Igra igranja uloga s elementima fantasy tematike u Unityu

Sabo, Željko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski studij računarstva

IGRA IGRANJA ULOGA S ELEMENTIMA FANTASY TEMATIKE U UNITYU

Završni rad

Željko Sabo

Osijek, 2018.

Sadržaj

1. UVOD

Tema ovog završnog rada je igra igranja uloga (*engl. Role-Playing Game*) s elementima fantasy tematike u Unity-u [1]. Igre igranja uloga su svoje početke doživjele na samom kraju dvadesetog stoljeća. Ova vrsta igara se po prvi put pojavila za Gameboy igraću konzolu, gdje su najpopularnije igre iz tog perioda bile Pokemon-i i Legend of Zelda. U ovom završnom radu je stavljen veći fokus na izradu igre koja je kao Legend of Zelda, odnosno igre koja ima klasične elemente igara igranja uloga kao što su sistem za izvršavanje zadataka (*engl. quest system*), sistem za pohranu stvari (*engl. inventory system*), sistem za borbu (*engl. combat system*) i ostali. Igra će imati mogućnost odabira između više različitih likova s kojima je moguće igrati igru, a to su vitez (engl. *Knigth* - nadalje korišten termin), čarobnjak (engl. *Wizard* - nadalje korišten termin) i strijelac (engl. *Archer* - te nadalje korišten termin). Svaki lik je jedinstven, a cilj igre je pronaći izlaz te pobijediti zlog vladara kostura zvanog *Skulleton King* koji je zavladao katakombama u kojima se glavni lik Philip (koji može biti *knight*, *wizard* ili *archer*) našao. Philip na svojem putovanju kroz katakombe nailazi na nekolicinu ljudi koji su čuvali katakombe i živjeli u njima dok se nije pojavio zli Skulleton King i odlučio ih protjerati. Oni se zovu *The Keepers* te će upravo oni glavnom liku zadavati zadatke (*engl. quest*) koji će se morati obaviti kako bi se porazio zli Skulleton King i pronašao izlaz iz katakombi. Cijela igra je napravljena unutar više programa. U programu Unity sve je spojeno u funkcionalnu cjelinu, Aseprite [2] je služio za crtanje modela (*engl. sprite*), Bosca Ceoil [3] za kreiranje tematske glazbe. Programski kod koji je potreban da bi sve funkcioniralo je pisan unutar programskog okruženja Microsoft Visual Studio, a programski jezik koji je korišten je C#.

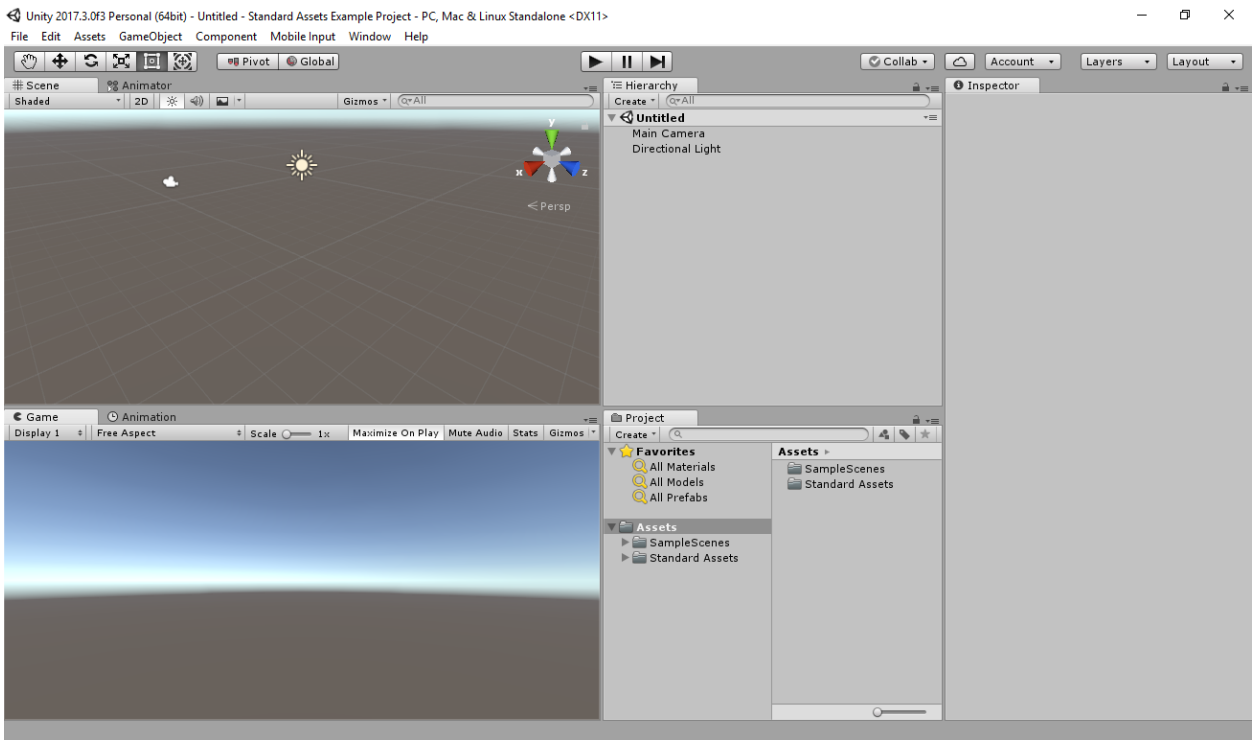
1.1. Zadatak završnoga rada

Zadatak završnog rada je izraditi igru igranja uloga s fantasy tematikom koja ima sve po čemu su klasične igre igranja uloga prepoznatljive. Neke od značajki igri igranja uloga su *quest system*, *inventory system*, *spell system* i mogućnosti odabira glavnog lika između više različitih likova. Predznanja koja su potrebna su osnove programiranja u C#-u, poznavanje rada u Unity-u, osnove rada u Aseprite-u ili bilo kojem programu za crtanje i poznavanje rada u Bosca Ceoli.

