Izrada igre beskonačnog trčanja u programskom alatu Unity

Sorić, Zvonimir

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zvonimir Sorić

Izrada igre beskonačnog trčanja u programskom alatu Unity

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zvonimir Sorić

Matični broj: 35918/16–R

Studij: Informacijski sustavi

Izrada igre beskonačnog trčanja u programskom alatu Unity

ZAVRŠNI RAD

Mentor:

Dr. sc. Konecki Mladen

Varaždin, rujan 2019.

Zvonimir Sorić

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/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj završni rad temelji se na izradi igrice u alatu Unity. Ukratko ću objasniti osnovno korištenje alata. Igrica se bazira na tipu beskonačnog trčanja. Glavni lik (igrač) neprestano trči te se neprestano približava preprekama koje mora preskočiti ili zaobići. Programski kod je pisan u programskom jeziku C# u Microsoft Visual Studio.

Ključne riječi: Unity, 2D, razvoj igre, C#, beskonačno trčanje, programiranje

Sadržaj

1. Uvod

Cilj ovog završnog rada je bolje upoznati alat Unity koji je namijenjen za izradu igrica. Igricu koju ću koristiti je tipa beskonačnog trčanja koja je ujedno i 2D. 2D igricu sam odabrao zbog svoje ljubavi prema platformerskim igricama kao što je Super Mario Bros. Jedan od ključnih razloga za moj odabir ove teme je bilo moje korištenje ovog alata u prošlosti. U vrijeme mog srednjoškolskog obrazovanja velik broj funkcionalnosti nije postojao. Te sam uživao u ponovnom učenju ovog alata koje nije bilo teško kao u mojoj prošlosti. Drago mi je da sam odabrao Unity, a ne neki drugi alat.

2. Metode i tehnike rada

Za izradu ovog završnog rada su korišteni alat Unity i Microsoft Visual Studio za programiranje. Sama ideja ove igrice nastala je gledanjem drugih igrica. Nije korištena niti jedna poznata metoda ili tehnika.

3. Alat Unity

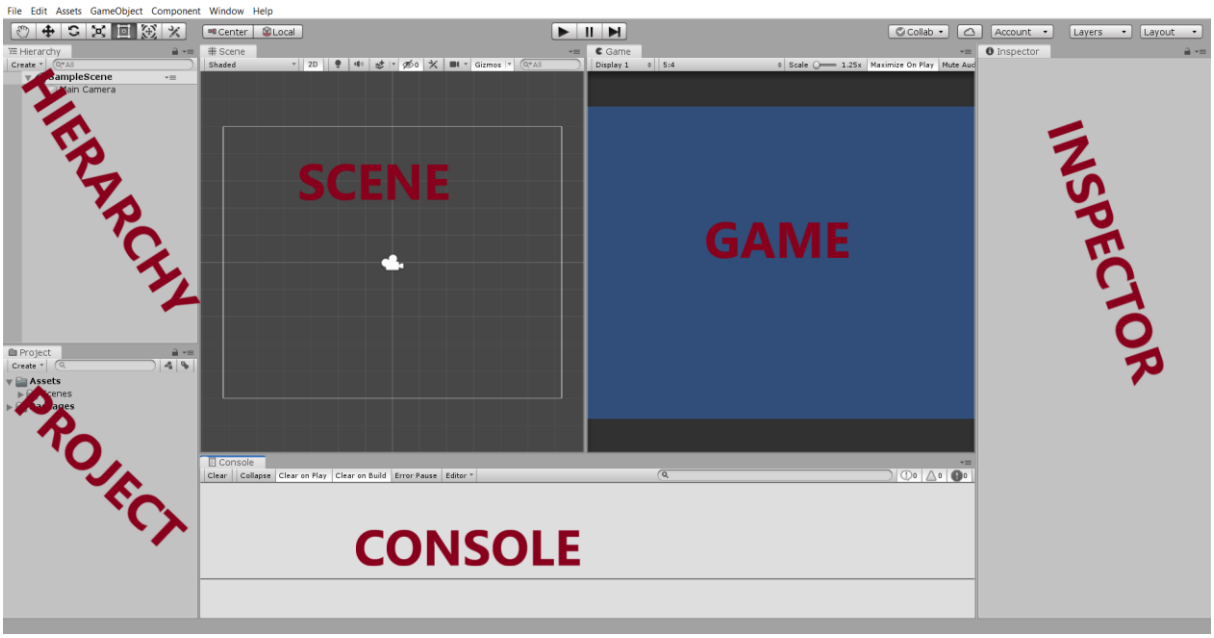
Unity alat je predstavljen javnosti 2005. godine. Koristio se isključivo za razvijanje OS igrica. Na današnji dan podržava 27 platformi. Trenutna verzija i verzija koju sam koristio je Unity 2019.1.0f2. Unity omogućuje kreiranje dvodimenzionalne i trodimenzionalne igrice. Primarno se koriti C# programski jezik. Unity ima jednu besplatnu Personal verziju i tri verzije koje se plaćaju Plus, Pro i Enterprise. Unity je početno napravljen za Mac OS te se kasnije dodala podrška za Microsoft Windows i Web pretraživače [1].

3.1. Povijest Unity alata

Unity 2.0 pokrenut je 2007. s velikim brojem novih značajki. Neki od glavnih značajki su optimizirani terenski motor za detaljna 3D okruženja (eng. optimized terrain engine for detailed 3D environments), dinamičke sjene, usmjerena svjetla, osvjetljenja, reprodukcija videa i dodan je mrežni sloj koji je olakšao rad na multiplay-er igricama. Unity 3.0 pokrenut je 2010. koji je pretežno poboljšao grafičke značajke za desktop računala i konzole. Uz to dodana je podrška za Android. U Unity 4.0 dodana je podrška za DirectX 11 i Adobe Flash, novi animacijski alat Mecanim. Unity 5 izašao je u 2015. godini te napravio još jedan korak prema tome da postane univerzalan i pristupačan. Unaprijeđeno je osvjetljenje i audio, preko WebGL developeri su mogli staviti svoje igrice na Web s tim da korisnici odnosno igrači nisu trebali instalirati nikakav plug-in. 2016. godine je došlo do promjene u nazivima verzija te je nakon verzije 5.6 slijedila verzija 2017 koja je dobila naziv na temelju godine izdavanja [1].

Ključni dio izrade igrice predstavljen je skriptama pomoću kojih možete definirati logiku komponenti i dodavati ponašanja. S već napravljenim skriptama (eng. pre-build), rješenje omogućeno od strane Unity-ja možete jednostavno kontrolirati kameru, odnos između vaših elementa u igrici i sistem animacije [2].

3.2. Korisničko sučelje



Slika 1: Korisničko sučelje

Korisničko sučelje se može prilagoditi osobnim potrebama. Ja sam svoje sučelje raspodijelio kao na slici 1 koje se sastoji od: hierarchy, scene, game, inspector, project i console. Game prozor služi da vidimo kako će naša igrica izgledati korisniku koji je igra. Scene je malo detaljnija od Game prozora jer se pomoću njega gledaju stvari koje se ne vide u Game prozoru te zapravo možemo vidjeti sve objekte koje igrica koristi. Hierarchy prozor prikazuje listu objekta koji se nalaze u Scene prozoru. Odabirom na bilo koji objekt dolazimo do detaljnijih informacija o tom objektu koje možemo vidjeti u Inspector prozoru. Project prozor sadrži sve naše datoteke koje mislimo koristiti za izradu igrice. Console prozor nam služi kod testiranja igrice i javlja nam ključne greške zbog kojih igrica ne može funkcionirati [3].

4. Opis igre

4.1. Žanr

Ova igrica spada pod platformer žanr, da budem još točniji ova igrica je primjer igrice beskonačnog trčanja koja spada pod platformer žanr.

