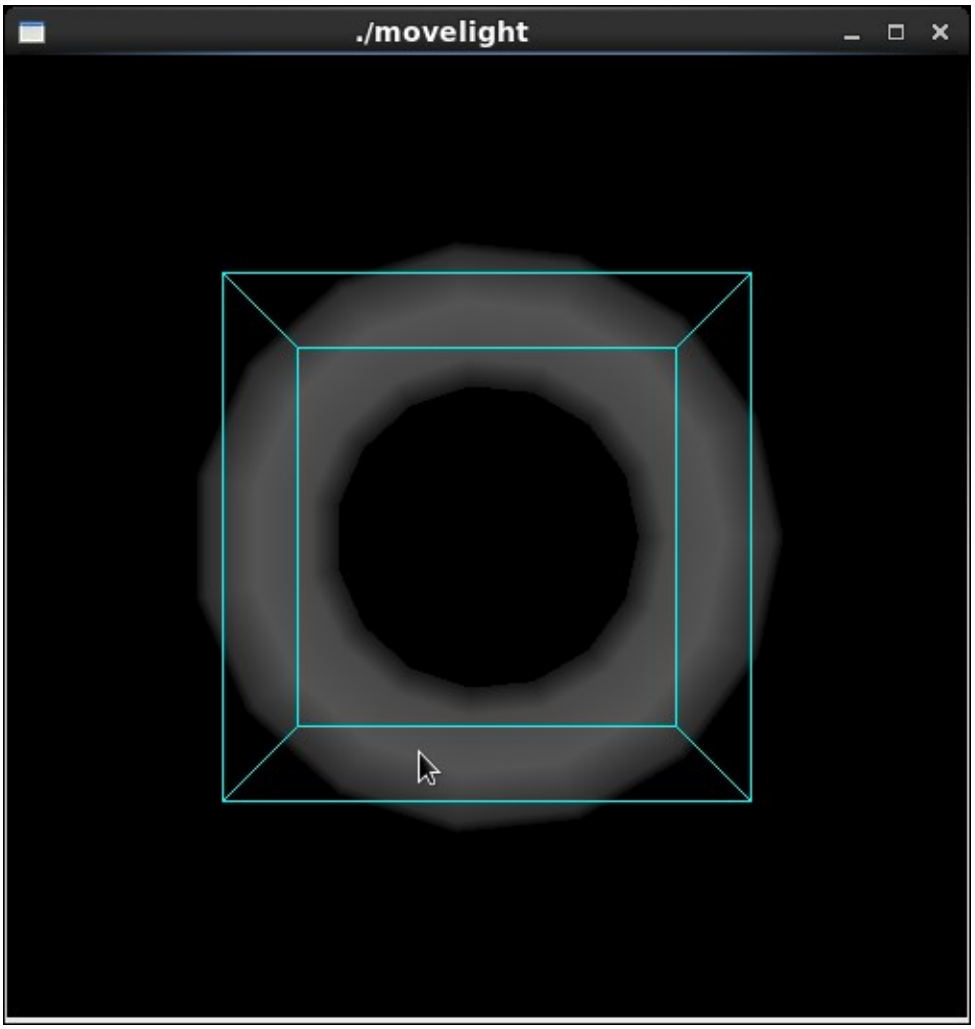
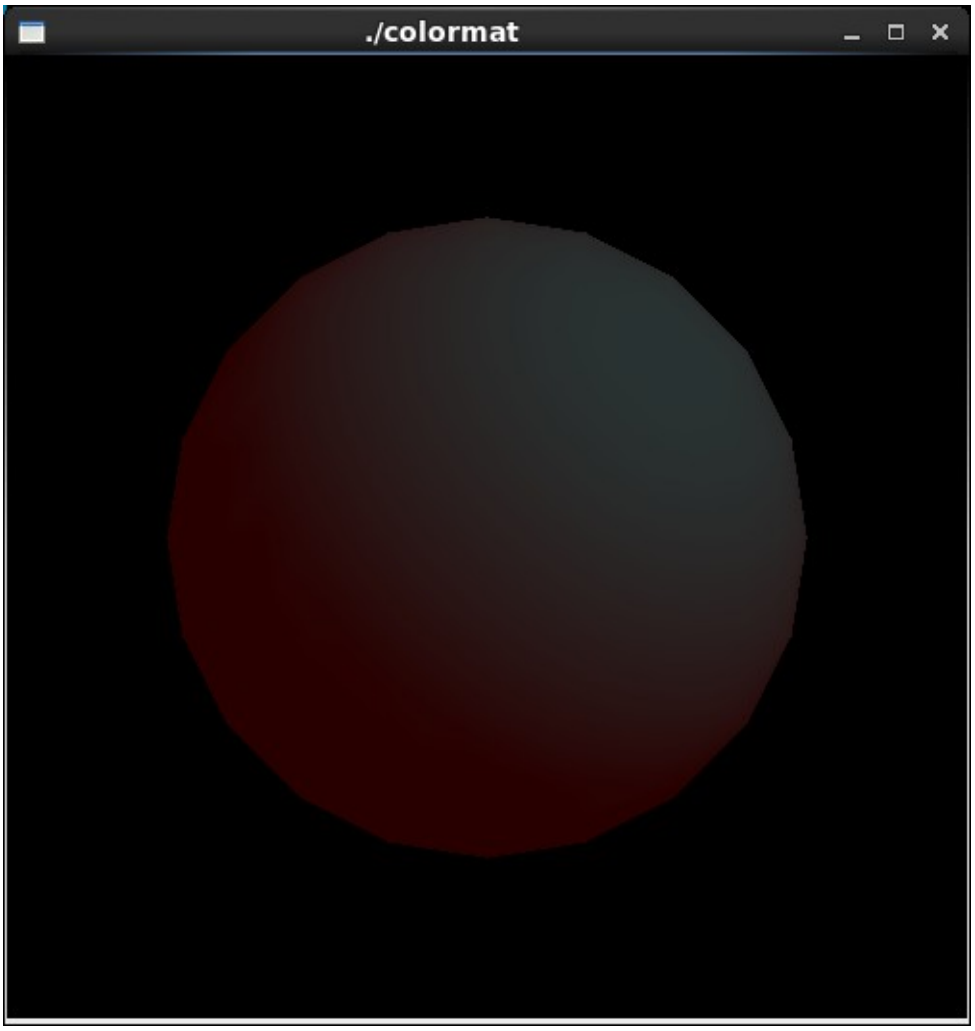
CSE420
Samuel Marrujo
Professor Yu
Lab 16

