Izrada klasične 2D igre poput "Super Mario Bros"

Cvrtila, Domagoj

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

DOMAGOJ CVRTILA

IZRADA KLASIČNE 2D IGRE POPUT “SUPER MARIO BROS”

ZAVRŠNI RAD

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

DOMAGOJ CVRTILA

IZRADA KLASIČNE 2D IGRE POPUT "SUPER MARIO BROS"

MAKING A CLASSIC 2D GAME SIMILAR TO "SUPER MARIO BROS"

ZAVRŠNI RAD

Mentor:

Nenad Breslauer, v.pred.

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

ODBOR ZA ZAVRŠNI RAD

Čakovec, 17. veljače 2021.

Država: **Republika Hrvatska**

Predmet: **Razvoj računalnih igara**

ZAVRŠNI ZADATAK br. 2020-RAČ-R11

Pristupnik: **Domagoj Cvrtila (0269107551)**

Studij: redovni preddiplomski stručni studij Računarstvo

Smjer: Programsko inženjerstvo

Zadatak: **Izrada klasične 2D igre poput „Super Mario Bros“**

Opis zadatka:

Cilj rada je izrada 2D platformer-a gdje igrač kontrolira lik koji trči i skače po platformama, sakuplja novčiće sa ciljem da dođe do kraja staze. Potrebno je izraditi scenu te sve potrebne elemente za igru, koristiti mogućnosti koje pruža podsustav za osvjetljenje scene, koristiti simulaciju fizikalnih svojstava, kontrolu kamere, game meni i zvučne efekte. Koristiti platformu Unity, programski jezik C# te dodatne programske alate.

Rok za predaju rada: 20. rujna 2022.

Mentor: Predsjednik povjerenstva za završni ispit:

_____ _____

Nenad Breslauer, v. pred.

ZAHVALA

Zahvaljujem svim profesorima i prijateljima, koje sam stekao, što su učinili zadnje 3 godine nezaboravnim i poučnim iskustvom. Također, želim se posebno zahvaliti mom mentoru koji je bio vrlo susretljiv i spreman pomoći, ne samo kod izrade završnog rada već i tokom cijelog studiranja.

Domagoj Cvrtila

1. SAŽETAK

Tema ovog završnog rada je izrada klasičnog 2D platformera poput „Super Mario Bros“ koristeći platformu Unity 2D. Osnovni cilj izrade je 2D platformer – vrsta računalne igre gdje se igračev lik kreće u 2-dimenzionalnom prostoru (visina i duljina) i njegov zadatak je doći do kraja staze odnosno cilja. Igrača na putu do cilja sprječavaju brojne nedaće poput neprijatelja, šiljaka, rupa, pomicajuće platforme i slično. Igra se pomoću tipkovnice, a, osim strelica za kretanje, glavni alat u platformeru je „spacebar“ odnosno skok – neizostavni i najprepoznatljiviji dio svakog platformera. Dizajn odnosno radnja videoigre smještena je na zamišljenim stazama, odnosno levelima, koje se međusobno razlikuju dizajnom razine, grafikom, zamkama i slično. U nastavku završnog rada naizmjenice će biti korištene riječi level, razina, staza ili nivo – sve one označavaju istu stvar. Scena je popunjena interaktivnim 2D-modelima: pomicajuće platforme, šiljci, cirkulari kao i novčići odnosno kiviji koje igrač mora sakupljati. Svi spomenuti elementi, izuzev šiljaka, će biti animirani. Igrač može vidjeti broj sakupljenih kivija u gornjem lijevom kutu. Kretanjem do cilja igračev lik mora izbjegavati šiljke i cirkulare te pravovremeno skočiti na pomicajuće platforme kako bi sigurno došao do cilja, te putem sakupljati kivije za bodove. Ako putem igrač, primjerice, naleti na cirkular on umire, pokreće se animacija umiranja, uz pripadajući zvuk, te se vraća na početak razine, odnosno levela. Svi igračevi sakupljeni bodovi, odnosno kiviji, su izbrisani u slučaju smrti te se razina mora ponoviti kako bi igrač uspješno došao do sljedećeg levela. Dolaskom na cilj igrač je prešao tu razinu i zatim odlazi na drugu ili natrag na glavni izbornik. Za izradu igre koristit će se programski jezik C# i Unity 2D programsko sučelje. Videoigra sadrži sve osnovne elemente koje definiraju 2D platformer, a to su: prelazak staze slijeva na desno, odnosno kretanje od točke A do točke B, a glavni elementi kretanja su skok i trčanje.

Ključne riječi: *Platformer, skok, Unity 2D, 2D modeli, C#*

Sadržaj

2. UVOD

Za uspješnu videoigru potrebno je imati mnoge vještine i znanja jer je izrada računalne igre vrlo kompleksna i zahtjevna zbog same činjenice što sve videoigre obuhvaćaju. Videoigre možemo igrati na raznim platformama: mobiteli, računala ili konzole. Osim što je za izradu videoigre potrebna dobra ideja, kao i sam dizajn, također su neophodna znanja iz programiranja, kao i ovladavanje raznim elementima koji ulaze u izradu videoigre: zvukovi, animacije, modeli, logika. Za izradu jednostavnijih igara nije potrebno više od jedne osobe i tu dolazi pouzdani Unity sa svojim višestrukim resursima i opcijama. Što se kompleksnijih videoigara tiče, njih rade veliki studiji koji imaju timove ljudi koji se bave pojedinim dijelovima same izrade poput dizajniranja levela.

