Лекция 2. OpenGL — шейдеры и текстуры

Илья Макаров

ИТМО

21 сентября 2022

Санкт-Петербург

Еще раз о шейдерах

Нас интересуют 2 вида шейдеров:

- **вершинный** - обрабатывает каждую вершину геометрии;
- **фрагментный** - обрабатывает каждый фрагмент (упрощенно пиксель).

Типичный шейдер:

```
#version version_number

in type in_name; // and others
out type out_name; // and maybe others
uniform type uni_name; // and maybe others

void main() {
    out_name = calculations;
}
```

Типы в шейдерах

Широкоиспользуемые:

- **int, float, double, uint, bool** - численные значения;
- **vec2, vec3, vec4** - вектора заданной размерности типа float;
- **mat2, mat3, mat4, ...** - матрицы заданной размерности, тип float.
- **bvecN, ivecN, uvecN, ...** - вектора, заданной размерности, но другого числового типа;

Вектора в шейдерах

Для векторных типов можно доступаться либо по индексу (не рекомендуется), либо по имени (**xyzw, rgba, stpq**).
Поддерживается swizzling.

```
void main() {
    vec4 color = vec4(1., 0., 0., 1);
    vec4 other_color = color.argb;

    vec4 pos = vec4(1., 1., 1., 1);
    vec4 other_pos = pos.xxyw; // swizzling

    vec2 tex = vec2(0., 0.5);
    vec3 other_tex = vec3(tex.st, 1.);
}
```

Примеры функций

- Работа с векторами и матрицами (cross, dot, reflect, normalize, length, ...);
- Тригонометрия (sin, cos, tan, asin, radians, degrees, ...);
- Математические функции (pow, sqrt, abs, log, ...);
- Фильтры (clamp, mix, step, ...);
- ...

Еще раз о location

Для связывания атрибутов из основного кода и значений внутри шейдеров используется **location**).

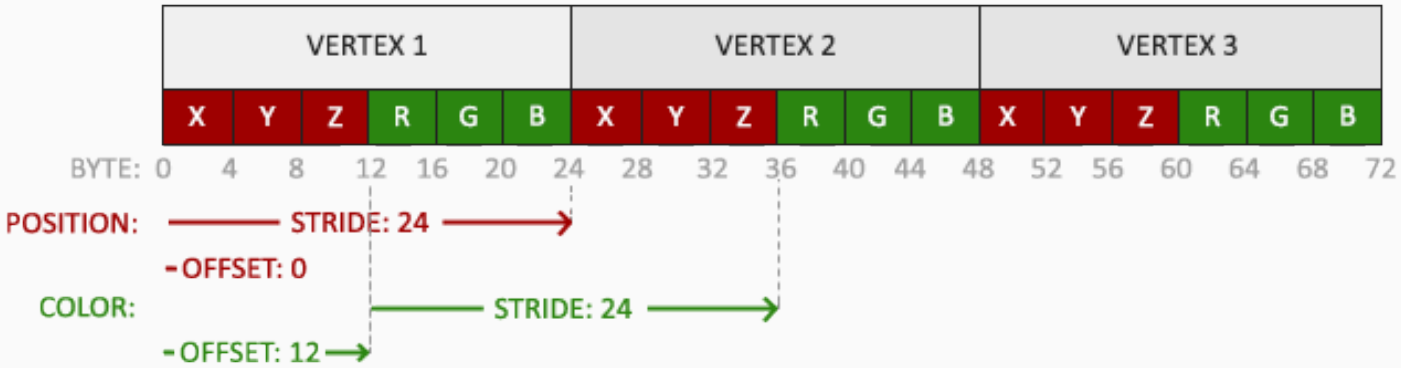
```
#version 330 core
layout (location = 0) in vec3 position; // 0
layout (location = 1) in vec4 color; // 1

out vec4 vectex_color;

void main() {
    vectex_color = vec4(0.5 * color.rgb, color.a);
    gl_Position = vec4(position, 1.0);
}
```

Еще раз о атрибутах

Сами данные хранятся как непрерывная последовательность байтов.



Еще раз юниформах

Пример код на C++:

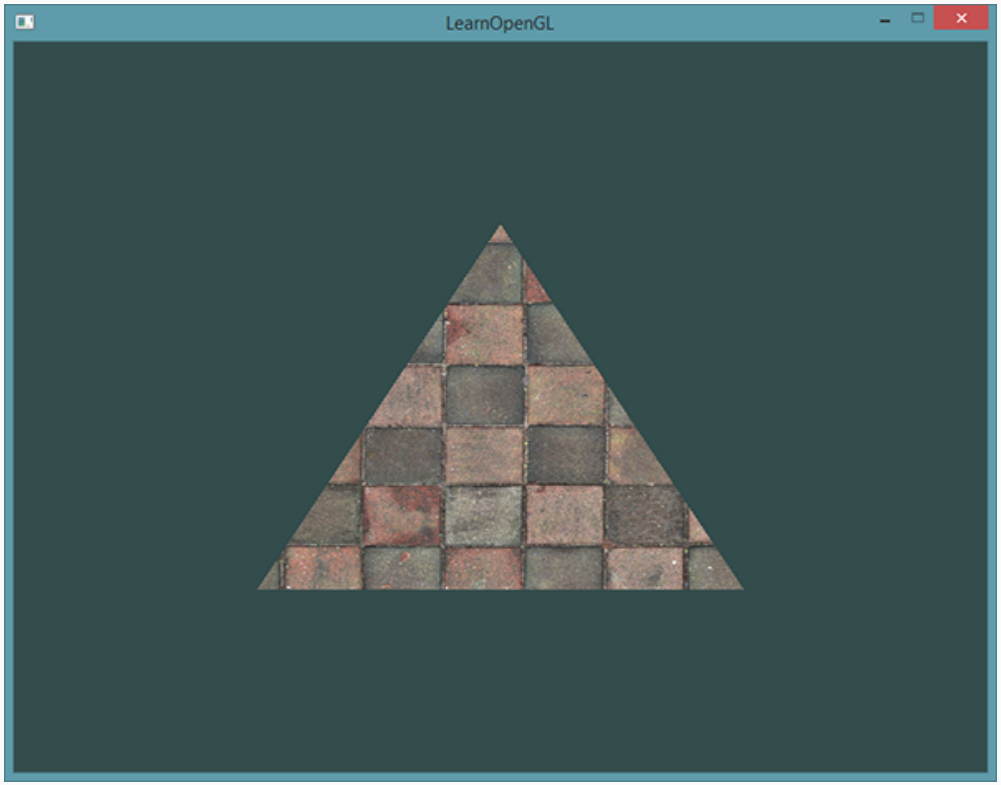
```
GLfloat green = (sin(time()) / 2) + 0.5;
GLint loc = glGetUniformLocation(shaderProgram,
                                 "u_color");
glUseProgram(shaderProgram);
glUniform4f(loc, 0.0f, green, 0.0f, 1.0f);
```

