**VILNIAUS UNIVERSITETAS**

**MATEMATIKOS IR INFORMATIKOS FAKULTETAS**

**PROGRAMŲ SISTEMŲ KATEDRA**

**LĖKTUVO TŪPIMO IR KILIMO TRAJEKTORIJŲ  
TRIMATIS VIZUALIZAVIMAS**

**3D VISUALISATION OF  
PLANE LANDING AND TAKE-OFF TRAJECTORIES**

**Bakalaurinis darbas**

Martynas Žuravliovas (parašas)

Darbo vadovas:

doc. Kristina Lapin (parašas)

Vilnius, *2016*

Turinys

[Įvadas 3](#_Toc481771741)

[Motyvacija 3](#_Toc481771742)

[Tikslas 3](#_Toc481771743)

[Uždaviniai: 3](#_Toc481771744)

[1. Oro uosto tūpimo ir kilimo procedūrų analizė 4](#_Toc481771745)

[2. Idealios trajektorijos integracija 7](#_Toc481771746)

[3. Trajektorijos rizikos valdymas 9](#_Toc481771747)

[4. Nukrypimo nuo idealios trajektorijos atvaizdavimas 10](#_Toc481771748)

[5. Kameros 11](#_Toc481771749)

[6. Realaus reljefo pritaikymas terrain objektui 12](#_Toc481771750)

[Rezultatai ir išvados 15](#_Toc481771751)

[Šaltiniai 16](#_Toc481771752)

# Įvadas

Daugelyje kompiuterinių sistemų galima atrasti trūkumų, netikslumų, nepatogumų, tačiau tai kas liečia žmogaus, o šiuo atveju, kai analizuojami lėktuvų skrydžiai, daugelio žmonių saugumą, klaidų ir netikslumų pasitaikyti negali. Tokiuose oro uostuose, kur nėra galingų lėktuvo tūpimą valdančių sistemų, skrydžio vadovas yra žmogus reguliuojantis eismą. Kuriama sistema yra skirta tam, kad šis žmogus galėtų lengviau stebėti kaip leidžiasi lėktuvai, kuriems jis paskyrė vienokią ar kitokią nusileidimo procedūrą. Sistema turėtų aiškiai pateikti vykdomo skrydžio vaizdą.Papildyti

## Motyvacija

Pagal žinomas koordinates vizualizuoti trajektoriją, kai visi trajektorijos duomenys yra žinomi iškart. Duomenys – koordinačių seka. Šiuo metu skrydžio vadovas duoda leidimą pilotui užimti tūpimo-kilimo erdvę. Patį tūpimą vykdo pilotas. Leidimas apima tūpimo procedūrą, kuri nustato visus tūpimo parametrus. Kuriama sistema turi parodyti skrydžio vadovui ar pilotas laikosi nurodytos procedūros. Sprendimas turi reikalauti kuo mažiau mintinių pastangų, turi būti paremtas atpažinimu, o ne mintiniais skaičiavimais. Ši sistema yra kuriama mažiems oro uostams, neturintiems instrumentinės tūpimo įrangos. Sistema turi veikti lidaro ir radaro pagrindu. Radaras neteikia duomenų mažame aukštyje, todėl lėktuvai yra nematomi skrydžių vadovui, tačiau aukštis netrukdo lidarui, taip atsiranda galimybė patikrinti, ar pilotas laikosi nurodymų. Ruošiamas prototipas yra 2-ojo lygio skalėje iš 9, pagal NASA technologijų parengimo tvarką. Papildyti, kad negazdinti žmonių.

## Tikslas

Sukurti atpažinimu grįstą programos prototipą, kuri leistų skrydžių vadovui ekrane stebėti lėktuvo nusileidimo ir pakilimo trajektorijas.

## Uždaviniai:

* Oro uosto tūpimo ir kilimo procedūrų (toliau darbe – ideali trajektorija) analizė.
* Idealios trajektorijos integravimas į esamą prototipą.
* Trajektorijos rizikų valdymas.
* Išanalizuoti būdus, kaip pažymėti nukrypimą nuo trajektorijos ir parodyti skrydžio vadovui atsiradusius netikslumus (Informacinės lentelės, su aiškiu būsenos atvaizdavimu, taip pat galimybė išjungti vieną ar kitą vizualų komponentą).
* Kameros buvimo vietos analizė bei galimybė persijungti į kitą perspektyvą, siekiant sumažinti skrydžio vadovo dėmesio poreikį.

