KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERINIŲ TINKLŲ KATEDRA

Vytautas Mikalauskas

Planetinio kūno modeliavimo algoritmų tyrimas

Magistro darbas

Darbo vadovas
doc. dr. K. Baniulis

Lietuvių kalbos konsultantė

Kaunas
2004

Turinys

1. Įvadas

Paskutinįjį XX a. dešimtmetį labai sparčiai pradėjo plėstis kompiuterinės grafikos panaudojimo sritys. Dėl personalinių kompiuterių didelio populiarumo ir greito tobulėjimo jiems tapo prieinamos dauguma vizualizacijos technologijų, kurias ankstesniais metais galėjo atlikti tik specializuotos grafinės stotys, pavyzdžiui, gerai žinomos firmos "Silikon Graphics Inc." kompiuteriai. Ši technika nėra skirta masiniam vartotojui, yra gana brangi ir dažnai tinka tik siauros specializacijos uždaviniams spręsti. Situacija pasikeitė, ėmus masiškai gaminti personaliniams kompiuteriams skirtus trimačio vaizdo spartintuvus. Šie įrenginiai leido realizuoti daugelį vaizdo pateikimo galimybių, kurios yra reikalingos norint formuoti ekrane realistiškai atrodančius objektus. Šiuo metu vidutiniškai du kartus per metus grafinių spartintuvų gamintojai išleidžia dvigubai padidėjusios spartos vaizdo procesorius. Dėl didelės konkurencijos šiuo metu masinę vaizdo apdorojimo procesorių (*angl.* graphic processing unit, GPU) gamybą vykdo tik dvi firmos: JAV įsikūrusi bendrovė "Nvidia Inc." ir Kanados firma "ATI Inc.". Pirmoji leidžia GPU seriją "Geforce", antroji – "Radeon". Pastaruoju metu į rinką bando įžengti senas GPU gamybos tradicijas turinti firma "S3 Graphics Inc." su savo originaliu vaizdo apdorojimo procesoriumi "Delta Chrome".

Pagrindiniai vaizdo apdorojimo procesorių panaudojimo privalumai yra šie:

- Realizuojamas paralelinis kompiuterio procesoriaus (*angl.* central processing unit, CPU) ir vaizdo apdorojimo procesoriaus darbas, leidžiantis neapkrauti CPU vaizdo formavimo operacijomis, kas didina jo spartą
- Geometrijos ir apšvietimo intensyvumo skaičiavimo operacijas, kurios yra imlios skaičiavimams, galima perduoti GPU
- Trimačio vaizdo spartintuvo (toliau – 3D spartintuvas) atmintinėje laikomos objektų tekstūros taupo operatyvinės atmintinės resursus. Trūkstant spartintuvo atmintinės, galima panaudoti dalį operatyvinės atmintinės kaip grafikos tekstūrų ir daugiakampių viršūnių saugyklą.

Sparčiam duomenų persiuntimui iš operatyviosios atmintinės į 3D spartintuvo atmintinę realizuota speciali grafinių duomenų magistralė AGP (*angl.* advanced graphic port), kurios pradinis 66 MHz greitis buvo priimtas už vienetą. Šiuolaikiniai 3D spartintuvai turi 8x (aštuonis kartus greičiau dirbančią) AGP magistralę, netrukus ją pakeis naujas standartas "PCI Express", savo sparta prilygsiantis 16x AGP magistralės spartai. Be to, 3D spartintuvo atmintinė išaugo iki 256 MB, kas mažina duomenų persiuntinėjimo poreikį ir greitina vaizdo posistemės darbą.

Visos šios techninės galimybės įgalino kompiuteriu modeliuoti objektų vaizdus, kuriems matematiškai aprašyti reikalingas labai didelis (kartais iki keliolikos milijonų) daugiakampių. 3D spartintuvai operuoja trikampiais, jų vaizdo generavimo sparta matuojama formuojamais vaizdo kadre trikampiais per sekundę arba apdorojamomis objektų viršūnėmis per sekundę. Pavyzdžiui Geforce 5800 GPU gali apdoroti iki 30 mln. viršūnių per sekundę. Tokie dideli greičiai leidžia realiame laike formuoti natūralius gamtos fenomenus, tokius kaip landšaftai, reljefai, atmosferos reiškiniai, vandens paviršiai, augalai, gyvūnai ir t.t. Šios galimybės labai reikalingos ten, kur mokymo tikslams reikalinga pateikti vaizdinę medžiagą. Kuo ji bus vizualiai artimesnė originaliam vaizdui, tuo geriau tą informaciją įsisavins besimokantys žmonės, nes ji atitiks jų patirtį. Viena iš tokių sričių šio darbo autoriui yra mokyklinis astronomijos kursas, kurio mokymu jam tenka užsiimti. Jau pradėjus daryti šį darbą kursas buvo panaikintas, tačiau medžiaga apie Saulės sistemos planetų tyrinėjimus buvo perkelta į 10 klasės fizikos kursą. Taip pat grafinės technologijos labai tinka informatikos kurse vaizdinei medžiagai sukurti ir parodyti, kokie algoritmai gali būti panaudoti realistiniams vaizdams formuoti, moksleiviai gali pamatyti kompiuterio galimybes vaizdo formavime ir patys sukurti planetinių objektų pavyzdžius. Ypač naudinga susipažinti su šia medžiaga moksleiviams, lankantiems papildomus užsiėmimus informatikos būreliuose. Tokiu būdu šiame darbe buvo nuspręsta sukurti programinę sistemą, demonstruojančią planetinių kūnų modeliavimo galimybes ir tam panaudojamus algoritmus, pateikiančią duomenis apie dirbančios kompiuterio vaizdo posistemės darbinius parametrus.

Šis planetinių objektų modeliavimo algoritmų tyrimas buvo atliktas, siekiant nustatyti efektingiausius algoritmus, kurie leistų sukurti programinę sistemą, vartojančią sąlyginai nedaug kompiuterinių resursų (atsižvelgiant į tai, kad nepaisant mokyklų kompiuterizavimo proceso spartėjimo nemaža mokyklų turi gana nedidelę kompiuterinių resursų bazę) ir leidžiančią moksleiviams praktiška išbandyti realių planetinių objektų kūrimo technologiją ir susipažinti su jos teikiamomis galimybėmis.

Išvada: Šiuolaikinės kompiuterių grafikos galimybės leidžia kurti mokomąsias sistemas, kuriose dominuoja aukštas vizualizacijos kokybės lygis, ko pasėkoje vartotojas lengviau priima kompiuterinę mokomąją medžiagą.

2. Uždavinio aplinkos analizė

Pagrindinis darbo uždavinys: sudaryti planetinių kūnų modeliavimo sistemos projektą

Kad galėtume šį uždavinį įvykdyti reikalinga apžvelgti šio tipo technologijose panaudojamus algoritmus ir susipažinti su analogiškomis jau sukurtomis sistemomis.

Pasaulyje pradėjus sparčiai tobulėti kompiuterinei įrangai, tame tarpe trimatės grafikos įrenginiams, ne viena mokslo įstaiga pradėjo tyrimus kompiuterinės grafikos algoritmų srityje. Nuo XX a 8-ojo dešimtmečio vidurio pasirodė nemaža sukurtų efektyvių algoritmų, leidžiančių modeliuoti gamtinius fenomenus (mūsų atveju – planetinius kūnus ir jų reljefą) realaus laiko uždaviniuose. Pagrindinius jų apžvelgsime čia.

Efektingiausi (ir sudėtingiausiai realizuojami) algoritmai buvo ir tebėra kuriami dinaminio reljefo detališkumo keitimo srityje. Tai dažniausiai paviršių supaprastinimo algoritmai, kurie gali būti taikomi:

- Aukščių masyvams ir paviršiams iš parametrinių kreivių
- Uždaro kontūro kreiviniams paviršiams
- Neuždaro kontūro kreiviniams paviršiams.

Bendru atveju šie algoritmai charakterizuojami skaičiavimų greičiu ir dviem parametrais: viršūnių pradiniuose duomenyse skaičiumi n ir viršūnių kaip algoritmo darbo rezultatų skaičiumi m. Dažniausiai $m << n$.

Kadangi šiame darbe nagrinėjami tik aukščių masyvų pagalba formuojami planetinių kūnų modeliai, apsiribosime algoritmais, kurie paprastina būtent tokius paviršius. Paprastai išskiriamos šešios tokių algoritmų grupės:

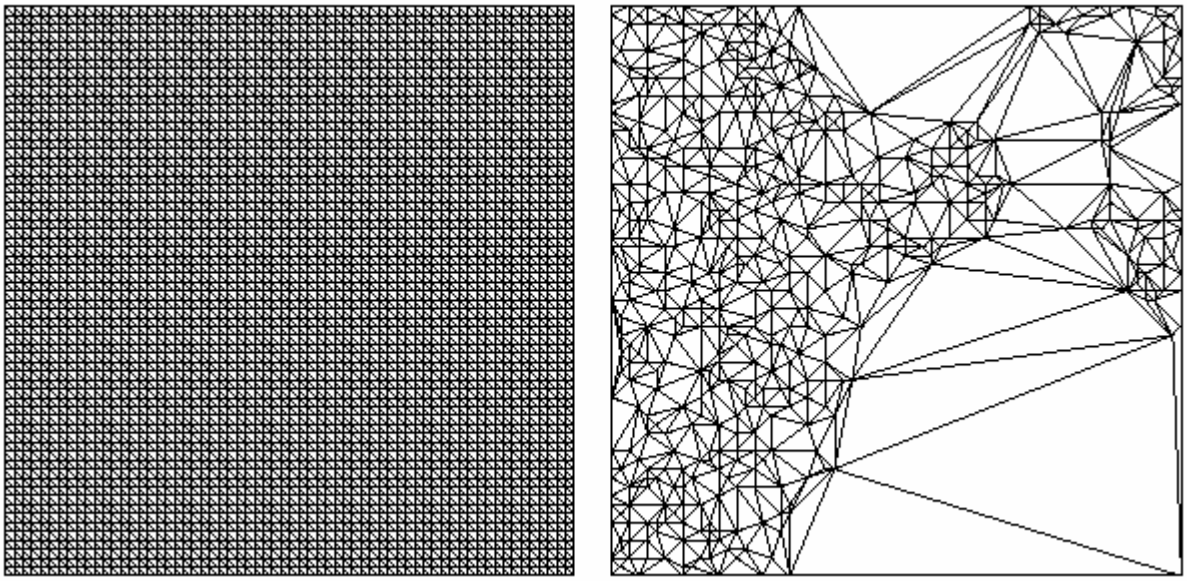
- Vienodo sudalinamo tinklelio metodai
- Hierarchinio sudalinimo metodai
- Ypatingų viršūnių metodai (angl. *feature metods*)
- Kokybės gerinimo metodai (angl. *refinement metods*)
- Viršūnių atmetimo (angl. *decimation*)
- Optimumo paieškos metodai

Pirmieji, vienodai sudalinamo tinklelio metodai yra patys paprasčiausi, naudoja lygiai sudalintą x ir y koordinačių tinklelį. Hierarchinio sudalinimo algoritmai naudoja medžio tipo (kvadro- ir octa- medžiai) struktūras rekursyviai dalindami paviršius į regionus, kurie matematiškai atitinka medžio šakas. Abu pirmieji metodai vadinami „skaldymo ir valdymo“ strategijos metodais. Hierarchinio dalinimo

algoritmas panaudotas ir šiame darbe. Jo privalumas – algoritmas konstruktyvus ir nereikalauja didelės apimties matematikos žinių, nors medžio tipo struktūrų valdymas gali būti labai sudėtingas. Paviršiai formuojami iš kvadratų arba trikampių. Kadangi trimačio vaizdo spartintuvai gerai dirba su paviršiais iš abiejų rūšių daugiakampių, šiuos algoritmus lengviau optimizuoti šioms aparatūrinėms konfigūracjoms.

Likusios kitos keturios algoritmų grupės naudoja matematiškai sudėtingesnius paviršių skaidymo į daugiakampius būdus, dažniausiai Delaunay skaidymą į trikampius. Ypatingų viršūnių algoritmai išskiria iš įėjimo viršūnių sekos aibę „ypatingų“ ir visus skaidymo į daugiakampius veiksmus atlieka su jomis. Kokybės gerinimo metodai, apdorodami viršūnių masyvus, dirba principu „iš viršaus į apačią“, sukurdami naujas, papildomas viršūnes ir pritaikydami parametrinių kreivių skaidymo metodus trikampiams formuoti. Viršūnių atmetimo metodai dirba atvirkštiniu principu – mažina viršūnių skaičių. Tai yra principas „iš apačios į viršų“. Na, o optimalūs metodai dažniausiai yra teoretiniai ir su praktika susiję mažai.

Didelę svarbą turi dalinimo į trikampius metodai. Vienas iš populiariausių bendrųjų skaidymo metodų – TIN (angl. *Triangulated Irregular Network*). Atliekant tokį skaidymą aukščio koordinatė yra x ir y koordinačių funkcija.



Pav. 1 Skaidymas trikampiais TIN metodu

Kitas skaidymo trikampiais metodas – Delaunay skaidymas trikampiais nenaudoja aukščio koordinatės ir yra grynai dvimatis metodas. Teorinės galimybės didelės, bet praktikoje taikomas retokai.

Paskutiniais metais paplito nuosekliai kintančio dinaminio detališkumo algoritmai (*angl. santr.* CLOD). Jie patogūs tuo, kad juos lengviau suderinti su kitais uždaviniais, kurie iškyla norint atvaizduoti realistišką landšaftą. Tai ir daugiakampių tekstūravimas, apšvietimas, ir eilė kitų uždavinių. Šiuolaikiniai trimačio vaizdo spartintuvai gali šiuos uždavinius spręsti efektyviai, bet informacijos kiekis ir

skaičiavimų apimti yra tokie, kad be optimizuojančių algoritmų išsiversti dar ilgai bus sunku. Pats pirmasis tokio tipo algoritmas buvo P. Lindstromo 1996 m. paskelbtas supaprastinto aukščių masyvų vaizdavimo algoritmas. Nors šis algoritmas turi vieną praktinį trūkumą (dirba pagal principą „iš apačios į viršų“, kas praktikoje nėra patogu), tačiau jis tapo „pamatais“ eilei puikių algoritmų, naudojamų iki šiol. Ši ištobulinta kryptis vyrauja šiandien tarp kompiuterinės grafikos kūrėjų. Algoritmų yra trys:

- ROAM algoritmas (angl. *Realtime Optimized Advanced Mesh*). Paremtas trikampių skaidymų pagal medianą į smulkesnius.
- Algoritmai kvadromedžio pagrindu. Hierarchinio dalijimo algoritmai, dažniausiai skaido paviršių į kvadratinius regionus, o jau po to į trikampius. Smarkiai apkrauna procesorių, todėl dažnai optimizuojami nukraunant dalį operacijų ir perduodant trimačio vaizdo spartintuvui. Vienas populiariausių tokio tipo algoritmų – **Roetinger 98**, naudojamas komerciniame žaidime „Aquanox“.
- Algoritmai naudojantys „bangeles“ (angl. *wavelets*). Patogūs tuo kad galima naudoti grafinių duomenų suspaudimą realiame laike (taupoma atmintinė)

Šiuo metu, ROAM algoritmams šiek tiek praradus populiarumą dėl sunkaus integravimosi su tekstūravimo ir kolizijų apskaičiavimų uždaviniais, rinkoje populiariausi algoritmai kvadromedžių pagrindu.

CLOD algoritmai turi ir trūkumų, bet jie atsiranda dėl kompiuterio resursų ribotumo. Pavyzdžiui, modeliuojant didelį planetinį kūną reikalinga žinoti milijonų viršūnių koordinates. Darbui reikalinga tik keli tūkstančiai viršūnių. Tačiau kur laikyti likusias ? Kraunant tokius viršūnių kiekius į atmintinę mums teks laukti programos pasileidimo kelias ar net keliolika minučių. Nors procesoriai dirba greitai, kompiuterio magistralės šiuo atveju bus perkrautos. Dar blogiau, jei įkrovinėti duomenis reikės programos darbo metu – vietoje normalaus vaizdo keitimosi matysime mirksintį vaizdelį, į kurį bus nemalonu žiūrėti. Atmintinė irgi ribotos talpos – nedažnai sutiksime kompiuterį su 1 GB operatyviosios atmintinės. Taigi iškyla duomenų srauto sumažinimo problema. Vienas iš būdų šiai problemai spręsti – duomenų įkrovimas puslapiais – ribotomis porcijomis pagal poreikį. Kadangi kompiuterių OS realizuoja daugiaužduviniškumą pakankamai gerai, galima duomenų įkrovimą reguliuoti ir valdyti. Realizaciją galima rasti Internete adresu http:// tulrich.com/geekstuff/chunklod.htm . Programai reikia pakankamai galingo kompiuterio – bent jau Celeron 2 GHz su 256 MB RAM atmintinės, Geforce 2 MX klasės 3D spartintuvo. Kitu atveju programos generuojamas vaizdas bus trūkčiojantis, arba programa atsisakys pasileisti, pranešdama apie resursų trūkumą.

Atsižvelgdamas į esamą situaciją šiame darbe dinaminiam detalizavimui pavaizduoti panaudojau CLOD algoritmą kvadromedžių pagrindu.