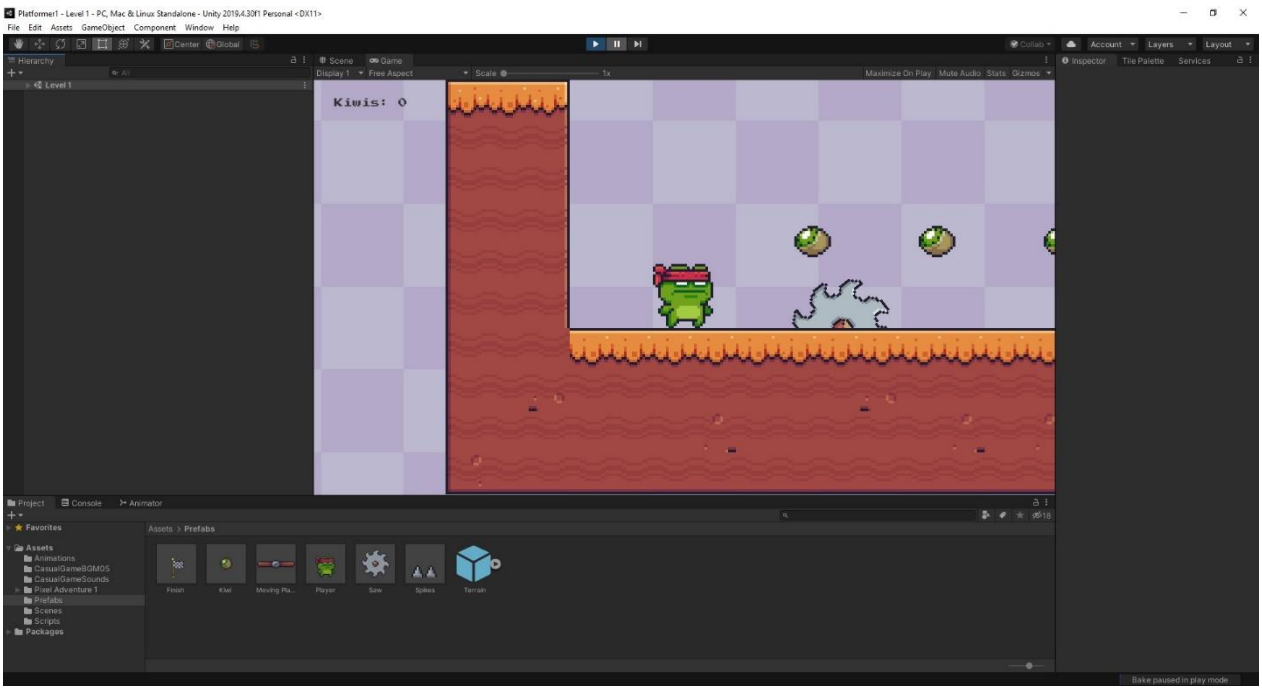
Postoje mnogi žanrovi videoigara, a među najpopularnijima su definitivno: FPS, MMO, RPG, strateške igre, trkaće igre, akcijske igre i sportske igre. Mnogi žanrovi, također, imaju svoje podžanrove, primjerice platformeri se dijele na 2D i 3D platformere. Super Mario Bros je 2D platformer, dok je Super Mario 3D World igra u 3 dimenzije. Jedan od najprepoznatijih likova iz videoigara je gore spomenuti Super Mario – glavna zvijezda japanske tvrtke Nintendo. Ostali poznati platformeri su: Spelunky, Shovel Knight, Sonic Mania, Donkey Kong i mnogi drugi. [1]

3. KORIŠTENI ALATI I TEHNOLOGIJE

Programski alat Unity će biti korišten u izradi videoigre, a jezik C# za programski kod. Materijali (engl. *Assets*) korišteni za izradu su preuzeti iz Unity Asset Store portala.

3.1 Unity Game Engine

Sada već davne 2005. godine pokrenut je Unity game engine. Tvrtka Unity Technologies svojom platformom omogućuje stvaranje 2D ili 3D igara, kao i izradu VR igre. Unity je dostupan na najpopularnijim operacijskim sustavima: Linux, macOS i Windows. Platforma Unity nam omogućava izradu igre "od nule" što znači da kreator upravlja svim aspektima izrade: modeliranje i izrada scene, level dizajn, dodavanje zvukova i glazbe, animacija, programiranje kretnje glavnog lika, kontrolu kamera i raznih drugih značajki (engl. *Feature*). Unity koristi tzv. *Assets* odnosno resurse koji nam služe za izradu igre, a oni mogu biti bilo što od 2D modela, 3D modela, animacija, zvukova, tekstura i slično. *Assets*-i su preuzeti iz Unity Asset Storea koji nudi besplatne, ali i *Asset*-e koji se plaćaju. [2]



Slika 1. Izgled Unity sučelja

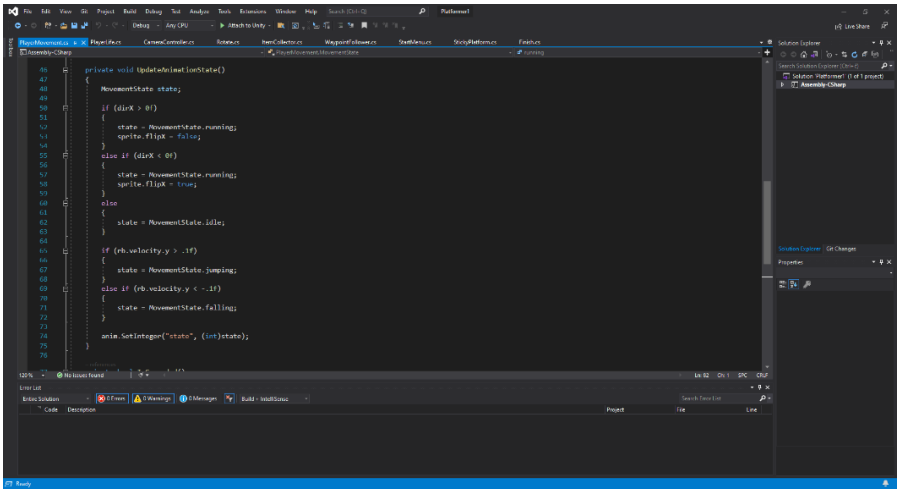
Izvor: Autor

3.2 C# programski jezik

C# (C Sharp) nastao je 2000. godine u američkoj tvrtki Microsoft, a njegovi kreatori su Anders Hejlsberg i Scott Wiltamuth. C# nastao je radi potrebe za jednostavnim, modernim i praktičnim objektno orijentiranim jezikom za .NET platformu. Za korištenje novog sučelja za programiranje aplikacija (API) koristimo .NET platformu kao razvojnu platformu. C# koristi tehnike polimorfizma, nasljeđivanja i enkapsulacije, a *Garbage collection* stavka služi za brisanje objekata koji programu više ne trebaju pa samim time oslobađa memoriju. [3]

3.3 Microsoft Visual Studio

Microsoft Visual Studio je integrirano razvojno okruženje napravljeno od strane Microsofta. Visual Studio služi za razvoj programa koji su namijenjeni operacijskom sustavu Windows, za izradu aplikacija, web-stranica i usluga. Također, podržava gotovo sve programske jezike pod uvjetom da su usluge za isti dostupne. Neki od programskih jezika dostupni uz Visual Studio: C, C#, C++, F# i VB.NET, dok je za ostale programske jezike potrebna dodatna instalacija: Phyton, XML, HTML, JavaScript, CSS i Ruby. Uređivač koda unutar Visual Studia zvan IntelliSense (predlaže kod), sadrži program za otkrivanje i otklanjanje grešaka (engl. *Debugger*). Operacijski sustavi koji podržavaju Visual Studio su MacOS i Windows. Microsoft Azure i GitHub su integrirani servisi za spremanje i organizaciju projekata.