# Oro uosto tūpimo ir kilimo procedūrų analizė

Šios sistemos prototipo įeities duomenys yra Lidaro pateiktos koordinatės (1 pav.), kurios gaunamos lėktuvui esant nedideliame aukštyje, todėl, iš analizuotos procedūros (2 pav.), svarbiausi yra tuptinės duomenys. Oro uosto pateikiama lėktuvo nusileidimo arba pakilimo procedūra yra sudėtinga, o sistemai automatiškai ją nuskaityti yra dar sudėtingiau, todėl, net ir kuriant galutinį prototipo produktą, bandyti iš šios procedūros išgauti idealią trajektoriją yra netikslingas ir daug laiko kainuojantis darbas. Todėl nuspręsta idealios trajektorijos koordinates leisti pateikti pačiam naudotojui. Tokiu atveju sistema taptų daug lengviau pritaikoma skirtingiems oro uostams.

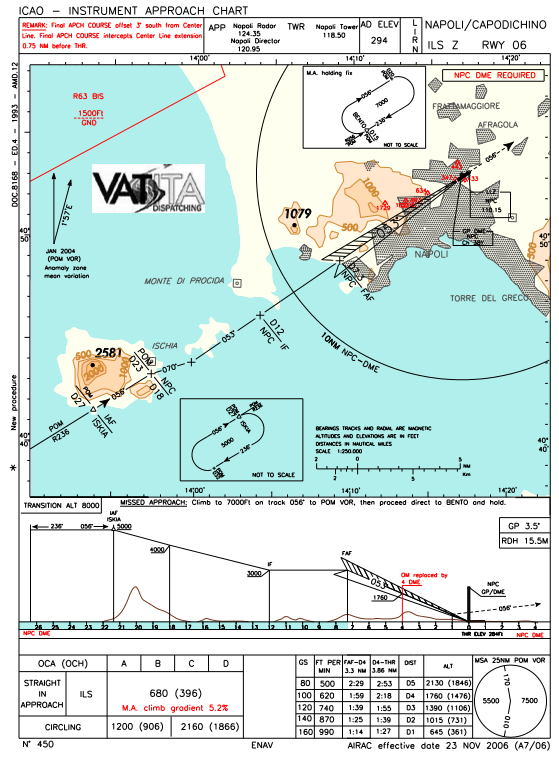
1 pav. Kuriamos sistemos struktūra

Kuriama sistema

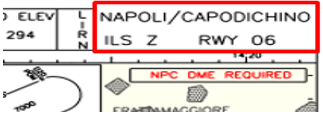
Esama lėktuvo pozicija, rizikos, skrydžio būsena

Oro uosto procedūros

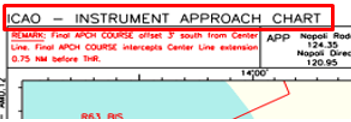
Eismo ore valdymo taisyklės



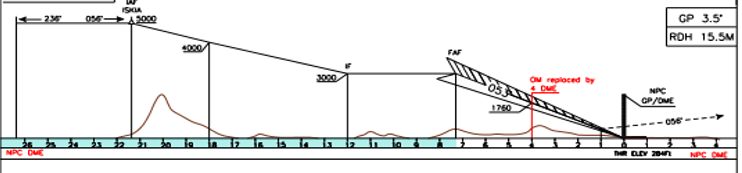
2 pav. Oro uosto pateikiama lėktuvo nusileidimo procedūra

1. Oro uosto pavadinimas, paskirtos nusileidimo procedūros kodas –ILS Z bei nusileidimo kampas RWY 06 (24 pav.).

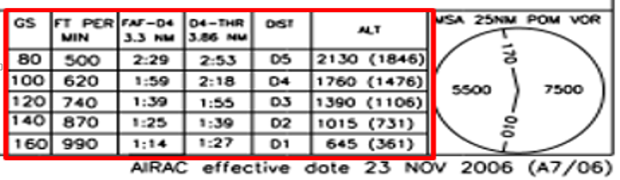
3 pav. Oro uosto pavadinimas, nusileidimo procedūros kodas, nusileidimo kampas.

2. Oro uoste esanti instrumentinė sistema (25 pav.).

4 pav. Instrumentinė oro uosto sistema

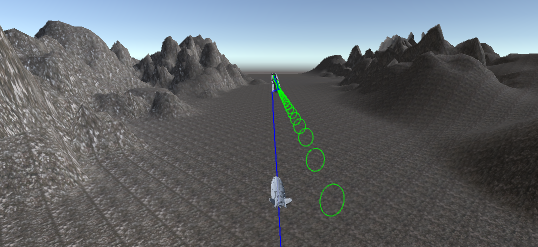
3. Paskirtos procedūros nusileidimo trajektorijos nustatymai. Šioje dalyje atvaizduojami atstumai iki tūpimo taško pradžios, bei kokiame aukštyje (matavimo vienetas – pėdos) tam tikru atstumu (matavimo vienetas – jūrmylės) iki tako pradžios turi būti lėktuvas (26 pav.).

