Графика OpenGL

Семестр 1

Семинар 9

OpenGL

- открытая графическая библиотека
- спецификация, то есть документ, описывающий набор функций и их точное поведение
- стабильность и надежность
- кроссплатформенность (Windows, Unix, Playstation 3, Mac OS)
- создание
  - компьютерных игр,
  - виртуальной релаьности,
  - систем автоматизированного проектирования,
  - визуализации в научных исследованиях
- не путать с DirectX (только Windows)
- SGI, 3Dlabs, Matrox, Evans & Sutherland, ATI, NVIDIA, IBM, Apple, Dell, Hewlett-Packard, Sun Microsystems, id Software

Возможности

- Геометрические (точки, линии, полигоны) и растровые (массив bitmap и образ image) примитивы.
- Видовые и модельные преобразования (расположение обьектов в пространстве, вращение, изменение формы, изменение положения камеры).
- Работа с цветом.
- Удаление невидимых линий и поверхностей. Z-буферизация.
- Двойная буферизация. Изображение каждого кадра сначала рисуется во втором(невидимом) буфере, а потом, когда кадр полностью нарисован, весь буфер отображается на экране.
- Наложение текстуры.
- Сглаживание (изменяет интенсивность и цвет пикселей около линии, при этом линия смотрится на экране без всяких зигзагов).
- Освещение (источники света, расположение, интенсивность).
- Атмосферные эффекты (туман, дым).
- Прозрачность объектов.
- Использование списков изображений.

