

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, 2019

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU

Katedros vedėjas

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Parašas)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Vardas, pavardė)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(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

Programų sistemų studijų kryptis

**Vadovas**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

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

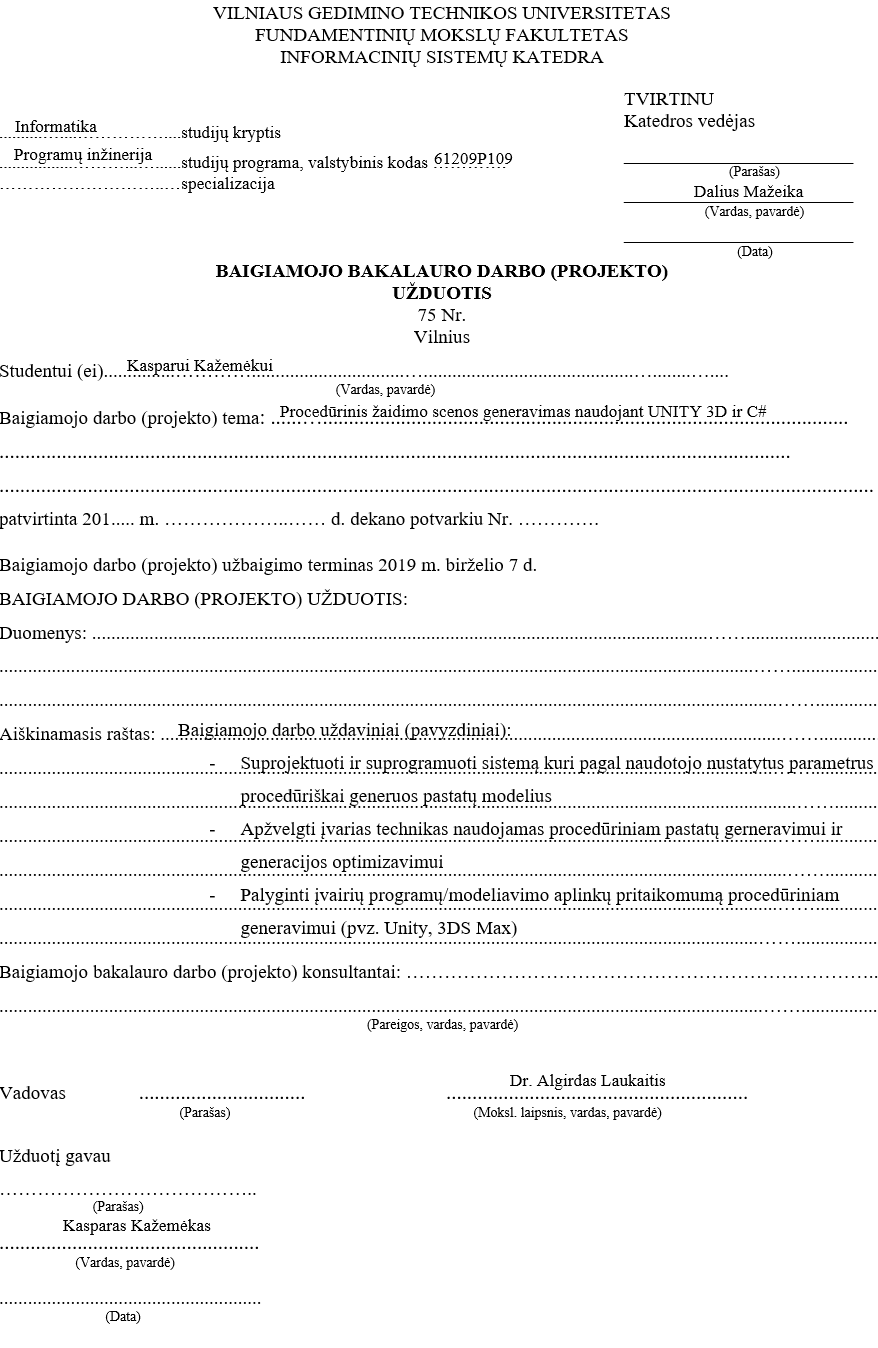
**Konsultantas**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

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

**Konsultantas**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

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

Vilnius, 2019



**Anotacija**

Šioje medžiagoje aprašoma bakalaurų baigiamųjų darbų paskirtis ir turinys, pateikiami bakalaurų baigiamųjų darbų rengimo patarimai, pavyzdžiai ir įforminimo taisyklės.

Be to ši medžiaga gali būti naudojama, kaip šablonas, rengiant baigiamąjį bakalauro darbą, pradedant nuo turinio ir baigiant literatūros sąrašu. Bakalarų baigiamųjų darbų anotacija lietuvių kalba ir anglų kalba čia nepateikiami. Jos turi būti pildomos ir spausdinamos iš mano.vgtu.lt.

**Turinys**

[1. Įvadas 8](#_Toc33275080)

[Darbo tikslas ir uždaviniai 8](#_Toc33275081)

[2. Procedūrinio modelių generavimo technologijų analizė 9](#_Toc33275082)

[2.1. Viduramžių namo architektūrinė analizė 9](#_Toc33275083)

[2.2. Programų apžvalga 10](#_Toc33275084)

[2.2.1. Houdini Building Generator [5] 10](#_Toc33275085)

[2.2.2. Maya Structures [6] 11](#_Toc33275086)

[2.2.3. Building Generator v0.7 [7] 12](#_Toc33275087)

[2.2.4. BuildR 2 [8] 13](#_Toc33275088)

[2.2.5. SceneCity [9] 13](#_Toc33275089)

[2.3. Procedūrinio generavimo algoritmų apžvalga 14](#_Toc33275090)

[2.3.1. L-sistema ir jos plėtiniai 14](#_Toc33275091)

[2.3.2. Greuter metodas 15](#_Toc33275092)

[2.3.3. Momentinė architektūra 16](#_Toc33275093)

[2.4. Skyriaus išvados 18](#_Toc33275094)

[3. Projektinė dalis 18](#_Toc33275095)

[3.1. Rekalavimų specifikacija 18](#_Toc33275096)

[3.1.1. Formuluojamos užduotys 18](#_Toc33275097)

[3.1.2. Funkciniai sistemos reikalavimai 20](#_Toc33275098)

[3.1.3. Nefunkciniai reikalavimai 21](#_Toc33275099)

[3.2. Programų sistemos projektiniai reikalavimai 23](#_Toc33275100)

[3.2.1. Programų sistemos dekompozicija 23](#_Toc33275101)

[3.2.2. Reikalavimų lokalizavimo matrica 25](#_Toc33275102)

[3.2.3. Reikalavimų ryšio matrica 25](#_Toc33275103)

[3.3. Programų sistemos architektūra 27](#_Toc33275104)

[3.3.1. Sistemos užduočių scenarijų vygdymas 27](#_Toc33275105)

[3.3.2. PS struktūrinis modelis 31](#_Toc33275106)

[3.3.3. PS dinaminis modelis 32](#_Toc33275107)

[3.4. Programų sistemos maketai 33](#_Toc33275108)

[4. Testavimas 34](#_Toc33275109)

[4.1. Automatinis testavimas 35](#_Toc33275110)

[4.1.1. Stogo pozicijos apskaičivimo testavimas kai stogas didesnis už viršutinį aukštą. 35](#_Toc33275111)

[4.1.2. Pirmo aukšto pozicijos apskaičiavimo testavimas kai visos ugiasienės išjungtos 35](#_Toc33275112)

[4.1.3. - 35](#_Toc33275113)

[4.1.4. Testavimo scenarijai 35](#_Toc33275114)

[4.1.5. Generacijos rezultato testavimas ir demonstracija 36](#_Toc33275115)

[4.1.6. Reikalavimo, kad sistema iš vartotojo nereikalautų bazinio modelių rinkinio įgyvendinimas 37](#_Toc33275116)

[5. Išvados ir siūlymai 38](#_Toc33275117)

[Literatūra ir šaltiniai 39](#_Toc33275118)

[PRIEDAI 40](#_Toc33275119)

Santrumpų ir terminų žodynas

Verteksas – 3D objekto taškas erdvėje, viršūnė.

Mesh – verteksų tinklas sudarantis 3D objekto formą

Poligonas – Plokštuma gaunama kelis verteksus sujungus kraštinėmis (pavyzdžiui trikampis)

Modelis – 3D objektas.

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

Low poly – 3D grafikos stilius kai kuriami objektai, dėl estetinių ar techninių prižasčių, turi salyginai mažą kiekį poligonų.

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

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

# 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 kiekvienam rinkos segmentui naudojama skirtingo pajėgumo techninė įrangą, 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 populiarius low poly stilius [2] (geri jo pavyzdžiai būtų „Monument Valley“ ir „Astroneer“). Žinoma grafinis paprastumas nereiškia, kad žaidimo aplinką lengvą sukurti, atvirkščiai, 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 bazinę 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 procedūriniame modelių generavime naudojami iš anksto paruošti bazinių modelių rinkiniai (tarkim durys, sienos, langai, stogo segmentai) turintys ir iš anksto sukurtas tekstūras. 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 juo sukūrus vėliau gali būti sudėtinga atlikti pakeitimus. Tokios generacijos pakaitalas – procedūrinis visų objekto dalių generavimas nuo pradžios iki pabaigos įskaitant ir tekstūras, ko ir bus siekiama kuriant šią sistema.

Šiuo metu siūlomi pastatų ir miestų generatoriai arba orientuoti į realistinį šiuolaikinių pastatų generavimą bei universalumą. Akivaizdu, kad universalūs įrankiai primtini 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, low poly grafinio stiliaus populiarumą ir į šiuolaikinę architetūra orientuotus generatorius iškyla 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 bei jų taikymą, sukurt viduramžių pastatų generatoriaus prototipą.

