Izrada trkače videoigre s pogledom odozgo u programskom alatu Unity

Drežnjak, Mihael

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mihael Drežnjak

Izrada trkače videoigre s pogledom odozgo u programskom alatu Unity

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mihael Drežnjak

Matični broj: 0016150291

Studij: Informacijski/Poslovni sustavi

Izrada trkače videoigre s pogledom odozgo u programskom alatu Unity

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Mladen Konecki

Varaždin, Rujan 2024.

Mihael Drežnjak

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

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

_____

Sažetak

Rad se bavi izradom videoigre u programskom alatu Unity. Igra je stvorena kombinacijom skripti napisanih u programu visual studio u jeziku C# i objekata stvorenih u programu Unity. Izabrani stil i osjećaj igre je temeljen na sličnim starijim igrama u pregledniku. Fizika igre je implementirana sa ciljem stvaranja arkadne igre s fokusom na proklizivanje automobila (eng. *Drifting*). Igra je beskonačna trkaća igra (eng. *Infinite racer*) u kojem se igrač natječe u vožnji okolo staze protiv računalno upravljanog protivnika dok izbjegava postavljene prepreke. Zvuk igre se sastoji od zvukova motora, guma i sudara automobila. Igra je postavljena na jednoj stazi koja zatvoreni krug(eng. *Closed circuit*).

Ključne riječi: Unity, skripta, C#, objekt, arkadno, Visual studio

Sadržaj

1. Uvod 1

2. Metode i tehnike rada 2

3. Razrada teme 3

3.1. Hijerarhija i struktura objekata 3

3.2. Mape i njihov sadržaj 16

3.3. Fizika automobila 19

3.4. Skripte 20

3.4.1. Automobil.cs 20

3.4.2. AutomobilInput.cs 24

3.4.3. Kamera.cs 25

3.4.4. TragoviHandler.cs 25

3.4.5. AutoSFX.cs 26

3.4.6. AIautomobil.cs 28

3.4.7. PutNode.cs 30

3.5. Postizanje zvuka u igri 31

3.4. Stvaranje igrive verzije (Build) 35

4. Zaključak 36

4. Popis literature 37

4. Popis slika 38

1. Uvod

Tema završnog rada je izrada dvodimenzionalne trkaće videoigre s pogledom odozgo. Izrada igre je inspirirana igrama u web pregledniku koje sam igrao tijekom svojeg djetinjastva. Videoigra će biti arkadna trkaća igra bez kraja(eng. *Infinite Racer*). Cilj igre će biti uspješno voziti oko staze koristeći vlastiti automobil bolje od protivnika kojeg će upravljati računalo.

Automobil će reagirati na unos igrača te će kretnje automobila biti utemeljene na jednostavnoj fizici automobila. Sile koje će utjecati na automobil će biti kretnja unaprijed i kretnja u stranu. Obje sile su posljedica unosa igrača te se kroz logiku programskog koda primjenjuju na objekt automobila unutar igre. Različite količine ovih brzina će također određivati i popratne efekte unutar igre poput zvuka i vizualnih efekata.

Logika igre će biti sadržana u skriptama koje su podijeljene ovisno o tome što rade. Biti će napisane u programskom jeziku C# koristeći program Visual Studio. Sve što igrač vidi i čuje će biti dio programa Unity koristeći već ugrađene komponente kao:

- Box Collider 2D – komponenta koja omogućava kolizije
- Rigid Body 2D – komponenta koja pomaže u implementaciji fizike nad objektom igre
- Trail Renderer – komponenta korištena za tragove
- Audio Mixer i Audio Source – komponente korištene za postavljanje i postizanje zvuka u igri

Tema je izabrana jer je primjer da sam razumio određene principe razvoja videoigara te da sam spreman raditi na razvoju igara tijekom svoje karijere. Razvoj videoigara je posao koji mi je uvijek bio zanimljiv i koji sam zavoljeo još više tijekom edukacije na Fakultetu organizacije i informatike zbog čega sam izabrao ovo kao temu završnog rada.

2. Metode i tehnike rada

U izradi ove videoigre biti će korišten program Unity i skripte pisane u jeziku C# sa programom Visual studio.

Program Unity (Unity Game Engine) je višeplatformski pogon za stvaranje videoigara stvoren 2005. godine. Može biti korišten za stvaranje trodimenzionalnih i dvodimenzionalnih videoigara te simulacija. Trenutna verzija programa i ujedno ona korištena u izradi videoigre je Unity 2022.3 izbačena u 2023. godini [7].

Unutar igre sve korištene teksture su uvedene iz projekta Transporter[1] dostupnog na Moodle stranici predmeta Razvoj računalnih igara. Uvedene su korištenjem opcije Assets u Unity uređivaču(eng. *Editor*). Redoslijed akcija je **Assets** > **Import new package** > **Custom package...** > odabir paketa > odabir željenih komponenti paketa > **Import**.

Na scenu su postavljeni objekti stvoreni u hijerarhiji projekta koji su obojani ili su im pridodane odgovarajuće ikone (eng. *Sprites*). Svaki objekt je postavljen na svoje mjesto, ako je potrebno pridodani su im sudarači (eng. *Collider*) i komponente za fiziku (eng. *Rigidbody 2D*). Nakon toga su im pridodane potrebne skripte.

Skripte su stvorene u mapi Scripts koja je podmapa mape Project. Skripte su pisane u jeziku C# koristeći alat Visual studio 2022 i one upravljaju logikom igre.

Tragovi guma automobila su prazni objekti kojima je dodana komponenta Prikazivač tragov(eng. *Trail Renderer*) i stvoreni materijal koji im daje željenu boju.

Zvuk igre je stvoren uvozom zvučnih zapisa koji su pridodani izvorima zvuka te podešeni koristeći Unity Audio Mixer. Zvukovi su preuzeti sa web stranice javno dostupnih zvukova Pixabay [3] [4] [5].

3. Razrada teme

U slijedećim poglavljima detaljno je razrađen svaki postupak u izradi videoigre. Prvo će biti pokrivena struktura i hijerarhija objekata u Unity uređivaču. Nakon toga ću opisati sve mape i što one sadrže. Potom ću objasniti fiziku automobila u videoigri, sve napisane skripte, postupke korištene za pravilno ostvarenje zvuka u videoigri te proces stvaranja igrive verzije (eng. *Build*).

3.1. Hijerarhija i struktura objekata

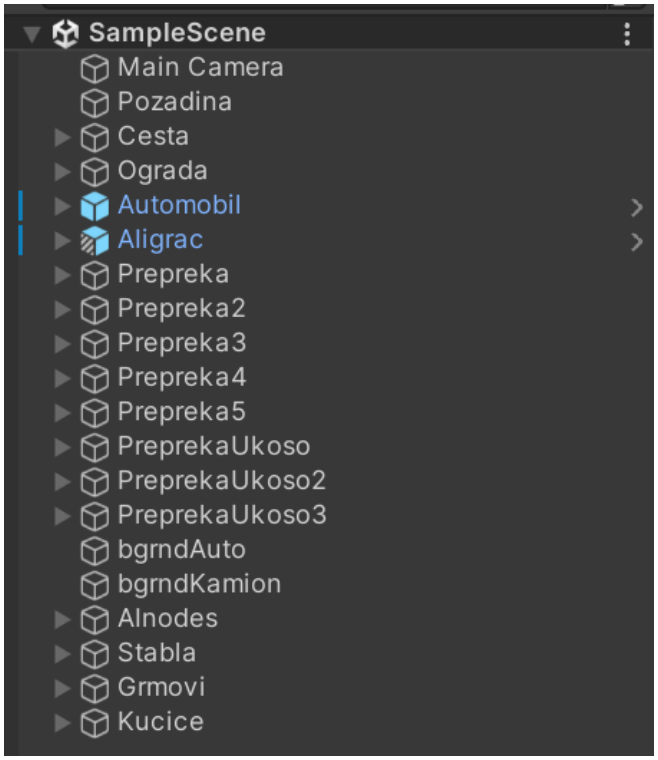
Hijerarhiija objekata u igri sastoji se od 20 objekata prve razine od samo 4 nemaju svoje podobjekte. Stvaranje objekata koji su vidljivi u svijetu igre i igraču se radi na idući način:

+ > **2D Object** > **Sprites** > odabir određenog tipa objekta

Stvaranje objekata koji služe za funkcionalnosti koje ne trebaju biti vidljivi igraču ili nisu objekti koji se konstantno pojavljuju na ekranu radi se na idući način:

+ > **Create Empty** ili **Create** > **Create Empty Child** za stvaranje praznoga podobjekta

Objekti u hijerarhiji su:

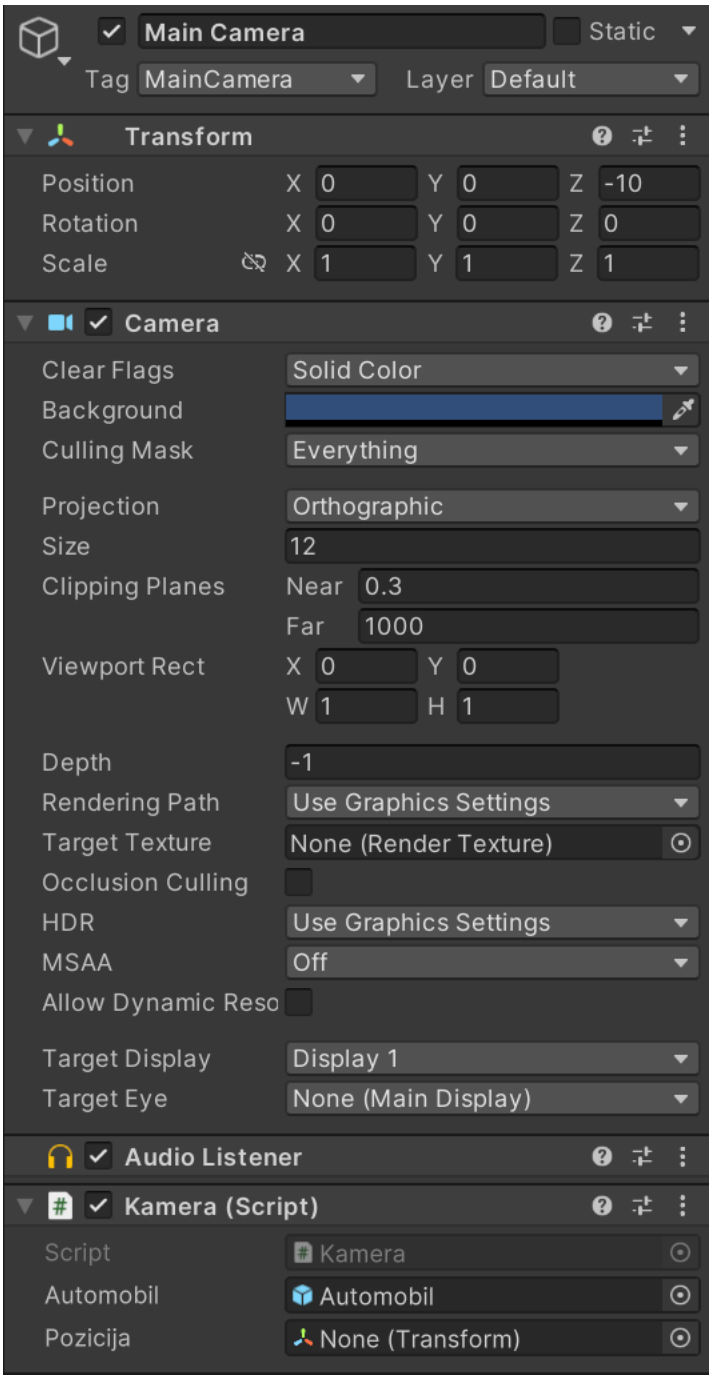


Slika 1: Prikaz hijerarhije objekata u alatu Unity

- Main Camera – kamera koja prati igrača tijekom igranja
- Pozadina – kvadrat obojen u tamnozelenu boju (hex. 298530)
- Cesta – objekt koji pod sobom sadrži sve ostale objekte od kojih je sastavljena staza
- Ograda – objekt koji pod sobom sadrži sve objekte od kojih je sastavljena ograda oko staze
- Automobil – predstavlja igrača
- AIigrac – predstavlja računalno upravljanog protivnika
- Prepreka – bijelo-narančasto obojane prepreke na stazi
- Prepreka2 – bijelo-narančasto obojane prepreke na stazi
- Prepreka3 – bijelo-narančasto obojane prepreke na stazi
- Prepreka4 – bijelo-narančasto obojane prepreke na stazi
- Prepreka5 – bijelo-narančasto obojane prepreke na stazi
- PreprekaUkoso – bijelo-narančasto obojane prepreke na stazi postavljene ukoso
- PreprekaUkoso2 – bijelo-narančasto obojane prepreke na stazi postavljene ukoso
- PreprekaUkoso3 – bijelo-narančasto obojane prepreke na stazi postavljene ukoso
- bgrndAuto – predstavlja ukras na stazi iza prepreka postavljenih ukoso
- bgrndKamion – predstavlja ukras na stazi iza prepreka postavljenih ukoso
- Ainodes – prazni objekti korišteni za upravljanje igračevog protivnika
- Stabla – ukrasi pored staze
- Grmovi – ukrasi pored staze, također korišteni kao ograde na nekim dijelovima staze
- Kucice – ukrasi pored staze

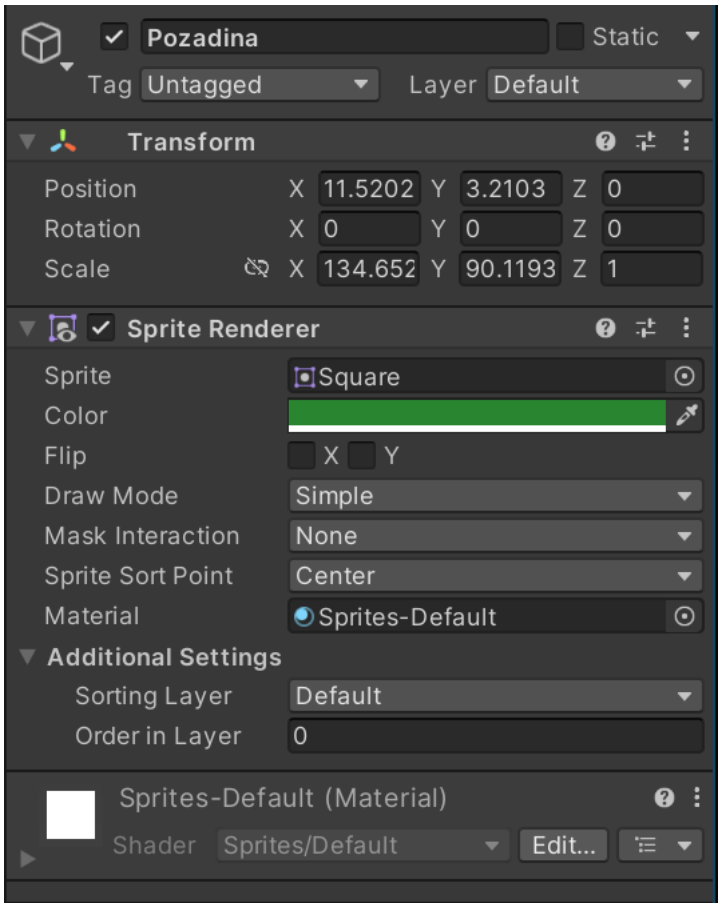
Svaki od gore navedenih objekata je prvo postavljen na scenu te postavljen na mjesto gdje je trebao biti nakon čega su obojani ili im je pridodana odgovarajuća ikona korištenjem komponenete Sprite Renderer.

Objektu Main Camera je pridodana skripta Kamera.cs za praćenje i pomicanje pozicije na poziciju igrača.



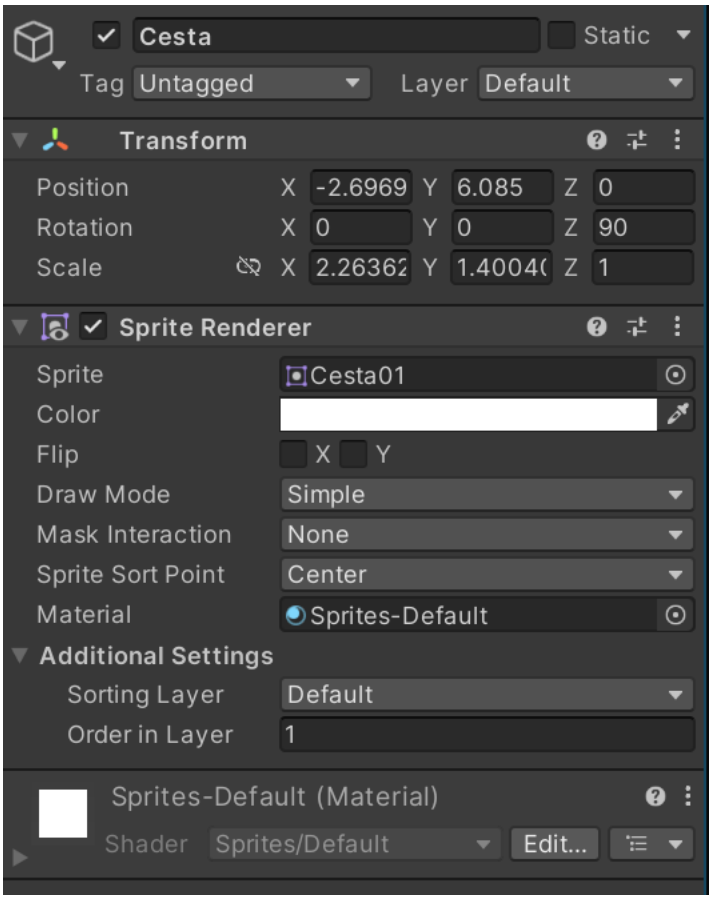
Slika 2: Main Camera objekt u inspektor prozorčiću

Pozadina je postavljena na skalu x, y, z(134.652, 90.11936, 1). Postavljena je na sloj(eng. *Layer*) 0 kako bi uvijek bila iza svih ostalih objekata u videoigri.

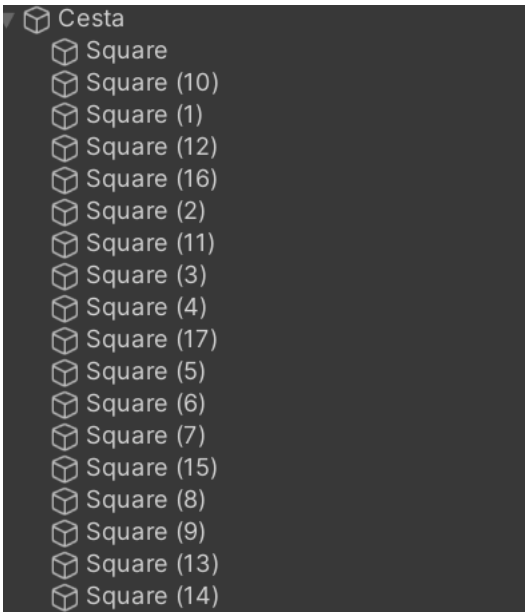


Slika 3: Objekt Pozadina

Nakon pozadine postavljeni su objekti ceste i stavljeni pod jedan glavni objekt cesta te su svi postavljeni na 1. sloj. Svakom od 18 podobjekata cesta je dodijeljena prikladna ikona ceste, ravno ako je ravna staza, skretanje ako je u pitanju skretanje.

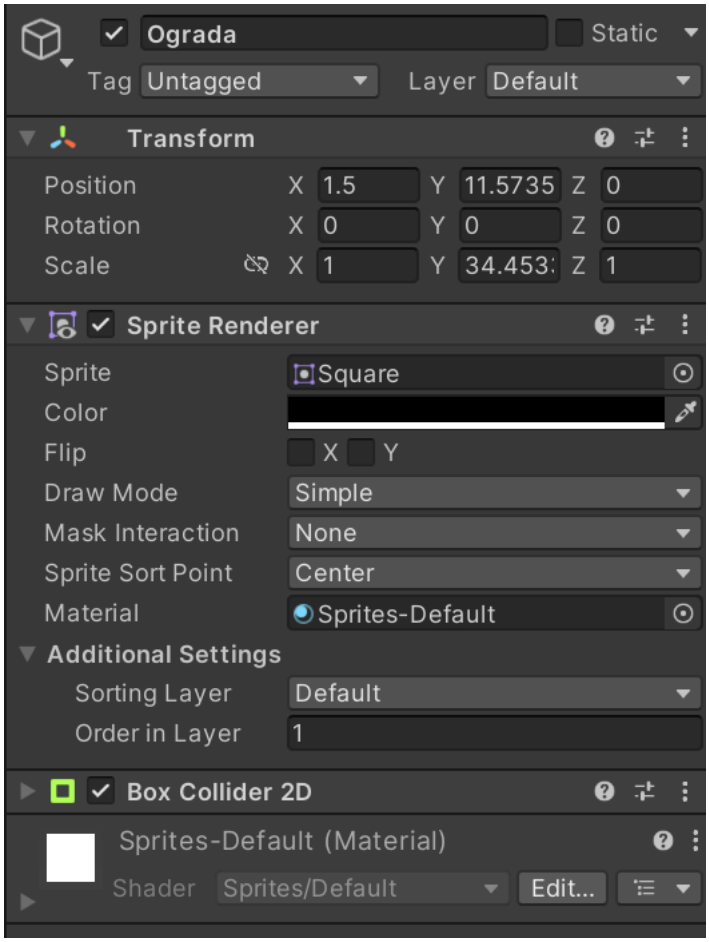


Slika 4: Prvi objekt Cesta

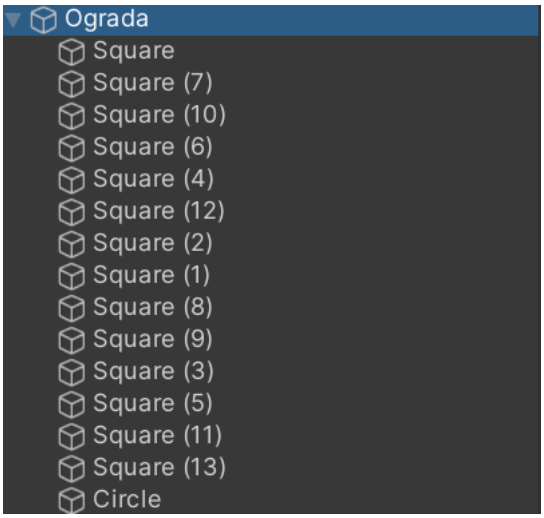


Slika 5: Objekt Cesta sa svim podobjektima koji čine stazu

Objekt ograda se sastoji od 15 podobjekata koji svi tvore cjelinu ograde okolo staze koja služi za zadržavanje igrača i protivnika na stazi. Svaka ograda je postavljena na 1. sloj, dodijeljena im je crna boja(hex. 000000) i dodijeljena im je komponenta Box Collider 2D.



