KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

GEDIMINAS KVIETKAUSKAS

PASKIRSTYTŲ SKAIČIAVIMŲ ĮTAKA FIZIKOS UŽDAVINIŲ SPRENDIMŲ SPARTAI

Magistro baigiamasis darbas

Darbo vadovas
lekt.dr. Š. Packevičius

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

GEDIMINAS KVIETKAUSKAS

PASKIRSTYTŲ SKAIČIAVIMŲ ĮTAKA FIZIKOS UŽDAVINIŲ SPRENDIMŲ SPARTAI

Magistro baigiamasis darbas

Darbo vadovas
lekt. dr. Š. Packevičius
2013-05-23

Recenzentas
lekt. dr. K. Jankauskas
2013-05-23

Atliko
IFM-1/2 gr. studentas
Gediminas Kvietkauskas
2013-02-23

KAUNAS, 2013

AUTENTIŠKUMO PATVIRTINIMAS

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

2013 - 05 - 22 d.
Kaunas

Autorius, Gediminas Kvietkauskas
(vardas, pavardė)

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis magistro darbas (toliau vadinama – Kūrinys) Paskirstytų skaičiavimų įtaka fizikos uždavinių sprendimų spartai
(kūrinio pavadinimas)

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

Gediminas Kvietkauskas _____
(vardas, pavardė) (parašas)

SANTRAUKA

Iš programuotojo perspektyvos, riba tarp aparatūrinės ir programinės įrangos sparčiai mažėja. Kol programuotojai stengiasi pasiekti reikalaujamą spartą šiuolaikinėms sistemoms, jiems teks išnaudoti alternatyvius skaičiavimų elementus, tokius kaip vaizdo plokštes.

Šiame darbe apžvelgiama esama lygiagrečių skaičiavimų naudojant vaizdo plokštes situacija. Apžvelgiamas fizikos simuliacijos skaičiavimo uždavinių sprendimas panaudojant vaizdo plokštes kaip skaičiavimo vienetus. Analizuojant esamus produktus ir pritaikant technologijas paskirstytiems skaičiavimams naudojant OpenCL atliekami eksperimentai. Šie eksperimentai parodys teigiamas ir neigiamas, paskirstytų skaičiavimų fizikos uždaviniams spręsti, puses.

SUMMARY

The line between hardware and software is shrinking. While programmers and developers desperately try to reach the required performance for applications that require huge computations, they will be forced to use multiprocessing. Not only using a CPU as a main computation unit, but other resources like GPU.

In these theses we will review the current situation in multiprocessing using GPU‘s. An investigation of computation of physics will be carried out. While analyzing current products and applying technologies for GPU computation using OpenCL we will execute appropriate experiments. These experiments will show the up and down sides of GPU computing for specific physics tasks.

TURINYS

LENTELŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Procesorius (angl. processor)	loginis informacijos apdirbimo įrenginys.
API (angl. Application programming interface)	aplikacijų programavimo sąsaja suteikia kompiuterinė sistema, biblioteka ar programa tam, kad programuotojas galėtų pasiekti jos funkcionalumą ar apsikeistų su ja duomenimis.
Lygiagretūs skaičiavimai (angl. multiprocessing)	skirtingų operacijų vykdymo paskirstymas keliems procesoriams ar kompiuteriams. Lygiagrečiojo programavimo taikymas prasmingas tik tada, jei darbai gali būti padalinami ir vykdomi vienu metu.
GPU (angl. graphics processing unit)	grafikos apdorojimo įrenginys, dažniausiai randasi vaizdo plokštėje, kartais ir pagrindiniame procesoriuje.
CPU (angl. central processing unit)	pagrindinis procesorius. Funkcinis vienetas, kurį sudaro vienas arba keli procesoriai ir jų vidinė atmintis.
PĮ	programinė įranga
GPGPU (angl. purpose computing on graphics processing units)	bendro panaudojimo skaičiavimai naudojantis vaizdo plokštėmis.
3D	trimatė erdvė.
Zlib	programinės įrangos licencijos tipas leidžiantis naudotis programine įranga nemokamai.
I/O (angl. input/output)	įėjimas/išėjimas. Kompiuteriniuose terminuose dažnai naudojama kaip bendravimo galimybė tarp sistemų ar posistemių.
XML (angl. Extensible Markup Language)	yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.
UML (angl. Unified Modeling Language)	modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.
Pikselis (angl. pixel)	mažiausias rastrinio (iš taškų sudaryto) vaizdo elementas ekrane.
Pixel shader	grafikos apdorojimo įrenginio komponentas, kuris gali būti programuojamas atlikti operacijas susijusias su pikseliais (apšvietimas ir pan.).
Vertex shader	grafikos apdorojimo įrenginio komponentas, kuris gali būti programuojamas atlikti operacijas su scenų

	geometrija.
AABB (angl. Axis aligned bounding box)	pagal ašis gaubiantis stačiakampis gretasienis.
Broadphase fazė	skaičiavimų fazė, kai objektai scenoje yra apgaubiami pagal ašis stačiakampiu gretasieniu. Taip apskaičiuojamas grubus objektų susikirtimas.
Narrowphase fazė	skaičiavimų fazė, kai objektų kolizija scenoje skaičiuojama poromis, tiksliai pagal objektų paviršių.
OpenGL	įvairias kalbas ir platformas palaikanti programavimo aplinka atvaizduoti kompiuterinę grafiką dvimačiu arba trimačiu pavidalu.
GLUT	OpenGL biblioteka programoms rašyti.
Kadrai per sekundę (angl. FPS, Frames per second)	metrika, nusakanti kiek kadrų per sekundę parodoma ekrane.
CPPUNIT	testavimo biblioteka skirta C++ parašytoms programoms testuoti.
CUDA įvertis	NVIDIA sukurtas vienetas įvertinti skaičiavimų spartą, kurią gali pasiekti NVIDIA grafikos apdorojimo įrenginys.
Poligonas	plokščia figūra sudaryta iš tiesių linijų.
Objektų poligonų kiekis (angl. object polygon count)	poligonų visuma sudaranti objektus. Objektai grafikos apdorojimo įrenginyje sudaryti iš poligonų.
Implementacija	programinės įrangos realizacija

ĮVADAS

Paskirstytų skaičiavimų įtaka fizikos uždavinių sprendimų spartai magistrinis darbas priklauso programų sistemų inžinerijos studijų programai.

Darbo problematika ir aktualumas

Nagrinėjama problema – lėtas fizikos uždavinių skaičiavimas. Problema pasireiškia esant dideliems objektų kiekiams sukurtoje scenoje. Aparatinei įrangai spartėjant vis lėčiau, didesnė dalis skaičiavimų optimizavimo tenka programinei įrangai.

Darbo tikslas ir uždaviniai

Esant tokiam poreikiui kyla pagrindinis darbo tikslas – pagreitinti fizikos skaičiavimus panaudojant vaizdo plokštes, atlikti spartos tyrimą ir palyginti skaičiavimų spartą su kitomis fizikos bibliotekomis.

Fizikos bibliotekos papildymas turi leisti programuotojams lengvai ir greitai panaudoti jau naudojamos bibliotekos kodą ir biblioteką pritaikyti savo reikmėmis greitai, be daug nustatymų. Papildymas turi būti savaime intuityvus, kad neskaitant kodo reikiamos programinės įrangos dalys galėtų būti pakeičiamos kitomis. Tam reikalaujama naudoti panašius, atliekamus skaičiavimus atitinkančius klasių, metodų ir kintamųjų pavadinimus. Panaudojus sukurtą PĮ fizikos uždavinių skaičiavimas turi pagreitėti esant dideliam objektų skaičiui scenoje. Nustatomi pagrindiniai realizacijos kriterijai – atvirojo kodo naudojimas, komponentinis programų kūrimas, galimybė paleisti sistemą įvairiose aplinkose.

Darbo uždaviniai – išanalizuoti paskirstytųjų skaičiavimų literatūrą, ištirti panašias naudojamas programines įrangas, nustatyti jų privalumus ir trūkumus. Surinkti reikalavimus fizikos bibliotekai ir juos panaudoti projektuojant fizikos bibliotekos papildymo modelį. Vadovaujantis sukurtu projektu, atlikti programinės įrangos realizaciją.

Darbo rezultatai ir jų svarba

PĮ turi turėti galimybę ne tik pakeisti ir panaudoti alternatyvų skaičiavimų procesorių, tačiau ir paleisti simuliacijos demonstraciją, kuri būtų atvaizduojama be programinio kodo kūrimo. Vartotojui neturi reikėti programiškai kurtis scenos tik tam, kad išmėginti ar programinė įranga jo kompiuteryje veikia arba patikrinti numatytosios scenos skaičiavimo spartų skirtumus. Šiems tikslams pasiekti svarbios PĮ dalys yra perrašomos skaičiavimams naudojant vaizdo plokštę atlikti. Atliekamas tyrimas ir palyginamos įvairių esamų bibliotekų sparta tokiomis pat sąlygomis.

Darbo struktūra

Darbas susideda iš šių skyrių:

- Probleminės srities analizė – analizuojama su darbo problematika susijusi informacija.

- Sprendimo reikalavimų specifikacija ir projektas, formalus aprašas – aprašomas projekto sprendimas, pagrindiniai projekto sprendimai pateikiami grafiškai.
- Sprendimo realizacija ir testavimas – pateikiama sprendimo realizacija ir veikimo aprašas, testavimo modelis.
- Eksperimentinis sprendimo tyrimas – pateikiamas eksperimento planas, jo rezultatai.
- Rezultatų apibendrinimas ir išvados – pateikiamas tezių apibendrinimas ir išvados.

1 PROBLEMINĖS SRITIES ANALIZĖ

Paralelių skaičiavimų spartos ateitis priklauso grafikos apdorojimo įrenginiams, jų sugebėjimui didelį kiekį duomenų apdoroti vienu metu[1].

Naudojantis esamomis technologijomis galime paskirstyti skaičiavimus keliems įrenginiams. Populiarėjant lygiagretiems skaičiavimams, šie skaičiavimai keliasi į vaizdo plokštes ir jų klasterius[2][3].

Bendro panaudojimo skaičiavimų naudojantis vaizdo plokštėmis sritis yra gan nauja, jos idėjos vystomos tik dešimtmetį. Pirmiausia paraleliai skaičiavimai buvo panaudoti pagrindiniuose procesoriuose ir išsivystė į atskirą bendro panaudojimo skaičiavimų naudojant vaizdo plokštes sritį. Grafikos apdorojimo įrenginiai yra orientuoti į kompiuterinės grafikos apdorojimą. Kompiuterinės grafikos srityje dominuoja operacijos su matricomis ir vektoriais.

Atliekama fizikos uždavinių skaičiavimus atliekančios keliais įrenginiais besinaudojančios PĮ analizę. Analizės metu bandoma nustatyti esamos programinės įrangos privalumus ir trūkumus. Palyginti esamą programinę įrangą tarpusavyje.

Pagal atliktą analizę kituose darbo skyriuose bus modeliuojami įvairūs sistemos aspektai, projektuojami sistemos komponentai.

1.1 Egzistuojantys sprendimai skaičiavimų lygiagretinimui atlikti

Pasinaudoti vaizdo plokščių skaičiavimais leidžia keletas skirtingų sistemų. Jos suteikia programavimo sąsajas(API) skirtas darbui tarp pagrindinių ir vaizdo plokščių procesorių (Stream processing)[4]. Jos leidžia manipuliuoti duomenimis panaudojant keletą branduolių skaičiavimui nesinchronizuojant ir nekomunikuojant tarpusavyje. Šių sprendimų privalumai ir trūkumai pateikiami žemiau.

Close to Metal

Tai yra viena iš didžiųjų rinkos žaidėjų biblioteka, leidžianti programuotojui prieiti prie žemo lygio funkcijų. Ją kūrė ATI(dabar priklausanti AMD Graphics Products Group). Ji taikoma bendrų skaičiavimų apdorojimui naudojant vaizdo plokščių branduolius (GPGPU – General purpose computing on graphics processing units) [5][6].

Buvo išleista tik bandomoji versija, ji buvo pervadinta į Stream SDK. CTM (Close To Metal) leido programuotojams tiesiogiai pasiekti atmintį ir pagrindinius paralelių skaičiavimų elementus moderniose AMD vaizdo plokštėse. CTM leido programuotojams prieiti prie funkcijų, kurios seniau buvo jiems nepasiekiamos.

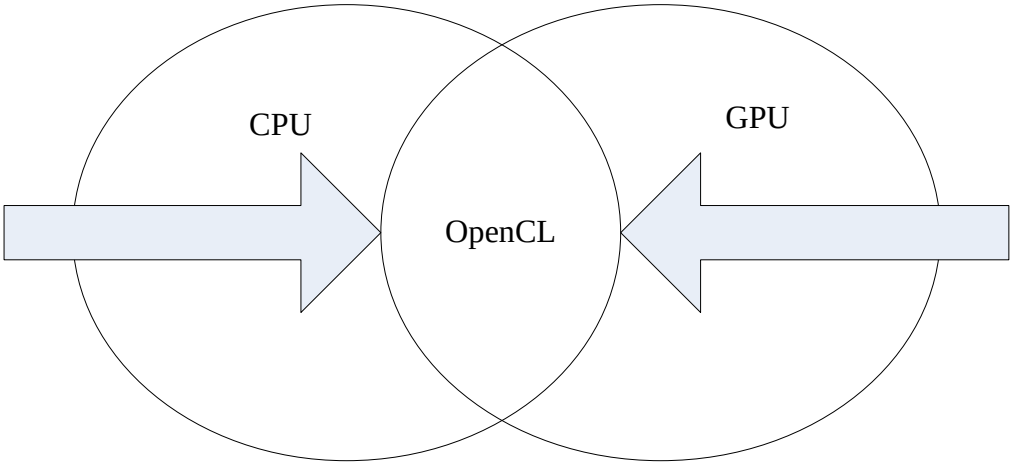
CUDA

CUDA (Compute Unified Device Architecture) yra paralelių skaičiavimų architektūra sukurta NVIDIA[7]. CUDA skaičiavimų variklis prieinamas programinės įrangos kūrėjams netgi keliomis programavimo kalbomis. Programuotojai gali naudoti „C for CUDA“ kuri turi papildomų NVIDIA įskiepių ir apribojimų ir taip kurti paralelius algoritmus skirtus vykdyti naudojant vaizdo plokštės branduolius.

