Interaktivna simulacija leta zrakoplova Dassault Rafale M

Seminarski rad iz kolegija "Interaktivni simulacijski sustavi"

Alberto Kerim

12. siječnja 2024.

Djelovođa: izv. prof. dr. sc. Siniša Popović

***Sažetak* – Seminarski rad istražuje implementaciju interaktivne simulacije leta Dassault Rafale M s pomoću Unity platforme. Fokus je na tehničkim aspektima modeliranja leta, realističnosti vizualnih efekata i zvukova te opće implementacije funkcija leta borbenog zrakoplova. Kroz analizu implementiranih elemenata, rad pruža uvid u kompleksnost stvaranja simulacije leta vojnih zrakoplova u virtualnom okruženju. S naglaskom na Unity alatu, istražuju se izazovi i rješenja prilikom integracije realističnih letnih karakteristika u interaktivni doživljaj. Osim toga, rad predstavlja potrebu za dodatnim istraživanjem u području simulacija leta unutar razvojnih okvira za igre.**

1 Uvod

U uvodu se ističe značaj simulacija leta za razvoj vojnih sustava, obuku pilota i razvoj zrakoplovnih tehnologija. Fokus rada leži na tehničkim izazovima modeliranja realističnih letnih karakteristika i njihovoj integraciji u Unity platformu. Kroz analizu ovih aspekata, rad će istražiti doprinos interaktivnih simulacija leta u virtualnom okruženju, naglašavajući važnost ovakvih sustava u kontekstu razvoja suvremenih tehnologija i obuke pilota. Ovakvi sustavi su se kroz godine pokazali kao efikasan i jeftin način obuke novih pilota zbog mogućnosti simulacije raznih uvjeta i kvarova na pojedinim komponentama zrakoplova bez opasnosti u stvarnom životu koja uči nove pilote kako reagirati u takvim situacijama. Ovaj seminarski rad se primarno bazira na sustav leta zrakoplova tako da piloti koji ga koriste mogu ga doživjeti stvarno. Simulacija leta se ne bazira samo na letenju kako bi ga mnogi zamislili dok igraju računalne igre (npr. avion ima samo palicu koja ide gore dolje, lijevo desno), nego je bazirana na sustavima koje korisniku daju osnovne informacije, poput zvukova pravog aviona, sustava za radiovisinomjer (engl. GPWS), sustava za izbjegavanje terena (engl. TAWS) i sustava za kontroliranje G sila.

2 Uloga i opis posla

U sveukupnim poslovima na izradi seminarskog rada te pisanju ovog izvješća, sudjelovano je na sljedeći način:

- Alberto Kerim - izrada zrakoplovnih i matematičkih modela leta, izrada sustava za radiovisinomjer, sustava za izbjegavanje terena, sustav za kontrolu G sila, sustava za zvuk i sustava "Heads Up Display"-a, sustava za naoružanje, postavljanje pogleda aviona, animacije pojedinih komponenti zrakoplova, izrada mape Zagreba, izrada HDRP-a za poboljšanje grafike, izrada 3D modela zrakoplova, spajanje simulatora s Xbox One kontrolerom, izrada Volumetric oblaka u simulaciji

3 Implementacija simulatora

U ovoj sekciji analizirat ćemo detalje implementacije interaktivne simulacije leta borbenog zrakoplova Dassault Rafale M [1], uređenog preko community add-on MSFS2Blend [4] i alata Gimp da bi se dobio vjerni prikaz budućeg Hrvatskog Rafala, pomoću Unity platforme. Fokus će biti na tehničkim izazovima, metodama korištenim za postizanje realističnosti te rezultatima ostvarenim kroz implementaciju. Korištenjem Unity alata ovdje ćemo navesti implementaciju simulatora, od same jezgre (implementacija zadataka), i grafičkog sučelja te izgleda (izgled i funkcionalnost simulatora i zrakoplova). Više o pokretanju simulatora u sekciji 4.

3.1 Modeliranje osnovnih funkcija leta zrakoplova

Prva podsekcija posvećena je implementaciji osnovnih funkcija leta zrakoplova. Ovdje su opisana fizikalna svojstva i jednostavni matematički modeli koji su korišteni za postizanje realističnih letnih karakteristika. Uključeni su parametri

poput mase, aerodinamičkog otpora, uzgona i sile potiska. Tijekom cijele sekcije korišteni su materijali za dobivanje tehničkih specifikacija zrakoplova [2] [3].

3.1.1 Potisak motora i otpor zrakoplova

Sila otpora zrakoplova i njegove komponente uzrokuju usporavanje zrakoplova, dok mlažnjaci pružaju silu potisaka za ubrzanje. Ravnoteža između otpora i potiska održava konstantnu brzinu te se otpor povećava kvadratno s brzinom.

3.1.2 Uzgon

Sila gravitacije prirodno vuče zrakoplov prema tlu, ali sila uzgona koje je generirana strujanjem zraka preko krila podiže ga od tla. Brzina zrakoplova utječe na njegovu silu uzgona, na primjer s većom brzinom stvara se više sile uzgona. Kad su sila gravitacije i sila uzgona izjednačeni, zrakoplov održava konstantnu visinu. Nedovoljna brzina rezultira smanjenim uzgonom i zrakoplov ulazi u stanje gubitka uzgona, takozvani “Stall”.

