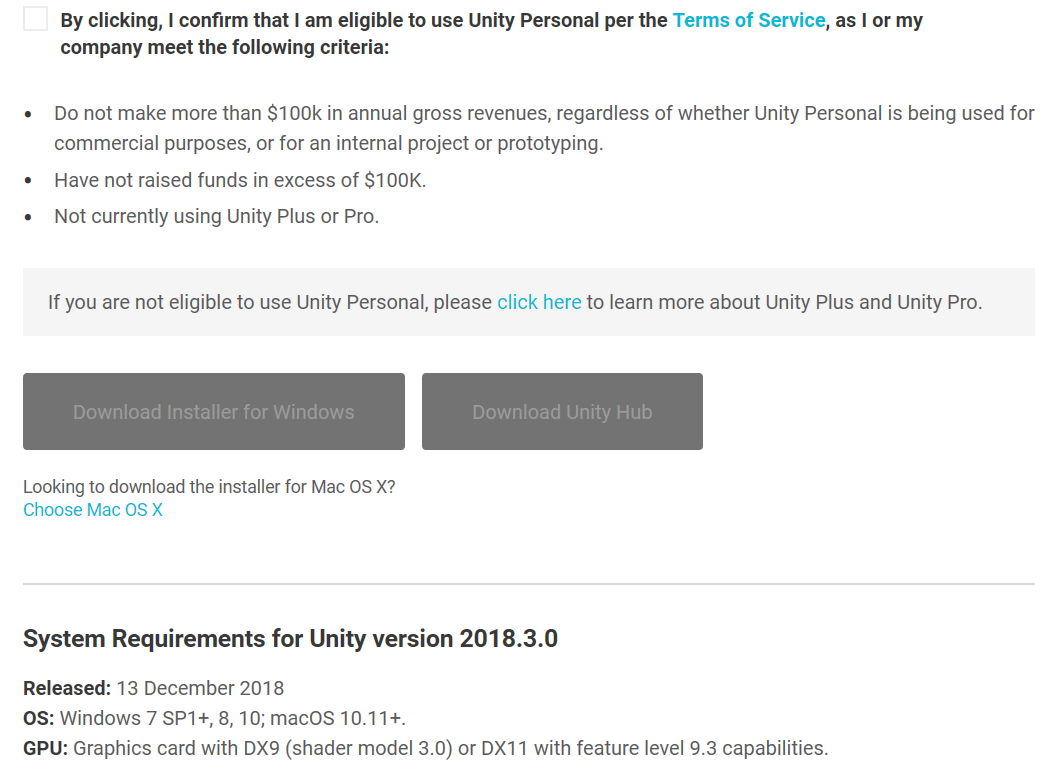
Krenimo od instalacije Unity-a. Na službenoj Unity stranici[[1]](#endnote-1) pronalazimo dugme nazvano: „get started“, koje nas odvodi do sljedeće stranice[[2]](#endnote-2).

Otvara se stranica sa sljedećim ponudama, Slika 2‑1. Ovaj projekt je izrađen koristeći „Personal“ verziju Unity proizvoda. Ta verzija je dostupna svima, pod određenim uvjetima koji su dostupni prilikom odabira samog proizvoda.



Slika 2‑1 Unity ponuda proizvoda

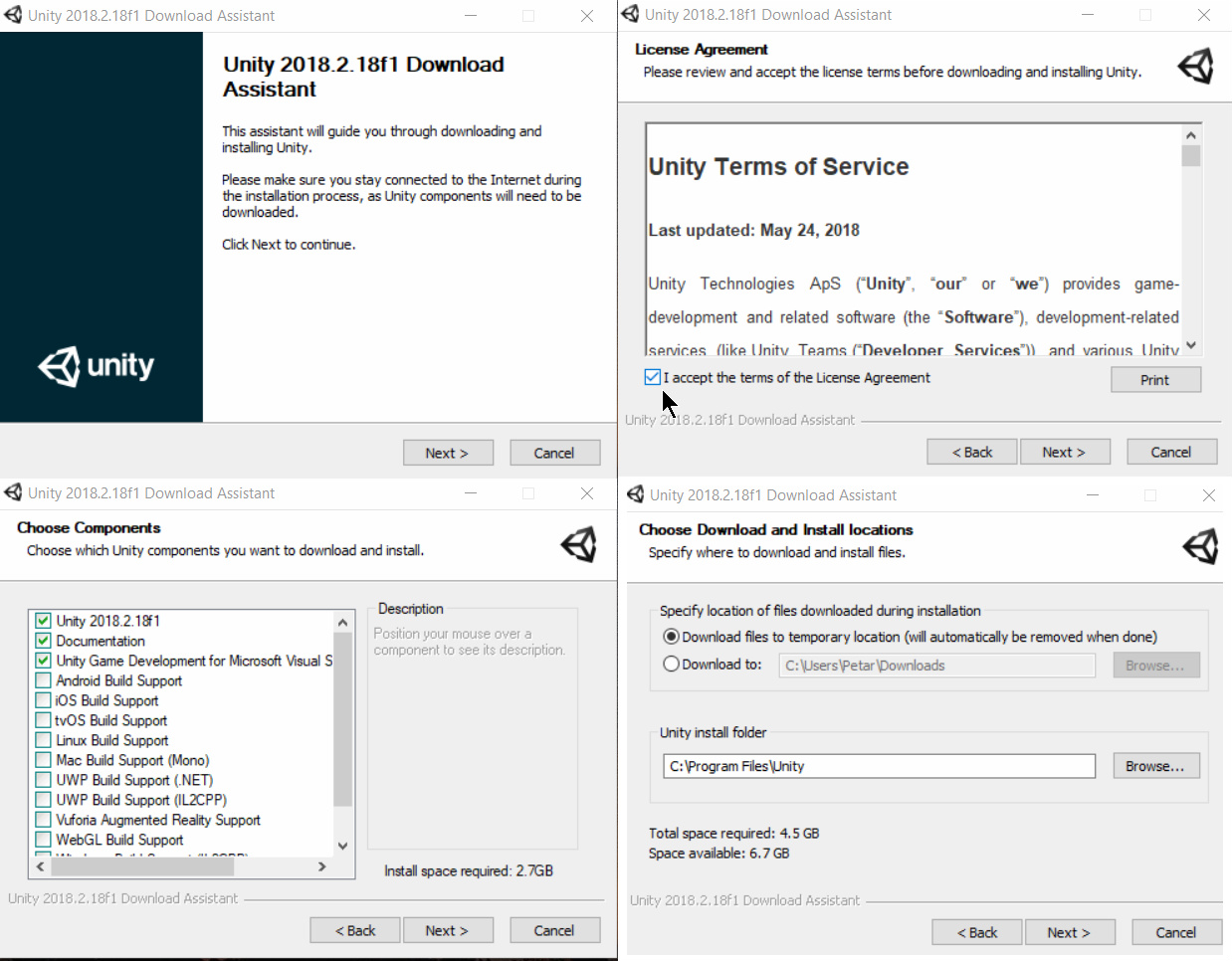
Sljedeći korak je klik na dugme „Try Personal“, koje nas vodi na novu stranicu[[3]](#endnote-3). Na njoj prihvaćamo uvjete Unity-a, Slika 2‑2, te klikom na dugme: „Downaload Installer for Windows“, preusmjereni smo na novu stranicu te se pokreće prozor za skidanje Unity-a.   
Odabiremo „spremi datoteku“ opciju. Ispod ove ponude nalazi se i ponuda za Mac operacijski sustav.



Slika 2‑2 Unity prihvačanje uvjeta

Nakon skidanja instalera, pokrenemo ga, te se otvara prozor s ponuđenim opcijama na kojim odabiremo „Next“ opciju.   
Na sljedećem prozoru prihvaćamo uvjete, te klikom na „Next“ otvaramo postavke. Odabiremo „Unity 2018...“ opciju, „Documentation“, te „Unity game develpment for Microsoft Visual Studio Enterprise“ ako ne posjedujemo neku drugu verziju Visual Studio-a.

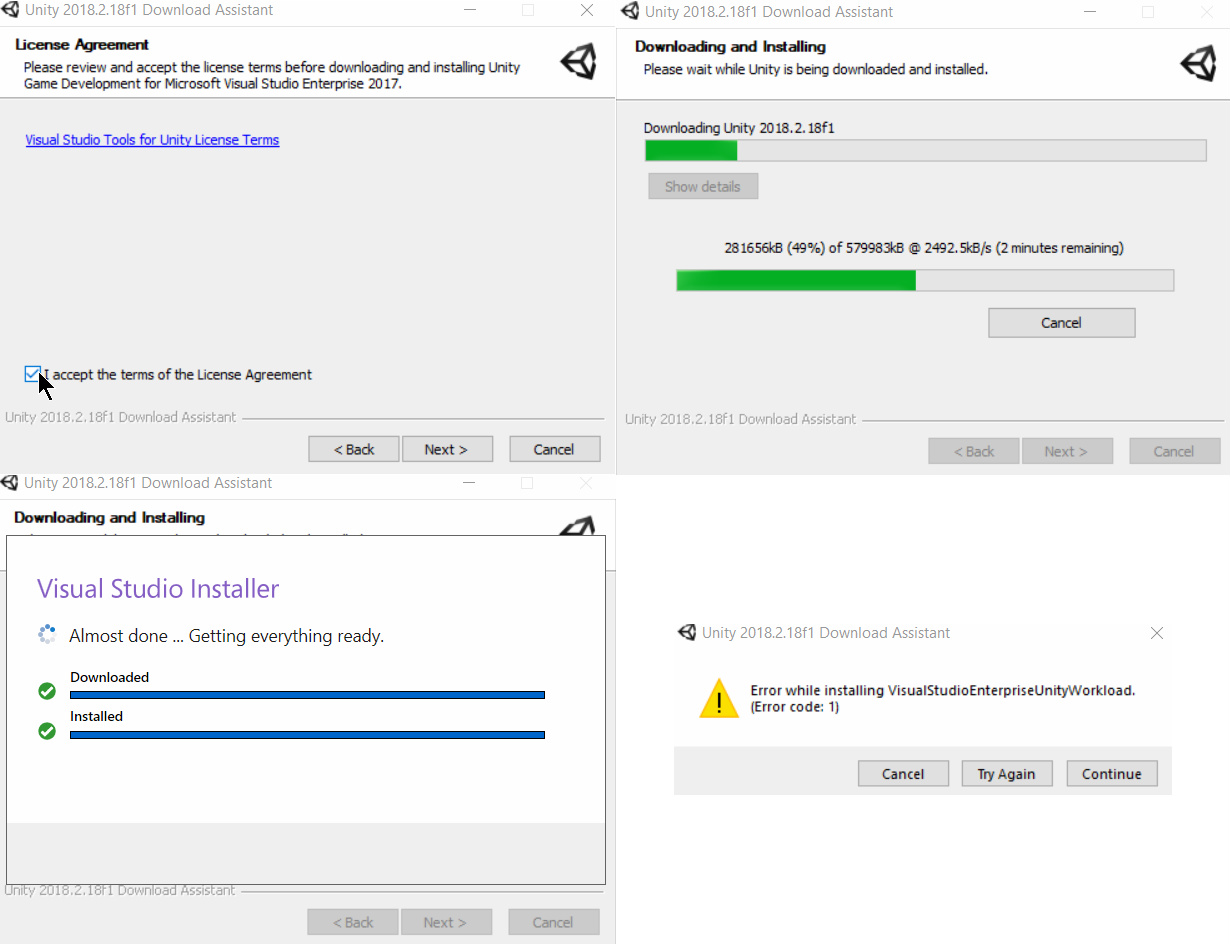
Ovisno o vrsti projekta ponuđene su ostale opcije, koje nude pomoćne komponente koje se koriste za izgradnju igara za Linux, iOS, WebGL, te slično.  
Pritiskom na „Next“ otvara se novi prozor u kojemu odabiremo lokaciju na kojoj želimo spremiti Unity. Koraci su vidljivi na Slika 2‑3.



Slika 2‑3 Instalacija Unity-a (1)

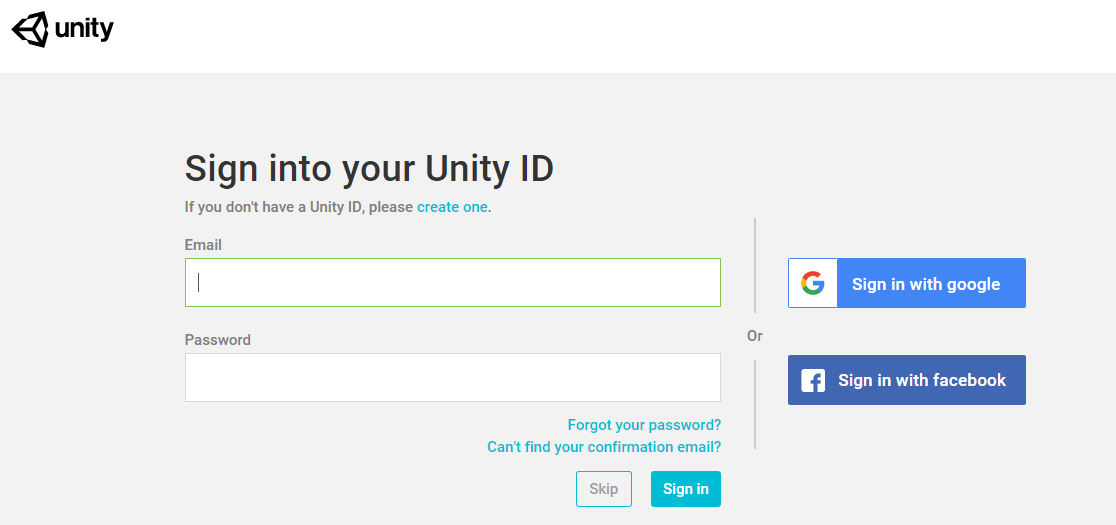
Kada smo zadovoljni s lokacijom, pritiskom na „Next“ otvara se novi prozor na kojemu prihvaćamo uvjete, te pritiskom na „Next“ dolazimo do instalacije.  
Instalacija traje nekoliko minuta, ovisno o tome koliko je datoteka potrebno za skinuti te instalirati.

Moguće su greške poput „Error 1“, Slika 2‑4, koja nastaje prilikom instalacije VS-a ako već postoji. Ako se pojavi, pritiskom na „Continue“ nastavlja se instalacija bez obzira na VS[[4]](#footnote-1). Instalacija i proces instalacije vidljivi su na Slika 2‑3, Slika 2‑4.



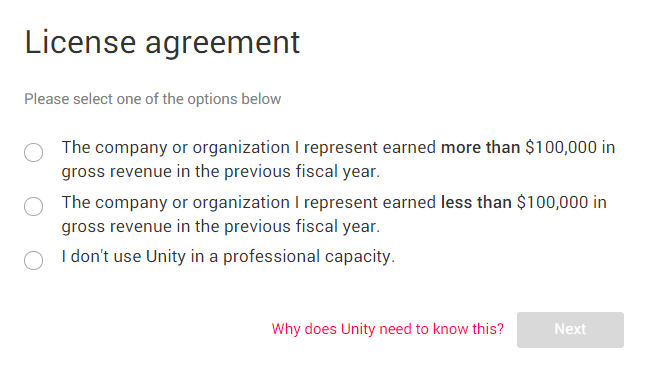
Slika 2‑4 Instalacija Unity-a (2)

Nakon instalacije, otvaramo Unity program, otvara nam se prozor za prijavu. Potrebno je napraviti Unity račun, te se prijaviti s tim podatcima, Slika 2‑5.