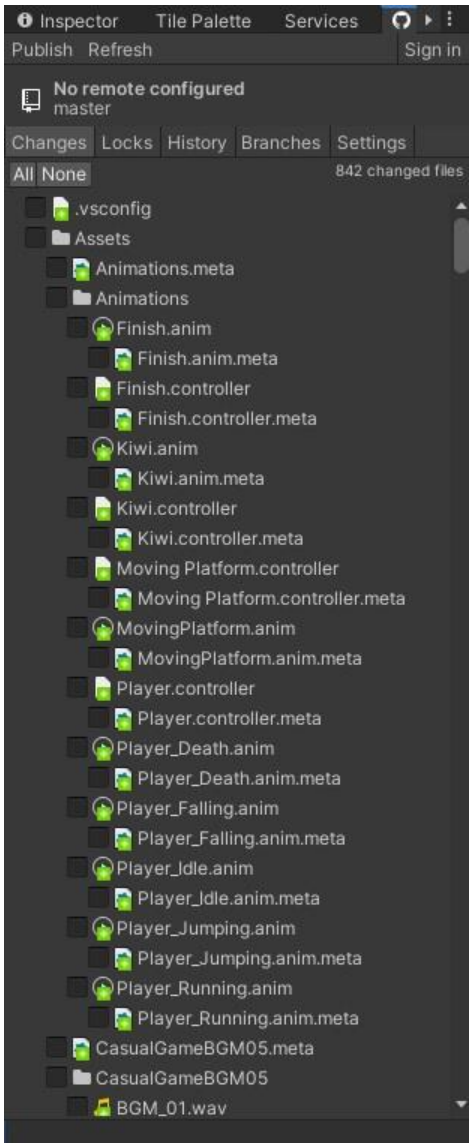


Slika 2. Izgled Visual Studio sučelja

Izvor: Autor

3.4 GitHub

GitHub je oblak (engl. *Cloud*) odnosno internetska servis platforma koja nam omogućuje spremanje i organizaciju projekata. Oni po prirodi mogu biti privatni (vidljivi samo korisniku) ili pak javni (vide ih svi). Na projektu može raditi i više osoba i svaka od njih ima svoj vlastiti ključ koji sprema korisnikove promjene, a nakon što je sigurno da su promjene zadovoljavajuće one se stavljaju na glavni ključ. Proces povezivanja GitHub-a na Unity je vrlo jednostavan pošto se radi o Plugin-u koji se skida preko Unity Asset Store-a. Nakon instalacije pojavljuje se GitHub prozor preko kojega se lako spremaju promjene u projektu.



Slika 3. Izgled GitHub prozora unutar Unity-ja

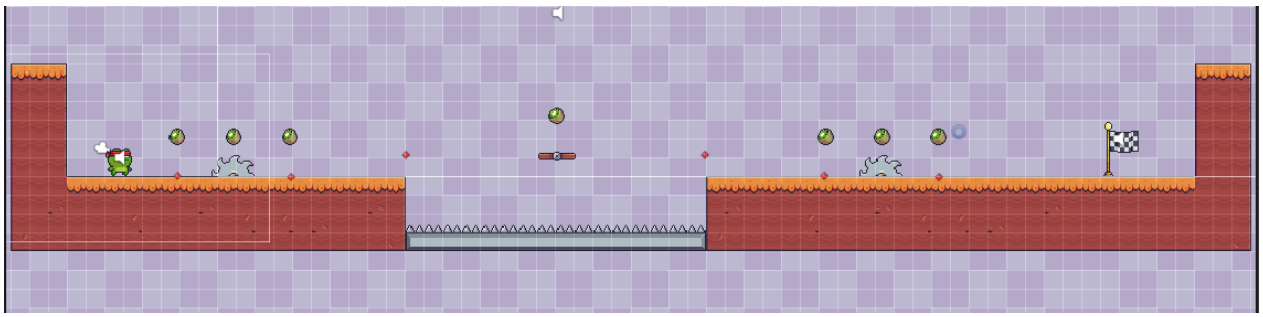
Izvor: Autor

4. MATERIJALI I METODE ZA RAZVOJ IGRE

Za izradu igre korišten je Microsoft Visual Studio i Unity. Postupak izrade, odnosno pojedinačni koraci, bit će objašnjeni u dolje navedenom tekstu.

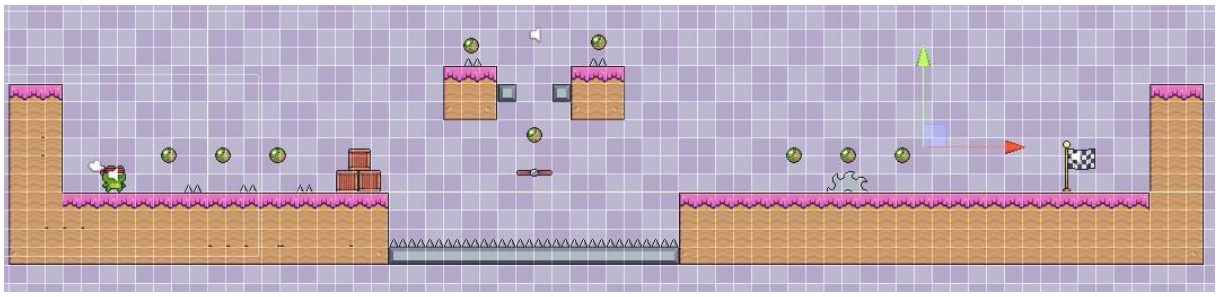
4.1. Izrada scene

Prvi korak svakog Unity projekta jest izrada scene na kojoj se odvijaju igračeve radnje. Scena se sastoji od 2D terena na kojem se nalaze razni *assets-i* odnosno resursi poput modela. U scenu spadaju dvije stijene između kojih se nalazi provalija sa zamkama, a iznad nje leteća platforma. Na sceni se, također, nalazi 7 kivija kao i 2 cirkulara koje je potrebno izbjeći. Lijevi rub scene je napravljen dovoljno visoko da igračev lik ne može preskočiti, a desni je gotovo identičan uz iznimku da ima i zastavicu cilja koja označava kraj staze odnosno levela. Na sljedećoj sceni (slika 5.) nalazi se 1 cirkular, mnogo više šiljaka kao i 2 dodatna kivija, sve zajedno 9, za čije je sakupljanje potrebno pravovremeno skočiti sa platforme kako bi se mogli pokupiti.



Slika 4. Izgled scene prve razine

Izvor: Autor



Slika 5. Izgled scene druge razine

Izvor: Autor

4.2. Modeli

Razni 2D modeli prisutni su unutar scene radi postizanja jedinstvenog vizualnog stila. Primjerice, u drugoj razini prisutne su 3 kutije koje služe samo za estetiku.

