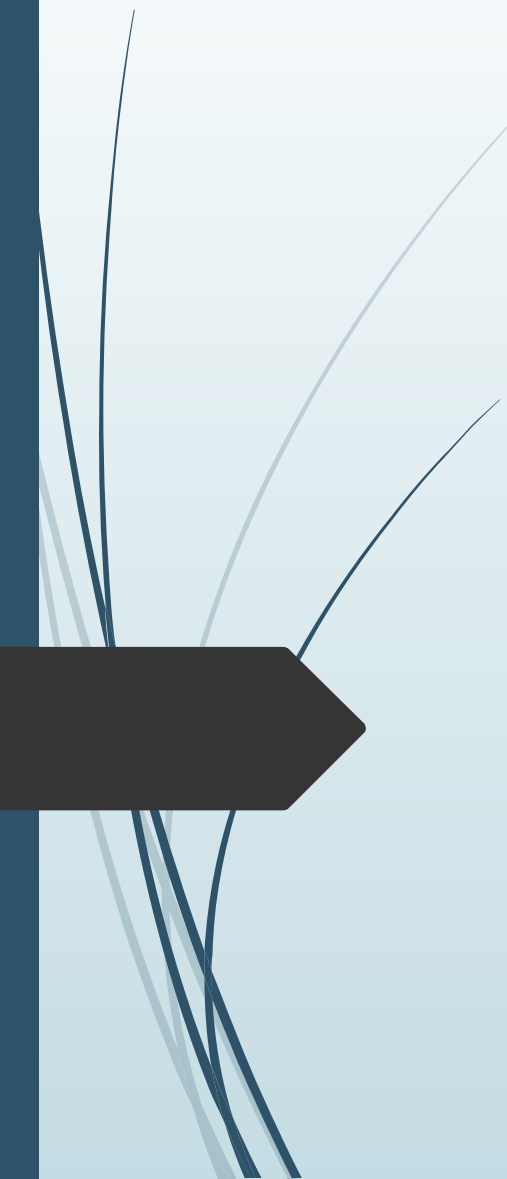

Знакомство с OpenGL

Графические функции

- В графических пакетах общего назначения пользователям предлагают ряд функций для создания рисунков и выполнения действий над ними.
- Эти функции можно классифицировать по тому, имеют ли они дело с графическим выходом, входом, атрибутами, преобразованиями, визуализацией, делением изображений или общим контролем.
- Основные блоки, из которых составляются изображения, называются ***графическими выходными примитивами***. К ним относятся символьные строки и геометрические объекты, такие как точки, прямые линии, закрашенные цветные участки (как правило многоугольники) и формы, которые задаются массивами цветных точек. В некоторых графических пакетах предлагаются функции для изображения более сложных форм – окружности, цилиндры, конусы. Функции для создания результирующих примитивов представляют основные средства для создания изображений.
- Атрибуты – свойства результирующих примитивов. То есть атрибут описывает, как следует изображать отдельный примитив. Сюда относится описание цвета, типа линии, шрифт текста и узоры заполнения отдельных участков.

Графические функции

- Размер, положение и ориентацию объекта на сцене можно менять с помощью ***геометрических преобразований***. В некоторых графических пакетах есть дополнительный набор функций для преобразований моделирования, которые используют для построения сцены, где описания отдельный объектов задаются в локальных координатах. В таких пакетах предлагается механизм описания сложных объектов (таких как электрический контур или велосипед) с помощью иерархической структуры (дерева). В других пакетах просто предоставляются функции геометрических преобразований, подробности моделирования предоставляются программисту.
- После того, как сцена составлена, с помощью функций для описания формы объектов и их атрибутов графический пакет создает проекцию изображения на устройстве вывода. ***Преобразования наблюдения*** помогают выбрать точку наблюдения на экране, вид проекции, которую следует использовать в данном случае, и место на мониторе, где будет располагаться изображение. Существуют и другие стандартные функции для управления областью изображения на экране путем задания его координат, размера и структуры. Для 3D сцен определяются видимые объекты и налагаются соответствующие условия освещения.

