Izrada horor videoigre preživljavanja u programskom alatu Unity

Zagorščak, Noel

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Noel Zagorščak

IZRADA HOROR VIDEOIGRE PREŽIVLJAVANJA U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Noel Zagorščak

Matični broj: 0016142903

Studij: Informacijski sustavi

IZRADA HOROR VIDEOIGRE PREŽIVLJAVANJA U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Mladen Konecki

Varaždin, rujan 2022.

Noel Zagorščak

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Izrada horor videoigre preživljavanja smještena u post apokaliptičnom svijetu u kojem prevladavaju zombiji. Upoznavanje sa višeplatformskim game engine-om koji je korišten prilikom izrade videoigre. Upoznavanje sa logikom izrade horor videoigre. Prije prikazivanja izrade videoigre, teorijski će biti objašnjene tehnologije koje su se koristile za izradu videoigre.

Ključne riječi: horor; videoigra; višeplatformski; zombiji; game engine; tehnologije;

Sadržaj

1. Uvod

Prije objašnjenja izrade videoigre, prvo je potrebno objasniti tehnologiju pomoću koje je sama videoigra napravljena. Nakon što je tehnologija objašnjena, slijedi upoznavanje sa logikom koja se koristila prilikom izrade završnog rada. Videoigra je napravljena uz pomoć višeplatformskog game engine-a koji nosi naziv Unity. Unity je engine specifičan za izradu video igara, bile one trodimenzionalne (3D) ili dvodimenzionalne (2D). Također, osim za desktop, mogu se izraditi i mobilne videoigre te je zbog tih razloga jedan od najzastupljenijih game engine-a na tržištu. U radu će biti opisane metode i tehnike koje su korištene upotrebom već spomenutog engine-a kako bismo realizirali ideju u videoigru.

Prvi dio dokumentacije fokusirat će se na opisivanje tehnologija korištene za izradu. Točnije, na već spomenuti Unity te na programski jezik korišten prilikom izrade videoigre, C# / ASP.NET Core te besplatni IDE (eng. Integrated development environment) za C# / ASP.NET Core, Microsoft Visual Studio.

U drugom dijelu dokumentacije, opisuje se logika smještena u pozadini videoigre, odnosno programski kod koji se koristio kako bi videoigra bila funkcionalna te izrada same videoigre. Neke od tih logika su pucanje, brojanje ubijenih zombija, mijenjanje oružja, broj metaka određenog oružja i slično.

2. Tehnologije

Za izradu videoigre korištene su sljedeće tehnologije:

- Unity,
- Microsoft Visual Studio,
- C# / ASP.NET Core

2.1. Unity

Unity je višeplatformski game engine izdan od kompanije naziva Unity Technologies 2005. godine. Unity je primarno korišten za razvoj video igara i simulacija za računala, konzole i mobilne uređaje. Jedan od razloga zašto je ovaj game engine toliko popularan je taj što je omogućen za korištenje na dvadeset sedam različitih platformi. Neke od tih platformi su Windows, iOS, Android, Linux, Playstation te Xbox. Unity omogućava izradu videoigara u dvije dimenzije (2D), ali također i u tri dimenzije (3D) te to sve podupire skriptiranjem u programskom jeziku C#. Osobito je popularan za izradu mobilnih igara. Osim toga jedan je od boljih odabira za virtualnu stvarnost. [1]

Dvije godine kasnije, točnije 2007. godine, na tržište je izašla nova, poboljšana verzija Unity-ja, Unity 2.0. Ona je donijela pedeset novih značajki koji su dodatno unaprijedili izradu igrica. Neke od njih su bile dinamičke sjene u stvarnom vremenu, usmjerena svjetla i reflektori te video reprodukcija. Također je uključivao mrežni sloj za programere za stvaranje igara za više igrača.

2010. godine izlazi Unity 3.0 koji je dodatno poboljšao grafičke značajke za stolna računala i igrače konzole. Osim toga Unity 3.0 uključuje integraciju Beast Lightmap alata Illuminate Labs, ugrađeni uređivač stabala, audio filtre i još mnogo drugih značajki.

2012. godine Unity Technologies je objavio Unity 4.0 koji je dodao podršku za DirectX 11 i Adobe Flash, nove alate za animaciju zvan Mecanim, koji će također biti korišten pri izradi igrice, i pristup Linuxu. 2013. godine Facebook je integrirao komplet za razvoj softvera za igre koristeći Unity, što je dodatno doprinijelo popularnosti ovog game engine-a te su tri godine kasnije Facebook i Unity razvili novu platformu za PC igrice.

Poboljšanja za svjetlost i zvuk donijela je nova verzija Unity-ja, Unity 5 2015. godine. Uz dodatak WebGL, programeri mogu dodati svoje igre kompatibilnim web preglednicima bez dodataka potrebnih za igrače. Osim toga, Unity 5 nudio je globalno osvjetljenje u stvarnom vremenu, Unity Cloud, novi audio sustav i Nvidia PhysX 3.3 engine za fiziku.

Verzija Unity-ja koja se i danas koristi izašla je 2017. godine. Alati te verzije sadržavali su mehanizam za renderiranje grafike u stvarnom vremenu, izgradnju svjetla, analitiku operacija uživo i izvješćivanje o učinku. No Unity je tada napravio korak više. Alatima Timeline programerima je omogućeno povlačenje i ispuštanje animacija u igre, a alatom Cinemachine, omogućeni su sustavi pametnih kamera unutar igara.

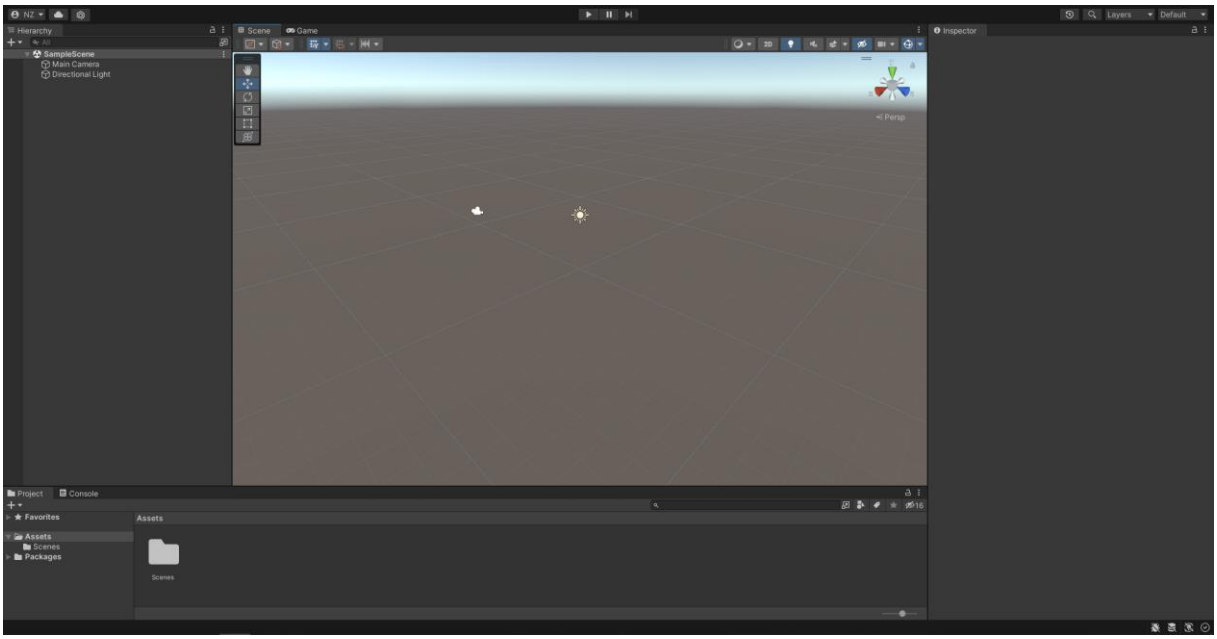
Od 2020. godine, softver izrađen pomoću Unity-jevog game engine-a radio je na više od 1,5 milijardi uređaja. Prema Unity-ju, aplikacije izrađene s njihovim game engine-om čine pedeset posto svih mobilnih igara te se one preuzmu više od tri milijarde puta mjesečno. Osim toga na dnevnoj bazi se pokrene oko petnaest tisuća novih projekata. [2]

Neke od najpopularnijih Unity igara su:

- Cuphead,
- Pillars Of Eternity,
- Ori And The Blind Forest,
- Heartstone,
- My Friend Pedro,
- Rust,
- Cities: Skylines
- Fall Guys: Ultimate Knockout,
- Pokemon Go,
- Beat Saber (VR) [3]

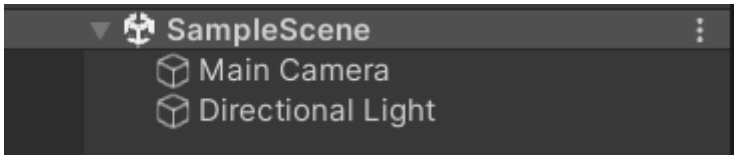
2.1.1. Početni zaslon

Ulaskom u Unity, prvi prozor koji se otvara je scene prozor. Desno od prozora scene, nalazi se game prozor koji omogućava pregled igrice bez pokretanja iz perspektive glavne kamere.



Slika 1: Scene prozor (vlastita izrada)

S lijeve strane vidimo hijerarhiju te dva objekta koja se nazivaju glavna kamera (eng. Main Camera) i osvjetljenje (eng. Directional Light), te su oba prikazana u prozoru scene (slika 1).



Slika 2: Main Camera i Directional Light (vlastita izrada)

S desne strane prozora scene, imamo Inspektor. Inspektor nam omogućava promjene nad objektima igrice. Odabirom određenog objekta, možemo mu dodatno postaviti postavke koje želimo, npr. to može biti pozicioniranje, pozadina, projekcija itd.

U donjem dijelu prozora nalaze se projektni prozor i konzola. Projektni prozor je potreban kako bi programeri mogli koristiti sredstva i skripte koje možemo implementirati u igricu. Također unutar prozora imamo određeni raspored, tako da s lijeve strane se nalaze mape unutar kojih se nalaze proizvoljne stvari te klikom na jednu od njih prikazuju se stavke koje se nalaze u njoj na desnoj strani prozora. U mapi asset, nalaze se scene koje možemo koristit te po potrebi možemo napraviti još mapa za korištenje, a unutar tih mapa određene skripte. Prozor konzole prikazuje greške prilikom kompiliranja skripte, upozorenja te poruke koje programer odluči da ispiše.

2.1.2. Asset Store

Unity Technologies, ali ujedno i članovi zajednice stvaraju sredstva potrebna za izradu igre. Navedena sredstva objavljuju na Asset Store-u. Postoje razne vrste sredstava u trgovini, od tekstura, animacija i modela do cijelih primjera projekata, vodiča i proširenja.

Neka sredstva su besplatna dok su neka pristupačna po određenoj cijeni te nakon kupovine sredstva isti se može i preuzeti. Također ukoliko neki član napravi vlastito sredstvo to isto može objaviti na Asset Store-u te na taj način postaje izdavač. [9]

Neki od sredstava koje možete koristiti su:

- 3D sredstva
- 2D sredstva
- Dodatci
- Zvuk
- Alati
- VFX

2.2. Microsoft Visual Studio

Microsoft Visual Studio je Microsoft-ov IDE koji se koristi za razvoj računalnih programa kao i web stranica, web aplikacija, web usluga i mobilnih aplikacija. Visual Studio koristi Microsoftove platforme za razvoj softvera kao što su Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Visual Studio uključuje uređivač koda koji podržava IntelliSense koji se koristi kao komponenta za dovršetak koda, kao i refaktoriranje koda.

Ostali ugrađeni alati uključuju alat za profiliranje koda, dizajner za izradu GUI aplikacija, web dizajner, dizajner klasa i dizajner sheme baze podataka.