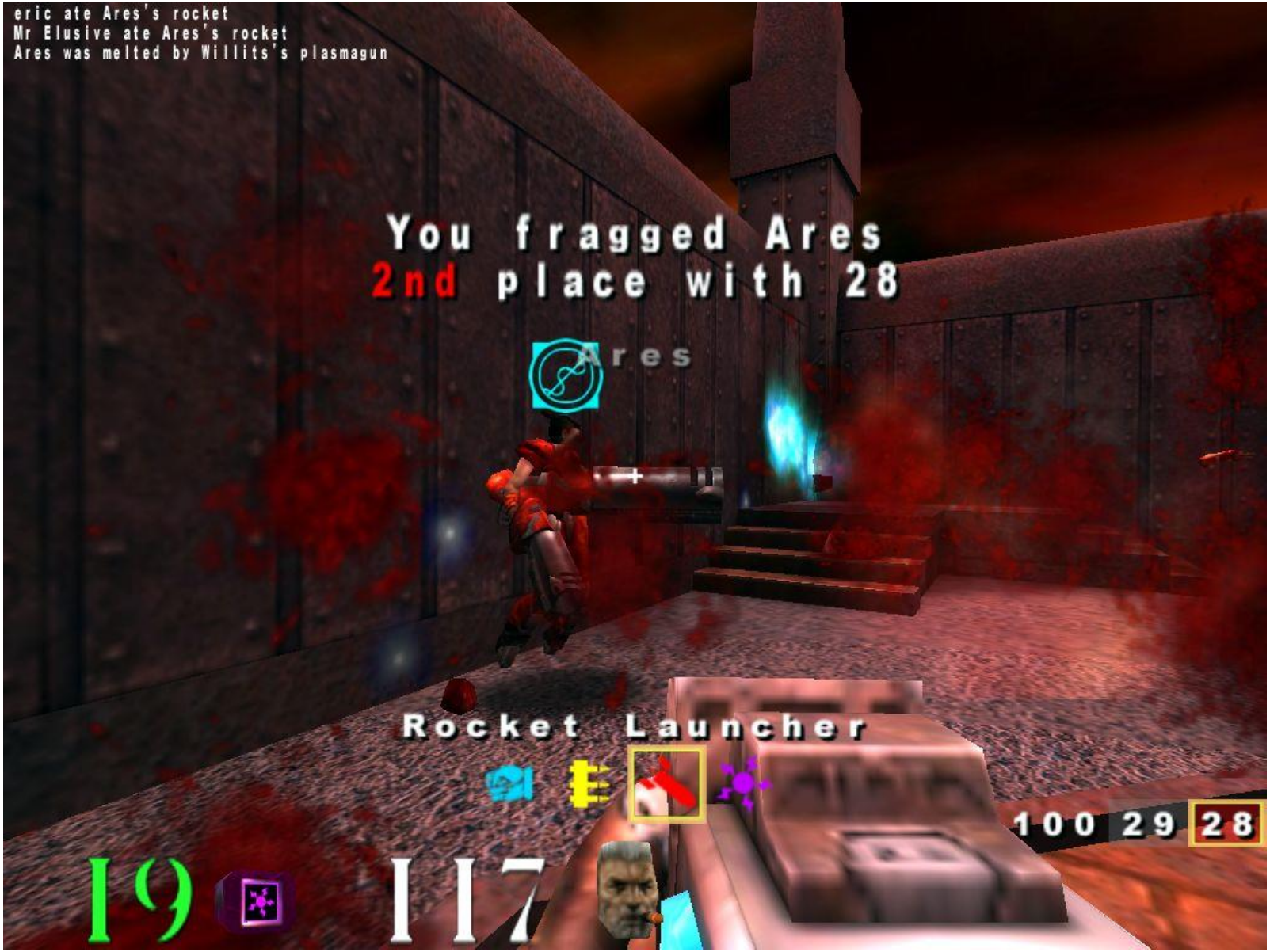
GLU

- OpenGL Utility
- «надстройка» над OpenGL
- Популярные геометрические примитивы – куб, шар, диск и т.д.
- Сплайны
- Дополнительные операции над матрицами

GLUT

- OpenGL Utility Toolkit
- «надстройка» над OpenGL
- полная кроссплатформенность
- библиотека, которая в основном отвечает за системный уровень операций ввода-вывода при работе с операционной системой
- не требует знаний API операционной системы

Doom 3



Quake III Arena

Minecraft

Tiny Wings

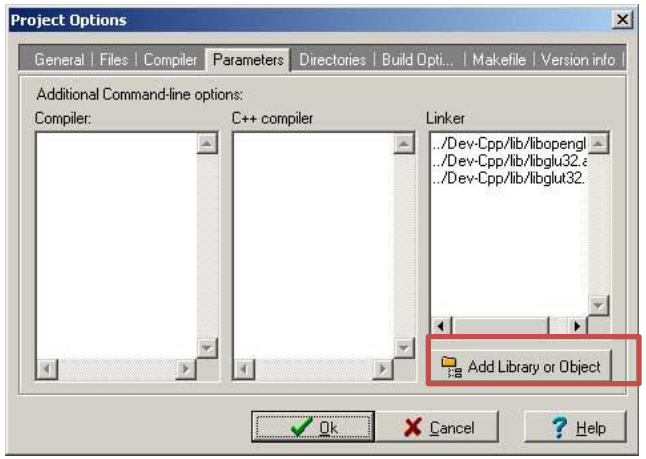
Установка (Windows, DevCpp)

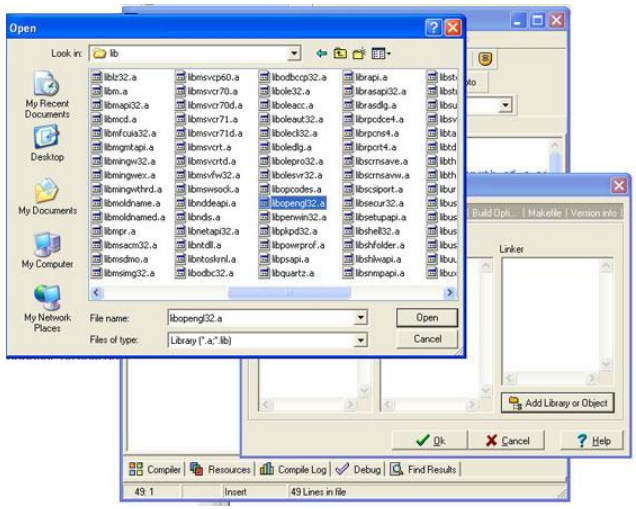
1. glut.h → D:\Dev-Cpp\include\GL
2. libglut32.a → D:\Dev-Cpp\lib
3. glut32.dll → C:\Windows\System32
4. Создать пустой проект
5. Добавить библиотеки линкеру в свойствах проекта:

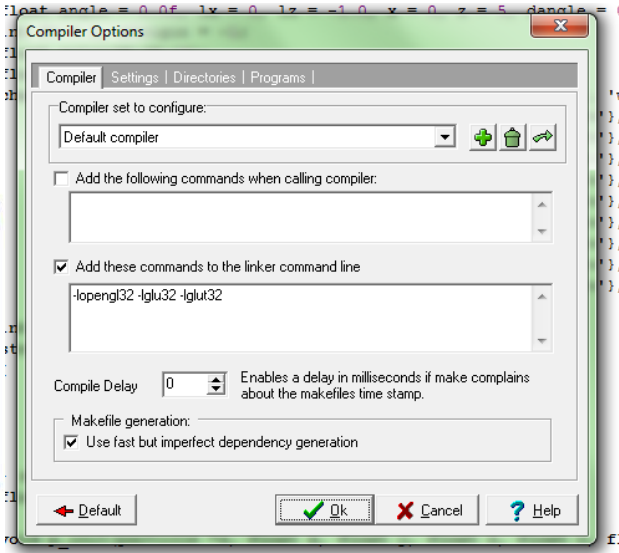
   /lib/libopengl32.a /lib/libglu32.a /lib/libglut32.a

4б. Добавить в свойства линкера.

   -lopengl32 -lglu32 -lglut32







Установка (Linux)

1. Установить пакеты freeglut и freeglut-dev (может называться freeglut3 или freeglut-devel в зависимости от дистрибутива).

   Скорее всего, консольные команды будут выглядеть так:

   sudo apt-get install freeglut3 freeglut3-devel (ubuntu)

   sudo yum install freeglut freeglut-dev (fedora)

   Или используйте установщик пакетов (synaptic для ubuntu, yum для fedora).

   sudo apt-get install synaptic (ubuntu)

   sudo yum install yum-installer (fedora)

2. Компилировать с добавлением библиотек:

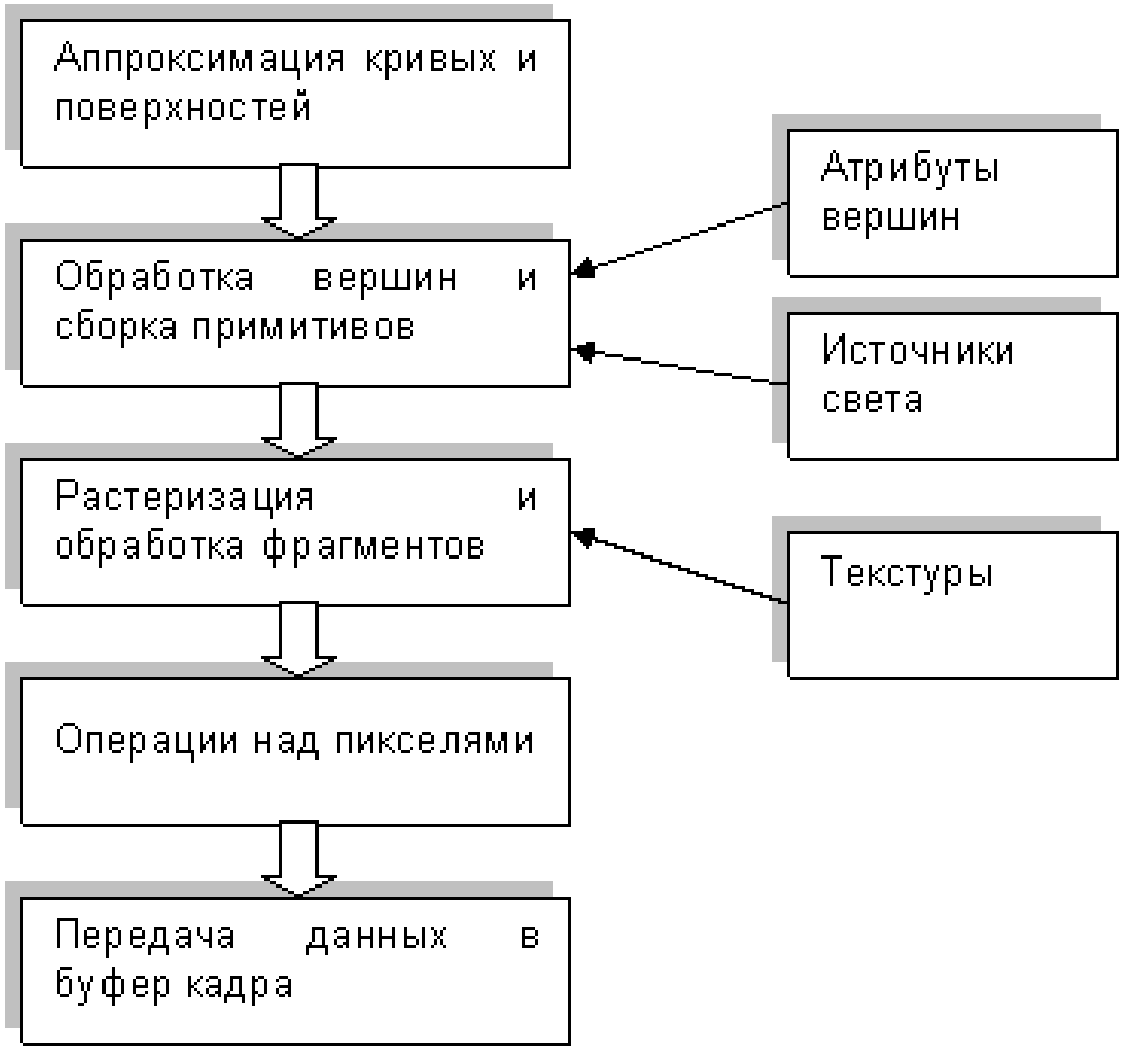
   g++ main01.cpp -o gl -lGL -lGLU –lglut

3. В коде нужно сделать замену подключаемого хедера:

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

   +++ #include <GL/freeglut.h>

Конвейер


Аппроксимация кривых и поверхностей
Обработка вершин и сборка примитивов
Растеризация и обработка фрагментов
Операции над пикселями
Передача данных в буфер кадра
Атрибуты вершин
Источники света
Текстуры

Минимальная программа

main00.c

1. Инициализация GLUT

   void glutInit(int *argcp, char **argv); //обычно из main

2. Установка параметров окна.
3. Создание окна.

   int glutCreateWindow(const char *title);

4. Установка функций, отвечающих за рисование в окне и изменении формы окна.
5. Вход в главный цикл GLUT.

Установка параметров окна

- **void glutInitWindowSize(int width, int height);**

  размер окна
- **void glutInitWindowPosition(int x, int y);**

  положение относительно верхнего левого угла
- **void glutInitDisplayMode(unsigned int mode);**

  режим отображения информации: используемая цветовая модель, количество различных буферов, и т.д.

  void glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);

Установка параметров окна

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.
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.
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 нужна специальная аппаратура.

