# CCSI 3161 Assignment #2

**Objectives:** To develop an understanding of OpenGL transformations and projections, along with the display of geometric objects.

**Due date**: Nov 3rd, 11:59pm.

**Hand in**: Electronic submission of entire Visual C project (source, compiled, project files, etc.) using dal.ca/brightspace. Please zip up your whole project directory and submit the zip file. In addition to your project files, also include an up-to-date runnable build.

**Note, before beginning your assignment, please read:**

1. The coding style guideline.
2. The policy on plagiarism.

**General Comments**:

In this assignment you do not have to handle a window re-shape.

You must use the built-in OpenGL transformation commands to rotate, scale and move your objects in this assignment.

You should also use depth buffering and double buffering.

## The Hitch Hiker's Guide to the Planets [22 marks total]:

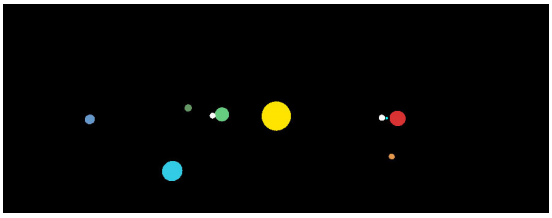
**(i)** Planetary System [8]

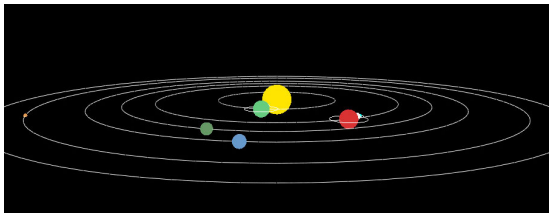
Construct a plausible planetary system which has at least a sun, 6 planets and 4 moons. But do not make it our solar system, instead make an imaginary one that you might see in Sci Fi.

Give your planets unique sizes, orbits and orbital speeds. *Give at least one planet an elliptical orbit*.

The user can also toggle all orbital paths (rings) to be seen or unseen by pressing the 'r' key. Use GL_LINES to draw the circle paths when on.

Here is an example of what it might look like (but the number of planets, their sizes and color will be different) :





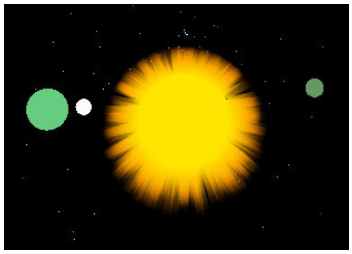
Hint: you can use gluSphere to draw a sphere, or there is source code on the assignment webpage called 'earth.c' that constructs a sphere. It's up to you what you prefer to use.

**(ii)** Stars [1]

Draw random star points in 3D outside of the solar system. Make the stars twinkle with randomly changing colors. The user can toggle the stars on and off by pressing the 's' key. Here's what they look like:

**(iii)** The Sun's Corona [2]

Give the sun a fiery corona. Use a number of semi-transparent lines that begin just inside the sun and stretch outwards, becoming transparent at their tips. The position of the lines at the surface should be random, and all the lines should point outwards from the center of the sun. At each frame of animation the random position/direction of the line should change. That will give it the fiery appearance. It should look similar to this:



The user can toggle the corona on and off by pressing the 'c' key.

Hints:

1. You only need to make the lines lie on the XY plane.
2. Make the color of the lines the same as the sun at the end of the line that is touching the sun. Then you can make the tips of the lines a different color.
3. You can also experiment with the line width.
4. You should draw transparent objects last or you might get strange results when depth buffering is turned on.

**(iv)** Starship Enterprise [4]

Read in the file 'enterprise.txt'. It contains vertices that look like:

```
v 0.242636 0.170825 -0.0272018
v 0.269521 0.170825 -0.0192831
v 0.269521 0.170825 0.195895
…
```

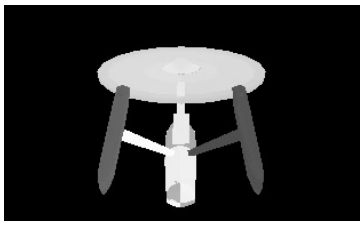
Each 'v' represents a vertex its 3D coordinates.

And, at the end of the file are faces that look like:

```
f 29 30 31
f 29 33 31
f 32 31 33
…
```

These faces are just triangles. The three numbers refer to the three vertices of the triangle (ordered from the start of the text file).

For the model file see: www.cs.dal.ca/~sbrooks/csci3161/assignments/index.html



This is just a big collection of triangles. You need to read in the vertices, and the triangles (which are a set or 3 vertex numbers). Then, you just need a big for loop to display each triangle separately.

Place the enterprise just below and in front of the camera at an appropriate scale.

Hints:

1. Don't panic.
2. You can increase each triangle's color from (1/1989) to (1989/1989), if you think the enterprise looks too flat using all one color. But that is optional.

**(v)** Camera Control [2]

Allow the user to control the position of the camera (at the enterprise) with keyboard keys:

| | | |
|---|---|---|
| GLUT_KEY_PAGE_UP | : | moves the camera forward |
| GLUT_KEY_PAGE_DOWN | : | moves the camera backward |
| GLUT_KEY_ UP | : | moves the camera up |
| GLUT_KEY_ DOWN | : | moves the camera down |
| GLUT_KEY_RIGHT | : | moves the camera right |
| GLUT_KEY_ LEFT | : | moves the camera left |

The user can hold down a key to continuously move in a direction, so you might want to use glutKeyboardUpFunc (which tells you when the user has released a key) as well as glutKeyboardFunc (which tells you when a key has been pressed).

Use perspective projection with gluPerspective and use gluLookAt to position the camera.

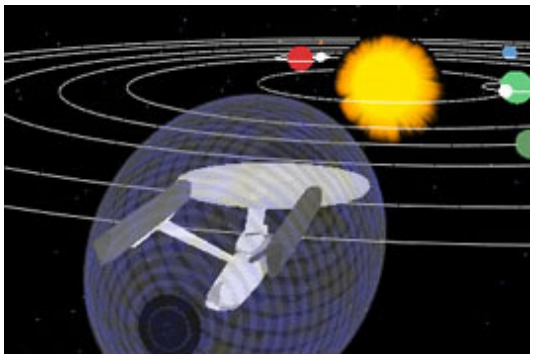
You don't need to tilt the enterprise in the direction that it moves (but it's a nice effect!).

Hint:

1. Use glutSpecialFunc(specialKeys) to register a callback function for the special keys.

**(vi)** Add something cool to the Enterprise [5]

In my example, I made a shield around the enterprise. But, do not copy my addition, instead think of your own addition. **Add a readme file that states exactly what your inventions are and how to use them.**



---

Keyboard controls [0]

Use printf to dump out a listing of all the keyboard key functions that you are supporting. Get the keyboard controls working.