Visual Studio prihvaća dodatke koji proširuju funkcionalnost na gotovo svim razinama. Zbog toga Visual Studio podržava trideset šest različitih programskih jezika i omogućuje programeru da podrži gotovo bilo koji programski jezik, pod uvjetom da postoji usluga specifična za jezik. [4]

Neke od dodataka uključuju :

- GitHub dodatak,
- CodeMaid
- ReSharper

- Visual Assist [5]

Najosnovnije izdanje Visual Studija, Visual Studio Community je besplatan i dostupan svima.

Glavni programski jezik Visual Studia je C# / ASP.NET Core. [6]

2.3. C# / ASP.NET Core

ASP.NET Core je višeplatformski, besplatni okvir za izgradnju modernih aplikacija. Uz ASP.NET Core možete izraditi web aplikacije i usluge i IoT aplikacije. [6]

Programski jezik C# dizajnirao je Anders Hejlsberg iz Microsofta 2000. godine. 2003. godine Microsoft je predstavio C# zajedno s .NET Framework-om i Visual Studiom te su oba bila zatvorenog koda. 2004. godine započeo je projekt otvorenog izvornog koda pod nazivom Mono, pružajući višeplatformski kompajler i runtime okruženje za C# programski jezik. Desetljeće kasnije, Microsoft je objavio Visual Studio Code koji je služio kao uređivač koda, Roslyn koji je služio kao kompilator i jedinstvenu .NET platformu koja je služila kao softverski okvir. Svi su podržavali C# te su besplatni, otvorenog koda i mogu se koristiti na više platformi.

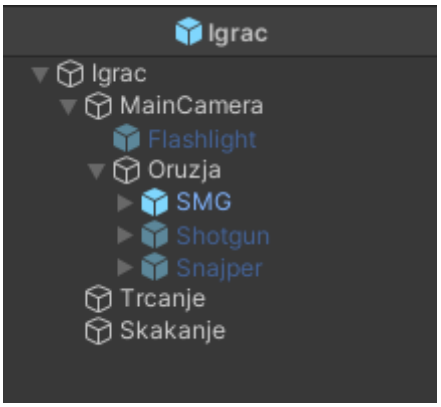
Od srpnja 2022. najnovija stabilna verzija jezika je C# 10.0, koja je objavljena 2021. u .NET 6.0. [7]

3. Izrada računalne igre

Kako bi bilo moguće započeti izrađivati igricu, potrebno je prvo imati viziju kako bi igrica trebala izgledati. U ovom projektu, to je horor igrica preživljavanja. S obzirom na temu, postojat će glavni lik u prvom licu koji se kreće terenom te ubija zombije. Cilj mu je ubiti dvadeset zombija koji su smješteni na različitim lokacijama na terenu. Glavni lik ima tri puške na raspolaganju, uzi, sačmaricu i snajper. Svako oružje je drugačije, od toga kako izgledaju, broj metaka, načinjena šteta zombiju do vremena potrebnog kako bi se mogao ispucati novi metak. Diljem terena postavljeni su stvari koji se mogu pokupiti. Glavni lik također uz sebe ima i svjetiljku koja s vremenom postaje sve tamnija.

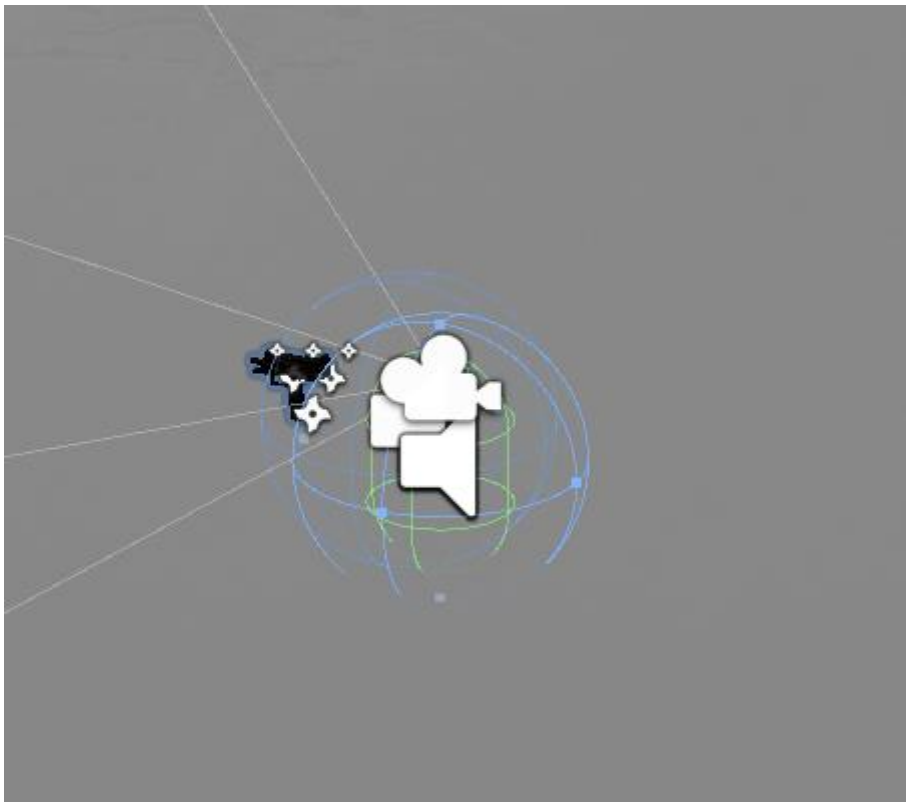
3.1. Glavni lik

Glavni lik je kreiran koristeći već izrađeno sredstvo u Unity Asset Store-u te je uz pomoć toga ubačen u igru. Uz glavnog lika, dobiju se određene skripte kako bi ga se moglo pomicati, skakati, trčati i slično.



Slika 3: Hijerarhija igrača (vlastiti izrada)

Kako je već navedeno, glavni lik je u prvom licu te je iz tog razloga potrebno glavnu kameru napraviti djetetom glavnog lika, odnosno u hijerarhiji mora biti ispod glavnog lika kako bi ga kamera stalno pratila. Osim kamere, oružje i svjetiljka također moraju biti u hijerarhiji ispod glavnog lika, ali također i ispod glavne kamere kako bi bili stalno dostupni i vidljivi. Već spomenuto, svjetiljka se s vremenom smanjuje i postaje sve slabija. Od oružja su dostupni uzi, sačmarica i snajper. Naknadno su dodani objekti za trčanje i skakanje, no to je potrebno samo za zvuk.



Slika 4: Igrač prefab (vlastiti izrada)

3.1.1. Svjetiljka

Funkcionalnost svjetiljke se očituje u njezinoj skripti naziva Flashlight.

```
private void Start()
{
    mojeSvijetlo = GetComponent<Light>();
}

private void Update()
{
    SmanjiBateriju();
    SmanjiRadijus();
}

public void ObnoviFlashlight(float obnoviRadijus)
{
    mojeSvijetlo.spotAngle = obnoviRadijus;
}

public void ObnoviBateriju(float intensityKolicina)
{
    mojeSvijetlo.intensity += intensityKolicina;
}

private void SmanjiBateriju()
{
    if (mojeSvijetlo.spotAngle <= minSmanjenjeRadijusa)
```

```
            return;
        else
            mojeSvijetlo.spotAngle -= smanjenjeRadijusa * Time.deltaTime;
    }

    private void SmanjiRadijus()
    {
        mojeSvijetlo.intensity -= smanjenjeBaterije * Time.deltaTime;
    }
```

Programski kod 1: Skripta za svjetiljku, Flashlight

Pozivom metode Start(), inicijalizira se varijabla mojeSvijetlo te se dohvaća svjetiljka koja je u hijerarhiji dijete glavnog lika.

Metodom Update() se pozivaju druge dvije metode, odnosno metode SmanjiBateriju() i SmanjiRadijus(). Metoda SmanjiBateriju() će smanjivati snagu baterije, odnosno svjetiljka će biti sve slabija sve dok ne dođe do minSmanjenjeRadijusa koji iznosi 40. Metoda SmanjiRadijus() s vremenom smanjuje opseg kruga.

Osim navedenih metoda imamo i metode ObnoviFlashlight(float obnoviRadijus) i ObnoviBateriju(float intensityKolicina) koje se aktiviraju kada glavni igrač pokupi bateriju na terenu. One resetiraju, odnosno zbrajaju već postojećoj količini proslijeđenu vrijednost.

3.1.2. Oružje

Oružje kao mapa nema potrebnih skripti, već se skripte nalaze na prefabu određenog oružja te glavnog igrača. Na svakom oružju nalazi se skripta naziva Oruzje koja služi za pucanje, prikaz metaka i prikaz VFX-a vezanog za oružje.

```
    void Update()
    {
        PrikaziMetke();
        if (Input.GetButton("Fire1") && mozePucati)
        {
            StartCoroutine(Pucaj());
        }
    }

    private void PrikaziMetke()
    {
        int trenutacniMeci =
municijaOruzja.KolikoImaMunicije(tipoviMetaka);
        tekstZaMetke.text = trenutacniMeci.ToString();
    }

    IEnumerator Pucaj()
    {
        mozePucati = false;
        if (municijaOruzja.KolikoImaMunicije(tipoviMetaka) > 0)
        {
            if(!GetComponentInParent<IgracHealth>().mrtavIgrac)
            {
                GetComponent<AudioSource>().Play();
```

```
                PokaziFlash();
                RaycastMetoda();
                municijaOruzja.SmanjiMuniciju(tipoviMetaka);
            }
        }
        yield return new WaitForSeconds(delay);
        mozePucati = true;
    }

    private void PokaziFlash()
    {
        flash.Play();
    }

    private void RaycastMetoda()
    {
        RaycastHit pogodak;
        if (Physics.Raycast(FPCamera.transform.position,
FPCamera.transform.forward, out pogodak, domet))
        {
            PrikaziFlash(pogodak);
            HealthZombija zombi =
pogodak.transform.GetComponent<HealthZombija>();
            if (zombi == null) return;
            zombi.SkiniHealth(damage);
        }
        else
        {
            return;
        }
    }
    private void PrikaziFlash( RaycastHit pogodak)
    {
        GameObject impact = Instantiate(pogodakEffect, pogodak.point,
Quaternion.LookRotation(pogodak.normal));
        Destroy(impact, .1f);
    }
```

Programski kod 2: Skripta za oružje, Oruzje

Metoda Update() ima zadatak prikazati broj metaka na zaslon kako bi igrač znao koliko ih ima, što se odvija u metodi PrikaziMetke() te provjerava je l' igrač stisnuo tipku za pucanje, odnosno „Fire1“ što predstavlja lijevi klik miša. Osim toga provjerava se također može li puška pucati te ukoliko su oba uvjeta zadovoljena, pokreće se korutina Pucaj(). Pucaj() je tipa IEnumerator te se korutina može pauzirati u bilo kojem trenutku pomoću naredbe yield. [11] Unutar te metode se prvo provjerava ima li puška dovoljno metaka za pucati, odnosno ima li više od 0. Ukoliko uvjet nije ispunjen puška neće pucati. Ukoliko je zadovoljen, provjerava se je l' igrač mrtav kako bi se izbjeglo pucanje nakon što igrač umre. Ako je igrač živ, odnosno još uvijek ima dovoljan broj životnih bodova oružje će moći pucati te će se izvršiti dvije nove metode PokaziFlash() koja služi kao VFX metoda prikaza bljeska nakon pucanja. Druga metoda koja će se izvršiti je RaycastMetoda() kojoj je cilj usmjeriti metak. Unutar te metode imamo uvjet ukoliko metak pogodi nešto, odnosno zombija onda će se izvršiti metoda

PrikaziFlash(pogodak) koja će prikazati pogodak na zombiju te će se skinuti životni bodovi istoga.

