OpenGL

Лекция 1 (19.09.2013)

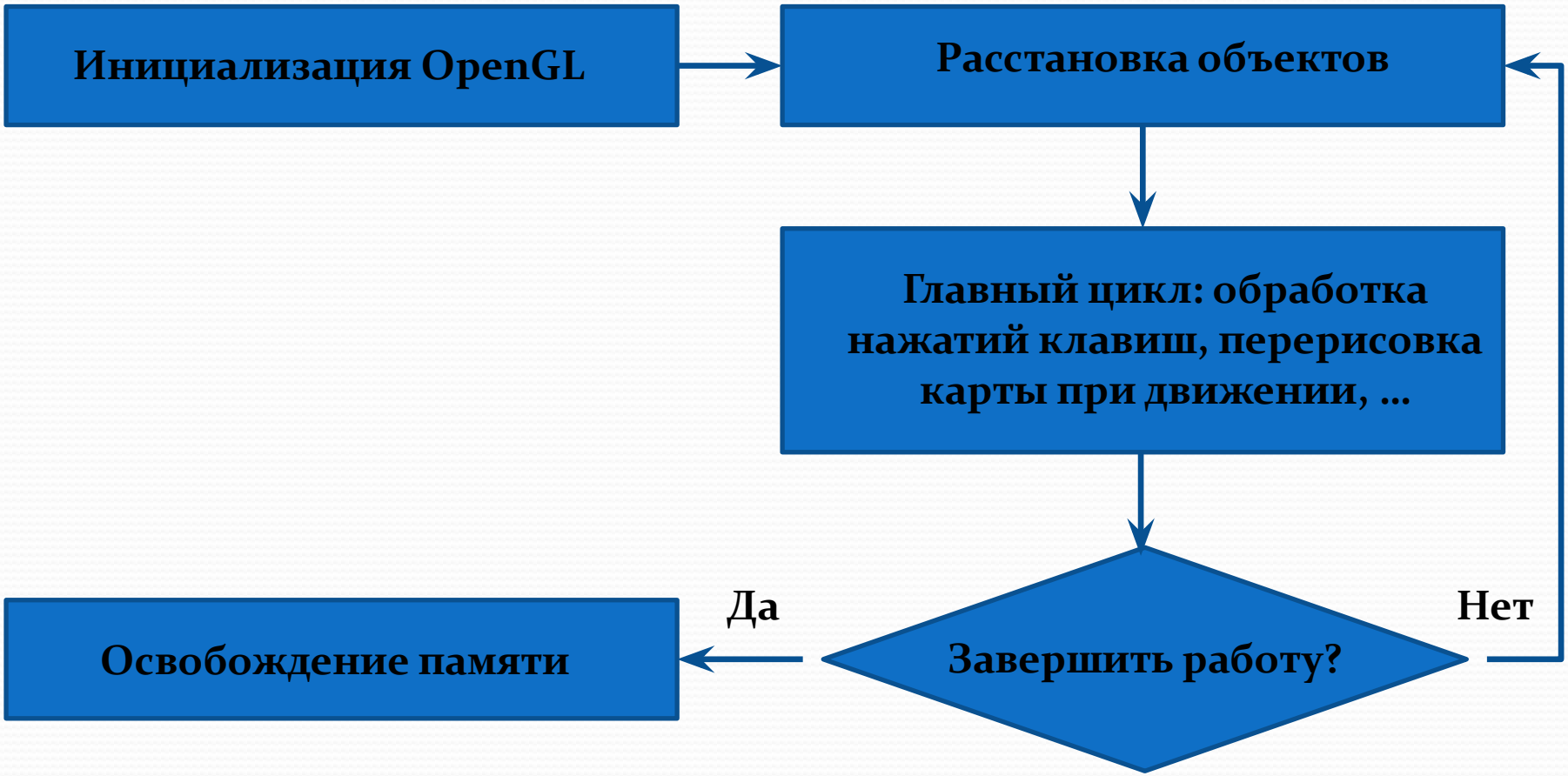
Основные возможности OpenGL

- Набор базовых примитивов: точки, линии, многоугольники и т.п.
- Видовые и координатные преобразования
- Удаление невидимых линий и поверхностей (z-буфер)
- Использование сплайнов для построения линий и поверхностей
- Наложение текстуры и применение освещения
- Добавление специальных эффектов: тумана, изменение прозрачности,сопряжение цветов (blending), устранение ступенчатости (anti-aliasing).

Библиотеки OpenGL

- На данный момент реализация OpenGL включает в себя несколько библиотек (описание базовых функций OpenGL, GLU,GLUT,GLAUX и другие).
- Библиотека GLAUX уступает по популярности написанной несколько позже библиотеке GLUT, хотя они предоставляют примерно одинаковые возможности.
- В состав библиотеки GLU вошла реализация более сложных функций, таких как набор популярных геометрических примитивов (куб, шар, цилиндр, диск), функции построения сплайнов, реализация дополнительных операций над матрицами и т.п. Все они реализованы через базовые функции OpenGL.