5 pav. Idealios trajektorijos ašis ir jos nustatymai: aukštis, atstumas, nusileidimo kampas.

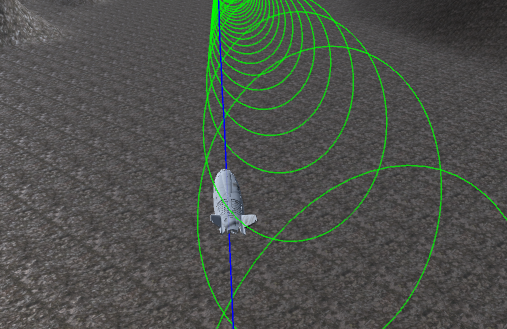
4. Detali lėktuvo judėjimo, paskirta procedūra, lentelė (27 pav.). GS(Ground speed) – horizontalus greitis matuojamas mazgais, FT PER MIN(Foot per min) – vertikalus greitis matuojamas pėdomis, FAF-04 3.3 NM – laikas, kada bus pasiektas FAF taškas judant konkrečiu greičiu.

6 pav. Lėktuvo teisingo judėjimo duomenys.

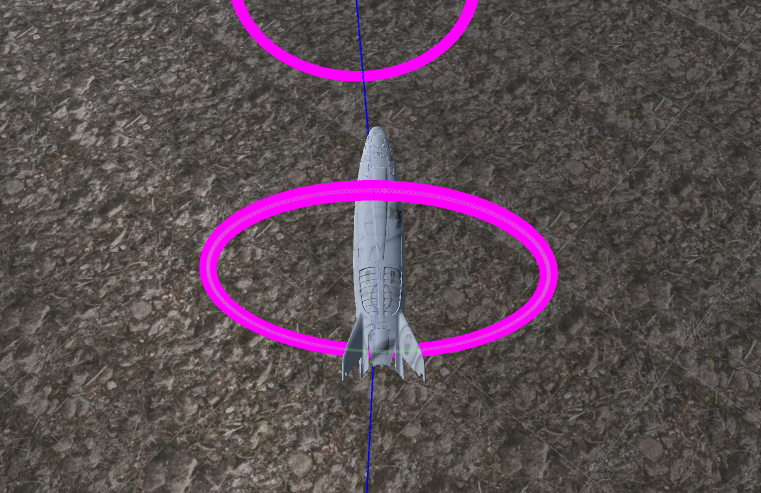
# Idealios trajektorijos integracija

Idealios trajektorijos nuskaitymui buvo panaudotas tas pats metodas, kuris nuskaito realios trajektorijos koordinates. Sukurtas atskiras idealios trajektorijos koordinačių masyvas. Pakoregavus esamą „PathFollower“ skriptą taip, kad tunelis būtų piešiamas naujai nuskaitytai idealiai trajektorijai. Taip gaunamas pirminis, realios skrydžio trajektorijos palyginimo su idealia, vaizdas (1 pav.).