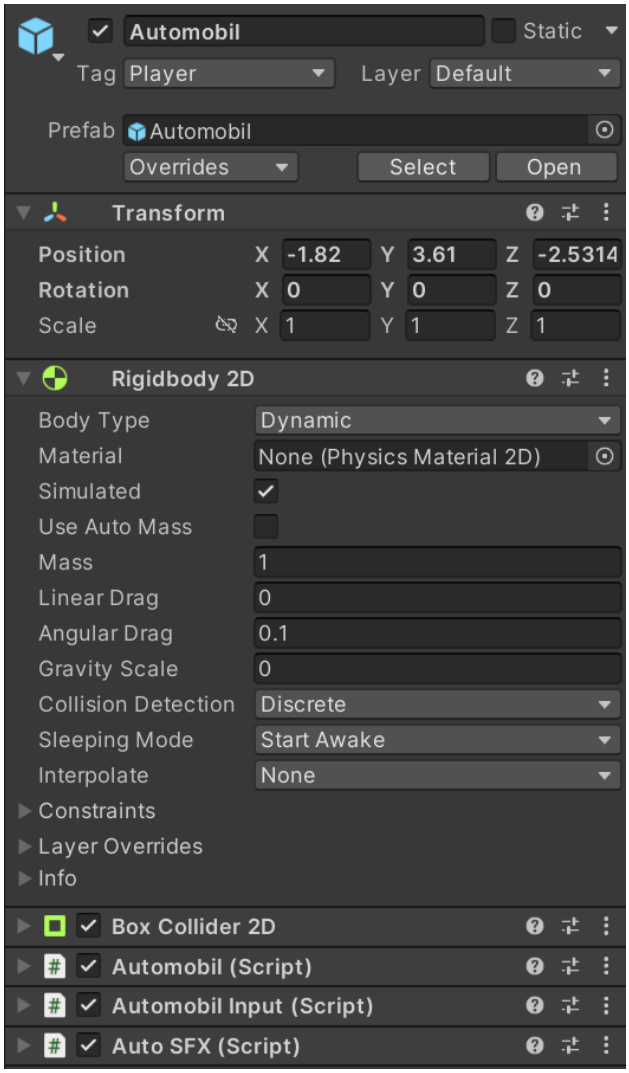
Slika 6: Glavni objekt Ograda



Slika 7: Objekt Ograda sa svim podobjektima koji čine ogradu okolo staze

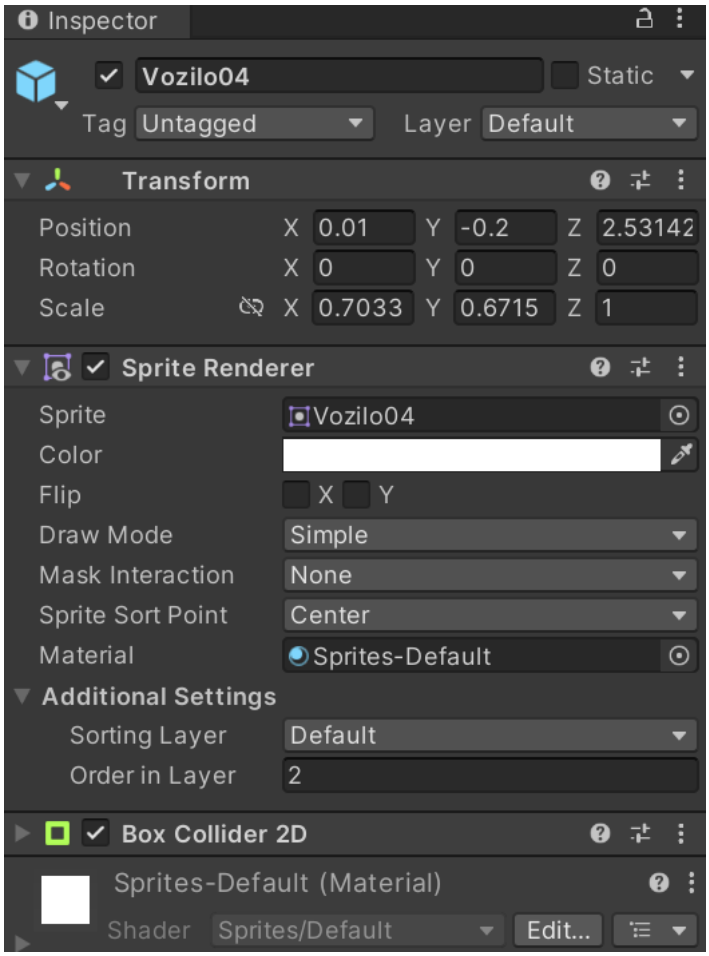
Iduće je postavljen objekt automobil kojem je pridodana ikona naziva vozilo04. Ovaj objekt je postavljen na sloj 2 te su mu tijekom razvoja pridodane komponente Rigidbody 2D za simulaciju fizike, Box Collider 2D za mogućnost sudaranja te skripte Automobil.cs, AutoSFX.cs i AutomobilInput.cs koje omogućuju upravljanje, kretnju i zvukove automobila. Koristeći ovaj objekt stvoren je i konfigurirani objekt (eng. *Prefab*) u svrhu lakšeg stvaranja drugih objekata vozila i u ovom slučaju olakšanog stvaranja objekta Aiigrac. Objektu automobil su pridodana dva podobjekta:

- vozilo04 – ima dva podobjekta, TragGumaL i TragGumaR, koji služe za učitavanje tragova guma kada je to potrebno.
- SFX – ima tri podobjekta, MotorSFX, SudarSFX i GumeSFX kojima je dodjeljena Audiosource komponenta kako bi se uvezeni zvukovi mogli aktivirati.
- Objekt Aiigrac je napravljen kao instanca Prefab objekta automobila te mu je uklonjena

