Aurinkokuntasimulaattori

T-106.1243 Ohjelmoinnin jatkokurssi L1
Projektin loppudokumentti
Aihe 129

Gravitoni – GRaafinen Aurinkokunnan VIsualisaattori TOsi Nätisti Integroiden

Konsta Hölttä
79149S
AUT 4. vk
konsta.holtta@aalto.fi
20.4.2011

1 Yleiskuvaus

Ohjelma simuloi taivaankappaleiden ja rakettien liikkeitä. Ennalta aseteltujen kappaleiden ratoja lasketaan eteenpäin fysiikan lakien mukaisesti, ja radat voidaan tallentaa tiedostoon sekä laskentaa voi tarkastella reaaliajassa graafisella käyttöliittymällä kolmiulotteisessa maailmassa OpenGL:n avulla kuvakulmaa jatkuvasti säädellen. Jo simuloitua aikajanaa voi kelata taaksepäin sekä simulointia voi jatkaa koska tahansa loppupisteestä eteenpäin.

Simulaatio keskittyy gravitaatiovoimiin ja pienet häiriötekijät kuten ilmanvastus ja suhteellisuusteoria jätetään huomiotta. Algoritmi on kuitenkin matemaattisesti tarkka ja varsin todenmukainen, ja helposti laajennettavissa ulkoisten voimien lisäämiseksi. Integrointiin käytetään RK4:ää, mutta integrointimenetelmää on helppo vaihtaa koodista jos siihen tulee tarvetta.

Ohjelma toimii joko täysin itsenäisesti ilman käyttöliittymää vain lokiin tulostaen tai graafisen käyttöliittymän kautta ohjattuna. Kappaleiden ominaisuudet voidaan lukea asetustiedostosta sekä niitä voidaan lisätä käyttöliittymästä jälkeenpäin. Ohjelma ohjeistetaan ajamaan joko tietty aikamäärä eteenpäin, tai kunnes kaksi kappaletta törmää toisiinsa. Mukana on myös raketteja, joita voi laukoa kiertoradoille (tai muualle) tietyllä alkunopeudella. Tämä ominaisuus tekee ohjelmasta pelin, jossa pyritään esim. optimoimaan raketin moottorin käyttämä polttoaine, raketin kulkeman matkan pituus tms.

Olennaisia muutoksia suunnitelmaan nähden ei tullut. Työn toteutettiin haastavana.

2 Käyttöohje

Ohjelman voi käynnistää ajamaan pelkkää simulaatiota komentorivillä tai lisäksi graafista käyttöliittymää. Komentorivimoodissa asetukset luetaan normaalisti, mutta maailma simuloidaan loppuun saakka eikä ohjelma ota muita syötteitä; gui-tilassa taas avataan ikkuna ja maailmaan voi vaikuttaa ja sitä voi tarkastella monin tavoin.

Kaikkein tärkein ohjelman toimintaan vaikuttava asia on sille annettava asetustiedosto, josta tietoa myöhemmin tiedostot-osiossa.

Koska ohjelma käyttää OpenGL:ää (JOGL), se pitää käynnistää oikeilla ympäristömuuttujilla. Ohjelman mukana tulee hakemistot lib32 ja lib64, joissa on natiivit kirjastot 32- ja 64-bittisille linux-järjestelmille, sekä lib-hakemisto, jossa on jarit ajoa varten ja jogl-zippi eclipsen ohjeita varten. Mukana on myös apuskriptit runraw.sh, rungui32.sh ja rungui64.sh, jotka asettavat ympäristömuuttujat oikein ajoa varten; näille voi antaa suoraan asetustiedoston (ja simulointiajan) parametreiksi. Ohjelman voi myös käynnistää Eclipsestä suoraan, jos sille on aseteltu projektin polut oikein.

Projektin voi kääntää Eclipsestä suoraan tai sitten komentoriviltä seuraavasti:

```
javac -sourcepath src -classpath lib/jogl.jar src/gravitoni/Gravitoni.java -d bin
```

2.1 Komentoriviajo

Komentoriviltä ohjelmalle annetaan vain tiedostonimi ja simulaation loppuaika sekunteina. Simulaatiota tapahtuu joko tähän ajanhetkeen saakka tai kunnes asetuksissa määrätty loppuehto täyttyy (esim. törmäys). Esimeriksi seuraava simuloi vuoden verran eteenpäin:

```
java gravitoni.Gravitoni conf/all.conf 31536000
```

Tai apuskripteillä:

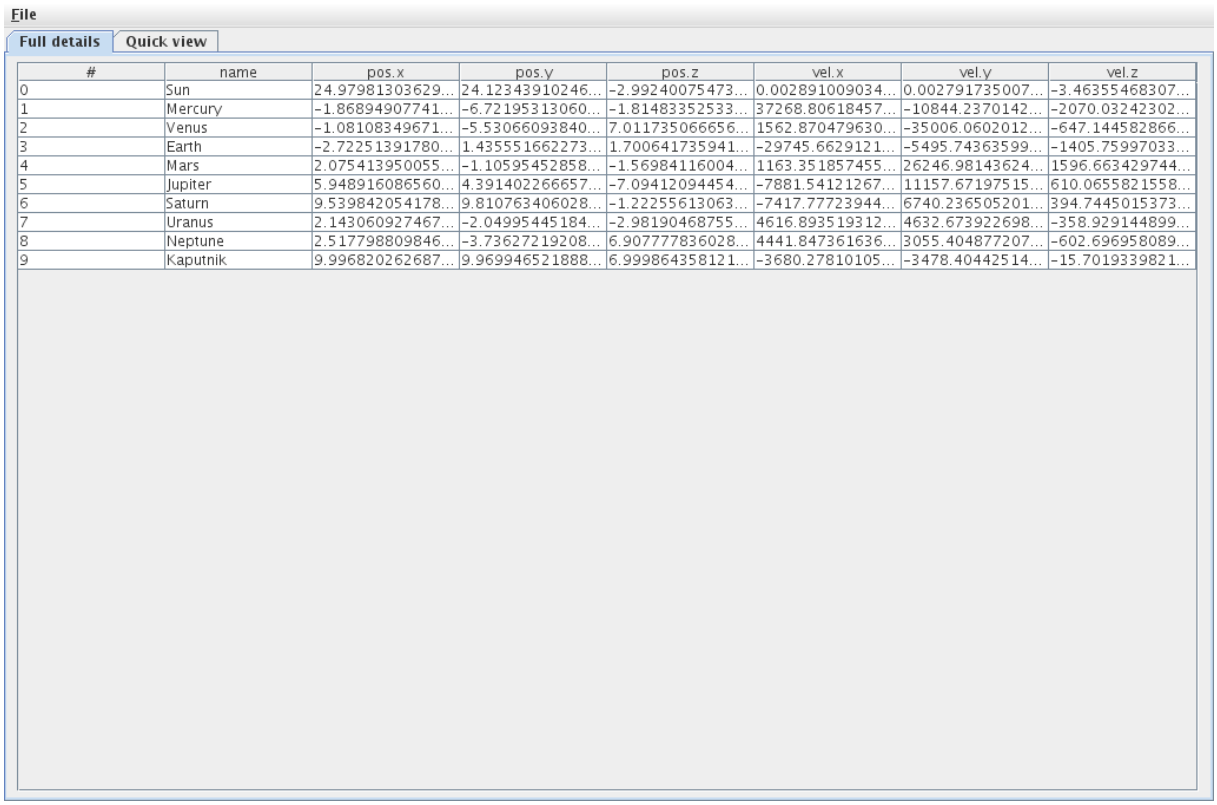
```
./runraw.sh conf/all.conf 31536000
```

Jos runrawilta unohtuu aikamäärä, se yrittää käynnistää guita muttei onnistu siinä ellei classpath ole kunnossa. Tästä lentää poikkeus.

Komentorivin ajosta kiinnostava tuloste on lokitiedosto(t), jotka määrätään asetuksissa. Ohjelma tulostaa hieman debugdataa käynnistyessään ja mm. törmäysten sattuessa. Mukana tulee collisionplot.gnuplot, jonka voi ladata gnuplotiin (komennolla load 'collisionplot.gnuplot') törmäystestiajon ajettuaan (tiedostossa conf/collisiontest.conf).

Suurissa määrin olevia planeettoja voi testailla randomworld.py-skriptin avulla. Se tulostaa komentoriviparametrina annettavan numeron verran kappaleita satunnaisiin paikkoihin. Tämän tulostuksen voi ohjata johonkin asetustiedostoon, joka ladataan itse ohjelmaan.

