ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Saulius Levaginas
Informatikos specialybės II kurso dieninio skyriaus studentas

Android programų kūrimo įrankių galimybių spręsti šilumos pernešimo uždavinius analizė

Analysis of the Tools for Android Applications in Heat Transfer Problem

MAGISTRO DARBAS

Darbo vadovė:
Doc. dr. S. Turskienė

Recenzentas:
Doc. dr. V. Sirius

Šiauliai, 2013

„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta naudojant literatūros sąraše pateiktus informacinius šaltinius bei savo tyrimų duomenis“

Darbo autoriaus _____

(vardas, pavardė, parašas)

Darbo tikslai ir uždaviniai

Darbo tikslas: Išanalizuoti „Android“ OS programų kūrimo įrankių galimybes, sprendžiant šilumos pernešimo uždavinį baigtinių elementų metodu (BEM).

Tikslui pasiekti išsikelti uždaviniai:

1. Programinių įrankių, naudojamų šio darbo tikslui pasiekti, pasirinkimo analizė.
2. Realizuoti plokštelės diskretizavimo ir vizualizavimo funkcionalumą.
3. Išspręsti šilumos pernešimo uždavinį BEM, pritaikant plokštelės diskretizavimo duomenis.
4. Ištestuoti sukurtą programinę priemonę.

Darbo vadovo _____
(vardas, pavardė, parašas)

Turinys

Įvadas

Šiame darbe analizuojamos „Android“ operacinės sistemos (OS) programų kūrimo įrankių galimybės, sprendžiant šilumos pernešimo uždavinį baigtinių elementų metodu (BEM). Toks siekis pasirinktas dėl dviejų priežasčių: 1) atlikus paiešką (paieškos sistemose, bibliotekose bei mokslinių straipsnių duomenų bazėse) tokios analizės nebuvo rasta; 2) išmaniųjų mobiliųjų įrenginių rinka yra naujai susiformavusi bei sparčiai auganti (atitinkamai ir jiems skirtos operacinės sistemos), tad šio darbo rezultatai galės pasitarnauti dar ankstyvoje technologijų vystymosi stadijose.

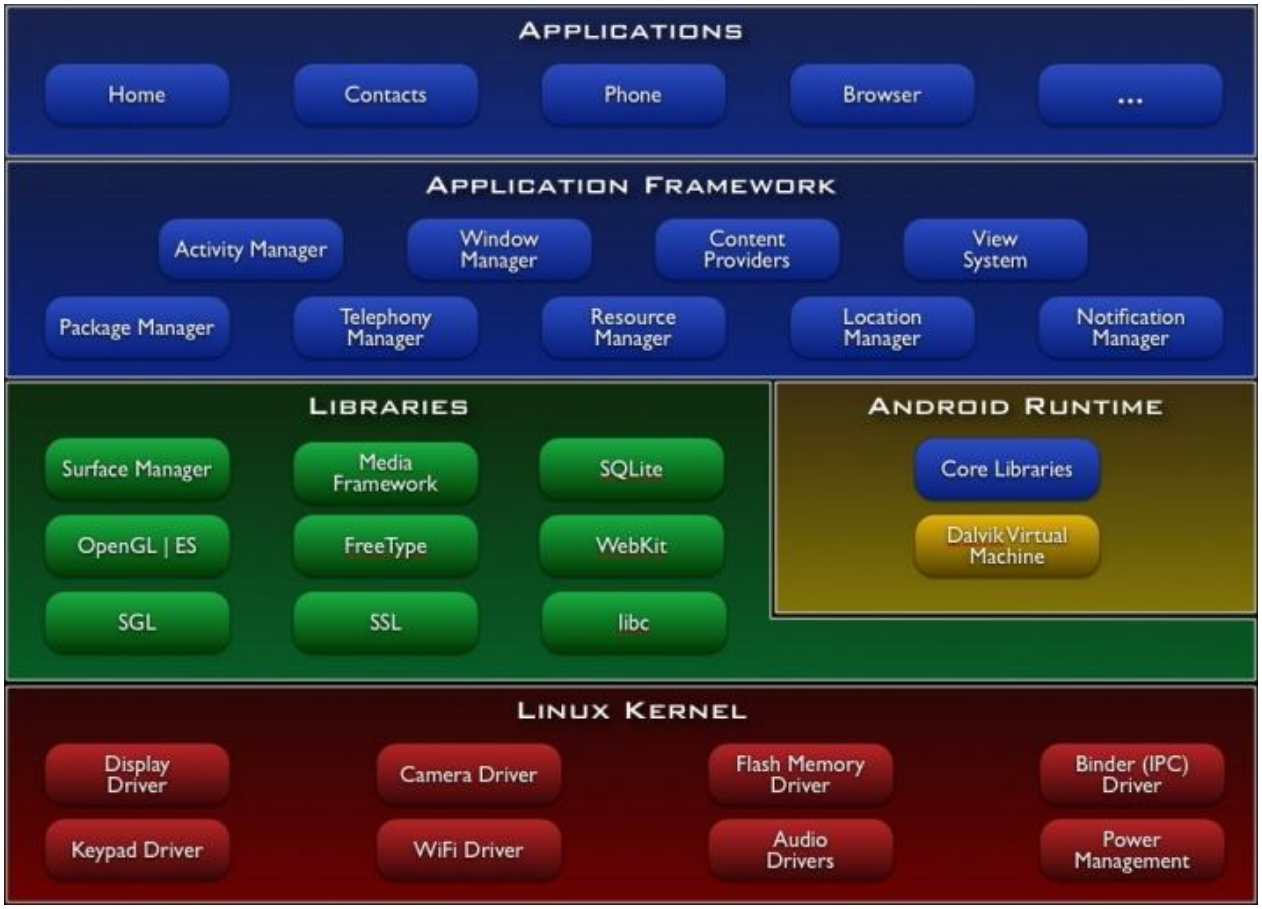
Kadangi šilumos pernešimo problemų sprendimų variacijos yra reikalingos tiek tarp mokslo visuomenės, tiek tarp verslo žmonių, kad būtų galima pasiekti aukštų sprendimų kokybės, buvo nuspręsta pasirinkti tokio tipo uždavinius, taip pat tai bus galimybė ištirti autoriui naują mokslo sritį.

Reikėtų paminėti, kad daugelis šiai OS sukurtų programų yra primityvaus funkcionalumo, vaizdus daugiausiai vaizduojančius dvimatėje erdvėje, su trimatės erdvės imitacija, sprendžiamos nesudėtingos matematinės problemos. Susidaro vaizdas, kad kol kas išmanieji telefonai, naudojami tik patenkinti eilinių vartotojų norus, su primityviomis programomis, kurios panaudoja tik nedidelę dalį įrenginių resursų.

1. Teorinė dalis

1.1. „Android“ operacinės sistemos architektūra

Kadangi darbo tematikoje „Android“ operacinė sistema (OS) yra esminė, reikia išanalizuoti šios platformos sandarą bei atsakyti į kai kuriuos esminius klausimus, pvz. kodėl pasirinkta „Dalvik“ virtuali mašina, o ne „Java“, gal tai turės įtakos tolimesnei darbo eigai. Taip pat turi būti aišku, ar pati platforma yra kuom tai ypatinga – galinti taip pat įtakoti darbo eigą. Žemiau (1 pav.) yra pateikiama OS architektūra.



1 pav. Android OS architektūra [23]

Iš pateiktos architektūros (1 pav.) matome, kad ši OS naudoja „Linux“ branduolį. Reikėtų paminėti, kad jis yra „Linux“ branduolio modifikacija. Kai kurie branduolio pakitimai ir patobulinimai pateikiami žemiau [25]:

1. Realizuotas „Android“ specifinis bendravimo mechanizmas tarp procesų – „Binder“. Šiuo papildiniu vieni procesai gali kviesti funkcijas kituose procesuose perduodant parametrus.

2. Įgyvendintas bendros atminties valdymo mechanizmas – „Ashmen“, kuris palaiko mažo kiekio atminties turinčius įrenginius, kadangi jis gali lengvai atlaisvinti bendros atminties nustatytą kiekį virtualioje atmintyje.
3. Atlikta procesų atminties skirstytuvo realizaciją – „Pmem“, kuri iš dalies yra panaši į antrame punkte nurodytą funkcionalumą, tačiau skirtumas tas, kad „Pmem“ manipuliuoja realia fizine atmintimi ir atitinkamai procesai negali dalintis tarp savęs tuo pačiu atminties bloku.

Toliau aukštesniame lygyje (1 pav. virš „Linux“ branduolio) yra sisteminės bibliotekos (Libraries), sekantis į viršų eina aplikacijoms skirti servisai (Application Framework) ir toliau pačios aplikacijos. Dar liko nepaminėta viena esminė dalis – „Dalvik“ virtuali mašina (esanti „Android Runtime“ apvalkale). Ši virtuali mašina skiriasi nuo „Java“ virtualios mašinos keliais esminiais punktais, pateiktais žemiau [13]:

1. „Dalvik“ vykdomieji failai (DEX) užima mažiau (apie 50%) vietos negu „Java“ failai (JAR).
2. Ši VM (Dalvik) naudoja registrus (Register-Based) vietoj steko (Stack-Based), tai leidžia sumažinti apie 47% panaudojamų VM vykdomų instrukcijų, tačiau kodo apimtis padidėja apie 25%.

Atsižvelgiant į tai, kad „Dalvik“ vykdomieji failai naudoja mažiau įrenginių resursų negu „Java“ failai, galime teigti, kad „Google“ todėl ir pasirinko šią VM, taip siekdama taupyti mobiliųjų įrenginių panaudojamus resursus. Tačiau remiantis šališku „Oracle“ tyrimu, „Dalvik“ VM yra 2-3 kartus lėtesnė [14]. Taigi galima teigti, kad VM pasirinkimo lemiamasis faktorius buvo rinkos pasidalijimo klausimas.

1.2. Baigtinių elementų metodo (BEM) samprata

Šis skaitinis metodas yra ypatingas tuo, kad su juo sprendžiami tokio tipo uždaviniai, kurių negalima išspręsti analiziniu būdu. Tokie uždaviniai dažniausiai yra realaus gyvenimo uždaviniai, juose retai rasime pavyzdžiui kaip „kvadratinis“ medis, kurio neveikia jokie išoriniai ar vidiniai poveikiai.

