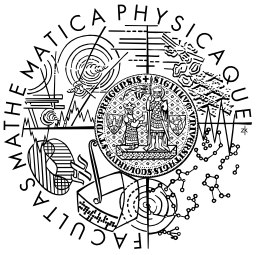


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Mário Mitro

Smoking Rubber - závodní hra v Unity

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek, Ph.D.

Studijní program: Informatika

Studijní obor: Programování a softwarové systémy

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád by som sa poďakoval svojej rodine za podporu nie len pri štúdiu, ale aj pri písaní tejto práce. Menovite by som sa chcel poďakovať môjmu bratovi Erikovi, ktorý pri mne stál počas celého môjho štúdia a bol mi veľkou oporou. Veľká vďaka patrí aj môjmu vedúcemu Mgr. Pavlovi Ježkovi Ph.D. za to, že si na mňa našiel čas a spoločne sme túto prácu dokázali dotiahnuť dokonca. Osobne sa chcem poďakovať Jakubovi Stachovi, ktorého nápady a skúsenosti mi boli veľkou pomocou. Vďaka patrí aj všetkým mojim priateľom, ktorí ma neustále podporovali a pomohli mi v ťažkých časoch. Na záver sa chcem poďakovať Adamovi Gajdošovi a Martinovi Ďurčekovi, ktorí neváhali moju hru testovať a písať spätnú väzbu, ktorá pomohla formovať súčastnú podoby hry.

Název práce: Smoking Rubber - závodní hra v Unity

Autor: Mário Mitro

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek, Ph.D., Katedra distribuovaných a spolehlivých systémů

Abstrakt: Závodné hry patria k jednému z najpopulárnejším videoherným žánrom súčasnosti. Cieľom tejto práce bolo navrhnúť a implementovať 3D závodnú hru, ktorá užívateľom umožňuje výber, odomykanie a vylepšovanie vozidiel, s ktorými môžu pretekať na rôznych tratiach. Tieto trate si hráč môže odomykať. Hra hráčovi poskytuje herné módy ako sú časový mód, eliminačný mód a klasický pretekársky mód, pričom každý mód stanovuje iné podmienky pre úspešné ukončenie preteku. Hráč si v hre vytvára, maže a prihlasuje sa do hráčskych účtov, ktoré obsahujú informácie o aktuálnom bodovom stave hráča a rovnako uchovávajú všetky rekordy, odomknuté trate a autá, ktoré si užívateľ počas hrania odomkol. Súčasťou hry je možnosť Replay, ktorá hráčovi umožňuje pozrieť si záznam jeho jazdy. Samotný projekt obsahuje editor, ktorý vývojárom poskytuje jednoduchú manipuláciu, opravu a pridávanie tratí a áut do hry. Počas implementácie hry sme použili herný engine Unity, ktorý poskytuje veľkú podporu pri vytváraní hier. Táto práca riešila návrh, dizajn a implementáciu tratí, vozidiel, hráčskych účtov a užívateľského rozhrania, ktoré hra obsahuje. V analýze sme sa snažíli nájsť najlepšie riešenia na implementačné problémy zahrňujúce vytváranie modelov, spôsob vytvárania záznamu či spôsob implementácie a chovanie umelej inteligencie. Záverom práce sme popísali, ako ovládať editory, ktoré sme do projektu v Unity implementovali.

Klíčová slova: Unity, závodná hra, vozidlo

Title: Smoking Rubber - Unity Racing Game

Author: Mário Mitro

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Ježek, Ph.D., Department of Distributed and Dependable Systems

Abstract: Racing games are one of the most popular video game genres today. The main goal of this work was to design and implement a 3D racing game that allow users to select, unlock and improve vehicles in which players race on various tracks. These tracks can be unlocked by player. The game provides players with game modes such as time mode, elimination mode and classic race mode each setting different conditions for the successful completion of the race. Players in the game can create, delete and log in to players accounts, which contain informations about the current point status of the player as well as keep all records, unlocked tracks and cars that the user unlocks during the game. Part of the game is the Replay option, which allows player to view a record of his ride. The project itself includes an editor that allow developers easy manipulation, repair, and addition of new tracks and cars. During implementation of the game, we used the Unity game engine, which provides us with great support in creating games. This work addresses the design, engineering and implementation of tracks, vehicles, player accounts and the user interface that the game will contain. In the analysis we tried to find the best solutions to implementation problems, including the creation of models, method of creating a record or the method of implementation and the behaviour of artificial intelligence. Finally, we described how to control the editors that we implemented in the project in Unity.

Keywords: Unity, racing game, vehicle

Obsah

1. Úvod

Hry sa za posledné roky stali súčasťou mainstreamovej kultúry, pričom herné spoločnosti neustále posúvajú hranice svojich možností tak, aby svojim zákazníkom poskytli čo možno najlepší herný zážitok či už z pohľadu grafiky, príbehu alebo dychberucích sekvencíi. K jedným z najpopularnejších a najstarších herných žánrov, ktoré sa na trhu stále vyskytujú, patria bez pochyby závodné hry.

Závodné hry existujú už viac ako štyri desaťročia. Tento žáner sa rozvetvil do rôznych odlišných podžánrov a vždy bol o krok vpred pred ostatnými žánrami videohier, čo sa týka inovácií aj výnosov [1].

Hlavným cieľom tejto práce je navrhnúť a implementovať našu vlastnú závodnú hru, ktorá bude užívateľovi poskytovať herný zážitok, ktorý sa dá porovnať so závodnými hrami, ktoré boli vydané za posledných 10 rokov.

V tejto kapitole sa pozrieme na závodné hry ako také, uvedieme si základné delenie závodných hier do jednotlivých podžánrov a nakoniec uvedieme cieľe tejto práce.

1.1 Závodné hry

Závodné hry sú žáner počítačových hier, ktoré sa hrajú buď z pohľadu tretej alebo prvej osoby (zväčša sú ponúknuté obe možnosti), v ktorých sa hráč účastní nejakého typu preteku alebo pretekárskej sútaže na rôznych typov povrchov. Tieto preteky môžu byť navrhnuté podľa skutočných pretekov alebo môžu obsahovať rôzne sci-fi a fantasy prvky

Závodné hry je možné deliť do mnohých podžánrou, na základe rôznych kritérií. Obecne sa závodne hry delia do troch hlavných kategórií:

- Sim racing
- Arcade style racers
- Kard caring games

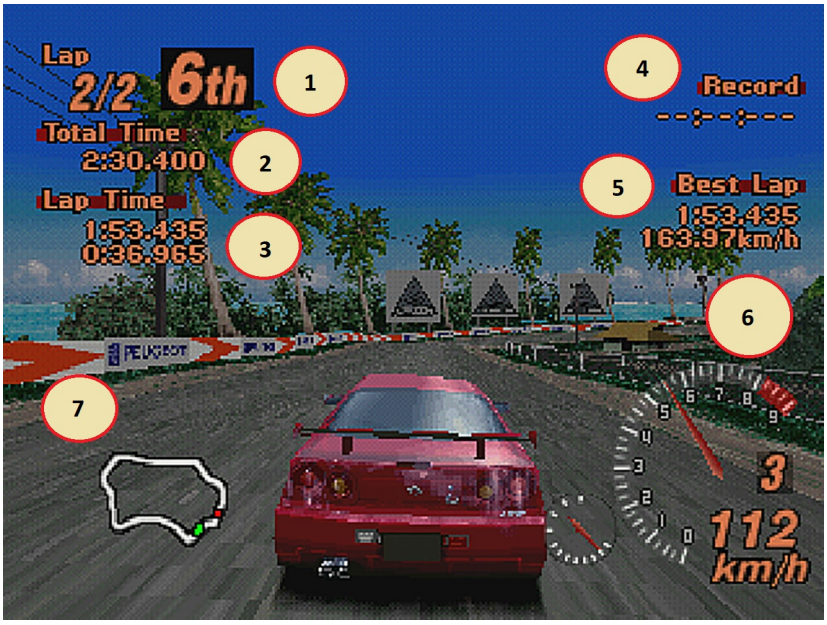
Vo zvyšku podkapitoly si detailnejšie popíšeme jednotlivé podžánre závodných hier, vymenujeme si ich kľučové charakteristiky, povieme ktoré z hlavných charakteristík chceme implementovať v našej hre a nakoniec vymenujeme najpopulárnejšie herné tituly, ktoré tieto žánre reprezentujú.

1.1.1 Sim racing

Sim racing alebo závodné simulácie sú súhrnným pojmom pre žáner závodných hier, ktorý sa pokúša presne simulovať automobilové preteky. Tento žáner je doplnený skutočnými premennými, ako sú spotreba paliva, poškodenie vozidla, opotrebenie pneumatík, priľnavosť a nastavenie pruženia [2].

Aby bol hráč v tomto žánre konkurencieschopný, musí rozumieť všetkým aspektom ovládania vozidla, ktoré robia skutočné preteky také náročné [3], ako napríklad brzdenie šmykom, udržanie kontroly nad vozidlom, ktorého pneumatiky strácajú trakciu, a ako správne vstupovať a opúšťať zákrutu bez straty rýchlosti.

Práve táto úroveň obtiažnosti odlišuje simulátory od **Arcade style racers**, v ktorých sú prvky reálneho sveta vyňaté z preteku a hlavným cieľom je vytvoriť dojem rýchlosti namiesto dojmu reality [4].



Obr. 1.1: Hráčska obrazovka z hry *Grand Turismo II*. Zdroj: gran-turismo.com

Gran Turismo (GT) je séria sim racing videohier vyvinutých spoločnosťou *Polyphony Digital*. Táto séria bola vytvorená pre herné konzoly *PlayStation*. Hry Grand Turismo majú napodobniť vzhľad a výkon veľkého množstva vozidiel, z ktorých väčšina sú licencované reprodukcie automobilov z reálneho sveta. S celkovým predajom 80 milionov kusov na celom svete sa Gran Turismo stalo medzinárodnou senzáciou a má silné zastúpenie najmä v Európe a Severnej Amerike. Prvá hra zo série vtrhla na scénu videohier v roku 1997 a postupne sa vyvíjala spolu s konzolou *PlayStation* pričom naďalej podporuje pôžitok z riadenia u ľudí všetkých vekových skupín po celom svete [5].

Na obrázku (1.1), ktorý zachytáva pohľad na hráčsku obrazovku v hre *Grand Turismo II*, si môžeme všimnúť, že obrazovka ponúka hráčovy veľa informácií o stave preteku a auta. Konkrétne mu poskytuje informácie o jeho umiestnení a kole v ktorom sa nachádza (1), uplynulom čase od začiatku preteku (2), čase, za ktorý hráč prešiel jedno kolo (3), celkovom časovom rekorde trate (4), informáciu o tom, aké je jeho najrýchlejšie zajazdené kolo a rýchlosť, ktorou dané kolo prešiel (5) a samozrejme aktuálna rýchlosť hráča spolu s otáčkami vozidla a rýchlostným stupňom znázornená tachometrom (6) a nejaká minimapa, ktorá ukazuje pozíciu hráča a ostatných vozidiel (7).

Prvky Sim racers, ktoré chceme priniesť do našej prácé sú reálnejšie modely vozidiel (nechceme fantasy alebo rozprávkové vozidlá) a prehľadné užívateľské rozhranie.

1.1.2 Arcade style racers

Arcade style racers kladú zábavu a rýchlosť nadovšetko, predovšetkým kvôli faktu, že vozidlá zvyčajne súťažia jedinečnými spôsobmi. Kľúčovou vlastnos-

ťou arcade style racers, ktorá ich odlišuje od sim racers, je ich podstatne voľnejšia fyzika. Zatiaľ čo v skutočných pretekoch musí vodič pri väčšine zákrut výrazne znížiť svoju rýchlosť, arcade style racers všeobecne povzbudzujú hráča k tomu, aby vozidlu udržal rýchlosť driftovaním cez zákrutu. Na rozdiel od sim racers, zrážky s inými pretekármi, prekážkami na trati alebo dopravnými prostriedkami sú prehnané. Vo väčšine prípadov arcade style racers jednoducho odstránia presnosť požadovanú od sim racers a zamerajú sa na samotný závodný prvok.

Obr. 1.2: Hráčska obrazovka z hry *Need For Speed: Hot Pursuit 2*. Zdroj: konzolyahry.sk

Need for Speed (NFS) je **arcade style racers** séria závodných videohier vydaná spoločnosťou *Electronic Arts* a v súčasnosti vyvíjaná spoločnosťou *Criterion Games* [6]. Séria sa sústreďuje na nedovolené pouličné preteky a vo všeobecnosti dáva hráčom za úlohu absolvovať rôzne druhy pretekov a vyhnúť sa miestnym policajným zložkám v policajných naháňačkách. Séria vydala svoj prvý titul *The Need For Speed* v roku 1994. Posledná hra *Need for Speed: Hot Pursuit Remastered* bola vydaná 6. novembra 2020. Táto séria bola v priebehu rokov vyvíjaná viacerými tímami, vrátane *EA Canada, EA Black Box, Slightly Mad Studios* a *Ghost Games*. *Need For Speed* bola ako séria prijatá kritikmy pozitívne a je jednou z najúspešnejších videoherných serií všetkých čias, v ktorej sa predalo viac ako 150 miliónov kópií hier [7]. Vďaka veľkému predaju sa séria rozšírila do ďalších foriem médií, vrátane filmovej adaptácie a licencovaných hračiek *Hot Wheels* [8].

Na obrázku (1.2) môžeme vydieť pohľad na hráčsku obrazovku v hre *Need For Speed: Hot Pursuit 2* z roku 2002. Hráčska obrazovka poskytuje hráčovi informáciu o umiestnení hráča (1), bodovému svatu v danom móde (2), minimape (3), informáciám o preteku ako je aktuálne kolo, čas a najlepší čas za kolo (4) a na záver ukazuje hráčovy rýchlosť jeho vozidla pomocou tachometra (5).

Z prvkov, ktoré arcade style racing žáner ponúka, si zvolíme do našej hry herné módy, aby sme hru obohatili o viacero herných situácií a voľnejšiu fyziku vozidla

z dôvodu menšej náročnosti implementácie a dynamickejšieho pocitu počas hry.

1.1.3 Kart racing games

Kart racing games sú známe zjednodušenou jazdnou mechanikou, pričom do hry pridávajú prekážky, neobvyklé závodné dráhy a rôzne akčné prvky. Tento žáner je tiež známy obsadzovaním fiktivných postáv, najmä z mediálnych sérií, do role pretekárov vo vozidlách, ktoré majú často veľmi netradičný dizajn často odrážajúci osobitnú črtu alebo osobnosť postavy, ktorá ich riadi. Kart racing games sú arkádovejším zážitkom ako iné závodné hry a zvyčajne ponúkajú špičkovú hrateľnosť, v ktorej môžu postavy po sebe strieľať, zbierať štíty a body na získanie výhody v preteku. Typicky sú v takýchto hrách vozidlá reprezentované ako motokáry a skútre, ktorým chýba všetko v duchu radiacej páky a spojkového pedálu.

Obr. 1.3: Hráčska obrazovka z hry *Crash Team Racing*. Zdroj: threevoicesmedia.com

Crash Team Racing (CTR: Crash Team Racing) je videohra z roku 1999, ktorá bola vyvinutá spoločnosťou *Naughty Dog* a publikovaná spoločnosťou *Sony Computer Entertainment* pre hernú konzolu *PlayStation*. Je to štvrtý diel zo série *Crash Bandicoot*. Príbeh hry sa zameriava na úsilie *Crash Bandicoota, Doctora Neo Cortexa* a iného legendárneho tímu postáv zo série *Crash Bandicoot*, ktorí musia závodiť proti egomaniackému mimozemštanovi *Nitros Oxide*, aby zachránili Zem pred zničením. V hre môžu hráči prevziať kontrolu nad jednou z pätnástich postáv série *Crash Bandicoot*, hoci spočiatku je postáv k dispozícii iba osem. Počas závodov je možné získať výhodu vďaka zbieraniu vylepšení, ktoré môže hráč použiť k ofenzíve alebo defenzíve.

Na obrázku (1.3) sa nachádza hráčska obrazovka z už spomínanej hry *Crash Team Racing*. Môžeme si všimnúť, že počas hrania sa hráčovi neustále zobrazujú informácie o uplynutom čase od začiatku kola (1), informácia a totožnosti prvých štyroch pretekárov (2), pozícia hráča (3), zbraň ktorú má hráč aktuálne k dispozícií (4), počet bodov (5), informácia o aktuálnom kole (6) a samozrejme

minimapa. Kedže sa jedná o *Kartovú závodnú hru*, z obrázku vidíme, že v nej nechýbajú zbrane či známe postavy z hier. V tomto prípade sa jedná o postavy z hernej série *Crash Bandicoot*.

Záverom si rovnako ako v minulých žánroch zvolíme prvky Kard Racing game, ktoré budeme implementovať v našej hre. Konkrétne si z tohto žánru privedieme koncept bodov, ktoré hráč získava v pretekoch, a prvok upgradu vozidla, či už sa jedná o vyššiu rýchlosť vozidla alebo iné vlastnosti, ktoré si hráč bude môcť vylepšiť (na vylepšovnie vozidla budú slúžiť body, ktoré hráč získa).

1.2 Zhrnutie cieľov

V tejto kapitole sme si definovali a popísali čo sú to závodne hry a do akých hlavných žánrov ich delíme. Každý žáner sme si popísali a u každého sme uviedli prvky, ktoré by sme chceli implementovať v tejto práci.

Na záver si zhrnieme ciele tejto práce. Ako sme spomenuli na začiatku, cieľom tejto práce je navrhnúť a implementovať vlastnú závodnú hru, ktorá bude užívateľovi poskytovať herný zážitok, ktorý sa dá porovnať so závodnymi hrami, ktoré boli vydané za posledných 10 rokov. K tomuto cieľu pridávame v dôsledku tejto kapitoly nasledujúce kriteriá (detailne sa o nich budeme baviť v kapitole 2):

1. Hrateľné autá
2. AI pre autá
3. Intuitívne UI
4. Replay
5. Body
6. Herné módy
7. Hráčske účty
8. Editor na jednoduché pridávanie tratí a áut

2. Náhrv

Cieľom tejto kapitoly je presne špecifikovať ciele, ktoré sme si stanovili v sekcii 2.11. Tieto ciele rozvinieme o podciele, ktoré budeme pri implementácii požadovať. Na záver kapitoly si stručne zhrnieme spresnené ciele, ktoré má táto práca naplniť.

2.1 Auto

Návrh našej hry začína pravdepodobne tým najdôležitejším aspektom závodných hier a to samotným vozidlom. V tejto sekcií sa pokúsime čo možno najdetailnejšie špecifikovať, ako má vozidlo v našom projekte vyzerať, aké prvky musí obsahovať, čo všetko bude schopné vykonávať na pokyn hráča alebo počítača a aké efekty má byť vozidlo schopné vykonávať.

2.1.1 Modely áut

Skôr ako začneme popisovať, čo všetko máme od vozidla v našej hre očakávať, najpr si musíme stanoviť, ako majú vozidla v hre vyzerať. Výzor vozidiel má dopad na výzor celej hry. Ak zvolíme viac rozprávkové vozdilo tak od tratí a celkojej hry by sme očakávali to isté. My sme si záverom sekcie 1.1.1 stanovili, že do našej hry chceme implementovať realnejšie modely vozidiel. Čo konkrétne pod tým myslíme?

Pravdepodobne si vieme všetci predstaviť, že modely vozidiel vo videohrách môžeme deliť podľa mnohých kategórií. V tomto prípade nás zaujímajú dve kategórie:

- kvalita modelu
- reálnosť modelu

Pod pojmom kvalita modelu chápeme množstvo detailov, ktoré daný model obdahuje. V závislosti od týchto detailov môžeme na jednej strane spektra vidieť modely vozidiel, ktoré sú tvorené len kockou a štyrmi kruhmy reprezentujúcimi kolesá, pričom na druhej strane spektra sa nachádzajú modely, ktorých kvalita sa dá len ťažko rozlíšiť od skutočných vozidiel.

Vozidla v závodných hrách sa líšia nie len kvalitou ale taktiež reálnosťou. Na jednej strane máme v hrách ako je *Grand Turismo* k dispozícií značky a druhy vozidiel, ktoré je možné riadiť aj v reálnom živote, pričom na strane druhej máme hry ako *Mario Kards*, kde vozidla používané hráčmi sú navrhnuté netradične a často fantasticky. Kritérium reálnosti modelu popisuje, ako veľmi je vozidlo založené na skutočnom vozidle.

Obr. 2.1: Ukážka modelu vozidla

Teraz, keď sme si vysvetlili základné kritéria, do ktorých rozdeľujeme modely vozidiel, špecifikujeme si vzhľad vozidiel, ktoré chceme v našej hre používať. Určite sa zameriame na reálnejšie modely vozidiel (napr. existujúce značky a modely), pretože práve takéto modely majú určité prvky ako sú napr. svetlá, ktoré budeme v hre potrebovať.

Čo sa týka kvality modelov, budeme trocha liberálnejší a nebudeme nutne požadovať *high-quality* modely, hoci aj takéto modely sú samozrejme prípustné. Namiesto toho si vystačíme s menej kvalitnými modelmy vozidiel.

Na obrázku 2.1 máme zobrazené vozidlo, ktoré kvalitou odpovedá naším potrebám. Na druhú stranu reálnosťou tento model pokrýva minimum prípustnosti pre naše účely (ma svetlá). Najdôležitejšie pri výbere modelov do našej hry je konštantnosť kvality a reálnosti modelov. Aby hra nepôsobila nekoherente, je nutné, aby sa modely vozidiel od seba v týchto aspektoch veľmi nelíšili.

2.1.2 Fyzika

Keď sme si stanovili vzhľad vozidla, poďme si špecifikovať fyziku našich vozidiel, ktoré bude naša hra implementovať. V sekcií 1.1.2 sme si stanovili, že v našej hre si vystačíme s voľnejšou fyzikou vozidiel. Voľnejšou fyzikou rozumieme zjednodušenie reálneho správania vozidla v prospech rýchlosti a celkovej svižnosti vozidla.

Samozrejme, nechceme, aby všetky fyzikálne aspekty vozidla boli rovnako voľné. Preto si poďme popísať, ako by sa malo vozidlo v našej hre správať a čo všetko by malo zvládať z hľadiska jazdy a interakcie s prostredím.

Interakcia s prostredím, gravitácia

Nie je žiadným prekvapením, že vo väčšine závodných hier vozidlá interagujú minimálne s prostredím a inými vozidlami. Tieto interakcie môžu byť interpretované reálne, alebo môžu byť prehnané za účelom "wowëfektu, či iného zámeru autora. Náš zámer je interpretovať interakcie vozidla čo možno najreálnejšie, avšak v niektorých prípadoch si zvolíme trochu voľnejšiu reakciu.

Aby sme mohli interagovať s prostredím v nejakej zmysluplnej miere, potrebujeme gravitáciu alebo aspoň simuláciu gravitácie a základnej fyziky pevných objektov, ktorá zaistí, že sa správanie objektov nebude veľmi odlišovať od reality.

Keď teraz máme gravitáciu, potrebujeme, aby mali vozidlá hmotnosť, ktorá by odpovedala vozidlám v skutočnom svete. Fakt, že vozidlo je pevné teleso s nejakou hmotnosťou v kombinácií rozumnej simulácie fyziky a gravitácie by malo zaistiť korektnú reakciu s prostredím trate ako je napr. cesta, prekážky, či iné vozidlá. Je nutné poznamenať, že napriek korektnej reakcii s prostredím nie je nášou požiadavkou, aby vozidlá a prostredie odrážali dopad danej interakcie, teda nepotrebujeme, aby sa vozidlá pri kolizií poškodili, alebo aby sa prekážky pri kolízií s vozidlom deformovali.

Aby sme to zhrnuli, požadujeme, aby mali vozidlá hmotnosť a reagovali s prostredím približne rovnako ako by reagovali v skutočnom svete, pričom nebudeme vyžadovať deformáciu vozidiel ani iných objektov na trati.

Pohyb

Kľúčovým aspektom väčšiny závodov je schopnosť vozidiel pohybovať sa a dosahovať veľkej rýchlosti. V skutočnom svete je táto rýchlosť a samotné zrýchlenie vozidla závislé na mnohých faktoroch. Kedže nám nejde o super reálny fyzikálny model čo sa týka fyziky vozidiel, no napriek tomu nechceme, aby fyzika pôsobila príliš nereálne, potrebujeme podriadiť pohyb, zrýchlenie a rýchlosť vozidla istým kritériám, ktoré ich budú ovplyvňovať. Tieto kritéria sú:

- hmotnosť vozidla
- výkon vozidla
- trenie kolies

Kombinácia týchto faktorov by mala zaistiť relatívne reálne zrýchlenie resp. spomalenie vozidla počas preteku.

Zatáčanie kolies

Vo väčšine prípadov by sa požiadavka, aby bolo vozidlo schopné otáčať kolesá v závodnej hre, mohla zdať trochu zbytočná, ale po chvíľke zamyslenia si uvedomíme, že to nie je úplne triviálna požiadavka.

Uvážujme dve vozidlá, ktoré by boli schopné otáčať kolesami maximálne v 35 stupňovom uhle. Jedno vozidlo dosiahne rýchlosť 150 km/h a druhé vozidlo dosiahne rýchlosť 20 km/h. Ak hráč v takomto prípade dá pokyn k otočeniu kolies (za predpokladu, že pokyn vydal napr. klávesnicou) o koľko stupňov sa majú kolesá na jednotlivých vozidlách otočiť? Ak by prvé vozidlo otočilo kolesá o maximálny uhol 35 stupňov, tak pri jeho rýchlosti by takáto akcia musela neodvratne skončiť kolíziou. Naopak, druhé vozidlo by tento uhol pokojne zvládlo.

Týmto príkladom sa snažíme povedať, že požiadavok na vozidlo nie je len aby zatáčalo, ale aby zatáčalo korektne. Teda pri implementácií musíme zaistiť hráčovi asistenciu pri zatáčaní, aby sa možný uhol otočenia kolies menil s rýchlosťou vozidla. Týmto požiadavkom zaistíme reálne zatačánie vozidla bez ohľadu na jeho rýchlosť.

Brzdy a šmyky

Keď sme teraz schopný vozidlo rozbehnúť, zatáčať a potencionálne nabúrať, musíme zaistiť, aby sme vozidlo vedeli v prípade potreby ubrzdiť. K tomuto účelu slúži v reálnom svete samozrejme brzda, konkrétne dve druhy bŕzd:

- manuálna brzda
- ručná brzda

V reálnom svete majú obe druhy bŕzd za úlohu zastaviť vozidlo, a v našom prípade to nebude inak. Manuálnu brzdu chceme implementovať tak, aby hráča ubrzdila v prípade potreby. Na druhú stranu chceme využiť ručnú brzdu nie len k spomaleniu vozidla, ale aj k tvorbe šmyku. Vďaka tejto mechanike by si hráč mohol zvoliť, či vozidlo počas prudšej zákruty ubrzdí, alebo vďaka ručnej brzde prejde do driftu a tak nepríde o podstatnú časť rýchlosti, ktorá by mohla byť kľúčová k výhre v preteku.

Keďže sme v tejto sekcii spomenuli šmyky a driftovanie, môžeme tento požiadavok rozšíriť. Vozidlo sa môže dostať do šmyku nie len vplyvom ručnej brzdy, ale aj v prípade veľkej rýchlosti a dlhého bočenia kolies. Musíme však podotknúť, že miera šmyku by v takomto prípade nemala byť taká výrazná ako v prípade šmyku z ručnej brzdy.

2.1.3 Efekty

Ako sme spomenuli v sekcií 1.1.1, budeme v našej hre požadovať reálnejšie modely vozidiel. V sekcií 2.1.1 sme popísali vzhľad našich vozidiel, no reálny vzhľad vozidla zosilňujú efekty, ktoré sa samostatne nemusia zdať dôležité (a možno to tak aj je), ale kombináciou týchto efektov dosiahneme nie len lepšie vyzerajúce vozidlá, ale aj celkový lepší herný zážitok u hráčov. V tejto sekcií sa pozrieme na niektoré efekty, ktoré by sme chceli v našej hre implementovať.

Svetlá

Začneme efektom, ktorý si vie každý, kto už videl nejaké vozidlo, predstaviť a tým efektom sú svetlá. Konkrétne sa budeme zaujímať o tri druhy svetiel:

- predné svetlá
- brzdové svetlá
- cúvacie svetlá

Obr. 2.2: Ukážka svetelných efektov vozidla. 1 - Brzdové svetlá 2 - Predné svetlá 3 - Cúvacie svetlá

Nie je nutné vysvetľovať, čo presne tieto druhy svetiel znamenajú, pretože to vyplýva z ich názvov. Obrázok 2.2 ilustruje, ako by tieto efekty mohli vyzerať na nejakom vozidle. Poďme si teraz stručne vysvetliť, ako by tieto svetlá mali v hre fungovať a ako by to mohlo hráčovi pri hre pomôcť.

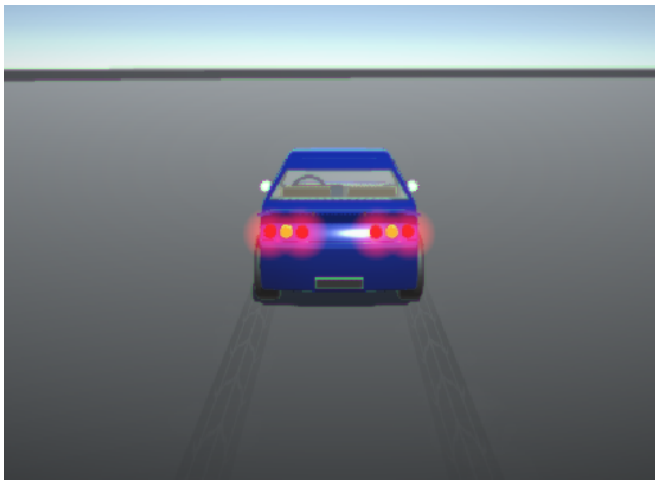
Predné svetlá je efekt, ktorého aktiváciu chceme ponechať na rozhodnutí hráča v tom zmysle, že chceme, aby bol hráč schopný kedykoľvek tieto svetlá zapnúť alebo vypnuť. Ak budeme jazdiť po trati, ktorá je dostatočne osvetlená, zrejme nemá zmysel mať rozsvietené predné svetlá na vozidle (i keď to hráčov nijako neobmedzuje). Naopak, jazda na trati, ktorá je modelovaná v nočnom prostredí bez svetiel môže, a pravdepodobne bude viesť k opakovaným kolíziám.

To isté ale neplatí u ostatných druhov svetiel. Tieto svetlá slúžia ako vizuálny dopad na jazdu vozidla (ovladaných hráčom alebo AI), ktoré brzdí alebo cúva. Tieto typy svetiel majú pomáhať hráčovi odhadnúť správanie okolitých vozidiel a tak mu uľahčiť jazdu.

Efekty pri brzdení

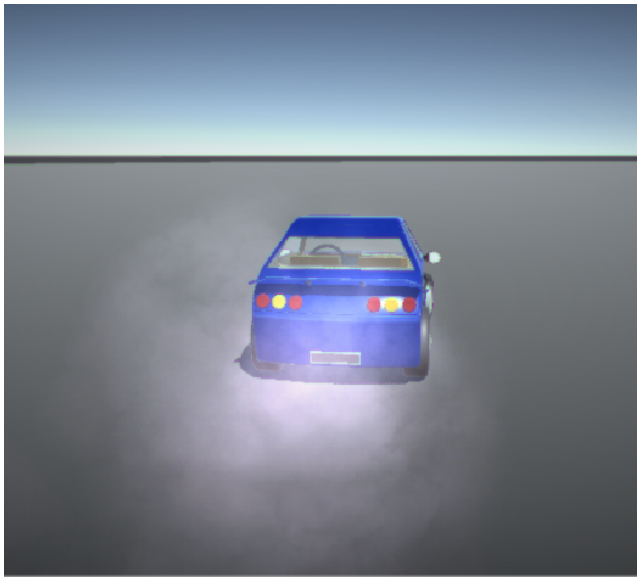
Keď sme teraz schopný rozsvietiť svetlá pri bŕzdení, poďme si uviesť dva ďalšie efekty, ktoré s bŕzdením vozidla súvisia. Jedná sa o:

- stopy po brzdení
- dym počas šmyku



Obr. 2.3: Ukážka bŕzdných stôp vozidla

Pod stopamy po brzdení si predstavujeme situáciu, akú môžeme pozorovať na obrázku 2.3. Na tomto obrázku je znázornené vozidlo, ktoré brzdí (vieme to vďaka brzdovým svetlám) a počas brzdenia zanecháva na ceste brzdné stopy. V reálnom svete sú tieto stopy časti pneumatiky, ktoré sa trením po asfatle uvoľnili a otali na ceste. V našom prípade pôjde o efekt, ktorý bude mať za úlohu zanechať obraz stopy po brzdení na ceste bez opotrebovania pneumatík.



Obr. 2.4: Ukážka dymu pri šmyku kolies vozidla

V prípade, že sa vozidlo dostane do šmyku, alebo začne brzdiť vo veľmi vysokých rýchlostiach budeme od hry požadovať okrem stôp po brzdení ďalší efekt, ktorým je dym z kolies. Tento efekt pomôže zdôrazniť hráčovi prudkosť týchto akcií a kombinácia týchto dvoch efektov bude mať za následok vierohodnejšie vykreslenie skutočného preteku.

Animácia Kolies

Tento konkrétny efekt je pravdepodobne niečo, nad čim väčšina čitateľov ani neuvažuje ako nad efektom a možno to niektorí berú ako samozrejmosť. No musíme si uvedomiť, že v kontexte hier, herného návrhu a implementácie sú kolesá vozidla brané ako obekty, ktoré sú súčasť vozidla. Animácia týchto objektov je efekt, ktorý je v súčasnej dobe takpovediac nevyhnutný pre pocit normálnosti vozidla. Užívateľ si možno nevšimne stopy po šmyku či dym z kolies, ale určite si všimne že sa na jeho vozidle a ostatných vozidlách vôbec neotáčajú kolesá.

Z tohto dôvodu je požiadavok na animáciu kolies jedným z kľúčových požiadavkov pre navodenie reálneho pocitu z vozidla. Keď sa bavíme o animácií kolies, myslíme tým vlastne dve nezávislé pohyby kolies. Konkrétne sa jedná o:

- pohyb kolies pri zatáčaní
- pohyb kolies pri pohybe vozidla

Zvuk

Posledným efektom, ktorý spomenieme, sú zvukové efekty. Tieto typy efektov patria v súčastnosti k jednému zo základných kameňov vieohier. Mnohé herné série používajú hudbu z populárnych médií, či dokonca majú hudbu a zvukové efekty tvorené na mieru rôznymi skladateľmi.

V našej hre nebudeme klásť veľký dôraz na zvuk. Preto budeme zvukové efekty stavať na kritériách, ktoré sme od nášich vozidiel vyžadovali. Konkrétne, zvukové efekty vozidiel budú tvoriť tieto dve zložky:

- zvuk šmyku
- zvuky motora

Pre zvuk šmyku požadujeme, aby vozidlá v prípade šmyku vydali zvuk odpovedajúci tejto akcií.

Zvuk motoru je o niečo komplikovanejší. Komplikuje to fakt, že počas jazdy nemá vozidlo rovnaký zvuk. Preto požadujeme, aby vozidlo interpretovalo zvuky motora čo možno najkorektnejšie. Konkrétne stavy, ktoré musíme interpretovať sú:

- voľnobeh
- cúvanie
- zrýchľovanie
- spomaľovanie
- preradenie

Reálne vozidlo pravdepodobne obsahuje viac stavov, ktoré by bolo treba interpretovať, no pre naše účely by interpretácia týchto stavov mala poskytnúť dostatočnú úroveň hodnovernosti.

2.1.4 Kamery

V tejto podkapitole sa pokúsime navrhnúť vhodnú kameru pre našu hru. Skôr, ako sa dotkneme návrhu kamery, musíme si stanoviť počet kamier reps. kamerových módov v našej hre. Veľa závodných hier ponúka rôzny počet kamerových módov, ktoré si hráč môže počas hry meniť a tak meniť celkový pohľad na hru. Iné ponúkajú iba jeden mód, s ktorým sa hráč musí proste zmieriť.

Pre našu hru zvolíme dve kamerové módy. Tieto módy si pomenujeme ako:

- hlavná kamera
- quick kamera

Obr. 2.5: Ukážka hlavnej kamery vozidla. 1 - Vozidlo stojí, 2 - vozidlo v pohybe, 3 - vozidlo odbočuje

Pod hlavnou kamerou budeme rozumieť pohľad na vozidlo, ktoré bude mať hráč od začiatku preteku. Táto kamera ma za úlohu vozidlo sledovať tak, aby ho bol hráč schopný priviesť do cieľa. Sledovanie vozidla kamerou sa dá navrhnúť mnohými spôsobmy. Spôsob, ktorý sme navrhli, by sa dal nazvať dynamický.

Na obrázku 2.5 môžeme pozorovať, ako by sa takáto dynamická kamera mala chovať. V prípade, že vozidlo stojí (1), mala by vozidlo sledovať z relatívne blízkej vzdialenosti. V momente, keď sa vozidlo rozbehne (2), mala by sa kamera postupne od vozidla vzďaľovať do chvíle, kedy dosiahne maximálnu prípustnú vzdialenosť. Naopak, ak vozidlo spomaľuje alebo brzdí, tak by sa mala vzdialenosť kamery a vozidla zmenšovať. Nakoniec v prípade bočenia (3) by kamera nemala presne nasledovať vozidlo, ale poskytnúť trochu miesta, aby hráč mohol vidieť prednú stranu vozidla.

Tento návrh kamery v kombinácií s efektmy, ktoré od vozidla vyžadujeme, spôsobí pocit rýchlosti. Každé zrýchlenie, spomalenie a odbočenie bude mať oveľa väčší dopad než v prípade, kedy by bola kamera v konštantnej vzdialenosti od vozidla bez ohľadu na jeho rýchlosť.

Obr. 2.6: Ukážka quick kamery vozidla

Druhý typ kamery, ktorý sme nazvali quick kamera, bude ukazovať hráčovi čo sa nachádza za vozidlom. Na obrázku 2.6 môžeme vidieť návrh, ako by takáto kamera mohla vyzerať.

Takáto quick kamera slúži ako spätné zrkadlo v skutočnom vozidle. Umožní hráčovi vidieť, či má za sebou nejaké vozidlá resp. či sú tie vozidla blízko a podľa výsledku sa môže tieto vozidlá rozhodnúť blokovať, či dokonca odstrčiť.

2.1.5 Respawn

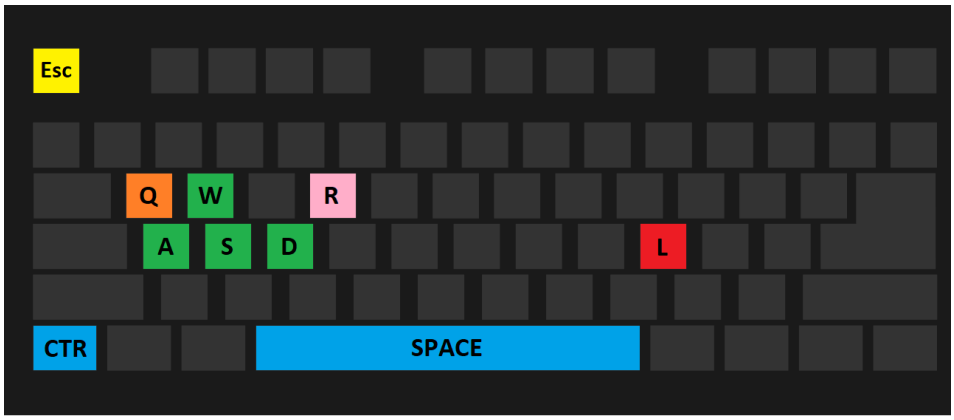
Slovo "respawn"je odvodené od slovíčka spawn, ktoré v kontexte videohier znamená vytvorenie postavy alebo NPC (non playable chracter) do sveta v ktorom sa hra odohráva. Respawn je znovuvytvorenie danej postavy alebo NPC.

Táto akcia sa v závodných hrách používá v prípadoch, že vozidlo opustí dráhu na určitý čas, pohybuje sa v protismere, alebo v dôsledku kolízie skončí otočené na kapote. V takýchto prípadoch respawn zaistí navrátenie vozidla naspäť na trať a otočí ho v smere jazdy.

Kedže nemôžeme zaručiť, že sa tieto prípady v našej hre nevyskytnú, stanovujeme si túto požiadavku na naše vozidlá (bez ohľadu na to, či sú ovládané hráčom alebo AI).

2.1.6 Vstup Hráča

Posledná požiadavka, ktorú máme na naše vozdilá, je schopnosť reagovať na vstup hráča prostredníctvom klávesnice.



Obr. 2.7: Zobrazenie povoleného vstupu klávesnice. Zdroj: pixy.org Zelená - pohyb vozidla, Modrá - brzdy, Červená - predné svetlá, Oranžová - quick kamera, Ružová - respawn, Žltá - pauza

Hráč musí byť počas hry schopný pomocou klávesnice vozidlo:

- rozhýbať
- ubrzdiť
- zapínať/vypínať predné svetlá
- zapínať/vypínať quick kameru
- respawnovať sa

Na obrázku 2.7 možeme vidieť návrh reprezentácie týchto požiadaviek na klávesnici. Pohybovú sekciu klávesnice pravdepobodne nie je potreba vysvetľovať, pretože sa jedná o štandartné ovládacie klávesy. Čo sa týka brzdenia, máme k dispozícii *medzerník*, ktorý by v našom návrhu reprezentoval pedálovú brzdu a ľavý control, ktorý by mal reprezentovať ručnú brzdu.

V sekcií 2.1.4 sme spomenuli, že základnú kameru chceme mať zapnutú od začiatku preteku. To isté samozrejme neplatí o quick kamere. Túto chceme používať na rýchle skontrolovanie toho, čo sa deje za nami. Teda najrozumnejšia varianta, čo sa týka spúštania tejto kamery, je spúštať ju pomocou stlačenia tlačítka na klávesnici. V tomto návrhu sa jedná o klávesu Q. Konkrétne sa quick kamera spustí stlačením klávesy Q a vypne pustením klávesy Q.

Na záver tu máme spustenie/vypnutie predných svetiel a respawn. K otázke zapínania a vypínania predných svetiel sme sa vyjadrili v sekcií 2.1.3. Konkrétne, spášťanie predných svetiel by bol hráč schopný vykonávať klávesou L. Čo sa týka respawnu, hráč môže svoje vozidlo znovu uviesť na cestu v prípade potreby stlačením klávesy R. V návrhu sme sa rozhodli pre manuálny respawn, pretože neexistuje žiadna výhoda v preteku, ktorú hráč získava použitím respawnu okrem faktu, že sa ocitne na ceste.

2.2 AI

Pravou podstatou závodných hier, ako už z názvu vyplýva, je samotný pretek medzi niekoľkými vozidlami. Kedže nie je možné (alebo preferované), aby hráč ovládal viac než jedno vozidlo, musí ostatné vozidla ovládať PC. Systém, ktorý tieto vozidlá ovláda označujeme ako AI (artificial intelligence).

V tejto podkapitole si navrhneme AI pre naše vozidlá. Pozrieme sa, aké aspekty by mala AI pre naše vozidlá spĺňať a ako by tieto aspekty mali byť preprezentované v samotnej hre. Konkrétne sa pozrieme na dva hlavné aspekty, ktoré by mala naša AI spĺňat. Tieto aspekty sú:

- schopnosť prekonať trať
- vyhýbanie sa prekážkam

Schopnosť prekonať trať

Tento požiadavok vyžaduje, aby boli vozidlá, ktoré ovláda AI, schopné prejsť celú trať a v rámci tohoto procesu sa príliš nelíšiť od vozidla, ktoré ovláda hráč.

To znamená, že od AI požadujeme, aby bolo schopné vozidlo rozpohybovať, riadiť a v prípade nutnosti taktiež ubrzdiť. Ďalej, v prípade potreby, by malo byť AI schopné vozidlo respawnovať.

V prípade prudkého brzdenia alebo šmyku by malo byť vozidlo schopné používať všetky efekty, ktoré majú vozidlá k dispozícií či sa už jedná o efekty zvukové, svetelné či efekty, ktoré má vozidlo vykonať počas šmyku, o ktorých sme sa bavili v sekcií 2.1.3.

Vyhýbanie sa prekážkam

Tento požiadavok je pomerne samovysvetľujúci. Počas preteku sa na trati môžu vyskytovať statické alebo dynamické prekážky. Potrebujeme teda, aby bolo AI schopné sa týmto prekážkam vyhnúť.

Obr. 2.8: Ukážka vyhýbaniu sa dynamickej prekážke. 1 - Detekcia prekážky, 2 - Počiatok vyhýbania sa, 3 - Zakončenie

Statické prekážky na trati môže tvoriť čokoľvek od zvodidiel po stĺpy (záleží na trati). Naopak, v prípade dynamických prekážok sa bude jednať výlučné o ostatné vozidlá. Na obrázku 2.8 môžeme pozorovať, ako by vyhýbanie sa takému vozidlu mohlo vyzerať.

Splnenie všetkých týchto požiadaviek nedosiahne požadovaného efektu pokiaľ tieto úkony nebude AI vykonávať plynule tak, aby v očiach hráča pôsobili tieto vozidlá ako keby boli ovládané inými hráčmi.