Za snajper također ima skripta naziva Zoom pomoću koje se omogućava korištenje nišana.

```
private void Update()
{
    if (Input.GetMouseButtonDown(1))
    {
        if (!zoomUpaljen)
        {
            zoomUpaljen = true;
            zoomCamera.fieldOfView = zoomedIN;
            fpsController.mouseLook.XSensitivity = zoomedINSensitivity;
            fpsController.mouseLook.YSensitivity = zoomedINSensitivity;
        }
        else
        {
            odzumiraj();
        }
    }
}

private void odzumiraj()
{
    zoomUpaljen = false;
    zoomCamera.fieldOfView = zoomedOUT;
    fpsController.mouseLook.XSensitivity = zoomedOUTSensitivity;
    fpsController.mouseLook.YSensitivity = zoomedOUTSensitivity;
}
```

Programski kod 3: Skripta za snajper, Zoom

Skripta Zoom ima dvije metode, Update() i odzumiraj(). Dok metoda odzumiraj() vraća prethodne postavke nišana, metoda Update() provjerava je l' pritisnuta tipka za zumiranje, odnosno desni klik miša. Ukoliko je, provjerava se je l' zum upaljen. Ako je upaljen, poziva se metoda odzumiraj(). Ako nije upaljen onda se postavke nišana smanjuju kako bi prikaz na ekranu bio što veći, odnosno bliži.

3.1.3. Broj metaka

Kako bi igrač znao broj metaka za određeno oružje koristi skriptu Municija.

```
private class MunicijaOruzja
{
    public TipoviMetaka tipMetka;
    public int kolicinaMetaka;
}

public int KolikoImaMunicije(TipoviMetaka tipMetka)
{
    return DohvatiMetkeOdredenogTipaOruzja(tipMetka).kolicinaMetaka;
```

```
    }

    public void SmanjiMuniciju(TipoviMetaka tipMetka)
    {
        DohvatiMetkeOdredenogTipaOruzja(tipMetka).kolicinaMetaka--;
    }

    public void DodajMuniciju(TipoviMetaka tipMetka, int kolicinaMunicije)
    {
        DohvatiMetkeOdredenogTipaOruzja(tipMetka).kolicinaMetaka +=
kolicinaMunicije;
    }


    private MunicijaOruzja DohvatiMetkeOdredenogTipaOruzja(TipoviMetaka
tipMetka)
    {
        foreach(MunicijaOruzja item in Slots)
        {
            if(item.tipMetka == tipMetka)
            {
                return item;
            }
        }
        return null;
    }
```

Programski kod 4: Broj metaka oružja, Municija

```
    public enum TipoviMetaka
      {
      MetakZaSMG,
      MetakZaShotgun,
      MetakZaSniper
      }
```

Programski kod 5: Tipovi metaka, TipoviMetaka

Prije metoda, imamo privatnu klasu unutar klase Municija naziva MunicijaOruzja koja sadrži dvije varijable. Varijablu tipMetka tipa TipoviMetaka i količinaMetaka cjelobrojnog tipa.

Prilikom poziva bilo koje od metoda unutar skripte, prosljeđuje se tip metka kako bi se znalo za koje oružje smanjujemo metke. Uzi koristi MetakZaSMG, sačmarica koristi MetakZaShotgun te snajper koristi MetakZaSniper.

Metoda KolikoImaMunicije(TipoviMetaka tipMetka) vraća broj metaka određenog tipa te se vraćena vrijednost koristi kao prikaz broja metaka na zaslon kako bi igrač znao preostalu količinu metaka koje može koristiti.

Metode SmanjiMuniciju(TipoviMetaka tipMetka) i DodajMuniciju(TipoviMetaka tipMetka) pozivaju metodu DohvatiMetkeOdređenogTipaOružja(TipoviMetaka tipMetka). Zadnje navedena metoda iterira kroz tipove metaka te vraća one metke za oružje koje igrač ima trenutačno vidljivo. Zatim metoda SmanjiMuniciju smanjuje broj metaka tog oružja, odnosno metoda DodajMuniciju povećava broj metaka tog oružja ukoliko se na terenu pokupi stavka predodređena za to.

3.1.4.Životni bodovi

Upravljanje životnih bodova igrača se svodi na skriptu IgracHealth.

```
public void SkiniHealth(float damage)
{
    health -= damage;
    PrikaziHealth();
    if (health <= 0)
    {
        mrtavIgrac = true;
        GetComponent<Smrt>().NakonSmrtiPokaziCanvas();
    }
}

public float VratiBrojUbijenihZombija()
{
    return brojUbijenihZombija;
}

public void PovecajBrojUbijenihZombija()
{
    brojUbijenihZombija++;
}

public float VratiHealth()
{
    return health;
}

private void PrikaziHealth()
{
    if(VratiHealth() < 0)
        tekstZaHealth.text = "0 HP";
    else
        tekstZaHealth.text = VratiHealth().ToString() + " HP";
}

public void DodajHealth(float dodajHealth)
{
    if ((health + dodajHealth) > 100)
        health = 100;
    else
        health += dodajHealth;
    PrikaziHealth();
}
```

Programski kod 6: Životni bodovi igrača, IgracHealth

Metoda SkiniHealth(float damage) poziva se kada se zombi dovoljno približi igraču te ga udari. Ukoliko se to dogodi, igrač gubi petnaest životnih bodova od ukupno sto. Ako životni bodovi padnu na nulu, igrač je mrtav te se prikazuje platno (eng.canvas) koji prikazuje kako je igra gotova.

Metoda VratiBrojUbijenihZombija() se koristi kako bi se brojalo koliko je zombija ubijeno u igrici, odnosno koliko ih je igrač ubio. Ova metoda se primarno koristi kako bi se mogao

ispisati broj ubijenih zombija na canvas predodređen za to. Metoda PovecajBrojUbijenihZombija() povećava broj ubijenih zombija igrača.

Metoda VratiHealth() vraća trenutačni broj životnih bodova igrača te se to koristi u metodi PrikaziHealth().

Ako korisnik pokupi prvu pomoć, aktivira se metoda DodajHealth(float dodajHealth) koja igraču povećava broj životnih bodova za pedeset. Broj životnih bodova ne može biti ispod nule niti može biti iznad sto.

3.1.5.Smrt

Skripta Smrt zadužena je za platno koji se prikazuje nakon što broj životnih bodova igrača padne na nulu.

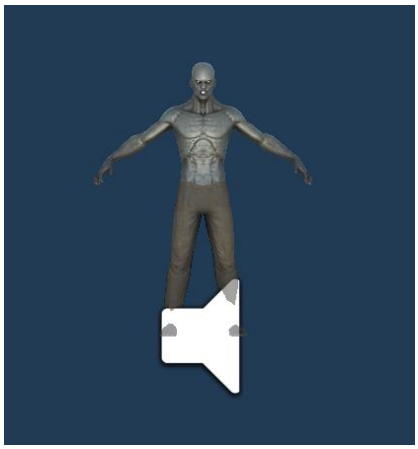
```
public void NakonSmrtiPokaziCanvas()
{
    gameOver.enabled = true;
    Time.timeScale = 0;
    FindObjectOfType<PromijeniOruzje>().enabled = false;
    GetComponent<AudioSource>().Stop();
    Cursor.lockState = CursorLockMode.None;
    Cursor.visible = true;
}
```

Programski kod 7: Smrt igrača, Smrt

Metoda NakonSmrtiPokaziCanvas() prikazuje platno na kojem je korisniku ponuđeno da ponovno pokrene igricu ili da se vrati na početni zaslon.

3.2. Neprijatelj

Neprijatelj, odnosno zombi, isto kao i igrač, već je unaprijed napravljeno sredstvo koje je uvezeno iz Asset Store-a. Kako je cilj igre ubiti dvadeset zombija i svi moraju biti isti, napravljene su neke preinake u prefabu samog zombija. Prefab u Unity-ju su unaprijed konfigurirani objekti igre za višekratnu upotrebu koje se stvaraju u sceni te su pohranjene u projektu. [10]



Slika 5: Prefab zombija (vlastita izrada)

Kako bi zombi bio funkcionalan u igrici, odnosno kako bi ga igrač mogao ubiti i obrnuto, potrebne su određene skripte koje svaki zombi mora imati.

Prva i glavna skripta koju zombi ima naziva se EnemyAI.

```
    void Start()
    {
        navMeshAgent = GetComponent<NavMeshAgent>();
        health = GetComponent<HealthZombija>();
        igrac = FindObjectOfType<IgracHealth>().transform;
        igrac2 = FindObjectOfType<IgracHealth>();
    }

    void Update()
    {
        float brojMrtvihZombija = 0;
        if (health.JelMrtav())
        {
            igrac2.PovecajBrojUbijenihZombija();
            brojMrtvihZombija = igrac2.VratiBrojUbijenihZombija();
            tekstZaZombije.text = brojMrtvihZombija.ToString() + "/20";
            smrt.Play();
            enabled = false;
            navMeshAgent.enabled = false;
        }

        if(brojMrtvihZombija == 20)
        {
            igrac.GetComponent<PredenaIgra>().PokaziCanvas();
        }

        udaljenostDoIgraca = Vector3.Distance(igrac.position,
transform.position);

        if (loviIgraca)
            UhvatiIgraca();
        else if(udaljenostDoIgraca <= range)
            loviIgraca = true;
    }
```

```
    private void UhvatiIgraca()
    {
        UsmjeriSe();
        if(udaljenostDoIgraca  >= navMeshAgent.stoppingDistance)
        {
            LoviIgraca();
        }
        if(udaljenostDoIgraca <= navMeshAgent.stoppingDistance)
        {
            NapadniIgraca();
        }
    }


    public void UpucajNeprijatelja()
    {
        loviIgraca = true;
    }

    private void LoviIgraca()
    {
        GetComponent<Animator>().SetBool("napadni", false);
        GetComponent<Animator>().SetTrigger("pokreni");
        navMeshAgent.SetDestination(igrac.position);
    }

    private void NapadniIgraca()
    {
        GetComponent<Animator>().SetBool("napadni", true);
    }

    private void UsmjeriSe()
    {
        Vector3 direction = (igrac.position -
transform.position).normalized;
        Quaternion lookRotation = Quaternion.LookRotation(new
Vector3(direction.x, 0, direction.z));
        transform.rotation = Quaternion.Slerp(transform.rotation,
lookRotation, Time.deltaTime * brzinaOkretanja);
    }
```

Programski kod 8: Skripta zombija, EnemyAI

Metoda Start() dodjeljuje komponente varijablama kako bismo ih mogli koristiti u ostalim metodama. Nakon nje dolazi metoda Update() koja provjerava životne bodove zombija. Ukoliko su životni bodovi zombija nula, odnosno ako ga je igrač ubio, povećava se broj ubijenih zombija za jedan te se igraču na platnu ispisuje broj ubijenih zombija od ukupno dvadeset. Također se zombi onemogućuje, odnosno enabled i navMeshAgent.enabled se postavljaju na false kako zombi ne bi nastavio pratiti igrača iako je mrtav. Nakon toga se provjerava broj mrtvih zombija. Ukoliko je taj broj jednak dvadeset, igra je pređena te se prikazuje platno određen za tu svrhu. Ako zombi nije mrtav i ako igrač nije ubio svih dvadeset zombija, zombi počinje loviti igrača ukoliko se on nalazi na određenoj udaljenosti od njega.

Za to postoji metoda UhvatiIgraca(). Unutar metode se pozivaju tri nove metode, svaka za drugu svrhu. Metoda UsmjeriSe() koristi Vector3 kako bi usmjerio poziciju zombija da se okreće prema igraću, u smislu da kad lovi igrača torzo i lice zombija okrenuti su prema igraču. Druga metoda koja se poziva je LoviIgraca(). Ona se poziva ukoliko se igrač previše približio zombiju. Ako se zombi previše približi igraču, on će ga napasti te se to odvija u trećoj pozvanoj metodi NapadniIgraca().

