Razvoj videoigre temeljene na utrkivanju u programskom alatu Unity

Lucić, Ivan

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Lucić

Razvoj videoigre temeljene na utrkivanju u programskom alatu Unity

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Lucić

Matični broj: 0016145723

Studij: Informacijski sustavi

Razvoj videoigre temeljene na utrkivanju u programskom alatu Unity

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Mario Konecki

Varaždin, rujan 2022.

Ivan Lucić

Izjava o izvornosti

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

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

Sažetak

Opisivanje alata Unity kao i žanr igara utrkivanja te navesti i ukratko opisati glavne predstavnike. Nakon toga kreirati igru utrkivanja, prikazati postupak njenog razvoja, kao i gotovu računalnu igru te opisati njene komponente i osnovne funkcionalnosti igre.

Ključne riječi: programiranje, C#, videoigra, unity, razvoj igre, utrkivanje, komponente

Sadržaj

1. Uvod

Kroz ovaj završni rad prikazat ću i pobliže objasniti proces izrade video igre s tematikom utrkivanja u programskom alatu Unity. Fokusirat ću se na objašnjavanje izrade igre u alatu korak po kora. Također ću u radu povezati naučeno na studiju te dotaknuti područja kolegija programiranja 1 i 2, algoritama te programskog inženjerstva. Uz pomoć naučenih koncepata iz programiranja 1, 2 i programskog inženjerstva iskoristit ću svoje znanje o objektno orijentiranom programiranju te ga primijeniti u programskom alatu Unity te svojim znanjem iz algoritama napraviti optimiziran kod sa što manjom vremenskom složenosti.

Sama motivacija za izbor ove teme kao završnog rada proizlazi iz mog interesa za programiranjem, te video igrama i autima. Cilj ovog završnog rada je prikazati moju kompetenciju kao programera za izvršavanjem složenog projektnog zadatka, ali i prikazati moje razumijevanje i znanje o funkcionalnosti osnovnih i složenih funkcija koje se nalaze što iza video igara, a što iza automobila. Žanr utrkivanja izabrao sam upravo zbog složenosti koja se vidi u pravom, fizičkom, svijetu te vlastitim iskustvom s druge strane igra utrkivanja. Također, kako bi ovaj završni rad, ali i sam projekt dobio na integritetu, koristio sam modele vlastite izrade, a video igra napravljena je u programskom jeziku C#.

2. Metode i tehnike rada

2.1. Izrada igre

Za glavnu izradu igre i motor koji će pokretati igru korišten je Unity. Unity je besplatan cross-platform alat za izradu 3D, 2D ili AR igrica. Sam alat je izrađen u C++ programskom jeziku, ali omogućava korisnicima izradu igri u pristupačnijem C# jeziku. [1]

Slika 1. Logo aplikacije unity

2.2. Programiranje

Za pisanje C# koda koristit će se Microsoft-ov alat Visual Studio. Verzija alata je Visual Studio Community koji je besplatan za studente te ću koristiti prijašnje iskustvo stečeno na kolegiju Programskog Inženjerstva.

2.3. Modeli

Modeli unutar igrice će biti izrađeni unutar programskog alata Blender. Besplatan i open-source alat za 3D grafiku, simulacije, virtualnu realnost i ostalo. Alat će nam koristiti za izradu 3D modela različitih objekata unutar igre poput auta i drveća za pozadinu.

Slika 2. Logo aplikacije Blender

2.4. Teksture

Za izradu i uređivanje tekstura igre koristit će se alati paint.net. Besplatan alat namijenjen za 2D grafiku s mogućnostima rada s različitim formatima.

3. Povijest

Povijest videoigrica utrkivanja je dugačka i opsežna te započinje u 1900-tima. Teme u igricama utrkivanja se protežu kroz cijeli spektar od realističnih simulacijskih igrica sve do arkadnih fantazija. Prva igrica utrkivanja se smatra mehanička igrica pod nazivom „Mechanical Yacht Race“ u kojoj bi igrači kontrolirali brod s komadićem užeta. Prvi igrač koji bi došao do plutače koja se nalazila na drugom kraju ploče bi bio pobjednik.

Prvom modernom igricom utrkivanja se smatra igra „Space Race“ napravljena od strane tvrtke Atari puštena u srpnju 1973. godine.[2] U igri dva igrača kontroliraju svemirski brod i utrkuju se s dna ekrana na vrh izbjegavajući asteroide koji se kreću po ekranu.

3.1. Glavni predstavnici danas

Svaki žanr utrkivanja ima svoje predstavnike, predstavnike koji su postali popularni čak izvan svoga kruga obožavatelja te jedan od takvih predstavnika je Richard Burns Rally igrica od developera Warthog Games. Richard Burns Rally (RBR) je inspirirala skoro sve današnje moderne rally igrice te sama ima veliku popularnost danas.

Jedan od glavnih predstavnika je developer Codemasters koji su također napravili rally igrice inspirirane od strane RBR poput DiRT franšize, ali jedan od najpopularnijih igrica od Codemastersa su njihove F1 igrice, od 2009 do danas svake godine Codemasters objavljuje novu F1 igricu koja svaki put postaje sve popularnija.

Project CARS 2 od developera Slightly Mad Studios, iRacing od iRacing Motorsport Simulations i Assetto Corsa Competizione od Kunos Simulazioni su još neki predstavnici simulacija na četiri kotača, jedan od glavnih predstavnika na dva kotača je MotoGP franšiza od Milestone developera.

Ako se okrenemo od simulacija na arkadne igre, predstavnici tamo bi bili Forza Horizon 5 koja bi bila neki miks arkade i simulacije te druge arkadne igrice poput TrackMania od Ubisofta i Need for Speed franšiza od developera Criterion Games. Igrica koja često zna biti zaboravljena zbog svog crtanog stila, a mora se spomenuti zbog popularnosti su Mario Kart igrice od Nintendoa.

4. Unity, Unity Hub i početak izrade igrice

4.1. Unity

Unity je kreiran u Danskoj te objavljen 2005. godine, postao popularan zbog demokratiziranja izazova izrade igrica, ali sam Unity se koristi i van industrije videoigara zbog svojih simulacija i animacija.[1] Unity pruža snažan grafički uređivač koji omogućava kreiranje modela i teksturiranje bez pisanja ijedne linije koda. Svaki objekt unutar Unityja se sadrži od više komponenti poput mreže (eng. *mesh*) za definiranje fizičkih karakteristika objekta, renderer (eng. *mesh renderer*) koji primjenjuje teksturu i svjetlost objekta te fizička komponenta koja pridodaje fizičku simulaciju objektu poput gravitacije i kolizije te se sam objekt može proširiti s vlastitim C# kodom. Unity je kreiran u C++ kodu, ali korisnici unutar Unitya mogu koristiti mnogo pristupačniji C# kod. Unity također dopušta korištenje drugih programskih jezika poput JavaScripta, ali je takav pristup mnogo rjeđi.

