

Project 4

CSCI-4961/6961: 3D Computer Graphics

Due: Monday, December 4, 2006, 11:59:59pm

1 Overview

This project is intended to familiarize you with performing texture mapping and creating spline and quadric surfaces in OpenGL. Imagine you are a game designer. You are to write a program to display an indoor or outdoor scene for the game of your dreams, and enable the viewer to move about the scene. You will use both OpenGL and GLUT functions for modeling the scene and texture mapping the scene, and use mouse and keyboard input events. You will also need to set up the viewing and projection transformations.

Please see the course web page at www.cs.rpi.edu/~sakella/graphics/ for project updates and additional information. Also check the course WebCT bulletin board for relevant posts.

2 Project Tasks

You are to write an OpenGL program to create a 3D scene for a game of your design. The scene must include at least five texture mapped objects, including a polygon, a cube, a sphere, a cylinder (or a cone), and a spline surface. You may create or add other object shapes to the scene or otherwise enhance it to make it more realistic. You may assume that the objects do not translate in the scene, and that only the viewer moves about the scene.

1. Select the initial camera location to be at $(0, 0, 0)$, and looking towards $(0, 0, 1)$ so it is pointing along the positive Z axis. Let the default projection mode be perspective projection. The viewer is going to move parallel to the XZ plane, and the positive Y axis is the vertical axis for this world.

Select appropriate lighting and material properties to illuminate your scene. You should locate all objects within a square region with sides of length 50 units centered at the origin. That is, the X coordinate can vary between -25 and $+25$, and the Z coordinate can vary between -25 and $+25$.

2. The cube should be texture mapped so that the textures on at least two of its faces are different. The cube should continuously rotate in place about the vertical.
3. The sphere and cylinder (or cone) objects may be specified using GLU routines for quadrics objects.

4. The spline surface may be specified as a Bezier surface or as a NURBS surface. The spline surface should be animated by moving one or more of its control points (for example, it could be a window curtain for an indoor scene, or a flag for an outdoor scene). The spline surface should be texture mapped as well.
5. Your program can perform texture mapping both with texture image files and using procedural textures (e.g., checkerboard). However you should use at least one texture from an image file.

A simple example file `TextureMap.cpp`, which illustrates how to read in images in BMP file format using the classes in `pixelmap.h` and `pixelmap.cpp`, is available on the course web page. We will also provide a few sample textures in BMP format on the web page. You can create additional texture files in BMP format or use those obtained from other sources. Please convert all texture files in other formats to BMP format.

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. (Note: You must implement both sets of controls described below for maximum user convenience.)
 - When the up arrow key (or *W* key) is pressed, the viewer should translate forward one step (without turning).
 - When the down arrow key (or *S* key) is pressed, the viewer should translate backward one step (without turning).
 - When the left arrow key (or *A* key) is pressed, the viewer should translate left one step (without turning).
 - When the right arrow key (or *D* key) is pressed, the viewer should translate right one step (without turning).
 - 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.

Additionally, provide the ability to rotate the viewer left by five degrees with each click of the 'comma' key (which also has the `<` symbol) and rotate the viewer right by five degrees with each click of the 'period' key (which also has the `>` symbol). This may be helpful during debugging.

Note that you will need the `glutSpecialFunc()` function to use the arrow keys.

7. The remaining keyboard events and the corresponding desired behaviors are:
 - *m/M* key: Toggles the animation of the spline surface on/off. The default mode is for the animation to be turned on.
 - *t/T* key: Toggles texture mapping on/off. The default mode is for the texture mapping to be turned on.
 - *i* key: The camera is returned to its initial configuration.

- `q` key: To quit the program.
8. If you are looking for additional fun and challenge, here are some ideas: Consider texture mapping a torus, or creating a lava river (with moving textures), or creating a glowing surface, or environment mapping an object, or bump mapping an object, or implementing a jump operation, or other cool things.

3 Grading

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

- 25 points for creating a basic scene with an appropriate set of objects and lighting.
- 25 points for texture mapping objects in the scene.
- 20 points for keyboard and mouse event functions (to control viewpoint, to toggle texture mapping, to animate, to reinitialize, to quit).
- 10 points for the realism and creativity of your scene design.
- 10 points for code structure, clarity, and documentation.

Describe any enhancements or special features 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 Monday, December 4, 2006**. You will follow the same submission procedure as for Project 1. Specific instructions for handing in your code are 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 submit a single zip file containing your README file, all texture files your program uses (along with the code to read them in to your program), 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 you create (except the texture files) should have your name in a comment line at the top. The README file should contain the following information: your name, email address, instructions on how to compile the code and run it, known bugs or limitations, any 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 in a protected directory of your RCS or CS account 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.