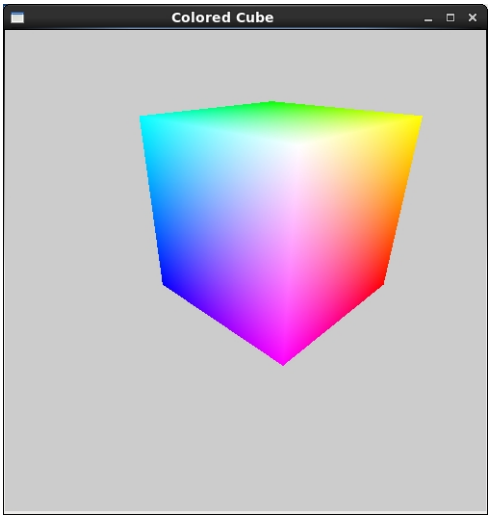
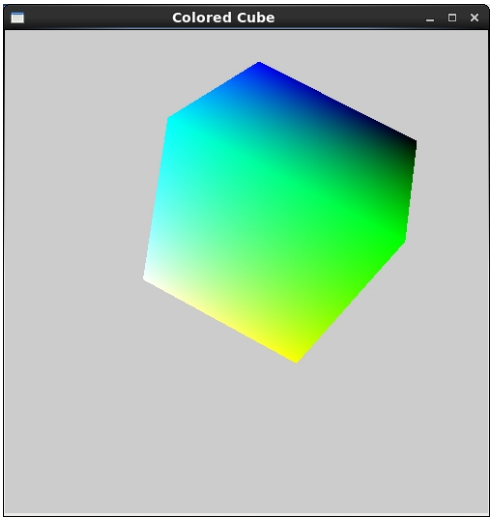
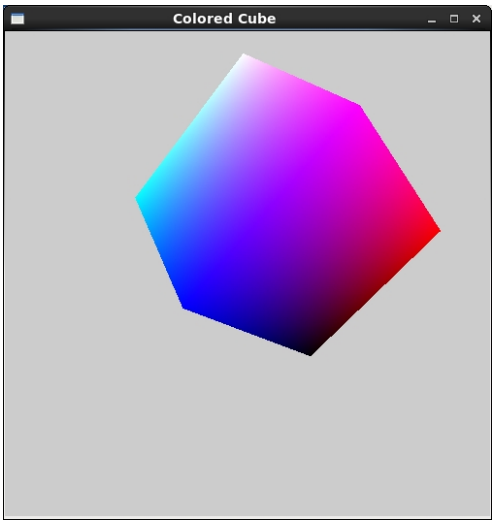
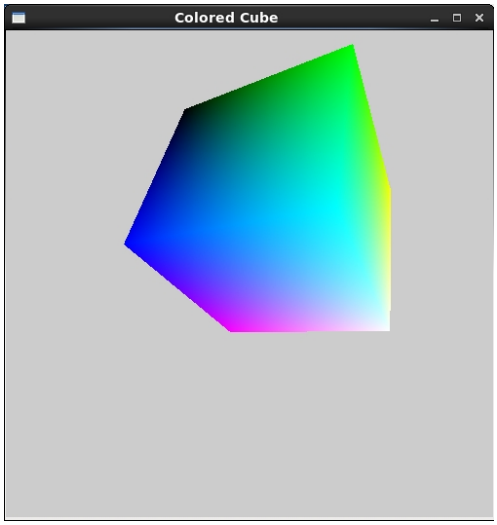
Yazhuo Liu
Homework 4

