KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMATIKOS STUDIJŲ PROGRAMA

Ričardas Jonaitis

Vokselių taikymo kompiuterinėje grafikoje metodų ir galimybių tyrimas

Magistro darbas

Darbo vadovas:
doc. dr. Antanas Lenkevičius

Kaunas, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMATIKOS STUDIJŲ PROGRAMA

Ričardas Jonaitis

Vokselių taikymo kompiuterinėje grafikoje metodų ir galimybių tyrimas

Magistro darbas

Vadovas:
doc. dr. Antanas Lenkevičius
_____

Recenzentė
prof. Lina Nemuraitė
_____

Studentas
Ričardas Jonaitis IFM-3/1
_____

Kaunas, 2015

Summary

At the current time the most advanced types of computer graphics are based on polygons and textures. This kind of method is used widely in every kind of graphics, so it can be called traditional. Although now these methods are starting to use different kind of data structures to suppliment themselves to achieve more complex graphics effects. These kind of structures are called voxel data. This change in used data lead to question if it is possible to entirely get rid of polygons and switch to full voxel data for graphical representations. Voxel graphics is an old technology, it came into existance at about the same time as polygon graphics, but at that time polygon technology was way more efficient for computing with limited computational power. As the time progressed more and more technology was designed to further improve effectivennes of polygonal graphics, and voxel graphics were forgotten or reside only in specialized fields. At the present time now that the computational power is increasing exponentialy it is worth exploring the available possibilities to achieve using voxel technologies. Maybe it is time to switch from polygons to voxels.

In this work voxel technology applications are researched. Several of the available voxel imaging techniques will be analyzed, and evaluated by their strenghts and weakneses. Voxel and polygonal graphic capabilities will be compared.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Terminų ir santrumpų žodynėlis

Vokselis	Diskretizuotos tūrinės erdvės vienetas
Iso-surface	Tūrinius duomenis gaubiantis paviršius
Ray-casting	Duomenų paieška per erdvinę struktūrą leidžiant spindulį
Poligoninė grafika	Kompiuterinės grafikos rūšis, kurioje objektai apibrėžiami naudojant erdvines plokštumas
Vokselinė grafika	Kompiuterinės grafikos rūšis, kurioje objektai apibrėžiami diskretizuotais erdvės tūrio taškais
Octree	Hierarchinė duomenų struktūra rekursyviai dalinanti erdvę į 8 dalis
.obj	3D Geometrijos duomenų failas
Marching cubes	Algoritmas sukuriantis tūrį gaubiančią geometrinę figūrą
CUDA	Lygiagretaus programavimo platforma palaikoma NVidia grafinių procesorių
SIMD	Single instruction multiple data, operacijos atlikimas su keletu duomenų rinkinių
Normalė	Trimatis vektorius statmenas plokštumai

Įvadas

Šiuo metu kompiuterinės grafikos srityje labiausiai išvystyta vaizdų generavimas pagrįstas plokštumomis ir tekstūromis. Toks metodas jau ilgą laiką plačiai taikomas visose kompiuterinės grafikos srityse, todėl šį būdą galima laikyti tradiciniu. Tačiau tradicinėje kompiuterinėje grafikoje siekiant atvaizduoti sudėtingesnius ir kokybiškesnius vaizdus sukuriama vis daugiau pagalbinių algoritmų naudojančių vokselines duomenų struktūras. Ši tendencija sukėlė žymų susidomėjimą vokselinės grafikos piešimo metodais, bei iškėlė klausimą ar nevertėtų atsisakyti poligoninės grafikos metodų ir pilnai pereiti prie vokselinės grafikos. Vokselinė grafika nėra nauja technologija, jos idėja atsirado beveik tuo pačiu metu kaip ir poligoninės grafikos, tačiau tuometinės kompiuterių skaičiavimo galimybes buvo palankesnės poligonams. Per daugelį metų poligoninės grafikos metodai ir jai specializuota techninė įranga sparčiai ištobulėjo ir išpopuliarėjo, o vokselinė grafika buvo naudojama tik labai specifinėse srityse, todėl didelio populiarumo nesulaukė. Dabar išaugus kompiuterių skaičiuojamajai galiai, bei atsiradus vokselinės grafikos poreikiui atėjo metas įvertinti ar vokselinė grafika gali būti pakaitalu poligoninei grafikai.

Šiame darbe bus tiriama vokselių pritaikymo galimybės kompiuterinėje grafikoje. Analizuojama vokselinės grafikos piešimo metodai ir duomenų struktūros, įvertinama metodų teikiami privalumai ir trūkumai. Atliekamas vokselinės ir tradicinės kompiuterinės grafikos galimybių palyginimas.

1 Vokselių taikymo kompiuterinėje grafikoje analizė

1.1 Analizės tikslas

Analizės metu išsiaiškinti šiuo metu taikomus vokselinės grafikos piešimo metodus. Palyginti metodus tarpusavyje, nustatyti jų trūkumus ir privalumus. Surinkti informaciją reikalingą metodų tyrimui naudosimai programinei įrangai suprojektuoti.

1.2 Analizės tikslas

Tyrimo objektas – vokselinės grafikos piešimo metodai.

Vokselis – tai elementas, turintis informaciją apie diskretizuotos trimatės erdvės sritį. Erdvės diskretizavimas atliekamas norimą erdvę padalinant į baigtinį kiekį sričių, kiekvieną sritį apibūdinanti informacija yra patalpinama į vokselį. Vokselyje gali būti talpinama įvairi informacija, tai gali būti spalvos, permatomumas, fizikinės savybės, bei kiti duomenys. Diskretizavimo metu gaunama labai dideli duomenų kiekiai, pavyzdžiui diskretizuojant erdvę į 1024x1024x1024 sritis, kiekvienai sričiai saugant informaciją apie jos spalvą ir permatomumą (RGBA – 4 baitai) gaunama daugiau nei 4 GB duomenų. Tokie informacijos kiekiai palyginus su dabartinėmis kompiuterių skaičiavimo galimybėmis yra ganėtinai dideli, todėl kyla problema, kaip efektyviai saugoti ir atvaizduoti tokius duomenis.

1.3 Tyrimo tikslas ir uždaviniai

Tyrimo tikslas - ištirti vokselinės grafikos piešimo metodų spartą ir efektyvumą.

1.4 Tyrimo planas

- Analizės metodų pasirinkimas
- Informacijos rinkimas apie esamus vokselinės grafikos piešimo metodus
- Mokslinių straipsnių analizavimas siekiant įsigilinti į matematinį ir programinį algoritmų realizavimą.
- Sudaryti reikalavimus programinei įrangai, realizuojančiai vokselinės grafikos piešimo metodus.
- Suprojektuoti programinę įrangą vokselių piešimui.
- Realizuoti programinės įrangos prototipą, realizuojantį vokselinės grafikos piešimo metodus.
- Atlikti programinės įrangos testavimą
- Atlikti eksperimentus
- Apibendrinti eksperimentų rezultatus
- Parengti baigiamąjį aprašą

1.5 Analizės metodai

Analizės metu bus aptariama vokselių technologijos pasiekimai programinėse įrangose, bei apžvelgiama vokselių atvaizdavimo metodų veikimo principai. Taip pat aptariami vokselių technologijos trukumai su kuriais šiuo metu yra susiduriama.

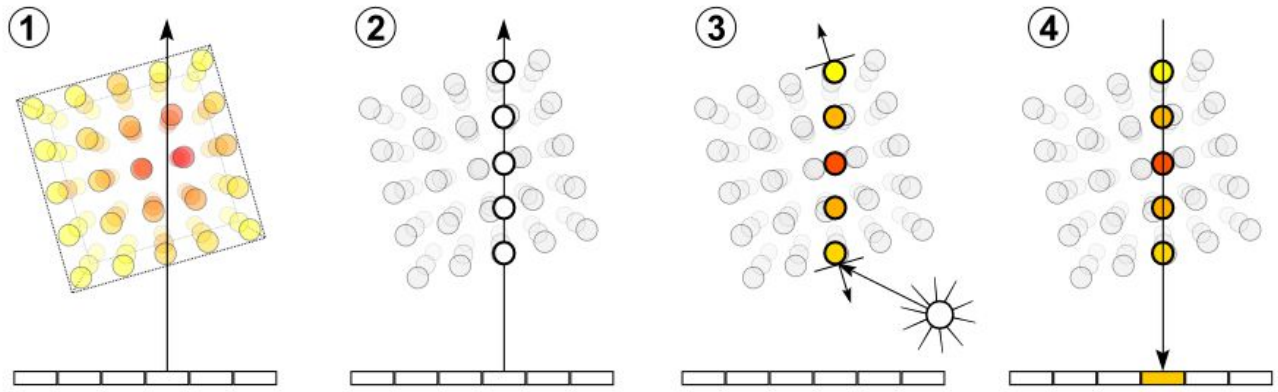
1.6 Vokselių vaizdavimo metodų analizė

Vokselių atvaizdavimo algoritmai skirstomi į dvi pagrindines grupes pagal vokselių duomenų transformavimą [1]. Pirmoji grupė, tai tiesioginis tūrinės informacijos atvaizdavimas projekcinėje plokštumoje. Šio vaizdavimo metu vaizdo reikšmės yra gaunamos manipuliuojant vokselių spalvų ir permatomumo reikšmėmis. Antroji grupė, tai netiesioginis tūrinės informacijos vaizdavimas. Šio vaizdavimo esmė – vokselių duomenys yra konvertuojami į kitą tarpinį formatą, dažniausiai tekstūras arba vokselių paviršių padengiančią geometriją.

1.6.1 Tiesioginis vokselių atvaizdavimas

1.6.1.1 Spindulių trasavimas (ray-tracing)

Šis metodas geriausiai išnaudoja vokselių duomenis, kadangi jiems nepriskiria jokios geometrinės struktūros, ir leidžia tiksliai atvaizduoti permatomumo lygius. Šio algoritmo veikimo principas (1 pav) – kiekvienam ekrano pikseliui yra trasuojamas spindulys per vokselių struktūrą(1), surenkant kertamų vokselių duomenis, ir iš jų apskaičiuojant galutinę pikselio reikšmę. Spindulių trasavimas skirstomas į du variantus, ray-casting ir ray-tracing.



1 pav. spindulių trasavimo etapai, 1-spindulio krypties sudarymas, 2-reikšmių surinkimas, 3-apšvietimo apskaičiavimas, 4-reikšmių sukomponavimas

Ray casting atveju trasuojami tik pirminiai spinduliai. Kiekvienam ekrano pikseliui apskaičiuojama spindulio kryptis ir nuo ekrano projekcijos plokštumos vienodais intervalais surinkinėjamos erdvės taškuose esančios reikšmės. Erdvės taško spalvos gaunamos interpoliavus aplink jį esančių vokselių reikšmes. Surinkus pakankamai reikšmių jog būtų pasiektas visiškas nepermatomumas, arba spinduliui išėjus už vokselių erdvės ribų, surinktos reikšmės sukomponuojamos į galutinę pikselio spalvą. Sukomponavimas vykdomas pagal formulę

$$C_f^b = \sum_{i=f}^{b} a_i C_i \prod_{j=f}^{i-1} T_j;$$

čia C_f^b reiškia bendrą visų reikšmių f,f+1...b sukomponuotą spalvą, a_j - reikšmės permatomumas [0;1], C_j - vokselio spalva, $T_j = 1 - a_j$ - vokselio reikšmės intensyvumas. Šis variantas neįvertina apšvietimo sąlygų, todėl galutinė pikselio spalva yra įtakojama tik vokseliuose saugomų spalvų. Apšvietimo efektas gali būti išgaunamas tik tada jeigu pačioje vokselių struktūroje yra saugoma statinio apšvietimo informacija. Algoritmas reikalauja daug skaičiavimo operacijų, priklausomai nuo generuojamo vaizdo rezoliucijos. Skaičiavimo operacijų skaičių galima žymiai sumažinti pritaikius keletą paprastų optimizacijų [2]. Pagrindinės naudojamos optimizacijos:

1. Priešlaikinis spindulio sustabdymas

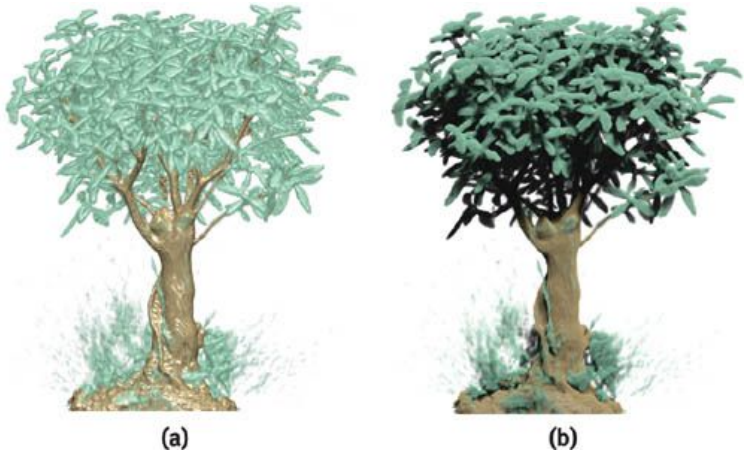
Pridedant naujas spindulio taško reikšmes, apskaičiuojamas bendras jų permatomumo koeficientas. Jeigu jis pasiekia visišką nepermatomumą stabdomas spindulio trasavimas ir vaizdo pikseliui apskaičiuojama spalva iš iki tol surinktų vokselių reikšmių.

2. Hierarchinės duomenų struktūros

Vokselių duomenis patalpinus hierarchinėje struktūroje, tokioje kaip octree medis, galima pritaikyti reikšmių surinkimą kintamais intervalais. Kadangi medis yra sudalinamas iki vienalyčių erdvės sričių, spinduliui įėjus į bet kurią sritį, galima apskaičiuoti tašką, kuriame spindulys turėtų išeiti iš tos srities. Tokiu būdų surenkamos vokselių reikšmės tik jų pasikeitimo vietose, tai leidžia efektyviai peršokti per tuščias arba vienalytes erdves, iš jų paimant tik vieną reikšmę.

