

# Assignment 3      COMP 557

Instructor:            Prof. Michael Langer  
Teaching Assistant:    Fahim Mannan  
Posted:                Friday, Nov. 7, 2008 (part 1), modified Sunday, Nov. 16  
Due:                    Friday, Nov. 28, 2008 at midnight (part 1)

## Overview

- There are two parts to this assignment, both of which involve texture mapping and rendering (lighting and material). The first involves environment mapping. The second applies texture mapping to terrains and to the problem of level of detail. *The second part will be posted during the week of Nov. 17-21.*
- *Late assignments will be penalized by 4 points per day.*
- Hand-in instructions are the same as in Assignment 1.

## Part 1: Environment Mapping

### Introduction

You are given a room whose walls are texture mapped. You are also given a mirror object that is placed in the center of the room. (Use key 'm' to toggle the presence of the mirror object.)

The environment map for the mirror object is a cube map, and is initialized with textures that are independent of the given room. That is, the initial environment map is *incorrect*.

### Requirements

#### 1. (15 points)

Design your own room, which has a visually identifiable character. It could be a bedroom, a kitchen, a restaurant, etc. The lighting, materials, and objects should be chosen appropriately for your room. In particular, you must:

- Replace the textures on the walls, floor and ceiling. e.g. you could replace the walls by a uniform reflectance (paint) and add paintings to the room by texture mapping a photograph to part of a wall;
- add objects to the room, including light sources and furniture.

Your room should be contained in the function `drawSceneNoMirror()`.

This question is an opportunity for you to experiment with various light source and material properties in OpenGL. Have fun!

### Hint:

As I mention below, starter code for lights and material can be found online from Angel's website.

For texture mapping a real image onto a rectangle, see Angel's book (Sec. 8.11) and accompanying code from on his website (see URL below). This code shows how to texture map a PPM image onto a rectangle. If you would like to texture map a JPG image, rather than a PPM, then an easy way to do this is to use the unix/linux command `convert` i.e.

```
convert -compress none myimage.jpg myimage.ppm.
```

If you don't include the `-compress none` option, then it will store the file in "P6" rather than "P3" format, and Angel's code won't work.

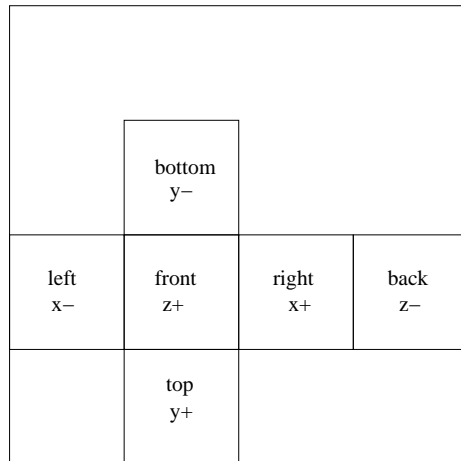
### 2. (20 points)

Compute a *correct* environment map of the room, using a cube map representation, by filling in the function `makeEnvMap()`. You should render the six images corresponding to the faces of the cube map in six separate viewports, all within the right viewport of the window. You should then use the OpenGL function `glReadPixels` to read from these six viewports into the cubemap array i.e. write over the incorrect environment map.

An example solution is shown in executable file 'room\_solution'. For this example solution, the faces of the cube in the right viewport are shown below. In particular, note that the cubemap is flipped: "bottom" is on top and "top" is on the bottom and, when you run the code, you will notice that the faces in the viewport are also upside down. This inversion is not an error. It just turned out to be the easiest way (for me, at least) to get the cubemap to work correctly. Such up-down inversions are quite common in image manipulation, as I have mentioned earlier in the course. (Recall lecture 7, p. 1.)

### Hint:

You may find you need to play with the transformations to get the cube map to work properly. In particular, to flip images left-right or up-down, you can use `glScalef()` with the scale factor of the flipped coordinate set to -1. e.g. `glScalef(-1.0, 1.0, 1.0)` flips left-right.



## Starter Code:

The file `room.c` contains several functions, including:

- `drawMirrorObject()` which reads a “cube map” from an array in main memory, and environment maps it onto the mirror object.
- `draw3DScene()` which calls `drawSceneNoMirror()` and `drawMirrorObject()`.

In addition,

- the file `main.c` contains the keyboard and mouse functions and the initialization code. You should examine this code to see what it does. In particular, notice that you are able to translate and rotate the mirror object as well the camera.
- We strongly recommend that you see example code (Chapter 6) at [http://www.cs.unm.edu/~angel/BOOK/PRIMER/THIRD\\_EDITION/PROGRAMS](http://www.cs.unm.edu/~angel/BOOK/PRIMER/THIRD_EDITION/PROGRAMS) to get you started with lighting and material, and see Chapter 8 for examples on how to do texture mapping using a given image, in particular, see example `ppmtex.c`.

*Note that environment mapping is not covered in Angel’s book “OpenGL: a primer”. For this reason, we have provided you with the starter code you need.*