7 pav. Pirminis realios skrydžio trajektorijos palyginimo su idealia vaizdas

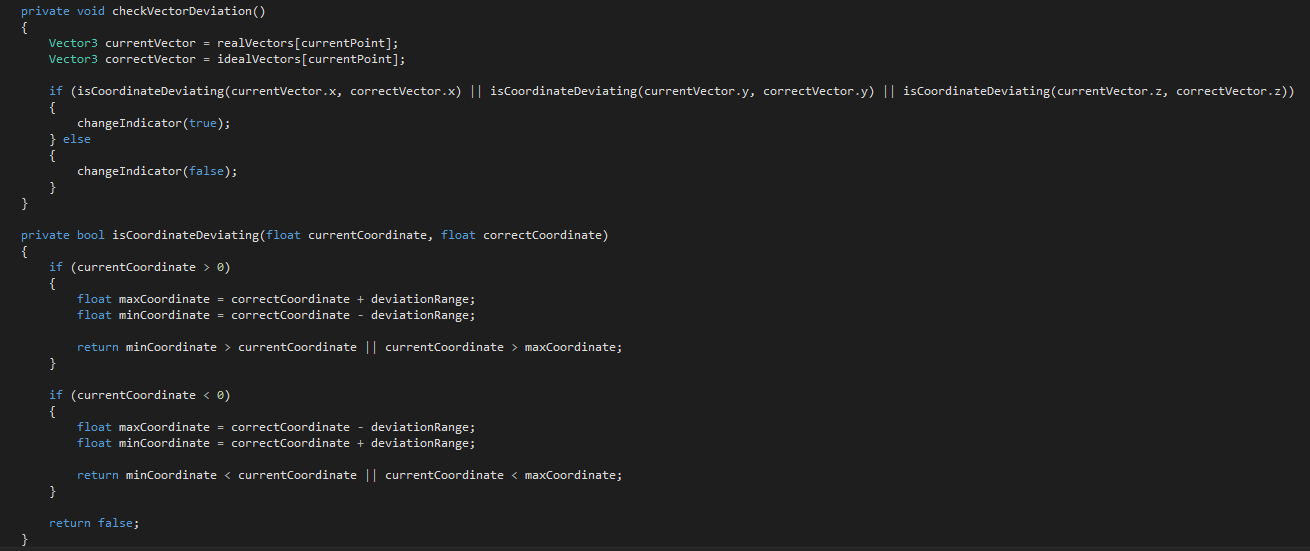
Tačiau atskyrus šias dvi trajektorijas pasimato nauja problema – idealios trajektorijos tunelis piešiamas naudojant Gizmos objektus, kurie matomi tik programavimo rėžime ir yra pirminiai, t.y. lėktuvas, būdamas išorėje tunelio atrodo, lyg būtų po tuneliu, nes Gizmos objektai uždengia lėktuvą (2 pav.). Todėl šis tunelio piešimo būdas naudojant Gizmos yra netinkamas.

8 pav. Neaiški lėktuvo buvimo vieta, idealios trajektorijos atžvilgiu

Šiai problemai išspręsti, buvo sugalvota, vietoje Gizmos objektų naudoti realius žaidimo objektus. Tam prireikė sukurti naują objektą – 3D apskritimą. Unity standartinių žaidimo objektų bibliotekoje tokio objekto nėra, todėl jį tenka pasigaminti. Unity bendruomenės puslapyje yra patalpintas būtent tokio objekto sukūrimo skriptas[[1]](#footnote-1). Šio skripto pagalba, sukuriamas, vadinamasis, Torus objektas. Tačiau šis objektas yra tik nematomi kontūrai mums reikalingo apskritimo. Pasinaudodami Unity įrankiais, apvelkame kontūrus matoma medžiaga ir gauname tinkamą objektą idealios trajektorijos tuneliui formuoti. Suformavus tunelį, lėktuvo pozicija idealios trajektorijos atžvilgiu tapo aiškiai matoma (3 pav.).

9 pav. Aiškiai matoma lėktuvo pozicija, idealios trajektorijos tunelio atžvilgiu

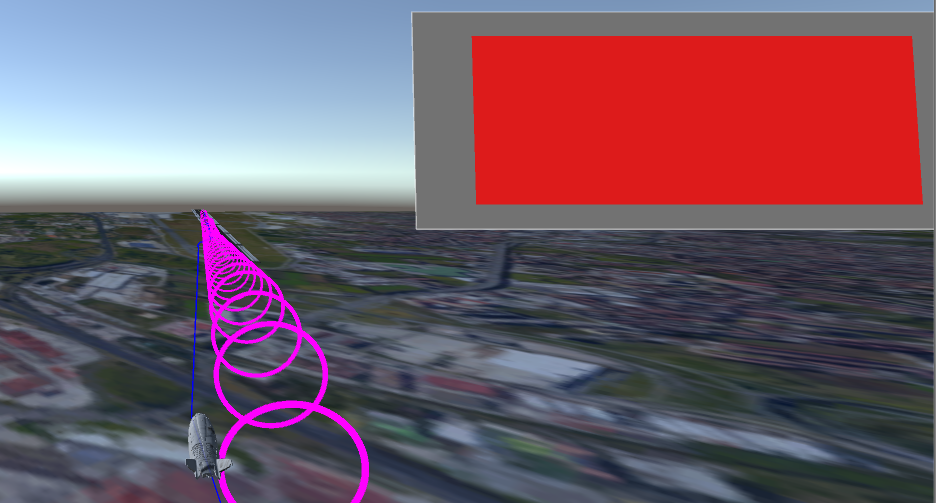
# Trajektorijos rizikos valdymas

Besileidžiantis ar kylantis lėktuvas turi judėti pagal jam paskirtą procedūrą. Procedūroje yra pateikiama trajektorijos nustatymų lentelė, kurioje atvaizduojami atstumai iki tūpimo ar kilimo taško, bei kokiame aukštyje tam tikru atstumu iki tako pradžios ar pabaigos turi būti lėktuvas. Šie duomenys yra pateikiami idealiu nusileidimo ar kilimo atveju, tačiau jiems gali būti taikomos paklaidos, t.y. lėktuvo reali buvimo vieta gali skirtis nuo procedūroje nurodytos per tam tikrą toleruojamos rizikos paklaidą. Trajektorijų rizikos valdymui buvo sukonstruotas koordinačių paklaidos matavimo metodas (10 pav.). 

