HIBBO

Tietotekniikan sovellusprojekti

Matti Eskelinen
Olli Karppinen
Harri Kosunen
Riikka Rikkola

Sovellusraportti
Versio: 1.0
29.5.2003

Jyväskylän Yliopisto
Tietotekniikan laitos

Tekijät:

- Matti Eskelinen (me@amjayee.net)
- Olli Karppinen (ollkarp@cc.jyu.fi)
- Harri Kosunen (hmkosune@cc.jyu.fi)
- Riikka Rikkola (rerikkol@cc.jyu.fi)

Työ: Sovellusraportti tietotekniikan sovellusprojektiin

Työtila: Agora, huone AgC223.3, puhelinnumero 014-260 4965

Kotisivu: http://kotka.it.jyu.fi/hibbo/

Tiivistelmä

Tämä dokumetti on Jyväskylän yliopistossa keväällä 2003 toteutetun Hibbo-projektin sovellusraportti. Dokumentissa selostetaan, kuinka sovellus on toteutettu ja millaisia ratkaisuja on käytetty.

Avainsanat

Tietotekniikan Sovellusprojekti, fysiikan laitos, hila-Boltzmann, simulointi, graafinen käyttöliittymä, visualisointi, Kylix, OpenGL, Delphi

Dokumentin versiohistoria

Versio	Päivämäärä	Tekijät	Kuvaus
1.0 -1	13.5.2003	RR	Alustava versio
1.0 -2	20.5.2003	RR, ME	Lisäilty paljon toteutusosioon
1.0 -3	27.5.2003	ME	Lisäilty pari kaaviota ja algoritmikuvauksia sekä jatkokehitysideat
1.0	29.5.2003	ME	Korjailtu virheet ja puutteet

Tekijöiden lyhenteet

ME Matti Eskelinen

OK Olli Karppinen

HK Harri Kosunen

RR Riikka Rikkola

Sisältö

1 Johdanto

Hibbo-projekti suunnitteli ja toteutti Jyväskylän yliopiston fysiikan laitoksen hila-Boltzmann-simulaattoriin graafisen käyttöliittymän, jonka olennainen osa on laskentatulosten visualisointi. Projekti toteutettiin Jyväskylän yliopiston tietotekniikan laitoksen Sovellusprojektina. Projektiryhmän jäseniä olivat Matti Eskelinen, Olli Karppinen, Harri Kosunen ja Riikka Rikkola. Projektin vastaavana ohjaajana toimi Kari Kärkkäinen, ja Markus Inkeroinen harjoitteli vastaavan ohjaajan tehtäviä. Lisäksi teknisenä ohjaajana toimi Rainer Koreasalo ja avustajana Jonne Itkonen.

Tässä dokumentissa kuvataan projektin puitteissa toteutettu sovellus. Tavoitteena on, että mahdolliset jatkokehittäjät saavat selkeän kuvan sovelluksen toteutusratkaisuista ja pystyvät kehittämään uusia ominaisuuksia. Luvussa 2 käydään läpi sovellukseen läheisesti liittyvät termit. Sovellukselle asetettujen vaatimusten toteutuminen käydään läpi luvussa 3 ja luvussa 4 kuvataan sovelluksen käyttöliittymää. Luvussa 5 käydään läpi sovelluksen ohjelmointiratkaisuja ja ideoita mahdollista jatkokehitystä varten käsitellään luvussa 6. Yhteenveto on koottu lukuun 7 ja lähteet käyvät ilmi luvusta 8.

2 Termit ja käsitteet

Tässä luvussa käydään läpi sovellukseen läheisesti liittyviä termejä.

2.1 Tiedostot

.sample on tiedosto, joka sisältää laskentageometrian.

.dat on simulointiohjelman tuottama tulostiedosto.

.evol on simulointiohjelman tuottama aikakehitystiedosto.

.field on simulointiohjelman tuottama tiedosto, jossa kerrotaan hilapisteen tyyppi, nestepisteen nopeudet x-, y- ja z-suunnassa sekä paine.

2.2 Ohjelmointikielet ja -työkalut

CLX on Borlandin kehittämä käyttöliittymien ohjelmointiin tarkoitettu luokkakirjasto, joka pohjautuu Qt-kirjastoon. Delphi ja Kylix käyttävät CLX-kirjastoa, ja se toimii sekä Windowsissa että Linuxissa.

Delphi on Borlandin kehittämä Windows-käyttöjärjestelmissä toimiva IDE (Integrated Development Environment) eli ohjelmankehitysympäristö, jossa ohjelmointikielenä käytetään Object Pascal -kieltä.

Kylix on Delphin vastine Linux-ympäristöön.

Object Pascal on olio-ohjelmointilaajennus Pascal-ohjelmointikieleen.

OpenGL on Silicon Graphics Inc:in kehittämä vapaa grafiikkakirjasto, lyhenne sanoista Open Graphics Library. Tässä projektissa OpenGL-kirjastoa käytetään laskentatulosten graafiseen esittämiseen.

Qt on Trolltechin kehittämä useissa ympäristöissä toimiva kirjasto ikkunointiohjelmien kehittämistä varten. Borlandin CLX-kirjasto perustuu Qt:hen.

VCL on toinen Borlandin kehittämä luokkakirjasto, jossa on pääosin samat rajapinnat kuin CLX:ssä, mutta se perustuu Windows APIin ja toimii vain Windows-ympäristössä

2.3 Simulointiprosessi

Hila-Boltzmann on algoritmi, jolla voidaan simuloida nesteen virtausta.

Laskentageometrian luonti on sovelluksen ominaisuus, joka luo keinotekoisesti huokoisen aineen hilarakenteen.

Visualisointi on tulosten esittämistä graafisesti.

Parametrit ovat simulointiohjelmalle tai näytteenluontiohjelmalle annettavia ohjeita simuloinnin tai näytteenluonnin suorittamiseksi.

Tulokset ovat simuloinnin tuloksena saatavia tiedostoja, joita on neljä kappaletta: laskentageometrian sisältävä tiedosto, tulostiedosto, aikakehitystiedosto ja tiedosto, joka sisältää hilapisteen tyypin, nestepisteen nopeudet x-, y- ja z-suunnassa sekä paineen.

3 Vaatimusten toteutuminen

Tässä luvussa käydään läpi, kuinka vaatimusmäärittelyn [1] yhteydessä asetetut vaatimukset toteutuivat.

3.1 Sovelluksen päämäärä

Hibbo-sovelluksen yleinen päämäärä oli toimia visualisointityökaluna hila-Boltzmann-menetelmällä tuotetuille virtaussimulaatiotuloksille. Sovelluksesta tuli voida käynnistää simulaatiolaskennan suorittava ohjelma sekä simulointiin tarvittavan näytetiedoston luova ohjelma. Hibbo-sovelluksen tuli siis toimia täysin itsenäisenä ohjelmana hyödyntäen kahta Jyväskylän yliopiston fysiikan laitoksella toteutettua ohjelmaa. Sovellusta itseään ei näin ollen voi käyttää minkäänlaiseen laskentaan, ainoastaan edellä mainittujen ohjelmien laskentatulosten visualisointiin.

3.2 Käyttöliittymä

Koska Hibbo-sovellusta tullaan käyttämään sekä opetuksessa että tutkimustyössä, vaatimuksena oli, että eritasoiset käyttäjät otetaan huomioon. Käyttöliittymän tuli olla helppokäyttöinen sekä havainnollinen.

Käyttöliittymän suurin rooli on visualisointinäkymällä, jonka avulla käyttäjä voi havainnollistaa luotua näytettä tai simuloinnin tuloksia. Erilaisten visualisointien tutkiminen on pyritty tekemään helpoksi ja havainnolliseksi siten, että käyttäjä voi pyörittää ja zoomata kuvaa hiiren avulla. Lisäksi näytteenluonti- ja simulointiohjelmalle annettavat parametrit sekä ohjelmien käynnistys hoidetaan käyttöliittymässä olevien välilehtien kautta. Tällä tavoin käyttäjä näkee myös esimerkiksi simuloinnin aikana, mitkä parametrit ohjelmalle on syötetty. Myös simuloinnin etenemistä on pyritty havainnollistamaan. Käyttäjä voi seurata simuloinnin etenemistä käyttöliittymälle piirrettävän permeabiliteettikäyrän avulla. Lisäksi käyttöliittymän tilapalkista näkyy, missä tilassa projekti on; tilapalkissa lukee, onko projekti tyhjä, näyte luotu tai onko simulointi käynnissä vai suoritettu. Käyttöliittymän otsikkopalkista näkyy myös auki olevan projektin ja näytteen nimi.

Toteutettu käyttöliittymä käydään tarkemmin läpi luvussa 4.

3.3 Ohjelman käynnistäminen ja sulkeminen

Sovelluksessa tulee aina olla auki jokin projekti. Vaatimusten mukaisesti käyttäjä voi avata tai luoda uuden projektin. Jokaisen projektin tiedot tal-

lennetaan omaan kansioon `Projects`-hakemiston alle. Sovelluksen käynnistyksen yhteydessä käyttäjän pitää avata tai luoda uusi projekti. Jos edellisellä käyttökerralla sovellus on suljettu siten, että simulaatio on jäänyt käyntiin, avautuu simulaatio automaattisesti sovellukseen valmiina tai keskeneräisenä. Sovelluksesta voidaan siis poistua siten, että simulointi jätetään käyntiin.

3.4 Asetukset

Projektissa käytettäviä visualisointivärejä ja kuville annettavaa perusnimeä voi muuttaa. Tiedot projektikohtaisista asetuksista tallennetaan projektitiedostoon. Luvussa 5.7 on käyty tarkemmin läpi projektitiedoston sisältö.

3.5 Näyte

Hibbo-sovelluksesta voidaan asetettujen vaatimusten mukaisesti käynnistää erillinen näytteenluontiohjelma, joka luo laskentageometrian annettujen parametrien perusteella. Näytteenluontiohjelman käynnistäminen tapahtuu käyttöliittymän `Sample`-välilehden kautta. Välilehdelle syötetään pyydetyt parametrit ja näytteenluontiohjelma käynnistetään `Start`-painiketta painamalla. Syötetyille parametreille tehdään oikeellisuustarkistus ja ohjelman käynnistäminen on mahdollista vasta sitten, kun kaikki parametrit on syötetty minimi- ja maksimiarvoja noudattaen. Myös näytteen avaaminen on mahdollista.

Näytettä voi tutkia siitä muodostettavan kolmiulotteisen kuvan avulla. Näytettä voi tarkastella pyörittelemällä, siirtelemällä ja zoomaamalla sitä.

3.6 Simulointi

Asetettujen vaatimusten mukaisesti Hibbo-sovelluksesta voidaan käynnistää erillinen simulointiohjelma, joka laskee hila-Boltzmann-menetelmällä virtaussimulaatiotuloksen annetussa näytteessä. Simulointiohjelman käynnistäminen tapahtuu käyttöliittymän `Simulation`-välilehden kautta. Välilehdelle syötetään vaaditut parametrit ja simulointiohjelma käynnistetään painamalla välilehdellä olevaa `Start`-painiketta. Syötetyille parametreille tehdään oikeellisuustarkistus ja ohjelman käynnistäminen on mahdollista vasta sitten, kun parametrit on syötetty minimi- ja maksimiarvoja noudattaen. Simulaation etenemistä voi seurata käyttöliittymään piirretävän permeabiliteettikäyrän avulla.

Myös simulaation avaaminen on mahdollista. Simuloinnin voi myös jättää käyntiin, vaikka käyttöliittymä suljetaan. Tällöin seuraavan kerran käyttöliittymään palatessa käyntiin jätetty simulaatio avautuu automaattisesti valmiina tai keskeneräisenä. Simulaation keskeyttäminen on myös mahdollista.

Simulointituloksia voi tutkia leikkeiden, partikkelien ja virtaviivojen avulla. Visualisoinnin toteuttamista on käyty tarkemmin läpi luvussa 5.

3.7 Tekniset vaatimukset

Asetettujen vaatimusten mukaisesti sovellus toimii sekä Linux- että Windows-ympäristössä. Sovelluksen toteuttaminen molempiin edellä mainittuihin käyttöjärjestelmiin onnistui käyttämällä Borlandin Delphi- ja Kylix-sovelluskehittimiä sekä CLX-luokkia, jotka toimivat molemmissa käyttöjärjestelmissä.

3.8 Toteuttamatonta

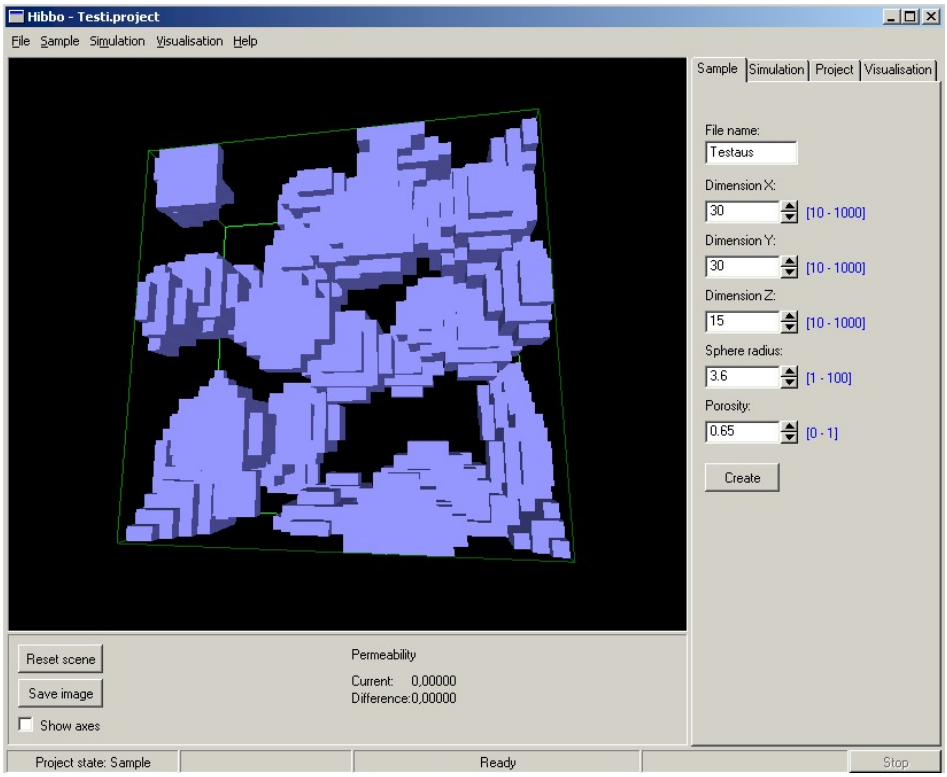
Simulointiohjelman käynnistäminen etäajona fysiikan laitoksen rinnakkaiskoneella jäi ominaisuudeksi, jota projektin puitteissa ei toteutettu. Ominaisuus rajattiin kuitenkin projektin ulkopuolelle jo vaatimusmäärittelyn [1] yhteydessä. Myös valinnaiseksi ominaisuudeksi asetettu videokuvan tallentaminen jäi ajan puutteen vuoksi pois toteutetuista ominaisuuksista. Nykyisessä ohjelman versiossa ei myöskään voi vielä leikata vain osaa näytteestä tarkasteltavaksi, mutta valmiudet tähän ovat olemassa. Kaiken kaikkiaan sovellus on toteutettu siten, että laajentaminen ja uusien ominaisuuksien kehittäminen on helppoa.

4 Käyttöliittymä

Tässä luvussa kuvataan Hibbo-sovelluksen käyttöliittymä. Käyttöliittymän ulkoasu ja rakenne käydään läpi, ja kaikki käyttöliittymässä olevat toiminnallisuudet selostetaan.

