Izrada računalne igre utrkivanja s pogledom odozgo

Radotović, Emanuel

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Emanuel Radotović

Izrada računalne igre utrkivanja s pogledom odozgo

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Emanuel Radotović

Matični broj: 0069078505

Studij: Informacijski sustavi

Izrada računalne igre utrkivanja s pogledom odozgo

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Radošević Danijel

Varaždin, rujan 2021.

Emanuel Radotović

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada je izrada dvodimenzionalne računalne igre na Platformi *Unity*. Rad se bavi teorijskom stranom izrade računalnih igara, kao i praktičnim dijelom same izrade grafike, animacija, zvukova, mehanika i svega ostalog što je potrebno da bi igra bila funkcionalna i zanimljiva. Razvijene su dvije glavne mogućnosti igranja, a to su *Singleplayer*, odnosno mogućnost igranja protiv *AI (Artificial intelligence)* programski napravljenih suigrača, te *Multiplayer*, odnosno mogućnost lokalnog igranja sa još jednom osobom. Uz platformu *Unity* koristi se programski jezik *C#* za izradu skripti koje sadrže programsku potporu, te razni alati i izvori sa interneta za izradu grafike i muzike. Objašnjen je način stvaranja objekata, mogućnosti rada nad i sa njima, te način na koji međusobno komuniciraju u programskom smislu. Prikazan je način njihovog uređivanja u smislu grafike i zvukova.

Ključne riječi: Dizajn, Mehanika, Artifical intelligence, Programiranje, Računalna igra, Unity, C#

Sadržaj

1. Uvod

Ovaj rad bazira se na praktičnoj izradi računalne igre, te teorijskoj pratnji izrađenog. Radi se o računalnoj igri žanra *utrke* koja ima dvije mogućnosti igranja. Prva mogućnost je da osoba igra sama, odnosno da se utrkuje protiv programski izrađenih protivnika, odnosno *AI*. Druga mogućnost je utrka dvije osobe međusobno, gdje AI nije uključen. Detaljno su objašnjeni i prikazani načini izrade obje mogućnosti koje u suštini imaju istu glavnu mehaniku vožnje no u nekim detaljima se razlikuju, što u dizajnu, a što u programskom dijelu. Igra je dvodimenzionalna te su u skladu s time prikazane mogućnosti platforme *Unity* i njezine funkcionalnosti koje pruža vezano uz to. Objašnjena je izrada skripti, izrađenih u programskom jeziku *C#,* koje su u interakciji sa *Unity-om* i zaslužne za sve funkcionalnosti.

Ova tema detaljno opisuje sinergiju programskog jezika i softvera za dizajn i izradu računalnih igra. To je korak više u programiranju gdje programer napisani kod može u virtualnom svijetu vidjeti kako funkcionira, odnosno kod koji se primjenjuje na objekte postaje vidljiv kroz pokrete, boje i razne dizajne objekata. Kod postaje nešto vidljivo što više nije apstraktno nego pristupačno puno široj populaciji, na temelju čega se i sam može slikovito objasniti.

Motivacija za odabir teme je zainteresiranost za računalne igre, programiranje, ali i za učenjem novih stvari. Na ovu temu gledam kao mogućnost i priliku da proširim znanja iz programiranja i dizajna, odnosno da na neki način slikovito rečeno *oživim* programski kod, te dizajn koji je vezan uz njega. Jedan dio motivacije je i znatiželja jer se sa programiranjem može napraviti gotovo sve što želimo, za što imamo volje, vremena i znanja.

2. Metode i tehnike rada

Za izradu računalne igre kao i svega ostalog potrebna je dobra priprema, što vezano uz teorijsku podlogu, a što vezano uz programsku potporu. Stoga su za početak proučeni desetci znanstvenih radova, članka, videa i priručnika. Prvo je bilo potrebo nakon proučenih izvora odabrati najkorisnije i najpovezanije sa tematikom ovog rada. Nakon toga iz odabranih izvora izvući najbitnije u smislu teorijske pozadine vezane uz načine pristupa izrade 2d računalne igre utrka, načina i mogućnosti oblikovanja modela levela, mapa, odnosno svih objekata koji su uključeni u cjelinu. Nakon sakupljanja informacija, bilo je potrebno istražiti tehnologiju tj. Platformu *Unity* koja služi kao okružje za izradu kompletne računalne igre. Kao platforma za izradu igrica jako je kompleksna i napredna te je bilo potrebno savladati sve potrebne funkcionalnosti kako bi proces izrade bio što lakši i da bi finalni proizvod bio što bolji.

Unity je kompaktna platforma koja nudi integraciju svih vrsta medija u jednu cjelinu te mogućnost njihova povezivanja i manipulacije. Sa programerske strane moguće je razvijati skripte u programskom okruženju *Microsoft Visual Studio* i programskom jeziku *C#.* Postoje dodatci za *MS Visual Studio* koji uvelike olakšavaju rad sa skriptama vezanim za *Unity* jer sadrže predefinirane klase i funkcije povezane sa radom nad objektima.

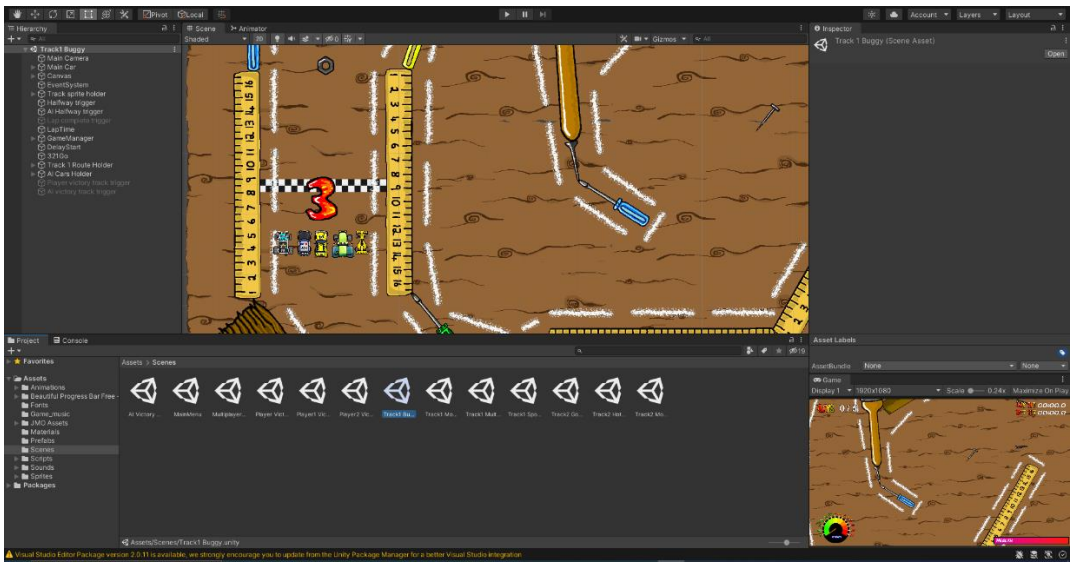
Dizajn računalnih igrica je velika zasebna tema, o kojoj se može napisati zaseban rad, te su zbog toga većina grafičkih komponenata preuzete sa internetskih izvora. Dio koji nije preuzet napravljen je uz pomoć besplatnih programskih alata kao što su *Audacitay* koji služi za izradu zvukova, muzike i posebnih efekata i *Maketext.io* stranica za izradu i uređivanje tekstualnog sadržaja.

3. Platforma Unity

U ovom dijelu objašnjen je i prikazan način rada *Unity-a*. Prikazane su najvažnije funkcionalnosti koje su bile potrebne za izradu ovog projekta. *Unity* kao platforma za izradu računalnih igrica jako je kompleksan i sofisticiran skup alata od kojih je većina objašnjena ili u ovom dijelu ili u kasnijim dijelovima gdje je detaljnije prikazan razvoj pojedini dijelovi igrice.