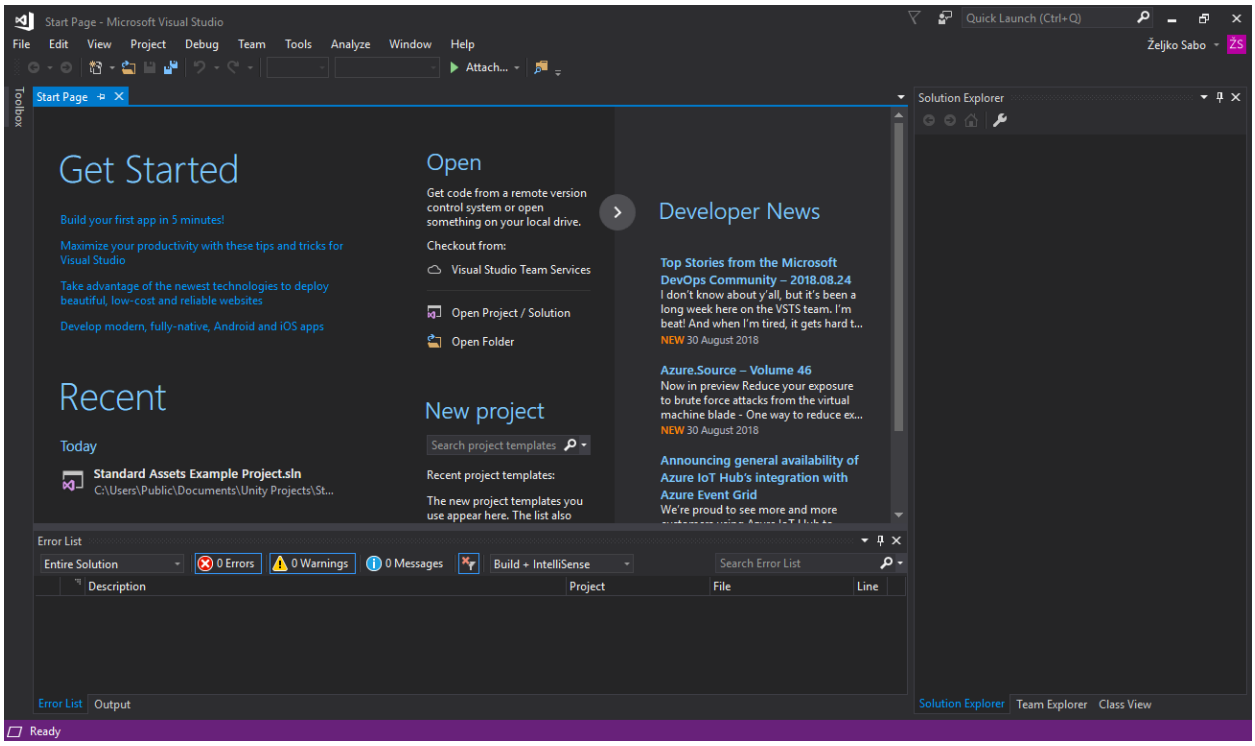
2. KORIŠTENI PROGRAMI

2.1. Unity

Unity je višeplatformski game engine koji omogućava izradu igara i simulacija za više od 27 različitih platformi među kojima su Windows, OSX, Linux, Android, iOS, PlayStation, Nintendo i mnoge druge platforme. Unity je razvijen od strane Unity Technologies, gdje su osnivači David Helgason, Nicholas Francis i Joachim Ante došli na ideju napraviti svoj game engine pošto su tadašnji game engine-ni bili jako skupi i kompliciran za korištenje. U 2005. godini su kao prvu platformu odabrali OSX te su izbacili prvu verziju Unity-a. Do 2008. godine Unity je u potpunosti podržavao OSX i iOS operacijske sustave. U 2010. godini su lansirali Unity Asset Store trgovinu na kojoj se može trgovati raznim asse-tima kao što su 3D modeli, sprite-ovi, animacije za 3D modele, glazba i razni drugi asset-i koji se koriste prilikom razvoja računalnih igara. Unity je od 2004. godine pa do danas imao pet velikih verzija a to su Unity 3.x, Unity 4.x, Unity 5.x Unity 2017.x i Unity 2018.x koji je najnoviji verzija. Sve skripte se mogu uređivati unutar MonoDevelop-a ili Visual Studia koji dolaze s Unity-em, a programski jezici kojima se može služiti prilikom izrade skripti, odnosno programski jezici pomoću kojih se može pisati kod u skriptama su Javascript, Boo i C#. Prilikom izrade završnog rada korišten je isključivo programski jezik C# i Visual Studio Comunity za uređivanje skripti.



Slika 2.1. Izgled otvorenog testnog projekta unutar Unity-a

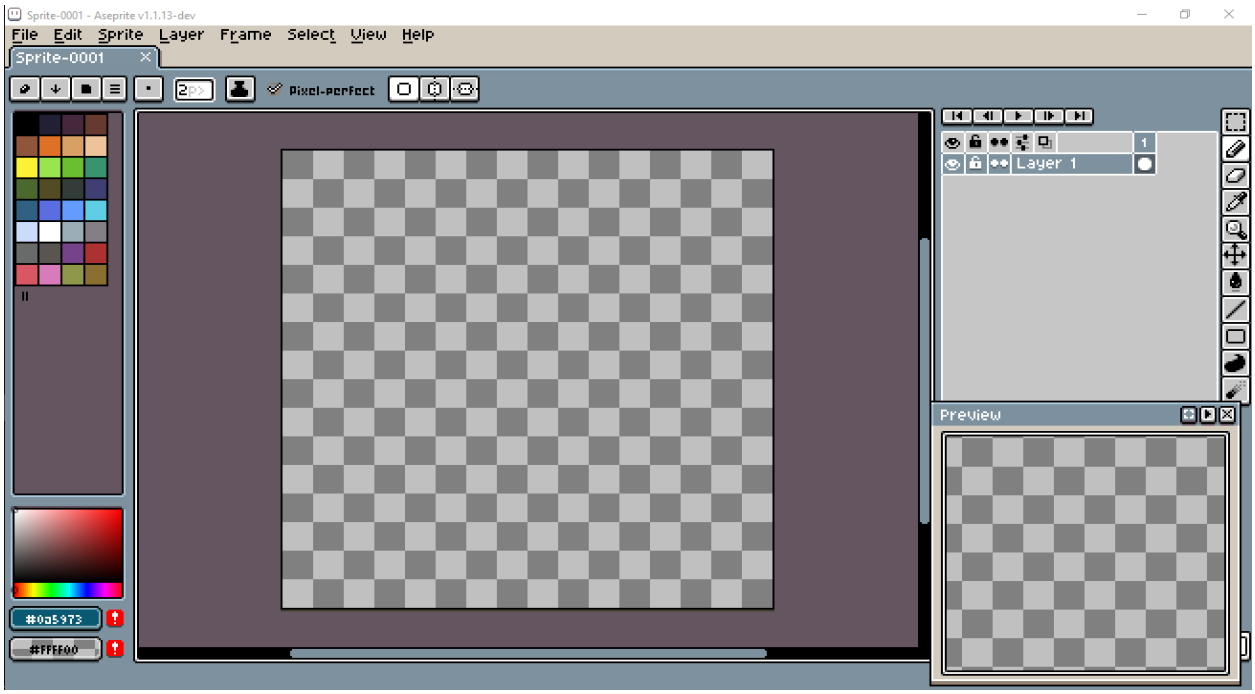


Slika 2.2. Izgled početnog ekrana Microsoft Office Visual Studio 2017 Community

Unity zajedno sa Microsoft Visual Studio-m se može pronaći na službenoj Unity [1] stranici potpuno besplatno ukoliko će se koristiti u edukacijske svrhe, dok je za komercijalnu uporabu potrebno kupiti licencu.

2.2. Aseprite

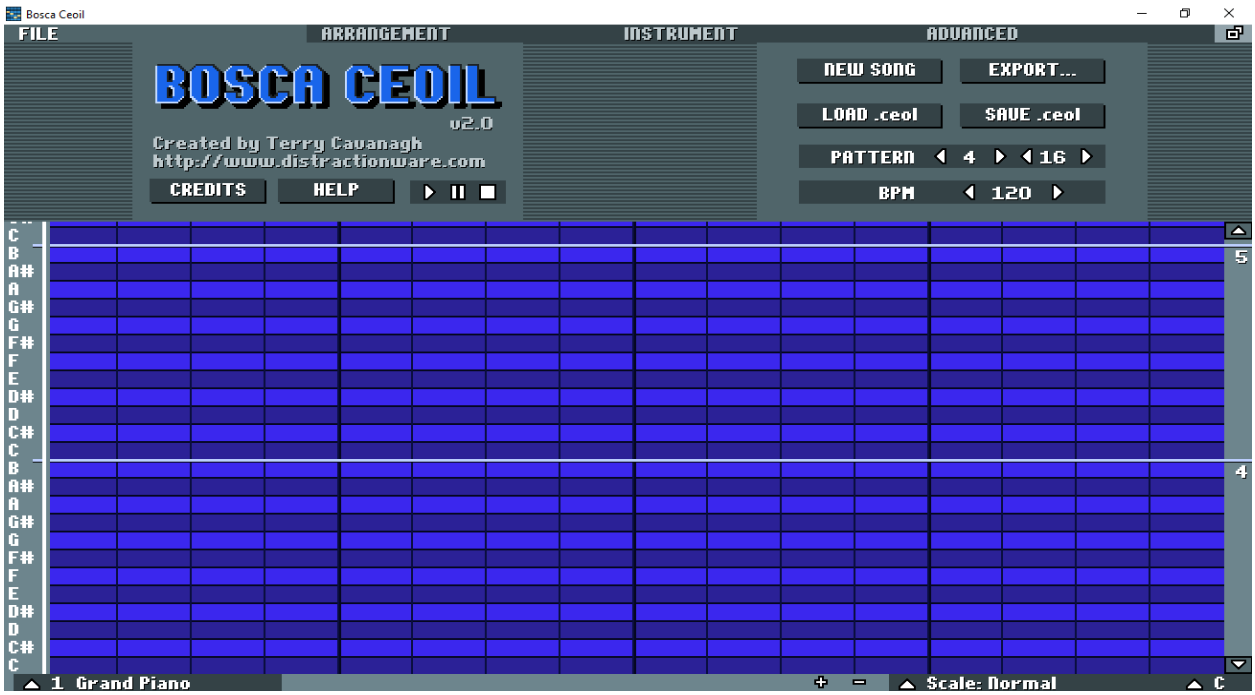
Aseprite [2] je programski alat namijenjen za izradu dvodimenzionalnih bit mapa koje se koriste prilikom izrade scena u računalnim igrama, kraćeg naziva sprite [3]. Sprite-ovi se mogu promatrati kao male sličice koje predstavljaju likove ili nekakve druge objekte unutar igre. Pomoću programa Aseprite koji se može preuzeti na njihovoj službenoj stranici [4] su napravljeni svi likovi i objekti koji se nalaze unutar igre te svi sprite-ovi potrebni za izradu animacija hodanja, mirnog stajanja u mjestu i napadanja.



Slika 3.1. Izgled otvorenog novog projekta u Aseprite-u

2.3. Bosca Ceoil

Bosca Ceoil [5] je program koji služi za izradu 8-bitne glazbe popularno prozvane *Chiptune* [6]. Bosca Ceoil je besplatan alat koji pruža velike mogućnosti svojim korisnicima, ima jako veliki izbor instrumenata koji se mogu koristiti te omogućava online korištenje bez potrebe za preuzimanjem programa i instaliranjem istog.