**Tyrimo objektas** – automatiniai virtualių pastatų generavimo metodai.

**Tyrimo problema** – Pastatų generatorių, orientuotų į stilizuotą viduramžišką stilių, trūkumas.

**Darbo uždaviniai:**

1. Apžvelgti rinkoje egzistuojančių pastatų generatorių galimybes;
2. Išnagrinėti populiariausius procedūrinių 3D objektų generavimo metodus;
3. Atlikti pastatų generatoriaus projektavimą, pasirinkti tinkamą programinę įrangą;
4. Atrinkti geriausiai low poly viduramžių pastatų generavimui tinkamus metodus;
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
* Penktas skyrius - išvados

# Procedūrinio modelių generavimo technologijų analizė

## Viduramžių namo architektūrinė analizė

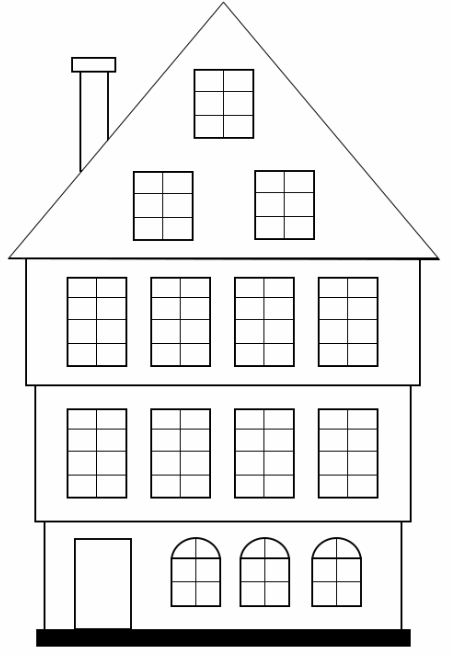
Kuriama sistema orientuosis į viduramžių namų generavimą, tad visų pirma reikia apibrėžti pagrindines tokių namų savybes. Kaip lengvai atpažystamas pavyzdys, savotiškas viduramžių miesto pastato etalonas, pasirinktas vokiškas fachverko stiliaus namas (1 pav.).

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



2.1 pav. Fachverkiniai namai [4]

Taigi, iš analizės galima sudaryti tokį generuojamo namo planą:



2.2 pav. Generuojamo pastato planas

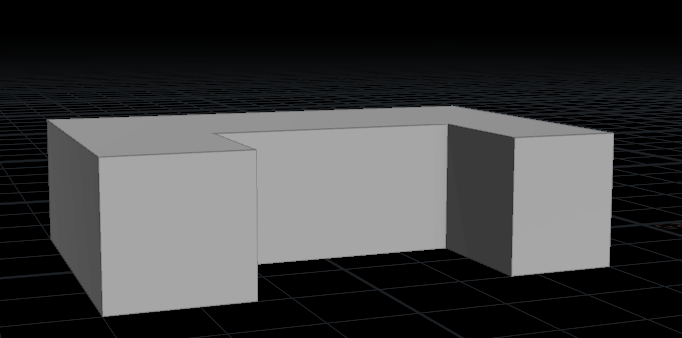
## Programų apžvalga

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

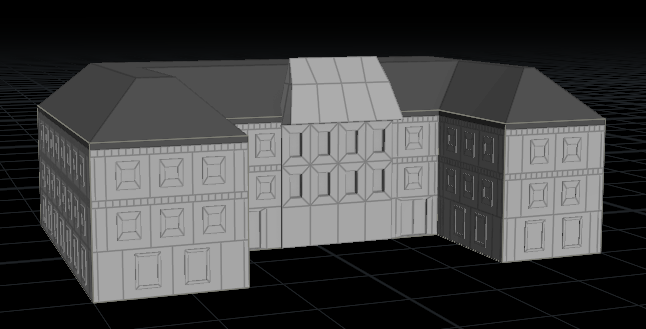
### Houdini Building Generator [5]

1 lentelė. Houdini Building Generator apžvalga

|  |  |
| --- | --- |
| Programa/aplinka kurioje veikia generatorius | **Houdini** |
| Naudojimo licenzijos tipas | Patentuota licencija |
| Kaina | Nemokamas su Houdini, tačiau Houdini licencija kainuoja 0-7000$. |
| Programos valdymas | Valdymui naudojamas vizualus programavimas ir nustatymų langai. |
| Veikimo prinicpas | Generacija vykdoma nustatčius pastato tūrį ir jį užpildant panaudojus iš anksto sukurtus komponentus (langus, duris ir t.t.). Komponentai dėliojami taip, kad būtų išlaikytas vieningas ir logiškas pastato stilius, pavyzdžiui 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]

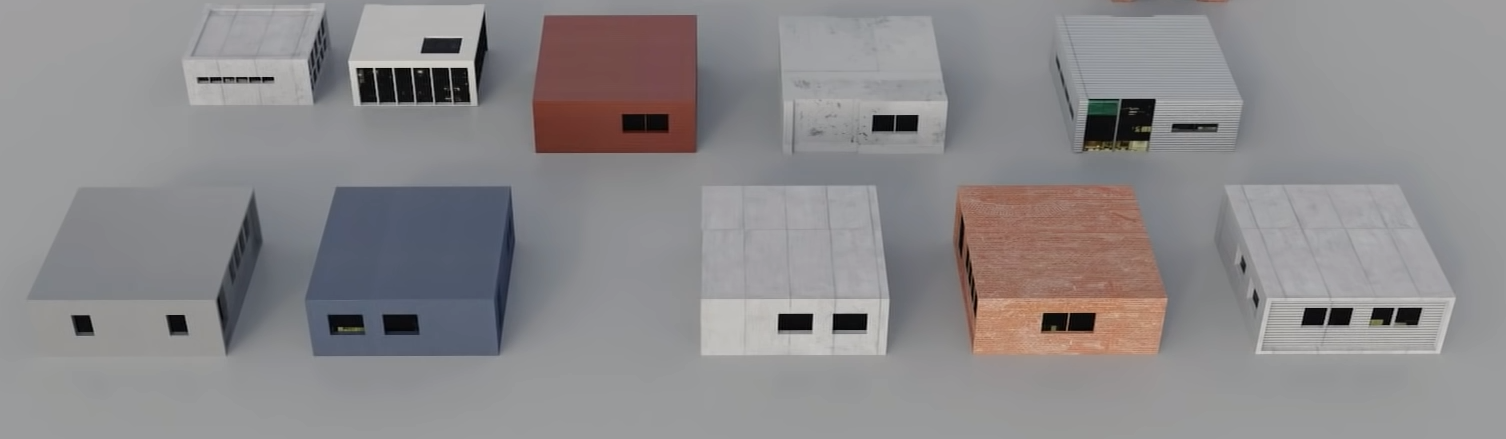


2.4 pav. Sugeneruotas pastatas [5]

### Maya Structures [6]

3 lentelė. Maya Structures apžvalga

|  |  |
| --- | --- |
| Programa/aplinka kurioje veikia generatorius | **3Ds Maya** |
| Naudojimo licenzijos tipas | Patentuota licencija |
| Kaina | ~$60 |
| Programos valdymas | Valdymui naudojami tik nustatymų langai. |
| Veikimo prinicpas | 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. 5 pav). Programa keisdama modelių geometrines savybes (dydį, pasukimą) ir pozicijas sujungia juos į viena, taip sukurdama skirtingai atrodančius pastatus ir mechanizmus. |
| Panaudojimas | Filmų bei 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 smulkių detalių prarandamas mažiems ar senoviniams pastatams reikalingas detalumas.  Valdymas nustatymų langais leidžia lengviau naudoti programą ir nereikalauja papildomų žinių, kaip vizualinį programavimą naudojantys įrankiai. |



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

### Building Generator v0.7 [7]

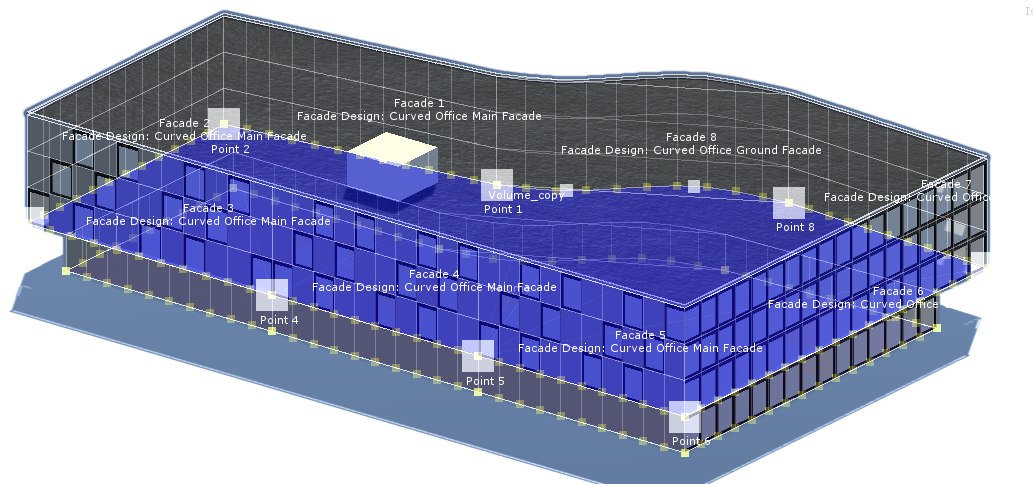
4 lentelė. Building Generator apžvalga

|  |  |
| --- | --- |
| Programa/aplinka kurioje veikia generatorius | **3Ds Max** |
| Naudojimo licenzijos tipas | Atviras Kodas |
| Kaina | Nemokamas |
| Programos valdymas | Valdymui naudojami nustatymų langai |
| Veikimo prinicpas | Generacija, kaip ir Houdini generatoriuje, vykdoma panaudojant iš anksto sukurtų pastato komponentų (langų, durų ir t.t.) rinkinį. Tačiau skirtingai nei Houdini generatoriuje, šiame neįmanoma nustatyti ivarių formų pradinio tūrio, pastatas visuomet yra stačiakampis. Programa keisdama modelių pozicijas ir pasukimo kampą sujungia juos į viena, taip sukurdama skirtingai atrodančius pastatus. Pastatų išvaizda kuriama nustant įvairius parametrus, tokius kaip: aukštų skaičius, balkonų, langų išvaizda, pastato forma ir t.t. |
| Panaudojimas | Filmų bei žaidimų aplinkų kūrimas, kompiuterinio meno kūrimas, prototipavimas. |

### BuildR 2 [8]

5 lentelė. BuildR 2 apžvalga

|  |  |
| --- | --- |
| Programa/aplinka kurioje veikia generatorius | **Unity** |
| Naudojimo licenzijos tipas | Patentuota licencija |
| Kaina | €88 |
| Programos valdymas | Valdymui naudojami nustatymų langai. |
| Veikimo prinicpas | Įrankis leidžia kurti salyginai primytiviai atrodančius pastatus, kadangi kūrimas susideda iš paprastų 3d modeliavimo operacijų – tokių kaip papildomų verteksų sukūrimas ar tūrio ištraukimas (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 smulkių detalių.  Objekto skaidymas į segmentus leidžia keisti kiekvieno elemento tekstūras atskirai. |



