Izrada igre akcijsko arkadne utrke u programskom alatu Unity

Horvat, Marko

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Horvat

IZRADA IGRE AKCIJSKO ARKADNE UTRKE U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Horvat

Matični broj: 45963/17–R

Studij: Informacijski sustavi

IZRADA IGRE AKCIJSKO ARKADNE UTRKE U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Mladen Konecki

Varaždin, rujan 2020.

Marko Horvat

Izjava o izvornosti

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

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

Sažetak

Tema ovog rada je izrada igre akcijsko arkadne utrke koristeći programski alat Unity i programski jezik C# te će to biti i razrađeno u ovom radu. Bit će prikazana implementacija kretanja igrača, sustav „checkpointeva“, način na koji funkcioniraju AI igrači te sami akcijski elementi koji igračima dodaju ili oduzimaju prednost na neki način. Biti će prikazani programski alat Unity te proces kreiranja igre u istom. Biti će opisan i žanr 2D „top-down“ trkaćih igara.

Ključne riječi: računalna igra, utrka, „Top-Down“, Unity, metode, C#, vozilo, AI

Sadržaj

1. Uvod

U današnje vrijeme, videoigre su jedan od najpopularnijih načina zabave i „ubijanja vremena“. Prve primjere videoigara možemo naći već u 60-im godinama prošlog stoljeća, a od onda, videoigre su jako napredovale, te možemo samo zamisliti kako će izgledati u budućnosti.

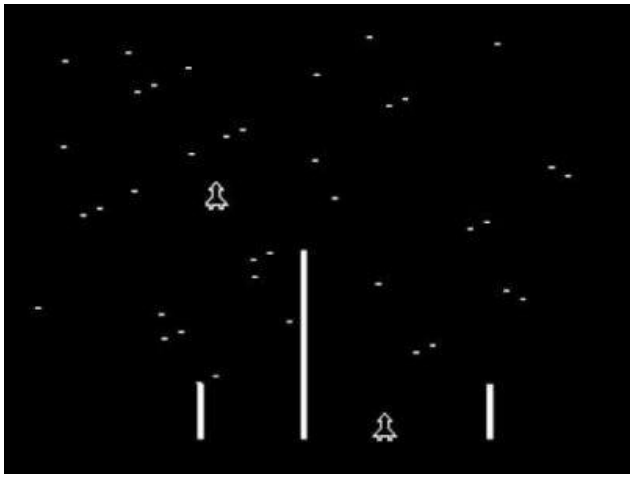
Tema ovo završnog rada je izrada igre akcijsko arkadne utrke u programskom alatu Unity i programskom jeziku C#. Kroz ovaj završni rad bit će prikazan program Unity te kako se pomoću njega može stvoriti videoigra. Osim toga, biti će objašnjen i način funkcioniranja moje videoigre, dizajn staze, kontrola vozila, AI igrači te akcijski elementi(„boost“, „landmine“, „oilspill“) koji igračima na neki način dodaju ili oduzimaju prednost nad drugima.

2. Trkaća igra

Trkaća igra je tip videoigre gdje igraći sudjeluju trkaćem natjecanju sa bilo kojim tipom vozila. Mogu se temeljiti na trkaćim ligama iz stvarnog svijeta ili na nekim znanstveno-fantastičnim temama. Trkaće igre mogu biti bilo gdje na spektru od simulacija do jednostavnih arkadnih igara. [1]

2.1. Povijest trkaćih igara

Prva trkaća igra u povijesti je nastala 1973. godine od strane tvrtke Atari pod nazivom **„Space Race“**. Igrači su pomoću kontrolera trebali upravljati svemirskim brodom i izbjegavati meteore prilikom utrkivanja sa drugim svemirskim brodom.[2]

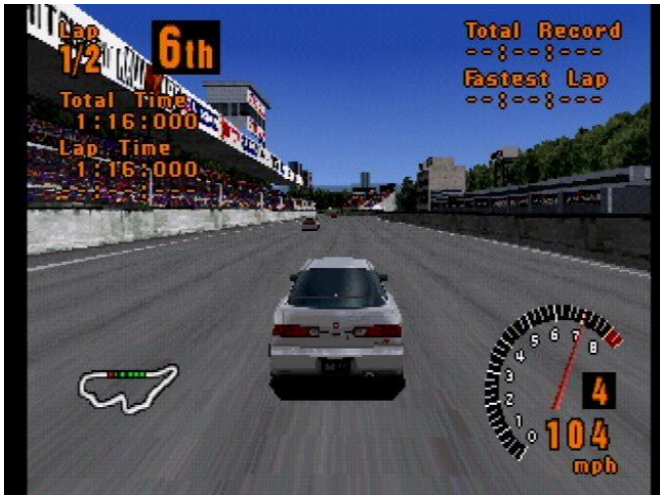


Slika 1: Space Race

Devedesete godine prošlog stoljeća bile su ključne za razvoj trkaćih videoigara. 1991. godine, nastala je „**1991 Formula One World Championship“** koja je nudila igračima na odabir vozače, vozila i staze iz stvarnog života.

Nintendo je imao veliki uspjeh sa **„Super Mario Kart“** igrom. Igra je imala fokus na zabavi i druženju te je imala veliki utjecaj na svijet videoigara.

Sredinom 90-ih, tehnologija je već dovoljno napredovala pa su počele izlaziti i 3D igre, što je u ono vrijeme igračima davalo dojam kao da se fizički nalaze u automobilu. 1997. godine je izašao prvi **„Gran Turismo“** što je tada bilo smatrano najboljom trkaćom igrom svih vremena. [2]



Slika 2: Gran Turismo

U 2000. godini, predstavljena je prva „free roam“ trkaća igra **„Midnight Club: Street Racing“** koje igračima omogućila slobodnu vožnju po mapi te odabir utrka po volji.[1]

U nadolazećim godinama, počele su osvanjivati igre koje sa velikim brojem nastavaka postoje još i danas poput **„Need for Speed-a“, „Forza-e“, „Dirt Rally-a“** i svake godine guraju granicu industrije videoigara u grafičkom smjeru i pružaju sve realniji doživljaj, fiziku i animacije.

Slika 3: Forza Horizon 4

3. Programski alat Unity

Unity je najpopularniji programski alat za razvoj videoigara kojeg je razvila tvrtka „Unity Technologies“. Predstavljen je 2005. godine na Apple-ovoj konferenciji te ga se je tada moglo koristiti samo na MAC OS-u, a danas podržava više od 25 platformi kao što su Windows, Linux, MAC OS, Android, iOS, PS4, Xbox One i mnogi drugi. Unity je veoma fleksibilan i ima jako puno značajki te se može koristiti za izradu 3D, 2D, VR i AR igara te simulacija i drugih iskustva. [3]