2.2 GUI



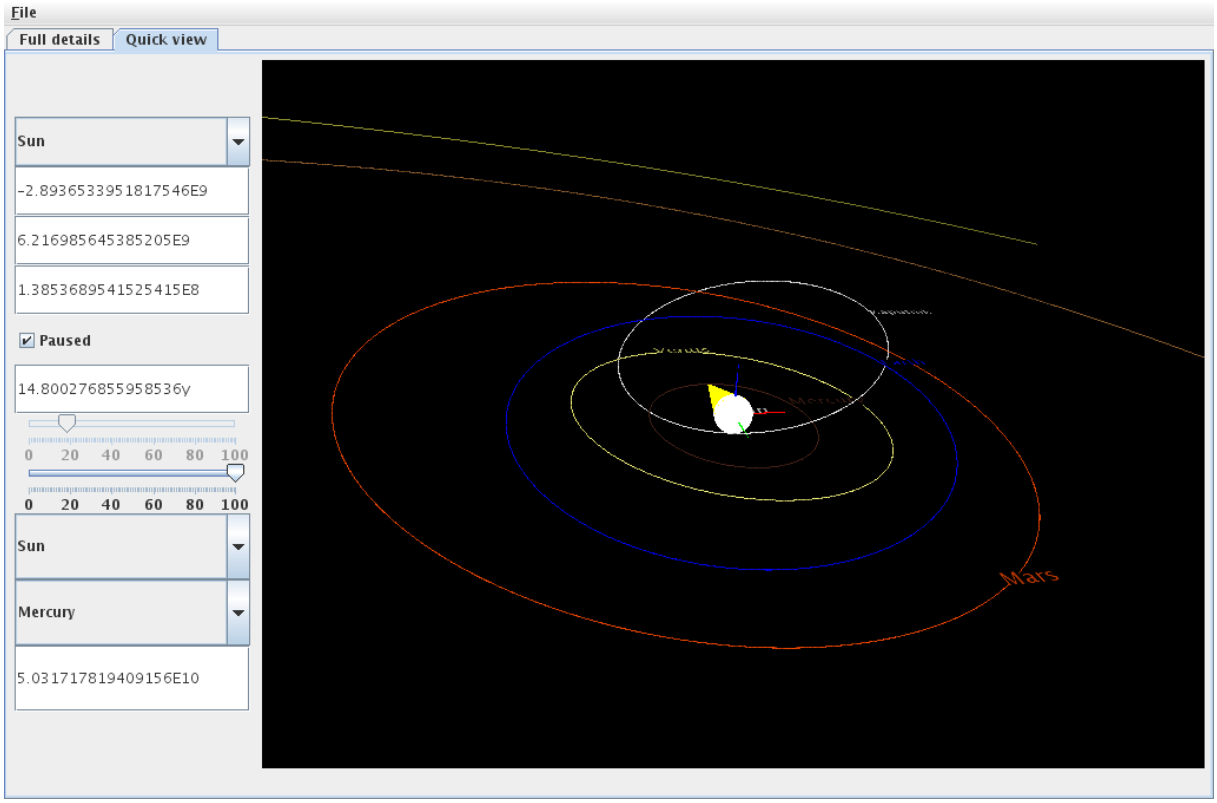
Kuva 1: Numeerinen yksityiskohtatila

Graafisessa näkymässä maailmaa voi tarkastella joustavammin ja siihen voi vaikuttaa kesken kaiken. Näkymässä on kaksi tilaa: numeerinen yksityiskohtainen näkymä (kuva 1 ja testailunäkymä (kuva 2)). Numeronäkymässä maailman tilaa voi katsella excel-tyyliin numeerisesti taulukossa. Testailunäkymässä avaruuden tila näkyy opengl-canvasissa ja joitain nippelitietoja kappaleista ja maailmasta näkee vasemmalla infopaneelissa.

GUI-tila käynnistetään muuten samoin kuin komentorivi, mutta askelmäärä jätetään antamatta.

2.2.1 Numerotila

Numerotila on excelin tyyppinen näkymä, josta näkee nykyhetken tilan laajemmin koko konfiguraatiolle. Eri kappaleiden sijainnit, nopeudet yms. tiedot näkee samanaikaisesti yhdestä taulukosta sekä niitä voi muokata taulukon soluista suoraan, jolloin kappaleiden tilat muuttuvat välittömästi.



Kuva 2: Testailutila, jossa maailman näkee omin silmin ihmiselle mukavassa muodossa

2.2.2 Katselutila

Katselutilan pääpaino on 3d-näkymässä, jota voi raahailla hiirellä eestaas ja josta voi valita planeettoja. Se tottelee seuraavia käyttäjän komentoja:

- Hiiren rulla zoomaa koko näkymää
- Hiiren rulla shift pohjassa zoomaa kappaleiden kokoja
- Hiiren raahaus vasen näppäin pohjassa pyörittää näkymää
- Hiiren raahaus keskinäppäin pohjassa siirtää näkymää
- Välilyöntinappi asettaa valitun kappaleen suhteelliseksi origoksi
- Nappi > valitsee seuraavan kappeleen
- Nappi < valitsee edellisen kappaleen
- Nappi r resetoi näkymän
- Nappi p togglaa pausen
- Nappi c togglaa origokursorin näkymisen
- Nappi e togglaa debuggaustason näkymisen

Lisäksi 3d-tilasta oikealla hiiren napilla klikkaamalla saa auki valikon, josta voi valita samoja asioita kuin infopaneelista ja näppäimistöltä.

Infopaneelista voi seurata/säätää seuraavia asioita:

- Nykyisen planeetan sijainti ja nopeus

- Simulaation nopeus ja aika (ylempi slider on nopeus, alempi aikaprosentti)
- Kahden valitun planeetan välinen etäisyys

Jos simulaatio on pysäytettynä (pause-tilassa), sen nopeutta ei voi säätää, mutta ajanhetkeä kylläkin. Ajanhetki vaikuttaa siihen, mihin simuloinnin indeksiin asti ruutua piirretään. Jatkuvasti simuloivassa tilassa taasen nopeutta voi säätää, ja aika on koko ajan simuloinnin loppuhetkessä.

Nopeussliderin keskikohta 50 vastaa yhtä asetustiedostossa asetettua aika-askelta. Tätä pienemmät ja suuremmat simuloivat nopeammin tai hitaammin eksponentiaalisesti.

Jos planeetat törmäävät ja asetuksissa on säädetty tälle maailmanloppu, ohjelma menee pysähdystilaan eikä sitä voi enää jatkaa tästä pisteestä.

2.2.3 Valikko

Valikosta voi sulkea ohjelman, säätää pausetilaa sekä lisätä ja planeettoja. Tiedoston avaaminen pysäyttää simulaation ja sulkee nykyisen maailmakonfiguraation ja avaa uuden kuin ohjelma olisi käynnistetty tällä uudella tiedostolla. Ohjelmasta poistuminen myös sulkee konfiguraation välittömästi. Planeettojen lisääminen onnistuu vain numerotilassa turvallisuuden vuoksi, koska ohjelman kannattaa olla pausetilassa kappaletta muokatessa mihin se meneekin itsestään numerotilaan siirtyessä.

3 Ohjelman rakenne

Rakenne (kuvassa 3) säilyi pääpiirteittäin samanlaisena kun mitä suunniteltiin; jotkut yksityiskohdat vähän elivät eri suuntiin. Koodin rakenne hajautuu neljään pääkategoriaan:

