VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Lėktuvo trajektorijos 3D vizualizavimas pasirinktam oro uostui: Vilniaus atvejis

3D Visualization of Aircraft trajectory for a Specific Airport: Vilnius Case

Bakalauro baigiamasis darbas

Atliko:	4 kurso 3 grupės studentas Kajus Naujokaitis	(parašas)
Darbo vadovas:	doc. dr. Vytautas Čyras	(parašas)
Darbo recenzentas:	doc. dr. Kristina Lapin	(parašas)

Vilnius, 2021

Santrauka

Šio darbo tikslas – pademonstruoti, kaip galima sukurti patobulintą lėktuvo tūpimo ir pakilimo trajektorijos trimatę vizualizaciją, kurią galimą pritaikyti bet kokiam oro uostui. Potencialūs programų sistemos naudotojai būtų skrydžių vadovai. Remiantis įgytomis avionikos ir programų sistemų kūrimo žiniomis modifikuojamas 2020 m. sukurtas prototipas, kuris vaizduoja lėktuvų poziciją ir trajektoriją tūpimo bei pakilimo metu trimatėje erdvėje Neapolio „Capodichino“ oro uoste. Šio prototipo pagrindu kuriamas naujas sprendimas, pritaikytas Vilniaus oro uostui ir kitiems, siekiant kuo lengvesnio pakartotinio panaudojimo ir modifikavimo funkcionalumo. Naujas prototipas yra lengviau keičiamas ir pritaikomas kitiems oro uostams ir jų specifikoms.

Kuriant ir modifikuojant prototipą nagrinėjamos avionikoje naudojamos instrumentinės skrydžių taisyklės, kuriomis vadovaujasi skrydžių vadovai tikrindami, ar lėktuvas nepažeidžia apibrėžtų tūpimo ar kilimo procedūrų. Remiantis 2020 m. vizualizacija yra sukurtas sistemos prototipas, kuris, pagal iš anksto duotas koordinates trimatėje erdvėje, vaizduoja lėktuvų trajektorijas ir suteikia galimybę naudotojui įvertinti jų atitikimą pasirinktai procedūrai Vilniaus oro uoste. Šis prototipas, lyginant su sukurtu 2020 m., yra lengviau modifikuojamas ir pritaikomas kitoms vietovėms bei oro uostams.

Raktiniai žodžiai: lėktuvas, trajektorija, trajektorijos vaizdavimas, trajektorijos vizualizacija, trajektorijos tikrinimas, moduliarumas, interpoliavimas, vizualizacija, Unity, Vilniaus oro uostas.

Summary

The aim of this work is to demonstrate how an improved three-dimensional visualization of aircraft landing and take-off trajectory, which can be applied to any airport, can be developed. Potential users of the system would be air traffic controllers. Based on the knowledge gained in the field of avionics and development of software systems, a prototype was created in 2020 that displays the position and movement of an aircraft during take-off and landing in three-dimensional space at Naples Capodichino Airport. On the basis of this prototype, a new solution is developed, adapted to Vilnius Airport and any others, ensuring the easiest possible reuse and modification functionality. The new prototype is easier to modify and adapt to other airports and their specifications.

Instrument landing rules, followed by air traffic controllers to verify that an airplane is flying in accordance with defined landing or take-off procedures, are researched during development and modification of the prototype. Based on the visualization created in 2020, a new prototype has been developed, which, according to the pre-given coordinates in three-dimensional space, depicts the trajectories of aircraft and allows the user to assess compliance with the selected flight procedure at Vilnius Airport. This prototype is easier to modify and adapt to other locations and airports compared to the one created in 2020.

Keywords: airplane, trajectory, trajectory mapping, trajectory visualization, trajectory checking, modularity, interpolation, visualization, Unity, Vilnius airport.

TURINYS

ĮVADAS

Tyrimo objektas – tai programų sistema, skirta vizualizuoti lėktuvo tūpimo ir kilimo trajektorijų atitikimą norminėms procedūroms. Siekiama sistemą iliustruoti Vilniaus oro uosto atveju, jos potencialūs naudotojai būtų skrydžių vadovai.

2020 m. sukurtas prototipas gali padėti potencialiems skrydžių vadovams teisingai ir efektyviai valdyti orlaivių eismą, tačiau vizualizacija nėra universali, pritaikyta tik Neapolio „Capodichino“ oro eismo valdymui. Tokio tipo sistemos projektavimas nėra optimalus ir yra vengiamas realiuose programinės įrangos kūrimo projektuose. 2020 m. prototipas stokoja programų sistemų projektavimo principų, susijusių su moduliškumu, išplėtimu ir kintamumu. Šio prototipo pritaikymas kitam oro uostui ir jame vykdomiems skrydžiams reikalauja daug darbo, galimai net programinio kodo keitimo. Šiame darbe siekiama ankstesnio prototipo pagrindu sukurti naują vizualizaciją, kuri yra lengvai keičiama ir pritaikoma kitiems oro uostams ir jų specifikoms. Siekiama sukurti programų sistemą, kuri atitiktų potencialių vartotojų – skrydžių vadovų – poreikius ir galėtų būti naudojama tarptautiniu mastu.

Nagrinėjama problema. Programų sistemos pritaikymas keičiant vieną oro uostą kitu. Įprasta, kad prototipas kuriamas konkrečiam oro uostui. Sistemos modifikavimas kitam oro uostui nėra toks paprastas, kaip keleto parametrų pakeitimas. Keičiasi, pavyzdžiui, reljefas, žemėlapis, oro uosto geografija, nusileidimo tako orientacija bei topologija ir draudžiamų skristi zonų virš miesto išsidėstymas.

Aktualumas. 2010–2012 m. vykdytame Europos Sąjungos 6-osios Bendrosios programos projekte SKY-Scanner buvo kuriama nauja programinė ir techninė įranga, skirta nustatyti tikslesnę poziciją tų lėktuvų, kurie yra oro eismo zonoje (angl. *air traffic zone,* toliau ATZ).

2020 m. prototipe ši sistema buvo tobulinama kuriant naują, intuityvesnę vizualizaciją, kuri palengvintų lėktuvų trajektorijų stebėjimą. Sistema buvo pritaikyta specifiniam oro uostui – Neapolio „Capodichino“ – ir jame naudojamoms tūpimo bei kilimo procedūroms. Šis sprendimas stokojo universalumo ir galimybės pritaikyti programinę įrangą kitiems oro uostams. Šį trūkumą siekiama išspręsti kuriant išplėstą prototipą.

Darbo tikslas

Programų sistemos, skirtos trajektorijų vizualizavimui, modifikavimo problemos tyrimas ir prototipo sukūrimas. Naudojama Unity (Unity3D) žaidimų kūrimo mašina. Siekiama ištirti programų sistemos pritaikymą Vilniaus oro uosto atveju ir šitaip pademonstruoti modifikavimo procesą.

Darbo uždaviniai

1. Programų sistemos modifikavimo problemos analizė. Ištirti SKY-Scanner projekte ir jo pasekoje kurtus prototipus, įvertinti jų lankstumą bei nustatyti, kurie moduliai gali būti pakartotinai naudojami.
2. Surinkti darbui aktualią informaciją apie orlaivių tūpimo bei kilimo procedūras Vilniaus oro uoste ir ištirti ją.
3. Modifikuoti ankstesnį prototipą siekiant pritaikyti gautą informaciją Vilniaus oro uosto atveju. Sumodeliuoti Vilniaus oro uosto aplinką (žemėlapį, radaro poziciją, nusileidimo taką) Unity įrankiais.
4. Prototipo sukūrimas.

Tyrimo metodas

Literatūros apžvalga, anksčiau kurtų prototipų analizė, informacijos apie Vilniaus oro uostą rinkimas ir analizė. Oro uosto vietovė modeliuojama Unity aplinkoje, tobulinamas programų sistemos prototipas. Prototipo demonstravimas kaip eksperimentas.

Darbo atlikimo procesas

1. Apžvelgiama darbui aktuali literatūra ir kita informacija, pvz., tūpimo ir kilimo procedūros. Tiriama teorija apie programų sistemų modifikavimą.
2. Apžvelgiami anksčiau kurti prototipai ir įvertinamos modifikuotinos dalys. Siekiama išsiaiškinti, kurie prototipo moduliai gali būti pakartotinai naudojami arba turi būti keičiami, siekiant pritaikyti prototipą Vilniaus oro uostui.
3. Modeliuojama Vilniaus vietovė, nusileidimo takas ir radaras trimatėje erdvėje Unity įrankiais. Siekiama sukurti kuo detalesnį, tačiau lengvai suprantamą ir neperkrautą vietovės vaizdą.
4. Kuriamas prototipas – keičiami ir kuriami nauji moduliai, skirti duomenų valdymui, trajektorijos vizualizavimui bei atitikimui norminėms procedūroms pavaizduoti.
5. Prototipas demonstruojamas.

Literatūros šaltiniai ir kita informacija

Darbo metu nagrinėjami šaltiniai, kurie pateikia informaciją apie programų sistemų modifikavimą, skrydžių valdymą, trajektorijų vizualizavimą įvairiuose tyrimuose, Vilniaus oro uosto tūpimo ir kilimo procedūrų specifiką. Naudojama literatūra ir šaltiniai apie palydovinių nuotraukų, reljefo žemėlapių naudojimą kuriant trimatį vietovės modelį. Taip pat pasitelkiami šaltiniai, kuriuose pateikiama informacija apie Unity įrankių naudojimą.

1. AVIONIKOS DALYKINĖ SRITIS

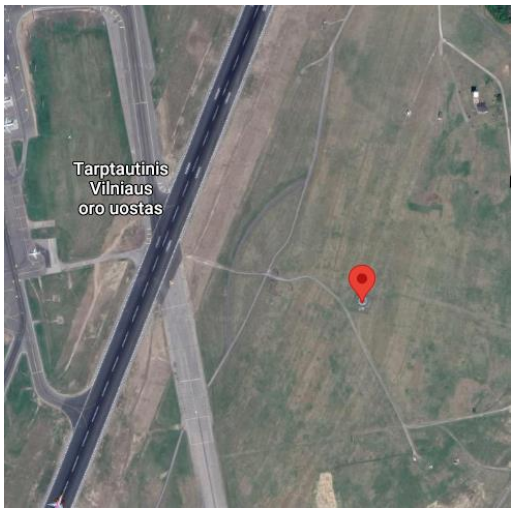
Darbe nagrinėjamas Vilniaus oro uostas, jame esanti lėktuvų tūpimui ir kilimui naudojama įranga ir instrumentinės skrydžių taisyklės, skirtos pilotams laikytis tūpimo ar pakilimo metu. Taip pat nagrinėjami Neapolio Capodichino oro uoste anksčiau vykdytų skrydžių duomenys, kuriuos surinko Vilniaus universiteto dėstytojai ir studentai, anksčiau dirbę prie panašaus projekto. Šie duomenys yra pakankamai tikslūs, kad būtų galima vizualizuoti tikslias skrydžių trajektorijas. Verta paminėti, kad realioje skrydžių valdymo sistemoje tokie duomenys būtų gaunami iš radarų programos vykdymo metu, tačiau nuspręsta, kad prototipe trajektorijoms vaizduoti pakanka iš anksto surinktų skrydžių duomenų. Skrydžiai simuliuojami naudojant duomenyse esančias koordinates.

1.1. Oro uostas

Lėktuvo trajektorijos 3D vizualizacija kuriama modifikuojant 2020 m. sukurtą vizualizacijos prototipą pritaikant jį Lietuvos Vilniaus oro uostui. Vilniaus oro uoste pilotai ir skrydžių vadovai naudojasi iš anksto nustatytomis tūpimo ir kilimo procedūromis. Pagal šias skrydžių vadovai tikrina lėktuvo trajektorijos atitikimą norminėms. Vilniaus oro uoste naudojama ši įranga, kuri aktuali prototipo įgyvendinimui:

1. Radijo navigacijos stotis VOR/DME „VNO“. Stotį sudaro du radijo švyturiai: VOR (angl. *very high frequency omni-directional (radio) range*) ir DME (angl. *distance measuring equipment*) [CNA09]. VOR skirtas nustatyti kampą tarp stoties ir lėktuvo imtuvo. DME skirtas atstumo tarp orlaivio ir stoties nustatyti. Šiais įrenginiais nustatomi navigaciniai taškai (angl. *navigational fix*). Tarptautiniuose avionikos žemėlapiuose VOR/DME stotis žymima simboliu  . Tikslios Vilniaus oro uosto VOR/DME „VNO“ stoties geografinės koordinatės nėra lengvai prieinamos, tačiau, naudojant Google Maps įrankį, galima nustatyti gana tikslią radaro poziciją pagal palydovines nuotraukas. Prototipo kūrimui bus naudojamos šios VOR/DME radijo navigacijos stoties „VNO“ koordinatės, gautos pagal Google Maps informaciją (žr. 1 pav.): 54,63609°, 25,29362° (laipsniai šiaurės platumos ir rytų ilgumos). Nuspręsta, kad kuriamame prototipe užtenka pasirinkto koordinačių tikslumo, kuris atitinka apie 1 m. tikslumą realybėje.
2. ILS (angl. *instrument landing system*) – tai sistema, kuri orlaiviui suteikia horizantalias ir vertikalias gaires visai prieš tūpimą bei tūpimo metu. ILS taip pat tam tikruose fiksuotuose taškuose nurodo atstumą tarp orlaivio ir nustatyto nusileidimo taško oro uoste. Šio, kaip ir VOR/DME radaro, koordinatės nėra viešinamos, tačiau prototipui kurti pasirinkta naudoti nusileidimo tako geometrinį centrą kaip nustatytą orlaivio nusileidimo tašką. Šio taško

Vilniaus oro uoste koordinatės: 54,63420° šiaurės platumoje, 25,28585° rytų ilgumoje. Pilotams ILS rodmenys, t. y. nuokrypis nuo trajektorijos, kuria privaloma skristi, pateikiami lėktuve esančioje įrangoje, kurios pagalba pilotai atitinkamai koreguoja orlaivio trajektoriją, žr. 2 pav.