Lighting

In this part of the lab, it was a modification of the original movelight.cpp and colormat.cpp programs. After some modifications to the program, and adding in the functions asked for, the result was the following reproduction of the figure. The change of various functions ended up producing various different instances of lighting. Because of this change to the functions and the addition of the figure, I believe I was able to accomplish this task successfully, since there were no errors and the display window was changed. Here are my results for this program:




./colormat

```
//movelight.cpp
#include <GL/gl.h>
#include <GL/glut.h>
#include <stdlib.h>

static int z = 0;

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}

/*  Here is where the light position is reset after the modeling
 *  transformation (glRotated) is called.  This places the
 *  light at a new position in world coordinates.  The cube
 *  represents the position of the light.
 */
void display(void)
{
   GLfloat position[] = { 0.0, 0.0, 1.5, 1.0 };
   GLfloat lposition[] = { 0.5, 0.5, 0.5, 0.5 };

   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glPushMatrix ();
   glTranslatef (0.0, 0.0, -5.0);

   glPushMatrix ();
   //glRotated ((GLdouble) spin, 1.0, 0.0, 0.0);
   glTranslatef(0,0,z);
   glLightfv (GL_LIGHT0, GL_POSITION, position);
   glLightfv (GL_LIGHT0, GL_LINEAR_ATTENUATION, lposition);
   glTranslated (0.0, 0.0, 1.5);
   glDisable (GL_LIGHTING);
   glColor3f (0.0, 1.0, 1.0);
   glutWireCube (0.5);
   glEnable (GL_LIGHTING);
   glPopMatrix ();

   glutSolidTorus (0.275, 0.85, 8, 15);
   glPopMatrix ();
   glFlush ();
}

void reshape (int w, int h)
{
```

```
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(40.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
   switch (button) {
      case GLUT_LEFT_BUTTON:
         if (state == GLUT_DOWN) {
            z = (z + 1);
            glutPostRedisplay();
         }
         break;
      case GLUT_RIGHT_BUTTON:
         if (state == GLUT_DOWN) {
            z = (z - 1);
            glutPostRedisplay();
         }
         break;
      default:
         break;
   }
}
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMouseFunc(mouse);
   glutMainLoop();
   return 0;
}
```

```
/*
 *  colormat.cpp
 *  After initialization, the program will be in
 *  ColorMaterial mode.  Interaction:  pressing the
 *  mouse buttons will change the diffuse reflection values.
 */
#include <GL/glut.h>
#include <stdlib.h>

GLfloat diffuseMaterial[4] = { 0.2, 0.2, 0.2, 0.2};
float colors[4] = {0.8, 0, 0.0, 0};

/*  Initialize material property, light source, lighting model,
 *  and depth buffer.
 */
void init(void)
{
   GLfloat mat_specular[] = { 0.5, 0.5, 0.5, 0.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);
   glEnable(GL_DEPTH_TEST);
   //glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseMaterial);
   //glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialf(GL_FRONT, GL_SHININESS, 15.0);
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, colors);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);

   glColorMaterial(GL_FRONT_AND_BACK, GL_DIFFUSE);
   glEnable(GL_COLOR_MATERIAL);
}

void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glutSolidSphere(1.0, 20, 16);
   glFlush ();
}

void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
```

```
    glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
      1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
  else
    glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
      1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
  switch (button) {
    case GLUT_LEFT_BUTTON:
      if (state == GLUT_DOWN) {
        diffuseMaterial[0] += 0.1;
        if (diffuseMaterial[0] > 1.0)
          diffuseMaterial[0] = 0.0;
        glColor4fv(diffuseMaterial);
        glutPostRedisplay();
      }
      break;
    case GLUT_MIDDLE_BUTTON:
      if (state == GLUT_DOWN) {
        diffuseMaterial[1] += 0.1;
        if (diffuseMaterial[1] > 1.0)
          diffuseMaterial[1] = 0.0;
        glColor4fv(diffuseMaterial);
        glutPostRedisplay();
      }
      break;
    case GLUT_RIGHT_BUTTON:
      if (state == GLUT_DOWN) {
        diffuseMaterial[2] += 0.1;
        if (diffuseMaterial[2] > 1.0)
          diffuseMaterial[2] = 0.0;
        glColor4fv(diffuseMaterial);
        glutPostRedisplay();
      }
      break;
    default:
      break;
  }
}

void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
```

```
        break;
    }
}

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