Aurinkokuntasimulaattori

Aleksi Korsman, 544126, Elektroniikka ja sähkötekniikka, 24.4.2018

Yleiskuvaus

Ohjelma simuloi aurinkokuntaa, jossa on keskipisteenä yksi kappale, jonka suhteen kaikki liike tapahtuu. Lainataan tehtävänantoa:

”Tee ohjelma, joka simuloi taivaankappaleiden ja satelliittien liikkeitä laskemalla niiden ratojen pisteitä, kunnes kappaleiden välillä tapahtuu törmäys tai simuloitava aikajakso kuluu umpeen. Voit jättää huomiotta painovoimaan liittymättömät häiriötekijät (ilma-kehän vastus tms.), mutta simuloi gravitaation ja Newtonin lakien mukaisia liikkeitä mahdollisimman tarkasti.

Perusaurinkokunnan kaikilla elementeillä on nopeus, ja paikka - jopa auringolla, joskin sen alkunopeus voidaan asettaa nollaksi. Muilla kappaleilla täytyy olla järkevät alkuarvot, jotta ne eivät tipu suoraan aurinkoon tai karkaa aurinkokunnasta jne.

Ideana tehtävässä on tarkastella satelliittia/satelliitteja, jotka lähtevät annetuista kolmiulotteisen avaruuden pisteestä annetulla alkunopeudella (voivat olla vaikka matkalla kuuhun, Marsiin tai maan kiertoradalla tms.).

Käyttäjän tulee voida määrätä satelliittien alkupiste, massa ja nopeus, simulaatiojakson ja simulaatioaskelen pituus, sekä mahdollisia muita parametreja, jotka katsot simulaation kannalta oleellisiksi. Taivaankappaleiden alkutilanne voidaan lukea asetustiedostosta.

Kiinnitä huomiota ohjelman käyttöliittymään (yritä tehdä erilaisten lähtötietojen syöttäminen helpoksi ohjelman käytön ja testaamisen kannalta) sekä laskenta-tehokkuuteen. Käytä syötössä ja tulostuksessa todellisia yksiköitä (m, kg,...)”

Tarkoituksena oli tehdä vähintään keskivaikea toteutus, jonka edellytyksinä ovat seuraavat ominaisuudet:

• merkkipohjainen käyttöliittymä

• Esitä kaikkien kappaleiden liikkeet jollakin tavalla. Voit esim. tulostaa koordinaatteja

sellaisessa muodossa, että niistä on helppo tuottaa käyriä UNIXin gnuplot- ohjelmalla (käytä komentoa ”man gnuplot” saadaksesi lisätietoja ohjelmasta).

• Tee laskennasta tarkempaa mutta nopeampaa. (pidempi askel, mutta pienempi virhe) Käytä esim neljännen asteen Runge-Kutta menetelmää. Lyhyt tutoriaali aiheesta löytyy esim täältä [gafferongames.com]. Huomaa että kaikkien kappaleiden tilaa tulee käsitellä yhdessä.

Kaikessa tässä on onnistuttu: ohjelma laskee koordinaatteja käyttäen Runge-Kutta menetelmää, ja tulostaa ne sellaisessa muodossa, että niiden piirtämisessä on helppo käyttää gnuplot-ohjelmaa. Käyttöliittymä on merkkipohjainen, eli vaikeustaso on keskivaikea.

Kansiosta löytyy myös aloitettu graafinen käyttöliittymä, mutta se ei valmistunut deadlineen mennessä.

Käyttöohje

Ohjelma käynnistetään navigoimalla komentopaneelissa ”Skriptit” kansioon ja ajamalla komento ”python3 main.py”. Tämän jälkeen käyttäjältä kysytään planeettatiedoston sijainti. Ohjelman mukana tulee toimiva planeettatiedosto, jota voi käyttää, ja josta voi ottaa mallia omien planeettojen lisäämiseen.

Tämän jälkeen avautuu päävalikko, jossa voi lisätä ja poistaa satelliitteja, vaihtaa planeettatiedostoa, simuloida ja muuttaa simulointiasetuksia.

