SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

ODSJEK ZA INFORMACIJSKU TEHNOLOGIJU

LUKA ATELJ

ZAVRŠNI RAD

IZRADA PACMAN 3D IGRE NA UNITY PLATFORMI

Split, srpanj 2016.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

ODSJEK ZA INFORMACIJSKU TEHNOLOGIJU

PREDMET: Programske metode i apstrakcije

ZAVRŠNI RAD

KANDIDAT: Luka Atelj

TEMA ZAVRŠNOG RADA: Izrada Pacman 3D igre na unity platformi

MENTOR: Ljiljana Despalatović

Split, srpanj 2016.

Sadržaj

Sažetak

Za završni rad sam izabrao temu "3D Pacman" koja je kreirana na Unity platformi. Cilj igre je skupiti sve bodove prije isteka vremena te igrač mora paziti da se tokom igre ne dogodi kolizija sa ostalim objektima. Unity nam omogućuje da se nakon izrade igre ona može igrati na skoro svim platformama. Za igru su potrebni modeli kako bi vizualno bila zanimljivija. Za modele sam koristio Blender alat te sam kreirane modele u njemu uključio u Unity.

Summary

3D game in Unity platform

For the final work I have chosen the theme "3D Pacman" that was created on the unity platform. The aim of the game is to collect all the points before time runs out and the player must pay attention that the player does not collides with other objects. Unity allows us after making game that it can be played on almost all platforms. For the game we need models to make the game visually more interesting. For models i used Blender tool and included created models in Unity.

1 Uvod

Tema završnog rada je izrada igre na unity platformi. Unity platforma je za mnoge najbolja platforma za izradu igara jer je jednostavna za shvatiti te uz malo znanja c sharp ili java script programskog jezika svako može kreirati željenu igru. U svom radu koristio sam c sharp programski jezik jer pruža više mogućnosti za rad. Kolegiji koji su mi pomogli za izradu igre su: programske metode i apstrakcije, programiranje u c sharp-u, napredno windows programiranje i strukture podataka i algoritmi. Kroz izradu igre sam se postepeno upoznavao sa Unity platformom koja je iznimno korisna i jednstavna za izradu igara na željenim konzolama. Za ovakav rad me motivirala želja za izradom igara te trenutno tržište prodaje igara koje raste iz dana u dan. U drugom poglavlju su navedene i kratko opisane tehnologije koje su se koristile pri izradi igre. To su Unity i Blender. Treće poglavlje opsežno opisuje unity platformu kako bi se predočila kompleksnost unity-a. Osnove koje se trebaju znati za izradu igre se nalaze u ovom poglavlju. U četvrtom poglavlju su opisane scene u igri. Ova igra sastoji se od dvije scene "MainMenu" i "Game". Svaka scena za sebe ima svoje objekte i skripte. Peto poglavlje prikazuje i opisuje sve objekte koji su korišteni u igri. Bez objekata igra nije igriva te su sve komponente svakog objekta objašnjene u petom poglavlju. Šesto poglavlje opisuje klase. Klase upravljaju objektima i u ovom poglavlju se vide sve funkcionalnosti svakog objekta. Sedmo poglavlje su animacije. U ovom poglavlju je opisana izrada animacija i objašnjenje zašto su animacije važne. Osmo poglavlje objašnjava važnost zvuka i njegovu upotrebu a u devetom poglavlju se opisuju mogučnosti izrade igre na različitim platformama.

2 Korištene tehnologije

Za izradu ovog rada koristili su se unity i blender alati.

2.1 Unity

Unity je najbrže rastući game engine za izradu 2D i 3D igara na svijetu. Jednostavan je za korištenje, može se raditi u timu, podržava bitne platforme kao što su: PC, Mac, Linux, Android, iOS, xBox, Play Station, itd. Sve što od vas zahtjeva je znjanje c sharp programskog jezika. Može se koristiti i java script programski jezik, ali puno je jednostavnije zbog pronalazaka pogreški tokom pisanja skripte, koristiti c sharp programski jezik. Skripte se mogu pisati u bilo kojem editoru. Ukoliko korisnik želi napraviti unikantnu igru bilo bi poželjno da zna samostalno izrađivati 3D ili 2D objekte. Samostalno izrađivanje objekata nije nužno jer danas postoji mnogo gotovih objekata koje korisnik može besplatno preuzeti i koristiti se s njima. Unity se može besplatno koristiti te se može na njemu zarađivati. Ukoliko autor neke igre zaradi preko 100.000 dolara tada mora platiti profesinalnu verziju Unity-a.

2.2 Blender

Blender je besplatan alat za za izradu 3D računalne grafike koji se koristi za izradu animiranih filmova, vizualnih efekata, modela za 3D printere, itd. Blender je razvila softverska kuća "Not a Number Technologies". Radna površina blendera je podjeljena na tri djela: glavni meni na vrhu, središnji 3D prostor za modeliranje i prostor sa dugmićima i opcijama na dnu prozora. Nakon što se kreira model potrebno je kreirati datoteku sa 3ds ekstenzijom te je takvu ubaciti u Unity alat.

3 Unity

3.1 Sučelje

Glavno sučelje unity-a sadrži više prozora koji se mogu grupirati po želji korisnika. Sučelje se sastoji od alatne trake, hierarhijskog prozora, pogleda na scenu, inspektora i prozora projekta.

3.1.1 Prozor projekta

Prozor projekta prikazuje sve datoteke koje se koriste u projektu. Preporučava se napraviti direktorije u kojima se raspoređuju skripte, zvukovi, animacije i ostali objekti. Kada uključujemo neke segmente u projekt oni će se pojaviti u prozoru projekta.

3.1.2 Alatna traka

Alatna traka omogućuje najbitnije funkcionalnosti za kreiranje igre. Na lijevoj strani se nalaze alati pomoću kojih možemo upravljati sa scenom i sa objektima koji su u sceni. Objektima možemo postaviti visinu, dužinu, koordinate trenutnog položaja itd. Na sredini se nalaze "play", "pause" i "stop" kontrole koje omogućuju testiranje igre. Na desnoj strani se nalaze kontrole za oblak servis i osobni unity račun te kontrole za postavljanje rasporeda sučelja po korisnikovoj želji.

3.1.3 Pogled na scenu

Scena nam prikazuje 3D ili 2D objekte, ovisno o vrsti projekta na kojem radimo. Pogled na scenu omogućava programeru da vizualno upravlja svojom igrom. Osim pogleda na scenu imamo i pogled na igru (game view) koji programeru omogućava testiranje svoje igre te pronalaženje pogrešaka. Pogled na igru koristimo da ne trebamo nakon svake izmjene u igri ponovno izrađivati igru. On se pokrene tako da na alatnoj traci odaberemo opciju "play". Glavne opcije pogleda na igru su "Free Aspect", "Scale", "Maximize on play", "Mute audio" i "Stats".

