Izrada trkače 2D igre u Unity-u

Klopotan, Jakov

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

JAKOV KLOPOTAN

IZRADA TRKAČE 2D IGRE U UNITY-U

ZAVRŠNI RAD

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

JAKOV KLOPOTAN

IZRADA TRKAČE 2D IGRE U UNITY-U

MAKING A RUNNER 2D GAME IN UNITY

ZAVRŠNI RAD

Mentor:
Nenad Breslauer, viši predavač

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
ODBOR ZA ZAVRŠNI RAD

Čakovec, 5. siječnja 2021.

Polje: **2.09 Računarstvo**

ZAVRŠNI ZADATAK br. 2020-RAČ-R-12

Pristupnik: **Jakov Klopotan (0313022139)**
Studij: redovni preddiplomski stručni studij Računarstvo
Smjer: Inženjerstvo računalnih sustava i mreža

Zadatak: **Izrada trkače 2D igre u Unity-u**

Opis zadatka:

Ideja je napraviti trkaću igru s autima uz pomoć Unity game Engine platforme. 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. Igra će biti trkaćeg tipa u kojem auti skupljaju bodove. Cilj igre je postići što bolji rezultat. Koristiti platformu Unity, programski jezik C# te dodatne programske alate, Blender i ostale. Završni rad mora sadržavati sažetak, sadržaj i uvod, nakon čega slijedi poglavlje u kojem je potrebno navesti i pojasniti osnovne ciljeve rada te očekivani rezultat. U narednom poglavlju potrebno je opisati primijenjene postupke, alate i metode. Poglavlje koje slijedi obrađivati će postignute rezultate nakon čega slijedi poglavlja u kojem se kritički raspravlja o primijenjenim metodama i postupcima te se u narednom poglavlju iznose glavni zaključci rada. Rad se završava poglavljima s popisom literature te prilozima.

Zadatak uručen pristupniku: 5. siječnja 2021.
Rok za predaju rada: 20. rujna 2021.

Mentor:

Nenad Breslauer, v. pred.

Predsjednik povjerenstva za završni ispit:

Zahvala

Zahvaljujem svom mentoru Nenadu Breslaueru na strpljenju, savjetima, razumijevanju i pomoći kod izrade završnog rada.

Želim se zahvaliti i svojoj obitelji što je uvijek bila uz mene tijekom studija i što su mi bili podrška tijekom cijelog školovanja.

Također hvala svim profesorima Međimurskog Veleučilišta na prenesenom znanju i savjetima.

Jakov Klopotan

Sažetak

Prije nekoliko godina mišljenje većine ljudi kretalo se u smjeru da je igranje videoigra gubitak vremena. Smatralo se da igranje videoigara ostavlja isključivo negativne posljedice na psihofizički razvoj čovjeka, izaziva ovisnost, uništava vid i potiče razvoj nasilničkih obrazaca ponašanja.
Današnja istraživanja pokazuju da igranje videoigra ima mnogo pozitivnih strana [1]. Osim što mnogo djece, a i starijih ljudi, zarađuje i živi od igranja videoigra, one dokazano daju veliki značaj psihološkom razvoju djece. Istraživanja su pokazala da pridonose razvoju logičkog razmišljanja, sposobnosti rješavanja problema te poboljšavaju kritičko razmišljanje. Kod djece se razvija i znatno bolja kombinacija oko – ruka te preciznost. Također uče kontrolirati svoje reakcije i umanjiti utjecaj stresa na vlastite odluke i postupke, raditi u timu, voditi tim do uspjeha i željenog cilja. Iz svega navedenog može se zaključiti kako igranje videoigra nije toliko loše.
Ovaj završni rad prikazivat će izradu 2D igre korak po korak. Koristit će se jedno od najpoznatijih razvojnih okruženja za izradu videoigra *Unity*. Proces izrade igre ne zahtjeva samo znanje i vještine unutar programa *Unity*, već i osnovne vještine programiranja u C# jeziku i dizajniranje određenih modela. Za funkcionalnost objekta i same scene koriste se kodovi pisani u programskom jeziku C#, za pisanje skripti koristi se program *Visual Studio*, a kod izrade slika ceste, i dizajniranja početne i završne scene koristi se program *Adobe Photoshop 2020*.
Igra će biti trkaćeg tipa u kojem glavni automobil izbjegavajući ostale automobile i prepreke skuplja novčiće i na taj način osigurava prijelaz na iduću razinu. Jedan novčić donosi deset bodova, a na svakoj razini potrebno je ostvariti sto bodova, odnosno pokupiti deset novčića kako bi se taj razina uspješno završila. Svaka nova razina je sve zahtjevnija, brzina stvaranja i kretanja protivničkih automobila se povećava, a glavni automobil se kreće većom brzinom te ga je zbog toga sve teže kontrolirati. Pojavljuju se i nove prepreke, odnosno smetnje na cesti koje dodatno otežavaju prelazak razine.

Ključne riječi: *videoigra, 2D, Unity, objekt, scena, C#*

SADRŽAJ

1. UVOD 7
2. IGRAČ I OKOLINA 8
2.1. Postavljanje projekta i scene 8
2.2. Dodavanje glavnog objekta 11
2.3. Postavljanje rezolucije i pozadine 12
2.4. Osvjetljenje scene 15
2.5. Pomicanje automobila 16
2.6. Protivnički automobili 19
2.7. Sudari 25
2.8. Izrada gumba za pauzu 35
3. POČETNA SCENA 38
3.1. Panel s uputama 42
4. BODOVI 45
4.1. Novčići 46
5. GAME MENI 49
6. ZVUK 55
6.1 Zvuk automobila 55
6.2. Zvuk novčića 58
7. EKSPLOZIJA 61
8. PRIJELAZ NA IDUĆU RAZINU 64
8.1. Izrada druge razine 65
8.1.2. Dodavanje prepreke 66
8.2. Izrada treće razine 68
9. ZAVRŠNA SCENA 69
10. ZAKLJUČAK 73
POPIS LITERATURE 74
POPIS PROGRAMSKIH KODOVA 75
POPIS SLIKA 76

1. UVOD

U ovom završnom radu ideja je napraviti trkaču igru s automobilima uz pomoć *Unity* razvojnog okruženja. Kod prikupljanja modela za automobile, novčiće i prepreke koristi se *Unity Asset Store* koji nudi mogućnost kupnje ili besplatnog preuzimanja gotovih modela i na taj način smanjuje vrijeme izrade same igre. Osim službene trgovine okruženja *Unity,* koriste se i druge stranice preko kojih se besplatno mogu preuzeti razni zvukovi i modeli potrebni za izradu igre.

Prvi dio rada objašnjava način postavljanja same igre, promjenu rezolucije i dodavanje ceste kao pozadine. Pojašnjeno je kako funkcionira svjetlost u *Unity* okruženju i dodavanje glavnog i protivničkog automobila sa potrebnim komponentama koje omogućavaju sudare. Dodaju se novčići pomoću kojih se ostvaruju bodovi i implementiraju se zvukovi na različite objekte i scene. Objašnjena je funkcija prelaska na iduću razinu i način dodavanja i izrade svake sljedeće razine. Na kraju se dodaje završna scena koja se učitava nakon prelaska svih razina.

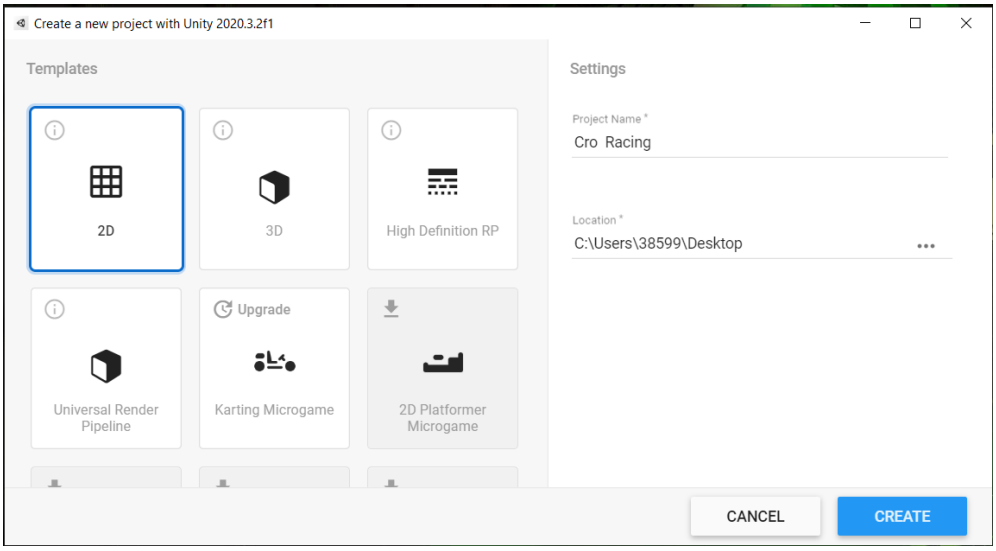
2. IGRAČ I OKOLINA

2.1. Postavljanje projekta i scene

Prije izrade same igre, postavlja se projekt u programu *Unity Hub*. Projekt će se zvati „Cro Racing“, a lokacija na koju će se spremiti može biti po vlastitom izboru. U ovom slučaju to će biti radna površina. Premda je tip igre dvodimenzionalan, odabire se 2D predložak (engl. *template*).

Koraci:

1. Otvara se *Unity Hub*
2. U gornjem lijevom kutu odabere se *Project*
3. Odabere se *New project* i otvara se novi prozor
4. Pod *Project name* upiše se „*Cro Racing*“
5. Lokacija spremanja postavi se na radnu površinu , a predložak na 2D



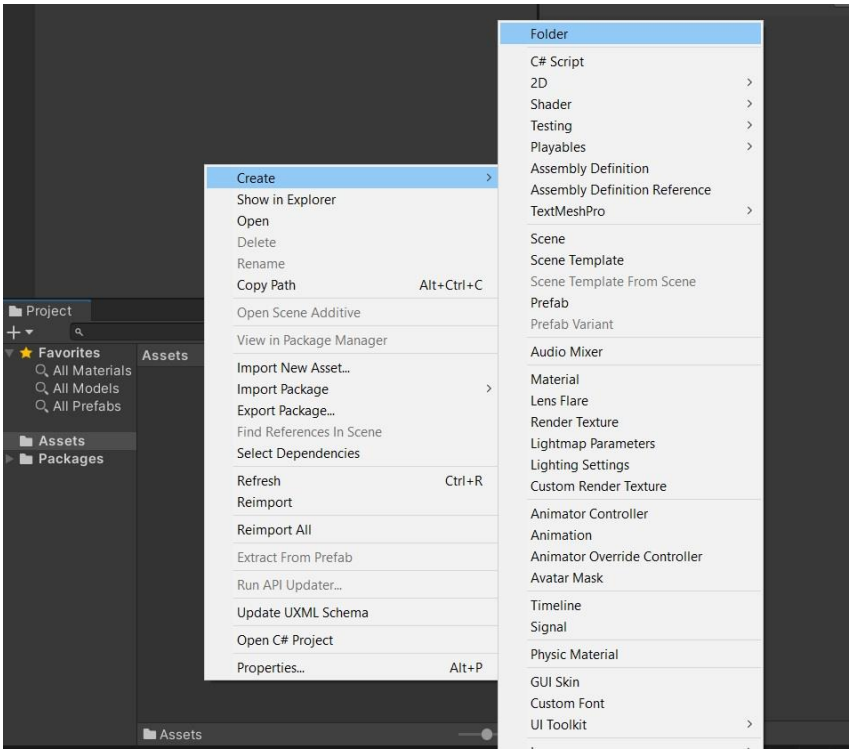
Slika 1. Izrada novog projekta

Izvor: Autor

Nakon postavljanja projekta potrebno je spremiti scenu. Međutim, prije spremanja scene poželjno je prilagoditi njen izgled, to jest, promijeniti raspored (engl. *layout*). Ovisno o rezoluciji i tipu igre, zbog bolje preglednosti i lakšeg snalaženja raspored se postavlja na sljedeći način:

1. U gornjem desnom kutu, pored ikone *Layers* iz *default* rasporeda promijeni se u „2 by 3“
2. Postavi se prozor *Scene* pored *Game* prozora
3. Spremi se izgled trenutnog rasporeda i imenuje se u „*racing*“

Budući da će se igra sastojati od više razina i scena, zbog lakšeg snalaženja potrebno je napraviti mapu pod nazivom „*Scenes*“ u koju će se spremati sve scene. Mapa se radi na način da se u mapi „*Assets*“ koja se nalazi u prozoru Project pritisne desni klik i u padajućem se izborniku odabire opcija *Create -> Folder*.



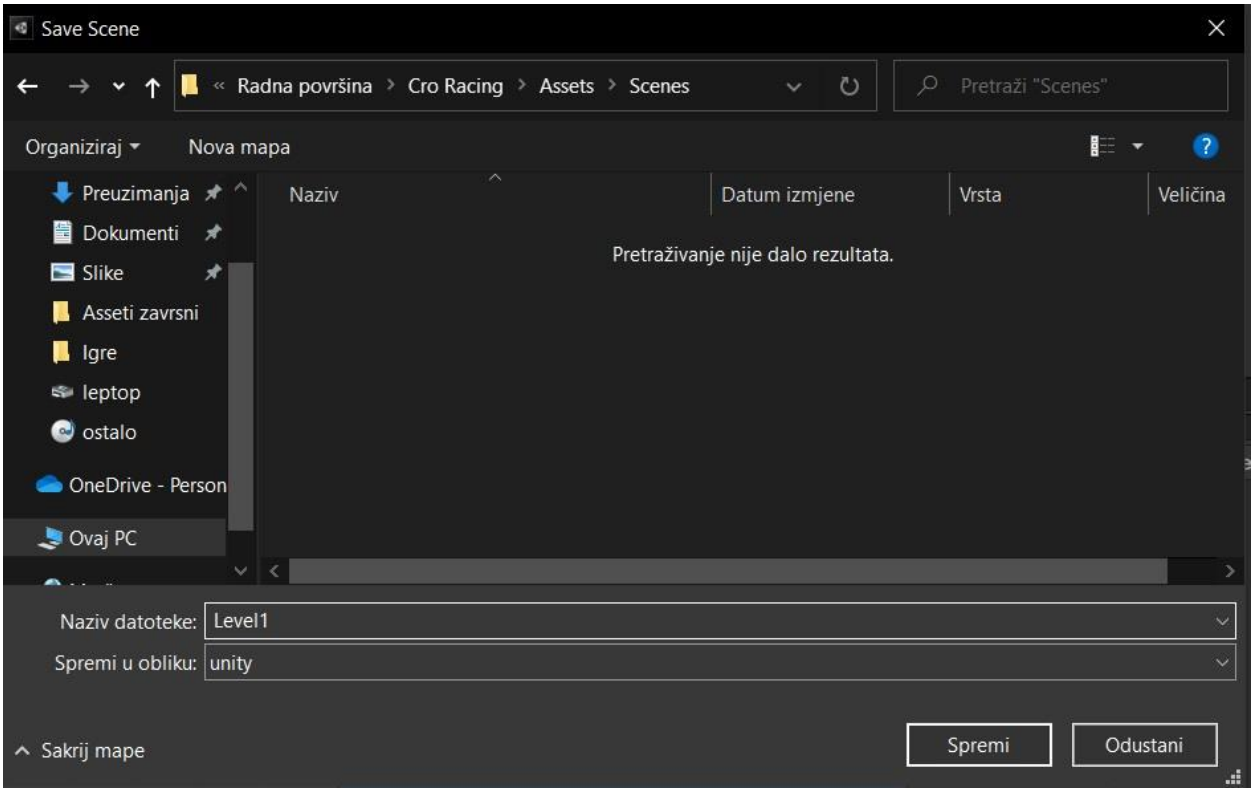
Slika 2. Izrada mape "Scenes"

Izvor: Autor

Dvostrukim klikom na ikonu mape označi se njeno ime i preimenuje se u „*Scenes*“. Osim mape *Scenes* potrebna je mapa u koju će se spremati skripte, odnosno C# datoteke kreirane u programu *Visual Studio*. Također je potrebna i mapa u koju će se spremati slike koje će se koristiti. Mapa za skripte zvat će se „*Scripts*“, a za slike „*Sprites*“. Nakon toga sprema se trenutna scena.

Koraci:

1. Otvara se izbornik *File* u gornjem lijevom kutu
2. Odabire se *Save Scenes as.*.
3. Scena se sprema pod nazivom „Level 1“ i putanja spremanja postavlja se na novoizrađenu mapu *Scenes*.



