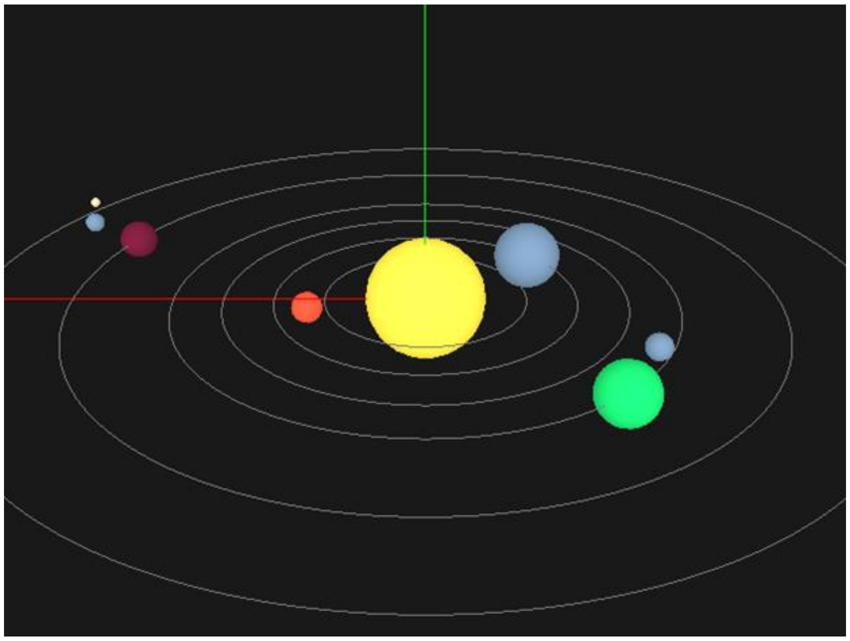
Lab 3: Solar System



Description:

You now are familiar with transformations and the order of operations that must take place. You now must understand how to build a hierarchy of objects (a scene graph) that can inherit properties to prepare you for Assignment 2 (Scene). In this lab, we will be building a solar system to understand how transformations are carried from one object to another through the model-view stack.

Your Task:

- You will build a sun, a few planets, and a few moons (or rings) that orbit the planets.
    - You will do this in the render() function in the SolarSystem class.
    - Make your own crazy solar system! Physical laws do not have to apply to your imaginary universe!
- Become more familiar with the rotate, translate, and scale operations
- Work with multiple objects that have (hierarchically) related coordinate systems
- Understand how to push and pop onto the matrix.
- Have fun!!

glPushMatrix() and glPopMatrix()

These two functions are important for this lab. Consider the following example:

```
glPushMatrix();
     glTranslate3f(1, 0, 0);
     glPushMatrix();
          glScale3f(2, 2, 2);
          glSolidSphere();  //draw sphere1
     glPopMatrix();
     glSolidSpehre();  //draw sphere2
glPopMatrix();
```

Sphere1 in this case will be: (1) scaled by 2, 2, 2, and (2) translated along +X by 1 unit. Sphere2 will only be translated along the +X by 1. Note that this code will have the same effect as:

```
glTranslate3f(1, 0, 0);
glScale3f(2, 2, 2);
glSolidSphere();  // draw sphere1
glScale3f(0.5, 0.5, 0.5);
glSolidSpehre();  // draw sphere2
```

However, this version is: (1) more expensive to run (one additional matrix multiplication), and (2) harder to keep track of in terms of the matrices and their inverses.

Files Given:

main.cpp – You do not need to modify this
MyGLCanvas.cpp and .h – This is the same code as the previous labs, but separated out into separate files for easier management
SolarSystem.cpp and .h – You will write the render function for the solar system.

Try out these following GL commands!

1. glPushMatrix();//These can be nested
2. glPopMatrix();
3. glLoadIdentity();
4. glTranslatef(0,0,0);
5. glRotatef(0,0,0);
6. glScalef(0,0,0);
7. glMultMatrixf(const GLfloat *matrix)
8. glutSolidSphere(radius,slices,stacks);

Going Further:
Did you enjoy this in class assignment?

- Try adding alpha blending to the planets rings. Start looking into textures and other materials that can make the planets appear more interesting.
- Add satellites that can orbit the planets
- Add asteroids that orbit the solar system
- Create multiple solar systems that all rotate around a galaxy
- Add some interesting simulation
    - If a moon gets too close to a planet, will it get sucked into another planets gravitational pull and rotate about it?
- Add more planets with irregular orbits
    - Pluto for example has a much more egg shaped orbit