Компьютерная графика

Лекция 8: Stencil bufer, framebuffer, renderbuffer, пост-обработка

2021

Stencil buffer (буфер трафарета)

- Особый буфер, хранящий произвольные данные, с особыми побитовыми операциями над ними

Stencil buffer (буфер трафарета)

- Особый буфер, хранящий произвольные данные, с особыми побитовыми операциями над ними
- Чем-то аналогичен буферу глубины: тоже хранит какие-то данные, такого же размера (как и цветовой буфер), тоже позволяет рисовать или не рисовать пиксель по какому-то условию (depth test)

Stencil buffer (буфер трафарета)

- Особый буфер, хранящий произвольные данные, с особыми побитовыми операциями над ними
- Чем-то аналогичен буферу глубины: тоже хранит какие-то данные, такого же размера (как и цветовой буфер), тоже позволяет рисовать или не рисовать пиксель по какому-то условию (depth test)
- У дефолтного фреймбуфера часть есть 8-битный stencil буфер (зависит от настроек контекста OpenGL)

Stencil buffer (буфер трафарета)

- Особый буфер, хранящий произвольные данные, с особыми побитовыми операциями над ними
- Чем-то аналогичен буферу глубины: тоже хранит какие-то данные, такого же размера (как и цветовой буфер), тоже позволяет рисовать или не рисовать пиксель по какому-то условию (depth test)
- У дефолтного фреймбуфера часть есть 8-битный stencil буфер (зависит от настроек контекста OpenGL)
- Можно (и нужно, если вы его используете) очищать как `glClear(GL_STENCIL_BUFFER_BIT)`

Stencil buffer (буфер трафарета)

- Особый буфер, хранящий произвольные данные, с особыми побитовыми операциями над ними
- Чем-то аналогичен буферу глубины: тоже хранит какие-то данные, такого же размера (как и цветовой буфер), тоже позволяет рисовать или не рисовать пиксель по какому-то условию (depth test)
- У дефолтного фреймбуфера часть есть 8-битный stencil буфер (зависит от настроек контекста OpenGL)
- Можно (и нужно, если вы его используете) очищать как `glClear(GL_STENCIL_BUFFER_BIT)`
- Настроить значение, которым очищается буфер: `glClearStencil`

Stencil тест

- Включить/выключить stencil тест: `glEnable/glDisable(GL_STENCIL_TEST)`

Stencil тест

- Включить/выключить stencil тест: `glEnable/glDisable(GL_STENCIL_TEST)`
- Настроить stencil тест: `glStencilFunc`
  - `func` - одна из констант `GL_ALWAYS`, `GL_LESS`, `GL_GREATER`, `GL_EQUAL`, ...
  - `ref` - референсное значение для теста
  - `mask` - побитовая маска для теста
- Stencil тест: `func(ref & mask, stencil & mask)`

Stencil тест

- Включить/выключить stencil тест: `glEnable/glDisable(GL_STENCIL_TEST)`
- Настроить stencil тест: `glStencilFunc`
  - `func` - одна из констант `GL_ALWAYS`, `GL_LESS`, `GL_GREATER`, `GL_EQUAL`, ...
  - `ref` - референсное значение для теста
  - `mask` - побитовая маска для теста
- Stencil тест: `func(ref & mask, stencil & mask)`
- Так же, как с depth тестом: если stencil тест не прошёл, пиксель не будет нарисован

Stencil тест

- Как записать значение в stencil буфер?

Stencil тест

- Как записать значение в stencil буфер? `glStencilOp`
  - `sfail` - что делать, если пиксель не прошёл stencil тест
  - `dpfail` - что делать, если пиксель не прошёл depth тест
  - `dppass` - что делать, если пиксель прошёл оба теста

