HIBBO

Tietotekniikan sovellusprojekti

Matti Eskelinen
Olli Karppinen
Harri Kosunen
Riikka Rikkola

Sovellussuunnitelma
Versio: 1.0 -7
29.4.2003

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: Sovellussuunnitelma tietotekniikan sovellusprojektiin

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

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

Tiivistelmä

Tämä dokumentti on Jyväskylän yliopistossa keväällä 2003 toteutettavan Hibbo-projektin sovellussuunnitelma. Dokumentissa kuvataan, kuinka sovellus toteutetaan. Lisäksi käydään läpi ohjelmointiin liittyviä käytäntöjä ja sovelluksen sisäistä rakennetta sekä yhteyksiä muihin ohjelmiin ja tiedostoihin.

Avainsanat

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

Dokumentin versiohistoria

Versio	Päivämäärä	Tekijät	Kuvaus
1.0 -1	3.3.2003	HK	Alustava versio
1.0 -2	11.3.2003	OK, RR	Täydennetty ja lisätty kuvia
1.0 -3	18.3.2003	OK, RR	Lisätty sisältöä ja parannettu rakennetta
1.0 -4	28.3.2003	OK, ME, RR	Lisätty sisältöä ja kuvia, viimeistelty
1.0 -5	8.4.2003	OK, RR	Täydennetty ja tarkennettu sisältöä
1.0 -6	15.4.2003	ME, OK, RR, HK	Täydennetty toteutusosiota ja korjailtu muualta
1.0 -7	29.4.2003	ME, RR	Korjattu ja täydennetty toteutusosiota

Tekijöiden lyhenteet

ME Matti Eskelinen

OK Olli Karppinen

HK Harri Kosunen

RR Riikka Rikkola

Sisältö

1 Johdanto

Hibbo on Jyväskylän yliopiston tietotekniikan sovellusprojekti, joka suunnittelee ja toteuttaa Jyväskylän yliopiston fysiikan laitokselle graafisen käyttöliittymän hila-Boltzmann-simulaattoriin. Käyttöliittymän olennainen osa on laskentatulosten visualisointi.

Tämä dokumentti toimii suunnitelmana projektin puitteissa toteutettavalle sovellukselle. Tavoitteena on antaa niin selkeä kuva sovelluksen toiminnasta ja rakenteesta sekä yhteyksistä muihin sovelluksiin ja tiedostoihin, että Delphi-ohjelmointiin perehtynyt lukija pystyisi ohjelmoimaan sovelluksen, vaikka tietenkään suunnitelmassa ei pystytä kuvaamaan aivan kaikkia yksityiskohtia. Tässä dokumentissa ei käsitellä erikseen sovellukselle asetettuja tavoitteita tai käyttötapauksia, sillä ne on listattu vaatimusmäärittelyssä. Myös tiedostojen rakenteet on kuvattu esimerkkien avulla vaatimusmäärittelyn [1] yhteydessä.

Luku 2 sisältää sovellukseen liittyvän termistön. Sovelluksen yhteyksistä muihin sovelluksiin ja tiedostoihin, sekä tiloista ja toiminnoista kerrotaan luvussa 3. Käyttöliittymän ulkoasua ja toimintoja käydään puolestaan läpi luvussa 4. Toteutuksen yksityiskohdat ja ohjelmointiin liittyvät käytännöt käydään läpi luvussa 5.

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.

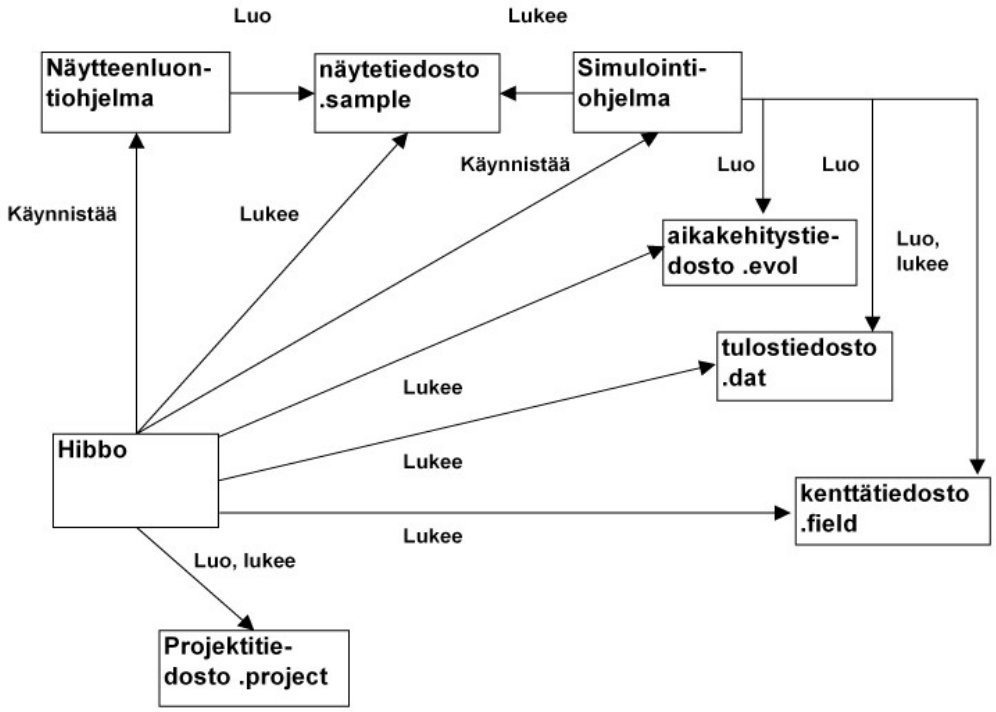
Tulokset ovat simuloinnin tuloksena saatavia tiedostoja, joita on neljä kappaletta: laskentageometrian sisältävä tiedosto, tulostiedosto, aikakehitystiedosto ja tiedosto, joka sisältää hilapisteen tyypin, nestepisteen nopeudet x-, y- ja z-suunnassa sekä paineen.

3 Sovelluksen yhteydet, tilat ja toiminnot

Tässä luvussa kuvataan sovelluksen yleistä toimintaa hahmottelemalla sen toiminnallisia yhteyksiä eri tiedostoihin ja toisiin sovelluksiin. Lisäksi kuvataan ohjelman tilat sekä tilojen väliset siirtymät eri toimintojen kautta.

3.1 Yhteydet tiedostoihin ja sovelluksiin

Hibbo-sovellus on itsenäinen ohjelma, joka hyödyntää kahta Jyväskylän yliopiston fysiikan laitoksella luotua ohjelmaa: hila-Boltzmann-simulaattoria ja näytteenluontiohjelmaa. Kommunikointi kyseisten ohjelmien kanssa on tiedostopohjaista: Hibbo-sovellus lukee simulaattorin tekemiä tulostiedostoja ja näytteenluontiohjelman tekemää näytetiedostoa. Kuva 4 havainnollistaa Hibbo-sovelluksen toiminnalliset yhteydet toisiin sovelluksiin ja tiedostoihin.



Kuva 1: Sovelluksen toiminnalliset yhteydet tiedostoihin ja sovelluksiin.

