



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Kasparas Kažemėkas

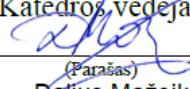
Procedūrinis pastatų generavimas naudojant UNITY 3D
Procedural Buildings Generation Using UNITY 3D

Baigiamasis bakalauro darbas

Programų sistemų inžinerijos studijų programa, valstybinis kodas 612I30003
Programų sistemų studijų kryptis

Vilnius, 2020

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

(Parašas)
Dalius Mažeika
(Vardas, pavardė)
2020-05-28
(Data)

Kasparas Kažemėkas

Procedūrinis pastatų generavimas naudojant UNITY 3D
Procedural Buildings Generation Using UNITY 3D

Baigiamasis bakalauro darbas

Programų sistemų inžinerijos studijų programa, valstybinis kodas 612I30003
Informatikos studijų kryptis

Vadovas Dr. Algirdas Laukaitis  2020-05-28
(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

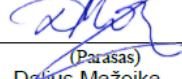
Konsultantas _____
(Pedag. vardas, vardas, pavardė) _____ (Parašas) _____ (Data)

Konsultantas _____
(Pedag. vardas, vardas, pavardė) _____ (Parašas) _____ (Data)

Vilnius, 2020

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

.....Informatikos.....studijų kryptis
....Programų inžinerija..studijų programa, valstybinis kodas 612I30003
.....specializacija

TVIRTINU
Katedros vedėjas

(Parasas)
Dalius Mažeika
(Vardas, pavardė)
2020-05-28
(Data)

BAIGIAMOJO BAKALAURO DARBO (PROJEKTO)

UŽDUOTIS

.....Nr.
Vilnius

Studentui (ei).....Kasparas Kažemėkas
(Vardas, pavardė)

Baigamojo darbo (projekto) tema:Procedūrinis pastatų generavimas naudojant UNITY 3D

.....Procedural Buildings Generation Using UNITY 3D
.....patvirtinta 201...m. d. dekano potvarkiu Nr.

Baigamojo darbo (projekto) užbaigimo terminas 2020 m. birželio 3 d.

BAIGIAMOJO DARBO (PROJEKTO) UŽDUOTIS:

Duomenys:

Aiškinamasis raštas:Darbo uždaviniai:
.....Apžvelgti rinkoje egzistuojančių pastatų generatorių galimybes;
.....Išnagrinėti populiariausius procedūrinių 3D objektų generavimo metodus;
.....Parengti sistemos kūrimui reikalingus funkcinius bei nefunkcinius reikalavimus;
.....Atlikti pastatų generatoriaus projektavimą;
.....Realizuoti programą, atlikti bandymus, suformuluoti išvadas;

Baigamojo bakalauro darbo (projekto) konsultantai:

.....(Pareigos, vardas, pavardė)

Vadovas
.....(Parašas)

.....Dr. Algirdas Laukaitis....
.....(Mokslo laipsnis, vardas, pavardė)

Užduotį gavau

.....
.....(Parašas)

.....
.....(Vardas, pavardė)

.....(Data)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių sistemų katedra

ISBN ISSN
Egz. sk.
Data-.....-

Pirmosios pakopos studijų **Programų inžinerijos** programos bakalauro baigiamasis darbas 3
Pavadinimas **Procedūrinis pastatų generavimas naudojant UNITY 3D**
Autorius **Kasparas Kažemėkas**
Vadovas **Algirdas Laukaitis**

Kalba: lietuvių

Anotacija

Bakalauro darbe projektuoojamas procedūrinis pastatų generatorius. Atlikus algoritmu bei panašių sistemų analizę buvo nuspresta taikytis į siaurą nišą, būtent mažo poligonų kiekių stilistikos viduramžių pastatus. Nustatyti pagrindiniai generuojamų pastatų elementai bei aprašyta koks algoritmas bus naudojamas generavimui. Aprašyti funkciniai bei nefunkciniai reikalavimai ir pasitelkus UML diagramas suprojektuota sistemos architektūra. Generatorius sukurtas, ištестuotas ir pateiktas bandymų rezultatai bei kodo pavyzdžiai. Darbos struktūra: įvadas, algoritmu ir analogiškų sistemų analizė, sistemos projektavimas, sistemos realizacija bei testavimas ir išvados bei pasiūlymai. Darbo apimtis: 54 puslapių be priedų; 35 iliustracijos; 13 lentelių; 13 literatūros šaltinių; Atskirai pridedami priedai.

Prasminiai žodžiai: Mažas poligonų kiekis, procedūrinis generavimas, algoritmai, procedūrinis poligonų tinklas

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Information Systems

ISBN ISSN
Copies No.
Date-.....-.....

Bachelor Degree Studies **Software Engineering** study programme Bachelor Graduation Thesis 3

Title **Procedural Buildings Generation Using UNITY 3D**

Author **Kasparas Kažemėkas**

Academic supervisor **Algirdas Laukaitis**

Thesis language: Lithuanian

Annotation

Bachelor thesis is devoted to the designing of a procedural buildings generator. After algorithms and similar systems analysis, it was decided to aim for a specific niche - low poly medieval style building generator. Work contains main building parameters, as well as the algorithm which was used to the generation itself. Functional and non-functional requirements were set and system architecture was defined using UML diagrams. Buildings generator was created, tested and summary was made. Some code examples are shown and analysed. Work consists out of 5 parts: introduction, algorithm and analogical systems analysis, system design, realisation and testing, summary an improvement suggestions. The thesis consists of: 54 pages without appendixes; 35 pictures; 13 tables; 13 references.

Keywords: low poly, procedural generation, algorithms, procedural mesh

SUTIKIMAS DĖL ASMENS DUOMENŲ NAUDOJIMO

2020-05-26

(Data)

Šiuo sutikimu aš, Kasparas Kažemėkas (toliau – Duomenų subjektas) sutinku, kad Vilniaus Gedimino technikos universitetas, juridinio asmens kodas 111950243, adresas Saulėtekio al. 11, LT-10223 Vilnius (toliau – Duomenų valdytojas), tvarkytu mano asmens duomenis kitų studentų mokymosi tikslu. T. y. tvarkytu (*pažymėkite tinkamą*):

- vardą, pavardę, bakalauro baigiamajį darbą;
- bakalauro baigiamajį darbą, nenurodant vardo, pavardės;
- vardą, pavardę, magistro baigiamajį darbą;
- magistro baigiamajį darbą nenurodant vardo, pavardės.

Šiuo tikslu tvarkomu asmens duomenų Duomenų valdytojas neperduos jokiems tretiesiems asmenims, studentams su baigiamaisiais darbais bus leidžiama susipažinti vidinėje informacinėje sistemoje. Duomenų subjekto asmens duomenys šiuo tikslu bus naudojami ne ilgiau nei 5 metai.

Šiuo sutikimu Duomenų subjektas patvirtina, kad yra supažindintas su šiomis teisėmis:

- Susipažinti su savo duomenimis ir kaip jie yra tvarkomi (teisė susipažinti);
- Reikalauti ištaisyti arba, atsižvelgiant į asmens duomenų tvarkymo tikslus papildyti asmens neišsamius asmens duomenis (teisė ištaisyti);
- Savo duomenis sunaikinti arba sustabdyti savo duomenų tvarkymo veiksmus (išskyrus saugojimą) (teisė sunaikinti ir teisė „būti pamirštam“);
- Reikalauti, kad asmens duomenų valdytojas apribotų asmens duomenų tvarkymą (teisė apriboti);
- Teise į duomenų perkėlimą (teisė perkelti);
- Nesutikti, kad būtų tvarkomi asmens duomenys, kai šie duomenys tvarkomi ar ketinami tvarkyti kitais tikslais;
- Pateikti skundą Valstybinei duomenų apsaugos inspekcijai;

Duomenų subjektas turi tėisę bet kuriuo metu atšaukti savo sutikimą.

Kasparas Kažemėkas 
[Duomenų subjekto vardas, pavardė, parašas]

Duomenų valdytojo rekvizitai:

Vilniaus Gedimino technikos universitetas

Juridinio asmens kodas: 111950243

Adresas: Saulėtekio al. 11, LT-10223 Vilnius

Tel. (8 5) 274 5030

Faks. (8 5) 270 0112

El. paštas: vgtu@vgtu.lt

PVM mokėtojo kodas: LT1119502413

Duomenų apsaugos pareigūno tel. (8 5) 251 2191, el. paštas: dap@vgtu.lt

Vilniaus Gedimino technikos universiteto
egzaminų sesijų ir baigiamųjų darbų
rengimo bei gynimo organizavimo tvarkos
aprašo
2 priedas

(Baigamojo darbo sažiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Kasparas Kažemėkas, 20174353

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Programų inžinerija, PRIf-16/1

(Studijų programa, akademinė grupė)

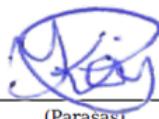
BAIGIAMOJO DARBO (PROJEKTO) SAŽININGUMO DEKLARACIJA

2020 m. gegužės 28 d.

Patvirtinu, kad mano baigiamasis darbas tema „Procedūrinis pastatų generavimas naudojant UNITY 3D“ patvirtintas 2019 m. lapkričio 15 d. dekano potvarkiu Nr. 401fm, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas daktaras Algirdas Laukaitis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių išstatymų nenumatyty piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).



(Parasas)

Kasparas Kažemėkas
(Vardas ir pavardė)

Turinys

1.	Ivadas	11
	Darbo tikslas ir uždaviniai	11
2.	Procedūrinio modelių generavimo technologijų analizė.....	13
2.1.	Viduramžių namo architektūrinė analizė	13
2.2.	Programų apžvalga.....	14
2.2.1.	Houdini Building Generator [5]	14
2.2.2.	Maya Structures [6].....	15
2.2.3.	Building Generator v0.7 [7]	16
2.2.4.	BuildR 2 [8]	16
2.2.5.	SceneCity [9].....	17
2.3.	Procedūrinio generavimo algoritmų apžvalga.....	18
2.3.1.	L-sistema ir jos plėtiniai.....	18
2.3.2.	Greuter metodas	19
2.3.3.	Momentinė architektūra	20
2.4.	Skyriaus išvados	21
3.	Projektinė dalis.....	22
3.1.	Reikalavimų specifikacija	22
3.1.1.	Formuluojamos užduotys.....	22
3.1.2.	Funkciniai sistemos reikalavimai.....	24
3.1.3.	Nefunkciniai reikalavimai.....	24
	Programų sistemos projektiniai reikalavimai.....	26
3.1.4.	Programų sistemos dekompozicija.....	26
3.1.5.	Reikalavimų lokalizavimo matrica	28
3.1.6.	Reikalavimų ryšio matrica	29
3.2.	Programų sistemos architektūra	31
3.2.1.	Sistemos užduočių scenarijų vykdymas.....	31
3.2.2.	PS struktūrinis modelis	37
3.2.3.	PS dinaminis modelis.....	40
3.3.	Programų sistemos maketai.....	41
4.	Testavimas ir įgyvendinimas	43
4.1.	Automatinis testavimas	43
4.1.1.	Stogo pozicijos apskaičiavimo testavimas kai stogas didesnis už viršutinį aukštą.....	43
4.1.2.	Pirmo aukšto pozicijos apskaičiavimo testavimas kai visos ugniasienės išjungtos	44

4.2. Rankinis testavimas	44
4.2.1. Testavimo scenarijai	44
4.2.2. Testavimo atvejai	45
4.2.3. Generacijos greičio tyrimas	46
4.2.4. Sukurtos sistemos veikimo ir vartotojo sąsajos demonstracija	47
4.2.5. Reikalavimo, kad sistema nereikalautų bazinio modelių rinkinio įgyvendinimas.....	50
5. Išvados ir pasiūlymai	54
Literatūra ir šaltiniai.....	55
PRIEDAI	56

Santrumpū ir terminų žodynas

Verteksas – 3D objekto taškas erdvėje, viršūnė.

Mesh – verteksų tinklas sudarantis 3D objekto formą

Poligonas – Plokštuma gaunama kelis verteksus sujungus kraštinėmis, paprastai tai stačiakampis

Modelis – 3D objektas.

LOD (level of detail) – modelio detalumas, kuo objektas toliau nuo stebėtojo, tuo LOD gali būti mažesnis.

Low poly/Mažo poligonų kiekio stilistika – 3D grafikos stilius kai kuriami smarkiai stilizuoti objektai, dėl estetinių ar techninių priežasčių, turintys salyginai mažą kiekį poligonų.

UI (user interface) – vartotojo sąsaja, vartotojo interfeisas.

Ugniasinė – pastato siena kurioje negali būti langų, skirta kad tarp vienas prie kito sustatyti pastatų neplistų gaisras.

Tekstūra – spalvinė 3D modelio informacija paprastai pateikiama kaip paveikslėlis turintis nuspalvintą pastato paviršiaus išklotinę arba spalvų paletę.

1. Įvadas

Žaidimų industrija 2019 metais gavo 152 milijardus dolerių pajamų, per metus paaugdama 9.6% [1]. Daugiau nei trečdalį pajamų atnešė mobiliems įrenginiams skirti žaidimai. Tai svarbu, kadangi kiekviename rinkos segmente naudojama skirtingo pajėgumo techninė įranga, o silpniausių įrenginių dominavimas parodo, kad daugelis žaidimų, lyginant su asmeniniams kompiuteriams skirtais žaidimais, yra grafiškai paprasti. Prie mobilių telefonų žaidimo grafinio paprastumo prisideda ir visose žaidimų platformose populiarus mažo poligonų kiekio stilistiką [2] (geri jo pavyzdžiai būtų „Monument Valley“ ir „Astroneer“). Žinoma grafinis paprastumas nereiškia, kad žaidimo aplinką lengvą sukurti, atvirkšciai, maži poligonų limitai reikalauja kokybiškos optimizacijos ir sunkiai palaikomo balanso tarp detalumo ir veikimo greičio.

Žaidimų pardavimo kaina jau dešimtmetį išlieka ta pati – \$60 [3], atsižvelgiant į infliaciją, akivaizdu, kad kasmet ta pati kaina duoda vis mažesnę vertę žaidimų kūrėjams, taigi jiems tenka, be kitų pelno auginimo priemonių, pastoviai optimizuoti žaidimų kūrimo procesą. Procedūrinis žaidimo aplinkos generavimas – vienas iš kaštų mažinimo būdų. Procedūriškai generuoti objektai gali būti naudojami greitam prototipavimui – kurti bazine žaidimo aplinkos išvaizdą, vėliau procedūrinius modelius pakeičiant kurtais rankomis. Tačiau vis tobulėjanti techninė įranga leidžia lengvai kurti ir kokybiškus galutiniame produkte naudojamus objektus ar net generuoti ištisą visatą su unikaliomis planetomis turinčiomis savo ekosistemas (No Man's Sky).

Paprastai procedūriname modelių generavime naudojami iš anksto paruošti bazinių modelių rinkiniai (tarkim durys, sienos, langai, stogo segmentai) turintys ir iš anksto sukurtas tekštūras. Rinkiniai panaudojami procedūriškai surinkti objektus. To pliusas, kad generuoti objektai gali būti geometriškai sudėtingi ir itin realistiški. Iš kitos pusės, bazinių modelių kūrimas reikalauja nemažai laiko, o juos sukūrus vėliau gali būti sudėtinga atligli pakeitimuis. Tokios generacijos pakaitalas – procedūrinis visų objekto dalių generavimas nuo pradžios iki pabaigos išskaitant ir tekštūras, ko ir bus siekiama kuriant šią sistemą.

Šiuo metu siūlomi pastatų ir miestų generatoriai paprastai orientuoti į realistinį šiuolaikinių pastatų generavimą bei universalumą. Akivaizdu, kad universalūs įrankiai priimtinai plačiam vartotojų ratui, o griežtos modernių pastatų formos leidžia nesunkiai, pasinaudojus tomis pačioms taisyklėmis, generuoti platų spektrą pastatų. Tačiau, nemažo skaičiaus žaidimų veiksmas vyksta viduramžių ar fantastinėse aplinkose, kur tokie generatoriai sunkiai pritaikomi.

Taigi, atsižvelgus į mobilių įrenginių dominavimą, mažo poligonų skaičiaus grafino stiliaus populiariumą ir į šiuolaikinę architektūrą orientuotus generatorius išskyta poreikis įrankiui leidžiančiam generuoti labiau organinius, stilizuotus viduramžių pastatus.

Darbo tikslas ir uždaviniai:

Darbo tikslas – apžvelgus egzistuojančių pastatų generatorių galimybes, bei išsiaiškinus galimus pastatų generavimo metodus ir jų taikymą, sukurti procedūrinio viduramžių pastatų generatoriaus prototipą.

Tyrimo objektas – automatiniai virtualių pastatų generavimo metodai.

Tyrimo problema – Pastatų generatorių, orientuotų į mažo poligonų kiekio bei viduramžių stilistiką, trūkumas.

Darbo uždaviniai:

1. Apžvelgti rinkoje egzistuojančių pastatų generatorių galimybes;
2. Išnagrinėti populiariausius procedūrinį 3D objektų generavimo metodus;
3. Parengti sistemos kūrimui reikalingus funkcinius bei nefunkcinius reikalavimus;
4. Atligli pastatų generatoriaus projektavimą;

5. Realizuoti programą, atlikti bandymus, suformuluoti išvadas;

Darbo struktūra:

- Antras skyrius – egzistuojančių pastatų generatorių apžvalga bei pastatų generavimo algoritmų analizė
- Trečias skyrius – projektinė dalis
- Ketvirtas skyrius – testavimas ir įgyvendinimas
- Penktas skyrius – išvados ir pasiūlymai

2. Procedūrinio modelių generavimo technologijų analizė

2.1. Viduramžių namo architektūrinė analizė

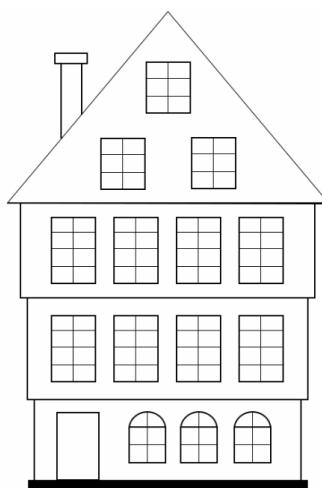
Kuriama sistema orientuosis į viduramžių namų generavimą, tad visų pirmiai reikia apibrėžti pagrindines tokį namų savybes. Kaip lengvai atpažįstamas pavyzdys, savotiškas viduramžių miesto pastato etalonas, pasirinktas vokiškas fachverko stiliaus namas (2.1 pav.).

