Мультимедиа
Лекция №7

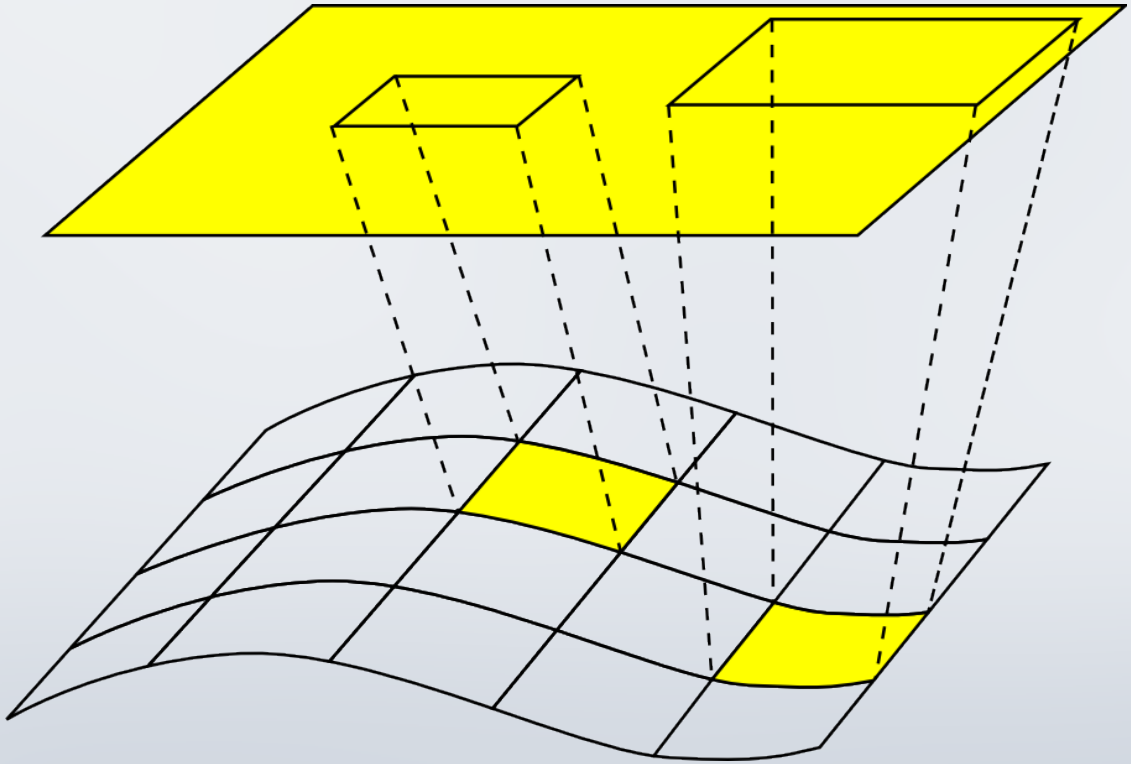
Рябинин Константин Валентинович

e-mail: icosaeder@ya.ru
jabber: icosaeder@jabber.ru

Пермь, 2013

Текстура

Текстура – растровое изображение, накладываемое на поверхность полигонов, из которых состоят 3D-модели, для придания ей цвета, окраски и иллюзии рельефа

