

```

/*
 * Copyright (c) 1993-2003, Silicon Graphics, Inc.
 * All Rights Reserved
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose and without fee is hereby granted, provided that the above
 * copyright notice appear in all copies and that both the copyright
 * notice and this permission notice appear in supporting documentation,
 * and that the name of Silicon Graphics, Inc. not be used in
 * advertising or publicity pertaining to distribution of the software
 * without specific, written prior permission.
 *
 * THE MATERIAL EMBODIED ON THIS SOFTWARE IS PROVIDED TO YOU "AS-IS" AND
 * WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE,
 * INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR
 * FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SILICON
 * GRAPHICS, INC. BE LIABLE TO YOU OR ANYONE ELSE FOR ANY DIRECT,
 * SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND,
 * OR ANY DAMAGES WHATSOEVER, INCLUDING WITHOUT LIMITATION, LOSS OF
 * PROFIT, LOSS OF USE, SAVINGS OR REVENUE, OR THE CLAIMS OF THIRD
 * PARTIES, WHETHER OR NOT SILICON GRAPHICS, INC. HAS BEEN ADVISED OF
 * THE POSSIBILITY OF SUCH LOSS, HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE POSSESSION, USE
 * OR PERFORMANCE OF THIS SOFTWARE.
 *
 * US Government Users Restricted Rights
 * Use, duplication, or disclosure by the Government is subject to
 * restrictions set forth in FAR 52.227.19(c)(2) or subparagraph
 * (c)(1)(ii) of the Rights in Technical Data and Computer Software
 * clause at DFARS 252.227-7013 and/or in similar or successor clauses
 * in the FAR or the DOD or NASA FAR Supplement. Unpublished - rights
 * reserved under the copyright laws of the United States.
 *
 * Contractor/manufacturer is:
 *   Silicon Graphics, Inc.
 *   1500 Crittenden Lane
 *   Mountain View, CA 94043
 *   United State of America
 *
 * OpenGL(R) is a registered trademark of Silicon Graphics, Inc.
 */

/*
 * teapots.c
 * This program demonstrates lots of material properties.
 * A single light source illuminates the objects.
 */
#include <stdlib.h>
#include <GL/glut.h>

GLuint teapotList;

/*
 * Initialize depth buffer, projection matrix, light source, and lighting
 * model. Do not specify a material property here.
 */
void init(void)
{
    GLfloat ambient[] = {0.0, 0.0, 0.0, 1.0};
    GLfloat diffuse[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat position[] = {0.0, 3.0, 3.0, 0.0};

    GLfloat lmodel_ambient[] = {0.2, 0.2, 0.2, 1.0};
    GLfloat local_view[] = {0.0};

    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
    glLightModelfv(GL_LIGHT_MODEL_LOCAL_VIEWER, local_view);

    glFrontFace(GL_CW);

