ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Щекин С.В. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2 |
| **Работа с трехмерными объектами** |
| по дисциплине: Компьютерная графика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы**

Создать приложение, выводящее объемный объект средствами Open GL

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <GL/glut.h>

#include <cstdio>

#include <math.h>

char title[] = "КАЧАЛКА";

GLfloat ligtht\_radius = 5;

GLfloat angle\_y =0.0f;

GLfloat angle\_x = 0.0f;

GLfloat pos\_x = 0;

GLfloat pos\_y = 0.0f;

GLfloat zoom = -4.0f;

int refreshMills = 20;

//для света

GLfloat pos0[4] = { 0, 0, 0, 1 };

GLfloat pos1[4] = { 0, 0, 0, 1 };

GLfloat pos2[4] = { 0, 0, 0, 1 };

GLfloat angle1 = 0;

GLfloat angle2 = 120;

GLfloat angle3 = 240;

GLfloat diffuseColor0[] = { 1, 0.2, 0, 1.0 };

GLfloat diffuseColor1[] = { 0, 0.8,0, 1 };

GLfloat diffuseColor2[] = { 0.3, 0.5, 0.9, 1.0 };

GLfloat ambientColor2[] = { 0.6, 0.8, 0.8, 1.0 };

GLfloat ambientColor0[] = { 0.2, 0.2, 0.2, 1 };

GLfloat con\_pos[3] = { 1, 0 , -1 };

GLfloat miror\_cone[] = { 0.6, 0.0, 1.0, 1.0 };

GLfloat dif\_cone[] = { 0.2, 0.2, 0.2, 1.0 };

GLfloat AMB\_cone[] = { 0.0, 0.04, 0.2, 1.0 };

GLfloat EMIS\_cone[] = { 0.1, 0.1, 0.1, 1.0 };

GLfloat pot\_pos[3] = { 0, 0 , 0 };

GLfloat miror\_pot[] = { 0.3, 0.2, 0.4, 1.0 };

GLfloat dif\_pot[] = { 0.0, 1.0, 0.88, 1.0 };

GLfloat AMB\_pot[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat EMIS\_pot[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat miror\_ico[] = { 0.1, 0.2, 0.4, 1.0 };

GLfloat dif\_ico[] = { 0.2, 0.2, 0.2, 1.0 };

GLfloat AMB\_ico[] = { 0.4, 1.0, 0.3, 0.2 };

GLfloat EMIS\_ico[] = { 0.3, 0.3, 0.3, 0.3 };

GLfloat ico\_pos[3] = { -1, 0, 0 };

void display() {

//прозрачность

glEnable(GL\_BLEND);

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

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

glMatrixMode(GL\_MODELVIEW);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, AMB\_ico);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, miror\_ico);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, dif\_ico);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, EMIS\_ico);

glTranslatef(ico\_pos[0], ico\_pos[1], ico\_pos[2]);

glutSolidIcosahedron();

glTranslatef(-ico\_pos[0], -ico\_pos[1], -ico\_pos[2]);

glFlush();

glDisable(GL\_TEXTURE\_2D);

angle1 += 10;

pos0[0] = ligtht\_radius \* cos(angle1 / 180);

pos0[1] = ligtht\_radius \* sin(angle1 / 180);

angle2 += 10;

pos1[2] = ligtht\_radius \* cos(angle2 / 180);

pos1[1] = ligtht\_radius \* sin(angle2 / 180);

angle3 += 10;

pos2[0] = ligtht\_radius \* sin(angle3 / 180);

pos2[1] = ligtht\_radius \* cos(angle3 / 180);

glLoadIdentity();

glTranslatef(pos\_x, pos\_y, zoom);

glRotatef(angle\_x, 0.0f, 1.0f, 0.0f);

glRotatef(angle\_y, 1.0f, 0.0f, 0.0f);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glPopMatrix();

glutSwapBuffers();

}

void initGL() {

glClearColor(0.1f, 0.1f, 0.2f, 1.0f); // цвет фона

glClearDepth(1.0f); // глубина фона

glEnable(GL\_DEPTH\_TEST); // врубаем depth test

glDepthFunc(GL\_LEQUAL); // тип depth test

glShadeModel(GL\_SMOOTH);

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);

}

void process\_Normal\_Keys(unsigned char key, int x, int y) {

switch (key) {

// rot\_x

case '6':

angle\_x -= 5;

break;

case '4':

angle\_x += 5;

break;