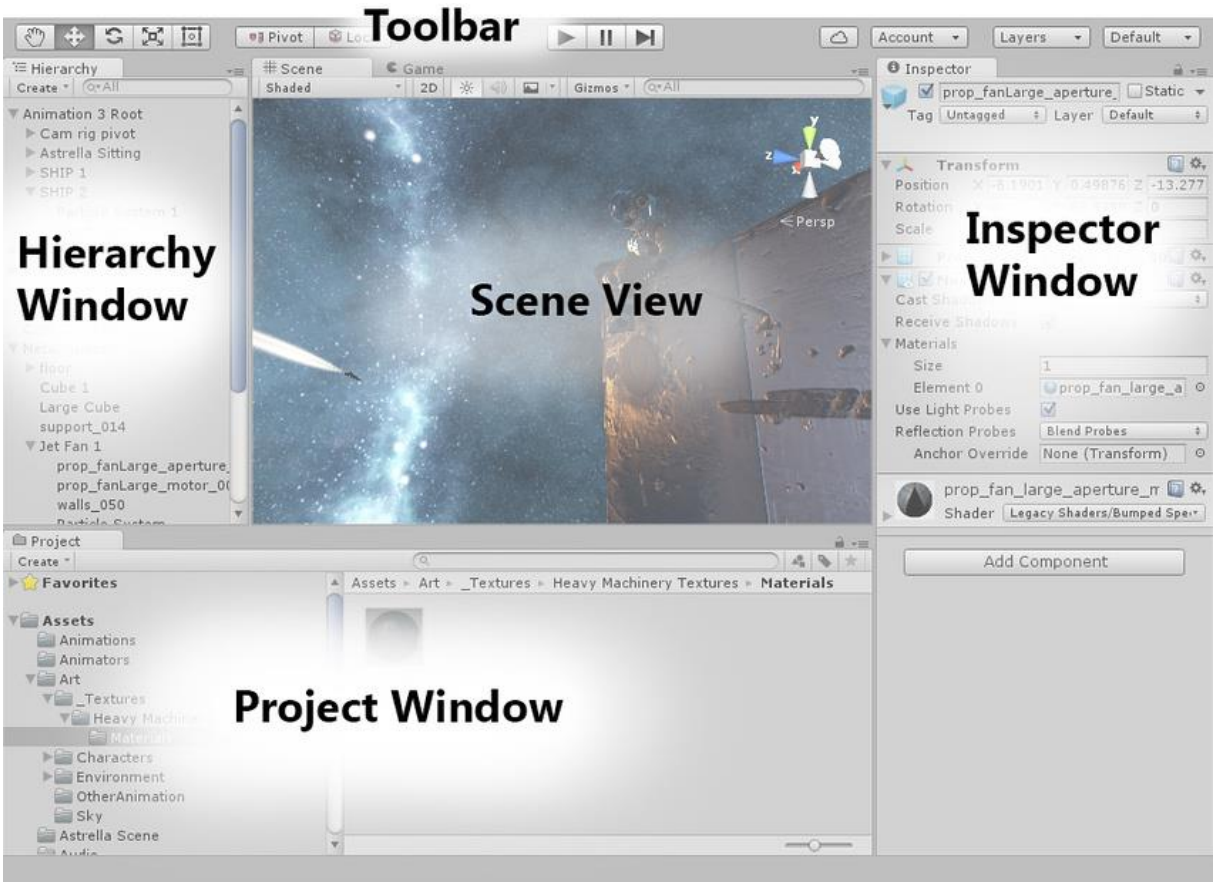
4.2. Unity hub

Kako bi započeli izradu igre treba se instalirati Unity Hub, programsko sučelje za interakciju s Unity motorom. Unutar Unity Huba postoje različite opcije poput izrade već spomenutih 3D, 2D, AR projekta te odabir fokusne platforme za odabrani projekt. Odabiranjem platforme nas ne limitira nužno za tu platformu te igru možemo kasnije urediti više da paše stilu druge željene platforme i buildati tako. Također se može eksperimentirati s projektima tako da se u 3D projektu dizajnira 2D igrica. Tako možemo postići unikatan izgled dubine koji ne bi mogli postići kada bi bili limitirani na dvije dimenzije. Naravno, suprotna mogućnost je moguća te se u 2D projektu može simulirati 3D prostor te kao popularan primjer možemo uzeti igricu Doom koja je pokrenula revoluciju 3D igrica. Takav tip igrica također se naziva 2.5D igrice, gdje se treća dimenzija simulira. Nakon otvaranja projekta otvara nam se Unity editor u kojem možemo započeti izradu igre.

4.3. Početak izrade igrice – upoznavanje s editorom

Glavni prostor uređivača sastoji se od alatne trake, prikaz scene, prozor inspektora (eng. *inspector window*), prozor hijerarhije i prozor projekta. Ti dijelovi se mogu preurediti, grupirati, odvojiti i usidriti.[3] Raspored prozora može ovisiti o osobnim preferencama i vrsti posla za svaku osobu. Zadani prozor od Unity editora daje najčešće korištene prozore za praktičan pristup te se ti isti prozori mogu prepoznati po nazivu na kartici. Prozor projekta

prikazuje biblioteku sredstava dostupnih za projekt te unutar prozora također se mogu uvoziti nova sredstva u projekt. Prikaz scene omogućavanje vizualno kretanje i uređivanje scene igre, a prozor hijerarhije je tekstualni prikaz svakog objekta u sceni te su inherentno ta dva prozora povezana. Inspektor nam omogućava uređivanje svojstava trenutno odabranog projekta dok alatna traka omogućuje pristup najvažnijim radnim značajkama. Sve prozore možemo vidjeti na slici 3.



Slika 3. Unity editor interface

Kako bi opisali no-code pristup u Unityu možemo kreirati plohu unutar toolbara i kreirati novi 3D objekt plohe. Uz plohu možemo dodati i objekt cilindra i kugle. Za uređivanje dodanog možemo pritisnuti y za micanje, rotiranje i skaliranje odabranog objekta. Kamera unutar scene nam daje prikaz onoga što igrač vidi. Pritiskom na pokretanje primjećujemo da se ništa ne događa. Odlaskom u inspektor i pod komponenta objekata kojih smo dodali možemo dodati novu komponentu pod nazivom rigid body. Pokretanjem sada možemo primijetiti da su naši objekti pod utjecajem gravitacije. Kako bi uredili objekt možemo kreirati novi materijal te ga samo povući na željeni objekt. Ovime smo napravili podlogu za igru bez uporabe koda.

4.4. Proces izrade igrice

Sam proces izrade igrice se može podijeliti u tri dijela: pretproizvodnju, proizvodnju i post proizvodnju. Pretproizvodnja je faza planiranja, proizvodnja je faza izrade, a post proizvodnja je faza održavanja.

4.4.1. Pretproizvodnja

Faza pretproizvodnje će definirati kako će naša igrica izgledati te kakvog će tipa biti. U ovoj fazi trebalo bi se isplanirati na kojoj će platformi igra biti izrađena, koliko vremena će biti alocirano za izradu igre, kakvo je tržište i komercijalizacija proizvoda. Ova faza uglavnom zauzima 20% totalne izrade igre. Pošto igrica bude izrađena samo od strane mene ja ću preuzimati sav posao. U ovoj fazi izgled igre se dosta može mijenjati.

4.4.2. Proizvodnja

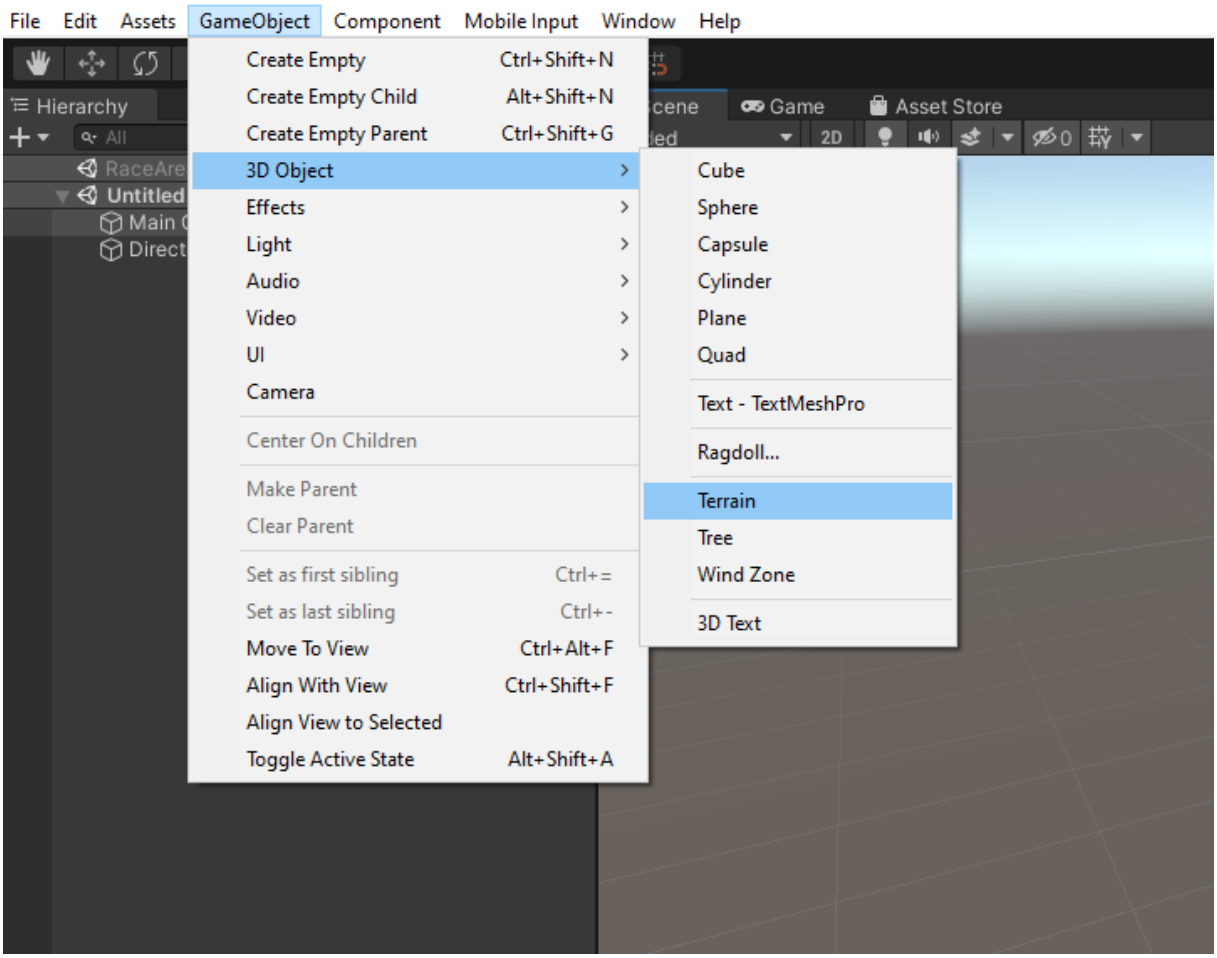
Faza proizvodnje je najduža faza u proizvodnji igre. U ovoj fazi se izrađuju materijali za igru, sama igra te dizajn igre. Dizajn igrice utrkivanja će biti low poly stil, radit ću sve modele osobno uz pomoć programa blendera, ali proces izrade modela neće biti opisan u radu pošto nije povezan direktno s Unityu game engineom. Programiranje će se odvijati unutar Visual Studija od Microsofta, te će se kod obrađivati u radu i biti kompiliran u Unityu.

