SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2328

SIMULACIJA ZRAČNE ZAPREGE LETJELICE DRON I JEDRILICE

Bruno Bijelić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2328

SIMULACIJA ZRAČNE ZAPREGE LETJELICE DRON I JEDRILICE

Bruno Bijelić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2328

Pristupnik:	**Bruno Bijelić (0036490476)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Simulacija zračne zaprege letjelice dron i jedrilice**

Opis zadatka:

Proučiti fizikalne osnove leta letjelice dron te jedrilice. Proučiti načine modeliranja terena na osnovi realnih podataka. Ostvariti prikaz realističnog terena te simulacije zračne zaprege drona i jedrilice. Razraditi i ostvariti simulacijski model uz ostvarivanje animiranog prikaza rezultata. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranih algoritama. Izraditi odgovarajući programski proizvod. Koristiti programski grafički pogon Unity. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

SADRŽAJ

1. Uvod

Bespilotne letjelice (dronovi) su letjelice bez posade. Dronovi mogu biti raznih veličina te mogu biti inspirirani avionom ili helikopterom. Jedrilice su letjelice sličnog izgleda avionima koje lete dinamički reagirajući za zrakom kao i avioni. Međutim, za razliku od aviona, jedrilice nemaju nikakav motorni pogon. Kako jedrilica nema motorni pogon, ona mora biti puštena s velike visine da bi se mogao iskoristiti njen potencijal. Stoga se javlja ideja za zračnom zapregom drona i jedrilice, to jest da dron može nositi jedrilicu i dići je do velike visine i nakon toga je pustiti. Nakon što dron pusti jedrilicu, tada ona može slobodno manevrirati zrakom.

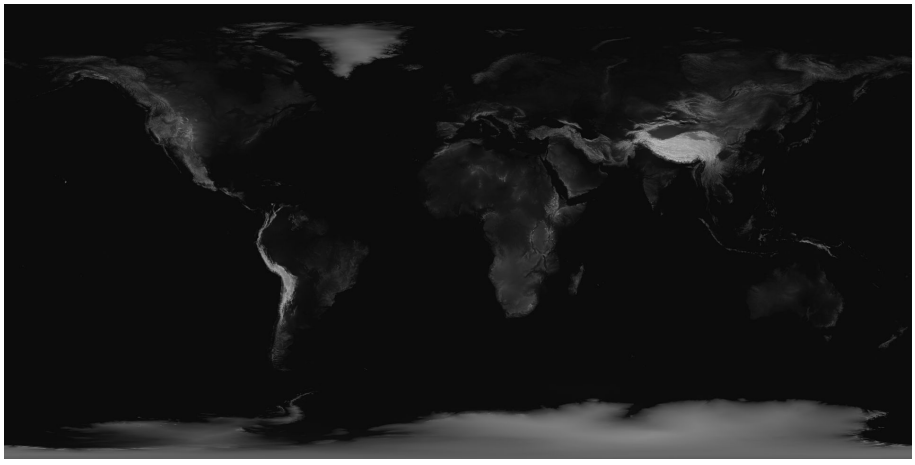
Ovaj rad se bavi grafičkom simulacijom opisane zračne zaprege. U radu će biti opisana implementacija drona i jedrilice, te njihovog spajanja i odvajanja. Velik dio rada će biti generiranje terena po kojem se kreću dron i jedrilica. Naime, bit će opisano kako implementirati generiranje terena prema stvarnim satelitskim podacima. Cijela simulacija će biti napravljena u programskom paketu Unity.

2. Izrada terena na osnovi realnih podataka

Prva faza rada je izrada terena pomoću realnih podataka. To znači da teren neće biti proizvoljan, već će biti baziran na pravim satelitskim podacima. Postoji više načina kako ovo napraviti za korištenje u Unity-u. Kroz iduća 3 potpoglavlja ćemo proći kroz nekoliko načina izrade realnog terena.

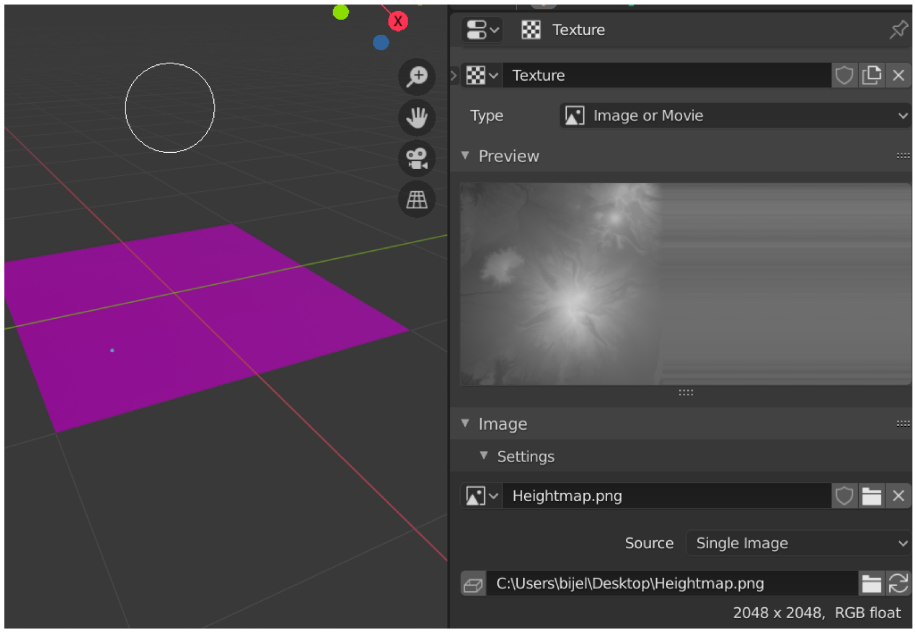
2.1. Programski alati Blender i terrain.party

Blender je vrlo moćan programski alat za izradu 3D grafike. Koristi se za izradu specijalnih efekata, animiranih filmova te modela za računalne igrice. Također, izvrstan je za izradu terena te se objekti napravljeni u Blenderu vrlo lako integriraju u Unity. Sjajna stvar kod Blendera je lakoća izrade 3D terena iz visinske mape. Naime, visinska mapa je crno bijela slika koja sadrži podatke o uzvisinama na nekom prostoru. Svaki piksel na slici je podatak o visini. Što je piksel bjelji to je taj dio slike viši, a što je crniji to je taj dio slike niži. Na slici 2.1 možemo vidjeti visinsku mapu karte svijeta.

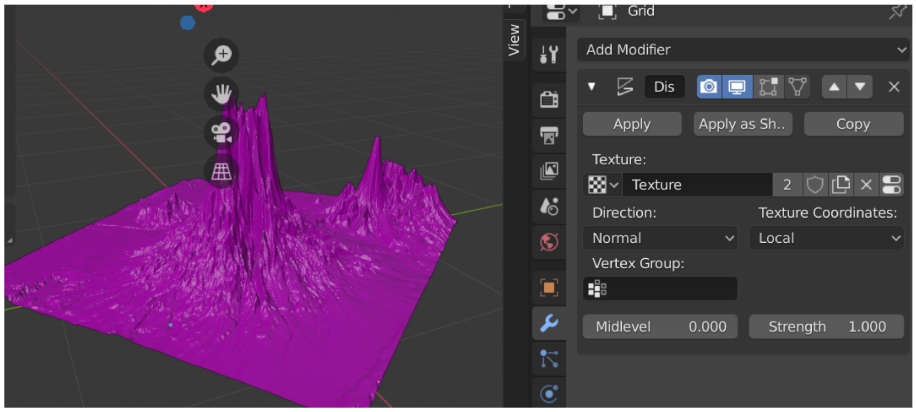


Slika 2.1: Visinska mapa svijeta

Ako imamo sliku u normalnom formatu kao što je .png format koja predstavlja visinsku mapu, izrada 3D terena iz te visinske mape se više manje svodi na kreiranje novog *Grid* objekta u Blenderu te dodavanje slike visinske mape na *Texture* dio tog objekta. Na naš *Grid* objekt samo još trebamo dodati *Displace modifier* i dobili smo našu visinsku mapu. Na slikama 2.2 i 2.3 su pokazani koraci kako iz ničega doći do jednostavne visinske mape u Blenderu.



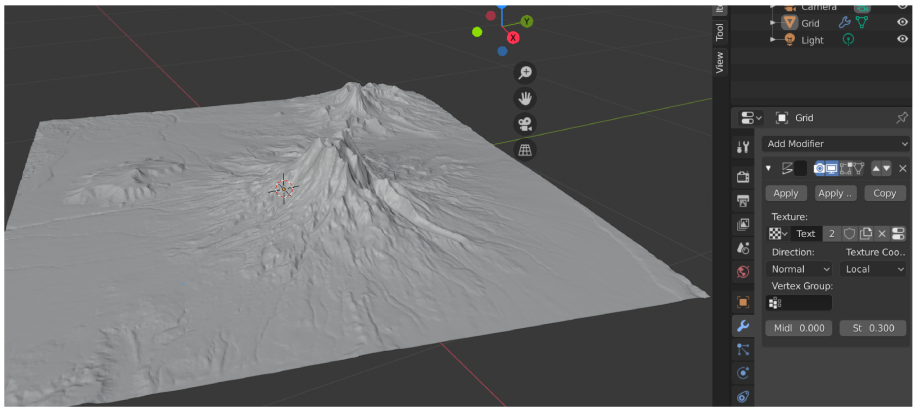
Slika 2.2: Podešavanje Texture postavki



Slika 2.3: Podešavanje Displace modifier postavki

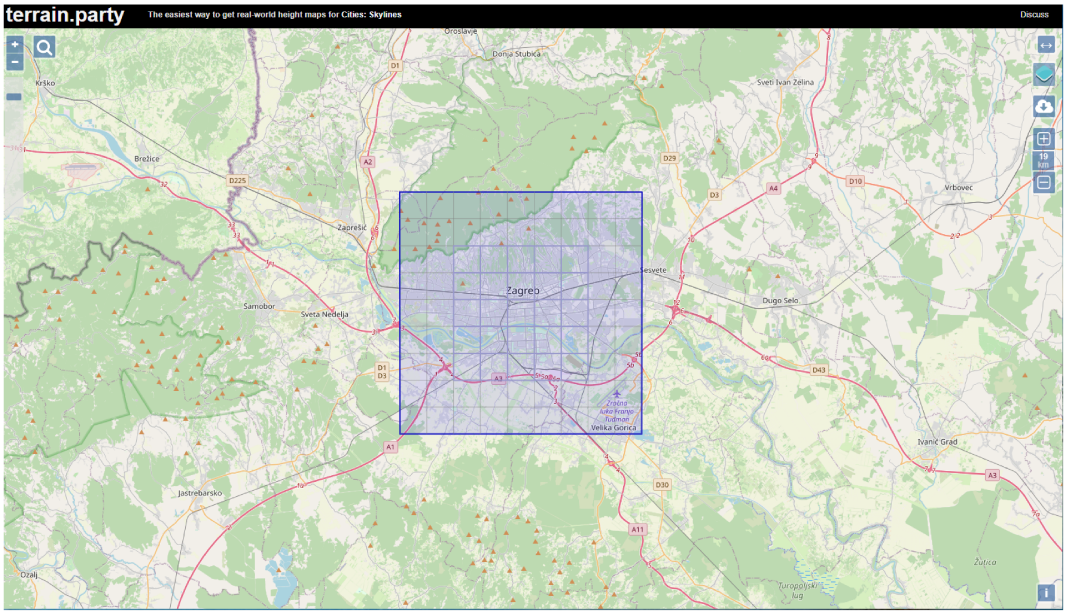
Kao što vidimo na slici 2.3 naša visinska mapa izgleda malo čudno, međutim to je samo zato što nam dimenzije nisu dobro podešene. Potrebno je još samo podesiti *Strenght* postavku objekta. Ne postoji *Strenght* koji će biti dobar za sve visinske mape,

već je to broj kojeg treba malo podešavati dok ne dobijemo mapu s kojom smo zadovoljni. Na slici 2.4 se vidi visinska mapa s podešenom postavkom *Strenght* na 0.3.



