МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Кафедра компьютерной технологий и программной инженерии

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асс. |  |  |  | И.М. Лозоватский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6 |
| «Вывод трехмерных объектов с динамическим расчетом проекционных теней» |
| по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4831 |  |  |  | К.А.Корюнщенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

**Цель работы**

Целью работы является получение практических навыков динамического расчета теней для трехмерных объектов.

**Задание на лабораторную работу**

Вывести трехмерную сцену с движущимся объектом, который отбрасывает тень на другой объект (напр. плоскость). Тень должна перемещаться вместе с движением объекта исходя из взаимного положения источника света, объекта, который отбрасывает тень и объектов, на которые тень проецируется.

**Ход работы**

**Листинг программы:**

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <cmath>

#include <GLUT/glut.h>

#include <cmath>

#include <stdlib.h>

#include <stdio.h>

#include <iostream>

#include <math.h>

#define GL\_PI 3.1415f

GLfloat al = 0.5f;

GLfloat xRot = 0.45f;

GLfloat yRot = 0.35f;

GLboolean bOutline = (GLboolean)true;

GLfloat specref[] = { 1.0f, 1.0f, 1.0f, 1.0f };

float mat1\_dif[] = { 1.0f, 0.3f, 0.8f };

float mat1\_amb[] = { 0.3f, 0.9f, 0.2f };

float mat1\_spec[] = { 1.0f, 0.5f, 0.8f };

float mat1\_shininess = 0.7f \* 128;

GLfloat light\_ambient[] = { 0.5, 0.5, 0.5, 0.2 };

GLfloat light\_diffuse[] = { 0.3, 0.4, 0.9, 1.0 };

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

GLenum shademode = GL\_FLAT;

GLfloat colv[4];

GLfloat ex = 100.0f;

GLfloat ey = 259.0f;

GLfloat ez = 100.0f;

GLfloat delta = 0.1f;

GLfloat deltaR = 0.01f;

GLfloat lightPos[] = { 100.0f, 80.0f, 85.0f, 1.0f };

GLfloat floor\_1[] = { 0.0f, 1.0f, 0.0f, 0.0f };

GLfloat shadowMat[16];

GLfloat x1 = 100.0;

GLfloat y\_1 = 25.0;

GLfloat z1 = 130.0;

GLfloat x2 = 10.0;

GLfloat y2 = 10.0;

GLfloat z2 = 120.0;

GLfloat x3 = 110.0;

GLfloat y3 = 10.0;

GLfloat z3 = 115.0;

GLfloat cx = (x1 + x2 + x3) / 3;

GLfloat cy = (y\_1 + y2 + y3) / 3;

GLfloat cz = (z1 + z2 + z3) / 3;

GLfloat vercube[8][3] = { { 80, 10, 0 },{ 100, 10, 0 },{ 100, 10, 20 },{ 80, 10, 20 },{ 80, 30, 0 },{ 100, 30, 0 },{ 100, 30, 20 },{ 80, 30, 20 } };

GLfloat centr\_cube[3];

GLfloat vertringle[3][3] = { { 100, 25, 130 },{ 120, 10, 120 },{ 110, 10, 115 } };

void gltMakeShadowMatrix(GLfloat vPlaneEquation[], GLfloat vLightPos[], GLfloat destMat[]) {

GLfloat dot;

dot = vPlaneEquation[0] \* vLightPos[0] + vPlaneEquation[1] \* vLightPos[1] +

vPlaneEquation[2] \* vLightPos[2] + vPlaneEquation[3] \* vLightPos[3];

destMat[0] = dot - vLightPos[0] \* vPlaneEquation[0];

destMat[4] = 0.0f - vLightPos[0] \* vPlaneEquation[1];

destMat[8] = 0.0f - vLightPos[0] \* vPlaneEquation[2];

destMat[12] = 0.0f - vLightPos[0] \* vPlaneEquation[3];

destMat[1] = 0.0f - vLightPos[1] \* vPlaneEquation[0];

destMat[5] = dot - vLightPos[1] \* vPlaneEquation[1];

destMat[9] = 0.0f - vLightPos[1] \* vPlaneEquation[2];

destMat[13] = 0.0f - vLightPos[1] \* vPlaneEquation[3];

destMat[2] = 0.0f - vLightPos[2] \* vPlaneEquation[0];

destMat[6] = 0.0f - vLightPos[2] \* vPlaneEquation[1];

destMat[10] = dot - vLightPos[2] \* vPlaneEquation[2];

destMat[14] = 0.0f - vLightPos[2] \* vPlaneEquation[3];

destMat[3] = 0.0f - vLightPos[3] \* vPlaneEquation[0];

destMat[7] = 0.0f - vLightPos[3] \* vPlaneEquation[1];

destMat[11] = 0.0f - vLightPos[3] \* vPlaneEquation[2];

destMat[15] = dot - vLightPos[3] \* vPlaneEquation[3];

