

Компьютерная графика

Лекция 5: Текстуры

2021

Что такое текстуры?

- ▶ Изображения в памяти GPU

Что такое текстуры?

- ▶ Изображения в памяти GPU
- ▶ Их можно читать из шейдеров

Что такое текстуры?

- ▶ Изображения в памяти GPU
- ▶ Их можно читать из шейдеров
- ▶ В них можно загружать пиксели

Что такое текстуры?

- ▶ Изображения в памяти GPU
- ▶ Их можно читать из шейдеров
- ▶ В них можно загружать пиксели
- ▶ В них можно рисовать (вместо рисования в окно)

Что такое текстуры?

- ▶ Изображения в памяти GPU
- ▶ Их можно читать из шейдеров
- ▶ В них можно загружать пиксели
- ▶ В них можно рисовать (вместо рисования в окно)
- ▶ Поддерживают много специфичных для изображений операций

Что нужно знать про текстуры?

- ▶ Типы текстур в OpenGL

Что нужно знать про текстуры?

- ▶ Типы текстур в OpenGL
- ▶ Как работать с текстурами в шейдере

Что нужно знать про текстуры?

- ▶ Типы текстур в OpenGL
- ▶ Как работать с текстурами в шейдере
- ▶ Какие есть настройки чтения из текстур

Что нужно знать про текстуры?

- ▶ Типы текстур в OpenGL
- ▶ Как работать с текстурами в шейдере
- ▶ Какие есть настройки чтения из текстур
- ▶ Как загрузить данные в текстуру

Что нужно знать про текстуры?

- ▶ Типы текстур в OpenGL
- ▶ Как работать с текстурами в шейдере
- ▶ Какие есть настройки чтения из текстур
- ▶ Как загрузить данные в текстуру
- ▶ Как привязать текстуру к шейдеру

Что такое текстуры в OpenGL?

- ▶ Объекты OpenGL

Что такое текстуры в OpenGL?

- ▶ Объекты OpenGL
- ▶ `glGenTextures`, `glDeleteTextures`, `glBindTexture`

Что такое текстуры в OpenGL?

- ▶ Объекты OpenGL
- ▶ `glGenTextures`, `glDeleteTextures`, `glBindTexture`
- ▶ `target` - тип текстуры (об этом чуть позже)

Что такое текстуры в OpenGL?

- ▶ Объекты OpenGL
- ▶ `glGenTextures`, `glDeleteTextures`, `glBindTexture`
- ▶ `target` - тип текстуры (об этом чуть позже)
- ▶ Если текстура сделана текущей для какого-то `target`, её никогда нельзя сделать текущей для другого `target`
 - ▶ Таким образом, тип текстуры определяется первым вызовом `glBindTexture` для неё

Что такое текстуры в OpenGL?

- ▶ Объекты OpenGL
- ▶ `glGenTextures`, `glDeleteTextures`, `glBindTexture`
- ▶ `target` - тип текстуры (об этом чуть позже)
- ▶ Если текстура сделана текущей для какого-то `target`, её никогда нельзя сделать текущей для другого `target`
 - ▶ Таким образом, тип текстуры определяется первым вызовом `glBindTexture` для неё
- ▶ ID текущей для `target` текстуры хранится не в контексте OpenGL, а в т.н. `texture unit`'ах (о них позже)

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)

Типы текстур

- ▶ GL_TEXTURE_1D - одномерная текстура (полоска пикселей)
- ▶ GL_TEXTURE_2D - двумерная текстура

Типы текстур

- ▶ GL_TEXTURE_1D - одномерная текстура (полоска пикселей)
- ▶ GL_TEXTURE_2D - двумерная текстура
- ▶ GL_TEXTURE_3D - текстура текстура

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)
- ▶ `GL_TEXTURE_2D` - двумерная текстура
- ▶ `GL_TEXTURE_3D` - текстура текстура
- ▶ `GL_TEXTURE_1D_ARRAY` - массив одномерных текстур одинакового размера

