Šolski center Novo mesto
Srednja elektro šola in tehniška gimnazija
Šegova ulica 112
8000 Novo mesto

Izdelava igre - Parkour

(Maturitetna seminarska naloga)

Predmet: Računalništvo

Avtor: Gašper Klepec, T4A
Mentor: dr. Albert Zorko, univ. dipl. inž. el.

Novo mesto, april 2022

Povzetek

Cilj seminarske naloge je spoznati proces izdelave 2D računalniške igre v programu Unity. Ugotovil sem, da moram najprej ustvariti objekt ter ga urediti, da se ob odzval na ostale objekte in na naše ukaze. Vse to sem naredil s pomočjo kode zapisane v programskem jeziku C#. Na koncu sem igro tudi testiral.

Ključne besede

- Unity
- C#
- 2D igra
- Igralec

Kazalo vsebine

1 Uvod 1
2 Teoretični del 2
2.1 Unity 2
2.1.1 Unity kot motor 2
2.1.2 Unity IDE 2
2.1.3 Trgovina sredstev (ang. Asset Store) 3
2.1.4 Orodja za animacijo 3
2.2 C# 4
2.2.1 C# kot programski jezik 4
2.2.2 .NET 4
3 Praktični del 5
3.1 Začetek izdelave 5
3.2 Premikanje igralca 5
3.3 Animacije 8
3.4 Nasprotniki 9
3.5 Kamera 10
3.6 Scene 11
3.7 Kovanec 12
3.8 Zaustavitveni meni 13
3.9 Končne scene 14
4 Zaključek 15
5 Zahvala 16
Bibliografija 17

Kazalo slik

Slika 1: Dodajanje novega objekta 5
Slika 2: Komponente objekta Igralec 5
Slika 3: Koda za premik objekta Igralec 6
Slika 4: Obrat objekta glede na smer 6
Slika 5: Koda za skok 7
Slika 6: Nastavitve hitrosti in moči skoka 7
Slika 7: Animacija objekta igralec 8
Slika 8: Povezava animacij 8
Slika 9: Odstranitev zamika med animacijami 8
Slika 10: Koda za delovanje animacij 8
Slika 11: Koda za nasprotnike 9
Slika 12: Koda za premikanje objekta 9
Slika 13: Zamrznitev premikanja in rotacije 9
Slika 14: Koda za premik kamere 10
Slika 15: Koda za prehod v druge teme 10
Slika 16: Ozadje scene 11
Slika 17: Izbris kovanca ob stiku z igralcem 12
Slika 18: Vrednost kovanca 12
Slika 19: Prištevanje vrednosti kovanca 12
Slika 20: Meni za zaustavitev igre 13
Slika 21: Začetna scena 14
Slika 23: Druga scena 14
Slika 22: Tretja scena 14
Slika 24: Končna scena 14

1 Uvod

Namen seminarske naloge je izdelati igro s pomočjo Unity in jo predstaviti. Moj cilj je bil narediti igro stila »platformer«, ki bo za uporabnika zabavna in ne pretežka. Pri izdelavi sem uporabljal programsko okolje Unity in programski jezik C#, s pomočjo katerih sem izdelal igralca, animacije, mapo, ovire in nasprotnike. Napisal sem skripte, katere omogočajo samo delovanje. V nadaljevanju seminarske naloge pa bom vsako skripto tudi posebej opisal.

2 Teoretični del

2.1 Unity

2.1.1 Unity kot motor

Unity je motor (ang. engine) za 2D in 3D igre, ki obstaja že od leta 2005. Razvilo ga je podjetje Unity Technologies, z namenom, da bi več razvijalcem zagotovili dostop do orodja za razvoj iger. To je bil v tistih časih nov podvig. Od ustanovitve se je motor dramatično spremenil in razširil ter uspel slediti najnovejšim praksam in tehnologijam. (1) (2)

Še danes je glavni poudarek zagotoviti čim bolj robusten nabor orodij za industrijo razvoja iger, ne glede na raven znanja uporabnika. Svoj doseg so razširili tudi na druge industrije z velikim poudarkom na razvoju 3D v realnem času, zaradi česar je eden najmočnejših motorjev, ki so trenutno na voljo. (3) (2)