Skripta HealthZombija je zadužna za praćenje životnih bodova zombija.

```
public bool JelMrtav()
{
    return mrtav;
}

public void SkiniHealth(float damage)
{
    BroadcastMessage("UpucajNeprijatelja");
    health -= damage;
    if(health <= 0)
    {
        Smrt();
    }
}

private void Smrt()
{
    if (mrtav) return;
    mrtav = true;
    GetComponent<AudioSource>().Stop();
    GetComponent<Animator>().SetTrigger("umri");
    GetComponent<CapsuleCollider>().enabled = false;
}
```

Programski kod 9: Životni bodovi zombija, HealthZombija

Pozivom metode JelMrtav() provjeravaju se životni bodovi zombija. Ukoliko su oni nula, varijabla mrtav je false te je zombi mrtav. Ako zombi ima više od nula životnih bodova, varijabla mrtav je true te je zombi živ. Ova metoda se koristi u skripti EnemyAI prilikom prve provjere unutar metode Update().

Metoda SkiniHealth(float damage) poziva se ukoliko igrač pogodi zombija te mu skida određeni broj životnih bodova, ovisno o puški koju je igrač koristio. Metoda BroadcastMessage poziva sve metode (u projektu postoji samo jedna) naziva UpucajNeprijatelja(). [12] Ako broj životnih bodova zombija iznosi nula, poziva se metoda Smrt().

Zadnja skripta koju svaki zombi ima je EnemyAttack.

```
void Start()
{
    target = FindObjectOfType<IgracHealth>();
}

public void AttackHitEvent()
{
    if (target == null) return;
    target.SkiniHealth(damage);
    target.GetComponent<UIPrikazDamagea>().PrikaziCanvas();
}
```

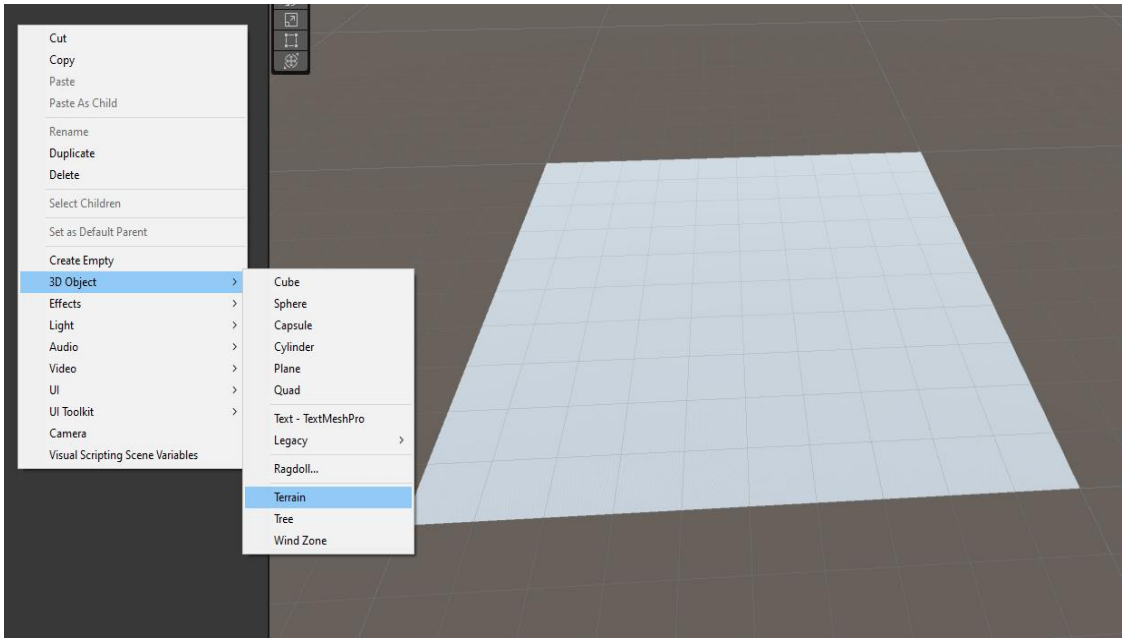
Programski kod 10: Napad zombija, EnemyAttack

Metoda Start() inicijalizira metu, odnosno igrača te ukoliko se zombi dovoljno približi igraču, on će ga napasti te se to odvija u metodi AttackHitEvent(). Unutar nje se poziva metoda SkiniHealth(damage) koja se nalazi u skripti IgracHealth. Nakon što se skinu životni bodovi igrača, prikazuje se canvas kako bi igrač imao vizualnu reprezentaciju napada.

3.3. Teren

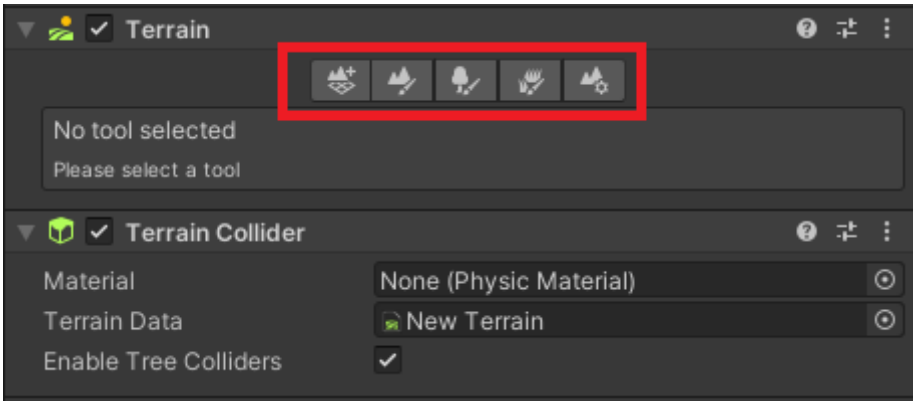
Nakon što je glavni lik napravljen, odnosno igrač i njegovi neprijatelji, potrebno je to sve staviti na teren gdje će se igrač kretati i pronalaziti zombije na različitim dijelovima mape.

Desnim klikom na prozor Hijerarhije otvara nam se izbornik te odabirom *3D Object* > *Terrain* stvaramo kvadrat na kojem će biti teren igrice.



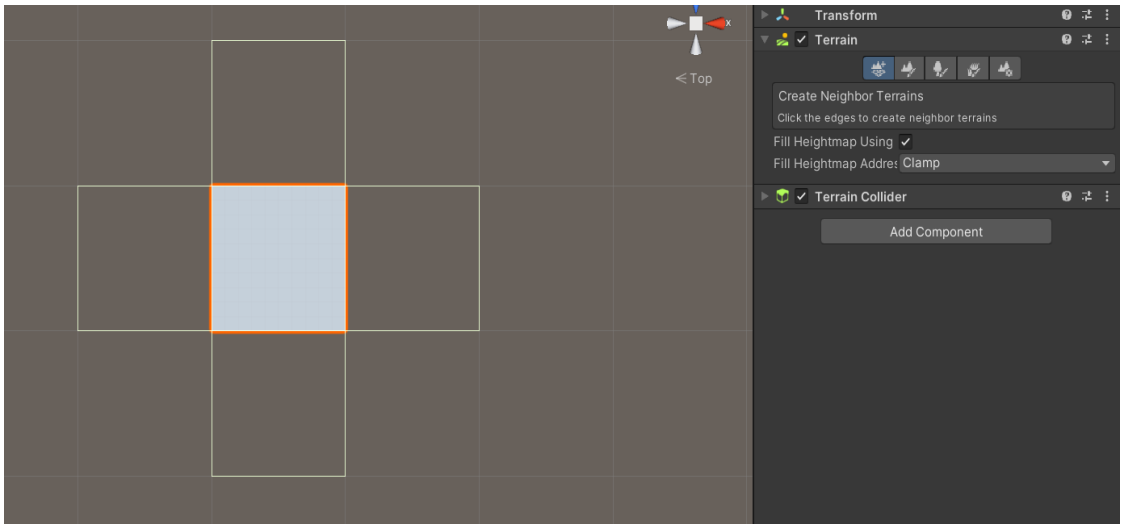
Slika 6: Teren (vlastita izrada)

Klikom na teren, u prozoru Inspektor, dobivamo određene mogućnosti kako bi teren mogli prilagoditi vlastitim potrebama.



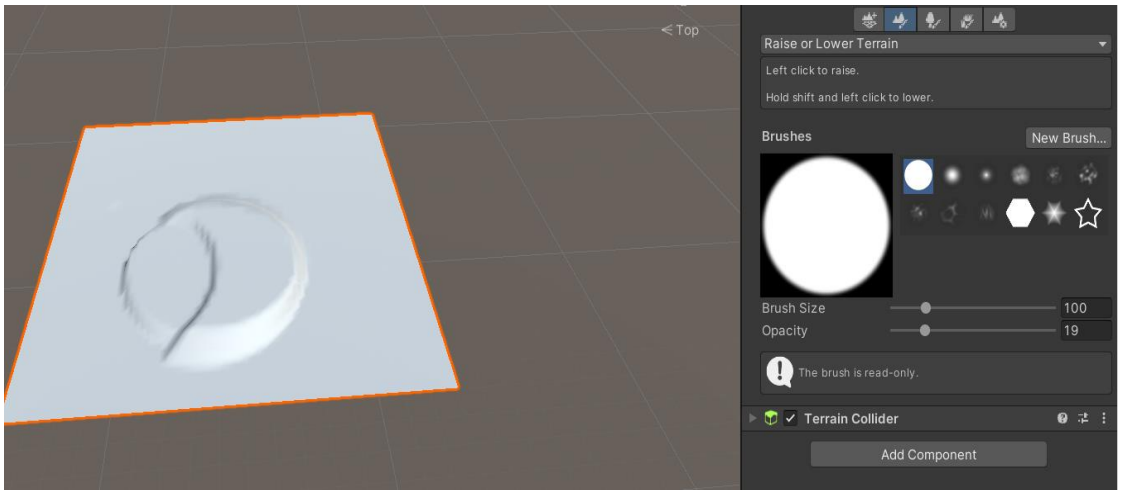
Slika 7: Mogućnosti terena (vlastita izrada)

Odabirom određenog alata drugačije su dostupne mogućnosti. Skroz lijevo (Slika 7, crveni pravokutnik) je alat koji stvara susjedni teren, odnosno mogućnost proširenja terena ovisno o potrebama programera.



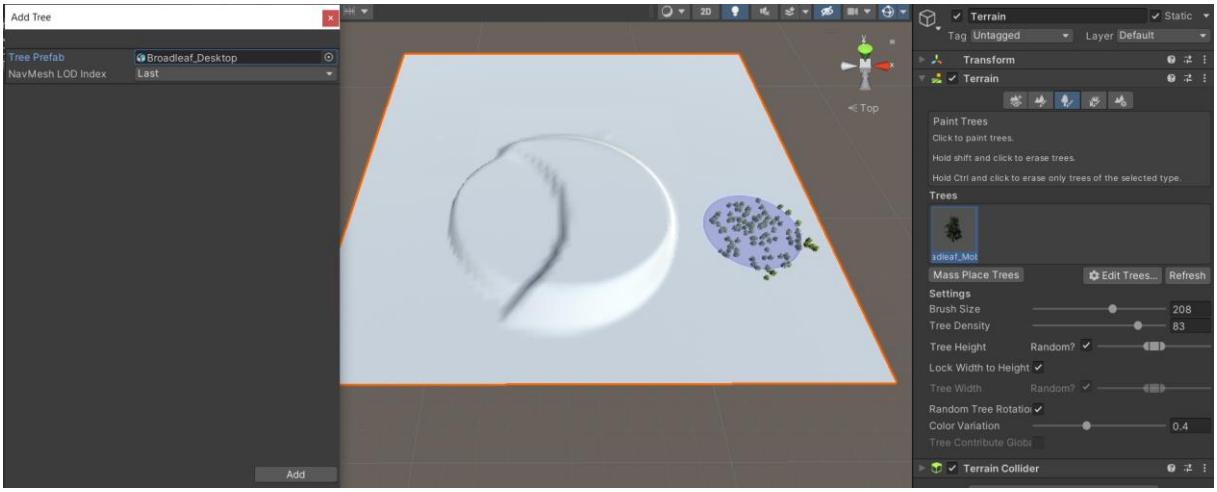
Slika 8: Stvori susjedni teren (vlastita izrada)

Sljedeći alat je za uređivanje terena te se mogućnost Raise or Lower Terrain primarno koristila za izradu ovog projekta. Odnosno pomoću toga se mogu raditi udubine, odnosno uzvisine na terenu. Kako bi teren podigli klikne se lijevim klikom miša, a kako bi ga udubili, potrebno je prvo držati shift pa kliknuti lijevim klikom miša.