Slika 4.1. Izgled početnog ekrana programa Bosca Ceoil

3. RAZVOJ IGRE

3.1. Ideja

Ideja je bila napraviti 2D igru koja će biti spoj između Zelde [7] i World of Warcraft-a [8]. Glavni izvor ideje za samu mapu i sami dizajn likova su uzeti iz igre Legend of Zelda, dok je mehanika za bacanje *spell*-ova (čarolija) uzeta iz World of Warcraft-a.

Svi protivnici koji se pojavljuju u igri će biti Skulleton-i, rasa koja je na prvi pogled ista kao i kosturi, no razlikuju se u tome što Skulleton-i imaju veliku lubanju. Sve Skulleton-e predvodi njihov kralj i vladar Skulleton King koji je ujedno i konačni šef (engl. *final boss*, nadalje korišten izraz) same igre. Skulleton-i se u izgledu ne razlikuju po ničemu no svaki Skulleton radi određenu količinu štete (engl. *damage*, nadalje korišten izraz) te ima određeni količinu zdravlja (engl. *health*, nadalje korišteni izraz)..

Svaki Skulleton kada se označi prikazuje svoj bar za zdravlje (engl. *health bar*, nadalje korišten izraz) koji se prilikom napadanja Skulleton-a odnosno primanja *damage*-a smanjuje. Isto tako pored *health bara* je moguće vidjeti i ikonu od označenog neprijatelja.

3.2. Priča

Priča prati Philipa koji može biti čarobnjak (engl. *wizard),* vitez (engl. k*night)* ili streličar (engl. *archer)*. Philip se pronašao unutar katakombi u kojima žive The Keepers-i te njihov prijateljski nastrojeni duh Zhuti. Zhuti na samom početku igre Philipa uči nekakve osnove kao što su kretnje, interakcije, dodavanje čarolija (engl. *spell*, nadalje korišten izraz) te na kraju i korištenje *spell*-ova. Tijekom prolaska kroz katakombe Philip susreće brojne likove s kojima može pričati, vršiti interakcije ili od kojih može dobiti nekakve zadatke (engl. *quest*, nadalje korišten izraz) koje može izvršiti. *Quest*-ove Philipu može davati i Zhuti te mu ih i zadaje nekoliko puta tijekom cijele igre. Philipov glavni zadatak je izaći iz katakombi, no usput skupljajući i rješavajući *quest*-ove, jedan od zadataka mu postaje poraziti Skulleton King-a i osloboditi katakombe od Skulleton-a kako bi The Keepers-i mogli slobodno živjeti kao što su živjeli prije pojave Skulleton-a.

3.3. Likovi

U igri se nalazi više različitih likova koji se razlikuju po količini *health*-a kojeg imaju, *damage*-u koji rade i brojnim dugim stvarima. Likovi koji se susreću tijekom igre su Philip koji je glavni lik, čuvari katakombi u igri poznati kao The Keepers, Skulleton-i i Skulleton King, Zhuti, Hose i Hulio. Svi likovi koji u svom nazivu imaju Skulleton su neprijatelji te se tijekom igre glavni

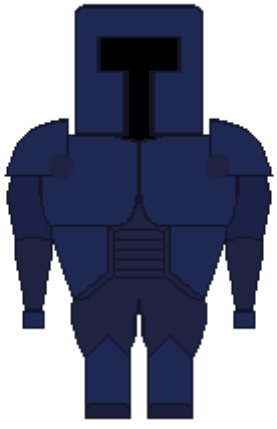
junak bori protiv njih kako bi oslobodio katakombe. Svi Skulleton-i nakon svoje smrti mogu ispustiti (engl. *dropp*, nadalje korišteni izraz) stvari (engl. *item*, nadalje korišteni izraz) koje Philip može pokupiti i koristiti tijekom igre.

3.3.1. Glavni lik – Philip

Glavni lik je Philip koji je ovisno o izabranoj klasi čarobnjak (engl. *wizard)*, vitez (engl. k*night)* ili streličar (engl. *archer)*. Philip je ljudske rase te je kao mali odlučio da će postati heroj i pomagati ljudima. Kako obični ljudi ne mogu biti heroji jer su slabi, Philip je odlučio da će postati dobar u jednoj od tri klase koje se smatraju najmoćnijima na svijetu, a to su borba s mačevima, borba pomoću luka i strijela ili borba magijom. Nakon što je ovladao željenom klasom on je postao heroj te je odlučio lutati svijetom i pomagati ljudima. Nakon nekog vremena Philip se odlučio vratiti kući no odjednom se je našao u katakombama koje su prepune Skulleton-a te od tu počinje igra.

Ovisno o izabranoj disciplini Philip ima određene moći. Svaka disciplina ima unikatni način igranja te unikatne *spell*-ove.

Knight je klasa koja za napade koristi rezove (engl. *slash*, nadalje korišten izraz). Postoje tri različita *slash*-a te svaki koristi određenu količinu mane, određeno drugo se baca (engl. *cast*, nadalje korišten izraz) i radi određenu količinu *damage*-a. *Knight* ima pristup *Fire Slash*-u, *Frost Slash*-u i *Lightning Slash*-u.



Slika 3.3.1.1. Izgled knight-a u izbornom ekranu

Archer je drukčiji od *knight*-a u puno stvari, a jedna od glavnih stvari su *spell*-ovi koje *archer* koristi. *Spell*-ovi kojima *archer* ima pristup su *Fire Arrow*, *Frost Arrow* i *Lightning Arrow*. Svaki od navedenih *spell*-ova radi određenu količinu *damage*-a, koristi određenu količinu mane i određeno dugo se mora *cast*-ati.

Slika 3.3.1.2. Izgled archer-a u izbornom ekranu

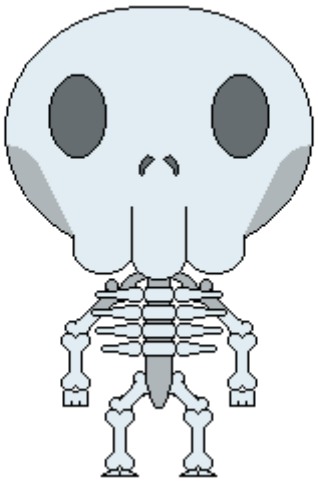
Wizard je klasa koja slična *archer*-u po stilu igre, no jedna od razlika je ta što *wizard* za napade koristi *spell*-ove kao što su *Fireball*, *Frostbolt* i *Lightning Strike*. Isto kao i kod *knight*-a i *archer*-a, svaki *spell* radi određenu količinu *damage*-a, određeno dugo se *cast*-a i zahtjeva određenu količinu mane.

Slika 3.3.1.3. Izgled wizard-a u izbornom ekranu

3.3.2. Skulleton-i

Skuleton-i su rasa koja se od kostura, kao i što je ranije spomenuto, razlikuje samo u tome što imaju velike lubanje. Tijekom igranja igre oni su glavni protivnici a njihov vođa je zli *Skulleton King*. *Skulleton*-i su osmišljeni prilikom izrade ove igre te su iz tog razloga veoma unikatni, jer u niti jednoj drugoj igri koja je danas na tržištu ne postoji rasa koja je izgleda kao običan kostur no velike lubanje.

Njihov vođa, *Skulleton King* koji je vladar svih *Skulleton*-a se od ostatka *Skulleton*-a razlikuje u tome što je znatno veći. Za *Skulleton*-e i *Skulleton King*-a nisu raditi posebni *sprite*-ovi i animacije, već se samo jedan od već postojećih *Skulleton*-a povećao tako što je mu je podešen *scaling*, odnosno skaliranje koje množi postojeće dimenzije objekta s nekim brojem tj. vrijednosti. Ukoliko je vrijednost veća od jedan skalirani objekt će se povećati te će mu dimenzije narasti. Ukoliko je vrijednost manja od jedan, skalirani objekt će se smanjiti te će mu se sukladno tome i dimenzije smanjiti.



Slika 3.3.2.1. Izgled Skulleton-a i Skulleton King-a

3.3.3. Ostali likovi

Uz glavnog lika i zle *Skulleton*-e, u igri se još pojavljuje nekolicina likova koji pomažu glavnom liku te vrše interakcije s njim. Jedan od najbitnijih likova koji pomažu Philipu (glavnom liku) je *Zhuti*. *Zhuti* je duh koji se javlja *Philip*-u po prvi put na samom početku igre, gdje ga uči osnove poput osnova kretnji unutar igre, kako vršiti interakcije i kako napadati. Uz to *Zhuti Philip*-u zadaje i nekoliko *quest*-ova koje *Philip* mora riješiti kako bi pobijedio igru.