U našem simulatoru sila uzgona i sila otpora su se računali u centru mase zrakoplova. Te se u Unity okruženju bilo koja promjena sile uzgona i ostalih sila računala preko centra mase. Tako da je implementirana jednostavna fizika leta, dok bi za matematičko računanje strujanja zraka bilo prijeko potrebna veća snaga računala koja je za potrebe ovog projekta bila nedostupna.

3.1.3 Inducirani otpor

Povećana sila uzgona uzrokuje i veći inducirani otpor. Ravno letenje minimizira inducirani otpor, dok okretanje zrakoplova uzrokuje njegovo povećanje zbog povećanja strujanja zraka na komponente zrakoplova koji su u većoj površinskoj interakciji sa strujanjem zraka. Inducirani otpor dovodi do gubitka brzine.

3.1.4 Kut napada (AOA)

Sila uzgona ovisi o kutu napada (AOA), to je kut između vektora brzine i vektora usmjerenja nosa zrakoplova. Veći napadni kut stvara više sile uzgona do određene granice, nakon čega se vrlo brzo smanjuje, uzrokujući gubitak uzgona zrakoplova. Negativan napadni kut generira negativnu silu uzgona, gurajući zrakoplov prema dolje kombinirano sa silom gravitacije.

3.1.5 Upravljačke površine zrakoplova

Zrakoplov se okreće putem upravljačkih površina na krilima, horizontalnim i vertikalnim stabilizatorima. Krilca na krilima zrakoplova kontroliraju njegovo okretanje, kormila visine utječu na naginjanje nosa, dok kormilo pravca kontrolira skretanje. Nadalje implementirana je funkcionalnost zakrilca koja svojim spuštanjem povećavaju silu uzgona.

3.1.6 Implementacija sustava

Implementacija navedenih sustava se vršila preko C# skripti koje su dobivale ulazne parameter od kontrolera I trenutnih vrijednosti iz simulacije te su matematičkim modelom pojednostavljenih jednadžbi računate sile uzgona, otpora i ostalih komponenti.

3.2 Implementacija HUD-a

Ova podsekcija fokusira se na Heads-Up Display (HUD) koji pruža ključne informacije pilotu tijekom leta. Analizirat ćemo jednostavnu implementaciju kompasa, umjetnog horizonta, visinomjera, brzinomjera, napadnog kuta te mjerača G sile. Kvalitetna implementacija HUD-a poboljšava korisničko iskustvo, omogućujući pilotima precizno praćenje letnih parametara, povećavajući situacijsku svijest i poboljšavajući sigurnost leta.

3.2.1 Osnovni implementirani elementi HUD-a

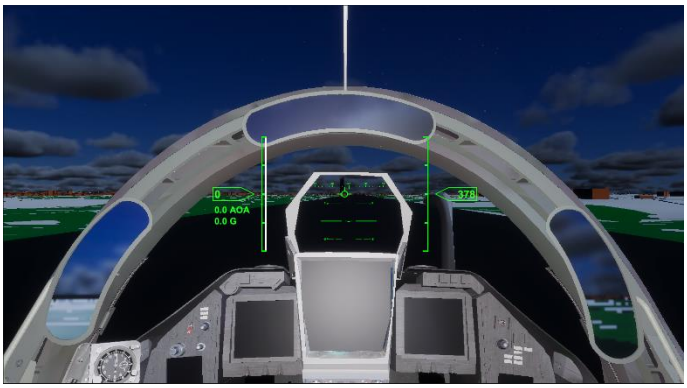
- Kompas - daje informacije o trenutnom smjeru kretanja zrakoplova.

- Umjetni horizont - simulira horizont, pomažući pilotu u održavanju ravnoteže i orijentaciji.
- Visinomjer - prikazuje trenutnu visinu zrakoplova, u stopama, u odnosu na referentnu točku.
- Brzinomjer - informira pilota o trenutnoj brzini kretanja, u čvorovima, zrakoplova.
- Indikator napadnog kuta: mjeri napadni kut zrakoplova u odnosu na relativni zračni tok.
- Indikator G sile: mjeri G silu koja djeluje na pilota, pružajući informacije o opterećenju.

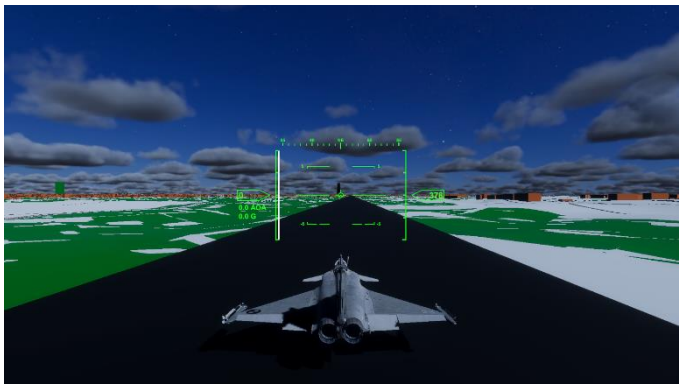
3.2.2 Važnost implementacije HUD-a

Kvalitetna integracija HUD-a u simulator leta zahtjeva precizno modeliranje svakog pojedinog elementa s naglaskom na vizualnu jasnoću i funkcionalnost. Implementacija se mora prilagoditi specifičnostima zrakoplova kako bi osigurala autentičnost letnog iskustva.