Slika 2‑5 Unity prijava

Nakon prijave moramo odabrati vrstu Unity paketa kojeg koristimo, u ovom slučaju „Personal“, te odabrati jedno od polja vidljivih na Slika 2‑6.

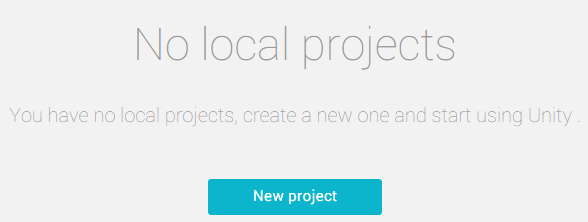


Slika 2‑6 Unity Licenca

Nakon toga, klikom na „Next“, obavlja se provjera licence. To je ujedno i zadnji korak instalacije Unity-a.

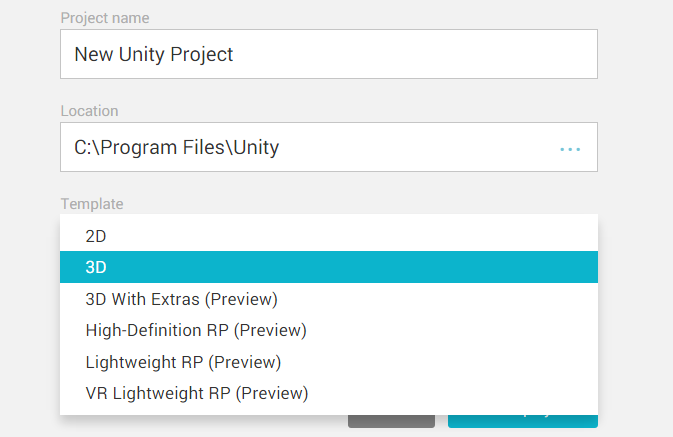
## Izrada novog projekta

Izgradnja novog Unity projekta počinje odabirom „New Project“ u Unity programu, Slika 2‑7.



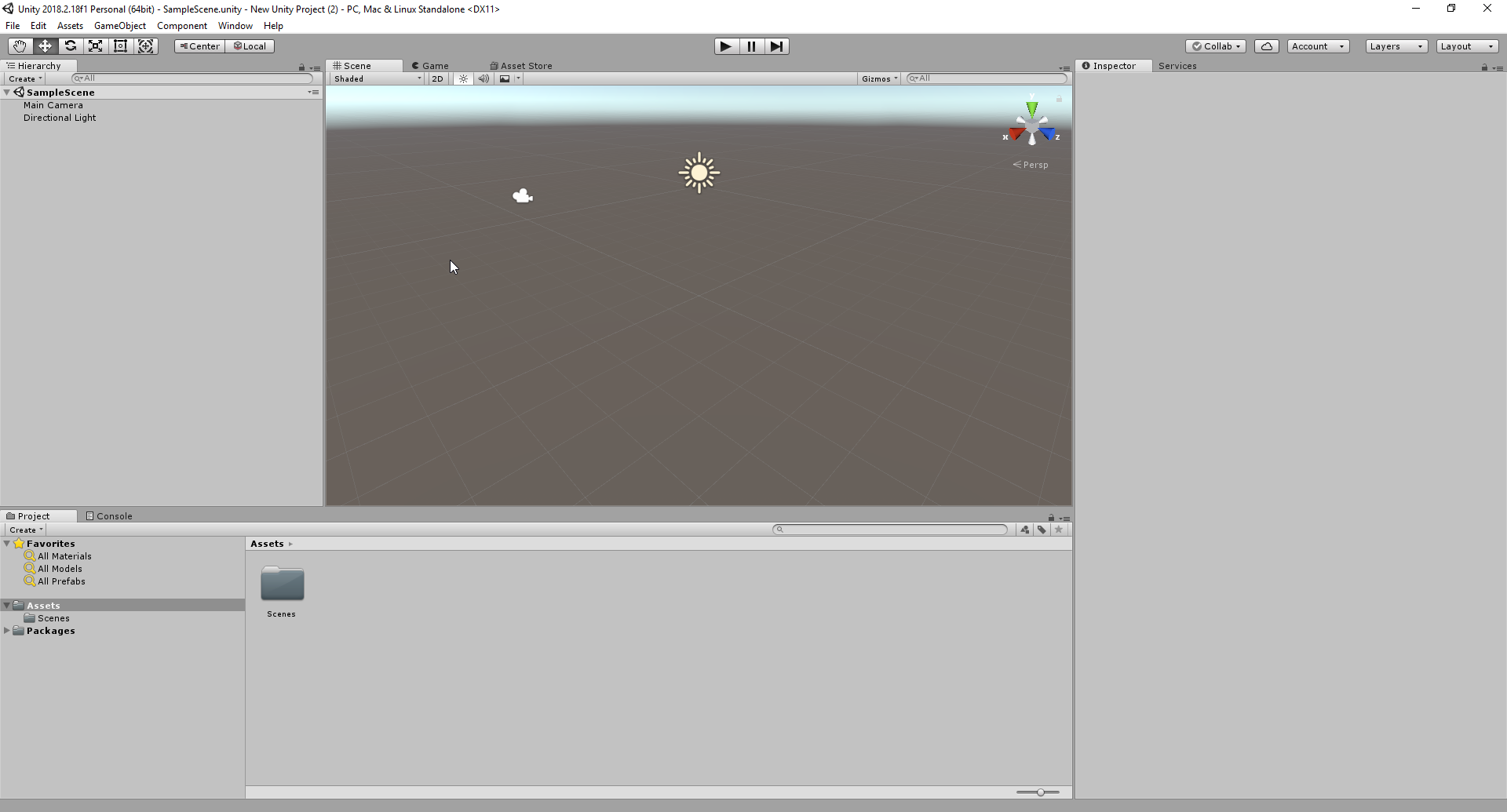
Slika 2‑7 Novi projekt

Time otvaramo opcije, Slika 2‑8, kod kojih odabiremo naziv projekta, gdje ga želimo pohraniti te vrstu projekta na kojem ćemo raditi.



Slika 2‑8 Izrada novog projekta

Nakon kratkog čekanja stvoren je novi projekt te otvoren prozor prikazan na Slika 2‑9.



Slika 2‑9 Pokrenut Unity projekt

# Izrada osnovnog nivoa

Da bismo mogli raditi u projektu potrebno je izgraditi osnovni nivo koji će nam služiti za testiranje i izradu ostalih elemenata koje nadodajemo na projekt.

## Upoznavanje s radnom okolinom

U svrhu izrade nivoa moramo se upoznati s radnom okolinom.

Unity prozor sastoji se od alatne trake koja ima sljedeće opcije, Slika 3‑1. Alatna traka omogućava nam pristup upravljanju datotekama, uređivanju i opcijama uređivanja, imovinom, odnosno skriptama koje pišemo, materijalima koje ubacujemo i slično, koji se nalaze pod „Assets“, materijalima koje koristimo kao prethodno napravljenim elementima pod „GameObject“, „Component“ koji nam omogućava obradu pojedine komponente scene, „Window“ koji nam nudi mogućnost upravljanja scenom te „Help“ opcijom koja nam nudi pomoć pri izradi projekta.



Slika 3‑1 Unity alatna traka

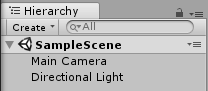
Ispod alatne trake nalazi se traka za upravljanje projektom. Na njoj se nalazi primjerice upravljanje u 3D okolini, 2D okolini, rotacija, skaliranje, pokretanje projekta, zaustavljanje, preskakanje scena, postavljanje slojeva, okvira itd. Pogledati Slika 3‑2.



Slika 3‑2 Unity traka za upravljanje projektom

Na lijevoj strani ekrana nalazi se hijerarhijski prikazan projekt, s njegovim materijalima i imovinom, Slika 3‑3.

Prazan projekt započinje s glavnom kamerom, koja se koristi za pozicioniranje slike igrača, te direkcionalnim svijetlom, koje se koristi kao izvor svjetlosti za scene.  
Svaka nova imovina nalazi se na ovom prozoru, te se koristi za ubrzani rad s imovinom.



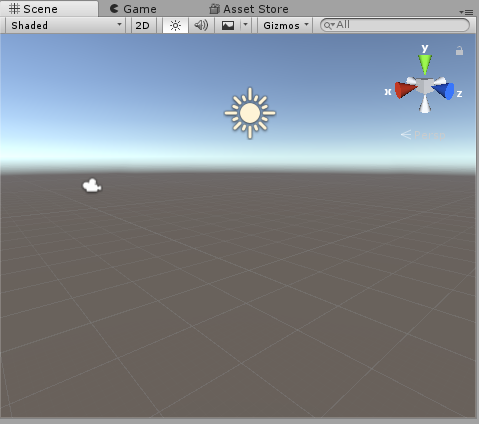
Slika 3‑3 Hijerarhijski prikaz projekta

Sredina Unity prozora nosi ekran u kojemu se nalazi scena, nad kojom izrađujemo projekt, igra u kojoj se može testirati kako bi izgledala zapravo igra, te dućan koji sadrži prethodno napravljene elemente, ali se plaća.

Ispod toga se nalazi upravljanje sa svijetlom, trenutno postavljeno na „Shaded“, prebacivanje s 2D na 3D prikaz prostora, postavljanje svjetla na upaljeno ili ugašeno itd...   
Pod tim opcijama nalazi se prozor na kojemu vidimo raster prostora, postavljen izvor svjetlosti te kameru koja pokazuje iz kojeg smjera i pod kojim kutem se vidi scena.

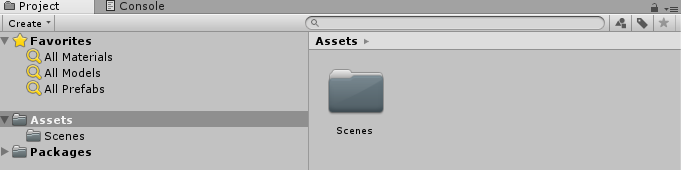
Klikom na „sunce“ ili „kameru“ prikazuje se njihove nove opcije. Ako želimo pomicati elemente na ovom prozoru, potrebno je kliknuti na „ručicu“, koja se vidi na Slika 3‑2.

Postavljanje osobne kamere za pregled prostora ne utječe na izgled igre, ni raspored kamere u samoj igri. U desnom kantunu vidljiva je ikona rastera prostora, kojom možemo pomaknuti osobni pogled na bilo koju os. Čitav prozor vidljiv je na Slika 3‑4.



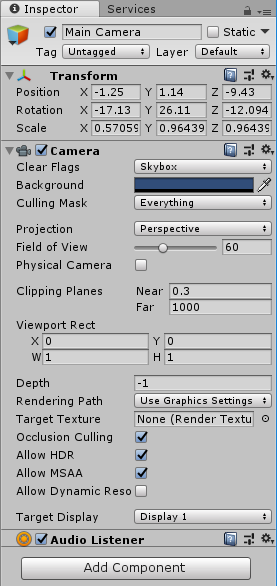
Slika 3‑4 Centralni prozor Unity-a

Ispod ovog prozora nalazi se prostor, Slika 3‑5, u kojemu vidimo trenutne materijale, skripte, pakete i scene koje koristimo. Dopuštena je potpuna sloboda nad upravljanjem ovim dijelom. Podjela se sastoji od materijala, modela, te „prefab-a[[5]](#footnote-2)“.



Slika 3‑5 Prostor s imovinom

Te na kraju, s desne strane Unity prozora vidljiva je traka za upravljanje pojedinom imovinom projekta, Slika 3‑6. Ovdje se postavljaju vrijednosti poput kuta rotacije, pozicije, imena imovine, skaliranja te sve ostale komponente koje ćemo iskoristiti kroz projekt.



Slika 3‑6 Traka za postavljanje vrijednosti imovine

U ovom djelu dodajemo sve skripte na željene elemente, postavljamo efekte, animacije itd...

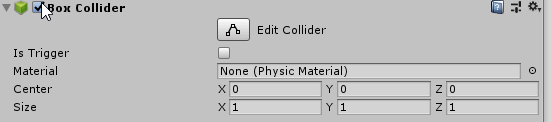
## Izrada nivoa

Nakon što smo se upoznali s osnovnim elementima Unity-a, možemo krenuti s izradom nivoa.

Desnim klikom na hijerarhiji otvara se prozor u kojemu su ponuđene razne opcije. Odabiremo 3D objekt, te kocku(Cube).

Nakon toga na istom prostoru odabiremo isto desnim klikom prazni objekt igre. Na desnoj strani prozora vidimo traku u kojoj ćemo promijeniti ime praznog objekta u „Nodes“, resetirati vrijednosti klikom na kotačić , te u taj novo stvoreni objekt postaviti prethodno stvoreni objekt kocke.

Nakon što smo to obavili, selektiramo kocku te njoj mijenjamo ime na „Node“ i postavljamo poziciju na 4-1-4, te skaliramo po želji.  
Sada taj objekt „Node“ povlačimo u polje nazvano „Prefab“, te se sada može koristiti kao projektna imovina. Pritiskom tipke „F“ fokusiramo se na objekt koji je trenutno selektiran.



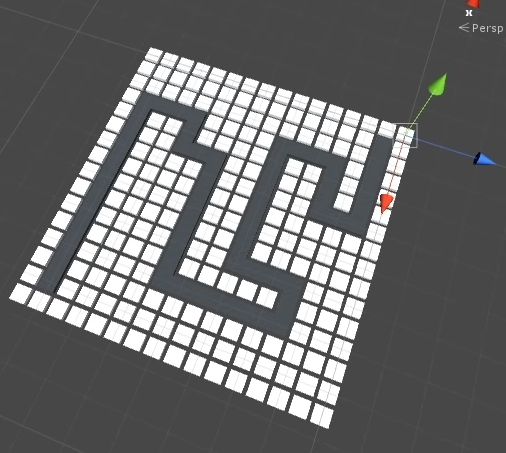
Slika 3‑7 Kontrola sudaranja

Kopiramo trenutno selektiran objekt kocke, koristeći kombinaciju „Ctrl+D“ tipki. Tako novo stvoreni objekt pomičemo držeći „Ctrl“ tipku te pritiskom na strelicu na glavnom ekranu pomičemo objekt. Moguće je kopirati više objekata istovremeno, selektiranjem s „Ctrl“ u hijerarhiji te zahvaćanjem željenih objekata. Nastavljamo tako kopirati i pomicati objekte dok ne stvorimo željenu mapu.

