

## CSE-A1121 Ohjelmoinnin peruskurssi Y2

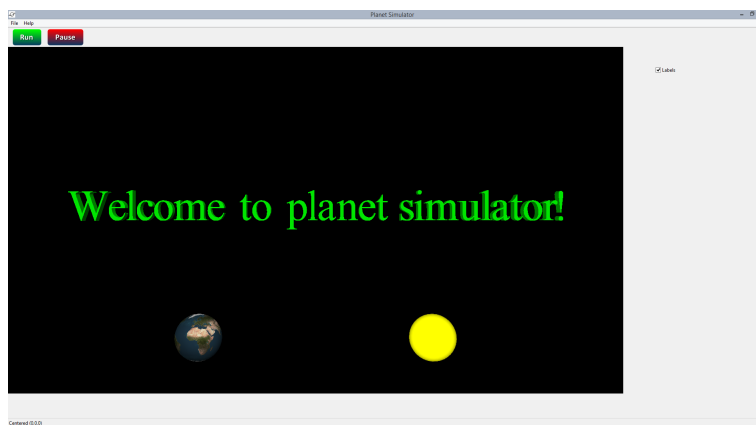
Planeetta simulaattori  
Projektidokumentti

Henri Merilä 356194  
Sähkötekniikan koulutusohjelma  
3. vuosikurssi  
Assari: Petri Leskinen

Dokumentti laadittu: 15.5.2015

# Yleistä

Loin projektityönäni planeetta-simulaattorin. Simulaattorilla pystyy lukemaan tekstitiedostosta planeettojen sijainnit simulointiavaruudessa, massat, nopeudet ja värin. Simulaationtilanteen lukemisen jälkeen simulaattorilla pystyy simuloimaan planeettojen liikettä avaruudessa kun aika kuluu eteenpäin. Ajan nopeutta pystyy säätämään, kuin myös laskennallista aikahyppyä ja piirtotaajuutta. Simulaattori piirtää 3D-mallinnuksen tilanteesta ja kykenee katsomaan tilannetta erilaisista näkökulmista. Tilanteen pystyy myös tallentamaan tekstitiedostoon.



Toteutus seuraa suurimmilta osin suunnitelmaa, mutta ominaisuuksia on jäänyt jonkin verran pois johtuen ajankäytöllisistä syistä. 3D-visualisointi kirjaston yhteensopivuusongelmat käyttöliittymäkirjastojen kanssa veivät suuren osan projektin toteutusajasta. Projekti on kokonaisuudessaan ollut minulle opettavainen ja vaativa.

# Käyttöohje

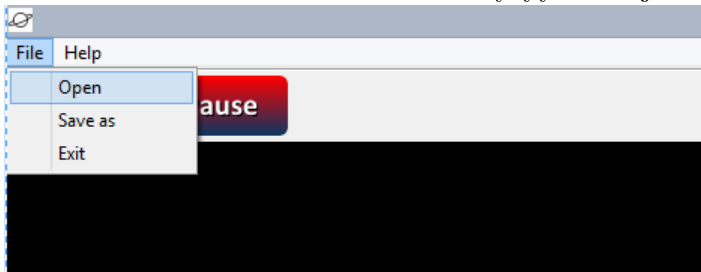
## Kirjasto ja käynnistys

Ohjelma vaatii toimiakseen VPython moduulin ja Pythonista version 2.7. Molemmat näistä voi ladata suoraan osoitteesta [http://www.vpython.org/contents/download\\_windows.html](http://www.vpython.org/contents/download_windows.html). Kun nämä on asennettu voi ohjelman ajaa suoraan tuplaklikkaamalla 'run.bat' tiedostoa ohjelman juurikansiossa. Klikkaamalla 'test-run.bat' ohjelma käynnistyy *debug*-tilaan käytyään yksikkötestit läpi. Debug-tilassa konsoliin tulostuu tietoa ohjelman ajosta.

