KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Laura Stankevičiūtė

AUTOMATIZUOTŲ VIRTUALIŲ MIESTŲ GENERAVIMO METODŲ TYRIMAS

Magistro darbas

Vadovas
lekt. dr. A. Noreika

KAUNAS, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
Prof. dr. D. Rubliauskas
2012-05-28

AUTOMATIZUOTŲ VIRTUALIŲ MIESTŲ GENERAVIMO METODŲ TYRIMAS

Informatikos inžinerijos magistro baigiamasis darbas

Vadovas
lekt. dr. A. Noreika
2012-05-30

Recenzentas
doc. dr. G. Vilutis
2010-05-28

Atliko
IFN 0/1 gr. stud.
L.Stankevičiūtė
2012-05-28

KAUNAS, 2012

SUMMARY

Analysis of Virtual Cities Generation Methods

The main aim of the work is to determine the possible object generation methods as well as their application in developing the generators for cities and to select the most applicable methods for developing the city generator. These objectives were chosen in order to achieve the aim of the work: to make an analysis of the existing city generators and the mathematical methods used in them, to design the system of the city generator, to choose the most appropriate hardware and software, to establish the generation methods for a classical Roman colony, to make a realization, experimental tests and the conclusions of the work.

The field of the research is computer graphics, automatic and semi-automatic object generation (generation of graphic elements), procedural generation, representation and management.

The object of the research is the automatic generation methods for virtual cities.

The issue of the research is the potential limitations of the city generators in consideration of the historical period and style.

The first part of the work reviews the structure of the Roman civilization`s architecture, cities and the parameters of the city generators. Also, the basic generation methods and their application in developing the city generators were analyzed. The second part of the work represents the stages of the project realization, reviews the software, determines the methods applicable to the generation of cities and introduces to the user interface. Furthermore, the third part of the work emphasizes the program testing as well as the conclusions that were made.

Terminų ir santraupų žodynas

GIS – geografinė informacinė sistema.

OpenGL (angl. *Open Graphics Library*) – atvira grafikos biblioteka.

LRU (angl. *Least Recently Used*) - mažiausiai paskutiniu laiku naudoto puslapio keitimo algoritmas.

L-sistema – Lindenmayer sistema.

DV/reality – plačių galimybių programinė įranga, pritaikyta dirbi superkompiuteriuose.

IFS – literuotųjų funkcijų sistema

GPU (angl. *graphics processing unit*) – procesorius, naudojamas vaizdo plokštėse.

GNU GPL (angl. *GNU General Public License*) – GNU Bendroji Viešoji Licencija – yra laisvosios programinės įrangos licencija, pradžioje sukurta GNU projektui, šiuo metu tai viena populiariausių atvirojo kodo licencijų.

DFT – diskrečioji Furjė transformacija.

GFT – greitoji Furjė transformacija.

TURINYS

1. ĮVADAS

Animuotuose meniniuose ar fantastiniuose filmuose, žaidimuose, įvairių sričių vizualizacijose vis dažniau panaudojami virtualūs miestai**.** Jų kūrimo procesas apima reljefo, gatvių tinklo, augmenijos ir kitų panašaus pobūdžio objektų modeliavimą.

Aukšta kokybė bei realistinis vaizdas ir mastas paverčia procesą dar sudėtingesniu. Kaip ir kiekvieno produkto kūrime, laikas ir pinigai, kurie galėtų būti skiriami produkto kokybei ar naujų funkcijų įgyvendinimui, yra prarandami dėl aplinkos kūrimo. Galimas šios problemos sprendimo variantas – aplinką kurti automatinių procedūrinių metodų pagalba. Procedūrinių metodų technologijos kompiuterinėje grafikoje yra naudojamos tekstūrų kūrimui, specialiųjų efektų simuliacijai ir sudėtingų natūralių modelių (medžių, krioklių) generavimui.

Su kompiuterinės grafikos kūrimu susijusios kompanijos naudoja įvairią trimačio vaizdo generavimo programinę įrangą. Panaudojant jų programavimo galimybes, kuriamos paprogramės, kurių pagalba yra generuojami miestai, įvairūs elementai, kurie vėliau yra naudojami žaidimuose. Atsižvelgiant į darbo tematiką - trimatės grafikos kūrimo programos miesto generavimo paprogramė yra skirta automatiniam miesto generavimui pagal nustatomus parametrus. Priklausomai nuo poreikių, vartotojas gali valdyti įvairias generuojamo miesto charakteristikas.

Šiuo metu siūlomi įvairūs miestų generatoriai yra susitelkę į šio laikmečio miestų generavimą, todėl iškyla poreikis skirtingų laikmečių ir stilių miesto generatoriams.

Darbo tikslas - išsiaiškinti galimus objektų generavimo metodus bei jų taikymą kuriant miestų generatorius. Parinkti tinkamiausius metodus kuriamam miesto generatoriui.

Tyrimo sritis – kompiuterinė grafika, automatinis ir pusiau automatinis objektų generavimas (grafikos elementų generavimas), procedūrinis generavimas, atvaizdavimas ir valdymas.

Tyrimo objektas – automatiniai virtualių miestų generavimo metodai.

Tyrimo problema – miestų generatorių trūkumas, atsižvelgiant į istorinį laikotarpį ir stilių.

Žemiau pateikiami suformuluoti darbo uždaviniai:

1. Atlikti esamų miestų generatorių ir juose naudojamų matematinių metodų analizę;
2. Atlikti miestų generatoriaus sistemos projektavimą, parinkti tinkamą techninę ir programinę įrangą;
3. Sudaryti klasikinio Romos imperijos kolonijos miesto generavimo metodus;
4. Atlikti realizaciją ir eksperimentinius bandymus, suformuluoti išvadas.

Darbo struktūra:

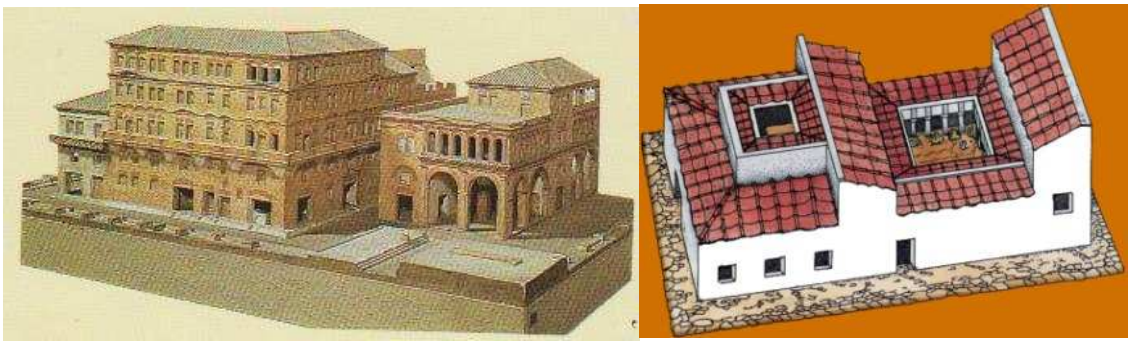
- Antrajame skyriuje atlikta Romėnų civilizacijos architektūros ir miestų struktūros, miestų generatorių parametrų apžvalga;
- Trečiajame skyriuje atlikta bazinių generavimo metodų ir jų taikymo miestų generatorių kūrimui analizė;
- Ketvirtajame skyriuje aprašyti projekto realizavimo etapai, pateikta programinės įrangos apžvalga, pateikti miestų generavimui taikomi metodai ir supažindinama su vartotojo sąsaja;
- Penktajame skyriuje aprašytas programos testavimas;
- Šeštajame skyriuje pateikiamos išvados

2. PROBLEMOS ANALIZĖ

2.1.Romėnų civilizacijos architektūros ir miestų struktūros analizė

Romėnai daugelį dalykų perėmė iš graikų. Nors jų pastatai neprilygo graikų pastatų lengvumui ir kolonų grakštumui, buvo iškilmingi, griežtai simetriški, ekonomiškai ir apgalvotai panaudotas kiekvienas akmens gabalas. Viskas buvo išpuošta marmuru, mozaikomis, daug kas paauksuota. Miestą puošė daugybė fontanų, kolonų, statulų, turtuolių namų, pirčių kompleksų.

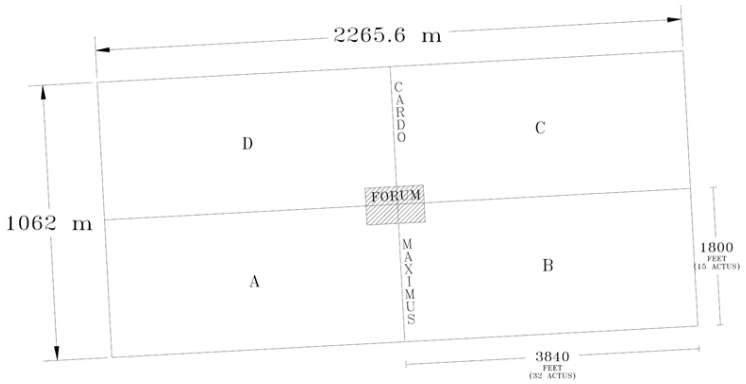
Mieste dominavo du pagrindiniai namų tipai: *domus,* kuriuos sudarė grupė kambarių, ribojančių kiemą, ir *insulės* – daugiabučiai namai (žr. 1 pav.). IV a. 28% namų buvo domus tipo ir 72% – insulės, išsisklaidžiusios beveik po visą miestą. Pasiturinčios šeimos gyveno greta vargšų, įsikūrusių ankštuose butuose. Kilmingieji turėjo ir tvoromis aptvertas prabangias vilas. [1]



Insulės Domus

1 pav. Romėnų pastatai. [1]

Klasikinio romėnų miesto gatvių tinklas buvo taisyklingas stačiakampis, su dviem įstrižai besikertančiomis gatvėmis centre. Pagrindinės gatvės buvo vadinamos d e c u m a n u s m a x i m u s ir c a r d o m a x i m u s, kurių sankirtoje buvo pagrindinė aikštė – forumas (lot. *forum*) (žr. 2 pav.). Aplink pagrindinę aikštėję išsidėstę pastatai atlikdavo politinį, religinį bei socialinį miesto vaidmenį. [2]



2 pav. Pagrindinės miesto gatvės [2]

Pagrindiniai aplink pagrindinę aikštę išsidėstę pastatai – comitium, bazilika, šventovė, macellum. Pagrindinių pastatų kiekis bei jų atliekamos funkcijos buvo skirtingos. Tai priklausydavo nuo miesto gyventojų skaičiaus, religijos, kultūros ir t.t.

Miesto gatvių tinklo sistemą pirmieji sukūrė graikai. Laikui bėgant šią gatvių tinklo sistemą tobulino kitos kultūros, įkurdamos miestus. Viena geriausiai žinomų gatvių tinklo suskirstymo sistemų – romėnų tinklo sistema (žr. 3 pav.). Šią sistemą romėnai naudojo įkurdami Romos imperijos kolonijas.

3 pav. Romėnų kolonija – Florencija [3]

Nepaisant to, kad romėnų gatvių sistema buvo panaši į graikų, romėnų gatvių tinklas buvo praktiškiau pritaikomas įkuriant miestą. Romėnai miestus statė mažiau kalvotose vietovėse bei kuo arčiau vandens telkinių ar prekybos susikirtimo taškų. Gatvių tinklo kraštinės dažnai būdavo vienodos (655 m.) ir retai kisdavo. [2], [1], [3]

Romėnų miestų planavimas išsiskyrė griežtomis ir sistemizuotomis charakteristikomis, todėl jų miesto generavimui reikalingas tiksliai aprašomas generatorius.

2.2.Miestų generatorių analizė

Šiame skyriuje yra atliekama miestų generatorių parametrų ir kūrimo metodų apžvalga. Miestų generatorių parametrai bus vertinami atsižvelgiant į tokius kriterijus: autonomiškumas, licenzijos tipas, failo dydis, išskirtinės savybės, galimi pasirenkami parametrai.

2.2.1. Parametrų analizė

1 lent. Miestų generatoriaus „KludgeCity“ parametrų analizė

KludgeCity

Programa, kurioje veikia	Maya, AutoDesk
Licenzijos tipas	Atviras kodas

Failo dydis	708.4 KB
Komentarai	Procedūrinė sistema pastatų generavimui. Sukuriamas geometrijos fragmentas, kuris bus pastato pagrindas. Sukurtas pastato (ar kelių, keliolikos) pagrindas įkeliamas į sistemą, pakeičiami kintamieji ir paspaudžiamas vykdymo mygtukas. Labiau pritaikyta ne miestų, o atskirų sudėtingų pastatų generavimui. [4]

2 lent. Miestų generatoriaus „CityEngine“ parametrų analizė

CityEngine 2010

Programa, kurioje veikia	Autonominė
Licenzijos tipas	Shareware
Kaina	299.00$
Failo dydis	496 KB
Komentarai	Autonominė programinė įranga, vartotojus aprūpinanti pramogine, architektūrine, miestų planavimo GIS ir bendrojo 3D turinio produkcija su unikaliais konceptualaus projektavimo ir modeliavimo sprendimais efektyviam 3D pastatų ir miestų kūrimui. Labiau rankinis, nei automatinis generavimas. [5]

3 lent. Miestų generatoriaus „ghostTown“ parametrų analizė

ghostTown

Programa, kurioje veikia	3ds Max 2011, AutoDesk
Operacinė sistema	Windows
Komentarai	Miestų generatorius ir įrankis miesto aplinkos kūrimui. Aukštos ir žemos rezoliucijos pastatai (angl. *hi– poly, lo– poly*), pagrindinių kelių kūrimas, lengva medžiagų sistema.[6]

Parametrai

Pagrindiniai įrankiai:	Setup, Build, reBuild
Pastatų parametrai	• Namų aukštis • Namų aukščio svyravimas

	• Aukštų aukštis • Biurams skirtų pastatų aukštis • Biurams skirtų pastatų aukščio svyravimas • Kelių plotis • Kelių plotis prie biurams skirtų pastatų • Parko kelių plotis • Namų skaičius kvartale • Namų aukščio svyravimas kvartale • Šaligatvio aukštis • Šaligatvio plotis • Biurų procentinis skaičius mieste

Tekstūravimas	• Paletės naudojimas
	• Paveikslėlių paletės naudojimas
	• Tekstūravimas:
	  ○ Pirmas aukštas
	  ○ Detalės
	  ○ Didingi pastatai
	  ○ Durys
	  ○ Fasadai
	  ○ Žolė
	  ○ Išoriniai fasadai
	  ○ Aikštės
	  ○ Keliai
	  ○ Stogai
	  ○ Medžiai
	  ○ Sienos

Bendri parametrai	• Kampuoti stogai • Gatvių smulkmenos • Stogų smulkmenos • Parkai • Aukšta/žema rezoliucija

Komentarai	Trimačio vaizdo lange, pele parenkamas plotas miesto generavimui. Pasirinkimas atliekamas įrankių juostos „Show Guide“ meniu pagalba. Suteikiama galimybė sukurti objektą, priskirti jam savybes ir išsaugoti programoje. Generuojant iš naujo, objektas yra įtraukiamas prie numatytųjų nustatymų. [6]

4 lent. Miestų generatoriaus „Suicidator 3D City Generator“ parametrų analizė

Suicidator 3D City Generator

Programa, kurioje veikia	Blender
Licenzijos tipas	Atviras kodas
Operacinė sistema	Windows, Linux, Mac
Failo dydis	263 kb
Komentarai	Suteikiama galimybė automatiškai sukurti trimatį šiuolaikišką miestą, pritaikant įvairius parametrus, pavyzdžiui: miesto dydį ir sudėtingumą. Pagrindinė generatoriaus idėja – atsitiktinumas, todėl kiekvieną kartą kuriant naują miestą, kiekvienas pastatas ir gatvė yra atsitiktiniai ir unikalūs. Generatorius yra greitas, sukuria sudėtingai atrodančius miestus, sunaudodamas mažai atminties, todėl jį galima naudoti ir turint negalingą kompiuterį. Pasitelkiant sudėtingesnes konfigūracijas, atsiranda galimybė sukurti miestus, užpildančius horizontą. Numatyta galimybė kurti dieninį arba naktinį miestą. [7]
Parametrai	
Tekstūrų generatorius	• Sienų dydis ir spalvos • Langų dydis ir spalvos • Galimybė turėti didesnes sienas • Paros metas
Pastatų generatorius.	Sukuriami ir išdėstomi pastatai. Naudojamasi populiacijos žemėlapiu – juodai baltu paveiksliuku, kuriame balta spalva parodo tankiai apgyvendintas zonas, o juoda – mažai. Tai turi įtakos pastatų dydžiui.
Gatvių	• Aukščių žemėlapis. Juodai baltas paveiksliukas, kuriame balta spalva yra

generatorius	didelio aukščio zonose, o juoda – mažo. Jo pagalba miestas yra sukuriamas ne ant plokščio paviršiaus. • Tankumas • Ilgis • Linkis • Pradiniai kampai • Didžiausias nuolydis
Komentarai	Generavimas yra grįstas šiais trimis pagrindiniais parametrais. Visi jie veikia kartu. Kiekvienas turi atskirą parinkčių rinkinį, sudėtą į įrankių juostą. Kiekvienas generatorius valdomas per jam priskirtus parametrus.