Pojedine objekte ćemo zamijeniti s novim „Ground“ objektima, tako da ih selektiramo, izbrišemo te istim postupkom dodamo nove objekte kao i za „Node/s“. Za te nove objekte, koje potrebno je promijeniti boju, jer ćemo od njega izgraditi put, te se mora razlikovati od zidova.

Također, da bi dopustili kretanje drugih likova preko njega, moramo isključiti sudaranje, vidljivo na Slika 3‑7.

Za nove materijale iz kojih izgrađujemo put, koristimo skaliranje, postavljamo na željenu vrijednost pojedini element i povezujemo ih s drugima samo dodirom na glavnom prozoru. Time smo povezali put za naše objekte protivnika, po kojemu će se kretati.  
Na kraju bi trebali dobiti željeni prozor, u ovom slučaju naš izgleda ovako, Slika 3‑8.



Slika 3‑8 Osnovni nivo

Dakako izgled same igre ovisit će i o pozadini. Tamno siva podloga koja se nalazi ispod igre napravljena je koristeći platno, koje se postavlja u 2D prostor te rastegne na veličinu koja je više puta veća od samog prostora nivoa igre.

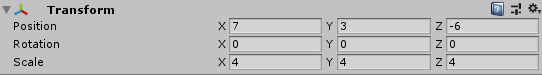
Platno izgrađujemo desnim klikom na područje hijerarhije, te dodavanjem platna u 2D prostoru. Takvo platno postavljamo malo ispod područja postolja, te boju mijenjamo u sivu.

# Protivnici

U prethodnom poglavlju dotakli smo se izrade objekata, te smo spomenuli objekte protivnika. Kao u svakoj igri ovoga tipa, potrebno je napraviti početak, odnosno izvor protivnika, te njihovo krajnje odredište.

## Izvor i krajnje odredište za objekte protivnika

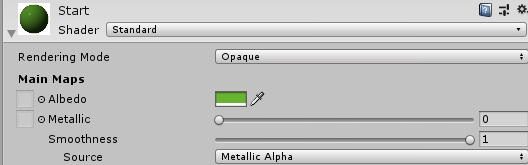
Za naš izvor, stvaramo novi objekt kocke kao i u prethodnom poglavlju, te ga nazivamo „START“. Postavljamo ga nad željeno mjesto te podižemo po y osi, tako da se nalazi iznad područja staze, te ga skaliramo da bude malo veći od okoline, Slika 4‑1.   
Također, potežemo taj objekt iz hijerarhije u „prefab“ dio, tako da ga možemo koristiti u projektu.



Slika 4‑1 Pozicija izvora

Osim toga, za selektirano polje postavljamo boju kojom će se isticati od okoline.  
Za to je potrebno stvoriti novi materija.

Da bismo kreirali materijal, postavljamo miš nad prostorom s imovinu, Slika 3‑5, nakon čega, desnim klikom otvaramo opcije, u kojima biramo „Material“ opciju.  
Za novi materijal odredimo ime, te ga postavimo na START objekt, tako da ga povučemo na „START“ objekt u glavnom prozoru.  
Boju postavljamo na desnoj strani Unity prozora, vidljivo na Slika 4‑2.



Slika 4‑2 Postavljanje boje

Na isti način stvaramo odredište, postavljamo na željenu lokaciju, stvaramo novi materijal, koji povlačimo na odredište kao i u prethodnom koraku te ga bojamo u željenu boju.

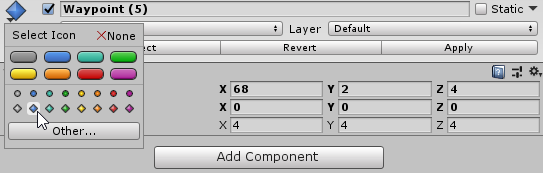
## Izrada puta

Da bi se protivnici kretali potrebno je odrediti putanju kojoj se kreću. To ćemo ostvariti tako što kreiramo „Waypoint“ objekte.  
„Waypoint“ objekt stvaramo kao „EmptyObject“, nazivamo ga „Waypoint“ te ga postavljamo u novi prazni objekt, nazvan „Waypoints“, da bude sortirano.

Da bi takve objekte vidjeli, moramo ih postaviti na neki oblik. To ćemo ostvariti tako da klikom miša otvorimo dodatni meni na desnom meniju Unity prozora, vidljivo na Slika 4‑3.

Takve objekte nećemo vidjeti u glavnoj igri jer ih ne postavljamo u „prefab“ dio projekta, no koristit će nam za postavljanje puta.

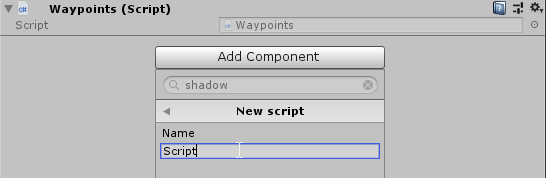
Kopiramo objekte te ih postavljamo na željenu putanju, pazeći pri tom da se nalaze nad objektima koji imaju isključenu opciju kolizije.   
Nakon što smo postavili objekte puta, potrebno je napraviti skriptu, odnosno kod, koji će im dati željenu funkcionalnost.



Slika 4‑3 Postavljanje Waypoint-a

Da bismo napravili skriptu koja je vezana uz objekt, selektiramo mapu objekta u kojemu se nalaze svi objekti „waypoint“.

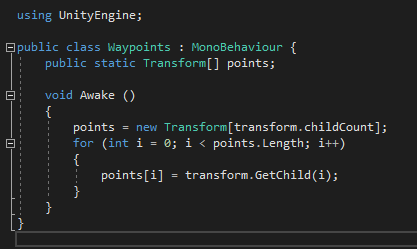
Na desnoj strani ekrana, Slika 3‑6, vidimo na dnu opciju dodavanja nove komponente. Klikom na to otvara se ponuda mogućih elemenata koje možemo dodati, selektiramo skriptu, te je nazovemo željenim imenom. Pritiskom na „Enter“ otvaramo skriptu u VS-u.



Slika 4‑4 Dodavanje skripte

U VS-u bit će kreirana skripta s prethodno formiranom strukturom.   
Kreiramo novu klasu, „public[[6]](#footnote-3)“. Unutar te klase deklariramo „public static[[7]](#footnote-4)“ niz, gdje „Transform“ oblik referencira objekte iz igre, te ga nazivamo „points“.

Nakon toga kreiramo funkciju tipa „void Awake[[8]](#footnote-5)“. Ta funkcija će pronaći sve objekte kojima je „Waypoints“ roditelj, te ih pohraniti u niz „points“.  
Prvo ćemo kreirati niz „points“, te postaviti njegovu duljinu na „transform.childCount“, odnosno na broj djece unutar roditelja „Waypoints“. Nakon toga iteriramo kroz svako dijete te ga pohranjujemo u i-ti član niza „points“. Na kraju kod izgleda ovako, Slika 4‑5.



Slika 4‑5 Waypoint kod

Ovako kreiran sistem organiziranja putanja je koristan jer objekt protivnika ne mora znati gdje se nalaze sve točke koje mora obići, već zna da ih nije obišao sve dokle god postoji još jedna u nizu, stoga nosi informaciju samo za sljedeću točku.

## Izrada protivnika

Sada, kad znamo putanju kojom će se kretati objekti protivnika, kreiramo te objekte.

Kao i u prethodnim koracima kreiramo nove objekte protivnika, desnim klikom na polje hijerarhije, te selektiramo oblik sfere. Kao i prije, kreiramo novi materijal koji postavljamo na njega, dajemo mu boju, te ga postavljamo na početnu lokaciju.

Sada kada smo zadovoljni s njime, povučemo ga u „prefab“ dio, tako ga ga možemo koristiti kao imovinu projekta. Kao i u prethodnom poglavlju kreiramo skriptu te je nazivamo „Enemy“. Ta skripta upravljat će protivnicima, odnosno njihovim kretanjem.

U liniju 5, kreiramo varijablu kojom ćemo upravljati brzinom protivnika. Tako možemo dinamički mijenjati brzinu u igri. Te nakon toga inicijaliziramo 2 privatne varijable, jer ih koristimo samo u ovoj skripti.

U funkciji „Start[[9]](#footnote-6)“ referenciramo target na „Waypoints.points“ niz iz prethodne skripte, sa Slika 4‑5.

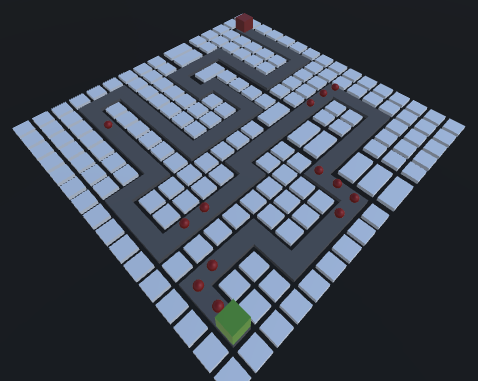


Slika 4‑6 Enemy skripta

U „Update“ funkciji postavljamo vektor za orijentaciju kretanja, kroz vektor „dir“, te se odredi kretanje s translacijom tijela u sljedećoj liniji koda, u kojoj postavljamo normaliziranu brzinu kretanja. Ona je osnova kretanja u Unity igrama.   
Pomnožimo je s brzinom, te „Time.deltaTime“ koja omogućava da se brzina računa za svaki okvir, tako da ne bi došlo do pre velike brzine ili pre male ovisno o performansama računala, te na kraju postavljamo gdje se želimo kretati, to je „Space.World“, predefiniran prostor Unity-a. Pogledati Slika 4‑6.

Zasad pronalazimo samo jedan „Waypoint“, da bismo pronašli novi, potrebno je napraviti „Update[[10]](#footnote-7)“ funkciju, nakon što je dosegnuta točka, uz mali otklon (<= 0.2f), jer u igrama nikad preciznost nije 100%.   
Pozivamo novu metodu, „GetNextWaypoint“, koja će obaviti pronalazak novog objekta „Waypoint“ sve dok ne nađe zadnji. Odnosno „waypointIndex“ će rasti i pridodijeliti „targetu“ novi „Waypoint“.  
Osim toga, obavlja se provjera svaki put da li je dosegao zadnji „Waypoint“, ako je, obavlja se „Destroy“ metoda, koja uništava dotični objekt protivnika. S obzirom na to da je potrebno neko vrijeme dok se uništi taj objekt, programski kod često nastavlja s radom, jer se obavlja sinkrono, te stoga stavljamo „return“ koji osigurava da se izađe iz ove metode za taj objekt.

Time smo izgradili objekt protivnika, koji sada možemo testirati u Unity-u , pokretanjem na tipku strelice iznad glavnog prozora, te prebacivanjem u „Game“ prozor, Slika 4‑7.



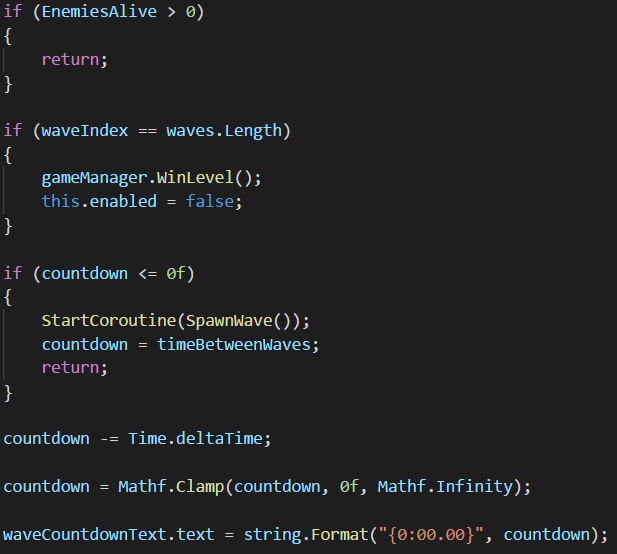
Slika 4‑7 Kretanje protivnika

## Skripta za generiranje protivnika

Da bi ostvarili generator na izvoru potrebno je osmisliti način na koji ćemo stvarati nove protivnike. Najlakši način bi bio stvoriti ih u intervalu, no onaj koji koristimo je generiranje protivnika kada su svi prethodni uništeni. Stoga radimo proračun broja živih protivnika.

* Kroz prvi uvjet provjeravamo da li postoje entiteti protivnika koji su aktivni.
* Drugi uvjet je provjera da li je završena razina igre. Ako je, referencira se metoda iz „gameManager“ skripte, te se zaustavlja izvođenje ove funkcije.
* Treći uvjet stvara protivnike sve dok brojač ne dođe do 0 sekunda. Brojač se modificira pomoću umanjivanja s realnim vremenom igre.

Uvjeti kojim se odlučuje o stvaranju neprijatelja vidljivi su na Slika 4‑8.



Slika 4‑8 Uvjeti stvaranja neprijatelja

U istoj skripti stvaramo i pozvanu metodu „spawnWave“. Ta metoda se poziva za svaku instancu, te stoga koristimo „IEnumerator“ koji je ekvivalent „For-each“ pozivu.

Ova metoda ažurira status igrača, točnije njegovog napretka kroz runde, te ako ne postoje aktivni protivnici stvara nove protivnike pozivajući „SpawnEnemy“ metodu. Također nakon svakog stvorenog protivnika obavlja se „yield return“ koji omogućava rad enumeratora, jer prividno stvara klasu za svaki novi objekt, za zadani interval vremena koji se proračuna.

Samo stvaranje protivnika obavlja se kroz „SpawnEnemy“ metodu, koja referencira se na „Enemy“ skriptu, te instancira protivnike na poziciji izvora protivnika, te njegovoj rotaciji.

Čitav kod vidljiv je na Slika 4‑9.



Slika 4‑9 Val i stvaranje neprijatelja

Na ovaj način ostvarili smo ubrzavanje izlaska protivnika ovisno o nivou, odabir vrste protivnika, njihovo instanciranje te odluku o novom stvaranju.

## Skripta za obradu dolaska na odredište

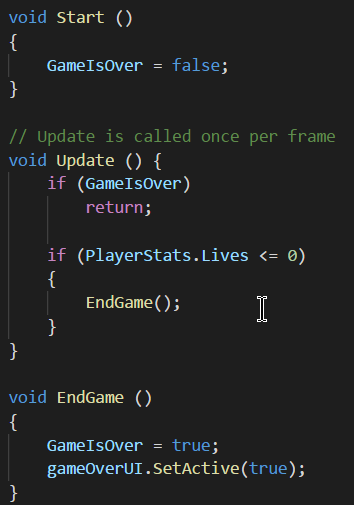
Nakon što objekt protivnika dođe do kraja, osim što ga želimo uništiti, potrebno je obraditi podatke poput oduzimanja života, provjere kraja igre i slično.

Za obradu oduzimanja života brine se sama skripta „Enemy“, koja će podatke života iz statusne skripte „PlayerStats“, umanjiti za jedan svaki put kada protivnik dođe do odredišta. Slika 4‑10.