Slika 2.4: Visinska mapa sa Strenght postavkom podešenom na 0.3

Međutim, postavlja se pitanje kako nabaviti .png sliku visinske mape. Tu će nam pomoći web stranica `https://terrain.party/`. terrain.party nam omogućuje dohvat visinskih mapa bilo gdje u svijetu. Podaci za sve dijelove svijeta nisu jednako precizni. Dohvat podataka je vrlo jednostavan te na slici 2.5 možemo vidjeti kako dohvatiti podatke za grad Zagreb, a na slici 2.6 kako izgleda dobivena visinska mapa. Na ovoj slici možemo vidjeti da je terrain.party puno zanimljiviji za prirodni teren nego za gradski, jer jedino što na slici možemo razaznati je Medvednica.



Slika 2.5: Dohvat visinske mape Zagreba

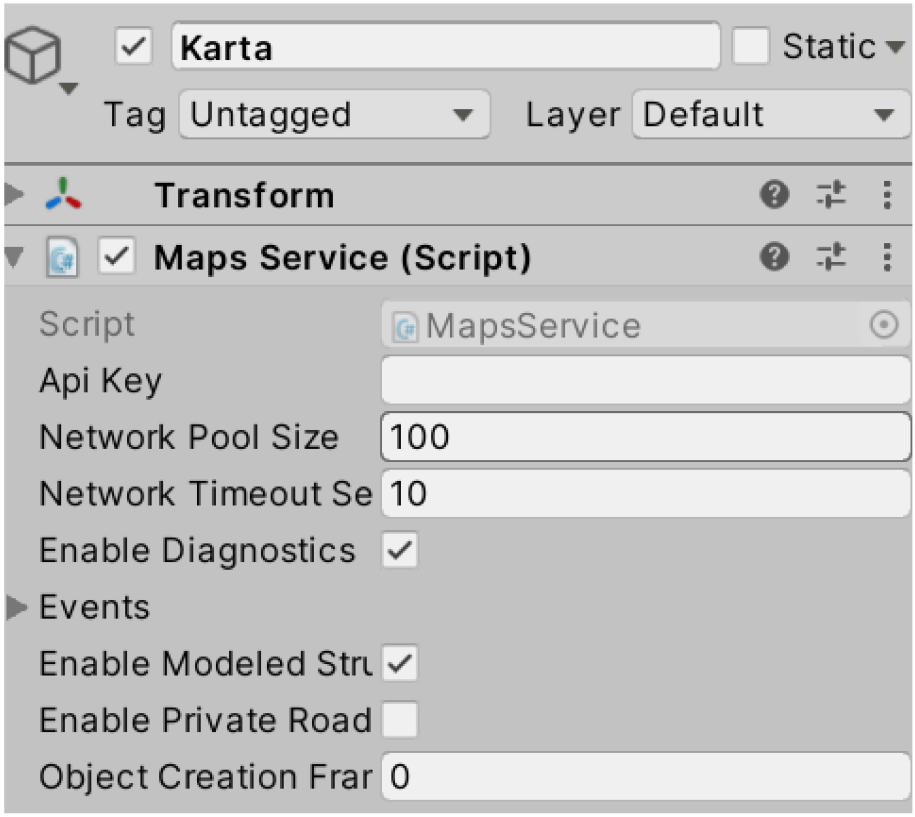
Slika 2.6: Visinska mapa Zagreba

Blender i terrain.party su odlična kombinacija za brzu izradu visinskih mapa, međutim ako želimo da to zaista lijepo izgleda potrebno je više truda, jer mi smo ovdje samo opisali kako napraviti visinsku mapu, međutim tu visinsku mapu je dodatno potrebno uređivati pomoću satelitskih slika terena kako bi dobili 3D teren koji zaista nalikuje stvarnom terenu. Također, ovaj način nam nije baš dobar u slučaju da želimo dodavati podatke o zgradama i ulicama.

2.2. Programsko sučelje Google Geocasts

Sada ćemo razmotriti izradu mapa pomoću Googleovog programskog sučelja za izradu mapa u Unity-u na osnovu podataka iz Google karata. Za ovaj način nam ne treba Blender jer Google nudi dodatak za Unity koji nam nakon instalacije omogućuje brzo stvaranje mapa. Potrebno je napraviti račun na web stranici: `https://developers.google.com/maps/documentation/gaming` na kojoj se ujedno nalaze sve potrebne upute za instalaciju i korištenje programskog sučelja. Prvi

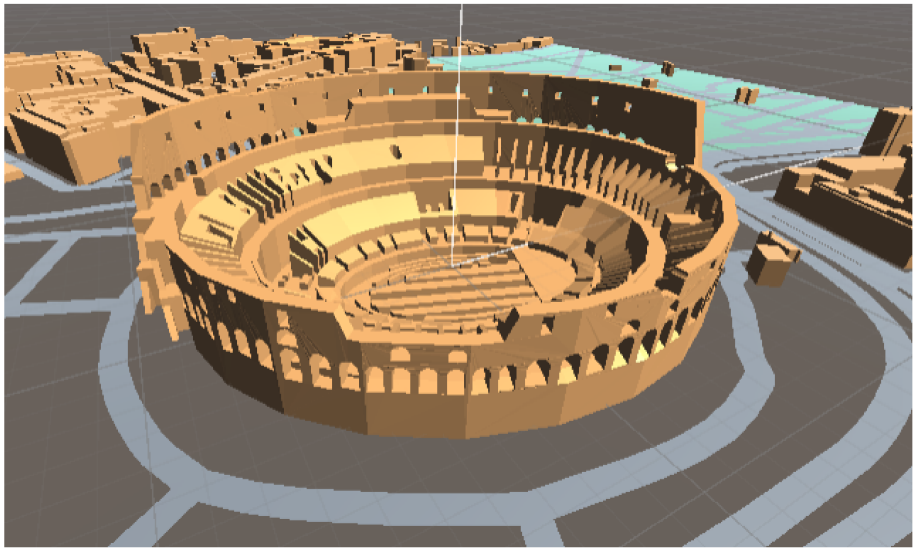
korak je instalacija programskog dodatka za Unity. Nakon toga napravimo prazni Unity *GameObject*. Na kreirani objekt treba dodati *MapsService* skriptu koja nam je dostupna nakon instalacije programskog paketa. Na slici 2.7 možemo vidjeti kako treba izgledati Unity objekt da bi mogli stvoriti kartu.



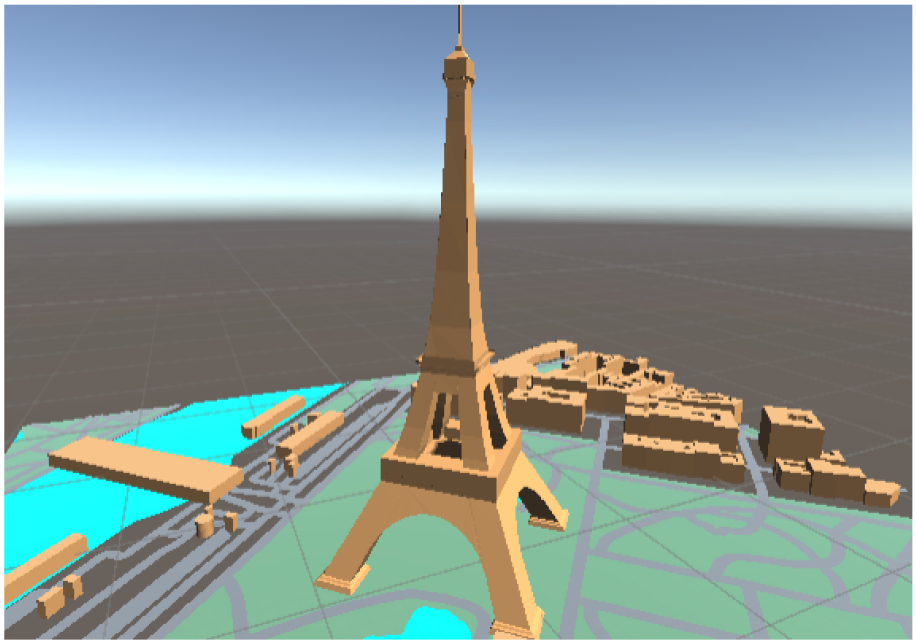
Slika 2.7: Unity objekt za mapu

Skripta ima jako puno opcija koje se mogu podešavati. Najzanimljivije za početnu izradu su *Zoom Level* čije je značenje očito te *Location* opcija koja prima 2 decimalna brojeva koje označavaju geografski smještaj centra naše mape. Slika 2.8 pokazuje kako izgleda kada za lokaciju odaberemo Kolosej u Rimu, a slika 2.9 pokazuje Eiffelov toranj u Parizu.

Kao što vidimo Geocasts omogućuje jednostavno stvaranje terena u Unity-u koji je više fokusiran na zgrade nego na sam teren. Također, teksture građevinskih objekata izgledaju generično, pošto nažalost ne postoje podaci o teksturama za svaki objekt, stoga je potrebno ručno dodati teksturu ako želimo pravi izgled Koloseja ili Eiffelovog tornja. Dodatno, za hrvatske gradove u Gooogle kartama nema tako puno podataka o zgradama. Možemo zaključiti da kombinacija Blendera i terrain.party-a nije dobra za zgrade, a Geocasts nije dobar za prirodni teren. Treći način izrade mapa će pokušati riješiti oba problema.