3.1. Korisničko sučelje

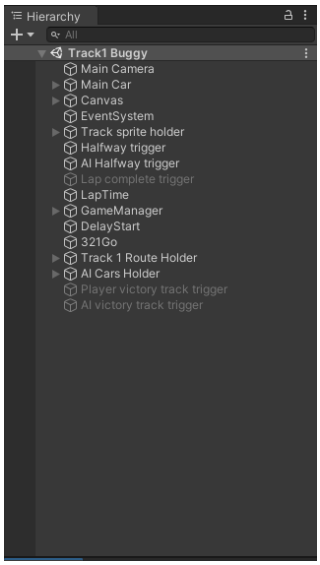
Korisničko sučelje platforme *Unity* nudi visoku razinu prilagođavanja što omogućava korisniku da posloži prozore onako kako njemu odgovara. Svaki prozor predstavlja određenu funkcionalnost. Kao što je spomenuto moguće ih je ukloniti i dodavati, te micati, povećavati i smanjivati. Slika 1 prikazuje konfiguraciju prozora, odnosno funkcionalnosti koje su služile prilikom rada. Svaka od njih je u nastavku potanje opisana.



Slika 1 Korisničko sučelje (Vlastita izrada)

3.2. Opis radnih prozora

Prozor *Hierarhija (eng. Hierarchy)* prikazan na slici 2 omogućuje sortiranje svih objekata koji se koriste u određenoj *Sceni (eng. Scene)* koja predstavlja određeni level, odnosno područje igrice koje trenutno uređujemo. Igrica je podijeljena u više *Scena (eng. Scenes)* koje omogućuje jednostavnije modeliranje zasebnih levela, te se na kraju sve *Scene* povezuju u jedan gotov proizvod[1]. U *Hijerarhiji* se nalazi sva korištena multimedija određenog levela, odnosno svi objekti koje možemo micati po *Sceni,* te manipulirati njima kroz skripte.



Slika 2 Prozor Hierarhija (Vlastita izrada)

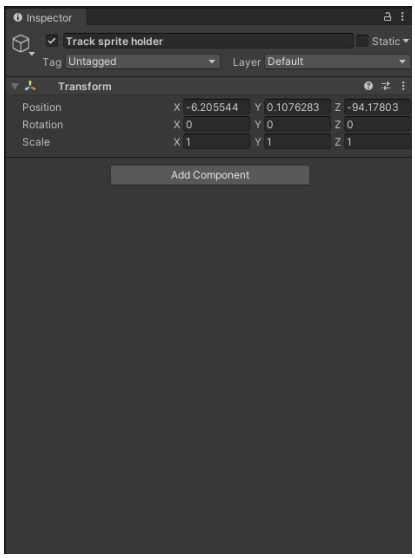
Sljedeći prozor je *Projekt (eng. Project)* u kojem se nalaze sve datoteke ikad stvorene vezane uz trenutni projekt. U njemu se nalazi *Glavni folder Assets.* U njemu se stvara sve što koristimo u igrici. Kroz taj prozor može se direktno mijenjati folder projekta kao i ako se nešto promjeni van tog prozora, odnosno van *Unity-a* promjene će odmah biti vidljive i u njemu [1]. Prikazano na slici 3.

Slika 3 Prozor Projekt (Vlastita izrada)

Prozor Scene, prikazan na slici 4 omogućuje uređivanje levela. Zbog dvodimenzionalnosti igrice na *Sceni* su u određenim trenucima vidljive dvije osi x i y, po kojima se objekti pomiču. Cijeli koncept *Scena* bazira se na različitim dimenzijama čija je upotreba objašnjena i prikazana kasnije u radu. Važno je za istaknuti kameru pomoću koje određujemo što igrač tj. Korisnik vidi u datom trenutku i *Slojeve (eng. Layers)* pomoću kojih se određuje koji objekti budu u pozadini, iza nje ili na vrhu. Opcija *Gizmos,* koja se može aktivirati i deaktivirati (Na slici 4 je deaktivirana) nudi vizualnu reprezentaciju x i y osi, granice objekata i mnoge druge pojedinosti vezane uz grafiku, objašnjene kasnije u radu.

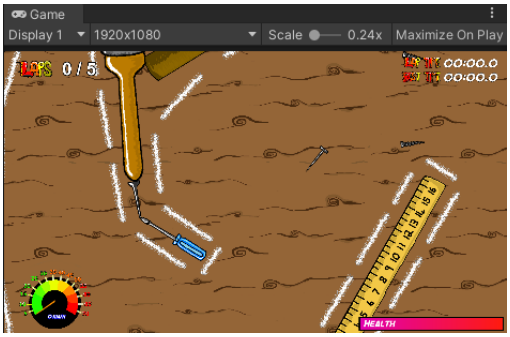
Slika 4 Prozor Scene (Vlastita izrada)

Jedan od najkorištenijih prozora uz prozor *Scene* je prozor *Inspektor (eng. Inspector).* U njemu su prikazane sve moguće postavke pojedinog objekta. Kada kliknemo na željeni objekt bilo u prozoru Scene ili Hijerarhiji, otvara se niz postavki kojima možemo konfigurirati njegove atribute (npr. pozicija, izgled, razne varijable, fizika itd.). Sadržaj prozora je za svaki objekt drugačiji, ali na slici 5 prikazane su osnove postavke osi koje su iste za svakog.



Slika 5 Prozor Inspektor (Vlastita izrada)

Uz posljednji prozor koji predstavlja prozor *Igre (eng. Game)* postoje kao što je prije navedeno i ostali koji će biti pokazani i objašnjeni u nadolazećim poglavljima. Dakle prozor *Igre* prvenstveno omogućuje pregled i testiranje napravljenog levela. Uz testiranje nudi opcije podešavanja rezolucije u kojoj će se igra testirati isto kao i opciju *Maximize On Play* koja uveća prozor preko cijelog ekrana za vrijeme testiranja.



Slika 6 Prozor Igre (Vlastita izrada)

4. Razvoj auta i AI

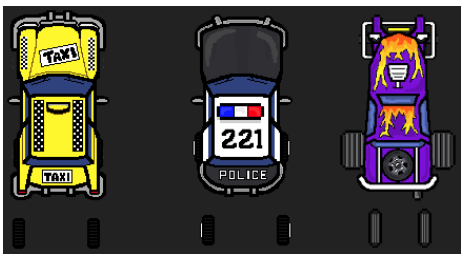
Proces razvoja virtualnog auta, odnosno kontrolera kojeg igrač koristi tijekom igranja, može se podijeliti u 3 faze. Faza dizajna i animiranja odnosi se na odabir i primjenu dizajna koji će se koristit, što je važno zbog kasnijeg animiranja. U ovom dijelu opisani su načini izrade, odnosno implementacije i animiranja dizajna u *Unity-u*. Sljedeća faza je implementacija mehanika, koja se odnosi na pisanje skripti u *programskom jeziku C# tj.* Programskom okruženju *MS Visual Studio.* U ovom dijelu detaljno su prikazana i objašnjena dva slična postupka jer igrica ima mogućnost igranja u jedan (eng. *Singleplayer*) i dva igrača (eng. *Multiplayer*). Posljednja je faza razvoja umjetne inteligencije (eng. *Artificial intelligence, skraćeno AI*) koja služi za kompjuterski upravljanje suigrače u verziji gdje igrač igra sam jer u drugoj verziji *AI* nije uključen.

4.1. Dizajn i animiranje auta.

Općenit pojam dizajna je jako važan za računalnu igricu jer to ono što igrač prvo vidi i čuje. Dizajn je podijeljen na grafički i zvukovni te je shodno tome objašnjen i prikazan. Nakon toga je u zasebnom poglavlju razrađeno animiranje grafičkog dizajna.