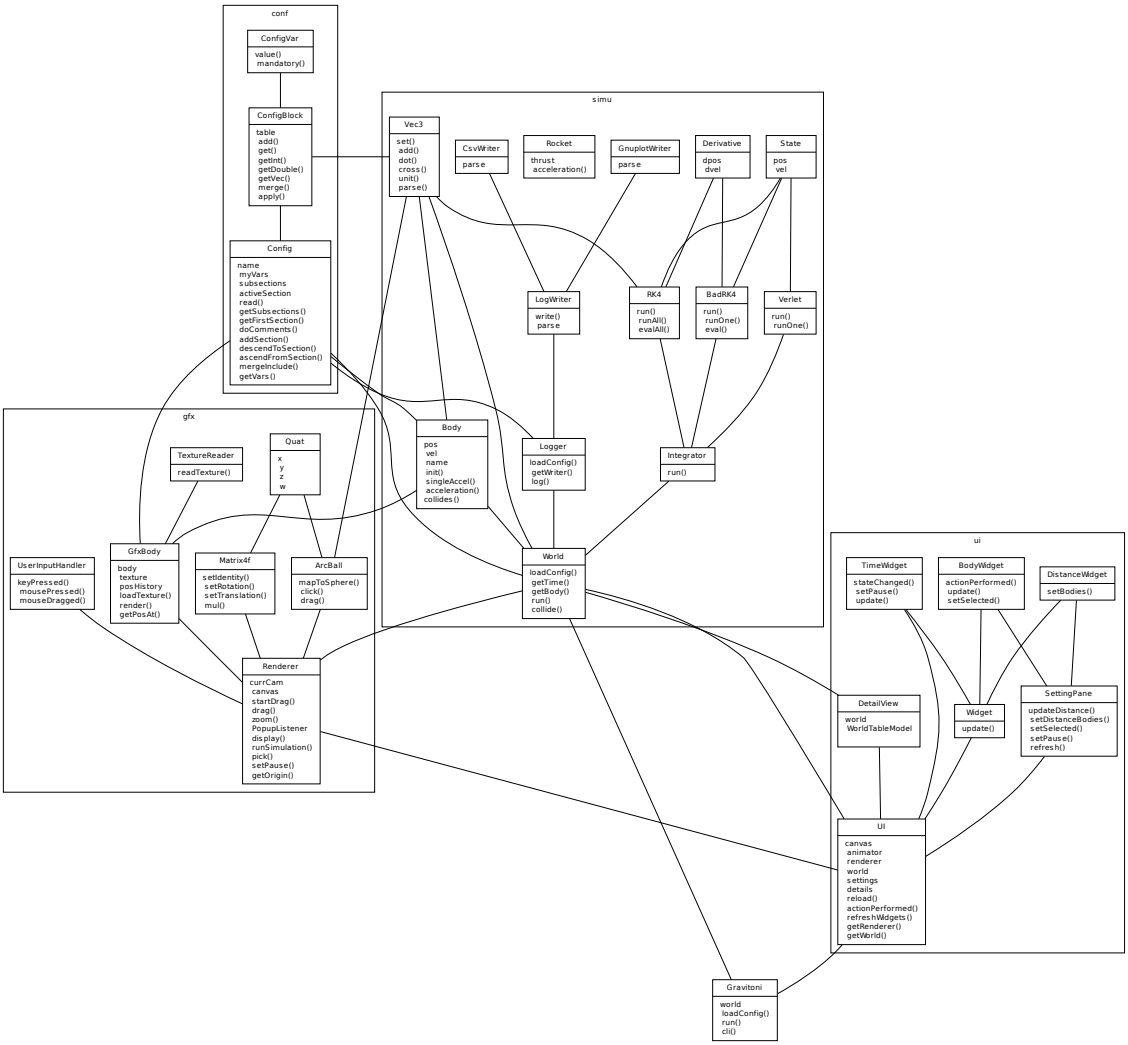
- Konfiguraatiotiedostojen hallinta: geneerinen konffijärjestelmä, joka ei tiedä mihin sitä käytetään.
- Simulaatio integrointeineen: kappaleet jotka sisältävät mm. paikan, nopeuden ja massan; maailma, joka sisältää kappaleet, sekä oheisluokkia mm. integrointiin ja lokittamiseen.
- Käyttöliittymä ja dialogit: käli on pääasiassa yksi ikkuna, jossa on modulaarisesti eri asetuspaneeleita sekä 3d-canvas erillisinä luokkina.
- Graffarenderöinti ja sen käyttöliittymä: renderöijä kyselee kappaleiden tiedot maailmalta ja piirtää ne näytölle, sekä käsittelee mm. näppäimistösyötteet joilla näkymässä voi liikkua.

Javadoc-html löytyy hakemistosta doc/api.

3.1 Konfiguraatiot

Tämä on ohjelman kokonaisuuden kannalta varsin pieni osa (kuten oli suunnitelmassakin), mutta ei sovellu sinänsä suoraan mihinkään kolmesta muusta kategoriasta, ja on siinä määrin geneerinen että tämän erikseen erottelu on vain kaunista. Seuraavassa esitellään itse luokkien toiminta; katso tiedostoformaatin tarkka muoto kohdasta Tiedostot.

Config-luokka vastaa yhtä asetusryhmää, jolla on jokin tyyppinimi (esim. body) ja joka on konffitiedostossa aaltosulkujen sisällä. Asetusryhmän sisällä voi olla asetuksia (nimi/arvo-pari; esim. kappaleen nimi tai sijainti) sekä muita asetusryhmiä. Tämä näkyy siten, että sillä on jäseninään yksi ConfigBlock sekä lista toisia Configeja (taulukkomapissa, koska ryhmiä haetaan nimen perusteella aina). Configin voi yhdistää toiseen merge-metodilla, jolloin yhdistettävän sisältämän ConfigBlockin ja Config-mapin asiat liittyvät yhdistäjään.



Kuva 3: Toteutunut luokkajako ja joitain tärkeimpiä metodeita. Kaavio ei ole UML:ää eikä mitään muutakaan standardia muotoa. Viivat kuvaavat tärkeimpiä interaktioita luokkien välillä.

ConfigBlock on yksi asetusläjälohko, joka sisältää yleensä yhdelle asialle (esim. Body) kuuluvat ominaisuudet. Se sisältää ne nimi/arvo-parit, joita voi hakea mm. funktiolla get (palauttaa Stringin) tai getDouble (yrittää parsia arvon doubleksi ja palauttaa sen). Sen taianomainen apply-metodi toimii annotaatioilla ja sillä voidaan lukea lohkon arvot suoraan johonkin olioon, jonka jäsenmuuttjat on sidottu annotaatioiden avulla konfiguraatiomuuttujien nimiin.

3.2 Simulaatio

Grafiikka-agnostinen simulaatio-osio on projektin matemaattisin osa ja se osa missä itse simulogiikka tapahtuu lokittamisen kera. Komentorivitilassa softa käyttää ainoastaan konffitiedostoja ja simulaatiota.

Vektoriluokka Vec3 kuvaa kaikkia kolmiulotteisia matemaattisia vektoreita. Näitä ovat mm. kappaleiden sijainnit, nopeudet ja kiihtyvyydet. Vektoria hyväksikäytetään myös mm. grafiikkakoodissa rgb-värien varastointiin. Vec3 sisältää vektorien perusominaisuudet kuten toiseen vektoriin summaamisen sekä piste- ja ristitulon ja lisäksi yksinkertaisia apumetodeita kuten sijoittamisen toisesta vektorista tai olion kloonausmetodin. Metodit muokkaavat itse oliota ja palauttavat

referenssin itseensä, jolloin metodikutsujen ketjutus onnistuu näppärästi.

Kappaleen perusluokka on Body joka varastoi yhden kappaleen kaikki tiedot. Bodyllä on myös aliluokka Kepler johon voidaan lukea asetuksista keplerkoordinaatit, jolloin sillä muunnetaan ne karteesiseen koordinaatistoon. Kappaleen päätarkoitus on erotella eri lentävien objektien ominaisuudet eri olioihin. Kappaleella ei ihmeempää toiminnallisuutta ole; tärkeimpänä kuitenkin mainittakoon collides()-metodi jolla tarkistetaan törmäys muihin vertaamalla etäisyyttä.

Tavallisille planeetoille riittää Body-luokka. Rakettiluokka Rocket periytyy Bodystä, ja sillä on kiihtyvyysfunktiossa lisäksi johonkin aikaan loppuva vakiovoima joka kuvaa polttoaineen työntövoimaa. Tätä voitaisiin jatkokehittää siten, että voimaa voisi ohjata tarkemmin käyttäjän syötteen perusteella.

Varsinainen maailmankaikkeuden toiminnallisuus on World-luokassa. Se varastoi listassa Body-oliot ja hoitaa interaktion simulaatiossa muiden luokkien välillä ja integroi pyydettäessä. Metodi loadConfig ottaa konfiguraatio-osion, josta haetaan Bodyjä vastaavat lohkot joilla instantioidaan Body-olioita. Metodi run ajaa integraattoria yhden parametrina tulevan aika-askeleen määrän verran. Integraattorin käyttämä metodi acceleration palauttaa tietyn hetken kiihtyvyyden annetulle planeetalle. Runissa myös tarkistetaan törmäykset collide-metodilla integraattorin ajon jälkeen, ja tehdään toimenpiteitä asetustiedostojen mukaan, kuten ilmaistaan maailmanloppu tai törmäytetään planeettoja.

Worldilla on Integrator-rajapinnan toteuttava jäsenolio joka hoitelee itse algoritmin. Se kyselee Worldilta kiihtyvyyksiä tiettyinä ajanhetkinä eri kappaleille, ja siirtää kappaleita niiden mukaan. RK4 on oletuksena toimiva integraattori, mutta esimerkin vuoksi on tehty myös Verlet, sekä huonompi RK4 joka pitää muuta maailmaa paikallaan laskiessa yhdelle kappaleelle kiihtyvyyttä ajanhetkellä t+dt uudessa paikassa (luokka BadRK4). Integratorin selventämiseksi simussa on myös luokat State ja Derivative, jotka säilövät sijainteja ja kiihtyvyyksiä ilman erityisempää toiminnallisuutta.

