Fall Quarter 2015

ECS175 Project 2 Information Sheet

This is a general overview of project 2, please refer to project guideline published on SmartSite and the project prompt for additional requirements. The command syntax here is for demonstration purposes only, you don't necessarily have to follow them.

Requirements

You can use glBegin/glEnd/glVertex… to draw your lines, BUT we highly recommend you stay with your PixelBuffer. It will make things easier for Project 3, where you have to go back to the pixel buffer. You CAN NOT use glRotate/glRotatef/gluOrtho2D

Along with the requirements outlined in project guideline, here's a rough list of things we will check during grading. Make sure the following are working and you should do well. (Point value associated with specific part is in [])

1. General functions [10]: In this project, you SHOULD hardcode the size of your viewport. Additionally, your program should work with normalized device coordinates (This means all shapes and objects should be always "in view," translation vector can be given as world or pixel coordinate). Your program should allow input data from file and export (save) data to file. These files should be in world coordinate. Also, remember your program should at least be able to handle 3 objects.
2. Transformations [60]: User should be able to use the interface to apply various 3D transformations to the polygons in the scene.
    a. Scale [15]:

        User should be able to scale any object in the scene with respect to it's centroid like the following:

        >>Scale 1 (2,2,2) //this should scale object with internal ID 1 by 2 in all directions

    b. Translate [15]:

        User should be able to translate any object in the scene, you should also specify the format of the translation vector in your README file (whether it is world coordinate or pixel coordinate)

    c. Rotate [30]:

        User should be able to rotate any object in the scene with an arbitrary axis by an arbitrary degree. Like the following:

>>Rotate 1 (0,0,0) (1,1,1) 90 //this should rotate object with ID 1 around axis (0,0,0)-(1,1,1) by 90 degree

Your program should also display the rotation axis in 3 projection planes in different color.

3. Orthogonal Projection [10]: Your program should display the xy, xz and yz projection of the scene. All 3 projection view of the scene should update once a transformation has been applied.
4. UI [15]: If your UI is functional without any issue (i.e. crashes when running some command, inconsistent with README instructions), you should receive full credit on this part. Refer to Project Guideline for some helpful resources for UI creation.
5. Manual [5]

Extra Credit

The following are published extra credit for this project.

1. Allow user to choose different projection scheme to view the scene (oblique or perspective) [5]
2. Animations! [5]
3. If your UI is stunning! [up to 5]

If you've done anything else extra, please document it in README file so we can assess it and try to give you more points!

Helpful Notes

Please do not fill the objects, wireframe picture will do!

It will be helpful if you first sketch out your scene on paper then look at your program to make sure all 3 projections look consistent.

For the UI, since you will be showing 3 projection planes, you can either create 3 different windows or create 1 main window then divide that into 3 sub-windows. Refer to glut-3.spec.pdf for information on how to create sub-windows.