IZRADA 3D IGRE U UNITY RAZVOJNOM OKRUZENJU

Pera, Ante

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Računarstvo

ANTE PERA

Z A V R Š N I R A D

IZRADA 3D IGRE U RAZVOJNOM OKRUŽENJU UNITY

Split, rujan 2023.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Računarstvo

Predmet: Objektno programiranje

Z A V R Š N I R A D

Kandidat: Ante Pera

Naslov rada: Izrada 3D igre u razvojnom okruženju Unity

Mentor: Ljiljana Despalatović, viši predavač

Split, rujan 2023.

SADRŽAJ

SAŽETAK

Ovaj završni rad fokusira se na razvoj i implementaciju *time trial* utrke unutar Unity razvojnog okruženja. *Time trial* utrke su popularan žanr u video igrama gdje igrači pokušavaju što brže prijeći određenu stazu, pri čemu se njihovo vrijeme bilježi i uspoređuje s rekordnim vremenom. Cilj rada je istražiti i demonstrirati ključne korake u stvaranju takvog iskustva koristeći Unity platformu.

U radu će se detaljno analizirati komponente potrebne za izgradnju *time trial* utrke, uključujući stvaranje staze, implementaciju igračevog vozila s upravljačkim mehanizmima, postavljanje vremenskog rekorda te implementaciju sustava za praćenje vremena.

Ključne riječi: Simulacija, Staza, *Time trial*, Unity, Video igre, Vozilo

SUMMARY

MAKING A 3D GAME IN UNITY DEVELOPMENT ENVIRONMENT

This thesis focuses on the development and implementation of a *time trial* race within the Unity development environment. *Time trial* races are a popular genre in video games where players try to complete a certain course as quickly as possible, with their time recorded and compared to the record time. The goal of the paper is to explore and demonstrate the key steps in creating such an experience using the Unity platform.

The paper will analyze in detail the components needed to build a *time trial* race, including creating a track, implementing a player's vehicle with control mechanisms, setting a time record, and implementing a time tracking system.

Keywords: Simulation, *Time trial*, Track, Unity, Video games, Vehicle

1. Uvod

U današnjem digitalnom dobu, igre su postale neizostavan dio naše svakodnevnice. Od zabave i rekreacije do edukacije i simulacija, igre su evoluirale iz jednostavnih formata u kompleksne i interaktivne svjetove koji angažiraju igrače na više razina. Unity *engine* istaknuo se kao jedan od najmoćnijih alata za razvoj takvih igraćih iskustava, omogućujući developerima da stvore vizualno impresivne, tehnički napredne i intuitivne igre za različite platforme.

Ovaj završni rad istražuje proces razvoja igre u Unity *engineu*, fokusirajući se na korake koje developeri poduzimaju kako bi stvorili igraće iskustvo koje je privlačno, funkcionalno i korisnički prijateljsko. Kroz detaljno istraživanje dizajna igara, upotrebe različitih komponenata i funkcionalnosti unutar Unity platforme, kao i tehničke implementacije, ovaj rad pruža uvid u kompleksnost i kreativnost koja stoji iza modernih video igara.

Cilj ovog završnog rada je bio razviti videoigru koja bi igrače upoznala s poznatim rutama uličnih trkača iz 90-ih godina prošlog stoljeća. Većina vozača koja voze brzinama iznad ograničenja su mladi vozači koji ne znaju rute i još ne znaju voziti. Ova videoigra bi upoznala igrače s rutama tako da ako budu vozili u stvarnosti budu već upoznati kojom brzinom mogu prijeći cijeli rutu. Također bi smanjila broj nesreća i potencijalnih opasnosti samim time što već nudi znanje same rute, a uz to vozači uopće ne moraju napuštati dom da bi vozili svoje rute i ne moraju riskirati svoj i tuđe živote.

Ovaj rad se sastoji od pet poglavlja. Nakon kratkog uvoda, u drugom poglavlju je opisan Unity editor. U trećem poglavlju je opisan program za modeliranje Blender zajedno s eksternim GIS alatom. U četvrtom poglavlju je opisan glavni dio rada, proces razvoja sustava i korišteni alati za razvoj. Zaključno mišljenje je opisano u posljednjem poglavlju.

2. Unity engine

Unity je *game engine* razvijen 2005. godine za OS X platformu, ali ubrzo se proširio i na ostale platforme. Cilj Unityja je bio da bude dostupan po prihvatljivoj cijeni široj javnosti. Postao je popularan jer je svima omogućio izradu igrica te se povećava broj igrica razvijene u njemu. Namijenjen je za izradu 3D i 2D igrica. Od 2009 godine dostupna je besplatna verzija Unityja. Unity je dostupan na linku [1]. Trenutačna stabilna verzija Unityja je Unity 2022.3.4f1 (LTS).

Unity dolazi u četiri verzije u odnosu na plaćanje i mogućnosti:

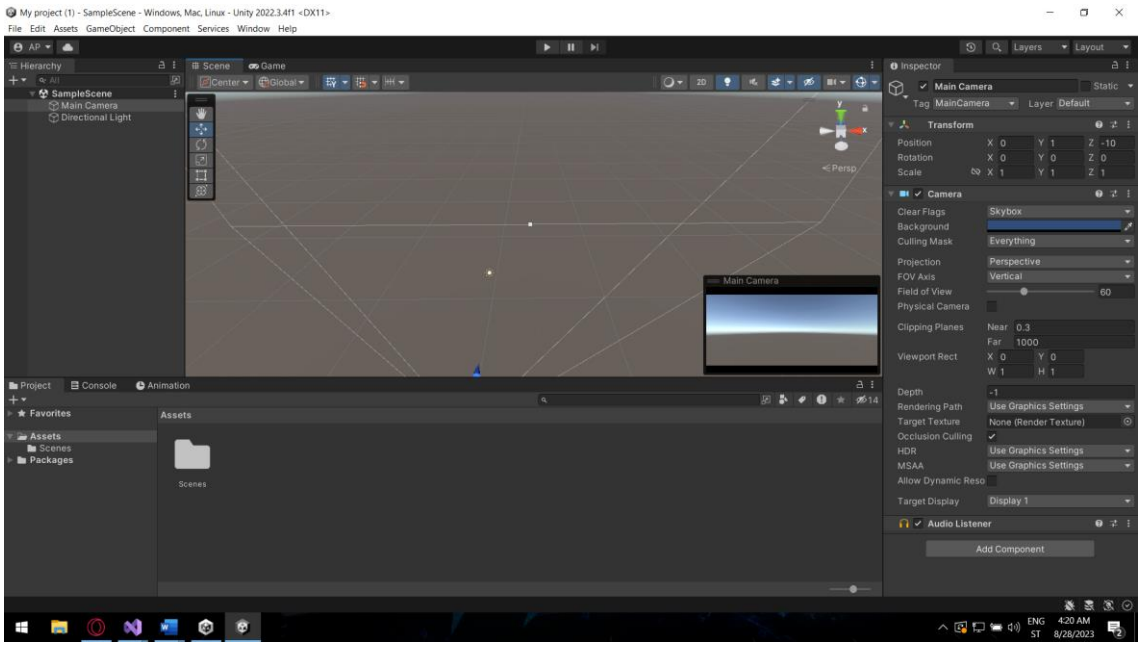
- **Personal** je besplatna verzija Unityja koja je limitirana. Zarada od igrica razvijene u ovoj verziji ne smije preći $100,000 godišnje. Osim ograničene zarade u ovoj verziji tema editora je bijele boje te se ne može promijeniti. I pri pokretanju igrice uvijek će se prikazati logo Unityja.
- **Plus** je verzija Unitya koja košta $369, ali se plaća $30.75 mjesečno na 12 rata. Ova verzija dolazi sa crnom temom i pri pokretanju igrice se ne mora prikazati logo Unityja. Zarada ne smije preći $200,000 godišnje. U ovoj verziji omogućen izvještaj o performansi i „real time“ osvijetljenje.
- **Pro** je verzija Unityja koja košta $1877, ali se plaća $157 mjesečno na 12 rata. Razlikuje se od Plus verzije po tome što ima premium podršku i neograničenu zaradu.
- **Enterprise** je verzija namijenjena tvrtkama s više od 20 zaposlenih, cijena verzije je po dogovoru. Ima sve mogućnosti i svu podršku uključujući izvorni kôd samog engina.

Najveća prednost Unityja je to što radi na 27 platformi, neke od njih su: PC, *Android*, iOS, *Playstation 4*, *Xbox* i druge. Veliki broj platformi mu omogućuje to što je podržava *DirectX* i *OpenGL*. Igricu koju napravimo će raditi na svim platformama. Unity ima jako dobru dokumentaciju, službena videa i commUnity te je zbog toga dobar za početnike.

Nedostatak mu je to da je lošiji u grafičkom pogledu u odnosu na *Unreal Engine* 5.

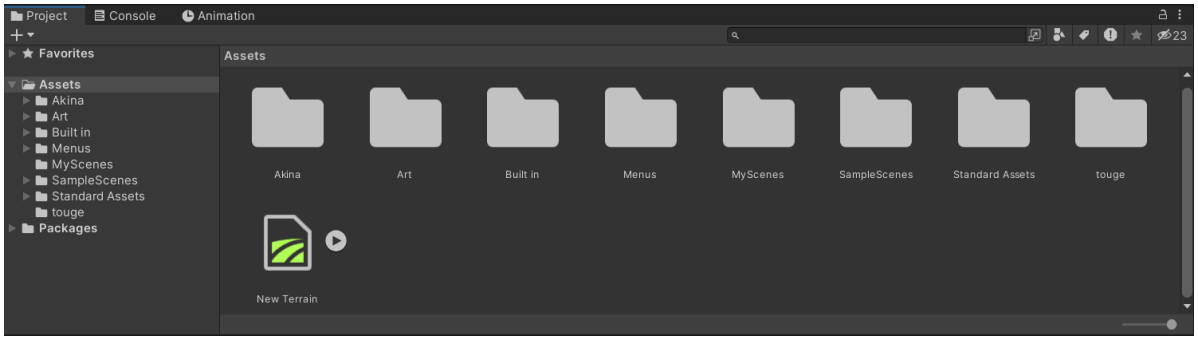
2.1. Osnovni dijelovi Unityja

Unity uređivač (engl. *editor*) je centralni alat koji pruža integrirano razvojnookruženje za stvaranje interaktivnih 2D i 3D sadržaja. Omogućava developerima da vizualno dizajniraju, programiraju, testiraju i optimiziraju svoje projekte. Unity *editor* je prikazan na slici 2-1.