Slika 6. Izgled kutija koje se nalaze u pozadini

Izvor: Autor

4.3. Kretanje igrača

Osnovne funkcionalnosti kretanja igrača su definirane u skripti, a pomoću skripte igrač se kreće u sceni sa tipkama A za kretanje u lijevo, D za kretanje u desno i SPACEBAR za skakanje. Na igračev lik su također povezane dodatne komponente poput: *Animator* koja upravlja

animacijama, a one su sljedeće: animacija trčanja, animacija stajanja, animacija skakanja i animacija padanja, *Rigidbody* koja služi za postavljanje gravitacijske sile, *Audio Source* za reprodukciju zvukova - primarno zvuk za skakanje, *BoxCollider2D* koji služi za davanje oblika i dodirnih točaka igračevog lika, unutar 2D prostora, kako bi on imao poželjnu interakciju sa komponentama na sceni. U skripti su također posloženi parametri visine skoka i brzine trčanja igrača.

```
//Funkcija kretanja igrača
public class PlayerMovement : MonoBehaviour
{
    private Rigidbody2D rb;
    private BoxCollider2D coll;
    private SpriteRenderer sprite;
    private Animator anim;

    [SerializeField] private LayerMask jumpableGround;

//Funkcija za brzinu kretanja i skakanja igrača
    private float dirX = 0f;
   [SerializeField] private float moveSpeed = 7f;
   [SerializeField] private float jumpForce = 14f;

//Funkcija za određivanje igračevog stanja
    private enum MovementState { idle, running, jumping, falling }

//Funkcija za određivanje pripadajućeg zvučnog efekta
    [SerializeField] private AudioSource jumpSoundEffect;
    private void Start()
```

```
    {
        rb = GetComponent<Rigidbody2D>();
        coll = GetComponent<BoxCollider2D>();
        sprite = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //Funkcija koja ažurira jednom po frame-u
    private void Update()
    {
        dirX = Input.GetAxisRaw("Horizontal");
        rb.velocity = new Vector2(dirX * moveSpeed, rb.velocity.y);

        if (Input.GetButtonDown("Jump") && IsGrounded())
        {
            rb.velocity = new Vector2(rb.velocity.x, jumpForce);
            jumpSoundEffect.Play();
        }

        UpdateAnimationState();
    }

//Funkcija koja ažurira animacije na temelju stanja igrača
    private void UpdateAnimationState()
    {
        MovementState state;

        if (dirX > 0f)
```

```
        {
            state = MovementState.running;
            sprite.flipX = false;
        }
        else if (dirX < 0f)
        {
            state = MovementState.running;
            sprite.flipX = true;
        }
        else
        {
            state = MovementState.idle;
        }

        if (rb.velocity.y > .1f)
        {
            state = MovementState.jumping;
        }
        else if (rb.velocity.y < -.1f)
        {
            state = MovementState.falling;
        }

        anim.SetInteger("state", (int)state);
    }
    private bool IsGrounded()
    {
```

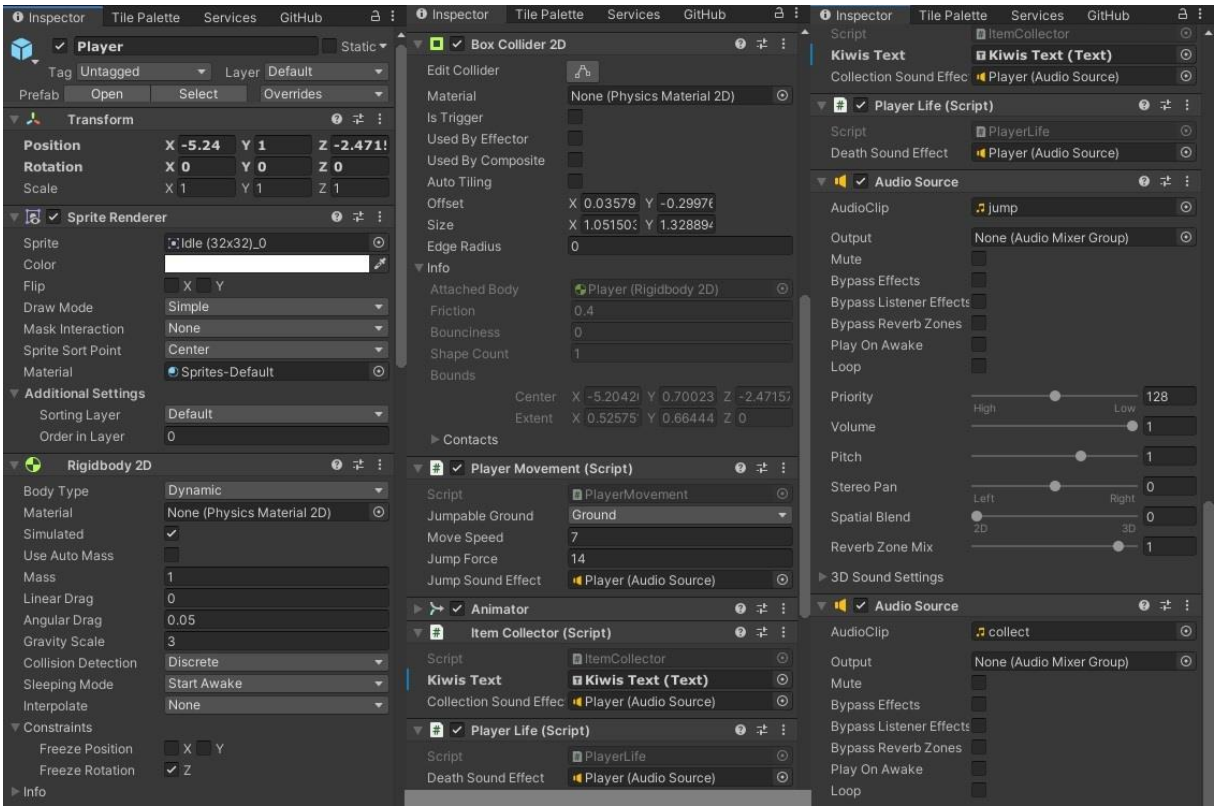
```
        return Physics2D.BoxCast(coll.bounds.center,
coll.bounds.size, 0f, Vector2.down, .1f, jumpableGround);
    }
}
```

