Введение.

Создаем проект Win32 Console.

Для начала сделаем несколько определений . Рендеринг - это процесс , с помощью которых компьютер создает образы обьектов . Обьекты состоят из геометрических примитивов - точек , линий , полигонов , которые определяются с помощью вершин . Пиксел - элементарный наименьший видимый элемент дисплея . Информация о пикселах хранится в памяти в форме bitplane - областей , где на каждый пиксел отводится по 1 биту. Bitplanes организованы в framebuffer . Рассмотрим простой пример - как нарисовать белый прямоугольник на черном фоне :

```
main() {
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
glVertex3f (0.25, 0.25, 0.0);
glVertex3f (0.75, 0.25, 0.0);
glVertex3f (0.75, 0.75, 0.0);
glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
}
```

glClearColor() устанавливает черный цвет фона , glClear() очищает фон .В дальнейшем , всякий раз , когда glClear () будет вызываться , она будет очищать окно в черный цвет . glColor3f() устанавливает цвет прорисовки - белый цвет . glOrtho() определяет координатную систему . glBegin() и glEnd() определяют обьект , который будет прорисован . glVertex3f() определяет вершины полигона , в качестве параметров - 3 координаты x,y,z . glFlush() гарантирует нам , что прорисовка полигона будет выполнена немедленно .

Все команды OpenGL начинаются с префикса gl , все константы также начинаются с префикса GL_ . Например , в команде glColor3f() цифра 3 говорит о том , что координат 3, а префикс 'f' говорит о том , что аргумент имеет тип floating-point .

Вообще , в OpenGL имеется 8 сновных типов Data Type :

1. 8-bit integer
2. 16-bit integer
3. 32-bit integer
4. 32-bit floating-point
5. 64-bit floating-point
6. 8-bit unsigned integer
7. 16-bit unsigned integer
8. 32-bit unsigned integer

Так , 2 команды glVertex2i(1, 3) и glVertex2f(1.0, 3.0) фактически эквивалентны , хоть и имеют различные типы. Некоторые команды имееют последним символом 'v' , что указывает на вектор .

Например :

```
glColor3f(1.0, 0.0, 0.0);
```

GLfloat color_array[] = {1.0, 0.0, 0.0};
glColor3fv(color_array);

Здесь определяем массив и даем векторный указатель на него . Доступ к этому массиву определяется с помощью часто употребляемой команды GLvoid() . Как только мы установили цвет с помощью glColor3f() , все последующие обьекты будут выводиться именно установленным цветом , пока мы его не поменяем вновь .

К числу базовых понятий относятся current viewing , projection transformations, line , polygon stipple patterns, polygon drawing modes, pixel-packing conventions, positions и characteristics lights, material properties .

Доступ к переменным можно осуществить с помощью glEnable() или glDisable(). Функции glGetBooleanv(), glGetDoublev(), glGetFloatv(), glGetIntegerv(), glGetPointerv(), glIsEnabled() - устанавливают тип данных . glPushAttrib() и glPushClientAttrib() пишут в стек , glPopAttrib() и glPopClientAttrib() восстанавливают .

OpenGL очень четко организован в смысле очередности выполнения операций .

Как правило , все OpenGL-исходники имеют в заголовке 2 хидера :

#include <GL/gl.h>
#include <GL/glu.h>

При использовании GLUT также указывается <GL/glut.h> . В библиотеку GLUT встроен интерфейс взаимодействия с операционной системой на уровне API . GLUT рекомендован на начальном этапе изучения OpenGL .

Window Management :

- glutInit(int *argc, char **argv) - самая первая команда инициализации
- glutInitDisplayMode(unsigned int mode) - устанавливает цветовую модель - RGBA или color-index .
  Например , glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH) - устанавливает двойной буфер , цветовую модель RGB .
- glutInitWindowPosition(int x, int y) - установим начало координат окна
- glutInitWindowSize(int width, int size) - размер окна
- int glutCreateWindow(char *string) - создает окно
- glutDisplayFunc() - вызывается всякий раз при перерисовке окна .
- glutMainLoop(void) - эта функция вызывается после всех остальных .