- Matoma, kad siekiant kuo efektyviau išnaudoti mažą žemės sklypą, kiekvienas namo aukštasis būdavo statomas vis labiau išsikišęs į gatvę.
- Tokie namai turi stačius, šlaitinius, čerpių stogus, palėpėje sutalpinant po porą aukštų.
- Dominuoja aukšti, tankiai sudėti segmentuoti langai.
- Langai bei durys dviejų formų – arkiniai arba stačiakampiai
- Namai nesimetriškai, sienos kreivokos.
- Pirmo aukšto išvaizda skiriasi nuo sekancių aukštų.
- Vidutiniškai 5 aukštai – 3 pagrindiniai ir 2 palėpėje
- Namai turi kaminus
- Fasadai skirtinę spalvą bei detalės skirtinę spalvą



2.1 pav. Fachverkiniai namai [4]

Taigi, iš analizės galima sudaryti tokį tipinio generuojamo namo maketą:



2.2 pav. Generuojamo pastato planas

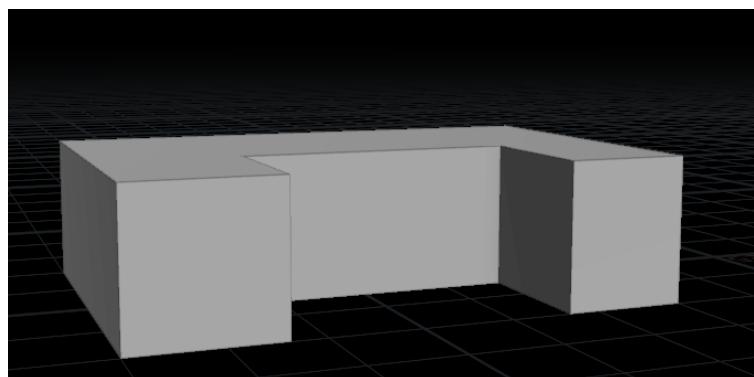
2.2. Programų apžvalga

Šiame poskyryje atliekama populiarų pastatų/miestų generatorių apžvalga. Kiekvienas įrankis vertinamas pagal jo licencijos tipą, valdymą ir generacijai naudojamus resursus. Taip pat pateikimas kiekvieno įrankio generacijos pavyzdys. Kiekvienas iš pristatomų įrankių veikia vis kitoje aplinkoje. Apžvelgiamų įrankių sugeneruotų pastatų pavyzdžius galima pamatyti prieduose.

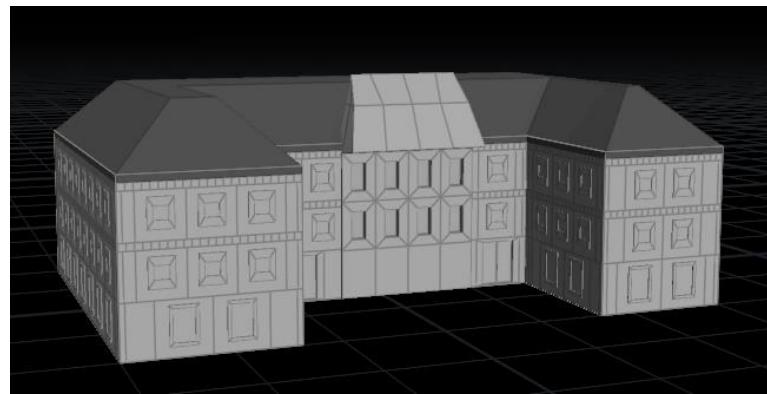
2.2.1. *Houdini Building Generator [5]*

2.1 lentelė. Houdini Building Generator apžvalga

Programa/aplinka kurioje veikia generatorius	Houdini
Naudojimo licencijos tipas	Patentuota licencija
Kaina	Nemokamas su Houdini, tačiau Houdini licencija kainuoja ~\$7000.
Programos valdymas	Valdymui naudojamas vizualus programavimas ir nustatymų langai.
Veikimo principas	Generacija vykdoma nustačius pastato tūri ir jį užpildant panaudojus iš anksto sukurtus komponentus (langus, duris ir t.t.). Komponentai dėliojami taip, kad būtų išlaikytas vieningas ir logiškas pastato stilius, pavyzdžiu durys tik pirmame aukšte ar visame pastato aukšte tos pačios stilistikos ir išmatavimų langai.
Panaudojimas	Filmų ir žaidimų aplinkos kūrimas, kompiuterinio meno kūrimas, prototipavimas.
Komentarai	Vizualinis programavimas leidžia ne tik gilią įrankio kontrolę, tačiau ir reikalauja minimalių programavimo žinių.



2.3 pav. Pradinis tūris [5]

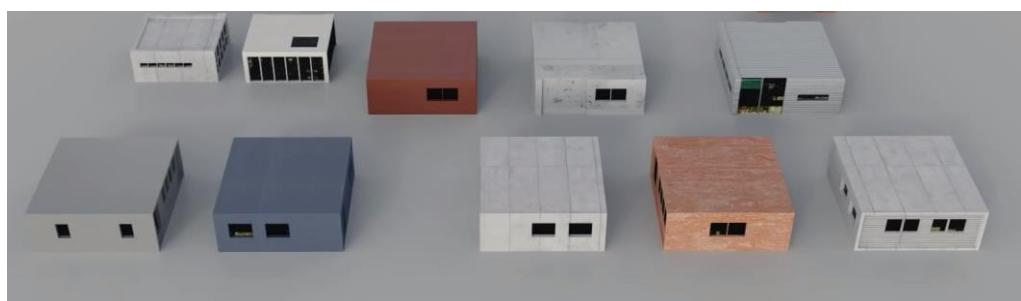


2.4 pav. Sugeneruoto pastato geometrija [5]

2.2.2. *Maya Structures* [6]

2.2 lentelė. Maya Structures apžvalga

Programa/aplinka kurioje veikia generatorius	3Ds Maya
Naudojimo licencijos tipas	Patentuota licencija
Kaina	~\$60
Programos valdymas	Valdymui naudojami tik nustatymų langai.
Veikimo principas	Generacija vykdoma panaudojant iš anksto sukurtų modelių rinkinį. Šie modeliai, skirtingai nei kitose pristatomose panašaus veikimo programose, yra ne smulkūs komponentai, kaip langai ar durys, o ištisi pastatų blogai (žr. 2.5 pav.). Programa keisdama modelių geometrines savybes (dydį, pasukimą) ir pozicijas sujungia juos į viena, taip sukurdama skirtingai atrodančius pastatus ir mechanizmus.
Panaudojimas	Žaidimų aplinkos kūrimas, kompiuterinio meno kūrimas, prototipavimas.
Komentarai	Generacija iš pastatų blokų lemia, kad generuojami tik dideli, modernūs ar futuristiniai pastatai ir mechanizmai, kadangi nenaudojant smulkų detalių prarandamas mažiemis ar senoviniams pastatams reikalingas detalumas. Valdymas nustatymų langais leidžia lengvai naudoti programą ir nereikalauja papildomų žinių, kaip vizualinių programavimą naudojantys įrankiai.



2.5 pav. Maya Structures generacijai naudojamų blokų pavyzdys [6]

2.2.3. Building Generator v0.7 [7]

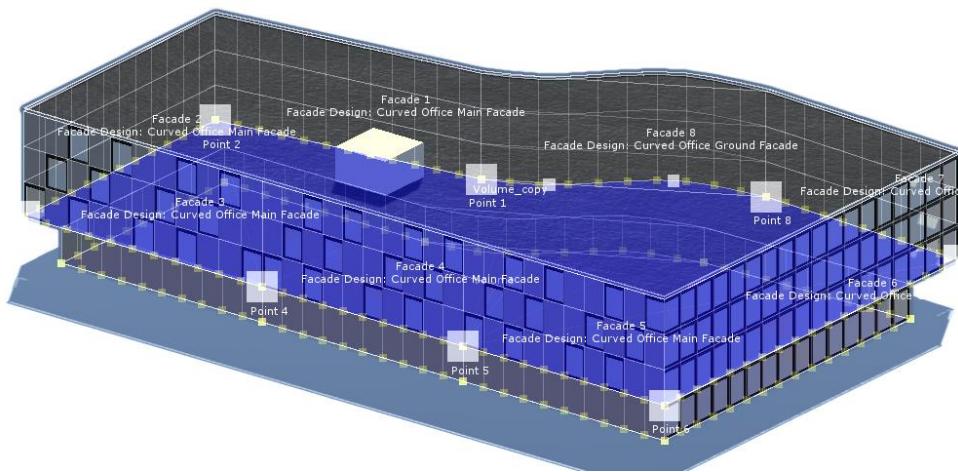
2.3 lentelė. Building Generator apžvalga

Programa/aplinka kurioje veikia generatorius	3Ds Max
Naudojimo licencijos tipas	Atviras Kodas
Kaina	Nemokamas
Programos valdymas	Valdymui naudojami nustatymų langai
Veikimo principas	Generacija, kaip ir Houdini generatoriuje, vykdoma panaudojant iš anksto sukurtą pastato komponentų (langų, durų ir t.t.) rinkinį. Tačiau skirtingai nei Houdini generatoriuje, šiame neįmanoma nustatyti ivarių formų pradinio tūrio, pastatas visuomet yra staciakampis. Programa keisdama modelių pozicijas ir pasukimo kampą sujungia juos į viena, taip sukurdama skirtingai atrodančius pastatus. Pastatą išvaizda kuriama nustatant įvairius parametrus, tokius kaip: aukštų skaičius, balkonų, langų išvaizda, pastato forma ir t.t.
Panaudojimas	Žaidimų aplinkų kūrimas, prototipavimas.

2.2.4. BuildR 2 [8]

2.4 lentelė. BuildR 2 apžvalga

Programa/aplinka kurioje veikia generatorius	Unity
Naudojimo licencijos tipas	Patentuota licencija
Kaina	€88
Programos valdymas	Valdymui naudojami nustatymų langai.
Veikimo principas	Įrankis leidžia kurti salyginai primityviai atrodančius pastatus, kadangi kūrimas susideda iš paprastų 3D modeliavimo operacijų – tokią kaip papildomų verteksų sukūrimas ar tūrio išstraukimas (angl. extract). Procedūrinis šio įrankio aspektas yra automatiškai atliekamas sumodeliuoto objekto skaidymas į segmentus: langus, duris, stogą, sienas, grindis ir jų pavertimas į atitinkamai atrodančius modelius.
Panaudojimas	Žaidimų aplinkų kūrimas, prototipavimas
Komentarai	Šis įrankis, dėl savo primityvumo, leidžia kurti tik minimalistinius modernius pastatus, kadangi beveik neįmanoma pridėti smulkį detalių. Objekto skaidymas į segmentus leidžia keisti kiekvieno elemento tekstūras atskirai.



2.6 pav. BuildR 2 generacijos pavyzdys su matomais verteksais

2.2.5. *SceneCity [9]*

2.5 lentelė. SceneCiy miestų generatoriaus apžvalga

Programa/aplinka kurioje veikia generatorius	Blender
Naudojimo licencijos tipas	Patentuota licencija
Kaina	\$97
Programos valdymas	Valdymui naudojamas vizualus programavimas ir nustatymų langai.
Veikimo principas	Įrankis leidžia generuoti logiškai išdėstyta kelių tinklą, realistiškai atrodantį žemės paviršių ir išdėlioti tūkstančius pastatų. Įrankis gaunamas su paruoštais pastatų pavyzdžiais, tačiau palaiko ir naudotojo sukurtus pastatus. Pastatai išdėliojami keičiant tik jų pasukimą, bet neliečiant jų formos. Skirtingai nei kiti apžvelgiami įrankiai, šis neskirtas pilnai procedūriškai generuoti pastatus. Procedūrinio pastatų generavimo galimybės apsiriboja paprastais iš stačiakampių sudėliotais tūriais. Tačiau, siekiant, kad sugeneruoto miesto pastatai neatrodytų vienodi, kiekvienam procedūriniam pastatui sugeneruojamos unikalios tekstūros. Tai pasiekama „sukarpant“ ir „suklijuojant“ segmentus iš anksto paruoštų tekstūrų failų su dideliu kiekiu įvairiai atrodančiu elementų, pavyzdžiui langų, durų, stogų ir t.t.
Panaudojimas	Žaidimų aplinkų kūrimas, kompiuterinio meno kūrimas, prototipavimas
Komentarai	Įrankis turi Unity žaidimų variklio papildinį leidžiantį sugeneruotus miestus lengvai panaudoti žaidimų kūrime.

2.3. Procedūrinio generavimo algoritmų apžvalga

2.3.1. L-sistema ir jos plėtiniai

L-sistema tai septintame praejusio amžiaus dešimtmetyje vengrų biologo Aristrid Lindenmayerio sukurtas rekursinis modelis aprašantis augalų vystymąsi. Dėl sistemos paprastumo ir universalumo ji prigijo ir matematikai. Sistema tai formaliai kalba kuri yra aprašoma kaip eilė simboliumi, kurių kiekvienas reiškia konkrečią komandą, taip pat sistemoje naudojama eilė parametru [10]. Kad būtų pasiektas rezultatas, sistema iteruoja per simbolius nustatyta skaičių kartą, kiekvienos iteracijos rezultatui taikydama tas pačias pradžioje aprašytas komandas bei pasinaudodamas duodamais parametrais. Pavyzdys [11]:

2.6 lentelė. Simboliu reikšmės

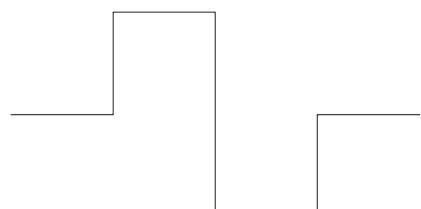
Simbolis	Reikšmė
F	pajudėti per linijos ilgį ir nupiešti liniją
-	Pasisukti į kairę per nurodytą kampą
+	Pasisukti į dešinę per nurodytą kampą

$$\text{Aksioma} = F+F+F+F$$

$$F \rightarrow F+F-F-FF+F+F-F$$

$$\text{Kampus} = 90^\circ$$

Pradinė šio pavyzdžio simboliu eilutė(aksioma) duoda kvadratą, o pakaitinė eilutė grafiškai pavaizduota 2.7 pav. Po vienos iteracijos gaunamas 2.8 pav. vaizdas.



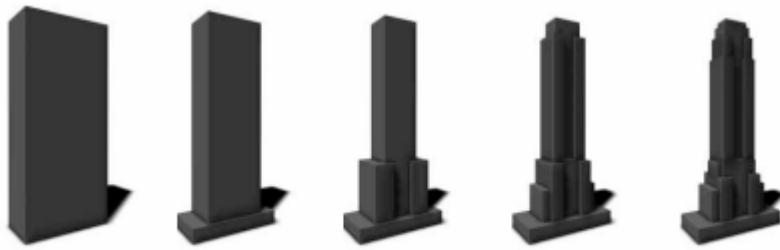
2.7 pav. $F \rightarrow F+F-F-FF+F+F-F$ grafinis vaizdas [11]

Šis originaliai augalų vystymuisi ir fraktalų aprašymams naudotas modelis gali būti panaudotas ir pastatų generacijai. Pascalis Mülleris bei Yoav I H Parishas savo publikacijoje „Procedural Modeling of Cities“ [10] šiam tikslui ir naudoja stochastinę, parametrinę L-Sistemą. Jų L-Sistemos objektais tai 3D objektų transformacijos operacijos, tokios kaip dydžio keitimas ar pajudinimas, tūrio ištraukimas (angl. extrusion), išsišakojimas ir panaikinimas bei sudėtingesniems architektūriniams elementams, tokiemis kaip stogai, langai ar antenos iš anksto paruošti modeliai.

Tokios sistemos generacijos pavyzdys matomas 2.9 pav. Kaip matome, šis generacijos būdas taip pat leidžia nesunkiai optimizuoti gaunamus modelius – kiekvienas iteracijos žingsnis gali būti panaudojamas kaip vis detalesnio LOD pastato modelis.

Šios sistemos išeiga – simboliu eilutė, nusiunciama į interpretatorių kuris ją paverčia į 3D pastatą;

2.8 pav. Vienos iteracijos rezultatas [11]



2.9 pav. Pastatų generacija su L-sistema kur aksioma yra pastato maksimalus tūris, o kiekvienas paveikslėliukas yra vienas papildomas iteracijos žingsnis [10]

2.3.2. Greuter metodas

Stefanas Greuteris, Jeremy Parkeris, Nigelis Stewartas ir Geoffas Leachas publikavo tyrimą aprašantį kitokio pobūdžio nei L-sistemos pastatų generaciją.

Šiame metode pastatai kuriami pasinaudojant pseudo atsitiktinių skaičių generatoriumi – generatorius sugeneruoja „atsitiktinę“ skaičių eilutę panaudodamas iš vartotojo gautą raktą. Tas pats raktas visada sugeneruos tokius pačius skaičius. Skaičiai nusako įvairiausio pastato charakteristikas: aukštį, aukštų skaičių ir t.t. [12]. Kadangi jų tyryme buvo kuriamas nesibaigiančio miesto generatorius, tai raktas buvo pastato koordinatė, kas leido visuomet toje pačioje vietoje sukurti/atkurti tokį patį pastatą.

Kiekvienas pastato aukštas sudaromas iš 2D grindų plano. Grindų planas tai įvairių formų poligonai kurių išvaizdą ir vietą nusako skaičių generatoriaus išeiga (grindų ploto generacija matoma 2.10 pav.). Atlirkus nustatyta vieno aukšto grindų plano keitimo iteracijų kiekį, grindų plokštuma yra ištraukiama paverčiant ją 3D tūriu, taip baigiant vieno aukšto generaciją [12]. Aukštai generuojami tol, kol pasiekiamos nustatytas pastato aukštis (aukštų generacija 2.11 pav.).

Kad geriau iliustruoti algoritmo veikimą, pateikiu pseudo kodą:

```

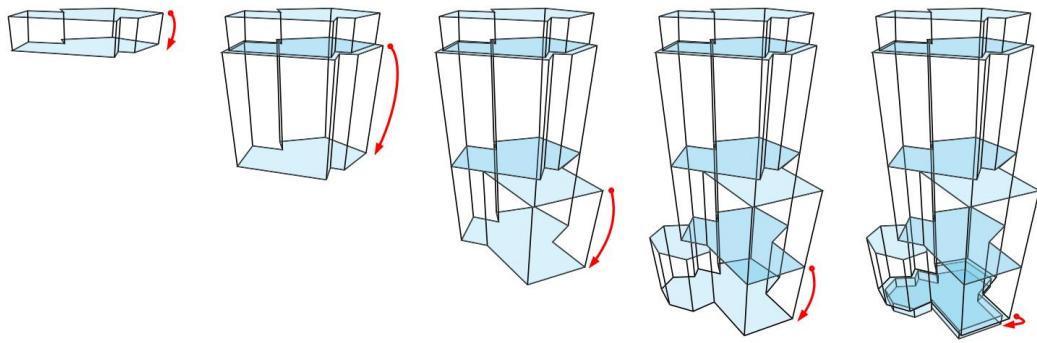
var pastatas;
ciklas(aukštų kiekis) {
    var grinduPlanas;

    ciklas(iteracijų skaičius aukšte)
    {
        grinduPlanas = PridetiFigura(grinduPlanas);
    }
    pastatas += IstrauktiTuri(grinduPlanas);
}

```



2.10 pav. Grindų plano generacija [12]

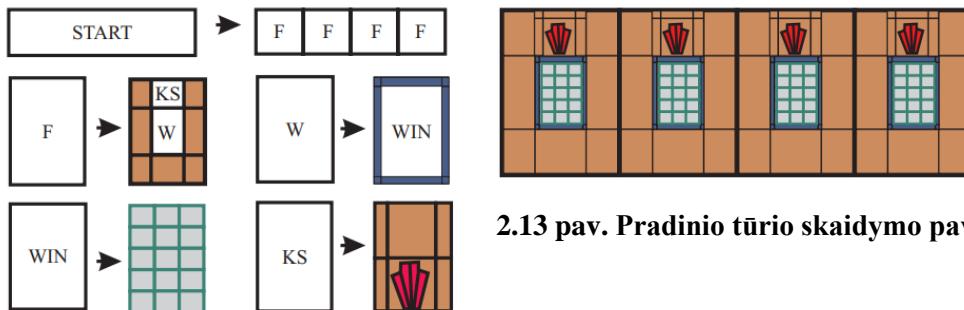


2.11 pav. Pastato aukštų generacija [12]

2.3.3. Momentinė architektūra

Kaip ir L-sistema, tai dar viena formalia kalba (gramatika), aprašoma sistema, tačiau ji paremta ne simboliais, o geometrinėmis figūromis ir vadinama skaidymo gramatika. Šios gramatikos principas – figūra suskaidoma ir jos segmentai pakeičiami kitomis figūromis kurios gali būti toliau skaidomos ir keičiamos kol pasiekianta kiekvienos figūros galutinė (nebeskaidoma) būseną. Sistema palaiko skirtinges architektūrinius stilius ir pastatų generaciją iš įvairiai sudėliotų stačiakampių, prizmių ir cilindrų. [13]

Pastatas paprastai sudeda iš kelių pradinių figūrų, pavyzdžiu 2 aukštai po 4 stačiakampius. Tuomet pasitelkė 2.12 paveiksle matomą antro aukšto generacijos vizualizaciją, matome kaip pradinis vieno stačiakampio plotas (F) skaidomas į tinklą kuriame išskiriamos plotai langui (W) ir dekoracijai (KS). Tuomet W plotas skaidomas į rėmą ir patį langą stiklą, kas yra galutinės šių figūrų būsenos. KS parenkama stilių atitinkanti dekoracija ir taip pasiekianta šios dalis galutinę būseną. 2.13 paveikslas parodo gaunamą antro aukšto rezultatą.