Kod 1. Skripta za kretanje igrača

Izvor: Autor

Slika 7. Prikaz kretanja igrača

Izvor: Autor



Slika 8. Pregled postavki unutar Inspector-a

Izvor: Autor

4.4. Zdravlje igrača

Igrač ima implementirani sustav zdravlja. Svaki dodir sa neprijateljem donosi štetu i igračev lik je bačen na početak staze. Na igrača je, također, povezana animacija i zvuk kada on umre.

```
//Funkcija zdravlja igrača
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class PlayerLife : MonoBehaviour
```

```
{
    private Rigidbody2D rb;
    private Animator anim;

//Zvuk koji svira dok igrač umre
    [SerializeField] private AudioSource deathSoundEffect;

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
    }

//Funkcija koja ubija igrača ako dodirne objekt Trap
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Trap"))
        {
            Die();
        }
    }

    private void Die()
    {
        rb.bodyType = RigidbodyType2D.Static;
        anim.SetTrigger("death");
        deathSoundEffect.Play();
    }
```

```
    private void RestartLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```

Kod 2. Skripta za zdravlje igrača

Izvor: Autor

Slika 9. Prikaz skripte za zdravlje igrača kao i pripadajući zvuk u Unity-ju

Izvor: Autor

4.5. Kontrola kamere

Kontrola kamere je kratka skripta koja je postavljena da prati igrača kako se on kreće kroz stazu. Dakle, postavljena je tako da prati igrača po x, y i z osi.

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

public class CameraController : MonoBehaviour
{
```

```
    [SerializeField] private Transform player;


//Kod koji nalaže kameri da prati igračevu poziciju cijelo vrijeme
    private void Update()
    {
        transform.position = new Vector3(player.position.x,
player.position.y, transform.position.z);
    }
}
```

Kod 3. Skripta za kontrolu kamere

Izvor: Autor

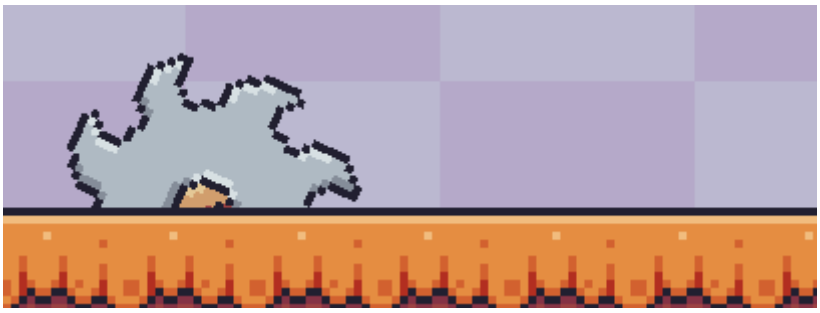
4.6. Rotacija cirkulara

Rotaciju cirkulara omogućuje sljedeća skripta, a radi na principu da određenom brzinom vrti sliku, odnosno *sprite*, oko vlastite osi.

```
//Funkcija postavlja brzinu cirkulara
public class Rotate : MonoBehaviour
{
    [SerializeField] private float speed = 5f;


    private void Update()
    {
        transform.Rotate(0, 0, 360 * speed * Time.deltaTime);
    }
}
```

Kod 4. Skripta za rotaciju cirkulara

Izvor: Autor



Slika 11. Cirkular u akciji

Izvor: Autor

4.7. Sakupljanje bodova

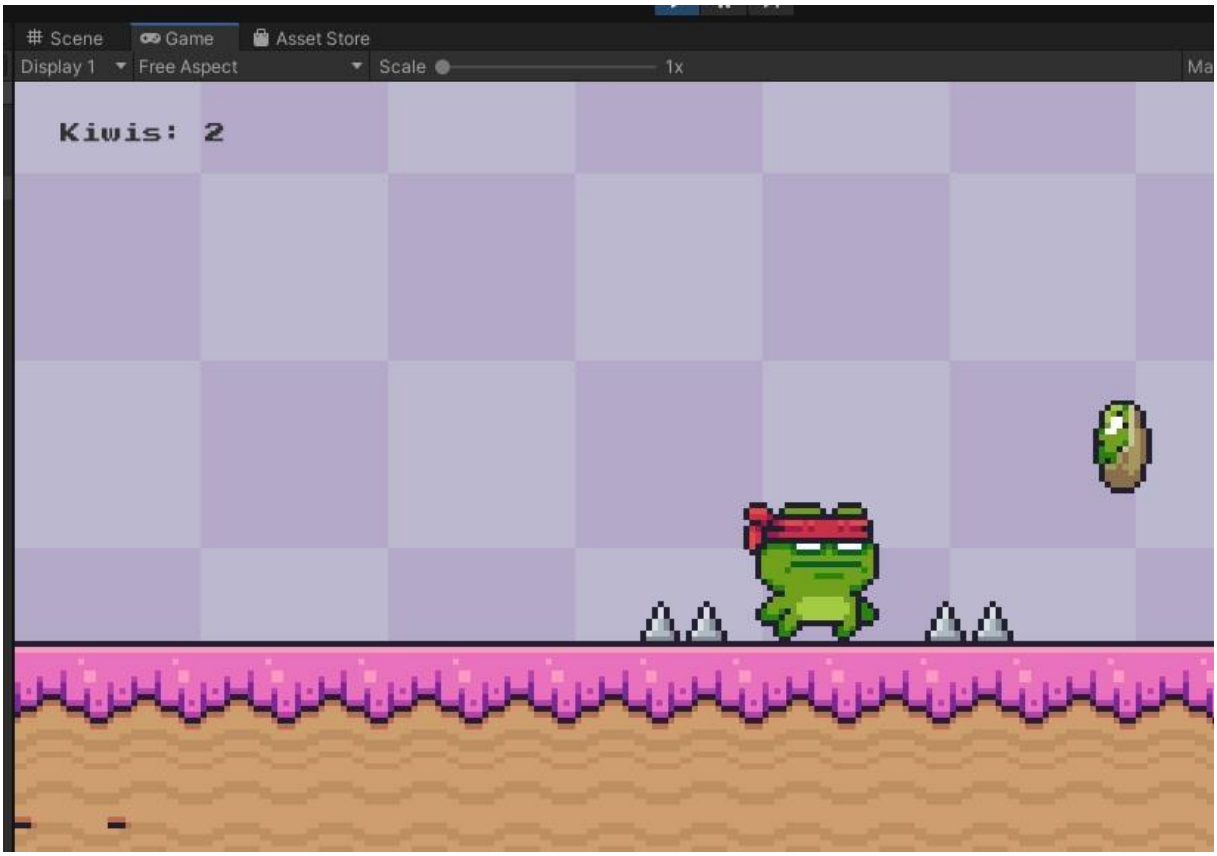
Sakupljač bodova, odnosno kivija, temelji se na tome da kad igračev *collider box* dotakne *collider box* kivija, onda taj isti kivi sakupi i njihov se broj nalazi u gornjem lijevom kutu. Također, uz navedenu akciju dolazi i odgovarajući zvučni efekt.

```
//Skripta za ispis i sakupljanje bodova, odnosno kivija
public class ItemCollector : MonoBehaviour
{
    private int kiwis = 0;

    [SerializeField] private Text kiwisText;

//Funkcija zvučnog efekta kada se sakupi kivi
    [SerializeField] private AudioSource collectionSoundEffect;
```

```
//Uvijet ako se dogodi kolizija igrača i kivija da se taj isti kivi sakupi i zapiše u gornjem lijevom kutu
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Kiwi"))
        {
            Destroy(collision.gameObject);
            kiwis++;
            kiwisText.text = "Kiwis: " + kiwis;
            collectionSoundEffect.Play();
        }
    }
}
```