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)
- ▶ `GL_TEXTURE_2D` - двумерная текстура
- ▶ `GL_TEXTURE_3D` - текстура текстура
- ▶ `GL_TEXTURE_1D_ARRAY` - массив одномерных текстур одинакового размера
- ▶ `GL_TEXTURE_2D_ARRAY` - массив двумерных текстур одинакового размера

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)
- ▶ `GL_TEXTURE_2D` - двумерная текстура
- ▶ `GL_TEXTURE_3D` - текстура текстура
- ▶ `GL_TEXTURE_1D_ARRAY` - массив одномерных текстур одинакового размера
- ▶ `GL_TEXTURE_2D_ARRAY` - массив двумерных текстур одинакового размера
- ▶ `GL_TEXTURE_CUBE_MAP` - текстуры, хранящиеся как 6 граней куба

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)
- ▶ `GL_TEXTURE_2D` - двумерная текстура
- ▶ `GL_TEXTURE_3D` - текстура текстура
- ▶ `GL_TEXTURE_1D_ARRAY` - массив одномерных текстур одинакового размера
- ▶ `GL_TEXTURE_2D_ARRAY` - массив двумерных текстур одинакового размера
- ▶ `GL_TEXTURE_CUBE_MAP` - текстуры, хранящиеся как 6 граней куба
- ▶ `GL_TEXTURE_CUBE_MAP_ARRAY` - массив `subemap`'ов (OpenGL 4.0)

Типы текстур

- ▶ GL_TEXTURE_1D - одномерная текстура (полоска пикселей)
- ▶ GL_TEXTURE_2D - двумерная текстура
- ▶ GL_TEXTURE_3D - текстура текстура
- ▶ GL_TEXTURE_1D_ARRAY - массив одномерных текстур одинакового размера
- ▶ GL_TEXTURE_2D_ARRAY - массив двумерных текстур одинакового размера
- ▶ GL_TEXTURE_CUBE_MAP - текстуры, хранящиеся как 6 граней куба
- ▶ GL_TEXTURE_CUBE_MAP_ARRAY - массив cubemap'ов (OpenGL 4.0)
- ▶ GL_TEXTURE_RECTANGLE - прямоугольные текстуры

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)
- ▶ `GL_TEXTURE_2D` - двумерная текстура
- ▶ `GL_TEXTURE_3D` - текстура текстура
- ▶ `GL_TEXTURE_1D_ARRAY` - массив одномерных текстур одинакового размера
- ▶ `GL_TEXTURE_2D_ARRAY` - массив двумерных текстур одинакового размера
- ▶ `GL_TEXTURE_CUBE_MAP` - текстуры, хранящиеся как 6 граней куба
- ▶ `GL_TEXTURE_CUBE_MAP_ARRAY` - массив `subemap`'ов (OpenGL 4.0)
- ▶ `GL_TEXTURE_RECTANGLE` - прямоугольные текстуры (плохое название, другие текстуры тоже могут быть прямоугольными; в современном OpenGL бесполезны)

Типы текстур

- ▶ `GL_TEXTURE_1D` - одномерная текстура (полоска пикселей)
- ▶ `GL_TEXTURE_2D` - двумерная текстура
- ▶ `GL_TEXTURE_3D` - текстура текстура
- ▶ `GL_TEXTURE_1D_ARRAY` - массив одномерных текстур одинакового размера
- ▶ `GL_TEXTURE_2D_ARRAY` - массив двумерных текстур одинакового размера
- ▶ `GL_TEXTURE_CUBE_MAP` - текстуры, хранящиеся как 6 граней куба
- ▶ `GL_TEXTURE_CUBE_MAP_ARRAY` - массив cubemap'ов (OpenGL 4.0)
- ▶ `GL_TEXTURE_RECTANGLE` - прямоугольные текстуры (плохое название, другие текстуры тоже могут быть прямоугольными; в современном OpenGL бесполезны)
- ▶ `GL_TEXTURE_BUFFER` - текстуры, берущие данные из `buffer object`'а