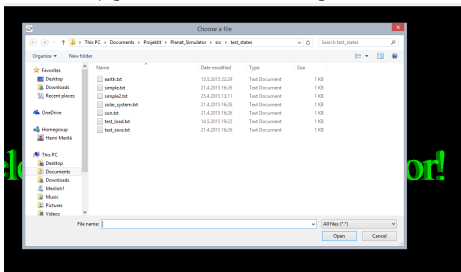
Documentation	15.5.2015 22:08	File folder
src	15.5.2015 22:05	File folder
run.bat	15.5.2015 22:05	Shortcut
test-run.bat	15.5.2015 22:04	Shortcut

## Valikko

Yläreunasta avautuvasta 'File'-valikosta löytyy ohjelman kannalta keskeiset tiedostosta lataamis(*Open*) ja tiedostoon kirjoittamis(*Save as*) toiminnallisuudet. Lisäksi valikosta löytyy *Exit* jolla voi poistua ohjelmasta.



Sekä 'Open' että 'Save as' valikkonäppäimiä käyttäessä avautuu tiedostodialogikkuna, josta tiedostosijainnit on kätevä osoittaa.



## Työkalut

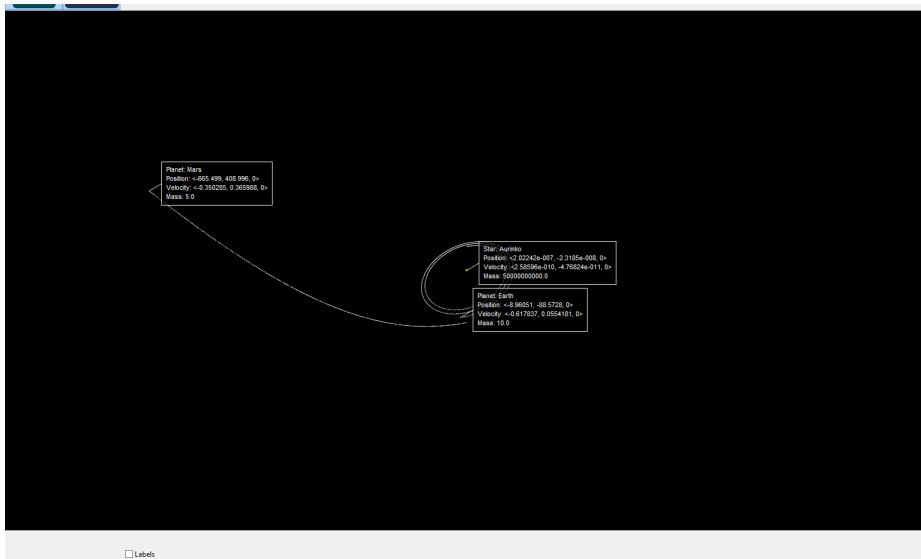
Työkalupalkista löytyvät *Run* ja *Pause* näppäimet. Kun simulaatiotilanne on ladattu, run näppäimellä simulaatio lähtee käyntiin, kun taas pause näppäin pysäyttää simulaation ajamisen. Pysäytetyn simulaatio-tilanteen voi tallentaa valikon save-näppäimellä



## Visualisointi-paneeli

Visualisointi ikkuna on koko simulaation sydän. Sitä voi hallita hiirellä seuraavilla komennoilla.