Kot okolje Unity zagotavlja številne najpomembnejše vgrajene funkcije, zaradi katerih igra deluje. To so stvari kot so; fizika, 3D upodabljanje in zaznavanje trkov, izračunavanjem vsakega zadnjega gibanja vsakega materiala ali načina, kako naj se svetloba odbija od različnih površin. (4)

2.1.2 Unity IDE

Poleg motorja za igre je Unity tudi integrirano razvojno okolje (ang. Integrated Development Environment). Opisuje vmesnik, ki nam omogoča dostop do vseh orodij, ki jih potrebujemo za razvoj ne enem mestu. Programska oprema Unity ima vizualni urejevalnik, ki ustvarjalcem omogoča, da elemente preprosto povlečejo in spustijo v prizore in nato manipulirajo z njihovimi lastnostmi. (1) (5)

2.1.3 Trgovina sredstev (ang. Asset Store)

Ne glede na to, ali potrebujemo grafična sredstva, posebne predloge zvrsti iger, zvok, učinke delcev ali kaj drugega, nam Unity to omogoča. Vsebuje namreč veliko trgovino sredstev, opremljeno z raznimi plačljivimi ali brezplačnimi sredstvi, ki jih lahko uporabimo za kateri koli projekt.. (4) (6)

2.1.4 Orodja za animacijo

Unity ponuja robusten nabor orodij za animacijo, ki deluje tako za 3D kot 2D grafiko. Medtem ko lahko uvozimo animacije iz drugih programov, kot je npr. Blende, nam Unity ponuja možnost animacije projektov kar v motorju samem. To vključuje prilagajanje položaja in vrtenja celotnega predmeta, da dejansko fizično manipulira s objekti v 3D modelu. Unity nam celo ponuja možnost dodajanja kosti 2D slikam. (6)

Do vseh teh funkcij lahko seveda dostopate tudi iz »Scripting API-ja«, kar vam omogoča neprimerljiv nadzor nad delovanjem animacij. (7)

Poleg tega animator omogoča preprosto ustvarjanje stoja stanja animacije. To pomeni, da ne moramo predvajati samo animaciji, ki temeljijo na tem, kaj igralec počne, temveč prehajamo gladko in ustrezno med vsako animacijo. Animator je predstavljen v bolj vizualnem slogu grafov, zato je enostavno razumeti, kako se vse povezuje. (7)

2.2 C#

2.2.1 C# kot programski jezik

C# je programski jezik, ki ga je razvil Microsoft leta 2001. Namen je bil razviti programski jezik, ki se ga ni le enostavno naučiti, ampak podpira tudi sodobno funkcionalnost za vse vrste razvoja programske opreme. Je preprost, sodoben in objektno usmerjen jezik, ki je bil zasnovan tako, da upošteva potrebe podjetij in podjetnikov. Ponuja funkcionalnost za podporo sodobnemu razvoju programske opreme ter potrebe po razvoju spletnih in mobilnih aplikacij. (8) (9)

2.2.2 .NET

Programi C# se izvajajo na .NET, navideznem izvajalnem sistemu, imenovanem skupni jezik izvajanja (ang. Common Language Runtime - CLR) in nizu knjižnic. CLR je Microsoftova implementacija skupne jezikovne infrastrukture (ang. Common Language Infrastructire - CLI), mednarodnega standarda. CLI je osnova za ustvarjanje izvedbenih in razvojnih okolij, v katerih jeziki in knjižnice nemoteno delujejo skupaj. (9)

Izvorna koda, napisana v C# je prevedena v vmesni jezik (ang. Intermediate Language) (IL), ki je skladen s specifikacijo CLI. Koda IL in viri, kot so bitne slike in nizi so shranjeni v sklopu in se običajno končajo s končnico .dll. Sestav vsebuje manifest, ki zagotavlja informacije o vrstah, različic in kultur sklopa. Ko se program C# izvede, se sklop naloži v CLR, kateri izvede kompilacijo ravno-v-času (ang. Just-In-Time) (JIT), da pretvori kodo IL v izvorna strojna navodila. (10)