Planetų modeliavimo kompiuterinės grafikos priemonėmis darbų Internete yra paskelbta nemažai, bet pavykusių realizacijų yra nelabai daug.

Fraktalinės geometrijos pagalba realizuotas modelis

Interneto svetainė: www.baddoggames.com/planet

Pilnai modeliavimo galimybes realizuojantis darbas. Tai modeliuojanti planetą programa (autorius Rob James, profesionalus programuotojas). Reljefas formuojamas atsitiktinių skaičių generatoriaus pagalba generuojant Perlino triukšmą. Paviršiaus tekstūros taip pat generuojamos, kiekvieną kartą paleidus programą modelis šiek tiek skiriasi nuo ankstesnio.

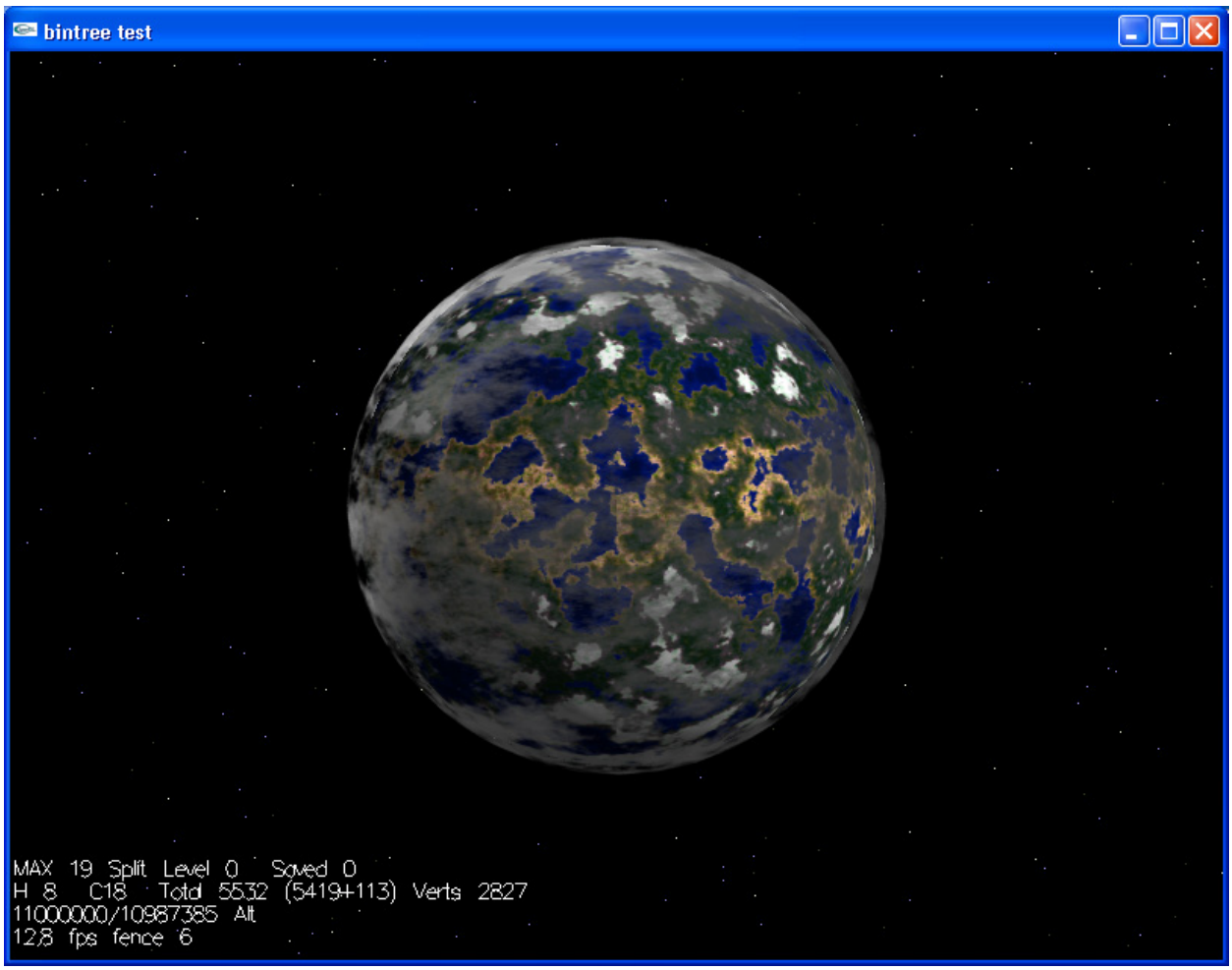
Privalumai:

1) Gerai vizualizuojama stebėtojo buvimo taško aplinka.
2) Reljefas pakankamai atitinka natūralų, gamtinį
3) Realistinį įspūdį sustiprina tekstūrų pagalba realizuoti atmosferiniai efektai
4) Gerai išnaudotos 3D spartinimo technologijos – didelis kadrų dažnis net esant silpnam spartintuvui

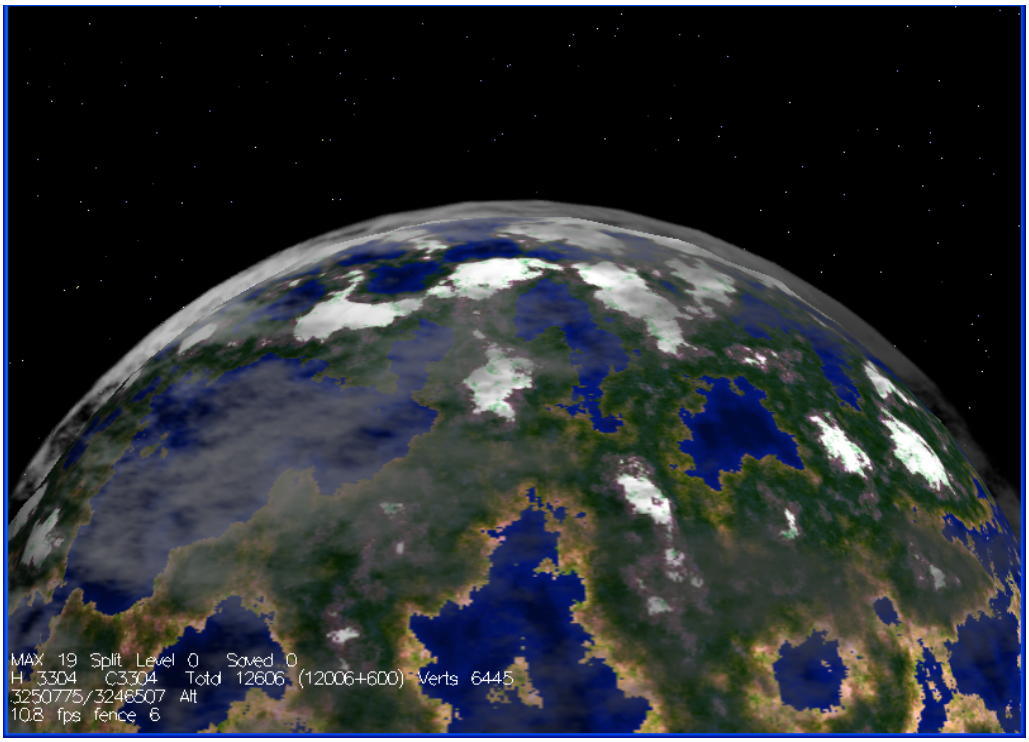
Trūkumai:

1) Dėl generavimo fazės programa pasileidžia per 2 – 3 minutes, fraktaliniai algoritmai imlūs laikui.
2) Nepakankamai universali – atskiros programos versijos skirtos AMD ir Intel sistemų procesoriams
3) Negalima keisti modelio parametrų

Žemiau pateikiami programos generuojamo reljefo pavyzdžiai. Programa ekrane pateikia pagrindinius kadrų generavimo ir paviršiaus detalizacijos algoritmo parametrus.

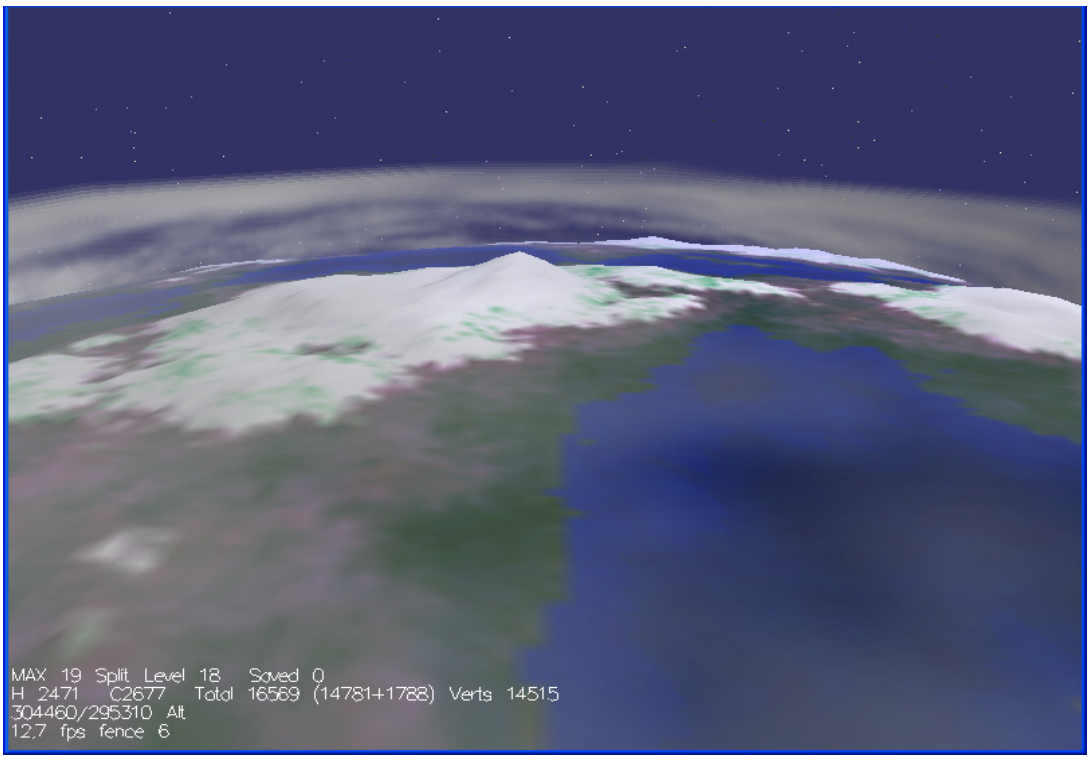


Pav. 2 Planetos modelis iš toli



Pav. 3 Vaizdas priartėjus

Priartėjus jau galima išskirti reljefo detales.



Pav. 4 Vaizdas netoli paviršiaus

Nedideliame aukštyje matomas pilnas reljefas

Modelio ypatybė – kiekvieną kartą generuojamas skirtingas reljefas, todėl ši programa ideali kūrimo technologijos galimybių demonstravimui

Paprastinamo daugiakampių tinklelio pagalba realizuotas modelis

Interneto svetainė: www.gamasutra.com/features/soneil01.htm

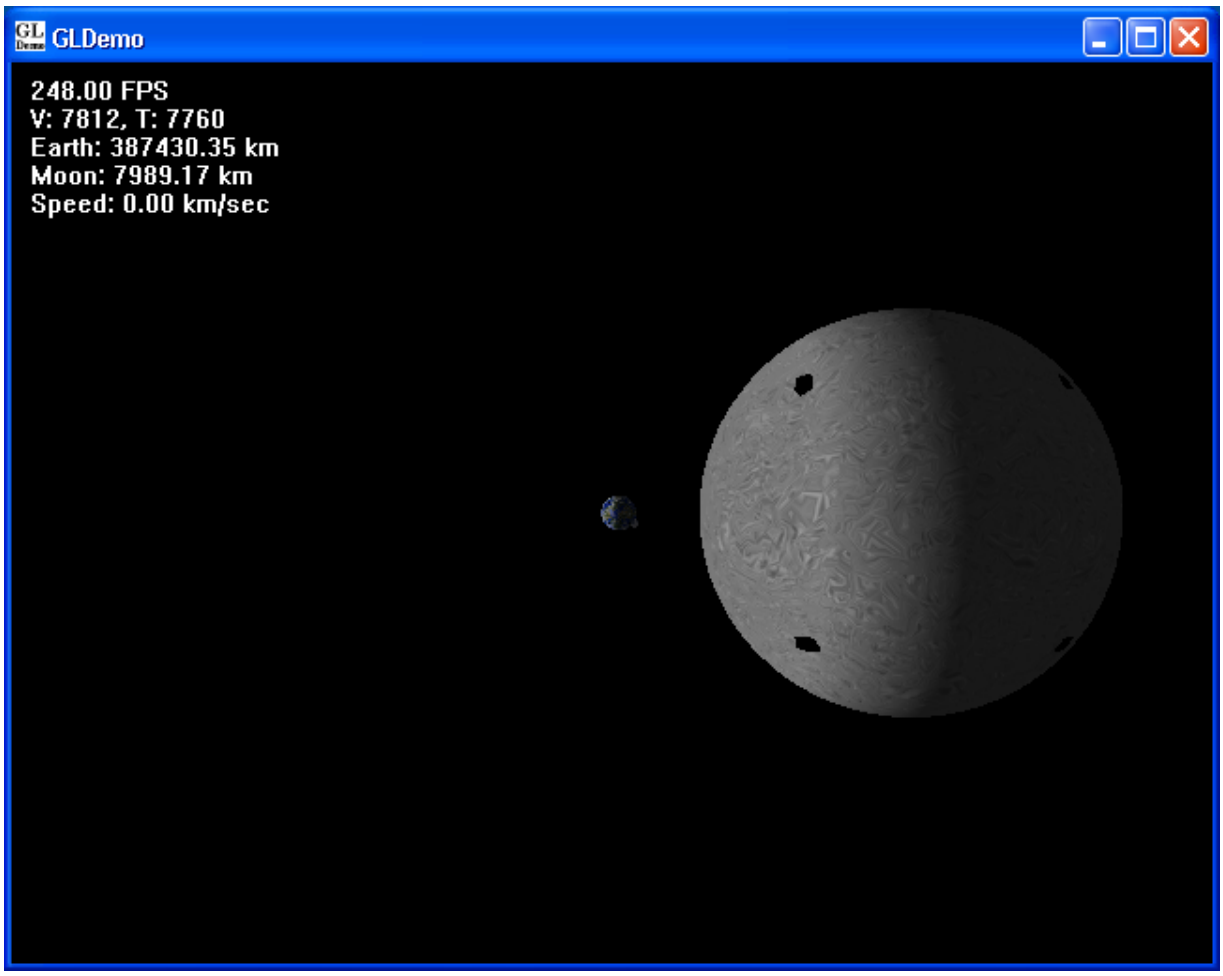
Darbas realizuotas kaip pavyzdys žaidimų kūrėjams ir realizuoja ROAM algoritmą pritaikytą sferiniam kūnui. Realizacija panaši į aukščiau apžvelgtą darbą, bet tekstūravimas realizuotas stacionariomis, o ne dinaminėmis tekstūromis. Reljefas atsitiktinis, generuojamas Perlino triukšmas reljefui gauti.