4.1 Ulkoasu

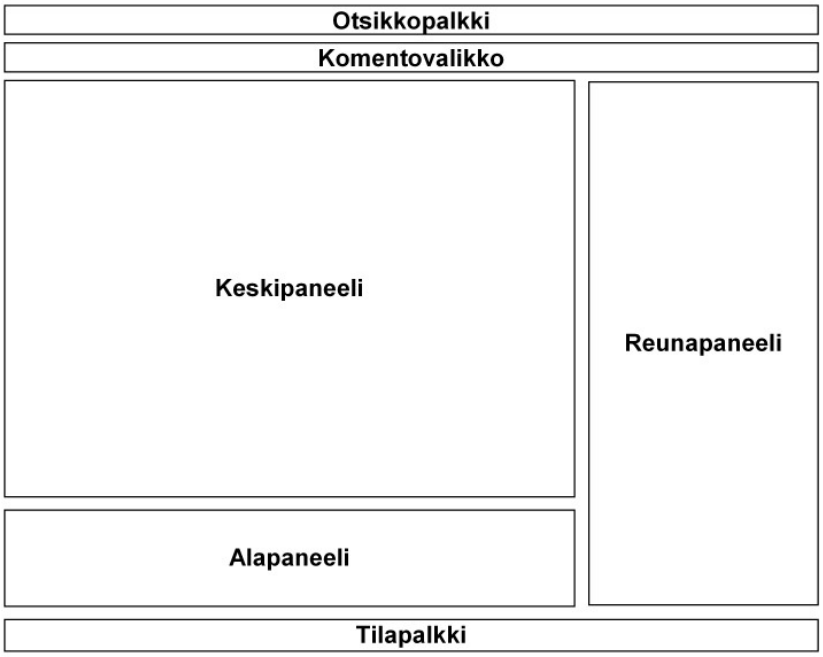
Hibbo-sovelluksen olennaisin osa on simulointitulosten visualisointi. Tämä näkyy myös käyttöliittymän ulkoasussa, jossa visualisointinäkymällä on suurin rooli. Käyttöliittymässä on myös oma alueensa näytteenluonti- ja simulointiohjelman tarvitsemien parametrien syöttöä varten. Käyttäjä ei siis anna parametreja ilmestyvien dialogien avulla, vaan parametrit syötetään käyttöliittymän oikeassa reunassa oleville välilehdille, joista syötettyjä tietoja voi tarkastella myös jälkikäteen. Kuvassa 1 näkyy sovelluksen ulkoasu, kun projektissa on luotu näyte.



Kuva 1: Sovelluksen ulkoasu.

4.2 Rakenne

Käyttöliittymän rakenne on jaettu paneelien avulla pienempiin kokonaisuuksiin. Jokaisella paneelilla on oma toiminnallinen tarkoituksensa. Kuvassa 2 näkyy käyttöliittymän rakenne.



Kuva 2: Käyttöliittymän rakenne.

Kuvasta 2 käy ilmi, että käyttöliittymä koostuu seuraavista osista: otsikkopalkki, komentovalikko, keskipaneeli, reunapaneeli, alapaneeli ja tilapalkki. Seuraavissa alaluvuissa on käsitelty yksityiskohtaisesti jokaisen osan sisältämät toiminnallisuudet.

4.3 Otsikkopalkki

Käyttöliittymän otsikkopalkki käsittää normaalit ikkunan sulkemis-, suurennus- ja pienennystoiminnot. Lisäksi otsikkopalkissa lukee sovelluksen nimi, ”Hibbo”, sekä käytössä olevan projektin nimi.

4.4 Komentovalikko

Käyttäjä voi suorittaa toimintoja käyttöliittymän yläosassa olevan komentovalikon kautta. Valikkorakenne hahmottuu seuraavan listan avulla.

File

- ☐ New Project
- ☐ Open Project
- ☐ Save Project
- ☐ Close Project
- ☐ Exit

Sample

- ☐ New sample file
- ☐ Open sample file

Simulation

- ☐ New simulation
- ☐ Open simulation
- ☐ Stop simulation

Visualisation

- ☐ Sections
- ☐ Particles
- ☐ Flow lines
- ☐ State
  - ☐ Zoom
  - ☐ Rotate

Help

- ☐ Help
- ☐ About

4.4.1 File-valikko

`File`-valikon suoritettavissa olevat toiminnot kohdistuvat projekteihin. Projekti voidaan sulkea, avata, tallentaa tai luoda. Lisäksi `File`-valikosta voidaan lopettaa sovelluksen toiminta.

New Project -toiminnon avulla käyttäjä voi luoda uuden tai valita vanhan projektin. Käyttäjälle avautuu dialogi, johon hän kirjoittaa uudelle projektille haluamansa nimen. Uusi projekti luodaan painamalla nimen kirjoittamisen jälkeen dialogilla olevaa `OK`-painiketta. Luodun projektin nimi vastaa dialogille syötettyä nimeä. Tämän niminen kansio luodaan projektihakemistoon, ja siihen tallennetaan kaikki projektiin liittyvät tiedostot. Kun uusi projekti on luotu, käyttäjä voi suorittaa haluamiaan toimintoja, joista tiedot tallentuvat projektitiedostoon. Käyttäjä voi myös peruuttaa projektin luomisen dialogin `Cancel`-painiketta painamalla. Mikäli annetulla nimellä on jo olemassa projekti, kysytään käyttäjältä haluaako hän korvata tämän.

Open Project -toiminnolla käyttäjä voi avata tallennetun projektin. Edellytyksenä on, että avattava projektitiedosto on olemassa. Toiminnossa käyttäjälle aukeaa dialogi, jonka avulla hän voi valita avattavan projektitiedoston. Kun tiedosto on valittu, suoritetaan projektin avaaminen painamalla dialogilla olevaa `Open`-painiketta. Projektin avaaminen voidaan myös peruuttaa painamalla dialogilla olevaa `Cancel`-painiketta.

Save Project -toiminto tallentaa avoinna olevan projektin. Projektille on annettu nimi, ja sille on luotu projektitiedosto projektinluomisen yhteydessä. Tämän toiminnon avulla saadaan siis projektiin tehdyt muutokset tallennettua sille luotuun kansioon.

Close Project -toiminnolla voidaan sulkea avoinna oleva projekti. Tällöin sovellus siirtyy tilaan, jossa yksikään projekti ei ole auki. Näytteen luominen, simulointi ja visualisointi sekä alapalkin toiminnot ovat käytettävissä vasta sitten, kun sovelluksessa on auki jokin projekti.

Exit -toiminnolla voidaan lopettaa sovelluksen toiminta. Käytössä oleva projekti tallennetaan automaattisesti lopetettaessa.

4.4.2 Sample-valikko

`Sample`-valikon avulla hallitaan näytetiedostoon kohdistuvia toimenpiteitä.

New Sample file -toiminnolla käyttäjälle avautuu käyttöliittymän oikeassa reunassa oleva `Sample`-lehti. Tältä välilehdeltä käyttäjä voi luoda uuden näytetiedoston syöttämällä tarvittavat parametrit ja käynnistämällä näytteenluontiohjelman. Näytetiedosto luodaan vasta, kun

kaikki parametrit on syötetty oikein minimi- ja maksimiarvoja noudattaen. `New Sample File` -valinta ei ole käytössä, jos projektissa on jo jokin simulointitulos. Tällöin uuden näytteen luominen on mahdollista vain luomalla uusi projekti.

Open Sample file -toiminnon avulla käyttäjä pääsee avautuvan dialogin avulla valikoimaan ja avaamaan valmiin näytetiedoston. Kun haluttu tiedosto on valittu, suoritetaan avaaminen painamalla dialogin `Open`-painiketta. Näytetiedoston avaaminen voidaan peruuttaa painamalla dialogin `Cancel`-painiketta. Avattu näyte visualisoidaan välittömästi. `Open Sample File` valinta ei ole käytössä, jos projektissa on jo jokin simulointitulos. Tällöin uuden näytteen luominen on mahdollista vain luomalla uusi projekti.

4.4.3 Simulation-valikko

`Simulation`-valikon kautta hallitaan simuloinnin suorittamiseen liittyviä toimenpiteitä.

New Simulation -toiminto avaa käyttöliittymän oikeassa reunassa olevan `Simulation`-lehden. Tältä lehdeltä käyttäjä voi käynnistää uuden simulaation syöttämällä tarvittavat parametrit ja käynnistämällä simulointiohjelman. Simulointi aloitetaan vasta, kun kaikki parametrit on syötetty oikein.

Open Simulation -toiminnolla käyttäjä voi avautuvan dialogin avulla etsiä jo valmiin simulaation ja avata sen. Kun valmis simulaatio on avattu, sitä voi visualisoida välittömästi.

Stop Simulation -toiminto ei ole valittavissa, ellei simulaatio ole käynnissä. Simulaation ollessa käynnissä `Stop Simulation` -valinta keskeyttää simuloinnin suorituksen. Simulaatio voidaan keskeyttää myös tilapalkissa tai `Simulation`-välilehdellä olevien `Stop`-painikkeiden avulla.

4.4.4 Visualisation-valikko

Visualisation-valikon kautta päästään toteuttamaan sovelluksen visualisointitoimintoja.

Sections -toiminnolla saadaan aktiiviseksi käyttöliittymän oikeassa laidassa oleva `Sections`-lehti. Kyseinen välilehti sijaitsee hierarkisesti

`Visualisation`-lehden alla. `Sections`-lehden toimintojen kautta voidaan edelleen tutkia simuloinnin tuloksia muodostamalla kaksiulotteisia leikkeitä. `Sections`-toiminto on mahdollinen ainoastaan silloin, kun projektissa on olemassa simulointitulos.

Particles -toiminnolla saadaan aktiiviseksi käyttöliittymän oikeassa laidassa oleva `Particles`-lehti. Kyseinen välilehti sijaitsee hierarkisesti `Visualisation`-lehden alla. `Particles`-lehden toimintojen kautta voidaan tutkia simuloitua näytettä laskemalla liikkeelle nestepartikkeleita. `Particles`-toiminto on mahdollinen ainoastaan silloin, kun projektissa on olemassa simulointitulos.

Flow lines -toiminnolla saadaan aktiiviseksi käyttöliittymän oikeassa reunassa oleva `Flow lines` -lehti. Kyseinen välilehti sijaitsee hierarkisesti `Visualisation`-lehden alla. `Flow lines` -lehden toimintojen avulla päästään tutkimaan simuloitua näytettä piirtämällä virtaviivoja nestepartikkelien liikeradoista. `Flow lines`-toiminto on mahdollinen ainoastaan silloin, kun projektissa on olemassa simulointitulos.

State -toiminnon kautta voidaan valita visualisointinäkymän tutkimiseen liittyvä tila. `Rotate`-tilassa visualisointinäkymää voidaan pyörittää ja `Zoom`-tilassa näkymää voidaan tutkia lähentämällä ja loitontamalla sitä.

4.4.5 Help-valikko

`Help`-valikon kautta voidaan tutkia sovelluksen käyttöön liittyvää avustusta tai sovelluksen tietoja.

Help -toiminnon avulla päästään lukemaan sovelluksen käyttöohjeita.

About -toiminnon kautta saadaan lisätietoja sovelluksesta avautuvan dialogin avulla. Painamalla dialogilla olevaa `OK`-painiketta, dialogi sulkeutuu.

4.5 Keskipaneeli

Keskipaneelille toteutetaan sovelluksen visualisointitoimintojen näkymä. Se ottaa vastaan hiiri- ja näppäinkomentoja, joilla voidaan muuttaa näkymää. Visualisointinäkymää voidaan pyöritellä hiirtä liikuttelemalla. Zoomaus tapahtuu hiiren rullan tai näppäimistön z- ja x-näppäinten avulla. Visualisointitoimintojen toteutuksesta kerrotaan tarkemmin luvussa 5.

4.6 Reunapaneeli

Käyttöliittymän oikeassa reunassa sijaitsevat eri toimintojen hallintaan, toteutukseen sekä asetuksiin tarkoitetut välilehdet. Paneelilla on omat välilehtensä simuloinnille, projektin tiedoille, näytteen luonnille ja visualisoinnille.

Perustoimintamallina on, että välilehdet on jaoteltu oikeaan paneeliin hierarkisesti. Kerrallaan on auki neljä ylemmän tason lomaketta: `Sample`, `Project`, `Simulation` ja `Visualisation`. Näistä `Visualisation`-välilehti sisältää alilehdet leikkeille, partikkeleille ja virtaviivoille. Välilehtien kokoonpano vastaa pitkälti käyttöliittymän päävalikon rakennetta ja lehdet ovatkin käytettävissä valikossa tehtyjen valintojen kautta. Lehtiä pääsee kuitenkin käyttämään myös suoraan.

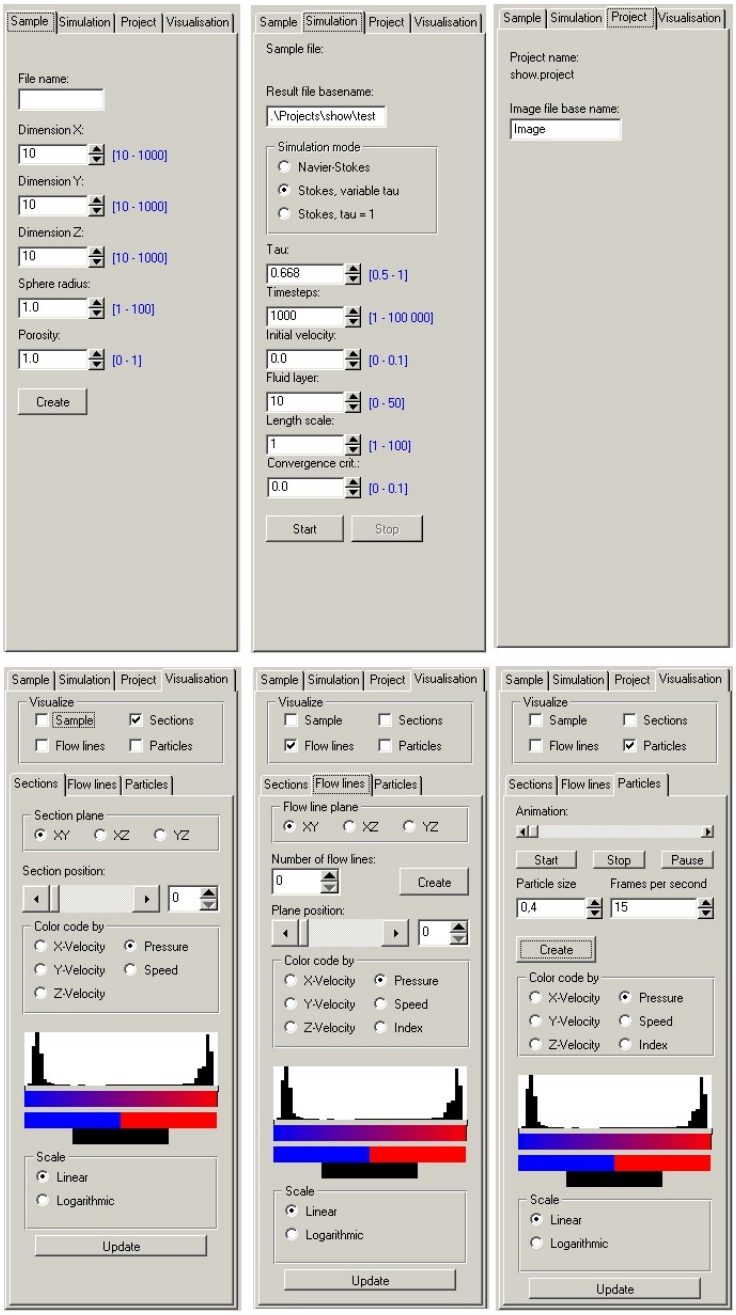
4.6.1 Sample

`Sample`-lehden sisältämillä toiminnoilla voidaan syöttää näytteen luomiseen tarvittavat parametrit ja käynnistää varsinainen näytteenluontiohjelma. Käyttäjän tulee syöttää tarvittavat parametrit minimi- ja maksimiarvoja noudattaen. Sallitut arvot näkyvät parametrien syöttökenttien oikealla puolella. Jos syöttökentästä poistutaan ja parametri on syötetty väärin, muuttuu kenttä väriltään punaiseksi. Mikäli kenttien arvot ovat vääriä vielä silloin, kun näytteenluontiohjelmaa yritetään käynnistää, ilmoitetaan virheellisistä numeerisista arvoista käyttäjälle dialogin avulla. Näytteenluontiohjelma on mahdollista käynnistää vasta sitten, kun kaikki parametrit on syötetty oikein. Käyttäjä voi syöttää numeerisen datan haluttuun kenttään rullaamalla hiirellä haluamaansa lukuarvoon tai kirjoittamalla arvon suoraan kyseiseen kenttään. `Sample`-välilehti ei ole käytettävissä, jos projektissa on olemassa jokin simulointitulos. Tällöin näytteen luominen edellyttää uuden projektin luomista. Kuvassa 3 näkyy käyttöliittymän `Sample`-välilehti.

