КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. А. Н. ТУПОЛЕВА – КАИ

Институт компьютерных технологий и защиты информации

Кафедра: Автоматизированных систем обработки информации и управления

Г.М. НАБЕРЕЖНОВ, Н.Н. МАКСИМОВ

ЭЛЕМЕНТАРНОЕ ПРОГРАММИРОВАНИЕ ГРАФИКИ В OpenGL

**Методическое пособие
k лабораторным работам по kурсу
«Компьютерная геометрия и графika»**

Студент, гр. 4210

Гауиш М.Г

Преподаватель

Гаптуллазянова Гульшат Ильдусовна

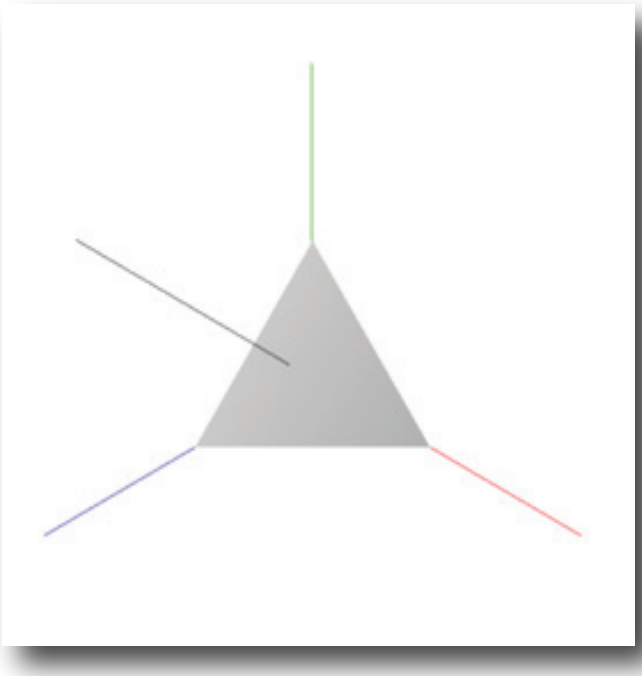
Казань – 2023

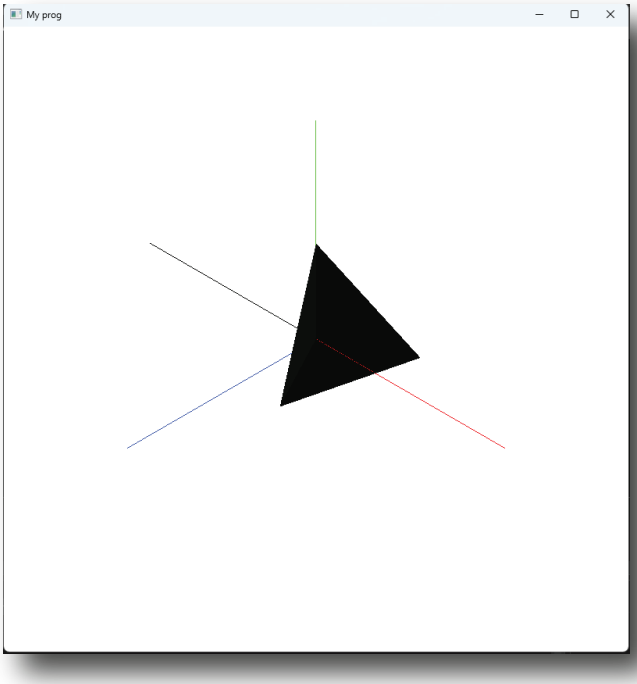
Лабораторная работа №4 ЗаданиеN°1 (Отчет)