Slika 2.8: Rimski Kolosej

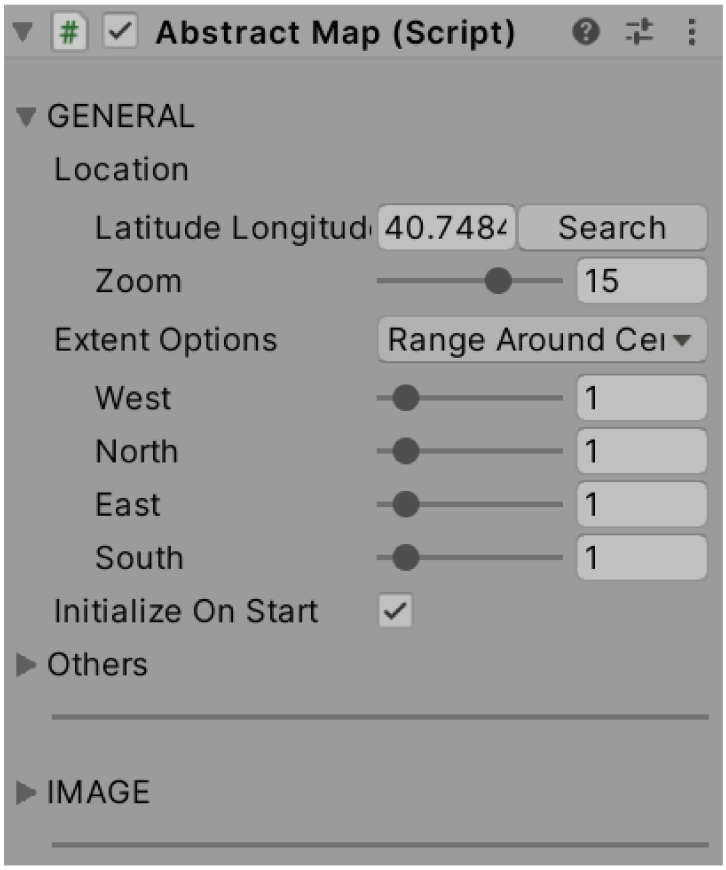


Slika 2.9: Eiffelov toranj

2.3. Programsko sučelje Mapbox

Mapbox je programsko sučelje za Unity koje omogućuje jednostavnu izradu raznih mapa. Za razliku od prva 2 načina, pomoću Mapboxa možemo lako izraditi i prirodne, i gradske terene. Kao i kod Geocastsa, prvi korak je instalirati dodatak za Unity koji se može preuzeti sa sljedeće web lokacije: `https://www.mapbox.com/unity`. Nakon toga imamo pristup Mapboxovom Unity objektu koji se zove *Map*. Dodavanjem objekta za mape u Unity scenu možemo generirati mapu. Naravno, objekt nudi

razne opcije za podešavanje mape. Podešavanje postavka mape se čini mijenjanjem postavki skripte *Abstract Map* (slika 2.10). Napravit ćemo dvije mape, jedna mapa će prikazivati Manhattan da pokažemo kako se Mapbox ponaša u gradovima, a jedna je će prikazivati Medvednicu da vidimo kako izgleda prirodni teren napravljen u Mapboxu.



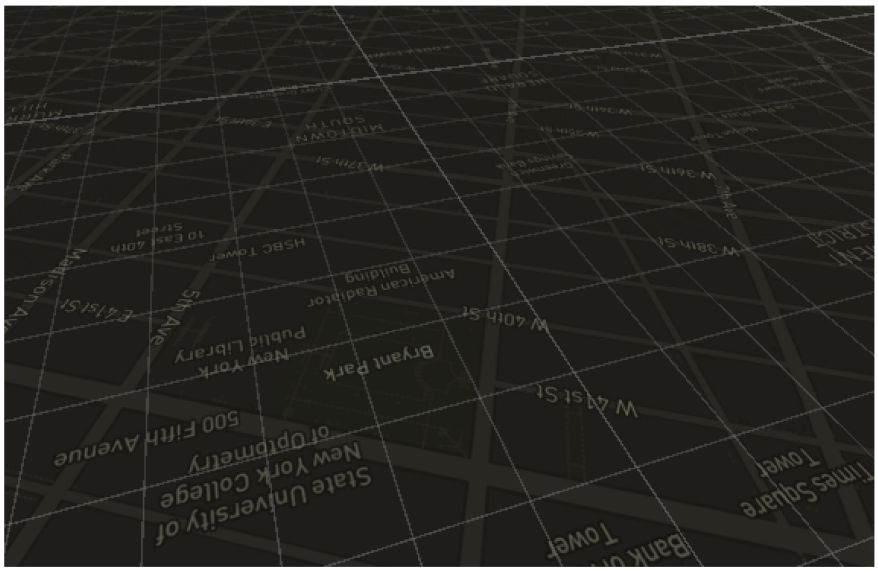
Slika 2.10: Neke postavke mape

2.3.1. Izrada mape Manhattana pomoću Mapboxa

Prvo što ćemo napraviti je podesiti *Latitude Logitude* parametar na (40.7484665, -73.985542) tako da se pozicioniramo u New York. Tada pokretanjem scene odmah ćemo vidjeti da smo zaista pozicionirani u Manhattanu (slika 2.11).

Međutim, ne želimo ovakvu mapu na kojoj samo pišu imena ulica. Prvo ćemo postaviti *Data Source* parametar u *Image* dijelu postavki na *Mapbox Dark*. Time će cijela mapa postati manje detaljna i suptilnija što želimo, zato što nam je najbitnije da se zgrade lijepo vide, a za ulice je dovoljno da su iscrtane. Na slici 2.12 možemo vidjeti izgled mape nakon ovog koraka.

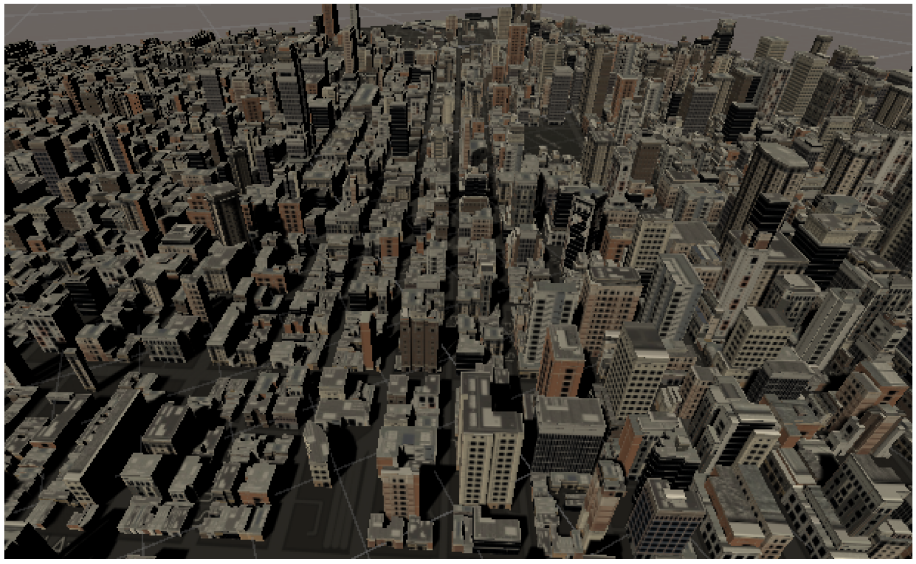
Slika 2.11: Početna mapa Manhattana



Slika 2.12: Manje detaljna mapa Manhattana

Sada slijedi najvažniji dio, a to je dodavanje zgrada. Da bi to učinili, potrebno je *Data Source* parametar u *Map Layers* dijelu postavki postaviti na *Mapbox Dark*. To će nam omogućiti da u *Features* dijelu postavki dodamo zgrade. Mapbox ima svoje teksture za zgrade što je odlično za gradski izgled mape. Na slici 2.13 možemo vidjeti konačni rezultat.

Treba napomenuti da su teksture za zgrade koje Mapbox pruža generične, tako da ako želimo da neka zgrada izgleda isto kao u stvarnosti, moramo dodati sami svoju teksturu. Na slici 2.14 možemo vidjeti Bijelu kuću u Washingtonu s Mapboxovom teksturom za zgrade. Očekivano, naša zgrada ne izgleda baš kao prava Bijela kuća.



Slika 2.13: Mapa Manhattana sa zgradama

Slika 2.14: Generirana Bijela kuća

2.3.2. Izrada mape Medvednice pomoću Mapboxa

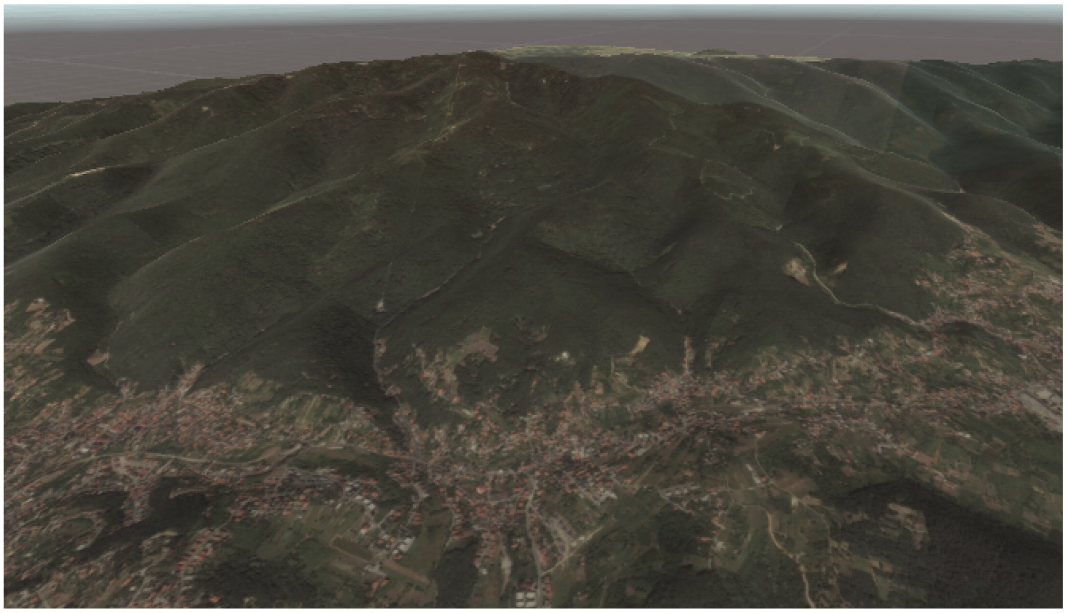
Nakon što smo demonstrirali što se može napraviti pomoću Mapboxa za gradove, vrijeme je da istražimo možemo li napraviti dobar prirodni teren, a to ćemo demonstrirati na primjeru Medvednice. Za Medvednicu ćemo *Latitude Logitude* parametar postaviti na (45.916663, 15.9666628). Kao što vidimo na slici 2.15, zaista smo se pozicionirali na Medvednici, ali mapa za sada ne izgleda zanimljivo.