5 lent. Miestų generatoriaus „Blended Cities“ parametrų analizė

Blended Cities

Programa, kurioje veikia	Blender
Licenzijos tipas	Atviras kodas
Operacinė sistema	Windows, Linux, Mac
Komentarai	Galimybė greitai sukurti didelį kiekį įvairių formų gatvių ir pastatų. Priešinamasi nusistovėjusiai kampuotų gatvių ir stačiakampių gretasienių pastatų tvarkai: vingiuotos gatvės ir netaisyklingų formų arba cilindriniai pastatai. Yra galimybė kurti ir senamiesčius, ne vien modernius miestus.
Bruožai	• Vyraujanti įvairovė, chaosas, išskyrus tuos atvejus, kai parametrais pasirenkama kitaip. Gatvių ir sankryžų nereguliarumas, nekvadratiškumas, kuriamų pastatų unikalumas. Procedūrinių funkcijų pagalba, pastatai yra kuriami nuo nulio ir/arba primityvių objektų, įkeltų iš bibliotekos. Pagal koncepciją, generatorius turi galimybę naudoti iš anksto sukurtus objektus, sukurtus 3D bendruomenės. • Moduliškumas: galimybė sukurti pastatus su „Python“, pradedant nuo turimų pastatų nustatytų funkcijų ir/arba įkeliant jau sukurtus objektus. Pastatų ir gatvių objektų bibliotekos pergrupuoja procedūrinių pastatų kodus ir vartotojo objektus, patalpintus papildomame „Blender“ faile. Yra atskira paprogramių sistema, leidžianti praplėsti generatoriaus funkcijas.

	• Prieinamumas: vartotojo sąsaja yra daugiakalbė, patogi.
Parametrai	
Pagrindinis meniu	• Atsitiktinumas • Kalba • Miesto nustatymai (klaidos pranešimai, sufleravimas virš mygtukų)
Įvestys	• Miesto žemėlapis: o dydžio nustatymas, o dislokacijos vieta. • Miesto išsidėstymas aukščio atžvilgiu (juodai baltas paveiksliukas) • Zonavimas (spalvotas paveiksliukas, kiekviena spalva turi reikšmę, pvz.: pastatų tipą, tankumą ir pan.)
Gatvės	• Savybės: o kokybės parametrai (nustatomas detalumo lygis, pagrindinės šaligatvių ir sankryžų savybės), o gatvių sluoksniai, o objektų sluoksniai. • Keliai: o kelių kategorijos, o miesto žemėlapio redagavimas, o šaligatvių kategorijos, o šviesoforų kategorijos. • Linijos: o automobilių, o autobusų, o dviračių, o gatvių plotis. • Šaligatviai: o aukštis, o plotis, o medžiai.

Pastatai	• Savybės: o objekto pavadinimas (nustato objekto kategoriją), o aukštų skaičius – ▪ minimalus, ▪ maksimalus, ▪ atsitiktinė variacija. o aukšto aukštis – ▪ minimalus, ▪ maksimalus, ▪ pirmo aukšto aukštis (jis paprastai būna aukštesnis už visus kitus).
	• Kategorijos. Šis meniu leidžia nustatyti, kuris pastatas kurio kito pastato dalyje bus: o galimos kategorijos: pastato biblioteka gali priklausyti vienai arba kelioms pastatų kategorijoms. Čia yra pasirenkama kategorija miesto kūrimui – ▪ gyvenamoji, ▪ industrinė, ▪ namų. o apibrėžiama pasirinkta kategorija, o galimos bibliotekos. • Bibliotekos
Miesto kūrimas	• Pastatus • Gatves • Miestą [8]

2.2.2. Metodų analizė

Šiame skyriuje bus apžvengti ir įvertinti procedūrinių miestų generatorių kūrimo metodai. Miestų generatorius yra sukuriamas pereinant keletą stadijų, iš kurių kiekviena reikalauja skirtingų metodų: kelių, pastatų struktūrų ir tekstūrų, aplinkos kūrimo.

6 lent. Miestų generatoriaus „Undiscovered City“ metodų analizė

Undiscovered City

Komentarai	Generuojamas kelių tinklas ant paprasto tinklelio, iš kurio, naudojant kombinuotus geometrinius primityvus, yra generuojami pastatai.

	Programa skirta realaus laiko generavimui.
Keliai	Miesto keliai kuriami pagal reljefo šabloną, remiantis modernių Amerikos miestų modeliais. Reljefas yra taisyklingos formos, kiekvieno bloko dydis yra pastovus, bet gali būti koreguojamas globaliu mastu.
Pastatai	Realaus laiko programa optimizuoja veikimo laiką ir erdvinio vaizdo peržiūrą. Įkeliant ir atvaizduojant pastatų rinkinius yra sunaudojama daug atminties ir vietos, todėl šiame miesto generatoriuje yra naudojamas vaizdo peržiūros metodas, kuris atvaizduoja tik tuos pastatus, kurie yra matomi iš žiūrėjimo kampo, o ne pilną miesto planą. Pastatai yra generuojami ir aprašomi, kaip OpenGL atvaizdų sąrašai, kurie gali būti talpinami laikinojoje atmintyje. Laikinoji atmintis dirba LRU algoritmu, kai dažniau naudojami pastatų modeliai yra laikomi laikinojoje atmintyje, o senesni, mažiau naudojami, yra pakeičiami. Laikinosios atminties naudojimas pastatų modelių atkūrimui yra rezultatyviau, nei generuojant pastatų modelius iš naujo. [9], [11]

7 lent. Miestų generatoriaus „CityEngine“ metodų analizė

CityEngine

Komentarai	Pagrindinis miesto generatoriaus generavimo metodas yra L– sistemos. Šios sistemos tradiciškai yra naudojamos natūraliems gamtos reiškiniams ar objektams atkurti, bet taip pat yra tinkamos ir miestų generavimui dėl savo konstruktyvios prigimties, efektyvaus skaičiavimo ir duomenų aplikacijos savybių.
Keliai	Kelių tinklo generavimo pagrindinis metodas yra L– sistemos. Yra naudojamos praplėstos L– sistemų formos, vadinamos „Self– sensitive“ L– sistemos, skirtos generuoti kelių tinklus auginimo principu. Pavyzdys yra imamas iš dvimačių žemėlapių. Iš jų yra imama tokio pobūdžio informacija, kaip: geografinė informacija apie aukštį, augalijos ir vandens ribos, socialinis statistikos vaizdų žemėlapis, įtraukiant informaciją apie gyventojų tankumą, žemės panaudojimą, gatvių struktūrą, pastatų aukščius. Yra naudojami du pagrindiniai taisyklių rinkiniai: globalūs tikslai ir vietiniai apribojimai. Kelių segmentai iš pradžių yra generuojami pagal pasaulinių tikslų modelį, kuris yra panašus į tokį, kokį miesto dizaineris turėjo omenyje. Vėliau šie planai yra pritaikomi prie vietinių apribojimų, kurie atspindi realią kelių tinklo būklę.

	• Globalūs tikslai:
	  o yra du skirtingi kelių tipai: greitkeliai arba pagrindiniai keliai, sujungiantys labiausiai apgyvendintus miesto taškus ir maži keliai, sujungti su artimiausiu greitkeliu;
	  o gatvės laikosi nustatytų geometrinių kreivių;
	  o gatvės kuriamos ant žemiausių, reljefe esančių, kreivių.
	• Vietiniai apribojimai:
	  o kelių segmentai yra apkarpomi tam, kad tilptų į numatytą erdvę: linijų segmentai, nusitęsę į vandenį yra nutrinami;
	  o keliai yra pasukami tam, kad tilptų į numatytą erdvę: keliai palei pakrantę yra sulenkiami;
	  o greitkeliams yra leidžiama kirsti tam tikrus plotus, kurie yra negalimi kitiems keliams: per vandenį greitkelis eina tiesiai, kaip tiltas;
	  o kelių segmentai yra patikrinami tam, kad būtų įsitikinta ar jie susikerta su kitais keliais reikiamu atstumu, pagal numatytas funkcijas.

Pastatai	Sugeneruotoje kelių tinklo sistemoje, pastatai konstruojami keliais etapais: nustatomas pastatų pasiskirstymas, sukuriama pastato geometrija ir tekstūra. Pastatų pasiskirstymo nustatymui yra naudojami duomenys iš ankstesnių, kelių tinklo generavimo etapų (žr. 4 pav.).  **4 pav.** Kelių tinklo generavimo etapai Pastatų grupių išsidėstymas yra apskaičiuojamas iš pirmojo kelių generavimo etapo. Sugeneruoti blokai yra naudojami kaip pastatų grupės rėmai. Kiekvienas blokas yra atsitiktinai sudalintas į potencialias pastatų grupes. Pastatų geometrija yra generuojama panaudojant L– sistemas. Pastatų tipai yra skirtingi: dangoraižiai, komerciniai ir gyvenamieji pastatai – visi jie naudoja skirtingas L– sistemų formas. L– sistemų operacijas sudaro transformacijos, geometrinių šablonų naudojimas – stogai, antenos, ir t.t. L– sistemos suteikia galimybę

	platesniam modelių pasirinkimui bei sudėtingiems sprendimams. Esant keletui tinklinių struktūrų, yra naudojamos funkcijos, kurios aprašo struktūrų sąveiką. Funkcijos nurodo, kuri ląstelė ir kurios vietos tarpininkaus, sudarant tekstūrą. Ląstelės paprastai saugo langų, durų ir kitų pastato elementų tekstūras. Galutiniam atvaizdavimui yra naudojami duomenys, kurie gali būti importuojami į „Maya“ komercinį 3D paketą. Yra galimybė išgauti vaizdą realiu laiku – naudojant Dimension DV/reality programinę įrangą. [10]

8 lent. Miestų generatoriaus „CityBuilder“ metodų analizė

CityBuilder

Komentarai	Sistema yra sukurta remiantis „NetLogoTM“ platforma, kuri yra multi–agentinė programuojama modeliavimo aplinka, grįsta „Logo“ programine kalba, skirta vartotojų supažindinimui su atsiradusiais reiškiniais. Šis miesto generatorius yra realizuojamas, naudojant veiksnių rinkinius, kurių pagalba yra sumodeliuojami specifiniai miesto subjektai. Sistemos modelis ne tik sukuria kelių tinklą ir pastatus, bet taip pat simuliuoja miesto augimą ir plėtrą.
Keliai	Kelių tinklas yra kuriamas iš kelio segmentų, kurie yra išdėstyti pagal miesto pagrindo kreives. Yra galimas nuokrypis nuo kreivės, pasirenkant specifinį parametrą. Kelių susijungimai gali būti keičiami konstantų redagavimo pagalba, kurios nustato kelio galutinį taško ir atstumą tarp susikirtimų. Pagal įvestą aukščių žemėlapį, su nurodytu vandens lygiu, yra sudaromos galimos vietovėms keliams ir pastatams. Kelių segmentai yra kuriami pagal du veiksnių rinkinių tipus – išplėstuvus (*extenders*) ir prijungėjus (*connectors*): • Išplėstuvai – reljefe ieško laisvų plotų, kurie dar nėra užstatyti kelių tinklu. Atradus laisvą plotą, yra vertinamas kelių tankis, artimiausios sankryžos ir nuokrypis nuo pradinio taško. Laisvame plote atsiranda nauji keliai, jei jie nepadaro per didelių pokyčių bendram kelių tinklo aukščiui. • Prijungėjai – tikrina atstumus tarp taškų keliautojo atžvilgiu. Jei atstumas tarp dviejų galimų taškų yra per ilgas, arba jie nesusiekia, bus pasiūlomas naujas kelio segmentas.

Pastatai	Kiekvienas būsimojo miesto ploto sklypas gali būti užimtas keliu ar pastatu, arba būti nenaudojamu (dėl netinkamo aukščio, pasvirimo kampo, vandens ir pan.). Pagal veiksnių rinkinius plotai yra sugrupuojami į sklypus – kurie bus skirti keliams, kurie pastatams. Yra trys skirtingi pastatų tipai: gyvenamieji, komerciniai ir pramoniniai. Komercinius pastatus stengiamasi statyti kuo arčiau miesto centro ir kuo arčiau gatvių susikirtimo taškų. Gyvenamųjų namų zonos yra projektuojamos vidurinėje meišto dalyje, o pramoninių – pakraštyje. Yra šioks toks atsitiktinis išsiskaidymas. Sistema kuria kelių tinklą ir apibrėžia žemės plotus pastatams pagal jų tipus, tačiau negeneruoja nei pačių pastatų, nei tekstūrų. [11]

3. MIESTŲ GENERATORIŲ KŪRIMO METODŲ ANALIZĖ

3.1.Baziniai generavimo metodai

Procedūriniai metodai gali būti įvairūs. Vienas iš pagrindinių metodų, kuris gali būti naudojamas trimačių primityvų generavimui su atsitiktiniais parametrais, yra atsitiktinio aukščio stačiakampis gretasienis. Paprasti algoritmai, naudodami pseudo atsitiktines funkcijas, gali būti naudojami tekstūros triukšmui ir natūralioms formacijoms generuoti. Sudėtingesni rekursiniai algoritmai, tokie kaip fraktalai ar Lindenmayer sistema gali būti naudojami organinių struktūrų, randamų gamtoje, tokių kaip sniegas ir medžiai, atkūrimui. Svarbūs procedūrinių metodų bruožai:

Abstrakcija: geometriniai ir tekstūriniai duomenys nėra nurodomi įprasta prasme. Vietoje to, detalės yra reziumuotos į algoritmus ar procedūrų rinkinius. Svarbus kuo mažesnis detalių skaičius, kad būtų galima lengviau manipuliuoti informacija.

Parametrų kontrolė: parametrai yra nustatomi ir koreguojami taip, kad tiesiogiai reguotų į specifinius nustatymus. Kūrėjas gali nustatyti tiek reikiamų valdymo parametrų, kiek reikia varotojo efektyviam naudojimui. Pavyzdžiui: į parametrus įtraukti vietovės kalnų aukštį, ar segmentų skaičių.

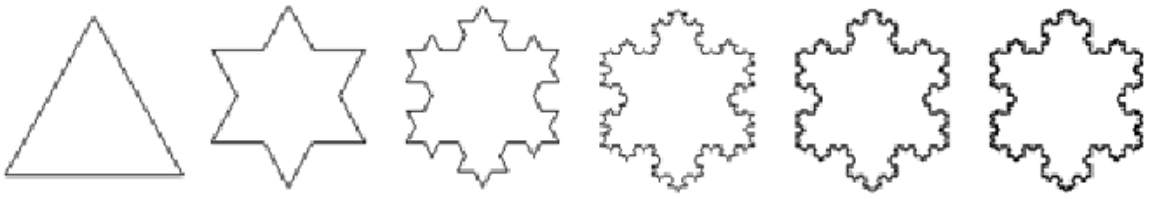
Lankstumas: įmanoma sukurti ne tik realaus pasaulio modelį, bet ir peržengti jo ribas. Parametrai gali būti įvairiapusiai, padedantys sukurti platų diapazoną įvairių rezultatų, kurie nebūtinai turi atitikti realų modelį.

