Izrada "shoot 'em up" igre u programskom alatu Unity

Baukovac, Erik

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Erik Baukovac

IZRADA „SHOOT 'EM UP“ IGRE U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Erik Baukovac
JMBAG: 0016133052
Studij: Informacijski sustavi

Izrada „Shoot 'em up“ igre u programskom alatu Unity

ZAVRŠNI RAD

Mentor:
Doc. dr. sc. Mladen Konecki

Varaždin, srpanj 2021.

Erik Baukovac

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U posljednjih desetak godina, gaming industrija je doživjela veliki zamah po pitanju razvoja programskih alata koji u sebi sadrže programsko sučelje koje omogućava razvoj video igara, animacija i sl. (eng. game engine). Takvi programski alati korisnicima omogućuju uživanje u interaktivnim 2D/3D sadržajima u stvarnom vremenu.

Tema ovog rada je proces izrade „Shoot 'em up“ 2D igre „Stargazer“ u programskom alatu Unity. Unity je razvijen u programskom jeziku C++, a nudi podršku za pisanje programskih skripti kroz vanjsko razvojno okruženje Visual Studio.

Autor rada definira žanr igre, opisuje postupak izrade i objašnjava rad koda, te navodi najznačajnije igre koje su bile inspiracija za izradu 2D igre „Stargazer“. Cilj ove igre je izbjeći neprijateljsku paljbu i tako skupiti što više bodova.

Ključne riječi: Shoot 'em up, Unity, Visual Studio, C#, Stargazer, game development, 2D, video game.

Sadržaj

1. Uvod 1
2. Unity 2
3. Slične igre 3
   3.1. Space Invaders 3
   3.2. Chicken Invaders 4
   3.3. Raptor: Call of Shadows 5
4. Stargazer 7
   4.4. Shoot 'em up 7
   4.5. Funkcionalnosti 7
   4.6. Kod 8
      4.6.1. Izbornik 8
      4.6.2. Glavna scena 9
         4.6.2.1. Spaceship.cs 9
         4.6.2.2. Spawn.cs 13
         4.6.2.3. Enemy.cs 13
         4.6.2.4. BossBehaviour.cs 16
         4.6.2.5. Weapon.cs 18
         4.6.2.6. Bullet.cs 19
         4.6.2.7. Killzone.cs 21
         4.6.2.8. Health.cs 21
         4.6.2.9. ScoreTxt.cs 22
         4.6.2.10. Death.cs 22
5. Zaključak 24
6. Popis literature 25
7. Popis slika 26

1. Uvod

Ubrzanim razvojem tehnologije proporcionalno raste i broj razvijenih video igara. Video igre odgovor su na goruće pitanje kako pobjeći od realnosti, a usput se i dobro zabaviti. Igranje video igre je interakcija jednog ili više sudionika s igrom koja se sastoji od raznih izazova, s ciljem razonode ili zabave. Igre se sastoje od pravila, napora igrača, te ishoda igre koji ovisi o uspjehu igrača. Video igre su specifične po tome što sadrže i digitalnu komponentu, što ih razlikuje od tradicionalnog tipa igara [1].

U današnje vrijeme, možemo primijetiti značajan napredak u procesu razvoja video igara u pogledu kvalitete grafike te samog korisničkog sučelja, pa možemo reći da je proces razvoja video igara na neki način postao umjetnost iza koje stoji mnogo uloženog truda i vremena. S rastom popularnosti igre rastu i očekivanja igrača, što rezultira neprekidnim procesom unaprjeđenja video igre. U ovom procesu programerima posao olakšavaju programski alati koji u sebi sadrže programsko sučelje koje omogućava razvoj video igara, animacija i sl. (eng. game engine). Jedan primjer takvog programskog alata je i programsko okruženje Unity koje je korišteno prilikom izrade „Shoot 'em up“ 2D igre „Stargazer“.

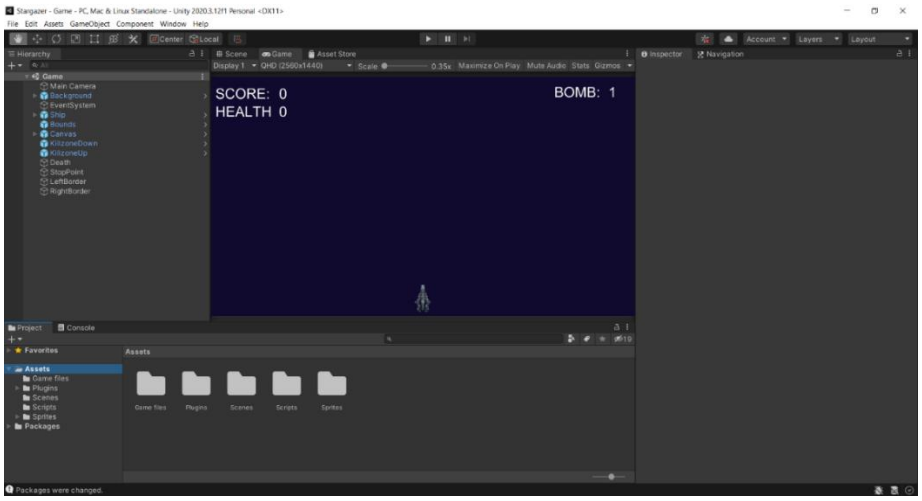
„Shoot 'em up“ je jedan od mnogobrojnih pod žanrova pucačkih igara (eng. shooter video games) [2]. Glavna karakteristika „Shoot 'em up“ igre je da se igrač po razinama bori protiv neprijatelja, a glavno oružje su mu vlastiti refleksi koje koristi kako bi izbjegao neprijateljski napad. Kao inspiracija za izradu računalne igre „Stargazer“ poslužile su mi „Shoot 'em up“ igre poput: Chicken Invaders, Space Invaders i Raptor: Call of Shadows.

U nastavku rada ukratko ću objasniti što je to Unity i za što se koristi, ukratko ću opisati igre koje su mi poslužile kao inspiracija za izradu rada. Za kraj definirat ću žanr računalne igre „Stargazer“ te ću opisati postupak izrade ove igre i objasnit ću rad programskog koda.