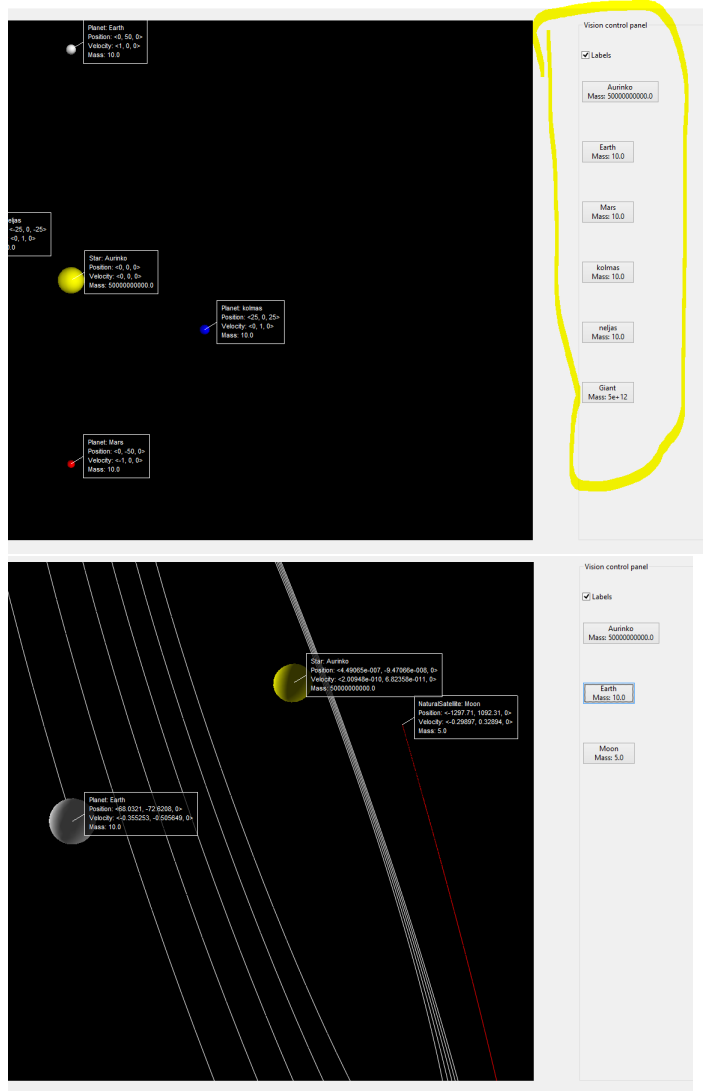
- Hiiren oikea näppäin pohjaan painettuna - Ohjaa kamerakulmaa
- Hiiren rulla näppäin pohjaan painettuna - Ohjaa lähennystä (Eteen taakse)



Lisäksi visualisointi paneelin alla on labels-checkbox, jolla visualisoinnista voi halutessaan poistaa elementtien merkkilaatat.















## Näkymän ohjauspaneeli

Näkymän ohjauspaneeli ilmestyy kun sopiva simulointitiedosto on ladattu. Siinä näkyy tiedot simulaation kaikista elementeistä. Paneelin tietyn elementin nappia painamalla kamera siirtyy seuraamaan kyseistä elementtiä.



# Ohjelman rakenne








Src kansion alta koostuva ohjelman lähdekoodi koostuu main- funktiosta ja kahdesta eri moduulista - Simulation ja GUI.

	GUI	21.5.2015 23:25	File folder
	Simulation	21.5.2015 23:29	File folder
	test_states	16.5.2015 20:53	File folder
	icon.jpg	18.4.2015 21:45	JPEG image
	main.py	21.5.2015 23:26	PY File
	main_debug.py	21.5.2015 23:23	PY File
	planet.ico	18.4.2015 21:45	Icon
	README.md	18.4.2015 21:45	MD File
	run.bat	21.5.2015 23:36	Windows Batch File
	test.bat	16.5.2015 20:47	Windows Batch File
	test_basis.py	21.4.2015 16:26	PY File
	test_physics.py	7.5.2015 21:33	PY File
	test_simulation.py	7.5.2015 21:31	PY File
	test-run.bat	21.5.2015 23:16	Windows Batch File

Select a file to preview.










## Simulation

Simulation-moduuli sisältää ohjelman simulointi-osuuden - avaruuden sen kappaleet ja niiden visualisointi. Simulointi sisältää myös fysiikka-laskuihin tarvittavat funktiot sekä simulaation parsimisen tiedostosta.

	Elements	14.5.2015 19:29	File folder
	_init_.py	18.4.2015 21:45	PY File
	physics.py	15.5.2015 0:33	PY File
	physics_test.py	22.4.2015 0:36	PY File
	run_test.py	28.4.2015 10:29	PY File
	simulation.py	14.5.2015 23:57	PY File
	space.py	15.5.2015 0:32	PY File

## GUI

GUI-moduuli sisältää ohjelman käyttöliittymään liittyvät osat. GUI-moduuli käyttää kutsuu simulaatiota.

