# Tekninen suunnitelma

Aurinkokuntasimulaattori, Aleksi Korsman, 544126, elektroniikka ja sähkötekniikka,
2. vuosikurssi, 27.2.2018

## Ohjelman rakennesuunnitelma

UML-kaavio löytyy samasta kansiosta kuin tämä suunnitelma. Ohjelma jakautuu selvästi kahteen eri lohkoon: Aurinkokunta ja siitä ylöspäin lähtevät haarat, sekä Simulaatio ja siitä lähtevät haarat. Näiden ohella on tietysti Käyttöliittymä, joka kutsuu simulaatiota.

Aurinkokunta:
Aurinkokuntaan kuuluu yksi Aurinko, lista planeetoista ja lista satelliiteista. Auringolla ja planeetoilla on oma halkaisija, mutta satelliitit oletetaan niin pieniksi, että niiden kokoa ei oteta huomioon. Satelliitilla taas on tila, eli satelliitti voi olla ehjä tai tuhoutunut. Satelliitti tuhoutuu, jos se törmää johonkin planeettaan tai Aurinkoon.

Jokainen edellä mainituista on taivaankappale, jolla on massa, nimi ja rata. Rata koostuu paikasta ja nopeudesta, jotka molemmat ilmoitetaan paikkavektorina. Koordinaatiston origo on Auringon keskipiste.

Simulaatio:
Simulaatio sisältää aurinkokunnan, simulaatiojakson pituuden, simulaatioaskeleen pituuden, kuluneen ajan ja laskentamekanismin. Jakson, askeleen ja kuluneen ajan pituudet ilmoitetaan Aika-luokalla, jossa aika voidaan ilmoittaa useassa eri muodossa. Laskentamekanismina käytetään Runge-Kutta-menetelmää, jossa on omat metodit uusien parametrien laskemiseksi. Näiden toiminta selitetään myöhemmin tässä dokumentissa.

Käyttöliittymä:
Käyttöliittymä-luokka sisältää tarvittavat metodit, jotta voidaan lukea asetustiedosto, lukea käyttäjältä arvot satelliiteille, ja aloittaa simulointi.

## Käyttötapauskuvaus

Ideaalinen käyttötapaus menisi seuraavasti:

1. Käyttäjän avatessa ohjelman kysytään tiedostonimi, jossa on planeettojen ja Auringon parametrit yleissuunnitelmassa esitellyllä tavalla.
2. Käyttäjältä kysytään satelliittien parametrit yleissuunnitelmassa esitellyllä tavalla.
3. Käyttäjä valitsee simuloinnin ja syöttää järkevät simulointijakson ja simulointiaskeleen pituudet
4. Ohjelma tulostaa satelliittien parametrit jokaisen simulointiaskeleen jälkeen.

5. Käyttäjä voi poistaa tai lisätä satelliitteja ennen uutta simulointia, tai sitten käyttäjä poistuu ohjelmasta.

# Algoritmit

Ainoa tehtävään liittyvä algoritmi on taivaankappaleiden uusien parametrien laskeminen. Tähän käytetään Runge-Kutta-menetelmää, joka on neljännen asteen integraattori. Runge-Kutta-menetelmää käytetään seuraavasti:

1. Tehdään kaksi eri tilamuuttujaparia: $y_{n,1} = \begin{bmatrix} y \\ \dot{y}_1 \end{bmatrix}$ ja $y_{n,2} = \dot{y}_{n,1} = \begin{bmatrix} \dot{y}_2 \\ \ddot{y} \end{bmatrix}$, joissa jokainen muuttuja on vektori, joka sisältää x-, y-, ja z-komponentit. Syötetään alkupiste $y$ ja alkunopeus $\dot{y}_1$.
2. Lasketaan seuraava askel, eli uudet tilat molempien muuttujaparien jokaiselle alkioille. Yhden askeleen aikana $y_{n,1}$ pysyy vakiona, kun taas $\dot{y}_{n,1}$ päivittyy, ja se annetaan aina eteenpäin seuraavalle k-arvolle:

   $\dot{y}_{n,1}(t) = f(t, y),\ y_{n,1}(t_0) = y_0$

   $y_{n+1,1} = y_{n,1} + \frac{h}{6}(k_1 + 2(k_2 + k_3) + k_4)$, jossa

   $k_1 = f(t_n, y_{n,1})$
   $k_2 = f(t_n + \frac{h}{2}, y_{n,1} + h\frac{k_1}{2})$
   $k_3 = f(t_n + \frac{h}{2}, y_{n,1} + h\frac{k_2}{2})$
   $k_4 = f(t_n + h, y_{n,1} + hk_3)$