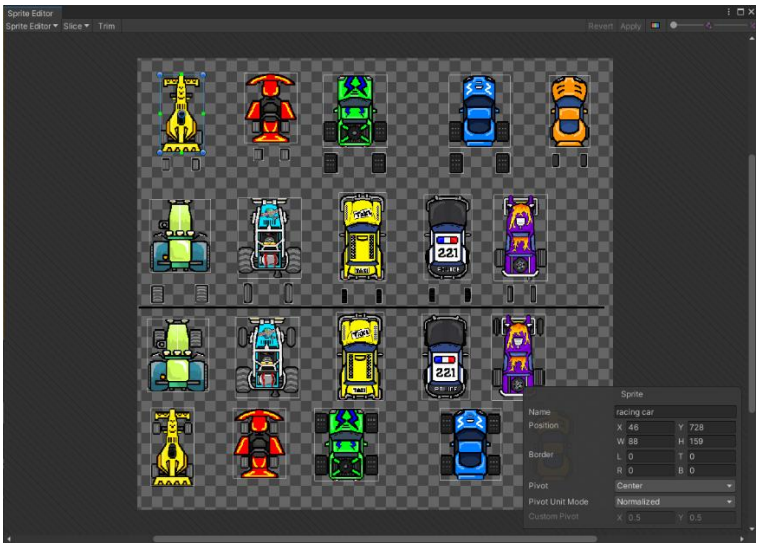
4.1.1. Grafički dizajn auta

Ovo poglavlje je jednako kompleksan dio kao i programerski dio te je stoga dizajn auta preuzet sa interneta. U nastavku prikazani su načini implementacije i konfiguracije dizajna u *Unity-u*. Dizajn auta sastoji se od dva dijela (slika 7)[multimedija1], prednjih guma i ostatka. Podijeljen je na taj način zbog kasnijeg animiranja guma na način da se mogu vizualno rotirati lijevo i desno ovisno unosu igrača.



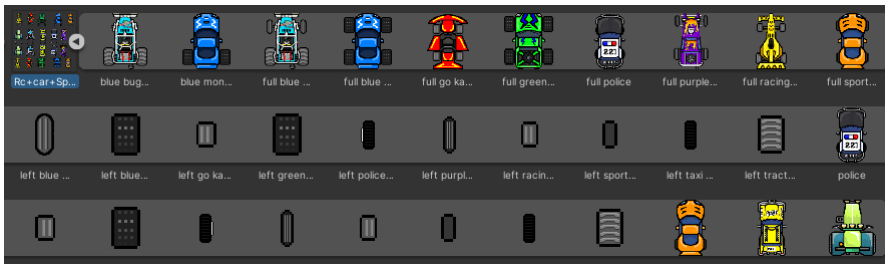
Slika 7 Neki od modela auta (Vlastita izrada)

Prije samog početka rada sa modelima auta, potrebno ih je uvesti u *Unity,* a najjednostavniji način je prevući željeni sadržaj u prozor Projekt u folder *Assets.* Isti postupak je za svu ostalu multimediju te se neće više ponavljati kroz rad. Nakon što se željena slika uveze potrebno ju je urediti u posebnom prozoru *Sprite Editor* prikazan na slici 8. Taj postupak je potreban da se izrežu nepotrebni dijelovi te da model ima što manje praznog prostora oko sebe jer on utječe na kasnije interakcije sa ostalim objektima. Svi grafički objekti nazivaju se *Sprites:*

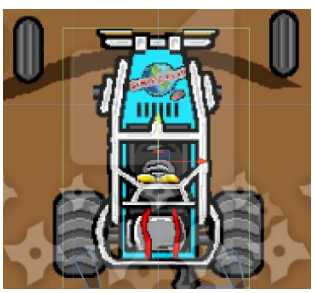


Slika 8 Prozor Sprite Editor (Vlastita izrada)

Nakon toga, u prozoru Projekt, modeli su prikazani odvojeno i kao takvi mogu se dodavati u Hijerarhiju, odnosno Scenu (slika 9 i 10). S time su grafički modeli spremni za konfiguracije i manipulacije koje se mogu vršiti putem prozora Inspektor ili zasebnih skripti.



Slika 9 Izrezani modeli (Vlastita izrada)



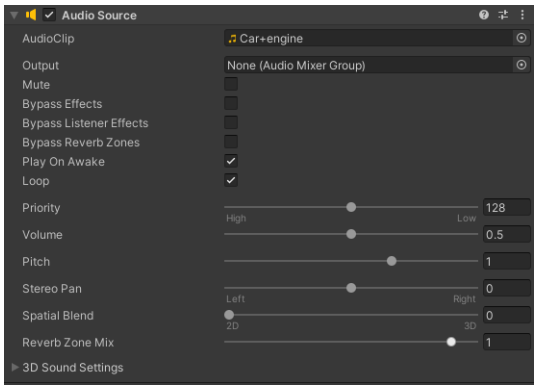
Slika 10 Prikaz modela auta na Sceni (Vlastita izrada)

4.1.2. Dizajn zvuka

Svi zvukovi, vezani uz auto, preuzeti su sa *Unity Assets Store* službene stranice koja je dio *Unity-a* i posebne stranice *artlist.io* koja nudi specijalne efekte i ostale zvukove [multimedija3]. U *Unity Assets Store*-u nalaze se sve vrste multimedije koje su ostali korisnici napravili te se mogu kupiti ili preuzeti besplatno [multimedija2], dok je druga stranica besplatna verzija.

Zvučni efekti od kojih se sastoji model auta su zvuk motora, zvuk skretanja i zvuk eksplozije. Isto kao i ostala multimedija zvuk se može konfigurirati putem prozora Inspektor ili zasebnih Skripti. Neke od važnijih postavki su volumen i mogućnost aktiviranja ili deaktiviranja dva faktora: pokretanja zvuka s početkom igrice i beskonačno pokretanje zvuka. S tim opcijama vrlo se realno može prikazati rad auta i njegov odnos s okolinom, što je detaljnije prikazano u kasnijem poglavlju.

Unity ima vlastiti *(eng. Namespace) UnityEngine* koji omogućuje glavnu komunikaciju sa objektima. Specifično za zvukove koristi se tip varijable *AudioSource* preko kojeg se može pristupiti svim funkcionalnostima prikazanim na slici 11. Dakle, cijeli proces funkcionira na način da u određenoj skripti definiramo *public* varijablu tipa *AudioSource*, jer se sve *public* varijable, u prozoru Inspektor (slika 11), vide kao postavke u koje potom mogu dodati željeni objekt, u ovom slučaju zvuk, i zatim dalje u kodu raditi razne manipulacije nad njim npr. Stop i Play (slika 12).



Slika 11 Postavke AudioSource (Vlastita izrada)

```
if (isPaused)
{
    Time.timeScale = 0f;
    engine.Stop();
    policeCar.Stop();
    taxi.Stop();
    tractor.Stop();
    racingCar.Stop();
}
else
{
    Time.timeScale = 1f;
    engine.Play();
    policeCar.Play();
    taxi.Play();
    tractor.Play();
    racingCar.Play();
}
```

Slika 12 Prikaz manipulacije zvukom u kodu (Vlastita izrada)

4.1.3. Animiranje

Nakon implementirane i konfigurirane grafike dodane su animacije kako bi sve izgledalo što uvjerljivije funkcioniranju auta. Rad sa animacijama moguć je na dva načina. Prvi lakši način, ali nešto ograničavajući je da se preuzmu gotovi predlošci [multimedija4], koji se u *Unity-u* nazivaju *Prefabs,* i uz pomoć skripti pokrenu na željenoj lokaciji u *Sceni*. *Prefabs* omogućuje da se bilo koji objekt spremi sa svim svojim svojstvima i konfiguracijom kao predložak kojeg možemo kasnije koristit na drugim područjima igre npr. na drugoj sceni. Slika 13 prikazuje mogućnost da se objekt jednostavno aktivira ili deaktivira i način na koji je to iskorišteno u skripti. Slika 14 prikazuje mogućnost da se određeni objekt stvori na željenom mjestu u željeno vrijeme, što u ovom slučaju predstavlja eksploziju.