2.6 pav. BuildR 2 generacijos pavyzdys su matomais verteksais

### SceneCity [9]

6 lentelė. SceneCiy mietų generatoriaus apžvalga

|  |  |
| --- | --- |
| Programa/aplinka kurioje veikia generatorius | **Blender** |
| Naudojimo licenzijos tipas | Patentuota licencija |
| Kaina | $97 |
| Programos valdymas | Valdymui naudojamas vizualus programavimas ir nustatymų langai. |
| Veikimo prinicpas | Įrankis leidžia generuoti logiškai išdestytą kelių tinklą, realistiškai atrodantį žemės paviršių ir išdėlioti tūkstančius pastatų. Įrankis gaunamas su paruoštais pastatų pavyzdžiais, taičiau palaiko ir naudotojo sukūrtus pastatus. Pastatai išdėliojami keičiant tik jų pasukimą, bet neliečiant jų formos.  Skirtingai nei kiti apžvelgiami įrankiai, šis neskirtas pilnai proceduriškai 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 pasiekiama „sukarpant“ ir „suklijuojant“ segmentus iš iš anksto paruoštų tekstūrų failų su dideliu kiekiu įvariai atrodančių elementų, pavyzdžiui langų, durų, stogų ir t.t. |
| Panaudojimas | Filmų bei žaidimų aplinkų kūrimas, kompiuterinio meno kūrimas, prototipavimas |
| Komentarai | Įrankis turi Unity žaidimų variklio papildinį leidžiantį sugeneruotus miestus lengvai panaudoti žaidimų kūrime. |

## Procedūrinio generavimo algoritmų apžvalga

### L-sistema ir jos plėtiniai

L-sistema tai septintame praėjusio amžiaus dešimtemetyje vengrų biologo Aristrid Lindenmayerio sukurtas rekursinis modelis aprašantis augalų vystymasi. Dėl sistemos paprastumo ir universalumo ji prigijo ir matematikoje. Sistema tai formali kalba kuri yra aprašoma kaip eilė simbolių, kurių kiekvienas reiškia konkrečią komandą, taip pat sistemoje naudojama eilė parametrų [10]. Kad būtų pasiektas rezultatas, sistema iteruoja per simbolius nustatytą skaičių kartų, kiekvienos iteracijos rezultatui taikydama tas pačias pradžioje aprašytas komandas bei pasinadodamas duodamais parametrais. Pavyzdys [11]:

7 lentelė. Simbolių reikšmės

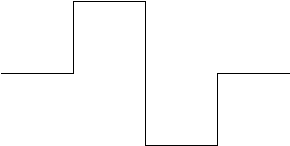
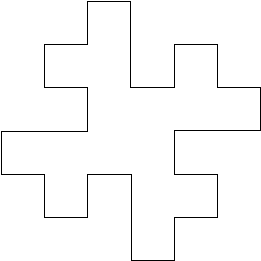
|  |  |
| --- | --- |
| Simbolis | Reikšmė |
| F | pajudėti per linijos ilgį ir nupiešti liniją |
| - | Pasisukti į kairę per nurodytą kampą |
| + | Pasisukti į dešinę per nurodytą kampą |

Aksioma = F+F+F+F

F--> F+F-F-FF+F+F-F

Kampas = 90°

Pradinė šio pavydžio simbolių eilutė(aksioma) duoda kvadratą, o pakaitinė eilutė grafiškai pavaizduota 6 pav. Po vienos iteracijos gaunamas 7 pav. vaizdas.

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 naudoja stochastinę, parametrinę L-Sistemą. Jų L-Sistemos objektai tai 3d objektų transformacijos operacijos, tokios kaip dydžio keitimas ar pajudinimas, tūrio ištraukimas (extrusion), išsišakojimas ir panaikinimas bei sudėtingesniems architektūriniams elementams, tokiems kaip stogai, langai ar antenos iš anksto paruošti modeliai.

2.7 pav. F--> F+F-F-FF+F+F-F grafinis vaizdas [11]

Tokios sistemos generacijos pavyzdys matomas 12 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 nusiunčiama į intepretatorių kuris simbolių eilutę paverčia į 3d objektus;

2.8 pav. Vienos iteracijos rezultatas [11]



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

### 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 įvairiais pastato charakteristikas: aukštį, aukštų skaičių ir t.t. [12]. Kadangi jų tyrime 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 ivairių formų poligonai kurių išvaizdą ir vietą nusako skaičių generatoriaus išeiga (grindų ploto generacija matoma 9 pav.). Atlikus nustatytą vieno aukšto grindų plano keitimo iteracijų kiekį, grindų plokštuma yra ištraukiama (extrude) paverčiant ją 3D tūriu, taip baigiant vieno aukšto generaciją [12]. Aukštai generuojami tol, kol pasiekiamas nustatytas pastato aukštis (aukštų generacija 10 pav.).

Kad geriau iliustruoti algoritmo veikimą, pateikiu pseudo kodą:

var pastatas;

ciklas(aukštų kiekis){

var grinduPlanas;

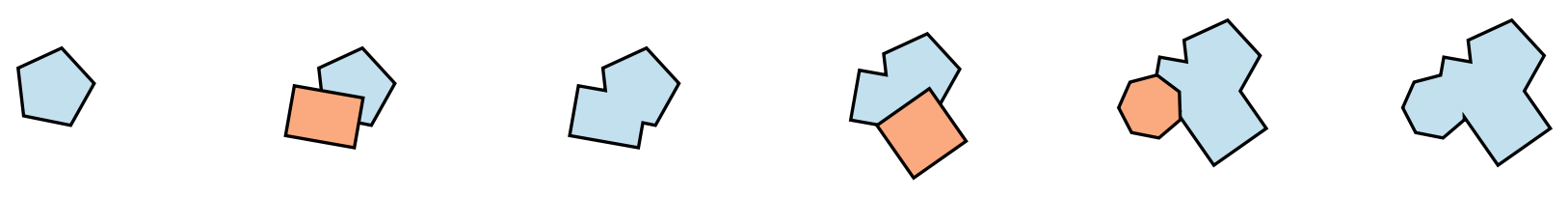
ciklas(iteracijų skaičius aukšte){

grinduPlanas = PridetiFigura(grinduPlanas);

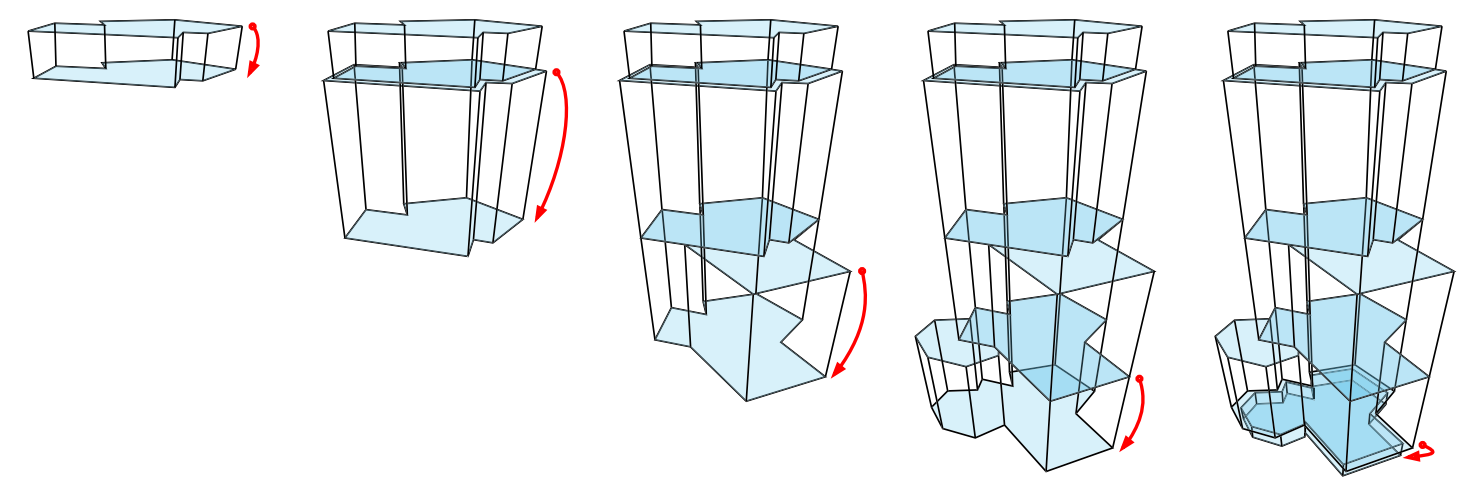
}

pastatas += IstrauktiTuri(grinduPlanas);

}



2.10 pav. Grindų plano generacija [12]

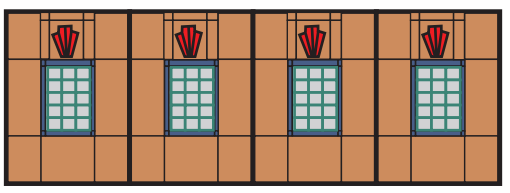
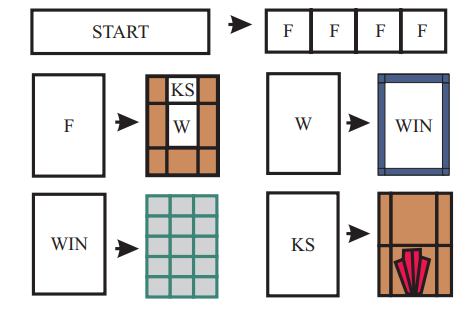


2.11 pav. Pastato aukštų generacija [12]

### 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 segementai pakeičiami kitomis figūromis kurios gali būti toliau skaidomos ir keičiamos kol pasiekiama kiekvienos figūros galutinė (nebeskaidoma) būsena. Sistema palaiko skirtingus architektūrinius stilius ir pastatų generaciją iš įvairiai sudeliotų stačiakampių, prizmių ir cilindrų. [13]

Pastatas paprastai sudeda is kelių pradinių figūrų, pavydžiui 2 aukštai po 4 stačiakampius. Tuomet pasitelkę 11 paveikslėlyje matomą antro aukšto generacijos vizualizaciją, matome kaip pradinis vieno stačiakampio plotas (F) skaidomas į tinklą kuriame išskiriami plotai langui (W) ir dekoracijai (KS). Tuomet W plotas skaidomas į rėmą ir patį lango stiklą, kas yra galutinės šių figūrų būsenos. KS parenkama stilių atitinkanti dekoracija ir taip pasiekiama šios dalis galutinė būsena. 12 paveikslėlis parodo gaunamą antro aukšto rezultatą.