Privalumai:

1) Tinka algoritmų darbui tyrinėti, yra šaltinių tekstai C++ kalba
2) Nėra generavimo fazės, greitai paleidžiama programa

Trūkumai:

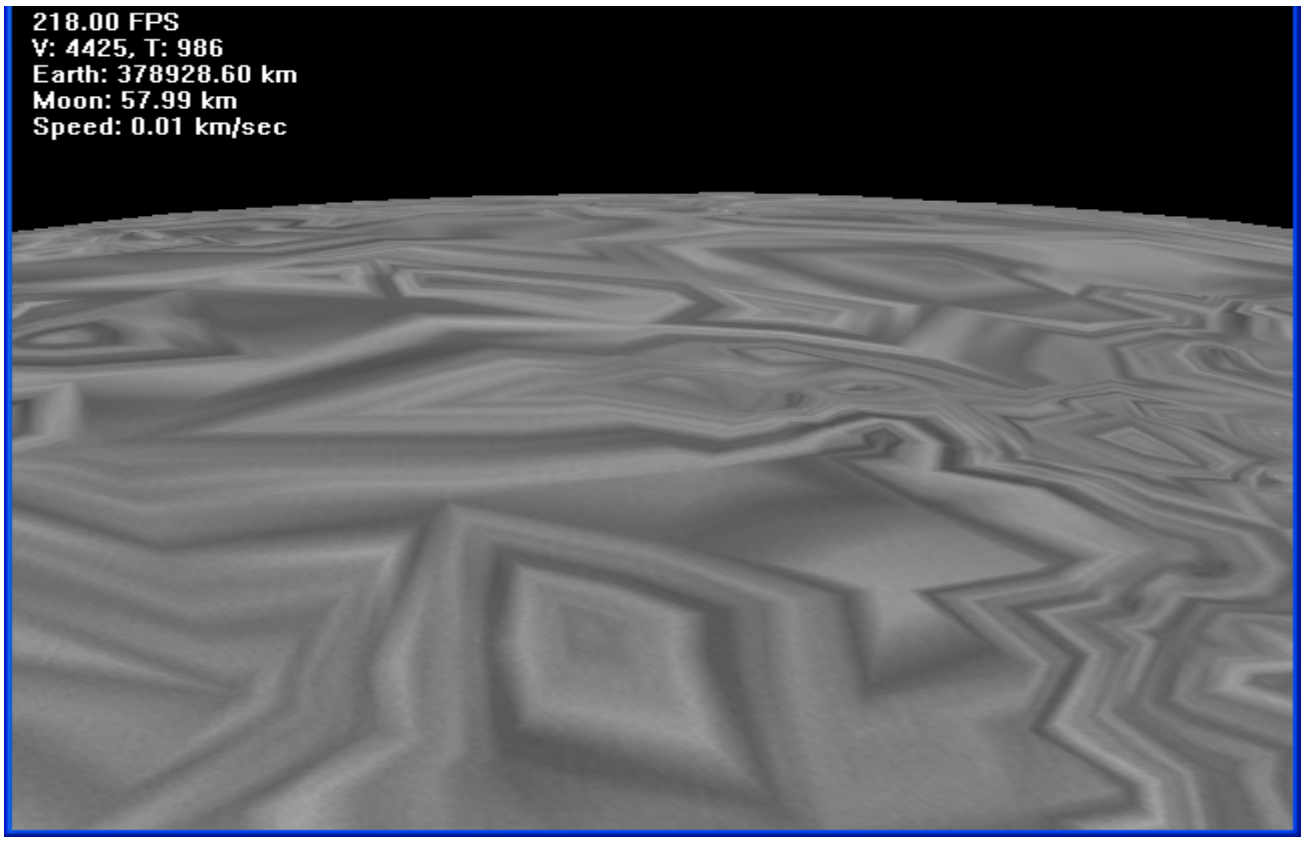
1) Nepakankamai realistiški landšaftai
2) Žema tekstūravimo kokybė
3) Nepakankamai stabilus algoritmas, tam tikrais atvejais “lūžta”



Pav. 5 Vaizdas iš toli

Vaizde iš toli matyti Mėnulis, Žemė – antrame plane

Pav. 6 Vaizdas priartėjus



Pav. 7 Vaizdas arti paviršiaus

Pagrindinis modelio trūkumas - stiprus mirgėjimas nuo „šokinėjančių“ naujai generuotų viršūnių. Nors ir su trūkumais, modelis gerai demonstruoja ROAM algoritmo galimybes. Mirgėjimą galima sumažinti naudojant spartesnį 3D spartintuvą (Nvidia Geforce 4 arba ATI Radeon 8500 lygio). Vaizdo realistiškumas nukenčia dėl paviršiaus generavimo algoritmo suprimityvinimo – jis pralaimi fraktaliniams algoritmams, kurie yra netiesiniai ir generuoja realistišką reljefą.

Realios planetos reljefo duomenų pagalba realizuotas modelis

Interneto svetainė: drtypo.free.fr

Tai Marso planetos modelis, panaudoti JAV Nacionalinės aeronautikos valdybos NASA inicijuoto MOLA (*angl.* Mars observer landscape analysis) projekto geografiniai Marso duomenys. Duomenis surinko Marso dirbtinis palydovas “Mars Global Surveyor”. Reljefas šiek tiek „padailintas“ viršūnių apdorojimo programomis ant trimačio vaizdo spartintuvo. Ekrano nuotraukos daro įspūdį, gal kiek problematiškiau judinti kamerą – nėra pagreitinto judėjimo, tačiau realistiškumo atžvilgiu tai vienas geriausių pavyzdžių. Dirba stabiliai ir jokių problemų darbo metu nebuvo. Reikalauja gana galingo 3D akseleratoriaus, bet yra galimybė naudoti mažesnio tikslumo tekstūras, jei pastarasis silpnesnis.

Privalumai:

1) Tiksliai atkuriamas planetos reljefas
2) Landšaftas realistiškas, kiek leidžia 3D spartintuvo galimybės
3) Nėra generavimo fazės, greitai paleidžiama programa
4) Pilnai išnaudojamos 3D spartintuvų technologijos galimybės, šešėliavimo technika
5) Yra skirtingo tikslumo duomenų rinkiniai, pagal 3D spartintuvo atmintinės kiekį

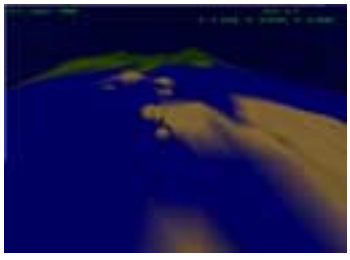
Trūkumai:

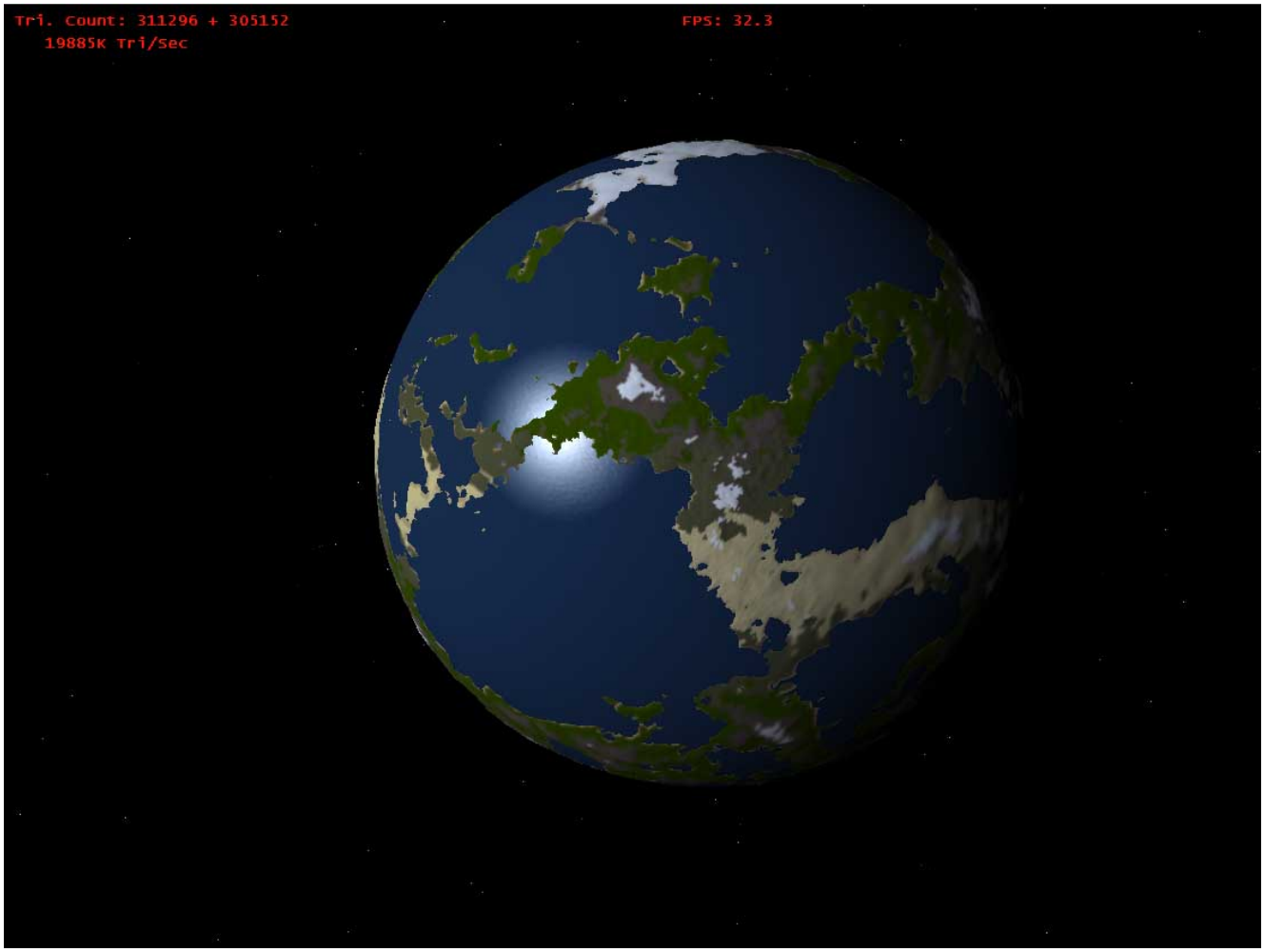
1) Reikalauja galingo 3D spartintuvo, didelės apimties video atmintinės (128 MB pilnai raiškai)
2) Paskutinę versiją 0.75 galima paleisti tik su Microsoft DirectX 9.0 palaikančiu spartintuvu (Geforce FX arba Radeon 9500 GPU – minimumas), nes programa naudoja tekstūrų šešėliavimo (*angl.* shaders) galimybes
3) Esant tam tikriems stebėtojo judėjimo greičiams mirga tekstūros – pasireiškia tekstūrų užpildymo greičio trūkumas šiuolaikiniuose 3D spartintuvuose

Pav. 8 Marso vaizdas iš toli

Dar vienas bandymas yra adresu
http://www.chem.pwf.cam.ac.uk/~jdh30/programming/opengl/lod/index.html
Autorius Jon Harrop pabandė realizuoti reljefo detalizacijos algoritmą ant dodekaedro. Veikiančios programos nėra, bet bazinio algoritmo kodą galima parsisiųsti. Skirtas Linux sistemai, naudoja matematinę "Blitz++" biblioteką vektorių koordinačių skaičiavimams.

Reikia paminėti magistrinį darbą, publikuojamą adresu www.dgp.toronto.edu/~dh/research.html . Autorius 2001 m. apgynė magistrinį darbą apie planetos modeliavimą. Svetainėje pateikiama nemažai medžiagos, momentinių nuotraukų ir demonstracinės programos. Chronologine tvarka parodyta, kaip buvo tobulinama programos realizacija





Pav. 9 Žemės modelis Dave Hill magistriniame darbe

Internete publikuojama medžiaga leidžia manyti, kad planetinių kūnų modeliavimo programos yra gana sudėtingi darbai. Pasiekti vaizdo realistiškumo pavyksta toli gražu ne visiems. Daugumas darbų nėra laisvai platinami. Tačiau techniškai sukurti realaus laiko reljefo planetos modelį visiškai įmanoma. Realistiškumui didžiausią įtaką turi ne daugiakampių vaizdavimas, bet tekstūrų suderinamumas, kas yra šiek tiek daugiau menininko, o ne inžinieriaus programuotojo uždavinys.

Apžvelgta medžiaga padėjo susidaryti bendrą vaizdą apie egzistuojančius šio tipo uždavinių sprendimus specifinėje programų rinkoje. Taip pat gavau vertingų duomenų apie algoritmų efektyvumą tokio tipo uždavinių sprendimų realizacijose.

Išvada: Atlikus programinės įrangos analizę buvo nustatyti specializuotai planetinių kūnų projektavimo programinei įrangai keliami reikalavimai bei vartotojų poreikiai.

3. Programos grafinę sąsają (interfeisą) palaikančios bibliotekos

Kuriant trimačio vaizdo generavimo sistemas šiuo metu plačiausiai naudojami du grafinės sąsajos (interfeiso) standartai :

Microsoft DirectX (šiuo metu galutinė yra 9.0b versija)

Privalumai:

1) Profesionaliai realizuotas, šiuo metu yra pagrindinis standartas *de facto*
2) Pripažįstamas didžiausių programinės įrangos kūrėjų
3) Orientuotas į naujausias 3D vaizdo spartinimo technologijas

Trūkumai:

1) Siauro panaudojimo profilio – tik MS Windows operacinėms sistemoms
2) Sudėtinga architektūra, surišta su Microsoft COM objektine technologija
3) "Vienos firmos kūrinys" – nederinamas su kitų kompiuterių sistemų kūrėjais, nėra tobulinamas iš išorės, licencijavimo tvarka – komercinė.

Šis standartas labiausiai atitinka komercinių kompiuterinių žaidimų kūrėjų poreikius, nes žaidimų industrija kuria savo gaminius beveik išimtinai MS Windows operacinėms sistemoms.

Open Graphic Library (OpenGL), (šiuo metu galutinė versija 1.5, yra projektas versijai 2.0)

Privalumai:

1) Egzistuoja atviro kodo platforma (standarto realizatoriams)
2) Palaikoma grafinių stočių gamintojų
3) Yra multiplatforminė – tinka beveik visoms kompiuterių architektūroms ir operacinėms sistemoms
4) Yra tobulinama ir standartizuojama visų realizatorių konsorciumo OpenGL ARB, atvira pasiūlymams "iš išorės", programų kūrėjų bendruomenei

Trūkumai:

1) Tobulinama per išplėtimų mechanizmą, kuris gana sudėtingas
2) Buvo pradėta kurti UNIX sistemoms, paveldėjo kai kuriuos architektūrinius trūkumus
3) Ilgai trunkanti standartizavimo procedūra

Šis standartas yra labai populiarus nepriklausomų programinės įrangos kūrėjų tarpe. Sukurtas firmos "Silicon Graphics Inc.", iš pradžių naudotas tik šios firmos grafinėms stotims, vėliau buvo paskelbtas atviru standartu, reguliuojamu OpenGL ARB standartizacijos komiteto. Šiuo metu naujos architektūros projektą (versija 2.0) kuria ir tobulina firma "3D Labs Inc.".

Atsižvelgdamas į licecijavimo pobūdį ir multiplatforminės architektūros palaikymą planetinių modelių sistemos kūrimui pasirinkau OpenGL standartą ir jo realizaciją MS Windows sistemoje. Perspektyvoje numatomas sistemos perkėlims į Linux operacinę sistemą.

4. Programos vartotojo sąsają (interfeisą) palaikančios bibliotekos

Kuriant planetinio kūno modeliavimo sistemą iškilo poreikis pasirinkti vartotojo sąsajos palaikymo biblioteką. Šiai bibliotekai taikomi reikalavimai:

- Ji turi būti pagal galimybes multiplatforminė - paklaikyti ne tik MS Windows, bet ir Linux operacinę sistemą, pageidautina ir kitas
- Užtikrinti standartinės grafinės vartotojo sąsajos palaikymą, turėti visus standartinius valdymo elementus
- Būti licecijuojama nekomercine tvarka

Kadangi projektas buvo numatomas atlikti C++ programavimo kalba, teko rinktis iš didelio kiekio rinkoje esančių tokio tipo bibliotekų. Pradinis pasirinkimas buvo šia kalba parašytos laisvai platinamos bibliotekos GLUT (*angl* . graphic library utility tolkit), bibliotekos autorius Mark Kilgard (šiuo metu dirbantis "Nvidia Inc.") ir ją praplečianti GLUI biblioteka (autorius Paul Rademacher). Jau pradėjus programavimo darbus paaiškėjo, kad reikalinga realizuoti duomenų struktūras XML kalbos pagrindu, jas integruojant su vartotojo sąsajos elementais. XML duomenų aprašymo standartas reikalinga norint praplėsti modeliavimo sistemos funkcionalumą ir padaryti programos kodą universaliu. Minėtos bibliotekos panašių funkcijų neturi, renkantis pakartotinai vartotojo bibliotekos sąsajos pasirinkimas buvo firmos Trolltech biblioteka 'Qt", kurios pagrindu realizuota žinoma grafinė Linux OS vartotojo aplinka KDE. Tačiau baigiant programavimo darbus firma Trolltech paskelbė komercializuojanti

Windows aplinkai skirtą bibliotekos variantą, todėl modeliavimo sistema veikia susipažinimui skirtos versijos (angl. Evaluation version) pagrindu. Linux OS variantas yra platinamas pagal laisvo platinimo GNU licenciją.

5. Programinės įrangos projektas

5.1 Įvadas

Projekto "Planetinių kūnų modeliavimo algoritmų tyrimas" programinė įranga yra interaktyvi grafinė programa, leidžianti vartotojui įvestų duomenų pagalba formuoti planetinio kūno modelio paviršių. Suformuotas paviršius vizualizuojamas peržiūros režime, vartotojas gali keisti stebėjimo taško padėtį ir stebėti planetinio kūno modelį su dinaminio vaizdo detališkumo keitimo galimybe.

Projekto užsakovas – Daugų technologijos ir verslo mokykla. Programa bus naudojama 10 – 11 klasių informatikos ir fizikos kurso dalies, nagrinėjančios astronomijos pagrindus, pamokose. Programos vartotojai bus šių klasių moksleiviai. Projekto vykdytojas yra šio darbo autorius.

5.2 Projekto tikslai

1) Programinės įrangos pagalba leisti vartotojams nesudėtingu būdu formuoti planetinio kūno modelį;
2) Realizuoti modelio vizualizacijos technologijas demonstraciniame peržiūros režime, supažindinant vartotojus su dinaminio vaizdo detališkumo keitimo technologija.

5.3 Projekto apžvalga

Programinė įranga priima iš vartotojo šiuos pradinius duomenis:

- planetos radiusą;
- bazinio kubo sieneles padengiančius aukščių masyvus (.jpg failus).
- modelių tekstūras (.jpg failus).