Integrointi tulee siis hoitaa koko ”maailmavektorille” kerrallaan: RK4 ja Verlet laskevat kaikille kappaleille omat uudet sijannit, joita käyttävät askeleen välilaskuissa antamalla kaikki sijainnit taulukossa kiihtyvyysfunktiolle, joka käyttää niitä maailman nykytilan sijaan.

World osaa myös lokittaa ajon yhteydessä. Logger-luokka hoitaa kaiken lokittamisen kokonaisuudessaan, ja maailma kutsuu log()-metodia tarvittaessa. Loggerin sisällä on LogWriteristä periytyviä olioita kuten GnuplotWriter ja CsvWriter. Nämä eri tyyppiset lokikirjoittimet kirjoittavat eri tyyppistä dataa tiedostoon nimensä mukaan.

3.3 Käyttöliittymä

Kuten käyttöohjeessa mainittiin, käyttöliittymä on kaksiosainen. Pääikkunaa kuvaa UI-luokka; se hoitaa ikkunan korkeimman tason asiat ja komentaa eri näkymiä, testailu- ja numerotilaa. Näiden kahden näkymän välillä siirrytään välilehtipaneelilla. Asetushässäkkä on taulukkoinputti joka hoitelee Bodyjen välillä interaktiota AbstractTableModelista periytyvällä luokalla.

3D-näkymän asiat hoidetaan grafiikan puolella GLCanvasia kuuntelevassa GLEventListeneristä periytyvässä Rendererissä, joka on UI:n sisällä myös. Tässä näkymässä on myös SettingPane, joka sisältää Widgettejä.

BodyWidget: kappaleen tiedot. Koostuu valikosta, josta valitaan käsiteltävä kappale, sekä muutamasta tekstilaatikosta joista näkee mm. kappaleen paikan. Käsiteltävän kappaleen voi klikata myös 3d-näkymästä, jolloin laatikon valittu elementti vaihtuu samalla.

TimeWidget: simulaation ajan ja nopeuden säätö. Sisältää pausenapin sekä säätöpalkit ajanhetkelle ja simulaation piirtämisen nopeudelle. Ajanhetki säätyy eksponentiaalisesti ja ajanhetki lineaarisesti.

DistanceWidget: etäisyyden laskenta. Valitaan kaksi kappaletta kuten BodyWidgetistä, ja näiden välinen etäisyys näkyy tekstilaatikossa. Kappaleet tähänkin voi valita 3d-näytöltä.

SettingPane hoitaa widgettien puuhat eteenpäin tylsillä proxymetodeillaan. Käyttöliittymäjutut ovat tylsiä ja hoidetaan Swingillä ihan normaalisti.

3.4 Grafiikat

Grafiikkojen renderöinti keskittyy GLCanvas-luokasta periytyvään Renderer-luokkaan, joka hoitaa kappaleiden piirtelyn. Renderer sisältää GfxBody-olioita, joiden piirtometodeita (render()) kutsuu tarvittaessa. Renderer rekisteröi muutamia syötteenkäsittelijöitä UserInputHandler-luokkaan, joka hoitaa hiiren ja näppiksen interaktion. Näkymän katselusuunta muuttuu maailmaa pyörittäessä, zoomatessa ja siirtäessä. Nämä hoituvat erityisen kameramatriisin avulla, jota tuo handler pyytää päivittämään kun käyttäjä niin toivoo. Kameramatriisi annetaan OpenGL:lle suoraan kertomaan kappaleiden koordinaatteja.

UserInputHandler toteuttaa erinäisiä tapahtumakuuntelijainterfaceja. Kun jotain näppäintä painetaan, se tekee jotain tarvittaessa. Hiiren napin mennessä pohjaan alkaa pyöritys- tai pannausoperaatio, ja kun hiirtä raahataan, handleri pyytää Rendereriä päivittämään näkymän kameraa muuttuneiden parametrien suhteen.

Koska maailman pyörittelyoperaatio vaatii hieman erikoista toimintalogiikkaa, se on ulkoistettu ArcBall-luokalle (joka on pienin muutoksin kopioitu raa'asti NeHe:n esimerkeistä). Kun pyöritys alkaa, ArcBallille kerrotaan hiiren alkusijainti. ArcBall projisoi tämän pisteen tietyn kuvitellun pallon pinnalle ja ottaa pintapisteen muistiin. Kun hiirtä raahataan, pyörittely lasketaan kvaternioilla uuden hiiripisteen projisoituun paikkaan. Maailman pyörittelyn avuksi grafiikkapuolella on luokat Quat (kvaternio) ja Matrix4f (4x4 matriisi). Kuvitellun pallon keskipisteestä ajatellaan vektorit klikattuihin pisteisiin pallon pinnalla, ja lasketaan, minkä akselin suhteen ja kuinka paljon tässä palloa pitäisi pyörittää. Tällainen arcball-pyöritys toimii samaan tyyliin kuin mm. Blender-ohjelmassa.

GfxBody on erillinen luokka, joka sisältää viittauksen Bodyyn jota se esittää. GfxBody ei periydy Bodystä, koska nämä ovat kuitenkin varsin eri asiat; piirrettävä objekti on tavallaan kääre Bodylle, eikä mikään Body erikoistoiminnoilla. Tämä myös selventää ohjelman toimintaa muualta, kun ei tarvitse tarkistaa, annetaanko simulaattorille listaa GfxBodyistä UI:n tapauksessa tai Bodyistä CLI:n tapauksessa. Periyttäminen ei itseasiassa olisi edes mahdollista, koska Bodyjen tilalla voi olla Rocketejakin. GfxBody lukee asetuksista lisämääreitä kappaleelle: toistaiseksi värin ja tekstuurin. Tekstuuria käytetään kappaleen piirtämiseen ja väriä liikehistoriakäppyrään; nämä selkeyttävät kappaleiden erottamista toisistaan. GfxBodyn päätehtävä on piirtää kappale liikehistorioineen, jonka se myös varastoi jotta historian takaisinkelaus voisi olla mahdollista.

TextureReader on myös muualta lainattua koodia, ja sen tehtävä on eristää tekstuuritiedostojen lukeminen tiedostoista opengl-luvuiksi muualle. Sitä on hieman yksinkertaistettu ja siihen on lisätty cachaus koska kymmenien kappaleiden tekstuurien lataus hidastaa ohjelman käynnistymistä ikävästi.

4 Algoritmit

4.1 Simulaatio

4.1.1 Integrointi

Pääalgoritmi on neljännen asteen Runge-Kutta-menetelmä (RK4). Kappaleiden tilat y ja aika t muuttuvat seuraavasti (alaindeksit y:lle ja t:lle kuvaavat integrointikierrosten indeksejä):

$$y_{n+1} = y_n + \tfrac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right) \quad (1)$$

$$t_{n+1} = t_n + h \quad (2)$$

Arvot k_i kertovat tietoja kulmakertoimesta ja määräytyvät seuraavasti:

$$k_1 = f(t_n, y_n) \tag{3}$$
$$k_2 = f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_1) \tag{4}$$
$$k_3 = f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_2) \tag{5}$$
$$k_4 = f(t_n + h, y_n + hk_3) \tag{6}$$

k_1 on kulmakerroin nykyhetkellä; k_2 on Eulerin menetelmällä laskettu puolikkaan aika-askeleen jälkeen; k_3 vastaava, mutta kulmakertoimen k_2 avulla; k_4 lopussa eli aika-askeleen päässä nykyhetkestä k_3:lla. Näiden painotettu keskiarvo määrää oletetun derivaatan kohdassa n.

Koska maailman tila on yhtenäinen eri kappaleiden suhteen ja kyseessä on periaatteessa yksi suuri tilavektori, ei kappaleiden tiloja päivitellä vaiheiden välillä erikseen, vaan kaikkien kappaleiden tilat lasketaan askeleiden välillä. Teknisesti tilavektori on hahmottamissyistä hajautettu erillisiin kappaleluokkiin. Esimerkiksi siis k_2:n laskemisessa yhtä kappaletta varten muuta maailmaa ei pidetä paikallaan, vaan vaikutus lasketaan kaikelle. Toisin sanoen muuttujat y ovat vektoreita. Luokka BadRK4 on esimerkin vuoksi toteutettu väärällä tavalla, eli yhtä kappaletta siirrettäessä pidetään muuta maailmaa mikroaskelien kohdilla paikallaan. Luokka RK4 toimii oikein.

Vaihtelun vuoksi toteutettiin myös ns. Velocity Verlet.

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\,\vec{a}(t)\Delta t^2 \tag{7}$$
$$\vec{v}(t + \Delta t) = \vec{v}(t) + \frac{\vec{a}(t) + \vec{a}(t + \Delta t)}{2}\Delta t \tag{8}$$

Tämä on helppo muuttaa koodiksi siten, että uusi paikka lasketaan ensin, sitten uuden paikan kohdassa lasketaan kiihtyvyys uudessa kohdassa, ja lopuksi uuden nopeuden laskemiseen käytetään vanhaa ja uutta kiihtyvyyttä.