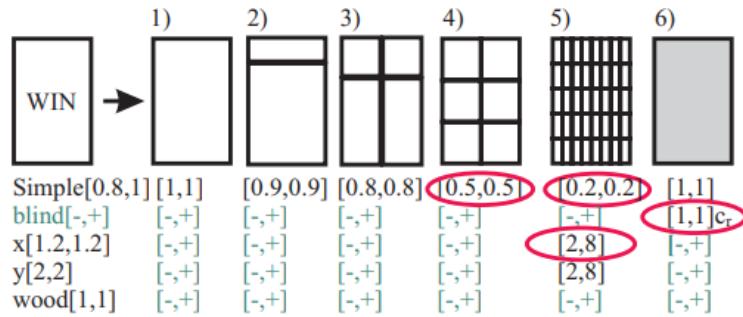
2.13 pav. Pradinio tūrio skaidymo pavyzdžio rezultatas [13]

2.12 pav. Pradinio tūrio segmento skaidymo pavyzdys [13]

Natūralu, kad generacija būtų sėkminga, algoritmas turi išlaikyti vieningą pastato stilių ir logiškai išdėlioti architektūrinius elementus (duris, langus, stogą ir t.t.). Algoritmas turi sėkmingai atsirinkti taisykles iš visų tai būsenai pritaikomų taisyklių rinkinio. Tai pasiekianta prie gramatikos taisyklių prijungiant parametrus. Parametrai gali būti žemo lygio informacija – sienų spalva, arba aukšto lygio – pastato stilius. Aukšto lygio parametrai kuriant sistemą paprastai užpildomi ranka, žemo lygio apskaičiuojami automatiškai. Už parametru perdavimą, tarp to pačio aukšto ar viso statinio segmentų, atsakingos kontrolinės gramatikos taisyklių. Jų iškvietimas reguliuojamas specifinių parametru prijungtų prie kiekvieno tūrio. Prieš kiekvieną tūrio skaidymą iškviečiamos kontrolinės taisyklių kurios padalina tėvinio tūrio parametrus visoms joms priklausančios erdvės (pvz. vienam aukštui) naujai suformuotoms figūroms (pvz. to aukšto langams).

Panagrinėkime konkretnų objektų atrankos pavyzdį (2.14 pav.). Kaip matome yra išskirtas naujas plotas langui (WIN) su 5 parametrais (simple, blind, x, y, wood). Langams egzistuoja 6 skirtinges dizainai.

„Simple“ parametras turi intervalą $[0.8, 1]$ kuris leidžia atmesti 4 ir 5 dizainus. Blind parametras su intervalu $[-\infty, \infty]$, leidžia atmesti 6 dizainų. X pakartotinai atmesta 5 dizainą. Taigi lieka 3 lango variantai, iš kurių atsitiktinai galima pasiimti bet kurį. Taigi tokia pastato segmentų išvaizdos atranka užtikrina tiek sąlyginai atsitiktinę pastato išvaizdą, tiek taisykles atitinkantį stilių. (Šiuo algoritmu sugeneruotas pastatus galima pamatyti prieduose.)



2.14 pav. Segemento išvaizdos atmetimas pasinaudojant taisyklėmis (gramatika) [13]

2.4. Skyriaus išvados

Apžvelgus dabartinę žaidimų rinkos situaciją, egzistuojančias procedūrinio pastatų generavimo programas bei algoritmus galima daryti keletą išvadų. Didelis mobilių žaidimų populiarumas reikalauja įrankių leidžiančių kurti juose naudojamus grafinius objektus. Natūralu, kad procedūrinė generacija tam tinka, kadangi veikia nepalyginti greičiau, bei gali pasiekti artimą ar net tokią pačią vizualinę kokybę kaip rankinis modeliavimas.

Procedūrinio generavimo tinkamumą žaidimų kūrimui rodo ir platus programų bei įrankių pasirinkimas įvairiomis platformomis. Tiesa, visi šie įrankiai stengiasi būti kuo universalesni ir orientuojasi į modernių, realistiškų pastatų generavimą. Atsižvelgus į minėta mažo poligonų kiekio stilistikos populiarumą, matosi, kad apžvelgti įrankiai neužpilda šio rinkos segmento. Taip pat, šie įrankiai naudotojo reikalauja išankstinio bazinių dalių rinkinio, kas su modeliavimu nesusipažinusiam galiapti neįveikiama kliūtimi.

Taigi, šiame darbe pristatoma ir kuriama sistema bus orientuota atlanti konkretų vaidmenį – generuoti mažo poligonų kiekio stilistikos viduramžių pastatus, nenaudodama jokių iš anksto sukurtų modelių rinkinio ir nereikalaudama rankinio tekštūravimo.

3. Projektinė dalis

3.1. Reikalavimų specifikacija

3.1.1. Formuluojamos užduotys

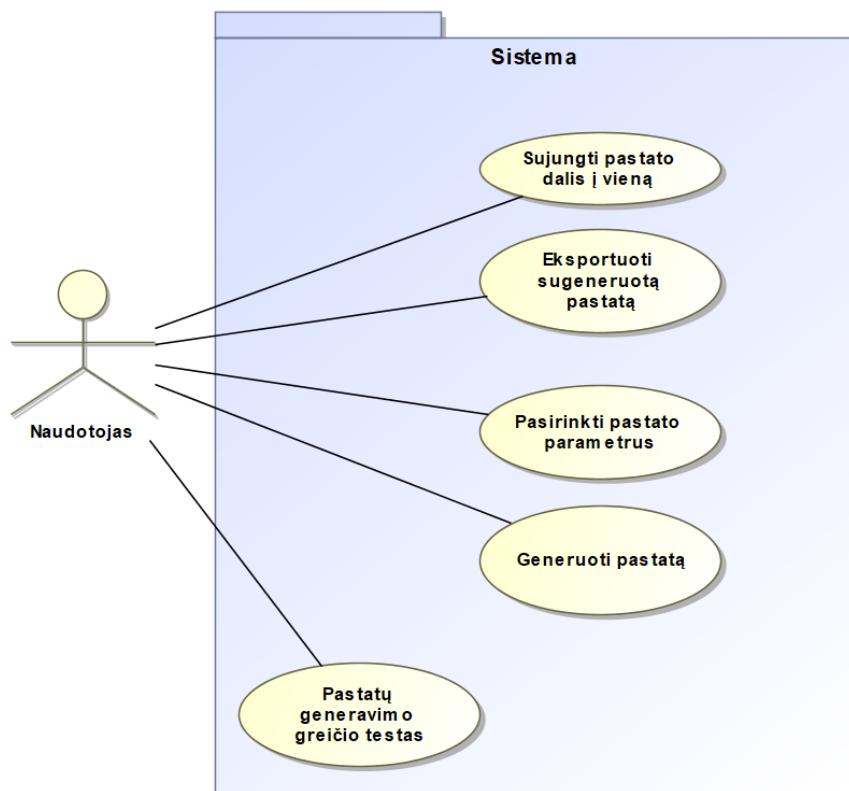
3.1.1.1. Pagrindinės užduotys

Pagrindinės sistemos atliekamos užduotys:

- 3.1.1.1.1. Pastato generavimas – procedūriškai sugeneruojamas pastatas nenaudojant jokio fizinio pradinių objektų rinkinio. Visa pastato geometrija pilnai generuojama kodu, vienintelė iš anksto pateikiamą dalį – tekštūros failas.
- 3.1.1.1.2. Sujungti pastato dalis į vieną – surenkami visų pastato segmentų poligonų tinklai (angl. mesh) ir sujungiami į vieną objektą. Gauto objekto geometrija nesikeičia, bet visi anksčiau atskiri objektai traktuojami kaip vienas;
- 3.1.1.1.3. Sugeneruoto pastato eksportavimas – galima eksportuoti bet kokį 3D objektą taip, kad jį būtų galima atsidaryti kitose 3D modelių palaikančiose programose;
- 3.1.1.1.4. Generavimo parametrų nustatymas – galima keisti daugelį pastato parametrų, kas leidžia sumažinti arba padidinti atsitiktinumo kiekį sugeneruotame pastate;

Pagalbinės užduotys:

- 3.1.1.1.5. Pastatų generavimo greičio testas – skirtas generuoti didelį kiekį pastatų skaičiuojant generacijos trukmę.



3.1 pav. Pagrindinės sistemos užduotys

3.1.1.2. Užduočių formulavimo kalbos reikalavimai

Vartotojo sąsaja yra Unity Editor plėtinys.

Vartotojo sąsaja susideda iš:

1. Nustatymų lango, kuriame yra sudėtas visas vartotojo kontroliuojamas funkcionalumas;
2. Unity scenos lango, kuriame vaizduojamas generacijos rezultatas;
3. Unity konsolės, kurioje rodomi pranešimai susiję su sistemos darbu;

Nustatymų lango sudėtis:

1. Mygtukai įjungti/išjungti ugniasienes
2. Mygtukas keisti langų apšvietimo išdėliojimą
3. Mygtukas įjungti/išjungti kampines kolonas
4. Mygtukas keisti aukštų stilių
5. Mygtukas įjungti naudotojo nustatyto pastato dydžio naudojimą
6. Slankiojantys intervalai nustatyti pastato dydžiui
7. Minimalaus/maksimalaus aukštų skaičiaus nustatymo slankiojantis intervalas
8. Mygtukas sujungti generuoto pastato poligonų tinklus (angl. mesh) į vieną
9. Mygtukas pradėti generavimą
10. Mygtukas pradėti generavimo testą
11. Sąrašas pasirinkti langų ir durų formą
12. Laukelis įvesti eksportuojamo objekto saugojimo vietą
13. Laukelis įkelti tekštūrai

3.1.1.3. Interfeiso darnos ir standartizavimo reikalavimai

Vartotojo sąsaja (nustatymų langas) turi tenkinti „MS Windows“ standartus.

Interfeisas kuriamas anglų kalba.

3.1.1.4. Pranešimų formulavimo reikalavimai

Pranešimai sistemoje skirtomi į dvi kategorijas: informacinius ir klaidos. Pranešimai formuluojami anglų kalba. Pavyzdžiai:

- Informacino: „Building generation done. Duration 15ms“.
- Klaidos: „Building must be atleast 1 stories high“.

Kadangi įrankis yra Unity sistemos plėtinys, tai konsolė atvaizduoja tiek sistemoje aprašytus, tiek pačio Unity sukurtus pranešimus.

3.1.1.5. Interfeiso individualizavimo reikalavimai

- Ugniasienės – leidžia pažymeti kurie pastato šonai bus ugniasienės.
- Šviečiančių langų išdėliojimo stilis – galima pasirinkti, kad kiekvienas aukštas turėtų vienodai šviečiančius langus, arba atsitiktinai apšviestus langus.
- Kampinių kolonų būsena – pastatas gali arba turėti kolonas šonuose, arba ne.
- Aukštų dydis – aukštai gali būti vienodi arba platėjantys į viršų.
- Norimas pastato dydis – galima pasirinkti apatinio aukšto dydi.
- Aukštų skaičius - galima pasirinkti kelių aukštų pastatas bus generuojamas.
- Angų forma – galima pasirinkti durų bei langų formą arba leisti atsitiktinį parinkimą.

3.1.2. Funkciniai sistemos reikalavimai

Funkciniai reikalavimai tiesiog išvesti iš pagrindinių užduočių.

3.1.2.1. Dalykiniai reikalavimai

3.1 lentelė. Dalykiniai sistemos reikalavimai

RNr. 1	Statusas - E	Galiojimo laikas – S	Kritiškumo laipsnis – S	Užduoties Nr. 3.1.1.1.1
Sistema turi generuoti 3D pastato modelį				
RNr. 2	Statusas - D	Galiojimo laikas – S	Kritiškumo laipsnis – L	Užduoties Nr. 3.1.1.1.2
Sistema turi sujungti sugeneruoto pastato dalis į vieną objektą.				
RNr. 3	Statusas - D	Galiojimo laikas – S	Kritiškumo laipsnis – A	Užduoties Nr. 3.1.1.1.3
Sistema turi eksportuoti sugeneruotą pastatą universaliu formatu				
RNr. 4	Statusas – E	Galiojimo laikas – S	Kritiškumo laipsnis – S	Užduoties Nr. 3.1.1.1.4
Sistema turi leisti pasirinkti parametrus pagal kuriuos bus generuojamas pastatas				

3.1.2.2. Pagalbinės sistemos funkcijos

3.2 lentelė. Pagalbinės sistemos užduotys

RNr. 5	Statusas - O	Galiojimo laikas – U	Kritiškumo laipsnis – L	Užduoties Nr. 3.1.1.5
Sistema turi turėti didelio kiekiečio pastatų generavimo greičio testą				

3.1.3. Nefunkciniai reikalavimai

3.1.3.1. Vidinio interfeiso reikalavimai

3.1.3.1.1. Operacinės sistemos naudojimo reikalavimai:

3.1.3.1.1.1. Sistemai kurti turi būti naudojama Windows operacinė sistema;

3.1.3.1.2. Sąveikos su duomenų bazėmis reikalavimai:

Reikalavimų nėra

3.1.3.1.3. Dokumentų mainų reikalavimai:

Reikalavimų nėra

3.1.3.1.4. Darbo kompiuterių tinkluose reikalavimai:

Reikalavimų nėra

3.1.3.1.5. Sąveikos su kitomis programomis reikalavimai:

Reikalavimų nėra

3.1.3.1.6. Programavimo aplinkos reikalavimai:

3.1.3.1.6.1. Sistema kuriama Unity 2019.2 aplinkoje naudojant Visual Studio programavimo aplinką;

3.1.3.2. *Veikimo reikalavimai*

3.1.3.2.1. Tikslumo reikalavimai:

3.1.3.2.1.1. Ilgio matai – metrai ir centimetrai;

3.1.3.2.1.2. Nustatomo namo pagrindas turi būti tarp 2x2m ir 10x10m dydžio;

3.1.3.2.1.3. Namas gali turėti tarp 1 ir 6 aukštų;

3.1.3.2.1.4. Namas gali turėti iki 3 ugniasienių;

3.1.3.2.1.5. Pastato segmentas gali būti spalvojamas ne daugiau nei dviem spalvom;

3.1.3.2.2. Patikimumo reikalavimai:

3.1.3.2.2.1. Sistema turi pilnai veikti 95% laiko be trykių.;

3.1.3.2.3. Robastiškumo reikalavimai:

3.1.3.2.3.1.1. Trykius iššaukiančių įvykių procentas negali viršyti 1%.;

3.1.3.2.4. Našumo reikalavimai.

3.1.3.2.4.1. Reakcijos laikas:

3.1.3.2.4.1.1. Vidutinis pastato generavimo laikas neturi viršyti 100ms;

3.1.3.2.4.2. Pralaidumas (throughput):

3.1.3.2.4.2.1. Sistema vienu metu generuoja viena pastatą;

3.1.3.2.4.3. Masto keitimasis (scalability):

3.1.3.2.4.3.1. Sistema kuriama taip, kad norint būtų galima padidinti leidžiamą pastato pagrindo dydžio limitą bei pastato aukštų limitą;

3.1.3.3. *Diegimo reikalavimai*

3.1.3.3.1. Instaliuojamumas

3.1.3.3.1.1. Sistema turi būti instaliuojama per porą minučių, įkeliant reikalingus failus į Unity projektą;

3.1.3.3.2. Įsisavinamumas:

3.1.3.3.2.1. Vartotojas, kuris moka naudotis bazinėmis Unity funkcijomis, turi galėti įsisavinti visą programą ne ilgiau nei per 5 minutes.

3.1.3.3.3. Išmokstamumas

3.1.3.3.3.1. Sistema turi turėti ne daugiau nei vieną langą tai sistema naudotis išmokstama per keletą minučių

3.1.3.4. Ruošinio reikalavimai

3.1.3.4.1. Sistema negali reikalauti naudotojo bazinio generacijai naudojamo objektų rinkinio, jis turi būti iš anksto realizuotas naudojamos sistemos kode.

3.1.3.5. Aptarnavimo ir priežiūros reikalavimai

3.1.3.5.1. Taisomumo

3.1.3.5.1.1. Trykio ištaisymas turėtų užimti ne ilgiau nei 5min.

3.1.3.5.2. Keičiamumo

3.1.3.5.2.1. Kadangi sistema labai paprasta, turėtų užimti ne ilgiau nei viena dieną;

3.1.3.5.3. Plečiamumo

Reikalavimų nėra

3.1.3.5.4. Perkeliamumo

3.1.3.5.4.1. Sistema kuriama taip kad visi veikimui reikalingi failai būtų viename aplanke ir norint perkelti sistemą naudojimui kitame įrenginyje užtektų įkelti aplanką į norimą Unity projektą;

3.1.3.5.5. Testuojamumo

3.1.3.5.5.1. Sistemos testų kūrimui neturi būti skiriamos daugiau savaitės

3.1.3.6. Tiražuojamumo reikalavimai

Reikalavimų nėra

3.1.3.7. Apsaugos reikalavimai

Reikalavimų nėra

3.1.3.8. Juridiniai reikalavimai

L.R. asmens duomenų teisinės apsaugos įstatymas: 6. Straipsnis - sistemoje nebus saugomi nei vartotojų, nei darbuotojų asmeniniai duomenys todėl jokias atvejais nebus pažeidžiami individu duomenų apsaugos įstatymai ar kiti LR teisės aktai.

