Aurinkokuntasimulaattori

Eemeli Oksanen
1015557

Tietotekniikka, 2021

3.5.2022

YLEISKUVAUS

Tekemäni projekti on aurinkokuntasimulaattori, jolla pystytään simuloimaan aurinkokunnan ja muiden avaruuskappaleiden käyttäytymistä. Ohjelma laskee kappaleiden sijainnit karteesisessa koordinaatistossa käyttäen neljännen kertaluvun Runge-Kutta-menetelmää tarkkuuden parantamiseksi.

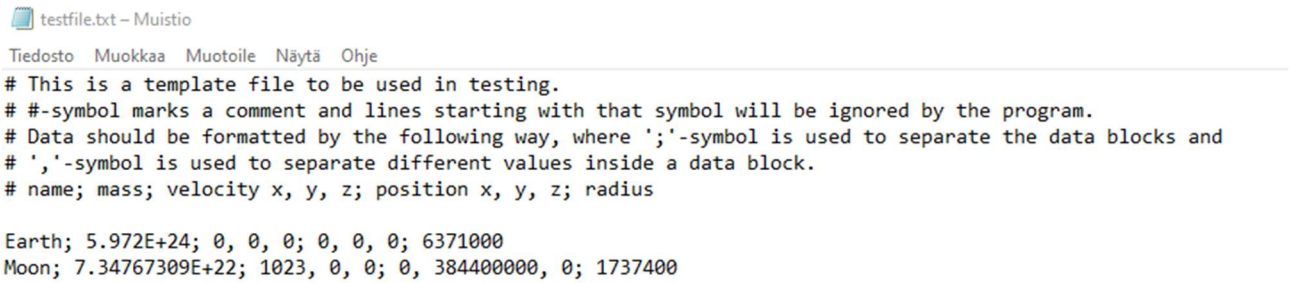
Työ on mielestäni toteutettu vähintään keskivaikeana, mutta kaikki vaikean kriteerit eivät täyty.

KÄYTTÖOHJE

Ohjelmaa on tällä hetkellä mahdollista käyttää kahdella eri tavalla. Konsolipohjaisella käyttöliittymällä ja graafisella käyttöliittymällä, joka kuitenkaan ei ole täysin valmis eikä kykene kaikkiin haluttuihin toimenpiteisiin. Tässä käyttöohjeet ohjelmalle.

TIEDOSTON LUONTI

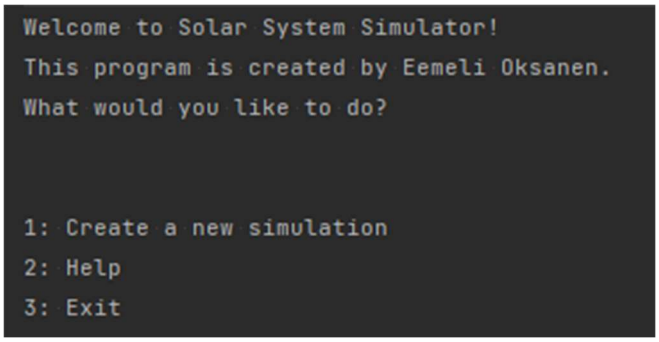
Ohjelma lukee avaruuskappaleiden sijainnin tekstitiedostosta. Tekstitiedostolla tulee olla seuraava rakenne:



Ohjelma sivuuttaa kaiken tekstin ”#” -merkin jälkeen jokaisella rivillä. Kommentteja voi siis lisätä suoraan uudelle riville tai datarivien perään. Rivin sisällä eri arvot erotetaan ”;” -merkillä ja arvojen komponentit ”,” -merkillä. Esimerkiksi siis kappaleen nopeuden x-, y- ja z -komponentit erotetaan pilkulla. Arvojen täytyy olla ennalta määrätyssä järjestyksessä, jotta ohjelma pystyy lukemaan ne. Desimaalierottimena käytetään pistettä ”.”.

MERKKIPOHJAINEN KÄYTTÖLIITTYMÄ

Aja tiedosto SimulatorApp.scala avataksesi merkkipohjaisen käyttöliittymän



Kun ohjelman ajaa, ilmestyy konsoliin kuvan teksti. Valittavissa on kolme eri vaihtoehtoa. Ensimmäinen luo uuden simulaation, toinen tulostaa lyhyen tiivistelmän ohjelman käytöstä ja kolmas sulkee ohjelman.

Valittaessa ensimmäisen vaihtoehdon ohjelma kysyy käyttäjältä seuraavat kysymykset järjestyksessä:

1. Aika-askel, jota halutaan simulaatiossa käyttää

2. Simulaation maksimikesto (tulee olla suurempi kuin aika-askel)

3. Simulaatiotiedoston tiedostopolun. Esim. D:\Ohjelmointi\simulaatiotiedosto.txt

Näiden jälkeen ohjelma kysyy käyttäjältä, miten laskettu data halutaan saada. Ensimmäinen vaihtoehto tulostaa datan Gnuplot-ohjelmalla luettavissa olevaan tiedostoon (http://www.gnuplot.info/) ja toinen tulostaa datan konsoliin.