В GLUT встроена прорисовка 3-мерных обьектов :

cone	icosahedron	teapot
Cube	octahedron	tetrahedron
Dodecahedron	sphere	torus

Следующий пример показывает работу команды glutSwapBuffers() , делающий свопинг буффера .Имеется 2 буффера , и пока на экран не выводится полностью один из них , второй остается полностью за кадром , и не произойдет наложения одного на другой .

Переход в 3D

На необходимо подключить, для компиляции данного проекта 2 библиотеки - Glaux.lib и Opengl32.lib. Добраться до установки библиотек можно с помощь Project Setting - Link - Object library module.

В этих библиотеках не код естественно, а ссылки на DLL.

В этом варианте мы получим отдельное окно OpenGL. На нем и будем тренироваться.

```
#include "stdafx.h"
```

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#pragma comment(lib, "opengl32.lib")
#pragma comment(lib, "glu32.lib")
#pragma comment(lib, "glaux.lib")

int main(int argc, char** argv)
{
    auxInitDisplayMode(AUX_RGB);
    auxInitPosition (0, 0, 500, 500);
    auxInitWindow ("Step1");
    return 0;
}
```

OpenGL является вероятно самым распространеным индустриальным программным интерфейсом API для разработки 3D-приложений. Он представляет собой открытый стандарт, созданный специалистами SGI и находящийся в ведении специального комитета Architecture Review Board, в работе которого участвуют и SGI и Microsoft.

Через #include необходимые заголовки с описанием функций.

auxInitDisplayMode Создает окно с переданными атрибутами. У нас это окно будет в палитре RGB, о чем и говорит параметр.

auxInitPosition Установка разрешения окна или максимальных координат или позиции, смотря как вы это воспринимаете. Любое из высказываний вроде верно.

auxInitWindow Открывает окно исходя и параметров установленных предыдущими командами и с именем в параметрах.

Если вы запустите и выполните программу, то появиться и изчезнет окно, скорость в зависимости от скорости вашего ПК.

Добавляем строку

```
void CALLBACK display(void);      auxMainLoop(display);
```

и

```
void CALLBACK display(void)
{
      glBegin(GL_LINES);
      glVertex2f (0,0);
      glVertex2f (100,100);
      glEnd();
      glFlush();
}
```

У нас добавилась функция auxMainLoop. Эта функция вызывает другую функцию для прокрутки команд OpenGL. Эта функция должна пользоваться передачей параметров типа FAR PASCAL. Передача параметров происходит через стек, а Pascal и C, передают параметры в разной последовательности. Так вот эта процедура должна передавать параметры как PASCAL.

void CALLBACK display(void); == void FAR PASCAL display(void); == void __stdcall display(void); == void WINAPI display(void); . Данный тип вызовов установлен для всех типов API.

Любая последовательность команд по рисованию , это рисовка по вершинам. Вершины определяются от glBegin(тип) до glEnd(). Между этими командам устанавливаются вершины функцией glVertex2f(x,y) для 2D точек.

glFlush() Прорисовывает экран. Без этой команды вы нечего не увидите.

Создайте функцию для рисования:

void CALLBACK display(void);

Вызовите её:

auxMainLoop(display);

Начинайте рисовать и заканчивайте так:

glBegin(GL_LINES);

//.....

glEnd();

Прорисуйте изображение:

glFlush();

Изменения функции display.

Нам необходимо подключить еще одну библиотеку - Glu32.lib

Изменяем строку на auxInitDisplayMode (AUX_SINGLE | AUX_RGB); и void WINAPI display(void);

Меняем операции в функции `display`