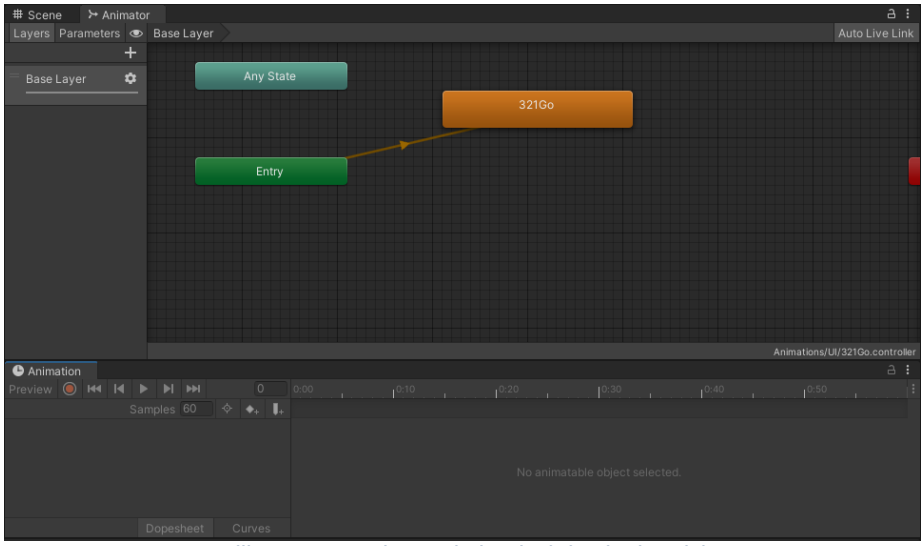
```
if (PlayerVictoryTrigger.currentLapPlayer == 1)
{
    leftFire.SetActive(true);
    rightFire.SetActive(true);
}
```

Slika 13 Aktivacija i deaktivacija objekta (Vlastita izrada)

```
if (currentHealth <= 0)
{
    Instantiate(explosion, transform.position, transform.rotation);
    gameObject.SetActive(false);
    carExplosion.Play();
    GameManager.instance.KillPlayer();
}
```

Slika 14 Stvaranje objekta na željenoj lokaciji (Vlastita izrada)

Drugi način je izrada vlastite animacije. Prije objašnjenja potrebno je spomenuti *Particle System* koji je djelomično korišten u animaciji. Dakle, to je sistem koji za svoj rad koristi mnogo komponenata, a glavne su *izvor (eng. emitter)* koji kreira čestice (npr. dima), *animator (eng. animator)* koji definira njihovo ponašanje i varijacije tokom vremena i *eng. renderer* koji definira njihov izgled i parametre prikaza [1]. Navedeni *Particle System* korišten je za izradu dima koji izlazi iz auspuha auta. Dio oko izrade vlastite animacije omogućuju funkcionalnosti s kojima radimo putem dva prozora prikazana na slici 15. Prozor *Animator* omogućuje kreiranje vlastitog slijeda prelaza animacija, koji su uvjetovani stanjem određenih varijabli. Na primjer u izmišljenoj skripti zadana je varijabla imena *prolaz* tipa *bool* i u animatoru je zadano da se iz stanja mirovanja prelazi u stanje kretanja naprijed (u pozitivnom smjeru osi y) kada je varijabla jednaka *true*. Tako se prelazi iz stanja u stanje, koje može biti reverzibilno, i aktivira određena animacija. Drugi prozor služi da se ručno izrađene slike, koje predstavljaju npr. pokrete ruku, spoje u jednu animaciju konfigurirajući željeni broj slika u sekundi.



Slika 15 Prozori za animiranje (Vlastita izrada)

4.2. Implementacija mehanika

Mehanike računalne igre čine ju zanimljivom i to je dio gdje se kao što je ranije spomenuto dizajn „oživljava“ i pomoću pritiska određene tipke ili aktiviranja okidača na mapi dešavaju pokreti objekata na *Sceni*. U ovom dijelu objašnjen je i prikazan razvoj svih mehanika koje su bile potrebne kako bi se auto, odnosno objekt auto mogao kretati.

4.2.1. Teorijska pozadina funkcioniranja računalne igre

Za početak prije daljnje razrade potrebno je objasniti nekoliko koncepata pod koje spadaju arhitektura i način funkcioniranja igre. Arhitektura računalne igre sastoji se od *Upravljača događajima (eng. Event Handler)* koji zajedno sa *unosom (eng. Input)* korisnika omogućavaju prijenos, odnosno reprezentaciju akcija igrača računalnoj igri. Nadalje akcije su povezane sa logikom igre *(eng. Game Logic)* koja je zaslužna za sve prikazane rezultate igrice kao što je u ovoj npr. broj krugova utrke. Grafika i zvukovi osiguravaju muziku, zvučne efekte, slike i drugo objektima koje igrač vidi u igri te koji su pohranjeni u *datoteci levela (eng. Level Data)* što u ovom projektu predstavlja sve ono što se nalazi u *Hijerarhiji* pojedine *Scene*. I na kraju *dinamički modul (eng. Dynamic Module),* koji predstavlja programski dio u Skriptama, konfigurira dinamičko ponašanje objekata [2]. Način funkcioniranja igre može se prikazati kao petlja koja u svakoj iteraciji procesira unos igrača te se stanja igre i virtualni svijet mijenjaju na temelju programske logike sve dok igra ne završi [2].

4.2.2. Razvoj mehanika kretanja auta

Nakon što je završen dizajn auta može se prijeći na izradu programske logike. U *Unity-u* postupak implementacije bilo koje skripte funkcionira na način da se prvo stvori u glavnom folderu *Assets,* a potom pridruži željenom objektu.

Mehanika kretanja auta razvijena je pomoću dvije glavne skripte od kojih je jedna zadužena za dinamiku karoserije auta i stražnjih guma, dok je druga zadužena za rotiranje prednjih guma. Programski dio prve skripte naziva *MainCar* prilično je kompleksan te su najvažniji dijelovi pojašnjeni u nastavku.

Potrebno je za početak razjasniti 4 glavne funkcije koje direktno utječu na dinamiku igre i dio su *UnityEngine-a*. To su funkcije *Awake(), Start(), Update() i FixedUpdate()*. Funkcija *Awake()* poziva se prije početka igre, odnosno određenog levela koji sadrži tu skriptu i u njoj se inicijalizira početno stanje. *Start()* se izvodi s početkom levela, odnosno prvom slikom *(eng. Frame)* i u njoj se na primjer pridjeljuju objekti varijablama. Nakon toga po potrebi se koriste funkcije *Update() i FixedUpdate().* Glavna razlika između njih je da se funkcija *Update()* izvodi jednom po slici u minuti, dok se *FixedUpdate()* izvodi nijednom, jednom ili više puta. Stoga je u funkciji *Update()* implementirana logika igre kao što je grafika i zvuk, a fizika igre kao što je kretanje objekta u *FixedUpdate().*

Kako bi se u funkciji *FixedUpdate(),* mogle izraditi programske upute računalu na koji način da pomakne objekt na *Sceni,* potrebno je dohvatiti unos korisnika. To je izvedeno pomoću klase *Input* i njezine metode *GetAxis* koja dohvaća vrijednost tražene osi. Unos korisnika može se predefinirati na način da se za određene tipke predefinira naziv npr. za tipke a i d naziv *Horizontal*, tipku w *Accelerator* i za tipku s *Reverse*. Nadalje je prikazan i objašnjen programski kod (slika 16). Za početak se u varijablu *turningForce* sprema unos korisnika i koji se isto tako provjerava kroz dva uvjeta i na temelju njih izračunava varijabla *speed*. Zatim se računa varijabla *direction* i to na način da se gleda rezultat metode *Mathf.Sign()* koja vraća 1 ako je rezultat pozitivan ili 0, a -1 ako je rezultat negativan. Rezultat se dobiva pomoću funkcije *Vector2.Dot(rb.velocity, rb.GetRelativeVector(Vector2.up))* koja prima dva parametra tj. dva vektora, u ovom slučaju prima vektor smjera kretanja objekta i vektor usmjerenosti objekta. Ako vektori imaju isti smjer vraća se 1 u suprotnom -1 [3]. Nakon toga mijenja se rotacija objekta po z-osi, te zbog toga auto ima efekt skretanja. Na kraju se funkcijom *AddRelativeForce()* dodaje snaga/brzina objektu i to prvi puta u smislu ubrzanja i drugi puta u smislu skretanja, odnosno da objekt u zavojima izgleda prirodno.

```
Unity Message | 0 references
void FixedUpdate()
{
    turningAmount = -Input.GetAxis("Horizontal");
    speed = 0f;
    if (Input.GetAxis("Accelerator") > 0)
    {
        speed = Input.GetAxis("Accelerator") * forward;
    }
    else if (Input.GetAxis("Reverse") < 0)
    {
        speed = Input.GetAxis("Reverse") * reverse;
    }

    direction = Mathf.Sign(Vector2.Dot(rb.velocity, rb.GetRelativeVector(Vector2.up)));
    rb.rotation += turningAmount * turningForce * rb.velocity.magnitude * direction;
    rb.AddRelativeForce(Vector2.up * speed);
    rb.AddRelativeForce(-Vector2.right * rb.velocity.magnitude * turningAmount / 2);
}
```