Stencil тест

- Как записать значение в stencil буфер? `glStencilOp`
  - `sfail` - что делать, если пиксель не прошёл stencil тест
  - `dpfail` - что делать, если пиксель не прошёл depth тест
  - `dppass` - что делать, если пиксель прошёл оба теста
- Возможные значение `sfail`, `dpfail` и `dppass`:
  - `GL_KEEP` - не менять записанное значение
  - `GL_ZERO` - записать 0
  - `GL_INVERT` - побитово обратить
  - `GL_REPLACE` - записать `ref` из функции `glStencilFunc`
  - `GL_INCR` - увеличить на 1, если значение меньше максимального
  - `GL_DECR` - уменьшить на 1, если значение больше минимального (0)
  - `GL_INCR_WRAP` - увеличить на 1 с целочисленным переполнением
  - `GL_DECR_WRAP` - уменьшить на 1 с целочисленным переполнением

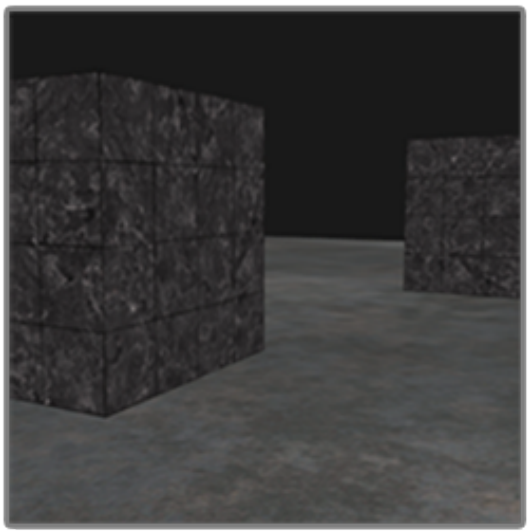
Stencil тест

- Дополнительно можно включать/выключать запись отдельных битов stencil буфера: `glStencilMask`

Stencil тест

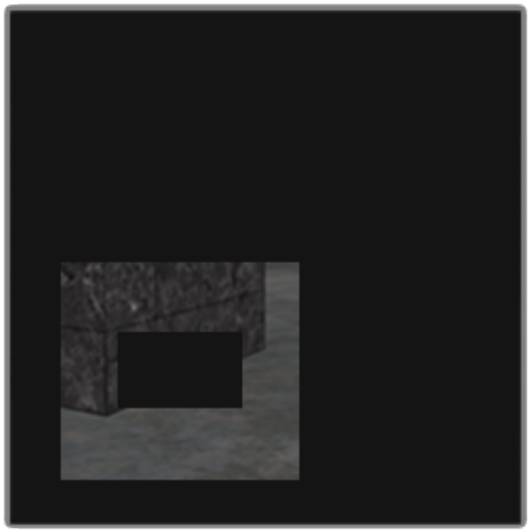
- Дополнительно можно включать/выключать запись отдельных битов stencil буфера: `glStencilMask`
- Все параметры stencil теста можно настраивать отдельно для front и back граней функциями `glStencilFuncSeparate`, `glStencilOpSeparate`, `glStencilMaskSeparate`