Elementi HUD-a se računaju preko skripti u C# koje temeljem podataka koje zrakoplov vraća preko Unity objekata može mjeriti svoju brzinu, visinu te ostale elemente. Sustav se pri trenutnom pokretanju simulacije postavlja na objekt kamere i djelomično je statična, ovisno o odabrano pogledu. Ukoliko pilot odabere pogled iz kokpita, na HUD ekranu u kokpitu mu se nalaze umjetni horizont te Kompas, dok su svi ostali elementi implementirani tako da se pomiču s pilotovom glavom kao što bi se i događalo u pravoj kacigi.



Slika 1. Heads Up Display u načinu pogleda iz kokpita (noćni uvjeti)



Slika 2. Heads Up Display u načinu pogleda iz 3. lica (noćni uvjeti)

3.3 Sustavi za letenje

Treća podsekcija istražuje implementaciju sustava za letenje kao što su sustav radiovisinomjera (GPWS), sustav za izbjegavanje terena (TAWS) i sustav za kontroliranje G sila na pilota. Analizirati ćemo što su i kako su ovi sustavi integrirani u simulator te kako doprinose stvaranju realističnog iskustva leta.

Radiovisinomjer (GPWS) je vitalan sustav za sigurnost, dizajniran za otkrivanje potencijalno opasnih situacija povezanih s visinom leta. Ovaj sustav koristi radiovalove za precizno mjerenje udaljenosti od zemlje ili drugih objekata. Kada GPWS detektira prijetnju, aktivira uzbunu kako bi piloti imali dovoljno vremena za reakciju. Ova funkcionalnost je ključna za sprječavanje sudara s terenom ili drugim letjelicama, čime pridonosi temeljnoj sigurnosti letenja. U simulaciji je sustav radiovisinomjera implementiran uz pomoć Physics.Raycast funkcije koja računa trenutnu vertikalnu udaljenost od objekta ili mape. Ukoliko je ta vrijednost manja od zadane pilotu se u kokpitu daje zvučni zapis o blizini terena tako da pilot pravovremeno može poduzeti pravilne korake.

Sustav za izbjegavanje terena (TAWS) nadopunjuje sigurnosnu funkciju GPWS-a. TAWS koristi sofisticirane algoritme i senzore kako bi identificirao terenske prepreke ispred zrakoplova. Ovisno o kompleksnosti, TAWS može pružiti vizualna i zvučna upozorenja, kao i preporuke za izbjegavanje terena. U našem slučaju sustav se simulira preko trenutne udaljenosti od tla koju dobivamo te ukoliko je vertikalna brzina, to jest vertikalno propadanje veće od propisanog te njegova visina manja od propisane sustav pušta zvučni zapis s informacijama za izbjegavanje terena.

Sustav za kontroliranje G sila ima ključnu ulogu u održavanju sigurnosti pilota i letjelice tijekom manevara i promjena brzine. G sile, koje djeluju na pilota u smjeru gravitacije tijekom promjena brzine ili smjera letenja, mogu imati ozbiljne posljedice ako premaše sigurne granice. Ovaj sustav prati i kontrolira G sile kako bi spriječio neželjene situacije,

uključujući i onesposobljavanje pilota uslijed prevelikih G sila. Ukoliko pozitivna G sila nadmaši dopuštenu, pilot može doći u stanje “Blackout”-a u kojem tijelo pumpa svu krv prema nogama, dok ukoliko je G sila negativna, pilot dolazi u stanje “Redout”-a koje je opasnije jer tijelo pumpa svu krv prema mozgu zbog čega može doći do oštećenja mozga. Ukoliko sustav primijeti da pilot premašuje dozvoljene G sile dobiva zvučni zapis upozorenja o prevelikoj sili.

Sustav za letenje implementiran je putem C# skripti (Slika 3.) koje se izvode cijelo vrijeme u pozadini i odgovaraju na ulaze od strane pilota. Skripta radi neovisno o drugim skriptama kao i u pravom zrakoplovu zbog važnosti tih zadataka.

```
void Update() {
    float distanceToGround = CalculateDistanceToGround();
    float verticalSpeed = GetComponent < Rigidbody > ().velocity.y;

    if (planeHUD.SpeedPub > 680 && isSonic == false) {
        StartCoroutine(PlaySonicBoom());
        isSonic = true;
    }
    if (planeHUD.SpeedPub < 680 && isSonic == true) {
        isSonic = false;
    }

    if (((distanceToGround * 3.28083) < gpwsThreshold) && ((distanceToGround * 3.28083) > 20) &&
(planeHUD.SpeedPub > 200)) {
        if (!isPlayingOther) {
            isPlayingOther = true;
            StartCoroutine(PlayGPWS());
        }
    }
    if (planeHUD.GForcePub > 9 f || planeHUD.GForcePub < -4 f) {
        if (!isPlayingOther) {
            isPlayingOther = true;
            StartCoroutine(PlayOverG());
        }
    }
    if (verticalSpeed < -(sinkRateThreshold) && distanceToGround < 1000) {
        if (!isPlayingOther) {
            isPlayingOther = true;
            StartCoroutine(PlayDontSink());
        }
    }
}
```

Slika 3. Metoda koja mjeri i vraća koje upozorenje sustav mora prenijeti pilotu

3.4 Zvukovi u simulaciji

Ova podsekcija obrađuje implementaciju zvukova unutar simulacije, kao što su zvuk motora, zvukove upozorenja od sustava za letenje, zvuk raketa te ostale. Proučiti ćemo kako su zvukovi prilagođeni različitim situacijama tijekom leta, doprinoseći realističnosti simulacije.