Kod 5. Skripta za sakupljanje bodova

Izvor: Autor



Slika 12. Sakupljanje kivija

Izvor: Autor

4.8. Sljedbenik putne točke

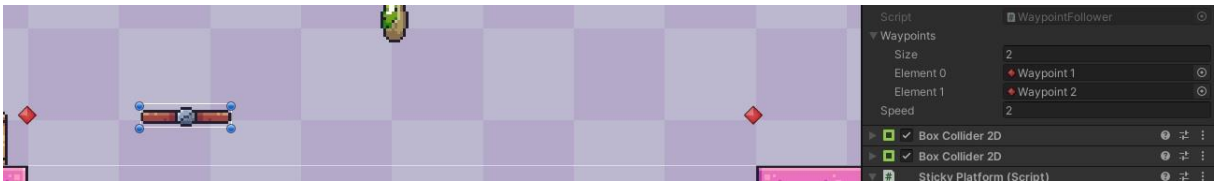
Waypoint Follower skripta je složenijeg oblika i ona nam služi kako bi omogućili kretanje lebdećih platformi u igri. Također, moramo postaviti i dio skripte koji omogućuje da se igračev lik kreće zajedno s platformom.

```
public class WaypointFollower : MonoBehaviour
{
    [SerializeField] private GameObject[] waypoints;
    private int currentWaypointIndex = 0;
```

```
//Kod za brzinu kretanja platforme
    [SerializeField] private float speed = 2f;


   private void Update()
    {
        if
(Vector2.Distance(waypoints[currentWaypointIndex].transform.position
, transform.position) < .1f)
        {
            currentWaypointIndex++;
            if (currentWaypointIndex >= waypoints.Length)
            {
                currentWaypointIndex = 0;
            }
        }

//Funkcija Time.deltaTime služi da lebdeće platforme uvijek rade
neovisno od toga jesu li prikazane na ekranu ili ne
        transform.position = Vector2.MoveTowards(transform.position,
waypoints[currentWaypointIndex].transform.position, Time.deltaTime *
speed);
    }
}
```

Kod 6. Skripta za ponašanje lebdeće platforme

Izvor: Autor



Slika 13. Prikaz puta lebdeće platforme unutar Unity-ja

Izvor: Autor

4.9. Početni meni

Start Menu skripta je vrlo jednostavna i služi nam za pozivanje funkcije početnog menu-a.

```
public class StartMenu : MonoBehaviour
{
    public void StartGame()
    {

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex +
1);
    }
}
```

Kod 7. Skripta za glavni meni

Izvor: Autor

Slika 14. Izgled početnog menu-a

Izvor: Autor

4.10. Lebdeća platforma

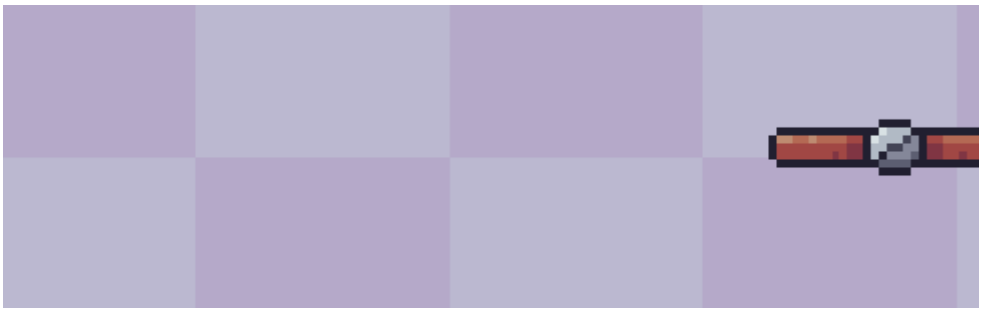
Skripta za lebdeću platformu omogućuje nam da kada igračev lik skoči na nju da se automatski pomiče zajedno sa njom.

```
public class StickyPlatform : MonoBehaviour
{

//Funkcija koja omogućuje igraču da se “zaljepi” odnosno stoji na
platform i kreće se zajedno sa njome
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.name == "Player")
```

```
        {
            collision.gameObject.transform.SetParent(transform);
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.name == "Player")
        {
            collision.gameObject.transform.SetParent(null);
        }
    }
}
```

Kod 8. Skripta za lebdeće platforme

Izvor: Autor



Slika 15. Animacija lebdeće platforme

Izvor: Autor

4.11. Kraj

Skripta za kraj uključuje pripadajući zvučni efekt i omogućuje nam ili početak nove razine ili odlaska na glavni izbornik.

```
public class Finish : MonoBehaviour
{

    private AudioSource finishSound;

    private bool levelCompleted = false;

    private void Start()
    {
        finishSound = GetComponent<AudioSource>();
    }

//Funkcija koja provjerava je li igrač prešao posljednju razinu
odnosno dotaknuo zastavicu sa odgovarajućim zvučnim efektom
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.name == "Player" &&
!levelCompleted)
        {
            finishSound.Play();
            levelCompleted = true;
            Invoke("CompleteLevel", 2f);

        }
```

```
    }


    private void CompleteLevel()

    {
SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex +
1);

    }

}
```

Kod 9. Skripta za meni kraj

Izvor: Autor

Slika 16. Izgled kraj menu-a.

Izvor: Autor

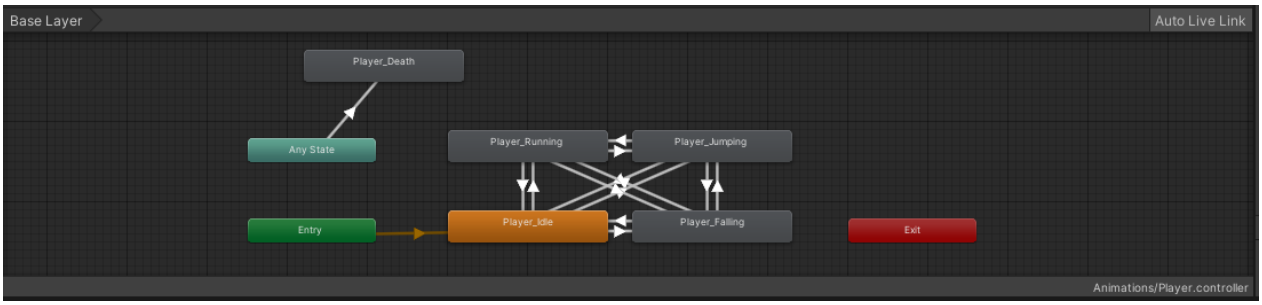
4.12. Animacije

Uloga animacija jest da daju vizualni dokaz na ekranu da smo, primjerice, pritisnuli tipku spacebar za skok i da je svijet u kojem se nalazi glavni lik živ.