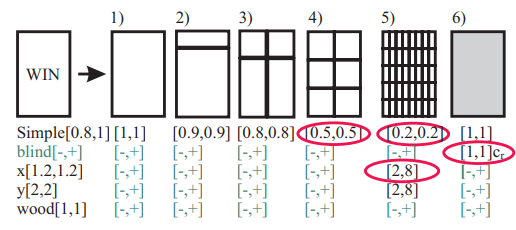


2.12 pav. Pradinio tūrio segmento skaidymo pavyzdys [13]

2.13 pav. Pradinio tūrio skaidymo pavyzdžio rezultatas [13]

Natūralu, kad generacija būtų sėkminga, algoritmas turi išlaikyti vieningą pastato stilių ir logiškai išdelioti architektūrinius elementus (duris, langus, stogą ir t.t.). Algoritmas turi sėkmingai atsirinkti taisyklęs iš visų tai būsenai pritaikomų taisyklių rinkinio. Tai pasiekiama 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 parametrų perdavimą, tarp to pačio aukšto ar viso statinio segementų, atsakingos kontrolinės gramatikos taisyklės. Jų iškvietimas reguliuojamas specifinių parametrų prijungtų prie kiekvieno tūrio. Prieš kievieną tūrio skaidyma iškviečiamos kontrolinės taisyklės kurios padalina tėvinio tūrio parametrus visoms joms priklausančios erdvės (pvz. vienam aukštui) naujai suformuotoms figūroms (pvz. to aukšto langams).

Panagrinėkime konkretų objektų atrankos pavyzdį (13 pav.). Kaip matome yra išskirtas naujas plotas langui (WIN) su 5 parametrais (simple, blind, x, y, wood). Langams egzistuoja 6 skirtingi dizainai. „Simple“ parametras turi intervalą [0.8,1] kuris leidžia atmesti 4 ir 5 dizainus. Blind parametras su intervalu [−∞,∞], leidžia atmesti 6 dizainą. X pakartotinai atmeta 5 dizainą. Taigi lieka 3 lango variantai, iš kurių atsitiktinai galima pasiimti betkurį. Taigi tokia pastato segmentų išvaizdos atranka užtikrina tiek salyginai atsitiktinę pastato išvaizdą, tiek taisykles atitinkantį stilių.



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

## 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 įvairioms platformoms. Tiesa, visi šie įrankiai stengiasi būti kuo universalesni ir fokusuojasi į modernių, realistiškų pastatų generavimą. Atsižvelgus į minėta paprastų grafikų, low poly grafikų populiarumą, matosi, kad apžvelgti įrankiai neužpildo šio rinkos segmento. Taip pat, šie įrankiai naudotojo reikalauja išankstinio bazinių dalių rinkinio, kas su modeliavimu nesusipažinusiam gali tapti neįveikiama kliūtimi

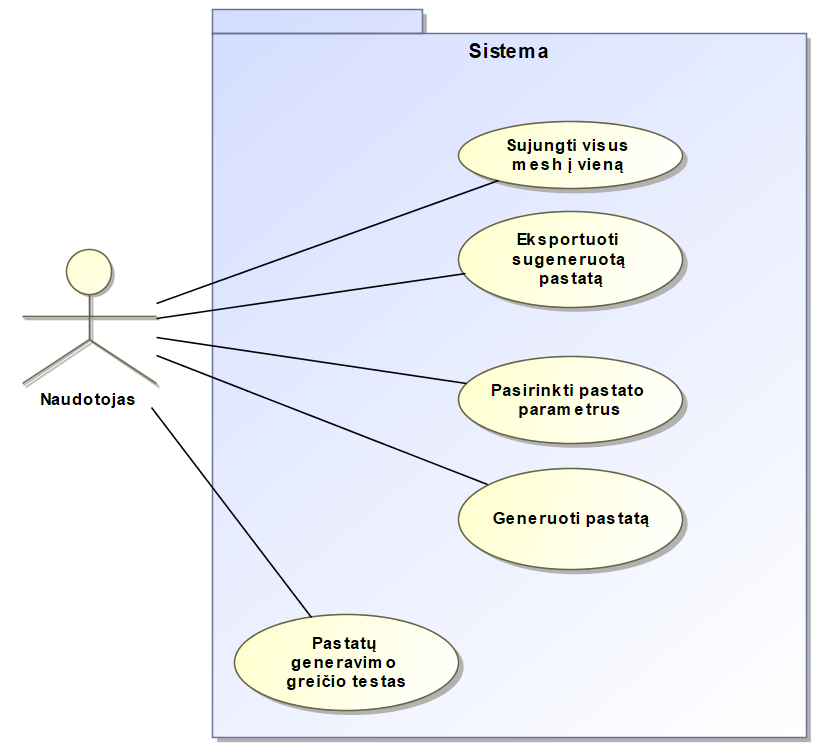
Taigi, šiame darbe pristatoma ir kuriama sistema bus orientuotą atlikti konkretų vaidmenį – generuoti low poly stilistikos viduramžių pastatus, nenaudodama jokių iš anksto sukurtų modelių rinkinio ir nereikalaudama rankinio tekstūravimo.

# Projektinė dalis

## Rekalavimų specifikacija

### Formuluojamos užduotys

#### Pagrindinės užduotys



Pagrindinės sistemos atliekamos užduotys:

* + - * 1. Pastato generavimas;
        2. Mesh sujungimas į vieną;
        3. Sugeneruoto pastato eksportavimas;
        4. Generavimo parametrų nustatymas;

Pagalbinės užduotys:

* + - * 1. Pastatų generavimo greičio testas

#### Užduočių formulavimo kalbos reikalavimai

Vartotojo sąsaja yra Unity Editor plėtinys.

Vartotojo sąsaja suseda iš:

1. Nustatymų lango kuriame yra sudėtas visas vartotojo kontroliuojuomas 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 meshus 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 tekstūrai

#### Interfeiso darnos ir standartizavimo rekalavimai

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

Interfeisas kuriamas anglų kalba.

#### Pranešimų formulavimo reikalavimai

Pranešimai sistemoje skirstomi į dvi kategorijas: informacinius ir klaidos. Pavyzdžiai:

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

Pranešimai formuluojami anglų kalba.

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

#### Interfeiso individualizavimo reikalavimai

* Ugniasienės – leidžia pažymėti kurie pastato šonai bus ungiasienės.
* Šviečiančių langų išdėliojimo stilius – galima pasirinkti, arba, kad kievienas auštas turėtų vienodai šviečiančius langus, arba atstiktinai 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.

### Funkciniai sistemos reikalavimai

#### Dalykiniai 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 ekportuoti sugeneruotą pastatą universaliu formatu | | | | |
| **RNr. 4** | **Statusas – E** | **Galiojimo laikas – S** | **Kritiškumo laipsnis – S** | **Užduoties Nr. 3.1.1.1.4** |
| Sistema turi leisti pasrinkti parametrus pagal kuriuos bus generuojamas pastatas | | | | |

#### Pagalbinės sistemos funkcijos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RNr. 5** | **Statusas - O** | **Galiojimo laikas – U** | **Kritiškumo laipsnis – L** | **Užduoties Nr. 3.1.1.1.5** |
| Sistema turi turėti didelio kiekio pastatų generavimo greičio testą | | | | |

### Nefunkciniai reikalavimai

#### Vidinio interfeiso reikalavimai

##### Operacinės sistemos naudojimo reikalavimai:

Sistemai kurti turi būti naudojama Windows operacinė sistema;

##### Sąveikos su duomenų bazėmis reikalavimai:

Reikalavimų nėra

##### Dokumentų mainų reikalavimai:

Reikalavimų nėra

##### Darbo kompiuterių tinkluose reikalavimai:

Reikalavimų nėra

##### Sąveikos su kitomis programomis reikalavimai:

Reikalavimų nėra

##### Programavimo aplinkos reikalavimai:

Sistema kuriama Unity 2019.2 aplinkoje naudojant Visual Studio programavimo aplinką;

#### Veikimo reikalavimai

##### Tikslumo reikalavimai:

Ilgio matai – metrai ir centimetrai;

Nustatomo namo pagrindas turi būti tarp 2x2 ir 10x10 dydžio ;

Namas gali turėti tarp 1 ir 6 aukštų;

Namas gali turėti iki 3 ugniasienių;

##### Patikimumo reikalavimai:

Sistema turi pilnai veikti 95% laiko be trykių.;

##### Robastiškumo reikalavimai:

Trykius iššaukiančių įvykių procentas turi būti 1%.;

##### Našumo reikalavimai.

##### Reakcijos laikas:

Vidutinis pastato generavimo laikas neturi viršyti 100ms;

##### Pralaidumas (throughput):

Sistema vienu metu generuoja viena pastatą;

##### Masto keitimas (scalability):

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

#### Diegimo reikalavimai

##### Instaliuojamumas

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

##### Įsisavinamumas:

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

##### Išmokstamumas

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

#### Ruošinio reikalavimai

* + - * 1. Sistema negali reikalauti naudotojo bazinio generacijai naudojamo objektų rinkinio, jis turi būti iš anksto realizuotas naudojamos sistemos kode.

#### Aptarnavimo ir priežiūros reikalavimai

##### Taisomumo

Trikio ištaisymas turėtų užimti ne ilgiau nei 5min.

##### Keičiamumo

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

##### Plečiamumo

Reikalavimų nėra

##### Perkeliamumo

Reikalavimų nėra

##### Testuojamumo

Sistemos testų kūrimui neturi būti skiriama daugiau savaitės

#### Tiražuojamumo reikalavimai

Reikalavimų nėra

#### Apsaugos reikalavimai

Reikalavimų nėra

#### Juridiniai reikalavimai

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

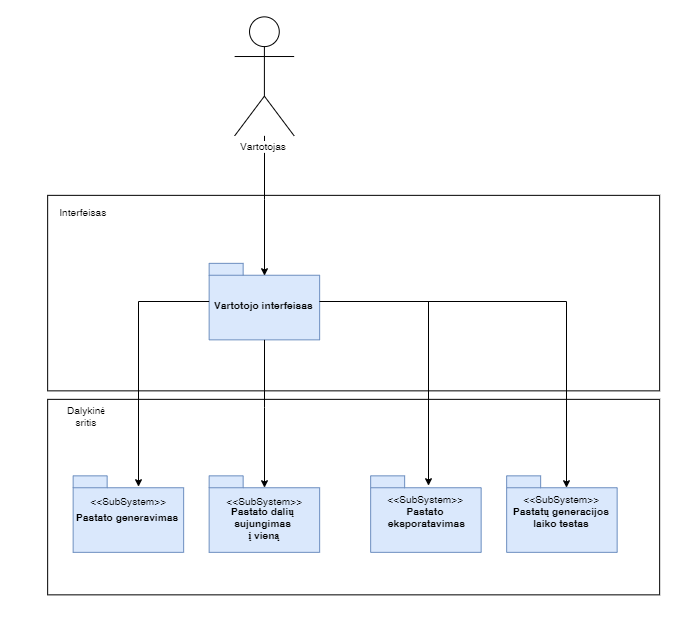
## Programų sistemos projektiniai reikalavimai

### Programų sistemos dekompozicija

#### Skaidymas į paketus

Sistema skaidoma į du paketus:

* Interfeiso
* Dalykinį



**Interfeiso paketo reikalavimai**

RNr. 4 Pastato generavimo parametrų valdymas:

* + - 1. Interfeisas turi būti įgyvendintas laikantis MS Windows standartų;
      2. Interfeisas turi veikti Unity 2019.2 aplinkoje ir panaudoja jos teikiamus langus rodyti generacijos rezultatui ir pranešimam;
      3. Sukurtas interfeisas susideda iš nustatymų lango;
      4. Interfeise yra pastato generacijos nustatymus leidžiantys keisti UI elementai;
      5. Interfeise yra visas sistemos užduotis leidžiantys įgyvendinti mygtukai
      6. Sugeneruotas pastatas turi būti matomas vartotojo sąsajoje

**Dalykinio paketo reikalavimai**

Dekomponuojama į:

RNr. 1 Pastato generavimas:

* + - 1. Turi būti įmanoma generuoti pastatą su ugniasienėmis norimose namo pusėse;
      2. Turi būti įmanoma generuti pastatą su visiškai atsitiktinai apšviestais langais arba apšviestu ištisu aukštu;
      3. Turi būti įmanoma generuoti pastatą su norimu skaičiumi aukštų;
      4. Turi būti įmanoma generuoti pastatą su iš anksto nustatytu pamatų dydžiu;
      5. Turi būti įmanoma generuoti pastatą kurio aukštai vienodo dydžio arba kiekvienas vis didesnis už buvusį;
      6. Turi būti įmanoma generuoti pastatą su angomis kurios yra tik arkinės,tik stačiakampės arba atsitiktinai vieno iš šių stilių;
      7. Turi būti įmanoma generuoti pastatą kurio kampuose, jei pasirinkta, būtų kolonos

RNr. 2 Pastato dalių sujungimas į vieną:

* + - 1. Pastato segmentai, nekeičiant jų geometrijos, turi būti sujungiami į vieną mesh;
      2. Sujungtas pastatas turi būti naujas objektas;
      3. Sujungtas objektas naudoja tą pačią tekstūrą kaip ir nesujungto pastato segmentai;
      4. Po sujungimo ištrinamas senas pastatas iš kurios segmentų sukurtas sujungtas pastatas.

RNr. 3 Pastato eksportavimas:

* + - 1. Turi būti galima eksportuoti tiek į vieną sujungtą objektą tiek atskirų meshų rinkinį;
      2. Objektas eksportuojamas .obj formatu;
      3. Eksportuojamas objektas talpinamas interfeise nurodytoje vietoje;

RNr. 5 Pastato generacijos laiko testas:

* + - 1. Turi būti sugeneruojamas kode nustatytas skaičius pastatų apskaičiuojant vidutinį pastato generacijos laiką;
      2. Pastatų generuojami tokie pagal interfeise pasirinktus nustatymus;

### Reikalavimų lokalizavimo matrica

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Reikalavimai | Vartotojo interfeisas | Dalykinė sritis | Reikalavimai | Vartotojo interfeisas | Dalykinė sritis |
| 3.2.1.2 | X |  | 3.2.1.13 | X | X |
| 3.2.1.3 | X |  | 3.2.1.14 | X | X |
| 3.2.1.4 | X |  | 3.2.1.15 |  | X |
| 3.2.1.5 | X |  | 3.2.1.16 |  | X |
| 3.2.1.6 | X |  | 3.2.1.17 |  | X |
| 3.2.1.7 | X | X | 3.2.1.18 |  | X |
| 3.2.1.8 | X | X | 3.2.1.19 |  | X |
| 3.2.1.9 | X | X | 3.2.1.20 |  | X |
| 3.2.1.10 | X | X | 3.2.1.21 | X | X |
| 3.2.1.11 | X | X | 3.2.1.22 |  | X |
| 3.2.1.12 | X | X | 3.2.1.23 |  | X |

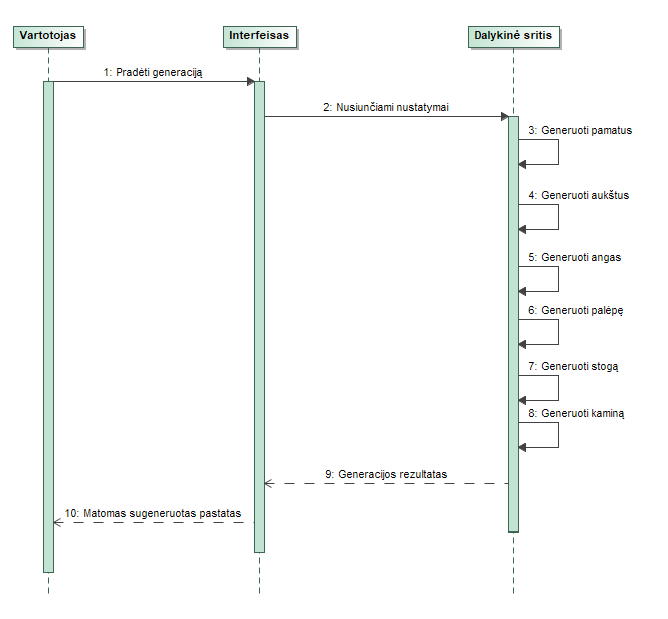
### Reikalavimų ryšio matrica

|  |  |  |
| --- | --- | --- |
| 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. 1 | Dalykinė sritis |
| 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. 2 | Dalykinė sritis |
| 3.2.1.16 | RNr. 2 | Dalykinė sritis |
| 3.2.1.17 | RNr. 2 | Dalykinė sritis |
| 3.2.1.18 | RNr. 2 | Dalykinė sritis |
| 3.2.1.19 | RNr. 3 | Dalykinė sritis |
| 3.2.1.20 | RNr. 3 | Dalykinė sritis |
| 3.2.1.21 | RNr. 3 | Dalykinė sritis |
| 3.2.1.22 | RNr. 5 | Dalykinė sritis |
| 3.2.1.23 | RNr. 5 | Dalykinė sritis |

## Programų sistemos architektūra

### Sistemos užduočių scenarijų vygdymas

##### 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;

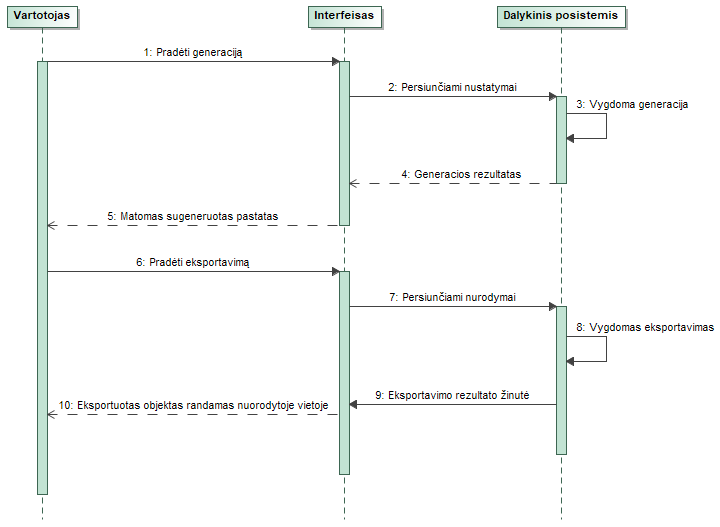
**„Prieš“ sąlygos:** Vartotojas yra įjungęs ir paruošęs naudojimui sistemą;

**„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 varotojo interfeise.

##### 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;

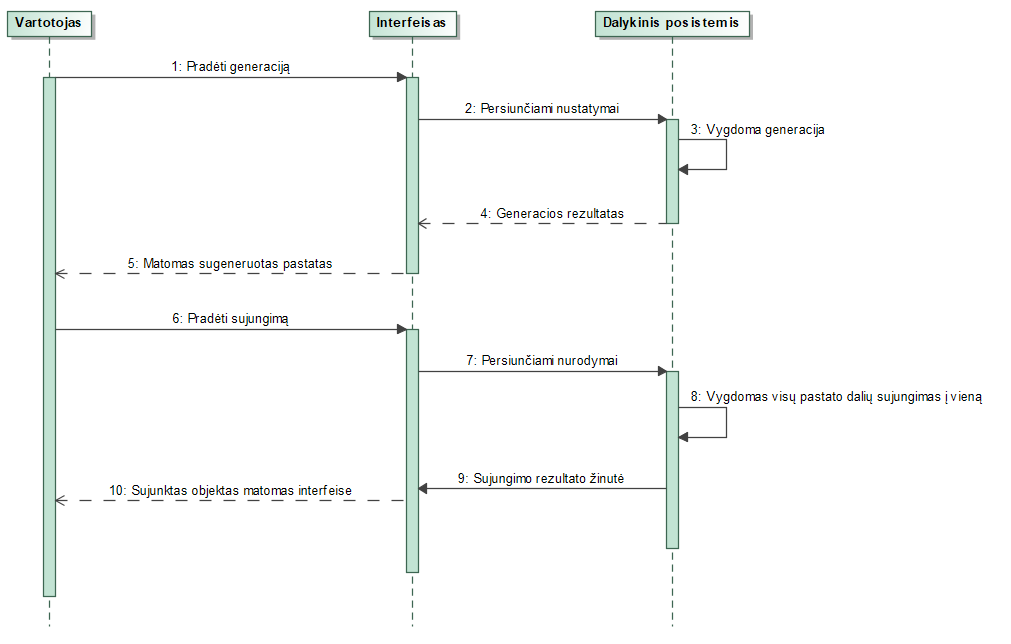
**„Prieš“ sąlygos:** Vartotojas generatoriaus interfeisę įrašęs kurioje vietoje bus talpinamas eksportuotas pastato modelis;

**„Po“ sąlygos:** Išeksportuotas pastatas .obj formatu;

**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 varotojo interfeise.
5. Generatoriaus interfeise paspaudžiamas eksportavimo mygtukas.
6. Dalykinė posistemė ekportuoja sugeneruotą pastatą į vartotojo nurodytą vietą.

##### Užduoties „Sujungti visus mesh į vieną“ įgyvendinamumas



**Scenarijus:** Pastato dalių sujungimas į vieną objektą.

**Versija:** 1.0

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

**Pirmas agentas:** Vartotojas;

**Antriniai agentai:** dalykinė, interfeiso posistemės;

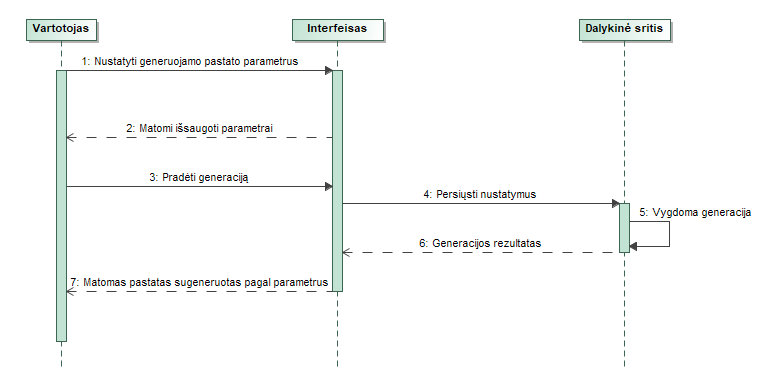
**„Prieš“ sąlygos:** Vartotojas nori sujungti pastatą;

**„Po“ sąlygos:** Sugeneruoto pastato dalys(mesh) sujungtos 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 varotojo interfeise.
5. Generatoriaus interfeise paspaudžiamas mesh sujungimoo mygtukas.
6. Dalykinė posistemė sujungta pastato dalis į vieną objektą.
7. Sujungtas pastatas atvaizduojamas vartotojo interfeise.