Platformer je video igra u kojoj se igra odvija oko igrača koji kontrolira lika koji trči i skače na platforme, podove, izbočine, stepenice i druge predmete koji se pomiču horizontalno ili okomito po zaslonu igre. Prve platformer igrice su razvijene početkom 1980-ih što ih čini jednim od prvih žanra za igrice. Neke od prvih takvih igrica su Space Panic (1980. godina), Donkey Kong (1981. godina) i Mario Bros. Što se tiče zaslona igrice razlikujemo dvije vrste igre na platformi s jednim zaslonom i igra sa zaslonom koji se pomiče vertikalno i horizontalno u odnosu na poziciju igrača tj. lika [4].

Igre na platformi s jednim zaslonom se igraju na jednom zaslonu na kojem se prelaze prepreke nakon kojih se ide na drugi zaslon. Najbolji primjer igre s jednim zaslonom je Donkey Kong gdje se Mario penje, zaobilazi i preskače prepreke kako bi došao do cilja. Nakon što dođe do cilja igrač prelazi na drugi zaslon te igrica postaje teža sa svakim novim zaslonom [4].

Platformer igra sa zaslonom koji se pomiče po vertikalnoj i horizontalnoj osi pomiče zaslon u skladu s igračem te tako prati igrača. Ovakav tip igre sadrži više razina. Igrač je nužan putovati preko zaslona i ispuniti određeni cilj kako bi mogao otići na sljedeću razinu. Brojne ovakve igrice imaju „šefove“ na kraju razine koje trebaju pobijediti kako bi mogli nastaviti na sljedeću razinu. Pod ovakvu skupinu igara najpoznatije su Super Mario Bros i Sonic The Hedgehog. Kako je grafika postala naprednija tako su i video igre postale naprednije. Popularnost platformera se znatno smanjila krajem 1990-ih [4].

Platformer beskonačnog trčanja je oblik igre u kojoj se lik igrača uvijek kreće naprijed. Popularni primjer ovakve igrice je Canabalt u kojoj igrač koristi samo jedan gumb za preskakanje praznina i prepreka [5].

U ovom odjeljku ću spomenuti neke popularne platformere beskonačnog trčanja na Androidu. Jetpack Joyride je igrica koja sadrži sve komponente ovakvog žanra. Lik u igrici se neprestano kreće prema naprijed te igrač pritiskom na ekran aktivira njegov ruksak za letenje

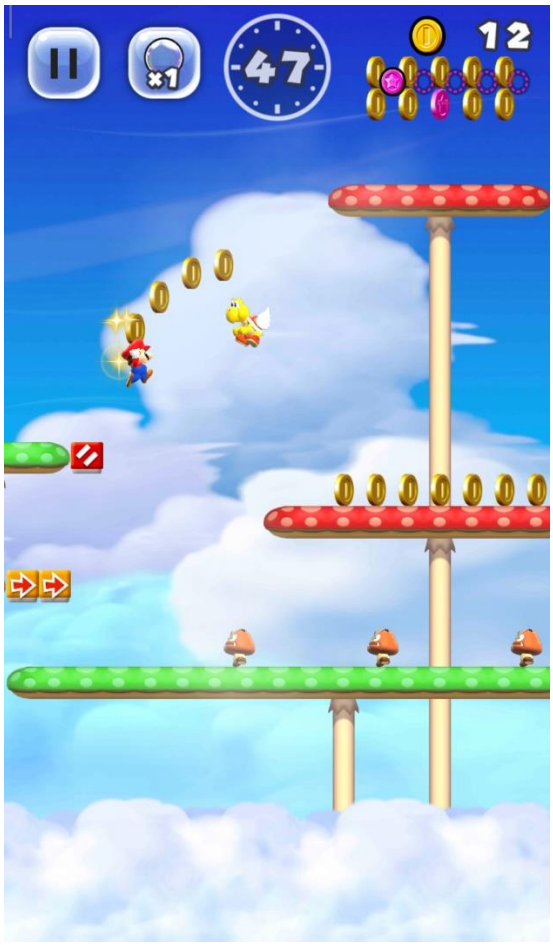
koji mu omogućuje letenje odnosno kretanje prema gore. Igrač treba balansirati letenje kako bi pravovremeno izbjegao prepreke koje mogu biti postavljene na bilo kojoj visini zaslona. Sliku zaslona ove igrice možete vidjeti na slici 1. U gornjem lijevom kutu nalaze se svi važni podaci vezani za trenutni rezultat igrača, u gornjem desnom kutu se nalazi tipka za pauzu i opcije igrice. Ova igrica ima više od 100 milijuna preuzimanja na Play Store-u [6]. Ovakve igrice su specifične zbog svoje jednostavnosti što se tiče kontrola. Igrač može samo upravljati pomicanjem lika prema gore, lik nikad ne stoji na istoj visini već kad ruksak za letenje ne radi lik pada prema podu.

Slika 2: Jetpack Joyride

Rayman Adventures je igrica napravljena za Android i iOS uređaje. Ova igrica je izašla 3. prosinca 2015 godine od strane Ubisoft-a. Besplatna je, ali sadrži mikrotransakcije [7]. Rayman Adventures je auto trkač, što bi značilo da nema kontrole za trčanje već stalno trči. U ovoj igrici nije zadan samo jedan smjer trčanja, mi kao igrači biramo temopo igre i prelaskom preko ekrana mijenjamo smjer kretanja. Igrica se bazira na spašavanju Incrediballsa. Ta čudna stvorenja pomažu Raymanu da izraste što veće stablo. Igrač može pozvati Incrediballs-e da mu pomognu na trenutnoj razini u igrici. Igrica se dijeli na tri tipa igranja: istraživanje, borba i sakupljanje [8]. Na slici 3 vidimo kako izgleda igrica, glavni lik Rayman se nalazi na lijevoj strani ekrana. Kod ove igrice mi se izrazito dopala grafika koja uvijek izgleda nova i svježa .

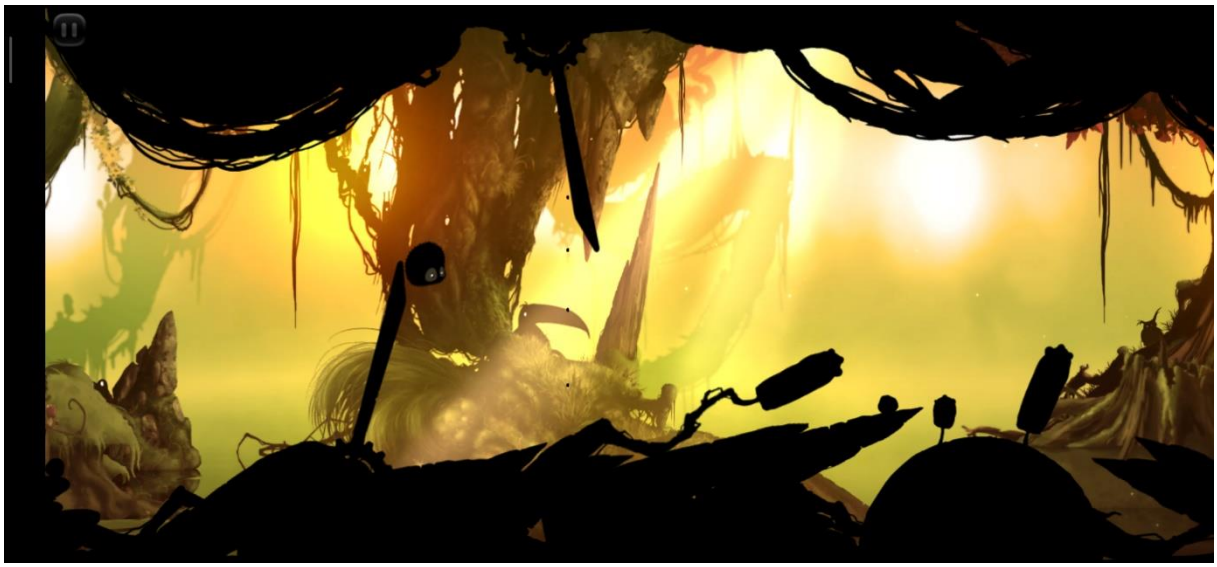
Slika 3: Rayman Advetures

Super Mario Run igricu je izradio Nintendo koja nije besplatna nego se plaća ($9.99) [10]. Ovo je Nintendo-va prva igrica na mobilnim platformama. Super Mario Run je izašla 15. prosinca 2016. godine za iOS i kasnije za Android. Ona se sastoji od tri tipa igranja: Tour, Rally i Build. Kontrole igrice su jednostavne, pritiskom na ekran Mario skoči, što dulje držimo to će on više skočiti. Likovi (možemo birati između više likova, a ne samo Maria) samo trče prema naprijed (desno) i preskaču male prepreke i neprijatelje bez korisničkog unosa [11].