Slika 17. Prikaz animacija unutar projekta u programu Unity

Izvor: Autor

Svaka od gore navedenih animacija mora biti povezana u skladu sa njenom ulogom. To postižemo tako što postavljamo stanja igračevog lika i spajamo ih sa ulogama, odnosno animacijama. Vrlo bitnu ulogu ima postavljeni okvir na igračevom liku, odnosno granice njegovog tijela u 2D svijetu. Uzmimo u obzir Player kontroler u Animatoru: kada igračev lik miruje u petlju je spojena animacija mirovanja, kada se igrač miče onda počinje animacija kretanja odnosno trčanja, kada skoči ili pada je opet posebna animacija skoka to jest pada. Zadnja 2 stanja su smrt – koja je povezana i sa odgovarajućim zvukom, a označava događaj kada se okvir igračevog lika dotakne, primjerice, cirkulara. Posljednje stanje je sakupljanje kivija, odnosno kada okvir igračevog lika dotakne kivi odigra se odgovarajući zvuk te je taj isti kivi sakupljen.

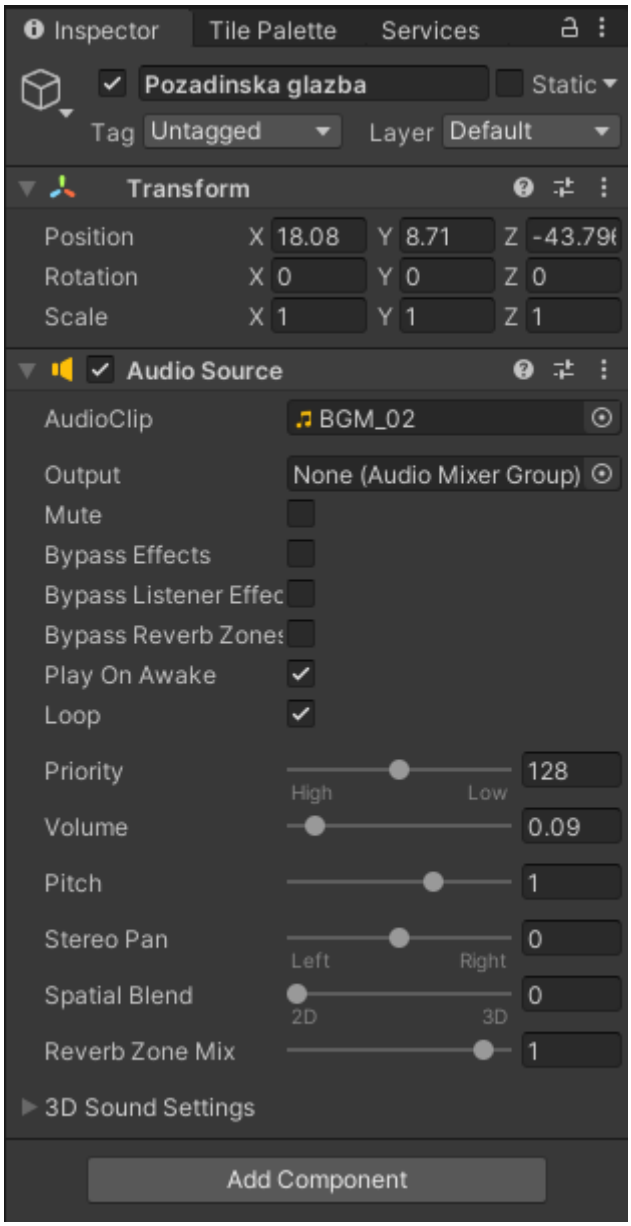


Slika 18. Prikaz AnimationsPlayer.controller-a

Izvor: Autor

4.13. Zvukovi

Unutar svake scene integrirana je pozadinska glazba koja je zadužena za atmosferičnost igre. U igri se, također, nalazi i zvuk za smrt, za završetak staze i zvuk za uspješni prelazak razine. [5]