Programinė įranga realizuoja šias grafinio apdorojimo funkcijas:

- bazinio kubo formavimas pagal duomenis iš konfigūracijos failų;
- vaizdo kameros pozicijos inicializacija;
- aukščių masyvų vaizdavimas;
- aukščių masyvų sferinis projektavimas

- detalizacijos lygių grafo (kvadromedžio) suformavimas
- modelio daugiakampių srauto, perduodamo trimačio vaizdo spartintuvui, optimizacija

Programinė įranga taip pat realizuoja šias pagalbines funkcijas:

- modelio failo išsaugojimas (.xml failai)
- modelio failo atidarymas ir įkrova
- planetos apšvietimo modeliavimas

Programinė įranga išveda šiuos duomenis:

- vaizdo formavimo posistemės darbiniai parametrai
- vizualizuotas modelis stebėjimo taško judėjimo režime.

5.4 Pagrindinės programinės įrangos funkcijos

Vartotojo sąsaja

Modeliavimo programos vartotojo sąsaja užtikrina pradinių duomenų įvedimo priėmimą iš vartotojo, leidžia sąsajos elementų pagalba pasirinkti planetinį objektą, kurį stebėsime, o taip pat leidžia keisti programos darbinius parametrus. Programa leidžia keisti planetinio kūno detališkumo lygį, perjungti tekstūruoto ir karkasinio modelio režimu, įjungti ir išjungti nematomų daugiakampių atkirtimo algoritmą.

Duomenų apdorojimas

Programinė įranga negeneruoja aukščių masyvų, jie kuriami trečiųjų firmų sukurtais aukščio masyvų redaktoriais. Programos duomenų apdorojimo mechanizmas iš aukščio masyvų generuoja vientisą modelio struktūrą, sukuria detalizacijos grafą (kvadromedį) ir atlieka aukščių masyvų viršūnių sferinį projektavimą. Iš viršūnių sudaromi daugiakampiai, kuriems skaičiuojamas apšvietimas, atliekamas tekstūravimas.

Rezultatų išvedimas

Programos suformuotas modelis gali būti pagal poreikį įkrautas redaguojant konfigūracijos failo parametrus. Vizualizuotas modelis vaizduojamas programos lange. Informacinėje eilutėje peržiūros režime pateikiami dinaminiai programos darbo parametrai.

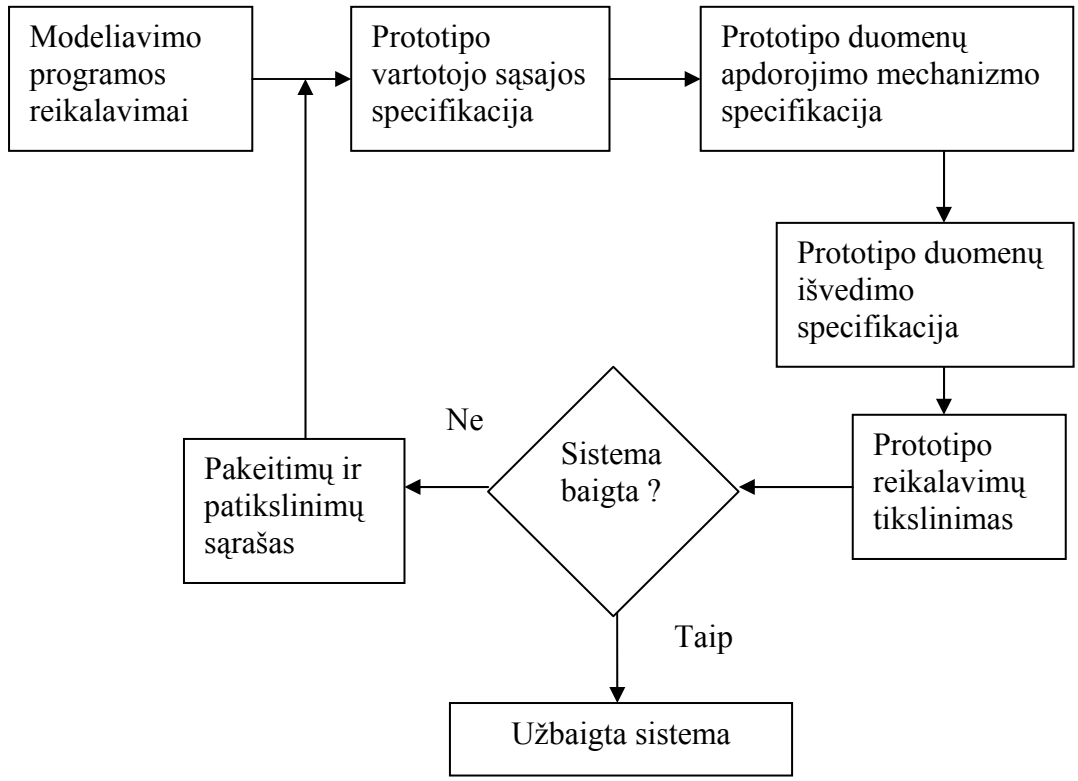
5.5 Vykdymas ir palaikančios aplinkos

Programinė įranga skirta vykdyti Microsoft Windows 9x / 2000 / XP operacinėse sistemose. Programos moduliai projektuojami ir aprašomi Microsoft Visual C++ 6 / 7 programavimo terpėse. Moduliuose kodas atitinka ANSI C++ standarto reikalavimus, todėl perspektyvoje gali būti perkeltas ir su minimaliais pakeitimais kompiliuojamas kitose programavimo terpėse, tarp jų ir kitoms operacinėms sistemoms (pavyzdžiui, GNU gcc kompiliatoriui operacinėje sistemoje Linux). Programinėje įrangoje panaudota grafikos išvedimo sąsaja OpenGL yra multiplatforminė. Ta pačia savybe pasižymi ir sąsajos palaikymo biblioteka Trolltech Qt (www.trolltech.com). Ši biblioteka palaiko vartotojo sąsajos grafinius elementus, jų integraciją su XML duomenų failais.

5.6 Valdymas ir techniniai apribojimai

Magistrinio darbo „Planetinio kūno modeliavimo algoritmų tyrimas“ programinės įrangos projekto įvykdymo terminas yra 2003-12-01.

Projekto vykdymui pasirinktas laipsniško sistemos kūrimo modelis:



Pav. 10 Sistemos laipsniško kūrimo modelis

5.7 Projekto resursai

Projekto autorius ir vykdytojas yra vienas asmuo, jis atlieka šiuos darbus:

1) MS Visual C++ / OpenGL programuotojas;
2) Vartotojo dokumentacijos kūrėjas;
3) Beta testuotojas;

Projektavimui nereikalinga speciali sistemų kūrimo įranga, programa naudoja bendrus personalinio kompiuterio resursus ir serijinės gamybos programinę įrangą.

Projektavimui reikalinga aparatinė įranga:

- 1 personalinis kompiuteris(Celeron 500 MHz, RAM 128 MB, 3D spartintuvas su 16 MB videoatmintine, HDD 10 GB, 1 įrašantis CD-ROM įrenginys).
- Programinė įranga:
    1) MS Windows 98 SE (1 licencija);
    2) MS Visual C++ 6 (1 licencija);
    3) MS Windows 2000 (1 licencija);
    4) MS Windows XP Professional (1 licencija);
    5) MS Office XP (1 licencija);

5.8 Projekto kaštai

Projekto kaštus sudaro projekto kūrimo laikas, įvertintas pinigine išraiška (pritaikius programuotojo darbo įkainį). Iš viso projektui numatyta skirti 280 darbo valandų, valandinis įkainis – 10 Lt. Kaštų piniginė išraiška: 280 val. x 10 Lt = 2800 Lt

5.9 Projekto rizikos faktoriai

Pagrindiniai projekto rizikos faktoriai yra šie:

Rizikos faktorius	Įvykimo tikimybė	Įtaka
Aparatūriniai gedimai	10%	Nedidelė
Projektavimo programinės įrangos klaidos	10%	Nedidelė
Reikalavimų pasikeitimas	20%	Rizika toleruotina
Nukrypimas nuo inžinerinių standartų	30%	Rizika toleruotina
Pavėluotas pristatymas	30%	Kritiška
Blogas kodo komentavimas	10%	Nedidelė

1 Lentelė. Rizikos faktorių įvertinimas

5.10 Projekto eiga

Veiksmai:

1) Planavimas/projektavimas
2) Rizikos analizė
3) Programavimas
4) Testavimas
5) Pristatymas vartotojui ir gynimui

Uždaviniai:

1) Reikalavimų nustatymas
2) Sąsajos projektavimas
3) Algoritmų programavimas
4) Pagalbos sistemos projektavimas
5) Testavimas

5.11 Projekto sudėtis

Dokumentacija:

1) Reikalavimų specifikacija
2) Projekto planas
3) Sistemos specifikacija
4) Vartotojo aprašymas
5) Testavimo planas
6) Projekto pristatymo planas

Kodas:

1) Komponentų moduliai
2) Programos karkasas
3) Pilnas produktas

6. Reikalavimų specifikacija

6.1 Vartotojo reikalavimai

6.1.1 Funkciniai reikalavimai

Programa turi:

1) Sukurti planetos modelį kompiuterinės grafikos pagalba pagal konfigūracijos faile nurodytus parametrus;
2) Nuosekliai didinti (mažinti) planetos paviršiaus detalizacijos lygį artėjant (tolstant) stebėjimo taškui planetos paviršiaus atžvilgiu;
3) Rodyti informaciją apie programos darbo greitį ir naudojamus vaizdo posistemės resursus;
4) Leisti vartotojui keisti peržiūros režimus algoritmų darbo vaizdumui užtikrinti;

Programa turėtų:

1) Demonstruoti daugiakampių tekstūravimo efektus vizualizacijos realistiškumo priartinimui prie realaus planetos paviršiaus detalizacijos kitimo.

Programa turėtų generuoti vaizdą bent 5 kadrų per sekundę greičiu – esant lėtesniam generavimo greičiui realumo įspūdis dingsta. Bendras programos kadrų generavimo našumas esant minimaliems aparatūriniams resursams turėtų būti apie 10 kadrų per sekundę, esant optimaliems – 25-30 kadrų per sekundę. Programos kadrų generavimo greitis yra kritinis sistemos parametras, apsprendžiantis jos darbo galimybę.

Kadangi programa demonstruoja algoritmų darbą, dalis ekrano peržiūros režime turi būti skirta darbinių parametrų išvedimui. Vartotojo sąsaja – grafinė, atitinkanti GUI standarto reikalavimus. Peržiūros režime vartotojo sąsaja komandinė, komandos realizuotos atitinkamais klaviatūros klavišais.

6.1.1 Nefunkciniai reikalavimai

Minimalūs reikalavimai kompiuteriui:

Celeron / Duron 500 MHz procesorius, 128 MB RAM, 3D spartintuvas su 16 MB videoatmintine, 20 MB laisvos vietos HDD, operacinė sistema - Windows 95 OSR2.1/ 98 / Me / 2000 / XP.

Optimalus kompiuteris:

Pentium IV / Athlon XP 1400 MHz, 256 MB RAM, 3D spartintuvas Nvidia GeForce FX 5600 arba ATI Radeon 9500 su 128 MB videoatmintinės.

6.2 Programos architektūros aprašymas

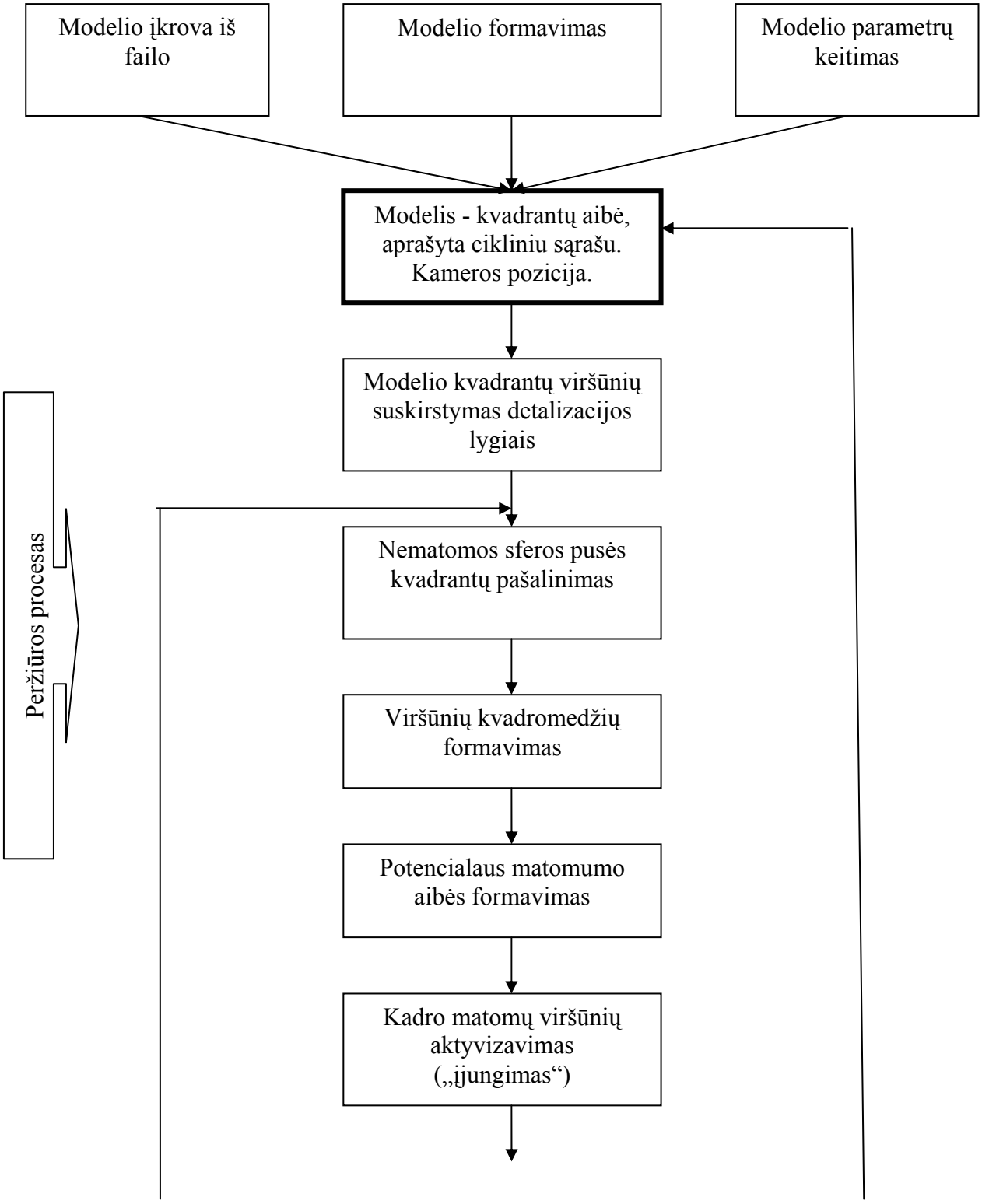
Programa darbo metu sprendžia šiuos uždavinius:

1) Sferos generavimas.
   Pagal vartotojo įvestus duomenis konfigūraciniame faile programa generuoja bazinį kubą.
2) Aukščio masyvų projektavimas ant kvadrantų.
   Masteliavimo būdu aukščių masyvai "priklijuojami" prie bazinio kubo kvadrantų.
3) Sferinio projektavimo būdu aukščių masyvų viršūnės projektuojamos ant sferos.
4) Planetoido modelio erdvinių koordinačių skaičiavimas ir daugiakampių modelio sudarymas. Programa prieš inicializuodama peržiūros režimą apskaičiuoja modelio viršūnių erdvines koordinates. Sudaromi detalizacijos grafai (kvadromedžiai).
5) Pradinio detalizacijos lygio parinkimas ir nematomų paviršių pašalinimas.
   Atsižvelgdama į stebėjimo taško padėtį, programa nustato pradinį paviršiaus detalizacijos lygį ir pašalina nematomus sferos daugiakampius (*angl.* backface culling). Tai liečia tik iš stebėjimo taško nematomą sferos pusę – stebėjimo taško judėjimo metu reljefo užstojami daugiakampiai šalinami dinamiškai OpenGL grafinės bibliotekos funkcijų pagalba ir iš anksto negali būti pašalinti šiame etape.