Slika 4: Super Mario Run

Badland je malo starija igrica koja je izašla 2013. godine te je pokupila brojne nagrade [12]. U ovoj igrici igrač kontrolira jednog stanovnika šume koji pokušava saznati što se događa. Upravljanje je jednostavno kao kod svih igrica ovog podžanra. Pritiskom na ekran to malo stvorenje zamahne krilima i poleti. Dok skupljate nove stanovnike šume njima upravljate kao roj. Kako bi prošli razinu samo jedan treba doći do kraja, a ostale ćete vjerojatno žrtvovati i izgubiti putem (zato se ne smijete vezati za njih) [13].



Slika 4: Badland

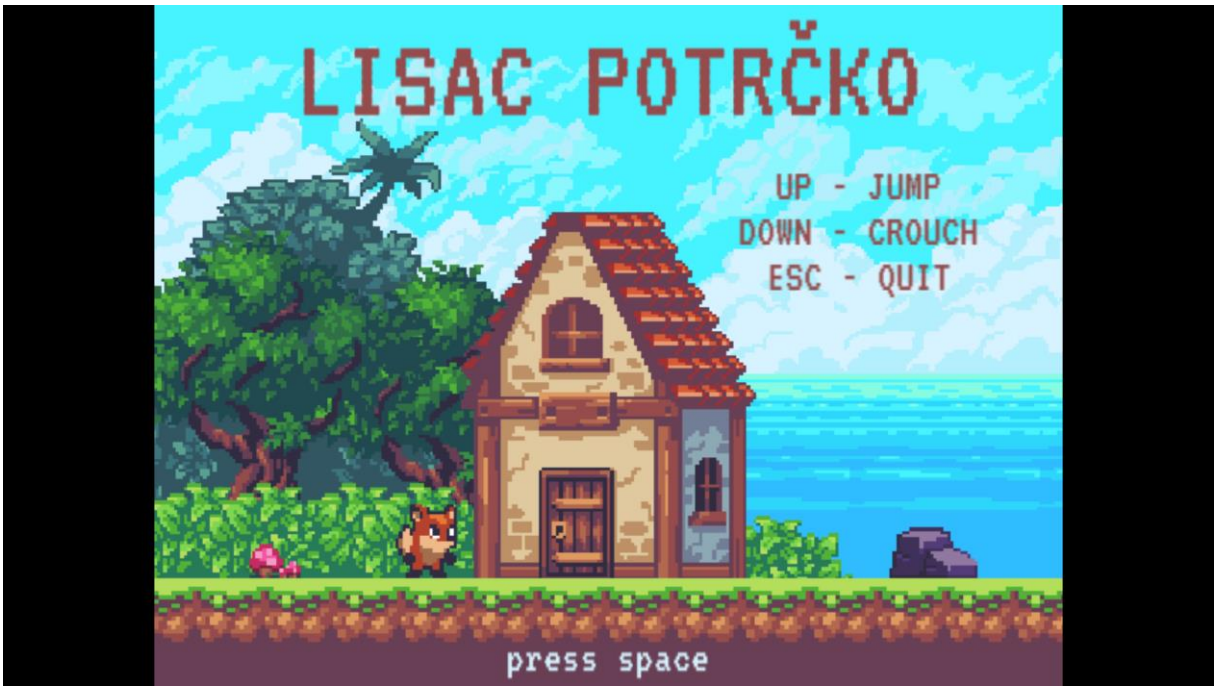
King of Thieves je još jedan zanimljivi primjer ovog podžanra u kojem je mehanika vrlo jednostavna, ali opet zahtjevna [14]. Igrate kao crna mrlja (kockica) koja se kreće automatski, a pritiskom na ekran on skači. Ako dođe do zida i skoči onda promjeni smjer kretanja. Ta funkcija mijenjanja smjera preko zida omogućuje brzo penjanje i kretanje cik-cak tempom sa zida na zid. Na ravnoj površini kreće se u jednom smjeru dok ne dosegne zamku, ne umre ili stigne do zida i zaustavi se. Igra zahtijeva savršeno tempirane skokove i dobru taktiku s vaše strane [16].

Slika 5: King of Thieves

4.2. Lisac Potrčko

U ovom poglavlju ukratko ću opisati funkcionalnosti igrice radi lakšeg razumijevanja sljedećeg poglavlja u kojem ću pretežito objašnjavati proces izrade igrice.

Igrica se sastoji od dvije scene. Prva scena se prikazuje pri našem otvaranju igrice, radi lakšeg razumijevanja zvat ćemo je „Meni“. Dok druga scena reprezentira samu igricu odnosno „Gameplay“.



Slika 6: Meni scena

Meni (slika 6) je zadužen za kratak opis kontrola igrice. Pritiskom na tipku Space prebacujemo se s Meni scene u Gameplay scenu. Izlazak iz igrice je moguć u bilo kojem trenutku pritiskom na tipku Esc. Lisica skače kad pritisnemo tipku Space ili strelicu prema gore. Lisica čučne pritiskom tipke strelice prema dolje.