- "Free Aspect" nam omogučuje da testiramo kako će igra izgledati na različitim proporcijama ekrana.
- "Scale" nam omogučuje da zumiramo igru da vidimo detalje koje se nalaze u igri.
- "Maximize on play" postavka nam služi da vidimo igru preko cijelog zaslona dok je testiramo u pogledu na igru.

- "Mute audio" postavka nam služi da tokom testiranja igre isključimo ili uključimo zvuk.
- "Stats" nam prikazuje sve statistike zvuka i grafike u igri.

3.1.4 Hierarhiski prozor

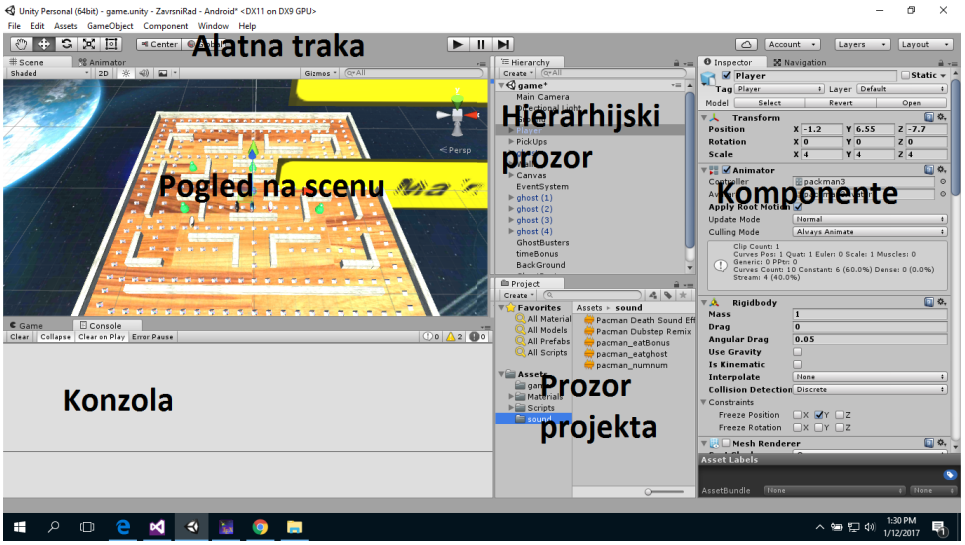
Ovdje možemo vidjeti kako su objekti pridruženi drugim objektima vizualno, što pomaže pri pisanju koda. Hierarhiski prozor će prikazati one objekte koji su u sceni koju trenutno uređujemo. Klikom na neki objekt inspektor će prikazati njegove komponente. U hierarhiskom prozoru možemo desnim klikom duplicirati bilo koji objekt. Također možemo vidjeti relacije roditelja i djece.

3.1.5 Konzola

Konzola nas obavještava kada se dogodi neka greška, upozorenje ili iznimka. Konzola je dosta praktična jer preko nje možemo saznati što nam se u kojem trenutku događa u projektu i to preko funkcije:
"Debug.Log();".

3.1.6 Inspektor

Inspektor omogućuje da imamo uvid u sva svojstva trenutno odabranog objekta. Preko njega možemo mjenjati komponente objekta. Odabirom nekog objekta inspektor ce automatski komponente tog objekta prikazati u svom prozoru. U koliko nije nijedan objekt selektiran inspektor će biti prazan. U koliko je neki objekt napravljen u nekom drugom alatu inspektor nudi opciju "open" koja će korisnika odvesti u taj alat i u njemu prikazati selektirani objekt. Ova opcija je dosta korisna jer ako smo navikli raditi objekte u drugom alatu i vidimo u pogledu na igru da neki dio objekta želimo izmjeniti možemo vrlo jednostavno u tom alatu izmjeniti objekt i nastaviti ga normalno koristiti u unity-u.



Slika 1: Prikaz sučelja

3.2 Assets

Assets-i su ukjučeni ili kreirani objekti. Assets-i mogu biti 3D modeli, zvuk, tekst itd. Kada neki objekt uključimo u unity on će se automatski pojaviti u asset direktoriju.

3.3 Objekti

Sve što se nalazi na sceni je objekt. Objekti se mogu kreirati u unity-u ili pridružiti ih iz drugog alata. Svaki objekt u sebi sadrži sljedeće osobine: ime, tag, sloj i static. Autor sam može proizvoljno dati ime svom objektu po želji. Tagovi su jako poželjni kada se referenciraju objekti u kodu. Traženje objekata preko tagova olakšava posao programeru pogotovo kada se radi na većim projektima. Kada više različitih objekata rade istu stvar onda je poželjno koristiti tag. Objekti mogu biti dodjeljeni drugom sloju. Static se koristi za objekte koji su statični. Svaki objekt ima svoje komponente koje se nalaze u inspektor prozoru.

3.4 Komponente

Svaki objekt mora sadržavati neke komponente. Pri kreiranju objekta njemu će se dodjeliti zadana "Transform" komponenta koja prikazuje poziciju, rotaciju i visinu objekta. Bez ove komponente objekt bi bio nevidljiv na sceni. Komponente u inspektor prozoru možemo poredati po želji. Neke od komponenata su "Character Controller", "Nav Mesh Agent", "Rigidbody" itd.

3.5 Montaža (Prefab)

Prefab možemo gledati kao objekt na koji je korisnik postavio instancu. Kada se napravi neka promjena na prefabu ta promjena će se primjeniti za instance tog objekta.

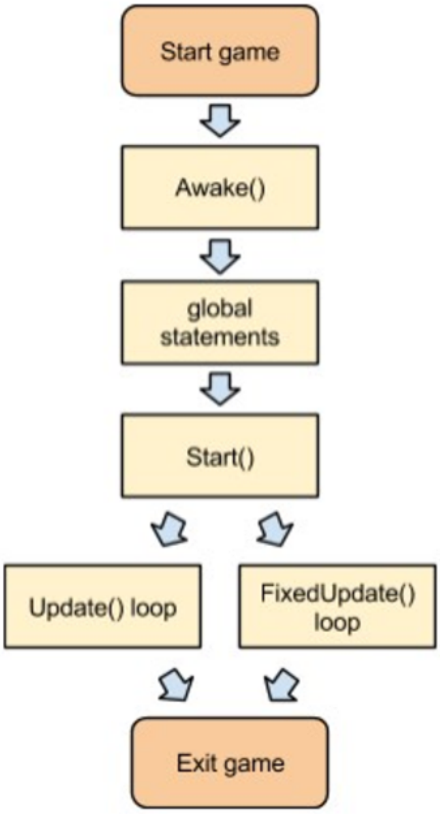
3.6 Pisanje koda

3.6.1 Programski jezici

U unity platformi se može kreirati igra korištenjem jednog od tri programska jezika,a oni su c sharp, java script i Boo. Prednosti c sharp programskog jezika prema java scriptu je to što se lakše otkrivaju pogreške u kodu. Java script se često koristi u unity-u. Često se naziva i Unity Script. Prednosti java scripta prema c sharp-u su jednostavnija sintaksa java script programskog jezika i mnogo primjera koje se mogu naći na internetu. Programski jezik Boo se jako malo koristi u unity-u i ima samo nekolicinu primjera na internetu što korisnika automatski udaljuje od ovog jezika.