Zvuk motora ima ključnu ulogu u doživljaju leta. U našem simulatoru implementiran je pozadinski zvuk i šum mlaznih motora. Implementirani su i zvukovi sustava za letenje poput zakrilaca, podvozja, i upravljačkih površina proizvode karakteristične zvukove tijekom njihovog korištenja. Ovi zvukovi ne samo da pridonose stvarnosti simulacije, već služe i kao audio-indikatori za pilota. Na primjer, promjene zvuka tijekom pomicanja zakrilaca mogu ukazivati na promjene aerodinamičkih uvjeta, pružajući pilotu dodatne informacije koje mogu poboljšati situacijsku svijest. Precizna implementacija zvukova sustava za letenje igra ključnu ulogu u pružanju realističnog zvučnog iskustva koje odražava stvarne uvjete letenja. Kao što je navedeno u sekciji 3.3 Sustavi za letenje, implementirana su zvučna upozorenja pilotu.

Ako je simulacija opremljena zrakoplovima s raketnim naoružanjem, zvuk raketa ima poseban značaj. Ovaj aspekt zvukova u simulaciji dodatno poboljšava doživljaj letenja u borbenim scenarijima. Različite vrste raketa trebaju generirati specifične zvukove prilikom lansiranja, leta i udara. Osim estetske vrijednosti, ovi zvukovi također pružaju pilotima korisne informacije o stanju i ponašanju njihovog naoružanja tijekom leta.

Zvukovi su implementirani putem C# skripta u Unity-u koje koriste AudioSource te za svaki zvuk izvode svoju posebnu rutinu tako da nemaju interakciju s ostalim zvukovima i neovisni su jedni od drugih.

```
public void ToggleFlaps() {
    if (isPlayingFlaps == false) {
        isPlayingFlaps = true;
        audioSourceFlap = gameObject.AddComponent < AudioSource >
();
        audioSourceFlap.clip = flapSound;
        audioSourceFlap.Play();

        Invoke("DestroyAudioSourceFlap",
audioSourceFlap.clip.length);
        isPlayingFlaps = false;
    }
}

private void DestroyAudioSourceFlap() {
    if (audioSourceFlap != null) {
        Destroy(audioSourceFlap);
    }
}
```

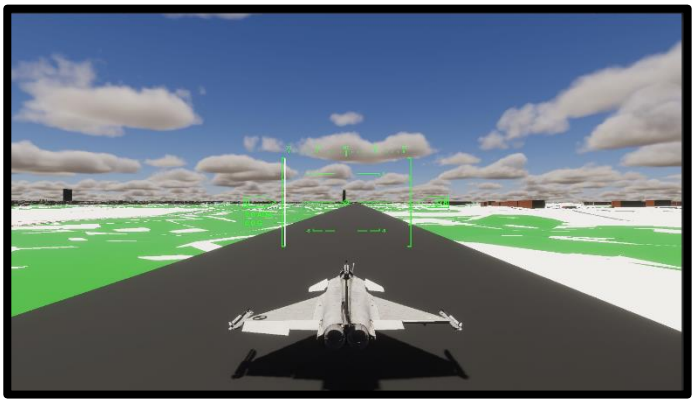
Slika 4. Implementacija metode koja reproducira zvuk pri izvlačenju i uvlačenju zakrilca

3.5 Animacije na komponentama zrakoplova

Ova podsekcija opisuje implementaciju animacija na komponentama zrakoplova kako bi se postigla veća realističnost vizualnog prikaza. Razmotrit ćemo animacije krila, repa i drugih pokretnih dijelova zrakoplova.

Animacije krila su od suštinske važnosti jer jasno vizualiziraju promjene u položaju krila tijekom različitih faza leta. Kada pilot mijenja kuteve nagiba ili koristi upravljačke površine poput krilca, animacije krila precizno odražavaju te promjene. Pokreti repa igraju ključnu ulogu u stabilnosti i manevriranju zrakoplova. Implementacija animacije repa omogućuje jasno praćenje promjena u nagibu i smjeru repa tijekom leta. Ukupno gledano, implementacija animacija na komponentama zrakoplova ključna je za postizanje vrhunskog vizualnog doživljaja u letačkim simulatorima, pridonoseći funkcionalnosti, obrazovnom aspektu i ukupnom zadovoljstvu korisnika.