Da bi mapa postala zanimljiva, želimo vidjeti mapu onako kako se ona vidi iz satelita. To ćemo napraviti tako da *Data Source* parametar u *Image* dijelu postavki postavimo na *Mapbox Satellite*. Pošto je Medvednica planina, naravno da za mapu

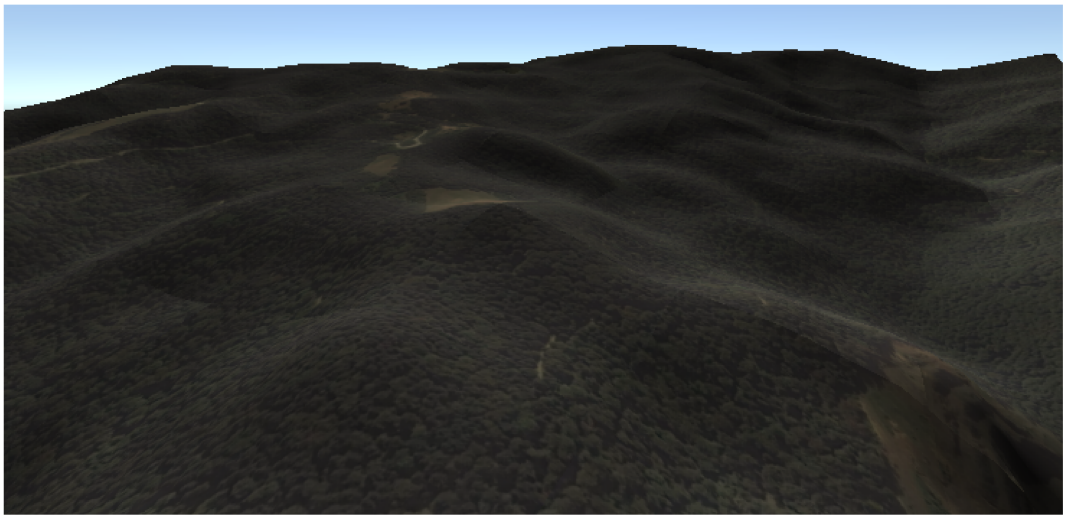
Slika 2.15: Početna mapa Medvednice

Medvednice ne želimo ravnu mapu, stoga je još potrebno podesiti *Elevation Layer Type* parametar u *Terrain* dijelu postavki na *Terrain With Elevation*. To je sve što je potrebno da dobijemo dobru mapu Medvednice koju možemo vidjeti na slikama 2.16 i 2.17.

Kao što možemo vidjeti, pomoću Mapboxa se u Unity-u mogu vrlo brzo i jednostavno stvarati razne mape načinjene po stvarnim podacima, stoga je to način koji ćemo koristiti za krajnju simulaciju.



Slika 2.16: Umanjena mapa Medvednice i Zagreba



Slika 2.17: Uvećana mapa dijela Medvednice

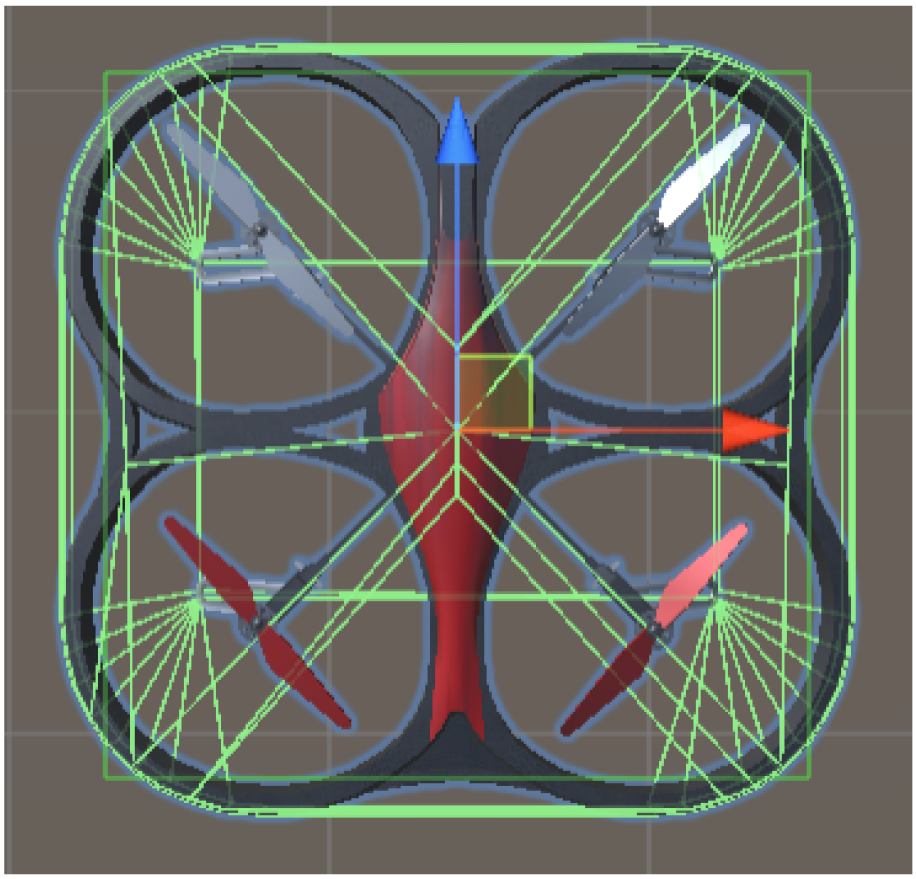
3. Izrada drona

Nakon izrade terena, potreban nam je model drona. Da bi simulirali dron treba nam za početak dizajn drona. Dizajn drona je preuzet s interneta te se može vidjeti na slici 3.1. Od estetskih stvari valja još naglasiti da se propeleri drona konstantno kreću. Taj efekt se postiže dodavanjem animacije na objekt u Unity-u. Općenito, u Unity-u je moguće raditi animacije koje su modelirane kao dijagrami stanja, to jest neki objekt ima više mogućih stanja koja se izmjenjuju ovisno o nekim pravilima [3]. Tako propeleri na ovom dronu se konstantno rotiraju, međutim rotiraju se brže u trenutku kada korisnik drži naredbu za akceleraciju drona.

Slika 3.1: Dizajn drona

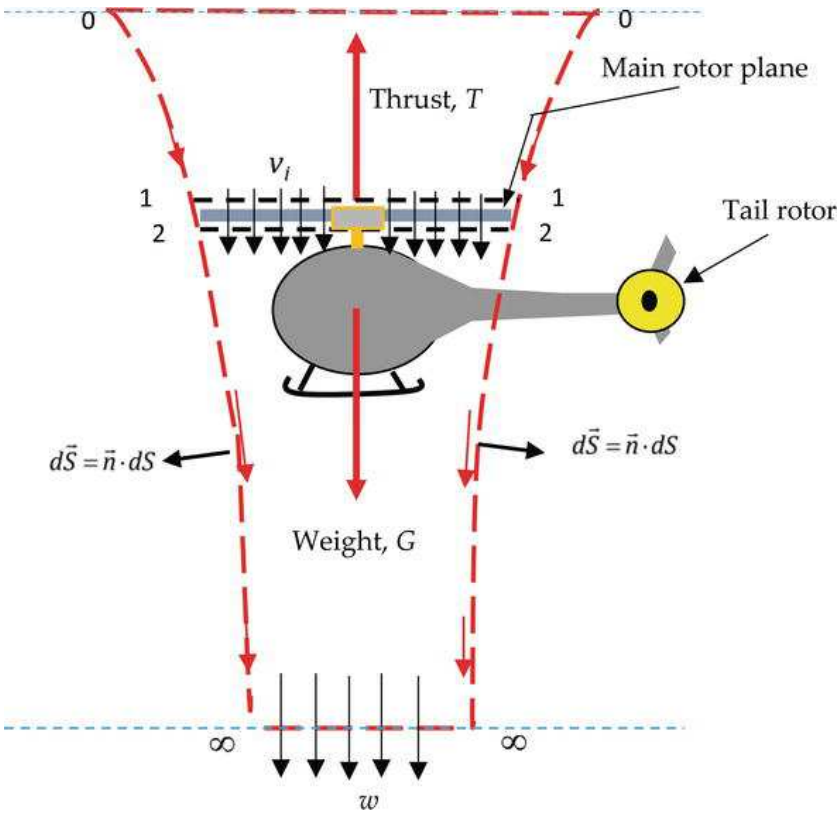
Naravno, potrebno je modelirati način na koji se dron kreće, međutim prije toga je potrebno dodati još dvije stvari na Unity-ov objekt drona, a to su sudarač i komponenta krutog tijela. Svaki objekt u Unity-u za koji želimo da može međudjelovati s drugim

objektima mora imati sudarač, zato što Unity uočava kolizije isključivo preko sudarača [3]. Sudarač je zapravo nacrt koji predstavlja način na koji Unity vidi objekt u fizičkom prostoru. Tako na primjer, možemo imati nekakav objekt vrlo kompleksnog dizajna s puno poligona, međutim ako je sudarač tog objekta običan kvadar, Unity-u je u fizičkom prostoru taj objekt ništa drugo nego kvadar. Sudarač na dronu neće biti običan kvadar, već ćemo koristiti Unity-ov *Mesh Collider*. To je vrlo koristan tip sudarača koji funkcionira tako da Unity sam pokuša nacrtati sudarač iz dizajna objekta [3]. Ponekad to ispadne jako dobro, a ponekad se objekt i nacrtani sudarač baš i ne poklapaju. U slučaju drona ispalo je dobro, te možemo vidjeti rezultat na slici 3.2. Kod jedrilice ćemo vidjeti kako je to kada *Mesh Collider* ne generira baš dobar sudarač. Osim sudarača, spomenuli smo da nam još treba komponenta krutog tijela. Komponenta krutog tijela je komponenta bez koje se objekt u Unity-u ne može kretati. Dodavanjem te komponente, objekt u Unity-u se automatski ne smatra statičkim objektom, što nam je bitno jer želimo upravljati dronom preko tipkovnice. Komponenta krutog tijela također sadrži neka fizikalna svojstva objekta kao što su masa te utjecaj gravitacije, trenja i otpora zraka na objekt [3].