Структура программы на OpenGL



Названия команд

- type **glCommand_name**[1 2 3 4][**b s i f d ub us ui**][**v**](type1 arg1,…,typeN argN)
- gl это имя библиотеки, в которой описана эта функция: для базовых функций OpenGL, функций из библиотек GLU, GLUT, GLAUX это gl, glu, glut, aux соответственно.
- Command_name имя команды [1 2 3 4]число аргументов команды [b s i f d ub us ui]тип аргумента:
- символ b означает тип GLbyte (аналог char в C\C++), символ f тип GLfloat (аналог float), символ i– тип GLint(аналог int) и так далее. Полный список типов и их описание можно посмотреть в файле gl.h
- [v] наличие этого символа показывает, что в качестве параметров функции используется указатель на массив значений

Инициализация OpenGL

- Инициализация проводится с помощью функции **glutInit**(int *argc, char **argv)
- Эта функция проводит необходимые начальные действия для построения окна приложения, и только несколько функций GLUT могут быть вызваны до нее. К ним относятся:
- **glutInitWindowPosition** (int x, int y) **glutInitWindowSize** (int width, int height) **glutInitDisplayMode** (unsigned int mode) Первые две функции задают соответственно положение и размер окна, а последняя функция определяет различные режимы отображения информации, которые могут совместно задаваться с использованием операции побитового “или”(|):

Режимы отображения

- **GLUT_RGBA** Режим RGBA. Используется по умолчанию, если не указаны явно режимы GLUT_RGBA или GLUT_INDEX. **GLUT_RGB** То же, что и GLUT_RGBA. **GLUT_INDEX** Режим индексированных цветов (использование палитры). Отменяет GLUT_RGBA. **GLUT_SINGLE** Окно с одиночным буфером. Используется по умолчанию. **GLUT_DOUBLE** Окно с двойным буфером. Отменяет GLUT_SINGLE. **GLUT_DEPTH** Окно с буфером глубины.

Виды буферов

- Двойной буфер обычно используют для анимации, сначала рисуя что-нибудь в одном буфере, а затем меняя их местами, что позволяет избежать мерцания. Буфер глубины или z-буфер используется для удаления невидимых линий и поверхностей.

Листинг инициализации

- glutInit(&argc, argv);
- glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGB);
- int cx = GetSystemMetrics(SM_CXFULLSCREEN); int cy = GetSystemMetrics(SM_CYFULLSCREEN);
- glutInitWindowSize(cx, cy);
- glutCreateWindow(“Lecture 1");

Обработка событий

- void **glutDisplayFunc** (void (*func) (void))
- void **glutReshapeFunc** (void (*func) (int width, int height))
- void **glutMouseFunc** (void (*func) (int button, int state, int x, int y))
- void **glutIdleFunc** (void (*func) (void))
- Параметром для них является имя соответствующей функции заданного типа. С помощью glutDisplayFunc() задается функция рисования для окна приложения, которая вызывается при необходимости создания или восстановления изображения. Для явного указания, что окно надо обновить, иногда удобно использовать функцию void **glutPostRedisplay**(void)
- Через glutReshapeFunc() устанавливается функция обработки изменения размеров окна пользователем, которой передаются новые размеры.
- glutMouseFunc() определяет обработчика команд от мыши, а glutIdleFunc() задает функцию, которая будет вызываться каждый раз, когда нет событий от пользователя.
- Контроль всех событий происходит внутри бесконечного цикла в функции void **glutMainLoop**(void) которая обычно вызывается в конце любой программы, использующей GLUT.

Структура приложения, использующего анимацию

- #include <GL/glut.h>
- void MyIdle(void){ … };
- void MyDisplay(void){ … glutSwapBuffers(); };
- void main(int argc, char **argv){
- glutInit(&argc, argv);
- glutInitWindowSize(640, 480);
- glutInitWindowPosition(0, 0);
- glutCreateWindow("My OpenGL Application");
- glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
- glutDisplayFunc(MyDisplay);
- glutIdleFunc(MyIdle);
- glutMainLoop();
- };

Очистка экрана

- Для задания цвета фона используется команда void **glClearColor**(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha). Значения должны находиться в отрезке [0,1] и по умолчанию равны нулю. После этого вызов команды void **glClear**(GLbitfield mask) с параметром GL_COLOR_BUFFER_BIT устанавливает цвет фона во все буфера, доступные для записи цвета (иногда удобно использовать несколько буферов цвета).

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();

- Вызов функции gLoadIdentity() устанавливает начало системы координат в центр экрана, причем ось X идет слева направо, ось Y вверх и вниз, а ось Z к и от наблюдателя. Центр OpenGL экрана находится в точке 0, 0, 0. Координаты, расположенные слева, снизу и вглубь от него, имеют отрицательное значение, расположенные справа, сверху и по направлению к наблюдателю – положительное.