4.4.3. Post proizvodnja

Faza post proizvodnje sadržava popravljanje bugova i poboljšavanje sitnih stvari unutar igrice i dizajna igrice. Primjer bi bio poboljšavanje rasporeda trake unutar igrice kako bi igricu činilo zabavnijom ili kako bi bolje pristajala uz tematiku igre. U ovoj bi fazi također bio nastavak marketinga za igru.

5. Izrada podloge, teksture i uvoz auta

Kreiranjem novog projekta stvara se scena, a u sceni je automatski dodan objekt glavne kamere te svijetlo unutar scene. Kako bi započeli s igricom utrkivanja dodat ćemo 3D objekt u nasu scenu. Prvi objekt koji ćemo dodati u igru će biti teren (*eng. Terrain*). U toolbaru se nalazi GameObject tab u kojem odabiremo 3D Object te Terrain kao što je prikazano na slici 4.

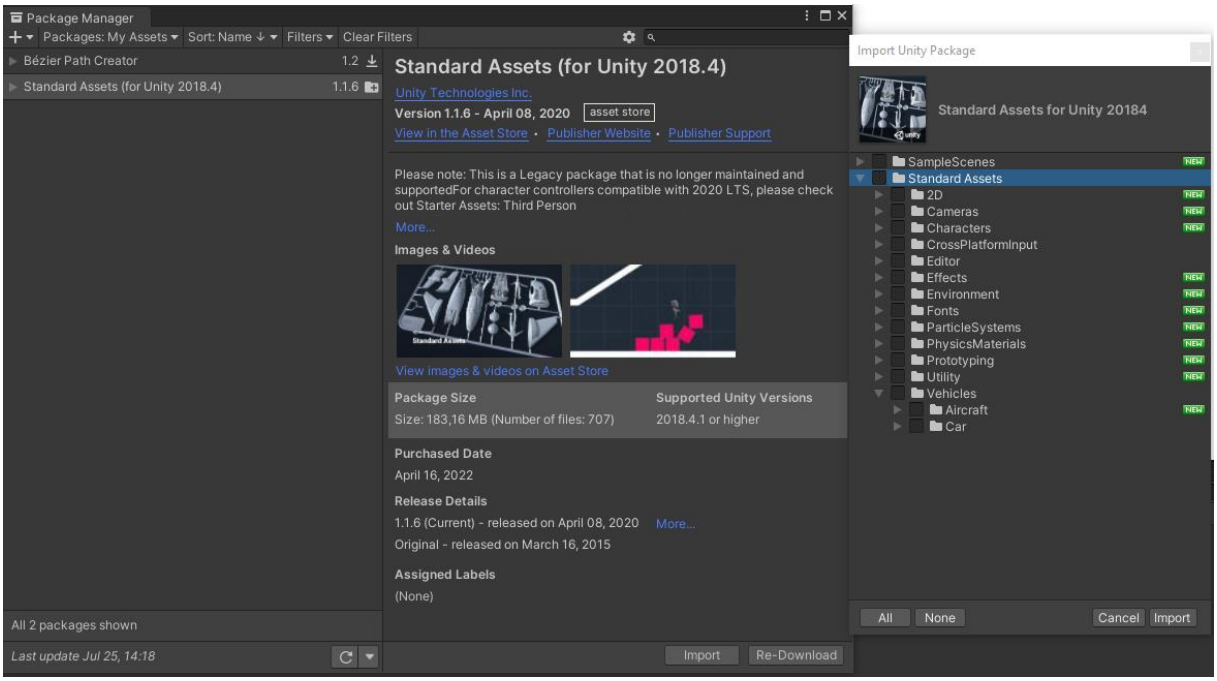


Slika 4. Game Object opcije

Nakon dodavanja terena, sâm teren će biti bez teksture. U prozoru inspektora vidimo komponente (*eng. Components*) odabranog objekta. Ispod komponente transformiranja se nalazi komponenta teren. U komponentu terena možemo vidjeti različite opcije od kojih ćemo najčešće koristiti opciju bojanja terena (*eng. Paint texture*) koju možemo koristiti za bojanje tekstura na terenu ili za dizanje odnosno spuštanje terena te glađenje razlike u visinama terena. U assets folderu unutar prozora projekta dodajemo folder s teksturama. Unutar tog foldera dodajemo teksture koje će poslužiti u izradi igre. Nakon što smo dodali teksture, teksture možemo duplicirati te unutar duplikata u prozoru inspektora promijenimo tip teksture

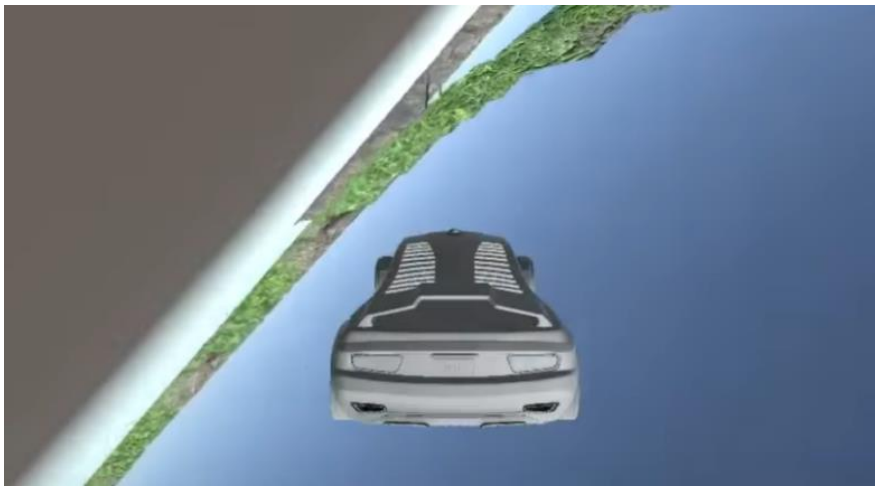
u normalnu mapu (*eng. Normal map*) te tijekom bojanja možemo dodati normalnu mapu kako bi lažirali refleksije neravnina i udubljenja korištenjem RGB-a same teksture bez dodavanja ekstra poligona. Normalne mape su prepoznatljive u Unityju uz pomoć njihove plave nijanse.

Kako bi dodali auto u projekt, trebamo ga samo uvesti preko Unity menadžera paketa (*eng. Package manager*) te uvezujemo standard assets i odabiremo pod Standard Assets: CrossPlatformInput, Editor i pod Vehicles Car. Proces uveza se može vidjeti na slici 5.



Slika 5. Uvoz auta preko package managera

Nakon uvezivanja auta pridružimo kameru našem autu tako da povučemo kameru i ispustimo na auto. Problem s trenutnom kamerom je da prati auto uz sve osi. Te u slučaju da prevrnemo auto kamera će se prevrnuti s autom kao na slici 6.



