КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

## Работа с буфером глубины, прозрачностью

по курсу: Компьютерная графика

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № _____ _____ _____
подпись, дата инициалы, фамилия

Санкт-Петербург 2022

1) Листинг программы:

Main.cpp

```
#include <iostream>
#include <cmath>

#include <GL/freeglut.h>

#define W_WIDTH 1280
#define W_HEIGHT 720

/* параметры материала шара */

// угол поворота камеры
double angle_x=0.0;
double angle_y=-5000.0;
// координаты вектора направления движения камеры
float lx=0.0f, ly=0.0f, lz=-1.0f;
// XZ позиция камеры
float x=0.0f, y=0.0f, z=5.0f;

int refreshMills = 60;

float fraction = -0.5f;
float fraction_angle = 0.1f;

float light_fraction = -0.5f;

bool use_mouse = true;

bool forward = false;
bool back = false;
bool left = false;
bool right = false;

bool light_up = false;
bool light_down = false;
bool light_forward = false;
bool light_back = false;
bool light_left = false;
bool light_right = false;

float mat_dif_Teapot[] = {0.2f, 0.2f, 0.2f, 1.0f};
float mat_spec_Teapot[] = {0.0f, 0.0f, 0.0f};
float mat_amb_Teapot[] = {0.4f, 4.0f, 4.0f};
float mat_shininess_Teapot = 0.1f * 128;

float mat_dif_Cube[] = {0.0f, 0.0f, 0.0f, 1.0f};
float mat_spec_Cube[] = {0.9f, 0.9f, 0.9f};
float mat_amb_Cube[] = {0.0f, 0.0f, 0.0f};
float mat_shininess_Cube = 0.1f * 128;
```

```
float mat_dif_Sphere[] = {0.0f, 0.0f, 0.0f, 1.0f};
float mat_spec_Sphere[] = {0.9f, 0.9f, 0.9f};
float mat_amb_Sphere[] = {0.0f, 0.0f, 0.0f};
float mat_shininess_Sphere = 0.1f * 128;

float mat_dif_Icosahedron[] = {0.9f, 0.9f, 9.9f, 1.0f};
float mat_spec_Icosahedron[] = {0.9f, 0.9f, 0.9f};
float mat_amb_Icosahedron[] = {0.9f, 0.9f, 0.9f};
float mat_shininess_Icosahedron = 0.1f * 128;

float mat_dif_center[] = {0.9f, 0.9f, 9.9f, 0.5};
float mat_spec_center[] = {0.9f, 0.9f, 0.9f};
float mat_amb_center[] = {0.9f, 0.9f, 0.9f};
float mat_shininess_center = 0.1f * 128;

float mat_dif_light[] = {0.0f, 0.0f, 0.0f, 1.0};

/* параметры источника света */
int index_light = 0;
GLfloat light_position[3][4] = {
   {0.0, 0.0, 0.0, 1.0},
   {0.0, 0.0, 0.0, 1.0},
   {0.0, 0.0, 0.0, 1.0}
};

GLfloat ambientColor1[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat diffuseColor1[] = { 0, 1, 0, 0.6 };
GLfloat emis1[] = { 0.1, 0.1, 0.1, 0.1 };

GLfloat ambientColor2[] = { 0.4, 0.2, 0.4, 1.0 };
GLfloat diffuseColor2[] = { 1, 1, 0, 0.6 };
GLfloat emis2[] = { 0.1, 0.1, 0.1, 0.1 };

GLfloat ambientColor3[] = { 0.6, 0.2, 0.6, 2.0 };
GLfloat diffuseColor3[] = { 1, 0, 0, 0.6 };
GLfloat emis3[] = { 0.1, 0.1, 0.1, 0.1 };

GLfloat Teapot_rotation = 0.0f;

void drawText2f(float x, float y, std::string text) {
   glColor3d(1.0, 0.0, 0.0);
   glRasterPos2f(x, y);
   glutBitmapString(GLUT_BITMAP_8_BY_13, (const unsigned char*)text.c_str());
}

void drawText3f(float x, float y, float z, std::string text) {
   glColor3d(1.0, 0.0, 0.0);
   glRasterPos3f(x, y, z);
   glutBitmapString(GLUT_BITMAP_8_BY_13, (const unsigned char*)text.c_str());
}
//прорисовка, для не искажения окна
```

```
void Reshape(int w, int h) {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(40.0, (GLfloat) w / h, 1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);

}

void move() {
    // camera
    if (forward) {
        z -= lz * fraction;
        x -= lx * fraction;
        y -= ly * fraction;
    }

    if (back) {
        z += lz * fraction;
        x += lx * fraction;
        y += ly * fraction;
    }

    if (left) {
        z += lx * fraction;
        x -= lz * fraction;
    }

    if (right) {
        z -= lx * fraction;
        x += lz * fraction;
    }

    // light
    if (light_forward) {
        light_position[index_light][0] += light_fraction;
    }

    if (light_back) {