Slika 7: Gamaplay scena

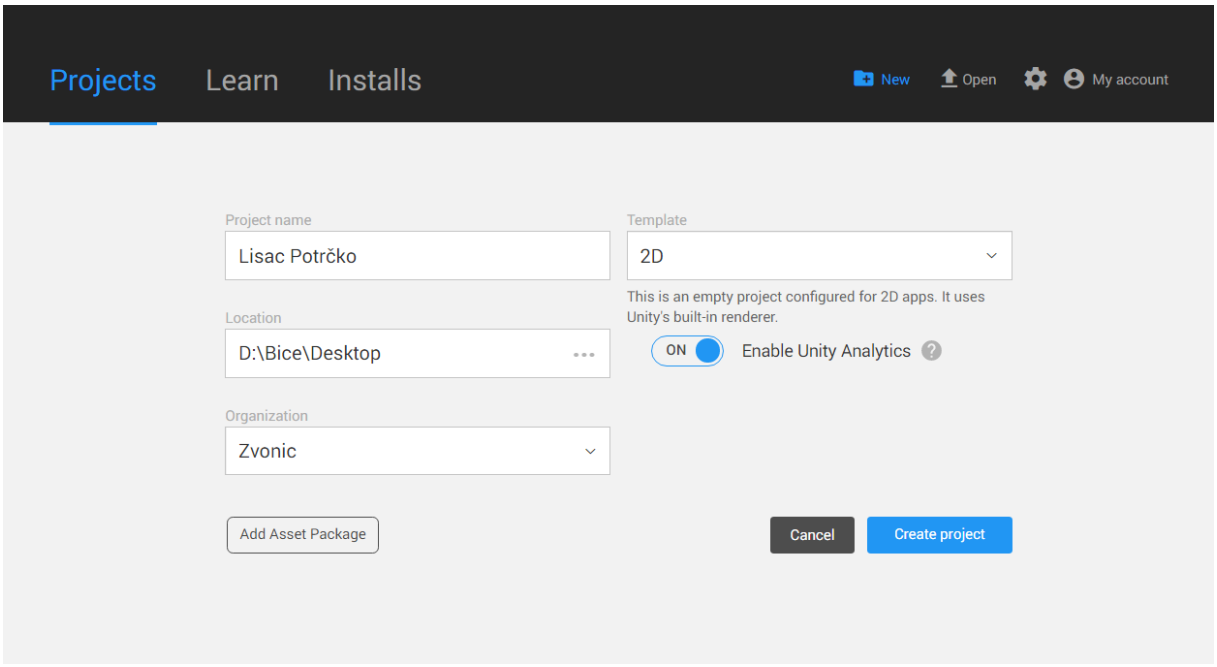
Nakon što smo pritisnuli Space tipku izašli smo iz Menija te smo ušli u Gameplay dio igrice. Lisica trči prema desno te neprestano treba preskakati ili zaobići (čučnuti) prepreke. Ako dođe u kontakt s preprekama dolazi do kraja igrice te izlazimo iz Gameplay scene i ulazimo ponovno na Meni igrice. Tijekom neprestanog trčanja lisica nailazi na četiri vrste prepreka, a to su žaba, orao, oposum i bodlje. Uz prepreke lisica može naići na dijamant i voćkicu odnosno trešnju. Ako dođe u kontakt s dijamantom ili voćkicom dobiva bodove koji se mogu vidjeti u gornjem lijevom kutu pod „Score:“.

5. Izrada igre

U ovom poglavlju ću ukratko opisati i objasniti kako sam napravio svoju igricu. Prvo ću proći kroz osnove kreiranja novog projekta te nakon toga sam igricu podijelio u dva dijela odnosno podijelio prema scenama: Meni scena i Gameplay scena. Meni scena je jednostavnija te ću krenuti s njom i završiti s Gameplay scenom koja je zapravo srž ove igrice kao što smo vidjeli iz prethodnog poglavlja.

5.1. Kreiranje projekta

Prilikom prvog pokretanja Unity alata trebamo napraviti novi projekt te nam se ubrzo otvara prozor kao na slici 8.



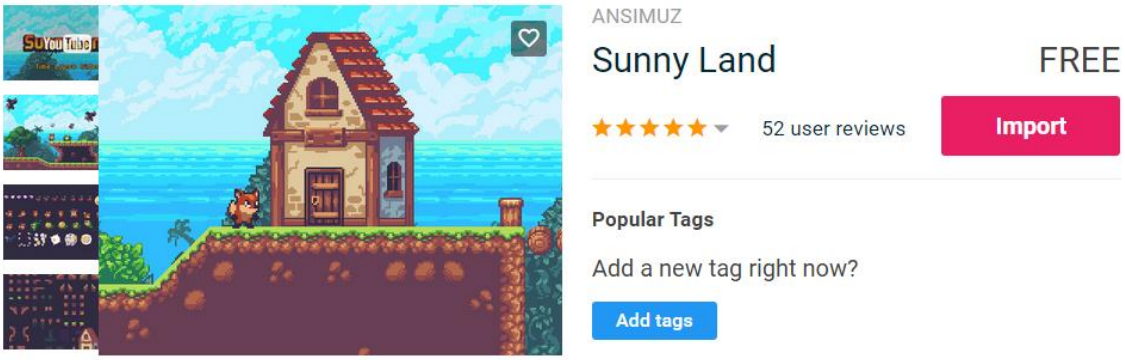
Slika 8: Novi projekt

Biramo naziv projekta odnosno igrice, lokaciju i naravno odabiremo 2D template koji se poslije može još mijenjati. Zahvaljujući 2D template-u sučelje i pojedine funkcionalnosti su

prilagođene dvodimenzionalnom prostoru. Dodavanje paketa „Add Asset Package“ ćemo koristiti kad se projekt kreira.

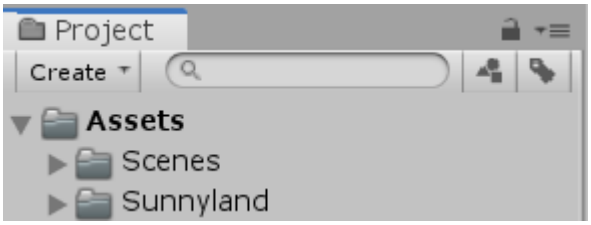
5.1.1. Uvoz paketa

Unity nam omogućuje pristup Unity store-u koji sadrži velik broj već napravljenih pomagala. Kako sam ne bi trebao napraviti svaku sliku koja je korištena u igrici, jednostavno ću uvesti paket (eng. „Import Asset Package“) koji sadrži meni potrebne slike. Ja ću uvesti paket koji se zove Sunny Land (slika 9) [17].



Slika 9: Asset store

Uvoz paketa je uspješno obavljen kao što možemo vidjeti na slici 10. U project prozoru se nalaze sve datoteke iz paketa Sunny Land te su spremni za korištenje. U nastavku ću početi s kreiranjem Meni scene.



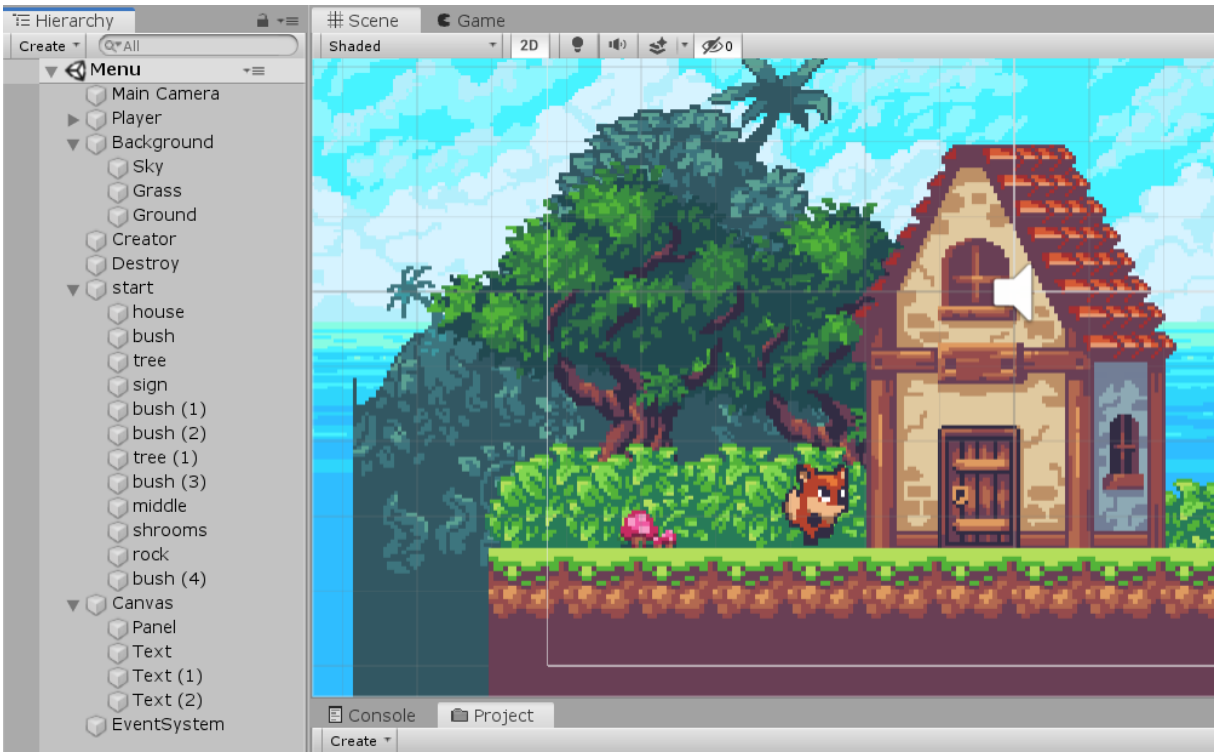
Slika 10: Project prozor

5.2. Meni scena

Meni je prva stvar koja je vidljiva prilikom otvaranja igrice, znatno je jednostavnija od glavnog dijela igrice u kojoj igrač aktivno utječe na ishod lisice. Zbog toga sam odlučio prvo objasniti kreiranje Meni scene (slika 6).