Kompiuterinių žaidimų industrijoje, šios technologijos yra naudojamos ne tik vaizdo apdorojimui, bet ir žaidimų fizikos skaičiavimams (dūmai, ugnis, skysčiai, objektų sąveika).

OpenCL

OpenCL (Open Computing Language) yra karkasas, skirtas kurti programas, kurios leidžia bendradarbiauti įvairiems procesoriams kartu. OpenCL leidžia bet kuriai programai prieiti prie vaizdo plokštės branduolių ir taip juos panaudoti ne tik vaizdui apdoroti. Pav. 1 pavaizduota OpenCL vieta aparatiniu pjūviu.



Pav. 1 Bendradarbiavimas tarp CPU ir GPU naudojant OpenCL

OpenCL yra gerai žinomų OpenGL ir OpenAL, skirtų 3D grafikos ir garso apdorojimui, analogas. Jį kuria pelno nesiekianti grupė Khronos Group. Ši technologija buvo priimta visų įvairios paskirties branduolių kūrėjų (Intel, AMD, Nvidia, ARM).

OpenCL pradininkai buvo Apple Inc. Jie turi visas teises į šia programinę įrangą, tačiau dirba kartu su AMD, IBM, Intel ir NVIDIA. Pirmoji versija OpenCL 1.0 buvo išleista kartu su „Mac OS X Snow Leopard“ 2008 metais [8]. Lentelėje 1 pavaizduota vaizdo plokščių ir pagrindinių procesorių palaikymas atliekant lygiagrečius skaičiavimus.

Lentelė 1 GPGPU skaičiavimus atliekančių PĮ palyginimas

API	CUDA	OpenCL	CTM
Nuosavybės teisės	Taip, NVIDIA	Taip, Chronos group	Taip, AMD
GPU palaikymas	Tik NVIDIA GPU	Intel, NVIDIA, AMD	Tim AMD GPU
CPU palaikymas	Nėra	Yra	Nėra

1.2 Egzistuojantys sprendimai fizikos skaičiavimams atlikti

PhysX

Sukurta 2008 metais. Šią fizikos biblioteką sukūrė Ageia, tačiau 2008 metais ją nusipirko NVIDIA[9]. Dabar ji tapo viena pagrindinių bibliotekų, kurios įgalina procesoriaus ir vaizdo plokštės procesorių darbą kartu fizikos skaičiavimams ne tik asmeniniuose kompiuteriuose, bet ir Wii, PlayStation 3, Xbox 360, Linux.

Privalumai:

- Palaiko įvairias platformas
- Yra viena daugiausiai naudojamų šiuo metu

Bullet

Bullet yra atviro kodo fizikos variklis, kuris palaiko 3D objektų kolizijos nustatymą, kietų bei minkštų kūnų dinamiką[10]. Ji naudojama žaidimuose ir vizualiems efektams filmuose. Ši fizikos biblioteka yra išleista pagal zlib licenziją[11]. Autorius Erwin Coumans, dabar dirba AMD. Naudojantis šia biblioteka buvo kuriami tokie filmai kaip 2012, Hancock, The A–team. Žaidimai GTA 4, 3d Mark 2011.

Privalumai:

- Visiškai nemokama ir atviro kodo
- Turi optimizacijų skirtų PS3, CUDA ir OpenCL.

Havok

Sukūrė airių kompanija Havok. Pagrindinis šios fizikos bibliotekos tikslas – vaizdo žaidimai. Ši biblioteka nėra plačiai paplitusi, tačiau turi labai gerus dinaminius tvirtų objektų kolizijos skaičiavimo algoritmus. Taip pat palaiko skeletinę animaciją bei inversinę kinematiką . Havok leidžia sukurti realistiškesnį virtualų pasaulį vaizdo žaidimuose. Kompanija tapo labiau pastebima tarp jos konkurentų, kai Intel 2007 metais pasirašė su ja sutartį dėl bendradarbiavimo.

Privalumai:

- Palaiko įvairias platformas (Unix, Linux, PS3, PS2, PSP, GameCube, Wii, Mac, Xbox 360, Xbox, Windows, Android)

Open Dynamics Engine (ODE)

Pradėta kurti 2001 m. ir buvo naudojama daug sistemų bei žaidimų[12]. Žymiausi žaidimai, kuriuose buvo panaudota ši biblioteka: BloodRayne2, S.T.A.L.K.E.R, Titan Quest, World of Goo. Šos bibliotekos autorius Russel Smith.

Open Dynamics Engine biblioteka naudojama simuliuoti kūnų kolizijai erdvėje. Ji nesurišta su jokia atvaizdavimo biblioteka, tačiau turi paprastą atvaizdavimo mechanizmą pavadinimu *drawstuff*. ODE dažnai renkamasi simuliuojant robotiką.

Newton Dynamics

Ši biblioteka yra atviro kodo ir platinama pagal Zlib licenciją. Ji taip pat skirta kūnų simuliacijai sukurtoje erdvėje žaidimuose ir kitose realaus laiko aplikacijose. Newton Dynamics[13] leidžia programuotojams panaudoti šią fizikos biblioteką tiek asmeniniuose tiek komerciniuose projektuose. Bibliotekos autoriai Julio Jerez, Alain Suero.

Palaikomos operacinės sistemos: Windows, Mac OS X, iPhone, Linux.

1.3 Analizės išvados

Šiuo metu nėra atvirojo kodo PĮ palaikančios įvairias vaizdo plokštes, kuri galėtų atlikti fizikos skaičiavimus. Pasirinktai problemai spęsti pasirinkome vieną iš esamų fizikos bibliotekų ir paskirstytų skaičiavimų biblioteką. Fizikos uždaviniams simuliuoti pasirinkta Bullet skaičiavimų biblioteka. Ji pasirinkta, nes yra atvirojo kodo, galima lengvai perkrauti dalį metodų ir panaudoti savo idėjas skaičiavimams. Ši biblioteka yra plačiai paplitusi, gerai dokumentuota ir lengvai keičiama.

Skaičiavimams lygiagretinti pasirinkta Chronos Group OpenCL programavimo sąsaja. Ji palaiko daugumos didžiųjų kompanijų įrenginius. AMD, Intel, NVIDIA, Radeon kuriamos vaizdo plokštės ar procesoriai gali veikti naudojant OpenCL.

Šios technologijos pasirinktos, nes rinkoje kol kas nėra atviros fizikos skaičiavimų bibliotekos, kuri palaikytų keletą įrenginių. NVIDIA sukurta PhysX sistema veikia tik su NVIDIA vaizdo plokštėmis. PhysX diegiama kartu su NVIDIA vaizdo plokščių tvarkyklėmis, todėl yra gan plačiai naudojamos, tačiau ne atvirojo kodo ir veikia tik su pačiomis NVIDIA vaizdo plokštėmis.

ATI pasiūlė savo paskirstytų skaičiavimų sistemą ATI Stream. Joje analogiškai kaip ir NVIDIA PhysX palaikomi vaizdo plokščių ir bendrieji procesorių skaičiavimai. Ši technologija veikia tik su ATI vaizdo plokštėmis.

Dėl tokios susiklosčiusios situacijos, kai kiekviena vaizdo plokštė reikalauja programuoti atskiromis technologijomis, buvo pasirinkta Bullet ir OpenCL. Jos veikia visose platformose ir yra lengvai plečiamos. Patobulinta programinė įranga galės atlikti skaičiavimus sparčiau ir su bet kokiu pasirinktu įrenginiu.

1.4 Įgyvendinimo problemos

Aparatinės įrangos problemos:

- Kompiuteris gali neišnaudoti visų GPGPU skaičiavimo galimybių ir dėl kitų priežasčių. Centrinis procesorius gali nespėti skirstyti duomenis, arba gali neužtekti įėjimo/išėjimo (I/O) spartos[14].

- Šiuo metu vaizdo plokštės neišnaudoja savo viso galingumo, todėl nėra didelio poreikio į jas įdėti galingus aušinimo ventiliatorius. Ši problema dar labiau pasireiškia nešiojamuose kompiuteriuose, kadangi ten energijos taupymo sumetimais aušinimo ventiliatoriaus gali išvis nebūti. Apkrovus vaizdo plokštę papildomu darbu ji gali imti kaisti, tuo sukeldama pavojų sistemos darbui. Taip pat gali sutrumpėti našaus darbo laikas naudojamu kompiuteriu, nes naudojama daugiau energijos[15].

2 SPRENDIMO REIKALAVIMŲ SPECIFIKACIJA IR PROJEKTAS, FORMALUS APRAŠAS

2.1 Reikalavimų specifikacija

Prieš projektuojant sudaromi sistemos reikalavimai. Reikalavimai pateikiami lentelėmis. Sistemos vartotojas yra programinės įrangos kūrėjas.

Šiame skyriuje pateikiami svarbiausi sistemos funkciniai ir nefunkciniai reikalavimai.

2.1.1 Funkciniai reikalavimai

Lentelėje 2 aprašomas reikalavimas „Galimybė naudoti OpenCL išeities failus“. PĮ turi turėti galimybę nuskaityti ir derinti OpenCL palaikomus .cl tipo failus. Juose aprašomi branduoliai, kuriais remiantis atliekami skaičiavimų lygiagretinimai.

Lentelė 2 Reikalavimo „Galimybė naudoti OpenCL išeities failus“ aprašas

Aprašymas	Galimybė naudoti OpenCL išeities failus
Pagrindimas	Kuriama fizikos simuliacija paremta sistema su GPU palaikymu, todėl reikalingas OpenCL išeities failų vykdymas
Šaltinis	Programuotojas
Tikimo kriterijus	Bus galima panaudoti reikalingus išeities failus bei jų funkcijas

Lentelėje 3 aprašomas reikalavimas „Automatiškai nustatomas palaikomas įrenginys“. Paleidus simuliaciją, automatiškai surandamas grafikos įrenginys. Jis panaudojamas atliekamiems skaičiavimams.

Lentelė 3 Reikalavimo „Automatiškai nustatomas palaikomas įrenginys“ aprašas

Aprašymas	Automatiškai nustatomas palaikomas įrenginys
Pagrindimas	Nieko nenustatant demonstracija turi susirasti įrenginį
Šaltinis	Programuotojas
Tikimo kriterijus	Automatiškai parenkamas teisingas, veikiantis skaičiavimo įrenginys

Lentelėje 4 aprašomas reikalavimas „Paleidžiama demonstracija“. Sukurta PĮ turi turėti ne tik bibliotekos failus, kuriuos galima panaudoti kuriant PĮ, bet ir iš karto suderintus demonstracinius failus, kuriuos būtų galima paleisti. Vartotojui tai leis nieko nekuriant išmėginti sistemos privalumus.

Lentelė 4 Reikalavimo „Paleidžiama demonstracija“ aprašas

Aprašymas	Paleidžiama demonstracija
Pagrindimas	Galimybė paleisti suderintą demonstraciją
Šaltinis	Programuotojas
Tikimo kriterijus	Veikianti demonstracija su atitinkamomis vaizdo plokštėmis

Lentelėje 5 aprašomas reikalavimas „Testų skaičiavimų duomenys saugomi“. Sukurta PĮ turi turėti testus atskirame pakete. Leidžiamų testų istorija saugoma atskiruose failuose XML formatu.

Lentelė 5 Reikalavimo „Testų skaičiavimų duomenys saugomi“ aprašas

Aprašymas	Testų skaičiavimų duomenys saugomi
Pagrindimas	Testų rezultatai turi turėti istoriją, todėl turi būti saugomi
Šaltinis	Programuotojas
Tikimo kriterijus	Gauti rezultatai įsaugomi XML formatu, įrašoma paleidimo data

2.1.2 Nefunkciniai reikalavimai

Lentelėje 6 aprašomas reikalavimas „Aiški topologija ir suprantami vardai“. Sukurtos PĮ klasių, paketų ir metodų pavadinimai turi būti aiškiai suprantami, ką atlieka ir kam skirti.

Lentelė 6 Reikalavimo „Aiški topologija ir suprantami vardai“ aprašas

Aprašymas	Aiški topologija ir suprantami vardai
Pagrindimas	Programuotojams turėtų būti nesunku pradėti naudotis įrankiu
Šaltinis	Programuotojas
Tikimo kriterijus	Nereikia grįžti ir įsiminti funkcijas kelis kartus

Lentelėje 7 aprašomas reikalavimas „Didesnė sparta naudojant GPU vietoj CPU“. Panaudojus sukurtą PĮ turėtų būti pasiekiamas greitesnė simuliacijos apskaičiavimo sparta, negu naudojant standartinį metodą.

Lentelė 7 Reikalavimo „Didesnė sparta naudojant GPU vietoj CPU“ aprašas

Aprašymas	Didesnė sparta naudojant GPU vietoj CPU
Pagrindimas	GPU turi didesnę skaičiavimų spartą visumoje, todėl jeigu jis neapkrautas gaunami rezultatai turėtų būti spartesni
Šaltinis	Programuotojas
Tikimo kriterijus	Naudojant GPU veikia greičiau

Lentelėje 8 aprašomas reikalavimas „Demonstracija atidaro naują atvaizdavimo langą“. Programinė įranga pateikiama su suderintomis demonstracinėmis programomis, kurios grafiškai atvaizduoja vykdomą simuliaciją.

Lentelė 8 Reikalavimo „Demonstracija atidaro naują atvaizdavimo langą“ aprašas

Aprašymas	Demonstracija atidaro naują atvaizdavimo langą
Pagrindimas	Turi būti atidaromas naujas langas kuriame grafiškai atvaizduojami atliekami skaičiavimai
Šaltinis	Programuotojas
Tikimo kriterijus	Demonstracija vykdoma atskirame lange

Lentelėje 9 aprašomas reikalavimas „Biblioteka turi veikti su pažangiomis vaizdo plokštėmis“. Senos vaizdo plokštės (visų gamintojų senesnės negu 2008 metų gamybos) neturi veikti su šia PĮ. PĮ privalo palaikyti tik naujesnes negu 2008m. gamybos vaizdo plokštes.