4.1.2 Törmäysten hallinta

Törmäykset hallitaan kahden kappaleen välisinä tarkistuksina: integroinnin jälkeen tarkistetaan, onko kahden kappaleen välinen etäisyys pienempi kuin niiden säteet yhteenlaskettuna. Mikäli näin on, tehdään jokin törmäyshallinta: ei tehdä mitään, pysäytetään simulaatio tai kimmotetaan kappaleet.

Jos ei tehdä mitään, simulaatio jatkuu ja kappaleet menevät toistensa lävitse. Pienellä aika-askeleella kappaleiden välinen voima lasketaan väärin (jos ne ovat sisäkkäin, gravitaation vaikutus on erilainen) ja kappaleet singahtavat kauas toisistaan. Teoriassa kappaleet voivat päätyä myös tasan päällekkäin, jolloin niiden välinen nollaetäisyys aiheuttaa nollallajakovirheen.

Kimpoaminen hallitaan täysin kimmoisasti massakeskipisteen kautta. Liikemäärävektorit muunnetaan massakeskipistekoordinaatistoon ja kaavat pyöritellään sitä kautta niin, että tässä koordinaatistossa kappaleet liikkuvat suoraan toisiaan kohti jolloin niiden nopeuksien merkki vaihtuu. Massakeskipiste:

$$p_c m = (m_1 + m_2) \tag{9}$$
$$v_c m = p_1 + p_2 \tag{10}$$
$$v_c m = \frac{m_1}{m_1 + m_2} v_1 + \frac{m_2}{m_1 + m_2} v_2 \tag{11}$$

Uudessa koordinaatistossa:

$$v_{1,cm} = v_1 - v_{cm} \tag{12}$$

$$p_{1,cm} = p_1 - m_1 v_{cm} = m_1 v_1 - (m_1 \frac{m_1}{m_1 + m_2} v_1 + m_1 \frac{m_2}{m_1 + m_2} v_2) \tag{13}$$

$$= \frac{m_1 m_2}{m_1 + m_2}(v_1 - v_2) \tag{14}$$

Kappaleelle 2 vastaavasti.

Nyt liikemäärän muutos on molemmille kappaleille vain merkinvaihto, eli liikemäärän muutos on itse liikemäärä kaksinkertaisena ja negatiivisena. Tästä takaisin maailmakoordinaatistoon muuttamalla saadaan

$$\Delta v_1 = \Delta p_1 / m_1 = -2 \frac{m_1 m_2}{m_1 + m_2}(v_1 - v_2) \tag{15}$$

ja sama kappaleelle 2 vastaavasti. Tämä nopeuden muutos hoidetaan kappaleen nopeuteen World-luokassa, jos asetukset niin määrää.

4.2 Muuta

Asetustiedostot jaetaan tavallaan rekursiivisesti, mutta kuitenkin iteratiivisesti. Alkuperäinen jostain tekstivirrasta ladattava Config luo muita Config-olioita, jos niitä tulee tiedostossa vastaan. Näitä ei kuitenkaan ladata niiden Configien latausfunktioissa, vaan ylimmän tason Config instantioi kaikki. Koska Configeja voi näin olla sisäkkäin, pidetään eri asetusryhmien tasoista muistia pinossa (Deque). Nykytasolla käsiteltävä Config on erikseen muistissa, ja kun uuden asetusryhmän alku tulee vastaan, nykyryhmä laitetaan pinoon ja uudeksi nykyryhmäksi tulee tämä uusi. Kun ryhmä loppuu tietovirrassa, nykyryhmäksi popataan pinon päältä uusi. Tätä ei tehty rekursiivisesti, koska aluksi ConfigBlockit ladattiin vähän hassusti iteratiivisesti ja kiireessä oikea rekursio olisi vaatinut enemmän aikaa.

Grafiikan piirrossa suhteellinen origo on mainitsemisen arvoinen ominaisuus. Globaalin koordinaatiston origo on vaihdettavissa johonkin planeettaan, jolloin eri planeettojen sijainnit ja radat piirretään suhteessa siihen, missä tämä origokappale oli kuhunkin aikaan. Näin jos aurinko liikkuu globaalin koordinaatiston suhteen, voidaan se asettaa origoksi jolloin muiden planeettojen paikat ja radat nähdään selvemmin, kun kokonaisuus ei vaella pikkuhiljaa.

5 Tietorakenteet

Kappaleita voi lisäillä kesken kaiken, joten niitä varastoidaan ArrayListeissä eikä ihan perustaulukoissa. Sama kappaleiden historiadatalle yms. Suora indeksointi on suhteellisen harvaa ja kappaleita lähinnä iteroidaan läpi, joten tilalle sopisi myös linkitetty lista. Sen hyvistä puolista ei tosin juuri ole hyötyä, koska keskeltä ei poisteta tai lisäillä olioita juuri koskaan.

Historiadata vois olla linkitetyssä listassa, jos siihen olisi tehty muistioptimointi poistamalla alusta dataa sitä mukaa kun sitä tulee alkuun.

Asetusryhmän (Config) sisältämät toiset ryhmät varastoidaan taulukoita sisältävässä hashmapissa. Hashmapista saa taulukollisen tietyn nimisiä ryhmiä, koska tätä tarvitaan jossain (esim. haetaan kaikki body-ryhmät). Näin ryhmiä ei tarvitse hakea kaikkien ryhmien sisältävästä taulukosta aina, kun niitä tietyllä nimellä tarvitaan.

Raakoja taulukoita käytetään integraattorissa, kun tarvitaan useita väliaikaistaulukoita joiden koko ei muutu miksikään, vaan ne alustetaan valmiiksi oikean kokoisiksi.

6 Tiedostot

Tiedostoformaatti suunniteltiin helppoa muokattavuutta silmälläpitäen. Koska maailman tiedot kuitenkin kannattaa kirjoittaa ihmisen toimesta tiedostoon eikä naksutella GUIsta, muoto on varsin minimalistinen eikä kovin ihmeellinen.

Esimerkki 1, planeetan määritykset:

```
/* C-tyylisiä kommentteja tuetaan */
/* Kommentit voivat
 * olla monirivisiäkin */ /* ja monta per rivi */
body {
        name Earth /* kappaleella on nimi */
        mass 5.9736e24 /* luvut ovat Double.parseDouble-muodossa */
        radius 6371e3
        kepler { /* kepler-koordinaatit */
                center Sun
                a 149598261e3
                e 0.01671123
                i 7.155
                w 114.20783
                ma 357.51716
                O 348.73936
        }
        gfx {
                texture earth.png
                color 0,0,1 /* rgb */
        }
}
```

Esimerkki 2, itse asetustiedosto joka annetaan ohjelmalle:

```
include conf/constants.conf
dt 864 /* sadasosa päivästä */

log.defaults {
        type gnuplot
}
log {
        tick 1
        file full.log
}
log {
        tick 100 /* kerran päivässä, sadan askeleen välein */
        filter Sun /* lokita vain aurinko */
        file sun.log
}
log {
        tick 10
        filter Earth
        file earth.log
}
include conf/sun.conf
include conf/mercury.conf
include conf/venus.conf
```

```
include conf/earth.conf
include conf/moon.conf origin=body:Earth vorigin=body:Earth
```

Include-sana lukee toisen tiedoston ja sisällyttää sen siihen kohtaan, missä se sijaitsi alkuperäisessä. Tämä hoidetaan Configin merge-metodilla. Includaamiselle voi antaa erityisparametreja, toistaiseksi origin ja vorigin, joilla säädetään sen alla olevia globaaleja muuttujia. Näin alitiedostolle saadaan suhteellinen alkuorigo, eikä esim. kuun sijaintia tarvitse laskea maan koordinaateista käsin. origin=body:Earth käy, sekä myös koordinaattipohjainen muoto origin=pos:10,20,30. Myös nopeusorigon voi asettaa; se on vorigin.

Asetusryhmällä on jokin nimi (esim. body, kepler tai log) ja se on aina jonkin toisen asetusryhmän sisällä, poislukien ylimmän tason ladattava tiedosto joka ei ole minkään sisällä. Ryhmän sisältö ympäröidään aaltosuluilla. Parserissa ei ole kovinkaan paljon älyä, joten avaavan aaltosulun on oltava rivin lopussa, ja sulkevan on oltava rivin ainoa merkki. Välilyöntejä, tabeja ja kommentteja ei tosin lasketa tähän.

Kappaleille voi antaa sijainti- ja nopeusvektorin, tai sitten kepler-lohkossa kiertorataparametreina, mistä sijainti ja nopeus päätellään. Kiertorataparametrien kulmat ovat asteissa.