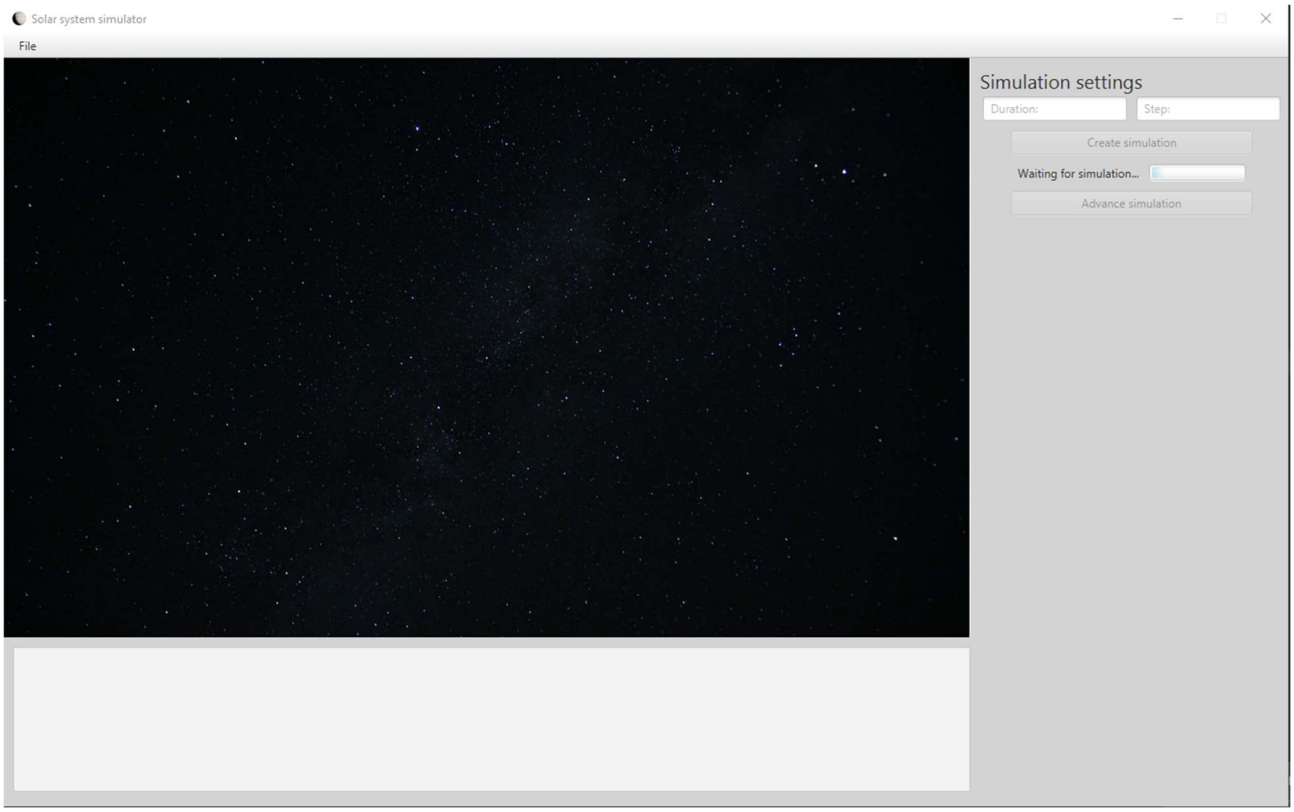
Valittaessa tiedostoon tallentamisen, ohjelma kysyy käyttäjältä, mihin tämä haluaisi tiedoston tallentaa. Esimerkiksi D:\Ohjelmointi\data.txt

Tämän jälkeen ohjelma joko tulostaa konsoliin tai tallentaa lasketun datan tiedostoon ja ilmoittaa joko toimenpiteen onnistumisesta tai epäonnistumisesta.

GRAAFINEN KÄYTTÖLIITTYMÄ (EI TÄYSIN TOIMIVA)

Vaikka graafinen käyttöliittymä ei ole täysin valmis, halusin silti sen sisällyttää palautukseen, koska sillä on hieman toiminnallisuutta. Graafisen käyttöliittymän saa avattua ajamalla SimulatorAppGUI.scala -tiedoston.

Ohjelman avatessasi ruudulle pitäisi ilmestyä seuraavanlainen ikkuna:

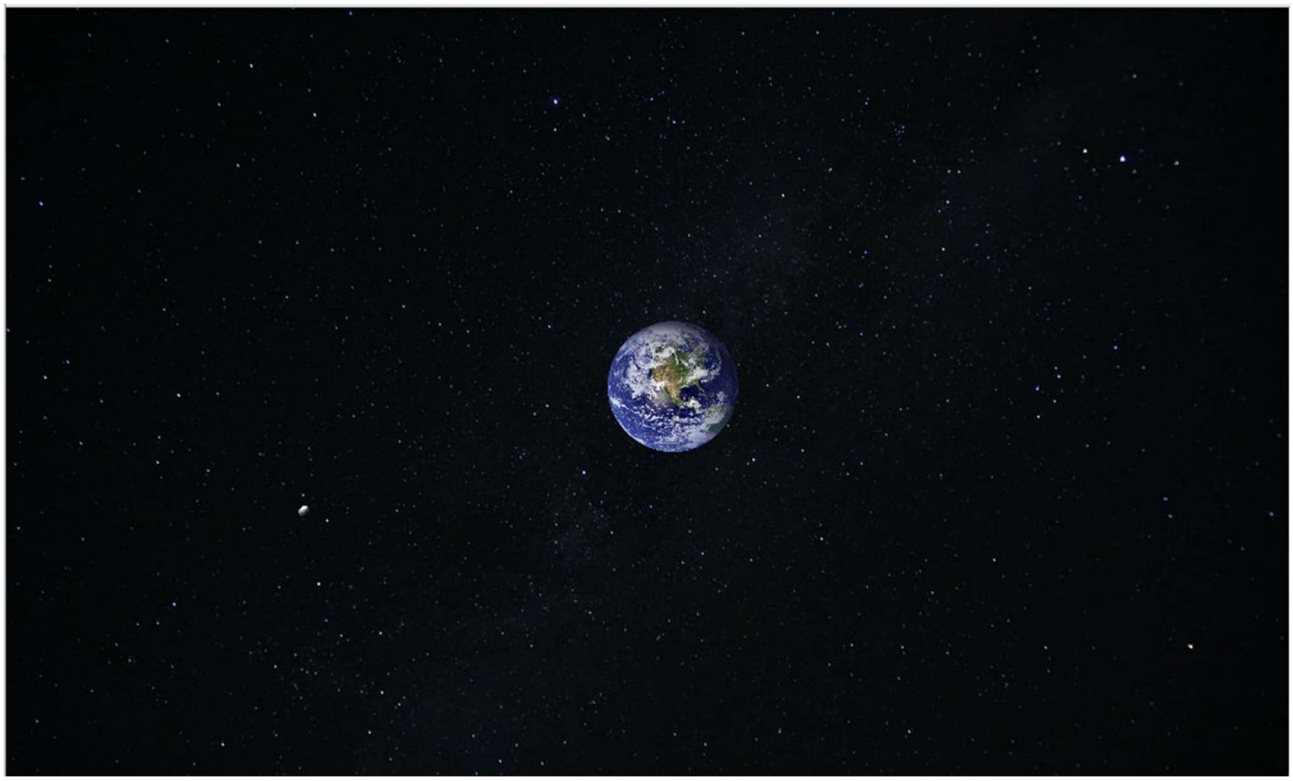


Ikkuna sisältää piirtoalustan (vasen yläreuna, avaruus), statuskonsolin (vasen alareuna) ja ohjauspaneelin (oikealla). Piirtoalustalle piirretään avaruuskappaleet, konsoli tulostaa statusviestejä ja ohjauspaneelista ohjataan simulaation kulkua.

Jotta saataisiin simulaatio käyntiin, on meidän ensin ladattava se tiedostosta (sama tiedostomuoto kuin tekstipohjaisessa käyttöliittymässä.) Tiedoston voimme avata valitsemalla vasemmasta yläkulmasta ”File”-valikosta ”New simulation from file…”. Valikosta löytyy myös ”Export as .dat file… ” -kohta, mutta sillä ei ole mitään toimintoa.

Nyt ohjelman pitäisi avata resurssienhallintaikkuna, jonka avulla voit valita simulaatiotiedoston. Jos tiedosto löydettiin onnistuneesti, pitäisi konsoliin ilmestyä viesti.

Tämän jälkeen meidän täytyy vielä asettaa ohjauspaneelista simulaation pituus ja käytettävä aika-askel. Ohjelma heittää varoituksen, mikäli yrität käynnistää simulaatiota väärillä arvoilla. Simulaatio aloitetaan ”Create simulation” -painikkeesta.



Aloitettuasi simulaation ruudulle pitäisi nyt ilmestyä luomasi planeetat. Ohjelma skaalaa piirtoalueen siten, että kaikki kappaleet mahtuvat ruudulle. Jos planeetta olisi liian pieni (halkaisija alle 10 pikseliä), piirretään se 10 pikselin kokoisena, jotta se näkyisi ruudulla. Simulaatiota voi tällä hetkellä ainoastaan liikuttaa manuaaliesti eteenpäin yhden askeleen verran. Tämä onnistuu ”Advance simulation” -painikkeella. Simulaatio loppuu itsestään joko silloin, kun aikaa on kulunut tarpeeksi tai jos tapahtuu törmäys. Simulaation voi aloittaa alusta painamalla uudelleen ”Create simulation” -painiketta.