Slika 6. Prevrnuta kamera

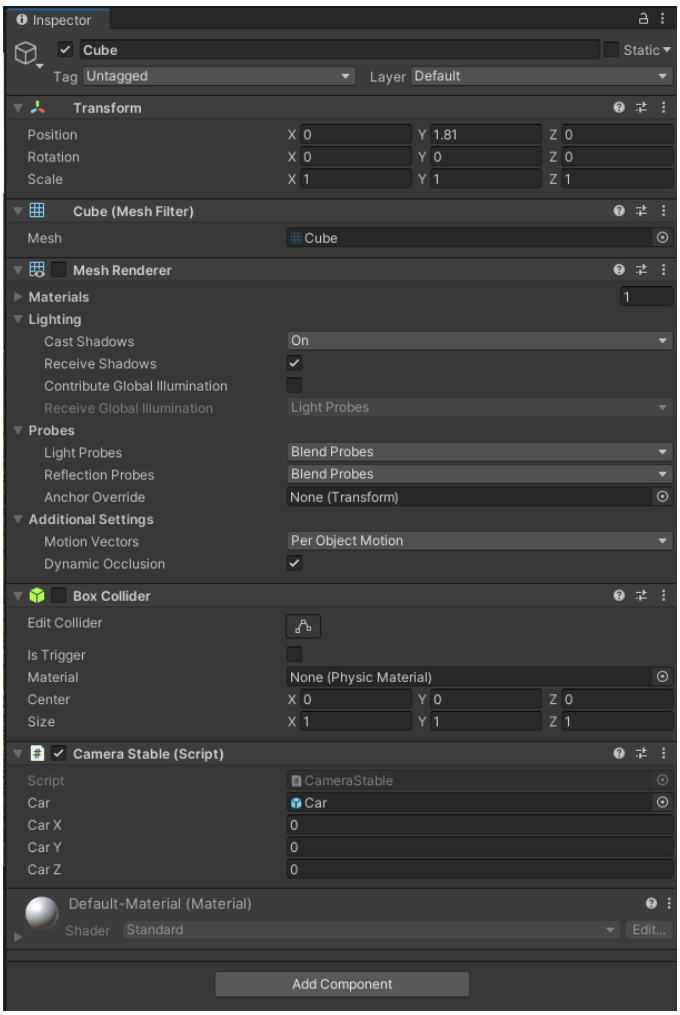
6. C# kodiranje i kamera

Za modificiranje ponašanja kamere koristit ćemo skripte. Odnosno ponašanje kamere ćemo promijeniti vlastitim kodom u C# jeziku. Prije samih skripti dodat ćemo novi 3D objekt koji će biti obična kocka. U prozoru inspektora ćemo maknuti mrežu (eng. Mesh renderer) i kocku (eng. Box collider). Samu kocku ćemo povezati s autom kako smo povezali kameru, a kameru ćemo premjestiti na kocku tako da kamera prati kocku koja će pratiti auto te će sama stabilizacija biti na kocki. Kako bi započeli sa skriptom kreirat ćemo folder za skripte te unutar tog foldera kreirati novu skriptu koju ćemo pošto je cijeli projekt na engleskom nazvati CameraStable. Duplim klikom na novokreiranu skriptu otvaramo Visual Studio editor i susrećemo se s inicijalnim uobičajenim Unity kodom. U inicijalnom kodu imamo deklariranje klase imena naziva skripte te dvije funkcije, start i update. Funkcija start će pokrenuti set radnji na početnoj slici kada je skripta omogućena, dok update će se zvati svakoj slici (eng. Frame) igre.

Započet ćemo skriptu deklariranjem objekta i tri float vrijednosti koje ćemo koristiti za rotaciju kocke. Objekt ćemo jednostavno nazvati auto na engleskom (eng. Car) te tri float vrijednosti CarX, CarY, CarZ za tri osi rotacije. Funkcija starta ne treba te ćemo je maknuti, a unutar update funkcije ćemo u float vrijednosti unijeti vrijednosti rotacije auta uz pomoć Eulerovih kutova. Transformirat ćemo kocku uz pomoć vector3 strukture koja će nam koristiti da 3D pozicije i putanje.

Da bi povezali kocku sa skriptom jednostavno povučemo i ispustimo skriptu iz prozora projekta na kocku na prozoru hijerarhije. Nakon dodavanja možemo vidjeti novu komponentu unutar kocke koja opisuje novododanu skriptu te kako bi povezali auto s objektom unutar

skripte samo povučemo auto iz hijerarhije na mjesto unutar komponente koji ga označava u obliku objekta. Završeni izgled u inspektoru se vidi na slici 7.



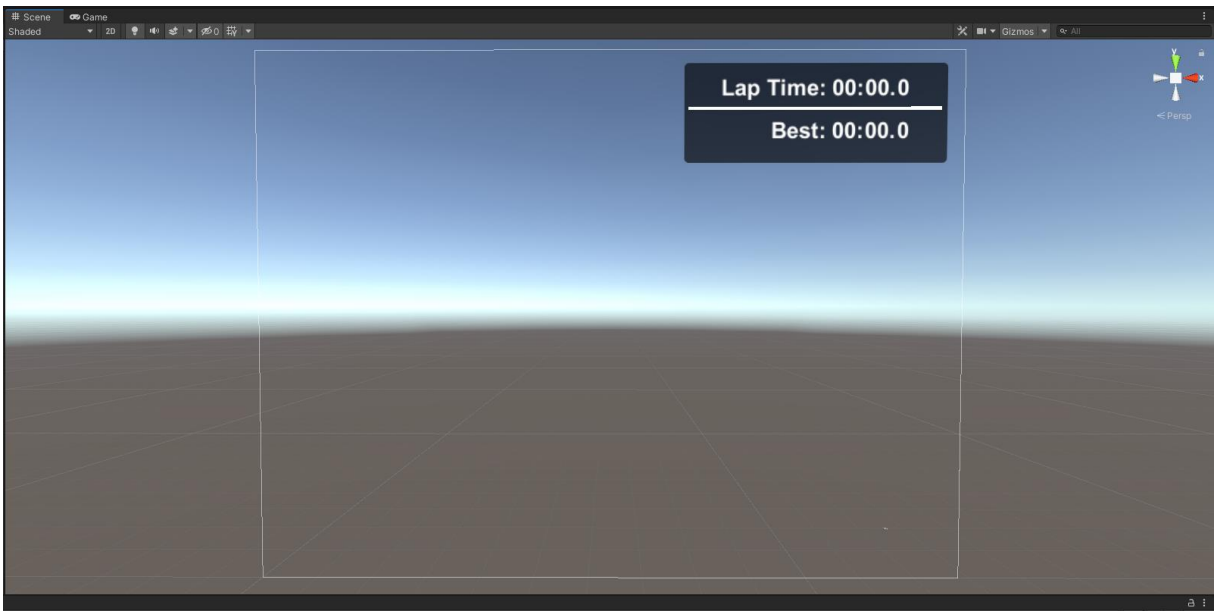
Slika 7. Inspektor kocke

7. Korisničko sučelje i okidači

7.1. Sučelje

Kako bi započeli sa sučeljem odnosno HUD-om (eng. Heads-on display) ili OSD-om (eng. On-screen display) unutar GameObjecta na toolbaru pod kategorijom UI stvaramo platno (eng. Canvas) objekt koji će nam omogućiti slaganje HUD-a. Ono što trenutno želimo pokazati je vrijeme trenutnog kruga i najbolje napravljeno vrijeme kruga. Prvu stvar koju ćemo dodati je ploča (eng. Panel) koja se može pronaći također pod UI elementima GameObject-a. Ploča će biti dijete platna, a djeca ploče će biti razni objekti teksta koje ćemo urediti u željenu poziciju.

Radi preglednosti koristit ćemo objekt sirove slike kao separator između trenutnog i najboljeg vremena kruga. Završni izgled sučelja se može vidjeti na slici 8.

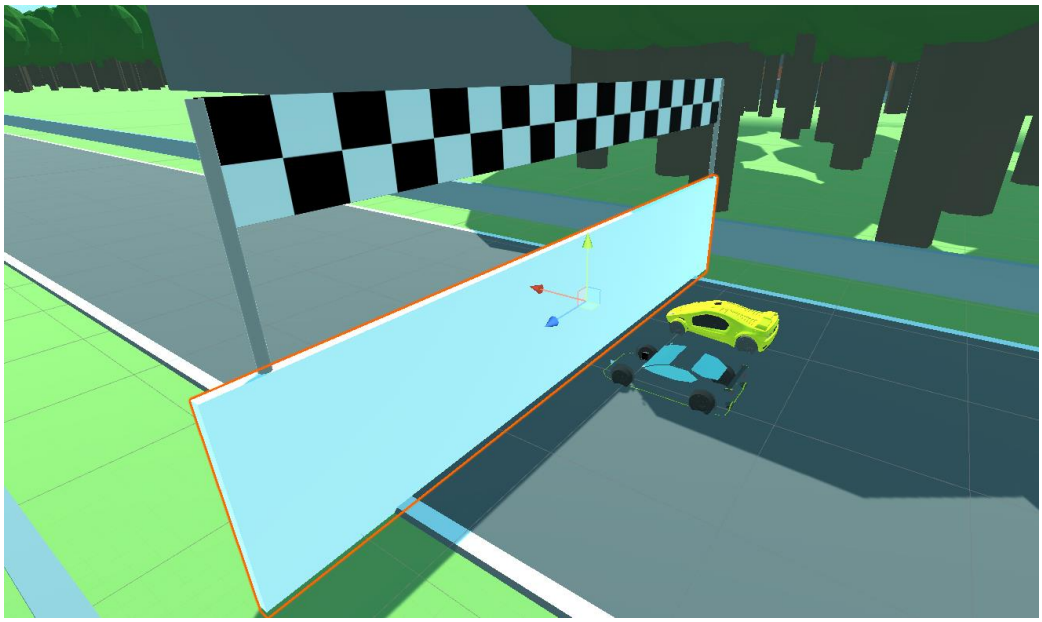


Slika 8. HUD

7.2. Okidači

Za kontroliranje vremena početka i završavanja kruga dodat ćemo startnu liniju. Startnu liniju možemo napraviti od dva cilindra koja će predstavljati stupove te pravokutnikom koji će povezivati ta dva stupa. Za materijal zastave dovoljan je jednostavna kockasta crno bijela tekstura, a da bi spriječili rastezanje teksture nad objektom u inspektoru se promijeni popločavanje (eng. Tiling) na veću vrijednost da se tekstura ponavlja umjesto rastezanja.