3.6.2 Petlje

Kada se kreira skripta unity će automatski kreirati "Start()" funkciju i "Update()" petlju. "Awake()" funkcija je funkcija koja će se prva pozvati kada se igra pokrene i ona se poziva samo jednom. "Awake()" funkcija se najčešće koristi da se u nju postave reference između skripti jer se objekti još nisu izvršili. "Start()" funkcija će se pozvati nakon "Awake()" funkcije, ona se poziva kada se svi objekti iz scene izvrše. "Start()" funkcija će se također pozvati samo jednom. "Start()" funkcija je najbolje mjesto za postaviti refence objekata. Nakon funkcije "Start()" unity ce pozvati petlju "Update()". Postoje dvije takve petlje a to su "Update()" i "Fixedupdate()". Petlja se izvršava onoliko puta po sekundi koliko je računalo može procesuirati. Razlika između ove dvije petlje je ta što se "FixedUpdate()" izvršava u određenom vremenu.



Slika 2: Arhitektura igre

3.7 Fizika

Svaki objekt koji ima veze sa fizikom mora sadržavati "RigidBody" i "Collider" komponente. Glavne komponente "RigidBody-a" su "Use Gravity" i "Is Kinematic". "Use Gravity" jednostavno znači da će se tom objektu postaviti gravitacija s obzirom na komponentu "Mass". Komponenta "Is Kinematic" se koristi kada korisnik želi tokom igre upravljati sa objektom preko transfom funkcije. "Use Gravity" i "Is Kinematic" komponente su tipa bool i njih postavljamo u inspektoru.

3.8 Kamera

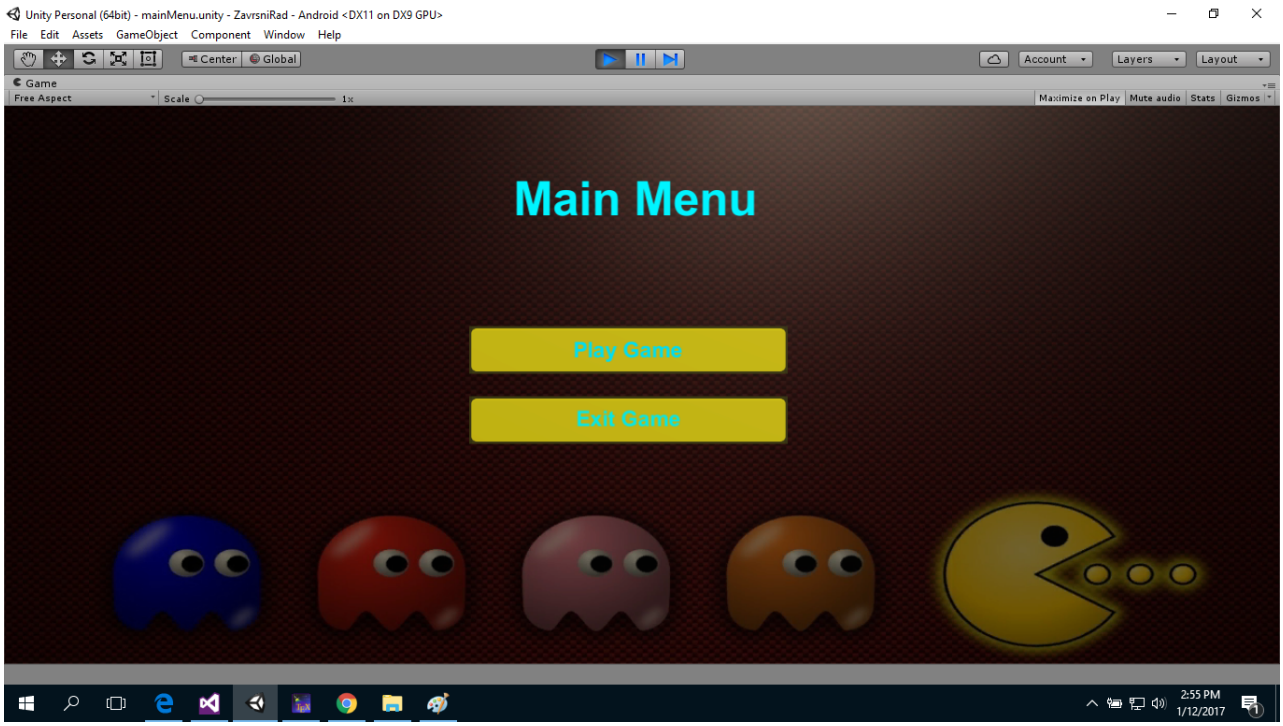
Svaka scena mora sadržavati barem jednu kameru. Kada se scena kreira, kreira se i objekt glavna kamera. Preko kamere određujemo da li je igra "first person game" ili "third person game". Zadane komponentne kamere nam omogučuja da u startu postavimo širinu pogleda, dizajn neba, projekciju, početnu poziciju i pozadinu.

4 Scene

Scena sadrži sve objekte igre koji se koriste. Scena nam može služiti za kreiranje izbornika, ukoliko igra ima više levela, svaki level će se nalaziti u svojoj sceni. Preko scene uređujemo okolinu, postavljamo objekte i prepreke. Kada selektiramo neki objekt u sceni pojavit će se strelice preko kojih možemo objekt pomicati prema x, y i z osi, možemo povećavati, smanjivati ili rotirati objekt ovisno o selektiranoj opciji na alatnoj traci. Scena zapravo sve sklopljene elemente pretvara u jedan sadržaj.