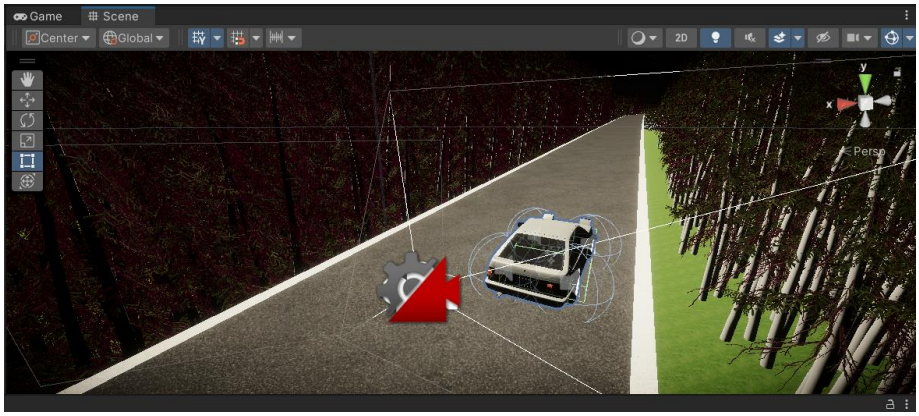
Slika 2-1 Unity editor

- **Projektni prikaz (engl. *Project view*)** je upravitelj resursima (engl. *resource manager*) Unityja, ovdje se nalaze sve komponente koji su potrebni za izradu igrice. Ovdje možemo kreirati mape tako da grupiramo komponente po vrsti. Sastoji se od dva dijela. Lijevi dio prikazuje hijerarhijsku listu mapa, a desni dio prikazuje komponente u ikonama. Imamo mogućnost kreiranja novih komponentih botunom *Create* koji se nalazi u gornjem lijevom kutu. Sve ovdje se nalazi spremljeno u mapi *Assets*. *Project view* je prikazan na slici 2-2.



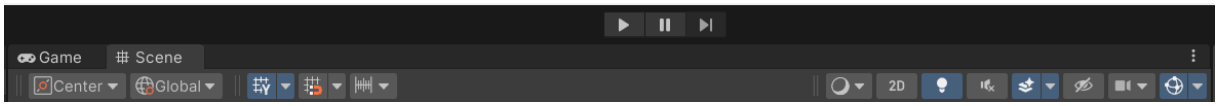
Slika 2-2 Project view

- **Scenski prikaz (engl. *Scene view*)** (slika 2-3) je scena *engina* u njoj se nalazi cijeli prostor igrice te se u njoj postavljaju objekti. U sceni možemo micati, rotirati i skalirati objekte. U gornjem desnom kutu se nalazi scene Gizmo. On prikazuje trenutnu rotaciju kameru scene te pomoću scene Gizma možemo mijenjati prespektivu gledanja. Svaka os u sceni je obojena drugom bojom. X (crvenom), Y (zelenom) i Z (plavom). U gornjem lijevom kutu se nalaze alati za transformaciju redom od gore prema dolje:
    - *View tool* - omogućava korisnicima da pomiču svoj pogled po sceni bez promjene položaja ili orijentacije objekata.
    - *Move tool* - Ovaj alat omogućava korisnicima da pomiču odabrane objekte duž različitih osi (X, Y, Z) unutar scene. To je korisno za precizno pozicioniranje objekata.
    - *Rotate tool* - omogućava korisnicima da rotiraju odabrane objekte oko različitih osi. To je korisno za postavljanje orijentacije objekata, kao i za kreiranje animacija rotacije.
    - *Scale tool* - omogućava korisnicima da promijene veličinu odabranih objekata duž različitih osi. To je korisno za prilagodbu veličine objekata ili za stvaranje animacija skaliranja.
    - *Rect tool* - omogućava korisnicima da kreira i uređuje tzv. "rectangles" ili pravokutnike unutar *Scene Viewa*, koristan je za definiranje površina, unutar scene koje mogu imati određenu funkcionalnost ili svojstva.
    - *Transform tool* - skup alata koji omogućava korisnicima da pomiču, rotiraju i skaliraju objekte unutar scene. *Transform Tool* uključuje tri osnovna alata: *Move Tool, Rotate Tool i Scale Tool*.



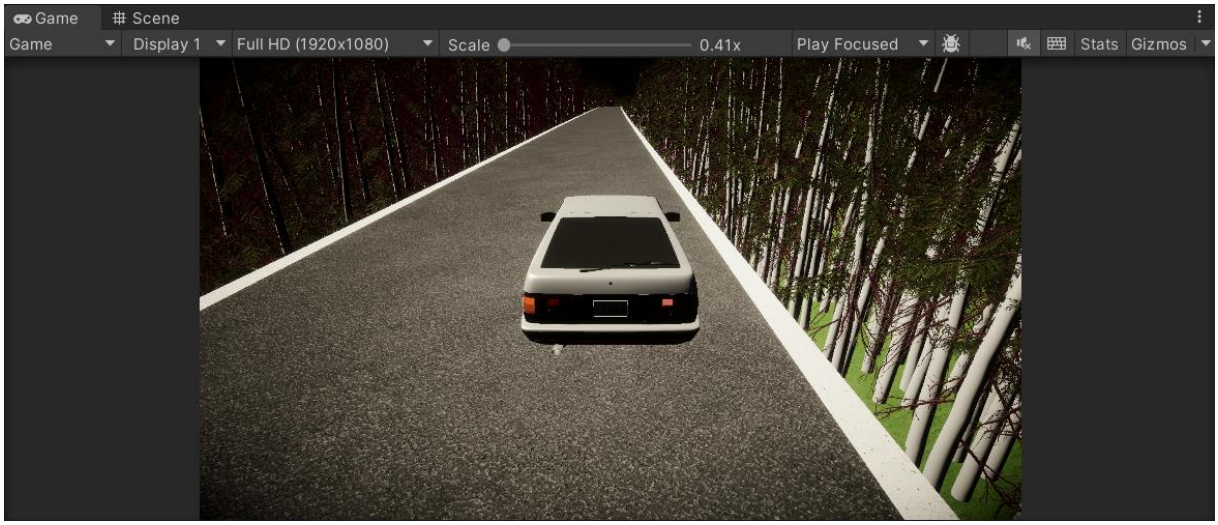
Slika 2-3 Scene view

- **Alatna traka (engl. *Toolbar*)** (slika 2-4) je traka koja se nalazi na vrhu. Sastoji se od dva dijela. Prvi dio je namijenjen za scenu (pomicanje, rotiranje, skaliranje), drugi dio za *game view* (*play*, *pause* i *step* botuni).



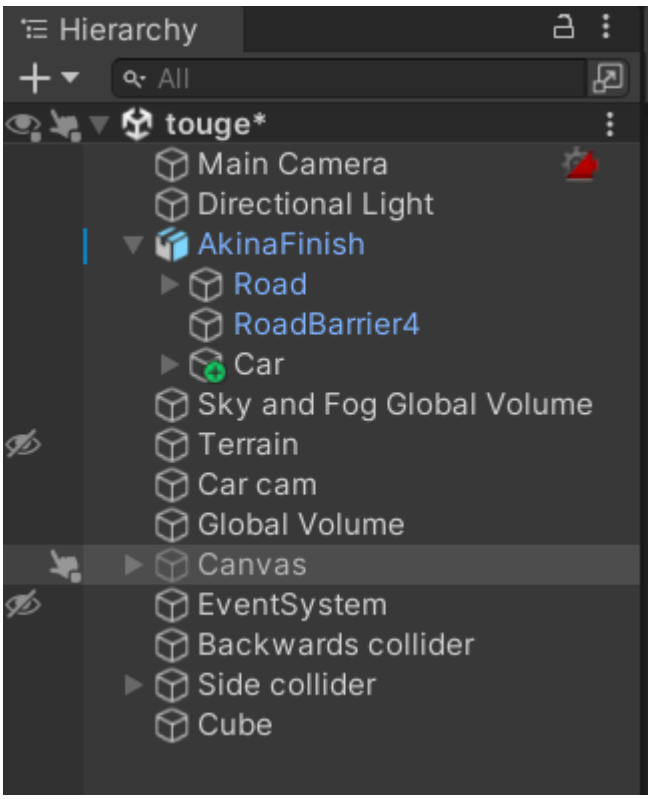
Slika 2-4 Toolbar

- ***Game view*** (slika 2-5) prikazuje ono što kamera vidi odnosno ono što igrač vidi kad igra igricu. *Game view* jest igrica, ali sa nepotpunim svojstvima zbog testiranja igrice. Testiranje pokrećemo sa *play* botunom na *toolbaru*.



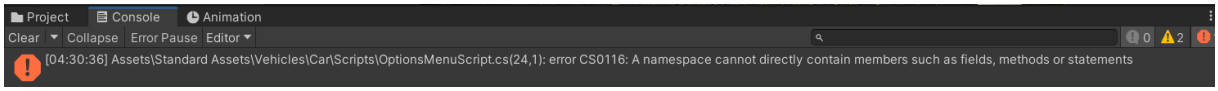
Slika 2-5 Game view

- **Hijerarhijski pogled (engl. *Hierarchy view*)** (slika 2-6) je hijerarhijska lista objekata u sceni. Služi da bismo lakše pronašli željeni objekt u sceni. U hijerarhiji imamo Roditelj (engl. *Parent)* – Dijete (engl. *Child*) sistem što znači da možemo umetnuti objekt u drugi objekt. Objekt u koji stavljamo drugi je *Parent* te on ima kontrolu nad umetnutim *Child* objektom. Ako naprimjer mičemo *Parent* miče se i *Child* objekt. Isto tako *Child* može biti istovremeno i *Parent* nekome drugome objektu.



Slika 2-6 Hierarchy view

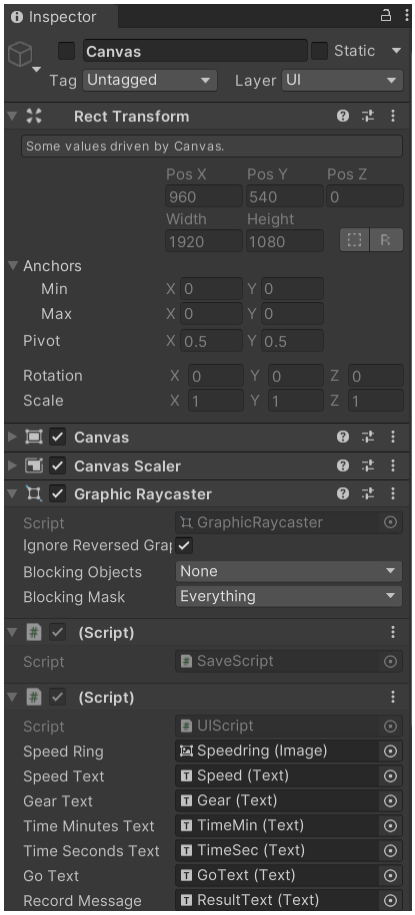
- **Konzola (engl. *Console view*)** (slika 2-7) je konzola koja prikazuje poruke, greške i upozorenja u izradi igrice.



Slika 2-7 Console view

- **Inspektor (engl. *Inspector view*)** (slika 2-8) je mjesto gdje se nalaze detaljne informacije o objektu (pozicija, rotacija, fizika, *collider* i dr.) koje možemo mijenjati. U njemu dodajemo komponente objektima. Isto tako Unity nam

omogućava da možemo kroz inspektor modficirati varijable skripte koja se nalazi u inspektoru. Za svaku vrstu materijala možemo neke dijelove modficirati po potrebi. Na vrhu inspektora imamo mjesta za upisati ime objekta, ime možemo mijenjati i u *hierarchy view*-u. Zatim možemo deaktivirati objekte u sceni, ali se oni ne brišu iz *hierarchy view*-a. Ovo koristimo kad objekt želimo aktivirati u nekom određenom trenutku (naprimjer kad stisnemo *escape* da se tad aktivira *pause* objekt). Možemo dodavati oznake (engl. *tagove)* objektima, *tagovi* su *stringovi* koji služe za prepoznavanje objekata u skriptama. Na dnu inspektora imamo botun *Add Component* kojim dodajemo komponente.