Hibbo-sovelluksesta voidaan siis käynnistää erillinen näytteenluontiohjelmisto. Tässä yhteydessä käyttäjän syöttämät parametrit välitetään näytteenluontiohjelmistolle, joka luo `.sample`-päätteisen näytetiedoston. Hibbo-

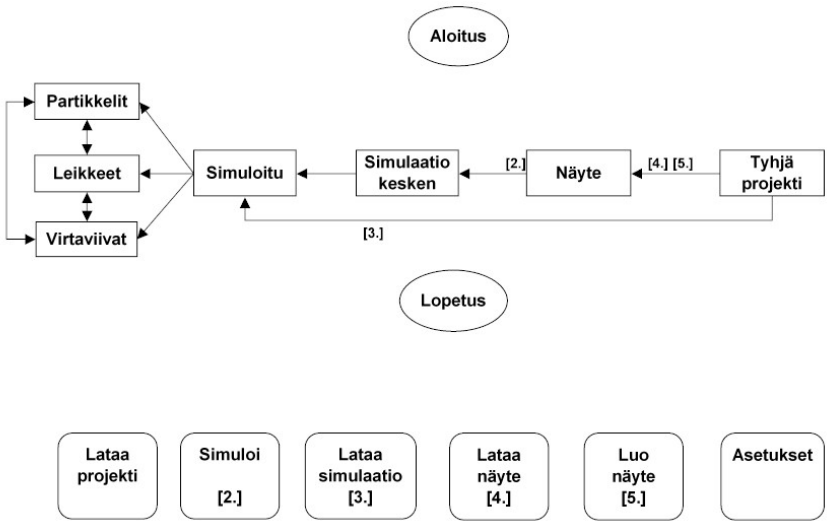
sovellus voi myös lukea valmiin näytetiedoston ja visualisoida luodun tai luetun näytteen.

Hibbo-sovelluksesta voidaan käynnistää myös erillinen simulointiohjelmisto, joka toteuttaa varsinaisen simuloinnin käyttäen `.sample`-päätteistä tiedostoa. Simulointiohjelmiston toteuttaman simuloinnin yhteydessä luodaan tiedostot, joiden tiedostopäätteet ovat: `.evol`, `.dat` ja `.field`. Hibbo-sovellus lukee simuloinnin aikana ėvol-päätteisestä tiedostosta simuloinnin etenemiseen liittyviä tietoja ja esittää ne käyttäjälle.

Hibbo-sovelluksen käytön yhteydessä luodaan `.project`-päätteinen projektitiedosto. Kyseiseen tiedostoon tallennetaan tieto simulointi- ja visualisointitapahtumasta, eli käytännössä tieto käytetyistä näyte- ja tulostiedostoista sekä ohjelman asetuksista.

3.2 Tilat ja toiminnot

Hibbo-sovellus voi olla neljässä eri tilassa riippuen siitä, mitä toimintoja käyttäjä on suorittanut. Sovelluksen toiminnot ovat käyttäjän suorittamia komentoja, jotka annetaan sovellukselle määrätyn asian suorittamiseksi. Kuvasta 2 käy ilmi sovelluksen eri tilat ja toiminnot.



Kuva 2: Kaavio sovelluksen tiloista ja tilasiirtymistä.

3.2.1 Tilat

Tyhjä projekti tarkoittaa tilakaaviossa eräänlaista alkutilaa. Näytettä ei ole vielä luotu, eikä simulaatiota aloitettu.

Näyte tarkoittaa sitä, että laskentageometria on valmis. Tässä vaiheessa tätä simuloimatonta näytettä voi visualisoida.

Simulaatio kesken tarkoittaa sitä, että simulaatio ei ole vielä valmistunut.

Simuloitu tarkoittaa tilakaaviossa sitä, että simulaatio on valmis. Tällöin tuloksia voidaan visualisoida partikkelien, virtaviivojen tai leikkeiden avulla.

3.2.2 Toiminnot

Aloitus tarkoittaa sovelluksen käynnistämistä.

Luo näyte -toiminnossa käyttäjä syöttää laskentageometrian luomiseen tarvittavat parametrit ja käynnistää näytteenluontiohjelman.

Lataa näyte -toiminnossa käyttäjä lataa käyttöön jo valmiina olevan näytetiedoston.

Lataa simulaatio -toiminnossa käyttäjä avaa jo valmiina olevan simulaatiotuloksen, jota voi visualisoida.

Lataa projekti -toiminnossa käyttäjä avaa olemassa olevan projektin. Projekti voi olla tyhjä, projektissa voi olla näyte ladattuna tai projektiin liittyvä simulaatio voi olla kesken tai valmis.

Simuloi -toiminnossa käyttäjä antaa simulaation käynnistämiseen tarvittavat parametrit ja käynnistää simulointiohjelman.

Asetukset -toiminnossa käyttäjä voi muuttaa ja asettaa projektin asetuksia.

Lopetus tarkoittaa sovelluksen lopettamista.

3.2.3 Toimintojen kuvaus

Sovelluksen varsinaisten tilojen väliset tilasiirtymät toteutuvat käyttäjän suorittamien toimintojen kautta.

Mikäli käyttäjä on sulkenut käyttöliittymän keskeyttämättä simulaatiota, avautuu simulaatio valmiina tai keskeneräisenä automaattisesti käyttöliittymän seuraavan kerran avautuessa. Käyttöliittymä voi avautua kuitenkin myös tyhjään projektiin, jos simulointi on lopetettu ennen käyttöliittymästä poistumista. Tällöin näytettä ei ole vielä luotu tai simulaatiota aloitettu.

