Вычислительная геометрия и алгоритмы компьютерной графики

Лекция №3

Рябинин Константин Валентинович

e-mail: kostya.ryabinin@gmail.com

Пермь, 2015

Использование OpenGL

Для упрощённого вхождения в OpenGL, на первом этапе рассмотрим «олдскульные» методы построения изображения

Они характеризуются:

- **Лёгкостью реализации**
- **Невысокой скоростью работы**
- **Сильно ограниченной свободой создания визуальных эффектов**

Использование OpenGL

- Данная функциональность поддерживается в OpenGL младших версий, а начиная с версии 3.1 была помечена как *deprecated* и в версии 3.3 уже удалена (дата смерти: 11 марта 2010 года, RIP)
- На сегодняшний день, даже имея самые свежие драйверы графического оборудования, есть возможность использовать старые (до 3.x) версии OpenGL
- Либо, есть возможность использовать OpenGL новой версии с *профилем совместимости*
- Однако, для достижения высокого быстродействия и высокого качества картинки, следует ориентироваться на новую функциональность
- Список функций с соответствующими им номерами версий (введение и удаление) может быть найден по адресу http://www.opengl.org/registry/api/gl.spec
- Спецификации OpenGL и описания модели «запрещения» (deprecation model) могут быть найдены в документах OpenGL X.X Core Profile Specification по адресу http://www.opengl.org/registry/

Атрибуты вершин

- Координаты
  void glVertex3f(GLfloat x, GLfloat y, GLfloat z)
- Координаты нормали
  void glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz)
- Координаты текстуры
  void glTexCoord2f(GLfloat u, GLfloat v)
- Цвет
  void glColor3ub(GLubyte r, GLubyte g, CLubyte b)
- . . .

Построение геометрии

Вывод множества вершин

```
glBegin(GL_QUADS);
    glColor3ub(255, 0, 0);
    glVertex2d(-0.5, -0.5);
    glColor3ub(0, 255, 0);
    glVertex2d(0.5, -0.5);
    glColor3ub(0, 0, 255);
    glVertex2d(0.5, 0.5);
    glColor3ub(255, 255, 0);
    glVertex2d(-0.5, 0.5);
glEnd();
```

Построение геометрии

Типы цепочек вершин

- GL_POINTS – вывод вершин точками
- GL_LINES – соединение каждой следующей пары вершин линией
- GL_TRIANGLES – соединение каждой тройки вершин в треугольник
- GL_QUADS – соединение каждой четвёрки вершин в четырёхугольник
- GL_POLYGON – соединение всей вершин в многоугольник
- . . .

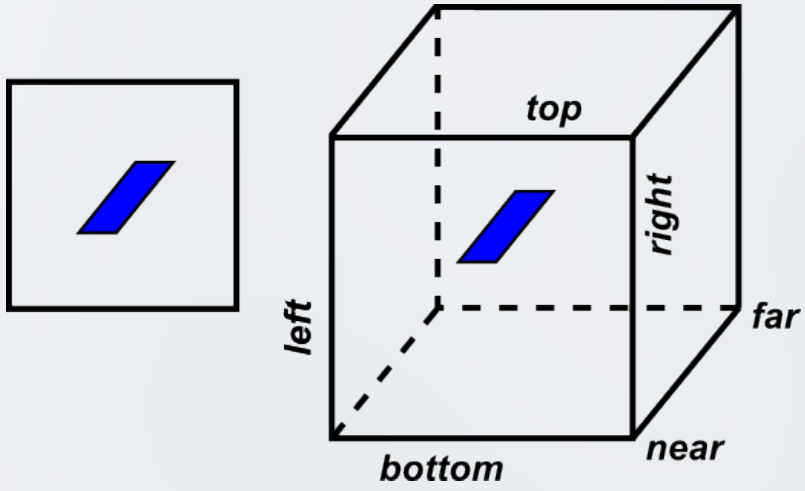
Способы отображения многоугольников

- Закрашенные полигоны
  glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
- Проволочный каркас
  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
- Точки вершин
  glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);