Dynamické AI

Posledný požiadavok, ktorý máme na AI v našej hre je, aby regulovalo a prispôsobovalo rýchlosť vozidiel podľa umiestnenia hráča. Konkrétne tým myslíme, že ak je hráč napr. tretí a vozidiel v preteku je osem, aby vozidlá pred hráčom trochu spomalili a vozidlá za hráčom trochu pridali. Toto uberanie/pridávanie rýchlosti závisi na pozícií pred/za vozidlom. Čím horšie umiestnenie za hráčom, tým väčší rýchlostný boost vozidlo obdrží. Týmto spôsobom udržíme hráča neustále v pozore a zároveň mu dáme možnosť zlepšiť svoju pozíciu a stále bojovať o víťazstvo v preteku (samozrejme len do určitej miery).

2.3 Trať

Keď sme dokončili našu predstavu vozidla, poďme sa pozrieť na druhý kameň, na ktorom stoja závodné hry, trate. Pod traťou rozumieme scénu, ktorá obsahuje nejaký pevný povrch ako je asfalt, štrk, ľad, atď na ktorom sa dá jazdiť s vozidlom.

V tejto podkapitole si popíšeme našu predstavu trate. Ako by mala vyzerať, čo všetko by malá obsahovať a aké informácie by pri hre mal byť schopný hráč vyčítať.

2.3.1 Vzhľad

Trate v súčastnosti obsahujú viac než len cestu. Tvoria jú scenérie a prepracované modely objektov ako sú napr. stromy, stĺpy, tráva, voda a mnohé iné. Čo sa týka vzhľadu našej trate, zameriame sa na základné prvky (všetko naviac je samozrejme vítané). V našej hre budeme požadovať základnú trať, čo v tomto prípade znamená, že si vystačíme s cestou a základnou scenériou. Modely scenérie nemusia byť príliš delailné alebo prepracované. Samozrejme, že vysokokvalitné modely sú taktiež vhodné, ale v tejto chvíli sa snažíme popísať akceptovateľné minimum.

Jediné požiadavky, ktoré budeme mať na trate v našej hre, budú:

- reálny vzhľad trate
- dostatočne široké cesty

Reálny vzhľad trate je požiadavka, ktorú si kladieme, aby sme neporušili kontinuitu so vzhľadom našich vozidiel. Trate, ktoré chceme používať by mali obsahovať prvky, nad ktorých existenciou nebude hráč uvažovať. Samozrejme, že sa nemusí jednať o hyperrealistické trate. Na druhú stranu fantasy a sci-fi prvky by sme v tratiach pre našu hru nechceli vidieť.

2.3.2 Tachometer

Náš návrh vozidla, efektov a dynamickej kamery slúži okrem iného k tomu, aby navodil pocit rýchlosti. Pocit, že sa hráč pohybuje rýchlo, je iste cenný pre každého dizajnéra závodných hier, no chce to posledný detail k tomu, aby to bolo dokonalé – tachometer.

Tachometer ako taký sa v závodných hrách používv, okrem informácie pre hráča, ako rýchlo sa pohybuje, aj k navodeniu akejsi ilúzie jazdy v skutočnom vozidle. Každý, kto kedy sedel v nejakom vozidle si iste všimol, že rovno pred zrakom vodiča je na palubnej doske zobrazený tachometer.

Obr. 2.9: Ukážka tachometru. Zdroj: konzolyahry.sk 1 - Rýchlosť vozidla, 2 - Aktuálny rýchlostny stupeň vozidla, 3 - Otáčky vozidla

Čo sa týka vzhľadu tachometra, závodné hry nám ponúkajú veľa možnosti. Z hľadiska výzoru tu máme dva veľké dizajny. Konkrétne sa jedná o:

- digitálny tachometer
- analógový tachometer

Ďalšou otázkou, nad ktorou sa musíme zamyslieť, je čo všetko má náš tachometer zobrazovať. Má to byť len rýchlosť vozidla alebo sa má jednať o viac vecí? Pri návrhu tachometra do našej hry sme sa inšpirovali tachometrom, ktorý je zobrazený na obrázku 2.9. Tento konkrétny tachometer je z hry *Need For Speed: Hot Pursuit 2* a ako môžeme vidieť, tak z hľadiska dizajnu sa jedná o hybrid, ktorý kombinuje analóg a digitál. Z hľadiska informácií dáva tento tachometer hráčovi tri informácie:

- otáčky vozidla (1)
- aktuálny rýchlostny stupeň vozidla (2)
- rýchlosť vozidla (3)

Podľa nášho názoru je toto perfektný dizajn tachometra, ktorý by sme radi použili v našej hre.

2.3.3 Minimapa

Niekedy je počas preteku ťažké odhadnúť prudkosť zatáčky alebo vzdialenosť ostatných vozidiel, ktoré su pred alebo za hráčom. Ako pomoc pre hráča uviedli dizajnéry do takýchto typov hier minimapy.

Obr. 2.10: Ukážka minimapy. Zdroj: konzolyahry.sk

Minimapa vo väčšine prípadov poskytuje hráčovi zjednodušený pohľad na trať z tkz. vtáčej perspektívy. Existuje veľmi veľa druhov minimáp, ktoré ponúkajú rôzne detaily a špecifické vychytávky. Pre návrh našej minimapy sme sa inšpirovali rovnako ako v prípade tachometra hrou *Need For Speed: Hot Pursuit 2*,

ktorej minimapu môžeme vidieť na obrázku 2.10. Táto minimapa poskytuje jednoduchý pohľad na cestu, ktorá je reprezentovaná ako biela čiara, a vozidlá, ktoré sú reprezentované ako farebné kruhy.

Inšpirovaný týmto dizajnom minimapy, v našej hre implementujeme dizajnovo podobnú minimapu s malými úpravami. V prvom rade chceme v našej minimape vymeniť reprezentáciu vozidiel tak, že namiesto kruhov budú vozidla reprezentované šípkami (v tvare trojuholníkov). Táto zmena dodá okrem vzdialenosti aj informáciu o smere vozidiel. Ďalšia zmena, ktorú chceme vykonať pri návrhu našej minimapy je dynamicky zväčšovať/zmenšovať plochu, ktorú minimapa pokrýva v závislosti na rýchlosti vozidla. Čím rýchlejšie sa vozidlo pohybuje, tým viac plochy minimapa odhaľuje. Táto úprava by mala hráčovi pomôcť zavčasu reagovať na prudké zákruty alebo prekážky na ceste (iné vozidlá).

2.3.4 Meranie Kôl / Meranie Pozícií / Meranie Času

Z názvu tejto podkapitoly je jasné ,že chceme merať tri informácie o stave preteku. Tieto merania majú predovšetkým informovať hráča o jeho umiestnení, čase a kole, v ktorom sa nachádza. Sekundárne budú tieto informácie pomáhať ostatným entitám v hre tak, aby hra fungovala korektne a podľa naších požiadaviek. Konkrétne vieme vďaka meraniu kôl, kedy ukončiť závod a vďaka meraniu pozícií hráča vieme ako upravovať AI u ostatných vozidiel (viac v sekcií 2.2).

Čo sa týka návrhu týchto meraní, chceme predovšetkým, aby boli prehľadné a v prípade kola a umiestnenia zobrazovali hráčovi aktuálne kolo/umiestnenie a celkový počet kôl/vozidiel. Takto bude hráč vedieť koľko kôl ho ešté čaká rep. koľko vozidiel je v tomto preteku (v K.O. Mode sa ich počet mení).

Obr. 2.11: Návrh panelu na meranie času.

Dizajn merania času bude o trochu komplikovanejší. Ako môžeme vidieť na obrázku 2.11, chceme, aby náš dizajn obsahoval dva informačné panely. Prvý panel by mal zobrazovať informácie o:

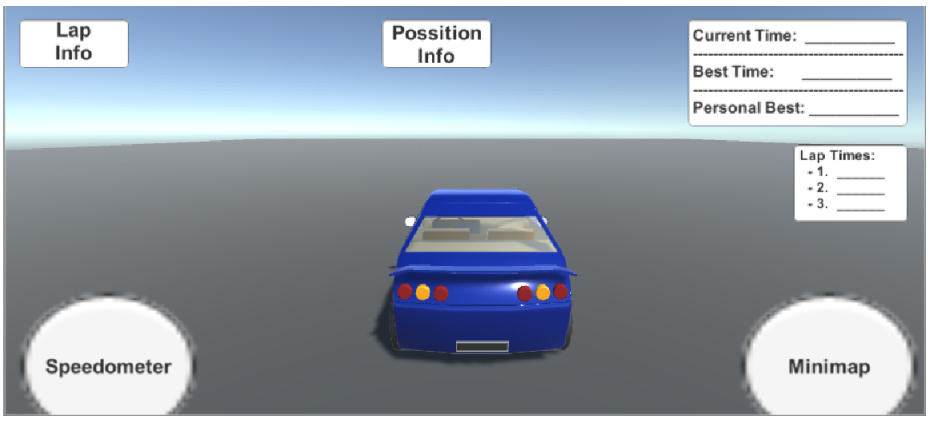
- aktuálnom čase – čas, počas ktorého hráč prejde jedno kolo. Tento blok sa po každom kole vynuluje.
- najlepší čas – obsahuje najlepší čas, ktorý bol stanovený na tejto trati bezohľadu na hráča.

- osobný najlepší čas – obsahuje najlepší čas hráča, ktorý na danej trati dosiahol.

Druhý panel, ktorý na obrázku 2.11 vidíme, obsahuje medzičasi každého kola, ktoré v danom preteku hráč dosiahol.

2.3.5 UI trate

Navrhli sme si a popísali všetky časti uživateľského rozhrania, ktoré sme počas preteku na trati chceli vidieť. Ostáva nám popísať, ako chceme, aby bolo toto rozhranie usporiadané na obrazovke. Usporiadanie užívateľského rozhrania je veľmi dôležité, pretože v najlepšom prípade poskytuje hráčovi prehľadné informácie o priebehu preteku, hráčovom progrese v preteku, rýchlosti vozidla a pozícií ostatných vozidiel. V najhoršom prípade rozhranie doslova bráni hráčovi v tom, aby videl samotný pretek, pretože je zle usporiadané a tým nepriehľadné a tažkopádne.

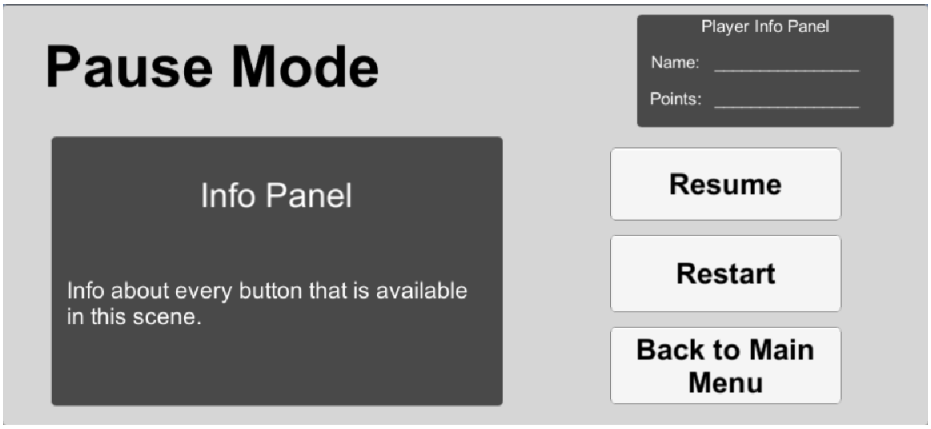


Obr. 2.12: Návrh UI trate počas preteku.

Na obrázku 2.12 vidíme návrh usporiadania nášho uživateľského rozhrania. Inšpiráciu sme si zobrali z hry *Grand Turisimo II*. Rozhranie danej hry je možné vidieť na obrázku 1.1. Ak sa na náš návrh pozrieme, všimneme si, že sa dá rozdeliť na dva časti resp. tvary. Horná časť nášho rozhrania je tvorená informáciami o kole, pozícií, časoch a medzičasoch hráča. Táto časť rozhrania je tvorená štvoruholníkovimy boxmy. Druhá časť informuje hráča o pozícií vozidla v preteku a rýchlosti vozidla. Tieto informácie sú obsiahnuté v kruhových boxoch. Veríme, že takto navrhnuté rozhranie bude pre hráča prehľadné a vizuálne príjemné.

Pauza

Počas samotného preteku sa môže udiať čokoľvek, čo hráčovi bráni pokračovať v hraní hry buď na krátku chvíľu, alebo si chce hráč pretek zopakovať, pretože sa mu veľmi nedarí, alebo chce hru proste ukončiť. Systém tkz. *Pauzy* umožňuje hráčovi hru kedykoľvek pozastaviť. Zapínanie takejto pauzy môžeme vidieť na obrázku 2.7.

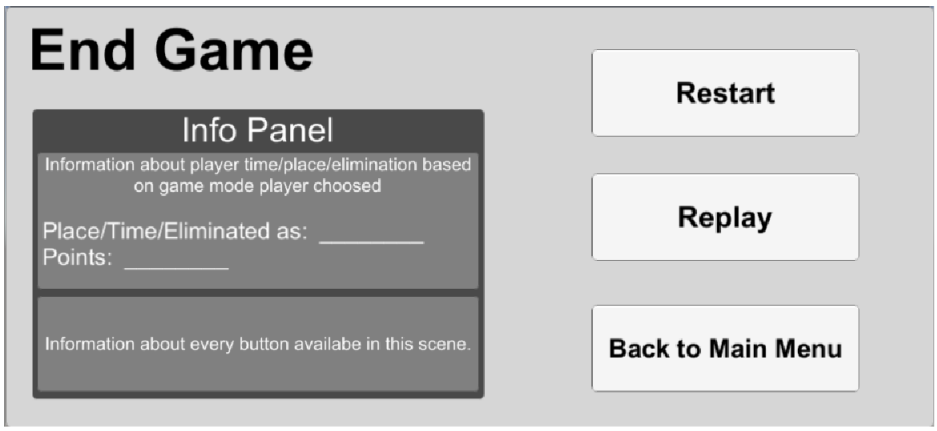


Obr. 2.13: Návrh panelu pre pauzu.

Chceme, aby pauza v našej hre hráčovi umožňovala možnosti ako sú návrat do hlavného menu, reštart preteku alebo pokračovať v rozohranej hre. Návrh vzhľadu našej pauzy môžeme vidieť na obrázku 2.13. Na tomto obrzáku vidíme panel informujúci hráča o funkcií jednotlivých tlačítok, ktoré mu dávajú možnosť hru ukončiť, reštartovať alebo v nej pokračovať. Pridali sme aj panel zobrazujúci uživateľské konto hráča a počet jeho bodov.

End Game

Koniec každého preteku bez ohľadu na to, aký úspešny, či neúspešny v ňom hráč skončil, je obvykle sprevádzaný panelom oznamujúcemu hráčovi informácie o výsledku preteku, prípadnom čase, ktorý dosiahol, rekordoch, ktoré prekonal alebo bodoch, ktoré získal. V našej hre chceme tak isto mať takýto panle, ktorý by sa zobrazil po preteku a hráčovi poskytol informácie a možnosti, ako v hre pokračovať.



Obr. 2.14: Návrh panelu na konci preteku.

Návrh nášho popretekového panelu môžeme vidieť na obrázku 2.14. V tomto návrhu hráčovi poskytujeme tri možnosti ako pokračovať:

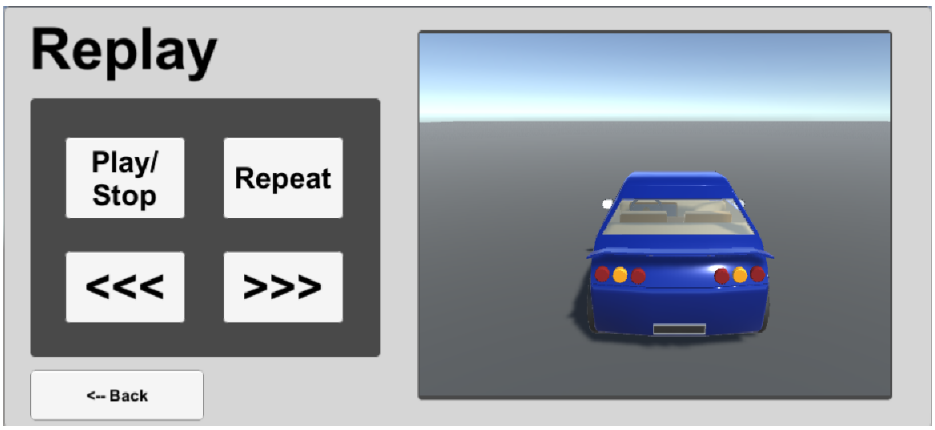
- reštartovať pretek od začiatku
- spustiť replay preteku

- uložiť výsledok a vrátiť sa do hlavného menu

Spolu s týmito možnosťami sa v tomto panely vyskytujú informácie o výsledku hráča v pretekoch (výsledok záleží na zvolenom móde) a počte bodov, ktoré mu jeho výsledok priniesol.

2.4 Replay

Možnosť znovuprehratia preteku je niečo, čo patrí nie len k závodným hrám, ale k pretekom ako takým. Hráč sa vďaka tejto možnosti môže po preteku pozrieť na záznam ako pretekal a vidieť všetky chyby, ktoré počas jazdy urobil. Alebo môže tento systém namiesto študijneho materiálu slúžiť ako prostá rekapitulácia skvelého preteku.



Obr. 2.15: Návrh replay panelu s možnosťou pretáčania.

Replay ako taký je niečo, čo sa v závodných hrách používa už od ranných počiatkov závodných hier a preto ho nechceme vynechať ani my v našej hre. Na obrázku 2.15 môžeme vidieť návrh replay panelu. Pri tomto návrhu nám šlo hlavne o jednoduché a intuitývne rozhranie. Tento návrh pozostáva z obrazovky, na ktorej beží prehrávaný záznam a sada štyroch tlačítok, vďaka ktorým je možné video spustiť/zastaviť, prehrať od začiatku alebo ho pretáčať dopredu či dozadu.

2.5 Herné módy

V sekcií 1.1.2 sme spomenuli, že jedným z prvkov *arcade style racers*, ktoré chceme implementovať v našej hre, sú herné módy. Herné módy menia mechaniku a správanie hry, či ponúkajú rozličné podmienky k výhre v danej hre. V závodných hrách herné módy zväčša oznamujú hráčovi podmienky, za ktorých je možné daný pretek vyhrať či prehrať. V našej hre sme si zvolili navrhnúť a implementovať tri hlavné herné módy. V nasledujúcich podkapitolách si tieto módy predstavíme a opíšeme.

2.5.1 Race Mode

Začneme módom, ktorý je najstarším a najklasickejším herným módom v histórií závodných hier. Týmto módom je tkz. Závodný Mód (Race Mode). V tomto

móde vozidlá navzájom pretekajú na trati niekoľko kôl. Vyhráva ten, kto je najlepší – teda prvý. Celkové víťazstvo v tomto móde závisí na umiestnení.

Krása tohto módu odráža krásu skutočných závodov. Konkretne sa jedná o fakt, že aj napriek skutočnosti, že sa hráč ocitol medzi poslednými, môže vďaka svojej vytrvalosti a hráčskemu umu skončiť medzi prvými. Herný mód ho v tomto nijako neobmedzuje a preto je to mód, ktorý sa zo závodných hier nevytratil ani po desaťročiach a ktorý chceme implementovať v našej hre.

2.5.2 Time Mode

Kdežto v Race Mode išlo o umiestnenie, v Časovom móde (Time Mode) ide o to poraziť jediného súpera, ktorým je čas. V tomto móde preteká hráč sám v súboji proti časovému limitu, ktorý je zväčša stanovený buď najrýchlejším hráčom, alebo je stanovený pevne. Tento mód je zvláštny tým, že sám o sebe nie je nikdy porazený. Smozrejme môže hráč poraziť pevne stanovenú časovú hranicu, ale nikdy neporazí všetkých ostatných hráčov, ktorý nastavia nové časové rekordy. Keď sa aj stane, že nový rekord nastaví práve náš hráč, iný hráč tento rekord môže prekonať. Práve preto je tento mód nekončiaci súboj rýchlosti a hráčskych schopností a preto nesmie chýbať v našom katalógu herných módov.

Ghost Car

Obr. 2.16: Návrh ghost car. Hráč - vľavo, Ghost car - vpravo

Viac než informácia o čase najlepšieho z najlepších je vidieť jazdu, ktorá danému hráčovi umožnila stať sa najlepším. Duch (Ghost) označuje vozidlo najrýchlejšieho hráča, ktorého jazda je simulovaná počas hrania módu. Ako ukazuje obrázok 2.16, tento duch nie je pevný objekt a hráč ním môže kedykoľvek prechádzať (ako cez ducha). Takto môže hráč vidieť jazdu, ktorú sa snaží prekonať a vidieť všetky jej silné a slabé stránky. V istom momente hry sa môže takýmto spôsobom pretekať sám proti sebe a tak sa tlačiť k tomu, aby bol lepší a lepší.

2.5.3 K.O Mode

Tento posledný mód má u hráčov vzbudiť inštinkt prežiť. Eliminačný mód (K.O. Mode) totiž po každom kole eliminuje posledného pretekára. Preteká sa do momentu, kedy neostane len jeden pretekár, ktorý vyhráva.

V tomto móde, na rozdiel od Race Mode, nestačí hrať dobre posledné kolá, pretože každá chyba môže znamenať vyradenie. Hráč v tomto móde sa musí snažiť všetkými silami o to, aby nebol nikdy posledný, čím sa tento mód líši od všetkých módov, ktoré sme tu doteraz spomenuli. Tieto kvality z neho robia ďalší a posledný mód, ktorý chceme priviesť do nášej hry.

2.6 Body a ich využitie

V úvodnej kapitole v sekcií 1.1.3 sme vyslovili požiadavku, že chceme, aby sa v našej hre vyskytovali body, ktoré môže hráč získavať. Body v hrách a v tomto prípade závodných hrách slúžia predovšetkým ako spôsob odmeny pre hráča za úspešné zvladnutie preteku. Tieto body môžu byť získavané za rôznych okolností a využité veľa rôznymi spôsobmy. V tejto sekcií sa pozrieme na spôsob získavania bodov v našej hre a spôsob, ako by ich hráč mohol v našej hre využiť.

2.6.1 Získavanie bodov

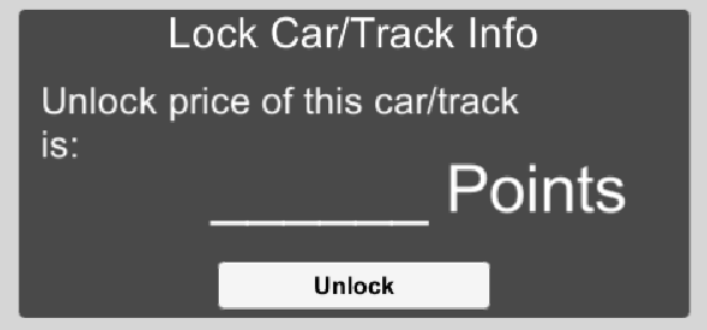
Body v závodných hrách je pre hráča možné získavať viacerými spôsobmy. Jeden zo spôsobov je zbierať body počas preteku, pričom body sú na trati reprezentované zvyčajne ako mince alebo iný cenný objekt. Tento spôsob núti hráča, aby schopnosť vítaziť v závodoch kombinoval so schopnosťou zbierať body počas preteku. Takýto spôsob umožnuje hráčovi v hre napredovať buď zbieraním bodov, víťazstvami v pretekoch alebo kombináciou týchto dvoch spôsobov.

Druhý spôsob funguje na systéme odmien. Jednoducho povedané, ak hráč zvíťazí v preteku, získa body. Počet bodov, ktoré hráč získa, by záviselo od herného módu, v ktorom práve preteká, umiestenia, poprípade v čase. Správne nastavené podmienky by mohli v prípade odmeňovania zaistiť, že sa hráč sústredí predovšetkým na výhru v preteku z dôvodu získania bodov.

Vhodné nastavenie bodových odmien by bolo ukončenie preteku medzi tromi najlepšími hráčmi, alebo byť eliminovaný v poslednej trojici hráčov, alebo ukončenie preteku v danom časovom intervale (záleží na hernom móde). V rámci nášho herného návrhu volíme spôsob odmeňovania v kombinácií s vhodne nastavenými podmienkamy odmien.

2.6.2 Unlock áut a tratí

Dôležitým faktorom, ktorý odlišuje závodné hry od dobrých závodných hier, je rozmanitosť tratí a vozidiel, ktoré má hráč počas hry k dispozícií. Väčšina hier, ktoré poskytujú hráčovi body počas hry, má drtivú väčšinu týchto kolekcií tratí a vozidiel zamknuté. Skutočnosť, že hráč nemá k dispozícií sto percent kolekcie vozidiel a tratí ho motivuje k získavaniu bodov, ktoré potrebuje na ich odomykanie.

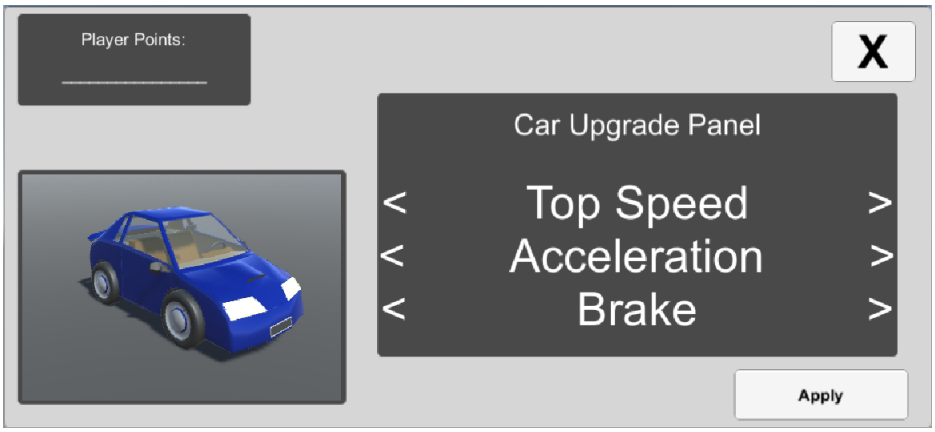


Obr. 2.17: Návrh odomykacieho panelu pre vozidlá a trate

Na obrázku 2.17 môžeme vidieť návrh odomykacieho panelu pre vozidlá a trate. Kedykoľvek, kedy hráč narazí na vozidlo alebo trať, ktorá je zamknutá, môže ju odomknúť za určitý počet bodov. Tieto body sú nevratné, čo znamená, že hráč nebude mať možnosť vozidlá a trate spätne zamykať, aby dostal body späť za ich uzamknutie.

2.6.3 Upgrade áut

V prekonávaniu časových rekordov a vyhrávaniu pretekov potrebuje hráč okrem hráčskych schopností a citlivosti aj výkonné vozidlá. Vďaka bodom si hráč môže odomknúť lepšie a rýchlejšie vozidlá. Ale čo ak na to nemá dosť bodov? Aby hráč necítil stagnáciu do chvíle, kedy má dosť bodov na odomknutie nového vozidla alebo trate, často sa v závodných hrách používa tkz. upgrade vozidiel.



Obr. 2.18: Návrh upgradovacie panelu.

Upgrade vozidiel umoňuje hráčovi upravovať parametre alebo vzhľad vozidla a tak zvyšovať jeho výkon počas preteku. Tieto vylepšenia sú často lacnejšie ako odomknutie nového vozidla a teda ich hráč môže vykonávať oveľa častejšie. Je veľa možností čo všetko si hráč môže na vozidle upraviť a ako by mal tento proces vyzerať. Na obrázku 2.18 môžeme vidieť návrh upgradovacieho panelu pre našu hru. V tomto panely má hráč k dispozícií tri veličiny, ktoré môže na svojom vozidle vylepšovať. Tieto veličny sú:

- maximálna rýchlosť

- zrýchlenie
- výkon bŕzd

Všetky tieto veličiny majú u každého vozidla svoj limit (nedajú sa vylepšovať do nekonečna). Každá úprava má jasne stanovenú cenu, ktorá sa mu bude strhávať z počtu bodov, ktoré hráč má.

2.7 Vytváranie nových účtov / prihlasovanie sa k starým účtom

V predchádzajúcich podkapitolách 2.6 a 2.5 sme spomenuli, že v našej hre chceme používať body vďaka ktorým môže hráč získavať nové vozidlá a trate, a taktiež sme spomenuli možné časové rekordy, ktoré môže hrač stanoviť resp. prekonávať. Všetky tieto úspechy by hráč po každom vypnutí hry stratil a musel začínať odznova. Z tohto dôvodu boli v mnohých hrách zavedené hráčske účty, ktoré hráčov progres a rekordy v hre ukladá vďaka čomu hráč nemusí začínať po každom zapnutí hry odznova.

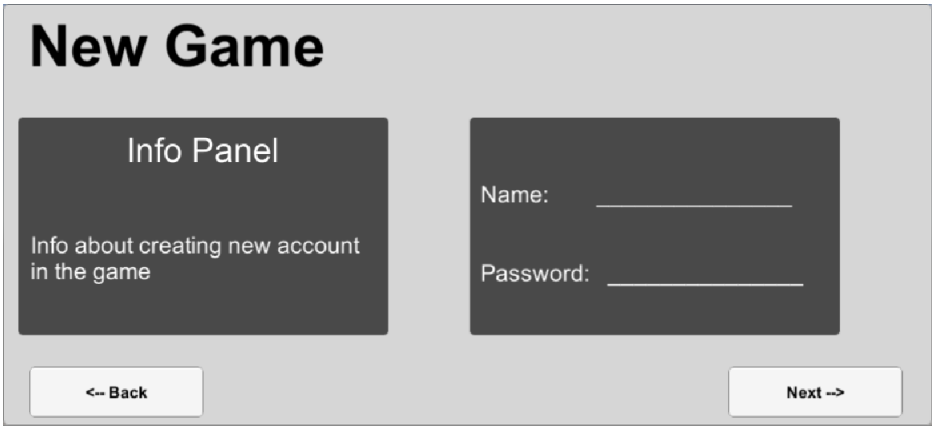
2.7.1 Čo majú účty obsahovať

Keďže sme spomenuli, že hráčske účty slúžia na uloženie progresu a prípadných rekordov hráča, v tejto sekcií popíšeme, čo všetko chceme uchovávať v hráčskych účtoch. Hráčske účty v našej hre musia uchovávať:

- meno hráča
- heslo hráča
- body
- odomknuté vozilá a trate
- vylepšenia jednotlivých vozidiel
- časové rekordy hráča na jednotlivých tratiach

Uchovanie týchto informácií by malo zaistiť, že hráč v našej hre nestratí progres, ktorý si počas hrania vybudoval, a bude mať k dispozícií všetky vozidlá, upgrady, trate a rekordy, ktoré počas hry dosiahol.

2.7.2 Vytváranie nových účtov



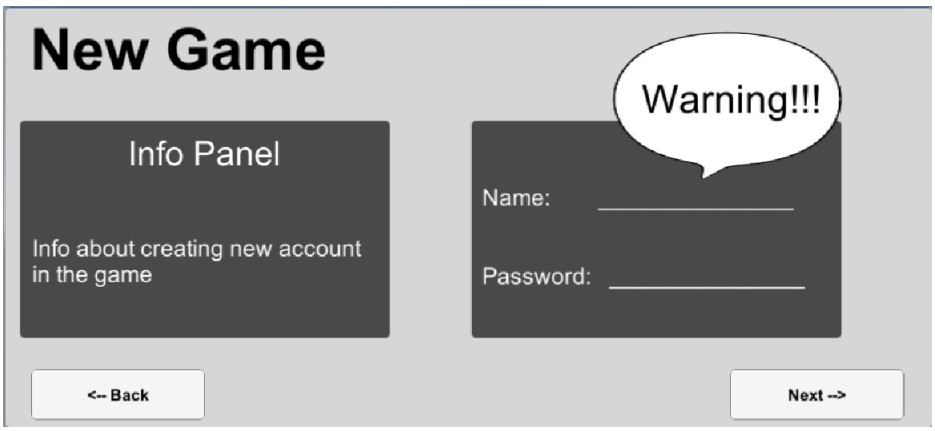
Obr. 2.19: Návrh panelu na vytváranie nových účtov.

Na to, aby hráč mohol používať svoj hráčsky účet, musí si byť schopný hráčsky účet vytvoriť. Na obrázku 2.19 vidíme návrh panelu, ktorý by mal hráčom umožnovať vytvorenie hráčskeho účtu. Tento panel obsahuje informácie o vytváraní nových účtov a textové polia na meno a heslo nového hráčskeho účtu. Tento návrh by mal byť dostatočne jednoduchý pre každého hráča.

Možné chyby v prihlasovaní

Bez ohľadu na jednoduchosť nejakého návrhu a kvalitu popisu k použitiu, nesmieme podceňovať schopnosť hráčov ignorovať akýkoľvek obmedzenie pri vytváraní nových účtov. Je nutné stanoviť obmedzenia, ktoré hráč musí pri vytváraní nových účtov dodržať. Tieto požiadavky sú:

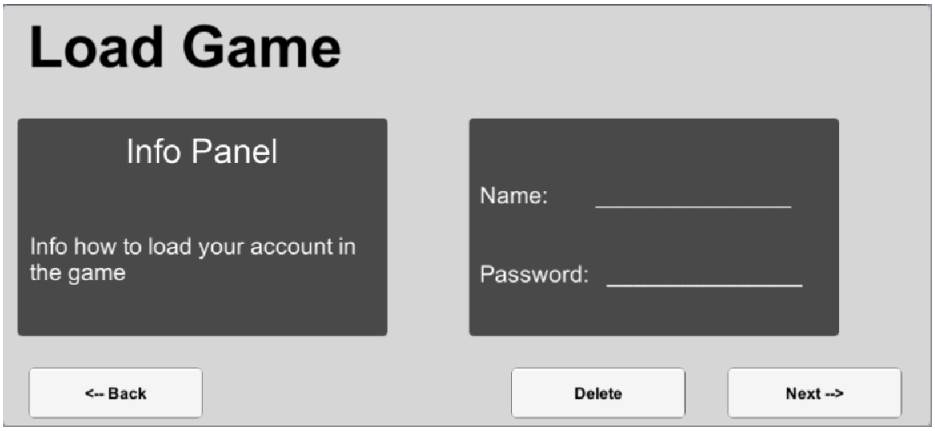
- meno účtu musí mať aspoň tri znaky
- meno musí byť unikátne (nesmie ho používať iný hráč)
- heslo musí mať aspoň tri znaky



Obr. 2.20: Ukážka upozornenia na chybný vstup pri vytváraní nových účtov.

Dodržanie týchto požiadavkov musíme kontrolovať a v prípade porušenia týchto podmienok hráča upozorniť na toto porušenie. Na obrázku 2.20 vidíme, ako by mal fungovať upozornenie hráča v prípade porušenia našich požiadavok. Upozornenie sa zobrazí v podobe textu, ktoré hráčovi presne popíše aké porušenie vykonal pri vytváraní nového účtu.

2.7.3 Prihlasovanie sa k starým účtov



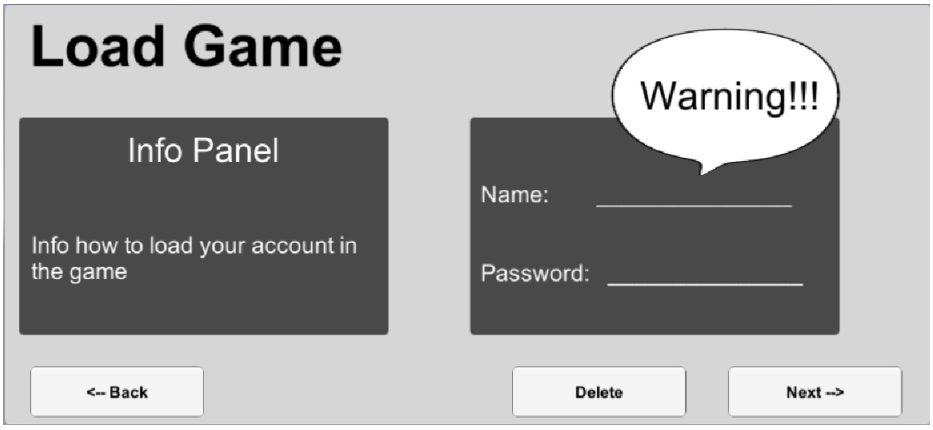
Obr. 2.21: Návrh panelu na prihlasovanie sa a vymazanie starých účtov.

Keď vieme vytvárať hráčske účty, musíme hráčovi umožniť prihlásiť sa k tomuto účtu, ak znovu navštívi našu hru (za predpokladu že si účet vytvoril). Samozrejme by sme mali dať hráčovi šancu si v prípade potreby svoj hráčsky účet odstrániť. Na obrázku 2.21 vidíme návrh rozhrania na prihlasovania a odstraňovania hráčskych účtov. Môžeme si všimnúť že toto rozhranie je skoro totožné s rozhraním na vytváranie hráčskych účtov a to hlavne z dôvodu konzistencie.

Možné chyby v prihlasovaní

V kontexte prihlásovania a vymazovania hráčskych účtov musíme rovnako ako v prípade vytvárania hráčskych účtov stanoviť nejaké podmienky, ktoré hráč musí pri prihlasovaní resp. mazaní účtov, dodržať. Tieto podmienky sú:

- meno účtu, ku ktorému sa chce hráč prihlásiť, reps. odstrániť, musí byť korektné (musí existovať)
- heslo musí odpovedať menu hráčskeho účtu.



Obr. 2.22: Ukážka upozornenia na chybný vstup pri prihlásovaniu sa a mazania starých účtov.

Tieto podmienky by mali zaistiť korektné prihlasovanie a mazanie hráčskych účtov. V prípade porušenia týchto podmienok rovnako ako v prípade porušenia podmienok pri vytváraní hráčskych účtov chceme hráča upozorniť textovou správou ako je to zobrazené v návrhu na obrázku 2.22.

2.8 Menu

Každá závodna hrá stojí na vozidlách a trati, ale k tomu, aby bola závodná hra kompletná, je nutné mať k dispozícií nejaké uživateľské rozhranie resp. Menu, ktoré hráčom umožní hru nastavovať, upravovať, vyberať trate, vozidlá atď.
Vzhľad a komplikovanosť týchto rozhraní sa líši a záleží na tom ako a kto dané rozhranie navrhuje.

Jednou z hlavných požiadaviek pre našu hru bolo intuitívne uživateľské rozhranie. V predošlých podkapitolách sme popísali návrhy rozhrání niektorých scén. V tejto sekcií si prejdeme zvyšok týchto scén a ich uživateľskych rozhraní.

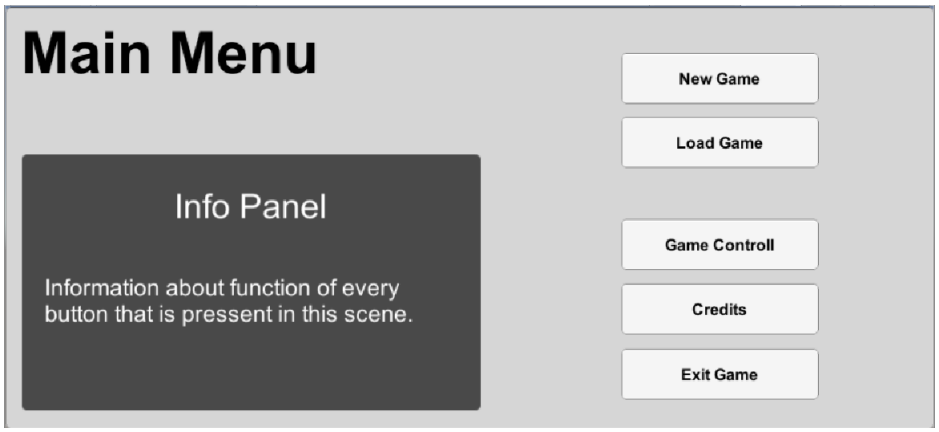
2.8.1 Welcome

Obr. 2.23: Návrh uvítacej scény.

Prvá scéna, ktorú chceme aby sa pri zapnutí hry hráčovi zobrazila, je scéna, ktorá hráča privíta v našej hre. Tieto scény sa vo vieohrách označujú ako *Welcome scenes*. Na obrázku 2.23 môžeme vidieť náš návrh takejto scény. Chceme, aby táto scéna bola jednoduchá a aby ju hráč mohol jednoducho preskočiť stlačením akehokoľvek klávesu na klávesnici alebo tlačítka myši.

2.8.2 Štart Menu

Prvý kontkakt hráča s každou hrou je scéna hlavného menu. Táto scéna poskytuje hráčovi možnosti vstúpiť do rôznych aspektov hry ako sú nastavenia, výber herných módov, informácie o tom, ako sa daná hra ovláda, alebo možnosť hru ukončiť.



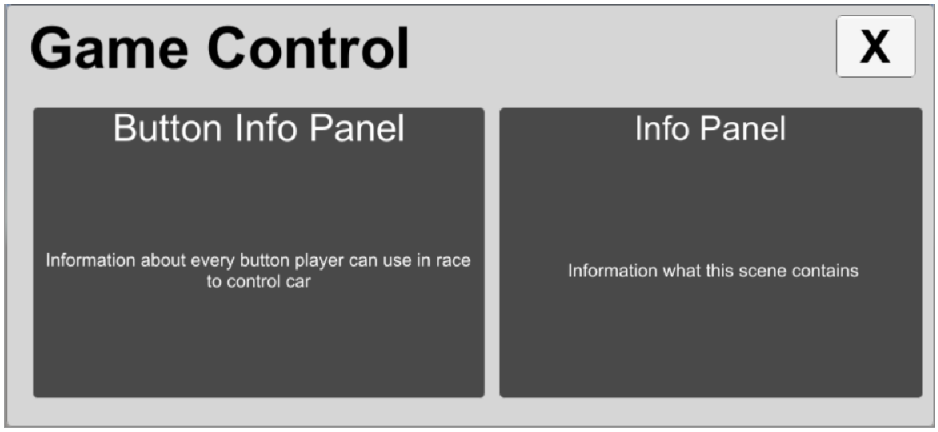
Obr. 2.24: Návrh hlavného menu.

Obrázok 2.24 zobrazuje návrh nášho hlavného menu. V našom návrhu sme chceli hráčovi poskytnúť možnosti ako sú:

- vytvorené nového účtu
- prihlásenie sa k starému účtu
- informácie pre hráča o ovládaní hry
- možnosť zobrazenia tituliek
- možnosť ukončiť hru

Súčasťou tohto návrhu je informačný panel, ktorý má za úlohu informovať hráča o funkcií jednotlivých tlačítiek.

Game Control Info



Obr. 2.25: Návrh panelu ktorý hráča informuje o tom ako sa naša hra ovláda.

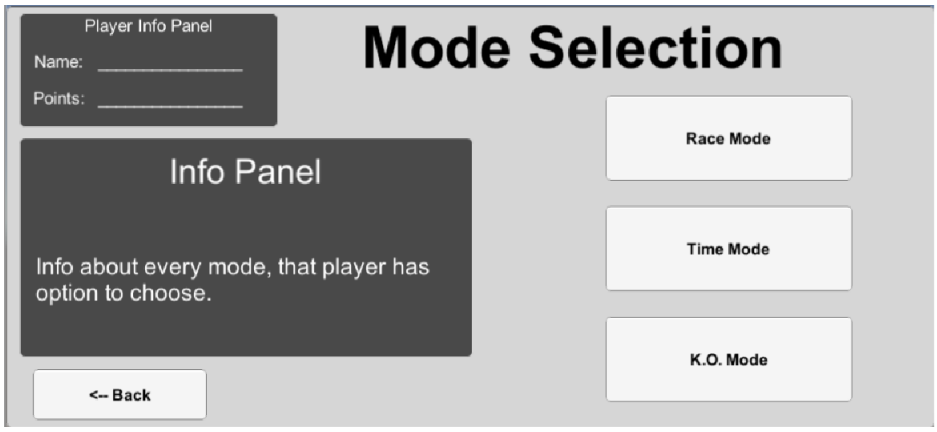
V našom hlavnom menu sme spomenuli možnosť hráča pozieť sa na informácie o ovládaní hry. Na obrázku 2.25 sa snažíme ukázať návrh takéhoto panelu. Tento panel by sa mal skladať z informácií aké tlačítka je možné počas hrania použiť a akú funkciu tieto tlačítka v hre majú. Tento panel by mal hráčom pomôcť rýchle pochopiť ovládanie našej hry.

End Credits

Ďalšia možnosť, ktorú si hráč v našom hlavnom menu môže zvoliť, je prejsť do scény s titulkami. Tieto scény vo svete video hier slúžia k tomu, aby hráčom poskytli informáciu o tom, kto sú autori, dizajnéri, programátori, animátori a mnohí iní, ktorý stoja za touto hrou.

V našom prípade chceme spomenúť autorov všetkých modelov, hudby, animácií, fondov a iných vecí, ktoré sme v našej hre použili. Zároveň chceme spomenúť testerov, ktorý nám hru pomáhali testovať a zlepšovať.

2.8.3 Mode Selection



Obr. 2.26: Návrh panelu na výber herného módu.