Slika 16 Mehanika kretanja auta (Vlastita izrada)

Druga skripta vezana uz mehaniku auta odnosi se na skretanje prednjih kotača te je mnogo jednostavnija od prethodne (slika 17). Jedine novosti su funkcija *LateUpdate()* koja se automatski poziva nakon svih ostalih izvršenih *Update()* funkcija [3] i vrijednost koja se pridružuje varijabli *angle* koja predstavlja količinu rotacije kotača. Ta vrijednost dobije se tako da naredba *transform.localEulerAngles* dohvati trenutnu poziciju objekta roditelja što je u ovom slučaju cijeli auto, zatim se u varijabli *angle* mijenja vrijednost osi z sa unosom korisnika, te ta nova vrijednost postaje nova rotacija kotača.

Mehanika auta u *Multiplayer* verziji, gdje nema umjetne inteligencije nego se utrkuju dvije stvarne osobe, razvijena na identičan način samo što je bilo potrebno napraviti dvije kopije iste skripte. Jedina stvar koju je bilo potrebno promijeniti su nazivi za predefinirane tipke koje igrač mora stisnuti kako bi se objekt pomaknuo, što je zahtijevalo dodavanje novih instanci tipki u opcijama.

```
Unity Message | 0 references
void Update()
{
    wheelAngle = -Input.GetAxis("Horizontal") * maxWheelAngle;
}

Unity Message | 0 references
void LateUpdate()
{
    angle = transform.localEulerAngles;
    angle.z = wheelAngle;
    transform.localEulerAngles = angle;
}
```

Slika 17 Mehanika kotača auta (Vlastita izrada)

4.3. Razvoj AI

Umjetna inteligencija se u današnje vrijeme koristi u svim aspektima računalne igre. Modeliranje iskustva igrača je pojam koji se odnosi na sve aspekte igre, a vođen je umjetnom inteligencijom. To je vrsta modeliranja koja prikuplja podatke na temelju akcija igrača i zatim te podatke koristi kako bi na primjer izradila posebne razine igre ili ponudila igraču vrstu objekta koje najviše koristi. Banalan primjer je ako igrač ima mogućnosti odabira kruga, trokuta i kvadra i određen broj puta odabere trokut, umjetna inteligencija ubuduće prikazuje više trokuta i time poboljšava iskustvo igranja [4].

Što se tiče računalnih igra utrkivanja veliki naglasak je na mehanici vozila, izradi puta kojim se ono vozi i načina na koji dolazi do cilja, usmjeravanih umjetnom inteligencijom. Izrada mehanike vozila i način na koji auto, odnosno umjetna inteligencija dolazi do cilja vođeni su sa nekoliko teorija i pristupa.

4.3.1. Teorije i pristupi razvoja AI

Tri pristupa vezana su uz razvoj umjetne inteligencije. *Pristup optimizacije* koristi algoritam optimizacije kojim se podešavaju aspekti igre tako dugo dok umjetna inteligencija ne postigne željenu optimalnu razinu, što ponekad zna dovesti do toga da se ne ponaša *normalno*. *Pristup inovacije* za razliku od *pristupa optimizacije* ima za cilj, umjesto razvoja optimalne umjetne inteligencije, razviti umjetnu inteligenciju koja se ponaša zanimljivo ili kompleksno. Zadnji od tri pristupa je *Pristup imitacije* gdje je u cilju kroz niz testova imitirati stvarnu osobu i njezine akcije, ali što paralelno može dovesti do toga da umjetna inteligencija imitira drugu umjetnu inteligenciju [5].

Način kretanja auta tj. umjetne inteligencije obrazložen je sa *algoritmom za rješavanje problema traženja puta (eng. pathfinding problem)* i njegovim modifikacijama. Ideja algoritma je da postoji sustav detekcije kolizije boja, odnosno sustav koji na primjer odredi da se dijelom mape obojanom bijelom bojom ne može voziti, a crnom može od koje je napravljena trkača staza. Modifikacija tog algoritma iznosi način određivanja najkraćeg puta. Postupak je taj da se računa udaljenost, tj. put između točaka te ako nije presječen bojom kolizije sljedeća točka postaje izvorna i cijeli postupak se ponavlja. Zadnja modifikacija početnog algoritma unapređuje cijeli proces još više gdje se eliminira testiranje svih mogućih točaka, odnosno smjerova kretanja auta i već se testiraju samo tri točke koje se nalaze ispred auta, ravno ispred, lijevo i desno. Ako jedna od tih točaka ima koliziju sa bojom kolizije tada se njoj mijenja boja po istom principu kao i sa stazom i samim time taj smjer postaje blokiran [6].

4.3.2. Implementacija AI

Za izradu umjetne inteligencije u ovom radu korištena je najčešća metoda implementacije, a to je kretanje vozila pomoću *navigacije prolaznih točaka (eng. Waypoint navigation).* To je metoda koja se sastoji od postavljanja prolaznih točaka po cijeloj mapi kako bi se po njima mogao kretati programski upravljan auto. Najveća mana ove metode je ta što je potrebno izdvojiti poviše vremena kako bi se sve prolazne točke ručno postavile.

Prvi korak je izrada skripte *AIPathWay* zadužene za omogućavanje izrade rute koja se sastoji od mnogo prolaznih točaka. Skripta se sastoji od tri funkcije, a to su *Awake(), Update() i DrawLine()*. U funkciji *Awake()* se kao i kod izrade auta inicijalizira vrijednost varijabli, odnosno u ovom slučaju varijable *route* koja predstavlja listu objekata tj. prolaznih točaka. U funkciji *Update()* nalazi se jedna specifična funkcionalnost vezana za *Unity,* a to je *predprocesorska naredba UNITY_EDITOR (slika 18).* Ta naredba je potrebna uvijek kada se koristi imenski prostor *UnityEditor* jer se on briše prilikom kompajliranja koda i pokretanje umjetne inteligencije ne bi bilo moguće. Unutar te naredbe je kod koji služi dodavanju objekata, koji predstavljaju rutu umjetne inteligencije, u listu i na kraju grafičko prikazivanje iste. Funkcija *DrawLine()* služi grafičkom prikazu rute jer prolazi kroz listu točaka i crta ravne linije od prošle do sljedeće točke. Vršenje te radnje traje tako dugo dok se ne dođe do zadnje točke u listi te se zadnja i početna spoje i povežu u cjelinu.

```
Unity Message | 0 references
void Update()
{
#if UNITY_EDITOR
    if (!EditorApplication.isPlaying)
    {
        if (route == null)
        {
            route = new List<GameObject>();
        }
        route.Clear();
        foreach (Transform node in transform)
        {
            route.Add(node.gameObject);
        }
    }
    if (route != null && route.Count > 1)
    {
        if (drawLine)
        {
            DrawLine();
        }
    }
#endif
}
```

Slika 18 UNITY_EDITOR (Vlastita izrada)