7 Testaus

Ohjelman päätoiminnallisuutta testattiin aluksi yksikkötesteillä sekä isompia ja grafiikkaan liittyviä testejä tehtiin käsin. Asetuksille on muutama yksikkötestaus, mutta niitäkin tuli kokeiltua käsin niin, että ne toimivat halutusti kokonaisuuden kera. Systemaattista JUnit-koodia voisi olla enemmän, mutta koska ohjelmassa käytetään vähän kaikkea kaikesta, ongelmat tulevat kätevästi systeemitestauksessa ilmi.

Simulaation oikeellisuus koestettiin pääosin testailunäkymästä katselemalla, että vuosia eteenpäin simuloimalla kappaleiden radat pysyvät oikeina sekä ajamalla yksinkertaisia automaattisia testejä ja seuraamalla ajolokeja. Lagrangen pisteiden kokeilu unohtui.

Törmäykset testattiin tekemällä asetustiedosto jossa on kaksi kappaletta paikallaan, ja toteamalla että komentoriviohjelma päättyy kun ne törmäävät; törmäyksen saa tarkasteltua lokitiedostoista helposti jälkeenpäin. Kimpoilemisenkin näkee selvästi graafisesta käyttöliittymästä.

Monia ohjelman toiminnallisuuksia tuli kokeiltua käyttöliittymästä, ja käyttöliittymän toimivuus tuli tarkistettua siinä samalla. Eri tiloista toisiin piti kokeilla että kaikki toimivat edelleen; esim. pausetilan syklittäminen ei hajota mitään. Kappaleiden muokkaus toimii testatusti ja planeetat siirtyvät kun niiden tietoja muokataan numeronäkymästä.

Renderöinti näyttää oikealta ja siinä navigointi (pyöritys ja siirto) toimii odotetusti. Origon muuttamisen oikeellisuuden näkee siitä, että auringon ollessa origo muiden radat eivät venähdä, ja esim. Maan origoksi asettaminen saa auringon näyttämään siltä, että se kiertäisi Maata.

Kaikki toteutetut asiat testattiin ja todettiin toimivaksi.

Esimerkiksi törmäystestaus hoituu helposti antamalla ohjelmalle conf/collisiontest.conf-tiedosto ja muokkaamalla siitä collisiontype-muuttujaa.

8 Ohjelman tunnetut puutteet ja viat

Virhetilanteiden hallinta ei ole erityisen käyttäjäystävällistä. Jotkut paikat heittävät tarkoituksella RuntimeErroreita, koska käyttöliittymän virheilmoitusten hallinta ei erityisemmin kiinnostanut eikä sitä ollut määritelty suunnitelmassa erityisen tarkasti. Ohimennen suunniteltiin, että käyttöliittymä käsittelisi asetustiedostojen poikkeukset, mutta niitä ei priorisoitu korkealle ja jäivät toteuttamatta; oletetaan, että käyttäjä tietää mitä tekee ja tarkastelee konsolia poikkeusten varalta. Myöskään esim. nimissä ei saisi olla duplikaatteja, mutta sitä ei testata. Jossain osissa koodia haetaan kappaleita nimen perusteella, ja tällöin saadaan ensimmäiseksi löytynyt.

Toisaalta poikkeuksia ei tavallisessa käytössä pitäisi ihmeemmin lennellä, vaan enimmäkseen asetustiedostojen virheellisyydestä ohjelmaa käynnistäessä. Erityispoikkeusten kuvausviestit ovat kuitenkin järkeviä. Pienemmistä virheistä tulostetaan varoituksia konsoliin kaatumatta.

Jotkut suunnitellut ominaisuudet jäivät toteuttamatta; näistä puutteista tarkemmin kohdassa Poikkeamat suunnitelmasta.

Oheisella esimerkkiaurinkokunnalla muita kappaleita ei oikein näe paitsi tekstin suhteen, koska aurinko on niin massiivinen. Tämä on pikemminkin ominaisuus eikä vika, koska koot renderöidään oikeassa suhteessa.

9 3 parasta ja 3 heikointa kohtaa

Teknisesti parhaita lienevät asetustiedoston nätti rekursiivisuus ja lokitiedostojen modulaarisuus, keino-origon säätö ja muu 3d-tilan toimivuus sekä ja yleinen koodin laajennettavuus mihin kuuluu lokitiedostojen säädettävyys ja integrointimenetelmän joustavuus. Toki RK4-integrointikin on mukavan tarkka, mutta sehän on prujattu wikipediasta eikä itse menetelmä ole omaa tuotantoa.

Heikoimpia kohtia ovat käyttöliittymän rumuus ja kankeus, virhetilanteiden viimeistelemättömyys käyttäjän näkökulmasta sekä automaattisten testien vähyys.

10 Poikkeamat suunnitelmasta

Joitain suunniteltuja ominaisuuksia puuttuu, koska projektiin ei käytetty ihan niin paljoa aikaa kuin mitä alussa tuli kuviteltua.

Suunniteltu anaglyfikolmiulotteisuus jäi toteuttamatta, mutta se on helppo lisätä, sillä opengl tukee sitä varsin suoraan. Se onnistuisi renderöimällä kahdesti päällekkäin vaihtamalla kameran sijaintia ja värimaskia välissä.

Kappaleita ei voi siirtää tai seurata 3d-näkymässä. Siirtäminen olisi kuitenkin hankalaa tehdä tarkasti käyttäjän kannalta, jolloin on parempi muokata suoraan numeroarvoja. Seuranta vain ei ehtinyt mukaan.

Satelliitin/raketin laukaisulle ei ole käyttöliittymässä nappia eikä niitä voi lisätä jälkeenpäin, vaan ne on laitettava asetustiedostoon valmiiksi (mallia voi ottaa conf-hakemiston esimerkkitiedostoista).

Törmäyksissä kappaleet eivät hajoa palasiksi.

Kappaleiden lopputilaa ei voi tallentaa konfigurointitiedostoon, mutta tunnin koodaamisella ja testauksella sekin olisi mahdollista. Config-luokassa on valmiiksi tiedostoonkirjoitusfunktio tätä varten; kappaleiden tiedot pitäisi päivittää asetuslohkoihin ja kirjoittaa ne johonkin käyttäjän määrittämään tiedostoon.

11 Toteutunut työjärjestys ja aikataulu

Projektin työjärjestys toteutui varsin oikein, sillä suunnitelmassa ollutta tuli seurattua miettiessä mitä koodaisi seuraavaksi, koska se oli rakenteellisesti hyvä. Aikataulu sen sijaan ei oikein toteutunut, koska lukukauden keskivaiheilla tuli niin merkittävästi muuta kiireellisempää puuhaa, että tämä jäi vähemmälle. Ajankäyttökin meni hieman pieleen, koska asioissa ei tullut keulittua lainkaan niin paljoa kuin mitä suunnitelmassa arvioitiin. Alussa tuli koodattua aika paljon, sitten hyvin vähän ja lopuksi deadlinen lähestyessä tuli loppuspurtti.

Seuraavassa viikkokohtaiset suunnitellut ja toteutuneet asiat:

7 Suunniteltu tutustumista ja vaihtoehtojen arviointia yms. Toteutui varsin tarkasti.

8 Suunniteltu edelleen hahmottelua ja perustoiminnallisuuksia vähän kaikesta. Toteutui hieman jäljessä.

9 Suunniteltu pientä kehittymistä edellisestä. Tässä otettiin edellistä hieman kiinni; toteutui.

10 Suunniteltu käytettävyysjuttuja guin ja 3d-näkymän suhteen. Toteutuivat puoliksi tällä ja puoliksi edeltävällä viikolla.

11 Suunniteltu 3d-näkymään tekstuurit ja navigointi sekä valinta. Ei toteutunut; kotimaan Rankka rokotti kouluhommista.

12 Suunniteltu lokitiedostot ja numerotilan aloittelu. Toteutui hieman, tuli aloiteltua vähän kaikkea.

13 Suunniteltu konffien virheellisyyden hallintaa ja konffauspaneelin reagointia muutoksiin. Tuli jatkettua viime viikon asioita ja aloiteltua näitä; toteutui kohtalaisesti.

14 Suunniteltu 3d-näkymään hienouksia. Toteutui varsin hyvin edellisiin verrattuna.

15 Suunniteltu viilausta; toteutui suunnitelmien puitteissa aika raskaasti.

16 Suunniteltu DL:ään viimeiset viilailut, toteutui aggressiivisena koodaamisena ja dokumentaation hakkaamisella viimeisinä päivinä ja öinä.

Aikaa tuli käytettyä karkeasti arvioiden kuutisenkymmentä tuntia koodailut, dokumentoinnit ja googlailut mukaanlukien.