Būtinybė sukurti baigtinių elementų metodą (*FEM – Finite Element Method*) kilo bandant apskaičiuoti inžinerinių konstrukcijų patvarumą ir atsparumą įvairioms fizikinėms modifikacijoms (formų ir kitų savybių kitimams).

BEM dažniau suvokiamas kaip universali skaičiavimų technologija fizikinių ir inžinerinių objektų ar sistemų elgsenos analizėms atlikti. Daugelį praktinių uždavinių šiandien galima išspręsti taikant rinkoje pateikiamomis BEM kompiuterinėmis programomis.

BEM pasižymi universalumu, leidžia įvertinti konstrukcijos ir aplinkos sąveiką (mechaniniai, temperatūros poveikiai, kraštinės sąlygos). Metodas turi paprastą fizinę interpretaciją ir tampriai susietas su statybinėje mechanikoje plačiai taikomu poslinkių metodu.
Metodo esmė: nagrinėjamas objektas (*domain*) yra padalijamas į daug mažų objektų (*subdomains*), vadinamų baigtiniais elementais. Vėliau šiems elementams yra pritaikomos algebrinės funkcijos ar kitos mažiau sudėtingos funkcijos, ir visų baigtinių elementų duomenys yra sudedami į globalią sistemą, taip suformuojama algebrinių lygčių sistema. Žinoma, skaičiavimo rezultato tikslumas priklauso nuo baigtinių elementų skaičiaus. Kuo baigtinių elementų skaičius didesnis (padalytų objektų skaičiui artėjant į begalybę), tuo rezultatas bus tikslesnis. [2]

1.3. Šilumos pernešimo uždavinio formulavimas

Žemiau pateikiama šilumos pernešimo uždavinio formuluotė. Taip pat algoritmas šiam uždaviniui išspręsti bei šio darbo dalis tame algoritme.

Šilumos kiekio judėjimas į ar iš kūno gali būti apskaičiuojamas tuomet, kai temperatūros sklidimas yra žinomas. Temperatūros kitimas turi įtakos įtempimų pasiskirstymui. Šiluminis įtempimas galimas kiekviename kūne. Šios reikšmės yra svarbios apsvarstymui projekto keitimosi priemonėms, tokioms kaip variklis ir garų generatorius. Pirminis žingsnis, norint įvertinti šilumos įtaką, yra apibrėžti temperatūros plitimą į kūną. [1]

Nestacionarus šilumos sklidimas trimatėje konstrukcijoje iš anizotropinės medžiagos aprašomas diferencialine lygtimi

$$k_x\frac{\partial^2 T}{\partial x^2}+k_y\frac{\partial^2 T}{\partial y^2}+k_z\frac{\partial^2 T}{\partial z^2}+Q=\rho c\frac{\partial T}{\partial t} \quad (1)$$

Čia k_x,k_y,k_z – medžiagos šiluminio laidumo koeficientai; ρ – medžiagos tankis c – specifinė šiluma; $Q(x,y,z)$ – vidinis šilumos srautas; $T(x,y,z)$ – temperatūra; t – laikas.

(1) lygčiai išspręsti įvedamos pradinės ir kraštinės sąlygos:

$$T(x,y,z,t=0)=T_0 \quad (2)$$

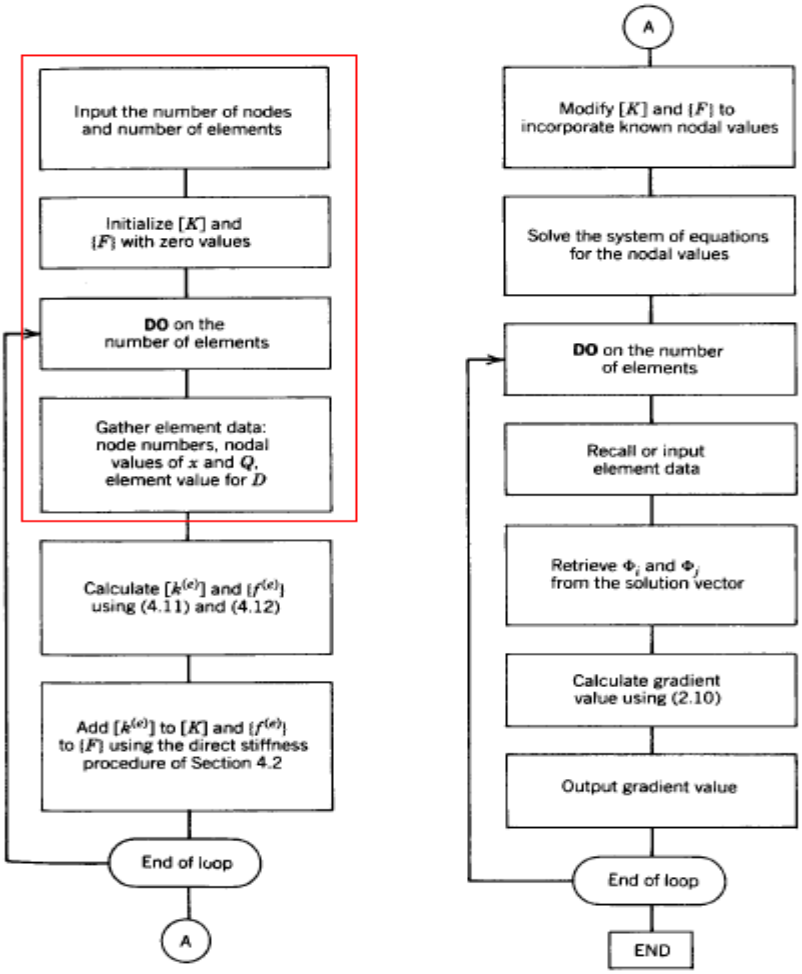
$$k_x\frac{\partial T}{\partial x}n_x+k_y\frac{\partial T}{\partial y}n_y+k_z\frac{\partial T}{\partial z}n_z+q+\alpha(T-T_\infty)=0 \quad (3)$$

Čia n_x,n_y,n_z – paviršiaus išorinės normalės krypties kosinusai, $q(x,y,z)$ – šilumos srautas, α – šilumos atidavimo koeficientas, T_∞ – žinoma išorinės aplinkos temperatūra.

(1) lygtis su pradinėmis sąlygomis (2) ir kraštinėmis sąlygomis (3) sprendžiant uždavinį baigtinių elementų metodu, užrašoma:

$$[C]\frac{\partial\{T\}}{\partial t}+[K]\{T\}=\{F\} \quad (4)$$

Čia $[K]$, $[C]$ ir $\{F\}$ yra atitinkamai šiluminio laidumo, šiluminės talpos matricos ir šiluminės apkrovos vektorius.



2 pav. Šilumos pernešimo uždavinio sprendimo BEM schema [2]

Bendra šilumos pernešimo uždavinio sprendimo BEM schema pateikta 2 pav. iš kurios matosi, kad šis darbas užima tik nedidelę dalį (raudonai apvestą) algoritmo panaudojime, kadangi dėl nenumatytų nukrypimų siekiant tikslo atsirado laiko stygius.

2. Projektinė dalis

2.1. „Android“ operacinės sistemos pasirinkimas

Renkantis tikslo operacinę sistemą, buvo pasirinktos kelios sąlygos: 1) kad darbo rezultatai turėtų kuo didesnę naudą jai; 2) kad informacijos kiekis apie ją būtų kuo didesnis ir išsamesnis; 3) nereikėtų patirti papildomų kaštų atliekant tyrimą susijusį su ja, vadinasi operacinė sistema turėtų būti atviro kodo arba turėtų būti leista ja naudotis nemokamai ant turimų kompiuterinių resursų.

Renkantis operacinę sistemą buvo padaryta prielaida: daugiausiai rinkos užimanti operacinė sistema turėtų tenkinti pirmą ir antrą sąlygas, kadangi didesnė auditorija didina tikimybę, kad bus daugiau kvalifikuotų operacinės sistemos naudotojų, kuriems šio darbo rezultatai gali būti naudingi ir atitinkamai jų pačių atliktų darbų informacijos panaudojimas šiam darbui.

Žemiau pateikiama statistinė pasaulinė išmaniųjų telefonų pardavimų lentelė (1 lent.) pagal OS gamintojus (2012 m. 4-as ketvirtis):

1 lent. Išmaniųjų telefonų, pagal OS, pardavimų statistika [3]

Operacinė sistema	Pardavimų skaičius (tukst.)	Rinkos dalis (%)
Android	144,720.3	69.7
iOS	43,457.4	20.9
Research In Motion	7,333.0	3.5
Microsoft	6,185.5	3.0
Bada	2,684.0	1.3
Symbian	2,569.1	1.2
Others	713.1	0.3
Viso	**207,662.4**	**100.0**

Iš 1-oje lentelėje pateiktų duomenų, galime teigti, kad apie du trečdalius rinkos užimanti operacinė sistema yra „Android“. Remiantis šio skyriaus pradžioje išsikeltomis sąlygomis ir prielaida, buvo pasirinkta būtent ši operacinė sistema.

2.2. „Android“ programų kūrimo įrankių pasirinkimas

Programų kūrimo įrankių, tikslo operacinei sistemai, pasirinkimas nėra didelis, vienas jų – „Android“ SDK (programinių priemonių paketas), visi kiti – universalios priemonės, kurių pagalba parašytas kodas tinka ne vienai operacinei sistemai. Pirmiausiai paanalizuosime universalius programinius įrankius, kurių keletas yra pateikiama 2-oje lentelėje, apibendrinanti, kurioms operacinėms sistemoms programavimo aplinka yra tinkama.