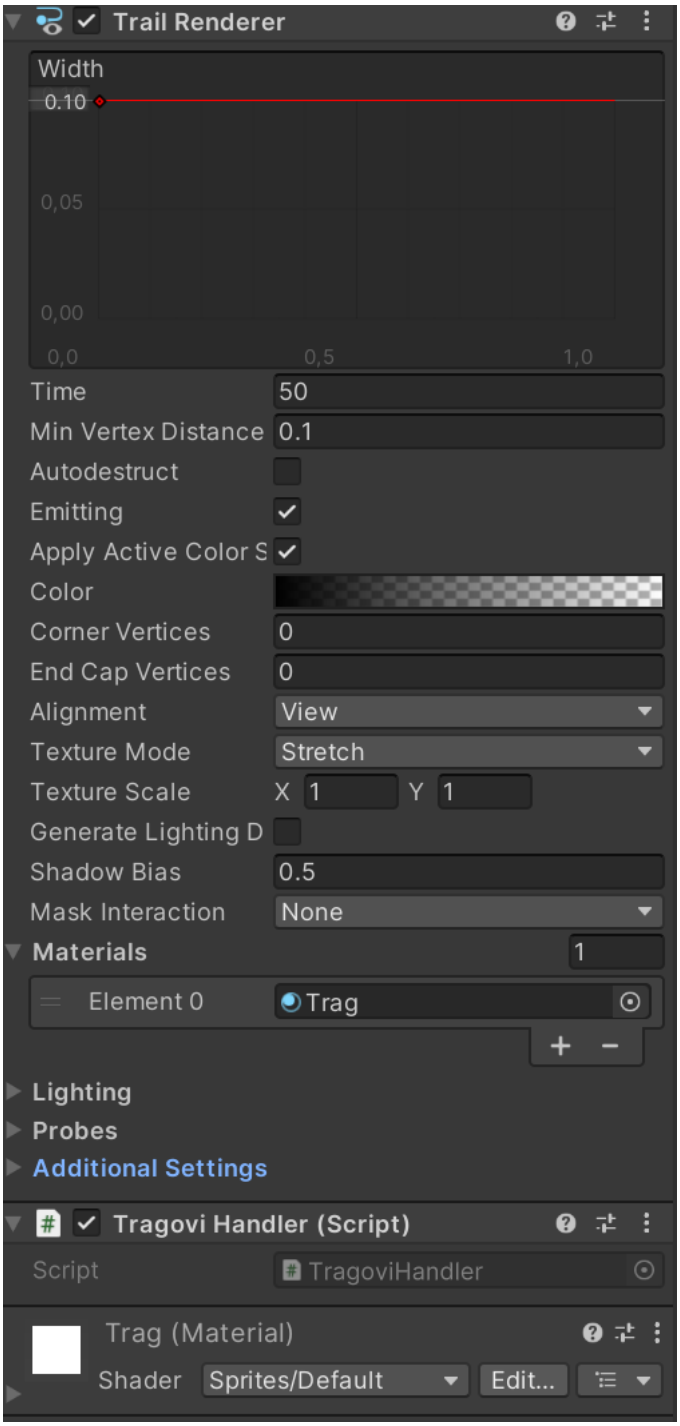


Slika 8: Objekt automobile

Slika 9: Struktura objekta I podobjekata Automobil

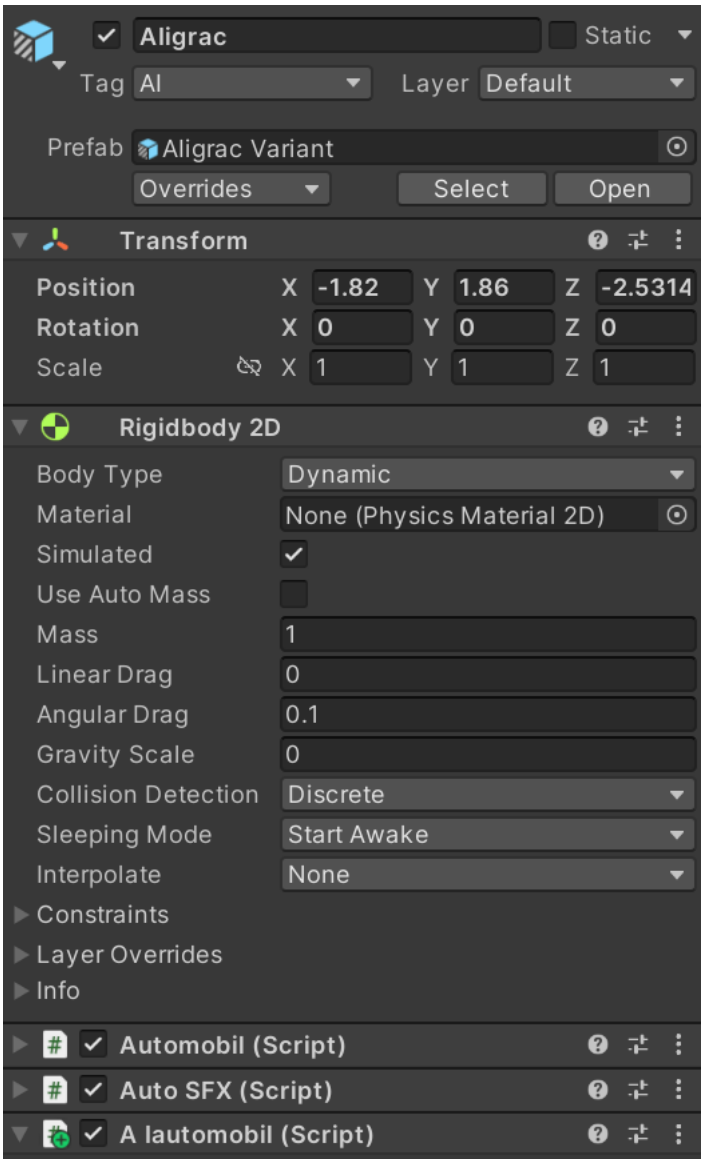


Slika 10: Vozilo04, object koji sadrži korištenu ikonu automobila



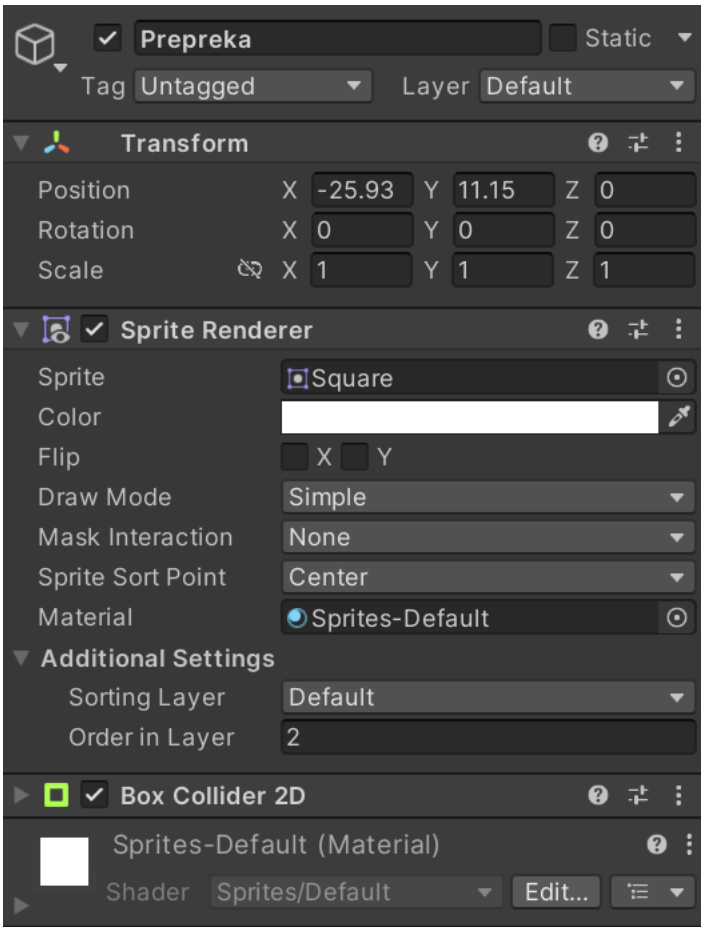
Slika 11: Objekt TragGumaL koji je zaslužan za tragove guma. TragGumaR je identičan osim što je na desnoj strain automobile

Objekt AIigrac je napravljen kao instanca Prefab objekta automobila te mu je uklonjena skripta AutomobilInput.cs te mu je dodana skripta Aiautomobil.cs za samostalno upravljanje. Ovome objektu je pridodana druga ikona, vozilo08, u svrhu lakšeg razaznanja između igrača i protivnika.

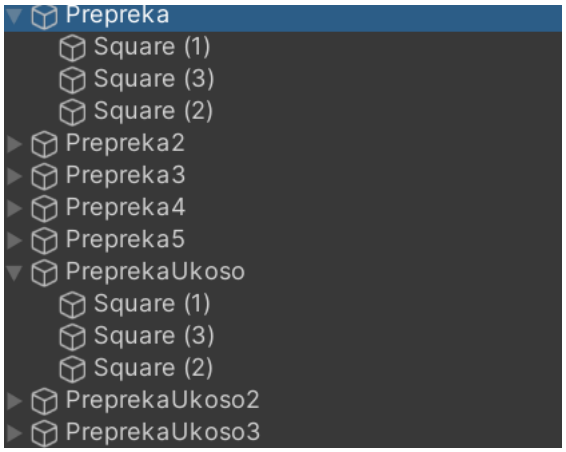


Slika 12: Objekt Aligrac

Nakon igrača i protivnika u hijerarhiji se nalazi 5 običnih i 3 prepreke ukoso. Svaka od ovih prepreka se sastoji od 4 kvadratna podobjekta obojanih naizmjenično bijelom(hex. FFFFFF) i narančastom(hex. FF6500) bojom. Prepreke su stvorene dupliciranjem prve stvorene prepreke tako da su sve postavljene na 2. sloj kako bi uvijek ostale vidljive iznad staze i pridodana im je komponenta Box Collider 2D.



Slika 13: Glavni objekt Prepreka

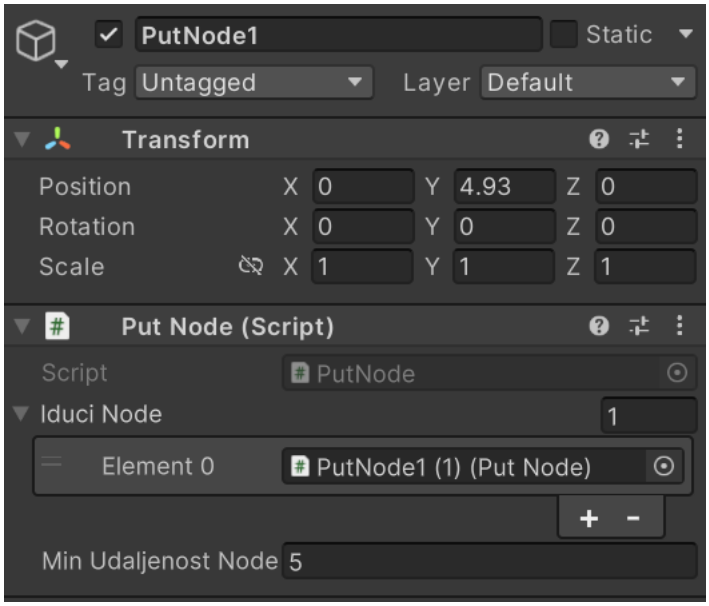


Slika 14: Sve prepreke I njihovi podobjekti

Objekti bgrndAuto i bgrndKamion su samo ukrasni objekti koji su postavljeni na stazu i okruženi ogradama u svrhu stvaranja težeg dijela staze. Postavljeni su na 2. sloj kako bi uvijek bili vidljivi te su im pridodane ikone vozilo07 i vozilo11.

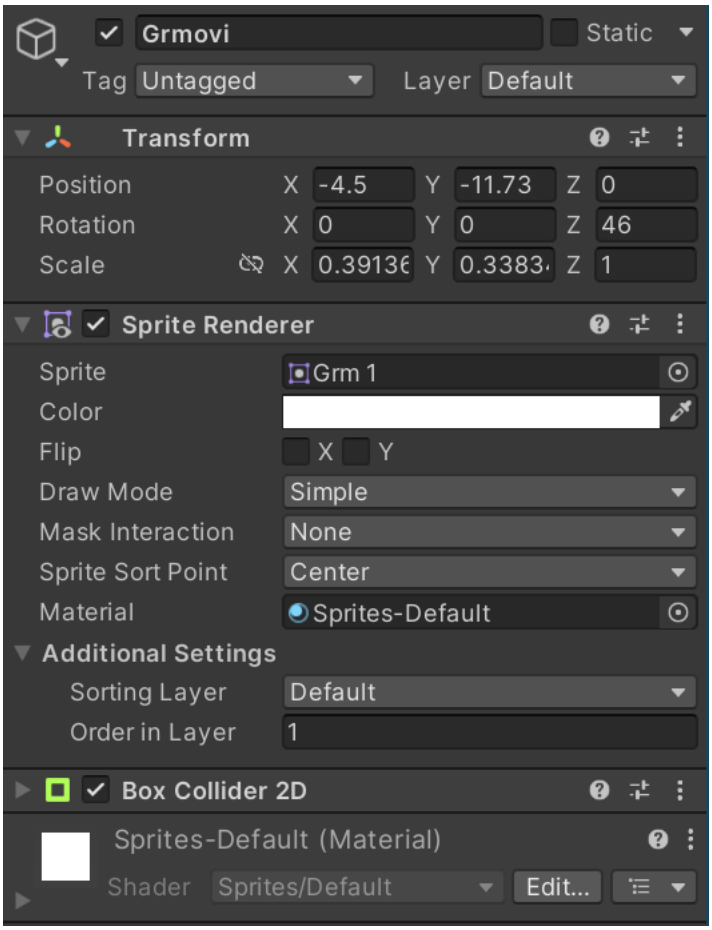
AInodes je iduća skupina objekata, ovo su prazni objekti koji u sebi imaju samo skriptu PutNodes s kojom se međusobno povezuju te tako stvaraju put koji protivnik prati po stazi. Ovaj put se sastoji od 15 međusobno povezanih točaka (eng. *Nodes*) od kojih je svaka točka povezana sa točno jednom idućom točkom i na koju je povezana isključivo jedna prethodna točka. Petnaesta i prva točka su povezane kako bi se stvorila petlja(eng. *Loop*) točaka.

Slika 15: Objekt AInodes i svi podobjekti PutNode1

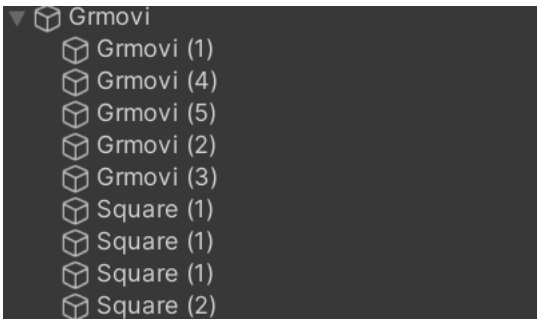


Slika 16: Objekt PutNode1 od kojeg počinje krug točaka koje prati protivnik

Grmovi je skupina objekata koja se sastoji od 9 podobjekata koji su svi postavljeni na 1. sloj zbog vidljivosti, pridodana im je Box Collider 2D komponenta jer igrač i protivnik mogu stupiti u koliziju s njima. Također služe kao ukras pored staze. Ikone pridodane grmovima su Grm 1 i Grm 2.



Slika 17: Objekt Grmovi



Slika 18: Struktura Objekta Grmovi I njegovih podobjekata

Kucice i Stabla su objekti koji služe iskjučivo kao ukras izvan staze do kojeg igrač i protivnik ne mogu doći. Pridodane su im ikone Kuca2Crvena, Kuca3Plava i Stablo.

3.2. Mape i njihov sadržaj

Slika 19: Sve mape u projektu

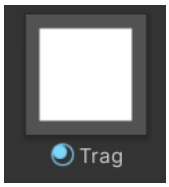
Glavna mapa Project sadrži 6 podmapa:

- AudioMix
- Materijal
- PreFab
- Scenes
- Skripte
- Sprajtovi[1]

Mapa AudioMix u sebi sadrži stvoreni mikser AudioMixer za upravljanje glasnoće zvukova u igri. Također sadrži i zvukovne zapise korištene u igri a to su clank-car-crash-collision-6206.mp3[4], engine-6000.m3[3] i rubber-tire-screech-2-202531.mp3[5] koji su preuzeti sa web stranice Pixabay.