Tekstūros, geometrija ar efektai, sudėti į procedūrinius algoritmus, nėra skirti rezoliucijos, ar daugiakampių skaičiui nustatyti. Procedūriniai metodai iš principo yra daugiarezoliuciniai, gali skirtis tik jų sudėtingumas. Ši savybė yra itin svarbi kompiuterinės grafikos srityje, pvz.: detalių skaičius yra svarbus bet kokiai trimatės grafikos atvaizdavimo sistemai, o ypač realaus laiko atvaizdavimo programoms. Pagrindinė mažo detalių skaičiaus idėja yra naudoti daugiau paprastų objektų versijų, jei tai prisideda prie bendro galutinio detalių skaičiaus mažinimo. [12], [13]

3.2.Pagrindiniai procedūriniai metodai ir algoritmai

3.2.1. Fraktalai

„Fraktalas yra sudėtinis geometrinis darinys, kurio atskiri fragmentai yra panašūs arba identiški visumai arba kitiems fragmentams. Pagrindinė fraktalų bendra savybė yra panašumas į save, t. y. išdidinta maža geometrinės struktūros dalis atrodo identiška didesnei daliai, kaip trikampiai Koch snaigėje (žr. 5 pav.). Fraktalai turi begalinį skaičių detalių, todėl, žiūrint į bet kurį duotą fraktalą, iš kuo arčiau bus žiūrima, tuo daugiau detalių bus matoma.“



5 pav. Koch snaigės 6 pirmieji pasikartojimai [14]

„Fraktalams generuoti yra skirti rekursiniai algoritmai. Nuosekli rekursija suteikia detalesnes pagrindinių formų versijas. Nėra teorinio limito rekursijų skaičiui sukurti, taiga begalinis detalių skaičius gali egzistuoti kiekvienoje formoje.

Fraktalai gali būti perteikti pasitelkiant IFS. Turint omenyje, kad fraktalas susideda iš jo paties kopijų, kiekviena IFS transformacija apibūdina ir charakterizuoja tarpusavyje sujungtų transformacijų kiekvieną kopiją.“ Tarpusavyje sujungtos transformacijos yra funkcija:

$$f(x,y) = (a * x + b * y + e, c * x + d * y + f); \quad (1)$$

čia $x, y \in X$, parametrai: a, b, c, d, e, f.

Tai taip pat gali būti užrašyta:

$$f(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \binom{x}{b} \binom{e}{f}. \quad (2)$$

[27], [15]

3.2.2. L– sistemos

L– sistemos buvo sukurtos septintame praėjusio amžiaus dešimtmetyje. Vengrų biologas ir botanikas teoretikas Aristid Lindenmayer siekė sukurti sistemą, tinkamą aprašyti augalų augimui. Dėl sistemos paprastumo ir tinkamumo ji prigijo ir matematikoje. [17]

Norint nubraižyti brėžinį remiantis šia sistema, yra nustatomi tokie parametrai:

Kintamieji:

- F – judėti žingsnį pirmyn; žingsnio ilgis – d. Kursoriaus padėtis keičiasi (x′, y′, α′), kai $x' = x + d\cos\alpha$ (10) ir $y' = y + \sin\alpha$. (11) Yra nupiešiamas linijos fragmentas tarp (x, y) ir (x', y') taškų.
- f – judėti žingsnį į priekį nepiešiant linijos; žingsnio ilgis – d.

Konstantos:

- + pasukti į kairę; pasukimo kampas – δ. Kita kursoriaus padėtis yra (x, y, $\alpha + \delta$).
- – pasukti į dešinę; pasukimo kampas – δ. Kita kursoriaus padėtis yra (x, y, $\alpha - \delta$).

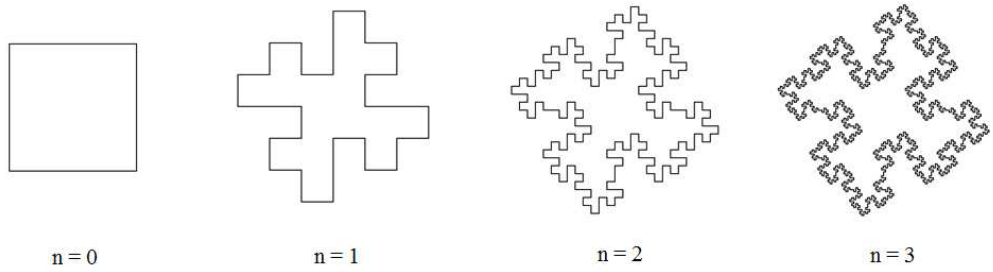
Pradžios sąlyga: F

Taisyklė: $F \rightarrow F - F + +F - F$

8-ame paveikslėlyje parodytos keturios kvadratinės Kocho salos aproksimacijos. Šios iliustracijos buvo gautos interpretuojant sugeneruotas L– sistemų eilutes:

$\omega: F - F - F - F$

$P: F - F - F + F + FF - F - F + F$



8 pav. Generuojama kvadratinė Kocho sala [17]

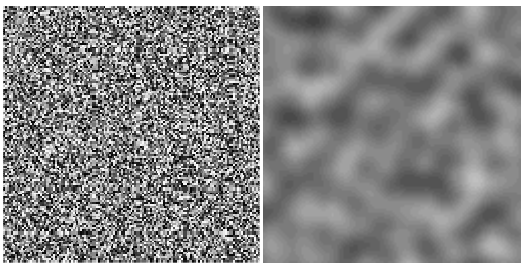
Pateiktas pavyzdys parodo glaudų ryšį tarp Kocho konstrukcijų ir L– sistemų. Modifikuotojo Kocho kreivėje vietoje trikampio atsiranda keturkampis. L– sistemos taisyklė yra pakeičiama į: $F \to F - F + F + F - F$, o sukimo kampas pakeičiamas į statų.

Atsitiktinę Kocho kreivę galima taikyti ir modifikuotai, ir nemodifikuotai Kocho kreivei. Modifikuotos kreivės taisyklė: $F \to FxFyyFxF$; čia x, y – kampai. Nemodifikuotos kreivės taisyklė: $F \to FxFyFyFxF$. Kampai x ir y yra parenkami taip, kad $x = -y$.

Ši atmaina yra naudojama kuriant žaidimus su dvimatėje erdvėje esančiais realistiškais žemėlapiais. [17]

3.2.3. Perlino triukšmas

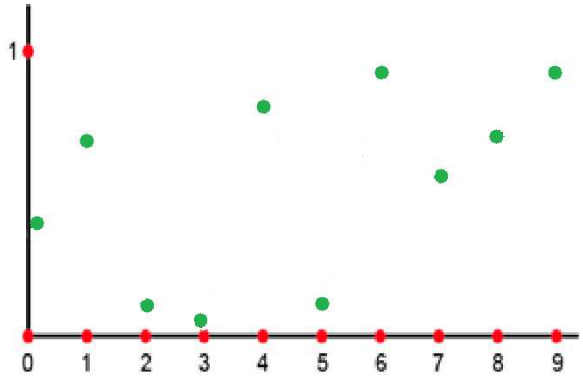
Perlino triukšmas yra funkcija, skirta nuoseklaus triukšmo erdvėje generavimui. Nuoseklus triukšmas reiškia, kad funkcijos tolygiai keičiasi judėdamos nuo vieno taško iki kito, nėra jokių trūkių (žr. 9pav.).



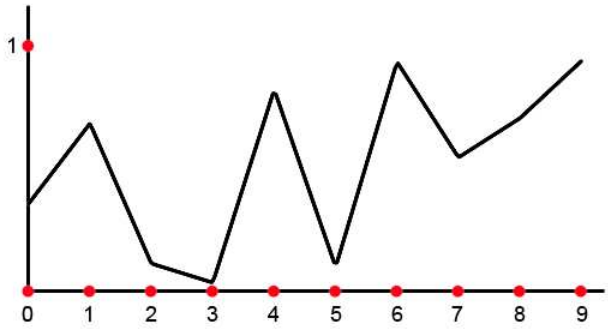
9 pav. Nekoherentinis ir Perlino triukšmai [18]

Interpoliacija – tai procesas, kai kreivė eina tiksliai per duomenų taškus. Ši funkcija gali generuoti naujus duomenų taškus, atsižvelgiant į žinomas taškų reikšmes, šiuo atveju, įėjimo taškai yra reikšmės, gautos iš triukšmo funkcijos. Esant baigtinių taškų aibei, gaunama begalinė taškų aibė. Šiai interpoliacijai atlikti yra galimi keli skirtingi algoritmai. Šie

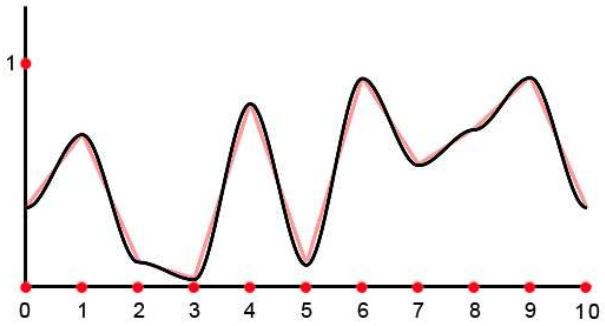
algoritmai skiriasi įėjimo taškų skaičiumi, tikslumu ir t.t. Žemiau pavaizduoti grafikai demonstruoja du skirtingus algoritmus; imant reikšmes iš 10-o paveikslėlio, 11-ame paveikslėlyje yra pavaizduota linijinė interpoliacija, 12-ame paveikslėlyje sudėtingesnė – kubinė. [18], [19]



10 pav. Atsitiktinio triukšmo generatoriaus sugeneruotos reikšmės



11 pav. Linijinė interpoliacija



12 pav. Kubinė interpoliacija

Perlino triukšmas gali būti naudojamas dvimatėje ir trimatėje grafikoje, kietų objektų arba tūrio tekstūrų pavidalu. Tūrinės tekstūros nuo įprastų dvimačių tekstūrų skiriasi tuo, kad jos neapsibrėžia tiksliais geometriniais taškais, bet geba susilieti su kitomis kieto kūno

tekstūromis. Tūrinės tekstūros suteikia didesnį realybės pojūtį nei dvimatės. Tūrinės tekstūros skaičiavimais nėra brangūs, tačiau turi labai aukštus atminties ir saugojimo reikalavimus. Perlino triukšmas reikalauja laiko saugojant informaciją, taip panaikindamas didelius saugomos informacijos kiekius. Naudojant „pixel-shader“ įrangą bei nauja GPU procesorių, yra galimybė tūrinius objektus atvaizduoti realiu laiku. Perlino triukšmas suteikia daug privalumų. Parametrinė kontrolė suteikia kūrėjui galimybę lanksčiai kontroliuoti aukštus išėjimo parametrus. Naudojant algoritmą, kuriant įvairias geometrines detales bei tekstūras, galimas mažas atminties eikvojimas, viską apibrėžiant keliais parametrais. [18], [19]

3.2.4. Sluoksninis tekstūravimas

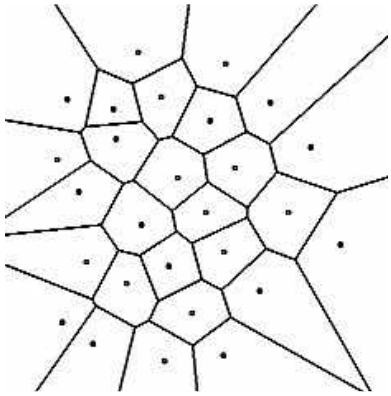
Sluoksninis tekstūravimas yra vienas iš pagrindinių procedūrinių metodų ir yra tradiciškai taikomas žaidimų kūrime. Iš pradžių sluoksninis tekstūravimas buvo naudojamas dvimatėje grafikoje, suskaidant vaizdą į mažas dalis, kurios yra pakartojamos bei surenkamos, sudarant virtualaus pasaulio vaizdą. Šiuo metu sluoksninio tekstūravimo technika naudojama kuriant didelio sudėtingumo tekstūras, jas suskaidant į daug sluoksnių ir sudedant. Šis sprendimas leidžia tekstūruoti tuos objektus, kurių neįmanoma tekstūruoti naudojant aukštos kokybės struktūrą kitais metodais.

Išplėstiniai algoritmai naudoja stochastinę informaciją sudarydami sudėtingas tekstūras. Paveikslas sudaromas iš daug skirtingų dalelių. Naudojant išplėstinius algoritmus yra naudojami dalelių išsidėstymo draudimai, pagal kuriuos dalelės gali būti sujungtos esant tam tikroms sąlygoms, arba yra sujungiamos tiesiogiai. Naudojant pseudo atsitiktinių reikšmių funkciją, įmanoma iš vieno žemėlapio sudaryti tūkstančius paveikslėlių. Sudėtingos bei didelės tekstūros gali būti sukuriamos naudojant stochastinę informaciją ir mažas tekstūros daleles. Tokiu būdu sukurtos tekstūros gali būti panaudojamos internetiniuose žaidimuose, kai yra dalijamasi žaidimo resursais; tokiu būdu yra minimizuojami reikalavimai atminčiai ir suteikiama galimybė atvaizduoti plataus masto virtualius pasaulius. [17], [11]

3.2.5. Voronoi diagramos

Voronoi diagramos buvo pademonstruotos kaip S.Worley metodas, kuriame jis patikslino algoritmą, kuris suskaido erdvę į atsitiktinės tvarkos masyvus, sukuriant ląstelių tekstūras. Ši technika buvo sukurta tokių esamų procedūrų papildymui, kaip Perlino triukšmas ir procedūrinės kartos metodai, tekstūruojant odos ar žievės ląstelių paviršius. Voronoi diagramos yra taikomos įvairiose mokslo programose: erdvės analizėje, miestų planavime, robotų technikoje, ekologijoje.

Voronoi diagramos yra metrinės erdvės skaidymas, kuris apibrėžta atstumo skaidymą į diskrečius objektus toje erdvėje. 13-ame paveikslėlyje matoma, kaip erdvė suskaidoma į ląsteles, apibrėžiant jas linijomis. Kiekviena sienos linija yra vienodai nutolusi nuo kiekvienos kaimyninių taškų poros. Keičiant konfigūraciją, galima sudaryti įvairius ląstelių pasiskirstymo modelius, keičiant taškų pasiskirstymo vietą. [11], [20]



13 pav. Voronoi diagrama [11]

3.3.Bazinių generavimo metodų taikymas miestų generatoriams

Prieš tai buvusiame poskyryje aptarti procedūriniai metodai yra skirti natūralių objektų ir tekstūrų, sutinkamų gamtoje, generavimui. Pastaruoju metu mokslininkai savo dėmesį sutelkė į kitas jų pritaikymo sritis – žmogaus sukurtų reiškinių ir objektų, tokių, kaip didmiesčiai, generavimą. Miestų generatoriai buvo kuriami daugeliu etapų, jiems sukurti buvo naudojami įvairūs kombinuoti metodai.Šiame poskyryje bus analizuojami moksliniai tyrimai, kurie buvo atlikti automatinių miestų generatorių kūrimo srityje.

3.3.1. Reljefo generavimas

Trimatei teritorijai modeliuoti trimatėje erdvėje yra du pagrindiniai metodai ir jų modifikacijos ar atmainos. Vienas iš metodų yra išsaugoti kiekvienos tinklelio viršūnės aukštį saugykloje. Kitas metodas – generuoti teritoriją, naudojant procedūrinius generavimo metodus.

- ***Aukščių žemėlapiai***

Kompiuterinės grafikos srityje, informacijos laikymui yra populiaru naudoti b i t m a p formato paveiksliukus. Aukščių žemėlapyje nespalvotas paveiksliukas yra naudojamas kiekvieno taško aukščiui plokštumoje atvaizduoti. Kuo tamsesnis taškas, tuo žemiau jis yra, kuo šviesesnis, tuo aukščiau.

Šis metodas dažnai naudojamas, kai teritorija turi būti specifiškai sumodeliuota ir nesikeičianti. Daug video žaidimų naudoja šį metodą teritorijų žemėlapių kūrimui.

- ***Procedūriniai metodai***

Pasitaiko atvejų, kai reljefas turi būti sugeneruotas atsitiktiniu būdu, arba dinamiškai kisti einant laikui (pavyzdžiui, modeliuojant vandenį). Procedūriniai metodai yra algoritmai, kurie sugeneruoja reljefą, kuriam nereikia saugyklos kiekvienos viršūnės vietai plokštumoje atsiminti; jie taip pat leidžia vartotojui modifikuoti trimatį objektą.

- *Poslinkių algoritmas*

Plokštuma briaunomis suskaidoma į daug dalių. Kiekviena iš tų dalių yra pakeliama arba nuleidžiama, taip yra kartojama keletą kartų, kol plokštuma tampa deformuota.

- *Apskritimo algoritmas*

Šis algoritmas yra panašus į poslinkių algoritmą, bet vietoj to, kad dalintų plokštumą briaunomis, ji yra sudalinama apskritimais. Vienas taškas, esantis apskritime yra pakeliamas aukštyn, o visi likę – nuleidžiami žemyn.