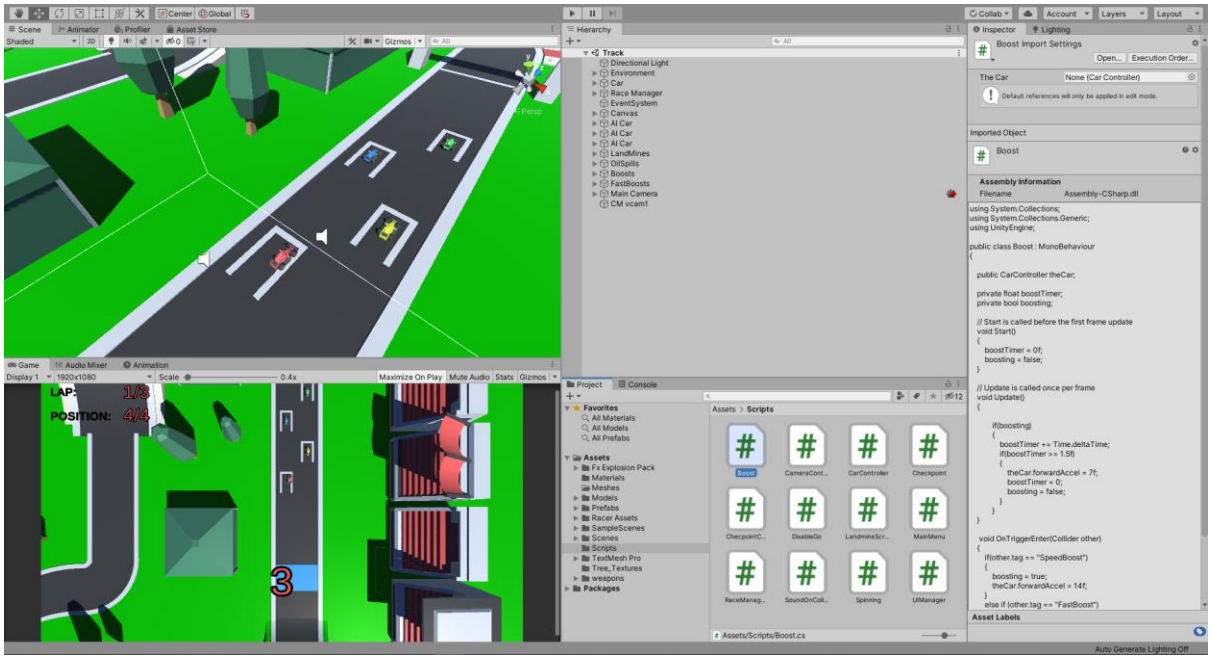
Slika 4: Unity logo

Za izradu igre i programskom alatu Unity, potrebno je znati programiranje u nekom programskom jeziku, a primarni jezik Unitya je C#. U prošlosti je postojala mogućnost korištenja Boo i JavaScript(„UnityScript“) platformi, no one su izbačene jer su gotovo svi korisnici koristili .NET(C#).

Unity je toliko popularan prvenstveno jer ga je vrlo lako koristiti te ima veoma „user-friendly“ sučelje. Rukovanje elementima je također veoma jednostavno zbog korištenja „Drag, Drop and Animate“ tehnike. Iako zahtjeva znanje programiranja, u Unityu se oko 80% igre može napraviti bez programiranja. Također, jedan od razloga njegove popularnosti je „Unity Asset Store“, koji pruža korisnicima dijeljenje modela, animacija, zvukova i sl. [4]

Neke od poznatijih igara koje su izrađene u Unityu su „Superhot“, „Firewatch“, „Angry birds“, „Ori and the Blind Forest“, „Besiege“, „Pokemon GO“.

3.1. Unity editor



Slika 5. Unity editor

Na slici je prikazan Unity editor sa svim potrebnim elementima za razvoj igre. U gornjem lijevom kutu se nalazi „Scene“ pogled, te nam je to glavni pogled za razvoj igre. U taj pogled se dodaju svi elementi te se upravlja njima. Ispod toga se nalazi „Game“ pogled. U tom pogledu se može vidjeti kako scena trenutno izgleda iz perspektive postavljene kamere. Prikaz „Game“ prikazuje sliku koju prvu vidimo kad pritisnemo „Play“ gumb(sredina na vrhu) koji zapravo pokreće igru. U sredini se nalaze „Hierarchy“ i „Project“. U „hierarchy“ prostoru vidimo sve objekte koji su korišteni u sceni te koji objekt je dijete kojeg objekta. „Project“ prostor nam služi za navigaciju kroz datotečni sustav projekta. Otuda možemo „drag and drop-ati“ objekte u scenu. Skroz desno je „Inspector“ prostor gdje su navedene sve komponente nekog objekta.

Unity nudi i druge poglede, te se oni mogu postavljati u iste prostore kao već navedeni te se prebacujemo između njih pomoću sustava kartica. Neki od korisnijih su: „Audio mixer“, „Animation“ i „Console“.

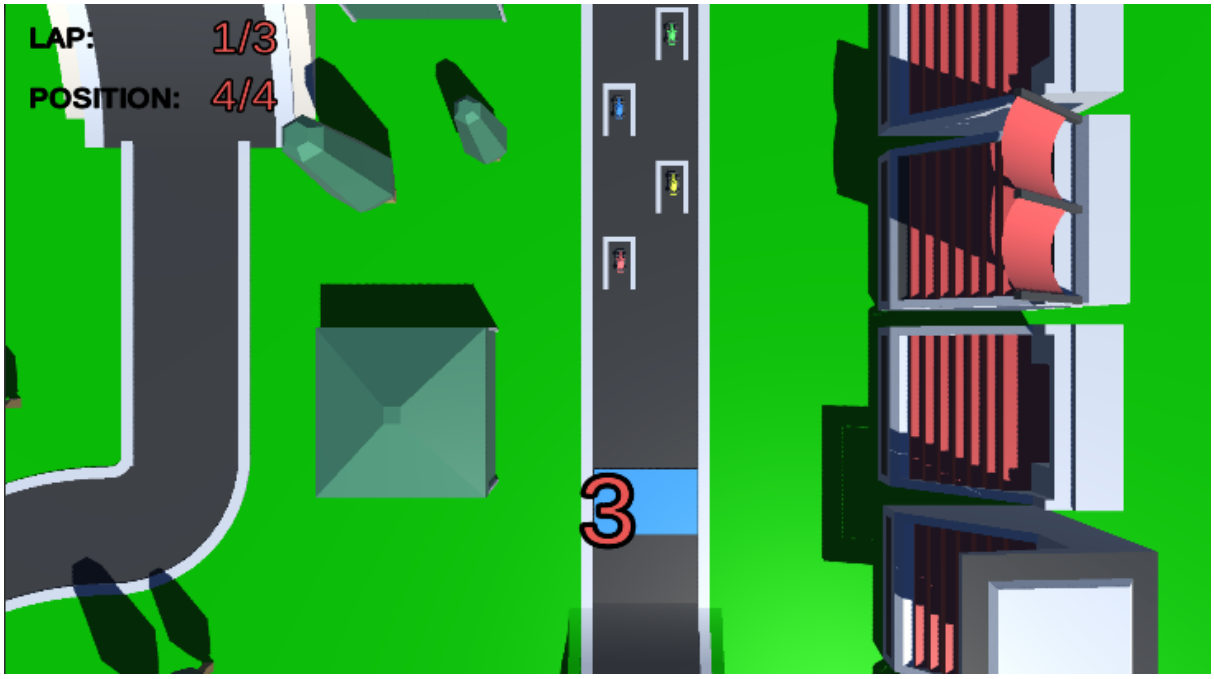
4. Izrada računalne igre

4.1. Opis igre

Moja igra je zamišljena kao 2D „top-down“ igra akcijsko arkadne utrke. U utrci sudjeluju četiri formule te se vozi 3 kruga. Po stazi su postavljene kontrolne točke(„checkpointi“) kroz koje igrač mora sve proći kako bi mu se priznao krug. Po stazi su razbacani različiti elementi, a igraču ju u namjeri te elemente pokupiti ili izbjeći, ovisno o elementu jer neki elementi dodaju prednost, a neki ju oduzimaju. Pobjednik je onaj igrač koji prvi završi tri kruga utrke.

Slika 6: Početni zaslone igre

Početni zaslon je veoma jednostavan. Na vrhu zaslona je nazive igre, a ispod toga su dva gumba, „Play“ i „Quit“. Pritiskom na gumb „Play“, pokreće se igra, a pritiskom na gumb „Quit“ izlazi se iz aplikacije.



Slika 7: Glavni ekran igre

Kada se igra pokrene, pokreće se utrka, a započinje odbrojavanjem od „3“. Kada dođe do nule, ispiše se „GO!“ te formule kreću sa utrkom.