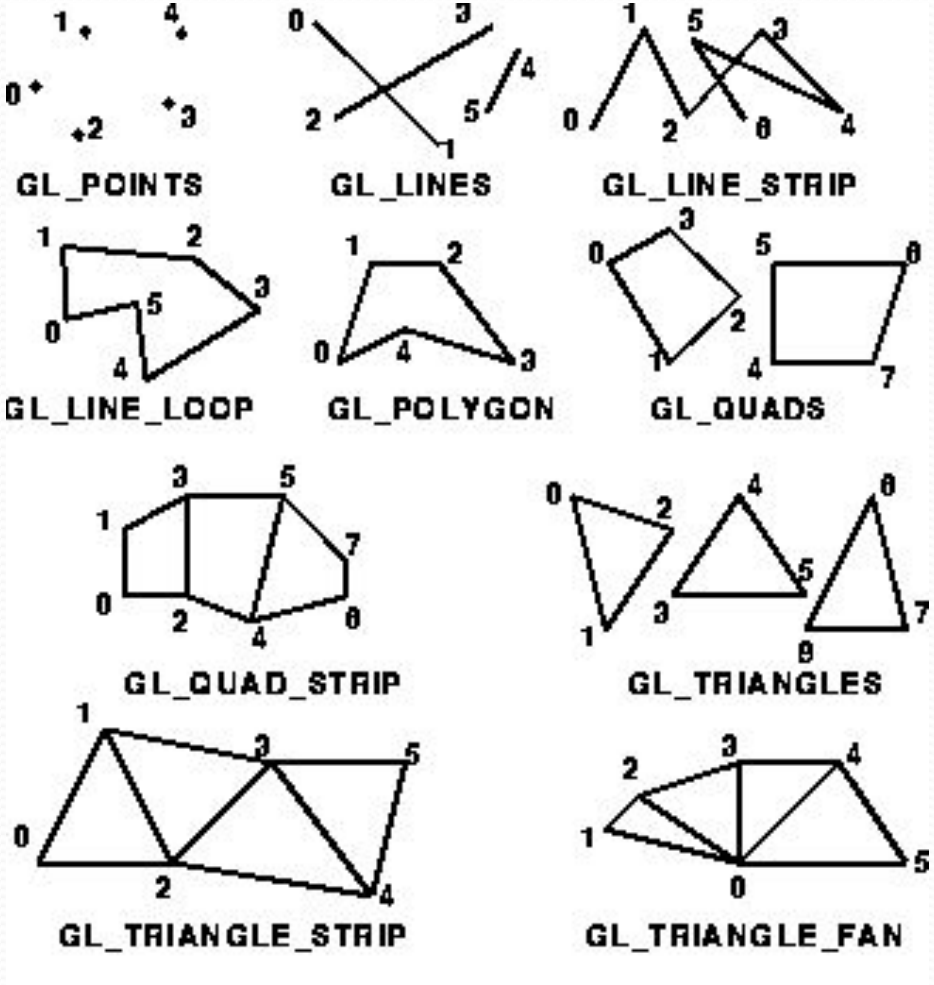
Вершины

- 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, другие системы координат будут рассмотрены ниже.

Примитивы

- Задание примитива происходит внутри командных скобок:
- void **glBegin**(GLenum mode)
- void **glEnd**(void)
- 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.
- **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.

Примитивы


GL_POINTS
GL_LINES
GL_LINE_STRIP
GL_LINE_LOOP
GL_POLYGON
GL_QUADS
GL_QUAD_STRIP
GL_TRIANGLES
GL_TRIANGLE_STRIP
GL_TRIANGLE_FAN

Цвет вершин

- Для задания текущего цвета вершины используются команды
- void **glColor**[3 4][**b s i f**](GLtype components)
- void **glColor**[3 4][**b s i f**]**v**(GLtype components)
- Первые три параметра задают R, G, B компоненты цвета, а последний параметр определяет alpha-компоненту, которая задает уровень прозрачности объекта. Если в названии команды указан тип ‘f’ (float), то значения всех параметров должны принадлежать отрезку [0,1], при этом по умолчанию значение alpha-компоненты устанавливается равным 1.0, что соответствует полной непрозрачности. Если указан тип ‘ub’ (unsigned byte), то значения должны лежать в отрезке [0,255].
- Разным вершинам можно назначать различные цвета и тогда будет проводиться линейная интерполяция цветов по поверхности примитива.
- Для управления режимом интерполяции цветов используется команда void **glShadeModel**(GLenummode) вызов которой с параметром GL_SMOOTH включает интерполяцию (установка по умолчанию), а с GL_FLAT отключает.

Как нарисовать треугольник?

- GLfloat BlueCol[3]={0,0,1};
- glBegin(GL_TRIANGLES);
- glColor3f(1.0, 0.0, 0.0); //красный
- glVertex3f(0.0, 0.0, 0.0);
- glColor3ub(0,255,0); //зеленый
- glVertex3f(1.0, 0.0, 0.0);
- glColor3fv(BlueCol); //синий
- glVertex3f(1.0, 1.0, 0.0);
- glEnd();