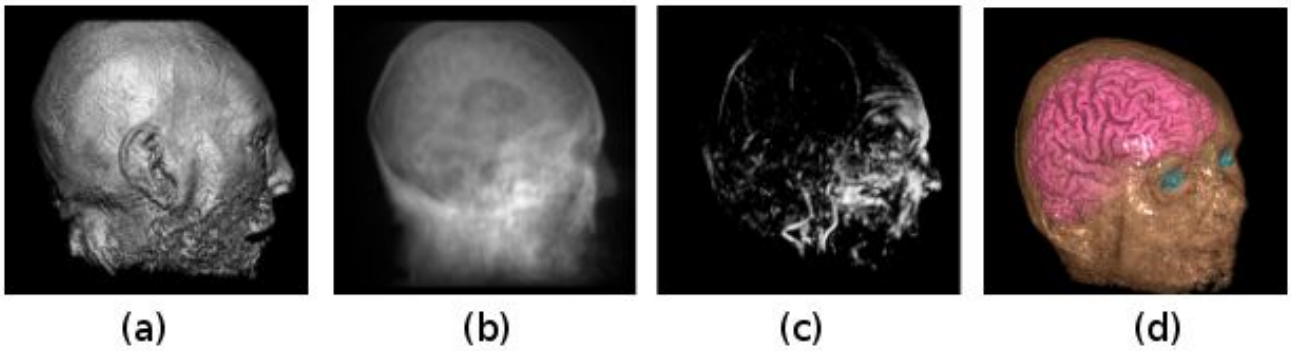
Kadangi šio algoritmo metu vykdomos labai paprastos vektorinės ir aritmetinės operacijos, jo efektyvumą galima žymiai padidinti, perkeliant skaičiavimų vykdymą į vaizdo plokštės procesorius.

Ray-tracing yra ray-casting metodo plėtinys, papildomai trasuojantis antrinius spindulius apšvietimo apskaičiavimui. Kiekvienos reikšmės surinkimo metu yra trasuojami antriniai spinduliai į visus apšvietimo šaltinius. Tai leidžia sukurti šešėliavimo efektus (2) ir įvertinti objekto spalvą keičiančias šviesos savybes, tokias kaip absorbciją, sklaidą ir spinduliavimą. Šis vaizdavimo metodas sukuria realistiškiau ir tiksliau atrodančius vaizdus, tačiau padidėjęs skaičiavimų kiekis gali prailginti vieno vaizdo generavimo trukmę net iki kelių valandų trukmės. Taip pat šis algoritmo variantas tampa nebeefektyvus vykdymui su grafine plokšte, kadangi antrinių spindulių trasavimas sukuria daug sąlyginių kodo išsišakojimų. Kiekvienas sąlyginis išsišakojimas vaizdo plokštės efektyvumą mažina geometrine progresija., taigi šį metodo variantą daug efektyviau vykdyti centriniu procesoriumi.



2 Pav. Vokselių vaizdavimas be ir su šešėliavimo efektais

Priklausomai nuo pasirinktos reikšmių sukomponavimo funkcijos spindulių trasavimu galima išgauti įvairaus stiliaus atvaizdus (3 pav). Dažniausiai taikomi keturi komponavimo tipai: (a) pirmasis, (b) vidutinis, (c) maksimalaus intensyvumo, (d)kaupiantysis. Pirmasis komponavimas vaizdui perteikia tik pačio pirmo vokselio reikšmę, to rezultatas vokselių struktūrą dengiančio paviršiaus atvaizdavimas. Vidutinis komponavimas apskaičiuoja spindulio surinktų reikšmių vidurkį, tokiu būdų sukuriamas rentgeno nuotrauką primenantis vaizdas. Maksimalaus intensyvumo komponavimas išrenka iš visų spindulio surinktų reikšmių pačią didžiausią, tai gautame vaizde leidžia lengvai identifikuoti vidines objektų struktūras. Kaupiantysis komponavimas sujungia visas spindulio surinktas reikšmes, pagal jų spalvas ir permatomumo koeficientus, tokiu būdų leidžiama atvaizduoti permatomus sluoksnius ir po jais esančias struktūras. Pirmasis ir vidutinis komponavimo tipai yra specialūs kaupiamojo komponavimo atvejai.



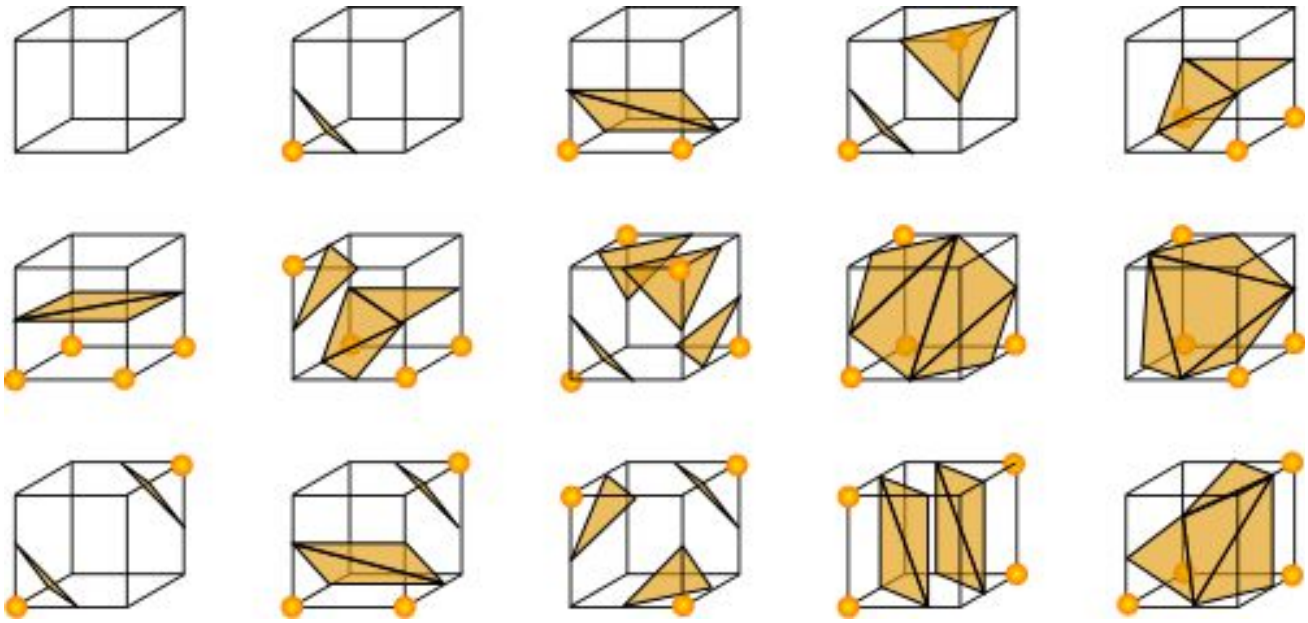
3 pav. Komponavimo tipų rezultatai.

1.6.2 Netiesioginis vokselių atvaizdavimas

1.6.3 Dengiančiojo paviršiaus išskyrimas (Isosurfacing)

Iš vokselių tūrinės informacijos išskiriamas objektą dengiantis paviršius. Ši operacija atliekama lyginant greta esančių vokselių būsenas. Jeigu vokseliuose saugoma tik binarinės reikšmės, dengiantis paviršius sukuriamas sudarant plokštumą tarp skirtingas reikšmes turinčių vokselių. Jeigu vokseliai turi kintamo intensyvumo reikšmes, prieš pradedant sudaryti dengiantį paviršių, vokselių reikšmėms pritaikoma ribojimo kriterijai (thresholding), kurie apibrėžia ar vokselis yra užpildytas. Dengiantiesiems paviršiams sudaryti dažniausiai taikomas Marching Cubes algoritmas.

1.6.3.1 Marching Cubes

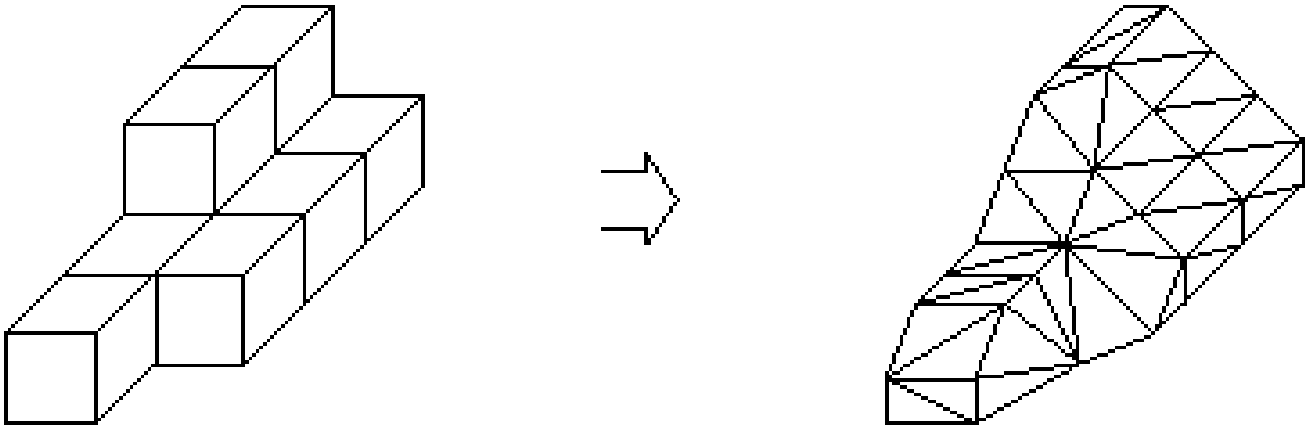


4 pav. unikalios kubo viršūnių konfigūracijos

Šio algoritmo esmė – dengiantį paviršių galima sudaryti pagal kubo viršūnių užimtumo konfigūraciją [3]. Algoritmas veikia nagrinėdamas vokselių duomenis grupėmis po 8 vokselius sudarančius kubą. Kiekvieno vokselio užpildymo reikšmė yra priskiriama kubo viršūnėms, tada atsižvelgiant pagal užpildytų viršūnių konfigūraciją kubo erdvėje sudaromos dengiančiojo paviršiaus plokštumos. Kubas turi 256 galimas viršūnių užpildymo kombinacijas, tačiau atmetus pasukimo ir simetrijos variantus lieka 15 unikalių kombinacijų (4 pav). Galutinis dengiantis paviršius gaunamas surenkant ir sujungiant visas sudarytų kubų erdvėse esančias plokštumas. Šis sprendimas turi keletą ryškių trūkumų. Kadangi dengiančiosios plokštumos sudaromos lokaliai, kartais gali kilti dviprasmybių, kurią kubo konfigūraciją pasirinkti, todėl tarpusavyje apjungus visas sudarytas plokštumas dengiančiajame paviršiuje gali atsirasti skylių. Kadangi šis algoritmas yra vokselių paviršiaus aproksimacija naudojant primityvias geometrines formas, jo rezultate sumažėja duomenų tikslumas ir prarandamos smulkios detalės. Taip pat prarandama objekto vidaus informacija, nebent pradiniai vokselių duomenys yra saugojami kartu su dengiančiuoju paviršiumi. Pagrindinė šio metodo problema yra didelis atminties kiekis reikalingas saugoti dengiančiojo paviršiaus informacijai, kadangi šis algoritmas sugeneruoja daugybę mažų trikampių, kurie sudaro perteklinę informaciją. Kita vertus šis algoritmas pasižymi itin didele sparta, kadangi visi sudėtingi plokštumų generavimo skaičiavimai yra atlikti iš anksto sudarant kubo viršūnių konfigūracijų paieškos lentelę. Pačio algoritmo veikimo metu su kubais atliekamos tik labai paprastos operacijos. Kadangi kiekviena kubo konfigūracija yra nagrinėjama nepriklausomai nuo kitų greta esančių konfigūracijų, šis algoritmas puikiai tinka skaičiavimų išlygiagretinimui.

Išskirtas dengiantysis paviršius toliau yra piešiamas naudojant tradicinius kompiuterinės

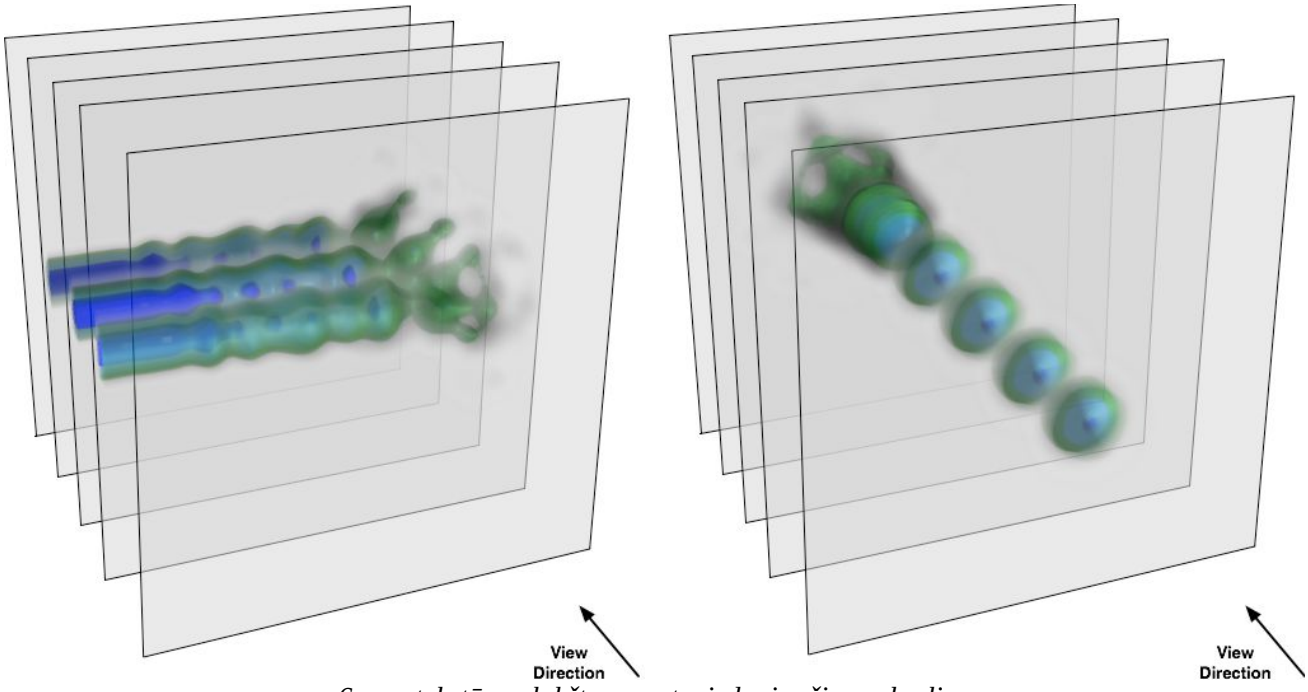
grafikos metodus (5 pav).



