***ЛАБОРАТОРНАЯ РАБОТА 4***

**По дисциплине «Алгоритмические средства компьютерной графики»**

***ТЕМА. Пространственные преобразования и построение проекции***

**Цель работы:** Изучить реализацию пространственных преобразований в OpenGL и изучить работу с матрицей проекции

**Порядок работы:**

* 1. В среде Visual Studio создайте пустое решение (solution) с именем PW5. В нем создайте консольный проект на языке C++ Win32 Console Application. Назовите его PW5.
  2. В проект добавьте пустой файл с исходным кодом source.cpp
  3. Наберите следующий текст программы:

#if defined(linux) || defined(\_WIN32)

#include <GL/glut.h> /\*Для Linux и Windows\*/

#else

#include <GLUT/GLUT.h> /\*Для Mac OS\*/

#endif

void reshape(int w, int h);

void display();

void processNormalKeys(unsigned char key, int x, int y);

void processSpecialKeys(int key, int x, int y);

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowSize(800, 600);

glutCreateWindow("OpenGL lesson 6");

glutReshapeFunc(reshape);

glutDisplayFunc(display);

glutKeyboardFunc(processNormalKeys);

glutSpecialFunc(processSpecialKeys);

glutMainLoop();

return 0;

}

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-15, 15, -15, 15, -15, 15);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glClearColor(1, 1, 1, 0);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0);

glutWireTeapot(5);

glutSwapBuffers();

}

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

{

if (key == 27)

exit(0);

}

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

switch (key) {

case GLUT\_KEY\_UP:

glMatrixMode(GL\_MODELVIEW);

glTranslated(0, 1, 0);

display();

break;

case GLUT\_KEY\_DOWN:

glMatrixMode(GL\_MODELVIEW);

glTranslated(0, -1, 0);

display();

break;

}

}

* 1. Запустите программу. Нажмите клавиши UP, DOWN и ESC. Проанализируйте работу программы. Изучите текст программы.
  2. Измените третьи параметры функции glTranslated(). Запустите приложение и проанализируйте его работу.
  3. Дополните программу таким образом, что бы на экране по мимо объекта прорисовывались оси OX, OY и OZ. Запустите приложение и проанализируйте его работу.
  4. Измените параметры функции glOrtho(). Запустите приложение и проанализируйте его работу.
  5. Задайте обработку следующих клавиш:
     + LEFT – движение влево,
     + RIGHT – движение вправо,
     + + - увеличение объектов,
     + - - уменьшение объектов,
     + HOME – вращение объектов против часовой стрелки вокруг оси OX,
     + END – вращение объектов против часовой стрелки вокруг оси OY,
     + DELETE – вращение объектов против часовой стрелки вокруг оси OZ,
     + PG UP – вращение объектов по часовой стрелки вокруг всех трех осей одновременно,
     + PG DN – вращение объектов против часовой стрелки вокруг всех трех осей одновременно

Запустите приложение и проанализируйте его работу.

* 1. Дополните программу обработкой клавиш LEFT – движение объекта влево и RIGHT – движение объекта в право.
  2. Замените функцию glOrtho(…) на gluPerspective(60, 1, 0, 20). Запустите приложение и проанализируйте его работу. Измените параметры функции gluPerspective(). Запустите приложение и проанализируйте его работу.
  3. После функции gluPerspective() добавьте функцию gluLookAt(0, 0, 10, 0, 0, 0, 0, 1, 0); Запустите приложение и проанализируйте его работу. Замените предпоследний параметр этой функции на -1. Запустите приложение и проанализируйте его работу. Поэкспериментируйте с параметрами функции gluLookAt().
  4. Исследуйте работу следующих функций (укажите их вместо функции glutWireTeapot(5)):
* void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks)
* void glutWireSphere(GLdouble radius, GLint slices, GLint stacks)

Команда glutSolidSphere() строит сферу, а glutWireSphere() -каркас сферы радиусом radius, число разбиений сферы вокруг оси z задается параметром slices, а вдоль оси z параметром stacks.

* void glutSolidCube(GLdouble size)
* void glutWireCube(GLdouble size)

Эти команды строят куб или каркас куба с центром в начале координат и длиной ребра size.

* void glutSolidCone(GLdouble base, GLdouble height, GLint slices, GLint stacks)
* void glutWireCone(GLdouble base, GLdouble height, GLint slices, GLint stacks)

Эти команды строят конус или его каркас высотой height и радиусом основания base, расположенный вдоль оси z. Основание находится в плоскости z=0. Остальные параметры имеют тот же смысл, что и в предыдущих командах.

* void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings)
* void glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings)

Эти команды строят тор или его каркас в плоскости z=0. Внутренний и внешний радиусы задаются параметрами innerRadius, outerRadius. Параметр nsides задает число сторон в кольцах, составляющих ортогональное сечение тора, а rings- число радиальных разбиений тора.

* void glutSolidTetrahedron(void)
* void glutWireTetrahedron (void)

Эти команды строят тетраэдр (правильную треугольную пирамиду) или его каркас, при этом радиус описанной сферы вокруг него равен 1.

* void glutSolidOctahedron(void)
* void glutWireOctahedron(void)

Эти команды строят октаэдр или его каркас, радиус описанной вокруг него сферы равен 1.

* void glutSolidDodecahedron(void)
* void glutWireDodecahedron(void)

Эти команды строят додекаэдр или его каркас, радиус описанной вокруг него сферы равен квадратному корню из трех.

* void glutSolidIcosahedron(void)
* void glutWireIcosahedron(void)

Эти команды строят икосаэдр или его каркас, радиус описанной вокруг него сферы равен 1.