2 lent. Programavimo aplinkų mobiliesiems įrenginiams OS palaikymas [24]

Operacinė sistema/Programavimo aplinka	**Appcelerator Titanium**	**PhoneGap**	**Rhodes**	**AIR for Mobile Devices**	**OpenPlug Elips Studio**
Android	Taip	Taip	Taip	Taip	Taip
iOS (iPhone, iPod, iPad)	Taip	Taip	Taip	Ne	Taip
BlackBerry	Ne	Taip	Taip	Ne	Ne
Symbian	Ne	Taip	Ne	Ne	Taip
Palm	Ne	Taip	Ne	Ne	Ne
Windows Phone 7	Ne	Ne	Ne	Ne	Ne
Viso:	2	5	3	1	3

Iš 2-os lentelės pateiktų duomenų matome, kad visos nurodytos programavimo aplinkos palaiko „Android“ operacinę sistemą (OS). Tačiau visos jos vis tiek reikalauja specifinių OS programinių paketų įdiegimo (pvz. Android SDK), taip pat, kai kurios funkcijos yra specifinės OS, tad tenka atlikti kodo taisymus [24]. 2-oje lentelėje paminėtos programavimo aplinkos yra tarsi kitas sluoksnis konkrečių platformų programiniams paketams (SDK), dėl šios priežasties buvo pasirinktas „grynas“ tikslo operacinės sistemos SDK. Be to, galime teigti, kad iš esmės tėra tik vienas įrankių paketas, programų kūrimui, pasirinktai operacinei sistemai – „Android“ SDK.

Kadangi „Android“ SDK atkeliauja su suderintu „Eclipse“ IDE įskiepiu, tai privedė prie šios aplinkos pasirinkimo. Taip pat, pasirinktas SDK įspraudžia į programavimo kalbos rėmus – programos yra rašomos „Java“ programavimo kalba.

2.3. Programinių įrankių (PĮ) skirtų šilumos pernešimo uždaviniams spręsti analizė

Šiame skyrelyje apžvelgsime programinius įrankius naudojamus šilumos pernešimo uždaviniams spręsti (ir ne tik) BEM. Apžvalgos esmė, išsiaiškinti kokie PĮ egzistuoja pasaulinėje rinkoje, kokioms operacinėms sistemoms jie yra pritaikomi bei kokioms programavimo kalboms yra skirti. Žinoma, svarbiausias aspektas – kokį įrankį pasirinkti darbo tikslo įgyvendinimui, kadangi tiksle paminėtas uždavinys yra tik priemonė pasiekti tikslą, tad norėdami sutaupyti laiko,

buvo nuspręsta, kad geriausiai uždavinio sprendimui būtų panaudoti jau realizuota programinį įrankį, kuris turėtų būti atviro kodo, tenkinantis programavimo kalbos pasirinkimą (žr. 2.2 skyrių) bei lengvai integruojamas su pasirinkta programavimo aplinka (žr. 2.2 skyrių). Žemiau yra pateikiama PĮ analizės lentelė (3 lent.).

3 lent. Programinių įrankių, šilumos pernešimo uždaviniams spręsti, analizė

Programinio įrankio pavadinimas	**Paskirtis**	**OS**	**Programavimo kalba vartotojui**	**Ar komercinė?**	**Atnaujinimo data**
Code_Aster	Struktūrinė ir šilumos mechanika	Windows, Linux, Mac OS X	Fortran, Python, C/C++	Ne	2011-12-7
Impact	Struktūrinė mechanika	Daugiaplat formė	Java	Ne	2012-01-26
FEFLOW	Šilumos ir skysčių pernešimas	Windows, Linux	C/C++	Taip	2010-05-01
ANSYS	Skysčių mechanika, elektromagnetiniai ir kiti uždaviniai	Windows, Unix/Linux	Nerasta	Taip	Nerasta
STAAD.Pro	Struktūrinė inžinerija	Windows	Visual Basic, Java, C, C++, Fortran	Taip	2008-07-20
COMSOL (FEMLAB)	Fizikiniai reiškiniai, inžinerija, skysčių pernešimas	*Windows*, *Mac* OS X, *Linux*, *Unix*	C++, Java	Taip	2011-05-18
ALGOR	Bendrosios *paskirties fizikos baigtinių elementų analizė*	*Windows*, *Linux*	Nerasta	Taip	2010-11-19
Abaqus	Kompiuterinė inžinerija ir baigtinių elementų metodas	*Windows*, *Linux*, IBM AIX	C++	Taip	2011-05-01
Cosmos	Kieto kūno, skysčių, dujų, elektrostatinių ir elektromagnetinių laukų analizė baigtinių elementų metodu	Nerasta	C#	Taip	Nerasta
FuturEye	1D, 2D ir 3D dalinėms diferencialinėms lygčių (PDE) išvestinėms	Daugiaplat formė	Java	Ne	2013-03-05
Maple	Įvairūs matematiniai skaičiavimai ir vizualizavimas,	Windows, Linux, Macintosh, Solaris	Maple	Taip	2012-04-28

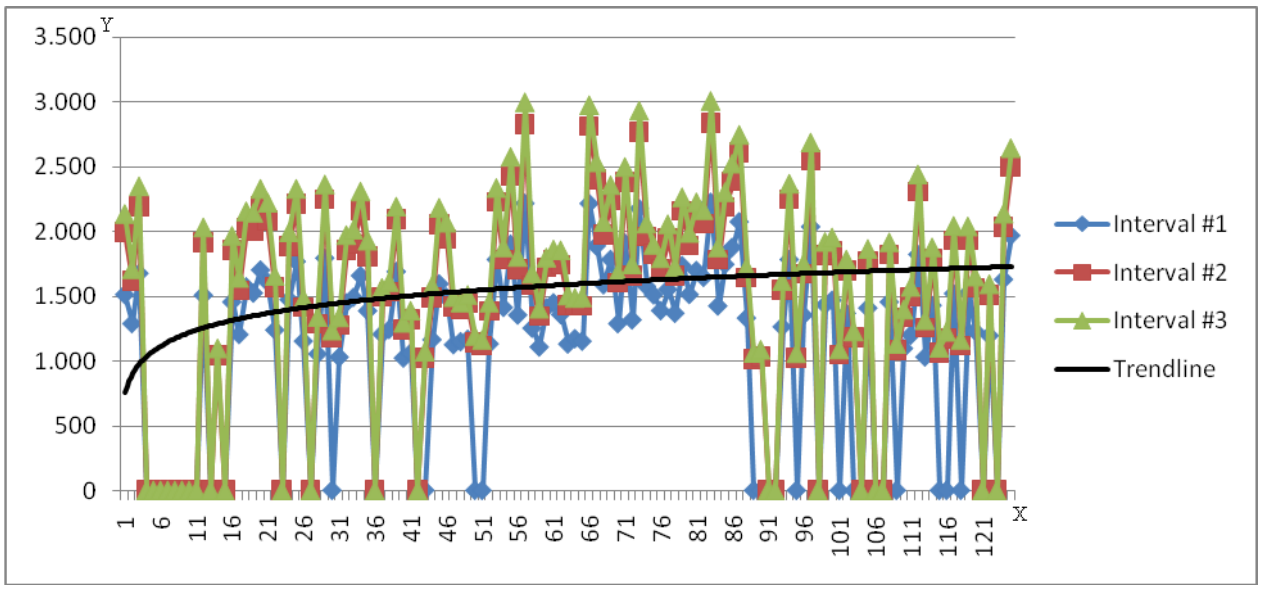
Iš 3 lent. apžvalgos matosi, kad dauguma programinių įrankių tinka ne vienai operacinei sistemai, dauguma jų yra dažnai atnaujinamos ir komercinės. Tačiau komerciškumas neturi įtakos kiek dažnai PĮ yra atnaujinami.

Iš atliktos analizės rezultatų, darbo tikslui buvo pasirinktas „FuturEye“ įrankis (biblioteka), kadangi jis tenkina šio skyriaus pradžioje įvardytus reikalavimus. Tikimąsi, kad ją bus lengviausiai integruoti į pasirinktą programavimo aplinką, nes patirtis rodo, kad įrankiai, su kuriais reikia bendrauti per operacinės sistemos kanalus nustatytu protokolu, jų konfigūracija trunka gerokai ilgiau negu tiesioginis bibliotekos naudojimasis. Be to, įrankis turi būti lengvai įdiegiamas į pačią „Android“ operacinę sistemą, kas su dauguma programinių paketų neįmanoma, nes jie kurti buvo kitoms operacinėms sistemoms.

2.4. „FuturEye“ bibliotekos detalizavimas

„FuturEye“ yra „Java“ programavimo kalba parašyta klasių biblioteka. Jos paskirtis yra spręsti nežinomų funkcijų (turinčių skaitinius ir/ar vektorinius parametrus) įvairių dimensijų (1D, 2D ir 3D) diferencialinių lygčių sistemas. Taip pat šiuo įrankiu galima spręsti atvirkštines problemas (Reverse Problems), tik kai kuriais atvejais reikia pirma išspręsti pirmines problemas (Forward Problems). [15]

Siekiant įsitikinti, ar ši biblioteka, kurios autorius yra dr. Yueming Liu (skaičiuojamoji matematika) – „Nankai“ universitetas, integruojasi kartu su pasirinkta programavimo aplinka (žr. 2.2 skyrių), buvo pabandyta paleisti testą, gautą kartu su šia biblioteka. Testas susideda iš dvimačių trikampių masyvo (Triangle Mesh) aprašo ir kodo, kurio paskirtis apskaičiuoti šilumos kitimą laiko intervalais (kas 0,2 sekundės), trejais bandymais. Buvo nustatyta nulinė Dirichle baigtinių elementų kraštinių sąlyga.



3 pav. Šilumos pernešimo uždavinio testinis bandymas su „FuturEye“ biblioteka

Ašyje X (3 pav.) yra laiko atkarpa iki 121 sekundės. Ašyje Y yra vaizduojamas temperatūros T kitimas laiko intervale. Kadangi šie rezultatai buvo apskaičiuoti integravus biblioteką į programavimo aplinką, vadinasi galime teigti, kad biblioteka buvo sėkmingai integruota.

Testavimo metu buvo iškilęs vienas nesklandumas – išorinių bibliotekų paieška, nuo kurių priklauso „FuturEye“ biblioteka. Tačiau buvo susisiekta su autoriumi ir jis tą pačią dieną atnaujino internetinėje svetainėje (http://code.google.com/p/futureye/) bibliotekos archyvą parsisiuntimui, įkeldamas reikalingas išorines bibliotekas.