4.1 Main menu

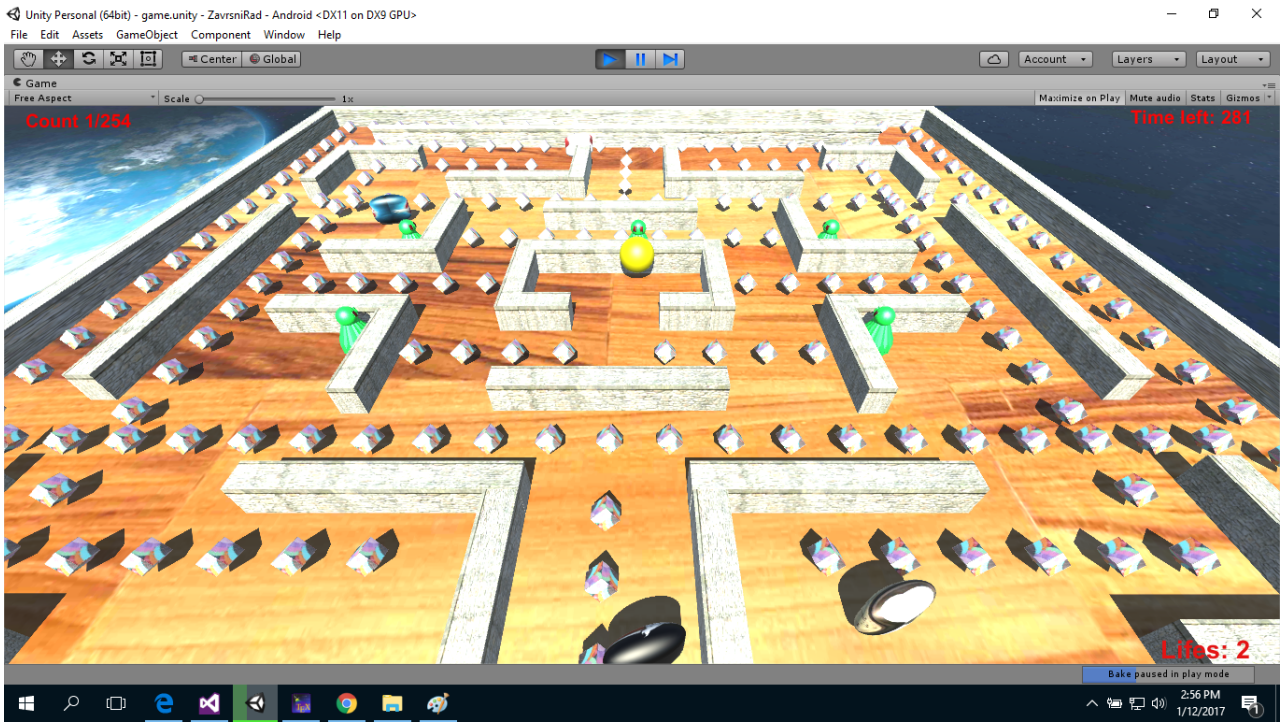
Ovo je početna scena u igri. Scena sadrži platno sa tekstom i dva botuna te objekt "plane" na kojega je zaljepljena slika radi boljeg ugođaja. Klasa koja upravlja sa ovom scenom smještena je u "Main Camera" objektu. Pritiskom na jedan od botuna pozove se događaj "On Click()".



Slika 3: Scena "Main Menu"

4.2 Game

"Game" je glavna scena na koju korisnik dolazi pritiskom botuna "Play" na "Main Menu" sceni. Sadrži sve dolje napisane objekte i njihove komponente. Također sadrži i platno koje se pojavljuje po potrebi i preko kojeg se može vratiti na početnu scenu. Platno na "Game" sceni i platno na "Main menu" sceni imaju istu skriptu "Main Menu" s kojom se navigira između ove dvije scene. Svaka scena koristi svoje funkcije. "Main Menu" scena ima mogučnost izlaska iz igre i prelazak na scenu "Game", a scena "Game" ima mogučnost prelaska na scenu "Main Menu" i ponovno igranje igre. Platno u "Game" sceni se pojavljuje ukoliko jedan od objekata "Ghost" se sudari sa glavnim objektom "Player" ili ako istekne određeno vrijeme predviđeno za prelazak igre. Objekti "Ghost" i "Player" su napravljeni u blender alatu dok su svi ostali objekti u ovoj sceni napravljeni u unity-u. Na ove objekte su zaljepljene slike radi boljeg ugođaja tokom igranja igre.



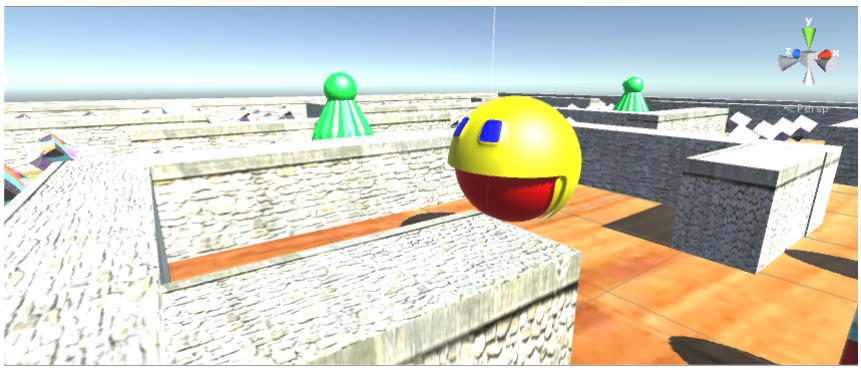
Slika 4: Scena "Game"

5 Objekti

5.1 Player

Ovo je glavni objekt u igri. Napravljen je od više objekata u blenderu. S ovim objektom upravlja klasa "packmanController". Sadrži sljedeće komponente:

- Transform –> Zadana komponenta svakog objekta koja određuje poziciju, rotaciju i veličinu objekta.
- Rigidbody –> Kontrolira poziciju objekta kroz fizičku simulaciju. Ovaj objekt ne koristi fiziku, ali je iskorištena njegova komponenta da se onemogući kretnja objekta po y osi.
- Sphere collider –> Služi za detekciju kolajdera u obliku kule. Time možemo postaviti veličinu radiusa za kolajder.
- Audio source –> Pridružen je objektu zbog ispuštanja zvuka kojeg objekt kontrolira.
- Animator –> Služi za kontrolu animacije objekta. U ovoj igri aniamcija je napravljena u blender alatu. Objekt preko ove komponente upravlja sa svojom animacijom.



Slika 5: Objekt "Player"

5.2 PickUps

Ovo je prazan objekt u kojemu su smješteni objekti "PickUp" koje igrač skuplja u igri. S tim objektima upravlja klasa "Rotator". Ovi objekti imaju i tag naziva "pickUp" kako bi se lakše pristupilo svim objektima odjednom. Njihove komponente su:

- Transform
- Box Collider –> Služi za detekciju kolajdera u obliku kocke. Možemo postaviti x, y i z veličinu kolajdera.

- Mesh renderer –> Pomoću njega dodjeljujemo materijale objektu(boja), možemo definirati veličinu i sjenu.