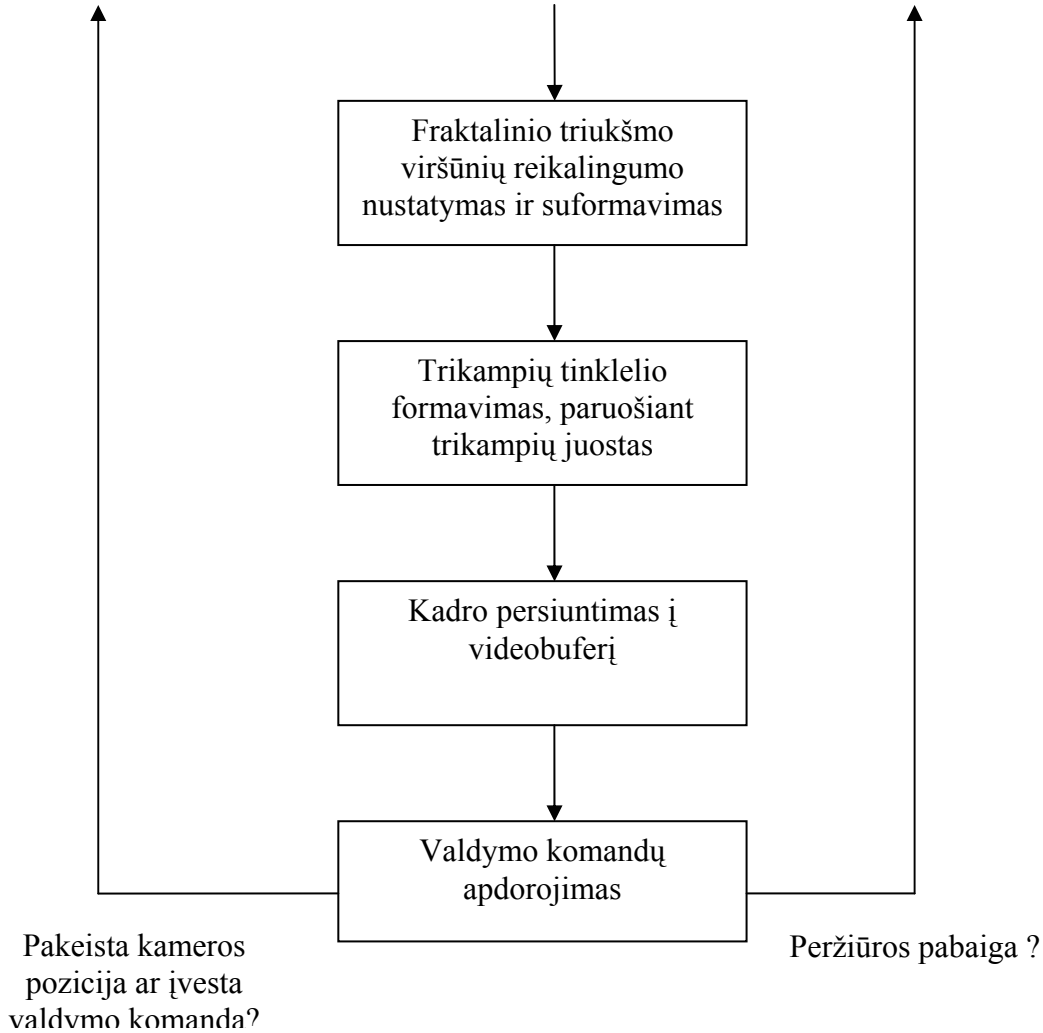
Tolesni uždaviniai sprendžiami vaizdo formavimo (angl. *rendering*) metu:

5) Procedūrinė detalizacija.
   Pagal stebėjimo taško atstumą nuo paviršiaus programa nustato, ar reikalinga generuoti papildomas (dinamines) viršūnes vaizdo realistiškumui užtikrinti. Taip pat reikalinga iš anksto pakrauti į atmintį potencialiai greitai būsiančias reikalingas daugiakampių viršūnes (*angl*. vertex cashing).
6) Kadro buferio duomenų masyvų formavimas ir jų siuntimo į 3D videoadapterį optimizavimas, bazinio ir video procesorių darbo paralelizavimas.
   Kadangi programa turi pasiekti didelį kadrų formavimo greitį, šiame etape reikalinga paruošti duomenis taip, kad jie kuo greičiau pasiektų kadro buferį. Tai padeda padaryti OpenGL grafinės bibliotekos išplėtimai – kompiliuoti viršūnių masyvai (angl. *CVA – compiled vertex arrays*) ir atskiru atveju "Nvidia" gamybos 3D videoadapteriams Geforce – viršūnių masyvų intervalai (angl. *VAR – vertex array range*). Jie leidžia programuotojui valdyti AGP magistralės atmintinės darbą.

6.3 Programos algoritmo vykdymo schema



Tęsinys kitame puslapyje



Pav. 11 Programos algoritmo schema

6.4 Rizika ir apribojimai, konkurencija

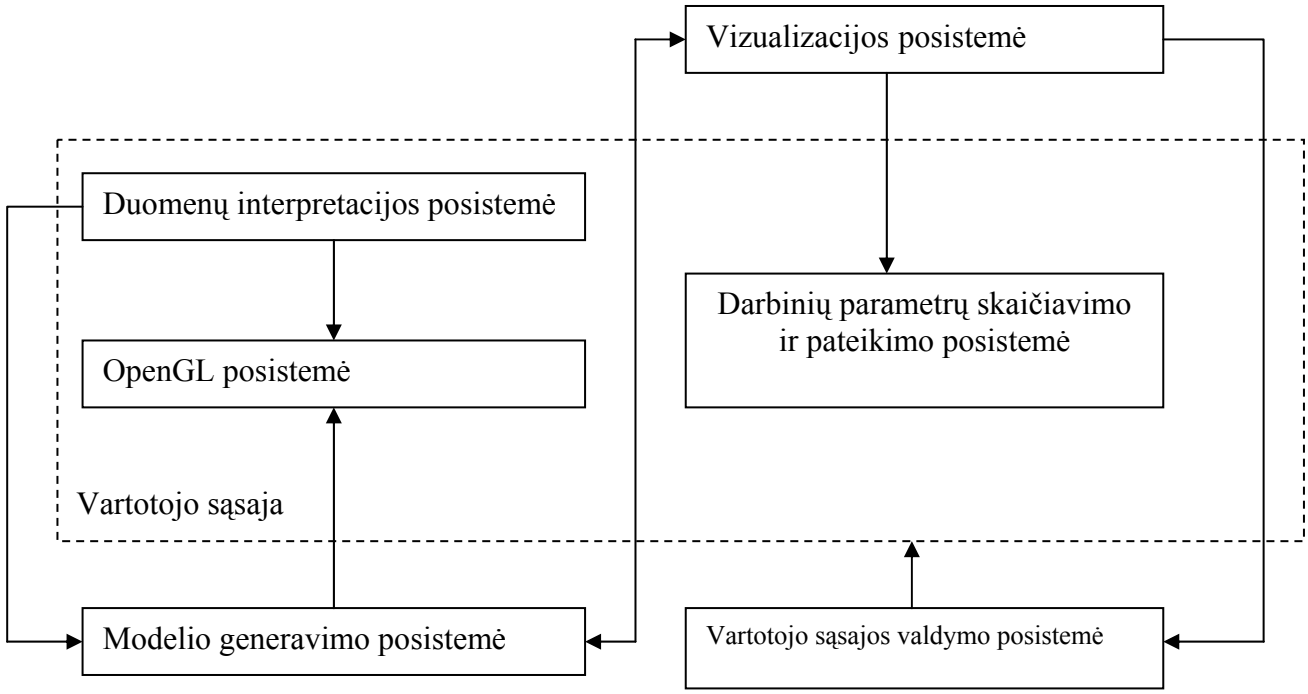
Dėl vartotojų nepatyrimo (programa naudosis mokyklos 10-11 kl. moksleiviai) ir palyginti nedidelio išprusimo programa gali tapti mažai naudojama arba algoritmų darbo rezultatų pateikimas gali būti per sudėtingas. Kadangi programa yra nekomercinė, autorius rizikuoja prarasti profesinę reputaciją.

Rinkoje panašaus profilio laisvai platinamų programų yra pakankamai daug. Dėl nedidelio programos autoriaus patyrimo yra nemaža rizika, kad vartotojas pasirinks tobulesnį konkurentų projekto variantą.

7. Sistemos specifikacija

7.1 Sistemos struktūros specifikacija

Programinė įranga yra grafinis demonstracinis įrankis, darbui su kuriuo vartotojui nėra reikalingi specifiniai darbo įgūdžiai. Sistemą sudaro keletas tarpusavyje bendraujančių darbinių posistemių, pavaizduotų žemiau pateiktame paveiksle.



Pav. 12 Sistemą sudarančių posistemių modelis

OpenGL posistemė: atsakinga už OpenGL grafinės sistemos inicializavimą ir valdymą.

Duomenų interpretacijos posistemė: atsakinga už duomenų nuskaitymą iš failų ir jų interpretaciją.

Modelio generavimo posistemė: atsakinga už modelio generavimą pagal turimus duomenis.

Vizualizacijos posistemė: atsakinga už vaizdo ekrane formavimą

Darbinių parametrų skaičiavimo ir pateikimo posistemė: atsakinga už sistemos darbinių parametrų apskaičiavimą ir pateikimą ekrane.

Vartotojo sąsajos valdymo posistemė: atsakinga už vartotojo įvedamų komandų priėmimą ir apdorojimą

7.2 Sistemos reikalavimų specifikacija

Įvertinus vartotojų reikalavimus bei įrangos analizės rezultatus, reikalavimai sistemai.yra šie:

- Leisti vartotojui įvesti kuriamo modelio parametrus konfigūraciniame ir planetų parametrų failuose
- Vaizduoti planetinio kūno modelį trimatėje erdvėje
- Keisti planetinio kūno paviršiaus detalizacijos lygį dinamiškai, priklausomai nuo atstumo tarp stebėjimo taško ir planetinio kūno paviršiaus
- Leisti vartotojui keisti stebėjimo taško padėtį
- Pateikti vartotojui informaciją apie vizualizacijos posistemės darbinius parametrus
- Leisti vartotojui keisti modeliuojamo planetinio kūno vizualizacijos režimus
- Modeliuoti planetinio kūno apšvietimą

Reikalavimai sistemos patikimumui ir kokybei:

- Programa turi pateikti kokybiškai detalizuotą vaizdą.
- Duomenų struktūros turi būti nesudėtingos.
- Dirbant sistemai turi būti pateikiama tiksli informacija apie darbo parametrus.
- Sistema turi greitai reaguoti į vartotojo valdymo komandas.

Projekto realizavimo būdai ir priemonės:

- Programavimo kalba: C++;
  Lanksti, daug bibliotekų ir didelę elementinę bei pagalbos bazę turinti programavimo kalba.
- Programavimo terpė: Microsoft Visual C++;
  Patogi ir plačiai naudojama projektavimo – programavimo terpė. Sistemos kūrėjai turi didelį darbo šioje aplinkoje patyrimą, todėl jiems nereikės iš naujo prisitaikyti prie kokios nors naujos programavimo aplinkos.
- Grafinė sistema: OpenGL;
  Gerai pritaikyta grafinio modeliavimo sistemoms kurti. Patogus funkcionalumas. Sistemos kūrėjai taipogi yra neblogai susipažinę su šia sistema.
- Operacinė sistema: Win9x, Win2000, WinXP;
  Tai plačiai vietinėje rinkoje naudojamos operacinės sistemos. Perspektyvoje numatomas perkėlimas į Linux operacinę sistemą.
- Prototipo darymas: MS Visual C++

MS Visual C++ pagalba sudaromas objektinis modelis bei sistemos architektūriniai sprendimai. Sistema pasirinkta dar ir todėl, kad programuotojas gerai susipažinęs su ja.

Išvados: 1) Sistemos duomenų aprašymas, dėl pačios sistemos objektinio projektavimo metodologijos ypatybių, yra lengvai realizuojamas XML standarto duomenų struktūromis.

2) Objektinė modeliavimo sistemos architektūra leidžia plėsti sistemos funkcionalumą pagal ateityje iškilsiančius poreikius.

7.3 Duomenų struktūrų specifikacija

Duomenų struktūra pasirenkama remiantis objektiniu modeliu ir informacijos reikalingos išsaugoti duomenims poreikiu. Duomenys failuose aprašomi XML duomenų struktūrų aprašymo kalba, kurios palaikymas realizuotas Trolltech Qt grafinės vartotojo sąsajos bibliotekoje, o taip pat atskirai sukurtoje duomenų palaikymo klasėje.

Konfigūracinio failo duomenų struktūra:

```
- <PARAMS>
  <PLAYER FILENAME="Data/xml/taskas.xml" PARENT="Moon" />
// Čia nurodomas stebėjimo taško lokacijos vietos failas
  <PLANETLIST FILENAME="Data/xml/planeta.xml" />
// Nurodomas kuriamų planetinių kūnų sistemos objektų sąrašo failas
  <DATE YEAR="2248" MONTH="2" DAY="20" HOUR="20" MINUTE="48"
    SECOND="30" aJULIANDATE="2451454.8" />
// Sistema palaiko Julijaus kalendorių (perspektyvoje bus plečiamas
    funkcionalumas)
  <FONT FILENAME="Data/fontai.bmp" NUMX="16" NUMY="16" CWIDTH="32"
    CHEIGHT="32" CSPACE="16" />
// Nurodomas OpenGL grafinių šriftų failas (lotyniškas raidynas)
  </PARAMS>
```

Planetinių kūnų sistemos objektų sąrašo failo duomenų struktūra:

<SOLARSYSTEM NAME="Sol">

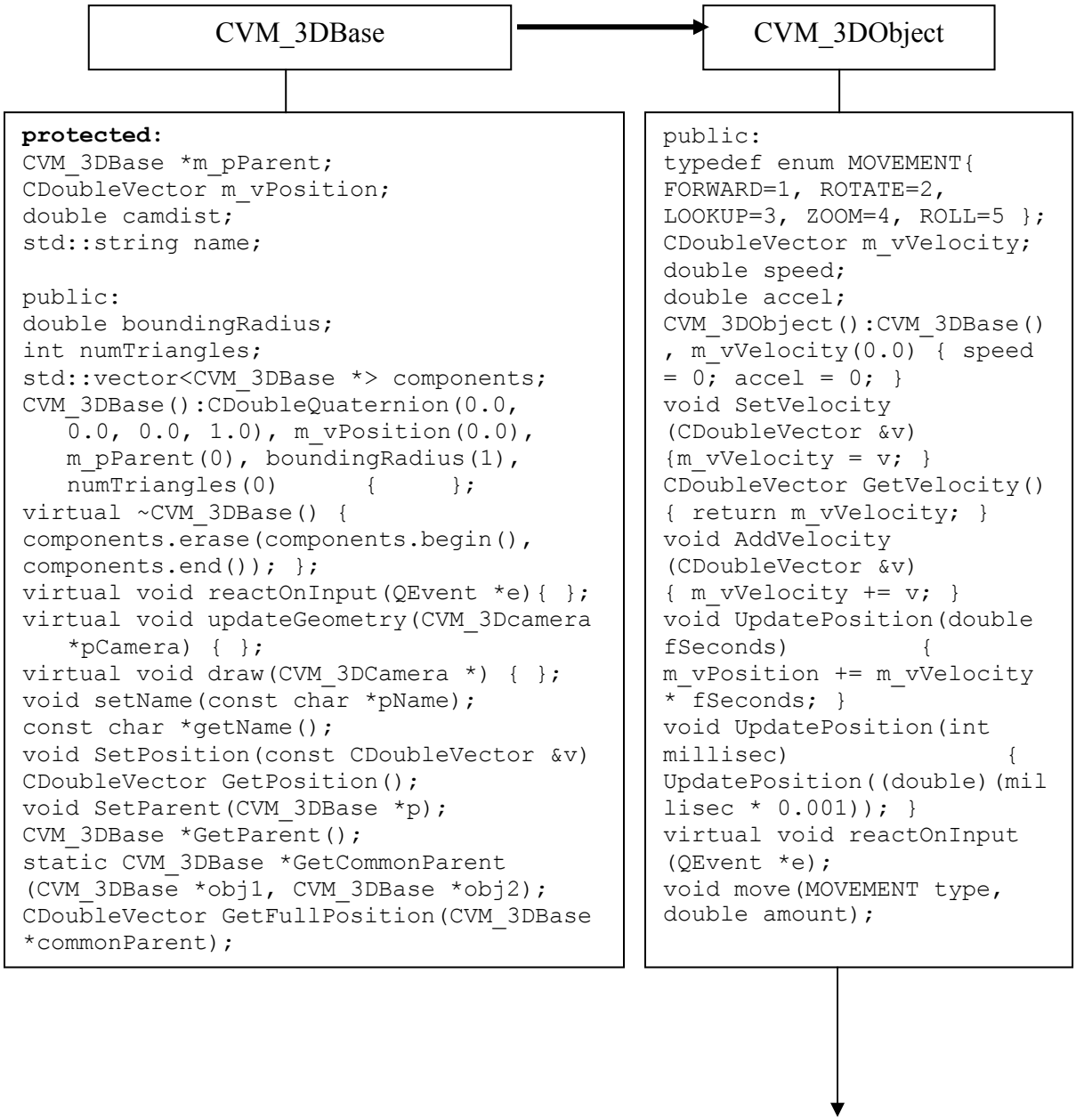
```
  <STAR NAME="Saule" TYPE="SUN" RADIUS="695000000.0"
HEIGHT="100000.0" ORBIT="0" YEAR_DURATION="0"
DAY_DURATION="0">
    <LIGHT NUMBER="0">
      <COLOR RED="1.0" GREEN="1.0" BLUE="1.0"/>
    </LIGHT>
    <CORONA aTEXTURE="Data/karuna.bmp">
      <COLOR_HOT RED="1.0" GREEN="1.0" BLUE="0.3"/>
      <COLOR_COLD RED="1.0" GREEN="0.5" BLUE="0.3"/>
    </CORONA>
    <SKY ATM_RANGE="100000000.0">
      <COLOR_DN RED="1.0" GREEN="1.0" BLUE="0.3"/>
      <COLOR_ME RED="1.0" GREEN="0.5" BLUE="0.3"/>
    </SKY>
  </STAR>
```

Aukščiau pateiktoje sekcijoje aprašomi planetinių kūnų sistemos šviesos šaltinio (žvaigždės) parametrai

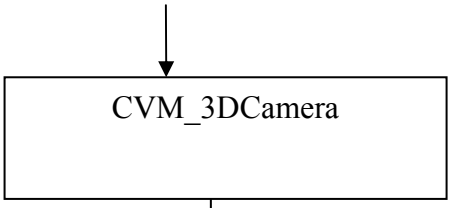
```
<PLANET NAME="Marsas" TYPE="MARS" RADIUS="3397000.0"
HEIGHT="25000.0" ORBIT="227940000.0e3" YEAR_DURATION="687"
DAY_DURATION="24.5" TEXTURE="Data/planetmaps/mars.jpg"
HEIGHTFIELD="Data/planetmaps/marsbump1k.jpg">
    <SKY ATM_RANGE="80000.0">
      <COLOR_DN RED="1.0" GREEN="0.3" BLUE="0.0"/>
      <COLOR_ME RED="0.5" GREEN="0.7" BLUE="0.0"/>
    </SKY>
  </PLANET>
```

Aukščiau pateiktoje sekcijoje aprašomi planetinio kūno parametrai. Tokių sekcijų faile gali būti iki 255. Atstumų dydžiai nurodomi metrais, spalviniai – intervale 0.0 – 1.0 pagal OpenGL standartą.