```
glColor3d(0,1,1);
      glBegin(GL_LINES);
            glVertex3f (0,0,0);
            glVertex3f (0,0,100);
      glEnd();
      glBegin(GL_LINES);
            glVertex3f (0,0,0);
            glVertex3f (0,100,0);
      glEnd();
      glBegin(GL_LINES);
            glVertex3f (0,0,0);
            glVertex3f (100,0,0);
      glEnd();
      glFlush();
```

Суть заключается в том, чтобы нарисовать оси координат. Вначале мы устанавливаем цвет glColor3f в палитре RGB устанавливает цвет рисуемых вершин.

Далее мы командами glBegin - glEnd создаем 3 линии. При создании линии, мы используем команду для установки вершин glVertex3f, которая имеет три координаты X,Y,Z.

Когда вы запустите программу, то увидите просто голубой угол. Одна из координат выродилась в точку.

Теперь создадим в нашей программа три разноцветные оси координат

```
#include "stdafx.h"
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#pragma comment(lib, "opengl32.lib")
#pragma comment(lib, "glu32.lib")
#pragma comment(lib, "glaux.lib")

void myinit(void);
void CALLBACK display(void);

void myinit (void)
{
}

void CALLBACK display (void)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective (130, 1, 50, 0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(150, 150, 150, 0, 0, 0, 0, 100, 0);
	glColor3f(1,0,0);
	glBegin(GL_LINE_LOOP);
		glColor3f(1,0,0);
		glVertex3f(0, 0, 0);
		glVertex3f(200, 0, 0);
	glEnd();
	glBegin(GL_LINE_LOOP);
		glColor3f(0,1,0);
		glVertex3f(0, 0, 0);
		glVertex3f(0, 200, 0);
	glEnd();
	glBegin(GL_LINE_LOOP);
		glColor3f(0,0,1);
		glVertex3f(0, 0, 0);
		glVertex3f(0, 0, 200);
	glEnd();
	glFlush();
}

int main(int argc, char** argv)
{
	auxInitDisplayMode (AUX_SINGLE | AUX_RGB);
	auxInitPosition (0, 0, 500, 500);
	auxInitWindow ("1-6");
	myinit();
	auxMainLoop(display);
	return(0);
}
```

Итак, что добавилось glMatrixMode (GL_PROJECTION); говорит о том, что команды относятся к проекту.

glLoadIdentity(); считывает текущую матрицу.

gluPerspective (130, 1, 50, 0); Настройка перспективы просмотра. Нам сейчас интересны первые два параметра. Первый параметр это охват в градусах от 0 до 180. Можете воспринимать его как обьектив на фотоаппарате. Либо все но мелкое , либо большое но одно. Создав и запустив проект, поменяейте это параметр, вы увидите изменение изображения больше-меньше. Посмотрите на рисунок ниже. Второй параметр это угол поворота по оси Y. Да бог с ним. Главное первый параметр.

glMatrixMode (GL_MODELVIEW); говорит о том, что работы будет теперь просмотром, а не проектом. Это важно. Дело в том , что проект и просмотр имеют разницу. Зачастую необходимо поворачивать фигуры друг относительно друга и т.п. это делается в разных матрицах и т.д. Ну это на примерах понятнее будет надеюсь. А пока факт. Перспективу для проекта, а взгляд для просмотра.

gluLookAt (150, 150, 150, 0, 0, 0, 0, 100, 0); Устанавливает точку наблюдения, камеру. Первые параметры откуда (x,y,z) и куда (x,y,z). Это пока главное.

Поэкспениментируйте с первыми тремя или вторыми тремя параметрами. Угол взгляда будет меняться. Лучше менять параметры на 2 - 5 единиц и у одной координаты, чтобы не потеряться.

Шпаргалка

Установите перспективу:

```
glMatrixMode(GL_PROJECTION);

glLoadIdentity();

gluPerspective(130, 1, 50, 0);
```

Угол взгляда:

```
glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

gluLookAt(150, 150, 150, 0, 0, 0, 0, 100, 0);
```