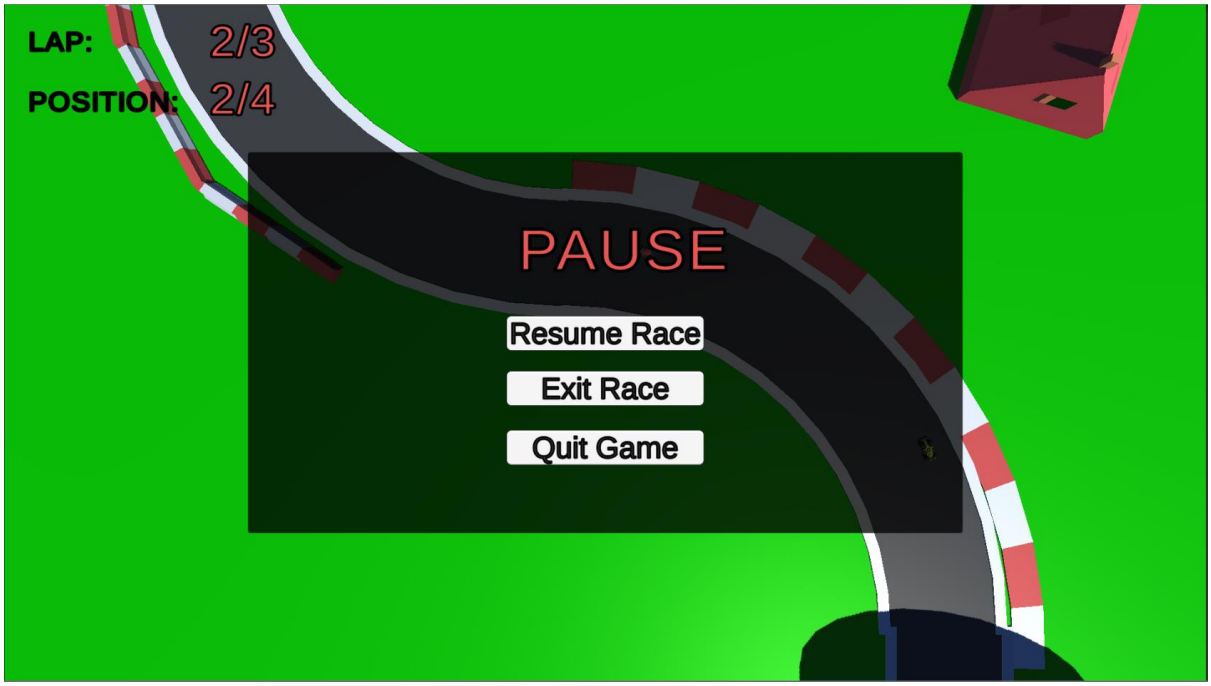
Korisničko sučelje je svedeno na minimum. U gornjem lijevom kutu imamo indikator tekućeg kruga te indikator trenutne pozicije. Na mapi se nalaze nepomični objekti poput drveća, gledališta i slično te elementi koji dodaju ili oduzimaju prednost: „Boost“, „Fast boost“, „Landmine“ i „Oil spil“.

„Boost“ je dio na stazi označen plavom bojom i proteže se od ruba do ruba staze. Kad igrač(ili AI igrač) svojim vozilom pređe preko toga područja, na sekundu i pola se povećava brzina vozila za duplo.

„Fast boost“ je kugla plave boje te daje trostruku brzinu vozilu na sekundu i pola.

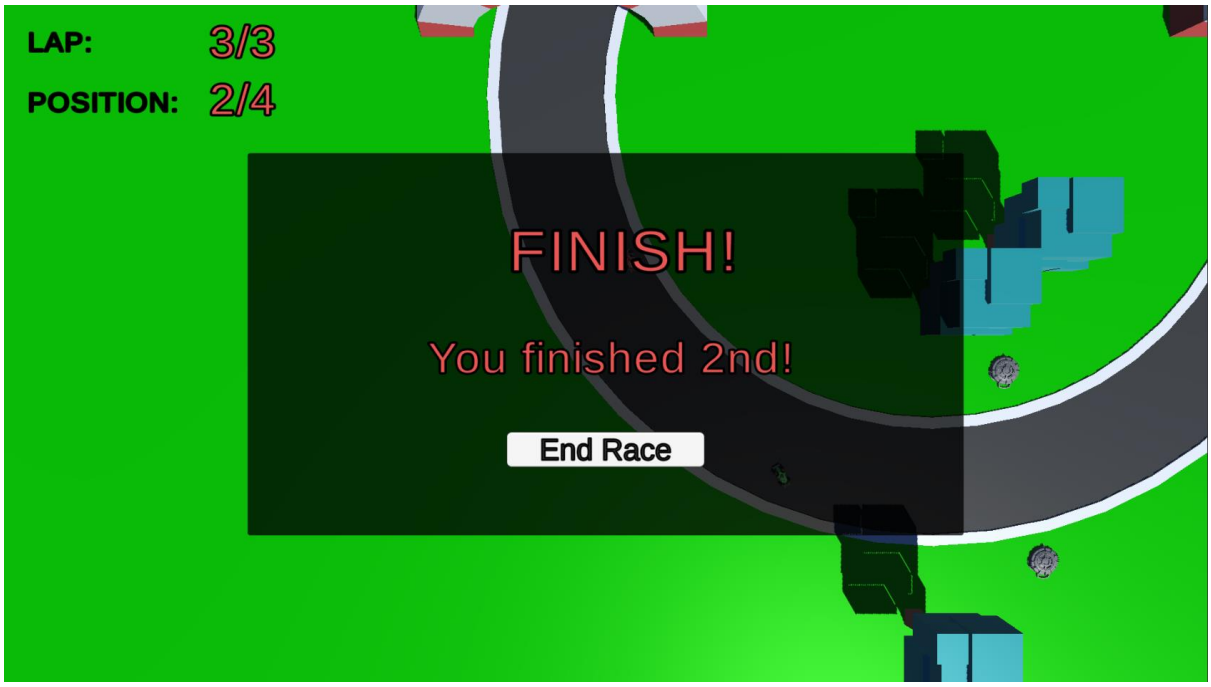
„Landmine“ je element koji kada ga nečije vozilo dotakne, eksplodira, te odbaci vozilo određenom snagom u smjeru sa kojeg ga je to vozilo udarilo. Nakon toga se ta mina uništi te se do kraja utrka druga vozila ne mogu udariti u nju.

„Oil spil“ je element koji kad vozilo dođe do njega, zaokrene vozilo par put na mjestu, a nakon toga, vozilo se može ponovo kretati normalno kao i prije.



Slika 8: „Pause menu“

Pritiskom na tipku „Escape“ igra se pauzira te je prikazan „Pause menu“. Sastoji se od riječi „Pause“ na vrhu i tri gumba. Pritiskom na gumb „Resume Race“ prekida se pauze te se utrka nastavlja. Pritiskom na gumb „Exit Race“ utrka se prekida te se izlazi na glavni izbornik. Pritiskom na gumb „Quit Game“ prekida se utrka i automatski se zatvara aplikacija.