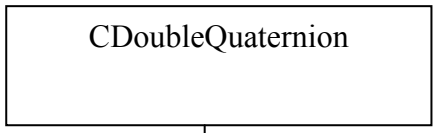
7.4. Pagrindinių projekto klasių paveldėjimo ir kompozicijos schema



Tęsinys kitame puslapyje



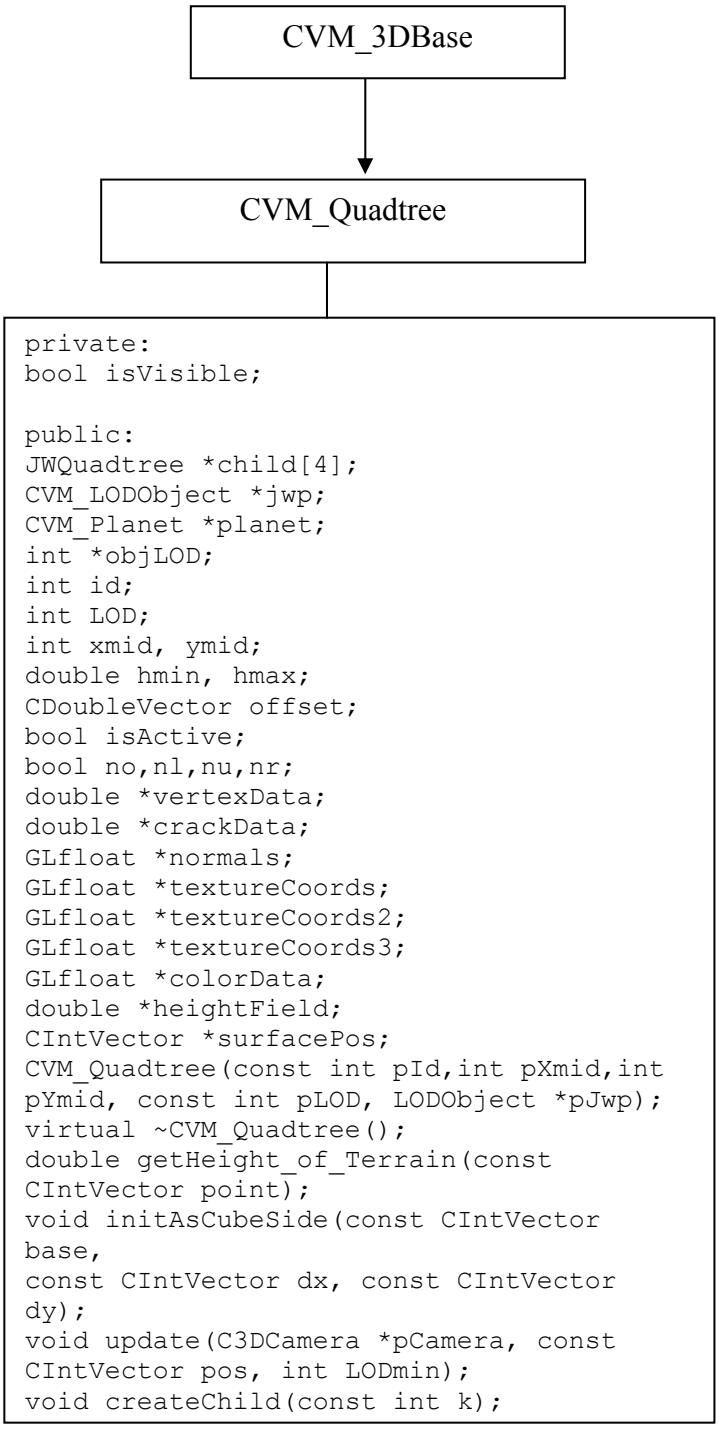

```
protected:
bool clipping;
CDoubleVector base;
CDoubleVector view;
CDoubleVector up;
CDoubleVector right;
CDoubleVector clip[5];

public:
double fovy,fovx;
inline bool isVisible(CVM_3DBase
*obj);
bool isVisible(CVM_3DBase *planet,
CDoubleVector newcenter, double
newboundingRadius);
void initFrustum(double pfovy, double
pfovx, bool pClipping);

private:
static int instancecount;     static
double screen_error;

public:
typedef enum TRIANGULATION {
TRIANGLES=0, TRIANGLE_FAN=1,
TRIANGLE_STRIP=2 };
static TRIANGULATION tmode;
static int grid_size , gs;
static int *RegularIndexList[4];
static int *indexList[32];
static int indexLenght[32];
static int *fanindexList[32];
static int fanindexLenght[32];
static int *stripindexList[32];
static int stripindexLenght[32];
static float *colorBlend[16];
JWNoise jwfrac;
char center[80];
CIntVector lastpos;
int HF_width, HF_height;
int HF_size;
int HF_LOD;
bool fineLOD;
int LODalt;
int LOD_diff;
std::vector<JWStaticObject *>
objects;
JWStaticObject *INVALID;
double H_RANGE;
int H_LOG;
double axis_angle;
double year_length;
```

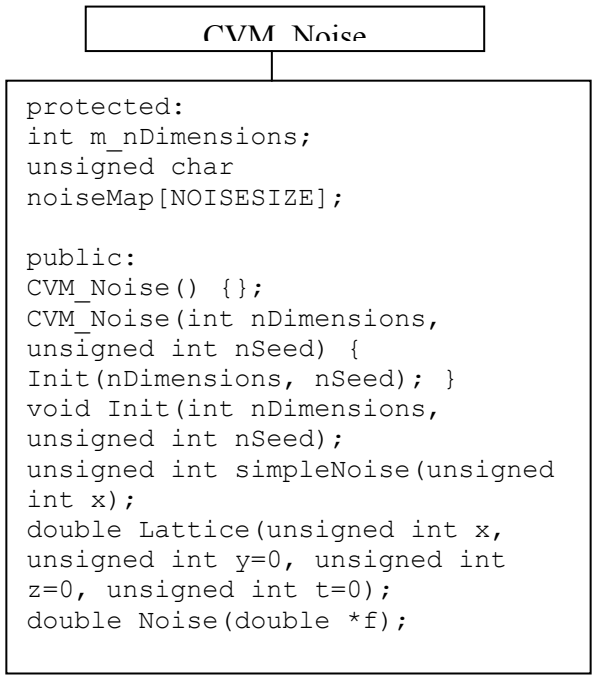


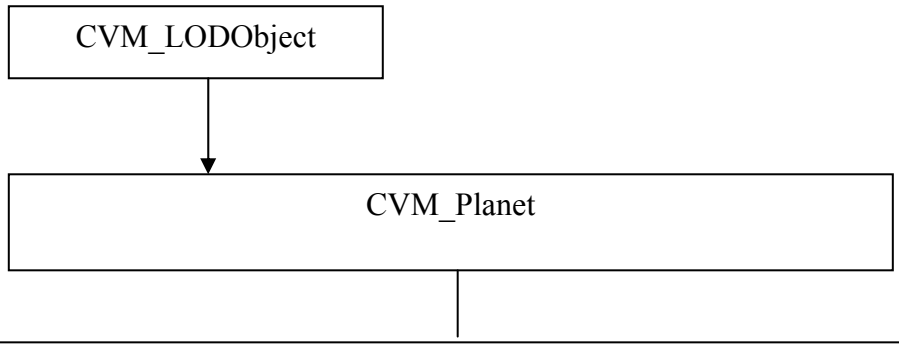

```
public:
double x, y, z, w;
CDoubleQuaternion(){}
CDoubleQuaternion(const double a,
const double b, const double c,
const double d){ x = a; y = b; z =
c; w = d; }
CDoubleQuaternion(const
CDoubleVector &v, const double f){
SetAxisAngle(v, f); }
CDoubleQuaternion(const
CDoubleVector &v) { *this = v; }
CDoubleQuaternion(const
CDoubleQuaternion &q) { *this = q; }
CDoubleQuaternion(const
CDoubleMatrix &m) { *this = m; }
CDoubleQuaternion(const double *p)
{ *this = p; }
operator double*(){ return &x; }
double &operator[](const int n) {
return (&x)[n]; }
double operator[](const int n) const
{ return (&x)[n]; }
CDoubleQuaternion operator-() const
{ return CDoubleQuaternion(-x, -y, -
z, -w); }
void operator=(const CDoubleVector
&v){ x = v.x; y = v.y; z = v.z; w =
0; }
void operator=(const
CDoubleQuaternion &q){ x = q.x; y =
q.y; z = q.z; w = q.w; }
void operator=(const CDoubleMatrix
&m);
void operator=(const double *p) { x
= p[0]; y = p[1]; z = p[2]; w =
p[3]; }
CDoubleQuaternion operator+(const
double f) const   { return
CDoubleQuaternion(x+f, y+f, z+f,
w+f); }
CDoubleQuaternion operator-(const
double f) const   { return
CDoubleQuaternion(x-f, y-f, z-f, w-
f); }
CDoubleQuaternion operator*(const
double f) const   { return
CDoubleQuaternion(x*f, y*f, z*f,
w*f); }
CDoubleQuaternion operator/(const
double f) const   { return
CDoubleQuaternion(x/f, y/f, z/f,
```

CDoubleMatrix

```
public:
union
  {
      double f1[16];
      double f2[4][4];
  };
CDoubleMatrix();
CDoubleMatrix(const double f);
CDoubleMatrix(const double *pf);
CDoubleMatrix(const CDoubleQuaternion &q);
void ZeroMatrix();
void IdentityMatrix();
operator double*() { return f1; }
double &operator[](const int n) { return f1[n]; }
double &operator()(const int i, const int j){ return f2[i][j]; }
double operator[](const int n) const { return f1[n]; }
double operator()(const int i, const int j) const{ return f2[i][j]; }
void operator=(const double f)
void operator=(const double *pf);
void operator=(const CDoubleQuaternion &q);
CDoubleMatrix operator*(const CDoubleMatrix &m) const;
void operator*=(const CDoubleMatrix &m)
CDoubleVector operator*(const CDoubleVector &v) const { return
TransformVector(v); }
CDoubleVector TransformVector(const CDoubleVector &v) const;
CDoubleVector TransformNormal(const CDoubleVector &v) const;
void TranslateMatrix(const double x, const double y, const double z);
void TranslateMatrix(const double *pf);
void Translate(const double x, const double y, const double z);
void Translate(const double *pf);
void ScaleMatrix(const double x, const double y, const double z);
void ScaleMatrix(const double *pf);
void Scale(const double x, const double y, const double z);
void Scale(const double *pf);
void RotateXMatrix(const double fRadians);
void RotateX(const double fRadians);
void RotateY(const double fRadians);
void RotateYMatrix(const double fRadians);
void RotateZMatrix(const double fRadians);
void RotateZ(const double fRadians);
void RotateMatrix(const CDoubleVector &v, const double f);
void Rotate(const CDoubleVector &v, const double f);
void ModelMatrix(const CDoubleQuaternion &q, const CDoubleVector
&vFrom);
void ViewMatrix(const CDoubleQuaternion &q, const CDoubleVector
&vFrom);
void ViewMatrix(const CDoubleVector &vFrom, const CDoubleVector
&vView, const CDoubleVector &vUp, const CDoubleVector &vRight);
void ViewMatrix(const CDoubleVector &vFrom, const CDoubleVector &vAt,
const CDoubleVector &vUp);
void ProjectionMatrix(const double fNear, const double fFar, const
double fFOV, const double fAspect);
void Transpose();
```


CVM_3DBase
CVM_Quadtree
private:
bool isVisible;
public:
JWQuadtree *child[4];
CVM_LODObject *jwp;
CVM_Planet *planet;
int *objLOD;
int id;
int LOD;
int xmid, ymid;
double hmin, hmax;
CDoubleVector offset;
bool isActive;
bool no,nl,nu,nr;
double *vertexData;
double *crackData;
GLfloat *normals;
GLfloat *textureCoords;
GLfloat *textureCoords2;
GLfloat *textureCoords3;
GLfloat *colorData;
double *heightField;
CIntVector *surfacePos;
CVM_Quadtree(const int pId,int pXmid,int pYmid, const int pLOD, LODObject *pJwp);
virtual ~CVM_Quadtree();
double getHeight_of_Terrain(const CIntVector point);
void initAsCubeSide(const CIntVector base,
const CIntVector dx, const CIntVector dy);
void update(C3DCamera *pCamera, const CIntVector pos, int LODmin);
void createChild(const int k);


protected:
int m_nDimensions;
unsigned char noiseMap[NOISESIZE];
public:
CVM_Noise() {};
CVM_Noise(int nDimensions, unsigned int nSeed) { Init(nDimensions, nSeed); }
void Init(int nDimensions, unsigned int nSeed);
unsigned int simpleNoise(unsigned int x);
double Lattice(unsigned int x, unsigned int y=0, unsigned int z=0, unsigned int t=0);
double Noise(double *f);




```
private:
static int planetcount;
static TEXTURE *staticTexDetail;
static TEXTURE *staticTexFineDetail
static TEXTURE *staticTexSuperFineDetail;

public:
typedef enum PLANET_TYPE { SUN=1, EARTH=2, MOON=3, MARS=4, GAS=5, WATER=6,
CLOUDS=7 };
PLANET_TYPE style;
CVM_Quadtree *jwq[6];
CIntVector dx_s[6];
CIntVector dy_s[6];
float *HF;
TEXTURE *Tex[6];
TEXTURE *TexDetail;
TEXTURE *TexFineDetail;
TEXTURE *TexSuperFineDetail;
TEXTURE *colorMap;
std::vector<ElevationGrid *> dems;
virtual void init(QDomElement node);
virtual void updateGeometry(C3DCamera *pCamera);
virtual void draw(C3DCamera *camera);
virtual CIntVector CoordinatesToSurfaceVector(const double x, const double y,
const double z,int *c_s, double *dist);
virtual CDoubleVector SurfaceVectorToCoordinates(CIntVector pos, int c_s,
double h);
virtual CDoubleVector SurfaceVectorNormal(CIntVector pos, int c_s);
double get_HeightField_And_RandomOffset(CIntVector point,
int LOD, double h_old);
CFloatVector4 computeColor(const double hf, CIntVector pos=CIntVector(0,0,0));
virtual double getHeight_above_Surface(C3DBase *obj);
void initPlanet(PLANET_TYPE pStyle, double rad, double H_R,
double paxis_angle = 0);
void initTextures(char *tex=0, char *finetex=0, char *superfinetex=0);
void createCubeTexture(const int id, const CIntVector base,
const CIntVector dx, const CIntVector dy);
void initHeightField(char *tex, bool HFisComplete, int pHF_size=256);
float* create_fractal(int MAP);
void loadDEM(QDomElement node);
```

Išvada: Grafinių modeliavimo algoritmų taikymas leidžia pasiekti optimalų kompiuterių grafinio vaizdo posistemės panaudojimą modeliavimo sistemose.

8. Testavimo planas

Kadangi sistemos programuotojas, dokumentuotojas ir testuotojas yra vienas ir tas pats asmuo, o programinės įrangos dydis neviršija 1200 kodo eilučių, projektui bus taikoma supaprastinta testavimo metodika. Kadangi projektavimas atliekamas naudojant objektinį projektavimą, suformuojant esybes (objektų klases) ir nustatant hierarchinius ir sąveikos ryšius, testavimo procesas turi užtikrinti, kad programinę įrangą sudarantys objektiniai komponentai korektiškai dirbtų autonominiu režimu, o atliekant prototipo sudėtinginimą, įvedant naujus komponentus, neiškiltų nesuderinamumo problemų. Komponentų ir karkaso testavimas vykdomi MS Visual C++ integruotoje aplinkoje.

Testavime naudojama programos vykdymas po žingsnį (eilutę) testavimo režime (*angl*. debug mode), stebint komponentų klasių narių reikšmes ir jų sąsajos darbą.