GNUPLOT-LUETTAVA TIEDOSTO

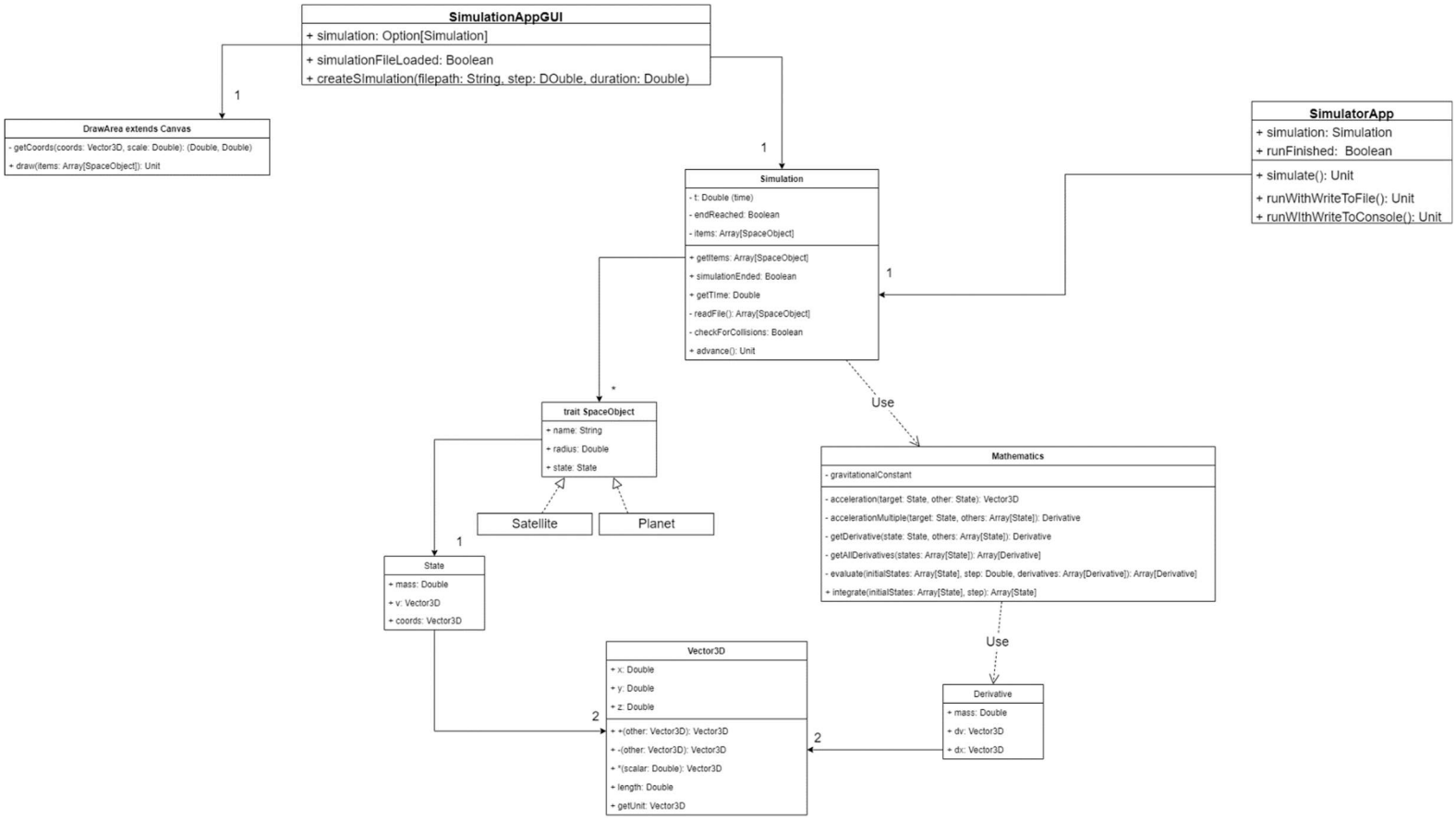
Tekstipohjainen ohjelma kykenee tulostamaan lasketun datan Gnuplot-luettavassa muodossa. Tulostetun tiedoston rakenne on seuraava:

```
0.0 0.0 0.0 0.0 0.0 3.844E8 0.0 0.0 -2.044E8 0.0
1000.0 -0.08318962692306545 -42.094376246524185 0.0 1022998.7953831125 3.8439864423111224E8 0.0 -1022992.0339438663 -2.0439522290830886E8 0.0
2000.0 -0.6655346175750149 -168.3787552847824 0.0 2045990.363079559 3.843945769346302E8 0.0 -2045936.270134929 -2.0438089154179567E8 0.0
3000.0 -2.2462802644759123 -378.8568879903204 0.0 3068967.475463329 3.843877981410977E8 0.0 -3068784.9035718585 -2.043570056261427E8 0.0
4000.0 -5.3248517506102715 -673.535025666418 0.0 4091922.90502972 3.8437830790142083E8 0.0 -4091490.1147535997 -2.043235647041662E8 0.0
5000.0 -10.400944176891054 -1052.4219198095643 0.0 5114849.424455996 3.8436610628686756E8 0.0 -5114004.062422824 -2.0428056813583496E8 0.0
6000.0 -17.97461268836097 -1515.528821779959 0.0 6137739.8066620305 3.843511933890673E8 0.0 -6136278.876301357 -2.042280150982967 3E8 0.0
7000.0 -28.54636273214215 -2062.8694823758724 0.0 7160586.824870965 3.843335693200101E8 0.0 -7158266.649814888 -2.0416590458591196E8 0.0
8000.0 -42.617240480229924 -2694.460151310366 0.0 8183383.25266985 3.843132342120459E8 0.0 -8179919.43280429 -2.040942354102956E8 0.0
9000.0 -60.6889234503325 -3410.319576588527 0.0 9206121.864070287 3.842901882178834E8 0.0 -9201189.224220827 -2.0401300620036674E8 0.0
10000.0 -83.26381135808563 -4210.469003783037 0.0 1.0228795433569066E7 3.8426443151058906E8 0.0 -1.0222027964802563E7 -2.0392221540240622E8 0.0
11000.0 -110.84511723412037 -5094.932175205544 0.0 1.1251396736208806E7 3.842359642835858E8 0.0 -1.1242387529729232E7 -2.038218612801228E8 0.0
12000.0 -143.93695883963247 -6063.735328970967 0.0 1.227391854763857E7 3.842047867506514E8 0.0 -1.2262219721252838E7 -2.0371194191472772E8 0.0
13000.0 -183.04445041429474 -7116.907197951488 0.0 1.3296353644174503E7 3.841708991459172E8 0.0 -1.3281476261301251E7 -2.0359245520501816E8 0.0
14000.0 -228.6737947905675 -8254.479008616654 0.0 1.4318694802860431E7 3.841343017238659E8 0.0 -1.430010878405200 1E7 -2.034633988674697E8 0.0
15000.0 -281.33237590869953 -9476.48447975562 0.0 1.534093480152848E7 3.840949947593301 5E8 0.0 -1.5318068828473508E7 -2.033247704363383E8 0.0
16000.0 -341.52885176697026 -10782.959821077216 0.0 1.6363066418859672E7 3.840529785474901E8 0.0 -1.6335307830830922E7 -2.031765672637 72E8 0.0
17000.0 -409.7732478420065 -12173.943731683126 0.0 1.7385082434444517E7 3.8400825340387 15E8 0.0 -1.7351777171753753E7 -2.030187865199329E8 0.0
18000.0 -486.57705101431054 -13649.477398409077 0.0 1.840697562884359E7 3.8396081966434294E8 0.0 -1.8367427895662423E7 -2.0285142519312945E8 0.0
19000.0 -572.4533040344647 -15209.60449402857 0.0 1.9428738783648115E7 3.839106776851139E8 0.0 -1.938221124915088E7 -2.026744800899601E8 0.0
```