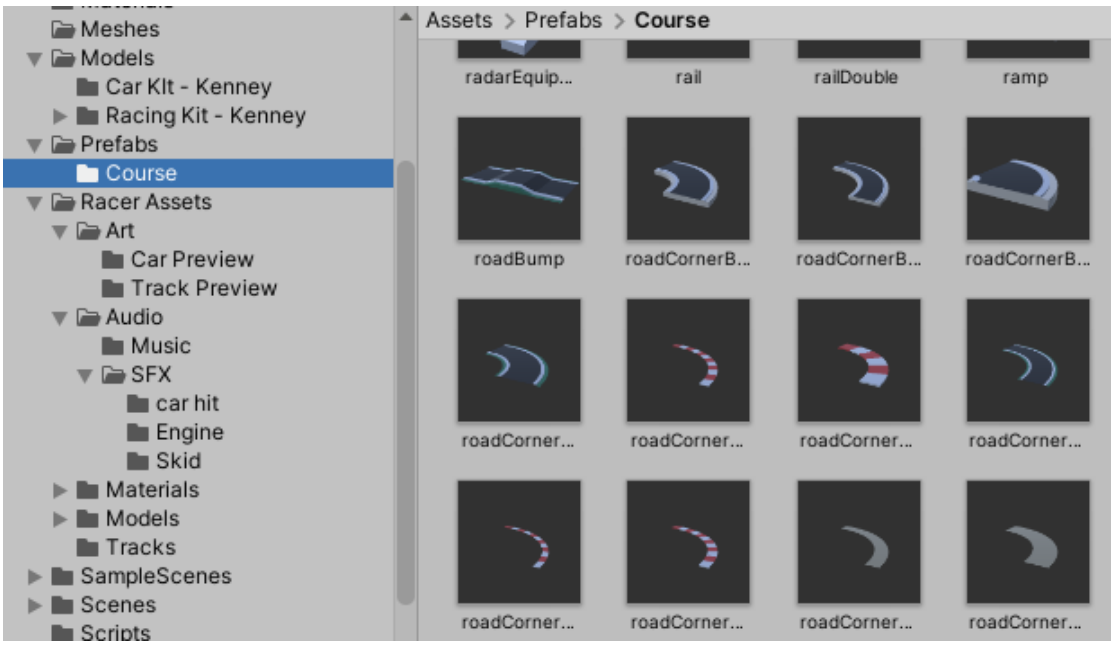
Slika 9: Ekran za kraj utrke

Kad utrka završi, igrač gubi kontrolu nad vozilom te ju preuzima umjetna inteligencija te se prikazuje rezultat utrke, odnosno ispisuje se pozicija na kojoj je igrač završio. Ispod toga se nalazi gumb „End Race“ koji vraća igrača na početni zaslon.

4.2. Staza(engl. Track)

Staza je kreirana koristeći preuzeti paket sa „Unity Asset Store-a“. U tom paketu nalazili su se razni dijelovi staze: ravni dijelovi, zavoji, usponi, most, drveća, ostali ukrasi i sl.

Stvoriti stazu je bilo veoma lako, koristeći „drag and drop“ metodu te rotiranjem i skaliranjem dijelova. Na kraju su dodani i nepomični elementi poput drveća i barijera kako mapa ne bi izgledala prazno.



Slika 10: Korišteni dijelovi staze

Vrijedi napomenuti da su se u tom istom paketu nalazili i modeli vozila kojih je isto bilo na izbor te set zvukova koji su također korišteni u igri.

4.3. Kretanje vozila

```
public Rigidbody theRB;

public float forwardAccel = 8f, reverseAccel = 4f, maxSpeed = 35f, turnStrength = 180, gravityForce = 10f, dragOnGround = 3f;

private float speedInput, turnInput;

private bool grounded;

public LayerMask whatIsGround;
public float groundRayLength = .5f;
public Transform groundRayPoint;

public float gravityMod = 10f;

public Transform leftFrontWheel, rightFrontWheel;
public float maxWheelTurn = 25f;

public AudioSource engineSound, skidSound;
public float skidFadeSpeed;

public int nextCheckpoint;
public int currentLap;

public bool isAI;

public int currentTarget;
private Vector3 targetPoint;
public float aiAccSpeed = 1f, aiTurnSpeed = .8f, aiReachPointRange = 5f, aiPointVariance = 3f, aiMaxTurn = 15f;
private float aiSpeedInput, aiSpeedMod;


void Start()
{
    theRB.transform.parent = null;

    if(isAI)
    {
        targetPoint = RaceManager.instance.allCheckpoints[currentTarget].transform.position;
        RandomiseAiTarget();

        aiSpeedMod = Random.Range(.8f, 1.1f);
    }
    UIManager.instance.lapCounter.text = currentLap + "/" + RaceManager.instance.numberOfLaps;
}
```

Slika 11: CarController skripta – početni dio

Na početku skripte definirani su atributi i varijable potrebni za kretanje vozila, a nekima od njih su dodane i vrijednosti koje se mogu kasnije promijeniti unutar Unity „inspector“ pogleda.

Unutar Start() metode, koja se izvršava prilikom inicijalizacije skripte, postavljene su postavke koje vrijede prilikom pokretanja igre. Postavljeno je slijedeće odredište za AI vozila, odnosno prema kojoj točci će se kretati te modifikator brzine AI vozila. Naime, ne želimo da nam sve vozila imaju identičnu brzinu, pa stoga postavljamo „aiSpeedMod“ na nasumičnu vrijednost od 0.8 do 1.1, što se kasnije množi sa brzinom kretanja. Na taj način, vozila će se uvijek kretati različitom brzinom što čini igru zanimljivijom. Također je postavljen brojač krugova.

```
if(!isAI)
{
 speedInput = 0f;
if(Input.GetAxis("Vertical") > 0)
{
    speedInput = Input.GetAxis("Vertical") * forwardAccel * 1f;
} else if(Input.GetAxis("Vertical") < 0)
{
    speedInput = Input.GetAxis("Vertical") * reverseAccel * 1f;
}

turnInput = Input.GetAxis("Horizontal");


if(grounded)
{
    transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles +
    new Vector3(0f, turnInput * turnStrength * Time.deltaTime * Input.GetAxis("Vertical"), 0f));
}
} else
{
    targetPoint.y = transform.position.y;

    if(Vector3.Distance(transform.position, targetPoint) < aiReachPointRange)
    {
        SetNextAITarget();

    }

    Vector3 targetDirection = targetPoint - transform.position;
    float angle = Vector3.Angle(targetDirection, transform.forward);

    Vector3 localPosition = transform.InverseTransformPoint(targetPoint);
    if(localPosition.x < 0f)
    {
        angle = -angle;
    }

    turnInput = Mathf.Clamp(angle / aiMaxTurn, -1f, 1f);

    if(Mathf.Abs(angle) < aiMaxTurn)
    {
        aiSpeedInput = Mathf.MoveTowards(aiSpeedInput, 1f, aiAccSpeed);
    } else
    {
        aiSpeedInput = Mathf.MoveTowards(aiSpeedInput, aiTurnSpeed, aiAccSpeed);
    }

    speedInput = aiSpeedInput * forwardAccel * aiSpeedMod;
}
```

Slika 12: CarController skripta- upravljanje vozilom

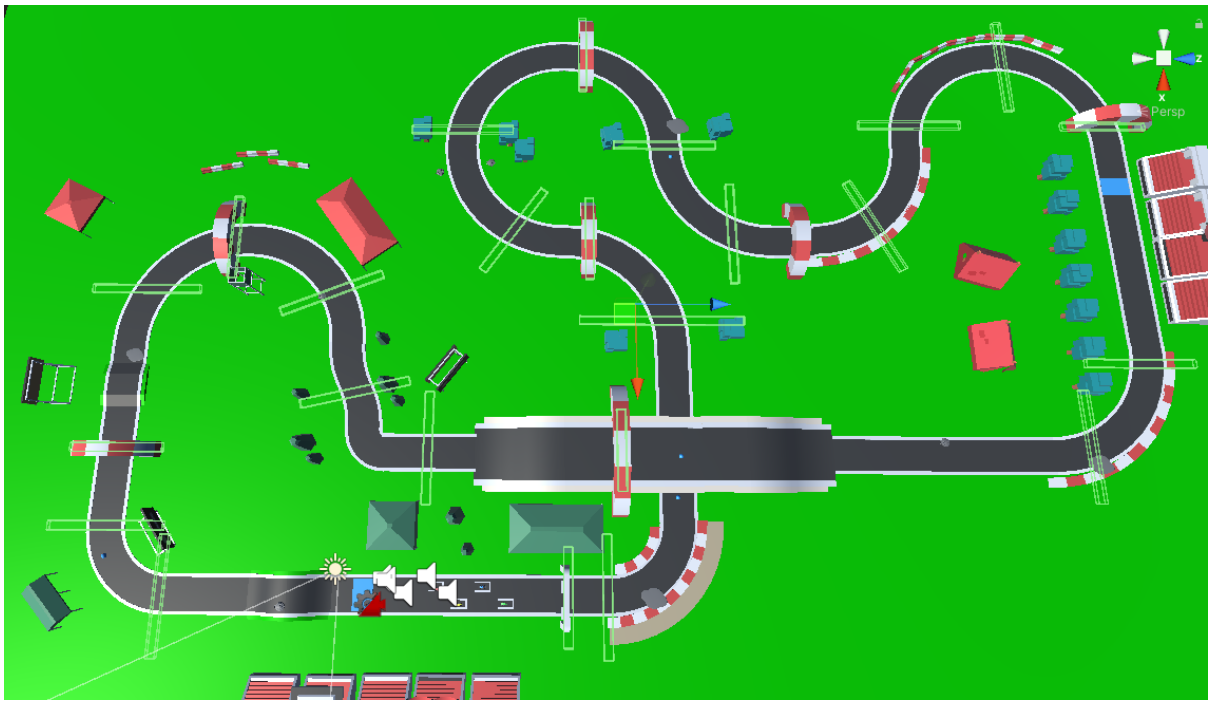
U metodi Update(), koja se poziva jednom po okviru, nalazi se dio koda koji nam omogućuje upravljanjem vozila.