File name -kenttään syötetään nimi, joka luotavalle näytetiedostolle halutaan. Tiedostonimen voi syöttää kenttään ilman `.sample`-tarkenninta, tai tarkentimen kanssa.

Dimension X, Y ja Z -kenttien avulla syötetään näytteen koko kolmessa ulottuvuudessa, x-, y- ja z-suunnassa.

Sphere radius -kenttään syötetään pallon säde. Näytteenluontiohjelma arpoo kuutiohilaan palloja, ja tämä parametri kertoo arvottavien pallojen säteen.



Kuva 3: Käyttöliittymän välilehdet.

Porosity -kenttään syötetään näytteen haluttu huokoisuus. Tällä parametrilla ilmaistaan, kuinka paljon valmiissa näytteessä pitää olla huokosia verrattuna kiinteään aineeseen.

Create -painiketta painamalla käynnistetään näytteenluontiohjelmisto, jolle viedään parametreina `Sample`-lehden kenttiin syötetyt tiedot. Mikäli painiketta painetaan, kun parametrit on syötetty kenttiin väärin, ilmoitetaan syöttövirheestä käyttäjälle. Näytteenluontiohjelmaa ei tällöin käynnistetä.

4.6.2 Simulation

`Simulation`-lehden sisältämillä toiminnoilla toteutetaan simulointiparametrien syöttäminen ja simuloinnin aloitus. Kaikki numeeriset arvot tulee syöttää kenttien oikealla puolella olevia minimi- ja maksimiarvoja noudattaen. Jos syöttökentästä poistutaan ja parametri on syötetty väärin, muuttuu kenttä väriltään punaiseksi. Mikäli kenttien arvot ovat vääriä vielä silloin, kun simulaatiota yritetään käynnistää, ilmoitetaan virheellisistä numeerisista arvoista käyttäjälle dialogin avulla. Simulaatio-ohjelma käynnistetään vasta sitten, kun parametrit on syötetty oikein. Käyttäjä voi syöttää numeerisen datan haluttuun kenttään rullaamalla hiirellä haluamaansa lukuarvoon tai kirjoittamalla arvon suoraan kyseiseen kenttään. Kuvassa 3 näkyy `Simulation`-lehden ulkoasu.

Sample file -kohdasta näkyy projektissa avoinna oleva näyte.

Result file basename -kenttään syötetään tulostiedostoille haluttu perusnimi. Tästä nimestä johdetaan tulostiedostojen nimet eri tarkentimin. Perusnimi syötetään kenttään ilman minkäänlaisia tiedostopäätteitä.

Simulation mode valitaan ruksaamalla yksi kolmesta valintamahdollisuudesta: `Navier Stokes`, `Stokes(variable tau)` tai `Stokes (tau=1)`.

Tau -kenttään syötettävä arvo määrää simulointinesteen viskositeetin. Mikäli `Simulation mode` -kohdassa valittiin jokin muu vaihtoehto, kuin `Stokes(tau=1)`, tulee relaksaatioparametri syöttää kenttään minimi- ja maksimiarvoja noudattaen. `Stokes(tau=1)` -moodilla tau on aina 1.0. Tällöin kenttään tulee luku 1 automaattisesti, eikä kenttään pysty kirjoittamaan.

Timesteps -kenttään syötetään tieto siitä, kuinka monta kierrosta hila-Boltzmann algoritmia halutaan suorittaa.

Initial velocity -kenttään syötetään tieto nesteen alkunopeudesta z-suunnassa.

Fluid layer -kenttään syötetään tieto siitä, kuinka monta laskentayksikköä nestettä on sekä näytteen ylä- että alapuolella.

Length scale -kenttään syötettävä parametri kertoo mallin koon suhteessa todelliseen maailmaan.

Convergence crit. -kenttään syötetään haluttu lopetusehto. Simulaatio voi siis päättyä jo ennen asetetun aika-askelmäärän täyttymistä, jos nesteen permeabiliteetti suhteellinen vaihtelu on pienempää kuin tähän kenttään syötetty arvo.

Start -painiketta painamalla käynnistetään simulointiohjelmisto, jolle viedään parametreina Simulation-lehden kenttiin syötetyt tiedot. Mikäli painiketta painetaan, kun parametrit on syötetty kenttiin väärin, ilmoitetaan syöttövirheestä käyttäjälle. Simulointiohjelmaa ei tällöin käynnistetä.

Stop -painiketta painamalla käynnissä oleva simulaatio voidaan keskeyttää.

4.6.3 Visualisation

`Visualisation`-lehden toimintojen avulla voidaan toteuttaa valmiin simulaation ja simuloimattoman näytteen visualisointeja. Välilehdeltä käyttäjä voi `Visualize`-osion avulla valita, mitä visualisointinäkymässä näkyy. Visualisointinäkymässä voi siis olla yhtäaikaa virtaviivoja, partikkeleja, leike ja näyte. Käyttäjä voi kuitenkin myös tarkastella edellä mainittuja visualisointimahdollisuuksia myös erikseen.

`Visualisation`-lehti jakaantuu kolmeen alilomakkeeseen: `Flow lines`, `Particles` ja `Sections`. Näiden välilehtien toimintojen avulla käyttäjä voi toteuttaa simulointituloksen visualisointeja haluamallaan tavalla.

Flow-lines -lehden toimintojen kautta käyttäjä voi tutkia simuloinnin tuloksia virtaviivojen avulla. Välilehden toiminnot ovat käytettävissä silloin, kun simulointi on suoritettu. Kuvassa 3 näkyy välilehden ulkoasu.

Plane -osion avulla valitaan taso, jolta alkaen virtaviivat piirretään.

Number of flow lines -kohtaan annetaan piirrettävien virtaviivojen lukumäärä.

Create -painikken avulla virtaviivat luodaan visualisointinäkymään.

Plane position -osiossa valitaan edellä mainitun tason kohta, jolta virtaviivat lasketaan liikkeelle.

Particles -lehden toimintojen kautta käyttäjä voi tutkia simuloinnin tuloksia nestepartikkelien avulla. Partikkelien lähtötaso voidaan määrittää `Flow lines` -välilehdeltä. Kuvassa 3 näkyy `Particles`-lehden ulkoasu.

Animation -liukusäätimen avulla käyttäjä voi itse vaikuttaa animaation kulkuun.

Start -painiketta painamalla käynnistetään animaatio.

Stop -painikkeesta animaatio pysäytetään kokonaan.

Pause -painiketta painamalla animaatio voidaan keskeyttää. Animaatiota voidaan jatkaa siitä kohdasta, mihin keskeytettäessä jäätiin, painamalla uudestaan `Pause`-painiketta.

Particle size -kenttään annetaan luotavien partikkelien koko hilakopeissa.

Frames per second -kenttään annetaan toivottu päivitysnopeus. Todellinen päivitysnopeus voi olla hitaampi, jos kone ei kykene päivittämään ruutua kyllin nopeasti.

Create -painiketta painamalla partikkelit luodaan visualisointinäkymään.

Sections -lehden toimintojen avulla käyttäjä voi tutkia simuloinnin tuloksia ottamalla visualisointinäkymästä kaksiulotteisia leikkeitä halutussa suunnassa ja kohdassa. Leikkeitä voi värikoodata halutun suureen mukaan ja käyttäjä voi itse määrittää värikoodauksessa käytetyt värit. Kuvassa 3 näkyy välilehden ulkoasu.

Plane -osion avulla valitaan taso, jossa suunnassa leikataan.

Section position -osioon annetaan tieto leikkauksen kohdasta.

Color code by -kohdasta valitaan suure, jonka mukaan leike värikoodataan.

Scale -kohdasta valitaan skaalataanko värikoodaus lineaarisesti vai logaritmisesti.

Update -painiketta painamalla voidaan päivittää visualisointinäkymä.

4.6.4 Project

`Project`-välilehdellä käyttäjä voi asettaa tallennettaville kuville perusnimen. Välilehdeltä näkyy myös käytettävän näytetiedoston nimi. Kuvassa 3 näkyy välilehden ulkoasu.

Project name -kohdasta näkyy käytössä olevan projektin nimi.

Image file basename -kenttään syötetään nimi, joka tallennettaville kuville halutaan. Kun kuva tallennetaan alapalkissa olevaa `Save`-painiketta painamalla, kuvan nimeksi tulee `Project`-lehdellä annettu nimi, jonka perään lisätään juokseva numerointi.

4.7 Alapaneelin toiminnot

Käyttöliittymän alareunassa sijaitsevalle paneelille on sijoitettu erilaisia simulointi- ja visualisointikontrolleja.

Save image -painikkeen avulla käyttäjä voi tallentaa visualisointinäkymässä olevan kuvan. Kuva tallennetaan `Project`-lehdellä annetulla nimellä lisäämällä perään juokseva numero.

Reset scene -painikkeen avulla käyttäjä voi halutessaan resetoida näkymän, eli palauttaa visualisointinäkymän samanlaiseksi, kuin se alunperin oli ennen pyörittelyä ja zoomausta.

Permeabiliteettikäyrä havainnollistaa permeabiliteetin arvojen muutosta ja sen avulla voidaan seurata simulaatio etenemistä. Käyrän vieressä näkyy permeabiliteetin nykyinen arvo ja ero edelliseen arvoon. Käyrä tulee näkyviin, kun simulaatio on käynnissä.

Show axes -kohdan avulla voidaan valita, näytetäänkö koordinaattiakselit visualisointinäkymässä.

4.8 Tilapalkki

Tilapalkissa kerrotaan, mikä sovelluksen tila on kyseessä; tyhjä projekti, näyte olemassa, simulointi kesken vai simulointi valmis. Simuloinnin ollessa käynnissä tilapalkkiin havainnollistetaan tieto simulaation kulusta: `ProgressBar` kuvaa simuloinnin etenemistä.

Stop -painiketta painamalla simulointi voidaan keskeyttää.

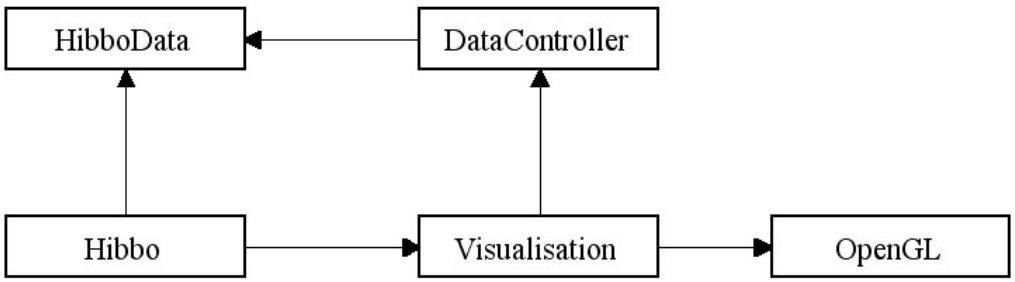
5 Toteutus

Tässä luvussa kerrotaan sovelluksen ohjelmoinnin toteuttamisesta. Luvussa käydään läpi käytetyt ohjelmointikielet, sovelluksen luokka- ja komponenttijako, koodauskäytännöt sekä visualisoinnissa käytetyt algoritmit.

Hibbo-sovellus toteutettiin Object Pascal -ohjelmointikielellä käyttäen Borlandin Delphi- ja Kylix-sovelluksia. Näiden sovelluskehittimien avulla pystyttiin tekemään sovellus, joka toimii sekä Linux- että Windows-ympäristössä. Hibbo-sovellus toteutettiin käyttäen Borlandin CLX-luokkia, jotka pohjautuvat Qt-kirjastoon ja toimivat molemmissa edellä mainituissa käyttöjärjestelmissä.

Seuraavissa luvuissa selvitetään, kuinka Hibbo-sovellus on toteutettu. Luvussa 5.1 kerrotaan, millaisiin osiin sovellus jakautuu. Luvussa 5.2 kuvataan tarkemmin sovelluksen *DataController*-osaa, luvussa 5.3 *OpenGL*-osaa, luvussa 5.4 *Visualisation*-osaa, luvussa 5.5 *HibboData*-osaa ja lopuksi luvussa 5.6 varsinaista *Hibbo*-osaa.

5.1 Yleiskuva



Kuva 4: Kaaviokuva sovelluksen rakenteesta.

Hibbo-projektissa ei toteutettu pelkästään yksinkertaista ohjelmaa, vaan yleiskäyttöinen visualisointijärjestelmä sekä sitä käyttävä räätälöity sovellus. Toteutettu järjestelmä jakautuu viiteen loogiseen osaan, jotka on esitetty kuvassa 4. Itse *Hibbo*-osa käsittää käyttöliittymän ikkunoinnin, projektinhallinnan ja yleisen käyttöliittymälogiikan. *Visualisation*-osa huolehtii visualisointinäkymien tuottamisesta, piirtämisestä ja ylläpitämisestä; piirtämisessä käytetään *OpenGL*-osan tarjoamia mahdollisuuksia. Näkymiä luodakseen *Visualisation*-osa tarvitsee dataa, jota se pyytää *DataController*-osan kautta, joka on rajapinta kaikenlaiseen numeeriseen dataan. *HibboData*-osa toteuttaa *DataController*-rajapinnan ja tarjoaa näkymiä Hibbo-so-

velluksen käsittelemään dataan. *Hibbo* ohjaa *HibboData*-osan toimintaa ja antaa sille tiedostot luettaviksi. Seuraavissa luvuissa tarkastellaan lähemmin kutakin näistä osista.

5.2 DataController

Toteutetun yleiskäyttöisen visualisointijärjestelmän ydin on rajapinta, jonka avulla voidaan käsitellä visualisoitavaa dataa. Rajapinnan ansiosta toteutetulla järjestelmällä voidaan visualisoida mitä tahansa numeerista, avaruutena esitettävää dataa.

5.2.1 TDataController

Tämä luokka on yleiskäyttöinen rajapinta, jonka avulla voi kapseloida tietoa. Rajapinta tarjoaa metodeja, joiden avulla saadaan selville, minkälaisia näkymiä tietoon on olemassa ja voidaan myös tutkia näitä näkymiä. Rajapinnassa ei oteta kantaa siihen, millä tavalla tieto tallennetaan luokan sisällä.

