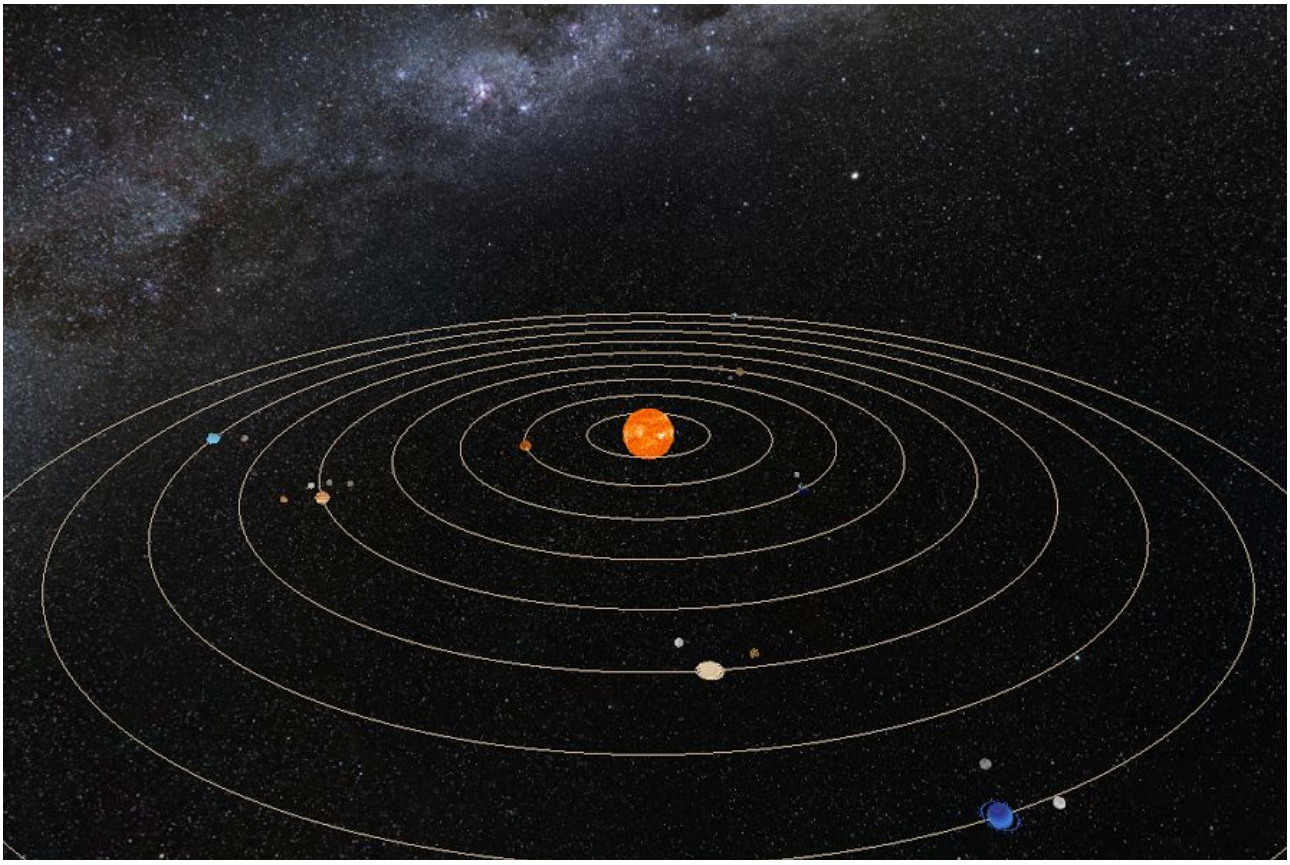
Technical Manual
PA 7 - Solar System



Group Members: Zeeshan Sajid, Vance Piscitelli

Required Libraries

In order to compile and run this program, some additional programs/libraries must be downloaded and installed. These programs/libraries are:
Magick++
Assimp
OpenGl

Setting Up Magick++

In your .cpp file, include:
```
#include <Magick++.h>
```

And initialize in main:
```
Magick :: InitializeMagick(*argv);
```

In the makefile, add:
```
LDFLAGS = `Magick++-config --cppflags --cxxflags --ldflags --libs`
```

and then add LDFlAGS to the compilation line like this :
```
$(CC) $(CXXFLAGS) ../src/shader.cpp ../src/main.cpp -o ../bin/Solar $(LDFLAGS)
```

Also, don't forget to add -lMagick++ to the LIBS to use.