12 Arvio lopputuloksesta

Projekti onnistui periaatteessa hyvin vaikka joitain hienouksia jäi toteuttamatta; oleelliset asiat löytyvät ja koodi on helposti laajennettavaa, jolloin kiinnostuksen jatkuessa hienoudet voi toteuttaa helposti vaikka pitkän ajan päästä. Koska perustoiminnallisuuteen ja hyvään koodiin sekä dokumentaatioon tuli kiinnitettyä eniten huomiota, huonona puolena käyttöliittymämukavuus ja viimeiset viilailut jäivät vähän vähemmälle.

Modulaarisuutta ja laajennettavuutta helpottavat mm. seuraavat ominaisuudet: Lisäasetuksia on helppo lisätä tarvittaessa, koska asetustiedostot ovat varsin käyttöagnostisia ja aidosti rekursiivisia. Lokitiedostotyyppejä voi kirjoittaa lisää lisäämällä Writer-luokkia. Integraattoria voi vaihtaa koodaamalla uuden luokan sitä varten ja vaihtamalla sen World-luokkaan käyttöön. Integrointi ja kiihtyvyyksien laskeskelu tuli toteutettua varsin hyvin, sillä parin viimeisen illan viilailut sen suhteen veivät hyvin vähän aikaa, vaikka muutokset olivatkin varsin radikaaleja. Bodystä voi periä uudenlaisia lentokappaleita helposti. GfxBodyn voi erikoistaa piirtämään joitain muita kappaleita kuin palloja. SettingPaneen voi lisätä uusia ominaisuuksia naputtelemalla Widget-luokkia ja lisäilemällä niille sinne tuen.

Mielenkiintoista olisi ollut saada jostain suuri läjä eri kappaleiden tietoja (luokkaa satoja tai tuhansia kappaleita) ja testata ohjelman suorituskykyä ja todenmukaisuutta niillä. Nyt suorituskykykokeissa voi tyytyä generoimaan asetustiedoston, jossa on suuri määrä satunnaiskappaleita.

13 Viitteet

Simulaatiosta tuli lueskeltua muutama RK4-artikkeli. Keplerkoordinaattien muunnoksesta tuli luettua nippelitietoa sieltä sun täältä. Taivaankappaleiden keplerratatiedot ovat Wikipediasta kunkin kappaleen omalta sivulta, joita on turha erikseen luetella kaikkia. Javan API tarjosi ohjeita Swingissä. Swingin tutoriaalit olivat myös hyödyksi. OpenGL:n saloja tuli heräteltyä muistista muutamalla pienellä oppaalla sekä api-speksillä. Myös toteuttamattomia ominaisuuksia tuli opiskeltua, kuten anaglyyfipiirtoa ja barnes-hut-optimointia.

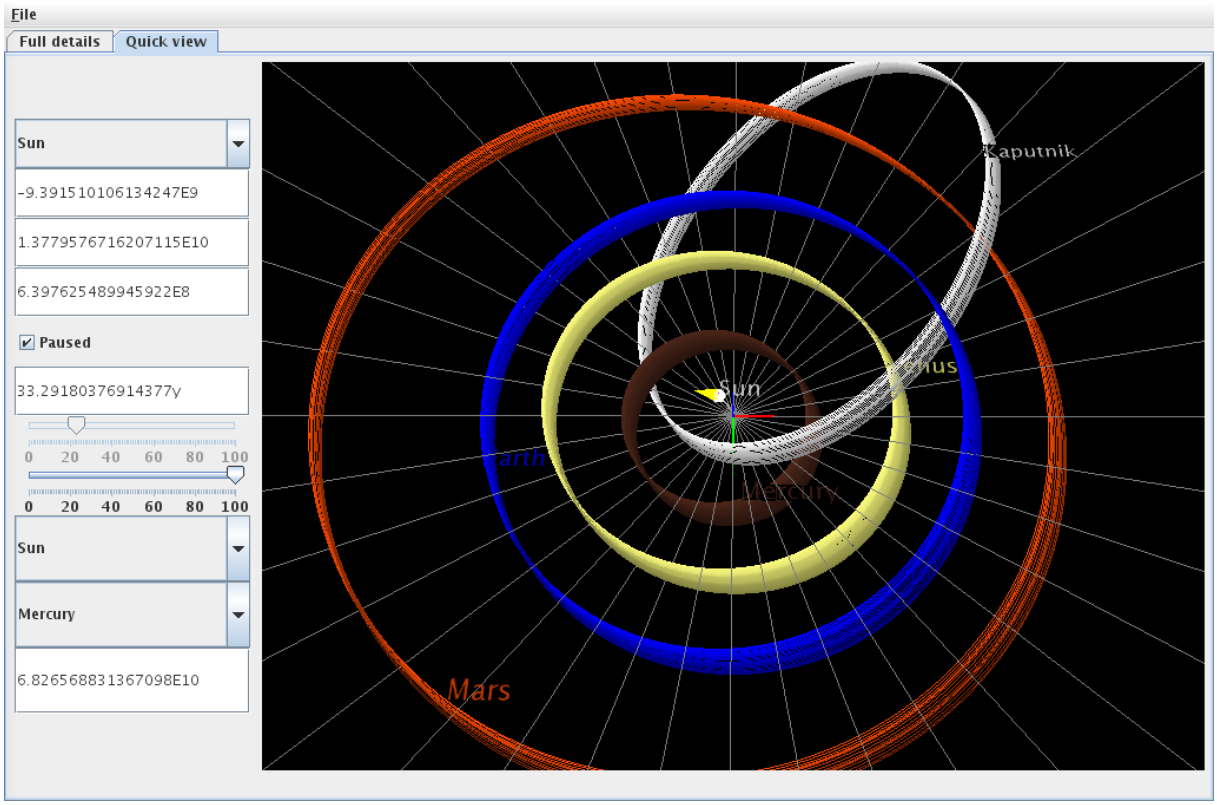
- `http://en.wikipedia.org/wiki/Verlet_integration`

- http://en.wikipedia.org/wiki/Runge-Kutta_methods
- http://microsat.sm.bmstu.ru/e-library/Ballistics/kepler.pdf
- http://ccar.colorado.edu/ASEN5070/handouts/kep2cart_2002.doc
- http://www.colorado.edu/ASEN/asen3200/download.html
- http://en.wikipedia.org/wiki/Kepler's_laws_of_planetary_motion
- http://en.wikipedia.org/wiki/Standard_gravitational_parameter
- http://en.wikipedia.org/wiki/Shell_theorem
- http://en.wikipedia.org/wiki/Solar_System
- http://en.wikipedia.org/wiki/Orbital_elements
- http://en.wikipedia.org/wiki/Earth ym. planeetat
- http://download.oracle.com/javase/6/docs/api/
- http://download.oracle.com/javase/tutorial/uiswing/components/index.html
- http://nehe.gamedev.net/
- http://timelessname.com/jogl/lesson01/
- http://people.eecs.ku.edu/~miller/Courses/JOGL/jogl-1.1.1-docs/
- http://paulbourke.net/texture_colour/anaglyph/
- http://paulbourke.net/miscellaneous/stereographics/stereorender/
- http://www.cs.princeton.edu/courses/archive/fall03/cs126/assignments/barnes-hut.html
- http://cnx.org/content/m26666/latest/
- http://en.wikipedia.org/wiki/Two-body_problem