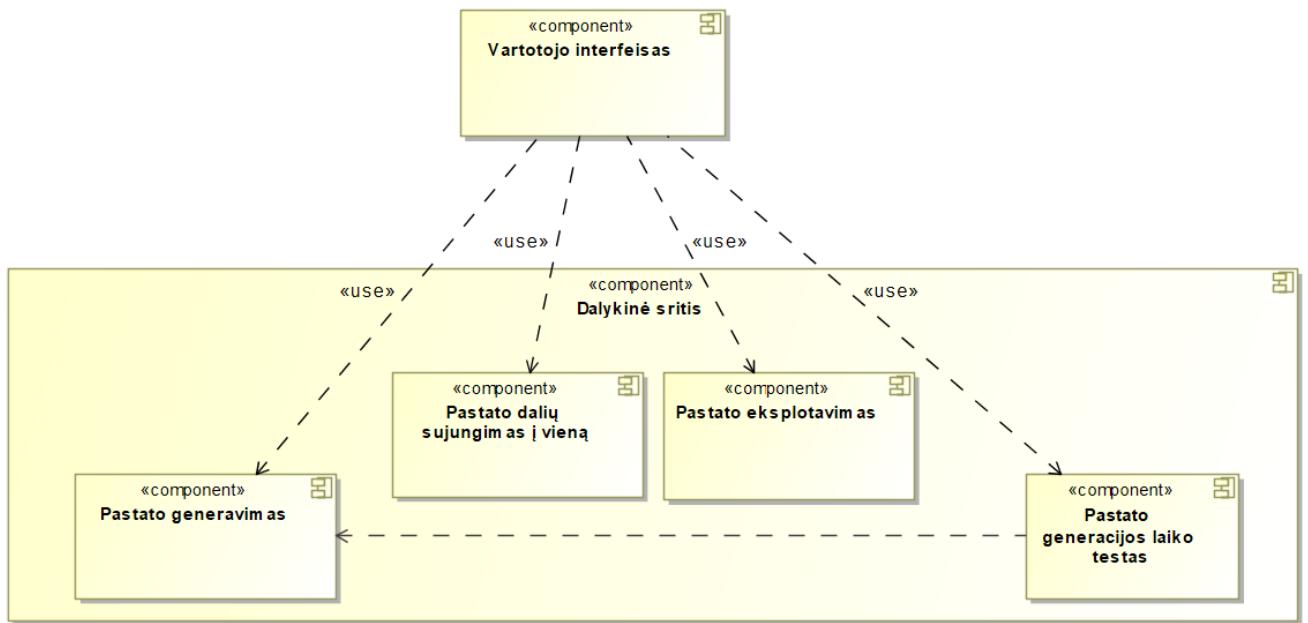
Programų sistemos projektiniai reikalavimai

3.1.4. Programų sistemos dekompozicija

3.1.4.1. Skaidymas į paketus

Sistema skaidoma į du paketus:

- Interfeiso
- Dalykinį



3.2 pav. Sistemos dekompozicija

Pradiniai, funkciniai, reikalavimai patikslinami juos suskaidant į mažesnius reikalavimus.

Interfeiso paketo reikalavimai

RNr. 4 Pastato generavimo parametrų valdymas:

- 3.1.4.2. Interfeisas turi būti įgyvendintas laikantis MS Windows standartų;
- 3.1.4.3. Interfeisas turi veikti Unity 2019.2 aplinkoje ir panaudoja jos teikiamus langus rodyti generacijos rezultatui ir pranešimam;
- 3.1.4.4. Sukurtas interfeisas susideda iš nustatymų lango;
- 3.1.4.5. Interfeise yra pastato generacijos nustatymus leidžiantys keisti UI elementai;
- 3.1.4.6. Interfeise yra visas sistemos užduotis leidžiantys įgyvendinti mygtukai;
- 3.1.4.7. Sugeneruotas pastatas turi būti matomas vartotojo sąsajoje;
- 3.1.4.8. Interfeise turi būti vieta įkelti generacijoje naudojamą tekstūrą;

Dalykinio paketo reikalavimai

Dekomponuojama į:

RNr. 1 Pastato generavimas:

- 3.1.4.9. Turi būti įmanoma generuoti pastatą su ugniasienėmis norimose namo pusėse;
- 3.1.4.10. Turi būti įmanoma generuoti pastatą su visiškai atsitiktinai apšviestais langais arba apšviestu ištisu aukštu;
- 3.1.4.11. Turi būti įmanoma generuoti pastatą su norimu skaičiumi aukštų;
- 3.1.4.12. Turi būti įmanoma generuoti pastatą su iš anksto nustatytu pamatų dydžiu;
- 3.1.4.13. Turi būti įmanoma generuoti pastatą kurio aukštai vienodo dydžio arba kiekvienas vis didesnis už buvusį;
- 3.1.4.14. Turi būti įmanoma generuoti pastatą su angomis kurios yra tik arkinės, tik stačiakampės arba atsitiktinai vieno iš šių stilių;
- 3.1.4.15. Turi būti įmanoma generuoti pastatą kurio kampuose, jei pasirinkta, būtų kolonos
- 3.1.4.16. Pastatas turi susidėti iš tarpusavyje nesujungtų nedidelių dalių, tokų kaip lango rėmai, stiklas, etc.

- 3.1.4.17. Generuojamo pastato dalis neturi turėti daugiau poligonų ir verteksų nei jų reikia perteikti norimą formą.
- 3.1.4.18. Pastato segmentai generuojami primityvius 3 dimensijų objektus (stačiakampis, plokštuma) išlankstant norima forma - perlenkus ištemptą stačiakampį per vidury gaunamas stogas.
- 3.1.4.19. Pastato segmentai spalvinami pasinaudojus viena iš anksto pateikta tekstūra. Tekstūroje saugoma visų generacijoje naudojamų spalvų paletė.
- 3.1.4.20. Pastato segmentai turi būti procedūriškai spalvinami kiekvienam verteksui priskiriant tinkamą spalvą iš spalvų paletės

RNr. 2 Pastato dalių sujungimas į vieną:

- 3.1.4.21. Pastato segmentai, nekeičiant jų geometrijos, turi būti sujungiami į vieną objektą;
- 3.1.4.22. Sujungtas pastatas turi būti naujas objeketas;
- 3.1.4.23. Sujungtas objektas naudoja tą pačią tekstūrą kaip ir nesujungto pastato segmentai;
- 3.1.4.24. Po sujungimo ištrinamas senas pastatas iš kurio segmentų sukurtas sujungtas pastatas.

RNr. 3 Pastato eksportavimas:

- 3.1.4.25. Turi būti galima eksportuoti tiek į vieną sujungtą objektą tiek atskirų modelių rinkinių;
- 3.1.4.26. Objektas eksportuojamas .obj formatu;
- 3.1.4.27. Eksportuojamas objektas talpinamas interfeise nurodytoje vietoje;

RNr. 5 Pastato generacijos laiko testas:

- 3.1.4.28. Turi būti sugeneruojamas kode nustatytas skaičius pastatų apskaičiuojant vidutinį pastato generacijos laiką;
- 3.1.4.29. Pastatų generuojami tokie pagal interfeise pasirinktus nustatymus;

3.1.5. *Reikalavimų lokalizavimo matrica*

Parodoma kuriose sistemos srityse bus įgyvendinami naujai išvesti reikalavimai. Nors reikalavimai išvesti iš vienos konkrečios sistemos srities, tačiau norint daugelį jų įgyvendinti reikalingas tiek interfeisais tiek dalykinė sritis.

3.3 lentelė. Reikalavimų lokalizavimo matrica

Reikalavimai	Vartotojo interfeisas	Dalykinė sritis
3.2.1.2	X	
3.2.1.3	X	
3.2.1.4	X	
3.2.1.5	X	
3.2.1.6	X	
3.2.1.7	X	X
3.2.1.8	X	
3.2.1.9	X	X

3.2.1.10	X	X
3.2.1.11	X	X
3.2.1.12	X	X
3.2.1.13	X	X
3.2.1.14	X	X
3.2.1.15	X	X
3.2.1.16		X
3.2.1.17		X
3.2.1.18		X
3.2.1.19		X
3.2.1.20		X
3.2.1.21	X	X
3.2.1.22		X
3.2.1.23		X
3.2.1.24		X
3.2.1.25		X
3.2.1.26		X
3.2.1.27		X
3.2.1.28		X
3.2.1.29		X

3.1.6. Reikalavimų ryšio matrica

Parodoma visų funkinių reikalavimų kilmė ir lokalizacijos sritis.

3.4 lentelė. Reikalavimų ryšio matrica

Reikalavimas	Iš ko išvestas	Kur lokalizuotas
RNr. 1	Pradinis	Interfeisas
RNr. 2	Pradinis	Dalykinė sritis
RNr. 3	Pradinis	Dalykinė sritis
RNr. 4	Pradinis	Dalykinė sritis
RNr. 5	Pradinis	Dalykinė sritis
3.2.1.2	RNr. 4	Interfeisas

3.2.1.3	RNr. 4	Interfeisas
3.2.1.4	RNr. 4	Interfeisas
3.2.1.5	RNr. 4	Interfeisas
3.2.1.6	RNr. 4	Interfeisas
3.2.1.7	RNr. 4	Interfeisas
3.2.1.8	RNr. 4	Interfeisas
3.2.1.9	RNr. 1	Dalykinė sritis
3.2.1.10	RNr. 1	Dalykinė sritis
3.2.1.11	RNr. 1	Dalykinė sritis
3.2.1.12	RNr. 1	Dalykinė sritis
3.2.1.13	RNr. 1	Dalykinė sritis
3.2.1.14	RNr. 1	Dalykinė sritis
3.2.1.15	RNr. 1	Dalykinė sritis
3.2.1.16	RNr. 1	Dalykinė sritis
3.2.1.17	RNr. 1	Dalykinė sritis
3.2.1.18	RNr. 1	Dalykinė sritis
3.2.1.19	RNr. 1	Dalykinė sritis
3.2.1.20	RNr. 1	Dalykinė sritis
3.2.1.21	RNr. 2	Dalykinė sritis
3.2.1.22	RNr. 2	Dalykinė sritis
3.2.1.23	RNr. 2	Dalykinė sritis
3.2.1.24	RNr. 2	Dalykinė sritis
3.2.1.25	RNr. 3	Dalykinė sritis
3.2.1.26	RNr. 3	Dalykinė sritis
3.2.1.27	RNr. 3	Dalykinė sritis
3.2.1.28	RNr. 4	Dalykinė sritis
3.2.1.29	RNr. 4	Dalykinė sritis

3.2.Programų sistemos architektūra

3.2.1. *Sistemos užduočių scenarijų vykdymas*

Užduoties „Pastato generavimas“ įgyvendinamumas

Scenarijus: Pastato generavimas;

Versija: 1.0

Siekiamas tikslas: Vartotojas mato sugeneruotą procedūriškai sugeneruotą pastatą;

Pirmas agentas: Vartotojas;

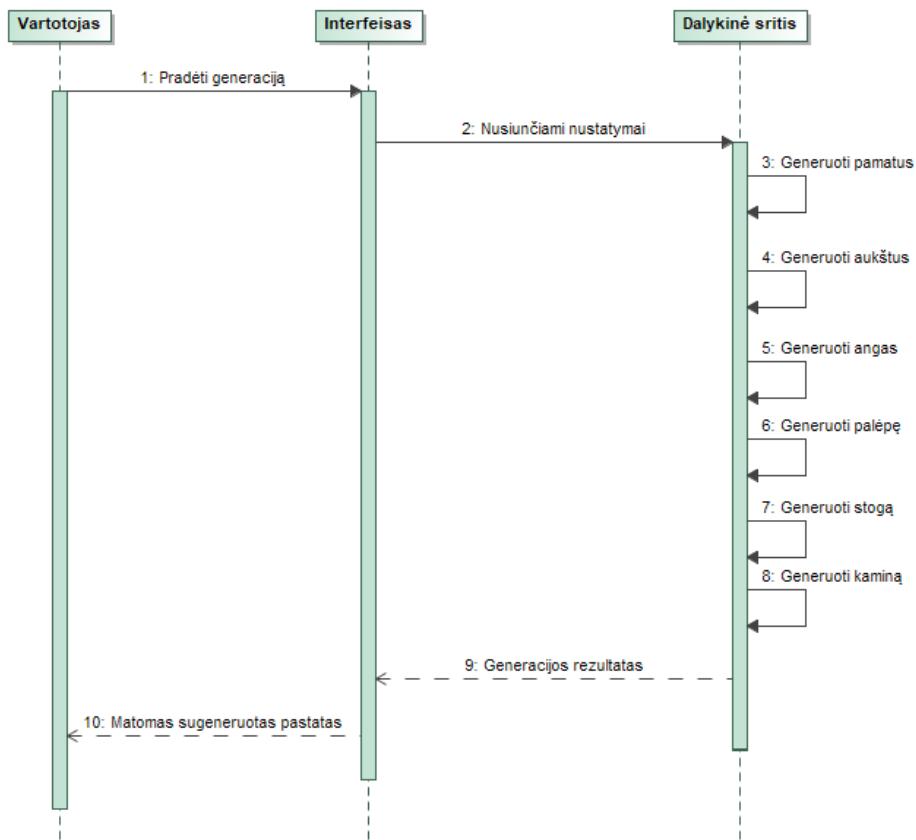
Antriniai agentai: Dalykinė, interfeiso posistemės;

„Prieš“ sąlygos: Vartotojas yra įjungęs ir paruošęs naudojimui sistemą;

„Po“ sąlygos: Interfeise matomas procedūriškai sugeneruotas pastatas;

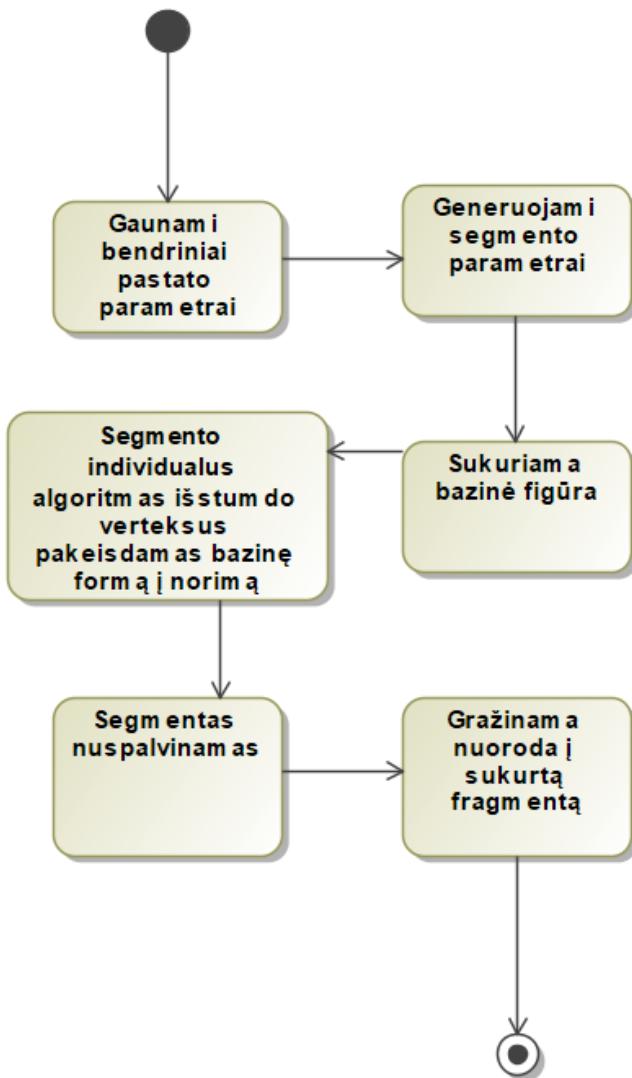
Scenarijus:

1. Vartotojas pastatų generatoriaus interfeise paspaudžia generavimo mygtuką.
2. Dalykinis posistemis gauna interfeiso posistemio parametrus.
3. Pastatas generuojamas iš eilės sukuriant visus jo segmentus: pamatus, aukštus, angas (langus ir duris), palėpę, stogą ir kaminą.
4. Sugeneruotas pastatas atvaizduojamas vartotojo interfeise.



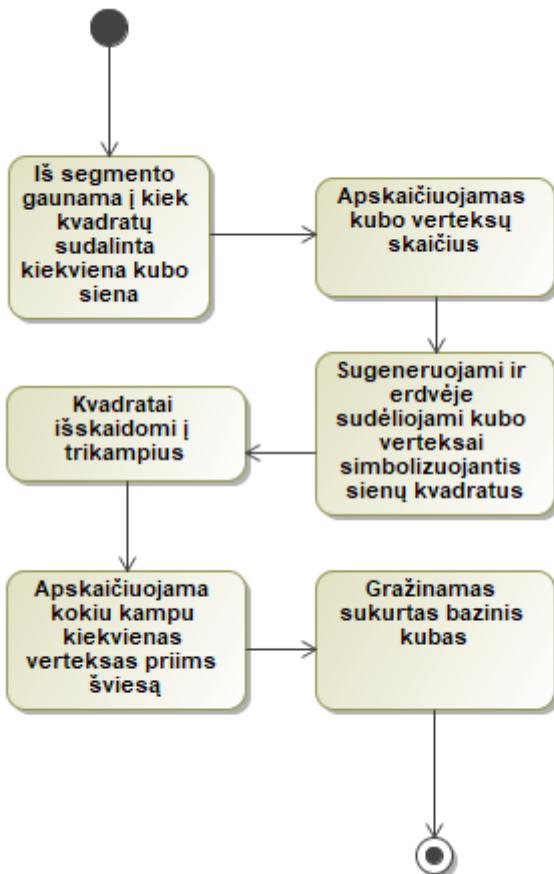
3.3 pav. Pastato generavimo užduoties sekų diagrama

Paveiksle 3.4 matoma diagrama vaizduoja kaip turi veikti segmentų (pamatai, langai, etc.) generavimo algoritmas. Visi segmentai generuojami vykdant tuos pačius žingsnius ta pačia tvarka, skiriasi tik galutinę išvaizdą jiem suteikiantis algoritmas, kadangi bazinio kubo verteksų perstumdymas generuojant stogą ir generuojant arkinį lango rėmas yra visiškai kitoks.



3.4 pav. Segmento generavimo algoritmo veiklos diagrama.

Paveiksle 3.4 pavaizduota diagrama nusako bazinio objekto (šiuo atveju kubo) generavimo algoritmo esminius žingsnius. Kiekvieno segmento generacija pradedama sukuriant bazine figūrą kurios kiekvienos plokštumos suskaidymo kubų skaičius yra iš anksto nustatytas kiekvieno segmento parametruose. Pamatyti tokios suskaidytois bazine figūros pavyzdį galima 7 priedų paveiksle.



3.5 pav. Bazinio kubo generavimo algoritmo veiklos diagramma.

Užduoties „Eksportuoti sugeneruotą pastatą“ įgyvendinamumas

Scenarijus: Pastato eksportavimas

Versija: 1.0

Siekiamas tikslas: Sugeneruotas pastatas (modelis) eksportuojamas universaliu .obj formatu.