Name	Date modified	Type
	_init_.py	18.4.2015 21:45 PY File
	boxer_test.py	12.5.2015 22:54 PY File
	dataPanel.py	18.4.2015 21:45 PY File
	mainWindow.py	15.5.2015 22:30 PY File
	pause_button.png	18.4.2015 21:45 PNG image
	run_button.png	18.4.2015 21:45 PNG image
	toolbarPanel.py	12.5.2015 22:19 PY File
	viewPanel.py	18.4.2015 21:45 PY File
	window_test.py	12.5.2015 22:04 PY File

# Algoritmit ja tietorakenteet

Ohjelmassa käytetään muutamaa fysiikan kaavaa laskemaan kappaleiden välisiä voimia ja uusia sijainteja. Laskut ovat pääosin vektori muotoisia johtuen 3-ulotteisesta avaruudesta.

Gravitaatiolaki

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

Kiihtyvyys

$$\vec{F} = m\vec{a}$$

Fysiikan päivityksestä piirtämiseen mennään seuraavalla kierrolla:

```
Simulation run
----> space calculate
-----> for each element calculate next
```

Tämän jälkeen kaikki päivitetään uuteen sijaintiinsa:

```
(Simulation run)
----> space update
-----> for each element update
```

Ruutu päivitetään run-komennolle annetun *frequency* parametrin avulla, käytännössä taajuudella tarkoitetaan kuinka monta kertaa sekunnissa ruudunpäivitys suoritetaan.

```
(simulation run)
---->If full second:
----->render visuals
```

## Tiedostot

Ohjelma pystyy parsimaan simulaatitiedot tekstitiedostosta. Parsiminen tapahtuu oheisella koodilla.

```
with open(file) as f:
    data = f.readlines()
    for line in data:
        if not line.startswith("#"):
            line_data = line.split()

            # If type planet
            if line_data[0].strip().lower() == "planet":
                label = line_data[1]
                vec = line_data[2].split(",")
                position = vector(float(vec[0]), float(vec[1]),
                                   float(vec[2]))
                vec = line_data[3].split(",")
                velocity = vector(float(vec[0]), float(vec[1]),
                                   float(vec[2]))
                mass = float(line_data[4])
                line_data[5].strip("\n")
                clr = Element.colors[line_data[5]]
                new_element = Planet(label, position, velocity,
                                     mass, clr)
                self.add(new_element)
```

Tiedostoon kirjoitus on hieman yksinkertaisempi operaatio:

```
f = open(file, "w")
f.write("# This is simulation state file\n")
f.write("# Tyyppi label position(muodossa: x,y,z)"
        "velocity(x,y,z) mass color\n")
for element in self.space.element_list:
    f.write(element.type + " " +
            element.label + " " +
            str(element.position.x)+","+str(element.position.y)+","
            + str(element.position.y) + " " +
            str(element.velocity.x)+","+str(element.velocity.y)+","
            + str(element.velocity.y)+" "+str(element.mass) + " " +
            element.get_color() + "\n")
f.close()
```



# Testaus

Testaus tiedostot sijaitsevat /src/ kansiossa. Ne voi ajaa painamalla test.bat tiedostoa. Yksikkötestit kattavat fysiikkamoottorin, simulaation ja elementit. Testausta varten tein ohjelmaan myös debug-moden, jossa ohjelma kirjoittaa ajonaikaisia kommentteja konsoliin. Tämän moden voi ajaa juuri kansion test-run.batista.

```
def test_element(self):

    # Init values
    self.assertEqual(self.planet.label, "Testi1",
                      "Name was not implemented properly")
    self.assertEqual(self.planet.position, vector(1, 2, 3),
                      "Position was not implemented properly")
    self.assertEqual(self.planet.velocity, vector(1, 2, 3),
                      "Velocity was not implemented properly")
    self.assertEqual(self.planet.mass, 100,
                      "Mass was not implemented properly")
    self.assertEqual(self.planet.space, None,
                      "Parent space was not implemented properly")

    # Init values
    self.assertEqual(self.star.label, "Testi2",
                      "Name was not implemented properly")
    self.assertEqual(self.star.position, vector(0, 0, 0),
                      "Position was not implemented properly")
    self.assertEqual(self.star.velocity, vector(0, 0, 0),
                      "Velocity was not implemented properly")
    self.assertEqual(self.star.mass, 100,
                      "Mass was not implemented properly")
    self.assertEqual(self.star.space, None,
                      "Parent space was not implemented properly")

    # Test for element methods update, __str__ and get_color
    new_pos = self.planet.velocity + self.planet.position
    self.planet.next_pos = new_pos
    self.planet.update()
    self.assertEqual(self.planet.position, vector(2, 4, 6),
                      "Element method update not working properly,"
                      + "got {}".format(self.planet.position))

def test_space(self):
    # Init
    self.assertTrue(self.space, "Initiating space is not working properly")

    # Testing add element
```