5.2.1. Pozadina i okolina

Vizualni aspekt Meni scene se sastoji točno od 16 objekta uključujući i glavnog lika ove igrice (lisica). Slike (eng. Sprites) su postavljene u scene prozoru kako bi igrica lijepo izgledala. U hierarchy prozoru je prikazano koji su sve objekti korišteni u Meni sceni (slika 11, lijeva strana). Na desnoj strani slike 7 je scene prozor koji nam pokazuje što se događa „iza kamere“.



Slika 11: Kreiranje prve scene

Kako bi dobili željeni poredak slika npr. stablo se nalazi iza grmlja, lisica se nalazi ispred svega itd. koristimo Sorting layers i poziciju objekta na z osi. Napravio sam 4 sloja (eng. Layer, slika 12, lijeva strana) koji su redom Background, Ground, Enemy i Player. Ako dva objekta koriste isti sloj, a nama se ne sviđa njihovo preklapanje mijenjamo red u sloju (eng. Order in Layer, slika 12, desna strana). Ako je red u sloju veći onda se on prikazuje ispred objekta koji ima isti sloj i manji red u sloju. Sloj i red u sloju odabiremo u Inspector prozoru (slika 1).

Slika 12: Sorting layers i pozicija objekta

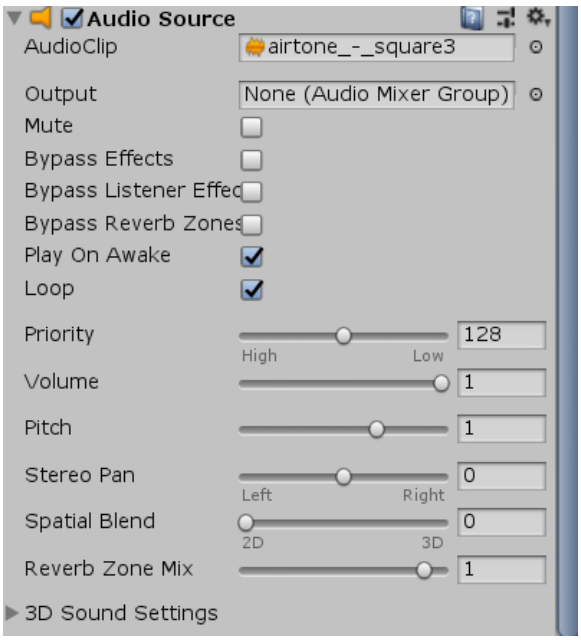
5.2.2. Animacija

Na Meni sceni nema previše događanja te koristim samo jednu animaciju, a to je animacija na glavnom liku lisici. Animacija prikazuje kako lisica diše te daje prirodnu i živu atmosferu koja definitivno čini ovu scenu zanimljivijom. Kod korištenja animacije potrebno je dodati komponentu Animator na objekt lisicu. Uz to još trebamo napraviti animaciju (eng. Animation) i animacijski kontroler (eng. Animation Controller). Više o animacijama u sljedećem poglavlju gdje ćemo promatrati animaciju kretanja lisice.

5.2.3. Audio

Na glavnoj kameri (eng. Main camera) dodao sam komponentu Audio Source koja omogućuje puštanje muzike (slika 13). U polje AudioClip stavio sam željenu muziku koju želim čuti kad upalim igricu. Pomoću bool varijable Loop osiguravam da se muzika neprestano svira, da bude u petlji. Play on Awake varijabla omogućuje da muzika počne

svirati prilikom kreiranja objekta u sceni, a glavna kamera je objekt koji se nalazi u sceni cijelo vrijeme te tako muzika počne svirati svaki put kad pristupimo Meni sceni.



Slika 13: Audio Source komponenta

5.2.4. Canvas

Canvas je područje u kojem bi trebali biti svi elementi korisničkog sučelja. Ja sam u svoj canvas stavio tri tekst objekta. Prvi se nalazi na vrhu ekrana: „Lisac Potrčko“. Drugi je s desne strane i objašnjava kako se igra i kako se izlazi iz igrice. Treći se nalazi na dnu ekrana „press Space“ koji informira igrača da pritiskom na tipku Space lisica počinje trčati.

5.2.5. Skripte

Igrica je zadana u rezoluciji 640x480. Skripta koja zadaje rezoluciju stavljena je na glavnu kameru tako da se igrica istog trenutka prilikom otvaranja pokreće u rezoluciji 640x480. Skripta se nalazi u Start funkciji te se poziva samo jednom.

```
void Start()
{
      Screen.SetResolution(640, 480, true);
}
```

U Meni sceni čekamo korisnika da pritisne tipku Space. Ako korisnik pritisne tipku Esc, igrica se zatvara.

```
private void Update()
    {
        if (Input.GetButtonDown("Jump"))
        {
            SceneManager.LoadScene(1);
        }
        if (Input.GetButtonDown("Escape"))
        {
            Application.Quit();
        }
    }
```

5.3. Gameplay scena

Gameplay scena je tehnički glavna scena ove igrice. Ona se sastoji od fiksne pozadine koja daje iluziju da se pomiče tj. da lisica trči prema desno. Lisica ima fiksnu x koordinatu te uz animaciju trčanja i pozadinu izgleda kao da se x koordinata stalno povećava. U nastavku ovog poglavlja ću pokriti cijelu scenu Gameplay.

5.3.1. Pozadina

Pozadina odnosno nebo, zemlja i trava se cijelo vrijeme nalaze na fiksnoj poziciji. Objekti pozadine su objekti na koje sam dodao komponentu materijal i tako mogu manipulirati offset varijablom koja omogućuje da zapravo pozadina stoji na istoj poziciji u prostoru. Jedino što mijenjanjem offset-a se slika pomiče po objektu. I za to je korištena sljedeća skripta koja se nalazi na nebu, zemlji i travi. Pošto je varijabla xVelocity public ja sam zadajem brzinu kojom želim da se pozadina „pomiče“. Da bi trčanje izgledalo što prirodnije nebo se pomiče sporije od trave i zemlje (xVelocity je 0.2 za nebo te 60 za zemlju i travu).

