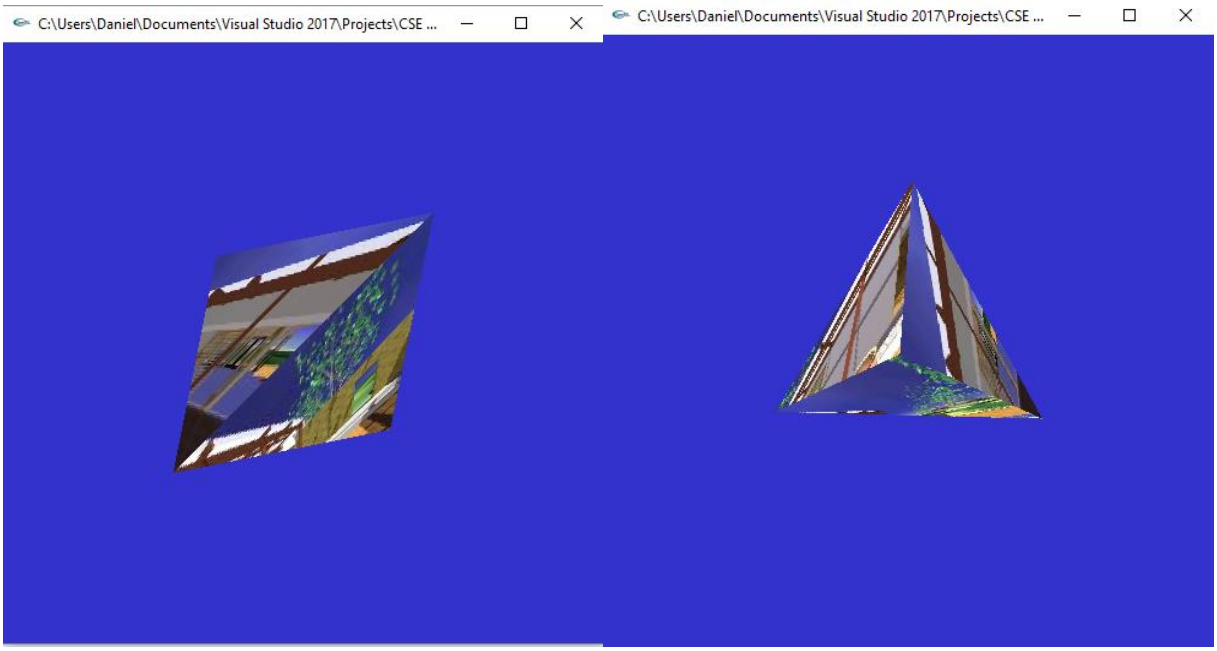
Daniel Meyer

CSE 520

Tong Yu

Lab 8 Report




```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
//#include “imageio.h”
#include <SOIL/SOIL.h>
#include <GL/glut.h>

#define PI 3.14159265359

int texImageWidth;
int texImageHeight;
int window;
static GLuint texName[6];              //texture names
int anglex = 0, angley = 0, anglez = 0;          //rotation angles
float xdiff = 0.0, ydiff = 0.0, zdiff = 0.0;
bool mouseDown = false;

char maps[][20] = { "cubemap_fr.png",  "cubemap_bk.png",
"cubemap_rt.png", "cubemap_lf.png",
            "cubemap_up.png", "cubemap_dn.png" };
```

```
//load texture image
GLubyte *makeTexImage(char *loadfile)
{
	int i, j, c, width, height;
	GLubyte *texImage;
	//texImage = loadImageRGBA( (char *) loadfile, &width, &height);
	texImage = SOIL_load_image(loadfile, &width, &height, 0,
SOIL_LOAD_RGBA);
	texImageWidth = width;
	texImageHeight = height;

	return texImage;
}

void init(void)
{
	glClearColor(0.2, 0.2, 0.8, 0.0);
	glShadeModel(GL_FLAT);

	glEnable(GL_DEPTH_TEST);

	glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
	//texName is global
	glGenTextures(6, texName);
	for (int i = 0; i < 4; ++i) {
		GLubyte *texImage = makeTexImage(maps[i]);
		if (!texImage) {
			printf("\nError reading %s \n", maps[i]);
			continue;
		}
		glBindTexture(GL_TEXTURE_2D, texName[i]);		//now we
work on texName
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_REPEAT);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_REPEAT);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_NEAREST);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
		glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, texImageWidth,
			texImageHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE,
texImage);

		delete texImage;			//free memory holding
texture image
```

```
	}
}

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glEnable(GL_TEXTURE_2D);
	glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
	glEnable(GL_CULL_FACE);
	glCullFace(GL_BACK);

	glPushMatrix();
	glRotatef(anglex, 1.0, 0.0, 0.0);		//rotate the
cube along x-axis
	glRotatef(angley, 0.0, 1.0, 0.0);		//rotate along
y-axis
	glRotatef(anglez, 0.0, 0.0, 1.0);		//rotate along
z-axis

	glBindTexture(GL_TEXTURE_2D, texName[0]);
	glBegin(GL_TRIANGLES);
	//front triangle
	glTexCoord2f(0.0, 0.0);		glVertex3f(-1.0f, 1.0f, -1.0f);
	glTexCoord2f(1.0, 0.0);		glVertex3f(1.0f, -1.0f, -1.0f);
	glTexCoord2f(0.0, 1.0);		glVertex3f(-1.0f, -1.0f, 1.0f);
	glEnd();

	glBindTexture(GL_TEXTURE_2D, texName[1]);
	glBegin(GL_TRIANGLES);
	//right side triangle
	glTexCoord2f(0.0, 0.0);		glVertex3f(1.0f, 1.0f, 1.0f);
	glTexCoord2f(1.0, 0.0);		glVertex3f(-1.0f, -1.0f, 1.0f);
	glTexCoord2f(0.0, 1.0);		glVertex3f(1.0f, -1.0f, -1.0f);
	glEnd();

	glBindTexture(GL_TEXTURE_2D, texName[2]);
	glBegin(GL_TRIANGLES);
	//left side triangle
	glTexCoord2f(0.0, 0.0);		glVertex3f(1.0f, 1.0f, 1.0f);
	glTexCoord2f(1.0, 0.0);		glVertex3f(-1.0f, 1.0f, -1.0f);
	glTexCoord2f(0.0, 1.0);		glVertex3f(-1.0f, -1.0f, 1.0f);
	glEnd();

	glBindTexture(GL_TEXTURE_2D, texName[3]);