}

void DrawLight(){

glPushMatrix();

glTranslatef(lightPos[0], lightPos[1], lightPos[2]);

glColor3ub(255, 255, 0);

glutSolidSphere(5.0f, 10, 10);

glPopMatrix();

}

void treugol\_i\_cube\_bez\_c(){

glBegin(GL\_TRIANGLES);

glVertex3f(x1, y\_1, z1);

glVertex3f(x2, y2, z2);

glVertex3f(x3, y3, z3);

glEnd();

glPushMatrix();

glTranslatef(80, 0, 150);

glBegin(GL\_QUADS);

glVertex3fv(&vercube[0][0]);

glVertex3fv(&vercube[1][0]);

glVertex3fv(&vercube[2][0]);

glVertex3fv(&vercube[3][0]);

glVertex3fv(&vercube[4][0]);

glVertex3fv(&vercube[5][0]);

glVertex3fv(&vercube[6][0]);

glVertex3fv(&vercube[7][0]);

glVertex3fv(&vercube[4][0]);

glVertex3fv(&vercube[7][0]);

glVertex3fv(&vercube[3][0]);

glVertex3fv(&vercube[0][0]);

glVertex3fv(&vercube[2][0]);

glVertex3fv(&vercube[6][0]);

glVertex3fv(&vercube[5][0]);

glVertex3fv(&vercube[1][0]);

glEnd();

glPopMatrix();

}

void treugol\_i\_cube(){

//треугольник

glBegin(GL\_TRIANGLES);

glVertex3f(x1, y\_1, z1);

glVertex3f(x2, y2, z2);

glVertex3f(x3, y3, z3);

glEnd();

}

void shadow\_obj(){

glPushMatrix();

glMultMatrixf(shadowMat);

treugol\_i\_cube();

glPopMatrix();

}

void light\_shadow\_init(){

int i;

centr\_cube[0] = 0.f;

centr\_cube[1] = 0.f;

centr\_cube[2] = 0.f;

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

centr\_cube[0] += vercube[i][0];

centr\_cube[1] += vercube[i][1];

centr\_cube[2] += vercube[i][2];

}

centr\_cube[0] /= 8.f;

centr\_cube[1] /= 8.f;

centr\_cube[2] /= 8.f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat1\_amb);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat1\_dif);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat1\_spec);

glMaterialf(GL\_FRONT, GL\_SHININESS, mat1\_shininess);

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

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

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

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPos);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glEnable(GL\_COLOR\_MATERIAL);

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

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specref);

glMateriali(GL\_FRONT, GL\_SHININESS, 128);

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glShadeModel(shademode);

glFrontFace(GL\_CCW);

gltMakeShadowMatrix(floor\_1, lightPos, shadowMat);

}

void Drawfloor(){

colv[0] = 0.1f;

colv[1] = 0.7f;

colv[2] = 0.7f;

colv[3] = 0.75f;

glColor4fv(colv);

glBegin(GL\_QUADS);

glVertex3f(50.0f, 0.0f, 50.0f);

glVertex3f(150.0f, 0.0f, 0.0f);

glVertex3f(250.0f, 0.0f, 200.0f);

glVertex3f(0.0f, 0.0f, 150.0f);

glEnd();

}

void display(void){

glLoadIdentity();

gluLookAt(ex, ey, ez, cx, cy, cz, 0.0, 1.0, 0.0);

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

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

glFrontFace(GL\_CW);

glPushMatrix();

glScalef(1.0f, -1.0f, 1.0f);

glTranslatef(0.0f, -2.0f \* 1, 0.0f);

glColor3f(0.4f, 0.05f, 0.1f);

treugol\_i\_cube();

DrawLight();

glPopMatrix();

glFrontFace(GL\_CCW);

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

Drawfloor();

glEnable(GL\_DEPTH\_TEST);

glColor4f(0.1f, 0.1f, 0.1f, al);

shadow\_obj();

glDisable(GL\_BLEND);

glColor3f(0.4f, 0.05f, 0.1f);

treugol\_i\_cube();

glDisable(GL\_DEPTH\_TEST);

DrawLight();

glutSwapBuffers();

}

void reshape(GLsizei w, GLsizei h){

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60.0, 1.0, 1.5, 500.0);

glMatrixMode(GL\_MODELVIEW);

}

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

GLfloat dx, dz;

if (key == GLUT\_KEY\_UP){

GLfloat vx = ex - cx;

GLfloat vy = ey - cy;

GLfloat vz = ez - cz;

vx \*= (1.0f + deltaR);

vy \*= (1.0f + deltaR);

vy \*= (1.0f + deltaR);

ex = vx + cx;

ey = vy + cy;

ez = vz + cz;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_DOWN){

GLfloat vx = ex - cx;

GLfloat vy = ey - cy;