Stencil тест



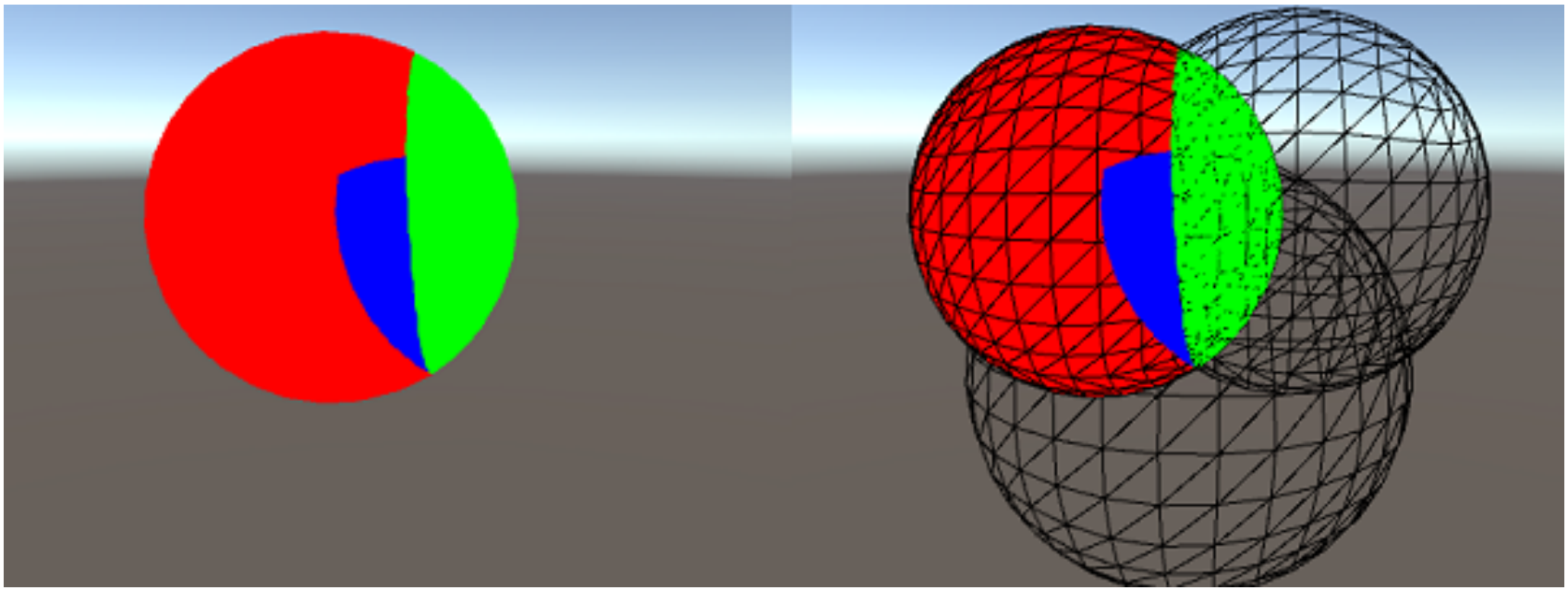
Color buffer

Stencil buffer

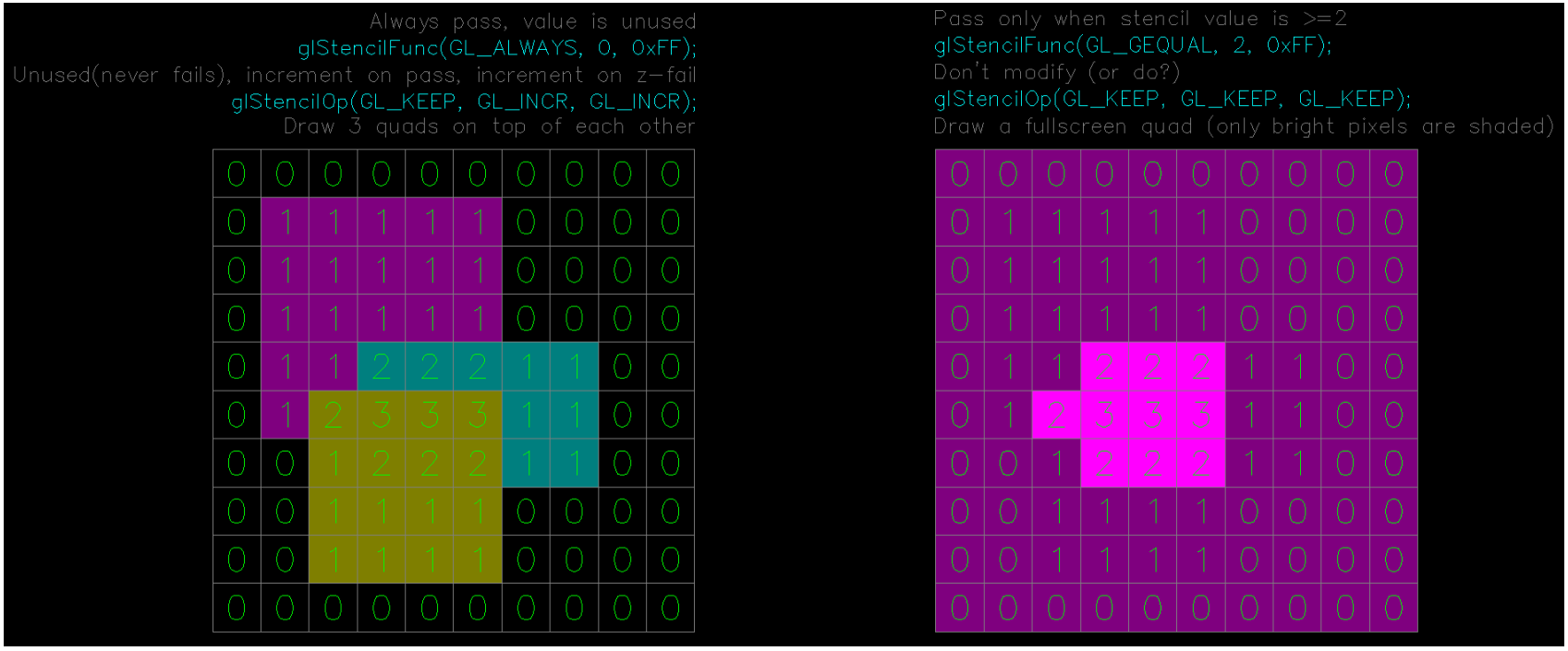


After stencil test

Stencil тест



Stencil тест



Stencil буфер: применение

- Некоторые алгоритмы рисования теней (shadow volumes)

Shadow volumes (stencil shadows)