2. Unity

Unity je besplatan softver namijenjen razvoju 3D i 2D igara te interaktivnih simulacija za raznovrsne platforme. Prva verzija razvijena je za platformu Mac OS, 6. lipnja 2005. godine pod nazivom Unity 1.0.0, a razvili su je David Helgason, Joachim Ante i Nicholas Francis [3]. Sam Unity pisan je u C++, a nudi podršku za pisanje skripti u programskim jezicima C#, Boo ili JavaScript [3]. Skripte se pišu kroz vanjsko razvojno okruženje Visual Studio koji koristi Mono, open source verziju .NET okvira [4]. Unity3D je sustav objekata baziran na komponentama što znači da se na svaki objekt može vezati skripta [3]. Unity-a ima više dostupnih verzija za preuzimanje. Glavna podjela im je na individualne i timske verzije. Besplatne verzije su „Student“ i „Personal“, a plaćene su „Plus“, „Pro“, i „Enterprise“. Personal verzija koju ja koristim iako je besplatna pruža brojne mogućnosti i ni u jednom slučaju nije ograničavala razvoj igre niti je smanjena kvaliteta. Plaćene verzije imaju mogućnosti cloud dijagnostike, tehničku podršku, personalizirane edukacije i još nekih pogodnosti koje korisnik može birati ovisno o planu [5]. Na slici 1 prikazano je sučelje koje ima jednostavan i intuitivan dizajn koje korisnik dodatno može prilagoditi po svojoj želji.

Programsko sučelje sadrži mnogobrojne prozore: prozor hijerarhije, prozor pogleda, prozor s atributima i projektni prozor. Prozor hijerarhije prikazuje sve objekte u trenutnoj sceni, a nalazi se na lijevoj strani aplikacije. Prozor pogleda se nalazi u sredini aplikacije i njegova uloga je prikaz pogleda trenutno odabrane scene ili simulacije projekta. S desne strane nalazi se prozor s atributima odabranog objekta kojeg možemo uređivati, a na dnu aplikacije nalazi se projektni prozor čija je uloga prikazati popis svih datoteka koje koristimo za izradu projekta.



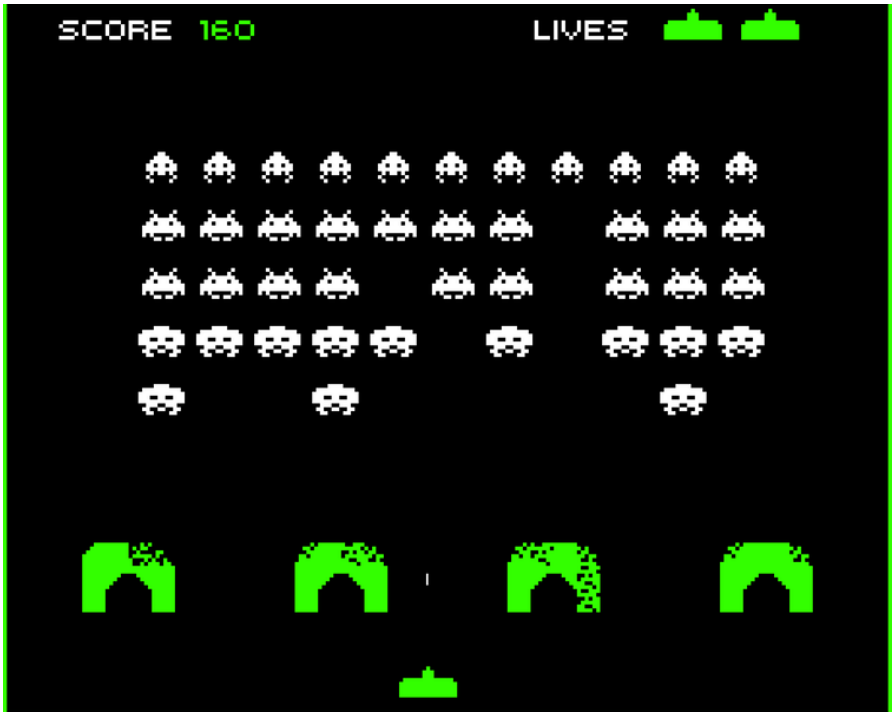
Slika 1. Unity sučelje

3. Slične igre

Inspiracija za kreiranje igre proizašla je iz klasika poput Chicken Invaders, Space Invaders i Raptor: Call of Shadows.

3.1.Space Invaders

Space Invaders je igra za arkade koja je izašla 1978. godine u Japanu [6]. Igra je koncipirana tako da neprijatelji po razinama dolaze u grupama od 55 to jest 5 redova po 11 neprijatelja, a cilj je bio sve ih ubiti i dobiti što više bodova. Povremeno bi se stvorio i poseban brod koji se kretao po vrhu ekrana i davao bi dodatne bodove ako se uništio. Kada korisnik ubije sve neprijatelje pojavljuje se nova grupa neprijatelja, ali se ona svaki put stvara red niže od prethodne grupe. Svaki put kada bi se pojavila nova grupa neprijatelja igra bi ubrzala muziku i neprijatelje. Neprijatelji su se kretali s jednog ruba ekrana prema drugom i svaki put kada bi došlo do promjene smjera spustili bi se za jedan red. Igra bi završila ako bi neprijatelji došli do dna ekrana ili ako bi uništili igrača. Igrač je također imao zaštitne bunkere koji su ga štitili od paljbe, ali su oni imali ograničeni život.



Slika 2. Speće Invaders igra [7]

