#include <iostream>

#include <windows.h>

#include <GL/glut.h>

#include <math.h>

using namespace std;

//my raytracing only traces five rays and interlotion the color in the sphere

//the ray is <0,0.14,-0.86> <0.> <0,0.14,-0.86> <0,-0.285,-0.91> <0.285,0,-0.91><-0.285,0,-0.91>

//we set the pixels in the screen as a 60\*60 matrix and emit rays ,and each element represents a 0.01\*0.01 pixel

//eye position is at 0,0,1.5 and screen position is at 0,0,1

float \*\*matrix;

float \* intersect;

float \*specular;

float directionx,directiony,directionz;

void getspecular(float \*intersect,float directionx, float directiony,float directionz,float \*specular)

{

float \*normal = new float[3];

float length = sqrt(intersect[0]\*intersect[0]+intersect[1]\*intersect[1]+intersect[2]\*intersect[2]);

normal[0]=intersect[0]/length;

normal[1]=intersect[1]/length;

normal[2]=intersect[2]/length;

length = normal[0]\*directionx+normal[1]\*directiony+normal[2]\*directionz;

length=-length;

length=abs(length);

specular[0]=2\*length\*normal[0]+directionx;

specular[1]=2\*length\*normal[1]+directiony;

specular[2]=2\*length\*normal[2]+directionz;

}

//this raytracing function find the interset of the original ray

bool rayfindinterset1(float directionx, float directiony,float directionz, float \*interset)

{

float x =directionx, y=directiony, z = directionz;

double t=0;

//delta <0 and there is no solution

if ((9\*z\*z-391\*(x\*x+y\*y)) < 0)

return false ;

else

{

t= (-4\*z-sqrt((9\*z\*z-391\*(x\*x+y\*y))/100))/(x\*x+y\*y+z\*z);

intersect[0]=t\*x;

intersect[1]=t\*y;

intersect[2]=t\*z+2;

}

return true;

}

//judge if this point can be reached by the light source

//this function is not true but i think it is thrick

bool ifshawdow(float \*intersect)

{

float x= intersect[0],y=intersect[1],z= intersect[2];

float directionx = -x,directiony = 5-y,directionz =-5-z,weight=0.01;

if( pow((x+weight\*directionx),2)+pow((y+weight\*directiony),2)+pow((z+weight\*directionz),2)<0.09)

return true;

if ( pow((x+weight\*directionx+0.7),2)+pow((y+weight\*directiony),2)+pow((z+weight\*directionz),2)<0.09)

return true;

return false;

}

float phongspecular(float \*intersect)

{

float light\_x =-intersect[0],light\_y=5-intersect[1],light\_z=-5-intersect[2];

double length = sqrt(light\_x\*light\_x+light\_y\*light\_y+light\_z\*light\_z);

float color = light\_x\*specular[0]+light\_y\*specular[1]+light\_z\*specular[2];

color/=length;

length = sqrt(specular[0]\*specular[0]+specular[1]\*specular[1]+specular[2]\*specular[2]);

color/=length;

return color;

}

//now there is no need to do this function because no ray in this model need to be traced@.@

//prehaps is my fallt

float phongdiffusor(float \*intersect)

{

return 1;

}

void matrixchange(float \*\*matrix)

{

float maxnum=0,minnum=1;

for(int i =0;i<60;i++)

{

for(int j=0;j<60;j++)

{

if(matrix[i][j]!=0)

{

if (maxnum<matrix[i][j])

maxnum=matrix[i][j];

if (minnum>matrix[i][j])

minnum=matrix[i][j];

}

}

}

for(int i =0;i<60;i++)

for(int j=0;j<60;j++)

matrix[i][j]=(matrix[i][j]-minnum)/(maxnum-minnum);

}

/\*

void display()

{

float screenx=0,screeny=0,screenz=1,screenstartx=-0.3,screenstarty=0.3;

glClear(GL\_COLOR\_BUFFER\_BIT);

for (int i =0;i<60;i++)

{

for(int j =0;j<60;j++)

{

if(matrix[i][j]>0)

{

screenx=screenstartx+i\*0.01;

screeny=screenstarty-j\*0.01;

glColor3f(0.0f,0.0f,matrix[i][j]+0.02);

glRectf(screenx,screeny,screenx+0.01,screeny-0.01);

}

}

}

//glRectf(-0.5f,-0.5f,0.5f,0.5f);

glFlush();

}

\*/

void display()

{

//glClearColor(1.0f, 1.0f, 1.0f,0.1f);

//matrix = new float[60][60];