Pirmas agentas: Vartotojas;

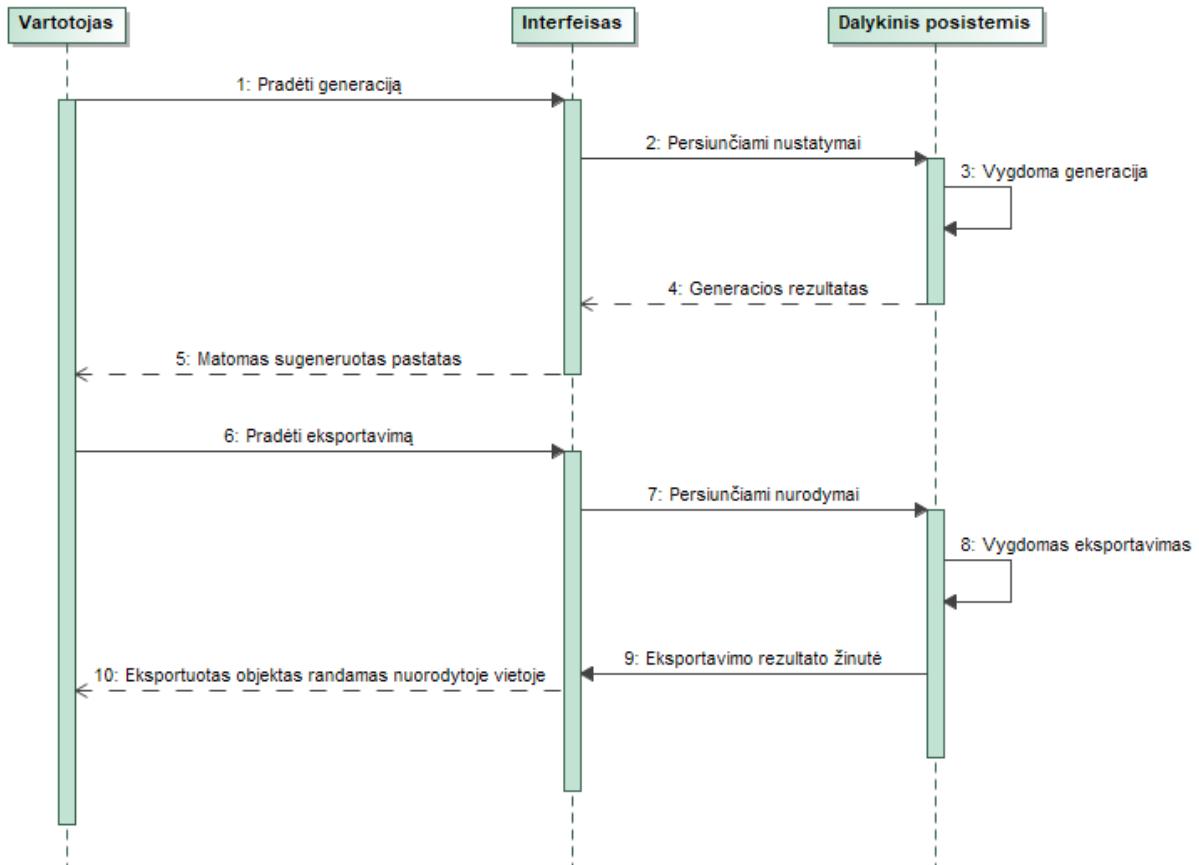
Antriniai agentai: dalykinė, interfeiso posistemės;

„Prieš“ sąlygos: Vartotojas generatoriaus interfeise įrašęs kurioje vietoje bus talpinamas eksportuotas pastato modelis;

„Po“ sąlygos: eksportuotas pastatas .obj formatu;

Scenarijus:

1. Vartotojas generatoriaus interfeise paspaudžia generavimo mygtuką.
2. Dalykinis posistemis gauna interfeiso posistemio parametrus.
3. Pastatas sugeneruojamas pagal nustatyti parametrus.
4. Sugeneruotas pastatas atvaizduojamas vartotojo interfeise.
5. Generatoriaus interfeise paspaudžiamas eksportavimo mygtukas.
6. Dalykinė posistemė eksportuoja sugeneruotą pastatą į vartotojo nurodytą vietą.



3.6 pav. Pastato eksportavimo užduoties sekų diagrama

Užduoties „Sujungti pastato dalis į vieną“ įgyvendinamumas

Scenarijus: Pastato dalii sujungimas į vieną objektą.

Versija: 1.0

Siekiamas tikslas: Sujungti visus sugeneruoto pastato modelius į vieną, siekiant optimizuoti pastatą.

Pirmas agentas: Vartotojas;

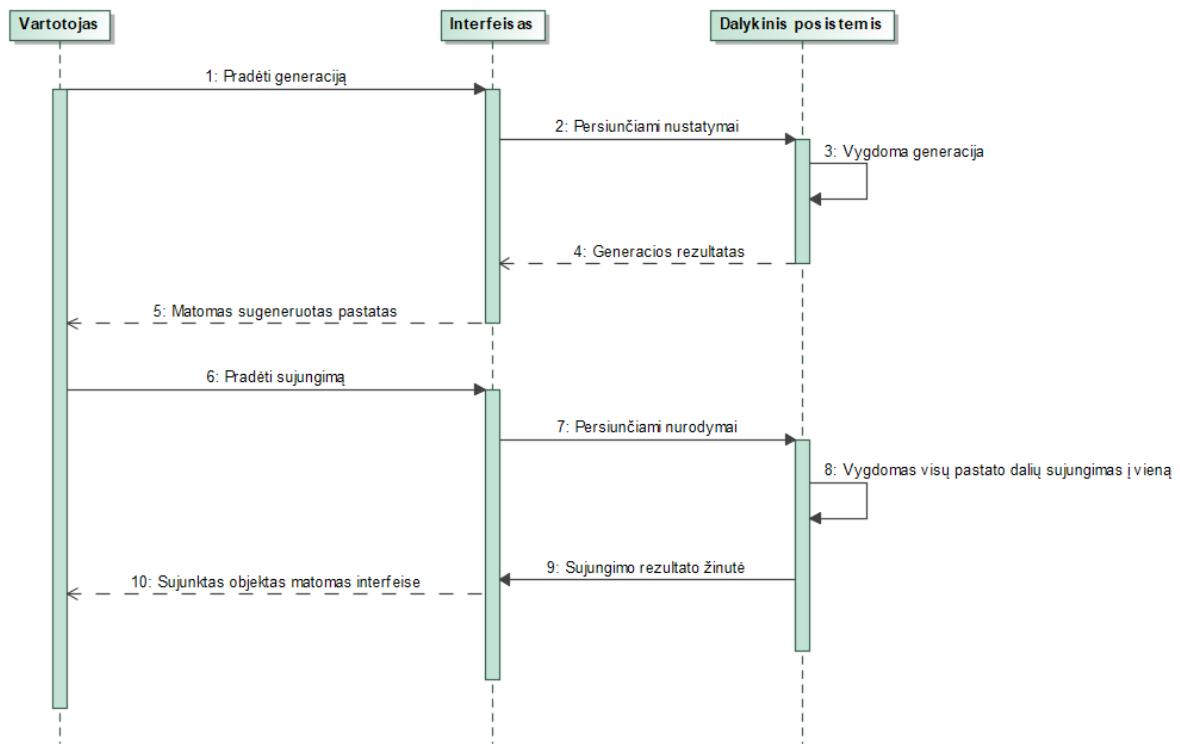
Antriniai agentai: dalykinė, interfeiso posistemės;

„Prieš“ sąlygos: Vartotojas nori sujungti pastatą;

„Po“ sąlygos: Sugeneruoto pastato poligonų tinklai sujungti į vieną;

Scenarijus:

1. Vartotojas generatoriaus interfeise paspaudžia generavimo mygtuką.
2. Dalykinis posistemis gauna interfeiso posistemio parametrus.
3. Pastatas sugeneruojamas pagal nustatytus parametrus.
4. Sugeneruotas pastatas atvaizduojamas vartotojo interfeise.
5. Generatoriaus interfeise paspaudžiamas „Merge Mesh“ mygtukas.
6. Dalykinė posistemė sujungta pastato dalis į vieną objektą.
7. Sujungtas pastatas atvaizduojamas vartotojo interfeise.



3.7 pav. Pastato dalių sujungimo užduoties sekų diagrama

Užduoties „Generavimo parametru nustatymas“ įgyvendinamumas

Scenarijus: Generavimo parametrų pasirinkimas.

Versija: 1.0

Siekiamas tikslas: Vartotojas turi galėti nustatyti kokį pastatą nori sugeneruoti

Pirmas agentas: Vartotojas;

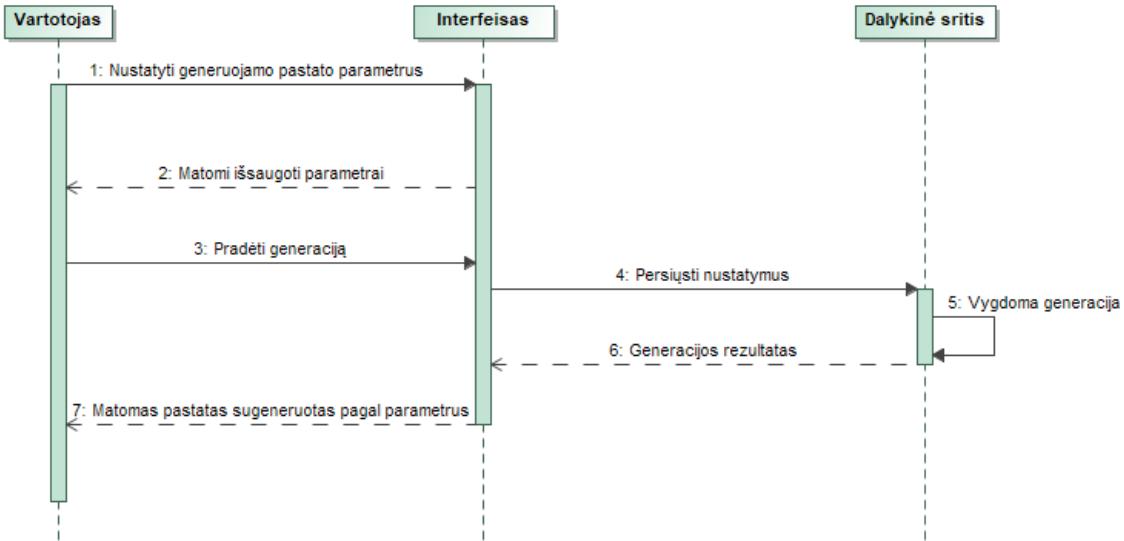
Antriniai agentai: dalykinė, interfeiso posistemės;

„Prieš“ sąlygos: Vartotojas yra įsijungęs ir paruošęs darbui sistemą, bei žino kokio pastato nori;

„Po“ sąlygos: Sugeneruotas pastatas atspindi vartotojo nustatytus parametrus;

Scenarijus:

1. Vartotojas interfeise pasirenka ir nustato norimus pastato parametrus
2. Vartotojas generatoriaus interfeise spaudžia generavimo mygtuką.
3. Dalykinis posistemis gauna interfeiso posistemio parametrus.
4. Pastatas sugeneruojamas pagal nustatytus parametrus.
5. Sugeneruotas pastatas atvaizduojamas vartotojo interfeise, matoma, kad pastatas sugeneruotas naudojantis įvestais parametrais.



3.8 pav. Generavimo parametrų nustatymo užduoties sekų diagrama

3.2.2. PS struktūrinis modelis

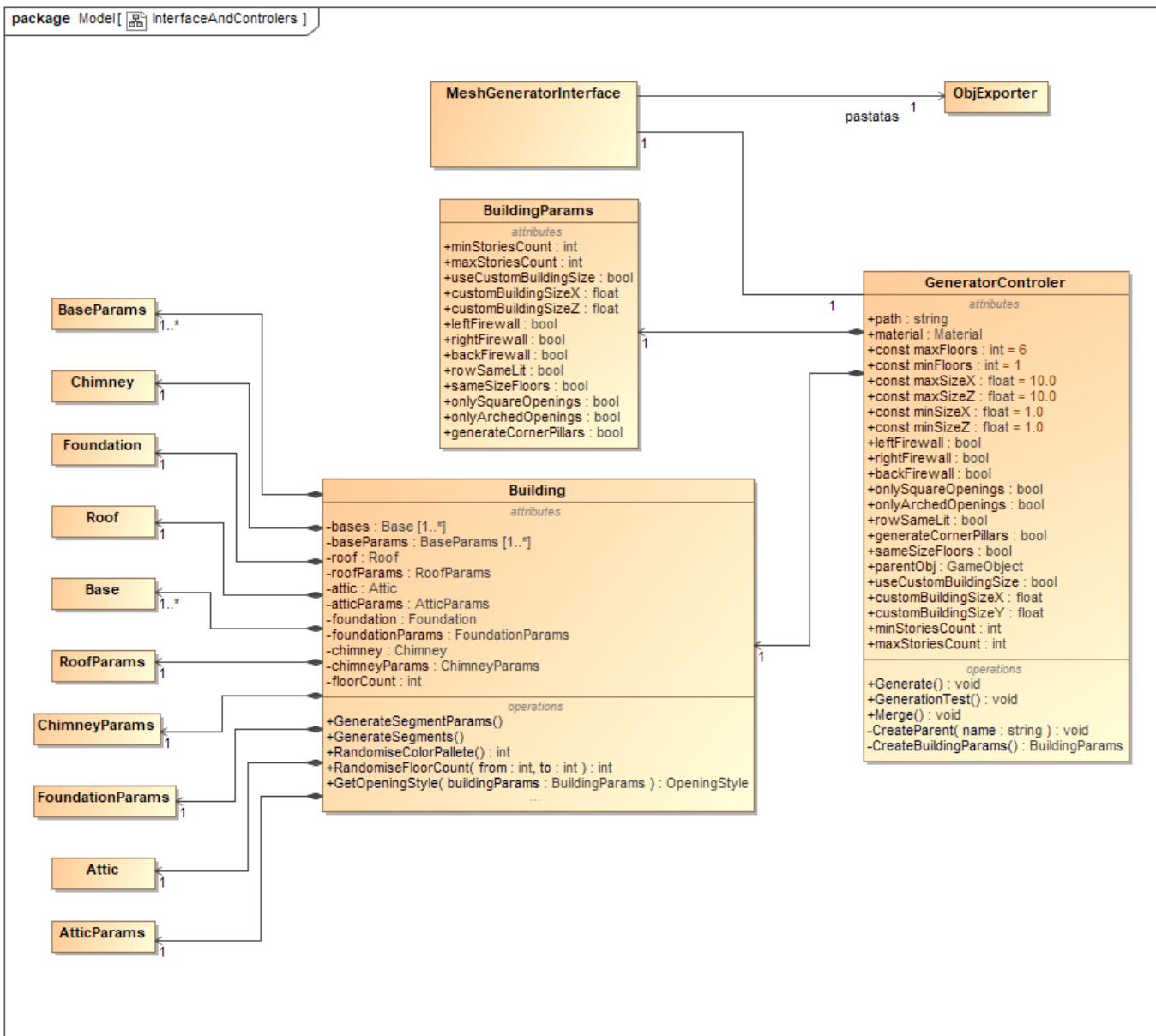
Pasirinkta klasių struktūra leidžia į generaciją lengvai pridėti naujus segmentus, kadangi segmentų parametrai ir generavimas visiškai atskirti vienas nuo kito. Generavimo proceso atskyrimas nuo parametrų taip leidžia turėti skirtinges išvaizdos tuos pačius segmentus (pavyzdžiui langus).

Sistemos klasės išskirstytos po 3 esmines grupes – interfeiso ir generacijos valdymo, segmentų parametru ir pačių segmentų klasės. Tokia pat tvarka sudėliotos ir diagramos.

Interfeiso ir generacijos valdymo klasių struktūra (3.9 pav.):

- MeshGeneratorInterface – skirta Unity aplinkoje atvaizduoti įrankio interfeisą, tiesiogiai susieta su GeneratorController klase.
- GeneratorController – atsakingas už interfeise atliekamų veiksmų perdavimą dalykinei sričiai. Aprašomi visi vartotojui matomi generuojamo pastato parametrai.
- BuildingParams – globalūs, visa pastato stilių nusakantys parametrai, tai iš esmės interfeise nustatyti parametrų objektas (ugniasienės, aukštų skaičių, angų stiliumi ir t.t.).
- Building – už pastato generaciją atsakinga klasė, turi ryšį su visais pastato segmentais ir jų parametrais. Pačio pastato klasių sistema susideda iš:
 - CombineMeshes – sujungia pastato segmentus į vieną objektą.
 - ObjExporter – eksportuoja sugeneruotą objektą .obj format.

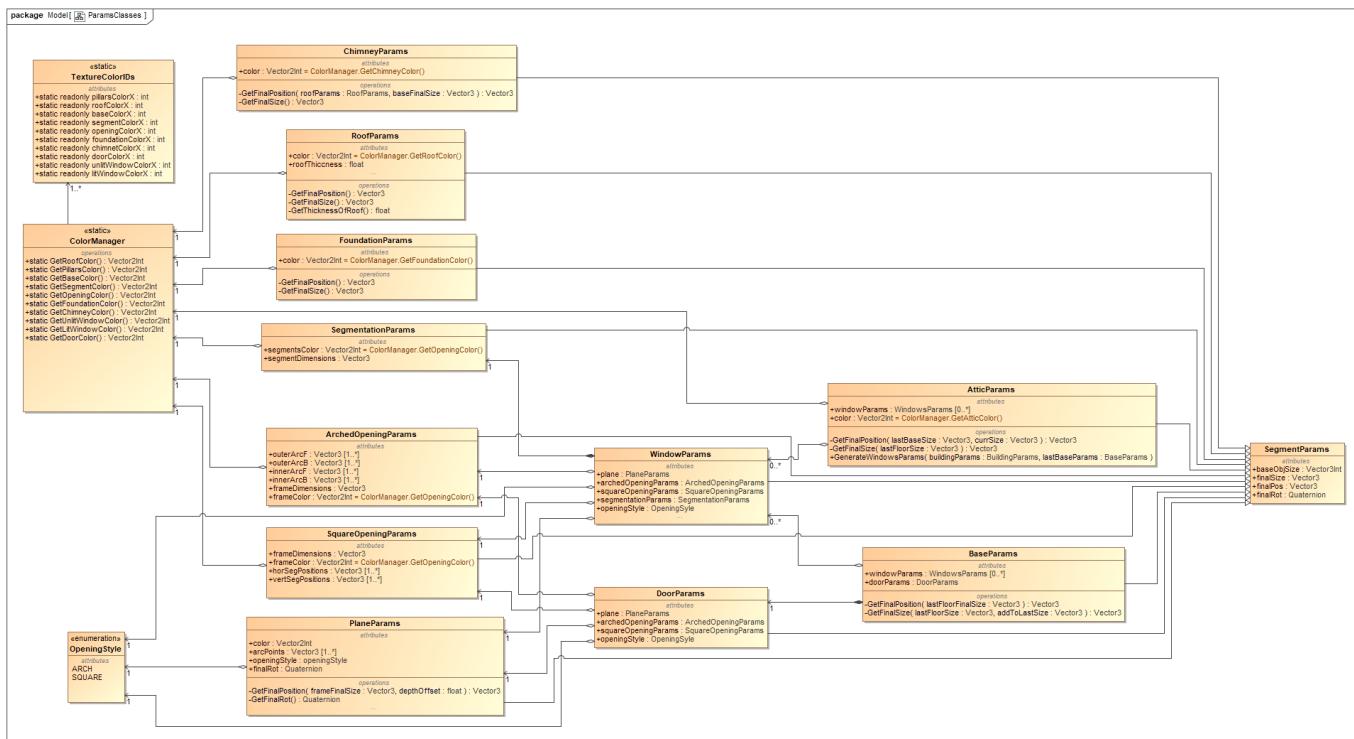
Segmentų parametru ir segmentų klasės aprašomos 3.10 ir 3.11 paveiksluose.