- *Vidurinio taško perkėlimas*

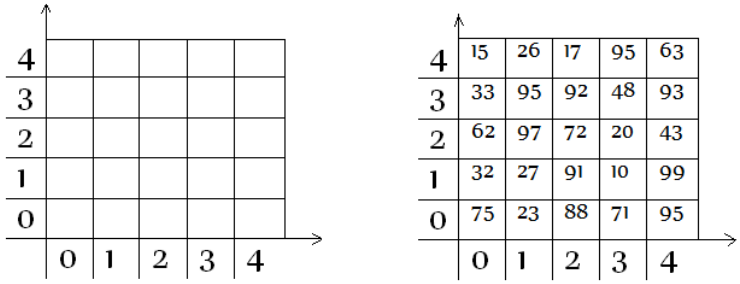
Iteracinis algoritmas, kuriame detalių ir gruoblėtumo skaičius išauga su kiekviena iteracija. Šis metodas yra labiau efektyvus, nei prieš tai buvę ir naudingas, kai resursai yra riboti. Kai gruoblėtumas yra lygus nuliui – reljefas yra plokščias, kuo šis skaičius didesnis, tuo gruoblėtesnis reljefas. [26]

3.3.2. Pastatų generavimas

Miesto pastatus procedūriškai sugeneruoti yra sunku dėl jų individualumo. Modernaus miesto pastatai yra skirtingų funkcijų ir stilių. Pastatai, kaip funkciniai vienetai, tarnauja specifiniam tikslui ir turi atitikti kaimyninio bloko ar rajono, bei paties miesto stilių. Yra daug skirtingų paskirčių pastatų ir jos yra derinamos su geografine kompozicija, kartu tai padaro miestą kompleksine sistema. Tokia kompleksinė sistema yra sudėtingas modelis, bet jos supaprastinta išeitis gali būti naudojama statistinės analizės srityje, kuri naudoja sumodeliuotų pastatų funkcijų klases ar grupes. Yra naudojamos komercinės, gyvenamųjų namų, pramoninės grupės gali būti naudojamos kaip apibendrinimas daugybei pastatų funkcijų ir paprastas, funkcinių pastatų modeliavimo be paties miesto, ar miesto viduje, mechanizmas. Pastato stilius, ypač geometrija ir medžiagos, dažnai yra architektūrinės ir kultūrinės įtakų pasekmės.

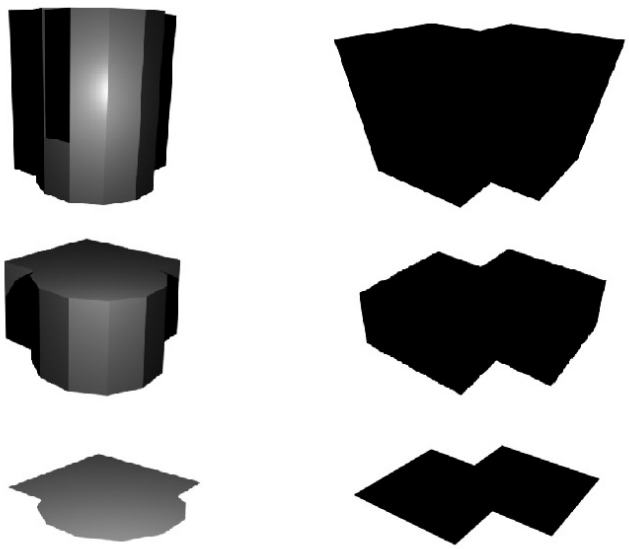
Pastatų generavimo sistema naudoja pastatų lokaciją tinklelio koordinatėse kaip „sėklą“ pastatų generavimui. Kiekvieno pastato atsiradimas priklauso nuo šios „sėklos“, įskaitant tokius nustatymus kaip aukštis, plotis, aukštų skaičius. Generuojant pastatus, naudojant panašius gretimus skaičius (tinklelio kaimyninių viršūnių), pastatai gali būti išdėstyti ne atsitiktinai, bet eilės tvarka. Tam, kad to būtų išvengta, yra naudojama maišymo funkcija (žr.

14 pav.). Paveikslėlyje yra parodyta, kaip skaičiai yra išdėliojami , kad būtų užtikrintas kuo didesnis pasiskirstymas. [12], [21]



14 pav. Tinklelis koordinačių ašyje ir funkcinis išsibarstymas

Pastato geometrija yra generuojama naudojant kombinuotus geometrinius primityvus, iš kurių yra statomos likusios pastatų dalys. Kiekviena pastato dalis yra konstruojama naudojant skirtingus pagrindų planus. Viršutiniai pastatų aukštai kuriami iš pastato pagrindo z ašimi ištempiant jo perimetrą. Pagrindai yra kuriami iš vienos ar kelių primityvių figūrų (žr.15 pav.).



15 pav. Pastatų generavimas iš plokštumos

3.3.3. Kelių generavimas

Kelių tinklas yra vienas iš pagrindinių miesto charakterio aspektų. Juos yra sunku sugeneruoti, nes jų generavimo algoritmuose yra susipynę kompleksinių sistemų komponentai.

Apžvelgiant miesto kelius arba miesto planą galime įžvelgti tam tikrą modelį. Yra daugybė kelių tinklų modelių realizuotų miestuose. Pradedant nuo smulkios kelių struktūros iki statmenų kelių modelių bei hierarchinių kelių modelių. Kelių sudarymo modeliui įtaką daro daug faktorių: geografinė padėtis, kultūra ir t.t. Miestai gali būti charakterizuojami pagal

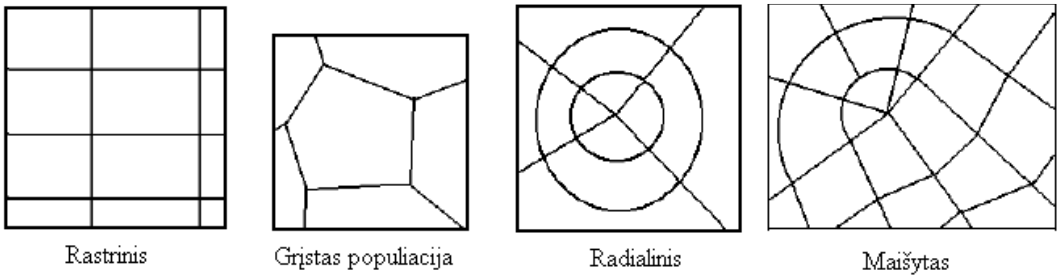
kelių tinkle esančius modelius: Niujorko keliai yra taškinio modelio rezultatas, Europos mietuose taikomas radialinis arba koncentrinis modeliai. Daugumoje miestų galima įžvelgti keletą skirtingų kelių tinklo modelių.

Vienas dažniausių kelių generavimo šaltinių yra centrinės linijos, kurios yra aprašomos vektoriais. Kelių tinklo topologija paprastai yra saugoma kaip susikirtimų taškai, kurie nurodo kelių segmentus, kurie juos sujungia. Kai kuriais atvejais keliai yra modeliuojami kaip:

- Vietoj polilinijų naudojamos kreivės, kurios naudojamos kaip centriniai keliai;
- Linijinės briaunos, nurodančios kelio ribas ar šaligatvio briauną;
- Keliai, sudaryti iš daugiakampių;
- Viena centrinė linija vienai eismo juostai, o ne visam keliui.

Keli procedūriniai metodai siūlo generuoti gatvių tinklus didelių miestų aplinkos kontekste. Toliau yra pateikiami keli populiariausi variantai (žr. 16 pav.):

- Gatvių tinklų generavimas, grįstas L– sistemomis;
- šabloninė kelio kreivė ir Voronoi diagramos: sukuriamas kelių tinklas priklausomai nuo gyventojų populiacijos;
- procedūriniai metodai: rastrinis, radialinis bei maišytas režimai tarnauja kaip paprasto modeliavimo įrankiai su kelių tinklu, prasidedančiu nuo pradinio taško ir besitęsiančio link išsiplėtimo ribų. Maišytas būdas yra tarpinis variantas tarp radialinio būdo ir rastrinio būdo.
- tenzorių laukų naudojimas gatvių grafams valdyti, kurie leidžia vartotojui interaktyviai redaguoti gatvės grafą, modifikuojant arba tenzorių, arba patį grafą tiesiogiai;
- alternatyvus būdas yra interaktyviai sintetinant didmiesčių gatvių tinklus. [22], [21], [20], [23]



16 pav. Skirtingi kelių šablonai [22]

Interaktyvus kelių keitimas ir modeliavimas

Buvo pasiūlyti keli galimi būdai interaktyviam kelių keitimui, modeliavimui ir eskizavimui:

- Sistema interaktyviai redaguoti didelius reljefus su labai detalizuotomis detalėmis, tokiomis kaip keliai, upės, ežerai, laukai;
- Sistema interaktyviai redaguoti kelių tinklą reljefe, kontroliuojant „piecewise clothoïd" kreives;
- Modeliuoti architektūrines scenas, keičiant jų formas ir kombinuojant su jau turimais tekstūruotais modeliais. Vėliau pagal scenos išsidėstymą sukuriant kelius. [23]

Variaciniai kreivių skaičiavimai

Keliai gali būti generuojami, naudojant variacines kreives, projektuojamas ant paviršiaus. Pasiūlytas algoritmas minimizuoja kvadratinės energijos funkcionalus, sutraukiant pirmą ir antrą išvestines. Keliai gali būti generuojami naudojant variacinės kreivės dizainą.

Daug metodų susitelkia ties izotropiniu atveju, kai sąnaudų funkcija, kuri priklauso tik nuo pozicijos. Keli algoritmai siūlo planinius regionus ir kliūčių erdves nusakyti daugiakampiais, kai sąnaudų funkcija yra nepriklausoma nuo greičio. Problemos sudėtingumas didėja, kai išlaidų koeficiento funkcija yra tęstinė bet vis dar nepriklausoma nuo greičio. Keletas efektyvių algoritmų izotropiniui trumpiausios kreivės sprendimui, buvo išsprendžiant diskrečias Hamilton Jacobi lygtis. [23]

3.3.4. Pastatų išdėstymo generavimas

Žemiau pateikiami dažniausiai naudojami pastatų išdėstymo metodai:

- Vektorinė sistema;
- Pastatų generavimas iš reljefo viršūnių, sudarant plokštumą;
- Pagal aukščio žemėlapį - keliai yra sugeneruoti aukščiau, nei likusi plokštumos dalis, ant likusios plokštumos dalies yra atsitiktinai generuojami pastatai;
- Kolizijos sprendimo būdų algoritmai. [12], [22]

4. PROJEKTO REALIZAVIMO ETAPAI

Išanalizavus sukurtus miestų generatorius ir galimus generavimo metodus, pasirenkami projekto realizavimo būdai. Miesto generatoriaus kūrimas yra skirstomas į tokius etapus:

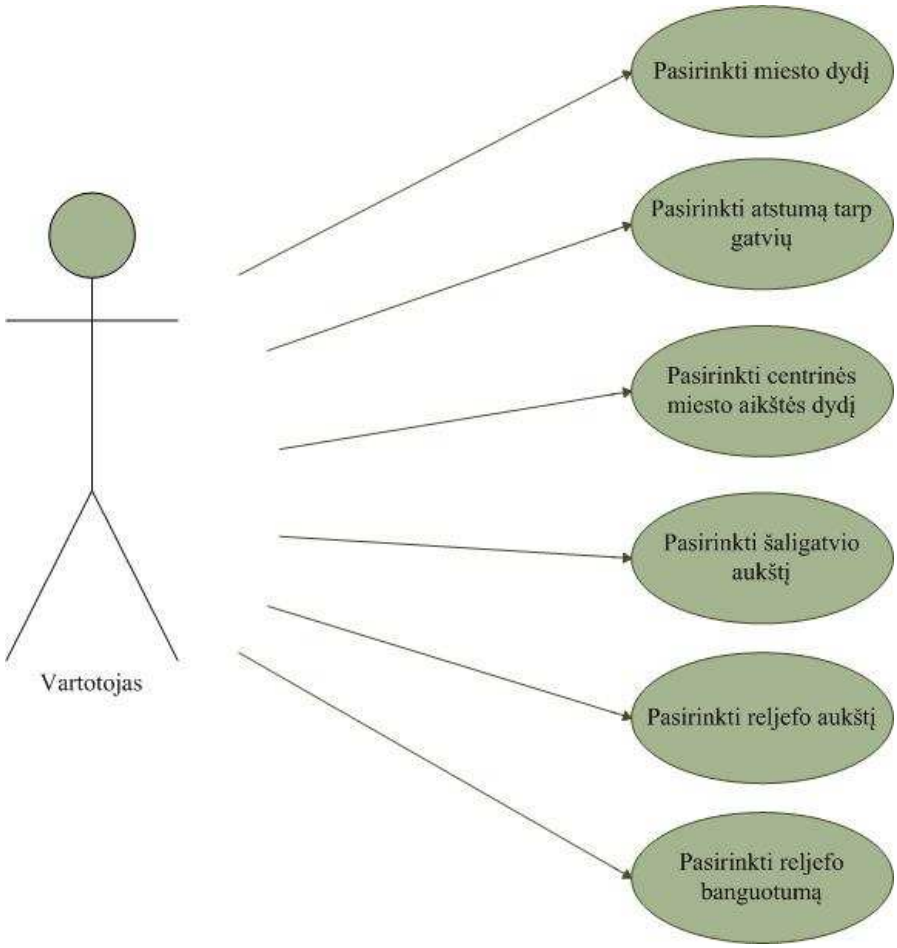
- projekto sukūrimas,
- programinės įrangos parinkimas,
- projekto įgyvendinimas. Projekto įgyvendinimą sudarantys etapai: reljefo generavimas, gatvių tinklo generavimas ir namų įkėlimas. Gatvių tinklo kūrimas yra skirstomas į tris etapus – šaligatvių generavimas, gatvių tinklo generavimas, centrinės miesto aikštės generavimas.

4.1.Projektavimas

Projekto kūrimas perteikiamas UML diagramų modeliais.

4.1.1. Panaudos atvejų diagrama

Panaudos atvejų diagrama vaizduoja vartotojo galimybes dirbant su programa (žr. 17 pav.).



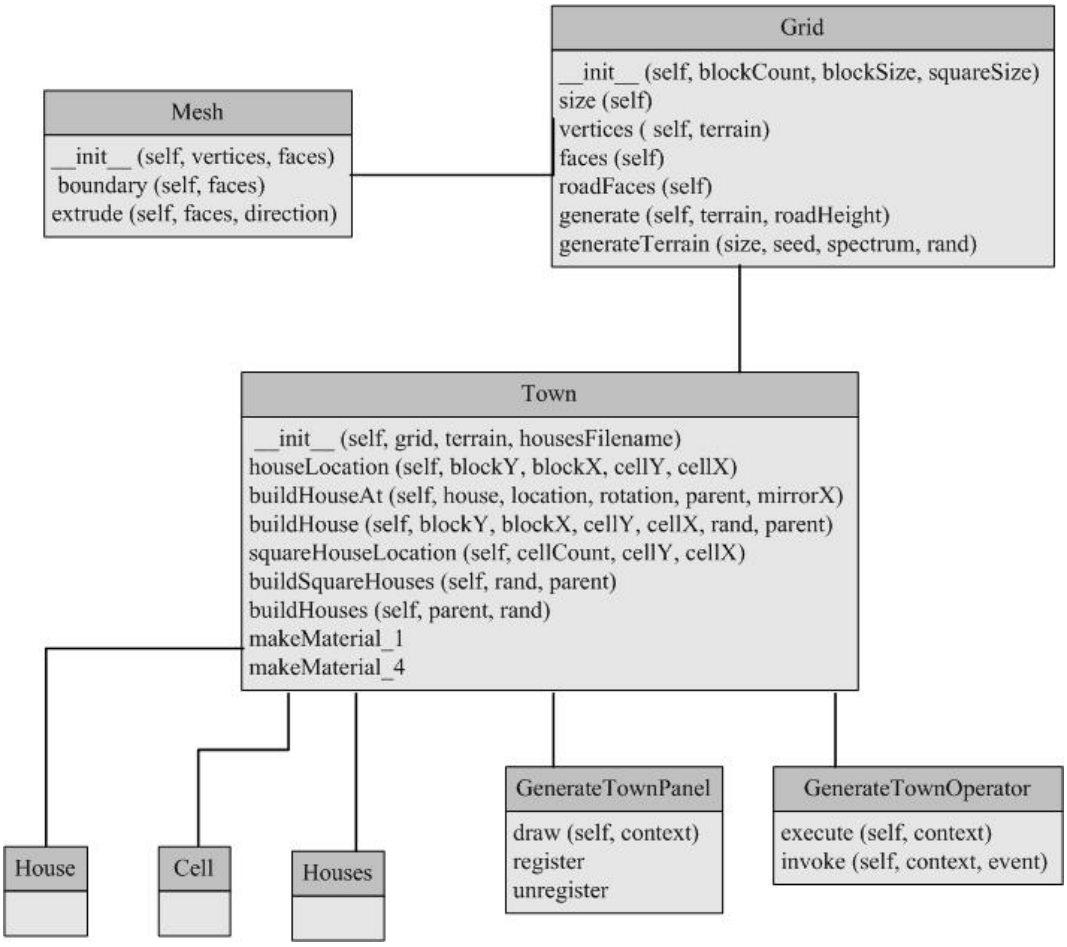
17 pav. Panaudos atvejų diagrama

9 lent. Panaudojimo atvejai

Panaudojimo atvejis	Aprašymas
Pasirinkti miesto dydį	Pasirenkamas miesto dydis
Pasirinkti atstumą tarp gatvių	Pasirenkamas atstumas tarp gatvių
Pasirinkti centrinės miesto aikštės dydį	Pasirenkamas centrinės miesto aikštės dydis
Pasirinkti šaligatvių aukštį	Pasirenkamas šaligatvių aukštis
Pasirinkti reljefo aukštį	Pasirenkamas reljefo bangų aukštis
Pasirinkti reljefo banguotumą	Pasirenkamas reljefo bangų tankis

4.1.2. Klasių diagrama

Klasių diagramoje yra pateikiamas projekto modelis, kuriame yra vaizduojamos klasių funkcijos ir jų tarpusavio ryšiai (žr. 18 pav.).