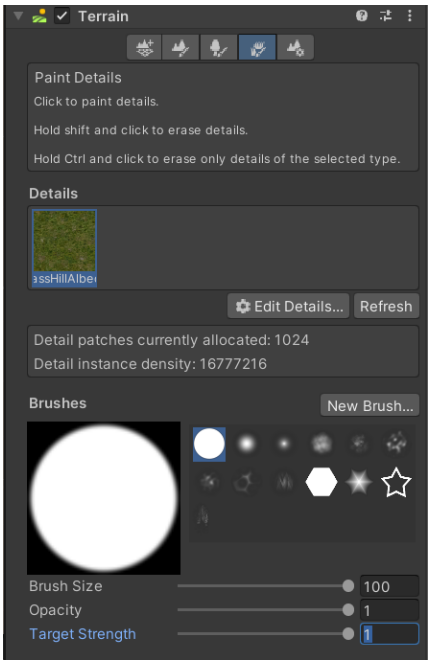
Slika 9: Podigni i spusti teren (vlastita izrada)

Nakon toga dolazi alat za dodavanje drveća. Kako bi drveće uspješno dodali na teren prvo se mora dodati kao mogućnost dodavanja. To se napravi tako što se klikne na *Edit Trees… > Add Tree*. Nakon što se otvori novi prozor klikom na kružić pokraj polja gdje piše Tree Prefab otvara se novi prozor sa svim dostupnim stvarima koje je moguće dodati. Nakon što se odabere željeno drvo, ono će se pojaviti kao jedna od mogućnosti dodavanja. Odabirom na tu mogućnost i desnim klikom na teren, stvorit će se određen broj drveća koji ovisi o gustoći i veličini kista kojim programer stavlja drveća.



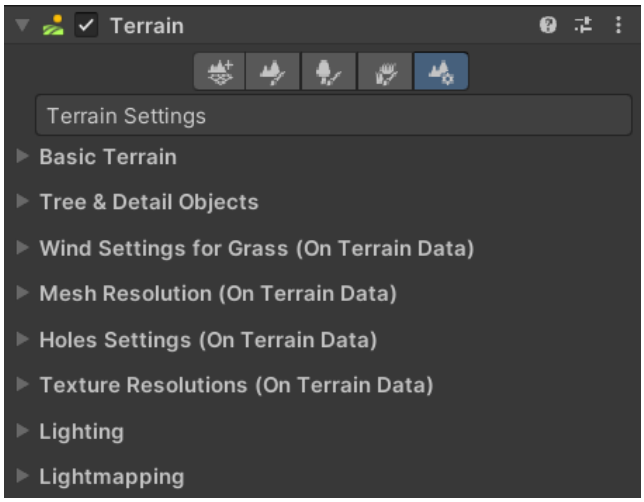
Slika 10: Dodavanje drveća (vlastita izrada)

Predzadnji alat koji je dostupan je za detalje, odnosno za travu i detalje vezane za nju. Princip korištenja je identičan kao i za dodavanje drveća. Najprije se mora dodati mogućnost dodavanja trave te se onda kistom postavlja na teren.



Slika 11: Dodavanje trave (vlastita izrada)

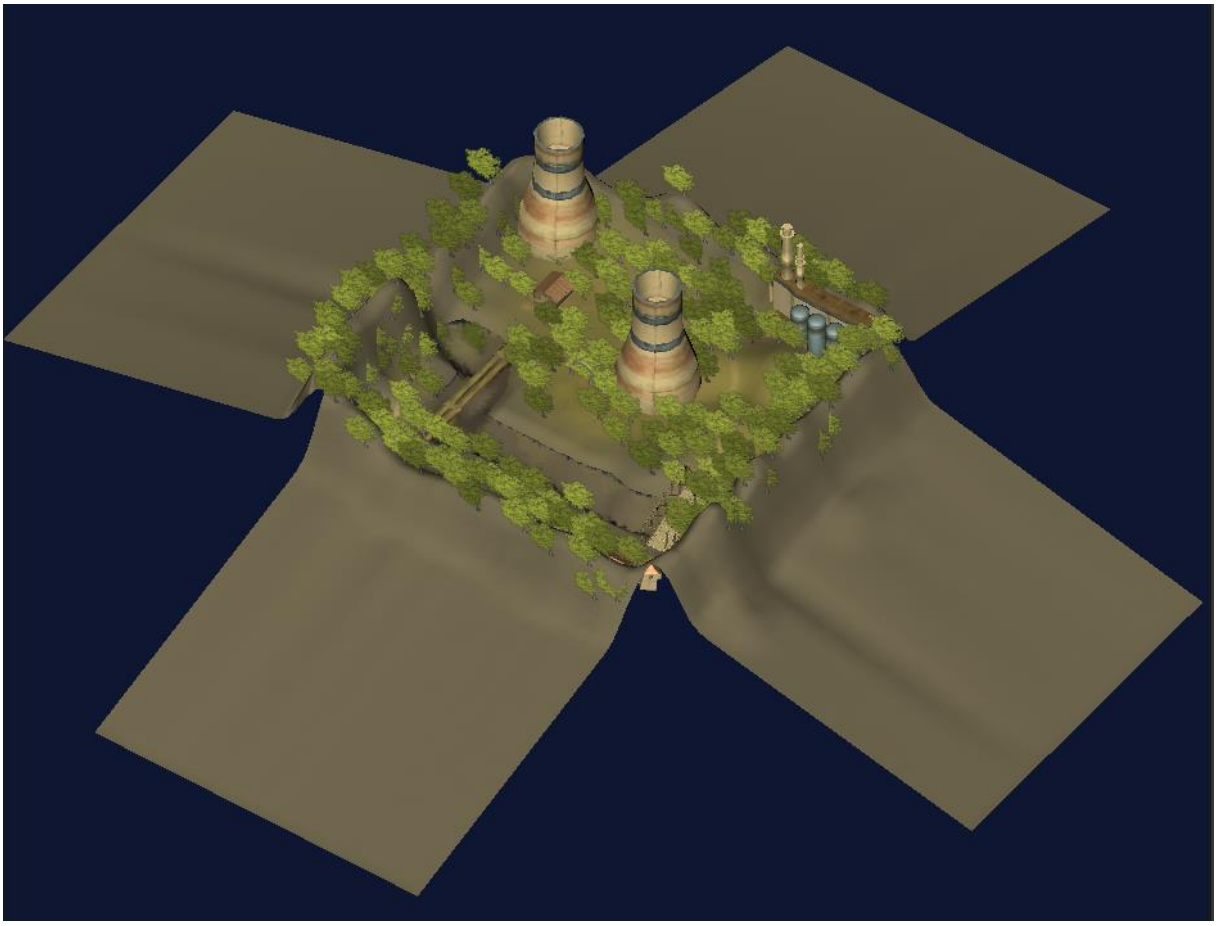
Zadnji alat koji programer može koristiti su svojstva terena. Pomoću toga programer na usluzi ima veliki broj mogućih svojstava koje može koristiti ovisno o potrebama za igricu.



Slika 12: Svojstva terena (vlastita izrada)

Nakon oblikovanja terena i dodavanja trave i drveća, pomoću Asset Store-a moguće je dodatno poboljšati izgled terena. Preuzimanjem besplatnih sredstava, dodavanjem željenih sredstava kao mogućnost dodavanja u igricu te koristeći isti postupak kao i za dodavanje drveća i trave, odnosno kistom, omogućeno je programeru dodati i razne građevine, oblike i predmete. Za potrebu ovog projekta neke od tih građevina su kolibe, mlin i tvornica. Za oblike su korišteni različiti tipovi kamenja te voda kako bi se napravio kanal. Klupe i mostovi su neki

od predmeta koji su korišteni. Kombiniranjem svega navedeno napravljen je teren za igricu. U nazivu projekta stoji horor igrica preživljavanja te će iz tog razloga osvjetljenje u igrici biti poprilično tamno (Na slici 13 je povećano osvjetljenje kako bi teren bio vidljiv). Koristit će se usmjerena svjetla jer su ona korisna za stvaranje efekata poput sunčeve svjetlosti. Ponašajući se na mnogo načina kao sunce, usmjerena svjetla se mogu smatrati udaljenim izvorima svjetlosti koje postoji beskrajno daleko. Ono nema prepoznatljiv položaj izvora pa se svjetlosni objekt može postaviti bilo gdje u sceni. Svi objekti u sceni su osvijetljeni kao da svjetlost uvijek dolazi iz istog smjera. Udaljenost svjetla od ciljanog objekta nije definirana pa se svjetlo ne smanjuje. [21]



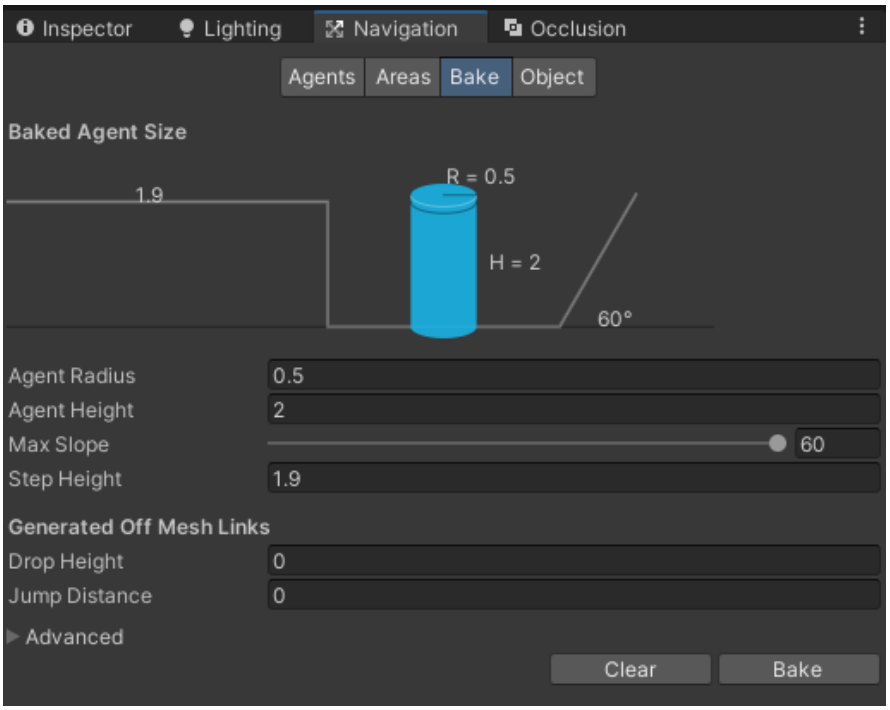
Slika 13: Teren za igricu (vlastita izrada)

3.3.1.Navesh

Kako bi se zombiji mogli kretati po tereni najprije je potrebno označiti područja po kojem se mogu kretati. Za to se koristi NavMesh. Navigacijska mreža ili NavMesh je apstraktna struktura podataka koja se koristi u aplikacijama umjetne inteligencije za pomoć agentima u pronalaženju puta kroz određeni prostor koji je označen kao statični. [13] ž

Izrada NavMesh-a se može napraviti u četiri koraka:

1. Odabere se geometrija (u ovom slučaju teren) scene koja bi trebala utjecati na navigaciju,
2. Uključiti Navigation Static kako bi uključili odabrane objekte u NavMesh procesu pečenja (eng. bake).
3. Prilagođavanje postavki pečenja tako da odgovaraju veličini agenta.