5 pav. Marching cubes rezultatas

1.6.3.2 Tekstūrinis vokselių piešimas

Vokseliniai duomenys gali būti atvaizduojami kaip tekstūros ant koordinačių ašims lygiagrečių plokštumų [4]. Šiam metodui priklausomai nuo techninės įrangos galimybių yra du realizacijos būdai. Pirmasis būdas yra kiekvienai koordinačių ašiai sukurti po rinkinį dvimačių tekstūrų, kurios atitiktų vokselinių duomenų skerspjūvius statmenus tai ašiai. Šios tekstūros yra uždedamos ant poligonų plokštumų išdėstytų taip, kad jos atitiktų atliktus vokselinių duomenų skerspjūvius. Antrasis būdas, jeigu techninė įranga palaiko trimates tekstūras, ši metodą galima realizuoti su viena trimate tekstūra, į ją sukeliant vokselių duomenis. Atvaizdavimo metu erdvėje sukuriama rinkinys poligonų plokštumų, kurių pozicijos atitinka vokselių duomenų skerspjūvius. Šie poligonai orientuojami lygiagrečiai ašims, kurios yra labiausiai lygiagrečios vaizdo projekcijos plokštumai (6 pav). Kiekvienam poligonui priskiriama tekstūra iš atitinkamo tekstūrų rinkinio. Naudojant trimatę tekstūra, plokštumos Poligonai piešiami eilės tvarka pradedant nuo toliausiai nuo kameros nutolusių plokštumų, atliekant spalvų komponavimą pagal permatomumo kanalą.



6 pav. tekstūrų plokštumos atvaizduojančios vokselius

Šis metodas yra itin spartus, kadangi grafiniai procesoriai turi gerai išvystytą tekstūrų apdorojimą. Taip pat tekstūrų naudojimas suteikia vokselių duomenims papildomas filtravimo ir

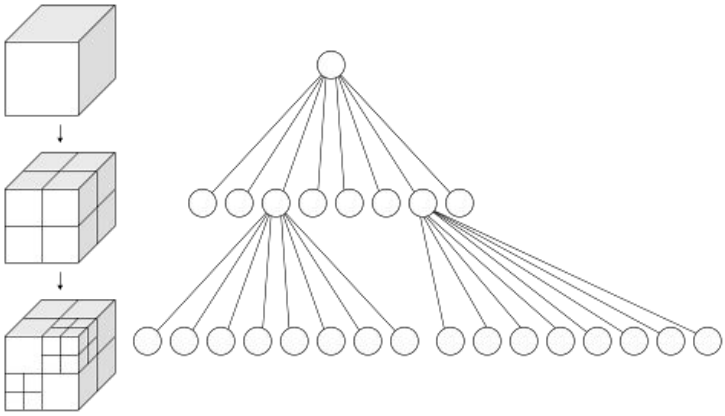
interpoliavimo galimybes be jokių papildomų reikalavimų skaičiavimams. Tačiau šis metodas tinkamas tik nedidelėms vokselių imtims vaizduoti, kadangi grafinių procesorių atminties talpa yra labiau ribota nei pagrindinė kompiuterio darbinė atmintis. Taip pat šis metodas neleidžia pritaikyti dinaminių apšvietimo efektų.

1.6.4 Vokselių duomenų struktūros

1.6.4.1 Masyvai

Pati paprasčiausia ir tiesioginė forma saugoti vokselių informacijai. Sukuriamas trimatis masyvas, kurio elementai yra vokselį apibūdinanti duomenų struktūra. Naudojant šį būdą nereikia saugoti vokselių pozicijos informacijos, kadangi ją tiesiogiai atitinka masyvo indeksai, kuriais pasiekiami vokselių duomenys. Masyvų naudojimas garantuoja greitą O(1) duomenų pasiekimo laiką pagal pozicijos indeksus. Tačiau tokia struktūra reikalauja x*y*z*v atminties (x,y,z – vokselių erdvės rezoliucijos, v - vokselyje laikomas informacijos kiekis). Kadangi didžioji dalis vokselinės erdvės būna tuščia, toks duomenų laikymo formatas yra labai neefektyvus, iššvaistoma dideli atminties kiekiai.

1.6.4.2 Octree medžiai



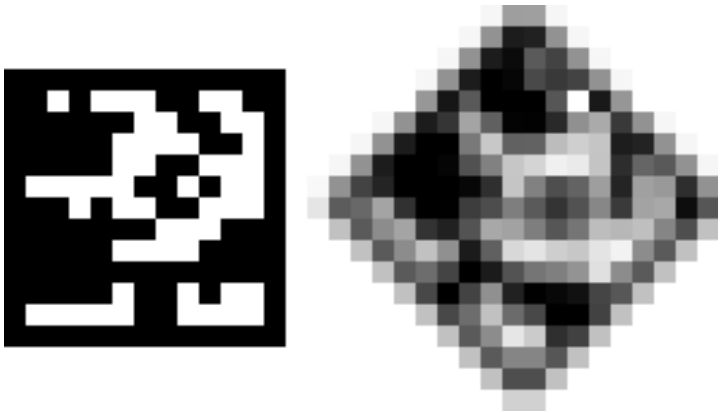
7 pav. Octree struktūra

Octree tai hierarchinis erdvės sudalinimo medis, kuriame kiekvienas medžio mazgas turi aštuonis vaikinius mazgus [5]. Octree medžio schema pavaizduota 7 pav. Kadangi Kiekvieno vaikinio mazgo duomenys yra nepriklausomi nuo kitų vaikinių mazgų, kiekvienas iš jų gali pilnai funkcionuoti kaip atskiras octree medis. Tokia duomenų atskirtis leidžia paversti vieno octree medžio konstravimą į lygiagretų aštuonių smulkesnių octree medžių sudarymą. Dėl šios priežasties taip pat norint dirbti su octree medžio duomenimis nebūtina laikyti viso medžio duomenų darbinėje atmintyje, tereikia įkelti tik duomenis esančius naudojamame mazge Šis medis gali būti pritaikytas vokselių duomenų saugojimui, darant prielaidą, jog kiekvienas medžio lapas atitinka vieną vokselį. Kadangi didžioji vokselinių duomenų dalis būna tuščia erdvė arba savyje turi dideles vientisos erdvės dalis, jas kuo puikiausiai galima apjungti į vieną octree mazgą. Tokiu būdų yra sumažinamas perteklinių duomenų kiekis. Esant ypatingai netolygiems duomenims, octree gali pareikalauti daugiau atminties negu sudaro pradiniai duomenys, kadangi neįmanoma atlikti apjungimų ir kartu reikia saugoti medžio hierarchijos duomenis. To galima išvengti jeigu leidžiamas duomenų kokybės praradimas, kadangi octree savaime sukuria mažesnio detalumo duomenų aproksimacijas.

1.7 Vokselių technologijos trūkumai

Pagrindinis vokselių technologijos trūkumas yra itin dideli informacijos kiekiai, reikalingi norint pakankamai detaliai aprašyti objektus. Pasitelkus hierarchines struktūras informacijos kiekį įmanoma sumažinti iki priimtino kiekio, tačiau tik statinio tipo duomenims. Norint perteikti dinaminio tipo vokselių duomenis yra du galimi būdai. Pirmasis būdas yra, kiekvieną animacijos kadrą laikyti kaip atskirą statinę duomenų struktūrą. Tokio atvaizdavimo metu kiekvieno laiko momentu algoritmams

pateikiama vis kitą statinė struktūra. Toks metodas yra visiškai nepraktiškas, kadangi ir taip daug atminties reikalaujantys vokselių duomenys yra duplikuojami daugybę kartų, taip viršijant technines atminties galimybes. Antrasis būdas yra saugoti pradinę statinę duomenų būseną, ir papildomai saugoti kiekviename kadre vykdomus duomenų pokyčius. Tokio atvaizdavimo metu pastoviai kuriamas vis naujas vokselių duomenų rinkinys, pagal konkrečiu laiko momentu aprašytus pokyčius. Šis būdas dalinai išsprendžia atminties kiekio problemą, tačiau pačio atvaizdavimo našumas sumažėja. Našumas krenta kadangi po kiekvieno duomenų modifikavimo, didžioji dalis hierarchinės struktūros turi būti sukurta iš naujo. Ši problema iškyla ir vietoj duomenų pokyčių naudojant



8 pav. pasukimo 45 laipsnių kampu transformacija

skeletines valdymo struktūras.

Kita problema susijusi su vokselių modifikavimu yra lanksčiosios transformacijos, tokios kaip pasukimas(8 pav), masteliavimas ir šlytis. Atliekant tokias transformacijas, retai kada įmanoma pritaikyti tiesioginį duomenų perkėlimą iš vieno vokselio į kitą. Kitaip nei atliekant poslinkio transformacijas, šių operacijų metu tenka pritaikyti interpoliavimo ir filtravimo funkcijas, dėl kurių duomenys gali žymiai pakisti ir tapti nebepanašūs į pradinius duomenis. Vokselių technologijos pasiekimai programinėse įrangose

1.7.1 „Atomontage”

„Atomontage“ tai realaus laiko vokselinės grafikos ir fizikinių reiškinių simuliavimo variklis [6]. Jame taikomų vokselių vaizdavimo algoritmų veikimas nėra paviešintas, tačiau užsimenama, jog jie pagrįsti kokybę prarandančiu duomenų suspaudimu ir yra spartinami pasitelkiant vaizdo plokštėje esančius procesorius. Ši technologija pasiekia įspūdingus rezultatus. Sugeneruoti vaizdai pasižymi didele raiška ir detalumu, bet svarbiausias aspektas, jog tai daroma realiu laiku ir pasiekiama net kelių šimtų kadrų per sekundę sparta. Taip pat suteikiama galimybė realiu laiku modifikuoti vokselių duomenis, pridedant naujus, arba šalinant senus vokselius, laisvai galima keisti jų vaizdinius parametrus. Tačiau šiuo metu šis variklis neturi galimybės atvaizduoti vokselius su permatomumo savybe (9 pav).

9 pav: Atomontage sugeneruotas vaizdas

1.7.2 „Unlimited Detail”

Unlimited Detail“ trimatės grafikos variklis, pagrįstas taškų debesies paieškos indeksavimo sistema. Šio variklio kūrėjai teigia, jog ši sistema gali atvaizduoti neribotą detalumo lygį realiu laiku [7]. Tai bandoma pagrįsti tuo kad jų sistema kiekvienam vaizdo pikseliui atvaizduoti iš vokselių duomenų išrenka tik vieną tašką. Kadangi variklis kiekvieno kadro metu rodo pastovų taškų kiekį, detalumą riboja tik atminties kiekis reikalingas sutalpinti visų taškų informacijai, o atvaizdavimo sparta priklauso tik nuo vaizdo rezoliucijos. Toks vokselių atvaizdavimo būdas netenka galimybės perteikti permatomumą, kadangi kiekvienam pikseliui analizuojamas tik vienas vokselis (10 pav).


10 pav. „Unlimited Detail" sugeneruotas vaizdas

Sistema naudodama tik centrinį procesorių sugeba realiu laiku sugeneruoti vaizdus iš informacijos kiekių siekiančių tūkstančius gigabaitų.

1.7.3 „Krakatoa”

„Krakatoa“ tai tūrinių dalelių simuliavimo ir vizualizavimo programinis paketas. Duomenų atvaizdavimui naudojamas ray-tracing metodas. Šis paketas orientuotas į foto realistiškų vaizdų generavimą (11 pav). Atvaizduojami vokseliai dažniausiai veikiami keleto šviesos šaltinių, tai žymiai padidina skaičiavimų trukmę. Priklausomai nuo naudojamų vokselių dydžių ir šviesos šaltinių kiekio, vaizdų generavimo trukmė gali svyruoti nuo kelių sekundžių iki valandų. O sukuriamų vokselių duomenų užimamos atminties kiekiai gali siekti iki šimtų gigabaitų [8].

11 pav. „Krakatoa“ programinė įranga

1.8 Analizės išvados

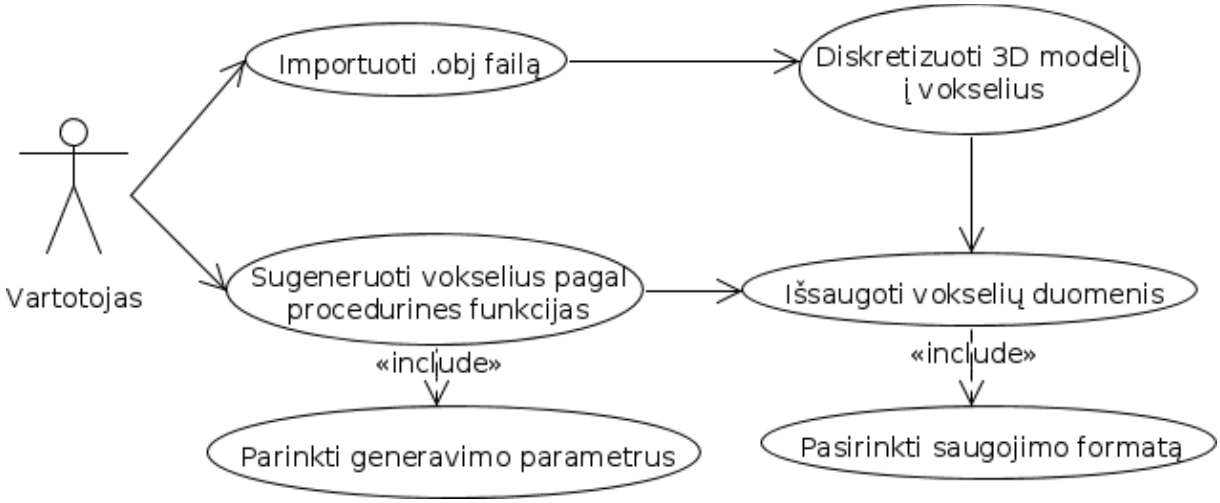
- Išanalizavus su vokseliniais duomenimis dirbančius algoritmus, nustatyta, jog jų struktūra puikiai tinka išlygiagretinimui.
- Apžvelgus šiuolaikinių kompiuterių pajėgumus, akivaizdu, jog jie yra pakankamai pajėgūs realiu laiku atvaizduoti didelius vokselių kiekius, tačiau atsisakant permatomumo savybės.
- Išanalizavus vokselių atvaizdavimo metodus pagal tikslumo kriterijų, nustatyta, jog spindulių trasavimas yra pats tiksliausias metodas vokselių atvaizdavimui.
- Išanalizavus vokselių duomenų saugojimo būdus, matoma, jog hierarchinių duomenų struktūrų panaudojimas leidžia sumažinti vokselių atminties sunaudojimą iki efektyviai panaudojamo kiekio.
- Įvertinus esamus vokselinės grafikos rezultatus matoma, jog vaizdine kokybe ji prilygsta poligoninei ir turi potencialą ją pralenkti, tačiau techninių reikalavimų atžvilgiu poligoninė grafika yra daug efektyvesnė.
- Apžvelgus vokselinių duomenų savybes pastebėta, jog šiuo metu vokseliai yra pakankamai efektyvūs tik statinio tipo duomenims.