   Funktio kutsutaan erikseen $\dot{y}_{n,1}$ :n molemmille muuttujille. Paikan muutoksen $\dot{y}_2$ päivitysfunktio on yksinkertaisesti $\dot{y}_2 = \dot{y}_1 + \ddot{y}h$. Kiihtyvyyden päivitysfunktio saadaan Newtonin 2. laista ja gravitaatiolaista: $F = m_1 a = \gamma\frac{m_1 m_2}{r^2} \Leftrightarrow a = \gamma\frac{m_2}{r^2}$, eli $\ddot{y} = \gamma\frac{m_2}{r^2}$, jossa $\gamma$ on gravitaatiovakio, $m_1$ satelliitin massa, $m_2$ planeetan tai Auringon massa ja $r$ etäisyys. Lasku suoritetaan jokaiselle komponentille erikseen, ja kokonaiskiihtyvyys määräytyy kaikkien kappaleiden yhteisvaikutuksesta.
3. On saatu seuraava askel $y_{n+1,1}$, joten voidaan ryhtyä laskemaan samalla periaatteella seuraavaa askelta.

# Tietorakenteet

Kaikki numerot talletetaan liukulukuina tarkkuuden säilyttämiseksi. Jonkinlaista tehokkuutta pyritään saamaan tallentamalla eri luvuille arvon lisäksi yksikkö: täten esimerkiksi etäisyyksiä, jotka voivat mennä gigametreihin asti, voidaan helposti laskea ja esittää kymmenpotenssimuodossa. Lisäksi aikaa on tarkoitus voida säilyttää sekunneissa, minuuteissa, tunneissa ja niin edelleen.

Python3:ssa suurin mahdollinen liukuluku on luokkaa 10^308, joten periaatteessa tämä ei ole pakollista. Muistia tämä systeemi säästänee kuitenkin jonkin verran. Arvojen säilytysmuodot hioutuvat vielä lopulliseen muotoonsa projektin edetessä.

Aurinkokunnan taivaankappaleet säilötään listoissa. Periaatteessa listat voisivat olla linkitettyjä, koska alustuksen jälkeen niihin ei lisätä ja niistä ei poisteta mitään, mutta yksinkertaisuuden vuoksi nämä tehtäneen perinteisinä listoina.

# Aikataulu

4.3.2018: UML-kaavion Aurinkokunta-luokasta ylöspäin lähtevät luokat implementoitu, ml. Aurinkokunta. Luokkia testataan.

11.3.2018: Merkkipohjainen käyttöliittymä on valmis.

18.3.2018: Simulaatiosta haarautuvat luokat valmiit, Runge-Kutta-simuloinnin testaus käynnissä.

25.3.2018: Ohjelma pääosin valmis, hienosäätö, virheidenkäsittely ja ankara testaus alkaa.

8.4.2018: Ohjelma on valmis. Dokumentaatiota viimeistellään.

15.4.2018: Dokumentaatio on valmis.

# Yksikkötestaussuunnitelma

Aurinkokuntaan liittyviä luokkia implementoidessa tulee testatessa kiinnittää huomiota siihen, että arvot tulevat tallennetuksi oikeassa muodossa kaikilla mahdollisilla yksiköillä. Myös on tärkeää löytää virheelliset syötteet ja ilmoittaa niistä käyttäjille.

Simulaation tarkka testaus on melko haastavaa, mutta etenkin törmäyksiä ja törmäämättömyyksiä tulee testata. Lisäksi Runge-Kuttan tarkkuutta tulee testata, esimerkiksi säilyykö stabiili kiertorata vai häviääkö energiaa jonnekin.

Käyttöliittymälle on helppo jopa ennen sen mallintamista määritellä muutamia erilaisia syötekombinaatioita ja halutut tulosteet näille.

# Kirjallisuusviitteet ja linkit

Runge-Kuttan menetelmä: https://gafferongames.com/post/integration_basics/, http://lpsa.swarthmore.edu/NumInt/NumIntFourth.html

Python3:n liukuluvun maksimiarvon saa selville kirjoittamalla python3 tulkkiin komennot
>>> import sys
>>> sys.float_info

# Liitteet

UML-malli löytyy samasta kansiosta kuin tämä dokumentaatio.