```

```

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_AUTO_NORMAL);
glEnable(GL_NORMALIZE);
glEnable(GL_DEPTH_TEST);
/* be efficient--make teapot display list */
teapotList = glGenLists(1);
glNewList (teapotList, GL_COMPILE);
glutSolidTeapot(1.0);
glEndList ();
}

/*
 * Move object into position. Use 3rd through 12th
 * parameters to specify the material property. Draw a teapot.
 */
void renderTeapot(GLfloat x, GLfloat y,
  GLfloat ambr, GLfloat ambg, GLfloat ambb,
  GLfloat difr, GLfloat difg, GLfloat difb,
  GLfloat specr, GLfloat specg, GLfloat specb, GLfloat shine)
{
  GLfloat mat[4];

  glPushMatrix();
  glTranslatef(x, y, 0.0);
  mat[0] = ambr; mat[1] = ambg; mat[2] = ambb; mat[3] = 1.0;
  glMaterialfv(GL_FRONT, GL_AMBIENT, mat);
  mat[0] = difr; mat[1] = difg; mat[2] = difb;
  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat);
  mat[0] = specr; mat[1] = specg; mat[2] = specb;
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat);
  glMaterialf(GL_FRONT, GL_SHININESS, shine * 128.0);
  glCallList(teapotList);
  glPopMatrix();
}

/**
 * First column: emerald, jade, obsidian, pearl, ruby, turquoise
 * 2nd column: brass, bronze, chrome, copper, gold, silver
 * 3rd column: black, cyan, green, red, white, yellow plastic
 * 4th column: black, cyan, green, red, white, yellow rubber
 */
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  renderTeapot(2.0, 17.0, 0.0215, 0.1745, 0.0215,
    0.07568, 0.61424, 0.07568, 0.633, 0.727811, 0.633, 0.6);
  renderTeapot(2.0, 14.0, 0.135, 0.2225, 0.1575,
    0.54, 0.89, 0.63, 0.316228, 0.316228, 0.316228, 0.1);
  renderTeapot(2.0, 11.0, 0.05375, 0.05, 0.06625,
    0.18275, 0.17, 0.22525, 0.332741, 0.328634, 0.346435, 0.3);
  renderTeapot(2.0, 8.0, 0.25, 0.20725, 0.20725,
    1, 0.829, 0.829, 0.296648, 0.296648, 0.296648, 0.088);
  renderTeapot(2.0, 5.0, 0.1745, 0.01175, 0.01175,
    0.61424, 0.04136, 0.04136, 0.727811, 0.626959, 0.626959, 0.6);
  renderTeapot(2.0, 2.0, 0.1, 0.18725, 0.1745,
    0.396, 0.74151, 0.69102, 0.297254, 0.30829, 0.306678, 0.1);
  renderTeapot(6.0, 17.0, 0.329412, 0.223529, 0.027451,
    0.780392, 0.568627, 0.113725, 0.992157, 0.941176, 0.807843,
    0.21794872);
  renderTeapot(6.0, 14.0, 0.2125, 0.1275, 0.054,
    0.714, 0.4284, 0.18144, 0.393548, 0.271906, 0.166721, 0.2);
  renderTeapot(6.0, 11.0, 0.25, 0.25, 0.25,
    0.4, 0.4, 0.4, 0.774597, 0.774597, 0.774597, 0.6);
  renderTeapot(6.0, 8.0, 0.19125, 0.0735, 0.0225,
    0.7038, 0.27048, 0.0828, 0.256777, 0.137622, 0.086014, 0.1);
  renderTeapot(6.0, 5.0, 0.24725, 0.1995, 0.0745,
    0.75164, 0.60648, 0.22648, 0.628281, 0.555802, 0.366065, 0.4);
  renderTeapot(6.0, 2.0, 0.19225, 0.19225, 0.19225,
    0.50754, 0.50754, 0.50754, 0.508273, 0.508273, 0.508273, 0.4);
  renderTeapot(10.0, 17.0, 0.0, 0.0, 0.0, 0.01, 0.01, 0.01,
    0.50, 0.50, 0.50, .25);
}

```

```

renderTeapot(10.0, 14.0, 0.0, 0.1, 0.06, 0.0, 0.50980392, 0.50980392,
0.50196078, 0.50196078, 0.50196078, .25);
renderTeapot(10.0, 11.0, 0.0, 0.0, 0.0,
0.1, 0.35, 0.1, 0.45, 0.55, 0.45, .25);
renderTeapot(10.0, 8.0, 0.0, 0.0, 0.0, 0.5, 0.0, 0.0,
0.7, 0.6, 0.6, .25);
renderTeapot(10.0, 5.0, 0.0, 0.0, 0.0, 0.55, 0.55, 0.55,
0.70, 0.70, 0.70, .25);
renderTeapot(10.0, 2.0, 0.0, 0.0, 0.0, 0.5, 0.5, 0.0,
0.60, 0.60, 0.50, .25);
renderTeapot(14.0, 17.0, 0.02, 0.02, 0.02, 0.01, 0.01, 0.01,
0.4, 0.4, 0.4, .078125);
renderTeapot(14.0, 14.0, 0.0, 0.05, 0.05, 0.4, 0.5, 0.5,
0.04, 0.7, 0.7, .078125);
renderTeapot(14.0, 11.0, 0.0, 0.05, 0.0, 0.4, 0.5, 0.4,
0.04, 0.7, 0.04, .078125);
renderTeapot(14.0, 8.0, 0.05, 0.0, 0.0, 0.5, 0.4, 0.4,
0.7, 0.04, 0.04, .078125);
renderTeapot(14.0, 5.0, 0.05, 0.05, 0.05, 0.5, 0.5, 0.5,
0.7, 0.7, 0.7, .078125);
renderTeapot(14.0, 2.0, 0.05, 0.05, 0.0, 0.5, 0.5, 0.4,
0.7, 0.7, 0.04, .078125);
glFlush();
}

void reshape(int w, int h)
{
glViewport(0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (w <= h)
glOrtho(0.0, 16.0, 0.0, 16.0*(GLfloat)h/(GLfloat)w,
-10.0, 10.0);
else
glOrtho(0.0, 16.0*(GLfloat)w/(GLfloat)h, 0.0, 16.0,
-10.0, 10.0);
glMatrixMode(GL_MODELVIEW);
}

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

/*
* Main Loop
*/
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(500, 600);
glutInitWindowPosition(50,50);
glutCreateWindow(argv[0]);
init();
glutReshapeFunc(reshape);
glutDisplayFunc(display);
glutKeyboardFunc (keyboard);
glutMainLoop();
return 0;
}

```