2 Reikalavimų specifikacija

Atlikus vokselių duomenų struktūrų ir atvaizdavimo metodų analizę, sudaryti reikalavimai sistemai, kuria bus tiriamas atvaizdavimo metodų efektyvumas. Sistema padalinta į dvi dalis: duomenų paruošimo(12 pav.) ir duomenų atvaizdavimo(13 pav.) posistemes.

2.1 Funkciniai reikalavimai

Sistemos funkciniai reikalavimai pateikiami UML panaudos atvejų diagramomis, kuriose pavaizduoti pagrindiniai sistemos funkcionalumai prieinami jos vartotojui.



12 Pav. Duomenų paruošimo panaudos atvejai

Lentelė 2.1: Panaudos atvejo „importuoti .obj failą" aprašas

Panaudojimo atvejis	importuoti .obj failą
Aprašymas	į sistemą įkeliamas 3D geometrijos modelis ir atvaizduojamas vartotojo sąsajoje
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa, .obj failas turintis geometrijos duomenis
Rezultatai	Geometrijos duomenys iš failo įkeliami į sistemą

Lentelė 2.2: Panaudos atvejo „diskretizuoti 3D modeli į vokselius" aprašas

Panaudojimo atvejis	Diskretizuoti 3D modelį į vokselius
Aprašymas	Atliekamas 3D geometrijos plokštumų skenavimas, kurio metu sukuriama geometrijos atitikmuo vokselių pavidalu
Aktoriai	vartotojas
Reikalaujamos sąlygos	Į sistemą įkelta 3D geometrijos duomenys
Rezultatai	Vokselių duomenų rinkinys atitinkantis pateiktus geometrijos duomenis

Lentelė 2.3: Panaudos atvejo „sugeneruoti vokselius pagal procedūrines funkcijas" aprašas

Panaudojimo atvejis	Sugeneruoti vokselius pagal procedūrines funkcijas

Aprašymas	Pagal vartotojo nustatytas matematines signalų funkcijas sugeneruojami vokseliniai duomenys
Aktoriai	vartotojas
Reikalaujamos sąlygos	Paleista programa
Rezultatai	Vokselinių duomenų rinkinys

Lentelė 2.4: Panaudos atvejo „parinkti generavimo parametrus" aprašas

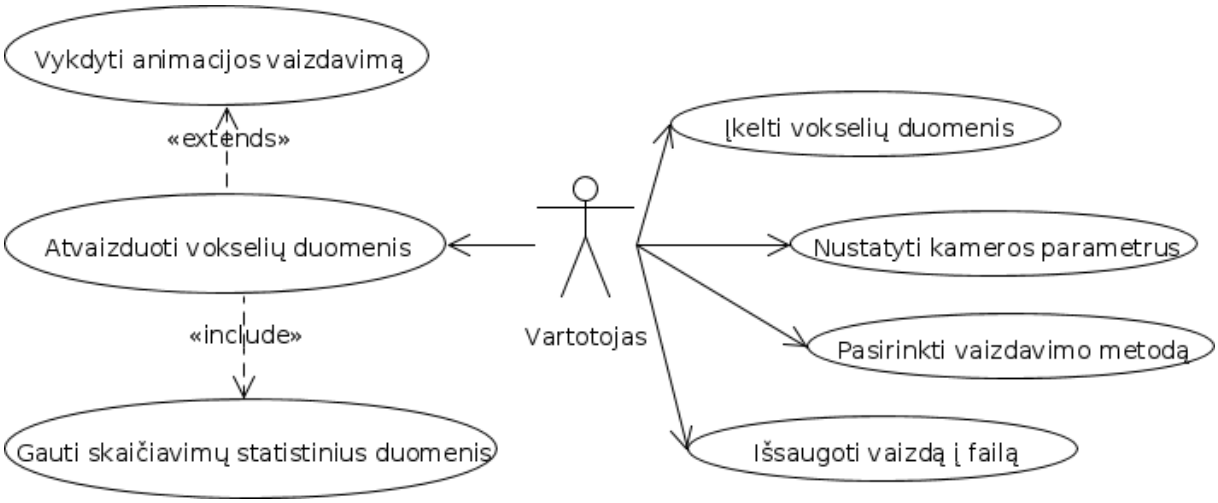
Panaudojimo atvejis	Parinkti generavimo parametrus
Aprašymas	Parenkamos vokselių generavimui naudojamos funkcijos ir jų parametrai
Aktoriai	vartotojas
Reikalaujamos sąlygos	Paleista programa
Rezultatai	Funkcijų rinkinys signalų generavimui

Lentelė 2.5: Panaudos atvejo „išsaugoti vokselių duomenis" aprašas

Panaudojimo atvejis	Išsaugoti vokselių duomenis
Aprašymas	Sistemoje esantys vokselių duomenys išsaugojami į kietąjį diską
Aktoriai	vartotojas
Reikalaujamos sąlygos	Sugeneruoti vokselių duomenys
Rezultatai	Kietajame diske sukurtas failas su vokselių duomenimis

Lentelė 2.6: Panaudos atvejo „pasirinkti saugojimo formatą" aprašas

Panaudojimo atvejis	Pasirinkti saugojimo formatą
Aprašymas	Parenkamas vokselių duomenų išsaugojimo formatas, gryni nesuspausti duomenys arba hierarchinė struktūra
Aktoriai	vartotojas
Reikalaujamos sąlygos	Sugeneruoti vokselių duomenys
Rezultatai	Pakeičiamas duomenų saugojimo formatas



13 Pav. Duomenų atvaizdavimo panaudos atvejai

Lentelė 2.7: Panaudos atvejo „įkelti vokselių duomenis" aprašas

Panaudojimo atvejis	Įkelti vokselių duomenis
Aprašymas	Iš kietajame diske esančio failo į sistemą įkeliami vokselių duomenys
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa, vokselių duomenų failas
Rezultatai	Iš failo įkelti duomenys paruošiami atvaizdavimui

Lentelė 2.8: Panaudos atvejo „nustatyti kameros parametrus" aprašas

Panaudojimo atvejis	Nustatyti kameros parametrus
Aprašymas	Parenkami atvaizdavimui naudojami kameros parametrai: pozicija, stebėjimo kryptis, matymo lauko kampas, vaizdo rezoliucija
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa
Rezultatai	Sukonfigūruojami kameros parametrai

Lentelė 2.9: Panaudos atvejo „pasirinkti vaizdavimo metodą" aprašas

Panaudojimo atvejis	Pasirinkti vaizdavimo metodą
Aprašymas	Iš duoto vaizdavimo metodų sąrašo pasirenkamas pageidaujamas metodas. Metodo reikalaujamiems parametrams priskiriamos norimos reikšmės
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa
Rezultatai	Sistema sukonfigūruojama darbui su pasirinktu metodu

Lentelė 2.10: Panaudos atvejo „išsaugoti vaizdą į failą" aprašas

Panaudojimo atvejis	Išsaugoti vaizdą į failą
Aprašymas	Sistemos sugeneruotas vaizdas išsaugojamas į kietąjį diską
Aktoriai	Vartotojas
Reikalaujamos sąlygos	paleista programa
Rezultatai	Išsaugotas sugeneruotas vaizdas

Lentelė 2.11: Panaudos atvejo „atvaizduoti vokselių duomenis" aprašas

Panaudojimo atvejis	Atvaizduoti vokselių duomenis
Aprašymas	Pradedamas vykdyti pasirinktas vokselių duomenų atvaizdavimo metodas. Metodo rezultatai atvaizduojami vartotojui
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa

Rezultatai	Paveikslėlis, metodo statistiniai duomenys

Lentelė 2.12: Panaudos atvejo „vykdyti animacijos vaizdavimą" aprašas

Panaudojimo atvejis	Vykdyti animacijos vaizdavimą
Aprašymas	Ciklišk ai vykdomas vokselių duomenų atvaizdavimas, po kiekvieno atvaizdavimo modifikuojami vokselių duomenys
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa
Rezultatai	Paveikslėlių seka, metodo statistiniai duomenys

Lentelė 2.13: Panaudos atvejo „gauti skaičiavimų statistinius duomenis" aprašas

Panaudojimo atvejis	Gauti skaičiavimų statistinius duomenis
Aprašymas	Surenkami vaizdavimo metodo statistiniai duomenys: trukmė, naudojamos atminties kiekis, vokselių skaičius,
Aktoriai	vartotojas
Reikalaujamos sąlygos	paleista programa, sugeneruotas vaizdas
Rezultatai	Metodo vykdymo statistiniai duomenys

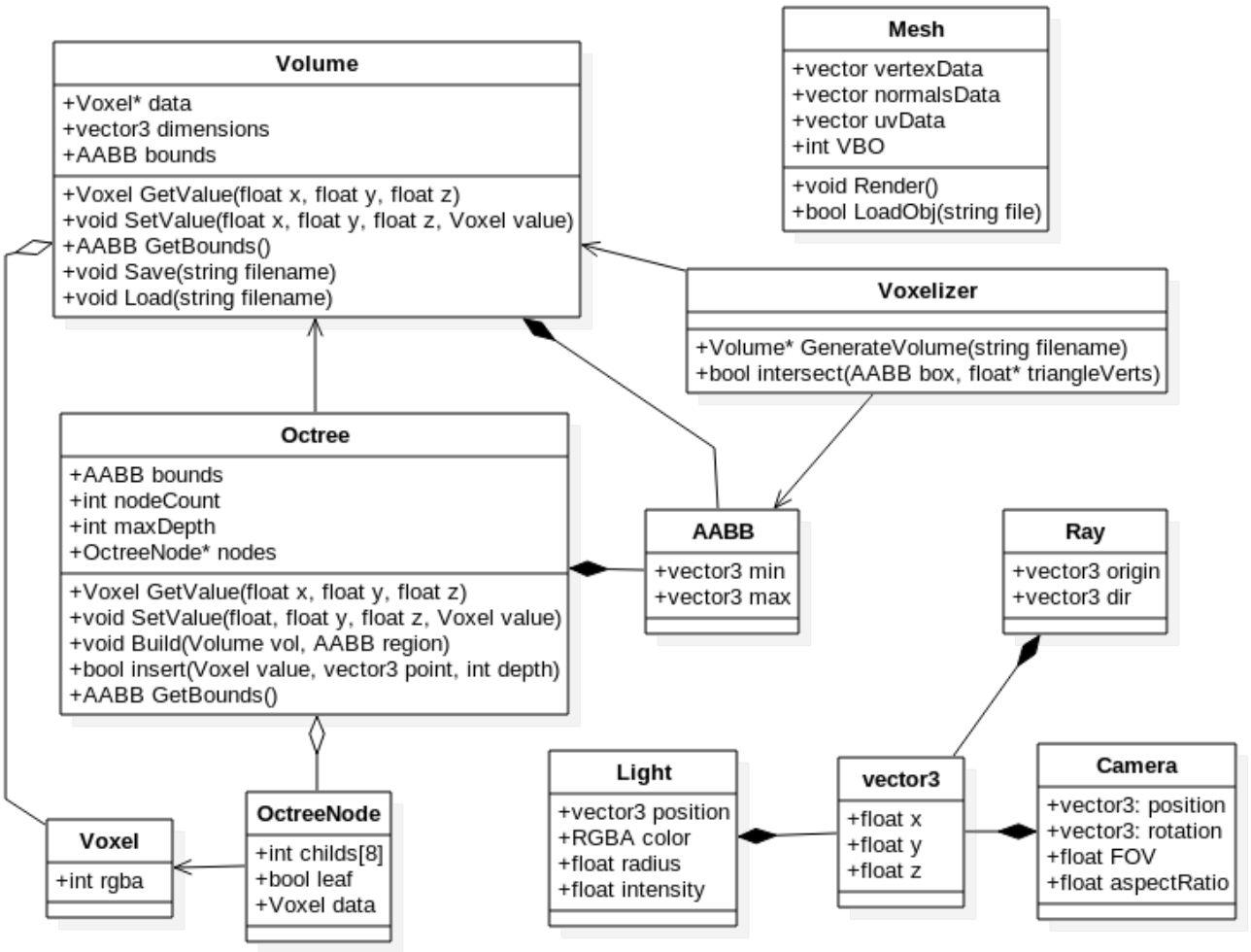
2.2 Nefunkciniai reikalavimai

- Sistema turi veikti Linux ir Windows operacinėse sistemose.
- Sistemos darbo metu turi būti rodoma informacija apie proceso eigą ir būseną

3 Sistemos realizacija

Įvertinus sistemos reikalavimus, sudarytas sistemos projektas. Sistema padalinta į du atskirus modulius: objektų diskretizavimo modulį ir spindulių trasavimo modulį. Atsižvelgiant į ketinamas panaudoti technologijas diskretizavimo moduliui pasirinktas objektinis programavimo stilius, o vokselių atvaizdavimo moduliui pasirinkta procedūrinis programavimo stilius. Visoje sistemoje naudojamų objektų ryšiai ir funkcionalumas pavaizduoti 14 pav.