Install these libraries with sudo apt-get install:
(These are the libraries installed on the ECC computers)
libgraphicsmagick3
libgraphicsmagick1-dev
libgraphics-magick-perl
libgraphicsmagick++1-dev
libmagickcore5-extra
libmagickwand5
imagemagick
libgraphicsmagick++3
graphicsmagick-libmagick-dev-compat
libmagickcore5
imagemagick-common

Or you can follow the installation instructions from:
http://www.imagemagick.org
To customize the installation.

Setting Up Assimp

To install Assimp, type:
sudo apt-get install libassimp-dev

Then modify your makefile to include `-lassimp` in the `LIBS` section:
`LIBS= -lglut -lGLEW -lGL -lGLU -lassimp -lMagick++`
Note: -lMagick++ is also added to use Magick++

In your main.cpp, you also need to add a few lines:
```
#include <assimp/Importer.hpp>
#include <assimp/scene.h>
#include <assimp/postprocess.h>
#include <assimp/color4.h>
```

Setting Up OpenGL

Most operating systems have some version of OpenGL on them but to run our program, a few additional programs need to be installed. The first two are GLUT and GLEW which can be easily installed with:
`sudo apt-get install freeglut3-dev freeglut3 libglew1.6-dev`

If you can type:
`glxgears`
into the terminal and some rotating gears appear, then you should have installed everything for OpenGL correctly.

You will also need GLM which can be installed by typing:
`sudo apt-get install libglm-dev`

Compiling the Program

To compile the program, simply type:
`make`
from the build folder or from the PA7 folder type:
`make -C ./build`

If an error occurs while trying to compile the program, try:
`make clean`
from the build folder or from the PA7 folder type:
`make clean -C ./build`
and then try compiling again.

Executing the Program

To run the program, from the PA7 folder, type:
`./bin/Solar`

The program cannot be executed from within the bin folder because the target paths for the objects are set from the PA7 folder.
e.g. `./bin/earth.jpg`

Program Implementation Details

Data Structures
The sun, planets, and moons are all stored in a vector of solarObjects. solarObjects is a struct that we created that keeps track of relevant information such as its parent, size, distance from the sun, orbital velocity, etc.

```
struct solarObjects
{
    char identifier;
    string myObjectName;
    string myTextureName;t
    int myPositionInArray = 0;
    int myDaddy = 0;
    float mySize = 0.0;
    float myRot = 0.0; // Rotational velocity
    float myOrb = 0.0; // Orbital velocity
    float myDistance = 0.0;
    float oAngle = 0.0; // orbital angle
    float rAngle = 0.0; // rotational angle

    glm :: mat4 myModel; // model matrix
    glm :: vec3 myVectorPosition; // current position
};
```

By using the same struct regardless of an object being a sun, planet, moon, or something else, we can easily add satellites or other objects that orbit anything without any complex changes to the code. So if we wanted an astronaut orbiting a spaceship orbiting a moon orbiting a planet orbiting a sun, the code to do so could be easily added.

We also have a vertex struct which holds the position, normals, and uv coords for a vertex.
```
struct Vertex
{
    GLfloat position[3];
    GLfloat normals[3];
    GLfloat uv[2];
};
```

Scaling

If the sun, all the planets, and all the moons are all on the screen at the same time and are scaled accurately in terms of sizes, orbital velocities, and rotational velocities, it is hard to see every object. For example, the Sun is over 100 times larger than pluto and is over 7 billion km away. In addition to having keyboard controls to navigate to different objects, we also included scaling.

When the scaling is toggled, each planet is scaled to the same size that is smaller than the Sun, and each moon is uniformly smaller than the planets. Also, each planet is the same distance from the previous planet. Overall, this allows the user to easily see all the planets at once giving a much better viewing experience.