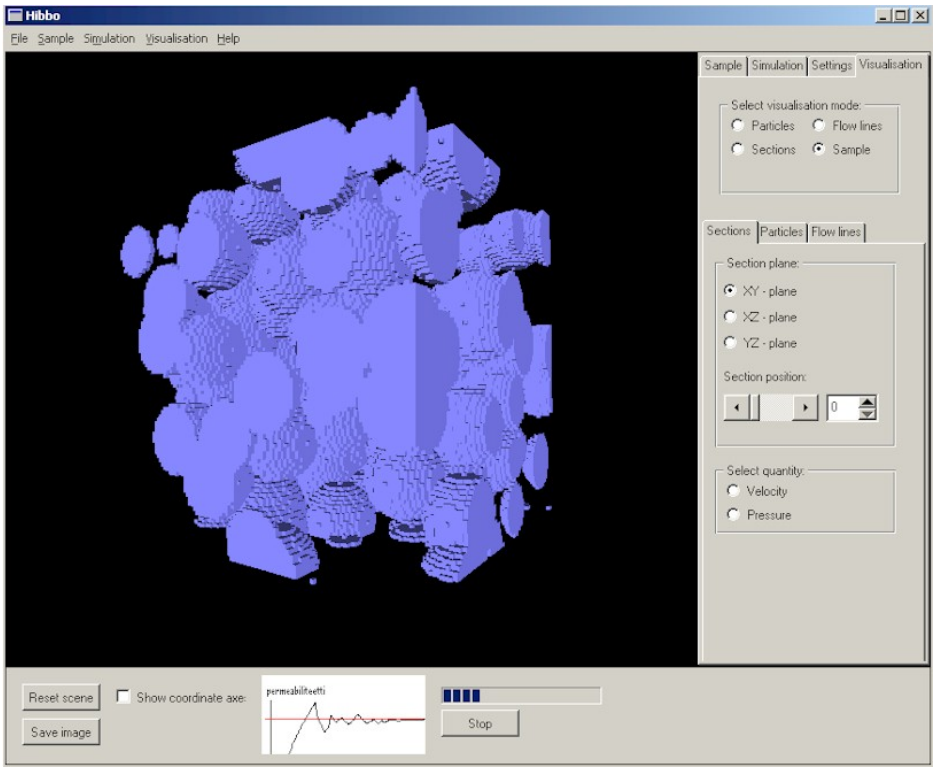
Käyttäjä voi halutessaan luoda uuden näytteen, mutta myös valmiin laskentageometrian lataaminen on mahdollista. Kun laskentageometria on luotu tai ladattu sitä voidaan visualisoida. Käyttäjä voi myös halutessaan ladata valmiin simulaation tai projektin. Valmista simulaatiota voidaan visualisoida saman tien: sitä voidaan tarkastella eri kuvakulmista ja visualisointi voi tapahtua myös partikkelien, virtaviivojen ja leikkeiden avulla. Jos käyttäjä sen sijaan lataa projektin, voi projekti olla missä vaiheessa tahansa: projekti voi olla tyhjä, näyte voi olla luotu tai simulaatio voi olla valmis tai keskeneräinen. Käyttäjä voi muuttaa asetuksia jokaisessa tilassa. Jokaisesta tilasta voi myös ladata projektin.

4 Käyttöliittymä

Tässä luvussa kuvataan käyttöliittymän rakennetta ja toimintaa. Käyttöliittymän toiminnot ja ulkoasu selostetaan yksityiskohtaisesti. Lisäksi kerrotaan kuvien avustuksella käyttöliittymän ulkoasusta erilaisissa käyttötilanteissa.

4.1 Ulkoasu

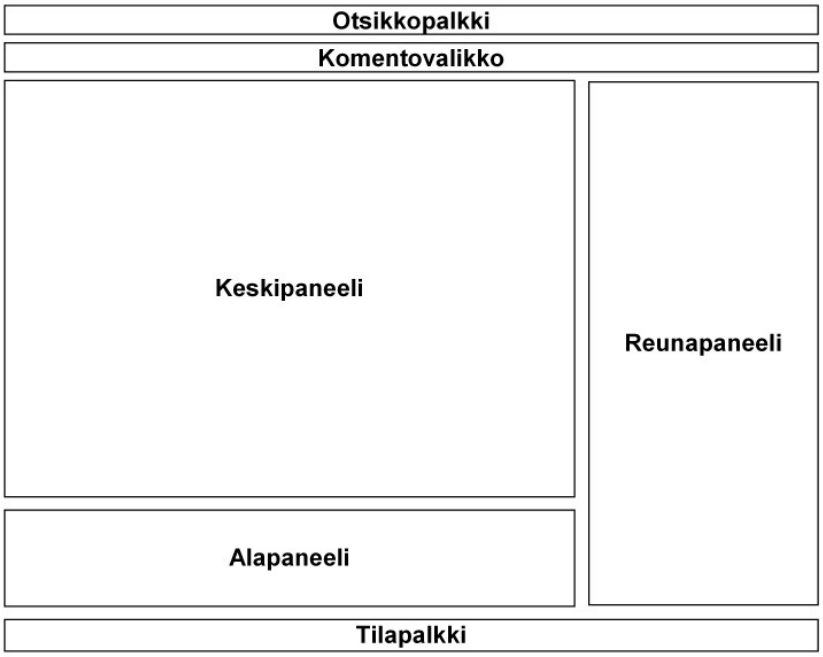
Sovelluksen olennaisin osa on simulointitulosten visualisointi. Tämä näkyy myös käyttöliittymän ulkoasussa, jossa visualisointinäkymällä on suurin rooli. Käyttöliittymässä on myös oma alueensa näytteenluonti- ja simulointiohjelman tarvitsemien parametrien syöttöä varten. Käyttäjä ei siis anna parametreja ilmestyvien dialogien avulla, vaan parametrit syötetään käyttöliittymän oikeassa reunassa oleville välilehdille, joista syötettyjä tietoja voi tarkastella myös jälkikäteen. Kuvassa 3 on hahmotelma sovelluksen ulkoasusta. Kyseessä ei kuitenkaan ole lopullinen näkymä.



Kuva 3: Sovelluksen ulkoasu prototyyppivaiheessa.

4.2 Rakenne

Käyttöliittymän rakenne on jaettu paneelien avulla pienempiin kokonaisuuksiin. Jokaisella paneelilla on oma toiminnallinen tarkoituksensa. Kuvasta 4 näkyy käyttöliittymän rakenne.



Kuva 4: Käyttöliittymän rakenne.

Kuvasta 4 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 käytettävissä olevan projekti- ja näytetiedoston nimi.

4.4 Komentovalikko

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

File

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

Sample

- ☐ New sample file
- ☐ Open sample file

Simulation

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

Visualisation

- ☐ Sample
- ☐ 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. Myös projektikohtaisten asetusten muuttaminen on mahdollista. Lisäksi `File`-valikosta voidaan lopettaa sovelluksen toiminta.

New Project -toiminnon avulla käyttäjä voi luoda uuden 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.

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 halutusta hakemistosta. 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 käyttäjä voi sulkea aktiivisena olevan projektin. Jos projektia ei ole tallennettu, kysytään käyttäjältä dialogin avulla sulkemisen yhteydessä, halutaanko projekti tallentaa.

Settings -toiminto avaa käyttäjälle käyttöliittymän oikean reunan paneelin päällä olevan `Settings`-lehden. Tältä välilehdeltä käyttäjä voi muuttaa projektikohtaisia asetuksia.

Exit -toiminnolla voidaan lopettaa sovelluksen toiminta. Jos tietoja on tallentamatta, käyttäjältä kysytään, haluaako hän tallentaa muutokset.

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.

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.

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 alapaneelissa olevan `Stop`-painikkeen avulla.

4.4.4 Visualisation-valikko

Visualisation-valikon kautta päästään toteuttamaan sovelluksen visualisointitoimintoja.

Sample -toiminnolla saadaan aktiiviseksi `Visualisation`-lehti. Samalla valitaan aktiiviseksi näytetiedoston visualisointi.

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.

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.

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.

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. Hiirikomennoilla on mahdollista halutulla tavalla siirrellä, pyöritellä tai zoomailla visualisointinäkymää. Samat toiminnot on mahdollista

toteuttaa myös määritellyin näppäinkomennoin. Visualisointitoiminnoista, niiden toteutuksesta ja niissä käytettävistä komponenteista, kerrotaan tarkemmin luvussa 5.