Графические функции

- В интерактивных графических приложениях используются различные типы входных устройств, в т.ч. Мышь, планшет или джойстик. Для проверки обработки потока данных, поступающего от этих интерактивных устройств, используют функции ввода.
- Некоторые графические пакеты предлагают также функции для деления описания изображения на именованные наборы составляющих частей.
- Наконец, в графических пакетах содержится ряд служебных задач, таких как закрашивание экрана монитора заданным цветом и инициализации параметров. Функции, выполняющие эту работу, можно объединить по названием операции управления.

Стандарты программного обеспечения

- Цель, преследуемая в стандартизированных графических программах - универсальность. Когда разрабатываются пакеты со стандартными графическими функциями, такие программы легко переносятся с одной машины на другую и могут использоваться в различных реализациях и приложениях.
- Международные и национальные организации по составлению стандартов во многих странах объединили свои усилия с целью разработки общепринятого стандарта КГ. В результате в 1984 была создана ***базовая графическая система*** (Graphical Kernel System - GKS). Эта системы принята в качестве первого стандарта графического программного обеспечения Международной организацией по стандартизации (International Standards Organization - ISO).
- Вторым разработанным и принятым стандартом программного обеспечения был PHIGS ()Programmer`s Hierarchical Interactive Graphics Standard – иерархическая интерактивная графическая система программиста. Этот стандарт отличался более широким диапазоном возможностей иерархического моделирования объектов, задания цветов, закрашивания поверхностей и выполнения действий над изображением.

Стандарты программного обеспечения

- Далее появилось продолжение PHIGS – PHIGS+, в котором предоставлялись возможности трехмерного закрашивания поверхностей, которых не было в PHIGS.
- Во времена разработки GKS и PHIGS очень популярными стали рабочие станции производства Silicon Graphics, Inc., SGI. Эти рабочие станции выпускались вместе с набором стандартных функций GL (Graphics Library), который скоро стал довольно популярным в кругах, где дело имелось с КГ. Функции GL разрабатывались для быстрого закрашивания в реальном времени, и вскоре этот программный пакет распространился и на другие программные средства. В итоге, в начале 1990х был разработан пакет OpenGL как аппаратно-независимая версия пакета GL. 11 августа 2014 года была опубликовала спецификации OpenGL 4.5. Разработчик – компания Silicon Graphics(SGI), которая в 1992 возглавила *OpenGL ARB* — группу компаний по разработке спецификации OpenGL.
- Библиотека OpenGL разработана специально для эффективной обработки трехмерных данных, но может работать и с описанием 2D сцен как с частным случаем 3D изображения.

Стандарты программного обеспечения

- Графические функции в любом пакете задаются как набор описаний, которые не зависят от какого либо ЯП. Затем задается привязка к языку для определенного ЯП высокого уровня. Эта привязка задает синтаксис, который позволяет пользоваться различными графическими функциями этого языка. Привязка к ЯП задается так, чтобы максимально использовать соответствующие возможности языка и управлять моментами, связанными с синтаксисом, такими как типа данных, задание параметров и обработка ошибок. Спецификация для реализации графического пакета в определенном языке устанавливается ISO.

Знакомство с OpenGL

- Основная библиотека функций в пакете OpenGL предлагается для спецификации графических примитивов, атрибутов, геометрических преобразований, преобразований наблюдения и многих других преобразований.
- OpenGL разрабатывался, чтобы быть независимым от аппаратных средств, поэтому многие операции, такие как функции ввода и вывода, не входят в число функций основной библиотеки. Но, стандартные функции ввода-вывода, а также множество дополнительных функций, доступны из вспомогательных библиотек, разработанных для OpenGL.