1 pav. Google Maps palydovinėse nuotraukose matoma VOR/DME „VNO“ radaro pozicija

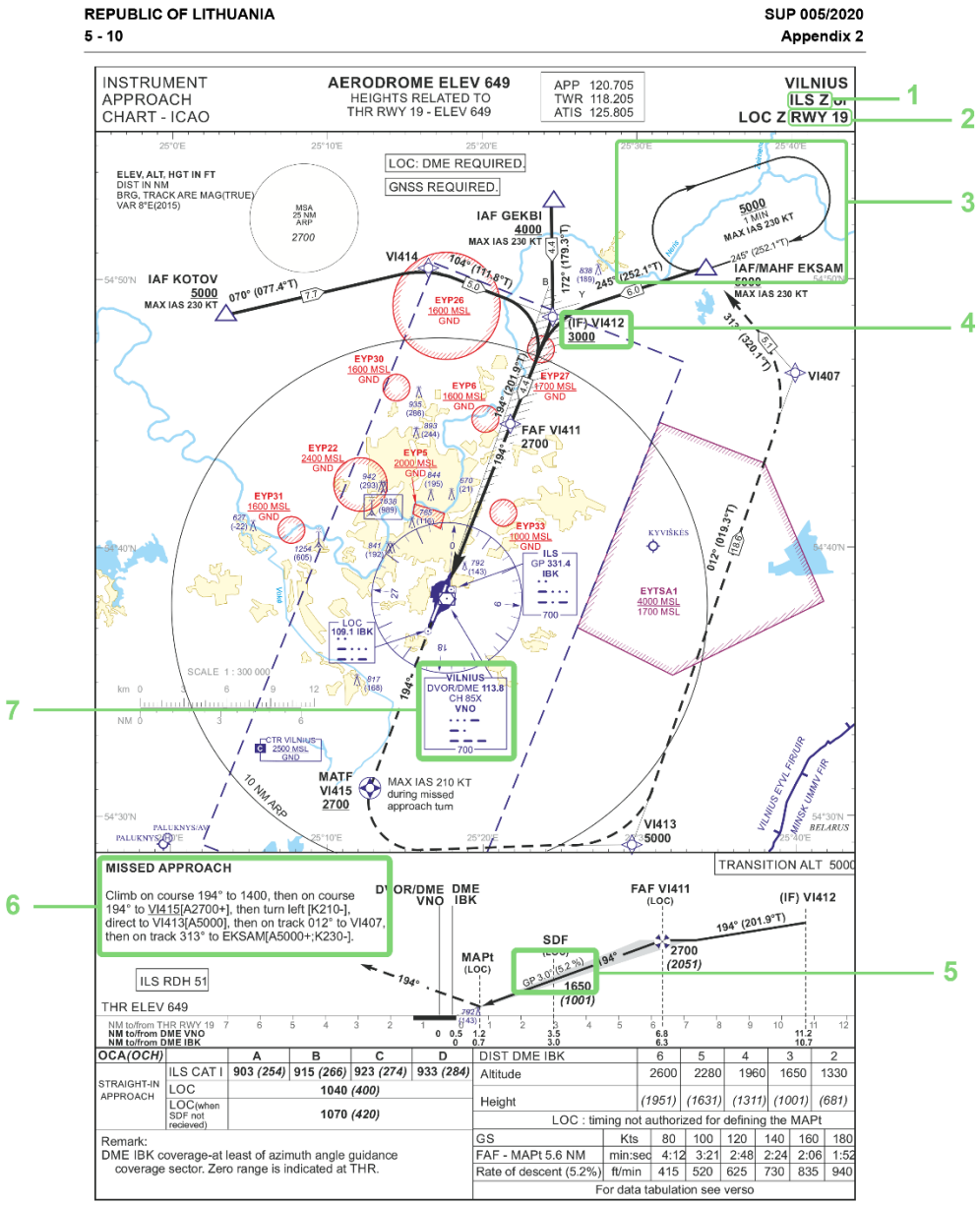
2 pav. ILS sistemos rodmenys, kuriuos mato pilotas

Renkant geografinę oro uosto informaciją taip pat pasirinkta prototipui reikalingos vietovės vaizdavimui naudoti 2500 km^2 kvadrato formos plotą aplink oro uosto pakilimo tako centrą. Šis vietovės plotas pakankamas pavaizduoti daugeliui orlaivių nusileidimo ar pakilimo trajektorijų. Šiam plotui nustatyti taip pat teko apskaičiuoti koordinates. Apskaičiuota pagal Google Maps palydovinių nuotraukų koordinačių duomenis, pasirenkant 4 taškus apie 25 km atstumu nuo oro uosto nusileidimo taško centro šiaurėje, pietuose, rytuose ir vakaruose. Atstumui apskaičiuoti naudojamas „latlongdata“[1] įrankis. Šiuo įrankiu apskaičiuotos vietovės „kvadrato“ kraštinių koordinatės – 54,85903°, 54,40937°, 25,67430°, 24,89740° (šiaurės platuma kvadrato viršuje ir apačioje bei rytų ilguma kvadrato dešinėje ir kairėje). Visos rastos koordinatės bus naudojamos modeliuojant tikslų Vilniaus vietovės vaizdą.

[1] https://latlongdata.com/distance-calculator/

1.2. Tūpimo ir kilimo procedūros

Orlaivių kilimo ir tūpimo procedūros aviacijoje aprašomos instrumentinėmis skrydžių taisyklėmis (diagramomis) (angl. *instrument approach chart*). Remiantis jomis prototipe bus tikrinamas skrydžio trajektorijos atitikimas nustatytoms normoms. Šiame darbe pasirinkta nagrinėti Vilniaus oro uosto „ILS-Z“ tūpimo procedūra (žr. 3 pav.).



3 pav. Vilniaus oro uosto „ILS-Z“ tūpimo procedūros instrumentinės skrydžio taisyklės

Tūpimo procedūra vaizduojama žemėlapyje, kuriame nurodoma detali informacija apie tūpimo taisykles. Darbui aktuali informacija [FAA17]:

1. Tūpimo procedūros pavadinimas;
2. Lėktuvo artėjimo prie nusileidimo tako kryptis;
3. Laukimo zona;

4. Vienas iš taškų, kuriuos tūpimo metu privalo praskristi orlaivis;
5. Leidimosi kampas;
6. Nesėkmingo tūpimo nurodymai;
7. VOR/DME „VNO“ radijo navigacijos stotis ir informacija apie ją.

Tūpimo procedūros žemėlapyje matomas pagrindinis lėktuvo skridimo kursas laikantis instrumentinių skrydžio taisyklių. Šioje procedūroje nurodyti trys maršrutai, kuriais gali atskristi lėktuvai. Leidimosi Skrydžių vadovui nurodžius procedūros pavadinimą pilotas privalo vadovautis atitinkama procedūra. Jei oro uosto pakilimo takas užimtas, pilotas privalo laukti žemėlapyje pažymėtoje laukimo zonoje (angl. *entry holding*).

Darbe procedūros ribas pasirinkta vizualizuoti remiantis tūpimo trajektorija matoma 3 pav. nuo taško „IAF KOTOV“ iki nusileidimo tako. Pagal reikalavimus, lėktuvas, atskridęs iki VI412 taško (3 pav. 4 nr.) ir skrendantis link VI411 taško, turi mažinti aukštį iki 2700 pėdų. Pasiekus VI411 tašką pradedamas vykdyti galutinis priartėjimas iki nutūpimo ant tako. Galutinio priartėjimo metu pilotas privalo laikytis nuolydžio nurodyto 3 pav. 5 nr. Vis dėlto, nėra realu viso nusileidimo metu laikytis tikslaus 5,2% nuolydžio kampo, tad pilotams leidžiama šį kampą pažeisti ±0,5%. Šiame darbe bus naudojamos tokios pačios leidžiamų pažeidimų ribos.

Taške MAPt (angl. MAPt – *Missed Approach Point*) pilotas turi arba tupdyti lėktuvą arba tūpimą nutraukti vykdant sąlygas, aprašytas nesėkmingo tūpimo nurodymuose (angl. *Missed Approach*), jei nusprendžiama, pavyzdžiui, kad oro sąlygos nusileidimui netinkamos. Pagal nesėkmingo tūpimo nurodymus (3 pav. 6 nr.), lėktuvas turi pakilti iki 1400 pėdų aukščio skrendant 194° kryptimi. Pakilus iki 1400 pėdų aukščio kilti iki 2700 pėdų aukščio ir skristi link taško VI415. Praskridus VI415 pasukti į kairę ir kylant iki 5000 pėdų aukščio skristi tiesiai link taško VI413. Pasiekus šį tašką vėl vykdomas posūkis į kairę ir skrendama 194° kryptimi link taško VI407, nuo kurio vėl sukama į kairę ir skrendama 313° kryptimi iki taško EKSAM. Pasiekus šį tašką pilotas privalo laikytis nežemesnio kaip 5000 pėdų aukščio ir laukti tolimesnių nurodymų laukimo zonoje (3 pav. 3 nr.).

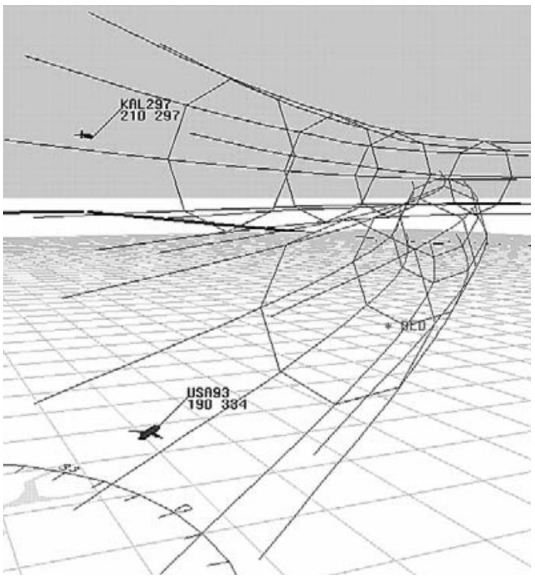
1.3. Trajektorijos vizualizavimas

Šiuo metu skrydžių vadovų naudojamos sistemos pateikia tik dvimatį oro eismo vaizdą, kuriame rodoma horizontali orlaivių padėtis, tačiau aukštis bei greitis pavaizduoti tik skaičiais. Skrydžių vadovai, norėdami įsitikinti, kad lėktuvas laikosi tūpimo procedūros taisyklių, privalo šią informaciją apdoroti savarankiškai [Lap10]. 2020 m. prototipas siekia palengvinti šį darbą vaizduojant orlaivių trajektoriją trimatėje erdvėje [Sav09b]. Vizualizuojant trajektoriją trimatėje erdvėje yra lengviau suprantama, kokioje pozicijoje yra orlaivis bet kuriuo momentu bei nebereikia atlikti skaičiavimų mintyse, norint įvertinti galimus tolimesnius lėktuvo judesius.

Trajektoriją vizualizuoti galima įvairiai ir kiekvienas sprendimas turi savų privalumų bei trūkumų. Reikia atkreipti dėmesį į tai, kad vizualizacija privalo aiškiai vaizduoti lėktuvą ir jo poziciją erdvėje, lėktuvo trajektoriją ir trajektorijos atitikimą nustatytoms procedūros normoms. Darbe pasirinkta analizuoti tris trajektorijų vizualizavimo tipus, įvertinti jų privalumus bei trūkumus ir išsirinkti tinkamiausią variantą prototipui.

1.3.1. Trimatis trajektorijos vizualizavimas „tuneliu“

Visų pirma analizuojamas vienas iš populiariausių ir lengviausiai suprantamų trajektorijos vizualizacijų tipų – aštuonkampiai tuneliai erdvėje (žr. 4 pav.).