Jeným z cieľov, ktoré sme spomenuli v sekcií 2.11, sú herné módy (informácie o konkrétnych módoch sa nachádzajú v sekcií 2.5). Ako súčasť užívateľského rozhrania chceme hráčovi poskytnúť možnosť zvoliť si, aký herný mód si chce hráč zvoliť. Obrázok 2.26 popisuje, ako by mla táto scéna vyzerať. Chceme, aby táto scéna bola priamočiara a jednoducho čitateľná. Náš návrh obsahuje herné módy, ktoré si hráč môže zvoliť. Info panel, ktorý popisuje obsah jednotlivých módov. Nakoniec sme pridali panel, ktorý obsahuje meno uživateľského účtu hráča a počet bodov, ktoré hráč doposiaľ získal.

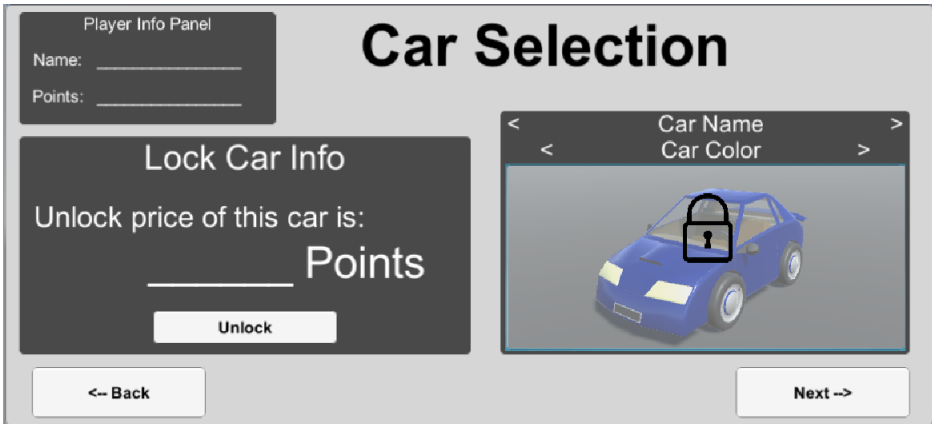
2.8.4 Car Selection

V podkapitole 2.1 sme v našej hre požadovali vozidlá a v podkapitole 2.6.2 sme si popísali odomykanie vozidiel, ktoré chceme v hre mať. V tejto časti si popíšeme ako by malo vyzerať uživateľské rozhranie panelu na odomykanie vozidiel v našej hre a čo všetko by malo obsahovať.

Obr. 2.27: Návrh panelu na výber vozidiel.

Na obrázku 2.27 vidíme vzhľad uživateľského rozhrania, ktoré by sme v našej hre chceli hráčom poskytnúť. V tomto rozhraní vidíme tri panely pričom prvý panel obsahuje meno hráčskeho účtu hráča spolu s počtom bodov, ktoré má hráč k dispozícií. Ďalší panel umožňuje hráčovi prechádzať jednotlivé odomknuté vozidlá a poprípade meniť ich farbu. Posledný panel dáva hráčovi možnosť upgradeovať zvolené vozdilá (viac o upgradoch v sekcií 2.6.3). Tento panel zároveň hráčom poskytuje informácie o vlastnostiach daných vozidiel ako sú:

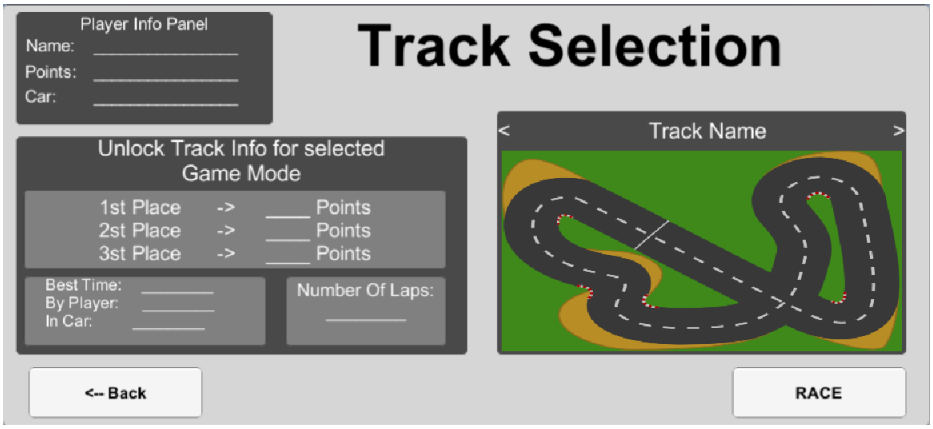
- maximálna rýchlosť
- výkon vozidla
- výkon bŕzd
- typ pohonu kolies
  - pohon predných kolies
  - pohon zadných kolies
  - pohon všetkých kolies



Obr. 2.28: Návrh panelu na výber vozidiel, ktorý zobrazuje zamknuté vozidlo.

Samozrejme sa môže stať (a pravdepodobne sa aj stane), že hráč pri prechádzaní vozidiel narazí na vozidlo, ktoré nie je odomknuté. V takomto prípade chceme hráča informovať o nedostupnosti daného vozdila. Zároveň ho chceme informovať o cene za odomknutie daného vozidla a poskytnúť hráčovi možnosť vozidlo odomknúť. Návrh tohto dizajnu môžeme vidieť na obrázku 2.28.

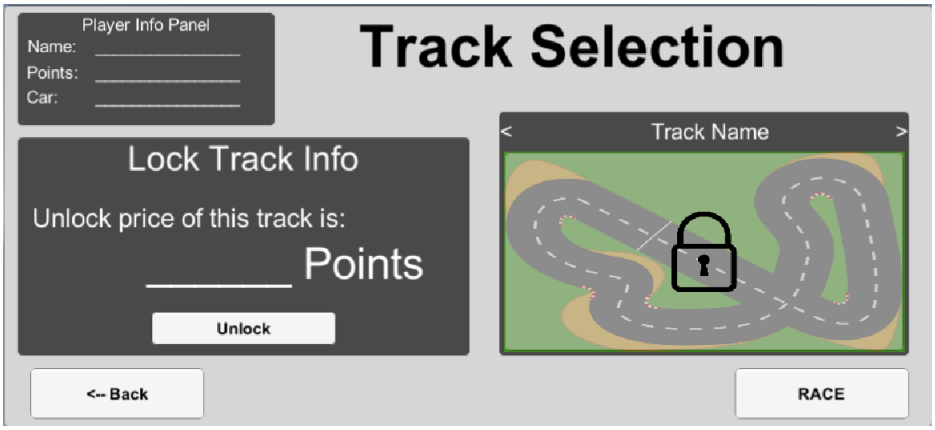
2.8.5 Track Selection



Obr. 2.29: Návrh panelu na výber tratí. Zdroj: dlpng.com

Rovnako ako v prípade výberu vozidiel chceme hráčovi poskytnúť vhodné užívateľské rozhranie na prechádzanie a výber tratí. Obrázok 2.29 ilustruje tri panely, ktoré toto rozhranie tvoria. Prvý panel obsahuje meno hráčskeho účtu hráča spolu s počtom bodov a oproti panelu na výber vozidiel obsahuje dodatočnú informáciu o vozidle, ktoré si hráč zvolil (predpokladáme, že hráč si najpr zvolí vozidlo). Druhý panel rovnako ako v prípade výberu vozidiel umožňuje hráčovi prechádzanie jednotlivých tratí. Posledný panel je trochu komplikovanejší. Obsahuje totiž informácie o tom, koľko bodov hráč dostane vo výhre v preteku. Táto informácia je závislá na hernom móde, ktorý si hráč zvolil (voľba herných módov je popísaná v sekcií 2.8.3). Na záver tento panel obsahuje informácie o počte kôl, ktoré hráč musí na danej trati vo vybranom hernom móde odohrať a informácie o časovom rekorde trate. Táto informácia pozostáva z:

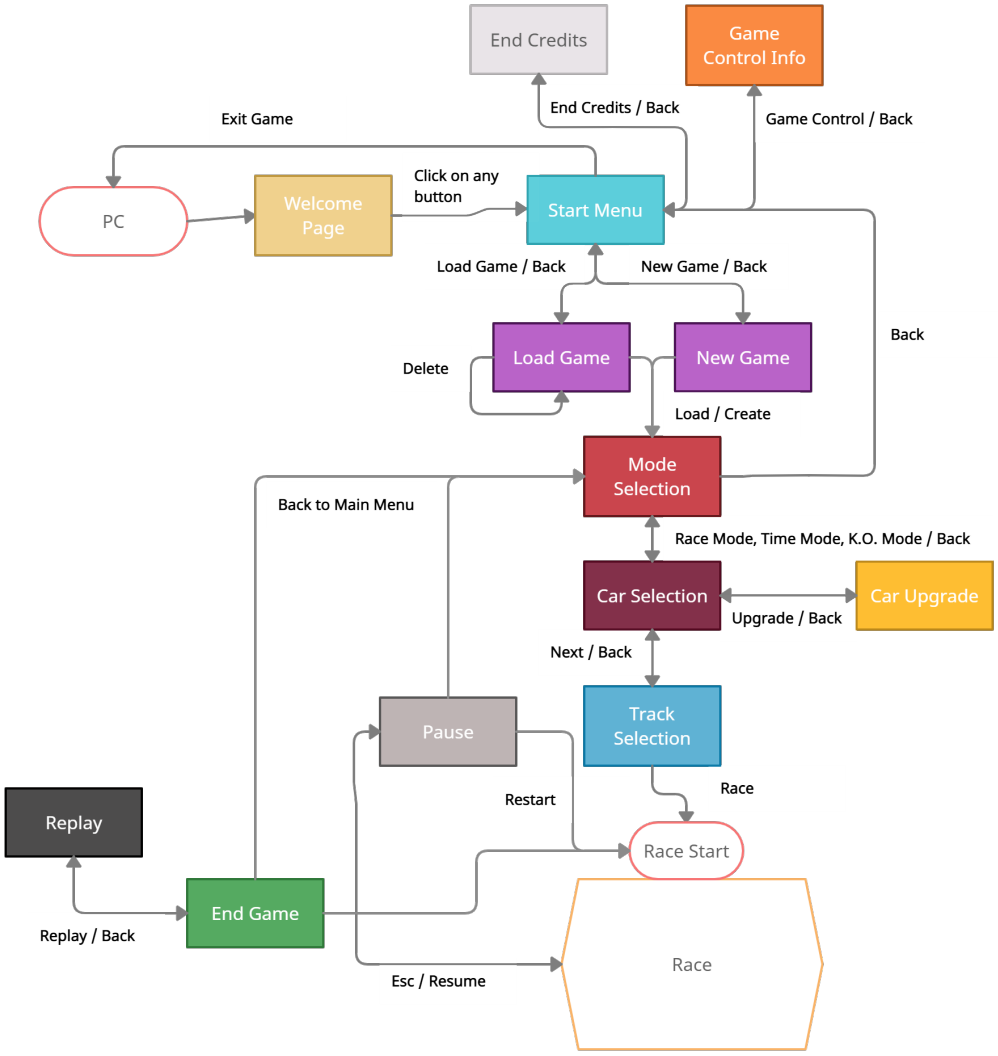
- čas rekordu
- meno hráča zodpovedného za rekord
- vozidlo v ktorom bol rekord vytvorený



Obr. 2.30: Návrh panelu na výber tratí so zamknutou traťou. Zdroj: dlpng.com

Rovnako ako v prípade výberu vozidiel, tak aj v prípade výberu tratí môžeme naraziť na trať, ktorá je zamknutá. Informáciu o jej zamknutí, informácia pre hráča koľko odomknutie tejto tratí stojí a možnosť trať odomknúť sme navrhli rovnako ako pri výberu vozidiel. Návrh situácie so zamknutou traťou môžeme vidieť na obrázku 2.30.

2.9 Pohyb hráča medzi scénami



Obr. 2.31: Ukážka pohybu hráča v hre.

V priebehu tejto kapitoly sme spomínali uživateľské rozhranie jednotlivých scén, možné panely ktoré obsahujú a mnohé iné detaily. Nespomenuli sme však, ako si predstavujeme hráčov prechod medzi jednotlivými scénami. Ktorá scéna nasleduje po ktorej? Aké tlačítko posunie hráča kam?

Na obrázku 2.31 môžeme vidieť mapu našej hry s hľadiská scén v ktorých sa hráč môže počas hry ocitnúť. Obdĺžniky v tejto mape reprezentujú jednotlivé scény, šípky reprezentujú smer prechodu medzi scénami a hodnota ší piek zobrazuje tlačítko, ktoré musí hráč stlačiť, aby sa do tejto scény dostal.

V tejto mape sa nachádzajú aj tri geometricky rozdielne tvary, ktoré reprezentujú:

- *PC* – počítač na ktorom je naša hra umiestnená
- *Race Start* – okamžik, kedy sa hra začína (potrebujeme ju na odlíšenie reštartu a návratu k rozohratej hre)

- *Race* – samotný pretek

2.10 Editor

V tejto kapitole sme sa pokúsili navrhnúť, ako bude naša hra vyzerať. Špecifikovali sme všeto, čo by sa malo v našej hre vyskytovať a aké správanie by sme očakávali od jednotlivých entít v našej hre. Všetky tieto návrhy popisovali mechaniky, výzor a správanie herných entít, ktoré by užívatelia mohli oceniť.

Nesmieme zabúdať, že pod pojmom užívateľa nemyslíme len hráčov, ale rovnako tak dizajnérov, ktorí by mohli v budúcnosti našu hru rozširovať.

Chceme teda, aby v našej hre existovala mechanika, na jednoduché pridávanie a úpravu tratí a vozidiel. Konkrétne chceme aby mohli dizajnéry pridávať vozidlá a trate bez nutnosti úpraviy skriptov, alebo zložitejších aspektov tratí a vozidiel.

2.11 Zhrnutie Cieľov

Na záver kapitoly si zhrnieme všetko, čo chceme obsiahnuť v našej hre. Budeme sa snažiť čo možno najlepšie popísať tieto ciele bez toho, aby sme zachádzali do úplných detailov, pretože od toho tu je celá táto kapitola.

Naším cieľom je vytvoriť závodnú hru, ktorá obsahuje realistické, aspoň semi kvalitné hrateľné vozidlá, ktoré budú schopné pohybovať sa a zatáčať po tratiach, schopné brzdiť a prechádzať plynule do šmyku. Tieto vozidlá by mali obsahovať zvukové a svetelné efekty v kombinácií s animáciami kolies. Záverom by vozidlá mali reagovať na vstup hráča, byť schopné respawnu a obsahovať dynamickú kameru s možnosťou spustenia quick kamery.

Ďalším cieľom je vytvoriť AI, ktorá bude schopná ovládať vozidlá v našej hre a dokázať korektne prekonať trať. Okrem samotného ovládania požadujeme, aby bola AI schopná vyhýbať sa statickým a dynamickým prekážkam na trati a dynamicky prispôsobovať rýchlosť vozidiel podľa pozície hráča.

Dôležitý cieľ, ktorý v našej hre očakávame, sú body pre našich hráčov, ktoré budú získavať za úspešné zvládnutie pretekov za určitých podmienok, ktoré sa budú líšiť v závislosti na hernom móde. Tieto body môže hráč využívať na odomykanie vozidiel a tratí alebo vylepšovanie vozidla.

Herné módy, ktoré sme spomenuli, sú ďalším cieľom, ktorý chceme v našej hre splniť. Požadujeme, aby naša hra ponúkala hráčovi herné módy. Konkrétne:

- *Race Mode* – mód, v ktorom hráč preteká o prvenstvo.
- *Time Mode* – mód, v ktorom sa hráč snaží skončiť pretek v určenom časovom interave. Tento mód obsahuje *ghost car*, ktorý simuluje jazdu najrýchlejšieho hráča.
- *K.O. Mode* – mód, v ktorom sa po každom kole eliminuje posledný pretekár. Cieľom je udržať sa v preteku čo najdlhšie.

Ďalší cieľ, ktorý si stanovujeme, je, aby naša hra obsahovala hráčske účty, ktoré hráčovi umožnia ukladanie svojho progresu. Tieto účty bude hráč schopný vytvárať, mazať a prihlasovať sa do nich. Každý účet bude obsahovať:

- meno účtu
- heslo
- odomknuté vozidlá a úroveň ich vylepšení
- odomknuté trate
- časové rekordy hráča na každej trati

Cieľ, ktorý od našej hry požadujeme, je poskytnúť hráčovi možnosť replay, vďaka ktorému si hráč môže prehrať záznam preteku. V tomto kontexte požadujeme, aby v našej hre bolo intuitívne užívateľské rozhranie (UI), v ktorom sa hráč bude schopný jednoducho orientovať a pohybovať. Toto rozhranie by malo pozostávať z týchto scén:

- uvítacej scény
- hlavného Menu
- titulky
- informácie o ovládaní hry
- vytvárania/prihlasovania sa k hráčskym účtom
- výberu herných módov
- výberu, odomykania a upgradu vozidiel
- výberu a odomykania tratí
- replay
- pauza
- scéna po ukončení preteku

Posledný cieľ je vytvoriť prieľadný editor na jednoduché pridávanie nových vozidiel a tratí, ktorý uľahčí prácu nie len nám, ale aj dizajnérom, ktorý by chceli v budúcnosti hru rozširovať.

3. Analýza

V minulej kapitole sme si popísali návrh a dizajn všetkých aspektov našej hry a špecifikovali sme jednotlivé ciele, ktoré chceme v hre implementovať. V tejto kapitole si uvedieme dôležité implementačné problémy, ktoré potrebujeme rozhodnúť, uvedieme si najpoužívanejšie možnosti implementácie spolu s výhodamy a nevýhodamy, ktoré nám tieto riešenia prinesú. Nakoniec popíšeme a odôvodníme, ktorou metódou sme si zvolili daný prvok implementovať.

3.1 Engine

V súčastnosti existujú nástroje, ktoré prográmatorom pomáhajú pri tvorbe videohier. Tieto nástroje umožňujúce jednoduché vytváranie, manipuláciu a skriptovanie objektov nazývame *Herné Enginy*. Keďže nechceme našu hru písať úplne sami (od základu), budeme sa aj my spoliehať na asistenciu vhodného herného enginu.

Engine, ktorý si vyberieme, musí spĺňať isté parametre, ktoré sú pre nás dôležité k tomu, aby sme boli schopní vytvoriť našu hru. Tieto parametre sú:

- renderovanie 3D objektov
- vhodná grafika výsledného produktu
- vytváranie 3D objektov a ich ľahká manipulácia
- možnosť importovania modelov
- možnosť vytvárať jednoduché animácie
- skriptovanie objektov pomocov jazyka C++ alebo C#
- bohatá dokumentácia a užívateľská komunita

V súčastnosti sa na trhu nachádza mnoho herných enginov, ktoré by sme mohli použiť. Bezkonkurenčne najpopulárnejšie sú herné enginy **Unity** a **Unreal**.

3.1.1 Unity

Unity bol vydaný v roku 2004 s cieľom sprístupniť herný development každému a nie len veľkým spoločnostiam. Určite nejeden developer nezávislých hier buď začínal svoju cestu s *Unity*, alebo ho stále používa. S jeho intuitívným dizajnom a možnosťou písania skriptou v jazyku C# je práca v *Unity* ľahko pochopiteľná. *Unity* sa teší veľkej komunite, obrovskému množstvu tutoriálov a online kurzov a taktiež veľkým množstvom *assetov* v *Unity Asset Store*.

3.1.2 Unreal

Tento engine bol vyvinutý a vydaný v roku 1998 spoločnosťou *Epic Games*. Spoločnosť *Unreal* bola vždy známa svojou prelomovou grafikou a prítomnosťou v elite *AAA* hier. Jeho dokumentácia nedosahuje tak kvalitnú dokumentáciu a nemá ani tak veľkú komunitu ako *Unity*. Tento engine taktiež obsahuje niektoré komplikované rozhrania, nehovoriac o tom, že kódovacím jazykom je C++, ktorý patrí na stranu komplikovanejších jazykov. Na druhú stranu, tento herný engine posúva grafiku na najvyššiu možnú úroveň pri zachovaní efektívnosti. Vďaka najnovším snahám spoločnosti *Epic Games* o to, aby bol *Unreal* viac prístupnejší začiatočníkom, bol zavedený koncept tkz. *Blueprints* spolu s pokračujúcou snahou o vylepšovanie dokumentácie, tutoriálov a silným dôrazom na budovanie silnejšej a väčšej komunity, ktorá by podporovala začinajúcich developerov.

3.1.3 Porovnanie

Oba enginy spĺňajú naše paramerte. *Unreal* je skvelý v mnohých ohľadoch, ale pre začiatočníkov je veľmi tažké a zdĺhavé naučiť sa ho správne a efektívne ho používať. Na druhú stranu je *Unreal* absolútna špička, čo sa týka vytvárania hier s high-end grafikou. Samozrejme, *Unity* má taktiež svoje chyby a pravdepodobne s ním nedosiahneme takú kvalitnú grafiku, ale aj napriek tomu produkuje skvelé výsledky.

Ak by sme to mali zhrnúť, tak vzhľadom na fakty, že v dobe písania tejto práce nemá autor veľké skúsenosti s vývojom hier, C# je jazyk, ktorý autor ovláda na vyššej úrovni ako C++, kvalitnejšej dokumentácií a väčšej komunite, si volíme *Unity* ako engine, v ktorom budeme tvoriť našu hru, kedže je to skvelá voľba pre začínajúcich herných developerov.

Keďže sme si ako herný engine zvolili *Unity*, nasledujúce časti tejto kapitoly budú pri svojich analýzach používať nástroje a techniky, ktoré sú v *Unity* dostupné. V prípade záujmu o viac informácií o *Unity* a nástrojoch, ktoré ponúka, je k dispozícií píloha ktorá obsahuje stručný popis *Unity*. Táto príloha je dotupná tu A.1.

3.2 Modely

Vieme, že v našej hre chceme používať modely vozidiel, tratí a poprípade aj nejaké UI modely tlačítok okien atď. Tieto modely, okrem základných UI modelov, sa v *Unity* bežne nenachádzajú a teda ich musíme do projektu importovať. Spôsob importovania modelov nie je až tak zaujímavý, no pozrieme sa na možnoti, ako môžeme modely získať a vytvárať.

3.2.1 Vytvorenie modelov

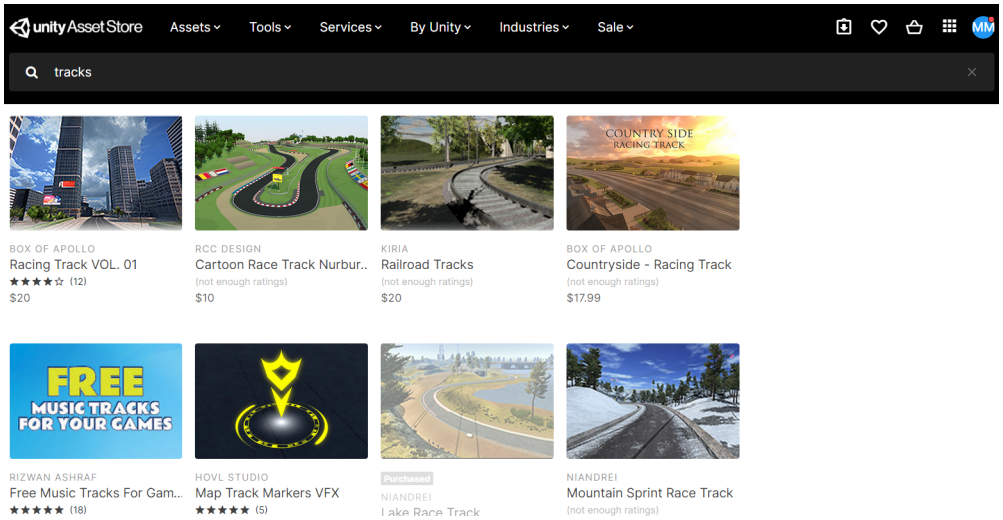
Prvá varianta získania modelov do našej hry je si jednoducho modely vyrobiť. Tento spôsob nám dá úplnú kontrolu nad vzhľadom našej hry a umožní nám plne uskutočniť náš herný návrh a dizajn. V súčastnej dobe existuje veľa nástrojov, ktoré umožňujú pomerne rýchle modelovanie akýchkoľvek objektov do hier či

iných aplikácií. Jedným z najznámejších nástrojov, ktorý sa bežne používa, je *Blender*.

Problém, ktorý v tomto prípade nastáva, je fakt, že v dobe písania tejto práce nemá autor skúsenosti z modelovaním vozidiel, závodných tratí či dokonca vlastných tlačítok a iných prvkov pre UI. Naučenie tohto procesu by nám zabralo čas, čím by sa celý proces tvory videohry skomplikoval.

3.2.2 Využitie Unity

Druhou možnosťou je použiť *Unity*, konkrétne *Unity Asset Store*, ktorý môžme vidieť na obrázku 3.1. Tento obchod vo forme webovej stránky slúži na jednoduchý nákup a import modelov, ktoré boli vyrobené pre engine *Unity*. Modely, ktoré sa tu nachádzajú, sa líšia svojou kvalitou aj cenou. Kedže sme počás návrhu našej hry spomenuli, že nepotrebujeme super detailné modely, budeme pracovať s modelmi, ktoré sú zákazníkom *Asset Storu* ponúkané zadarmo a ktoré sú kvalitatívne dostačujúce pre naše účely.



Obr. 3.1: Ukážka Unity Asset Store.

Toto riešenie nám ponúka hneď niekoľko výhod. Prvou z nich je, že nemusíme modelovať jediný model. Namiesto toho si môžeme modely kúpiť a importovať, vďaka čomu sa môžeme sto percentne sústrediť na kódovanie a implementáciu. Druhá veľká výhoda je obrovský výber z hľadiska modelov vozidiel a tratí, čo znamená, že do našej hry vďaka tomu môžeme priniesť oveľa viac herných tratí a vozidiel než keby sme tieto modely vytvárali samy.

Samozrejme, jasná nevýhoda tohto prístupu je, že môžeme použiť len modely, ktoré nám *Asset Store* poskytne. Myslíme tým, že týmto do istej miery strácame kontrolu nad vzhľadom našej hry v prospech jednoduchého získavania modelov.

Ďalšia nevýhodna *Asset Storu* sú licencie modelov, ktoré nám dovoľujú použiť modely v hre, ale nedovoľuje nám ich prenášať v projekte. Pre nás to znamená, že po vytvorení hry musíme z projektu *Unity* odstrániť všetky importované modeli, čím bude samotný projekt nedokončený.

3.2.3 Rozhodnutie

Nakoniec sme sa rozhodli ísť pohodlnejšou cestou a volíme si modely, ktoré budeme impornovať z *Asset Storu*. Tento prístup nám zaistí, aby sme nemuseli strácať čas pri modelovaní a namiesto toho sa venovať implementácií a kódovaniu. Problémy s licenciami modelov vyriešime tak, že vytvoríme návod, kde uvedieme, aké modely sme použili a ako ich treba nastaviť (návod najdeme v sekcií 5.2.4). Netvrdíme, že vytváranie vlastných modelov je strata času, ale kedže v dobe písania tejto práce autor nemá skúsenosti s modelovaním a vytváraním komplexnejších modelov, príde nám tento prístup proste jednoduchší.

3.3 Návrh AI

V návrhu našej práce sme si stanovili všetko, čo má naša AI spĺňovať. V tejto sekcií si musíme rozmyslieť, ako tieto podmienky splniť za použitia nástrojov *Unity*. Potrebujeme vyriešiť dva veľké problémy, v ktorých sú obsiahnuté všetky naše požiadavky. Tieto problémy sú:

- prejdenie trate
- detekovanie a vyhýbanie sa prekážkam

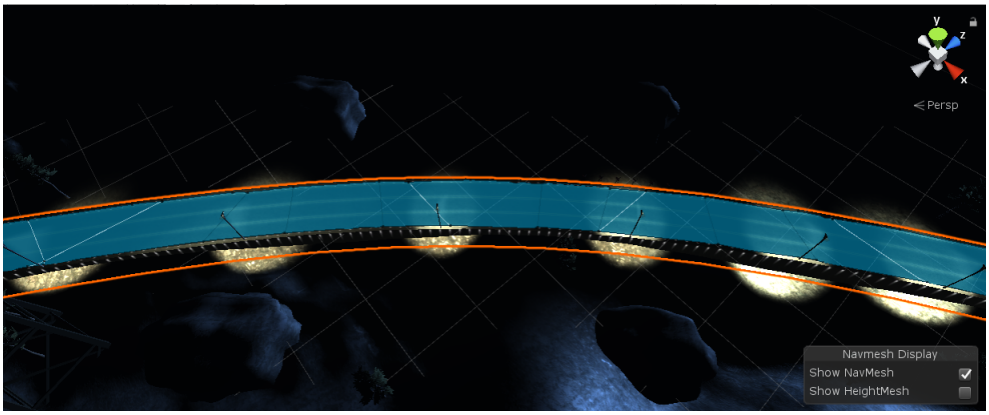
3.3.1 Prejdenie trate

Tento problém obsahuje všetky jazdné prvky vozidla ako sú zvuk vozidla, svetlá, dym a iné efekty. Pre všetky tieto prvky chceme vybudovať skripty, ktoré budú fungovať nezávisle na tom, kto vozidlo ovláda. To však samozrejme neplatí pre pohyb vozidla, ktoré potrebujeme previesť závodnou dráhou. V *Unity* existuje niekoľko spôsobou ako vozidlá v závodných hrách previesť po závodnej trati. Metódy, ktoré sa zdajú najpopulárnejšie, sú:

- NavMesh
- Waypointy

NavMesh

NavMesh je nástroj *Unity*, ktorý poskytuje jednoduché vytváranie a prácu s umelou inteligenciou. Tento nástroj umožňuje vytváranie úsekov, po ktorým sa smie umelá inteligenica pohybovať. Vytvorenie takéhoto úseku je zobrazené na obrázku 3.2. Po tomto úseku sa pohybujú NavMesh agenti, ktorý tvorí umelú inteligenciu.



Obr. 3.2: Ukážka systému NavMesh v Unity.

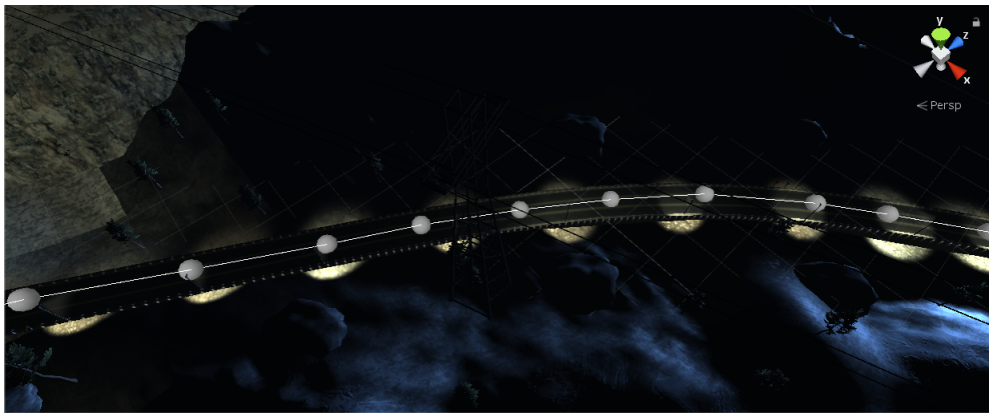
Obrovskou výhodou tohto nástroja je ľahké generovanie, nastavovanie a ovládanie agentov. Obrovskou nevýhodou, ktorú tento nástroj prináša, je fakt, že bol primárne navrhnutý pre osoby a nie pre vozidlá. Konkrétne agenti, ktoré NavMesh poskytuje, neobsahujú dostatok informácií pre správne fungovanie vozidla, čo by malo za následok neprirodzené správanie vozidla. Zaujimavú debatu o tomto probléme môžeme násť tu.

To znamená, že ak by sme tento nástroj chceli použiť, museli by sme preimplementovať správanie agentov tak, aby sa chovali ako vozidlá.

Systém waypointov

Metóda, pre ktorú sme sa rozhodli, je systém *waypointov*. Táto metóda delí cestu, po ktorej sa majú vozdilá ovládané AI pohybovať, na množinu bodov. Takéto rozdelenie môžeme pozorovať na obrázku 3.3. Umelú inteligenciu pre naše vozidlá naprogramujeme v tomto systéme tak, aby vyhľadali najbližší bod (v smere jazdy) a vydali sa k nemu. Ak tento bod dosiahnu, tak zamiera k ďalšiemu bodu. Tento systém nám umožňuje absolútnu kontrolu nad vozidlom.

Pre waypointy sme sa rozhodli hlavne pre to, že pri navMeshi by sme museli preimplementovávať definované chovania agentov, kdežto pri systéme waypointov si chovanie definujeme od základu samy.



Obr. 3.3: Ukážka systému waypointov v Unity.

Keďže sme v sekcií 3.8 spomenuli *WheelCollider*, tak pri implementácií tohto

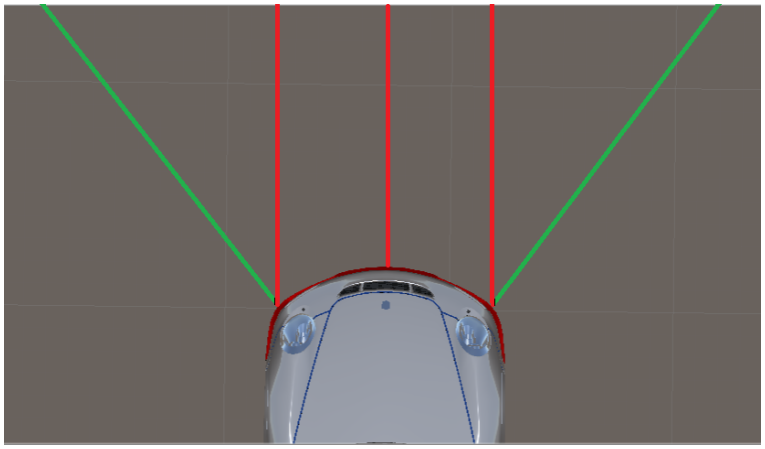
systému by sme proste tieto *Wheel Collider-e* otočili vždy smerom k najbližšiemu bodu vďaka premennej **steerAngle**. Takto by sme dosiahli, že sa vozidlo pohybuje v správnom smere. Samotný pohyb by sme rovnako, ako v prípade vozidla ovládaného hráčom, zaistili úpravou premennej **motorTorque**.

3.3.2 Detekovanie a výhybanie sa prekážkam

Riešenie, ktoré nám v tomto prípade *Unity* ponúka, sú senzory. Konkétne, statickú triedu **Physics** a v nej metódu *Raycast*, ktorá doslova vystreľuje senzor (do určitej dlžky), vďaka čomu môžeme detekovať akékoľvek prekážky na trati a podľa toho si zvoliť uhnutie alebo brzdenie vozidla. Poďme sa zamyslieť nad problémamy, ktoré sa môžu pri tejto implmentácií vyskytnúť a vďaka tomu prísť s vhodnými riešeniami.

Umiestnenie

Umiestnenie senzorov, ktoré môžeme pozorovať na obrázku 3.4, ukazujú optimálne pozície, ktoré by malo vozidlo kontrolovať. Tieto pozície nám boli doporučené komunitou vývojárov v *Unity*, ktorý majú skúsenosti s AI pre závodné hry.



Obr. 3.4: Rozmiestnenie senzorov pre vozidlá.

Dĺžka a Správanie

Čo sa týka dĺžky, tam môžeme naraziť na niekoľko problémov. Kratšie senzory sú efektívne pri malej rýchlosti, pretože v prípade detekcie bude vyhýbanie sa inému vozidlu či prekážke v poriadku, ale pri veľkej rýchlosti senzory nedokážu prekážku detekovať včas na to, aby sa jej vozidlo vyhlo bez kolízie. Z tohoto dôvodu volíme dynamickú dĺžku senzorov, ktoré sa budú meniť podľa rýchlosti vozidla. Konkrétne, pôjde o pevnú dĺžku senzorov (krátka dĺžka) s dynamickým offsetom. Čím väčšiu rýchlosť vozidlo dosiahne, tým väčší offset (vďaka čomu budú senzory dĺhšie) a naopak. Musíme si uvedomiť, že táto stratégia by sa mala vzťahovať len na predné (červené) senzory. Bočné senzory (zelené) my mali mať pevnú dĺžku, pretože od istej dĺžky by začali detekovať vozidlá, ktorým sa nie je nutné vyhnúť.

To nás privádza k správaniu, ktoré má vozidlo vykazovať na základe informácií, ktoré mu senzory poskytnú. Správanie by sa malo meniť na základe dĺžky senzoru a rýchlosti detekovaného vozidla. Toto správanie by sme mohli zhrnúť do týchto bodov:

- ak detekujeme vozidlo, ktoré je pomalšie ako my
  - ak je vozidlo ďaleko (máme dlhé senzory) → vyhni sa mu
  - ak je vozidlo blízko (máme dlhé senzory) → zabrzď a vyhni sa mu
- ak detekujeme vozidlo, ktoré je rýchlejšie ako my → nerob nič
- ak detekujeme vozidlo zelenými senzormy.
  - v prípade, že vozidlo nie je príliš blízko → nerob nič
  - inak sa mu jemne vyhni

3.4 Replay

Replay ako taký má poskytnúť hráčovi záznam o preteku, ktorý hráč práve dokončil. Aby sme hráčovi boli schopný tento požiadavok splniť, musíme získať záznam z preteku. Na získanie záznamu existuje veľa rôznych spôsobov. My sa pozrieme na dva, ktoré sa používajú v *Unity* najčastejšie.

3.4.1 Kamerový záznam

Prvý spôsob, ktorý je pravdepodobne najintuitívnejší, je obstarať si kamerový záznam celého preteku. Keďže naše vozidlo pozoruje počas celého preteku kamera, ktorá zobrazuje snímok na obrazovku, bolo by jednoduché zariadiť, aby táto kamera (alebo kombinácia iných kamier) nahrávala záznam.

Tento prístup, napriek jednoduchej implementácií, prináša niekoľko nepríjemností. Najväčšia z týchto nepríjemností je pamäťová záťaž. Ak by priemerný pretek trval 5 minút, potrebovali by sme nahrávať a dočasne skladovať 5 minútovú video nahrávku. Keďže nechceme nekvalitný záznam, toto video by zaberalo miesto v radoch niekoľko MB.

3.4.2 Pole

Ďalší možný prístup je vytvoriť si veľké pole, v ktorom budeme každý frame ukladať pozíciu a rotáciu každého vozidla v priestore. Po skončení preteku by sme navodili ilúziu videozáznamu tým, že by sme vozidlám menili pozíciu a rotáciu podľa záznamov v poli.

Takýto postup by bol zaiste náročnejší na implementáciu, ale za predpokladu, že budeme ukladať do poľa vektory pre pozície a rotácie veľkosti 12 bajtov a 16 bajtov, tak 5 minút preteku s frame ratom 60 snímkov za sekundu by zaberal približne 216kB + 288kB pamäte. To znamená, že do 1MB pamäte môžeme uložiť približne 600 sekúnd záznamu čo je 10 minút.

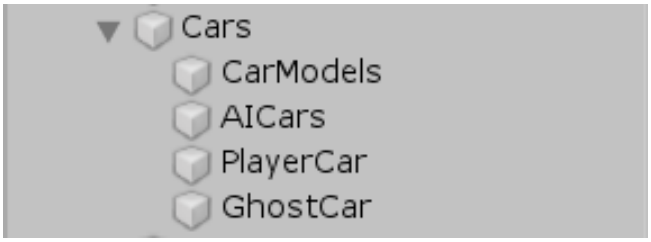
Toto je spôsob, ktorý sme sa rozhodli implementovať nie len z dôvodu pamäťovej úspory, ale aj preto, lebo sa jedná o zaujimavú implementáciu, ktorú si

chceme vyskúšať. *Unity* ponúka na ukladanie pozície premenné typu **Vector3** a na ukladanie rotácie typ **Quaternion**. Nesmieme však zabúdať na efekty ako sú predné, brzdové a cúvacie svetlá, ktoré chceme takisto počas záznamu vidieť. Tieto informácie si budeme skladovať v premenných typu **boolean**. Takto by sa nám malo podariť vytvoriť presvedčivý záznam o preteku.

3.5 Ukladanie vozidiel do tratí

Vieme, že počas preteku sa na našej trati musia vyskytovať vozidlá. V *Unity* musia byť objekty, ktoré sa nachádzajú v danej scéne, v tejto scéne uložené. Teda potrebujeme si rozmyslieť, ako budeme vozidlá v hre skladovať a poprípade ich do hry generovať.

Na obrázku 3.5 vidíme, ako by mal byť objekt, do ktorého si chceme modely vozidiel ukladať štrukturovaný. Táto štruktúra obsahuje miesto pre všetky modely vozidiel, z ktorých potom budeme vyberať vozidlá, ktoré bude ovládať AI a hráč (poprípade ghost car). My teraz potrebujeme rozhodnúť, ako sa bude naplňovať miesto s modelmi všetkých vozidiel.



Obr. 3.5: Objekt, v ktorom skladujeme a rozdeľujeme vozidlá počas preteku.

3.5.1 Drag and drop

Prvou možnosťou je samozrejme mať v každej scéne, v ktorej sa vyskytuje trať, tento objekt ručne naplnený všetkými modelmi vozidiel, ktoré by sa v preteku mohli nachádzať. Tento postup je z hľadiska implementácie triviálne jednoduchý, pretože stačí presunúť *prefabs* všetkých modelov vozidiel do daného objektu systémom *drag-and-drop*.

Z hľadiska updatov a prípadných rozšírení hry je táto možnosť veľmi neefektívna, pretože každá zmena vozidiel, či pridanie nového vozidla do hry, by viedla k tomu, že by sme tento nový či upravený model museli rozkopirovávať do všetkých našich tratí. V prípade pridania novej trate by sme túto trať museli naplniť kópiou všetkých modelov vozidiel. Takto rozširovať hru je pomerne pracné a po istej dobe iste otravné.

3.5.2 Generovanie modelov

Druhou možnosťou je pomocov skriptov generovať modely vozidiel do konkrétneho objektu. Tento spôsob by bol presný opak systému *drag-and-drop*, pretože by bol z hľadiska implementácie náročnejší, pretože by sme museli naprogramovať generátor modelov. Na druhú stranu vo chvíli, kedy by sme ho naprogramovali,

pridávanie nových vozidiel a tratí by sme nemuseli riešiť, pretože skript zaistí, že sa vozidlá budú generovať do nami udaného objektu.

Z hľadiska implementácie nám *Unity* ponúka možnosť, ako si celý proces uľahčiť. *Unity* totiž vo svojej štruktúre priečinkov obsahuje priečinok *Resources*, v ktorom môže užívateľ skladovať všetky data, ktoré sú dôležité pre správny chod hry. Tento priečinok je užitočný hlavne preto, že v *Unity* sa naň môžeme odkázať cez skript statickou triedov **Resources**. V našom prípade, ak by sme potrebovali generovať modely vozidiel do objektu nejakej trate, tak stačí, aby sme tieto modely uložili do priečinku *Resources*. Potom v skripte pomocou metódy *Resources.LoadAll* by sme načítali všetky modely vozidiel a pomocou metódy *Instantiate* by sme tieto modely generovali do daného objektu.

3.5.3 Rozhodnutie

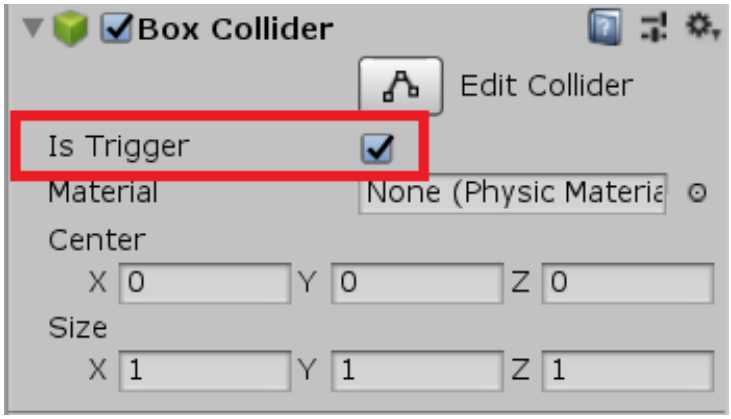
Keďže sa snažíme, aby sme pracovali čo možno najefektívnejšie, rozhodli sme sa pre generovanie modelov. Takto budeme musieť síce implementovať skriptový generátor vozidiel, no na druhú stranu nám to prinesie ľahšiu udržateľnosť a rozšíriteľnosť do budúcna.

3.6 Rozoznávanie pozicie hráča

Zisťovanie aktuálnej pozície hráča je pre nás dôležité nie len preto, aby sme hráča informovali o jeho pozícií, ale rovnako tak preto, aby sme mohli upravovať a dynamicky prispôsobovať rýchlosť vozidiel ovládaných AI (popis v sekcií 2.2). *Unity* nám s týmto problémom nemôže, bohužiaľ, nijako pomôcť v zmysle predpripravenej triedy alebo inej vychytávky, takže musíme vymyslieť, ako za použitia nástrojov, ktoré nám *Unity* poskytuje, tento problém vyriešime.