Satelliittien lisäämisessä käyttäjältä kysytään ensin referenssipiste koordinaateille: täten on helpompi laittaa satelliitti esimerkiksi kiertämään planeettaa. Huomaa, että koordinaatit lasketaan planeetan keskipisteestä, eli jos haluaa laittaa satelliitin vaikka 300km kiertoradalle Maan ympäri, tulee lukemaan summata myös maan säde. Sen jälkeen kysytään satelliitin nimi, massa, paikkavektori ja nopeusvektori. Ohjelma antaa ohjeet lukujen syöttämiselle.

Satelliittien poistamisessa ohjelma tulostaa lisättyjen satelliittien nimet, ja nimen kirjoittamalla satelliitti poistuu.

Planeettatiedoston vaihtaminen tapahtuu samalla tavalla kuin sen lataaminen ohjelmaa käynnistäessä. Kaikki lisätyt satelliitit poistetaan, sillä niiden referenssiplaneetta saattaa poistua.

Ennen simulointia kysytään simuloinnin askel ja jakso. Ohjelma antaa ohjeet lukujen syöttämiselle. Kun luvut on syötetty oikein, ohjelma kysyy käyttäjältä halukkuuden tallentaa tulokset tiedostoon. Oli tallennustiedostoa valittu tai ei, alkaa simulointi, jossa jokaisen aika-askeleen jälkeen tulostetaan koordinaatit komentoriville.

Kun simulointi loppuu, tilanne ei resetoidu. Resetoidakseen tilanteen käyttäjän tulee valita tilanteen resetointi. Tällöin satelliititkin resetoituvat, mutta eivät poistu.

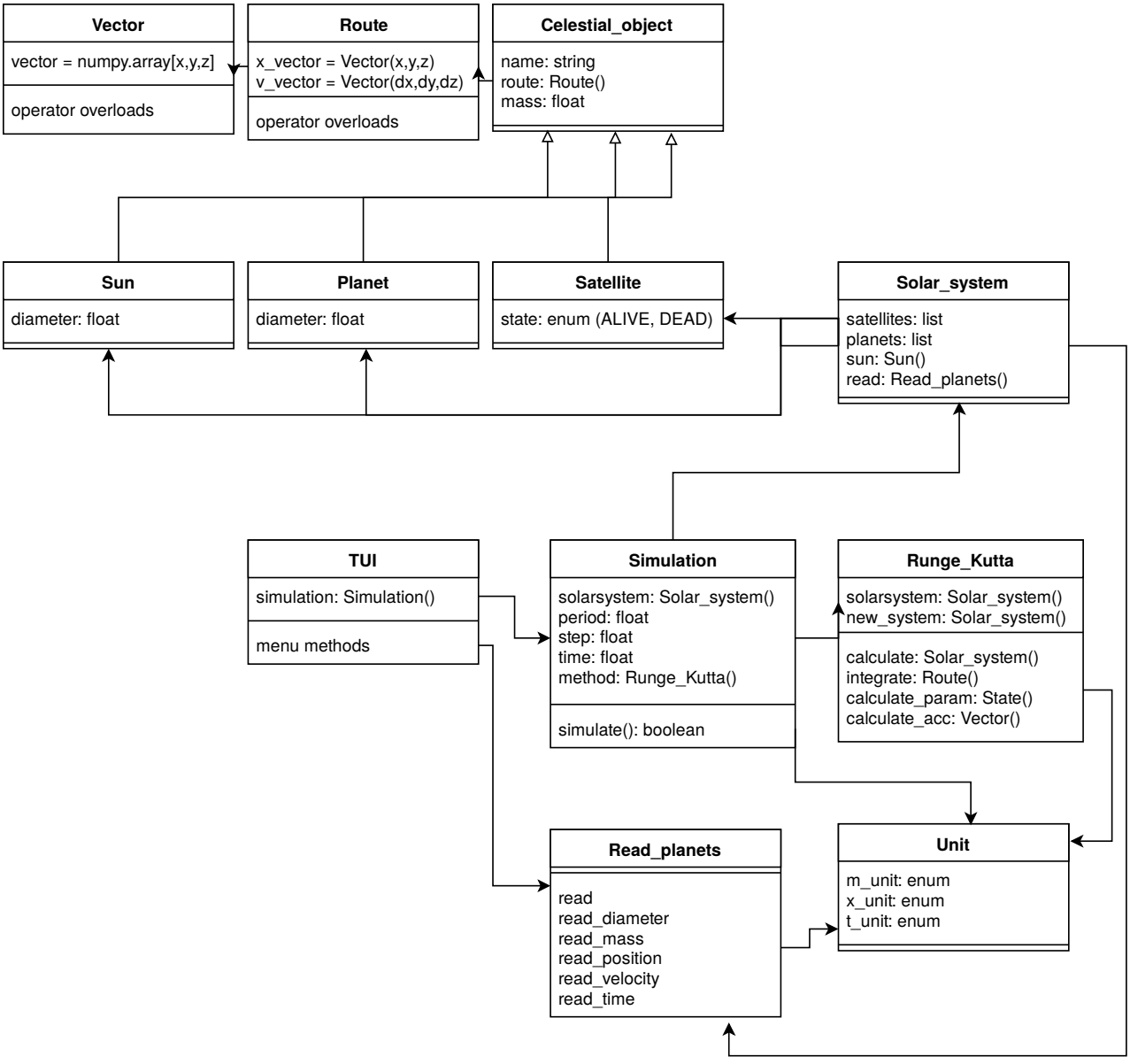
Tallennetut koordinaatit voi helposti tulostaa gnuplot-ohjelmalla, jolle löytyy käyttöohje Readme-tiedostosta.