Drugi korak je izrada skripte *AICars* koja objektu na koji je dodana daje snagu kretanja, odnosno simulira fiziku kretanja auta. Sastoji se od četiri funkcije, a to su *Start(), Update(), FixedUpdate() i Movement().* Funkcija *Start()* zadužena je za početnu inicijalizaciju varijabli. Funkcija *Update()* s obzirom brzine promjene lokacije objekta modificira glasnoću, odnosno zvučni efekt rada motora pomoću varijable *pitch. U funkciji FixedUpdate()* je kao i u prethodnim primjerima implementirana fizika objekta. To se odrađuje pomoću metode *AddForce()* koja objekt dodaje određenu snagu i pomiče ga u smjeru osi y. Zadnja funkcija *Movement()* je bitna jer se koristi u trećoj skripti objašnjenoj u sljedećem odlomku i zajedno s njom omogućuje kretanje objekta po mapi (slika 19). To radi na način da prima dva argumenta, smjer skretanja koji može biti lijevo ili desno i varijablu *isPeddlePressed* tipa *bool* koja ako je *true* signalizira skripti da doda snage objektu, odnosno da ga krene pomicati. Ako je smjer skretanja lijevo rotira objekt lijevo sa metodom *Rotate()* i obratno za desno.

```
1 reference
public void Movement(string turning, bool isPeddlePressed)
{
    if (turning == "left")
    {
        transform.Rotate(Vector3.forward * (currentSpeed + currentSpeed));
    }
    else if (turning == "right")
    {
        transform.Rotate(Vector3.forward * -(currentSpeed + currentSpeed));
    }
    if (isPeddlePressed)
    {
        isMoving = true;
    }

    currentSpeed = curSpeed.magnitude / turnSpeed;
    curSpeed = new Vector2(rb.velocity.x, rb.velocity.y);
}
```

Slika 19 Metoda Movement (Vlastita izrada)

Treći korak je izrada zadnje skripte *AIMovement* koja sinkronizira rad prethodne dvije i omogućuje kretanje auta po mapi. Dakle, u funkciji *Strat()* se inicijalizira varijabla *driver* koja predstavlja objekt auta i može pristupiti svim funkcijama i varijablama iz skripte *AICars.* Nadalje ako postoji ruta zadanog objekta tada se inicijaliziraju i ostale važne varijable za rad umjetne inteligencije. Te varijable su *wayPoint* preko koje se komunicira sa varijablama iz skripte *AIPathWay,* varijabla *route* koja postaje lista prolaznih točaka objekta, varijable *currentIndex* koja ima ulogu brojača, *totalWayPoints* predstavlja ukupni broj prolaznih točaka i *currentWayPoint* predstavlja trenutnu prolaznu točku na kojoj se objekt auta nalazi (slika 20). Zatim se u funkciji *Update()* vrši niz provjera stanja varijabli i pozicije objekta. Najvažnije varijable su *turning,* koja predstavlja smjer u kojem je pozicioniran objekt auta (lijevo ili desno) i *isAccPressed* koja je inicijalno zadana na *false* i postaje *true* u trenutku pokretanja levela. Zadnja funkcija je *FixedUpdate()* u kojoj se zapravo vrši finalna radnja pokretanja objekta, odnosno ovdje umjetna inteligencija pokreće auto. Ako postoji zadana ruta auta objekt se pomoću naredbe *driver.Movement(turning, isAccPressed)* kreće u određenu stranu.

```
Unity Message | 0 references
void Start()
{
    driver = GetComponent<AICars>();
    if (AIPathHolder != null)
    {
        driver.isAIMovement = true;
        wayPoint = AIPathHolder.GetComponent<AIPathWay>();
        route = wayPoint.route;
        currentIndex = 0;
        totalWayPoints = route.Count;
        currentWayPoint = route[0].transform;
    }
}
```

Slika 20 Inicijalizacija važnih varijabli (Vlastita izrada)

```
Unity Message | 0 references
private void FixedUpdate()
{
    if (driver.isAIMovement)
    {
        driver.Movement(turning, isAccPressed);
    }
}
```

Slika 21 Naredba pokretanja AI (Vlastita izrada)

5. Razvoj mapa i korisničkog sučelja

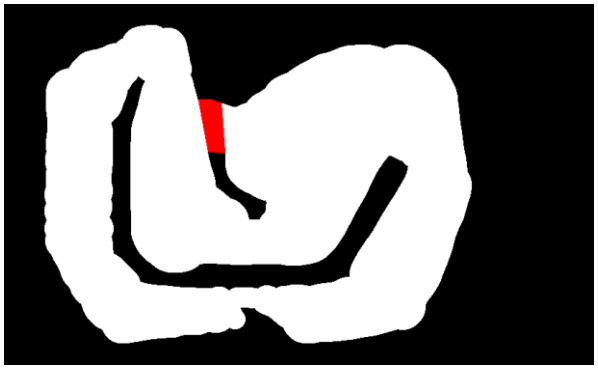
Iskustvo korisnika nije isključivo vezano uz dizajn i mehanike objekt koji predstavlja auto nego i mapa po kojoj se utrkuje, ali i način na koji su mu prezentirani podaci o stanju igre. U ovom poglavlju objašnjene u dvije razvijene mape i korisničko sučelje koje se mijenja ovisno o akcijama korisnika. Kako zadnja stavka navedene su i objašnjene dodatne mehanike koje su implementirane prilikom interakcije objekata i čine ovu računalnu igru utrkivanja posebnom.

5.1. Dizajn mapa

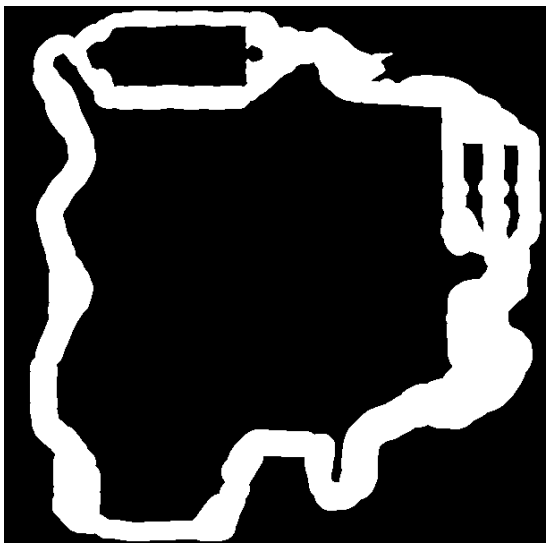
Uz sve dosad navedene stavke dizajn mape direktno utječe na iskustvo igranja korisnika. Postupak dizajniranja mapa za ovu računalnu igru bio je jednostavan i vođen mišlju da je potrebna jedna veća mapa i jedna manja, jedna mapa koja će biti zahtjevnija i traži od korisnika više iskustva, dok je druga poprilično jednostavna i na neki način upoznaje korisnika sa mehanikama. Na jednostavnijoj mapi je mnogo lakše savladati protivnike vođene umjetnom inteligencijom, dok je na drugoj obratno.

Uz jednostavan postupak postoji i nešto stručniji i mnogo zahtjevniji koji funkcionira na način da nudi korisniku određen broj različitih dizajna mapa koje zatim sam ocjenjuje. Dakle, umjetna inteligencija generira određen broj mapa i zatim korisnik pomoću dvije metode može ocijeniti mape. Prva metoda je ocjenom jedan do pet, a druga je jednostavnija gdje korisnik može odabrati je li mu se sviđa ili ne. Druga metoda je nešto bolja jer otklanja mogućnost pojave mapa koje se korisniku ne sviđaju, dok prva metoda ostavlja mogućnost pojave onih osrednje ocjenjenih mapa i na taj način se iskustvo igranja smanjuje s obzirom na drugu metodu [7].

Za potrebe rada razvijene su dvije mape kao što je ranije navedeno. Prva mapa je nešto manja sa više broja krugova, kompleksnija u smislu zavoja i iskustva koje iziskuje od igrača jer sadrži prepreke (slika 22). Dok je druga mapa veća, sa manje krugova i nema prepreka te ostavlja igraču mogućnost pogrešaka (slika23). Stavke koje direktno utječu na iskustvo igranja navedene su u kasnijem poglavlju *Interakcija objekata*.