I etapas

Programos karkaso testavimas

Parašytas programos karkaso kodas testuojamas pagal šiuos kriterijus:

1) Programos karkasas turi veikti kaip programa, užtikrinanti bazines paslaugas, kurias teikia operacinė sistema. Pagrindinė paslauga – lango valdymo funkcijos (uždarymas, minimizavimas / maksimizavimas, perkėlimas į kitą ekrano vietą ir lango dydžio keitimas).
2) Karkasas palaiko vartotojo sąsajos bazinius elementus – kontekstinį meniu ir panelį, talpinantį smulkesnius valdymo komponentus.

II etapas

Komponentų testavimas

Pagrindinis komponentas – klasė CVM_GameEngine, realizuojanti visus sistemos modelio formavimo metodus. Šios klasės nariai yra smulkesni komponentai, kurie turi būti nuosekliai įtraukiami į modelio klasę, o po to tikrinami autonomiškai ir sąveikoje su karkasu. Šiame etape netestuojama komponentų ir vartotojo sąsajos valdymo elementų sąveika.

Testuojami šie komponentai (pradedant nuo smulkiausių):

1) Vektoriai (klasės CDoubleVector, CIntVector) – tikrinamas vektoriaus transformacijų metodų darbas, parenkant pradines reikšmes, kurių transformacijų rezultatai žinomi iš anksto.

2) Keturmatis vektorius (klasė CDoubleQuaternion) - tikrinamas vektoriaus transformacijų metodų darbas, parenkant pradines reikšmes, kurių transformacijų rezultatai žinomi.
3) Matrica (klasė CDoubleMatrix) – tikrinamas modelio ir peržiūros matricų transformavimo metodai su pradinėmis reikšmėmis, kurių transformacijų rezultatai žinomi iš anksto.
4) Į modelio klasę įtraukiamas narys – bazinis kubas, aprašytas 8 vietų sveikųjų skaičių masyvu.
    A) Realizuojamas kubo braižymo metodas, jis testuojamas vizualiai, stebint kubo vaizdą ekrano lange.
    B) Realizuojamas kubo viršūnių projektavimo metodas, testuojamas vizualiai pagal įvestų duomenų komplekso projekcijos ekrane rezultatus.

5) Kamera (klasė CVM_ 3DCamera) – tikrinami kameros judinimo orbita ir artyn / tolyn modelio atžvilgiu metodai. Testuojama vizualiai stebint formuojamą vaizdą ekrano lange.
6) Kameros ir modelio sąveikos su karkasu testavimas.
7) Kvadromedis (keturšakis medis – vizualizacijos lygių grafas) (klasė CVM_Quadtree) – testuojama esant tik dviems detalizacijos lygiams su pakankamai mažu (ne daugiau 10 – 15) aukščių masyvo viršūnių skaičiumi. Dėl sudėtingumo ir svarbos šio komponento testavimas skaidomas į smulkesnius etapus:
    A) Aukščio masyvo buferio užpildymo iš failo testavimas – testuojamas užpildymo teisingumas, stebint masyvo reikšmes programos derinimo režime.
    B) Grafo formavimo testavimas – tikrinama su nedideliu viršūnių skaičiumi, parinkus tokias pradines reikšmes, kurių skirsytmas lygiais atliktas rankiniu skaičiavimu ir rezultatai žinomi. Atliekamas rezultatų palyginimas.
    C) Vizualinis lygių keitimosi stebėjimas, siekiant surasti paslėptas klaidas.
8) Kvadromedžių sistemos iš šešių komponentų testavimas – tikrinama vizualiai stebint modelio detalizacijos lygių kitimą esant kameros judėjimui modelio atžvilgiu.
9) Kubo sferinio projektavimo testavimas – vykdomas stebint projektuotą modelį vizualiai, o taip pat lyginant daugiakampių transformuotų viršūnių masyvų reikšmes su rankinio skaičiavimo rezultatais.
10) Daugiakampių šešėliavimo komponento testavimas – tikrinama vizualiai pagal viršūnių sudarytą reljefą ir apšvietimo modelį. Testavimas vizualus, pagal stebėjimų rezultatus galima nustatyti, kuriuose skaičiavimuose įsivėlė klaidos.

III Etapas
Vartotojo sąsajos testavimas

Šio etapo metu testuojami vartotojo sąsajos grafiniai valdymo elementai, o taip pat valdymo komandų iš klaviatūros veikimas. Veikimo teisingumas tikrinamas vizualiai stebint programos reakciją į valdymo elementų įjungimą bei reakciją į klaviatūros komandas. Testavimo etapai:

1) Integruotos derinimo aplinkos pagalba tikrinamos programos nustatomos pradinės kintamųjų, priskirtų valdymo elementams, reikšmės;
2) Vizualiai stebimas vaizdo kitimas programos lange, reaguojant į valdymo elementų įjungimą. Esant neatitikimams specifikacijai arba nekorektiškiems vaizdo pakitimams tikrinamas valdymo elementų kontrolinių kintamųjų reikšmės integruotoje programos derinimo aplinkoje.
3) 2 punkto veiksmai atliekami klaviatūros valdymo komandoms.

Priėmimo testavimas

Priėmimo testavimą atlieka užsakovo parinkta 7 žmonių komanda, kurią sudaro penki potencialūs programos vartotojai – 10 – 11 klasių moksleiviai ir du mokytojai – dalykininkai: fizikos ir informatikos. Komanda supažindinama su programos vartotojo dokumentacija, išbando programą savarankiškai ir kritines pastabas bei pastebėjimus pateikia programos autoriui. Klaidų ištaisymui skiriamas dviejų savaičių laikotarpis, kuriam pasibaigus programa pateikiama užsakovui.

9. Vartotojo aprašymas

Programa yra skirta demonstruoti planetinio kūno modelio kūrimo technologijai, planetos reljefo detalizacijos lygio kitimą priklausomai nuo stebėjimo taško judėjimo ir kompiuterinės grafikos algoritmų veikimui. Programos potencialūs vartotojai – bendrojo lavinimo vidurinių mokyklų ir gimnazijų vyresniųjų klasių moksleiviai, besimokantys fizikos kurso dalies, nagrinėjančios planetų sandaras ir reljefus, taip pat besimokantys informatikos kurso. Programa leidžia išsaugoti modelius failuose ir įkrauti į atmintį programos darbo metu pagal vartotojo poreikį.

Programos darbiniai reikalavimai

Programos darbui reikalingas kompiuteris su Intel ® Celeron™ 500 MHz arba aukštesnio dažnio procesoriumi, 128 MB operatyviosios atmintinės ir 20 MB laisvos vietos pastoviojoje atmintinėje. Būtinas trimačio vaizdo spartintuvas ATI arba Nvidia® firmų grafinio procesoriaus pagrindu, turintis ne mažiau 16 MB videoatmintinės. Programa veikia Microsoft® Windows 98 / Me /2000 / XP operacinių sistemų aplinkoje. OS Windows 95 palaiko programos darbą tik nuo versijos OSR 2.1 .

Programos darbui papildomai reikalingos pagalbinės bibliotekos: IJG JPEG formato palaikymo biblioteka (Interneto svetainė www.ijg.org) , platinama instaliaciniame komplekte. Taip pat reikalinga Trolltech firmos grafinės sąsajos biblioteka Qt, ją parsisiųsti iš Interneto svetainės www.trolltech.com . Programa kaip pradinius duomenis naudoja aukščių masyvus (1024 x 1024 ekrano taškų raiškos .jpg failus), kuriuos galima kurti laisvai platinamais redaktoriais (pvz. GIMP, Interneto svetainė www.gimp.org). Atskirame kataloge talpinamos 256x256 ekrano taškų raiškos tekstūros (taip pat .jpg formate).

Programos platinimas

Programos instaliacinis komplektas platinamas laikmenose, kurių talpa siekia 15 MB. Instaliacijos programa – Nullsoft Installer (www.nullsoft.com). Platinimas laisvas, neteikiamos jokios garantijos.

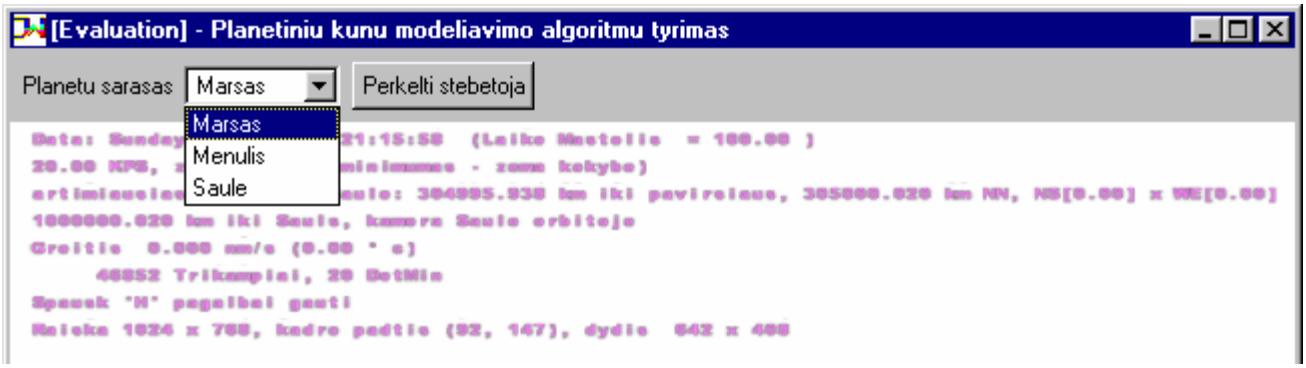
Programos paleidimas

Programa paleidžiama vykdyti standartiniu OS MS Windows būdu: startinio meniu komandos arba sukurtos nuorodos pagalba.

Darbo pradžia ir pradiniai duomenys

Darbas su programa pradedamas užpildant konfigūracinio ir planetinių kūnų sistemos parametrų failus. Pildant pastarąjį būtina užpildyti apšvietimo šaltino sekciją ir nors vieną planetos sekciją (planetų maksimalus skaičius – 255).

Paleidus programą kompiuterio ekrane atsiranda 640x480 ekrano taškų raiškos langas, kuriame yra pasirinkimo panelis ir grafinė sritis su pateiktais sistemos modelio parametrais ir pagalbine informacija. Pelės pagalba vartotojas turi pasirinkti vieną planetą iš pasirinkimo panelyje esančio kombinuoto sąrašo:



Programos valdymo sąsaja

Įvykdžius pasirinkimą iš sąrašo, ekrane piešiamas vaizdas iš 10 km atstumo nuo planetinio kūno. Valdymo klavišų reikšmes vartotojas gali gauti paspaudęs klavišą “H” arba “F1” klaviatūroje.

Valdymo komandos:

- Klavišai “+” / “-“ artina /tolina planetos vaizdą kameros fokuse
- Klavišai nuo “1” iki “0” judina kamerą pastoviu greičiu: “1” – 0.1 m/s, “2” – 1 m/s ir t.t.
- Klavišas “Tarpas” sustabdo kameros judėjimą
- Klavišas “C” atsuka kamerą į geometrinį planetos centrą
- Klavišas “R” atsuka kamerą kryptimi nuo planetos centro
- Klavišas “Q” keičia planetos modelio detalizacijos lygį (jų yra penki, nuo smulkaus iki stambaus, leidžia padidinti kadrų dažnį silpniems 3D spartintuvams)
- Klavišas “B” įjungia arba išjungia uždengtų daugiakampių atkirtimo algoritmą
- Klavišas “W” perjunginėja iš karkasinio į tekstūruoto vaizdo rodymo režimus

Kamerą galima judinti klavišais „↑“, „↓“, „→“, „←“

Programos darbo pabaiga

Programos darbo užbaigimui reikalinga pelės pagalba nuspausti lango sisteminio uždarymo mygtuką viršutiniame dešiniajame lango kampe.

10. Produkto kokybės įvertinimas

Modeliavimo sistemos kokybė buvo vertinama sistemos testavimo metu.

Testavimo rezultatai:

1) PĮ sistema atitinka projekte numatytus vartotojo reikalavimus
2) Sistemos specifikacijoje numatytas funkcionalumas realizuotas pilnai
3) Sistemoje pastebėti keli nekritiški nestabilaus darbo atvejai.
4) Testavimo metu kritiškų programos darbui klaidų neišryškėjo

Galutinis testavimas bus atliekamas įvertinant vartotojų atsiliepimus (anketos vartotojams). Gauti vartotojų pasiūlymai leis patobulinti silpnąsias sistemos puses ir pasiekti, kad sistemos darbas būtų maksimaliai stabilus.

11. Produkto ir rinkos analogų funkcionalumo palyginimas

Realizuotas produktas savo bazinėmis funkcinėmis galimybėmis atitinka uždavinio aplinkos analizėje apžvelgtus analogus.

Skirtumai nuo rinkos analogų:

1) Vartotojui leidžiama kurti ne tik su sistema pateiktus, bet ir savos kūrybos planetinių kūnų modelius.
2) Sistemos duomenys aprašomi X3D standartu XML duomenų aprašymo kalba.
3) Modelių parametrai keičiami duomenų failų korekcijos pagalba.

12. Išvados

1) Atlikus programinės įrangos analizę buvo nustatyti specializuotai planetinių kūnų projektavimo programinei įrangai keliami reikalavimai bei vartotojų poreikiai.
2) Šiuolaikinės kompiuterių grafikos galimybės leidžia kurti mokomąsias sistemas, kuriose dominuoja aukštas vizualizacijos kokybės lygis, ko pasėkoje vartotojas lengviau priima kompiuterinę mokomąją medžiagą.
3) Sistemos duomenų aprašymas, dėl pačios sistemos objektinio projektavimo metodologijos ypatybių, yra lengvai realizuojamas XML standarto duomenų struktūromis.
4) Planetinių kūnų modeliavimo algoritmų tyrimas leido nustatyti pagrindinius modeliavimo sistemos kūrimui reikalingus algoritmus ir jų pritaikymo galimybes.
5) Objektinė modeliavimo sistemos architektūra leidžia plėsti sistemos funkcionalumą pagal ateityje iškilsiančius poreikius.
6) Grafinių modeliavimo algoritmų taikymas leidžia pasiekti optimalų kompiuterių grafinio vaizdo posistemės panaudojimą modeliavimo sistemose.

13. Literatūra

1. The Art & Science of making games, [interaktyvus] [žiūrėta 2003-10-15], prieiga per internetą: http://www.gamasutra.com/features/programming .
2. Virtual Terrain Project; [interaktyvus] [žiūrėta 2003-11-10], prieiga per internetą http://www.vterrain.org .
3. Game Developers network, [interaktyvus] [žiūrėta 2003-11-10], prieiga per internetą http://www.gamedev.net .

16. Bad Dog Games software, [interaktyvus] [žiūrėta 2003-11-15], prieiga per internetą http://www.baddoggames.com/planet .
17. Planet Engine Project, [interaktyvus] [žiūrėta 2003-11-15], prieiga per internetą http://drtypo.free.fr .
18. Thatcher Ulrich home page, [interaktyvus] [žiūrėta 2003-01-15], prieiga per internetą http://tulrich.com/geekstuff/chunklod.html
19. Brutzman D. X3D-Edit Authoring Tool for Extensible 3D Graphics, [interaktyvus] [žiūrėta 2003-11-05], prieiga per internetą: http://www.web3d.org/TaskGroups/x3d/translation/X3D-EditAuthoringTool_files/frame.html