Rivien arvot on erotettu välilyönnillä. Ensimmäinen arvo on aina simulaation alusta kulunut aika. Seuraavilla riveillä ovat kappaleiden sijaintien x-, y-, ja z-komponentit. Nämä on myös

eroteltu välilyönnillä. Sijainnit ovat aina siinä järjestyksessä, jossa ne on simulaatiotiedostossa esitetty.

OHJELMAN RAKENNE



Ylhäällä näkyy UML-kaavio, joka kuvaa tärkeimmät ohjelman käyttämän luokkajaon.

Koko ohjelman keskiössä on Simulation-luokka, joka kuvaa ajettavaa simulaatiota. Simulaatiolla on muuttujia kuvaamaan aikaa, simulaation sisältämiä avaruuskappaleita ja sitä, onko simulaation loppu saavutettu.

getItems,- ja getTime-metodit palauttavat simulaation sisältämät kappaleet sekä ajan. readFile-metodi lukee simulaatiotiedoston ja luo sen avulla uusia SpaceObject-olioita. checkForCollisions -metodi tarkistaa, onko törmäyksiä tapahtunut ja advance() -metodi ajaa simulaatiota eteenpäin yhden askeleen verran.

Satellite,- ja Planet -luokat ovat SpaceObject-piirteen perillisiä. SpaceObjectilla on nimi, säde ja kappaleen tilaa kuvaava State-olio.

State-luokka sisältää kappaleen massan, nopeusvektorin ja koordinaattivektorin. Nämä vektorit ovat Vector3D-luokan ilmentymiä. Vektoreilla on x-, y-, ja z-komponenttien lisäksi metodeita, joiden avulla vektoreilla voidaan suorittaa laskutoimituksia. Lisäksi vektoreilla

on length-metodi, joka palauttaa vektorin pituuden ja getUnit-metodi, joka palauttaa yksikkövektorin.

Mathematics-objekti sisältää Simulation-luokan tarvitsemat matemaatiiset työkalut. Näistä oleellisin on integrate-metodi, joka laskee avaruuskappaleiden uuden tilan (State) yhden aika-askeleen päähän. Matematiikkaolio käyttää laskuissa apuna Derivative-luokkaa, joka kuvaa State-olion derivoitua muotoa. Se sisältää kappaleen massan, nopeuden muutoksen eli kiihtyvyyden sekä paikan muutoksen eli nopeuden.

Simulaatiota käytetään kahdella eri rajapinnalla. Näistä ainoa täysin toteutettu on SimulatorApp-luokka, joka toimii tekstipohjaisena käyttöliittymänä. SimulatorApp-luokalla on yksi Simulation-olio sekä tieto siitä, onko päästy simulaation loppuun.

simulate-metodi toimii ohjelman ydintoimintojen käynnistämisessä. Se luo simulaation, laskee halutun datan ja tulostaa sen käyttäjän päättämällä tavalla.

runWithWriteToFile()-metodi ajaa ohjelman siten, että laskettu data kirjataan tiedostoon. runWithPrintToConsole()-metodi taas tekee saman tulostaen datan konsoliin.

SimulationAppGUI-luokka toteuttaa graafisen käyttöliittymän, joka ei kuitenkaan ole täysin valmis. Sillä on simulation-muuttuja, joka pitää sisällään Option-kääreeseen pakatun Simulation-olion. simulationFileLoaded-metodi kertoo ohjelmalle, onko simulaatiotiedosto ladattu. CreateSImulation-metodi luo uuden simulaation ohjelmalle.

ALGORITMIT

Simulaation tärkein algoritmi on ehdottomasti Mathematics-objektin käyttämä neljännen kertaluvun Runge-Kutta-metodi, joka toimii koko simulaation perustana. Tiivistettynä Mathematics-luokan metodi evaluate käyttää muita luokan apumetodeita ja laskee approksimaation kaikkien simulaation kappaleiden seuraavalle sijainnille.

$$y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

for n = 0, 1, 2, 3, ..., using[3]

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$

Koska laskenta tapahtuu kolmiulotteisessa avaruudessa, ei laskenta ole ihan yhtä yksinkertaista kuin kaavasta näkee, mutta periaate on sama. Yksittäisten arvojen sijaan käytetään kolmiulotteisia vektoreita kuvaamaan nopeutta, sijaintia ja kiihtyvyyttä. Ohjelmassa approksimoidaan kappaleiden sijaintia käyttämällä Newtonin gravitaatioteoriaa.

$$F = \gamma \frac{m_1 m_2}{d^2}$$

Kaavassa F on kappaleisiin kohdistuva voima, m_1 ja m_2 ovat kahden eri kappaleen massat, d niiden välinen etäisyys ja γ gravitaatiovakio.

