IZRADA 3D TRKAĆE IGRE U ARKADNOM STILU

Barišić, Josip

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:380299

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-12-28**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni prijediplomski studij Računarstvo

JOSIP BARIŠIĆ

Z A V R Š N I R A D

IZRADA 3D TRKAĆE IGRE U ARKADNOM STILU

Split, rujan 2024.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni prijediplomski studij Računarstvo

Predmet: Objektno programiranje

Z A V R Š N I R A D

Kandidat: Josip Barišić

Naslov rada: Izrada 3D trkaće igre u arkadnom stilu

Mentor: Ljiljana Despalatović, viši predavač

Split, rujan 2024.

SADRŽAJ

SAŽETAK

Izrada 3D trkaće igre u arkadnom stilu realizacija je trkaće igre u kojoj igrač može birati između različitih vozila i staza s jedinstvenim karakteristikama, a natječe se protiv botova te cilja postaviti najbolje vrijeme staze. Tijekom utrke, igrač upravlja vozilom pomoću tipkovnice, izbjegavajući kolizije te pokušava svladava trenutnu stazu. Osvojeni bodovi omogućuju mu napredovanje kroz igru, otključavanjem novih vozila i staza. Rezultati utrka se rangiraju, prikazujući najbolje vrijeme kruga i ukupno vrijeme utrke. Ovaj rad detaljno opisuje alate, tehnologije i funkcionalnosti odnosno korisničke priče korištene da bi se postigao cilj izrade 3D trkaće igre u Unityju.

Opisan je proces izrade igre korištenjem Unity razvojnog okruženja i C# programskog jezika. Unity je odabran zbog velike količine informacija dostupnih za sve razine iskustva razvojnog programera u razvoju igara, odlično dokumentiranog sučelja i količine resursa dostupnih za integraciju s projektom.

Rad obuhvaća cjelokupan razvojni ciklus igre, uključujući postavljanje projekta, dizajn i implementaciju okruženja i likova, testiranje rubnih uvjeta, kao i razvoj igračkih mehanika te interaktivnih elemenata.

Ključne riječi: 3D trkaća igra, Unity, C#, arkadna igra

SUMMARY

3D arcade-style racing game creation

3D arcade-style racing game creation involves developing a game where the player can choose between different vehicles and tracks, each with unique characteristics, and compete against bots with the goal of setting an All-Time High Score for the track time. During the race, the player controls the vehicle using the keyboard, avoiding collisions, and attempting to master the current track. Points earned allow the player to progress through the game by unlocking new vehicles and tracks. Race results are ranked, displaying the best lap time and the overall race time. The development of the game also includes learning the C# programming language and becoming familiar with the Unity development environment. This paper provides a detailed description of the tools, technologies, and functionalities, as well as the user stories used to achieve the ultimate goal of creating a 3D racing game in Unity.

The game development process using the Unity development environment and the C# programming language is presented. Unity was chosen due to the vast amount of information available for all experience levels in game development, its well-documented interface, and the abundance of resources available for project integration.

The paper covers the entire game development cycle, including project setup, design and implementation of environments and characters, testing, as well as the development of game mechanics and interactive elements.

Keywords: 3D racing game, Unity, C#, arcade game

1. UVOD

Razvoj videoigara predstavlja dinamično i kreativno polje koje spaja umjetnost, tehnologiju i interaktivni dizajn. S obzirom na široku primjenu videoigara u zabavnoj industriji, obrazovanju, potražnja za alatima koji omogućuju jednostavno i efikasno stvaranje igara značajno je porasla. Jedan od najpopularnijih i najpristupačnijih alata za razvoj igara je Unity.

Unity je moćan softver koji omogućuje stvaranje igara za različite platforme, uključujući računala, konzole, mobilne uređaje i web preglednike. Njegova fleksibilnost, bogatstvo resursa i podrška za C# programski jezik čine ga omiljenim izborom među profesionalnim programerima, kao i među entuzijastima i početnicima. Unity se ističe po svojoj pristupačnosti, što ga čini idealnim alatom za nastanak igara svih žanrova, od jednostavnih 2D igara do složenih 3D simulacija. Cilj ovoga rada bio je stvoriti igru jedinstvenog vizualnog i zvučnog stila s naglaskom na detalje. Velikom izazovu pristupljeno je bez prethodnog iskustva u razvoju igara, a i programskom jeziku C# što je dodatno otežavalo postavljeni cilj.

U ovom završnom radu detaljno će se prikazati proces stvaranja projekta, od inicijalne ideje do završne implementacije. Kroz analizu i opis ključnih faza, čitateljima je pružen uvid u logički slijed odluka i koraka koji su vodili ka finalnom proizvodu.

Tijekom rada, predstavljene su slike Unity razvojnog okruženja koje ilustriraju različite aspekte nastanka igre, uključujući dizajn okoline, modeliranje likova i vozila te implementaciju igračkih mehanika. Ove vizualne prezentacije pomoći će u boljem razumijevanju praktičnih aspekata rada u Unityju.

Pored toga, u radu su istaknute prednosti korištenja Unity softvera, poput njegove fleksibilnosti, podrške za različite platforme, velikog broja dostupnih resursa te integracije s različitim alatima i tehnologijama. Diskutirat će se i o iskustvima stečenim tijekom rada s Unityjem, uključujući izazove i rješenja koja su primijenjena kako bi se postigao željeni rezultat.

Na taj način, rad pruža cjelovit pregled procesa nastanka igre, naglašavajući važnost dobro osmišljenog misaonog procesa te prednosti korištenja Unity softvera u izradi visokokvalitetnih igara.

Svi dodaci i zvučni efekti su besplatno preuzeti sa Unity trgovine dodataka[1] i po potrebi prilagođeni projektu. Izrada projekta provedena je uz pomoć vodiča s Youtubea[2] i službene Unity dokumentacije[3].

Poglavlje "Specifikacija igre" opisuje žanr, platformu, funkcionalnosti i ukratko predstavlja igru. Igrač se natječe protiv botova, zarađuje novac i otključava nove staze i vozila. Funkcionalnosti uključuju odabir vozila i staza, sustav napredovanja, kolizije, kontrolne točke i prikaz napretka. Poseban naglasak stavljen je na dizajn korisničkog sučelja, vizualne i zvučne efekte te stil igre.

U poglavlju "Metode i alati" opisane su ključne tehnologije korištene u razvoju igre: C# programski jezik, Unity razvojni alat i Visual Studio. **C#** je istaknut kao objektno orijentirani jezik pogodan za razvoj aplikacija i igara, posebno u kombinaciji s Unity softverom. Njegova sličnost s drugim jezicima i podrška za asinkrono i funkcionalno programiranje olakšava rad programerima. Unity je predstavljen kao svestran alat za razvoj 2D i 3D igara, sa značajkama poput izrade igara za različite platforme, intuitivnog sučelja, bogatstva ugrađenih alata, podrške za C# skriptiranje, vizualno skriptiranje, renderiranje u stvarnom vremenu i zajednice koja podržava korisnike. Visual Studio opisan je kao glavno razvojno okruženje za pisanje i uređivanje kôda u Unityju, s integriranim alatima za automatsko dovršavanje kôda, ispravljanje grešaka i upravljanje verzijama.

Implementacija igre opisuje stvaranje terena i staza, dodavanje vozila i implementaciju skripti za kontrolu vožnje, kamere i protivnika, a i izbornike za odabir vozila i staza te scene

[1] Unity Asset Store, Početna stranica, Unity Technologies, pristupljeno 11. rujna 2024, https://assetstore.unity.com.

[2] YouTube, Početna stranica, pristupljeno 11. rujna 2024, https://www.youtube.com

[3] Unity Documentation, Početna stranica, Unity Technologies, pristupljeno 11. rujna 2024, https://docs.unity.com

koje prikazuju pobjedu i poraz. Razvijeno je korisničko sučelje koje prikazuje ključne informacije tijekom igre, a pohranjeni su i prikazani najbolji rezultati za svaku stazu.

Zaključak projekta ističe korištenje Unity okruženja za razvoj igre, s fokusom na programiranje uz istovremeno održavanje profesionalnog vizualnog dizajna. Projekt je donio iskustvo u upravljanju vremenom i zadacima, te važnost balansiranja tehničkog i kreativnog aspekta. Korišteni su gotovi 3D modeli kako bi se smanjilo vrijeme razvoja, dok je optimizacija kôda bila ključna za performanse. Razvoj je prepoznat kao složen proces, a iskustvo s C# stečeno kroz rad na projektu. Buduće nadogradnje mogu uključivati poboljšanja u zvučnim efektima i dodatnim funkcionalnostima.

Zaključak naglašava razvoj igre s Unity okruženjem, s fokusom na programerski aspekt i vizualni dizajn kako bi se postigao profesionalan i suvremen izgled. Projekt je donio iskustvo u upravljanju vremenom i zadacima, te balansiranju između tehničkih i kreativnih elemenata. Korišteni su gotovi 3D modeli s manje poligona, što je omogućilo optimizaciju performansi i smanjenje zahtjeva sustava. Projekt je prepoznat kao unaprediv, s preporukama za daljnju optimizaciju i proširenje funkcionalnosti.

2. SPECIFIKACIJA IGRE

Žanr igre: 3D trkaća igra

Ciljana platforma: PC

Opis igre: Igra pruža igraču posebno iskustvo gdje je uronjen u svijet šarenih boja. Igrač pokušava pobijediti botove koristeći prvo otključano besplatno vozilo i zarađivati novac pobjeđivanjem protivnika na početnoj stazi, sve dok ne prikupi dovoljno sredstava za otključavanje nove staze na kojoj ga očekuju teži protivnici. U svakom trenutku igrač može otključati novo vozilo koje će mu pomoći da lakše svlada trenutno otključane staze (slika 1).