Uz *Zhutog* ne smije se zaboraviti niti na *Hulia* i *Hose*-a. *Hose* se prvi put tijekom igre pojavljuje na samom početku gdje *Philip* može kliknuti desnim klikom miša na njega te aktivirati *quest* koji nakon toga mora obaviti, dok Hulio nešto kasnije, gdje ga *Philip* i *Zhuti* spase od zlih *Skulleton*-a koji su ga napali.

Na samom kraju tu su i *The Keepers*-i. Njih se unutar igre najčešće spašava od *Skulleton*-a koji su ih opkolili. Važno je napomenuti za su *Hulio* i *Hose* također i *The Keepers*-i te se samim tim ne razlikuju po svom izgledu od drugih *The Keepers*-a.

Slika 3.3.1. Izgled svih The Keepers-a

3.4. Kamera

Kako bi mogli vidjeti sve što se dešava unutar igre prilikom izrade igre za svaku scenu je korištena po jedna kamera. Ukoliko bi kamera bila statična odnosno stajala na jednom mjestu prilikom igranja igre ne bi mogli vidjeti sve što se dešava, lik bi nam izlazio iz okvira kamere te ga ne bi mogli vidjeti.

Kako bi se taj problem izbjegao kreirana je skripta *CameraFollow* te dodana kameri u sceni. Isto tako je potrebno da u nekim scenama kamera bude statična te se ne miče, na kameru u tim scenama nije dodana *CameraFollow* skripta. *CameraFollow* skripta uzima referencu na glavnog lika i *tilemap* [14] [15] objekt te na osnovu toga proračunava položaj kamere.

Proračun položaja kamere se vrši pomoću funkcije *SetLimits()* koja kao parametre uzima minimalni i maksimalni položaj u kojem se kamera smije nalaziti te sprječava glavnog lika i kameru da napuste područje koje je određeno tim granicama.

```
public class CameraFollow : MonoBehaviour
{
    [SerializeField]
    private Tilemap tilemap;

    private Transform target;
    private float xMin, xMax, yMin, yMax;
    private Player player;

    // Use this for initialization
    void Start ()
    {
        target = GameObject.FindGameObjectWithTag("Player").transform;
        player = target.GetComponent<Player>();

        Vector3 minTile = tilemap.CellToWorld(tilemap.cellBounds.min);
        Vector3 maxTile = tilemap.CellToWorld(tilemap.cellBounds.max);

        SetLimits(minTile, maxTile);
        player.SetLimits(minTile, maxTile);
    }

    private void LateUpdate()
    {
        transform.position = new Vector3(target.position.x, target.position.y, -20);
    }

    private void SetLimits(Vector3 minTile, Vector3 maxTile)...
}
```

Slika 3.4.1. Izgled CameraFollow skripte

```
private void SetLimits(Vector3 minTile, Vector3 maxTile)
{
    Camera cam = Camera.main;
    float height = 2f * cam.orthographicSize;
    float width = height * cam.aspect;

    xMin = (minTile.x + width) / 2;
    xMax = (maxTile.x - width) / 2;

    yMin = (minTile.y + height) / 2;
    yMax = (maxTile.y - height) / 2;
}
```

Slika 3.4.2. izgled funkcije SetLimits()

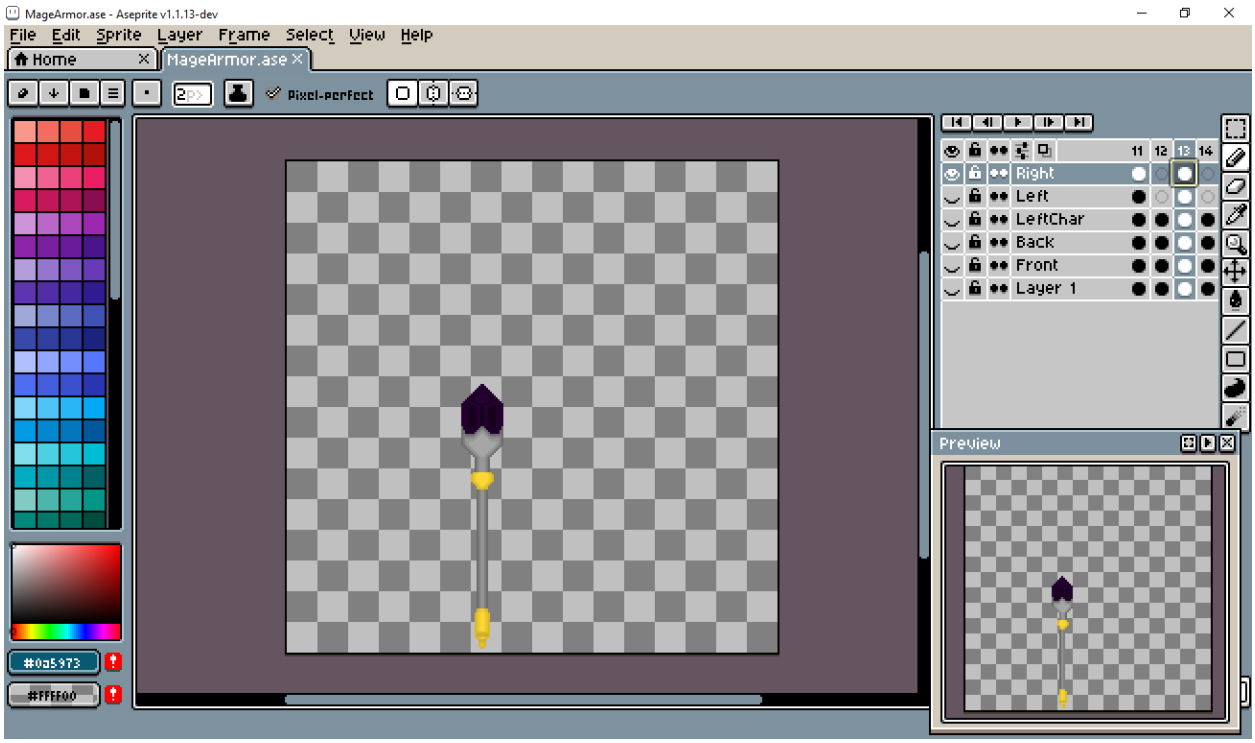
3.5. Grafika

Sva grafika koja je vidljiva unutar igre je napravljena unutar *Aseprite*-a [2], no moguće ju je napraviti u bilo kojem programu koji omogućavaju obradu slike poput Adobe Photoshopa [9] i drugih sličnih programa.

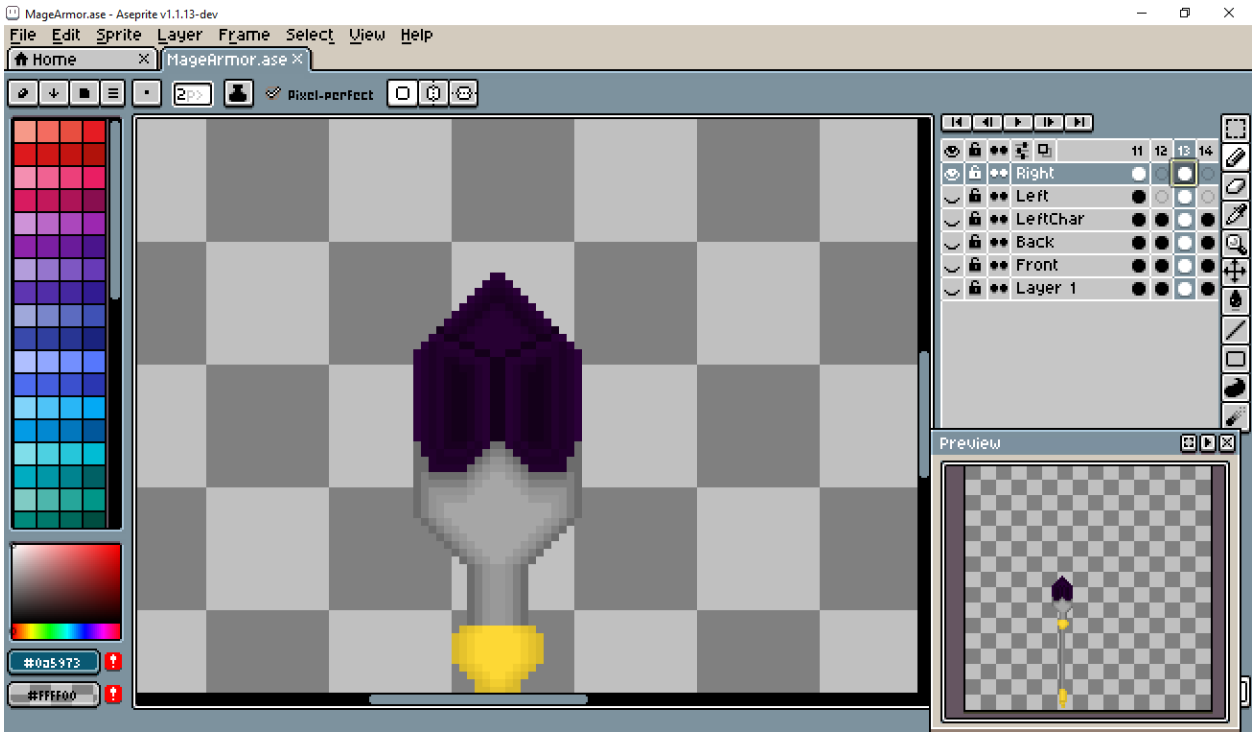
Razlog radi kojeg je izabran baš Aseprite [2] je radi njegovog jednostavnog korisničkog sučelja, veoma jednostavnog rukovanja i snalaženja unutar programa, velike količine početnih paleta boja i male cijene. Program nije besplatan te je potrebno kupiti licencu za korištenje.