3.2. Chicken Invaders

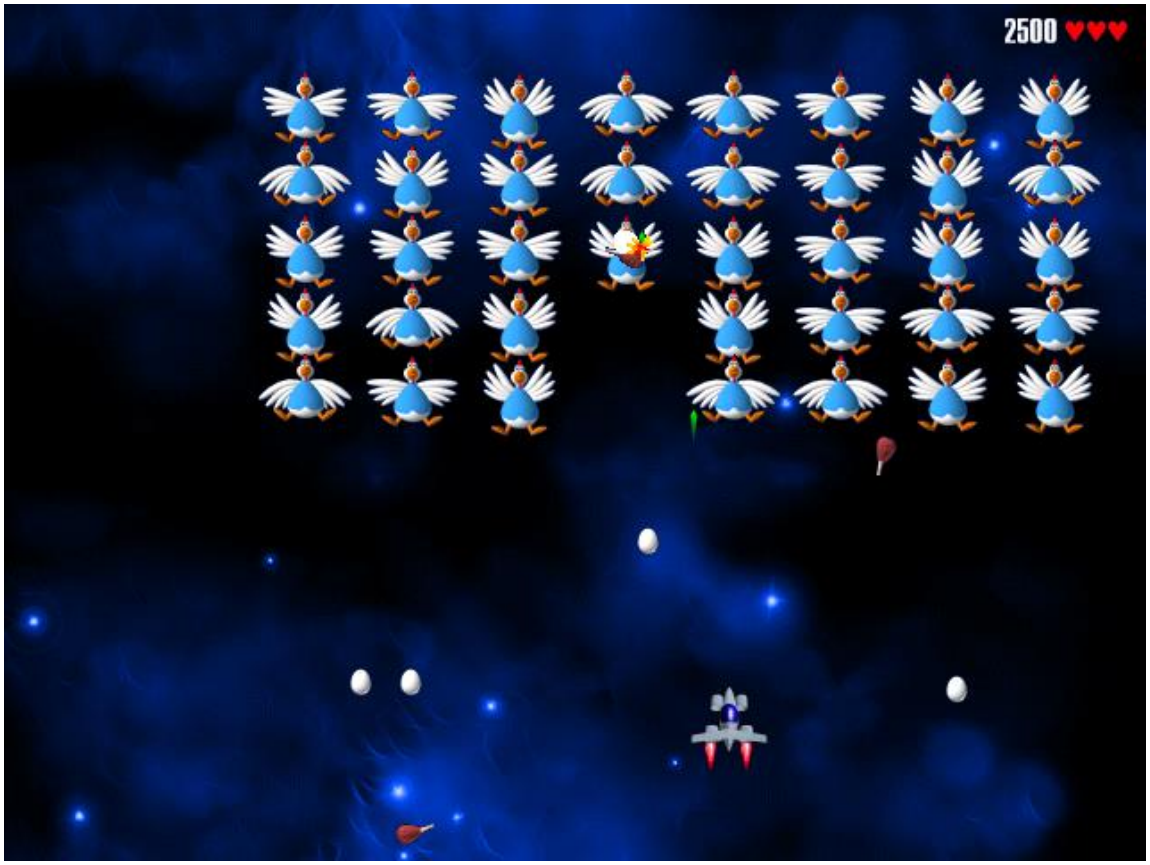
Chicken Invaders je parodija na igru Space Invaders gdje igrač u brodu brani Zemlju od vanzemaljskih jaja, kokoši, pilića i pijetlova. Cilj igre je dobiti što veći rezultat, a kada bi igrač izgubio sve živote to bi označavalo kraj igre. Ova igra ima i mogućnost igranja u dvoje lokalno ili online. Specifičnost igre očituje se u mnogobrojnim mehanikama koje posjeduje kao što su razna oružja, nadogradnje, sateliti, hrana i mnoge druge. Svako oružje ima svoj način pucanja, smjer kretanja metaka i tip neprijatelja na kojem radi veću ili manju štetu. Trenutno najnovija verzija igre ima 15 različitih oružja. Igraču se automatski mijenja oružje kada posjeduje poklon drugačije boje od trenutnog oružja, a pokloni se pseudo nasumično generiraju kada neprijatelji umru. Ovdje ne postoji municija, ali se zato generira toplina koja može dovesti do pregrijavanja oružja.

Nadogradnja oružja je vrlo važna stavka igre i uvelike pomaže prilikom prelaska razina. Nadogradnje funkcioniraju na način da ili povećavaju štetu oružja ili mu daju neke posebne promjene poput povećanja broja metaka ili brzine pucanja. Nadogradnja oružja može se dogoditi u slučaju da igrač pokupi poklon iste boje kao i oružje ili ako pokupi „Atomic Powerup“. Sateliti su dodatna oružja s ograničenom municijom koja se mogu pokupiti i koristiti u borbi protiv neprijatelja. Sateliti su dodani u kasnijim verzijama igre. Bodovi se skupljaju tako što igrač ubija neprijatelje i skuplja hranu koja pada. Hrana daje bodove ovisno o vrsti, a osim za bodove služi za dobivanje specijalnog napada ili dodatnog života ovisno o verziji igre. Specijalni napad ima ograničen broj korištenja i radi veću štetu od običnih napada. U igri postoje i ključevi koji služe za otključavanje posebnih nadogradnji koje imaju različitu svrhu i mogu se birati hoće li se koristiti. Neke nadogradnje samo poboljšavaju UI igre, neke olakšavaju igru ili otežavaju, a neke samo služe vizualnoj promjeni broda.

Postoji mnogo neprijatelja koji se razlikuju po veličini, brzini, napadima, kretanju, životu i otpornosti. Neprijatelji napadaju tako što se zaletavaju u igrača ili pucnjavom, a mogu se pojaviti s bilo koje strane. Oni također posjeduju različita oružja koja koriste protiv igrača.

Što se tiče dizajna, svaka razina je raznovrsna. Postoje razine gdje je potrebno samo ubiti sve neprijatelje koji se kreću ekranom, dok postoje i razine gdje je potrebno preživjeti kišu meteora, obraniti Zemlju, pobijediti glavnog neprijatelja. Osim običnih razina postoje bonus razine koje služe kao prilika igraču da dobije nadogradnju oružja. Glavni neprijatelj se pojavljuje nakon određenog broja razina i njega je puno teže ubiti jer ima posebne vrste kretanja i napada koji su ponekad nepredvidivi. Zanimljivo je da igra ima ugrađene načine varanja koje igrači mogu koristiti u svoju korist, ali u većini verzija ta mogućnost je uklonjena kako bi se sačuvao

smisao sustava bodovanja i ljestvica. Samo prva verzija Chicken Invadersa je bila neograničena to jest nije imala fiksan broj razina dok sve ostale imaju.



Slika 3. Chicken Invaders v1.3

3.3.Raptor: Call of Shadows

Raptor: Call of Shadows također je shoot 'em up igra, ali se dosta razlikuje od dvije prethodno spomenute igre. U ovoj igri neprijatelji ne dolaze grupno i nadogradnje se kupuju tek kada se pobjedi razina. U igri postoje četiri različita lika od kojih igrač može odabrati jednog. Oružja se dijele na primarna i sekundarna oružja te oružje specijalnog napada. Primarna oružja su oružja koja igrač ima konstantno odabrana i njih ne može mijenjati tijekom razine, a sekundarna oružja može mijenjati i prilagoditi situaciji u kojoj se nalazi. Igrač ima poseban napad koji ubija sve neprijatelje osim glavnih, a odjednom može imati pet napada koje sakupi ili kupi. Sustav života ove igre drugačiji je od ostalih igara jer igrač ima određenu štetu koju može primiti prije nego izgubi život što znači da igrač može samo jednom umrijeti. Kroz nadogradnje može kupiti štitove koji mu dodatno povećavaju život i jedan od tih štitova se

oporavlja, ali jako sporo i samo dok igrač ne puca. Igrač je prošao igru ako je pobijedio sva tri sektora s devet razina, dok je gubitak života označavao kraj igre. Ako igrač izgubi život može ponovno učitati zadnji spremljeni napredak jer igra ima sustav spremanja napretka tako da nije potrebno sve ispočetka rješavati. Kada igrač jednom prođe sektor može ga ponovno igrati, ali se težina automatski povećava. Neprijatelji su osim aviona još i stacionarni tornjevi, vozila i brodovi koji pucaju u smjeru igrača.

Slika 4. Raptor v2.4

4. Stargazer

Stargazer pripada akcijskom žanru video igara „Shoot 'em up“. Ovo poglavlje započinje opisom „Shoot 'em up“ žanra video igara, u kojem su opisane neke od glavnih karakteristika i razlika koje igre ovog žanra imaju. U sljedećem potpoglavlju je detaljno opisan kod završnog rada uz popratne slike koje prikazuju kako izgleda korisničko sučelje igre. Na kraju ovog poglavlja opisane su igre koje su poslužile kao inspiracija za izradu ovog završnog rada.