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ `sampler1D/isampler1D/usampler1D`

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ `sampler1D/isampler1D/usampler1D`
- ▶ `sampler2D/isampler2D/usampler2D`

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ `sampler1D/isampler1D/usampler1D`
- ▶ `sampler2D/isampler2D/usampler2D`
- ▶ `sampler3D/isampler3D/usampler3D`

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ `sampler1D/isampler1D/usampler1D`
- ▶ `sampler2D/isampler2D/usampler2D`
- ▶ `sampler3D/isampler3D/usampler3D`
- ▶ `sampler1DArray/isampler1DArray/usampler1DArray`

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ sampler1D/isampler1D/usampler1D
- ▶ sampler2D/isampler2D/usampler2D
- ▶ sampler3D/isampler3D/usampler3D
- ▶ sampler1DArray/isampler1DArray/usampler1DArray
- ▶ sampler2DArray/isampler2DArray/usampler2DArray

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ sampler1D/isampler1D/usampler1D
- ▶ sampler2D/isampler2D/usampler2D
- ▶ sampler3D/isampler3D/usampler3D
- ▶ sampler1DArray/isampler1DArray/usampler1DArray
- ▶ sampler2DArray/isampler2DArray/usampler2DArray
- ▶ samplerCube/isamplerCube/usamplerCube

Текстуры в шейдере - samplers

- ▶ В шейдере текстура представлена uniform-переменной с особым типом, соответствующим типу текстуры и формату пикселей (floating-point / integer / unsigned integer)
- ▶ sampler1D/isampler1D/usampler1D
- ▶ sampler2D/isampler2D/usampler2D
- ▶ sampler3D/isampler3D/usampler3D
- ▶ sampler1DArray/isampler1DArray/usampler1DArray
- ▶ sampler2DArray/isampler2DArray/usampler2DArray
- ▶ samplerCube/isamplerCube/usamplerCube
- ▶ samplerBuffer/isamplerBuffer/usamplerBuffer

Чтение из текстур

- ▶ Как читать из текстуры в шейдере?

Чтение из текстур

- ▶ Как читать из текстуры в шейдере?
- ▶ `texture(sampler, coords)` - возвращает `vec4/ivec4/uvec4`

Чтение из текстур

- ▶ Как читать из текстуры в шейдере?
- ▶ `texture(sampler, coords)` - возвращает `vec4/ivec4/uvec4`
- ▶ Тип и смысл `coords` зависят от типа текстуры

Чтение из текстур

- ▶ Как читать из текстуры в шейдере?
- ▶ `texture(sampler, coords)` - возвращает `vec4/ivec4/uvec4`
- ▶ Тип и смысл `coords` зависят от типа текстуры
 - ▶ Для 1D/2D/3D текстур `float/vec2/vec3`: нормированные координаты (`[0..1]` по каждой оси)

Чтение из текстур

- ▶ Как читать из текстуры в шейдере?
- ▶ `texture(sampler, coords)` - возвращает `vec4/ivec4/uvec4`
- ▶ Тип и смысл `coords` зависят от типа текстуры
 - ▶ Для 1D/2D/3D текстур `float/vec2/vec3`: нормированные координаты (`[0..1]` по каждой оси)
 - ▶ Для 1D/2D array текстур `vec2/vec3`: первые одна/две координаты - нормированные, последняя - номер "слоя"

Чтение из текстур

- ▶ Как читать из текстуры в шейдере?
- ▶ `texture(sampler, coords)` - возвращает `vec4/ivec4/uvec4`
- ▶ Тип и смысл `coords` зависят от типа текстуры
 - ▶ Для 1D/2D/3D текстур `float/vec2/vec3`: нормированные координаты (`[0..1]` по каждой оси)
 - ▶ Для 1D/2D array текстур `vec2/vec3`: первые одна/две координаты - нормированные, последняя - номер "слоя"
 - ▶ Для cubemap текстур `vec3`: вектор направления из центра куба, возвращается значение из пересечения луча с поверхностью куба