`TDataController`-järjestelmän takana on idea, että samasta datasta tarjotaan erityyppisiä näkymiä. Näkymätyyppejä voivat olla totuusarvo-, skalaari- ja vektorikenttänäkymät. Data ajatellaan avaruudeksi, josta kysellään arvoja koordinaattien avulla. Avaruuden dimensiota ei ole rajattu. Koordinaatit annetaan taulukkona, ja niiden arvot voivat olla joko kokonaislukuja tai reaalilukuja, eli dataan voidaan tarjota sekä jatkuvia että diskreettejä näkymiä. Datasta voidaan myös leikata vain osa antamalla kussakin koordinaattisuunnassa koordinaattien minimi- ja maksimiarvot. `TDataController`-luokan käyttämät tyyppimääritykset, luokan ominaisuudet sekä toiminnot on selostettu taulukoissa 1, 2 ja 3.

Taulukko 1: Luokan TDataController tyypinmääritykset.

Tyyppi	**Kuvaus**
TViewType	Tyyppi, jonka arvoina ovat kaikki mahdolliset näkymän tyypit. Mahdollisia arvoja ovat `vtNone`, `vtBinary`, `vtScalar`, `vtVector`.
jatkuu...	

Taulukko 1: Luokan TDataController tyypinmääritykset.

Tyyppi (jatkuu)	**Kuvaus**
TBinaryData	Datan näkymät, jotka ovat tyyppiä `vtBinary`, sisältävät tämän tyyppisiä alkioita. Alustavasti tämä tyyppi on määritelty samaksi kuin `Boolean`, mutta tarvittaessa tyyppi voitaisiin myöhemmin määritellä toisinkin.
TScalarData	Datan näkymät, jotka ovat tyyppiä `vtScalar`, sisältävät tämän tyyppisiä alkioita. Alustavasti tämä tyyppi on määritelty samaksi kuin `Double`, mutta tarvittaessa tyyppi voitaisiin myöhemmin määritellä toisinkin.
TVectorData	Datan näkymät, jotka ovat tyyppiä `vtVector`, sisältävät tämän tyyppisiä alkioita. Alustavasti tämä tyyppi on määritelty samaksi kuin `array of Double`, mutta tarvittaessa tyyppi voitaisiin myöhemmin määritellä toisinkin. Kyseessä on siis dynaaminen taulukko `Double`-tyyppisiä alkioita. Taulukon rajat saadaan kysyttyä käyttäen Delphin standardikirjaston *High-* ja *Low*-funktioita.
TBlockView	Tietue, joka sisältää osan näkymää taulukkona. Tietueen kenttinä ovat *Type*, joka on tyyppiä `TViewType`, *Data*, joka on osoitin datataulukkoon (tyyppiä `Pointer`, *Dimensions*, joka kertoo datan dimension, ja *Size*, joka kertoo datan koon eri koordinaattisuunnissa taulukkona. *Size*-taulukossa on *Dimensions* alkiota. *Data*-taulukko on yksiulotteinen ja se sisältää peräkkäin kaikki kyseessä olevan avaruuden osajoukon alkiot siten, että viimeinen koordinaattisuunta juoksee nopeimmin. Siis, alkio, jonka koordinaatit ovat $(k_1, k_2, \ldots, k_n)$ on taulukon paikassa $\sum_{i=1}^{n-1}(k_i \prod_{j=i}^{n-1} s_j)+k_n$, missä $(s_1, s_2, \ldots, s_n)$ ovat datan koot koordinaattisuunnissa.
TDescriptives	Tietue, joka sisältää datan tunnuslukuja. Tietueen kenttinä ovat *Min*, *Max* ja *Mean*, jotka ovat kaikki `Double`-tyyppisiä.
jatkuu...	

Taulukko 1: Luokan TDataController tyypinmääritykset.

Tyyppi (jatkuu)	**Kuvaus**
EDataController	Poikkeustyyppi, jota `TDataController`-rajapinnan toteuttavat luokat heittävät rajapinnan käsittelyyn liittyvissä virhetilanteissa. Poikkeusluokka on peritty suoraan `Exception`-luokasta, eikä se sisällä mitään uutta.
TInterpolation-Algorithm	Osoitin funktioon, jonka avulla voidaan saada jatkuva näkymä interpoloimalla diskreetin näkymän arvojen perusteella. *Parametrit:* Data (tyyppiä `TDataController`), ViewIndex (tyyppiä `Integer`), Coordinates (tyyppiä `array of Double`). *Paluuarvo:* interpoloitu arvo tyyppiä `TScalarData`.
THistogram	Luokka, joka kapseloi yksinkertaisen histogrammin. Luokalla on ominaisuuksina *Min* ja *Max* (tyyppiä `Extended`), jotka määräävät pienimmän ja suurimman arvon, mitä histogrammiin voi syöttää, *Divisions* (kokonaisluku), joka määrää, miten moneen osaan histogrammi jaetaan, *Frequency* (taulukko kokonaislukuja, jossa *Divisions* alkiota), joka sisältää kunkin jakovälin frekvenssit, *MaxFrequency* (kokonaisluku), joka kertoo histogrammin suurimman frekvenssin, *ValueCount* (kokonaisluku), joka kertoo histogrammiin syötettyjen arvojen lukumäärän sekä metodi *AddValue*, jolle annetaan parametrina `Extended`-tyyppinen arvo; vastaavan kohdan frekvenssiä lisätään yhdellä.

Taulukko 2: Luokan TDataController ominaisuudet.

Ominaisuus	**Kuvaus**
Dimensions	Kokonaisluku, joka kertoo datan ulottuvuuksien määrän, joka voi olla mikä tahansa positiivinen luku. Tämän ominaisuuden arvon voi lukea, mutta sitä ei voi muuttaa.
jatkuu...	

Taulukko 2: Luokan TDataController ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
Size	Kokonaislukutaulukko, joka sisältää datan koon kaikissa koordinaattisuunnissa. Taulukossa on *Dimensions* alkiota. Jos kokoa ei ole määritelty kysytyssä suunnassa tai kysytään tietoa taulukon ulkopuolelta, alkion arvo on negatiivinen luku. Tämän ominaisuuden arvoja voi lukea, mutta niitä ei voi muuttaa.
Min	Kokonaislukutaulukko, joka sisältää pienimmät käytössä olevat koordinaattien arvot kaikissa koordinaattisuunnissa. Nämä minimiarvot rajoittavat tarkasteltavissa olevaa dataa. Taulukossa on *Dimensions* alkiota. Jos minimiarvoa ei ole määritelty kysytyssä suunnassa tai kysytään tietoa taulukon ulkopuolelta, alkion arvo on negatiivinen luku. Tämän ominaisuuden arvoja voi sekä lukea että muuttaa. Alkion arvoksi ei kuitenkaan voi asettaa suurempaa lukua kuin *Max*-taulukon vastaavassa kohdassa oleva arvo. Jos tätä kuitenkin yritetään, tai yritetään kirjoittaa taulukon ulkopuolelle, heitetään poikkeus `EDataController`.
Max	Kokonaislukutaulukko, joka sisältää suurimmat käytössä olevat koordinaattien arvot kaikissa koordinaattisuunnissa. Nämä maksimiarvot rajoittavat tarkasteltavissa olevaa dataa. Taulukossa on *Dimensions* alkiota. Jos maksimiarvoa ei ole määritelty kysytyssä suunnassa tai kysytään tietoa taulukon ulkopuolelta, alkion arvo on negatiivinen luku. Tämän ominaisuuden arvoja voi sekä lukea että muuttaa. Alkion arvoksi ei kuitenkaan voi asettaa pienempää lukua kuin *Min*-taulukon vastaavassa kohdassa oleva arvo. Jos tätä kuitenkin yritetään, tai yritetään kirjoittaa taulukon ulkopuolelle, heitetään poikkeus `EDataController`
ViewCount	Kokonaisluku, joka kertoo saatavilla olevien näkymien määrän. Tämän ominaisuuden arvon voi lukea, mutta sitä ei voi muuttaa.
jatkuu...	

Taulukko 2: Luokan TDataController ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
ViewName	Taulukko merkkijonoja, joka kertoo eri näkymien nimet. Taulukossa on *ViewCount* alkiota. Tämän ominaisuuden arvoja voi lukea, mutta niitä ei voi muuttaa.
ViewType	Taulukko `TViewType`-tyyppisiä alkioita, joka kertoo eri näkymien tyypit. Taulukossa on *ViewCount* alkiota. Tämän ominaisuuden arvoja voi lukea, mutta niitä ei voi muuttaa.
Interpolate	Osoitin käytettävään interpolointifunktioon; ellei tätä ole asetettu, jatkuvia näkymiä ei voi muodostaa. Tyyppiä `TInterpolationAlgorithm`.

Taulukko 3: Luokan TDataController toiminnot.

Toiminto	**Kuvaus**
GetBinaryView	Palauttaa totuusarvotyyppisen alkion halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* `TBinaryData`-tyyppinen arvo.
GetBinaryView (jatkuva)	Palauttaa totuusarvotyyppisen alkion halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Koordinaatit annetaan reaalilukuina, joten näkymää voi tarkastella jatkuvana funktiona. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko reaalilukuja). *Paluuarvo:* `TBinaryData`-tyyppinen arvo.
jatkuu...	

Taulukko 3: Luokan TDataController toiminnot.

Toiminto (jatkuu)	**Kuvaus**
GetScalarView	Palauttaa skalaariarvotyyppisen alkion halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* `TScalarData`-tyyppinen arvo.
GetScalarView (jatkuva)	Palauttaa skalaariarvotyyppisen alkion halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Koordinaatit annetaan reaalilukuina, joten näkymää voi tarkastella jatkuvana funktiona. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko reaalilukuja). *Paluuarvo:* `TScalarData`-tyyppinen arvo.
GetVectorView	Palauttaa vektoriarvotyyppisen alkion halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* `TVectorData`-tyyppinen arvo.
GetVectorView (jatkuva)	Palauttaa vektoriarvotyyppisen alkion halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Koordinaatit annetaan reaalilukuina, joten näkymää voi tarkastella jatkuvana funktiona. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko reaalilukuja). *Paluuarvo:* `TVectorData`-tyyppinen arvo.
jatkuu...	

Taulukko 3: Luokan TDataController toiminnot.

Toiminto (jatkuu)	**Kuvaus**
GetBinaryBlockView	Palauttaa taulukollisen totuusarvotyyppisiä alkioita halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Koordinaatiksi voidaan antaa negatiivinen luku, jolloin palautetaan tässä suunnassa kaikki alkiot koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* `TBlockView`-tyyppinen arvo.
GetScalarBlockView	Palauttaa taulukollisen skalaariarvotyyppisiä alkioita halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Koordinaatiksi voidaan antaa negatiivinen luku, jolloin palautetaan tässä suunnassa kaikki alkiot koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* `TBlockView`-tyyppinen arvo.
GetVectorBlockView	Palauttaa taulukollisen vektoriarvotyyppisiä alkioita halutusta näkymästä näkymäindeksin ja koordinaattien perusteella. Koordinaatiksi voidaan antaa negatiivinen luku, jolloin palautetaan tässä suunnassa kaikki alkiot koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* `TBlockView`-tyyppinen arvo.
jatkuu...	

Taulukko 3: Luokan TDataController toiminnot.

Toiminto (jatkuu)	**Kuvaus**
GetViewMin	Palauttaa minimiarvon halutusta osasta näkymää. Parametrina annetaan koordinaatit, ja asettamalla koordinaatin arvoksi negatiivinen luku huomioon otetaan kaikki arvot tässä suunnassa koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. Paluuarvo on skalaarityyppinen, joten totuusarvokentille toiminta on määrittelemätön. Vektorikentän ollessa kyseessä palautetaan minimiarvo vektorien normeista. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* minimiarvo `TScalarData`-tyyppisenä.
GetViewMax	Palauttaa maksimiarvon halutusta osasta näkymää. Parametrina annetaan koordinaatit, ja asettamalla koordinaatin arvoksi negatiivinen luku huomioon otetaan kaikki arvot tässä suunnassa koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. Paluuarvo on skalaarityyppinen, joten totuusarvokentille toiminta on määrittelemätön. Vektorikentän ollessa kyseessä palautetaan maksimiarvo vektorien normeista. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* maksimiarvo `TScalarData`-tyyppisenä.
jatkuu...	

Taulukko 3: Luokan TDataController toiminnot.

Toiminto (jatkuu)	**Kuvaus**
GetViewMean	Palauttaa aritmeettisen keskiarvon halutusta osasta näkymää. Parametrina annetaan koordinaatit, ja asettamalla koordinaatin arvoksi negatiivinen luku huomioon otetaan kaikki arvot tässä suunnassa koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. Paluuarvo on skalaarityyppinen, joten totuusarvokentille toiminta on määrittelemätön. Vektorikentän ollessa kyseessä palautetaan keskiarvo vektorien normeista. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* keskiarvo `TScalarData`-tyyppisenä.
GetViewDescriptives	Palauttaa minimi-, maksimi- ja keskiarvon halutusta osasta näkymää. Tekee siis saman kuin *GetViewMin*, *GetViewMax* ja *GetViewMean* yhdellä kertaa. Parametrina annetaan koordinaatit, ja asettamalla koordinaatin arvoksi negatiivinen luku huomioon otetaan kaikki arvot tässä suunnassa koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. Palautettavat tunnusluvut ovat skalaarityyppisiä, joten totuusarvokentille toiminta on määrittelemätön. Vektorikentän ollessa kyseessä palautetaan tunnusluvut vektorien normeista. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* tunnusluvut `TDescriptives`-tyyppisenä.
jatkuu...	

Taulukko 3: Luokan TDataController toiminnot.

Toiminto (jatkuu)	**Kuvaus**
GetViewHistogram	Palauttaa histogrammin halutusta osasta näkymää. Parametrina annetaan koordinaatit, ja asettamalla koordinaatin arvoksi negatiivinen lukuu huomioon otetaan kaikki arvot tässä suunnassa koordinaattien minimi- ja maksimiarvojen väliltä. Jos koordinaattien arvot ovat annettujen minimi- ja maksimiarvojen ulkopuolella, heitetään poikkeus `EDataController`. Paluuarvo on tyyppiä `THistogram`. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* histogrammi `THistogram`-tyyppisenä.

5.2.2 TPathSolverAlgorithm

Tämä luokka määrittää abstraktin rajapinnan polunratkaisija-algoritmille. Se tutkii jotakin `TDataController`in kautta saamaansa vektorikenttänäkymää ja etsii seuraavan pisteen polulla, kun sille annetaan polun edellinen piste. Käytettävät vektorit voidaan haluttaessa skaalata. Luokan rajapinta on esitetty taulukossa 4.

Taulukko 4: Luokan TPathSolverAlgorithm ominaisuudet.

Ominaisuus	**Kuvaus**
Data	Data, johon polut sijoitetaan. Tyyppiä `TDataController`.
ViewIndex	Käytettävän näkymän indeksi, tyyppiä `Integer`.
ViewType	Kertoo käytössä olevan näkymän tyypin. Tämä arvo kysytään suoraan Datalta. Tyyppiä `TViewType`. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
Scale	Totuusarvo, joka määrää, skaalataanko käytettyjä vektoreita vai ei. Tyyppiä `Boolean`.
ScaleFactor	Jos vektorien skaalaus on valittu käyttöön, kaikki vektorit kerrotaan tällä skalaarilla. Tyyppiä `Double`.
jatkuu...	