Ulkoiset kirjastot

Ohjelman ainoa ulkoinen kirjasto on numpy, joka on hyvin kätevä vektorilaskennassa. Graafista käyttöliittymää varten olisi käytetty PyQt-kirjastoa.

Ohjelman rakenne

Alla on ohjelman suuntaa antava UML-kaavio:



Ohjelman kappaleet, eli Aurinko, planeetat ja satelliitit, ovat kaikki kerätty omien luokkiensa lisäksi pääluokkaan Taivaankappale (Celestial_object), johon säilötään niille ominaiset parametrit.

Taivaankappaleiden koordinaatit säilötään Route-luokkaan, jonka parametrit käyttävät Vektori-luokkaa sekä paikka- että nopeusvektorien säilömiseen. Vektori-luokka puolestaan käyttää numpy-kirjastoa, ja suorittaa vektorilaskennat.

Aurinkokunta (Solar_system) koostuu edellä mainittujen taivaankappaleiden lisäksi Read_planets oliosta, joka suorittaa planeettojen lukemisen tekstidatasta.

Simulaatioon (Simulation) sisältyy yksi aurinkokunta, yksi laskentametodi (tässä tapauksessa Runge-Kutta), simulointiaskel ja -jakso sekä simuloitu aika. Itse simulointi tapahtuu kutsumalla simulate()-metodia, joka puolestaan kutsuu Runge-Kutta luokkaa.

Runge-Kutta luokassa tapahtuu yhden simulointiaskeleen laskenta käyttäen Runge-Kutta menetelmää. Kun laskenta on valmis, lähettää se päivitetyn aurinkokunnan Simulaatio-luokalle.

Yksikkö (Unit) luokassa säilötään eri yksiköitä: näiden avulla mikä tahansa saatavilla oleva yksikkö voidaan muuntaa SI-yksiköksi, joilla suoritetaan laskenta.

Itse ohjelman käyttäminen tapahtuu TUI-luokassa, joka tarkoittaa merkkipohjaista käyttöliittymää. Main-funktio kutsuu tätä luokkaa.

Algoritmit

Ohjelma käyttää fysiikoiden laskemiseen Runge-Kutta menetelmää. Runge-Kutta menetelmä kuvailtiin Teknisessä suunnitelmassa näin:

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 γ 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.

Pseudokoodiesitys:

```
laske:
   jokaiselle taivaankappaleelle (pl. Aurinko):
        reitin_muutos = integroi(taivaankappale, askel)
        uusi_reitti = vanha_reitti + reitin_muutos*askel
   päivitä_aurinkokunta

integroi:
    tila = Tila(taivaankappaleen x-vektori ja v-vektori)
    a = laske_parametri(tila, 0*askel, Tila(0-vektorit))
    b = laske_parametri(tila, 0.5*askel, a)
    c = laske_parametri(tila, 0.5*askel, b)
    d = laske_parametri(tila, 1*askel, c)
    x_muutos = 1/6*(a.x + 2*(b.x + c.x) + d.x)
    v_muutos = 1/6*(a.v + 2*(b.v + c.v) + d.v)

laske_parametri(tila, askel, derivaattatila):
    x = tila.x + derivaattatila.x*askel
    v = tila.v + derivaattatila.v*askel
    uusi_tila = Tila(x, v)
    dx = uusi_tila.v
    dv = laske_kiihtyvyys(uusi_tila)
    palauta Tila(dx, dv)

laske_kiihtyvyys(tila):
    jokaiselle planeetalle ja Auringolle (pl. laskettavalle):
        r = vektori kohteesta laskettavaan
        kiihtyvyys = gravitaatiovakio*kohteen_massa/r^2
    palauta Vektori(kiihtyvyysvektori)
```