Чтение из текстур

- ▶ Текстура хранит набор дискретных пикселей, но обращение к ней происходит по floating-point координатам

Чтение из текстур

- ▶ Текстура хранит набор дискретных пикселей, но обращение к ней происходит по floating-point координатам
- ▶ \Rightarrow режимы фильтрации текстур
 - ▶ GL_NEAREST - использовать ближайший к указанным координатам пиксель
 - ▶ GL_LINEAR - использовать линейную/билинейную/трилинейную интерполяцию между соседними 2/4/8 пикселями (для 1D/2D/3D соответственно)
 - ▶ Для array-текстур номер слоя всегда ближайший!

Чтение из текстур

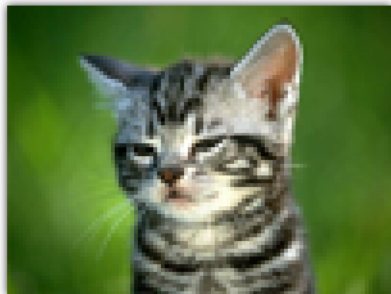
- ▶ Текстура хранит набор дискретных пикселей, но обращение к ней происходит по floating-point координатам
- ▶ \Rightarrow режимы фильтрации текстур
 - ▶ GL_NEAREST - использовать ближайший к указанным координатам пиксель
 - ▶ GL_LINEAR - использовать линейную/билинейную/трилинейную интерполяцию между соседними 2/4/8 пикселями (для 1D/2D/3D соответственно)
 - ▶ Для array-текстур номер слоя всегда ближайший!
- ▶ Настраиваются отдельно для случая, когда
 - ▶ Пиксель текстуры больше пикселя на экране (magnification) - GL_TEXTURE_MAG_FILTER
 - ▶ Пиксель текстуры меньше пикселя на экране (minification) - GL_TEXTURE_MIN_FILTER

Чтение из текстур

- ▶ Текстура хранит набор дискретных пикселей, но обращение к ней происходит по floating-point координатам
- ▶ \Rightarrow режимы фильтрации текстур
 - ▶ GL_NEAREST - использовать ближайший к указанным координатам пиксель
 - ▶ GL_LINEAR - использовать линейную/билинейную/трилинейную интерполяцию между соседними 2/4/8 пикселями (для 1D/2D/3D соответственно)
 - ▶ Для array-текстур номер слоя всегда ближайший!
- ▶ Настраиваются отдельно для случая, когда
 - ▶ Пиксель текстуры больше пикселя на экране (magnification) - GL_TEXTURE_MAG_FILTER
 - ▶ Пиксель текстуры меньше пикселя на экране (minification) - GL_TEXTURE_MIN_FILTER