Stencil буфер: применение

- Некоторые алгоритмы рисования теней (shadow volumes)
- Любая ситуация, в которой нужно ограничить рисование определённых пикселей:

Stencil буфер: применение

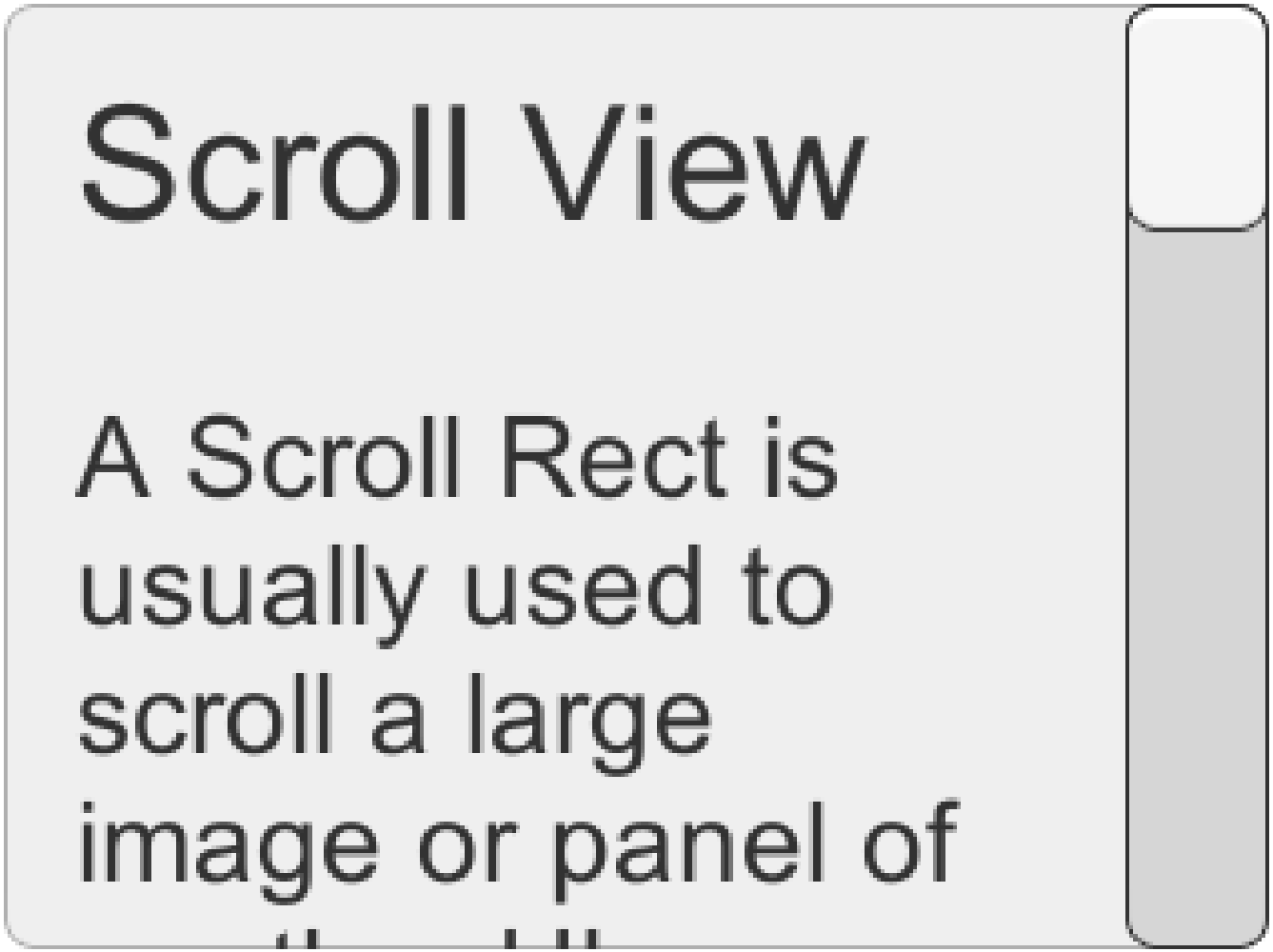
- Некоторые алгоритмы рисования теней (shadow volumes)
- Любая ситуация, в которой нужно ограничить рисование определённых пикселей:
  - Симулятор самолёта: сначала рисуется внутренность самолёта, затем - окружающий мир, только там, где не был нарисован самолёт $\Rightarrow$ можно избежать проблем с точностью буфера глубины