Za sam početak i završetak mjerenja koristi ćemo okidače koje stvaramo tako da objekt, u ovom slučaju pravokutnik koji se nalazi na startnoj liniji te mu u inspektoru promijenimo unutar komponente BoxCollider stanje okidača u aktivno. Također ćemo objekt napraviti nevidljivim pošto nam je potreban samo okidač. Tako ćemo omogućiti objektu nakon što ga dotaknemo da aktiviramo skriptu povezanu uz taj objekt. Na sljedećoj slici možemo vidjeti izgled okidača na startnoj liniji.



Slika 9. Okidač na startnoj liniji

Za skriptu početka koristit ćemo funkciju Unity-a kale Time zvane deltaTime. Ta funkcija će nam omogućiti mjerenje vremena. Funkcija mjeri intervale u sekundama [izvor], te za naše potrebe milisekundi, odnosno desetke sekundi tu vrijednost ćemo pomnožiti s deset. Vrijednosti ćemo ispisati u text komponentu našeg objekta za svaku vrijednost te primjer koda za desetinke koji se ponavlja za sekunde i minute je sljedeći:

```
MiliCount += Time.deltaTime * 10;

        if (MiliCount > 9)
        {
            MiliCount = 0;
            SecondCount += 1;
        }
 MiliBox.GetComponent<Text>().text = "" + MiliDisplay;
```

Nakon početka mjerenja trebamo provjeriti da korisnik prolazi kroz stazu te otprilike na sredini staze ćemo staviti okidač koji će omogućiti okidač kruga. Okidač kruga će inicijalno biti onemogućen tako da se omogući samo kada korisnik prođe pola staze, u protivnom može doći do greške mjerenja kruga. Unutar skripte za središnji okidač će se nalaziti OnTriggerEnter funkcija koja se pokreće nakon što drugi objekt prođe kroz objekt te unutar funkcije se samo nalazi aktiviranje okidača za kraj kruga.

Okidač za kraj kruga će nakon aktivacije uzeti vrijeme trenutnog kruga te ga staviti u najbolje vrijeme. Kada se doda spremanje vremena dodat ćemo usporedbu s prijašnjim vremenom. Izgled zapisivanja zadnjeg kruga te resetiranja trenutnog vremena je sljedeći :

```
	void OnTriggerEnter()
	{
		MinuteBoxBest.GetComponent<Text>().text = 
LapTimeBox.GetComponent<LapTimeManager>().MinuteBox.GetComponent<Text>().te
xt;
		SecondBoxBest.GetComponent<Text>().text = 
LapTimeBox.GetComponent<LapTimeManager>().SecondBox.GetComponent<Text>().te
xt;
		MiliBoxBest.GetComponent<Text>().text = 
LapTimeBox.GetComponent<LapTimeManager>().MiliBox.GetComponent<Text>().text
;

		LapTimeManager.MinuteCount = 0;
		LapTimeManager.SecondCount = 0;
		LapTimeManager.MiliCount = 0;

		HalfLapTrig.SetActive(true);
		LapCompleteTrig.SetActive(false);
	}
```

7.3. Spremanje

Za spremanje podataka korisnika koristit ćemo klasu PlayerPrefs. Ta klasa služi za spremanje preferenci između različitih sesija igre, može spremati riječi, decimalne vrijednosti ili cjelobrojne vrijednosti. [10] Spremanje kruga će se sastojati na isti način kao prikazivanje teksta kruga. Primjer jednog spremanja, u ovom slučaju sekundi je sljedeći:

```
	PlayerPrefs.SetInt("SecondSave", LapTimeManager.SecondCount);
```

Ponovit ćemo isto za sekunde i desetinke, ali u slučaju desetinki je decimalna (*eng. Float*), a ne cjelobrojna vrijednost. Za učitavanje najbržeg kruga koristit ćemo funkciju get umjesto set kod postavljanja. Napravit ćemo novu skriptu koja će dohvaćati spremljene podatke te će ih postavljati kao na primjeru zapisivanja unutar željenih komponenti.

Kada imamo namješteno spremanje vremena vratit ćemo se na pokazivanje najbržeg vremena za korisnika. Kako bi prepoznali koje je najbrže vrijeme u mjerenju vremena ćemo

dodati još jednu varijablu koja će mjeriti čisto vrijeme. Nakon prolaska kroz krug to ćemo vrijeme spremiti kao što smo spremali prijašnja vremena. Potreba drugog mjerača je u usporedbi, drugo vrijeme nećemo vraćati na nulu da pokazujemo sekunde i minute. Unutar zapisivanja najboljeg kruga ćemo dodati jednu if izjavu koja će usporediti trenutno čisto vrijeme sa spremljenim te ako je trenutno manje će zamijeniti staro vrijeme, spremiti novo i novo prikazati kao brže vrijeme.

7.4. Odbrojavanje

Nakon dodavanja okidača dolazi odbrojavanje za početak utrke. Za dodavanje odbrojavanja koristit ćemo animacije unutar Unity-a. Na vrhu projektnog prozora na desnoj strani otvaramo karticu animacije pošto prilikom početnih postavki nije otvorena. Unutar kanvasa dodajemo novi tekst i upisujemo broj 3 pošto će odbrojavanje početi od 3 sekunde, za animaciju ćemo koristiti rotaciju na x osi te quaternion interpolaciju. Razlog korištenja quaternion interpolacije umjesto standardne rotacije jest da Eulerovi kutovi na standardnoj rotaciji pate od gubitka „stupnja slobode“, odnosno ako se dogodi da prvi i druga rotacija rezultira trećom osi koja pokazuje u istom smjeru gubi se ta rotacije jer se ne može primijeniti oko jedinstvene osi.[8] Na 60. slici, odnosno na sekundi će rotacija biti 90° te će tekst nestati. Za ostatak odbrojavanja ćemo koristiti skripte. Skripta za odbrojavanje će se sastojati od postavljanja teksta na željeni broj i puštanje zvuka, čekanje sekunde te ponovno postavljanje teksta na drugi broj i puštanje zvuka. Na zadnjoj sekundi aktiviramo odbrojavanje kruga i dajemo kontrolu nad autom s funkcijom SetActive. Kada pridružimo skriptu odbrojavanju možemo isključi kontrolu auta nad autom te odbrojavanje vremena, te će se ono uključiti preko skripte, a ne pri paljenju igre.

Za zvuk prilikom odbrojavanja trebamo pridružiti željene datoteke zvuka na kameru tako da stvorimo prazno dijete te tom djetetu pridružimo zvuk. U inspektoru isključimo opciju koja se zove „Play on awake“ pošto ćemo zvuk paliti samo preko skripte. U skripti odbrojavanja kako bi pustili zvuk trebamo samo pozvati objekt s funkcijom „Play().“

7.5. Brojanje krugova

Slično kao prikazivanje sučelja, ovaj put na suprotnoj strani ćemo staviti brojanje krugova. Na sučelju će se nalaziti jednostavan brojač krugova koji će sadržavati napravljeni broj krugova i potreban broj krugova. Unutar skripte za okidač završetka kruga dodajemo jednostavnu cjelobrojnu varijablu koja će se povećavati za jedan nakon svakog kruga te ćemo uz pomoću te varijable osvježavati broj napravljenih krugova. Za kraj utrke koristit će se

jednostavna skripta osvježavanja koja će provjeriti je li se ispunio željeni broj krugova. Ako se željeni broj krugova ispunio, aktivirat će se okidač za kraj utrke.

Za kraj utrke će se nalaziti jedna scena (*eng. Cutscene*) koja će rotirati kameru na kraju oko auta. Napravit ćemo jednostavnu kameru i objekt kocke koji će se povezati na auto, a kamera na kocku. Ideja iza završne scene je rotiranje kocke, a rotiranjem kocke će se rotirati kamera povezana s kockom oko auta. Na kocki ćemo maknuti prikaz te će kocka biti nevidljiva.

Unutar skripte će se nalaziti isključivanje okidača za završetak kruga, isključivanje kontrole nad autom i stavljanjem brzine na nulu. Na kraju će biti aktivacija završne kamere. Završni proizvod će imati efekt nakon završetka utrke smrzavanja auta te filmsko rotiranje kamere oko modela auta.

8. Umjetna inteligencija

Namještanje umjetne inteligencije (AI) u Unity-u je vrlo lagano. Za početak se dodaje novi auto koji će biti kontroliran od strane računala. Iz Unity-a dodajemo AI auto koji u sebi sadrži standardnu AI skriptu za kontroliranje te započinjanjem scene možemo primijetiti kako se drugi auto kreće bez našeg unosa. Također se može primijetiti da bez ikakvog uređivanja auto se vozi u krug jer meta, odnosno cilj kretanja auta je on sam, te se time kreće u krug. Cilj kretanja auta se može pogledati u inspektoru auta.