Animacije krila i repa postignute su preko C# skripte zbog malog broja komponenti potrebnih za animiranje (rotacija po jednoj osi u odnosu na objekt Rafala). Oprema za slijetanje ima veći broj komponenti koje su potrebno istovremeno rotirati u određenom vremenskom okviru. Za njihovu animaciju korišteni su ugrađeni alati u Unity uređivaču, gdje je na model Rafala dodana komponenta animatora zajedno s napravljenim upravljačem animacija preko kojeg su zadani uvjeti tranzicije između pojedinih animacijskih klipova. Animacijski klipovi spuštanja i podizanja opreme za slijetanje su isti, samo im je preokrenut redoslijed, što nam omogućuje korištenje istog klipa.



Slika 5. Prikaz jednog od mogućih stanja animacije krilca, zakrilca, te kormila visine i pravca

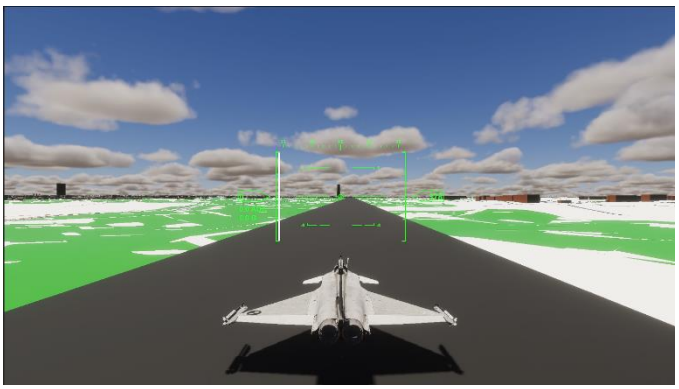
3.6 HDRP u Unity s Day-Night Controllerom i Oblacima

U ovoj podsekciji analizirat ćemo korištenje High Definition Render Pipeline (HDRP) u Unity platformi, zajedno s implementacijom Day-Night Controllera i Volumetric oblaka.

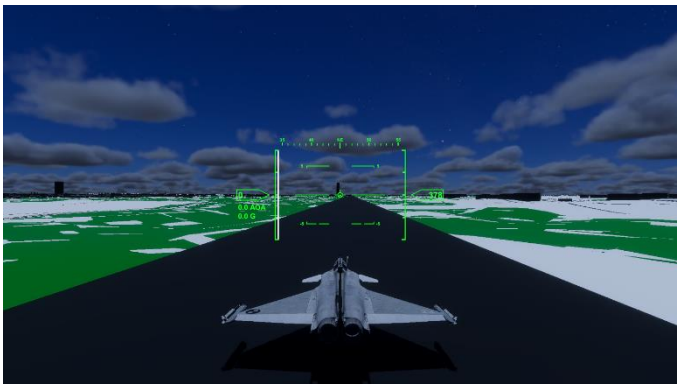
Ključne značajke HDRP koje doprinose simulaciji uključuju fotorealistične efekte kao što su poboljšane teksture, refleksije i sjene koje pridonose stvaranju uvjerljivijih vizualnih doživljaja, globalna iluminacija s HDRP koja omogućava bolje upravljanje svjetlom u sceni, što rezultira realističnijim osvjetljenjem i sjenama te praćenje sjena koje bolje doprinose dubini i dimenzionalnosti objekata u prostoru. Day-Night Controller predstavlja sustav koji dinamički kontrolira ciklus dana i noći u simulaciji. Ovaj kontroler omogućuje promjene svjetlosnih uvjeta tijekom vremena stvarajući dinamičnu atmosferu. Kroz ovu implementaciju, simulacija letenja postaje realističnija s promjenama osvjetljenja i boje neba tijekom dana i noći. Simulacija dana i noći postignuta je preko dva objekta izvora svjetlosti koji predstavljaju sunce i mjesec, te njihovom rotacijom oko x-osi koja odgovara položaju sunca i mjeseca oko scene. U uređivaču postavljene su odgovarajuće postavke za boju i intenzitet izvora svjetlosti, dok samo upravljanje njihove rotacije i drugih postavki urađenu su preko C# skripte. U skripti DayNightCycle.cs uzimamo trenutno vrijeme računala i preko njega postavljamo vrijeme u simulatoru. Na rotaciju mjeseca postavljamo pomak za 180° u odnosu na sunce, te kontroliramo koji će izvor svjetlosti bacati sjenu, jer ne smiju različiti izvori ovog svjetla bacati sjenu istovremeno. Zvjezdano nebo postignuto je dodavanjem slike zvjezdane mape u komponentu “Sky and Fog Volume” u kojoj su definirana svojstva neba, dok u skripti ovisno u rotaciji mjeseca postavljamo vidljivost na odgovarajuću vrijednost.

Implementacija sustava oblaka dodatno obogaćuje vizualni aspekt simulacije letenja. Dinamični oblaci stvaraju dodatne vizualne slojeve, pridonoseći atmosferskoj dubini i kompleksnosti. High definition render pipeline (HDRP) ima ugrađene postavke za volumetrijske oblake što nam omogućuje dodavanje realističnih oblaka u simulator.

Integracija HDRP u Unity s Day-Night Controllerom i sustavom oblaka pruža simulaciji letenja visokokvalitetnu vizualnu prezentaciju. Ove tehnologije zajedno stvaraju uvjerljivo okruženje s dinamičkim svjetlosnim uvjetima, prateći prirodne promjene tijekom vremena i doprinoseći ukupnom doživljaju stvarnog letenja.