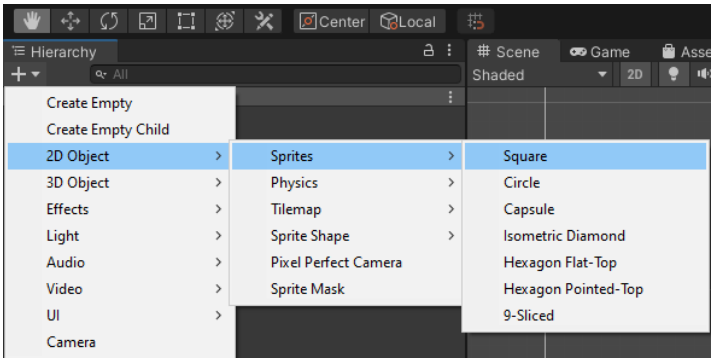
Razvojna interoperabilnost je ključna značilnost .NET. Koda IL, ki jo izdela prevajalnik C#, je skladna s specifikacijo običajnega tipa (ang. Common Type Specification - CTS). Koda IL, ustvarjena iz C#, lahko komunicira s kodo, ki je bila ustvarjena iz .NET različic. Obstaja več kot 20 drugih jezikov, ki so skladni s CTS.(11)

Poleg izvajalnih storitev .NET vključuje tudi obsežne knjižnice. Te knjižnice podpirajo veliko različnih delovnih obremenitev. Organizirani so v imenske prostore, ki zagotavljajo široko paleto uporabnih funkcij.(12)

3 Praktični del

3.1 Začetek izdelave

Izdelavo igre sem začel tako, da sem ustvaril nov projekt v programu Unity. Ustvaril sem nov 2D objekt kot je razvidno na sliki in ga poimenoval Igralec (ang. Player).

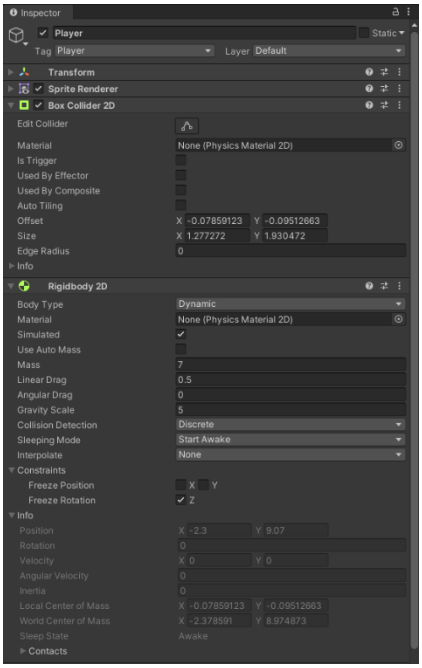


Slika 1: Dodajanje novega objekta

3.2 Premikanje igralca

Da se bo igralec lahko premikal sem mu dodal komponento imenovano togo telo 2D (ang. RigidBody 2D), kjer sem objektu nastavil maso in vpliv gravitacije nanj. Da se bo pa lahko odzval na ostale objekte, ki sem jih kasneje dodal, sem uporabil še eno komponento poimenovano škatlasti trkalnik 2D (ang. BoxCollider 2D). Nato sem objektu dodal še oznako (ang. Tag) »Player«, ki jo bom pozneje uporabljal v skriptah kode za delovanje igre.

Na sliki 2 je razvidno, kako je izgledalo po tem, ko sem dodal vse komponente v objekt Igralec.



Slika 2: Komponente objekta Igralec

Da objekt ne bo padal v neskončnost sem dodal nov objekt katerega sem pomenoval tla (ang. Ground) in mu prav tako dodal škatlasti trkalnik in oznako Ground, da se bo igralec lahko nanj odzval.