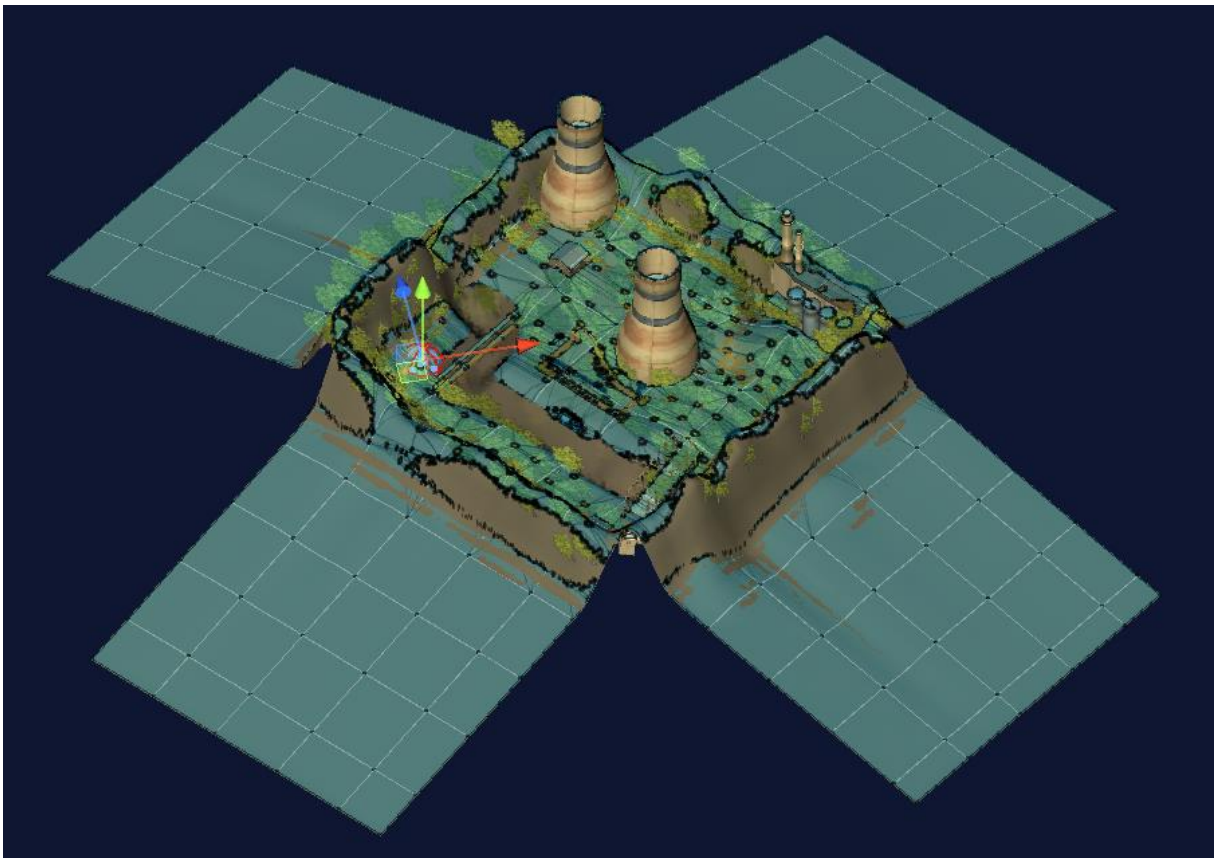


Slika 14: NavMesh Bake (vlastita izrada)

- Agent Radius definira koliko se središte agenta može približiti zidu ili izbočini,
- Agent Height definira koliko su niski prostori koje agent može dosegnuti,
- Max Slope definira koliko su strme rampe po kojima se agent može penjati,
- Step Height definira koliko su visoke prepreke na koje agent može stati.

4. Potrebno je pritisnuti gumb Bake za izradu NavMesh-a [14]

Nakon što je NavMesh izrađen, prostor na terenu na kojem agent može hodati bit će prikazan plavom bojom.



Slika 15: NavMesh agenta (vlastita izrada)

3.4. Stvari za pokupiti

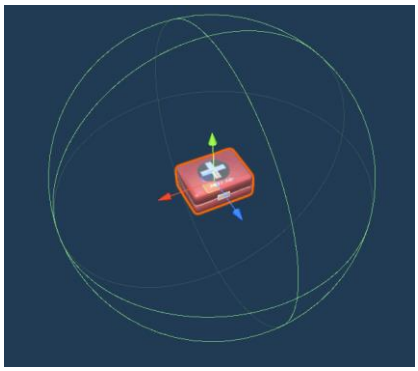
Stvari koji se mogu pokupiti su:

- Prva pomoć
- Baterija za svjetiljku
- Meci za uzi
- Meci za sačmaricu
- Meci za snajper

3.4.1. Prva pomoć

Skripta MedKit zadužena je za dodavanje životnih bodova igraču ukoliko se kolajderi igrača i kutije prve pomoći sudare.

Unity upravlja kolizijom između objekata igre pomoću kolajdera koji se spajaju na određeni objekt igre te definiraju oblik tog objekta za potrebe sudara. Kolajder je nevidljiv. Najjednostavniji kolajderi su primitivni tipovi kolajdera. U 3D-u to su Box Collider, Sphere Collider i Capsule Collider. [15]



Slika 16: Kolajder prve pomoći (vlastita izrada)

Kada se objekt igre sudari sa drugim objektom, Unity poziva metodu OnTriggerEnter. U skripti MedKit koja je dodijeljena kutiji prve pomoći, ukoliko dođe do kolizije kolajdera igrača i kutije, poziva se prijašnje navedena metoda. [16]

```
  private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Player")
        {

other.GetComponentInChildren<IgracHealth>().DodajHealth(obnoviHealth);
            Destroy(gameObject);
        }
    }
```

Programski kod 11: Kutija prve pomoći, MedKit

Najprije se provjerava tko se sudario sa kolajderom kutije prve pomoći, odnosno je l' to igrač ili nešto drugo. Ukoliko je igrač, poziva se metoda DodajHealth(float dodajHealth) skripte IgracHealth kako bi se životni bodova igrača povećali za pedeset. Nakon što se dodaju životni bodovi, kutija prve pomoći se pomoću metode Destroy(Object obj) uništava. Metoda Destroy(Object obj) je zadužena za ukljanjanje objekata igre, komponenti ili sredstava.[17]

3.4.2.Baterija za svjetiljku

Poput kutije prve pomoći, baterija za svjetiljku također funkcionira na principu kolajdera te je za to zadužena skripta Baterija.

```
private void OnTriggerEnter(Collider other)
    {
        if(other.gameObject.tag == "Player")
        {

other.GetComponentInChildren<Flashlight>().ObnoviFlashlight(obnovi);
```

```
other.GetComponentInChildren<Flashlight>().ObnoviBateriju(intensity);
            Destroy(gameObject);
        }
    }
```

Programski kod 12: Baterija za svjetiljku, Baterija

Ako se prilikom kolizije kolajdera utvrdi da je to igrač, pozivaju se metode ObnoviFlashlight(float obnoviRadijus) i ObnoviBateriju(float intensityKolicina) koje se nalaze u skripti Flashlight koja je dodjeljena igraču. Nakon što je svjetiljka obnovljena, objekt baterije se uništava metodom Destroy(Object obj).

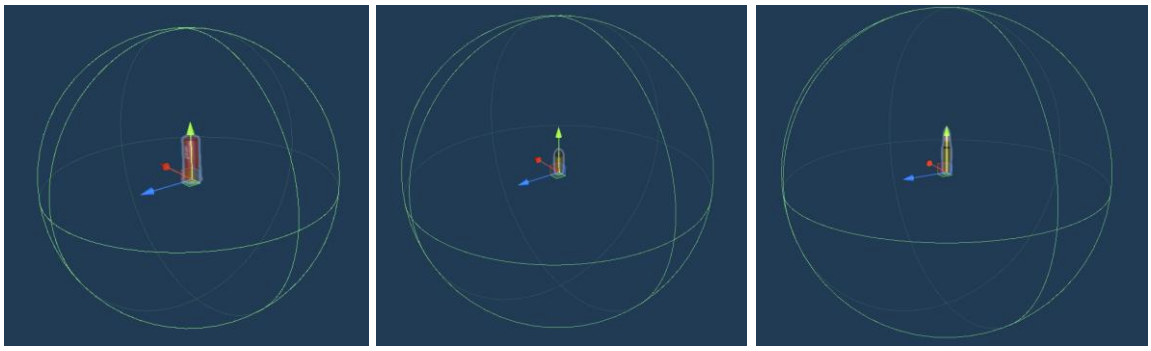
3.4.3.Metci

Neovisno za koje oružje, svaki metak koji se može pokupiti sadrži istu skriptu naziva Pickups.

```
    private void OnTriggerEnter(Collider other)
    {
        if(other.gameObject.tag == "Player")
        {
            FindObjectOfType<Municija>().DodajMuniciju(tipoviMetaka,
kolicinaMunicije);
            Destroy(gameObject);
        }
    }
```

Programski kod 13: Metci za pokupiti, Pickups

Kao i prva pomoć i baterija za svjetiljke, i metci koji se mogu pokupiti funkcioniraju na principu kolajdera. Prilikom kolizije i utvrđivanja da je to uistinu bio igrač, poziva se metoda DodajMuniciju(TipoviMetaka tipMetka, int kolicinaMunicije) skripte Municija te se ovisno o tome koji je tip metak zbraja na već postojeći broj metaka. Nakon toga se poziva metoda Destroy(Object obj) kako bi uništio objekt metka.



Slika 17: Meci za pokupiti, lijevo za sačmaricu, u sredini za uzi, desno za snajper (vlastita izrada)

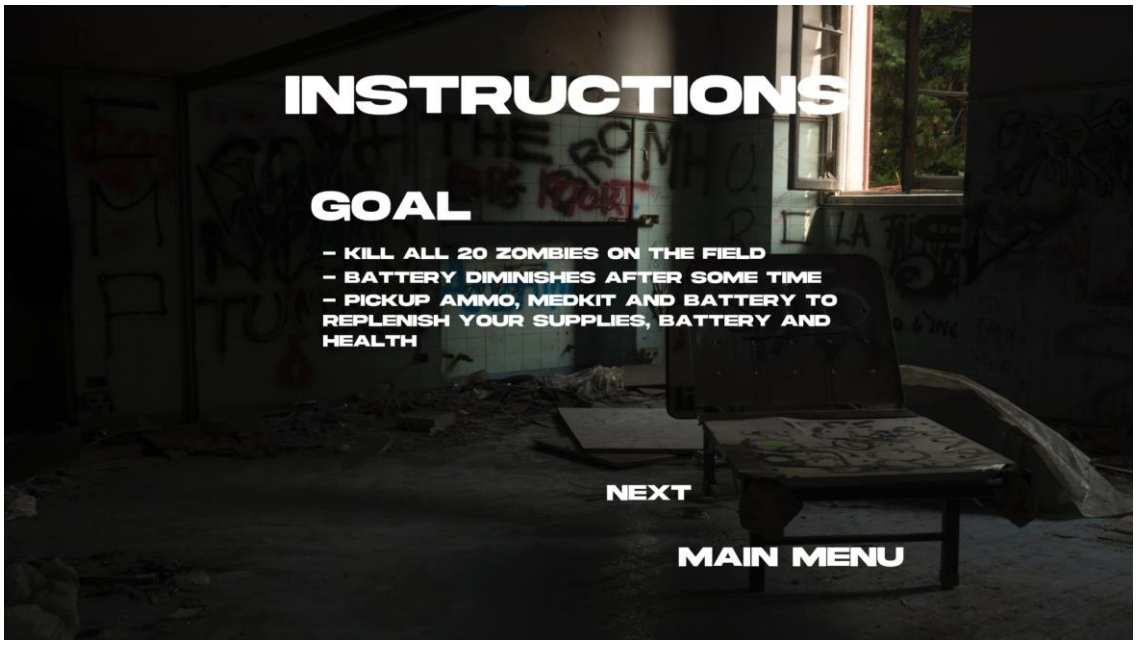
3.5. Početni zaslon

Prilikom pokretanja igrice otvara se početni zaslon na kojem postoje tri gumba.