Na početku se provjerava da li je vozilo AI, odnosno, ako nije, to znači da je to vozilo kojim upravlja igrač i omogućen je unos kontrola. Zatim se provjerava da li želimo ići naprijed ili natrag, odnosno da li je „Vertical“ unos veći od nule, što znači da igrač drži strelicu gore ili slovo W na tipkovnici i želi ići naprijed, ili je „Vertical unos manji od nule, što znači da igrač

drži strelicu dolje ili slovo S i želi ići natrag. Kada se otkrije želi li igrač naprijed ili natrag, množimo njegov unos sa „forwardAccel“ (ubrzanje unaprijed) ili „reverseAccel“ (ubrzanje unatrag) te tako postižemo kretanje unaprijed i unatrag.

Zatim se provjerava da li je vozilo na tlu, te ako je, rotiramo vozilo oko Y osi te tako dobivamo radnju skretanja.

Sada dolazimo do dijela, što ako vozilo uistinu je AI vozilo. Princip kretanja AI vozila je zapravo veoma jednostavan. Naime, po stazi su postavljene kontrolne točke, a da bi određenom vozilu, igraču ili AI vozilu, vrijedio odvoženi krug, vozilo kroz jedan krug staze mora proći kroz baš svaku kontrolnu točku. Stoga se AI vozila cijelo vrijeme usmjeravaju prema slijedećoj kontrolnoj točci.



Slika 13: Sustav kontrolnih točki

Kontrolne točke su označene rednim brojevima, a kao početna kontrolna točka prema kojoj će se AI vozila kretati bit će „Checkpoint 0“. Kad AI vozilo dostigne „Checkpoint 0“, pozove se metoda SetNextAITarget() koji namješta atribut *currentTarget* na slijedeću kontrolnu točku.

```
public void SetNextAITarget()
{
    currentTarget++;
    if(currentTarget >= RaceManager.instance.allCheckpoints.Length)
    {
        currentTarget = 0;
    }

    targetPoint = RaceManager.instance.allCheckpoints[currentTarget].transform.position;
    RandomiseAiTarget();
}
```

Slika 14: CarController skripta- postavljanje slijedeće kontrolne točke AI vozilima

Kada AI vozilo poprimi novu vrijednost za atribut *currentTarget*, vozilo se orijentira prema toj kontrolnoj točci. I tako AI vozila cijelo vrijeme „love“ kontrolne točke, i kada stignu do kontrolne točke *n*, nastavljaju voziti prema kontrolnoj točci *n+1* i tako cijelo vrijeme u krug.

Kako se sva AI vozila nebi kretala identično u liniji jedna iza druge, za to nam služe atribut tipa float *aiSpeedMod* i metoda *RandomiseAiTarget()*.

```
public void RandomiseAiTarget()
{
    targetPoint += new Vector3(Random.Range(-aiPointVariance, aiPointVariance),
    0, Random.Range(-aiPointVariance, aiPointVariance));
}
```

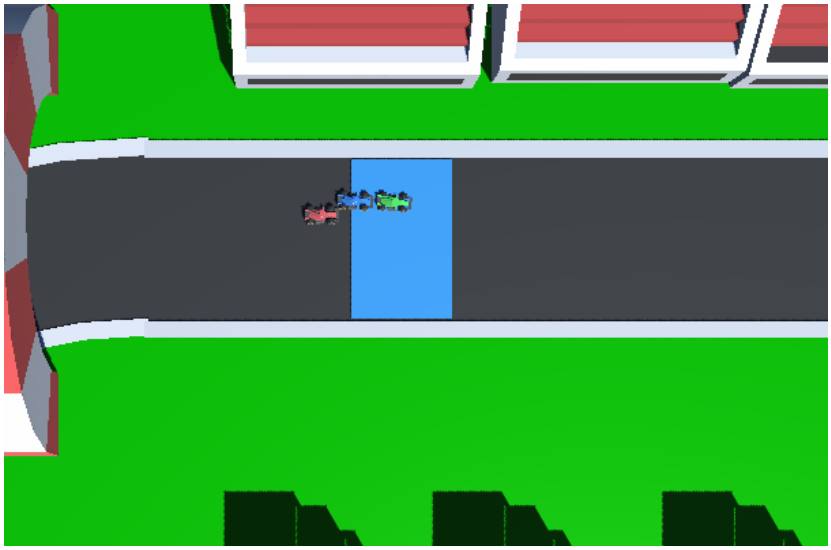
Slika 15: CarController skripta- dio koda za nasumično kretanje AI vozila

Atribut *aiSpeedMod* služi za nasumičnu brzinu kretanja AI vozila, a metoda *RandomiseAITarget*() služi za nasumično kretanje. Naime, metoda *RandomiseAITarget()* pomiče točku prema kojoj se vozilo kreće za neku vrijednost na X i Z osima od centra kontrolne točke. Za koliku vrijednost će se točka prema kojoj se vozilo kreće pomaknuti ovisi o vrijednosti atributa *aiPointVariance* koja je definirana na početku i postavljena na vrijednost 3, što znači da je raspon pomaka točke prema kojoj se vozilo kreće u rasponu od -3 do 3 na X i Z osi.

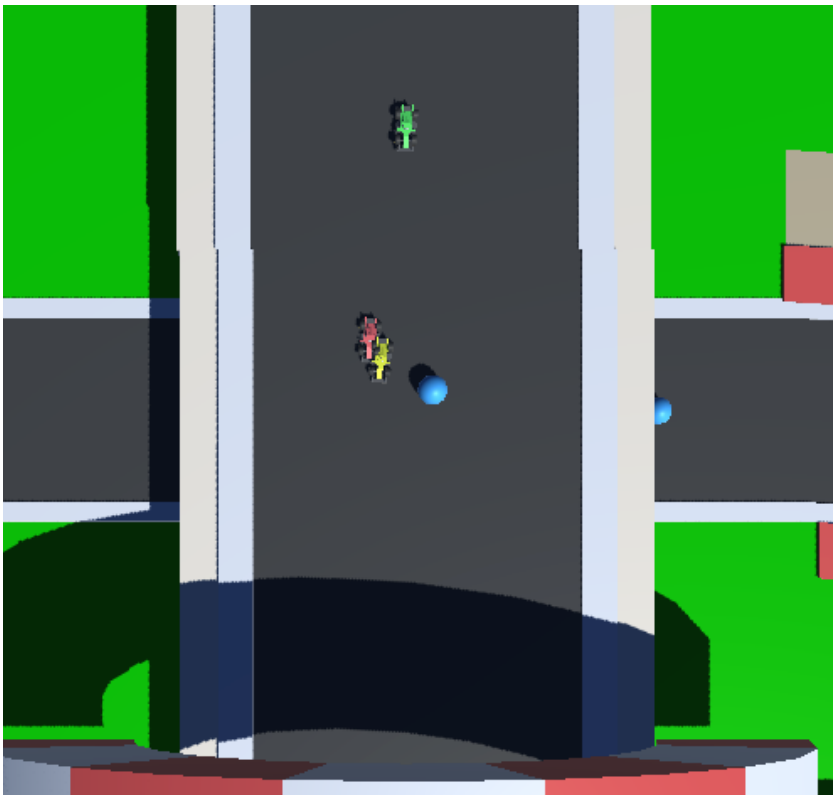
4.4. Pojačanja i zamke

4.4.1. Ubrzanje

U igri postoje dvije vrste ubrzanja. Traka za ubrzanje i kugla za ubrzanje.