Taulukko 4: Luokan TPathSolverAlgorithm ominaisuudet.

Ominaisuus (jatkuu)	Kuvaus
NextPoint	Aliohjelma, joka palauttaa seuraavan pisteen polulla. *Parametrit:* referenssit alku- ja loppupisteeseen, tyyppiä `array of Double`; haluttu askelpituus, tyyppiä `Double`.

5.3 OpenGL

Visualisointinäkymien piirtämisessä käytetään apuna OpenGL-kirjastoa, jonka hyödyntämiseksi käytössä on joitakin yleiskäyttöisiä luokkia. Nämä luokat eivät kuulu Hibbo-sovellukseen, mutta tässä luvussa selostetaan tämän sovelluksen kannalta tärkeimpiä asioita näiden luokkien toiminnasta

5.3.1 TOpenGLPanel

Tämä luokka on komponentista `TCustomControl` peritty paneelikomponentti. Se toimii kuten normaalit Delphin paneelit, mutta sen päälle voi piirtää mitä tahansa, mitä OpenGL-käskyillä on mahdollista tehdä. Luokka `TOpenGLScene` hoitaa kaiken yhteyden paneeliin, joten Hibbo-projektin puitteissa tästä ei tarvitse huolehtia. Paneeli luodaan kuten `TPanel`-luokan olio, sille annetaan *Parent* ja se asetetaan paikalleen. Sitten paneeli sijoitetaan luokan `TVisualisationController` *Panel*-ominaisuuden arvoksi, ja sen jälkeen kaikki visualisoinnit tulevat piirretyiksi tälle paneelille.

5.3.2 TOpenGLObject

Tämä luokka on peruspalikka OpenGL-näkymien rakentamisessa. Se määrittelee rajapinnan ja kantaluokan josta peritään erikoistuneempia objekteja. Luokan rajapinta on esitetty taulukossa 5.

Taulukko 5: Luokan TOpenGLObject ominaisuudet.

Ominaisuus	**Kuvaus**
Owner	Viittaus näkymään, joka omistaa tämän objektin ja johon objekti kuuluu. Tyyppiä `TOpenGLScene`. Näkymä huolehtii kaikkien sille kuuluvien objektien piirtämisestä.
RotationAxis	Akseli, jonka ympäri objektia kierretään. Kolmiulotteisen avaruuden vektori, tyyppiä `TVector3f`, joka on käytännössä kolmialkioinen taulukko reaalilukuja.
Angle	Kulma, jonka verran objektia kierretään asetetun akselin ympäri. Kiertokulma annetaan asteissa, ja positiiviset kulmat kasvavat vastapäivään. Tyyppiä `Single`.
TranslateVector	Vektori, joka osoittaa, miten objektia siirretään. Kolmiulotteisen avaruuden vektori, tyyppiä `TVector3f`, joka on käytännössä kolmialkioinen taulukko reaalilukuja.
Visible	Totuusarvo, joka säätää, piirretäänkö objektia vai ei. Näkymäolio jättää piirtämättä sellaiset objektit, joiden *Visible*-arvo on `False`. Tyyppiä `Boolean`.

5.3.3 TOpenGLCamera

Tämä luokka on kamera, joka näyttää kuvaa kulloisestakin näkymästä. Tämän luokan toiminnasta ei tarvitse kummemmin välittää: ainoa, mitä tarvitsee tietää, on se, että luokalla `TVisualisationController` on ominaisuus nimeltä *Scene*, joka on tyyppiä `TOpenGLScene`. Tällä taas on ominaisuus *ActiveCamera*, joka on tyyppiä `TOpenGLCamera`. Kameraa voi kääntää käskyillä *RotateHorizontal* ja *RotateVertical*, siirtää käskyillä *PanHorizontal* ja *PanVertical* ja lähentää ja loitontaa käskyllä *Zoom*.

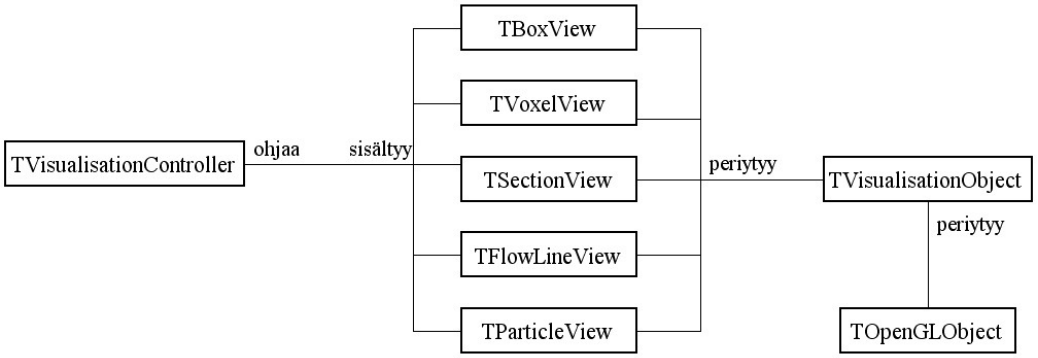
5.3.4 TOpenGLScene

Tämä luokka on näkymä, joka sisältää joukon `TOpenGLObject`-luokan olioita. Luokan toiminnasta ei tarvitse tietää mitään muuta kuin se, että sillä on yllä mainittu *ActiveCamera*-ominaisuus. `TVisualisationController`-luokka luo näkymän ja siihen erilaiset visualisointinäkymät, ja tähän näkymäolioon pääsee käsiksi *Scene*-ominaisuuden kautta.

5.4 Visualisation

Sovelluksessa käytetty visualisointiosa on myös yleiskäyttöinen. Se hyödyntää `THibboData`-rajapintaa ja luo siltä saamansa datan perusteella visualisointinäkymiä. Samat näkymät voidaan luoda mistä tahansa datasta. Tällä hetkellä datan pitää olla kolmiulotteista, mutta järjestelmää on tarkoitus kehittää siten, että näkymiä voitaisiin luoda kaikenlaisesta datasta.

Visualisointijärjestelmä toimii siten, että kutakin näkymätyyppiä kohti on oma luokka, joka saa datan ja käytettävän näkymäindeksin, ja muodostaa itse näkymän tämän datan perusteella. Uusia näkymiä voidaan kehittää kohtuullisen helposti luomalla uudenlaisia näkymäluokkia. Näkymäluokkien lisäksi on olemassa ohjainluokka, joka kontrolloi näkymien toimintaa. Visualisointiosa on esitetty kaaviona kuvassa 5.



Kuva 5: Kaaviokuva Visualisation-osasta.

5.4.1 TVisualisationObject

Tämä luokka on abstrakti rajapinta erilaisia visualisointinäkymiä toteuttavia luokkia varten. Luokka on peritty luokasta `TOpenGLObject`. Jäljempänä luvussa 5.4.7 esiteltävä `TVisualisationController`-luokka hyödyntää tätä rajapintaa. Hibbo-sovelluksen hyödyntämässä visualisointijärjestelmässä on käytössä viisi erikoistunutta versiota `TVisualisationObject`ista: `TBoxView`, `TVoxelView`, `TSectionView`, `TFlowLineView` ja `TParticleView`. Nämä luokat esitellään myöhemmin.

Visualisointinäkymien on tarkoitus luoda jonkinlainen kolmiulotteinen malli niille annetun `TDataController`-rajapinnan perusteella. Mallin koko tulee asettaa siten, että datan suurin ulottuvuus on ykkösen pituinen. Silloin näkymiä on helppo skaalata halutun kokoisiksi. Samoin näkymät tulee keskittää origoon, ja datan ensimmäinen koordinaattisuunta

tulee asettaa x-akselin suuntaan, toinen y-akselin suuntaan ja kolmas z-akselin suuntaan. Tällä lailla kaikki näkymät saadaan vastaamaan toisiaan ja ne voidaan näyttää samassa koordinaatistossa.

Luokka periytyy luokasta `TOpenGLObject`, ja sen sisältämät lisäykset rajapintaan on esitetty taulukossa 6.

Taulukko 6: Luokan TVisualisationObject ominaisuudet.

Ominaisuus	**Kuvaus**
Controller	Visualisointiluokka, joka omistaa tämän objektin. Tyyppiä `TVisualisationController`. Tämän ominaisuuden arvon voi vain lukea, sitä ei voi muuttaa kesken kaiken. Alkuperäinen arvo annetaan konstruktorissa.
Panel	Paneeli, jolle tämä objekti piirretään. Tyyppiä `TOpenGLPanel`.
Enabled	Totuusarvo, jolla voidaan asettaa objekti päälle ja pois päältä. Hyödyllinen esimerkiksi silloin, kun halutaan muutaa objektin asetuksia ja estää näkymän päivitys, kunnes kaikki asetukset ovat kohdillaan.
Data	Data, jota käytetään näkymän muodostamiseen. Tyyppiä `TDataController`.
ViewIndex	Käytettävän datan näkymän indeksi. Tyyppiä `Integer`.
ColorCode	Käytettävä värikoodaus. Tyyppiä `TColorCode`.
Width	Näkymän leveys mallikoordinaateissa. Tyyppiä `Real`, väliltä $[0, 1]$. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
Height	Näkymän korkeus mallikoordinaateissa. Tyyppiä `Real`, väliltä $[0, 1]$. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
Depth	Näkymän syvyys mallikoordinaateissa. Tyyppiä `Real`, väliltä $[0, 1]$. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
SizeScaleFactor	Näkymän skaalauskerroin. Kaikki näkymän ulottuvuudet, jotka saadaan edellä olevista ominaisuuksista *Width*, *Height*, *Depth*, kerrotaan tällä luvulla. Näkymän ulottuvuudet mallikoordinaateissa tulevat siis loppujen lopuksi olemaan väliltä [0,SizeScaleFactor].
jatkuu...	

Taulukko 6: Luokan TVisualisationObject ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
Create	Konstruktori, joka luo objektin. *Parametrit:* näkymä, joka omistaa objektin, tyyppiä `TOpenGLScene`; visualisointiluokka, joka ohjaa tätä objektia, tyyppiä `TVisualisationController`.
CreateView	Metodi, joka luo näkymän nykyisten asetusten mukaisesti.
DestroyView	Metodi, joka tuhoaa näkymän. Tämän metodin kutsun jälkeen objekti ei saa enää piirtää mitään ennen uutta `CreateView`-metodin kutsua.

5.4.2 TBoxView

Tämä luokka toteuttaa annetusta datasta laatikkonäkymän. Se siis piirtää dataa esittävän kolmiulotteisen laatikon origoon keskitettynä. Se on kätevä, kun se näytetään yhtä aikaa muiden näkymien kanssa: jos jokin toinen näkymä esittää vain osan datasta, kuten esimerkiksi `TSectionObject` esittää vain yhden kaksiulotteisen leikkauspinnan ja `TFlowLineObject` esittää joukon virtaviivoja, laatikkonäkymän avulla näkee helposti, mistä osasta dataa on kyse.

Luokka periytyy luokasta `TVisualisationObject`. Luokassa ei ole mitään muuta omaa kuin uusi versio `TOpenGLObject`in *Draw*-metodista, joka piirtää laatikon oikealla tavalla oikean kokoisena oikeaan paikkaan.

5.4.3 TVoxelView

Tämä luokka käyttää jotakin datasta saatavaa totuusarvonäkymää ja piirtää sen perusteella eräänlaisen kolmiulotteisen binäärikuvan, vokselinäkymän. Siinä datassa arvon tosi saavat pisteet esitetään kuutiona eli vokselina ja arvon epätosi saavat pisteet tyhjänä. Totuusarvokenttä hahmottuu siis kasana laatikoita.

Näkymä muodostetaan siten, että data käydään läpi asetettua totuusarvonäkymää käyttäen. Jokaiseen kohtaan, johon saadaan tulokseksi `True` asetetaan kuutio. Varsinaisia kuutioita piirrettäessä piirretään vain näkyvissä olevat sivut: esimerkiksi jos kuution vasemmalla puolella on kuutio, vasenta sivua ei piirretä ollenkaan. Näin saadaan hiukan optimoitua piirtämistä.

Luokka periytyy luokasta `TVisualisationObject`. Luokassa ei ole mitään omaa, vain päivitetty versio metodeista *CreateView*, *DestroyView* ja

Draw.

5.4.4 TSectionView

Tämä luokka käyttää jotakin skalaarinäkymää ja luo datasta kaksiulotteisia poikkileikkauksia haluttuun suuntaan ja halutusta kohdasta. Luokalle voi antaa värikoodauksen, jonka mukaan poikkileikkauspinnalla esiintyvät arvot värikoodataan valitun näkymän arvojen mukaisesti. Poikkileikkauspinta piirretään oikeaan paikkaan datassa.

Poikkileikkauspinnat ovat teksturoituja suorakaiteita, jotka asetetaan oikeaan paikkaan. Tekstuurit muodostetaan siten, että poikkileikkauspinnan leveydestä ja korkeudesta valitaan suurempi, ja tekstuurin leveydeksi valitaan tätä arvoa lähinnä suurempi kakkosen potenssi; neliön muotoisia tekstuureja on helpointa käsitellä, ja OpenGL-tekstuurin leveyden ja korkeuden täytyy olla kakkosen potenssi. Data käydään läpi leikkauspinta kerrallaan käyttäen `TDataController`ia; jokaisessa pisteessä luetaan asetetun näkymäindeksin perusteella visualisoitava arvo ja etsitään värikoodista luettua arvoa vastaava väri. Värit tallennetaan taulukkoon, ja lopuksi taulukko skaalataan halutun tekstuurin kokoiseksi. Jokaisen pinnan tekstuuri lasketaan valmiiksi keskusmuistiin ja leikkauspintoja piirrettäessä tarvittava tekstuuri luetaan näytönohjaimen muistiin.

Luokka periytyy luokasta `TVisualisationObject`, ja sen sisältämät lisäykset rajapintaan esitetään taulukoissa 7 ja 8. Näiden lisäksi luokkaan kuuluu päivitetty versio metodeista *CreateView*, *DestroyView* ja *Draw*.

Taulukko 7: Luokan TSectionView tyypinmääritykset.

Tyyppi	**Kuvaus**
TPlane	Määrittää kolmiulotteisen avaruuden tason. Arvoja ovat `pXY`, `pXZ` ja `pYZ`.

Taulukko 8: Luokan TSectionView ominaisuudet.

Ominaisuus	**Kuvaus**
Plane	Taso, jonka suuntaan dataa leikataan. Tyyppiä `TPlane`.
jatkuu...	

Taulukko 8: Luokan TSectionView ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
Position	Kohta, jolta leikkaus tehdään; käytännössä koordinaatti kohtisuorassa leikkaustasoa vastaan olevassa suunnassa. Tyyppiä `Integer`.
MinPosition	*Position*-ominaisuuden pienin sallittu arvo. Tämä kysytään suoraan *Data*n Min-ominaisuudesta käyttäen oikeaa koordinaattisuuntaa. Tyyppiä `Integer`. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
MaxPosition	*Position*-ominaisuuden suurin sallittu arvo. Tämä kysytään suoraan *Data*n Max-ominaisuudesta käyttäen oikeaa koordinaattisuuntaa. Tyyppiä `Integer`. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
Size	Datan koko leikkauspintaa vastaan kohtisuorassa koordinaattisuunnassa. Kysytään suoraan *Data*sta käyttäen oikeaa koordinaattisuuntaa. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
SectionWidth	Datan koko leikkauspinnan leveyssuunnassa. Kysytään suoraan *Data*sta käyttäen oikeaa koordinaattisuuntaa. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
SectionHeight	Datan koko leikkauspinnan korkeussuunnassa. Kysytään suoran *Data*sta käyttäen oikeaa koordinaattisuuntaa. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.