Lentelė 9 Reikalavimo „Biblioteka turi veikti su pažangiomis vaizdo plokštėmis“ aprašas

Aprašymas	Biblioteka turi veikti su pažangiomis vaizdo plokštėmis
Pagrindimas	Reikia išnaudoti naujas(nuo 2008m. gamybos) vaizdo plokštes
Šaltinis	Programuotojas
Tikimo kriterijus	Galima naudoti pažangias vaizdo plokštes

Lentelėje 10 aprašomas reikalavimas „Lengvai plečiama ir tobulinama sistema“. PĮ turi būti suprogramuota komponentiškai, turi turėti aiškią hierarchiją ir paketų, klasių, metodų pavadinimus.

Lentelė 10 Reikalavimo „Lengvai plečiama ir tobulinama sistema“ aprašas

Aprašymas	Lengvai plečiama ir tobulinama sistema
Pagrindimas	Pridėti naujų funkcijų neturėtų būti sunku, sistema turi būti išplečiama
Šaltinis	Programuotojas
Tikimo kriterijus	Lengvai plečiama sistema su gera hierarchija ir klasių, metodų, kintamųjų ir kitų objektų įvardinimu

2.2 Projektinė dalis

Šis skyrius supažindina su kuriamos programinės įrangos architektūriniu vaizdu. Pateikiami keletas architektūrinių vaizdų, kurie parodo pagrindinius skirtingus kuriamos sistemos aspektus.

Skyrius skirtas bendrauti sistemos programuotojams ir architektams. Architektūros tikslai ir reikalavimai pateikiami skyriuje „Architektūros tikslai ir apribojimai“. Sistemos vaizdas pateikiamas įvairiomis diagramomis, reikalingomis suprasti sistemos architektūra ir veikimo principus.

2.2.1 Sistemos architektūros pateikimas

Šiame skyriuje sistemos architektūra yra pateikiama keliais vaizdais: loginiu vaizdu, procesų ir išdėstymo vaizdu. Visi jie pateikiami naudojant UML modeliavimo kalbą.

Reikalingiems vaizdams parodyti naudojamos šios diagramos:

- Statinis vaizdas
    - Sistemos išskaidymas į paketus
    - Paketų detalizavimas klasėmis
- Dinaminis vaizdas
    - Sekų diagramos

2.2.2 Architektūros tikslai ir apribojimai

Sistema turi keletą reikalavimų ir apribojimų:

- Sistemai realizuoti panaudota Bullet atvirojo kodo fizikos biblioteka.
- Interneto prieiga nėra reikalinga.
- Sukurtą sistemą galima panaudoti kuriant kitą programinę įrangą. Ji susideda iš sukurtos fizikos bibliotekos ir demonstracinės programinės versijos.

Reikalavimai aparatinei įrangai:

Procesorius:	1Ghz ir greitesnis
Operacinė sistema:	Windows (su reikalingomis tvarkyklėmis ir Linux, Mac OS)
Atmintis:	128MB RAM ir daugiau
Vaizdo plokštė:	NVIDIA arba RADEON vaizdo plokštės nuo 2008m. leidimo

Norint paleisti sistemą ir išmėginti jos galimybes naudojant vaizdo plokštės pajėgumą reikalingos vaizdo plokštės nuo 2008m. gamybos, kurios palaiko „pixel shader“ ir „vertex shader“.

Projektavimo ir įgyvendinimo strategija:

- Naudojami įrankiai, pagalbinės programos:
  - Bullet biblioteka.
  - OpenCL karkasas.
  - MS Visual Studio.

Sistemos kūrimo etapai:

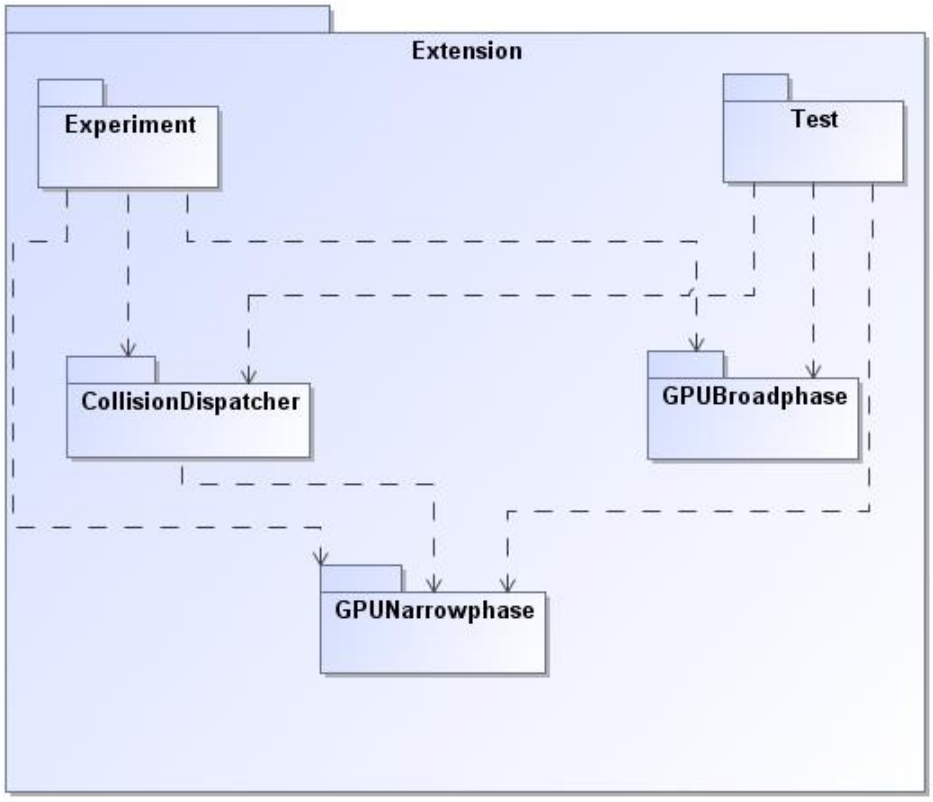
- Sukurti demonstracinę versiją panaudojant fizikos biblioteką.
- Išlygiagretinti naudojamus fizikos skaičiavimų algoritmus naudojant OpenCL.
- Sukurti spartos testus algoritmų spartai patikrinti.
- Sukurti visus pagalbinius metodus, kurie bus reikalingi demonstracinei programai.
- Sukurti grafinį sistemos atvaizdavimą panaudojant OpenGL.

2.3 Sistemos statinis vaizdas

Šiame skyriuje apžvelgiamas sistemos statinis vaizdas. Pateikiamas bendras paketų vaizdas tada jis detalizuojamas iki paketo klasių, metodų ir kintamųjų. Taip pat pateikiamas klasių/paketų aprašymas ir jų paskirtis.

2.3.1 Apžvalga

Pav. 2 pavaizduotas bendras sistemos vaizdas pasinaudojant paketų diagrama.



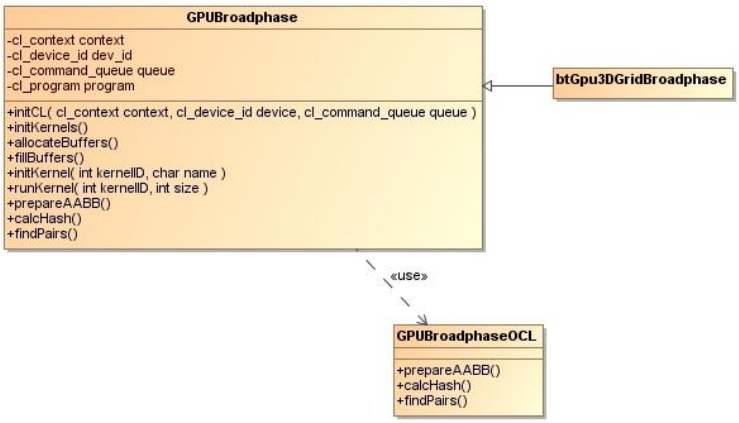
Pav. 2 Sistemos paketai

2.3.2 Paketų detalizavimas

2.3.2.1 Paketas GPUBroadphase

Šiame pakete saugomos klasės atliekančios veiksmus tiesiogiai susijusius su fizikos skaičiavimų *broadphase* faze. Tai yra pirmoji fazė, kurioje grubiai atrenkami fizikiniai kūnai, kurie turėtų būti įtraukti į tolimesni, detalesnį skaičiavimą.

Taip pat šiame pakete laikomi ir OpenCL failai susiję su *broadphase* faze. Pav. 3 pavaizduota *broadphase* fazės implementacija.



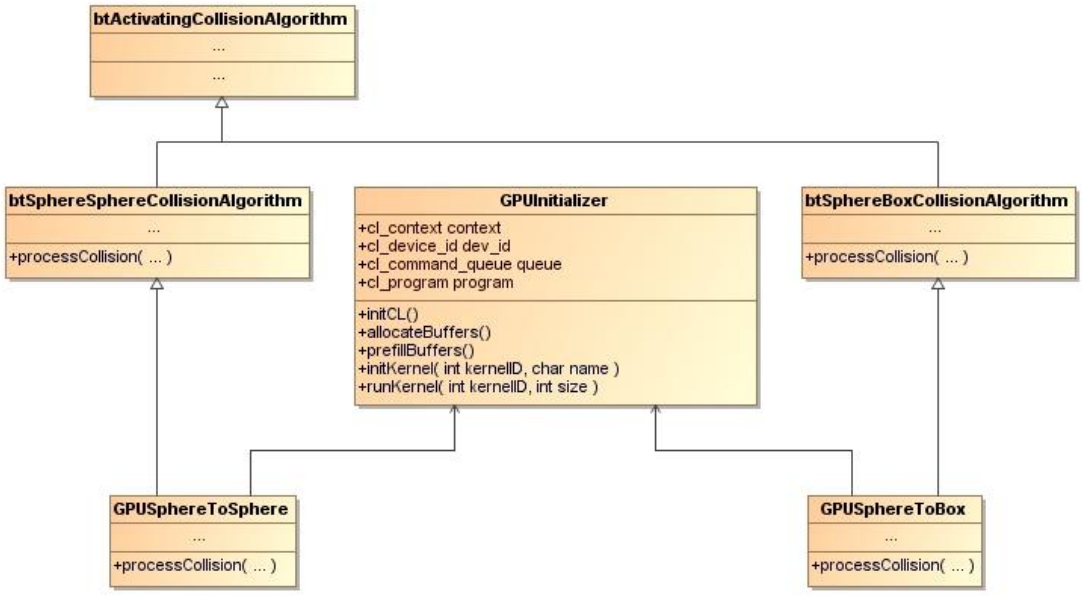
Pav. 3 GPUBroadphase paketo detalizavimas

Faile GPUBroadhpaseOCL saugomas dar nesuderintas OpenCL branduolių išeities kodas. Klasėje GPUBroadphase įvykdomas pasiruošimas leisti OpenCL kodą. Imamas įrenginys, kuriuo naudojantis bus skaičiuojama, nustatomas gijų kiekis ir suderinamas reikalingų metodų OpenCL išeities kodas.

2.3.2.2 Paketas GPUNarrowphase

Šiame pakete saugomos klasės atliekančios veiksmus su tolimesne skaičiavimų faze – *narrowphase*. Po *broadphase* fazės išrenkami kūnai, kurie galimai kertasi, yra patikrinama ar jie tikrai susikerta šioje fazėje.

Taip pat šiame pakete laikomi ir OpenCL failai susiję su *narrowphase* faze. Standartiniai tikrinantys metodai yra perkrauti. Kiekvienas GPUSPhereToSphere, GPUSphereToBox turi atitinkamus .cl failus, kuriuose atliekami skaičiavimai. Pav. 4 pavaizduota *narrowphase* fazės įgyvendinimo klasių diagrama.



Pav. 4 GPUNarrowphase paketo detalizavimas

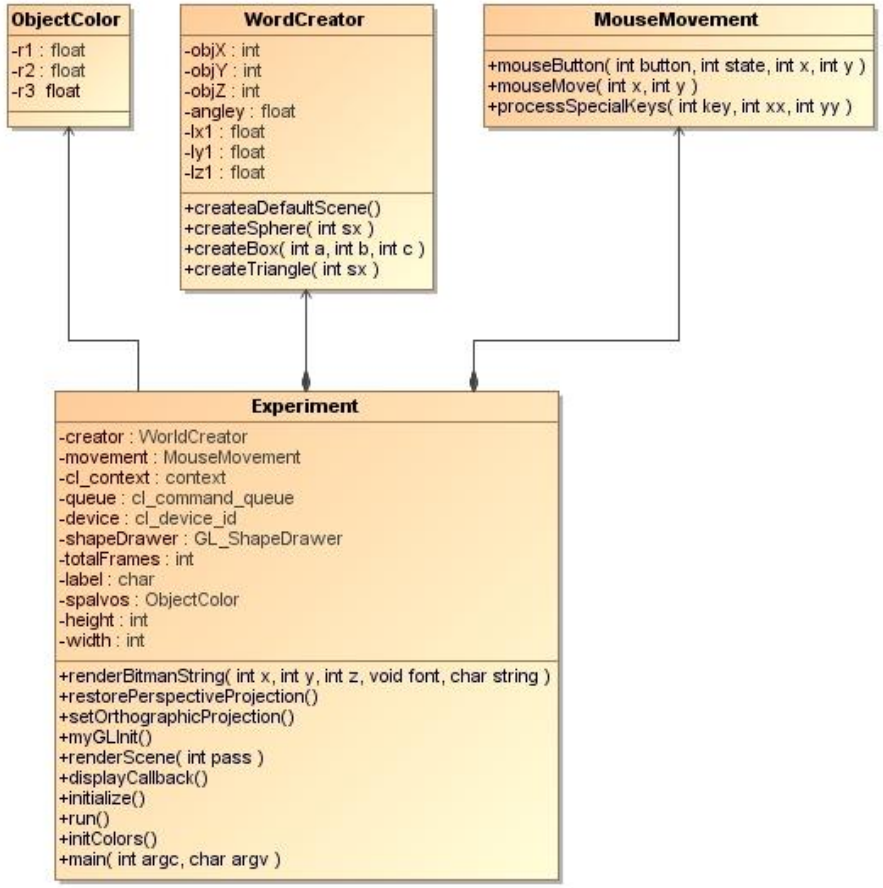
2.3.2.3 Paketas CollisionDispatcher

Jame inicijuojamos funkcijos pagal objektų tipus. Fizikiniai scenos objektai gali būti įvairūs – sfera, kubas ir pan. Taigi paėmus du skirtingus objektus reikia iškviesti atitinkamą metodą, kuris apdoroja situaciją. Šio paketo klasės rūpinasi teisingų metodų iškvietimu.

