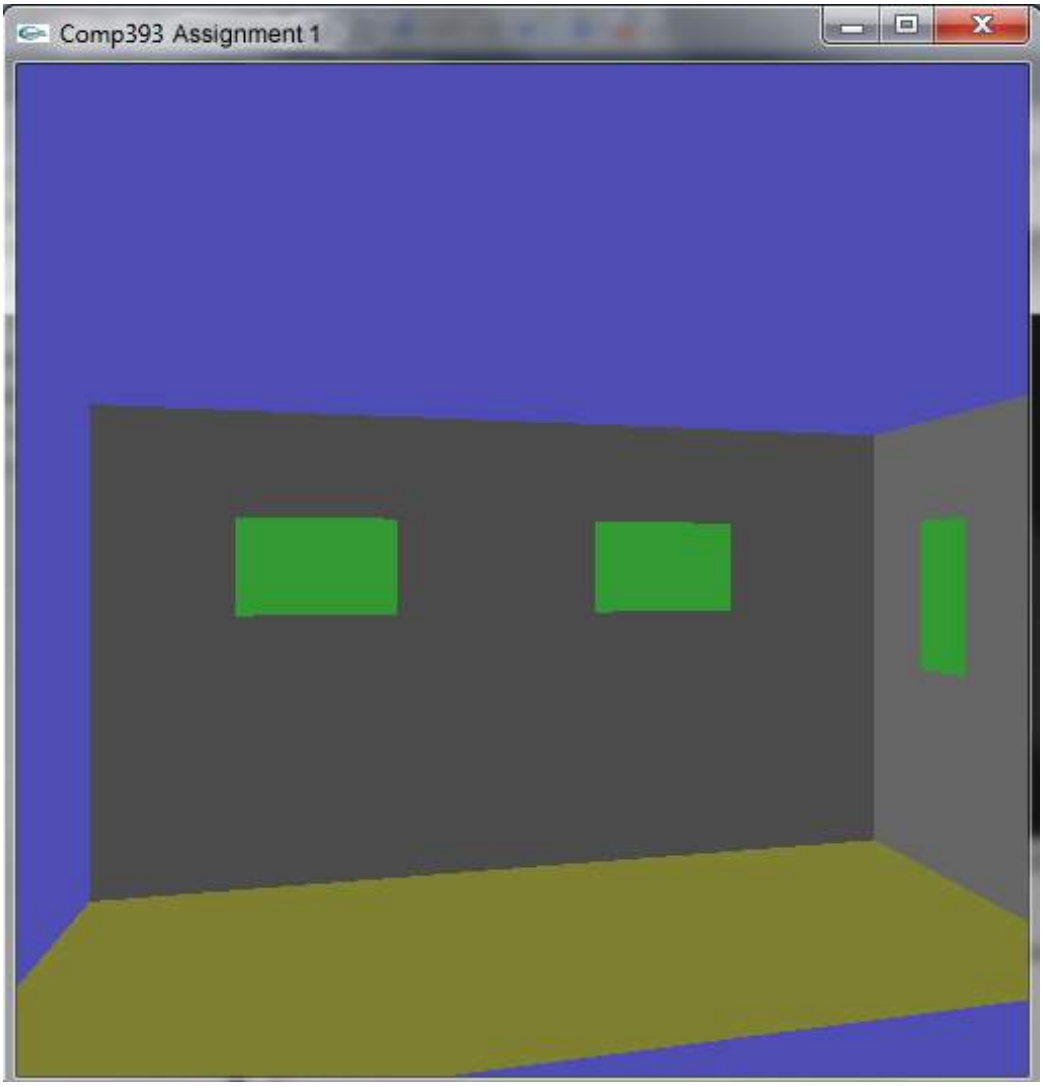
Assignment 1: Surface Rendering

Weight: 5% of your final grade

Due: After Unit 5

The idea of this assignment is to apply the texturing techniques discussed in Unit 5.

The following is a simple object model of a scene:



Requirements

1. The source program that renders the above image is given below. Note that all coordinates are hard coded. Feel free to modify the program using rendering routines.