4.6 Reunapaneeli

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

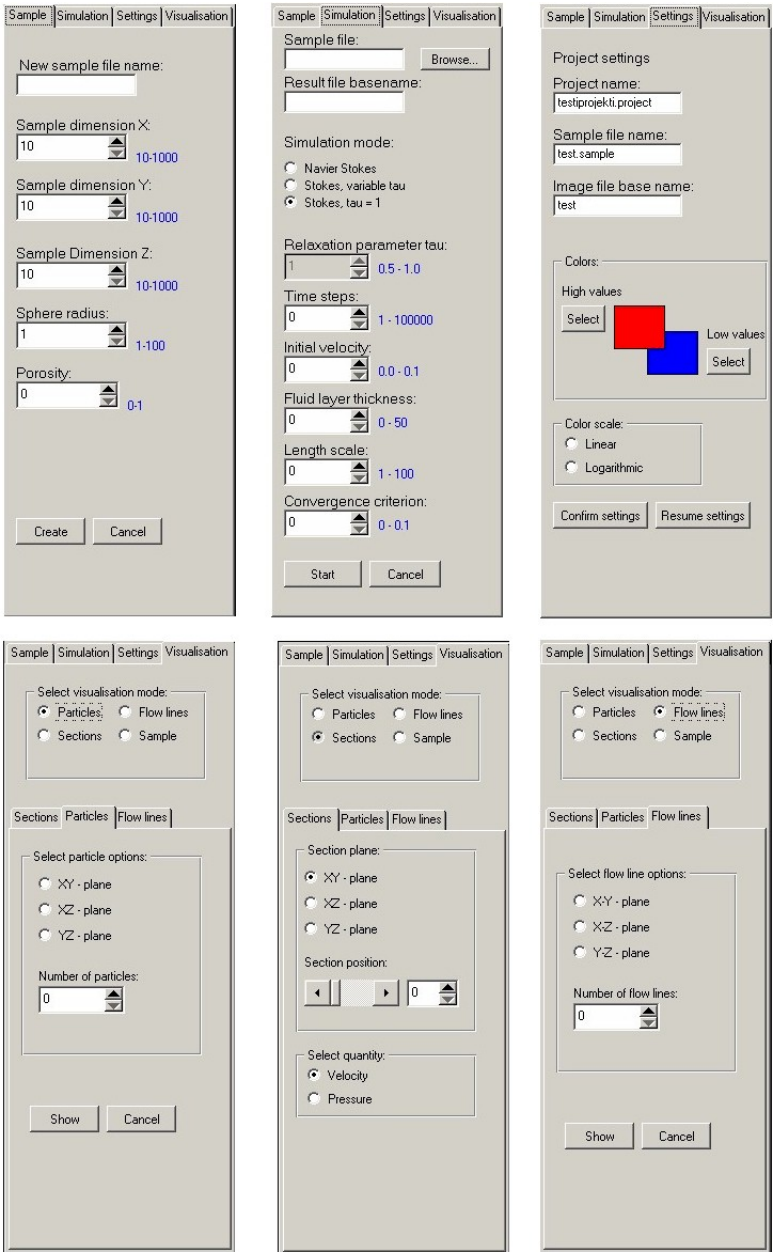
Perustoimintamallina on, että välilehdet on jaoteltu oikeaan paneeliin hierarkisesti. Kerrallaan on auki neljä ylemmän tason lomaketta; `Sample`, `Settings`, `Simulation` ja `Visualisation`. Nämä välilehdet voivat sisältää edelleen alilehtiä, jotka tulevat aktiivisiksi, kun ylemmän tason lehti on valittuna. 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 oikella puolella. Jos syöttökentästä poistutaan ja parametri on syötetty väärin, muuttuu kentän numeerinen arvo väriltään punaiseksi. Mikäli kenttien arvot ovat vääriä vielä silloin, kun näytteenluontiohjelma 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. Kuvassa 5 näkyy käyttöliittymän `Sample`-välilehti.

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

Sample dimension -kenttien avulla syötetään näytteen koko kolmessa ulottuvuudessa, x-, y- ja z-suunnassa.



Kuva 5: Käyttöliittymän välilehdet.

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

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

Create -nappia painamalla käynnistetään näytteenluontiohjelmisto, jolle viedään parametreina `Sample`-lehden kenttiin syötetyt tiedot. Mikäli nappia 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 numeerinen arvo väriltään punaiseksi. Mikäli kenttien arvot ovat vääriä vielä silloin, kun simulaatiota yritetään käynnistää, ilmoitetaan virheellisistä numeerisista arvoista käyttäjälle dialogin avulla. Simulaatioohjelma 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 5 näkyy `Simulation`-lehden ulkoasu.

Sample file -kenttään syötetään näytetiedosto, josta simulaatio halutaan luoda. Näytetiedoston voi syöttää ilman `.sample`-tarkenninta, tai tarkentimen kanssa. Näytetiedoston voi myös valita painamalla kentän vieressä olevaa `Browse`-painiketta. Kyseistä painiketta painamalla aukeaa dialogi, jonka avulla näytetiedosto voidaan valita.

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)`.

Relaxation parameter 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 pystytä kirjoittamaan.

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

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

Fluid layer thickness -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 criterion -kenttään syötetään haluttu lopetusehto. Simulaatio voidaan siis lopettaa jo ennen asetetun aika-askelmäärän täyttymistä, jos nesteen permeabiliteetti vaihtelee tähän kenttään syötetyn arvon määrittelemällä vaihteluvälillä.

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

4.6.3 Visualisation

`Visualisation`-lehden toimintojen avulla voidaan toteuttaa valmiin simulaation tai simuloimattoman näytteen visualisointeja. `Visualisation`-lehti jakaantuu kolmeen alilomakkeeseen: `Flow lines`, `Particles` ja `Sections`. Näiden välilehtien toimintojen avulla käyttäjä voi toteuttaa valmiin simulaation tai simuloimattoman näytteen visualisointeja haluamallaan tavalla.

Flow-lines -lehden toimintojen kautta käyttäjä voi tutkia simuloinnin tuloksia virtaviivojen avulla. Kuvasta 5 näkyy välilehden ulkoasu.

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

Number of flow lines -kohtaan annetaan piirrettävien virtaviivojen lukumäärä.

Particles -lehden toimintojen kautta käyttäjä voi tutkia simuloinnin tuloksia nestepartikkelien avulla. Kuvasta 5 näkyy välilehden ulkoasu.

Plane -osion avulla valitaan taso, jolta nestepartikkelit lasketaan liikkeelle.

Number of particles -kenttään syötetään liikkeelle laskettavien nesteparikkelien määrä.

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. Kuvasta 5 näkyy välilehden ulkoasu.

Plane -osion avulla valitaan taso, jossa suunnassa leikataan.

Select section point -kenttään syötetään tieto leikkauksen kohdasta.

Select quantity -kohdasta valitaan suure, jonka mukaan leike värikoodataan.

4.6.4 Settings

`Settings`-lehdellä käyttäjä voi muokata projektin asetuksia. Tämän lehden toimintojen avulla käyttäjä voi vaikuttaa esimerkiksi visualisoinnissa käytettyihin väreihin. Kuvasta 5 näkyy välilehden ulkoasu.

Image file basename -kenttään syötetään nimi, joka tallennettaville kuville halutaan. Kun kuva tallennetaan alapalkissa olevaa `Save`-painiketta painamalla tulee talletusnimeksi `Setting`-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 `Settings`-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.