Slika 2-8 Inspector view

2.2. Skriptiranje

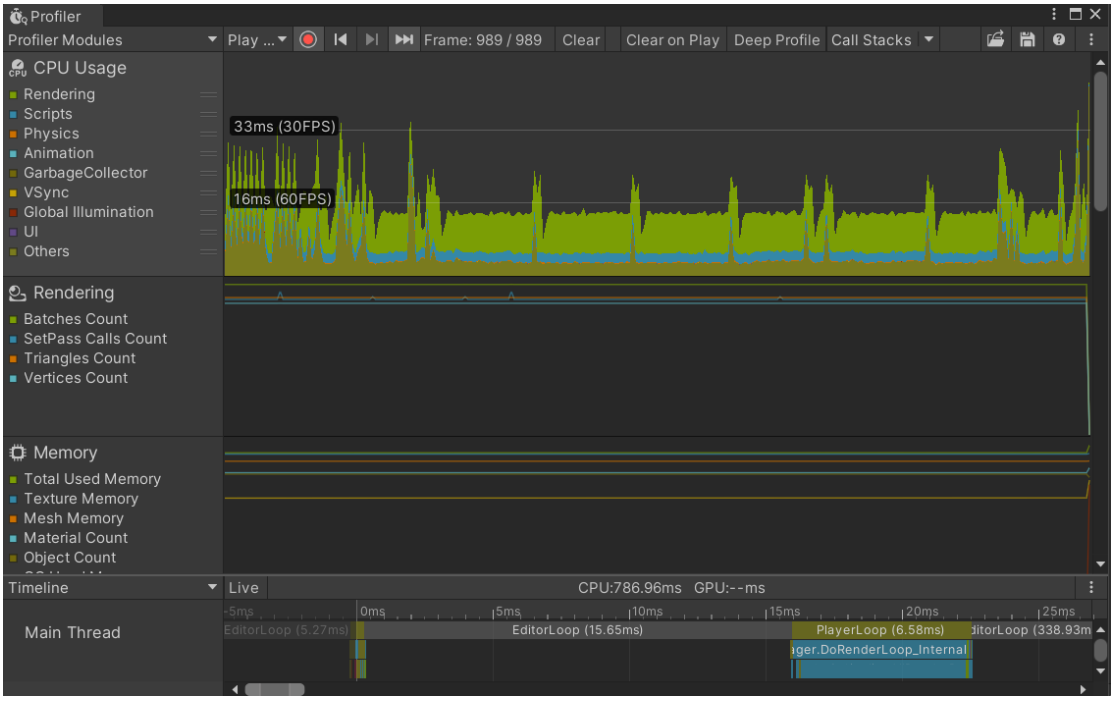
Za programiranje u Unityju se koristi C#. Prije verzije 2018 Unity je još podržavao i JavaScript, koja je nazvana UnityScript jer je to modificirani JavaScript koj se potpuno razlikovao od JavaScripta za browsere. Kako se JavaScript nije puno koristio te je bio tu zbog početnika odlučili su ga maknuti u 2018 verziji. Sam Unity Editor je isprogramiran u C# stoga je u C# moguće kreirati vlastite komponente za Unity. C# je moćniji jezik od JavaScripta jer programeru daje precizniju kontrolu. C# je objektno orijentiran programski jezik jer radi sve sa klasama i objektima. Skripte odnosno kodovi koje napišemo jesu komponente koje su spremljene u mapi *Assets* te možemo pristupiti preko *project view*a. Kad napišemo skriptu mi nju moramo staviti u scenu da bi se ona izvršila. Ako je skripta vezana za fizičke objekte kao što kretanje igrača tada skriptu dodajemo objektu koji će predstavljati igrača u inspektoru. U slučaju kada napišemo skriptu koja se odnosni na logiku ili tok igrice tada skriptu dodajemo u prazni objekt. Na ispisu 1 Skripta glavnog izbornika igre je prikazana skripta za glavni izbornik igre.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MainMenuScript : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Application.Quit();
        }
        else if (Input.GetKeyDown(KeyCode.Return))
        {
            SceneManager.LoadScene(1);
        }
        else if(Input.GetKeyDown(KeyCode.Space))
        {
            SceneManager.LoadScene(1);
        }
    }
}
```

Ispis 1 Skripta glavnog izbornika igre

2.3. Optimizacija

O optimizaciji ovise performanse i zahtijevanja same igre i zbog toga je vrlo važna. Unity ima integrirani alat *Profiler* (Slika 2-9 Profiler) koji služi za praćenje koliko je vremena oduzima pojedini dio igrice. Najčešće za optimizaciju se prepravljaju dijelovi kôda kao što su petlje, ti dijelovi se rješavaju na drugačiji način koji će izbjeći veliki broj uvjeta, ugniježđene petlje ili korištenjem manjim brojem varijabli. Mogu se optimizirati 3D modeli i animacije. Za ući u *profiler* se koristi prečica Ctrl + 7.



Slika 2-9 Profiler

3. Blender

Blender je besplatan i *open source* 3D program za modeliranje, animaciju, renderiranje, postprodukciju i više. S moćnim i sveobuhvatnim mogućnostima, Blender omogućuje korisnicima stvaranje složenih 3D sadržaja, uključujući modele, animacije, vizualne efekte i interaktivna iskustva. Pružajući alate za modeliranje, teksturiranje, *rigging*, animaciju i druge aspekte, Blender je popularan među umjetnicima, animatorima, dizajnerima i entuzijastima koji se bave 3D grafikom. Trenutačna stabilna verzija je 3.6 (LTS).

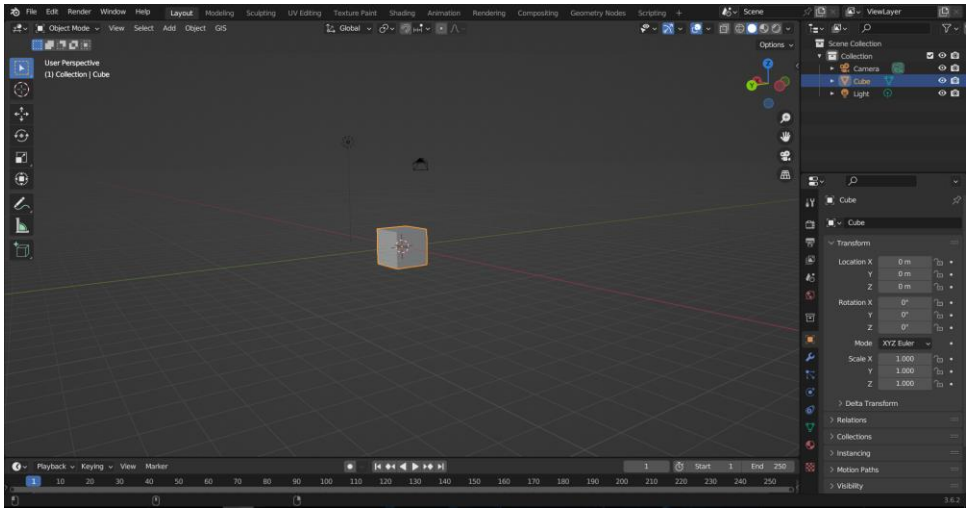
Prednosti Blendera su: besplatna uporaba, *open source*, mnoštvo resursa kao što su tutorijali i aktivna zajednica, proceduralni tijek rada i ciklus razvijanja.

Nedostaci su: Strma krivulja učenja, *bugovi*, nije industrijski standard jer većina kompanija najčešće imaju plaćene animacijske programe koji im se uklapaju u sistem i rade s drugim procesima.

3.1. Blender editor

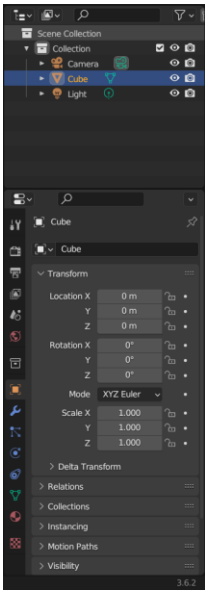
