#include <stdlib.h>

#include <GL/glut.h>

GLfloat X = 0.5; /\* A scaling factor \*/

static GLfloat theta[] = {0.0,0.0,0.0};

static GLint axis = 2;

GLfloat GlobalVertices[][3] = {{-1.0,-1.0,1.0},

{-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},

{-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},

{1.0,-1.0,-1.0}};

// These will be the coordinates of the vertices of the cube

GLfloat CubeVertices[][3] = {{-1.0,-1.0,1.0},

{-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},

{-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},

{1.0,-1.0,-1.0}};

// shading ...

typedef struct materialStruct {

GLfloat ambient[4];

GLfloat diffuse[4];

GLfloat specular[4];

GLfloat shininess;

} materialStruct;

materialStruct brassMaterials = {

{0.33, 0.22, 0.03, 1.0},

{0.78, 0.57, 0.11, 1.0},

{0.99, 0.91, 0.81, 1.0},

27.8

};

materialStruct redPlasticMaterials = {

{0.3, 0.0, 0.0, 1.0},

{0.6, 0.0, 0.0, 1.0},

{0.8, 0.6, 0.6, 1.0},

32.0

};

materialStruct whiteShinyMaterials = {

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

100.0

};

typedef struct lightingStruct {

GLfloat ambient[4];

GLfloat diffuse[4];

GLfloat specular[4];

} lightingStruct;

lightingStruct whiteLighting = {

{0.0, 0.0, 0.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0}

};

lightingStruct coloredLighting = {

{0.2, 0.0, 0.0, 1.0},

{0.0, 1.0, 0.0, 1.0},

{0.0, 0.0, 1.0, 1.0}

};

materialStruct \*currentMaterials;

lightingStruct \*currentLighting;

GLfloat normals[][3] = {{0.0, 0.0, 1.0}, {1.0, 0.0, 0.0},

{0.0, -1.0, 0.0}, {0.0, 1.0, 0.0}, {0.0, 0.0, -1.0},

{-1.0, 0.0, 0.0}};

void a3dpolygon(GLfloat vertices[][3], int a, int b, int c, int d) {

/\* draw a polygon via list of vertices \*/

glBegin(GL\_POLYGON);

glVertex3fv(vertices[a]);

glVertex3fv(vertices[b]);

glVertex3fv(vertices[c]);

glVertex3fv(vertices[d]);

glEnd();

}

void colorcube()

{

/\* map vertices to facets \*/

glNormal3fv(normals[0]);

a3dpolygon(CubeVertices, 0,3,2,1);

glNormal3fv(normals[1]);

a3dpolygon(CubeVertices, 2,3,7,6);

glNormal3fv(normals[2]);

a3dpolygon(CubeVertices, 3,0,4,7);

glNormal3fv(normals[3]);

a3dpolygon(CubeVertices, 1,2,6,5);

glNormal3fv(normals[4]);

a3dpolygon(CubeVertices, 4,5,6,7);

glNormal3fv(normals[5]);

a3dpolygon(CubeVertices, 5,4,0,1);

}

void display()

{

/\* display callback, clear frame buffer and z buffer,

rotate cube and draw, swap buffers \*/

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(theta[0], 1.0, 0.0, 0.0);

glRotatef(theta[1], 0.0, 1.0, 0.0);

glRotatef(theta[2], 0.0, 0.0, 1.0);

colorcube();

GLfloat light0\_pos[4] = {0.0, 0.0, -3.0, 0.0};

glLightfv(GL\_LIGHT0, GL\_AMBIENT, currentLighting -> ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, currentLighting -> diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, currentLighting -> specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_pos);

glutSwapBuffers();

}

void spinCube()

{

/\* idle callback, spin cube about selected axis \*/

theta[axis] += 0.9;

if( theta[axis] > 360.0 ) theta[axis] -= 360.0;

glutPostRedisplay();

}

void specialkey(int key, int x, int y) {

switch (key) {

case GLUT\_KEY\_LEFT:

axis = 0;

break;

case GLUT\_KEY\_UP:

axis = 1;

break;

case GLUT\_KEY\_RIGHT:

axis = 2;

break;

}

}

void key(unsigned char k, int x, int y) {

switch(k) {

case '1':

glutIdleFunc(NULL);

break;

case '2':

glutIdleFunc(spinCube);

break;

case '3':

currentMaterials = &redPlasticMaterials;

break;

case '4':

currentMaterials = &whiteShinyMaterials;

break;

case '5':

currentMaterials = &brassMaterials;

break;

case '6':

currentLighting = &whiteLighting;

break;

case '7':

currentLighting = &coloredLighting;

break;

case 'q':

exit(0);

break;

}

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, currentLighting -> ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, currentLighting -> diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, currentLighting -> specular);

glutPostRedisplay();

}

void init()

{

int i, j;

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_NORMALIZE);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

currentMaterials = &redPlasticMaterials;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

currentLighting = &whiteLighting;

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

for (j = 0; j < 3; j++) {

for (i = 0; i < 8; i++) {

CubeVertices[i][j] = GlobalVertices[i][j] \* X; }

}

}

int main(int argc, char \*\*argv)

{

glutInit(&argc, argv);

/\* need both double buffering and z buffer \*/

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutCreateWindow("Colourcube");

glutDisplayFunc(display);

glutIdleFunc(spinCube);

glutSpecialFunc(specialkey);

glutKeyboardFunc(key);

glEnable(GL\_DEPTH\_TEST); /\* Enable hidden-surface removal \*/

init();

glutMainLoop();

}

#include <stdlib.h>

#include <GL/glut.h>

GLfloat X = 0.5; /\* A scaling factor \*/

static GLfloat theta[] = {45.0,45.0,45.0};

static GLint axis = 2;