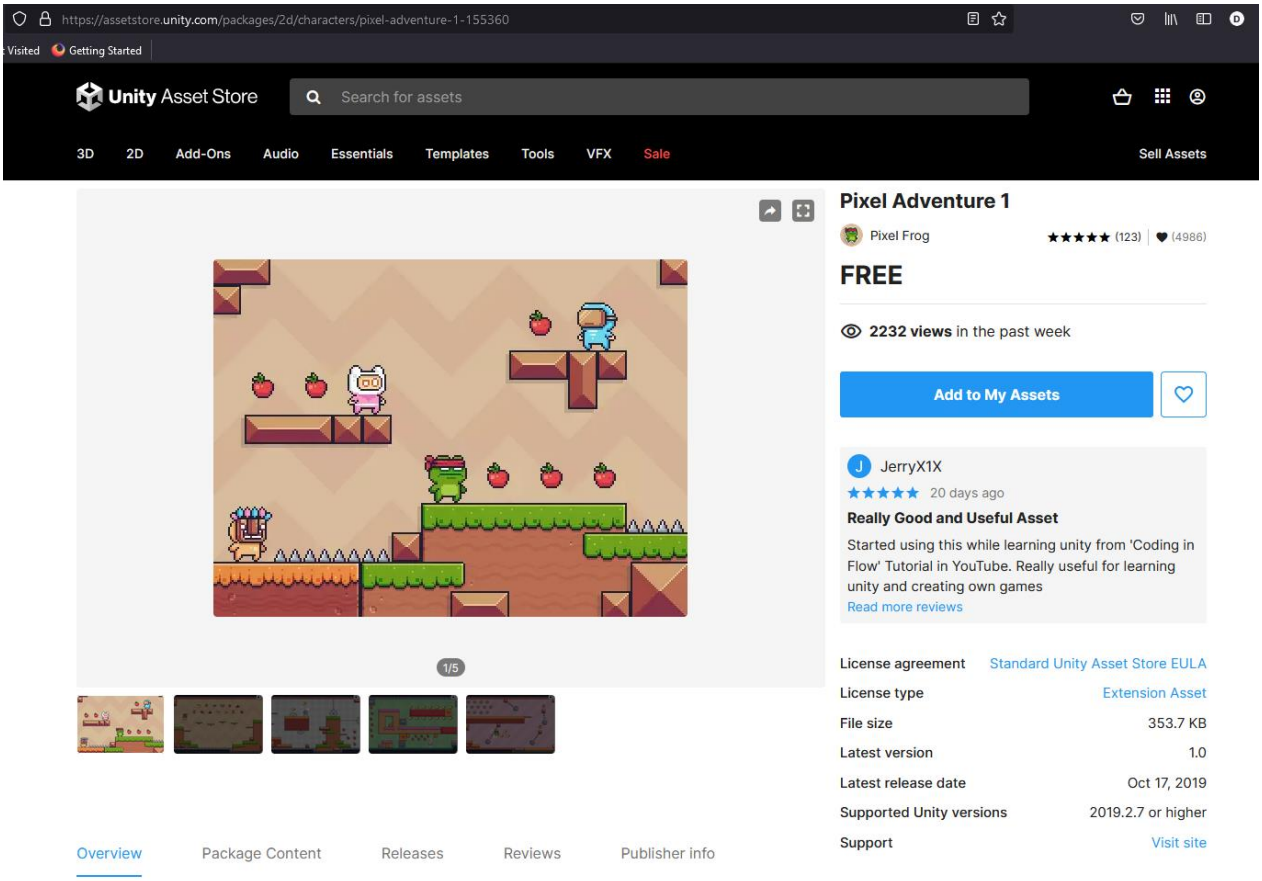


Slika 19. Prikaz Inspector *tab-a* za BG Music

Izvor: Autor

4.14. Pixel Adventure 1

Pixel Adventure 1 je besplatni *asset* preuzet sa Unity Asset Store-a i on sadržava svu vidljivu grafiku, od igračevog lika, do zamki, podloge, pozadine. [4]

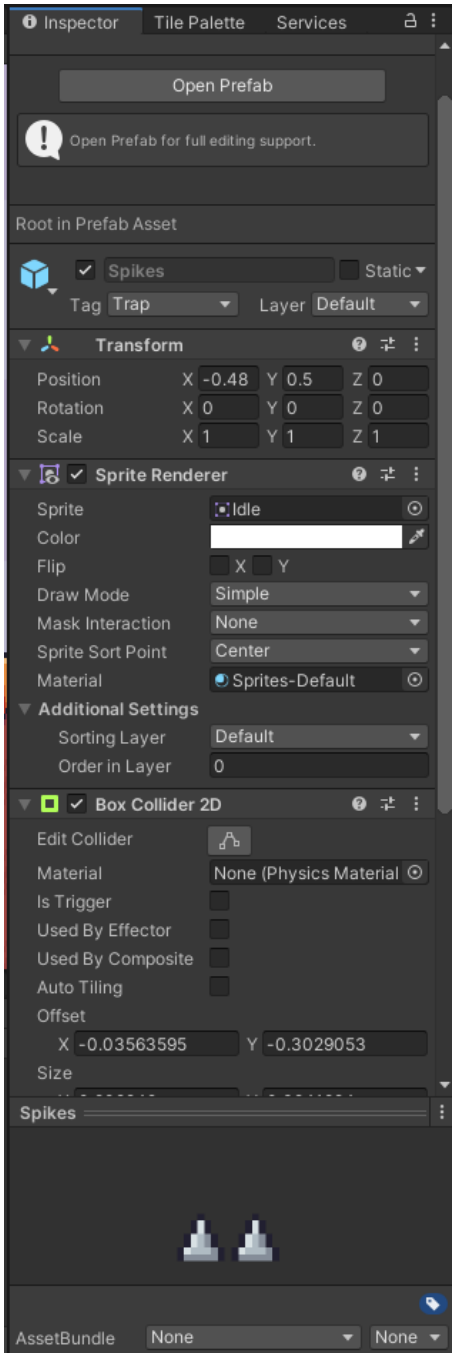


Slika 20. Pixel Adventure 1 na web-stranici Unity Asset Store-a

Izvor: Autor

4.15. Prefabs

Prefabs-i su kombinacija grafike preuzete iz Pixel Adventure 1 i podešavanja unutar izbornika Inspector.



Slika 21. Prikaz *prefab-a* šiljaka unutar izbornika Inspector

Izvor: Autor

5. ZAKLJUČAK

Izradu 2D platformera omogućava Unity 2D razvojni alat, a C# programski jezik izradu skripti za platformer. Koristeći zadane tehnologije moguće je napraviti bilo koju igru u dvodimenzionalnom prostoru – jer Unity spaja kreativnost, odnosno dizajn igre, s programiranjem. Platforma Unity nam omogućava korištenje svakakvih animacija, zvukova, glazbe, 2D i 3D modela kao i više različitih načina za korištenje istih. Unity sadržava sve aspekte koji su nam potrebni za izradu videoigre: nudi vlastiti game engine, podršku za programske jezike, animator, doslovno se može izgraditi video igra od nule. Unity Asset store pak nam nudi besplatne, a i one koji se plaćaju, resurse koji nam pomažu u izradi videoigre. Od glazbe i zvukova pa sve do grafike koje koristimo.

Smatram da je Unity jako dobar program za početnike, kao i seniore, koji se bave izradom videoigara zbog svoje sveobuhvatnosti. Dizajniranje videoigara nikada nije bilo lakše jer Unity nudi pregršt tutorial-a za izradu mnogih vrsta videoigara, od klasičnih platformera i *shooter-a*, do kompliciranijih strategija i mozgalica. Timovi zaposleni u velikim tvrtkama koje se bave videoigrama su gotovo uvijek profilirani, na način da je za svaki segment igre zadužen jedan dio zaposlenika, što uvelike ubrzava proces izrade kao i razne druge aspekte poput *brainstorminga* ili rješavanja problema. Zadatak, odnosno izrada 2D platformera, je izvršena, videoigra je igriva i funkcionalna – a može se uvelike poboljšati implementacijom novih elemenata, izradom drugih levela, dodavanjem novih mehanika i sl. Gotovo beskrajne mogućnosti i podrška koju nudi čine Unity game engine jednom od najboljih platformi za izradu videoigara.

6. POPIS SLIKA

7. POPIS KODOVA

8. LITERATURA

[1] 2D platformer (side-scroller), https://www.techopedia.com/definition/27153/side-scroller (11.3.2022.)

[2] Unity Documentation, https://docs.unity.com/ (11.3.2022.)

[3] Learn to code in Visual Studio, https://visualstudio.microsoft.com/vs/getting-started/ (11.3.2022.)

OSTALI RESURSI

[4] Pixel Adventure 1, https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360

[5] Casual Game Sounds, https://assetstore.unity.com/packages/audio/sound-fx/free-casual-game-sfx-pack-54116