Slika 3. Spremanje scene

Izvor: Autor

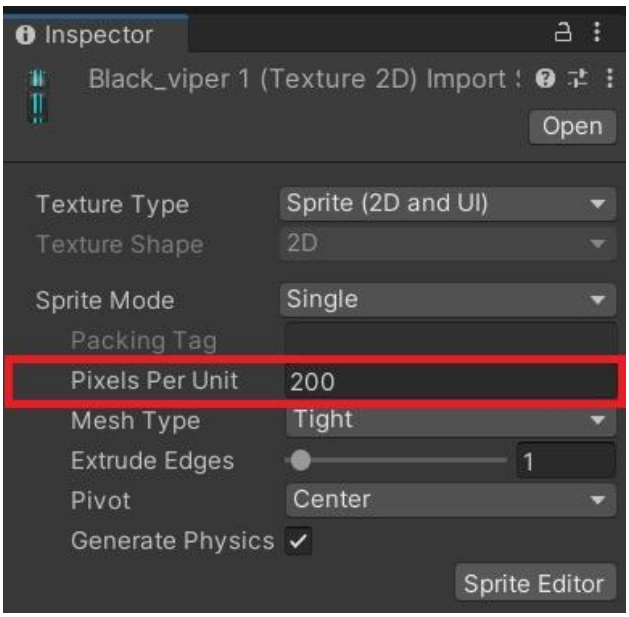
2.2. Dodavanje glavnog objekta

Prvi objekt koji se postavlja na scenu je ujedno glavni objekt - automobil kojeg mi kontroliramo. Postavlja se na sljedeći način.

Koraci:

1. U mapi *Sprites* pritisne se desni klik i nakon toga „Import new asset“
2. Odabire se automobil koji će se koristiti i pritisne gumb „Import“
3. Iz mape *Sprites* naš automobil se lijevim klikom miša povlači na scenu

Kako bi se najjednostavnije promijenila veličina (dužina i širina) automobila, a da se pritom ne uništi njegova kvaliteta, to se napravi u prozoru kontroler (engl. *Inspector*) koji se otvori nakon klika na objekt. Broj piksela po jedinici postavi se na 200.



Slika 4. Promjena rezolucije automobila

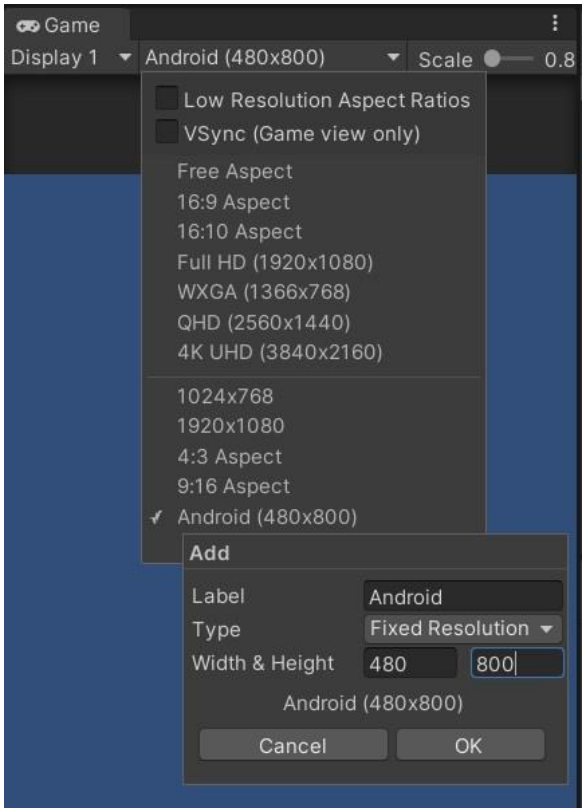
Izvor: Autor

2.3.Postavljanje rezolucije i pozadine

S obzirom na to da se glavni objekt, automobil, ne može kretati po zraku niti po praznoj površini, potrebno je dodati podlogu po kojoj će se kretati. To će biti cesta dizajnirana u programu *Adobe Photoshop 2020*. Naravno, prije početka dizajniranja potrebno je odlučiti koje će rezolucije biti igra, pa tako i sama cesta. Igra će biti rezolucije 480 × 800 i na sljedeći način postavlja se u programu Unity.

Koraci:

1. U *Game* prozoru odabire se kartica *Free Aspect*
2. Klikom na znak „+“ otvara se prozor u kojem se postavljaju dimenzije i ime rezolucije
3. Oznaka (engl. *label*) se imenuje u „Android“ a vrijednosti dužine i širine postavljaju se na 480 i 800



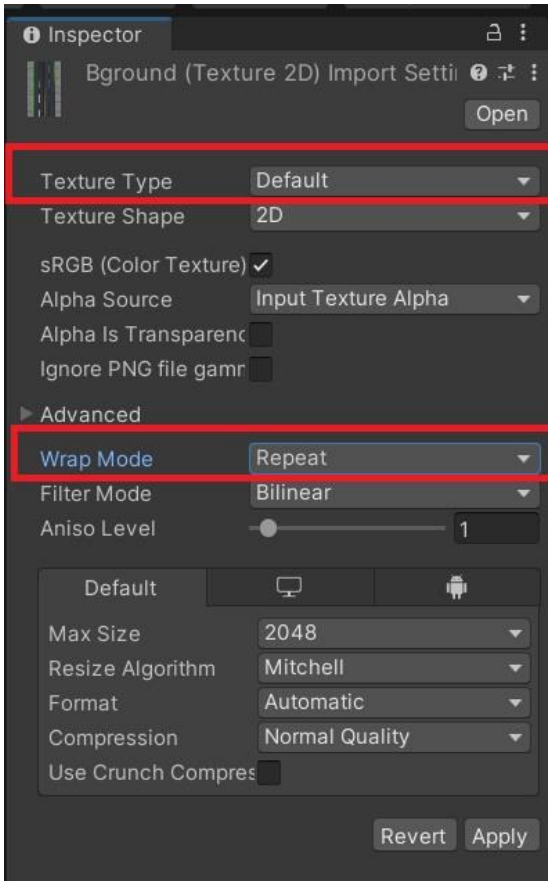
Slika 5. Postavljanje rezolucije

Izvor: Autor

Slika ceste koja se prethodno kreirala umeće se u mapu *Sprites* na isti način kao i automobil. Nakon što se u *Unity* umetne neka slika, ona se automatski generira kao s*prite*. To je vrsta grafike koja se može pomicati na zaslonu i predstavlja se kao jedna cjelina (engl. *entity*). No cesta prvenstveno predstavlja podlogu po kojoj se giba automobil i bitno je da se ne pomiče nego zapravo ponavlja iznova kako bi se dobio vizualni dojam brzog kretanja automobila. Stoga je bitno da se unaprijed promijene određena svojstva ceste [3].

Koraci:

1. U mapi *Sprites* označuje se naša pozadina i u *Inspectoru* se vrsta teksture (engl. *texture type)* postavlja na „Default“, a „Wrap Mode“ na „Repeat“
2. Pritisne se gumb *Apply* na dnu Inspectora da bi se spremile promjene



Slika 6. Promjena *Texture* i *Wrap* komponente ceste

Izvor: Autor

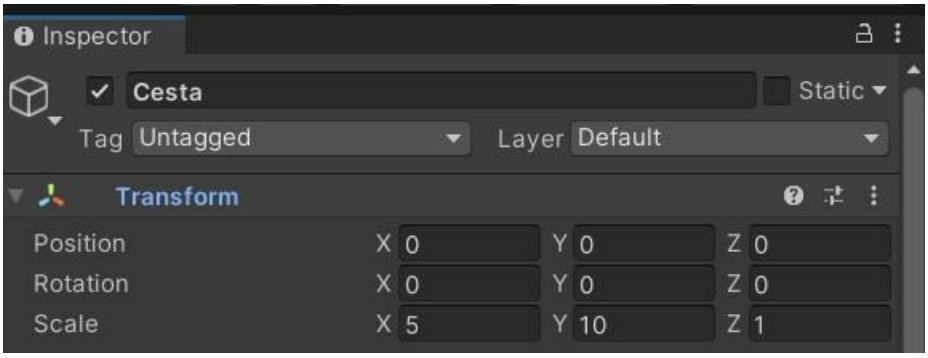
Nakon što se izmjene svojstva ceste, potrebno je postaviti cestu na scenu. Za to je prije svega potreban novi *Game Object*. Odabrat će se standardni Unity-ev 3D objekt naziva kvadrat (engl. *quad*).

Dodaje se na scenu i pravilno postavlja u sredinu scene, a njegove dimenzije prilagođavaju se rezoluciji same scene. To će se najlakše napraviti u *inspector* prozoru koji se otvara u desnom kutu nakon što se u mapi ili na sceni označi željeni objekt. Ukoliko mu zadani položaj nije u centru scene potrebno ga je resetirati, odnosno vrijednosti koordinata x,y i z postaviti na 0. Dužina i širina kvadrata može se postaviti ručno pomoću alata *resize* ili preciznije pomoću *transform scale* alata unutar *inspector* prozora. Vrijednosti koordinate x postavljaju se na 5, a koordinate y na 10. Koordinatu z nije potrebno mijenjati zbog toga što ona označava dubinu, a s obzirom da je igra koja se stvara 2D, dubina nije potrebna.

Nakon što se sve navedeno postavi i prilagodi, potrebno je jednostavno povući model ceste iz mape *Assets* na scenu. S obzirom na to da se cesti prethodno postavio zadani tip teksture, veličina se automatski prilagođava objektu kvadrata, odnosno sceni. Na kraju, radi lakše preglednosti i snalaženja u *hierarchy* prozoru preimenuje se *Quad* u „Cesta“.

Koraci:

1. U gornjem lijevom kutu klikom na karticu *GameObject*, otvara se padajući izbornik
2. Odabire se *3D Object -> Quad*
3. Preimenuje se *Quad* u „Cesta“
4. Označuje se cesta kako bi se otvorio *inspector*
5. Klikom na tri točke kod *transform* komponente otvara se padajući izbornik i pritiskom na *reset* centrira se *quad*
6. Nakon što se resetiranjem vrijednosti svih koordinata postave na 0, *position* i *rotation* koordinate nije potrebno mijenjati, dok se vrijednost *scale* koordinate x postavlja na 5, a y koordinate na 10.



Slika 7. Transform komponenta ceste

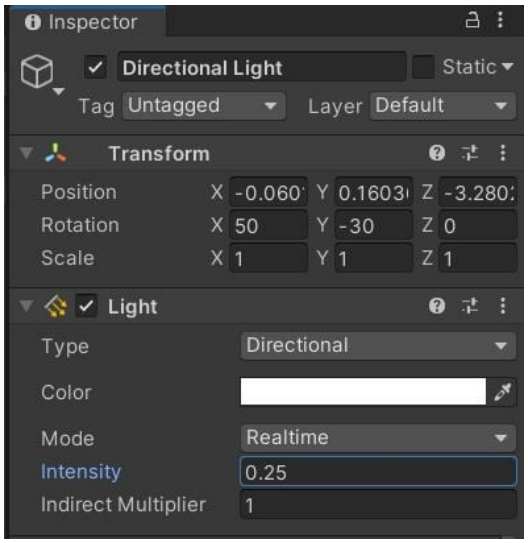
Izvor: Autor

2.4. Osvjetljenje scene

Primjećuje se da je scena mračna i da joj je potrebno osvjetljenje. Taj problem se riješi dodavanjem objekta koji se zove *Directional Light* i postavljanjem intenziteta svijetlosti po želji.

Koraci:

1. U gornjem lijevom kutu otvara se *GameObject -> Light -> Directional* Light
2. Postavlja se intenzitet na 0,25 u *Inspector* prozoru

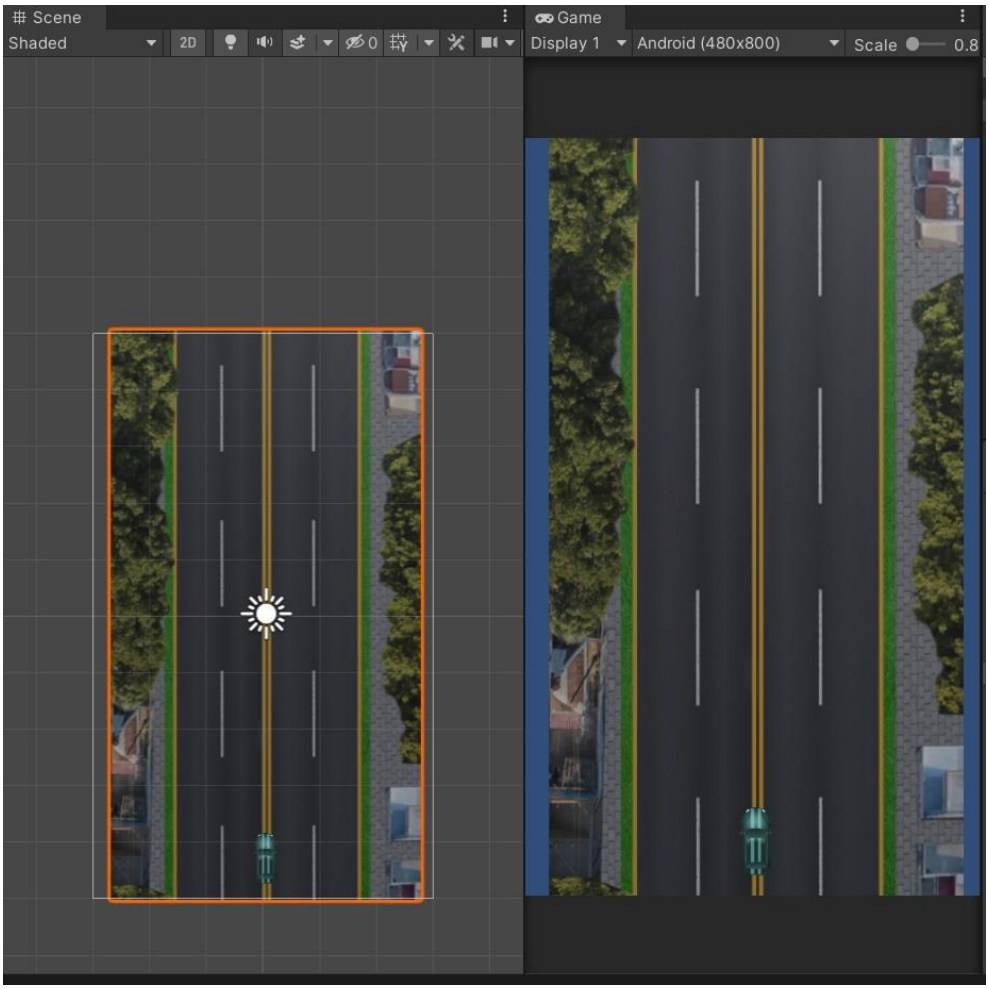


Slika 8. Inspector objekta Directional Light

Izvor: Autor

2.5. Pomicanje automobila

Nakon što se na scenu postavi glavni objekt auto i pozadina, odnosno cesta, vrijeme je da se pomoću programa *Visual Studio 2019* i programskog jezika C# omogući kretanje automobila.



Slika 9. Izgled scene i igre nakon umetanja automobila, ceste i svijetlosti

Izvor: Autor

Pomicanjem ceste unatrag određenom brzinom, dobit će se vizualni dojam kretanja automobila prema naprijed. Prije nego što se kreira C# datoteka u *Unity*-u, poželjno je izraditi mapu u koju će

se stavljati sve C# datoteke koje će se koristiti. Stoga, zbog lakšeg snalaženja u mapi *Assets*, izrađuje se nova mapa imena *Scripts*.

Postoji više načina kako C# datoteka stavlja u *Unity* i pridružuje određenom objektu. Otvara se mapa *Scripts* i pritiskom na desni klik odabire se *Create* i zatim *C# Script*. Skripta će se zvati GibanjeCeste.cs i pridružit će se cesti. Njena glavna i jedina zadaća je omogućiti teksturi ceste da se vertikalno giba ovisno o brzini i vremenu. Na početku je potrebno deklarirati varijable koje ćemo koristiti. Varijabla *speed* je javna, što znači da će se njena vrijednost moći određivati preko Inspectora i određivat će brzinu gibanja ceste, a pomoću varijable *offset* omogućuje se gibanje teksture objekta ceste.