2.5. Techninės įrangos parametrų parinkimas

Pirmiausiai reikia apibrėžti kompiuterinius resursus bei operacinę sistemą, kurie bus panaudojami pasirinktam programiniam įrankiui (Android SDK), tikslo operacinės sistemos programoms kurti. Kadangi turimi resursai tenkina pasirinktų įrankių minimalius resursų reikalavimus, nuspręsta pasirinkti šiuos parametrus:

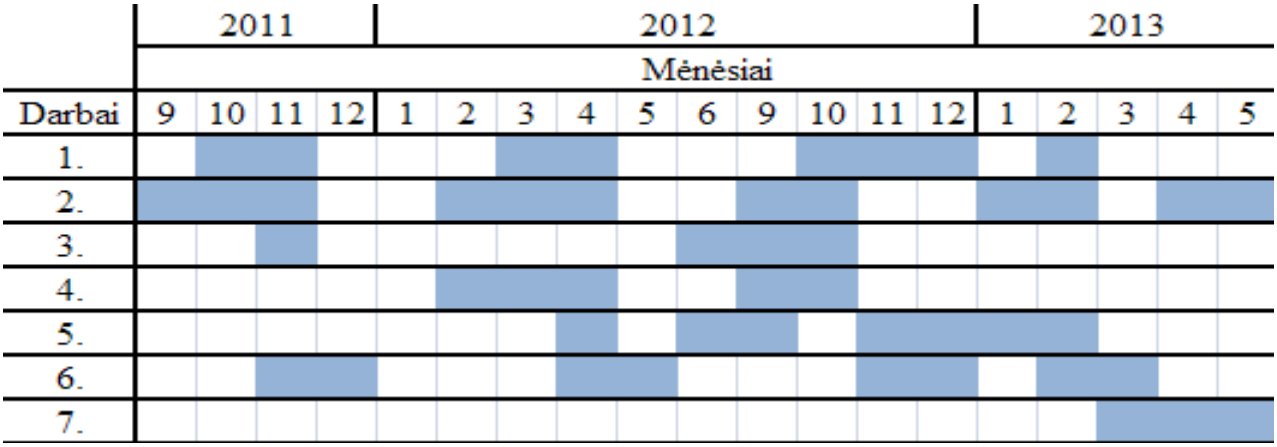
1. Operacinė sistema – „Microsoft Windows XP Pro., Service Pack 3, 5.1 Build 2600“.
2. Fizinės atminties dydis – 4 GB.
3. Procesorius – „AMD Athlon“, ~1,8 GHz.
4. Vaizdo korta – „NVIDIA GeForce 6200“, 512 MB fizinės atminties.

Įvertinant tai, kad kuriama programa bus emuliuojama „Android“ emuliatoriaus, kas suteikia galimybę programą paleisti ant įvairių virtualių mobiliųjų įrenginių ir savo ruoštu reiškia, kad iš RISC (Reduced Instruction Set Computer) architektūros vykdomosios instrukcijos bus verčiamos į CISC (Complex Instruction Set Computer), o tai leidžia daryti prielaidą, kad konvertavimo metu gali iškilti problemų, tad būtina tikslo programą paleisti ant realaus (fizinio) įrenginio, tad tam tikslui buvo pasirinktas „HTC Desire HD“ mobilusis įrenginys:

1. Fizinės atminties dydis – 768 MB.
2. Procesorius – „Scorpion“, 1 GHz.
3. Grafikos procesorius – „Adreno“ 205.

2.6. Darbų eigos plano projektas

Žemiau (4 pav.) yra pateikiamas pradinis darbų vykdymo planas, kuris eigoje gali keistis. Kiekvienas darbas turi savo numeri su žemiau šio paveiksliuko pateiktais jų aprašymais.



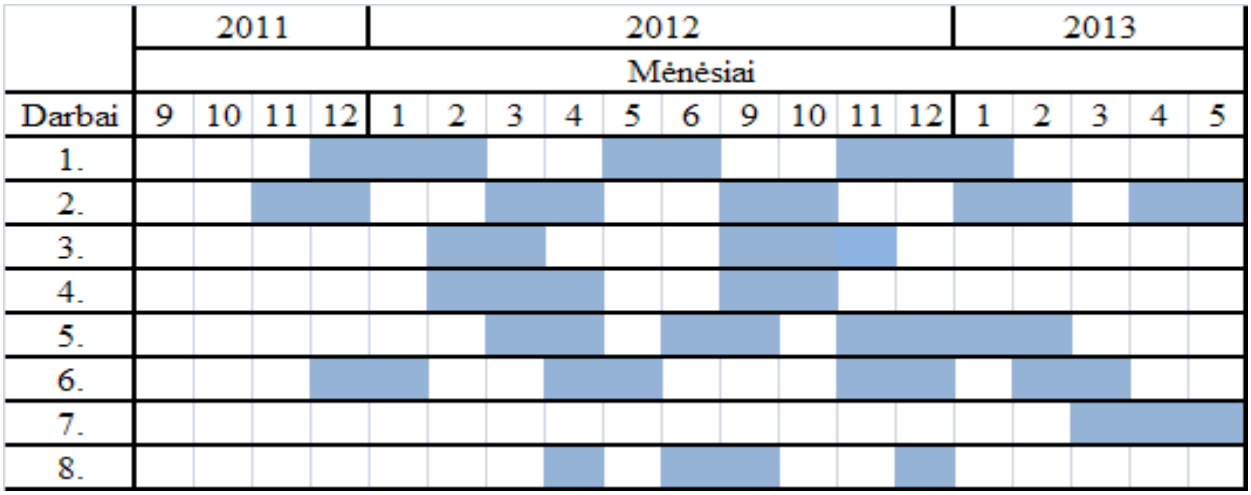
4 pav. Pradinis darbų vykdymo planas

1. Informacijos paieška ir analizė – įvairių elektroninių (pvz. elektroninės bibliotekos) bei stacionarių (pvz. bibliotekos) šaltinių paieška, jų atrinkimas bei panaudojimas.
2. Konsultacijos su darbo vadove – periodiškai vykdomos konsultacijos, kad darbas neišeitų už jam neskirtų ribų.
3. Darbo tikslo uždavinių formulavimas – tai bus galima daryti, tik jau turint reikalingą informaciją.
4. Tikslo programos projektavimas – bus siekiama suprojektuoti tikslo programos klasių sąryšius.
5. Tikslo programos kūrimas ir testavimas – lygiagrečiai bus realizuojamas programos funkcionalumas ir atliekami testavimai.
6. Darbo aprašymo sudarymas – šio darbo aprašymo rašymas.
7. Ruošimasis darbo pristatymui – skaidrių kūrimas bei pristatymo praktikavimasis.

3. Realizacinė dalis

3.1. Darbų eigos realus planas

Iš darbų eigos plano (5 pav.) matome, kad prie pradinio darbų vykdymo plano (žr. 2.6 skyrių) dar prisidėjo vienas punktas (8 punktas). Kiti pakitimai yra tik laiko persiskirstymas.

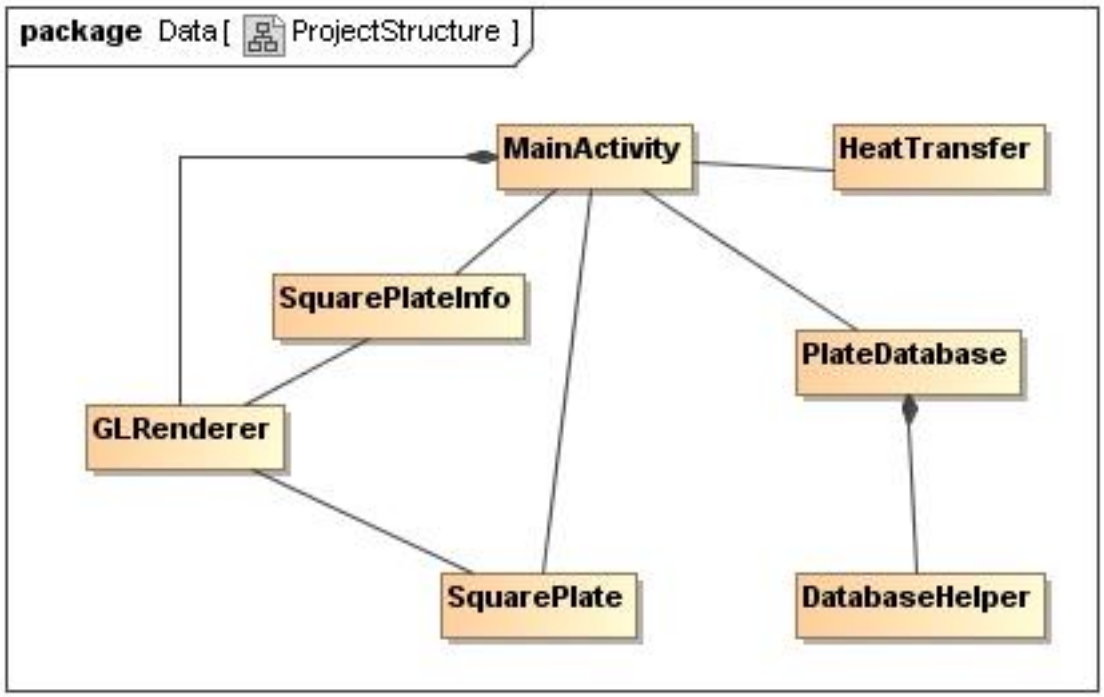


5 pav. Darbų eigos realus planas

1. Informacijos paieška ir analizė – įvairių elektroninių (pvz. elektroninės bibliotekos) bei stacionarių (pvz. bibliotekos) šaltinių paieška, jų atrinkimas bei panaudojimas.
2. Konsultacijos su darbo vadovę – periodiškai vykdomos konsultacijos, kad darbas neišeitų už jam neskirtų ribų.
3. Darbo tikslo uždavinių formulavimas – tai bus galima daryti, tik jau turint reikalingą informaciją.
4. Tikslo programos projektavimas – bus siekiama suprojektuoti tikslo programos klasių sąryšius.
5. Tikslo programos kūrimas ir testavimas – lygiagrečiai bus realizuojamas programos funkcionalumas ir atliekami testavimai.
6. Darbo aprašymo sudarymas – šio darbo aprašymo rašymas.
7. Ruošimasis darbo pristatymui – skaidrių kūrimas bei pristatymo praktikavimasis
8. Diskretizuotos plokštelės vaizdavimo realizacija – tai yra dalis 5 punkto, ji buvo išskirta, kadangi pareikalavo išskirtinai laiko su likusiomis programos dalimis.

