Aurinkokuntasimulaattori – projektityön dokumentti

Mikko Pohjalainen 354390
Tietotekniikan koulutusohjelma 2013
7.5.2015

Projektin yleiskuvaus

Projektini aiheena oli aurinkokuntasimulaattori. Tarkoituksena oli toteuttaa simulaatio, jossa kuvataan aurinkokuntaa ja kiertoradoilla olevia taivaankappaleita. Ohjelma pystyy lataamaan asetustiedostoista taivaankappaleiden alkuasetukset ja sen jälkeen simuloimaan niitä kolmiulotteisessa avaruudessa. Käyttäjällä on mahdollisuus lähettää käyttöliittymän kautta taivaankappaleita liikkeelle ja tutkia, miten painovoima vaikuttaa niihin. Toteutin projektin keskivaikeaa hieman laajempana monipuolisen graafisen käyttöliittymän kanssa.

Käyttöohje

Ohjelma käynnistetään SimulaatioNäkymä-luokasta, joka avaa käyttäjälle aluksi valintaruudun, jossa voi valita simulaatioikkunan alkuperäisen resoluution. Resoluutiota pystyy muuttamaan myös ajon aikana. Kun ohjelma on käynnistynyt, se lataa automaattisesti kaikki Presets-kansiossa sijaitsevat asetustiedostot ja tekee niistä listan. Näistä tiedostoista yhden ennalta määrätyn mukaan se asettaa kappaleet ruudulle. Jos ohjelma ei löydä tätä tiedostoa, avaa se sen sijaan käyttäjälle tyhjän simulaatioruudun. Käytettävää asetustiedostoa voi vaihtaa milloin vain File-valikosta löytyvästä Load-pudotuslistasta. File-valikosta löytyy myös Exit-nappi, jolla ohjelmasta voi halutessaan poistua. Valikkopalkista löytyy myös Settings-valikko, jossa käyttäjä voi muuttaa lähetettävien kappaleiden tyyppiä ja muutamaa grafiikka-asetusta. Valikon alapuolella olevassa työkalupalkissa on vihreä Start-nappi, josta simulaatio käynnistyy. Kun simulaatio on käynnistetty, muuttuu sama nappi punaiseksi Pause-napiksi, josta simulaation voi keskeyttää. Seuraavaksi työkalupalkissa on kolme View-nappia, joita painamalla voi vaihtaa kuvakulmaa, josta simulaatiota katsotaan. Näiden lisäksi käyttöliittymässä on vielä kaksi liukusäädintä. Time factor-liukusäätimellä voi säätää kuinka monta askelta simulaatiossa otetaan kerralla eteenpäin. Tämä säädin on rajoitettu välille 0-10. Mass-liukusäätimellä voi muuttaa käyttäjän lähettämien kappaleiden massaa. Tämä säädin on rajoitettu välille 1-100.

Ohjelman rakenne

Ohjelma koostuu kolmesta pääluokasta ja muutamasta pienestä apuluokasta. Pääluokista ensimmäinen on LiikkuvaMassa-luokka, joka kuvaa simulaatioon asetettavaa kappaletta. LiikkuvaMassa-olio pitää koko aika tiedossaan sen nykyistä paikkaa sekä sen hetkellistä nopeutta simulaatioavaruudessa. LiikkuvaMassa-luokassa on metodit muun muassa simulaatiokappaleiden välisten etäisyyksien ja voimien laskemiseen. Näiden metodien avulla se kykenee laskemaan itselleen uuden paikan ja nopeuden, kun simulaatio päivitetään. Toinen ohjelman keskeinen osa on Simulaatio-luokka. Se pitää muistissaan kaikki simulaatiossa olevat LiikkuvaMassa-oliot ja käskee ne päivittämään itsensä kun ohjelma on ajossa. Se myös tarkkailee koko aika simulaation tilaa ja tarkistaa muun muassa onko kappaleita törmännyt toisiinsa tai mennyt simulointirajojen ulkopuolelle. Simulaatio-luokassa on myös tiedostonlukijametodi, joka pystyy lukemaan ulkoisten tiedostojen datan ja lisäämään niiden perusteella LiikkuvaMassa-olioita simulaatioon.
Kolmas keskeinen luokka ohjelmassa on SimulaatioNäkymä, joka vastaa grafiikan piirtämisestä näytölle Simulaatio-olion tietojen perusteella sekä käyttöliittymän luomisesta. SimulaatioNäkymän tärkein osa on SimulaatioRuutu-paneeli, jossa on määritelty piirtometodit sekä run-metodi, joka käskee pyöriessään Simulaatio-oliota päivittämään ja piirtää tämän jälkeen simulaation kappaleet uudestaan ruudulle. Tämän SimulaatioRuutu tekee 60 kertaa sekunnissa.

Ohjelman tarvitsemia apuluokkia ovat Nopeus, AvaruudenPiste sekä Voima-case luokat, jotka koostuvat kukin kolmesta desimaaliluvusta. Näiden lisäksi on vielä Tyyppi-enumeraatioluokka, jossa on eritelty LiikkuvaMassa-olioiden tyypit.