Установка функций рисования и изменения формы окна

- **void glutDisplayFunc(void (*func)(void));**

  Перерисовка окна (первый запуск, разворот)

  void Draw();

  glutDisplayFunc(Draw);

- **void glutReshapeFunc(void (*func)(int width,int height));**

  Изменение размеров окна.

  void Reshape(int w, int h);

  glutReshapeFunc(reshape);

Установка функций рисования и изменения формы окна

```
void reshape(int w, int h)
{
    glViewport(0, 0, w, h); //область вывода изображения
                            //размером со все окно
    glMatrixMode(GL_PROJECTION); //загрузим матрицу проекции
        //отвечает за добавление в нашу сцену перспективного вида
    glLoadIdentity(); //заменим ее единичной (сброс)
    gluOrtho2D(0, w, 0, h); //и установим ортогональную проекцию

    glMatrixMode(GL_MODELVIEW); //загрузим модельно-видовую матрицу
        //"место", где сохранена информация о наших объектах
    glLoadIdentity(); //и заменим ее единичной
}
```

Вход в главный цикл GLUT

void glutMainLoop(void);

Сердце GLUT:

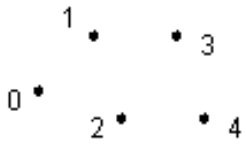
- бесконечный цикл – условие выхода нужно прописывать отдельно
- взаимосвязь между операционной системой и теми функциями, которые отвечают за окно, получают информацию от устройств ввода/вывода

Примитивы

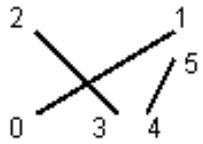
glBegin (тип_примитива)…glEnd ()

- **GL_POINTS** — каждая вершина задает точку
- **GL_LINES** — каждая отдельная пара вершин задает линию
- **GL_LINE_STRIP** — каждая пара вершин задает линию (т.е. конец предыдущей линии является началом следующей)
- **GL_LINE_LOOP** — аналогично предыдущему за исключением того, что последняя вершина соединяется с первой и получается замкнутая фигура
- **GL_TRIANGLES** — каждая отдельная тройка вершин задает треугольник
- **GL_TRIANGLE_STRIP** — каждая следующая вершина задает треугольник вместе с двумя предыдущими (получается *лента* из треугольников)
- **GL_TRIANGLE_FAN** — каждый треугольник задается первой вершиной и последующими парами (т.е. треугольники строятся вокруг первой вершины, образуя нечто похожее на диафрагму)
- **GL_QUADS** — каждые четыре вершины образуют четырехугольник
- **GL_QUAD_STRIP** — каждая следующая пара вершин образует четырехугольник вместе с парой предыдущих
- **GL_POLYGON** — задает многоугольник с количеством углов равным количеству заданных вершин