3.2. Plokštelės diskretizavimo ir vizualizavimo įrankio schema

Žemiau yra pateikiamas sukurtos programos projektas (6 pav.), kuris buvo kuriamas siekiant išanalizuoti tikslo operacinės sistemos programų kūrimo įrankius. Klasių metodų aprašas nebuvo įkeltas, kadangi buvo norima parodyti tik pačią struktūrą neapkraunant vaizdo.



6 pav. Sukurto tikslo programinio įrankio klasių schema

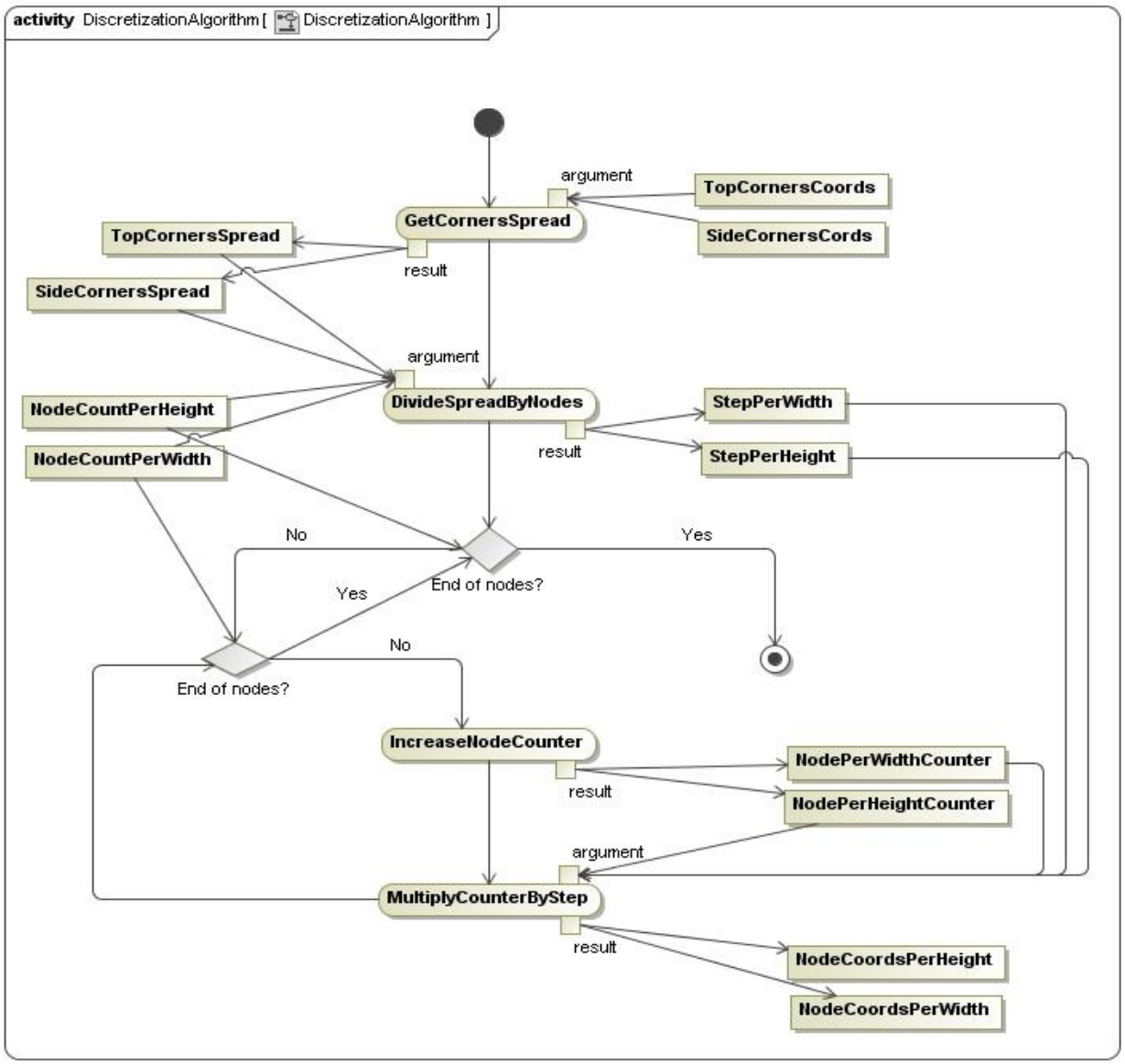
Programą sudaro 7 klasės, iš kurių labiausiai norėtųsi atkreipti dėmesį į keturias esmines klases, kurios atlieka visą diskretizavimo ir vaizdavimo darbą:

1. SquarePlate – ši klasė dalija aprašytą plokštelę į baigtinius elementus ir mazgus – iš esmės tai mazgų koordinačių generatorius, kurios yra susiejamos su baigtiniais elementais duomenų struktūrose. Plačiau apie plokštelės dalijimą žiūrėti 3.3 skyrių.
2. PlateDatabase – realizuoja pirmo punkto gautų rezultatų saugojimo lokalioje duomenų bazėje bei nuskaitymo iš jos manipuliavimo funkcionalumą. Plačiau apie duomenų saugojimą žiūrėti 3.5 skyrių.
3. HeatTransfer – ši klasė iš esmės yra „FuturEye“ bibliotekos bandomojo pavyzdžio (žr. 2.4 skyrių) abstrakcija, kuri buvo pritaikyta (trikampių masyvo aprašo failas būtų nuskaitomas iš programos vykdomojo failo dalies) prie sukurtos programos.

4. GLRenderer – šio komponento funkcionalumas yra pirmo punkto gautų rezultatų vaizdavimas virtualaus ar fizinio įrenginio ekrane. Plačiau apie duomenų vaizdavimą žiūrėti 3.4 skyrių.

3.3. Keturkampės plokštelės diskretizavimo algoritmas

7 paveikslėlyje vaizduojamas keturkampės plokštelės mazgų ir baigtinių elementų aprašo generavimas. Iš esmės, tai mazgų koordinačių sąsajos su baigtinių elementų numeriais. Šie duomenys bus reikalingi šilumos pernešimo uždaviniui spręsti. Ši dalis yra „SquarePlate“ (žr. 3.2 skyrių) klasės realizacijos projektas.



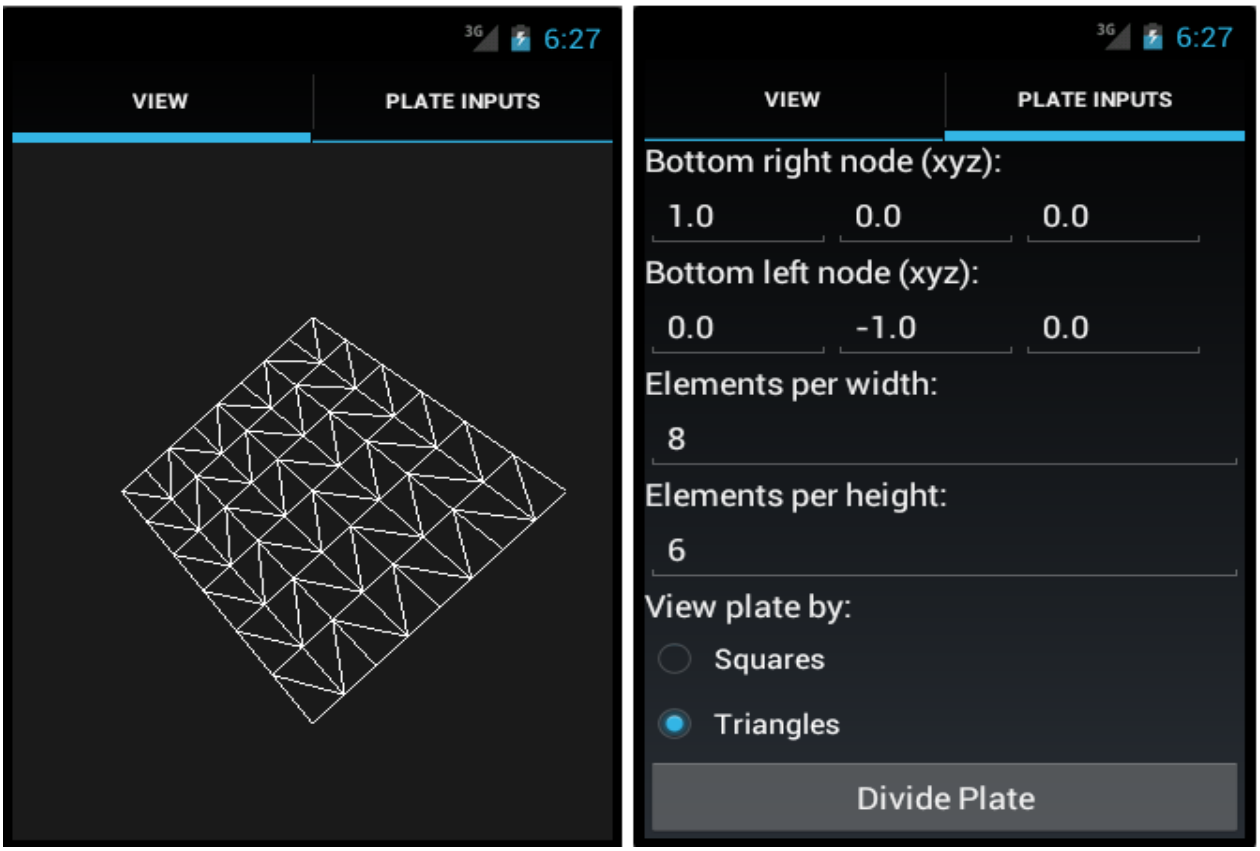
7 pav. Keturkampės plokštelės diskretizavimo algoritmas

Pradiniai duomenys yra plokštelės kampų koordinačių aprašymas trimatėje erdvėje bei elementų skaičius. Visi gaunami baigtiniai elementai yra vienodo dydžio.