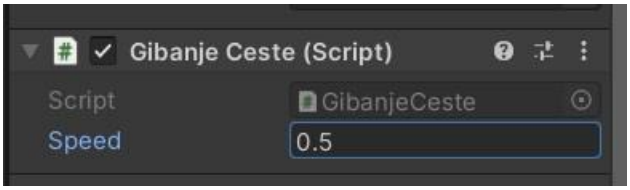
Nakon deklariranja varijabli potrebno je unutar *Update* metode deklarirati varijablu *offset*. Iz fizike nam je poznato da je prijeđeni put zapravo omjer brzine i vremena, dakle što se većom brzinom gibamo, to nam je manje vremena potrebno za prelazak nekog puta. Tu formulu potrebno je pravilno uvrstiti u kod za gibanje koordinatom y. S obzirom na to da će se cesta gibati samo vertikalno, vrijednost koordinate x postavit ćemo na 0. Zatim se *offset* varijabla pridružuje *renderer* komponenti koja omogućuje teksturi ceste da se ponavlja iznova kako prolazi vrijeme i na taj način se dobiva neprekidno gibanje ceste.

```
public float speed;
Vector2 offset;

void Update()
    {
        offset = new Vector2(0, Time.time * speed);
        GetComponent<Renderer>().material.mainTextureOffset =
offset;

    }
```

Kod 1. Omogućavanje vertikalnog gibanja ceste

Izvor: Autor

Nakon što je kod ispravno napisan, potrebno je pridružiti C# datoteku objektu ceste. Označuje se cesta i jednostavno se iz mape *Scripts* povuče datoteka unutar *inspector* prozora objekta ceste. S obzirom na to da će prva razina biti malo lakša, vrijednost brzine postavlja se na 0,5.



Slika 10. Određivanje brzine gibanja ceste

Izvor: Autor

Kad se igra pokrene, vidljivo je kako se automobil po cesti giba pravocrtno. Idući korak je omogućiti gibanje tog automobila lijevo i desno. Za to nam je potrebna C# skripta koju ćemo nazvati *CarController*.

Nakon što se otvori skripta u programu *Visual Studio*, najprije će se deklarirati varijabla koja će određivati brzinu automobila, vektor koji će predstavljati trenutnu poziciju automobila i vrijednost pozicije kad se automobil pomakne krajnje lijevo na sceni.

```
public float carSpeed;

float maxPos = 1.6f;

Vector3 position;
```

Kod 2. Deklariranje varijabli u skripti CarController.cs

Izvor: Autor

Nakon toga se unutar Start metode odredi trenutna pozicija automobila, a u *Update* metodi povećava vrijednost pozicije automobila na X osi ovisno o tome koliko dugo je pritisnuta tipka za lijevo ili desno. Vrijednost ulazne jedinice množimo s brzinom auta i s vremenom tako da se uvijek

giba istom brzinom i taj umnožak povećavamo s trenutnom pozicijom automobila na x osi. Na taj način naš se automobil može pomicati lijevo i desno.

Važno je odrediti do koje granice se automobil može pomicati lijevo i desno, a da pritom ne ode van scene. Pomoću naredbe *Mathf.Clamp* najjednostavnije će se postaviti granice, točnije ograničiti vrijednost pozicije x.

```
void Start()
    {
        position = transform.position;
    }

void Update(){
        position.x += Input.GetAxis("Horizontal") * carSpeed * Time.deltaTime;
        position.x = Mathf.Clamp(position.x, -1.3f, maxPos);
        transform.position = position;
}
```

Kod 3. Program kontroliranja kretnji automobila

Izvor: Autor

2.6. Protivnički automobili

Kako bi naša igra bila zahtjevnija i zanimljivija, dodat će se protivnički automobili koje treba izbjegavati i prestići. Ukoliko se naš automobil sudari s protivničkim, dolazi do eksplozije i razina

se mora ponoviti. Najprije se u mapu *Sprites* umetnu ranije odabrani modeli automobila koji se nalaze na linku: https://unluckystudio.com/game-art-giveaway-7-top-down-vehicles-sprites-pack/.

Slika 11. Modeli protivničkih automobila u mapi *sprites*

Izvor: Autor

Na isti način kao što smo promijenili veličinu našeg automobila, to se učini i s protivničkim. U *inspectoru* broj piksela po jedinici se postavi na 200, odabere se model automobila i povuče iz mape *Sprites* na scenu. Nakon toga promijeni se ime objekta u „Protivnicki1“.

U mapi *Scripts* kreira se nova C# skripta i imenuje se „ProtivnickiIstiSmjer“. Otvara se skripta u programu *Visual Studio* kako bi se odredio način kretanja protivničkog automobila.

Najprije se deklarira javna varijabla *speed* i njena vrijednost se za početak postavi na 5f. Nakon toga potrebno je pomoću *transform.translate* funkcije omogućiti protivniku da se vertikalno giba u smjeru kojim ide naš automobil. Za razliku od našeg automobila koji se giba X koordinatom, protivnički će se gibati Y koordinatom određenom brzinom, a kod izgleda ovako.

```
public float speed = 5f;

void Update()

    {

        transform.Translate(new Vector3(0, -1, 0) * speed * Time.deltaTime);

    }
```

Kod 4. gibanje protivničkog automobila po Y koordinati

Izvor: Autor

Iduće što je potrebno dodati igri je nasumično stvaranje, odnosno pojavljivanje protivničkih automobila na sceni. Najprije se treba odrediti pozicija na sceni u kojoj će se protivnici pojavljivati. Za to će biti potreban novi *GameObject* koji će se nazvati „*SpawnPosition*“, u *inspector* prozoru se resetira njegova pozicija i nakon toga postavi iznad gornjeg dijela scene.

Koraci:

1. Dodaje se novi *GameObject* u gornjem lijevom kutu (*Create Empty-> GameObject*)
2. Preimenuje se u „*SpawnPosition*“
3. U *inspector* prozoru desnim klikom na *transform* otvara se izbornik i odabire *Reset*
4. Odredi se pozicija u kojoj će se stvarati protivnički automobili

Slika 12. Položaj *spawn* objekta

Izvor: Autor

Zatim će biti potrebna skripta koja će nakon početka igre početi sa nasumičnim stvaranjem automobila. Skripta se imenuje u „*CarSpawner*“, pridoda objektu *CarSpawnPosition* i otvara u *Visual Studiu*. Najprije će se dodati javni objekt *car* i u *Start* metodi omogućuje njegovo pojavljivanje. Zapravo se autu daju iste karakteristike kao *spawneru*.

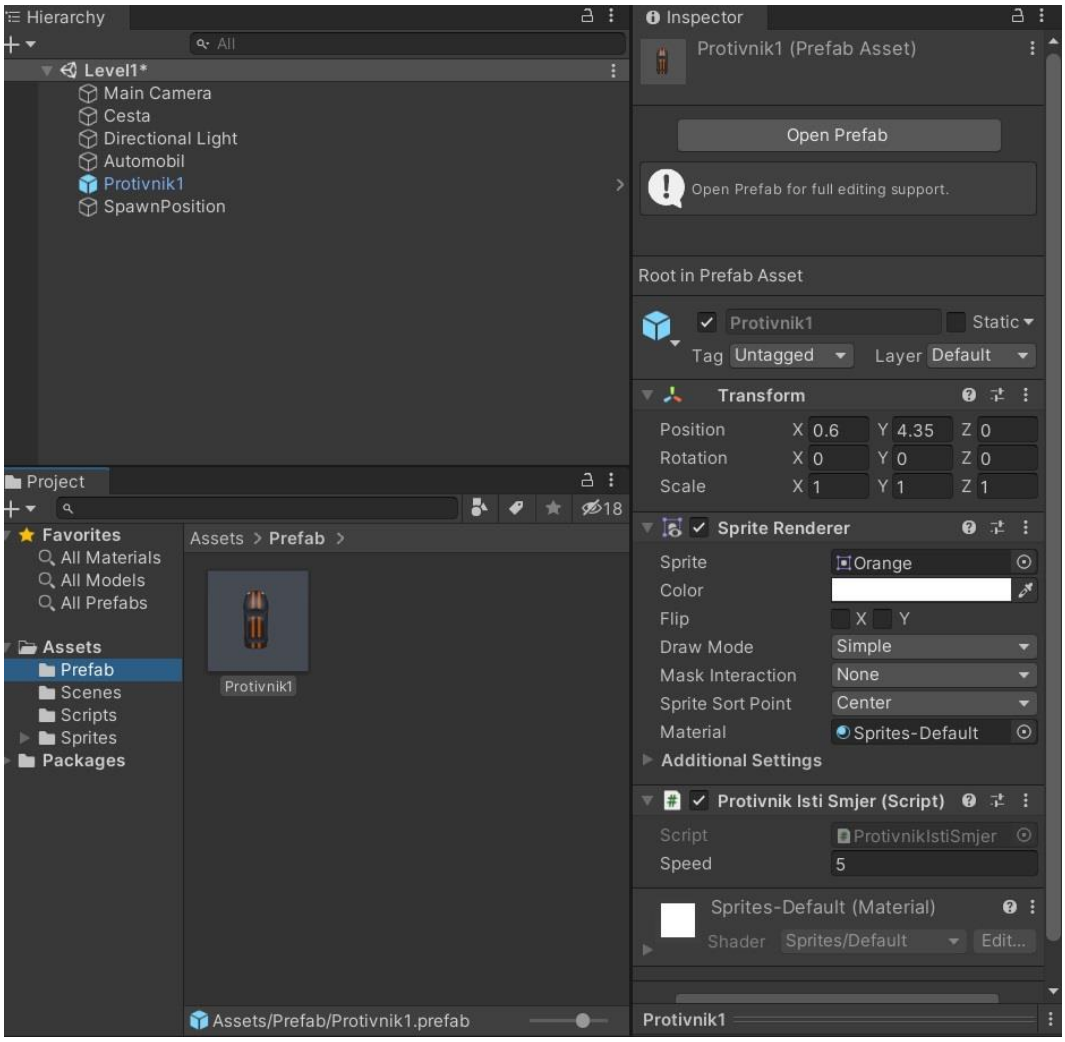
```
public gameObject car;


void Start() {
Instantiate (car, transform.position, transform.rotation);
}
```

Kod 5. Stvaranje objekta Car

Izvor: Autor

Nakon toga je u *Inspectoru* prazan *GameObject* u koji je potrebno dodati auto. U ovom dijelu potrebno je koristiti komponente naziva *Prefab*. To je posebna komponenta koja omogućuje spremanje potpuno konfiguriranih i dizajniranih *GameObjecta* u projektu za ponovnu upotrebu. Na taj način ih možemo dijeliti između scena pa čak i koristiti na drugim projektima bez ponovnog konfiguriranja.

S obzirom na to da ćemo imati više protivničkih automobila, oni će biti *prefabi*. Najprije se u mapi *Assets* napravi nova mapa i imenuje se u „*Prefabs*“. Otvara se mapa i jednostavno se iz scene povuče protivnički auto u mapu. Primjećujemo da je naziv objekta promijenio boju iz bijele u plavu, a to je pokazatelj da je pretvoren u *prefab*. Zatim izbrišemo objekt Protivnicki1 iz scene zbog toga što ga sad imamo u mapi *Prefabs*.



Slika 13. Protivnik kao *prefab*

Izvor: Autor

Nakon toga se na sceni označi objekt *CarSpawnPosition* i u *inspector* umetne Protivnicki1 unutar praznog objekta *Car*.

Potrebno je odrediti s kojih pozicija će se stvarati protivnički automobili. U skripti *CarSpawner* odrede se minimalne i maksimalne vrijednosti unutar kojih će se stvarati protivnički automobili. U našem slučaju to će biti -1,3 i 1,6. Zatim je potrebna nova varijabla koja će spremati nasumične pozicije i ona će biti unutar Vektora3. S obzirom da se kreće samo po X koordinati, postavimo da su te vrijednosti nasumični brojevi između -1,3 i 1,6b, a za koordinate Y i Z stavimo trenutnu poziciju.

```
void Start() {

Vector3 carPos = new Vector3(Random.Range(-1.3, 1.6),
transform.position.y, transform.position.z);
```

Primjećuje se da se auto pojavi jednom na početku igre. Potrebno je da se pojavljuje iznova, svakih nekoliko sekundi, stoga se izrađuje brojač (engl. *timer*) i kod unutar metode Start stavlja u metodu *Update*. Deklarira se varijabla „*timer*“ tipa *float* i javna varijabla *delayTimer* koja će biti vrijeme potrebno za pojavu novog protivničkog automobila. Njena vrijednost postavi se na 1f.

Zatim se unutar *Update* metode smanjuje vrijednost *timera*, a unutar *Start* metode se vrijednost *delayTimer* postavlja unutar varijable *timer*. Nakon svake sekunde vrijednost *timera* pada na 0 i u tom trenu želimo da se protivnički automobil pojavi. To će se ostvariti pomoću *if* naredbe.

```
public float delayTimer = 1f;

  float timer;


void Start()

    {

        timer = delayTimer;

    }


 void Update()

    {
```

```
        timer -= Time.deltaTime;


      if(timer <= 0)
        {
            Vector3 carPos = new Vector3(Random.Range(-1.3f,
1.6f), transform.position.y, transform.position.z);


      Instantiate(cars[carNo], carPos, transform.rotation);


      timer= delayTimer; //vrijednost se ponovno postavlja na 1
        }
     }
```

Kod 6. Stvaranje automobila nakon svake sekunde

Izvor: Autor

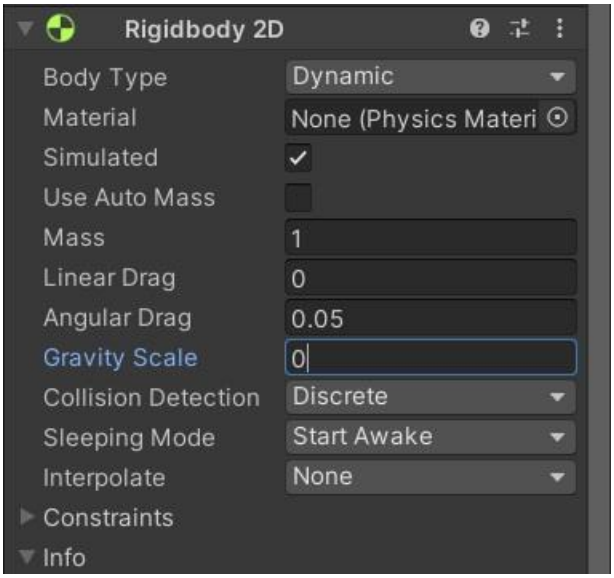
2.7. Sudari

Nakon što se omogući pojavljivanje više protivničkih automobila na cesti, potrebno je aktivirati mogućnost sudara. Odabire se glavni automobil i pridružuju mu se komponente *Rigidbody 2D* i *Box Collider 2D*. Komponenta *Rigidbody* služi za jednostavno dodjeljivanje standardnih fizičkih svojstva objektu kao što su masa i gravitacija, dok komponenta *collider* daje objektu svojstvo krutosti, odnosno omogućuje da se objekti sudare [4].

Koraci:

1. Označi se objekt automobil

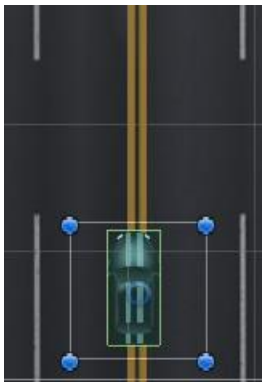
2. Unutar *inspectora* klik na gumb *Add Component -> Physic2D -> Rigidbody 2D*
3. *GravityScale* unutar *Rigidbody* komponente postavlja se na 0



Slika 14. *Rigidbody 2D* komponenta za Automobil

Izvor: Autor

4. *Add Component -> Physic2D -> Box Collider 2D*
5. Unutar *Box Collider* komponente pritisne se na *Edit Collider* i prilagodi veličina



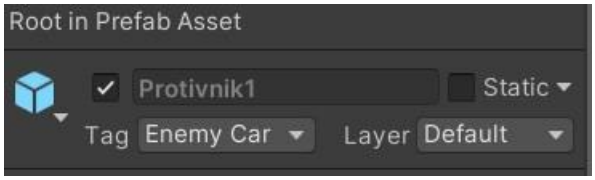
Slika 15. *Box Collider* objekta Automobil

Izvor: Autor

Zatim se objektu protivničkog automobila na isti način doda *BoxCollider 2D* kako bi sudar bio moguć. Potrebno je dodati oznaku (engl. *tag*) protivničkom automobilu.

Koraci:

1. Označi se objekt Protivnik1
2. Unutar *inspectora* klikom na gumb *Untagged* otvara se padajući izbornik i odabire *Add tag*
3. Otvara se prozor *Tags and Layers* i klikom na znak plus dodaje se nova oznaka koja se imenuje u „*Enemy Car*“
4. Ponovo se označuje auto i oznaka se postavi na *Enemy Car*
5. Kako bi se spremile promjene na svaki budući *prefab*, odabire se gumb *Apply*



Slika 16. Oznaka *Enemy Car* na objektu Protivnik1

Izvor: Autor

Skripta *carController* za sad samo kontrolira kretanje automobila, no njen zadatak će između ostalog biti i da upravlja sudarima. Koristit će se posebna funkcija programa *Unity*, *OnCollisionEnter2D* koja će se pozvati svaki put kad dođe do sudaranja. Unutar te funkcije uništit će se glavni automobil kad dođe u sudar s protivničkim automobilom.