Проекции

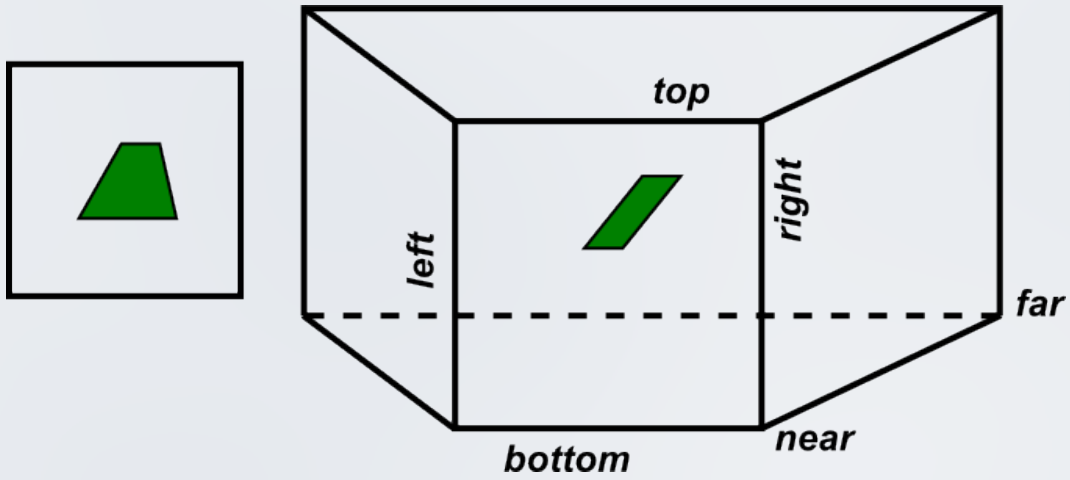
- Заданием параметров проекции определяется видимая область пространства

Параллельная проекция




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

Перспективная проекция

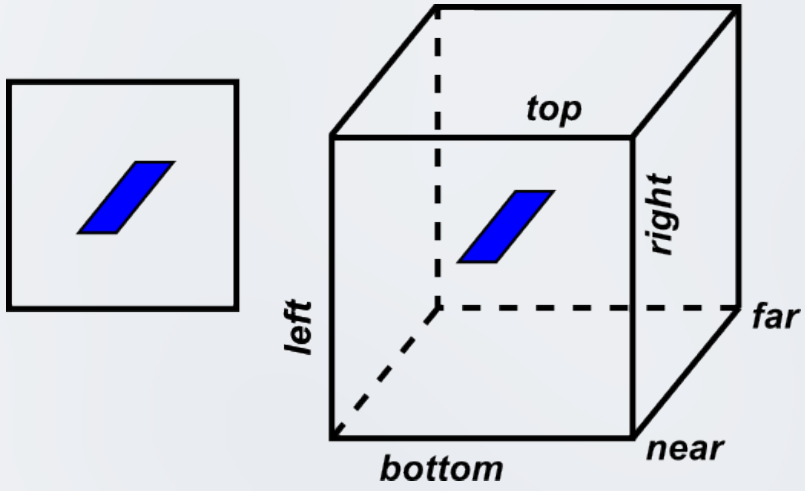



```
glFrustum(GLdouble left,
          GLdouble right,
          GLdouble bottom,
          GLdouble top,
          GLdouble near,
          GLdouble far)
```

Проекции

- Заданием параметров проекции определяется видимая область пространства

Параллельная проекция

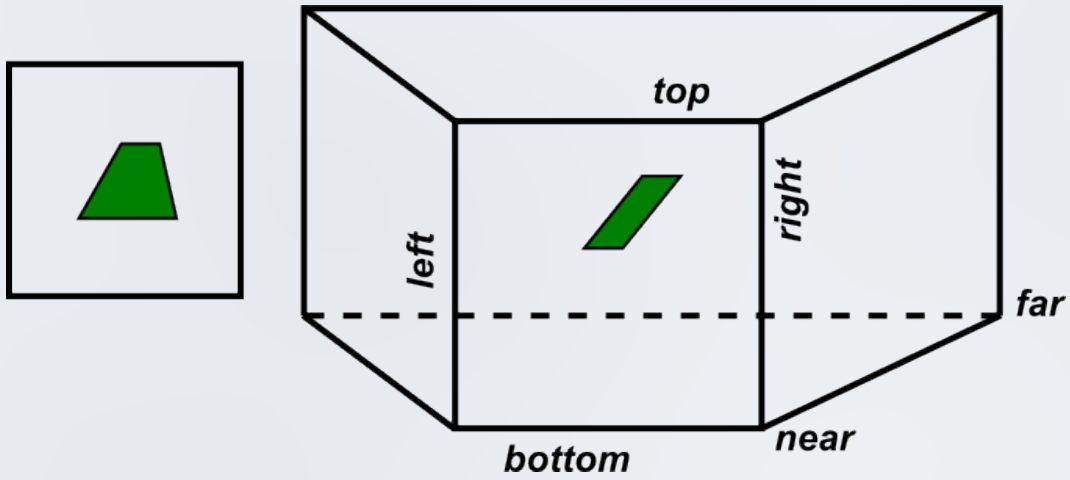



```
gluOrtho2D(GLdouble left,
           GLdouble right,
           GLdouble bottom,
           GLdouble top)
```