        light_position[index_light][0] -= light_fraction;
    }

    if (light_left) {
        light_position[index_light][2] -= light_fraction;
    }

    if (light_right) {
        light_position[index_light][2] += light_fraction;
    }

    if (light_up) {
```

```
        light_position[index_light][1] -= light_fraction;
    }

    if (light_down) {
        light_position[index_light][1] += light_fraction;
    }

}

void keyUp(unsigned char key, int xx, int yy) {
    switch (key) {
            // camera
        case ('w'):
            forward = false;
            break;

        case ('s'):
            back = false;
            break;

        case ('a'):
            left = false;
            break;

        case ('d'):
            right = false;
            break;

            // light
        case ('u'):
            light_forward = false;
            break;

        case ('j'):
            light_back = false;
            break;

        case ('h'):
            light_left = false;
            break;

        case ('k'):
            light_right = false;
            break;

        case ('y'):
            light_up = false;
            break;

        case ('i'):
            light_down = false;
            break;
```

```
            // switch move light
        case ('t'):
            index_light += 1;
            if (index_light >= 3)
                index_light = 0;
            break;

            // включить (выключить курсор)
        case ('g'):
            use_mouse = !use_mouse;
            break;

            // выход
        case 27:
            //glutDestroyWindow ( Win.id );
            exit (0);
            break;

    }
}

void keyDown(unsigned char key, int xx, int yy) {
    switch (key) {

            // camera
        case ('w'):
            forward = true;
            break;

        case ('s'):
            back = true;
            break;

        case ('a'):
            left = true;
            break;

        case ('d'):
            right = true;
            break;


            // light
        case ('u'):
            light_forward = true;
            break;

        case ('j'):
            light_back = true;
            break;
```

```
        case ('h'):
            light_left = true;
            break;

        case ('k'):
            light_right = true;
            break;

        case ('y'):
            light_up = true;
            break;

        case ('i'):
            light_down = true;
            break;

    }
}

double sensivity = 0.001;

void mouseMove(int xx, int yy) {
    if (use_mouse) {
        angle_x -= (W_WIDTH/2 - xx) * sensivity;
        if ((angle_y + (W_HEIGHT/2 - yy) * sensivity < -4998.3) && (angle_y +
(W_HEIGHT/2 - yy) * sensivity > -5001.3)) {
            angle_y += (W_HEIGHT/2 - yy) * sensivity;
        }

        // lx = sin(angle_x);
        // lz = -cos(angle_x);
        // ly = sin(angle_y);
        lx = sin(angle_y) * sin(angle_x);
        ly = -cos(angle_y);
        lz = -sin(angle_y) * cos(angle_x);
        glutWarpPointer(
                    W_WIDTH / 2,
                    W_HEIGHT / 2
                    );
    }
}
//вывод на экран, созд сцены
void Display(void) {

    move();

    glLoadIdentity();

    gluLookAt(x, y, z,
            x+lx, y+ly,  z+lz,
            0.0f, 1.0f,  0.0f );
```

```
glClearColor(0, 0.3, 0.3, 1);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

drawText3f(0, 0, 0, "Teapot");
drawText3f(-10, 0, 10, "Dodecahedron");
drawText3f(10, 1, 10, "Cylinder");
drawText3f(0, 1, 20, "Sphere");

glEnable(GL_DEPTH_TEST);

glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseColor1);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientColor1);
glLightfv(GL_LIGHT0, GL_EMISSION, emis1);

glLightfv(GL_LIGHT0, GL_POSITION, light_position[0]);