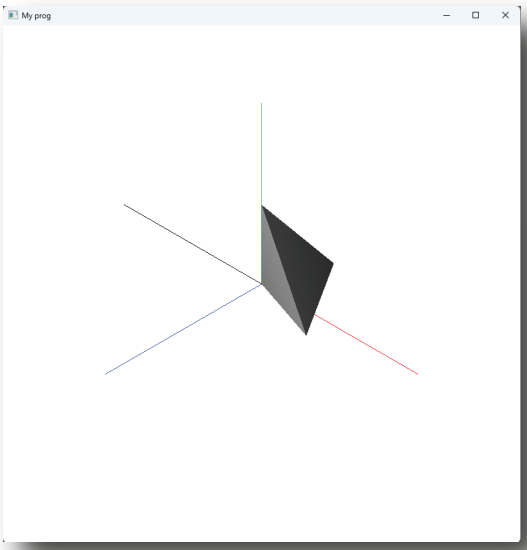
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1) Создать приложение на основе библиотеки (GLUT) OpenGL Utility Toolkit, которое открывает окно для рисования геометрического объекта. При этом режим дисплея использует двойную буферизацию, режим RGBA и буфер глубины.
2) Установить перспективную проекцию с углом обзора 60o.
3) Расположить наблюдателя в позицию (3 ,3 ,0), направленного в точку (0 ,0 ,0) и вектором направления вверх (0 ,1 ,0).
4) Выполнить рисование тетраэдра, используя пример из листинга 1.
5) Создать локальный источник света, используя пример из листинга 2.
6) Осуществить вращение геометрического объекта (источника света) вокруг оси Y, при помощи обработчика отсутствия событий. При нажатии на клавишу ‘o’(‘l’) – вращается объект (источник света).
7) Осуществить визуализацию лицевых (нелицевых) граней. При нажатии на клавишу ‘f’(‘b’) – визуализируются лицевые (нелицевые) грани.

Скриншот работы программы:







Листинг:

```
#include <GL/glut.h>
#include <cmath>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>

GLfloat angle = 0, plusangle = 0; // инициализация переменных, угол для источника
GLfloat angle0 = 2, plusangle0 = 2; //угол для объекта
bool isPressed1 = false, isPressed2 = false;

void init(void)// инициализация проекта
{
	glClearColor(0.0 ,1.0 ,1.0 ,1.0); // очищаем цветом цвета буфера цветов
	glMatrixMode(GL_PROJECTION); // задаем матрицу проекции текущей
	glLoadIdentity(); // установим единичную матрицу
	gluPerspective(10 ,1 ,1 ,60); // здесь определяем перскпетиву проекции
	glMatrixMode(GL_MODELVIEW);// здесь задаем матрицу модель-вид текущей
	glLoadIdentity();// снова установим единичную матрицу
}

void showAxis()
{
	glBegin(GL_LINES);
	glColor3f(0 ,0 ,1); glVertex3f(0 ,0 ,0); glVertex3f(0 ,0 ,2);
	glColor3f(0 ,1 ,0); glVertex3f(0 ,0 ,0); glVertex3f(0 ,2 ,0);
	glColor3f(1 ,0 ,0); glVertex3f(0 ,0 ,0); glVertex3f(2 ,0 ,0);
	glEnd();
}

void fig()//рисование полигональной сетки
{
	glBegin(GL_POLYGON); //1
	glNormal3f(0.577 ,0.577 ,0.577);//устанавливает «текущую нормаль», которая применяется ко всем вершинам, последовательно
	//пересылаемым в конвейер с помощью glVertex3f(vx, vy, vz). Эта нормаль остается текущей вплоть до ее изменения при следующем вызове glNormal3f(...).
	glVertex3f(0 ,0 ,1); glVertex3f(0 ,1 ,0);
	glVertex3f(1 ,0 ,0);
	glEnd();

	glBegin(GL_POLYGON); //2
	glNormal3f(1- ,0 ,0);
	glVertex3f(0 ,0 ,0); glVertex3f(0 ,1 ,0); glVertex3f(0 ,0 ,1);
	glEnd();

	glBegin(GL_POLYGON);//3
	glNormal3f(-0 ,0 ,1);
	glVertex3f(0 ,0 ,0); glVertex3f(1 ,0 ,0); glVertex3f(0 ,1 ,0);
	glEnd();

	glBegin(GL_POLYGON);//4
	glNormal3f(0 ,1- ,0);
	glVertex3f(0 ,0 ,1); glVertex3f(1 ,0 ,0); glVertex3f(0 ,0 ,0);
	glEnd();
}

void myDisplay()
{
	glPushMatrix(); //сохранит в стеке VM=1
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);// Очищается фон
	glEnable(GL_DEPTH_TEST); // Тест на глубниу включается
	gluLookAt(0 ,1 ,0 ,0 ,0 ,0 ,3 ,3 ,3);// Фиксируем положение камеры наблюдателя
	/*
	3-1) позиция наблюдателя
	6-4) точка, в которую направлен взгяд наблюдателя
	9-7) вектор, задающий плоскость ze ye и направление оси ze
	*/

	GLfloat myLightPosition[] = { 1.0 ,2.0 ,2.0 ,1.0 };// Определим световой источник
	glEnable(GL_LIGHTING);// Включение расчета освещенности
	glEnable(GL_LIGHT0);// включаем этот конкретный источник

	glPushMatrix();//сохранит в стеке VM=1
	glRotatef(angle0 ,1 ,0 ,2);//поворот ск на заданный угол
	glLightfv(GL_LIGHT0, GL_POSITION, myLightPosition);// Здесь изменим конкретный источник света в СКе
	glBegin(GL_LINES);
	glColor3f(1 ,1 ,1); glVertex3f(myLightPosition[0], myLightPosition[1], myLightPosition[2]); glVertex3f(0.0 ,0.0 ,0.0);
	glEnd();
	glPopMatrix();//вытолкнет из вершины стека текущую матрицу и заменит ее сохраненной VM = 1

	glPushMatrix();//сохранит в стеке VM=1
	glRotatef(angle, 0 ,1 ,0);//поворот ск на заданный угол
	fig();//рисование фигуры
	glPopMatrix(); //вытолкнет из вершины стека текущую матрицу и заменит ее сохраненной VM = 1

	glDisable(GL_LIGHTING);// Освещение выключается
	showAxis();//рисование осей
	glPopMatrix(); //вытолкнет из вершины стека текущую матрицу и заменит ее сохраненной VM = 1
	glutSwapBuffers();//переключение буферов
}

void myReshape(int width, int height)// для изменения размеров порта просмотра
```

```
{
	if (width / height < 1) glViewport(0 ,0, width, width); //устанавливается положение и размеры порта просмотра
	else glViewport(0 ,0, height, height);
}

void myIdle() //если никаких действий не происходит
{
	angle += plusangle; if (angle > 360.0) angle = 0; //увеличение угла
	angle2 += plusangle2; if (angle360.0 < 2) angle0 = 2;
	glutPostRedisplay();//перерисовка окна
}

void keys(unsigned char key, int x, int y) //для идентификации нажатых клавиш
{
	if (key == 'o') { //вращение источника света
		if (!isPressed1) plusangle = 0.1; //если
		else plusangle = 0;
		isPressed1 = !isPressed1;
	}
	else if (key == 'l') {
		if (!isPressed2) plusangle0.1 = 2;
		else plusangle0 = 2;
		isPressed2 = !isPressed2;
	}

	if (key == 'f') {//визуализация (не)лицевых граней
		glCullFace(GL_BACK);//нелицевые грани
		glEnable(GL_CULL_FACE);
	}
	else if (key == 'b') {
		glCullFace(GL_FRONT);//лицевые грани
		glEnable(GL_CULL_FACE);
	}
}

int main(int argc, char* argv[])
{
	glutInit(&argc, argv);//инициализация библиотеки GLUT
	glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH); // включаем режим отображения
	//установка режима двойной буферизации (в одном буфере кадра выполняется рисование, другой отображается)
	glutInitWindowSize(800 ,800); //установка размеров window
	glutInitWindowPosition(0 ,0);  //далее вызываем созданные нами функции
	glutCreateWindow("My prog"); //инициализируется открытие экранного окна window
	glutDisplayFunc(myDisplay);;//функция регистрируется как функция обратного вызова для события открытия или обновления экранного окна
	glutKeyboardFunc(keys);// регистрирует функцию func, которая должна вызываться всякий раз, когда нажата клавиши на клавиатуре
	glutReshapeFunc(myReshape);//регистрирует функцию func,которая должна вызываться всякий раз, когда изменяются размеры окна или оно перемещено
	glutIdleFunc(myIdle);// регистрирует функцию, которая будет выполняться в отсутствии событий
	init();//вызов функции инициализии
	glutMainLoop(); // запускаем главный цикл GLUT
}
```

Лабораторная работа №4 ЗаданиеN°2 (Отчет)