18 pav. Klasių diagrama

Town: sudedamas visas miestas, naudojamasi „Grid“ klase reljefo kūrimui, yra įkeliami ir išdėstomi pastatai.

Mesh: iš objekto ištempiamos plokštumos, automatiškai sukuriant šonines sienas (angl. *extrude*). Skaičiuojamos plokštumos ir viršūnės, jos aprašomos, sudaromi sąrašai perduodami kitai klasei.

Grid: kuriamas miesto žemėlapis, naudojama „Mesh“ klasė, perduodami duomenys, kaip iš objekto turėtų būti ištempiamos plokštumos, priskiriamos tekstūros ir „Blender“ programos objektas, kuris yra generuojamas su reljefu ir keliais.

House, Houses, Cell: tuščios klasės.

4.2.Programinė įranga

„Blender“ – tai daugiaplatformė 3D grafikos kūrimo programinė įranga. Yra pritaikoma animacijos, žaidimų kūrime. Skirtingai nuo šios programinės įrangos analogų, „Blender“ yra nemokama, turinti GNU GPL licenciją. „Blender“ pasižymi didele funkcijų gausa ir mažu atminties dydžiu kietajame diske. Papildomų funkcijų kūrimui yra naudojama „Python“ programavimo kalba. [24]

„Python“ – interaktyvi programavimo kalba, leidžianti naudoti įvairius programavimo stilius: objektinį, funkcinį, struktūrinį, aspektinį. Pagrindiniai „Python“ programavimo kalbos privalimai – lengvai suprantama, išmokstama, skaitoma kalba. Šiai kalbai pritaikomi ir kitų programavimo kalbų kompiliatoriai: Java (Jython), C (Cython), .NET (Iron Python), tai labai praplečia „Python“ panaudojimo galimybes. [25]

4.3.Metodai

4.3.1. Kelių generavimo metodas

Kelių tinklas buvo kuriamas, remiantis standartiniu Romos imperijos kolonijos kelių tinklu: visi keliai vienodo pločio, vienas nuo kito nutolę vienodais atstumais, miesto centre yra aikštė. Generuojamo miesto kelių tinklą sudaro skirtingos dalys – keliai, šaligatviai ir centrinė miesto aikštė.

Šaligatviai yra (d – 1) x (d – 1) kvadratai (d – atstumas tarp kelių; d–1 – šaligatviai, nes kelias užima 1 d). Miestas sudarytas iš b x b tokių kvadratų; b – kvadratų (d) skaičius, – lyginis (dėl simetrijos). Visas miestas padalintas i n x n vienetinių langelių, kur $n = db - 1$ (– 1 del to, kad pakraščiuose nėra kelių).

Langelių koordinatės – $(x; y) = 0 \dots n - 1$, centrinis langelis – $x = y = (n - 1)/2$.

x krypties kelių langeliai:

$$R_x = \{(x, df - 1): x = 0 \dots n - 1, k = 1 \dots b - 1\} \quad (3)$$

y krypties keliai (naudojantis simetrija):

$$R_y = \{(y,x): (x,y) \epsilon R_H\} \quad (4)$$

Aikštė per vidurį:

$$R_c = \{(x,y): x, y = (n-1)/2 - s \ldots (n-1)/2 + s\} \quad (5)$$

Visi langeliai, kurie yra keliai:

$$R = R_x \cup R_y \cup R_c \quad (6)$$

4.3.2. Objekto generavimo metodas

Landšaftas ir keliai yra sudaryti iš plokštumų. Kiekviena plokštuma turi 4 viršūnes ir 4 briaunas. Briauna yra viršūnių pora. Jei plokštumos viršūnės yra (v1, v2, v3, v4), tai jos briaunos – (v1, v2), (v2, v3), (v3, v4), (v4, v1). Plokštumos f viršūnių aibė žymima fv, briaunų aibė – fe.

Iš pradžių yra sudaromi šaligatviai, be kelių. Kiekvienas langelis yra plokštuma. Langelio (x, y) plokštumos viršūnės –

$$\big((x,y),(x+1,y),(x+1,y+1),y+1,x\big). \quad (7)$$

Viršūnės (x, y) koordinatės — $\big(x, y, h\,(x,y)\big)$, čia $h\,(x,y)$ — landšafto aukštis taške (x,y).

4.3.3. Paviršiaus iškėlimas

Sudaroma kelių plokštumų aibė R. Randamos briaunos, kurios riboja būsimąjį kelią ir šaligatvį. Jei briauna skiria dvi kelio plokštumas, tai šiose plokštumose briaunos viršūnių tvarka yra priešinga. Tegul

$$E = \bigcup_{f \epsilon R} f_e \quad (8)$$

yra visos kelio plokštumų briaunos, o

$$E' = \{(v_2, v_1): (v_1, v_2) \epsilon\, V\} \quad (9)$$

yra apsuktos briaunos. Tuomet $E_B = E \backslash E'$ bus ieškomos briaunos. Suskirstome į viršūnes.

$$V = \cup_{f \epsilon R} f_v \quad (10)$$

Viršūnės, esančios kelio pakraštyje:

$$V_B = \{v_1: (v_1, v_2) \epsilon\, E_B\} \quad (11)$$

Vidinės viršūnės:

$$E_1 = V \backslash V_B \quad (12)$$

Kiekvienai viršūnei $v \epsilon V$su koordinatėm (x, y, z) yra sukuriama nauja viršūnė $v' = (x, y, z - \Delta z)$, kur Δz yra šaligatvio aukštis. Visų kelio plokštumų ir briaunų viršūnės yra pakeičiamos atitinkamomis naujomis. Kiekvienai briaunai $(v_1, v_2)\, \epsilon\, E_B$ yra sukuriama nauja plokštuma su viršūnėmis (v_1, v_2, v'_2, v'_1). Tai yra tos vertikalios plokštumos, kurios jungia šaligatvį su keliu. Viršūnių V_1 nelieka, programa pakeičia jų koordinates, o ne naikina ir kuria atitinkamas naujas. Viršūnėms V_B yra kuriamos naujos viršūnės.

4.3.4. Landšafto generavimo metodas

Landšafto generavimui buvo naudojamas „Python“ atsitiktinių skaičių generatorius ir atvirkštinė Furjė transformacija. Taip buvo sukurtas fraktalinis reljefas.

Furjė transformacija yra grindžiama idėja, kad funkcija gali būti išreikšta skirtingų dažnių sinusų ar kosinusų bangomis. Furjė transformacija išreiškia funkciją šia forma, ji konvertuoja funkciją savo dažnių diapazone.

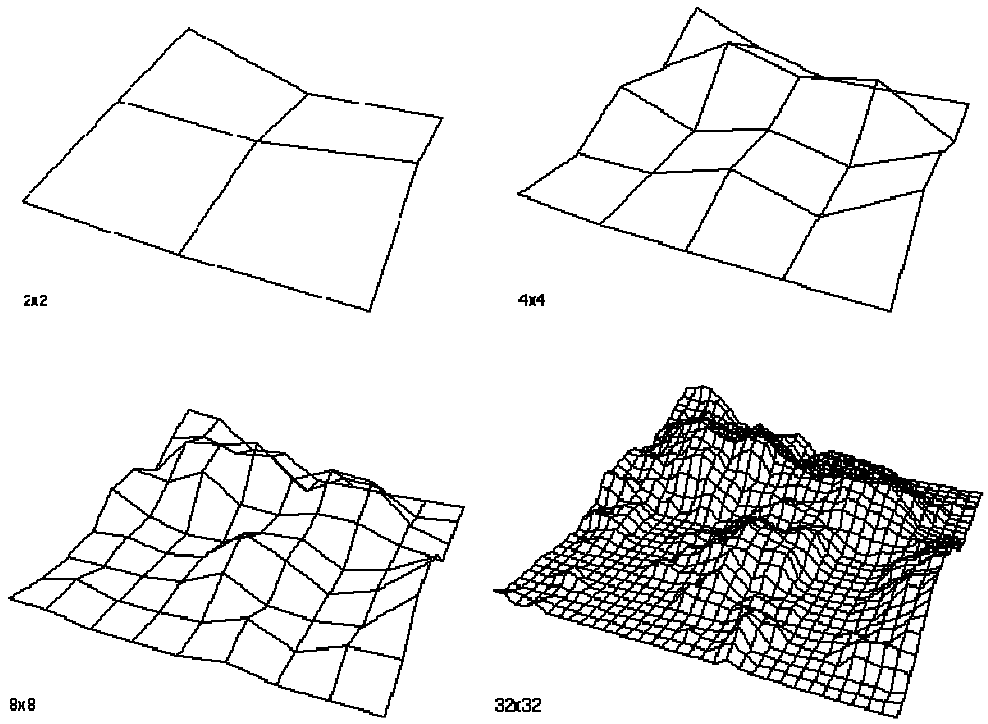
DFT yra Furjė transformacija, kuriai yra taikoma diskrečiųjų reikšmių aibė. Diskrečiųjų kompleksinių skaičių aibės rezultatai yra tokio paties dydžio, kaip pradinės aibės. Gautos reikšmės gali būti traktuojamos kaip pradinės aibės įvairių dažnių amplitudės. Procesas gali būti pakeistas nekeičiant pradinės aibės. DFT gali būti atliekama, kaip O(n·log(n)), naudojant sparčiosios Furjė Transformacijos algoritmą. DFT gali būti pritaikomas sudėtinėms dimensijoms.

Furjė transformacijas galima rišti su fraktalų reljefais, traktuojant reljefus kaip skirtingų dažnių bangų sumą. Pastebima, kad didėjant dažniui, mažėja bangų amplitudė.

Landšafto sukūrimui buvo naudojama tokia veiksmų seka:

1. Sukuriamas dvimatis reljefas, naudojant „Python“ atsitiktinių skaičių generatorių.
2. Pritaikoma dvimatė GFT.
3. Ištempiant arba sutraukiant (angl. *scale*) kiekvieną transformacijos reikšmę 1/fr, kur reikšmė atspindi dažnį ir yra šiurkštumo parametras.
4. Pritaikoma atvirkštinė GFT. Rezultatas yra fraktalinis reljefas.

Fraktaliniuose reljefuose nėra nenatūralių gūbrių ar viršūnių. Reljefai turi įdomią (kartais ir pageidautiną) atsikartojančių kraštinių savybę (ang. *tiled*). Viena reljefo pusė puikiai susilieja su kita (žr 19 pav.). [26]



19 pav. Fraktalinis reljefas [26]

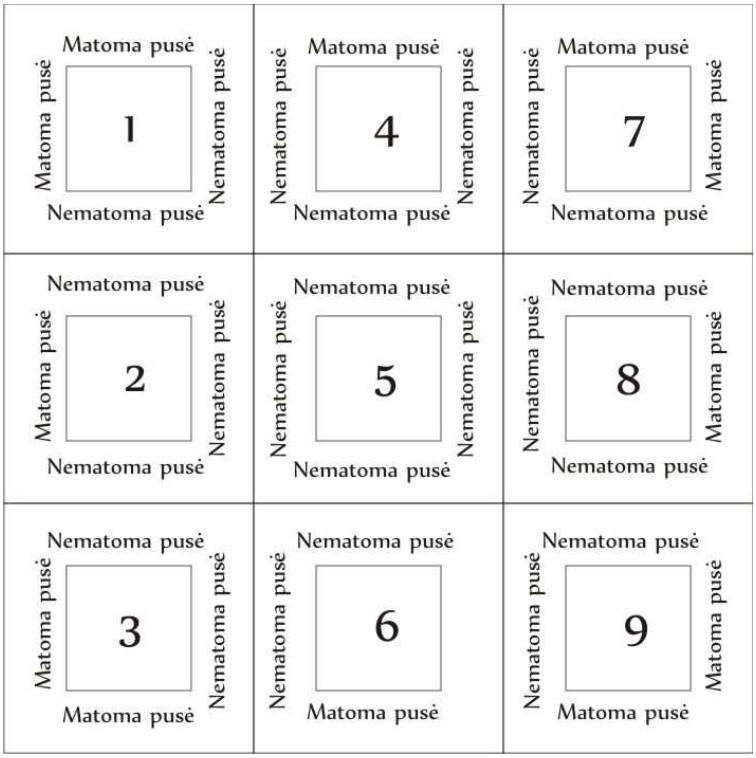
4.3.5. Pastatų įkėlimo metodas

Miesto generatoriuje naudojami pastatai yra suskirstyti į du tipus – paprastus ir centrinės miesto aikštės. Pastatai skiriasi tiek savo architektūra, tiek išdėstymo būdu. Pastatų įkėlimą sudaro du etapai – pasukimo kampas ir pastato pozicija z ašies atžvilgiu. Nepriklausomai nuo pastatų pasukimo kampo, jų padėtis z ašies atžvilgiu visada yra priklausomas nuo kvadrato, ant pastatas bus statomas. Pastato ašies centras yra sutapatinamas su žemiausia kvadrato viršūne. Taip yra užtikrinama, kad pastatai nebūtų per daug sulindę į reljefo vidų ir nekabotų ore.

4.3.6. Gyvenamųjų pastatų įkėlimo metodas

Visi generatoriuje naudojami pastatai yra sukelti į vieną failą. Kiekvieno pastato lokalios ašys sutampa su globaliomis – z ašies kryptis – į viršų, x ašies kryptis – dešinę, y ašies kryptis – gilyn. Pastato pavadinime yra nurodomos galimos matomos pastato pusės. Gali būti nuo 0 iki 4 matomų pastato pusių.

Tarp gatvių susidarę kvadratai yra skirstomi į 9 lygias dalis, taip kvadrate sukuriami 9 sklypai, ant kurių yra įkeliami pastatai. Vienas pastatas užima vieną sklypą. Priklausomai nuo to, kurios pastato pusės yra matomos, yra parenkama, kuriame sklype jis gali stovėti (žr. 20 pav.).



20 pav. Gyvenamųjų pastatų įkėlimas

Jei pastatas neturi nei vienos matomos pusės, jis gali būti statomas tik kvadrato viduryje esančiame langelyje (5 lang.). Jei pastatas turi 1 matomą pusę, jis gali būti statomas į kvadrato viduryje santį langelį, arba į vienos iš šoninių kraštinių vidurinįjį langelį (2, 4 , 5, 6 arba 8 lang.) . Jeigu pastatas turi 2, 3 arba 4 matomas puses, jis gali būti statomas bet kuriame langelyje.

Priklausomai nuo to, kuriame iš kvadratų stovės pastatas, yra parenkama, kuria puse jis bus atsukamas.