Slika 1: Odabir vozila

Popis funkcionalnosti / korisničkih priča:

1. **Mogućnost odabira vozila s vlastitim karakteristikama:** Igrač može birati između četiri različita vozila koja se drugačije ponašaju na stazi.
2. **Odabir različitih trkaćih staza:** Igrač može birati između dvije staze različitog dizajna s različitim zahtjevima.
3. **Sustav napredovanja:** Igrač pobjeđivanjem zarađuje novac kojim može otključati staze i druga vozila.
4. **Korisničko sučelje s prikazom brzine:** Brzina u km/h prikaza je u donjem desnom rubu ekrana.
5. **Upravljanje vozilom preko tipkovnice:** Igrač vozilom upravlja strelicama i razmaknicom**.**
6. **Detekcija kolizije s drugim vozilima ili krajevima staze:** Vozilo igrača, vozila protivnika i krajevi staze sadrže komponente koje međusobno definiraju fiziku kolizije**.**
7. **Ubrzanje, kočenje i najveća brzina ovisna o vozilu:** Svako vozilo ima drugačiji iznos akceleracije i maksimalnu brzinu.
8. **Otključavanje novih vozila:** Na samom početku igre zaključana su sva vozila osim početnog.
9. **Sustav rangiranja rezultata utrka:** Na kraju svake dobivene utrke igračev rezultat uspoređuje se s pet najboljih rezultata od početka igračeve karijere
10. **Spremanje i učitavanje podataka:** Podatci kao zadnje odabrano vozilo, novčano stanje i najbolja vremena po potrebi se učitavaju i spremaju u unaprijed određenu datoteku na igračevom računalu.
11. **Prikaz najboljeg vremena kruga i ukupnog vremena utrke:** Igrač na prvoj stazi kao dio korisničkog sučelja vidi trenutno vrijeme, bez prikaza najboljeg kruga (prva staza realizirana je kao sprint utrka). Na ostalim stazama igraču je prikazano i vrijeme najboljeg kruga i trenutno vrijeme kruga.
12. **Početni izbornik s odabirom vozila, staze i postavkama:** Nakon učitavanja glavnog izbornika igrač može odabrati vozilo, pregledati najboljih pet rezultata

svake staze, onemogućiti ili omogućiti glazbu, izabrati vozilo ili izaći iz igre (uz prikladni izbornik dodatne potvrde gašenja igre). Početni izbornik mu također prikazuje trenutno novčano stanje.

13. **Implementacija protivnika protiv kojih se igrač utrkuje:** Ključni element igre koji povećava izazov i dinamiku utrkama. Ova funkcionalnost zahtijeva pažljivo osmišljavanje i programiranje ponašanja protivničke skripte koja kontrolira njihova vozila, kako bi stvorila izazov bio veći i realističniji.
14. **Jedinstveni zvučni efekti za svaku scenu:** Pomno izabrana glazba i zvučni efekti primijenjeni su na svaku scenu.
15. **Izbornik pauze:** Pritiskom na tipku za izlaz igrač pauzira igru i odabire završiti utrku, izaći iz igre, nastaviti igru ili započeti trenutnu utrku ispočetka.
16. **Dva skrivena elementa:** Igrač istraživanjem okoline može otkriti dva tajna elementa igre.
17. **Dva tajna postignuća:** Igrač može trajno otključati dvije titule koje se onda prikazuju u glavnom izborniku.
18. **Vozilo protivnika se pokvari pri sudaru:** Ako neki protivnik izgubi kontrolu nad vozilom, izleti sa staze i sudari s krajevima staze njegovo vozilo će se zaustaviti, a iz poklopca motora počet će se dimiti.
19. **Kontrolne točke za igrača:** Kako bi se osiguralo da igrač ne može preskakati dijelove staze i time skratiti ukupno vrijeme kruga, svaka staza ima kontrolne točke na kritičnim dijelovima kroz koje igrač mora proći kako bi završio utrku. U slučaju da igrač promaši kontrolnu točku, dužan je vratiti se i ispravno proći taj dio staze.
20. **Prikaz napretka kroz stazu kao postotak:** Određene su kontrolne točke namijenjene za računanje napretka kao postotak završenoga kruga te je izračunat postotak i prikazan kao dio korisničkoga sučelja.
21. **Prikaz posebne grafike za određene brzine vozila:** Ako vozilo pređe brzinu od sto kilometara na sat, posebna grafika je prikaza na ekranu igrača radi stvaranja osjećaja veće brzine. Nakon dvjesto kilometara na sat efekt postaje još izraženiji i mijenja mu se boja.
22. **Odbrojavanje prije početka utrke: „**3,2,1, kreni!“ prikazano je na sredini ekrana uz odgovarajuće zvučne efekte.

Napredovanje: Igrač na početku može voziti samo jedno vozilo i na jednoj stazi. Pobjede mu osiguravaju napredovanje otključavanjem drugih staza ili vozila. Svako vozilo ima bolje karakteristike od prethodnoga i olakšava igraču daljnje napredovanje.

Vizualni stil: Projekt koristi modele s umanjenim brojem poligona, ovakvi modeli s odabirom boja i tekstura daju igri poseban, blago crtani stil. Odabir boja i tekstura je razan.

Zvučni dizajn: Posebna pažnja obraćena je u osmišljanju zvučnog ambijenta, različita je glazba namijenjena za različite scene. Zvuk pobjede različit je onome gubitka. Isto tako, sama glazba odgovara stilu staze pa tako za prvu stazu igrač sluša veseliju glazbu arkadnog stila koja se postepeno ubrzava s vremenom. Glazba druge staze je misterioznija, sporija i odgovara crno bijelom stilu staze.

3. METODE I ALATI

3.1. C#

C# je programski jezik koji je razvio Microsoft i prvi je put predstavljen 2000. godine. Namijenjen je za razvoj različitih vrsta aplikacija, a posebno se ističe u okruženju .NET platforme. C# je objektno orijentiran jezik, usmjeren je na rad s objektima i klasama. Ovo omogućuje organiziranje kôda na logičan način i pomaže pri izradi modularnog kôda, što znatno olakšava razvoj složenih aplikacija.

Jedna od velikih prednosti C# jezika je njegova sličnost s drugim popularnim jezicima poput C++ i Jave. Ovo olakšava prijelaz za programere koji već imaju iskustva s tim jezicima. Također, C# dolazi s bogatom standardnom bibliotekom koja pruža mnoge korisne funkcionalnosti, uključujući mogućnost upita podataka putem LINQ (engl. *Language Integrated Query*).

Danas, C# se široko koristi u razvoju aplikacija za različite platforme zahvaljujući .NET Core radnom okviru, koji omogućava razvoj aplikacija za Windows, Linux i macOS. Također, C# je izuzetno popularan u razvoju igara, osobito s Unity softverom, jednim od najrasprostranjenijih alata za izradu 2D i 3D igara. Ovaj jezik pruža snažnu podršku za moderni razvoj, uključujući asinkrono programiranje i funkcionalno programiranje, što čini rad s njim učinkovitijim i fleksibilnijim.

3.2. UNITY

Unity je moćan i svestran razvojni alat koji se koristi za izradu 2D i 3D igara, simulacija i interaktivnih aplikacija. Izdan 2005. godine, Unity je do danas postao jedan od najpopularnijih i najšire korištenih alata u industriji razvoja igara. Njegova fleksibilnost, bogatstvo značajki i pristupačnost doprinose njegovoj popularnosti među programerima svih razina.

Ključne značajke Unityja:

1. Višenamjenska platforma: Unity podržava razvoj za brojne platforme uključujući Windows, macOS, Linux, iOS, Android, PlayStation i Xbox. Ova višestruka podrška omogućuje programerima da kreiraju aplikacije i igre koje se mogu pokretati na različitim uređajima, što povećava doseg i fleksibilnost proizvoda.

2. Intuitivno sučelje: Unity nudi intuitivno korisničko sučelje koje uključuje različite prozore za rad s scenama, hijerarhijom, skriptama i inspektorom objekata. Ovaj dizajn omogućava lako i brzo postavljanje elemenata igre, uređivanje scena, te prilagodbu i optimizaciju igre u stvarnom vremenu.

3. Bogatstvo alata i resursa: Unity dolazi s velikim brojem ugrađenih alata koji pomažu u stvaranju igara. To uključuje fiziku, animacije, grafiku, zvuk i sustave za upravljanje korisničkim sučeljem. Također, Unity trgovina dodataka nudi ogroman katalog resursa i alata koje programeri mogu koristiti za ubrzanje razvoja, uključujući modele, teksture, skripte i druge komponente.

4. Skriptiranje i programiranje: Programiranje u Unityju koristi C#, jedan od najpopularnijih programskih jezika. C# omogućuje razvoj složenih funkcionalnosti i ponašanja unutar igre, kao što su kontrole i logika igre. Unity nudi opširnu dokumentaciju i zajednicu koja pomaže programerima u učenju i rješavanju problema.