3.9 pav. Interfeiso ir valdymo klasės

Segmentų parametrų klasų struktūra (3.10 pav.):

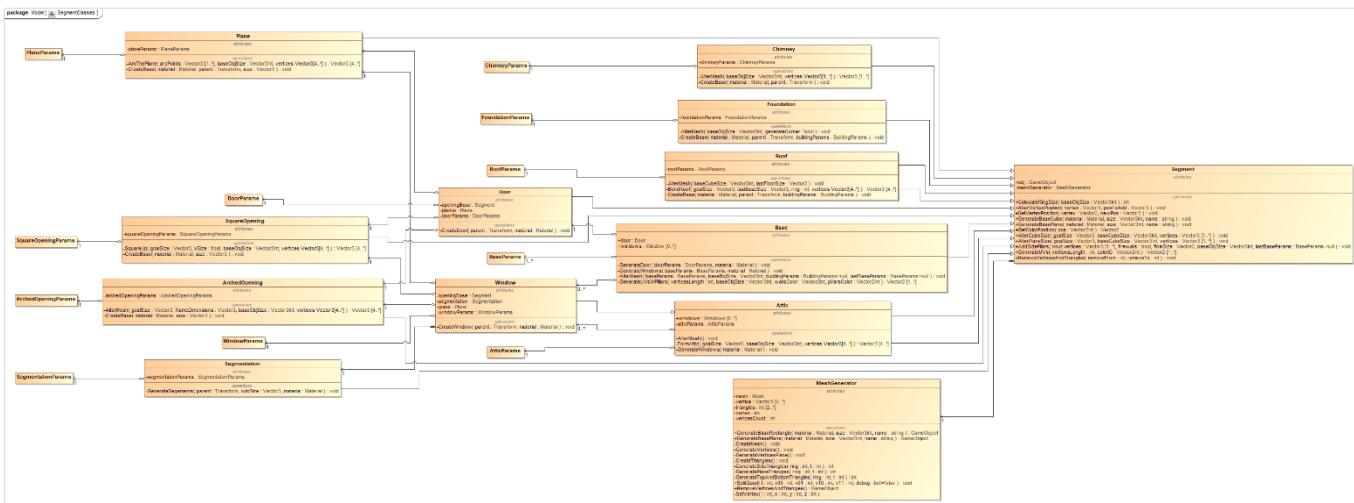
- SegmentParams – tėvinė klasė iš kurios paveldi kiekvieno pastato segmento parametrų vaikinė klasė. Skirta generuoti ir saugoti visus tos pastato dalies parametrus (aukštį, dydį, poziciją ir t.t.). Kiekvienas parametrų objektas, jei yra poreikis, turi galimybę saugoti unikalius tik jam skirtus parametrus bei metodus.
- OpeningStyle – enumeratorius skirtas saugoti angų stilium. Kuriamoje sistemoje yra 2 stiliai – kvadratinis ir arkinis.
- Sistema taip pat turi statines klases skirtas globaliems parametrams saugoti ir naudoti visoje sistemoje:
 - ColorManager – gražina norimo segmento spalvą.
 - TextureColorIDs – saugo kiekvieno pastato segmento spalvos koordinatę tekstūroje.



3.10 pav. Segmentų parametrų saugojimo ir apskaičiavimo klasės

Segmentų klasų struktūra (3.11 pav.):

- Segment – tėvinė klasė iš kurios paveldi kiekvieno pastato segmento vaikinė klasė. Skirta bazinių objektų generacijos valdymui ir saugoti pačio segmento geometriją.
- Vaikinės klasės turi unikalius metodus leidžiančius pasiekti norimą pastato dalies formą. Forma pasiekiama sugeneravus bazinej objektą ir jo verteksus išstumdžius pagal poreikį.
- MeshGenerator – klasė atliekantį patį bazinejo objekto geometrijos generavimą. Gali sugeneruoti į norimą kiekį dalij suskaidytą kubą ar kvadratinę plokštumą.



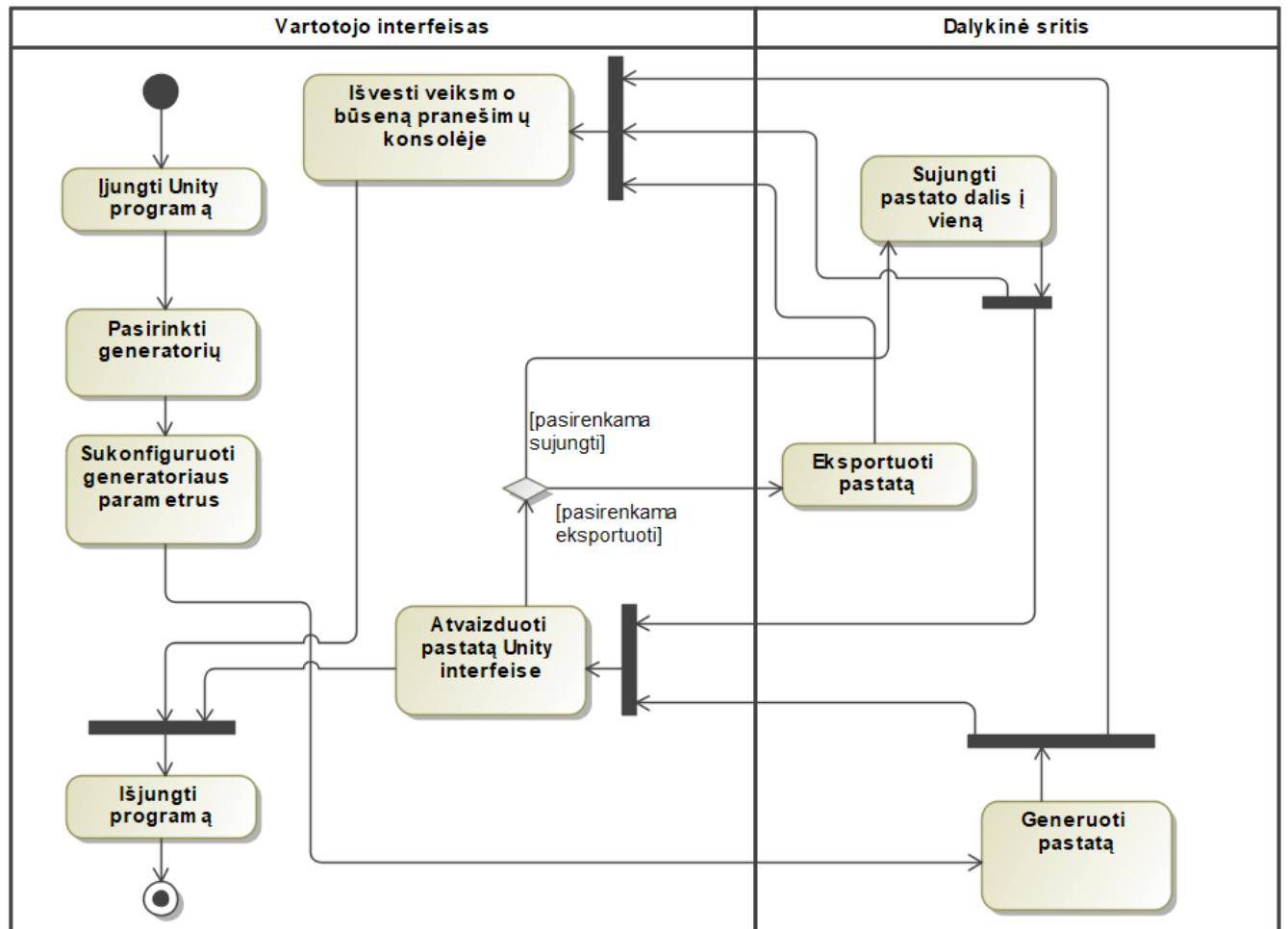
3.11 pav. Segmentų klasės

3.2.3. PS dinaminis modelis

Pati sistema veikia kaip Unity programos plėtinys bei išnaudoja iš anksto programoje esančius langus, taigi norint pasinaudoti sistema reikia įjungti Unity programą. Įjungus hierarchijoje pasirenkamas “Generator” objektas, kas padaro matomą įrankio interfeisą.

Interfeisas leidžia naudotis visomis numatytomis sistemos funkcijomis – valdyti generuojamų pastatų nustatymus, pradėti pačią pastato generaciją ar generacijos testą bei sujungti pastato segmentus į vieną ir eksportuoti sugeneruota pastatą.

Sistemos naudojamas “Scene” langas leidžia 3D aplinkoje matyti įrankio generacijos rezultatą, o “Console” langas suteikia prieigą prie sistemos pranešimų.

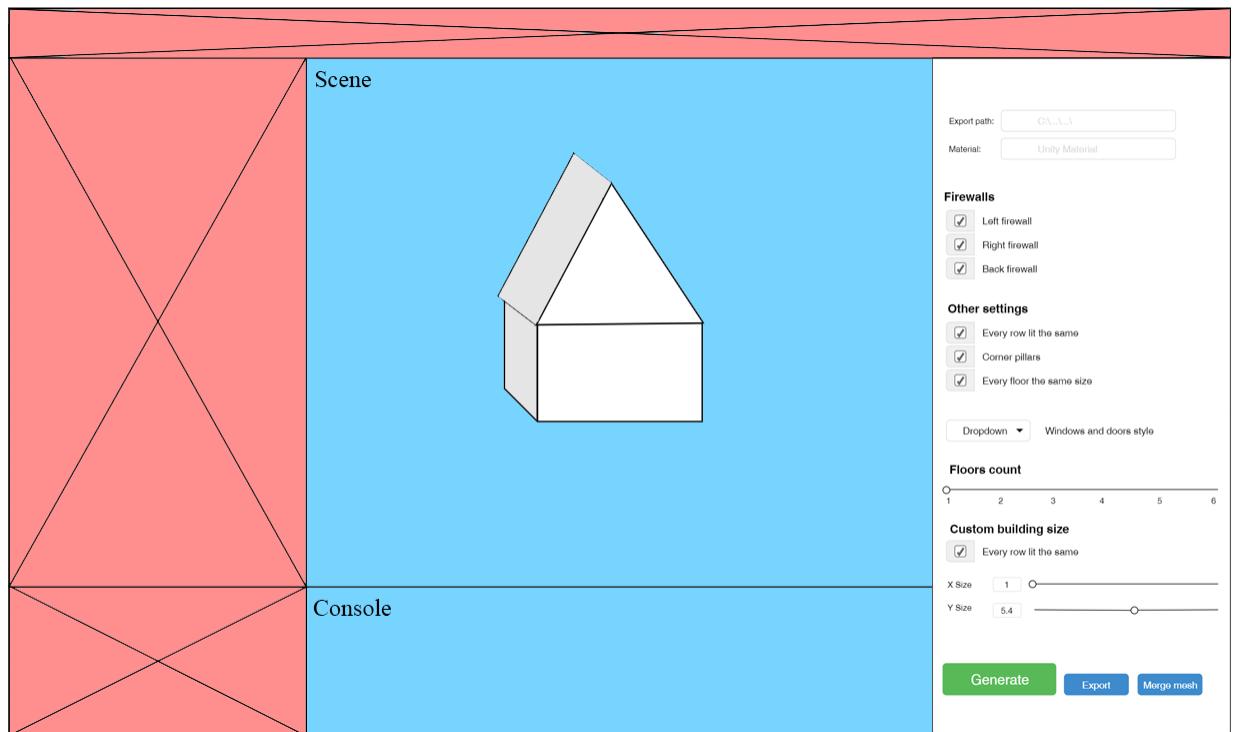


3.12 pav. Sistemos dinaminis modelis

3.3. Programų sistemos maketai

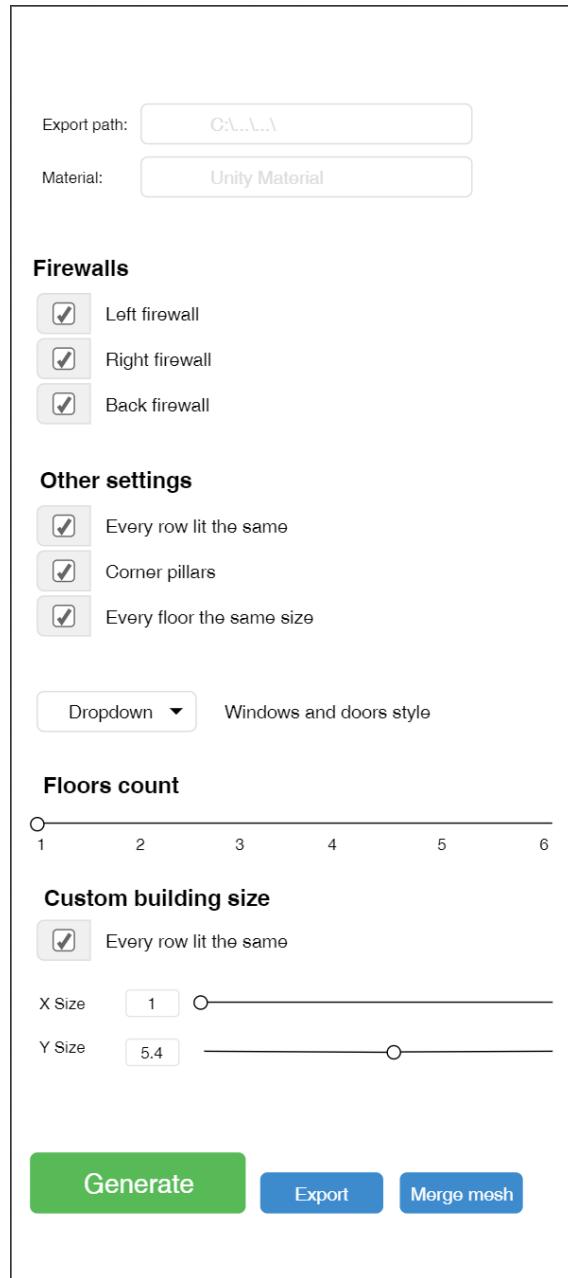
Paveiksle 3.13 matomas maketas simbolizuoją Unity vartotojo aplinką kurioje ir veikia kuriama sistema. Maketo dalių reikšmės:

- Raudona – kuriamam įrankiui nesvarbios Unity interfeiso dalys;
- Mėlyna – Unity egzistuojančios ir kuriamos sistemos naudojamos interfeiso dalys;
 - Scene – pastatų generatoriaus išeigos 3D vizualizacija;
 - Console – pranešimų langas;
- Balta – kuriamas generavimo nustatymų langas;



3.13 pav. Sistemos interfeiso maketas (Unity aplinka)

Žemiau pateiktame 3.14 paveiksle matomas maketas tai 3.1.1.2 poskyryje aprašyta vartotojo sasaja (generavimo nustatymų langas).



3.14 pav. Pastatų generavimo nustatymų lango maketas

4. Testavimas ir įgyvendinimas

Skyrius skirtas parodyti kaip sistema buvo įgyvendinta bei ištestuoti jos veikimą, kokybę ir atitikimą aprašytiems reikalavimams.

Pats vartotojo interfeisas suprojektuotas taip, kad kiekvienas vartotojo galimas pasirinkti nustatymas yra griežtai kontroliuojamas nesudarant jokios galimybės įvesti reikšmes galinčias iššaukti sistemos klaidas. Vienintelė išimtis yra laukelis skirtas įvesti pastato eksportavimo vietą.

Projekto rezitorijs: <https://github.com/KasparasK/ProceduralBuildings>

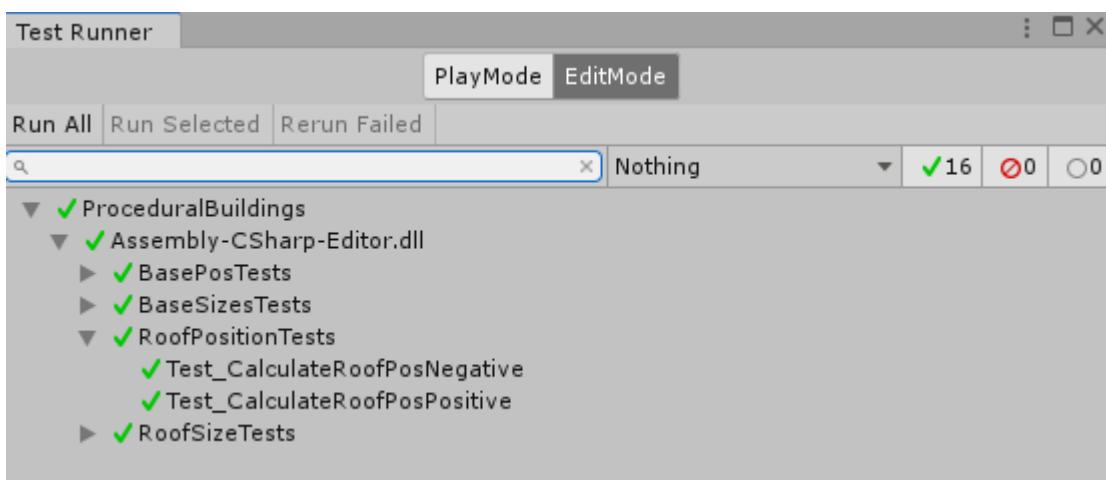
4.1. Automatinis testavimas

Poskyryje aprašomi modulių (angl. unit) testai bei pateikiamas keleto jų kodas.

Modulių testavimui naudojamas Unity aplinkoje esantis „Test runner“. Testai rašomi pasinaudojus NUnit C# biblioteka. NUnit tai atviro kodo unit testavimo įrankis skirtas .NET karkasui. Pasirinktas todėl, kad tai yra standartas kai testuojamas Unity varikliui skirtas kodas.

Modulių testavimui buvo pasirinktos BaseParams ir RoofParams klasės ir jose esantys objekto dydžio bei pozicijos apskaičiavimo metodai.

Viso buvo sukuri 16 unit testų kurių dalį apžvelgsiu, dviejų iš jų kodas pateikiamas žemiau.



Pav. 4.1 Unity Test Runner aplinka

4.1.1. Stogo pozicijos apskaičiavimo testavimas kai stogas didesnis už viršutinį aukštą.

Sukuriamas tuščias stogo parametrų objektas. Sukuriami objektai saugantys paskutinio aukšto dydį bei numatomo stogo dydį. Sukurti parametrai nusiunčiami į pozicijos apskaičiavimo funkciją ir gautas rezultatas lyginamas su norimu.

```
[Test]
public void Test_CalculateRoofPosPositive()
{
    RoofParams roofParams = new RoofParams(Vector3.zero, Vector3.zero);
    Vector3 lastBaseSize = new Vector3(2, 2, 2);
    Vector3 finalRoofSize = new Vector3(2.5f, 3, 2.5f);

    Vector3 expectedPos = new Vector3(0, 2, -0.25f);

    Vector3 pos = roofParams.GetFinalPosition(lastBaseSize, finalRoofSize);
```

```
        Assert.That(pos, Is.EqualTo(expectedPos)) ;  
    }  
}
```

4.1.2. Pirmo aukšto pozicijos apskaičiavimo testavimas kai visos ugniasienės išjungtos

Sukuriamas tuščias pastato parametrų objektas, paskutinio aukšto dydis, nustatoma kiek didesnis kitas aukštasis. Sukuriamas aukšto parametrų objektas kuriam priskiriami anksčiau sukurti kintamieji. Apskaičiuojamas naujo aukšto dydis ir galiausiai apskaičiuojama aukšto pozicija bei palyginama su norima pozicija.

```
[Test]  
public void Test_BasePosGroundFloor()  
{  
  
    BuildingParams buildingParams = new BuildingParams();  
    Vector3 lastFloorSize = new Vector3(2.5f, 2f, 2f);  
    Vector3 addToLastSize = new Vector3(1f, 1f, 1f);  
  
    BaseParams baseParams = new BaseParams(lastFloorSize, buildingParams, 1,  
OpeningStyle.ARCH);  
    baseParams.finalSize = lastFloorSize + addToLastSize;  
  
    Vector3 expectedPos = new Vector3(-0.5f, 2, -0.5f);  
  
    Vector3 pos = baseParams.GetGroundFloorFinalPosition(lastFloorSize);  
  
    Assert.That(pos.x, Is.EqualTo(expectedPos.x).Within(0.001));  
  
}
```

4.2. Rankinis testavimas

Poskyryje aprašomi sistemos testavimo scenarijai ir pagal juo sukurti testavimo atvejai. Dalis testavimo atvejų įgyvendinami ir aprašomas jų rezultatas. Taip pat poskyryje pademonstruojamas sukurtos sistemos veikimas ir jos įgyvendinimas bei pateikiamas kodo ištraukos.