3.4. Diskretizuotos plokštelės vaizdavimas „Android“ emuliatoriaus ekrane

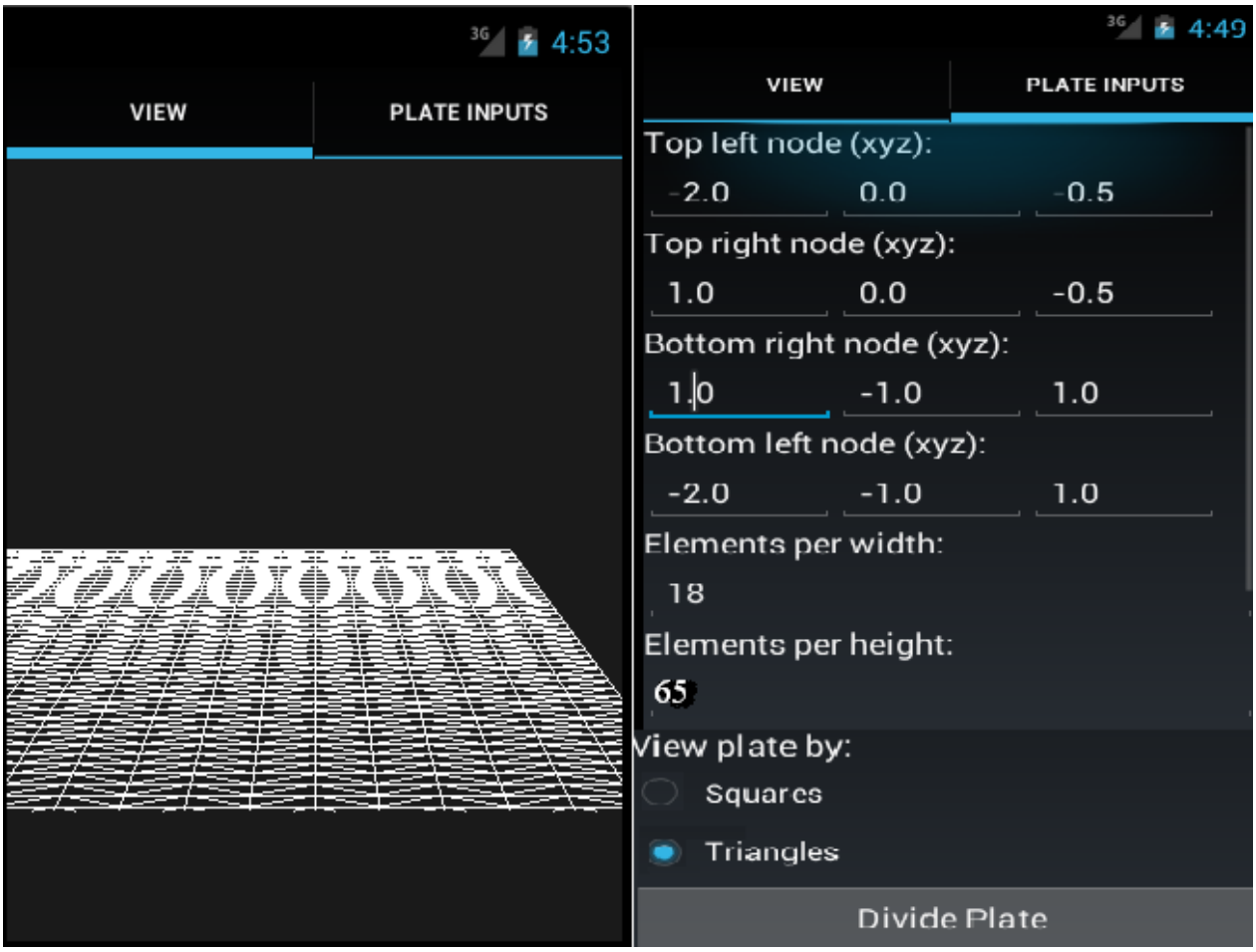
Siekiant išsiaiškinti, ar diskretizuotos plokštelės rezultatų vaizdavimas yra galimas ir naudingas (kaip informacijos pateikimas), buvo nuspręsta realizuoti vaizdavimą.

Žemiau pateiktuose paveikslėliuose (8 ir 9 pav.) galime matyti padalintos plokštelės į baigtinius elementus duomenų vaizdavimą trimatėje erdvėje, nors pati plokštelė yra dvimatė. Pradiniai duomenys yra plokštelės kampų koordinatės (x, y, z), elementų kiekis pagal plotį ir aukštį ir elementų tipas (trikampis ar stačiakampis). Baigtinius elementus (kurie yra vienodo dydžio) aprašo trys arba keturi mazgai. 8 pav. baigtiniai elementai (trikampiai) turi po tris mazgus, patys mazgai yra linijų sankirtos, o dalijimo žingsnis yra elemento kraštinės ilgis (žingsnis gali skirtis atitinkamai per aukštį ir plotį). Šiame skyriuje pateikiami vaizdiniai testavimo rezultatai (8 ir 9 pav.) yra „GLRenderer“ (žr. 3.2 skyrių) klasės realizacijos gautas vaizdas.



8 pav. Diskretizuotos plokštelės vaizdavimas virtualiame mobiliajame įrenginyje

Paveiksluose (8 ir 9 pav.) yra vaizduojami darbo programos testavimo rezultatų du vaizdai, pirmas (View), tai kuriame yra atvaizduojami diskretizuotos plokštelės duomenys, antras (Plate Inputs), tai pradinių plokštelės duomenų įvestis. Vienu metu galima matyti tik vieną iš pasirinktų vaizdų.



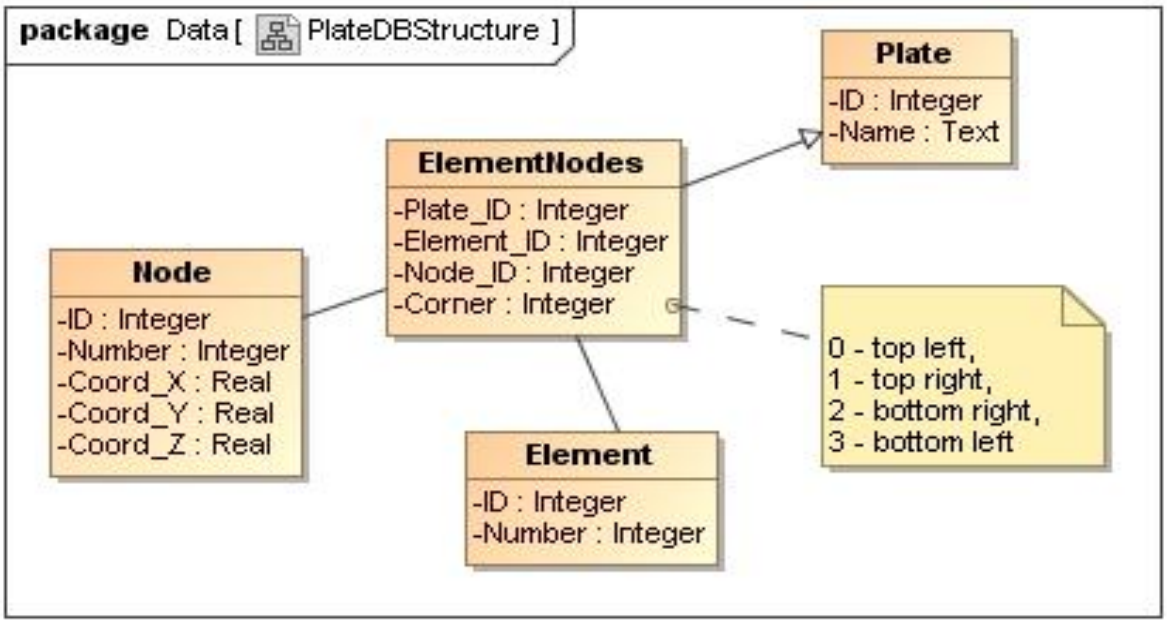
9 pav. Diskretizuotos plokštelės vaizdavimas virtualiame mobiliajame įrenginyje

9 pav. matome, kad vaizdas (kairėje pusėje) yra sunkiai įžiūrimas, kadangi pasirinktas baigtinių elementų skaičius yra didelis (1170, (18 x 65)).

Atsižvelgiant į gautus rezultatus, galime spręsti, kad diskretizuotos plokštelės duomenų vaizdavimas yra galimas, tačiau naudingas, kai plokštelė turi nedaug elementų.

3.5. Plokštelės diskretizavimo rezultatų saugojimas

Gautus plokštelės padalijimo į baigtinius elementus rezultatus reikia išsaugoti duomenų struktūroje, kad juos vėliau būtų galima panaudoti ar pratęsti skaičiavimus nuo konkretaus laiko momento. Duomenų saugykla buvo pasirinkta duomenų bazė, kadangi duomenų saugojimas failuose būtų pareikalavęs naujo duomenų analizatoriaus (Parser) arba papildomų bibliotekų, kai jau yra funkcionalumas darbui su duomenų bazėmis. Žemiau (10 pav.) yra pateikiamas duomenų bazės projektas (struktūra).