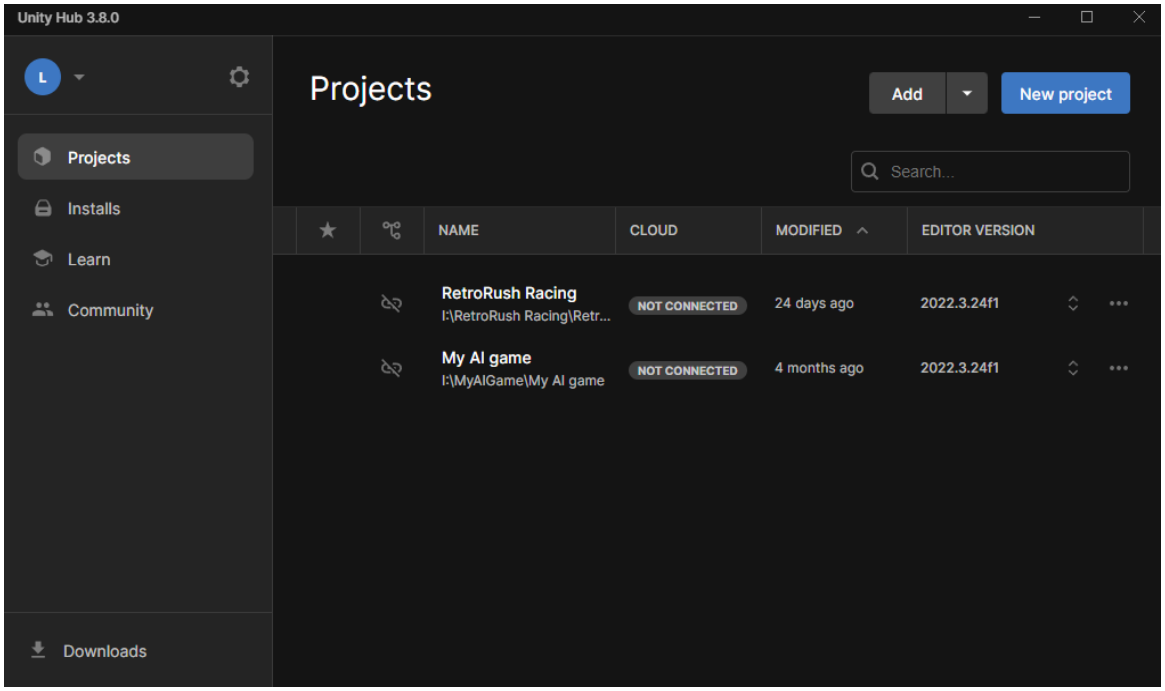
5. Vizualno skriptiranje: Za one koji nisu skloni pisanju kôda, Unity nudi opciju vizualnog programiranja pomoću alata poput Bolta. Ovaj pristup omogućava kreiranje logike igre koristeći grafičko sučelje za povezivanje čvorova, što može biti korisno za dizajnere i umjetnike koji žele brže prototipiranje ideja.

6. Renderiranje u stvarnom vremenu i virtualna stvarnost: Unity omogućava renderiranje u stvarnom vremenu, što znači da promjene u sceni mogu biti vidljive odmah. Ovo je posebno korisno za razvoj virtualne stvarnosti i proširene stvarnosti.

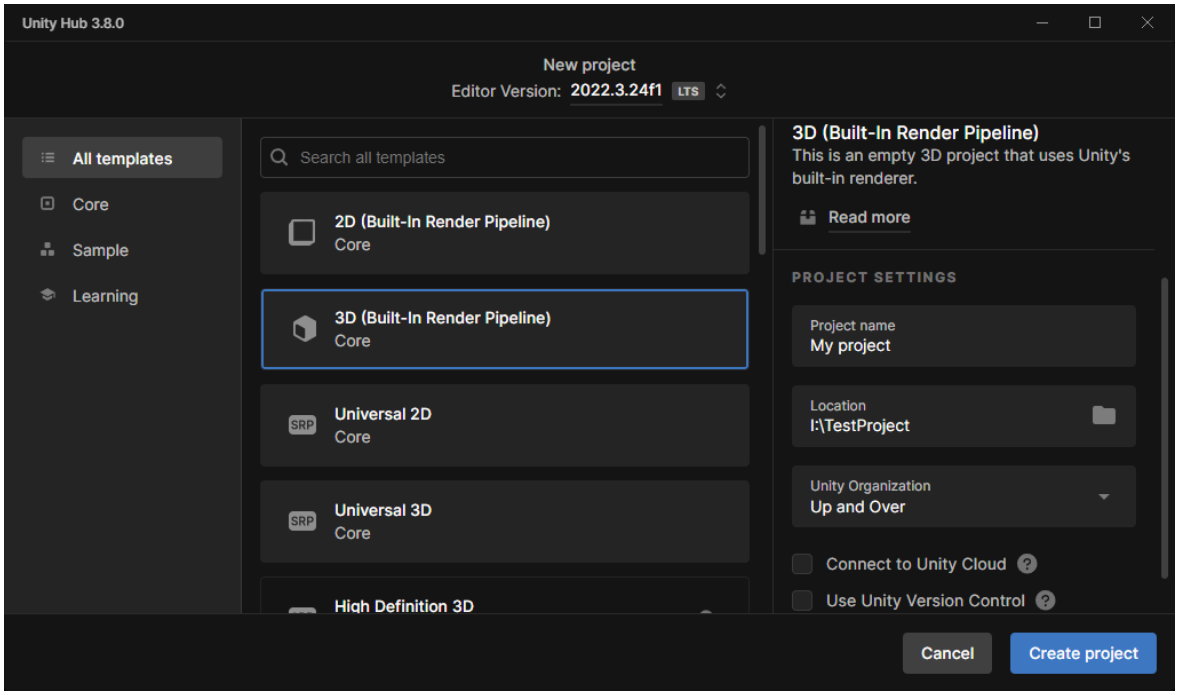
7. Dokumentacija i zajednica: Unity nudi opsežnu dokumentaciju, vodiče i resurse za podršku programerima. Zajednica je aktivna i uključuje brojne forume, vodiče i video materijale koji pomažu u učenju i rješavanju problema. Ova podrška čini Unity pristupačnijim i lakšim za korištenje, bez obzira na razinu iskustva.

3.2.1. KREIRANJE NOVOG PROJEKTA

Nakon pokretanja Unity Huba, klikom na botun *New project* (slika 2) otvaramo prozor za izradu novog projekta (slika 3).

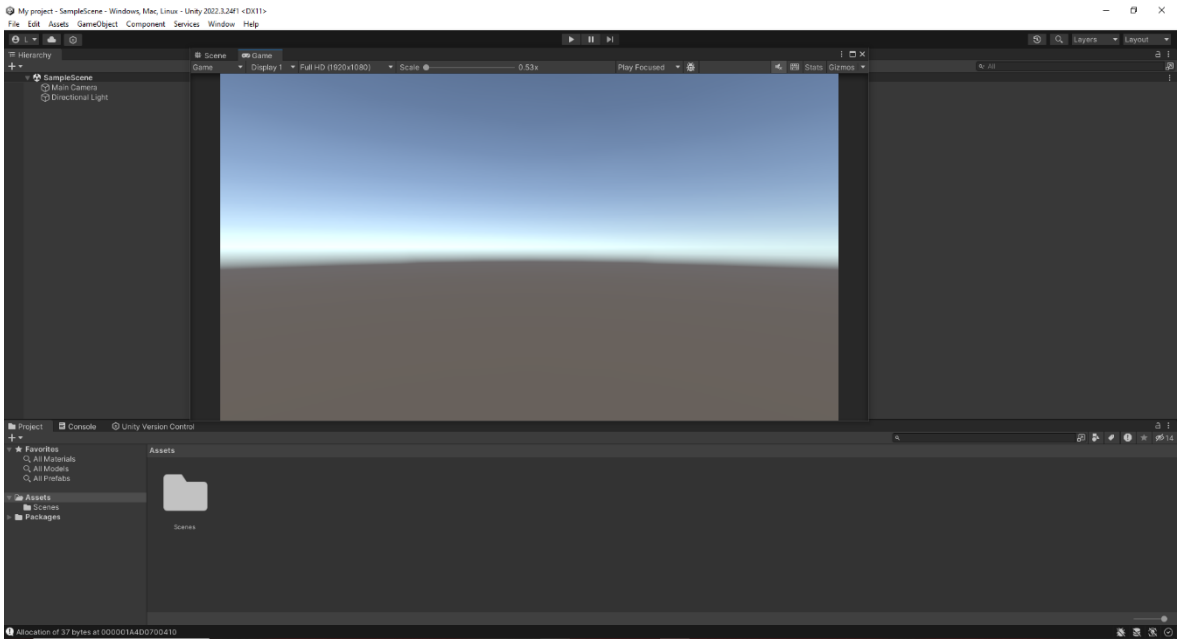


Slika 2: *Unity Hub* početni zaslon



Slika 3: *Unity, stvaranje novog projekta*

Plavi pravokutnik označava predložak koji nudi način renderiranja za naš projekt (slika 3), nakon toga, na desnoj trećini prozora odabiremo naziv projekta, lokaciju pohrane i organizaciju. Unity nam također nudi odabir povezivanja s Unity oblakom i Unity kontrolom verzije za potrebe kolaboracije. Klikom na botun *Create Project* otvara se prozor s novostvorenim projektom (slika 4).