Microsoft flight simulator

Stencil буфер: применение

- Некоторые алгоритмы рисования теней (shadow volumes)
- Любая ситуация, в которой нужно ограничить рисование определённых пикселей:
  - Симулятор самолёта: сначала рисуется внутренность самолёта, затем - окружающий мир, только там, где не был нарисован самолёт $\Rightarrow$ можно избежать проблем с точностью буфера глубины
  - UI, который нужно нарисовать в какой-то ограниченной области экрана (например, scroll)

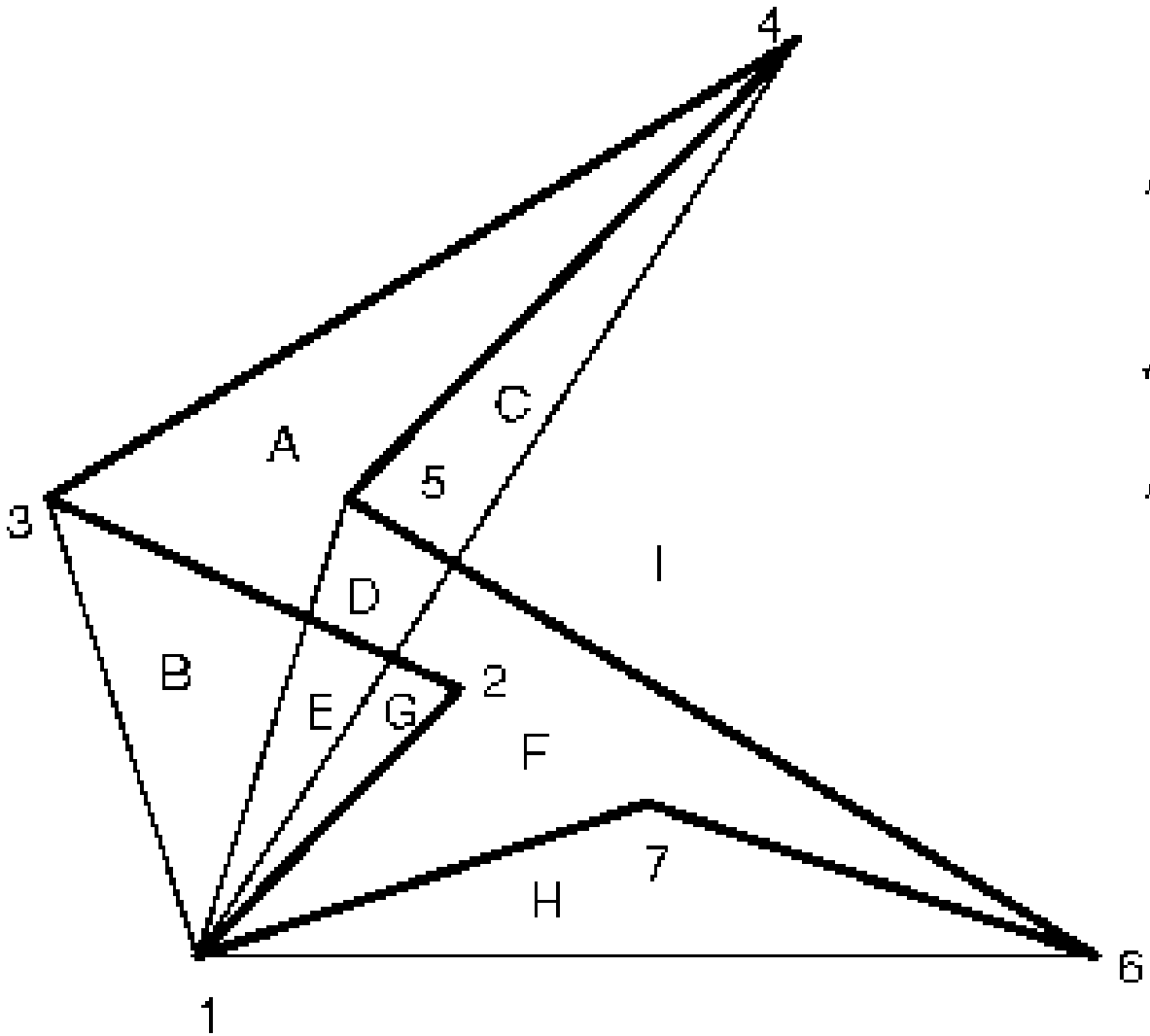
Unity Scroll View



Stencil буфер: применение

- Некоторые алгоритмы рисования теней (shadow volumes)
- Любая ситуация, в которой нужно ограничить рисование определённых пикселей:
  - Симулятор самолёта: сначала рисуется внутренность самолёта, затем - окружающий мир, только там, где не был нарисован самолёт $\Rightarrow$ можно избежать проблем с точностью буфера глубины
  - UI, который нужно нарисовать в какой-то ограниченной области экрана (например, scroll)
  - Рисование невыпуклых полигонов (odd-even rule)

Невыпуклый полигон



^A: 134
B: 123 134
C: 134 145
*D: 134 145 156
E: 123 134 145 156
^F: 156
G: 123 156
H: 156 167
I: (none)

Stencil буфер: ссылки

- khronos.org/opengl/wiki/Stencil_Test
- learnopengl.com/Advanced-OpenGL/Stencil-testing
- open.gl/depthstencils
- en.wikibooks.org/wiki/OpenGL_Programming/Stencil_buffer

Framebuffer (FBO, кадровый буфер)

- OpenGL-объект, хранящий настройки того, куда осуществляется рисование

Framebuffer (FBO, кадровый буфер)

- OpenGL-объект, хранящий настройки того, куда осуществляется рисование
- Не владеет памятью, только ссылается на неё

Framebuffer (FBO, кадровый буфер)

- OpenGL-объект, хранящий настройки того, куда осуществляется рисование
- Не владеет памятью, только ссылается на неё
- Может иметь depth buffer, stencil buffer, и несколько color buffer'ов

Framebuffer (FBO, кадровый буфер)

- OpenGL-объект, хранящий настройки того, куда осуществляется рисование
- Не владеет памятью, только ссылается на неё
- Может иметь depth buffer, stencil buffer, и несколько color buffer'ов
- Общепринятая аббревиатура: FBO

Framebuffer

- Создание/удаление: `glGenFramebuffers/glDeleteFramebuffers`

Framebuffer

- Создание/удаление: `glGenFramebuffers/glDeleteFramebuffers`
- Сделать текущим: `glBindFramebuffer`

Framebuffer

- Создание/удаление: `glGenFramebuffers`/`glDeleteFramebuffers`
- Сделать текущим: `glBindFramebuffer`
- Два значения target:
  - `GL_DRAW_FRAMEBUFFER` - в этот фреймбуфер рисуют все операции рисования (`glDrawArrays`, etc), этот фреймбуфер очищает `glClear`