10 pav. Plokštelės mazgų ryšių duomenų bazės struktūra

Duomenų struktūrą (10 pav.) sudaro 4 lentelės, kuriose galima aprašyti neribotą (atsižvelgiant į fizinės atminties dydžius) kiekį diskretizuotų plokštelių duomenų. Detalus struktūros aprašymas:

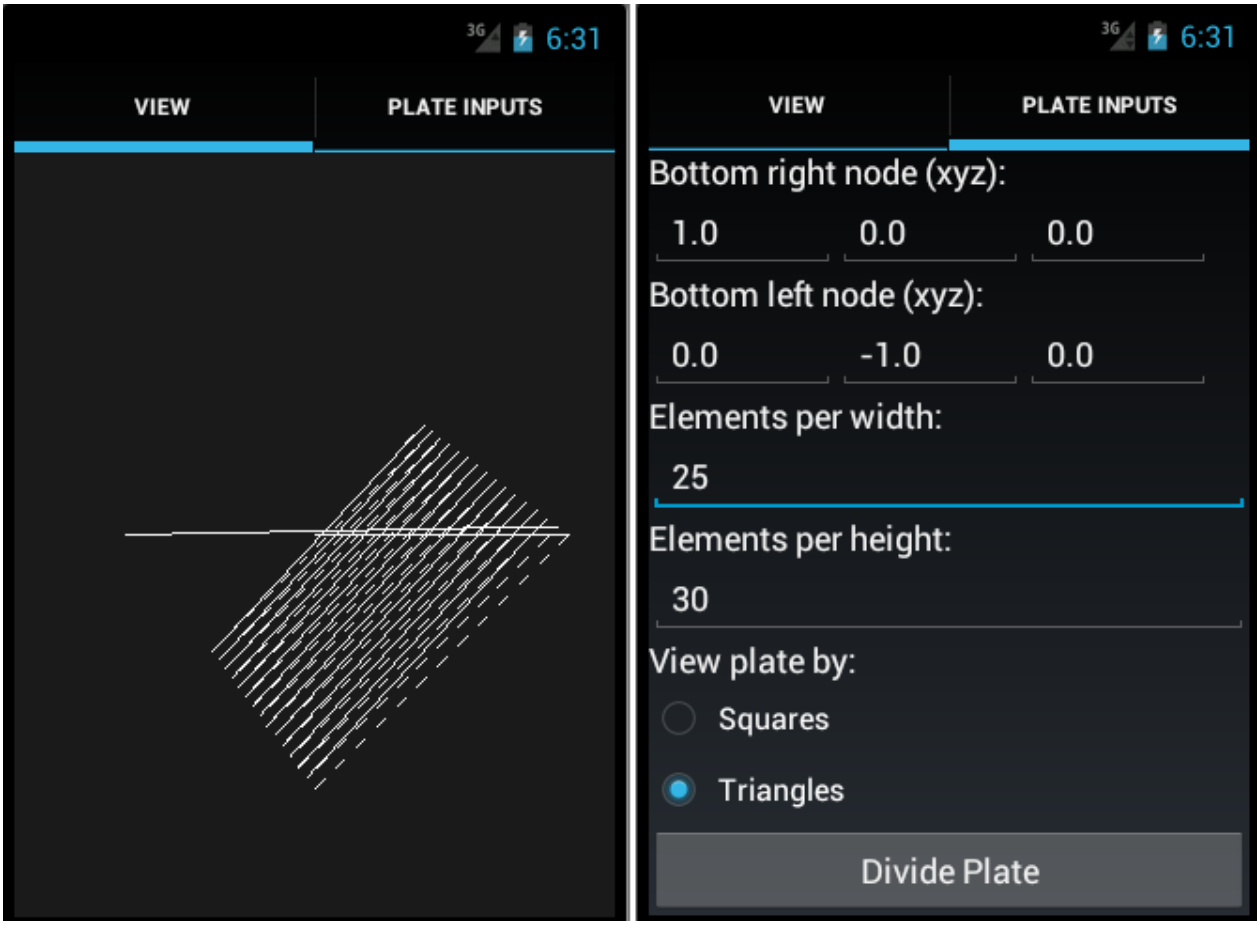
1. Plate – šioje lentelėje yra saugoma informacija apie plokšteles, unikalus duomenų bazės (DB) plokštelės identifikatorius (ID) bei jos pavadinimas (Name).
2. Element – ši struktūra labai panaši į pirmame punkte nurodytą struktūrą savo sandara, kurią sudaro unikalus DB elemento identifikatorius bei elemento numeris pačioje plokštelės struktūroje.
3. Node – elementų mazgų informacijos struktūra, kurioje saugoma unikalus DB mazgo identifikatorius, jo numeris pačioje plokštelės struktūroje bei trimatės mazgo koordinatės (Coord_X, ...).

4. ElementNodes – baigtinių elementų ir mazgų jungiamoji lentelė, kurioje yra aprašomas baigtinių elementų išsidėstymas erdvėje. Baigtinis elementas gali turėti daug sąsajų su mazgais. Taip pat šioje lentelėje yra aprašoma mazgo padėtis elemente (Corner), šio darbo tikslo įgyvendinimui pakanka keturių mazgo padėčių (viršutinis kairysis ir dešinysis, apatinis dešinysis ir kairysis kampai).

3.6. Problemos ir jų sprendimo būdai

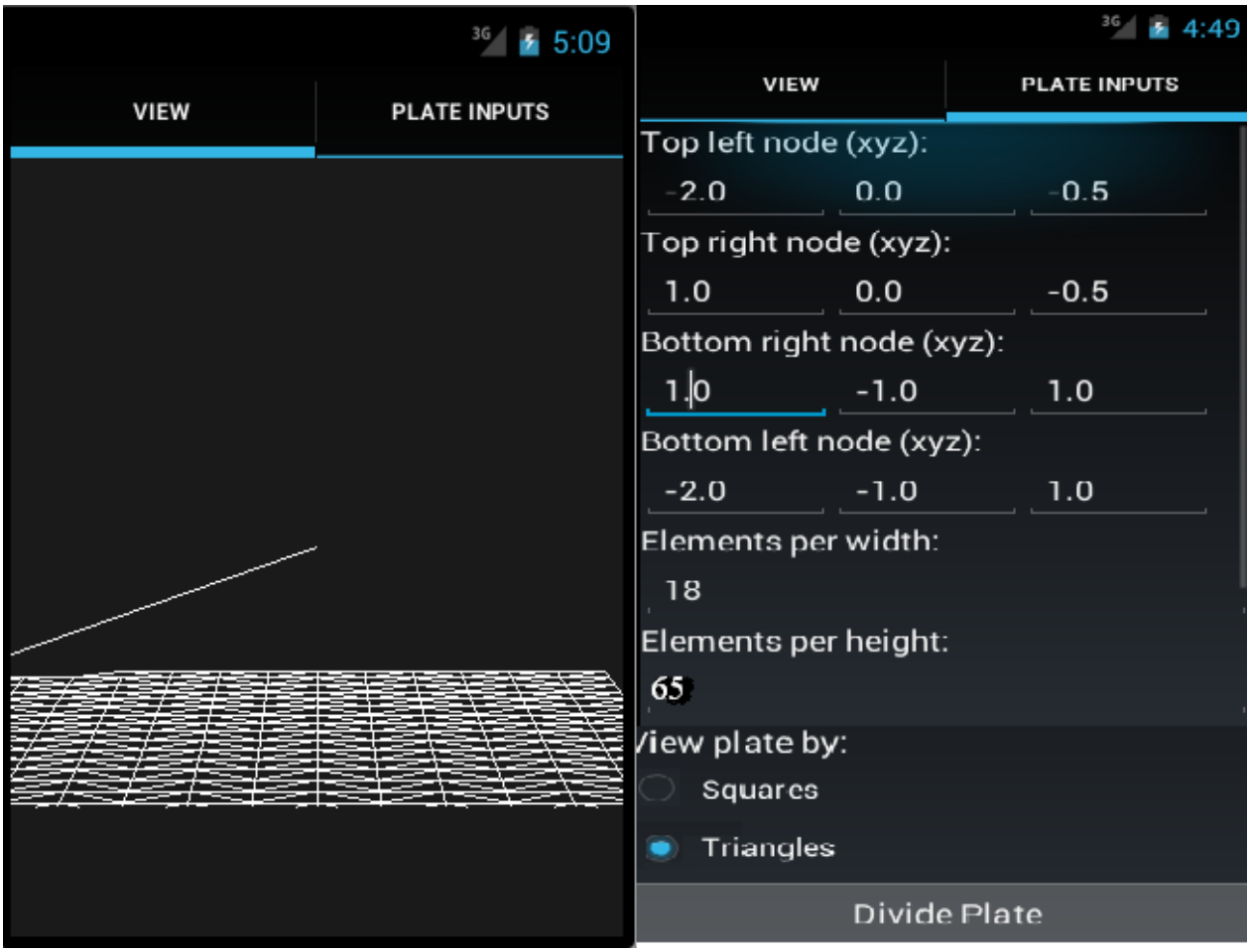
3.6.1. Diskretizuotos plokštelės vaizdavimo problema

Vizualizuojant padalintą į baigtinius elementus plokštelę iškilo vaizdo iškraipymo problema, kuri matosi 11 ir 12 pav. (kairinėse dalyse). Remiantis 2.5 skyriuje pateikta architektūrų vykdomųjų instrukcijų konvertavimo informacija, galime daryti prielaidą, kad būtent dėl šios priežasties įvyksta vaizdo iškraipymas. Taip pat reikėtų paminėti, kad duomenys pateikti vaizdavimui nesikeičia. Be to, vaizdo iškraipymo momentai yra atsitiktiniai.



11 pav. Padalintos plokštelės nepavykęs vaizdas

Atlikus programos testavimą mobiliajame įrenginyje (įrenginio aprašymas 2.5 skyriuje), vaizdo iškraipymo problema dingo. Tad manoma, kad šio skyriaus pradžioje aprašyta prielaida gali būti reali.



12 pav. Padalintos plokštelės nepavykęs vaizdas

Iškraipytas vaizdas (12 pav.) yra 9 pav. kopija, pradinių duomenų požiūriu. Pastebėta, kad daugelyje vaizdo iškraipymų, būna atsikišę keletas linijų iš visos plokštelės. Iš gautų rezultatų, matome, kad būtina gautus vaizdinius rezultatus patikrinti realiame įrenginyje, kadangi „Android“ emuliatorius dėl vaizdo iškraipymo problemos nėra patikimas.

3.6.2. Vaizdo papildomos informacijos funkcionalumo problema

Norint, palietus ekraną, gauti (ar atvaizduoti) informaciją apie mazgą (koordinates bei elemento numerį kuriame jis yra) ar baigtinį elementą (mazgų sudarančių šį elementą koordinatės

bei elemento numeris), reikia konvertuoti lango koordinates į „OpenGL ES“ pasaulio objekto koordinates. Tačiau problema yra ta, kad tokios konvertavimo funkcijos nėra realizuota „OpenGL ES“ 2.0 įrankyje, kuris yra naudojamas piešimui.

Buvo pabandyta apeiti šią situaciją ir šeideriams (OpenGL ES Shaders) apdorojant viršūnių piešimą, išsaugoti trečią koordinatę į tos viršūnės spalvos duomenų struktūrą, tačiau bandymas nedavė vaisių, o ir pats objektas nuo to keitė spalvas.