Nato sem ustvaril novo skripto jo poimenoval “PlayerMovement” in vanjo napisal kodo za premik objekta Igralec. Ker je igra narejena 2D za premikanje potrebujemo samo vodoravne premike. Slika 3 prikazuje kodo ki poskrbi da se objekt premika levo in desno glede na pritisk tipk na tipkovnici.

```
float horizontalInput = Input.GetAxis("Horizontal");
body.velocity = new Vector2(horizontalInput * speed, body.velocity.y);
```

Slika 3: Koda za premik objekta Igralec

Dodal sem še kodo kjer se objekt obrne v smeri premikanja. To je narejeno tako, da preveri če je vrednost premikanja v vodoravni smeri pozitivna (desna), če je mu dodeli že obstoječi vektor, kateri je obrnjen v desno, če pa je smer premikanja negativna (leva) mu določi nov vektor in objekt obrne za 180 stopinj kot je razvidno na sliki 4.

```
if (horizontalInput > 0.01f)
    transform.localScale = Vector3.one;
else if (horizontalInput < -0.0f)
    transform.localScale = new Vector3(-1, 1, 1);
```

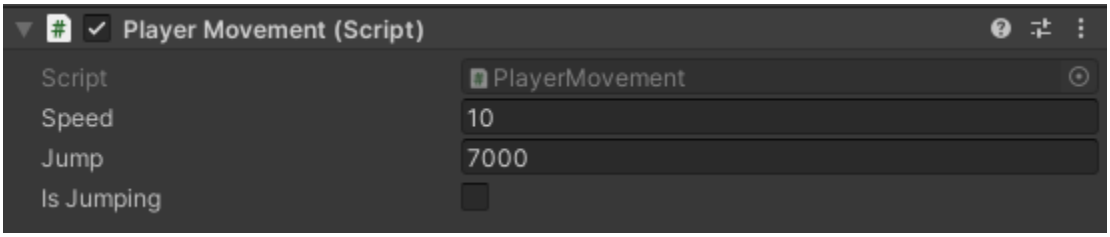
Slika 4: Obrat objekta glede na smer

V isto skripto v katero sem napisal kodo za premikanje in obrat, sem dodal še kodo za skok. Dodamo nov vektor da ohranimo vodoravno gibanje ter objekt premaknemo navzgor. Dodal sem še, da objekt ne mora skočiti če ni v stiku z tlemi. Prej sem tla označil z oznako Ground katero sem uporabil v kodi, kjer program zazna če je objekt v stiku z tlemi, če je mu omogoči skok, če ni to program zazna in mu omogoči ponovni skok šele takrat ko je ponovno v stiku z tlemi

```
    if (Input.GetButtonDown("Jump") && isJumping == false)
    {
        body.AddForce(new Vector2(body.velocity.x, jump));
    }
private void OnCollisionEnter2D(Collision2D other)
{
    if(other.gameObject.CompareTag("Ground"))
    {
        isJumping = false;
    }
}
private void OnCollisionExit2D(Collision2D other)
{
    if(other.gameObject.CompareTag("Ground"))
    {
        isJumping = true;
    }
}
```

Slika 5: Koda za skok

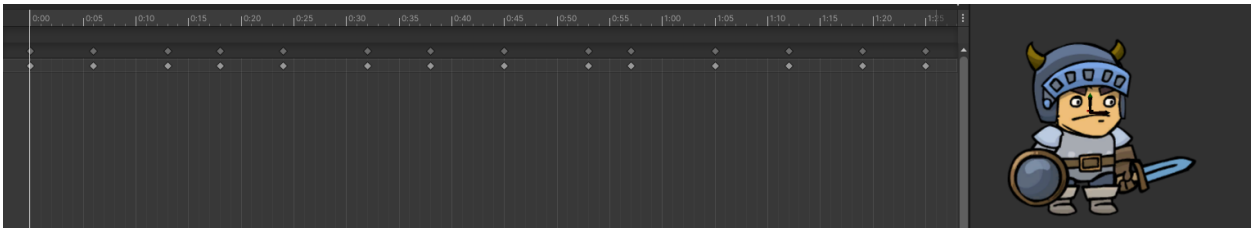
Ko sem končal celotno kodo za premikanje sem jo dodelil objektu Igralec in nastavil hitrost in moč skoka (glej slika 6).