Kako bi započeli s uređenom linijom kretanja prvo ćemo promijeniti oznaku auta. Naziv oznake nije bitan te se može proizvoljno staviti. Nakon oznake možemo krenuti dodavati kontrolne točke auta (*eng. Checkpoints*) tako da unesemo novi 3D objekt kocke. Prva kocka se stavlja na inicijalnu točku kretanja auta. Ta točka se služiti kao cilj kretnje autu. Možemo duplicirati točku i staviti drugu točku dalje niz traku. Tom metodom možemo dodati koliko želimo i koliko nam je potrebno točaka da nam auto prati traku. Nakon što su sve željene točke za traku dodane, napravimo još jedan objekt kocke koji će nam služiti kao dinamički cilj. Kreiramo novu skriptu te će se sada implementirati logika iza položenih kocaka. Unutar skripte trebat ćemo objekte za kocke te brojač, brojač će od nule nakon svakog cilja se povećavati za jedan. Naša kocka koja nam služi kao dinamički cilj će nam biti točka koju će AI auto pratiti, u trenutku kada AI auto dotakne dinamičku točku uzimamo sljedeću točku na stazi i mijenjamo poziciju dinamičke točke na sljedeću korištenjem metode transform.position. Skripta se sprema te se pridružuje dinamičkom cilju.

Nakon što smo odredili sve točke i pripremili skriptu trebamo još promijeniti autu cilj kretanja. Već je spomenuto da auto kao cilj kretanja ima sam sebe te kako bi promijenili taj cilj

uzimamo naš dinamički cilj i povlačimo ga na zadano mjesto cilja u inspektoru. Pokretanjem scene možemo testirati točke kretanja, ako auto bude spriječen u dolasku do sljedeće točke možemo jednostavno dodati novu točku između dvije problematične ili ukloniti prepreku koja sprječava auto.

9. Dodaci

9.1. Dodatne kamere

Ideja dodatnih kamera jest da postoje različiti kutovi iz kojih igrač može gledati na svoj auto i samim time mijenjati perspektivu igraču na igru. Trenutno se nalazi standardna kamera na autu, kamera se nalazi sa zadnje strane auta bez velike udaljenosti. Jedna od dodatnih kamera će imati isti princip kao glavna, ali će se nalaziti više iza i više iznad auta samim time dozvoljavati igraču veću preglednost na auto te okolinu auta. Druga kamera će se nalaziti na haubi motora, odnosno ispred samog auta. Kamera na toj poziciji će dati igraču veću preglednost ispred auta te povećati preciznost, ali spušta percepciju okoline, takva kamera će više pristajati načinu igre za postavljanje najbržeg vremena bez protivničkog auta.

Za početak trebamo dodati novi unos tipke u igru, kako bi to ostvarili otvaramo alatnu traku, edit, project settings i unutar novootvorenog prozora otvaramo karticu pod nazivom input. Prva stvar koja se mijenja je broj veličine unosa, dižemo unos za jedan te dupliciramo zadnji unos i novom unosu mijenjamo željeni naziv. Pod pozitivnim gumbom stavljamo željeni gumb s kojim želimo da se mijenja kamera. Nakon postavljanja dodatne tipke stvaramo novu skriptu. Unutar skripte stvaramo objekte za kamere, u našem slučaju tri objekta za tri dodatne kamere, također postavljamo varijablu cijelog broja koja će određivati na kojoj smo kameri.

Prepoznavanje pritiska tipke stavljamo u metodu osvježavanja (*eng. Update*), koristimo klasu Input i Unity-jev Input Manager. Pošto koristimo tipku koja će nam služiti kao događaj, a ne kretnja u igri, koristit ćemo GetButtonDown funkciju. [9] Logika iza skripte se sastoji da će cijeli broj se rotirati oko 3 broja koja će simbolizirati tri kamere, nakon što dođemo da najveći broj trebamo resetirati na najmanji, odnosno početnu kameru. Tu logiku namještamo unutar if izjave, ako broj nije najveći povećavamo ga za jedan i pokrećemo drugu funkciju nakon provjere. Druga funkcija služi za mijenjanje kamera, provjerava koji je broj zadan, aktivira sljedeću kameru i deaktivira staru kameru. Bitno je napomenuti da je važno da se nova kamera prvo upali prije nego što se stara izgasi jer u protivnom može doći do jedne crne slike između izvođenja radnji, pogotovo na sporijim računalima. Na kraju stvaramo novi prazni objekt,

pridodajemo skriptu praznom objektu i unutar inspektora pridružujemo kamere skripti. Na sljedećoj slici možemo vidjeti različite načine kamera.

Slika 10. Različiti oblici kamere

9.2. Mini mapa

Mini mapa u igri dozvoljava igraču brzi i lagani pogled na rub ekrana kako bi se uvjerili da idu u pravom smjeru. U igrici utrkivanja mini mapa može dodatno poslužiti u opažavanju protivnika na stazi. Kako bi napravili mini mapu unutar Unity-a potrebna nam je još jedna kamera. Kameru namještamo po želji, za potrebe mini mape najbolja pozicija je pogled direktno na auto pod kutem od 90°. Visina kamera je dovoljna da se vide otprilike dvije do tri dužine auta iza i ispred našeg, ali također jedna od mogućnosti namještanja mini mape je pogled na cijelu mapu, nedostatak tog pristupa je gubitak detalja i preciznosti auta i okoline auta.

Kako bi kameru povezali s korisničkim sučeljem potrebna nam je nova tekstura. Na projektnom prozoru desnim klikom otvaramo meni te stvaramo novu teksturu tipa render texture. Povratkom na kameru u inspektoru se nalazi opcija za ciljanu teksturu. Povlačenjem novostvorene teksture na zadanu lokaciju na kameri smo stvorili teksturu koja se mijenja ovisno što kamera vidi.

Unutar našeg korisničkog sučelja stvaramo novu sliku koja će reprezentirati prostor za mini mapu. Kako mapa ne bi bila odvraćala pozornost poželjno ju je smjestiti u jedan od kutova s relativno malom veličinom. Unutar inspektora slike samo povučemo našu teksturu povezanu s kamerom na sliku. Pokretanjem igre možemo primijetiti da mini mapa funkcionira, ali pošto je kamera direktno povezana s autom, nju također trebamo stabilizirati.

Stabilizacija za mini mapu nam je bitna na dvije osi, os x i os z. Os y će pratiti auto te rotirati mini mapu u odnosu kako je auto rotiran kako bi mapa stalno bila orijentirana u pogled auta. Jednostavna skripta koja služi za stabilizaciju auta koristi objekt auta od kojeg će dohvaćati rotaciju i decimalnu vrijednost u koju će zapisati rotaciju osi y. Skripta će nakon toga

promijeniti rotaciju naše kocke koja će nam služiti za stabilizaciju kamere. Opisani kod je sljedeći:

```
public GameObject Car;
public float CarY;

void Update()
{
    CarY = Car.transform.eulerAngles.y;

    transform.eulerAngles = new Vector3(0, CarY, 0);
}
```

Na slici 11. možemo vidjeti konačan izgled mini mape.