Slika 20: Sadržaj podmape AudioMix

Materijal je mapa koja u sebi ima materijal korišten za tragove guma automobila, Trag koji je korišten kao materijal za Materials dio komponente trailrenderer.

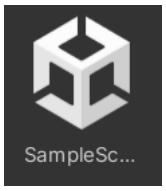


Slika 21: sadržaj mape Materijal

PreFab mapa sadrži konfigurirani objekt Automobil i njegovu instancu AIigrac Variant koji predstavljaju igrača i protivnika u videoigri.

Slika 22: Sadržaj mape PreFab

Mapa Scenes sadrži SampleScene scenu u kojoj je napravljeno sve što je opisano u izradi videoigre.



Slika 23: sadržaj mape Scenes

U mapi Skripte se nalaze sve stvorene skripte koje služe za igranje igre, mijenjanje stanja objekata i održavanje postavljenih pravila nad objektima. Skripte koje se nalaze u ovoj mapi su:

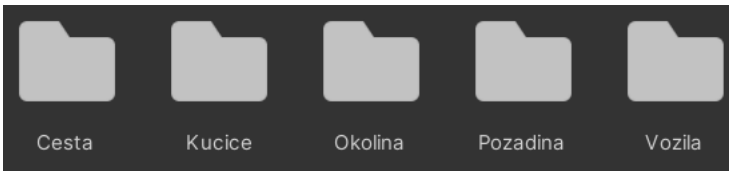
- AIautomobil.cs
- Automobil.cs
- AutomobilInput.cs
- AutoSFX.cs
- Kamera.cs

- PutNode.cs
- TragoviHandler.cs

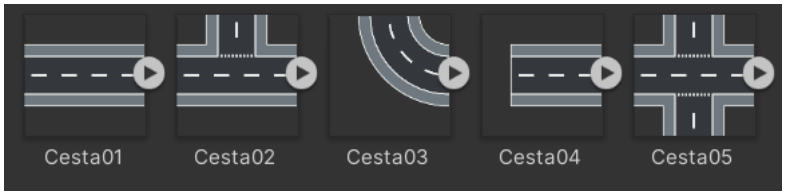
Slika 24: Sadržaj mape Skripte

Podmapa Sprajtovi je mapa uvezena sa paketom Transporter koja ima svoje vlastite podmape i ikone unutar njih [1]:

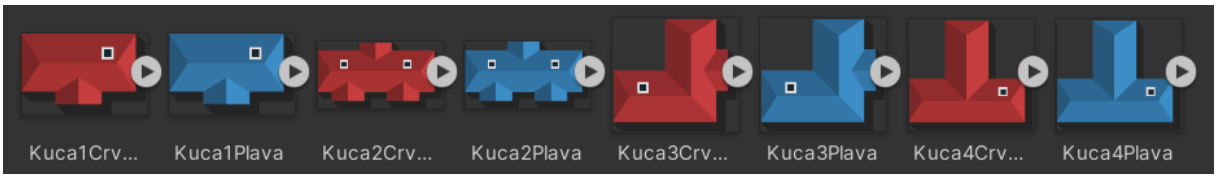
- Cesta – Cesta01.png, Cesta02.png, Cesta03.png, Cesta04.png, Cesta05.png
- Kucice – Kuca1Crvena.png, Kuca2Crvena.png, Kuca3Crvena.png, Kuca4Crvena.png, Kuca1Plava.png, Kuca2Plava.png, Kuca3Plava.png, Kuca4Plava.png
- Okolina – Bazen.png, Grm 1.png, Grm 2.png, Palma.png, Stablo.png
- Pozadina – Pozadina.png
- vozila – Vozilo01.png, Vozilo02.png, Vozilo03.png, Vozilo04.png, Vozilo05.png, Vozilo06.png, Vozilo07.png, Vozilo08.png, Vozilo09.png, Vozilo10.png, Vozilo11.png, Vozilo12.png



Slika 25: Podmape mape Sprajtovi

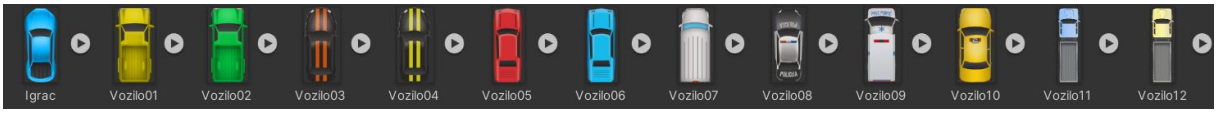


Slika 26: Sadržaj podmape Cesta



Slika 27: Sadržaj podmape Kucice

Slika 28: Sadržaj podmape Okolina

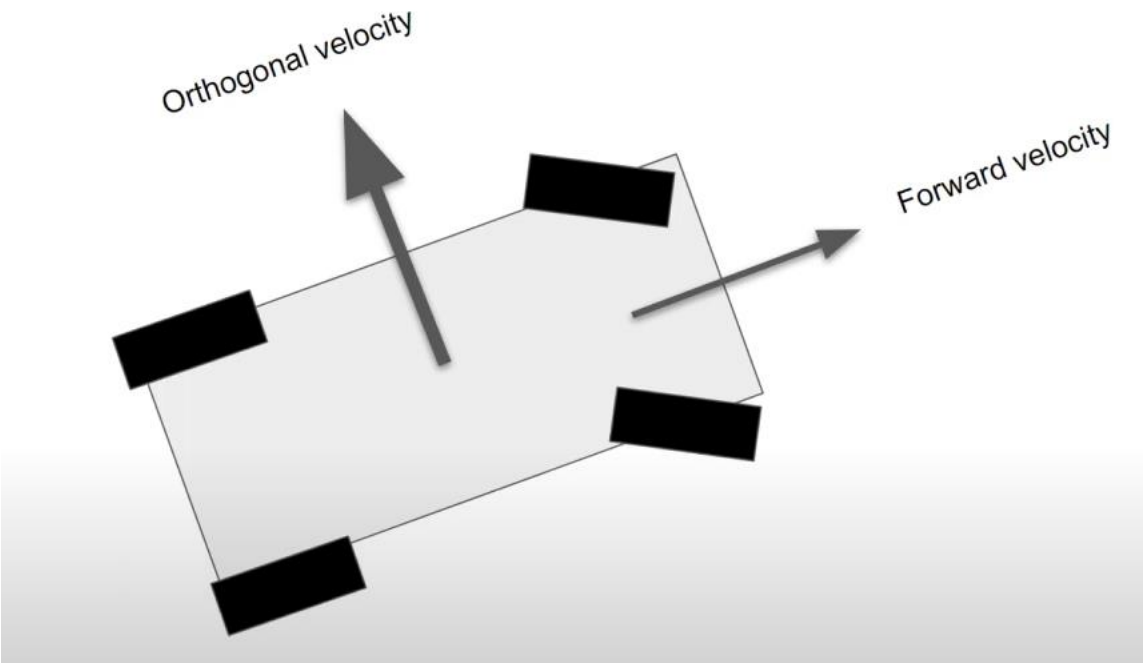


Slika 29: Sadržaj podmape Vozila

3.3. Fizika automobila

Fizika automobila u simulacijskim igrama uzima u obzir puno više sila i varijabli u kretnji nego što sam ja tijekom izrade ove igre. To je zato što je priroda igre arkadna. Simulacije u obzir uzimaju brzinu unaprijed, bocnu brzinu, silu nad svakim kotačem, silu između automobila i ceste, otpor zraka, otpor vozila, masu vozila i brojne druge[8]. Fizika igre u ovome slučaju se temelji na brzini unaprijed, bočnoj brzini i kutnom otporu koji je dio komponente Box Collider 2D.

Svaka kretnja automobila se bazira na inputu igrača i trenutnoj brzini automobila. Ovo se najviše vidi kod skretanja i pojavljivanja tragova guma. Skretanje i količina skretanja ovisi o brzini kretnje automobila dok se tragovi guma pokazuju samo kada se automobil kreće unazad ili kada je brzina u stranu automobila veća od određene količine unutar programskog koda[2].



Slika 30: Prikaz potrebnih sila za implementaciju fizike arkadne igre (Izvor: Pretty Fly Games, 2021)

3.4. Skripte

Sve skripte su napisane u programskom jeziku C# pomoću alata Visual studio. Prije stvaranja i pisanja skripti potrebno je postaviti Visual studio kao zadani program za uređivanje skripti. Postupak postavljanja zadanog programa je: **Edit** > **Preferences** > **External Tools** > **External Script Editor** > odabir Visual Studio > **Regenerate roject files**.

Nakon toga pozicioniramo se u stvorenu mapu Skripte te tamo stvaramo sve potrebne skripte, a to su:

- AIautomobil.cs
- Automobil.cs
- AutomobilInput.cs
- AutoSFX.cs
- Kamera.cs
- PutNode.cs
- TragoviHandler.cs

3.4.1. Automobil.cs

Prva skripta koja je stvorena je Automobil.cs. U ovoj skripti se nalaze sve funkcije i varijable koje omogućavaju pravilno izvođenje fizike automobila i njegovu kretnju. Također sadrži pomoćne funkcije za neke od ostalih skripti za tragove guma i zvukove automobila. Odmah na početku možemo vidjeti korištene biblioteke koje će biti iste u svakoj skripti.

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

Ovdje ćemo vidjeti sve potrebne varijable za izvršavanje skripte. Varijable su public te se njihove vrijednosti mogu izmjenjivati unutar Unity uređivača. Njihovo značenje je redom:

- driftFaktor – faktor koji se koristi za računanje količine klizanja automobila.
- faktorUbrzanja – faktor korišten za ubrzanje(eng. *Acceleration*) automobila.
- faktorSkretanja – faktor kojim se računa količina skretanja koju auotmobil izvrši nakon što igrač pritisne tipku za skretanje.
- maxBrzina – maksimalna brzina koju automobil može postići.
- ubrzanjeInput – varijabla koja je sastavni dio formule za ubrzanje automobila, prima pritisak igrača.
- skretanjeInput – varijabla korištena u formuli za skretanje, prima pritisak igrača.
- brzinaVsUp – Varijabla u kojoj je pohranjena brzina automobila.
- kutRotacije – varijabla koja određuje rotaciju automobila pri skretanju.
- Automobil2D – daje nam pristup komponenti Rigidbody2D i njenim funkcionalnostima.

```
public class Automobil : MonoBehaviour
{
    public float driftFaktor = 0.05f;
    public float faktorUbrzanja = 30f;
    public float faktorSkretanja = 3.5f;
    public float maxBrzina = 20;

    float ubrzanjeInput = 0;
    float skretanjeInput = 0;

    float brzinaVsUp = 0;

    float kutRotacije = 0;

    Rigidbody2D automobil2D;
```

Ovdje je inicijalizirana varijabla auotmobil2D u Awake funkciji kako bi se osiguralo inicijaliziranje bez obzira na moguće pogreške sa ostatkom skripte.

```
    void Awake()
    {
        automobil2D = GetComponent<Rigidbody2D>();
```

```
    }

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
```

Pozivi funkcija za brzinu, skretanje i bočnu brzinu automobila.

```
    void FixedUpdate()
    {
        SilaMotora();

        Skretanje();

        OrtagonalnaBrzina();
    }
```