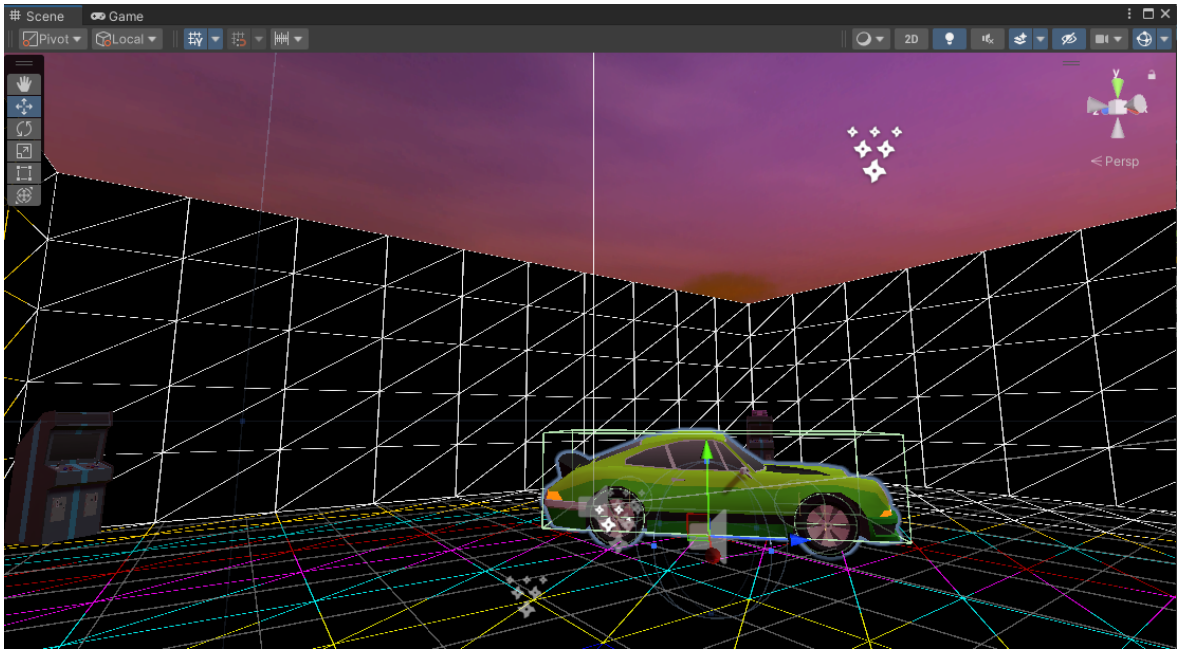
Slika 4: *Unity*, novostvoreni projekt

3.2.2. KORIŠTENJE SUČELJA

Sučelje Unity Editora je intuitivno, prilagodljivo i modularno. Ovo omogućava korisnicima da optimiziraju svoj radni prostor prema potrebama projekta i osobnim preferencijama. Sučelje se sastoji od nekoliko glavnih prozora i alata, svaki sa specifičnom funkcijom koja doprinosi ukupnom tijeku razvoja. Svaki prozor se po potrebi može premjestiti, otvoriti ili zatvoriti, a veličine prozora su promjenjive, omogućavajući dodatnu fleksibilnost pri organizaciji radnog okruženja. Raspored prozora se može spremiti kao konfiguracija i uključuje:

1. Pogled na scenu (engl. *Scene view*)

Pogled na scenu je osnovni prozor u kojem korisnici vizualno uređuju i sastavljaju 3D ili 2D scene (slika 5). Ovdje se dodaju i raspoređuju objekti, postavljaju svjetla, kamere i drugi elementi igre. Pogled omogućuje navigaciju kroz scenu koristeći miša i tipkovnicu, ali i manipulaciju objekata putem alata za premještanje, rotiranje i skaliranje. Ovaj pogled je interaktivan, što znači da se promjene odmah reflektiraju i u pogledu na igru, omogućujući korisnicima da lako vide kako njihova igra izgleda u stvarnom vremenu.



Slika 5: *Pogled na scenu*

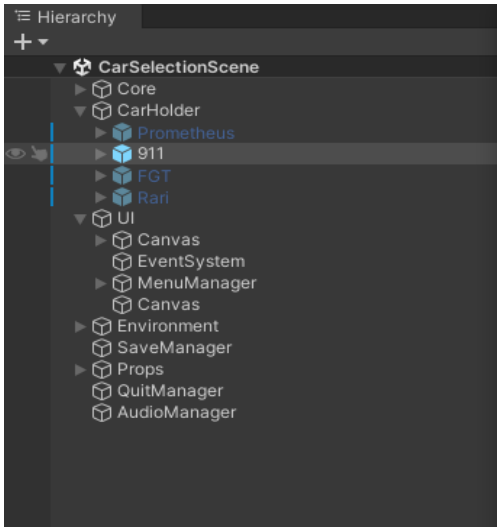
2. Pogled na igru (engl. *Game view*)

Pogled na igru prikazuje kako će scena izgledati i funkcionirati kada se igra pokrene (slika 6). To je simulacija konačnog proizvoda, gdje korisnici mogu testirati i pregledati svoju igru u stvarnom vremenu. Ovaj pogled se automatski ažurira kako igrač unosi promjene u scenu, što omogućuje neposrednu provjeru igrivosti i vizualnog stila.

Slika 6: *Pogled na igru*

3. Hijerarhija (engl. *Hierarchy*)

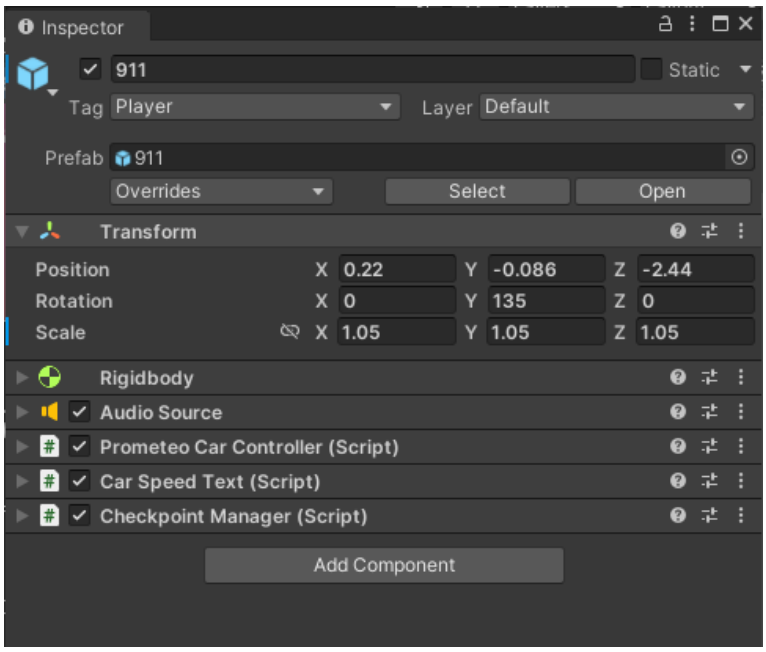
Prozor hijerarhije prikazuje sve objekte unutar aktivne scene u hijerarhijskom prikazu (slika 7). Ovdje su svi objekti organizirani u strukturu stabla koja prikazuje odnose između roditelja i djece među objektima. To omogućuje lako upravljanje kompleksnim scenama, grupiranje objekata i definiranje odnosa između njih. Hijerarhija je ključna za organizaciju elemenata u sceni i njihovom logičkom povezivanju.



Slika 7: *Hijerarhija*

4. Inspektor (engl. *Inspector*)

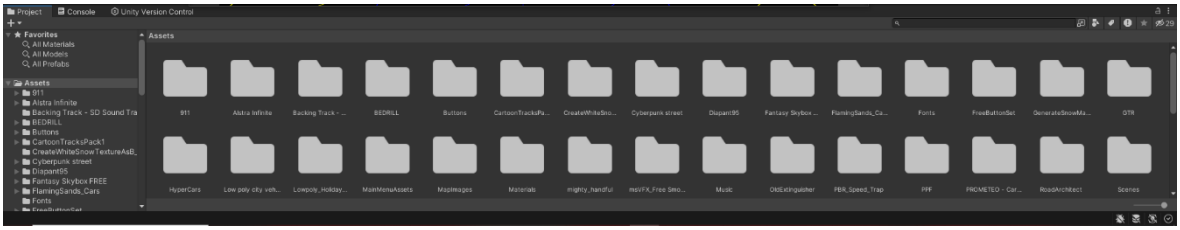
Inspektor je prozor u kojem korisnici mogu pregledavati i uređivati svojstva odabranih objekata u sceni (slika 8). Svaki objekt može imati različite komponente, poput transformacija, materijala, skripti i fizičkih svojstava, koje se sve mogu prilagoditi kroz Inspektor prozor. Ovaj prozor omogućuje detaljno podešavanje objekata bez potrebe za pisanjem kôda, što ubrzava proces razvoja igre.



Slika 8: *Inspektor*

5. Projekt (engl. *Project*)

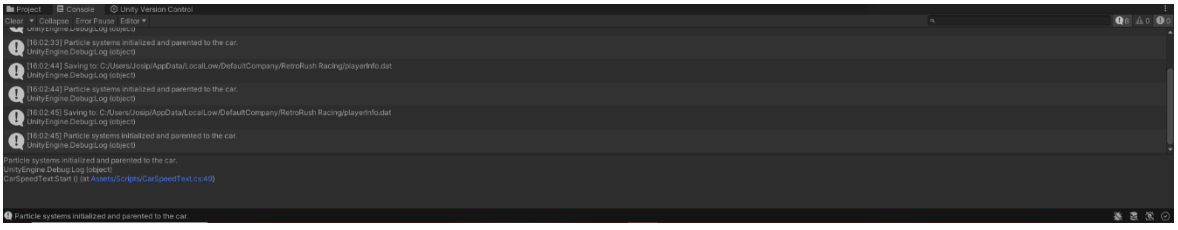
Projekt je prozor koji prikazuje sve datoteke i resurse koji su dostupni unutar projekta, uključujući skripte, modele, teksture, zvukove i predloške (slika 9). Prozor Projekt djeluje kao preglednik datoteka unutar Unityja, omogućujući korisnicima organizaciju resursa u mape i podmape. Kroz ovaj prozor, korisnici mogu povlačiti resurse direktno u pogled na scenu ili hijerarhiju, što ubrzava proces dodavanja novih elemenata u scenu.



Slika 9: *Projekt*

6. Konzola (engl. *Console*)

Konzola je prozor u kojem se prikazuju poruke sustava, upozorenja i greške tijekom rada na projektu (slika 10). Ovaj prozor je ključan za praćenje izvršavanja skripti i otklanjanje grešaka. Svaka pogreška ili problem u kôdu bit će ovdje zabilježen, pružajući korisnicima uvid u ono što treba popraviti ili optimizirati.



Slika 10: Konzola

3.2.3. VISUAL STUDIO

Visual Studio je jedno od najpopularnijih integriranih razvojnih okruženja (engl. *integrated development environment "IDE")* i ključni alat za razvoj aplikacija u raznim programskim jezicima, uključujući C#. U kontekstu Unity razvoja, Visual Studio se koristi kao primarno okruženje za pisanje i uređivanje skripti, testiranje kôda te praćenje i ispravljanje pogrešaka.