Slika 18: Početni zaslon (vlastita izrada)

Prvi gumb je gumb Play, odnosno gumb koji pokreće igricu.

Drugi gumb je gumb Instructions koji daje uputstva za igricu, odnosno koji je njen cilj, koje su kontrole koje su dostupne igraču, koja su oružja dostupna te što se sve može pokupiti na terenu. Osim toga, instrukcije sadržavaju i savjete za prelazak igrice poput toga koliko životnih bodova daje kutija prve pomoći i slično.

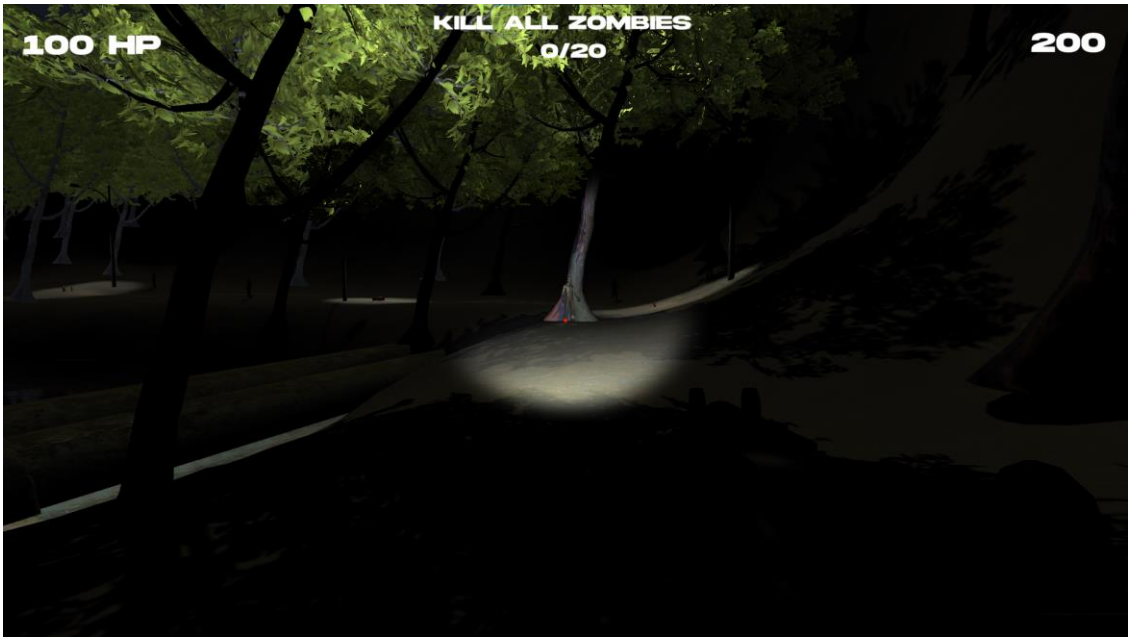


Slika 19: Instrukcije (vlastita izrada)

Treći gumb je gumb Quit koji, prilikom klika na njega, zatvara igricu.

3.6. User Interface

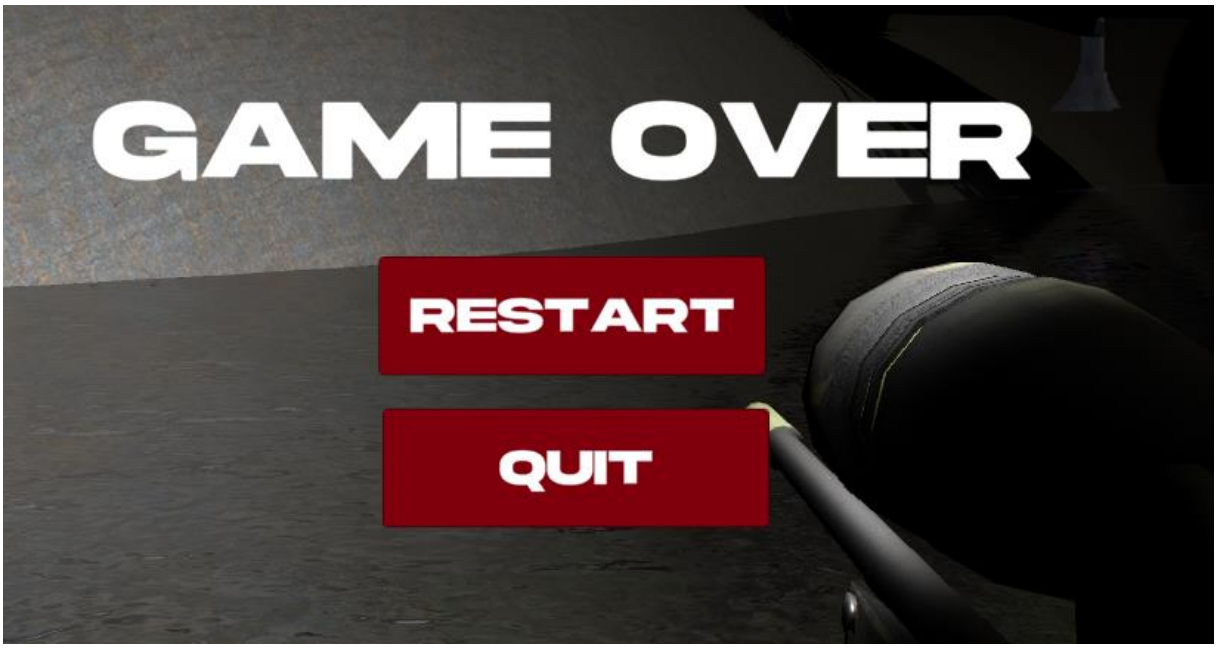
Prilikom klika na gumb Play na početnom zaslonu, igrica se pokreće te je igraču prikazan prizor kao na slici 20.



Slika 20: Pokretanje igrice (vlastita izrada)

U gornjem lijevom kutu prikazani su životni bodovi igrača. Životni bodovi ne mogu preći brojku sto niti mogu biti manji od nule. Nasuprot tome, odnosno u gornjem desnom kutu je prikazan broj dostupnih metaka trenutačno odabrane puške. U ovom slučaju puška uzi ima dvjesto metaka koje igrač može koristiti. Između broja životnih bodova i broja dostupnih metaka nalazi se cilj igrice, odnosno tekst „KILL ALL ZOMBIES“ koji znači ubij sve zombije. Ispod teksta je napisano 0/20 što znači da igrač još nije ubio nijednog zombija, a ima ih dvadeset sveukupno. Crvena točkica koja se vidi u sredini koristi kako bi igrač znao u kojem smjeru puca.

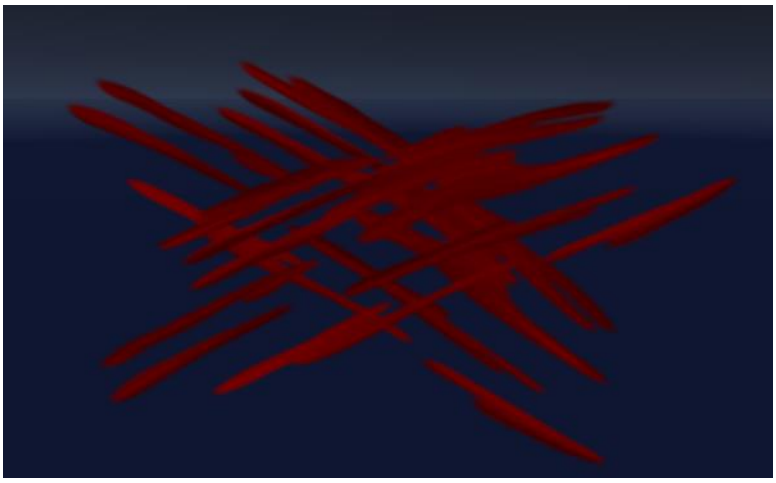
Ako igrač izgubi sve životne bodove ili pokuša prijeći rijeku drugim putem osim mostom, prikazan mu je sljedeći prikaz:



Slika 21: Game Over (vlastita izrada)

Kao što je prikazano na slici 21., igrač ima dvije opcije. Može ponovno pokušati preći igricu klikom na gumb Restart ili se može vratiti na početni zaslon gumbom Quit.

Kada se zombi previše približi igraču, on će ga raniti te će mu skinuti određeni broj životnih bodova. Napad zombija će igraču biti i vizualno prikazan.



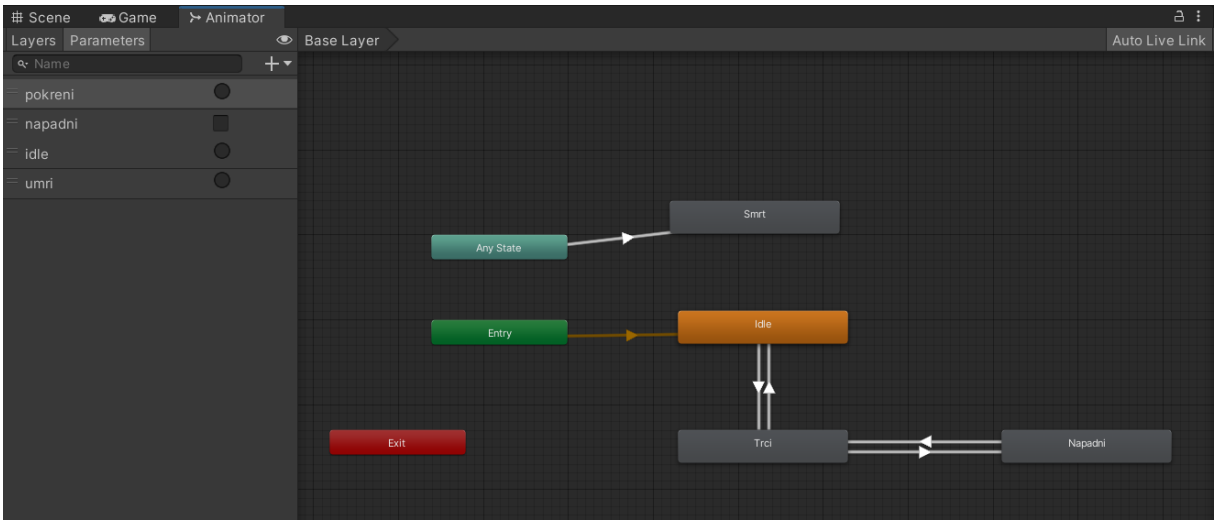
Slika 22: Prikaz napada zombija (vlastita izrada)

Nakon što igrač dođe do cilja, odnosno kad ubije svih dvadeset zombija, prikazan mu je sljedeći prizor:

Slika 23: Pređena igra (vlastita izrada)

3.7. Animator

Animator je zapravo sučelje za upravljanje animacijskim sustavom Mecanim.[18] Pristupa mu se klikom na *Window* > *Animation* > *Animator* te se, za zombija, otvara sljedeći prizor.