Основной синтаксис OpenGL

- Именами функций основной библиотеки OpenGL ставят префикс **gl**, а каждое слово, которое входит в имя функции, начинается с заглавной буквы(glBegin, glClear, glCopyPixels,glPolidonMode).
- Определенные функции требуют, чтобы одному или нескольким из ее аргументов присваивалось значение символьной константы, обозначающей, например, имя параметра, значение параметра или определённого режима. Все такие константы начинаются с заглавных букв **GL**. Кроме того, слова, составляющие имя этой константы пишутся заглавными, а в качестве разделителя между словами, составляющими одно имя, используют нижнее подчеркивание(GL_2D, GL_RGB, GL_CCW,GL_POLYGON,GL_AMBIENT_AND_DIFFUSE).
- Функции пакета OpenGL также воспринимают особые типы данных. Например, например параметр функции OpenGL может воспринимать значения, которые задаются как 32-разрядные целые числа. Но размер спецификации целого числа на разных машинах может отливаться. Для обозначения особого типа данных в пакете OpenGL используются специальные встроенные названия типов данных:
- GLbyte GLshort Glint GLfloat GLdouble GLboolean

Родственные библиотеки

- Кроме основной библиотеки OpenGL, существует еще ряд связанных с ней библиотек для выполнения специальных операций. Набор программ OpenGL (OpenGL Utility – GLU) предоставляет стандартные функции, позволяющее настраивать матрицы проекций и визуализации, описывать сложные объекты через приближения прямых и прямоугольников, изображать квадратичные и би – сплайны с помощью линейного приближения, закрашивать поверхности и др. Каждая реализация пакета OpenGL включает библиотеку GLU, а все имена функций GLU начинаются с префикса glu.
- Чтобы создать графическое изображение с помощью пакета OpenGL, прежде всего необходимо открыть на рабочем столе окно изображения – прямоугольная область экрана, на которой будет строится наше изображение.
- Окно изображения нельзя создать непосредственно с помощью основных функций пакета OpenGL, т.к. эта библиотека содержит только независящие от прибора графические функции, а операции управления зависят от компьютера, на котором мы работаем. Однако существует несколько библиотек систем окон, которые поддерживают функции для различных машин.

Родственные библиотеки

- Расширение OpenGL для системы X-Windows (GLX) предлагает набор стандартных функций, которые начинаются с префикса glX.
- Расширение OpenGL для системы Apple (AGL) предлагает набор стандартных функций, которые начинаются с префикса agl.
- Для системы Microsoft Windows стандартные функции WGL предлагает интерфейсс Windows-to-OpenGL. Эти функции начинаются с префикса wgl.
- OpenGL Utility Toolkit (GLUT) предлагает библиотеку функций для работы с любой системой окон на экране. Библиотеке функций GLUT соответствует префикс glut. Кроме того, в этой библиотеке содержатся метолы, позволяющие описывать и закрашивать кривые и поверхности второго порядка.
- Та как GLUT – интерфейс для других систем окон со специальными устройствами, мы можем воспользоваться программой GLUT и сделать так, чтобы наши программы не зависели от приборов.

Файлы заголовков

- Во всех наших графических программах должен быть файл заголовка для корневой библиотеки OpenGL. Для большинства приложений нужна еще и библиотека GLU. И мы должны включать файл заголовка для системы окон. Например, для системы Microsoft Windows файл заголовка для доступа к стандартным функциям WGL – windows.h. Этот файл заголовка должен находится перед файлами заголовков для библиотек OpenGL и GLU, т.к. в нем записан макрос, необходимый для библиотек OpenGL версии Microsoft Windows.
- Исходный файл может начинаться так:

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

- Однако, если для выполнения операций управления окном пользоваться библиотекой GLUT, то не нужно включать заголовочные файла gl.h и glu.h, т.к. в библиотеке GLUT уже подразумевается, что они будут учтены правильным способом. Поэтому можно заменить файлы заголовков библиотек OpenGL b GLU на

```
#include <GL/glut.h>
```