Stop -painiketta painamalla simulointi voidaan keskeyttää. Painiketta painettaessa käyttäjälle ilmestyy dialogi, jossa kysytään, halutaanko simulointi varmasti lopettaa. Painamalla dialogin `Yes`-painiketta, simulointi keskeytetään, kun taas `No`-painikkeen painallus peruuttaa simulaation keskeyttämisen.

Kuvaajassa voidaan havainnollistaa halutun muuttujan kehitystä simuloinnin edistyessä. Muuttujana voi olla esimerkiksi permeabiliteetti.

Liukusäätimien avulla voidaan leikata haluttu osa näkymästä. Säätimillä asetetaan minimi- ja maksimiarvot koordinaattiakselien suunnassa.

Toisto/pysäytyspainkkeet ovat käytettävissä partikkelianimaation aikana. Käyttäjä voi halutessaan keskeyttää tai toistaa animaation.

Histogrammi kuvaa visualisoitavan suureen jakaumaa leikepinnalla leikkeen ollessa näkyvissä. Lisäksi samassa yhteydessä voidaan erillisellä säätimellä muuntaa värikoodaus halutunlaiseksi. Muuntaminen tapahtuu liukusäätimellä, jossa palkin vasemmassa laidassa on visualisoitavan suureen pienimpiä arvoja kuvaava väri, ja oikeassa laidassa vastaavasti suurimpia arvoja kuvaava väri. Käyttäjä voi valita haluamansa alkuarvon värikoodaukselle.

Tietokenttiin tulee näytteen ollessa näkyvissä tietoja näytteestä, esimerkiksi huokoisuus.

4.8 Tilapalkki

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

5 Toteutus

Tässä luvussa kerrotaan sovelluksen ohjelmoinnin toteutuksesta. Luvussa selostetaan käytetyt ohjelmointikielet, sovelluksen luokka- ja komponenttijako, eri toimintojen toteuttamisen periaatteet ohjelmointitasolla sekä koodauskäytännöt.

Hibbo-sovellus toteutetaan Object Pascal - ohjelmointikielellä käyttäen Borlandin Delphi- ja Kylix-sovelluksia. Näiden sovelluskehittimien avulla voidaan tehdä koodia, joka voidaan kääntää ohjelmaksi sekä Windows- että Linux-ympäristöön. Hibbo-sovelluksessa käytetään Borlandin CLX-luokkia. Ne pohjautuvat Qt-kirjastoon ja toimivat molemmissa käyttöjärjestelmissä. Windows-versiossa olisi voitu käyttää myös VCL-luokkia, jotka ovat rajapinnoiltaan pääosin samanlaisia mutta toimivat vain Windowsissa. Tämä olisi kuitenkin aiheuttanut hankaluuksia eri käyttöjärjestelmille tarkoitettujen versioiden hallinnassa ja siitä luovuttiin.

Toteutuksessa on käytetty eräänlaista protoiluperiaatetta. Ennen tämän dokumentin kirjoittamista ohjelmasta on jo toteutettu prototyyppiversioita, ja tässä dokumentissa kirjataan ylös toteutusratkaisut, joihin on päädytty. Varsinaisen sovelluksen ohjelmointi aloitetaan tyhjästä, mutta luotuja yleiskäyttöisiä komponentteja kuitenkin käytetään hyväksi.

5.1 Komponenttirajapinnat

Hibbo-ohjelmassa käytetään joitakin yleiskäyttöisiä Matti Eskelisen toteuttamia visualisointikomponentteja. Tässä luvussa kerrotaan sovelluksen toteutuksen kannalta oleelliset asiat näiden komponenttien rajapinnoista.

5.1.1 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 seuraavissa taulukoissa.

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

Tyyppimääritys (jatkuu)	**Kuvaus**
TDescriptives	Tietue, joka sisältää datan tunnuslukuja. Tietueen kenttinä ovat *Min*, *Max* ja *Mean*, jotka ovat kaikki `Double`-tyyppisiä.
EDataController	Poikkeustyyppi, jota `TDataController`-rajapinnan toteuttavat luokat heittävät rajapinnan käsittelyyn liittyvissä virhetilanteissa. Poikkeusluokka on peritty suoraan `Exception`-luokasta, eikä se sisällä mitään uutta.

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

Ominaisuus (jatkuu)	**Kuvaus**
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 kuitenkaa 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.
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.

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.
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.

Toiminto (jatkuu)	Kuvaus
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.
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.

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.
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.

Toiminto (jatkuu)	Kuvaus
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 on skalaarityyppisiä, joten totuusarvokentille toiminta on määrittelemätön. Vektorikentän ollessa kyseessä palautetaan tunnusluvut vektorien normeista. *Parametrit:* näkymäindeksi (kokonaisluku), koordinaatit (taulukko kokonaislukuja). *Paluuarvo:* tunnusluvut `TDescriptives`-tyyppisenä.

5.1.2 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 vaikuta Hibbo-sovelluksen toteuttamiseen.

`VisualisationController`ia on tarkoitus käyttää siten, että ensin luodaan jokin `TDataController`-tyyppinen olio, joka annetaan `VisualisationController`ille. Visualisointiin luodaan näkymä tai useampia. Sen jälkeen näkymät asetetaan paikoilleen ja odotetaan käyttäjän komentoja näkymien muuttamiseksi. Luokan käyttämät tyyppimääritykset sekä sen ominaisuudet ja toiminnot on selostettu seuraavissa taulukoissa.

Tyyppimääritys	Kuvaus
TPlane3d	Määrittää kolmiulotteisen avaruuden tason. Arvoja ovat `pXY`, `pXZ` ja `pYZ`.

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 tämäm ominaisuuden arvoksi.
SectionPlane	Rajoittaa datan tarkastelua johonkin tiettyyn tasoon kolmiulotteisessa avaruudessa. Tällä määrätään leikkeiden muodostustaso; tyyppiä `TPlane3d`. Aina kun tasoa vaihdetaan, leikkeet muodostetaan uudelleen, ja tähän kuluu jonkin verran aikaa, datan koosta riippuen.
ParticlePlane	Osoittaa tason, jolta partikkelit ja virtaviivat lähetetään liikkelle; tyyppiä `TPlane3d`. Aina kun tasoa vaihdetaan, partikkelien polut lasketaan uudelleen, ja tähän kuluu jonkin verran aikaa, datan koosta riippuen.
SectionPosition	Kokonaisluku, joka määrää paikan, jolta leike muodostetaan valitulla tasolla.
FlowLinePosition	Kokonaisluku, joka määrää paikan, jolta virtaviivat ja partikkelit lähetetään liikkeelle valitulla tasolla.
ParticleAmount	Kokonaisluku, joka määrää lähetettävien virtaviivojen ja partikkelien määrän. Tämä määrä partikkeleja jaetaan tasaisesti leikepinnalle.
ViewCount	Kokonaisluku, joka kertoo visualisointinäkymien määrän.
View	Taulukko `TOpenGLPanel`-tyyppisiä paneeleita. Taulukossa on *ViewCount* alkiota.