3.1 Sistemos architektūra



14 pav. Klasių diagrama

3.2 Klasė Volume

Ši klasė skirta saugoti grynus vokselių duomenis. Visi duomenys saugomi vientisame atminties regione, kuriems gauti galima daryti užklausą pagal taško koordinates erdvėje. Kiekvieno vokselio dydis erdvėje yra apskaičiuojamas pagal tūrinės erdvės ribinių taškų koordinates. Vokselių duomenis galima išsaugoti į failą binariniu pavidalu ir vėliau juos vėl užkrauti. Klasės sandara pateikta Lentelė 3.1

Lentelė 3.1 Volume sandara

data	Vientisas nuoseklus atminties blokas talpinantis vokselių duomenis
dimensions	Erdvės dimensijų rezoliucijos
bounds	Koordinačių ašims lygiuotas gaubiantysis tūris, erdvės išmatavimai

GetValue	Gražina vokselio reikšmę esančia nurodytose koordinatėse
SetValue	Nustato vokselio reikšmę nurodytame taške
Save	Išsaugo vokselių duomenis į failą
Load	Užkrauna vokselių duomenis iš failo

3.3 Klasė Octree

Ši klasė atsakinga už vokselių duomenų konvertavimą į hierarchinę struktūrą. Ji atlieka atminties resursų tvarkymą ir manipuliavimą duomenimis struktūroje. Hierarchinėje struktūroje saugoma tik medžio mazgų ryšiai ir juose esančios reikšmės. Kiekvieno mazgo erdvės dydis yra apskaičiuojamas pagal pagal pagrindinio mazgo erdvės ribinių taškų koordinates. Ši klasė leidžia gauti arba nustatyti medžio mazgo reikšmę pagal koordinačių taško užklausą. Taip pat suteikiama galimybė įterpti vokselių duomenis norimam medžio taške, į pasirinktą medžio gylį. Octree sandara pateikta Lentelė 3.2.

Lentelė 3.2 Octree sandara

bounds	Koordinačių ašims lygiuotas gaubiantysis tūris, erdvės išmatavimai
nodeCount	Medyje esančių mazgų kiekis
maxDepth	Maksimalus gylis iki kurio leisti dalinti medį
nodes	Medžio mazgų sąrašas patalpintas nuoseklioje atmintyje
GetValue	Gražina mazgo duomenis esančius žemiausiame medžio lygyje, gaubiančiame duotąjį tašką.
SetValue	Nustato medžio mazgo reikšmę duotame taške, pakeičiamas tik esantis medžio mazgas
Build	Iš duotų vokselių duomenų sugeneruojama hierarchinė struktūra, su pasirinktu maksimaliu gyliu
insert	Įterpia vokselio duomenis į medį pasirinktame taške, norimame gylyje. Jeigu medis nėra pasiekęs norimo gylio, jis yra dalinamas, kol sukuriamas mazgas norimame taške ir gylyje.

3.4 Klasė OctreeNode

Sudėtinis Octree klasės elementas, saugantis vokselio duomenis ir turintis nuorodas į tolimesnius medžio mazgus, atminties masyve. Gali būti naudojamas be Octree klasės, bet tik skaitymo operacijoms.

Lentelė 3.3 OctreeNode sandara

childs	Vaikinių mazgų indeksai duomenų masyve
leaf	Žymė rodanti ar mazgas turi vaikų
data	Vokselio duomenys erdvės tūryje

3.5 Klasė Voxel

Pagrindinė klasė nusakanti naudojamų duomenų formatą ir dydį. Šiuo atveju sistema orientuota tik į grafinį atvaizdavimą, todėl vokselių duomenys apsiriboja tik spalvine ir permatomumo reikšme. Spalva ir permatomumas nurodomas vienu 32 bitu kintamuoju, kiekvienai spalvai skiriant po 8 bitus.

Kiekvienos spalvos informacija gali būti išskirta taikant dvejetaines kaukes. Spalvos yra pateikiamos RGBA formatu, kur raudona spalva yra vyriausiose skiltyse, o permatomumas žemiausiose.

3.6 Klasė Camera

Klasė skirta suformuoti spindulių trasavimo kryptis ir apribojimus. Pagal duotus matymo kampo ir vaizdo kraštinių santykius suformuojama spindulių masyvą. Spindulių kiekis sukuriamas priklausomai nuo norimos paveikslėlio rezoliucijos. Klasės kintamieji pateikti Lentelė 3.4.

Lentelė 3.4 Camera sandara

position	Kameros pozicija erdvėje
rotation	Kameros orientacija ašių atžvilgiu
FOV	Horizontalus kameros matymo laukas laipsniais
aspectRatio	Matomo vaizdo pločio ir aukščio santykis

3.7 Klasė Light

Klasė skirta pritaikyti apšvietimo efektams. Šiuo atveju tai yra paprasčiausias taškinis šviesos šaltinis, turintis savo poziciją, spalvą, intensyvumą, bei įtakojamos sferinės erdvės spindulį.

3.8 Klasė Voxelizer

Ši klasė tai procedūrų rinkinys skirtas konvertuoti geometrines figūras į diskretizuotos erdvės taškus. Savyje turi procedūras reikalingas aptikti trikampių plokštumų persidengimus su koordinačių ašims lygiuotais stačiakampiais.

3.9 Klasė Mesh

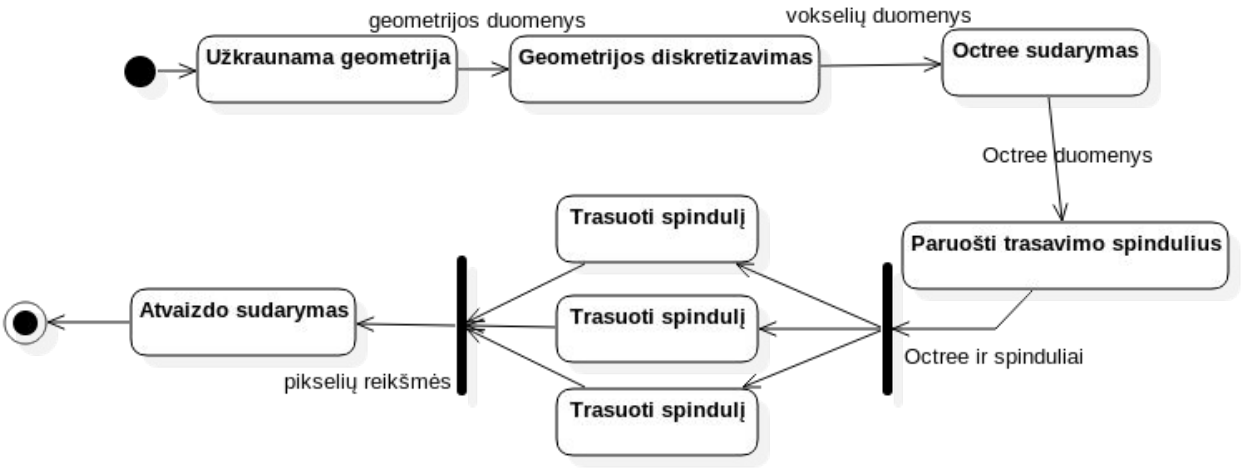
Ši klasė skirta saugoti ir apdoroti trimačių geometrinių objektų duomenis. Joje saugoma geometrinės figūros erdvinių taškų koordinatės, kiekvieno taško normalė ir trikampius sudarančių taškų sąrašas. Šie duomenys yra užkraunami iš tekstinio .OBJ formato failo. Taip pat klasė suteikia galimybę atvaizduoti geometrines figūras naudojantis OpenGL aplinka.

Lentelė 3.5 Mesh sandara

vertexData	Trikampių plokštumų taškų koordinatės
normalsData	Plokštumų normalės kiekviename kampiniame taške
uvData	Plokštumų taškų projekcinės koordinatės tekstūros erdvėje
VBO	Duomenų masyvo indeksas OpenGL sistemoje

3.10 Sistemos būsenos

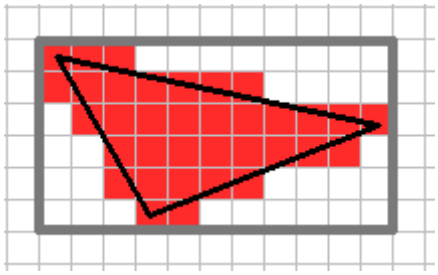
Sistemą sudaro vienas pagrindinis duomenų srautas, prasidedantis geometrijos užkrovimu ir baigiantis paveikslėlio atvaizdavimu. Tipinė programos darbo seka ir būsenos pavaizduota 15 pav. Iš failo užkraunami objekto geometrijos duomenys, jie perduodami erdvės diskretizavimo funkcijai, kurioje pagal pasirinkus diskretizavimo erdvės matmenis ir rezoliucija sugeneruojama vokselių duomenys. Vokselių duomenys perduodami Octree klasei, kuri juos naudodama sugeneruoja jų hierarchinę tūrių sistemą. Octree perduodamas spindulių trasavimo funkcijai, ši funkcija turi būti vykdoma kiekvienam ekrano pikseliui, todėl vienu metu galima lygiagrečiai vykdyti keletą spindulių trasavimų. Laukiama kol bus užbaigta visi spinduliai, tada sinchronizuojami iš jų gauti duomenys ir perkeliami į bendrą paveikslą. Paveikslo duomenys pateikiami vartotojui grafinėje sąsajoje arba išsaugomi į failą.



15 pav. Sistemos būsenų diagrama

3.11 Objektų diskretizavimas

Trimačių objektų geometrijos diskretizavimo metu sukuriamas vokselių laukas atitinkantis objekto paviršių. Procedūra atliekama užpildant kiekvienos objekto plokštumos trikampio kertamus vokselius. Norint sumažinti reikalingų palyginimų kiekį, aplink trikampį sudaroma jį gaubiantis stačiakampis, pagal kurį iš vokselių erdvės išskiriama tik nedidelis kiekis vokselių, dvimatis atvejis pavaizduotas 16 pav. Kiekvienas iš šių vokselių yra patikrinamas ar kertasi su trikampiu. Aptikus susikirtimą, vokselis pažymimas kaip užpildytas. Idealiu atveju kiekvienas trikampis turėtų būti vieno vokselio ribose. Šis metodas užtikrina, jog nebus praleista smulkių detalių, tačiau to pasekoje smulkios detalės gali išaugti priklausomai nuo vokselių rezoliucijos.



16 pav. Vokselių sankirta su trikampe plokštuma

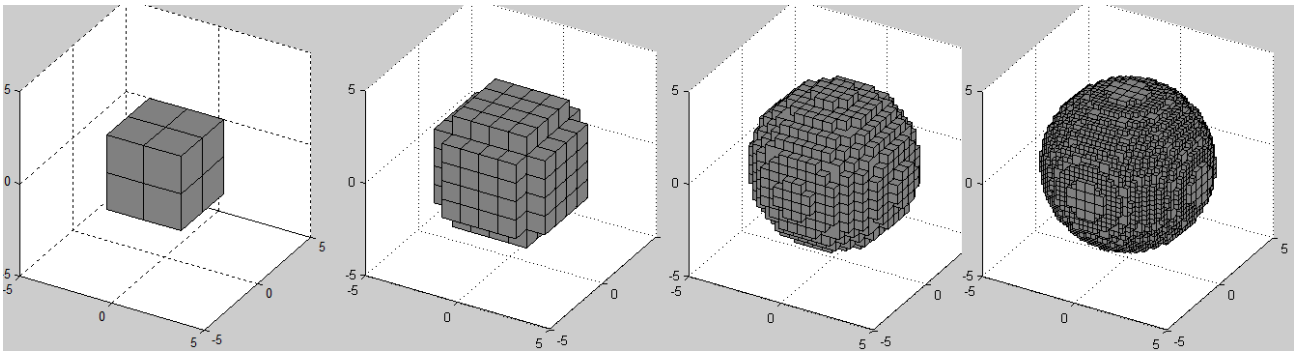
3.12 Octree

Vokselių duomenims saugoti pasirinktas MX tipo octree medis. MX tipo medis yra baigtinės erdvės medis, kurio lapų koordinačių centrai yra numanomi pagal medžio gylį ir lapo indeksą tėviniame mazge. Šiame tyrime bus saugoma tik spalvinė vokselių informacija, tai yra ar konkretus erdvės taškas yra užpildytas, todėl galutinis informacijos kiekis saugomas medžio mazge bus aštuoni indeksai į vaikinius mazgus ir erdvės užpildymo žymė. Algoritmas yra pagrįstas top-down principu, į octree įterpiami tik vokselių struktūroje užpildyti taškai. Taškai įterpiami nurodant jo konkretų medžio gylį. Toks įterpimo metodas iš anksto sukuria medžio lapus gretimiems vokselių įterpimams, išlaikant minimalų medžio mazgų kiekį. Įterpiant vokselius tikrinama ar medžio mazgo vaikai turi identiškas reikšmes, jeigu jos vienodos, vaikiniai mazgai yra pašalinami iš medžio, o jų buvusi reikšmė tiesiogiai priskiriama tėviniam mazgui.

Galutiniame medyje paliekamas minimalus informacijos kiekis galintis pilnai atkurti pradinius vokselių duomenis be jokių kokybės nuostolių. Kiekviename medžio lygyje yra laikoma skirtingo detalumo duomenys, todėl esant poreikiui dar labiau sumažinti naudojamą duomenų kiekį, galima

pašalinti giliausius medžio lygius. Toks variantas priimtinas jeigu nėra reikalavimo neprarasti duomenų kokybės.