## Ohjelman tunnetut puutteet ja viat

Ohjelmassa on vielä paljon kehitettävää, mutta siinä on myös mielestäni erittäin hyviä puolia. 3D-visualisointi on suhteellisen näyttävä ja fysiikka on moduulimuotoisena myös kehitettävissä. Hyvää on myös parametrinä annettava simulointi nopeus ja timestep jotka mahdollistavat muuten hyvin hitaasti tapahtuvan simuloinnin nopeuttamisen. Erityisesti olen koodissa ylpeä näkymän hallinta paneelista joka dynaamisesti luo näppäimet ja funktiot kullekin simulaation elementille. Ohjelman kehitettävistä puolista mainitsen samaisen näkymän hallinta paneelista, sillä sen tila on rajallinen. Paneeli elementti tulisi muuttaa esim rullattavaksi, jotta siihen mahtuisi enemmän kappaleita. Lisäksi ohjelman staattinen ikkunakoko (ei skaalaudu pienentäessä) on miinus. Käytin paljon aikaa muutettavan koon toteuttamiseen, mutta käyttämäni kirjastot eivät pelanneet yhteen ja jouduin luopumaan ajatuksesta.

Plussat     + 3D-visualisointi kokonaisuudessaan  
             + Käyttöliittymän ominaisuudet  
             + Laajennettavuus

Miinukset   - Näkymän hallintapaneelin rajallinen koko  
             - Skaalaaminen  
             - Testejä voisi olla lisää

## Aikataulu

Sain projektin alkuun erittäin sujuvasti ja suurin osa suunnitellusta aikataulustani piti paikkansa, mutta tuli pari suurempaa vastoinkäymistä jotka veivät varsinkin toteutuksen keskivaiheilta paljon aikaani. Ehdottomasti suurin ongelma oli 3D-ikkunan upottaminen pääikkunaan. Grafiikkakirjastoni käytti pohjanaan wxPython kirjastoa, joten jouduin luopumaan kurssilla suositellusta PyQt kirjastosta alkuunsa. Halusin pitkään vielä tehdä ohjelmastani venytettävän, eli *Sizereita* käyttämällä ohjelman eri paneelit reagoisivat automaattisesti koonmuutos tapahtumaan. Kaikki muu toimikin hyvin, mutta 3D-ikkuna ei vain suostunut näkymään sizerissa. Internetistä ei löytynyt aiheeseen juuri mitään apuja ja ainut vaihtoehto olisi ollut modata käyttämäni kirjastoa ja se olisi ollut turhan iso taakka tähän projektiin. Muuten pysyin suht hyvin aikataulussani, mutta jouduin jättämään ajanpuutteen vuoksi suunnittelemani lisäominaisuuksia pois.

## Yhteenveto

Kokonaisuudessaan olen ohjelma-projektiini tyytyväinen. Kyseessä oli ensimmäinen laajempi itse suunnittelemani ja toteuttamani kokonaisuus. Suunnitteluni oli onnistunut ja se edesauttoi toteuttamista hyvin paljon. Lopputulos tekee perusasiansa hyvin, joten kokonaisuutena koen suorituksen itselleni erittäin hyväksi. Projektia tehdessä opin paljon luokkaperinnästä ja uutena tuttavuutena ohjelmoinnin aikana tuli *lambda*-ilmaisu, joka on toinen tapa tehdä funktioita. Tällä toteutin dynaamisen näkymänhallinta paneelini.

## Viitteet

<http://www.vpython.org/contents/docs/index.html>

<http://www.stackoverflow.com>

zet