Helpompi tapa tehdä laskeminen olisi ollut ns. eksplisiittinen Eulerin menetelmä, jossa olisi joka pisteessä laskettu kiihtyvyys ja tämän perusteella muokattu nopeusvektoreita. Tämä on hyvin epätarkka metodi etenkin pitkillä askelilla.

Runge-Kutta menetelmä on neljännen asteen integraattori, joka ottaa huomioon myös radan kaarevuutta, tuottaen paljon tarkemman lopputuloksen kuin Eulerin menetelmä. Keskivaikeaan toteutukseen vaadittiin myös tarkemman algoritmin käyttämistä.

Tietorakenteet

Planeetoiden koordinaattien esittämiseen käytetään Reitti (Route) ja Vektori (Vector) luokkia. Reitti koostuu kahdesta vektorista: paikka- ja nopeusvektorista. Vektori taas koostuu yhdestä numpy-oliosta, johon on tallennettu x-, y-, ja z-komponentit. Lisäksi Runge-Kutta laskennan yhteydessä on luotu Tila (State) olio, joka on hyvin samankaltainen kuin Route, mutta alustetaan eri tavalla, ja sen toteutus on paljon yksinkertaisempi.

Näiden olioiden käyttäminen tekee muusta koodista luettavampaa, kun tarpeelliset vektorit ovat kaikki samanlaisia, ja niihin on käytetty operator overloadingia peruslaskutoimituksien tekemiseen.

Tiedostot

Ohjelmaa käynnistäessä käyttäjältä kysytään planeettatiedostoa. Ohjelman mukana tulee esimerkkitiedosto, mutta sellainen on myös helppo tehdä itse. Planeetat tulee lisätä seuraavassa formaatissa:

nimi
massa (luku ja yksikkö, esim. 5.97e24 kg)
halkaisija (luku ja yksikkö, esim 6000 km)
paikkavektori verrattuna aurinkoon*
nopeusvektori verrattuna aurinkoon*

*vektorit annetaan komponenttimuodossa ”x: luku ja yksikkö jne”. Esimerkiksi ”x: 149.6e6 km y: 0m z: 0m” paikkavektorille. Nopeusvektorilla sama formaatti, mutta yksikkö on luonnollisesti oma: esimerkiksi kilometriä sekunnissa kirjataan ”km/s”.

Planeettatiedostossa tulee olla yksi ”planeetta”, jonka nimi on Aurinko tai Sun (ei ”the” etuliitettä), ja tätä käytetään systeemin keskipisteenä.

Testaus

Ohjelmassa käytettiin yksikkötestausta sen rakennusvaiheessa. Yksikkötesteissä testataan satelliitin lisäämistä, planeettatiedoston lukemista ja reitin päivittämistä.

Käyttöliittymän testaus on suoritettu käsin, sillä erilaisia käyttöskenaarioita ei ole kovin paljon. Etenkin virheellisistä syötteistä johtuvaa ohjelman kaatumista on testattu paljon.

Testaussuunnitelmassa ei ole täysin pysytty, sillä etenkin käyttäjän syötteen simuloiminen osoittautui aika haasteelliseksi, joten koko ohjelmaa ei ole kyetty suorittamaan yksikkötestauksilla.

Ohjelman puutteet ja viat

Suunnitelmassa kerrottiin, että ohjelmasta löytyisi ominaisuus, joka tunnistaisi jos satelliitti törmää planeettaan. Tätä ei ole implementoitu. Käytännössä jokaisen simulointiaskeleen jälkeen tulisi käydä jokainen satelliitti läpi ja tarkistaa, ovatko ne jonkin planeetan sisällä.

Planeettatiedoston luku ei ole kovin virheystävällistä. Se suodattaa ylimääräiset whitespacet pois, mutta jos siellä on yksikin virhe, voi olla, että koko tiedoston lukeminen epäonnistuu. Sen ei pitäisi kuitenkaan päästää virheellistä tiedostoa koskaan läpi.

