

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

struct Vector

{

float fx, fy, fz;

};

struct Color

{

float Ir, Ig, Ib;

};

float KaIa;//环境光强度

float Kd, n;

Vector H, light;

Color mLight, mColor;

GLboolean bLight = false;

//利用圆的八向对称性生成圆上的点

void CirclePt(int x0, int y0, int x, int y, Color mColor)

{

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib);

glBegin(GL\_POINTS);

glVertex2i(x + x0, y + y0);//x,y

glVertex2i(-x + x0, y + y0);//-x,y

glVertex2i(x + x0, -y + y0);//x, -y

glVertex2i(-x + x0, -y + y0);//-x, -y

glVertex2i(y + x0, x + y0);//y, x

glVertex2i(y + x0, -x + y0);//y, -x

glVertex2i(-y + x0, x + y0);//-y, x

glVertex2i(-y + x0, -x + y0);//-y, -x

glEnd();

}

//中点圆生成算法

void MidCircle(int x0, int y0, int r, Color mColor)

{

int x, y, deltax, deltay, d;

x = 0;

y = r;

deltax = 3;

deltay = 5 - r - r;

d = 1 - r;

CirclePt(x0, y0, x, y, mColor);

while (x<y)

{

if (d<0)

{

d += deltax;

deltax += 2;

deltay += 2;

x++;

}

else

{

d += deltay;

deltax += 2;

deltay += 4;

x++;

y--;

}

CirclePt(x0, y0, x, y, mColor);

}

}

//根据中点圆算法填充圆域

void FlatCircle(int x0, int y0, int r, Color mColor)

{

int x, y, deltax, deltay, d;

x = 0;

y = r;

deltax = 3;

deltay = 5 - r - r;

d = 1 - r;

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib);

glBegin(GL\_POINTS);

{

for (int i = -x; i <= x; i++)

{

glVertex2i(i + x0, y + y0);

glVertex2i(i + x0, -y + y0);

}

for (int i = -y; i <= y; i++)

{

glVertex2i(i + x0, x + y0);

glVertex2i(i + x0, -x + y0);

}

while (x<y)

{

if (d<0)

{

d += deltax;

deltax += 2;

deltay += 2;

x++;

}

else

{

d += deltay;

deltax += 2;

deltay += 4;

x++;

y--;

}

for (int i = -x; i <= x; i++)

{

glVertex2i(i + x0, y + y0);

glVertex2i(i + x0, -y + y0);

}

for (int i = -y; i <= y; i++)

{

glVertex2i(i + x0, x + y0);

glVertex2i(i + x0, -x + y0);

}

}

}

glEnd();

}

//初始化设定

void Init()

{

float mo;

Vector eye;

mLight.Ir = 0;

mLight.Ig = 175;

mLight.Ib = 0;

KaIa = 30;

Kd = 1;

n = 10;

light.fx = 0.50; light.fy = 0.50;

light.fz = sqrt(1 - (light.fx\*light.fx) - (light.fy\*light.fy));

eye.fx = 0; eye.fy = 0; eye.fz = 1;

H.fx = light.fx + eye.fx;

H.fy = light.fy + eye.fy;

H.fz = light.fz + eye.fz;

mo = sqrt(H.fx\*H.fx + H.fy\*H.fy + H.fz\*H.fz);

H.fx = (H.fx / mo); H.fy = (H.fy / mo); H.fz = (H.fz / mo);

H.fx = (H.fx / mo); H.fy = (H.fy / mo); H.fz = (H.fz / mo);

glClearColor(0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_SMOOTH);

}

//根据Phong模型计算光强

Color Phong(int x0, int y0, int r, int x, int y)

{

Vector N;

float z, alpha, theta, Ks;

Ks = 1.0 - Kd;

z = sqrt((float)(r\*r - (x - x0)\*(x - x0) - (y - y0)\*(y - y0)));

N.fx = (x - x0)\*1.0 / r;

N.fy = (y - y0)\*1.0 / r;

N.fz = z\*1.0 / r;

theta = N.fx \* light.fx + N.fy \* light.fy + N.fz \* light.fz;

if (theta<0)

theta = 0;

alpha = H.fx\*N.fx + H.fy\*N.fy + H.fx\*N.fz;

if (alpha<0)

alpha = 0;

mColor.Ir = KaIa + mLight.Ir\*Kd\*theta + mLight.Ir\*Ks\*pow(alpha, n);

mColor.Ig = KaIa + mLight.Ig\*Kd\*theta + mLight.Ig\*Ks\*pow(alpha, n);

mColor.Ib = KaIa + mLight.Ib\*Kd\*theta + mLight.Ib\*Ks\*pow(alpha, n);

return mColor;

}

//根据计算的光强按球体的结果着色

void Sphere(int x0, int y0, int r)

{

int x, y, deltax, deltay, d;

x = 0;

y = r;

deltax = 3;

deltay = 5 - r - r;

d = 1 - r;

glBegin(GL\_POINTS);

{

for (int i = -x; i <= x; i++)

{

mColor = Phong(x0, y0, r, i + x0, y + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, y + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, -y + y0);

}

for (int i = -y; i <= y; i++)

{

mColor = Phong(x0, y0, r, i + x0, x + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, x + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, -x + y0);

}

while (x<y)

{

if (d<0)

{

d += deltax;

deltax += 2;

deltay += 2;

x++;

}

else

{

d += deltay;

deltax += 2;

deltay += 4;

x++;

y--;

}

for (int i = -x; i <= x; i++)

{

mColor = Phong(x0, y0, r, i + x0, y + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, y + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, -y + y0);

}

for (int i = -y; i <= y; i++)

{

mColor = Phong(x0, y0, r, i + x0, x + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, x + y0);

glColor3ub(mColor.Ir, mColor.Ig, mColor.Ib); glVertex2i(i + x0, -x + y0);

}

}

}

glEnd();

}

void myDisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0f, 1.0f, 1.0f);

Color clr;

clr.Ir = 180, clr.Ig = 180, clr.Ib = 180;

FlatCircle(280, 240, 70, clr);

Sphere(540, 240, 70);

glFlush();

}

void Reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowPosition(100, 100);

glutInitWindowSize(640, 480);

glutCreateWindow("Hello World!");

Init();

glutDisplayFunc(myDisplay);

glutReshapeFunc(Reshape);

glutMainLoop();

return 0;

}