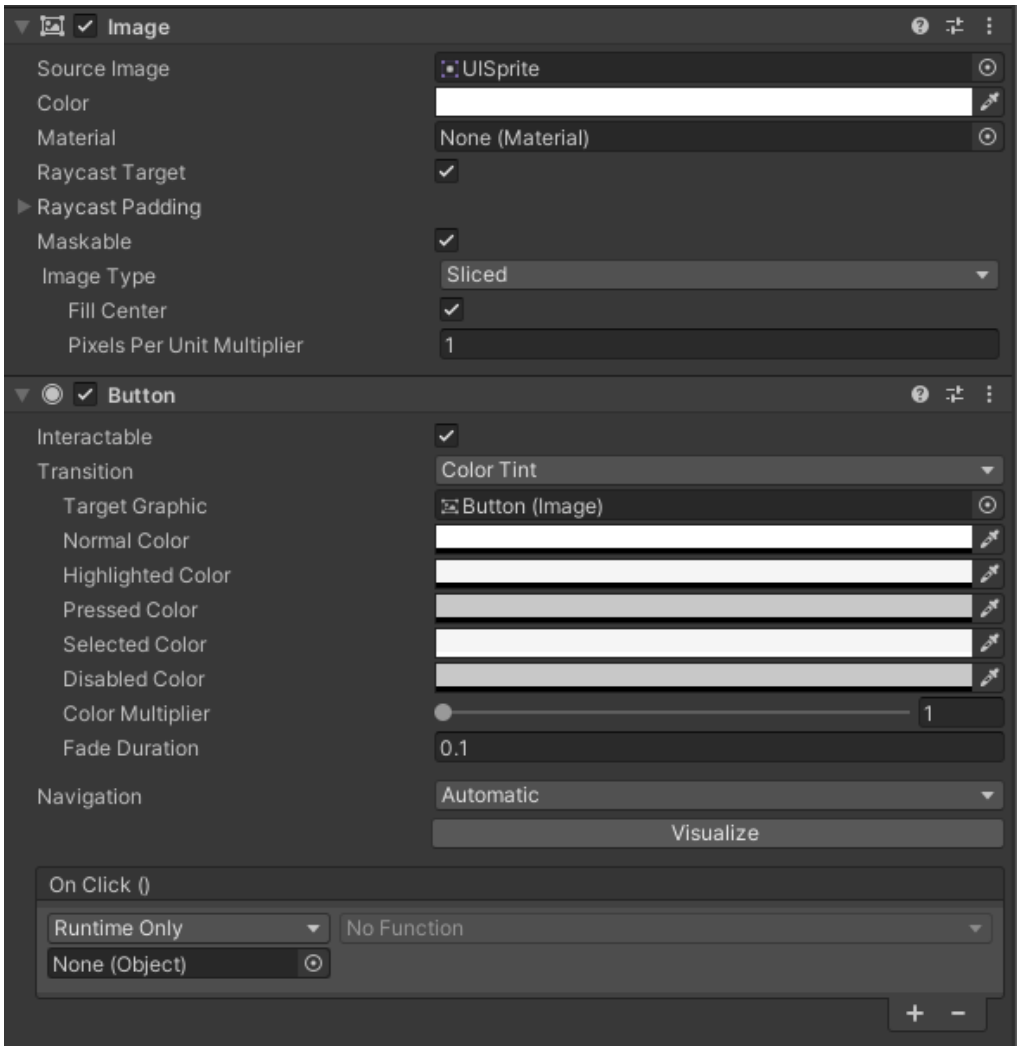
Slika 11. Mini mapa

10. Scene

Scene unutar Unity-a služe kao različiti kadri igre. Cijela igra je podijeljena u različite scene. Jednu scenu igre možemo smatrati kao jednom razinom. Prijašnji rad na igri je pokazivao izradu utrke i ta utrka se nalazi unutar jedne scene. Ta scena se sada sprema kako bi kreirali novu scenu. Unutar nove scene nećemo raditi još jednu scenu, nego ćemo koristiti sljedeće scene za meni i različite sekcije unutar menija.

10.1. Glavni meni

Možemo primijetiti da nakon kreiranja nove scene, u prozoru hijerarhije ili na pregledu scene nam se sve vratilo na standardne vrijednosti, odnosno sve nam je prazno. Izrada glavnog menija će koristiti samo elemente korisničkog sučelja. Jedan od novijih elemenata je gumb (*eng. Button*) koji će nam služiti za navigaciju između scena. Stvaranjem gumba možemo primijetiti da se stvorio odnos roditelj-dijete između dva objekta. Jedan objekt reprezentira sam gumb i za njega možemo vidjeti razne opcije u paneli inspektora. Gumbu možemo mijenjati boju, ali mu također možemo mijenjati istaknutu boju kao i pritisnutu ili onemogućenu boju. Razne druge opcije su nam dostupne koje možemo vidjeti na slici 12:



Slika 12. Inspektor gumba

Drugi objekt gumba, dijete, je tekst koji se nalazi na gumbu te opcije teksta su iste kao sve opcije teksta unutar Unity-a. Gumbima ćemo urediti glavni meni da nam koristi za odabir željenog auta, brzu opciju igranja, pregled zasluga i izlazak iz igre.

10.2. Meni za odabir

Koristeći prije spomenute metode otvaramo novu scenu te kreiramo meni za odabir. Izgled samog menija nije bitan nego su bitne funkcionalnosti odabira modela auta, vrste igre i odabir staze na kojoj će se voziti. Neke od navedenih funkcionalnosti ćemo obraditi u nastavku rada. Za dodavanje slike na gumb trebamo uvesti teksturu u projekt te unutar inspektora promijeniti opcije teksture, specifičnije tip teksture na grafički znak (*eng. Sprite*). Teksturu možemo staviti na source image unutar inspektoru gumba.

10.3. Funkcije gumba

Prva stvar kod funkcionalnosti gumba je kreiranje skripte za njih. Unutar skripte trebamo definirati radnje svakog gumba. Te radnje ćemo definirati unutar funkcija i gumb će biti povezan s funkcijom. Kako bi mogli upravljati između scena unutar Unity-a trebamo scene poredati unutar opcija za buildanje. Odlaskom na toolbar, file i build settings nam se otvara prozor s različitim scenama. Unutar prozora postoji tipka za dodavanje otvorene scene u build. Poredak scena unutar build opcija je poredak scena koji se otvara u finalnoj verziji aplikacije. Prva scena koja bi se dodala bi bila glavni meni ili u slučaju nekakvog splash ekrana prije, ta scena bi prva bila, a nakon nje glavni meni.

Pritiskom dodavanja scene, sceni se unutar prozora pridodaje broj, prvoj sceni nula, a nakon nje se povećava za jedan. Dodavanjem svih scena možemo se orijentirati po njima koristeći zadane brojeve ili imena. Za orijentiranje po scenama koristit ćemo ugrađenu klasu u Unity-u pod nazivom SceneManagment koja će nam omogućiti lagano prebacivanje s jedne scene na drugu.

Nakon stvaranja skripte zadanim gumbovima pridodajemo funkcije. Kako bi gumbu pridodali funkciju unutar inspektora gumba, već prije prikazan na slici 9, unutar sekcije pri dnu za radnju prilikom klika stišćemo plus za dodavanje nove funkcije. Unutar sekcije za objekt pridodajemo prazan objekt koji sadrži novonastalu skriptu te za funkciju odabiremo funkciju koju želimo na tom gumbu. Trenutnim gumbima za stazu možemo dodati jednostavno otvaranje scene sa zadanom stazom.

11. Odabir auta

Prilikom odabira trake, omogućit ćemo odabir auta. Odabir će se odnositi na boju auta unutar igre. Kako bi započeli s odabirom, unutar menija za odabir na kojem bi u ovom trenutku imali samo odabir trake, promijenimo meni tako da postavimo nekakvu vrstu selekcije auta. Nakon postavljanja gumba selekcije, postavimo skripte vezane za meni. Ideja iza odabira je da korisnik prvo odabere auto pa onda odabir staze. Kako bi osigurali da korisnik odabire na takav način, prvo onemogućimo odabir staze, a unutar gumba za aute ćemo omogućiti selekciju, nakon što korisnik odabere auto. Skripta će se sastojati od onoliko funkcija koliko ćemo imati auta, te će u svakoj funkciji zadati vrijednost cijelog broja koji će reprezentirati jedan auto, odnosno jednu boju auta. Nakon zadavanja vrijednosti ostalo je još samo omogućavanje odabira staze. Prisjetimo se da nakon odabira staze igra nam otvara odabranu stazu te mi trebamo samo omogućiti odabir trake koji će prebaciti scenu nakon odabira.

Kako bi završili s odabirom auta, trebamo željeni dio auta koji će biti u različitim bojama duplicirati. Dupliciramo onoliko puta koliko želimo boja na autu, odnosno svakom novokreiranom objektu mijenjamo materijal na željenu boju. Objekte onemogućimo te ćemo određenu boju koju korisnik odabere omogućiti skriptama. Unutar skripte deklariramo objekte za obojene dijelove auta te cjelobrojni broj koji će uzeti vrijednost odabranog auta unutar menija za odabir. Tijekom paljenja skripte cjelobrojnoj vrijednosti dajemo vrijednosti odabira auta, te s tom vrijednošću omogućujemo određenu boju koju je korisnik odabrao. Iz perspektive korisnika, nakon odabira auta korisniku će nakon paljenja trake biti aktiviran auto sa željenom bojom.

12. Načini igre

Unutar igre korisnik će imati dvije mogućnosti za različit način igranja igre. Jedna od mogućnosti dostupna korisniku će biti normalna utrka, a druga od mogućnosti će biti način vremenskog napada (*eng. Time attack mode*). Korisniku će se odabir predstaviti na meniju za odabir zajedno s odabirom auta i odabirom staze.

12.1. Utrka

Standardna utrka će sadržavati sve trenutno implementirane mogućnosti unutar igre. Broj krugova na korisničkom sučelju ćemo staviti unutar kolekcije za korisničko sučelje za utrku

te time lakše omogućavati ili onemogućavati po potrebi. Također u utrci će se nalaziti neprijateljski auto dok unutar time attack načina neprijateljski auto i broj krugova neće postojati.