3.6.1 Trigger

Postup, ktorý nás napadol ako prvý, využíval tkz. *Trigger-y* alebo spúštače či senzory, ktoré v *Unity* obsahuje každý *Collider*. Na obrázku 3.6 môžeme vidieť ukážku tkz. *Box Collidera*. Môžeme si všimnúť, že obsahuje premennú **Is Trigger**. V prípade, že je táto hodnota nastavena na **true**, začne sa tento *Collider* správať ako senzor, ktorého správanie možno upravovať podľa metód *OnTriggerEnter, OnTriggerLeave, OnTriggerStay*.



Obr. 3.6: Zapnutie senzoru *Box Collidera*.

Náše riešenie využívalo tieto senzory, ktoré boli vždy pripevnené na každom vozidle. Po vygenerovaní vozidiel na trať by dostalo každé vozidlo poradové číslo, v ktorom bolo vygenerované. Počas preteku v prípade, že jedno vozidlo predbehne druhé vozidlo (čo zachytíme senzormy), si tieto vozidlá vymenia poradie, čím budeme mať vždy aktuálne poradie vozidiel.

Tento postup však môže zlyhať pri situáciách, keď sa predbehujú viac než len dve vozidlá. Taktiež nemôžeme dopredu vedieť, aká široká bude cesta v našich tratiach a teda nemôžeme zaistiť, že senzory budú dostatočne veľké. Nna druhú stranu sa môže stať, že budú natoľko veľké, že budú detekovať vozidlá v úplne inej časti trate.

3.6.2 Systém trojitého sita

Naša prvotná myšlienka sa s počiatku zdala správna, ale pri hlbšom zamyslení sa ukazála ako nedostačujúca. Pri hľadaní vhodného riešenia nášho problému nám veľmi pomohla komunita vývojárov a programátorv v *Unity*, ktorý nás naviedli správnym smerom. Riešenie, ktoré sme si zvolili by sa dal nazvať systémom troch sít.

Toto riešenie rozdelí trať pomocou niekoľkých *Checkpointov* (ilustrované na obrázku 3.7). Tieto *checkpointy* obsahujú rovnaké senzory ako by sme pôvodne umiestňovali na vozidlá.

Obr. 3.7: Ukážka rozdelenia trate pomocou checkpointov.

Systém trojitého sita každý frame kontroluje všetky vozidlá na trati. Konkrétne sa jedná o tri kroky, vďaka čomu vždy presne určí pozíciu hráča:

1. všetky vozidlá, ktoré sú vo vyššom kole ako je hráč, sú na lepšej pozícií
2. všetky vozidlá, ktoré prešli väčším checkpointom ako hráč, sú na lepšej pozícií
3. všetky vozidlá, ktoré sú bližšie k najbližšiemu checkpointu ako hráč, sú na lepšej pozícií

Tieto tri kroky postupne vyberajú vozidlá, ktoré sú pred nami a to práve systémom sita. Vozidla, ktoré prejdú prvým sitom prejdú do druhého síta atď. Na záver je naša pozícia rovná počtu vozidiel, ktoré sú pred nami plus my. Toto riešenie perfektne hrá do karát implementácií dynamickej AI. Vozidlá v každom site môžeme zrýchľovať/spomaľovať podľa toho, ako veľmi sú pred nami či za nami, čo v podstate znamená, čím viac je vozidlo pred nami, tým pomalšie sa bude pohybovať a naopak, čím bližšie sme pri danom vozidle resp. pred nejakým vozidlom, tým rýchlejšie sa bude vozidlo pohybovať (samozrejme v rozumnej miere).

3.7 Pauza

Pod pauzou ako takou rozumieme dočasné zastavenie nie len všetkých pohybujúcich sa objektov, ktoré sa v hre vyskytujú, ale aj všetkých meraní počas samotného preteku ako je meranie času, rýchlosti vozidla atď. Musíme zaistiť, aby sa všetky tieto entity zastavili, ale pri tom nestratili informácie, ktoré doposiaľ obsahujú. Napadli nás dva spôsoby ako by sme mohli pauzu v našej hre implementovať:

- hrubá sila
- využitie *Unity*

Pod hrubou silou si predstavujeme vytvorenie skriptu, ktorý by zastavil všetky akcie, ktoré v preteku nastávajú. Tento spôsob riešenia poskytuje množstvo problémov. Konkrétne sa tu vyskytuje problém so zastavením vozidiel – akým spôsobom ich budeme zastavovať? Ako ich zasa rozbehneme? Tieto problémy nie su síce nemožné na vyriešenie, ale určite by to nebolo najefektívnejšie z riešení.

Unity nám na druhú stranu ponúka triedu **Time**, ktorá nám do istej miery umožňuje manipuláciu s časom v hre. V našom prípade by sme využili premennú **Time.timeScale**, ktorej základná hodnota je 1. Hodnota 1 znamená, že čas v hre plynie reálne (jedna sekunda v hre je rovnako dlhá ako v reálnom svete). Ak túto premennú zmeníme na 0, potom čas v hre doslova zamrzne a všetky objekty sa zastavia. Toto je pre nás skvelé riešenie, no musíme si dávať pozor. V prípade napr. reštartu počas pauzy by sa hra mohla dostať na začiatok preteku, ale stále by bola zamrznutá. Preto musíme s týmto nástrojom pracovať opatrne.

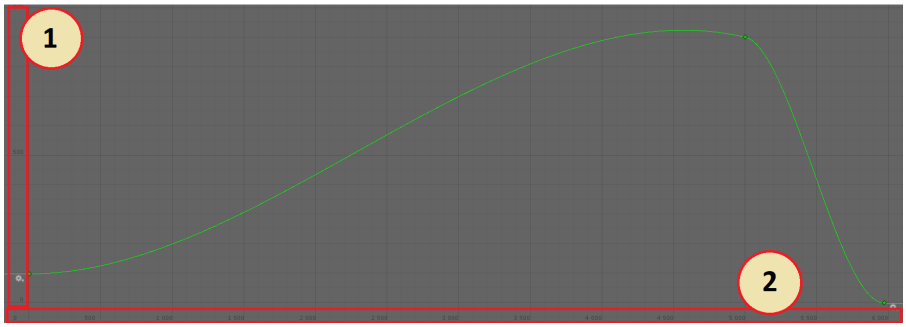
3.8 Pohyb vozidla

Pri implementácií pohybu našeho vozidla budeme používať *WheelCollider*, ktorý je v *Unity* vytvorený k simulácií kolesa. Vďaka tomuto nástroju môžeme vozidlom pohybovať, zatáčať a brzdiť. Musíme si však rozmyslieť, akou metódou sa rozhodneme vozidlom pohybovať. Konkrétne ako bude vozidlo zrýchľovať a spomaľovať.

3.8.1 Klasický spôsob vs. využitie krivky

Unity ponúka skvelý tutoriál, ako pracovať s *WheelCollider-om*, ktorý obsahuje veľa skvelých a užitočných rád, z ktorých väčšinu sme využili pri implementácií týchto *Collider-ov* do našich vozidiel. *WheelCollider* obsahuje premennú **motorTorque**, ktorý reprezentuje krútiaci moment kolesa v Newton metroch. Podľa toho, akú hodnotu tejto premennej nastavíme, podľa toho bude naše vozidlo zrýchľovať.

Klasický spôsob, ako nastavovať hodnotu, je pevne si stanoviť počet koní, ktoré chceme, aby vozidlo málo. Túto hodnotu prenasobýme hodnotou **Time.deltaTime**, ktorá reprezentuje čas od posledného framu a slúži na uhladenie prechodov medzi zmenamy rýchlosti. Kedže vozidlo má reagovať vstup hráča, celý výsledok sa bude násobiť výsledkom metódy *Input.GetAxisRaw*, ktorá sníma, či hráč vozidlo zrychľuje (vracia 1), spomaľuje (vracia -1) alebo sa nehýbe (vracia 0). Celý tento výsledok sa následne dosadí do premennej **motorTorque** vďaka čomu sa vozidlo pohybuje.



Obr. 3.8: *Animation Curve* vozidla.

Takýto prístup ale neumožňuje vozidlu, aby zrýchľoval resp. spomaľoval tak, ako reálne vozidlo. Spôsob, ktorý to zaistí, využíva *AnimationCurve*. Pod týmto pojmom si môžeme predstaviť graf, ktorého priebeh využijeme v našom prípade k simulovaniu výkonnostnej krivky motora. Na obrázku 3.8 vidíme *AnimationCurve*, ktorá takúto krivku reprezentuje. Osa **X** na tomto obrázku reprezentuje otáčky motora a osa **Y** reprezentuje krútiaci moment, ktorý chceme v daných otáčkach uplatniť v našich *WheelCollider-och*, pričom do premennej **motorTorque** budeme vkladať hodnotu krivky pri súčastných otáčkach kolies (vo *WheelCollider-y* sa tieto otáčky nachadzajú v premennej **rpm**).

Tento prístup nám umožní korektné zrýchľovanie a spomaľovanie vozidla a uľahčí nám simuláciu prevodovky, keďže nám stačí otáčky motora násobiť vhodnými konštantamy, ktoré budú reprezentovať jednotlivé prevody. Tieto konštanty by boli s každým preradením menšie a tým by vytvarali pokles otáčok, ktorý je pri prerazdzovaní do vyššieho stupňa typický.

3.9 Minimapa

Minimapu v našej hre sme si navrhli tak, aby neukazovala len hráčov na ceste, ale aby pritom dynamicky zväčšovala resp. zmenšovala plochu, ktorú môžeme vidieť v závislosti od rýchlosti hráčom ovládaného vozidla a zároveň sa dokázala otáčať podľa toho, ako hráč s vozidlom zatáča v dôsledku zmeny jeho pozície a rotácie. Existuje veľa spôsobov, ako sa dá v *Unity* minimapa implementovať. Počas úvah nad spôsobom implementácií sme narazili na problém. Čo vlastne má minimapa zobrazovať? Má sa jednať o nejaký obrázok trate na ktorom sa budú pohybovať vozidlá, alebo pôjde skôr o niečo ako real-time kamerový prenos, ktorý bude zachytávať vozidlo hráča?

3.9.1 Minimapa ako obrázok

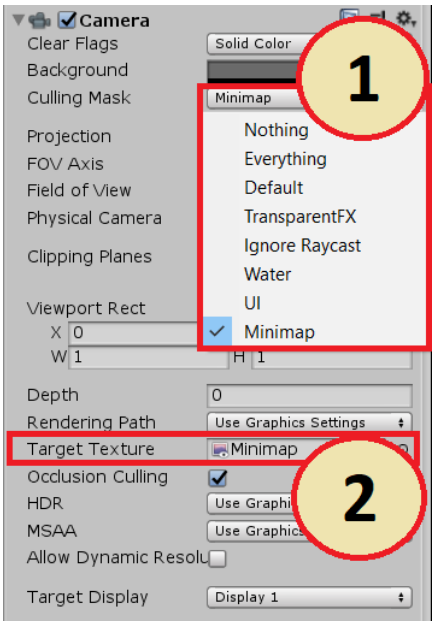
Prvý spôsob, ktorý sme spomínali vyššie, implementuje minimapu ako obrázok, na ktorom je znázornená trať, na ktorej sa práve preteká. Obrázok trate by v našom prípade musel byť upravený tak, aby spĺňal náš návrh (2.10). Na tento obrázok by sme museli zoomovať tam a späť na základe toho, ako rýchlo sa vozidlo pohybuje a nesmieme zabúdať na pohyb jednotlivích vozidiel. Aby sme boli schopný pohybovať vozidlami po našej minimape, museli by sme minimapu neustále prekresľovať aspoň raz za frame, aby pohyb na minimape nepôsobil trhavo a neplynule.

Netvrdíme, že pre správny herný žáner by tento prístup k minimape nemohol fungovať, no pre naše účely prináša toto riešenie zopár nepríjemností. Najväčšou s nich je fakt, že pre každú trať by sme museli vyrobiť obrázok trate, ktorý by presne kopíroval trať tak, aby spĺňal náš dizajn pre minimapu. Samotný dynamický zoom by mohol pôsobiť problémy hlavne v prípadoch, kedy by vozidlo drasticky spomalilo. V neposlednom rade je tu problém s neustálym otáčaním, kedže chceme, aby sa minimapa otáčala s vozidlom, museli by sme neustále centrovať daný obrázok.

3.9.2 Minimapa za použitia kamery

Vytvoriť minimapu, ktorá by reprezentovala pohľad kamery na vozidlo ovládané hráčom z hora je riešenie, vďaka ktorému sa dájú jednoducho splniť takmer všetky naše požiadavky. Otáčanie minimapy podľa zatáčania hráča by sme vyriešili tak, že by kamera kopírovala rotáciu osí **Y** a **Z** hráčom ovládaneho vozidla čím by vytvorila požadovany efekt. Samotný dynamický zoom by sa dal vyriešiť pripočítavaním offsetu k **Y** súradnice pozície kamery, ktorý by sa menil podľa rýchlosti vozidla. No nastáva tu problém: ak kamera sleduje hráča zhora, tak ako dosiahneme, aby výzor našej minimapy vyzer tak ako sme popísali? Ako vôbec môžeme zaistiť, aby sa v našej minimape vykresľovala kamera?

Na riešenie tohoto problému musíme použiť *textúry*. Textúra je v *Unity* bitmapa, ktorá sa umiestňuje na nejaký povrch. V našom prípade použijeme textúru, ktorá zobrazuje, na čo sa aktuálne pozerá kamera. Túto textúru aplikujeme na *panel*, ktorý bude reprezentovať nášu minimapu. Obrázok 3.9 ukazuje, ako sa v *Unity* pripevňuje textúra na nejakú kameru.



Obr. 3.9: Nastavenie viditeľných vrstiev a pridanie textúry.

Posledný problém, ktorý potrebujeme vyriešiť, je, aby zobrazovanie vozidiel a trate na minimape odpovedalo nášmu návrhu. Riešenie, ktoré sme zvolili, sa opiera o nástroj *Unity* zvaný *Layers* resp. vrstvy. Na obrázku 3.9 si môžeme všimnúť, že môžeme kamere povedať, ktoré vrstvy má snímať a ktoré ignorovať

(na obrázku sme si zvolili, aby snímala iba vrstvu *Minimap*). Vďaka tomu stačí, aby sme vozidlá a trať zaradili do inej vrstvy než je ta, ktorú má snímať kamera minimapy.

Na to, aby sme získali požadovaný dizajn vozidiel, doslova stačí pripevniť ku každému vozidlu šípku, ktorá by mala reprezentovať vozidlo v minimape a nastaviť jeho vrstvu na Minimap. Tým sa začne vykresľovať na minimape. Je dôležité, aby všetky ostatné kamery mali vrstvu Minimap vypnutú, inak by hráč počas preteku videl na vozidlách šípky.

Nakoniec k dosiahnutiu požadovaného vzhľadu trate sme použili *waypointy*, ktoré používa AI. Tieto *waypointy* kopírujú trať, čo je perfektné pre naše účely. Na to aby sme vedeli cestu, ktorú tieto *waypointy* vytvárajú, použijeme triedu *Unity*, ktorá sa volá **LineRendered**, vďaka nej môžeme vyrenderovať cestu pomocov bodov, ktoré sme na trati rozmiestnili. Túto cestu samozrejme zaradíme do vrstvy Minimap a tým by sme mali získať minimapu, ktorá spĺňa všetky naše požiadavky z hľadiska dizajnu a funkčnosti.

3.10 Vstup hráča

V sekcií 2.8.2 sme si ukázali návrh panelu, ktorý má hráča informovať o ovládaní hry. Na toto ovládanie využijeme **Input** *Unity*, vďaka čomu zaistíme, že užívateľ v prípade potreby, môže meniť ovládanie hry. Toto ovládanie bude schopný meniť v konfiguračnom okne hry, ktoré *Unity* po vybudovaní hry hráčovi poskytuje (prítomnosť okna je po vybudovaní nepovinná). Problém, ktorý v tomto prípade nastáva, je, že v prípade zemny vstupu hráčom sa tieto zmeny neprejavia na paneli, ktorý má hráča informovať o ovládaní hry, pretože *Unity* neumožňuje čítať tieto hodnoty za runtime-u.

Rozhodli sme sa, konfiguračné okno v hre ponechať a namiesto toho hráča informovať, že prípadná zmena vstupu nebude zaznamenaná v hernom paneli. Pre toto riešenie sme sa rozhodli preto, že chceme hráčovi umožniť prispôsobyť si ovládanie hry tak, aby mu to čo najviac vyhovovalo.

4. Programátorská dokumentácia

Táto kapitola má poskytnúť prehľad o tom, ako celá hra funguje a aké mechanizmy, scény, objekty a skripty obsahuje. Popíšeme si, ako tieto prvky komunikujú a akú prácu vykonávajú. V prípade skriptov poskytneme ich funkciu a dôležité metódy. Každý skript obsahuje dokumentačné komentáre ku každej metóde a triede, ktorú obsahuje. Preto nebudeme pri opise skriptov zachádzať v tejto kapitole do príliš veľkých detailov.

V priloženej elektronickej prílohe sa nachádza celý projekt vytvorený v *Unity 2019.1.0f2 Personal* v zložke **BakalarskaPraca_RaceGame**. Všetky dôležité prvky tohto projektu je možné nájsť v zložke *Assets*. Táto zložka obsahuje:

- *Resources* – obsahuje všetky použité textúry, dáta, animácie a obrázky použité v našej hre.
- *Scenes* – obsahuje scény, ktoré sú v našej hre použité. Tieto scény sú delené do scén obsahujúcich menu a scén obsahujúcich trate.
- *Skripts* – zložka, v ktorej sa nachádzajú všetky skripty, ktoré sme v tomto projekte použili.

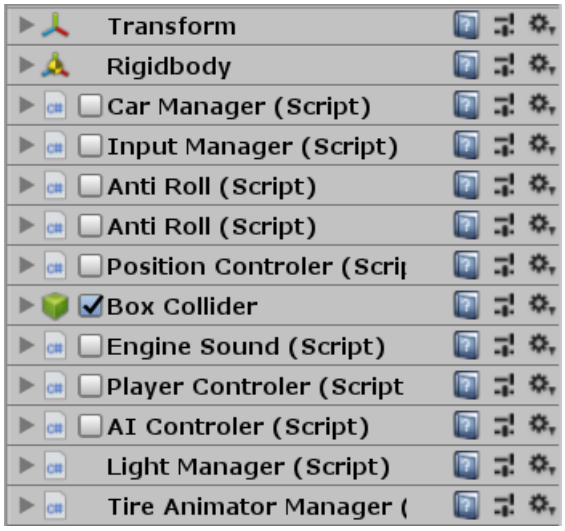
Všetky obrázky, ktoré sa v hre nachádzajú sme získali zo stránok *pixabay.com*, *pexels.com* a *openclipart.org*. Niektoré tieto obrázky boli pre účeli hry upravené.

Z licenčných dôvodov sme všetky assety, ktoré sme stiahli z *Unity Asset Storu* nemoli ponechať v projekte a museli sme ich vymazať. Teda projekt neobsahuje žiadne vozidlá, trate, hudbu a štýly panelov a fondov. Informácie o tom, ako v prípade záujmu projekt obnoviť do pôvodného stavu, je možné nájsť v sekcií 5.2.4.

4.1 Car

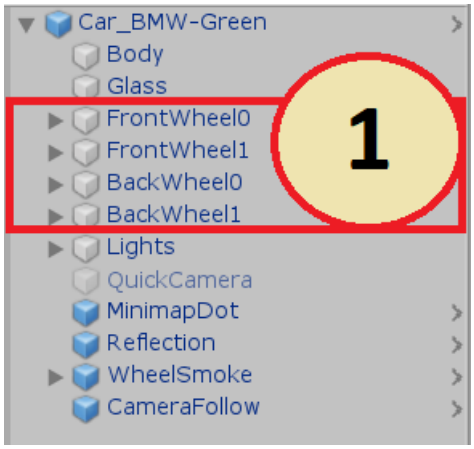
Každé vozidlo v našej hre pozostáva z niekoľkých skriptov a objektov. Na obrázku 4.1 vidíme zloženie vozidla. Každé vozidlo obsahuje sadu skriptov, o ktorých sa detailnejšie porozprávame od sekcie 4.1.3. Okrem toho každé vozidlo obsahuje nejaký typ **Collider-a** - v tomto prípade **BoxCollider**. Tento prvok zaisťuje, že vozidlo bude detekované senzormi, triggermi a že počas kolízie bude vozidlo pôsobiť ako pevné teleso.

Ďalší prvok, ktorý musí obsahovať každé vozidlo, je **Rigidbody**, vďaka ktorému získa vozidlo trenie, váhu a pôsobí naň fyzika *Unity*.



Obr. 4.1: Obsah každého vozidla z pohľadu inšpektora *Unity*.

To všetko samozrejme nestačí k tomu, aby bolo vozidlo v našej hre funkčné. Na obrázku 4.2 vidíme, aké objekty má vozidlo obsahovať.



Obr. 4.2: Objektový obsah vozidla v našej hre.

V červenom pásme máme označenú štvoricu objektov, ktoré reprezentujú kolesá vozidla. Každé z týchto objektov obsahuje **WheelCollider**.

Ďalej vozidlo obsahuje:

- *Lights* – objekt obsahujúci predné, brzdové a cúvacie svetlá vozidla. Svetlá sú reprezentované triedou *Unity* **Light**. Konkrétne sa jedná o **Spotlight**.
- *QuickCamera* – **camera**, ktorá slúži ako quick kamera pre dané vozidlo.
- *MinimapDot* – objekt reprezentujúci vozidlo na minimape. Tento objekt má vrstvu nastavenú na *Minimap*.
- *Reflection* – objekt obsahujúci **Reflection Probe**, vďaka ktorému pôsobia vozidlá reálnejšie. Ide o čisto kozmetický prvok.

- *WheelSmoke* – objekty, ktoré zodpovedajú za generovanie šmyku a dymu počas driftu. Tieto objekty obsahujú **Particle System**, ku ktorému je pripevnená textúra reprezentujúca dym.
- *CameraFollow* – prázdny objekt, ktorého poloha odpovedá defaultnej polohe kamery počas preteku.

4.1.1 Interfaces and Abstract Classes

V tejto sekcií si popíšeme všetky zohrania a abstraktné triedy, ktoré v tomto projekte používame. Tieto rozhrania a triedy sme zaviedli pre prípad možného rozšírenia tejto práce. Chceme poskytnúť možnosť meniť a vylepšovať ovládače bez toho, aby sme museli meniť **CarManager** vozidla, ktorý riadi celé vozidlo.

Controler

Táto abstraktná metóda má jedinú metódu, ktorou je *Respawn*. Metóda ma za úlohu respawnovať vozidlo v prípade, že sa ocitne v kolízií alebo mimo trate alebo ho užívateľ spustí manuálne. Túto triedu musia implementovať všetky ovládače vozidla, či už sa jedná o ovládače pre hráča alebo AI.

AbstractPlayerControler

Ďalšia abstraktná trieda, ktorá ma za úlohu poskytnúť metódy na ovládanie vozidla, ktoré riadi hráč. Táto trieda implementuje abstraktnú triedu **Controler**. Okrem toho obsahuje metódu *Accelerate* na zrýchľovanie/cúvanie vozidla, ktorá prijíma informácie o rýchlosti vozidla. Ďalšia metóda *Steer* má za úlohu zmeniť smer jazdy vozidla na základe daného uhla, o ktorý sa má vozidlo vychýliť. Poslednou metódou v tejto triede je metóda *Brake*, ktorou úloha je ubrzdiť vozidlo.

AbstractAIControler

Posledná abstraktná trieda, ktorá implementuje triedu **Controler**, popisuje metódy ovládača vozidla pre AI. Trieda obsahuje metódy na zrýchlenie vozidla (*Accelerate*) a metódu na manévrovanie vozidla (*Steer*). Naviac obsahuje metódu *ObsticaleAvoidence*, ktorá slúži na detekciu a vyhnutie sa akýmkoľvek prekážkam.

ILightControler

Rozhranie na kontrolu svetiel. Obsahuje základné metódy na manipuláciu so svetlami vozidla. Všetky tieto metódy pracujú s objektom **Transform**, ktorý má reprezentovať objekt, v ktorom sú umiestnené svetlá vozidla. Rozhranie obsahuje metódy na zapínanie (*BrakeLightON*) a vypínanie (*BrakeLightsOFF*) brzdových svetiel, zapínanie (*ReverseLightON*) a vypínanie (*ReverseLightsOFF*) cúvacích svetiel a metódu na ovládanie predných svetiel (*FrontLights*).

ITireAnimatorControler

Toto rozhranie popisuje základne metódy pre animáciu kolies vozidla. Tieto metódy majú byť implementované pre animáciu rotácie kolies (*RotateTires*) a bočenie kolies (*SteerTires*). Obe tieto metódy pracujú s *WheelColliderm-y*, ktoré reprezentujú skutočné kolesá vozidla, vďaka ktorým sa pohybuje, a **Transform** kolesá, ktoré reprezentujú objekty kolies, ktoré sú v hre viditeľné.

4.1.2 Controlers

V tejto časti prejdeme všetky ovládače vozidla. Pod ovládačmi chápeme skripty, ktoré ovládajú časti vozidla alebo samotné vozidlo.

PlayerControler

Tento skript slúži na ovládanie vozidla v prípade, keď ho ovláda hráč. Trieda skriptu je odvodená od abstraktnej triedy **AbstractPlayerControler**. V tomto skripte sa v metóde *Start* získa prístup k waypointom danej trate, ktoré budeme potrebovať na respawn. Okrem toho implementuje všetky metódy abstraktnej triedy **AbstractPlayerControler** a dva pomocné metódy na nájdenie a získanie waypointov: (*FindPath*) – voláme v metóde *Start*, a metódu *GetClosestPathPoint* – vracia najbližší waypoint, pri ktorom sa vozidlo nachádza, vo forme *Transform*. Oba tieto metódy slúžia ako pomocné metódy k metóde *Respaw)*.

AIControler

Skript zodpovedá za pohyb vozidla, ktoré je ovládané AI. Tento skript obsahuje triedu, ktorá dedí rozhranie od abstraktnej triedy **AbstractAIControler**. V metóde *Start*, rovnako ako v prípade **PlayerControleru**, získame prístup k waypointom nie len kvôli respawnu, ale aj kvôli tomu, aby bolo vozidlo schopné prejsť trať. K získaniu waypointov slúžia dva pomocné metódy: *FindParen)* – na nájdenie umiestnenia waypointon (vráti nám pozíciu objektu s waypointamy vo forme **Transform**), a metódu *GetPath* – slúži na vybratie waypointov v danom objekte. Potom tento skript implementuje rozhranie abstraktnej triedy **AbstractAIControler**. Skript je doplnený o metódu *AvoidSteer*, ktorá slúži ako pomocná metóda k metóde *ObsticaleAvoidance* a zodpovedá za otočenie *WheelCollider-ov* vozidla, ktoré reprezentujú predné kolesá, v uhle a smere, v ktorom sa vozidlo vyhýba prekážke.

4.1.3 Managers

CarManager

Tento manažér zodpovedá za správne fungovanie celého vozidla. Je vytvorený tak, aby bol nezávislý na ovládačoch a manažéroch, ktoré implementujú nejaké rozhrania či abstraktné triedy.

Metóda *Start* slúži na inicializáciu všetkých ovládačov a manažérov a na získanie všetkých podstatných objektov vozidla. Na to využíva pomocné metódy:

- *SetWheelMeshesAndColliders* – metóda na získanie *WheelCollider-ov* spolu s objektmi, ktoré reprezentujú pneumatiky. Používa pri tom metódy na získanie *WheelCollider-ov* (*GetWheelColliders*) a získanie objektov kolies (*GetWheelMeshes*).

- *SetControlersAndManagers* – metóda, ktorá slúži na získanie správnych ovládačov a manažerov ako sú **InputManager, ILightManager, ITireAnimatorManager** a **AbstractPlayerControler/AbstractAIControler** – záleží na tom, kto vozidlo ovláda. Vidíme, že metóda nehľadá konkrétne ovládače a manažére (až na **InputManager**), ale namiesto toho hľadá triedy, ktoré implementujú dané rozhrania, resp. abstraktné triedy, vďaka čomu je možné kedykoľvek si vytvoriť nové ovládače či manažére.

- *SetCarValues* – metóda, ktorá načíta parametre vozidla ako sú maximálna rýchlosť, počet koní, sila brzdenia a maximálny uhol otáčania kolies.

- *AddPlayerUpgradeValues* – je metóda, ktorá v prípade vozidla ovládaného hráčom pridá k parametrom vozidla všetky upgrady, ktoré si hráč zakúpil.

- *SetUpEngineCurve* – nastavuje maximum krivky **AnimationCurve**, ktorá simuluje výkon motora.

- *SetUpGhostCar* – sa používa v prípade, že je vozidlo použité ako ghost car – vypne možnosť efektu šmyku.

Ďalej tu máme metódu *Update*, ktorá zodpovedá za efekty vozidla. Na to neustále volá metódy:

- *TiresAnimation* – metóda, ktorá zodpovedá za animáciu kolies. Využíva pri tom metódy rozhrania **ITireAnimationManager**.

- *LightsCheck* – zodpovedá za korektné fungovanie predných, brzdových a cúvacích svetiel, ktoré vozidlo používa. Na základe hráčskeho vstupu využíva metódy **ILightManager**.

- *QuickCameraCheck* – metóda, ktorá kontroluje, či hráč chce zapnúť alebo vypnúť quick kameru.

- *SteeringWheelAnimation* – metóda, ktorá animuje volant vozidla. Táto metóda je nepovinná, pretože vozidlo nemusí nutne obsahovať volant.

- *GearShift* – mení aktuálny zaradený rýchlostný stupeň na základe otáčok a rýchlosti, ktorou sa vozidlo pohybuje.

Posledná hlavná metóda je *FixedUpdate*, ktorá zodpovedá za fyziku vozidla. Táto metóda prepočítava aktuálny výkon motora pomocou metódy *EnginePower*, ktorá neustále vyhodnocuje **AnimationCurve** na základe otáčok, ktoré dosahujeme. Tieto otáčky sa prepočítavajú v metóde *WheelRPM*, ktorá priemeruje otáčky kolies. Okrem tejto metódy volá metóda *FixedUpdate* aj metódy na ovládanie vozidla podľa toho, či vozidlo ovláda hráč - volá všetky verejné metódy **AbstractPlayerControler** a metódu na kontrolu a uvedenie vozidla do

šmyku (*AdjustTraction*), alebo vozidlo ovláda AI – volá všetky verejné metódy **AbstractAIControler**.

Záverom tento manažér obsahuje metódu *GetWheelsThatShouldHaveTorgue*, ktorú využíva ovládač AI na získanie kolies, ktoré majú pohon (predné, zadné, všetky), metódu na získanie predných kolies *GetSteerinfWheels*, ktorú volá AI na získanie kolies, ktoré na vozidle zatáčajú, a metódu *SetMaxSpeedQuotient*, ktorú používa **PositionManager** na úpravu rýchlosti vozidiel ovládaných AI.

TireAnimationManager

Táto trieda zodpovedá za korektné animácie kolies. Konkrétne obsahuje metódu *RotateTires*, ktorá zodpovedá za korektnú rotáciu objektov, ktoré reprezentujú kolesá. Ďalej obsahuje metódu *SteerTires*, ktorá má objekty reprezentujúce kolesá otáčať v rovnakom uhle v akom sa otáčajú *WheelCollider-e* vozidla, a metódu *PositionTires*, ktorá zodpovedá za správnu pozíciu objektov reprezentujúcich kolesá. Správnou pozíciou myslíme prípady, kedy vozidlo nadíde na jamu alebo výmoľ, či skončí nejakým kolesom vo vzduchu. Pretože *WheelCollider-y* simulujú perovanie kolies, v závislosti na týchto situáciách *WheelCollider-y* perujú hore dole. Táto metóda teda mapuje objekty pneumatík na tieto *WheelCollider-e*.

LightManager

Ďalší manažér zodpovedá za správne zapínanie a vypínanie všetkých druhov svetiel, ktoré vozidlo obsahuje. Tento skript obsahuje metódy na zapínanie brzdových (*BrakeLightsON*) a cúvacích svetiel (*ReverseLightsON*). Rovnako tak obsahuje metódy na vypínanie brzdových *BrakeLightsOFF* a cúvacích svetiel (*ReverseLightsOFF*). Záverom obsahuje metódu *FrontLight*, pri ktorej zavolaní sa otočí boolovská hodnota aktivácie predných svetiel, tj. ak boli svetlá aktivované (hodnota **true**), tak po zavolaní metódy sa svetlá deaktivujú (hodnota **false**).

InputManager

Táto trieda slúži na kontrolu, či bolo na klávesnici stlačený kláves, ktorý nejako ovláda vozidlo či iný prvok. Táto trieda obsahuje len metódu *Update*, ktorá neustále kontroluje, či nebola stlačená klávesa na rozsvietenie predných svetiel, pauzy, zrýchlenia vozidla, brzdy, bočenia, ručnej brzdy, quick kamery alebo respawnu. Všetky tieto informácie sú dostupné ostatným triedam len na čítanie, okrem informácie o brzdení, pretože tú potrebuje prepisovať **AIControler** v prípade, že AI vozidlo brzdí.

CheckpointManager

Posledný manažér vozidla zodpovedá za korektne značenie prejdených checkpointov, ktoré sa na trati vyskytujú. V metóde *Start* získame všetky checkpointy, ktoré sa na trati nachádzajú, za použitia pomocnej metódy *FindCheckpoints*, ktorá nájde a získa tieto chceckpointy. Potom v metóde *Update* neustále kontrolujeme, či sme neprešli nejaký checkpoint pomocou volania metódy *CheckDistance*, ktorá zisťuje vzdialenosť od najbližšieho checkpointu s vyšším ID než sme naposledy prešli. Ak prejdeme ďalší checkpoint, tak si ID zvýšime

(**checkpointPassedID**). V prípade, ak prejdeme všetky checkpointy, ID vynulujeme a zvýšime počet kôl (**lab**), ktoré sme prešli, o 1.

4.1.4 Skidding

Skript je zodpovedný za vytváranie zvukového efektu šmyku a stopy po šmyku. Počas metódy *FixedUpdate* dáva pozor, či nie je vozidlo v šmyku. Ak sa vozidlo dostane do šmyku, tak vytvorí zvuk šmyku (*SkidSound*), stopu po šmyku (*SkidMark*) a podľa toho, či so šmykom začíname alebo končíme, buď spustí dym *StartSmoke* alebo ho vypne *StopSmoke*. Tento skript je pripevnený na každom kolese vozidla.

4.1.5 DestroyTimer

Trieda s jedinou metódou *Update*, ktorá počká 2 sekundy a potom zničí objekt, na ktorom je táto metóda umiestnená. Tento skript používame na inštanciách zvuku, šmyku a rovnako tak inštanciách objektov stôp šmyku, ktoré sa neustále vytvárajú počas driftov a brzdení. Ak by sme ich neničili, po čase by mohol program spotrebovať vďaka hromadeniu týchto objektov veľa pamäte.

4.1.6 AntiRoll

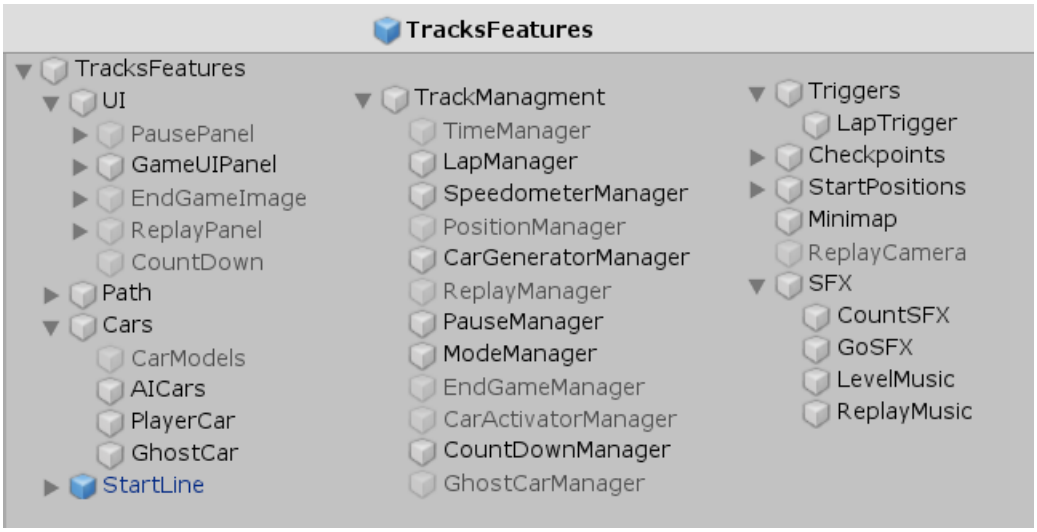
Táto trieda slúži ako osa na dvojici kolies. Tieto osy musia byť vždy dve - predná s zadná osa. Teda každé vozidlo obsahuje dve inštancie tohto skriptu pre predné a zadné kolesá. Tento skript v metóde *Start* získa **Rigidbody** vozidla a v metóde *FixedUpdate* neustále vyvažuje vozidlo kontrolou kolies. Ak sa jedno koleso dvíha, tak naň aplikuje silu a tým sa vozidlo vyrovná. Skript bráni vozidlu v tom, aby sa za normálnych okolností pretočilo.

4.1.7 EngineSound

Posledný skript vozidla slúži na prehrávanie zvukov, ktoré vydáva motor. Tento skript je prevzatý z **Standard Assets**, ktoré ponúka *Unity*.

4.2 Trať

V tejto sekcií sa pozrieme na objekty a skripty, ktoré sú obsiahnuté v každej trati. V našej hre sa nachádzajú 4 rôzne trate a každá z nich je reprezentovaná jednou scénou. Na obrázku 4.3 môžeme vidieť obsah objektu **TrackFeatures**, ktorý obsahuje všetky objekty, ktoré sa majú na trati vyskytovať.



Obr. 4.3: Povinný objektový obsah každej trate v našej hre.

Konkrétne sa v objekte vyskytuje:

- *UI* – panel, ktorý obsahuje všetky UI prvky vyskytujúce sa v preteku.
- *Path* – objekt, obsahujúci všetky waypointy.
- *Cars* – objekt, ktorý obsahuje všetky modely vozidiel (*CarModel*), objekt, v ktorom budú vozidlá ovládané AI (*AICars*), objekt, v ktorom bude vozidlo ovládané hráčom (*PlayerCar*) a ghost car (*GhostCar*).
- StartLine – objekt, ktorý reprezentuje štartovaciu značku.
- TrackManager – obsahuje všetky manažére, ktoré ovládajú danú trať.
- Triggers – obsahuje trigger, ktorý má byť umiestnený na rovnakej pozícií ako *StartLine*.
- *Checkpoints* – obsahuje všetky checkpointy, ktoré potrebujeme na korektné meranie pozícií vozidla.
- StartPositions – obsahuje objekty, ktoré označujú štartovacie pozície pre vozidlá.
- Minimap – kamera, ktorá zachytáva hráčovo vozidlo zhora. Jej obraz slúži ako minikamera.
- ReplayCamera – kamera, ktorá prehráva replay preteku.
- SFX – objekty zvukov, ktoré počas preteku hrajú.
  - *CountSFX* – zvuk odpočtu preteku.
  - *GoSFX* – zvuk po odpočte - značí štart.
  - *LevelMusic* – zvuk, ktorý sa prehráva počas preteku.
  - *ReplayMusic* – zvuk, ktorý sa prehráva počas replay.

4.2.1 Managers

V tejto sekcii si predstavíme manažérov, ktorí ovládajú, kontrolujú a poprípade nahrávajú všetko, čo sa na trati deje.

CarActivatorManager

Táto trieda slúži na manipuláciu s vozidlami pred a po preteku. Obsahuje metódu na ubrzdenie všetkých vozidiel (*BrakeAllCars*), ktorú používame na začiatku preteku, aby sa vozidla nepohybovali (môže sa stať, ak pretek začína na nejakom kopci). Ďalej trieda obsahuje metódu na aktivovanie všetkých vozidiel (*StartCars*), ktorá po úvodnom odpočte spustí všetkým vozidlám skript **CarManager**. Nakoniec táto trieda obsahuje metódu na vypnutie všetkých zvukov, ktoré vozidlo vydáva (*MuteCars*). Zvuky vypíname na konci preteku, pretože prechádzame do popretekového panelu a nechceme užívateľa rozptyľovať týmito zvukmi.

CarGeneratorManager

Keďže sme sa rozhodli do našich tratí vozidla generovať, potepujeme skript, ktorý bude tieto vygenerované vozidlá rozdeľovať a prispôsobovať pre pretek. Túto úlohu vykonáva práve táto trieda. Všetky tieto úlohy sú vykonávané v metóde *Start*, ktorá volá pomocné metódy. Okrem toho metóda *Start* aktivuje skripty **SpeedometerManager** a **PositionManager**, nakoľko tieto skripty musia byť aktivované až v momente, kedy sú vygenerované vozidla.

Prvou z nich je metóda, ktorá generuje vozidlá pre AI *GenerateAICars*. Toto generovanie je realizované tak, že sa prekopírujú vozidla, ktoré sa volia náhodne z objektu **CarModels** (obsahuje všetky odomknuté vozidlá), do objektu **AICars**. Následne je nutné tieto vozidlá upraviť. K tomu sa použije metóda *SetAICarValues*, ktorá vozidlám nastaví **Tag** na hodnotu **AI**, zapne im všetky riadiace skripty (okrem **CarManager**), nastaví ovládanie na AI pomocou premennej **TypeOfDriver** v triede **CarManager** a nakoniec nastaví farbu **MinimapDot** na modrú, čo je farba AI vozidiel na minimape.

Pre vygenerovanie vozidla pre hráča slúži metóda *GeneratePlayerCar*, ktorá nájde hráčom zvolené vozidlo v katalógu vozidiel, ktoré sú v objekte **CarModels**, pomocou metódy *FindPlayerCar*. Hráčovo vozidlo je generované do objektu **PlayerCar**. Metóda po vygenerovaní vozidla spustí skript **MinimapManger**, ktorý sa musí aktivovať až po vygenerovaní hráčovho vozidla. Parametre hráčovho vozidla sa nastavia pomocou metódy *SetPlayerCarValues*, ktorá nastaví všetko tak, ako to bolo v prípade AI. Rozdiel je v hodnote **Tagu** – nastavená na **Player**, ďalej v hodnote **TypeOfDriver** – nastavená na **PLAYER**, a farbe **MinimapDot**, ktorá je pre hráča červená.

CountDownManager

Táto trieda má za úlohu odpočítať štart preteku. Počas tohto odpočtu táto trieda spúšťa niekoľko skriptov, ktoré nemôžu byť aktivované skôr. Celý odpočet riadime pomocou systému koruntín, ktoré sa spustia v metóde *Start*. V tejto koruntíne sa postupne odpočítajú tri sekundy, v ktorých sa odanimuje odpočet na

obrazovke, spustí sa **CarActivatorManager**, **SpeedometerManager**, **TimeManager** a aktivuje sa UI celého preteku. Všetky tieto aktivácie sa spúšťajú na konci odpočtu. Počas celého tohto odpočtu metóda *Update* brzdí vozidlá pomocou metódy *BrakeAllCars*, ktorá je obsiahnutá v triede **CarActivatorManager** 4.2.1.

EndGameManager

Manažér sa stará o korektné ukončenie preteku a manipuláciu s popretekovým panelom **EndGamePanel**. Na začiatku v metóde *Start* spustí koruntínu *EndGameSequence*, ktorá vykoná animáciu popretekového panelu, a uvedie naň korektné informácie o preteku vďaka metóde *SetEndGamePanelTexts*. Táto metóda vypíše čas, umiestnenie alebo poradie eliminácie v závislosti od módu (*SetTimeOrPlaceOrElimination*). Ďalej vypíše body, ktoré hráč na základe tohto výsledku získal (*SetPoints*), pochvalný alebo posmešný text na základe výsledku (*SetFinnishText*) a nakoniec vypne zvuky vozidiel (*MuteCarAudio*).

Ďalej tu máme metódy, ktoré ovládajú tlačidlá panelu **ReplayPanel**. Tieto metódy sú *BackToMenu*, ktorá skontroluje, či hráč nezískal traťový (*CheckForTrackRecord*) alebo osobný (*CheckForPersonalRecord*) rekord – pridá mu body na jeho účet a vráti sa do scény **ModeSelection** 4.4.4. Ďalej obsahuje metódu *Back*, ktorá vráti hráča z **ReplayPanel-u** späť do **EndGamePanel-u**. Okrem toho tu máme **EventTrigger** metódy pre vyskakovacie info okná pre jednotlivé tlačidlá.

GhostCarManager

Tento skript obsahuje okrem manažéra aj dve serializovateľné dátové štruktúry **Node** – jeden záznam ghost car, a **GhostData** – list týchto záznamov, spolu s informáciami o vozidle, čase a mene hráča (pretože len ten, kto vytvorí rekord trate, má ghost car).

Samotný manažér sa stará o korektne vytváranie, riadenie a nahrávanie ghost car. Na začiatku v metóde *Start* zistí, či existuje záznam o nejakom ghost car pomocou metódy *LoadIfPresent*. Ak tento záznam existuje, tak si ho nahráme. Tieto dáta boli uložené pomocou metódy *Serialize* triedy **BinaryFormater**, vďaka čomu načítanie týchto dát prebieha pomocou deserializácie (*Deserialize*). V prípade, že existujú záznamy o ghost car, tak vozidlo vygenerujeme a upravíme (*InitializeGhostCar*). Konkrétne mu vypneme **Collider**, aby sa cez neho dalo prechádzať, vypneme mu všetky **AudioSource** objekty a všetky skripty. Nastavíme mu **Tag** na hodnotu **Ghost** a farbu objektu **MinimapDot** na fialovú. Po pripravení a vygenerovaní ghost car začneme so simulovaním jazdy a zároveň s nahrávaním jazdy súčasného hráča (*PlayAndRecordGhost*).