Integracija s Unityjem

Kada se radi u Unityju, Visual Studio služi za razvoj skripti napisanih u C#. Unity automatski prepoznaje Visual Studio kao zadani IDE i omogućava jednostavan prijelaz između Unity sučelja i Visual Studia. Kada se dvaput klikne na skriptu unutar Unityja, Visual Studio se automatski otvara i omogućuje uređivanje kôda.

Jedna od ključnih prednosti korištenja Visual Studia u Unity razvoju igara je integracija s Unity *API-jem*. Visual Studio nudi podršku za automatsko dovršavanje kôda (IntelliSense). Ovo olakšava pisanje kôda jer omogućuje brzo pronalaženje metoda, klasa i varijabli koje su dio Unity *API-ja*. Osim toga, Visual Studio automatski prepoznaje sve Unity klase i komponente, što omogućava bržu navigaciju kroz projekt.

Alati za ispravljanje pogrešaka (engl. *debugging*)

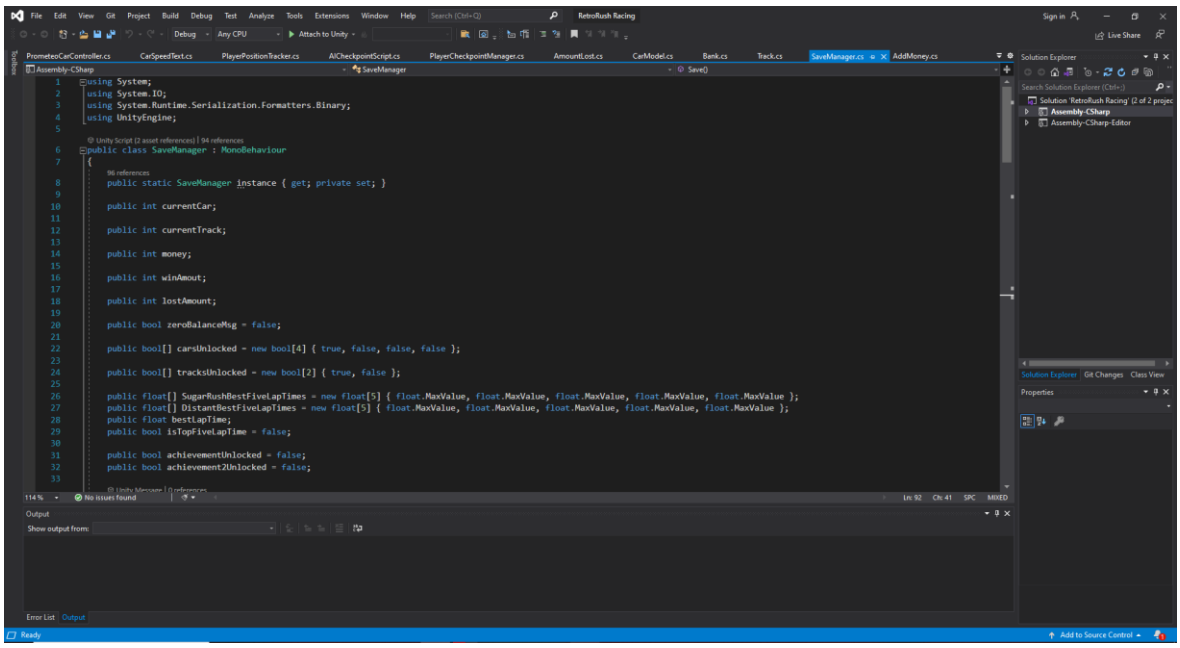
Visual Studio nudi moćne alate za ispravljanje pogrešaka koji su neophodni u razvoju igara. Budući da je ispravljanje pogrešaka u igrama često složen proces, mogućnost postavljanja točaka prekida (engl. *breakpoints*) u kôdu i praćenja tijeka izvršavanja skripti u stvarnom vremenu je neprocjenjiva. Kada se postavi točka prekida, Unity se pauzira tijekom izvršavanja igre u točno određenom trenutku, omogućujući pregled stanja varijabli, istraživanje problematičnih dijelova kôda i pronalaženje rješenja.

Upravljanje verzijama (engl. *Version Control*)

Za veće projekte, Visual Studio nudi integraciju s različitim sustavima za upravljanje verzijama, kao što su Git i SVN. To omogućava timski rad na Unity projektima, gdje više programera može raditi na istom projektu, dijeliti izmjene kôda te pratiti povijest izmjena.

Personalizacija i proširenja

Visual Studio se može personalizirati kroz različite teme, postavke i ekstenzije (slika 11). Postoji niz ekstenzija dostupnih putem Visual Studio trgovine koje dodatno proširuju funkcionalnost Visual Studia u Unity razvoju. Neke popularne ekstenzije uključuju Unity alate i ReSharper, koji pružaju dodatne mogućnosti za automatizaciju zadataka i analizu kôda.



Slika 11: *Visual Studio sučelje*

4. IMPLEMENTACIJA IGRE

Razvoj igre odvijao se u sljedećim fazama:

1. Stvaranje terena i dodavanje prvog vozila u scenu
2. Dodavanje skripte za kameru i skripte koja pokreće vozilo igrača
3. Stvaranje staza
4. Izbornik vozila i staza
5. Izrada scena pobjede i poraza
6. Izrada skripte za protivnike
7. Izrada skripti za upravljanje utrkom
8. Korisničko sučelje
9. Prikaz pet najboljih rezultata svake staze

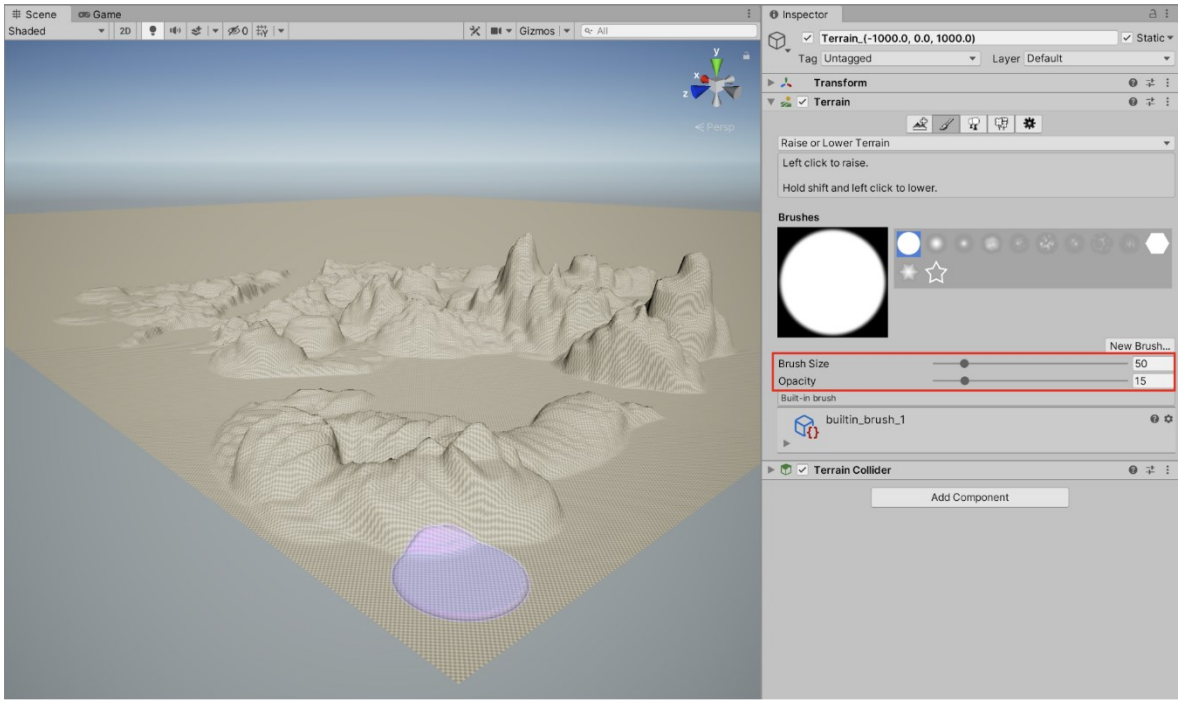
4.1. STVARANJE TERENA I DODAVANJE PRVOG VOZILA U SCENU

U sklopu izrade terena za dvije staze unutar igre korišten je uređivač terena, moćan alat unutar Unity platforme koji omogućuje stvaranje kompleksnih i detaljnih okoliša (slika 12). Ovaj alat nudi niz funkcionalnosti, uključujući alate za glatko podizanje i spuštanje visine terena, što omogućava oblikovanje prirodnog i vizualno privlačnog pejzaža.

Proces oblikovanja uključuje pažljivo korištenje alata za modeliranje terena kako bi se postigla željena topografija, dok su dijelovi predviđeni za cestu izravnani, osiguravajući ravnu površinu predviđenu za modele ceste.

Unity uređivač terena omogućuje i jednostavnu integraciju različitih tekstura, vegetacije i ostalih detalja, čime se dodatno unapređuje realističnost i vizualna konzistentnost okoliša. Ovaj pristup rezultirao je funkcionalnim stazama koje doprinose boljem korisničkom iskustvu i cjelokupnom vizualnom identitetu igre. Pravokutnik s crvenim stranicama (slika 12) označava opcije veličine kista (engl. *Brush Size*)

i neprozirnosti (engl. *Opacity*). Veličina kista određuje veličinu prostora na koje primjenjujemo efekt, a neprozirnost kontrolira intenzitet djelovanja kista na tom području.