##### Užduoties „Generavimo parametrų 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;

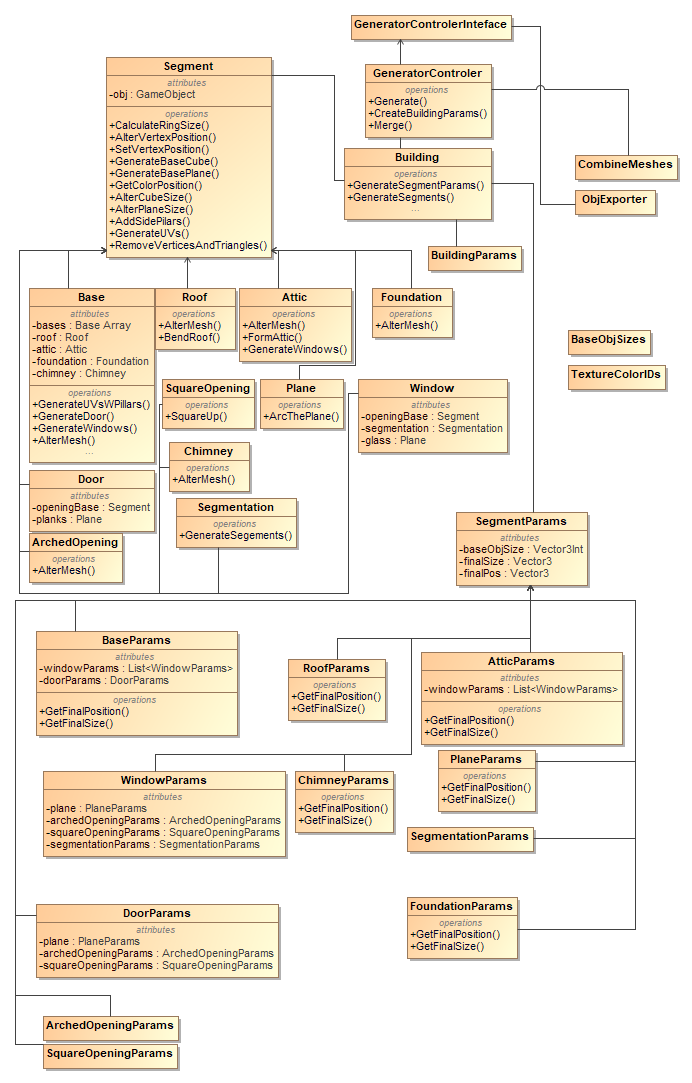
**„Prieš“ sąlygos:** Vartotojas yra įsijungęs ir paruošęs darbui sistemą, bei žino kokio pastato nori;

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

**Scenarijus:**

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

### PS struktūrinis modelis

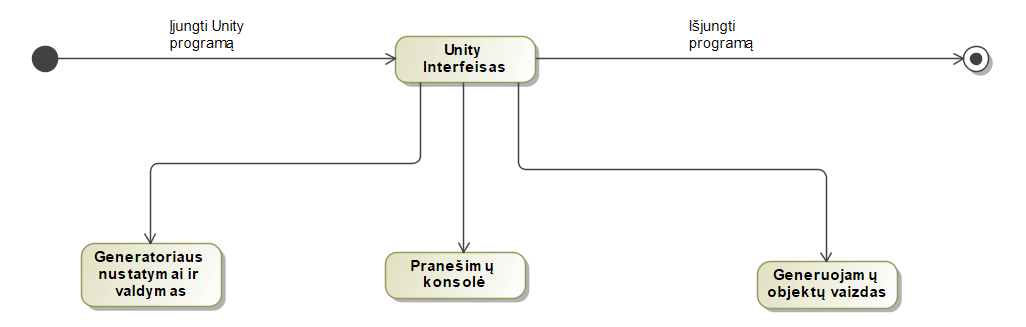


Klasių struktūra:

* GeneratorInterface – skirta Unity aplinkoje atvaizduoti įrankio interfeisą
* GeneratorControler – atsakingas už interfeise atliekamų veiksmų perdavimą dalykinei sričiai
* Building – už pastato generaciją atsakinga klasė, turi ryši su visomis pastato dalimis. Pačio pastato klasių sistema susideda iš:
  + BuildingParams – visam pastatui galiojančios taisyklės (ugniasienės, aukštų skaičius, angų stilius ir t.t.).
  + SegmentParams – tėvinė klasė iš kurios paveldi kiekvieno pastato segment parametrų vaikinė klasė. Skirta generuoti ir saugoti visus tos pastato dalies parametrus (aukštį, dydį, poziciją ir t.t.)
  + Segment – tėvinė klasė iš kurios paveldi kiekvieno pastato segmento vaikinė klasė. Skirta generuoti ir saugoti pačio segmento geometriją.
* CombineMeshs – sujungia pastato segmentus į vieną objektą.
* ObjExporter – eksportuoja sugeneruotą objektą .obj format.
* Sistema taip pat turi statines klases skirtas globaliems parametrams saugoti ir naudoti visoje sistemoje:
  + BaseObjSize – saugo kiekvieno pastato segmento bazinį dydį (poligonų skaičių kiekvienoje ašyje).
  + TextureColorIDs – saugo kiekvieno pastato segemento spalvos koordinatę tekstūroje.

Ši klasių struktūra leidžia į generaciją lengvai pridėti naujus segmentus, kadangi segmentų parametrai ir generavimas visiškai atskirti vienas nuo kito. Geravimo proceso atskirimas nuo parametrų taip leidžiai turėti skirtingos išvaizdos tuos pačius segmentus (pavyzdžiui langus).

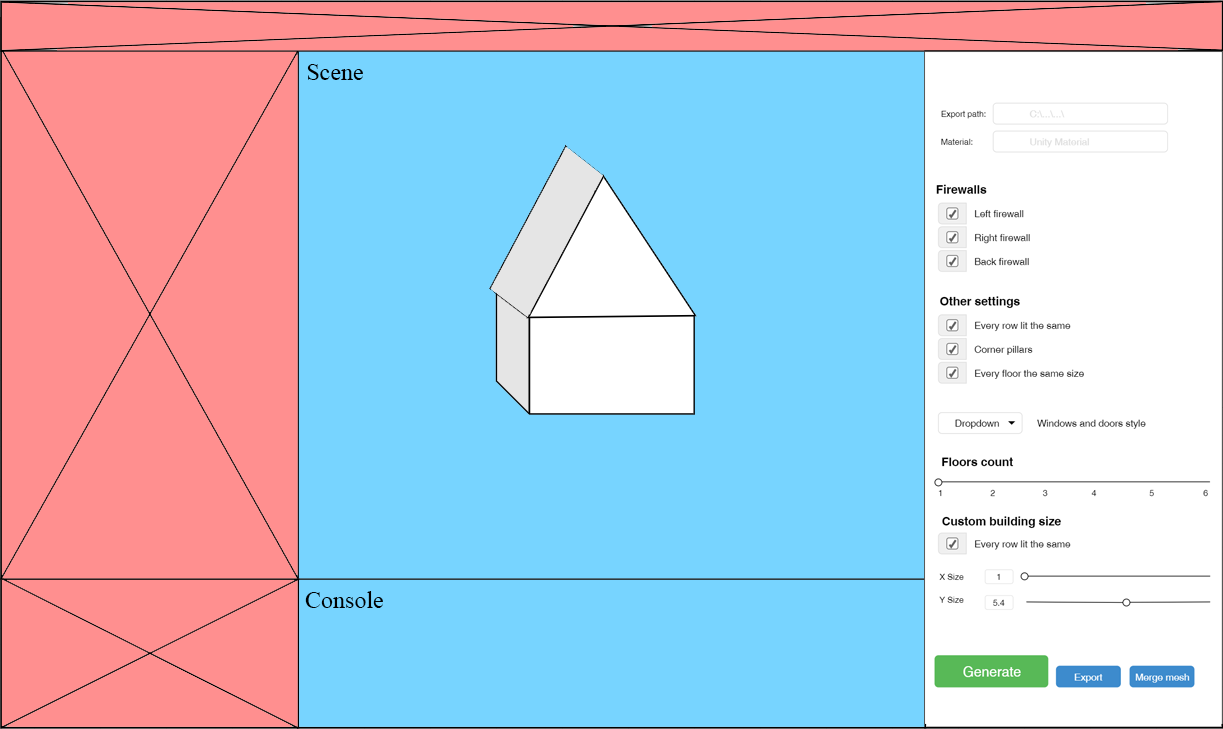
### PS dinaminis modelis



Pati sistema veikia “Unity” 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.