10 pav. Koordinačių paklaidos matavimo metodas

Šiam metodui paduodama dinaminė reikšmė – koordinatės paklaida. Šią paklaidą pridėjus bei atėmus iš idealios trajektorijos koordinatės gauname rėžius, kuriuose privalo atsidurti realios trajektorijos koordinatė konkrečiu skrydžio momentu. Įteruojant per realaus skrydžio koordinates, kiekvienoje iteracijoje iškviečiamas šis metodas, kuris atlieka paprastą patikrinimą ar reali koordinatė priklauso šiems rėžiams ar ne. Metodas grąžina reikšmę, kuri padeda nuspręsti ar lėktuvo skrydis vykdomas kaip numatyta. Tačiau problema atsiranda tada, kai reikia apskaičiuoti koordinatės paklaidą konkrečiam trajektorijos taškui, nes paklaidos dydis skiriasi idealios trajektorijos pradžioje ir pabaigoje. Tai reiškia, kad reikia žinoti, kada ir kokią paklaidą perduoti metodui. Taip pat, kiekviena koordinačių ašis turi savo paklaidos dydi, todėl atliekami skaičiavimai tampa dar sudėtingesni. Šiai situacijai palengvinti buvo nutarta skrydžio trajektoriją padalinti į kelis etapus, kur kiekvienas iš jų turėtų savo rizikos reikšmes. Analizuojamoje procedūroje pateikiama detali lėktuvo judėjimo duomenų lentelė (5 pav.), kurioje pateikiami 5 DST etapai, nurodantys kokiame aukštyje turi būti lėktuvas tam tikrame taške. (Tolimesni darbai – suskaidyti trajektoriją, kiekvienam etapui pateikt būtent jam skirtą paklaidą, tuomet, nustatyti maksimaliai leidžiamo nukrypimo nuo trajektorijos atstumą (2 laipsniai kampas), kuris bus mūsų piešiamo tunelio spindulys R. Šis dydis, nebūtinai turi būti konstanta, galima padaryti reguliuojamą tunelio spindulį kiekvienam etapui.)

# Nukrypimo nuo idealios trajektorijos atvaizdavimas

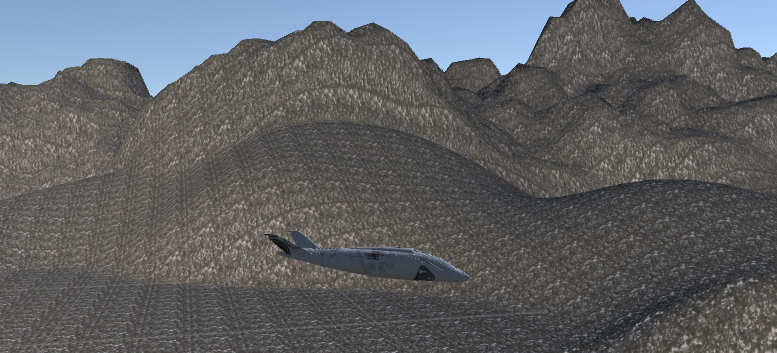
Pagrindinė sistemos savybė, kurios siekiama kuriant šį protipą, yra tai, kad lėktuvo nusileidimo ar pakilimo būsenai nustatyti pakaktų trumpo žvilgtelėjimo, todėl nukrypimo nuo trajektorijos atvaizdavimui sukuriamas vartotojo sąsajos dialogas su spalviniu indikatoriumi (10 pav.). Pasinaudojant anksčiau minėtu metodu, kuris grąžina požymį apie lėktuvo nukrypimą nuo trajektorijos, keičiame šio indikatoriaus spalvą. Esant nukrypimui užsidega raudona spalva, priešingu atveju dega žalia. Taip skrydžio vadovui nereikia didelių pastangų, kad suprasti ar skrydis vykdomas teisingai. Jeigu skrydis vykdomas tisingai ir indikatoriau spalva yra žalia, jokių papildomų veiksmų imtis nereikia. Tačiau jeigu atsiranda nukrypimas ir indikatorius paraudonuoja, skrydžio vadovui gali prireikti nustatyti, kurią idealios trajektorijos dalį lėktuvas pažeidžia. Todėl atsiranda poreikis pakoreguoti indikatoriaus dialogą taip, kad neapkrauti pertekilne informacija, bet suteikti galimybę nustatyti nukrypimą. Kadangi trajektorijos nukrypimo apskaičiavimo metodas tikrina kiekvieną koordinatę, galima sukurti tris atskirus indikatoriaus tipo mygtukus kiekvienai koordinačių ašiai x, y ir z. Pasirinkus bet kurį iš jų, kamera perjungiama į būtent tą ašį geriausiai matomą. Tokiu būdu, skrydžio vadovui žvilgtelėjus ir pastebėjus bent vieną raudoną indikatorių, galima jį paspausti ir pamatyti detalesnį nukrypimo vaizdą.