Yritin testata ohjelmaa Lagrangen pisteessä, mutta se ei toiminut odotetunlaisesti, eli joko simuloinnissa on jotain vikaa, tai en löytänyt oikeaa pistettä.

3 parasta ja 3 heikointa kohtaa

Parhaat:

1. Simulointitulosten kirjoitus tiedostoon ja gnuplot: vaikka graafista käyttöliittymää ei olekaan toteutettu, saadaan planeettojen ja satelliittien radat piirrettyä gnuplotilla.

2. Yksiköt: arvoja voidaan syöttää useassa eri yksikössä. Erityisen kätevää tämä on simulointiasetuksia kirjoittaessa. Laskut ja tulostukset tapahtuvat kuitenkin aina SI-yksiköissä.

3. Planeettatiedosto: planeettatiedosto on selkokielinen ja sitä on helppo muokata itse, kunhan sen kirjoittaa oikein.

Heikoimmat:

1. Ei täyttä varmuutta simuloinnin toimivuudesta: Maa kyllä kiertää Auringon kerran vuodessa, mutta Lagrangen pisteitä ei ole löytynyt.

2.

3.

Poikkeamat suunnitelmasta

Käyttöliittymän luonnos muuttui hieman sekä satelliitin että planeetan parametrien syöttämisessä: nopeus annetaan suoraan vektorina, eikä itseisarvona ja yksikkövektorina erikseen. Lisäksi parametrien tulostuksessa ei aikaa tulosteta lainkaan, vaan pelkästään paikkavektori.

Satelliitin törmäystä ei testata millään lailla, mikä voi myös johtaa melko erikoisiin simulointituloksiin, jos satelliitti joutuu planeetan sisälle. Tämän kanssa tulee käyttäjän olla tarkkana.

Ohjelmaan lisätty mahdollisuus resetoida simulaatio, joten esimerkiksi yhden simuloinnin jälkeen ei tarvitse käynnistää ohjelmaa uudestaan, jos haluaa kokeilla samoja simulointiparametreja vaikkapa uuden satelliitin kanssa.

Ajankäyttösuunnitelma ei toteutunut täysin muiden kiireiden vuoksi.

Toteutunut työjärjestys ja aikataulu

Viikko 10: Luokat aurinkokunnan luomiseen, mukaan lukien tiedostonluku. Testaus yksikkötesteillä.

Viikko 10-12: Luokat simulointiin ja laskentaan, testaus edelleen yksikkötesteillä.

Viikko 12-14: Käyttöliittymä ja tulosten kirjoittaminen tiedostoon. Käyttöliittymän testausta.

Viikko 14-17: Viilailuja ja dokumentaatio

Arvio lopputuloksesta

Ohjelma laskee käyttäjän syöttämiä planeetta- ja satelliittiparametreja käyttämällä Runge-Kutta menetelmää ja Newtonin gravitaatiolakia. Simuloinnin tarkkuudesta ei ole täyttä varmuutta, sillä Lagrangen pisteissä sitä ei ole saatu toimimaan. Kuitenkin, etenkin planeettatasolla simulointi toimii hyvin.

Mielestäni ohjelma on rakenteeltaan varsin hyvä. Jokaisella luokalla on selkeät tehtävät, ja perintäjärjestys on selkeä, mikä näkyy myös UML-kaaviosta (toki siihen ei ole ihan kaikkia suhteita piirretty).

Ohjelmaa on helppo laajentaa etenkin graafiselle käyttöliittymälle, jota yritettiin implementoida, mutta ajanpuutteen vuoksi jätettiin kesken. Merkkipohjaisesta käyttöliittymästä saa hyvin otettua mallia graafiselle. Lisäksi muiden simulointimetodien kokeileminen on helppoa, kun simulointi on omassa luokassaan.

Viitteet

Glenn Fiedler, Integration Basics, https://gafferongames.com/post/integration_basics/, 29.4.2018 19:50

Michael Zeltkevic, Runge-Kutta Methods, http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html, 29.4.2018 19:50

Liitteet

Projektin lähdekoodi löytyy Skriptit-kansiosta. Pari ajoesimerkkiä löytyy Esimerkkiajot-kansiosta.