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

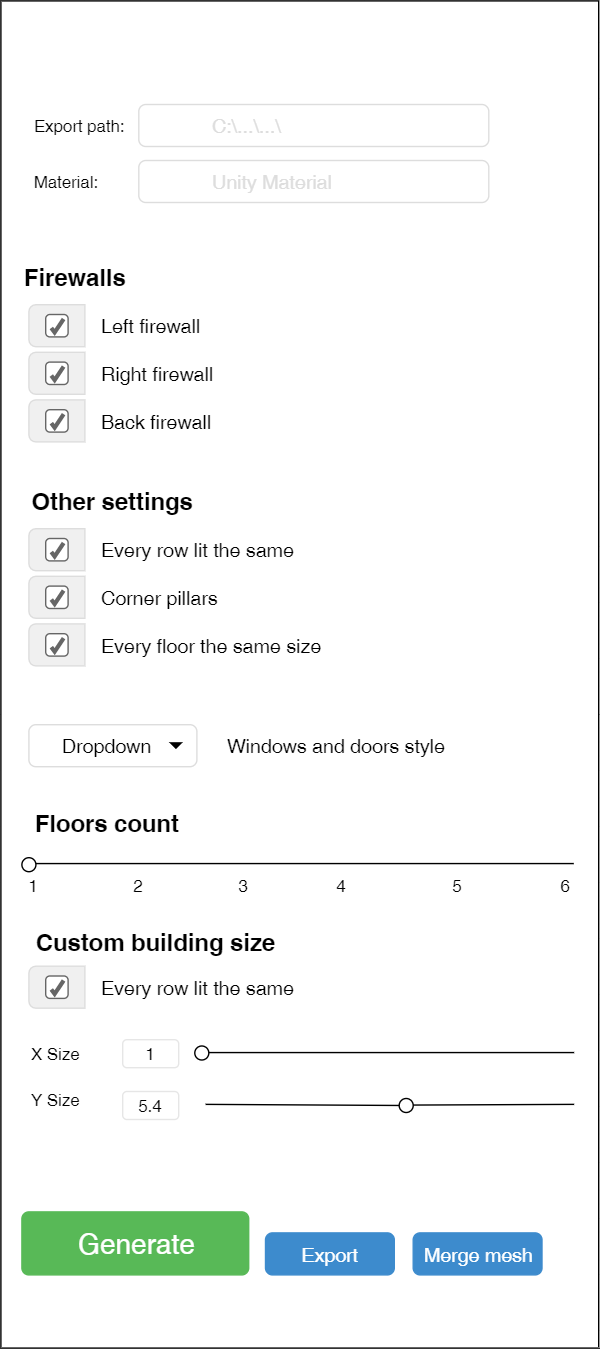
## Programų sistemos maketai



Maketas simbolizuoja Unity vartotojo aplinką kurioje ir veikia kuriama sistema. Maketo dalių reikšmės:

* Raudona – kuriamam įrankiui nesvarbios Unity interfeiso dalys.
* Mėlyna – Unity egizstuojančios ir kuriamos sistemos naudojamos interfeiso dalys
  + Scene – pastatų generatoriaus išeiga 3D formatu
  + Console – pranešimų langas
* Balta – kuriamas generavimo nustatymų langas

Žemiau pateikiamame makete matomas 3.1.1.2 poskyryje aprašyta vartotojo sąsaja (geravimo nustatymų langas).



# Testavimas

Programos testavimas skirtas programos kokybės ir atitikimo projektinėje dalyje aprašytiems reikalavimams nustatymui. Jis suteikia informaciją apie programą veikiačią nustatytoje aplinkoje.

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

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

## Automatinis testavimas

Unit testavimui naudojamas Unity aplinkoje esantis „Test runner“. Testai rašomi pasinaudojus NUnit C# biblioteka.

Unit testavimui buvo pasirinktos BaseParams ir RoofParams klasės ir jose easantys objekto dydžio bei pozicijos apskaičiavimo metodai.

Viso buvo sukuri 16 unit testų kurių dalį apžvelgsiu.



Pav. 4.1.1 Unity Test Runner aplinka

### Stogo pozicijos apskaičivimo testavimas kai stogas didesnis už viršutinį aukštą.

[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));

}

### Pirmo aukšto pozicijos apskaičiavimo testavimas kai visos ugiasienės išjungtos

[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 expextedPos = new Vector3(-0.5f, 2, -0.5f);

Vector3 pos = baseParams.GetGroundFloorFinalPosition(lastFloorSize);

Assert.That(pos.x, Is.EqualTo(expextedPos.x).Within(0.001));

}

### -

### Testavimo scenarijai

Scenarijų svarbos lygiai:

* Kritinė
* Aukšta
* Vidutinė
* Maža

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenarijaus Nr.** | **Scenarijų atitinkančio reikalavimo Nr.** | **Scenarijaus aprašymas** | **Scenarijaus svarba** |
| **TS1** | 3.2.1.2 | Patikrinti ar sistema vadovaujasi „MS Windows“ standartu. | Aukšta |
| **TS2** |  | Patikrinti ar sistemos pranešimai veikia. | Vidutinė |
| **TS3** | 3.2.1.3 | Patikrinti ar sistema veikia Unity 2019.2 aplinkoje | Aukšta |
| **TS4** | 3.1.3.2.1.2 | Patikrint ar sugeneruoto namo plotas telpa į nurodytą intervalą | Aukšta |
| **TS5** | 3.1.3.2.1.3 | Patikrint ar sugeneruoto namo aukštų skaičius telpa į nurodytą intervalą | Aukšta |
| **TS6** | 3.1.3.2.1.4 | Patikrint ar sugeneruoto namo ugniasienių skaičius ir pusė atitinka pažymėtas interfeise | Aukšta |
| **TS7** | 3.1.3.2.2.1 | Patikrinti kiek laiko sistema veiks be trykių | Maža |
| **TS8** | 3.1.3.2.4.1.1 | Patikrinti ar vieno pastato generacijos trukmė atitinka reikalavimą | Maža |
| **TS9** | 3.1.3.3.1.1 | Patikrinti ar sistemos diegimo trukmė atitinka reikalavimą | Maža |
| **TS10** | 3.1.3.3.2.1 | Patikrinti keik laiko vartotojui trunka įsisavinti sistemos valdymą | Vidutinė |
| **TS11** | 3.1.3.4.1 | Patikrinti ar generacija vygdoma nenaudojant iš anksto sukurto bazinių modelių rinkinio | Kritinė |

### Generacijos rezultato testavimas ir demonstracija

**Testavimui naudojama techninė įranga:**

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

GPU – GeForce Gtx 1070

RAM – 16gb

**Pastatų generacijos greičio ir geometrijos testavimas**

Testo parametrai:

* 1000 pastatų
* Tarp 1 ir 5 aukštų
* Įjungtos kampinės kolonos
* Viena ugniasienė
* Kitos reikšmės atsitiktinės
* Testas atliekamas 5 kartus

Testo rezultatai:

Vidutinė pastato generavimo trukmė 22 ms. 1 aukšto pastatas generuojamas vidutiniškai 7.5ms, 5 aukštų 36ms.



4.1.5.1 pav. Iš sugeneruotų pastatų sustatyto miesto pavyzdys

Paveikslėlyje matomas demonstracijos tikslais sugeneruotas miestas parodantis galima sistemos generuojamų pastatų panaudojimą.



4.1.5.2 pav. Sugeneruoto namo pavyzdys ir namo segmentų demonstracija

Paveikslėlyje kairėje matomas 1 aukšto pastato generacijos pavyzdys. Dešinėje matoma iš kokių segementų sukomponuotas šis pastatas.

### Reikalavimo, kad sistema iš vartotojo nereikalautų bazinio modelių rinkinio įgyvendinimas

Šis reikalavimas įgyvendinamas visus galimus pastato segmentus (langus, duris, stogą, kaminą ir t.t.) aprašant sistemos kode.

Kiekvieno segmento generacija vygdoma keliais žingsniais:

1. Sukuriama bazinė figūra (kubas arba plokštuma)
   1. Sugeneruojami verteksai.
   2. Is verteksu sukuriami poligonai.
   3. Apskaičiuojama kokiu kampu kris šviesa ant kiekvieno poligono.
2. Pradinės figūros dimensijos ištampomos iki norimų.
3. Figūros verteksai išstumdomi taip, kad būtų sukuriama siekama forma (pvz. lango rėmas).
4. Kievkienam verteksui iš bendros tekstūros prisikiriama 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.

Verteksų generavimo funkcijos kodas:

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 virsus

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 skaidymo į poligonus (trikampius) funkcija:

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 figūrų kūrių yra tiek kiek pastatas turi aukštų
  + Kiekviena bazė turi langų rinkinį
  + Primas aukštas turi duris
* 1 Stogo
* 1 Palėpės
  + Palėpė gali turėti langų rinkinį
* 1 Kamino

Langai ir durys susideda iš tų pačių bazinių figūrų:

* Plokštumos
* Segementų (tik langai)
* Arkinio arba stačiakampio rėmo

Palėpės suformavimo iš bazinio kubo funkcijos kodas:

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);

}

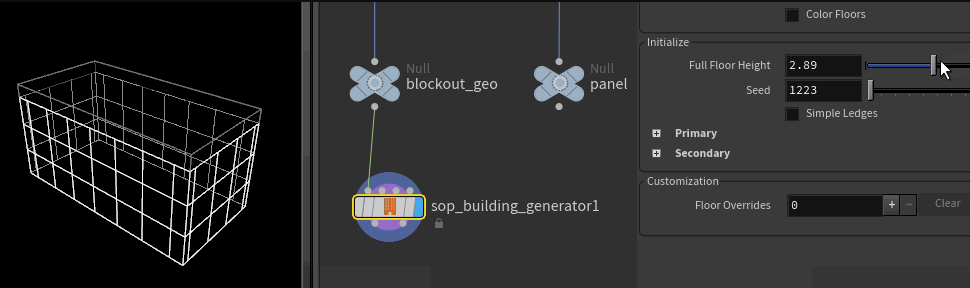
# Išvados ir siūlymai

# Literatūra ir šaltiniai

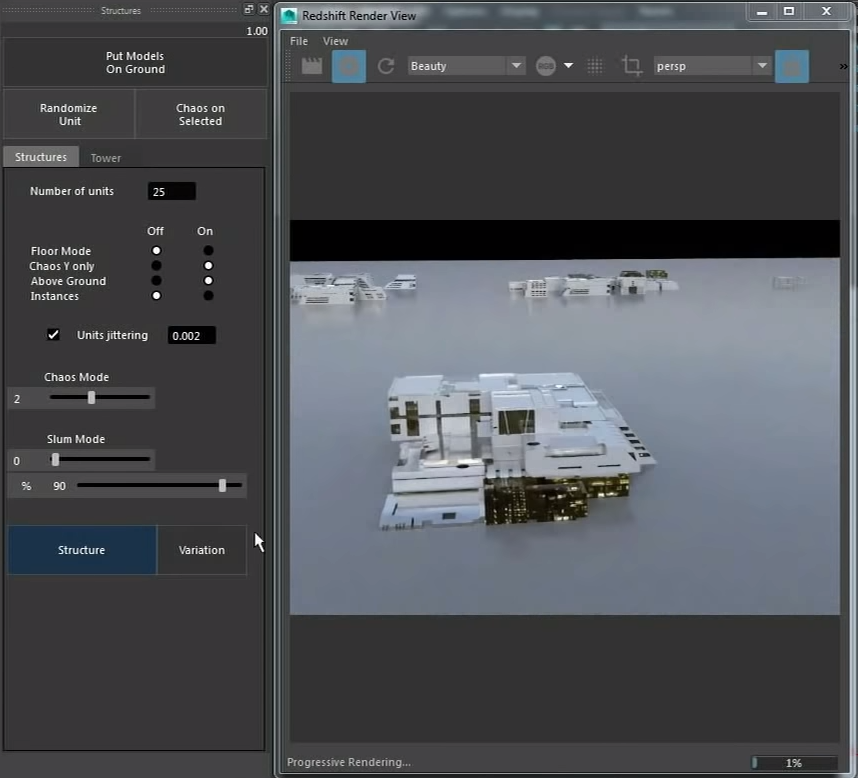
|  |  |
| --- | --- |
| [1] | „newzoo,“ [Tinkle]. Available: https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/. [Kreiptasi 27 12 2019]. |
| [2] | J. Couture, „Gamasutra,“ 01 07 2016. [Tinkle]. Available: https://www.gamasutra.com/view/news/273904/Why\_are\_so\_many\_devs\_employing\_a\_retro\_lowpoly\_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://tysonibele.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]. |
| [10] | P. M. Yoav I H Parish, „Procedural Modeling of Cities,“ [Tinkle]. Available: https://cgl.ethz.ch/Downloads/Publications/Papers/2001/p\_Par01.pdf. [Kreiptasi 21 11 2019]. |
| [11] | P. Bourke, „L-System User Notes,“ [Tinkle]. Available: http://paulbourke.net/fractals/lsys/. [Kreiptasi 21 11 2019]. |
| [12] | J. P. N. S. G. L. Stefan Greuter, „Real-time Procedural Generation of ‘Pseudo Infinite’ Cities,“ [Tinkle]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.7296&rep=rep1&type=pdf. [Kreiptasi 21 11 2019]. |
| [13] | M. W. F. X. S. W. R. Peter Wonka, „Instant Architecture,“ [Tinkle]. Available: https://hal.inria.fr/inria-00527500/file/instant\_architecture.pdf. [Kreiptasi 22 11 2019]. |