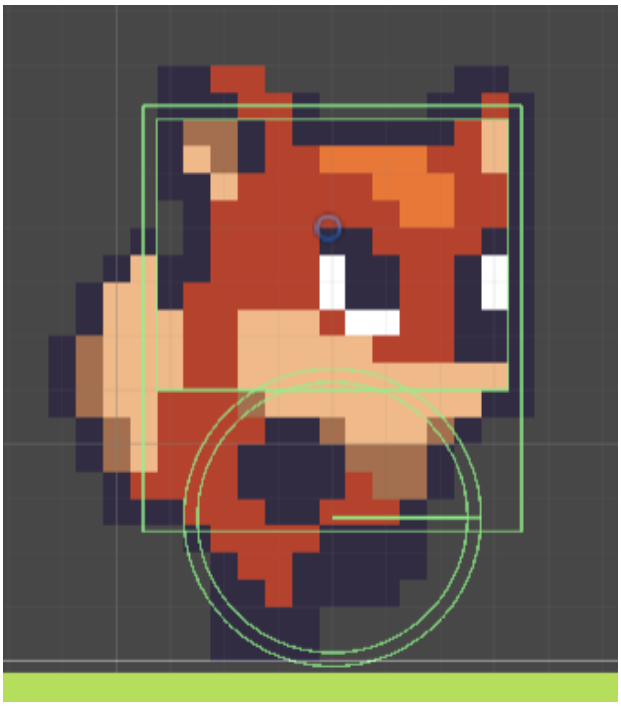
```
    Material material;
    Vector2 offset;
    public float xVelocity;
```

```
private void Awake()
{
    material = GetComponent<Renderer>().material;
}
void Update()
{
    offset = new Vector2(xVelocity/10, 0);
    material.mainTextureOffset += offset * Time.deltaTime;
}
```

Isto kao i u prethodnom poglavlju na glavnoj kameri sam stavio komponentu AudioClip za muziku u pozadini koja je drugačija od muzike u Meni sceni. Te odmah nakon prijelaza u Gameplay scenu počinje nova muzika.

5.3.2. Lisica (audio, animacija, kretanje)

Lisica je glavni lik ove igrice koji trči prema naprijed, preskače prepreke i skuplja voćkice i dijamante. Kao što sam već naveo lisica tehnički stoji na mjestu. Lisica može skočiti i čučnuti. Lisica se sastoji od četiri collider2D komponente, dvije su okidači koji bilježe koliziju s drugim objektima (slika 14). Lisica ima mogućnost duplog skoka (eng. double jump). Dok je lisica u zraku igrač ima mogućnost još jednom skočiti.



Slika 14: Lisica i Collider2D

Koristim dva okidača jer kad lisica čučne njena veličina se smanji pa stoga koristim jedan okidač za gornji dio tijela i jedan za donji dio tijela. Prilikom čučanja okidač za gornji dio tijela se isključi kako ne bi davao krive rezultate. Okidač koji se nalazi na donjem dijelu uvijek ostaje uključen. Druga dva collider-a sam stavio kako lisica ne bi bila samo slika već kako bi imala svoje tijelo tj. ne bi propala kroz zemlju.

Kod lisice koristim dvije skripte, u jednoj je definiran skok i čučanj dok u drugoj čekam unos igrača te pozivam prvu skriptu [18]. U drugoj skripti su isto definirani postupci prilikom kolizije s preprekama i drugim objektima. Stoga ću tu skriptu podijeliti na dva dijela. Čekanje unosa igrača i pozivanje druge skripte.

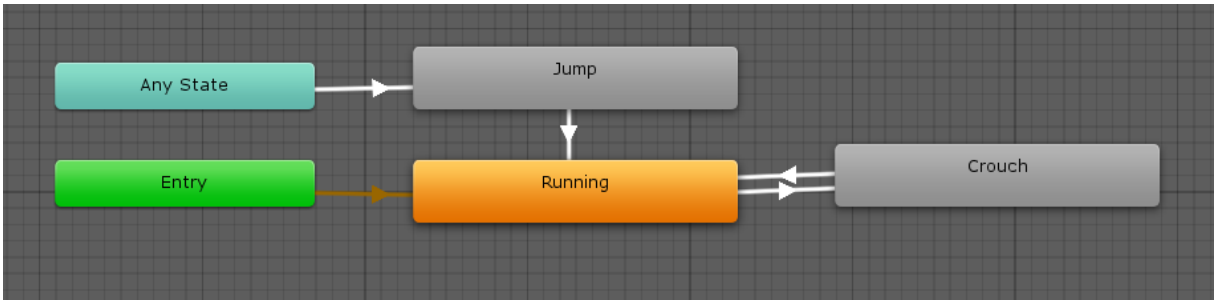
```
bool jump = false;
bool crouch = false;
void Update()
{
    if (Input.GetButtonDown("Jump"))
    {
        jump = true;
```

```
            animator.SetBool("IsJumping", true);
        }
        if (Input.GetButtonDown("Crouch"))
        {
            crouch = true;
        }
        else if (Input.GetButtonUp("Crouch"))
        {
            crouch = false;
        }
        if (Input.GetButtonDown("Escape"))
        {
            Application.Quit();
        }
    }
    public void OnLanding()
    {
        animator.SetBool("IsJumping", false);
    }
    public void OnCrouching(bool isCrouching)
    {
        animator.SetBool("IsCrouching", isCrouching);
    }
    void FixedUpdate()
    {
        controller.Move(0, crouch, jump);
        jump = false;
    }
```

Igrač može izaći iz igrice u bilo kojem trenutku pa stoga čekam ako pritisne tipku Esc kako bi se igrica ugasila. To je vidljivo u Update funkciji, a komanda koja omogućuje zatvaranje igrice je Aplication.Quit().

Kretanje ne bi bilo fluidno bez animacije. Lisica u ovoj sceni ima tri animacije trčanje, skakanje i čučanje. Animacija trčanja je zadana na početku i radi bez prestanka. Iznimke su

kad lisica čuči ili skače. Tada se prekida animacija trčanja i stavlja animacija koja je potrebna.



Slika 15: Kontroler animacija

Kao što je vidljivo na kontroleru animacija (slika 15) animacija skoka se može aktivirati u bilo kojem trenutku dok u animaciju čučnja možemo prijeći samo iz trčanja. Animacija skakanja i čučanja traju određeno vrijeme te se vrate na zadanu animaciju odnosno trčanje.

5.3.3. Prepreke

Prepreke su motivacija lisici da skače i čuči. Ova igrica ima četiri prepreke, a to su žaba, orao, oposum i bodlje. Žaba i orao se kreću istom brzinom dok se oposum kreće brže od njih.

Slika 16: Žaba, orao i oposum

Kako što vidimo na slici 16 orao leti iznad zemlje. Lisica može preskočiti orla, ali to nije preporučljivo jer je znatno lakše čučnuti i proći ispod njega.

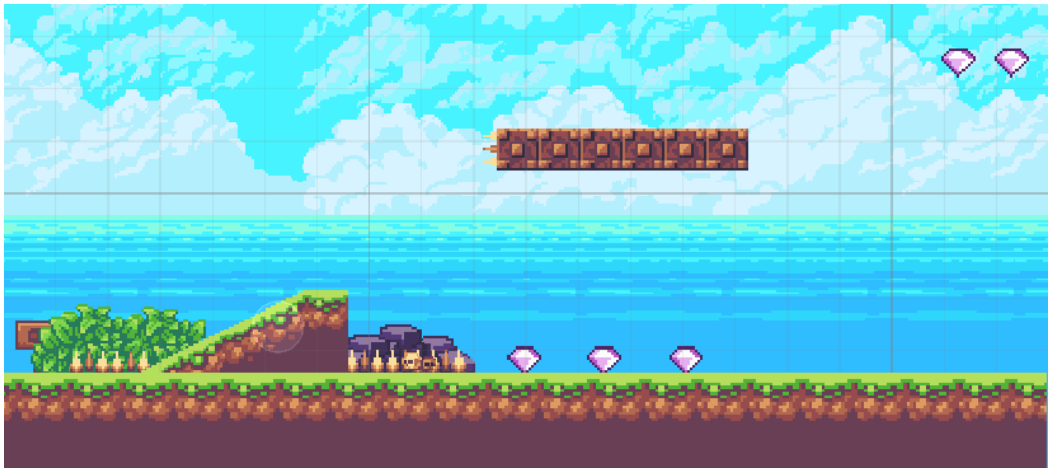
Svaka prepreka (žaba, orao i oposum) ima skriptu na sebi. Skripta služi za pomicanje prepreke prema lijevo jer igrica funkcionira tako da lisica stoji na istom mjestu i pozadina se isto ne pomiče. Stoga se prepreka pomiče prema lijevo.

```
        public float yAxis;
        public float speed = 20f;
        public Rigidbody2D rb;

        void Start()
        {
            transform.position = new Vector2(transform.position.x,
yAxis);
            rb.velocity = transform.right * (-speed);
        }
```

Pomicanje je public varijabla što znači da je možemo mijenjati po želji u Unity Inspector prozoru te tako zadati brzinu oposuma veću od brzine žabe i orla.