Slika 16: Traka za ubrzanje



Slika 17: Kugla za ubrzanje

Traka za ubrzanje koja se proteže dužinom staze i veoma je lako dobiti to ubrzanje. Dobivanjem tog ubrzanja, brzina vozila se udvostručuje i traje sekundu i pola. Kuglu za ubrzanje je relativno teže dobiti jer je manjih dimenzija, ali zato vozilo koje prođe kroz tu kuglu je nagrađeno trostrukim ubrzanjem na sekundu i pola.

```
void Start()
{
    boostTimer = 0f;
    boosting = false;
}

void Update()
{

        if(boosting)
        {
            boostTimer += Time.deltaTime;
            if(boostTimer >= 1.5f)
            {
                theCar.forwardAccel = 7f;
                boostTimer = 0;
                boosting = false;
            }
        }
}

 void OnTriggerEnter(Collider other)
{
    if(other.tag == "SpeedBoost")
    {
        boosting = true;
        theCar.forwardAccel = 14f;
    }
    else if (other.tag == "FastBoost")
    {
        boosting = true;
        theCar.forwardAccel = 21f;
    }
}
```

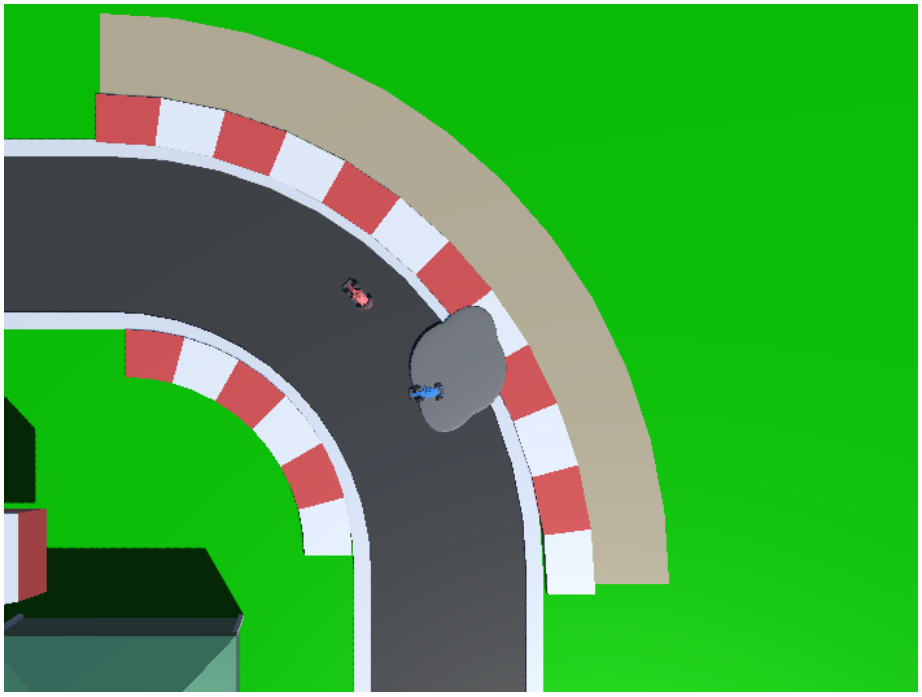
Slika 18: Boost skripta – ubrzanje vozila

Na početku *Boost* skripte atribut *boostTimer* se postavlja na vrijednost nula, a atribut *boosting* se postavlja na vrijednost *false.* U metodi *OnTriggerEnter* se mijenja brzina vozila. Ovisno s kojim se elementom vozilo dotakne, *boosting* se postavlja na vrijednost *true* te se brzina udvostručuje ili utrostručuje.

U *update()* metodi se konstantno provjerava da li je aktivno ubrzanje. Ako je, brojač se u svakoj sličici povećava za vrijednost *deltaTime*, i kad dostigne sekundi i pola, brzina se vrati na početnu vrijednost, brojač se resetira, a *boosting* se vraća na vrijednost *false*.

4.4.2.Mrlja ulja

Mrlja ulja je element u igri koji igraču, ili AI vozilima, oduzima prednost na način da im oteža kretanje nasumičnim izvrtavanjem na stazi. U igri su 4 mrlje ulja nasumično postavljene po stazi, i kada god vozilo dođe u doticaj s bilo kojom od njih, vozilo će izgubiti kontrolu i zavrtjeti se na mjestu.



Slika 19: Mrlja ulja na stazi

```
void OnTriggerEnter(Collider other)
{
    if(other.tag == "Oil")
    {
        theCar.forwardAccel = 0f;
        startSpinning();
    }
}
```

Slika 20: CarSpinning skripta – doticaj s uljem

U update() metodi, koja se poziva svaki okvir igre, se konstanto provjerava da li je vozilo došlo u doticaj sa mrljom ulja, odnosno da li se tijelo vozila „sudarilo“ sa elementom kojem je oznaka „Oil“, što označava mrlju ulja. Kada igra detektira „sudar“ tijela vozila sa mrljom ulja, skripta zaustavlja vozilo na licu mjesta postavljajući atribut *theCar.forwardAccel* na nulu, te se poziva metoda *startSpinning()* koja zavrti vozilo.

```
void startSpinning()
{
    if(isSpinning == true)
    {
        return;
    }
    StartCoroutine(SpinningRoutine());
}

IEnumerator SpinningRoutine()
{
    isSpinning = true;


    while (spinningTime < spinningDuration)
    {
        float spinningProgress = spinningTime/spinningDuration;
        spinningTime += Time.deltaTime;

        VisualsParent.transform.localRotation = Quaternion.Euler(0f,spinningProgress * spinningRotations * 360f, 0f);

        yield return null;
    }
    VisualsParent.transform.localRotation = Quaternion.identity;
    isSpinning = false;
    spinningTime = 0f;

}
```

Slika 21: CarSpinning skripta – Metode *startSpinning()* i *Spinning Routine()*

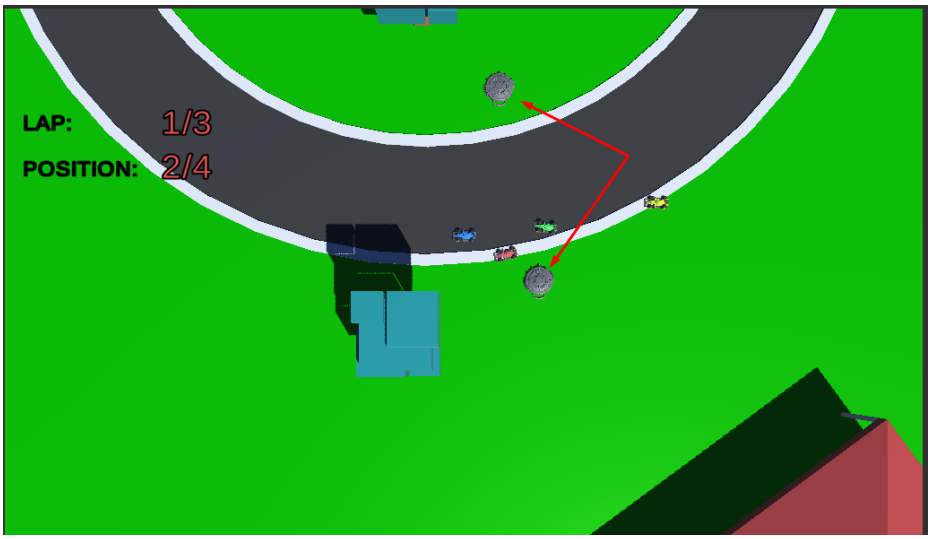
Metoda *startSpinning()* na samom početku provjerava da li se vozilo već vrti. Ako se već vrti, metoda *startSpinning()* završava, a ako se vozilo ne vrti, tada se pokreće korutina *SpinningRoutine()*.

