*Горшков Алексей Олегович*

*Группа 4212*

**Отчет о выполнении лабораторной работы №3 «Установка отображаемого пространства»**

1. **Задание:**

*Создайте программу, где геометрический примитив из библиотеки GLUT помещен в начало МСК, освещен неподвижным источником и в программе реализованы функции обработки нажатий клавиатуры и клавиш мыши в соответствии с таблицей.*

(Вариант 4):

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | октаэдр | кл. X - наблюдатель вращается в пл. (осьZ, Е) вокруг объекта по ч.с.  по кл. Y - наблюдатель вращается в пл. (осьZ, Е) вокруг объекта против ч.с. | левая кл. – объект масштабируется с увеличением размеров по X,Y  правая кл. – объект масштабируется с уменьшением размеров по X,Y |

1. **Листинг программы:**

#include <gl\glut.h>

#include <math.h>

#include <windows.h>

//Переменные для вращения наблюдателя.

GLfloat angle\_rotate{};

GLboolean no\_rotate{ true }, clockwise{ false };

//Переменные для масштабирования объекта.

GLfloat scale\_last{ 1.0 }, scale\_next{ 1.0 };

GLboolean flag\_scale\_up{ false }, flag\_scale\_down{ false };

void init(void)

{

glClearColor(0.5, 0.5, 0.71, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 15);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

//Функция для "отлова" событий мыши ("мышиных" событий).

void MouseClickEvents(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP && (scale\_next + 0.1) <= 1.6)

{

flag\_scale\_up = true;

flag\_scale\_down = false;

}

else if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_UP && (scale\_next - 0.1) >= 0.4)

{

flag\_scale\_down = true;

flag\_scale\_up = false;

}

}

//Функция для "отлова" событий клавиатуры (клавиатурных событий).

void KeyBoardEvents(unsigned char key, int x, int y)

{

if (key == static\_cast<unsigned char>('X'))

{

no\_rotate = false;

clockwise = true;

}

else if (key == static\_cast<unsigned char>('Y'))

{

no\_rotate = false;

clockwise = false;

}

}

void myDisplay()

{

glPushMatrix(); //Сохраняем VM = 1.

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

glEnable(GL\_DEPTH\_TEST);//Включаем тестовую глубину для реализации изображения.

// Задаем VM = Fwe \* (матрицу для движения по окружности наблюдателя).

gluLookAt(5\*sin(angle\_rotate), 2, 6\*cos(angle\_rotate), 0, 0, 0, 0, 1, 0);

GLfloat myLightPosition[] = { 3.0, 0.0, 4.0, 1.0};// Источник света в CKw.

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition); /\*Позиция источника света будет преобразована в CKe.\*/

//Включаем источник света.

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPushMatrix(); //Сохраняем VM=Fwe или преобразованную в результате движения по окружности наблюдателя.

glScalef(scale\_next, scale\_next, 1.0);//Сохранение масштабности сохранённой и не изменённой VM = Fwe.

//Масштабирование по Х и У VM=Fwe по нажатию кнопок мыши.

if (flag\_scale\_up)

{

scale\_last = scale\_next;

scale\_next += 0.1;

glScalef(scale\_next / scale\_last, scale\_next / scale\_last, 1.0);

flag\_scale\_up = false;

}

else if (flag\_scale\_down)

{

scale\_last = scale\_next;

scale\_next -= 0.1;

glScalef(scale\_next / scale\_last, scale\_next / scale\_last, 1.0);

flag\_scale\_down = false;

}

glutSolidOctahedron();//Создание октаэдра.

glPopMatrix(); // Восстанавливаем VM=Fwe или преобразованную в результате движения по окружности наблюдателя.

glDisable(GL\_LIGHTING); //Выключаем освещение для рисования линий и прочего в цвете.

//Рисуем координатные оси разного цвета в CKe.

glBegin(GL\_LINES);

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(5, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 3, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 5);

glEnd();

//Расположение источника света для наглядности в СКе.

glPointSize(20.0);

glBegin(GL\_POINTS);

glColor3f(1.0, 1.0, 1.0);

glVertex3f(3.0, 0.0, 4.0);

glEnd();

//Рисуем направление к источнику света в CKe.

glBegin(GL\_LINES);

glColor3f(1.0, 1.0, 1.0); glVertex3f(3.0, 0.0, 4.0); glVertex3f(0.0, 0.0, 0.0);

glEnd();

glPopMatrix();

glutSwapBuffers();

}

void myReshape(int width, int height)

{

if (width / height < 1) glViewport(0, 0, width, width);

else glViewport(0, 0, height, height);

}

void myIdle()

{

if (!no\_rotate)

{

if (clockwise)

angle\_rotate -= 0.0025;

else

angle\_rotate += 0.0025;

Sleep(15);

}

if (angle\_rotate >= 360 || angle\_rotate <= -360) angle\_rotate = 0;

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("OpenGl\_Lab3");

glutDisplayFunc(myDisplay);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

glutKeyboardFunc(KeyBoardEvents);

glutMouseFunc(MouseClickEvents);

init();

glutMainLoop();

}

1. **Работа программы:**

    