GLfloat GlobalVertices[][3] = {{-1.0,-1.0,1.0},

{-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},

{-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},

{1.0,-1.0,-1.0}};

// These will be the coordinates of the vertices of the cube

GLfloat CubeVertices[][3] = {{-1.0,-1.0,1.0},

{-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},

{-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},

{1.0,-1.0,-1.0}};

// shading ...

typedef struct materialStruct {

GLfloat ambient[4];

GLfloat diffuse[4];

GLfloat specular[4];

GLfloat shininess;

} materialStruct;

materialStruct brassMaterials = {

{0.33, 0.22, 0.03, 1.0},

{0.78, 0.57, 0.11, 1.0},

{0.99, 0.91, 0.81, 1.0},

27.8

};

materialStruct redPlasticMaterials = {

{0.3, 0.0, 0.0, 1.0},

{0.6, 0.0, 0.0, 1.0},

{0.8, 0.6, 0.6, 1.0},

32.0

};

materialStruct whiteShinyMaterials = {

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

100.0

};

typedef struct lightingStruct {

GLfloat ambient[4];

GLfloat diffuse[4];

GLfloat specular[4];

} lightingStruct;

lightingStruct whiteLighting = {

{0.0, 0.0, 0.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0}

};

lightingStruct coloredLighting = {

{0.2, 0.0, 0.0, 1.0},

{0.0, 1.0, 0.0, 1.0},

{0.0, 0.0, 1.0, 1.0}

};

materialStruct \*currentMaterials;

lightingStruct \*currentLighting;

GLfloat normals[][3] = {{0.0, 0.0, 1.0}, {1.0, 0.0, 0.0},

{0.0, -1.0, 0.0}, {0.0, 1.0, 0.0}, {0.0, 0.0, -1.0},

{-1.0, 0.0, 0.0}};

void a3dpolygon(GLfloat vertices[][3], int a, int b, int c, int d) {

/\* draw a polygon via list of vertices \*/

glBegin(GL\_POLYGON);

glVertex3fv(vertices[a]);

glVertex3fv(vertices[b]);

glVertex3fv(vertices[c]);

glVertex3fv(vertices[d]);

glEnd();

}

void lightpoint(){

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(theta[axis], 1.0, 0.0, 0.0);

glRotatef(theta[axis], 0.0, 1.0, 0.0);

glRotatef(theta[axis], 0.0, 0.0, 1.0);

GLfloat light0\_pos[4] = {0.0, 0.0, -3.0, 0.0};

glLightfv(GL\_LIGHT0, GL\_AMBIENT, currentLighting -> ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, currentLighting -> diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, currentLighting -> specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_pos);

}

void colorcube()

{

/\* map vertices to facets \*/

glNormal3fv(normals[0]);

a3dpolygon(CubeVertices, 0,3,2,1);

glNormal3fv(normals[1]);

a3dpolygon(CubeVertices, 2,3,7,6);

glNormal3fv(normals[2]);

a3dpolygon(CubeVertices, 3,0,4,7);

glNormal3fv(normals[3]);

a3dpolygon(CubeVertices, 1,2,6,5);

glNormal3fv(normals[4]);

a3dpolygon(CubeVertices, 4,5,6,7);

glNormal3fv(normals[5]);

a3dpolygon(CubeVertices, 5,4,0,1);

}

void display()

{

/\* display callback, clear frame buffer and z buffer,

rotate cube and draw, swap buffers \*/

lightpoint();

glutSwapBuffers();

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(45, 1.0, 0.0, 0.0);

glRotatef(45, 0.0, 1.0, 0.0);

glRotatef(45, 0.0, 0.0, 1.0);

colorcube();

}

void spinCube()

{

/\* idle callback, spin cube about selected axis \*/

theta[axis] += 0.9;

if( theta[axis] > 360.0 ) theta[axis] -= 360.0;

glutPostRedisplay();

}

void specialkey(int key, int x, int y) {

switch (key) {

case GLUT\_KEY\_LEFT:

axis = 0;

break;

case GLUT\_KEY\_UP:

axis = 1;

break;

case GLUT\_KEY\_RIGHT:

axis = 2;

break;

}

}

void key(unsigned char k, int x, int y) {

switch(k) {

case '1':

glutIdleFunc(NULL);

break;

case '2':

glutIdleFunc(spinCube);

break;

case '3':

currentMaterials = &redPlasticMaterials;

break;

case '4':

currentMaterials = &whiteShinyMaterials;

break;

case '5':

currentMaterials = &brassMaterials;

break;

case '6':

currentLighting = &whiteLighting;

break;

case '7':

currentLighting = &coloredLighting;

break;

case 'q':

exit(0);

break;

}

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, currentLighting -> ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, currentLighting -> diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, currentLighting -> specular);

glutPostRedisplay();

}

void init()

{

int i, j;

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_NORMALIZE);

currentMaterials = &redPlasticMaterials;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

currentLighting = &whiteLighting;

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

for (j = 0; j < 3; j++) {

for (i = 0; i < 8; i++) {

CubeVertices[i][j] = GlobalVertices[i][j] \* X; }

}

}

int main(int argc, char \*\*argv)

{

glutInit(&argc, argv);

/\* need both double buffering and z buffer \*/

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutCreateWindow("Colourcube");

glutDisplayFunc(display);

glutIdleFunc(spinCube);

glutSpecialFunc(specialkey);

glutKeyboardFunc(key);

glEnable(GL\_DEPTH\_TEST); /\* Enable hidden-surface removal \*/

init();

glutMainLoop();

}