4 pav. Trimatis trajektorijos vaizdavimas naudojant aštuonkampius tunelius [Lap10]

Ši vizualizacija buvo pristatyta „EUROCONTROL Innovative Research Workshop & Exhibition“ parodoje 2010 m. Kristinos Lapin darbe „SKY-Scanner: time-critical decision support system surveilling aircraft landing and take-off“ [Lap10]. Čia minima, kad ši vizualizacija buvo naudojama Bostono Logano oro uosto terminalo zonoje konfliktams aptikti. Šis „grynas“ 3D sprendimas turi trūkumų, vienas iš jų – sudėtinga nustatyti orlaivio poziciją vietovės reljefo atžvilgiu. Taip pat reikia atkreipti dėmesį, jog nėra matoma visa orlaivio skrista trajektorija, t. y. ankstesnė pozicija erdvėje nėra išsaugoma. Vis dėlto, aštuonkampiai tuneliai padeda lengviau suvokti procedūros ribas ir orlaivio trajektorijos atitikimą joms, lyginant su dvimatėmis vizualizacijomis. Vietoje reljefo pateikiamas kvadratų tinklas, tinkamas matuoti atstumui, bet stokojantis kito funkcionalumo ir patogumų.

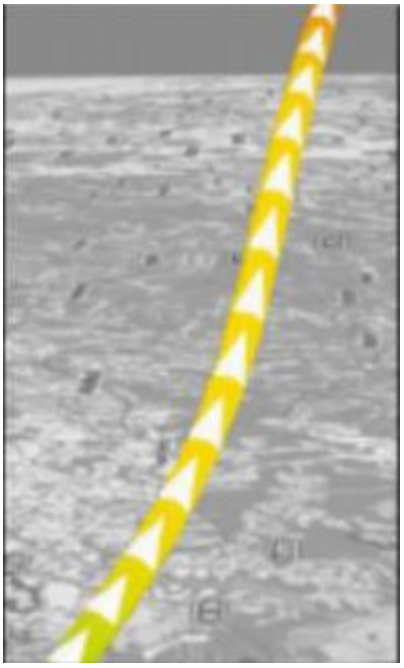
Išskiriami šie vizualizacijos privalumai, kurie bus įtraukiami į kuriamą prototipą:

- Aštuonkampiai tuneliai tinkamai pateikia oficialios procedūros ribas

- Orlaivio modelio dydis netrikdo vaizdo
- Nesudėtingai nustatoma aukštis ir kryptis erdvėje

1.3.2. Trimatis trajektorijos vizualizavimas „tuneliu“ su reljefu

Trajektorijos vizualizavimas tuneliu su reljefu (žr. 5 pav.) yra tarsi patobulinimas ar išplėtimas 4 pav. matomos vizualizacijos.



5 pav. Trajektorijos vaizdavimas lygiais tuneliais su reljefu [BTD15]

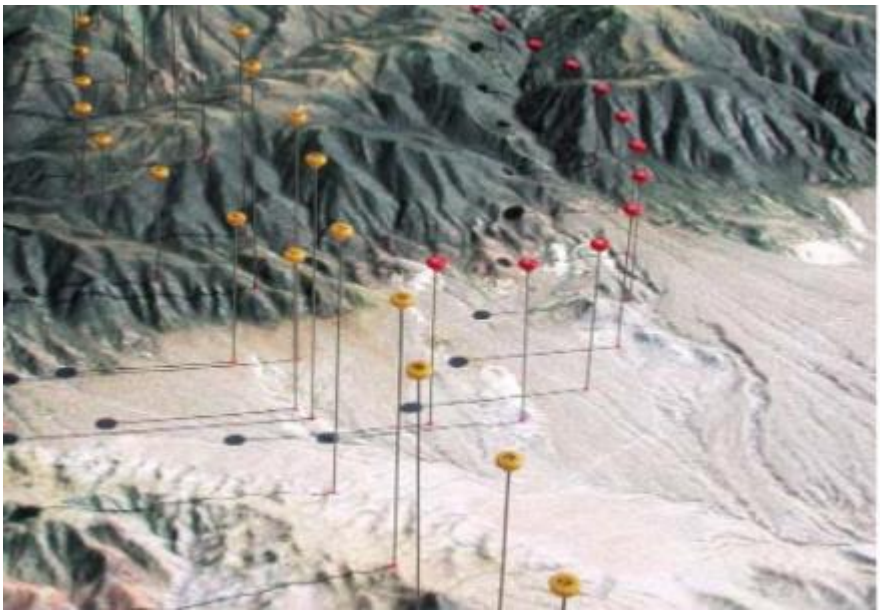
Visų pirma atkreipiamas dėmesys į tai, jog pateikiamas daug tikslesnis ir natūralesnis reljefo vaizdas. Nėra matomi visi reljefo ypatumai, tačiau patyrę skrydžių vadovai lengviau orientuotųsi erdvėje pagal tam tikras žinomas vietas reljefe, kaip miškai, kalnai ar kt. Taip pat svarbu, kad orlaivis erdvėje vaizduojamas kaip kūgis. Šis vaizdavimo būdas padeda geriau suprasti orlaivio kryptį ir poziciją erdvėje, ypač todėl, nes skrydžių vadovai neprivalo visą laiką matyti itin tikslų orlaivio modelį. Taip pat atkreipiamas dėmesys į tai, jog yra pateikiama visa orlaivio skrydžio trajektorija tunelio forma. Vis dėlto, yra pastebimas vienas esminis trūkumas, lyginant šia vizualizaciją su analizuota 1.3.1 skyriuje: nėra matomos jokios tūpimo ar pakilimo procedūros ribos. Šis aspektas yra itin svarbus skrydžių vadovų darbui.

Išskiriami šie vizualizacijos privalumai, kurie bus įtraukiami į kuriamą prototipą:

- Tikslesnis reljefo vaizdas padeda orientuotis erdvėje
- Aiškiai matoma visa orlaivio skrydžio trajektorija
- Kūgio formos orlaivio modelis yra lengviau interpretuojamas

1.3.3. Trimatis trajektorijos vizualizavimas su reljefu ir „tvora“

Paskutinis vizualizavimo būdas pateikiamas 6 pav.



6 pav. Trajektorijos vizualizavimas su reljefu ir „tvora“ [STM13]

Ši vizualizacija buvo sukurta norint prisiekusiesiems teisme parodyti lengvai suprantamą vaizdą vietovės, kur įvyko orlaivio avarija [STM13]. Veiksniai, kuriuos buvo norima aptarti ir analizuoti buvo lėktuvo padėtis prieš ir po variklio gedimo, oras, reljefas ir piloto veiksmų seka.

Vizualizacijoje akcentuojamas itin tikslus ir kuo paprasčiau interpretuojamas orlaivio aukščio pateikimas. Taip pat daug dėmesio skirta tiksliam reljefo vaizdui. Prisiekusieji, kuriems buvo pristatyta ši vizualizacija, teigė, kad tai padėjo jiems lengviau suprasti situaciją ir aptarti problemas [STM13]. Realus reljefo vaizdas suteikia galimybę nesudėtingai nustatyti orlaivio poziciją žemės atžvilgiu. Tuo tarpu pateikiama „tvora“ padeda greitai suprasti orlaivio aukštį bet kuriuo skrydžio momentu.

Šioje vizualizacijoje išskiriami šie privalumai, kurie bus įtraukiami į kuriamą prototipą

- Itin tikslus reljefo vaizdas padeda lengviau suprasti skrydžio aplinkybes
- „Tvora“ padeda nustatyti orlaivio aukštį bet kuriuo momentu

1.4. Pasirinktas vizualizavimo būdas

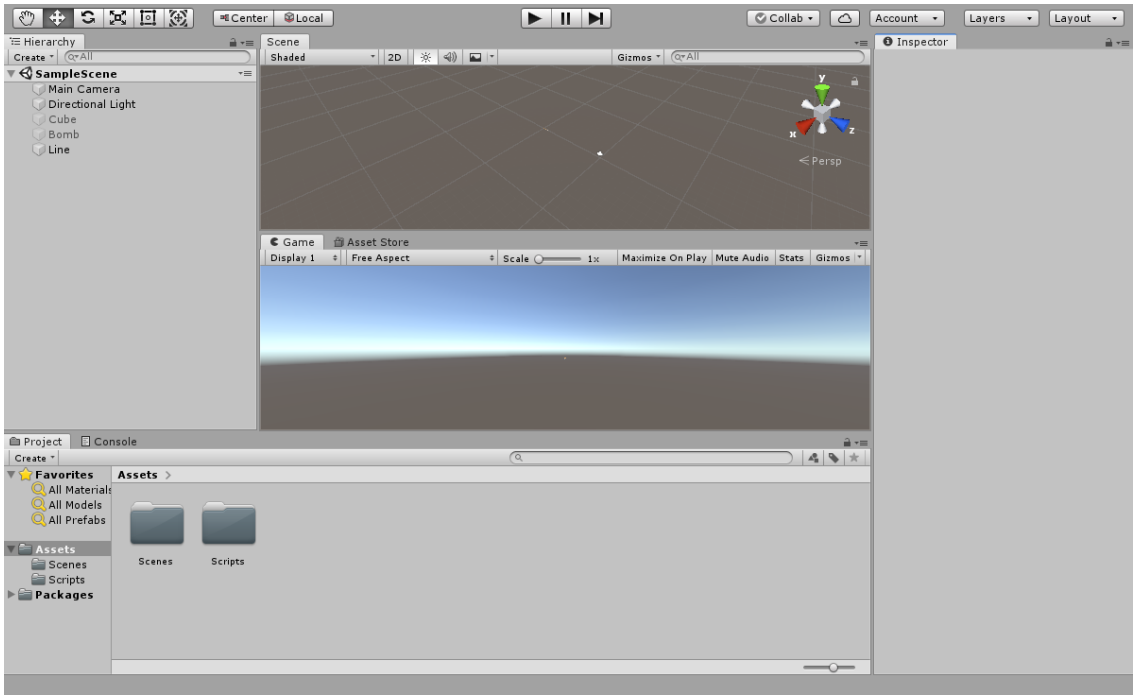
Išanalizavus ankščiau minėtus vizualizavimo būdus nuspręsta panaudoti visų trijų būdų privalumus, bandant išspręsti kiekvieno trūkumus. Trajektorijos vizualizavimas prototipe yra itin svarbus, tad, siekiant geriausių ir aiškiausių rezultatų, pasirinkta naudoti trimatį trajektorijos vizualizavimą „tuneliu“, o procedūrą vaizduoti „tuneliu“ [Lap10] bei „tvora“ [STM13].

2. ANKSČIAU KURTO PROTOTIPO ANALIZĖ

Šio darbo tikslas – patobulinti 2020 m. kurtą prototipą ir išspręsti problemas, susijusias su programų sistemų dizainu ir projektavimu. Tam atlikti vykdoma prototipo analizė, kad būtų galima nustatyti, kurie sistemos aspektai ir komponentai turi būti tobulinami arba keičiami. Siekiama, kad sukurtas galutinis prototipo variantas būtų kuo lengviau keičiamas arba plečiamas, pritaikomas kitiems oro uostams. Analizės metu įžvelgtos problemos bus sprendžiamos remiantis įgytomis programų sistemų projektavimo žiniomis, pritaikant geras programinės įrangos kūrimo praktikas. Analizė vyksta trimis etapais, kurių kiekviename analizuojama viena specifinė prototipo dalis (modulis).

2.1. Naudojama sistema ir įrankiai

Prototipui ir jo aplinkai kurti naudojama Unity žaidimų kūrimo mašina (žr. 7 pav.) (versija 2018.4.16f1).



7 pav. Unity 3D (versija 2018.4.16f1) žaidimų kūrimo mašinos pagrindinis langas

Pasirinkta žaidimų kūrimo mašina, nes joje naudojami įrankiai suteikia didžiausią prototipo lankstumą. Žaidimų kūrimo mašinos kaip Unity yra dažnai naudojamos kuriant sudėtingus ir itin didelio masto žaidimus 3D erdvėje, tad internete prieinama daug pagalbinės medžiagos, susijusios su šiuo darbu. Unity taip pat dažnai naudojama kuriant moksliniams darbams ir tyrimams skirtus įrankius bei žaidimus [GPS15], todėl galima teigti, kad ši žaidimų kūrimo mašina yra pakankamai išplėtota ir pajėgi darbui atlikti. Taip pat verta paminėti, kad Unity sistema yra nemokama kuriant

asmeninius projektus. Naudojant Unity integruotus įrankius daug naudos suteikia Unity sistemos dokumentacija ir pamokos, prieinamos internete [UniC18] ir [Uni19]. Visas programinis kodas prototipe rašomas C# programavimo kalba, dažnai naudojami numatytieji .NET karkaso tipai, klasės ir bibliotekos. Šiame darbe prototipo kūrimo sistema ir programavimo kalba nebus keičiami.

2.2. Prototipo komponentai

Prototipas sudarytas iš keleto esminių komponentų matomų Unity aplinkos „Hierarchy“ skiltyje (žr. 8 pav.).