Na samom početku te korutine, atribut *isSpinning* se postavlja na vrijednost „*true*“, a zatim se pokreće petlja koja zapravo vrti vozilo. Petlja se izvodi sve dok je trenutno vrijeme vrtnje manje od postavljene dužine vrtnje. Postavljena dužina vrtnje je 0,5 sekundi, odnosno 3 vrtnje vozila oko svoje osi .Dok traje *while* petlja, vrijeme vrtnje se povećava za vrijednost *deltaTime*, što je zapravo dužina trajanja jedne sličice(*framea*), i vrti se vozilo. Jednom kad se petlja završi, atribut *isSpinning* se postavlja na vrijednost „*false*“ čime se vrtnja zaustavlja, a vrijeme vrtnje se resetira i postavlja na nulu.

Za vrijeme trajanja *Update()* metode, provjerava se da li se izvodi *isSpinning()* metoda, ako se izvodi, pokreće se brojač *spinTimer* koji se svaki ekran igre povećava za vrijednost *deltaTime*.

Kada je brojač *spinTimer* jednak ili veći od 0,5 sekundi, vrtnja se zaustavlja postavljajući atribut *isSpinning* na vrijednost *false*, *spinTimer* se ponovno postavlja na nulu, a vozilo se opet može kretati postavljanjem atributa *startSpinning()* na početnu postavljenu vrijednost.

4.4.3.Mina



Slika 22: Mine

Mina je objekt u igri koja na doticaj sa vozilom eksplodira te svojom silom odbaci vozilo. Vozilo se u tom trenutku ne može željeno kretati, već mora čekati da ponovo padne na zemlju. U igri je 5 mina, ali nakon aktivacije mine, ta mina se uništi.

```
public float radius = 10f;
public float explosionForce = 10f;

void Start()
{

}

void Update()
{

}

private void OnCollisionEnter(Collision other)
{
    Explode();
}

private void Explode()
{
    Collider[] colliders = Physics.OverlapSphere(transform.position, radius);

    foreach (Collider col in colliders)
    {
        Rigidbody rig = col.GetComponent<Rigidbody>();

        if(rig != null)
        {
            rig.AddExplosionForce(explosionForce, transform.position, radius, 2f, ForceMode.Impulse);
        }
    }
    Instantiate(explosionEffect, transform.position, transform.rotation);
    Destroy(gameObject);
}
```

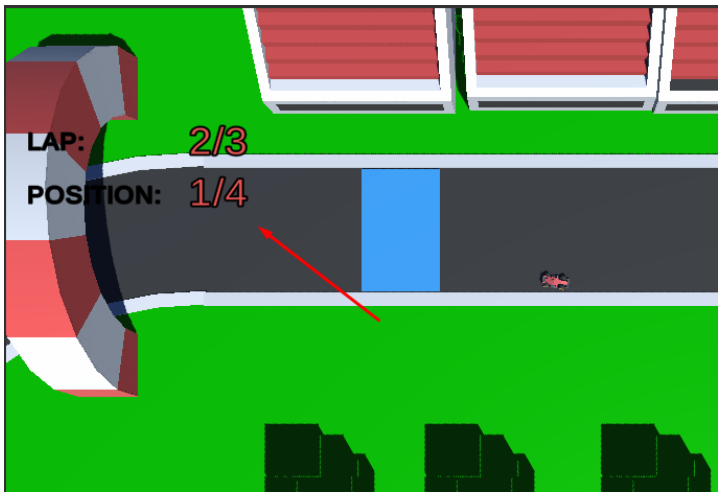
Slika 23: Landmine skripta

Kada ova skripta detektira da je neko vozilo dotaklo minu, mina se aktivira. Aktivirana mina utječe na sva vozila koja su u radijusu eksplozije, a ne samo na vozilo koje ju je aktiviralo. Skripta dohvaća *Rigidbody* komponentu svih vozila u radijusu, i utječe *impulse* silom na sva vozila čije je *Rigidbody* komponente dohvatilo.

Instancira se vizualni efekt eksplozije te se na samom kraju aktivirana mina uništava te se više ta specifična mina ne pojavljuje u igri i ne može se više aktivirati.

4.5. Krugovi i pozicija

Kroz igru se prati i bilježi trenutna pozicija i krug igrača, te se to prikazuje u gornjem lijevom kutu igre.



Slika 24: Pozicija igrača i trenutni krug

4.5.1. Trenutna pozicija igrača

Pozicija igrača se početno postavlja na „1“, a u svakom ekranu igre se izvršava nekoliko provjera kako bi se utvrdila točna pozicija igrača.

```
playerPosition = 1;

foreach(CarController aiCar in allAICars)
{
    if(aiCar.currentLap > Car.currentLap)
    {
        playerPosition++;
    } else if(aiCar.currentLap == Car.currentLap)
    {
        if(aiCar.nextCheckpoint > Car.nextCheckpoint)
        {
            playerPosition++;
        } else if(aiCar.nextCheckpoint == Car.nextCheckpoint)
        {
            if(Vector3.Distance(aiCar.transform.position, allCheckpoints[aiCar.nextCheckpoint].transform.position)
             < Vector3.Distance(Car.transform.position, allCheckpoints[Car.nextCheckpoint].transform.position))
            {
                playerPosition++;
            }
        }
    }
}
```

Slika 24: RaceManager skripta - Praćenje trenutne pozicije igrača

Za svako AI vozilo se provjerava da li je krug u kojem se trenutno nalaze veći od trenutnog kruga igrača. Za svako vozilo koje je krug ispred igrača, *playerPosition* se uvećava za 1. Zatim se za svako AI vozilo provjerava da li je njihova slijedeća kontrolna točka veće od slijedeće kontrolne točke igrača, što znači da su AI vozilo i igrač u istom krugu, ali je AI vozilo naprijed. Za svako takvo AI vozilo, *playerPosition* se uvećava za 1.

Na kraju, ako neko AI vozilo i igrač imaju istu slijedeću kontrolnu točku, provjerava se tko je bliže slijedećoj kontrolnoj točci. Za svako AI vozila čija je udaljenost do slijedeće kontrolne točke manja od udaljenosti vozila igrača do slijedeće kontrolne točke, *playerPosition* se uvećava za 1.

4.3.2. Trenutni krug igrača

Trenutni krug igrača se bilježi prateći kontrolne točke kroz koje je igrač prošao. Ponovimo, vozilo ima postavljeni *nextCheckpoint* te kada ga dosegne, *nextCheckpoint* se postavlja na slijedeću kontrolnu točku u nizu.

Jednom kada vozilo dosegne posljednju točku u nizu, *nextCheckpoint* se postavlja na 0(prva kontrolna točka u nizu), te se poziva metoda *LapCompleted()*.

```
public void LapCompleted()
{
    currentLap++;

    if(currentLap <= RaceManager.instance.numberOfLaps)
    {
        if(!isAI)
        {
            UIManager.instance.lapCounter.text = currentLap + "/" + RaceManager.instance.numberOfLaps;
        }
    } else
    {
        if(!isAI)
        {
            isAI = true;
            aiSpeedMod = 1f;

            targetPoint = RaceManager.instance.allCheckpoints[currentTarget].transform.position;
            RandomiseAiTarget();

            RaceManager.instance.finishRace();
        }
    }
```

Slika 25: CarController skripta – metoda *LapCompleted()*

Pri samom pokretanju *LapCompleted()* metode, atribut koji prati trenutni krug igrača *currentLap* se uvećava za 1. Zatim se provjerava da li utrka još traje, odnosno da li je trenutni krug igrača manji ili jednak od ukupnog broja krugova utrke, što je ukupno tri kruga. Ako je taj uvjet zadovoljen, sve što se dogodi je da se promijeni oznaka trenutnog kruga u gornjem lijevom kutu igre.