Slika 4‑10 Kod za umanjivanje života

Isto tako, potrebno je napraviti obavijest o izgubljenoj igri, ako broj života dođe do 0. To se obavlja u novoj skripti „GameManager“ koja prvotno postavlja vrijednost završene igre na „false“, te se radi provjera, na svakom novom izgrađenom ekranu, da li je broj života manji ili jednak nuli. Ako je, poziva se metoda „EndGame“, koja postavlja vrijednost „GameIsOver“ na istinu, te vrijednost „gameOverUI“ varijable iz „GameOver“ skripte na istinu. Slika 4‑11.  
Ta skripta pak postavlja natpis o kraju igre i drugim stvarima, no bit će obrađena kasnije.

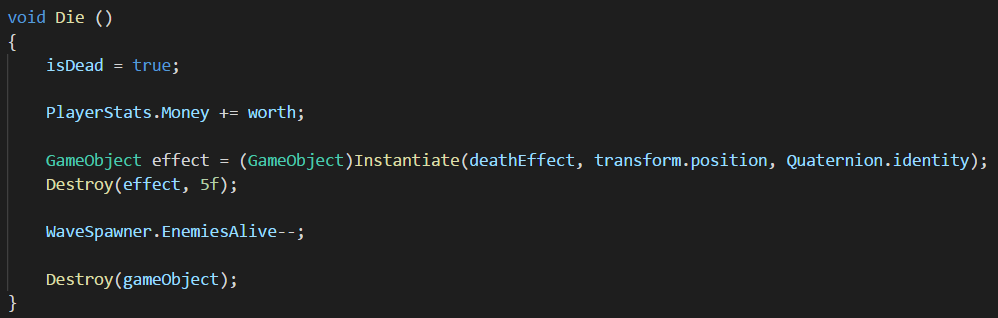


Slika 4‑11 Kod za obavijest korisnika o kraju igre

## Obrada uništavanja protivnika

Ako objekt protivnika uništimo na vrijeme, odnosno prije no uspije doći do zadnjeg odredišnog markera, potrebno je uvećati broj novca kojeg imamo. Također stvaramo efekt nestajanja protivnika, da bi se bolje označio čin uništavanja.

Kako bi nagradili korisnika za uništavanje protivnika jednostavno uvećavamo „PlayerStats.Money“ varijablu referenciranu iz „PlayerStats“ statusne skripte za vrijednost neprijatelja, koja je predefinirana u varijabli „worth“. Slika 4‑12.



Slika 4‑12 Metoda uništavanja protivnika

Efekt uništavanja protivnika stvoren u Unity okružju sada referenciramo u skripti „Enemy“, te ga pozivamo pri metodi „Die“. Linija koda efekta instancira objekt igre, postavlja efekt pomoću „deathEffect“ varijable Unity-a, postavlja poziciju sa „transform.position“ i postavlja da nema rotacije pomoću „Quaternion.identety“ svojstva.

Nakon što se inicijalizirao objekt igre toga tipa, poziva se u „Destroy“ metodi, s duljinom trajanja efekta od 5 sekunda.

Također, obavještavamo „WaveSpawner“ skriptu da se broj protivnika smanjio, jer nas taj broj obavještava kada završava nivo[[11]](#footnote-8). Te nakon toga uništavamo sam objekt iz igre, tako da nestane s korisnikova ekrana i iz memorije.

Ovim je završeno poglavlje o protivnicima, te će se kasnije samo referencirati na isto.

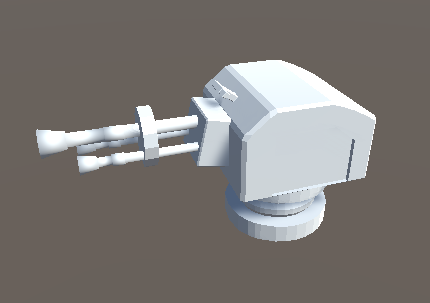
# Obrambene utvrde

Utvrde se koriste kako bi se zaustavio napredak neprijatelja i obranila igračeva baza.

Unutar ove igre imat ćemo 3 vrste utvrda:

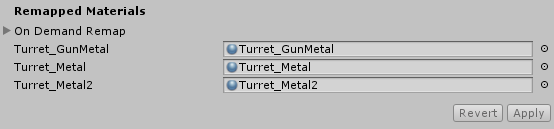
|  |  |  |
| --- | --- | --- |
| Slika 5‑5‑1 Mitraljez | Slika 5‑5‑2 Bacač Raketa | Slika 5‑5‑3 Laser |

Utvrde i njihove odgovarajuće projektile se unosi u unity dodavanjem njihovih 3d modela. To se radi „drag&dropom“ „blend“ ili „fbx“ datoteke unutar otvorenog unity prozora.Novododana utvrda se sprema u poseban direktorij nazvan resursi.



Slika 5‑5‑4 Novi model utvrde

Utvrda su početno bezbojne Slika 5‑4 ,a bojaju se dodjeljivanjem materijala pojedinim komponentama utvrde Slika 5‑5.



Slika 5‑5‑5 Dodjeljivanje materijala

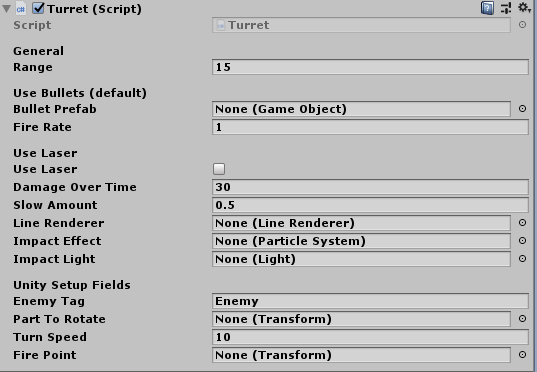
Način rada utvrda opisan je u Turret.cs skripti.

Utvrde, iako su različite dijele određene funkcionalnosti te se radi jednostavnosti spremaju u zajedničku datoteku.

Turret.cs sadrži attribute utvrda, te funkcije za ciljanje i gađanje.

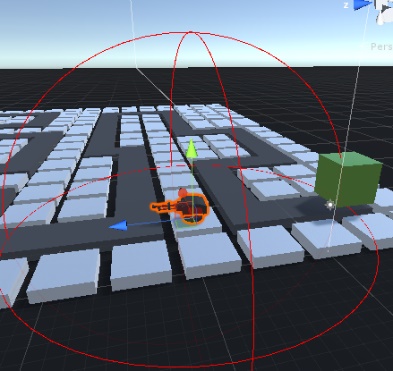
## Atributi Utvrda

Atributi utvrda se opisuju u Turret.cs datoteci Slika 5‑6 .



Slika 5‑5‑6 Atributi utvrda

* Domet ili „range“ Slika 5‑7 određuje maksimalnu udaljenost u krugu od 360 stupnjeva na kojoj utvrda može pronalaziti neprijatelje, prikazana je kružnicom na Slika 5‑7.



Slika 5‑5‑7 Domet utvrde

* „Bullet Prefab“ određuje tip projektila koji se koristi za specifičnu utvrdu Slika 5‑8, Slika 5‑9.

|  |  |
| --- | --- |
| Slika 5‑5‑8 Metak | Slika 5‑5‑9 Raketa |

* „Fire rate“ označava koliko će se metaka ispaliti u sekundi
* „Use Laser“ je posebna funkcija koja se radi za lasersku utvrdu i kreira liniju od utvrde do mete te definira koliko života (eng. Hitpoints) oduzima po sekundi.Može imati efekt usporavanja neprijatelja.
* „Line Renderer“ određuje tip linije koja se prikazuje.Pomoću tog atributa možemo za normalni laser staviti žutu liniju a za laser koji usporava zelenu.
* „Impact-effect“ je efekt najčešće eksplozija koja se dešava kad se projektil dođe u kontakt s metom.
* „Impact-light“ je potreban pošto smo omogućili više razlicih boja lasera. Promjenom boje lasera sukladno se mijenja boja „impact-effecta“ određena „Impact-light-om“.
* „Unity Setup Field“ sadržava generalnu konfiguraciju levela.
* „Enemy Tag-om“ određujemo oznaku koju treba imati objekt da bi se prepoznao kao neprijatelj.
* „PartToRotate“ je dio same utvrde na slici Slika 5-10 određen crvenom bojom te se odnosi na dio utvrde koji će pratiti kretanje neprijatelja.

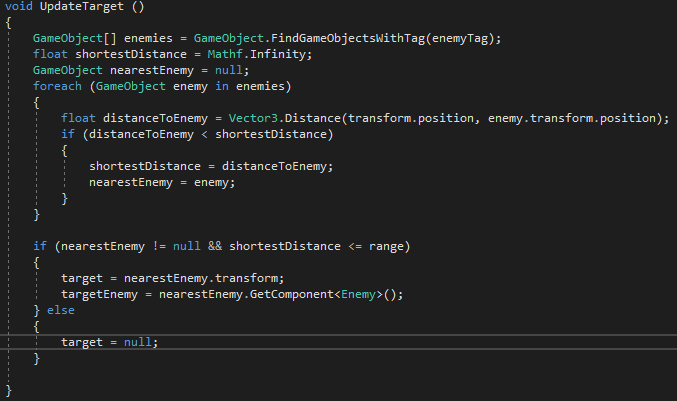


Slika 5‑5‑10 PartToRotate Bacača raketa

* „Turn speed“ određuje brzinu rotacije „PartToRotate“ objekta, ukoliko bi ovo bilo izostavljeno prijelazi utvrde bi bili instantni.
* „FirePoint“ je dio „PartToRotate“ objekta, koji sluzi kao točka ispaljivanja projektila tj. točka u kojoj se instanciraju projektili.

## Ciljanje

Funkcija UpdateTarget Slika 5‑11 se koristi za ciljanje i ista je za sve utvrde



Slika 5‑5‑11 Funkcija za ciljanje

Radimo niz objekata zvanih „enemies“ I punimo ga sa svim objektima koji imaju tag(oznaku) „enemy“ Slika 5‑12 .



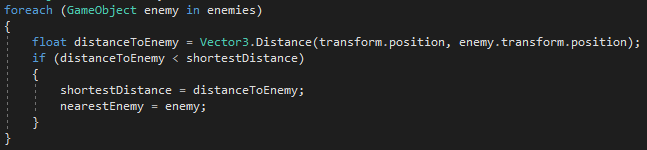
Slika 5‑5‑12 Pohranjivanje neprijatelja

„shortestDistance“ varijabla se inicijalizira na beskonačnost, a objekt „nearestEnemy“ na null jer pri početku igre ne postoji ni jedan neprijatelj Slika 5‑13.



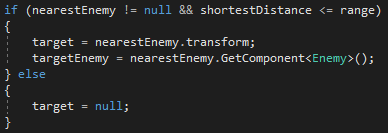
Slika 5‑5‑13 Početno stanje funkcije

Prilikom svake iteracije funkcije update, prolazimo kroz sve objekte u „enemies“ nizu objekata te računamo udaljenosti svakog neprijatelja od utvrde. Ako je udaljenost nekog neprijatelja X manja od trenutačne najmanje , neprijatelj X se postavlja kao najbliži neprijatelj a njegova udaljenost se postavlja kao najmanja udaljenost Slika 5‑14.



Slika 5‑5‑14 Traženje najbližeg neprijatelja

Nakon provjere udaljenosti gleda se da li postoji najbliži neprijatelj, te da li se nalazi unutar dometa utvrde. Ukoliko su uvjeti zadovoljeni najbliži neprijatelj se postavlja se kao trenutna meta utvrde. Važno je napomenut da se utvrda neće držati istog neprijatelja kao metu sve dok ga ne uništi nego samo dok joj je on najbliži, u trenutku kada neki drugi neprijatelj postane najbliži , fokus se automatski prebacuje na njega. Ukoliko se ne može pronaći najbliža meta, varijabla se postavlja na null i utvrda čeka novu metu Slika 5-15.



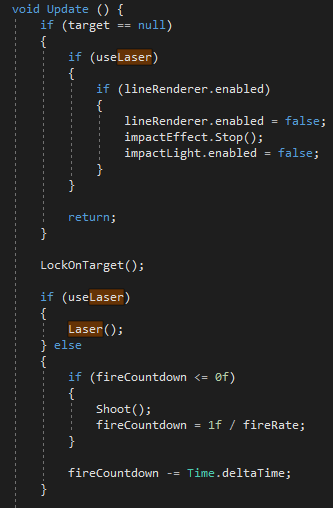
Slika 5‑5‑15 Određivanje najbližeg neprijatelja

### Pucanje

Pucanje se izvršava kroz dvije funkcije:

* Update
* Shoot

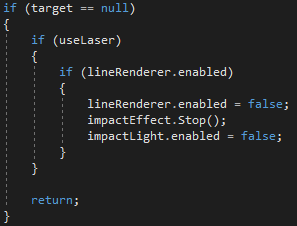
#### Update



Slika 5‑5‑16 Funkcija za pucanje

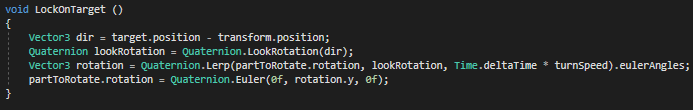
Funkcija update Slika 5‑5‑17 pokreće se svaki okvir.

Prvo gleda za laserske utvrde da li postoji meta i isključuje se laser ukoliko ne postoji Slika 5‑17 .



Slika 5‑5‑18 Funkcija za paljenje i gašenje lasera

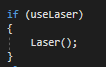
Nakon toga poziva „LockOnTarget“ Slika 5‑18 funkciju koja računa razliku između položaja utvrde i mete te određuje rotaciju koju „partToRotate“ treba napraviti kako bi pratila metu.



Slika 5‑5‑19 Funkcija za ciljanje

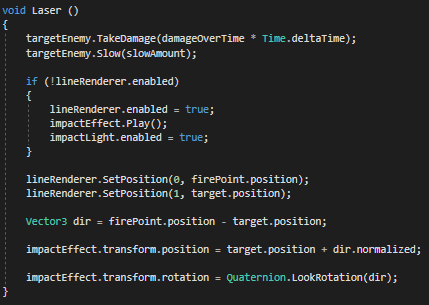
Nakon što partToRotate objekt dođe u odgovarajući položaj započinje se proces pucanja.

U procesu pucanja prvo se provjerava da li je utvrda koristi lasere Slika 5‑19 .



Slika 5‑5‑20 Provjera tipa utvrde

Ako se uistinu koristi utvrda tipa laser, a protivnik je u dometu, poziva se funkcija „Laser“ Slika 5‑20 koja upravlja pucanjem lasera.



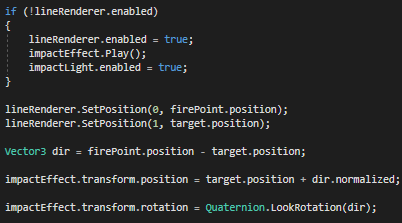
Slika 5‑5‑21 Funkcija lasera

Ona automatski obavlja oduzimanje života, te usporava objekt ako ima taj efekt Slika 5‑21.



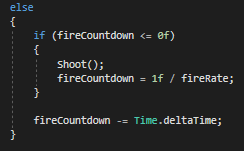
Slika 5‑5‑22 Oduzimanje života za laser

Uz to radi projektil tj. liniju koja se proteže od „firePoint-a“ utvrde do mete te uključuje sve pripadajuće efekte Slika 5‑22.