8. Grafinių tekstūrų šaltinis – interneto svetainė http://www.space-graphics.com .

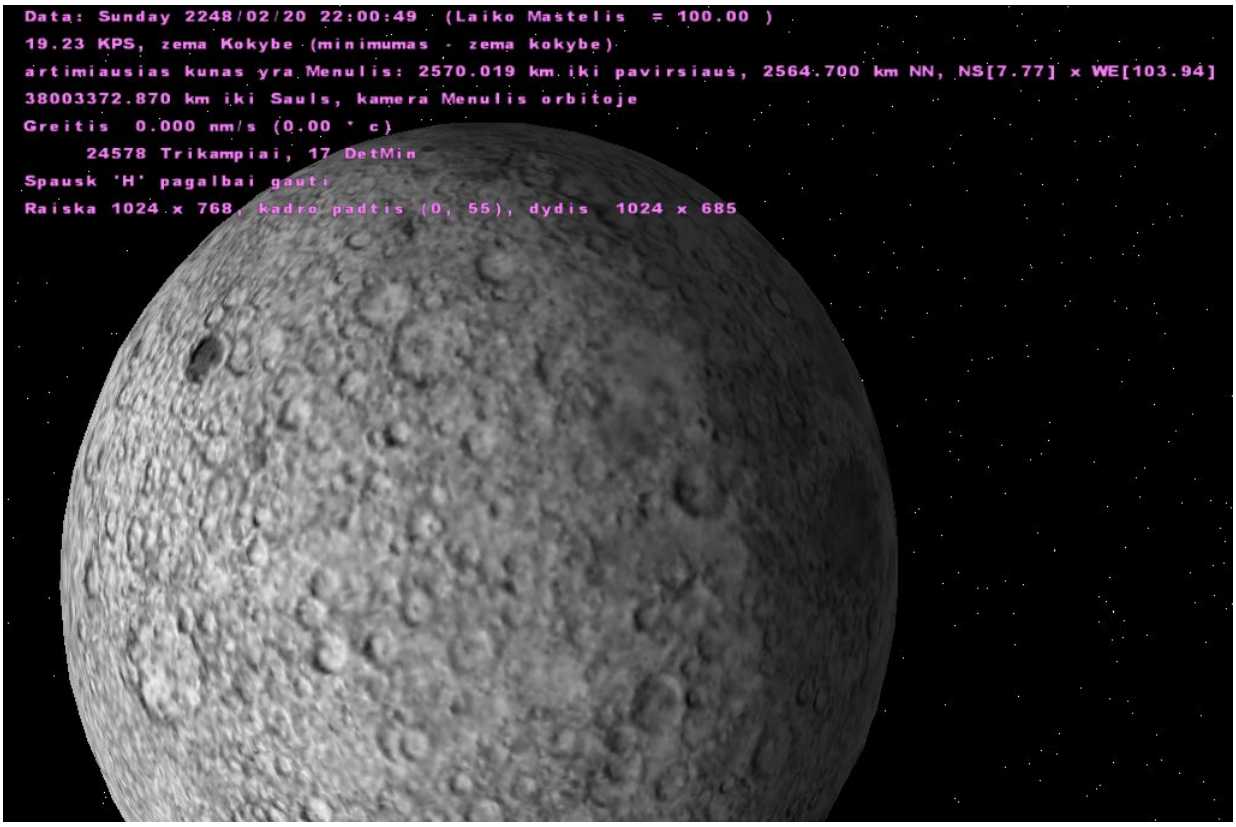
14. Summary

Analysis of planetary body creation algorithms

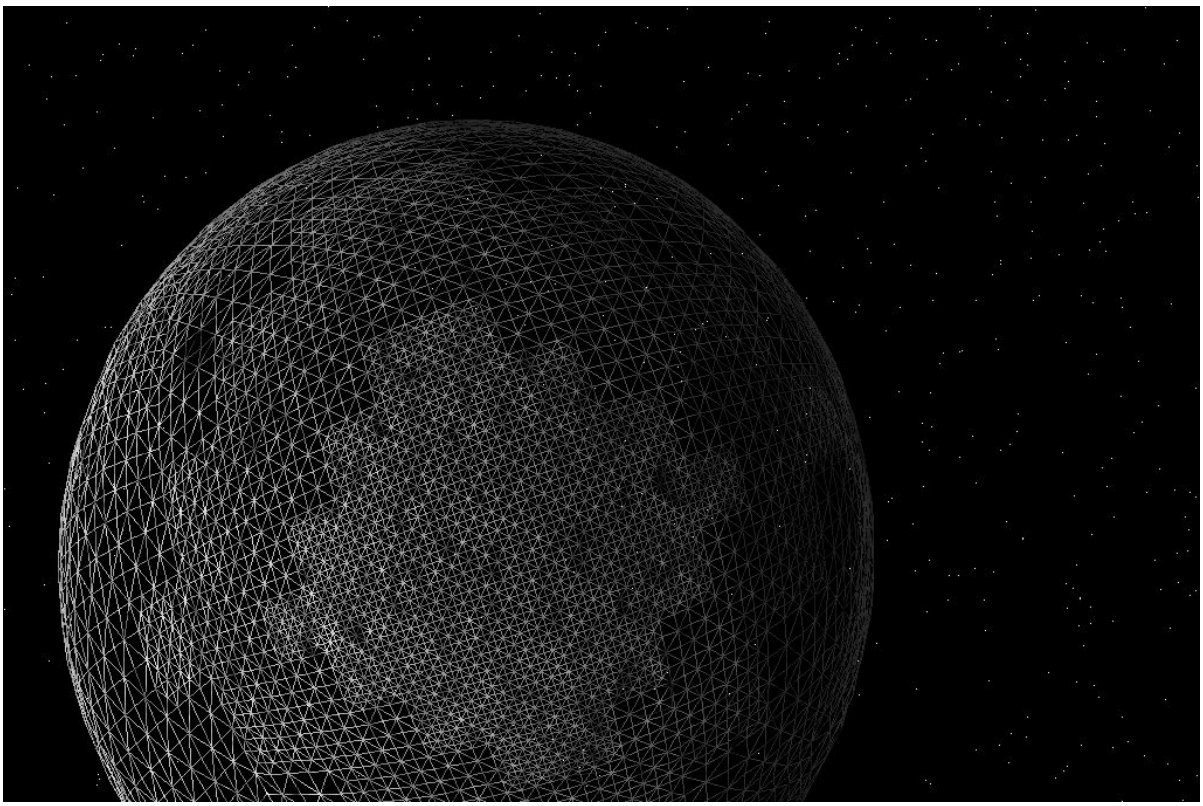
The bursted developing of 3D computer graphics on PC makes possible using it for increasing visual quality of teaching materials which are used in physic and informatic courses of secondary school. This master of science work is provided to research 3D graphic potential capabilities for interactive planetary bodies modeling. There is provided observation and analysis material about such existing modeling systems. The work contains software project for planetary body modeling system which will be used for teaching purposes. Data for system can be prepared by system's user using Extended Markup Language (XML) standard X3D. The system makes generation and visualization of described planenetary body allowing interaction with user and providing graphic subsystem work parameters for analysis.

Building of such system project requires using algorithhms which allows paralel work of PC's central processing unit and 3D acceleration chip. They are the key for succesive system realization according users requirements. The system's algorithms are designed in object oriented manner for making it more extensible and adjustable for needs of school's teaching personal. It is possible to extend system functionality in the future providing demonstration of space physic laws.

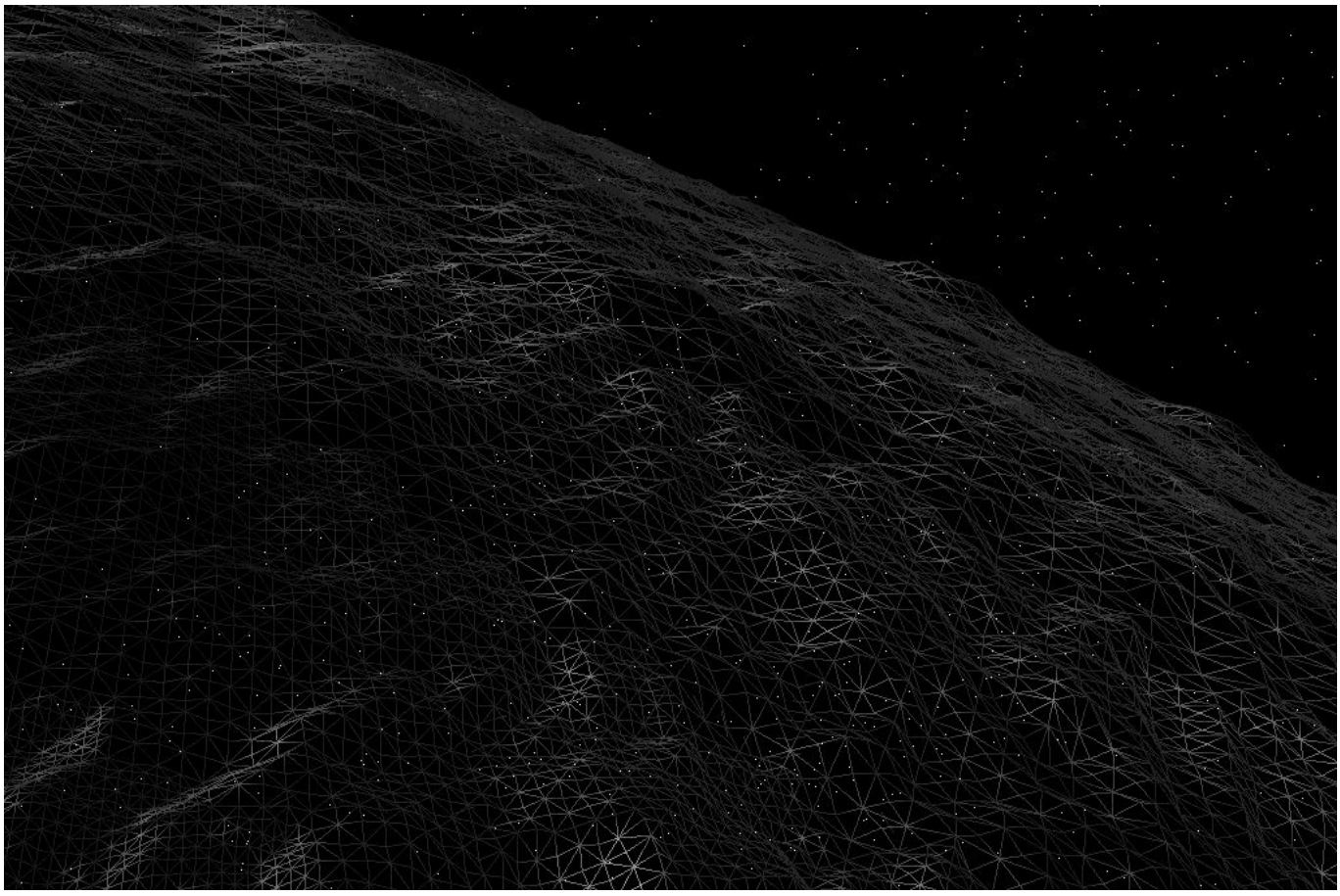
15. Priedai



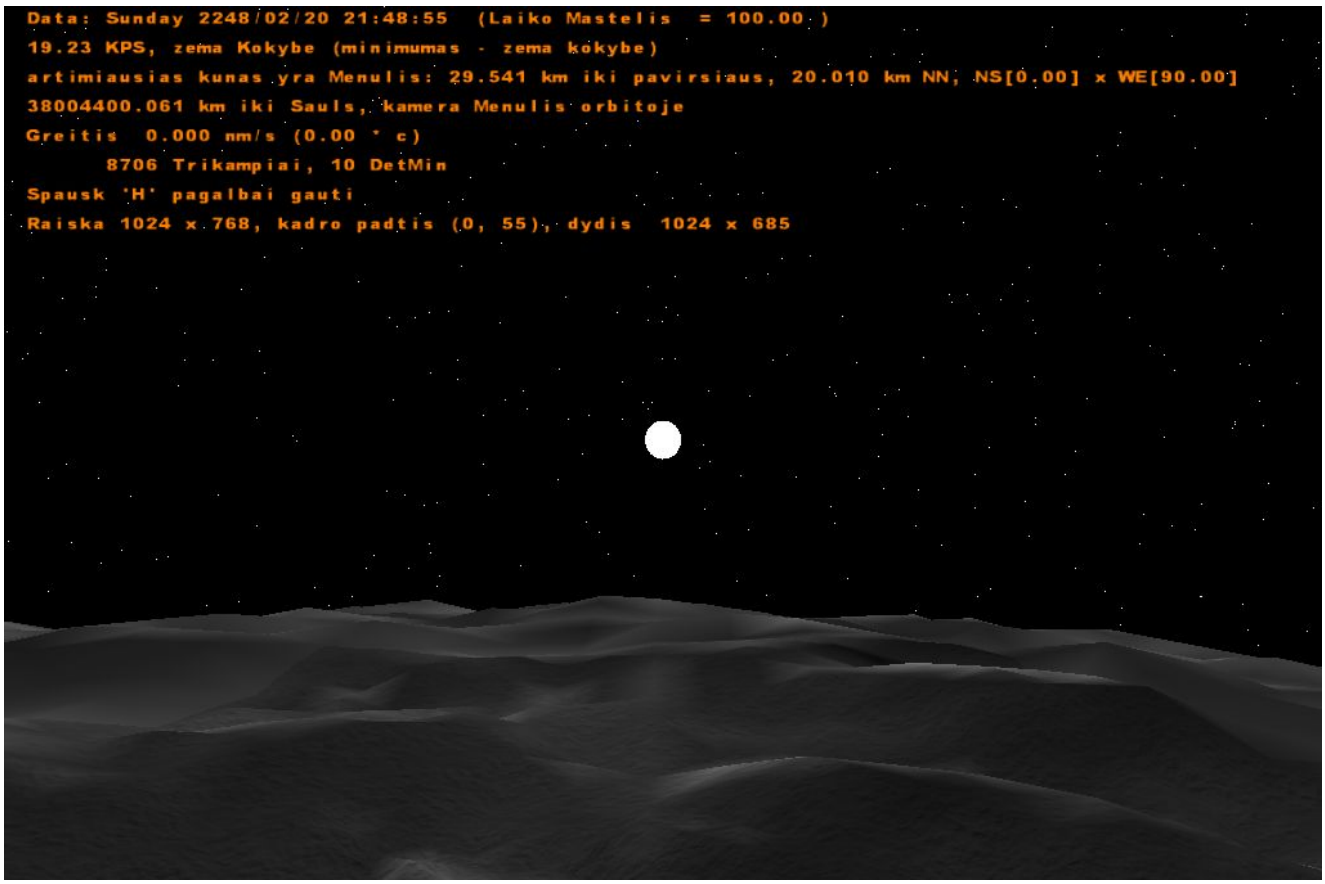
Pav. 13 PĮ ekranvaizdis: Tekstūruotas modelis



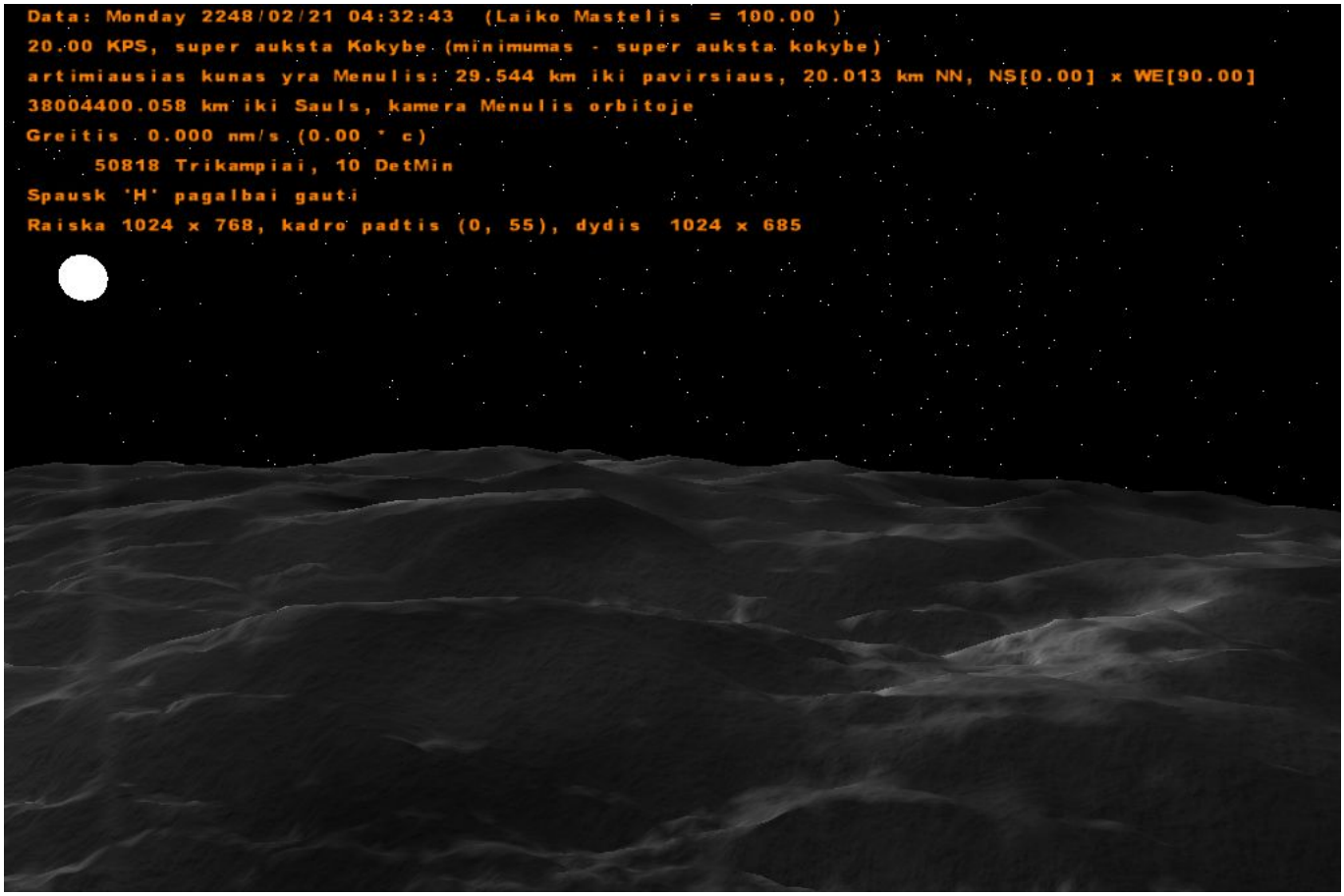
Pav. 14 PĮ ekranvaizdis: kontūrinis modelis



Pav. 15 PĮ ekranvaizdis: kamera apžvalgos padėtyje



Pav. 16 PĮ ekranvaizdis: žemos raiškos reljefas



Pav. 17 PĮ ekranvaizdis: Super aukštos raiškos reljefas

Pav. 18 PĮ ekranvaizdis: Marso vaizdas iš orbitos