Prvo što se događa u funkciji SilaMotora prikazanoj ispod je pohranjivanje brzine automobila u varijablu brzinaVsUp. Nakon toga funkcija prolazi kroz niz if selekcija gdje se osigurava poštivanje maksimalne brzine automobila unaprijed i unatrag, poštivanje brzine pri kretnji u strani ili pri kretnji koja je rezultat guranja protivnika. Selekcija if else prati input te ako on ne postoji primjenjuje otpor na automobil koji ga usporava. Otpor je postavljen na nulu ako se auto kreće uz input igrača. Na kraju funkcije vidimo računanje brzine automobila i pridodavanje sile brzine unaprijed na automobil.

```
    void SilaMotora()
    {
        brzinaVsUp = Vector2.Dot(transform.up, automobil2D.velocity);

        if (brzinaVsUp > maxBrzina && ubrzanjeInput > 0)
        {
            return;
        }

        if (brzinaVsUp < -maxBrzina * 0.5f && ubrzanjeInput < 0)
        {
            return;
        }

        if (automobil2D.velocity.sqrMagnitude > maxBrzina * maxBrzina &&
ubrzanjeInput > 0)
        {
            return;
        }

        if (ubrzanjeInput == 0)
        {
            automobil2D.drag = Mathf.Lerp(automobil2D.drag, 3.0f,
Time.fixedDeltaTime * 3);
```

```
        }
        else
        {
            automobil2D.drag = 0;
        }

        Vector2 VektorSileMotora = transform.up * ubrzanjeInput * faktorUbrzanja;
        automobil2D.AddForce(VektorSileMotora, ForceMode2D.Force);
    }
```

Funkcija Skretanje zaslužna je za održavanje pravilnog skretanja automobila. Prvo što je definirano je minimalna brzina koju automobil treba postići da bi skretanje bilo uspješno. Ovo je postignuto dijeljenjem brzine automobila brojem osam te korištenjem Clamp01 funkcije da bi se dobivena vrijednost postavila u opseg od 0 do 1 nakon čega je vrijednost iskorištena u računanju kuta rotacije koji se primjenjuje na automobil.

```
    void Skretanje()
    {
        float minBrzinaSkretanje = (automobil2D.velocity.magnitude / 8);
        minBrzinaSkretanje = Mathf.Clamp01(minBrzinaSkretanje);
        kutRotacije -= skretanjeInput * faktorSkretanja * minBrzinaSkretanje;
        automobil2D.MoveRotation(kutRotacije);
    }
```

Ova funkcija definira unos i dodjeljuje ga pravilno skretanju i ubrzanju.

```
    public void SetInputVector(Vector2 inputVector)
    {
        skretanjeInput = inputVector.x;
        ubrzanjeInput = inputVector.y;
    }
```

OrtagonalnaBrzina je funkcija koja pronalazi brzinu automobila unaprijed i u stranu te množi vrijednost brzine u stranu sa faktorom klizanja automobila kako bi se smanjila bočna brzina i postiglo proklizivanje automobila u igri.

```
    void OrtagonalnaBrzina()
    {
        Vector2 BrzinaNaprijed = transform.up * Vector2.Dot(automobil2D.velocity, transform.up);
        Vector2 BrzinaDesno = transform.right * Vector2.Dot(automobil2D.velocity, transform.right);

        automobil2D.velocity = BrzinaNaprijed + BrzinaDesno * driftFaktor;
    }
```

StupanjBrzine je funkcija koja vraća veličinu brzine i koristi se u drugoj skripti za upravljanje količine proizvedenog zvuka.

```
    public float StupanjBrzine()
    {
        return automobil2D.velocity.magnitude;
    }
```

Funkcija nadiBocnuBrzinu vraća vektor bocne brzine. Korištena je u funkciji za upravljanje škripanja guma.

```
    float nadiBocnuBrzinu()
```

```
    {
        return Vector2.Dot(transform.right, automobil2D.velocity);
    }
```

Ova funkcija osigurava pravilno praćenje stanja guma automobila. Prvo što se događa je pronalazak bocne brzine automobila. Ovo nam je potrebno u drugoj if selekciji kako bi saznali ako je auto u dovoljno jakom proklizivanju za postavljanje varijable kocenje na vrijednost istina(eng. *True*). Prva selekcija prati ako se automobil kreće unaprijed, a igrač daje unos prema nazad da također postavi varijablu kocenje na vrijednost istina. Ako ništa od navedenog nije ispunjeno varijabla kocenje ostaje na vrijednosti lažno(eng. *False*).

```
    public bool skripanjeGuma(out float bocnaBrzina, out bool kocenje)
    {
        bocnaBrzina = nadiBocnuBrzinu();
        kocenje = false;

        if(ubrzanjeInput < 0 && brzinaVsUp > 0)
        {
            kocenje = true;
            return true;
        }

        if(Mathf.Abs(nadiBocnuBrzinu()) > 6.0f)
        {
            return true;
        }

        return false;
    }
}
```

3.4.2. AutomobilInput.cs

AutomobilInput.cs je iduća stvorena skripta. Ova skripta prima unos igrača i primjenjuje ga na objekt automobil kako bi igrač mogao pokrenuti automobil. Korištene su iste bibilioteke kao i prije te je stvorena referenca na skriptu Automobil.cs koja je opet inicijalizirana u funkciji Awake za pristup potrebnim funkcijama iz skripte.

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

public class AutomobilInput : MonoBehaviour
{
    Automobil automobil;

    void Awake()
    {
        automobil = GetComponent<Automobil>();
    }
```

U Update funkciji stvorimo vektor inputVector u čije x i y komponente pohranjujemo horizontalnu i vertikalnu os te koristimo funkciju SetInputVector skripte Automobil.cs za ostvarenje pokreta.

```
        // Update is called once per frame
        void Update()
        {
            Vector2 inputVector = Vector2.zero;
            inputVector.x = Input.GetAxis("Horizontal");
            inputVector.y = Input.GetAxis("Vertical");
            automobil.SetInputVector(inputVector);
        }
    }
```

3.4.3. Kamera.cs

Odmah je stvorena i skripta Kamera.cs koja je jednostavna skripta u kojoj kamera mijenja svoju poziciju na poziciju objekta automobil te tako prati igrača tijekom igre. Na početku je korištena referenca GameObject za ustanovljenje objekta čija se pozicija pronalazi i također je stvorena varijabla pozicija za poziciju kamere na početku igre. Korišten je atribut SerializeField kako bi se u Unity uređivaču mogao kameri pridodati objekt koji mora pratiti.

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

    public class Kamera : MonoBehaviour
    {
        [SerializeField] GameObject automobil;
        public Transform pozicija;
        // Start is called before the first frame update
        void Start()
        {
            transform.position = pozicija.position;
        }

        // Update is called once per frame
        void LateUpdate()
        {
            transform.position = automobil.transform.position + new
Vector3(0f, 0f, -1f);
        }
    }
```

3.4.4. TragoviHandler.cs

Skripta TragoviHandler.cs je iduća stvorena skripta. Koristi se za upravljanjem komonentama Trail Renderer i pojavljivanjem tragova iza automobila tijekom igre. Opet je postavljena referenca na skriptu Automobil.cs te je stvorena i referenca na komponentu TrailRenderer.

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

```
using UnityEngine;

public class TragoviHandler : MonoBehaviour
{
    Automobil automobil;
    TrailRenderer tragovi;
```

Ovdje su inicijalizirane obje varijable koje su reference. Također je svojstvo tragova emitiranje(eng. *Emitting*) postavljeno na lažno kako se tragovi nebi pojavljivali konstatno od početka igre.

```
    void Awake()
    {
        automobil = GetComponentInParent<Automobil>();
        tragovi = GetComponent<TrailRenderer>();
        tragovi.emitting = false;
    }
    // Start is called before the first frame update
    void Start()
    {

    }
```

Unutar funkcije Update se odvija proces upravljanja svojstva emitiranja. Funkcija koristi funkciju iz skripte Automobil.cs kako bi provjerila ako gume skripe te ovisno o rezultatu postavlja stanje emitiranja na točno ili netočno.

```
    // Update is called once per frame
    void Update()
    {
        if(automobil.skripanjeGuma(out float bocnaBrzina, out bool
kocenje))
        {
            tragovi.emitting = true;
        }
        else
        {
            tragovi.emitting = false;
        }
    }
}
```

3.4.5. AutoSFX.cs

AutoSFX.cs je skripta koja upravlja emitiranjem zvukova unutar igre. Kao i svaka druga skripta ima referencu na skriptu Automobil.cs i također sadrži tri reference na AudioSource komponentu, jedna za svaki od mogućih zvukova igre.

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

```
public class AutoSFX : MonoBehaviour
{
    public AudioSource gumeAudiosource;
    public AudioSource motorAudiosource;
    public AudioSource sudarAudiosource;

    Automobil automobil;
```

Inicijalizacija varijable automobil.

```
    void Awake()
    {
        automobil = GetComponent<Automobil>();
    }
    // Start is called before the first frame update
    void Start()
    {

    }
```

Funkcija Update konstantno poziva funkcije za zvuk motora i zvuk guma.

```
    // Update is called once per frame
    void Update()
    {
        UpdateMotorSFX();
        UpdateGumeSFX();
    }
```

Kako bi se zvuk emitirao na određenoj razini prvo što funkcija treba je brzina automobila za što koristi funkciju iz skripte Automobil.cs i pohranjuje tu vrijednost u lokalnu varijablu. Ta varijabla se nakon toga koristi u računici za varijablu RazinaZvuka tako što se množi sa vrijednosti 0.05f. kako bi se dobio rezultat koji se onda pretvara u vrijednost između 0 i 1 funkcijom Clamp. Ovoj vrijednosti je određen raspon zvuka između 0.2f i 1.0f kao maksimum i minimum razine zvuka. Nakon toga razinu zvuka postepeno pomičemo prema toj vrijednosti.

```
    void UpdateMotorSFX()
    {
        float RazinaBrzine = automobil.StupanjBrzine();
        float RazinaZvuka = RazinaBrzine * 0.05f;
        RazinaZvuka = Mathf.Clamp(RazinaZvuka, 0.2f, 1.0f);
        motorAudiosource.volume = Mathf.Lerp(motorAudiosource.volume,
RazinaZvuka, Time.deltaTime * 10);
    }
```

Funkcija UpdateGumeSFX pregledava stanje guma automobila. Ako gume skripe i ako je varijabla kocenje ostavljena u istinito stanje zvuk škripanja guma se pomiče sa ugašenog do vrijednosti 1.0f dok su uvjeti istiniti. Slučaj else uzima bocnu brzinu automobila i bazira glasnoću zvuka na toj brzini. Ovako će se zvuk uključiti i pri proklizivanju automobila te će se njegova glasnoća temeljiti na brzini proklizivanja.

```
    void UpdateGumeSFX()
    {
        if(automobil.skripanjeGuma(out float bocnaBrzina, out bool
kocenje))
        {
```

```
                if (kocenje)
                {
                    gumeAudiosource.volume =
Mathf.Lerp(gumeAudiosource.volume, 1.0f, Time.deltaTime * 10);
                }
                else
                {
                    gumeAudiosource.volume = Mathf.Abs(bocnaBrzina) *
0.05f;
                }
            }
            else
            {
                gumeAudiosource.volume =
Mathf.Lerp(gumeAudiosource.volume, 0f, Time.deltaTime * 10);
            }
        }
```

Ovdje koristimo funkciju definiranu unutar programa Unity za otkrivanje sudara između objekata. Opet tražimo brzinu automobila ali ovaj put tijekom kolizije. Glasnocu postavljamo na vrijednost brzine pomnožene sa 0.1f te je na kraju postavljena if selekcija koja je tu kako bi osigurala da se zvuk sudara uključi ako već nije do sada.

```
        void OnCollisionEnter2D(Collision2D collision)
        {
            float relativnaBrzina = collision.relativeVelocity.magnitude;
            float glasnoca = relativnaBrzina * 0.1f;
            sudarAudiosource.volume = glasnoca;

            if (!sudarAudiosource.isPlaying)
            {
                sudarAudiosource.Play();
            }
        }
    }
```

3.4.6. AIautomobil.cs

AIautomobil skripta je iduća skripta u izradi videoigre. Zadatak ove skripte je upravljanje kretnjom suparnika od strane računala. Ovo je postignuto kroz sustav točaka koje su definirane u skripti PutNodes.cs te je u ovoj skripti stvoren način praćenja tih točaka. Rva stvar koja je drugačija u ovoj skripti je korištenje biblioteke System.Linq korišten za osposobljavanje sustava točaka koje suparnik prati. Skripta također sadrži enum za različite načine funkcioniranja računalnog protivnika od kojih je ovdje implementiran sustav točaka. Također je definiran vektor koji će biti korišten za traženje iduće pozicije računalnog suparnika. Definirane su i reference na skriptu PutNode.cs koje označavaju iduću točku u nizu varijablom trenutniNode i lista svih točaka varijablom sviNodes.