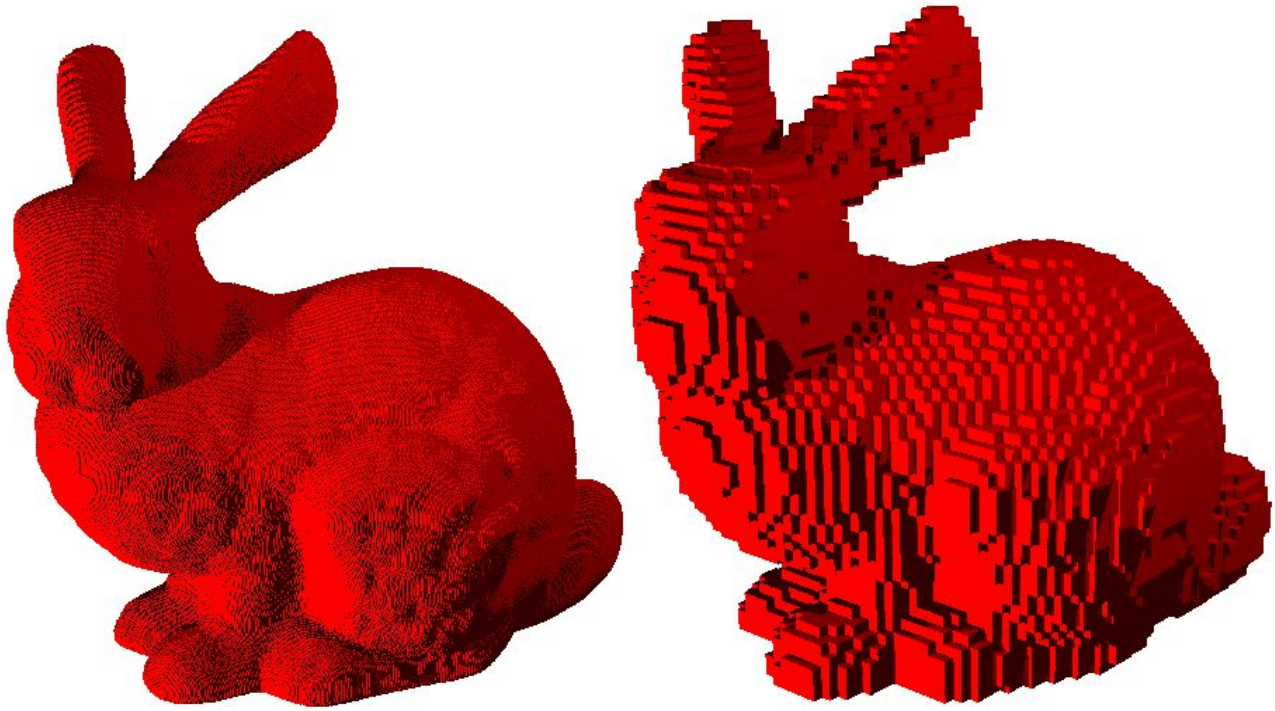
Testinių duomenų esančių octree medyje grafiniam atvaizdavimui pasirinkta poligoninė grafikos metodas. Šiuo metodu rekursiškai einama per hierarchinę struktūrą, kiekvieno žingsnio metu apskaičiuojant medžio mazgo centro koordinates. Apskaičiavus mazgo centrą tame erdvės taške nupiešiamas kubas kurio kraštinės ilgis yra lygus pradinės medžio erdvės dydžiui padalintam iš log2(medžio gylis). Atvaizdavimo metu yra piešiami tik medžio lapai. Arba norint atvaizduoti pasirinktą detalumo lygį yra piešiami visi medžio lapai esantys iki pasirinkto detalumo lygio ir medžio mazgai esantys pasirinktame detalumo lygyje. Toks atvaizdavimo metodas yra neefektyvus, kadangi bandoma piešti ir plokštumas kurių neįmanoma matyti, tačiau testavimo reikmėms jis yra priimtinas. Keleto skirtingų gylių octree atvaizdavimas matomas 17 pav.



17 pav. Sferos octree atvaizdavimas poligonais. 1, 2, 3, 4 gylio medžiai

3.13 Spindulių trasavimas

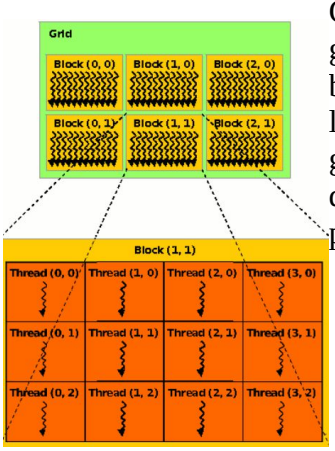
Spindulių trasavimo metu leidžiamas spindulys į hierarchinę struktūrą ir nustatoma kurį erdvės tašką reikia atvaizduoti paveikslėlio pikselyje. Šio darbo metu realizuotas paprastas spindulių trasavimo algoritmas, leidžiantis pirminius spindulius užimto tūrio aptikimui, ir leidžiantis antrinius spindulius atlikti apšvietimo apskaičiavimams. Pirminiai spinduliai suteikia tik objekto spalvos informaciją norimame taške, tačiau norint apskaičiuoti objekto apšvietimo informaciją, reikia žinoti kokiu kampu ir atstumu nuo jo yra šviesos šaltiniai. Paviršiaus apšvietimas priklauso nuo šviesos spindulių kritimo kampo, todėl norint apskaičiuoti ši kampą, reikia žinoti paviršiaus normalės vektorių. Tačiau pavieniai vokseliai yra tiesiog taškai erdvėje, kurie neturi normalės vektorių, nebent jis būtų išskirtinai nurodytas kiekvienam vokseliui kaip papildomi duomenys. Norint apskaičiuoti normalės vektorių, galima analizuoti gretimų vokselių grupę ir išvesti grupės paviršiaus kitimo gradientą. Tačiau toks variantas yra netinkamas taikant hierarchinę struktūrą, kadangi greta esantys medžio mazgai gali turėti visiškai skirtingus išmatavimus, todėl struktūroje esančių duomenų paviršius gali turėti didelę paklaidą nuo apskaičiuojamo vektoriaus. Taip pat kyla problema norint atvaizduoti pavienius vokselius, kadangi jie neturi vokselių grupės paviršiaus gradiento skaičiavimams. Šiai problemai išspręsti pasirinkta vokseliams suteikti kubo formą, tuomet priklausomai nuo trasuojamo spindulio kryties yra parenkama viena iš kubo sienų normalių. Ši normalė panaudojama apskaičiuoti apšvietimo kampui, kadangi vieną vokselį gali kirti keli gretimi spinduliai, jiems visiems yra gražinama ta pati normalė. Todėl galutiniame paveikslėlyje gali būti įžvelgiama duomenų kubinė struktūra, priklausomai nuo duomenų rezoliucijos kaip pavaizduota 18 pav.



18 pav. Tūrio atvaizdavimas naudojant kūbines formas apšvietimo skaičiavimams

3.14 CUDA posistemė

CUDA tai programavimo platforma skirta vykdyti bendro pobūdžio skaičiavimus su grafiniais procesoriais. Centriniai procesoriai dažnai turi vos keletą branduolių, todėl jų galimybės vienu metu vykdyti lygiagrečias procedūras yra apribotos branduolių skaičiumi. Lygiagrečių procedūrų kiekiui viršijus branduolių kiekį skaičiavimo sparta nebedidėja, ir netgi gali pradėti mažėti dėl laiko gaištamo keičiant vykdomas procedūras. Grafiniai procesoriai pasižymi itin dideliu skaičiavimų branduolių kiekiu, todėl suteikia žymiai didesnes galimybes algoritmų išlygiagretinimui. Tačiau kitaip nei centriniai procesoriai grafiniai procesoriai turi ribotas operacijų galimybes, todėl nevisi algoritmai gali juose efektyviai veikti. CUDA suteikia galimybę programas parašytas centriniam procesoriui vykdyti grafiniuose procesoriuose, atliekant minimalius pakeitimus. Kadangi CUDA nepilnai palaiko objektinį programavimą, spindulių trasavimo procedūros parašytos naudojant C kalbos funkcijas, tai suteikia galimybę palyginti identiškų algoritmų vykdymą tiek su CPU, tiek su GPU. Priklausomai nuo techninės įrangos galimybių grafinis procesorius gali leist iki 1024 lygiagrečiai veikiančių gijų, kurios yra vykdomos blokais, kaip pavaizduota 19 pav. Algoritmo užduotis yra padalinama į lygiagrečiai veikiančius blokus, o kiekviename bloke, dar dalinama į gijas. Spindulių trasavimui ši dalinimo sistema labai tinka, paveikslas yra dalinamas į 32x32 pikselių dydžio blokus, kiekvienam bloke esančiam pikseliui tenka viena gija.



19 pav. CUDA išlygiagretinimo schema

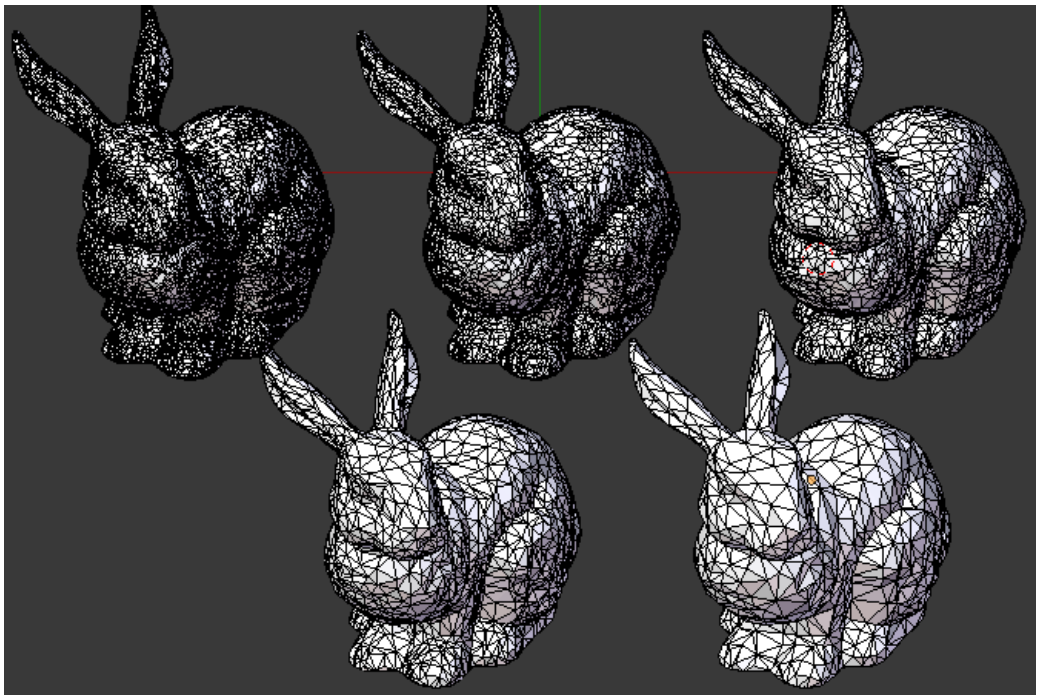
4 Eksperimentinis sistemos tyrimas

Pagrindiniai vokselių atvaizdavimo algoritmų vertinimo kriterijai yra laikas sugaištas vieno paveikslėlio sugeneravimui ir atminties kiekis sunaudojamas tūrinės informacijos saugojimui. Antriniai kriterijai yra priklausomi nuo taikomo algoritmo pobūdžio, tai gali būti laikas sugaištas konvertuojant vokselių duomenis į algoritmui priimtiną formatą arba hierarchinių erdvės dalinimo medžių sudarymo trukmė. Taip pat gali būti vertinamas algoritmų išlygiagretinimo efektyvumas multi procesorinesė skaičiavimo sistemose.

4.1 Eksperimentavimo planas

Pirmasis sistemos tyrimo etapas susideda iš pradinių įvesties duomenų paruošimo. Šio etapo metu paruošiama keletas vokselių duomenų rinkinių, panaudojant matematines generavimo funkcijas ir trimatės geometrijos diskretizavimą. Matematinėmis funkcijomis sugeneruojama elementarių geometrinių figūrų formos ir atsitiktinį išsidėstymą turintys duomenys. Tokios duomenų formos lengvai paruošiamos, tačiau retai pasitaikančios praktiniame panaudojime. Norint sukurti duomenų rinkinius kuo panašesnius į realiai pasitaikančius atvejus, panaudojama 3D objektų geometrinės formos iš Stenfordo universiteto duomenų bazės [9]. Naudojantis objektų geometrinėmis plokštumomis sukurti vokselių duomenys, atitinkantys objektų išorinį kiautą.

Antrajame etape tiriamas vokselių konvertavimas į Octree hierarchinę duomenų struktūrą. Šiame etape atliekamas Octree medžių generavimas iš objektų turinčių skirtingą kiekį geometrinių plokštumų. Tiriama medžio sudarymo laiko priklausomybė nuo norimos vokselių erdvės rezoliucijos ir objektų geometrinių plokštumų kiekio. Bandymai atliekami su 128^3, 256^3, 512^3, 768^3 rezoliucijos vokselių duomenimis. Bandymuose keičiant objekto plokštumų kiekį, norint išlaikyti medžio mazgų pasiskirstymo tendenciją, naudojama to paties objekto geometrinės formos aproksimacijos, kaip pavaizduota 20 pav.



20 pav. Geometrijos aproksimacijos, 32k, 17k, 8k, 4k, 2k trikampių

Trečiajame etape tiriamas vokselių duomenų atvaizdavimas. Atvaizdavimui naudojamas ray-casting spindulių trasavimo algoritmas. Tiriama algoritmo vykdymo laiko trukmės priklausomybė nuo generuojamo paveikslėlio rezoliucijos ir naudojamų vokselių duomenų kiekio. Vykdomos dvi algoritmo versijos, pirmoji tik pirminių spindulių trasavimas, antroji versija tęsia ir antrinių spindulių trasavimą, apšvietimo efektų apskaičiavimui. Taip pat tiriama algoritmo veikimo laiko

priklausomybė nuo išlygiagretinimo laipsnio. Išlygiagretinimo efektyvumas tiriamas atliekant skaičiavimus pasinaudojant procesoriumi turinčiu keletą branduolių. Taip pat išbandoma ir algoritmo greitaveika skaičiavimams pasitelkiant bendros paskirties grafinės plokštės procesorius naudojant CUDA technologiją.

Ketvirtasis etapas apdoroti eksperimentų metu gautus duomenis, bei nustatyti rezultatuose vyraujančias tendencijas. Pateikti gautus duomenų rezultatus grafiškai, ir jais vadovaujantis pateikti išvadas.

4.2 Techninė įranga

Sistemos tyrimams atlikti naudojamos techninės įrangos duomenys. Tyrimas atliktas stacionaraus ir nešiojamo tipo kompiuteriuose. Žemiau esančiose Lentelė 4.1 ir, pateikti sistemų parametrai. Pagrindinis įrangos skirtumas centrinio procesoriaus branduolių skaičius ir jų taktiniai dažniai. Visi kiti sistemų parametrai yra identiški, išskyrus grafinį procesorių, tačiau jis neturi įtakos centrinio procesoriaus darbui.