Slika 5‑5‑23 Kreiranje lasera

Ako utvrda nije tipa laser gleda se da li je utvrda spremna za pucanje.To je određeno varijablom „fireCountdown“ koja se s vremenom smanjuje Slika 5‑23.  
„fire rate“ pak određuje koliko puta će se u sekundi ispaliti metak Slika 5‑23.

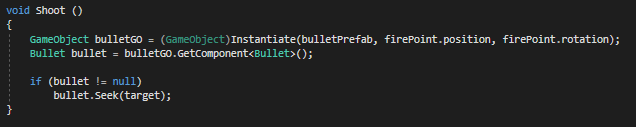


Slika 5‑5‑24 Brzina pucanja ne-laserskih utvrda

#### Metoda pucanja

„Shoot“ Slika 5-24 je jednostavna funkcija koja na poziciji firePoint točke radi novu instancu objekta projektila.

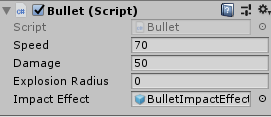
Svojstva i funkcionalnosti projektila su opisane u Bullet.cs skripti.



Slika 5‑5‑25 Funkcija „Shoot“

## Bullet.cs

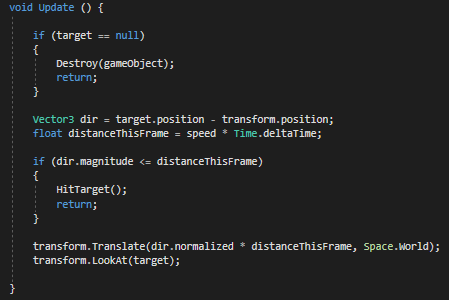
Sadrži atribute i funkcionalnosti projektila Slika 5‑25.



Slika 5‑5‑26 Atributi metka

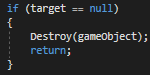
* „Speed“ određuje brzinu metka.
* „Damage“ određuje koliko života oduzima po pogođenom projektilu.
* „Explosion Radius“ se odnosi samo na raketne projektile te određuje mali radius u kojem se primjenjuje puna ili djelomična šteta koju projektil izazove primarnoj meti.
* „Impact Effect“ određuje efekt koji će se prikazati prilikom pogotka projektila.

### Update



Slika 5‑5‑27 Funkcija „Update“ za metke

Funkcija se poziva svaki okvir Slika 5-26.  
Provjerava se da li postoji meta,te ukoliko nepostoji osigurava da se projektil neće kreirati Slika 5-27.



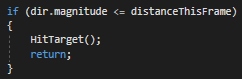
Slika 5‑5‑28 Provjera postoji li meta

Računa distancu koju će projektil preć u trenutnom okviru Slika 5‑28.



Slika 5‑5‑29 Brzina metka

Te se provjerava da li je veća od trenutne udaljenosti projektila i mete Slika 5‑29. Registracija pogotka proizlazi iz poziva funkcije „HitTarget“ Slika 5‑31.



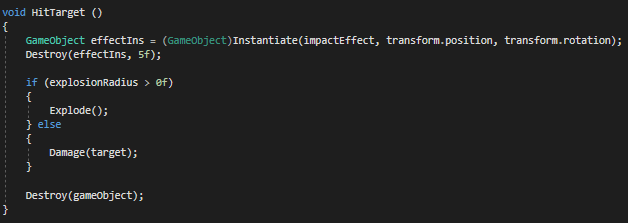
Slika 5‑5‑30 Detekcija udarca

Na kraju pomiče projektil prema meti Slika 5‑30.



Slika 5‑5‑31 Translacija metka

### HitTarget



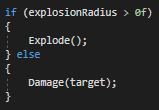
Slika 5‑5‑32 Funkcija „HitTarget“

S obzirom da već znamo da je projektil pogodio metu, stvaramo efekt pogotka za projektil, te ga uništavamo Slika 5‑31, Slika 5-32.



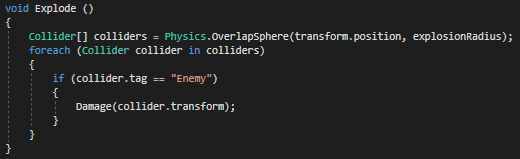
Slika 5‑5‑33 Uništavanje metka nakon pogotka

Ako je projektil raketa, što registriramo preko „explosion Radius-a“ koji je onda veći od nula, pozivamo funkciju „Explode“ Slika 5‑33 .



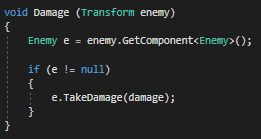
Slika 5‑5‑34 Domet explozije rakete

„Explode“ funkcija koristi fiziku „Unity Engine-a“ kako bi odredila koji neprijatelji ulaze u zonu udarca Slika 5‑34, te primjenjuje na njih štetu kroz funkciju „Damage“ Slika 5‑35.



Slika 5‑5‑35 Funkcija „Explode“

Ako je projektil običan metak odmah se poziva „Damage“ funkcija Slika 5‑35 jer ne posotji komponenta šrapnela.. „Damage“ funkcija dohvaća podatke mete te oduzima joj životne bodove referenciranjem funkcije iz „Enemy.cs“.



Slika 5‑5‑36 Funkcija „Damage“

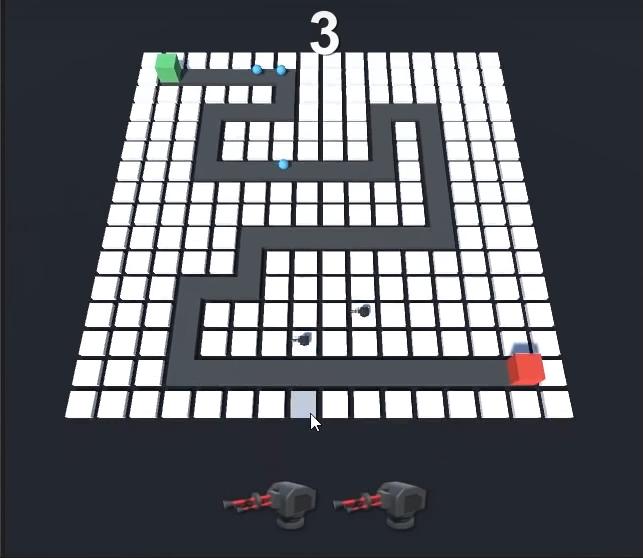
## Postavljanje utvrda

Da bismo omogućili korisniku da koristi naše definirane utvrde, potrebno je ostvariti interaktivnu skriptu koja će povezivati statičke elemente igre, poput postolja, samih utvrda…

### Interakcija s korisnikom

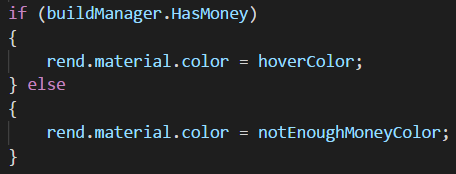
Stvaramo skriptu za „Nodo-ve“, koja će omogućiti postoljima da prihvaćaju utvrde. Prvo postavljamo interaktivnu jedinicu za sama postolja, s naumom da korisnik vidi na koje točno postolje pokušava postaviti utvrdu, odnosno iznad koje se nalazi pokazivač miša.

To smo ostvarili promjenom boje kada korisnik pređe mišem preko postolja, te makli taj efekt, odnosno resetirali na početno čim se više ne nalazi na tom polju, Slika 5‑1.



Slika 5‑1 Interakcija podloge i korisnika

Korisnika upozoravamo pri gradnji ukoliko gradi iznad već postojeće utvrde. U tom slučaju korisnika obaviještamo porukom o nemogućnosti gradnje. Također ako korisnik pokušava izgraditi utvrdu, a nema dovoljno novca, postolje iznad koje je miš postaje crveno. Primjer koda za to vidljiv je na, Slika 5‑2.

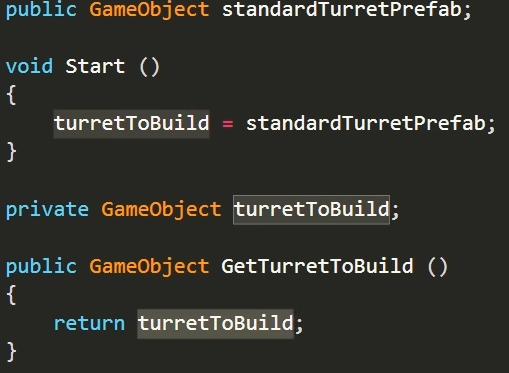


Slika 5‑37 Postavljanje boje ovisno o uvijetu

### Gradnja utvrda

Nakon što smo se pozabavili s interakcijom te osnovnim problemima na koje se može naići pri korištenju igre, fokusiramo se na gradnju samih utvrda.

Za to stvaramo skriptu „BuildManager“, za koju osiguravamo da je jedinstvena, jer želimo imati samo jednu „BuildManager“ varijablu koja se povezuje s igrom.  
Referenciramo se na objekte iz igre, te omogućavamo korištenje „GetTorrentToBuild“ metode bilo gdje, jer nam je potrebna i za vanjske skripte, Slika 5‑3.



Slika 5‑38 Inicijalizacija objekata igre u skripti

Sama gradnja utvrda postavlja se povezivanjem „prefab“ komponenti iz igre, s internom varijablom iz skripte. Osiguravanje da se utvrda nalazi na lokaciji miša, obavljamo tako što radimo izračun pozicije preko „transform“ metoda. Slika 5‑4.



Slika 5‑39 Kod za izgradnju utvrde

Prilikom postavljanja, došli smo do zaključka da pozicija nije u potpunosti željena, te smo postavili pomak u poziciji, jer je potrebno osigurati da se utvrda nalazi iznad postolja, ne unutar.   
Taj problem je riješen koristeći varijablu skripte, koju podešavamo direktno iz Unity-ja, nazvanu „positionOffset“ koju pridodijelimo Y vrijednosti „transfrom“ metode unutar događaja lijevog klika miša.

### Kupovina utvrda

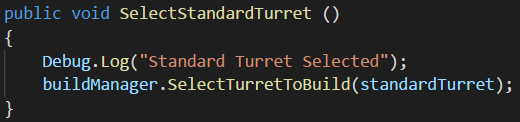
Kupovina utvrda sastoji se komunikacije s korisnikom, obrađeno u poglavlju (7.2) Trgovina utvrda. Ovaj se dio pak odnosi na skripte koje će omogućiti kupovinu.

Da bi omogućili kupovinu stvaramo nove vrste „GameObject-a“ u skripti „BuildManager“ koje povezujemo s novom „Shop“ skriptom. U njoj pozivamo metodu koja dohvaća tijela, odnosno varijable u skripti, te ih prosljeđuje „BuildManager“ skripti. Ta skripta sada zna koje tijelo je potrebno dohvatiti, odnosno koji „prefab“ element, Slika 5‑5.   
Tako možemo izabrati koje tijelo želimo graditi, odnosno osigurati da se razlikuju utvrde pri odabiru za gradnju.



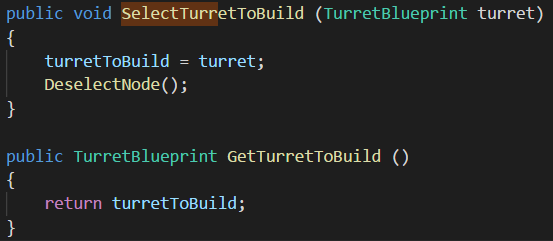
Slika 5‑40 Postavljanje poveznice Shop-BuildManager

Nakon toga ovisno o odluci vrste utvrde, pozivamo gradnju događajem iz „Shop“ skripte. Šalje se vrsta utvrde koja se gradi iz „Shop“ skripte u metodu „SelectTurretToBuild“ iz „BuildManager“ skripte, Slika 5‑6. Postoje 3 ovakve funkcije, svaka šalje informaciju za drugačiju utvrdu.



Slika 5‑41 Shop metoda za referenciranje gradnje

Sama metoda koja postiže izgradnju utvrde pozvana je u „BuildManager“ skripti. Ona na selektiran „Node“ postavlja utvrdu, čija vrsta je poslana iz „Shop“ skripte. Nakon što se obavi gradnja, pozivanjem „turretToBuild“ metode, opisane na Slika 5‑4, prema „TurretBlueprint“ objektu, koji se sastoji od slike, animacija, projektila, uklanja se selekcija tog postolja.   
Kod se može vidjeti na Slika 5‑7.



Slika 5‑42 Pozivanje izgradnje utvrde

Cijena utvrda je postavljena u Unity sučelju, dok provjera da li korisnik (igrač), ima novca nalazi se u „BuildManager“ skripti pomoću „HasMoney“ svojstva, Slika 5‑8, a status njegovog novca nalazi se u skripti „PlayerStats“, gdje se nalaze generalne statistike igrača.

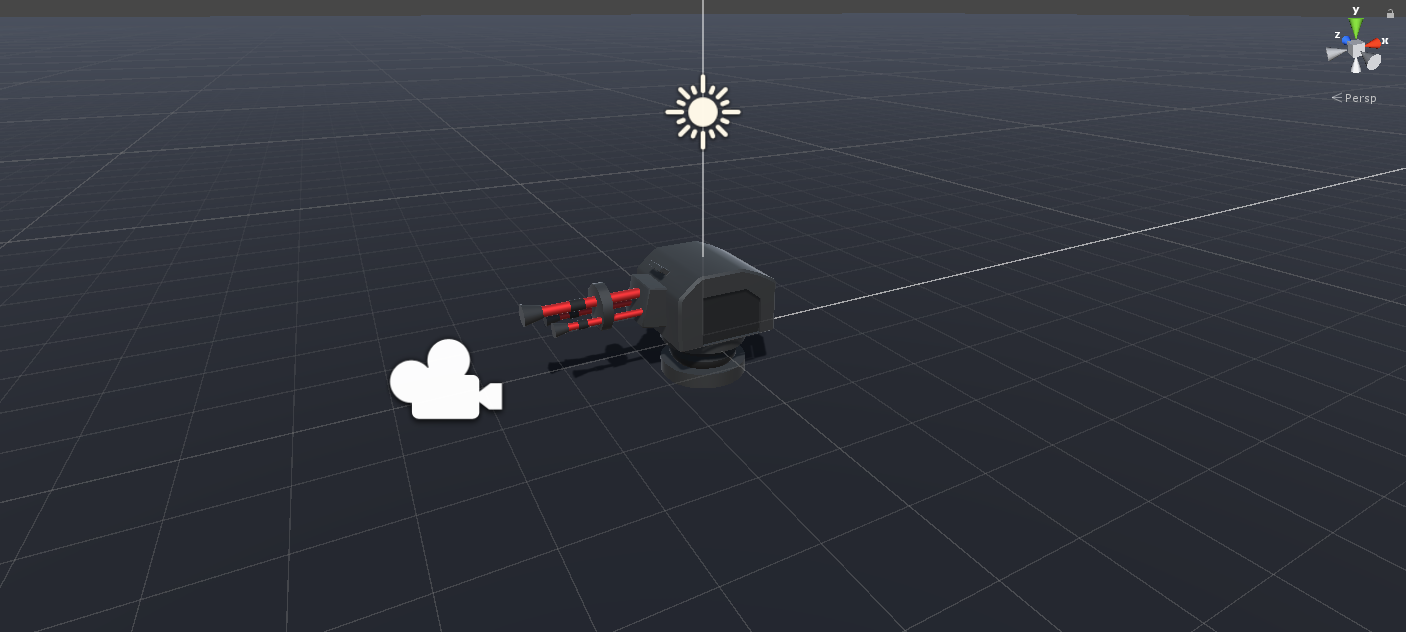


Slika 5‑43 Provjera statusa novca pri kupovini

# Izbornici

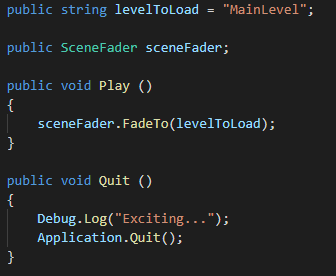
## Glavni izbornik

Glavni izbornik je sučelje koje se otvori pri pokretanju igrice. Za izraditi Glavni izbornik, napravi se nova scena koju ćemo nazvati „MainMenu“ u kojoj su kreirani objekti „Main Camera“ i „Directional Light“. Kreiramo i objekt „GroundPlane“ koji nam služi kao pozadina. Iz glavne scene uzmemo „Prefab“ osnovne utvrde „StandardTurret“ i pozicioniramo je u sredinu scene, Slika 6-1.



