Изучение растеризации, реализация и описание проекции перспективы, разработка скелетной анимации 2D моделей.

Цель работы:

Ознакомиться с технологиями компьютерной графики.

Изучить процесс растеризации с использование API OpenGL.

Реализовать проекцию перспективы при помощи матриц преобразования.

Разработка скелетной анимации для 2D моделей.

Введение

Сначала дадим понятия основным терминам, используемых в работе:  
рендеринг –

Растеризация –

Матрица преобразования –

Проекция перспективы –

Скелетная анимация –

Кость (rig) –

В ходе работы будет описана реализации проекции перспективы и скелетной анимации, основанные на матрице преобразования.

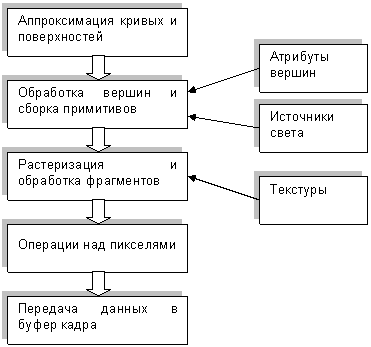
Работа реализована на языке C++.

# Введение в OpenGL

С точки зрения архитектуры графическая система OpenGL является конвейером, состоящим из нескольких последовательных этапов обработки графических данных.

Команды OpenGL всегда обрабатываются в том порядке, в котором они поступают, хотя могут происходить задержки перед тем, как проявится эффект от их выполнения. В большинстве случаев OpenGL предоставляет непосредственный интерфейс, т.е. определение объекта вызывает его визуализацию в буфере кадра.

С точки зрения разработчиков, OpenGL – это набор команд, которые управляют использованием графической аппаратуры. Если аппаратура состоит только из адресуемого буфера кадра, тогда OpenGL должен быть реализован полностью с использованием ресурсов центрального процессора. Обычно графическая аппаратура предоставляет различные уровни ускорения: от аппаратной реализации вывода линий и многоугольников до изощренных графических процессоров с поддержкой различных операций над геометрическими данными.

1. 

Инициализация окна состоит из настройки соответствующих буферов кадра, начального положения и размеров окна, а также заголовка окна.

Функция glutInit(&argc, argv) производит начальную инициализацию самой библиотеки GLUT.

Команда glutInitDisplayMode(GLUT\_RGB) инициализирует буфер кадра и настраивает полноцветный (непалитровый) режим RGB.

glutInitWindowSize(Width, Height) используется для задания начальных размеров окна.

Наконец, glutCreateWindow("Red square example") задает заголовок окна и визуализирует само окно на экране.

glClearColor(0, 0, 0, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

очищает окно и выводит на экран квадрат, задавая координаты четырех угловых вершин и цвет.

Как правило, задачей программы, использующей OpenGL, является обработка трехмерной сцены и интерактивное отображение в буфере кадра. Сцена состоит из набора трехмерных объектов, источников света и виртуальной камеры, определяющей текущее положение наблюдателя.

Обычно приложение OpenGL в бесконечном цикле вызывает функцию обновления изображения в окне. В этой функции и сосредоточены вызовы основных команд OpenGL. Если используется библиотека GLUT, то это будет функция с обратным вызовом, зарегистрированная с помощью вызова glutDisplayFunc(). GLUT вызывает эту функцию, когда операционная система информирует приложение о том, что содержимое окна необходимо перерисовать (например, если окно было перекрыто другим). Создаваемое изображение может быть как статичным, так и анимированным, т.е. зависеть от каких-либо параметров, изменяющихся со временем. В этом случае лучше вызывать функцию обновления самостоятельно. Например, с помощью команды glutPostRedisplay(). За более подробной информацией можно обратиться к приложению A.

Приступим, наконец, к тому, чем занимается типичная функция обновления изображения. Как правило, она состоит из трех шагов:

очистка буферов OpenGL;

установка положения наблюдателя;

преобразование и рисование геометрических объектов.