// rot\_y

case '2':

angle\_y -= 5;

break;

case '8':

angle\_y += 5;

break;

// zoom

case ('w'):

zoom += 0.5;

break;

case ('s'):

zoom -= 0.5;

break;

default:

break;

// pos\_x

case ('d'):

pos\_x -= 0.5;

break;

case 'a':

pos\_x += 0.5;

break;

// pos\_y

case ('9'):

pos\_y += 0.1;

break;

case('3'):

pos\_y += -0.5;

break;

case ('+'):

ligtht\_radius += 0.5;

break;

case ('-'):

ligtht\_radius -= 0.5;

break;

case ('f'):

angle1 += 10;

pos0[0] = ligtht\_radius \* cos(angle1 / 180);

pos0[1] = ligtht\_radius \* sin(angle1 / 180);

break;

/\*\*/

case ('F'):

angle1 -= 10;

pos0[0] = ligtht\_radius \* cos(angle1 / 180);

pos0[1] = ligtht\_radius \* sin(angle1 / 180);

break;

case ('g'):

angle2 += 10;

pos1[2] = ligtht\_radius \* cos(angle2 / 180);

pos1[1] = ligtht\_radius \* sin(angle2 / 180);

break;

case ('G'):

angle2 -= 10;

pos1[2] = ligtht\_radius \* cos(angle2 / 180);

pos1[1] = ligtht\_radius \* sin(angle2 / 180);

break;

case ('h'):

angle3 += 10;

pos2[0] = ligtht\_radius \* sin(angle3 / 180);

pos2[1] = ligtht\_radius \* cos(angle3 / 180);

break;

case ('H'):

angle3 -= 10;

pos2[0] = ligtht\_radius \* sin(angle3 / 180);

pos2[1] = ligtht\_radius \* cos(angle3 / 180);

break;

case ('.'):

pot\_pos[1] += 0.1;

break;

case (','):

pot\_pos[1] -= 0.1;

break;

case 'q':

con\_pos[0] +=0.2;

ico\_pos[0] += -0.2;

break;

case 'Q':

con\_pos[0] += -0.2;

ico\_pos[0] += 0.2;

break;

}

}

void reshape(GLsizei width, GLsizei height) {

if (height == 0) height = 1;

GLfloat aspect = (GLfloat)width / (GLfloat)height;

glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0f, aspect, 0.1f, 100.0f);

}

void timer(int value) {

glutPostRedisplay();

glutTimerFunc(refreshMills, timer, 0);

}

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

glutInit(&argc, argv);

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

glutInitWindowSize(640, 480); // размеры окна

glutInitWindowPosition(50, 50); // позиция окна на экране

glutCreateWindow(title); // создаём окно

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHT1);

glEnable(GL\_LIGHT2);

initGL();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutTimerFunc(0, timer, 0);

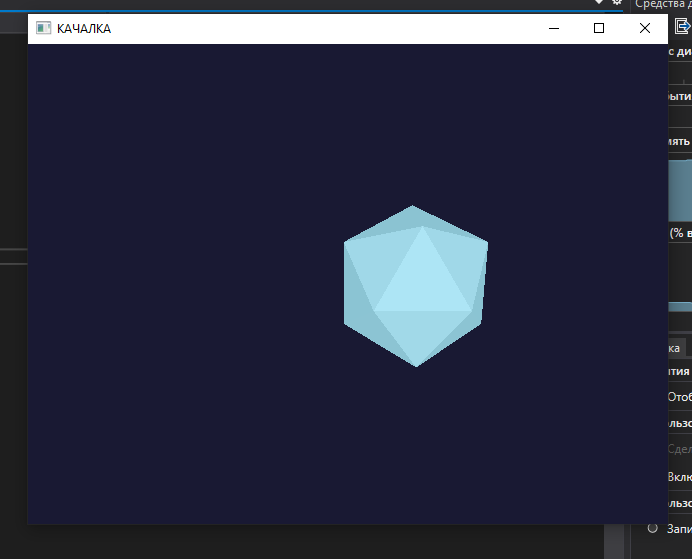
glutKeyboardFunc(process\_Normal\_Keys);

glutMainLoop();

return 0;

}

Результат работы



Вывод

Я освоил работу с OpenGL