Report 10.3

So far I've mostly worked on geometric operations, in the package geometry, and implemented the “engine” of the project, in the classes SimulatorApp.scala and the package system. As I have quite a lot of classes I'll just breakdown packagewise what they do.

<u>geometry</u>

Vector.scala Works as a mathematical vector in R3. Contains methods like .unit() and .magnitude()

Line.scala A line that basically works as a container of a vector an a point. I thought it would be more useful, but now it’s basically only used to find the inersection of a line between the focalpoint and an object, and a plane.

Plane.scala A mathematical plane as defined by Ax + By + Cz + D = 0. Has a method to find the intersection point of a line. It is used in the camera.

Point.scala Similar to a vector but should be thought of as a position in space.

Matrix.scala Used for rotating the camera. The gauss-jordan method is not made by me. Sources are in the comments.

<u>System</u>

System.scala A solar system that contains several bodies. It is owned by the user interface. The system calls the bodies move-methods that uses numerical integration to calculate the new position and velocities.

Body.scala Contains the abstract class Body and the (current) classes star, planet, satelite and asteroid. I will probably not keep it as an abstract class, but use member variables to define the type of body. The further I got the more I reallized that I couldn’t justify using multiple classes.

The Runge-Kutta 4 method gave me some problems. I think I’ve understood the principle, but I’m not really sure if my implementation is correct. If I’ve understood it correctly, one should take four values k1, k2, k3 and k4 ,
make k1 to the acceleration given by the current force of the system,
make k2 what the current acceleration would be if the object had travelled with an acceleration of k1 for half of delta time from the initial position
make k3 what the current acceleration would be if the object had travelled with an acceleration of k2 for half of delta time from the intitial position
make k4 what the current acceleration would be if the object had travelled with an acceleration of k3 for delta time from the initial position.
and then finally take a weighted average.
I have the following questions:

- My implementation works, but I’m not sure if I’ve just made a slightly more complicated euler’s method. How can I test the accuracy?

- What is a “good” timestep? One hour? One day?

Camera.scala

A camera that basically projects a 3d-point onto a plane. It then does a gauss jordan operation that finds an x- and y- value for the point on a 2d canvas. It has a focalpoint that can be used for zooming. The method capture() returns an image of what the camera would see. Although it has some bugs, I was surprised that I managed to make it by myself and would really like some feedback on this class in particular.

Problems

My biggest problem which took almost 14 hours to solve was a double rounding error that made the planets' orbits look like stairs, but only when the camera was aimed in any other direction than 90 degrees. The problem was that because I was modelling a real camera with a focal length of 35mm, the values would become so small that a double value couldn't represent them accurately. This wasn't immeadiately noticed, because the same method that produced the rounding error also enlarged the values. I solved it by changing the focal length to 2 meters.

Tests

I have done a few unit tests for the geometric classes, but not that many. My primary form of testing has been to print out csv-data of the planets movements and using https://scatterplot.online/ to check if the values change as intended. This is how I found the aformentioned rounding error by the way.

Workload

Difficult to say since it's been quite a lot of passive work. But approximately nine days of six hours of work, so about 54 hours.

I haven't *exactly* followed the plan, but I have the most time demanding methods behind me. What I have left is only making the UI, tools for displaying and changing data, and some file management. So I do think that I am ahead.

Report 24.3

Classes

I’ve mostly updated the GUI, by adding panels. I made the camera position changeable from the window. The idea is that it always points at the center of the screen. I also made a fileformat for a state in the system and a working filereader/writer.

Tests

The filereader has been tested with a variety of files that it’s working properly with correctly formatted files. I haven’t done any error handling yet.

Problems

I’m having a few problems with implementing the GUI. Is there any better way to make the panels than just doing it programmatically? Like a tool or something? I find it difficult to control the widths and heights of for example text fields.

Workload

Maybe 6-10 hours since 10.3. I do think that I’m ahead still. What I have left to do is mostly testing, finishing up the GUI, improving the documentation, fixing small bugs and solve some TODOs that I left for later. I should also review the RK4-method.