Пример шейдер:

```
#version 330 core
out vec4 color;
uniform vec4 u_color;
void main() {
    color = u_color;
}
```

Текстуры

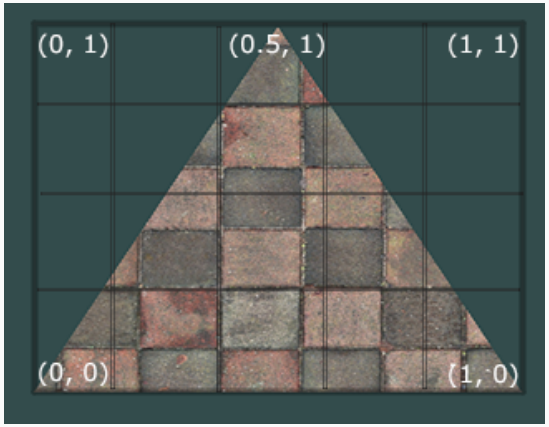
Текстура — это 2D изображение (1D и 3D текстура также существуют), используемое для добавления деталей объекту.



Текстуры

Для того, чтобы привязать текстуру к геометрии используются текстурные координаты.

- Получение цвета из текстуры - сэмплирование (**sampling**).
- **u, v** - обозначение, **[0, 1]** - диапазон значений типа float;



Texture wrapping

Возможно настроить поведение при выходе за диапазон координат.

GL_REPEAT GL_MIRRORED_REPEAT GL_CLAMP_TO_EDGE GL_CLAMP_TO_BORDER

Код для установки параметров:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
                GL_MIRRORED_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
                GL_MIRRORED_REPEAT);
```

Texture filtering

Текстурные координаты нормализованы и выбор конкретного пикселя (текселя) из текстуры может быть параметризован. Существуют различные варианты фильтрации:

- **GL_NEAREST** - ближайший.
- **GL_LINEAR** - билинейная фильтрация.

Texture mipmap

Служат для получения текстур меньшего размера. Для генерации существует соответсвующий метод **glGenerateMipmaps**.

Различные методы фильтрации:

- **GL_NEARESET_MIPMAPNEAREST**;
- **GL_LINEAR_MIPMAPNEAREST**;
- **GL_NEAREST_MIPMAPLINEAR**.

Texture filtering

Различные режимы фильтрации дают различные картинки.

GL_NEAREST

GL_LINEAR

Код для установки параметров:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                GL_LINEAR);
```

Загрузка текстуры

Генерация текстуры:

```
GLuint texture;
glGenTextures(1, &texture);
```

Заливка в видеопамять:

```
glBindTexture(GL_TEXTURE_2D, texture);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,
             width, height, 0, GL_RGB,
             GL_UNSIGNED_BYTE, image);
glGenerateMipmap(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, 0); // use default
```

Передача текстуры в шейдер

Сам объект текстуры же передаем как uniform переменную специального типа **sampler2D** .

```
#version 330 core
in vec2 tex_coord;
out vec4 color;
uniform sampler2D tex_2d;

void main() {
    color = texture(tex_2d, tex_coord);
}
```

Текстурные координаты передаем как атрибуты вершин.

Передача нескольких текстур

По умолчанию используется текстурный блок с индексом 0. Однако есть возможность передать несколько текстур.

```
// Activate block
glActiveTexture(GL_TEXTURE1);
// Now texture is binded for 1 block
glBindTexture(GL_TEXTURE_2D, texture);
```

Всего доступно 16 текстурных блоков **GL_TEXTURE0, GL_TEXTURE1, ..., GL_TEXTURE15**.

Передача нескольких текстур

Финальный шаг связать наши uniform-переменные и сами текстуры.

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture1);
glUniform1i(glGetUniformLocation(sp, "tex_2d"), 0);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, texture2);
glUniform1i(glGetUniformLocation(sp, "tex_2d_2"), 1);
```

Передача нескольких текстур

Фрагментный шейдер же примет вид:

```
// ...

uniform sampler2D tex_2d;
uniform sampler2D tex_2d_2;

void main() {
    color = mix(
        texture(tex_2d, tex_coord),
        texture(tex_2d_2, tex_coord), 0.4);
}
```