5.4.5 TFlowLineView

Tämä luokka käyttää jotakin datasta saatavaa vektorikenttänäkymää ja luo kenttään virtaviivoja sille annetun polunratkaisualgoritmin avulla. Viivojen lähtötaso ja -suunta voidaan antaa.

Virtaviivat tallennetaan listana tietueita, joissa on polun pisteen paikka taulukkona `Double`-tyyppisiä reaalilukuja, näkymien arvot samoin taulukkona `Double`-tyyppisiä reaalilukuja sekä kyseisen polun pisteen väri `TColor`-tyyppisenä. Kutakin pistettä kohti tallennetaan arvo jokaisesta datan tarjoamasta skalaarityyppisestä näkymästä kyseisessä pisteessä. Tällä tavoin voidaan tarvittaessa vaihtaa visualisoitavaa suuretta.

Polkuja luodaan *Amount*-ominaisuuden osoittama määrä. Polkujen muo-

dostus tapahtuu siten, että kunkin polun aloituspaikka arvotaan valitulta tasolta. Sitten etsitään tästä pisteestä alkaen kullekin polulle uusia pisteitä käyttäen asetettua polunratkaisualgoritmia, kunnes päädytään ulos näytteestä tai polulla on enemmän kuin 200 pistettä. Askelpituutena käytetään ykköstä.

Luokka periytyy luokasta *TVisualisationObject*, ja sen sisältämät lisäykset rajapintaan esitetään taulukossa 9. Lisäksi luokka sisältää päivitetyt versiot metodeista *CreateView*, *DestroyView* ja *Draw*. On huomattava, että ominaisuus *ViewIndex* tarkoittaa virtaviivojen yhteydessä viivojen värikoodauksessa käytettävää skalaarinäkymää, eikä vektorinäkymää, jonka perusteella viivat lasketaan.

Taulukko 9: Luokan TFlowLineView ominaisuudet.

Ominaisuus	**Kuvaus**
Plane	Taso, jolta virtaviivat lähtevät liikkeelle. Tyyppiä `TPlane`.
Position	Kohta, jolle lähtötaso asetetaan; käytännössä koordinaatti lähtötasoa vastaan kohtisuorassa olevassa koordinaattisuunnassa.
MinPosition	*Position*-ominaisuuden pienin sallittu arvo. Tämä kysytään suoraan *Data*n Min-ominaisuudesta käyttäen oikeaa koordinaattisuuntaa. Tyyppiä `Integer`. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
MaxPosition	*Position*-ominaisuuden suurin sallittu arvo. Tämä kysytään suoraan *Data*n Max-ominaisuudesta käyttäen oikeaa koordinaattisuuntaa. Tyyppiä `Integer`. Tämän ominaisuuden arvo voidaan vain kysyä, sitä ei voi muuttaa.
Amount	Lähetettävien virtaviivojen määrä. Tämä määrä viivoja asetetaan valitulle tasolle. Tyyppiä `Integer`.
PathSolver	Polunratkaisija virtaviivojen etsimistä varten. Polun pisteet haetaan käyttäen tätä algoritmia ja asetettuja lähtöpisteitä valitulla tasolla. Polkujen laskemiseen käytetään ratkaisijan käyttämää vektorinäkymää. Tyyppiä `TPathSolverAlgorithm`.
jatkuu...	

Taulukko 9: Luokan TFlowLineView ominaisuudet.

Ominaisuus (jatkuu)	Kuvaus
CreateColors	Metodi, jolla voi luoda uudelleen virtaviivojen värityksen tarvitsematta laskea viivojen reittiä uudestaan. Näkymäindeksiä ja käytettävää värikoodausta pystyy vaihtamaan.

5.4.6 TParticleView

Tämä luokka käyttää jotakin datasta saatavaa vektorikenttänäkymää ja luo sen perusteella partikkelianimaation. Partikkelien lähtötason ja -kohdan voi valita. Objektin näyttämän animaatioruudun numero voidaan asettaa. Partikkelien reitit lasketaan muistiin, ja animaation toistaminen tapahtuu valitsemalla sopivin väliajoin seuraavan ruudun numero näytettäväksi. Partikkelien reittien muodostus tapahtuu samalla lailla kuin virtaviivoilla. Ruutujen määräksi tulee pisimmän polun pisteiden lukumäärä.

Luokka periytyy luokasta `TFlowLineView`, ja sen sisältämät lisäykset rajapintaan esitetään taulukossa 10. Lisäksi luokka sisältää päivitetyn version metodeista *CreateView*, *DestroyView* ja *Draw*.

Taulukko 10: Luokan TParticleView ominaisuudet.

Ominaisuus	Kuvaus
FrameCount	Animaation sisältämien ruutujen määrä. Tyyppiä `Integer`. Tämän ominaisuuden arvon voi vain lukea, sitä ei voi muuttaa.
Frame	Nykyisen animaatioruudun numero. Tyyppiä `Integer`.
ParticleSize	Partikkelin koko datapisteissä. Arvo 1 tekee partikkeleista suhteessa saman kokoisia kuin datapisteetkin. Tyyppiä `Double`.

5.4.7 TVisualisationController

Tämä luokka pyytää tietoa `TDataController`-rajapinnan toteuttavista luokista ja luo datasta erilaisia visualisointinäkymiä. Luokka tarjoaa myös metodeja visualisointinäkymän muokkaamiseen. Visualisointiin käytetään OpenGL-komponentteja, mutta `VisualisationController` huolehtii yhteyksistä niihin, joten näiden komponenttien yksityiskohdat eivät vai-

kuta Hibbo-sovelluksen toteuttamiseen.

`VisualisationController`ia on tarkoitus käyttää siten, että ensin luodaan jokin `TDataController`-tyyppinen olio sekä `TOpenGLPanel`-luokan olio, jotka sijoitetaan `TVisualisationController`in ominaisuuksiin *Data* ja *Panel*. Luokan ominaisuudet ja toiminnot on esitetty taulukossa 11.

Taulukko 11: Luokan TVisualisationController ominaisuudet.

Ominaisuus	**Kuvaus**
Data	Viittaus *DataController*-rajapinnan toteuttavaan luokkaan, jolta kysellään visualisoitava data; tyyppiä `TDataController`. Ennen kuin visualisointeja voidaan tehdä, on luotava `TDataController`-olio ja asetettava se tämän ominaisuuden arvoksi.
Panel	Viittaus paneeliin, jonka päälle visualisointinäkymät piirretään. Tyyppiä `TOpenGLPanel`. Tämä arvo kopioituu automaattisesti kaikkien visualisointiobjektien vastaavan ominaisuuden arvoksi. Ennen kuin visualisointeja voidaan tehdä, on luotava `TOpenGLPanel`-olio, asetettava se paikoilleen ja asetettava luotu paneeli tämän ominaisuuden arvoksi.
Scene	Viittaus näkymään, johon visualisointiobjektit kuuluvat. Tyyppiä `TOpenGLScene`. `VisualisationController` luo itse tämän olion; ominaisuuden arvo voidaan vain lukea, sitä ei voi muuttaa.
Voxel	Viittaus `TVoxelView`-tyyppiseen olioon. Sen avulla voidaan säätää näytettyä vokselinäkymää. `VisualisationController` luo itse tämän olion; ominaisuuden arvo voidaan vain lukea, sitä ei voi muuttaa.
Section	Viittaus `TSectionView`-tyyppiseen olioon. Sen avulla voidaan säätää näytettyä leikenäkymää. `VisualisationController` luo itse tämän olion; ominaisuuden arvo voidaan vain lukea, sitä ei voi muuttaa.
jatkuu...	

Taulukko 11: Luokan TVisualisationController ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
FlowLine	Viittaus `TFlowLineView`-tyyppiseen olioon. Sen avulla voidaan säätää näytettyä virtaviivanäkymää. `VisualisationController` luo itse tämän olion; ominaisuuden arvo voidaan vain lukea, sitä ei voi muuttaa.
Particle	Viittaus `TParticleView`-tyyppiseen olioon. Sen avulla voidaan säätää näytettyä partikkelinäkymää. `VisualisationController` luo itse tämän olion; ominaisuuden arvo voidaan vain lukea, sitä ei voi muuttaa.
SizeScaleFactor	Näkymien skaalauskerroin. Tämän ominaisuuden arvo kopioituu automaattisesti kaikkien visualisointiobjektien vastaavan ominaisuuden arvoksi. Näin kaikki näkymät saadaan skaalattua samalla lailla.
ShowBoxView	Totuusarvo, joka säätää, näytetäänkö laatikkonäkymää vai ei. Tyyppiä `Boolean`.
ShowVoxelView	Totuusarvo, joka säätää, näytetäänkö vokselinäkymää vai ei. Tyyppiä `Boolean`.
ShowSectionView	Totuusarvo, joka säätää, näytetäänkö leikkausnäkymää vai ei. Tyyppiä `Boolean`.
ShowFlowLineView	Totuusarvo, joka säätää, näytetäänkö virtaviivanäkymää vai ei. Tyyppiä `Boolean`.
ShowParticleView	Totuusarvo, joka säätää, näytetäänkö partikkelinäkymää vai ei. Tyyppiä `Boolean`.
UpdateViews	Metodi, joka kutsuu kaikkien visualisointiobjektien *CreateView*-metodia.
ClearViews	Metodi, joka kutsuu kaikkien visualisointiobjektien *DestroyView*-metodia.

5.5 HibboData

Tähän osaan kuuluu vain luokka `THibboData`. Tämä luokka toteuttaa `TDataController`-rajapinnan ja tarjoaa sen kautta näkymiä simulointidataan. Luokka huolehtii tarvittavien tiedostojen lukemisesta ja niiden tietojen tallentamisesta. Luokassa tallennetaan tiedostoista luetut tiedot taulukossa, jossa on neljä reaalilukua (tyyppiä `Single`) jokaista hilakoppia kohti. Myös pelkän näytteen tiedot tallennetaan samassa taulukossa siten, että kiintoaineen pisteet asetetaan kakkosiksi ja nesteen pisteet nolliksi.

Tällä tavoin tarvitsee käsitellä vain yhtä taulukkoa, eikä tiedon toistoa ole; simuloidussa datassa on joka tapauksessa tieto kiintoaineen ja nesteen sijainnista. Haittapuolena on turha muistin käyttö silloin, kun käytössä on pelkkä näyte, ja on aavistuksen verran hitaampaa selvittää, onko piste nestettä vai kiintoainetta, kuin jos käytettäisiin `Boolean`-taulukkoa.

Rajapinnan kautta tarjotaan totuusarvotyyppinen näkymä näytteeseen (onko piste kiintoainetta vai ei), skalaarikenttinä paine, nopeuskomponentit ja vauhti eli nopeusvektorin normi, sekä vektorikenttinä nopeusvektorit sellaisinaan ja normeerattuina. Luokan käyttämät tyyppimääritykset sekä sen ominaisuudet on esitetty taulukoissa 12 ja 13.

Taulukko 12: Luokan THibboData tyypinmääritykset.

Tyyppi	**Kuvaus**
TQuantity	Tyyppi, jonka arvoina ovat erilaiset visualisoitavissa olevat suureet. Arvoina `vqPressure`, `vqSpeed`, `vqXVelocity`, `vqYVelocity`, `vqZVelocity` ja `vqIndex`.
TArrayType	Tietue, joka sisältää yhden datapisteen tiedot. Sisältää neljä `Single`-tyyppistä arvoa, *V_X*, *V_y*, *V_z* ja *P*, jotka sisältävät x-, y- ja z-suuntaiset nopeuskomponentit sekä paineen.

Taulukko 13: Luokan THibboData ominaisuudet.

Ominaisuus	**Kuvaus**
ReadFiles	Metodi, joka lukee tiedostosta datan. Jos parametrina annetaan tulostiedosto, luetaan vain se. Jos tulostiedostoa ei ole annettu, luetaan näytetiedosto. Jos sitäkään ei ole annettu, aiheutuu poikkeus. *Parametrit:* näytetiedosto ja tulostiedoston perusnimi; tiedostojen nimet polkuineen (merkkijonoja).

5.6 Hibbo

Hibbo-ohjelmassa on kolme omaa luokkaa, `Project`, `HibboData` ja `EvolFile`. Näiden lisäksi on luonnollisesti käyttöliittymän lomakeluokat. Seuraavassa selostetaan nämä kolme tietoja käsittelevää luokkaa. Lopuksi ku-

vataan visualisoinnissa käytettäviä algoritmeja sekä luokkien suhteita.

5.6.1 TProject

Tämä luokka huolehtii projektitiedoston lukemisesta ja kirjoittamisesta sekä ylläpitää projektin asetuksia ja simulaation parametreja. Projektin kautta voidaan luoda näyte ja aloittaa ja keskeyttää simulaatio. `Project`-luokka luo `THibboData`- ja `TEvolFile`-luokan oliot ja ohjaa niitä. Luokan käyttämät tyyppimääritykset ja sen ominaisuudet on kuvattu taulukoissa 14 ja 15.

Taulukko 14: Luokan TProject tyypinmääritykset.

Tyyppi	**Kuvaus**
TProjectState	Tyyppi, jonka arvoina ovat projektin erilaiset tilat. Arvoina `psEmpty`, `psSample`, `psSimulating`, `psSimulated`.
TColorSettings	Tyyppi, joka sisältää visualisoinnin ylä- ja alarajan värin sekä ulkopuolisen värin `TColor`-tyyppisinä.

Taulukko 15: Luokan TProject ominaisuudet.

Ominaisuus	**Kuvaus**
SampleFileName	Käytössä olevan näytetiedoston nimi merkkijonona. Kun tämä nimi asetetaan, se välitetään `THibboData`-luokalle, jotta se osaa lukea näytetiedoston.
ResultFileBaseName	Käytössä olevien tulostiedostojen perusnimi merkkijonona. Kun tämä nimi asetetaan, se välitetään `THibboData`-luokalle, jotta se osaa lukea tulostiedostot.
Relaxation-ParameterTau	Simulointiparametri, joka määrää simulointinesteen viskositeetin.
TimeSteps	Simulointiparametri, joka kertoo suoritettavat simulointikierrokset.
InitialVelocity	Simulointiparametri, joka kertoo nesteen alkunopeuden z-suunnassa.
jatkuu...	

Taulukko 15: Luokan TProject ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
FluidLayerThickness	Simulointiparametri, joka kertoo vapaan nestekerroksen paksuuden sekä näytteen ylä-, että alapuolella.
SimulationMode	Simulointiparametri, joka kertoo käytettävän simulointimoodin.
LengthScale	Simulointiparametri, joka kertoo mallin koon suhteessa todelliseen maailmaan.
ConvergenceCriterion	Simulointiparametri, simuloinnin lopetusehto.
State	Projektin tila. Tilana voi olla tyhjä projekti, näyte, simulointi kesken tai simuloitu.
Name	Projektin nimi merkkijonona. Tätä nimeä käytetään projektikansion sekä projektitiedoston nimeämiseen.
Directory	Hakemisto, johon projektin tiedostot on tallennettu, merkkijonona. Tämän ominaisuuden arvo voidaan vain lukea, sitä ei voi muuttaa.
ImageFileBaseName	Kuvatiedostojen perusnimi merkkijonona.
ImageCount	Tallennettujen kuvatiedostojen määrä kokonaislukuna; tämän perusteella määräytyy kuvatiedostojen juokseva mumerointi.
EvolFile	Tyyppiä `TEvolFile` oleva olio, jonka avulla hoidetaan aikakehitystiedoston käsittely.
Data	Tyyppiä `THibboData` oleva olio, tarjoaa `THibboData`-luokan käsittämät ominaisuudet.
SampleColors	Näytteen värikoodaukseen käytettävä väri; tyyppiä `TColor`.
SectionColors	Leikkeiden värikoodaukseen käytettävät värit; tyyppiä `TColorSettings`.
FlowLineColors	Virtaviivojen värikoodaukseen käytettävät värit; tyyppiä `TColorSettings`.
SectionColors	Partikkelien värikoodaukseen käytettävät värit; tyyppiä `TColorSettings`.
IncImageCount	Metodi, joka kasvattaa yhdellä kuvien lukumäärää kuvaavaa juoksevaa numerointia.
jatkuu...	