Táto metóda nahráva záznam a simuluje ghost car pomocou metódy koruntín. Túto koruntínu spúšťa pomocou metódy *PlayAndRecord*, ktorá v každom kroku simulácie zároveň nahráva pohyb hráčovho vozidla pomocou metódy *RecordGhost*. V prípade, že hráč vytvoril nový časový rekord, tak sa záznam o jeho hre uloží (*Save*).

LapManager

Táto trieda má za úlohu obnovovať aktuálne kolo preteku. To sa vykonáva v metóde *Update*, kde sa neustále vypisuje hodnotu statickej premennej **currentLap**, ktorá je obsiahnutá v triede **LapTrigger**.

MinimapManager

Tento manažér má za úlohu správne fungovanie minimapy. V metóde *OnEnable* sa pomocou triedy **LineRendered** vykreslí trať do minimapy pomocou waypointov. Samotné sledovanie vozidla hráča je obstarané v metóde *Update*, ktorá pomocou rýchlosti vozidla mení offset, ktorý sa pripočítava k y-novej súradnici kamery, vďaka čomu dosahujeme dynamické priblíženie alebo oddaľovanie kamery od vozidla hráča.

ModeManager

Ďalšou triedou, ktorá je obsiahnutá v každej trati, je trieda **ModeManager**. Trieda má na starosti korektný výzor UI v závislosti na vybranom móde. Konkrétne v metóde *Start* kontroluje, či si hráč nezvolil **TimeMode**. Ak áno, tak vypne panel zobrazujúci poradie hráčov, ktorý v tomto móde nie je nutný.

PauseManager

Tento manažér, má na starosti koreknté zobrazenie**PausePanel-u**. Tento manažér v metóde *Update* kontroluje, či bola užívateľom aktivovaná pauza (stlačením *ESC*). Ak áno, tak zavolá metódu *Pause*, ktorá vypne všetky UI prvky na obrazovke, zobrazí **PausePanel** a vypne zvuky všetkých vozidiel. V prípade vypnutia pauzy sa zavolá metóda *Resume*, ktorá všetky operácie metódy *Pause* zvráti na pôvodnú hodnotu.

Kedže **PausePanel** obsahuje tlačidlá, obsahuje tento manažér metódy, ktoré tieto tlačidlá obsluhujú. Takto máme k dispozícií metódu na návrat do preteku (*Resume*), metódu na návrat do hlavného menu (*ReturnToMainMenu*) a metódu na reštart preteku (*Restart*), pričom sa v tejto metóde uložia indexi vygenerovaných vozidiel pre AI z dôvodu, aby ich **CarGeneratorManager** vedel znova vygenerovať.

Okrem toho obsahuje **PausePanel** metódy na ovládanie vyskakovacích info okien. Tieto metódy sú pripevnené k **TriggerEvent-u** jednotlivých tlačidiel. Konkrétne k eventom *PointerDown - ResumeDown, RestartDown, ReturnToMenuDown*, a eventom *PointerExit – ResumeUp, RestartUp a BackToMenuUp*. Tieto eventy sa spúšťajú, ak je kurzor na tlačidlu, resp. mimo neho.

PositionManager

Úlohou tohto manažéra je korektne určovať pozíciu hráča a prideľovať rýchlostné zrýchlenie, resp. spomalenie ostatným vozidlám na základe toho, ako sú umiestnené v závislosti od hráča. Na začiatku v metóde *Start* získame prístup k vozidlám hráča a AI vozidiel a v metóde *Update* neustále updatuje pozíciu hráča v **Position** paneli. Zároveň sa volá metóda *PositionOfTheCar*, ktorá zisťuje pozíciu nášho vozidla.

Táto metóda zisťuje pozíciu hráča na základe vozidiel, ktoré majú lepšie kolo ako vozidlo hráča (*LapCheck*), prešli vyššie ID checkpointu než vozidlo hráča (*CheckpointPassCheck*), a ktoré sú bližšie najbližšiemu checkpointu než vozidlo hráča (*CheckpointDistanceCheck*). Každá z týchto metód vyberá vozidlá, ktoré majú lepšiu pozíciu ako vozidlo hráča (tieto vozidlá spomaľuje), a vozidlá, ktoré sú na tom rovnako ako vozidlo hráča. Tie sa prepadajú do ďalšej metódy a vozidlá, ktoré sú horšie než vozidlo hráča, zrýchlime.

PrefabsLoadManager

Tento manažér sa stará o to, aby boli po načítaní scény v objekte **CarModels** vygenerované všetky vozidlá, ktoré hra obsahuje. Táto úloha sa vykonáva v metóde *Start* tak, že načíta všetky **prefabs** z priečinka *Prefabs/Cars* v zložke *Resources* pomocou metódy *Resources.LoadAll*. Následné tieto načítané vozidla vygeneruje pomocou metódy *Instantiate*.

ReplayManager

Rovnako ako v prípade **GhostCarManager**, sa aj tento manažér stará o záznam. Konkrétne sa stará o replay záznam preteku a o správne fungovanie **ReplayPanel** panelu, ktorý umožňuje hráčovi záznam vidieť a ovládať ho. Tento záznam sa nikam neukladá a po opustení scény zaniká. Na začiatku sa v metóde *Start* zistí, v akom hernom móde sa hráč nachádza (aby sme vedeli, či sú prítomné AI vozidlá), a začne sa s nahrávaním preteku pomocou metódy *Record*.

Táto metóda vytvorí zoznam pre každé vozidlo, ktoré sa v preteku nachádza (zoznam hráča je prvý). Všetky záznamy obsahujú informácie o vozidlách, ktoré používame počas prehrávania záznamu a udržujeme si ich v nami skonštruovanej štruktúre **CarNode**, ktorá je definovaná v tomto skripte. Po príprave zoznamov sa spustí koruntína pomocou metódy *StartRecord*, ktorá zaznamenáva pohyb, čas hráča (*RecordPlayer*) a záznamy pre AI vozidlá (*RecordAI*). Záznam o čase sa pre hráča obnovuje v metóde *Update*.

V prípade, že sa hráč rozhodne po preteku spustiť replay, zapne metóda *Replay* **ReplayPanel**, pripraví vozidlá na prehratie záznamu pomocou metódy *PrepareCarsToBePlayed*, ktorá im vypne zvuk a všetky svetlá. Po príprave vozidiel spustí metóda *PlayRecord* koruntínu (*PlayRecordedData*), ktorá spustí záznam.

Manažér záverom obsahuje metódy, ktoré ovládajú tlačidla panelu **ReplayPanel**. Tieto metódy slúžia na zastavenie/spustenie záznamu (*PlayOrStop*), spustenie záznamu od začiatku (*Restart*) a posunutie záznamu dopredu alebo dozadu (*Rewind*). Tieto metódy sú umiestnené ako **EventTrigger-e** na jednotlivých tlačidlách.

SpeedometerManager

Táto trieda má za úlohu korektné ovládanie tachometra a všetkých jeho prvkov. Pri aktivovaní skriptu v metóde *OnEnable* sa nastaví objekt reprezentujúci ihlu tachometra (**needle**) na štartovaciu hodnotu. Ďalej sa metóda *Udpate* stará o to, aby sa jej na digitálnej časti tachometra ukazovala korektná rýchlosť a rýchlostný stupeň. Počas update-tu sa neustále volá metóda *UpgradeNeedle*, ktorá sa stará o to, aby ihla tachometra neustále ukazovala správne otáčky vozidla.

TimeManager

Úlohou tohto manažéru je korektne obsluhovať **TimePanel**, ktorý sa nachádza v objekte **UI/GameUIPanel**. Táto trieda v metóde *Start* vyčistí panely (*ClearPanels*) a nahrá doň rekord trate a najlepší čas hráča (*LoadBestAndPersonalTimes*). V metóde *Update* sa počíta čas, ktorý ubehol od začiatku preteku. Trieda obsahuje naviac verejnú metódu *SetLapTime*, ktorá po každom prejdenom kole vynuluje časomieru, a čas, za ktorý hráč prešiel kolo, zapíše do panelu **LapTimes**.

4.2.2 Triggers and Paths

LapTrigger

Tento skript je pripevnený k objektu **LapTrigger** umiestnenom na koncovej zástave preteku. Jeho úlohou je detekovať koniec preteku. Na začiatku si v metóde *Start* vynuluje všetky dôležité premenné a zistí si počet kôl v danom preteku (v premennej **TrackSelectionSceneManager.NumberOfLaps**).

Metóda *OnTriggerEnter* počíta vozidlá (pre K.O Mód), ktoré prešli týmto triggerom, a metóda *OnTriggerExit* počíta kolá, ktoré hráč prešiel. V prípade, že hráč prešiel požadovaný počet kôl, ukončí sa pretek (aktivuje **EndGameManager** skript) a vypnú sa všetky merania a nahrávania. Okrem toho, ak v K.O móde prejdu týmto triggerom všetky vozidlá okrem posledného (*TryFindAndDestroyLastCar*), skript posledné vozidlo deaktivuje. V prípade, že sa jedná o hráča, pretek ukončí.

TrackPath

Posledný skript, ktorý obsahuje každá trať, ma za úlohu vykresliť všetky waypointy z objektu **Path** na *Gizmos* scény. Samotný skript je umiestnený na objekt **Path** a vykresľovanie vykonáva pomocou metódy *OnDrawGizmos*, ktorá vyberie všetky waypointy (*GetPath*) a následne ich vykreslí pomocou metódy *Gizmos.DrawSphere*, ktorá každý waypoint vykreslí ako guľu. Následne *Gizmos.DrawLine* každé dva po sebe idúce waypointy spojí čiarou.

4.3 CameraControler

Hlavná kamera, ktorá pretek zaznamenáva, obsahuje jediný skript **CameraControler**, ktorý pri spustení (*OnEnable*) presunie hlavnú kameru na pozíciu objektu **FollowPoint**, ktorý obsahuje každé vozidlo, a otočí kameru smerom na hráčovo vozidlo. Potom sleduje hráča pomocou metódy *FixedUpdate*. Samotné dynamické správanie kamery dosahujeme metódou *Vector3.Lerp*, ktorá kameru približuje v objekte **FollowPoint** za istý čas. Tento čas meníme v závislostí na rýchlosti vozidla vďaka čomu dosahujem dynamický efekt.

4.4 Menu

V tejto sekcií sa pozrieme na všetky ostatné scény, ktoré sa nachádzajú v našej hre. Počet scén, ktoré tvoria menu v našej hre je 9. Teraz si opíšeme objekty a

skripty, ktoré sa v týchto scénach nachádzajú.

4.4.1 Welcome and EndCredits Scene

Tieto 2 scény sme spojili do jedného odseku, pretože obe obsahujú jeden skript. Scéna **Welcome** má za úlohu hráča privítať. Obsahuje skript **WelcomeManager**, ktorý je pripevnený k objektu **WelcomePanel**. Tento skript čaká, keď hráč klikne alebo stlačí akékoľvek tlačidlo na klávesnici. Po stačení okamžite prejde na scénu **StartMenu** pomocou metódy *SceneManager.LoadScene*.

Scéna **EndCredis** má za úlohu informovať hráčov o tom, kto je autor hry, kto boli testeri pre túto hru a taktiež ma hráčom povedať, kto sú autori assetov, ktoré sme v hre použili. Scéna **EndCredits** obsahuje skript **EndCreditsManager** umiestnený v objekte **EndCreditBackround**. Tento skript v metóde *Start* na začiatku spúšťa koruntínu *EndCreditRoll*, ktorá roluje titulky. Metóda *Update* rovnako ako **WelcomePanel** kontroluje, či hráč nestlačil nejaké tlačidlo. Ak áno, tak sa vrátime na scénu **StartMenu**.

4.4.2 StartMenu Scene

Scéna **StartMenu** slúži ako prvá skutočná scéna, v ktorej hráč môže rozhodnúť, čo chce robiť ďalej. Obsahuje skript **StartMenuManager**, ktorý obsahuje štyri metódy, ktoré slúžia ako **EventTrigger-e** pre jednotlivé tlačidla na scéne. Všetky prechádzajú na inú scénu podľa názvu metód, pričom možné scény sú **NewGame, LoadGame, Credits** a *ExitGame*, ktorá hru ukončuje.

Táto scéna naviac obsahuje objekt **CarContolInfo**, ktorý riadi pomocou skriptu zapnutie, resp. vypnutie **CarInfoPanelu**, ktorý obsahuje informácie o tom, ako sa hra ovláda. Tento skript obsahuje metódy na aktiváciu (*CarInfoPanelActivated*) a vypnutie (*Exit*) tohto panelu.

Záverom spomenieme, že táto scéna ako prvá obsahuje objekt **BGM**, ktorý nesie audio, ktoré sa prehráva počas všetkých scén (okrem scén Welcome, EndCredits a scén tratí, ktoré majú vlastnú hudbu). Aby sa hudba počas presunu zo scény na scénu neprehrávala znova, používame skript **SoundTransition**, ktorý v metóde *Awake* priradí inštanciu samého seba do statickej premennej **SoundTransition**. Na túto premennú použijeme metódu *DontDestroyOnLoad*, ktorá zaistí, že sa tento objekt nezničí pri prechode do inej scény. Týmto zariadime, aby sa hudba medzi scénami neresetovala. V metóde *Update* potom dávame pozor, či scéna, v ktorej sa nachádzame, má prehrávať danú hudbu. Ak nie, tak objekt zničíme pomocou metódy *Destroy*.

4.4.3 New Game and Load Game Scene

Obe tieto scény pracujú s hráčskymi účtami a obsahujú dve textové polia na hráčske meno a heslo. Obe tieto scény kontrolujú dve skripty - jeden ovláda prihlasovanie/vytváranie účtov a druhý kontroluje tlačidla na daných scénach.

New Game

Scéna **NewGame** obsahuje riadiaci skript **NewGameManeger**, ktorý je pripevnený k objektu **SceneManager**. Tento skript na začiatku (*Start*) zablokuje

tlačidlo **Create**, aby zabránil vytvoreniu účtu bez korektného mena a hesla. Ďalej v metóde *Update* kontroluje, či hráč zadal správne meno (*CheckName*) a heslo (*CheckPassword*). Ak áno, odblokuje tlačidlo **Create**. Ak meno alebo heslo nie je korektné, aktivuje na scénu výstražný panel, ktorý upozorní hráča na nekorektné meno alebo heslo (*ShowNameWarning, ShowPasswordWarning*). Pod korektným meno rozumieme meno, ktoré je dlhé aspoň tri znaky a nie je použité v inom účte. Korektné heslo pozostáva z aspoň troch znakov.

Ďalej táto scéna obsahuje skript **NewGameButtonHandler**, ktorý sa nachádza v objekte **ButtonHandler**. Obsahuje metódy, ktoré slúžia ako **EventTrigger-e** pri stlačení tlačidla. Metóda *Create* vytvorí hráčovi účet (pomocou metódy *CreateAccount* v skripte **NewGameManeger**) a posunie ho do scény **ModeSelection**. Metóda *Back* vráti hráča do scény **StartMenu**.

Load Game Scene

Táto scéna ma za úlohu umožniť hráčovi prihlásiť sa, poprípade vymazať svoj hráčsky účet. K tomu používa skript **LoadGameManager**, ktorý je pripevnený k objektu **SceneManager**. Tento skript na začiatku rovnako ako **NewGameManager** deaktivuje tlačidla – tentoraz **Load** a **Delete**. Potom v metóde *Update* kontroluje, či hráč zadal správne meno (*CheckName*) a heslo (*CheckPassword*). Ak áno, odblokuje tlačidlá **Load** a **Delete**. Ak meno alebo heslo nie je korektné, aktivuje na scénu výstražný panel, ktorý upozornia hráča na nekorektné meno alebo heslo (*ShowNameWarning, ShowPasswordWarning*). Pod korektným meno rozumieme meno, ktoré je použité na nejakom hráčskom účte a pod korektným heslom rozumieme heslo, ktoré odpovedá heslu danému menu hráčskeho účtu.

Ďalej táto scéna obsahuje skript **LoadGameButtonHandler**, ktorý sa nachádza v objekte **ButtonHandler**. Obsahuje metódy, ktoré slúžia ako **EventTrigger-e** pri stlačení tlačidla. Metóda *Load* prihlási hráča k jeho účtu (pomocou metódy *Load* v skripte **LoadGameManeger**) a posunie ho do scény **ModeSelection**. Metóda *Back* vráti hráča do scény **StartMenu** a metóda *Delete* vymaže hráčsky účet (pomocou metódy *Delete* v skripte **LoadGameManeger**).

4.4.4 Mode Selection Scene

Scéna má za úlohu možnosť pre hráča vybrať si herný mód, v ktorom chce hrať. Túto scénu ovláda skript **ModeSelectionButtonHandler** obsiahnutí v objekte **ButtonHandler**, ktorý v metóde *Awake* zistí, či hrá hudba u objektu **BGM** (ktorý táto scéna obsahuje). A ak nie, tak ju zapne. Táto kontrola sa vykonáva preto, lebo po skončení preteku sa ocitneme v tejto scéne a teda nemôžeme prenášať objekt **BGM** zo scény **StartMenu**. Okrem toho tento skript obsahuje metódy pre **EventTrigger-e** tlačítok, ktoré sa aktivujú pri ich stlačení (*RaceMode, KOMode, TimeMode*). Všetky tieto metódy slúžia na zvolenie herného módu a prechod do scény **CarSelection**. Informácia o zvolenom móde sa zapíše do statickej triedy **Player** (viac v sekcií 4.5.1). Na záver skript obsahuje **EventTrigger-e** tlačítok, ktoré sa aktivujú vstúpení na tlačítka kurzorom či ich opustení.

Rovnako, ako v prípade scén **NewGame** a **LoadGame**, obsahuje scéna tlačidlo *Back*, ktoré slúži na návrat do scény *StartMenu*.

4.4.5 Car Selection Scene

Táto scéna slúži na výber, upgrade a odomykanie vozidiel, ktoré máme v tejto hre k dispozícií. Na to potrebujeme niekoľko skriptov, ktoré všetky tieto akcie riadia. V tejto sekcií si všetky tieto skripty prejdeme, popíšeme si ich spoluprácu a spomenieme, aké objekty ich obsahujú.

PointsAndNameHandler

Tento skript ma za úlohu vyplniť panel **PlayerPanel** s korektnými informáciami o mene hráča a počte jeho bodov. Na začiatku (*Start*) vyplní meno hráča a potom (*Update*) neustále aktualizuje počet hráčových bodov, keďže je možné, že hráč body použije na odomknutie či upgrade.

CarSelectionGenerator

Úlohou tohto skriptu je vygenerovať vozidlá do objektu **CarSelection**, v ktorom je tento objekt umiestnený. V úvode si tento skript v metóde *Awake* načíta a vygeneruje všetky vozidlá, ktoré v hre máme (rovnako ako v 4.2.1). Záverom táto metóda aktivuje skript **CarSelection**. Tento skript okrem tejto metódy obsahuje aj metódu na rozdelenie vozidiel podľa modelov (*SeperateCarsColors*), ktorá vozidlá a všetky ich farebné varianty rozdelí do zoznamov a prevedie ich mená do správneho formátu. Tento prevod je vykonávaný metódou *ChangeCarNameToCorrectFormat*. Každé vozidlo, ktoré máme, ma meno vo formáte **Car**__*Značka vozidla-Farba vozila*. Po každom vygenerovaní vozidla sa k tomuto menu pridá postfix **(Clone)**. Korektný formát vozidla v tomto kontexte je *Značka vozidla-Farba vozila*.

CarSelection

Tento skript obsluhuje **CarSelectionPanel** a umožňuje hráčovi prezerať a prechádzať si jednotlivé vozidlá. Rovnako ako skript **CarSelectionGenerator** je umiestnení v objekte **CarSelection**. Skript na začiatku vozidlá v tomto objekte rozdelí pomocou farieb pomocou metódy *SeperateCarsColors*. Potom vozidlá zotriedi pomocou metódy (*SortCars*) tak, aby prvé vozidlá, ktoré sa zobrazujú hráčovi, boli odomknuté vozidlá. Ďalej deaktivuje všetky vozidlá a pomocou metódy *SlectCar* aktivuje prvé vozidlo v našom zotriedenom zozname. Metóda *Update* sa stará o to, aby v **CarSelectionPanel-i** bolo napísané korektné meno a farba daného vozidla. Zároveň zariadi, aby sa objekt **RotationPlatform**, ktorý slúži ako platforma, na ktorej sa objavujú vozidlá, pomaly otáčal.

Metódy *ChangeColor* a *ChangeCar* slúžia ako **EventTrigger-e** tlačítok po ich stlačení. Konkrétne menia vozidlo, ktoré sa na platforme zobrazuje a ak je to možné, tak aj jeho farbu. V prípade, že dôjdeme na koniec zoznamu vozidiel, začíname od začiatku. Teda tlačidlá na výber vozidiel a farby fungujú cyklicky.

UnlockManager

Tento skript má zaručovať správne odomykanie vozidiel. Je umiestnení na objekte **UnlockManager**. Pri aktivácií skriptu (*OnEnable*) vypne **Unlocked** panel, ktorý poskytuje informácie o vozidle, a aktivuje **Lock** panel, vďaka ktorému

je možné vozidlo odomknúť. Nakoniec aktivuje obrázok na **CarSelectionPanel-i**, ktorý ilustruje, že vozidlo je zamknuté. Pri vypnutí skriptu (*OnDisable*) sa vykoná presný opak.

Ďalej tu máme metódu *Unlock*, ktorá slúži ako **EvnetTrigger** tlačidla v prípade jeho stlačenia. Metóda v prípade, že má hráč dosť bodov, odomkne vozidlo a hráčovi aktivuje pochvalné okno (*Success*). V prípade, že hráč nemá dosť bodov, aktivuje výsmešné okno (*Fail*). Obe tieto okná sa dajú vypnúť pomocou metódy *Continue*, ktorá je takisto **EventTrigger** tlačidla.

UpgradeManager

Skript ovláda upgrade vozidiel a celý panel **UpgradePanel**. Skript je pripevnený k objektu **UpgradeManager** a v úvode (*Start*) pomocou metódy *FillSlidersValues* vyplní všetky slidre, ktoré reprezentujú ako veľmi je vozdlo vylepšené a ako veľmi sa dá ešte vylepšiť. Ďalej neustále kontrolujeme (*Update*), či je možné vozidlo vylepšovať, resp. či má hráč dosť bodov na vylepšenie niektorého z parametrov. Na tomto základe metóda *ButtonInteractibilityCheck* aktivuje/deaktivuje tlačidlá na úpravu parametrov.

Všetky ostatné metódy slúžia ako **EventTrigger-e** pre stlačenie tlačítok. Metóda *Apply* potvrdí zmeny, uloží nový bodový stav hráča a deaktivuje **UpgradePanel**. Metóda *Exit* deaktivuje panel. Všetky ostatné metódy slúžia na pridávanie/uberanie vylepšení na vozidle.

CarSelectionSceneManager

Tento skript riadi celú scénu a s ňou proces výberu, odomykania a upgradu vozidiel. Je umiestnení v objekte **SceneManager**. Na začiatku (*Awake*) sa uistí, že sú aktivované správne panely (a že panel na upgrade je vypnutí). Potom v metóde *Update* nastavíme maximálne hodnoty sliderov (*FillMaxValueForSliders*) a neustále aktualizujeme ich súčasne hodnoty. Zároveň, ak narazíme na odomknuté vozidlo, ktoré neodomkol hráč (bolo odomknuté hrou), tak toto vozidlo pridáme k odomknutým vozidlám hráča (*UnlockCarForPlayer*). Nakoniec metóda rozhoduje, či zobrazovať **Unlock panel** alebo **Lock panel**, na základe toho, či je vozidlo odomknuté (*EnableUnlock*) alebo nie (*DisableUnlock*).

Ďalej skript obsahuje dve statické metódy. Konkrétne *NewCar*, ktorý v prípade vybratia nového vozidla (v skripte **CarSelection**) získa informácie o vozidle z našej databázi (viac v sekcií 4.5.2), a metódu *IsCarUnlocked*, ktorú taktiež využíva skript **CarSelection** pri triedení vozidiel na zistenie, ktoré vozidlá sú odomknuté.

Posledná metóda slúži ako **EventTrigger** pre tlačidlo **Upgrade** (názov metódy rovnaký). Metóda ma za funkciu aktivovať **UpgradePanel** a zapnúť **UpgradeManager** skript.

CarSelectionButtonHandler

Posledný skript v tejto scéne poskytuje metódy ako **EventTrigger-e** na prácu s tlačidlami **Next** a **Back**. Metóda *Next* nás prenesie do scény **TrackSelection** a metóda *Back* späť do **ModeSelection** scény.

4.4.6 Track Selection Scene

Posledná scéna menu poskytuje hráčovi možnosť vybrať si trať, na ktorej chce pretekať. Rovnako ako v minulej scéne na výber vozidiel, aj tu je možné trate odomykať.

PlayerTableInfoManager

Rovnako ako v prípade výberu vozidiel, aj tu máme skript na ovládanie panelu **PlayerPanel**. Tento skript je umiestnení v objekte **PlayerTableManager** a predošlý skript rozširuje o vyplnenie informácie o vybranom vozidle hráča počas metódy *Start*.

TrackSelectButtons

Skript slúži na generovanie a výber tratí z panelu **TrackSelectionPanel**. Skript sa nachádza v objekte **SelectionHandler**. Skript na začiatku (*Start*) nahrá všetky obrázky tratí z priečinka *Images/TrackImages*, ktorý sa nachádza v priečinku *Resources*. Tieto obrázky sú v tejto zložke uložené ako **Sprite** a do skriptu si ich ukladáme ako **Texture2D**. Záverom metódy trate zotriedime a vyplníme ich meno. Každý obrázok trate v mene obsahuje reťazec **Track**. Takto vieme rozoznávať obrázky tratí od nejakých iných obrázkov.

Samotné trate v panely **TrackSelectionPalen** meníme metódou *Move*, ktorá ako **EventTrigger** mení trate pomocou jednoduchej zmene indexu v zázname obrázkov tratí.

TrackSelectionSceneManager

Záverom máme skript, ktorý ovláda celú scénu. Tento skript si pri prebudení (*Awake*) nahrá databázu tratí (viac v sekcií 4.5.3). Potom neustále (*Update*) rozhoduje o tom, či má byť tlačítko **Race** aktívne alebo nie. V prípde zamknutej trate alebo nekruhovej trate pri výbere K.O. módu (je nutné viac než jedno kolo) sa toto tlačítko zablokuje.

Metóda *NextTrack* má za úlohu zistiť, či je trať odomknutá a na základe toho aktivovať **LockPanel** alebo **InfoPanel** pomocou metód *Lock*, resp. *Unlock*. Túto metódu využíva skript **TrackSelectButtons** v oboch svojich metódach. V prípade metódy *Unlock* sa vyplnia do panela informácie o časových rekordoch trate a informácie o bodoch, ktoré môže hráč získať v daných časoch, umiestneniach alebo eliminaiciách. Na vyplnenie týchto informácií sa používa metóda *FillTrackInfo*.

Ďalej skript obsahuje metódy, ktoré sú **EventTrigger-e** pre tlačítka **Race, Back** a **Unlock**. Metóda *UnlockButton* odomkne trať v prípade, že má hráč dosť bodov, a oznámi mu úspešné zakúpenie trate (*Succes*). Ak nemá dosť bodov, tak ho na to metóda upozorní (*Fail*). Obe okná sa vypínajú tláčidlom, resp. metódou **Continue**. Metóda *Back* hráča premiestni do scény **CarSelection** a metóda *Race* hráča presunie do preteku na trati, ktorú si vybral.

4.5 Data

V minulých sekciách sme popisovali hráčske účty, vozidlá, trate a rôzne informácie, ktoré z nich čítame. Všetky dáta, ktoré v hre používame, reprezentujeme pomocou XML. V tejto sekcií sa pozrieme, ako si tieto dáta štrukturujeme, aké informácie obsahujú a pomocou akých nástrojov s nimi pracujeme.

4.5.1 Player Data

Začneme s dátami, ktoré popisujú každého nášho hráča. Tieto dáta musia poskytovať prehľad nie len o hráčovom účte a bodoch, ale taktiež informácie o vozidlách, tratiach a osobných rekordoch, ktoré hráč dosiahol.

Štruktúra dát XML

Na obrázku 4.4 môžeme vidieť, ako vyzerá štruktúra hráča v XML. U každého hráča si pamätáme heslo a počet bodov, ktoré hráč má. Naviac záznam každého hráča obsahuje zoznamy vozidiel a tratí, ktoré hráč odomkol. Každé vozidlo a trať má ID, ktoré je reprezentované menom modelu vozidla (s modelom hráč odomkne všetky farby vozidla). Spolu s ID vozidla tieto záznamy obsahujú informácie o tom, ako veľmi je každé vozidlo vylepšené.

```
<player password="_" points ="_">
  <cars>
    <car id="_" maxSpeed = "_" torgue = "_" brake = "_"/>
  </cars>
  <tracks>
    <track id="_" bestTime="_"/>
  </tracks>
</player>
```

Obr. 4.4: Štruktúra hráčovho XML.

V našej štruktúre nie je uvedené užívateľské meno hráča. Je to z tohto dôvodu, že hráčske účty ukladáme pomocou *PlayerPrefs*, ktoré fungujú podobne ako **Dictionary** v jazyku C#. Konkréte tým myslíme, že každý záznam obsahuje kľúč. Kľúčom pre každý záznam je práve meno užívateľa. Takto sme zaistili, že sa nevyskytnú dva rovnaké hráčske účty. Trate obsahujú ID trate a informáciu o osobnom časovom rekorde hráča.

Player

Spomínanú štruktúru ovládame pomocou skriptu *PlayerManager*, ktorý obsahuje statickú triedu *Player*. Statická trieda nám zaistí, že nebudeme vytvárať žiadne inštancie hráčov (v každej hre je len jeden hráč). Taktiež hodnota statických premenných nie je viazaná na žiadnu scénu, takže si takto môžeme prenášať informácie o zvolených módoch, vozidlách a tratiach počas našej hry. Teraz stručne popíšeme metódy, ktoré využívame v tejto metóde:

- *FillInfo* – z XML štruktúry získa všetky informácie o vozidlách (*FillCarInfo*), tratiach (*FillTrackInfo*) a bodoch *FillPoints*, a uloží ich do premenných, ktoré patria tejto triede.
- *UnlockCar* – pridá odomknuté vozidlo do XML štruktúry, obnoví počet bodov hráča a uloží aktuálnu XML štruktúru hráča.
- *UpgradeCar* – pridá upgrade k danému vozidlu, obnoví počet bodov a uloží aktuálnu XML štruktúru.
- *GetPassword* – vracia heslo od hráčskeho účtu. Slúži na kontrolu pri prihlasovaní sa k účtu.
- *GetMaxSpeedUpgrade, GetTorgueUpgrade, GetBrakeUpgrade* – vracajú hodnoty upgratov na danom vozidle. Tieto informácie potrebujeme pre scénu **TrackSelection**. Konkrétne pre správne vyplnenie sliderov o atribútoch a upgradoch vozidla.
- *IsCarUnlocked* – vracia informáciu, či je vozidlo zamknuté. Slúži pri zobrazovaní **Lock** panelu v scéne **CarSelection**.
- *UnlockTrack* – zapíše novo odomknutú trať do XML štruktúry, obnoví body a uloží aktuálny stav XML štruktúry pre hráča.
- *IsTrackUnlocked* – vracia informáciu, či je zvolená trať zamknutá. Slúži pri zobrazovaní **Lock** panelu v scéne **TrackSelection**.
- *AddPoints, SetNewRecord* – pridá do XML štruktúry body, resp. nový rekord k danej trati a uloží zmeny v štruktúre XML.

4.5.2 Car Data

Na obrázku 4.5 môžeme vidieť štruktúru nášho vozidla vo formáte XML. Každé vozidlo obsahuje meno, informáciu o tom, či je na začiatku odomknuté alebo nie, a cenu, za ktorú môžeme dané vozidlo odomknúť.

```
<cars>
  <car Name="_" IsLocked="True/False" Price="_">
    <DriveType Value="ALL/FRONT/BACK" />
    <SteerAngle Value="_" />
    <Upgradable>
      <MaxSpeed Upgrade="_" Shift="_" ShiftPrice="_" OriginalValue="_" />
      <Torgue Upgrade="_" Shift="_" ShiftPrice="_" OriginalValue="_" />
      <Brake Upgrade="_" Shift="_" ShiftPrice="_" OriginalValue="_" />
    </Upgradable>
    <Colors>
      <Color Name="_" />
    </Colors>
  </car>
</cars>
```

Obr. 4.5: Štruktura databázi vozidiel v XML.

Ďalej obsahuje vozidlo informácie o type pohonu (pohon všetkých kolies, predný náhon, zadný náhon) a maximálny uhol otáčania kolies. Nakoniec obsahuje každá štruktúra dva zoznamy. Prvý obsahuje informácie o prvkoch vozidla,

ktoré je možné vylepšiť, o koľko sa dajú vylepšiť, veľkosť jedného vylepšenia, cena za jedno vylepšenie a pôvodná hodnota daného prvku. Druhý zoznam je zoznam farieb vozidla.

Túto štruktúru udržujeme v priečinku *Resourses/DataFile*.

Ďalšia datová štruktúra

XML štruktúra je čitateľná a ľahko opraviteľná, no v skripte môže byť trocha komplikovaná, hlavne keď s ňou chceme často pracovať. Preto sme vytvorili štruktúru na ľahkú prácu s touto štruktúrou v skriptoch. Táto štruktúra sa volá **XmlCarData** a obsahuje všetky dáta, ktoré sa v nachádzajú v XML. Aby sme túto štruktúru spriehľadnili a zefektívnili si jej používanie, vytvorili sme štruktúry **CarUpgrade** a **CarColor**, ktoré skladujú vylepšiteľné prvky a farby našich vozidiel.

XmlCarManager

Rovnako ako v prípade hráčskych účtov aj na túto štruktúru sme si vybudovali statickú triedu na prácu s touto štruktúrov. Metódy, ktoré táto metóda obsahuje:

- *CheckAndAddCarToList* – skontroluje validitu mena vozidla (*IsCarNameValid*) a či sa vozidlo nenachádza v štruktúre (*IsCarAlreadyPressent*). V prípade, že je meno vozidla valídne a vozidlo sa ešte nevyskytlo v štruktúre, pridá vozidlo do štruktúry (*AddCar*). Túto metódu používa Editor.
- *CheckAndAddColorToList* – skontroluje validitu mena vozidla (*IsCarNameValid*) a či sa farba nenachádza v štruktúre vozidla (*IsColorAlreadyRegistered*). V prípade, že je meno vozidla validné a farba jedinečná, pridá farbu vozidla do štruktúry (*AddColor*). Túto metódu používa Editor.
- *TryDeleteCar* – v prípade, že sa vozidlo vyskytuje v štruktúre, vymaž ho. Túto metódu používa Editor.
- *TryGetCarValues* – v prípade, že dané vozidlo existuje, vytiahne z neho všetky informácie (*ExtractDataFromXml*) a vráti ich vo forme **XmlCarData**.
- *CreateUpgradeElement, FillUpgradeInfoToNode* – vytvorí XML element pre upgradovacie prvky vozidla, resp. ho naplní danými hodnotami.
- *CreateSimpleNode* – vytvorí jednoduchý XML uzol (bez potomkov).

4.5.3 Track Data

V prípade tratí, je štruktúra XML najkomplikovanejšia. Na obrázku 4.6 môžeme vidieť ako takáto štruktúra vyzerá v našej hre. Každá trať má meno, cenu, za ktorú sa dá odomknúť, a informácie o tom, či je zamknutá a či je trať okrúhla (nemá koniec). Informácie o tom, či trať nie je okrúhla, je nutná kvôli K.O. módu, ktorý sa nemôže odohrávať na neokrúhlych tratiach.

```xml
<tracks>
  <track Name="_" Price="_" isLocked="True/False" isCurcilar="True/False">
    <modes>
      <raceMode>
        <a Position="1st" Value="_" />
        <b Position="2nd" Value="_" />
        <c Position="3rd" Value="_" />
      </raceMode>
      <timeMode>
        <a Position="_" Value="_" />
        <b Position="_" Value="_" />
        <c Position="_" Value="_" />
      </timeMode>
      <koMode>
        <a Position="Wans't eliminated" Value="_" />
        <b Position="Eliminated as last" Value="_" />
        <c Position="Eliminated as penultimate" Value="_" />
      </koMode>
    </modes>
  </track>
</tracks>
```

Obr. 4.6: Štruktúra databázy tratí v XML.

Ďalej obsahuje štruktúra trate informácie o jednotlivých módoch – všetky bodovo ohodnotené pozície, časy a eliminácie. Zároveň obsahuje bodové odmeny za jednotlivé pozície, časy a eliminácie. V prípade, že trať nie je okrúhla, nemá informácie o K.O. móde.

Ďalšia datová štruktúra

Rovnako ako v prípade vozidiel, aj v tomto prípade používame štruktúru pri práci s XML. Táto štruktúra sa nazýva **XmlTrackData** a obsahuje všetky informácie z XML. Používa pri tom pomocnú štruktúru **TrackMode**, ktorá popisuje limity a body jednotlivých módov. Tieto štruktúry nám pomáhajú hlavne pri pridávaní nových tratí a vozidiel pomocou Editora (viď 5.2.3).

XmlTrackManager

Záverom si popíšeme statickú triedu *XmlTrackManager*, ktorá nám umožňuje prácu s XML tratí. Táto trieda obsahuje:

- *CheckAndAddTrack* – skontroluje, či sa daná trať nenachádza v štruktúre (*CheckAndAddTrack*), a pridá trať do štrukúry (*CreateNewTrack*). Túto metódu používa Editor na pridávanie nových tratí.
- *CreateTrackMode* – vytvára herný mód pre XML. Používa ho *CreateNewTrack*.
- *TryGetTrackData* – v prípade, že trať existuje, vytiahne z nej dáta (*GetTrackData*) a vráti ich v podobe *XmlTrackData*. Využíva sa v scéne *TrackSelection*.

5. Užívateľská Dokumentácia

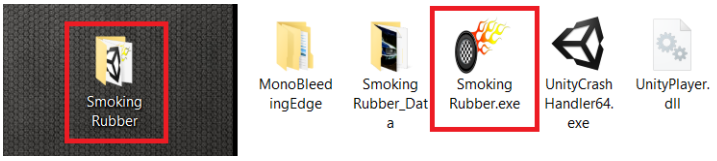
V tejto kapitole prevedieme užívateľov našou hotovou hrou. Ukážeme a popíšeme všetky možnosti, ktoré naša hra ponúka. Tento popis nebude zameraný len na hráčov našej hry, ale aj na dizajnérov a programátorov, ktorý by túto hru chceli v budúcnosti rozširovať.

5.1 Tutorial pre hráčov

Začneme návodom pre užívateľov, ktorý budú našu hru hrať. V tomto návode si prejdeme všetky možnosti, ktoré hra poskytuje hráčovi a ukážeme, ako úspešne hru spustiť a ako sa v nej orientovať.

5.1.1 Spustenie hry

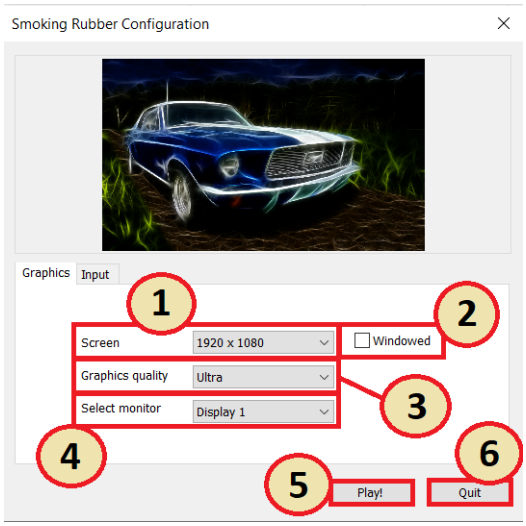
Samotná hra sa nachádza v priečinku **Smoking Rubber**, ktorý je súčasťou elektronickej prílohy. Hra bola vytvorená pre operačný systém **Windows**. Na obrázku 5.1 môžeme vidieť, ako po otvorení tohto priečinka budeme mať pred sebou niekoľko možností, pričom jedna z týchto možností bude aplikácia **BakalarskaPraca_RaceGame.exe**. Spustenie tejto aplikácie nám otvorí hernú konfiguráciu.



Obr. 5.1: Spustenie hry

Konfigurácia

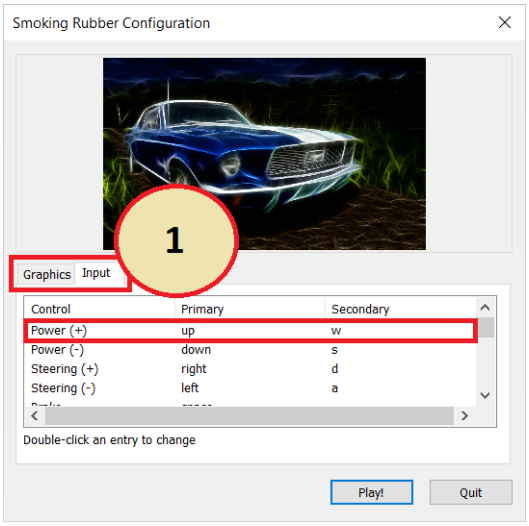
Konfigurácia hry zobrazená na obrázku 5.2 umožňuje hráčom nastaviť si isté grafické parametre. Konkrétne mu táto konfigurácia ponúka výber rozlíšenia hry (1) a taktiež možnosť neroztiahnutia hry na celú obrazovku (2). Ďalej nám konfigurácia ponúka možnosť výberu kvality renderovania hry (3) a v prípade, že hráč používa zariadenie s viacerými monitormi, aj možnosť výberu monitora (4).



Obr. 5.2: Konfigurácia grafiky.

Hráč samozrejme všetky tieto konfigurácie môže nastavovať kedykoľvek pred spustením hry. V prípade, že konfigurácie hráčovi vyhovujú, môže hru spustiť (5) alebo všetko ukončiť (6).

Na obrázku 5.3 si môžeme všimnúť, že v prípade, ak si hráč chce nastaviť ovládanie hry, môže tak vykonať prepnutím konfigurácie z **Grafics** na **Input**, čo mu túto zmenu umožní (1). Zmenu ovládania hráč vykoná dvojklikom na ovládací prvok (v stĺpci *Primary* alebo *Secondary*), ktorý chce zmeniť.



Obr. 5.3: Konfigurácia vstupu.

Teraz vypíšeme všetky prvky ovládania, ktoré hráč v konfigurácií môže zmeniť.

- *Power (+)* – zrýchlenie vozidla.
- *Power (-)* – spomalenie/cúvanie vozidla.
- *Steering (+)* – zatáčanie s vozidlom doprava.
- *Steering (-)* – zatáčanie s vozidlom doľava.

- *Brake* – brzenie vozidla.
- *Hand Brake* - ručná brzda vozidla.
- *Quick Camera* – spúšťanie quick kamery vozidla.
- *Pause* – zapínanie/vypínanie pauzy počas preteku.
- *Respawn* – manuálny respawn vozidla.
- *Front Lights* – zapínanie/vypínanie predných svetiel vozidla.

Okrem týchto parametrov sa v tomto zozname vyskytuje ešte niekoľko ovládaní. Tieto ovládania zodpovedajú za pohyb myši, potvrdzovanie a zatváranie prvkov v aplikácií. Tieto prvky nedoporučujeme meniť.

5.1.2 Úvod do hry

Keď po konfigurácií hru spustíme, predstaví sa nám uvítacia scéna hry, ktorú môžeme vidieť na obrázku 5.4. Táto scéna nemá okrem privítania žiadny iný účel a hráč ju môže preskočiť stlačením akejkoľvek klávesy, či stlačením ľubovoľného tlačidla myši.

Obr. 5.4: Úvod do hry

Po opustení uvítacej scény sa hráč presunie do hlavného menu. Toto menu poskytuje hráčovi možnosť vytvoriť si nový hráčsky účet (1), prihlásiť sa k už vytvorenému hráčskemu účtu (2) alebo zobraziť informačný panel o tom, ako sa hrá ovláda (3). Tento panel je možné vidieť na obrázku 5.6.

Obr. 5.5: Hlavná scéna hry.

Ďalej si hráč môže zvoliť vidieť kto je autor tejto hry, všetkých testerov a modely, ktoré sme v hre použili (4), alebo hru jednoducho ukončí (5).

Všetky tieto možnosti a ich význam má hráč popísané v informačnom paneli (6).

Obr. 5.6: Info panel.

Credits

V prípade, že sa hráč rozhodne vidieť kto je autorom tejto hry, všetkých testerov a modely, ktoré sú v hre použité, presunie sa do novej scény, ktorá mu tieto informácie prehrá vo forme záverečných titulok. Túto scénu môžeme pozorovať na obrázku 5.7.

Túto scénu hráč opustí stlačením akéjkoľvek klávesy, či stlačením ľubovoľného tlačítda myši.