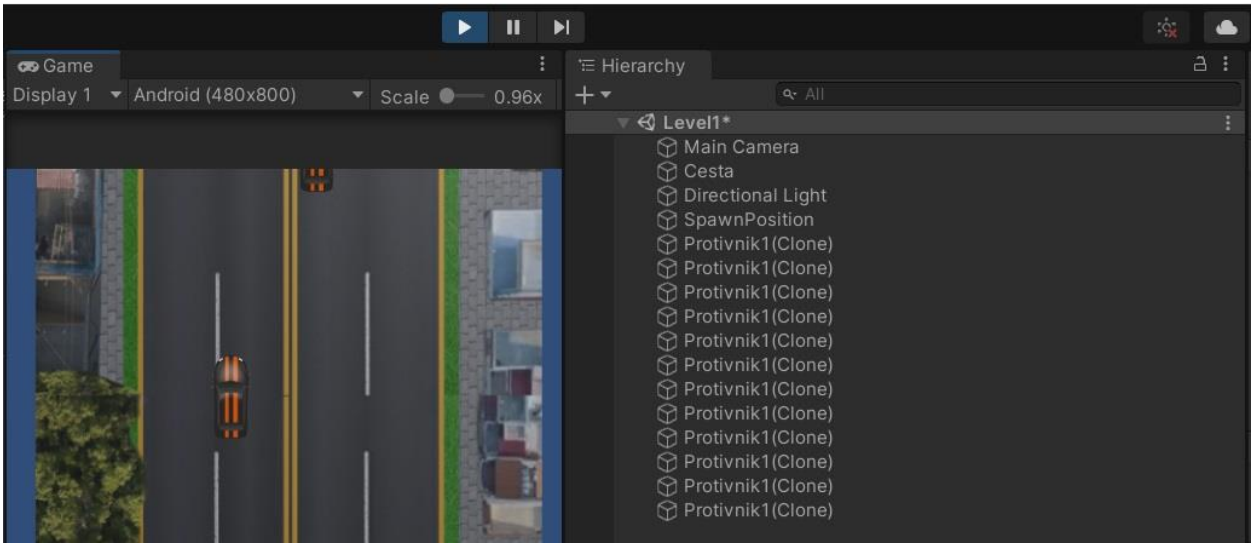
```
void OnCollisionEntter2D{

if(col.gameObject.tag == "Enemy Car")

        {


            Destroy(gameObject);
```

```
        }
}
```

Kod 7. Uništenje glavnog objekta ukoliko se sudari sa protivnikom

Izvor: Autor

Primjećuje se da se protivnički automobili ne unište nakon što izađu iz scene, nego se njihova imena konstantno stvaraju u *hierarchy* prozoru te na taj način usporavaju igru.



Slika 17. Usporavanje igre

Izvor: Autor

Potrebno je uništiti objekt Protivnik1 svaki put kada izađe iz scene i za to će biti potreban novi objekt kojem se daje ime „*Destroyer*“. Postavlja se na željeno mjesto ispod scene koje će biti predviđeno za uništenje protivničkih automobila.

Slika 18. Položaj objekta *Destroyer*

Izvor: Autor

Objektu *Destroyer* dodaje se komponenta *Rigidbody 2D* i *BoxCollider*, a veličina se prilagodi samom objektu. U mapu *Scripts* dodaje se nova skripta po imenu „DestroyerProtivnik“ i pridružuje objektu *Destroyer*. Otvara se skripta u programu *Visual Studio* i ponovno se poziva funkcija *OnCollisionEnter2D* i unutar nje se provjerava je li došlo do sudara i ukoliko je, uništava se protivnički automobil.

```
void OnCollisionEnter2D(Collision2D col)
    {
        if (col.gameObject.tag == "Enemy Car")
        {
            Destroy(col.gameObject);
        }
    }
```

Kod 8. Uništavanje protivničkog automobila nakon sudara s objektom *Destroyer*

Izvor: Autor

Primjećuje se da je na sceni samo jedan model protivničkog automobila i to čini igru monotonom. Dodat će se različiti modeli *prefaba* protivničkih automobila u našu igru i omogućiti da se stvaraju nasumično.

Koraci:

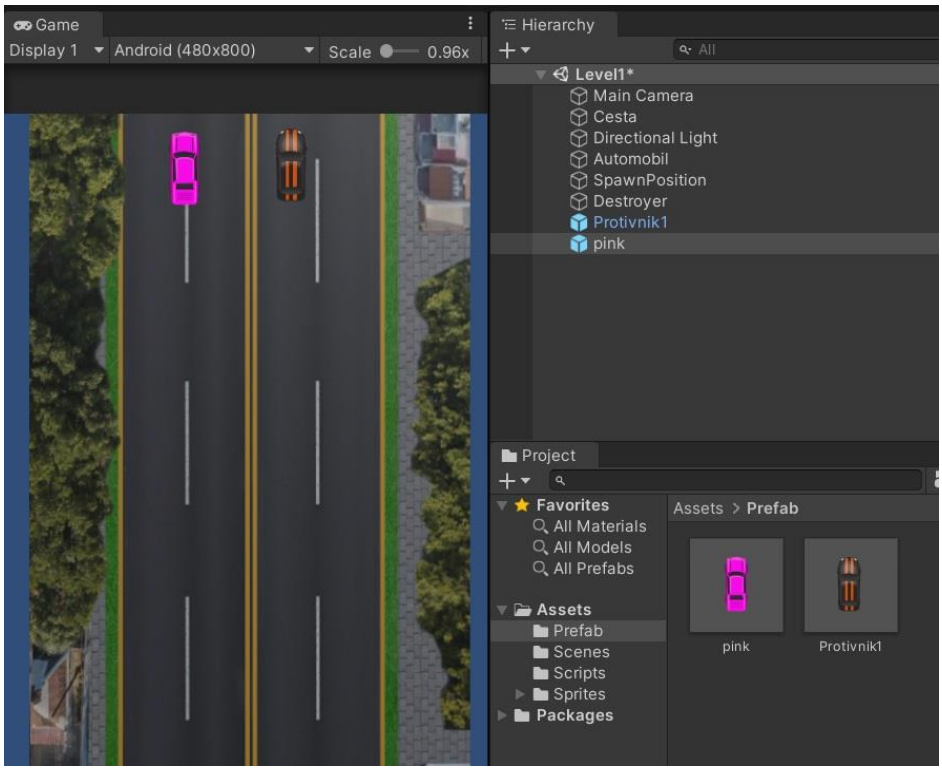
1. U *hierarchy* prozoru označi se objekt Protivnik1 i pritiskom na tipku CTRL + D, auto se duplicira
2. Označi se dupli cirani automobil
3. U *inspectoru* unutar *Sprite Renderer* komponente klikne se mala točka pored naziva *Sprite*
4. Otvara se mapa *Sprites* sa ranije umetnutim modelima automobila

Slika 19. Mapa *sprites* sa modelima automobila

Izvor: Autor

5. Dvostrukim klikom odabire se željeni model, u ovom slučaju rozi
6. U *inspector* prozoru, naziv objekta promijeni se u „*pink*“
7. Iz *hierarchy* prozora objekt se povuče u mapu *Prefab*

Nakon umetanja još jednog automobila, naša igra izgleda ovako.



Slika 8. Izgled igre nakon 2 protivnička automobila

Izvor: Autor

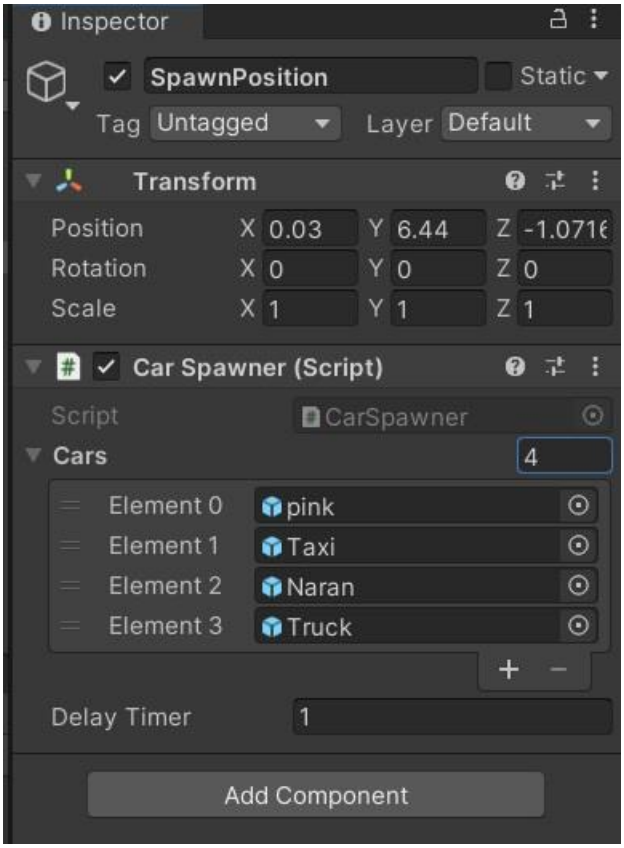
Na isti način kao što se dodao automobil roze boje, dodaju se i ostali automobili koji će se koristiti u našoj igri. Dodat će se svi modeli automobila bez obzira na kojoj razini će biti korišteni. Ključni korak je da se svaki od novih automobila ubaci u mapu *Prefab*, koja sad izgleda ovako.

Slika 9. Mapa *Prefab* sa svim modelima automobila

Izvor: Autor

Trenutno igra sadrži 11 različitih automobila, a na prvoj razini koristit će se njih 5. S obzirom na to da se svaki od njih duplicirao od objekta Protivnik1, imaju sva svojstva koja su njemu pridružena.

Potrebno je nasumično ubaciti izabrane modele automobila u našu igru. Za to će biti potrebna skripta *CarSpawner* koja se otvara u *Visual Studiu*. Primjećujemo da imamo deklariran javni objekt „car“ koji je potrebno pretvoriti u polje tako da se iza naziva *GameObject* dodaju uglate zagrade. Deklarira se nova varijabla *carNo* koja će odlučivati koji automobili će biti ubačeni u funkciju instanciranja. Nakon što se otvori *SpawnPosition* objekt u igri, primjećuje se da je unutar njega moguće odrediti veličinu polja *Cars*. S obzirom na to da ćemo za početak imati 4 različita automobila, upišemo broj 4 u polje *Size* i u svaki prazan element umeće se željeni *prefab* protivnika.



Slika 10. Protivnički automobili umetnuti u polje Cars

Izvor: Autor

Potrebno je nasumično odabrati jedan po jedan automobil iz polja, a to će se postići tako da se pomoću *random.range* komande varijabli *carNo* pridružuje određena nasumična vrijednost između 0 i 3. Ta vrijednost se pridružuje polju i automobil koji je pod tim elementom stvara se na sceni zbog čega nastaju promjene u skripti *CarSpawner*, najprije kod deklariranja varijabli, a zatim unutar *if* tvrdnje u *Update* metodi.

```
public GameObject[] cars;

int carNo;

public float delayTimer = 1f;

float timer;
```

```
 void Start()
    {
        timer = delayTimer;
    }

 void Update()
    {

        timer -= Time.deltaTime;
        (timer <= 0)
        {
            Vector3 carPos = new Vector3(Random.Range(-1.3f,
1.6f), transform.position.y, transform.position.z);
            carNo = Random.Range(0, 4);
            Instantiate(cars[carNo], carPos, transform.rotation);
            timer = delayTimer;
        }
```

Kod 9. Skripta za nasumično pojavljivanje različitih modela automobila

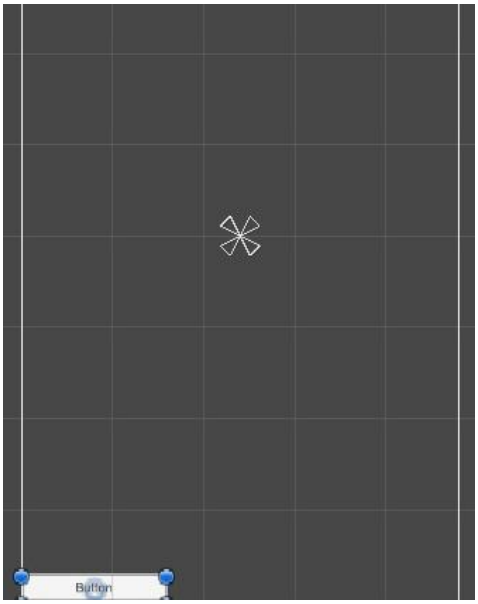
Izvor: Autor

2.8. Izrada gumba za pauzu

Kao i u svakoj, makar bila riječ o najjednostavnijoj igri, potreban je gumb za stanku (engl. *pause button*).

Koraci:

1. Desni klik u *hierarchy* prozoru -> *UI* -> *Button*
2. Označi se gumb
3. Pokazivač miša postavi se bilo gdje na sceni i pritisne tipka F kako bi se postavio fokus na gumb

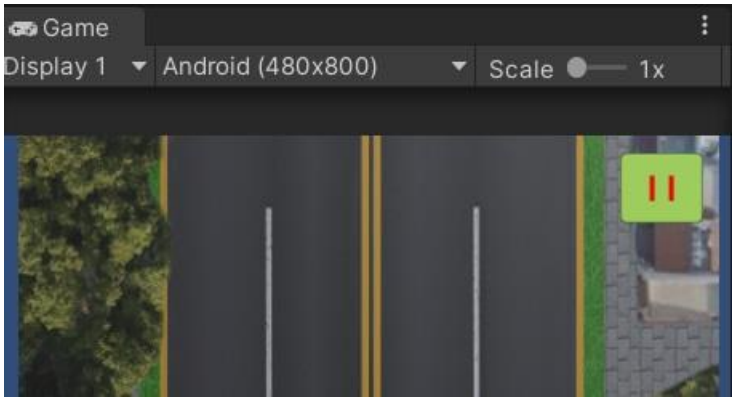


Slika 11. Promjena fokusa sa scene na UI elemente

Izvor: Autor

4. Odredi se pozicija gumba i veličina se prilagodi željenom obliku

5. Promjeni se tekst koji piše unutar gumba, koristeći dva velika tiskana slova I napravi se znak za pauzu
6. Unutar *inspector* prozora, odabere se boja gumba i boja teksta koji se nalazi unutar njega



Slika 12. Izgled i položaj gumba na sceni

Izvor: Autor

7. Novi gumb preimenuje se u *Pause Button*

S obzirom na to da ovo neće biti jedini gumb u našoj igri, izrađuje se skripta *uiManager* koja će kontrolirati i sve ostale UI elemente u našoj igri. Najprije se izrađuje novi prazan *GameObject* koji će se nazvati *uiManager* i njemu se pridružuje istoimena skripta. Otvara se skripta u programu *Visual Studio* i izrađuje javna funkcija *Pause* koja će kontrolirati naš gumb za pauzu. Svaki put kad se funkcija pozove, ona će pauzirati našu igru ukoliko je igra aktivna ili nastaviti igru ako je zaustavljena.

```
public void Pause()

    {

        if(Time.timeScale == 1)

        {

            Time.timeScale = 0;
```

```
        }
        else if (Time.timeScale == 0)
        {
            Time.timeScale = 1;



        }
```

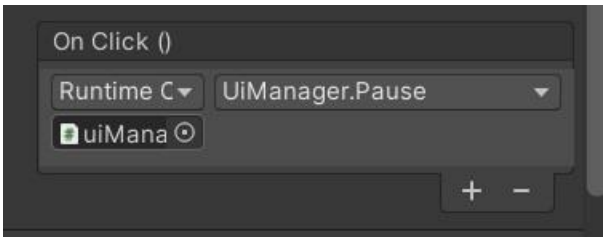
Kod 10. Deklariranje funkcije za pauziranje igre

Izvor: Autor

Potrebno je pridružiti skriptu i funkciju *Pause* novoizrađenom gumbu.