Toiminto	Kuvaus
CreateView	Luo näkymän visualisointiin. Luo `TCustomControl`-luokasta perityn OpenGL-paneelin ja palauttaa näkymän indeksin. Paneeli on asetettava paikalleen luomisen jälkeen. Näkymiä voidaan luoda useita. Paneeleihin pääsee käsiksi *View*-ominaisuuden ja indeksin avulla. *Parametrit:* näkymän nimi (merkkijono), parent (`TControl`). *Paluuarvo:* näkymän indeksi kokonaislukuna.
RotateView	Pyörittää haluttua näkymää pallopintaa pitkin halutun kulman verran. *Parametrit:* näkymäindeksi (kokonaisluku), vaakasuuntainen kulma, pystysuuntainen kulma (reaalilukuja).
PanView	Siirtää haluttua näkymää halutun etäisyyden verran. *Parametrit:* näkymäindeksi (kokonaisluku), vaakasuuntainen etäisyys, pystysuuntainen etäisyys (reaalilukuja).
ZoomView	Lähentää tai loitontaa näkymää halutun etäisyyden verran. *Parametrit*: näkymäindeksi (kokonaisluku), etäisyys (reaaliluku).

5.2 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 toteutetaan sekä mitkä toimintojen vaikutukset ohjelman tilaan ovat. Ajatus on, että jokaisesta toiminnosta tehdään aliohjelma. Käyttöliittymäkomponenttien tapahtumankäsittelijöissä pelkästään kutsutaan näitä aliohjelmia.

Seuraavassa on esitetty sovelluksessa käytettävät aliohjelmat ja niiden tehtävät.

CreateProject huolehtii projektin luomiseen liittyvistä toimista. Jos projekti on jo olemassa, käyttäjältä varmistetaan, haluaako hän sulkea sen. Sitten käynnistetään dialogi, joka kysyy luotavan projektin nimen. Tämän niminen hakemisto luodaan projektihakemistoon ja sinne luodaan projektitiedosto. Hakemisto ja tiedosto nimetään projektille annetun nimen mukaisesti. Projektin tila asetetaan tyhjäksi projektiksi.

ReadProject huolehtii vanhan projektin lukemiseen liittyvistä toimista.

Jos projekti on jo olemassa, käyttäjältä varmistetaan, haluaako hän sulkea sen. Sitten projektitiedosto luetaan ja asetetaan projektin tila tiedostossa olevien asetusten mukaiseksi. Projektiin kuuluvat näyte- ja tulostiedostot luetaan.

SaveProject tallentaa projektiin tehdyt muutokset olemassaolevaan projektitiedostoon.

Close Project huolehtii projektin sulkemiseen liittyvistä toimista. Käyttäjältä varmistetaan, haluaako hän sulkea projektin. Sitten suoritetaan SaveProject-aliohjelma ja kysytään käyttäjältä, haluaako hän luoda uuden projektin, avata vanhan vai sulkea ohjelman. Jokin projekti on aina oltava käynnissä. Vastauksen perusteella suoritetaan CreateProject- ReadProject- tai CloseProgram-aliohjelma.

CloseProgram sulkee Hibbo-sovelluksen. Jos projekti on auki, käyttäjältä kysytään, haluaako hän tallentaa muutokset; tällöin suoritetaan SaveProject-aliohjelma. 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.

ChangeState vaihtaa projektin tilaa. Vaihtoehdot: tyhjä projekti, projektissa on näyte ja näytettä ei ole simuloitu, näytteen simulointi on kesken tai näyte on simuloitu. Tilanvaihdon yhteydessä pitää ottaa 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.

CreateSample käynnistää erillisen näytteenluontiohjelman, joka luo näytteen annettujen parametrien perusteella. Kun luonti on valmis, luotu näyte ladataan käyttöön nykyiseen projektiin ja visualisointinäkymä muodostetaan. Näytteenluonnin parametrit luetaan suoraan lomakkeen syöttökentistä; ennen luonnin aloitusta syötetyt arvot tarkistetaan, ja jos ne ovat virheellisiä, luontia ei aloiteta vaan käyttäjälle annetaan virheilmoitus.

ReadSample lukee projektiin liittyvän näytetiedoston. Jos1 lukeminen onnistuu, näyte ladataan käyttöön nykyiseen projektiin ja visualisointinäkymä muodostetaan.

ConfirmSettings vahvistaa asetusvalikossa muutetut asetukset ja tallentaa ne Project-luokkaan; samalla niiden oikeellisuus tarkistetaan. Muutokset tulevat voimaan vasta, kun tämä toiminto on suoritettu.

LoadDefaultSettings lataa käyttöön oletusasetukset. Ne ovat tallennettuina ini-tiedostoon.

SaveAsDefaultSettings tallentaa projektin nykyiset asetukset oletusasetuksiksi.

StartSimulation 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.

SimulationFinished suorittaa tarvittavat toimenpiteet simulaation valmistumisen jälkeen. Jos simulointi onnistui, suoritetaan ReadSimulationaliohjelma.

ReadSimulation lukee tulostiedostot käyttöön ja ne visualisoidaan. Samalla projektin tilaa vaihdetaan, jolloin muutkin visualisointivaihtoehdot kuin pelkkä näytteen visualisointi tulevat käyttöön.

StopSimulation keskeyttää simulaation asettamalla simulaatio-ohjelman lopetusehdon todeksi. Aikanaan tulee tieto simulaation loppumisesta, jolloin tulostiedostot luetaan.

OpenHelp avaa sovelluksen käyttöohjeet html-selaimeen.

ShowAbout 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.

ShowSample saa parametrina totuusarvon, jonka perusteella se asettaa näytteen näkyväksi tai näkymättömäksi visualisointinäkymässä.

ShowSections toimii muuten samoin kuin `ShowSample`, mutta näyttää tai piilottaa leikenäkymän.

ShowParticles toimii muuten samoin kuin ShowSample, mutta näyttää tai piilottaa partikkelinäkymän.

ShowFlowLines toimii muuten samoin kuin ShowSample, mutta näyttää tai piilottaa virtaviivanäkymän.

SetSectionPlane saa parametrina TPlane3d-tyypisen arvon ja asettaa sen perusteella leikkaustason. Samalla visualisointinäkymä muuttuu.

SetParticlePlane saa parametrina T3dPlane-tyyppisen arvon ja asettaa sen perusteella tason, jolta partikkelit ja virtaviivat lähtevät liikkeelle. Samalla visualisointinäkymä muuttuu.

SetSectionPosition saa parametrina kokonaisluvun, jonka perusteella se asettaa leikkeen sijainnin tasolta halutuksi.

SetParticleStartingPosition toimii muuten samoin kuin SetSectionPosition, mutta se asettaa partikkelien ja virtaviivojen lähtötason.

SetParticleAmount saa parametrina kokonaisluvun, jonka perusteella se asettaa partikkelien määrän.

SetPlaneVisualisation saa parametrina TVisualisationQuantity-tyyppisen muuttujan (määritelty jäljempänä THibboData-luokan yhteydessä) ja asettaa sen perusteella leikepinnalla visualisoitavan suureen.