2.3.2.4 Paketas Experiment

Šiame pakete atliekami sukurtų funkcijų eksperimentai, sukuriamas fizikinis pasaulis, įkeliamas atvaizdavimas, valdymas. Taip pat aprašomos pagalbinės funkcijos greitesniam aplinkos sukūrimui.

Pakete įkeltas ir vartotojo įvesčių valdymas. Kameros valdymas pele ir klaviatūra.



Pav. 5 Experiment paketo detalizavimas

Klasėje WorldCreator sukuriama norima scena. Galima pasinaudoti sukurtais metodais kurti objektus. Klasėje MouseMovement skaičiuojamas kameros, esančios scenoje, pozicijos ir žiūrėjimo kampai. Klasėje ObjectColor kiekvienu leidimu sugeneruojamos atsitiktinės spalvos, kuriuos atvaizduoja skirtingus objektus scenoje.

Minėtos pagalbinės klasės panaudojamos klasėje Experiment. Experiment klasėje vyksta GLUT atvaizdavimo ciklas. Pasirinkto dydžio lange atvaizduojami sukurti objektai panaudojant GLDrawer klasės implementaciją. Taip pat lange rodomas kadrų per sekundę skaičius (FPS).

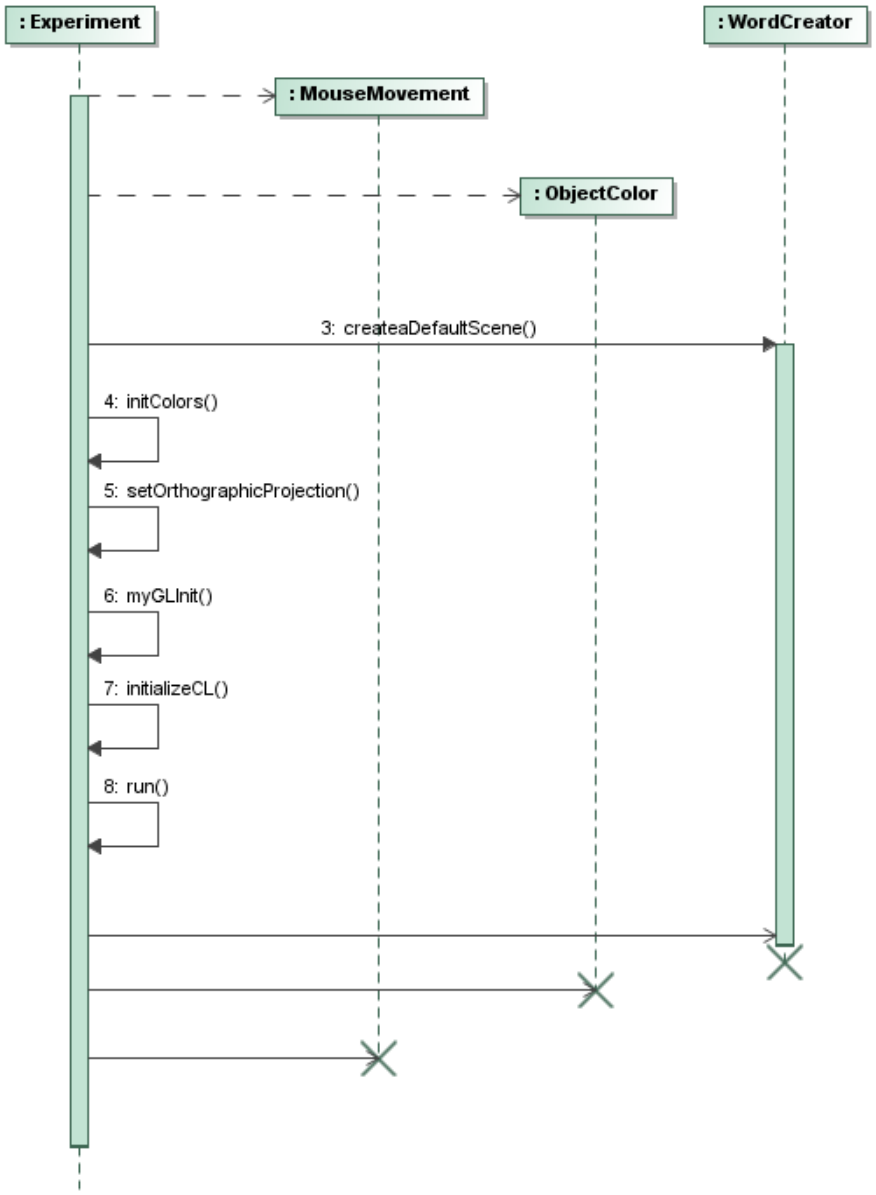
2.3.3 Paketas Test

Šiame pakete laikomi sukurtų funkcijų testai. Testai reikalingi, kad įsitikinti ar sukurtos funkcijos gauna tokį pat rezultatą kaip ir originaliosios funkcijos. Testavimui atlikti naudojama CPPUNIT testavimo biblioteka.

2.4 Sistemos dinaminis vaizdas

2.4.1 Sukūrimas

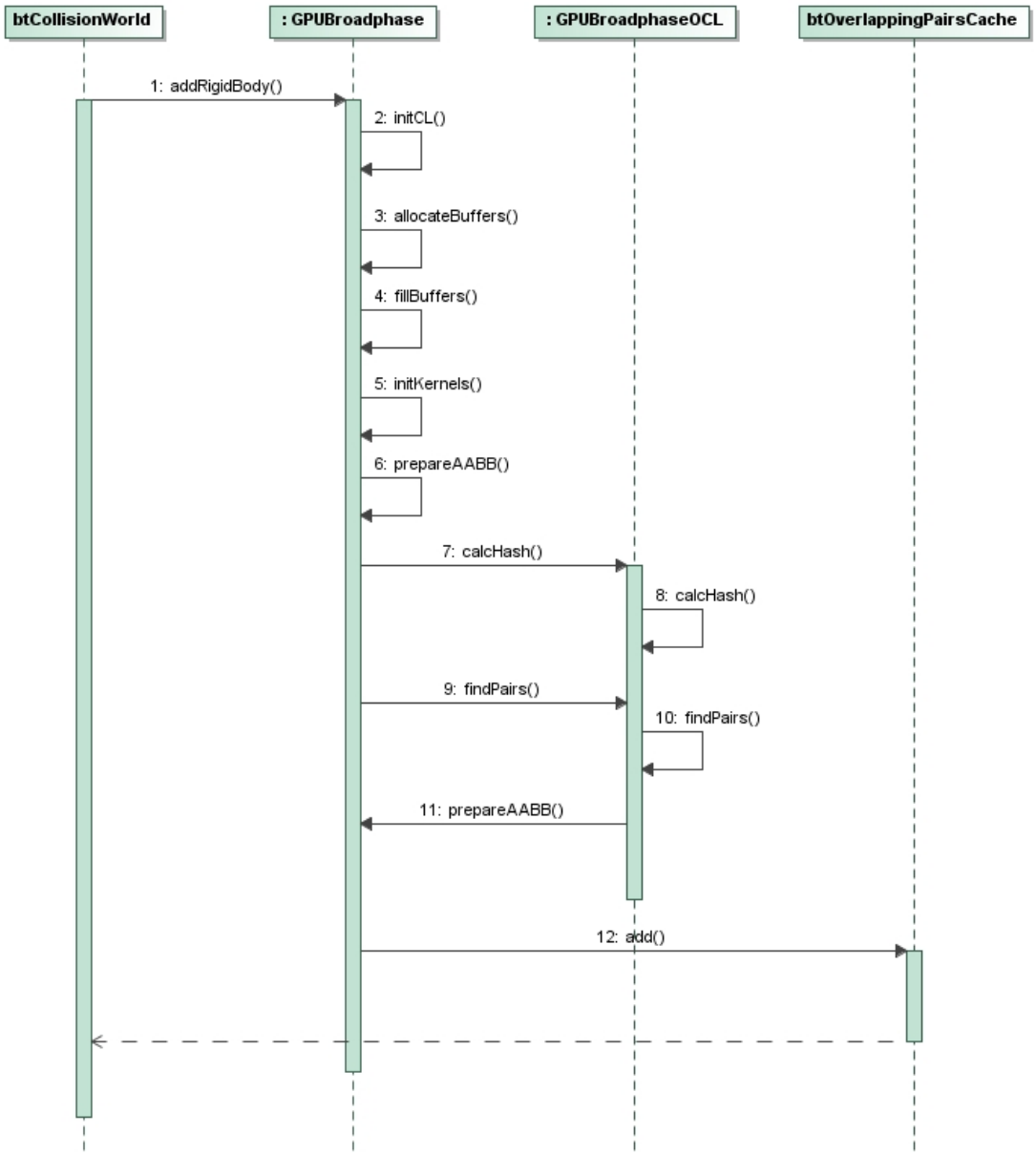
Paleidus simuliaciją pirmiausia įvykdoma reikalingų komponentų sukūrimas. Sukuriami pelės valdymo, objektų spalvų ir simuliacijos pasaulio komponentai. Pav. 6 pavaizduotas detalus simuliacijos sukūrimas.



Pav. 6 Simuliacijos sukūrimo sekos diagrama

2.4.2 *Broadphase* skaičiavimų fazė

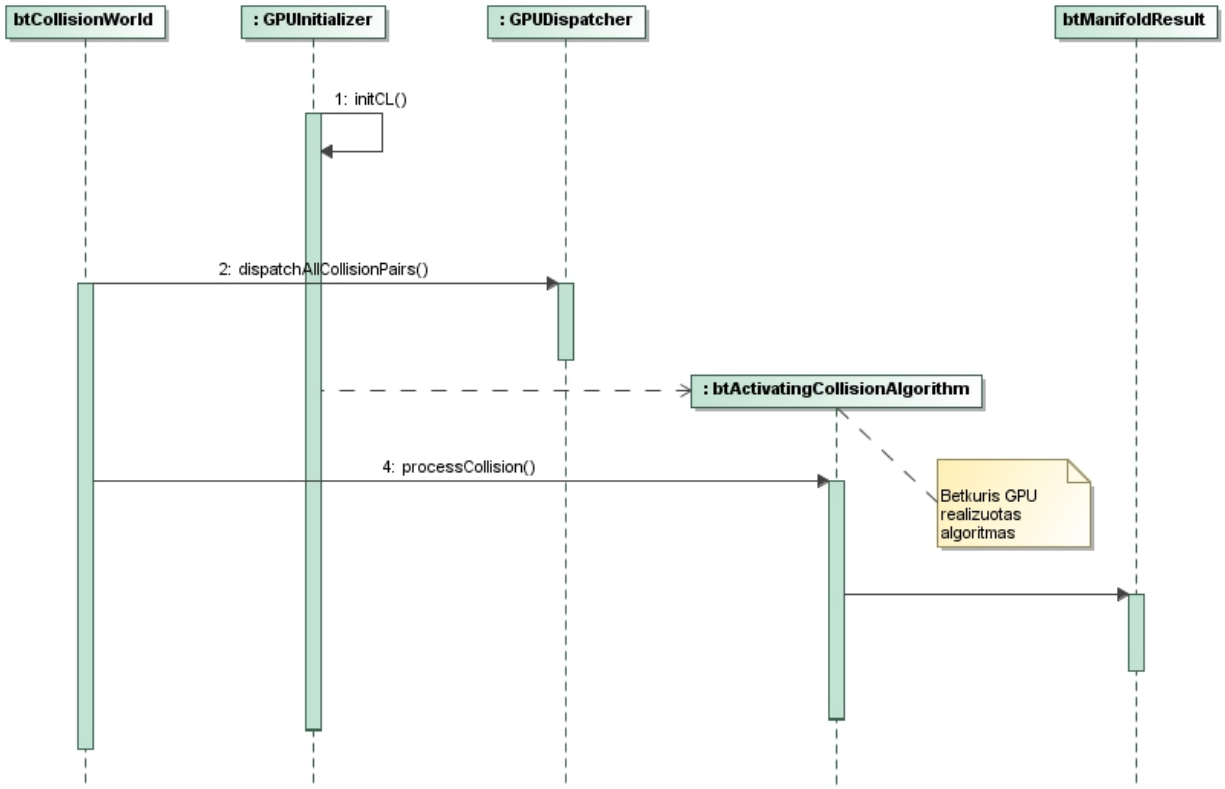
Pav. 7 pavaizduota *broadphase* skaičiavimų fazė. Fizikos pasaulis įkelia visus fizikos kūnus į GPUBroadphase klasės implementaciją. GPUBroadphase išsiskiria vietą vaizdo plokštėje ir atlieka skaičiavimus. Rasti susikirtimai įkeliami į btOverlappingPairsCache sąrašą.



Pav. 7 *Broadphase* fazės sekų diagram

2.4.3 *Narrowphase* saičiavimų fazė

Pav. 8 pavaizduota *narrowphase* skaičiavimų fazė. *Broadhpase* fazėje paskaičiuoti galimi kūnų susidūrimai apdorojami pagal tai, kokie objektai susidūrė. Tikslūs susidūrimo taškai įrašomi į btManifoldResult objektą.



Pav. 8 *Narrowphase* fazės sekų diagram

3 SPRENDIMO REALIZACIJA IR TESTAVIMAS

3.1 Realizacijos ir veikimo aprašas

Sistemos spartos didinimas išskaidomas į dvi dalis:

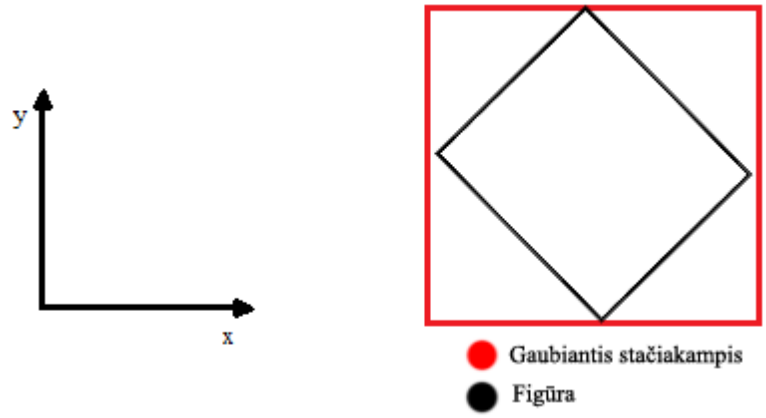
- *Broadphase* fazės spartos didinimas
- *Narrowphase* fazės spartos didinimas

Šių dalių detalus aprašymas pateikiamas tolimesniame skyriuje.