4.4. Shoot 'em up

Shoot 'em up je jedan od brojnih pod žanrova pucačkih igara [2]. Glavna karakteristika shoot 'em up igre je da se igrač po razinama bori protiv velikog broja neprijatelja, a glavno oružje su mu refleksi koje koristi kako bi izbjegao neprijateljsku paljbu [8]. Za ovaj žanr video igara karakteristično je da perspektiva igrača bude s gornje ili bočne strane glavnog lika. Igre ovog žanra većinom se razlikuju po cilju igre, neprijateljima i alatima kojima igrač raspolaže. Cilj igre ovog žanra može biti: uspješno prijeđeni određeni broj razina na kojima su glavni neprijatelji ili što veći broj skupljenih bodova tijekom jedne igre. Neprijatelji mogu napadati grupno ili pojedinačno, mogu biti stacionarni na jednom mjestu ili se kretati po pred definiranim putanjama. Uz to oni mogu posjedovati različite vrste oružja i tako na različite načine nanositi štetu igraču. Neke igre ovog tipa pružaju samo jedno oružje kojim raspolaže igrač dok druge pružaju veći broj različitih tipova oružja. Također neke igre pružaju mogućnosti nadograđivanja oružja, brodova ili opreme kako bi igraču olakšali prelazak razina igre, a time i uspješno rješavanje igre.

4.5. Funkcionalnosti

Stargazer je shoot 'em up igra pa je dizajnirana kao klasična igra tog žanra. Na početku igre igrač ima tri života. Kod sudara s neprijateljima ili ako je upucan gubi jedan život, a dobiva život skupljanjem specijalne baterije koju neprijatelji mogu stvoriti. Igrač se može kretati pomoću strelica ili tipki WSAD, a puca pomoću tipke Space. Ovisno o oružju način pucanja je drugačiji. Num 1 tipka odabire prvo oružje koje puca ravno jedan metak i nema ograničen broj municije. Num 2 tipka odabire sačmaricu koja puca tri projektila ravno i municija je ograničena ovisno o tome koliko je igrač skupio municije. Num 3 odabire zadnju vrstu oružja, a to je mina. Mine su stacionarne i gdje ih igrač postavi one ostaju dok neprijatelj ne naleti na njih. Mine su također ograničene municijom. Igrač ima jedan specijalan napad po razini i on se koristi

pritiskom na tipku Num Enter. Specijalan napad ubija sve neprijatelje osim glavnog kojemu nanosi 10 štete. Igrač je slobodan kretati se po cijelom ekranu, ali van ekrana ne može. Dok neprijatelji mogu i van ekrana. Neprijatelji se pojavljuju nasumično i ne dolaze grupno već pojedinačno.

4.6.Kod

Stargazer je projekt koji se sastoji od 4 scene, 11 skripata i 18 objekata koji su svi preuzeti s Unity Asset Store. Igra započinje prikazom scene izbornika (slika 2). Izbornik nudi dvije opcije, a to su opcije: Play koja pokreće igru i učitava sljedeću scenu igre i Quit koja zatvara aplikaciju.

Slika 5. Izbornik

4.6.1.Izbornik

Izbornik koristi MainMenu.cs skriptu koja u sebi ima 3 funkcije: PlayGame, LoadScene i QuitGame. PlayGame je funkcija koja se pokreće klikom na gumb Play. Kada se pozove funkcija PlayGame, rezultat se postavlja na 0, a zatim se poziva funkcija LoadScene klase SceneManager koja učitava sljedeću scenu pod nazivom Game. Funkcija QuitGame pokreće se klikom na gumb Quit, i zatvara aplikaciju pomoću Quit funkcije klase Application. Funkcija LoadScene učitava scenu po parametru koji je proslijeđen.

```
public void PlayGame()
    {
        PlayerPrefs.SetInt("Score", 0);
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex +
1);
    }
    public void QuitGame()
    {
        Application.Quit();
    }
    public void LoadScene(int a)
    {
        SceneManager.LoadScene(a);
    }
```

4.6.2. Glavna scena

Glavna scena koristi sve preostale skripte pomoću kojih se upravlja radom igrača, neprijatelja, metaka i samog prostora igre. Na slici 3 vidimo ekran tijekom igre. U gornjem lijevom kutu se nalazi život igrača i trenutno stanje bodova, a u desnom kutu broj specijalnih napada. Ostatak ekrana rezerviran je za kretanje neprijatelja i igrača. Van ekrana postoji granica koja zaustavlja igrača. Također van ekrana postoji i zona uništenja kako bi se svi objekti sigurno uništili ona je pomaknuta od granice kako se igrač ne bi uništio pri dodiru ruba i kako bi se neprijatelji mogli instancirati van ekrana da dobijemo efekt dolaska.

Slika 6. Glavna scena

4.6.2.1. Spaceship.cs

Namjena ove klase je upravljanje broda igrača i ostalim stavrima koje su vezane uz njega.

```
int delay = 0, health = 3, bomb=1, selectedWeapon=0;
GameObject bulletSpawn;
int[] ammo = {100,0,0};
Rigidbody2D body;
public GameObject explosion;
public GameObject[] bullet;
public float speed;
```

Na početku su deklarirane varijable. Uloga varijable delay je upravljanje pucanjem, te je s njom osigurano da igrač ne može prebrzo pucati. health je varijabla koja se koristi kao brojač života. Bomb je varijabla za specijalno oružje i selectedWeapon je varijabla kojom se provjerava koje oružje igrač trenutno koristi. bulletSpawn je varijabla tipa GameObject koja služi kao točka za instanciranje metka. ammo je niz s brojem municije za svako oružje. body služi za upravljanje tijelom broda. explosion je objekt koji prosljeđujemo pri smrti igrača kako bi instancirali efekt eksplozije. bullet je varijabla za instanciranje metaka različitih oružja i speed je varijabla za upravljanje brzinom broda.

```
void Awake()
{
    body = GetComponent<Rigidbody2D>();
    bulletSpawn = transform.Find("bulletSpawn").gameObject;
}
```

Awake je ugrađena funkcija koja se poziva kod učitavanja skripte. U ovoj funkciji dodijeljena je vrijednost varijabli body koja je Rigidbody2D komponenta broda i vrijednost varijable bulletSpawn se postavlja na trenutnoj lokaciji oružja broda.

```
private void Start()
{
    PlayerPrefs.SetInt("Health", health);
    PlayerPrefs.SetInt("Bomb", bomb);
}
```