Slika 6. Simulacija tijekom dana



Slika 7. Simulacija tijekom noći

3.7 Kontrola zrakoplova kontrolerom

Integracija kontrolera (u našem slučaju Xbox One) omogućuje korisnicima intuitivno i realistično upravljanje zrakoplovom u simulaciji. Kombinacija digitalnih gumba i analognih gumbi pruža svestranost i prilagodljivost, omogućujući pilotima da dožive autentično iskustvo letenja putem jednostavnog i ergonomski dizajniranog kontrolera. Danas se ti kontroleri koriste i za više od računalnih igara, poput upravljanja dronovima.

3.7.1 Implementacija sustava za kontroler

Sustav za kontroler je implementiran pomoću dodatnog paketa (Input System) u Unity-u koji se mora instalirati. On omogućuje upravljanje pilotima igrom ili aplikacijom pomoću uređaja, dodira ili gesti. Pri njegovom korištenju dodaje se C# skripta na koju se mapiraju određeni gumbi koji se mogu pritisnuti. Pri pritisku gumba na kontroleru skripta poziva funkciju koja joj je potrebna za obavljanje te radnje.

```
public void OnFireMissile(InputAction.CallbackContext context)
{   if (plane == null) return;

    if (context.phase == InputActionPhase.Performed) {
        plane.TryFireMissile();
        if (counter < 2) {
            counter++;
            sound.Missile();
        }
    }
}
```

Slika 8. Isječak koda za ispaljivanje rakete

3.8 Implementacija mape

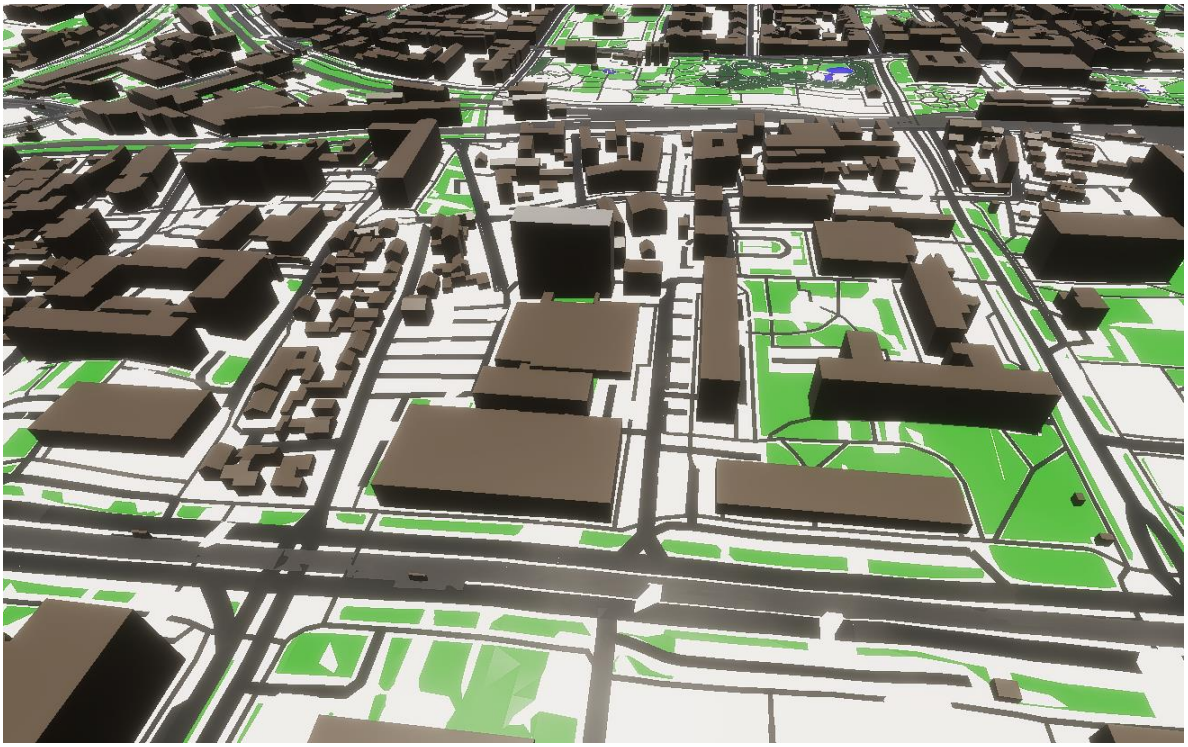
Treća podsekcija istražuje implementaciju mape Zagreba putem OpenStreetMap (OSM) [5] podataka, pružajući pilotu dodatne orijentire i okolinu za navigaciju.