Slika 6: Nastavitve hitrosti in moči skoka

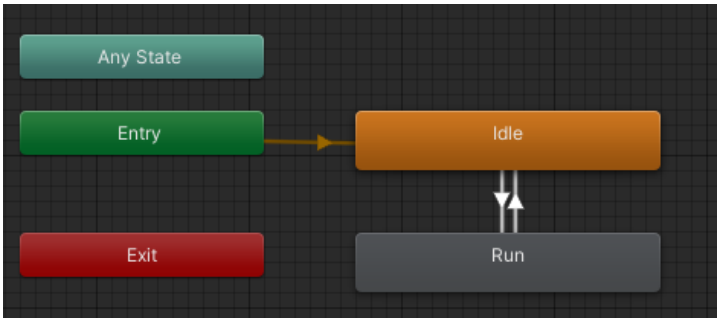
3.3 Animacije

Iz trgovine storitev (ang. Asset Store) ki ga ponuja Unity sem naložil slike s katerimi sem spremenil izgled objekta Igralec. Sliko sem dodal objektu ter naredil animacije, ko objekt miruje ter ko se premika levo ali desno. To sem naredil tako, da sem slike vsake animacije posebaj dodal na nekaj stotink razmika v že vgrajen animator (slika 7).



Slika 7: Animacija objekta igralec

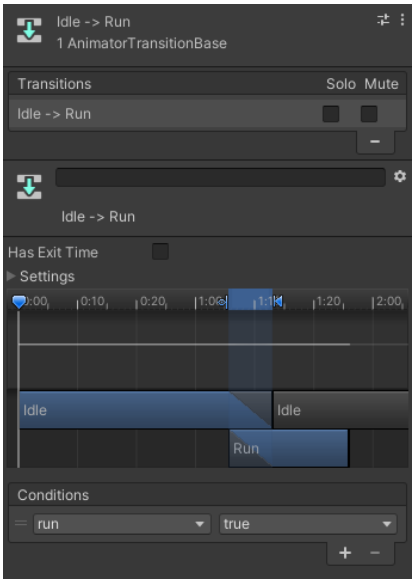
Obe animaciji sem povezal skupaj, tako da se animacija za mirujoč objekt (ang. idle) ustavi ko se ta začne premikati (ang. run) in spet začne delovati ko se objekt ustavi. Ker sem hotel da se to zgodi brez kakšnih zamikov sem odkljukal opcijo »has exit time«. V skripto PlayerMovement sem moral dodati še del kode, da je omogočila delovanje animacije ko se objekt premika. Vse to je razvidno na spodnjih slikah.



Slika 8: Povezava animacij

```
anim.SetBool("run", horizontalInput != 0);
```

Slika 10: Koda za delovanje animacij



Slika 9: Odstranitev zamika med animacijami

3.4 Nasprotniki

V igro sem dodal še nasprotnike, mirujoče špice (ang. spikes) in premikajočo se žago (ang. saw) za vsako od niju sem ustvaril nov objekt jima dodal robni trkalnik 2D (EdgeCollider 2D), katerega sem uporabil, da se lahko bolje približam obliki objektov saj nobeden ni štirikotnik. Objektu žaga sem dodal tudi RigidBody 2D saj bom pozneje dodal kodo da se bo lahko premikala. Ker sta objekta nasprotnika se jih v igri ne želimo dotakniti, zato sem naredil novo skripto z kodo, da če se jih dotaknemo nas pošlje nazaj na začetek ali na kontrolno točko (ang. checkpoint). Vsakemu objektu sem nato dodal skripto. Koda za nasprotnike je vidna na sliki 11.

```
public GameObject startPoint;
public GameObject Player;
private void OnCollisionEnter2D(Collision2D other)
{
    if (other.gameObject.CompareTag("Player"))
    {
        Player.transform.position = startPoint.transform.position;
    }
}
```