Koraci:

1. Označuje se *Pause Button*
2. Na dnu njegovog Inspectora vidimo *OnClick* događaje koji se aktiviraju kada se gumb pritisne. Pritiskom na znak plus postavlja se novi događaj.
3. Iz *hierarchy* prozora povuče se objekt kojem je pridružena skripta *uiManager*.
4. Klik na gumb *NoFunction ->Ui Manager -> Pause()*



Slika 13. Pridruživanje funkcije Pause gumbu

Izvor: Autor

3. POČETNA SCENA

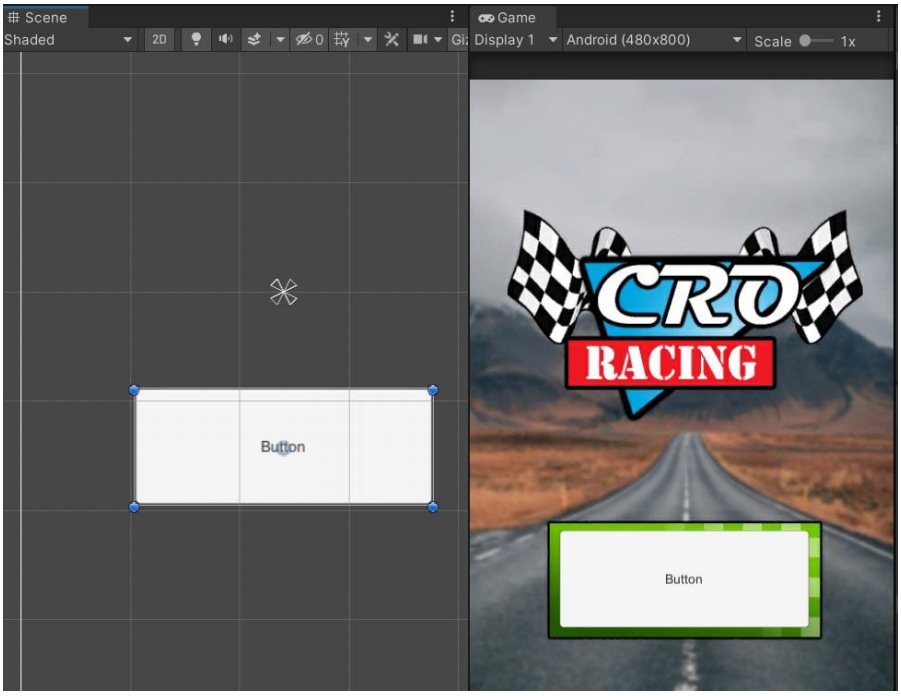
Kao i u svakoj igri važno je imati početnu scenu preko koje se zapravo ulazi u samu igru. Na našoj početnoj sceni bit će logo igre, ime igre, gumb za pokretanje i gumb koji otvara Panel s kratkim uputama vezanim uz igru.

Najprije se kreira nova scenu i sprema u mapu *Scenes* pod nazivom *Menu.* Scena se dodaje u prozor *Scenes in Build*“ koji se otvara u gornjem lijevom kutu pritiskom na tipku *File -> Build Settings*. U programu *Adobe Photoshop 2020* kreirala se slika početne scene, a tu sliku umetnemo u mapu *Sprites*.

Slika 14. *Game menu* slika u programu *Adobe Photoshop*

Izvor: Autor

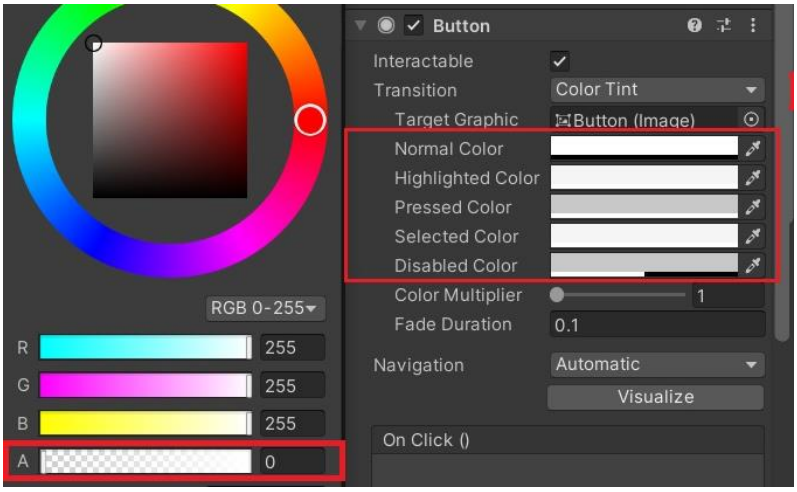
Slika se iz mape *Sprites* premjesti na našu scenu i veličina i rezolucija se prilagode *game* sceni. Premda se na slici već nalazi dizajniran gumb start, potrebno mu je pridodati određenu funkciju. Najprije se izrađuje novi gumb koji se postavi na mjesto zelenog okvira naše slike.



Slika 15. Izrada gumba za početak igre

Izvor: Autor

Cilj je da gumb bude nevidljiv, ali da i dalje izvršava svoju funkciju. Unutar *inspector* prozora, pritisne se na izbornik za promjenu boje gumba i vrijednost A postavlja se na 0.



Slika 16. Isključivanje vidljivosti gumba

Izvor: Autor

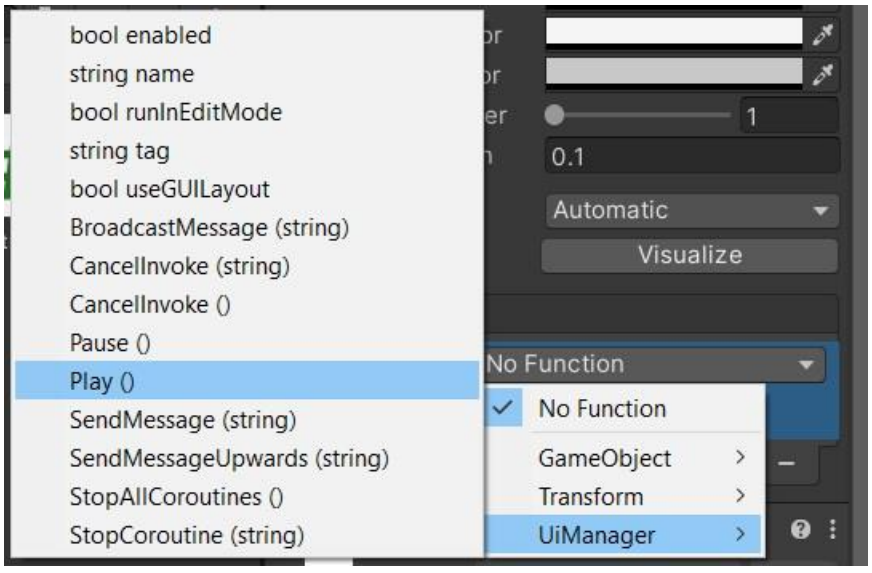
Potrebno je napraviti funkciju za pokretanje igre. Unutar skripte *uiManager.cs* dodaje se javna funkcija *Play* čiji je zadatak zapravo pokrenuti scenu koja predstavlja prvu razinu, u ovom slučaju to je scena Level1.

```
public void Play()
    {
        Application.LoadLevel("Level1");
    }
```

Kod 11. Funkcija *Play* za pokretanje prve razine

Izvor: Autor

Zatim se ponovo kreira novi *GameObject* imena *UiManager* i pridružuje mu se istoimena skripta u kojoj se prethodno kreirala funkcija *Play*. Označi se gumb i u *Inspector* prozoru kreira događaj kao i prethodno sa *Pause* gumbom. Ovog puta ne odabire se funkcija *Pause()*, nego *Play()*.



Slika 17. Aktiviranje funkcije *Play* na pritisak gumba

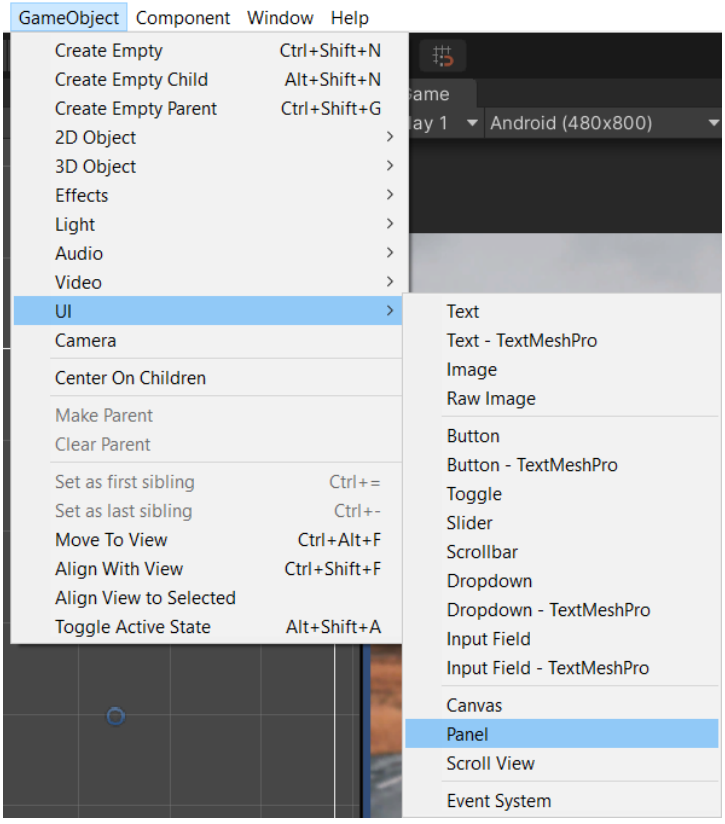
Izvor: Autor

3.1. Panel s uputama

Upute se mogu pročitati prije samog početka igre. Ukoliko se korisnik odluči da ih želi vidjeti i pritisne na gumb „Upute“ otvorit će se panel u kojem je kratko objašnjeno kako igra funkcionira i koji je cilj.

Za otvaranje i zatvaranje panela s uputama, potrebno je najprije dodati novi gumb. U gornjem lijevom kutu odabire se *GameObject* -> *UI* -> *Button* i imenuje se u „Button Upute“. Gumb se postavlja u gornji desni kut scene, odabire se crvena boja gumba i bijeli tekst fonta „*Cute Stitch*“.

Na isti način kao što se dodaje gumb i svaki drugi UI element, dodaje se i panel.



Slika 18. Dodavanje objekta panel

Izvor: Autor

Panel se postavlja na dnu scene i veličina se prilagodi sceni. Za boju panela sve RGB boje postave se na 0, a vrijednost A (engl. *alpha*) na 154. Desnim klikom na objekt Panel pridodaje mu se komponenta za tekst i unutar *inspector* prozora upisuje se tekst koji će biti prikazan u panelu nakon pritiska na gumb za upute. Ponovo se odabire font „*Cute Stitch*“, a veličina se postavi na 13. Ovako će izgledati scena nakon pritiska na gumb „Upute“.

Slika 19. Meni scena nakon otvaranja uputa

Izvor: Autor

Uklanja se kvačica pored imena objekta „Panel“ kako ne bi bio vidljiv na sceni. Zatim se u mapu *Scripts* dodaje skripta „*PanelOpen.cs*“ kojoj će biti zadatak da aktivira Panel nakon pritiska na gumb Upute.

Skripta se otvara u programu *Visual Studio* i najprije se deklarira varijabla objekta *Panel*, a nakon toga se izrađuje funkcija koja otvara *panel* ukoliko je on trenutno zatvoren ili ga zatvara ako je otvoren.

```
public GameObject Panel;


    public void OpenPanel()
    {
        if(Panel != null)
        {
            bool isActive = Panel.activeSelf;
            Panel.SetActive(!isActive);
        }
    }
```

Kod 12. Otvaranje i zatvaranje *panela*

Izvor: Autor

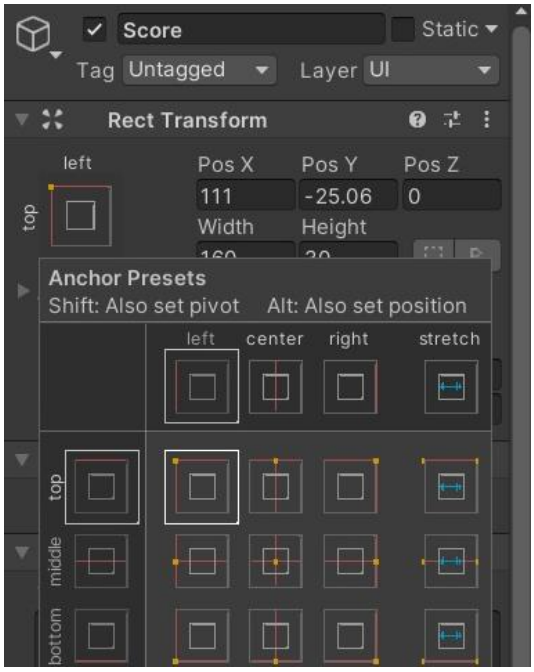
Zatim se skripta dodaje objektu „*Button* Upute“ i objekt *Panel* stavlja se unutar skripte. Unutar komponente *Button* postavlja se funkcija koja otvara *panel* klikom na gumb.

4. BODOVI

Cilj igre je sakupiti sto bodova koji će nas odvesti na iduću razinu i tako do samog kraja igre. Bodovi se povećavaju nakon što automobil pokupi novčić. Svaki novčić donosi deset bodova i nakon sakupljenog desetog novčića, prelazi se na sljedeću razinu. Najprije se na scenu dodaje tekst koji će prikazivati trenutni broj bodova.

Koraci:

1. U gornjem lijevom kutu odabere se izbornik *GameObject -> UI -> Text*
2. Objekt *Text* preimenuje se u *Score*
3. Pokazivač miša postavlja se na scenu i pritisne tipka F za fokus na tekst
4. Unutar Inspectora u polju *Text*, upišemo tekst koji želimo vidjeti na sceni, *Score*:
5. Font teksta postavi se na 26 i odabire se bijela boja
6. Klikom na kvadrat u lijevom kutu Inspectora, otvori se izbornik *Anchors*
7. Odabire se gornje lijevo (engl. *top left*) poravnanje. Time će naš tekst uvijek biti u gornjem lijevom kutu scene čak i ako dođe do promjene rezolucije



Slika 20. Odabir poravnanja objekta Score

Izvor: Autor

4.1. Novčići

Na scenu se iz mape *Sprites* povuče model novčića, ranije preuzet s linka https://www.pngwing.com/en/ free-png-yiacw/download. Novčić se postavi na scenu, označi i unutar Inspectora dodaje mu se komponenta *Circle 2D Collider*. Važno je da mu se pridoda i oznaka (engl. *tag*) pod nazivom „*MyCoin*“.

Slika 21. Model novčića

Izvor: https://www.pngwing.com/en/free-png-yiacw/download

S obzirom da je glavni automobil objekt koji odlučuje o tome kad se funkcija novčića aktivira, skripta u kojoj se daje funkcionalnost novčiću je zapravo *CarController.cs*. Najprije se naredbom „*using UnityEngine.UI;“* uključuje mogućnost korištenja UI elemenata u skripti. Zatim se deklarira varijabla koja predstavlja tekst sa scene i varijabla koja će predstavljati trenutni broj bodova, odnosno sakupljenih novčića. Unutar metode *Update*, postavlja se da brojčana vrijednost trenutnih bodova bude prikazana pored teksta „Bodovi:“ u gornjem lijevom kutu. Početna vrijednost bodova postavi se na 0.

Zatim se napravi nova funkcija koja se aktivira kad se glavni automobil i novčić dodirnu. Ukoliko dođe do sudara auta i objekta koji ima tag „*MyCoin*“, potrebno je naš trenutni broj bodova povećati za deset, nakon toga uništiti objekt novčića, a broj bodova koji se prikazuje na sceni povećati za deset, odnosno pridodati mu vrijednost varijable *score*.