3.8.1 Proces generiranja mape

Mapa grada Zagreba i njegova okolica ostvarena je koristeći besplatni community paket Blosm [6] za Blender alat. Nakon dodavanja dodatka u Blender projekt potrebno je izabrati koordinate mape. Zbog hardverskog ograničenja izabrana je manja površina. Unutar dodatka preko besplatnog API-a od OpetStreetMap (OSM) generiran je teren i svi bitni objekti kao što su zgrade, vodeni objekt, ceste, šume, željezničke pruge na temelju stvarnih geografskih svojstva izabranog prostora. Napravljenu mapu je onda potrebno iz Blendera izvesti u .fbx format koji podržava uređivač Unity. Nakon dodavanje mape unutar projekta u Unity i postavljanja na scenu, posljednji korak je na svaki od objekata mape (zgrada, cesta, ...) dodati komponentu mesh collider, kako bi objekt Rafala mogao fizički dodirnuti objekt mape, a ne proći kroz njega.

3.8.2 Razlog implementacije

Integracija stvarnog geografskog područja unutar simulacije pruža korisnicima autentično iskustvo letenja iznad poznatih lokacija, u ovom slučaju cca. 494 km^2 Zagreba. Piloti mogu prepoznati zgrade, ceste i druge značajke, što donosi stvarnost iskustvu.



Slika 9. Slika zgrade FER-a i njegove okolice generirana s OSM alatom

3.9 Perspektive simulatora

Ova podsekcija istražuje implementaciju dva različita pogleda simulacije - kokpita i treće osobe.

3.9.1 Pogled iz kokpita

Ova perspektiva pruža pilotu iskustvo letenja iz unutrašnjosti virtualnog kokpita zrakoplova. Korištenjem ove perspektive, piloti se mogu osjećati kao da zaista upravljaju zrakoplovom. Ove perspective se uvijek koriste zbog realističnosti.

3.9.2 Pogled iz trećeg lica

Ova perspektiva pruža širu sliku okoline zrakoplova izvana. Trener pilota može pratiti svoj zrakoplov iz različitih kuteva, pregledavati okolinu, pratiti let od straga ili sa strane. Ova perspektiva omogućuje pilotima bolje uočavanje odnosa između njihova zrakoplova i okoline te je često korištena u svrhu estetske i taktičke analize.

Pogledi su implementirani statičnim pomicanjem kamere i transformacijama (Slika 9.).

```
void LateUpdate() {
    if (plane == null) return;

    var cameraOffset = this.cameraOffset;

    if (plane.Dead) {
        isCockpitCamera = false;
        look += lookInput * deathSensitivity * Time.deltaTime;
        look.x = (look.x + 360 f) % 360 f;
        look.y = Mathf.Clamp(look.y, -lookAngle.y, lookAngle.y);

        lookAverage = look;
        avAverage = new Vector3();

        cameraOffset = deathOffset;
    } else {
        var targetLookAngle = Vector2.Scale(lookInput, lookAngle);
        lookAverage = (lookAverage * (1 - lookAlpha)) + (targetLookAngle * lookAlpha);

        var angularVelocity = plane.LocalAngularVelocity;
        angularVelocity.z = -angularVelocity.z;

        avAverage = (avAverage * (1 - movementAlpha)) + (angularVelocity * movementAlpha);
    }

    var rotation = Quaternion.Euler(-lookAverage.y, lookAverage.x, 0);
    var turningRotation = Quaternion.Euler(new Vector3(-avAverage.x, -avAverage.y, avAverage.z) *
movementScale);

    if (isCockpitCamera == true) {
        cameraTransform.localPosition = new Vector3(0 f, 1 f, 3.91 f);
    } else {
        cameraTransform.localPosition = rotation * turningRotation * cameraOffset;
    }
    cameraTransform.localRotation = rotation * turningRotation;
}

public void toogleCocpitCamera() {
    if (plane != null) {
        if (isCockpitCamera == false) {
            cameraOffset = new Vector3(0 f, 1 f, 3.91 f);
            lookAngle = new Vector2(110 f, 90 f);
            isCockpitCamera = true;
        } else {
            cameraOffset = new Vector3(0 f, 6 f, -20 f);
            lookAngle = new Vector2(170 f, 90 f);
            isCockpitCamera = false;
        }
    }
}
```

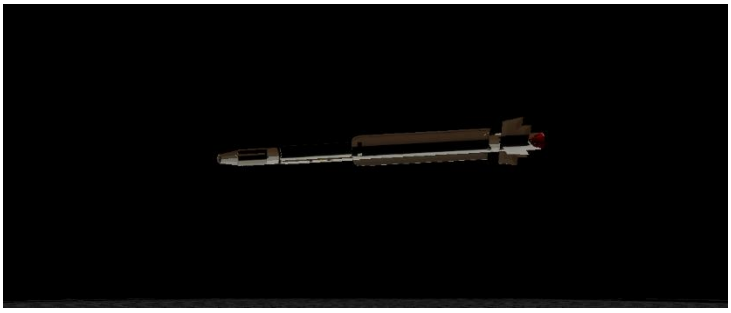
Slika 10. Metode korištene za implementaciju kamera

3.10 Sustav za naoružanje

Posljednja podsekcija analizira sustav za rakete, uključujući način ispaljivanja raketa te njihov utjecaj na let zrakoplova i okolinu. Kako svaki borbeni zrakoplov ima svoj sustav naoružanja tako je ovdje implementiran sustav naoružanja infracrvenih praćenih raketa Mica IR. U simulatoru je implementirano da Rafale M ima dvije rakete MICA IR koje može iskoristiti. Pritiskom gumba “A” (Xbox One) na kontroleru pilot simulatora ispaljuje jednu od dostupnih raketa. Raketa ja lansirana i putuje u ravnom pravcu te eksplodira pri sudaru s tlom, ili nekim drugim tijelom, ili pri isteku njenog

raktenog goriva. Rakete su popraćene s prikladnim vizualnim i audio efektima kako bi se postigao bolji ukupni doživljaj korištenja simulatora.