Slika 3.2: Sudarači drona (zelene linije)

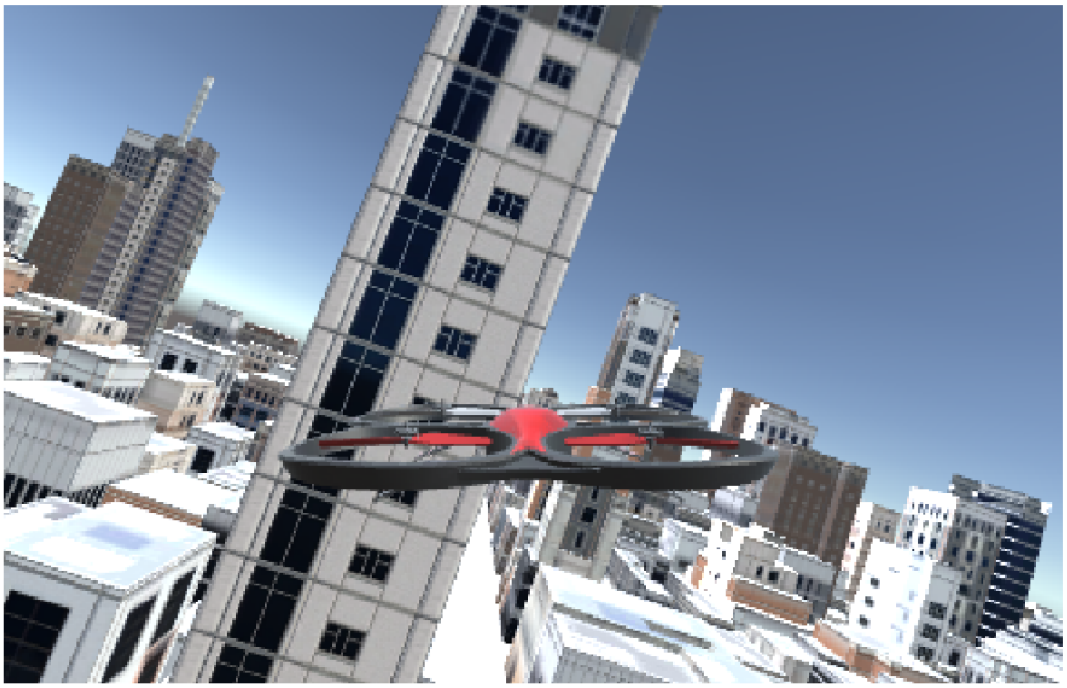
Zadnji korak je omogućiti korisniku kretanje drona pomoću tipkovnice. Htjeli bi da model drona bude što bliži fizikalnom modelu stvarnog drona, ili recimo helikoptera, pošto je naš dron relativno sličan helikopteru. Međutim, kada pogledamo fiziku iza helikoptera, to je vrlo kompleksno te se zapravo ne isplati ići implementirati sve to bez ikakvih aproksimacija i pojednostavljivanja, osim eventualno ako bi išli raditi simulator koji je toliko realan da se na njemu doslovno mogu obučavati piloti. Na primjer, samo da bi fizikalno opisali helikopter koji lebdi, potrebno je rješavati diferencijalne jednadžbe trećeg reda [2]. Na slici 3.3 možemo vidjeti koliko sila utječe na helikopter koji samo lebdi.



Slika 3.3: Prikaz sila koje utječu na helikopter

Zbog navedene kompleksnosti helikoptera, pokušat ćemo modelirati jednostavniji dron, ali naravno da želimo dron koji se ponaša što sličnije stvarnom dronu ili malom helikopteru. Kretanje ćemo modelirati tako da korisnik može dodavati akceleraciju u svih 6 smjerova (gore-dolje, ravno-natrag, lijevo-desno). Korisnik također može rotirati dron oko y osi (to je os koja predstavlja gore-dolje). Rotacija funkcionira jednostavno, kada korisnik drži naredbu za rotaciju, pomoću skripte rotiramo objekt drona oko y osi i to je to. Kada dodajemo akceleraciju dronu u jednom od 6 smjerova, ponašanje je ipak malo kompleksnije. Naime, helikopter kad se kreće ravno, malo se zarotira

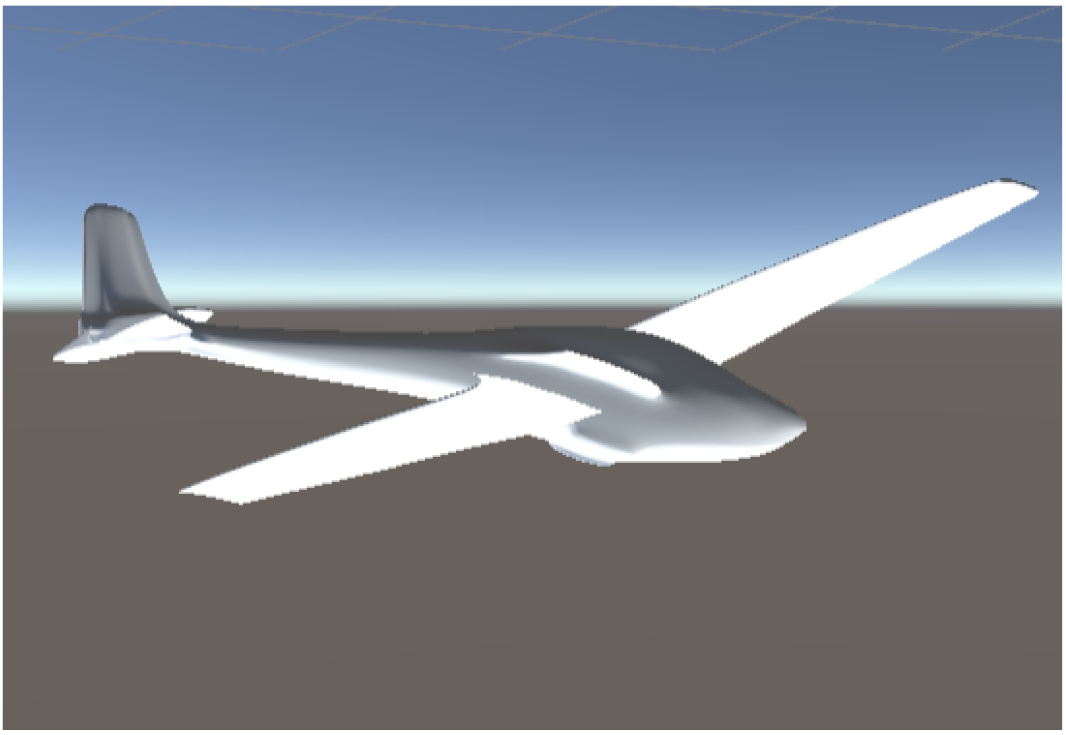
tako da mu nos gleda prema dolje, također kada helikopter ide lijevo, malo se zarotira tako da mu gornji propeler gleda prema lijevo. Želimo da se i naš dron tako ponaša, stoga kada korisnik drži naredbu za akceleraciju prema naprijed, preko skripte ćemo dronu dodati akceleraciju prema naprijed, ali također ćemo ga unutar iste skripte malo zarotirati tako da mu nos gleda prema dolje. U biti, dron kada lebdi je potpuno poravnat, a kada dobiva akceleraciju u nekom smjeru, onda ovisno o smjeru te akceleracije, malo će se zarotirati. Na slici 3.4 možemo vidjeti kako dron izgleda kada akcelerira ulijevo. Dodavanjem ove skripte za kretanje drona smo završili implementaciju drona.



Slika 3.4: Dron s akceleracijom ulijevo

4. Izrada jedrilice

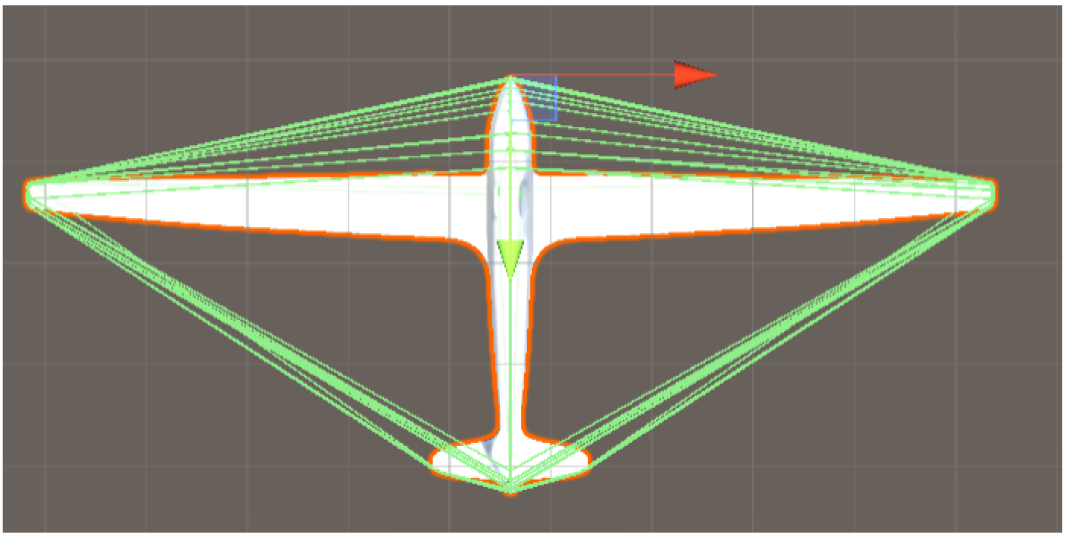
Dron nije jedini objekt kojim će korisnik moći upravljati. Također će moći upravljati jedrilicom, stoga moramo izraditi model jedrilice. Kao i kod drona, krenut ćemo s dizajnom jedrilice. Dizajn jedrilice je relativno jednostavan i također je preuzet s interneta. Dizajn jedrilice se može vidjeti na slici 4.1.



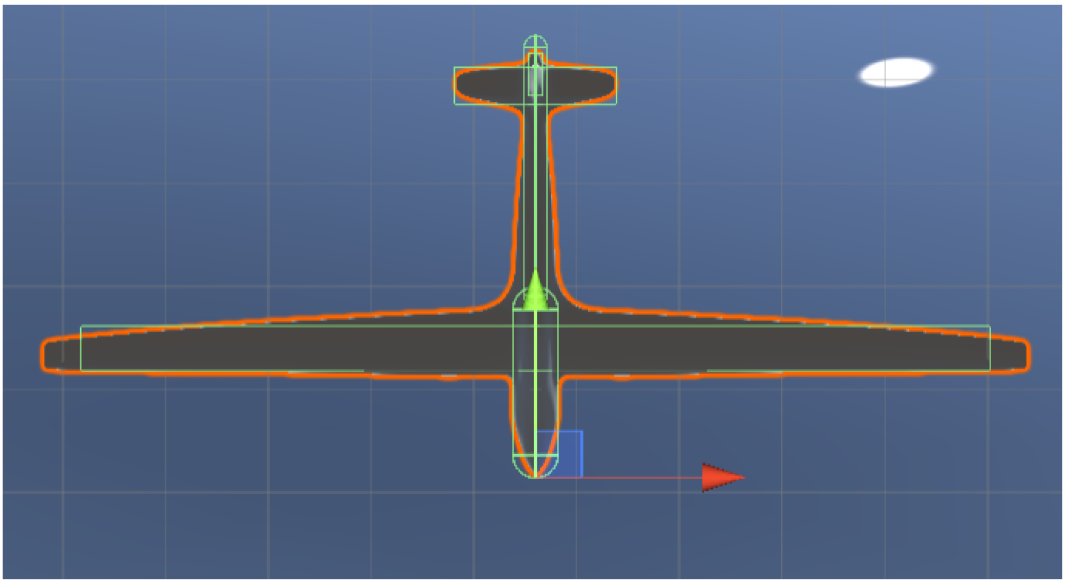
Slika 4.1: Dizajn jedrilice

Jedrilica neće biti statični objekt, stoga na Unity-ov objekt jedrilice treba dodati sudarač i komponentu krutog tijela. Dodavanje komponente krutog tijela je gotovo isto kao i za dron, ali dodavanje sudarača je nešto drugačije, zato što ovaj put Unity-ov *Mesh Collider* ne opisuje dobro jedrilicu, stoga ćemo morati sami nacrtati svoje sudarače. Srećom, dizajn jedrilice nije previše kompleksan, stoga to neće biti preveliki problem. Jedan objekt u Unity-u može imati više sudarača [3], što ćemo mi iskoristiti. Opisat ćemo jedrilicu pomoću nekoliko Unity-ovih sudarača u obliku kvadra i nekoliko

sudarača u obliku kapsule. Na slici 4.2 možemo vidjeti kako bi sudarači jedrilice izgledali da smo koristili Unity-ov *Mesh Collider*, a na slici 4.3 vidimo kako izgledaju naši ručno napravljeni sudarači.



