Fall Quarter 2015

ECS175 Project 3 Information Sheet

This is a general overview of project 3, 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

In this project, you have to use the pixel buffer and set pixel RGB value via your own functions. You CAN NOT use gluOrtho2D/glPerspective etc.

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. Phong Model [30]: You should illuminate your scene with Phong lighting model. Your program should compute the color using the formula provided in class. Your program should allow user to set various parameters (vectors) of the model via your UI. Since you are not required to implement CVM in this project, the scene could look abnormal due to the fact that you are using orthogonal projection to view the scene.
2. Gouraud Shading [20]: Your program should use Gouraud shading to compute intensity (color) of pixels whose intensity is not generated by Phong model.
3. Painter's Algorithm [15]: Your program should use painter's algorithm to remove surfaces not visible to the user. The "distance" of the surfaces should be computed with respect to x-infinity, y-infinity and z-infinity since we implement orthogonal projections to view the scene.
4. Halftone [15]: Your program should have an option to turn on/off half-tone mode. When in half-tone mode, your program should fill the object surface using the half-tone algorithm described in class (the mega-pixel!).
5. 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.
6. Manual [5]

Extra Credit

The following are published extra credit for this project.

1. Display non-plainer/intersecting objects [5]
2. Camera Viewing Model implementation [5]
3. Even cooler animations! (it has to be better than last time ☺) [5]
4. 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

Your program should implement all the requirement as one complete program instead of several small programs (although the prompt has part a) b)…)

There are a lot of parts in this program, keep in mind that graphics processing is a pipeline, so you should establish which process to be done first, this would make programming easier.

It would be easier if you test each part separately, Ex. for Gouraud shading, you should generate some simple picture to see if it's working before you put it in your processing pipeline.

Instead of computing normal vectors in your program, you can include normal vector information in your input file.