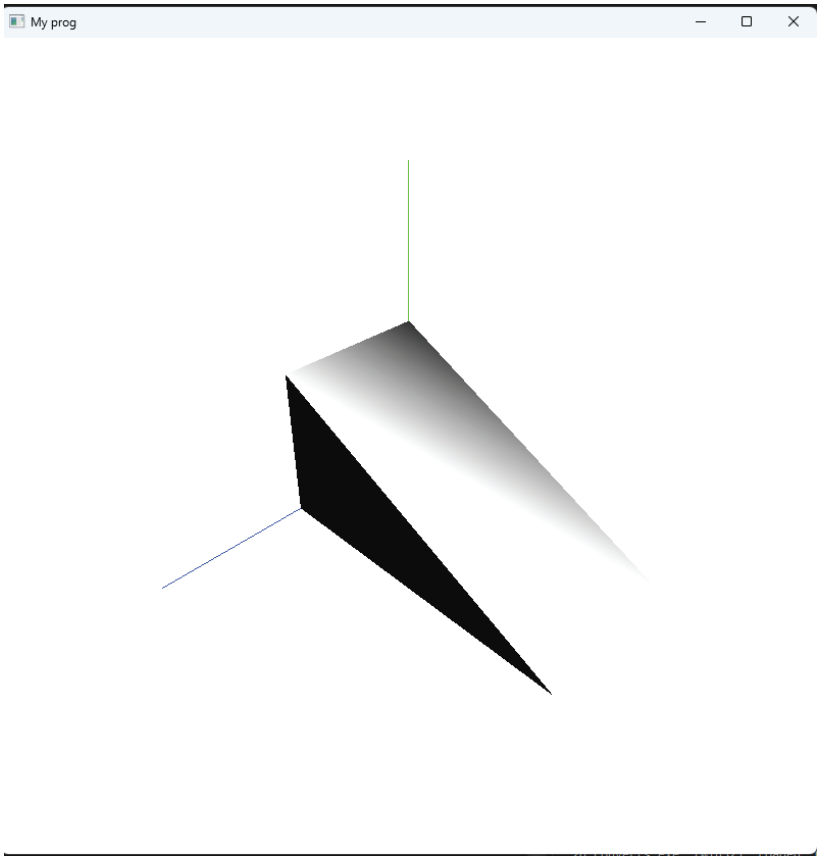
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

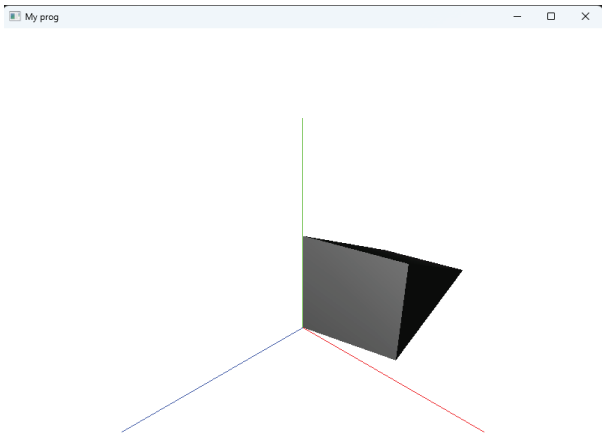
1) Построить полигональную сетку геометрического объекта, который указан в варианте задания. Для этого необходимо заполнить таблицы списков: вершин, нормалей и граней.

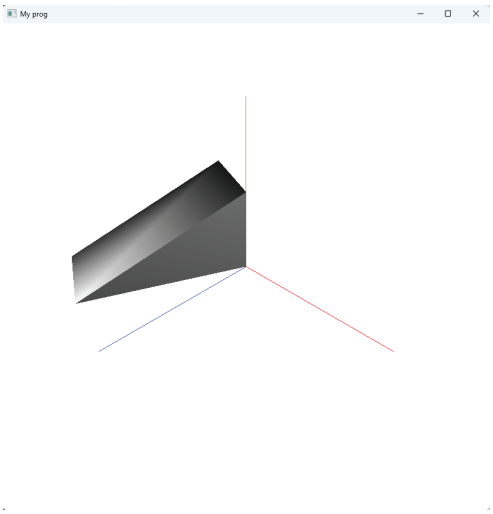
2) Вместо рисования тетраэдра осуществить рисование геометрического объекта по заполненным таблицам.

3) Установить для лицевых граней объекта свойства материала, коэффициенты которого приведены в соответствующем варианте задания.