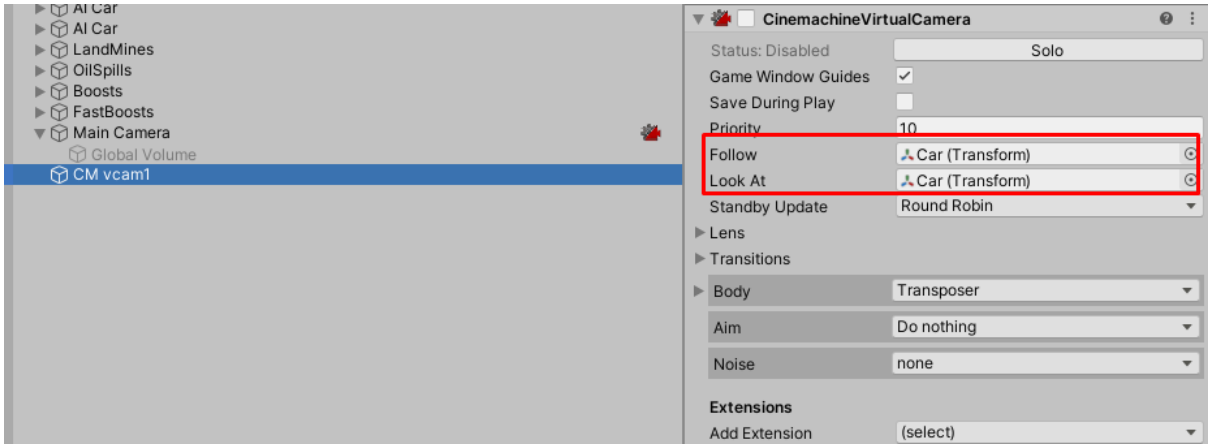
Ako uvjet nije zadovoljen, što znači da je trenutni krug igraća veći od ukupnog broja krugova utrke i da je utrka gotova, oduzima se kontrola igraču na način da njegovo vozilo također postaje AI i počne se ponašati kao i sva ostala AI vozila u igri.

4.6. Kamera

Kamera je objekt koji je dostupan u projektu čim se projekt kreira, bez potrebnog ručnog ubacivanja istog. Kamera omogućuje postavljanje vidnog polja i kuta gledanja u igri.

U mojoj igri kamera na igrača gleda iz ptičje perspektive i konstantno ga prati prilikom kretanja po stazi kretajući se po X i Z osi, ali bez rotiranja.

Kako bi se postigao efekt praćenja igrača, koristi se Unity paket „Cinemachine“. Koristeći taj paket, kreira se virtualna kamera koja se postavlja kao objekt dijete (*child object*) glavne kamere. Toj virtualnoj kameri, može se odabrati neki objekt iz igre kojeg ta kamera gleda i prati. Za obje stvari, odabire se vozilo igrača, objekt iz igre naziva „Car“.



Slika 26: Postavke kamere za praćenje vozila igrača

Problem sa trenutnim postavkama kamera je da će se kamera rotirati zajedno sa vozilom, a to je neželjeno ponašanje unutar igre. Kako bi se riješila problematika takvog ponašanja, koristi se skripta *CameraController.*

```
void Start()
{
    offsetDir = transform.position - target.transform.position;


}

void Update()
{
    transform.position = target.transform.position + offsetDir;
}
```

Slika 27: CameraController skripta

Ova skripta se dodaje na objekt kamere, i što ova skripta u suštini radi je to da u *start()* metodi postavlja fiksnu udaljenost od vozila, a u *update()* održava tu fiksiranu udaljenost i tako prati auto po X i Z osi, ali bez rotacije kamere.

4.7. Zvuk

4.7.1. Zvuk motora

Za zvuk motora se koristi zvuk reli automobila u neutralnoj brzini. U programu Unity, odabere se opcija da se taj zvuk konstantno vrti u krug(loop opcija), te se ovisno o brzini kretanja vozila mijenja visina zvuka(pitch) i tako se dobiva zvučni efekt ubrzavanja ili usporavanja vozila.

```
if(engineSound != null)
{
    engineSound.pitch = 1f + ((theRB.velocity.magnitude / maxSpeed) * 2f);
}
```

Slika 28: CarController skripta – *engineSound*

Opisani način rada zvuka motora je funkcionalan zahvaljujući ovom malom dijelu koda u *update()* metodi, koji provjerava, ako postoji zvuk motora, onda se visina zvuka povećava od početne vrijednosti ovisno o brzini vozila.

4.7.2. Cviljenje guma

Za cviljenje guma se koristi zvučni efekt *tire squeel* , kojem se isto uključi opcija konstantnog ponavljanja, ali je prilikom pokretanja igre isključen, a ponašanjem tog zvučnog efekta se upravlja sljedećim dijelom koda:

```
if(skidSound != null)
{
    if(Mathf.Abs(turnInput) > .5f)
    {
        skidSound.volume = 1f;
    } else
    {
        skidSound.volume = Mathf.MoveTowards(skidSound.volume, 0f, skidFadeSpeed * Time.deltaTime);
    }
}
```

Slika 29: CarController skripta – *skidSound*

Ovaj dio koda provjerava da li je unos skretanja (*turnInput*) veći od pola, odnosno 0,5, i ako je, uključuje se zvuk cviljenja guma. Ako unos skretanja nije veći od pola, onda se zvuk cviljenja guma sa vremenom mijenja od trenutne vrijednosti u nulu, odnosno isključen zvuk cviljenja guma. Time se dobiva efekt postepenog smanjivanja umjesto da se zvuk odjednom samo isključi.

5. Zaključak

Ovaj završni rad se sastoji od tri glavna poglavlja: žanr trkaćih igara, objašnjenje alata Unity te opis tehničke razrade ovog završnog rada.

Alat Unity je besplatan alat koji nudi velike mogućnosti razvoja u privatne amaterske svrhe, ali je veoma sposoban i za profesionalan razvoj igara. Mogu reći da je veoma intuitivan i jednostavan za korištenje.

Moram priznati da je početak izrade računalne igre bio zahtjevniji nego sam očekivao, ali uz trud te pomoć Unity zajednice i foruma te video tutoriala, u kratkom vremenu sam polovio osnove izrade računalne igre pomoću Unity-a i programskog jezika C#, te sam uspješno i potpuno samostalno dovršio igru.

Moram reći da sam zadovoljan konačnim ishodom ovog završnog rada, stvarima koje sam naučio te da je cijeli proces bio veoma nagrađujuć.

6. Literatura

[1] Racing video game. Preuzeto 09.09.2020. s https://en.wikipedia.org/wiki/Racing_video_game

[2] The History And Evolution Of Video Car Racing Games. Preuzeto 09.09.2020 s https://www.polebetting.com/news/history-evolution-video-car-racing-games

[3] Unity (game engine). Preuzeto 09.09.2020. s https://en.wikipedia.org/wiki/Unity_(game_engine)

[4] Reasons Why Unity3D Is So Much Popular In The Gaming Industry. Preuzeto 09.09.2020. s https://medium.com/@vivekshah.P/reasons-why-unity3d-is-so-much-popular-in-the-gaming-industry-705898a2a04

7. Popis slika

8. Izvori slika

Slika 1: https://i.ytimg.com/vi/0eBUoY6W8BY/hqdefault.jpg

Slika 2: https://s.uvlist.net/l/y2006/05/19558.jpg

Slika 3: https://www.noob.ba/wp-content/uploads/2018/09/forza-horizon-4_autumn-drive.jpg

Slika 4: https://unity.com/logo-unity-web.png