PA7: Solar System Report

Joriel Alves, Jordan DeNoce, and Lance Cantu

CS 480: Graphics

October 18, 2017

***Structure***

The solar system project has four folders and three files in its root directory. One of these files is the CMakeLists.txt, which gives cmake instructions on how to build the program with the files in the CMakeModules folder. The config\_file.conf file is a file that contains the values for the program’s objects and the file paths for the textures. These values can control various variables such as rotate and orbit speed. The textures loaded by the config file can be found on the textures folder. The include folder contains the header files for the cpp files, which can be found on the src folder. The third file is the README files, which gives users in GitHub instructions on how to run the program.

***Building***

The solar system project is built with cmake. In order to build this, the user must create a folder named “build” and go into that directory using the Linux terminal. When building the program with the command “cmake ..”, the user is required to have the libraries GLM, GLEW, SDL, Assimp, and Magick++ to be installed. If the user has those libraries installed, they can then proceed by typing “make” in their terminal. At this point, the program has been built and is ready. For the user to run the program, he must type “./Tutorial config\_file.conf” in the terminal, which it uses the configuration data file at the program’s root folder. Checking for the config file begins on the main.cpp file.

***Classes***

The main.cpp file contains the main function, which is the first function that the program runs. This file first checks if the config file the user provides is valid. If it is valid, it then runs the engine on engine.cpp.

The engine.cpp file is what keep the program running. It begins by initializing a window through the window.cpp file. It then initializes the graphics.cpp file. After that, it continually runs so it can update the delta time and detect inputs for the graphics.cpp file to utilize.

The graphics.cpp file is one of the most important files, as it handles all of the objects and renders them, as well as the camera and textures. It initializes the camera first through camera.cpp. It then opens the config file and stores all of the object values. These objects include planets, moons, rings, and orbit displays. After that, it applies the textures determined by the config file to each planet and moon. This is done through the textures.cpp file. After that, it loads the shaders and the matrices. It will then continue for the rest of the program updating the values for the objects and the cameras. These values determine various things such as speed, rotation, and position. The graphics.cpp file also receives signals from the engine.cpp file on what input has been detected and uses those inputs to update these values. With this, the program can determine where the user is moving the camera to, where the user is making the camera look at, and how fast the user wants the objects to rotate or move in their orbits. Users running the program can move the camera using the WASD keys and rotate the camera by using the arrow keys. Moving the camera can be done by changes the focus point and eye position vectors on the camera.cpp file while changing the speed of object movement can be done by giving the delta time a multiplier value that can be changed by the user through inputs. This multiplier value is then considered when loading the parameters for updating the objects through the object.cpp file.

The object.cpp file controls the objects in the program. These objects include the planets, the moon, the planet rings, and the orbit display rings. The skybox and the sun is considered a planet in this scenario. Objects get initialized by first loading the model’s obj file. It does this by reading the values in the file and pushing it to the vertice and mapping the indices with the uv coordinates. After the model has been loaded, the objects will then update for each frame. The update function they will run is determined by the config file, as there’s an update function for planets, moons, rings, and orbit displays. In these updates, they update the angle of their rotation and orbit, which is affected by the user given multiplier as these angles are dependent on the delta time variable.

***List of Features:***

* Model loading
* Textures
* All planets
* Pluto
* Satellites
* Sun
* Some type of interactive controls to change the view
* Orbit
* Planets
* Rotate
* Satellites orbit their planet
* Sizes are accurate (with respect to the other planets)
* Orbit paths are accurate (with respect to the other planets)
* Orbit speeds are accurate (with respect to the other planets)
* Rotate speeds are accurate (with respect to the other planets)
* Saturn rings
* Live adjustment of simulation speed
* Configuration file
* Proper Rings on other planets

***Extra Credit***

The total perceived extra credits for this project would be 4, but this can vary depending on whether or not the graders think that these requirements are truly met. In this project, live adjustment of simulation speed has been implemented, as users can press the “-“ key to slow down the speed, the “=” key to boost the speed, and Backspace to reset the simulation speed back to normal. The second extra credit that was implemented was the configuration file, which is present at the root directory of the program. With this configuration file, values for rotation, orbits, and various other factors can be changed for the program to read. The last implemented extra credit would be the rings on other planets. In total, these extra credits should total to 4 points.

***Issues***

During this program’s development many issues were encountered, but many of these were alleviated. One of these issues would be the camera movement. During early development, the camera would often get stuck or would move incorrectly. This was because camera movement and rotation was simply not done by incrementing and decrementing the eye position and focus point. By researching online, a math formula was implemented that made camera movement work properly. Another issue would be that the program would sometimes get out of memory errors. This often made the team wonder why their textures would not load. This was fixed by removing texture files that are too large. The team then tries to limit textures to not exceed 2 Megabytes.

Unfortunate, there were some issues that could not be alleviated. One of these issues was implementing a limit to how far the user can walk. The initial idea was to set variables that checks how much a user moved in a direction. This idea, however, does not work as directions can change when the user rotates the camera. The team could not form another formula and the idea was abandoned.