Управление окнами изображений с помощью библиотеки GLUT

- Т.к. мы будем пользоваться библиотекой GLUT, нашим первым шагом будет инициализация GLUT. Эта функция инициализации может обрабатывать любые аргументы в командной строке. Инициализация GLUT осуществляется с помощью команды:

  glutInit(&argc, argv);

- Затем на экране необходимо создать окно изображения с соответствующим названием в строке заголовка:

  glutCreateWindow("Пример рисования линии");

- Здесь единственный аргумент – строковая константа, которая содержит название нашего окна изображения.
- Затем необходимо задать, что конкретно будет содержаться в нашем окне изображения. Для этого с помощью функции OpenGL создается изображение и передается определение GLUT **glutDisplayFunc**, которая связывает наш рисунок с окном изображения.

Управление окнами изображений с помощью библиотеки GLUT

- В качестве простого примера предположим, что у нас есть код OpenGl для описания отрезка прямой в процедуре с названием my_line. Затем с помощью следующей функции описание линейного сегмента передается в окно изображения:

glutDisplayFunc(my_line);

- Но окна изображения все еще нет на экране. Чтобы завершить операции по обработке окна, нам нужна еще одна функция из библиотеки GLUT. После выполнения следующего оператора активизируются все окна изображений, которые были созданы вместе со своим графическим содержанием:

glutMainLoop();

- Эта функция должна размещаться в крайней точке нашей головной функции. Она служит для изображения начальных графических элементов и вводит программу в бесконечный цикл, в котором проверяются входные данные, поступающие от таких устройств, как мышь и клавиатура. Наш первый пример не будет интерактивным, поэтому программа будет изображать нарисованную нами линию до тех пор, пока мы не закроем окно изображения.
-

Управление окнами изображений с помощью библиотеки GLUT

- Также мы можем задать размер и местоположение нашего окна изображения, несмотря на то, что эти параметры заданы по умолчанию. С помощью функции glutInitWindowPosition из библиотеки GLUT мы можем задать исходное положение левого верхнего угла окна изображений. Это положение определяется через целые значения координат экрана. Начало отсчета находится в верхнем левом угле экрана. Например,

glutInitWindowPosition(0, 100);

прижмет наше окно изображения к левому краю экрана и расположит его на 100 пиксель ниже верхней границы экрана.

- Аналогично функция glutInitWindowSize используется, чтобы задать начальную ширину и высоту окна изображения в пикселях.
- После того, как окно изображений появится на экране, мы имеем возможность поменять его начальное местоположение, а также размер.

Управление окнами изображений с помощью библиотеки GLUT

- Также мы имеем возможность задать ряд других опций окна изображений, таких как буферизация и выбор цветового режима, использовав функцию glutInitDisplayMode. Аргументами этой функции являются символьные константы библиотеки GLUT.

Константа	Значение
GLUT_RGB	Для отображения графической информации используются 3 компоненты цвета RGB.
GLUT_RGBA	То же что и RGB, но используется также 4 компонента ALPHA (прозрачность).
GLUT_INDEX	Цвет задается не с помощью RGB компонентов, а с помощью палитры. Используется для старых дисплеев, где количество цветов например 256.
GLUT_SINGLE	Вывод в окно осуществляется с использованием 1 буфера. Обычно используется для статического вывода информации.

Управление окнами изображений с помощью библиотеки GLUT

Константа	Значение
GLUT_DOUBLE	Вывод в окно осуществляется с использованием 2 буферов. Применяется для анимации, чтобы исключить эффект мерцания.
GLUT_ACCUM	Использовать также буфер накопления (Accumulation Buffer). Этот буфер применяется для создания специальных эффектов, например отражения и тени.
GLUT_ALPHA	Использовать буфер ALPHA. Этот буфер, как уже говорилось используется для задания 4-го компонента цвета - ALPHA. Обычно применяется для таких эффектов как прозрачность объектов и антиалиасинг.

