Aurinkokuntasimulaattorin tekninen suunnitelma

Lauri Koskenniemi
652416
Elektroniikka ja sähkötekniikka
vuosikurssi 1
28.2.2018

Ohjelman Test-luokassa kutsutaan System-luokkaa, joka muodostaa Aurinkokunnan. GUI-luokka piirtää System-luokan oliot ObjectGraphics-luokan avulla näytölle, sekä käyttöliittymän, jolla käyttäjä voi muun muassa lisätä satelliitteja. System-luokka kutsuu Object-luokkaa luodessaan uusia olioita, sekä saadakseen tietoja, kuten sijainti ja massa, Aurinkokunnassa olevista kappaleista. System-luokka käyttää myös Physics-, Vector-, Coordinates- ja FileReader-luokkia. Physics-luokassa mallinnetaan simulaatiossa tarvittavia fysiikan ilmiöitä, kuten gravitaatiota ja Newtonin lakeja. Vector-luokassa määritellään vektoreiden laskukaavoja. Coordinates-luokassa muutetaan muun muassa käyttäjän antama sijainti karteesiseen koordinaatistoon. FileReader-luokka lukee tekstitiedostoa, jossa on alkuarvoja System-luokassa määriteltäville olioille. (UML-kaavio GitLab-projektikansiossa.)

Simulaattorin käynnistyessä ohjelma lukee FileReader-luokassa tiedostosta Aurinkokunnan pääelementit, jotka luodaan Object-luokassa ja kutsutaan System-luokassa. GUI-luokka piirtää oliot näytölle ObjectGraphics-luokan avulla. Tässä kohdassa käyttäjä voi lisätä satelliitin alkuarvoilla alareunassa olevassa kentässä, sekä määritellä simuloitavan ajanjakson. Coordinates-luokka muuntaa käyttäjän antamat arvot xyz-tasoon. Kun simulaatio alkaa, System-luokka päivittää olioiden sijainnit Physics-luokan kaavoilla, jossa käytetään apuna myös Vector-luokkaa laskennassa. System-luokka päivittää olioiden sijainteja, kunnes käyttäjän antama ajanjakso täyttyy. Käyttäjä voi vaihtaa kuvakulmaa simulaation aikana, mikä tapahtuu GUI-luokassa.

Simulaattorissa laskemisessa vektoreita. Olio sijaitsee xyz-koordinaatistossa, jolloin sijaintivektori on kolmialkioinen lista. Tällöin ohjelmassa tarvitaan muun muassa vektorien

yhteenlaskua, jossa listojen vastaavat alkiot lisätään yhteen, ja vektorin pituuden määrittäminen, jossa vektorin toiseen korotetut alkiot summataan ja sijoitetaan neliöjuureen. Simulaatiossa lasketaan kahden olion välinen gravitaatiovoima kaavalla:

$$F = G\frac{m_1 m_2}{r^2}$$

Kaavasta saatava F on voiman suuruus, joten voiman suunta täytyy laskea jakamalla sijaintien välinen vektori komponentteihin. Voimavektori saadaan muodostettua kertomalla voiman suuruus suuntavektorin yksikkövektorilla. Voimasta saadaan olion kiihtyvyys Newtonin toisella lailla jakamalla voiman komponentit kappaleen massalla. Nopeuden muutos saadaan kiihtyvyydestä ja sijainnin muutos nopeudesta kaavoilla:

$$dv = a * dt$$

$$dx = v * dt$$

Tällöin paikka ja nopeus saadaan lisäämällä muutos alkuperäiseen arvoon.

Ohjelmassa käytetään listoja tietorakenteena. Taivaankappaleiden ja satelliittien määrä tallennetaan listaan. Tiedon varastoimiseen riittää yksiulotteinen lista ja olioiden määrä on kohtuullisen pieni ja koko voi muuttua ohjelman suorituksen aikana. Kappaleiden tiedot, kuten massa, nopeus ja sijainti, on tallennettu joko muuttujiin tai listoihin vektorimuotoisena. Myös nämä arvot voivat muuttua jatkuvasti simulaation aikana.

Vaativimmat luokat toteuttaa ovat GUI ja System. Molempiin kuluu noin kymmenen tuntia. Muiden luokkien tekemisessä kestää noin 2-5 tuntia. Aloitan ohjelmoinnin GUI ja System luokista, koska ne ovat ohjelman toimimisen kannalta merkittävimmät luokat. Seuraavaksi kannattaa tehdä Object, Vector ja Physics luokat, jotta voin mallintaa kappaleiden liikettä. Kun GUI-luokka piirtää realistisesti taivaankappaleen liikettä, voin alkaa lisäämään muita käyttöliittymään tarvittavia toimintoja.

Test-luokassa ajetaan ohjelma ja käydään läpi yksikkötestausta. GUI-luokassa testataan reagoiko ohjelma oikein käyttäjän antamiin arvoihin, esimerkiksi virheellisiin arvoihin. System-luokassa tulisi testata muodostaako se Aurinkokunnan oikein annetuilla taivaankappaleilla. Physics, Vector ja Coordinates luokissa täytyy testata antaako metodit oikeita arvoja eri alkuarvoilla. Object-luokkaa voi testata antamalla sille sijainti- ja nopeusarvoja ja kokeilemalla liikkuuko olio haluttuun suuntaan. Myös FileReader:n toiminta tulisi testata mahdollisilla virheellisillä tiedostoilla.

Simulaattorin taustalla olevaan fysiikkaan olen käyttänyt lähteinä Integration Basics (https://gafferongames.com/post/integration_basics/) ja The Nature of Code (http://natureofcode.com/book/) sivuja. Graafisen käyttöliittymän tekemiseen käytän Qt:n dokumentaatiota (http://doc.qt.io/). Koordinaatistoon suunnitteluun ja planeettojen lähtöarvojen määrittelyyn olen käyttänyt sivua Helicentric Trajectories of Selected Planets (https://omniweb.gsfc.nasa.gov/coho/helios/planet.html).

UML-kaavio GitLab-projektikansiossa