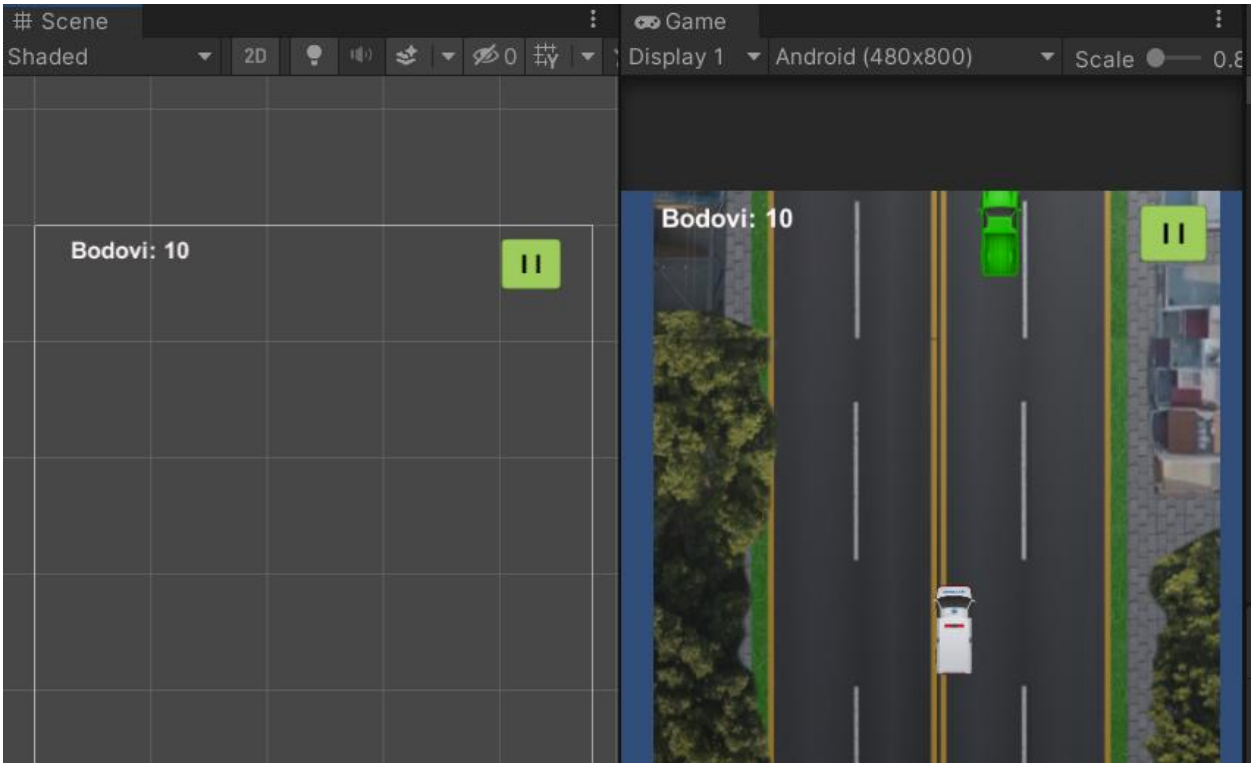
```
int score;

public Text scoreText;
```

```
 void Update()
    {
        scoreText.text = "Bodovi: " + score;
   }


private void OnTriggerEnter2D(Collider2D coin)
    {
        if(coin.tag == "MyCoin")
        {
            score += 10;
            Destroy(coin.gameObject);
        }
    }
```

Kod 13. Funkcija koja povećava broj bodova za deset nakon što automobil pokupi novčić

Izvor: Autor

Na sceni se označi glavni automobil i u *Inspectoru* se u polje „*Score Text*“ doda objekt *Score*. Pokreće se igra i sakupi se novčić kako bi se provjerila funkcionalnost prikupljanja bodova.



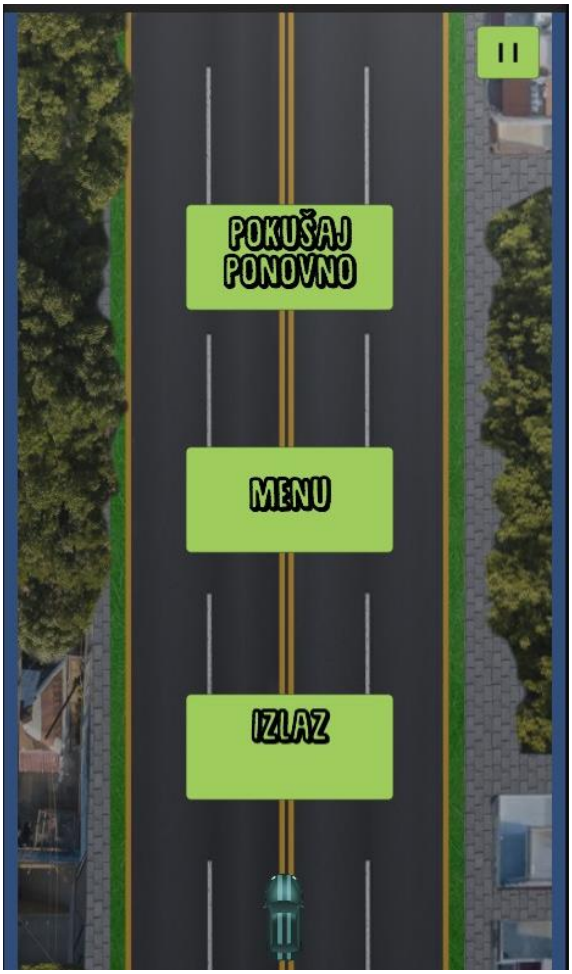
Slika 22. Prikaz bodova na sceni

Izvor: Autor

5. GAME MENI

Potrebno je kreirati izbornik koji će se aktivirati kad dođe do sudara. Taj izbornik nije potrebno raditi na novoj sceni, nego će se aktivirati na trenutnoj sceni. Najprije je potrebno umetnuti tri nova gumba od kojih će jedan služiti za ponovno pokretanje trenutne razine, drugi za odlazak na početnu scenu, a treći za izlaz iz igre.

Najjednostavniji način dodavanja novih gumba je taj da tri puta dupliciramo gumb *Pause*. Prvom gumbu daje se naziv *Replay Button*, drugom *Menu Button*, a trećem *Exit Button*. Postave se na sredinu scene s preciznim razmakom između svakog od njih. Za font teksta odabire se font "*Sugarpunch*" preuzet sa stranice www.dafont.com.



Slika 23. Izgled *game over* menija

Izvor: Autor

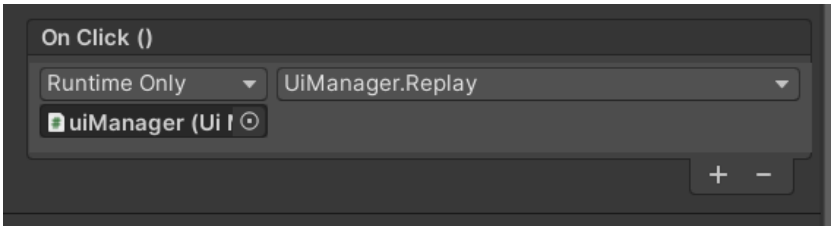
Potrebno je svakom gumbu dodati odgovarajuću funkciju. Kao što se ranije spomenulo, funkcije gumbima i ostalim UI (engl. *User Interface*) elementima postavljaju se u skripti *uiManager.cs*.

```
public void Replay()
    {
        Application.LoadLevel(Application.loadedLevel);
    }

    public void Menu()
    {
        Application.LoadLevel("Menu");
    }

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

Kod 14. Funkcije gumba aktiviranih nakon sudara

Izvor: Autor

S obzirom na to da se duplicirao gumb za pauziranje igre, trenutno svaki od tri gumba ima pridodanu funkciju *Pause*. Na isti način kao što se gumbu *Pause* pridodala njegova funkcija, na isti način se doda određena funkcija svakom drugom gumbu.



Slika 24. Primjer izgleda *Replay* gumba

Izvor: Autor

Gumbi „pokušaj ponovo“, „menu“ i „izlaz“ aktiviraju se samo ukoliko dođe do sudara s protivnikom. Najprije je svakog od njih potrebno isključiti i potom u skripti omogućiti njihovo uključivanje kad je igra gotova. Gumb se isključi tako da se makne kvačica u *inspector* prozoru pored njegovog imena.

Unutar skripte *UiManager.cs* deklarira se i varijabla naziva „*gameOver*“ tipa *bool* koja će određivati da li je igra završena ili ne. Unutar *start* funkcije vrijednost varijable *gameOver* postavlja se na *false*. Izrađuje se nova javna funkcija „*gameOverActivated*“ koja će se aktivirati svaki put kada se dogodi sudar s protivnikom, odnosno kada je igra završena.

```
bool gameOver;


void Start()
    {
        gameOver = false;
    }


public void gameOverActivated() {
        gameOver = true;
    }
```

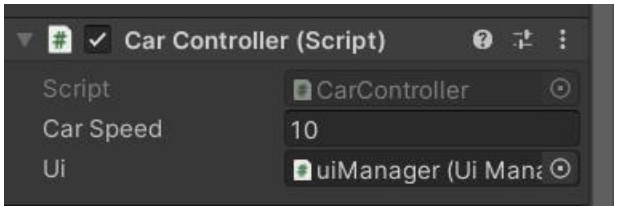
Kod 15. Izrada funkcije koja se aktivira kod završetka igre

Izvor: Autor

Unutar skripte *CarController* dodaje se nova javna varijabla kojom se daje pristup *UiManager* objektu preko *CarControllera*.

```
public UiManager ui;
```

Na sceni se označava glavni automobil i unutar za to predviđenog polja, povlači se objekt *UiManager*.



Slika 25. Dodavanje *UI Manager* skripte glavnom automobilu

Izvor: Autor

Unutar funkcije u skripti *CarController.cs* koja se poziva kad dođe do sudara s protivničkim automobilom, nakon uništenja glavnog automobila aktivirat će se funkcija *gameOverActivated*.

```
void OnCollisionEnter2D(Collision2D col)
    {
        if(col.gameObject.tag == "Enemy Car")
        {
            Destroy(gameObject);
            ui.gameOverActivated();
        }
```

```
}
```

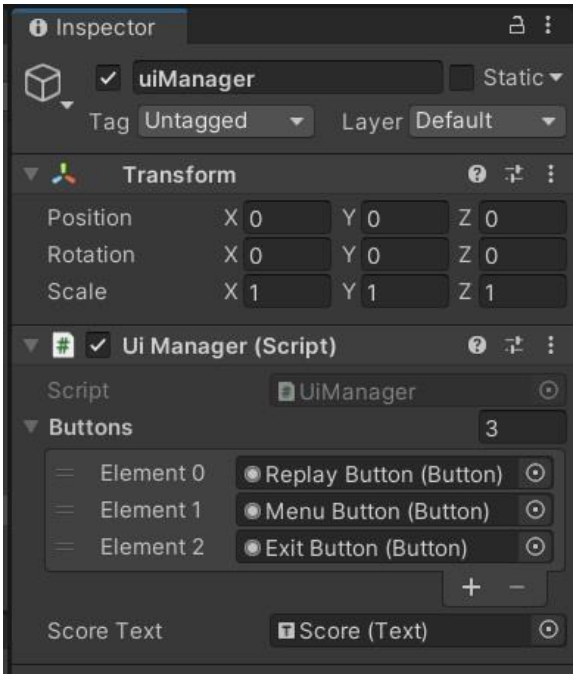
Kod 16. Pozivanje funkcije *gameOverActivated* nakon sudara s protivnikom

Izvor: Autor

Najprije se u skripti *UiManager.cs* kreira novo polje koje će činiti svi gumbi i aktivirat će se kad igra završi.

```
public Button[] buttons;
```

Ponovo se otvara program *Unity* i označuje objekt *uiManager*. Veličina polja postavi se na 3 i svakom elementu pridruži se gumb.



Slika 26. Gumbi kao elementi polja

Izvor: Autor

Zatim se unutar funkcije *gameOverActivated* pomoću *foreach* petlje uključuju gumbi koje smo prethodno isključili u samom *inspectoru*. Izrađuje se novi objekt *button* i unutar njega se pohranjuju elementi koji čine polje *buttons*.

```
foreach(Button button in buttons) {
            button.gameObject.SetActive(true);
        }
```

Kod 17. *Foreach* petlja kojom se aktiviraju gumbi

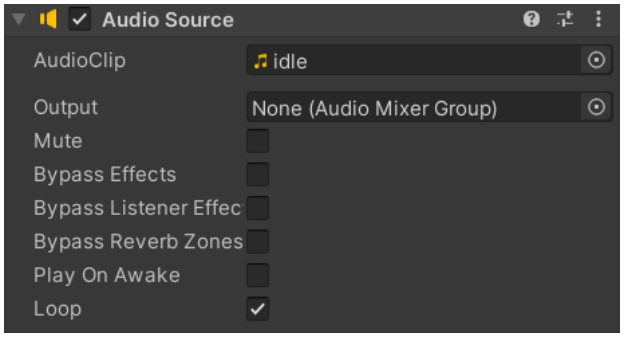
Izvor: Autor

6. ZVUK

6.1 Zvuk automobila

Najprije se izrađuje nova mapa unutar mape *Assets* i imenuje se u *Audio*. Preuzmu se besplatni zvukovi automobila sa službene stranice trgovine Unity-a dostupni na linku https://assetstore.unity.com/packages/audio/sound-fx/transportation/i6-german-free-engine-sound-pack-106037 i ubace se u novoizrađenu mapu.

Kreira se novi *GameObject* i imenuje se u *AudioManagerCar*. Zvuk se neće direktno projicirati s automobila već uz pomoć ovog posebnog objekta kojem je potrebno dodati komponentu *AudioSource* i njemu se pridodaje određeni zvuk. Odabire se zvučna datoteka imena „idle“. Uklanja se kvačica sa *Play on Awake* i postavlja se na *Loop* zbog toga što želimo da se zvuk automobila konstantno ponavlja tako dugo dok ne dođe do sudara.



Slika 27. *Audio Source* komponenta

Izvor: Autor

To će se postaviti u skripti koju kreiramo i dodamo objektu *AudioManagerCar*. Skripta će se zvati *AudioManager* i otvara se u programu *Visual Studio*. Najprije će se deklarirati varijabla kojom se daje pristup komponenti *AudioSource* i tako omogućiti kontrolu zvuka.

```
public AudioSource carSound;
```

Kod 18. Deklariranje varijable *carSound*

Izvor: Autor

Unutar skripte *CarController.cs* deklarira se varijabla *AudioManager* kojom se daje pristup skripti *AudioManager* tako da se bilo kad može zaustavljati i pokretati zvuk. Unutar metode *Awake* piše se naredba kojom se preko objekta *AudioManager* pokreće zvuk pri početku igre. S obzirom na to da nakon sudara i nestajanja automobila treba biti uklonjen i zvuk, unutar funkcije koja se poziva nakon sudara upisuje se naredba za prekid zvuka nakon uništavanja automobila i pozivanja „*gameOver*“ funkcije.

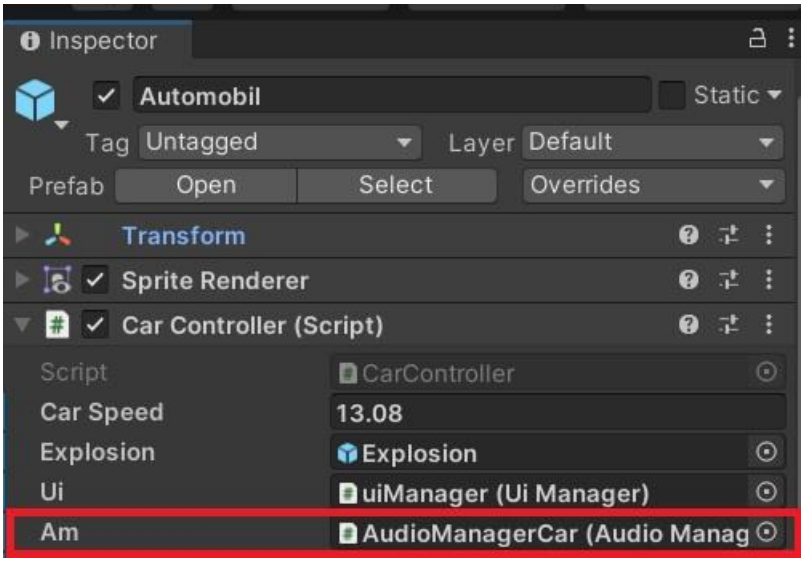
```
public AudioManager am;

private void Awake()
    {
        am.carSound.Play();
    }


void OnCollisionEnter2D(Collision2D col)
    {
        if(col.gameObject.tag == "Enemy Car")
      {
          ui.gameOverActivated();
          am.carSound.Stop();
      }
```

```
}
```

Kod 19. Kontrola pokretanja i zaustavljanja zvuka

Izvor: Autor

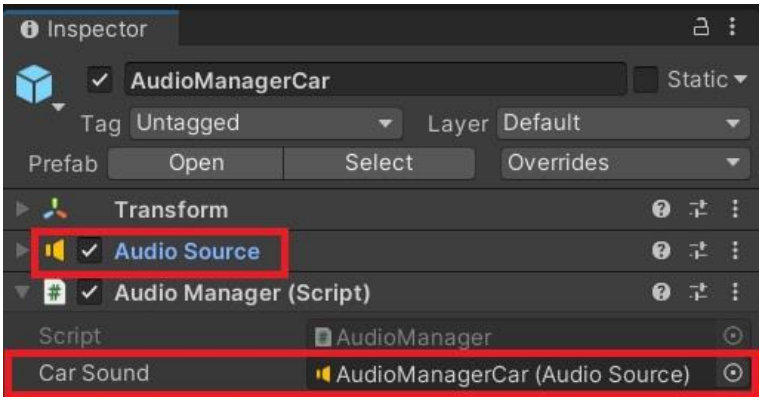
Ponovo se otvara *Unity* i unutar *inspector* prozora objekta Automobil, ubacuje se objekt *AudioManagerCar*.



Slika 28. Dodavanje objekta *AudioManagerCar* u automobil

Izvor: Autor

Također se označuje objekt *AudioManagerCar* i postavlja komponentu *AudioSource* unutar polja *Car Sound*.

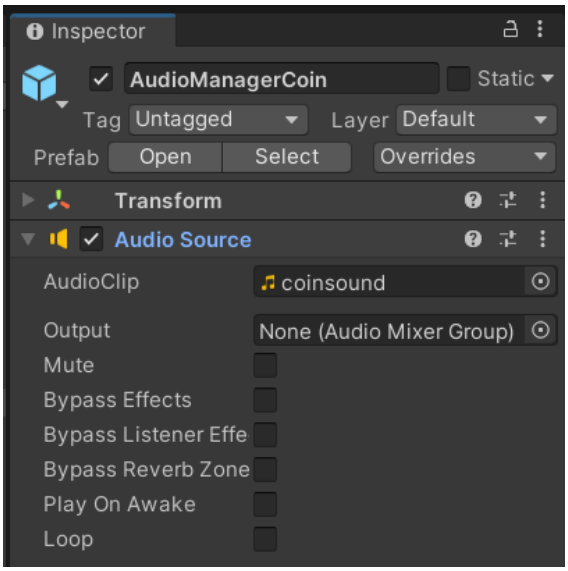


Slika 29. *Audio Source* unutar polja *Car Sound*

Izvor: Autor

6.2. Zvuk novčića

Unutar mape *Audio* ubacuje se zvuk novčića koji će se koristiti, preuzet sa stranice: https://mixkit.co/free-sound-effects/coin/. Kreira se novi objekt i imenuje u *AudioManagerCoin* . Dodaje mu se komponenta *AudioSource* i ubacuje se zvuk naziva *coinsound* koji će se koristiti.



Slika 30. *Audio Source* komponenta za novčić

Izvor: Autor

Objektu se doda skripta naziva *CoinAudioSc* koja se zatim otvara u programu *Visual Studio*. Deklarira se javna varijabla *coinSound* koja će kontrolirati *Audio Source* u kojem je smješten zvuk novčića.