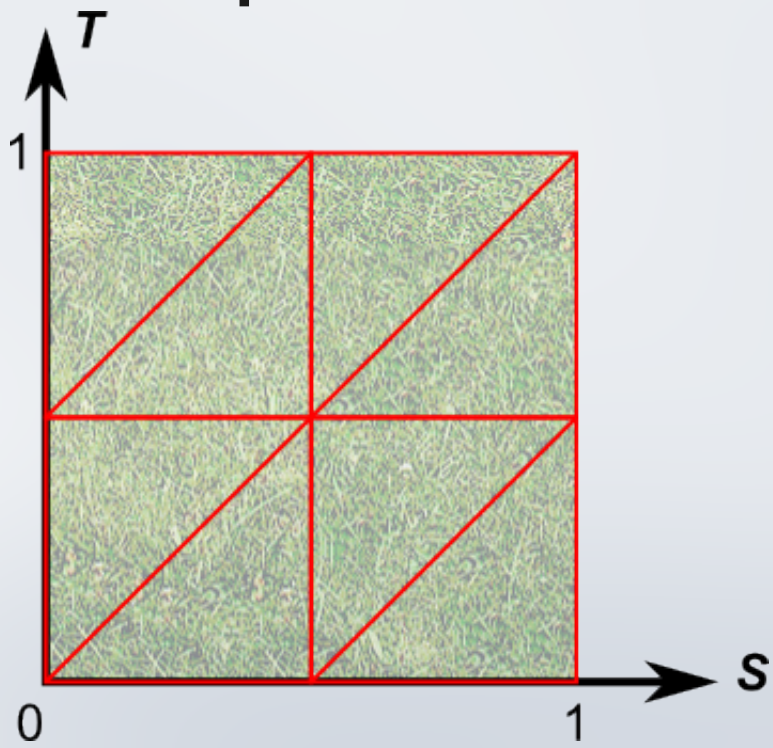


Текстурные координаты

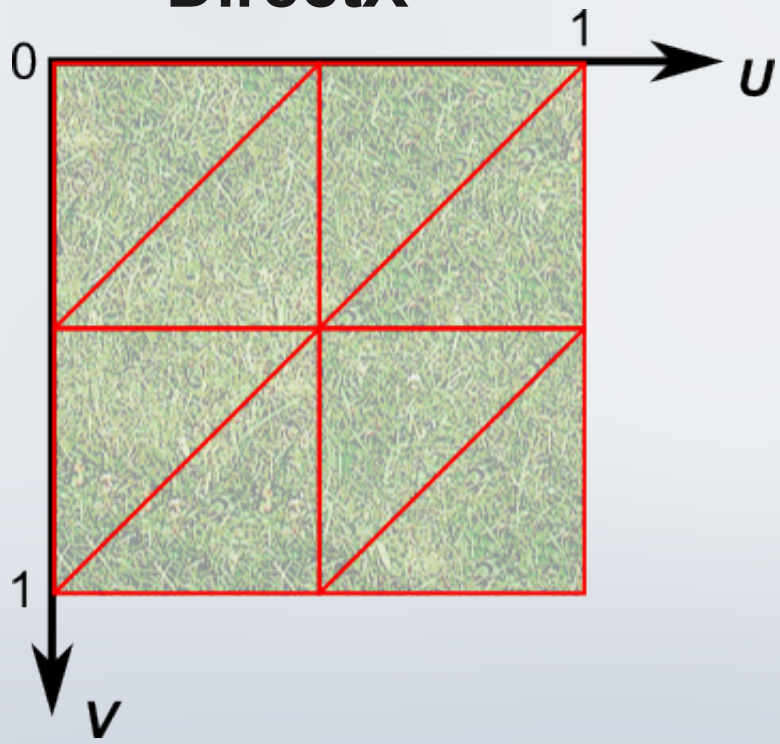
Текстурные координаты вершины – координаты проекции данной вершины на изображение

- **Система координат определяется соглашением внутри используемой спецификации**