8 pav. Pagrindiniai prototipo komponentai

Komponentus galima grupuoti į 3 pagrindines prototipo veikimo sritis: vietovė, duomenų valdymas ir simuliacija. Šiame skyriuje aptariami komponentai, esantys šiose srityse, ir nustatomi jų trūkumai bei problemos, kuriuos siekiama išspręsti. Taip pat atkreipiamas ypatingas dėmesys į prototipo moduliškumo (angl. *modularity*) tobulinimą.

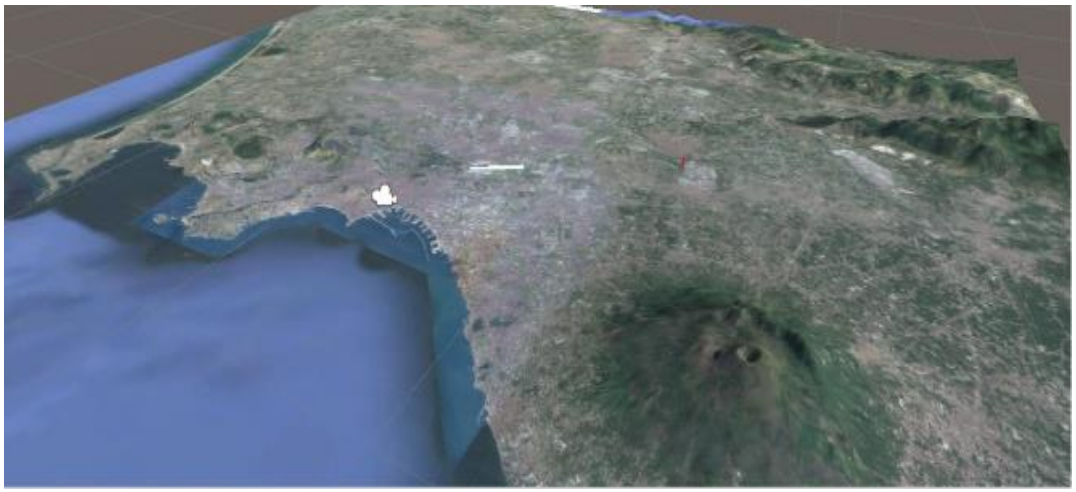
2.2.1. Vietovės komponentų analizė

Prototipe siekiama, kad aplinka atitiktų realų vietovės vaizdą, kad naudotojo matomas vaizdas būtų kuo intuityvesnis ir greičiau suprantamas. Orientavimasis erdvėje yra vienas svarbiausių tokio tipo sistemos aspektų. Unity aplinkoje sukurtas „Terrain“ objektas atitinka vietovę, kurioje vyks trajektorijos vizualizacija. Sukurtam Terrain objektui sugeneruotas reljefas naudojant „terrain.party“[2] įrankį. Terrain objekto matmenys pasirinkti remiantis turimų skrydžių informacija. Siekiant, kad vaizduojami skrydžiai būtų Terrain ribose, objekto plotis bei ilgis buvo keičiami keletą kartų.

Terrain objektui taip pat pritaikoma palydovinė vietovės nuotrauka, kad matomas vaizdas būtų kuo artimesnis realybei. Ši nuotrauka prototipe įkeliama virš Terrain objekto. Verta paminėti,

[2] https://terrain.party/

kad nuotrauka nepilnai užpildo visą Terrain objekto plotą, nes nėra priimti itin tikslūs vietovės matmenys. Norint, kad vietovės vaizdas būtų tikslesnis, nuotrauka dalinai iškraipoma naudojant Unity objektų transformavimo įrankius, „ranka“ keičiant jos ilgį ir plotį kol nuotrauka uždengia pakankamą Terrain objekto plotą. Prototipe sukurtas vietovės vaizdas matomas 9 pav.



9 pav. 2020 m. prototipo vietovės vaizdas

10 pav. VOR/DME radijo navigacijos stotis 2020 m. prototipe

Taip pat prototipe modeliuojama VOR/DME radijo navigacijos stotis (žr. 10 pav.). Anksčiau kurtame prototipe ši stotis modeliuojama paprasčiausiu cilindro objektu, kuris nedaro daug įtakos simuliacijos funkcionalumui. Stoties paskirtis 2020 m. prototipe yra iš esmės dekoratyvi – t.y. procedūros ir skrydžiai yra iš anksto nustatytose pozicijose, o ne priklausomai nuo šios stoties. Realybėje skrydžių vadovai remiasi duomenimis gautais iš VOR/DME radijo stoties, tad pasikeitus jos pozicijai turėtų keistis ir gaunami duomenys.

2020 m. prototipo vietovės komponentuose pastebėta keletas esminių trūkumų, susijusių su programų sistemų moduliškumu. Šiame darbe siekiama šias problemas išspręsti atsižvelgiant į tai, jog kiekviena programų sistema pagal taikytinus projektavimo principus, turėtų būti kuriama su

funkcionalumu, kuris leistų naudotojui sistemą naudoti taip, kad būtų tenkinami jo poreikiai. Tai reiškia, jog prototipe privalo būti galimybė pakeisti bet kurį esminį prototipo objektą taip, kad sistemos funkcionalumas nebūtų paveiktas, tačiau pakistų tiek, kiek reikia naudotojui [SGC01].

2020 m. prototipe problemos iškyla tuomet, kai norima orlaivio trajektorijos vizualizaciją pritaikyti kitai vietovei. Šio darbo atveju, ši vietovė keičiama iš Neapolio Capodichino oro uosto į Vilniaus tarptautinį oro uostą. Pastebimos kelios esminės problemos bandant vykdyti tokį pakeitimą:

1. Sukurtas Terrain objektas nėra lengvai modifikuojamas;
2. Nėra nustatyti universalūs Unity aplinkos matmenys Terrain objektui – galimi žymūs atstumų neatitikimai;
3. VOR/DME radijo navigacijos stotis nedaro jokios įtakos simuliacijai, t. y. stotis gali būti keičiama ar perkeliama į kitą vietą, bet orlaivio trajektorija bei procedūros ribos nekinta.

2.2.2. Duomenų valdymo komponentų analizė

Esminiai prototipe naudojami duomenys – skrydžių koordinatės kiekvienu laiko momentu ir tūpimo ar pakilimo procedūros ribų koordinates erdvėje. Šie duomenys saugomi CSV (angl. *Comma Separated Values*) tipo failuose, kur kiekviena eilutė atitinką kitą įrašą duomenyse (pvz., sekančios koordinatės eilėje).

Duomenų valdymas 2020 prototipe vyksta gan paprastai – visi skrydžių ir oro uosto tūpimo ar pakilimo procedūrų duomenys yra skaitomi iš CSV tipo failų esančių naudotojo kompiuteryje. Už šias procedūras atsakingas „Data manager“ komponentas. Failų skaitymas programiniame kode vyksta naudojant įprastas .NET karkaso bibliotekas, kreipiantis tiesiogiai į CSV failą tam tikroje direktorijoje (žr. 11 pav.).

```
using (var reader = new StreamReader("C:/Assets/ILS-V.CSV"))
{
    while (!reader.EndOfStream)
    {
        var line = reader.ReadLine();
        var nextLine = reader.ReadLine();
```

11 pav. Duomenų skaitymas iš failo programiniame kode

Prireikus naudoti naują duomenų failą, pavyzdžiui, jei reikia pavaizduoti kitą tūpimo ar pakilimo procedūrą, prireiktų keisti programinį kodą matomą 11 pav. kad būtų naudojamas tinkamas duomenų failas. Tokie atvejai programiniame kode dažnai vadinami „kietu

programavimu“ (angl. *hard coding*). Ši programavimo praktika pastebima įvairaus pobūdžio programiniame kode [LMZ21], dažnai tuo atveju, kai programuotojai neturi laiko įgyvendinti tinkamą duomenų valdymą. Ankstesniame prototipe pastebėtas atvejis nevisiškai atitinką kietojo programavimo apibrėžimą, nes programiniame kode nėra saugomi konkretūs duomenys. Vis dėlto, tokios praktikos vengiama, nes taip įgyvendinta sistema nėra lengvai plečiama.

„Data manager“ komponentas yra atsakingas už tinkamą ir tikslingą skrydžių koordinačių saugojimą programos veikimo metu. Prototipe simuliuojami skrydžiai visada kyla nuo arba leidžiasi link oro uosto pakilimo tako. Šio pozicija 2020 m. prototipe yra fiksuota ir nekintanti, nes prototipas kurtas tik Neapolio Capodichino oro uosto skrydžių valdymui. „Data manager“ komponentas yra atsakingas už šių skrydžių koordinačių atitikimą nusileidimo tako atžvilgiu – t.y. skrydžiai turi prasidėti arba baigtis ant erdvėje esančio nusileidimo tako. Tam 2020 m. prototipe buvo naudojamas konkretus skaičius, apskaičiuotas „ranka“ matuojant nusileidimo tako poziciją simuliacijoje (žr. 12 pav.).

```
var coordinates = new Coordinates(
    Convert.ToDouble(data[0]) * NM_M - 0.45132,
    Convert.ToDouble(data[1]) * NM_M - 0.56441,
    Convert.ToDouble(altitude) * FT_M + 0.16854,
    flightNumber);
```

12 pav. Skrydžių koordinačių skaičiavimas simuliacijoje pakilimo tako atžvilgiu

Tai, kaip ir anksčiau minėta problema, yra tipinis kietojo programavimo pavyzdys. Šiuos skaičius gauna programuotojas savo nuožiūra, o ne simuliacijoje naudojamų objektų pozicijų pagalba. Bandant pritaikyti šį prototipą kitam oro uostui, pvz., Vilniaus, tektų pakeisti šiuos skaičius kitais, naujai apskaičiuotais, pagal pasikeitusią pakilimo tako poziciją simuliacijoje. Verta paminėti, kad tokie, konkrečių skaičių programiniame kode, atvejai pasitaiko keletą kartų ankstesnio prototipo duomenų valdymo kode.

Dar vienas, lengvai nepastebimas duomenų valdymo aspektas, yra tinkamų duomenų skaitymas. Gali atsitikti, kad naudotojas nežino, kokiu formatu turi būti saugomi duomenys CSV failuose. Šiame prototipe naudojamas formatas skrydžių koordinatėms CSV failuose saugoti: laikas, skrydžio identifikatorius, koordinatė X, koordinatė Y, aukštis Z. Jei naudotojas, nežinantis šio formato, pateiktų kitokio tipo CSV failą, prototipas nustotų veikti ir nepavaizduotų jokios vizualizacijos. Naudotojas nesulaukia jokio grįžtamojo ryšio apie tinkamą duomenų formatą. Šiuo atveju turėtų būti siekiama, kad skaitomi tik teisingu formatu saugomi duomenys, kad nebūtų galimybės pvz., įrašyti žodžius į koordinatės vietą [LMZ21].

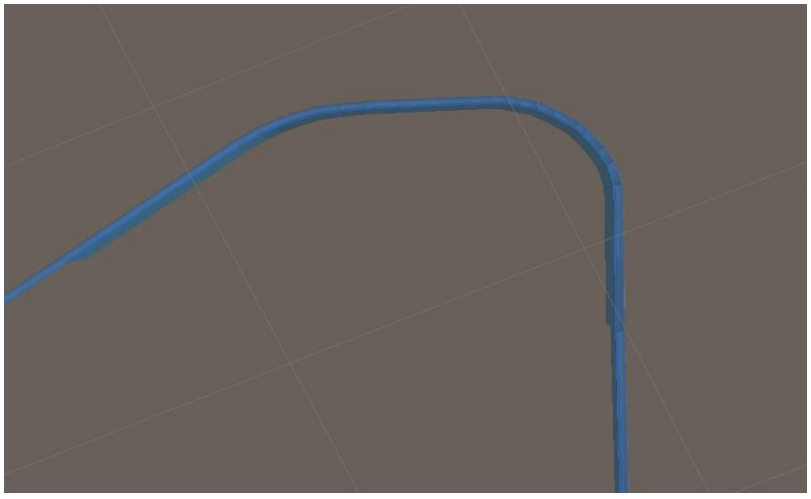
Taigi, plečiant šį prototipą, siekiant jį pritaikyti Vilniaus oro uostui, pastebimos šios esminės problemos duomenų valdymo srityje:

1. Duomenų skaitymas iš failų yra neišplečiamas – failų vardai saugomi programiniame kode;
2. Keletas „kietojo programavimo“ atvejų, kai yra nustatomas ir programiniame kode įrašomas apytiksliai tinkamas konkretus skaičius arba matmuo;
3. Koordinačių duomenų tikslingumas nėra tikrinamas, skaitomi bet kokie naudotojo pateikti failai.