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

```
public class AIautomobil : MonoBehaviour
{
    public enum AImode { nodeSustav };
    Automobil automobil;

    Vector3 metaPozicija = Vector3.zero;
    Transform metaTransform = null;

    PutNode trenutniNode = null;
    PutNode[] sviNodes;
```

Inicijalizacija objekta automobil i popunjavanje liste svih točaka.

```
    void Awake()
    {
        automobil = GetComponent<Automobil>();
        sviNodes = FindObjectsOfType<PutNode>();
    }

    // Start is called before the first frame update
    void Start()
    {

    }
```

FixedUpdate funkcija stvara vektor za unos kretnje za računalnog suparnika. Nakon toga se poziva funkcija za praćenje točaka, input vektoru se dodijeljuju vrijednosti i te se vrijednosti proslijeđuju skripti Automobil.cs u funkciju SetInputVector. Varijable kretnje su funkcija skretanjeNaMetu za x i 1.0f za y. Vrijednost za x je funkcija koja odrađuje skretanje automobila dok vrijednost za y 1.0f označava akceleraciju koja je konstantno na maksimalnoj razini.

```
    void FixedUpdate()
    {
        Vector2 inputVector = Vector2.zero;
        pracenjeNodes();
        inputVector.x = skretanjeNaMetu();
        inputVector.y = 1.0f;
        automobil.SetInputVector(inputVector);
    }
```

Funkcija koja upravlja skretanjem računalnog protivnika. Prvo što funkcija radi je računanje vektora do iduće mete koji računa tako da od pozicije mete oduzima trenutnu poziciju automobila i normalizira taj vektor. Nakon toga pronalazimo kut između smjera kretanja i vektora mete koji se invertira. Taj kut se dijeli sa vrijednosti od 45.0f kako bi se ublažilo skretanje i smanjilo drhtanje automobila pri skretanju. Koristimo funkciju Clamp kako bi količinu skretanja promijenili u raspon skretanja koji odgovara logici drugih funkcija za kretnju automobila. Funkcija vraća količinu skretanja.

```
    float skretanjeNaMetu()
    {
        Vector2 vektorMeta = metaPozicija - transform.position;
        vektorMeta.Normalize();
```

```
                float kutMeta = Vector2.SignedAngle(transform.up,
vektorMeta);
                kutMeta *= -1;

                float kolicinaSkretanja = kutMeta / 45.0f;
                kolicinaSkretanja = Mathf.Clamp(kolicinaSkretanja, -1.0f,
1.0f);

                return kolicinaSkretanja;
            }
```

Ova funkcija dodijeljuje iduću točku računalnom protivniku. Prvo što je provjereno je ako postoji trenutna točka koju automobil prati, ako ne postoji dodijeljena mu je najbliža točka. Nakon toga ako trenutna točka postoji vrijednost varijable metaPozicija postavlja se na trenutnu točku nakon čega se računa duljina udaljenosti između trenutne pozicije i pozicije mete. Ako je udaljenost od trenutne točke manja od varijable minUdaljenostNode trenutna točka ostaje iduća točka u nizu.

```
            void pracenjeNodes()
            {
                if (trenutniNode == null)
                {
                    trenutniNode = najbliziNode();
                }

                if(trenutniNode != null)
                {
                    metaPozicija = trenutniNode.transform.position;
                    float udaljenostNode = (metaPozicija -
transform.position).magnitude;

                    if (udaljenostNode <= trenutniNode.minUdaljenostNode)
                    {
                        trenutniNode = trenutniNode.iduciNode[Random.Range(0,
trenutniNode.iduciNode.Length)];
                    }
                }
            }
```

Funkcija koja traži listu svih točaka i pronalazi onu najbližu i nju vraća kao rezultat.

```
            PutNode najbliziNode()
            {
                return sviNodes
                    .OrderBy(i  => Vector3.Distance(transform.position,
i.transform.position)).FirstOrDefault();
            }
        }
```

3.4.7. PutNode.cs

Jednostavna skripta stvorena za definiranje iduće točke u nizu do koje automobil računaolnog protivnika mora doći i minimalnu udaljenost do te točke da funkcija u prethodnoj skripti prebaci trenutnu točku cilja na onu iduću.

```
using System.Collections;
```

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

public class PutNode : MonoBehaviour
{
    public PutNode[] iduciNode;