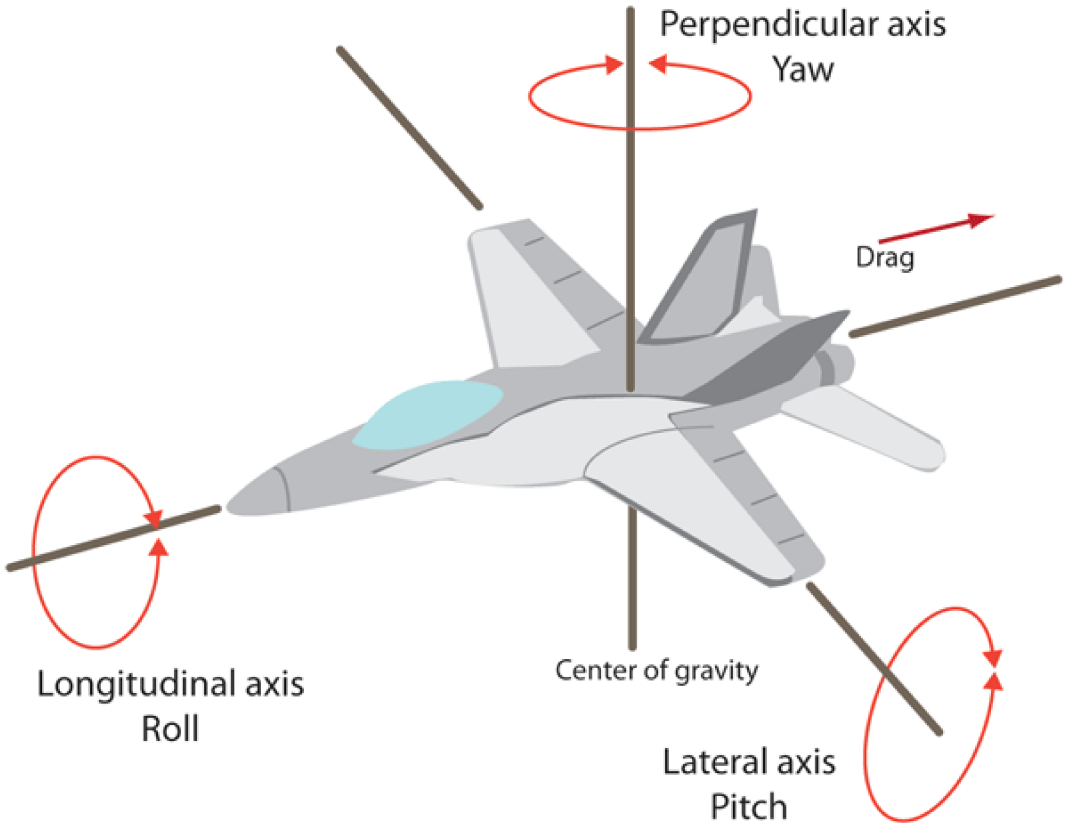
Slika 4.2: Mesh Collider za jedrilicu



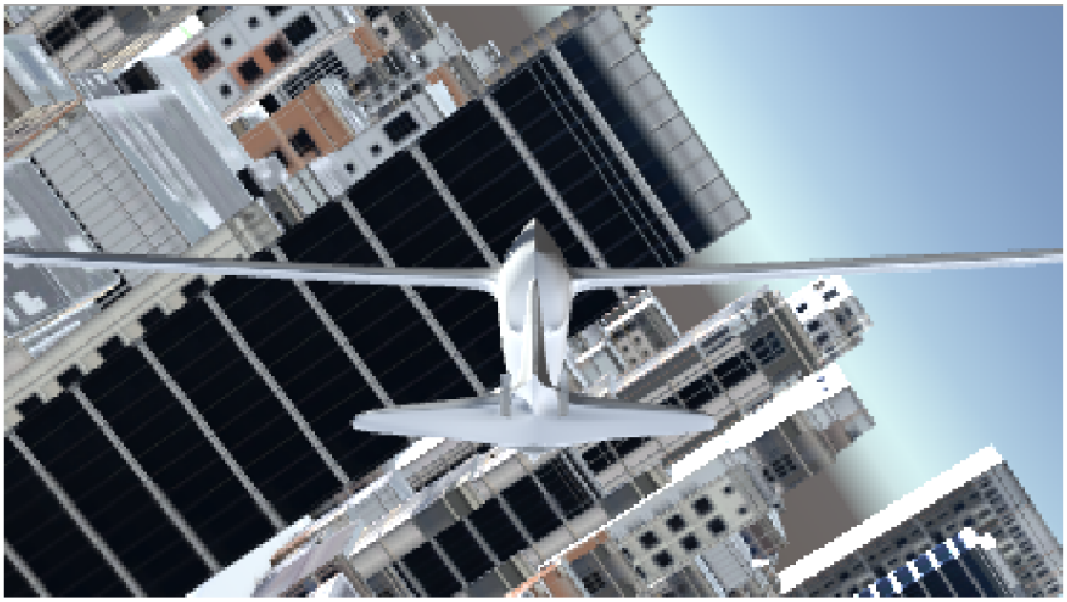
Slika 4.3: Ručno napravljeni sudarači za jedrilicu

Naposljetku, potrebno je korisniku omogućiti da upravlja jedrilicom pomoću tipkovnice. Jedrilicom će biti moguće upravljati tek nakon što se odspoji od drona. Više o tome kako i kada se može upravljati jedrilicom, te kako se dron i jedrilica spajaju i odspajaju će biti opisano u idućem poglavlju. U ovom poglavlju ćemo samo prodiskutirati o načinu na koji je modelirano upravljanje jedrilicom. Na jedrilicu možemo

gledati kao na avion bez motora, zato što jedrilica može dobivati akceleraciju samo pomoću gravitacije i pomoću drona kada je spojena na njega. Avion mijenja smjer u zraku pomicanjem svojih zakrilca. Općenito, na avion možemo gledati kao da ima 3 osi (engl. *yaw*, *pitch*, *roll*) [1]. Na slici 4.4 možemo vidjeti kako izgledaju te 3 osi. Pomicanjem zakrilca, avion se rotira oko 3 spomenute osi i tako može ići u bilo kojem smjeru u zraku. Znači, upravljanje avionom, to jest jedrilicom, možemo modelirati tako da korisnik pritiskom na tipkovnicom rotira jedrilicu oko jedne od te 3 osi, to jest postoje 6 mogućih naredbi koje korisnik može dati jedrilici, po 2 za svaku os. Pomoću rotiranja oko te 3 osi smo na jednostavan, ali relativno precizan način modelirali jedrilicu. Na slici 4.5 možemo vidjeti kako izgleda naša jedrilica kada je rotiramo oko njene longitudinalne osi (engl. *roll*) ulijevo.



Slika 4.4: Osi aviona



Slika 4.5: Model jedrilice nagnut ulijevo

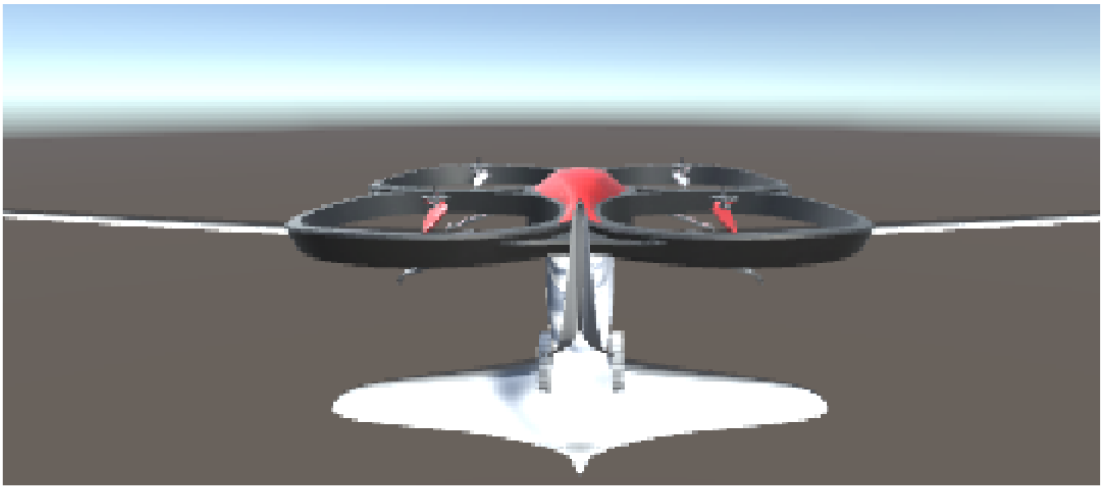
5. Spajanje komponenti i dovršetak simulacije

Obradili smo izrađivanje modela drona i jedrilice te generiranje terena. U ovom poglavlju ćemo se baviti spajanjem dosad napravljenih komponenti. Također ćemo dodati još par stvari u simulaciju kao što su vjetar i sliku okoliša (engl. *Skybox*), ali više o tome na kraju poglavlja.

