Project Report: Solar System

Razvan Alexandru Cioban - 40204595

This report will present the OpenGL project “Solar System” that I was working on this semester for the class COMP371 – Computer Graphics. The project is a simulation of our solar system. Obviously, the number of features is not as big as other simulations one can find, but with the limited amount of time, and with my relatively low experience with the OpenGL framework, this application has enough features for me to be quite proud. This project can and probably will continue to be developed in the future, outside of the scope of this class.

The simulation has a working physics engine that utilizes Kepler’s law of planetary motion. This report will do its best to explain the mathematics used in the project. I could’ve gone the easy way and just make the planet go around a circle or ellipse around the Sun, but using actual physics is more interesting, more challenging, and gives a better result at the end. I will go on each of my objectives and explain how it works and how I implemented everything.

1. Celestial Body Representation: The real size and scale of the solar system is very difficult to represent since distances in space are astronomically big. For instance, the Sun appears just like any other starts from Uranus or Neptune. So, I took the decision that all planets shall retain their correct proportional size and distance. Meaning that if the Earth has a radius of 1, the Sun will have a radius of 109, and Jupiter will be a bit over 10. Same for the distance, if the Earth is at 1 au from the Sun, then Mars will be around twice as far. But there is no scale between the size and the distance. I took all the publicly found data for the orbits and planet information using Wikipedia and Nasa data, more on that in the next objective. For the model and the texture, everything was found online, all models except the one for Venus was taken from Akshat on sketchfab.com. Venus was taken from Danny.Gallegos.

2. My application does in fact implement a working physics engine. I have 4 types of inputs, the rotational inputs, the orbit inputs, the orbit plane inputs, and all the description type of inputs. Annex A shows the information that are freely available on Wikipedia for the Earth. For the rotational inputs, I simply have the angle of inclination and the day period, here the math is very simple, a full rotation happens once every day period, and I rotate the planet by the z-axis with the angle of inclination. Now it gets more complicated, an orbit is not a perfect circle, indeed it is an ellipse, and this ellipse is itself on a plane at an angle from the ecliptic (the ecliptic is the plane at which the Earth revolves around the Sun, so my Sun – Earth plane will be my reference plane and will be on the xz-plane of the world). All other planets have an inclination in respect to the ecliptic. To find the plane I simply use 2 orthogonal vectors defined in this document. The p vector points at the periapsis, or simply the point where the planet is at its furthest from the sun, and q is the vector 90 degrees from p. With the plane defined, I need to find the position of the planet on its orbit at any time, to do so I need a reference point. In astronomy we use Epoch J2000 as a reference, to find the position around the ellipse of a planet, I need to know where it was

at that reference point. J2000 means January 1st 2000 at 12:00 UT, so the angle of a planet at that specific time, I then add to that the current date and time to find the angle of where my planet should be. But Kepler's law is more complicated than just t/2pi to find the angle. Here I will try my best to explain the physics.

a. The first step is to find the mean anomaly at time t, the mean anomaly, means that we assume the orbit is a circle and we find the angle on the circle.

$$M_t = M_0 + \frac{2\pi}{T} * t$$

Where M_t is the mean anomaly at time t, M_0 is the mean anomaly at the reference point of time, T is the orbital period and t is the current time.

b. Then we need to find the eccentric anomaly E at the same time t, basically to pass from a circle to an ellipse. Here is the formula.

$$M = E - e * \sin(E)$$

This formula is not algebraically solvable. We must estimate E using a numerical method. In my case I am using Newton's method to estimate E. e is the eccentricity value of the orbit.

c. After finding the Eccentric anomaly, I need to find the true anomaly v. The reason we need to find this value is because a celestial body will always be faster near its Star when it is closer, and slower when it is further. Since the orbits are ellipses with an eccentricity value, we need to keep in mind that when the planet is closer to the Sun, it moves faster.

$$\tan\left(\frac{v}{2}\right) = \sqrt{\frac{1+e}{1-e}} * \tan\left(\frac{E}{2}\right)$$

d. Next, we need to find the polar vector r that represent the position of the planet at time t.

$$r = \frac{a(1-e^2)}{1 + e * \cos(v)}$$

Where a is the semi-major axis of the orbit, basically the furthest distance from the focal point of the ellipse.

e. Finally, with r and v, we can transform the polar coordinates into cartesian coordinates. Using the method showed in the document above.

3. For this objective everything was successfully implemented, the user can move freely using the mouse and keyboard, as well as having key binds to help the user interact with the system.

4. This objective was also easily integrated, although the user cannot click on a planet, but instead they can iterate from a planet to another. I am using ImGUI library to aid me with the UI. Most of the interesting information is displayed on screen when the system is focused on a planet.

5. This objective is weird in the sense that objective 2 already puts the planets at the right place at the beginning of the simulation, when the application opens, it takes the current computer date and time and uses this to place the planets using Kepler's law. The simulation runs at 1 seconds is equal to 1 hours in real life, but I also added the option to go faster or slower. 1 second can go up to 25 days or -25 days. Yes, the simulation also works in reverse. I did not add a date or time.