Program omogućava izvoz slika u *png* formatu koji je jako bitan kako se pozadina slike ne bi vidjela unutar igre ukoliko to ne želimo. *PNG* je kratica za *Portable Network Graphics* [10] odnosno otvoreni grafički format koji se koristi u game development industriji prvobitno za izvoz *sprite*-ova za igre.

Prilikom izrade *sprite*-ova kako bi se dobio efekt dubine je potrebno sjenčati *sprite*-ove odnosno prikazivanje svijetlih područja svjetlijim bojama a tamnijih tamnijima.U nastavku se može vidjeti kako izgleda otvoreni projekt u *Aseprite*-u koji je u fazi izrade.



Slika 3.5.1. Izgled sjenčane slike iz daljine



Slika 3.5.2. Izgled sjenčane slike iz blizine

3.6. Vrste korištenih objekata prilikom izrade igre

Prilikom izrade igre koristili su se razni objekti unutar scena kao što su *GameObject*, *Image*, *AudioSource*, *Tilemap* text, i *Button* te će se u nastavku objasniti zašto su se koristili pojedini objekti. Bilo što što se nalazi unutar scene je samo po sebi objekt no postoji i tzv. prazni objekti naziva GameObject koji će se ukratko objasniti kao i sve ostale vrste objekata.

3.6.1. *GameObject*

GameObject [7] je jedan od najbitnijih vrsta objekata unutar *Unity*-a jer može koristiti bilo koju drugu vrstu objekta unutar *Unity*-a. Na primjer unutar scene u *Unity*-u se može kreirati *Empty GameObject* unutar kojeg se mogu staviti ostale vrste objekata. Isto tako prilikom programiranja ako napravimo varijablu tipa *GameObject*, ona onda može raditi s bilo kojom vrstom objekta koja joj je dodijeljena.

3.6.2. *Image*

Image [8] je vrsta objekta u *Unity*-u koja se koristi najviše prilikom izrade korisničkog sučelja te predstavlja ne interaktivnu sliku za korisnika. To bi značilo da korisnik kada klikne na sliku ili pritisne neku tipku na tipkovnici ne izazove nikakvu reakciju od strane slike, što bi laički bilo rečeno ne dođe do nikakve promjene slike.

3.6.3. *AudioSource*

AudioSource [9] je vrsta objekta koja predstavlja izvor zvuka unutar igre. On se koristi kako bi se unutar igre mogla reproducirati glazba (engl. *in-game music*), zvukovi kao što su na primjer zvukovi koračanja ili udaraca tupim predmetom te monolozi i dijalozi.

3.6.4. *Tilemap*

Tilemap [10][11] u *Unity*-u omogućava brzo i efikasno kreiranje mapa koje su bazirane na *sprite*-ovima. Kada se kreira *tilemap* objekt unutar scene u Unity-u te nakon što podesimo mrežu (engl. *grid,* na dalje korišten termin) koja nam određuje veličinu *sprite*-ova i razmak između njih veoma brzo možemo kreirati mapu bilo koje veličine.

3.6.5. *Button*

Button [12] nam predstavlja tipku koja se može pritisnuti pomoću klika miša te nakon što je tipka pritisnuta izvrši se nekakva akcija koja je unaprijed definirana. Svaki objekt koji je tipa *button* ima komponentu tekstualnog okvira i slike kako bi mu se izgled mogao prilagoditi po želji korisnika

3.6.6. *Text*

Text [13] unutar *Unity*-a predstavlja tekstualni okvir pomoću kojeg se unutar scene može prikazati tekstualni zapis. *Text* se često koristi zajedno u kombinaciji s *button*-om. Unutar *Text* objekta možemo mijenjati font, boju slova, veličinu slova, podebljati, staviti kurziv ili podcrtati tekst koji se nalazi unutar objekta.

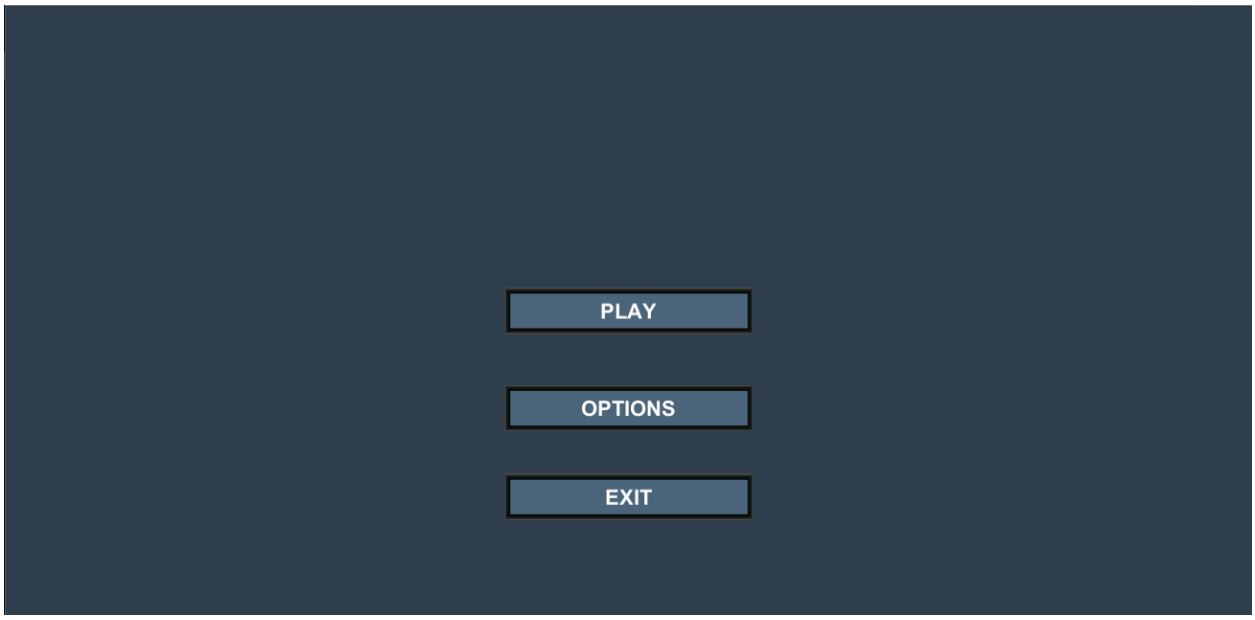
3.7. Grafičko sučelje - *GUI*

Grafičko sučelje *GUI* ili na engleskom *Graphics User Interface* predstavlja način interakcije čovjeka s računalom putem ikona, prozora i gumba s tekstom putem miša ili tipkovnice. *GUI* korisnicima omogućava da ne moraju oni sami upisivati naredbe već da se jednostavnim klikom miša ili pritiskom tipke na tipkovnici unaprijed unesena naredba izvrši. Svi programi koji su korišteni prilikom izrade ovog rada, pa tako i *Unity*, imaju svoje grafičko sučelje koje omogućava da putem klika miša se izvrše naredbe koje bi bile korisnicima složene za napisati.

3.7.1. Glavni izbornik

Prilikom pokretanja igre, glavni izbornik se prvi prikaže korisniku te korisnik pomoću glavnog izbornika vrši prvu interakciju s igrom. Glavni izbornik se najčešće sastoji od nekolicine gumba na kojima se nalazi tekstualni okvir unutar kojeg je vidljivo što taj gumb predstavlja. Nakon

klika miša na gumb glavnog izbornika otvara se novi izbornik unutar kojeg se isto tako pomoću klika miša može kliknuti na gumb.