Slika 6-1 Objekti glavnog izbornika

Dalje, kreiramo prazni objekt „Canvas“, u koji dodajemo dva nova „Text“ objekta. Jedan od njih će služiti kao „Play“ gumb, a drugi kao „Quit“ gumb. Kreiramo novu datoteku „MainMenu.cs“ u kojoj definiramo funkcionalnosti tih dvaju gumbova, Slika 6-2.



Slika 6-2 Funkcionalnosti Play i Quit gumbova

Finalni izgled glavnog izbornika je jako jednostavan i intuitivan, Slika 6-3.



Slika 6-3 Finalni izgled glavnog izbornika

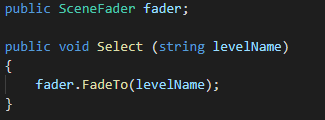
## Izbornik razine

Izbornik razine je sučelje koje nam omogućava biranje različitih razinau igri. Za ostvariti to sučelje prvo izradimo novu scenu koju nazovemo „LevelSelect“. Kao bazu za ovu scenu smo iskoristili scenu „MainMenu“ iz koje smo makli „Prefab“ i gumbove „Play“ i „Quit“. U sceni „LevelSelect“ stvorimo novi objekt koji nazovemo „UICanvas“. „UICanvas“ će sadržavati naslov i gumbove za svaku pojedinu razinu, Slika 6-4.



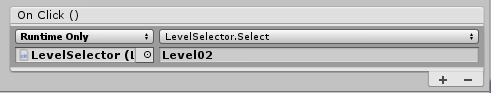
Slika 6-4 Hijerarhija i izgled scene LevelSelect

Funkcionalnost gumbova za odabir razine se ostvaruje u novoj datoteci „LevelSelector.cs“. Da bi taj dio olakšali, iskoristili smo prijelaze izrađene u prethodnom koraku, Slika 6-5.



Slika 6-5 Funkcija prijelaza između razina

Unutar samog Unity-a, za svaki gumb specificiramo koju scenu mora otvoriti prilikom klika na taj gumb, Slika 6-6.



Slika 6-6 „On Click“ događaj za gumb '2'

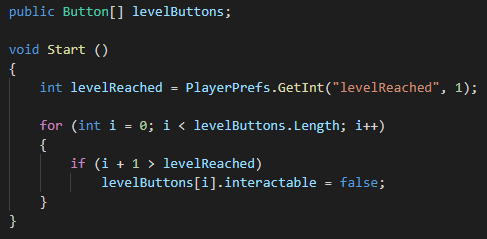
### Otključavanje i pobijeđivanje razina

Idući korak je povezati razine u jednu cjelinu i onemogučiti igraču da igra određenu razinu ako nije svladao prethodnu. To ćemo vizualno postići tako da u „LevelSelect“ sceni zatamnimo gumbove svih razina viših od one koju igrač još nije savladao, Slika 6-7.



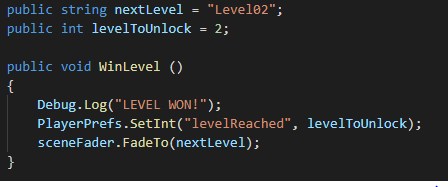
Slika 6-7 Onemogućene razine

Funkcionalno, ovo postižemo tako da nadogradilo skriptu „LevelSelector.cs“, Slika 6-8.



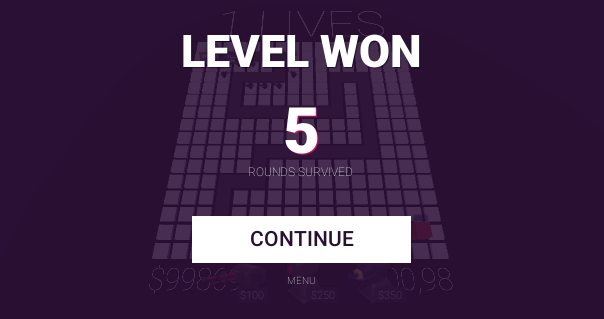
Slika 6-8 Onemogućavanje razina

Za prijelaz između pobijeđene razine i nove razine modificiramo funkciju „WinLevel“ unutar „GameManager.cs“ skripte, Slika 6-9.



Slika 6-9 Prijelaz sa pobijeđene na iduću razinu

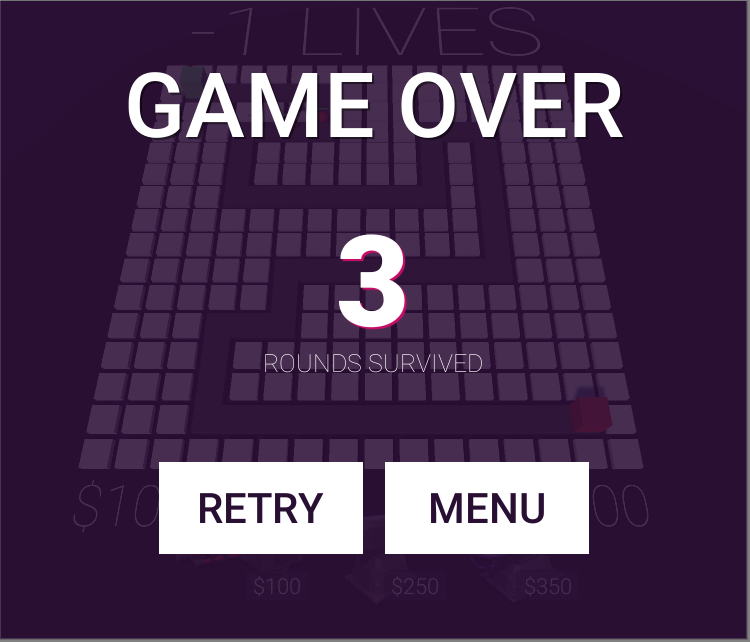
Za kompletiranje procesa prijelaza sa jedne na drugu razinu, izradit ćemo sučelje koje se pojavi prilikom savladavanja razine. Radi jednostavnosti, za izradu ovog sučelja možemo iskoristiti već napravljeno sučelje „GameOver“. Kopirajmo ga i preimenujmo u „CompleteLevel“. Sučelje će imati dva gumba, jedan koji vodi igrača na iduću razinu, i jedan koji gavraća na glavni izbornik, Slika 6-10.



Slika 6-10 Izgled sučelja CompleteLevel

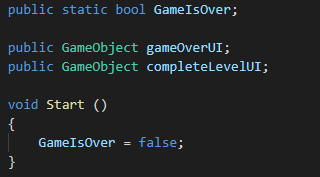
## „Game Over“ izbornik

„Game Over“ izbornik je sučelje koje se prikazuje kada igraču ponestane života. To sučelje ispunjava čitav ekran i prikazuje koliko je valova protivnika igrač uspio pobijediti. Da bi napravili to sučelje, prvo u objektu „OverlayCanvas“ stvorimo novi objekt „GameOver“ tipa „Panel“. Unutar toga ćemo napraviti tri objekta; „Backgroud“, „GameOverText“, „RoundsSurvived“, „Retry“ i „Menu“. „Background“ označava pozadinu ovog sučelja, „Text“ prikazuje tekst na toj pozadini. „RoundsSurvived“ sadrži objekt „Rounds“ (broj preživjelih rundi) i „Text“. „Retry“ i „Menu“ su gumbovi koji resetiraju igru ili nas vraćaju na glavni meni, respektivno. Sveukupni izgled sučelja možemo vidjeti na slici ispod, Slika 6-11.



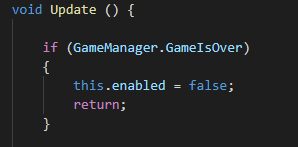
Slika 6-11 Izgled "Game Over" sučelja

Da bi ovo sučelje implementirali, stvaramo novu skriptu „Game Over“ koja je povezana za sami „GameOver“ objekt i onemogućimo prikaz „GameOver“ objekta. Unutar „GameManager.cs“ datoteke dodajemo varijablu „GameIsOver“, Slika 6-12.



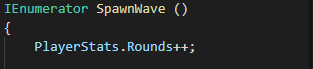
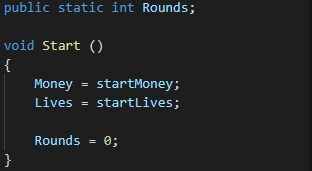
Slika 6-12 Varijabla GameIsOver

Kada igra završi, tj. Kada se prikaže „GameOver“ sučelje, onemogućimo micanje kamere unutar „CameraController.cs“ datoteke, Slika 6-13.



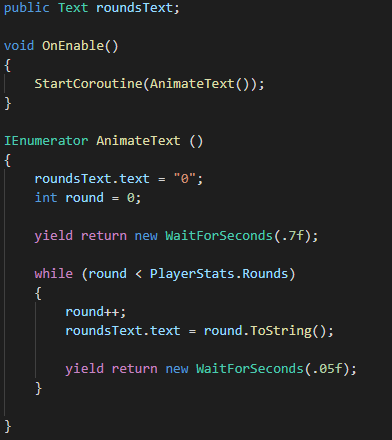
Slika 6-13 Onemogućavanje micanja kamere

Nadalje, moramo omogućiti brojanje valova koje je igrač uspio pobijediti. To radimo inicijalizacijom varijable „Rounds“, i njenim inkrementiranjem prilikom svakog novog vala, Slika 6-14.



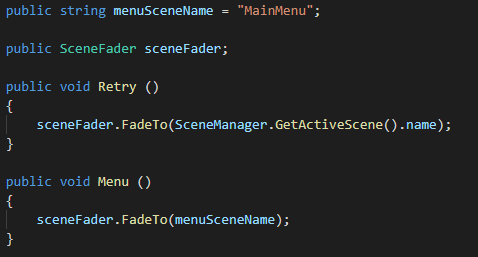
Slika 6-14 Inicijalizacija i inkrementiranje brojača rundi

Na kraju kreiramo novu datoteku „RoundsSurvived.cs“ u kojoj omogućimo prikaz broja rundi na „GameOver“ sučelju povezivanjem varijable „Rounds“ i objekta „Rounds“, Slika 6-15.



Slika 6-15 Prikaz broja pobijeđenih valova

Na kraju ostaje samo omogućiti da rade gumbovi „Retry“ i „Menu“ na „GameOver“ sučelju. Gumb „Retry“ zatvara ovo sučelje i započinje igru ispočetka, a gumb „Menu“ igrača vodi na glavni meni. Kreiramo novu datoteku „GameOver.cs“ u koju postavljamo ponašanje tih dvaju gumbova, Slika 6-16.



Slika 6-16 Ponašanje Retry i Menu gumbova

Ovim postupkom smo postigli „Game Over“ sučelje, i ujedno omogućili igraču da nakon izgubljene igre pokuša ponovno, ili se vrati natrag u glavni meni.

## Izbornik pauze

Izbornik pauze je sučelje koje se otvori kada igrač pauzira igru. Ovo sučelje se otvori pritiskom na tipku „Escape“ ili 'p' i zamrzne trenutna stanja bilo kakvih protivnika, utvrda ili projektila. Sadrži 3 gumba: „Continue“, „Retry“ i „Menu“ te objekt „PausedText“. Za izraditi ovo sučelje možemo iskoristiti već napravljeno sučelje „Game Over“, pošto su ta dva sučelja jako slična, Slika 6-17.



Slika 6-17 Izgled sučelja pauze

Za postizanje funkcionalnosti ovog sučelja izradimo novu datoteku „PauseMenu.cs“, Slika 6-18.



Slika 6-18 Funkcionalnosti sučelja pauze

# Izrada korisničkog sučelja

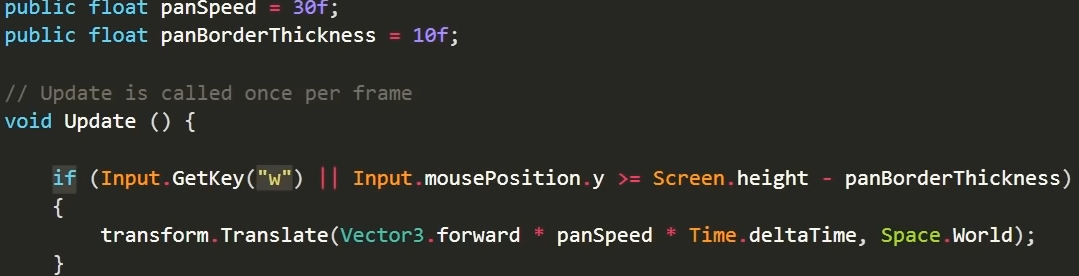
Da bi omogućili korisniku što bolju interakciju s igrom, moramo realizirati korisničko sučelje. Tema izrade sučelja proteže se i kroz prethodna poglavlja, ali tek u ovom se u potpunosti opisuju metode i načini izrade sučelja.

## Kamera

Da bi korisniku omogućili fluidan osjećaj igre, koristeći principe kretanja kamere iz RTS[[12]](#footnote-9) igara.

Za to stvaramo skriptu „CameraControler“ koja upravlja pomakom kamere. Već poznatom metodom „transform“ obavljamo pomicanje po vektoru koji ima oblik vektor3 (0f, 0f, 1f), za pomicanje unaprijed recimo, jer je postavljena vrijednost 1f na Z osi, koja je orijentirana prema naprijed.

S tim da postoje 2 vrste objekata u Unity-u, jedan se uvijek odnosi na početno stanje objekta, koje je pre-kompajlirano stanje za Unity, drugo koje mi definiramo. S obzirom na to da smo mijenjali poziciju kamere, Z os je sada orijentirana prema dolje, te smo testiranjem zaključili da je potrebno koristiti Unity varijablu lokacija, Space.World. Kod na Slika 7‑1.