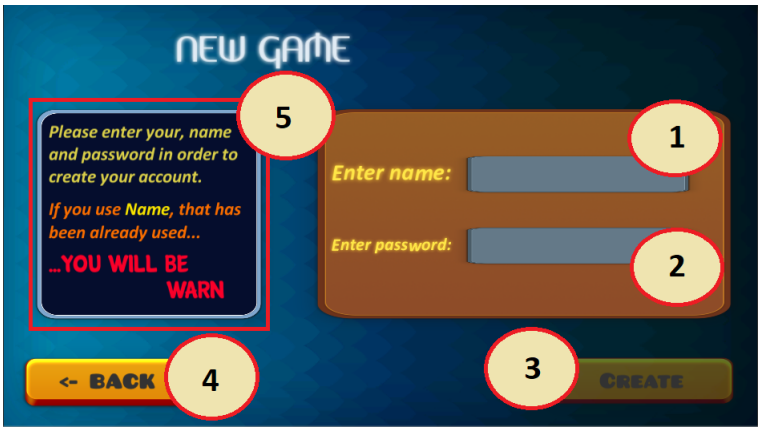
Obr. 5.7: Informácie o všetkých čo sa na hre podieľali

.

5.1.3 New Game

Ak sa hráč rozhodne vytvoriť si nový hráčsky účet, presunie sa do scény, ktorú môžeme vidieť na obrázku 5.8. Tu musí hráč vyplniť svoje užívateľské meno (1) a heslo (2) do textových polí.

Scéna poskytuje hráčovi možnosť vrátiť sa do hlavného menu (4) alebo vytvoriť účet a pokračovať (5). Toto tlačidlo je však, ako môžeme vidieť na obrázku, neaktívne. Informačný panel, ktorý je v tejto scéne prítomný (5), žiada hráča, aby vyplnil korektné registračné údaje.



Obr. 5.8: Vytváranie nového účtu.

V prípade, že meno a heslo sú pri registrácií korektné, tlačidlo na vytvorenie účtu sa odblokuje. Korektne vyplnené registračné údaje sú zobrazené na obrázku 5.9.

Obr. 5.9: Korektne vyplnené registračné údaje.

Chyby pri vytváraní účtu

Pod korektným účtom rozumieme prihlasovacie meno, ktoré je aspoň tri znaky dlhé a nie je použité iným hráčom, a heslo, ktoré je aspoň tri znaky dlhé. V prípade, že hráč pri registrovaní poruší nejaký z týchto požiadaviek, vyskočí naň bublina s upozornením na chybu, ktorú urobil. Tieto bubliny zmiznú, ak hráč klikne na textové pole, pri ktorom táto bublina vyskočila.

Možné chyby, ktoré môže hráč počas registrácie vykonať sú:

- *Krátke meno* – ukážka na obrázku 5.10.
- *Použité meno* – ukážka na obrázku 5.11.
- *Krátke heslo* – ukážka na obrázku 5.12.

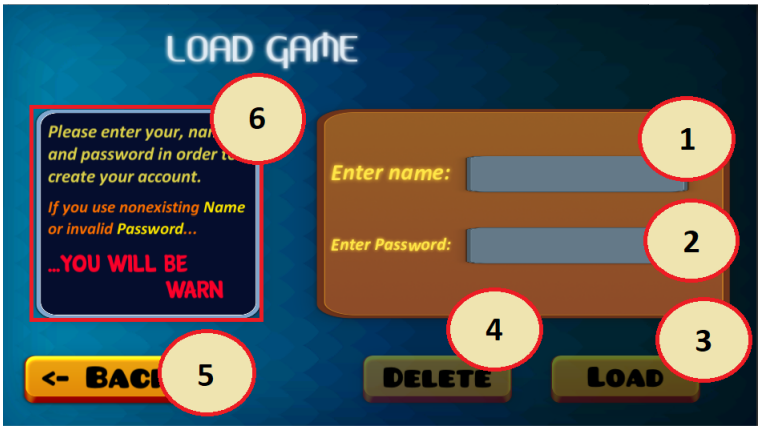
Obr. 5.10: Chyba – krátke užívateľské meno.

Obr. 5.11: Chyba – užívateľské meno je používané iným hráčom.

Obr. 5.12: Chyba – krátke heslo.

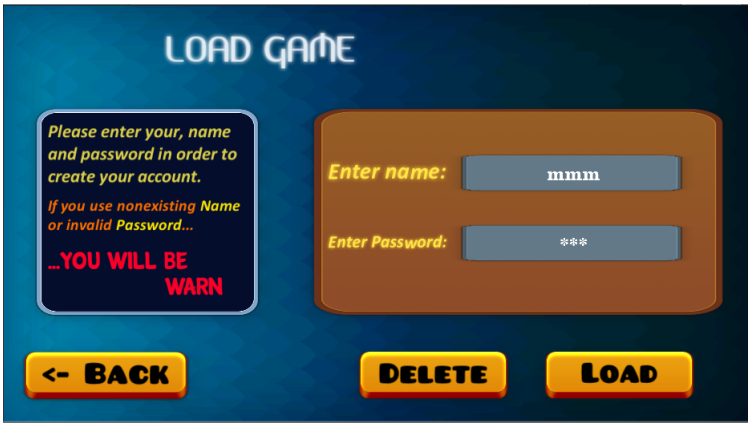
5.1.4 Load Game

Rovnako ako pri vyrváraní nových účtov, tak aj pri prihlasovaní sa do už existujúcich účtov sa hráč ocitne v takmer rovnako vyzerajúcej scéne, ktorú môžeme vidieť na obrázku 5.13. Tu hráč vyplní svoje užívateľské meno (1) a heslo (2). V prípade, že sú tieto údaje korektné, bude hráčovi umožnené sa do účtu prihlásiť (3) alebo účet vymazať (4). Obe tieto tlačidla sú v súčasnosti neaktívne, pretože textové polia neobsahujú žiaden text.



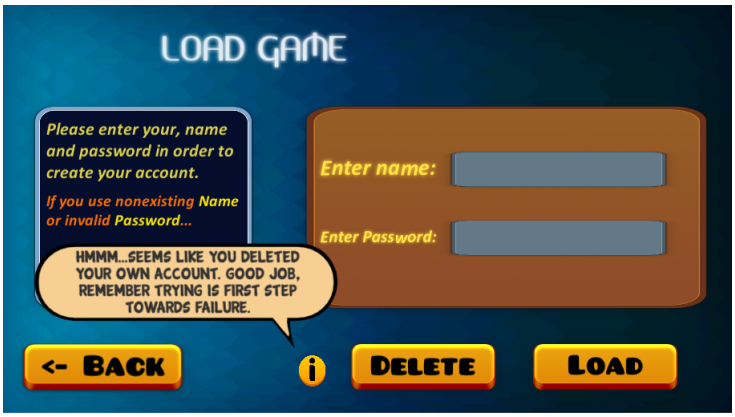
Obr. 5.13: Prihlásenie sa k existujúcemu účtu.

Hráča o nutnosti zadania korektných údajov informuje informačný panel (6). Samozrejme má hráč možnosť vrátiť sa do hlavného menu (5).



Obr. 5.14: Korektne vyplnené prihlasovacie údaje.

V prípade, že hráč uvedie korektné údaje, odblokujú sa mu možnosti zmazania účtu a prihlásenia sa k účtu ako môžeme vidieť na obrázku 5.14.



Obr. 5.15: Úspešné vymazanie účtu.

Ak sa hráč rozhodne pre zmazanie svojho účtu, vyskočí mu informačná bublina, ktorá ho informuje o tom, že jeho účet bol úspešne vymazaný. Túto udalosť

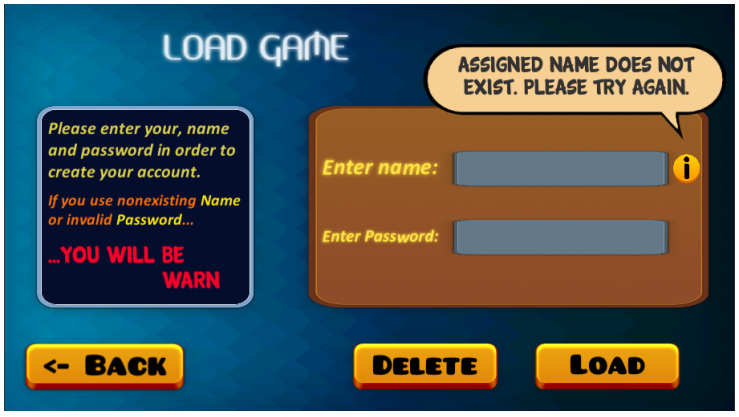
zachytáva obrázok 5.15.

Chyby pri prihlasovaní

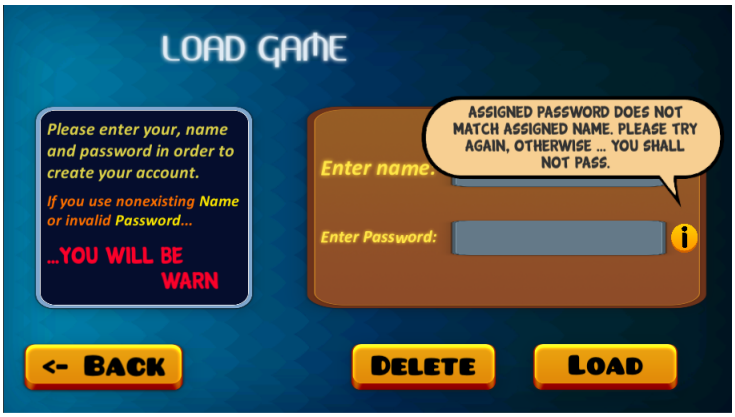
Rovnako ako v prípade vytvárania účtov, aj tu sa očakávajú korektné prihlasovacie údaje. Tieto údaje obsahujú platné prihlasovacie meno a heslo, ktoré patrí k danému účtu. V prípade, že hráč pri prihlasovaní poruší niektorú z týchto požiadaviek, vyskočí naň bublina s upozornením na chybu, ktorú počas prihlasovania urobil. Tieto bubliny zmiznú, ak hráč klikne na textové pole, pri ktorom táto bublina vyskočila.

Možné chyby, ktoré môže hráč počas registrácie vykonať sú:

- *Neexistujúce meno* – ukážka na obrázku 5.16.
- *Neplatné heslo meno* – ukážka na obrázku 5.17.



Obr. 5.16: Chyba – neexistujúce prihlasovacie meno.



Obr. 5.17: Chyba – neplatné heslo.

5.1.5 Henré Módy

Po vytvorení alebo prihlásení sa k účtu prejde hráč do scény, kde si vyberie herný mód, ktorý chce hrať.

Výzor tejto scény zachytáva obrázok 5.18, ktorý hráčovi dáva možnosť zvoliť si *Race Mode* (1), *Time Mode* (2), *K.O. Mode* (3) alebo dáva hráčovi možnosť vrátiť sa do hlavného menu (4).

Obr. 5.18: Výber herných módov.

Keďže je teraz hráč v hre prihlásený, môže vidieť svoje meno a bodový stav (5). Informácia o každom móde (6) sa zobrazí, ak hráč myškou vstúpi do oblasti tlačítka nejakého módu. Toto zobrazenie informácií ilustruje obrázok 5.19.

Obr. 5.19: Informácie o hernom móde.

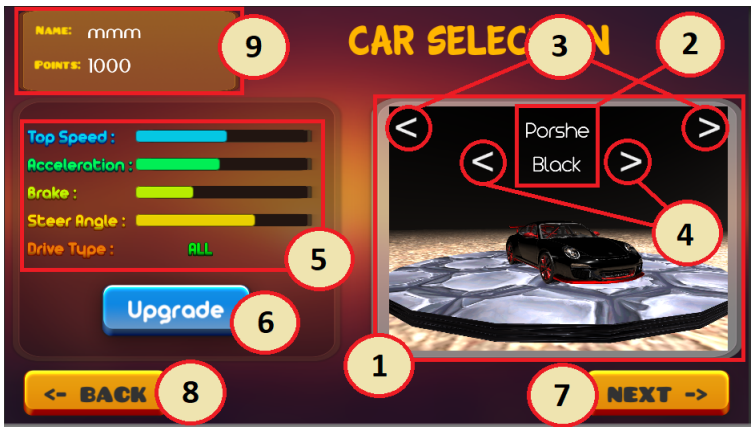
5.1.6 Výrer vozidiel

Bez ohľadu na to, aký herný mód hráč zvolí, ocitne sa v scéne, v ktorej si môže vyberať, odomykať a vylepšovať vozidlá. Táto scéna je znázornená na obrázku 5.20.

Na tomto obrázku môžeme vidieť panel, na ktorom je zobrazené vozidlo (1) a informácie o modely a farbe vozidla (2). V tomto panely môže hráč prechádzať modely vozidiel (3) a v prípade, že sú tieto modely odomknuté, tak si môže prechádzať aj farby jednotlivých vozidiel (4). Každé odomknuté vozidlo poskytne hráčovi prehľad o jeho parametroch (5). Tieto parametre obsahujú:

- informáciu o maximálnej rýchlosti vozidla
- informáciu o rýchlosti vozidla (počet koní)

- informáciu o sile brzdy vozidla
- informáciu o veľkosti maximálneho uhlu otáčania kolies vozidla
- informáciu o type náhonu vozidla



Obr. 5.20: Scéna výberu vozidiel.

Okrem toho každé odomknuté vozidlo poskytuje hráčovi možnosť toto vozidlo vylepšiť (5). Scéna, rovnako ako výber módov, obsahuje informácie o hráčovom mene a počtu jeho bodov (9). Záverom sa hráč z tejto scény môže vrátiť k výbere módov (8) alebo so zvoleným vozidlom pokračovať k výbere trate (7).

Odomykanie vozidla

V prípade, že narazíme na zamknuté vozidlo, ako môžeme vidieť na obrázku 5.21, panel s vozidlom sa zamkne a zmizne nám možnosť výberu farieb (3). Okrem toho sa nám zobrazí panel s informáciami o cene odomknutia vozidla (1) s možnosťou vozidlo odomknúť (2). Na koniec si môžeme všimnúť, že v prípade zamknutých vozidiel hráč nemá povolené pokračovať k výberu trate.

Obr. 5.21: Scéna výberu vozidiel so zamknutým vozidlom.

V prípade, že hráč nemá dostatok bodov na odomknutie vozidla, tak sa pri jeho pokuse o odomknutie objaví sčasti informatívna a sčasti výsmešná informačná

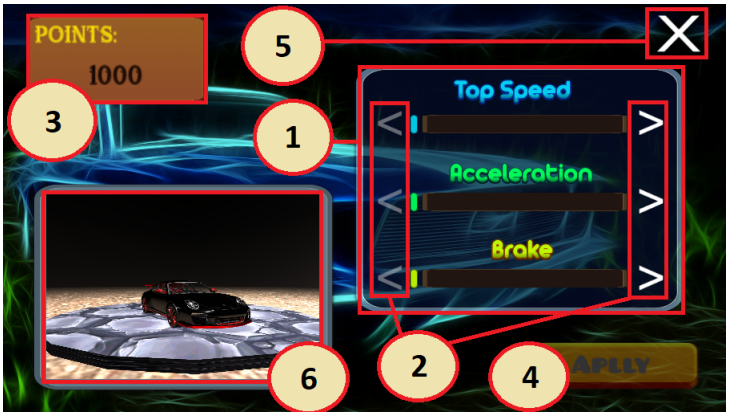
bublina, ktorá ho upozorní na to, že nemá dostatok bodov. Túto situáciu môžeme pozorovať na obrázku 5.22. Toto okno môže hráč jednoducho zavrieť (1).

Obr. 5.22: Neuspešné odomknutie vozidla.

Vylepšovanie vozidla

Ak si hráč bude chcieť svoje vozidlo vylepšiť, zobrazí sa mu panel, ktorý môžeme vidieť na obrázku 5.23. Tento panel obsahuje informácie o vlastnostiach vozidlá, ktoré môže hráč vylepšovať (1), a tlačidlá na vylepšovanie vozidla (2).

Počas tohto vylepšovania môže hráč sledovať svoj aktuálny bodový stav (3) a vozidlo, ktoré vylepšuje (6). Ak je s týmto vylepšením spokojný, môže ho aplikovať na vozidlo (4) alebo všetko zrušiť a vrátiť sa k výbere vozidla (5).



Obr. 5.23: Vylepšovanie vozidla.

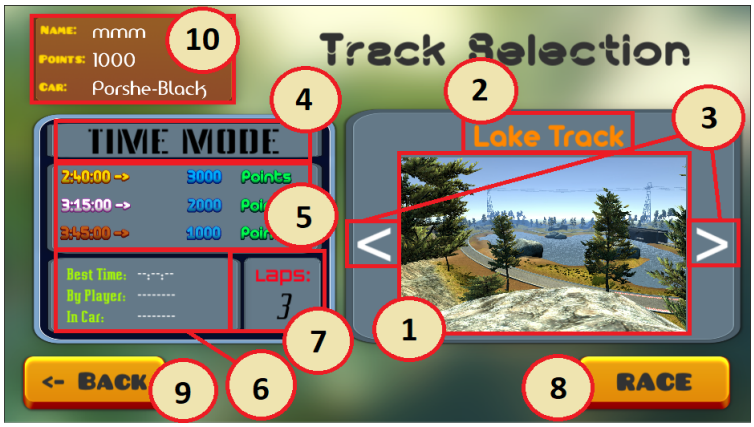
Na obrázku 5.24 môžeme vidieť, že ak vozidlo začneme vylepšovať, body sa nám začnú automaticky strhávať (1). Taktiež sa podľa zostávajúcich bodov hráča aktivujú, resp. deaktivujú tlačidlá na vylepšovanie jednotlivých vlastností vozidla na základe ceny vylepšenia daných vlastností.

Obr. 5.24: Odoberanie bodov počas vylepšovania.

5.1.7 Výber tratí

Ak si hráč vyberie vozidlo, s ktorým chce pretekať, ocitne sa v poslednej scéne herného menu a to je scéna výberu trate. Táto scéna obsahuje, ako môžeme vidieť na obrázku 5.25, panel s obrázkami (1) a menami (2) tratí, medzi ktorými môžeme prechádzať ľubovoľným smerom (3). Každá odomknutá trať obsahuje informačný panel, ktorý informuje užívateľa o zvolenom hernom móde (4) a pozícií/časoch/eliminácií, ktoré sú bodovo ohodnotené a ich bodovú hodnotu (5). Zároveň tento panel obsahuje informácie o počte kôl v danej trati (7) a časový rekord trate (6).

Hráčovi sa taktiež ponúka panel s jeho menom a počte bodov rozšírenom o informáciu, aké vozidlo a farbu si hráč zvolil (10). Záverom sa hráčovi ponúka možnosť vrátiť s k výberu vozidla (9) alebo, ak má zvolenú odomknutú trať, možnosť pretekať (8).



Obr. 5.25: Výber trate.

Samozrejme môže hráč naraziť na zamknutú trať. V takomto prípade, ako môžeme vidieť na obrázku 5.26, sa hráčovi zobrazí informačný panel, ktorý hráča informuje o počte bodov potrebných na odomknutie trate (2) a možnosť trať odomknúť (3). Môžeme si všimnúť, že panel zobrazujúci trať hráčovi oznamuje, že je trať zamknutá (1) a že tlačidlo pre začatie preteku je neaktívne.

Obr. 5.26: Zamknutá trať.

Hlásenia a upozornenia

Pri odomykaní trate môže nastať rovnaká situácia ako pri odomykaní vozidiel. Ak hráč nemá dostatok bodov, zobrazí sa mu informačná bublina, ktorá ho na to upozorní. Danú situáciu zachytáva obrázok 5.27. Bublinu môže hráč kedykoľvek vypnúť (1).

Obr. 5.27: Neúspešné odomknutie trate

Opačná situácia nastane, ak má hráč dostatok bodov na odomknutie trate. V tom prípade sa mu zobrazí informačná bublina, ktorá ho upozorní na úspešné odomknutie trate. Danú situáciu zachytáva obrázok 5.28. Bublinu môže hráč kedykoľvek vypnúť (1).

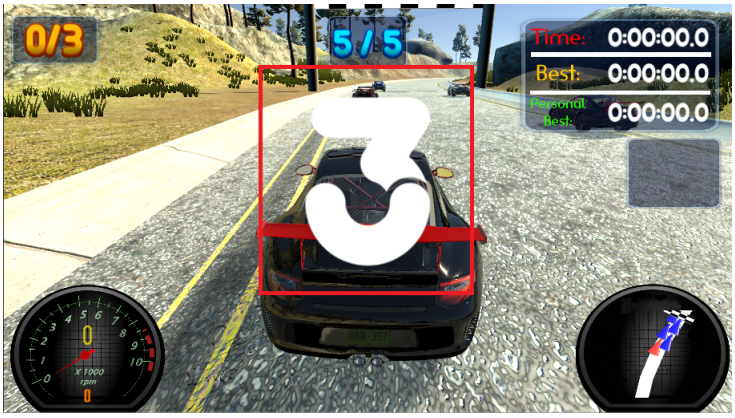
Obr. 5.28: Úspešné odomknutie trate

Nakoniec, ak si hráč zvolí K.O. móde, môže pri prechádzaní tratí naraziť na trať, ktorá nie je kruhová. V takom prípade si túto trať nemôže zvoliť, čo mu oznámi informačný panel. Situácia je zachytená na obrázku 5.29.

Obr. 5.29: Výber nekruhovej trate pri K.O Móde.

5.1.8 Pretek

Každý pretek sa začína odpočtom, počas ktorého hráč nemôže vozidlo ovládať.



Obr. 5.30: Odpočet pred začatím preteku.

Po skončení odpočtu hráč začína pretekať. Pri tom môže využiť všetky klávesy, ktoré sa mu zobrazovali v info panely hlavného menu na obrázku z5.6.

Počas preteku hráčovi poskytuje údaje o priebehu preteku niekoľko okien, ktoré môžeme vidieť na obrázku 5.31. Medzi tieto okná patrí okno oznamujúce aktuálne kolo, v ktorom sa hráč nachádza (1), a okno oznamujúce aktuálnu pozíciu hráča (2). Ďalej sa tu nachádza okno oznamujúce aktuálny čas, ktorý hráč v danom kole zatiaľ jazdí, časový rekord trate a osobný časový rekord hráča (3). Toto okno dopĺňa informácie o čase v každom kole, ktoré hráč prešiel (4).

Obr. 5.31: Začiatok preteku.

Záverom sa nám ponúka pohľad na minimapu (5) a tachometer (6) obsahujúci rýchlosť, otáčky a zaradený rýchlostný stupeň hráčovho vozidla.

Obr. 5.32: Použitie quick kamery.

V prípade, že hráč aktivuje quick kameru, naskytne sa mu pohľad na všetko, čo je za ním. Tento pohľad je zachytený obrázkom 5.32.

Pauza

V prípade, že hráč potrebuje aktivovať pauzu, otvorí sa mu panel, ktorý môžeme vidieť na obrázku 5.33. Tento panel hráčovi umožní vrátiť sa do preteku (1), spustiť celý pretek od znova (2), poprípade vrátiť sa do menu (3), čo hráča presunie do scény výberu módov. Hráč takisto môže počas pauzy vidieť informácie o mene a bodoch v jeho účte (5).

Obr. 5.33: Pauza počas preteku.

Hráčovi všetky tieto informácie poskytuje informačný panel (4), ktorý v prípade, že hráč vstúpi do oblasti nejakého tlačidla, zobrazí, čo sa týmto tlačítkom dá vykonať. To je zobrazené na obrázku 5.34.

Obr. 5.34: Informácie o jednotlivých možnostiach hráča.

5.1.9 Koniec Preteku

Po každom preteku sa hráčovi zobrazí popretekový panel, ktorý zachytáva obrázok 5.35. Tento panel umožňuje hráčovi zopakovať si pretek od začiatku (1), pozrieť si záznam o preteku (3) alebo sa vrátiť do menu hry, čo je v tomto prípade scéna výberu herného módu.

Obr. 5.35: Panel po ukončení preteku.

Panel hráčovi poskytuje pochvalný alebo výsmešný odkaz (4), spolu s informáciami o čase/umiestnení/eliminácií a počte bodov, ktoré za svoj výkon získal.

V prípade, že hráč myšou vojde do priestorov tlačidiel, zobrazia sa mu informácie o danom tlačidlu a čo sa kliknutím naň vykoná. Táto situácia je zachytená obrázkom 5.36.

Obr. 5.36: Informácia o možnostiach hráča.

Replay

Ak si hráč zvolí, že chce vidieť záznam o preteku, objaví sa mu panel, ktorý môžeme vidieť na obrázku 5.37. Tento panel obsahuje samotný záznam preteku (1), ovládací panel s možnosťou záznam pretočiť, prehrať od začiatku a zastaviť/spustiť (2). Ak hráč už nechce záznam vidieť, môže sa vrátiť do popretekového panelu (3).

Obr. 5.37: Replay panel.

5.2 Dizajnérska

Táto časť bude venovaná dizajnérom a programátorom, ktorý by túto hru chceli v budúcnosti rozširovať. Zameriame sa v nej na pridávanie vozidiel, tratí a povieme si, ako celú hru sfunkčniť.

Pre uľahčenie pridávania vozidiel a tratí do našej hry sme v *Unity* vytvorili editory, ktoré nám uľahčia prácu. V nasledujúcich častiach si popíšeme prácu s týmito editormi.

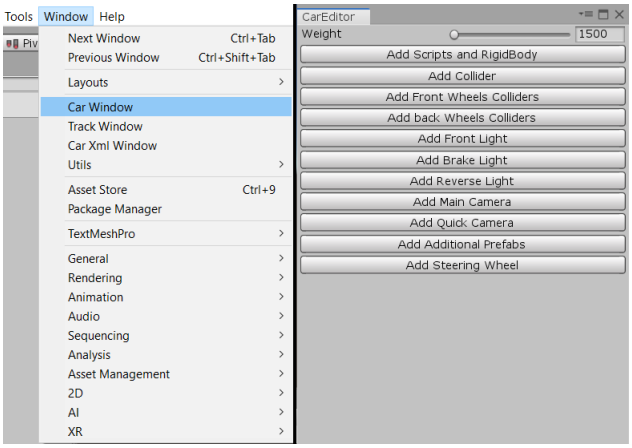
5.2.1 Tutoriál na pridanie vozidla

Začneme pridaním vozidla do nášho projektu. Každé vozidlo, ktoré do tohto projektu importujeme by malo byť vo forme **prefab**. Na obrázku 5.38 vidíme príklad vozidla vo forme **prefab**, ktoré je zobrazené v prehľadávači *Unity*.

Obr. 5.38: Importovaný prefab v priehľadači *Unity*.

Tento **prefab** obsahuje objekty kolies (2) a **Collider** (1). Tieto objekty budú dôležité o niečo neskôr.

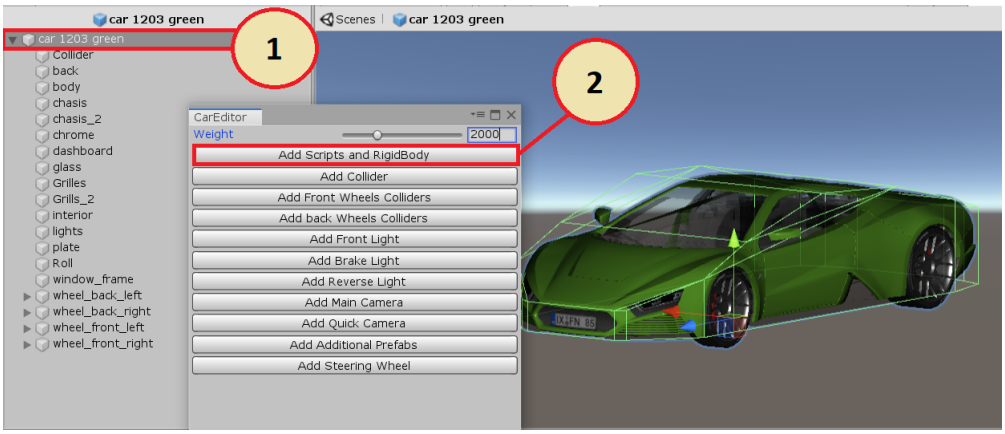
Teraz, keď máme importovaný **prefab** vozidla, otvoríme si editor na vyplnenie vozidla všetkými objektami, skriptami a inými prvkami, ktoré pre korektne vozidla v našej hre potrebujeme. Názov tohto editoru je *Car Window* a je možné k nemu pristúpiť cez **Window** → **Car Window**. Obrázok 5.39 ilustruje na ľavej strane túto cestu a na pravej strane zobrazuje samotný editor. Počas tohto návodu si prejdeme všetky jeho vlastnosti.



Obr. 5.39: Prístup k editoru vozidla.

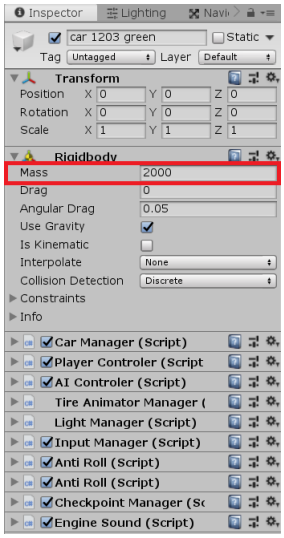
Pridanie Skriptov a RigidBody

Začiatok práce s editorom nám zobrazuje obrázok 5.40. Na ňom vidíme, že sme si označili koreň **prefabu** vozidla (1), potom sme si v editore nastavili slidier **Wheight** na váhu 2000kg a stlačili sme tlačidlo na pridanie skriptov a **RigidBody** (2).



Obr. 5.40: Nastavenie váhy a pridanie skriptov.

Po tejto akcií sa do **prefabu** vloží **RigidBody** a všetky skripty, ktoré vozidlo potrebuje. Môžeme to vidieť na obrázku 5.41. Na tomto obrázku môžeme vidieť, že **Mass** (váha) vozidla sa nastavila na 2000kg.



Obr. 5.41: Inšpektor **prefabu** po pridaní skriptov a rigidbody.

V skripte **AIControler** musíme nastaviť verejné premenné. Konkrétne sa jedná o:

- *Front sensor position* – vzdialenosť senzorov od stredu vozidla.
- *Front side sensor position offset* – vzdialenosť krajných senzorov od stredného senzora.
- *Front side sensor angle* – uhol krajných senzorov.

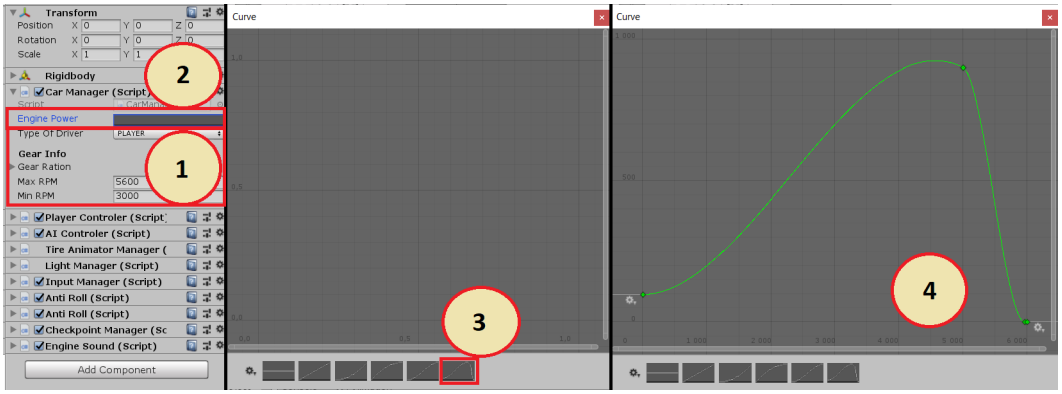
Tieto premenné môžeme vidieť na obrázku 5.42.

Obr. 5.42: Inšpektor **prefabu** po pridaní skriptov a rigidbody.

Musíme nastaviť verejné premenné aj v skripte **CarManager**. Jedná sa o:

- *Type Of Driver* – informácia, kto vozidlo ovláda. Používa sa na ladiace účely. Obsahuje hodnoty AI, PLAYER a GHOST.
- *GearRation* – pole reprezentujúce počet rýchlostných stupňov vozidla. Nastavujú sa doň čísla v klesajúcom poradí. Teda číslo v prvom políčku by malo byť väčšie než druhé atď.
- *MaxRPM/MinRPM* – otáčky, kedy preraďujeme/podraďujeme rýchlostny stupeň vozidla.

Všetky tieto premenné môžeme vidieť na obrázku 5.43, ktorý obsahuje všetky premenné, ktoré sme spomenuli (1). Okrem toho sa tu nachádza aj premenná *Engine Power* (2), ktorá reprezentuje výkonnostnú krivku motora. Po kliknutí na túto premennú si zvolíme nami predmodelovanú krivku (3), ktorá sa po tom vykreslí (4).

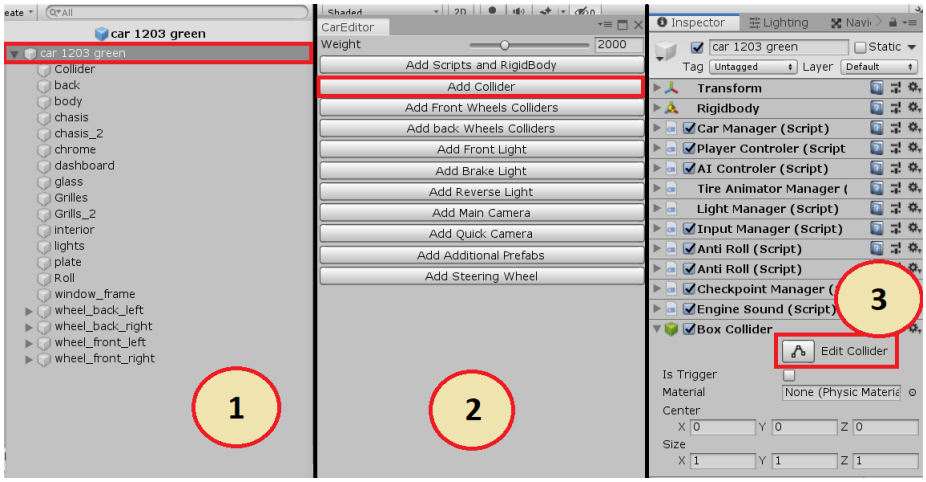


Obr. 5.43: Nastavenie Engine Power.

Na záver musíme nastaviť verejné premenné aj v skripte **EngineSound**. Do týchto premenných sa musia priradiť ľubovoľné zvukové stopy vozidla. V našom prípade používame audio objekty z assetu **Rotary x8 - FREE Engine Sound Pack**. Konkrétne sa jedná o stopy *high_on, high_off, low_on, low_off*.

Pridanie a nastavenie Collidera

Prefab, ktorý používame ako príklad, obsahuje **Collider**, no môže sa stať, že **prefab** vozidla nebude obsahovať žiadny **Collider**. V takomto prípade opäť označíme koreň **prefabu** vozidla a stlačíme tlačidlo na pridanie **Collidera**. Táto akcia pridá do vozidla **Box Collider**.



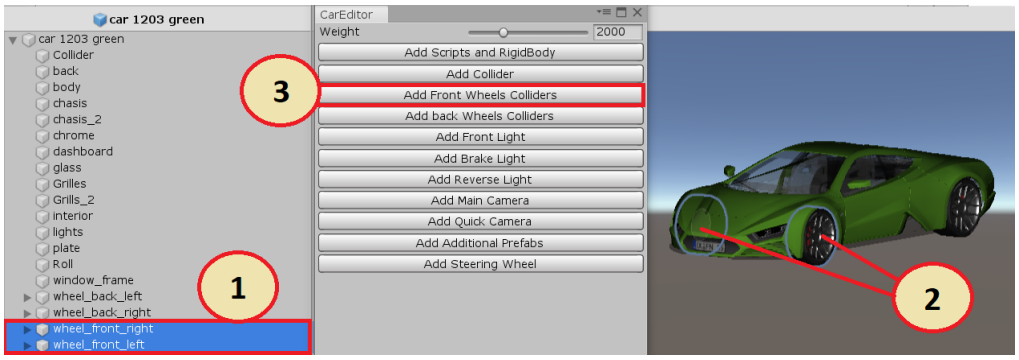
Obr. 5.44: Pridanie a úprava *Box Collidera*.

Obrázok 5.44 zachytáva tento proces označenia koreňa (1) a pridanie **Box Collidera** (2). Tento **Box Collider** však nebude mať veľkosť a rozmery vozidla,

preto je nutné tento **Box Collider** upraviť (3) tak, aby čo možno najviac odpovedal veľkosti, šírke a výške vozidla.

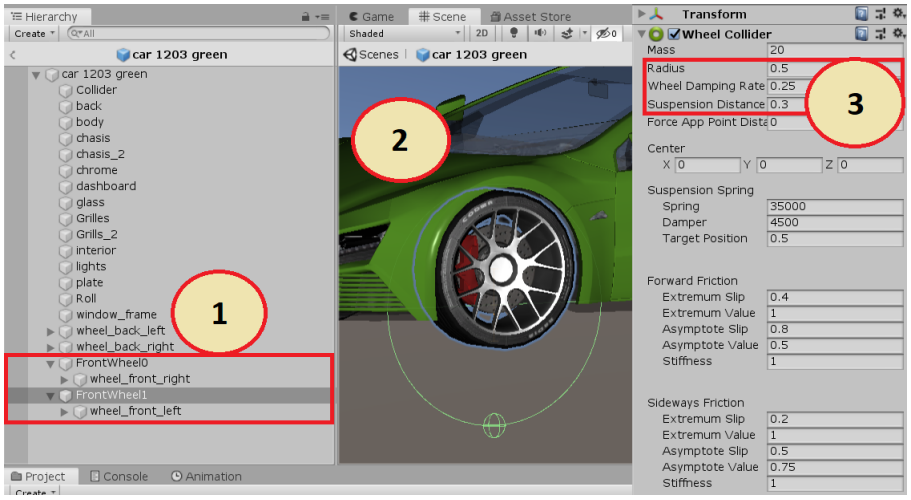
Pridanie a nastavenie Wheel Colliderov

Teraz musíme zaistiť, aby malo vozidlo funkčné kolesá. Preto mu musíme pridať **Wheel Collider** do každého objektu kolesa. Obrázok 5.45 nám ukazuje, že musíme označiť kolesá, do ktorých chceme pridať **Wheel Collider** (1). Musíme si dávať pozor, aby sme vždy označili buď predný alebo zadný pár kolies. Pomôckou nám bude prehliadač *Unity*, ktorý tieto kolesá na vozidle vyznačí (2). Na záver pridáme **Wheel Collider-e** do vozidla (3). V tejto ukážke sme pridávali predné kolesá. Zadné kolesá sa pridávajú úplne rovnako (**Add Back Wheel Colliders**).



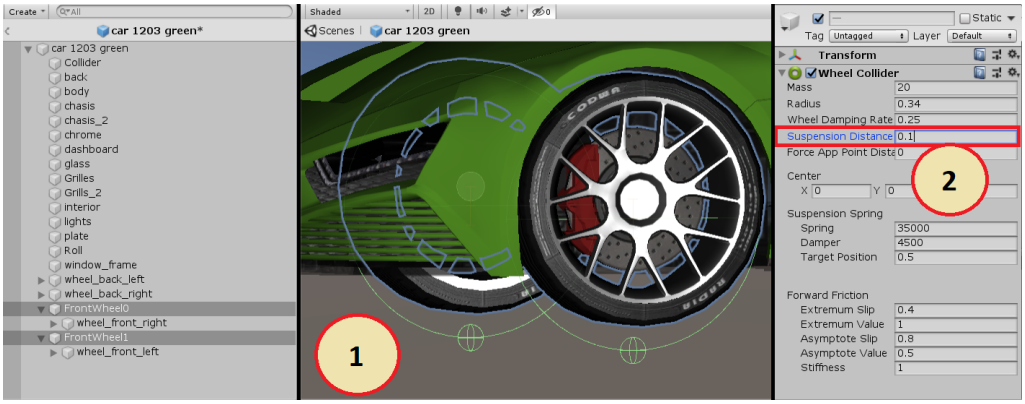
Obr. 5.45: Pridanie *Wheel Colliderov* do vybraných kolies.

Na obrázku 5.46 môžeme vidieť, ako nám táto akcia zmenila štruktúru objektov v **prefabe** vozidla. Objekty predných kolies získali nových rodičov (1), z ktorých každý obsahuje **Wheel Collider**. Objekty kolies získali nový skript, ale to bude relevantné až v sekcií 5.2.1. Vidíme, že **Wheel Collider** je väčší ako objekt kolesa (2), preto musíme nastaviť jeho rozmery tak, aby presne kopíroval polomer kolesa (3). Ak bude **Wheel Collider** trocha väčší než samotné koleso, tak nám to v tomto prípade nevadí.



Obr. 5.46: Zmena štruktúry *prefabu* a nastavenie *Wheel Colliderov*.

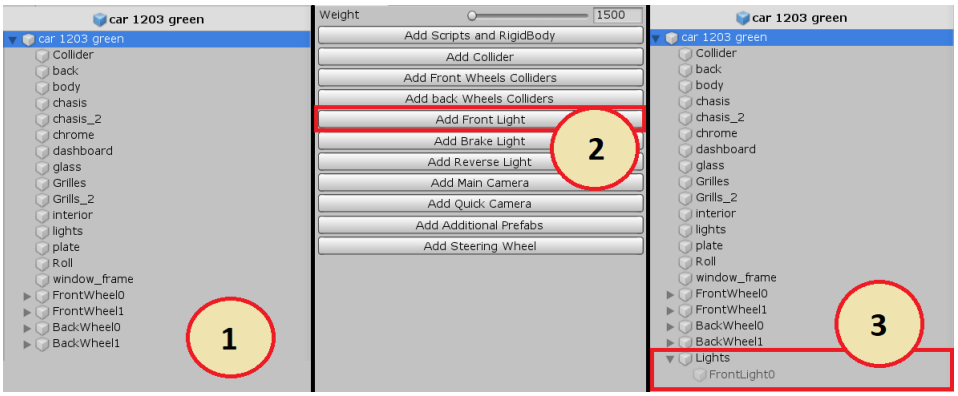
Na obrázku 5.47 máme zobrazené korektné nastavenie **Wheel Colliderov**, ktoré rozmerovo presne odpovedajú rozmeru kolies. Môžeme si ale všimnúť, že samotné **Wheel Collidere** sú umiestnené nižšie než koleso (1). Je to z toho dôvodu, že sme premennú **Suspension Distance** (2) nastavili na hodnotu, ktorá je väčšia ako 0. Táto premenná simuluje vo **Wheel Collider-i** pérovanie kolesa, takže čím väčšia je premenná, tým väčšie pérovanie vozidlo má.



Obr. 5.47: Správne nastavenie *Wheel Colliderov*.

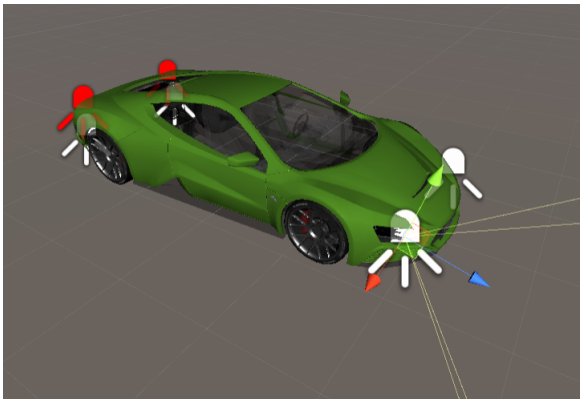
Pridanie svetiel

Ak chceme do vozidla pridať svetlá, musíme začať prednými svetlami. Na obrázku 5.48 je znázornené pridanie predných svetiel. Najprv si označíme koreň **prefabu** (1) a potom pridáme predné svetlá (2). Táto akcia vytvorí v **prefabe** objekt *Lights*, ktorý bude obsahovať jedno predné svetlo (3). Každé ďalšie pridanie predného svetla (2) pridá do objektu *Lights* ďalšie svetlo.



Obr. 5.48: Pridanie predných svetiel.

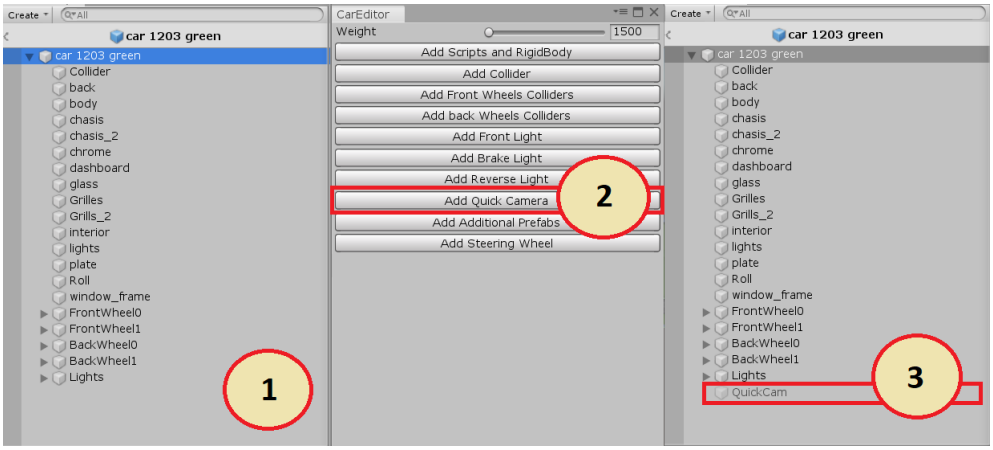
Všetky tieto svetlá musia byť ručne nastavené na pozíciu svetiel na vozidle. Všetky ostatné druhy svetiel sa budú tak isto pridávať do objektu *Lights* a rovnako tak je nutné presunúť ich na korektné miesto. Konečné umiestnenie svetiel môže vyzerať tak, ako to ilustruje obrázok 5.49.