2.2.3. Simuliacijos valdymo komponentų analizė

Esminis prototipo funkcionalumas įgyvendintas „Simulation manager“ komponente. Jame esantys vidiniai komponentai atsakingi už orlaivio judėjimą erdvėje, trajektorijos braižymą ir tikrinimą, ar orlaivis laikosi nustatytų pakilimo ar tūpimo procedūros ribų. Procedūrų laikymosi tikrinimas yra svarbiausias funkcionalumas, reikalingas potencialiems sistemos naudotojams – skrydžių vadovams. Procedūrų ribos simuliacijoje privalo būti lengvai ir aiškiai matomos, naudotojas neturi priimti sprendimų remiantis nuojauta. Atsižvelgiant į šiuos reikalavimus ankstesniame prototipe sumodeliuotos lėktuvo tūpimo procedūros ribos matomos 13 pav.

Šiame prototipe procedūros vaizdavimas erdvėje vykdomas gan paprastai – procedūra vaizduojama 3D objektu, kuris privalo būti sumodeliuotas iš anksto, prieš pradedant naudoti prototipą. Modelis atstovavo bet kurią vieną oro uoste naudojamą ILS tipo tūpimo ar pakilimo procedūrą. Sukurtas modelis yra šešiakampio tunelio formos, į kurį lėktuvas turėtų įskristi tūpimo metu arba išskristi iš jo pakilimo metu.



13 pav. Tūpimo procedūros modelis erdvėje 2020 m. prototipe

Lėktuvui patekus į šį tunelį naudotojui būtų rodoma, kad pilotas laikosi nustatytų oficialių tūpimo ar pakilimo procedūrų. Orlaiviui nukrypus nuo tunelio ribų, naudotojas taip pat matytų,

kad pilotas nesilaiko procedūrų ribų, atsiranda rizika. Šis funkcionalumas iš esmės pakankamas, kai sistema pritaikyta tik vienam oro uostui ir vienai jame naudojamai tūpimo ar pakilimo procedūrai. Šiuo atveju parinkta tūpimo procedūra buvo „ILS-V“, naudojama Neapolio Capodichino oro uoste.

Šio funkcionalumo problema pastebima norint sistemoje naudoti, pavyzdžiui, kitą procedūrą arba tos pačios procedūros pakilimo ribas – tokiu atveju naudotojas turėtų sumodeliuoti naują procedūros ribų 3D modelį. Modelis gali būti kuriamas Unity sistemoje arba naudojant kitą 3D modelių kūrimui skirtą programą kaip „Blender“. Akivaizdu, kad potencialus šio prototipo naudotojas ne visada turės galimybę sukurti naują modelį kiekvienai naujai ar pakeistai pakilimo ar tūpimo procedūrai. Čia, panašu, vėl pastebima programų sistemos moduliškumo problema – naudotojui turėtų būti suteikiama galimybė pasirinkti naudoti bet kokią ILS procedūrą sistemos veikimo metu. „ ... a portfolio of options is worth more than an option on a portfolio. It is in this sense that modularity, per se, adds value.“ [SGC01].

Neskaitant šio trūkumo pastebimas dar vienas – sukurtas procedūros ribų modelis yra nesusijęs su anksčiau minėtu oro uosto VOR/DME radaru. Tai gali sukelti problemų, nes naudotojas nežino, kurioje pozicijoje erdvėje tiksliai turėtų būti apibrėžiamos tūpimo ar pakilimo procedūros ribos. Tai reiškia, kad net ir sukūrus naują modelį naujai procedūrai, naudotojas turėtų pats nuspręsti, kur procedūros ribos turėtų būti, kad būtų galima tikrinti, ar pilotas laikosi šių nustatytų ribų. Naudojantis įgytomis žiniomis avionikos srityje, žinoma, kad procedūros ribos turėtų būti apibrėžtos VOR/DME radijo navigacijos stoties pozicijos atžvilgiu, kad būtų užtikrintas kuo geresnis pozicijos ir koordinačių tikslumas.

Taigi, plečiant šį prototipą, siekiant jį pritaikyti Vilniaus ar kitiems oro uostams ir ypač kitoms ILS procedūroms, pastebimos šios esminės problemos simuliacijos valdymo srityje:

1. Naudotojas neturi galimybės lengvai pakeisti vaizduojamos ILS procedūros ir jos ribų – pakeitimui reikalingos 3D modelių kūrimo žinios;
2. Vaizduojama procedūra erdvėje yra dėstoma remiantis naudotojo ar programuotojo spėjimu, pozicija nėra užtikrintai tiksli.

2.3. Analizės išvados

Atlikus anksčiau kurto prototipo analizę pastebėta keletas svarbių sistemos trūkumų. Dauguma problemų susiję su programų sistemos moduliškumu, taip pat pastebimi keletas kietojo programavimo atvejų. Šio darbo metu kuriant naują, patobulintą prototipą siekiama išspręsti pastebėtas ankstesnio prototipo problemas. Naujas prototipas turėtų tinkamai įgyvendinti visą funkcionalumą, kurio galimai reikalautų potencialūs skrydžių vadovai, ir suteikti galimybę prototipą naudoti bet kokiame oro uoste su jame naudojamomis tūpimo ir pakilimo procedūromis.

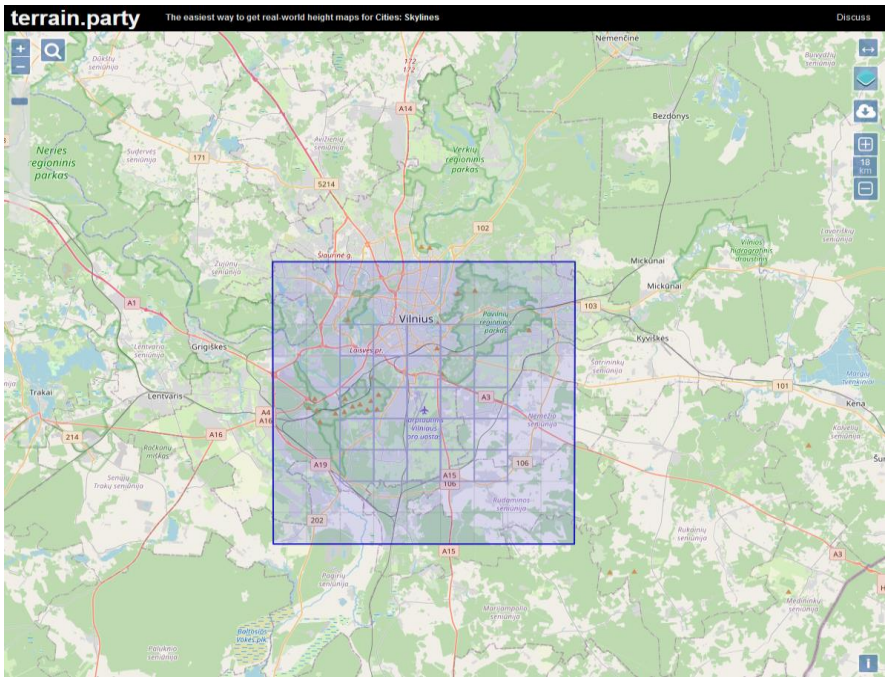
3. VILNIAUS ORO UOSTO APLINKOS MODELIAVIMAS

3.1. Mastelis

Kuriamoje programoje priimta, kad vienas Unity aplinkos matavimo vienetas (angl. *unit*) atitinka viena kilometrą (1000 m) realybėje. Visi programoje naudojami matavimo vienetai yra konvertuojami į Unity aplinkos matavimo vienetus. Taip pat verta paminėti, kad visi skrydžių koordinačių duomenys yra pateikiami jūrmylėmis (X ir Y pozicijoms) arba pėdomis (aukščiui, Z pozicijai). Konvertavimui iš jūrmylių ir pėdų į metrus C# programiniame kode naudojami statiniai kintamieji su atitinkamomis reikšmėmis 1852 ir 0,3048. Oro uosto pakilimo takas, lėktuvo dydis bei radijo navigacijos stotis VOR/DME yra padidinti dėl aiškumo.

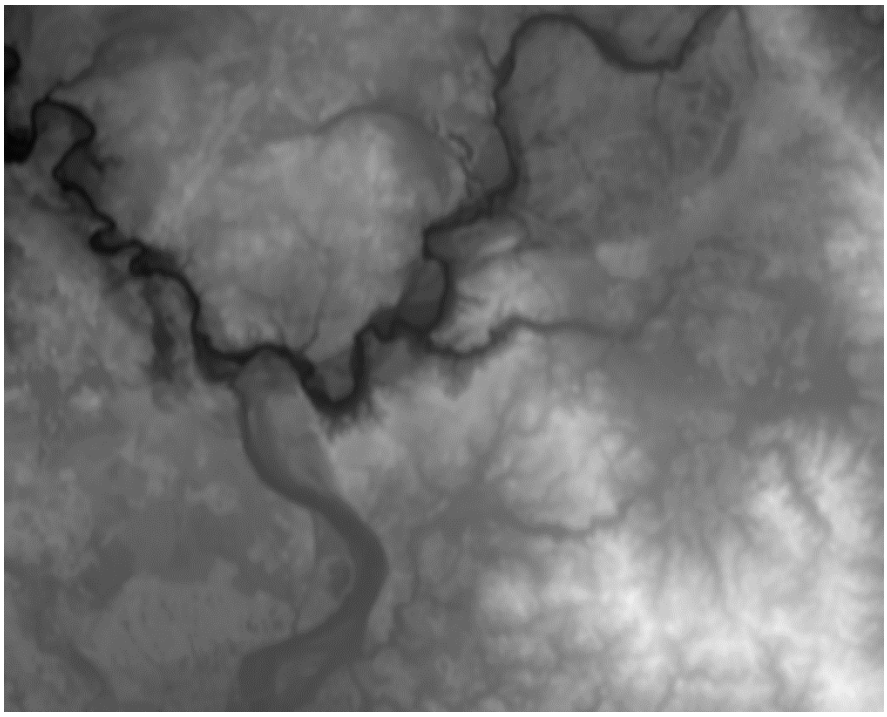
3.2. Aplinkos ir reljefo kūrimas

Prototipe vaizduojama aplinka yra 2500 kvadratinių kilometrų dydžio (50 km x 50 km). Lyginant su 2020 m. prototipu, modifikuojami Terrain objekto kūrimo etapai – objektas sukuriamas tik vieną kartą ir jam nustatomi fiksuoti matmenys – 50 x 50 Unity sistemos vienetų. Terrain objektas nebus keičiamas erdvėje, tačiau galimi pakeitimai jo Texture ir Layer parametrams. Šie parametrai priima JPEG, JPG ir PNG formatų nuotraukas, kurios atitinkamai lemia Terrain objekto išvaizdą (palydovinė vietovės nuotrauka) ir reljefą (vietovės aukščių žemėlapis). Terrain objekto reljefo nuotraukai sugeneruoti naudojamas „terrain.party“ įrankis (žr. 14 pav.).



14 pav. „terrain.party“ įrankis

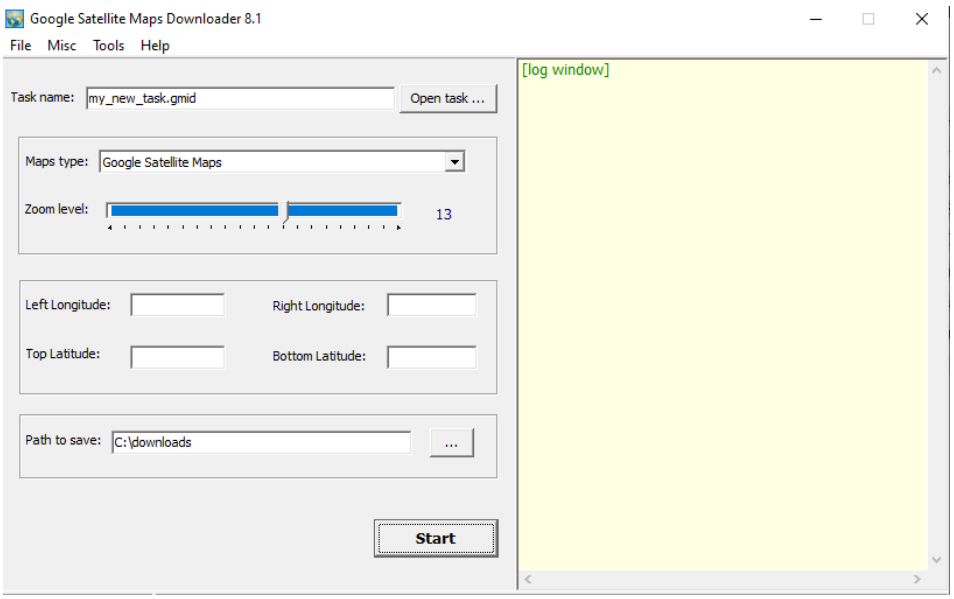
Naudojant šį įrankį pažymimas 50 km^2 plotas, kurio centre yra Vilniaus tarptautinis oro uostas. Pagal pasirinktą plotą sugeneruojami keletas reljefo žemėlapių (angl. *heightmap*). Iš šių prototipui pasirinkta naudoti „Merged“ (sulietas) sugeneruotą reljefo žemėlapį, kuriame pateikiama visa reikiama reljefo aukščių informacija (žr. 15 pav.).