***1. Rotating Cube***



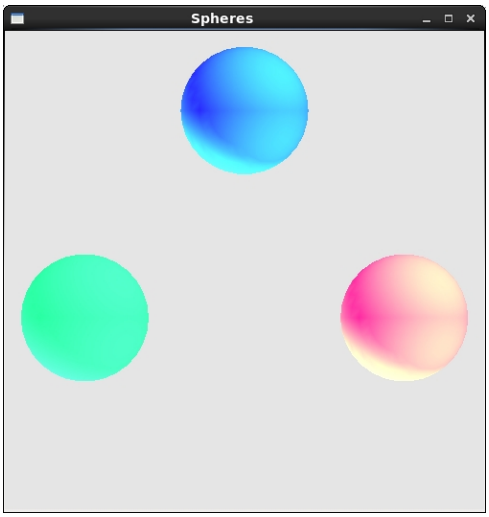




```
//cube.cpp
...
void display(void)
{
  static GLint vertices[] = {-1, -1, -1,   //0
                    1, -1, -1,
                    1, 1, -1,
                    -1, 1, -1,       //3
                          -1, -1, 1,
                          1, -1, 1,
                          1, 1, 1,         //6
                          -1, 1, 1};       //7

  static GLfloat colors[] = {0.0, 0.0, 1.0,
                             0.0, 0.0, 0.0,
```

```
                        0.0, 1.0, 0.0,
                        0.0, 1.0, 1.0,
                        1.0, 0.0, 1.0,
                        1.0, 0.0, 0.0,
                        1.0, 1.0, 0.0,
                        1.0, 1.0, 1.0
                        };

  glClear (GL_COLOR_BUFFER_BIT);
  glColor3f (1.0, 1.0, 1.0);
  glLoadIdentity ();
  gluLookAt (-3.0, 2.5, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  glTranslatef ( 0, 1, 0.5 );
  glRotatef ((GLfloat) xangle, 1.0, 0.0, 0.0);
  glRotatef ((GLfloat) yangle, 0.0, 1.0, 0.0);
  glRotatef ((GLfloat) zangle, 0.0, 0.0, 1.0);

  glVertexPointer (3, GL_INT, 0, vertices);
  glEnable( GL_CULL_FACE );
  glCullFace ( GL_BACK );
  glColorPointer (3, GL_FLOAT, 0, colors);

  static GLubyte allIndices[] = {4, 5, 6, 7, 1, 2, 6, 5,
        0, 1, 5, 4, 0, 3, 2, 1,
        0, 4, 7, 3, 2, 3, 7, 6};

  glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, allIndices);

  glFlush ();
}
...
void keyboard (unsigned char key, int x, int y)
{
  switch (key) {
    case 'x':
      xangle += 5;
      glutPostRedisplay();
      break;
    case 'y':
      yangle += 5;
      glutPostRedisplay();
      break;
    case 'z':
      zangle += 5;
      glutPostRedisplay();
      break;
    case 'X':
      xangle -= 5;
```

```
      glutPostRedisplay();
      break;
    case 'Y':
      yangle -= 5;
      glutPostRedisplay();
      break;
    case 'Z':
      zangle -= 5;
      glutPostRedisplay();
      break;
    case 27:
      exit(0);
      break;
    default:
      break;
  }
}
...
```

2.




```
void init(void)
{
  GLfloat light_position_0[] = { 1.0, 1.0, 1.0, 0.0 };
  GLfloat light_position_1[] = { -1.0, -1.0, -1.0, 0.0 };
  GLfloat light_position_2[] = { 0.0, -1.0, 0.0, 0.0 };
  GLfloat ambientLight[] = { 0.2, 0.2, 1.0, 0.0 };
  GLfloat diffuseLight[] = { 0.2, 1.0, 0.2, 0.0 };
  GLfloat specularLight[] = { 1.0, 1.0, 1.0, 0.0 };
  GLfloat emission[] = {0.3, 0.2, 0.2, 0.0};

  glClearColor (0.9, 0.9, 0.9, 0.9);
  glShadeModel (GL_SMOOTH);
  glEnable(GL_DEPTH_TEST);
```

```
  glMaterialf(GL_FRONT, GL_SHININESS, 25.0);

  glLightfv(GL_LIGHT0, GL_POSITION, light_position_0);
  glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
  glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
  glLightfv(GL_LIGHT0, GL_EMISSION, emission);

  glLightfv(GL_LIGHT1, GL_POSITION, light_position_1);
  glLightfv(GL_LIGHT1, GL_AMBIENT, ambientLight);
  glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseLight);
  glLightfv(GL_LIGHT1, GL_SPECULAR, specularLight);
  glLightfv(GL_LIGHT1, GL_EMISSION, emission);

  glLightfv(GL_LIGHT2, GL_POSITION, light_position_2);
  glLightfv(GL_LIGHT2, GL_AMBIENT, ambientLight);
  glLightfv(GL_LIGHT2, GL_DIFFUSE, diffuseLight);
  glLightfv(GL_LIGHT2, GL_SPECULAR, specularLight);
  glLightfv(GL_LIGHT2, GL_EMISSION, emission);
  glLightf(GL_LIGHT2, GL_SPOT_EXPONENT, 3.0);

  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glEnable(GL_LIGHT1);
  glEnable(GL_LIGHT2);

  glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
  //glColor3f (0.5, 1.0, 2.5);
  glColorMaterial(GL_FRONT, GL_SPECULAR);
  //glColor3f (0.0, 2.5, 2.5);
  glColorMaterial(GL_FRONT, GL_EMISSION);
  //glColor3f (0.5, 0.5, 0.5);
  glEnable(GL_COLOR_MATERIAL);

}

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

  glPushMatrix();

  glPushMatrix();
  glTranslatef (0.0, 1.0, 0.0);
  glColor3f (0.0, 0.0, 1.0);
  glutSolidSphere(0.4, 30, 16);
//   glDisable(GL_LIGHT1);
  glPopMatrix();
```

```
   glPushMatrix();
   glTranslatef (-1.0, -0.3, 0.0);
   glColor3f (0.0, 1.0, 0.0);
   glutSolidSphere(0.4, 30, 16);
//   glEnable(GL_LIGHT1);
//   glDisable(GL_LIGHT0);
   glPopMatrix();

   glPushMatrix();
   glTranslatef (1.0, -0.3, 0.0);
   glColor3f (1.0, 0.0, 0.0);
   glutSolidSphere(0.4, 30, 16);
//   glEnable(GL_LIGHT0);
//   glDisable(GL_LIGHT2);
   glPopMatrix();

   glPopMatrix();
   glFlush ();
}
```

Report:

Problem 1 is pretty easy. I can't figure out how to do problem 2. This is the best I could do.