Slika 11: Koda za nasprotnike

Da se bo žaga lahko premikala sem moral narediti novo skripto in napisati kodo, naj se objekt premika v eno smer dokler ne zadane stene in se nato začne premikati v drugo smer dokler ponovno ne zadane stene nato ponovi postopek (slika 12). Ker bodo to posebne stene ki v igri ne bodo vidne sem zanje naredil novo prazno skripto kateri sem samo spremenil ime v »Wall«, da bom lahko z njo označil stene od katerih naj se žaga odbije. Žagi sem še zamrznil možnost rotacije in premikanja navzgor ali navzdol (slika 13) saj sem hotel, da se premika samo levo in desno. Na koncu sem vsem objektom dodal ustrezne skripte.

```
private float dirX;
private float moveSpeed;
private Rigidbody2D rb;
private bool factingRight = false;
private Vector3 localScale;

void Start()
{
    localScale = transform.localScale;
    rb = GetComponent<Rigidbody2D>();
    dirX = -1f;
    moveSpeed = 5f;

}

private void OnTriggerEnter2D(Collider2D collision)

if (collision.GetComponent<Wall>())
{
    dirX *= -1f;
}
```

Slika 12: Koda za premikanje objekta

Slika 13: Zamrznitev premikanja in rotacije

3.5 Kamera

V Unity že podana kamera zato je nisem rabil narediti na novo, napisal sem samo kodo s katero kamera sledi igralcu ki se premika po mapi (slika 14). Naredil sem, da je kamera rahlo dvignjena za boljše in lepše igranje končne igre.

```
transform.position = new Vector3(target.transform.position.x, target.transform.position.y + 1 , -10);
```

Slika 14: Koda za premik kamere

Ker sem v igri nameraval naredil več kot eno sceno (ang. Scene) sem na koncu prvega postavil objekt v obliki zastave. Ko se jo z Igralcem dotaknejo nas ta pošlje v naslednjo. V kodo sem moral dodati del »using UnityEngine.SceneManagement« saj brez tega prehod v druge scene ni mogoč.

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

public class FinishLine : MonoBehaviour
{

    private void OnCollisionEnter2D(Collision2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            SceneManager.LoadScene("level2");
        }
    }
}
```

Slika 15: Koda za prehod v druge teme

3.6 Scene

Za tem sem se lotil izdelave scen, izdelal sem dve. Prva je krajša in lažja za končati, druga pa je daljša in nekoliko bolj zahtevna. V obe sem dodal tudi nekaj premikajočih se platform za katere sem uporabil isto kodo za premikanje vodoravno kot pri premikajočih se nasprotnikih. Vsaki sceni sem dodal tudi nekaj vizualnih dodatkov.

Nato sem vsaki sceni dodal ozadje katero je sestavljeno iz sedmih delov. Vsakemu sem določil svoj vrstni red v postavitvi (ang. Order In Layout). Del ki je najbolj oddaljen je dobil najnižjo številko. Zaradi tega sem naredil celotno ozadje tipa paralaksa. Vsakemu delu sem uredil kako hitro se bo premikal v primerjavi z igralcem. Kar je bilo mogoče s kodo ki je prikazana na sliki 16.

```
private float lenght, startpos;
public GameObject cam;
public float parallaxEffect;
void Start()
{
   startpos = transform.position.x;
   lenght = GetComponent<SpriteRenderer>().bounds.size.x;
}
void Update()
{
    float dist = (cam.transform.position.x * parallaxEffect);

    transform.position = new Vector3(startpos + dist, transform.position.y , transform.position.z);
}
```

Slika 16: Ozadje scene

3.7 Kovanec

V vsako sceno sem dodal tudi kovance katere lahko igralec pobere ter se izpišejo levo zgoraj. To dosežemo tako da vsakemu kovancu določimo vrednost (slika 17). Ko igralec pride v stik z kovancem se ta iz igre izbriše in njegova vrednost se prišteje levo zgoraj (slika 18). V skripto »PlayerMovement« sem moral dodati kodo s katero sem omogočil izbris kovanca iz igre (slika 19). Vsakemu kovancu sem dodal krožni trkalnik 2D (CircleCollider 2D) saj se lepo prilega obliki kovanca. Kovancem sem dodal tudi animacijo katero sem izdelal isto kot animacijo za premikanje pri Igralcu.

