

**Assignment 3, Deadline for paper copy Tue. 24th March 2009, 12.00noon. Demonstrations must be completed by Mon. 23rd March 2009 at 3.00pm. Worth 25% of CA.**

Late assignments without documented extenuating circumstances **will lose 10% of marks per day or part of day late. Those seven days late will receive zero marks.** If you have extenuating circumstances, you should contact me as soon as possible; in particular, be careful about wasting time on completing an assignment only to hand it in well after the deadline; whether you have valid reasons or not I cannot give any marks for work handed in after outline answers have been handed out or discussed in class.

**Computing Department Rules.** Assignments must typed and neatly presented and spell-checked. Sloppily presented work will be severely penalised and/or returned for correction and resubmission. Assignments must use the department cover sheet and declaration.

Poor layout of programs will be penalised.

**Expect to be questioned about the content of assignments.**

Please use only one side of the page. There is no need for ornate front sheets; simply include the heading above (including my name), and your own name. A staple at the top left corner is the preferred binding — I can staple them if you wish. There is no need for plastic or other covering. Please use the Computing Department cover sheet and signed declaration.

**For numerical answers, where appropriate, show your working; a numerical answer written down with no explanation of working is likely to get zero marks.**

Software will be available in my public folder.

1. Figure 1 shows a cone, and teapot, a sphere and a torus demonstrating ambient, diffuse, and specular lighting and materials. To see the colours, read the PDF version of the assignment on the website; also, I'll supply the pictures (movelight71.png, movelight72.png).

The code is based on `movelight.cpp` (ch06cpp). In this program, one can rotate the light source (shown as the little cube in the images) about any of the three axes.

I have added code to allow interactive switching on and off of lights, camera movement, toggling on and off of camera-mounted light, etc.

```
ESC, Q, q exits
0 switches light0 on and off
1 switches light1 on and off
W, w increases FOV angle (up to 150)
N, n decreases FOV angle (down to 10)
X, x rotates light0 about x-axis
Y, y rotates light0 about y-axis
Z, z rotates light0 about z-axis
A, a decreases x-coordinate of camera
S, s decreases y-coordinate of camera
D, d decreases z-coordinate of camera
```

```
R, r resets to original parameters
```

Submit code (50 marks) and demonstrate (50 marks).

Code etc. at <http://www.jgcampbell.com/graphics1/progs/gp12009a3.zip> and my public folder.

Recreate the program described, and I will supply an exe of the complete program (`movelight7.exe`); `movelight7x.cpp` as supplied is missing some code which you will add.

Properties of the materials of the affected objects and the light should be:

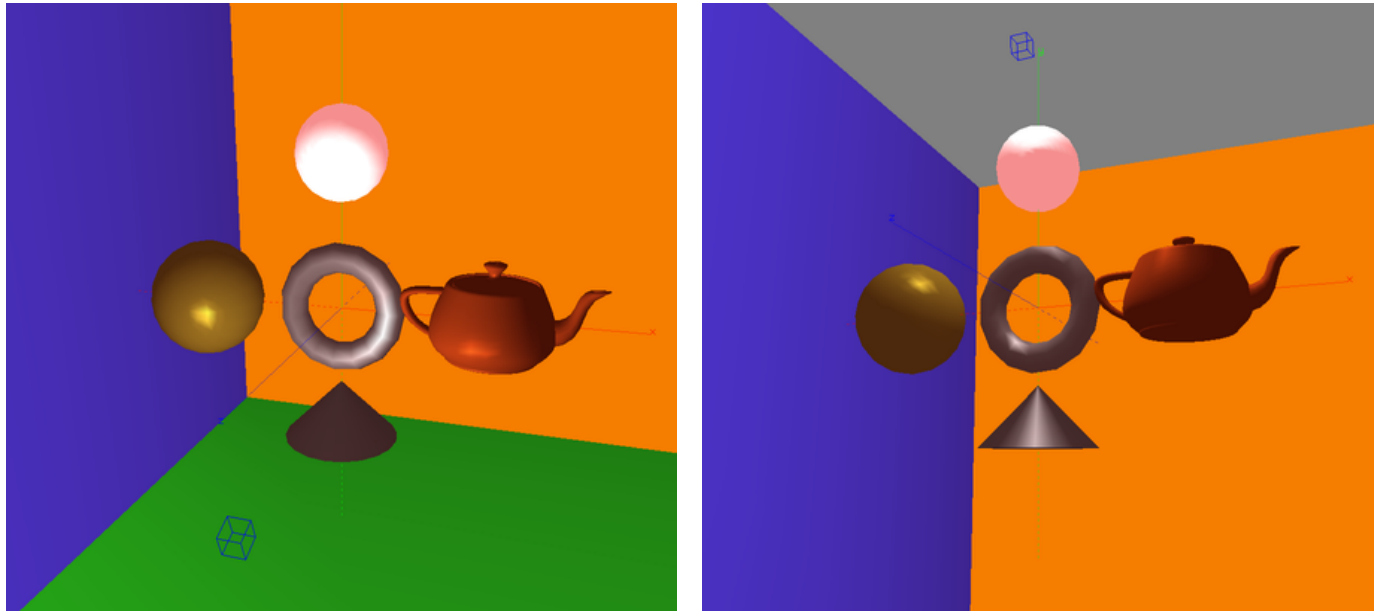


Figure 1: Sphere, cone, teapot, and torus demonstrating lighting.

- Teapot (add the following at //A.

```
// polished copper, see McReynolds p. 51
GLfloat m_am3[] = { 0.2295, 0.08825, 0.0275, 1.0 };
GLfloat m_di3[] = { 0.5508, 0.2118, 0.066, 1.0 };
GLfloat m_sp3[] = { 0.580594, 0.223257, 0.0695701, 1.0 };
GLfloat m_sh3[] = { 51.2 };

glMaterialfv(GL_FRONT, GL_AMBIENT, m_am3);
glMaterialfv(GL_FRONT, GL_DIFFUSE, m_di3);
glMaterialfv(GL_FRONT, GL_SPECULAR, m_sp3);
glMaterialfv(GL_FRONT, GL_SHININESS, m_sh3);
glutSolidTeapot(1.0);
```

- Sphere (add the following at //B.

```
// polished gold, see McReynolds p. 51
GLfloat m_am2[] = { 0.24725, 0.2245, 0.0645, 1.0 };
GLfloat m_di2[] = { 0.34615, 0.3143, 0.0903, 1.0 };
GLfloat m_sp2[] = { 0.797357, 0.723991, 0.208006, 1.0 };
GLfloat m_sh2[] = { 83.2 };

glMaterialfv(GL_FRONT, GL_AMBIENT, m_am2);
glMaterialfv(GL_FRONT, GL_DIFFUSE, m_di2);
glMaterialfv(GL_FRONT, GL_SPECULAR, m_sp2);
glMaterialfv(GL_FRONT, GL_SHININESS, m_sh2);
glutSolidSphere(1.0, 1.0, 20, 20);
```

And now set the materials as we did above for the teapot.

- Draw the cone using:

```
glutSolidCone(1.0, 1.0, 20, 20);
```

- Lights, have already been set up; note there are two, LIGHT0, LIGHT1.

[50 + 50 = 100 marks]

2. Add *emission* to the teapot. 100% of marks for demonstration. If you do not reset the emission properties to zero, all remaining objects will have this emission — see below.

```
GLfloat m_em0[] = { 0.0, 0.0, 0.0, 1.0 }; // already declared
glMaterialfv(GL_FRONT, GL_EMISSION, m_em);
```

[20 marks]

3. 100% of marks for demonstration. Demonstrate making the light (GL\_LIGHT0) *directional*, see notes.

[20 marks]

4. Change the light (GL\_LIGHT0) to be a *spotlight*. 100% of marks for demonstration.

[20 marks]

5. 50% of marks for demonstration. Use the GLUT *timer* function to animate the zooming of the camera lens; continuously move `fovAngle` from its starting point,  $50^\circ$ , up to  $150^\circ$ , and down to  $5^\circ$ . `fovAngle` should move  $5^\circ$  every  $\frac{1}{5}$  of a second, i.e. set the time to go off every 200 milliseconds. 100% of marks for demonstration.