SetParticleVisualisation toimii muuten samoin kuin SetPlaneVisualisation mutta asettaa partikkelien ja virtaviivojen visualisoitavan suureen.

SaveImage tallentaa visualisointi-ikkunassa olevan kuvan Settings-välilehdellä olevan tiedostonimen ja juoksevan numeron mukaisesti.

RotateViewHorizontal saa parametrina kulman asteina `Double`-tyyppisenä ja pyörittää kameraa pallopinnalla vaakasuunnassa sen mukaisesti.

RotateViewVertical toimii muuten samoin kuin RotateViewHorizontal, mutta pyörittää pystysuunnassa.

PanViewHorizontal saa parametrina siirrettävän etäisyyden `Double`-tyyppisenä ja siirtää näkymää vaakasuunnassa sen mukaisesti.

PanViewVertical toimii muuten samoin kuin PanViewHorizontal, mutta siirtää näkymää pystysuunnassa.

ZoomView saa parametrina etäisyyden `Double`-tyyppisenä ja lähentää tai loitontaa näkymää sen perusteella.

5.3 Luokkarakenne

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

5.3.1 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 toiminnot on kuvattu seuraavissa taulukoissa.

Tyyppimääritys	**Kuvaus**
TQuantity	Tyyppi, jonka arvoina ovat erilaiset visualisoitavissa olevat suureet. Arvoina `vqPressure`, `vqSpeed`, `vqXVelocity`, `vqYVelocity` ja `vqZVelocity`.

Toiminto	Kuvaus
ReadFile	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).

5.3.2 TProject

Tämä luokka huolehtii projektitiedoston lukemisesta ja kirjoittamisesta sekä ylläpitää projektin asetuksia. `Project`-luokka luo `THibboData`- `TEvolFile`- ja `TVisualisationController`-luokan oliot ja ohjaa niitä. Luokan ominaisuudet ja toiminnot on kuvattu seuraavissa taulukoissa.

Ominaisuus	Kuvaus
Name	Projektin nimi merkkijonona. Tätä nimeä käytetään projektikansion sekä projektitiedoston nimeämiseen.
SampleFile	Käytössä olevan näytetiedoston nimi merkkijonona. Kun tämä nimi asetetaan, se välitetään `THibboData`-luokalle, jotta se osaa lukea näytetiedoston.
SimulationFile	Käytössä olevien tulostiedostojen perusnimi merkkijonona. Kun tämä nimi asetetaan, se välitetään `THibboData`-luokalle, jotta se osaa lukea tulostiedostot.
BaseImageName	Kuvatiedostojen perusnimi merkkijonona.
ImageCount	Tallennettujen kuvatiedostojen määrä kokonaislukuna; tämän perusteella määräytyy kuvatiedostojen juokseva mumerointi.
HighColor	Värikoodauksen ylärajalla käytettävä väri; tyyppiä `TColor`.
LowColor	Värikoodauksen alarajalla käytettävä väri; tyyppiä `TColor`.
OutsideColor	Värikoodauksen ulkopuolinen väri; tyyppiä `TColor`.

Toiminto	Kuvaus
ReadProjectFile	Lukee projektitiedoston asetetun nimen perusteella. Jos nimeä ei ole asetettu, aiheutuu poikkeus.
WriteProjectFile	Kirjoittaa projektitiedoston asetetun nimen perusteella. Jos nimeä ei ole asetettu, aiheutuu poikkeus.
ReadDataFiles	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.

5.3.3 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, toiminnot ja tapahtumat on selostettu seuraavissa taulukoissa.

Ominaisuus	Kuvaus
FileName	Luettavan tiedoston nimi merkkijonona.
Delay	Tiedoston lukemisen aikaväli millisekunteina.

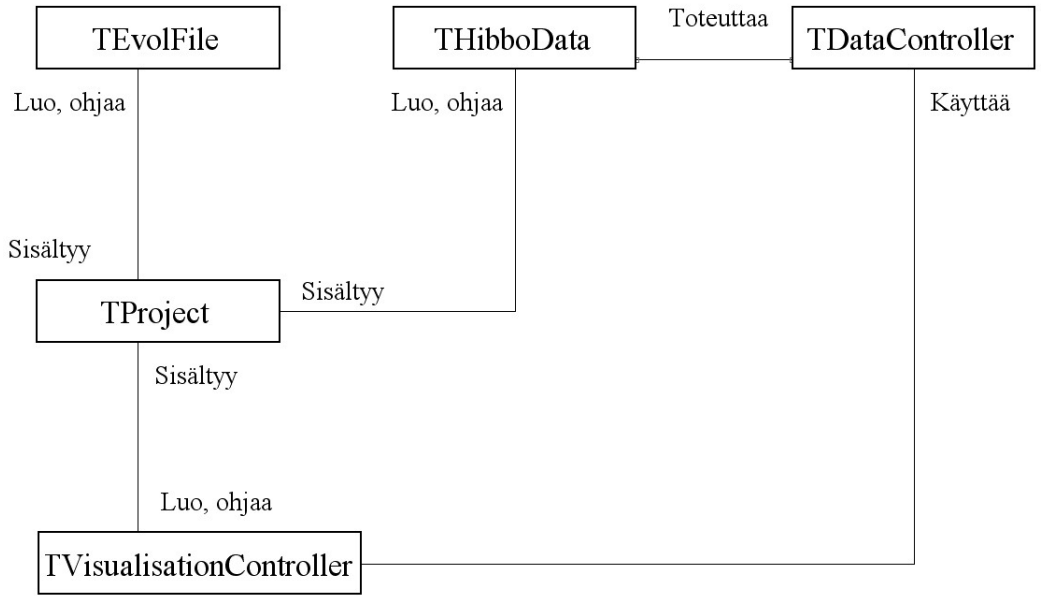
Toiminto	Kuvaus
GetPermeability	Palauttaa viimeisimmän luetun permeabiliteetin arvon `Double`-tyyppisenä.

Tapahtuma	Kuvaus
OnBegin	Tiedoston lukeminen aloitetaan.
OnRead	Tiedostosta on luettu rivi.
OnError	Tiedoston lukemisessa tapahtui virhe tai simulointi päättyi virhetilanteeseen. Tapahtuman mukana välitetään luettu virheilmoitus.
OnEnd	Tiedoston lukeminen loppui onnistuneesti, simulaatio on päättynyt.

5.3.4 Luokkien väliset suhteet

Kuvassa 6 olevassa kaaviossa on esitetty ohjelman sisältämät luokat sekä niiden yhteydet muihin luokkiin. Viivan päissä on kerrottu, mikä on toisen luokan suhde luokkaan, jota lähinnä olevassa päässä teksti on. Käyttöliittymän lomakeluokkia ei ole otettu huomioon kaaviossa.

Luokka `THibboData` toteuttaa `TDataController`-rajapinnan, jota `TVisualisationController` käyttää ja jonka kautta se pyytää dataa visualisaatiota varten. TProject luo luokat `TEvolFile`, `THibbodata` ja `TVisualisationController` ja ohjaa niiden toimintaa. Kaikki kolme luokkaa sisältyvät `TProject`-luokkaan.



Kuva 6: Sovelluksen luokat.