Taulukko 15: Luokan TProject ominaisuudet.

Ominaisuus (jatkuu)	**Kuvaus**
ReadDataFiles	Metodi, joka antaa `THibboData`-luokalle käskyn lukea datatiedostot annettujen nimien perusteella. Luokka lukee näytetiedoston, jos tulostiedoston nimeä ei ole asetettu, ja muuten vain tulostiedoston. Jos kumpaakaan nimeä ei ole asetettu, aiheutuu poikkeus.
ReadProjectFile	Metodi, joka lukee projektitiedoston asetetun nimen perusteella. Jos nimeä ei ole asetettu, aiheutuu poikkeus. Tiedosto yritetään lukea ajettavan ohjelman kansiossa olevasta kansiosta nimeltä *Projects*, ja alikansiosta, jolla on sama nimi kuin projektilla.
WriteProjectFile	Metodi, joka kirjoittaa projektitiedoston asetetun nimen perusteella. Jos nimeä ei ole asetettu, aiheutuu poikkeus. Tiedosto kirjoitetaan ajettavan ohjelman kansiossa olevaan kansioon nimeltä *Projects*, ja alikansioon, jolla on sama nimi kuin projektilla. Jos näitä kansioita ei ole, ne luodaan.
ReadSampleFile	Metodi, joka lukee näytetiedoston `Data`-olion kautta.
ReadResultFile	Metodi, joka lukee tulostiedoston `Data`-olion kautta. Tulostiedostolla täytyy olla nimi ja simulaation on täytynyt onnistua.
CreateSample	Metodi, joka luo projektiin uuden näytteen. Metodille annetaan parametreina näytteenluontiohjelman parametrit. Palauttaa True, jos näytteenluonti suoritettiin onnistuneesti.
StartSimulation	Metodi, joka aloittaa simulointiohjelman suorittamisen projektiin asetetuilla parametreilla, käynnistää myös aikakehitystiedoston lukemisen. Paluuarvona `True`, mikäli simulointiprosessi alkoi onnistuneesti.
StopSimulation	Metodi, joka lopettaa simuloiniohjelman suorittamisen luomalla tyhjän, tulostiedostojen kanssa samannimisen, mutta *.stop*-päätteisen tiedoston.
SimulationSuccesful	Metodi, joka tarkastaa simuloinnin jälkeen, onnistuiko simulointi. Mikäli *.dat*-päätteinen tiedosto on olemassa, simulointi on onnistunut.

5.6.2 TEvolFile

Tämä luokka lukee simulaatio-ohjelman tuottamaa `.evol`-päätteistä tiedostoa tietyin väliajoin. Väliaika voidaan säätää. Kulloisellakin lukukerralla luetaan tiedoston viimeinen rivi ja tulkitaan siitä permeabiliteetin arvo, tai jos simulaatio on päättynyt, tulkitaan lopputila. Jos simulaatio onnistuu, viimeisellä rivillä lukee 'OK'. Jos taas simulaatio on päättynyt virhetilanteeseen, viimeisellä rivillä lukee 'ERROR: ' sekä selkokielinen virheilmoitus. Luokan ominaisuudet ja tapahtumat on selostettu taulukossa 16.

Taulukko 16: Luokan TEvolFile ominaisuudet.

Ominaisuus	**Kuvaus**
FileName	Luettavan tiedoston nimi merkkijonona.
Delay	Tiedoston lukemisen aikaväli millisekunteina.
GetPermeability	Metodi, joka palauttaa viimeisimmän luetun permeabiliteetin arvon `Double`-tyyppisenä.
GetTimeStep	Metodi, joka palauttaa viimeisimmän luetun aika-askeleen arvon `Integer`-tyyppisenä.
Start	Metodi, joka käynnistää *.evol*-tiedoston lukemisen.
OnBegin	Tapahtuma, jokaa tulee, kun tiedoston lukeminen aloitetaan.
OnRead	Tapahtuma, joka tulee, kun tiedostosta on luettu rivi.
OnError	Tapahtuma, joka tulee, kun tiedoston lukemisessa tapahtui virhe tai simulointi päättyi virhetilanteeseen. Tapahtuman mukana välitetään luettu virheilmoitus.
OnEnd	Tiedoston lukeminen loppui onnistuneesti, simulaatio on päättynyt.

5.6.3 TCorrectingSolver

Tämä luokka periytyy luokasta `TPathSolverAlgorithm` ja se toteuttaa Hibbo-ohjelmassa käytetyn polunratkaisualgoritmin. Luokalla on siis funktio nimeltä *NextPoint*, joka palauttaa partikkelin polun seuraavan pisteen, kun sille annetaan edellinen piste ja otettavan askelen pituus. Seuraavassa selostetaan tässä funktiossa käytettävän algoritmin periaatteet.

Funktio $P(t) : \mathbb{R} \rightarrow \mathbb{R}^3,\ t \in [0, t_0]$ kuvaa reaaliakselin janan poluksi

kolmiulotteiseen avaruuteen. $\vec{V}(x)$ ilmoittaa nopeusvektorin pisteessä x. Polun löytäminen vektorikentässä vaatii differentiaaliyhtälön numeerista ratkaisemista. On ratkaistava alkuarvotehtävä

$$P'(s) = \vec{V}(P(s)),\ P(0) = x_0 \in \mathbb{R}^3.$$

Hibbo-ohjelmassa tämän alkuarvotehtävän ratkaisemiseen käytetään korjaavaa algoritmia, jossa otetaan ensin askel eksplisiittisellä Eulerin algoritmilla ja korjataan sitten lopullista tulosta ottamalla keskiarvo alkupisteen ja loppupisteen nopeudesta ja etsimällä uusi loppupiste tämän nopeuden avulla. Idea on se, että mahdollisesti harhainen suunta, joka laskettiin yhden pisteen perusteella, tarkentuu kun huomioon otetaan nopeus myös lasketussa pisteessä. Olkoon p_0 lähtöpiste ja p_1 seuraava piste, h askelpituus, v_0 nopeus pisteessä p_0 ja v_1 nopeus pisteessä p_1.

1. Otetaan askel eksplisiittisellä algoritmilla:

$$p_1 = p_0 + hv_0.$$

2. Sitten etsitään parempi ratkaisu käyttämällä nopeuksien keskiarvoa pisteissä p_0 ja p_1 :

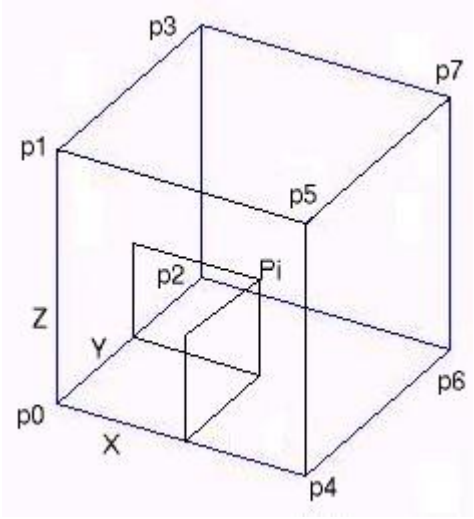
$$p_1 = p_0 + \frac{h}{2}(v_0 + v_1).$$

3. Tätä voidaan toistaa useamman kerran paremman ratkaisun löytämiseksi. Toistoja voidaan tehdä tietty määrä tai kunnes ratkaisu ei enää parane merkittävästi.

5.6.4 TrilinearInterpolation

Hibbo-ohjelman käsittelemä diskreetti data pitää joissakin tilanteissa saada muunnettua jatkuvaksi, ja tähän tarvitaan interpolointia. Hibbo-ohjelmassa käytetään trilineaarista interpolointimenetelmää, joka toteuttaa kappaleessa 5.2 esitellyn interpolointifunktiorajapinnan. Algoritmissa hilakopin suureen arvojen ajatellaan sijoittuvan koppien keskipisteeseen. Interpoloitua arvoa laskettaessa etsitään suureen arvot kahdeksan lähimmän kopin keskipisteissä, ja kuvitellaan etsitty piste näiden kahdeksan kopin keskipisteen muodostaman kuution sisään.

Kuva 6 selventää tilannetta. Pisteet $p_o \ldots p_7$ ovat ympäröivien kuutioiden keskipisteet ja p_i on piste, jolle etsitään interpoloitua arvoa. Kuution sivu on ykkösen pituinen, ja koordinaattiakselit sijoittuvat kuten kuvassa,



Kuva 6: Tilanne interpolointialgoritmissa.

x-akseli oikealle ja z-akseli ylös. Pisteen p_0 koordinaatit olkoot (x_0, y_0, z_0) ja pisteen p_i koordinaatit (x_i, y_i, z_i). Suureen arvot pisteissä $p_0 \ldots p_7$ olkoot $v_0 \ldots v_7$. Interpoloitu arvo v_i lasketaan seuraavasti:

1. Merkitään pisteen p_i koordinaatteja kuution sisällä $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$. Ne lasketaan näin:

$$\begin{aligned} \hat{x}_i &= x_i - x_0, \\ \hat{y}_i &= y_i - y_0, \\ \hat{z}_i &= z_i - z_0. \end{aligned}$$

2. Lasketaan arvo v_i trilineaarisella algoritmilla:

$$\begin{aligned} v_i = {} & v_0 \cdot (1 - \hat{x}) \cdot (1 - \hat{y}) \cdot (1 - \hat{z}) + \\ & v_4 \cdot \hat{x} \cdot (1 - \hat{y}) \cdot (1 - \hat{z}) + \\ & v_2 \cdot (1 - \hat{x}) \cdot \hat{y} \cdot (1 - \hat{z}) + \\ & v_1 \cdot (1 - \hat{x}) \cdot (1 - \hat{y}) \cdot \hat{z} + \\ & v_5 \cdot \hat{x} \cdot (1 - \hat{y}) \cdot \hat{z} + \\ & v_3 \cdot (1 - \hat{x}) \cdot \hat{y} \cdot \hat{z} + \\ & v_6 \cdot \hat{x} \cdot \hat{y} \cdot (1 - \hat{z}) + \\ & v_7 \cdot \hat{x} \cdot \hat{y} \cdot \hat{z}. \end{aligned}$$

5.6.5 Käyttöliittymä

Käyttöliittymän ikkunointi toteutetaan käyttäen Delphin ja Kylixin tarjoamia erinomaisia mahdollisuuksia, eikä yksityiskohtiin puututa tässä. Sen sijaan tässä luvussa selostetaan toiminnot, jotka käyttöliittymästä voidaan käynnistää, miten toiminnot on toteutettu sekä mitkä toimintojen vaikutukset ohjelman tilaan ovat. Jokaisesta toiminnosta on tehty aliohjelma. Käyttöliittymäkomponenttien tapahtumankäsittelijöissä pelkästään kutsutaan näitä aliohjelmia. Taulukossa 17 on esitetty sovelluksessa käytettävät aliohjelmat ja niiden tehtävät.

Taulukko 17: Käyttöliittymän aliohjelmat.

Aliohjelma	**Kuvaus**
DoStartProgram	Huolehtii sovelluksen käynnistyksen yhteydessä suoritettavista toimenpiteistä. Käynnistyksen yhteydessä tarkistetaan, onko simulointi jäänyt edellisellä kerralla käyntiin. Jos näin on, niin sovellus aukaistaan kyseiseen projektiin. Muuten kutsutaan `DoChangeProject`-aliohjelmaa.
DoCloseProgram	Huolehtii sovelluksen sulkemisen yhteydessä suoritettavista toimenpiteistä. Jos simulointi on käynnissä, kysytään jätetäänkö se käyntiin. Jos simulointi jää käyntiin, ohjelman *ini*-tiedostoon tallennetaan tieto siitä, että seuraavan kerran ohjelmaa käynnistettäessä avataan ohjelma tähän kyseiseen projektiin. Projekti tallennetaan `SaveProject`-aliohjelman avulla.
DoChangeProject	Huolehtii projektin vaihtamiseen liittyvistä toimista ja saa parametrina `Boolean`-tyyppiset CreateProject- ja OpenProject-arvot. Parametrien avulla tiedetään, luodaanko uusi vai avataanko vanha projekti. Uuden projektin luomisen yhteydessä kutsutaan `DoCreateProject`-aliohjelmaa, kun taas vanhan projektin avaamisen yhteydessä kutsutaan `DoOpenProject`-aliohjelmaa.
jatkuu...	

Taulukko 17: Käyttöliittymän aliohjelmat.

Aliohjelma (jatkuu)	**Kuvaus**
DoCreateProject	Huolehtii projektin luomiseen liittyvistä toimista ja saa parametrina `String`-tyyppisen Name-arvon, joka on projektille asetettava nimi. Jos sovelluksessa on luomisen yhteydessä jokin projekti auki, se tallennetaan ja suljetaan. Parametrina saadun nimen mukainen hakemisto luodaan projektihakemistoon ja sinne luodaan projektitiedosto. Hakemisto ja tiedosto nimetään projektille annetun nimen mukaisesti. Projektin tila asetetaan tyhjäksi projektiksi.
DoOpenProject	Huolehtii vanhan projektin lukemiseen liittyvistä toimista. Jos projekti on jo olemassa, se tallennetaan ja suljetaan. Myös visualisointinäkymä tyhjennetään. Sitten luodaan uusi projekti asettamalla luomisen yhteydessä `Boolean`-tyyppinen OldProject-parametri todeksi. Tällöin projektin mahdollisten tiedostojen luku tapahtuu `TProject`-luokan konstruktorissa. Myös projektin tila asetetaan.
DoSaveProject	Tallentaa projektiin tehdyt muutokset olemassaolevaan projektitiedostoon.
DoCloseProject	Huolehtii projektin sulkemiseen liittyvistä toimista. Mikäli jokin projekti on auki, suoritetaan SaveProject-aliohjelma ja suljetaan projekti. Koska sovellus menee tilaan, jossa projektia ei ole auki, ja ohjelmassa on aina oltava jokin projekti, kaikki muut toiminnot paitsi ohjelma sulkeminen ja projektin luominen ja avaaminen asetetaan pois käytöstä
DoChangeState	Vaihtaa projektin tilaa. Vaihtoehdot: tyhjä projekti, projektissa on näyte ja näytettä ei ole simuloitu, näytteen simulointi on kesken tai näyte on simuloitu. Tilanvaihdon yhteydessä otetaan huomioon rajoitukset, jotka kyseinen tila aiheuttaa. Kun projektissa on vain simuloimaton näyte tai simulointi on kesken, muut visualisointitoiminnot paitsi näytteen visualisointi ovat poissa käytöstä. Kun simulointi on kesken tai valmis, näytettä ei voi vaihtaa, jolloin näytteenluonti ja lataus ovat poissa käytöstä.
jatkuu...	

Taulukko 17: Käyttöliittymän aliohjelmat.