~

```
glOrtho(left, right, bottom, top, -1, 1)
```

Перспективная проекция




```
gluPerspective(GLdouble fov,
               GLdouble aspect,
               GLdouble near,
               GLdouble far)
```

~

```
h = tan((fov / 2) / 180 * M_PI) * near
w = h * aspect
glFrustum(-w, w, -h, h, near, far)
```

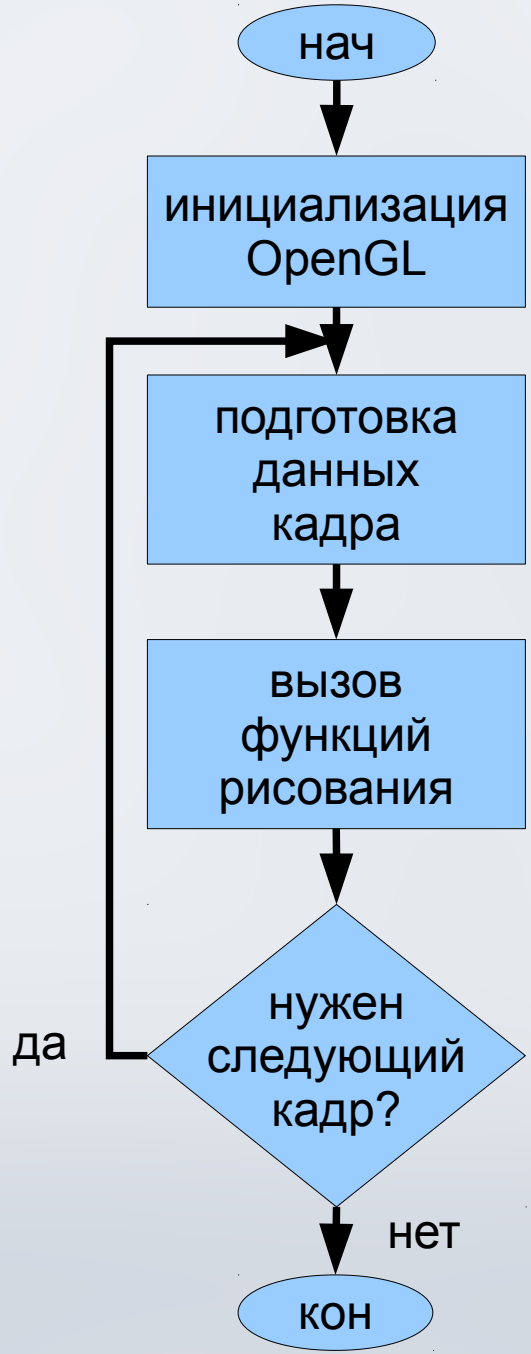
Камера

Камера – это псевдообъект в трёхмерном пространстве, характеризующий положение наблюдателя

- Далеко не во всех системах камера в явном виде имеет место
- Человеку удобно работать с камерой, поэтому в системах, не предполагающих её наличия, она вводится в качестве метафоры
- В OpenGL камера отсутствует, вместо неё – унифицированный механизм матричных преобразований (MODELVIEW)
- Функция-обёртка, моделирующая камеру:

```
gluLookAt(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
          GLdouble centerX, GLdouble centerY, GLdouble centerZ,
          GLdouble upX, GLdouble upY, GLdouble upZ)
```

Структура программы



Организация интерактивности

- Наиболее естественный способ – использование событийно-ориентированных систем
- Имитацию событийной ориентированности предоставляет GULT при помощи регистрации функций обратного вызова на нажатие клавиши (glutKeyboardFunc), перемещение мыши (glutMotionFunc) и клик мыши (glutMouseFunc)
- При отсутствии событийно-ориентированных средств следует осуществлять последовательный опрос устройств ввода

Анимация

Анимация – это вдыхание жизни в сцену :)

Анимация – это последовательное отображение кадров с различным содержанием
(конструктивно)

Программно-аппаратная поддержка анимации:
механизм двойной буферизации

→ **Строго говоря, любое свойство объекта может быть анимировано**

Важное требование к анимации:
сохранение межкадрового соответствия
(***frame-to-frame coherence***)

Скорость анимации

Анимация по таймеру:

- Идея: запуск некоторого таймера (должен поддерживаться системой), который в наперёд заданные моменты времени вызывает функцию обновления состояния
- Способ имеет право на существование лишь в исключительных случаях
- Таймер лучше использовать как триггер анимации, а не как её «драйвер»

Непрерывная анимация:

- Идея: обновление состояния происходит с максимально возможной скоростью, а величина изменения вычисляется на основе желаемой и фактической скорости