Slika 12: *Modulacija terena*

4.2. DODAVANJE SKRIPTE ZA KAMERU I SKRIPTE KOJA POKREĆE VOZILO IGRAČA

Skripta za upravljanje vozilom igrača

Za kontrolu vožnje odabrana je skripta **`Prometeo Car Controller`**, besplatno preuzeta s Unity trgovine dodataka koja je cjelovito napisana s lako razumljivim kôdom. Ova skripta je odabrana zbog svoje visoke kvalitete fizičkih simulacija koje nadmašuju mogućnosti koje bi početnik mogao postići pisanjem vlastitog kôda. `Prometeo Car Controller` pruža preciznu i realističnu simulaciju vožnje vozila, što značajno doprinosi autentičnosti i uživljenosti u igri. Zbog jednostavnosti i čitljivosti kôda, omogućena je i laka prilagodba specifičnim potrebama projekta, čime se osigurava da mehanika vožnje odgovara željenom iskustvu igre.

Skripta za kameru (engl. *Camera Follow*)

Za praćenje vozila napisana je skripta za kameru, koja je dizajnirana da bi osigurala glatko i dinamično praćenje vozila iz različitih perspektiva. Ova skripta omogućuje prijelaz između klasičnog pogleda na vozilo i ptičje perspektive, što igraču nudi veću fleksibilnost i različite načine doživljaja igre.

Odabir ovog pristupa pisanju skripte temelji se na potrebi za jednostavnošću i efikasnošću, gdje su osnovne funkcionalnosti poput glatkog pomicanja i rotacije kamere te prilagodbe na različite scene ključne za iskustvo igranja. Implementacijom opcije za prelazak na ptičju perspektivu, skripta omogućuje brz prijelaz između različitih perspektiva.

Ovakav pristup kodiranju odabran je zbog svoje prilagodljivosti i jednostavnosti u korištenju, što olakšava buduće nadogradnje i prilagodbe bez potrebe za opsežnim izmjenama u osnovnoj strukturi kôda. Korištenjem jednostavnih, ali učinkovitih metoda kao što su ***Lerp*** za glatko pomicanje i rotaciju, skripta omogućuje visok stupanj kontrole nad ponašanjem kamere u različitim načinima igranja.

```
void HandleTopDownView()

    {

        transform.position = Vector3.Lerp(transform.position, topDownPosition,
moveSmoothness * Time.deltaTime);

        Quaternion topDownQuat = Quaternion.Euler(topDownRotation);

        transform.rotation = Quaternion.Lerp(transform.rotation, topDownQuat,
rotSmoothness * Time.deltaTime);

    }
```

Ispis 1: Upravljanje pozicijom kamere za pogled iz ptičje perspektive

4.3. STVARANJE STAZA

Prilikom izrade staza za igru, pažljivo su odabrani modeli kako bi se postigao željeni vizualni stil i optimalne performanse igre. Modeli su preuzeti s Unity trgovine dodataka,

gdje su odabrani modeli s niskim brojem poligona kako bi se smanjila veličina igre i olakšala izvedba na različitim računalima.

Za izradu staze, korišten je set modela iz Unity trgovine dodataka koji nudi različite krivine i pravce. Ovaj set je pažljivo postavljen tako da staza bude različita i zanimljiva, pružajući igraču zabavno i dinamično iskustvo vožnje. Prilikom raspoređivanja okuka i pravaca, cilj je bio stvoriti stazu koja ne samo da je izazovna, već i vizualno privlačna.

Objekti poput barijera, palmi i guma su strateški postavljeni na stazi kako bi poboljšali igračko iskustvo. Barijere su postavljene uzduž kritičnih točaka staze, s ciljem sprječavanja izlaska vozila sa staze i povećanja razine izazova. Ove barijere su raspoređene na promišljen način kako bi se osiguralo da igrač ne vara u postavljanju vremena kruga (slika 13).

Osim barijera, dodatni objekti poput palmi i guma su smješteni tako da budu vizualno dojmljivi, pridonoseći ukupnom vizualnom dojmu igre. Palme su postavljene na planiranim pozicijama kako bi dodale poseban izgled i stvarale atmosferu koja odgovara tematici igre.

Ovakav pristup osigurava da staza bude ne samo funkcionalna i izazovna, već i vizualno privlačna, čime se poboljšava ukupno iskustvo igre i povećava uživljenost igrača u virtualni svijet.

Slika 13: Staza s barijerama

4.4. IZBORNIK VOZILA I STAZA

U procesu izrade izbornika za odabir vozila i staza, primijenjen je pristup koji omogućava korisnicima jednostavno i intuitivno upravljanje odabirom vozila i staza.

Izbornik za odabir vozila

Za implementaciju izbornika za odabir vozila korištena je skripta naziva ***`CarSelection`***, koja omogućava dinamičko mijenjanje vozila i upravljanje kupovinom. Ovaj pristup je izabran zbog svoje fleksibilnosti i jednostavnosti.

`CarSelection` skripta omogućava korisnicima da koriste prethodni i sljedeći botun za navigaciju kroz dostupna vozila (slika 14). Pritom se svako vozilo prikazuje na temelju trenutnog indeksa, što omogućava vizualizaciju odabranog vozila. Ovaj pristup osigurava da se korisnici mogu jednostavno kretati između vozila klikom na strelicu desno ili lijevo, a skripta koristi ***`SaveManager`*** klasu za pohranu i učitavanje stanja vozila, čime se osigurava da korisnička konfiguracija ostaje dosljedna između sesija igre.

Ova skripta nudi jasan način za odabir i kupovinu vozila. Korištenjem ***`SaveManager`*** skripte za upravljanje stanjem vozila i novcem igrač osigurava se integritet podataka i održava korisnički napredak.

Slika 14: Izbornik vozila

```
public void ChangeCar(int _change)
    {
        currentCar += _change;

        if (currentCar > transform.childCount - 1)
            currentCar = 0;
        else if (currentCar < 0)
            currentCar = transform.childCount - 1;

        SaveManager.instance.currentCar = currentCar;
        SaveManager.instance.Save();
        SelectCar(currentCar);
    }
```

Ispis 2: Promjena auta aktivnog u sceni

Skripta ***TrackDisplay*** upravlja prikazom, odabirom i otključavanjem staza. Ovaj pristup omogućava predstavljanje staza kroz opise i 2D *spriteove (*dvodimenzionalne slike ili animacije koje se koriste za prikaz objekata u video igrama i drugim digitalnim medijima*)*, čime se omogućuje igraču da donese odluku o odabiru ili otključavanju staze (slika 15).

Skripta koristi ***Track*** objekt za pohranu informacija o svakoj stazi, uključujući naziv, opis, cijenu i status otključavanja. Ovaj pristup omogućava prikaz bitnih informacija o stazi, što doprinosi organizaciji i preglednosti.

Osim toga, ***TrackDisplay*** skripta koristi igraj botun ili kupi ako staza nije otključana. Klasa ***SaveManager*** koristi se za upravljanje stanjem staze (otključana ili zaključana) kao i za prikaz trenutnog novčanog stanja.

Slika 15: Izbornik staze

```
public void DisplayTrack(Track _map)
    {
        trackName.text = _map.trackName;
        trackDescription.text = _map.trackDescription;
        trackImage.sprite = _map.trackImage;
        bool trackUnlocked = _map.isUnlocked;
        lockIcon.SetActive(!trackUnlocked);
        playButton.interactable = trackUnlocked;

        if (trackUnlocked)
        {
            trackImage.color = Color.white;
            playButton.gameObject.SetActive(true);
            buyButton.gameObject.SetActive(false);
            playButton.onClick.RemoveAllListeners();
            playButton.onClick.AddListener(() => {
                LoadScene(_map.scenePath);
            });
        }
        else
        {
            trackImage.color = Color.gray;
            playButton.gameObject.SetActive(false);
            buyButton.gameObject.SetActive(true);
            buyButton.interactable = (SaveManager.instance.money >= _map.price);

            Text buyButtonText = buyButton.GetComponentInChildren<Text>();

            if (buyButtonText != null)
            {
                buyButtonText.text = _map.price + "HRK";
            }

            buyButton.onClick.RemoveAllListeners();
            buyButton.onClick.AddListener(() => BuyTrack(_map));
        }
```

Ispis 3: Promjena staze u izborniku

Korištenje *TrackDisplay* skripte za prikaz i odabir staza pruža korisnicima informativan i vizualno privlačan prikaz.

```
class PlayerData_Storage
    {
        public int currentCar;
        public int currentTrack;
        public int money;
        public int winAmout;
        public int lostAmount;
        public bool zeroBalanceMsg;
        public bool[] carsUnlocked;
        public bool[] tracksUnlocked;
        public float[] SugarRushBestFiveLapTimes;
        public float[] DistantBestFiveLapTimes;
        public float bestLapTime;
        public bool isTopFiveLapTime = false;
        public bool achievementUnlocked = false;
        public bool achievement2Unlocked = false;
    }
```

Ispis 4: Podatci koji se serijaliziraju i deserijaliziraju iz klase `SaveManager`

4.5. IZRADA SCENA POBJEDE I PORAZA

U sljedećoj fazi projekta, kreirane su scene poraza i pobjede koje se prikazuju nakon završetka utrke, ovisno o ishodu. Ove scene su dizajnirane s ciljem pružanja korisnicima jasnih povratnih informacija o njihovom uspjehu ili neuspjehu, uz integraciju odgovarajućih zvučnih efekata.

Pobjednička scena

Pobjednička scena aktivira se u slučaju pobjede igrača nad protivnicima (slika 16). Ova scena je osmišljena kako bi igraču pružila pozitivan i nagrađujući osjećaj prilikom uspjeha. Pri učitavanju scene, pokreće se veseli zvučni efekt.