	glBegin(GL_TRIANGLES);
	//bottom triangle
```

```
	glTexCoord2f(0.0, 0.0);		glVertex3f(1.0f, 1.0f, 1.0f);
	glTexCoord2f(1.0, 0.0);		glVertex3f(1.0f, -1.0f, -1.0f);
	glTexCoord2f(0.0, 1.0);		glVertex3f(-1.0f, 1.0f, -1.0f);
	glEnd();
	glPopMatrix();

	glFlush();
	glDisable(GL_TEXTURE_2D);
}

void keyboard(unsigned char key, int x, int y)
{
	switch (key) {
	case 'x':
		anglex = (anglex + 3) % 360;
		break;
	case 'X':
		anglex = (anglex - 3) % 360;
		break;
	case 'y':
		angley = (angley + 3) % 360;
		break;
	case 'Y':
		angley = (angley - 3) % 360;
		break;
	case 'z':
		anglez = (anglez + 3) % 360;
		break;
	case 'Z':
		anglez = (anglez - 3) % 360;
		break;
	case 27: /* escape */
		glutDestroyWindow(window);
		exit(0);
	}
	glutPostRedisplay();
}

void mouse(int button, int state, int x, int y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		mouseDown = true;
		xdiff = x - angley;
		ydiff = -y + anglex;
	}
```

```
	/*
	//For XYZ rotation using different mouse buttons Pt.2
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		//mouseDown = true;
		mouseDown = 1;
		xdiff = x - yrot;
		//ydiff = -y + xrot;
	}
	else if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		//mouseDown = true;
		mouseDown = 2;
		ydiff = -y + xrot;
	}
	else if (button == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
	{
		mouseDown = 3;
		//zdiff = x - zrot;
		anglez += 5.0 * (PI / 180);
	}
	else
	{
		mouseDown = 0;
		//mouseDown = false;
	}
	*/
}

void mouseMotion(int x, int y)
{
	if (mouseDown == true)
	{
		angley = x - xdiff * (PI / 180);
		anglex = y + ydiff * (PI / 180);

		glutPostRedisplay();
	}

	/*
	//For XYZ rotation using different mouse buttons Pt.3
	if (mouseDown == 1)
	{
		yrot = x - xdiff * (PI / 180);
		//xrot = y + ydiff * (PI / 180);
```

```
            //glutPostRedisplay();
        }
        else if (mouseDown == 2)
        {
            xrot = y + ydiff * (PI / 180);
        }
        else if (mouseDown == 3)
        {
            zrot = x - zdiff * (PI / 180);
        }
        glutPostRedisplay();
        */
}

void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 1.0, 30.0);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);
}


int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        window = glutCreateWindow(argv[0]);
        init();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutKeyboardFunc(keyboard);
        glutMouseFunc(mouse);
        glutMotionFunc(mouseMotion);
        glutMainLoop();
        return 0;
}
```

Summary:

This assignment was similar to Part 1 of Homework 3, but with a different object type. Instead of creating a 6-sided textured cube we were to create a 4-sided Tetrahedron which involves triangles. As such I modified the code used for Homework 3 to use only 4 different images as opposed to 6, changed the object type to triangles instead of quads, and finally changed the texture coordinates to match a triangle (using only 3 as opposed to 4). Overall, the program compiled and ran correctly with both key and mouse rotation and I feel I have earned the full 20 points for the assignment.