Start je ugrađena funkcija koja se poziva prije prvog poziva varijable Update, to jest čim se skripta učita. Ova funkcija služi za postavljanje broja života i govori nam koliko puta se može iskoristiti specijalan napad u varijable koje služe za ispis na ekran i korištenje na drugim scenama.

```
void Update()
{
    ammo[0] = 100;
    body.AddForce(new Vector2(Input.GetAxis("Horizontal") * speed, 0));
    body.AddForce(new Vector2(0, Input.GetAxis("Vertical") * speed));

    if (Input.GetKey(KeyCode.Keypad1))
        selectedWeapon = 0;
    if (Input.GetKey(KeyCode.Keypad2) && ammo[1] > 0)
        selectedWeapon = 1;
    if (Input.GetKey(KeyCode.Keypad3) && ammo[2] > 0)
        selectedWeapon = 2;
```

```
        if (selectedWeapon != 0 && ammo[selectedWeapon] == 0)
            selectedWeapon = 0;

        if (Input.GetKey(KeyCode.Space) && delay > 120)
            Shoot();
        delay++;

        if (Input.GetKey(KeyCode.KeypadEnter) && bomb==1)
            NukeEm();
}
```

Update je ugrađena funkcija koja se poziva svaki okvir. Funkcija osigurava da primarno oružje ne ostane bez municije, tako što za svaki okvir postavlja vrijednost municije na 100. Zatim dodaje silu kretanja broda po osi unosa korisnika i množi ju sa proslijeđenom brzinom. Ako korisnik želi promijeniti oružje, provjerava ima li municije u nizu ammo za to oružje i mijenja odabrano oružje. Funkcija kontrolira pucanje tako što provjerava delay i specijalne napade ako je vrijednost varijable bomba postavljena na 1.

```
    void Shoot()
    {
        delay = 0;
        if (selectedWeapon==1)
        {
            Instantiate(bullet[selectedWeapon],
bulletSpawn.transform.position, Quaternion.identity);
            Instantiate(bullet[selectedWeapon],
bulletSpawn.transform.position + new Vector3(0.3f,0), Quaternion.identity);
            Instantiate(bullet[selectedWeapon],
bulletSpawn.transform.position + new Vector3(-0.3f,0),
Quaternion.identity);
        }
        else
            Instantiate(bullet[selectedWeapon],
bulletSpawn.transform.position, Quaternion.identity);

        ammo[selectedWeapon]--;
    }
```

Shoot je funkcija koja upravlja pucanjem. Prvo postavlja vrijednost varijable delay na 0 kako korisnik ne bi mogao pucati ponovno, a zatim provjerava o kojem se oružju radi te instancira metak u prostoru kako bi ga njegova skripta mogla kontrolirati. Na kraju funkcija smanjuje broj municije za oružje.

```
    void NukeEm()
    {
        bomb = 0;
        PlayerPrefs.SetInt("Bomb", 0);
        var existingEnemies = FindObjectsOfType<Enemy>();
        var boss = FindObjectOfType<BossBehaviour>();
        foreach (var enemy in existingEnemies)
        {
            enemy.Damage(10);
        }
```

```
        if (boss!=null)
        {
            boss.Damage(10);
        }

    }
```

NukeEm je funkcija koja upravlja specijalanim napadom. Prvo postavlja vrijednost varijable bomb na 0, tako da se ne može ponovno iskoristiti napad, a zatim dohvaća sve neprijatelje na ekranu i nanosi im štetu.

```
    public void Damage()
    {
        health--;
        PlayerPrefs.SetInt("Health", health);
        StartCoroutine(Blink());
        if (health==0)
        {
            Instantiate(explosion, transform.position,
Quaternion.identity);
            Destroy(gameObject,0.1f);
        }
    }
```

Damage je funkcija koja služi za indikaciju štete koju je igrač primio. Prvo smanjuje vrijednost varijable života, a zatim postavlja vrijednost varijable ekrana na vrijednost smanjenog života. Nakon toga se poziva korutina Blink i ako je vrijednost varijable život na 0 poziva se eksplozija i uništi se brod.

```
    IEnumerator Blink()
    {
        GetComponent<SpriteRenderer>().color = new Color(1, 0, 0);
        yield return new WaitForSeconds(0.1f);
        GetComponent<SpriteRenderer>().color = new Color(1, 1, 1);
    }
```

Ova korutina služi za simuliranje štete broda gdje se mijenja boja broda jedan okvir u crvenu i zatim se vraća u normalnu.

```
    public void AddHealth()
    {
        health++;
        PlayerPrefs.SetInt("Health", health);
    }
    public void AddWeapon(int number)
    {
        ammo[number] += 10;
    }
```

Funkcije AddHealth i AddWeapon povećavaju život i municiju igrača, a uz to Addhealth povećava i vrijednost varijable ispisa na ekran.

4.6.2.2.Spawn.cs

```
int score;
public float rate;
bool bossSpawned=false;
public GameObject[] enemies;
public GameObject boss;
```

U ovoj skripti korištene su varijable: score, rate, bossSpawned, enemies i boss. Varijabla score prati trenutno stanje bodova, dok je rate varijabla brzine stvaranja neprijatelja. Varijabla bossSpawned je indikator kraja razine, dok je varijabla enemies niz koji sadrži moguće neprijatelje, a boss je objekt glavnog neprijatelja.

```
void Start()
{
    InvokeRepeating("SpawnEnemy", rate, rate);
}
```

Ova skripta prilikom učitavanja poziva funkciju SpawnEnemy, kako bi se izvršavala neprekidno svakih rate sekundi.

```
void SpawnEnemy()
{
    score = PlayerPrefs.GetInt("Score");

    if (score>100 && !bossSpawned)
    {
        bossSpawned = true;
        CancelInvoke("SpawnEnemy");
        Instantiate(boss, new Vector3(0, 6, 0), Quaternion.identity);
    }
    else
    {
        Instantiate(enemies[Random.Range(0, enemies.Length)], new
Vector3(Random.Range(-8, 8), 6, 0), Quaternion.identity);
    }

}
```

SpawnEnemy dohvaća trenutni broj bodova i sprema ih u varijablu score. Nakon što dohvati broj bodova provjerava trenutno stanje i ako je broj bodova iznad 100 instancira glavnog neprijatelja i zaustavlja ponovno pokretanje funkcije SpawnEnemy. U suprotnom, pseudo nasumično instancira neprijatelje iz niza enemies.