2. Apply a calculated texture to the two walls (greyish colors). You can use the brick pattern discussed in Unit 5.

3. Apply a calculated texture to the floor (yellowish color) that resembles a laminated wooden floor. You can change the brick size, color, and percentage to mimic the effect.

4. Load three textures to the three smaller green rectangles on the walls. You can use any images that you can find. At least one image must be applied using multiple textures.

☞ **Important**: Respect copyright when you use images from the Web. The safest bet is to use images from the public domain or those with a Creative Commons license. AU students also have access to ArtStor images. Contact AU Library Services for information.

Deliverables

1. All source code.
2. The images that you used.
3. Snapshots of the following:
   a. an overall image that captures all objects
   b. one close-up that shows the texture of both walls
   c. one close-up that shows the texture of the floor
   d. three close-ups that show the images on the walls

☞ **Important**: Compress all files into *one single zip file,* and upload it here to submit for marking and feedback from your tutor.

Grading Criteria

Marks will be assigned as follows:

- Visual effect – 50 marks
- Codes (including application setup, vertex, and fragment shaders) – 40 marks
- Documentation of the program – 10 marks

Source Program

```
// Athabasca University
// Unit 5 Assignment 1

#include <Windows.h>
#include "gl/glut.h"
```

```
void initialize(int argc, char * argv[]) {

        // set background color
        glClearColor(0.3, 0.3, 0.7, 0.0);

        // enable depth test
        glEnable(GL_DEPTH_TEST);
}

void drawPicture() {

        glColor3f(1.0, 1.0, 1.0);

        // wall 1
        glColor3f(0.3, 0.3, 0.3);
        glBegin( GL_POLYGON );
                glVertex3f(-8.0, -6.0, 0.0);
                glVertex3f(-8.0, 3.0, 0.0);
                glVertex3f(8.0, 3.0, 0.0);
                glVertex3f(8.0, -6.0, 0.0);
        glEnd();

        // wall 2
        glColor3f(0.4, 0.4, 0.4);
        glBegin( GL_POLYGON );
                glVertex3f(8.0, -6.0, 0.0);
                glVertex3f(8.0, 3.0, 0.0);
                glVertex3f(8.0, 3.0, 6.0);
                glVertex3f(8.0, -6.0, 6.0);
        glEnd();

        // floor
        glColor3f(0.5, 0.5, 0.2);
        glBegin( GL_POLYGON );
                glVertex3i(-8.0, -6.0, 6.0);
                glVertex3i(-8.0, -6.0, 0.0);
                glVertex3i(10.0, -6.0, 0.0);
                glVertex3i(10.0, -6.0, 6.0);
        glEnd();

        // frame 1
        glColor3f(0.2, 0.6, 0.2);
        glBegin( GL_POLYGON );
                glVertex3f(-5.5, -0.854, 0.05);
                glVertex3f(-5.5, 1.0, 0.05);
                glVertex3f(-2.5, 1.0, 0.05);
                glVertex3f(-2.5, -0.854, 0.05);
        glEnd();

        // frame 2
        glBegin( GL_POLYGON );
                glVertex3f(1.5, -0.854, 0.05);
                glVertex3f(1.5, 1.0, 0.05);
                glVertex3f(4.5, 1.0, 0.05);
                glVertex3f(4.5, -0.854, 0.05);
        glEnd();
```

```
        // frame 3
        glBegin( GL_POLYGON );
                glVertex3f(7.95, -2.0, 1.5);
                glVertex3f(7.95, 1.0, 1.5);
                glVertex3f(7.95, 1.0, 2.646);
                glVertex3f(7.95, -2.0, 2.646);
        glEnd();
}

void display(void) {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        gluLookAt(-3.0, 0.0, 15.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

        drawPicture();

        glutSwapBuffers();
}

void reshape(int w, int h) {
        glViewport(0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 40.0);
        glMatrixMode(GL_MODELVIEW);
}

void main(int argc, char * argv[])
{
        glutInit( &argc, argv );
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH) ;
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);

        int windowHandle = glutCreateWindow("Comp393 U5 Assignment 1");
        glutSetWindow(windowHandle);

        glutDisplayFunc( display );
        glutReshapeFunc( reshape );

        initialize(argc, argv);

    glutMainLoop();
}
```