Visi pastatai yra modeliuojami ant vienodo dydžio kvadratų. Pastatų koordinačių centras sutampa su kvadrato centru. Įkeliant pastatą į generatorių yra parenkamas pastato pasukimo kampas ir, priklausomai nuo miesto dydžio ir atstumo tarp gatvių – pakeičiamas pastato 37 mastelis.

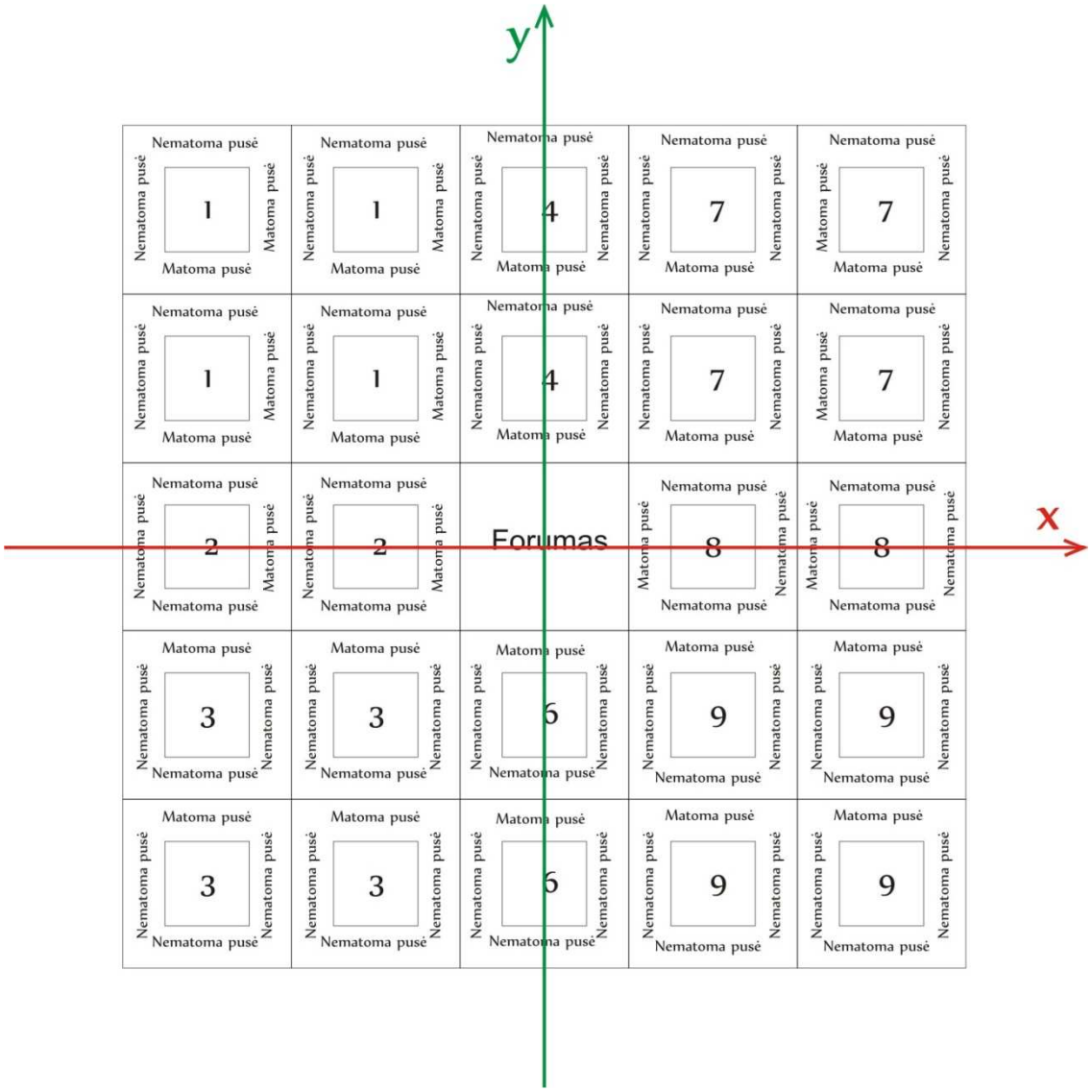
Pastatų išsidėstymas mieste yra parenkamas atsitiktinai.

4.3.7. Centrinės miesto aikštės pastatų įkėlimo metodas

Pastatų išsidėstymo tankumas centrinėje miesto aikštėje yra pasirenkamas dydis. Yra galimybė pasirinkti centrinės miesto aikštės pastatų tankumą nuo 0 iki 1 (kai 0 – aikštė tuščia, kai 1 – maksimaliai užpildyta). Didžiausias pastatų skaičius aikštėje yra 10.

Aikštės centrinėje dalyje yra tuščias plotas, neapstatytas pastatais, vadinamas forumu. Jo dydis yra tiesiogiai proporcingas aikštės dydžiui.

Visa aikštė yra sudalinta į langelius. Viename langelyje telpa vienas pastatas. Kiekvienas pastatas gali turėti nuo 0 iki 4 matomų pusių. Pastatas yra atsukamas taip, kad jeigu jis yra toje pačioje ašyje kaip ir forumas, tai matoma pusė bus tik ta, kuri yra atsisukusi į forumo pusę. Jeigu pastatas yra šalia forumo ašies, tai jis yra pasukamas taip, kad pratęsus forumą x ir/arba y ašimis ir pastato gerąją pusę viena iš šių ašių iki begalybės, jie susikirstų (žr. 21 pav.).



21 pav. Centrinės miesto aikštės pastatų įkėlimas

5. VARTOTOJO DOKUMENTACIJA

5.1. Funkcinis aprašymas

Programos paskirtis – galimybė kurti trimačius klasikinės Romos imperijos kolonijos miestus. Kiekvienas sukurtas miestas yra unikalus net ir pasirenkant tokius pačius parametrus.

Programa kuria įvairaus dydžio miestus, galimi tokie parametrų pasirinkimai:

- miesto dydis,
- atstumas tarp kelių,
- aikštės dydis,
- šaligatvio aukštis,
- reljefo aukštis,
- reljefo tankis.

Tai yra „Blender“ programos paprogramė, todėl, iškilus poreikiui, vartotojas gali pats tobulinti programą.

5.2.Reikalavimai programinei ir techninei įrangai

Operacinės sistemos reikalavimai:

- Windows XP,
- Windows Vista,
- Windows 7,
- Mac OS X 10.5 ir vėlesnės versijos,
- Linux ,
- FreeBSD.

Minimalūs techninės įrangos reikalavimai:

1 GHZ Single Core CPU,
512 MB RAM,
1024 x 768 px dydžio vaizduoklis palaikantis 16 bitų spalvas,
3 mygtukų pelė ,
Open GL grafikos plokštė, turinti 64 MB RAM.

5.3.Programos paleidimo dokumentas

Programa nėra autonominė, jos veikimui reikalingos kitos programos, kurių pagalba ji atlieka savo funkcijas. Reikalingų programų sąrašas:

- Blender 2.62 – parsisiųsti galima iš:
  http://download.blender.org/release/Blender2.62/

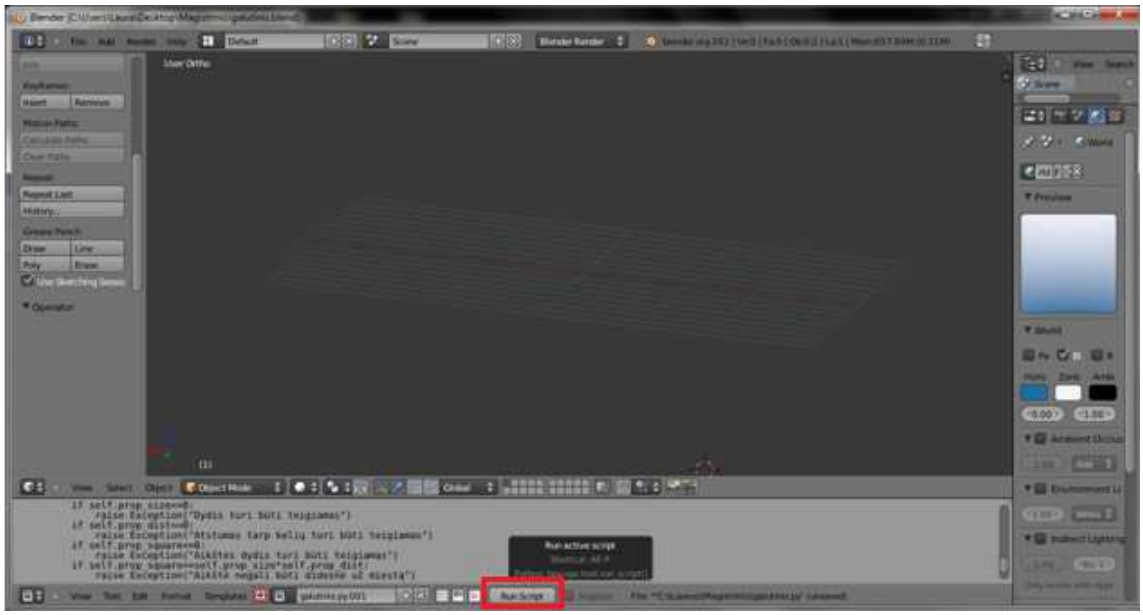
- Python 3.2.3 – parsisiųsti galima iš: http://www.python.org/ftp/python/3.2.3/python-3.2.3.msi
- Numpy 1.6.1 – parsisiųsti galima iš: http://sourceforge.net/projects/numpy/files/NumPy/1.6.1/

Parsisiuntus ir instaliavus reikalingas papildomas programas, gali būti pradedamas darbas su miestų generatoriumi

Prie generatoriaus yra pridedamas failas namai.blend. Jis turi būti patalpintas toje pačioje direktorijoje, kurioje yra generatorius.

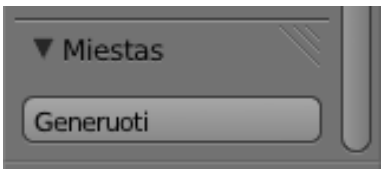
5.4.Programos vadovas

Pradedant naudotis programa paleidžiamas failo “generatorius.blend” vykdymas (žr. Priedas nr. 1). Generatoriaus veikimas pradedamas paspaudžiant mygtuka “Run Script” (žr. 22 pav.) arba paspaudžiant „Alt + P“.

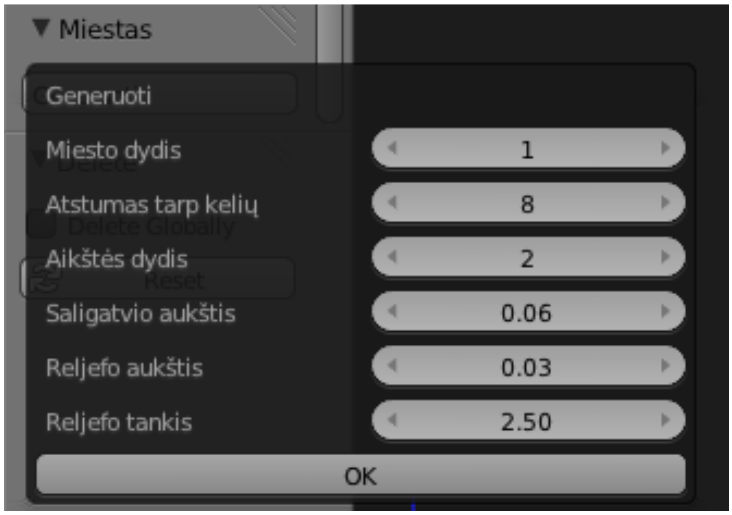


22 pav. Generatoriaus vykdymo pradžia

Paleidus programos vykdymą, atsiranda naujas meniu, kuriame yra mygtukas „Generuoti“ (žr. 23 pav.). Pasirinkus šią funkciją, sukuriamas naujas meniu, kuriame galima pasirinkti būsimo miesto parametrus (24 pav.).



23 pav. Mygtukas „Generuoti“



24 pav. Miesto parametrai

Reljefas susideda iš kvadratinių plokštumų, kurių dydis yra 1x1. Sakykime, kad 1x1 = a, čia a – vienas matavimo vienetas *(Blender unit).*

Į miesto generavimo meniu yra suvedami dydžiai. Šie dydžiai yra santykiniai, tikrieji būsimojo miesto metmenys yra apskaičiuojami pagal žemiau pateiktas formules.

Gatvės plotis visada yra 1 a.

Miesto dydis apskaičiuojamas: Dydis = c*b – 1,

čia c – atstumo tarp kelių reikšmė, b – miesto dydžio reikšmė.

Atstumas tarp kelių apskaičiuojamas: Dydis = c– 1,

čia c – atstumo tarp kelių reikšmė.

Aikštės dydis apskaičiuojamas: Dydis = d * 2 + 1,

čia d – aikštės dydžio reikšmė.

Šaligatvio aukštis: e,

čia e – šaligatvio aukščio reikšmė.

Reljefo generavimui yra naudojama funkcija –

$$\text{Dydis} = f * x^{-g}\ , \tag{13}$$

čia f – landšafto aukštis, g – landšafto tankis, x – kintamasis (dažnis).

6. TESTAVIMAS

Programos testavimas yra skirtas programos kokybės nustatymui. Jis suteikia informaciją apie programos kokybę aplinkoje, kurioje ji turėtų veikti. Šiame skyriuje aprašomas miestų generatoriaus testavimas, padalintas į dvi dalis. Pirmojoje dalyje testuojama programos reakcija į neteisingai įvestas reikšmes. Antrojoje dalyje yra ieškomos optimalios parametrų reikšmės generuojamam miestui, atsižvelgiant į jo dydį.

6.1.Klaidų apdorojimas

Programoje esantys parametrų apribojimai:

- miesto dydis turi būti teigiamas,
- atstumas tarp kelių turi būti teigiamas,
- aikštės dydis turi būti teigiamas,
- aikštė negali būti didesnė už miestą.

Šiame poskyryje yra aprašomas testavimas, kurio metu programai įvedami parametrai, kurių reikšmės prieštarauja duotiems apribojimams.

10 lent. Parametrų apribojimas „Miesto dydis turi būti teigiamas“

Miesto dydis turi būti teigiamas

Reikšmė	Rezultatas
– 1	Miestas negeneruojamas, pateikiamas klaidos pranešimas: „Dydis turi būti teigiamas“ (žr 25 pav.).  **25 pav.** Klaidos pranešimas „Dydis turi būti teigiamas“
a	Parametro įvedimo laukelyje įvedus reikšmę „a“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.
&	Parametro įvedimo laukelyje įvedus reikšmę „&“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.

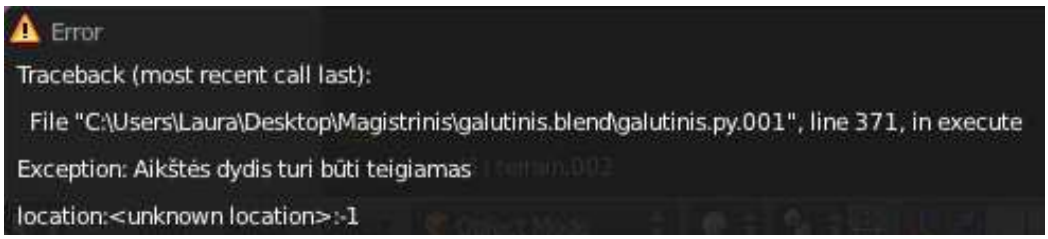
11 lent. Parametrų apribojimas „Atstumas tarp kelių turi būti teigiamas“

Atstumas tarp kelių turi būti teigiamas

Reikšmė	Rezultatas
– 1	Miestas negeneruojamas, pateikiamas klaidos pranešimas: „Atstumas tarp kelių turi būti teigiamas“ (žr. 26 pav.).  **26 pav.** Klaidos pranešimas „Atstumas tarp kelių turi būti teigiamas“
b	Parametro įvedimo laukelyje įvedus reikšmę „b“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.
*	Parametro įvedimo laukelyje įvedus reikšmę „*“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.

12 lent. Parametrų apribojimas „Aikštės dydis turi būti teigiamas“

Aikštės dydis turi būti teigiamas

Reikšmė	Rezultatas
– 1	Miestas negeneruojamas, pateikiamas klaidos pranešimas: „Aikštės dydis turi būti teigiamas“ (žr. 27 pav.).  **27 pav.** Klaidos pranešimas „Aikštės dydis turi būti teigiamas“
c	Parametro įvedimo laukelyje įvedus reikšmę „c“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.

^	Parametro įvedimo laukelyje įvedus reikšmę „^“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.