11 pav. Vartotojo sąsajos dialogas su spalviniu indikatoriumi

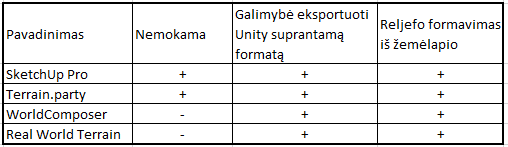
# Kameros

Pagrindiniam lėktuvo objektui sukuriamos papildomos, kito vaizdo perspektyvos kameros, kurias būtų galima keisti skrydžio metu norint matyti kitas skrydžio detales. Kamerų keitimo funkcionalumas pasiekiamas paspaudus klaviatūros klavišą „C“, kadangi didžioji dauguma žaidimų ir kitų simuliatorių naudoja būtent šį klavišą kameros keitimui. Šis funkcionalumas leidžia paprastu būdų pakeisti kamerą. Paspaudus klavišą, iškviečiamas kameros keitimo metodas, kuris pagal eilę, deaktyvuoja esamą kamerą bei aktyvuoja sekančią. Tačiau, ankščiau minėtam funkcionalumui, kuomet kamera perjungiama į konkrečia, paspaudus ant rizikos indikatoriaus, reikia metodo, kuris priimtų indeksą ir įjungtu konkrečią kamerą. Iš esmės, kamerą būtų galimą įjungti automatiškai, jeigu nors viena koordinatė neatitiktų idealios trajektorijos. Tačiau tokiu atveju atsiranda problema, kad vienu metu rizika gali atsirasti keliose koordinatėse, arba greitai keistis iš patenkančios į tenkinančius rėžius ir nepatenkačios, tuomet nėra aišku kurią kamerą reikia įjungti, o greitas junginėjimas tarp kelių kamerų blaškytų skrydžio vadovo dėmesį ir jam taptu neaišku kas vyksta su šiuo skrydžiu. Todėl kameros perjungimui nuspręsta naudoti indikatorius kaip mygtukus.