3.2 Sistemos tobulinimo galimybės

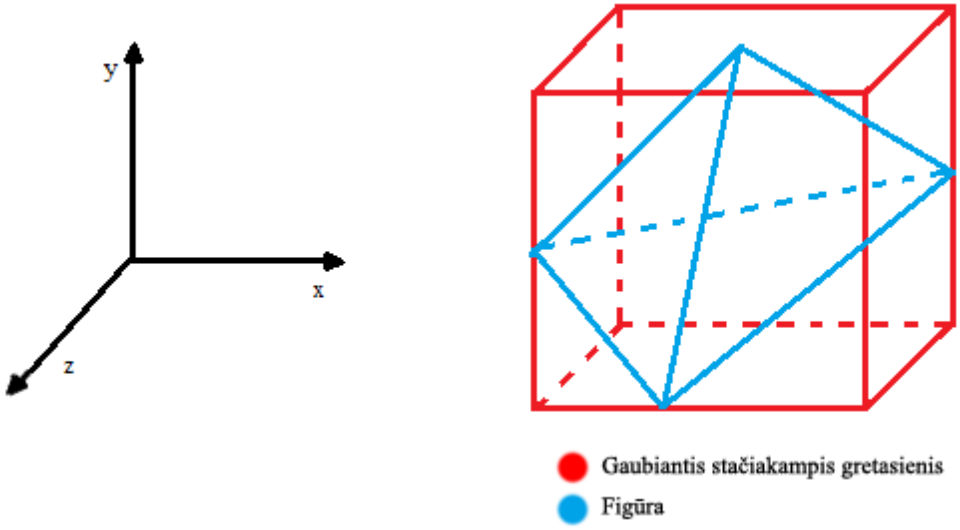
Dabartinė Bullet fizikos bibliotekos versija nepalaiko keleto gijų skaičiavimų. To neatlieka ir bet kokia kita populiari, atvirojo kodo nemokama biblioteka. Šio tyrimo metu mėginsime padaryti, kad šios bibliotekos fizikos skaičiavimai būtų atliekami naudojant ne tik keletą gijų, tačiau ir naudojant vaizdo plokščių procesorius.

Pirmiausia atskyrėme vietas, kuriose yra galimas sistemos sulėtėjimas. Tai yra *broadphase* ir *narrowphase* skaičiavimų fazės. Pirmojoje yra atliekamas grubus objektų susikirtimų įvertinimas. Objektai yra įvertinami labai grubiai t.y. jeigu objektas yra sfera, imamas šios sferos gaubiamasis stačiakampis gretasienis pagal ašis (AABB – axis aligned bounding box). Pav. 9 pavaizduotas kvadratas, kuris yra gaubiamas stačiakampio.



Pav. 9 Dvimatis figūros gaubimas

Pav. 10 pavaizduotas trimačio objekto gaubimas stačiakampiu gretasieniu.



Pav. 10 Trimatis figūros gaubimas

Toks procesas atliekamas su visais objektais. Tada atliekamas skaičiavimas su šiais duomenimis. Taip filtruojami objektai, kurių gali prireikti tikslesniems skaičiavimams atlikti. Šio filtravimo skaičiavimus ir galutinių duomenų perdavimą galima atlikti ir su alternatyviu skaičiavimų procesoriumi pvz. vaizdo plokštės procesoriumi.

Antroji vieta, kurią galima pakeisti yra *narrowphase* fazė. Šioje fazėje turime jau išfiltruotų objektų sąrašus. Iš sąrašo imami objektai, kurie galbūt susikerta (nėra tiksliai žinoma, nes pirmoji fazė nepaskaičiuoja tiksliai) ir pagal jų tikruosius tipus skaičiuojamas susikirtimo taškas. Jeigu abu objektai yra sferos, taikomas vienas skaičiavimo algoritmas, jeigu vienas objektas sfera, o kitas kubas tada taikomas kitas skaičiavimo algoritmas. Šie skaičiavimai taip pat atliekami naudojant pagrindinį sistemos procesorių. Galima pakeisti šiuos skaičiavimus ir atlikti juos naudojas kitus resursus. Pagrindinius skaičiavimo algoritmus pakeisime atitinkamais algoritmais, kurie naudoja vaizdo plokščių procesorius.

Tyrimo tikslas yra įsitikinti ar išlygiagretinus skaičiavimo algoritmus įmanoma pasiekti geresnių skaičiavimo rezultatų. Taip pat reikėtų įvertinti situacijas, kuriose panaudojamas šis pakeitimas. Stipriai vaizdo plokštę apkraunantis žaidimas gali dar labiau sulėtinti skaičiavimų rezultatus ir duoti neigiamą rezultatą. Tuo tarpu jeigu vaizdo plokštė nėra apkrauta rezultatas turėtų būti teigiamas.

3.3 Siūlomų patobulinimų pagrindimas

Sparčiausiai turėtų būti skaičiuojamas toks fizikos objektų kiekis, kiek vaizdo plokštėje yra nedirbančių branduolių skaičius. Pvz. Geforce 8600GT vaizdo plokštė turi daugiau nei 20 branduolių, kurie nepriklausomai vienas nuo kito gali atlikti skaičiavimus. Jeigu jie nėra užimti, skaičiavimų rezultatų sparta yra didžiausia. Didėjantis objektų skaičius turės neigiamos įtakos skaičiavimų spartai. Kai objektų skaičius yra daugiau už laisvų skaičiavimo branduolių skaičių, susidarys skaičiavimų eilė, ir reikės laukti kol atsilaisvins kuris nors branduolys tolimesniam skaičiavimui atlikti.

Panaši situacija susidaro vaizdo plokštei atliekant papildomus skaičiavimus t.y. ne vien tik fizikos skaičiavimus. Beveik visada pasitaikys būtent tokios situacijos. Fizikos simuliacija dažniausiai atvaizduojama daugiau ar mažiau detaliu vaizdu. Jeigu tokia biblioteka būtų naudojama žaidimuose, kurie reikalauja daug vaizdo plokštės resursų, gaunamas rezultatas suprastėtų arba išvis netektų naudos. Tačiau yra kitų panaudojimo sričių, kuriose atvaizdavimas nėra atvaizdavimas nėra pagrindinis resursus naudojantis skaičiavimas. Trimačio modeliavimo programinėje įrangoje dažniausiai atvaizduojamas tik eskizai, vaizdo plokščių resursai yra neišnaudojami. Tokioje situacijoje bibliotekos naudojimas programinės įrangos kūrėjui padėtų paspartinti savo įrankį.

3.4 Sistemos testavimas, testavimo rezultatai

Spartai matuoti sukuriami spartos testai:

- Testuojamas nurodytų sistemos žingsnių kiekio skaičiavimų laikas. Šis testavimo metodas tiksliausiai parodo spartos skirtumą, nes vaizdo plokštė nėra apkrauta.
- Testuojamas kadrų per sekundę skaičius kai atvaizduojami objektai atvaizduojami nedideliu kiekiu poligonų. Šis testavimo metodas padės nustatyti kokią įtaka fizikos skaičiavimams gali turėti nedidelis vaizdo plokštės apkrautumas.
- Testuojamas kadrų per sekundę skaičius kai atvaizduojami objektai atvaizduojami dideliu kiekiu poligonų. Toks atvaizdavimo būdas stipriai apkrauna vaizdo plokštę, o tuo pačiu metu ji dar turi skaičiuoti ir fiziką. Šis metodas padės nustatyti kokią įtaką fizikos skaičiavimams turi didelis vaizdo plokštės apkrautumas.

Taip pat kuriami testai programos teisingumui patikrinti naudojant baltosios dėžės metodiką. Lentelėje 11 aprašomi testų atliekami tikrinimai.

Lentelė 11 Testų metodų aprašymai

Metodas	Aprašymas
testCompileCLProgram	Patikrinama, ar sukurtas OpenCL kodas yra kompiliuojamas sėkmingai.
testOverlappingPair	Tikrinama kiek išvis kūnų susikerta
testPosition	Patikrinamos kūnų pozicijos. Tikrinama ar jos yra teisingos pradžioje simuliacijos ir simuliacijos pabaigoje
testPositionAfter100Frames	Patikrinamos fizikinių kūnų pozicijos po 100 žingsnių simuliacijos
testPerformanceTestGPU	Atliekami testai su norimais parametrais naudojant OpenCL ir grafikos procesorių. Šiame teste taip pat imituojamas realus atvejis, kuriame vartotojui rodomas ir vaizdas
testPerformanceTestCPU	Atliekami testai su norimais parametrais naudojant OpenCL ir grafikos procesorių. Šiame teste taip pat imituojamas realus atvejis, kuriame vartotojui rodomas ir vaizdas
testPerformanceTestGPUNoRendering	Atliekami testai su norimais parametrais Naudojant OpenCL ir grafikos procesorių. Matuojami tik fizikos skaičiavimai. Vartotojui nepateikiamas joks vaizdas, tik galutinis rezultatas, per kiek laiko buvo aprodotas nurodytas simuliacijos žingsnių skaičius
testPerformanceTestCPUNoRendering	Atliekami testai su norimais parametrais naudojant standartinį metodą. Matuojami tik fizikos skaičiavimai. Vartotojui nepateikiamas joks vaizdas, tik galutinis rezultatas, per kiek laiko buvo aprodotas nurodytas simuliacijos žingsnių skaičius
testCreateSpheres	Pagalbinio metodo, sukuriančio sferas testas
testCreateBox	Pagalbinio metodo, sukuriančio kubus testas

Testų rezultatai saugomi XML failuose. Failo pavadinime nurodyta testų paleidimo data. Pačiuose XML failuose suformuoti rezultatai, pavaizduota ar testai praėjo sėkmingai. Keleto testų rezultatų pavyzdys pateikiamas žemiau:

```
<TestRun>
  <FailedTests></FailedTests>
  <SuccessfulTests>
    <Test id="1">
      <Name>ManoTestKlase::testCompileCLProgram</Name>
    </Test>
    <Test id="2">
      <Name>ManoTestKlase::testOverlappingPair</Name>
    </Test>
    <Test id="3">
      <Name>ManoTestKlase::testPosition</Name>
    </Test>
    <Test id="4">
      <Name>ManoTestKlase::testPositionAfter100Frames</Name>
    </Test>
  </SuccessfulTests>
  <Statistics>
    <Tests>4</Tests>
    <FailuresTotal>0</FailuresTotal>
    <Errors>0</Errors>
    <Failures>0</Failures>
  </Statistics>
</TestRun>
```

4 EKSPERIMENTINIS SPRENDIMO TYRIMAS

4.1 Eksperimento planas

Numatytas vykdomų eksperimentų planas:

- Fizikos bibliotekos
- PĮ bandymai su įvairiomis sistemų konfigūracijomis ir scenomis
- Alternatyvių fizikos PĮ bandymas
- Atliktų bandymų su fizikos bibliotekomis spartos matavimas
- Atliktų bandymų su fizikos bibliotekomis rezultatų palyginimas

Šie numatyti žingniai įgyvendinami eksperimento dalyje.

4.2 Naudota aparatinė įranga

Tyrimui atlikti buvo naudojama įranga:

Pirmasis kompiuteris:

- Geforce 8600GT ir Geforce GT240 vaizdo plokštės
- 3.2Ghz AMD Phenom Quad–Core procesorius
- 3.5GB operatyviosios atminties
- 750GB kietasis diskas

Antrasis kompiuteris:

- Geforce GTX560
- 3.8Ghz Intel core i5–3210
- 8GB operatyviosios atminties
- 500GB kietasis diskas

Skaičiavimų rezultatai remiasi vaizdo plokščių pajėgumais. Lentelėse nr. 12, 13 ir 14 pateikiama reikalinga išsamesnė informacija apie minėtas vaizdo plokštes.

Lentelė 12 Geforce 8600GT vaizdo plokštės pagrindinės charakteristikos

Geforce 8600GT	
Procesorių skaičius	32
Vaizdo atmintis	256 MB
Šerdies dažnis	540 MHz
Grafikos dažnis	1180 MHz
Atminties dažnis	700 MHz
CUD įvertis	1.1

Lentelė 13 Geforce GT240 vaizdo plokštės pagrindinės charakteristikos

Geforce GT240	
Procesorių skaičius	96
Vaizdo atmintis	1024 MB
Šerdies dažnis	550 MHz
Grafikos dažnis	1360 MHz
Atminties dažnis	1000 MHz
CUDA įvertis	1.2

Lentelė 14 Geforce GTX 560 vaizdo plokštės pagrindinės charakteristikos

Geforce GTX 560	
Procesorių skaičius	336
Vaizdo atmintis	1024 MB
Šerdies dažnis	810 MHz
Grafikos dažnis	1620 MHz
Atminties dažnis	2004 MHz
CUDA įvertis	2.0

4.3 Eksperimento rezultatai

Atliekant eksperimentą tikrinama fizikos simuliacijos skaičiavimų sparta. Sistema turi sėkmingai dirbti ir su nedideliu objektų skaičiumi ir su dideliu objektų skaičiumi.

Pirmiausiai atliekamas bibliotekų skaičiavimų spartos palyginimas. Sistemų palyginimui sukuriamos vienodos sąlygos:

- Vienodas fiksuotas simuliacijos žingsnis (0,016 ms).

- Vienodas objektų skaičius (nuo 250 iki ribų, kurias palaiko pasirinktoji biblioteka)
- Vienodas leidžiamų susilietimų kiekis vienam objektui(10)
- Galimas minimalus atvaizdavimas, kuris neįtakoja skaičiavimo rezultatų.

Pasirinkome dar dvi fizikos skaičiavimų bibliotekas. Jos abi taip pat parašytos ta pačia C++ programavimo kalba.

