Assignment 3: "3D World Walk Through"
CSI4105: Computer Graphics
Assigned: May 2nd, 2017 (Tuesday)
Due: May 18st (Thursday), 2017 (by 11:55PM uploaded to YSCEC)

Purpose of this Assignment

- To learn how to place 3D objects in the "world"
- To learn how to setup a light source and set object materials in OpenGL
- To learn how to navigate the camera position in the 3D world.

TASK

You are to make an OpenGL/GLUT application in C/C++ that populates a 3D world with 3D objects that have simple animation. The user should be able to "walk through" your 3D world by changing the camera position and viewing direction. Your application should also allow for a "bird's eye" view where the camera is placed above the world looking down. Additionally, the user should be able to move the light sources as well as the objects.

Minimum Requirements

1) Populate the 3D world

Establish a 3D world coordinate frame and draw a large polygon to represent the ground plane. Populate your 3D world with 3D objects. The GLUT API provides several calls to draw 3D geometric objects including: spheres, teapots, torus (the "donut"), cubes, cones, and a few others. You can also write simple routines to draw your own geometry. Your 3D objects should have some simple animation associated with them, such as rotation.
In a "selection mode", you should be able to select on a 3D object and move them around. That is:

- Define a key for the selection mode.
- In the selection mode, you should be able to select an object and translate them with either a mouse or a keyboard.

2) Moving the user (camera)

Establish a camera that moves through your 3D world. Your motion should be done as follows:
Use the keyboard or mouse to move the viewer's direction left/right.
- This will give the effect that the view changes, but the position does not.

Use the keyboard or mouse to move viewer forward/backward.
- The camera moves in the current direction, either forward or backward.

NOTE: The user should move along the ground plane only. Think of this as a typical 3rd person shooter camera. This means you can't arbitrarily translate vertically in space.

3) Toggling camera views + avatar (space key)

1) One camera view should be from the viewer's point of view as described above in "2".

2) If the user hits the "space" key, toggle into a view that shows the entire world. When you show the world, you should show 3D geometry that represents the viewer (i.e. an avatar). In the sample program the avatar is a white teapot and its spout represents the viewing direction. You are free to use any geometry you want to represent your viewer, but whatever you use, it should be very clear what the viewing direction is. Note that in this bird's eye view, you should still be able to move the user. Also, any animations of the 3D objects in your 3D world should appear in this new view.

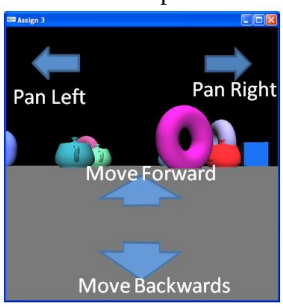
4) Light Source

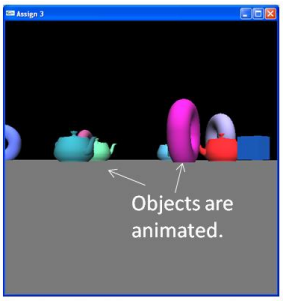
You should have at least one light source (you can have more) in your 3D world. Its position should make sense for your 3D world and illuminate the objects in a pleasing manner (that is, your objects shouldn't be black due to bad lighting!).

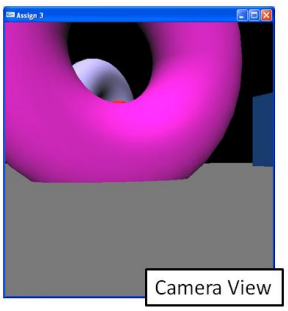
You should be able to move at least one light source. That is,

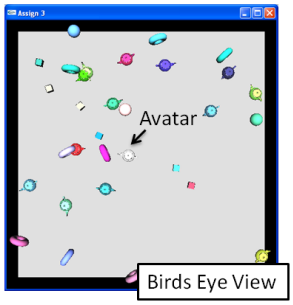
- Use the keyboard or mouse to move the light source direction. (use different keys from 2)
- Use the keyboard to translate the position of the light source. (use different keys from 2)

Some screen captures









☼ **Rubric**

To get an "average" grade, the minimum you are required to do is to accomplish task 1-4 described above. If you cannot do this, you're grade will be below average.

- +35 pts: Populate the 3D world with animated 3D objects (Task 1).
- +10 pts: Implementation of selection mode in Task 1.
- +30 pts: Implementation of Task 2, creating a manipulable user (camera). This includes implementation of the manipulable avatar in Task 3.
- +10 pts: Implementation of camera toggling with "space" key in Task 3.
- +15 pts: Implementation of movable light source in Task 4.
- -5*n pts: 5 points are lost every time the instructions for the submission format are not followed (More importantly, filename convention is worth 10 points!)

☼ **Extra Credit (up to 10 pts)**

To get an "above average" grade, you need to perform additional interesting "Bells and Whistles" beyond the basic requirement. Some suggestions:

- up to 2 pts: Interesting 3D objects
- up to 5 pts: Additional interaction (like collision detection)
- up to 3 pts: Being able to "shoot" items
- up to 2 pts: More interesting animation for your 3D objects (simple animations will not earn any points)
- and so on

Assessment

- You will be assessed individually in the lab sessions
- You'll only have approximately 7 minute Q&A session with either the Professor or the TA's.
- Your uploaded file will be downloaded directly from YSCEC and compiled in the lab
    - You cannot use a different C file (don't even ask!)
    - Code must compile and run.
    - You can be asked anything about your code (every single line)
    - Be prepared to answer the questions, I forgot isn't a good answer!

Submission (READ CAREFULLY)
Please follow the instructions on the filename convention, it is important!

You must implement your assignment in a **single** C/C++ file. Even if you are an expert in C and know how to split the program into multiple headers/source files, it is required that the entire program is within a **single** C file; this is a hard constraint, no exceptions! This is necessary to expedite compiling for our in-class assessment. This rule will apply to all assignments. *Also, it is very important that you use the proper filename convention.*

NAMING CONVENTION
File name:
{ID}_{surname}_{firstnames}_A3.cpp
Example
2015147000_Kim_SeonJoo_A3.cpp

Please upload your file to YSCEC.
A letter grade lower every day turned in late. Marks deducted for incorrect filename.