OpenGL



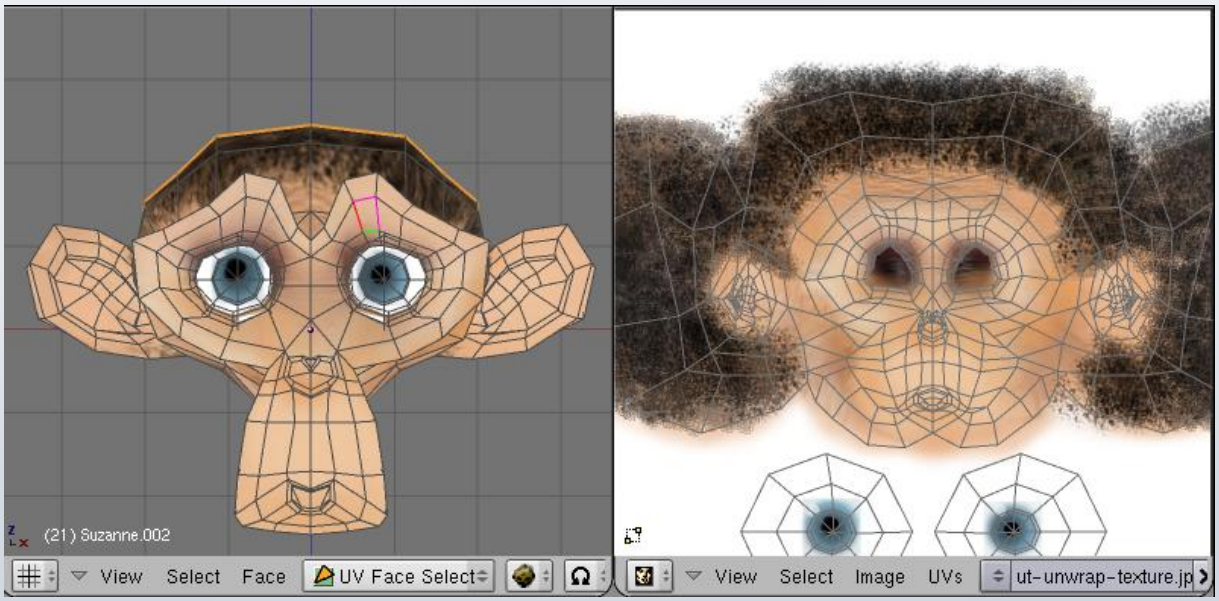
DirectX



Карта текстуры

Карта текстуры для 3D-модели – это множество координат для каждой вершины данной модели

- **В частных случаях карта может быть сгенерирована для модели автоматически, но в общем случае это невозможно. Поэтому как правило карта текстуры создаётся высокоуровневыми средствами (в 3D-редакторах) и загружается в программу вместе с моделью**



Получение текстуры

Подходы к созданию текстуры:

- Аналитически (процедурная текстура)
  - Простые шаблоны
    - Повторяющиеся регулярные узоры
    - Штриховки
    - . . .
  - Фракталы
  - Графики и изолинии функций
- Наперёд заданным растром (загрузка из файла)
- Рендеринг
  - Без постобработки
  - С постобработкой

→ Раньше было важно, чтобы длины сторон текстуры (в пикселях) были выражены степенями двойки

Наложение текстуры

Современные библиотеки вывода трёхмерной графики в обязательном порядке поддерживают наложение текстуры:

- Функции управления текстурными координатами
- Функции встраивания текстуры в материал объекта
  - Указание взаимодействия текстуры с моделью освещения
- Использование текстуры для создания сложных эффектов
  - Бамп-мэппинг
  - Воздействие на свойства материала
  - . . .
- Мультитекстурирование
- . . .

Текстуры в OpenGL

- Текстура в рамках спецификации OpenGL – это линейный массив байтов и не более того

Основные функции работы с текстурами:

- **Генерация текстурного «объекта»**
  glGenTextures(GLsizei n, GLuint *textures)
- **Указание текстуры, активной в данный момент**
  glBindTexture(GLenum target, GLuint texture)
- **Установка параметров текстуры**
  glTexParameterf(GLenum target, GLenum pname, GLfloat param)
  glTexParameteri(GLenum target, GLenum pname, GLint param)
  glTexParameterfv(GLenum target, GLenum pname, const GLfloat *params)
  glTexParameteriv(GLenum target, GLenum pname, const GLint *params)
- **Загрузка текстуры в видеопамять**
  glTexImage2D(GLenum target, GLint level, GLint internalFormat,
  GLsizei width, GLsizei height,
  GLint border, GLenum format, GLenum type, const GLvoid *data)

Текстуры в OpenGL

- Текстура в рамках спецификации OpenGL – это линейный массив байтов и не более того

Основные функции работы с текстурами:

- **Получение текстуры из видеопамяти**

```
glGetTexImage(GLenum target, GLint level, GLenum format, GLenum type, GLvoid *img)
```

- **Загрузка в видеопамять фрагмента текстуры**

```
glTexSubImage2D(GLenum target, GLint level,
                GLint xoffset, GLint yoffset, GLsizei width, GLsizei height,
                GLenum format, GLenum type, const GLvoid *data)
```