15 pav. Sugeneruotas sulietas reljefo žemėlapis

Šis failas įkeliamas į Unity aplinką ir priskiriamas anksčiau sukurto „Terrain“ objekto „Texture“ parametrui.

Terrain objekto „Layer“ taip pat pritaikoma palydovinė vietovės nuotrauka, kad matomas vaizdas būtų kuo artimesnis realybei. Palydovinei nuotraukai gauti naudojamas „Google Satellite Maps Downloader“ įrankis (žr. 16 pav.). Įrankyje pasirenkamos Vilniaus vietovės koordinatės ir spaudžiamas mygtukas „Start“. Įrankiui baigus darbą atsiunčiamos palydovinės nuotraukos. Toliau spaudžiamas mygtukas „Tools“ ir „Map Combiner“. Atsidariusiame lange pasirenkamos atsiųstos palydovinės nuotraukos ir spaudžiamas mygtukas „Combine“. Šis įrankis sujungia visas atsiųstas nuotraukas į vieną, kurią vėliau galima panaudoti Unity aplinkoje. Sujungta palydovinė nuotrauka priskiriama Terrain objekto „Layer“ parametrui.

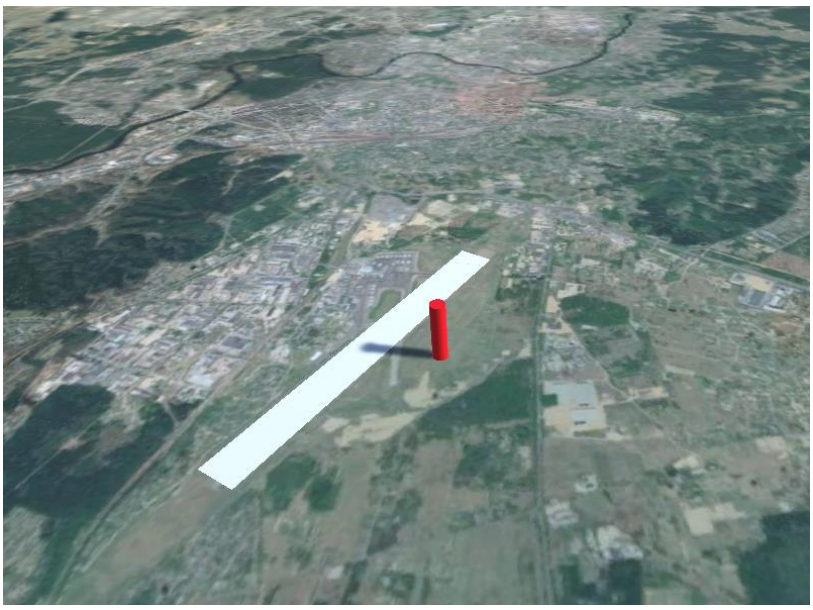


16 pav. „Google Satellite Maps Downloader“ įrankis

Atsižvelgiant į analizės metu pastebėtus ankstesnio prototipo trūkumus Vilniaus vietovei pavaizduoti skirtas Terrain objektas sukurtas naudojant paprastas reljefo ir palydovines nuotraukas. Terrain objektas yra universalaus, 50 x 50 km dydžio, tad norint pakeisti vietovę užtenka žinoti tik vietovės koordinates ir, naudojant anksčiau minėtus arba kitus įrankius, gauti reljefo žemėlapį bei palydovinę vietovės nuotrauką 2500 km^2 vietovės plote. Tuomet nuotraukos lengvai importuojamos į Unity aplinkoje esantį Terrain objektą. Šis funkcionalumas yra patobulintas, lyginant su ankstesniu prototipu, nes naudotojas turi galimybę bet kuriuo metu pakeisti Terrain objekto parametrus Layer ir Texture, taip pakeičiant matomą vietovę. Šis funkcionalumas .

3.3. Pakilimo tako ir radaro kūrimas

Aiškiam ir intuityviam prototipo vaizdui taip pat modeliuojamas Vilniaus oro uosto pakilimo takas. Šį vaizduoja balta, stačiakampio gretasienio formos dėžė. Ilgis atitinka realybėje esantį ~2,6 km, plotis padidintas dėl aiškumo. VOR/DME „VNO“ radijo navigacijos stotis pavaizduota raudonu, 200 m aukščio cilindru, aukštis padidintas dėl aiškumo. Pakilimo takas ir „VNO“ radijo stotis matomi atitinkamai 17 pav.

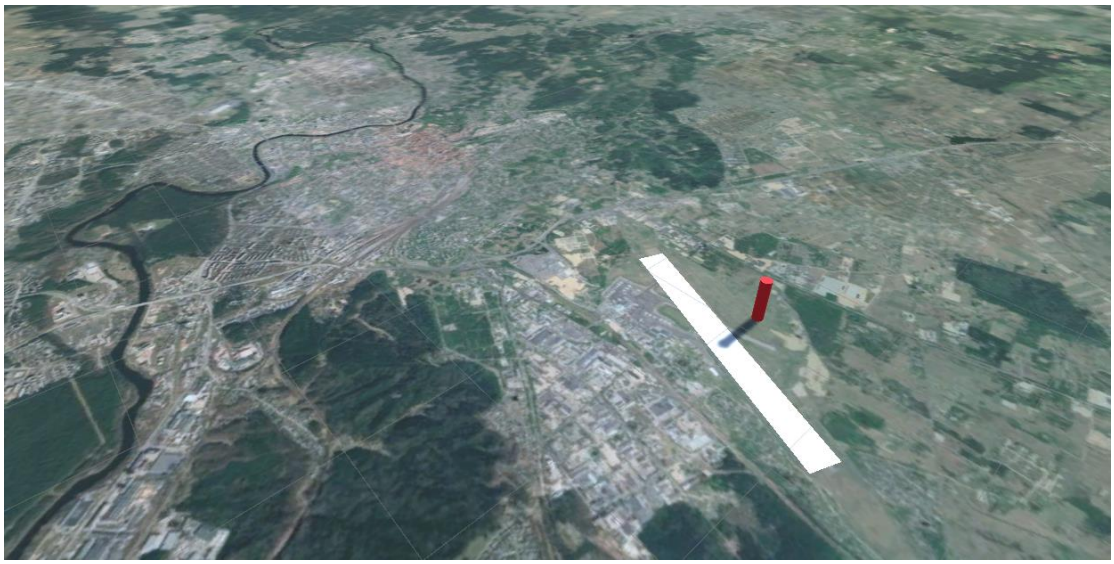


17 pav. Sumodeliuotas Vilniaus oro uosto pakilimo takas bei VOR/DME „VNO“ radijo navigacijos stotis

Siekiant kuo tobulesnio vietovės tikslumo VOR/DME „VNO“ radijo stoties pozicija nustatyta naudojant „Google Maps“ prieinamas Vilniaus vietovės palydovines nuotraukas. Radijo stoties atstumas nuo pakilimo tako buvo tiksliai išmatuotas ir VOR/DME modelis Unity aplinkoje buvo atitinkamai padėtas tikslioje pozicijoje. Radaro padėties realiame gyvenime pagal „Google Maps“ palydovo vaizdą ir objekto, esančio Unity 3D aplinkoje, padėties palyginimas matomas 18 pav. Verta vėl paminėti, jog pakilimo takas ir radaras prototipe yra žymiai padidinti dėl aiškumo.

18 pav. Kairėje – realus Vilniaus oro uosto palydovinės nuotraukos vaizdas, dešinėje – prototipe sumodeliuota vietovė

Taip pat priimtas sprendimas, kad VOR/DME radijo navigacijos stotis yra reikšminga nustatant orlaivio poziciją ir tūpimo ar pakilimo procedūros ribas erdvėje. Daroma prielaida, kad pritaikant prototipą bet kuriam kitam oro uostui ir jame vykdomiems skrydžiams, visada bus reikalingas bent vienas pagrindinis VOR/DME radaras. Pagal šį radarą yra orientuojami skrydžiai ir procedūrų ribos simuliacijoje. Vilniaus oro uosto atveju, lyginant su Neapolio Capodichino oro uostu, radaras yra arčiau pakilimo tako, tad yra būtinybė keisti radaro poziciją ant Terrain objekto. Šiame darbe buvo atlikti komponento „VOR/DME“ ir „Simulation manager“ pakeitimai, kurie leidžia procedūros riboms būti nustatomoms prototipo paleidimo metu, priklausomai nuo radaro pozicijos.



19 pav. Unity aplinkoje sumodeliuota vietovė

Baigtas vietovės vaizdas matomas 19 pav. Potencialūs Vilniaus oro uosto skrydžių vadovai matytų tokį aplinkos vaizdą trajektorijos vizualizacijos prototipe. Reljefas atitinka realybę, tačiau verta paminėti, kad „Google Satellite Maps Downloader“ įrankis nėra idealiai tikslus – 50 km^2 vietovės vaizdas „suklijuotas“ iš keleto atskirų palydovinių nuotraukų. Šio trūkumo pasekmė yra minimalūs reljefo ir palydovinės nuotraukos neatitikimai. Verta paminėti, kad pritaikant kito oro uosto vietovės nuotraukas naudotojas galimai turėtų prieigą prie tikslesnių palydovinių nuotraukų, naudojant kitus įrankius. Reljefo žemėlapis (angl. *heightmap*) sugeneruotas naudojant „terrain.party“ įrankį, kuris naudoja keletą skirtingų, patikimų duomenų rinkinių, tad šio įrankio generuojami žemėlapiai yra daug tikslesni, nei anksčiau gautos palydovinės nuotraukos. Reljefo vaizdas yra pagalbinis skrydžių vadovui, tai nėra realių koordinačių ir radarų duomenų pakaitalas.

4. TRAJEKTORIJOS VIZUALIZAVIMAS

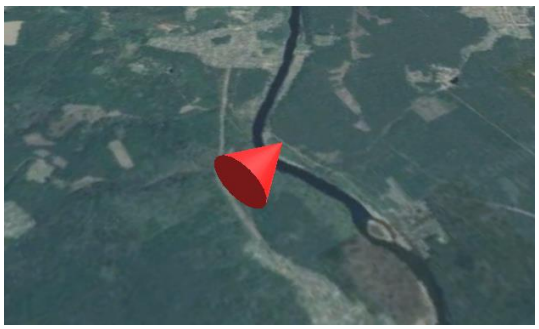
Trajektorijos vizualizavimas susideda iš dviejų pagrindinių dalių – lėktuvo modeliavimo ir valdymo simuliacijoje bei lėktuvo trajektorijos perteikimo. Trajektorijos generavimui bei lėktuvo valdymui rašomas programinis kodas „Scripts“ aplanke. Projekte stengiamasi kodą skirstyti į atskirus valdiklius (angl. *manager*), kiekvieno komponento valdymui kuriamas naujas valdiklis. Visas kodas Unity aplinkoje rašomas C# programavimo kalba.

4.1. Lėktuvo modeliavimas ir valdymas

Skrydžių vadovui itin svarbu suprasti, kokia orlaivio pozicija erdvėje bei kokia kryptimi jis juda. Lėktuvo modelis, dydis ar spalva nėra taip svarbu, tokios detalės simuliacijoje gali painioti naudotoją. Remiantis panašiais tyrimais aviacijoje, nuspręsta, kad skrydžių vadovui nėra būtina matyti idealų lėktuvo modelį. Panašūs tyrimai aviacijos srityje dažnai lėktuvą trimatėje erdvėje vaizduoja kaip kūgio formos objektą [BTD15], kurio smaigalys atitinka lėktuvo priekį. Remiantis šiomis žiniomis, šiame darbe nuspręsta lėktuvą taip pat vaizduoti kūgiu (žr. 13 pav.). Kūgiui sukurti Unity aplinkoje nėra integruotų įrankių, tad šio objekto generavimui naudojamas kodas rastas Unity bendruomenės sukurtame Unity3D „wiki“ puslapyje [UniC18]. Šis kodas gali būti kviečiamas bet kada simuliacijos veikimo metu, tad yra patogu kurti naujus lėktuvų objektus, pakeisti jų poziciją, dydį ir kiekį.

Simuliacijoje visi skrydžių duomenys yra gauti iš ankstesnių Vilniaus universiteto studentų bei dėstytojų darbų [LČŠ12]. Pagal šiuos radarų duomenis matoma, kokie skrydžiai leidosi arba kilo iš Neapolio Capodichino oro uosto 2015 m. spalio 6 d.

Dauguma Neapolyje vykdytų skrydžių buvo „Embraer“ įmonės E170 modelio lėktuvais, kurių ilgis ~26 m, sparnų plotis ~30 m. Remiantis šiais duomenimis nuspręsta kūgio išmatavimus daryti panašius – aukštis (lėktuvo ilgis) 30 m, skersmuo (lėktuvo sparnų plotis) 30 m. Išmatavimai šiek tiek padidinti dėl aiškumo. Taip pat nuspręsta, kad nebus būtinybės naudotojui keisti kūgio dydį, nes itin tikslūs lėktuvo išmatavimai nedaro daug įtakos simuliacijos vaizdavimui. Lėktuvo modelis Unity aplinkoje matomas 20 pav.



