

Project 1

CSCI-4961/6961: 3D Computer Graphics

Due: Friday, September 22, 2006, 11:59:59pm

1 Overview

This project is intended to familiarize you with OpenGL and its 3D modeling, viewing, and projection transformations. You are to write a program to display an image of an outdoor scene with a windmill and some hills. You will use both OpenGL and GLUT functions such as `glTranslatef()` and `glutWireCone()`, and use mouse and keyboard input events. You will also need functions such as `gluLookAt()` and `gluPerspective()` to set up the viewing and projection transformations.

Please see the course web page at www.cs.rpi.edu/~sakella/graphics/ for an example piece of OpenGL code, and for project updates and additional information.

2 Project Tasks

You are to write an OpenGL program to draw a windmill and some hills. They are positioned on a horizontal ground plane. You can use functions such as `glutWireCube()` and `glutWireCone()` to construct your scene objects.

1. First draw the ground plane. The ground plane is the XZ plane, and has its origin at $(0, 0, 0)$. The positive Y axis is the vertical axis for this world. The ground plane has dimensions 20 units in X (going from -10 to 10) and 20 units in Z (going from -10 to 10). Draw a set of grid lines at a spacing of 1 unit along the X axis and the Z axis.
2. Draw the X, Y, Z axes at the origin using three lines colored red, green, and blue respectively. The length of each line should be 1 unit.
3. Select the initial camera location to be at $(0, 1, 5)$ and pointing parallel to the negative Z -axis. Let the default projection mode be perspective projection.
4. Draw the windmill with the center of its base at $(0, 0, -3)$. The windmill should have two blades perpendicular to each other forming a cross. The tower and the blades can be modeled as rectangloids using the `glutWireCube()` function.
5. Create hills using the `glutWireCone()` function. You should have at least five hills in your scene. Some of the hills should be along the back of the scene (that is, the $z = -10$ edge of the scene), and additional hills may be placed along the perimeter of the scene.

Use the `glColor3f()` function to display the ground plane, windmill, and hills in different colors.

6. To enable user control of the viewpoint, write functions to translate the viewer forward, backward, to the left, to the right, and to rotate in place. These viewer interactions will be performed using both the mouse and keyboard as specified below.

The main keyboard events and the corresponding desired behavior are:

- `w/s` keys: The viewpoint moves 0.1 unit forward/backward respectively, with no change in orientation.
- `a/d` keys: The viewpoint moves 0.1 unit to the left or 0.1 unit to the right respectively, with no change in orientation.
- Provide the ability to rotate the viewer left by five degrees about the Y axis with each click of the 'comma' key (which also has the `<` symbol) and rotate the viewer right by five degrees about the Y axis with each click of the 'period' key (which also has the `>` symbol). This can be helpful during debugging. Note that these rotations are in place.
- `r` key: Add the ability to rotate the windmill. The windmill should rotate by 10 degrees about its horizontal axis at each key press.
- `i` key: The camera and objects are returned to the initial configuration.
- `q` key: The program quits.

The main mouse event is:

- With the left mouse button pressed, a leftward horizontal motion of the mouse should rotate the viewer left, and a rightward horizontal motion of the mouse should rotate the viewer right. You will need the `glutMouseFunc()` and `glutMotionFunc()` functions for this. Remember that these rotations are to be performed in place.

3 Grading

Your project will be graded for a total of 100 points as follows:

- 45 points for modeling and displaying the windmill, hills, and ground plane.
- 15 points for viewpoint translation operations.
- 15 points for viewpoint rotation operations.
- 10 points for animating rotation of the windmill.
- 10 points for program functionality, structure, and documentation.
- 5 points for the creativity of your scene design and any special features or enhancements.

Describe your enhancements in your README file, and ensure that they do not interfere with the required project functionality.

Lateness policy: Please read the lateness policy in the course syllabus. Use your late days carefully, if at all.

4 Submission

The code must be submitted no later than **11:59:59 pm on September 22, 2006**. Specific instructions for handing in your code will be provided on the course web page. Your source code must be readable and commented. The comments need not be extremely long, just explain clearly the purpose of each block of code.

You must hand in a README file, your source code (source and header files), and if necessary, a Makefile to compile it. Your submission should NOT include any object files or executable files. Your submission should not use `glaux.h` or Windows API files (e.g., `windows.h`). Every file should have your name in a comment line at the top. The README file should contain the following information: your name, instructions on how to compile the code and run it, known bugs or limitations, any extra credit enhancements, and any other relevant information.

Proper submission is entirely **your responsibility**. Contact the TA if you have any doubts whatsoever about your submission. Do **NOT** submit your project via email. Be sure to keep copies of your files and **do not change them after submitting**. After grades are posted, you have exactly one week to resolve all problems. All grades are final two weeks after they are posted.

A project that does not follow the submission guidelines will receive a **10 point deduction**. Please observe the academic integrity guidelines for the course as described in the course syllabus.