Blender *editor* je centralni radni prostor unutar Blendera, 3D programskog alata, u kojem korisnici mogu stvarati, uređivati i manipulirati 3D sadržajem. Blender *editor* sastoji se od niza panela, alata i prozora koji omogućavaju korisnicima da obavljaju različite zadatke, uključujući modeliranje, animaciju, teksturiranje, renderiranje i više. Nekoliko ključnih aspekata Blender *editora*:

- ***3D Viewport*** (Slika 3-1) Glavni prozor unutar Blender *editora* je *3D Viewport*, gdje se prikazuje scena u 3D okruženju. U ovom prozoru se može manipulirati objektima, izvoditi transformacije i raditi na modeliranju.



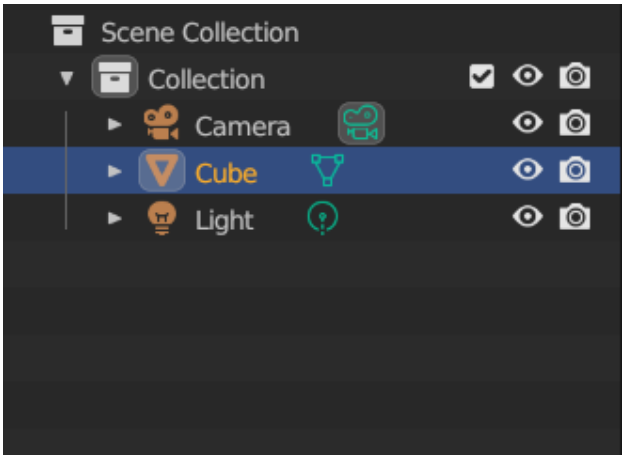
Slika 3-1 3D viewport

- ***Properties Panel*** (Slika 3-2) omogućava pristup i uređivanje svojstava odabranih objekata, materijala, svjetla i drugih elemenata unutar scene.



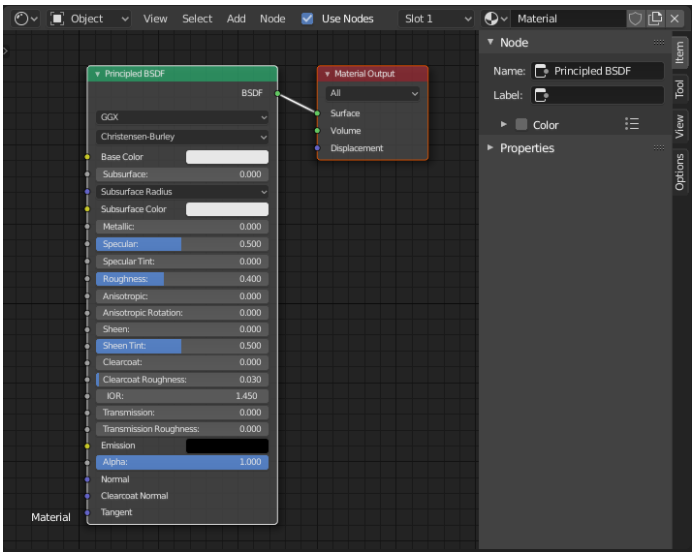
Slika 3-2 Properties panel

- ***Outliner*** (Slika 3-3) prikazuje hijerarhiju svih objekata u sceni, omogućavajući njihovo organiziranje, grupiranje i manipuliranje.



Slika 3-3 Outliner

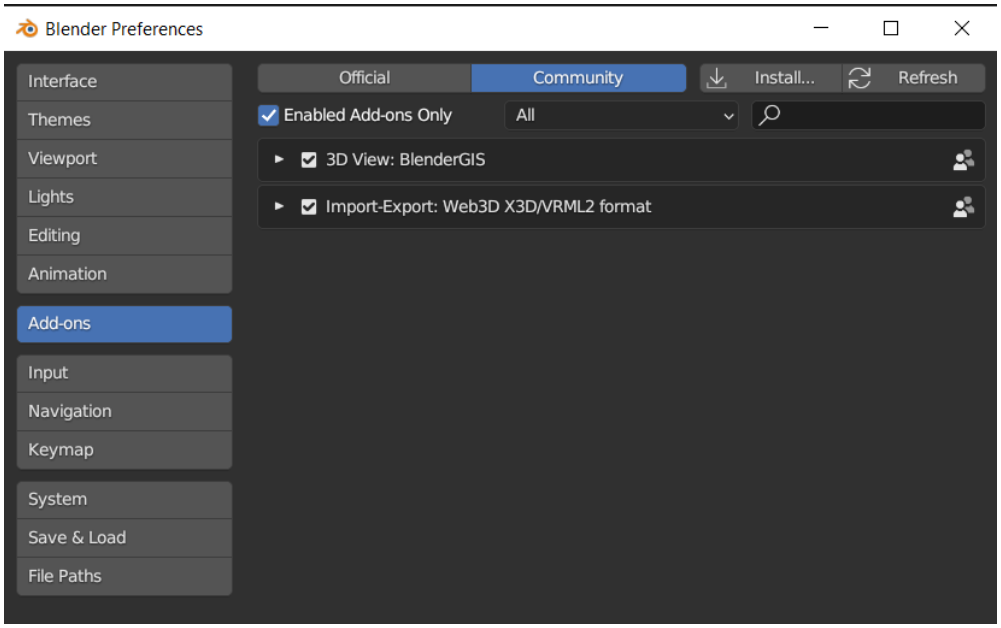
- ***Shader editor*** (Slika 3-4) se koristi za uređivanje materijala koji se koristе za renderiranje. Materijali koje koriste *Cycles* i *Eevee* definirani su pomoću stabla čvorova. Stoga je glavni prozor *Shader editora* uređivač čvorova.



Slika 3-4 Shader editor

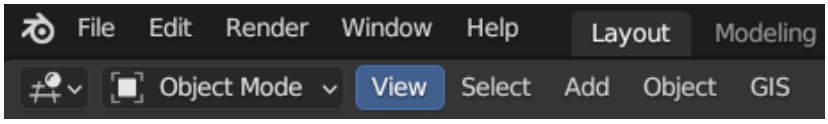
3.2. GIS (Geografski Informacijski Sustav) Alat

GIS (Geografski informacijski sustav) alat u Blenderu omogućava integraciju geografskih podataka i 3D modeliranja, što je korisno za kreiranje vizualizacija, simulacija i analiza koje kombiniraju prostorne informacije i 3D sadržaj. GIS software GIS se može preuzeti s linka [2]. Nakon što se preuzimanje završilo u Blenderu se GIS alat instalira preko *Edit – Preferences – Add-ons* i omogućuje se GIS alat(Slika 3-5).



Slika 3-5 GIS instalacija

Nakon instalacije se pojavi GIS opcija na *toolbaru*(Slika 3-6).



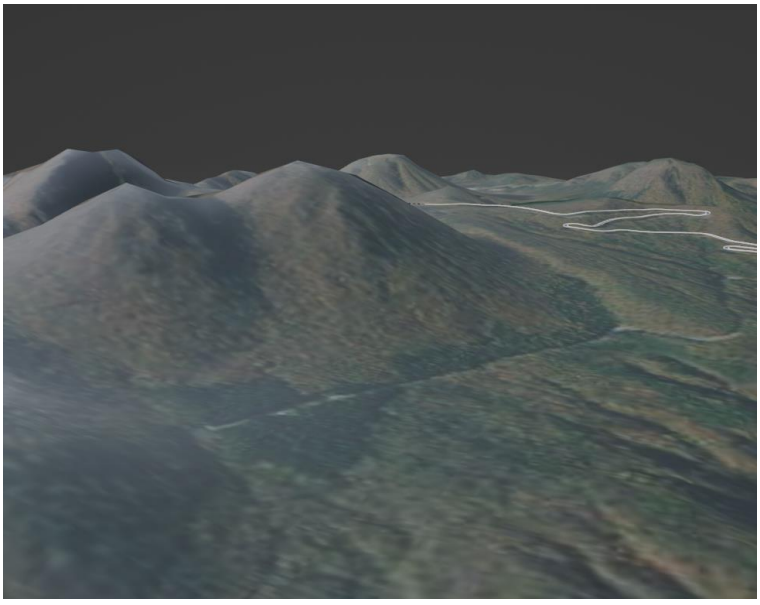
Slika 3-6 GIS toolbar

Iako Blender nije specifično GIS softver, postoji nekoliko načina kako se GIS podaci mogu koristiti u Blenderu:

- **Uvoz (engl. *Import*) GIS podataka** (Slika 3-7) Blender omogućava uvoz različitih formata geografskih podataka kao što su *Shapefile* (.shp), GIS rasteri i drugi formati. Dodaci kao što je "BlenderGIS" olakšavaju uvoz i manipulaciju ovih podataka unutar Blendera.

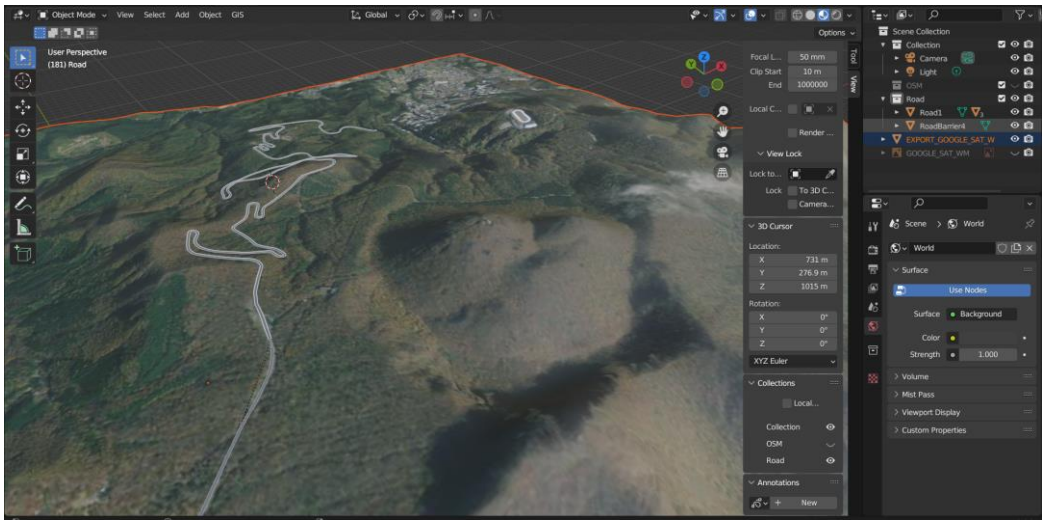
Slika 3-7 GIS podaci Zemlje

- **Kreiranje Terena:** Koristeći alate za modeliranje, možemo koristiti GIS visinske podatke za kreiranje realističnih terena i reljefa unutar Blendera(Slika 3-8).



Slika 3-8 Kreiranje terena

- In**tegracija u 3D Scene** (Slika 3-9) GIS podaci kao granice, ceste, rijeke i obale mogu se koristiti kao referenca za izradu realističnih 3D okruženja, gore desno na slici se može vidjeti outliner s objektima u trenutnoj sceni.



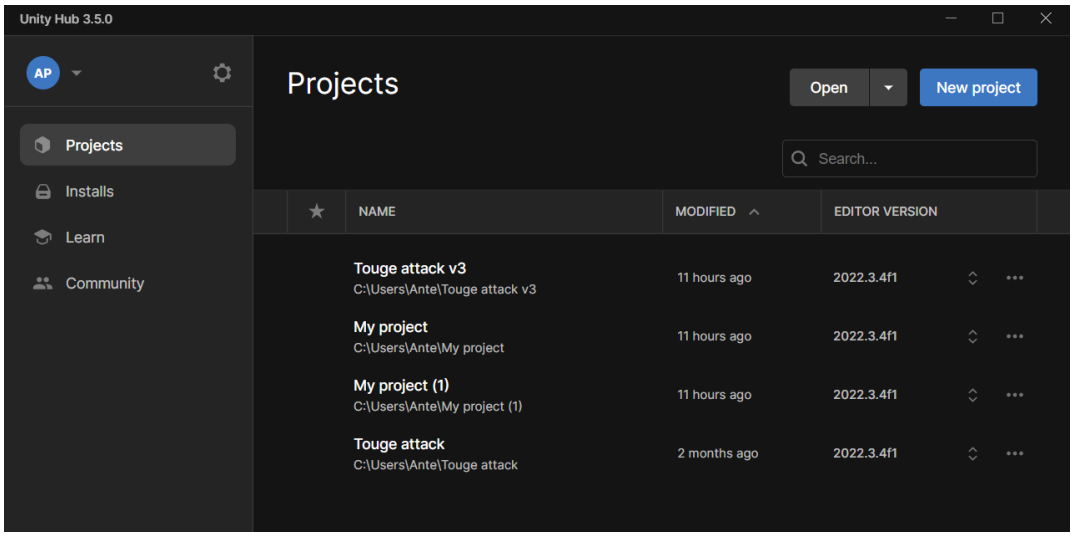
Slika 3-9 Integracija GIS-a

4. Unity – praktični dio