//interset = new float[3];

float width=0.05,height=0.05,startx=0.3,starty=0.3;

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

glEnable(GL\_COLOR\_MATERIAL);

glLightModeli(GL\_FRONT,GL\_AMBIENT\_AND\_DIFFUSE);

glColor3f(1.0f,1.0f,0.0f);

//glTranslatef(0.4,0,0);

//glutSolidSphere(0.3,30,30);

glRotatef(0.1,0.0,0.0,-1.0f);

glTranslatef(-0.7,-0,-0);

glutSolidSphere(0.2,30,30);

//when we need to set the color we should first let some function

glTranslated(0.7,0,0);

//glDisable(GL\_LIGHT0);

float screenx=0,screeny=0,screenz=1,screenstartx=-0.3,screenstarty=0.3;

//glClear(GL\_COLOR\_BUFFER\_BIT);

for (int i =0;i<60;i++)

{

for(int j =0;j<60;j++)

{

if(matrix[i][j]>0)

{

screenx=screenstartx+i\*0.01;

screeny=screenstarty-j\*0.01;

glColor3f(0.0f,0.0f,matrix[i][j]+0.02);

glRectf(screenx,screeny,screenx+0.01,screeny-0.01);

}

}

}

glFlush();

}

bool rayfindinterset2(float directionx,float directiony,float directionz ,float \* intersect)

{

float dx =directionx,dy=directiony,dz= directionz,

x=intersect[0],y= intersect[1],z= intersect[2],t=0;

double delta = pow((dx\*(x+0.7)+dy\*y+dz\*z),2)-(dx\*dx+dz\*dz+dy\*dy)\*((x+0.7)\*(x+0.7)+y\*y+z\*z-0.04);

if (delta<0)

return false;

else

t=(-2\*(dx\*(x+0.7)+dy\*y+dz\*z)-delta)/2/(dx\*dx+dz\*dz+dy\*dy);

intersect[0]=t\*x+dx;

intersect[1]=t\*y+dy;

intersect[2]=t\*z+dz;

return true;

}

void raytracing\_display()

{

float screenx=0,screeny=0,screenz=1,screenstartx=-0.3,screenstarty=0.3;

for(int i =0;i<60;i++)

{

for(int j =0; j<60;j++)

{

screenx=screenstartx+i\*0.01;

screeny=screenstarty-j\*0.01;

//

if(rayfindinterset1(screenx,screeny,screenz-2,intersect))

{

//calculate the phong illumination model

if(!ifshawdow(intersect))

{

getspecular(intersect,screenx,screeny,screenz-2,specular);

matrix[i][j]=phongspecular(intersect);

directionx= specular[0];

directiony= specular[1];

directionz= specular[2];

if(rayfindinterset2(directionx,directiony,directionz,intersect))//if the second ray has the intersect

{

cout<<" aaa";

if(!ifshawdow(intersect)) //if the intersect can be reached by the light

matrix[i][j]+=phongdiffusor(intersect);

}

}

//matrix[i][j]=1;

else

matrix[i][j]=0;

}

//the position is that this ray intersects with nothing

else

{

matrix[i][j]=0;

}

}

}

matrixchange(matrix);

}

void GL\_myInitial()

{

//set the coefficient for the illumination model

GLfloat light\_position[] = {0.0,5.0,-5.0,1.0};

GLfloat light\_ambient [] = { 0.0, 0.0, 0.0, 1.0 }; //the forth dimension is used in the matrix change

GLfloat light\_diffuse [] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glLightfv(GL\_LIGHT0, GL\_AMBIENT , light\_ambient );

glLightfv(GL\_LIGHT0, GL\_DIFFUSE , light\_diffuse );

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glDepthFunc(GL\_LESS);

glEnable(GL\_DEPTH\_TEST);

}

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

{

int winwidth=1080,winheight=1080;

glutInit(&argc,argv);

glutInitWindowSize (winwidth,winheight);

glutCreateWindow("opengl and my function!");

//gluLookAt(1.0,0,0,0,0,0,0,0,1);

//this is to initial the opengl raytracing and i choose to close it

GL\_myInitial();

matrix = new float\*[60];

for(int i =0;i<60;i++)

matrix[i]=new float[60];

intersect= new float[3];

specular=new float[3];

raytracing\_display();

glutDisplayFunc(display);

for(int i =0;i<60;i++)

{

for (int j =0;j<60;j++)

{

if(matrix[i][j]>0)

//cout<<matrix[i][j]<<" ";

cout<<"1 ";

else

cout<<"0 ";

}

cout<<"\n";

}

glutMainLoop();

}