Na ekranu se prikazuje nekoliko ključnih informacija:

1. **Dobiveni iznos** - Igraču se prikazuje koliko je novca zaradio pobjedom u utrci.
2. **Novo stanje novca** - Ažurirano stanje novca nakon dodavanja dobitka, čime se osigurava da igrač ima pregled nad svojim financijama.
3. **Vrijeme utrke** - Prikazuje se ukupno vrijeme koje je igraču trebalo da završi utrku, uz naznaku ako je to vrijeme ušlo u pet najboljih rezultata.
4. **Poruka o pobjedi** - Poruka koja igraču potvrđuje da je pobijedio utrku.
5. **Botun za povratak na glavni izbornik** - Omogućuje igraču da se vrati na glavni izbornik igre, pružajući mu mogućnost nastavka igre.

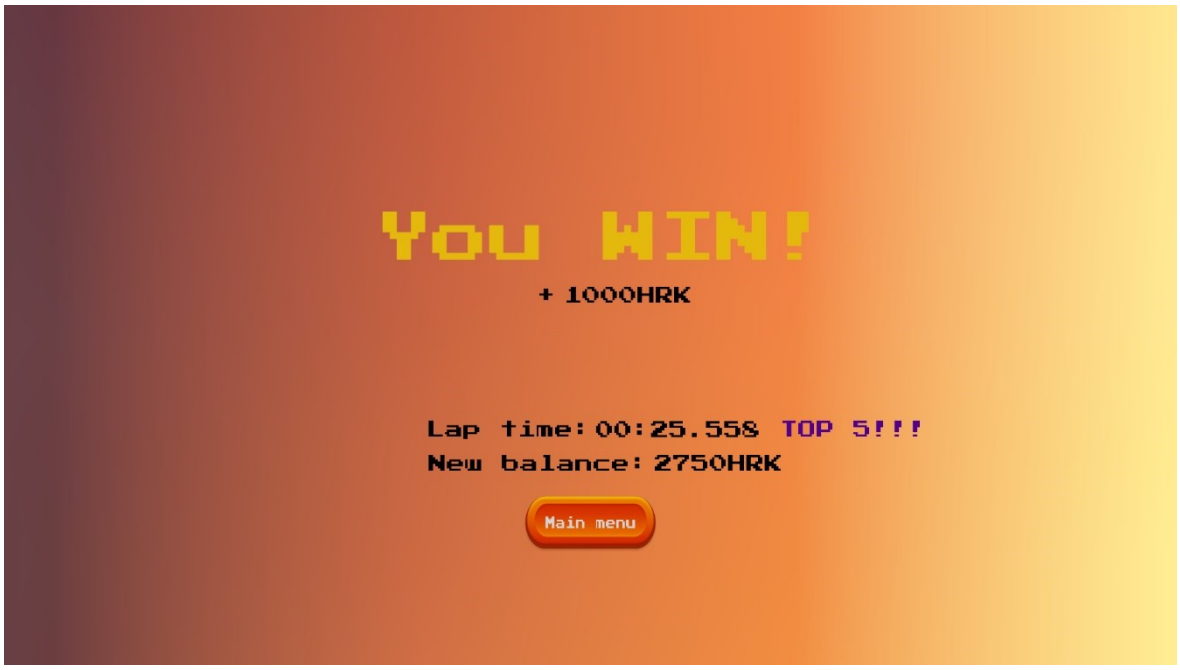
Ovakav pristup dizajnu scene omogućava igraču da na jednostavan i informativan način dobije povratne informacije o uspjehu, istovremeno pružajući osjećaj postignuća i napretka u igri.

Scena poraza

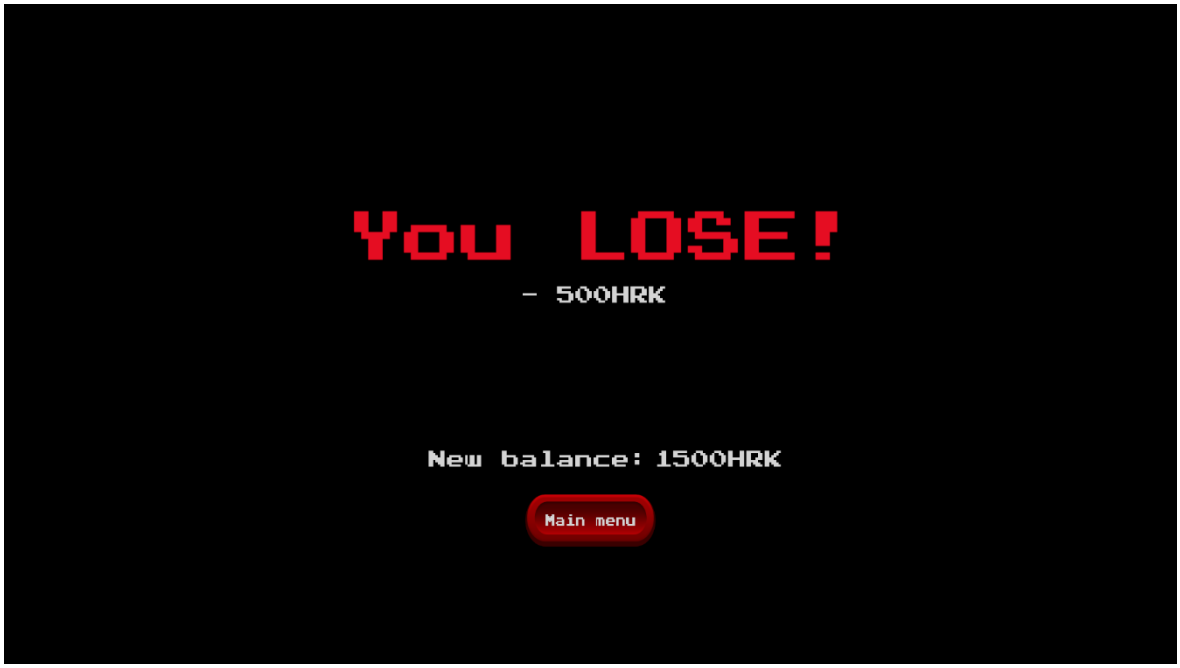
Scena se aktivira kada igrač izgubi utrku. Cilj ove scene je pružiti mu jasnu povratnu informaciju o neuspjehu (slika 17). Prilikom učitavanja scene, pokreće se odgovarajući tužni zvučni efekt.

U sceni se prikazuju sljedeći elementi:

1. **Izgubljeni iznos novca** – Igraču se prikazuje koliko je novca izgubio zbog poraza u utrci.
2. **Novo novčano stanje** - Ažurirano stanje novca nakon oduzimanja izgubljenog iznosa, što omogućava igraču da prati svoje novčano stanje.
3. **Botun za povratak na glavni izbornik** - Služi za jednostavan povratak na glavni izbornik, gdje igrač može odlučiti hoće li ponoviti utrku, promijeniti vozilo ili stazu.



Slika 16: Pobjednička scena



Slika 17: Scena poraza

4.6. IZRADA SKRIPTE ZA PROTIVNIKE

Izrada skripte koja će upravljati zahtjevnim, ali ne i nemoguće teškim protivnicima predstavljalo je velik izazov. Odlučeno je da će se skripta optimizirati za jednog protivnika i njegova skripta će predstavljati najbržeg mogućeg protivnika dok će ostali protivnici biti realizirani kao sporije, lošije verzije potonjeg. Nakon ekstenzivnog istraživanja načina implementacije odlučeno je da će se protivnici orijentirati prema nevidljivim objektima pozicioniranim duž cijele staze i da će se njihove koordinate uspoređivati s koordinatama nevidljivih, brojčano sortiranih objekata te da će se koordinate vozila usmjeravati prema koordinatama sljedećeg objekta u redoslijedu.

Ovakav pristup odabran je zbog njegove jednostavnosti i efikasnosti u simuliranju realnog ponašanja vozača na stazi. Korištenjem brojčano sortiranih objekata omogućeno je da skripta precizno detektira trenutnu poziciju vozila i odredi optimalnu putanju prema sljedećoj kontrolnoj točki. Ova metoda ne samo da poboljšava realizam i fluidnost kretanja protivnika, već omogućava i jednostavno proširivanje i prilagodbu skripte za različite konfiguracije staza. Bitan dio ove skripte je i upravljanje kolizijom s barijerama ili drugim objektima koji čine granice staze. U takvoj situaciji, protivnička vozila će se zaustaviti i početi će se dimiti iz motornog prostora (efekt dima radi se van skripte te poziva po želji i potrebi u programu). Implementacija ovog sustava rezultirala je protivnicima koji pružaju izazovan, ali pošten i predvidljiv način igranja, ključan za održavanje balansa i interesa igrača tijekom dužih perioda igranja.

```
private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Tree") ||
            collision.gameObject.CompareTag("Tyre") ||
            collision.gameObject.CompareTag("Barrier"))
        {
            StopCar();
            CreateSmoke();
        }
    }
    private void StopCar()
    {
        isStopped = true;
        rearLeftWheel.motorTorque = 0f;
        rearRightWheel.motorTorque = 0f;
        rearLeftWheel.brakeTorque = brakeForce;
        rearRightWheel.brakeTorque = brakeForce;
        frontLeftWheel.brakeTorque = brakeForce;
        frontRightWheel.brakeTorque = brakeForce;
        rb.velocity = Vector3.zero;
        rb.angularVelocity = Vector3.zero;
    }
private void CreateSmoke()
    {
        if (smokePrefab != null && smokePosition != null)
        {
            GameObject smoke = Instantiate(smokePrefab, smokePosition.position,
Quaternion.identity);
            smoke.transform.parent = this.transform;
        }
    }
```

Ispis 5: Detekcija kolizije s barijerama i pripadajuće ponašanje vozila

4.7. IZRADA SKRIPTI ZA UPRAVLJANJE UTRKOM

Skripte koje upravljaju raznim aspektima utrke su: skripta koja upravlja detekcijom protivničke pobjede, skripte koje upravljaju novčanim stanjem i prikazom prilikom kraja utrke, izbornik pauze (s opcijama ponovnog početka utrke, kraja utrke i gašenja igre). Izbornik pauze realiziran je minimalističkim pristup, sam program je jednostavan kako bi se izvršavanje naredbi izvelo što brze, a vizualni dizajn izbornika jednostavan je, intuitivan i pamtljiv (slika 18).

```
    public void ResumeGame()
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        isPaused = false;
    }
    void PauseGame()
    {
        pauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        isPaused = true;
    }

    void QuitGame()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
            Application.Quit();
#endif
    }
    void ReloadScene()
    {
        Time.timeScale = 1f;
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
```

Ispis 6: Neke od funkcija koje upravljaju izbornikom

Slika 18: Izbornik pauze

4.8. KORISNIČKO SUČELJE

Korisničko sučelje podrazumijeva prikaz sljedećih elemenata: trenutna brzina vozila kojim igrač upravlja, vrijeme proteklo od početka utrke, trenutni krug, najbolje vrijeme kruga (za drugu stazu s više od jednog kruga) što je vidljivo na slici 19, napredak kao postotak (za prvu stazu s jednim krugom, realiziranu kao sprint utrku) prikazano slikom 20.