U ovom poglavlju su opisani svi dijelovi i procesi tokom izrade 3D igrice. Prvi korak je instalacija Unityja i Visual Studia. Videoigra ima tri scene: start izbornik, *racetrack* izbornik i samu utrku. Utrka je glavna scena jer u njoj igramo, start izbornik nam je prva scena prilikom pokretanja igrice i u njoj pokrećemo *racetrack* scenu te gasimo start izbornik scenu, dok druga scena *racetrack* izbornik omogućuje odabir staza i učitavanje treće scene odnosno utrke, kada utrka završi vraćamo se na drugo scenu odnosno race track izbornik za odabir staze po želji. Videoigra je 3D trkaća igra u trećem licu i cilj videoigre je oboriti rekorde koji su trenutno postavljeni na stazama u pravom životu i upoznavanje sa stazom u slučaju ako se bude prisustvovalo događajima u stvarnom životu.

4.1. Instalacija Unityja

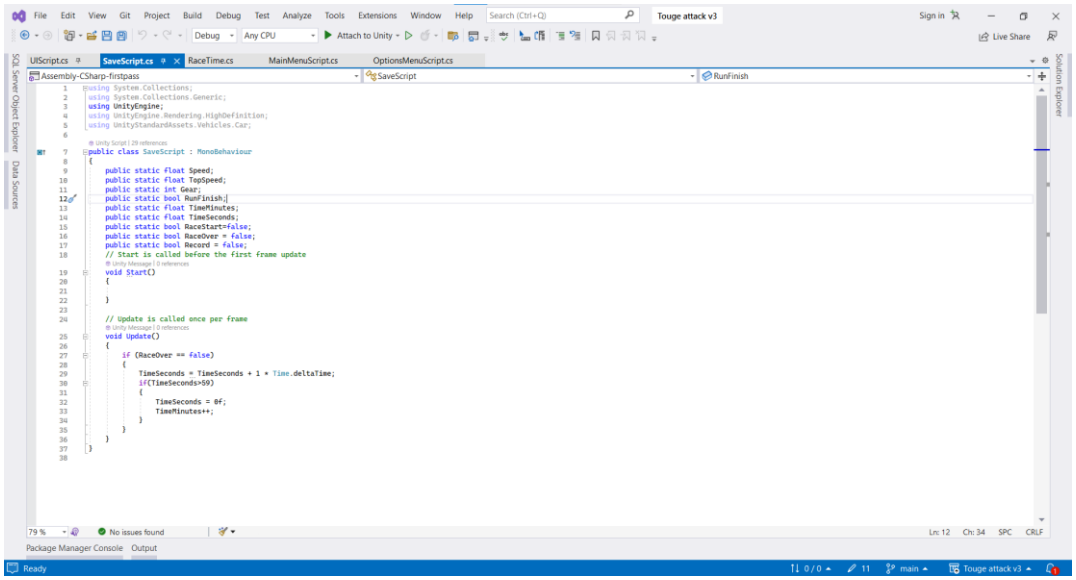
Unity je razvio takozvani Unity *Hub* (Slika 4-1), aplikaciju koja služi za instaliranje Unityja u raznim verzijama. Isto tako Unity *Hub* omogućava pristupanju svih naših Unityjevih projekata i mogućnost njihovih pokretanja u različitim verzijama koje su instalirane na našem sustavu. Kada se instalira i pokrene Unity *Hub*, preko *Installs* izbornika se vide zadnje verzije Unityja te se tu preuzima Unity *game engine*.



Slika 4-1 Unity Hub

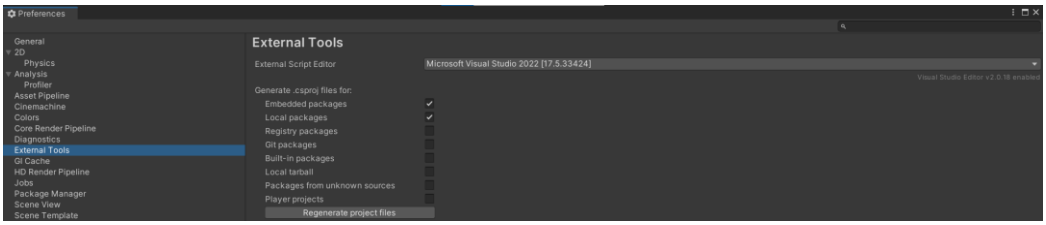
4.2. Instalacija Visual Studia

Visual Studio (Slika 4-2) je Microsoftov IDE (Integrated Development *Enviroment*) koji sadrži sve potrebne alate za pisanje kôda, kompajliranje i otklanjanje grešaka. Visual Studio je jako kompleksan softver, koristi se za pisanje kôda odnosno programiranje. Koristi se besplatna verzija *Community* koja je dostupna na linku [3].



Slika 4-2 Visual Studio

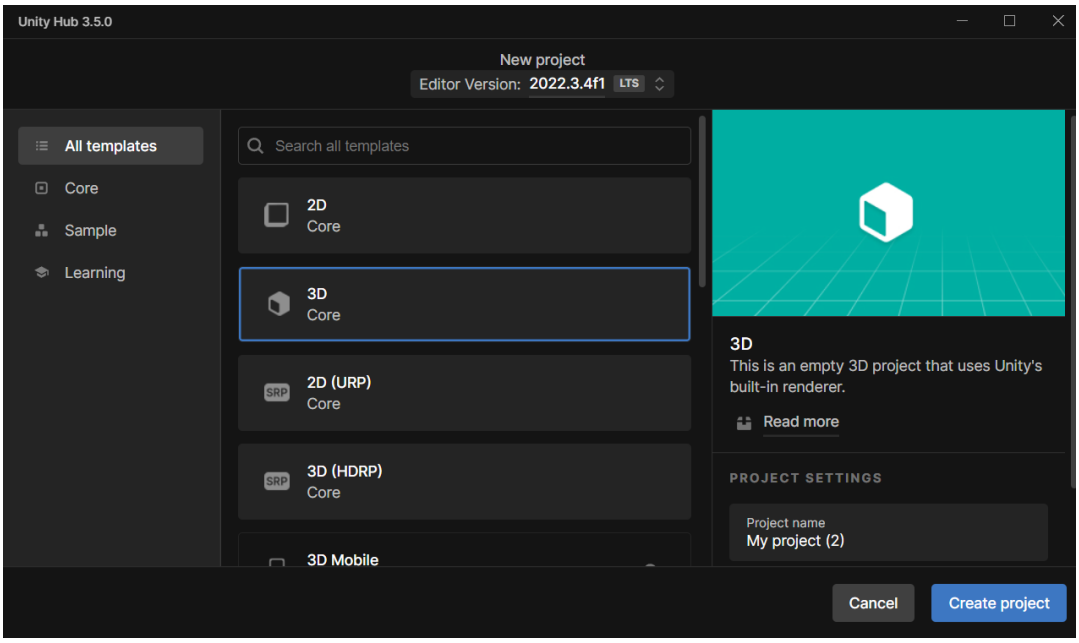
Ukoliko Unity automatski ne prepozna Viusal Studio potrebno je ručno namjestiti na Visual Studio, u Unityju se mijenja preko *Edit - Preferences - External Tools - External Script Editor* gdje se odabere Visual Studio, a ako nam Visual Studio nije ponuđen, potrebno je pronaći njegovu *exe* datoteku u *browse* izborniku (Slika 4-3).



Slika 4-3 Unity Preferences

4.3. Kreiranje projekta

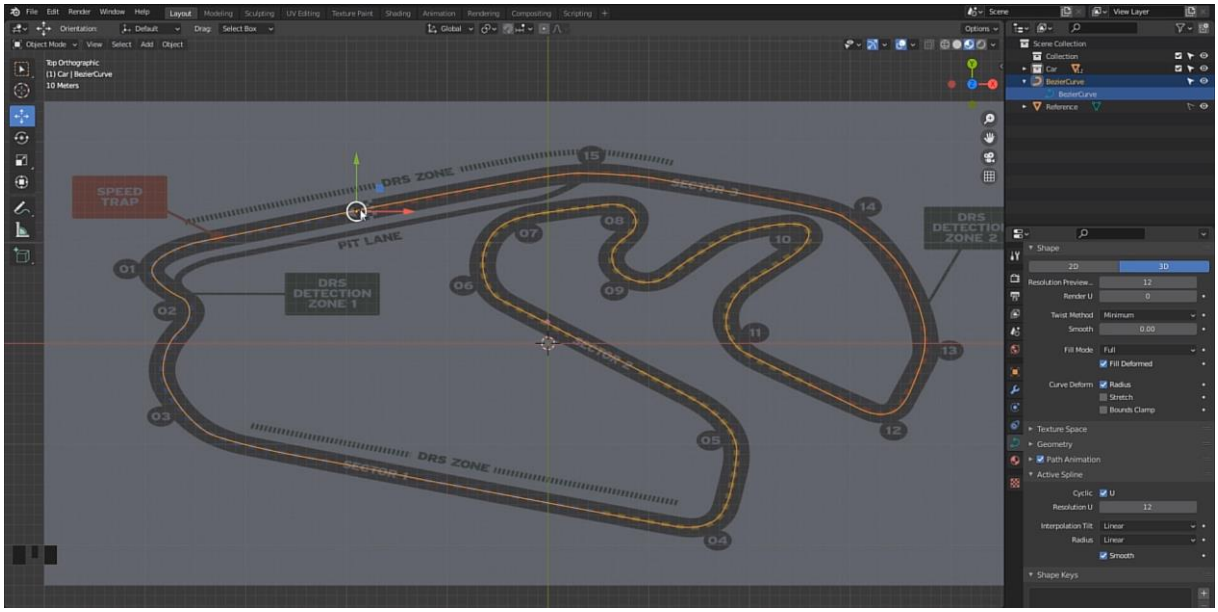
Projekt je igrica koju se razvija i svi potrebni materijali koji se budu koristili će biti u projektu. Da bi se kreirao novi projekt u Unity *Hubu* odabire se *Create project* u donjem desnom kutu (Slika 4-4), odabere se ime projekta, verzija Unityja, lokacija projekta i 3D jer se radi 3D igrica. Nakon što je projekt kreiran Unity se pokreće sa osnovnom scenom.



Slika 4-4 Kreiranje projekta

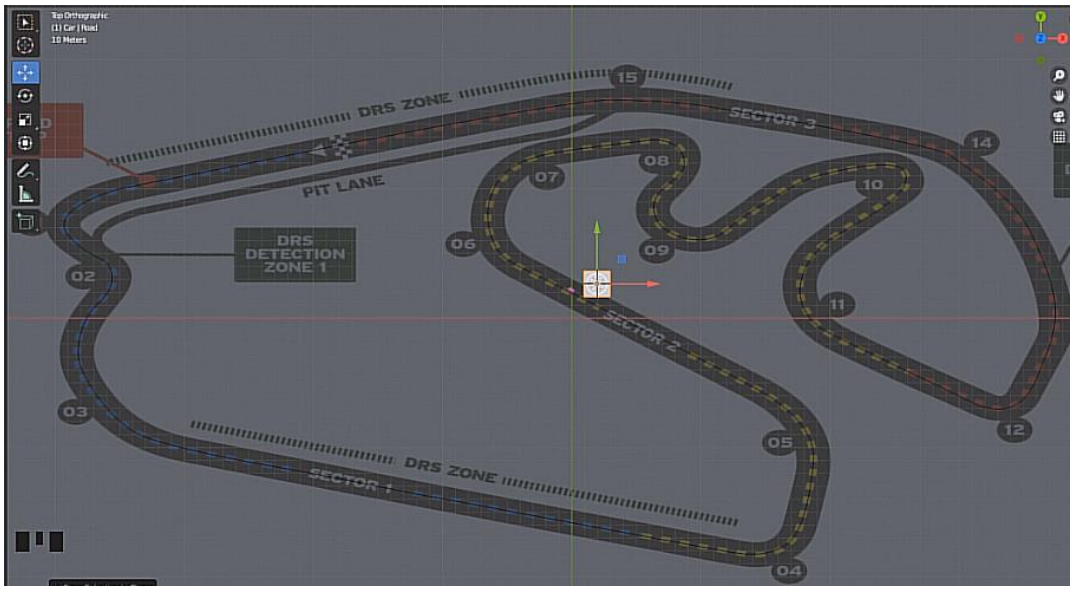
4.4. Dizajn staze

Dizajn staze se radi u Blenderu pomoću GIS alata da se postigne što realističniji izgled terena i raspored (engl. *layout)* cijele mape. Kada je tek dobiven izrezani dio GIS mape sa svom infrastrukturom, nastaje problem jer su ceste izvedene u obliku linija i treba ih napraviti u 3D plohe (engl. *plane*) koje bi predstavljale cestu. Taj problem se rješava tako da sve linije koje predstavljaju željene dijelove ceste pretvorimo u Bezierove krivulje (Slika 4-5).



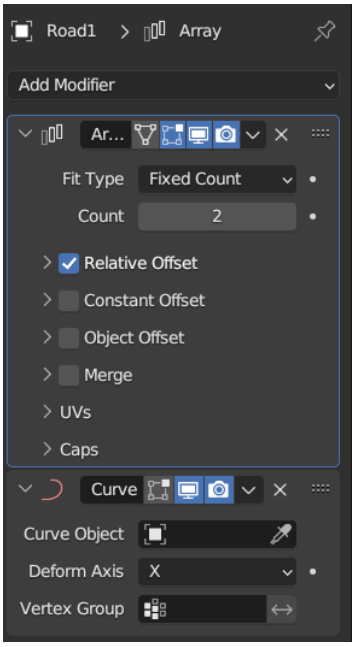
Slika 4-5 Cesta kao Bezierova krivulja

Zatim se kursor postavlja (crveno bijeli krug) u centar samog objekta u ovom slučaju ceste na način da odaberemo *toolbar* Object – Set *Origin* – *Origin* to *Geometry*, zatim *shift + s – Cursor to origin* da postavimo kursor u centar objekta. Zatim *plane* koji bi predstavljao cestu označimo kursorom i namjestimo ga u centar objekta tj. na mjesto kursora(*shift + s – Selection to cursor*) (Slika 4-6).