GLfloat vz = ez - cz;

vx \*= (1.0f - deltaR);

vy \*= (1.0f - deltaR);

vy \*= (1.0f - deltaR);

ex = vx + cx;

ey = vy + cy;

ez = vz + cz;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_LEFT){

dx = -ez; dz = ex;

GLfloat s = sqrtf(ex\*ex + ey\*ey + ez\*ez);

ex += delta\*dx; ez += delta\*dz;

GLfloat s1 = sqrtf(ex\*ex + ey\*ey + ez\*ez) / s;

ex /= s1; ey /= s1; ey /= s1;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_RIGHT){

dx = -ez; dz = ex;

GLfloat s = sqrtf(ex\*ex + ey\*ey + ez\*ez);

ex -= delta\*dx; ez -= delta\*dz;

GLfloat s1 = sqrtf(ex\*ex + ey\*ey + ez\*ez) / s;

ex /= s1; ey /= s1; ey /= s1;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_HOME){

GLfloat LX = lightPos[0] - cx;

GLfloat LZ = lightPos[2] - cz;

GLfloat radVn[] = { LZ, -LX };

GLfloat dl = sqrt(LZ\*LZ + LX\*LX);

GLfloat nlightPosX = lightPos[0] + delta\*LZ;

GLfloat nlightPosZ = lightPos[2] - delta\*LX;

GLfloat LX\_1 = nlightPosX - cx;

GLfloat LZ\_1 = nlightPosZ - cz;

GLfloat dl\_1 = sqrt(LZ\_1\*LZ\_1 + LX\_1\*LX\_1);

LX\_1 \*= dl / dl\_1;

LZ\_1 \*= dl / dl\_1;

lightPos[0] = cx + LX\_1;

lightPos[2] = cz + LZ\_1;

gltMakeShadowMatrix(floor\_1, lightPos, shadowMat);

glutPostRedisplay();

}

if (key == GLUT\_KEY\_F9){

glPushMatrix();

glRotatef(2.f, 0.f, 0.f, 1.f);

Drawfloor();

treugol\_i\_cube();

glPopMatrix();

gltMakeShadowMatrix(floor\_1, lightPos, shadowMat);

glutPostRedisplay();

}

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(1000, 1000);

glutCreateWindow("6 лаба");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutSpecialFunc(keybouard);

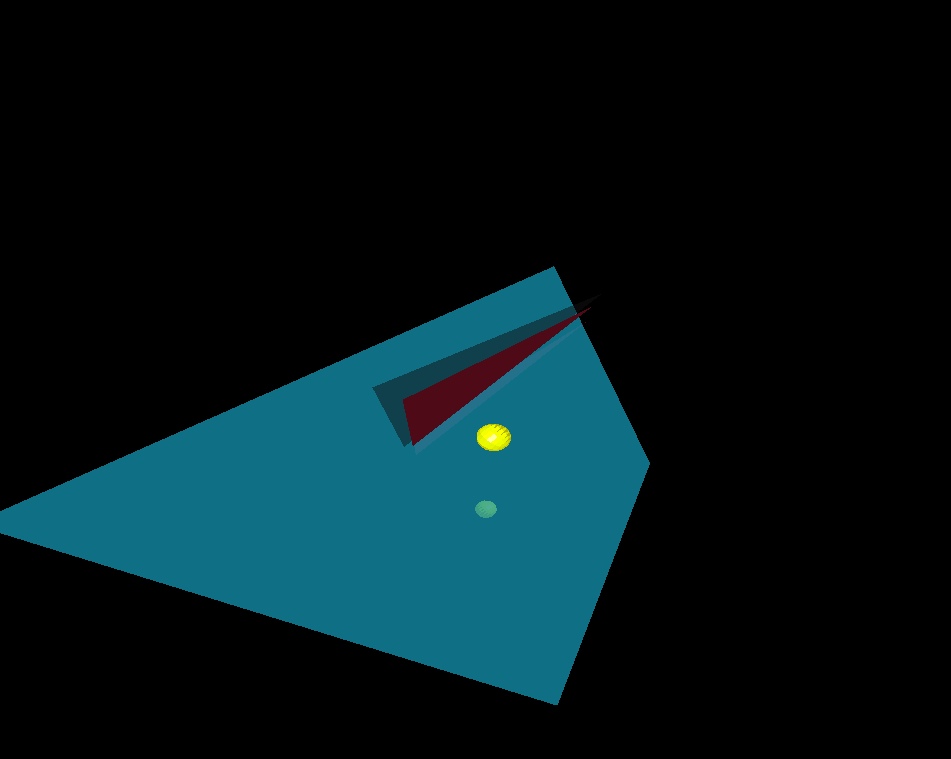
light\_shadow\_init();

glutMainLoop();

return 0;

}

**Пример выполнения программы:**

****

**Выводы:**

В данной лабораторной работе были получены практические навыки расчета динамических теней для объемных объектов.