Sukuriama demonstracinė scena. Norint įvertinti bibliotekos skaičiavimų spartą, kuo daugiau fizikinių kūnų turi liestis tarpusavyje. Tam panaudojame krintančių objektų matricą ant plokštumos. Taip sukuriami susijungimo taškai, einant laikui jų vis didėja ir apkrautumas sistemai didėja. Skaičiuojama tokios scenos 1000 žingsnių simuliacijos trukmės laikas.

Kiti eksperimentai vykdomi naudojant sukurtą biblioteką keliais skirtingais tipais: be atliekamo vaizdavimo ir su vaizdavimu esant vidutiniam ir dideliam vaizdo plokštės apkrovimui. Išsamiai ištiriama sistemos sparta su įgyvendinta PĮ. Apkrovimas atliekamas naudojant atvaizdavimo elementų kiekio(poligonų) didinimą vienam objektui atvaizduoti. Vidutinis apkrovimas naudoja 25 elementus vienam objektui atvaizduoti. Dideliam objektui atvaizduoti naudojama 500 elementų skaičius.

Svarbiausios eksperimentų matavimo metrikos – simuliacijos įgyvendinimo laikas ir kadrų per sekundę skaičius.

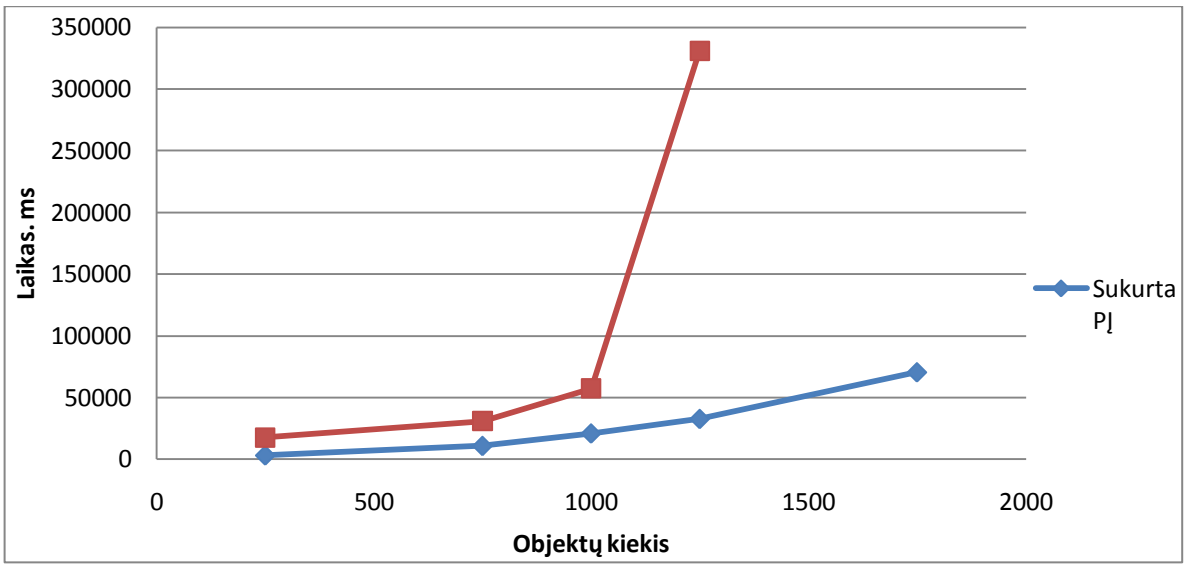
4.3.1 Palyginimas su kitomis fizikos bibliotekomis

Pasirinktos dar dvi populiarios, nemokamos, atvirojo kodo ir platinamos pagal zlib licenciją fizikos bibliotekos:

- ODE (Open Dynamics Engine)
- Newton Dynamics

Eksperimentams naudojama pirmasis kompiuteris su 8600 GT vaizdo plokšte. Paruošus ir atlikus demonstracinės ODE scenos tūkstančio žingsnių simuliacijas.

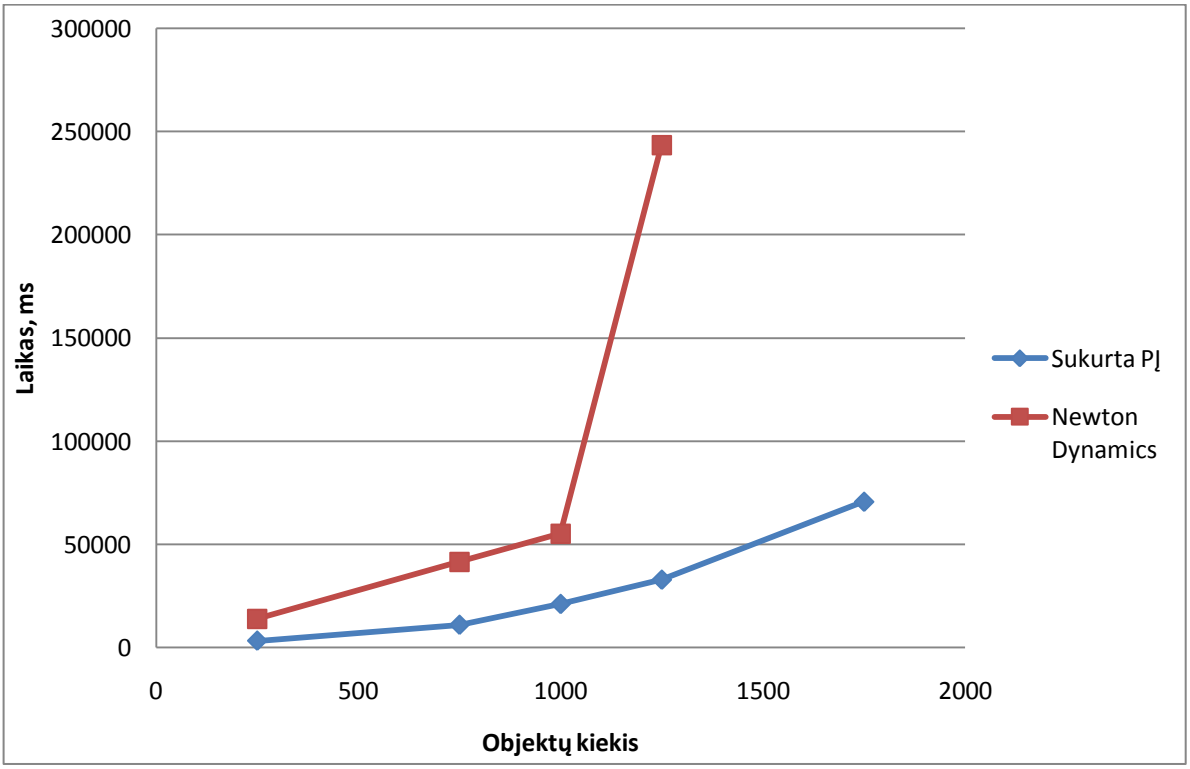
Naudojant šią biblioteką atkreipiamas dėmesys į simuliacijos tipus. Yra dviejų tipų simuliacija, pilnoji (dWorldStep) ir greitoji (dWorldQuickStep). Kadangi visoms sistemoms sudaromos vienodos sąlygos, naudojama pilnoji. Naudojant greitąją simuliaciją pasireiškia pašaliniai efektai, kaip objektų drebėjimas, netikslus susikirtimo taškų paskaičiavimas, todėl naudojama pilnoji simuliacija. Pav. 11 pavaizduota sukurtos PĮ ir ODE simuliacijos spartos grafikas.



Pav. 11 Sukurtos PĮ ir ODE bibliotekų spartos palyginimas

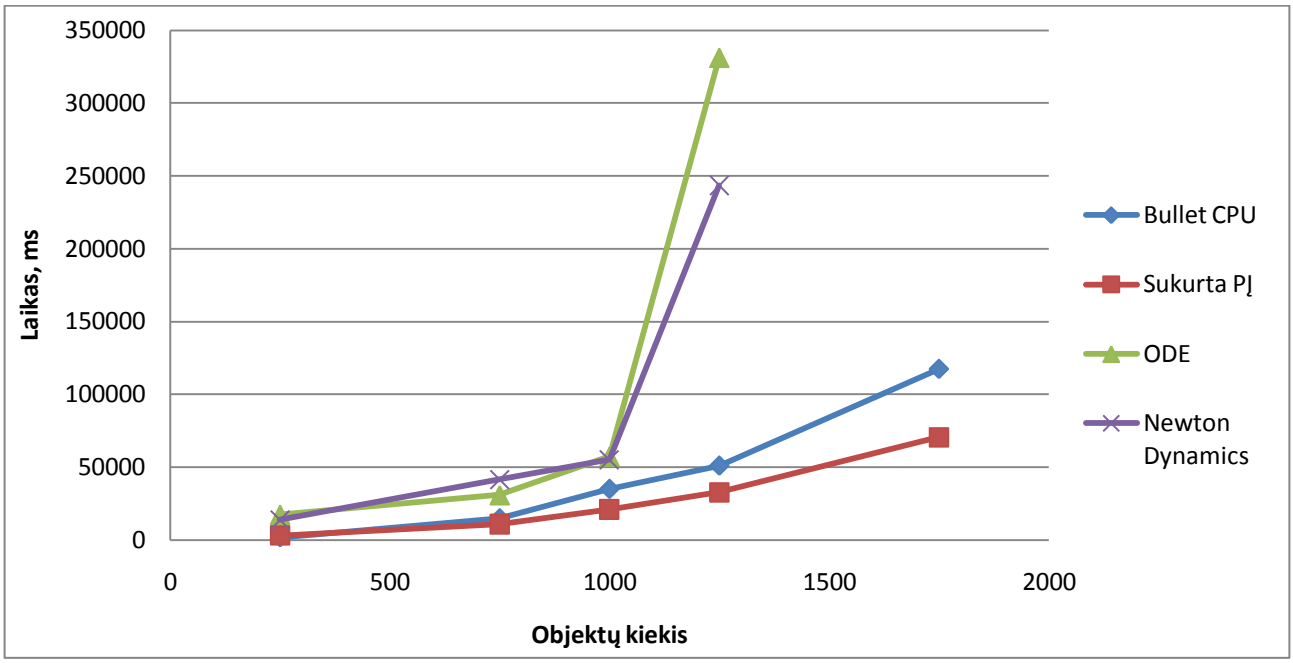
Iš grafiko matoma, kad ODE bibliotekos skaičiavimai nuo ~800 objektų scenoje kiekio pradėjo stipriai lėtėti. ODE sistema neleido panaudoti daugiau nei 1300 elementų kiekių scenoje.

Toks pat eksperimentas buvo atliktas ir su Newton Dynamics sistema. Ji neturi papildomų nustatymų išjungti pilnąja simuliaciją. Pav. 12 pavaizduota sukurtos PĮ ir Newton Dynamics simuliacijos spartos grafikas.



Pav. 12 Sukurtos PĮ ir Newton Dynamics bibliotekų spartos palyginimas

Pav. 13 pavaizduota visų matuotų sistemų spartos grafikas. Atvaizduojami Bullet spartos rezultatai naudojant CPU, sukurtos sistemos rezultatai ir ODE bei Newton Dynamics spartos rezultatai.



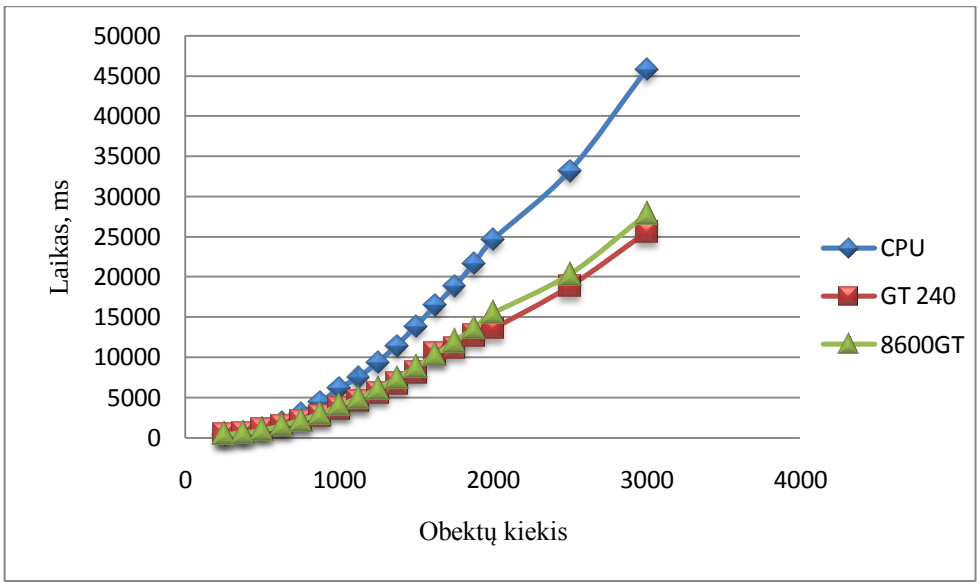
Pav. 13 Visų bibliotekų spartų palyginimas

Bullet net ir naudojant tik CPU buvo greitesnė nei ODE. Spartos skirtumas įdiegus sistemos pakeitimus tik padidėjo. Iš grafiko matoma, kad sukurta PĮ veikia 6 kartus greičiau, nei ODE ar Newton Dynamics. Šios bibliotekos nepalaikė daugiau negu 1200 objektų scenoje.