5.1. Spajanje drona i jedrilice

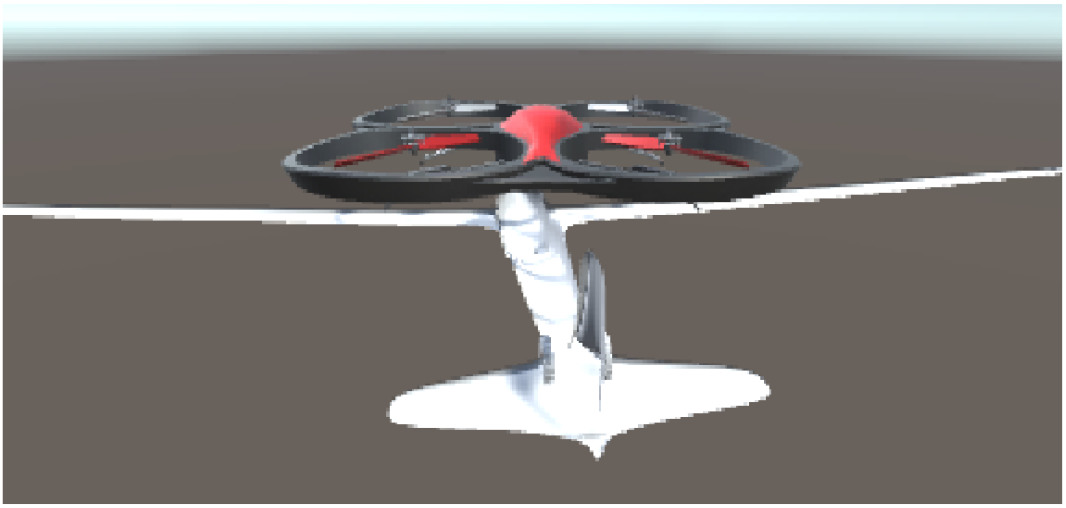
Kao što je već bilo rečeno, dron i jedrilica su dva objekta kojima korisnik simulacije može upravljati. Ono što simuliramo je zračnua zaprega jedrilice. To znači da dron može nositi jedrilicu u zraku. U bilo kojem trenutku dron može pustiti jedrilicu i tada se jedrilica može slobodno kretati. Ako se odspojeni dron i jedrilica poravnaju u zraku, mogu se ponovno spojiti (nešto slično kao što je moguće točiti gorivo iz jednog aviona u drugi avion tijekom leta). Krenimo prvo od implementacije odspajanja drona i jedrilice. Simulacija počinje tako da su u spojenom stanju (slika 5.1). Dok su u spojenom stanju, korisnik može upravljati samo dronom. Upravljanjem dronom u spojenom i odspojenom stanju je vrlo slično, samo što je u spojenom stanju malo sporiji jer jedrilica ima određenu masu. U bilo kojem trenutku korisnik može pritisnuti tipku za odspajanje (konkretno tipka za razmak, to jest *Space*). Tada se dron i jedrilica odspajaju. U trenutku odspajanja jedrilica će imati skoro identičnu brzinu kao što ju ima dron. Jedina mala razlika je zbog toga što pri odspajanju dron malo "odgurne" jedrilicu od sebe da bi se spriječio sudar između drona i jedrilice odmah nakon spajanja. Kada su dron i jedrilica odspojeni, korisnik može upravljati i dronom i jedrilicom. Pritiskom na tipku *"b"* na tipkovnici korisnik prebacuje između upravljanja dronom i jedrilicom. Kada korisnik upravlja dronom, kamera simulacije se nalazi odmah iza drona, a jedrilica se kreće i ponaša se kao kada korisnik njome upravlja, a ništa ne pritišće. Kada korisnik prebaci na upravljanje jedrilicom, tada se kamera također prebacuje na kameru koje

se nalazi odmah iza jedrilice, a dron će potpuno zakočiti, tako da se korisnik ne treba brinuti da će se dron sudariti dok on upravlja jedrilicom.



Slika 5.1: Spojeni dron i jedrilica

Potrebno je još implementirati spajanje drona i jedrilice. Ideja je da se dron i jedrilica mogu spojiti samo ako su zadovoljeni svi uvjeti koje ćemo sada navesti, zato što nije baš lako u stvarnosti spojiti dva objekta usred leta, stoga to neće biti lako ni u simulaciji. Prvi uvjet je da se dron mora nalaziti točno iznad jedrilice, otprilike moraju biti pozicionirani onako kako su kada su spojeni. Drugi uvjet je da su im zračne brzine usklađene, naime neće se moći spojiti ako se jedrilica kreće puno sporije ili brže od drona jer u stvarnosti bi spajanje ipak trajalo barem par sekundi, što znači da dron i jedrilica moraju biti barem par sekundi u kontinuitetu poravnati, što je nemoguće ako su im brzine dosta različite. Treći i zadnji uvjet je da su im rotacija usklađene, nije dovoljno da se dron nalazi točno iznad jedrilice, već trebaju biti poravnati onako kako su poravnati dok su spojeni. Na slici 5.2 možemo vidjeti dron i jedrilicu koji nisu dovoljno poravnati da bi se mogli spojiti, a na slici 5.3 vidimo dron i jedrilicu koji se mogu spojiti. Kada su svi uvjeti zadovoljeni, korisnik ih može spojiti pritiskom na tipku za spajanje (isto kao tipka za odspajanje, dakle tipka *Space*). Tada će se dron i jedrilica spojiti te će opet biti u spojenom stanju kao na samom početku simulacije. Unity dozvoljava pisanje skripti u programskom jeziku C# [3]. U Unity-u se pomoću skripti mogu implementirati razna ponašanja. Mi smo već implementirali kretanje drona i jedrilice pomoću skripte. Cjelokupna mehanika spajanja i odspajanja drona i jedrilice opisana u ovom poglavlju je također implementirana pomoću C# skripte.



Slika 5.2: Neporavnati dron i jedrilica

Slika 5.3: Poravnati dron i jedrilica

5.2. Dodavanje vjetra i slike okoliša

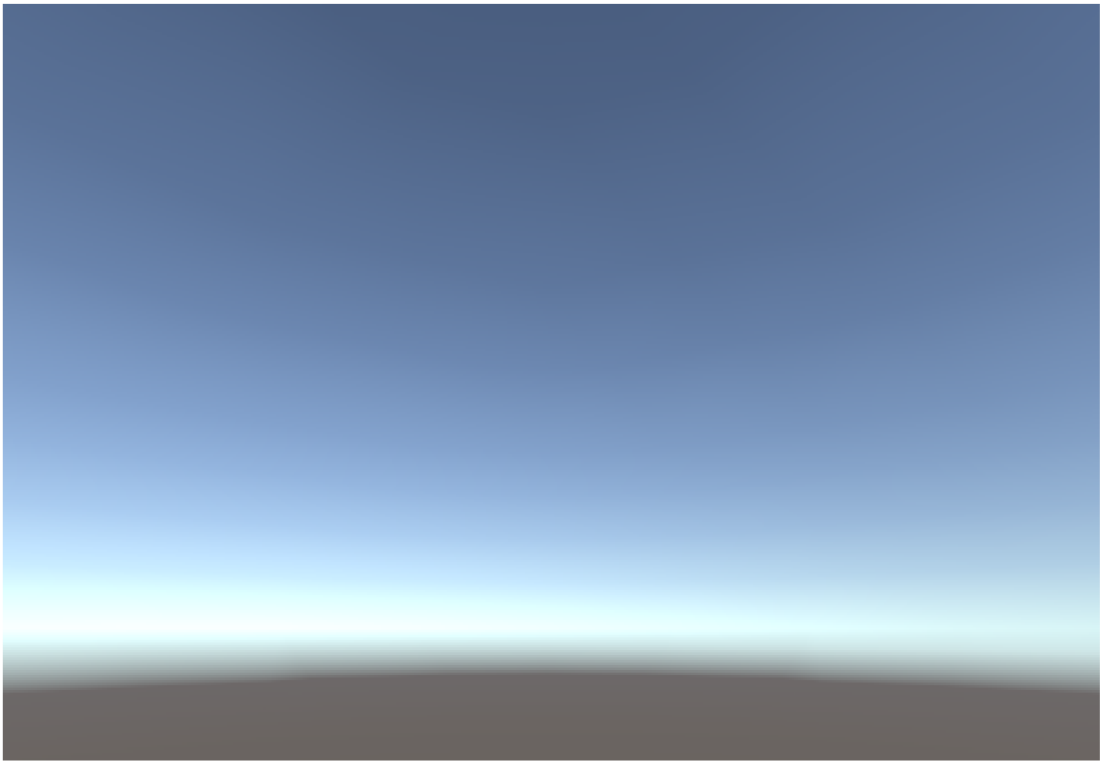
Nakon implementacije spajanja i odvajanja drona i jedrilice, te generiranja terena, svi bitni dijelovi simulacije su završeni. Međutim, u ovom potpoglavlju ćemo još dodati dva dodatka, jedan estetski, i jedan koji čini simulaciju malo zanimljivijom. Dodatak koji čini simulaciju zanimljivijom je vjetar. Vjetar neće biti vidljiv ni na koji način, ali će osoba koja upravlja dronom i jedrilicom moći osjetiti da nekakva dodatna sila utječe na dron i jedrilicu. Estetski dodatak je slika okoliša (engl. *Skybox*). Njegovim dodavanjem nebo izgleda puno realnije. Gdje god da se nalazimo u simulaciji, u daljini će se vidjeti nebo i oblaci, pa time cijela simulacija izgleda ljepše i realnije.

Prvo ćemo se pozabaviti vjetrom. Dodavanje vjetra u Unity-u nije teško, zato što u Unity-u postoji komponenta koja se zove *Constant Force* (na hrvatskom bi ovaj naziv

značio "konstanta sila"). Kao što i sam naziv kaže, dodavanjem ove komponente na objekt, na objekt će tijekom cijelog vremena simulacije djelovati određena sila. Ovime smo na vrlo brz i jednostavan način simulirali vjetar. Na slici 5.4 možemo vidjeti *Constant Force* komponentu koju smo dodali na dron. Komponenta na slici simulira vjetar u smjeru X osi.

Slika 5.4: Constant Force komponenta

Za kraj ćemo još dodati sliku okoliša kako bi simulacija izgledala ljepše. Dodavanje slike okoliša također nije teško. Naime, čim napravimo novu Unity scenu, automatski će postojati Unity objekt pod imenom *Main Camera* [3]. Na tom objektu je potrebno postaviti da je pozadina kamere *Skybox*. Postavka pozadine kamere u Unity-u može biti mnogo stvari, pa je primjerice moguće da je pozadina kamere jednobojna (engl. *Solid Color*), međutim mi želimo *Skybox*. Nakon toga, Unity će kao pozadinu simulacije pokazivati materijal koji je postavljen kao *Skybox Material* u postavkama osvjetljenja (engl. *Lightning settings*). Na slici 5.5 možemo vidjeti kako izgleda nebo s Unity-ovim zadanim *Skybox* materijalom, a na slici 5.6 možemo vidjeti kako to izgleda s malo detaljnijim materijalom preuzetim s interneta. Koristit ćemo *Skybox* materijal preuzet s interneta.