# Realaus reljefo pritaikymas terrain objektui

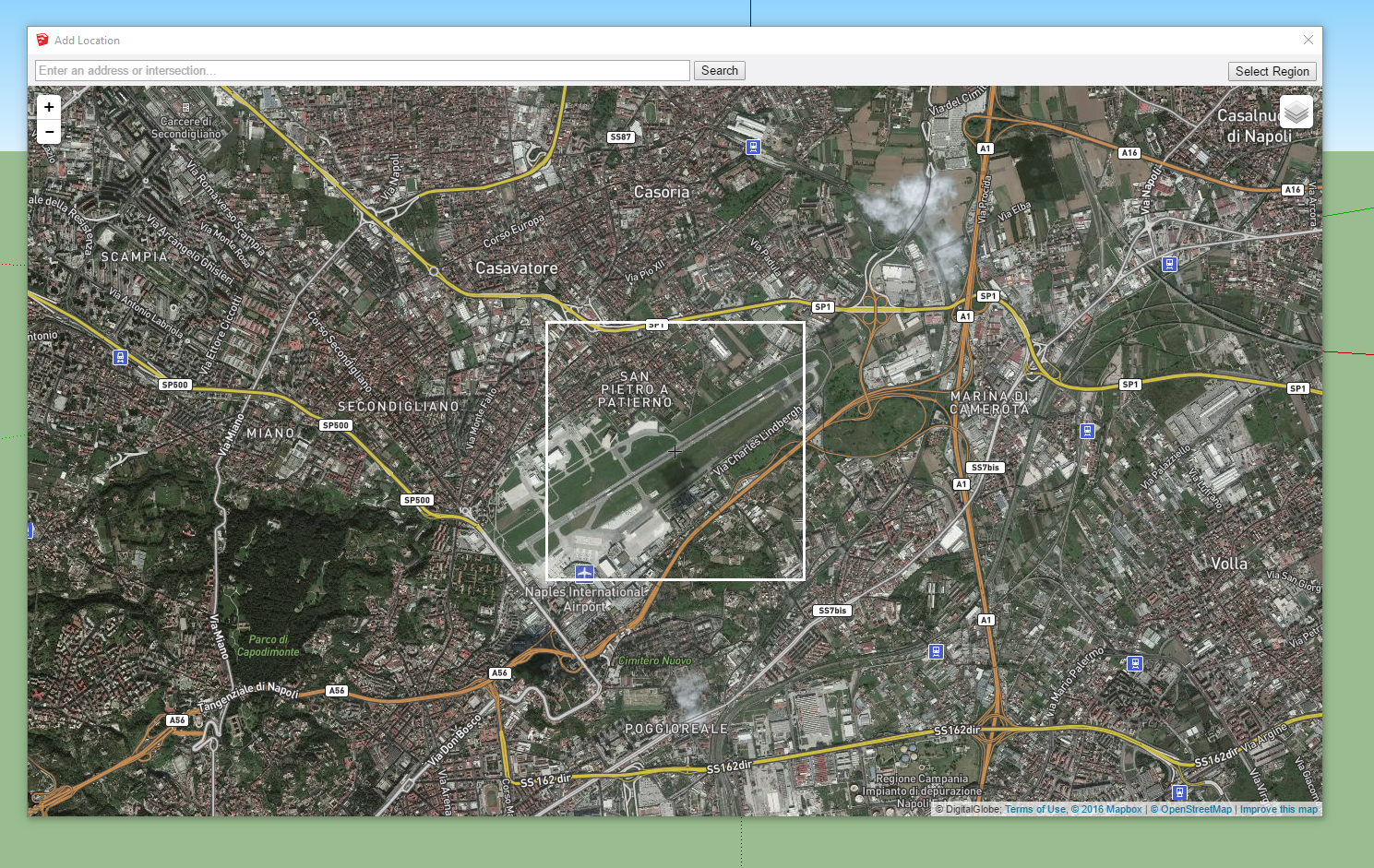
Žemės objektui pritaikomas vienas iš bazinių Unity medžiagų apipavidalinimų, bei suformuojami kalnai. Unity programa turi savo elektroninę parduotuvę, kurioje galima rasti mokamų ir nemokamų žaidimo kūrimo paketų. Šiuos paketus sudaro įvairūs apipavidalinimo, fizikos, valdymo ir kiti modeliai. Tarp jų buvo rastas ir lėktuvo objekto apipavidalinimo modelis, kuris buvo pritaikytas sukurtam rutuliui tam, kad vaizdas atrodytų realistiškiau (9 pav.), o judantis objektas turėtų aiškiai matomą kryptį.

12 pav. Sukurti lėktuvo ir žemės objektai

Kadangi prototipo veikimas yra grindžiamas atpažinimu, vien atsitiktiniu būdu suformuotų kalnų neužtenka. Tam, kad suprasti konkrečią lėktuvo buvimo vietą, reikia Terrain objektą padaryti kuo panašesnį į realaus nusileidimo tako teritorijos reljefą. Kad tai padaryti, reikia išgauti oro uosto teritorijos reljefo duomenis, tokiame formate, kurį Unity mokėtų suprasti ir gebėtų pritaikyti Terrain objektui. Yra nemažai įrankių, kurie sugeba eksportuoti duomenis Unity suprantamam formate. Tam, kad išsirinkti tinkamą buvo sudaryta kriterijų lentelė (12 pav.).

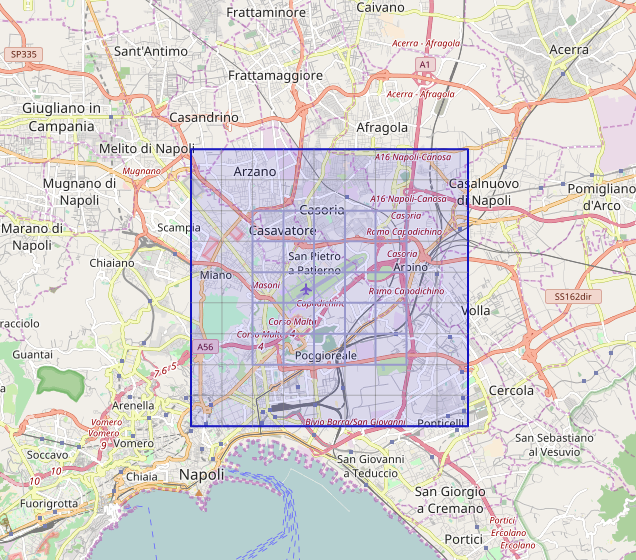
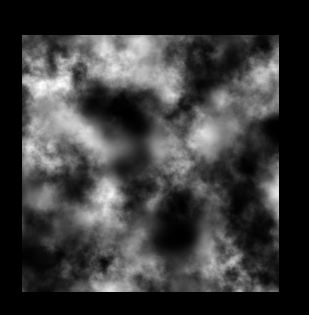
13 pav. Reljefo formavimo įrankio kriterijų lentelė