4.6.2.3.Enemy.cs

```
public GameObject bullet,explosion,battery;
public GameObject[] weapons;
Rigidbody2D body;
public Color bulletColor;
public float xSpeed, ySpeed, fireRate, health;
public int score;
```

```
public Transform player;
```

Skripta Enemy.cs koristi varijable: bullet kako bi instancirala metak kada neprijatelj puca; explosion koja je indikator smrti neprijatelja; battery koja služi kao život za igrača; weapons za moguća oružja koja može stvoriti pri smrti; body za upravljanje RigidBody2D komponentom; bulletColor da bi promjenili boju metka; fireRate kako bi prikazali brzinu pucanja i može li igrač uopće pucati; health za prikaz stanja života; score kako bi prosljedio bodove kada umre; player za dohvaćanje koordinata neprijatelja, te xSpeed i ySpeed za upravljanje brzinom kretanja neprijatelja.

```
void Awake()
{
    body = GetComponent<Rigidbody2D>();
}
```

Awake je ugrađena funkcija koja se poziva prilikom učitavanja skripte. U ovoj funkciji dodijeljena je vrijednost varijabli body koja je Rigidbody2D komponenta neprijatelja.

```
void Start()
{
    if (fireRate>0)
        InvokeRepeating("Shoot", fireRate, fireRate);
}
```

Skripta prilikom učitavanja, ako neprijatelj može pucati, poziva funkciju Shoot kako bi se izvršavala neprekidno svakih fireRate sekundi.

```
void Update()
{
    body.velocity = new Vector2(xSpeed, ySpeed * -1);
}
```

Funkcija Update svaki okvir postavlja brzinu neprijatelja na brzinu koja je proslijeđena.

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag=="Player")
    {
        collision.gameObject.GetComponent<Spaceship>().Damage();
        Die();
    }
}
```

Skripta prilikom sudara provjerava je li se neprijatelj sudario s igračem i ako je poziva Damage funkciju za igrača, a neprijatelj umire.

```
void Die()
{
    SpawnDrop();
    Instantiate(explosion, transform.position, Quaternion.identity);
```

```
        PlayerPrefs.SetInt("Score", PlayerPrefs.GetInt("Score") + score);
        Destroy(gameObject);
    }
```

Funkcija Die poziva funkciju SpawnDrop zatim instancira eksploziju i povećava broj bodova igre nakon čega uništava neprijatelja.

```
    private void SpawnDrop()
    {
        if (Random.Range(0, 10) == 0)
        {
            if (Random.Range(0, 5) == 0)
                Instantiate(battery, transform.position,
Quaternion.identity);
            else
                Instantiate(weapons[Random.Range(0, weapons.Length)],
transform.position, Quaternion.identity);

        }
    }
```

Funkcija SpawnDrop pseudo nasumično generira oružje, život ili ništa za igrača da pokupi nakon što neprijatelj umre.

```
    public void Damage()
    {
        health--;
        if (health == 0)
            Die();
    }

    public void Damage(int d)
    {
        health-=d;
        if (health <= 0)
            Die();
    }
```

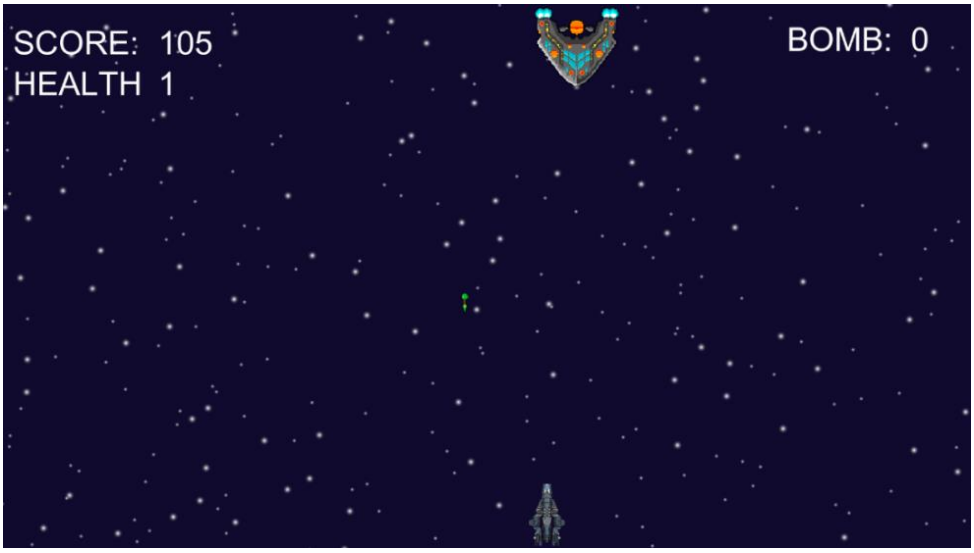
Funkcija Damage ima dvije deklaracije. U oba slučaja smanjuje život neprijatelja i ako je njegov život 0 onda ga uništava. Jedina razlika je što kod preopterećene funkcije ne smanjuje život za 1, već za proslijeđeni broj. Preopterećena funkcija se koristi za specijalni napad gdje se nanosi velika šteta neprijateljima.

```
    void Shoot()
    {
        GameObject temp = (GameObject)Instantiate(bullet,
transform.position, Quaternion.identity);
        player = FindObjectOfType<Spaceship>().transform;
        Bullet metak = temp.GetComponent<Bullet>();

        metak.ChangeDirection();
        metak.ChangeColor(bulletColor);
        metak.SetAtr(player, fireRate);
    }
```

Funkcija Shoot služi za pucanje. Prvo instancira metak na poziciji neprijateljskog oružja. Zatim dohvaća koordinate igrača kako bi mogli usmjeriti metak prema njemu i nakon toga postavlja atribute poput boje i smjera metka pomoću klase Bullet.

4.6.2.4.BossBehaviour.cs



Slika 7. Glavni neprijatelj

Na slici 4 možemo vidjeti glavnog neprijatelja koji se nakon zaustavljanja počeo kretati udesno. Od trenutka instanciranja do točke zaustavljanja njegovo kretanje je predefinirano kao pravocrtno kretanje prema dolje, a nakon toga, smjer kretanja mu je s lijeva na desno ili s desna na lijevo.

```
    public GameObject bullet, explosion, battery;
    Rigidbody2D body;
    public Color bulletColor;
    public float xSpeed, ySpeed, fireRate, health;
    public int score;
    bool stop = false;
    bool movRight = true;
    public Transform player;
```

Klasa BossBehaviour sadrži iste funkcije kao i klasa Enemy.cs pa su ovdje opisane samo one koje su različite.

```
void Update()
    {
        if (!stop)
        {
            body.velocity = new Vector2(xSpeed, ySpeed * -1);
        }
        if (!movRight && stop)
        {
```

```
            body.velocity = new Vector2(ySpeed * -1, 0);
        }
        if(movRight && stop)
        {
            body.velocity = new Vector2(ySpeed, 0);
        }
}
```

Kod svake promjene okvira funkcija Update provjerava tri uvjeta. Prvi uvjet je kod njegovog stvaranja gdje on dolazi na ekran i mora se zaustaviti na određenoj točci. Drugi i treći uvjeti su uvjeti za provjeru kretanja, u kojima se provjerava treba li krenuti u lijevo ili u desno.

