

CS 537/CPE 537: Homework Assignment 3
Due: November 11, 6:15pm

Philippos Mordohai
Department of Computer Science
Stevens Institute of Technology
Philippos.Mordohai@stevens.edu

Requirements. For each programming assignment you should submit a zip file containing the code and a brief report stating: what you did; how you did it; any particular features you want to draw attention to; or any problems with the program you know about. Submit the zip file by email to me with a subject line like “CS537: Homework 3”.

Make sure your programs compile and run on a generic PC with Windows XP with VC++ and glut installed. You run the risk of getting no credit if your program does not compile. Prefer simple portable code to alternative fancy solutions.

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment. I will assume that you will be taking the responsibility of making sure that you personally understand the solution to any work arising from collaboration.

Late Policy. The penalty for late submission is 20% of the grade per day, enforced at 6:15 each day after the due date. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please send me an e-mail explaining the situation.

Problem 1.

Write a program that provides the user with an interface to apply color or texture to the faces of a stationary cube. The display should look like the one in Fig. 1. The user should be able to click on a face of the cube and then click on one of the four panels at the bottom to apply the corresponding color or texture to the selected face. The requirements are the following:

1. Specify a stationary cube so that three faces are visible. To simplify the problem, the cube may consist of only the three visible faces (squares). **(10%)**
2. Provide the user with at least four options to apply color and texture to the selected face of the cube. Each option should be drawn in a square at the bottom of the window. At least one of the options must be a texture map. **(10%)**

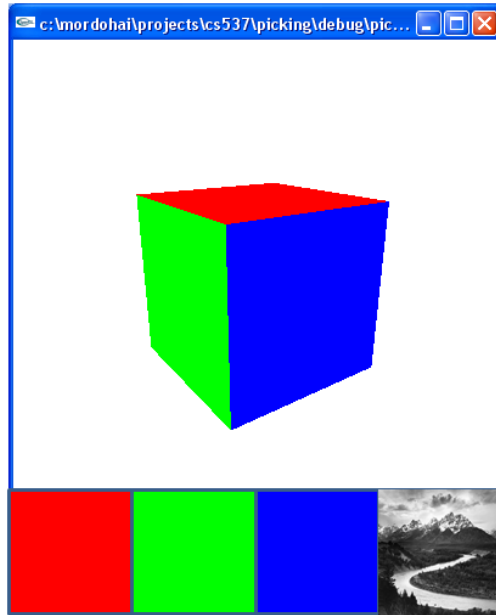


Figure 1: Example of the window that has to be generated by the program.

3. Implement picking so that the user can select one of the faces. Once a face has been selected, the user can click on one of the panels to select the color or texture. Implementation details, such as whether a face is active at start-up or if all faces are deselected if the user clicks on the background, are up to each of you. All reasonable choices will be accepted. **(25%)**
4. Once a face and a color or texture have been selected, update the appearance of the selected face. Make sure that texture images appear upright. **(25%)**
5. Write a brief report explaining what you did, emphasizing aspects that are not obvious from running or looking at the code. If something does not work, give possible explanations. **(30%)**

Hints:

- You can use either orthographic or perspective projection. Also, assume that the window cannot be reshaped.
- One possible implementation is to write two polygon display functions: one that displays polygons in solid color and one that maps a texture image onto them. Then, call the appropriate function according to the state of each polygon.
- Picking using selection mode or bounding boxes can be used for the panels. Selection mode is necessary for the faces of the cube. For the panels at the bottom you can use a command like `glDrawPixels(im_w, im_h, GL_RGB, GL_UNSIGNED_BYTE, image)` and then

decide where the user clicked by reading the mouse coordinates. Make sure that the y coordinate is inverted when necessary.

- The width and height of the texture image must be multiples of 2 and larger than 64.

Resources:

- Red Book Chapter 9 on texture mapping. (<http://www.glprogramming.com/red/chapter09.html>)
- See how a cube with texture mapping is specified in `tex_cube.c` that can be found on the author's website: http://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/FIFTH_EDITION/PROGRAMS/CHAPTER08/
- The picking example using selection mode that I showed in class can be found at <http://gpwiki.org/index.php/OpenGL:Tutorials:Picking>
- A brief tutorial on color-based picking can be found at <http://www.lighthouse3d.com/opengl/picking/index.php3?color1>
- I have provided very simple functions to read and write ppm images on the class webpage. The functions do not check the validity of the inputs, other than the existence of the image file. To read an image include the header file and use the following:

```
GLubyte *image;  
image = read_ppm_image("Stevens_logo.ppm", &im_w, &im_h);
```

- Image libraries compatible with OpenGL that can handle several formats include:
 - The Simple OpenGL Image Library <http://lonesock.net/soil.html>
 - The Developer's Image Library (DevIL, formerly known as OpenIL) <http://openil.sourceforge.net/>
- Images can be converted to ppm using IrfanView (Windows) or `convert` (Unix/Linux).