Obr. 5.49: Ukážka umiestnenia svetiel.

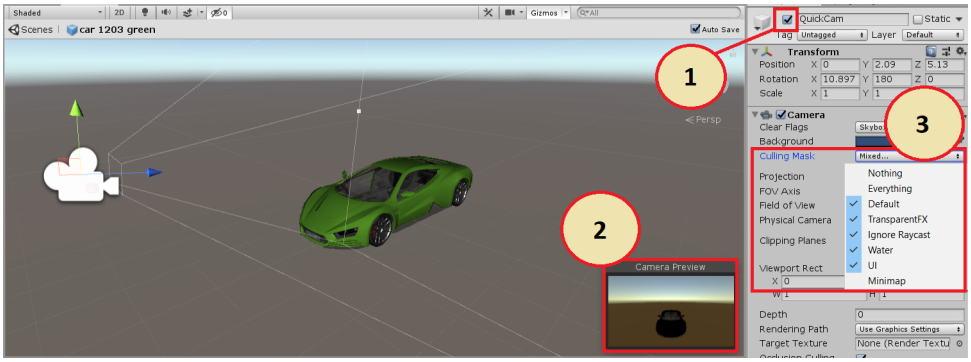
Pridanie quick kamery

Pridanie quick kamery funguje veľmi podobne ako pridanie svetiel. Na obrázku 5.50 vidíme, že si musíme označiť koreň **prefabu** (1) a pridať quick kameru (2). Do **prefabu** nám tak pribudne quick kamera (3).



Obr. 5.50: Pridanie quick kamery.

Túto kameru je nutné nastaviť tak, aby korektne zachytávala vozidlo a všetko, čo je za ním. Obrázok 5.51 zachytáva korektné nastavenie a pozíciu kamery. Kameru sme si v tomto obrázku aktivovali (1), na základe čoho sa nám zobrazil obraz, ktorý kamera zachytáva (2). Vďaka tomu môžeme kameru nastaviť na korektné miesto.

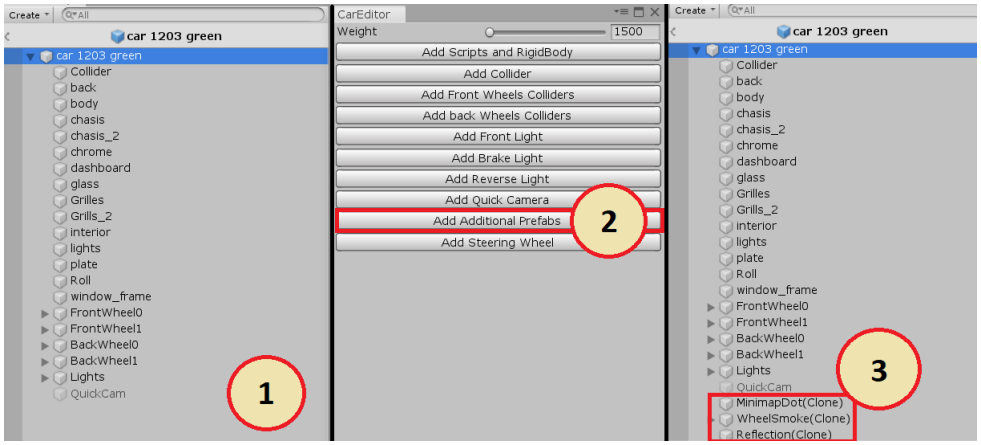


Obr. 5.51: Korektné nastavenie quick kamery.

Nesmieme zabudnúť na vypnutie vrstvy *Minimap* z našej kamery, aby nezachytávala to, čo zachytáva minimapa (3). Túto vrstvu vypneme v premennej **Culling Mask**. Po nastavení quick kamery kameru opäť deaktivujeme (1).

Pridanie dymu, šípky pre minimapu a odlesku

Nakoniec do vozidla pridáme dym, šípku na minimape a odlesk. Na obrázku 5.52 vidíme, že musíme označiť koreň **prefabu** (1) a pridať požadované objekty (2).

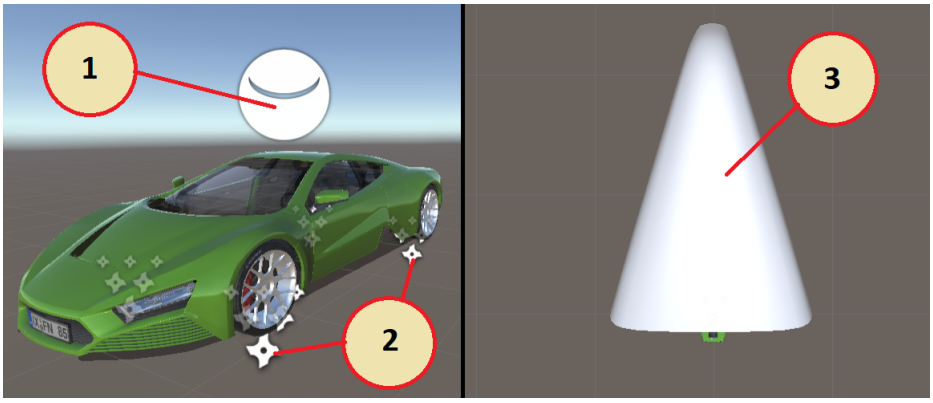


Obr. 5.52: Pridanie dymu, šípky na minimape a odlesku.

To nám do **prefabu** pridá objekty:

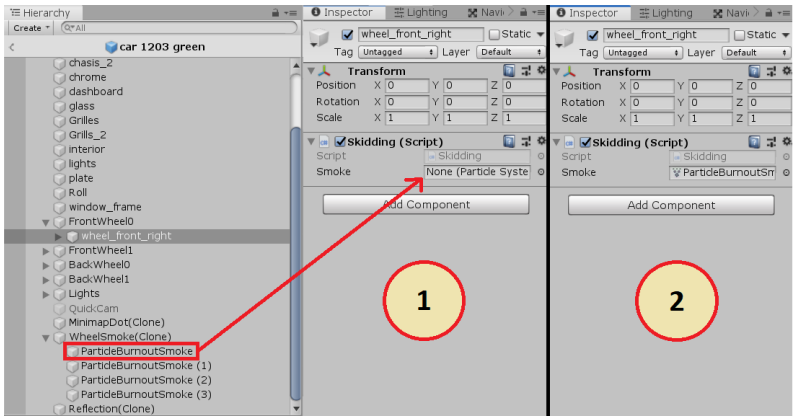
- *MinimapDot* – šípka na minimape.
- *WheelSmoke* – objekt, obsahujúci dym pre každé koleso.
- *Reflection* – odlesk vozidla.

Tieto objekty nie je nutné nastavovať, pretože sa vytvárajú na vopred predpočítaných pozíciách aj keď je možné, že v prípade *WheelSmoke* bude nutné jednotlivé dymové objekty trocha posunúť tak, aby boli pod pneumatikami vozidla. Na obrázku 5.53 vidíme korektné umiestnenie odlesku (1), dymu (2) a šípky na minimape (3).



Obr. 5.53: Korektné nastavenie.

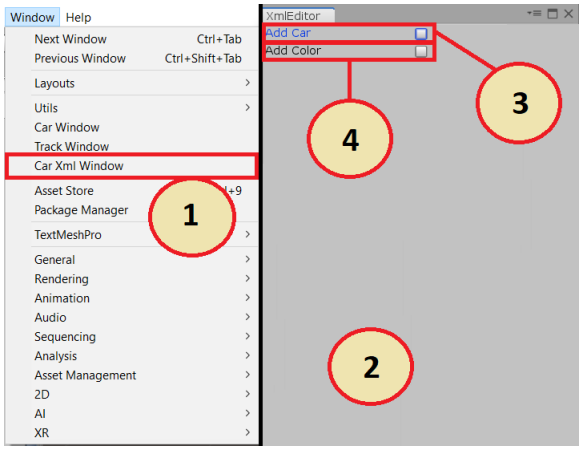
V sekcií 5.2.1 sme spomenuli, že každý objekt kolesa dostal skript. Tento skript obsahuje premennú *Smoke*, do ktorej je nutné umiestniť objekt dymu, ktorý sa pod týmto objektom kolesa nachádza. Obrázok 5.54 ilustruje pridanie korektného objektu do premennej (1) a ako vyzerá umiestnenie objektu v premennej (2).



Obr. 5.54: Pridanie objektu dymu do skriptu kolesa.

5.2.2 Registrácia vozidla v XML

Keď sme do vozidla vložili všetky objekty a skripty, musíme vozidlo ešte zaregistrovať do našej XML databázy vozidiel. Túto registráciu vykonáme pomocou ďalšieho editora. Názov tohto editoru je *Car Xml Window* a je možné k nemu pristúpiť cez **Window** → **Car Xml Window**. Na obrázku 5.55 môžeme vidieť ako pristúpiť k XML editoru vozidla (1) a ako vyzerá samotný editor (2).

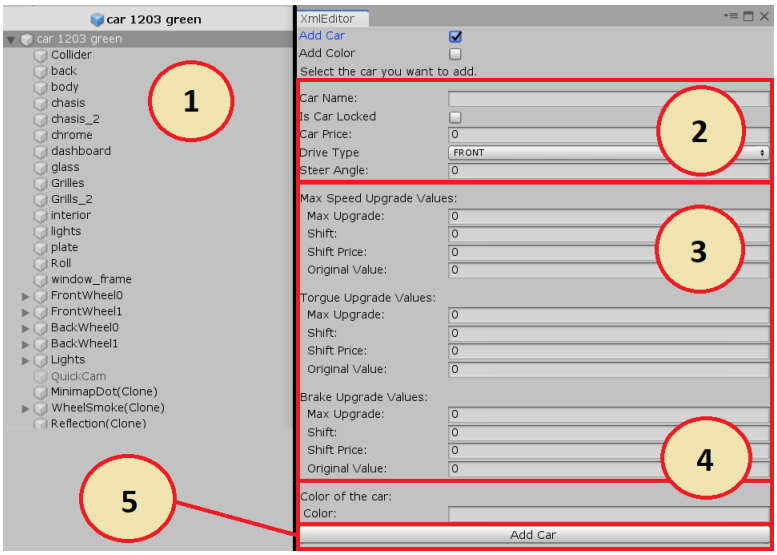


Obr. 5.55: Cesta k editoru a ukážka editora.

Vidíme, že editor obsahuje možnosť pridať vozidlo (3) alebo farbu vozidla (4). Postupne si obe tieto možnosti vysvetlíme.

Pridanie vozidla v XML

Obrázok 5.56 popisuje spôsob pridania vozidla do XML databázy. Pri registrácií musíme označiť koreň **prefabu**, ktorý chceme registrovať (1). Následne sa nám v editore objavia údaje, ktoré musíme vyplniť.



Obr. 5.56: Registrácia vozidla do XML databázy.

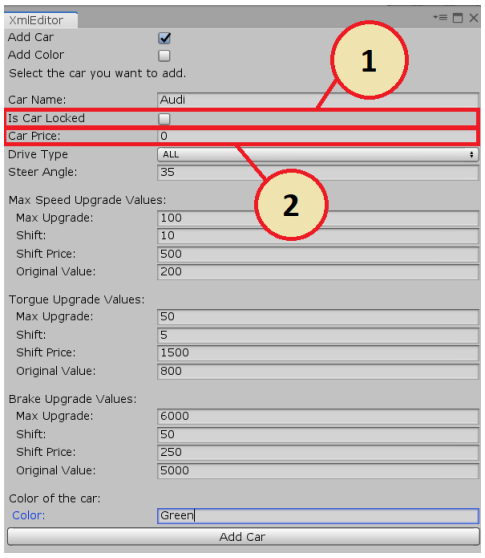
Údaje môžeme rozdeliť do troch častí: Základné údaje o vozidle (2), údaje o vylepšiteľných parametroch vozidla (3) a údaj o farbe vozidla (4). V prípade, že sa tieto údaje vyplnili správne, vozidlo zaregistrujeme do XML databázy (5).

Tieto údaje obsahujú:

- *Car Name* – meno vozidla (očakáva text).
- *Is Car Locked* – informácia o tom, či je vozidlo odomknuté.
- *Car Price* – cena vozidla, za ktorú ho hráč môže odomknúť (očakáva číslo).

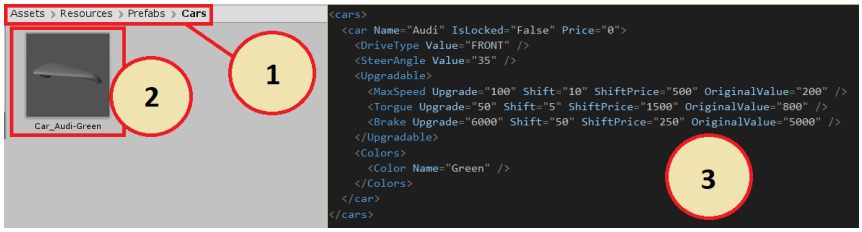
- *Drive Type* – typ náhonu vozidla. Obsahuje tri možnosti:
  - *FRONT* – náhon predných kolies.
  - *BACK* – náhon zadných kolies.
  - *ALL* – náhon všetkých kolies.
- *Steer Angle* – maximálny uhol otáčania vozidla (očakáva číslo).
- *Max Speed, Torgue, Brake* – maximálna rýchlosť, výkon a sila bŕzd u vozidla. Keďže sa všetky tieto parametre dajú vylepšovať, obsahujú dodatočné údaje o:
  - *Max Upgrade* – maximálne možné vylepšenie vozidla (očakáva číslo).
  - *Shift* – množstvo, o ktoré sa vozidlo vylepší pri jednom kliku vylepšenia vozidla (očakáva číslo).
  - *Shift Price* – cena za jedno vylepšenie (očakáva číslo).
  - *Original Value* – pôvodná hodnota, na ktorej vozidlo začína (očakáva číslo).
- *Car Color* – farba vozidla.

Príklad korektne vyplnených údajov môžeme vidieť na obrázku 5.57. Môžeme si všimnúť, že toto vozidlo je odomknuté (1), teda ho hráč bude mať odomknuté v scéne výberu vozidiel. Z toho dôvodu nie je nutné udávať cenu vozidla (2).



Obr. 5.57: Korektné vyplnené XML údaje pre vozidlo.

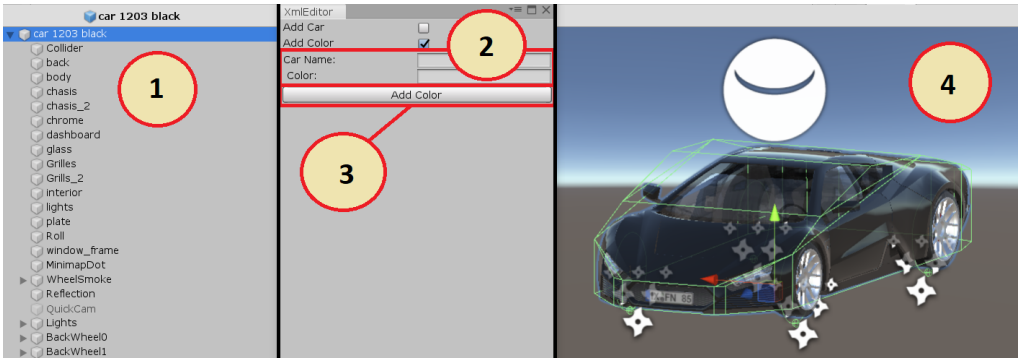
Po vyplnení týchto údajov, ako môžeme vidieť na obrázku 5.58, sa vozidlo pridá vo XML databázy vozidiel (3) a samotný **prefab** (2) sa nám presunie do zložky, kde si budeme držať všetky naše vozildá (1).



Obr. 5.58: Vozidlo sa presunulo do našej zložky a pridalo sa do databázy.

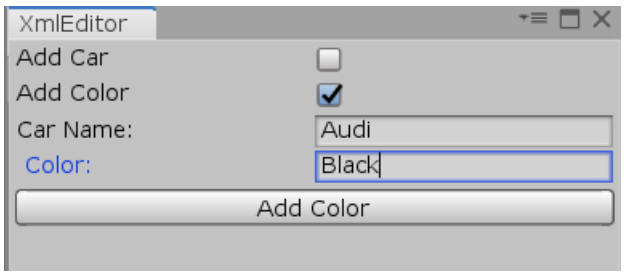
Pridanie novej farby vozidla v XML

Teraz si ukážeme, ako pridávať farby vozidiel do hry. Tento proces popisuje obrázok 5.59. Na ňom môžeme vidieť, že sme si vytvorili čierny model vozidlá (4), ktoré sme si do XML databázy vozidiel pridali v sekcií 5.2.2. Na pridanie farby vozidla opäť označíme koreň **prefabu** vozidla (1). Korektne vyplníme údaje o pridávanom vozidle (2) a pridáme vozidlo do databázy XML (3).



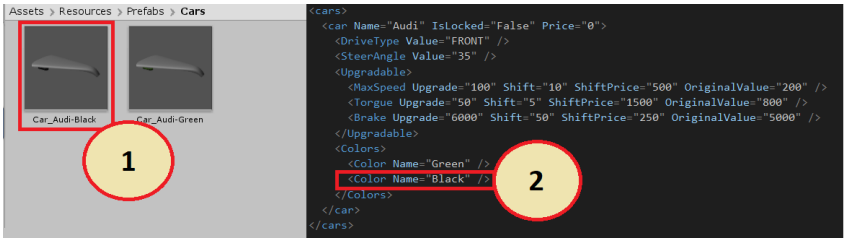
Obr. 5.59: Pridanie novej farby vozidla do databázi XML.

Údaje, ktoré musíme v tomto prípade vyplniť sú meno a farba vozidla. Obrázok 5.60 zobrazuje korektné vyplnenie týchto údajov. Musíme si dať pozor na to, aby bolo meno vozidla, ktoré chceme pridať, v databáze už pridané a aby farba, ktorú pridávame, nebola prítomná v databáze vozidla. Inak nastane chyba.



Obr. 5.60: Korektné vyplnené údaje na pridanie farby vozidla.

Ak pridáme novú farbu do databázy vozidiel XML, môžeme na obrázku 5.61 vidieť, že sa nám nový model uložil do priečinku s vozidlami (1) a že sa nová farba pridala do XML štruktúry (2).



Obr. 5.61: Pridanie nového vozidla do priečinku s vozidlami a pridanie novej farby vozidla do databázi XML.

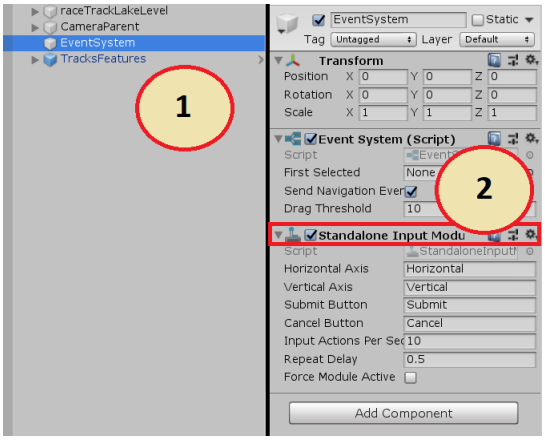
Obrázok 5.62 zachytáva časť scény hry na výber vozidiel, na ktorom môžeme vidieť, že obe vozidlá boli úspešne pridané do hry.

Obr. 5.62: Ukážka pridaných vozidiel pri výbere vozidiel v hre.

Je nutné si dať pozor, aby vozidlá svojou veľkosťou odpovedali veľkosti ostatných vozidiel v hre. Túto veľkosť je možné meniť v inšpektorovi vozidla - v premennej **Scale** objektu **Transform**.

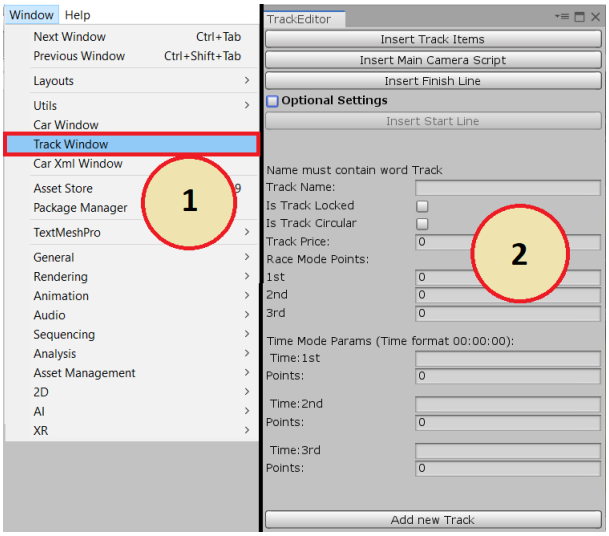
5.2.3 Tutoriál na pridanie trate

Ďalším veľkým prvkom, ktorý chceme byť schopný do projektu pridávať sú trate. Pod traťou rozumieme nejakú scénu, ktorá obsahuje aspoň trať. Začneme tým, že sa uistíme, že scéna obsahuje objekt **EventSystem**. Obrázok 5.63 nám ukazuje že do tohto prázdneho objektu (1) musíme vložiť komponentu **Standalone Input Module** (2). Ak to urobíme, tak potom komponentu **Event System** získame spolu s ňou. Tento objekt je nutný k tomu, aby bola hráč v danej scéne schopný kontrolovať tlačítka.



Obr. 5.63: Pridanie objektu *EventSystem* do trate.

Na pridávanie tratí do nášho projektu slúži editor **Track Window**, ktorý je možné nájsť v **Window** → **Track Window**. Obrázok 5.64 ilustruje cestu k editoru (1) a výzor editora (2).

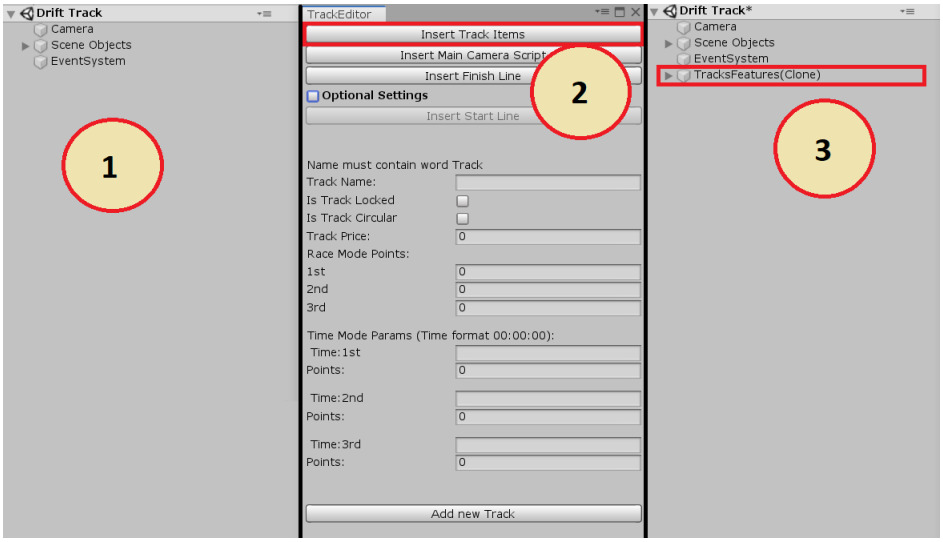


Obr. 5.64: Pohola a výzor editoru na pridávanie tratí.

Na začiatok presunieme scénu trate, ktorú chceme do hry pridať, do zložky **Assets/Scenes/Tracks**, v ktorej si budeme uchovávať všetky trate, ktoré sa v hre budú vyskytovať. Toto presunutie nie je povinné, ale uľahčí nám to prácu so scénami a je to priehľadnejšie ako mať scény v rôznych priečinkoch. Scéna, ktorú ideme do hry pridať, sa volá **Drift Track** - bude dôležité neskôr.

Pridanie potrebných objektov

Začneme pridaním všetkých dôležitých objektov, ktoré sa v trati majú vyskytovať. Túto akciu zachytáva obrázok 5.65, v ktorom môžeme vidieť objektovú štruktúru scény pred pridaním objektov (1) a po pridaní (3). Samotné pridanie objektov (2) nevyžaduje vyznačenie nejakého objektu (skript pridá objekt do scény).

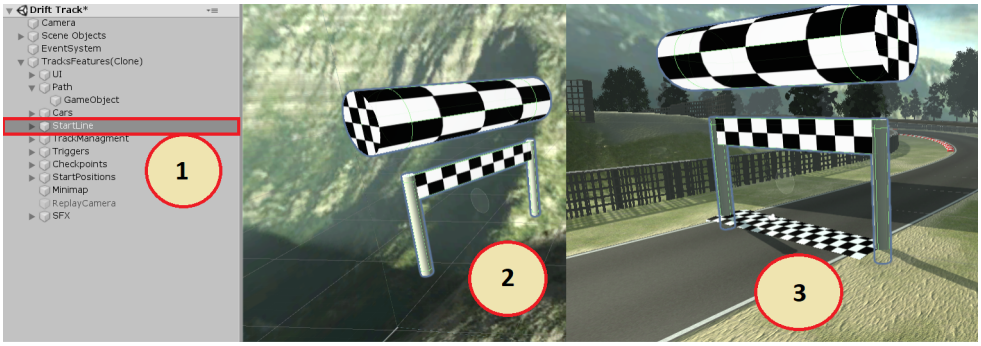


Obr. 5.65: Pridanie objektu *Track Features* do trate.

Objekt *Track Features*, ktorý sme pomocou editora pridali, obsahuje všetky objekty, ktoré pre trať potrebujeme. Opis týchto objektov zachytáva obrázok 4.3.

Nastavenie štartu

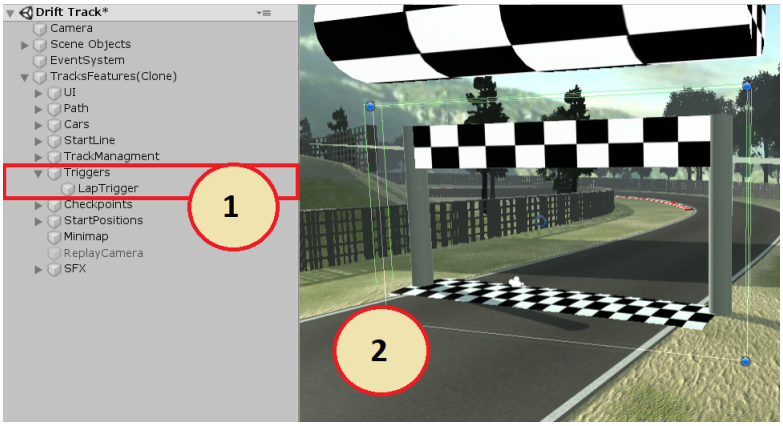
Teraz, keď máme všetky objekty k dispozícií, musíme ich správne nastaviť a umiestniť. Začneme umiestnením objektu, ktorý označuje začiatok preteku (v prípade kruhovej trate označuje aj koniec).



Obr. 5.66: Umiestnenie začiatku preteku.

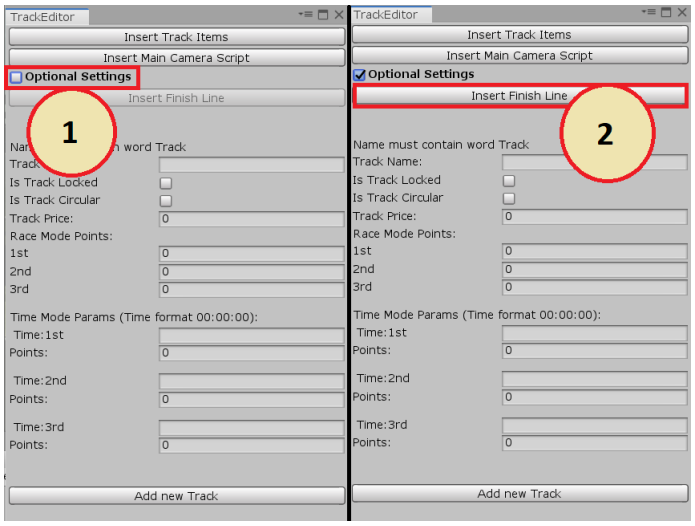
Obrázok 5.66 nám zobrazuje, že štartovacia pozícia preteku (1) bola po pridaní do scény umiestnená úplne mimo trať (2). Musíme ju teda premiestniť na miesto na trati, ktoré nám bude vyhovovať ako začiatok preteku (3).

Ak sa jedná o kruhovú trať (čo je v tomto prípade), predstavuje táto štartovacia pozícia zároveň koncovú pozíciu preteku. Na obrázku 5.67 je zobrazené presunutie senzora **LabTrigger** (1) na rovnakú pozíciu ako je štartovacia pozícia preteku (2).



Obr. 5.67: Presunutie senzora na meranie kôl.

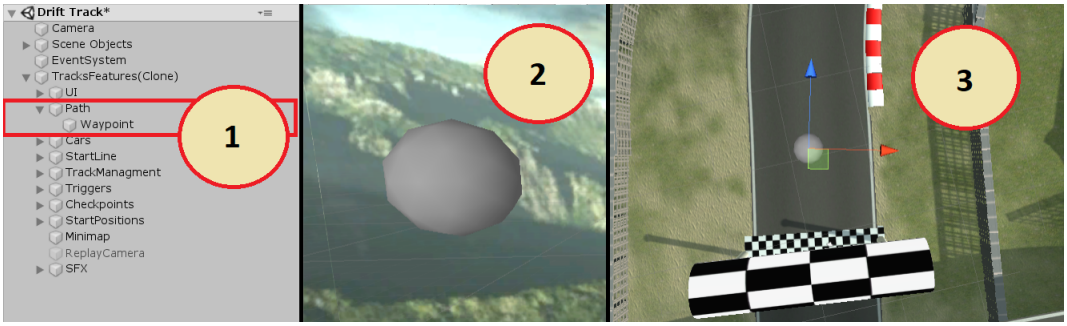
V prípade, že nie je trať kruhová, je nutné vykonať akciu, ktorú popisuje obrázok 5.68. V ňom aktivujeme možnosť editora pridať koncovú pozíciu (1) a následne ju pridáme (2). V takomto prípade by sa **LapTrigger** presunul na pozíciu tejto koncovej pozície.



Obr. 5.68: Pridanie koncovéj pozície do nekruhovej trate.

Vytvorenie waypointov

Umiestnenie waypointov bude určovať primárnu cestu, po ktorej sa bude AI pohybovať. Ako môžeme vidieť na obrázku 5.69, objekt **Path** obsahuje na začiatku len jeden waypoint (1), ktorý je úplne mimo trať (2). Prvý waypoint umiestnime pred štartovaciu pozíciu trate v smere jazdy (3).

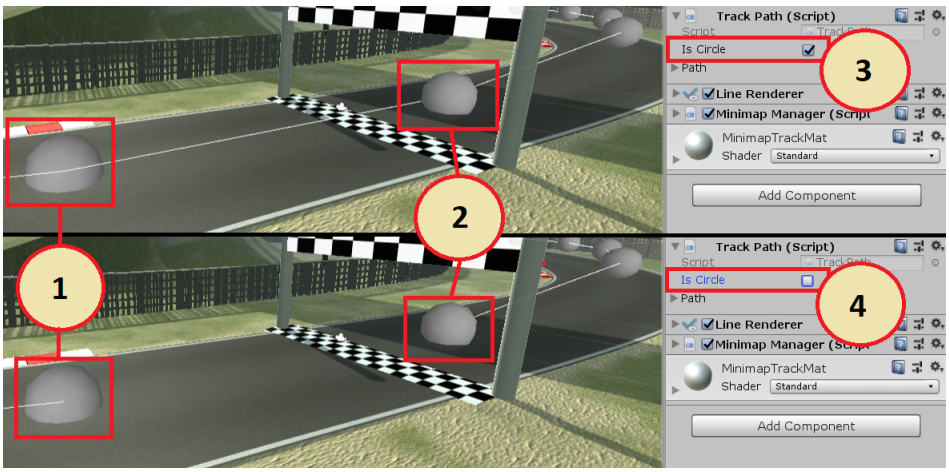


Obr. 5.69: Umiestnenie prvého waypointu.

Každý waypoint by mal mať osu **z** nastavenú v smere jazdy (kvôli respawnu). Waypointy môžeme v *Unity* duplikovať kombináciou kláves **Left Crtl + D**, vďaka čomu môžeme waypointy rýchlo duplikovať a vytvoriť cestu. Čím jemnejšie budú waypointy rozmiestnené, tým citlivejšie sa bude AI chovať počas jazdy.

Obr. 5.70: Korektné rozloženie waypointov na trati.

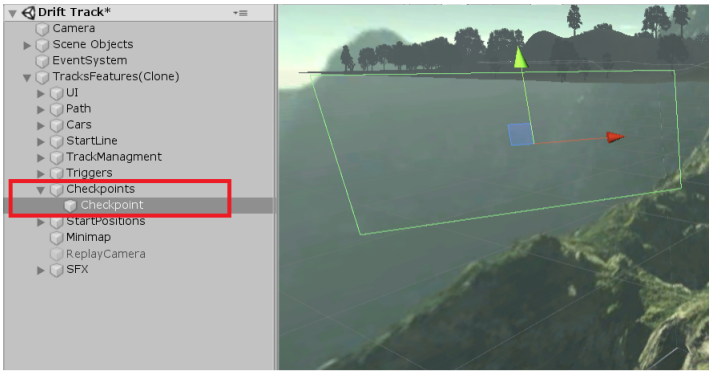
Keďže v tomto prípade sa jedná o okrúhlu dráhu, posledný a prvý waypoint sa spoja. Na obrázku 5.71 môžeme vidieť, že ak v skripte **Track Path** nastavíme trať na kruhovú (3), prvý (1) a posledný (2) waypoint sa prepojí čiarou. Ak trať nie je kruhová (4), tak sa neprepoja.



Obr. 5.71: Rozloženie waypointov na kruhovej a nekruhovej trati.

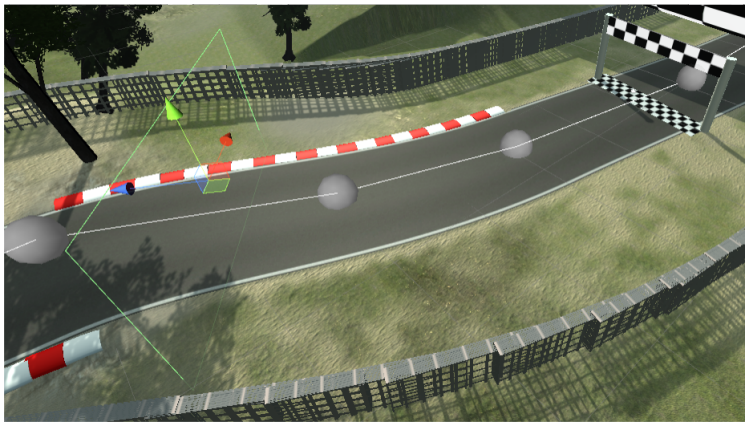
Vytvorenie Checkpointov

Rovnako ako v prípade waypointov, aj tu je nutné rozmiestniť checkpointy po trati. Obrázok 5.72 nám ilustruje, že na začiatku máme len jeden checkpoint, ktorý je umiestnený mimo trať.



Obr. 5.72: Prvý checkpoint.

Prvý checkpoint je nutné umiestniť, rovnako ako prvý waypoint, za počiatočný bod preteku ako ilustruje obrázok 5.73.



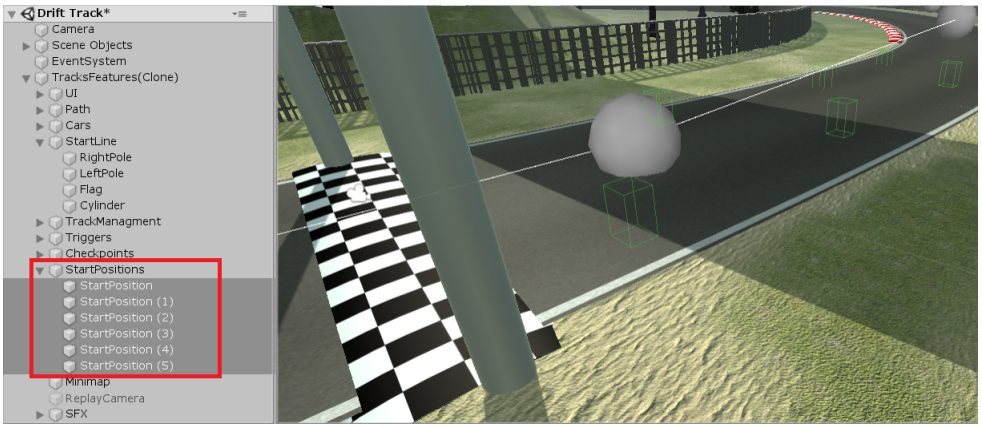
Obr. 5.73: Korektné umiestnenie prvého checkpointu.

Ostatné checkpointy odporúčame vytvárať duplikáciami prvého checkpointu pomocou kombinácie tlačidiel **Left Ctrl + D**. Hustota checkpointov na trati by nemala byť veľmi hustá ako môžeme vidieť na obrázku 5.74.

Obr. 5.74: Korektné rozloženie checkpointov na trati.

Umiestnenie štartovacích pozícií

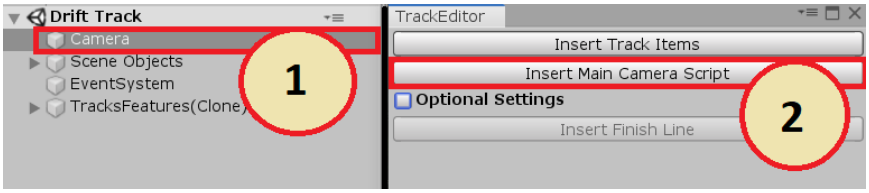
Všetky štartovacie pozície, na ktorých sa budú generovať vozidla, sa nachádzajú v objekte **StartPositions**. Ako ukazuje obrázok 5.75, tieto pozície je nutné rozmiestniť pred začiatočný bod preteku.



Obr. 5.75: Korektné rozloženie štartovacích pozícií na trati.

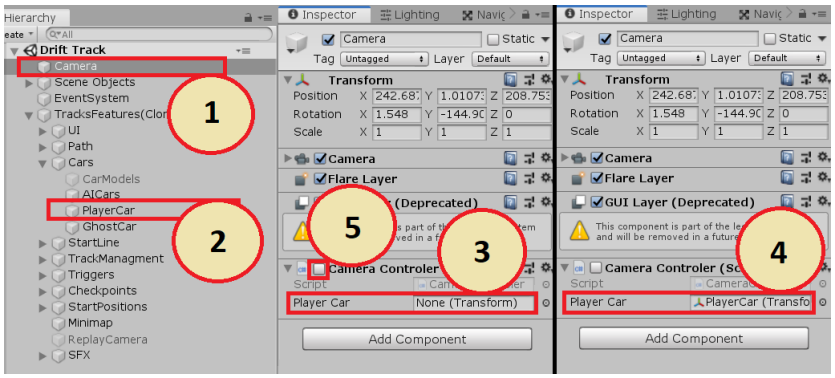
Nastavenie kamery

Do trate musíme ešte pridať skript na riadenie kamery. Obrázok 5.76 nám ukazuje, že si stačí označiť objekt kamery v danej scéne (1) a pomocou editora pridať skript do kamery (2).



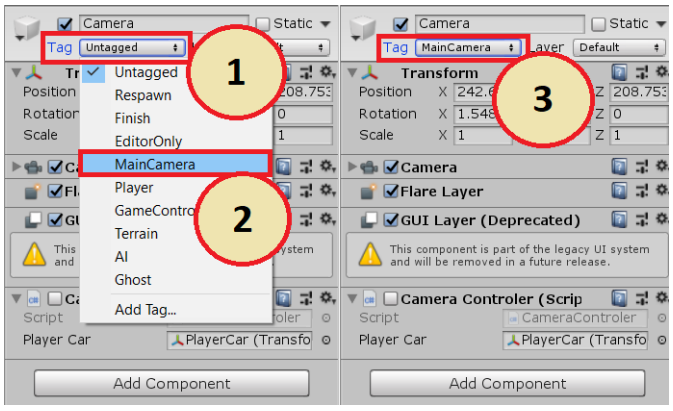
Obr. 5.76: Pridanie skriptu do kamery.

Tento skript je nutné nastaviť. Ako môžeme vidieť na obrázku 5.77, k samotnému skriptu pristupujeme kliknutím na objekt kamery (1). Skript po pridaní obsahuje prázdnu premennú **Player Car** (3). Do tejto premennej je nutné dosadiť objekt, do ktorého sa vygeneruje vozidlo hráča (2). Po priradení by sa mala premenná zobraziť ako zaplnená (4). Skript je po pridaní do kamery deaktivovaný (5) a za iných ako ladiacich okolností ho neaktivujeme.



Obr. 5.77: Nastavenie premennej skriptu kamery.

Na záver je nutné nastaviť **Tag** kamery. Na obrázku 5.78 vidíme, že **Tag** objektu kamery (1) musíme nastaviť (2) na hodnotu *MainCamera* (3).

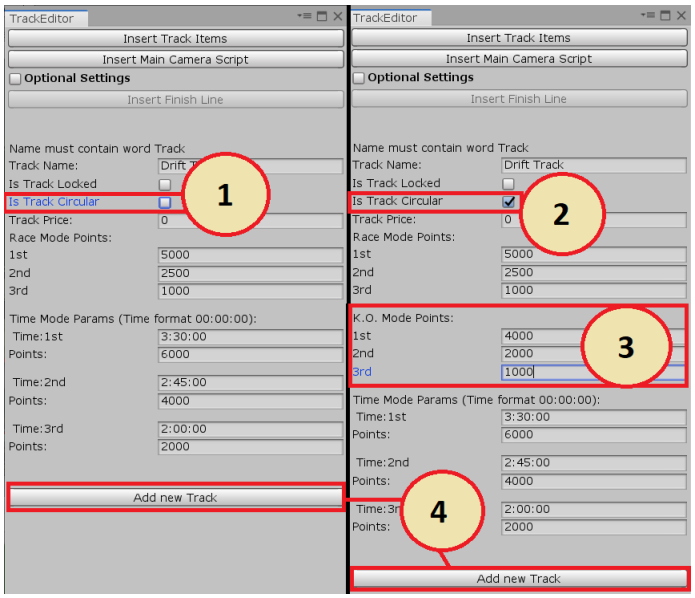


Obr. 5.78: Nastavenie tagu kamery.

Registrácia trate do XML

Po pridaní a nastavení všetkých objektov na trati musíme trať zaregistrovať do našej XML databázy. To vykonáme pomocou editora. Na obrázku 5.79 vidíme,

že ak trať nie je okrúhla (1), zmiznú nám z editora údaje o K.O móde (3). V prípade, že je trať okrúhla (2), sú tieto údaje dostupné.



Obr. 5.79: Korektné vyplnenie údajov editora pre pridanie trati do XML databázy.

Korektné vyplnenie údajov pre pridanie do XML databázy je možné vidieť na obrázku 5.79. Všetky údaje, ktoré je nutné vyplniť, sú:

- *Track Name* – meno trate. Musí byť rovnaké ako meno scény.
- *Is Track Locked* – informácia, či má byť trať zamknutá.
- *Track Price* – cena za odomknutie trate.
- *Modes Info* – informácie o herných módoch, ktoré pozostávajú z:
  - *Points* – body, ktoré hráč získa sa dané umiestnenie/čas/elimináciu.
  - *Time 1st/Time 2nd/Time 3rd* – v prípade časového módu je nutné vyplniť čas vo formáte 00:00:00.

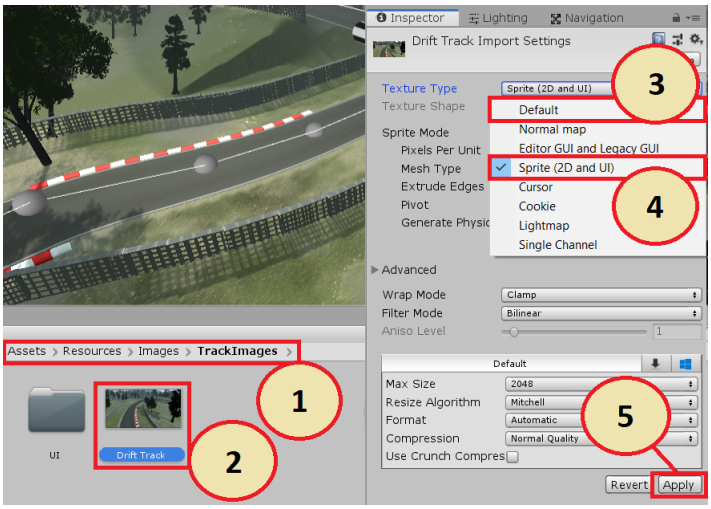
```
<tracks>
  <track Name="Drift Track" Price="0" IsLocked="False" IsCircular="True">
    <modes>
      <raceMode>
        <a Position="1st" Value="5000" />
        <b Position="2nd" Value="2500" />
        <c Position="3rd" Value="1000" />
      </raceMode>
      <timeMode>
        <a Position="3:30:00" Value="6000" />
        <b Position="2:45:00" Value="4000" />
        <c Position="2:00:00" Value="2000" />
      </timeMode>
      <koMode>
        <a Position="Wans't eliminated" Value="4000" />
        <b Position="Eliminated as last" Value="2000" />
        <c Position="Eliminated as penultimate" Value="1000" />
      </koMode>
    </modes>
  </track>
</tracks>
```

Obr. 5.80: Pridanie trati do XML databázy.