Törmäyksiä ohjelma tarkastaa katsomalla kappaleiden säteitä ja niiden välistä etäisyyttä. Jos kappaleen säde on suurempi kuin sen etäisyys toisesta kappaleesta, on tapahtunut törmäys.

TIETORAKENTEET

Oleellisimmat tietorakenteet on toteutettu Scalan valmiilla toiminnoilla, mutta vektoreiden kuvaamiseen käytetään luotua Vector3D-tietorakennetta. Vector3D-rakenne on muuttumaton kolmiulotteisen vektorin kuvaamiseen tarkoitettu tietorakenne. Se tukee tavallisimpia vektoreiden laskutoimituksia. Vector3D-rakenne sisältää kolme arvoa, x-, y-, ja z-komponenteille, jotka ovat Double-arvoja.

TIEDOSTOT

Ohjelma käsittelee kahdenlaisia tiedostoja. Näistä ensimmäinen on simulaation ajamiseen vaadittava tiedosto, joka sisältää tiedot avaruuskappaleiden nimistä, sijainneista, nopeuksista, massoista ja säteistä. Tiedoston rakenne on seuraava:

nimi; massa; nopeus x, y, z; sijainti x, y, z; säde

Eri kappaleet tulee lisätä eri riveille ja tietojen tulee olla oikeassa järjestyksessä. Eri arvot erotetaan ”;”-merkillä ja arvon sisäiset komponentit (nopeus, paikka) erotetaan pilkulla ”,”.

Arvot voi ilmoittaa kokonais, - tai desimaalilukuina. Desimaalierottimena käytetään pistettä ”.”. Projekti sisältää test-osion resources-kansiossa tiedoston testfile.txt, joka on toimiva mallitiedosto.

Tiedostoon voi lisätä kommentteja käyttämällä ”#”-merkkiä. Kommentin voi lisätä joko uudelle riville tai arvojen perään. Ohjelma lopettaa rivin lukemisen heti merkin löydettyään.

Toinen ohjelmaan liittyvä tiedosto on Gnuplot-ohjelmalla luettava datatiedosto. Tämän tiedoston pystyy ainoastaan luomaan tekstipohjaisella käyttöliittymällä.

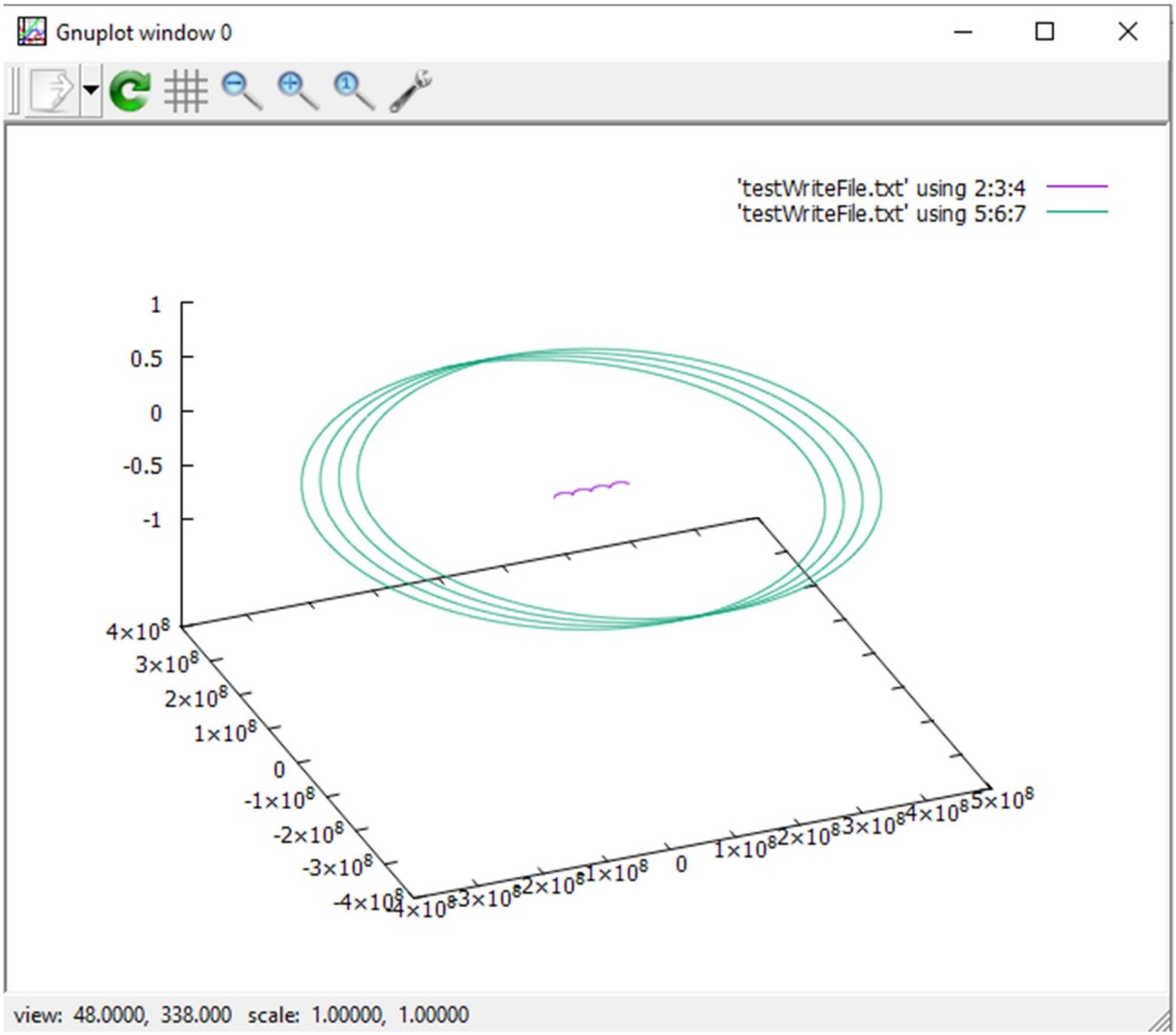
Tiedosto koostuu riveistä, joista jokainen sisältää seuraavat asiat:

aika x_1 y_1 z_1 x_2 y_2 z_2 …

Eli tiedosto sisältää simulaation alusta kuluneen ajan ja kaikkien kappaleiden sijainnit. Kappaleet on sijoiteltu siten, että ne ovat samassa järjestyksessä, kuin simulaatiotiedostossa. Projektin test-kansion resources-osiossa löytyy tiedosto testwrite.dat, joka on esimerkki kirjoitetusta tiedostosta.

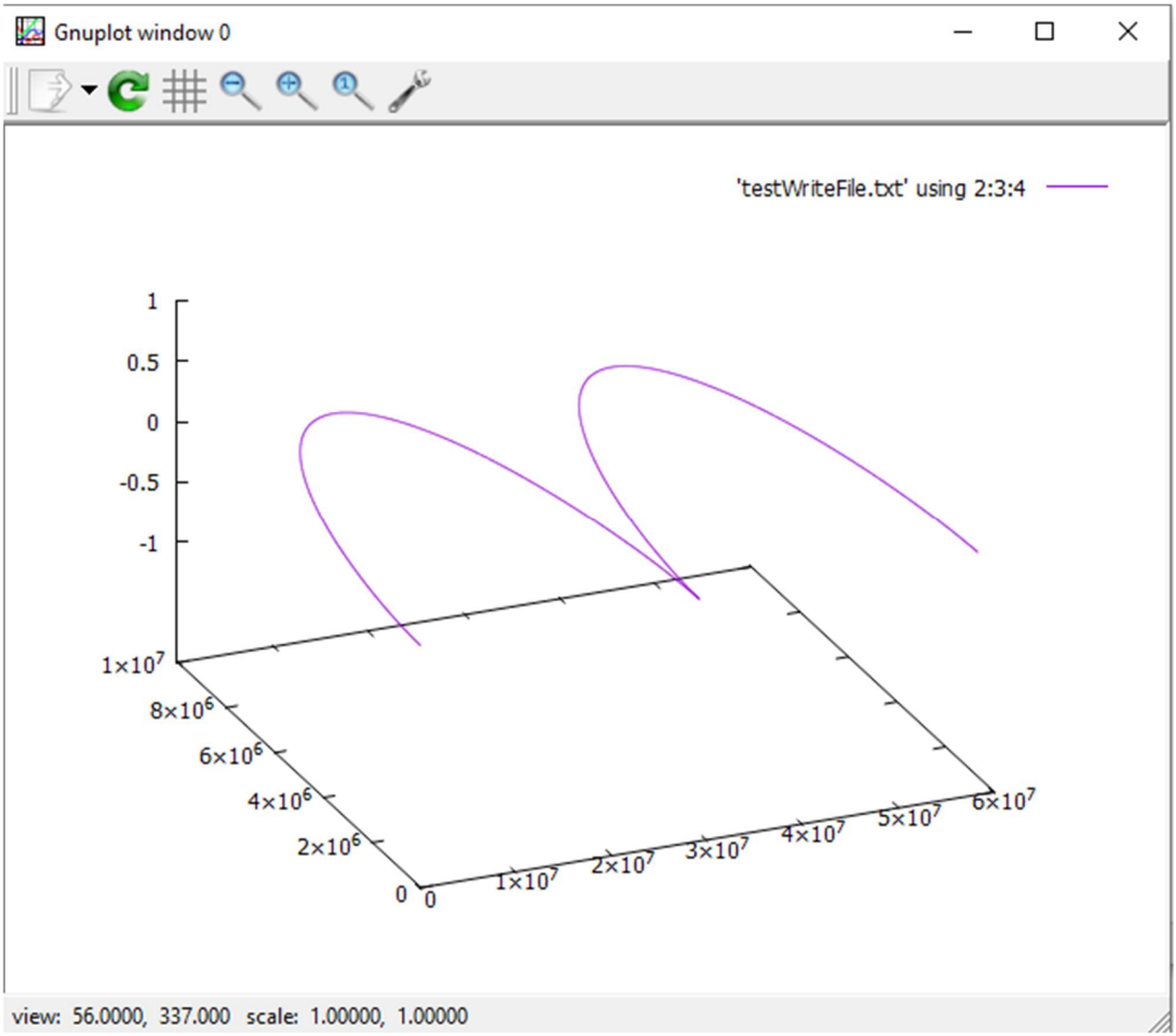
TESTAUS

Suunnitelmaan kuului alun perin yksikkötestauksia, joissa laskettaisiin kappaleiden sijainnit tietyn ajan päästä ja vertailtaisiin oikeisiin. Tämä kuitenkin osoittautui kovin työlääksi, joten ohjelman testaus koostui lähinnä datan laskemisesta ja sen sijoittamisesta kuvaajaan.