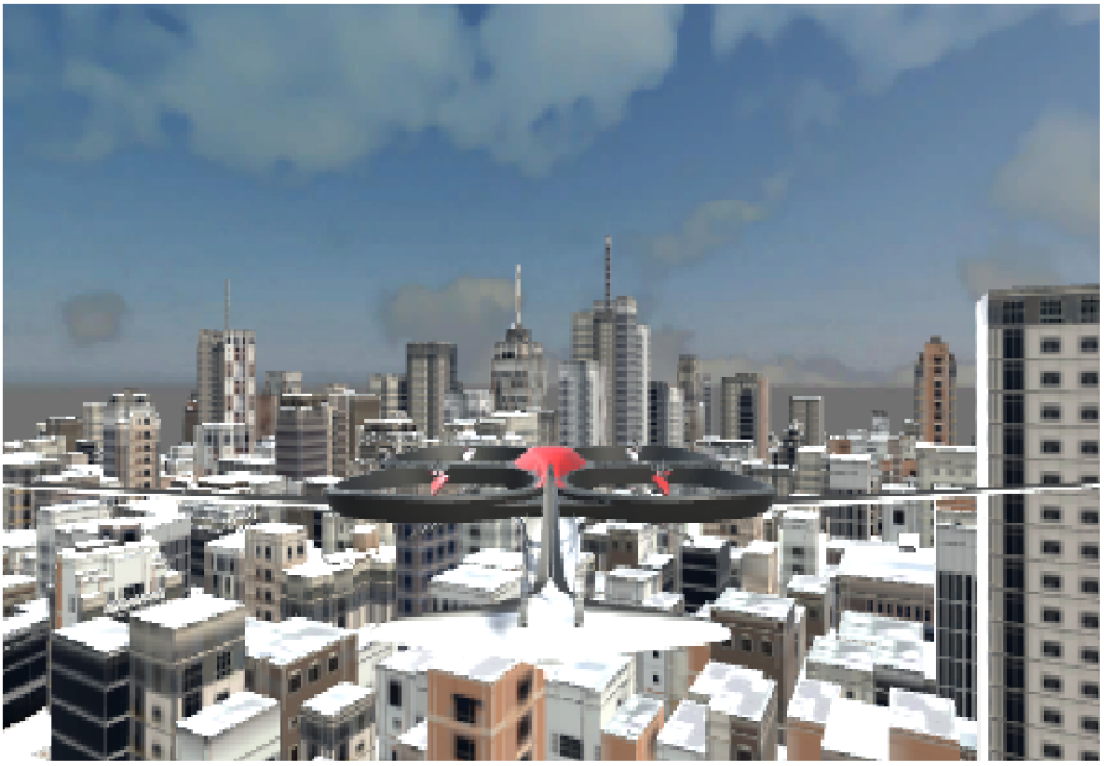
Slika 5.5: Slika okoliša s Unity-ovim Skybox materijalom

Slika 5.6: Slika okoliša sa Skybox materijalom preuzetim s interneta

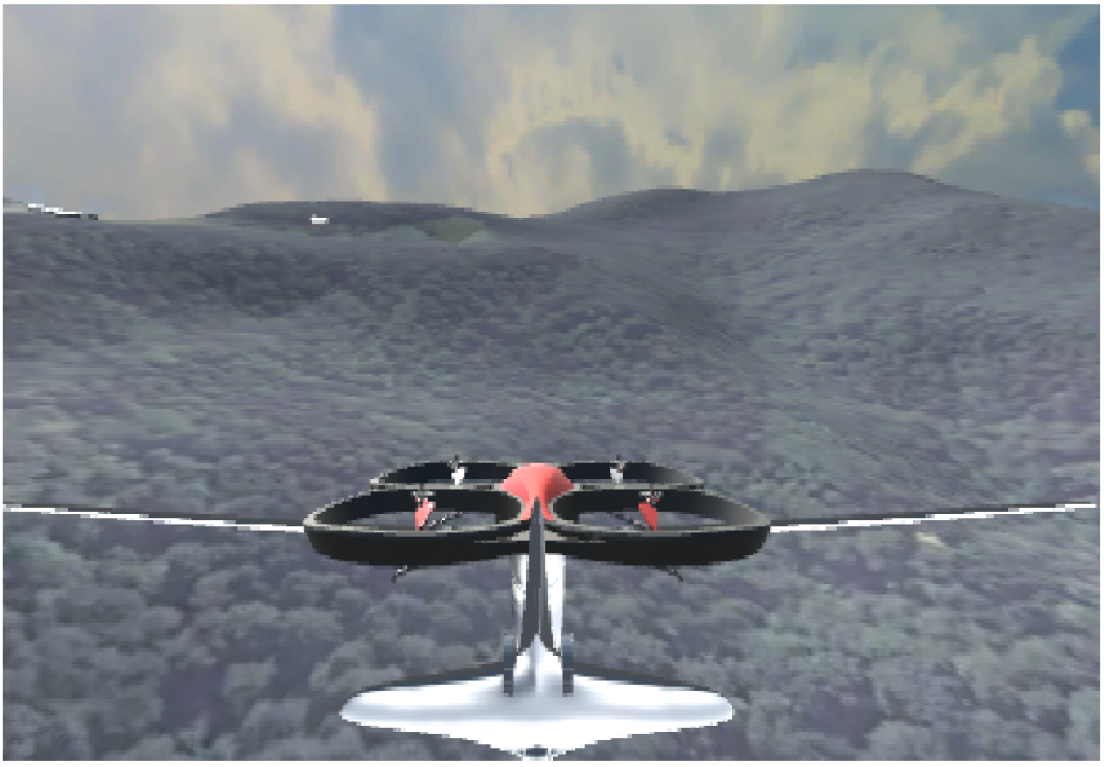
5.3. Rezultati simulacije i performanse

Dodavanjem vjetra i slike okoliša, simulacija je dovršena. Prokomentirajmo sada malo rezultate simulacije. Prvo ćemo se pozabaviti performansama. Simulacija je pokrenuta na prijenosnom računalu s Intel i7 procesorom, 16 GB RAM-a, integriranom grafičkom karticom i SSD diskom. Veličinu mape u Mapboxu zadajemo u 4 smjera (sjever, jug, istok, zapad). Te brojke nam govore koliko će se blokova mape generirati u kojem smjeru. Sve veličine moraju biti nenegativni cijeli brojevi i početno je postavljeno da su sve 4 veličine jednake 1 i tada se simulacija vrti fluidno i bez ikakvih problema. Kada je sve postavljeno na 1 dobit ćemo 9 puta veću mapu nego kada bi sve postavili na 0. Da dobijemo dojam veličine jednog generiranog bloka, kada je sve postavljeno na 0 generirat će se svega nekoliko ulica. Ako sve veličine postavimo na 3 (time je mapa znatno veća nego kada je sve postavljeno na 1), i dalje se sve vrti bez problema, ali je potrebno pričekati par sekundi da se mapa generira. Kada postavimo sve na 4, simulacija već počinje vidljivo štekati, a kada postavimo sve na 5, pojavljuje se znatno štekanje.

Pokretanjem simulacije možemo dobiti uvid u upravljanje dronom i jedrilicom, te njihovom odvajanju i spajanju. Upravljanje spojenog drona i jedrilice je slično kao upravljanje dronom na koji nije spojeno ništa. U simulaciji je napravljeno da je dron sporiji i tromiji kada nosi jedrilicu, nego kada je sam, ali i dalje nije nikakav problem upravljati dronom kada nosi jedrilicu. Upravljanje dronom, odnosno jedrilicom, kada su odvojeni je još jednostavnije, otprilike kao u video igrama, jedini je problem što se ne može upravljati i dronom i jedrilicom u isto vrijeme, pa treba paziti da se jedrilica ne sudari dok upravljamo dronom i obrnuto. Za razliku od ostalih dijelova simulacije, spajanje drona i jedrilice nije nimalo lako. Jako teško je usred leta poravnati dron i jedrilicu da se spoje, pogotovo jer ne možemo upravljati i dronom i jedrilicom u isto vrijeme. Da bi spajanje bilo moguće u stvarnosti, definitivno bi bila potrebna nekakva automatizacija tog procesa, primjerice mogućnost da se dron i jedrilica prebace u stanje autopilota kada su međusobno blizu te se automatski poravnaju i spoje, jer je čovjeku stvarno teško poravnati ih u letu kontrolirajući samo dron, ili samo jedrilicu. Sada kada je simulacija dovršena, možemo na slici 5.7 vidjeti simulirani dron i jedrilicu u New Yorku, a na slici 5.8 možemo vidjeti pejzaž drona i jedrilice iznad Medvednice.



Slika 5.7: Dron i jedrilica u New Yorku



Slika 5.8: Dron i jedrilica iznad Medvednice

6. Zaključak

Svrha ovog rada je bila da se pomoću grafičke simulacije simulira zračna zaprega drona i jedrilice te da se analizira generiranje terena prema stvarnim satelitskim podacima. U radu se analiziralo nekoliko načina za generiranje terena prema satelitskim podacima. Svaki način ima svoje prednosti i mane, te uzmimo u obzir da postoji još mnogo načina koji nisu opisani u ovom radu, tako da svakako postoji puno mogućnosti za uspješno generiranje terena prema realnim podacima za svačije potrebe. Simulacija zračne zaprege je pokazala da bi zajednički let drona i jedrilice te njihovo odvajanje trebali biti ostvarivi u stvarnosti. Međutim, ako kao korisnici probamo pokrenuti i upravljati simulacijom, brzo vidimo da ponovno spajanje drona i jedrilice usred leta zaista nije lako, tako da je to nešto što u stvarnosti definitivno ne bi bilo trivijalno za napraviti.

LITERATURA

[1] L.E. Goodman i W.H. Warner. *Dynamics*. Dover Civil and Mechanical Engineering. Dover Publications, 2013. ISBN 9780486151472. URL `https://books.google.hr/books?id=s2DDAgAAQBAJ`.

[2] Constantin Rotaru i Michael Todorov. *Helicopter Flight Physics*. 02 2018. ISBN 978-953-51-3807-5. doi: 10.5772/intechopen.71516.

[3] Unity Technologies. *Unity User Manual*, 2020.

Simulacija zračne zaprege letjelice dron i jedrilice

Sažetak

Ovaj diplomski rad implementira grafičku simulaciju drona i jedrilice koji se međusobno mogu spajati i odspajati u zraku. Detaljno je opisan proces izrade modela drona i jedrilice, te je također opisano nekoliko načina kako napraviti teren generiran prema stvarnim podacima. Sve simulacije su napravljene koristeći grafički pogon Unity.

Ključne riječi: dron, zračna jedrilica, mape generirane prema stvarnim podacima, grafička simulacija, Unity

Drone and Glider Flight Simulation

Abstract

This master's thesis implements a graphical simulation of a drone and a glider which can connect and disconnect to each other while airborne. The process of creating the model of the drone is thoroughly described as well as the process of creating the model of the glider. Several ways to create a real world terrain were also explored. All simulations were created using Unity.

Keywords: drone, glider, real world terrain, graphical simulation, Unity