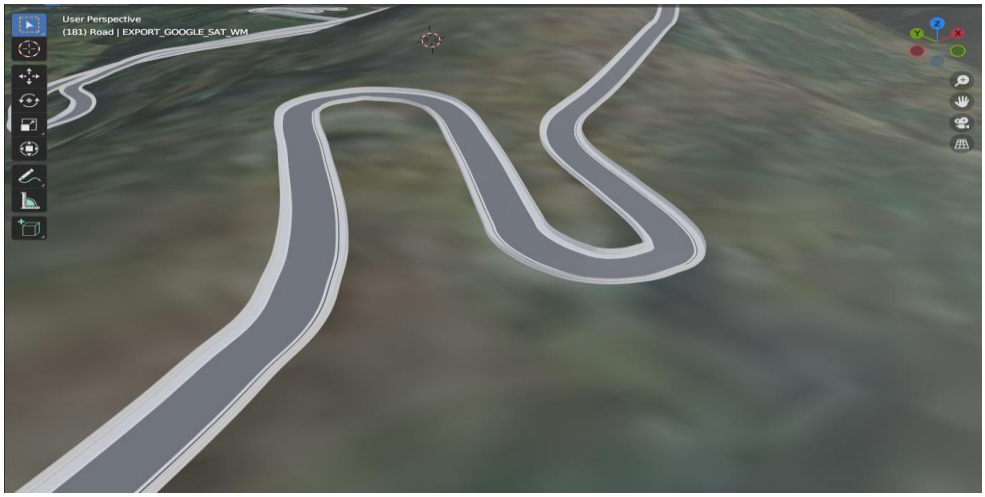
Slika 4-6 Postavljanje ceste u centar objekta

Nakon toga objektu ceste se dodaje modifikator (engl. modifier) polja (engl. array) i modifikator krivulje (engl.curve) (Slika 4-7).



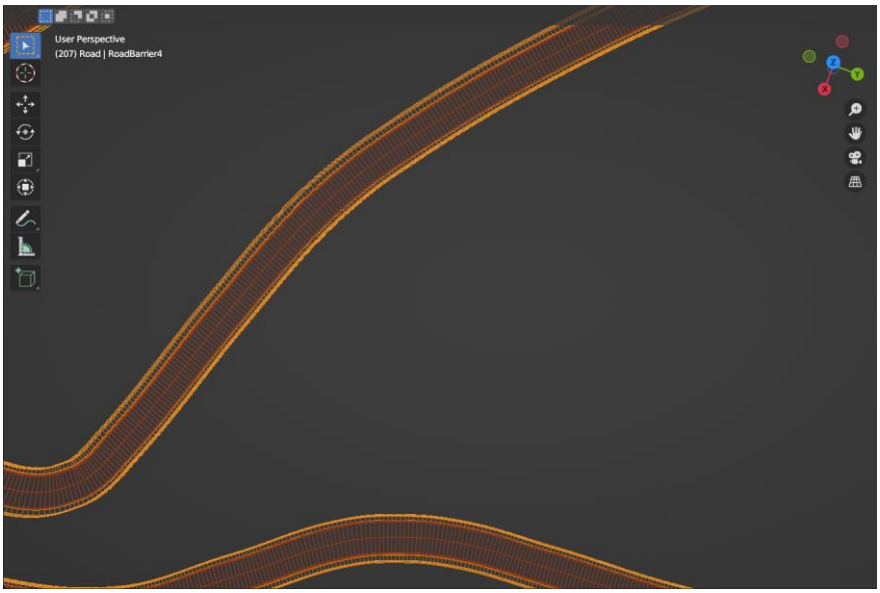
Slika 4-7 Modifier property

Na *curve modifieru* dodajemo *curve* objekt što je u ovom slučaju Bezierova krivulja, a na *array* modifieru određujemo kolika će nam biti duljina ceste tako što povećavamo polje brojač (engl. count) dok cijela Bezierova krivulja ne bude cesta (Slika 4-8 Bezierova krivulja pretvorena u cestu.



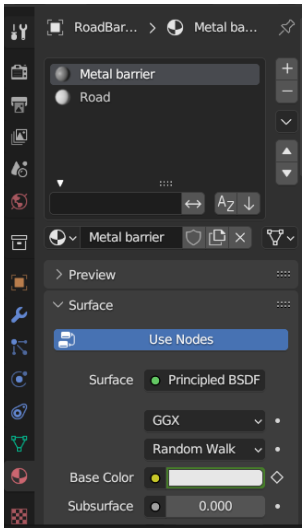
Slika 4-8 Bezierova krivulja pretvorena u cestu

U *wireframe* modu se podijeli objekt ceste na više manjih segmenata (lica) kojima se može dati drukčiji materijal (npr. bijela linija na rubovima ceste i metalne barijere) i da cesta koja prati krivulju bude što glađa i što realističnija(Slika 4-9).



Slika 4-9 Segmentirana cesta u wireframe modu

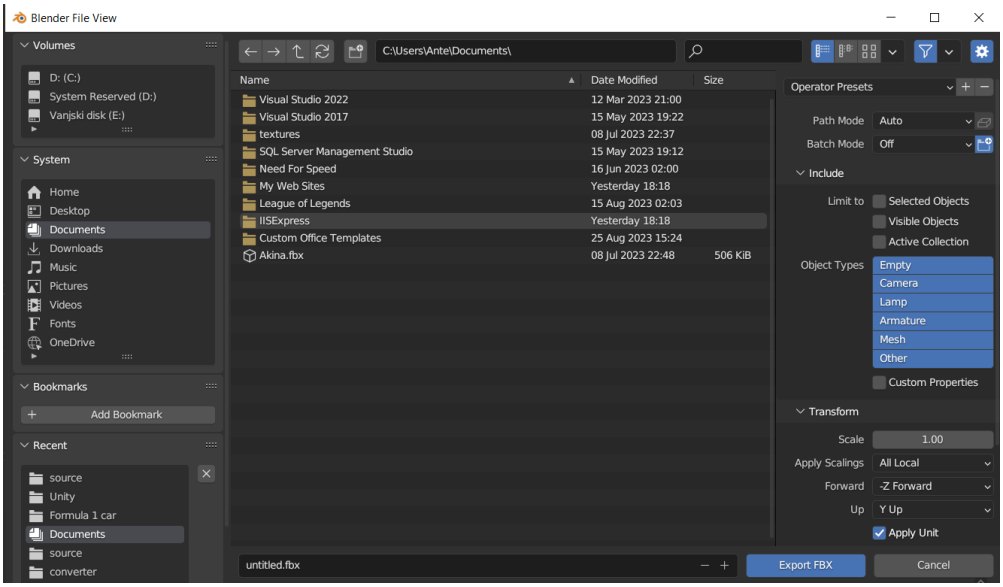
Također zbog segmentacije ceste se može odabrati pojedinačni segment i dodati mu novi materijal i teksturu tog materijala (Slika 4-10).



Slika 4-10 Material property

4.5. Exportanje datoteka u Unity

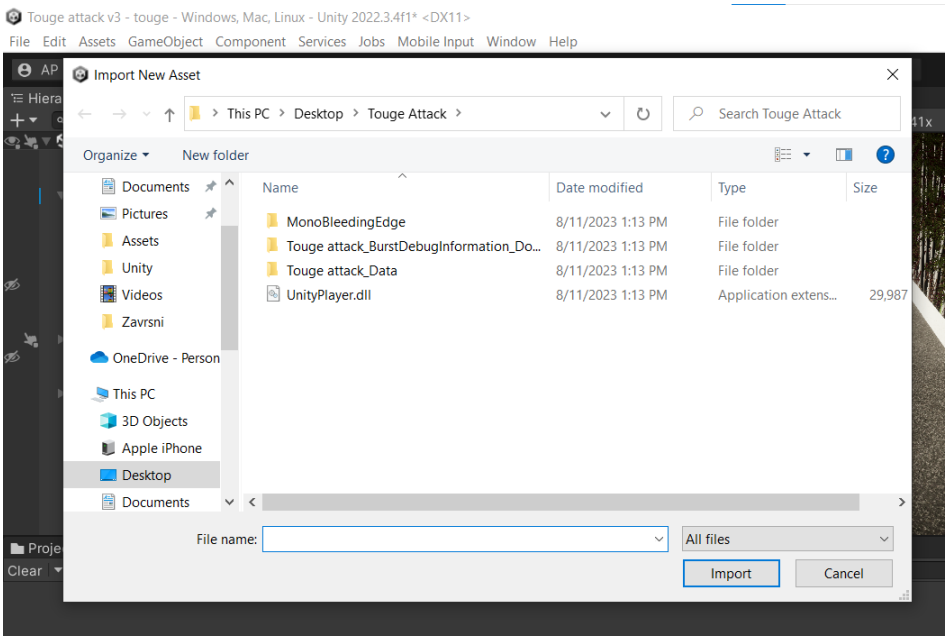
Unity izvorno uvozi Blender datoteke. Ovo radi u pozadini pomoću Blender FBX izvoznika (engl. *exportera*). Ako se koristi FBX format za *export* Blender modela u Unity, nekoliko stvari će biti zapakirane osim 3D modela. Za *export* modela u Blenderu idemo na File – Export – FBX i dobijemo *pop-up* prozor koji nam pokazuje gdje ćemo spremiti datoteku i imamo mogućnosti odabrati što želimo uključiti u datoteku koju *exportamo* (Slika 4-11).



Slika 4-11 Export modela

4.6. Importanje datoteka u Unity

Za *import* datoteka u Unity je dovoljno povući i pustiti datoteku u *project window* Unity *editora* ili preko *Import New Asset* naredbe (Slika 4-12).

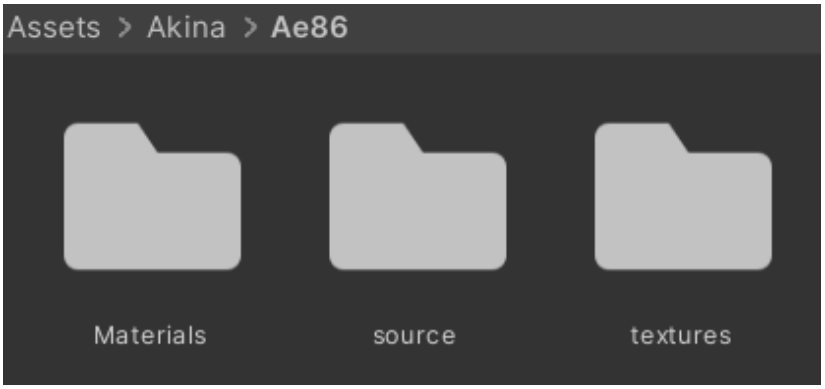


Slika 4-12 Import sredstava

Datoteke modela mogu sadržavati razne podatke, kao što su *mesh*, oprema za animaciju i isječci, materijali i teksture. Primarni format modela za Unity je FBX format. Bez obzira na podatke koji će se izvući iz model datoteke, proces je isti:

1. Otvoriti *Project* view i *Inspector* da se vidi oboje u isto vrijeme
2. Odabrati datoteku modela kojeg se importa iz *Asset* mape iz *Project windowa*. *Import* postavke se u inspektoru pokazujući na model.

U *project view*-u kôd *importanog* modela su još najčešće tri mape unutar mape samog modela, te mape su materijali, teksture i izvorni model (Slika 4-13).



Slika 4-13 Mape importane datoteke

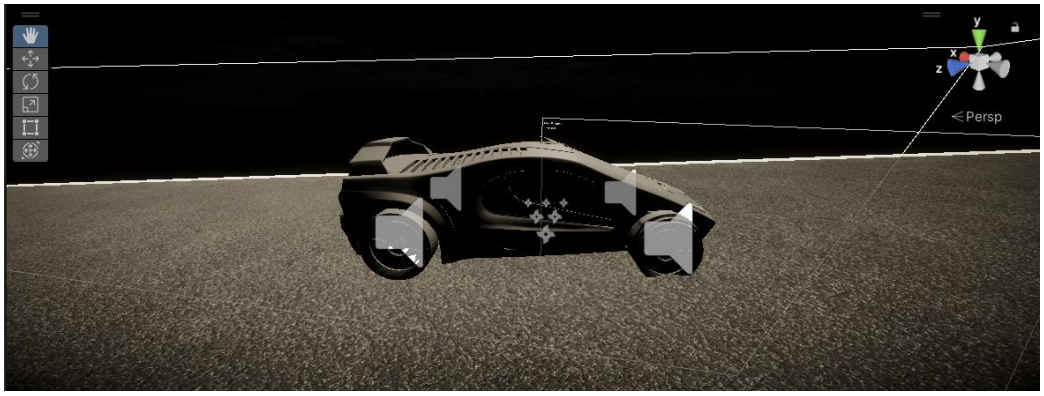
Da bi se model nakon *importanja* dodao u scenu treba ga mišem povući iz *project* prozora u scenu.

4.7. Dodavanje modela automobila u Unity

Dodavanjem paketa *Standard Assets* iz Unity trgovine sa linka [4] u projekt se dobije mogućnost korištenja *SkyCar* modela zajedno sa *SkyCar* kontrolerom koji je korišten kao kontroler za automobile (Ispis 2 Dio kôda car controller skripte). Nakon što je automobil postavljen na stazu skalira se na veličinu manju od jedne polovine ceste kako bi odgovarao jednoj cestovnoj traci u prometu(Slika 4-14).