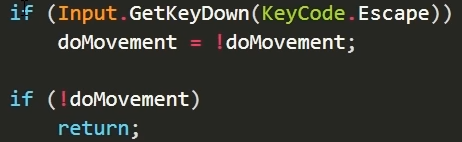
Slika 7‑1 Kod za pomicanje kamere

Osim toga da bi ubrzali kretanje odnosno translaciju, vektor kretanja pomnožimo sa statičnim brojem, po željenoj vrijednosti.

Također kao i prije, da bi optimizirali igru bez obzira na FPS[[13]](#footnote-10), odnosno kvalitetu mašine koja pokreče igru, moramo koristiti „Time.deltaTime“ Unity varijablu.   
Bit će nam potrebna 4 takva „if()“ uvjeta, koja će provjeravati za svaku stranu kretanje.

Da bi korisniku omogućili što poznatiju okolinu kretanje realiziramo koristeći tipke „W, A, S,D“, te također pomakom miša u stranu.  
Kretanje mišem moramo omogućiti samo ako korisnik dolazi na kraj ekrana, jer u protivnom će se ekran kretati uvijek, čak kada korisnik želi samo postaviti utvrdu ili slično.

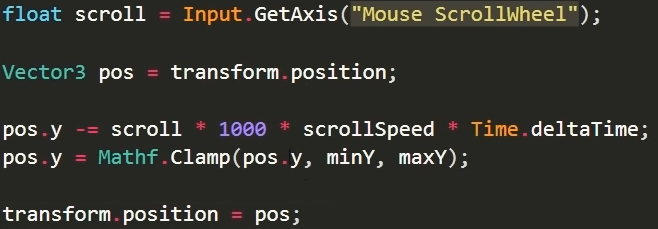
Stoga postavljamo da se kretanje dogodi tek kada je korisnik 10 piksela od dosega do kraja ekrana. Samo kretanje mišem je realizirano preko „Input.mousePosition“.  
Ako korisnik ne želi kretanje, moguće ga je isključiti pomoću tipke „Esc“, te putem iste uključiti ponovno, Slika 7‑2.



Slika 7‑2 Isključivanje kretanja

Isto tako, korisniku je omogućeno uvećavanje koristeći koračić miša, tako što smo pronašli vrijednost koju vrača taj događaj, te s njom smo postavili vrijednost Y pozicije kamere.

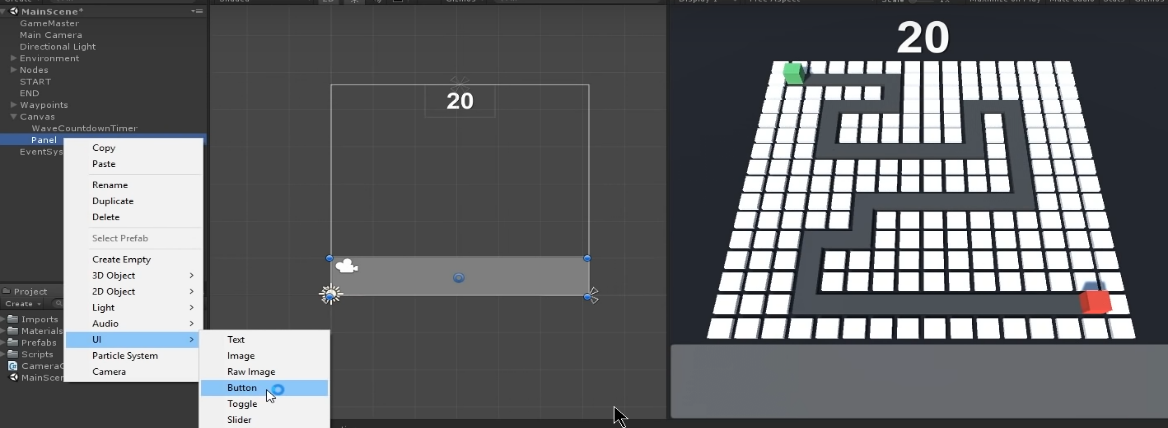
Također limitiramo kretanje u visinu i dubinu, zbog optimizacije same igre, Slika 7‑3.



Slika 7‑3 Interakcija s rotacijom kotačića miša

## Trgovina utvrda

Da bi korisnik mogao koristiti utvrde, mora postojati ugrađeno sučelje kojim će moći vidjeti što može kupiti, te gdje.   
To implementiramo koristeći aktivno područje putem Unity-a, na koje dodajemo roditeljski element dugmeta, u koje instanciramo po jedno dugme za svaku utvrdu koju možemo kupovati, Slika 7‑4.

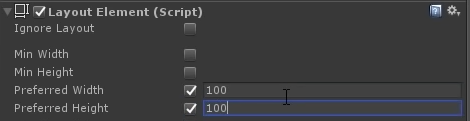


Slika 7‑4 Implementacija sučelja kupovine utvrda

Da bi postavili željenu veličinu svakog dugmeta, postavljamo širinu i visinu na određenu vrijednost kao što je moguće vidjeti iz slika, tako što smo isključili automatsko proširivanje da se popuni prostor, Slika 7‑5, te dodali komponentu „LayoutElement“ u kojoj postavljamo veličinu, te centriramo te elemente, Slika 7‑6.



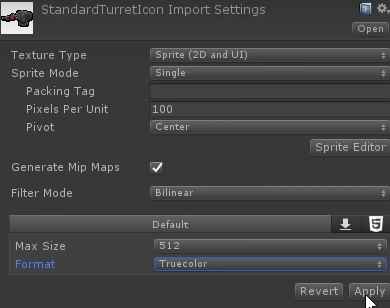
Slika 7‑5 isključivanje automatskog popunjavanja



Slika 7‑6 Postavljanje širine i visine dugmeta

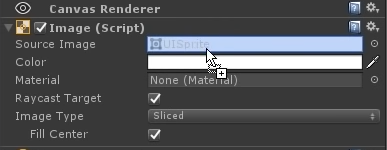
Da bi korisnik imao dojam na što se odnosi koje dugme, umjesto dugmeta stavljamo sliku utvrde. Slike smo obradili u PhotoShop-u, tako da nema pozadinu.

Takvu sliku povučemo u projekt, te joj postavljamo željene vrijednost, Slika 7‑7.



Slika 7‑7 Postavljanje vrijednosti slike

Povezivanje dugmeta i slike obavlja se povlačenjem slike u skriptu slike, vidljivo na Slika 7‑8.  
Također, da bi korisnik osjetio da postoji interakcija, postavljamo promjenu tona boje kada korisnik pređe preko slike utvrde, odnosno kada je klikne.



Slika 7‑8 Ubacivanje slike u skriptu slika

## Sučelje utvrde

Svaka utvrda koju napravimo mora imati mogućnosti nadogradnje i brisanja iste. Da bi to postigli moramo napraviti novo sučelje u koje ćemo staviti te dvije funkcije. Za početak kreiramo utvrdu za referencu. Te na njenoj točnoj lokaciji stvorimo novi objekt „NodeUI“. Unutar toga objekta napravimo objekt „Canvas“ koji nam služi kao pozadina sučelja, te ga pomaknemo iznad utvrde. Unutar toga objekta napravimo dva gumba: „Upgrade“ i „Sell“, Slika 7-9.



Slika 7-9 Izgled sučelja utvrde

Radi jednostavnosti, moramo osigurati da u jednom trenutku može biti selektirana ili utvrda iz trgovine, ili jedna već sagrađena utvrda. To postižemo uvođenjem varijable „selectedNode“ i modificiranjem datoteke „BuildManager.cs“, Slika 7-10, i modificiranjem datoteke „Nodes.cs“, Slika 7-11.

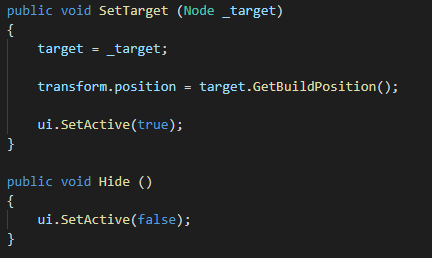


Slika 7-10 Selektiranje utvrde



Slika 7-11 Prioriteti selektiranja i izgradnje utvrde

Dosada nam je sučelje utvrde imalo konstantnu poziciju. Da bi omogućili da se to sučelje pojavi točno iznad trenutno seletirane utvrde, kreiramo novu datoteku „NodeUI.cs“. Pomoću funkcije „GetBuildPosition“ dobivamo točnu lokaciju trenutno selektirane utvrde. Također ćemo omogućiti da se sućelje utvrde sakrije kada nijedna utvrda nije selektirana, ili kada je selektirana utvrda za izgradnju iz trgovine, Slika 7-12.



Slika 7-12 Pomicanje sučelja utvrde

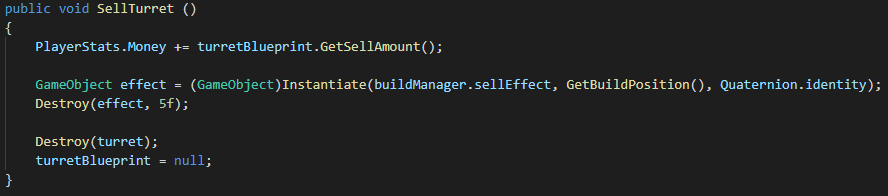
Iduća stvar koju želimo napraviti je osposobiti „Upgrade“ gumb. To postižemo tako što pritiskom na „Upgrade“ gumb jednostavno izbrišemo selektiranu utvrdu, i na njenu poziciju stavimo nadograđenu utvrdu. Dodajemo dvije nove varijable: „upgradedPrefab“ i „upgradeCost“ i funkciju „UpgradeTurret“ koja je samo izmijenjena verzija funkcije „BuildTurret“, Slika 7-13. Ta funkcija se poziva prilikom pritiska gumba „Upgrade“ u sučelju utvrde.



Slika 7-13 Nadogradnja utvrde

Unaprijeđivanjem utvrde se povećava udaljenost na koju može pucati, brzina pucanja, šteta projektila i sami vizualni izgled utvrde. Varijablom „isUpgraded“ se provjerava trenutno stanje nadograđivanja utvrde. Ako je utvrda jednom nadograđeno, ne može se više nadograđivati.

Na kraju, da bi sučelje utvrde bilo kompletno funkcionalno, moramo omogućiti i gumb „Sell“. Dodajemo novu funkciju „SellTurret“ koja se izvršava pritiskom gumba „Sell“ , briše selektiranu utvrdu te vraća dio novca potrošenog na tu utvrdu, Slika 7-14.

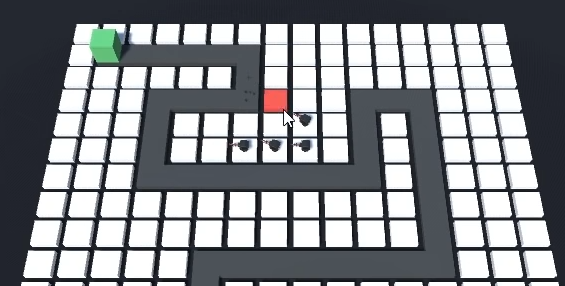


Slika 7-14 Brisanje utvrde

Funkcija „GetSellAmount“ vraća pola novca potrošenog na izgradnju utvrde.

## Interaktivna podloga

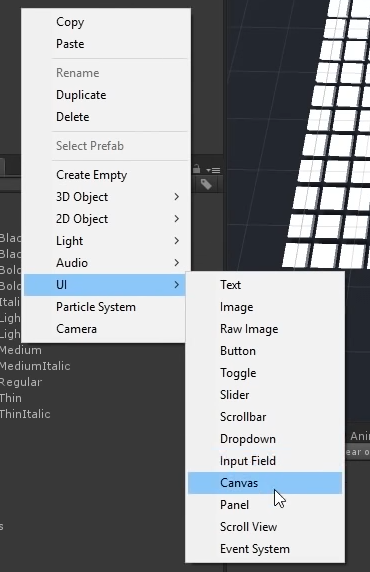
Tema ovog poglavlja obrađena je prethodno u poglavlju (5.1.1) Interakcija s korisnikom, u kojem je dan detaljan opis i referenciran prikaz sa slikama na kod. Dio je obrađen u tom poglavlju jer se nadovezuje na samu izradu utvrda. Stoga ćemo ovdje jedino imati prikaz te interakcije pomoću slike, Slika 7-15. Na slici je vidljivo da korisnik nije u mogućnosti graditi utvrde, ako nema novca, prikazano crvenim poljem nad kojim se nalazi pokazivač miša.



Slika 7‑15 Interakcija s korisnikom kada nema novca za utvrdu

## Statusni tekst

Da bi korisnik mogao vidjeti osnovne informacije poput broja života, novca, vremena, potrebno je napraviti u 2D prostoru novo platno, koje će nositi sve potrebne informacije. Novi element korisničkog sučelja moguće je dodati na način prikazan na Slika 7-16, desnim klikom na području hijerarhije.



Slika 7‑16 Stvaranje novog UI elementa

Sada na tom platnu možemo dodati bilo koji željeni element, primjerice tekstualni element, koji ćemo povezati sa skriptom statusa. Postavljamo to platno na željenu veličinu i na prostor koji želimo. Detaljan opis upravljanja Unity elementima dan je u poglavlju (3.1) Upoznavanje s radnom okolinom.

### Vrijeme

Da bi postavili aktivno vrijeme između rundi korisniku na vidljivo mjesto, stvaramo novi tekstualni element na platnu. To obavljamo desnim klikom na područje hijerarhije, točnije „Canvas“ prostora, te odabirom 2D elementa teksta.   
Tako stvoreni tekst postavljamo na željeno područje, tekst objekt iz hijerarhije potegnemo u „GameMaster“ skriptni objekt, pod „Wave Spawner“ „Wave Countdown Text“ područje.

Tako smo povezali Unity objekt sa skriptom koja se brine o stvaranju neprijatelja, u kojoj se nalazi i podatak o vremenu. Slika 7-17.



Slika 7‑17 Povezivanje skripte i objeta vremena

Sama obrada računanja vremena obavlja se u „GameMaster“ skripti koristeći format vremena dan na slici, Slika 7‑18.

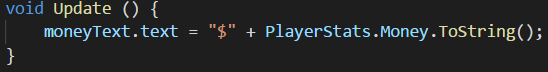


Slika 7‑18 Kod za ispis vremena

### Novac

Na isti način kao i u prethodnom poglavlju izrađujemo Unity element. Razlika je ovaj put što nemamo napravljenu skriptu za upravljanje novcem, stoga za sam novi tekstualni element stvaramo skriptu „MoneyUI“ dodavanjem nove komponente tipa skripte.