Slika 3.7.1.1. Izgled glavnog izbornika unutar igre

Skripta koja kontrolira glavni izbornik je vrlo jednostavna i kratka te sadrži funkciju samo za gumb koji služi za izlazak iz igre. Prilikom klika miša na gumb naziva „*EXIT*“ pokreće se funkcija *ExitGame()* čija je funkcionalnost dodana na gumb „*EXIT*“.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public void ExitGame()
    {
        Debug.Log("Game closed");
        Application.Quit();
    }
}
```

Slika 3.7.1.2. Izgled skripte glavnog izbornika

Ostatak funkcionalnosti koji je potreban kako bi glavni izbornik radio je postignuta pomoću *GameObject.SetActive()* funkcija. *GameObject.SetActive()* funkcija funkcionira tako da ona uključi ili isključi objekt koji se nalazi unutar scene. Ukoliko se nalazimo u glavnom izborniku i kliknemo mišem na gumb na kojem je napisano „*PLAY*“ glavni izbornik će postati neaktivan te će aktivirati izbornik za odabir glavnog lika tzv. „*selection screen*“.

3.7.2. Izbornik za odabir glavnog lika (*Selection Screen*)

Nakon što se unutar glavnog izbornika klikne na gumb „*PLAY*“ glavni izbornik će se deaktivirati te će se aktivirati izbornik za odabir glavnog lika (engl. *selection screen* te nadalje korišten termin). Unutar *selection screen*-a se mogu uočiti gumbovi „*PREVIOUS*“, „*NEXT*“, „*SELECT*“ i „*BACK*“ te objekt koji na sebi ima komponentu tipa *Image*. Prilikom klika miša na gumbove „*PREVIOUS*“ i „*NEXT*“ se može promijeniti odabrani lik. Ukoliko se pritisne gumb „*BACK*“ *selection screen* će se deaktivirati te će se aktivirati glavni izbornik, dok ukoliko se klikne mišem na tipku „*SELECT*“ zadani glavni lik će se učitati te će se učitati nova scena koja sadrži našeg odabranog lika.

Slika 3.7.2.1. Izgled selection screen-a

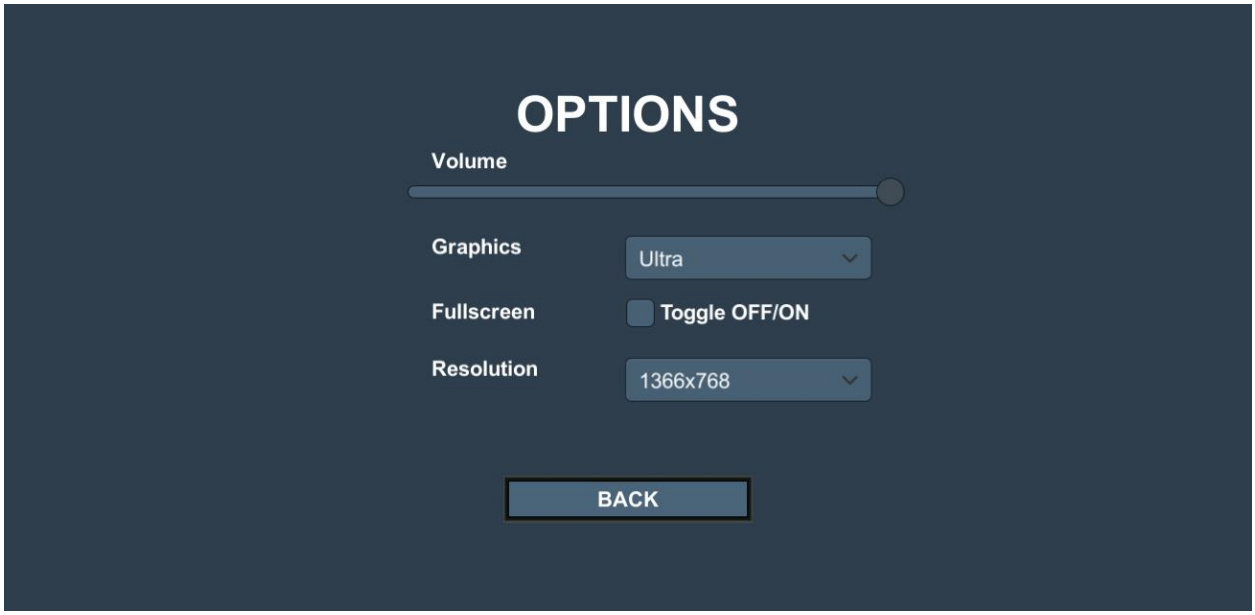
Priča koja se odvija iza *selection screena* je ta da umjesto da se glavni lik učita u scenu, za svakog lika se učitava posebna scena koja sadrži točno odabranog lika, odgovarajuće *item*-e te *spell*-ove. Svaka scena je ista te se razlikuje samo u prethodno navedenim stavkama koje se tiču glavnog lika. Skripta koja kontrolira *selection screen* se naziva *CharacterSelection* te su najbitnije funkcija *Next()* i *Previous()* pomoću kojih se mijenja objekt glavnog lika unutar *selection screen*-a prilikom klika na gumbe „*NEXT*“ i „*PREVIOUS*“.

```
public void Next()
{
    characterIndex++;
    characterList[characterIndex].SetActive(true);
    characterList[characterIndex - 1].SetActive(false);

    if (characterIndex > characterList.Length)
    {
        characterIndex = 0;
    }
}

public void Previous()
{
    characterIndex--;
    characterList[characterIndex].SetActive(true);
    characterList[characterIndex + 1].SetActive(false);

    if (characterIndex < 0)
    {
        characterIndex = characterList.Length;
    }
}
```

Slika 3.7.2.2. Funkcije Previous() i Next()

3.7.3. Izbornik s postavkama (*Options Screen*)

Izbornik s postavkama (engl. *options screen* te nadalje korišten termin) se aktivira ukoliko se unutar glavnog izbornika pritisne na gumb „*OPTIONS*“. Prilikom klika na gumb „*OPTIONS*“ glavni izbornik će se deaktivirati te će se aktivirati izbornik s postavkama unutar kojeg je moguće podešavati glasnoću glazbe unutar igre, kvalitetu slike, želimo li da nam igra bude preko cijelog ekrana ili unutar prozora te rezoluciju unutar koje želimo da nam igra bude pokrenuta.



Slika 3.7.3.1. Izgled izbornika s postavkama

Za kontrolu glasnoće se koristi funkcija *SetVolume()* koja kao parametar prima varijablu tipa *float* unutar koje je pohranjena vrijednost koja se pomoću klizača može postaviti unutar izbornika. Kvalitetu podešava funkcija *SetQuality()* koja prima parametar tipa *int* koji predstavlja indeks izabrane vrste kvalitete.

Odabir između prikaza putem cijelog zaslona ili prikaza unutar prozora odrađuje funkcija *SetFullscreen()*. Njoj se predaje varijabla tipa *bool* koja može poprimiti samo jedno od dva stanja a to su istina (engl. *true*) i laž (engl. *false*). Na samom kraju imamo funkciju koja postavlja rezoluciju igre. Funkcija koja postavlja rezoluciju unutar igre se zove *SetResolution()* i ona kao parametar prima indeks rezolucije. Pomoću indeksa rezolucije koji je odabran se postavlja željena rezolucija igre.

Sve rezolucije koje se mogu podesiti dohvaća i pohranjuje funkcija *GetResolutions()* koja se poziva unutar funkcije Start() kako bi se prilikom samog pokretanja igre dohvatile sve rezolucije koje se mogu podesiti na pokrenutom računalu.

```
public void SetVolume(float volume)
{
    audioSource.volume = volumeSlider.value;
}

public void SetQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
}

public void SetFullscreen(bool isFullscreeen)
{
    Screen.fullScreen = isFullscreeen;
}
```

Slika 3.7.3.2. Izgled funkcija SetVolume(), SetQuality() i SetFullScreen()

```
public void GetResolutions()
{
    resolutions = Screen.resolutions;
    resolutionDropdown.ClearOptions();

    List<string> options = new List<string>();

    int currentResolutionIndex = 0;
    for (int i = 0; i < resolutions.Length; i++)
    {
        string option = resolutions[i].width + "x" + resolutions[i].height;
        options.Add(option);

        if (resolutions[i].width == Screen.currentResolution.width &&
            resolutions[i].height == Screen.currentResolution.height)
        {
            currentResolutionIndex = i;
        }
    }

    resolutionDropdown.AddOptions(options);
    resolutionDropdown.value = currentResolutionIndex;
    resolutionDropdown.RefreshShownValue();
}

public void SetResolution(int resolutionIndex)
{
    Resolution resolution = resolutions[resolutionIndex];
    Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);
}
```

Slika 3.7.3.3. Izgled funkcija GetResolutions() i SetResolutions()

3.7.4. Pauzirajući izbornik (*Pause Menu*)

Pauzirajući izbornik (engl. *pause menu*) je izbornik koji se aktiviran kada se tijekom igranja igre (pod igranja igre misli se unutar scene koja je otvorena nakon što se unutar izbornika gdje se odabire glavni lik pritisne gumb „*SELECT*“) pritisne tipka *escape* koja se nalazi u gornjem lijevom kutu na tipkovnici te na sebi ima natpis „ESC“. Nakon što se pritisne tipka „ESC“ igra se pauzira te se otvara pauzirajući izbornik unutar kojeg su vidljivi „*RESUME*“, „*OPTIONS*“, „*MENU*“ i „*EXIT*“ gumbovi.

Slika 3.7.4.1. Izgled pauzirajućeg izbornika

Gumb „*RESUME*“ zatvara pauzirajući izbornik te prekida pauzu koja se dešava nakon što je pritisnuta tipka „*ESC*“. Ukoliko se klikne na gumb „*OPTIONS*“ otvorit će se izbornik s postavkama koji je prethodno objašnjen. Gumb „*MENU*“ će učitati glavni izbornik dok će gumb „*EXIT*“ zatvoriti igru.

