B581 — Computer Graphics, Fall 2008 (A. Hanson)

Problem Set 5, Project Instructions for PS6

Due by Midnight, Tuesday 25 November 2008.
This problem set has 40 points.
50 percent maximum score on any unauthorized late submission.

Submission Instructions: The homework will be graded on a Linux system. Put all your files into one `tar` file, do not upload files individually. Do NOT send executables; they will be returned ungraded for resubmission. GRADING is based on this written problem statement, not on demonstration code that may be shown or made available. Any demos shown are for illustration only.

Submit a working Linux version with a Linux Makefile and the complete source code *(no executables)* in a tar file as follows:
Use "tar cvf ps$< N >< login >$.tar *" to create a tar file.
$< N >$ – the number of the assignment
$< login >$ – your user name
For example: A student whose username is xxyy works on ps5. The submitted file should be called `ps5xxyy.tar`.

We will work in `(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH)` mode, except that the template code now has a simple lighting, material and texture model to replace the plain colors we used until now. You are expected to take responsibility for looking at the *OpenGL manual* sections on Projection utilities and glMaterial if you need further information; *glMaterial must be used to replace glColor when lighting is used.* You should redraw the entire scene with each redisplay and use glutSwapBuffers(). It is suggested that you look carefully at the *lightposition.linux* , *lightmaterial.linux* , and *texture.linux* demonstrations in `/l/b581/OGLtutor/`. You can get current versions from the web for Windows and MAC by going to `http://www.xmission.com/~nate/tutors.html`.

OFF file survival kit. The provided files include enough to get started on an OFF reader and interpreter. The information is provided in the readoff.h and readoff.c files, which will read a bare OFF file, but do not have normals supported. See also the lecture notes and the `3D-handout-OFF.pdf` descriptions in the `b581/problems/` directory.

Texture survival kit. If you want to start experimenting with texture,the readppm.h, readppm.c, and texture.hints files are provided as examples and utilities for working with texture. Some simple examples are integrated into the template. See the OGL manual and the *texture.linux* demonstration in `/l/b581/OGLtutor/`.

This piece of code *below* in the template is important: this is how you locate your own texture files (as many as you want) and make them available *without* including them with your code submission: **getppm.pl** is a simple Perl script that obtains binary PPM image file from the Web, puts it into the file *temp.ppm*, and allows you to use it in your program:

Getting your texture images from the Web, supposing no OFF files:

```
if (*argc == 2) {
    /* load image from URL specified on command line */
    sprintf(s,"./getppm.pl %s > temp.ppm", argv[1]);
    printf("Executing:\n   %s\n", s);
    system(s);
    i = ReadPPM("temp.ppm", &TMap);
} else if (*argc == 1) {
    /* load image from default URL */
    printf("Using default texture file 'fermat_raw.ppm'.\n");
    system("./getppm.pl URL.../fermat_raw.ppm > temp.ppm");
    i = ReadPPM("temp.ppm", &TMap);
}
```

Getting an OFF file into your program, supposing no textures:

```
Object3D object;
Point3 *facenormals;

/* Read in a file, no textures */

if (*argc == 2)
    printf("Reading in %s.\n", argv[1]);
    ReadOFF(argv[1], &object);
       }
  else if (*argc == 1)
    { printf("Using default object.\n");
      ReadOFF("cube.off", &object);
    }
   /* Hint: here is how to start face normal procedure. */
   facenormals = (Point3 *) malloc(object.Nfaces*sizeof(Point3));
```

5.1 **3D Models, Interaction with normals and lighting.** A template file is provided that sets up a sample 3D scene with predefined normals and materials. From that example, you should be able to start on your own simple scene that displays one square with a normal
(e.g. `glNormal3f(0.0,0.0,1.0); glRectf(-1.0,-1.0,1.0,1.0);`) and start the geometry reader, described next:

Hardcoded OFF display with one hardcoded texture. Begin by looking at the template and example reading utilities in the provided template files `readoff.c, readoff.h`. It is suggested that you first create a MAIN program with no arguments expected, and hardcode an OFF file to load, and use the default checkerboard texture or hardcode a texture file to load. Set a default glMaterial specification for the object; for the FIRST polygon, *use a texture instead of a material*. you will compute face normals by hand and specify them using `glNormal()`. Read the OpenGL manual and its examples to define *ambient and diffuse material* colors that replace the use of `glColor3f` for objects that respond to lighting, and incorporate a simple texture.

a. (20 points) **TASK: OFF Model Reading, Decoding, and Display.**

   Implement a Display callback that interprets the OFF object file format by converting the data into OpenGL geometry and attaching normals, using glNormal3f, and rendering the object on the screen. Note: glNormal3f() works pretty much like glVertex3f(), except that it is very highly advisable to use *unit vectors* so the normalization of the lighting equations is not rescaled.