3.6.3. Lėto „Android“ emuliatoriaus veikimo problema

Emuliatorius užsikrauna apytiksliai po 4 min., o veikimo metu atsako/reakcijos laikas apie 3 s. Labai galima priežastis yra ta, kad jis konvertuoja architektūrų instrukcijas, kaip kad paminėta 2.5 skyriuje.

Norint pagreitinti programų kūrimą, buvo pabandyta pasinaudoti galima alternatyva emuliatoriui – „Android-x86“ projektu (versija 4.0 RC2), kuris yra „Android SDK“ emuliatoriaus atkėlimas į x86 architektūrą. Šis projektas buvo užkrautas su „Oracle VM VirtualBox“ 4.2.4 versija. Tačiau šis projektas atkrito, kadangi nebuvo rasta jam parašytos grafikos tvarkyklės. [4]

3.6.4 „Android“ emuliatoriaus fizinės atminties nustatymo problema

Nustačius fizinės atminties dydį neįeinantį į 512-768 ribas, „Android“ emuliatorius neužsikrauna. Atlikus atminties nustatymo keitimo bandymus, paleidus emuliatorių virtualioje mašinoje keliose operacinėse sistemose (Windows Vista 32-bit, Windows 7 32-bit), problema išliko.

Išvados

Remiantis šio darbo rezultatais bei surinkta informacija iš įvairių šaltinių, galime daryti tokias išvadas:

1. Atsižvelgiant į mobiliųjų įrenginių operacinių sistemų rinkos užimamas vietas pagal pardavimų skaičių (1 lent.), remiantis 2012 m. 4-o ketvirčio duomenimis, galime teigti, kad „Android“ operacinė sistema (OS) yra daugiausiai naudojama pasaulinėje rinkoje.
2. Remiantis atlikta programinių įrankių, šilumos pernešimo uždaviniams spręsti baigtinių elementų metodu, analize (žr. 2.3 skyrių), seka, kad atviro kodo tokio tipo programinių įrankių, parašytų „Java“ programavimo kalba yra nedaug – rasta tik viena biblioteka – „FuturEye“.
3. Iš „FuturEye“ bibliotekos integravimo į tikslo programą rezultatų (žr. 2.4 skyrių), galime teigti, kad dalis bibliotekos realizacijos, susijusios su šilumos pernešimo funkcionalumu, yra veikianti.
4. Iš 1.1 skyriuje pateiktos informacijos seka, kad „Dalvik“ virtuali mašina yra taupesnė atminties prasme ir mažiau reikalaujanti instrukcijų kodo vykdymui negu „Java“ virtuali mašina. Tačiau pastaroji yra 2-3 kartus greitesnė už pirmąją.
5. Tikslo programos realizacijos metu susidurta su vaizdo iškraipymo problema (žr. 3.6.1 skyrių) bei su keliomis kitomis problemomis (žr. 3.6.3, 3.6.4 skyrius), kas veda prie to, kad „Android“ SDK nėra iki galo išdirbtas „Android“ programų kūrimui.
6. „OpenGL“ pritaikymas mobiliesiems įrenginiams, pašalinant atitinkamą kiekį funkcijų, turėjo neigiamų pasekmių diskretizuotos plokštelės vaizdo manipuliacijoms (žr. 3.6.2 skyrių).
7. Remiantis 2.2 skyriuje pateikta informacija, galime teigti, kad įvairios programavimo aplinkos, skirtos „Android“ operacinei sistemai, negali veikti be specifinės šios operacinės sistemos programavimo įrankių rinkinio (Android SDK).

Literatūra

1. Roland W. Lewis, Perumal Nithiarasu, Kankanhally N. Seetharamu. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. England, 2004, 327 p.
2. Larry J. Segerlind. *Applied Finite Element Second Edition.* USA, Canada, 1984, 411 p.
3. *Gartner*. Prieiga per internetą: <http://www.gartner.com/newsroom/id/2335616> [Žiūrėta 2013 m. Gegužės 2 d.]
4. *„Android-x86“ projekto dokumentacija*. Prieiga per internetą: <http://www.android-x86.org> [Žiūrėta 2013 m. Gegužės 2 d.]
5. J. N. Reddy. *An Introduction to the Finite Element Method Second Edition.* USA, 1993, 684 p.
6. R. Barauskas, R. Belevičius, R. Kačianauskas. *Baigtinių elementų metodo pagrindai*. Vilnius: VGTU leidykla „Technika“, 2004.
7. *Code aster*. Prieiga per internetą: <http://www.code-aster.org> [Žiūrėta 2013 m. Gegužės 2 d.]
8. *Impact.* Prieiga per internetą: <http://impact.sourceforge.net/index_us.html> [Žiūrėta 2013 m. Gegužės 2 d.]
9. *Feflow*. Prieiga per internetą: <http://www.feflow.com> [Žiūrėta 2013 m. Gegužės 2 d.]
10. V. Kildišas, T. Tekorius. *Procesų valdymo uždavinių sprendimas taikant MATLAB sistemą*. Kaunas, Technologija, 2000.
11. *Learn OpenGL ES.* Prieiga per internetą: <http://www.learnopengles.com> [Žiūrėta 2013 m. Gegužės 2 d.]
12. *Fundamental OpenGL tutorials.* Prieiga per internetą: <http://www.songho.ca/opengl> [Žiūrėta 2013 m. Gegužės 2 d.]
13. *The Dalvik Virtual Machine Architecture.* Prieiga per internetą: <http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf> [Žiūrėta 2013 m. Gegužės 02 d.]
14. *Java SE Embedded Performance Versus Android 2.2.* Prieiga per internetą: <https://blogs.oracle.com/javaseembedded/entry/how_does_android_22s_performance_stack_up_against_java_se_embedded> [Žiūrėta 2013 m. Gegužės 02 d.]
15. *FuturEye library documentation.* Prieiga per internetą: <http://code.google.com/p/futureye> [Žiūrėta 2013 m. Gegužės 02 d.]
16. *Abaqus overview*. Prieiga per internetą: <http://www.3ds.com/products/simulia/portfolio/abaqus/overview> [Žiūrėta 2013 m. Gegužės 2 d.]

17. *Maple.* Prieiga per internetą: <http://www.maplesoft.com/products/maple> [Žiūrėta 2013 m. Gegužės 12 d.]
18. *Comsol.* Prieiga per internetą: <http://www.comsol.com> [Žiūrėta 2013 m. Gegužės 2 d.]
19. *Ansys.* Prieiga per internetą: < http://www.ansys.com> [Žiūrėta 2013 m. Gegužės 2 d.]
20. *STAAD.Pro.* Prieiga per internetą: <http://www.bentley.com/en-US/Products/STAAD.Pro> [Žiūrėta 2013 m. Gegužės 2 d.]
21. *MoSync Documentation.* Prieiga per internetą: <http://www.mosync.com/documentation> [Žiūrėta 2013 m. Gegužės 2 d.]
22. *Corono SDK.* Prieiga per internetą: <http://www.coronalabs.com/products/corona-sdk> [Žiūrėta 2013 m. Gegužės 2 d.]
23. *Android Architecture – The Key Concepts of Android OS.* Prieiga per internetą: <http http://www.android-app-market.com/android-architecture.html> [Žiūrėta 2013 m. Gegužės 2 d.]
24. Timo Paananen. *Smartphone Cross-Platform Frameworks: A Case Study.* Suomija, 2011, 111 p.
25. *Android Kernel Features.* Prieiga per internetą: <http://elinux.org/Android_Kernel_Features> [Žiūrėta 2013 m. Gegužės 2 d.]
26. *OpenGL ES – The Standard for Embedded Accelerated 3D Graphics.* Prieiga per internetą: <http://www.khronos.org/opengles> [Žiūrėta 2013 m. Gegužės 2 d.]

Anotacija

Autorius: Saulius Levaginas

Tema: Android programų kūrimo įrankių galimybių spręsti šilumos pernešimo uždavinius analizė.

Šiaulių universitetas 2013

Esminis šio darbo siekis yra išanalizuoti „Android“ operacinės sistemos (OS) programų kūrimo įrankių galimybes. Remiantis darbo rezultatais, matome, kad ši OS užima daugiausiai rinkos pagal pardavimus. Taip pat yra įžvelgiama, kad pasirinktas programavimo įrankių komplektas (Android SDK) nėra iki galo išdirbtas programų kūrimui, tačiau nelabai yra iš ko rinktis, kadangi alternatyvios programavimo aplinkos reikalauja jos įdiegimo. Reikia pridurti, kad testavimui yra būtina naudoti realius fizinius įrenginius. Taip pat norėtųsi paminėti, kad šiai OS buvo rastas tik vienas programinis įrankis (FuturEye), skirtas šilumos pernešimo uždaviniams spręsti.

Summary

Author: Saulius Levaginas

Subject: Analysis of the Tools for Android Applications in Heat Transfer Problem.

Šiauliai university 2013

The main purpose of this work is to make an analysis of abilities of tools for development of Android operating system programs. In the view of the work results, we can conclude, that this OS has the biggest part of the market share by sales. Also we can see, that we are left with only one choice for development tools (Android SDK) of Android programs, because all other programming environments require that Android SDK must be installed. For a more accurate programming results, one should test applications on real devices. In addition, we should say, that only one programming tool (FuturEye) was found for heat transfer problems evaluation.

Priedai

Prie šio darbo yra pridedamas kompaktinis diskas, kurio turinį sudaro:

1. Kataloge „Aprasymas“ yra dvi elektroninės šio darbo aprašymo versijos (pdf ir doc formatu).
2. Kataloge „Projektas“ yra visi reikalingi kuriamos programos failai ir bibliotekos (reikalingi plokštelės padalijimui ir vizualizavimui).
3. Failas „JavaInstalll.exe“ yra skirtas įsirašyti aplinką leidžiančią naudotis ketvirtame punkte paminėta programavimo aplinka.
4. Kataloge „Android SDK“ yra programa, kurią įsidiegus yra leidžiama naudotis įrankiais padedančiais kurti programas „Android“ operacinei sistemai. Kartu su ja yra katalogas „eclipse“, kurio pagalba galima redaguoti antrame punkte paminėtus failus.
5. Faile „Disko_turinys.txt“ yra šis priedo aprašas.