5.3 Ghost

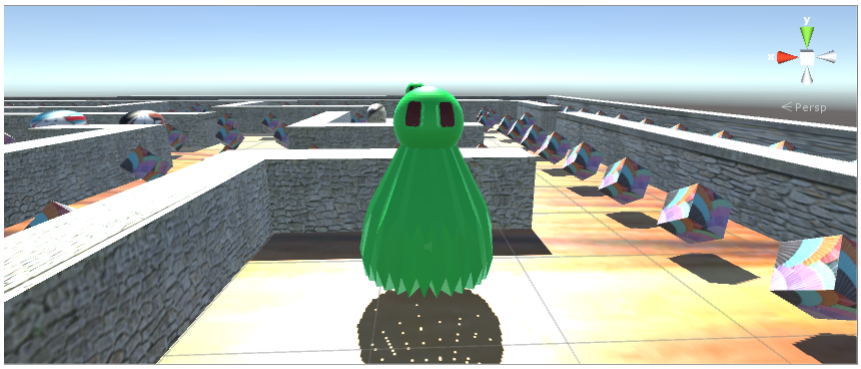
Postoje pet istih objekata "Ghost". Ovaj objekt je napravljen u blender alatu i kasnije se duplicirao u unity-u. Njima je pridružen tag "ghost" i "ghostCollor". Preko taga možemo upravljat s njima iz skripte drugog objekta. Njima upravlja klasa "ghostController" i svi imaju iste komponente:

- Transform
- Mesh renderer
- Capsule collider –> Služi za detekciju kolajdera u obliku kapsule. Možemo postaviti visinu i radijus kolajdera. Kolajder sadrži svojstvo "isTrigger" koje se koristi za "triggering" događaje.

```
void OnTriggerEnter(Collider other)
{
other.gameObject.SetActive(false);
}
```

Ispis 1: Prikaz funkcije "OnTriggerEnter()"

- Character controller –> Omogućuje kretanje objekta bez da se koristi Rigidbody i da se sudara sa drugim objektima tj. da ne prolazi kroz njih.
- Audio source
- Animator
- Nav Mesh Agent –> Omogućuje nam da se objekti ne sudaraju međusobno i sa drugim objektima tokom kretnje. Jedna od prednosti ove komponente je da će se objekt uvjek rotirati prema smjeru kretnje. Radius i visina objekta nam služi da bi odredili koliziju sa drugim objektima.



Slika 6: Objekt "Ghost"

5.4 Bonuses

Imamo četiri različita objekta bonus koji imaju iste komponente ali služe za različite stvari koje se kontroliraju u "packmanController" klasi. Ovim objektima je pridružena skripta Yrotator koja ih rotira po y osi radi boljeg ugođaja tokom igranja igre. Nazivi objekata su: "ghostBusters", "ghostBuster", "lifeBonus" i "timeBonus". Njihove omponente su:

- Transform
- Mesh renderer
- Capsule colider

5.5 Ground

Ovaj objekt je podloga na kojoj se igra odvija. Komponente podloge su:

- Transform
- Mesh renderer

5.6 BackGround

Ovaj objekt je podloga koja se nalazi ispod glavne podloge na koji je nalijepljena slika zbog boljeg ambijenta tokom igranja. Ima iste komponente kao objekt "ground".

5.7 Platno

Platno je objekt koji u sebi sadržava sve UI elemente. Kada želimo dodati neki tekst ili sliku automatski se kreira platno koje će u sebi sadržavati tu sliku ili tekst. Platno je roditelj svih UI elemenata. Ukoliko smanjujemo ili povečavamo prozor projekta platno će se automatski prilagoditi veličini prozora. Ovaj objekt u sebi sadži pet tekst komponenti i dva botuna. Tekst komponente služe za ispisivanje određenog sadržaja, a botuni za navigaciju kroz scene.

5.8 Direction light

Služi nam za postavljanje pozicije svjetla, namještanje sjene, jačina svijetla itd. Te opcije se nalaze u njenoj zadanoj komponenti "Light". Kao i svi drugi objekti sadrži transform komponentu koja određuje iz kojeg će kuta dolaziti sjena.

5.9 Main Camera

Glavna kamera u igri koja prati igrača tokom njegove kretnje. Kretnju tokom igre mu omogućuje klasa "cameraController" koja je priključena na ovaj objekt. Objektu je još dodjeljena i "Main Menu" klasa koja navigira između scena. "Main Camera" ima i svoju zadanu komponentu "Camera" gdje možemo postaviti boju neba, širinu pogleda kamere, pozadinu itd.

6 Klase

Svakom objektu je dodjeljena svoja zasebna klasa. Svaka klasa se nalazi u zasebnoj skripti. Klase koje nasljeđuju MonoBehavior ne smiju imati konstruktore, to su bazne klase. Klase koje su kreirane za ovu igru su "mainMenu", "cameraControler", "Rotator", "Yrotator", "ghostController" i "packmanController".

6.1 Main Menu

Klasa "mainMenu" se nalazi u oba dvije scene. Ona upravlja sa platnom koji sadrži botune koji su postavljeni preko "onClick()" funkcije.

- void Start() –> Omogučava prikaz objekata.
- public void exit() –> Omogučava zatvaranje igre.
- public void playGame() –> Otvara "game" scenu.
- public void menu() –> Otvara "mainMenu" scenu koja će se pozivati iz "game" scene. Prikaz klase se nalazi u sljedečem ispisu.

```
public class MainMenu : MonoBehaviour {
public Canvas mainMenu;
void Start () {
mainMenu.enabled = true;}
public void exit()
{
mainMenu.enabled = false;
Application.Quit();
}
public void playGame()
{
SceneManager.LoadScene("game");
}
public void menu(){
SceneManager.LoadScene("mainMenu");
}}
```

Ispis 2: Prikaz klase "MainMenu"

6.2 Camera Controller

Klasa "cameraController" se nalazi u "game" sceni i ona kontrolira kretanje kamere tokom igre. Funkcije koje se koriste:

- void Start () –> Postavlja kameru na početnu poziciju s obzirom na poziciju igrača.
- void Update() –> Prati kretanje igrača, pomiče i rotira kameru u tom smjeru.

```
public class cameraController : MonoBehaviour {

public GameObject packman;

private Vector3 offset;
// Use this for initialization
void Start () {
offset = transform.position - packman.transform.position;

}
// Update is called once per frame
void Update () {
transform.position = packman.transform.position + offset;
}
}
```

Ispis 3: Prikaz klase "cameraController"

6.3 Rotator

Klasa "Rotator" koristi samo Update() petlju. Ova klasa služi za objekte koji će se rotirati po x, y, z osi. Ona upravlja sa "pickUp" objektima. Klasa "Rotator" je prikazana u sljedečem ispisu.

```
public class Rotator : MonoBehaviour {

// Update is called once per frame
void Update () {
transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
}
}
```

Ispis 4: Prikaz klase "Rotator"

6.4 Yrotator

Klasa "Yrotator" radi isto što i klasa "Rotator" ali rotira objekte samo po y osi. Ona upravlja sa objektima koji predstavljaju bonuse.

6.5 Ghost Controller

Klasa koja upravlja s objektima "Ghost". Ti objekti imaju komponentu NavMeshAgent. NavMeshAgent komponenta omogućuje objektima koji se kreću ka istom cilju da se ne sudaraju.

- void Start () –> Postavlja "NavMeshAgent" komponentu objektima "Ghost". U skriptu je dodana komponenta AudioSource koja će izpustiti zvuk kada jedan od objekata "Ghost" se sudari sa objektom "Player". Postavlja se brzina objekta:
  "Ghost.speed = 30;".
  Objekt "Player" smo dohvatili preko svojstva "Tag":
  "packMan = GameObject.FindGameObjectWithTag("Player");"

- void Update() –>Funkcija koja objektima "Ghost" postavlja destinaciju:
  "Ghost.destination = packMan.transform.position;".