```
private void Start()
      {
         m_WheelMeshLocalRotations = new Quaternion[4];
         for (int i = 0; i < 4; i++)
         {
            m_WheelMeshLocalRotations[i] = m_WheelMeshes[i].transform.localRotation;
         }
         m_WheelColliders[0].attachedRigidbody.centerOfMass = m_CentreOfMassOffset;

         m_MaxHandbrakeTorque = float.MaxValue;

         m_Rigidbody = GetComponent<Rigidbody>();
         m_CurrentTorque = m_FullTorqueOverAllWheels -
(m_TractionControl*m_FullTorqueOverAllWheels);
         SaveScript.TopSpeed = m_Topspeed;
      }

      private void Update()
      {
         SaveScript.Speed = CurrentSpeed;
         SaveScript.Gear = m_GearNum;

      }

      private void GearChanging()
      {
         float f = Mathf.Abs(CurrentSpeed/MaxSpeed);
         float upgearlimit = (1/(float) NoOfGears)*(m_GearNum + 1);
         float downgearlimit = (1/(float) NoOfGears)*m_GearNum;

         if (m_GearNum > 0 && f < downgearlimit)
         {
            m_GearNum--;
         }

         if (f > upgearlimit && (m_GearNum < (NoOfGears - 1)))
         {
            m_GearNum++;
         }
}
```

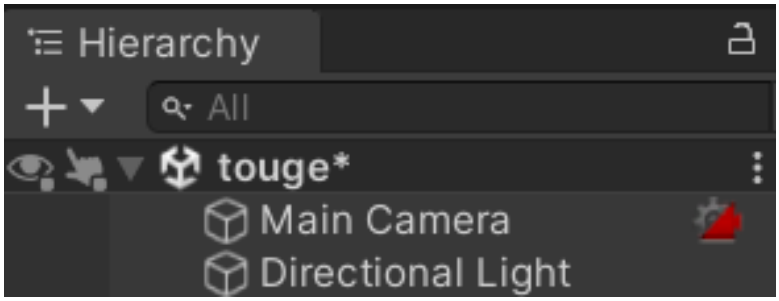
Ispis 2 Dio kôda car controller skripte



Slika 4-14 SkyCar model

Nakon skaliranja potrebno je dodati kameru koja prati igrača odnosno automobil. Za kameru se koristio *Cinemachine* paket koji se instalira preko alatne trake *Window – Package Manager – Cinemachine*.

Za kameru koja će pratiti automobil se koristila virtualna kamera kojoj se pristupi preko *GameObject – Cinemachine – Virtual Camera*. Nakon što je virtualna kamera dodana na glavnoj kameri se pojavi mala ikonica za *CinemachineBrain* (Slika 4-15).



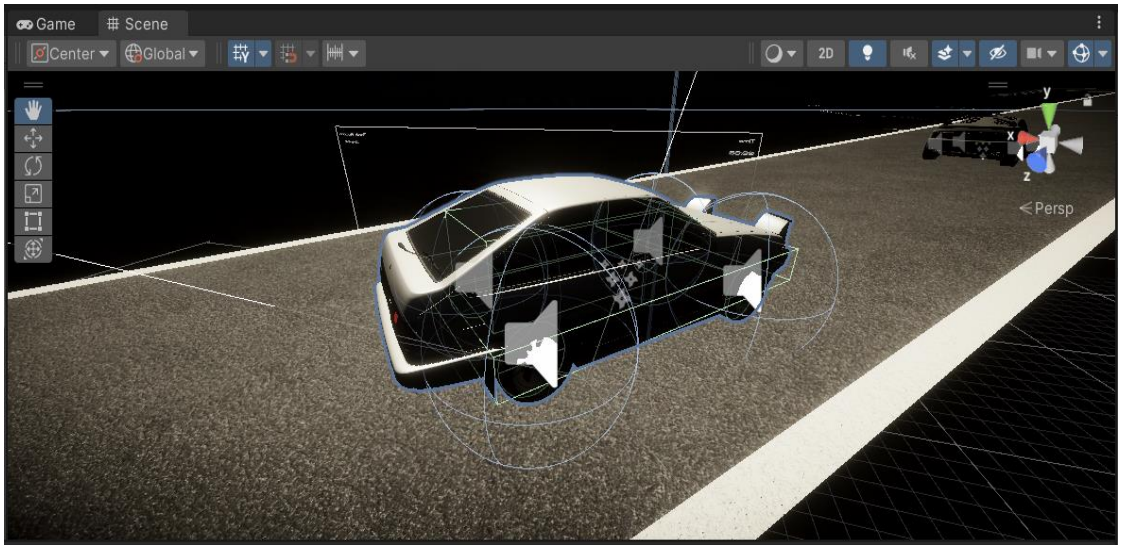
Slika 4-15 Cinemachine ikona

Nakon toga treba postaviti u inspektoru virtualne kamere da prati i gleda u automobil i također namjestiti pomak da bi se vidio automobil i staza iz trećeg lica (Slika 4-16).

Slika 4-16 Postavke kamere

4.8. Mijenjanje modela automobila

Za promijeniti model automobila nužno je da novi model ima tijelo (karoseriju) i četiri kotača, te nam za svaki kotač treba *wheel collider*. *Wheel collider* je poseban *collider* koji ima ugrađenu detekciju kolizije, fiziku kotača i model trenja gume koji se temelji na proklizavanju. Posebno je dizajniran za vozila s kotačima ali se može koristiti i za drukčije objekte. U Unity editoru *collideri* su najčešće vidljivi svijetlo zelenom bojom (Slika 4-17).



Slika 4-17 Novi model automobila s colliderima

4.9. Izrada brojača vremena

Cilj je izraditi brojač vremena (engl. *timer*) koji će početi brojati nakon što se pojavi signal za kretanje i *timer* će brojati sve dok igrač ne prođe cilj. Nakon toga uspoređuje se rekordno vrijeme i vrijeme koje je igrač postigao da bi se vidjelo je li rekord oboren ili ne.

Izrađuje se skripta u koju se pohranjuju svi bitni podatci u kojima će se raditi u drugim skriptama i *timer* same utrke (Ispis 3).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Rendering.HighDefinition;
using UnityStandardAssets.Vehicles.Car;

public class SaveScript : MonoBehaviour
{
    public static float Speed;
    public static float TopSpeed;
    public static int Gear;
    public static bool RunFinish;
    public static float TimeMinutes;
    public static float TimeSeconds;
    public static bool RaceStart=false;
    public static bool RaceOver = false;
    public static bool Record = false;

    void Update()
    {
        if (RaceOver == false)
        {
            TimeSeconds = TimeSeconds + 1 * Time.deltaTime;
            if(TimeSeconds>59)
            {
                TimeSeconds = 0f;
                TimeMinutes++;
            }
        }
    }
                                                    }
```