Prepreke su živa stvorenja stoga na njih trebam staviti animacije. Svaka prepreka (osim bodlji) ima svoju zadanu animaciju. Animacija je napravljena isto kao i animacija na lisici u Meni sceni. Prilikom dodira lisice s preprekama dolazi do kraja igre o čemu ću više na kraju poglavlja. Uz ove „žive“ prepreke tu su još i bodlje. Bodlje stoje na mjestu i one su napravljene u sklopu malog poligona (slika 17) koji igraču daje dvije opcije. Poligon se sastoji bodlji na tri lokacije, prvo na početku odnosno prije početka penjanja na prvu platformu, zatim odmah iza prve platforme te treća lokacija koja je za razliku od prve dvije rotirana za 90 stupnjeva i nalazi se na bočnoj strani druge platforme. Igrač može odrediti lakši put ili teži put koji će ga nagraditi s tri dijamanta. Lakši put je napravljen tako da igrač odmah treba skočiti na drugu platformu. Teži put je put ispod druge platforme.



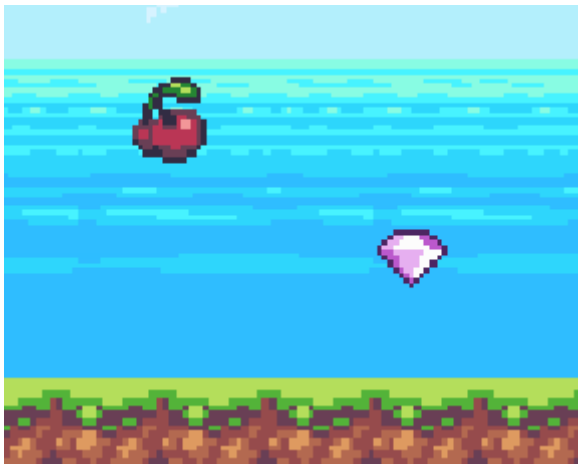
Slika 17: Poligon s bodljama

Uz navedeni poligon u igrici postoji još jedan poligon na kojeg možemo doći jedino pomoću duplog skoka. Napravljen je tako da ima bodlje s prednje strane i ne možemo skočiti unutar platforme jer će lisica udariti glavom i umrijeti.

Slika 18: Plaftorma za dupli skok

5.3.4. Voćkice i dijamanti

Igrač treba sakupljati dijamante i voćkice kako bi ostvario što bolji rezultat. Na svaku voćkicu i dijamant je dodana komponenta Animator isto kao i na lisici u Meni sceni. Uz zadanu animaciju voćkice i dijamanti imaju svoju skriptu koja funkcionira na isti način kao skripta za prepreke, skripta pomiče voćkice i dijamante prema lisici tj. prema lijevo.



Slika 19: Voćkica i dijamant

Voćkica vrijedi kao jedan bod, a dijamant vrijedi kao dva boda.

5.3.5. Kreiranje objekta

Kreiranje objekta (prepreka, dijamanta i voćkica) se odvija izvan glavne kamere te koristi funkciju spawn() za stvaranje objekta koja ima za argument broj objekta.

```
void spawn(int number)
{
switch (number)
    {
    case 1:
       Instantiate(enemy1, transform.position, Quaternion.identity);
       break;
    case 2:
       Instantiate(enemy2, transform.position, Quaternion.identity);
       break;
    case 3:
        Instantiate(coin2, transform.position, Quaternion.identity);
        break;
    case 4:
        Instantiate(enemy4, transform.position, Quaternion.identity);
        break;
    case 5:
```

```
            Instantiate(coin1, transform.position, Quaternion.identity);
             break;
        case 6:
            Instantiate(enemy3, transform.position, Quaternion.identity);
            break;
    }
}
```

Igrica funkcionira tako da se nakon zadanog fiksnog vremenskog perioda stvori žaba, orao ili voćkica. Svakih 15 sekunda se za redom stvaraju oposum pa dijamant, sve ukupno pet oposuma i pet dijamanta. Što čini igricu zanimljivijom i raznolikom.

```
IEnumerator enemyWave()
    {
        while (true)
        {
            if(special)
            {
                yield return new WaitForSeconds(respawnTime);
                for (int i = 0; i < 5; i++)
                {
                    yield return new WaitForSeconds(respawnTime / 2);
                    spawn(6);
                    yield return new WaitForSeconds(respawnTime / 2);
                    spawn(5);
                }
                special = false;
            }
            else
            {
                yield return new WaitForSeconds(respawnTime);
                num = Random.Range(1, 5);
                if (num == 4)
                {
                    yield return new WaitForSeconds(respawnTime);
                }
```

```
                    spawn(num);
                    if (num == 4)
                    {
                        yield return new WaitForSeconds(2*respawnTime);
                        spawn(2);
                    }
                }
            }
        }
```

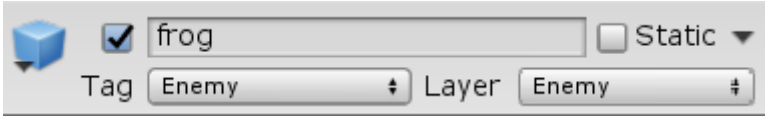
Bool varijabla special postaje true kad prođe 15 sekunda. Tada ulazimo u petlju koja neprestano pet puta stvara oposuma i dijamant, s pola sekunde između svakog. Pri izlasku iz for petlje varijabla special se postavlja na false. Specijal je false dok ne prođe sljedećih 15 sekunda. Za to vrijeme se svake sekunde stvara po jedan objekt žaba, orao, poligon bodlji ili voćkica. Za mjerenje vremena koristio sam Time.deltaTime u funkciji Update.

5.3.6. Uništavanje objekta

Uništavanje objekta je nužno jer nakon što orao izađe iz ekrana naše igrice samo otežava rad igrice. Ovaj objekt za uništavanje se nalazi s lijeve strane izvan okvira glavne kamere. Te uništava sve kreirane objekte s kojim dođe u kontakt. Okidač vraća objekt s kojim je došao u kontakt. Taj objekt ima svoju oznaku.

```
        private void OnTriggerEnter2D(Collider2D collider)
        {
            if (collider.tag == "Enemy" || collider.tag == "Coin1" ||
collider.tag == "Coin2" || collider.tag == "Destroy")
            {
                Destroy(collider.gameObject);
            }
        }
```

Skripta čeka koliziju s objektom te provjerava je li to objekt koji može uništiti. To je izvedivo pomoću tag opcije koju stavljamo na instancu svakog objekta. Napravio sam tri oznake (eng. tag) coin1, coin2 i enemy oznaku. Oznaku zadajem objektu u inspector prozoru (slika 20).



Slika 20: Oznaka

5.3.7. Kolizija lisice s objektima

Lisica sa svojim okidačem čeka koliziju s drugim objektima. Pomoću oznake na svakom objektu s kojim može doći u kontakt okidač razaznaje koji postupak slijedi. Voćkice i dijamant se razlikuju po broju bodova koje donose. Zbog toga je potrebno napraviti posebnu oznaku za voćkicu i dijamant. Njihove oznake su coin1 i coin2. Prilikom dodira voćkice ili dijamanta, objekt voćkice i dijamanta se uništava i stvara se animacija koja reprezentira pokupljeni objekt (voćkica ili dijamant). Ta animacija ima kratak vijek trajanja.