Framebuffer

- Создание/удаление: `glGenFramebuffers`/`glDeleteFramebuffers`
- Сделать текущим: `glBindFramebuffer`
- Два значения target:
  - `GL_DRAW_FRAMEBUFFER` - в этот фреймбуфер рисуют все операции рисования (`glDrawArrays`, etc), этот фреймбуфер очищает `glClear`
  - `GL_READ_FRAMEBUFFER` - из этого фреймбуфера читают операции чтения

Framebuffer

- Создание/удаление: `glGenFramebuffers/glDeleteFramebuffers`
- Сделать текущим: `glBindFramebuffer`
- Два значения target:
  - `GL_DRAW_FRAMEBUFFER` - в этот фреймбуфер рисуют все операции рисования (`glDrawArrays`, etc), этот фреймбуфер очищает `glClear`
  - `GL_READ_FRAMEBUFFER` - из этого фреймбуфера читают операции чтения
- Можно вызвать `glBindFramebuffer(GL_FRAMEBUFFER, fbo)` - это эквивалентно

```
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, fbo);
glBindFramebuffer(GL_READ_FRAMEBUFFER, fbo);
```

Default framebuffer

- Особый объект, имеет ID = 0

Default framebuffer

- Особый объект, имеет ID = 0
- Создаётся при создании OpenGL-контекста

Default framebuffer

- Особый объект, имеет ID = 0
- Создаётся при создании OpenGL-контекста
- Настроен, чтобы рисовать на экран, к которому привязан контекст

Default framebuffer

- Особый объект, имеет ID = 0
- Создаётся при создании OpenGL-контекста
- Настроен, чтобы рисовать на экран, к которому привязан контекст
- Для него форматы цвета, буфера глубины и stencil буфера настраиваются при создании контекста

Default framebuffer

- Особый объект, имеет ID = 0
- Создаётся при создании OpenGL-контекста
- Настроен, чтобы рисовать на экран, к которому привязан контекст
- Для него форматы цвета, буфера глубины и stencil буфера настраиваются при создании контекста
- Нельзя настроить по-другому или удалить

Чтение из фреймбуфера

- `glReadPixels` - достаёт пиксели из определённого участка текущего `GL_READ_FRAMEBUFFER` (можно прочитать цвет, глубину или stencil)

Чтение из фреймбуфера

- `glReadPixels` - достаёт пиксели из определённого участка текущего `GL_READ_FRAMEBUFFER` (можно прочитать цвет, глубину или stencil)
- `glBlitFramebuffer` - копирует пиксели из определённого участка текущего `GL_READ_FRAMEBUFFER` в определённый участок текущего `GL_DRAW_FRAMEBUFFER` (можно скопировать цвет, глубину или stencil)

Framebuffer attachments

- Текстура, на которую ссылается фреймбуфер, называется attachment

Framebuffer attachments

- Текстура, на которую ссылается фреймбуфер, называется attachment
- У каждого фреймбуфера есть набор attachment points

Framebuffer attachments

- Текстура, на которую ссылается фреймбуфер, называется attachment
- У каждого фреймбуфера есть набор attachment points
- `GL_COLOR_ATTACHMENT0`, ... `GL_COLOR_ATTACHMENT7` - цветовые буферы

Framebuffer attachments

- Текстура, на которую ссылается фреймбуфер, называется attachment
- У каждого фреймбуфера есть набор attachment points
- `GL_COLOR_ATTACHMENT0`, ... `GL_COLOR_ATTACHMENT7` - цветовые буферы
  - Максимальное количество: `glGet(GL_MAX_COLOR_ATTACHMENTS)`
  - Номер (0..7) - то, что мы указываем как location для выходной переменной фрагментного шейдера `layout (location = 0) ...`

Framebuffer attachments

- Текстура, на которую ссылается фреймбуфер, называется attachment
- У каждого фреймбуфера есть набор attachment points
- `GL_COLOR_ATTACHMENT0`, ... `GL_COLOR_ATTACHMENT7` - цветовые буферы
  - Максимальное количество: `glGet(GL_MAX_COLOR_ATTACHMENTS)`
  - Номер (0..7) - то, что мы указываем как location для выходной переменной фрагментного шейдера `layout (location = 0) ...`
- `GL_DEPTH_ATTACHMENT` - буфер глубины
- `GL_STENCIL_ATTACHMENT` - stencil буфер