```
public int coinValue = 1;

private void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.CompareTag("Player"))
    {
        ScoreManager.instance.ChangeScore(coinValue);
    }
}
```

Slika 18: Vrednost kovanca

```
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.CompareTag("Coins"))
    {
        Destroy(other.gameObject);
    }
}
```

Slika 17: Izbris kovanca ob stiku z igralcem

```
using TMPro;

public class ScoreManager : MonoBehaviour
{
    public static ScoreManager instance;
    public TextMeshProUGUI text;
    int score;
    void Start()
    {
        if(instance == null)
        {
            instance = this;
        }
    }
    public void ChangeScore(int coinValue)
    {
        score += coinValue;
        text.text= "" + score.ToString();
    }
}
```

Slika 19: Prištevanje vrednosti kovanca

3.8 Zaustavitveni meni

Izdelal sem se okno ki se pojavi ko pritisnemo tipko escape. Ko se pojavi nam igro ustavi nato se pojavita dva gumba eden za izhod iz igre in eden za nadaljevanje. Ko se igra ustavi postane viden tudi kazalec na miški kateri spet izgine ko nadaljujemo z igro (slika 20).

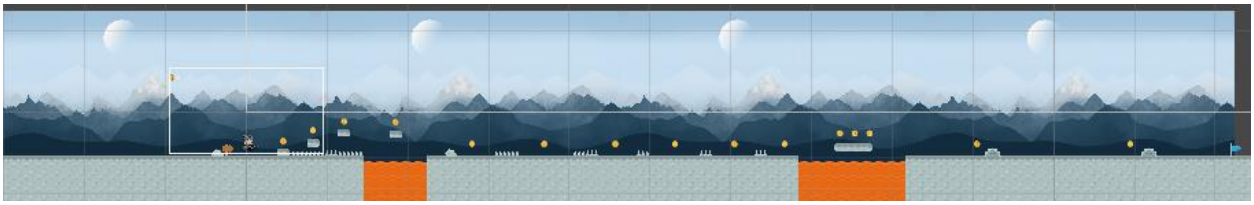
```
public GameObject pauseMenu;
void Start()
{
    Cursor.visible = false;
}