13 lent. Parametrų apribojimas „Aikštė negali būti didesnė už miestą**“**

Aikštė negali būti didesnė už miestą

Reikšmė	Rezultatas
Miesto – 1 Aikštės – 3	Miestas negeneruojamas, pateikiamas klaidos pranešimas: „Aikštė negali būti didesnė už miestą“ (žr. 28 pav.).  **28 pav.** Klaidos pranešimas „Aikštė negali būti didesnė už miestą“
d	Parametro įvedimo laukelyje įvedus reikšmę „d“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.
$	Parametro įvedimo laukelyje įvedus reikšmę „$“, ji automatiškai pakeičiama į reikšmę, esančią numatytuose parametruose.

6.2.Rezultatų priklausomybė nuo parametrų reikšmių dydžių

Nuo įvedamų parametrų reikšmių priklauso generuojamo miesto vaizdas. Pasvėrus santykius tarp įvedamų reikšmių yra gaunami skirtingi miestai. Šio testavimo tikslas yra surasti optimalias parametrų reikšmes generuojamiems miestams.

Tinkamam reikšmių parinkimui turi įtakos miesto dydis. Buvo parinkti standartizuoti miestų didžiai:

- mažas: miesto dydis nuo 1 iki 2,
- vidutinis: miesto dydis 3 nuo iki 4,
- didelis: miesto dydis nuo 5.

6.2.1. Mažo dydžio miestų parametrų reikšmių testavimas

Atsižvelgiant į miesto dydžiui keliamus reikalavimus, pirmojo bandymo miesto dydžio parametro reikšmė yra 1. Kitų galimų parametrų reikšmės taip pat yra lygios 1, išskyrus tuos atvejus, kai tokio dydžio reikšmės prieštarauja nurodytiems apribojimams.

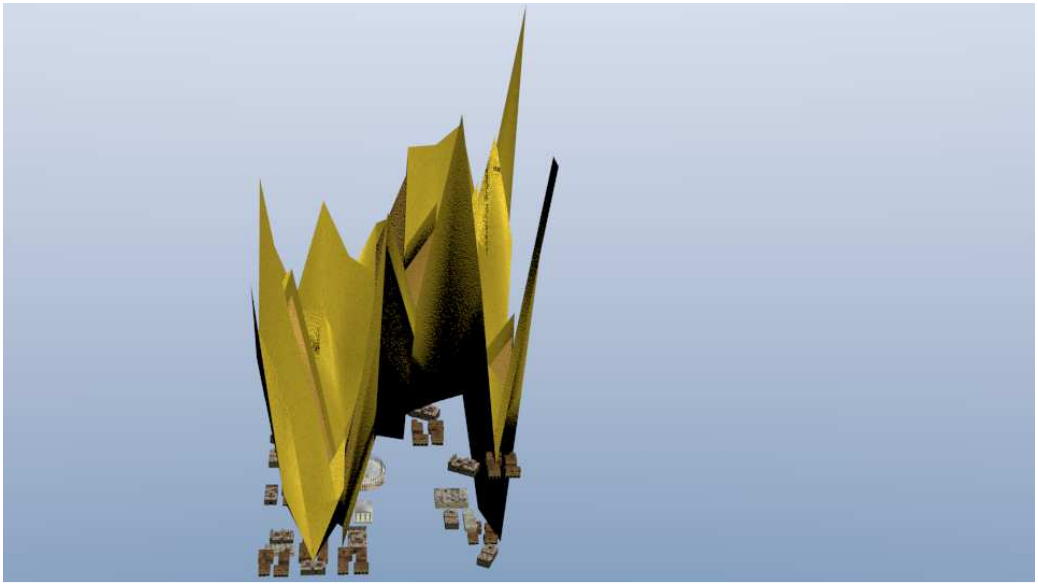
Skaičiuojant tikrąjį miesto aikštės dydį pagal formulę Dydis = d * 2 + 1, (miesto aikštės dydžio parametro reikšmė yra 1), gaunamas atsakymas 3. Yra žinoma, kad aikštės dydis negali būti didesnis už miesto dydį, todėl būsimojo miesto dydis yra $c*b - 1 \leq 3$. Jeigu miesto dydžio parametro reikšmė yra 1, tai atstumo tarp kelių reikšmė yra ≥ 4.

14 lent. Pirmojo bandymo reikšmės

Pirmojo bandymo metu miestų generatoriui suvedamos parametrų reikšmės

Parametras	Reikšmė
Miesto dydis	1
Atstumas tarp kelių	5
Aikštės dydis	1
Šaligatvių aukštis	1
Reljefo aukštis	1
Reljefo tankis	1

Suvedus reikšmes į generatoriaus parametrų reikšmių įvedimo meniu ir paleidus programos veikimą, buvo sugeneruotas miestas (žr. 29 pav.).



29 pav. Pirmasis bandymas

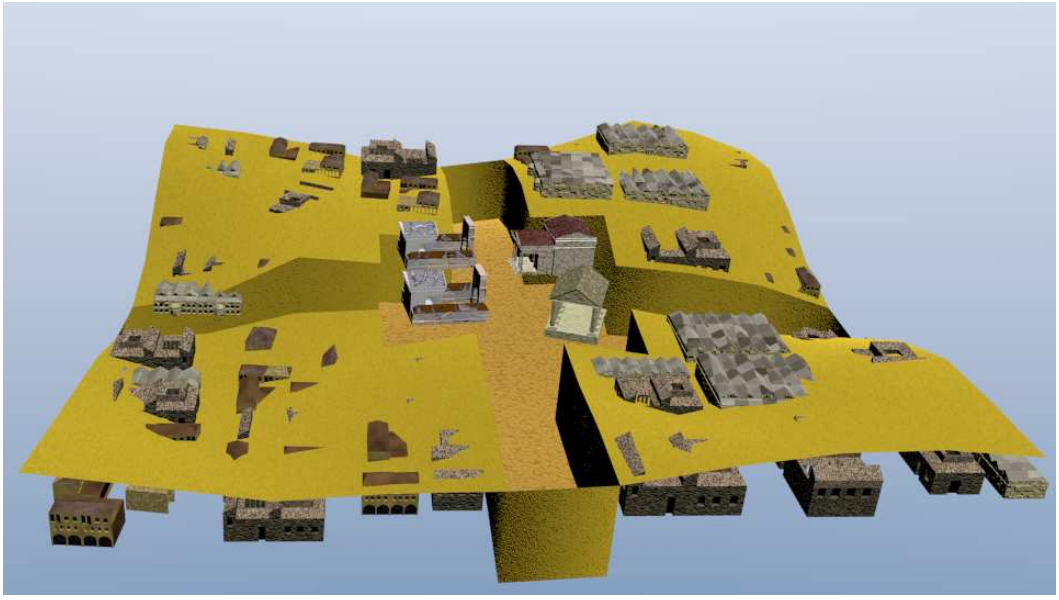
Iš paveikslėlio yra matoma, kad pagrindinė problema yra netinkamai parinktos landšafto aukščio ir banguotumo parametrų reikšmės. Antrojo bandymo metu bus naudojamos tos pačios parametrų reikšmės, kaip ir pirmojo, išskyrus reljefo aukščio ir banguotumo.

15 lent. Antrojo bandymo reikšmės

Antrojo bandymo metu miestų generatoriui suvedamos parametrų reikšmės

Parametras	Reikšmė
Miesto dydis	1
Atstumas tarp kelių	5
Aikštės dydis	1
Šaligatvių aukštis	1
Reljefo aukštis	0.1
Reljefo tankis	2

Suvedus reikšmes į generatoriaus parametrų reikšmių įvedimo meniu ir paleidus programos veikimą, buvo sugeneruotas miestas (žr. 30 pav.).



30 pav. Antrasis bandymas

Iš paveikslėlio yra matoma, kad yra netinkamai parinktos šaligatvių aukščio, atstumo tarp kelių, aikštės dydžio ir reljefo aukščio bei banguotumo parametrų reikšmės.

16 lent. Trečiojo bandymo reikšmės

Trečiojo bandymo metu miestų generatoriui suvedamos parametrų reikšmės

Parametras	Reikšmė
Miesto dydis	1
Atstumas tarp kelių	7
Aikštės dydis	2
Šaligatvių aukštis	0.1
Reljefo aukštis	0.05
Reljefo tankis	2.5

Suvedus reikšmes į generatoriaus parametrų reikšmių įvedimo meniu ir paleidus programos veikimą, buvo sugeneruotas miestas (žr. 31 pav.).

31 pav. Trečiasis bandymas

Iš paveikslėlio yra matoma, kad yra netinkamai parinktos šaligatvių aukščio, atstumo tarp kelių ir reljefo aukščio parametrų reikšmės.

17 lent. Ketvirtojo bandymo reikšmės

Ketvirtojo bandymo metu miestų generatoriui suvedamos parametrų reikšmės

Parametras	Reikšmė
Miesto dydis	1
Atstumas tarp kelių	8

Aikštės dydis	2
Šaligatvių aukštis	0.05
Reljefo aukštis	0.03
Reljefo tankis	2.5

Suvedus reikšmes į generatoriaus parametrų reikšmių įvedimo meniu ir paleidus programos veikimą, buvo sugeneruotas miestas (žr. 32 pav.).

32 pav. Ketvirtasis bandymas

Iš paveikslėlio yra matoma, kad parametrų reikšmės yra parinktos optimaliai. Daugiau optimaliai parinktais parametrais sugeneruotų mažų miestų rezultatų yra pateikiama pirmajame priede.

6.2.1. Vidutinio dydžio miestų parametrų reikšmių testavimas

Remiantis mažo dydžių miestų parametrų reikšmių testavimo rezultatais, vidutinio dydžio miestų testavimas yra vykdomas įvedant optimalias mažiems miestams pritaikytas parametrų reikšmes, išskyrus miesto dydį.

18 lent. Penktojo bandymo reikšmės

Penktojo bandymo metu miestų generatoriui suvedamos parametrų reikšmės

Parametras	Reikšmė
Miesto dydis	3

Atstumas tarp kelių	8
Aikštės dydis	2
Šaligatvių aukštis	0.05
Reljefo aukštis	0.03
Reljefo tankis	2.5

Suvedus reikšmes į generatoriaus parametrų reikšmių įvedimo meniu ir paleidus programos veikimą, buvo sugeneruotas miestas (žr. 33 pav.).

33 pav. Penktasis bandymas

Iš paveikslėlio yra matoma, kad yra netinkamai parinkta aikštės dydžio parametro reikšmė.

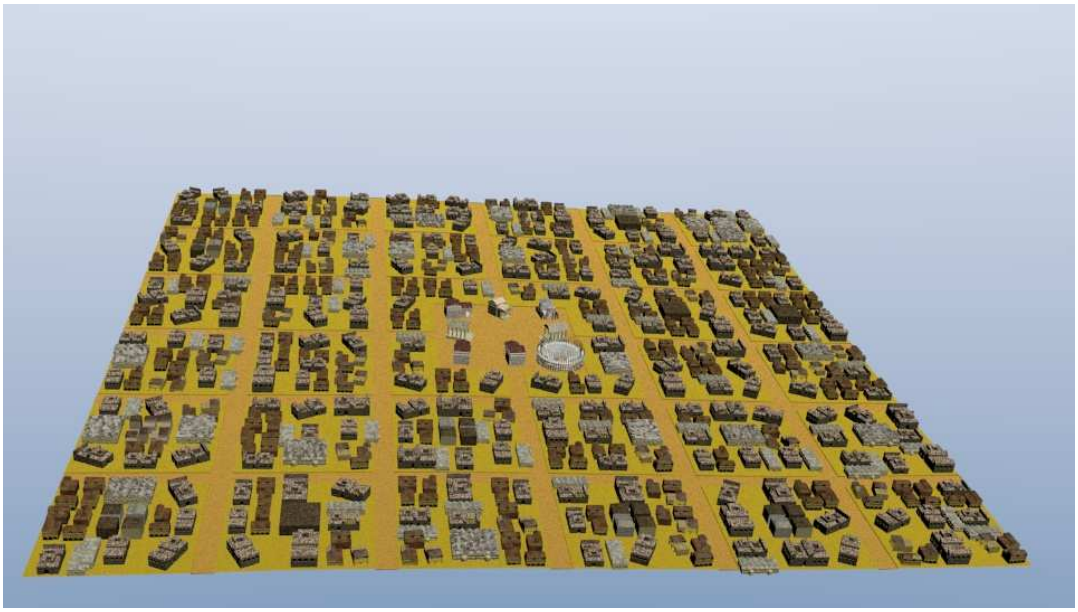
19 lent. Šeštojo bandymo reikšmės

Šeštojo bandymo metu miestų generatoriui suvedamos parametrų reikšmės

Parametras	Reikšmė
Miesto dydis	3
Atstumas tarp kelių	8
Aikštės dydis	4
Šaligatvių aukštis	0.05
Reljefo aukštis	0.03

Reljefo tankis	2.5

Suvedus reikšmes į generatoriaus parametrų reikšmių įvedimo meniu ir paleidus programos veikimą, buvo sugeneruotas miestas (žr. 34 pav.).



34 pav. Šeštasis bandymas bandymas

Iš paveikslėlio yra matoma, kad parametrų reikšmės yra parinktos optimaliai. Daugiau optimaliai parinktais parametrais sugeneruotų vidutinio dydžio miestų rezultatų yra pateikiama antrajame priede.

Didelių miestų generavimas reikalauja galingesnės kompiuterinės technikos, jų optimalios parametrų reikšmės apskaičiuojamos remiantis eksperimento eiga ir rezultatais.

.

7. IŠVADOS

1. Apžvelgus esamus miestų generatorius, nustatyta, kad jų galimybės generuoti senovės miestus yra labai ribotos.
2. Išanalizavus miestų generatorių generavimo metodus buvo pastebėta:
    - skirtingų objektų generavimui yra taikomi skirtingi generavimo metodai,
    - daugumos miestų generatorių kūrimui yra naudojami procedūriniai generavimo metodai,
    - visi miestų generatorių objektai yra generuojami, o ne įkeliami iš bibliotekų.
3. Atliktas klasikinių Romos imperijos kolonijų miestų generatoriaus sistemos projektavimas leidžia numatyti esminius sistemos blokus ir funkcines dalis, sutrumpinti sistemos realizacijos laiką.
4. Tinkamas programinės įrangos parinkimas suteikė galimybę prieiti prie sistemos programinio kodo – keisti, pritaikyti pagal poreikius, nebuvo reikiamybės kurti programos pagrindų, buvo sutaupytas laikas pagrindinės problemos sprendimo įgyvendinimui.
5. Išanalizavus Romėnų civilizacijos architektūrą ir miestų struktūrą, buvo sudarytas miesto išplanavimo taisyklių sąrašas.
6. Remiantis miestų ir erdvinių objektų generavimo analize, reljefo kūrimui buvo pasirinktas fraktalinis generavimo metodas.
7. Atlikus realizaciją buvo vykdomas eksperimentas, kurio pagalba buvo parinktos optimalios parametrų reikšmės estetiškai atrodančio miesto generavimui, atsižvelgiant į jo dydį.