12.2. Time attack

Time attack unutar igre će služiti korisniku da poboljša svoje najbolje vrijeme na traci. Također će korisničko sučelje biti pojednostavljeno i neprijateljski auto maknut i vožnja biti beskonačna dok korisnik ne odluči izaći. Kako smo u prošlom poglavlju postavili sve potrebno sučelje za utrku unutar jedne kolekcije, skripta za time attack može biti pojednostavljenja tako da se unutar skripte provjeri stanje načina igre te ako se potvrdi da je korisnik odabrao način time attack-a da igrica pojednostavi sučelje gašenjem potrebnog za utrku i deaktiviranjem neprijateljskog auta. Unutar skripte potrebno je naglasiti da se nalazimo u načinu igre time attack. Objašnjena skripta je sljedeća:

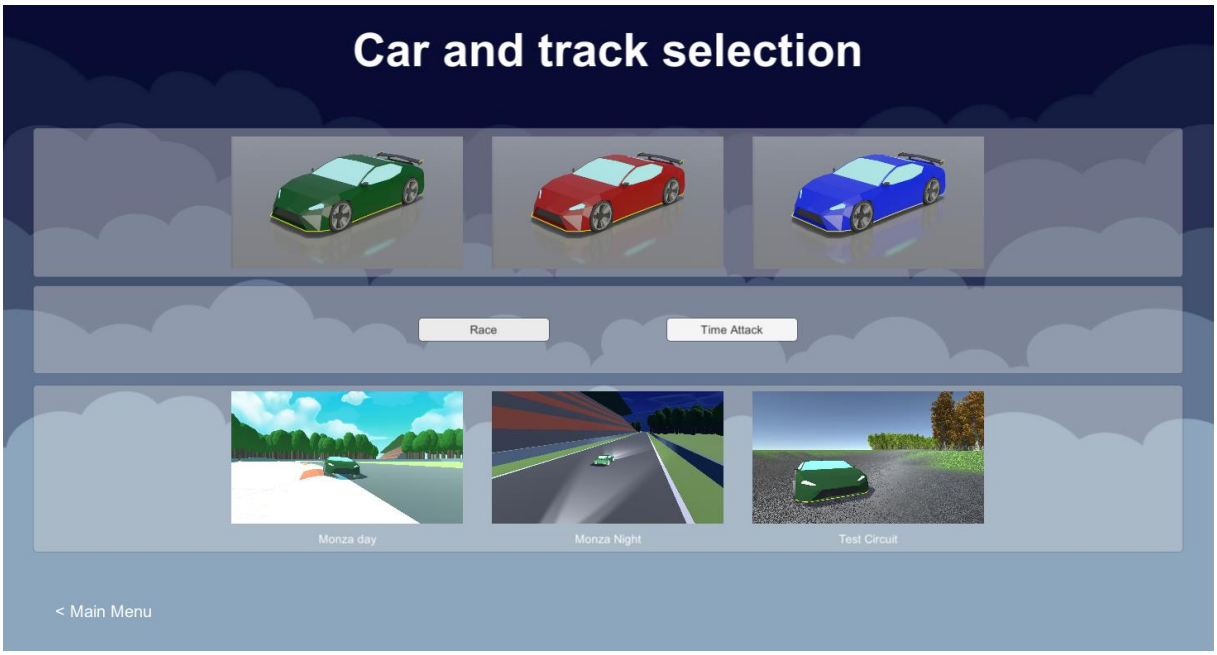
```
public int ModeSelection;
    public GameObject AICar;
    public GameObject RacePanels;

    public static bool TimeMode = false;

    void Start()
    {
        ModeSelection = ModeSelect.RaceMode;

        if(ModeSelection == 1)
        {
            TimeMode = true;
            AICar.SetActive(false);
            RacePanels.SetActive(false);
        }
    }
```

Potreba bool stanja za time attack nam je potrebno za prijašnju skriptu o krugovima za utrku. Provjera završenih krugova stavljamo u if uvjet koji provjera ako način igre nije time attack onda broji krugovi, ako jest, brojanje se preskače te se broji samo vrijeme. Konačan izgled menija koji sadrži panele koje se otključavaju jedne po jedne se može vidjeti na sljedećoj slici.



Slika 13. Izgled menija

13. Završavanje i buildanje igre

13.1. Dodavanje pozicija

U trenutnom stanju igra funkcionira, ali prilikom utrke ne znamo na kojoj smo poziciji. Za dodavanje pozicije unutar igre dodat ćemo tekst unutar kolekcije korisničkog sučelja za utrku. Ne želimo pokazivati poziciju tijekom načina za time attack jer korisnik se neće boriti s neprijateljskim autom u tom načinu. Kako bi dodali skriptu za poziciju te napravili samu logiku iza određivanja pozicije možemo koristiti dva načina. Jedan od načina je dodavanje okidača na određene dijelove staze te nakon što bilo koji auto prođe okidač da ga označi u polju pozicija te promijeni tekst pozicije. Za ovaj način broj okidača na stazi određuje reaktivnost samog mijenjanja pozicija jer što je više okidača to će se više osvježavati pozicija na stazi. Tim načinom se treba uskladiti količina okidača s isplatljivosti. Prednost navedenog načina je skalabilnost, odnosno, neovisno koliko je auta na stazi taj način će ih precizno pisati u polje i pridodavati im pozicije. Drugi način nije lagano skalabilan, ali u našem slučaju s jednim protivnikom je jako jednostavan.

Način na koji se mogu odrediti pozicije između dva auta je pomoću dva okidača koja se nalaze na samom autu. Oba okidača bi se nalazila na sredini auta, ali s malim istupom, jedan okidač će se nalaziti malo ispred drugog okidača, odnosno drugi malo iza prvog. S tim možemo implementirati pozicije da funkcioniraju na način aktiviranja okidača te ovisno o kojeg smo okidača zadnjeg dirali, ako smo dirali okidač koji se nalazi pozadi bi nam spustilo poziciju. U suprotnom, ako je zadnji okidač prednji, pozicija bi nam se digla. Takva implementacija je jednostavna i kratka s dva auta, ali povećanjem broja auta moramo implementirati za svaki auto njegove okidače te nam je skalabilnost mala naspram načina postavljanja okidača po cesti.

13.2. Build igre

Kako bi se konačna igra mogla igrati i testirati, trebamo napraviti build igre. Build igre služi kao izgradnja našeg projekta i spremanja tog projekta u executable datoteku ili u razne druge datoteke poput apk za android. Unity omogućava razne postavke tijekom izgradnje te odlaskom u toolbar, pod edit i project settings možemo vidjeti postavke projekta koje će se koristiti u buildu.

Pod karticom igrač možemo vidjeti informacije poput imena kompanije kojoj će se pripisati igra te samo ime igre i ikona. Sve se postavke mogu mijenjati ovisno o platformi na kojoj se igra izgrađuje. Pod karticom za druge postavke možemo vidjeti postavke vezane uz način renderiranja igre, logiranja grešaka (mogućnost bez logiranja, samo skripte ili sve) i razne druge postavke. Odabirom na karticu kvalitete (*eng. Quality*) možemo vidjeti postavke kvalitete u kojima će se igrica izgraditi te koje će druge kvalitete dopuštati.

Prije same izgradnje projekta, ako se projekt radi na drugim platformama treba se pod edit, unutar preferences skinuti razni eksterni alati za funkcionalnost na drugim platformama poput mac ili android.

Za konačan build igre odlazimo na datoteku (*eng. File*) unutar alatne trake i pod postavke builda možemo pritisnuti build i proces kreiranja igre je gotov.

14. Zaključak

U ovom završnom radu cilj mi je bio prikazati i pobliže objasniti proces izrade video igre s tematikom utrkivanja u programskom alatu Unity. Za izradu rada bilo je potrebno razumijevanje znanja stečenog na kolegijima algoritama, programiranja i programskog inženjerstva i strukturiranjem rada na ovakav način sam pokušao demonstrirati i na jednostavan način objasniti izradu igre utrkivanja te prikazati znanje stečeno na fakultetu organizacije i informatike. Kroz ovaj rad obradio sam programiranje same igrice, dizajn igre, pravljenje modela za igricu te korištenje unity alata. Ovim radom dokazujem svoju kompetenciju i želju nastavka u području programiranja i izrada igara.

15. Popis literature

[1] „Unity,“ (bez dat.) u Wikipedia, the Free Encyclopedia. Dostupno: https://en.wikipedia.org/wiki/Unity_(game_engine) [pristupano 22.6.2022.]

[2] „Space Race (video game),“ (bez dat.) u Wikipedia, the Free Encyclopedia. Dostupno: https://en.wikipedia.org/wiki/Space_Race_(video_game) [pristupano 22.6.2022.]

[3] „Unity manual,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/Manual/index.html [pristupano 23.6.2022.]

[4] „Start,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html [pristupano 25.6.2022.]

[5] „Update,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html [pristupano 25.6.2022.]

[6] „Vector3,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/ScriptReference/Vector3.html [pristupano 26.6.2022.]

[7] „Time,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/ScriptReference/Time-deltaTime.html [pristupano 27.6.2022.]

[8] „Rotation and Orientation in Unity,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/es/530/Manual/QuaternionAndEulerRotationsInUnity.html [pristupano 1.7.2022.]

[9] „Input Manager,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/Manual/class-InputManager.html [pristupano 1.7.2022.]

[10] „PlayerPrefs,“ (bez dat.) u Unity User Manual 2021.3 (LTS). Dostupno: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html [pristupano 3.7.2022.]

[11] Marco Monster, „Car Physics for Games“ , 2003. [Na internetu]. Dostupno: https://asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html [pristupano 23.6.2022.]

16. Popis slika

Slike bez izvora su slike vlastite izrade.

Slika 1: https://commons.wikimedia.org/wiki/File:Unity_Technologies_logo.svg

Slika 2: https://en.wikipedia.org/wiki/Blender_(software)

Slika 3: https://docs.unity3d.com/2017.4/Documentation/Manual/LearningtheInterface.html