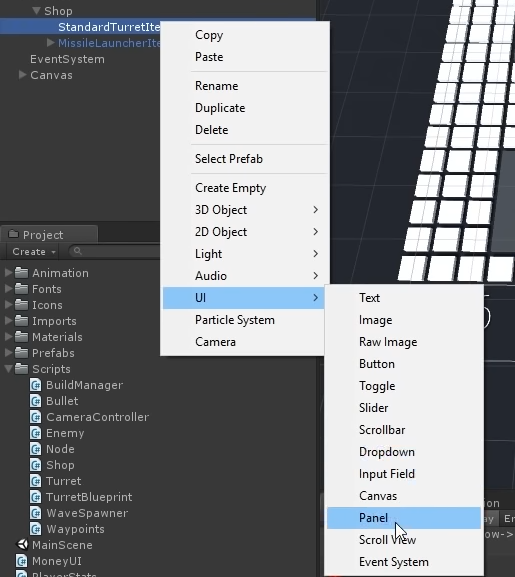
Potrebno je uključiti korištenje „UnityEngine.UI“, koji omogućava korištenje Unity elemenata, te napraviti „Update“ metodu, koja će postaviti vrijednost teksta na string „$“ te nadovezati na njega vrijednost novca, koja se nalazi u varijabli statusne skripte „PlayerStats“, Slika 7-19.



Slika 7‑19 Kod za ispis novca

### Cijena utvrda

Također želimo korisniku prikazati cijenu utvrda na ekranu. Na sličan način kao prije dodat ćemo Unity element. Ovog puta taj element stvaramo u „Canvas-Turret“ datoteci. Slika 7-20.



Slika 7‑20 Izrada Unity natpisa za cijenu utvrde

Takav element sada postavljamo na željeno mjesto, dodajemo mu boju, efekte, postavimo cijenu… Slika 7‑15.

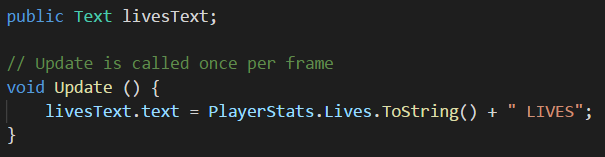


Slika 7‑15 Postavljanje natpisa na željeni prostor

### Životi igrača

Osim toga, korisnika želimo obavijestiti o njegovom broju života. Na isti način kao i u poglavlju (7.4.2), stvaramo novo platno, koje postavljamo na vrh glavnog ekrana, te na identičan način postavljamo željene vrijednosti za poziciju teksta.

Također stvaramo novu skriptu na tom elementu koja će pronaći podatke o broju života iz statusne skripte „PlayerStats“ te ispisati tu vrijednost kao „string“. Slika 7-21.



Slika 7‑21 Kod za ispis života

### Životi objekta protivnika

Na kraj, isto tako je potrebno omogućiti korisniku vizualan prikaz smanjivanja protivničkog života, u protivnom se stvara nelagoda zbog manjka napretka u igri.

Da bi to ostvarili stvaramo novo platno, na isti način kao i prije. Stvaramo ga u hijerarhiji, kao dijete „Enemy“ objekta. Postavljamo tako stvoreno platno u 2D prostornu dimenziju, te njegov atribut „Render-Mode“ postavljamo na „World-Space“ vrijednost.

Nad tako stvorenim platnom sada dodajemo novu sliku, koja će biti pozadinska slika za okvir života protivnika. Ta slika koristit će nam kao crna pozadina koja indicira dio života koji je umanjen.

Nakon toga, tom elementu stvaramo novu sliku koja će biti dijete prethodnog platna. Postavljamo tu sliku na zelenu boju, te ona indicira život protivnika.

Time smo stvorili platno koje indicira kretanje života protivnika. Slika 7-22.



Slika 7‑22 Indikator života protivnika

Kako bi ostvarili prikaz života u realnom vremenu, moramo u skripti protivnika napraviti promjenu života. Stoga u prvom koraku povezujemo napravljeno platno s enemy skriptom, povlačenjem u „Enemy“ objekt.

Samo kretanje ostvareno je pomoću Unity svojstva za popunjavanje slike, koje radi s postotcima. Popunjavanje slike ovisi o trenutnom životu podijeljenim s početnim životom. Kod je vidljiv na Slika 7-23.



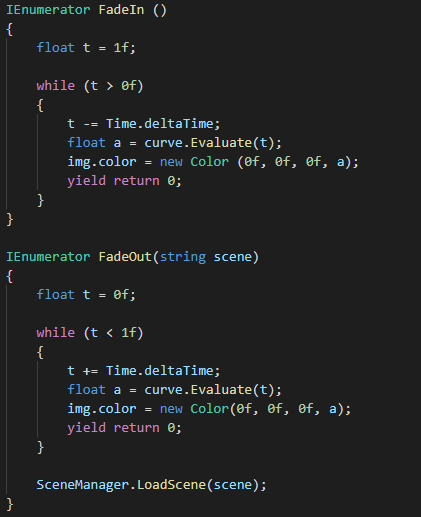
Slika 7‑23 Kod za podešavanje napretka životnog statusa

Ovim je zaključeno poglavlje o korisničkim sučeljima. Kao rezultat ostvarena je interaktivna podloga, koja upozorava igrača o svim mogućim efektima i djelovanjima, te dok ga istovremeno upoznaje s podatcima koji su nužni za samu igru.

## Prijelaz između scena

Prijelaz između scena je jednostavna animacija koja se dogodi prilikom prijelaza iz scene u scenu, npr. prilikom klika na gumb „Play“ u „MainMenu“ sceni, što nas odvodi nascenu „Level01“. To je ostvareno animacijom postepenog zacrnjivanja pozadine „fade out“, i obrnutog postupka „fade in“.

Za početak stvorimo novu datoteku „SceneFader.cs“. U njoj definiramo funkcije „FadeIn“ i „FadeOut“, Slika 7-24.



Slika 7-24 Funkcije FadeIn i FadeOut

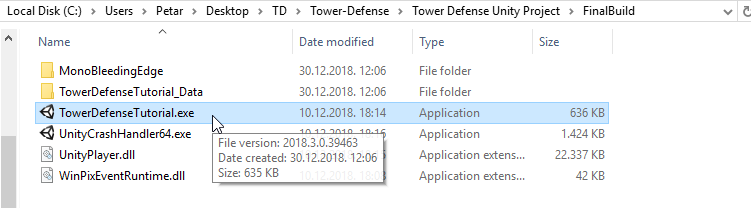
Funkcija „FadeIn“ uzima potpuno crnu sliku, te je kroz određeno vrijeme t izbijeljuje dok u potpunosti ne nestane, dok funkcija „FadeOut“ radi obratno.

# Vodič za igru

Kako bi korisnik što lakše pristupio igri, igra je izgrađena te spremljena na lokaciji koja će ovisiti o mjestu spremanja same igre, no svi dijele istu putanju u nastavku:

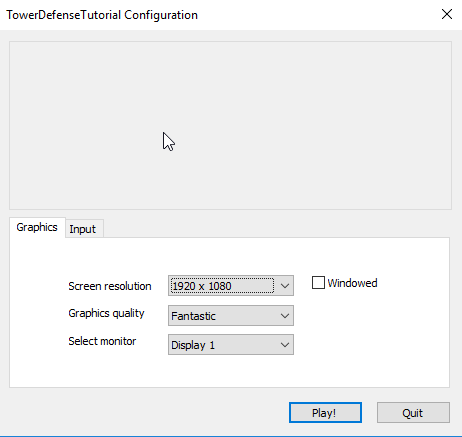
...\FinalBuild

Da bi se igra pokrenula potrebno je pronaći datoteku „TowerDefenseTutorial.exe“, kao što je vidljivo na slici, Slika 8‑1.



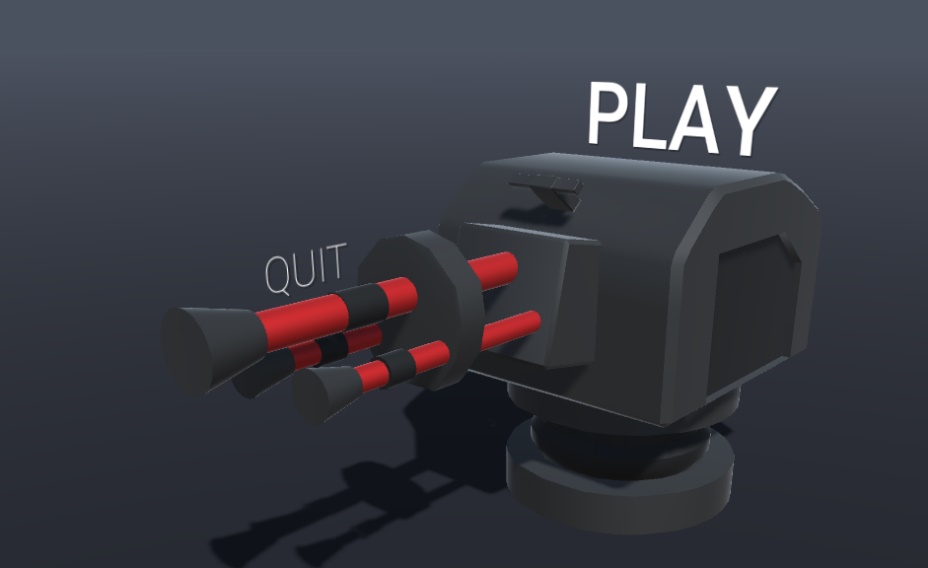
Slika 8‑1 Datoteka za pokretanje igre

Dvostrukim klikom nad tom datotekom otvara se Unity igra. Potrebno je prvotno namjestiti željenu kvalitetu igre, te nakon toga stisnuti „Play!“ dugme. Slika 8‑2.



Slika 8‑2 Pokretanje igre

Nakon što se pokrene „UnityEngine“ otvara se sljedeći prozor, Slika 8‑3. Klikom na „PLAY“ započinje se igra, te se otvara selektiranje nivoa. S obzirom na to da pri prvom ulasku u igru korisnik nije prošao ni jedan nivo, jedini otključan je prvi nivo. Taj i selektiramo.



Slika 8‑3 Početni meni

Tim smo započeli nivo igre, Slika 8‑4.



Slika 8‑4 Prikaz nivoa igre

Za samu igru smo korisniku omogućili neograničena sredstva, koja su vidljiva u lijevom donjem kantunu uz znak $.

Preko puta, s desne strane nalazi se brojač, koji se pokreće između svakog vala protivnika. On indicira vrijeme koje je ostalo korisniku do dolaska drugog vala neprijatelja.

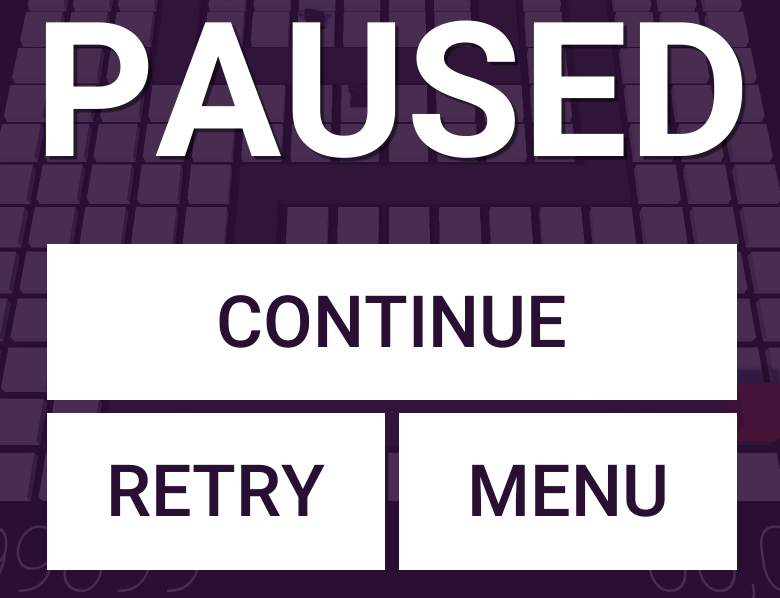
Na donjoj sredini ekrana nalaze se utvrde, koje služe za obranu od protivnika. Klikom na njih selektira se utvrda, a postavljanjem pokazivača miša nad željenim bijelim poljem postavlja se indikator da li je korisnik u mogućnosti izgraditi utvrdu. Ako nije, boja tog polja postaje crvena. Ako je pak moguća gradnja, korisnik klikom na to polje izgrađuje utvrdu. Cijena svake utvrde označena je ispod pripadajuće.

Na gornjem dijelu ekrana prikazani su preostali životi igrača. Ako protivnik dođe do crvene kutijice, igrač gubi jedan ili više života, ovisno o vrsti protivnika.

Protivnici nad sobom imaju indikator života, koji je prikazan zelenom bojom. Jednom kada indikator u potpunosti nestane, postupkom gađanja projektilima utvrda, protivnik se uništava.

Svaki nivo ima određen broj valova te svaki val ima slijed protivnika, a kao posljednji nivo pojavljuju se protivnici koji su znatno teži od prethodnih.  
Pokretanje kamere omogućeno je tipkama „W, A, S, D“ koje pokreću kameru u respektivnim smjerovima: gore, lijevo, dolje, desno. Također moguće je pokretanje kamere mišem, pomicanjem pokazivača u stranu.

Pritiskom tipke „Esc“ pokreće se meni, Slika 8‑5, koji zaustavlja igru te nudi opcije nastavljanja igre, ponavljanja nivoa ili povratka na početni meni, vidljiv na Slika 8‑3.



Slika 8‑5 Prikaz menija

# Literatura

<https://unity3d.com/>

<https://unity3d.com/learn/tutorials>

<https://docs.unity3d.com/ScriptReference/>

<https://docs.microsoft.com/en-us/dotnet/csharp/>

<https://strategywiki.org/wiki/Category:Tower_defense>

<https://www.youtube.com/watch?v=PgDpRMMyxw8&list=PLX-uZVK_0K_4uNwvKian1bscP9mVvOp1M>

# Popis kratica i reference

UE – Unity Engine

VS – Visual studio

FPS – Frame per second

1. https://unity3d.com/ [↑](#endnote-ref-1)
2. https://store.unity.com/?\_ga=2.160024829.1727295570.1545236737-43671070.1544548591 [↑](#endnote-ref-2)
3. https://store.unity.com/download?ref=personal [↑](#endnote-ref-3)
4. Visual Studio [↑](#footnote-ref-1)
5. Montažni elementi [↑](#footnote-ref-2)
6. Omogućuje pristup bilo gdje u projektu [↑](#footnote-ref-3)
7. Omogućuje korištenje bilo gdje u projektu bez referenciranja na skriptu [↑](#footnote-ref-4)
8. Awake funkcija poziva se jednom prije svih drugih stvari pri pokretanju [↑](#footnote-ref-5)
9. Start funkcija poziva se jednom pri pokretanju, nakon awake funkcija [↑](#footnote-ref-6)
10. Update funkcija se poziva pri svakom novom ekranu koji se obradi u Unity sustavu [↑](#footnote-ref-7)
11. Kraj nivoa je ostvaren kada broj protivnika dođe do 0 [↑](#footnote-ref-8)
12. RTS-Real time strategy, žanr video igara [↑](#footnote-ref-9)
13. FPS-Frame per second, okvira po sekundi [↑](#footnote-ref-10)