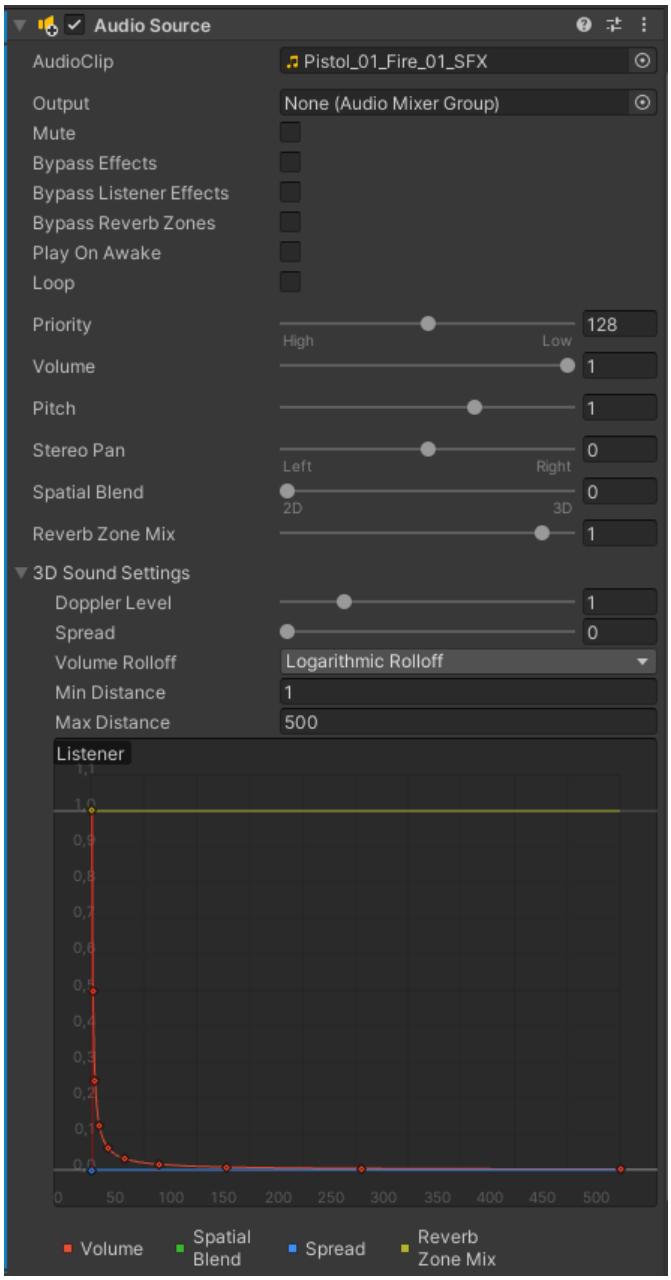


Slika 24: Animator zombija (vlastiti izvor)

Zombi ima četiri stanja ako ne računamo automatski generirana stanja. Prvo stanje je stanje Idle u kojem zombi stoji na mjestu. Ako se igrač dovoljno približi zombiju, aktivira se okidač pokreni te zombi prelazi u novo stanje, odnosno stanje Trci u kojem zombi lovi igrača. Ako se zombi previše približi igraču, parametar napadni postaje true te zombi napada igrača. Nakon napada ako se igrač odmakne od zombija, parametar napadi se postavlja na false te zombi ponovno počinje loviti igrača.

3.8. Zvuk

Kako bi se dodao zvuk na određeni objekt igre potrebno je prvo kliknuti odabrani objekt. Nakon toga se u inspektoru stisne na skroz donji gumb, Add Component gdje se odabere *Audio* > *Audio Source*.

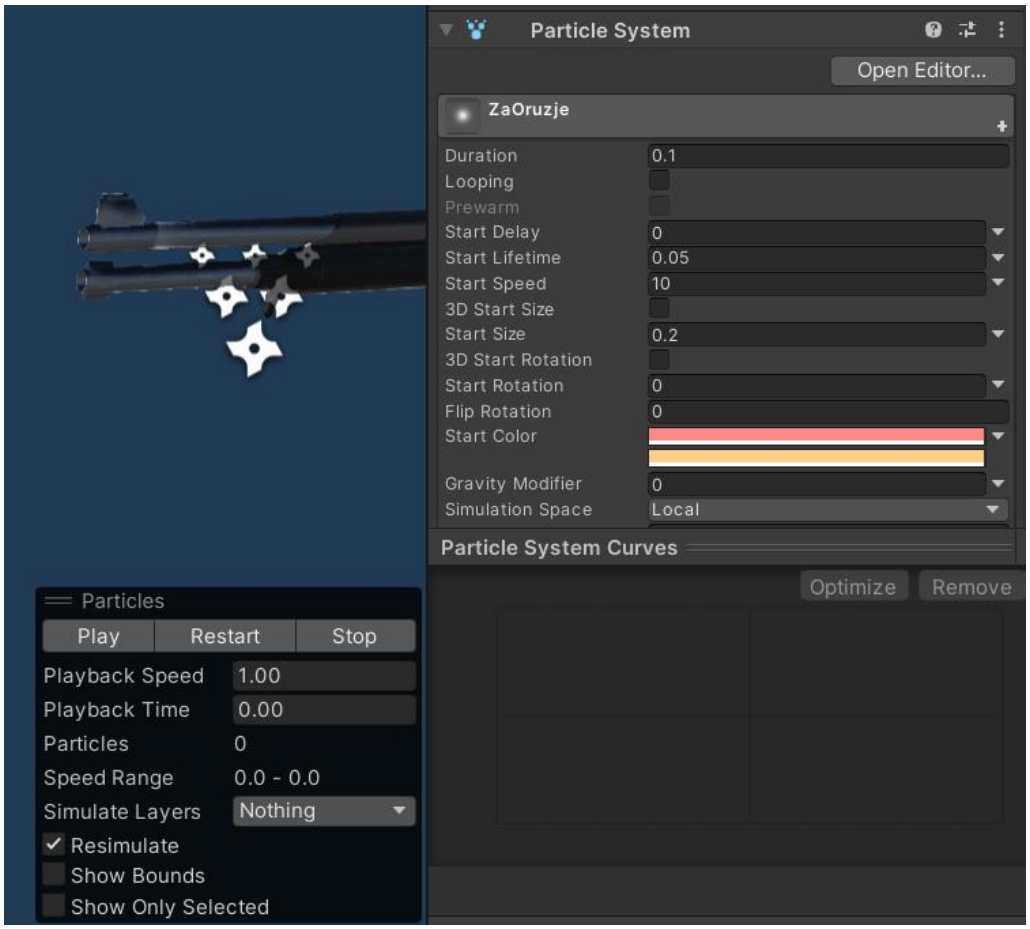


Slika 25: Zvuk (vlastita izrada)

AudioSource je tada dodijeljen odabranom objektu igrice te se on koristi za reprodukciju zvukova u 3D okruženju. Audio isječak (eng. AudioClip) je zapravo zvuk kojeg treba reproducirati dok su sve ostalo svojstva koje se mogu mijenjati nad dodanim zvukom. Koristeći Play, Pause i Stop reproducira se audio isječak te se također može podesiti njegova glasnoća tijekom igranja pomoću svojstva glasnoće. Više se zvukova može reproducirati na jednom audio isječku koristeći PlayOneShot. [19]

3.9. Sustav čestica

Unity ima sustav čestica u kojem se može simulirati pokretne tekućine, dim, oblaci, plamen, čarolije i čitav niz drugih efekata. [20]



Slika 26: Sustav čestica (vlastita izrada)

Pomoću gumbova Play, Restart i Stop moguće je provjeriti što će se točno dogoditi prilikom pokretanja sustava čestica. Pomoću raznih svojstava moguće je po potrebi promijeniti svojstva te dodatno prilagoditi sustav čestica. Duration označava koliko dugo će čestice biti prikazane te ako se Looping označi kvačicom, sustav će se ponavljati u beskonačnoj petlji.

4. Zaključak

Uz pomoć ugrađenih alata unutar Unity-ja, odnosno Asset Store-a uvelike je ubrzan proces izrade projekta. Osim toga prijašnje poznavanje C#-a je dodatno ubrzalo izradu. Postoje tri glavne komponente ovog projekta, a to su glavni lik, neprijatelj i teren. Fokus glavnog lika je stavljen na oružje koje posjeduje te na svjetiljku koja mu omogućava bolje osvjetljenje tijekom igranja. Na neprijatelju je korišteno nešto što na glavnom liku nije, a to je animacija. Za izradu terena korišteni su svi dostupni mogući alati koji su omogućeni korištenjem inspektora, odnosno dodavanje dodatnog terena, podizanje i spuštanje terena, dodavanje drveća i trave te promjena svojstava samog terena.

Realizacija projekta je uvelike lakša ukoliko se prijašnje odredi tema rada te plan za izradu istog. Izradom prijašnje navedenih glavnih komponenti te njihovim kombiniranjem većina je projekta bila napravljena. Sve što je još bilo potrebno je dodati detalje. Za glavnog lika i neprijatelje, to su sustav čestica i zvuk dok su za teren to bili detalji poput klupa, kamenja i slično.

Spajanjem svega navedenog, projekt je završen te je spreman za igranje.

Popis literature

- [1] freeCodeCamp (2020). Preuzeto 02.09.2022. s https://www.freecodecamp.org/news/unity-game-engine-guide-how-to-get-started-with-the-most-popular-game-engine-out-there/
- [2] Unity (game engine) (bez dat.). U Wikipedia. Preuzeto 02.09.2022. s https://en.wikipedia.org/wiki/Unity_(game_engine)
- [3] Jeff Drake (2020). *10 Great Games That Use The Unity Game Engine.* Preuzeto 02.09.2022. s https://www.thegamer.com/unity-game-engine-great-games/
- [4] Microsoft Visual Studio (bez dat.). U Wikipedia. Preuzeto 02.09.2022. s https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
- [5] Microsoft (bez dat.). *Extensions for Visual Studio*. Preuzeto 02.09.2022. s https://marketplace.visualstudio.com/
- [6] Microsoft (bez dat.). *Web Languages*. Preuzeto 02.09.2022. s https://visualstudio.microsoft.com/vs/features/web/languages/
- [7] C Sharp (programming language) (bez dat.). U Wikipedia. Preuzeto 02.09.2022. s https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
- [8] Unity Technologies (bez dat.) *Community-powered creator solutions*. Preuzeto 02.09.2022. s https://assetstore.unity.com/
- [9] Unity Technologies (bez dat.). *Quick guide to the Unity Asset Store*. Preuzeto 02.09.2022. s https://unity3d.com/quick-guide-to-unity-asset-store#:~:text=The%20Unity%20Asset%20Store%20is,examples%2C%20tutorials%20and%20Editor%20extensions.
- [10] Unity Technologies (2022). *Prefabs*. Preuzeto 02.09.2022. s https://docs.unity3d.com/Manual/Prefabs.html
- [11] Unity Technologies (2022). *MonoBehaviour.StartCoroutine*. Preuzeto 03.09.2022. s https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html
- [12] Unity Technologies (2022). *Component.BroadcastMessage*. Preuzeto 03.09.2022. s https://docs.unity3d.com/ScriptReference/Component.BroadcastMessage.html
- [13] Navigation mesh (bez dat.). U Wikipedia. Preuzeto 03.09.2022. s https://en.wikipedia.org/wiki/Navigation_mesh
- [14] Unity Technologies (2022). *Building a NavMesh*. Preuzeto 03.09.2022. s https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html

- [15] Unity Technologies (2022). *Introduction to collision.* Preuzeto 03. 09. 2022. s https://docs.unity3d.com/Manual/CollidersOverview.html
- [16] Unity Technologies (2022). Collider.OnTriggerEnter(Collider). Preuzeto 03.09.2022. s https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html
- [17] Unity Technologies (2022). *Object.Destroy*. Preuzeto 03.09.2022. s https://docs.unity3d.com/ScriptReference/Object.Destroy.html
- [18] Unity Technologies (2022). *Animator*. Preuzeto 03.09.2022. s https://docs.unity3d.com/ScriptReference/Animator.html
- [19] Unity Technologies (2022). *AudioSource.* Preuzeto 03.09.2022. s https://docs.unity3d.com/ScriptReference/AudioSource.html
- [20] Unity Technologies (2021). *Introduction to Particle Systems*. Preuzeto 03.09.2022. s https://learn.unity.com/tutorial/introduction-to-particle-systems#
- [21] Unity Technologies (2022). *Types of light.* Preuzeto 03.09.2022. s https://docs.unity3d.com/Manual/Lighting.html

Popis slika

Popis programskih kodova