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  
GL_NEAREST)
```

Nearest vs Linear



GL_NEAREST



GL_LINEAR

Mipmaps

- ▶ Когда пиксель текстуры меньше пикселя на экране (minification), текстура выглядит плохо:
 - ▶ GL_NEAREST - часть пикселей текстуры не попадают на экран
 - ▶ GL_LINEAR - нормально до $\times 2$ уменьшения, потом пиксели тоже не попадают на экран

Mipmaps

- ▶ Когда пиксель текстуры меньше пикселя на экране (minification), текстура выглядит плохо:
 - ▶ GL_NEAREST - часть пикселей текстуры не попадают на экран
 - ▶ GL_LINEAR - нормально до $\times 2$ уменьшения, потом пиксели тоже не попадают на экран
- ▶ Решение: mipmap-уровни

Mipmaps

- ▶ Mipmaps - уменьшенные копии текстуры для использования при минификации

Mipmaps

- ▶ Mipmaps - уменьшенные копии текстуры для использования при минификации
- ▶ Для текстуры $W \times H$ первый уровень имеет размер $\lceil \frac{W}{2} \rceil \times \lceil \frac{H}{2} \rceil$, второй уровень имеет размер $\lceil \frac{W}{4} \rceil \times \lceil \frac{H}{4} \rceil$, и т.д.
- ▶ Для array-текстур отдельный набор mipmap'ов для каждого слоя

Mipmaps

- ▶ Mipmaps - уменьшенные копии текстуры для использования при минификации
- ▶ Для текстуры $W \times H$ первый уровень имеет размер $\lceil \frac{W}{2} \rceil \times \lceil \frac{H}{2} \rceil$, второй уровень имеет размер $\lceil \frac{W}{4} \rceil \times \lceil \frac{H}{4} \rceil$, и т.д.
- ▶ Для array-текстур отдельный набор mipmap'ов для каждого слоя
- ▶ Их можно
 - ▶ Загрузить отдельно, так же, как саму текстуру
 - ▶ Попросить OpenGL сгенерировать на основе самой текстуры
 - ▶ Не использовать

Mipmaps

- ▶ Опции для `GL_TEXTURE_MIN_FILTER`:

Mipmaps

- ▶ Опции для `GL_TEXTURE_MIN_FILTER`:
 - ▶ `GL_NEAREST_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него выбирается ближайший пиксель

Mipmaps

- ▶ Опции для `GL_TEXTURE_MIN_FILTER`:
 - ▶ `GL_NEAREST_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него выбирается ближайший пиксель
 - ▶ `GL_LINEAR_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него делается интерполяция ближайших пикселей

Mipmaps

- ▶ Опции для `GL_TEXTURE_MIN_FILTER`:
 - ▶ `GL_NEAREST_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него выбирается ближайший пиксель
 - ▶ `GL_LINEAR_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него делается интерполяция ближайших пикселей
 - ▶ `GL_NEAREST_MIPMAP_LINEAR` - выбираются два ближайших mipmap уровня, с них выбираются ближайшие пиксели, между ними происходит линейная интерполяция (включено по умолчанию)

Mipmaps

- ▶ Опции для `GL_TEXTURE_MIN_FILTER`:
 - ▶ `GL_NEAREST_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него выбирается ближайший пиксель
 - ▶ `GL_LINEAR_MIPMAP_NEAREST` - выбирается ближайший mipmap уровень, с него делается интерполяция ближайших пикселей
 - ▶ `GL_NEAREST_MIPMAP_LINEAR` - выбираются два ближайших mipmap уровня, с них выбираются ближайшие пиксели, между ними происходит линейная интерполяция (включено по умолчанию)
 - ▶ `GL_LINEAR_MIPMAP_LINEAR` - выбираются два ближайших mipmap уровня, в них делается интерполяция ближайших пикселей, между ними происходит линейная интерполяция

Mipmaps

- ▶ По умолчанию включен `GL_NEAREST_MIPMAP_LINEAR`
- ▶ Если вы забыли сгенерировать/загрузить mipmap-уровни, текстура считается невалидной, и может рисоваться всегда чёрной
- ▶ Не забывайте выставлять правильные min/mag filter и/или генерировать mipmap'ы!

Анизотропная фильтрация

- ▶ Текстура может быть нормального размера по одной оси, и маленькая по другой

Анизотропная фильтрация

- ▶ Текстура может быть нормального размера по одной оси, и маленькая по другой
- ▶ Большой mipmap уровень хорош для одной оси, маленький - для другой

Анизотропная фильтрация

- ▶ Текстура может быть нормального размера по одной оси, и маленькая по другой
- ▶ Большой mipmap уровень хорош для одной оси, маленький - для другой
- ▶ Решение: анизотропная фильтрация

Анизотропная фильтрация

- ▶ Текстура может быть нормального размера по одной оси, и маленькая по другой
- ▶ Большой mipmap уровень хорош для одной оси, маленький - для другой
- ▶ Решение: анизотропная фильтрация
- ▶ Есть в OpenGL 4.6, но почти везде доступна через расширение `EXT_texture_filter_anisotropic`

Wrapping mode

- ▶ При чтении из текстуры координаты могут выходить за пределы диапазона

Wrapping mode

- ▶ При чтении из текстуры координаты могут выходить за пределы диапазона
- ▶ При `GL_LINEAR` для точек у границы текстуры может не найтись достаточное количество соседних пикселей

Wrapping mode

- ▶ При чтении из текстуры координаты могут выходить за пределы диапазона
- ▶ При GL_LINEAR для точек у границы текстуры может не найтись достаточное количество соседних пикселей
- ▶ ⇒ Wrapping mode:
 - ▶ GL_CLAMP_TO_EDGE - брать пиксель с границы
 - ▶ GL_REPEAT (по умолчанию) - повторять текстуру
 - ▶ GL_MIRRORED_REPEAT - повторять текстуру, отражая её на каждый повтор

Wrapping mode

- ▶ При чтении из текстуры координаты могут выходить за пределы диапазона
- ▶ При GL_LINEAR для точек у границы текстуры может не найтись достаточное количество соседних пикселей
- ▶ \Rightarrow Wrapping mode:
 - ▶ GL_CLAMP_TO_EDGE - брать пиксель с границы
 - ▶ GL_REPEAT (по умолчанию) - повторять текстуру
 - ▶ GL_MIRRORED_REPEAT - повторять текстуру, отражая её на каждый повтор
- ▶ Настраивается отдельно для каждой координаты: S, T, R

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,  
                GL_CLAMP_TO_EDGE)
```