5.4 Visualisointitoiminnot

Hibbo-sovelluksen visualisointitoiminnot voidaan jakaa neljään osaan: näytteen visualisointiin, simuloidun näytteen 2d-leikkeisiin sekä virtausta kuvaaviin merkkipartikkeleihin ja virtaviivoihin. Seuraavassa kerrotaan näiden toimintojen toteutustavat sekä käytettävät algoritmit.

Näytteen visualisointi tarkoittaa simuloimattoman näytteen visualisointia. Näytteestä luodaan kolmiulotteinen kuva siten, että jokaisen kiintoainepisteen kohdalle piirretään kolmiulotteinen pikseli, eli vokseli. Käytännössä vokseli on kuutio. Näytteestä siis muodostuu kasa pieniä kuutioita, ja tätä kasaa voi pyöritellä, siirrellä ja zoomailla. Käyttäjä voi ottaa tarkasteltavaksi myös vain osan näytteestä. Vokselinäkymän piirtämistä optimoidaan siten, että jokaisen kiintoainekopin kohdalla tutkitaan, onko kopin naapurissa kiintoainekoppeja. Vain jos naapuria ei ole, kuution sivu piirretään.

2d-leikkeet ovat halutusta kohdasta näytettä muodostettuja poikkileikkauksia. Poikkileikkauspinnoille värikoodataan haluttu suure. Visualisointi tapahtuu siten, että valitulta pinnalta etsitään visualisoitavan suureen minimi- ja maksimiarvot. Näiden mukaan skaalataan valitut värit käyttäjän toiveiden mukaan joko lineaarisesti tai logaritmisesti ja liitetään tietyille visualisoitavan suureen arvoille tietty väri. Halutessaan käyttäjä voi myös muokata värikoodausta asettamalla käsin minimi- ja maksimiarvot. Sitten halutusta poikkileikkauspinnasta muodostetaan tekstuuri värittämällä pikselit valitun värikoodauksen mukaisesti. Tämä tekstuuri liitetään suorakulmion päälle, joka piirretään poikkileikkauspinnan kohdalle. Muuta näytettä kuvaamaan piirretään suorakulmainen särmiö rautalankamallina.

Virtaviivoilla kuvataan nestehiukkasten liikerataa piirtämällä hiukkasten kulkema reitti polkuviivalla. Käyttäjä valitsee tason, jolta virtaviivat piirretään, sekä virtaviivojen määrän. Valitulle tasolle arvotaan satunnaisesti näin monta pistettä, ja piirretään viivat arvotuista pisteistä alkaen. Jos piste osuu kiintoaineeseen, viivaa ei piirretä. Polut lasketaan valmiiksi muistiin ja malliin piirretään kolmiulotteisia käyriä niiden mukaisesti. Käyrät voidaan värikoodata halutun suureen mukaisesti tai jokaiselle käyrälle voidaan asettaa oma väri.

Laskennassa tarvitaan jatkuva nopeusvektorikenttä. Sen saamiseksi käytetään trilineaarista interpolointia tiedostosta luettuun diskreettiin vekto-

rikenttään. Interpolointi lasketaan siten, että vektori ajatellaan sijoitetuksi hilakopin keskipisteeseen. Jokainen avaruuden piste sijaitsee kahdeksan hilakopin keskipisteen välissä. Näissä kahdeksassa pisteessä tunnetut vektorien arvot interpoloidaan pisteiden välille, jolloin saadaan jatkuva vektorikenttä. Samaa menetelmää voidaan käyttää myös skalaariarvoille.

Virtaviivaa piirrettäessä on ratkaistava numeerisesti differentiaaliyhtälö. Kuvaus $P(t) : \mathbb{R} \rightarrow \mathbb{R}^3$, $t \in [0, t_0]$ kuvaa reaaliakselin janan virtaviivaksi kolmiulotteiseen avaruuteen. On ratkaistava alkuarvotehtävä $P'(t) = \vec{V}(P(t))$, $P(0) = x_0 \in \mathbb{R}^3$. $\vec{V}(x)$ ilmoittaa nopeusvektorin pisteessä x. Tässä tarvitaan edellä mainittua jatkuvaa vektorikenttää. Ratkaisemisessa voi käyttää esimerkiksi tavallista eksplisiittistä Eulerin menetelmää, eli $P(t+1) = P(t) + h\vec{V}(P(t))$, missä h on askelpituus. Ratkaisumenetelmä sijoitetaan erilleen muusta koodista, joten sitä on myöhemmin helppo muuttaa.

Valitun ratkaisumenetelmän avulla lasketaan pisteitä polulta. Jokaista polun pistettä kohti tallennetaan pisteen paikka sekä vauhti ja paine kyseisessä pisteessä. Pisteet tallennetaan listaksi. Polun laskeminen päättyy, kun tullaan ulos näytteestä, joudutaan kiintoaineeseen tai virtaus pysähtyy.

Merkkipartikkeleilla havainnollistetaan kuvitteellisten nestepartikkelien liikettä näytteen sisällä. Partikkelien liikerata on sama kuin virtaviivoilla ja partikkelit värikoodataan samalla lailla kuin viivatkin. Partikkelit lukevat polun pistelistaan tallennettuja nopeuksia ja laskevat niiden avulla, missä polun pisteessä sijaitsevat kullakin ajan hetkellä. Partikkelit siis sijoitetaan aina polun pisteisiin, sillä pisteiden etäisyydet ovat joka tapauksessa hyvin pieniä. Kutakin partikkelia kuvaamaan piirretään pieni ympyrä, jota siirrellään mallissa lasketun liikeradan mukaisesti ottamalla huomioon nopeudet liikeradan eri osissa.

5.5 Ohjelmointikäytännöt

Ohjelmakoodia kirjoitettaessa noudatetaan itsekommentoivia käytäntöjä. Muuttujille annetaan pitkiä ja kuvaavia nimiä ja ohjelmakoodi asemoidaan selkeällä ja luettavalla tavalla. Koodi sisennetään käyttäen kahta välilyöntiä.

Kaikki nimet kirjoitetaan aloittaen jokainen sana isolla kirjaimella ja kirjoittaen sanat yhteen. Tyyppien nimet alkavat T-kirjaimella. Jos nimeämistyyli ei riitä selkiyttämään jonkin koodin osan tarkoitusta, voidaan kirjoittaa kommentteja, jotka alkavat kahdella kauttaviivalla (//) ja jotka kir-

joitetaan aina aivan rivin alkuun. Itsestään selvien kohtien kommentointia vältetään, koska se vain sekoittaa.

Aliohjelmien ja luokkien toiminta pyritään selittämään riittävän tarkasti niiden nimiöissä. Lopullista koodia kirjoitettaessa pyritään kirjoittamaan aliohjelmien ja luokkien kommentit ennen niiden ohjelmoimista. Tämä auttaa mahdollisten loogisten virheiden välttämisessä ja parantaa koodin laatua.

Kommentointi tehdään englannin kielellä. Hibbo-ryhmän tuottamien tiedostojen alkuun kirjoitetaan 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 kirjoitetaan seuraavanlainen nimiö:

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

6 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 11.4.2003.