Implementacija raketa je napravljena uz pomoć Unity-evog alata Physics.Raycast koji otkriva kolizije između objekta rakete i bilo kojeg drugog objekta te animacija poput eksplozije koju raketa reproducira pri zabijanju u objekt ili gubitku njenog goriva.



Slika 11. Model MICA IR rakete

```
void FixedUpdate() {
    timer = Mathf.Max(0, timer - Time.fixedDeltaTime);

    if (timer == 0) {
        if (exploded) {
            Destroy(gameObject);
        } else {
            Explode();
        }
    }

    if (exploded) return;

    CheckCollision();

    if (exploded) return;
    Rigidbody.velocity = Rigidbody.rotation * new Vector3(0, 0, speed);
}
```

Slika 11.1 Metoda koja upravlja logikom rakete sa Slike 11.

4 Pokretanje simulacije i zahtjevi sustava

4.1 Zahtjevi sustava

Simulator se može isprobati preko platforma Windows (isprobano na Windows 10 Home i Pro), MacOS te Linux (isprobano na Ubuntu 20.0.4). Oprema potrebna za korištenje simulatora: Kontroler sa mogućnošću spajanja na računalo.

Zahtjevi sustava (preporučeno):

Procesor – Intel (10 Gen+) ili AMD (5 Gen+)

Memorija – 16 GB RAM (min.)

Prostor na računalu – 1 GB

4.2 Pokretanje

Simulaciju je moguće pokrenuti s operativnog sustava Windows, MacOS, i Linux. Pri pokretanju simulacije zrakoplov je pozicioniran na zračnoj luci Zagreb u smjeru 44 stupnjeva prema sjeveroistočnoj strani. Avion ima spušteno podvozje i izvućena zakrilca te mu je početna brzina 0 čvorova.

Za korištenje zrakoplova koristite ovu mapu kontrolera (preporučujemo Xbox One kontroler) -

- Left Stick - upravljanje zrakoplovom
- Right Stick - upravljanje kamerom
- Left Trigger - pomicanje kormila pravca ulijevo
- Right Trigger - pomicanje kormila pravca udesno
- Right Bumper - povećanje brzine zrakoplova
- Left Bumper - smanjenje brzine zrakoplova
- D-pad up - spuštanje/uvlačenje podvozja
- D-pad right - pomicanje kamere između pogleda kokpita i pogleda 3. lica
- D-pad down - uvlačenje/izvlačenje zakrilca
- Bottom face button - ispucavanje raketa

5 Zaključak

Seminarski rad govori o implementaciji interaktivne simulacije leta borbenog zrakoplova Dassault Rafale M u Unity platformi koja pruža dubok uvid u tehničke aspekte modeliranja i realizacije realističnog iskustva letenja. Uspješno su istraženi i savladani izazovi povezane s integracijom realističnih letnih karakteristika, vizualnih efekata, zvukova te kompleksnosti stvaranja simulacije vojnog zrakoplova u virtualnom okruženju. Naravno ovakav sustav nije niti blizu komercijalnim niti vojnim sustavima koji se trenutno koriste. U takvim sustavima je računalni hardver daleko jači od korištenog za ovaj rad. Ovaj rad naglašava važnost simulacija leta ne samo za obuku pilota, već i za razvoj vojnih sustava. Predlaže se nastavak istraživanja u području simulacija leta unutar razvojnih okvira za igre, s ciljem poboljšanja performansi i stvaranja još realističnijih iskustava. Kroz primjenu različitih tehnologija, rad doprinosi razvoju virtualnih okolina za obuku pilota, otvarajući put za daljnje inovacije u području simulacija leta. Sugerira se daljnje proučavanje simulacija leta unutar razvojnih okvira kao što su implementacija funkcionalnosti kokpita te uporaba boljih matematičkih modela. U konačnici, postiguta je minimalna implementacija simulatora za model zrakoplova koja se može usporediti sa stvarnim karakteristikama.

6 Literatura i reference

[1] alpha60. (2023.). Rafale M. flightsim.to. https://flightsim.to/file/17785/dassault-rafale-m

[2] Dassault Aviation. (2024). Rafale Specifications and Performance Data. https://www.dassault-aviation.com/en/defense/rafale/specifications-and-performance-data/

[3] Seaforces. (2023). Dassault Rafale M. Seaforces.org. https://www.seaforces.org/marint/French-Navy/AVIATION/Rafale-M.htm

[4] Bestdani. (2023.). MSFS2Blend. GitHub. https://github.com/bestdani/msfs2blend

[5] OpenStreetMap Contributors. (2023.). https://www.openstreetmap.org/#map=7/44.523/16.460

[6] Vvoovv. (2023.). Blosm. GitHub. https://github.com/vvoovv/blosm