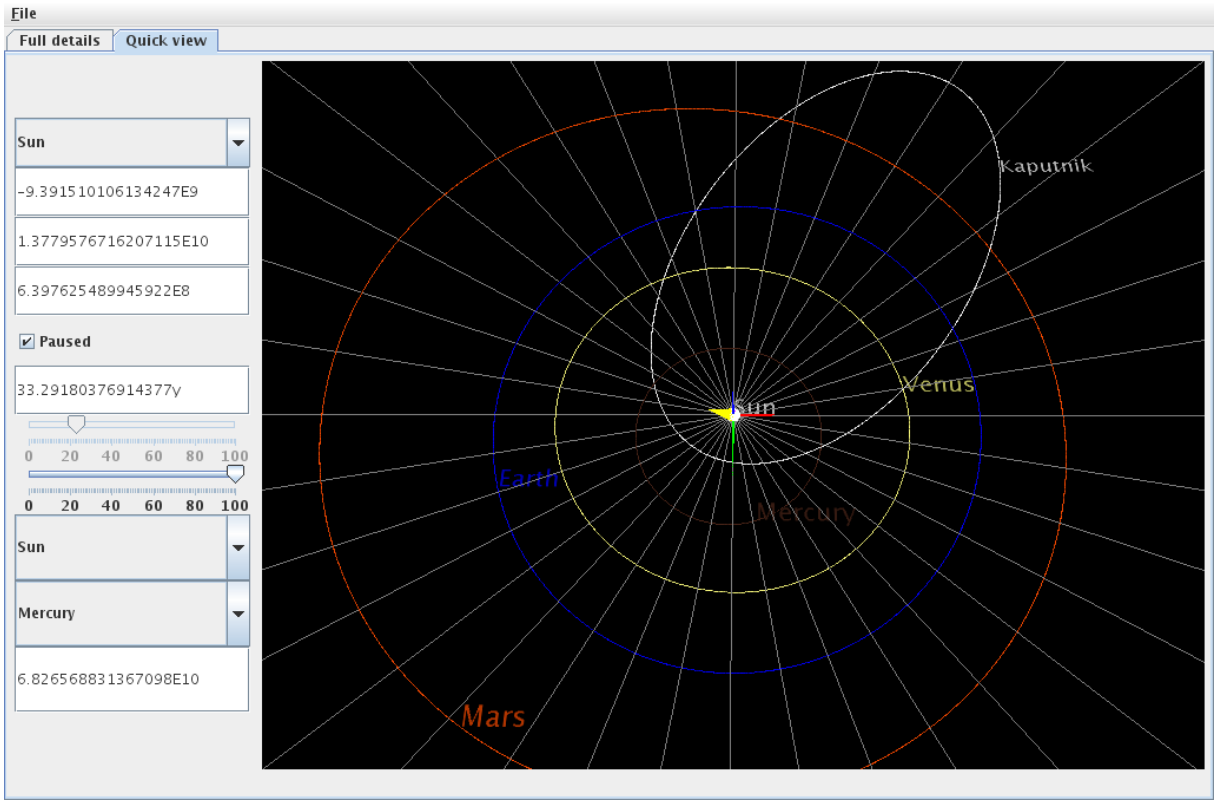
14 Liitteet

Lopuksi pari hassua esimerkkiä piirtelystä. Kuvassa 4 näkyy, kuinka kappaleiden radat vaeltavat suhteessa globaaliin koordinaatistoon, koska aurinko vaeltaa myös. Kuvassa 5 on sama tilanne, kun aurinko on asetettu origoksi. Kuvassa 6 origon on maapallo, ja muiden kappaleiden radat menevät vinksin vonksin. Kuvassa 7 on kuva satunnaisgeneroidusta 200 kappaleen tilanteesta.

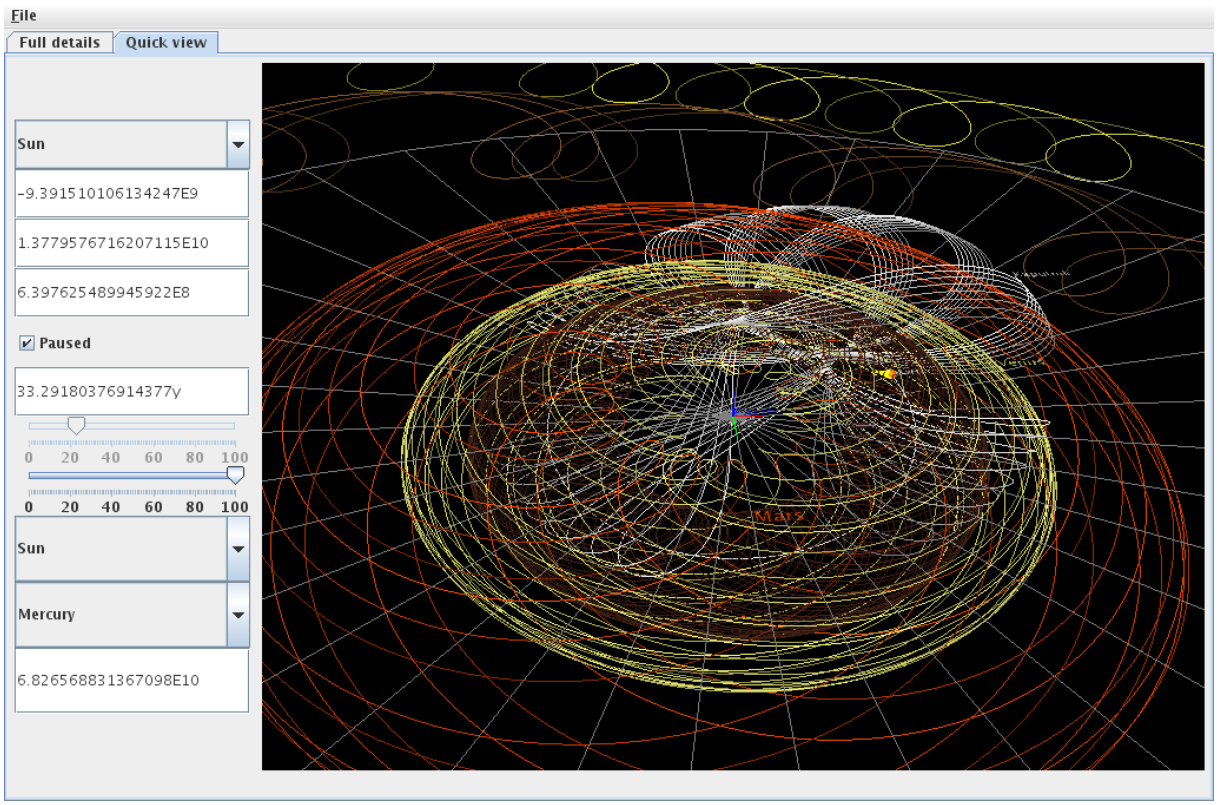
Ohjelmakoodi on myös liitteenä erillisenä tiedostona.



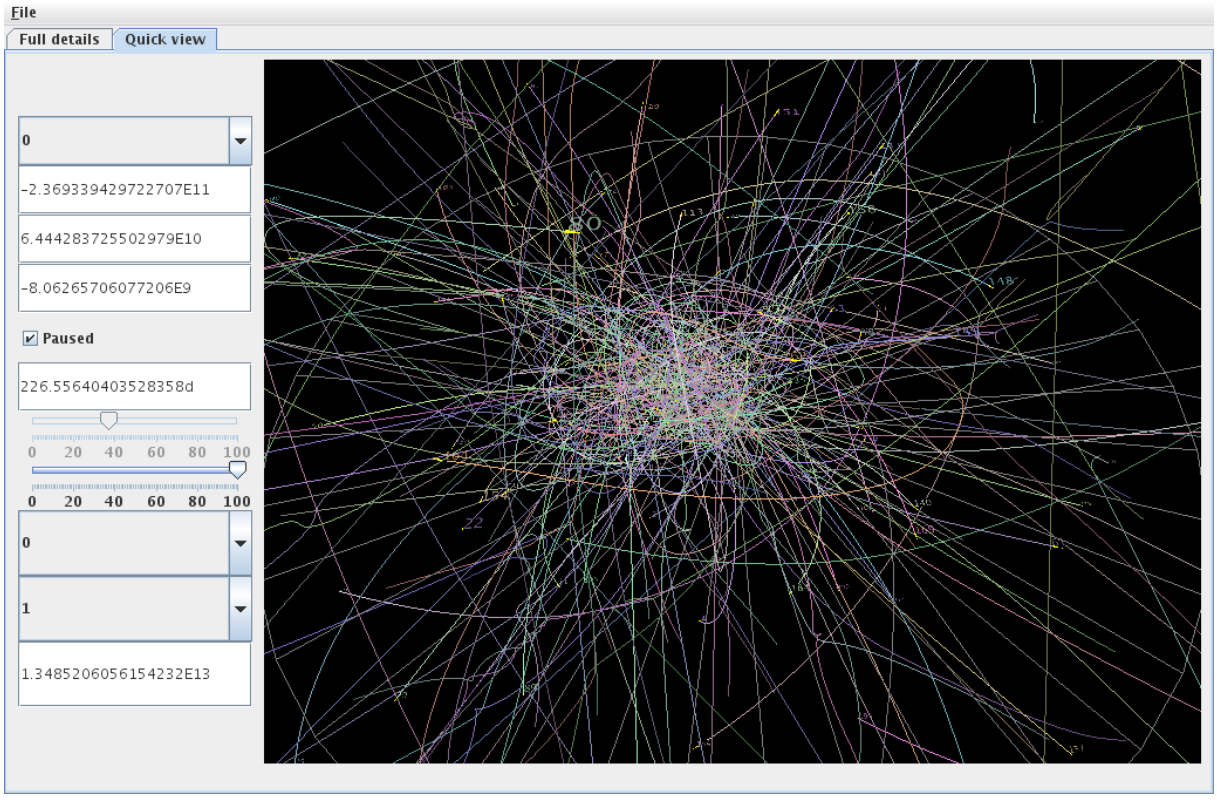
Kuva 4: Radat vaeltavat



Kuva 5: Vaeltaminen eliminoitu



Kuva 6: Vempularatoja suhteessa maapalloon



Kuva 7: Satunnaiskappaleita