    public float minUdaljenostNode = 5;
}
```

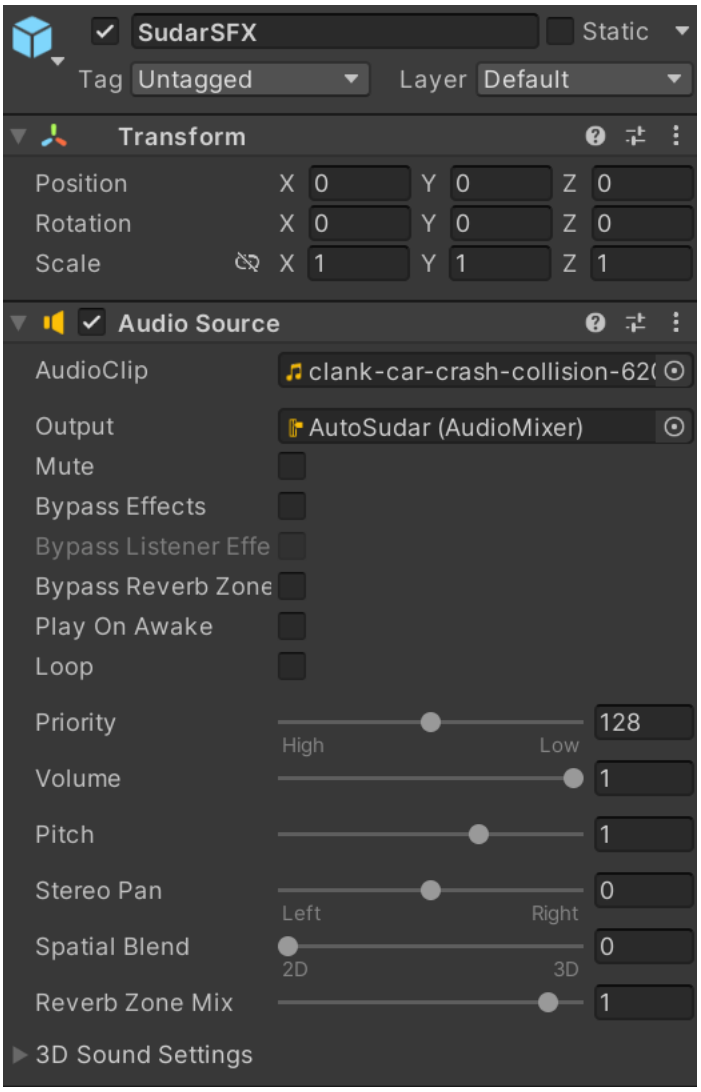
3.5. Postizanje zvuka u igri

Zvuk u igri je postignut koristeći AudioMixer i Audiosource komponente. Prvo su pronađeni odgovarajući zvukovi na web stranici javno dostupnih besplatnih zvukova. Svi zvukovi korišteni u ovoj videoigri su preuzeti sa web stranice PixaBay. Preuzeti zvukovni zapisi su:

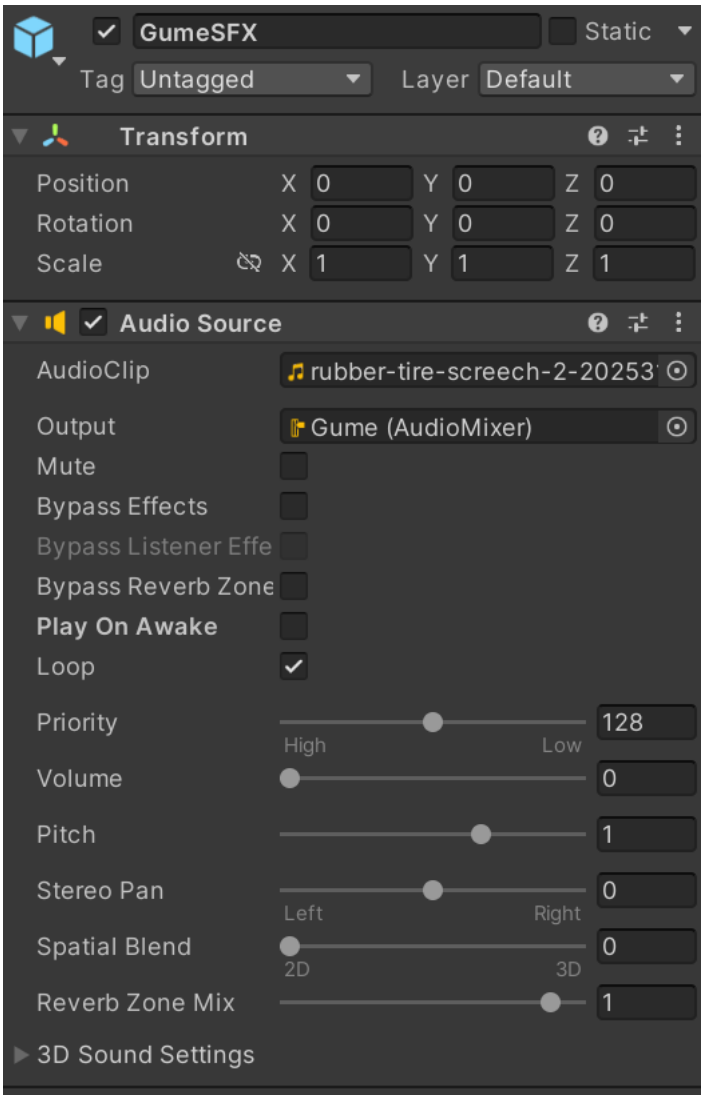
- engine-6000.mp3
- tirescreech.mp3
- clank-car-crash-collision-6206.mp3

Prvo sam stvorio AudioMixer i postavio ga u AudioMix mapu gdje su postavljeni i uvezeni preuzeti zvukovi. Komponenta AudioMixer unutar sebe ima tri zasebna miksera, AutoSudar, ZvukMotor i Gume. Objektu Automobil pridodan je podobjekt SFX kojemu su dodana 3 podobjekta MotorSFX, SudarSFX i GumeSFX. Svakom od napomenutih podobjekata objekta SFX pridodana je komponenta Audiosource. MotorSFX Audio Source komponenti pod opcijom zvučni isječak(eng. *Audio clip*) pridružen je zvučni isječak engine-6000.mp3 i pod ociju Output pridodan je ZvukMotor audiomixer komponenta. Isto je napravljeno sa SudarSFX objektom kojem su pridodani zvučni isječak clank-car-crasah-collision-6206.mp3 i AutoSudar audiomixer komponenta i GumeSFX kojem su priključeni tirescreech.mp3 zvučni isječak i Gume audiomixer. MotorSFX je postavljen sa opcijama pokreni pri buđenju(eng. *Play On Awake*) i petlja(eng. *Loop*). SudarSFX nema ni jednu od navedenih opcija zato što se uključuje jedino u slučaju kolizije. GumeSFX ima postavljenu opciju petlje ali ne i pokretanja pri buđenju.

Slika 31: MotorSFX objekt



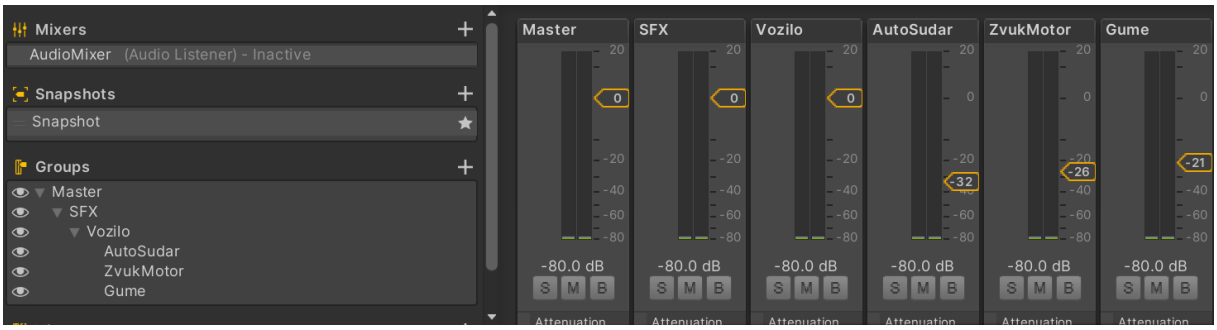
Slika 32: SudarSFX objekt



Slika 33: GumeSFX objekt

Nakon ovoga upaljena je igra u uređivaču alata Unity i na AudioMixer odjeljenju donjeg prozorčića pritisnut je gumb Edit in Play mode koji dopušta mijenjanje razine zvuka tijekom igranja u uređivaču. Ovako je postavljena maksimalna moguća razina zvuka koju svaki zvuk može postići tijekom igranja. Vrijednosti ovih zvukova su:

- AutoSudar = -32
- ZvukMotor = -26
- Gume = -21



Slika 34: Audiomixer i svi njegovi objekti

3.6. Stvaranje igrive verzije (Build)

Stvaranje igrive verzije za izvoz započinjemo preuzimanjem potrebnih modula u UnityHub programu. Izvoz ove videoigre potreban nam je modul **Windows Build Support (IL2CPP)** kojeg možemo instalirati ako u UnityHub programu odaberemo opciju **Installs** unutar njega nađemo svoju verziju programa Unity i u postavkama odaberemo **Add Modules**. Kada pronađemo traženi modul označimo ga i instaliramo.

Nakon što imamo potrebni modul trebamo otvoriti naš projekt. Kada je projekt otvoren u gornjem lijevom kutu ekrana potrebno je pronaći opciju **File** i unutar iste pritisnuti opciju **Build Settings...** nakon čega će se otvoriti novi prozorčić na ekranu sa opcijama izvoza igre. Moramo odabrati opciju **Windows, Mac, Linux**. Na vrhu prozorčića moramo odabrati koje s jittering cene iz naše igre će biti uključene u izvoz, s obzirom da ova igra ima jednu scenu ona je već odabrana. Opciju ciljana platforma(eng. *Target Platform*) ostavljamo na željenu platformu, u ovom slučaju **Windows** i pritišćemo gumb Build.

Sada se prikazuje okvir za odabir ciljanog direktorija. Ovdje je dobro stvoriti novi direktorij u koji ćemo spremiti svoj izvoz igre. Nakon toga pritišćemo gumb Odaberi(eng. *Select Folder*) i čekamo kraj sastavljanja svih naših dokumenata koji su dio projekta. Kada je to gotovo automatski će biti otvoren prozor aplikacije File Explorer unutar direktorija u kojem smo stvorili izvoz. Sada trebamo otići na razinu direktorija iznad ovoga i tamo stvoriti zip verziju direktorija u koji smo spremili igru. Ovako je uspješno stvoren izvoz ili Build igre[6].

4. Zaključak

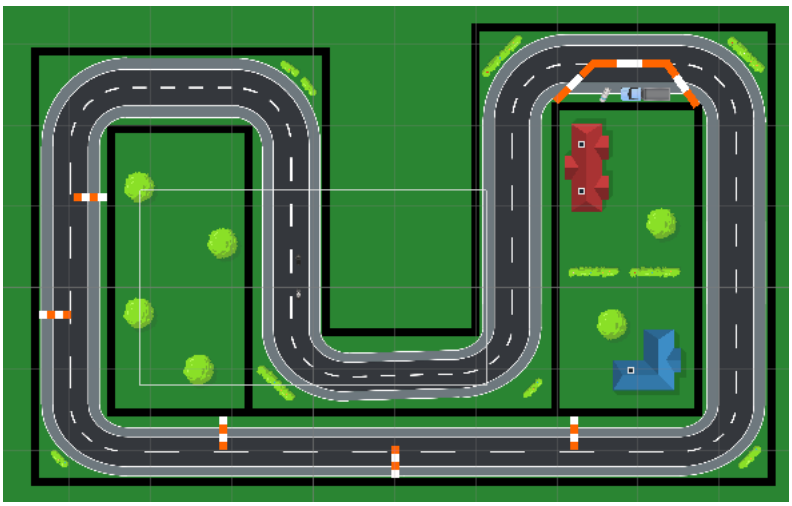
Uspješno stvorena i izvezena videoigra je rezultat završnog rada. Koristeći program za razvoj videoigara Unity version 2022.3 i program za uređivanje koda i skripti Visual Studio 2022 razvijena je trkaća igra arkadnog stila sa pogledom odzgo.

Koristeći jednostavnu fiziku temeljenu na brzini naprijed i u stranu igrač ima osjećaj kretnje kroz stazu automobilom koji ima mogućnost proklizivanja kako bi se poboljšao arkadni osjećaj videoigre.

Skripte su napisane efikasno i rade sve što je planirano. Jedini izuzetak je stvaranje više opcija za ponašanje računalnog protivnika u igri. Ono što je trebalo biti napravljeno jest da protivnik ima opciju koja kada je uključena osigurava da protivnik lovi igrača što nije implementirano zbog problema sa ponašanjem protivničkog automobila pri testiranju. Zbog toga je implementirana samo druga opcija kretnje koja je sustav točaka gdje protivnik prati sustav nevidljivih točaka postavljenih po stazi.

Vizualni i zvukovni efekti su uspješno implementirani u igru. Svaki zvuk i efekt ima svoje vlastite okidače u kodu koji rade točno kako je zamišljeno. Osim toga zvukovi se mijenjaju u glasnoći unutar raspona dopuštenog od strane igre ovisno o kretnji automobila.

Igra je uspješno izvedena i postoji kao izvršna datoteka unutar datoteke koja je komprimirana i priložena uz završni rad.



Slika 35: Izgled scene na kraju izrade videoigre

5. Popis literature

Cohen, D. S., (2021.). *Unity Tutorial - Creating a PC or Mac Build*. https://www.youtube.com/watch?v=USfiICS7HCM [Youtube]

Dealessandri, M., (16.1.2020.). *What is the best game engine: is Unity right for you?*. https://www.gamesindustry.biz/what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you

floraphonic, (bez dat.). *Rubber Tire Screech 7.* Preuzeto 27.8.2024 s https://pixabay.com/sound-effects/rubber-tire-screech-7-202580/

Konecki, M., (bez dat.). *Transporter*. Preuzeto 7.3.2024. s https://elfarchive2324.foi.hr/course/view.php?id=743&section=3 [Moodle]. Sveučilište u Zagrebu, Fakultet organizacije i informatike, Varaždin

Pixabay, (bez dat.). *Crash.* Preuzeto 27.8.2024. s https://pixabay.com/sound-effects/crash-7075/

Pixabay, (bez dat.). *Engine.* Preuzeto 27.8.2024. s https://pixabay.com/sound-effects/engine-6000/

Pretty Fly Games, (2022.). *2D Arcade style top down car controller in Unity*. https://www.patreon.com/prettyflygames/posts

The Henry Ford Organization, (bez dat.). *Forces in automobile racing*. Preuzeto 2.8.2024. s https://ophelia.sdsu.edu:8443/henryford_org/06-15-2014/education/erb/PhysicsAutoRacingUnitPlanBackground2A.pdf

6. Popis slika

Slika 1: Prikaz hijerarhije objekata u alatu Unity....3
Slika 2: Main Camera objekt u inspektor prozorčiću....5
Slika 3: Objekt Pozadina....6
Slika 4: Prvi objekt Cesta....7
Slika 5: Objekt Cesta sa svim podobjektima koji čine stazu....7
Slika 6: Glavni objekt Ograda....8
Slika 7: Objekt Ograda sa svim podobjektima koji čine ogradu okolo staze....8
Slika 8: Objekt automobil....9
Slika 9: Struktura objekta I podobjekata Automobil....10
Slika 10: Vozilo04, object koji sadrži korištenu ikonu automobila....10
Slika 11: Objekt TragGumaL koji je zaslužan za tragove guma. TragGumaR je identičan osim što je na desnoj strain automobila....11
Slika 12: Objekt AIigrac....12
Slika 13: Glavni objekt Prepreka....13
Slika 14: Sve prepreke I njihovi podobjekti....13
Slika 15: Objekt AInodes i svi podobjekti PutNode1....14
Slika 16: Objekt PutNode1 od kojeg počinje krug točaka koje prati protivnik....14
Slika 17: Objekt Grmovi....15
Slika 18: Struktura Objekta Grmovi I njegovih podobjekata....15
Slika 19: Sve mape u projektu....16
Slika 20: Sadržaj podmape AudioMix....16
Slika 21: sadržaj mape Materijal....17
Slika 22: Sadržaj mape PreFab....17
Slika 23: sadržaj mape Scenes....17
Slika 24: Sadržaj mape Skripte....18
Slika 25: Podmape mape Sprajtovi....18
Slika 26: Sadržaj podmape Cesta....18
Slika 27: Sadržaj podmape Kucice....19
Slika 28: Sadržaj podmape Okolina....19
Slika 29: Sadržaj podmape Vozila....19
Slika 30: Prikaz potrebnih sila za implementaciju fizike arkadne igre....20
Slika 31: MotorSFX objekt....32
Slika 32: SudarSFX objekt....33
Slika 33: GumeSFX objekt....34
Slika 34: Audiomixer i svi njegovi objekti....35
Slika 35: Izgled scene na kraju izrade videoigre....36