Pirmasis atitikęs kriterijus– „SketchUp Pro“. Šis įrankis suteikia galimybę suformuoti reljefą pagal tikrą žemėlapį tiesiog įvedant adresą arba rankiniu būdu pasirenkant vietą. Tačiau pagrindinė problema yra ta, kad pasirenkamos vietos maksimalus plotas yra labai nedidelis. Šiuo atveju, pasirenkamas plotas (11 pav.) net neapima viso Neapolio oro uosto. Šis įrankis leidžia atlikti tą patį veiksmą pakartotinai, taip išgaunant vis didesnį žemės plotą, tačiau eksportuojant tokį suklijuotą reljefą, Unity jį nuskaito kaip kelis objektus, o tam, kad ant Terrain objekto užmauti tokį reljefą, būtinas vientisas objektas. Todėl šis įrankis netinka.



14 pav. SketchUp Pro reljefo pasirinkimo dialogas

Kitas, taip pat visus kriterijus atitikęs, labai panašią funkciją turintis įrankis yra terrain.party. Terrain.party – tai virtualus įrankis pasiekiamas kaip internetinis puslapis, kuris yra skirtas būtent Terrain objektų kūrimui. Šio įrankio eksportuojami duomenys yra pagal žemės kalvas ir daubas (12 pav.) suformuojama aukščių nuotrauka, kompiuterinėje grafikoje kitaip dar vadinama – Heightmap[[2]](#footnote-2) (13 pav.). Tam, kad šią nuotrauką Unity galėtų pritaikyti Terrain objektui yra būtina įdiegti papildomą Unity funkciją „HeightmapFromTexture“ kurią galima rasti ankščiau minėtame Unity bendruomenės puslapyje. Pritaikius nuotrauką Terrain objektui, visi kalnai pradingo ir kadangi Neapolio oro uostas yra ganetinai lygaus reljefo žemės plote, gautas rezultatas atrodo kaip naujas lygus Terrain. (Neaišku ar taip ir paliksiu)



15 pav. Terrain.party reljefo pasirinkimas

16 pav. Heightmap pavizdys

# Rezultatai ir išvados

Šio darbo metu buvo sukurtas atpažinimu grįstas sistemos prototipas, kuris leidžia skrydžių vadovui ekrane stebėti lėktuvo nusileidimo ir pakilimo trajektorijas. Šiam tikslui pasiekti buvo įvykdyti tokie etapai:

1. Išanalizuota skrydžio procedūros duomenų lentelė ir gauta informacija panaudota trajektorijos rizikos paklaidų nustatymui.
2. Naudojant ankstesnio darbo metodus buvo integruota idealioji trajektorija. Iškilusi tunelio vizualizavimo problema buvo išspręsta sukūrus naujus tunelio objektus – Torus.
3. Nukrypimams nuo trajektorijos pažymėti sukurtas vartotojo sąsajos dialogas su spalviniu indikatoriumi.
4. Detalesniam vaizdui pateikti prijungtas kamerų valdymo funkcionalumas.
5. Realaus reljefo suformavimui išanalizuoti keli įrankiai: SketchUp Pro ir Terrain.party. Pagal nustatytus kriterijus pasirinktas labiau tinkantis Terrain.party. Šio įrankio pagalba suformuotas reljefas pritaikytas Terrain objektui.

# Šaltiniai

1. Oficialus „Unity“ puslapis: <https://unity3d.com>
2. Kristina Lapin, Vytautas Čyras, Laura Savičienė. Visualization of Aircraft Approach and Departure Procedures in a Decision Support System for Controllers. In: Barzdins, J., Kirikova, M. (eds.) Databases and Information Systems VI, Selected Papers from the Ninth International Baltic Conference, DB&IS 2010, Vol. 224 Frontiers in Artificial Intelligence and Applications, ISBN 978-1-60750-687-4, IOS Press Amsterdam, 2011, pp. 408-421.
3. Projekto „Sky-Scanner“ duomenų šaltinis: <http://www.mif.vu.lt/~moroz/SKY.html>
4. <https://en.wikipedia.org/wiki/Heightmap>
5. Unity papildymų internetinė parduotuvė <https://www.assetstore.unity3d.com/en/>

1. <http://wiki.unity3d.com/index.php?title=CreateTorus> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Heightmap> [↑](#footnote-ref-2)