Мипмэппинг

Мипмэппинг (*multum-in-parvo mapping*) – это метод текстурирования, использующий несколько копий одной текстуры с разной детализацией

- Уровни детализации переключаются в зависимости от удалённости объектов с целью:
  - Адаптации детализации текстуры объекта к его удалённости
    (для постоянной детализации:
    объект близко – текстура меньше, чем надо, и изображение размыто;
    объект далеко – текстура больше, чем надо, и возникает случайный шум)
  - Ускорения работы за счёт снижения нагрузки на систему при обработке текстур меньшего разрешения для удалённых объектов
- Переход между уровнями:
  - Без фильтрации (проблема – скачки)
  - С билинейной фильтрацией
  - С трилинейной фильтрацией
  - С анизатропной фильтрацией
- Увеличение занимаемой памяти

Генерация мип-уровней

- Чисто теоретически на разных мип-уровнях могут быть совершенно разные текстуры, однако на практике в большинстве случаев это не имеет смысла
- Большинство систем автоматически создают мип-уровни по текстуре максимального разрешения (автоматизировано создание уменьшенных копий)
- В OpenGL генерация мип-уровней осуществляется функцией

```
gluBuild2DMipmaps(GLenum target, GLint internalFormat,
                  GLsizei width, GLsizei height,
                  GLenum format, GLenum type, const void *data)
```

которая заменяет собой glTexImage2D.
Также при помощи функции glTexParameteri следует указать способ перехода между уровнями, например

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                GL_LINEAR_MIPMAP_LINEAR)
```

Форматы хранения изображений

Формат хранения изображения – способ представления изображения во внешней памяти

Способы хранения:

- Дамп изображения
  - → RAW
- Без сжатия
  - → BMP, DIB
- Со сжатием с потерями
  - → JPG
- Со сжатием без потерь
  - → PNG, TGA, GIF, TIFF

→ Для загрузки сжатых изображений используются готовые библиотеки, осуществляющие декомпрессию

Рендеринг в текстуру

- Используется для динамической генерации содержательно сложных текстур на основе целых трёхмерных сцен
- Резко снижает производительность
- Большинство сложных эффектов моделирования теней и отражений основаны на рендеринге в текстуру
- Так или иначе рендеринг в текстуру должен поддерживаться библиотекой визуализации
- Идея состоит в копировании буфера цвета в текстурную видеопамять либо в подмене буфера цвета областью текстурной видеопамяти

Рендеринг в текстуру

В OpenGL:

- **Копирование содержимого буфера цвета в область текстурной видеопамяти**

```
setProjectionForTexture();
glViewport(0, 0, SIZE, SIZE);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
drawWhaterverForTexture();
glFulsh();
setupTextureParameters();
glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 0, 0, SIZE, SIZE);
setProjectionAndViewportForScene();
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
drawWhateverForSceneUsingGeneratedTexture();
```

Рендеринг в текстуру

- Использование FBO (Frame buffer object): есть возможность создать дополнительный буфер кадра, связанный с текстурой.
- Этот буфер кадра может быть в определённый момент выставлен в качестве текущего, в результате чего рендеринг будет производиться в него, а не в экранный буфер кадра.
- В итоге отрендеренный кадр будет сохранён в видеопамяти как текстура и может быть использован для последующего рендеринга основной сцены (после того, как экранный буфер кадра вновь установлен в качестве текущего)

- Пример: http://www.songho.ca/opengl/gl_fbo.html

Снимок экрана

- Получение данных из буфера цвета – вообще говоря подзадача рендеринга в текстуру, но как правило рендеринг в текстуру осуществляется иными командами с целью повышения эффективности, так как буфер цвета сам находится в видеопамяти, а при снимке экрана данные копируются в оперативную память
- Получение данных из буфера цвета поддерживается библиотеками визуализации на уровне встроенных команд
- Нет необходимости перенастройки порта просмотра, так как не нужно соблюдать требование степени двойки
- В OpenGL:

```
glReadPixels(GLint x, GLint y, GLsizei width, GLsizei height,
             GLenum format, GLenum type, GLvoid *data)
```