Algoritmit

Ohjelma toimii melko pitkälti raa'alla voimalla ja päivittää kunkin taivaankappaleen tiedot yksitellen. Kappaleiden nopeuden ja paikan päivittäminen tapahtuu Eulerin-menetelmällä, jossa pidetään muistissa aina edellistä paikan ja nopeuden arvoa ja lasketaan klassisen fysiikan kaavojen perusteella uudet arvot. Keskeiset fysiikan kaavat, joita ohjelma hyödyntää, ovat: Newtonin toinen laki ($F = ma$), gravitaatiovoiman laki ($F = \text{gravitaatiovakio} * m_1 * m_2 / r^2$) sekä kappaleen paikan kaava ($x = x_0 + v_0t + 0.5at^2$). Ohjelmalla simulaatiota voi tarkastella kolmesta eri suunnasta ja ohjelmaan on pyritty luomaan kolmiulotteisuutta piirtämällä kappaleet näytölle järjestyksessä kauimmasta lähimpään sekä skaalaamalla piirrettäviä kappaleita eksponenttifunktion mukaan, jossa eksponentti määräytyy syvyysakselin mukaan.

Tietorakenteet

Ohjelma tallentaa tietoa muun muassa simulaatiossa mukana olevista kappaleista, törmäyksistä, jotka aiheuttavat räjähdyksen sekä tiedostonimistä jotka löytyvät Presets-kansiosta. Simulaation kappaleet tallennetaan muuttuvaan Buffer-tietorakenteeseen, joka mahdollistaa helpon kappaleiden lisäämisen ja poistamisen. Törmäyksistä tieto tallennetaan Queue-tietorakenteeseen. Jono on tähän tarkoitukseen hyvä,

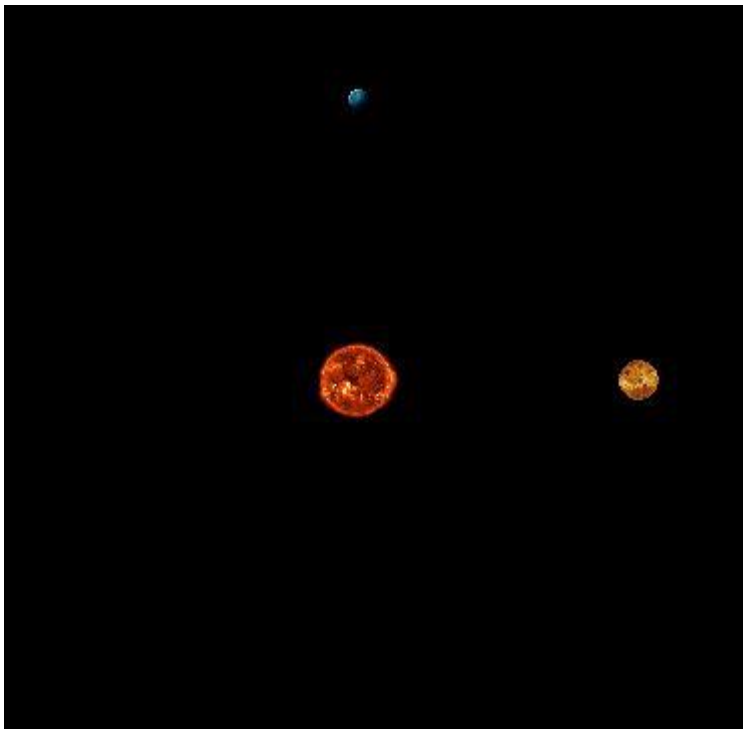
sillä räjähdysanimaatio kestää aina yhtä kauan, ja näin ollen räjähdystapahtumat lisätään ja poistetaan saapumisjärjestyksessä. Presets-kansiossa olevat tiedostonimet tallennetaan Lista-tietorakenteeseen, vaikkakin tähän olisi kelvannut melkein mikä tahansa muukin.

Tiedostot

Ohjelman alkuasetelman tiedot tallennetaan tekstitiedostoon, jossa yhden taivaankappaleen lähtötietoja kuvataan yhdellä rivillä. Rivi koostuu neljästä solusta, jotka on jaettu puolipilkulla. Soluja ovat T, M, P ja V, jotka tarkoittavat järjestyksessä tyyppiä, massaa, paikkaa ja nopeutta. Soluihin kirjoitetaan aina jokin kirjaimista etuliitteeksi ja tämän perään arvo. Välilyönneillä ja kirjainkoolla ei ole merkitystä tiedoston toimivuuden kannalta. T-solun arvoja voivat olla ”satellite”, ”small planet”, ”planet”, ”large planet” tai ”star” kirjoitettuna ilman heittomerkkejä. M-solun arvo voi olla mikä tahansa desimaaliluku, mutta tiedostonlukija rajoittaa arvot välille 1-500. P-solun arvo on kolme toisistaan pilkulla erotettua desimaaleina olevaa koordinaattia. Simulaatiossa ruudun keskipiste on piste(0,0,0) ja etuperspektiivistä katsottuna positiivinen y-suunta on ylöspäin, positiivinen x-suunta on oikealle ja positiivinen z-suunta on katsojaan päin. V-solussa pätee sama muoto kuin P-solussa. Kolme oikeaoppista tiedoston riviä saattaisi näyttää tältä:

T star; M 100; P 0,0,0; V 0,0,0
T planet; M 4; P 0,140,0; V 2.4,0,0
T large planet; M 10; P 140,0,0; V 0,-2.4,0

Tämä tiedosto piirtyisi näytölle alla olevalla tavalla etuperspektiivissä (Aurinko on pisteessä (0,0,0).