20 pav. Lėktuvo modelis matomas iš viršaus (esant pažeidimams lėktuvo spalva raudona)

Lėktuvo objekto valdymas vykdomas „Aircraft Manager“ komponente. Orlaivio objektas juda erdvėje pagal konkretaus skrydžio koordinates, gautas iš anksčiau minėtų radarų duomenų. Koordinačių nėra daug, tad sklandžiam lėktuvo judėjimui naudojama „MoveTowards“ funkcija [Uni19]. Šios veikimo principas primena interpoliaciją, nes objektas juda nuo vieno taško iki kito, tačiau pajuda tik apskaičiuotą atstumą tarp šių dviejų taškų. Šis tarpinis atstumas tarp dviejų taškų skaičiuojamas sudauginus lėktuvo greitį, kuris aprašomas vienetais per sekundę, ir baigimo laiką sekundėmis nuo praeito kadro (angl. *frame*) simuliacijoje. Pastarasis gaunamas naudojant „Time.deltaTime“ funkciją [Uni19].

Realybėje tūpimo metu skrydžių vadovai tikisi, kad pilotai neviršys nustatyto 200 mazgų greičio 6000 pėdų arba žemesniame aukštyje [FAA17]. Labai svarbu, kad pilotai dėtų visas įmanomas pastangas skrydžių vadovų numatytų nusileidimo veiksmų vykdymui, kad padėtų saugiai valdyti ir paspartinti oro eismą. Atsižvelgiant į tai, šiame darbe nuspręsta, kad lėktuvo greitis simuliacijoje turi būti 0.07 vienetų per sekundę (apie 250 km/h). Tai atitinka realų vidutinį lėktuvo greitį nusileidimo metu.

4.2. Lėktuvo skrydžio koordinačių duomenų valdymas

Verta paminėti, kad realioje situacijoje naudotojas turėtų tikslių skrydžio koordinačių šaltinį, pvz., radarus, kurie dažnai atnaujintų skrydžio trajektorijos informaciją. Šio prototipo funkcionalumui įgyvendinti nuspręsta, kad užtenka anksčiau gautų, tikslių Neapolio oro uosto skrydžių duomenų. Šie duomenys nedaro daug įtakos pritaikius juos Vilniaus oro uostui, nes nustatoma, kad kiekvienas skrydis turi prasidėti nuo arba leistis ant oro uosto pakilimo tako. Šis skrydžių „sulygiavimas“ su Vilniaus oro uosto pakilimo taku vykdomas „Data manager“ komponente. Šiame komponente skaitant duomenis iš CSV failo ieškoma skrydžio koordinačių, kurių „altitude“ reikšmė lygi nuliui. Ši orlaivio pozicija priimama kaip „atskaitos taškas“ visiems galimiems skrydžiams – t.y. visi skrydžiai „lygiuojami“ link arba nuo pakilimo tako pagal šias koordinates.

Taip pat, atsižvelgiant į ankstesnio prototipo duomenų valdymo srities analizės metu pastebėtas problemas, įgyvendinti keletas pakeitimų „Data manager“ komponente. Vienas svarbesnių – programiniame kode nebėra saugomi procedūrų ar skrydžių koordinačių duomenų failų vardai. Naudotojas turi galimybę parinkti duomenų failus, iš kurių bus skaitomos koordinatės skrydžių trajektorijoms ir procedūros ribų pavaizdavimui. Failų vardai nustatomi „Simulation manager“ komponento parametrų laukuose Unity aplinkoje prieš pradedant simuliaciją.

Taip pat ištaisyti kietojo programavimo atvejai „Data manager“ komponente. Vietoj skaičių naudojamos Unity objektų pozicijos, pvz., runway.position.y, kuris nurodo nusileidimo tako

poziciją Unity vienetais Y ašyje nuo pradinio taško (0;0;0). Šis pakeitimas iš esmės tobulina prototipo moduliškumą ir mažina galimų klaidų skaičių [LMZ21].

„Data manager“ komponente visiškai pakeistas duomenų skaitymas iš failų. Skrydžių koordinačių skaitymas vyksta naudojant reguliariąsias išraiškas (žr. 21 pav.) (angl. *regular expressions*).

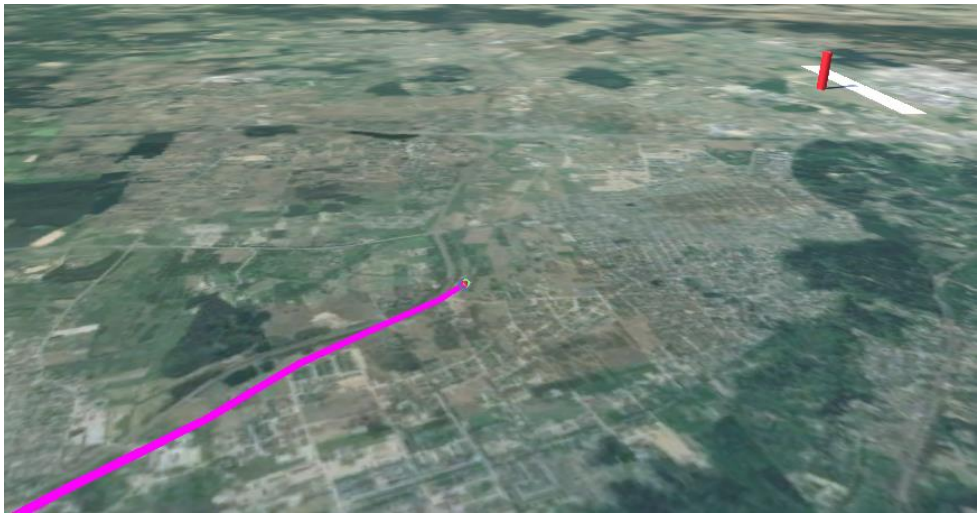
```
string coordinatesPattern = @"(-?\d{1,3}\.\d{1,6}|-?\d{1,3}),(-?\d{1,3}\.\d{1,6}|-?\d{1,3})";
string altitudePattern = @"([0-9]$|[1-9]\d{1,6}$)";
string flightNumberPattern = @"[A-Z]{1,4}(\d|[A-Z]){2,4}";
```

21 pav. Duomenų formato verifikavimui naudojamos reguliariosios išraiškos

Išraiškose nurodoma, kokiu formatu privalo būti aprašyti duomenys pateiktame faile. Pirmoji reguliarioji išraiška, matoma programiniame kode 21 pav., nurodo, kad koordinatės, pateikiamos duomenų faile, privalo būti sudarytos iš nemažiau kaip 2 skaičių, kurie gali būti nesveiki, tačiau negali būti sudaryti iš daugiau nei 3 skaitmenų. Tai iš esmės užtikrina tikslingą simuliacijos veikimą, nes naudotojas neturi galimybės pateikti netinkamus duomenis. Naudotojo pateikti failai nėra skaitomi, jei duomenų formatas neatitinka reguliariųjų išraiškų. Naudotojui taip pat pateikiamas pranešimas, jei formatas yra netinkamas.

4.3. Lėktuvo trajektorijos vizualizacija

Lėktuvo trajektorija vaizduojama naudojant Unity aplinkoje prieinamą „Line Renderer“ komponentą [Uni19]. Šis priskiriamas sukurtam lėktuvo modeliui kūgiui, kuriam judant erdvėje trajektorija automatiškai braižoma linija. Trajektorijos vaizdavimui pasirinkta naudoti ryškią, lengvai matomą spalvą (žr. 22 pav.).



22 pav. Lėktuvo trajektorijos vizualizacija prototipe

5. PROCEDŪROS IR JOS LAIKYMOSI VIZUALIZAVIMAS

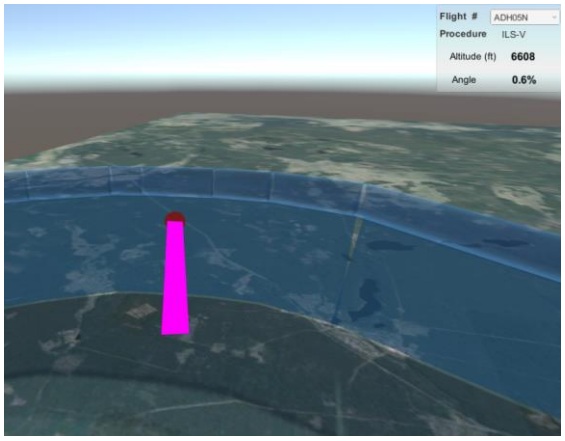
Skrydžių vadovai nusprendžia, kokios tūpimo ar kilimo procedūros pilotas privalo laikytis. Šiame prototipe nagrinėjama „ILS-Z“ Vilniaus oro uosto tūpimo procedūra, tačiau yra galimybė procedūrą pakeisti bet kokia kita. Tam atlikti naudotojas privalo pateikti CSV tipo failą su koordinačių sąrašu. Tūpimo procedūra naudotojui turi būti pateikta aiškiai ir intuityviai. Siekiant šių rezultatų procedūrą nuspręsta vaizduoti šešiakampiais tuneliais, nuo kurių nutiesta „siena“ į žemę. Siena padeda lengviau matyti atstumą iki reljefo iškilimų, tai leis skrydžių vadovui intuityviai interpretuoti orlaivio manevrus procedūros ribose.

ILS procedūros ribų (tunelio) vaizdavimas šiame prototipe yra visiškai pakeistas, siekiant kuo geresnio programų sistemos moduliškumo. Pagal analizės metu pastebėtas problemas nuspręsta, kad procedūrą geriausia vaizduoti generuojant kiekvieną procedūros ribų atkarpą pagal naudotojo pateiktą CSV duomenų failą. Šiame faile turėtų būti pateiktas sąrašas koordinačių VOR/DME radaro atžvilgiu, pagal kurias prototipo veikimo metu yra generuojamos procedūros ribos. Šis funkcionalumas naudotojui leidžia procedūrą pakeisti bet kuriuo metu, paprasčiausiai pateikiant kitą procedūros CSV failą. Ankstesniame prototipe procedūros vaizdavimui buvo reikalingas „ranka“ sumodeliuotas tunelis, kurio, nekuriant naujo, nebuvo įmanoma pakeisti. Šis naujas procedūros generavimas labai pagerina visos sistemos moduliškumą ir išsprendžia analizės metu pastebėtas problemas.

23 pav. Prototipe sugeneruota ILS-Z tūpimo procedūra

5.1. Procedūros ribų atitikimas

Skrydžių vadovas privalo lengvai suprasti, ar pilotas tinkamai laikosi procedūros nustatyto kurso. Šioje simuliacijoje lėktuvo spalva priklauso nuo objekto atstumo iki idealaus kurso. Jei modelis yra procedūros kurso ribose, spalva yra žalia, jei nukrypstama iš kurso ribų, modelis raudonuoja (24 bei 25 pav.). Ši informacija taip pat pateikiama naudotojo matomoje panelėje.



24 pav. Lėktuvas nukrypo nuo procedūros ribų

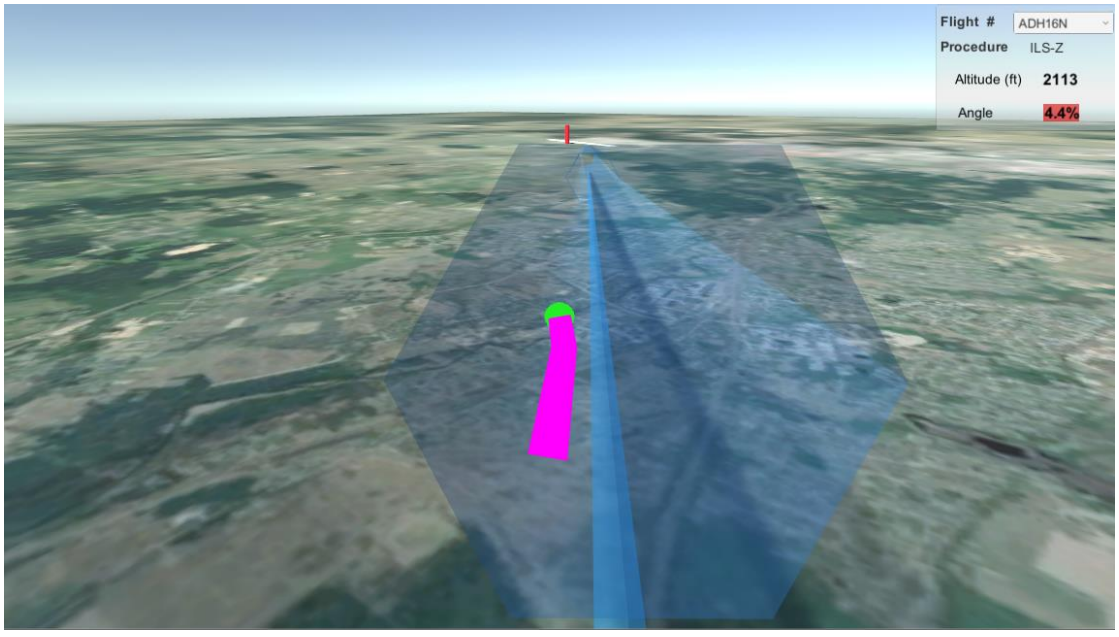
25 pav. Lėktuvas procedūros ribose

6. BAIGTAS PROTOTIPAS

Baigto prototipo generuojamas vaizdas parodytas 26 pav. Paveikslėlyje pavaizduotas naudotojui, t.y. skrydžių vadovui matoma vartotojo sąsaja. Kamera seka lėktuvą iš nugaros. Dešinėje kampe matoma panelė, kurioje pateikta skrydžių vadovui aktuali informacija – skrydžio identifikatorius, procedūros pavadinimas, orlaivio aukštis virš jūros lygio pėdomis bei leidimosi kampas procentais.