Очистка буферов производится с помощью команды:

|  |
| --- |
| void **glClearColor** (clampf r, clampf g, clampf b, clampf a)  void **glClear** (bitfield buf) |

КомандаglClearColor устанавливает цвет, которым будет заполнен буфер кадра. Первые три параметра команды задают R,G и B компоненты цвета и должны принадлежать отрезку [0,1]. Четвертый параметр задает так называемую альфа компоненту (см. п. 0). Как правило, он равен 1. По умолчанию цвет – черный (0,0,0,1).

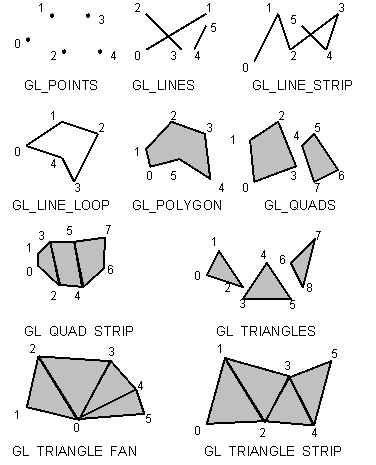
Команда glClear очищает буферы, а параметр *buf* определяет комбинацию констант, соответствующую буферам, которые нужно очистить (см. главу 6). Типичная программа вызывает команду

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

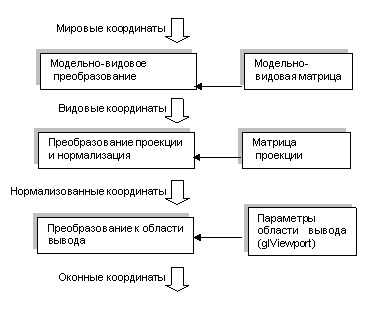
для очистки буферов цвета и глубины.

Установка положения наблюдателя и преобразования трехмерных объектов (поворот, сдвиг и т.д.) контролируются с помощью задания матриц преобразования. Преобразования объектов и настройка положения виртуальной камеры описаны в главе 3.

ак правило, разные типы примитивов имеют различную скорость визуализации на разных платформах. Для увеличения производительности предпочтительнее использовать примитивы, требующие меньшее количество информации для передачи на сервер, такие как **GL\_TRIANGLE\_STRIP**, **GL\_QUAD\_STRIP**, **GL\_TRIAGLE\_FAN.**

  
*Рис. 3. Примитивы OpenGL*

В целом, для отображения трехмерных объектов сцены в окно приложения используется последовательность, показанная на рисунке.

  
*Рис. 5. Преобразования координат в OpenGL*

|  |
| --- |
| **ПРИМЕЧАНИЕ**  Запомните: все преобразования объектов и камеры в OpenGL производятся с помощью умножения векторов координат на матрицы. Причем умножение происходит на *текущую матрицу* в момент определения координаты командой glVertex\* и некоторыми другими. |

# Структура классов

**Шаблонные математические классы**Matrix4f – матрица 4x4, произвольного типа. Интерфейс класса выполняет операции над матрицей преобразования для достижения проекции перспективы с указанными параметрами.

|  |
| --- |
| **template**<**typename** T>  **class** Matrix4f  {  **public**:  T m[4][4];  ...  } |

Vector3 – вектор из трёх элементов произвольного типа. Имеет базовый интерфейс необходимый для вектора:  
Арифметика с Vector3 и скалярами

Cross - возвращает вектор, который перпендикулярен к обоим исходным векторам

VDot - возвращает скалярное значение, которое является мерой проекции одного вектора на другой

Normalize - возвращает вектор той же направленности, но с длиной, равной единице

|  |
| --- |
| **template**<**typename** T>  **class** Vector3  {  **public**:  T x;  T y;  T z;  ...  } |

# Создание окна

Для создания окна использует простая высокоуровневая библиотека «OpenGL utility library» (GLUT). Она позволяет управлять окном приложения, обрабатывать события и контролирует ввод/вывод.