```
public AudioSource coinSound;
```

Kod 20. Deklariranje varijable *coinSound* unutar skripte *CoinAudioSc*

Izvor: Autor

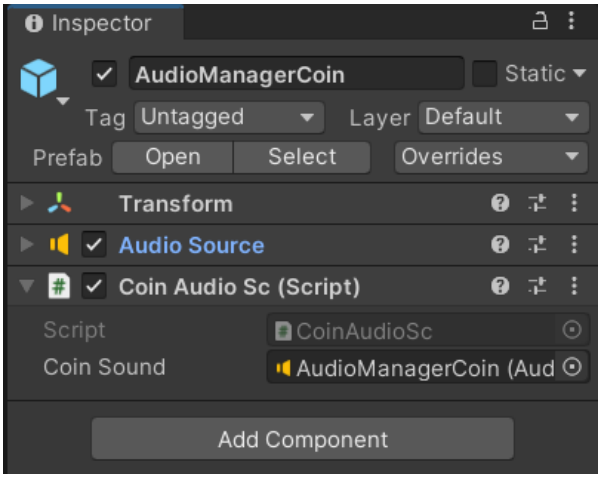
S obzirom na to da zvuk novčića želimo čuti nakon njegovog sudara s autom, potrebno je napraviti određene promjene unutar skripte *CarController*. Za početak se deklarira varijabla kojom se daje pristup klasi *CoinAudioSc*. Zatim se pokreće zvuk novčića unutar funkcije koja se poziva nakon sudara automobila s novčićem.

```
public CoinAudioSc ca;


private void OnTriggerEnter2D(Collider2D coin)
    {
        if(coin.tag == "MyCoin")
        {
            score += 10;
            Destroy(coin.gameObject);
            ca.coinSound.Play();
      }
```

Kod 21. Aktiviranje zvuka novčića nakon sudara s automobilom

Izvor: Autor

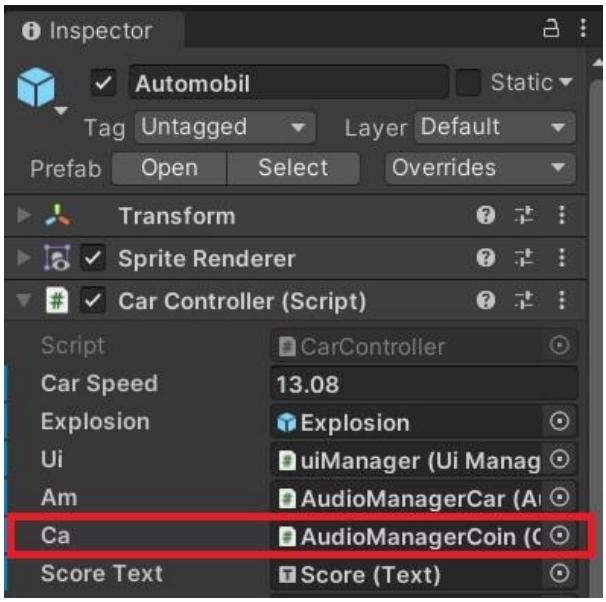
Ponovo se pokreće *Unity*, označuje objekt *AudioManagerCoin* i dodaje se komponenta *Audio Source* na mjesto predviđeno za zvuk automobila.



Slika 31. Dodavanje komponente *Audio Source* u polje *Coin Sound*

Izvor: Autor

U prozoru *inspector* objekta „Automobil“, unutar komponente za skriptu *CarController*, ubacuje se objekt *AudioManagerCoin*.

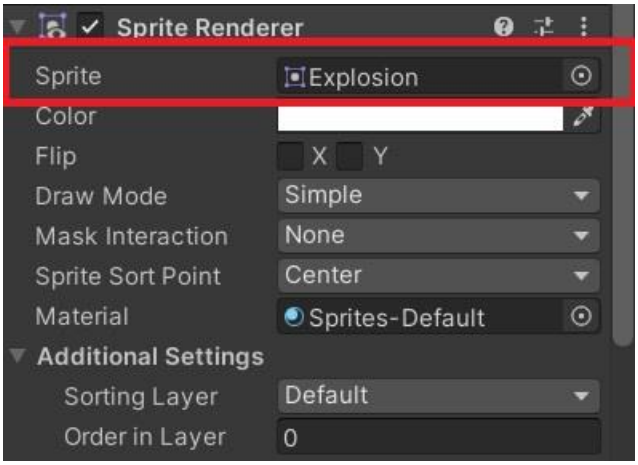


Slika 32. Dodavanje objekta *AudioManagerCoin* u skriptu automobila

Izvor: Autor

7. EKSPLOZIJA

Animacija eksplozije dogodit će nakon sudara glavnog i protivničkog automobila. Najprije se dodaje novi objekt kojem se daje naziv „*Explosion*“. U mapu *Sprites* dodaje se slika preuzeta sa stranice: https://opengameart.org/content/comic-explosion-kaboom i povuče unutar *Sprite Renderer* komponente novoizrađenog objekta.



Slika 33. Dodavanje slike eksplozije unutar *Sprite Renderer* komponente

Izvor: Autor

Zatim se kreira skripta „Eksplozija.cs“ i dodaje novoizrađenom objektu. Najprije se unutar skripte *CarController.cs* deklarira polje za objekt u koji će kasnije biti dodana eksplozija.

```
[SerializeField] private GameObject explosion;
```

Premda se eksplozija mora pojaviti nakon sudara, unutar funkcije koja se poziva nakon sudara s protivničkim automobilom dodaje se linija koda kojom se omogućuje pojavljivanje eksplozije na mjestu na sceni gdje se dogodio sudar.

Funkcija koja se poziva kod sudara sada izgleda ovako:

```
void OnCollisionEnter2D(Collision2D col)
```

```
    {
        if(col.gameObject.tag == "Enemy Car")
        {
            Instantiate (explosion, transform.position,
Quaternion.identity);
            Destroy(gameObject);
            ui.gameOverActivated();
            am.carSound.Stop();

        }
```

Kod 22. Funkcija koja se poziva kod sudara s protivničkim automobilom

Izvor: Autor

Potrebno je i odrediti koliko dugo će eksplozija trajati, odnosno koliko dugo će objekt biti prikazan na sceni. Otvara se skripta „Eksplozija.cs“ u programu *Visual Studio* i unutar *Start* metode se odredi koliko vremena nakon pojave će se uništiti objekt eksplozije.

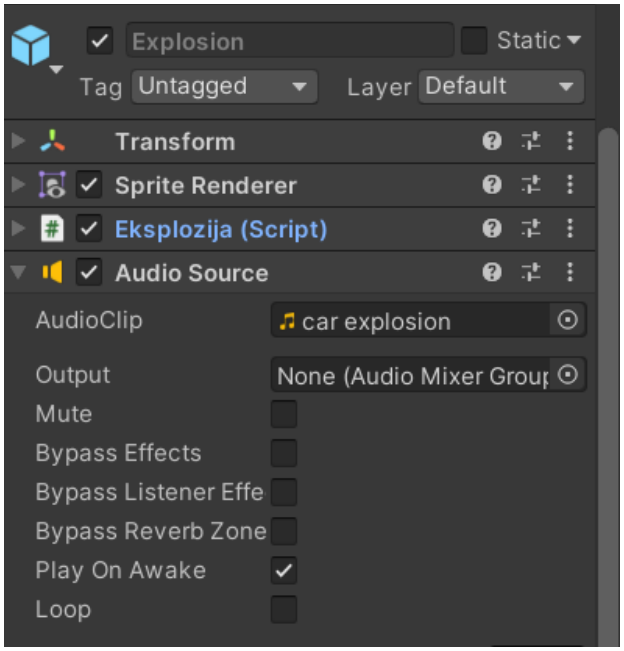
```
void Start()
    {
        Destroy(gameObject, 1.5f);
    }
```

Kod 23. Uništenje objekta nakon sekunde i pol

Izvor: Autor

Zvuk koji će se koristiti kod eksplozije preuzima se sa stranice: https://assets.mixkit.co/sfx/download/mixkit-car-explosion-debris-1562.wav.

Objektu *Explosion* dodaje se komponenta *Audio Manager* i unutar polja *Audio Clip*, ubacuje se zvuk koji se aktivira uz animaciju eksplozije.



Slika 34. *Audio Source* komponenta eksplozije

Izvor: Autor

8. PRIJELAZ NA IDUĆU RAZINU

Igra funkcionira tako da se nakon ostvarenih sto bodova, odnosno skupljenih deset novčića, prelazi na novu razinu. Najprije se otvara skripta *CarController.cs* u programu *Visual Studio*.

Na vrhu koda upisuje se naredba kojom će se koristiti mogućnost upravljanja scenom.

```
using UnityEngine.SceneManagement;
```

Zatim se kreira funkcija unutar koje se poziva učitavanje scene koja dolazi nakon trenutne [5].

```
public void LoadNextLevel()
    {

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

Kod 24. Funkcija učitavanja sljedeće scene

Izvor: Autor

Potrebno je novoizrađenu funkciju pozvati u onom trenutku kad vrijednost varijable *score* bude 100.

```
if (score >= 100)
            {
                LoadNextLevel();
            }
```

Kod 25. Učitavanje sljedeće razine nakon skupljenih 100 bodova

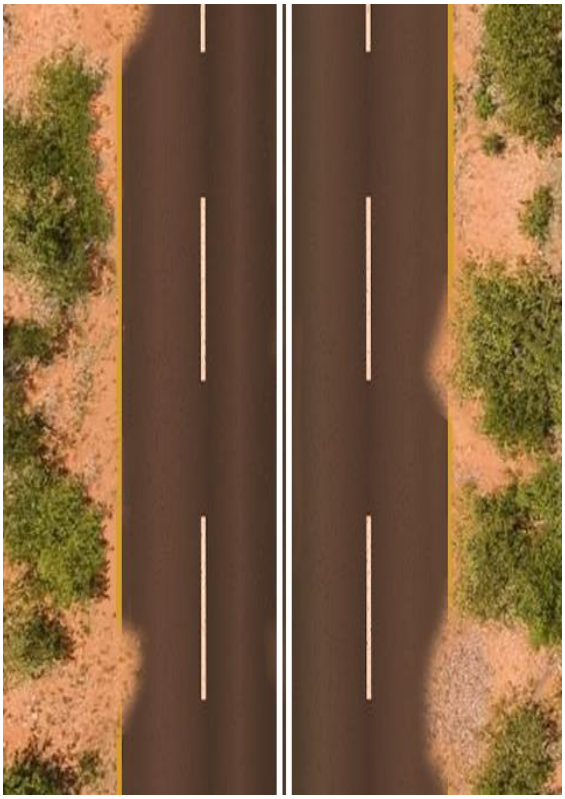
Izvor: Autor

8.1. Izrada druge razine

Kreirat će se nova razina u kojoj će se protivnički automobili kretati većom brzinom i pojavljivati brže na sceni. Pozadina će biti cesta drugačijeg izgleda i u drugačijim uvjetima, a slika ceste je izrađena je u programu *Adobe Photoshop*.

Da bi se kreirala nova razina, najprije je na trenutnoj potrebno napraviti *prefab* svih objekta koje smo koristili. Jednostavno se svi elementi iz *Hierarchy* prozora kopiraju u mapu *Prefabs* tako da se mogu koristiti na sljedećoj razini.

Zatim se kreira nova scena i sadržaj iz mape *Prefab* dodaje se u *hierarchy* prozor. Unutar mape *Sprites* dodaje se slika ceste koja će biti na ovoj razini. Potrebno je kao i na prvoj razini *Texture Type* modela ceste postaviti na „*Default*“, a *Wrap Mode* na „*Repeat*“.



Slika 35. Izgled ceste na drugoj razini

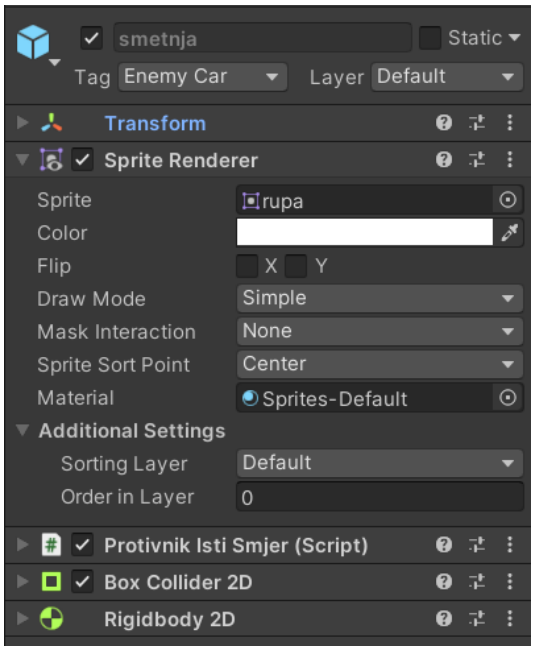
Izvor: Autor

Sprema se scena pod nazivom Level2 i nakon toga se dodaju novi objekti.

8.1.2. Dodavanje prepreke

Na drugoj razini dodat će se novi objekt prepreke. To će biti neka vrsta rupe u cesti, vrlo prigodno za igru imena „*CroRacing*“. Objekt će imati isti zadatak i kretat će se istim putanjama kao protivnički automobili. Nakon umetanja na samu scenu, ubacit će se u polje car *Car Spawner* tako da se nasumično pojavljuje na sceni. Prepreka je preuzeta sa linka: https://www.vhv.rs/dpng/f/5-57422_bullet-hole-png.png, a implementira se u ovu igru ovim koracima:

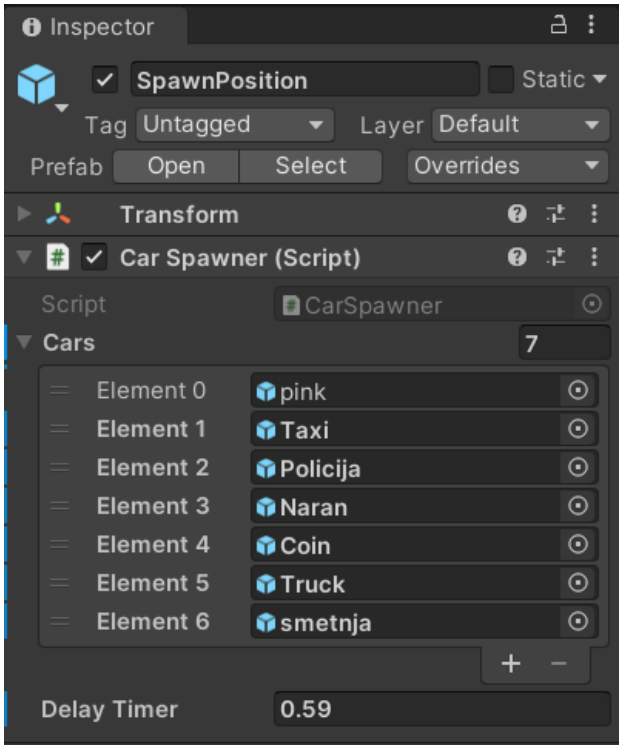
1. Slika prepreke ubaci se u mapu *Sprites*
2. Duplicira se objekt protivnika i ime promijeni u smetnja
3. Unutar komponente *Sprite Renderer*, ubaci se slika prepreke
4. Objekt ima ista svojstva kao i protivnički auto



Slika 36. *Inspector* objekta smetnje

Izvor: Autor

Označuje se objekt *CarSpawner* i u prozoru *inspector* u polje *Cars* dodaje se objekt smetnje, kamiona i policije koji će se pojavljivati na sceni zajedno s ostalim protivnicima u razmaku od 0,59 sekundi, dok se na prvoj razini pojavljuju u razmaku od 0,84 sekunde.