```
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "StoppingPoint" && !stop)
        {
            InvokeRepeating("Shoot", fireRate, fireRate);
            body.velocity = new Vector2(ySpeed, 0);
            stop = true;
        }
        if (collision.gameObject.tag == "LeftBorder")
        {
            changeDirection();
            movRight = true;
        }
        if (collision.gameObject.tag == "RightBorder")
        {
            changeDirection();
            movRight = false;
        }

    }
```

Prilikom sudara skripta provjerava je li glavni neprijatelj dotaknuo jednu od tri točke koje su namijenjene da on promjeni smjer kretanja. Prva točka je točka zaustavljanja, gdje je osigurano da se neprijatelj ne približi igraču, a druge dvije su odgovorne da ne izađe iz prozora.

```
    private void changeDirection()
    {
        body.velocity *= -1;
    }
```

Ova funkcija služi za promjenu smjera tako što množi trenutni vektor s -1. Pošto trenutni vektor ima brzinu po x osi 0, mijenja se samo kretanje po y osi.

```
    void Die()
    {
        if (Random.Range(0, 5) == 0)
            Instantiate(battery, transform.position, Quaternion.identity);

        Instantiate(explosion, transform.position, Quaternion.identity);
        PlayerPrefs.SetInt("Score", PlayerPrefs.GetInt("Score") + score);
        Destroy(gameObject);
        SceneManager.LoadScene(3);
```

```
    }
```

Funkcija Die kao i kod skripte Enemy.cs instancira objekte za pokupiti i povećava rezultat, ali i postavlja trenutnu scenu na scenu za pobjedu koja je prikazana na slici 5.

Slika 8. Pobjeda

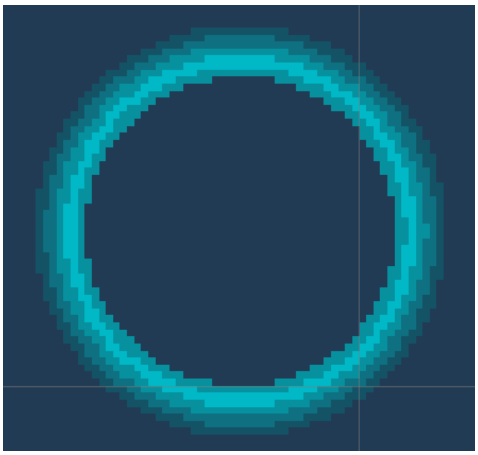
4.6.2.5.Weapon.cs

```
    public int weaponNumber;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Player")
        {

collision.gameObject.GetComponent<Spaceship>().AddWeapon(weaponNumber);
            Destroy(gameObject);
        }
    }
```

Skripta Weapon u sebi sadrži funkciju koja se poziva prilikom sudara dva objekta: u ovom slučaju oružja koje se može pokupiti i igrača. Prilikom pokretanja ove funkcije igraču se dodaje municija za oružje. Na slici 6 prikazano je kako izgleda entitet koji predstavlja oružje sačmarica koja puca 3 projektila odjednom, a na slici 7 prikazano je kako izgleda entitet mina koji stoji na mjestu i čeka da neprijatelj naleti na nju.



Slika 9. Oružje 1 sačmarica

Slika 10. Oružje 2 mina

4.6.2.6.Bullet.cs

```
Rigidbody2D body;
int dir=1;
public int dmg;
private Transform target;
private float fireRate;
Vector3 smjer;

public void SetAtr(Transform Target, float brzina) {
    target = Target;
    fireRate = brzina+10;
}
```

Funkcija SetAtr postavlja privatne atribute target i fireRate na vrijednosti koje smo proslijedili kod instanciranja metka.

```
public void ChangeDirection()
{
    dir *= -1;
}
```

Ova funkcija omogućava promjenu smjera tako što množimo s -1. Metak čija je vrijednost smjera 1 predstavlja igračev metak, a metak čija je vrijednost smjera -1 predstavlja neprijateljski metak.

```
private void Awake()
{
    body = GetComponent<Rigidbody2D>();
}
```

Awake je ugrađena funkcija koja se poziva prilikom učitavanja skripte. U ovoj funkciji dodijeljena je vrijednost varijabli body koja je Rigidbody2D komponenta metka.

```
void Start()
{
    if(dir==-1)
     smjer = target.position - transform.position;
}
```

Funkcija Start kod učitavanja skripte provjerava je li metak neprijateljski i ako je postavlja smjer kretanja metka prema igraču.

```
void Update()
{
    float speed = fireRate/10 * Time.deltaTime;
    transform.Translate(smjer.normalized * speed, Space.World);
    if (body.tag =="Mine")
        body.velocity = new Vector2(0,0);
    else
        body.velocity = new Vector2(0,5 * dir);
}
```

Funkcija Update svaki okvir postavlja smjer kretanja neprijateljskog metka prema zadnjoj lokaciji gdje je igrač bio. Ako igrač puca i oružje je mina tada se ona ne smije kretati, pa se brzina postavlja na 0.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (dir==1)
    {
        if (collision.gameObject.tag == "Enemy")
        {
            collision.gameObject.GetComponent<Enemy>().Damage(dmg);
            Destroy(gameObject);
        }
        if ( collision.gameObject.tag == "BossEnemy")
        {

collision.gameObject.GetComponent<BossBehaviour>().Damage(dmg);
            Destroy(gameObject);
        }
    }
    else
```

```
        {
            if (collision.gameObject.tag == "Player")
            {
                collision.gameObject.GetComponent<Spaceship>().Damage();
                Destroy(gameObject);
            }
        }

    }
```

Prilikom sudara metak provjerava svoj smjer i ovisno o smjeru radi štetu igraču ili neprijatelju. Smjer 1 označava da je metak ispalio igrač i da treba raditi štetu glavnom neprijatelju ili običnim neprijateljima, a ako je smjer -1 tada je metak namijenjen igraču. Nakon svakog sudara uništava se.

```
public void ChangeColor(Color col)
{
    GetComponent<SpriteRenderer>().color = col;
}
```

Ova funkcija mijenja boju metka. Boja metka se prosljeđuje na samom objektu koji poziva metak u procesu kreiranja objekta.

4.6.2.7. Killzone.cs

```
void OnTriggerEnter2D(Collider2D col)
{
    Destroy(col.gameObject);
}
```

Ova skripta uništava svaki objekt koji izađe van dopuštene zone.

4.6.2.8. Health.cs

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.tag == "Player")
    {
        collision.gameObject.GetComponent<Spaceship>().AddHealth();
        Destroy(gameObject);
    }
}
```

Skripta Health u sebi sadrži funkciju koja se poziva kada se dogodi sudar između dva objekta: u ovom slučaju života koji se može pokupiti i igrača. Ako se ova funkcija pokrene, onda se igraču dodaje jedan život i objekt za život se uništava.

Slika 11. Život

4.6.2.9.ScoreTxt.cs

```
void Update()
{
    GetComponent<Text>().text = PlayerPrefs.GetInt(txtName) +"";
}
```

Skripta ScoreTxt ima jednu funkciju, a to je da ispisuje sav potreban tekst na ekran.

4.6.2.10.Death.cs

```
GameObject player;
bool isDead = false;
void Start()
{
    player = GameObject.FindGameObjectWithTag("Player");
}
```

Skripta Death pri učitavanju traži objekt igrača po njegovoj oznaci.