- void OnTriggerEnter(Collider other) –> Funkcija koja detektira kada ce se objekti "Ghost" sudariti sa objektom "Player". U koliko se to dogodi aktivira se AudioSource komponenta. Također ukoliko se objekti "Ghost" sudare sa nekim objektom koji predstavlja bonus, aktivacija tog objekta će se postaviti na "false".
  Prikaz funkcije se nalazi u sljedečem ispisu.

```
void OnTriggerEnter(Collider other)
{
if (other.gameObject.CompareTag("Player"))
{
audio.PlayOneShot(death);
}
if (other.gameObject.CompareTag("ghostBusters"))
{
other.gameObject.SetActive(false);
}
if (other.gameObject.CompareTag("ghostBuster"))
{
other.gameObject.SetActive(false);
}
```

```
if (other.gameObject.CompareTag("timeBonus"))
{
other.gameObject.SetActive(false);
}
if (other.gameObject.CompareTag("lifeBonus"))
{
other.gameObject.SetActive(false);
}
}
```

Ispis 5: Prikaz funkcije "OnTriggerEnter()"

6.6 Packman Controller

Glavna klasa s kojom se kontrolira igrača i njegove komponente. Klasa je pridružena objektu "Player" koji je glavni objekt u igri.

Reference:

- public GameObject packman –> Referenca na igraca.
- public GameObject ghostBusters –> Referenca na bonus.
- public GameObject ghostBuster –> Referenca na bonus.
- public GameObject bonusTime –> Referenca na bonus.
- public GameObject bonusLife –> Referenca na bonus.
- public AudioClip numnum –> Referenca na određeni zvuk.
- public AudioClip eatGhost –> Referenca na određeni zvuk.
- public AudioClip eatBonus –> Referenca na određeni zvuk.
- public Text countObjects –> Referenca na text koji ispisuje koliko je objekata skupljeno.
- public Text time –> Referenca na text koji ispisuje koliko je vremena igraču ostalo.
- public Text winText –> Referenca na text koji se ispisuje ako igrač skupi sve "PickUp" objekte.
- public Text lifes –> Referenca na text koji ispisuje koliko igrač ima života.
- public Text countObjects –> Referenca na text koji ispisuje koliko je igrač skupio bodova.

- public Text pause –> Referenca na text koji ispisuje "Pause" kada korisnik pritisne bilo koji dio ekrana.
- private Vector3 up, right, left, down, currentDirection –> Određuje smjer kretnje igrača.
- int count –> Broji koliko je "PickUp" objekata ostalo.
- public float timeLeft –> Broji koliko je vremena ostalo.
- public GameObject[] ghostColor –> Niz djela "Ghost" objekta za bojanje.
- public GameObject[] ghosts –> Niz svih "Ghost" objekata.
- private Color color –> refrenca na boju.
- int lifeCount –> Broji koliko je života ostalo igraču.
- int tempTime –> Broji koliko je vremena ostalo do kraja igre.
- float bonusTimeLeft –> Broji koliko je vremena ostalo dok su bonusi aktivni.
- private float currentAcceleration –> Broj koji određuje poziciju y osi.

Metode:

- void Start () –> Inicializira sve reference.
- void finalText(int final) –> Metoda koja ispisuje jeli korisnik pobjedio ili izgubio. Funkcija je prikazana u sljedečem ispisu.

```
void finalText(int final)
{
if (final == 0)
{
winText.text = "You Win!";
winText.color = Color.blue;
mainMenu.gameObject.SetActive(true);
playAgain.gameObject.SetActive(true);
}
else
{
winText.text = "Game Over!";
winText.color = Color.red;
mainMenu.gameObject.SetActive(true);
playAgain.gameObject.SetActive(true);
lifeCount = 0;
```

```
packman.gameObject.SetActive(false);
}
}
```

Ispis 6: Prikaz funkcije "finalText()"

- void ghostStatus(bool status) –> Postavlja aktivnost objekata Ghost.
- void changeGhostColor(Color col) –> Postavlja boju objekata Ghost.
- Vector3 randomPosition() –> Vraća random vektor 3 za postavljanje pozicija bonusa na mapi.
- void pausePhone() –> Korisnik dodirom na ekran poziva ovu funkciju. Funkcija pauzira igru tako da zamrzne sliku i postavi aktivno platno koje ispisuje odgovarajući tekst i aktivira botun koji vodi korisnika na glavni izbornik. Ukoliko korisnik želi nastaviti igru treba opet pritisnuti dio ekrana na kojem se ne nalazi ovaj botun.

  Funkcija je prikazana u sljedečem ispisu.

```
void pausePhone()
{
if (Input.touchCount > 0)
{
paused = !paused;
}
if (paused)
{
Time.timeScale = 0;
pause.text = "Paused";
mainMenu.gameObject.SetActive(true);
}
else
{
Time.timeScale = 1;
pause.text = "";
mainMenu.gameObject.SetActive(false);
}}
```

Ispis 7: Prikaz funkcije "pausePhone()"

- void pausePc() –> Ova funkcija služi kada se igra koristi na PC-u, Mac-u ili Linux-u. Funkcija pauzira igru pritiskom na tipku "P".

Ova funkcija ima istu ulogu kao "pausePhone()" funkcija, ali se koristi za gore navedene platforme.

- void activateBonuses() –> Funkcija koja svakih 30 sekundi aktivira bonuse i drži ih aktivnima 10 sekundi. Nakon što prođe 10 sekundi bonusi se deaktiviraju i pojavljuju opet za 20 sekundi.

- void checkGhost() –> Funkcija koja provjerava aktivnost objekata "Ghost". Nakon što igrač skupi određeni bonus objekti "Ghost" se deaktiviraju na 10 sekundi. Ova funkcija provjerava da li je prošlo 10 sekundi od skupljanja bonusa te ako je aktivira objekte Ghost.

- void setFinalEnviroment() –> Funkcija koja provjerava je li igrač ispunio uvjete da se pozove metodu finalText(int final). Provjerava je li igrač ima dovoljno života, je li preostalo vrijeme za igru još aktivno i koliko je igrač skupio bodova.