Slika 22 Dizajn prve mape (Vlastita izrada)

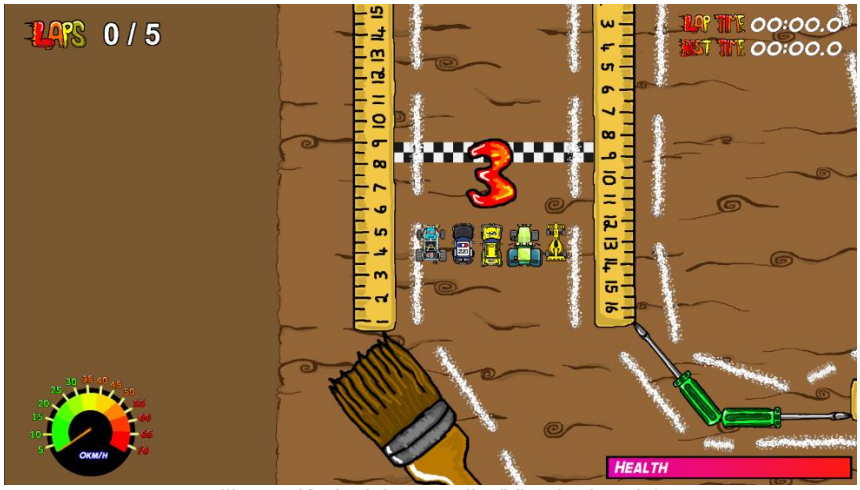


Slika 23 Dizajn druge mape (Vlastita izrada)

5.2. Dizajn korisničkog sučelja

Dizajn korisničkog sučelja prezentira korisnika sa posljedicama njegovih akcija. Ovdje je važno objasniti kako se dodatne funkcionalnosti reprezentiraju, a u sljedećem poglavlju je objašnjeno kako su one izvedene.

Korisničko sučelje se sastoji od 6 stavki (slika 24). Prva stavka je broj krugova u gornjem lijevom kutu koji prikazuje ukupan broj krugova i broj krugova koje je igrač završio. Druga i treća stavka su trenutno i nabolje vrijeme kruga. Četvrta stavka je u sredini i vidljiva je samo prije početka igre, a to je brojač sekundi to početka utrke. Peta stavka je brzinomjer u donjem lijevom kutu koji prikazuje brzinu izračunatu temeljem fizike objekta auta. Zadnja šesta stavka je pravokutnik koji prikazuje količinu života igrača. Zadnja stavka je detaljnije objašnjena u nastavku jer je posebna za ovu računalnu igru.



Slika 24 Korisničko sučelje (Vlastita izrada)

5.3. Interakcija objekata

Iskustvo i vještine u igranju računalnih igrica u današnje vrijeme variraju sve više i više zbog ubrzanog razvoja računalne industrije. Zbog tog razloga razvoj i konfiguracija svake računalne igre je zahtjevan posao jer je potrebno prilagoditi iskustvo igranja igračima [4]. Važno je razviti virtualno okruženje koje budi natjecateljski duh igrača i daje im osjećaj napretka u vještini igranja.

Navedene važnosti iz prethodnog odlomka implementirane su kroz interakciju objekta. Razvijena su dva sustava koja to omogućuju, a to su sustav brojanja krugova i vremena, te sustav koji simulira život auta. Oba su razvijena korištenjem komponenti *sudarača (eng. Colliders).* Objekti kojima je dodijeljena ta komponenta ne mogu proći jedan kroz drugog, odnosno u slučaju da je omogućena opcija *okidača (eng. Trigger)* služe kao detektori prolaska ili okidanja neke druge radnje.

Prvi sustav razvijen je tako što su tri objekta sa komponentama *Collider* postavljena na mapi. Objekti *LapCompleteTrigger i PlayerVictoryTrigger* postavljeni su na cilju, a objekt *HalfWayTrigger* na sredini staze. Funkcionira na način da su na početku staze *LapCompleteTrigger i PlayerVictoryTrigger* deaktivirani. Prolaskom objekta auta upravljanog od strane igrača kroz *HalfWayTrigger* aktiviraju se *LapCompleteTrigger i PlayerVictoryTrigger*, a *HalfWayTrigger* se deaktivira, dok se prolaskom objekta auta upravljanog umjetnom inteligencijom aktivira samo *PlayerVictoryTrigger.* To omogućava brojanje krugova i vremena na način da kada igrač napravi puni krug, ponovo prođe *LapCompleteTrigger i PlayerVictoryTrigger* koji su sada aktivirani skripte istih naziva pokrenu brojanje krugova, i vremena, određivanje najboljeg vremena i u slučaju da tri objekta auta, ne bitno kako su upravljani, prođu pet puta stazu utrka se završava i prikazuju se pobjednici.

Drugi sustav koji simulira život auta osmišljen je na način da kada se objekt auta upravljan igračem sudari sa preprekom (zid ili na stazi) ili objektom koji predstavlja policijski auto postepeno smanjuju životi sve dok ne ostane bez njih. Tada eksplodira i nakon određenog vremenskog perioda se ponovno stvori na istom mjestu. Implementiran je pomoću četiri skripte, animacije i zvučnog efekta. Sve četiri skripte na neki način međusobno komuniciraju te nema pravog redoslijeda objašnjenja. Sve na neki način započinje skriptom naziva *HealthManager* u kojoj se kao i u drugim u funkciji *Awake()* vrši inicijalizacija objekta. Zatim funkcija *Start()* ima ulogu reguliranja broja života i komunicira sa skriptom *HealthBar* na način da putem funkcije *SetMaxHealth()* puni pravokutnik sa životima. Funkcija *DamagePlayer()* služi postepenom smanjivanju života objekta auta i putem funkcije *SetHealth()* regulirala količinu života prikazanu u korisničkom sučelju. Kada je količina života manja ili jednaka nuli pokreće se animacija eksplozije, objekt auta se deaktivira te je nevidljiv, pokreće se zvučni efekt eksplozije i poziva se metoda *KillPlayer()* iz skripte *GameManager.* Ta metoda koristi korutinu kako bi simulirala ponovno stvaranje objekta auta nakon što je uništen. Funkcija *Reaspawn(),* definirana u skripti *HealthManager,* poziva se u skripti *GameManager* u korutini *RespawnCo().* Funkcionira na način da aktivira objekt auta, postavi maksimalan broj života i napuni objekt pravokutnika sa bojom koja predstavlja živote, a korutina taj cijeli proces odgodi za određen period vremena. Cijeli proces se ponavlja kada dođe do sudara objekt auta sa objektom koji na sebi ima skriptu *DamagePlayer* koja u tom trenutku poziva metodu *DamagePlayer().*

Prvi sustav je u *Multiplayer* načinu igranja izostavljen, dok je drugi sustav implementiran kroz zasebne skripte na siti način za svaki auto posebno. Time je maknut aspekt najboljeg vremena i ostavljeno igračima da se fokusiraju na što precizniju vožnju.

6. Razvoj izbornika i dodatnih zaslona

Kako bi računalna igra bila do kraja funkcionalna potrebni su izbornici preko kojih igrač vrši sve radnje odabira načina igre, auta i levela. Uz to implementirani su dodatni zasloni kako bi igraču bio jasan tok igre.

Izbornici od kojih se sastoji igrica su početni izbornik (slika 25), izbornik auta i levela (slika 26) i izbornik pauze (slika 27). Početni izbornik sastoji se od tri mogućnosti, a to su pokretanje *Singleplayer* opcije igranja, pokretanje *Multiplayer* opcije igranja i prekid programa. Ako se korisnik odluči za Singleplayer opciju igranja aktivira se prikaz izbora auta. Samim izborom auta pokreče se mapa za koju je taj auto određen. Odabir auta, odnosno mape i pokretanje levela odrađeno je u skripti *MainMenu* metodom *LoadScene(ime_scene)* koja učitava željenu *scenu.* Dakle svi izbornici i dodatni zasloni su zasebne *Scene* koje se pokreću klikom na određeni gumb. Ako se korisnik odluči za *Multiplayer* opciju prije početka igre prikazan je dodatni zaslon sa uputstvima za oba igrača. Na kraju u oba slučaja kada utrka završi prikazana je finala *Scena* koja sadrži rezultate pobjednika. U *Singleplayer* opciji igranja je to rang ljestvica tri najbolja vozača (slika 28), a u *Multiplayer* opciji je to prikaz naziva pobjednika (Player 1 ili Player 2) i njegov auto (slika 29). Ako korisnik želi pauzirati trenutnu igru, pritiskom na tipku *esc* otvara se izbornik pauze. Otvaranjem izbornika kroz skripte zaustavljaju se sve tekuće varijable, određeni zvukovi i sami objekti. U zaslonu pauze moguće je nastaviti igru, izaći u početni izbornik ili prekid programa. Odabirom nastavka igre sve varijable se nastavljaju mijenjati, a u slučaju povratka u početni izbornik sve potrebne varijable se resetiraju.

