Aalto-University
CS-A1121
11.3.2020

Benjamin Åkerlund
EST/ELEC
665762

Solar system simulator - Technical Plan

Program's structure plan

Superclass Planet:
- Superclass with initial values and parameters for radius, mass, speed and direction.
- All singular planets will be subclasses of this.
- One of the most important classes

Class Planet_graphics_item:
- Class to link the physical planets with the gui

Class Coordinates:
- Class for the polarized coordinates the solar system will be measured with, with the sun at origo.

Class direction:
- Class for describing direction with vectors.

Class SolarSystem/World:
- Where all of the other classes will be implemented

Class GUI:
- Class for implementing the GUI

Class Test:
- Class inheriting the features of the unittest.TestCase superclass.

Class Gravity:
- Class for how gravity will influence a satellite or other objects.

Class Satellite:
- One of the most important classes
- Similar class to the planet, only with different shape
- Initial values with size, mass, speed, direction and starting point
-

Use case description

The user could for example on a very basic level simulate if a satellite travelling at a certain speed will crash with a planet or other satellite. And how it will react to the different forces of gravity.

The user will in this case first input the simulation time and simulation speed. Then the initial location of the satellite and the values it has (mass, size). And finally the speed and direction the satellite is going in.

Benjamin Åkerlund
EST/ELEC
665762

Lastly the user will run the simulation and be offered choices of e.g. saving the results, keep running the simulation over a longer time, adding a new object/satellite to the simulation or terminating the sim.

Algorithms

The most important formulas used will be Newton's laws of motion with the most important one being F=ma. Newton's law of gravity will also be widely used.

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

Data structures

If the save function is implemented a very usable data structure will be the CHUNK structure. This is an easy way to save data in a .txt file, that we have already looked at in this course. Otherwise I believe Python's own predefined data structures will be enough.

Libraries

PyQt5 will be the main library I use for this project with all its handy graphics methods and so on.

Schedule

I will start by implementing a coordinate-system and a world. This is a vital first step in order to start adding items to the world later. After this I will want to implement at least some planets (the sun) and a way for them to orbit the sun. Here might be a good time to focus on the simulation time and the simulation speed as well in order to be able to test the orbits. To further the testing of this phase I will want to implement the barebones version of the UI and de graphics. Next I will implement the creation and adding of a satellite, since it is vital to the given assignment. And finally I will add Crashes and the possibility to save and load the results of a simulation.

Implementation order:

Aalto-University
CS-A1121
11.3.2020

Benjamin Åkerlund
EST/ELEC
665762

Week 1:
- World + coordinates
- Sun

Week 2:
- Planets
- Orbits, speed and direction of the planets

Week 3:
- Simulation time and speed
- GUI

Week 4:
- Satellites
- Crashes

Week 5:
- Load/Save

Week 6:
- Final touches to GUI and Planet graphics

Extra:
- If time, implementing the last part required for grade 5:
- Faster and more precise calculations

Unittesting plan

Testing will mainly be done with the GUI, mainly the simulation speed and time. Here I can also easily see the planetary orbits and that they work correctly. Later I can use this to test the satellites and the effects of gravity and crashes as well.

Literature references and links

https://theskylive.com/3dsolarsystem
https://www.solarsystemscope.com/
https://space.jpl.nasa.gov/
http://gafferongames.com/game-physics/integration-basics/
https://github.com/lukekulik/solar-system
https://gist.github.com/CodeDotJS/64f0d3d86d05b93af3b6
https://www.101computing.net/solar-system/