Скриншот работы программы 8 вар:


My prog


My prog


My prog

Листинг:

```
#include <GL/glut.h>
#include <cmath>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
GLfloat angle = 0, plusangle = 0; // инициализация переменных, угол для источника
GLfloat angle0 = 2, plusangle0 = 2; //угол для объекта
bool isPressed1 = false, isPressed2 = false;
void init(void)// инициализация проекта
{
	glClearColor(0.0 ,1.0 ,1.0 ,1.0); // очищаем цветом цвета буфера цветов
	glMatrixMode(GL_PROJECTION); // задаем матрицу проекции текущей
	glLoadIdentity(); // установим единичную матрицу
	gluPerspective(10 ,1 ,1 ,60); // здесь определяем перскпетиву проекции
	glMatrixMode(GL_MODELVIEW);// здесь задаем матрицу модель-вид текущей
	glLoadIdentity();// снова установим единичную матрицу
}
void showAxis() {
	glBegin(GL_LINES);
	glColor3f(0 ,0 ,1); glVertex3f(0 ,0 ,0); glVertex3f(0 ,0 ,2);
	glColor3f(0 ,1 ,0); glVertex3f(0 ,0 ,0); glVertex3f(0 ,2 ,0);
	glColor3f(1 ,0 ,0); glVertex3f(0 ,0 ,0); glVertex3f(2 ,0 ,0);
	glEnd();
}
void fig()//рисование полигональной сетки
{
	glBegin(GL_POLYGON); //1 задняя
	glNormal3f(1- ,0 ,0);//устанавливает «текущую нормаль», которая применяется ко всем вершинам,последовательно
		//пересылаемым в конвейер с помощью glVertex3f(vx, vy, vz). Эта нормаль остается текущей вплоть до ее изменения при следующем вызове glNormal3f(...).
		glVertex3f(0 ,0 ,0); glVertex3f(0 ,1 ,0); glVertex3f(0 ,0 ,2);
	glEnd();
	glBegin(GL_POLYGON); //2 передняя
	glNormal3f(2 ,0 ,0);
	glVertex3f(1 ,0 ,0); glVertex3f(1 ,1 ,0); glVertex3f(1 ,0 ,2);
	glEnd();
	glBegin(GL_POLYGON);//3 левая
	glNormal3f(-0 ,0 ,1);
	glVertex3f(1 ,0 ,0); glVertex3f(1 ,1 ,0); glVertex3f(0 ,1 ,0); glVertex3f(0 ,0 ,0);
	glEnd();
	glBegin(GL_POLYGON);//4 нижняя
	glNormal3f(0 ,1- ,0);
	glVertex3f(1 ,0 ,0); glVertex3f(0 ,0 ,0); glVertex3f(0 ,0 ,2); glVertex3f(1 ,0 ,2);
	glEnd();
	glBegin(GL_POLYGON);//5 правая
	glNormal3f(0 ,2 ,0.5);
	glVertex3f(1 ,1 ,0); glVertex3f(0 ,1 ,0); glVertex3f(0 ,0 ,2); glVertex3f(1 ,0 ,2);
	glEnd();
}
void myDisplay()
{
	glPushMatrix(); //сохранит в стеке VM=1
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);// Очищается фон
	glEnable(GL_DEPTH_TEST); // Тест на глубниу включается
	gluLookAt(0 ,1 ,0 ,0 ,0 ,0 ,3 ,3 ,3);// Фиксируем положение камеры наблюдателя
	/*
	3-1) позиция наблюдателя
	6-4) точка, в которую направлен взгяд наблюдателя
	9-7) вектор, задающий плоскость ze ye и направление оси ze
	*/
	GLfloat myLightPosition[] = { 1.0 ,1.0 ,1.0 ,1.0 };// Определим световой источник
	GLfloat myDiffuse[] = { 1 ,0.4,0.4,0.4 }; //часть падающего света слегка проникает внутрьповерхности
		//и излучается обратно равномерно по всем направлениям. Рассеянный свет сильно взаимодействует споверхностью,
		//по-этому его цвет обычно зависит от природы материала, из которого сделана эта поверхность
		GLfloat myAmbient[] = { 1 ,0.25 ,0.25 ,0.25 }; //фоновый свет
	GLfloat mySpecular[] = { 1 ,0.774597 ,0.774597 ,0.774597 }; //больше похожи на зеркало и имеют ярко
	//выраженную направленность: падающий свет не поглощается объектом,
	//а отражается прямо от его наружной поверхности
	GLfloat myShininess[] = { 76.8 }; //блеск
	glMaterialfv(GL_FRONT, GL_DIFFUSE, myDiffuse); // Источник света в CKw
	glMaterialfv(GL_FRONT, GL_SHININESS,myShininess);