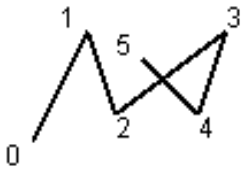
Примитивы



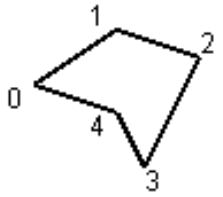
GL_POINTS



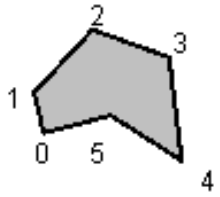
GL_LINES



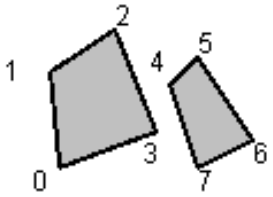
GL_LINE_STRIP



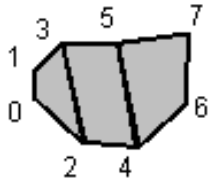
GL_LINE_LOOP



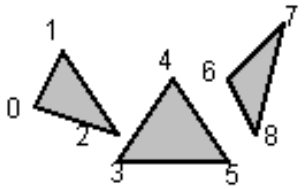
GL_POLYGON



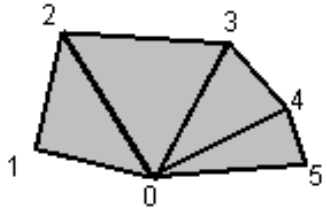
GL_QUADS



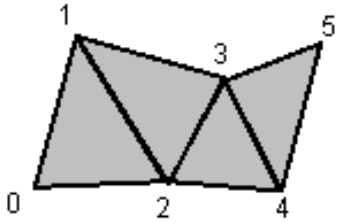
GL QUAD STRIP



GL TRIANGLES

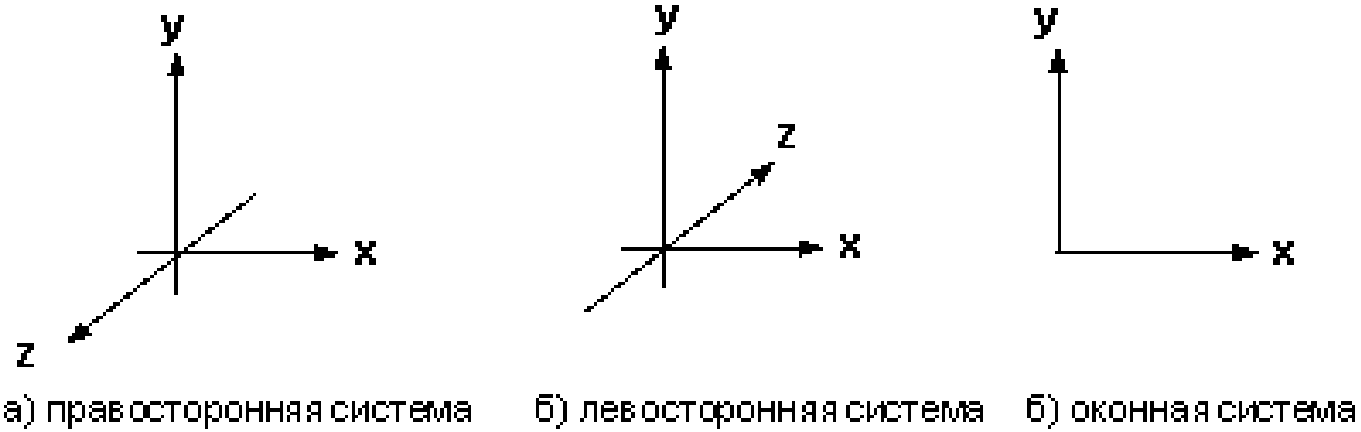


GL TRIANGLE FAN



GL TRIANGLE STRIP

Системы координат



а) правосторонняя система б) левосторонняя система б) оконная система

Левосторонняя: задание значений параметрам команды gluPerspective(), glOrtho(),

Правосторонняя: во всех остальных случаях.

Двумерная *оконная* система координат: итоговая.

Матрицы

Модельно-видовая

преобразования объекта в мировых координатах (параллельный перенос, изменение масштаба, поворот)