- void setCanvas() –> Tokom igre na ekranu stoje 3 teksta. Jedan je za broj bodova koje je igrač skupio, drugi je za prestalo vrijeme igre a treći za broj života koji je igraču ostao. Funkcija prikazuje na ekranu reference:

  "time.text

  "lifes.text"

  "countObjects.text"

- float? getPhonePosition() –> Nakon svake dvije sekunde vrača poziciju y osi. Prikaz funkcije se nalazi u sljedečem ispisu.

```
float? getPhonePosition()
{
int tl = (int)Mathf.Round(timeLeft);
float tmpcurrentAcceleration;
if (tl % 2 == 0)
{
tmpcurrentAcceleration = Input.acceleration.y;
return tmpcurrentAcceleration;
}
else
{
return null;
}}
```

Ispis 8: Prikaz funkcije "getPhonePosition()"

- void playerMoveAcceleration() –> Korisnik upravlja sa glavnim objektom "Player" preko akceleracije mobitela. Problem ovakvog upravljanja sa objektima je taj što se mobitel treba držati uvjek u istom položaju. Zato u sebi ova metoda sadrži metodu getPhonePosition(). Tako korisnik može igrati igru u bilo kojem položaju jer se svake dvije sekunde trenutna pozicija y osi postavlja na zadanu poziciju.

```
void playerMoveAcceleration(){
float x = Input.acceleration.x;
float y = Input.acceleration.y;
if (currentDirection != down)
{
if (y > currentAcceleration && (x < 0.04f || x > -0.04f))
currentDirection = up;
if (x > 0.04f)
{
currentDirection = right;
y = currentAcceleration;
}
if (x < -0.04f)
{
currentDirection = left;
y = currentAcceleration;
}
if (y < currentAcceleration - 0.02f && (x < 0.04f || x > -0.04f))
currentDirection = down;
}
if (currentDirection != up)
{
if (y > currentAcceleration + 0.02f && (x < 0.04f || x > -0.04f))
currentDirection = up;
if (x > 0.04f)
{
currentDirection = right;
y = currentAcceleration;
}
if (x < -0.04f)
{
currentDirection = left;
y = currentAcceleration;
```

```
}
if (y < currentAcceleration && (x < 0.04f || x > -0.04f))
currentDirection = down;
}}
```

Ispis 9: Prikaz funkcije "playerMoveAcceleration()"

- void playerMovePC() –> Funkcija za opravljanje objektom "Player" preko tipkovnice računala. U sljedečem ispisu su prikazane kontrole za računalo.

```
void playerMovePc()
{
if (Input.GetKey(KeyCode.UpArrow))
{
currentDirection = up;
}
else if (Input.GetKey(KeyCode.RightArrow))
{
currentDirection = right;
}
else if (Input.GetKey(KeyCode.DownArrow))
{
currentDirection = down;
}
else if (Input.GetKey(KeyCode.LeftArrow))
{
currentDirection = left;
}
else
isMoving = false;
}
```

Ispis 10: Prikaz funkcije "playerMovePc()"

- void OnTriggerEnter(Collider other) –> Ova metoda detektira kada su se dva objekta sudarila. U ovoj metodi se nalaze najbitnije funkcionalnosti igre. Igrač se sudara sa objektima koje mora skupiti("Pick Up"), sa objektima koji ga hvataju("Ghost") i sa bonus objektima. Sa uvjetima je detektirano koji se sudar dogodio i time se vrši radnja nad tim objektima. Prikaz funkcionalnosti kolizije objekta "Player" sa objektom "Ghost" se nalazi u sljedečem ispisu.

```
if (other.gameObject.CompareTag("ghost"))
{
int count=0;
ghostColor = GameObject.FindGameObjectsWithTag("ghostColor");
foreach (GameObject ghost in ghostColor)
{
if (ghost.GetComponent<Renderer>().material.color == Color.black )
{
count++;
}}
if (count > 0)
{
bonusTimeLeft = 10;
audio.PlayOneShot(eatGhost);
foreach (GameObject ghost in ghosts)
{
if (other.gameObject == ghost)
ghost.SetActive(false);
}}
else
{
lifeCount--;
if (lifeCount > 0)
packman.transform.position = new Vector3(0, 4, 0);
else
{
packman.gameObject.SetActive(false);
finalText(1);
}}}
```

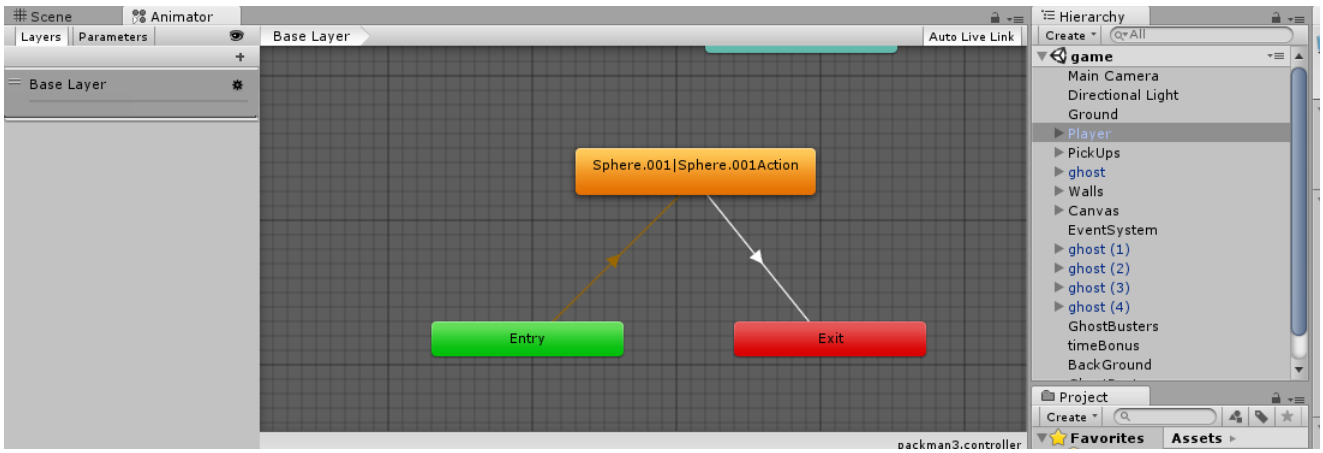
Ispis 11: Prikaz funkcionalnosti kolizije objekta "Player" sa objektom "Ghost"

7 Animacije

Igre se većinom rade tako da korisnik upravlja sa nekim objektom. Da bi taj objekt bio vizualno zanimljiv potrebno mu je dodati neke animacije. Animacije se u Unity-u rade u animacijskom prozoru. Animacijski prozor sadrži vremensku crtu koja nam služi da odredimo vrijeme animacije. Jedan objekt može imati više animacija. Sučelje za kontrolu svih animacija objekta se zove Animator kontrola.

7.1 Animator kontrola

Sa animator kontrolom određujemo kada će se koja animacija izvršiti. Najčešće je slučaj da objekt ima više animacija koje čekaju svoj red za izvršavanje. Animator kontrola tu postavlja redosljed izvršavanja animacija. Npr. ukoliko je određena animacija aktivna i kad korisnik pritisne određenu tipku ta animacija prelazi na drugu određenu animaciju. Na slici je prikazana animator kontrola za objekt "Player".



Slika 7: Dodjeljivanje animacija objektu

8 Zvuk

Da bi igra bila korisniku što poželjnija potrebno je u nju postaviti zvuk. Zvuk daje korisniku osjećaj da stvarno upravlja sa nekim objektom. Glavna komponenta zvuka u unity-u je "Audio Source". On reproducira zvuk u sceni preko "Audio Clip" svojstva. U tablici su prikazani formati zvuka koje podržava unity.