```
public void Resume()
{
    pauseMenu.SetActive(false);
    Time.timeScale = 1f;
    isPaused = false;
}

void Pause()
{
    pauseMenu.SetActive(true);
    Time.timeScale = 0f;
    isPaused = true;
}

public void MainMenu()
{
    Time.timeScale = 1f;
    SceneManager.LoadScene("MainMenu");
}

public void ExitGame()
{
    Application.Quit();
}
```

Slika 3.7.4.1. Prikaz funkcija koje kontroliraju ponašanje gumba unutar pauzirajućeg izbornika

Dok funkcija *Resume()* služi za deaktivaciju pauzirajućeg izbornika i isključivanje pauze unutar igre, funkcija Pause() služi za pauziranje igre te aktivaciju pauzirajućeg izbornika. Pomoću funkcije *MainMenu* se učitava glavni izbornik dok se pomoću funkcije *ExitGame()* izlazi iz igre.

3.7.5. Zaslon nakon smrti (*Death Screen*)

Ukoliko prilikom igranja igre se desi da igrač umre uključit će se zaslon nakon smrti (engl. *death screen*) na kojem su vidljiva 2 gumba. Ukoliko se pritisne gumb „*RESTART*“ cijela igra će se nanovo učitati i igrač će započeti igru iz početka. Ako se pritisne gumb „*MAIN MENU*“ igrač će se vratiti u početni izbornik nakon čega će moći pritisnuti jedan od tamo ponuđenih gumba.

Slika 3.7.5.1. Izgled zaslona nakon smrti

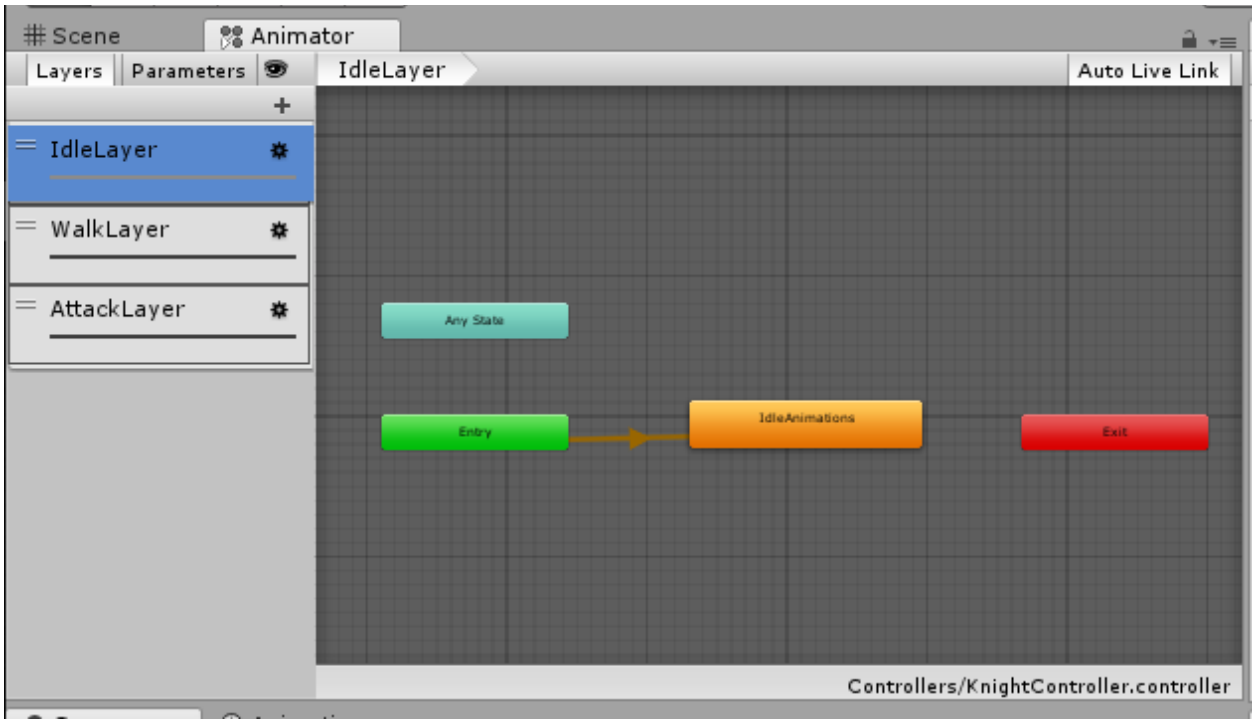
Funkcija koja će se izvršiti ukoliko se klikne na gumb „*MAIN MENU*“ je ista kao i u prethodnim slučajevima kada je gumb za glavni izbornik bio ponuđen. Ukoliko se klikne na gumb „*RESTART*“ izvršit će se funkcija *Restart()* koja će ponovo učitati scenu koja je bila učitana u trenutku kada je funkcija bila pozvana.