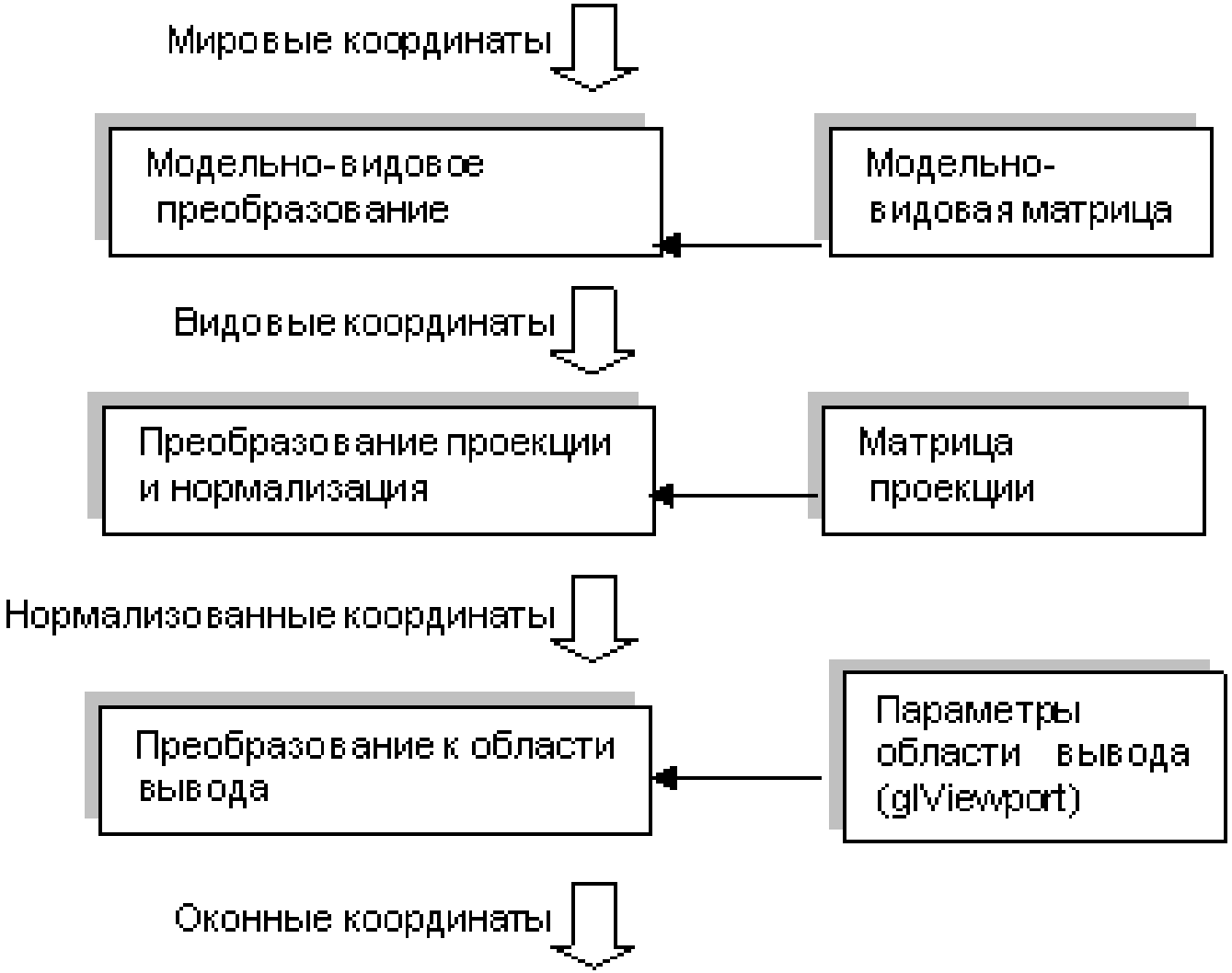
Проекций

как будут проецироваться трехмерные объекты на плоскость экрана

Текстуры

наложение текстуры на объект

Матрицы


Мировые координаты
Модельно-видовое преобразование
Модельно-видовая матрица
Видовые координаты
Преобразование проекции и нормализация
Матрица проекции
Нормализованные координаты
Преобразование к области вывода
Параметры области вывода (glViewport)
Оконные координаты

Матрицы

void **glMatrixMode** (GLenum *mode*)

GL_MODELVIEW GL_PROJECTION GL_TEXTURE

выбирает, какую матрицу менять

void **glLoadMatrix[f d]** (GLtype **m*)

m - массив из 16 элементов типа float или double

void **glLoadIdentity** (void)

заменяет текущую матрицу на единичную

void **glPushMatrix** (void) void **glPopMatrix** (void)

сохранить и восстановить матрицу

Модельно-видовые преобразования

void **glTranslate[f d]** (GLtype *x*, GLtype *y*, GLtype *z*)