```
void Update()
{
    if (player == null && !isDead)
    {
        GameOver();
    }
}
```

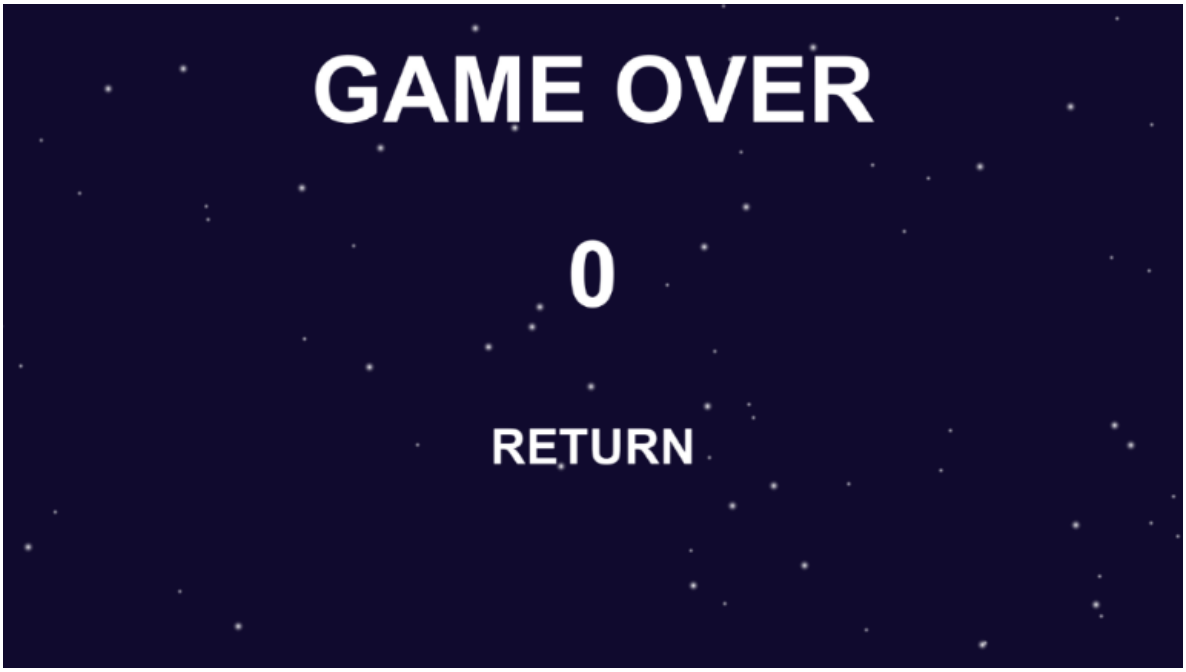
Svaki okvir aplikacija provjerava je li igrač živ, ako nije onda poziva funkciju GameOver.

```
void GameOver(){
    isDead = true;
    StartCoroutine(LoadGameOver());
}
```

GameOver postavlja varijablu isDead na true i pokreče korutinu LoadGameOver.

```
    IEnumerator LoadGameOver(){
        yield return new WaitForSeconds(2);
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex +
1);
}
```

LoadGameOver čeka 2 sekunde i nakon toga postavlja trenutnu scenu na scenu koja označava kraj igre.



Slika 12. Kraj igre

5. Zaključak

Izrada završnog rada pružila mi je zanimljivo i poučno iskustvo rada s Unity-em i detaljniji uvid u procese planiranja i izrade igre. Unity je jako jednostavan za korištenje i odličan je alat za početnike, jer je potrebno samo par minuta kako bi se korisnik upoznao sa sučeljem i mogao započeti s radom u istom. Uz programsko sučelje koje je bilo jednostavno za korištenje, izradu je uvelike olakšala i Unity zajednica koja pruža mnoštvo online materijala, i uvijek je spremna pomoći pojedincima koji su odlučni naučiti kako što jednostavnije i brže koristiti ovaj programski softver za izradu igrica.

Rad je podijeljen i razrađen u dva velika poglavlja. U prvom poglavlju opisao sam besplatan programski alat Unity i njegovo sučelje koje me iznenadilo jednostavnošću. Drugo poglavlje podijeljeno je u tri potpoglavlja. Prvo potpoglavlje opisuje shoot 'em up žanr i njegov razvoj kroz povijest. Drugo potpoglavlje je detaljna razrada koda koji je napisan u C# i prikaza različitih scena i elemenata igre, dok treće potpoglavlje opisuje igre koje su poslužile kao inspiracija za ovaj rad i povijest njihovog nastanka.

6. Popis literature

[1] Zackariasson, P. i Wilson L., T., The Video Game Industry: Formation, present State, and Future, New York, Routledge, 2012., str. 5.

[2] J. Davies, (17.06.2021) The shooting never stops, Dostupno:

http://www.gamespy.com/articles/895/895329p1.html

[3] J. Brodkin, (17.06.2021)How Unity3D Became a Game-Development Beast, Dostupno:

https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

[4] Learning C# and coding in Unity for beginners – Unity, Dostupno:

https://unity3d.com/learning-c-sharp-in-unity-for-beginners#:~:text=The%20language%20that%27s%20used%20in,variables%2C%20functions%2C%20and%20classes

[5] Powerful 2D, 3D, VR, & AR software for cross-platform development of games and mobile apps Dostupno: https://store.unity.com

[6] United States Copyright office, Dostupno:

https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=PA0000120007&Search_Code=REGS&PID=e4ENv-3Gaq0EGnPMkMU-_j643efG&SEQ=20210621122516&CNT=25&HIST=1

[7] Smithsonian Magazine Dostupno:

https://www.smithsonianmag.com/science-nature/original-space-invaders-icon-1970s-America-180969393/

[8] M. Bielby, „The complete ys guide to Shoot-'em-ups“,Your Sinclair Magazine, izd. 55 Dostupno: Internet Archive

https://archive.org/details/your-sinclair-55/page/n32/mode/2up

7. Popis slika

Slika 1. Unity sučelje 2
Slika 2. Speće Invaders igra [7] 3
Slika 3. Chicken Invaders v1.3 5
Slika 4. Raptor v2.4 6
Slika 5. Izbornik 8
Slika 6. Glavna scena 9
Slika 7. Glavni neprijatelj 16
Slika 8. Pobjeda 18
Slika 9. Oružje 1 sačmarica 19
Slika 10. Oružje 2 mina 19
Slika 11. Život 22
Slika 12. Kraj igre 23