Slika 19: Korisničko sučelje (druga staza)

Slika 20: Korisničko sučelje (prva staza)

U nastavku su objašnjene metode početka `Start` i ažuriranja *`Update`* zaslužne za prikaze korisničkog sučelja.

```
void Start()
    {
        Car = GetComponent<Rigidbody>();
        string sceneName = SceneManager.GetActiveScene().name;
        checkpointManager = FindObjectOfType<CheckpointManager>();
        cameraController = FindObjectOfType<CameraFollow>();
        if (sceneName == "Distant")
        {
            CarSpeed.color = new Color32(0, 0, 0, 255);
            SpeedUnit.color = new Color32(0, 0, 0, 255);
            Debug.Log("Speed Text color changed.");
        }
        else if (sceneName == "MapSelectionScene" || sceneName == "CarSelectionScene")
        {
            CarSpeed.enabled = false;
            SpeedUnit.enabled = false;
        }
        particleSystem100.Stop();
        particleSystem200.Stop();
        if (particleSystem100.transform.parent != Car.transform)
        {
            particleSystem100.transform.SetParent(Car.transform);
        }
        if (particleSystem200.transform.parent != Car.transform)
        {
            particleSystem200.transform.SetParent(Car.transform);
        }
        Debug.Log("Particle systems initialized and parented to the car.");
    }
```

Ispis 7: Metoda `Start` korisničkog sučelja

Ova se metoda izvršava na početku igre i služi za inicijalizaciju različitih elemenata unutar scene. Prvo dohvaća fizičku komponentu vozila kako bi se moglo kontrolirati njegovo ponašanje. Zatim, ovisno o trenutnoj sceni u kojoj se igra nalazi, prilagođava izgled korisničkog sučelja, poput boje teksta koji prikazuje brzinu vozila, kako bi odgovarala izgledu scene. Naime, druga staza nazvana *Distant* realizirana je kao i prva staza, ali je na kameru postavljen crno-bijeli filter kako bi igračev prikaz bio u skladu s idejom monotone atmosfere staze. Ovisno o učitanoj sceni, sakriva elemente korisničkog sučelja koji nisu potrebni (prikaz najboljeg kruga za prvu stazu i prikaz napretka kao postotak). Također, metoda postavlja i zaustavlja sustav čestica (engl. *particle systems*) te ih povezuje s vozilom kako bi se pratilo njegovo kretanje.

```
private void Update(){
        if (checkpointManager != null && !checkpointManager.IsCountdownFinished())
            { CarSpeed.enabled = false;
            SpeedUnit.enabled = false; }
        else {
            CarSpeed.enabled = true;
            SpeedUnit.enabled = true;
            float currentSpeed = Car.velocity.magnitude * 5f;
            CarSpeed.text = currentSpeed.ToString("0");
            if (currentSpeed > 400 && SaveManager.instance != null &&
!SaveManager.instance.achievement2Unlocked)
              { SaveManager.instance.achievement2Unlocked = true;
                SaveManager.instance.Save();
                Debug.Log("Speed Demon unlocked"); }
            if (!cameraController.isTopDownView){
                UpdateParticleSystems(currentSpeed);}
            else {
                StopParticleSystems(); }
            if (currentSpeed >= 100 && currentSpeed <= 200) {
                if (!particleSystem100.isPlaying){
                    particleSystem100.Play(); }
                if (particleSystem200.isPlaying){
                    particleSystem200.Stop(); }}
            else if (currentSpeed > 200){
                if (!particleSystem200.isPlaying){
                    particleSystem200.Play(); }
                if (particleSystem100.isPlaying){
                    particleSystem100.Stop(); }}
            else {
                if (particleSystem100.isPlaying){
                    particleSystem100.Stop();}
                if (particleSystem200.isPlaying){
                    particleSystem200.Stop();}}}}
```

Ispis 8: Metoda *Update* korisničkog sučelja

Metoda `Update` (ispis 8) ažurira stanje igre u stvarnom vremenu na temelju trenutne situacije i parametara igre. Prvo provjerava je li odbrojavanje na početku igre završeno; ako nije, sakriva prikaz brzne vozila. Kada odbrojavanje završi, prikaz brzine se aktivira, a trenutna brzina vozila se računa i prikazuje na ekranu. Ako vozilo postigne brzinu veću od 400, igrač otključava jedno od dva tajna postignuća, koje se tada sprema i kasnije prikazuje u glavnom izborniku.

Kôd također upravlja sustavima čestica koji su vezani uz brzinu vozila. Ako se brzina vozila nalazi unutar određenog raspona, odgovarajući sustavi čestica će se pokrenuti ili zaustaviti. Ako se kamera nalazi u ptičjoj perspektivi, svi sustavi čestica se zaustavljaju kako bi se izbjeglo ometanje prikaza.

4.9. PRIKAZ PET NAJBOLJIH REZULTATA SVAKE STAZE

Nakon svake pobjede igrača, rezultat najboljeg kruga se zapisuje u datoteku skupa s ostalim varijablama (ispis 4). Rezultat (kao jedna `float` varijabla) se prvo uspoređuje s pet najboljih rezultata tako da se traži najveći element, a potom ga se uspoređuje s novoostvarenim rezultatom. U slučaju da je novi rezultat manji onda se novi rezultat sprema na mjesto starog rezultata i niz se sortira kako bi se rezultati spremili pravilnim redoslijedom.

Igrač preko glavnog izbornika može pritisnuti na botun koji mu otvara prikaz pet najboljih rezultata (slika 21). Rezultati se iščitavaju kao `float` varijable te se jednostavnim algoritmom pretvaraju u vremenski format oblika „minute:sekunde.milisekunde“. Rezultati se igraču prikazuju iterativnim prolaskom kroz niz podataka, pri čemu se odgovarajući element niza ispisuje na odgovarajući tekstualni element na ekranu. Tekstualni elementi na ekranu također su organizirani u obliku niza.

Slika 21: Scena sa prikazom pet najboljih rezultata

5. ZAKLJUČAK

Za realizaciju projekta korišteno je Unity razvojno okruženje, a fokus je bio na programerskom razvoju igre, uz istovremeno pridavanje pažnje vizualnom dizajnu kako bi gotov projekt izgledao profesionalno i suvremeno. Osim tehničkih vještina, rad na ovom projektu donio je i iskustvo u upravljanju projektom, posebno u smislu vremenske organizacije i prioritiziranja zadataka. Kroz proces razvoja igre, naglašena je važnost balansiranja između tehničkog aspekta i kreativnog dizajna, kako bi se postigao rezultat koji je i funkcionalan i vizualno zadovoljavajući. Također, suočavanje s izazovima kao što su optimizacija performansi i korištenje novog programskog jezika, dodatno je osnažilo sposobnosti rješavanja problema i prilagodljivosti.

Unity je omogućio početnicima u razvoju igara da jednostavno stvore cjelovit projekt, zahvaljujući trgovini dodataka i velikoj količini besplatnih vodiča dostupnih na internetu koji pokrivaju sve aspekte razvoja. Stvaranje novoga projekta je iznimno lako i intuitivno kao i povezivanje skripti s objektima preko sučelja Unity Editora.

Za stvaranje igre korišteni su gotovi 3D modeli sa smanjenim brojem poligona iz Unity trgovine dodataka, čime je omogućeno usmjeravanje pažnje na programerski aspekt razvoja igre te završetak projekta unutar zadanog vremenskog ograničenja. Korištenje modela s manjim brojem poligona i fokusiranje na optimizaciju kôda smanjilo je ukupnu veličinu igre kao i zahtjeve sustava koji je pokreće.

Razvoj igara prepoznat je kao složena industrija, u kojoj dizajniranje razina, testiranje, zvuk, dokumentacija, 3D dizajniranje, planiranje i ostali ključni aspekti predstavljaju velik izazov za samo jednog programera.

Kôd je pisan u programskom jeziku C#, s kojim nije bilo prethodnog iskustva, no iskustvo s C++ i drugim objektno orijentiranim jezicima olakšalo je korištenje novog jezika. Projekt je prepoznat kao unaprediv, uz preporuku da se u budućnosti posveti više pažnje vremenskoj kompleksnosti svake metode. Nadogradnja projekta može uključivati i poboljšanje zvučnih efekata i proširenje funkcionalnosti igre.

LITERATURA

[1] Unity Asset Store, Unity Technologies, posjećeno 11. rujna 2024, https://assetstore.unity.com/

[2] Youtube, posjećeno 11. rujna 2024, https://www.youtube.com/

[3] Unity dokumentacija, Unity Technologies, posjećeno 11. rujna 2024, https://docs.unity.com