Kuvassa on testidatasta muodostettu kuvaaja, joka esittää aurinkoa (pinkki) ja kuuta (vihreä). Testausaika oli noin 89 päivää eli kolme kuunkiertoa.

Testissä havaitsin, että aurinko tekee hieman omituista liikettä. Seuraava kuva on samasta datasarjasta, mutta kuun data on jätetty pois, jotta saadaan auringon liike paremmin näkyviin.



Kuvasta huomataan, että aurinko tekee siniaallon itseisarvon muotoista liikettä. En lopulta saanut selville, mistä tämä johtuu, mutta simulaatio tuntuu käyttäytyvät hieman omituisesti silloin, kun kappaleiden massaerot ovat suuret.

Testausta on myös tehty esimerkiksi tiedoston kirjoittamisessa ja lukemisessa, jotka pääasiassa toteutettiin luomalla esimerkkitiedostoja ja lukemalla ne ohjelmalla. Tämä jälkeen vertailtiin odotettuja arvoja ja luettuja arvoja.

OHJELMAN TUNNETUT PUUTTEET JA VIAT

Aiemmassa kappaleessa mainitun vian lisäksi graafinen käyttöliittymä on varsin puutteellinen, eikä täytä tehtävänannon kriteereitä. Tämä johtui yksinkertaisesti ajan loppumisesta.

Graafisesta käyttöliittymästä puuttuu kätevä toiminnallisuus, sillä tällä hetkellä sitä voi ajaa ainoastaan manuaalisesti klikkailemalla. Myös tiedostoon tallentaminen puuttuu.

3 PARASTA JA 3 HEIKOINTA KOHTAA

Parhaat:

Mielestäni ohjelman toiminnallisuus on toteutettu hyvin. Ohjelma pystyy laskemaan halutut tiedot tehokkaasti käyttämällä Runge-Kutta-menetelmää.

Vaikka graafinen käyttöliittymä jäikin pahasti kesken, on se mielestäni silti esteettisesti miellyttävä, vaikka melko epäkäytännöllinen onkin. Se myös toimii hyvänä testausvälineenä.

Mielestäni koodi on melko hyvin luettavaa ja selkeää (pois lukien graafinen käyttöliittymä). Epäselvät kohdat on mielestäni hyvin kommentoitu.

Heikot:

Ohjelma jäi melko yksinkertaiseksi. Se tekee sen mitä pyydettiin, mutta ei juuri mitään enempää. käyttäjälle olisi voinut antaa enemmän valtuuksia päättää eri asioita.

Graafinen käyttöliittymä jäi kesken ja sen koodi on melko huonosti rakennettu. En ollut ennen tehnyt mitään käyttöliittymään liittyvää, joten asiaa opetellessa koodista tuli pirstaloitunutta ja huonosti luettavaa.

Testaus oli hankalaa. Käytin paljon aikaa testaukseen, mutta silti tuntuu, etten ehtinyt testata kaikkea. Oli todella aikaa vievää tutkia käyriä ja pohtia, kuvaavatko ne todellisuutta.

POIKKEAMAT SUUNNITELMASTA, TOTEUTUNUT TYÖJÄRJESTYS JA AIKATAULU

Toteutusaikataulu poikkesi aika reippaasti suunnitellusta. Esimerkiksi matemaattisen ytimen luomiseen meni enemmän aikaa kuin mitä olin odottanut. Suurimpia muutoksia kuitenkin aiheuttivat ulkoiset tekijät, joita en ollut osannut ottaa huomioon arvioinnissa. Ensimmäinen 2 viikkoa meni Mathematics-luokan tekemiseen ja seuraava viikko sen testaamiseen.

Tämän jälkeen toiminnallisuuden ja tekstikäyttöliittymän luomiseen meni noin 3 viikkoa. Graafiselle käyttöliittymälle jäi paljon vähemmän aikaa kuin olin suunnitellut. Projektin toteutusjärjestys oli kuitenkin suunnitelman mukainen.

KOKONAISARVIO LOPPUTULOKSESTA

Kokonaisuudessaan uskon, että projekti toteutui melko hyvin. Olisin halunnut tehdä työn vaikeana, mutta aikarajoitusten vuoksi jouduin jättämään työn kesken. Olen kuitenkin tyytyväinen siihen, mitä kaikkea olen projektin kautta oppinut. Sen aikana olen kokenut useita ahaa-elämyksiä, mikä on mielestäni kaikista tärkeintä.

Luokkarakenne jäi mielestäni hieman sekaiseksi varsinkin käyttöliittymien osalta. Olisin myös tahtonut toteuttaa ohjelmaan rinnakkaisohjelmointia, jolloin se olisi ehkä toiminut tehokkaammin.

VIITTEET

https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods

https://gafferongames.com/post/integration_basics/

http://www.scalafx.org/api/8.0/#package

LIITTEET

Testitiedosto ohjelman ajamiseen löytyy sijainnista ”test\resources\testfile.txt”

Kuvankaappauksia löytyy käyttöohje-kohdasta.