Lentelė 4.1. PC techninės įrangos parametrai

CPU	Intel Core i5-3570K, Quad Core, 3.40GHz, 6MB, LGA1155
GPU	GeForce GTX660 2GB GDDR5/N660 PCIE16, 1024 CUDA cores
RAM	2x4GB DDR3 1600MHz CL9
HDD	1TB, SATA/600, 64MB cache
OS	Linux Mint 17 64 bit

Lentelė 4.2. Notebook techninės įrangos parametrai

CPU	Intel® Core(TM) i7-4702MQ, 8 cores, 2.20 GHz, 6 MB
GPU	Intel® HD Graphics 4600, 2GB
RAM	2x4GB DDR3 1333MHz
HDD	1TB, SCSI, 64MB cache
OS	Linux Mint 17 64 bit

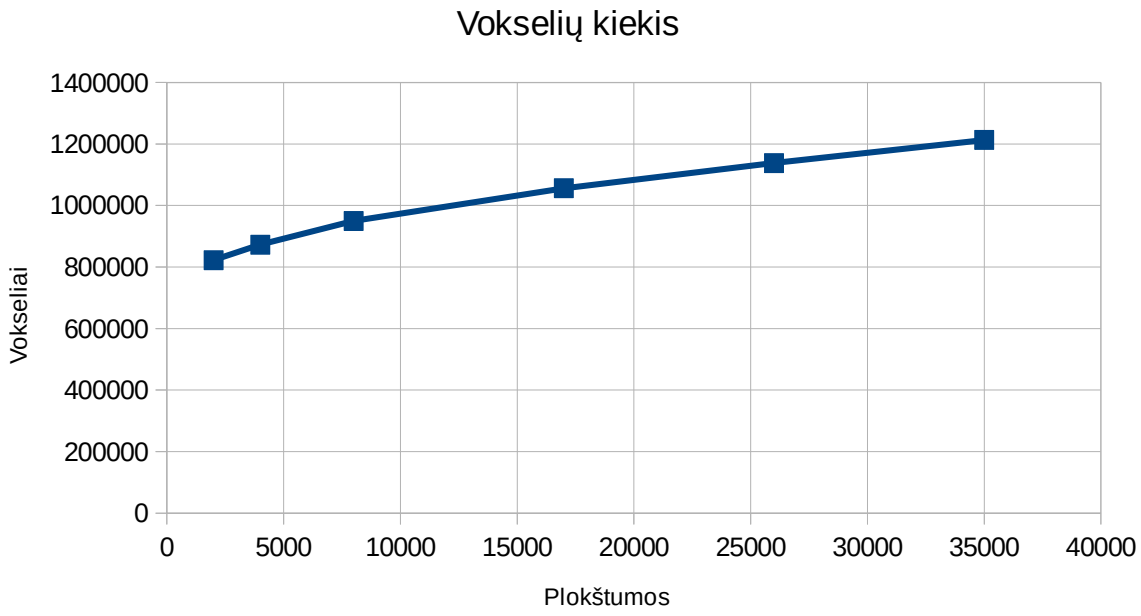
4.3 Eksperimentų rezultatai

4.3.1 Octree medžio sudarymo rezultatai

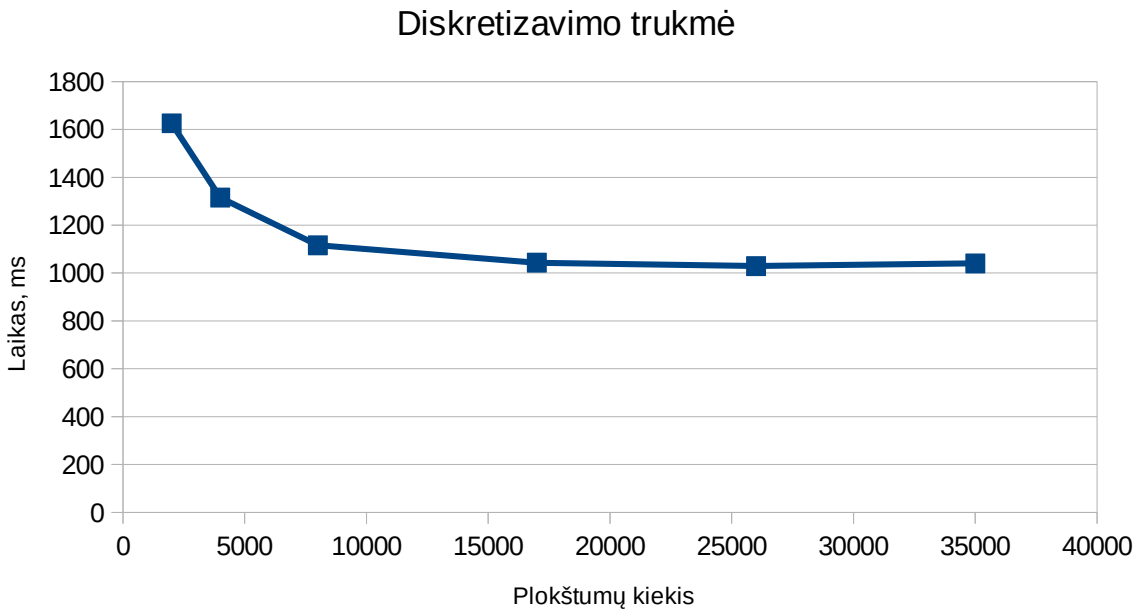
Atlikus eksperimentus su geometrijos diskretizavimu ir octree medžių sudarymu gauti rezultatai pateikti Lentelė 4.3 ir Lentelė 4.4. Atliekant skirtingo detalumo geometrijos diskretizavimą naudojant vienodos rezoliucijos tūrius nustatyta, jog vokselių kiekis reikalingas objekto detalėms perteikti, didėja logoritmiškai kaip matoma 21 pav. Kadangi didinant objekto geometrijos tankumą, geometrinės detalės tampa mažesnės nei vokselio dydis, jos yra aproksimuojamos vienu vokseliu. Bandymais pastebėta, jog geometrijos diskretizavimo laikas pavaizduotas 22 pav. yra atvirkščiai proporcingas geometrijos plokštumų kiekiui. Tačiau ši priklausomybė taip pat susieta su plokštumų ploto dydžiu erdvėje. Naudojant tos pačios geometrinės formos skirtingų detalumų lygius, objekto forma išlieka ta pati, tačiau pasikeičia vokselių kiekis tenkantis vienai plokštumai. Idealiu atveju vienam vokseliui turėtų tekti tik po vieną plokštumą, bet esant mažam geometrijos detalumui, dažnai viena plokštuma kerta daugybę vokselių. Ši priklausomybė galioja tik diskretizavimo algoritmams paremtiems plokštumos ir koordinačių ašims lygiuotų stačiakampių susikirtimu.

Lentelė 4.3 Geometrijos diskretizavimas esant 512^3 rezoliucijai

Geometrija	Turio rezoliucija	Diskretizavimo laikas, ms	Vokselių kiekis	Octree sudarymo laikas, ms
Bunny2k.obj	512^3	1626	822453	1015
Bunny4k.obj	512^3	1315	872525	101
Bunny8k.obj	512^3	1116	949583	993
Bunny17k.obj	512^3	1043	1056098	1004
Bunny26k.obj	512^3	1029	1137886	1000
Bunny35k.obj	512^3	1040	1213226	1001



21 pav. Diskretizavimo priklausomybė nuo plokštumų kiekio

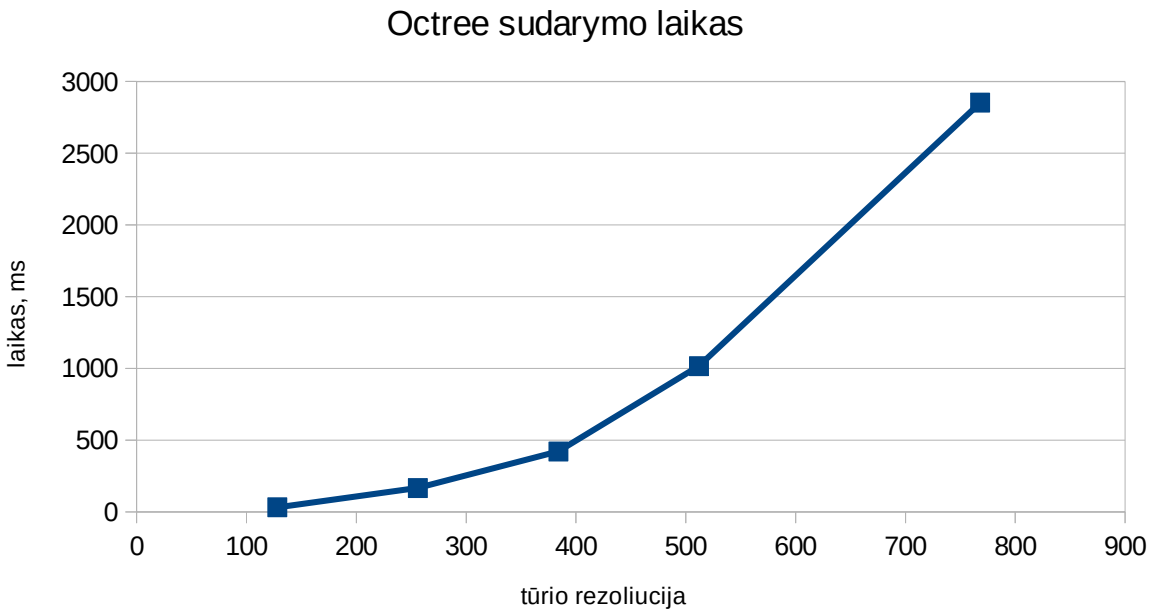


22 pav. Diskretizavimo laiko priklausomybė nuo plokštumų kiekio

Lentelė 4.4 Octree sudarymo priklausomybė nuo tūrio rezoliucijos

Geometrija	Tūrio rezoliucija	Diskretizavimo laikas, ms	Vokselių kiekis	Octree sudarymo laikas, ms
Bunny35k.obj	128	335	19042	32
Bunny35k.obj	256	428	447113	167
Bunny35k.obj	384	657	786309	421
Bunny35k.obj	512	1093	1213226	1016
Bunny35k.obj	768	2658	2330948	2853

Atlikus bandymus su skirtingų rezoliucijų tūriais nustatyta, jog octree medžio sudarymo laikas yra geometriškai priklausomas nuo tūrio rezoliucijos. Medžio sudarymo sudėtingumas susideda iš įterpiamų vokselių kiekio ir maksimalaus medžio gylio. Vokselių kiekis tūryje didėja kūbiškai, tačiau užimtų vokselių kiekis kinta kvadratiškai dėl geometrijos plokštumų dvimatės prigimties. Plokštumos kertamų vokselių kiekis priklauso nuo jos ploto. Todėl galutinis octree medžio sudarymo laikas yra kvadratiškai priklausomas nuo tūrio rezoliucijos tai pačiai geometrijai.



23 pav. Octree sudarymo laikas pagal tūrio rezoliuciją

4.3.2 Spindulių trasavimo rezultatai

Atlikus eksperimentus naudojant dvi kompiuterines sistemas surinkta vokselių atvaizdavimo algoritmo veikimo duomenys pateikti lentelėse (Lentelė 4.5 ir Lentelė 4.6). Išbandžius įvairias pikselių kiekio ir procesoriaus gijų kombinacijas, matoma, jog atvaizdavimo laikas yra tiesiškai priklausomas nuo atvaizduojamo paveikslėlio rezoliucijos. Teoriškai algoritmo veikimo laikas turėtų būti tiesiškai priklausomas nuo lygiagrečiai vykdomų gijų skaičiaus. Tačiau praktiškai dalis skaičiuojamosios galios yra sunaudojama gijų darbų paskirstymui ir sinchronizavimui, tai reiškia, jog du ar daugiau kartų padidinus vykdomų gijų kiekį, algoritmo vykdymo trukmė sumažės mažiau kartų. Šis efektas puikiai matyti 24 pav. , kuriame lygiagretinimo efektyvumas apskaičiuojamas santykinį laiko pokytį padalinus iš gijų skaičiaus. Šiuo atveju matomas žymus algoritmo lygiagretumas, keičiant gijų kiekį efektyvumas išlieka beveik pastovus. Padidinus gijų skaičių keturis kartus prarandama apie 15%-20% potencialios skaičiuojamosios galios, tai yra gaunamas 3.2 karto paspartėjimas. Kadangi tarp atskirų gijų nevyksta joks bendradarbiavimas, šis spartos nuostolis turėtų būti mažesnis. Šio nuostolio priežastis lėta darbinė atmintis, kadangi spindulių trasavimas per hierarchinę struktūrą nuolatos reikalauja naujų duomenų užklausų. Lėta darbinė atmintis nėra pajėgi aptarnauti visų branduolių užklausų, todėl dalį laiko gijos praleidžia neatlikdamos skaičiavimų, tai galima pastebėti iš laiko praleisto skaičiuojant kiekvieną pikselio reikšmę. 26 pav demonstruoja normalizuotas sugaišto laiko reikšmes, balta spalva žymi ilgesnį laiko tarpą. Galima pastebėti nepaprastai ilgai skaičiuojamus pikselius tuščios erdvės zonose, šie taškai indikuoja, kad gijos buvo priverstos laukti duomenų iš darbinės atminties. Abiejų procesorių atvejais pasiekiama maksimali algoritmo sparta naudojant gijų kiekį lygų procesoriaus branduolių kiekiui. Gijų kiekiui viršijus procesoriaus branduolių kiekį algoritmo lygiagretumo efektyvumas pradeda žymiai kristi. Atsižvelgiant į abiejų sistemų pasiektus rezultatus, matoma, jog vokselių atvaizdavimo spartos efektyvumas didėja naudojant didesnį branduolių skaičių, netgi jeigu jų taktiniai dažniai yra mažesni. Todėl vokselių atvaizdavimo sistemoms rekomenduotina turėti kuo didesnį skaičiavimams skirtų branduolių kiekį ir visus juos pajėgę aptarnauti darbinę atmintį.