   Minimal Object: A sample OFF file is given below:

```
OFF
8 6 12
 1.0   1.0   1.0
-1.0   1.0   1.0
-1.0  -1.0   1.0
 1.0  -1.0   1.0
 1.0   1.0  -1.0
-1.0   1.0  -1.0
-1.0  -1.0  -1.0
 1.0  -1.0  -1.0
4  0 1 2 3        #  THIS ONE should have a texture on it
4  7 6 5 4        #  The rest have a shared material...
4  0 3 7 4
4  3 2 6 7
4  2 1 5 6
4  0 4 5 1
```

   Note: `readoff.c` uses the utilities in `<stdio.h>` and `<string.h>`, including `fopen`, `fgets`, `strcmp`, `fread`, and `fclose`. `strcmp` is especially tricky in the C libraries since it returns ZERO when successful.

b. (5 points) **Accept arbitrary OFF files and textures.** Once you have the interpreter for OFF files working, change your main program so it takes one command line argument that is the name of an OFF file and an optional second argument

that is the name of a texture URL. If no second argument is provided, use the checkerboard texture.

c. (5 points) **Create your own OFF file.** Your model should contain one or more unique 3D objects consisting of triangles or flat polygons; the first polygon should be a square to make the texture (required only on the first polygon) look better.

d. (5 points) **3D Interaction.** Implement 3D rotation about the center of mass, translate in XY and in XZ using the mouse buttons as indicated below .

User Interface: When the Menu interaction mode is `MY_MENU_OBJECT`, move the object as follows in *user* coordinates:

* **Left Mouse Drag object: screen space** (x, y) **translation.**
* **Shift-Left-Mouse-Drag: motion of whole object in user** z**-direction** with the Shift modifier depressed.
* **Middle Mouse Drag: Rolling ball or Virtual Sphere rotation.**
* **Shift-Middle Mouse Drag: Rotate about the** $\hat{\mathrm{z}}$ **axis** with the Shift modifier depressed.

5.2. (5 points) **Start on your PS6 project.** Attached at the end is a description of the kind of projects you should be considering for your last coding assignment.

* **Write a proposal.** Include a plain text "mylogin-ps6-proposal.txt" file with your submission. If you have graphics, use a pdf file instead of plain text. Do not send proprietary-format Microsoft Word files.
* **Write some code.** A dummy Menu item is provided for the PROJECT mode; write some small piece of code that illustrates what you are thinking about for your project.

Notes: `MouseModifiers = glutGetModifiers();` generates a bitmask with `GLUT_ACTIVE_SHIFT, GLUT_ACTIVE_CTRL, GLUT_ACTIVE_ALT` ; you can only call this in the mouse button routine, so the global variable needs to be set to be checked in other routines. The mouse buttons are recorded in `mouseButtons` for later use. All transformations must be done relative to the USER frame, that is left-multiplied times the existing orientation and position. Use the Fixed Point Rule with the new center defined by each Translation.

YOUR PS6 PROJECT instructions:

Navigating a Knowledge Space or Cognitive Challenge

1. Choose a topic which is related to the navigation of virtual 3D world in which you are searching for some kind of real or simulated knowledge or information, or a cognitive challenge such as a 3D extension of the PS4 "bug-catcher." (2-person teams are permitted, but not highly recommended: you will need two distinct pieces of the topic for a team, separately gradable, and each of comparable content to a one-person project).
2. *Minimal requirements:* The application must involve texture, must support 3D navigation, and must involve some kind of 3D interaction with the scene material.
3. *Suggestions — knowledge:* There are many ways that you can represent some kind of simple personal or general knowledge, and you can choose the level of complexity that suits you. Examples include a kind of a "museum" in which you might organize photos of family and friends, a time-line that includes an overview of the courses you have taken, a sequence of historical events in history laid out in a topical as well as temporal fashion in the navigation, or covers of your favorite books organized in some way that groups them in space by their categories.
4. *Suggestions — cognition:* Computer graphics is well suited to creating puzzling situations, where the computer knows what is going on but you don't. Trying to find something that is trying to escape you, or interfering with the normal expected behavior of the interface can create a challenging learning situation for the user (that is essentially what we did in the PS4 bug-catcher).
5. *Interface.* There are literally an infinite number of ways you could navigate through any particular virtual world; look for something interesting, maybe with some analog to a real-world mechanical device.
6. *Tasks:* (1) Your goal in the PS6 project is to write a piece of code that implements a major task as described above, with mouse adjustment of view, parameters, etc. (2) Provide written documentation (plain text or pdf) of several pages explaining the topic you chose and your experience implementing it.

Some resources

The Siggraph conference proceedings:

`http://www.cs.brown.edu/~tor/sig2008.html`

`http://www.cs.brown.edu/~tor/sig2007.html`

`http://www.cs.brown.edu/~tor/sig2006.html`

`http://www.cs.brown.edu/~tor/sig2005.html`

`http://www.cs.brown.edu/~tor/sig2004.html`

etc. etc.

Here are some button-box interfaces you can make use of if you like:

GLUI: You can choose to use a public system called GLUI ("GL User Interface"). Documentation is linked to the syllabus in OnCourse, but can be found directly on `/l/b581/docs/glui-doc.pdf`. The include file `glui.h` and the libraries are in `/l/b581/include/` and `/l/b581/lib/linux`, respectively.

Qt: If you would like to use the more sophisticated Qt interface, you are welcome to do so. The OnCourse *Syllabus* section has some pointers, and the direct pointers to the URL's are `https://www.cs.indiana.edu/classes/b581/qt/` for basics, and `https://www.cs.indiana.edu/classes/b581/qt/template` for a template including instructions and sample code.