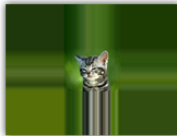

Wrapping



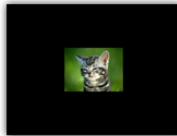
GL_REPEAT



GL_MIRRORED_REPEAT



GL_CLAMP_TO_EDGE



GL_CLAMP_TO_BORDER

Выбор mipmap уровня

- ▶ Как GPU понимает, какой выбрать mipmap-уровень, и происходит ли minification/magnification?

Выбор mipmap уровня

- ▶ Как GPU понимает, какой выбрать mipmap-уровень, и происходит ли minification/magnification?
- ▶ По значениям в соседних пикселях

Выбор mipmap уровня

- ▶ Как GPU понимает, какой выбрать mipmap-уровень, и происходит ли minification/magnification?
- ▶ По значениям в соседних пикселях
- ▶ Фрагментный шейдер всегда запускается на группах 2x2 пикселя (даже если часть из них не были растеризованы)

Выбор mipmap уровня

- ▶ Как GPU понимает, какой выбрать mipmap-уровень, и происходит ли minification/magnification?
- ▶ По значениям в соседних пикселях
- ▶ Фрагментный шейдер всегда запускается на группах 2x2 пикселя (даже если часть из них не были растеризованы)
- ▶ В GLSL есть функции $dFdx/dFdy$ - вычисляют разницу некой величины для пары соседних пикселей в группе 2x2

Выбор mipmap уровня

- ▶ Как GPU понимает, какой выбрать mipmap-уровень, и происходит ли minification/magnification?
- ▶ По значениям в соседних пикселях
- ▶ Фрагментный шейдер всегда запускается на группах 2x2 пикселя (даже если часть из них не были растеризованы)
- ▶ В GLSL есть функции $dFdx/dFdy$ - вычисляют разницу некой величины для пары соседних пикселей в группе 2x2
- ▶ По $dFdx(coords)$ и $dFdy(coords)$ вычисляется mipmap-уровень

Чтение из текстур

- ▶ `textureLod` - обратиться напрямую к указанному mipmap-уровню (LOD = level of detail)

Чтение из текстур

- ▶ `textureLod` - обратиться напрямую к указанному mipmap-уровню (LOD = level of detail)
- ▶ `texelFetch` - обратиться напрямую к указанному пикселю, минуя всю фильтрацию