Ako ilustruje obrázok 5.80, trať sa korektne pridala do našej XML databázy.

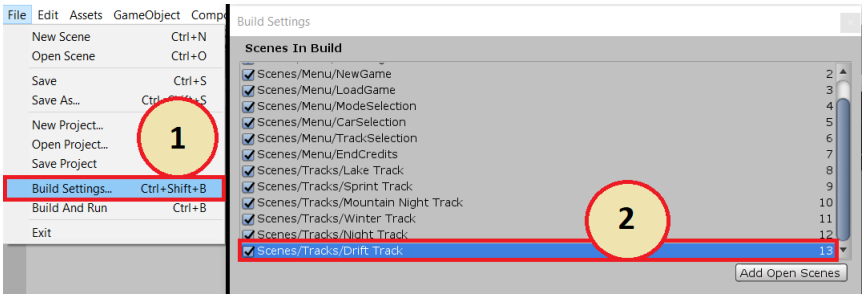
Dokončenie pridania trate

Na záver je nutné vytvoriť obrázok trate, ktorý bude trať reprezentovať v scéne na výber tratí. Na obrázku 5.81 vidíme, že obrázok trate (2) musíme umiestniť do priečinka **Images/TrackImages** v zložke **Resources** (1). Tomuto obrázku je nutné zmeniť typ textúry z počiatočného (3) na **Sprite** (4). Po potvrdení tejto zmeny (5) je možné tento obrázok používať ako textúru a teda ju môžeme využiť na reprezentovanie trate.



Obr. 5.81: Vytvorenie a umiestnenie obrázka pre reprezentáciu trate.

Aby bolo možné danú trať počas hry hrať, je nutné ju hneď po pridaní do XML pridať do *Build-u* v *Unity*. Takto *Unity* vie, že sa s touto scénou počíta pri vytvorení hry a umožní nám na ňu prechádzať aj počas ladenia. Obrázok 5.82 ilustruje, kde sa tento *Build* nachádza (1). Pridanie scény uskutočníme štýlom drag-and-drop do tabuľky scén (2), ktorá sa nám otvorí.



Obr. 5.82: Pridanie scény do buildu *Unity*.

Ako môžeme vidieť na obrázkoch 5.83 a 5.84, po tomto je v hre možné zvoliť si nami pridanú trať pri výbere tratí a po výbere je možné na tejto trati pretekať.

Obr. 5.83: Ukážka pridanej trate pri výbere tratí.

Obr. 5.84: Ukážka štartu na pridanej trati.

5.2.4 Tutoriál na rozbehnutie hry

V tomto projekte sme využívali assety z *Unity Asset Store*, ktoré licenčne neumožňujú, aby sme ich mali k dispozícií vo verejne dostupnom projekte (ako je práve tento projekt). Preto musíme pred odovzdaním projektu zmazať všetky assety, ktoré sme v projekte použili. To zahŕňa všetky vozidlá a trate, čo znamená, že projekt po zmazaní týchto assetov bude obsahovať iba scény menu a ako taký nebude funkčný.

V tomto návode ukážeme, ako projekt po stiahnutí opäť spojazdniť.

Použité assety

Na úvod si vypíšeme všetky assety, ktoré je nutné implementovať. Všetky tieto assety sa nachádzajú na *Unity Asset Store* a počas implementácie tejto práce boli ponúkané zadarmo. Ku každému assetu pripíšeme autora daného assetu.

- **Vozidla**
  - HQ Racing Car Model No.1202 – Azerilo
  - Sports Car Model and Shader – Mehdi Rabiee

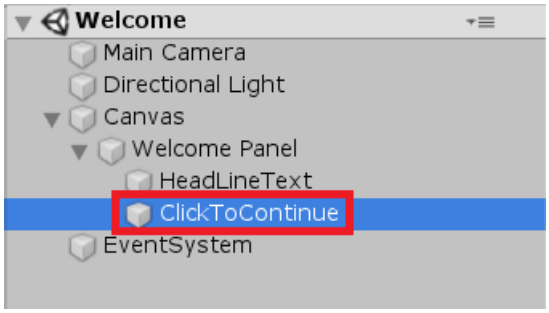
  - Sport Car - 3D model – Andrej Mikuš
  - Exotic Sportscar - Free Model – FreeFall Modelworks
  - Hot Rod – Mário Haberle
- **Trate**
  - Race Tracks – AbdulRahim
  - Lake Race Track – NIANDREI
  - Mountain race tracks – Shahbaz awan
  - Mountain Race Track - Night – NIANDREI
- **UI**
  - Clean Settings UI – Landan Lloyd
  - MK - Easy Text Light – Master Key
  - 2D Atlas Speech bubbles Alphabet Numbers – RAFMANIX
  - Slider Menu - Free – MyUnityDream
  - Simple Button Set 01 – That Witch Design
- **Zvuk**
  - ENGINES – Kristian Grundström
  - Rotary x8 - FREE Engine Sound Pack – slaczky - Skril Studio
  - Button Sound Editor – Fuzzy Games Lab
  - Free Fantasy Adventure Music Pack – Craze Music Productions
  - Take a Ride (Lounge Music) – TFSui
  - Free music and SFX collection – Alchemy Studio Music
  - FREE Casual Game SFX Pack – Dustyroom
- **Ostatné**
  - Hand Painted Stone Texture – LowlyPoly
  - Road Blocker – Pixel Games
  - Stylized Stone Tile Texture – LowlyPoly
  - Standard Assets (for Unity 2018.4) – Unity Technologies

Teraz, keď máme všetky potrebné nástroje, ukážeme si, ako pomocou nich správne nastaviť každú scénu nášho projektu. Prejdeme každú scénu, ktorú budeme mať k dispozícií. Tieto scény neobsahujú žiadnu scénu (ani obrázok) trate, keďže sme scény tratí nevytvorili. Návod na pridanie tratí sme do detailu prebrali v sekcií 5.2.3.

Terminológia

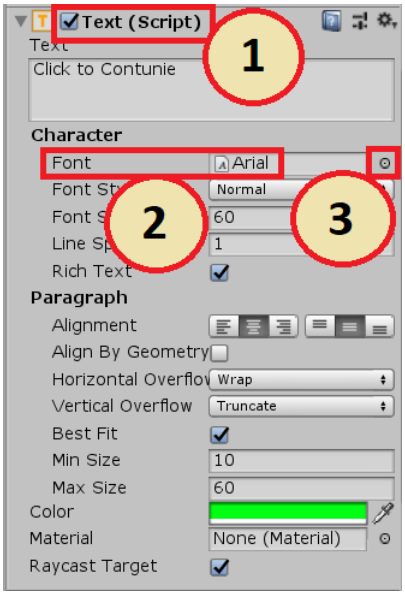
Kedže väčšina týchto úprav bude spočívať v zmene štýlov UI, nebudeme každú zmenu dokumentovať pomocou obrázkov. Namiesto toho vysvetlíme v tejto časti pojmy, ktoré budeme používať, a úpravy, ktoré budeme vykonávať. Ak nastane nejaký špecifický prípad, opíšeme ho detailne na mieste výskytu.

Počas návodu budeme používať absolútne cesty k daným objektom v danej scéne. Na obrázku 5.85 vidíme označený objekt **ClickToContinue**. Cestu k takémuto objektu by sme popísali takto: **Canvas/Welcome Panel/ClickToContinue**.



Obr. 5.85: Prístup k objektom.

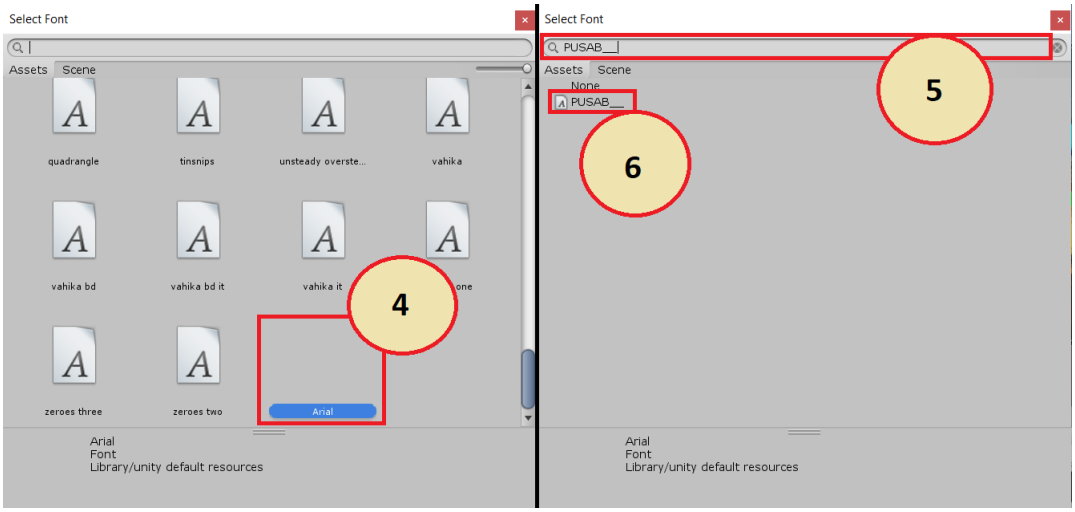
V tomto objekte chceme pristúpiť ku konkrétnemu skriptu a nastaviť mu novú premennú. Na obrázku 5.86 vidíme, že tento objekt obsahuje skript **Text** (1), ktorý obsahuje premennú **Font**. Táto premenná je nastavená na hodnotu *Arial* (2). Ak uvedieme, že chceme zmeniť hodnotu tejto premennej, myslíme tým zmenu za použitia tlačidla **Select** (3).



Obr. 5.86: Prístup k objektom.

Toto tlačidlo nám otvorí okno, ktoré je možné vidieť na obrázku 5.87. Toto okno obsahuje všetky dostupné hodnoty hľadaného typu, ktoré sa nachádzajú v našom projekte. Spočiatku toto okno označuje hodnotu (v tomto prípade font),

ktorá sa v premennej v súčasnosti vyskytuje (4). Názov fontu, ktorý chceme dosadiť napíšeme do vyhľadávača (5) a výsledok (6) priradíme do premennej kliknutím.



Obr. 5.87: Prístup k objektom.

Takže ukázaný postup bude odpovedať inštrukcií, že chceme objektu **Canvas/Welcome Panel/ClickToContinue** zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

Prefabs

V zložke **Assets/Resources/Prefabs** potrebujeme nastaviť:

1. objektu **SkidSound.prefab** v triede **Audio Source** priradiť premennej **AudioClip** hodnotu **CarSkid**.

2. objektu **Track/MinimapDot.prefab** zmeniť v komponente **Mesh Filter** premennú **Mesh** na hodnotu **Hood**.

3. objektu **UI/BGM.prefab** zmeniť v triede **Audio Source** priradiť premennej **AudioClip** hodnotu **Energetic rock loop 12**.

Welcome Scene

V tejto scnéne potrebujeme:

1. objektu **Canvas/Welcome Panel** v triede **Audio Source** priradiť premennej **AudioClip** hodnotu **Abandoned_Village___Short**.

2. objektu **Canvas/Welcome Panel/ClickToContinue** zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

3. objektu **Canvas/Welcome Panel/HeadLineText** zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep**.

Start Menu Scene

V tejto scnéne potrebujeme:

1. objektu **Canvas/Panel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **background1**.

2. objektu **Canvas/Panel/MY RACE GAME text** zmeniť v skripte **Text** premennú **Font** na hodnotu **montana bold**.

3. objektu **Canvas/Panel/ButtonPanel** a objektu **Canvas/Panel/AccountButtonPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No2-Plate**.

4. objektom **Canvas/Panel/ButtonPanel/ExitGameButton**, **../OptionsButton**, **../CreditsButton** a objektom **Canvas/Panel/AccountButtonPanel/LoadGameButton** a **../NewGameButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **round_button_up**.

    - všetky tieto objekty (tlačidlá) obsahujú objekt **Text** u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **AMERSN**.
    - všetky tieto objekty (tlačidlá) obsahujú skript **Button Sound** u ktorých je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

5. objektu **Canvas/Panel/TextPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

6. objektu **Canvas/Panel/TextPanel/InfoText** zmeniť v skripte **Text** premennú **Font** na hodnotu **Cranberry Cyr**.

7. objektu **Canvas/Panel/TextPanel/WelcomeText** zmeniť v skripte **Text** premennú **Font** na hodnotu **calibriz_se_rep**.

8. objektu **Canvas/Panel/TextPanel/SelectInfoText** zmeniť v skripte **Text** premennú **Font** na hodnotu **ARISTA2.0ALTERNATE-LIGHTTRIAL**.

9. objektu **Canvas/Panel/TextPanel/ChooseWisellyText** zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

10. objektom **Canvas/Panel/TextPanel/NewGameBulletPoint**, **../LoadGameBulletPoint**, **../OptionsBulletPoint**, **../CreditsGameBulletPoint** a **../ExitGameBulletPoint** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Hover**.

Ďalej v objekte **Assets/Resources/Prefabs/UI/CarControlInfo.prefab** (cesta v priečinkoch projektu, nie v konkrétnej scéne) je nutné:

1. objekte **CarControlInfo/CarInfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

2. objekte **CarControlInfo/CarInfoPanel/Headline** zmeniť v skripte **Text** premennú **Font** na hodnotu **montana bold**.

3. objekte **CarControlInfo/CarInfoPanel/ExitButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **x**.

    - objekt obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

4. objekte **CarControlInfo/CarInfoPanel/InfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

5. objektoch **CarControlInfo/CarInfoPanel/InfoPanel/MovementInfo/ MovementTextPanel**, **../InfoPanel/LightInfo/LightTextPanel**, **../InfoPanel/RespawnInfo/RespawnTextPanel**, **../InfoPanel/QucikCamInfo/QuickCamTextPanel**, **../InfoPanel/BrakeInfo/BrakeTextPanel** a **../InfoPanel/HandBrakeInfo/HandBrakeTextPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

    - tieto objekty v sebe obsahujú všetky textový objekt (meno poľda rodiča) u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **good times rg**.

6. objektoch **CarControlInfo/CarInfoPanel/InfoPanel/MovementInfo/ MovemenetButtonPanel**, **../InfoPanel/LightInfo/LightButtonPanel**, **../InfoPanel/RespawnInfo/RespawnButtonPanel**, **../InfoPanel/QucikCamInfo/QuickCamButtonPanel**, **../InfoPanel/BrakeInfo/BrakeButtonPanel** a **../InfoPanel/HandBrakeInfo/HandBrakeButtonPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No2-Plate**.

    - tieto objekty v sebe obsahujú všetky objekty obrázka u ktorých je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.
    - objekty obrázkov v sebe obsahujú textový objekt u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **vahika bd**.

7. objekte **CarControlInfo/CarInfoPanel/ExplanationPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

8. objekte **CarControlInfo/CarInfoPanel/ExplanationPanel/InfoText** zmeniť v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

Na záver je nutné objektu **Assets/Resources/Prefabs/UI/BGM.prefab** (cesta v priečinkoch projektu, nie v konkretnéj scéne) v triede **Audio Source** priradiť premennej **AudioClip** hodnotu **Energetic rock loop 12**.

New Game Scene a Load Game Scene

Obe scény sú štrukturované úplne rovnako. Zmeny, ktoré popíšeme, budú zamerané na scénu New Game. Scénu Load Game zmeníme úplne rovnako (mená objektov sa budú trocha líšiť).

1. objekte **Canvas/Panel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **background**.

2. objekte **Canvas/Panel/NamePanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

3. objektom **Canvas/Panel/NamePanel/NameField** a **../NamePanel/PasswordField** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

4. objektom **Canvas/Panel/NamePanel/EnterName** a **../NamePanel/EnterPass** zmeniť v skripte **Text** premennú **Font** na hodnotu **calibriz_se_rep**.

5. objektom **Canvas/Panel/NamePanel/WrongNameBuble**, **../NamePanel/WrongPassBuble** a **../NamePanel/ShortNameBuble** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **+Speechbubble_1**.

    - tieto objekty v sebe obsahujú textový objekt u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

6. objektom **Canvas/Panel/NamePanel/WrongNameImage** a **../NamePanel/WrongPassImage** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **button_round_400**.

    - tieto objekty v sebe obsahujú obrázkový objekt, u ktorého je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **info**.

7. objektom **Canvas/Panel/CreateButton** a **Canvas/Panel/BackButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

    - tieto objekty v sebe obsahujú textový objekt u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
    - objekty naviac obsahujú skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

8. objeku **Canvas/Panel/InfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

9. objektom **Canvas/Panel/InfoPanel/Top** a **../InfoPanel/Middle** zmeniť v skripte **Text** premennú **Font** na hodnotu **calibriz**.

10. objektom **Canvas/Panel/InfoPanel/End1** a **../InfoPanel/End2** zmeniť v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

11. objeku **Canvas/Panel/NewAccountName** zmeniť v skripte **Text** premennú **Font** na hodnotu **AMERSN__se__rep**.

Mode Selection Scene

V tejto scéne potrebujeme:

1. objektu **Canvas/Panel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **background3**.

2. objektu **Canvas/Panel/BackButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

    - objekt v sebe obsahuje textový objekt u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
    - objekt naviac obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

3. objektu **Canvas/Panel/ModesPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

4. objektom **Canvas/Panel/ModesPanel/RaceModeButton**, **../ModesPanel/TimeModeButton** a **../ModesPanel/KOModeButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **TextText-Effect-No2-Button-Normal**.

    - objekty v sebe obsahujú textový objekt u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **montana bold**.
    - objekt naviac obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

5. objektu **Canvas/Panel/ModeInfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

6. objektu **Canvas/Panel/ModeInfoPanel/HeadLineText** zmeniť v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

7. objekty **Canvas/Panel/ModeInfoPanel/RaceModeText**, **../ModeInfoPanel/TimeModeText** a **../ModeInfoPanel/KOModeText** obsahujú rovnaké objekty, pričom.

    - objektom **ModeInfo** a **ModeName** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Accuratist**.

- objektom **Quote** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **AMERSN__se__rep**.

8. objektu **Canvas/Panel/PlayerPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

9. objektom **Canvas/Panel/PlayerPanel/NameText** a **../PlayerPanel/PointsText** zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

10. objektu **Canvas/Panel/HeadLine** zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista__se__rep**.

Car Selection Scene

V tejto scéne potrebujeme:

1. objektu **Canvas/Panel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **background2**.

2. objektu **Canvas/Panel/PlayerPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

3. objektom **Canvas/Panel/PlayerPanel/NameText** a **../PlayerPanel/PointsText** zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

4. objektu **Canvas/Panel/CarSelectionPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No2-Plate**.

5. objektom **Canvas/Panel/CarSelectionPanel/RightButton** a **../CarSelectionPanel/ColorRightButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **arrow__R**.

   - objekty naviac obsahujú skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **DM-CGS-47**.

6. objektom **Canvas/Panel/CarSelectionPanel/LeftButton** a **../CarSelectionPanel/ColorLeftButton** zmeniť v skripte **Image** premennú
   Source Image na hodnotu **arrow__L**.

   - objekty naviac obsahujú skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **DM-CGS-47**.

7. objektom **Canvas/Panel/CarSelectionPanel/CarName** a **../CarSelectionPanel/ColorName** zmeniť v skripte **Text** premennú **Font** na hodnotu **ARISTA2.0-LIGHTTRIAL**.

8. objektu **Canvas/Panel/InfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

9. objektom **Canvas/Panel/InfoPanel/Unlocked/TopSpeedText**, **../Unlocked/AccelerationText**, **../Unlocked/BrakeText**, **../Unlocked/SteerAngleText**, **../Unlocked/DriveTypeText** a **../Unlocked/DriveTypeInfo** zmeniť v skripte **Text** premennú **Font** na hodnotu
Arista_se_rep.

10. objektom
Canvas/Panel/InfoPanel/Unlocked/TopSpeedSlider/Background, **../Unlocked/AccSlider/Background**, **../Unlocked/BrakeSlider/Background** a **../Unlocked/SteerAngleSlider/Background**
zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

11. objektu **Canvas/Panel/InfoPanel/Unlocked/UpgradeButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Button-Normal**.

    - objekt v sebe obsahuje textový objekt, u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista2.0-RegularTrial**.
    - objekt naviac obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

12. objektu **Canvas/Panel/InfoPanel/Lock/LockInfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

13. objektom **Canvas/Panel/InfoPanel/Lock/LockInfoPanel/Top** a **../LockInfoPanel/Bottom** zmeniť v skripte **Text** premennú **Font** na hodnotu **Accuratist**.

14. objektu **Canvas/Panel/InfoPanel/Lock/LockInfoPanel/PriceText** zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep**.

15. objektu **Canvas/Panel/InfoPanel/Lock/UnlockeButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Button-Normal**.

    - objekt v sebe obsahuje textový objekt, u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista2.0-RegularTrial**.
    - objekt naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **money** a premennú **Hower** na premennú **click1**.

16. objektom **Canvas/Panel/InfoPanel/Lock/FailPanel** a **../Lock/SuccessPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No2-Plate**.

17. objektom **Canvas/Panel/InfoPanel/Lock/FailPanel/FailImage** a **../Lock/SuccessPanel/SuccImage** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **button_round_400**.

    - objekty v sebe obsahujú obrázkový objekt u ktorého je nutné zmeniť v skripte **Raw Image** premennú **Texture** na hodnotu **info**.

18. objektom **Canvas/Panel/InfoPanel/Lock/FailPanel/FailSpeechBuble** a **../Lock/SuccessPanel/SuccesBuble** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **+Speechbuble_1**.

    - objekty v sebe obsahujú textové objekty, u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

19. objektom **Canvas/Panel/InfoPanel/Lock/FailPanel/ContinueButton** a **../Lock/SuccessPanel/ContinueButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

    - objekty v sebe obsahuje textový objekt, u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
    - objekty naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

20. objektom **Canvas/Panel/NextButton** a **Canvas/Panel/BackButton** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

    - objekty v sebe obsahuje textový objekt, u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
    - objekty naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

21. objeku **Canvas/Panel/Headline** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

22. objeku **RotationPlatform** je nutné zmeniť v triede **Mesh Renderer** prvú premennú poľa **Materials** na hodnotu **Stylize_Stone**.

23. objeku **Ground** je nutné zmeniť v triede **Mesh Renderer** prvú premennú poľa **Materials** na hodnotu **dirt**.

Upgrade

Tento panel je obsiahnutý v scéne Car Selection a jedná sa o obejkt **Canvas/Panel/UpgradePanel**. V tomto paneli musíme:

1. objektu **UpgradePanel/Upgrade** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

2. objektom **UpgradePanel/Upgrade/TopSpeedUpButton**, **../Upgrade/AccUpButton** a **../Upgrade/BrakeUpButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **arrow_R**.

    - objekty naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **DM-CGS-47** a premennú **Hower** na premennú **click1**.

3. objektom **UpgradePanel/Upgrade/TopSpeedDownButton**, **../Upgrade/AccDownButton** a **../Upgrade/BrakeDownButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **arrow_R**.

    - objekty naviac obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **DM-CGS-47** a premennú **Hower** na premennú **click1**.

4. objektom **UpgradePanel/Upgrade/TopSpeedSlidder/Background**, **../Upgrade/AccSlidder/Background** a **../Upgrade/BrakeSlidder/Background** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

5. objektom **UpgradePanel/Upgrade/TopSpeedText**, **../Upgrade/AccText** a **../Upgrade/BrakeText** zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep**.

6. objektu **UpgradePanel/ExitButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **X**.

    - objekty naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

7. objektu **UpgradePanel/ApllyButton** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

    - objekt v sebe obsahuje textový objekt, u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
    - objekt naviac obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **money** a premennú **Hower** na premennú **click1**.

8. objektu **UpgradePanel/CarShowPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

9. objektu **UpgradePanel/PointsPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

    - objekt obsahuje textové objekty, u kotrých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **demonius line**.

Track Selection Scene

V tejto scéne musíme:

1. objektu **Canvas/Panel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **background3**.

2. objektom **Canvas/Panel/BackButton** a **Canvas/Panel/RaceButton** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

   - objekty v sebe obsahuje textový objekt u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
   - objekty naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** pre **BackButton** a **startup** pre **RaceButton** a premennú **Hower** na premennú **click1**.

3. objektu **Canvas/Panel/PlayerPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

   - objekt v sebe obsahuje textové objekty, u ktorých je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

4. objektu **Canvas/Panel/TrackSelectionPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

5. objektu **Canvas/Panel/TrackSelectionPanel/TrackName** zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista2.0-RegularTrial**.

6. objektom **Canvas/Panel/TrackSelectionPanel/LeftButton** a **../TrackSelectionPanel/RightButton** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **arrow_L** a textbfarrow_R.

   - objekty naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **DM-CGS-47** a premennú **Hower** na premennú **click1**.

7. objektu **Canvas/Panel/TrackSelectionPanel/InfoPanel** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

8. objektom **Canvas/Panel/TrackSelectionPanel/InfoPanel/ModePanel**, **../InfoPanel/LapPanel**, **C../InfoPanel/RecordPanel** a **../InfoPanel/TrackPointInfoPanel** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

9. objektu **Canvas/Panel/TrackSelectionPanel/InfoPanel/ ModePanel/SelectedMode** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **braeside lumberboy**.

10. objektom **Canvas/Panel/TrackSelectionPanel/InfoPanel/LapPanel/LapsText** a **../LapPanel/LapsInfo**je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **zeroes two** a hodnotu **vahika it**.

11. všetkým objektom v **Canvas/Panel/TrackSelectionPanel/InfoPanel/RecordPanel** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **vahika it**.

12. všetkým objektom v **Canvas/Panel/TrackSelectionPanel/InfoPanel/TrackPointInfoPanel** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep** .

13. objektu **Canvas/Panel/Headline** zmeniť v skripte **Text** premennú **Font** na hodnotu **montana bold**.

14. objekt **Canvas/Panel/SuccessPanel** podlieha rovnakým zmenám ako **SuccessPanel** v sekcií 5.2.4.

15. objektu **Canvas/Panel/LockPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No6-Plate**.

16. objektom **Canvas/Panel/LockPanel/LockInfoPanel** a **../LockPanel/UnlockInfo** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

17. objektom **Canvas/Panel/LockPanel/LockInfoPanel/PriceText** a **../LockInfoPanel/Price** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **demonius line** a hondotu **Arista_se_rep**.

18. objektu **Canvas/Panel/LockPanel/UnlockInfo/UnlockInfoText** je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **demonius line** a hondotu **good times rg**.

19. objektu **Canvas/Panel/LockPanel/UnlockInfo/UnlockButton** je nutné zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Button-Normal**.

    - objekt obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **money** a premennú **Hower** na premennú **click1**.
    - objekt v sebe obsahuje textový objekt u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Arista2.0-RegularTrial**.

20. objekt **Canvas/Panel/LockPanel/FailPanel** podlieha rovnakým zmenám ako **FailPanel** v sekcií 5.2.4.

21. objektu **Canvas/Panel/KOModeWarningPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No6-Plate**.

22. objektu **Canvas/Panel/KOModeWarningPanel/LockInfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

23. objektu **Canvas/Panel/KOModeWarningPanel/LockInfoPanel** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

24. objektu **Canvas/Panel/KOModeWarningPanel/LockInfoPanel/Image** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **button_round_400**.

25. objektu **Canvas/Panel/KOModeWarningPanel/LockInfoPanel/Image/RawImage** zmeniť v skripte **Raw Image** premennú **Texture** na hodnotu **info**.

26. objektu **Canvas/Panel/KOModeWarningPanel/LockInfoPanel/SpeachBuble** zmeniť v skripte **Image** premennú **Source Image** na hodnotu **+Speechbubble_4**.

27. objektu **Canvas/Panel/KOModeWarningPanel/LockInfoPanel/SpeachBuble/InfoText** zmeniť v skripte **Text** premennú **Font** na hodnotu **good times rg**.

Track UI

UI trate sa nachádza v objekte **Assets/Resources/Prefabs/Track/TracksFeatures.prefab**. Je nutné aby sme:

1. v objektoch **UI/GameUIPanel/TimePanel**, **UI/GameUIPanel/LapTimes**, **UI/GameUIPanel/Lap** a **UI/GameUIPanel/Position** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

2. v objektoch **UI/GameUIPanel/Lap/Text** a **UI/GameUIPanel/Position/Text** nastavili v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep**.

3. v objektoch **UI/GameUIPanel/TimePanel/CurrentTime**, **../TimePanel/BestTime** a **../TimePanel/PersonalBest** nastavili v skripte **Text** premennú **Font** na hodnotu **AMERSN**.

4. v objektoch **UI/GameUIPanel/TimePanel/CurrentTimeText**, **../TimePanel/BestTimeText** a **../TimePanel/PersonalBestText** nastavili v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

5. v objekte **UI/GameUIPanel/TimePanel/LapTimes/Text** nastavili v skripte **Text** premennú **Font** na hodnotu **braeside lumberboy**.

6. v objekte **UI/CountDown** nastavili v skripte **Text** premennú **Font** na hodnotu **montana bold**.

End Game Panel

Tento panel sa nachádza v UI trate. Konkrétne v objekte **UI/EndGameImage/EndGamePanel**. Pre tento panel je nutné aby sme:

1. v objekte **EndGamePanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

2. v objekte **EndGamePanel/ButtonPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

3. v objektoch **EndGamePanel/ButtonPanel/RestartButton**, **../ButtonPanel/ReplayButton** a **../ButtonPanel/Back To Menu** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No3-Button-Normal**.

    - objekty v sebe obsahuje textový objekt, u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **zeroes three**.
    - objekty naviac obsahuje skript **Button Sound** u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

4. v objekte **EndGamePanel/InfoPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Plate**.

5. v objektoch **EndGamePanel/InfoPanel/RestartInfoText**, **../InfoPanel/ReplayInfoText** a **../InfoPanel/BackToMenuInfoText** nastavili v skripte **Text** premennú **Font** na hodnotu **zeroes one**.

6. vo všetkých ostatných objektoch v **EndGamePanel/InfoPanel** nastavili v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep**.

Replay Panel

Tento panel sa nachádza v UI trate. Konkrétne v objekte **UI/ReplayPanel**. Pre tento panel je nutné aby sme:

1. v objekte **ReplayPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No4-Plate**.

2. v objekte **ReplayPanel/ReplayText** nastavili v skripte **Text** premennú **Font** na hodnotu **zeroes one**.

3. v objekte **ReplayPanel/ButtonPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

4. v objekte **ReplayPanel/ButtonPanel/VideoText** nastavili v skripte **Text** premennú **Font** na hodnotu **Poppins-Bold**.

5. v objektoch **ReplayPanel/ButtonPanel/PlayStopButton**, **../ButtonPanel/BackButton**, **../ButtonPanel/ForwardButton** a **../ButtonPanel/RepeatButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **button_200**.

    - objekty naviac obsahujú skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

6. v objekte **ReplayPanel/ButtonPanel/PlayStopButton/PlayImage** nastavili v skripte **Image** premennú **Source Image** na hodnotu **play**.

7. v objekte **ReplayPanel/ButtonPanel/PlayStopButton/PauseImage** nastavili v skripte **Image** premennú **Source Image** na hodnotu **pause**.

8. v objekte **ReplayPanel/ButtonPanel/BackButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **arrow_left**.

9. v objekte **ReplayPanel/ButtonPanel/ForwarButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **arrow_right**.

10. v objekte **ReplayPanel/ButtonPanel/RepeatButton/Image** nastavili v skripte **Image** premennú **Source Image** na hodnotu **repeat**.

11. v objekte **ReplayPanel/ButtonPanel/GoBackButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

    - objekt v sebe obsahuje textový objekt u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.
    - objekt naviac obsahuje skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.

12. v objekte **ReplayPanel/ShowPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

Pause Panel

Tento panel sa nachádza v UI trate. Konkrétne v objekte **UI/PausePanel**. Pre tento panel je nutné aby sme:

1. v objekte **PausePanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

2. v objekte **PausePanel/Headline** nastavili v skripte **Text** premennú **Font** na hodnotu **Arista_se_rep**.

3. v objekte **PausePanel/InfoPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No5-Plate**.

4. v objektoch **PausePanel/InfoPanel/ResumeInfo/HeadText**, **../InfoPanel/RestartInfo/HeadText** a **../InfoPanel/BackToMenuInfo/HeadText** nastavili v skripte **Text** premennú **Font** na hodnotu **Cartwheel**.

5. v objektoch **PausePanel/InfoPanel/ResumeInfo/Panel**, **../InfoPanel/RestartInfo/Panel** a **../InfoPanel/BackToMenuInfo/Panel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No6-Plate**.

    - v objekte **InfoText** nastavili v skripte **Text** premennú **Font** na hodnotu **Arista2.0-RegularTrial**.

- v objekte **Laugh** nastavili v skripte **Text** premennú **Font** na hodnotu **zeroes three**.

6. v objekte **PausePanel/ButtonPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Plate**.

7. v objekte **PausePanel/ButtonPanel/ResumeButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No8-Button-Normal**.

8. v objekte **PausePanel/ButtonPanel/RestartButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No2-Button-Normal**.

9. v objekte **PausePanel/ButtonPanel/BackToMenuButton** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No7-Button-Normal**.

    - tieto tri tlačítka obsahujú skript **Button Sound**, u ktorého je nutné zmeniť premennú **Click** na hodnotu **click2** a premennú **Hower** na premennú **click1**.
    - tlačítka v sebe obsahujú textový objekt u ktorého je nutné zmeniť v skripte **Text** premennú **Font** na hodnotu **Abraeside lumberboy**.

10. v objekte **PausePanel/PlayerInfoPanel** nastavili v skripte **Image** premennú **Source Image** na hodnotu **Text-Effect-No6-Plate**.

11. v objektoch **PausePanel/PlayerInfoPanel/NameText** a **../PlayerInfoPanel/PointsText** nastavili v skripte **Text** premennú **Font** na hodnotu **PUSAB___**.

12. v objektoch **PausePanel/PlayerInfoPanel/PlayerName** a **../PlayerInfoPanel/PlayerPoints** nastavili v skripte **Text** premennú **Font** na hodnotu **AMERSN**.

SFX

Tento objekt sa nachádza v **TracksFeatures** trate. Konkrétne v objekte **SFX**. Pre tento panel je nutné aby sme:

1. v objekte **CountSFX** nastavili premennú **AudioClip** komponente **AudioSource** na hodnotu **DM-CGS-03**.

2. v objekte **GoSFX** nastavili premennú **AudioClip** komponente **AudioSource** na hodnotu **DM-CGS-10**.

3. v objekte **LevelMusic** nastavili premennú **AudioClip** komponente **AudioSource** na hodnotu **Loop 16 - 145 bpm - loop**.

4. v objekte **ReplayMusic** nastavili premennú **AudioClip** komponente **AudioSource** na hodnotu **Hot pepper - short loop 1**.

Závěr

V tejto práci sme sa venovali návrhu a implementácií 3D závodnej hry v *Unity*. V závere cheme zhodotiť, či sme úspešne splnili cieľe, ktoré sme si určili v sekcií 2.11, a rovnako tak spomenúť možné rozšírenia tohoto projektu do budúcna.

1. Hra obsahuje realistické vozidlá, ktoré sú schopné pohybovať sa a zatáčať po tratiach, brzdiť a prechádzať plynule do šmyku. Každé vozidlo obsahuje zvukové a svetelné efekty v kombinácií s animáciou kolies. Vozidlá reagujú na vstup hráča, sú schopné respawnu a obsahujú dynamickú kameru s možnosťou spustenia quick kamery.

2. Hra obsahuje AI, ktorá je schopná ovládať vozidlá a dokáže korektne prekonať trať. Okrem ovládania je AI schopná vyhýbať sa statickým a dynamickým prekážkam na trati a dynamicky sa rýchlosťou prispôsobuje rýchlosť vozidlu hráča.

3. Hráč v hre získava body za úspešné zvládnutie podmienok pretekov, ktoré sa líšia v závislosti na hernom móde. Tieto body môže hráč využiť na odomykanie vozidiel a tratí alebo vylepšovanie vozidla.

4. Hráč si v hre môže zvoliť jeden z troch herných módov, ktoré sú v hre implementované: *Race Mode*, *Time Mode* a *K.O. Mode*.

5. Hra obsahuje hráčske účty, ktoré hráčovi ukladajú jeho progres. Tieto účty je hráč schopný vytvárať, mazať a prihlasovať sa do nich. Každý účet obsahuje informácie o hráčovom mene, hesle, počte bodov a zoznamy odomknutých tratí a vozidiel spolu s mierov vylepšenia každého vozidla.

6. Hra poskytuje možnosť replay, vďaka ktorému si hráč môže prehrať záznam preteku.

7. Celá hra obsahuje intuitívne rozhranie, v ktorom sa hráč jednoducho orientuje.

8. Do hry sme vytvorili editor na jednoduché pridávanie a nových vozidiel a tratí.

Počas návrhu a implementácie tejto práce sa ukázalo, že sa bude jednať o náročnejší a väčší projekt než sme spočiatku čakali. Celkovo sme to zvládli najlepšie ako sme vedeli. Splnili sme všetky ciele, ktoré sme si v práci stanovili a preto prácu hodnotíme ako úspešnú.

Počas implementácie sme narážali na nové aspekty, nástroje a mechaniky, ktoré by bolo skvelé implementovať, ak by sme sa hru snažili v budúcnosti rozšíriť. Nasledujúci zoznam obsahuje prvky, ktoré vy zvýšili úroveň našej práce:

- Poskytnutie možnosti hry pre viac hráčov. Hráči by tak mohli pretekať proti sebe vo forme multiplayeru.

- Pridať do tratí turbo, ktoré by hráčov "vystrelilo"dopredu neskutočnou rýchlosťou. Tento prvok by hru urobil dynamickejšou.

- Vytvorenie mechaniky pre dynamické prekážky, ktoré by sa počas hry objavovali na trati. Hráči by tak museli byť v neustálom strehu.
- Vytvorenie mechaniky na poškodzovanie vozidiel v prípade kolízií. Každé poškodenie by ovplyvnilo výkon vozidla.

Seznam použité literatury

[1] StraitsResearch. Top 10 Most Popular Gaming Genres in 2020. https://straitsresearch.com/blog/top-10-most-popular-gaming-genres-in-2020/. Platnosť odkazu: 19.05.2021.

[2] Bob Bates. *Game Design*. Course Technology PTR, 2004. p. 59, ISBN 978-1-59200-493-5.

[3] porscheclub.com. What is Sim Racing? https://www.porscheclub.com/simracing/. Platnosť odkazu: 19.05.2021.

[4] SimTorquer. Arcade vs Simulation – What does it all really mean? https://simtorque.wordpress.com/2013/01/22/arcade-vs-simulation-what-does-it-all-really-mean/. Platnosť odkazu: 20.05.2021.

[5] Polyphony Digital & Sony Computer Entertainment. Products. https://www.polyphony.co.jp/products/. Platnosť odkazu: 20.05.2021.

[6] Wesley Yin-Poole. Criterion takes full control of Need for Speed and Burnout franchises. https://www.eurogamer.net/articles/2012-06-25-criterion-takes-full-control-of-need-for-speed-and-burnout-franchises. Platnosť odkazu: 20.05.2021.

[7] Electronic Arts. EA Signs Multi-Year Agreement With X-Games Medalist and Internet Phenom Ken Block for Need for Speed. https://www.businesswire.com/news/home/20131024005495/en/EA-Signs-Multi-Year-Agreement-With-X-Games-Medalist-and-Internet-Phenom-Ken-Block-for-Need-for-Speed. Platnosť odkazu: 22.05.2021.

[8] Alan. Hot Wheels are bringing a selection of Need for Speed cars to stores worldwide. https://ar12gaming.com/articles/hot-wheels-are-bringing-a-selection-of-need-for-speed-cars-to-stores-worldwide. Platnosť odkazu: 23.05.2021.

[9] Unity Technologies. Scenes. https://docs.unity3d.com/Manual/CreatingScenes.html. Platnosť odkazu: 15.05.2021.

[10] Unity Technologies. Transform. https://docs.unity3d.com/ScriptReference/Transform.html. Platnosť odkazu: 15.05.2021.

[11] Unity Technologies. Scripting. https://docs.unity3d.com/Manual/ScriptingSection.html. Platnosť odkazu: 15.05.2021.

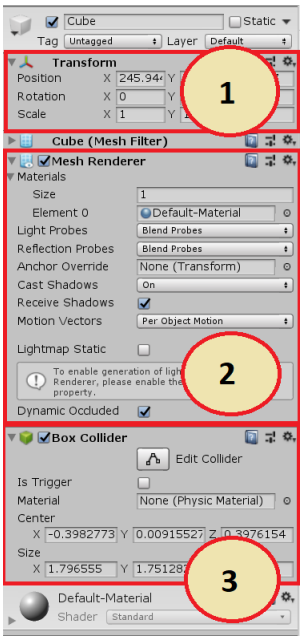
A. Přílohy

A.1 Herný engine Unity

V tejto časti sa pokúsime priblížiť základné pojmy a nástroje *Unity*, ktoré v tejto práci používame. *Unity* je multiplatformový herný engine, ktorý sa používa na vytváranie 2D a 3D hier. S týmto enginom je možné vytvárať hry na počítače s operačným systémom *Windown*, *Linux* či *MacOS*. Podporuje vytváranie hier pre mobilné zariadenie a herné konzoly. V prípade záujmu o podrobnejšie popisy doporučujeme navštíviť **Unity User Manual**, ktorý obsahuje celú dokumentáciu a veľa návodov k všetkým komponentom a triedam *Unity*. Tento manuál je možné nájsť na stránke *docs.unity3d.com*.

A.1.1 Scéna

Scény sú miesto, kde pracujeme s objektami *Unity* [9]. Obsahujú všetky objekty, ktoré sa v danej scéne používajú. Objekty v scéne môžeme presúvať, rotovať alebo meniť ich veľkosť. Vlastnosti každého objektu môžeme pozorovať, upravovať a pridávať pomocou *Inšpektora*. Na obrázku A.1 môžeme vidieť ukážku základného objektu **cube** (kocky) v *Unity*. Tento objekt obsahuje **Transform** (1), vďaka ktorému je možné pohybovať, rotovať a meniť veľkosť objektu. **Transform** je obsiahnutý v každom objekte scény [10]. Objekt ďalej obsahuje **Mesh Renderer** (2), ktorý umožňuje renderovanie objektu na scéne.



Obr. A.1: Prehľad objektu v *Inšpektore*.

Záverom obsahuje objekt **Box Collider** (3), ktorý mu umožňuje interakciu a kolíziu s objektmi, ktoré obsahujú nejaký typ **Collidera**.

A.1.2 Skripty

Tvorba a používanie skriptov je základným kameňom tvorby hier v *Unty*. Väčšina aplikácií potrebuje skripty na reagovanie na vstup užívateľa a zariadiť, aby všetky sa korektne vykonali všetky udalosti, ktoré sa počas hry vykonať majú [11]. Každý skript po vytvorení obsahuje triedu, ktorá je odvodená od triedy **MonoBehavior**. Vďaka tomuto dedeniu môžeme v každom skripte využiť sadu metód, ktoré potom scéna volá. Jedná sa o metódu *Awake*, ktorá sa volá pri vytvorení inštancie skriptu *Start*, ktorý sa volá v prvom frame, keď je daný skript aktivovaný. Medzi ďalšie dôležité metódy patria *Update*, ktorý sa volá každý frame a metóda *FixedUpdate*, ktorý sa volá niekoľko krát za frame a odporúča sa používať pri fyzikálnych výpočtoch.

Skripty taktiež umožňujú vytvárať objekty pomocou metódy *Instatiate*, pridávať komponenty do objektov vďaka metóde *AddComponent* a pristupovať ku komponentom objektov cez metódu *GetComponent*. V prípade potreby môžeme vďaka skriptom vyhľadávať objekty v scéne pomocou metódy *Find*.

A.1.3 Užívateľské rozhranie

Unity poskytuje užívateľom možnosť vytvárať a upravovať užívateľské rozhrania. Základným objektom pri tvorbe rozhraní je objekt **Canvas**, kde sa nachádzajú všetky prvky UI v danej scéne. Všetky tieto prvky musia byť herné objekty.

Jednotlivé UI objekty v scéne sa dajú rozdeľovať na sekcie pomocou objektu **Panel**. Neinteraktívne komponenty, ktoré tvoria základ UI, sú **Text**, *Raw Image* a **Image**, ktoré sa používajú na uchovávanie textu, obrázkov a textúr.

Unity ponúka aj interaktívne prvky ako komponentu **Button**, ktorá reprezentuje tlačidlo **Slider**, ktorý reprezentuje posuvník, alebo **Input Field**, ktorý reprezentuje vstupné pole. Všetky tieto interaktívne prvky v sebe obsahujú reakcie na udalosti ako kliknutie či posun myši. Všetky tieto udalosti môžeme upravovať pomocou skriptov.