Testaus

Luokille LiikkuvaMassa ja Simulaatio on kirjoitettu yksikkötestejä niiden keskeisille metodeille. LiikkuvaMassa-luokassa on muun muassa tarkistettu, että kappaleiden väliset etäisyydet ja voimat lasketaan oikealla tavalla. On myös tarkistettu, että kappale tunnistaa milloin se on tuhoutunut ja tulisi poistaa simulaatiosta. Simulaatio-luokassa on testattu muun muassa metodi, joka tarkistaa voiko kappaleen lisätä simulaatioon sekä se, että simulaatio poistaa tuhoutuneet tai rajojen yli menneet kappaleet simulaatiosta. Näiden lisäksi on tarkistettu myös, että tiedostonlukumetodi toimii oikein. Kaikki suunnitelmassa esitetyt testit menevät läpi. SimulaatioNäkymä-luokkaa on testattu vain manuaalisesti, sillä en ehtinyt perehtyä Scalan käyttöliittymätestaukseen.

Ohjelman tunnetut puutteet ja viat

Ohjelma toimii pääasiassa haluamallani tavalla, mutta parantamisen varaa jäi varsinkin laskentatarkkuuteen ja tehokkuuteen. Myös simulaatiossa tapahtuvat räjähdykset piirtyvät aina päällimmäiseksi, minkä seurauksena kappaleen takana tapahtuva räjähdys peittää sen simulaatiossa. Tämä johtuu tavasta, jolla räjähdysten piirtäminen käsitellään eri kohdassa kuin kappaleiden piirtäminen. Räjähdykset tosin ovat vain lisäominaisuus, jonka pystyy käyttöliittymässä halutessaan poistamaan käytöstä.

Parhaat ja heikot kohdat

Ohjelmani yksi parhaista puolista on käytettävyyden helppous käyttöliittymän selkeyden ja yksinkertaisen tiedostoformaatin ansiosta. Simulaation aikana käytettävissä olevat työkalut, kuten kuvakulman muuttaminen, simulaation nopeuden muuttaminen sekä valinnat räjähdysten ja kappaleiden polkujen piirtämiselle antavat käyttäjälle paljon mahdollisuuksia vaikuttaa. Käyttäjän kannalta on myös viihdyttävää, että pelkkien satelliittien sijasta pystyy lähettämään kolmea erikokoista planeettaa sekä tähtiä. Heikkona kohtana ohjelmassa mainittakoon, että suorituskyky saattaa koneesta riippuen kärsiä suurista yhtäaikaisista kappalemääristä simulaatiossa varsinkin, kun räjähdykset on asetettu päälle ja kappaleita törmää tiheään tahtiin.

Poikkeamat suunnitelmassa

Toteutin projektin osat melko lailla suunnitellussa järjestyksessä. Aikaakin projektiin kului yhteensä suunnilleen saman verran kuin olin suunnitellut, vaikka toteutin hieman enemmän ominaisuuksia kuin olin alun perin suunnitellut.

Toteutunut työjärjestys ja aikataulu

Koska minulla oli tämän kurssin kanssa samaa aikaa eräs toinenkin ohjelmointiprojektikurssi, joka osoittautui melko työlääksi, ei minulla ollut tämän kurssin alku- eikä keskivaiheessa juuri ollenkaan aikaa

edistää projektia. Tein aivan alussa ohjelman keskeiset komponentit valmiiksi, mutta vasta viimeisten viikkojen aikana toteutin kaikki ohjelmani työläät osat.

Arvio Lopputuloksesta

Vaikka aika kävi lopussa tiukoille enkä ehtinyt parantamaan ohjelmani suorituskykyä, olen tyytyväinen projektini lopputulokseen. Sain toteutettua mielestäni omaan tasooni nähden hyvin kaikki haluamani ominaisuudet. Ohjelma ei ole tieteellisen tarkka, mutta sillä on hauska kokeilla, miten kappaleet liikkuvat painovoiman lakien mukaan ja katsoa, kuinka ne törmäävät räjähtäen toisiinsa. Jos projektia jatkaisi, voisi ensisijaisesti parantaa vaihtamalla Eulerin-menetelmästä tarkempaan numeeriseen integrointimenetelmään. Tämän lisäksi voisi vielä lisätä käyttöliittymään enemmän asetettavia kappaleita ja grafiikkaominaisuuksia.

Viitteet:

http://www.scala-lang.org/api/current/#package

https://greengoblin.cs.hut.fi/o1_s2014/

Introduction to the Art of Programming Using Scala – Mark C. Lewis