Формат текстуры

- ▶ Текстура - набор (1D/2D/3D массив или 6 2D массивов) пикселей

Формат текстуры

- ▶ Текстура - набор (1D/2D/3D массив или 6 2D массивов) пикселей
- ▶ Пиксель - 1, 2, 3 или 4 компоненты в определённом формате

Формат текстуры

- ▶ Текстура - набор (1D/2D/3D массив или 6 2D массивов) пикселей
- ▶ Пиксель - 1, 2, 3 или 4 компоненты в определённом формате
- ▶ Форматов очень много, вот часто встречающиеся:
 - ▶ GL_RGB8 - 3 нормированных 8-битных канала
 - ▶ GL_RGBA8 - 4 нормированных 8-битных канала
 - ▶ GL_RGBA32F - 4 32-битных floating-point канала
 - ▶ GL_DEPTH_COMPONENT24 - специальный формат для z-буфера

Загрузить данные в текстуру

- ▶ 1D: `glTexImage1D`

Загрузить данные в текстуру

- ▶ 1D: `glTexImage1D`
- ▶ 2D, 1D array или грань cubemap: `glTexImage2D`

Загрузить данные в текстуру

- ▶ 1D: `glTexImage1D`
- ▶ 2D, 1D array или грань cubemap: `glTexImage2D`
- ▶ 3D или 2D array: `glTexImage3D`

Загрузить данные в текстуру

```
glTexImage2D(GLenum target, GLint level,  
             GLint internalFormat,  
             GLsizei width, GLsizei height, GLint border,  
             GLenum format, GLenum type, const GLvoid * data);
```

- ▶ `target` - тип текстуры (например, `GL_TEXTURE_2D`)
- ▶ `level` - mipmap level, который мы хотим загрузить (0 для основной текстуры)
- ▶ `internalFormat` - формат хранения пикселей (например, `GL_RGBA8`)
- ▶ `width, height` - размер в пикселях
- ▶ `border` - легаси, должно быть 0
- ▶ `format` - какие компоненты есть в массиве `data` (например, `GL_RGB`)
- ▶ `type` - какого типа компоненты в массиве `data` (например, `GL_UNSIGNED_BYTE`)
- ▶ `data` - указатель на массив пикселей (сначала первая строка пикселей, потом вторая, и т.д.)

Загрузить данные в текстуру

- ▶ Если `data = nullptr`, под текстуру выделится память, но данные не будут записаны

Загрузить данные в текстуру

- ▶ Если `data = nullptr`, под текстуру выделится память, но данные не будут записаны
- ▶ Есть ограничения на возможные пары `format` и `type` (например, `type = GL_UNSIGNED_SHORT_5_6_5` требует `format = GL_RGB`)

Загрузить данные в текстуру

- ▶ Если `data = nullptr`, под текстуру выделится память, но данные не будут записаны
- ▶ Есть ограничения на возможные пары `format` и `type` (например, `type = GL_UNSIGNED_SHORT_5_6_5` требует `format = GL_RGB`)
- ▶ `format + type` может не совпадать с `internalFormat`, тогда происходит преобразование в `internalFormat`

Загрузить данные в текстуру

- ▶ Если `data = nullptr`, под текстуру выделится память, но данные не будут записаны
- ▶ Есть ограничения на возможные пары `format` и `type` (например, `type = GL_UNSIGNED_SHORT_5_6_5` требует `format = GL_RGB`)
- ▶ `format + type` может не совпадать с `internalFormat`, тогда происходит преобразование в `internalFormat`
- ▶ По умолчанию ожидается, что начало каждой строки пикселей в массиве `data` выровнено на 4 байта
- ▶ Это может нарушиться, например, с `format = GL_RGB` и `type = GL_UNSIGNED_BYTE` с нечётной шириной текстуры
- ▶ Настраивается с помощью `glPixelStorei(GL_UNPACK_ALIGNMENT, 1)`

Генерація міртар

- ▶ После загрузки нулевого мipmap-уровня, можно сгенерировать все остальные с помощью `glGenerateMipmap(target)`