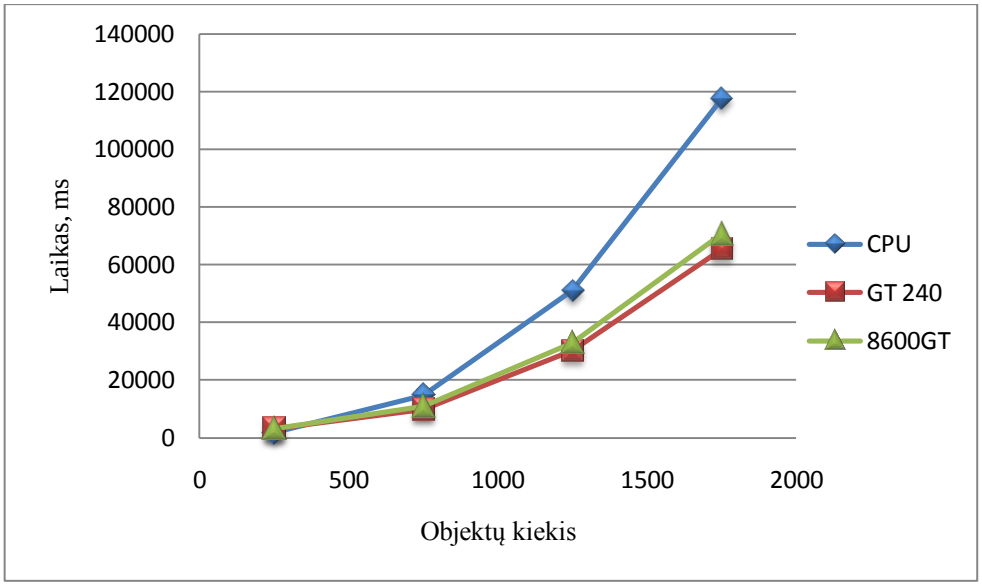
4.3.2 Eksperimentai be vaizdavimo

Atliekamas išsamus sukurtos PĮ eksperimentą didinant objektų skaičių scenoje. Pirmasis šio tipo eksperimentas buvo atliekamas daugumai scenoje esančių objektų liečiantis tarpusavyje visą simuliacijos laiką. Atliekama 100 žingsnių simuliacija. Antrasis šio tipo eksperimentas buvo atliekamas kai scenoje objektai dalyvavo kolizijoje bent kartą. Objektų susikirtimų skaičius apie 10% simuliacijos laiko. Tai reiškia, kad objektai buvo daugiau pasiskirstę scenoje. Atliekama 1000 žingsnių simuliacija.

Pav. 14 ir pav. 15 pavaizduotos laiko priklausomybės nuo objektų kiekio esančių scenoje naudojant pirmąją sistemą. Eksperimentas atliktas su skirtingomis vaizdo plokštėmis.



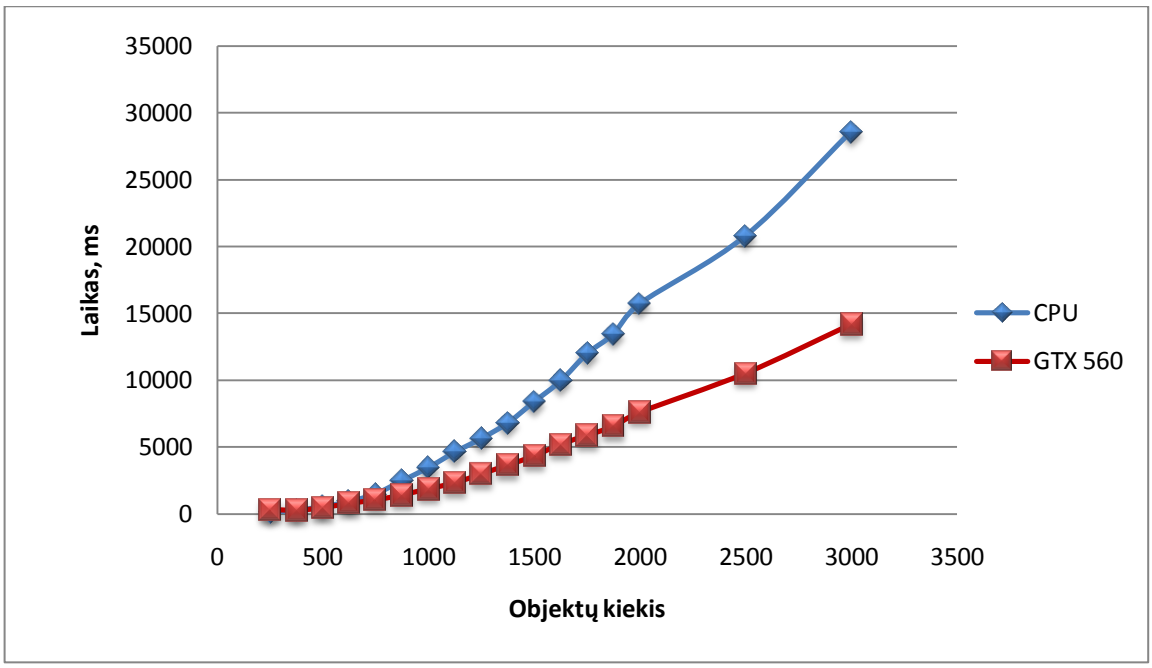
Pav. 14 Pirmosios sistemos 100 žingsnių simuliacijos laiko priklausomybė nuo objektų kiekio



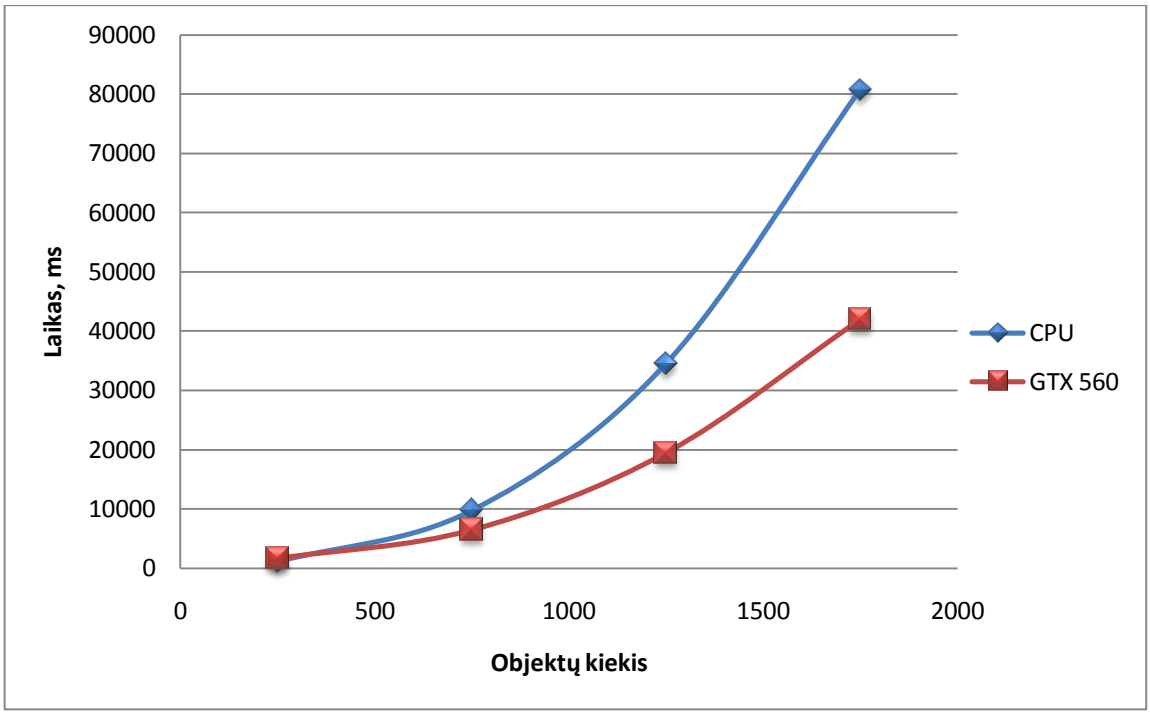
Pav. 15 Pirmosios sistemos 1000 žingsnių simuliacijos laiko priklausomybė nuo objektų kiekio

Siekiamas rezultatas, kuo greitesnis simuliacijos paskaičiavimas. Iš grafikų galima daryti išvadą, kad kuo daugiau objektų yra scenoje, tuo greičiau PĮ veikia naudojant vaizdo plokštes. Skaičiavimų spartos skirtumas tarp pagrindinio procesoriaus (CPU) ir naudotų vaizdo plokščių yra ~ 70 – 80%. Tačiau nėra tokio didelio skirtumo tarp Geforce GT 240 ir Geforce 8600GT skaičiavimo spartos duotajam uždaviniui. Spartos skirtumas sudaro 10% laiko. CUDA įvertis minėtoms vaizdo plokštėms skiriasi 0.1 (Geforce 8600 GT – 1.1, Geforce GT 240 – 1.2).

Pav. 16 ir pav. 17 pavaizduotos laiko priklausomybės nuo objektų kiekio esančių scenoje naudojant antrąją sistemą.



Pav. 16 Antrosios sistemos 100 žingsnių simuliacijos laiko priklausomybė nuo objektų kiekio



Pav. 17 Antrosios sistemos 1000 žingsnių simuliacijos laiko priklausomybė nuo objektų kiekio

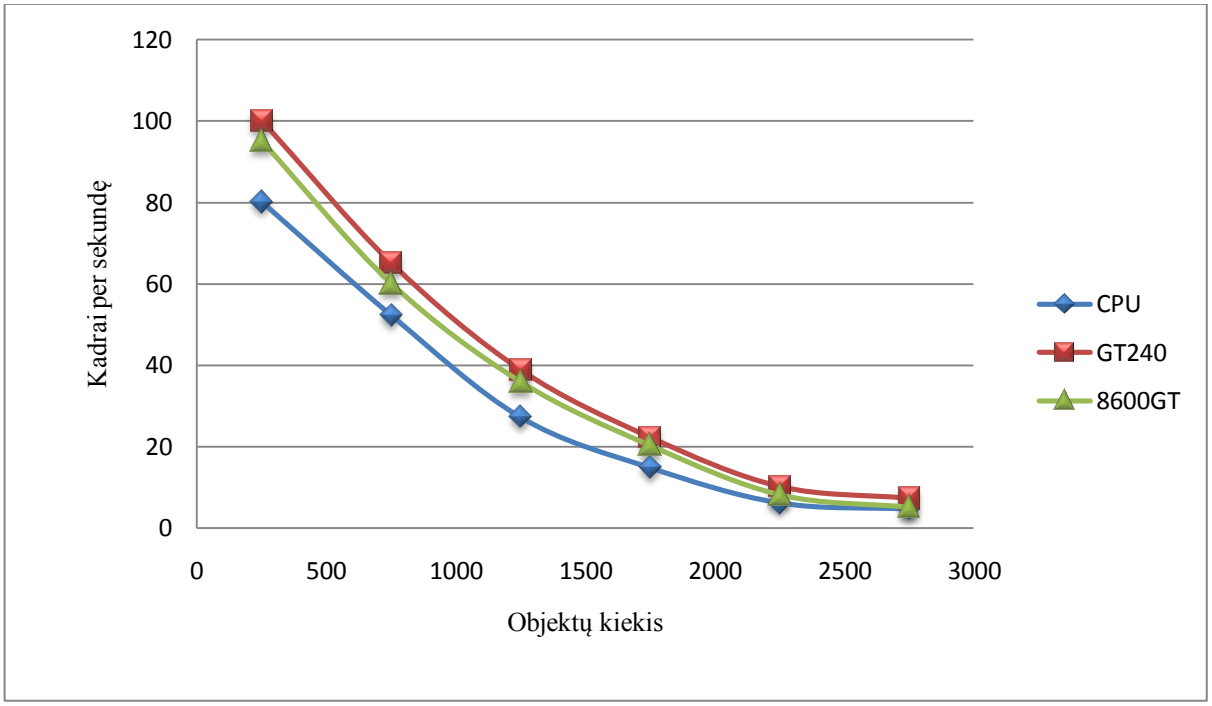
Antrajai sistemai eksperimentas buvo atliekamas tik su viena vaizdo plokšte. Skaičiavimų spartos skirtumas tarp pagrindinio procesoriaus (CPU) ir vaizdo plokštės sudarė ~ 100%. Pirmoji sistema 1000 žingsnių simuliaciją naudojant vaizdo plokštes įgyvendino per ~70 sekundžių, antroji sistema tai padarė per 40 sekundžių. Tai sudaro beveik dviejų kartų skirtumą tarp šių sistemų. Šie duomenys atitinka CUDA įverčius (antrosios sistemos CUDA įvertis 2.0, tuo tarpu pirmosios 1.1 ir 1.2).

4.4 Eksperimentai su vaizdavimu

Pirmasis šio tipo eksperimentas buvo atliekamas daugumai scenoje esančių objektų liečiantis tarpusavyje visą simuliacijos laiką. Taip pat ši simuliacija atvaizduojama naudojant OpenGL. Scenai atvaizduoti naudojama aplinka, kurioje yra apšvietimas, šešėliai ir pan. Vienam objektui atvaizduoti naudojami 25 elementai. Atliekama 100 žingsnių simuliacija.

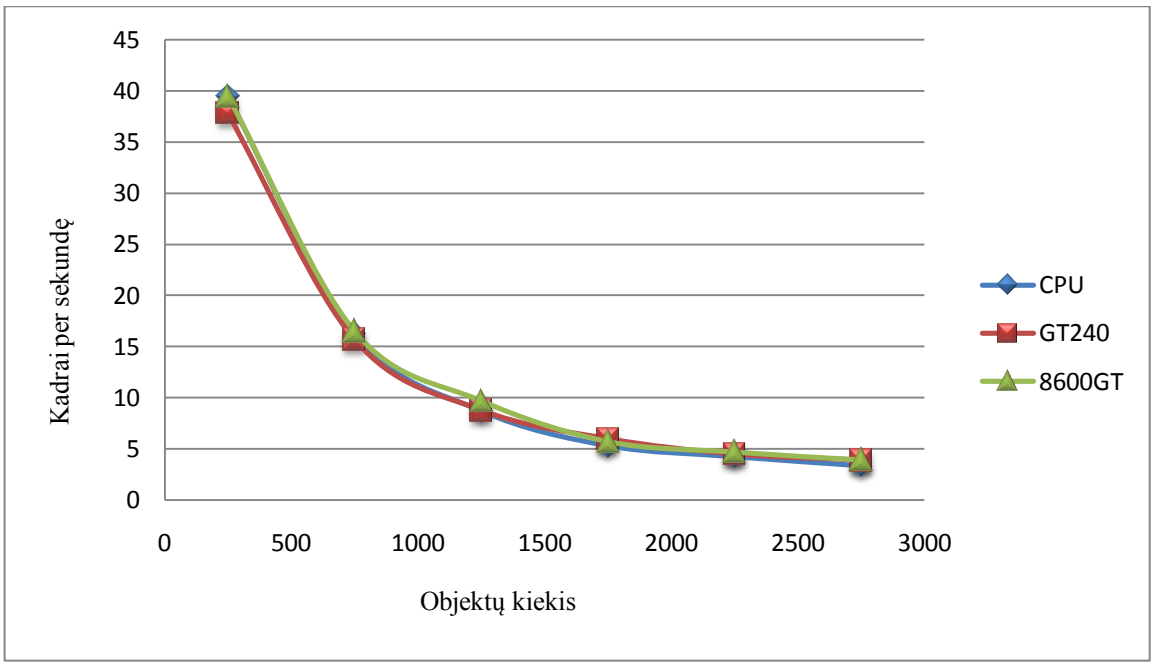
Antrasis šio tipo eksperimentas atliekamas su 1000 žingsnių simuliacija objektų atvaizdavimo elementų kiekį padidinus iki 400 vienam objektui. Taip randama skaičiavimų sparta vaizdo plokštei esant stipriai apkrautai.