Управление окнами изображений с помощью библиотеки GLUT

Константа	Значение
GLUT_DEPTH	Создать буфер глубины. Этот буфер используется для отсечения невидимых линий в 3D пространстве при выводе на плоский экран монитора.
GLUT_STENCIL	Буфер трафарета используется для таких эффектов как вырезание части фигуры, делая этот кусок прозрачным. Например, наложив прямоугольный трафарет на стену дома, вы получите окно, через которое можно увидеть что находится внутри дома.
GLUT_STEREO	Этот флаг используется для создания стереоизображений. Используется редко, так как для просмотра такого изображения н

Управление окнами изображений с помощью библиотеки GLUT

- Например, с помощью следующей функции задается, что для окна изображения будет использоваться один буфер регенирации, а для выбора цветовых значений – цветовой режим RGB (красный, зеленый, синий):

```
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

- Значения постоянных, которые передаются этой функции, объединяются с помощью логической операции ИЛИ. На самом деле, эту функцию можно было не прописывать в нашем примере, потому что указанные параметры задаются по умолчанию.

Управление окнами изображений с помощью библиотеки GLUT

- Итого, текст наше головной функции имеет вид:

```
void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(0, 100);
    glutInitWindowSize(400, 300);
    glutCreateWindow("Пример рисования линии");

    init();
    glutDisplayFunc(my_line);
    glutMainLoop();
}
```

Полная программа OpenGL

- Но у нас еще не все подготовлено и реализовано.
- Нам необходимо выбрать цвет фона для окна изображения, а также создать процедуру, в которой содержаться функции, подходящие для изображения линии, которую мы задумали изобразить.
- С помощью цветовых значений RGB устанавливаем через функцию OpenGL белый цвет фона окна изображений

glClearColor(1.0, 1.0, 1.0, 0.0);

Описание функции:

void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);

red, green, blue, alpha : значения, используемые для очистки буфера цвета. Значения по умолчанию 0.

glClearColor устанавливает значение RGBA компонент цвета для последующего их использования функцией glClear.

Значения должны быть в рамках [0;1].

Если ты мы задали первые 3 параметра равными 0, по получили бы черный фон.

Полная программа OpenGL

- Четвертый параметр называется альфа-фактором для заданного цвета. Одно из назначений этого параметра – «смешивание». Когда активизируются операции смешивания пакета OpenGL, значение альфа-фактора может использоваться для определения результирующего цвета двух перекрывающихся объектов. Значение альфа-фактора 0 говорит о том, что объект будет полностью прозрачным, а значение 1 указывает на полностью непрозрачный объект.
- Хотя команда glClearColor присваивает цвет окну изображения, она не перемещает его на экран. Чтобы увидеть окно изображений на экране, нужно вызвать следующую функцию OpenGl:

```
glClear(GL_COLOR_BUFFER_BIT);
```

- Эта функция фактически заполняет соответствующие буферы (в нашем случае окно изображения) значением, выбранным в функциях glClearColor, glClearIndex, glClearDepth, glClearStencil и glClearAccum .

Полная программа OpenGL

- Параметрами функции могут быть:
- GL_COLOR_BUFFER_BIT Очищает текущий буфер цвета, выбранный для записи.
- GL_DEPTH_BUFFER_BIT Очищает буфер глубины.
- GL_ACCUM_BUFFER_BIT Очищает аккумулирующий буфер.
- GL_STENCIL_BUFFER_BIT Очищает буфер трафарета.
- Если буфер для очистки не представлен, то glClear не вызывает никакого эффекта.

- В нашем решении мы запишем