Lentelė 4.5. Atvaizdavimo laikai milisekundėmis pagal pikselių kiekį ir gijų skaičių (4 CPU)

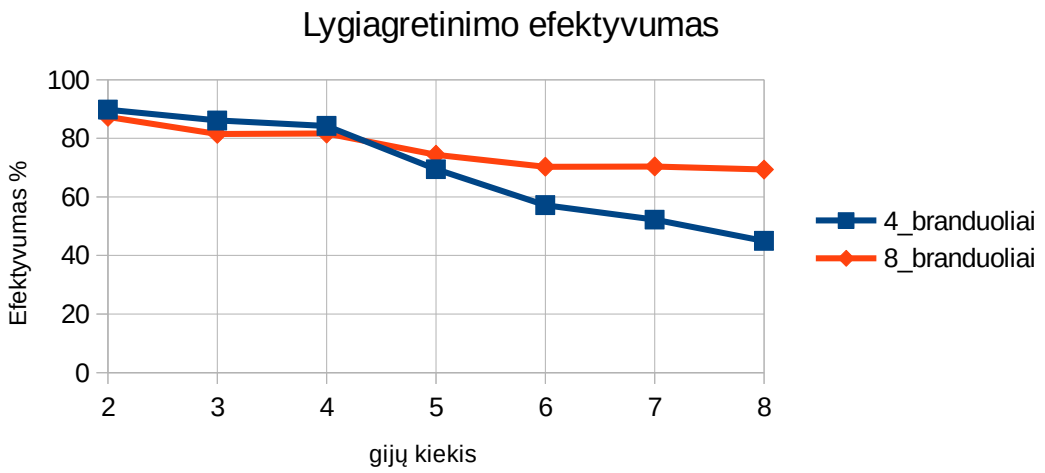
Pikseliai	1 gija	2 gijos	3 gijos	4 gijos	5 gijos	6 gijos	7 gijos	8 gijos
76800	205	117	80	64	67	74	61	61
307200	793	444	308	23	243	234	223	240
480000	1231	687	476	367	395	361	346	345
786432	2006	1116	775	607	610	595	564	544
1920000	4867	2708	1882	1450	1385	1409	1362	1376
2073600	5249	2923	2032	1558	1511	1530	1435	1458

Lentelė 4.6. Atvaizdavimo laikai milisekundėmis pagal pikselių kiekį ir gijų skaičių (8 CPU)

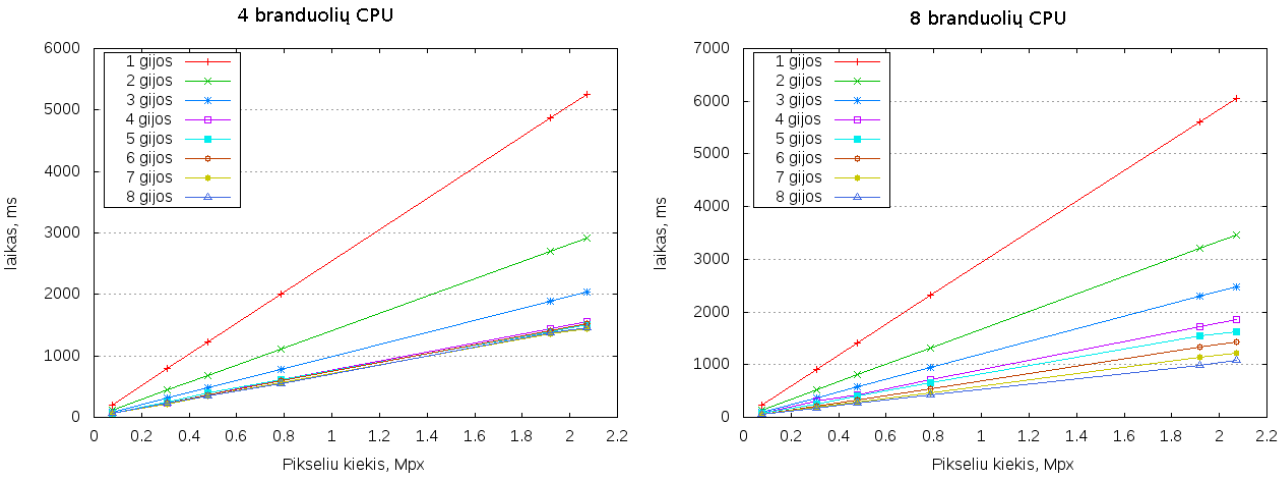
Pikseliai	1 gija	2 gijos	3 gijos	4 gijos	5 gijos	6 gijos	7 gijos	8 gijos
76800	240	136	97	80	68	57	51	54
307200	913	522	373	310	261	217	188	169
480000	1414	810	579	433	403	337	289	262
786432	2312	1320	944	706	659	550	471	427
1920000	5605	3203	2292	1715	1538	1333	1140	995
2073600	6046	3460	2473	1851	1625	1433	1227	1090

Lentelė 4.7. Išlygiagretinimo efektyvumas procentais pagal branduolių ir gijų kiekį

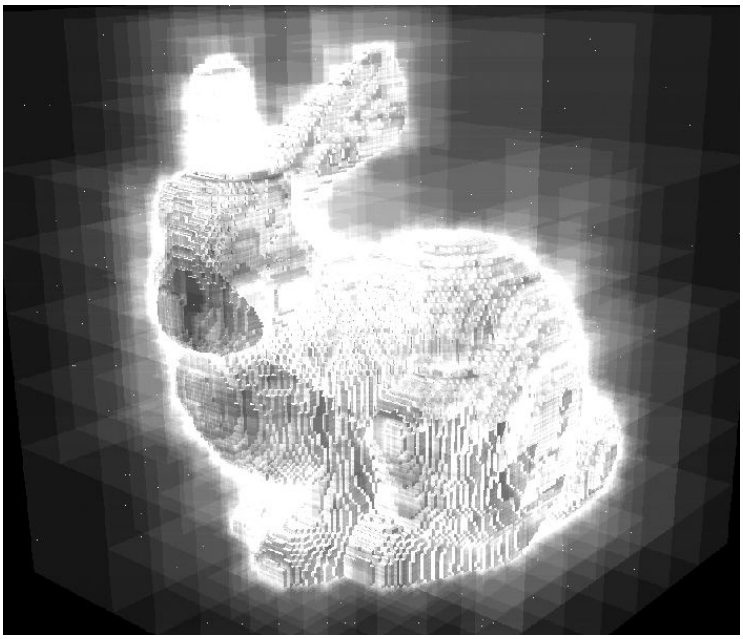
	2 gijos	3 gijos	4 gijos	5 gijos	6 gijos	7 gijos	8 gijos
4 branduoliai	89.8%	86.1%	84.2%	69.5%	57.2%	52.3%	45%
8 branduoliai	87.4%	81.5%	81.7%	74.4%	70.3%	70.4%	69.3%



24 pav. Lygiagretinimo efektyvumas

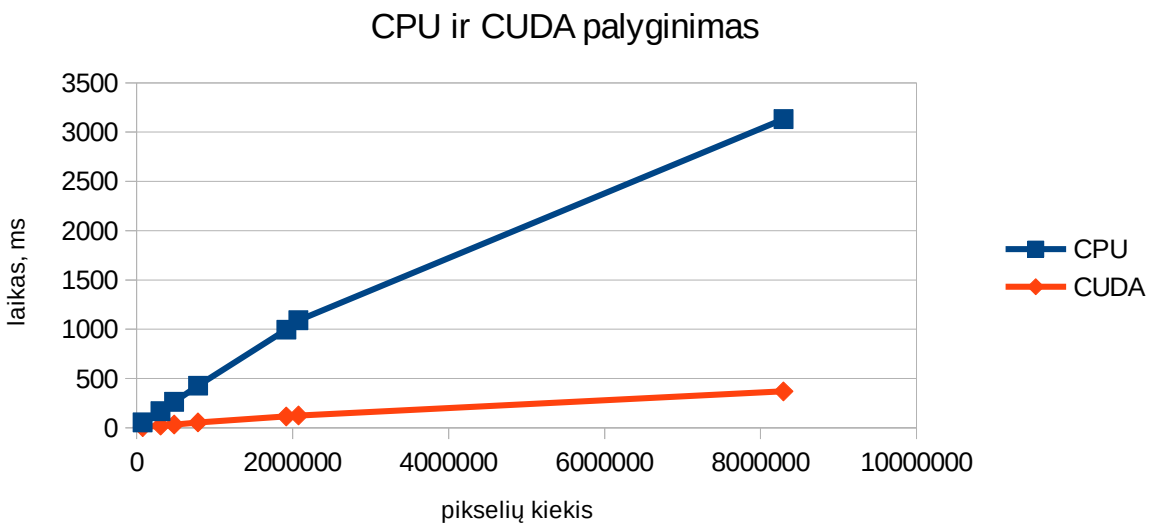


25 pav. Atvaizdavimo laikų diagramos



26 pav. Santykinės laiko trukmės kiekvienam pikseliui

Pasinaudojus CUDA technologija spindulių trasavimo algoritmas buvo perkeltas vykdymui naudojant grafinio procesoriaus resursus. Tyrimo metu naudota techninė įranga turi 1024 CUDA branduolius, tai reiškia, kad vienu metu lygiagrečiai galima vykdyti 1024 gijas. Nors grafinio procesoriaus taktinis dažnis yra gerokai mažesnis už centrinio procesoriaus, jis įgauna pranašumą vykdomų gijų kiekiu atlikdamas daugiau skaičiavimų per mažesnį laiko tarpą. Identiškų duomenų apdorojimo centriniu procesoriumi ir grafiniu procesoriumi laikų duomenys pateikti Lentelė 4.8 ir 27 pav. Matoma, jog pasiektas mažiausiai 8,5 karto paspartėjimas. Tikėtasi kur kas didesnio paspartėjimo, tačiau atsižvelgus, jog algoritmo realizacija taikyta centriniam procesoriui, ir nėra optimali grafinio procesoriaus architektūrai, šis spartos padidėjimas yra gana didelis. Naudojantis grafiniu procesoriumi galima atvaizduoti 800x600 rezoliucijos kadrus realiu laiku, tai yra 30 kadrų per sekundę. Pritaikius algoritmo atminties modelį geresniam duomenų kešavimui grafinio procesoriaus blokuose, būtų įmanoma pasiekti dar didesnės rezoliucijos realaus laiko atvaizdavimą.



27 pav. CPU ir CUDA laikų palyginimas

Lentelė 4.8. CPU ir CUDA laikų palyginimas

Pikselių kiekis	CPU 8 cores	CUDA 1024 cores	Pagreitėjimas kartais
76800	54	6	9
307200	169	22	7.68
480000	262	32	8.18
786432	427	53	8.05
1920000	995	115	8.65
2073600	1090	125	8.72
8294400	3132	369	8.48

4.4 Eksperimentavimo išvados

Atlikus eksperimentus matoma, jog su dabartinėmis techninėmis galimybėmis vokselių atvaizdavimas yra praktiškai naudingas kompiuterinėje grafikoje. Tiesioginiai atvaizdavimo algoritmai pasižymi savo paprastumu ir leidžia greitai pasiekti paprastus vaizdus Tačiau norint pasiekti geriausią įmanomą našumą reikia taikyti sudėtingas algoritmo optimizacijas. Tai gali būti centrinių procesorių SIMD savybių išnaudojimas, arba algoritmo pritaikymas bendros paskirties skaičiavimams naudojant grafinius procesorius. Pagrindinė tūrinių duomenų problema jų atminties sunaudojimas, tik palyginti nedideli tūrio kiekiai gali būti sutalpinami darbinėje arba grafinio procesoriaus atmintyje. Žinoma šia problemą galima išspręsti pasinaudojant hierarchinių struktūrų galimybe duomenis transliuoti dinamiškai.

5 Išvados

1. Realizavus vokselių atvaizdavimo sistemą, tyrimo metu nustatyta, jog vokselių atvaizdavimas realiu laiku jau yra įmanomas. Pastebėta, kad realizacijos atžvilgiu vokselių atvaizdavimo algoritmai yra labai paprasti, tačiau norint išgauti didesnę spartą būtina taikyti sudėtingas optimizacijas, tokias kaip procesų išlygiagretinimas ir atminties panaudojimo minimizavimas.
2. Atlikus testus su keliomis techninėmis konfigūracijomis nustatyta, jog vokselių atvaizdavimo algoritmų paremtų ray-casting principu išlygiagretinimo efektyvumas yra beveik optimalus. Jis siekia net 80% ir dar turi galimybę būti pagerintas. Algoritmus vykdant su bendros paskirties grafiniais procesoriais nustatyta,jog išgaunamas mažiausiai 10 kartų paspartėjimas lyginant su centriniu procesoriumi. Šią spartą galima dar padidinti algoritmus pritaikant prie grafinių procesorių atminties naudojimo modelio.
3. Bandymų metu pastebėta, jog octree medžio atminties sunaudojimas yra gana artimas plokštumų ir jas dengiančių tekstūrų užimamos atminties sumai, todėl octree yra puiki alternatyva saugoti ir geometriniams ir spalviniams duomenims.

Literatūra

1: M. Meißner, H. Pfister, R. Westermann, C.M.Wittenbrink, „Volume Visualization and Volume Rendering Techniques", 2000
2: J. Krüger, R. Westermann, IEEE Visualization 2003, „Acceleration Techniques for GPU-based Volume Rendering“,
3: Ömer Cengiz ÇELEBİ, „Scientific Visualization and 3D Volume Rendering”, 2014-11-09, http://www.byclb.com/TR/Tutorials/volume_rendering/
4: Milan Ikits, Joe Kniss, Aaron Lefohn, Charles Hansen, “GPU Gems”, Chapter 39. Volume Rendering Techniques,
5: James Arvo, "Graphics Gems II", 2013
6: Branislav Sileš, “Atomontage engine”, 2014-01-08, http://www.atomontage.com/
7: Euclideon, "Unlimited Detail", 2014-01-08, http://www.euclideon.com/technology-2/
8: Thinkbox Software, „Krakatoa Voxel Rendering“, 2014-01-09, http://www.thinkboxsoftware.com/krak-voxel-rendering/
9: Stanford University, The Stanford 3D Scanning Repository, 2015, http://graphics.stanford.edu/data/3Dscanrep/