Supported formats

Format	Extensions
MPEG layer 3	.mp3
Ogg Vorbis	.ogg
Microsoft Wave	.wav
Audio Interchange File Format	.aiff / .aif
Ultimate Soundtracker module	.mod
Impulse Tracker module	.it
Scream Tracker module	.s3m
FastTracker 2 module	.xm

Slika 8: Formati zvuka koje podržava Unity

8.1 Postavljanje zvuka

Postavljanje zvuka u unity-u je vrlo jednostavno. Da bismo stvorili zvuk koji će se koristiti tokom igre potrebno je taj zvuk prvo prebaciti u unity projekt. Kada je zvuk u projektu dodjeli se objektu sa kojim ima komponentu "Audio Source". "Audio Source" komponenta sadrži mnogo svojstva. Jedno od svojstva je "Play on Awake" koje se koristi u "Main Menu" sceni. Ovo svojstvo pokreće željeni zvuk kada se igra pokrene. Unity svaku datoteku zvuka gleda kao "Audio Clip".

8.2 Audio Clip

"Audio Source" komponenta je beskorisna bez svojstva "Audio Clip". "Audio Source" je zapravo kontrola koja upravlja sa svojstvom "Audio Clip". Ako koristimo samo jedan zvuk dovoljno ga je samo ubaciti u objekt, a unity će sam napraviti komponentu i ubaciti taj zvuk u nju.

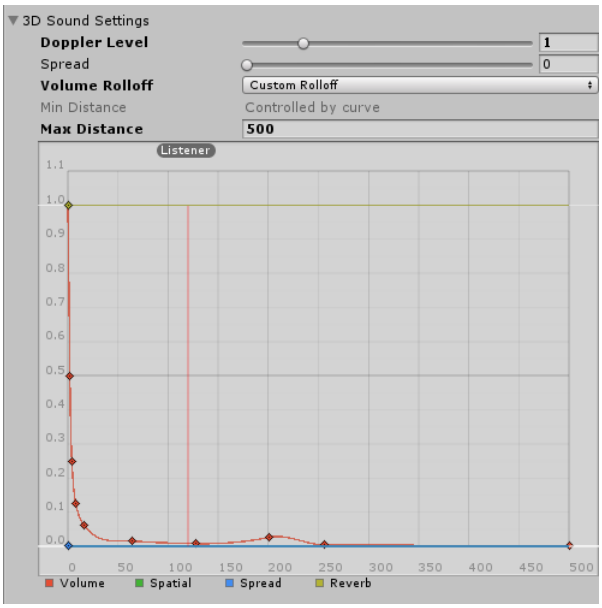
Primjer ispuštanja zvuka:

```
AudioSource audio;

public AudioClip death;

audio = GetComponent<AudioSource>();

audio.PlayOneShot(death);
```

Ispis 3: Prikaz ispuštanja zvuka

8.3 Zvučni oslušivač

Unity nam nudi više mogućnosti kako da prenesemo zvuk do korisnika. To nam omgućuju funkcije na daljinu (distance functions) koje se nalaze na kraju komponente "Audio Source" i mogu se namještati preko grafa. Ove funkcije se koriste da korisnik dobije dojam da se stvarno nalazi u igri. Možemo namjestiti kut odakle zvuk dolazi, jačinu zvuka, brzinu zvuka itd. Ove mogućnosti se koriste dosta za igre koje su kreirane za virtualnu realnost i ne mogu se koristiti za 2D igre.
Na sljedećoj slici je prikazan graf za zvučni osluškivač.



Slika 9: Prikaz grafa zvučnog osluškivača

9 Platforme

Postoji puno alata na kojima se mogu izrađivati igre. Kada kreiramo neku igru cilj nam je da ta igra se može koristiti na što više platforma. Naravno to ovisi o kompleksnosti igre. Neke od bitnijih platforma koje podržava uniti su:

- Android
- Windows, Windows Phone
- Linux
- Mac, iOS
- Facebook Gameroom, Web GL
- Play Station 4, Play Station Vita, xBox one, Wii U
- Gear VR, Steam VR, Google Cardbard, Oculus Rift

Svako računalo ima različite ulazne parametre. Igra koja zahtjeva puno ulaznih parametara za kontrolu će se najvjerovatnije napraviti za kompjutere jer nam tipkovnica nudi mogučnost da stavimo mnoštvo kontrola. Neke igre su dosta jednostavne i korisniku služe npr. dok čeka red u banci da to vrijeme iskoristi na zabavan način. Takva igra će se vjerovatno nalaziti na mobilnim uređajima. Ukoliko želimo napraviti igru koja će se nalaziti na skoro svim platformama potrebno je napraviti zabavnu ali kompleksnu igru sa što manje ulaznih parametara. U unity-u je to dosta jednostavno jer kada izrađujemo igru podijelimo kontrole za svaku platformu u kojoj želimo objaviti svoju igru. Ovaj rad se može koristiti na android i windows platformi. Nakon što se igra napravila samo je trebalo u postavkama izraditi za ove dvije platforme. U sljedečem ispisu se vidi kako su se u kodu podjelile kontrole koje služe za svaku platformu zasebno.

```
if (Application.platform == RuntimePlatform.WindowsPlayer)
  {
  playerMovePc();
  pausePc();
  }
if (Application.platform == RuntimePlatform.Android)
  {
  playerMoveAcceleration();
  pausePhone();
  }
```

Ispis : Prikaz dijeljenja kontrola za različite platforme

10 Zaključak

Cilj svake igre je da bude korisniku što jednostavnija za upotrebu i naravno da bude zanimljiva. Svaki programer može napraviti igru, ali pitanje je koliko će ta igra privlačiti korisnika. Da bi igra bila pristupačnija korisniku trebala bi biti lijepo dizajnirana jer većina korisnika koja želi igrati neku igru će prvo pogledati dizajn igre. Možemo reći da će dizajn igre privući korisnika a fukncionalnosti igre će pridobiti korisnika. Učenje svake tehnologije je u početku teško ali ukoliko je programer željan stvaranje nečeg novog i ima strast prema igrama unity je pravo rješenje za nega. Na internetu se može naći puno primjera i objašnjenja kako koristiti unity. Tjeko izrade ovog projekta sam se upoznao sa unity okolinom. Trebalo mi je nekoliko vremena da savladam glavne komponente unity-a, a nakon što sam to prošao samo sam trebao upotrebiti vlastitu maštu kako bih igru napravio što pristupačnijom korisniku. Kod izrade igara za mobilne uređaja se javlja problem kontrole nekog objekta. Objekte možemo kontrolirati preko dugmadi, dodira na ekran i preko akcelometra. Ja sam se odlučio za akcelometar zato što time ostavljam korisniku čist pogled na igru, ali sve ovisi o kompleksnosti igre koje ce autor kontrolu odabrati.

Literatura

- https://unity3d.com/learn/tutorials
- https://www.youtube.com/watch?v=n0VspDUOErE&t=452s
- https://unity3d.com/learn/tutorials/topics/scripting