[20 + 20 = 40 marks]

6. Consider now the teapot (copper) and the light; no emissive light; assume that the light is directional and that there is no attenuation whatsoever.

```
// polished copper, see McReynolds p. 51
GLfloat m_am3[] = { 0.2295, 0.08825, 0.0275, 1.0 };
GLfloat m_di3[] = { 0.5508, 0.2118, 0.066, 1.0 };
GLfloat m_sp3[] = { 0.580594, 0.223257, 0.0695701, 1.0 };
GLfloat m_sh3[] = { 51.2 };
```

Light.

```
GLfloat l_am[] = { 1.0, 0.5, 0.5, 1.0 }; // redish ambient light
GLfloat l_di[] = { 0.9, 0.9, 0.9, 1.0 }; // diffuse light
GLfloat l_sp[] = { 0.9, 0.9, 0.9, 1.0 }; // specular light
```

- (a) Compute the ambient colour of the teapot.

[20 marks]

- (b) Assuming that the light is in line with the surface normal, compute the diffuse colour. Include a diagram in your answer.

[20 marks]

- (c) Assuming that the light vector is  $30^\circ$  away from the normal, compute the diffuse colour. Include a diagram in your answer.

[20 marks]

- (d) Assuming that the light vector is  $85^\circ$  away from the normal, compute the diffuse colour. Include a diagram in your answer.

[20 marks]

For questions (e), (f), (g), to make calculations simpler, assume that we have:

```
GLfloat m_sh3[] = { 80.0 };
```

- (e) Assuming that the viewing angle is in line with the mirror reflection angle, compute the specular colour. Include a diagram in your answer.

[20 marks]

- (f) Assuming that the viewing angle is  $30^\circ$  away from the mirror reflection angle, compute the specular colour. Include a diagram in your answer.

[20 marks]

- (g) Using your results from (a), (b) and (e), *note: (a), (b) and (e)*, what would be the total colour as seen by the viewer?

[20 marks]

7. (a) If you create your own objects, for example the solid cube of Chapter 5, and lighting is involved, you need to supply a *normal* for each face / polygon. Explain why.

[10 marks]

- (b) Compute the normals for each of the six faces of the cube given by the following code. See Chapter 6.9.

```
/*
 7 --+ ..... 6 ++-

      back

 4 --- ..... 5 +--

 3 -++ ..... 2  +++

      front

 0 --+ ..... 1 ++-
*/

GLint vv[8][3] = {{-1, -1, 1}, // 0
                  {1, -1, 1}, // 1
                  {1, 1, 1}, // 2
                  {-1, 1, 1}, // 3
                  {-1, -1, -1}, // 4
                  {1, -1, -1}, // 5
                  {1, 1, -1}, // 6
                  {-1, 1, -1} // 7
                 };

void solidCube(GLint vi[][3], GLfloat s){
  GLfloat vf[8][3];
  GLfloat s2 = s*0.5f; // each side is 1.0

  // scale to size, could use glScale, but this is clearer here
  for(int v = 0; v < 8; v++){
    for(int d = 0; d < 3; d++){
      vf[v][d] = GLfloat(vi[v][d])*s2;
    }
  }

  glBegin(GL_QUADS);
  // front
  glColor3f(1.0, 0.0, 0.0);
  glVertex3fv(vf[0]);
  glVertex3fv(vf[1]);
  glVertex3fv(vf[2]);
  glVertex3fv(vf[3]);
```

```

// back but pointing out (back)
glColor3f(0.0, 1.0, 0.0);
glVertex3fv(vf[4]);
glVertex3fv(vf[7]);
glVertex3fv(vf[6]);
glVertex3fv(vf[5]);
// top, pointing up
glColor3f(0.0, 0.0, 1.0);
glVertex3fv(vf[3]);
glVertex3fv(vf[2]);
glVertex3fv(vf[6]);
glVertex3fv(vf[7]);
// bottom, pointing out (down)
glColor3f(0.0, 1.0, 1.0);
glVertex3fv(vf[0]);
glVertex3fv(vf[4]);
glVertex3fv(vf[5]);
glVertex3fv(vf[1]);
// rhs, pointing out
glColor3f(1.0, 1.0, 0.0);
glVertex3fv(vf[1]);
glVertex3fv(vf[5]);
glVertex3fv(vf[6]);
glVertex3fv(vf[2]);
// lhs, pointing out
glColor3f(0.2, 0.2, 0.2);
glVertex3fv(vf[0]);
glVertex3fv(vf[3]);
glVertex3fv(vf[7]);
glVertex3fv(vf[4]);
glEnd();
}

```

[6 × 10 = 60 marks]

8. 100% of marks for demonstration. Create a nice demonstration of OpenGL and lighting of your own based on `movelight7x.cpp` or otherwise.

[50 marks]