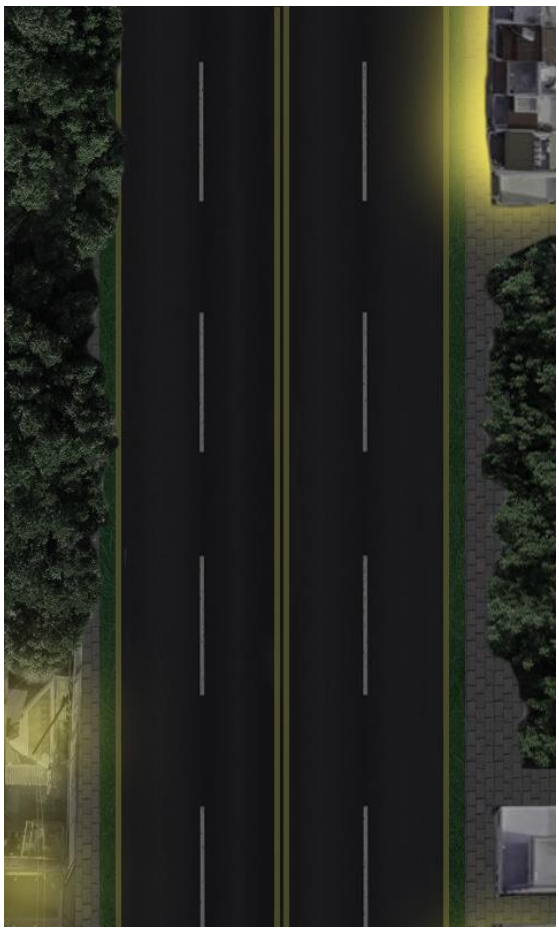
Slika 37. novi objekti u polju *Cars*

Izvor: Autor

Potrebno je i povećati brzinu kretanja glavnog automobila. Označuje se objekt „Cesta“ i varijabla *Speed* postavi se na 1,38.

8.2. Izrada treće razine

Postoji sljedeći način za izradu nove razine. Jednostavno se duplicira scena pod nazivom „Level2“ i promjeni se ime u „Level3“. U mapu *Sprites* ubacuje se model ceste ranije kreiran u programu *Adobe Photoshop*.



Slika 38. Izgled ceste na trećoj razini

Izvor: Autor

Brzina glavnog automobila postavi se na 1,38, a vrijeme pojavljivanja protivnika nakon 0,41 sekunde. Na ovoj razini potrebno je imati zaista dobre reflekse i oprezno upravljati automobilom. Brzina gibanja i pojavljivanja protivnika je sve brža.

9. ZAVRŠNA SCENA

Nakon što se doda treća razina, potrebno je izraditi završnu scenu koja će se otvoriti nakon sakupljenih 100 bodova na posljednjoj (trećoj) razini. Dodaje se nova scena pritiskom na *File -> New Scene*. Scena se sprema i ime se postavi u „Kraj“. Na sceni će biti dva gumba od kojih će jedan služiti za izlaz iz igre, a drugi za ponovnu igru. Unutar mape *Sprites* dodaje se slika završne scene dizajnirana u programu *Adobe Photoshop*. Kreira se novi objekt i imenuje u „KRAJ“. Dodaje mu se komponenta *Sprite Renderer* i unutar nje slika iz mape *Sprites*.

Slika 39. Završna scena

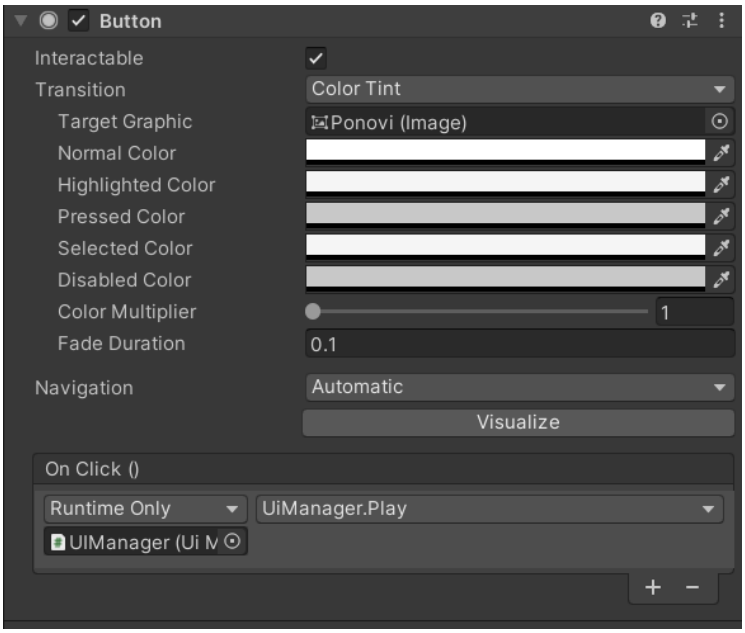
Izvor: Autor

Kreira se novi objekt *UIManager* i dodaje mu se skripta *UiManager.cs*. Na scenu se dodaju dva gumba desnim klikom unutar *hierarchy* prozora -> *UI* -> *Button* i postavljaju na predviđena mjesta. Jednom objektu gumbu dodijeli se naziv „Ponovi“, a drugome „Kraj2“.

Slika 40. Pozicije gumba na završnoj sceni

Izvor: Autor

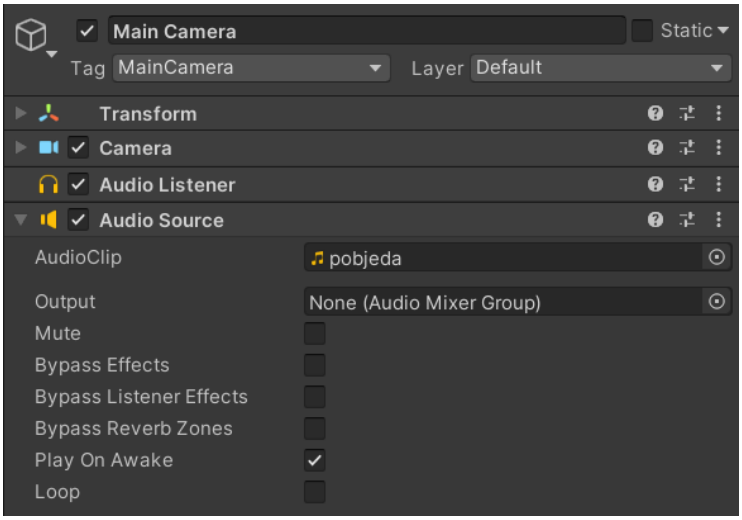
Gumbu „Ponovi“ doda se funkcija *Play*() koja se poziva klikom na gumb i vraća nas na prvu razinu, a gumbu „Kraj2“ funkcija *Exit*() koja izlazi iz igre.



Slika 41. Inspector gumba "Ponovi"

Izvor: Autor

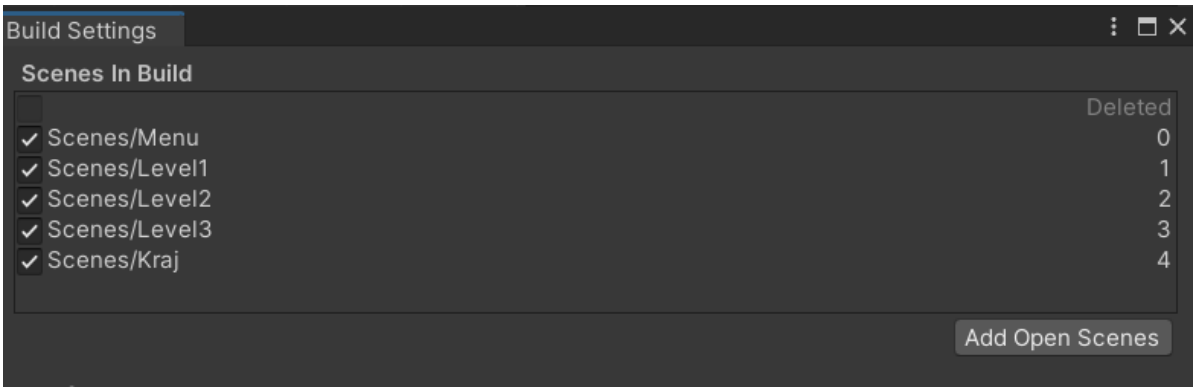
Završnoj sceni dodaje se i pobjednički zvuk koji je dostupan na stranici: https://mixkit.co/free-sound-effects/win/ i sprema se u mapu *Audio*. Označuje se objekt *Main Camera* i dodaje mu se komponenta *Audio Source*. Zvuk koji želimo koristiti iz mape *Audio* postavimo u polje *AudioClip*.



Slika 42. komponenta *AudioSource* na završnoj sceni

Izvor: Autor

Pritiskom na *File-> Build Settings* otvara se prozor u koji se ubacuju scene koje će se koristi i otvarati željenim redoslijedom.



Slika 43. Raspored učitavanja scena

Izvor: Autor

10. ZAKLJUČAK

Računalne igre su me zanimale, koliko i fascinirale još otkada sam prvi puta igrao jednostavne igre poput *Super Maria* i *Fife*. S vremenom me više od pobjede u igri počelo zanimati i kako one nastaju, što mi se uvijek činilo kompliciranim.

Kroz kolegij „Izrada računalnih igara“ stekao sam uvid u sam proces izrade igre od nule i shvatio koliko je znanja iz različitih područja potrebno za kreiranje videoigre. Osim znanja u programu *Unity*, potrebno je baratati osnovnim vještinama programiranja u C# ili nekom drugom programskom jeziku, a poželjno je i znanje korištenja *Blendera, Photoshopa*, ili nekog drugog dizajnerskog alata. U većim tvrtkama na izradi jedne igre radi preko stotinu developera, programera, dizajnera i drugih stručnjaka.

Odabirom ove teme završnog rada stekao sam priliku da se i sam okušam u izradi jednostavne videoigre i bilo je izuzetno zanimljivo. Shvatio sam da je stvaranje videoigre ujedno kompliciran i jednostavan proces. Potrebne su vještine iz različitih područja, no primjenom osnovnih znanja iz svakog od njih moguće je izraditi jednostavnu, dinamičnu i zabavnu videoigru.

POPIS LITERATURE

[1] Cheryl Olson, Sc.D, 9 Benefits of Video Games for Your Child (15.2.2022.)

https://www.parents.com/kids/development/benefits-of-video-games/

[2] Unity - Instalation and Setting Up (26.5.2021.)

https://www.tutorialspoint.com/unity/unity_installation_and_setting_up.htm

[3] Introduction to Unity: Getting Started – Part ½ (26.5.2021.)

https://www.raywenderlich.com/7514-introduction-to-unity-getting-started-part-1-2

[4] Unity - Understanding Collisions (2.6.2021.)

https://www.tutorialspoint.com/unity/unity_understanding_collisions.htm

[5] SceneManager.LoadScene (25.6.2021)

https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html

POPIS PROGRAMSKIH KODOVA

Kod 1. Omogućavanje vertikalnog gibanja ceste 17
Kod 2. Deklariranje varijabli u skripti CarController.cs 18
Kod 3. Program kontroliranja kretnji automobila 19
Kod 4. gibanje protivničkog automobila po Y koordinati 21
Kod 5. Stvaranje objekta Car 22
Kod 6. Stvaranje automobila nakon svake sekunde 25
Kod 7. Uništenje glavnog objekta ukoliko se sudari sa protivnikom 28
Kod 8. Uništavanje protivničkog automobila nakon sudara s objektom *Destroyer* 29
Kod 9. Skripta za nasumično pojavljivanje različitih modela automobila 34
Kod 10. Deklariranje funkcije za pauziranje igre 37
Kod 11. Funkcija *Play* za pokretanje prve razine 41
Kod 12. Otvaranje i zatvaranje *panela* 44
Kod 13. Funkcija koja povećava broj bodova za deset nakon što automobil pokupi novčić 47
Kod 14. Funkcije gumba aktiviranih nakon sudara 50
Kod 15. Izrada funkcije koja se aktivira kod završetka igre 52
Kod 16. Pozivanje funkcije *gameOverActivated* nakon sudara s protivnikom 53
Kod 17. *Foreach* petlja kojom se aktiviraju gumbi 54
Kod 18. Deklariranje varijable *carSound* 56
Kod 19. Kontrola pokretanja i zaustavljanja zvuka 57
Kod 20. Deklariranje varijable *coinSound* unutar skripte *CoinAudioSc* 59
Kod 21. Aktiviranje zvuka novčića nakon sudara s automobilom 59
Kod 22. Funkcija koja se poziva kod sudara s protivničkim automobilom 62
Kod 23. Uništenje objekta nakon sekunde i pol 62
Kod 24. Funkcija učitavanja sljedeće scene 64
Kod 25. Učitavanje sljedeće razine nakon skupljenih 100 bodova 64

POPIS SLIKA

Slika 1. Izrada novog projekta 8
Slika 2. Izrada mape "Scenes" 9
Slika 3. Spremanje scene 10
Slika 4. Promjena rezolucije automobila 11
Slika 5. Postavljanje rezolucije 12
Slika 6. Promjena *Texture* i *Wrap* komponente ceste 13
Slika 7. Transform komponenta ceste 15
Slika 8. Izgled igre nakon 2 protivnička automobila 31
Slika 9. Mapa *Prefab* sa svim modelima automobila 32
Slika 10. Protivnički automobili umetnuti u polje Cars 33
Slika 11. Promjena fokusa sa scene na UI elemente 35
Slika 12. Izgled i položaj gumba na sceni 36
Slika 13. Pridruživanje funkcije Pause gumbu 38
Slika 14. *Game menu* slika u programu *Adobe Photoshop* 39
Slika 15. Izrada gumba za početak igre 40
Slika 16. Isključivanje vidljivosti gumba 40
Slika 17. Aktiviranje funkcije *Play* na pritisak gumba 41
Slika 18. Dodavanje objekta panel 42
Slika 19. Meni scena nakon otvaranja uputa 43
Slika 20. Odabir poravnanja objekta Score 45
Slika 21. Model novčića 46
Slika 22. Prikaz bodova na sceni 48
Slika 23. Izgled *game over* menija 49
Slika 24. Primjer izgleda *Replay* gumba 51
Slika 25. Dodavanje *UI Manager* skripte glavnom automobilu 52
Slika 26. Gumbi kao elementi polja 53
Slika 27. *Audio Source* komponenta 55
Slika 28. Dodavanje objekta *AudioManagerCar* u automobil 57
Slika 29. *Audio Source* unutar polja *Car Sound* 58

Slika 30. *Audio Source* komponenta za novčić 58
Slika 31. Dodavanje komponente *Audio Source* u polje *Coin Sound* 60
Slika 32. Dodavanje objekta *AudioManagerCoin* u skriptu automobila 60
Slika 33. Dodavanje slike eksplozije unutar *Sprite Renderer* komponente 61
Slika 34. *Audio Source* komponenta eksplozije 63
Slika 35. Izgled ceste na drugoj razini 65
Slika 36. *Inspector* objekta smetnje 66
Slika 37. novi objekti u polju *Cars* 67
Slika 38. Izgled ceste na trećoj razini 68
Slika 39. Završna scena 69
Slika 40. Pozicije gumba na završnoj sceni 70
Slika 41. Inspector gumba "Ponovi" 71
Slika 42. komponenta *AudioSource* na završnoj sceni 71
Slika 43. Raspored učitavanja scena 72