Pav. 18 ir pav. 19 pavaizduotos laiko priklausomybės nuo objektų kiekio esančių scenoje naudojant pirmąją sistemą. Eksperimentas atliktas su skirtingomis vaizdo plokštėmis.



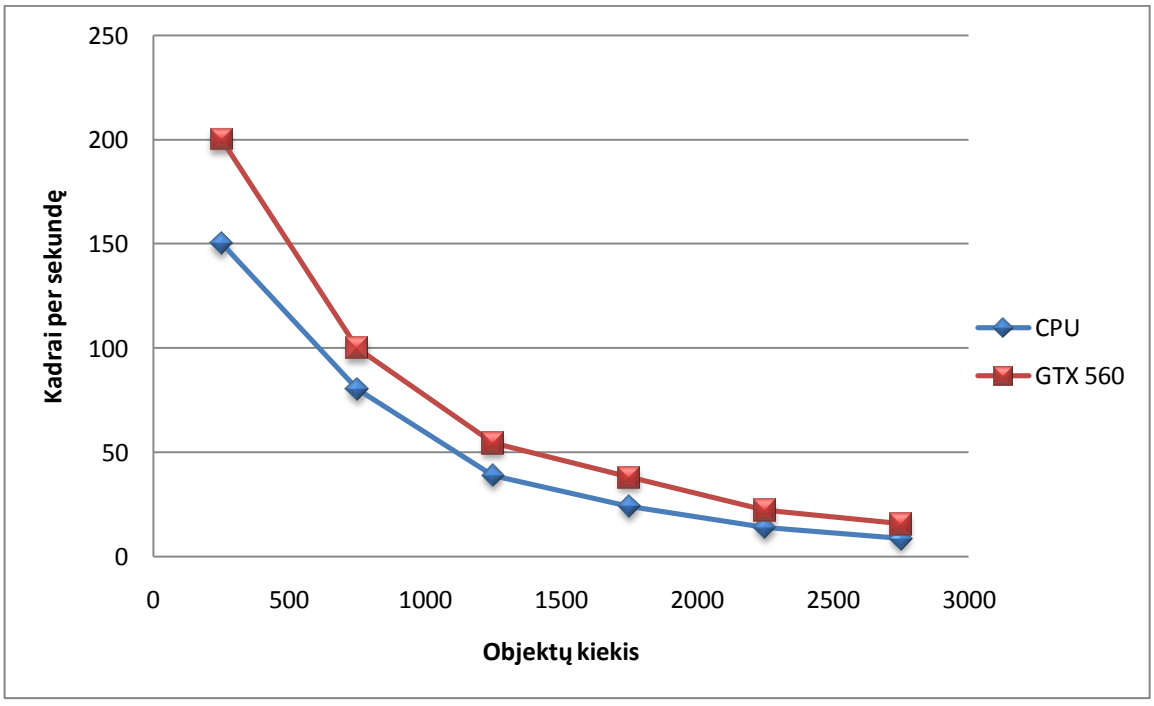
Pav. 18 Pirmosios sistemos kadrų per sekundę priklausomybė nuo objektų kiekio (25 elementai vienam objektui)

Esant nedidelei sistemos apkrovai spartos skirtumas nėra toks akivaizdus. Pirmosios sistemos sparta naudojant vaizdo plokštes padidėjo ~20%. Šiems rezultatams įtakos turi vaizdo plokštės užimtumas atliekant kitus skaičiavimo darbus vaizduojant sceną.



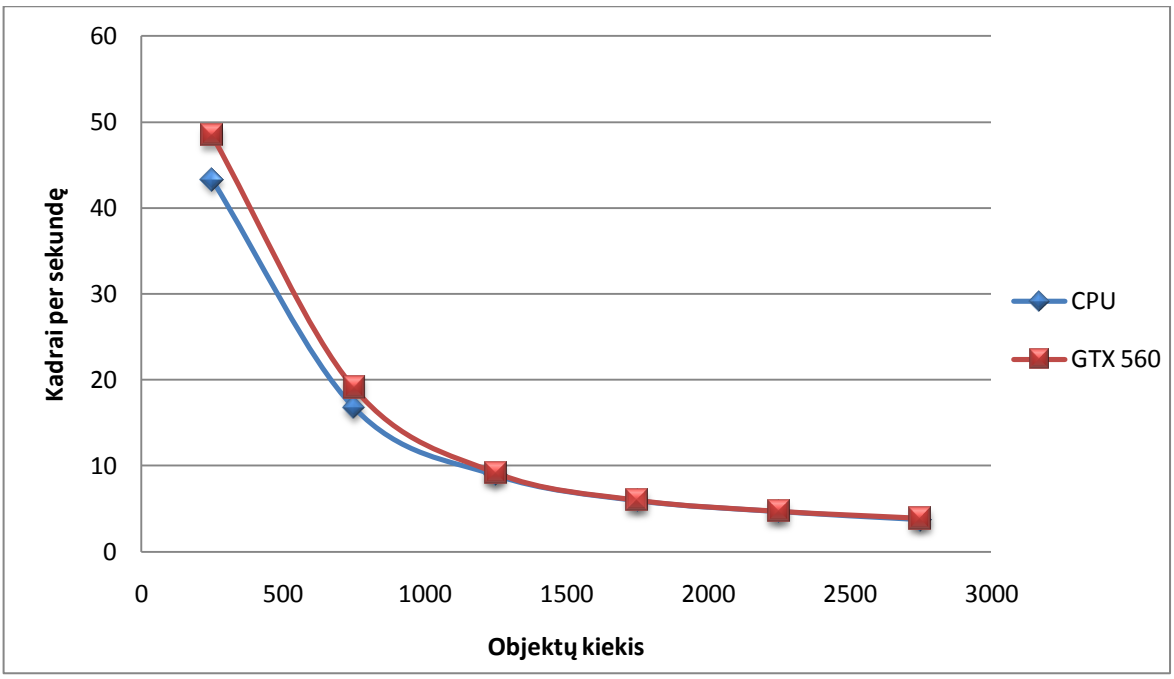
Pav. 19 Pirmosios sistemos kadrų per sekundę priklausomybė nuo objektų kiekio (400 elementų vienam objektui)

Esant didelei sistemos apkrovai pirmoji sistema veikė panašia sparta visomis naudotomis konfigūracijomis. Šiame eksperimente scenoje elementų kiekis sudarantis objektus siekė nuo 250 tūkst. iki 1 mln. elementų. Toks didelis elementų skaičius sudarantis scenas pirmosios kartos vaizdo plokštes apkrauna maksimaliai.



Pav. 20 Antrosios sistemos kadrų per sekundę priklausomybė nuo objektų kiekio (25 elementų vienam objektui)

Antroji sistema, kaip ir pirmoji esant nedidelei vaizdo plokštės apkrovai veikė ~20% greičiau nadojant vaizdo plokštę nei pagrindinį procesorių. Tai pastebima visų simuliacijos leidimų metu.



Pav. 21 Antrosios sistemos kadrų per sekundę priklausomybė nuo objektų kiekio (400 elementų vienam objektui)

Esant didelei vaizdo plokštės apkrovai antroji sistema veikė santykinai greičiau nei pirmoji sistema. Tai stebima dėl spartesnės sistemos vaizdo plokštės. Kritinis taškas, nuo kurio sistema sulėtėja yra 1250 objektų scenoje. Šiame taške atvaizdavimo scena sudaryta iš 0,5 mln. atvaizdavimo elementų, apšvietimo, šešėlių ir pan. Toks vaizdo plokštės apkrautumas nebeleidžia sėkmingai išnaudoti jos pajėgumo fizikos uždavinio skaičiavimams atlikti.

4.5 Sprendimo veikimo ir savybių analizė, kokybės kriterijų įvertinimas

Išanalizavus atliktų eksperimentų rezultatus matome, kad sukurta sistema daugiausia atvejų veikia teisingai t.y. atlieka skaičiavimus naudojant GPU greičiau negu CPU.

Didžiausias skaičiavimų pagreitėjimas stebimas be scenos atvaizdavimo. Be sistemos atvaizdavimo pasiekiamas 100% spartos pagreitėjimas. Su atvaizdavimu sistema veikė 20-50% greičiau priklausomai nuo skirtingų kompiuterių pajėgumų. Naudojant labai daug elementų scenai apkrauti sistema spartos pagreitėjimo nepasiekė.

Numatytieji sistemos kokybės kriterijai įvertinti 15 lentelėje.

Lentelė 15 Kokybės kriterijų įvertinimas

Kokybės kriterijai	**Rezultatas**
Spartos padidėjimas	Sistema veikia ne lėčiau negu naudojant pradinę implementaciją.
Aiški hierarchija, vardai	Klasių ir metodų vardai atitinka jų atliekamus skaičiavimus.

4.6 Sprendimo taikymo rekomendacijos

Sukurtą sistemą galima naudoti ir jau sukurtuose produktuose ir dar tik kuriamuose. Sistemos vartotojui reikia pakeisti tik keletą kodo eilučių esamoje sistemoje, kad pakeitimai pradėtų veikti. Sistemos dalį, kurioje jau suderinti demonstraciniai failai galima pasileisti ir vartotojui neturinčiam programavimo žinių.

PĮ geriausia taikyti aplinkoje, kuri nenaudoja daug vaizdo plokštės resursų pvz. 3D modeliavimo programinėje įrangoje situacijose nereikalaujančiose intensyvaus vaizdo plokštės darbo, tačiau ir esant vaizdo plokštės apkrovimui gaunamas rezultatas yra geresnis, negu numatytoji programinės įrangos realizacija.

REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Išanalizavus esamas sistemas ir galimybes paskirstyti skaičiavimus paaiškėjo, kad galima atskiras fizikos skaičiavimų dalis iškelti ir panaudoti skaičiavimams vaizdo plokštėje atlikti.
2. Realizavus sistemos prototipą paaiškėjo, kad nėra galimybės visų fizikos skaičiavimų atlikti naudojant vien vaizdo plokštę. Pagrindinis procesorius turi dirbti išvien su vaizdo plokšte.
3. Įgyvendinta sistema atitinka sistemos reikalavimus. Išpildyti ir funkciniai ir nefunkciniai sistemos reikalavimai.
4. Palyginus alternatyvias fizikos bibliotekas ir sukurtą paaiškėjo, kad pradinė pasirinkta sistema veikė greičiau už abi alternatyvias bibliotekas, o atlikus pakeitimus, sistema tapo dar greitesnė.
5. Atlikus eksperimentą paaiškėjo, kad tokią pačią skaičiavimų logiką atliekantys algoritmai sėkmingai veikia ir vaizdo plokštėje.
6. Pastebėta, kad vaizdo plokščių užimtumas įtakoja bendrą simuliacijos skaičiavimų laiką. Esant labai mažam vaizdo plokštės apkrautumui fizikos skaičiavimų sparta pagreitėjo apie 100%. Esant vidutiniam apkrautumui sparta išaugo apie 20%. O esant labai dideliam apkrautumui sistema veikė tokiu panašiu greičiu, kaip ir naudojant pagrindinį procesorių.

LITERATŪRA

[1] Nickolls, J., & Dally, W. J. (2010). The GPU computing era. Micro, IEEE, 30(2), 56-69.

[2] Fan, Z., Qiu, F., Kaufman, A., & Yoakum-Stover, S. (2004, November). GPU cluster for high performance computing. In Proceedings of the 2004 ACM/IEEE conference on Supercomputing (p. 47). IEEE Computer Society.

[3] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. Proceedings of the IEEE, 96(5), 879-899.

[4] Jayanth, G., Mendel R., Stream Processing in General-Purpose Processors. Stanford University, Stanford, CA 94305 [žiūrėta 2011 11 10], prieiga per internetą http://www.cs.utexas.edu/users/skeckler/wild04/Paper14.pdf

[5] Luebke, D., Harris, M., Govindaraju, N., Lefohn, A., Houston, M., Owens, J., ... & Buck, I. (2006, November). GPGPU: general-purpose computation on graphics hardware. In Proceedings of the 2006 ACM/IEEE conference on Supercomputing (p. 208). ACM

[6] Matt, P.; Randima, F. GPU Gems 2: Pogramming Techniques for High-Performance Graphics and General-Pupose Computation.

[7] NVIDIA CUDA [žiūrėta 2011 11 08], prieiga per internetą http://www.nvidia.com/object/cuda_home_new.html

[8] Apple Previews Mac OS X Snow Leopard to Developers [žiūrėta 2011 11 08], prieiga per internetą
http://www.apple.com/pr/library/2008/06/09Apple-Previews-Mac-OS-X-Snow-Leopard-to-Developers.html

[9] NVIDIA PhysX, Technology [žiūrėta 2011 11 08], prieiga per internetą http://www.geforce.com/hardware/technology/physx/technology

[10] Bulet Physics Simulation [žiūrėta 2011 12 08], prieiga per internetą http://bulletphysics.org/

[11] Zlib [žiūrėta 2011 11 08], prieiga per internetą http://www.zlib.net

[12] Open Dynamics Engine [žiūrėta 2011 12 01], prieiga per internetą http://www.ode.org/

[13] Newton Game Dynamics [žiūrėta 2011 12 01], prieiga per internetą http://newtondynamics.com/forum/newton.php

[14] The Problem(s) with GPGPU [žiūrėta 2011 11 08], prieiga per internetą http://blogs.intel.com/research/2007/10/the_problem_with_gpgpu.php

[15] Xiaohan, M., Mian, D., Lin, Z., Zhigang, D., Statistical Power Consumption Analysis and Modeling for GPU-based Computing [žiūrėta 2011 11 08], prieiga per internetą
http://www.sigops.org/sosp/sosp09/papers/hotpower_6_ma.pdf