```
glClear(GL_COLOR_BUFFER_BIT);
```

Что означает, что в буфере цвета (буфере регенерации) находятся значения битов, которые следует присвоить переменным, указанным в функции glClearColor .

Полная программа OpenGL

- Кроме определения цвета фона для она изображения, можно выбирать различные цветовые схемы для объектов, которые мы хотим изобразить на экране. Для нашей первой программы мы просто зададим цвет объекта (линии) красный с помощью функции

glColor3f(1.0, 0.0, 0.0);

- Эта функция задания текущего цвета вершины. Параметры соответствуют красному, зеленому и синему цвету. Уровень прозрачности в этой функции 1, что означает, что цвет полностью непрозрачен. F указывает на то, что значения всех 3 параметров варьируются в пределах от 0 до 1.
- Далее нам необходимо сообщить программе OpenGL, как мы хотим спроектировать нашу линию на окно изображения, поскольку создание двухмерного изображения рассматривается OpenGL как частный случай трёхмерного. OpenGL будет обрабатывать нашу линию с помощью операции полной трехмерной визуализации. Мы можем задать вид проекции (режим) и другие параметры визуализации, которые нам нужны, с помощью следующих двух функций:

glMatrixMode(GL_PROJECTION);

gluOrtho2D(0.0, 200.0, 0.0, 150.0);

Полная программа OpenGL

- Рассмотрим подробнее эти функции.
- Для задания различных преобразований объектов сцены в OpenGL используются операции над матрицами, при этом различают три типа матриц: видовая, проекций и текстуры. Все они имеют размер 4x4. Видовая матрица определяет преобразования объекта в мировых координатах, такие как параллельный перенос, изменение масштаба и поворот. Матрица проекций задает как будут проецироваться трехмерные объекты на плоскость экрана (в оконные координаты), а матрица текстуры определяет наложение текстуры на объект.
- Для того, чтобы выбрать, какую матрицу надо изменить, используется команда

  void glMatrixMode(GLenum mode)

- вызов которой со значением параметра mode равным GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE включает режим работы с видовой, проекций и матрицей текстуры соответственно. Для вызова команд, задающих матрицы того или иного типа необходимо сначала установить соответствующий режим.

Полная программа OpenGL

- В OpenGL существуют ортографическая (параллельная) и перспективная проекция. Первый тип проекции может быть задан командами

void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)

void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)

- Первая команда создает матрицу проекции в усеченный объем видимости (параллелограмм видимости) в левосторонней системе координат. Параметры команды задают точки (left, bottom, -near) и (right, top, -near), которые отвечают левому нижнему и правому верхнему углам окна вывода. Параметры near и far задают расстояние до ближней и дальней плоскостей отсечения по дальности от точки (0,0,0) и могут быть отрицательными.
- Во второй команде, в отличие от первой, значения near и far устанавливаются равными –1 и 1 соответственно.

Полная программа OpenGL

- В свете вышесказанного, наши команды

glMatrixMode(GL_PROJECTION);

gluOrtho2D(0.0, 200.0, 0.0, 150.0);

- Означают, что для отображения содержимого 2D прямоугольной области со внешними координатами на экран следует использовать ортогональную проекцию, и что значения координаты x этого прямоугольника должны лежать в диапазоне от 0 до 200, а значения y – в диапазоне от 0 до 150. Какие бы объекты мы не задавали в пределах этого прямоугольника внешних координат, они будут попадать в окно изображения. Ничего, что находится вне описанного диапазона отображаться не будет.
- Таким образом, с помощью функции *gluOrtho2D* задается, что система координат окна изображения будет такой: точка с координатами (0,0) находится в нижнем левом углу окна изображений, а точка с координатами (200,150) – в верхнем правом углу. Т.к. мы описываем 2D объект, действие ортогональной проекции будет ограничиваться вставкой нашего рисунка в окно изображения.

Полная программа OpenGL

- И, наконец, нам нужно вызвать соответствующие стандартные функции OpenGL, чтобы создать линию.