4.2.1. Testavimo scenarijai

Testavimo scenarijai rašomi daugeliui nefunkcinių reikalavimų, kadangi šie turi konkrečius, ištестуojamus kriterijus.

Scenarijų svarbos lygiai:

- Kritinė
- Aukšta
- Vidutinė
- Maža

4.1 lentelė. Testavimo scenarijai

Scenarijaus Nr.	Scenarijų atitinkančio reikalavimo Nr.	Scenarijaus aprašymas	Scenarijaus svarba
TS1	3.1.3.2.1.2	Patikrint ar rankiniu būdū nustačius namo pagrindo dydį, sugeneruoto namo plotas yra tarp 2x2m ir 10x10m dydžio.	Aukšta
TS2	3.1.3.2.1.3	Patikrint ar įmanoma sugeneruoti namus nuo 1 iki 6 aukštų	Aukšta
TS3	3.1.3.2.1.4	Patikrint ar sugeneruoto namo ugniasienių skaičius ir pusė atitinka pažymétas interfeise	Aukšta
TS4	3.1.3.2.2.1	Patikrinti kiek laiko sistema veiks be trykių	Maža
TS5	3.1.3.2.4.1.1	Patikrinti ar vieno pastato su generacijos trukmė neviršija 100ms	Aukšta
TS6	3.1.3.3.1.1	Patikrinti ar sistemos diegimo trukmė atitinka reikalavimą	Maža
TS7	3.1.3.3.2.1	Patikrinti keik laiko vartotojui trunka įsisavinti sistemos valdymą	Vidutinė
TS8	3.1.3.5.4.1	Patikrinti ar sistemas veiks perkėlus sistemos aplanką į kita Unity projektą	Aukšta

4.2.2. Testavimo atvejai

Testavimo atvejai parašyti aukštos ir kritinės reikšmės testavimo scenarijams.

4.2 lentelė. Testavimo atvejai

TA Nr.	Scen . Nr	Testavimo atvejis/ką testuojame	Kas turi būti padaryta prieš vykdant testą	Testavimo eiga/žingsniai	Laukiamas rezultatas
TA1	TS1	Patikrint ar rankiniu būdū nustačius namo pagrindo dydį, sugeneruoto namo plotas yra tarp 2x2m ir 10x10m dydžio.	Sistema pilnai paruošta darbui	Interfeise: 1) Pasirenkamas „Custom building size“ 2) Paeiliui nustatomi 2x2, 5x5 ir 10x10 dydžiai 3) Su kiekvienu dydžiu sugeneruojamas pastatas	Sugeneruojami norimo dydžio pastatai, nėra klaidos pranešimų
TA2	TS2	Patikrint ar įmanoma sugeneruoti namus nuo 1 iki 6 aukštų	Sistema pilnai paruošta darbui	Interfeise: 1) Paeiliui nustatyti nuo 1 iki 6 aukštų ir 2) Kiekvienam aukštui sugeneruoti po pastatą	Sugeneruojami pastai su norimu skaičiumi aukštų, nėra klaidos pranešimų
TA3	TS3	Patikrint ar sugeneruoto namo ugniasienių skaičius ir pusė atitinka	Sistema pilnai paruošta darbui	Interfeise: 1) Paeiliui pažymime tik kairę ugniasienę, tik dešinę ir tik galinę	Sugeneruojami pastai su noriomis ugniasienėmis , nėra klaidos pranešimų

		pažymėtas interfeise		2) Su kiekviena ugniasiene sugeneruojame pastatą 3) Pažymime visas ugniasienes vienu metu 4) Sugeneruojame pastatą	
TA4	TS5	Patikrinti ar vieno pastato su generacijos trukmė neviršija 100ms	Sistema pilnai paruošta darbui	Interfeise: 1) Nustatyti, kad būtų generuojami 6 aukštų pastatai 2) Nustatyti, kad būtų generuojamos šoninės kolonos 3) Ijungti pastatų generavimo greičio testą	Vidutinė pastato generacijos trukmė neviršija 100ms
TA5	TS8	Patikrinti ar sistema veiks perkėlus sistemos aplanką į kita Unity projektą	Sistema išbandyta ir veikianti	1) Sukurti naują Unity projektą 2) Perkelti sistemos aplanką į naują projektą 3) Naujame projekte pabandyti sugeneruoti pastatą	Pastatai generuojami, nėra klaidos pranešimų

4.2.3. *Generacijos greičio tyrimas*

Testavimui naudojama techninė įranga:

CPU - Ryzen 1600; 3.4 ghz; 6 cores; 12 threads

GPU – GeForce Gtx 1070

RAM – 16gb

Testo tikslas:

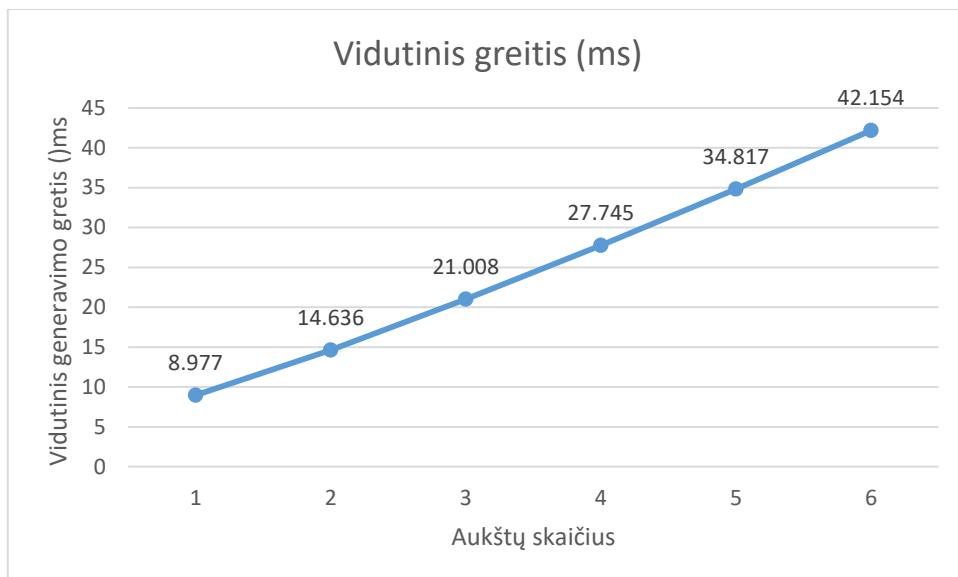
Patikrinti kiek vidutiniškai trunka sugeneruoti 1-6 aukštų pastatus, įvertinti generacijos greičio pokyčius ir įsitikinti, kad

Testo parametrai:

- 1000 pastatų kiekvienam aukštui, viso 6000 pastatų
- Nuo 1 ir 6 aukštų
- Ijungtos kampinės kolonos
- Viena ugniasiė
- Kitos reikšmės atsitiktinės
- Testas atliekamas 5 kartus

4.3 lentelė. Testo rezultatai

Aukštų sk.	Vidutinis greitis (ms)	Pokytis
1	8.977	0
2	14.636	5.659
3	21.008	6.372
4	27.745	6.737
5	34.817	7.072
6	42.154	7.337



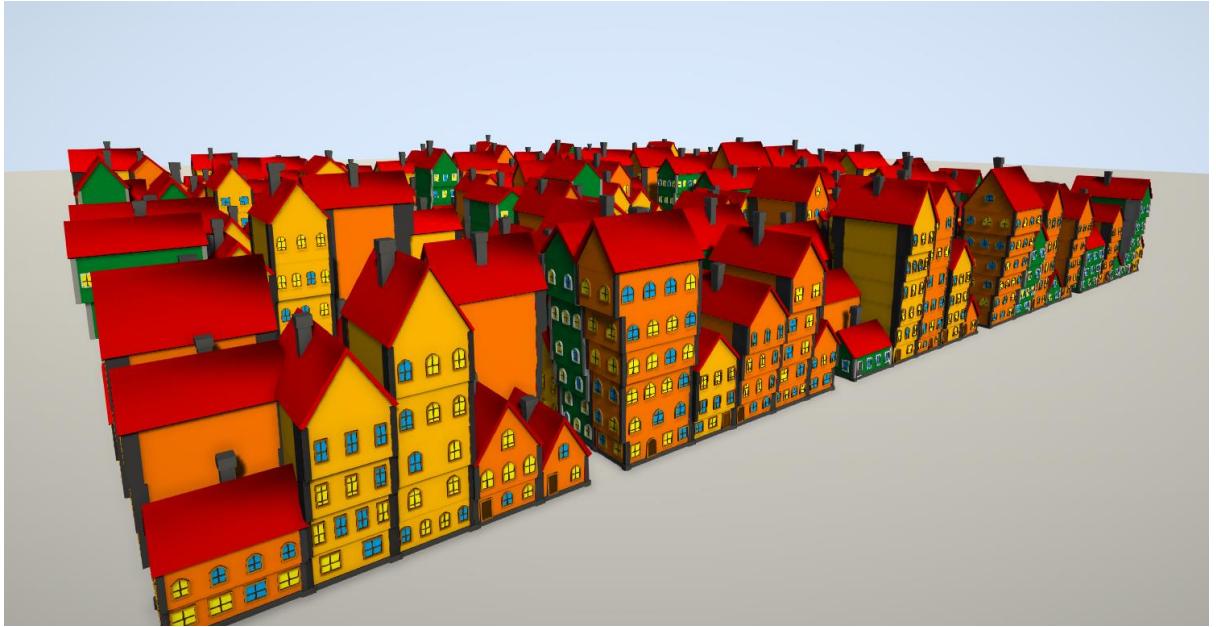
4.2 pav. Vidutinė pastato generavimo trukmė didinant aukštų skaičių

Iš pateiktų rezultatų matoma, kad 1-6 aukštų pastatų generavimo greitis svyruoja tarp 9 ir 42ms. Toks greitis tenkina ir nefunkcinį reikalavimą sakant, kad vieno pastato generacija negali viršyti 100ms.

Taip pat, nors iš grafiko greičio mažėjimas atrodo linijinis, paskaičiavus pokyti matosi, kad kiekvienas papildomas aukštasis prailgina generaciją labiau nei prieš tai buvęs.

4.2.4. Sukurtos sistemos veikimo ir vartotojo sąsajos demonstracija

Žemiau pateiktame 4.3 paveiksle matomas demonstracijos tikslais sugeneruotas miestas parodantis galimą sistemos generuojamą pastatų panaudojimą:



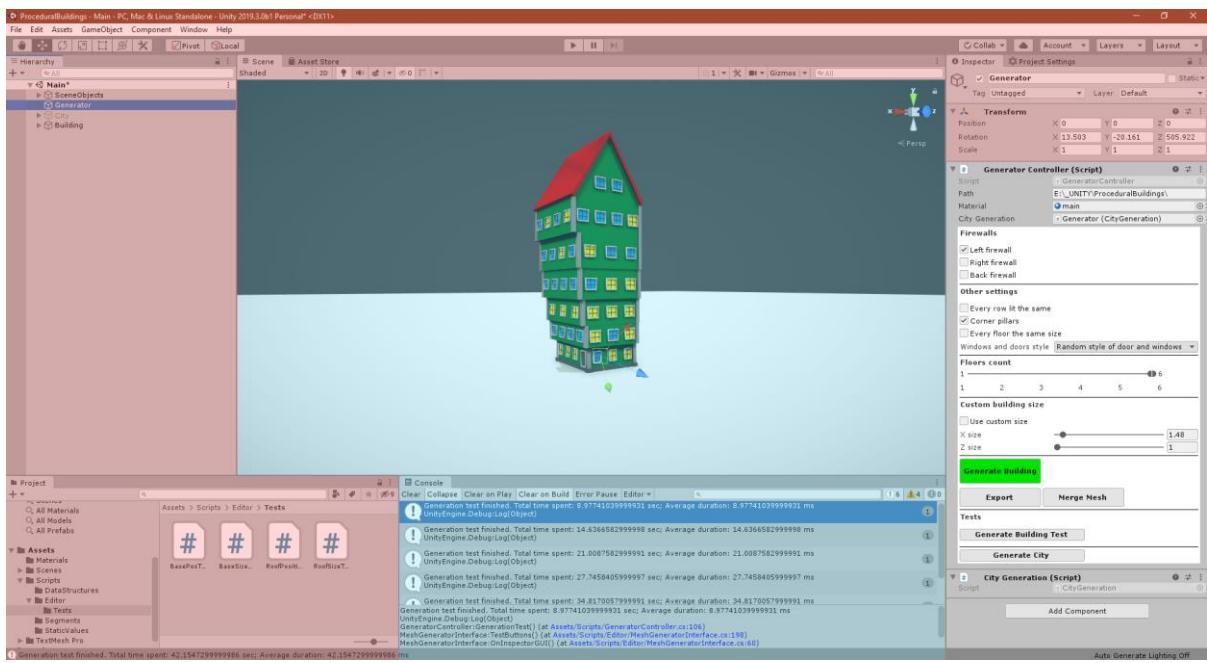
4.3 pav. Iš sugeneruotų pastatų sustatyto miesto pavyzdys

4.4 paveikslė kairėje matomas 1 aukšto pastato generacijos pavyzdys. Dešinėje matoma iš kokių segmentų sukomponuotas šis pastatas:



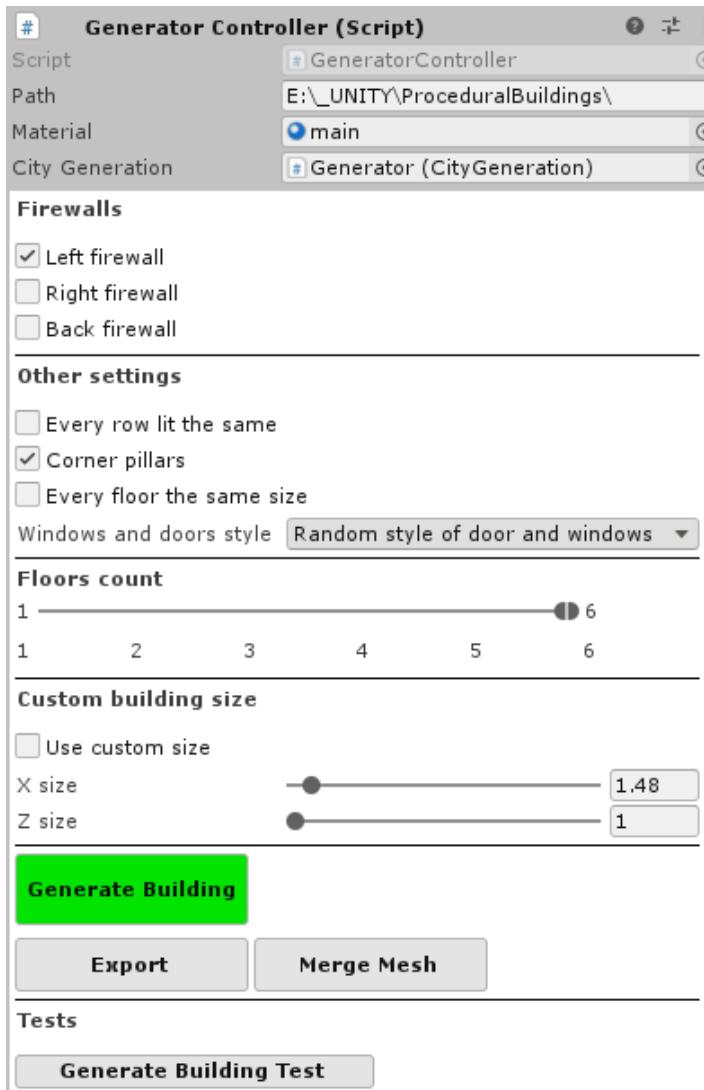
4.4 pav. Sugeneruoto namo pavyzdys ir namo segmentų demonstracija

Paveiksle 4.5 matoma visa Unity vartotojo sąsaja nuspalvinta pagal 3.13 paveikslę parodytą maketą, kad būtų galima palyginti planuotą ir realų sistemos vaizdą.



4.5 pav. Maketo spalvomis nuspavinta vartotojo sasaja

Galiausiai pateikiamas priartintas pastatų generavimo nustatymų langas kuriame matomi visi projektinėje dalyje suplanuoti interfeiso elementai:



4.6 pav. Pastatų generavimo nustatymų langas

4.2.5. Reikalavimo, kad sistema nereikalautų bazinio modelių rinkinio įgyvendinimas

Poskyryje parodoma, kad sistema iš tiesų sukurta taip, kad nereikėtų naudoti jokio bazinio modelių rinkinio. Šis reikalavimas įgyvendinamas visus galimus pastato segmentus (langus, duris, stogą, kaminą ir t.t.) aprašant sistemos kode.

Kiekvieno segmento generacija vykdoma sistemos projektavimo dalyje nustatytais žingsniais:

1. Sukuriama bazinė figūra (kubas arba plokštuma)
 - a. Sugeneruojami verteksai.
 - b. Is verteksų sukuriami poligonai.
 - c. Apskaičiuojama kokiu kampu kris šviesa ant kiekvieno poligono.
2. Pradinės figūros dimensijos ištampomos iki norimų.
3. Figūros verteksai išstumdomi taip, kad būtų sukuriama siekiama forma (pvz. lango rėmas).
4. Kiekvienam verteksui iš bendros tekstūros priskiriama spalva.

Pirmi du žingsniai yra standartiniai, tačiau trečias yra unikalus kiekvienai figūrai, kadangi figūros reikalauja skirtingo verteksų skaičiaus ir išdėliojimo, kad būtų pasiekta norima forma.

Vizualizuoti toliau pateikiamus kodo pavyzdžius padeda 7 priedo paveikslas.