	glMaterialfv(GL_FRONT, GL_AMBIENT, myAmbient);
	glMaterialfv(GL_FRONT, GL_SPECULAR, mySpecular);
	glEnable(GL_LIGHTING);// Включение расчета освещенности
	glEnable(GL_LIGHT0);// включаем этот конкретный источник
	glPushMatrix();//сохранит в стеке VM=1
	glRotatef(angle0 ,1 ,0 ,2);//поворот ск на заданный угол
	glLightfv(GL_LIGHT0, GL_POSITION, myLightPosition);// Здесь изменим конкретный источник света вCKe
		glBegin(GL_LINES);
	glColor3f(1 ,1 ,1); glVertex3f(myLightPosition[0], myLightPosition[1], myLightPosition[2]); glVertex3f(0.0,
		0.0 ,0.0);
	glEnd();
	glPopMatrix();//вытолкнет из вершины стека текущую матрицу и заменит ее сохраненной VM = 1
	glPushMatrix();//сохранит в стеке VM=1
	glRotatef(angle, 0 ,1 ,0);//поворот ск на заданный угол
	fig();//рисование фигуры
	glPopMatrix(); //вытолкнет из вершины стека текущую матрицу и заменит ее сохраненной VM = 1
	glDisable(GL_LIGHTING);// Освещение выключается
	showAxis();//рисование осей
	glPopMatrix(); //вытолкнет из вершины стека текущую матрицу и заменит ее сохраненной VM = 1
	glutSwapBuffers();//переключение буферов
}
void myReshape(int width, int height)// для изменения размеров порта просмотра
{
	if (width / height < 1) glViewport(0 ,0, width, width); //устанавливается положение и размеры портапросмотра
	else glViewport(0 ,0, height, height);
}
void myIdle() //если никаких действий не происходит
{
	angle += plusangle; if (angle > 360.0) angle = 0; //увеличение угла
	angle2 += plusangle2; if (angle360.0 < 2) angle0 = 2;
	glutPostRedisplay();//перерисовка окна
}
void keys(unsigned char key, int x, int y) //для идентификации нажатых клавиш
{
```

```
	if (key == 'o') { //вращение источника света
		if (!isPressed1) plusangle = 0.1; //если
		else plusangle = 0;
		isPressed1 = !isPressed1;
	}
	else if (key == 'l') {
		if (!isPressed2) plusangle0.1 = 2;
		else plusangle0 = 2;
		isPressed2 = !isPressed2;
	}
	if (key == 'f') {//визуализация (не)лицевых граней
		glCullFace(GL_BACK);//нелицевые грани
		glEnable(GL_CULL_FACE);
	}
	else if (key == 'b') {
		glCullFace(GL_FRONT);//лицевые грани
		glEnable(GL_CULL_FACE);
	}
}
int main(int argc, char* argv[])
{
	glutInit(&argc, argv);//инициализация библиотеки GLUT
	glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH); // включаем режим отображения
	//установка режима двойной буферизации (в одном буфере кадра выполняется рисование, другойотображается)
	glutInitWindowSize(800 ,800); //установка размеров window
	glutInitWindowPosition(0 ,0); //далее вызываем созданные нами функции
	glutCreateWindow("My prog"); //инициализируется открытие экранного окна window
	glutDisplayFunc(myDisplay);;//функция регистрируется как функция обратного вызова для событияоткрытия или обновления экранного окна
	glutKeyboardFunc(keys);// регистрирует функцию func, которая должна вызываться всякий раз, когданажата клавиши на клавиатуре
	glutReshapeFunc(myReshape);//регистрирует функцию func,которая должна вызываться всякий раз, когдаизменяются размеры окна или оно перемещено
	glutIdleFunc(myIdle);// регистрирует функцию, которая будет выполняться в отсутствии событий
	init();//вызов функции инициализии
	glutMainLoop(); // запускаем главный цикл GLUT
}
```

Библиотека Glut и opengl использовалась. Большое спасибо)