перенос

void **glRotate[f d]** (GLtype *angle*, GLtype *x*, GLtype *y*, GLtype z)

поворот против часовой стрелки

void **glScale[f d]** (GLtype *x*, GLtype *y*, GLtype *z*)

масштаб

void **gluLookAt** (GLdouble *eyex*, GLdouble *eyey*, GLdouble *eyez*, GLdouble *centerx*, GLdouble *centery*, GLdouble *centerz*, GLdouble *upx*, GLdouble *upy*, GLdouble *upz*)

изменить положение наблюдателя

Проекции

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

усеченный объем видимости в левосторонней системе координат

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

значения *near* и *far* устанавливаются равными

–1 и 1 соответственно

void **gluPerspective** (GLdouble *angley*, GLdouble *aspect*,
GLdouble *znear*, GLdouble *zfar*)

усеченный конус видимости в левосторонней системе координат: *angley* – угол по оси Y, *aspect* – угол видимости по оси X (отношение размеров окна), zfar и znear - плоскости отсечения по глубине

Область вывода

void **glViewPort** (GLint *x*, GLint *y*,
GLint *width*, GLint *height*)

размеры области вывода в оконной системе координат

void **gluPerspective** (GLdouble *angley*, GLdouble *aspect*, GLdouble *znear*, GLdouble *zfar*)

усеченный конус видимости в левосторонней системе координат: *angley* – угол по оси Y, *aspect* – угол видимости по оси X (отношение размеров окна), zfar и znear - плоскости отсечения по глубине

Анимация

main01.c : функция таймера

void glutTimerFunc
(unsigned int millis, void (*func)(int value), int value);

- время таймера
- функция, вызываемая по истечении таймера
- идентификатор таймера

```
void timf(int value) // Timer function
{
	glutPostRedisplay(); // Redraw windows
	glutTimerFunc(40, timf, 0); // Setup next timer
}

glutTimerFunc(40, timf, 0); // Set up timer for 40ms, about 25 fps
```

Мышь

Функционал мыши добавляется аналогично

void glutMouseFunc

(void (*func) (int button, int state, int x, int y));

- функция, вызываемая при клике мышью; должна принимать три параметра:
  - номер кнопки мыши (0 – левая, 1 – средняя, 2 - правая)
  - состояние (0 – нажали, 1 – отпустили)
  - координаты

```
void MouseFunc(int button, int state, int x, int y)
{
    printf(" %d", button);
}

glutMouseFunc(MouseFunc);
```

Клавиатура

Функционал клавиатуры добавляется аналогично

void glutKeyboardFunc

(void (*func)(unsigned char key, int x, int y));

- функция, вызываемая при нажатии клавиши; должна принимать три параметра:
  - код нажатого символа
  - координаты мыши в момент нажатия

```
void KBFunc(unsigned char key, int x, int y)
{
    printf(" %d", key);
}
glutKeyboardFunc (KBFunc);
```

Дальнейшее изучение

Выше были использованы следующие источники (их и рекомендую для дальнейшего изучения):

http://pmg.org.ru/nehe/

Перевод относительно известного комплекта туториалов – не нового, но все еще актуального.

http://grafika.me/lessons

Сборная солянка из различных уроков, комплект про GLUT более чем неплох.

http://rsdn.ru/article/opengl/ogltut2.xml

Хороший учебник и справочник. Идти по нему подряд не очень просто, но, если что-то непонятно в вышеуказанных туториалах, здесь все достаточно подробно.

Задачи

0. Нарисовать треугольник.
1. Заставить его вращаться.
2. По клику мышью рисовать на месте клика небольшой синий квадрат.
3. «Индикатор с планеты Блук». При клике на квадрат он перелетает на произвольный участок экрана и меняет цвет. После десяти кликов меняет цвет на черный и на клики не реагирует.
4. Нарисовать тетраэдр, заставить его вращаться.