Slika 25 Početni izbornik (Vlastita izrada)

Slika 26 Izbornik auta i levela (Vlastita izrada)

Slika 27 Izbornik pauze (Vlastita izrada)

Slika 28 Singleplayer pobjeda (Vlastita izrada)

Slika 29 Multiplayer pobjeda (Vlastita izrada)

7. Zaključak

Prilikom postupka odabira platforme bilo je važno da nudi sve važne funkcionalnosti i dodatke za izradu 2D igre utrkivanja što *Unity* nudi. Uz rad na samoj platformi korišten je programski jezik *C#* i programsko okruženje *MS Visual Studio* koji je kompatibilan sa *Unity-om* te je zbog toga izrada skripti poprilično intuitivna. Kao platforma nudi mogućnosti implementacije vlastitih grafika i zvukova kao i preuzimanja gotovih sa službene *Unity Assets Store* stranice, gdje se mogu naći besplatni sadržaji ali i oni koji se naplaćuju.

Proces izrade igre može se podijeliti na tri glavne cjeline, a to su izrada auta i umjetne inteligencije, izrada mapa tj. staza po kojima će se auti voziti te korisničkog sučelja i razvoj izbornika koji to sve objedinjuju. Izrada auta i umjetne inteligencije su podjednako zahtjevni jer svaki od njih zahtjeva svoj sustav. Za razvoj auta bilo je potrebno implementirati grafiku i zatim tu grafiku putem skripti animirati i pokrenuti. To je moguće zahvaljujući tehnologiji rada s fizikom objekta koju *Unity* pruža. Za razvoj umjetne inteligencije bilo je potrebno proučiti postojeće teorije i metode oko izrade i potom odabrati najprimjereniju, kojom se ispostavila metoda *navigacije prolaznih točaka (eng. Waypoint navigation)*. Prilikom razvoja mapa, odnosno trkaćih staza bilo je potrebno proučiti postojeće teorije razvoja staza, njihove selekcije i načina na koji utječu na iskustvo igrača. Time je zaključeno da je igraču potreban izazov i mogućnost napretka, kao i mogućnost da se upozna sa mehanikama igre. U skladu s time razvijene su dvije staze, prva koja je kraća ima više krugova za odvoziti i više je izazovna, dok je druga staza veća ima manje krugova i jednostavnija je za korisnika da savlada mehanike igre. Preko korisničkog sučelja prezentirane su igraču njegove akcije u igri, kao što su broj odvoženih krugova, potrebno vrijeme za krug, brzina kojom se kreće i količina života auta. Implementacija količine života auta je posebna za ovu igru jer većina ostalih ima u cilju prestići protivnika ali interakcija auta i virtualnog svijeta nema nikakvog utjecaja na daljnje ishode, te može doći do monotonosti i lošeg iskustva igranja. Kako bi igraču bio jasan put od pokretanja računalne igre do utrke razvijeni su izbornici i posebni zasloni koji igraču daju uvid o trenutnom stanju igre ili služe kao izvor dodatnih informacija.

Postupak izrade ove računalne igre pokazao je mnoge aspekte koji moraju biti zadovoljeni kako bi finalna računalna igra bila što bolja. Nakon što su svi aspekti zadovoljeni postoji mnogo načina i mogućnosti dodatnih promjena i dodavanja određenih segmenata koji mogu računalnu igru učiniti još boljom.

8. Popis literature

[1] Will Goldstone, *„Unity Game Development Essentials“,* Preuzeto: 04.08.2021, http://ds.nashobmenfiles.com/fo/get/2655976/Packt_Unity_Game_Development_Essentials_2009-vsotik_ru.pdf

[2] Junfeng Qu, Yinglei Song, Yong Wei, *„Applying Design Patterns In Game Programming“,* Preuzeto: 05.08.2021, http://worldcomp-proceedings.com/proc/p2013/SER2752.pdf

[3] Unity Documentation, *„Unity User Manual 2020.3 (LTS)“* Preuzeto: 10.08.2021, https://docs.unity3d.com/Manual/index.html

[4] Georgios N. Yannakakis, *„Game AI Revisited“,* Preuzeto: 04.08.2021, https://dl.acm.org/doi/pdf/10.1145/2212908.2212954?casa_token=MUn8FBNi1UcAAAAA:Vjh53TLSuTvdyNnBsNv6tClD44H17FqYRmrQ_dlDHmEmpePmlOpxflU8YOFcIRnWboEwgrg4HjahX1o

[5] Julian Togelius, Renzo De Nardi, Simon M. Lucas, *„Towards automatic personalised content creation for racing games“,* Preuzeto: 07.08.2021, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4219051&casa_token=_eMppFoAjksAAAAA:AMEzkahrPdkvbiHO0CBOFyk7V7Vj6KBC0dIvBSFNIYbiyz0-2gru2sXVdbyM83TNLZt1DDXQ3S9aQw&tag=1

[6] Jung-Ying Wang, Yong-Bin Lin, *„Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms“*, Preuzeto: 07.08.2021, http://ijmlc.org/papers/82-A1090.pdf

[7] Luigi Cardamone, Daniele Loiacono, Pier Luca Lanzi, *„Interactive Evolution for the Procedural Generation of Tracks in a High-End Racing Game“*, Preuzeto: 08.08.2021, https://dl.acm.org/doi/pdf/10.1145/2001576.2001631?casa_token=hl8bk-8bHM0AAAAA:t1VeEqD3WhzhEzEwbSC7tfuajntc3HadGRVXHUUfuJ2pK8HtXi0pOdAYIQirFJzgk5YfJAznGf60_Sc

9. Popis izvora multimedije

Ovdje su navedeni izvori za sve grafike i zvukove koji su korišteni u izradi računalne igre, a preuzeti su sa interneta. Oznaka [multimedija(broj)] označava određeni izvor na način da ga razlikuje od izvora literature.

[multimedija1] OpenGameArt.org, Nigel67, Preuzeto: 13.08.2021, https://opengameart.org/content/racing-game-2d

[multimedija2] Unity Asset Store, Vertex Studio, Preuzeto: 14.08.2021, https://assetstore.unity.com/packages/audio/music/absolutely-free-music-4883

[multimedija3] Artlist, BOOM Library, Preuzeto: 18.08.2021, https://artlist.io/sfx/

[multimedija4] Unity Asset Store, Jean Moreno (JMO), Preuzeto: 16.08.2021, https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565

[multimedija 5] Unity Asset Store, CYKO, Preuzeto: 23.08.2021, https://assetstore.unity.com/packages/2d/gui/icons/beautiful-progress-bar-free-194904

10. Popis slika

11. Popis korištenih tehnologija

Za razvoj računalne igre koristila se platforma *Unity.*

Poveznica: https://unity.com/

Za izradu programskog koda koristio se programski jezik *C#* u programskom okruženju *MS Visual Studio.*

Poveznica: https://visualstudio.microsoft.com/

Za izradu tekstualne grafike koristila se web aplikacija *Maketext.io.*

Poveznica: https://maketext.io/

Za izradu oblika staza za potrebe ovog dokumenta koristio se program *Aseprite.*

Poveznica: https://www.aseprite.org/

Za izradu zvučnih efekata koristio se program *Audacity.*

Poveznica: https://www.audacityteam.org/

Za izradu slika koristio se program *Lightshot.*

Poveznica: https://app.prntscr.com/en/download.html

Za izradu tekstualne dokumentacije ovog rada koristio se *MS Word.*

Poveznica: https://www.microsoft.com/en-ww/microsoft-365/word