```
glBegin(GL_LINES);
    glVertex2i(0, 0);
    glVertex2i(50, 145);
glEnd();
```

- Этот код задает двумерный прямолинейный отрезок с целочисленными декартовыми координатами концов (0,0) и (50,145).

- Давайте подробнее разберемся, что здесь написано.
- Чтобы задать какую-нибудь фигуру одних координат вершин недостаточно, и эти вершины надо объединить в одно целое, определив необходимые свойства. Для этого в OpenGL используется понятие примитивов, к которым относятся точки, линии, связанные или замкнутые линии, треугольники и так далее. Задание примитива происходит внутри командных скобок:

```
void glBegin(GLenum mode)
void glEnd(void)
```

Полная программа OpenGL

- Параметр mode определяет тип примитива, который задается внутри и может принимать следующие значения:
- GL_POINTS каждая вершина задает координаты некоторой точки.
- GL_LINES каждая отдельная пара вершин определяет отрезок; если задано нечетное число вершин, то последняя вершина игнорируется.
- GL_LINE_STRIP каждая следующая вершина задает отрезок вместе с предыдущей.
- GL_LINE_LOOP отличие от предыдущего примитива только в том, что последний отрезок определяется последней и первой вершиной, образуя замкнутую ломаную.
- GL_TRIANGLES каждая отдельная тройка вершин определяет треугольник; если задано не кратное трем число вершин, то последние вершины игнорируются.

Полная программа OpenGL

- GL_TRIANGLE_STRIP каждая следующая вершина задает треугольник вместе с двумя предыдущими.
- GL_TRIANGLE_FAN треугольники задаются первой и каждой следующей парой вершин (пары не пересекаются).
- GL_QUADS каждая отдельная четверка вершин определяет четырехугольник; если задано не кратное четырем число вершин, то последние вершины игнорируются.
- GL_QUAD_STRIP четырехугольник с номером n определяется вершинами с номерами 2n-1, 2n, 2n+2, 2n+1.
- GL_POLYGON последовательно задаются вершины выпуклого многоугольника.

Полная программа OpenGL

- Определение атрибутов вершины
- Под вершиной понимается точка в трехмерном пространстве, координаты которой можно задавать следующим образом:
- void glVertex[2 3 4][s i f d](type coords)
- void glVertex[2 3 4][s i f d]v(type *coords)
- Координаты точки задаются максимум четырьмя значениями: x, y, z, w, при этом можно указывать два (x,y) или три (x,y,z) значения, а для остальных переменных в этих случаях используются значения по умолчанию: z=0, w=1. Как уже было сказано выше, число в названии команды соответствует числу явно задаваемых значений, а последующий символ – их типу.
- Координатные оси расположены так, что точка (0,0) находится в левом нижнем углу экрана, ось x направлена влево, ось y- вверх, а ось z- из экрана. Это расположение осей мировой системы координат, в которой задаются координаты вершин объекта,

Полная программа OpenGL

- Теперь мы готовы оформить вспомогательные функции- процедуры.
- Все функции инициализации и присваивания соответствующих одноразовых параметров поместим в процедуру init:

```
void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
```

Полная программа OpenGL

- Геометрическое описание нашего рисунка поместим в процедуру

```
void my_line(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2i(0, 0);
    glVertex2i(50, 145);
    glEnd();
    glFlush();
}
```

- В конце этой процедуры находится функция glFlush(), которая ускоряет выполнения функций OpenGL, записанных в буферах, которые находятся в различных местах вычислительной системы.

Особенности создания приложения с OpenGL в среде MVS2015

- Создаем пустой консольный проект.
- Заходим в Свойства Проекта –> Компоновщик –> Ввод –> Дополнительные зависимости и прописываем следующие команды :
- opengl32.lib
  glu32.lib
  путь до\Glaux.lib
- В папку с проектом *имя проекта**имя проекта* необходимо добавить файлы glut.h, glut32.lib и glut32.dll.
- Непосредственно в исходнике прописать

```
// WinAPI
#include <Windows.h>
#include <tchar.h>
// OpenGL
#pragma comment(lib, "opengl32.lib")
// GLUT
#pragma comment(lib, "glut32.lib")
#include "glut.h"
```