Aliohjelma (jatkuu)	**Kuvaus**
CheckSampleValues	Tarkastaa, että näytteenluontiohjelman käynnistykseen annetut parametrit on annettu minimi- ja maksimiarvoja noudattaen. Jos kaikki arvot on annettu oikein, palautetaan `Boolean`-tyyppinen `True`-arvo. Jos parametrit on annettu väärin, palautetaan `False`.
DoCreateSample	Käynnistää erillisen näytteenluontiohjelman, joka luo näytteen annettujen parametrien perusteella. Parametrit tarkastetaan kutsumalla `CheckSampleValues`-aliohjelmaa ja näytteenluontiohjelmaa ei käynnistetä, jos parametrit ovat virheellisiä. Kun luonti on valmis, luotu näyte ladataan käyttöön nykyiseen projektiin ja visualisointinäkymä muodostetaan.
DoReadSample	Hoitaa projektin näytetiedoston lukemisen ja näytteen visualisointinäkymän luomisen. Näytetiedosto luetaan kutsumalla `TProject`-luokan `ReadSampleFile`-aliohjelmaa. Visualisointinäkymä muodostetaan luetun tiedoston perusteella.
CheckSimulationValues	Tarkastaa, että simuloinnin käynnistykseen annetut parametrit on annettu minimi- ja maksimiarvoja noudattaen. Jos kaikki arvot on annettu oikein, palautetaan `Boolean`-tyyppinen `True`-arvo. Jos taas parametrit on annettu väärin palautetaan `False`.
DoStartSimulation	Käynnistää erillisen simulointiohjelman annetuilla parametreilla. Simulointiohjelman loppumista ei jäädä odottamaan, vaan prosessi jää pyörimään taustalle. Samalla käynnistetään *.evol*-tiedoston lukeminen, vaihdetaan projektin tilaa ja tuodaan näkyviin permeabiliteetin muutoskuvaaja. Tulevan tulostiedoston nimi tallennetaan. Simulaation parametrit luetaan suoraan lomakkeen syöttökentistä; ennen simuloinnin aloitusta syötetyt arvot tarkistetaan, ja jos ne ovat virheellisiä, simulointia ei aloiteta vaan käyttäjälle annetaan virheilmoitus.
jatkuu...	

Taulukko 17: Käyttöliittymän aliohjelmat.

Aliohjelma (jatkuu)	**Kuvaus**
DoSimulationFinished	Suorittaa tarvittavat toimenpiteet simulaation valmistumisen jälkeen. Jos simulointi onnistui, suoritetaan `DoReadSimulation`-aliohjelma ja asetetaan projektin tila simuloiduksi. Tällöin muutkin visualisointitoiminnot kuin näytteen visualisointi tulevat käyttöön.
DoReadSimulation	Lukee tulostiedostot käyttöön ja ne visualisoidaan.
DoStopSimulation	Keskeyttää simulaation asettamalla simulaatio-ohjelman lopetusehdon todeksi. Aikanaan tulee tieto simulaation loppumisesta, jolloin tulostiedostot luetaan.
DoEvolOnBegin	Asettaa käyttöliittymään viestin "Reading evol file", kun simulointi on käynnistynyt ja *.evol*-tiedoston lukeminen on aloitettu.
DoEvolOnRead	Päivittää käyttöliittymän permeabiliteettikäyrää ja Progressbar:ia, kun *.evol*-tiedostosta on luettu viimeisin rivi.
DoEvolOnError	Suorittaa tarvittavat toimenpiteet, kun simulointi on päättynyt virheeseen. Käyttäjälle näytetään ilmoitus siitä, että simuloinnissa on tapahtunut virhe. Lisäksi kutsutaan `DoSimulationFinished`-aliohjelmaa ja vaihdetaan tila takaisin näyte luotu tilaan.
DoEvolOnEnd	Suorittaa tarvittavat toimenpiteet, kun simulaatio on päättynyt onnistuneesti, eli kutsuu `DoSimulationFinished`-aliohjelmaa.
DoOpenHelp	Avaa sovelluksen käyttöohjeet html-selaimeen.
DoShowAbout	Avaa `About`-dialogin, jossa kerrotaan tarkempaa tietoa sovelluksesta.
SetVisualisationState	Vaihtaa auki olevan projektin visualisaation tilaa. Vaihtoehtoina yleistila, jossa hiiren napeilla on erilaisia tehtäviä sekä pyörittely, siirtely ja zoomaus, joissa käytössä on vain yksi hiiren nappi. Tämä tila vaikuttaa hiiriviestien käsittelyyn.
DoShowSample	Saa parametrina totuusarvon, jonka perusteella se asettaa näytteen näkyvyyden visualisointinäkymässä.
jatkuu...	

Taulukko 17: Käyttöliittymän aliohjelmat.

Aliohjelma (jatkuu)	**Kuvaus**
DoShowSections	Saa parametrina totuusarvon, jonka perusteella se asettaa leikkeen näkyvyyden visualisointinäkymässä.
DoShowFlowLines	Saa parametrina totuusarvon, jonka perusteella se asettaa virtaviivojen näkyvyyden visualisointinäkymässä.
DoShowParticles	Saa parametrina totuusarvon, jonka perusteella se asettaa partikkelien näkyvyyden visualisointinäkymässä.
SetSectionPlane	Saa parametrina `TPlane`-tyypisen arvon ja asettaa sen perusteella leikkaustason. Samalla visualisointinäkymä muuttuu.
SetSectionPosition	Saa parametrina kokonaisluvun, jonka perusteella se asettaa leikkeen sijainnin tasolta halutuksi.
SetSectionVisualisation	Saa parametrina `TVisualisationQuantity`-tyyppisen muuttujan (määritelty edellä `THibboData`-luokan yhteydessä) ja asettaa sen perusteella leikepinnalla visualisoitavan suureen.
SetParticlePlane	Saa parametrina `Plane`-tyyppisen arvon ja asettaa sen perusteella tason, jolta partikkelit ja virtaviivat lähtevät liikkeelle.
SetParticleStartingPosition	Saa parametrina kokonaisluvun, jonka perusteella se asettaa partikkelien ja virtaviivojen lähtötason.
SetParticleAmount	Saa parametrina kokonaisluvun, jonka perusteella se asettaa partikkelien määrän.
SetFlowLineVisualisation	Saa parametrina `TVisualisationQuantity`-tyyppisen muuttujan (määritelty edellä `THibboData`-luokan yhteydessä) ja asettaa sen perusteella virtaviivoissaa visualisoitavan suureen.
SetParticleVisualisation	Saa parametrina `TVisualisationQuantity`-tyyppisen muuttujan (määritelty edellä `THibboData`-luokan yhteydessä) ja asettaa sen perusteella partikkeleissa visualisoitavan suureen.
DoSaveImage	Tallentaa visualisointi-ikkunassa olevan kuvan `Project`-välilehdellä olevan tiedostonimen ja juoksevan numeron mukaisesti.

jatkuu...

Taulukko 17: Käyttöliittymän aliohjelmat.

Aliohjelma (jatkuu)	Kuvaus
DoRotateViewHorizontal	Saa parametrina kulman asteina `Double`-tyyppisenä ja pyörittää kameraa pallopinnalla vaakasuunnassa sen mukaisesti.
DoRotateViewVertical	Saa parametrina kulman asteina `Double`-tyyppisenä ja pyörittää kameraa pallopinnalla pystysuunnassa sen mukaisesti.
DoPanViewHorizontal	Saa parametrina siirrettävän etäisyyden `Double`-tyyppisenä ja siirtää näkymää vaakasuunnassa sen mukaisesti.
DoPanViewVertical	Saa parametrina siirrettävän etäisyyden `Double`-tyyppisenä ja siirtää näkymää pystysuunnassa sen mukaisesti.
DoZoomView	Saa parametrina etäisyyden `Double`-tyyppisenä ja lähentää tai loitontaa näkymää sen perusteella.

5.7 Projektitiedosto

Jokaisesta projektista talletetaan projektikohtaiset tiedot `.project`-päätteiseen projektitiedostoon. Projektitiedoston sisältö käy ilmi alla olevan esimerkin avulla.

```
[Main]
Name=Testi
Sample=.\Projects\kokeilu\koe.sample
Simulation=.\Projects\kokeilu\koe
State=3
[Parameters]
TimeSteps=1000
[Images]
BaseName=Image
Count=0
[SectionColors]
High=255
Low=16711680
Outside=0
[FlowLineColors]
High=255
Low=16711680
```

```
Outside=0
[ParticleColors]
High=255
Low=16711680
Outside=0
[SampleColors]
Sample=16776960
```

5.7.1 Main

Name kertoo projektin nimen.

Sample kertoo projektissa olevan näytetiedoston hakemiston ja nimen.

Simulation kertoo projektissa olevan simuloinnin tulostiedoston perusnimen ja hakemiston.

State kertoo projektin tilan. Projekti voi olla tyhjä, näyte voi olla luotu, simulointi voi olla kesken tai simulointi voi olla valmis.

5.7.2 Parameters

Timesteps kertoo simuloinnissa otettujen aika-askelten määrän.

5.7.3 Images

BaseName kertoo kuvien tallennuksessa käytetyn perusnimen, jonka perään lisätään juokseva numerointi.

Count kertoo, missä kuvan talletukseen käytettävä juokseva numerointi on menossa.

5.7.4 SectionColors

High kertoo leikkeen värikoodauksessa käytetyn värin korkeille arvoille.

Low kertoo leikkeen värikoodauksessa käytettävän värin matalille arvoille.

Outside kertoo ulkopuolella olevan värin.

5.7.5 FlowLineColors

High kertoo virtaviivojen värikoodauksessa käytetyn värin korkeille arvoille.

Low kertoo virtaviivojen värikoodauksessa käytettävän värin matalille arvoille.

Outside kertoo ulkopuolella olevan värin.

5.7.6 ParticleColors

High kertoo partikkelien värikoodauksessa käytetyn värin korkeille arvoille.

Low kertoo partikkelien värikoodauksessa käytettävän värin matalille arvoille.

Outside kertoo ulkopuolella olevan värin.

5.7.7 SampleColors

Sample kertoo näytteen värin.

5.8 Ohjelmointikäytännöt

Ohjelmakoodia kirjoitettaessa noudatettiin itsekommentoivia käytäntöjä. Muuttujille pyrittiin antamaan mahdollisimman kuvaavia nimiä ja myös ohjelmakoodin asemoinnista pyrittiin tekemään selkeä ja luetteva.

Kaikki nimet kirjoitettiin aloittaen jokainen sana isolla kirjaimella ja kirjoittaen sanat yhteen. Tyyppien nimet alkoivat T-kirjaimella. Kommentointi tehtiin englannin kielellä.

Hibbo-ryhmän tuottamien tiedostojen alkuun kirjoitettiin seuraavanlainen nimiö:

```
// Unit:
//
// Project: Hibbo
// Created:
// Creator:
// Description:
//
//
```

```
// Modified:
// 25.4.2003 HK - Fixed bugs in function xx...
//
//
// Hibbo includes the following persons:
// ME - Matti Eskelinen, amjayee@cc.jyu.fi
// OK - Olli Karppinen, ollkarp@cc.jyu.fi
// HK - Harri Kosunen, hmkosune@cc.jyu.fi
// RR - Riikka Rikkola, rerikkol@cc.jyu.fi
//
// Copyright (c) 2003 Hibbo-group. All rights reserved.
//
// ----------------------------------------------
```

Myös kaikille luokille, aliohjelmille ja metodeille kirjoitettiin seuraavanlainen nimiö:

```
// ----------------------------------------------
// Class/Function/Procedure/Method:
// Created:
// Creator:
// Description:
//
// ----------------------------------------------
```

6 Ideoita jatkokehittäjälle

Sovellusprojekteissä käytössä oleva aika on rajallinen, eikä kaikkia mahdollisia hienoja ominaisuuksia pystytty toteuttamaan projektin puitteissa. Tässä luvussa luetellaan joitakin ideoita jatkokehitystä ajatellen, sekä vihjeitä siitä, miten niitä kannattaisi lähteä toteuttamaan.

Etäajo fysiikan laitoksen rinnakkaiskoneella kuului listattuihin vaatimuksiin, mutta se rajattiin pois jo vaatimusmäärittelyssä. Tämä on kohtalaisen helppoa toteuttaa nykyiseen ohjelmaan. `TProject`-luokka huolehtii kaikesta projektin tiedostojen luvusta ja simulointien käynnistämisestä. Tähän luokkaan voitaisiin sijoittaa toiminnallisuuksia verkkoyhteyden kautta tapahtuvaan tiedoston lukuun ja simulaation etäkäynnistykseen.

Uusien visualisointinäkymien kehittäminen on helppoa. Tarvitsee vain toteuttaa uusi luokka, joka peritään luokasta `TVisualisationObject`. Olemassaolevista voi ottaa mallia. Uusien näkymien lisäämistä helpottamaan pitäisi ehkä jotenkin järkeistää oikean reunan paneelia. Eräs ajatus voisi olla, että kaikki saatavilla olevat näkymät olisivat listana, ja listasta valitsemalla pääsisi muuttelemaan kunkin näkymän asetuksia. Samaan tapaan näytettävät näkymät voisi valita listalta.

Näkymien asetusten muuttamisen voisi saada näppärämmäksi. Kullakin näkymällä on asetuksia, joita voidaan muuttaa tietynlaisilla kontrollikomponenteilla. Jos näkymä itse loisi oman asetussettinsä reunapaneelille, uusien näkymien lisääminen helpottuisi huomattavasti.

Väriasetusten tallentaminen voisi olla näppärä lisäominaisuus. Käyttäjä voisi tallentaa värit jollekin nimelle, ja ladata ne käyttöön listasta valitsemalla ja nappia painamalla.

Liikuteltava kamera pelkästään pallopinnalla pyörivän sijaan voisi olla mainio lisä. Se toimisi niin, että hiiren käskyjen mukaan kameran suunta kääntyy tai kamera itse siirtyy. Käyttäjä voisi myös tallentaa muistiin hyviä näkymäpisteitä ja palata niihin. Tähän voitaisiin myös lisätä mahdollisuus animoida kameraa: tämä tapahtuisi niin, että käytetään käyttäjän tallentamia välipisteitä ja siirretään kameraa ja muutetaan sen katselusuuntaa interpoloiden aina jollakin sopivalla tavalla kahden reittipisteen sijainnin ja katselukulman välillä. Bezier-käyrien avulla kameran liikkeistä saataisiin hyvin sulavia.

Videon tallennus olisi hyvä lisä edelliseen sekä partikkelianimaatioon. Se tapahtuisi niin, että piirretään silmukassa animaatioruutuja ja kaapataan joka ruudusta kuva. Kaapatut kuvat voitaisiin sitten skaalata sopivan kokoisiksi ja pakata avi-videoksi vaikka divx:ää käyttäen.

7 Yhteenveto

Tässä dokumentissa kuvattiin Hibbo-nimisen sovelluksen toteutusta. Raportissa on käyty läpi asetettujen vaatimusten toteutuminen, käyttöliittymän toiminnot, sovelluksen rajapinnat ja luokat sekä ohjelmointikäytännöt. Dokumentin tarkoitus on selvittää sovelluksen rakenne seikkaperäisesti mahdollisille jatkokehittäjille.

8 Lähdeluettelo

– [1] Eskelinen Matti, Karppinen Olli, Kosunen Harri ja Rikkola Riikka, "Hibbo-projektin vaatimusmäärittely", Jyväskylän yliopisto, tietotekniikan laitos, saatavilla WWW-muodossa
<URL:http://kotka.it.jyu.fi/hibbo/vaatimusmaarittely/vaatimusmaarittely.pdf>,
viitattu 16.5.2003.