ScoreMenager je skripta koja zbraja ostvarene bodove pa se tako u slučaju kad lisica pokupi vočkicu (coin1) pridoda jedan bod, a u slučaju kad lisica pokupi dijamant (coin2) pridodaju dva boda.

```
        private void OnTriggerEnter2D(Collider2D collider)
        {
            if (collider.tag == "Enemy")
            {
                gameObject.GetComponent<Renderer>().enabled = false;
                GameObject efekt = Instantiate(deathEffect,
transform.position, Quaternion.identity);
                Destroy(efekt.gameObject, timeDestroyDeath);
                StartCoroutine(wait());
            }
            if (collider.tag == "Coin1")
            {
                GameObject efekt = Instantiate(coinEffect,
collider.transform.position, Quaternion.identity);
                Destroy(collider.gameObject);
                Destroy(efekt.gameObject, timeDestroyCoin);
                ScoreMenager.instance.changeScore(1);
```

```
            }
            if (collider.tag == "Coin2")
            {
                GameObject efekt = Instantiate(coinEffect,
collider.transform.position, Quaternion.identity);
                Destroy(collider.gameObject);
                Destroy(efekt.gameObject, timeDestroyCoin);
                ScoreMenager.instance.changeScore(1);
            }
        }
```

Ako lisica dođe u kontakt s orlom, žabom, oposumom ili bodljama, što provjeravamo s *colider.tag == „Enemy“*. Lisica instantno umire i to je kraj igre. Kako se ne bi odmah vratili na Meni scenu napravio sam čekanje od 1.2 sekunde prije nego što se vratimo u Meni scenu. Tako na primjer kad ne stignemo preskočiti žabu već skočimo na nju. Prvo ne uništavamo objekt lisice već je „sakrivamo“ tj. onemogućujemo komponentu Sprite Renderer koja je zadužena za sliku i izgled lisice. Od tad je lisica samo nevidljivi objekt u Gameplay sceni. Kako bi „smrt“ lisice izgledala zanimljivije stvaram animaciju te je stavljam u varijablu jer ta animacija ima svoj kratak vijek trajanja. Čim animacija prođe ja ju uništim Destroy() funkcijom koja za drugi parametar prima vrijeme nakon kojeg će uništiti objekt. Sad kad se više lisica ne vidi na ekranu kreče čekanje od 1.2 sekunde. Za to koristim korutinu IEnumerator().

```
    IEnumerator wait()
    {
        deathAudio.Play();
        yield return new WaitForSeconds(1.2f);
        SceneManager.LoadScene(0);
        Destroy(gameObject);
    }
```

U varijabli deathAudio je pohranjen zvuk koji se pušta prilikom umiranja lisice. Nakon što prođe 1.2 sekunde koristeći *yield return new WaitForSeconds(1.2);* učitavamo Meni scenu. Tako je ciklus igrice zatvoren i možemo iznova krenuti u avanturu s lisicom.

6. Zaključak

Moja tema završnog rada je izrada igrice. Potrudio sam se što bolje, ali u isto vrijeme bez previše kompliciranja i teorije objasniti kako sam napravio igricu. Prije same izrade igrice prošao sam kroz osnove Unity alata, kreiranje novog projekta i uvoz paketa. Uz to važno je razumjeti pravila i funkcije igrice koje sam objasnio u petom poglavlju Opis igrice. Moja igrica je podijeljena u dvije scene Meni i Gameplay. To mi je bila glavna podjela igrice, pa sam tako i napravio u ovom radu. Prvo sam objasnio izrazito jednostavnu scenu Meni koja se sastoji od pozadine i okoline, animacije, audija, canvas i vrlo jednostavne skripte. Gamplay scena je zapravo glavni dio igrice. Nju sam podijelio na pozadinu koja se pomiče, glavnog lika lisice, prepreka, voćkica i dijamanta, kreiranje objekta, uništavanje objekta i naravno kolizija objekta s glavnim likom lisicom. Na kraju mogu reći da mi je drago što sam uzeo igricu u Unity alatu za završni rad. Nisam siguran hoću li se u budućnosti profesionalno baviti izradom igrica, ali sam zato siguran da je ovo bilo korisno iskustvo iz kojeg sam izašao s više znanja.

7. Popis literature

[1] Wikipedija, Unity(game engine), 2019. [na internetu]. Dostupno: https://en.wikipedia.org/wiki/Unity_(game_engine) [pristupano 30.7.2019]

[2] Unity, 2019. [na internetu]. Dostupno: https://unity3d.com/what-is-a-game-engine [pristupano 29.8.2019]

[3] Unity Documentation, Learning the interface, 2019. [na internetu], Dostupno: https://docs.unity3d.com/Manual/LearningtheInterface.html [pristupano 9.9.2019]

[4] Lifewire, 2019. [na internetu]. Dostupno: https://www.lifewire.com/what-is-a-platform-game-812371 [pristupano 29.8.2019]

[5] Moby Games, 2019. [na internetu]. Dostupno: https://www.mobygames.com/game- group/genre-auto-run-platformer [pristupano 29.8.2019]

[6] Google Play - Jetpack Joyride, 2019. [na internetu]. Dostupno: https://play.google.com/store/apps/details?id=com.halfbrick.jetpackjoyride&hl=hr [pristupano 29.8.2019]

[7] Google Play, Rayman Adventures, 2019. [na internetu]. Dostupno: https://play.google.com/store/apps/details?id=com.ubisoft.raymanadventures&hl=en [pristupano 9.9.2019]

[8] TouchArcade, 'Rayman Adventures' Review – Incrediball, 2019. [na internetu]. Dostupno: https://toucharcade.com/2015/12/11/rayman-adventures-review/ [pristupano 9.9.2019]

[9] DestructoId, Review: Rayman Adventures, 2019. [na internetu], Dostupno: https://www.destructoid.com/review-rayman-adventures-325074.phtml [pristupano 9.9.2019]

[10] Google Play, Super Mario Run, 2019. [na internetu], Dostupno: https://play.google.com/store/apps/details?id=com.nintendo.zara [pristupano 9.9.2019]

[11] DestructoId, Review: Super Mario Run, 2019. [na internetu], Dostupno: https://www.destructoid.com/review-super-mario-run-406523.phtml [pristupano 9.9.2019]

[12] Google Play, Badland, 2019. [na internetu], Dostupno: https://play.google.com/store/apps/details?id=com.frogmind.badland&hl=en [pristupano 9.9.2019]

[13] AndroidCentrar, Badland is still one of the best games for Android, 2019. [na internetu],

Dostupno: https://www.androidcentral.com/badland-classic-review [pristupano 9.9.2019]

[14] Google Play, King of Thieves, 2019. [na internetu], Dostupno: https://play.google.com/store/apps/details?id=com.zeptolab.thieves.google&hl=en [pristupano 9.9.2019]

[15] TouchArcade, 'King of Thieves' Review, 2019. [na internetu], Dostupno: https://toucharcade.com/2015/02/17/king-of-thieves-review/ [pristupano 9.9.2019]

[16] BytesIn, [Review] King of Thieves, 2019. [na internetu], Dostupno: https://www.bytesin.com/review-king-thieves/ [pristupano 9.9.2019]

[17] Unity Asset Store – Sunny Land, 2019. [na internetu]. Dostupno: https://assetstore.unity.com/packages/2d/characters/sunny-land-103349 [pristupano 29.8.2019]

[18] 2D-Character-Controller, 2019. [na internetu]. Dostupno: https://github.com/Brackeys/2D-Character-Controller [pristupano 6.9.2019]

[19] Unity dokumentacija, 2019. [na internetu]. Dostupno: https://docs.unity3d.com/Manual/index.html [pristupano 29.8.2019]

8. Popis slika