8. LITERATŪRA

1. Šetkus B., Pobedinska L. Senovės istorija : vadovėlis. V.: Kronta, 2004. 22-36p.
2. David Gilman Romano. Corinth Computer Project: Roman Grid Plan. Iš corinth.sas.upenn [interaktyvus]. 2010, balandis [žiūrėta 2011-05-04]. Pieiga per internetą: http://corinth.sas.upenn.edu/gridplan.html.
3. Spiro Kostof. THE CITY SHAPED: Urban patterns and meanings through history. 2010, gruodis [žiūrėta 2011-03-15]. Prieiga per insternetą: http://203.77.194.71:81/Members/niyomi/THE%20GRID.pdf
4. Caner F. How to create a city by buildings in Maya?. 2010, gruodis [žiūrėta 2011-06-10]. Prieiga per insternetą: http://www.fxfx.org/index.php/2010/08/22/how-to-create-a-city-by-buildings-in-maya/
5. Allen B. CityEngine 2010 Indie. Iš 3D World Magazine [interaktyvus] .2010, sausis [žiūrėta 2011-03-22]. Prieiga per insternetą: http://www.mrcad.com/cityengine-2010-reviewed-procedural-city-generator/
6. Huber J. GhostTown Lite. Iš Scriptspot [interaktyvus] .2011, sausis [žiūrėta 2011-07-15]. Prieiga per insternetą: http://www.scriptspot.com/3ds-max/scripts/ghosttown-lite
7. Allen B. Suicidator City Generator. Iš 3D World Magazine [interaktyvus] .2011, rugsėjis [žiūrėta 2011-10-01]. Prieiga per insternetą: http://www.mrcad.com/cityengine-2010-reviewed-procedural-city-generator/
8. Reiter K. Blended Cities. Iš Jerome [interaktyvus] .2010, liepa [žiūrėta 2011-06-14]. Prieiga per insternetą: http://jerome.le.chat.free.fr/index.php/en/city-engine/documentation/libraries.html
9. Kelly G . Citygen: An Interactive System for Procedural City Generation. Iš Citygen [interaktyvus] .2010, lapkritis [žiūrėta 2011-10-023]. Prieiga per insternetą: http://procedural.googlecode.com/svn-history/r108/trunk/articles_cities/citygen_gdtw07.pdf
10. Aurbansim I. E01 – City Engine. Iš Iaarch [interaktyvus] .2010, spalis [žiūrėta 2011-04-10]. Prieiga per insternetą: http://www.ia.arch.ethz.ch/e01-cityengine-tutorial-part-1/
11. Kelly J.GA Survey of Procedural Techniques for City Generation. Iš ITB journal [interaktyvus] .2010, sausis [žiūrėta 2011-10-02]. Prieiga per insternetą: http://www.chrisleung.org/documents/UCL%20RG/Survey%20of%20Procedural%20City%20Generation.pdf
12. Jess M. Algorithmic Beauty of Buildings Methods for Procedural Building Generation. Iš Digitalcommons [interaktyvus] .2005, vasaris [žiūrėta 2011-07-10]. Prieiga per insternetą: http://digitalcommons.trinity.edu/cgi/viewcontent.cgi?article=1003&context=compsci_ho

nors&sei-redir=1&referer=http%3A%2F%2Fscholar.google.lt%2Fscholar%3Fq%3DAlgorithmic%2BBeauty%2Bof%2BBuildings%2BMethods%2B%2Bfor%2BProcedural%2BBuilding%2BGeneration%26hl%3Dlt%26as_sdt%3D0%26as_vis%3D1%26oi%3Dscholart%26sa%3DX%26ei%3DLWS5T7SZBdLY4QTM9Lz5CQ%26ved%3D0CAUQgQMwAA#search=%22Algorithmic%20Beauty%20Buildings%20Methods%20Procedural%20Building%20Generation%22

13. Jing S. Template-based generation of road networks for virtual city modeling. Iš Mendeley [interaktyvus] .2009, kovas [žiūrėta 2011-08-07]. Prieiga per insternetą: http://www.mendeley.com/research/templatebased-generation-of-road-networks-for-virtual-city-modeling/#
14. Alexandra D. THE RANDOM ITERATION ALGORITHM. Iš Rebe [interaktyvus] .2010, spalis [žiūrėta 2011-08-07]. Prieiga per insternetą: http://www.rebe.rau.ro/RePEc/rau/jisomg/SU08/JISOM-SU08-A28.pdf
15. Barnsley m. A fractal value random iteration algorithm and fractal hierarchy. Iš Anu [interaktyvus] .2011, spalis [žiūrėta 2011-10-04]. Prieiga per insternetą: http://wwwmaths.anu.edu.au/~barnsley/pdfs/barnsley_hutchinson_stenflo_a_fractal_valued_random_iteration_algorithm_and_fractal_hierarchy.pdf
16. Lanier E. Fractal Geometry and the Escape-time algorithm as Graphic Art Tools. Iš SDSU [interaktyvus] .2008, liepa [žiūrėta 2011-11-01]. Prieiga per insternetą: http://www.math.sdsu.edu/FractalPaper.pdf
17. Runions A. Graphical modeling using L-systems. Iš Algorithmicbotany [interaktyvus] .2010, kovas [žiūrėta 2011-11-18]. Prieiga per insternetą: http://algorithmicbotany.org/papers/abop/abop-ch1.pdf
18. Wijgerse S. Generating realistic city boundaries using two-dimensional Perlin noise. Iš Utwente [interaktyvus] .2007, vasaris [žiūrėta 2011-11-18]. Prieiga per insternetą: http://essay.utwente.nl/57836/1/scriptie_Wijgerse.pdf
19. Hart J. Perlin Noise Pixel Shaders. Iš Citeseerx [interaktyvus] .2009, rugsėjis [žiūrėta 2011-12-19]. Prieiga per insternetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.4088&rep=rep1&type=pdf
20. Ilangovan P. Procedural City Generator. Iš Buffalostate [interaktyvus] .2010, rugsėjis [žiūrėta 2011-10-21]. Prieiga per insternetą: http://www.buffalostate.edu/creativity/documents/pmccunef08.pdf

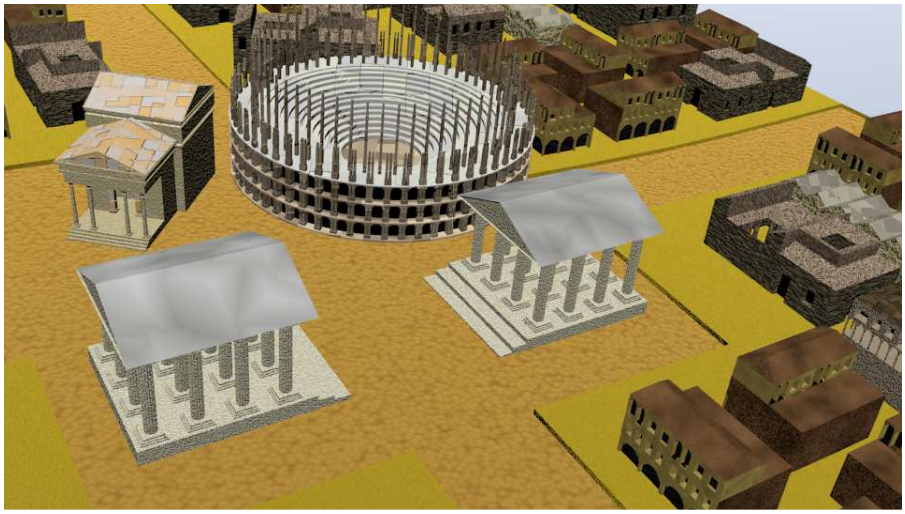
21. Teoh S. Algorithms for the Automatic Generation of Urban Streets and Buildings. Iš Sjsu [interaktyvus] .2011, gruodis [žiūrėta 2012-02-11]. Prieiga per insternetą: http://www.cs.sjsu.edu/~teoh/research/papers/CGVR08_city.pdf
22. Hausmann J. Automatic Generation of Cities in Real-time. Iš Unige [interaktyvus] .2010, kovas [žiūrėta 2011-05-16]. Prieiga per insternetą: http://www.qol.unige.ch/staff/hausmann/Experimentation_project_Hausmann.pdf
23. Galin E. Procedural Generation of Roads. Iš Eurographics [interaktyvus] .2010, sausis [žiūrėta 2011-05-18]. Prieiga per insternetą: http://liris.cnrs.fr/~egalin/Pdf/2010-roads.pdf
24. Wunnik J. A Graphical User Interface (GUI) for designing interactivity in game creation. Iš Blender [interaktyvus] .2010, kovas [žiūrėta 2011-07-11]. Prieiga per insternetą: http://www.blender.org/documentation/logic_editing_proposal.pdf
25. Swaroop C . A Byte of Python. Iš Ibiblio [interaktyvus] .2011, rugpjūtis [žiūrėta 2011-07-14]. Prieiga per insternetą: http://www.ibiblio.org/g2swap/byteofpython/files/120/byteofpython_120.pdf
26. Stanger K . Algorithms for Generating Fractal Landscapes. Iš Uwaterloo [interaktyvus] .2011, spalis [žiūrėta 2011-08-04]. Prieiga per insternetą: http://www.student.math.uwaterloo.ca/~pmat370/PROJECTS/2006/Keith_Stanger_Fractal_Landscapes.pdf
27. Valantinas J. Fraktalinė geometrija: vadovėlis. V.: Technologija, 1999. 22-36p.

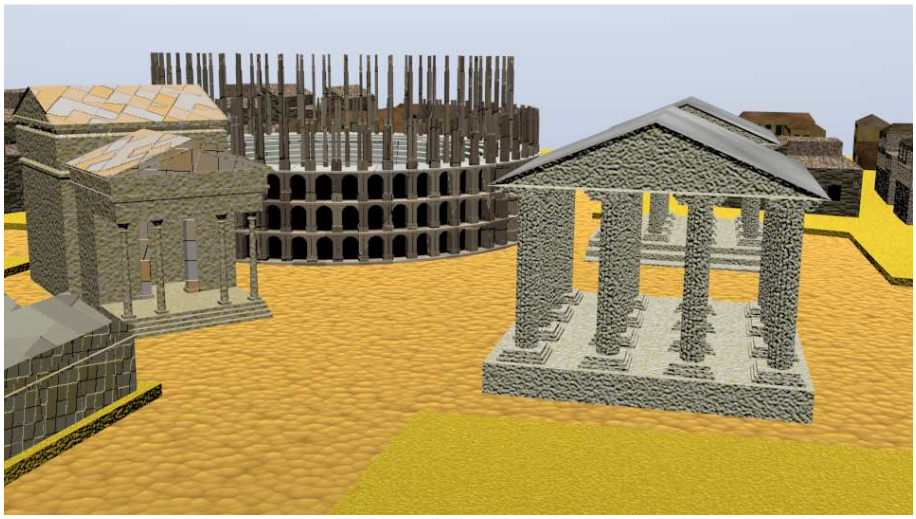
Paveikslėlių sąrašas

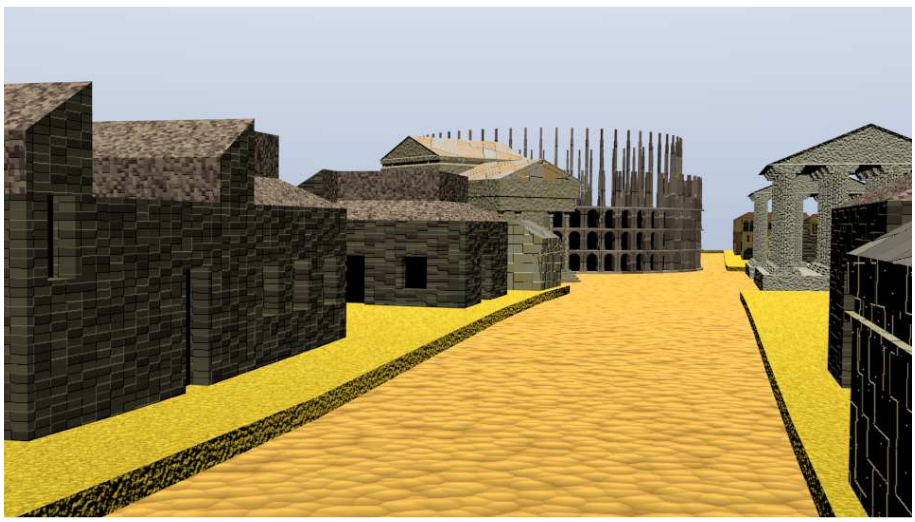
Lentelių sąrašas

9. PRIEDAI

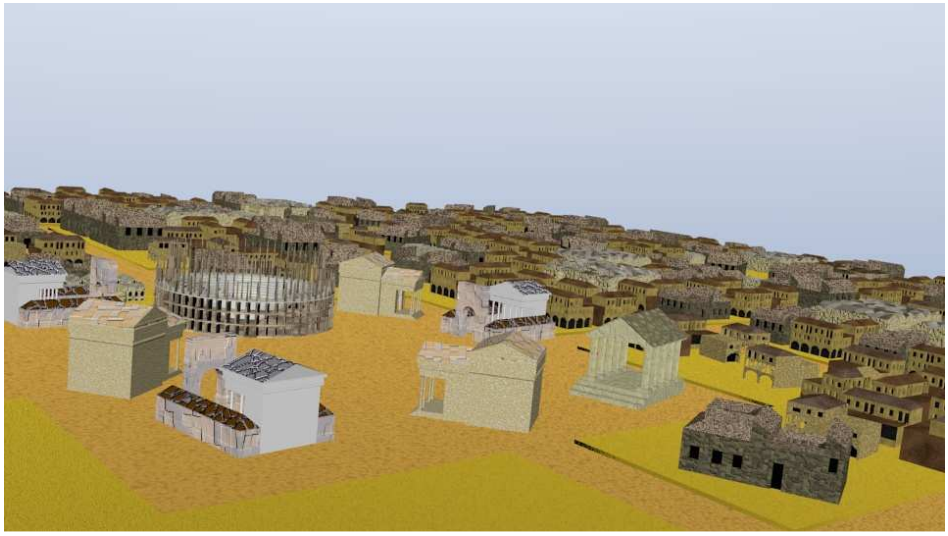
1 PRIEDAS. Mažo dydžio miestų generavimo rezultatai

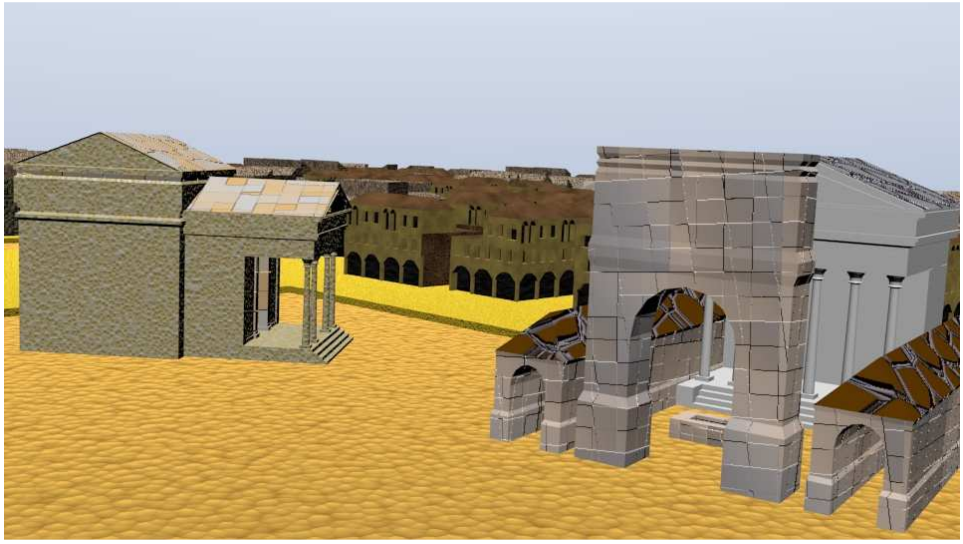


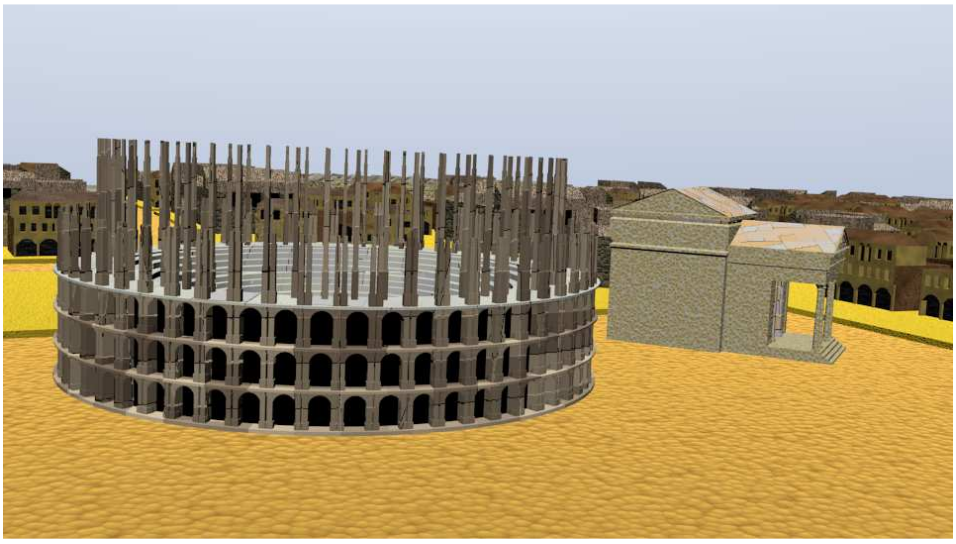




2 PRIEDAS. Vidutinio dydžio miestų generavimo rezultatai







3 PRIEDAS. Kompaktinis diskas