glTranslatef(light_position[0][0], light_position[0][1], light_position[0][2]);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_light);
glutSolidSphere(0.2, 32, 32);
glTranslatef(-light_position[0][0], -light_position[0][1], -light_position[0][2]);

glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseColor2);
glLightfv(GL_LIGHT1, GL_AMBIENT, ambientColor2);
glLightfv(GL_LIGHT0, GL_EMISSION, emis1);

glLightfv(GL_LIGHT1, GL_POSITION, light_position[1]);
glTranslatef(light_position[1][0], light_position[1][1], light_position[1][2]);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_light);
glutSolidSphere(0.2, 32, 32);
glTranslatef(-light_position[1][0], -light_position[1][1], -light_position[1][2]);

glLightfv(GL_LIGHT2, GL_DIFFUSE, diffuseColor3);
glLightfv(GL_LIGHT2, GL_AMBIENT, ambientColor3);
glLightfv(GL_LIGHT0, GL_EMISSION, emis1);

glLightfv(GL_LIGHT2, GL_POSITION, light_position[2]);
glTranslatef(light_position[2][0], light_position[2][1], light_position[2][2]);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_light);
glutSolidSphere(0.2, 32, 32);
glTranslatef(-light_position[2][0], -light_position[2][1], -light_position[2][2]);

/* включаем освещение и источник света */
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
glEnable(GL_LIGHT2);

Teapot_rotation += 0.5f;
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb_Teapot);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_Teapot);
```

```
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spec_Teapot);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess_Teapot);
    glRotatef(Teapot_rotation, 0.0f, 1.0f, 0.0f);
    glTranslatef(0.0f, -2.0f, 0.0f);
    glutSolidTeapot(2);

    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb_Cube);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_Cube);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spec_Cube);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess_Cube);
    glRotatef(-Teapot_rotation, 0.0f, 1.0f, 0.0f);
    glTranslatef(-10.0f, 0.0f, 10.0f);
    //glutSolidCube(2.0);
    glutSolidDodecahedron();

    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb_Sphere);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_Sphere);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spec_Sphere);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess_Sphere);
    glTranslatef(10.0f, 0.0f, 10.0f);
    glutSolidSphere(2.0, 32, 32);

    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb_Icosahedron);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_Icosahedron);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spec_Icosahedron);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess_Icosahedron);
    glTranslatef(10.0f, 0.0f, -10.0f);
    glutSolidCylinder(2.0, 4, 32, 32);

    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb_center);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif_center);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spec_center);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess_center);
    glTranslatef(-10.0f, 0.0f, 0.0f);
    glutSolidDodecahedron();

    glFlush();

}

void timer(int value) {
    glutPostRedisplay();
    glutTimerFunc(1000/refreshMills, timer, 0);
}

int main(int argc, char ** argv) {
    glutInit( & argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowPosition(150, 50);
    glutInitWindowSize(W_WIDTH, W_HEIGHT);
    glutCreateWindow("light");
    glutReshapeFunc(Reshape);
```

```
    glutDisplayFunc(Display);

    glutSetKeyRepeat(GLUT_KEY_REPEAT_OFF);

    glutKeyboardFunc(keyDown);
    glutKeyboardUpFunc(keyUp);

    glutPassiveMotionFunc(mouseMove);

    glutTimerFunc(0, timer, 0);

    glutMainLoop();

}
```

2) Вывод: были изучены принципы работы с буфером глубины и прозрачности.