Leidimosi kampas rodomas įprastai, jei yra ribose kampo, aprašyto procedūros saugumo reikalavimuose. Kitu atveju, leidimosi kampas paryškinamas raudonu fonu. Prototipe nustatytas leidžiamas leidimosi kampas yra 4,7–5,7%.

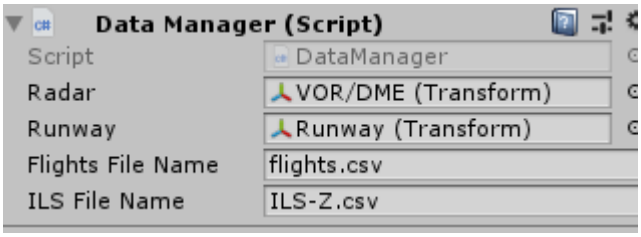
Priklausomai nuo skrydžių duomenų failo, naudotojas taip pat turi galimybę pakeisti stebimą skrydį simuliacijos vykdymo metu. Panelėje matomas skrydžio identifikatorius rodo tuo metu stebimą skrydį, tačiau, paspaudus ant jo, atsidaro sąrašas, kuriame naudotojas turi galimybę pasirinkti bet kurį atpažintą skrydį. Prototipo įgyvendinimo metu nuspręsta, kad toks funkcionalumas būtų privalomas potencialiems skrydžių vadovams, nes tikėtina, kad jų darbo metu prireiktų stebėti keletą skrydžių vienu metu.



26 pav. Baigtas prototipas. Lėktuvas artėja link Vilniaus oro uosto nusileidimo tako

Naujame prototipe išspręstos visos analizės metu pastebėtos problemos. Sistema yra iš esmės daug lankstesnė ir pajėgi prisitaikyti prie bet kokios vietoves, oro uosto ir jame naudojamų procedūrų. Daugiausiai dėmesio skirta „Data manager“ ir „Simulation manager“ komponentų funkcionalumui tobulinti.

Naujas „Data manager“ komponentas tikslingai skaito ir valdo duomenis prototipo veikimo metu. Duomenys skaitomi ir išsaugomi tik tuo atveju, kai jie yra pateikti tinkamu formatu CSV tipo faile. Naudotojas turi galimybę bet kuriuo metu pateikti naują koordinačių duomenų failą skrydžių trajektorijų ar procedūros vaizdavimui (žr. 27 pav.). Šie duomenų failų vardai nėra saugomi programiniame kode, taip išvengiamos problemos, kurios gali kilti dėl „kietojo programavimo“ atvejų kode.



27 pav. Laukai, kuriuose naudotojas gali nurodyti duomenų failus

Taip pat įgyvendintas naujas „Simulation manager“ funkcionalumas. Šiame komponente vykdomas patobulintas tūpimo ar kilimo procedūros ribų generavimas. Ribos vaizduojamos šešiakampiais tuneliais, kurie generuojami prototipo veikimo metu pagal naudotojo pateiktuose duomenų failuose esančias koordinates. ILS procedūrų ribų tikslios koordinatės nėra viešai prieinamos, tad šiame prototipe naudojamos koordinatės, matuotos pagal 3 pav. Vilniaus oro uosto „ILS-Z“ tūpimo procedūros instrumentines skrydžio taisykles.

Naudotojas turi galimybę tame pačiame oro uoste naudoti kitą tūpimo ar pakilimo procedūrą. Toks funkcionalumas reikalingas potencialiems skrydžių vadovams, kurie pilotams nurodo, kurios procedūros privaloma laikytis tūpimo ar pakilimo metu. Taikomos procedūros pakeitimas vykdomas pakeičiant pateikiamą procedūros ribų koordinačių duomenų failą kitu.

Rezultatai ir išvados

Pasiekti rezultatai:

1. Surinkta ir išanalizuota viešai prieinama aktuali informacija apie Vilniaus oro uostą ir jame naudojamas procedūras.

2. Apžvelgti aviacijoje naudojami skrydžių trajektorijų vizualizavimo būdai ir pasirinktas prototipui tinkamiausias.

3. Sėkmingai sumodeliuota Vilniaus oro uosto ir apylinkės vietovė Unity 3D aplinkoje.

4. Sukurtas naujas, patobulintas prototipas, skirtas Vilniaus uosto skrydžių valdymui.

5. Pademonstruotas modifikavimo procesas ir sukurtas lankstus prototipas, kurį galima pritaikyti bet kokiam oro uostui.

Sukurto prototipo programinis kodas ir visi prototipo naudojimui reikalingi failai bei duomenys patalpinti kodo versijų kontrolės sistemos „GitHub“ viešai prieinamoje saugykloje adresu https://github.com/Kajusn/Aircraft-trajectory-visualization.

Prieitos išvados:

1. Analizuojant Vilniaus oro uosto ir jo procedūrų informaciją buvo nustatyta, kuriuos aspektus bei duomenis reikia pateikti naudotojui prototipe. Nuspręsta, kad potencialiems skrydžių vadovams privalo būti pateikiamas tikslus orlaivio greitis, aukštis, leidimosi kampas, skrydžio identifikatorius bei normų (leidimosi kampo, procedūros ribų) pažeidimai. Taip pat nustatyta, kurie duomenys privalo būti naudojami trajektorijos vizualizacijos įgyvendinimui. Taip pat nuspręsta, kad prototipo demonstracijai tinkamiausia vaizduoti procedūros ribas pagal ILS-Z procedūroje matomą tūpimo maršrutą, nes šis yra gana sudėtingos formos. Tai suteikia galimybę pademonstruoti prototipe įgyvendintą procedūrų ribų generavimą.

2. Išnagrinėjus keletą galimų trajektorijos vizualizavimo būdų įvertinti jų privalumai bei trūkumai. Nustatyta, kad norint pasiekti geriausią prototipo efektyvumą ir išvaizdą, būtina derinti dvimates ir trimates vizualizacijas. Prototipe pasirinkta naudoti trimatį trajektorijos vizualizavimą tuneliu, o procedūrą vaizduoti trimačiu tuneliu bei dvimate tvora.

3. Ankstesnio prototipo analizės metu išskirtos keletas problemų, susijusių su „kietu programavimu“ (angl. *hard coding*) bei programų sistemos moduliškumu (angl. *modularity*). Trūkumai pastebėti duomenų valdymo, vietovės vaizdavimo ir simuliacijos valdymo komponentuose. Taip pat, remiantis šaltiniais apie programų sistemų moduliškumą ir „kietojo programavimo“ vengimą įrodyta, kad šie komponentai neatitinka gerųjų programų sistemų projektavimo praktikų. Taip pat, analizuojant ankstesnį prototipą, buvo nustatyta, kad būtini

patobulinimai, leidžiantys vizualizuoti skrydžio trajektoriją ir procedūrų ribas naudojant bet kokį naudotojo pateiktą duomenų rinkinį.

4. Naujo prototipo kūrimo metu buvo priimti programų sistemų projektavimo sprendimai, suteikiantys galimybę prototipą naudoti su bet kokiu oro uostu ir jame naudojamomis tūpimo ir pakilimo procedūromis. Tai buvo pasiekta išplėtus ir patobulinus vietovės vaizdavimo ir simuliacijos bei duomenų valdymo komponentų funkcionalumą. Patobulinimus sudarė naujas funkcionalumas, vartotojui suteikiantis galimybę pateikti bet kokius norimus duomenis skrydžio trajektorijos ir procedūrų ribų vizualizavimui. Be to, įgyvendintas funkcionalumas procedūrų ribų generavimui realiu laiku. Šis išplėtimas vartotojui suteikia galimybę pakeisti taikomą tūpimo ar pakilimo procedūrą bet kuriuo metu, taip patobulinant prototipo pritaikomumą kitiems oro uostams ir procedūroms.

Šaltiniai

[BTD15] S. Buschmann, M. Trapp, J. Dollner. Animated Visualization of Spatial-Temporal Trajectory Data for Air-Traffic Analysis, The Visual Computer, 2015, p. 1–8. Prieiga per internetą: https://www.researchgate.net/publication/285045621_Animated_visualization_of_spatial-temporal_trajectory_data_for_air-traffic_analysis. [Žiūrėta 2021-05-25]

[CNA09] Javier García-Heras Carretero, Francisco Javier Sáez Nieto, José Félix Alonso Alarcón. Criteria for Positioning Active Multilateration Stations Located Close to Distance Measuring Equipment. EUROCONTROL Headquarters, 2009, p. 1. Prieiga per internetą: https://www.researchgate.net/publication/258845417_Criteria_for_Positioning_Active_Multilateration_Stations_Located_Close_to_Distance_Measuring_Equipment. [Žiūrėta 2021-05-25]

[FAA17] Federal Aviation Administration. Instrument Procedures Handbook. U.S. Department of Transportation, 2017, p. 149–155. Prieiga per internetą: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/instrument_procedures_handbook/. [Žiūrėta 2021-05-25]

[GPS15] Pongpanote Gongsook, Janneke Peijnenborgh, Erik van der Spek, Jun Hu, Francesco Bellotti, Riccardo Berta, Alessandro de Gloria, Francesco Curatelli, Chiara Martinengo, Matthias Rauterberg, Jos Hendriksen. Designing a Serious Game as a Diagnostic Tool. Lecture Notes in Computer Science, vol. 9221, Springer, Cham, 2015, p. 63–66. Prieiga per internetą: https://doi.org/10.1007/978-3-319-22960-7_7. [Žiūrėta 2021-05-25]

[Lap10] Kristina Lapin. SKY-Scanner: time-critical decision support system surveilling aircraft landing and take-off. D. Schaefer (red.). 9th EUROCONTROL Innovative Research Workshop & Exhibition 2010. Curran Associates, Inc., Bretigny-sur-Orge, 2010, p. 19–26. Prieiga per internetą: http://web.vu.lt/mif/k.lapin/files/2013/12/INO2010Proc_Lapin.pdf. [Žiūrėta 2021-05-25]

[LČŠ12] Kristina Lapin, Vytautas Čyras, Laura Savičienė. The SKY-Scanner time-critical decision support system for surveillance and risk evaluation during landing and takeoff. Journal of Aerospace Operations, Faculty of Mathematics and Informatics, Vilnius University, 2012, p. 19–26. Prieiga per internetą: http://dx.doi.org/10.3233/AOP-2012-0016. [Žiūrėta 2021-05-25]

[LMZ21] Aurelia Tamo-Larrieux, Simon Mayer, Zaira Zihlmann. Not Hardcoding but Softcoding Privacy. Technology and Regulation, 2021, p. 7–10. Prieiga per internetą: https://www.alexandria.unisg.ch/publications/262254. [Žiūrėta 2021-05-25]

[Sav09] L. Savičienė. Žmogaus ir kompiuterio sąsajos projektavimas skrydžių valdymo sprendimų priėmimo sistemai. Informacijos mokslai, 50, 2009, p. 181-186. Prieiga per internetą: https://www.zurnalai.vu.lt/informacijos-mokslai/article/view/3241/2358. [Žiūrėta 2021-05-25]

[SGC01] Kevin J. Sullivan, William G. Griswold, Yuanfang Cai, Ben Hallen. The Structure and Value of Modularity in Software Design. Proceedings of the Joint International Conference on Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2001, p. 2–4. Prieiga per internetą: https://doi.org/10.1145/503271.503224. [Žiūrėta 2021-05-25]

[STM13] Case Study: Aviation Accident Litigation. Solid Terrain Modeling įmonės medžiaga, 2013, p. 1–2. Prieiga per internetą: https://www.solidterrainmodeling.com/pdf_brochures/litigation_case_study.pdf. [Žiūrėta 2021-05-25]

[Uni19] Unity. Unity User Manual (2019.4 LTS), 2018. Prieiga per internetą https://docs.unity3d.com/Manual/index.html. [Žiūrėta 2021-05-25]

[UniC18] Unity Community. Unity Community Unity3D wiki, 2018. Prieiga per internetą: http://wiki.unity3d.com/index.php/Main_Page. [Žiūrėta 2021-05-25]