Verteksų generavimo funkcijos kodas. Verteksai generuojami einant žiedais iš pradžių per kubo šonus vėliau per apatinę ir viršutinę plokštumas:

```
void GenerateVertices()
{
    int v = 0;
    //ziedu sluoksniai
    for (int y = 0; y <= ySize; y++)
    {
        //sukuriamas spirale pirmas ziedas
        for (int x = 0; x <= xSize; x++, v++)
        {
            SetVertex(v, x, y, 0);
        }
        for (int z = 0; z <= zSize; z++, v++)
        {
            SetVertex(v, xSize, y, z);
        }
        for (int x = xSize; x >= 0; x--, v++)
        {
            SetVertex(v, x, y, zSize);
        }
        for (int z = zSize; z >= 0; z--, v++)
        {
            SetVertex(v, 0, y, z);
        }
    }
    //dugnas ir virusus
    for (int y = ySize; y >= 0; y -= ySize)
    {
        for (int z = 0; z <= zSize; z++)
        {
            for (int x = 0; x <= xSize; x++, v++)
            {
                SetVertex(v, x, y, z);
            }
        }
    }
}
```

Plokštumos poligonų skaidymo į trikampius funkcija. Kiekvienas poligonas turi būti suskaidomas į 2 trikampius. Tai atliekama į skaicių masyvą teisinga tvarka sudedant verteksų indeksus:

```
int GenerateSideTriangles(int ring, int t)
{
    triangles = new int[
        (zSize*ySize* 12) + (xSize * ySize * 12) + (xSize * zSize * 12)
    ];

    for (int y = 0; y < ySize; y++)
    {
        for (int i = 0; i < 2; i++)
        {
            for (int x = 0; x < xSize; x++, vertex++)
            {
                SplitQuad(ref t,
                    vertex,
                    ring + vertex,
                    (vertex + 1),
                    ring + vertex + 1);
            }
        }
    }
}
```

```

        }
        vertex++;
        for (int z = 0; z < zSize; z++, vertex++)
        {
            SplitQuad(ref t,
                      vertex,
                      ring + vertex,
                      (vertex + 1),
                      ring + vertex + 1);
        }
        vertex++;
    }
}
return t;
}

```

Kiekvienas prototipinės sistemos sugeneruotas pastatas susideda iš tokio pačio figūrų rinkinio:

- Pamatų;
- Bazinių aukštus reprezentuojančių figūrų;
 - Kiekvienna bazė turi langų rinkinį;
 - Pirmas aukštasis turi duris;
- Stogo;
- Palėpės;
 - Palėpė gali turėti langų rinkinį;
- Kamino;

Langai ir durys susideda iš tų pačių bazinių figūrų:

- Plokštumos;
- Segmentų (tik langai);
- Arkinio arba stačiakampio rėmo;

Palėpės suformavimo iš bazinio kubo funkcijos kodas. Iš esmės šis kodas paima bazinį kubą, jį ištempia į didelio ilgio ir pločio stačiakampį ir vidurinius verteksus iškelia į viršų:

```

void FormAttic(Vector3 goalSize, Vector3Int baseObjSize, ref Vector3[] vertices)
{
    int ring = CalculateRingSize(baseObjSize);
    goalSize.x /= -2;

    Vector3 sizeToAdd = Vector3.zero;

    int tempId = 1;
    sizeToAdd.x = goalSize.x;

    for (int i = 0; i < ring/2; i++)
    {
        AlterVertexPosition(ref vertices[tempId], sizeToAdd);
        tempId++;
    }
    sizeToAdd.x *= -1;
    tempId += (ring / 2) - 1;
    AlterVertexPosition(ref vertices[tempId], sizeToAdd);
    tempId++;

    sizeToAdd.x = 0;
    sizeToAdd.y = -goalSize.y;
    AlterVertexPosition(ref vertices[tempId], sizeToAdd);
}

```

```
tempId++;
AlterVertexPosition(ref vertices[tempId], sizeToAdd);
tempId++;
AlterVertexPosition(ref vertices[tempId], sizeToAdd);
tempId++;
AlterVertexPosition(ref vertices[tempId], sizeToAdd);
tempId++;

sizeToAdd.y = 0;
sizeToAdd.x = -goalSize.x;
AlterVertexPosition(ref vertices[tempId], sizeToAdd);
tempId++;
AlterVertexPosition(ref vertices[tempId], sizeToAdd);
tempId++;
AlterVertexPosition(ref vertices[tempId], sizeToAdd);

}
```

5. Išvados ir pasiūlymai

Atlikus į kuriamą sistemą panašių sistemų analizę gauta, kad egzistuoja platus itin universalų procedūrinio pastatų generavimo įrankių pasirinkimas. Tačiau, nors jų veikimo principas ir platforma kurioje jos veikia skiriiasi, jos visos orientuotos į realistinius, bei, daugeliu modernius pastatus, taip pat, reikalauja išankstinio pastato segmentų rinkinio. Taigi buvo nuspręsta įgyvendinti sistemą kuri generuotų mažo poligonų kieko stilistikos viduramžių stiliaus pastatus bei nereikalautų jokio rankiniu būdu sukurto pastato dalių rinkinio.

Išanalizavus skirtingus procedūrinio generavimo metodus bei algoritmus buvo padaryta išvada, kad jie nėra visiškai tinkami kuriamai sistemai įgyvendinti. L-sistema tinkama sukurti pakankamai primityviems tūriams, o sudėtingi elementai turi būti sukuriami iš anksto. Greuter metodas yra arčiausiai to kas reikalinga kuriamam generatoriui, kadangi pastato parametrai generuojami pasinaudojant atsitiktinių skaičių generatoriumi. Galiausiai, momentinės architektūros algoritmas remiasi į procedūriškai kuriamą objekto geometriją įterpiamais iš anksto sukurtais segmentais. Taigi, kadangi nei vienas iš analizuotų metodų pilnai neatitinka keliamų sistemos uždavinį buvo nuspręsta kuriamai sistemai taikyti savo sukurtą metodą, kuris leistų generuoti pastatus pasinaudodamas atsitiktinių skaičių generatoriumi bei primityvių tūrių manipuliacija siekiant pasiekti norimas pastato formas.

Siekiant aiškiai apibrėžti kuriamą sistemą buvo suformuluoti funkciniai ir nefunkciniai reikalavimai. Jų kūrimas leido aiškiai nustatyti svarbiausias sistemos funkcijas, apribojimus. Tuomet šių reikalavimų pagrindu buvo atliktas visos sistemos projektavimas kuris įgalina efektyvų perėjimą prie realizacijos.

Galiausiai, itin universalioje Unity variklio aplinkoje buvo įgyvendinta pati sistema. Sistema generuoja pastatus pagal anksčiau nubrėžtus reikalavimus – nenaudodama išankstinio objektų rinkinio, visos pastato formos kuriamos pasinaudojus kodu. Sugeneruotas objektas tenkina mažo poligonų kieko stilistikos viduramžių stiliaus reikalavimus – yra minimalistinės, bet aiškiai išreikštų viduramžių pastatų formų ir žaismingos išvaizdos.

Sistema galėtų būti tobulinama optimizuojant kodą, sukuriant daugiau perpanaudojamų metodų, kad būtų paprasčiau prie generacijos pridėti naujus segmentus. Galų gale pats generuojamu segmentų skaičiaus didinimas smarkiai pagerintų kiekvieno įrankio sugeneruoto pastato unikalumą, kadangi turint, tarkim, didelį pasirinkimą langų formų būtų galima sugeneruoti didelį skaičių skirtinai atrodančių namų.

Kasparas Kažemėkas

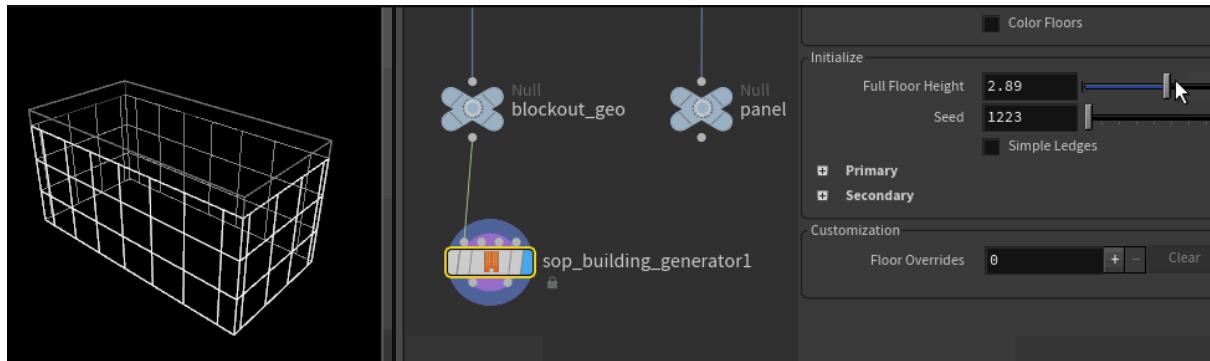


Literatūra ir šaltiniai

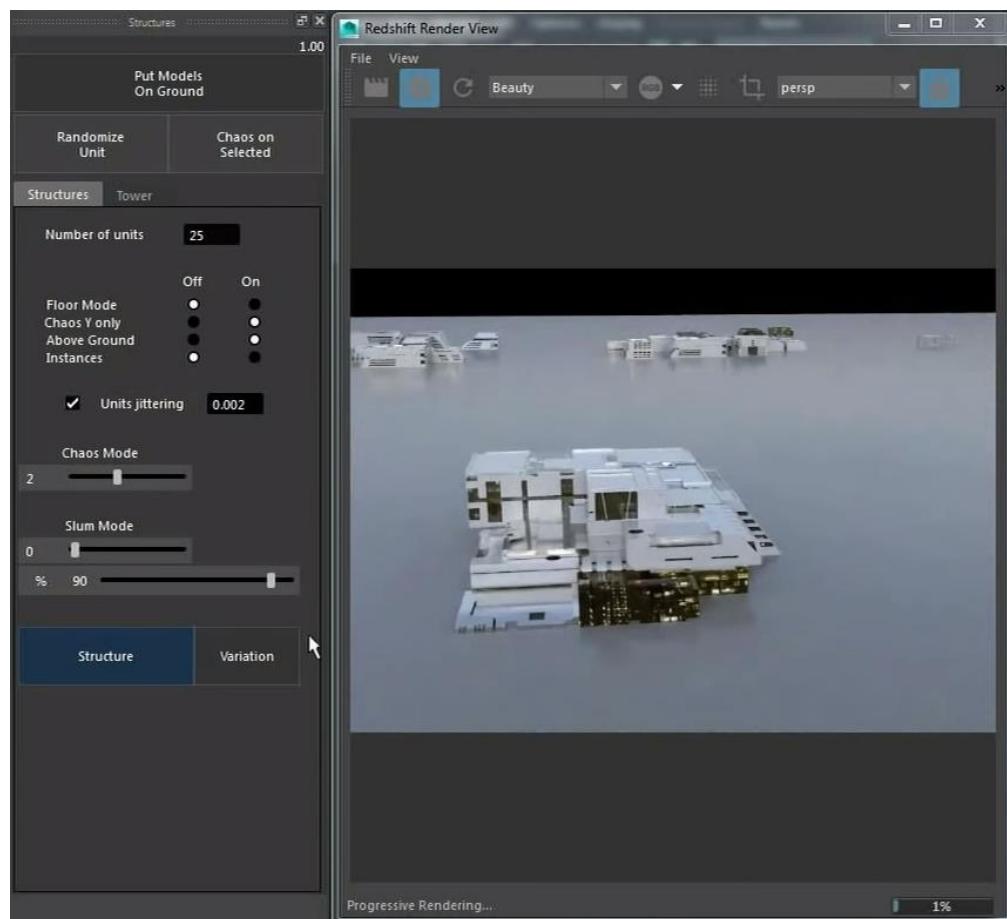
- [1] „newzoo,“ [Tinkle]. Available: <https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/>. [Kreiptasi 27 12 2019].
- [2] J. Couture, „Gamasutra,“ 01 07 2016. [Tinkle]. Available: https://www.gamasutra.com/view/news/273904/Why_are_so_many_devs_employing_a_retro_lowp_oxy_mid1990s_aesthetic.php.
- [3] E. Narcisse, „Kotaku,“ [Tinkle]. Available: <https://kotaku.com/why-new-video-games-still-cost-60-1545590499>.
- [4] [Tinkle]. Available: <https://panoramastreetline.com/news/deutsche-fachwerkstadt-fachwerkstrasse-panorama>.
- [5] „SideFX,“ pastatų generatorius, [Tinkle]. Available: <https://www.sidefx.com/tutorials/building-generator/>. [Kreiptasi 20 11 2019].
- [6] „Maya Scructures,“ pastatų generatorius, [Tinkle]. Available: <https://gumroad.com/l/UELQtt>. [Kreiptasi 20 11 2019].
- [7] „Building Generator,“ pastatų generatorius, [Tinkle]. Available: <http://tysonibebe.com/Main/BuildingGenerator/buildingGen.htm>. [Kreiptasi 20 11 2019].
- [8] „BuildR,“ pastatų generatorius, [Tinkle]. Available: <http://support.jasperstocker.com/buildr2/>. [Kreiptasi 20 11 2019].
- [9] „SceneCity,“ miestų generatorius, [Tinkle]. Available: <https://www.cgchan.com/store/scenecity>. [Kreiptasi 20 11 2019].
- [1] P. M. Yoav I H Parish, „Procedural Modeling of Cities,“ [Tinkle]. Available: 0] https://cgl.ethz.ch/Downloads/Publications/Papers/2001/p_Par01.pdf. [Kreiptasi 21 11 2019].
- [1] P. Bourke, „L-System User Notes,“ [Tinkle]. Available: <http://paulbourke.net/fractals/lsys/>. [Kreiptasi 1] 21 11 2019].
- [1] J. P. N. S. G. L. Stefan Greuter, „Real-time Procedural Generation of ‘Pseudo Infinite’ Cities,“ 2] [Tinkle]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.7296&rep=rep1&type=pdf>. [Kreiptasi 21 11 2019].
- [1] M. W. F. X. S. W. R. Peter Wonka, „Instant Architecture,“ [Tinkle]. Available: 3] https://hal.inria.fr/inria-00527500/file/instant_architecture.pdf. [Kreiptasi 22 11 2019].

PRIEDAI

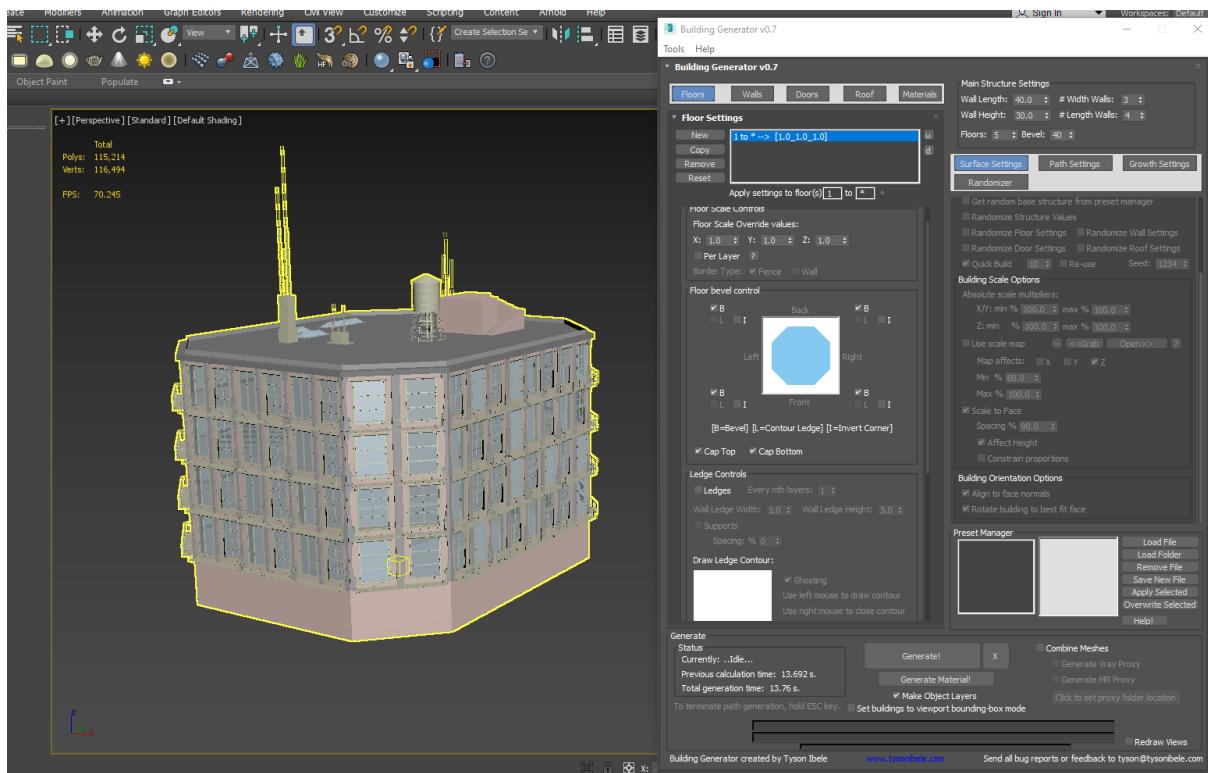
1 PRIEDAS. Pastatų ir miestų generatorių priedai



Priedas 1 Houdini building generator varotojo sąsajos pavyzdys [5]



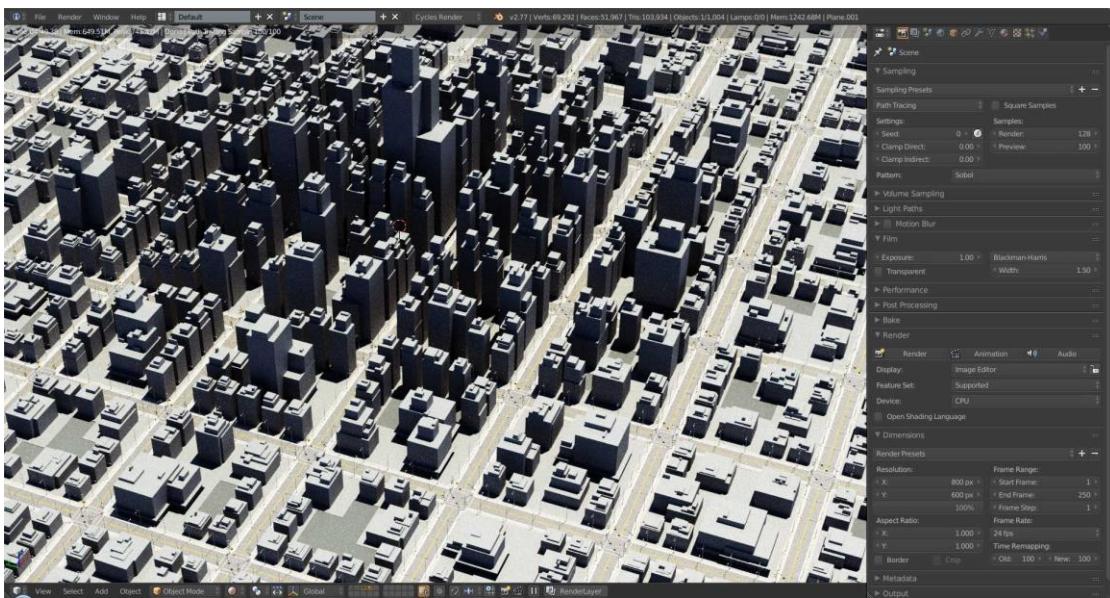
Priedas 2 Maya structures vartotojo sąsaja ir generacijos pavyzdys [6]



Priedas 3 Building Generator v0.7 vartotojo sąsaja ir generacijos pavyzdys



Priedas 4 SceneCity generacijos pavyzdys [9]



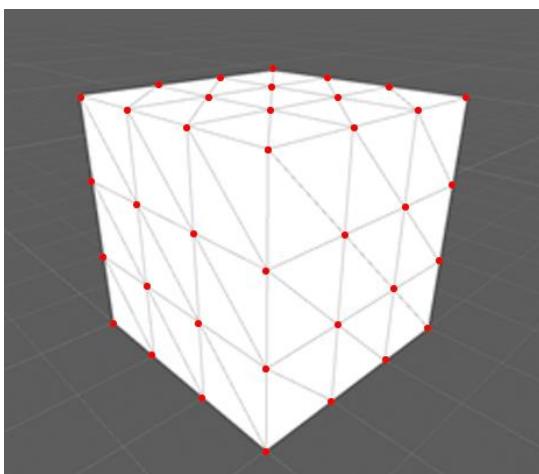
Priedas 5 SceneCity procedūriškai sugeneruoti pastatai [9]

2 PRIEDAS. Generacijos algoritmų priedai



Priedas 6 "Momentinės architektūros" algoritmo generacijos pavyzdys [13]

3 PRIEDAS. Programos architektūros priedai



Priedas 7 bazine figūra su matomais poligonais, trikampiais ir verteksais