# PRIEDAI

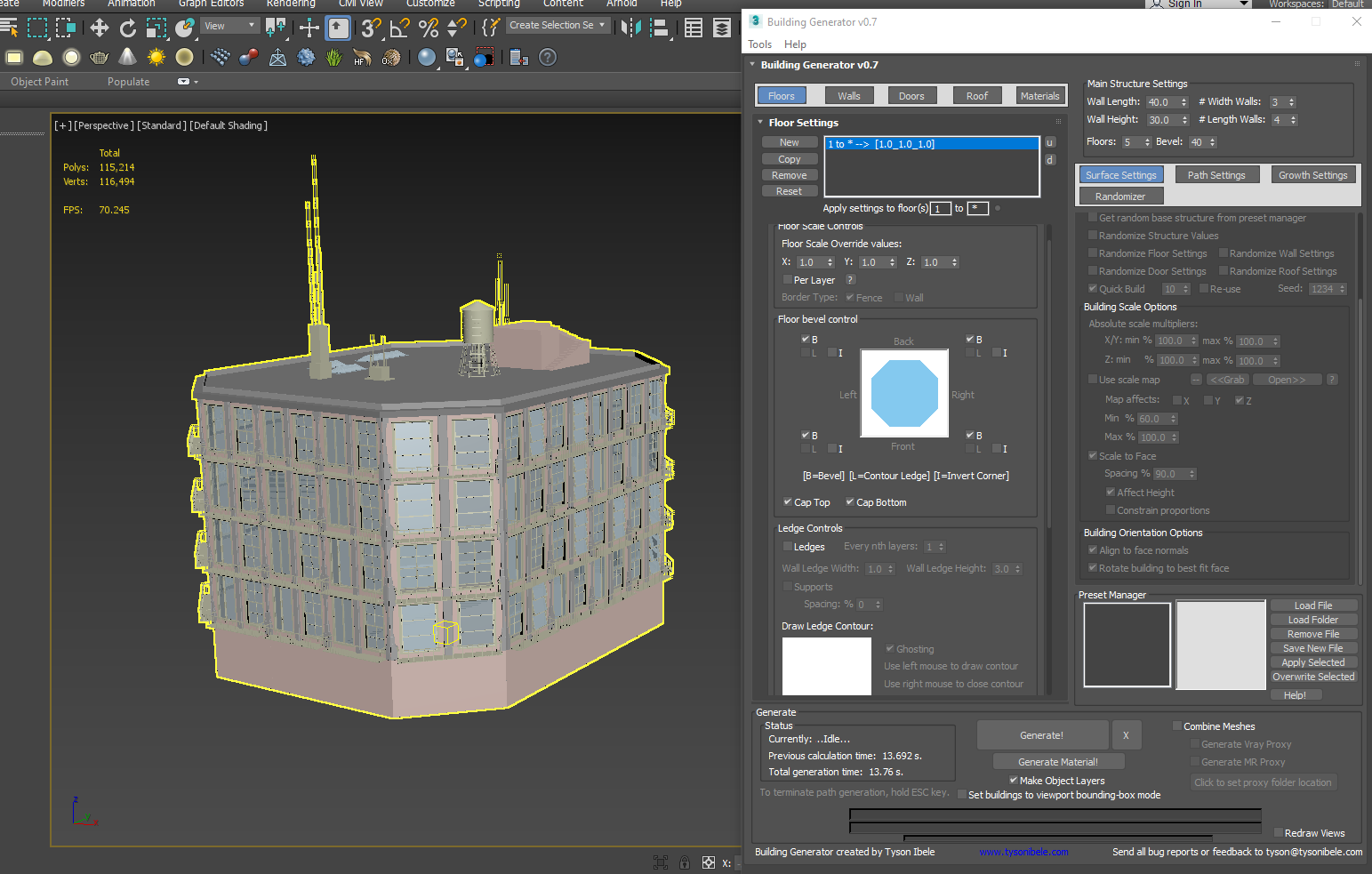
**1 PRIEDAS. Pastatų ir miestų generatorių priedai**



Priedas 1 Houdini building generator varotojo sąsajos pavyzdys [5]



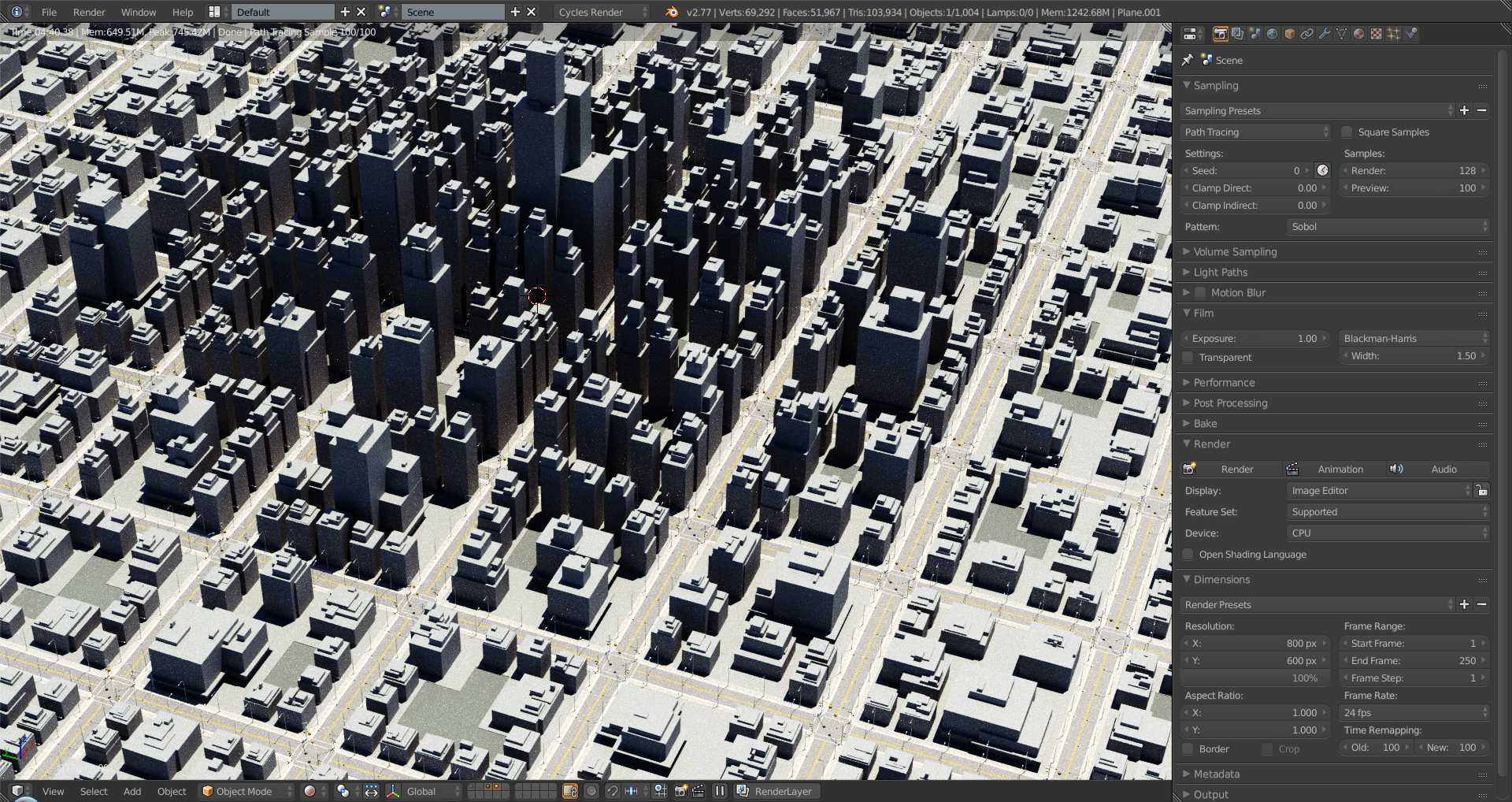
Priedas 2 Maya structures vartotojo sąsaja ir generacijos pavyzdys [6]



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

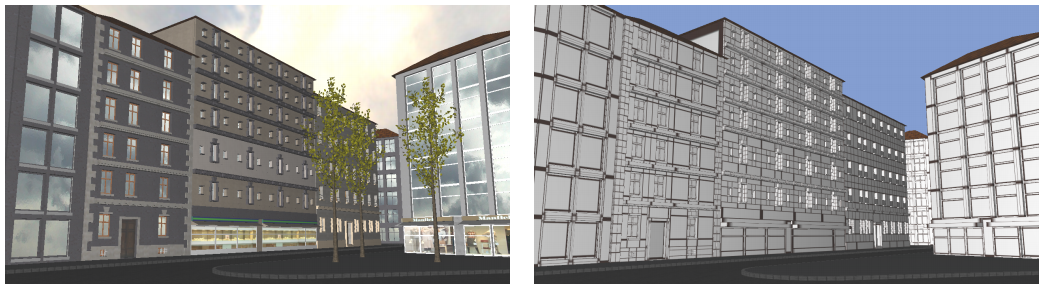


Priedas 4 SceneCity generacijos pavyzdys [9]



Priedas 5 SceneCity procedūriškai sugeneruoti pastatai [9]

**2 PRIEDAS. Generacijos algoritmų priedai**



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