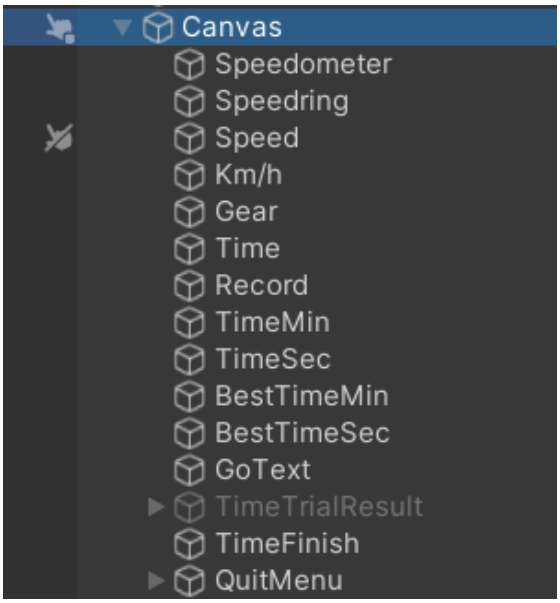
Ispis 3 Skripta za varijable i timer

Nakon skripte za *timer* se radi UI (*User Interface*) skripta koja će prikazivati trenutačnu brzinu vozila, brzinu na mjenjaču, ukupno vrijeme i sam *timer* (Ispis 4).

```
void Update()
  {

    float elapsedTime = Time.time - startTime;
    if (elapsedTime < displayDuration)
    {
      // Show the text
      GoText.enabled = true;
    }
    else
    {
      // Hide the text after the display duration
      GoText.enabled = false;
      SaveScript.RaceStart = true;
    }

    // Speedometer
    DisplaySpeed = SaveScript.Speed / SaveScript.TopSpeed;
    SpeedRing.fillAmount = DisplaySpeed;
    SpeedText.text = (Mathf.Round(SaveScript.Speed).ToString());
    GearText.text=(SaveScript.Gear + 1).ToString();


    //Minutes
    if (SaveScript.TimeMinutes <= 9)
    {
      TimeMinutesText.text= "0" + (Mathf.Round(SaveScript.TimeMinutes).ToString()) + ":";
    }
    else if (SaveScript.TimeMinutes >=10)
    {
      TimeMinutesText.text = (Mathf.Round(SaveScript.TimeMinutes).ToString()) + ":";
    }

    //Seconds
    if (SaveScript.TimeSeconds <= 9)
    {
      TimeSecondsText.text = "0" + (Mathf.Round(SaveScript.TimeSeconds).ToString());
    }
    else if (SaveScript.TimeSeconds >= 10)
    {
      TimeSecondsText.text = (Mathf.Round(SaveScript.TimeSeconds).ToString());
    }
```

Ispis 4 UI skripta

Za prikazati trenutnu brzinu i brzinu na mjenaču varijable se pretvaraju u *string*, a minute i sekunde se zaokružuju sa Mathf.Round i onda pretvaraju u *string*. Funkcija Mathf.Round zaokružuje vrijednost parametra na najbliži *integer*. Na *Canvas* elementu će biti prikazani svi UI elementi u *parent-child* odnosu gdje je *Canvas* element *parent*, a UI element *child* (Slika 4-18).



Slika 4-18 UI elementi

U donjem desnom kutu će biti prikazan brzinomjer i brzina na mjenjaču, u gornjem lijevom kutu će biti trenutno vrijeme, a u gornjem desnom rekord staze (Slika 4-19).

Slika 4-19 UI u game viewu

4.10. Ciljni collider

Nakon što je *timer* napravljen treba napraviti cilj. Cilj će biti *collider* koji gleda je li objekt taga „Player“ prošao kroz njega. Kada takav objekt prođe kroz taj collider igra završava i uspoređuje postavljeno vrijeme s rekordnim vremenom (Ispis 5).

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

public class FinishLine : MonoBehaviour
{

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            SaveScript.RaceOver = true;

        }
    }
}
```

Ispis 5 Skripta za ciljni collider

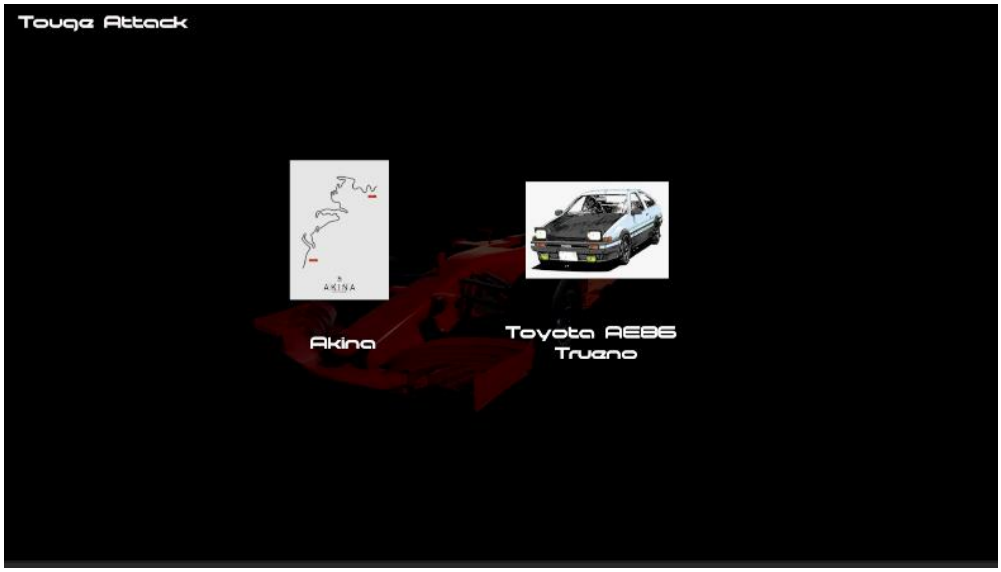
4.11. Start izbornik

Za kreiranje start izbornika potrebno je kreirati novu scenu. Scena će se sastojati od *Canvas* elementa na kojem će biti UI elementi poput naslova, pozadinske slike i natpisa (Slika 4-20). Start izbornik će se nadovezivati na glavni izbornik u kojem će se birati staza i nakon odabira staze se pokreće scena s utrkom.

Slika 4-20 Start izbornik

4.12. Glavni izbornik

Glavni izbornik se sastoji od Canvas elementa koji ima UI elemente kao djecu. U pozadini je slika automobila, a u gornjem lijevom kutu naslov s fokusom na stazu u sredini i ponuđeni automobil za tu stazu (Slika 4-21).

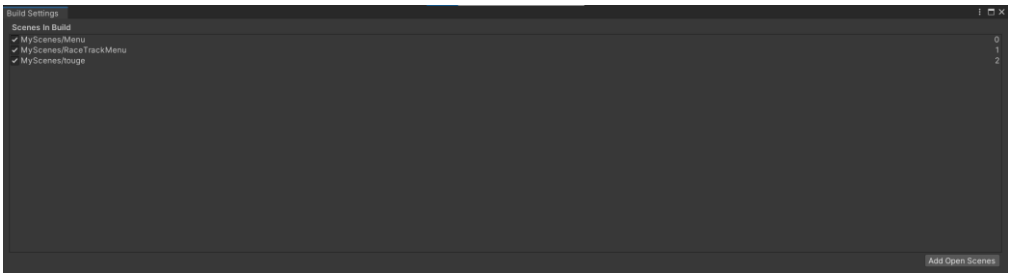


Slika 4-21 Glavni izbornik

Staza i automobil su prikazani slikom, a klikom na sliku staze se prikazuje izbornik s opcijom staze i automobila gdje će još uz navedene elementi bit i dva gumba u donjem lijevom i donjem desnom kutu, gumb u donjem lijevom kutu se vraća na odabir staza, a gumb u donjem desnom kutu počinje *loading screen* koji učitava scenu s utrkom (Slika 4-22).

Slika 4-22 Glavni izbornik s opcijama

Da bi se postigla željena tranzicija među scenama potrebno je scene dodati u build postavke. Scene su prikazane svojim datotečnim putom, no također su označene i brojevno počevši od nule. Za dodati scenu u *build* potrebno je ići na *File – Build Settings* (Slika 4-23).



Slika 4-23 Build Settings

4.13. Quit izbornik

Quit izbornik (Slika 4-24) je izbornik koji omogućava povratak na glavni izbornik usred utrke.

Slika 4-24 Quit izbornik

Samo treba postaviti komandu tj. tipku koja će pokretati izbornik usred scene s utrkom, ukoliko ne dođe do izlaženja iz igre prije cilja tada je generiran panel s gumbom (Slika 4-25)i rezultatom.

Slika 4-25 Continue panel

Quit izbornik i panel s rezultatom imaju istu funkciju, a ta funkcija je vraćanje na izbornik s utrkama, osim što panel s rezultatom prikazuje poruku sukladnu s rezultatom (Ispis 6).

```
if (SaveScript.RaceOver == true)
      {
         TimeTrialResult.SetActive(true);
         if (SaveScript.Record == true)
         {
            RecordMessage.text = "Congratulations, new track record";
         }
         else if (SaveScript.Record == false)
         {
            RecordMessage.text = "Try again";
         }
      }
      if(Input.GetKeyDown(KeyCode.Escape))
      {
         QuitPanel.SetActive(true);
      }
   }
   public void OnContinue()
   {
      SceneManager.LoadScene(1);
   }
   public void QuitClose()
   {
      QuitPanel.SetActive(false);
   }
}
```

Ispis 6 Kôd za prikaz rezultata i učitavanje glavnog izbornika

5. ZAKLJUČAK

U ovom završnom radu je istražen i demonstriran proces razvoja i implementacije *time trial* utrke unutar razvojnog okruženja Unity. Kroz analizu ključnih komponenata kao što su stvaranje staze, implementacija vozila, vremensko ograničenje i sustav za praćenje rezultata, stekne se razumijevanje kompleksnosti izrade ovakvog igračkog iskustva. Optimizacija performansi pokazala se neophodnom kako bi se osigurala fluidna igra, bez obzira na hardverske karakteristike računala.

U budućnosti, moguće je nadograditi ovu igru dodavanjem više različitih staza, vozila s različitim karakteristikama, te dodatnih modova natjecanja. Integracija naprednijih vizualnih efekata i tehničkih inovacija može dodatno poboljšati iskustvo igrača. Također, daljnja optimizacija i prilagodba igre različitim platformama omogućila bi širem broju igrača da uživaju u *time trial* utrkama.

Ovaj rad pruža uvid u proces razvoja igre žanra *time trial* utrka unutar Unity okruženja. Kroz kombinaciju tehničkih znanja i kreativnog pristupa, moguće je stvoriti izazovno, angažirajuće i zabavno iskustvo za igrače, čime se istovremeno doprinosi širem spektru raznolikosti unutar svijeta video igara.

LITERATURA

[1] *"Technologies, U.Download, Unity."* Dostupno na: *https://unity.com/download* (Pristupljeno: 01. Kolovoza 2023).

[2] *„Domlysz Domlysz/blendergis: Blender addons to make the bridge between Blender and geographic data, GitHub*." Dostupno na: https://github.com/domlysz/BlenderGIS (Pristupljeno: 06. Kolovoza 2023).

[3] *"Download visual studio tools - install free for Windows, mac, linux* (2023) *Visual Studio*." Dostupno na: https://visualstudio.microsoft.com/downloads/ (Pristupljeno: 12. Kolovoza 2023).

[4] *"The best assets for game making* (no date) *Unity Asset Store*." Dostupno na: https://assetstore.unity.com/ (Pristupljeno: 15. Kolovoza 2023).