6. I added Saturn's rings since I feel like Saturn without its rings wouldn't be as interesting. I did not add any other celestial bodies, even if adding them wouldn't be that hard, all the formulas concerning body movement is already there. Technically, if I added the moon, the eclipse would have been visible on the simulation.

This project improved my coding skills and my understanding of the graphics pipeline. It was difficult at first, there is a very steep learning curve I feel like. If an Italian sees my code, they would be proud of the spaghetti I cooked. My code is not at all efficient and does not follow standard software coding practices and principles. The physics engine was clearly the most difficult part of the project, it took me about a full day of searching online and asking ChatGPT, as well as asking people around. And then another full day of just coding, even though it looks easy enough on paper, implementation is harder than it seems. Also surprisingly, adding a skybox was hard for me, I could not make it work, I finally did it though.

Debugging this project was a pain. Also for some reason my IntelliSense did not work at all on any IDE, because I used docker, and I did not installed the dependencies directly on the app, therefore it was time consuming to have to write all the code without autocompletion, and I had to go online to search for documentation for every single thing.

ANNEX A:

Orbital characteristics	
	Epoch J2000[n 1]
Aphelion	152 097 597 km (94 509 065 mi)
Perihelion	147 098 450 km (91 402 740 mi)[n 2]
Semi-major axis	149 598 023 km (92 955 902 mi)[1]
Eccentricity	0.016 7086[1]
Orbital period (sidereal)	365.256 363 004 d[2] (1.000 017 420 96 a_j)
Average orbital speed	29.7827 km/s[3] (107 218 km/h; 66 622 mph)
Mean anomaly	358.617°
Inclination	7.155° – Sun's equator; 1.578 69° – invariable plane;[4] 0.000 05° – J2000 ecliptic
Longitude of ascending node	−11.260 64° – J2000 ecliptic[3]
Time of perihelion	2023-Jan-04[5]
Argument of perihelion	114.207 83°[3]
Satellites	1, the Moon

Physical characteristics			
Mean radius	6 371.0 km (3 958.8 mi)[6]		
Equatorial radius	6 378.137 km (3 963.191 mi)[7][8]		
Polar radius	6 356.752 km (3 949.903 mi)[9]		
Flattening	1/298.257 222 101 (ETRS89)[10]		
Circumference	40 075.017 km (24 901.461 mi), equatorial[8] 40 007.86 km (24 859.73 mi), meridional[11][n 3]		
Surface area	510 072 000 km^2 (196 940 000 sq mi)[12][n 4] Land: 148 940 000 km^2 (57 510 000 sq mi) Water: 361 132 000 km^2 (139 434 000 sq mi)		
Volume	1.083 21 × 10^{12} km^3 (2.598 76 × 10^{11} cu mi)[3]		
Mass	5.972 168 × 10^{24} kg (1.316 68 × 10^{25} lb)[13]		
Mean density	5 513 kg/m^3 (0.1992 lb/cu in)[3]		
Surface gravity	9.806 65 m/s^2 (32.1740 ft/s^2)[14]		
Moment of inertia factor	0.3307[15]		
Escape velocity	11.186 km/s (40 270 km/h; 25 020 mph)[3]		
Synodic rotation period	1.0 d (24h 00 m 00s)		
Sidereal rotation period	0.997 269 68 d[16] (23h 56 m 4.100s)		
Equatorial rotation velocity	0.4651 km/s[17] (1 674.4 km/h; 1 040.4 mph)		
Axial tilt	23.439 2811°[2]		
Albedo	0.367 geometric[3] 0.306 Bond[3]		
Temperature	255 K (−18 °C; −1 °F) (blackbody temperature)[18]		
Surface temp.	min	mean	max
Celsius[n 5]	−89.2 °C	14.76 °C	56.7 °C
Fahrenheit	−128.5 °F	58.568 °F	134.0 °F
Surface equivalent dose rate	0.274 µSv/h[22]		
Absolute magnitude *(H)*	−3.99		

https://en.wikipedia.org/wiki/Kepler%27s_laws_of_planetary_motion

https://sketchfab.com/shooter24994

https://sketchfab.com/Danny.Gallegos

Models and textures:

Sun: https://sketchfab.com/3d-models/sun-cd65c13799fd47a1aeb4c8ac018b2914

Mercury: https://sketchfab.com/3d-models/mercury-32347fa4ec1a4987b71f461a401d91c4

Venus: https://sketchfab.com/3d-models/venus-d6578116c4254271bc18bbefe5389990

Earth: https://sketchfab.com/3d-models/earth-41fc80d85dfd480281f21b74b2de2faa

Mars: https://sketchfab.com/3d-models/mars-9c7bbc64d8c74acfa9ec344c0fc10e1a

Jupiter: https://sketchfab.com/3d-models/jupiter-d252c96ae3de48d7968b1206522ba9f5

Saturn: https://sketchfab.com/3d-models/saturn-8fb67d3defd74aaa880df3a08317e641

Uranus: https://sketchfab.com/3d-models/uranus-4d2f0c3674904472ac413fdabbf491d7

Neptune: https://sketchfab.com/3d-models/neptune-947a405a0a4348f9a49ff4bd3ed3cc4b