```
public void Restart()
{
    Time.timeScale = 1f;
    deathScreen.SetActive(false);
    bagBar.SetActive(true);
    SceneManager.LoadScene(SceneManager.GetActiveScene().name);
}
```

Slika 3.7.5.1. Izgled funkcije Restart()

3.8. Animator i animacije

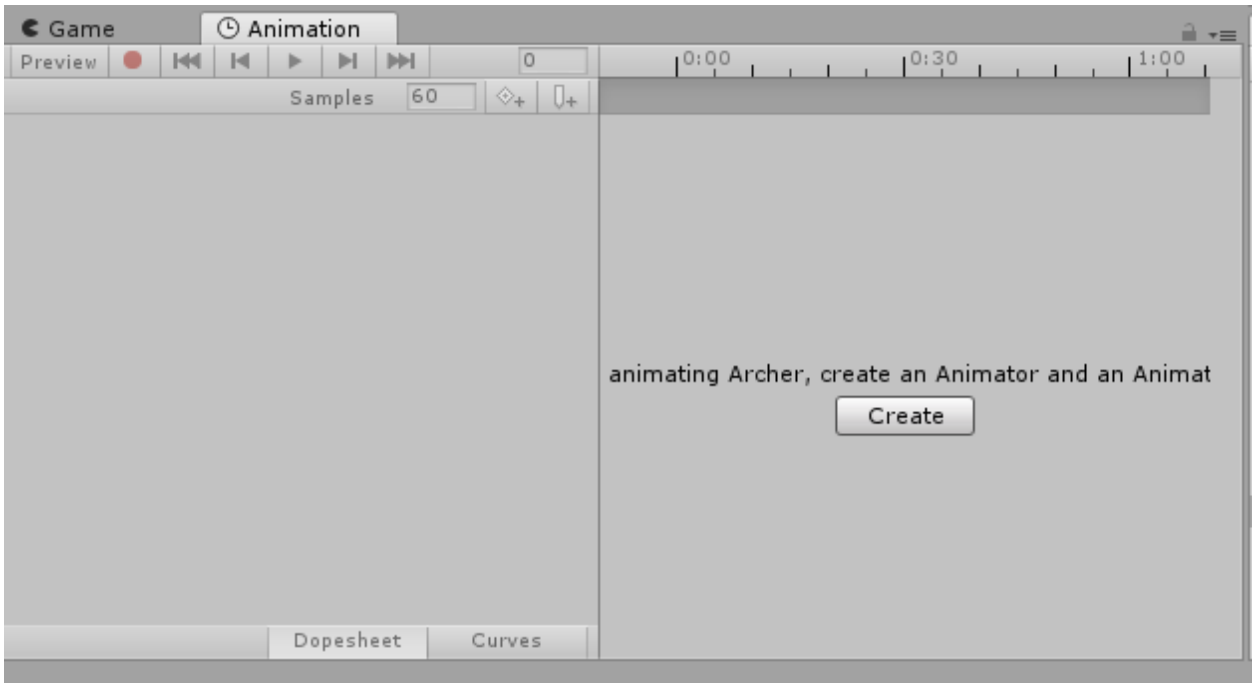
Sve animacije koje su vidljive unutar igre su izrađene pomoću animatora koji je sastavni dio *Unity*-a. Animacije su najjednostavnije rečeno slike koje se izmjenjuju jedna za drugom nakon nekog vremenskog perioda. U *Unity*-u za kontrolu animacija je korišten animator unutar kojeg su korištena tri layer-a po jedan za stanje kada igrač stoji na mjestu (*IdleLayer*), jedan za stanje hodanja (*WlakingLayer*) i jedan za napadanje (*AttackingLayer*) pri čemu se promjena *layer* kontrolirana putem skripte. Unutar svakog *layer*-a se nalazi *Animation Tree* pomoću kojeg se kontrolira koja će se animacija pokrenuti ovisno o tome u koju je stranu okrenut glavni lik. Sukladno tome se u svakom *animation tree*-u nalaze po četiri animacije. Ukoliko se glavni lik kreće prema dolje potrebno je da se pokrene animacija hodanja glavnog lika prema dolje, ukoliko bi glavni lik stao nakon što je hodao prema dolje potrebno je da se pokrene animacija na kojoj glavni lik stoji na mjestu nakon što je hoda prema dolje te na samom kraju ukoliko je glavni lik napao prema dolje, mora se pokrenuti animacija u kojoj glavni napada prema dolje.



Slika 3.8.1. Izgled Animatora unutar Unity-a

Kako bi sve što je iznad navedeno bilo moguće te prije bilo kakvog rada u animatoru potrebno je prvo napraviti animacije. Sve animacije koje su vidljive u igri su napravljene u *Unity*-u pomoću *Animation* prozora. *Animation* prozor funkcionira tako da se u njega ubacuju sličice ili bolje rečeno *sprite*-ovi od kojih želimo da se sastoji animacija. Nakon dodavanja *sprite*-ova može se podesiti vremenski razmak između svakog *sprite*-a pomoću kojeg se može odrediti koliko dugo

će određeni *sprite* biti prikazan. Svi *sprite*-ovi koji su vidljivi unutar igre su napravljeni kao i što je prethodno navedeno unutar *Aseprite*-a.



Slika 3.8.2. Izgled Animation prozora za kreiranje animacija

ZAKLJUČAK

U ovom završnom radu je obuhvaćen cijeli postupak izrade računalne igre, od same ideje pa do finalne realizacije. Igra je kao i što je prije navedeno izrađena pomoću nekolicine različitih programa od kojih je najbitniji *Unity* te programski jezik C#. Za uspješnu realizaciju igre je bilo potrebno nekakvo predznanje o snalaženju unutar *Unity*-a. Uz *Unity* su još korišteni *Aseprite* program za crtanje *sprite*-ova i *Bosca Ceoli* za izradu glazbe. Na igri su moguće još brojne izmijene i dorade poput dodavanja opcije za dva igraća kako bi dvije osobe mogle igrati zajedno na jednom računalu što bi bilo veoma zabavno iskustvo ili dodavanje govora na likove kako bi se činilo da likovi pričaju. Neke od mogućih dorada su i novi likovi s kojima se može igrati, novi *item*-i, novi *spell*-ovi, novi neprijatelji te brojne druge dorade vezane za proširenje sadržaja unutar same igre.

LITERATURA

[1] Unity Learn

https://unity3d.com/learn, pristupljeno: prosinac 2017.

[2] Aseprite

https://www.aseprite.org/docs/tutorial/, pristupljeno: prosinac 2017.

[3] Sprite

https://en.wikipedia.org/wiki/Sprite_(computer_graphics), pristupljeno: prosinac 2017.

[4] Chiptune

https://en.wikipedia.org/wiki/Chiptune, pristupljeno kolovoz 2018.

[5] Adobe Photoshop

https://www.adobe.com/products/photoshop.html, pristupljeno: prosinac 2017.

[6] Portable Network Graphics

https://hr.wikipedia.org/wiki/PNG, pristupljeno: prosinac 2017.

[7] GameObject

https://docs.unity3d.com/Manual/GameObjects.html, pristupljeno: veljača 2017.

[8] Image

https://docs.unity3d.com/Manual/script-Image.html, pristupljeno: veljača 2017.

[9] AudioSource

https://docs.unity3d.com/Manual/class-AudioSource.html, pristupljeno: kolovoz 2018.

[10] Tilemap

https://docs.unity3d.com/Manual/Tilemap.html, pristupljeno: lipanj 2018.

[11] Tilemaps in Unity by Brackeys

https://www.youtube.com/watch?v=ryISV_nH8qw, pristupljeno: lipanj 2018.

[12] Button

https://docs.unity3d.com/ScriptReference/UI.Button.html, pristupljeno: veljača 2017.

[13] Text

https://docs.unity3d.com/Manual/script-Text.html, pristupljeno: veljača 2017.

[14] Inventory System Tutorial in Unity 5

https://www.youtube.com/watch?v=ZW6RCKVnqT4&list=PLivfKP2ufIK78r7nzfpIEH89Nlnb__RRG, pristupljeno: kolovoz 2018.

[15] Unity RPG Tutorial

https://www.youtube.com/watch?v=Pk3GCgaNVTY&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm, pristupljeno: svibanj 2018.

[16] How To Make RPG in Unity

https://www.youtube.com/watch?v=nu5nyrB9U_o&list=PLPV2KyIb3jR4KLGCCAciWQ5qHudKtYeP7, pristupljeno: ožujak 2018.

[17] Official Unity Tutorials

https://www.youtube.com/watch?v=YvA1O7MYs_w&list=PLX2vGYjWbI0RmYmP19OimzWSy2PpLxMk7, pristupljeno: studeni 2017.

SAŽETAK

U završnom radu je obrađena tematika igre igranja uloga s fantasy tematikom unutar Unity-a, gdje je glavni zadatak igrača pobijediti Skulleton King-a i spasiti sve čuvare. U igri postoji nekoliko quest-ova koji se moraju obaviti kako bi se igra uspješno završila. Glavni lik se zove Philip, te je njegov cilj pobijediti zlog Skulleton King-a te pronaći izlaz iz katakombi unutar kojih se nalazi. Kako bi uspio u svojim zadatcima tu je dobri duh Zhuti koji mu pomaže tokom igre. Na samom početku Zhuti objašnjava igraču kako se kretati, vršiti interakcije i boriti se s neprijateljima. Možda bi bilo najjednostavnije reći da Zhuti priprema igrača za samostalno igranje igre. Igrač tokom igranja susreće The Keepers-e kod kojih može, ovisno o tome s kojim The Keepers-om vrši interakciju, pričati, uzeti quest-ove ili uzeti item-e. Glavni lik se može kretati prema naprijed, nazad, lijevo i desno, te ima mogućnost bacanja spell-ova pomoću kojih pobjeđuje svoje protivnike. Skulleton-i (protivnici) imaju iste mogućnosti kao i glavni lik, te samim napretkom glavnog lika unutar igre postaju jači. Na samom kraju igre glavni lik i igrač se moraju suočiti sa Skulleton King-om koji je final boss igre. Nakon što ga glavni lik porazi javlja se Zhuti koji igraču čestita na porazu Skulleton King-a te govori kako je uspješno prešao, odnosno završio igru.

Ključne riječi: igra igranja uloga, Philip, RPG, spell, Unity, quest

ABSTRACT

In this final paper was created fantasy role-playing game inside Unity, where the main goal of the player is to beat Skulleton King and save The Keepers. In this game there are few quests that have to be completed in order to finish the game. Main character is Philip and his goal is to beat evil Skulleton King and find the way out of catacombs. To succeed in his tasks there is good spirit Zhuti who is helping during the game. In the very beginning Zhuti explains to the player how to move, interact and fight with the enemies. Maybe the simplest to say would be that Zhuti is preparing the player for self-playing the game. Player, durning the game, meets The Keepers with whom he can, depending on The Keeper, interact, speak, take quest or take items. Main character can move up, down, left and right, and he can throw spells that help him to defeat his enemys. Skulletons (enemys) have same abilities as main character, and with progress of main character in game they are becoming stronger. In the very end of the game, player and main character must face Skulleton King who is the final boss of the game. After wining against Skulleton King, Zhuti comes up and congratulates the player for defeating Skulleton King and he tells him that he successfully compleated the game.

Key words: role-playing game, Philip, RPG, spell, Unity, quest

ŽIVOTOPIS

Željko Sabo rođen je 13.07.1995. godine u Bjelovaru. U ranoj dobi je otkrio ljubav prema video igrama, pa tako i ljubav prema programiranju. Od 2002. do 2010. godine je pohađao OŠ Lipik gdje se je već u petom razredu upoznao s programiranjem na dodatnim nastavnim aktivnostima. Nakon što je završio osnovnu školu odlučio je upisati Tehničku školu Daruvar u Daruvaru, smjer tehničar za računarstvo gdje je nastavio usavršavati svoja znanja koja je stekao u osnovnoj školi. Nakon što je završio srednju školu, iste godine je upisao Elektrotehnički fakultet u Osijeku koji još uvijek pohađa i gdje nastoji proširiti i usavršiti svoja znanja.