void Update()
{
    if(Input.GetKeyDown(KeyCode.Escape))
    {
        if(!pauseMenu.activeSelf)
        {
            Time.timeScale = 0f;
            pauseMenu.SetActive(true);
            Cursor.visible = true;
        }
        else
        {
            Time.timeScale = 1f;
            pauseMenu.SetActive(false);
            Cursor.visible = false;
        }
    }
}
public void quit()
{
    Application.Quit();
}
public void resume()
{
    Time.timeScale = 1f;
    pauseMenu.SetActive(false);
    Cursor.visible = false;
}
```

Slika 20: Meni za zaustavitev igre

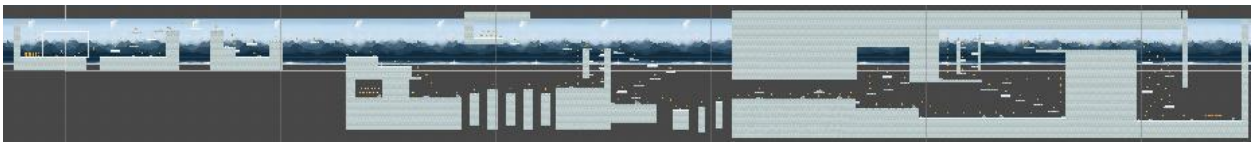
3.9 Končne scene

Na koncu sem izdelal še končno in začetno sceno; obe vsebujeta tekst in gumb. Na začetni sceni je tekst z imenom igre ter gumb za začetek, na končni pa je tekst za konec igre ter gumb za vrnitev na začetno sceno. Za delovanje gumbov sem uporabil isto kodo kot pri zastavi za prehod iz ene scene v drugo. Spodnje slike prikazujejo vse končne scene igre.

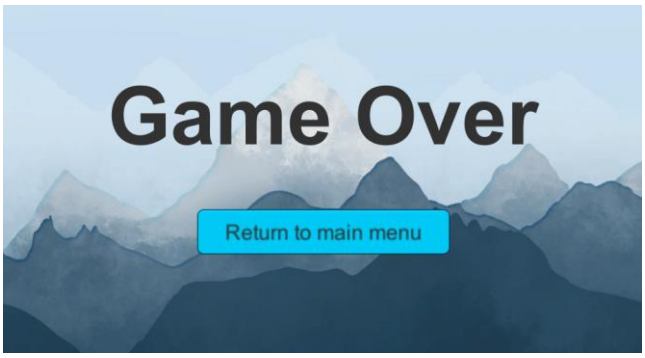
Slika 21: Začetna scena



Slika 23: Druga scena



Slika 22: Tretja scena



Slika 24: Končna scena

4 Zaključek

V seminarski nalogi sem predstavil izdelavo 2D računalniške igre v programu Unity. Potek izdelave sem predstavil kar se da podrobno. Podal sem vso kodo uporabljeno v igri. Koda je napisana v programskem jeziku C#.

Igro bi se seveda dalo še izboljšati, lahko dodamo še več scen ter nasprotnikov za še bolj zanimivo izkušnjo.

Med izdelavo seminarske naloge sem bolj podrobno spoznal program Unity ter samo izdelavo 2D iger.

5 Zahvala

Rad bi se zahvalil profesorjema in mentorjema Albertu Zorku ter Gregorju Medetu, ki sta me čez vsa 4 leta učila, me spodbujala k doseganju mojih ciljev in mi pomagala kadarkoli je bilo to mogoče.

Rad bi se zahvalil tudi mojim sošolcem, ki so mi vsa leta stali ob strani in so mi bili vedno pripravljeni pomagati.

Na koncu bi se rad zahvalil moji družini, ki me je skozi leta mnogo naučili, me vedno spodbujala, mi dajala neprecenljive nasvete ter mi priskočila na pomoč kadarkoli so se pojavili težave.

Bibliografija

1. Sinicki, Adam. AndroidAuthority. *What is Unity? Everything you need to know.* [Elektronski] 20. 4 2021. https://www.androidauthority.com/what-is-unity-1131558/.

2. Schardon, Lindsay. GameDev Academy. *What is Unity? – A Guide for One of the Top Game Engines.* [Elektronski] 22. 2 2022. https://gamedevacademy.org/what-is-unity/.

3. Microsoft . *What is Unity?* [Elektronski] https://dotnet.microsoft.com/en-us/apps/games/unity.

4. Petty, Josh. ConceptArtEmpire. *What is Unity 3D & What is it Used For?* [Elektronski] https://conceptartempire.com/what-is-unity/.

5. Zamojc, Ian. tutsplus. *Introduction to Unity3D.* [Elektronski] 12. 5 2012. https://code.tutsplus.com/tutorials/introduction-to-unity3d--mobile-10752.

6. study tonight. *Introduction to Unity 3D.* [Elektronski] https://www.studytonight.com/3d-game-engineering-with-unity/introduction-to-unity.

7. Arsalan. Information transformation services. *What is Unity 3D.* [Elektronski] https://it-s.com/what-is-unity-3d/.

8. Chand, Mashes. C# Corner. *What is C#.* [Elektronski] 7. 4 2020. https://www.c-sharpcorner.com/article/what-is-c-sharp/.

9. GeeksforGeeks. *Introduction to C#.* [Elektronski] 17. 12 2019. https://www.geeksforgeeks.org/introduction-to-c-sharp/.

10. Microsoft. *A tour of the C# language.* [Elektronski] 18. 3 2022. https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

11. w3schools. *What is C#?* [Elektronski] https://www.w3schools.com/cs/cs_intro.php.

12. JavaPoint. *C# Tutorial.* [Elektronski] https://www.javatpoint.com/c-sharp-tutorial.

Priloge

Github: https://github.com/KlepecG/Ga-per-Klepec.git

Stvarno kazalo

A

animacije · 1

B

BoxCollider · 5

C

C# · 3

G

gumb · 13

I

Igralec · 5

K

kode · 3

N

nasprotnika · 8

O

objekt · 3

P

PlayerMovement · 11
premikanje · 11
programski jezik · 4

R

računalniške igre · 3
RigidBody · 5

S

scene · 9
skripto · 6
slike · 7
špice · 8

T

Tag · 5
tekst · 13
Trgovina sredstev · 3

U

Unity · 3

Z

žaga · 9