Framebuffer attachments

- Связать текстуру с фреймбуфером: `glFramebufferTexture`

Framebuffer attachments

- Связать текстуру с фреймбуфером: `glFramebufferTexture`
  - `target` - текстура будет привязана к текущему фреймбуферу с таким таргетом (`GL_READ_FRAMEBUFFER` или `GL_DRAW_FRAMEBUFFER`)

Framebuffer attachments

- Связать текстуру с фреймбуфером: `glFramebufferTexture`
  - `target` - текстура будет привязана к текущему фреймбуферу с таким таргетом (`GL_READ_FRAMEBUFFER` или `GL_DRAW_FRAMEBUFFER`)
  - `attachment` - `GL_COLOR_ATTACHMENT0`, ...

Framebuffer attachments

- Связать текстуру с фреймбуфером: `glFramebufferTexture`
  - `target` - текстура будет привязана к текущему фреймбуферу с таким таргетом (`GL_READ_FRAMEBUFFER` или `GL_DRAW_FRAMEBUFFER`)
  - `attachment` - `GL_COLOR_ATTACHMENT0`, ...
  - `texture` - ID текстуры (делать её текущей не нужно)

Framebuffer attachments

- Связать текстуру с фреймбуфером: `glFramebufferTexture`
  - `target` - текстура будет привязана к текущему фреймбуферу с таким таргетом (`GL_READ_FRAMEBUFFER` или `GL_DRAW_FRAMEBUFFER`)
  - `attachment` - `GL_COLOR_ATTACHMENT0`, ...
  - `texture` - ID текстуры (делать её текущей не нужно)
  - `level` - mipmap-уровень текстуры, в который будет осуществляться рисование

Framebuffer attachments

- Можно привязать несколько текстур к одному фреймбуферу (в т.ч. иметь несколько цветовых буферов - соответственно, несколько `out`-переменных во фрагментном шейдере)

Framebuffer attachments

- Можно привязать несколько текстур к одному фреймбуферу (в т.ч. иметь несколько цветовых буферов - соответственно, несколько `out`-переменных во фрагментном шейдере)
- Можно привязать одну текстуру к нескольким фреймбуферам

Framebuffer attachments

- Можно привязать несколько текстур к одному фреймбуферу (в т.ч. иметь несколько цветовых буферов - соответственно, несколько `out`-переменных во фрагментном шейдере)
- Можно привязать одну текстуру к нескольким фреймбуферам
- Можно привязать несколько разных mipmap-уровней одной текстуры к одному или нескольким фреймбуферам

Framebuffer attachments

- Можно привязать несколько текстур к одному фреймбуферу (в т.ч. иметь несколько цветовых буферов - соответственно, несколько `out`-переменных во фрагментном шейдере)
- Можно привязать одну текстуру к нескольким фреймбуферам
- Можно привязать несколько разных mipmap-уровней одной текстуры к одному или нескольким фреймбуферам
- Можно привязать одномерные и двумерные текстуры, грани cubemap-текстуры (`glFramebufferTexture2D`), слои трёхмерных или 2D-array текстур (`glFramebufferTexture3D` или `glFramebufferTextureLayer`)

Framebuffer completeness

- У фреймбуферов есть особое свойство - completeness: означает, можно ли рисовать в этот фреймбуфер

Framebuffer completeness

- У фреймбуферов есть особое свойство - completeness: означает, можно ли рисовать в этот фреймбуфер
- Фреймбуфер считается complete, если
  - Размеры всех его attachment'ов совпадают

Framebuffer completeness

- У фреймбуферов есть особое свойство - completeness: означает, можно ли рисовать в этот фреймбуфер
- Фреймбуфер считается complete, если
  - Размеры всех его attachment'ов совпадают
  - Все attachment'ы имеют правильный формат

Framebuffer completeness

- У фреймбуферов есть особое свойство - completeness: означает, можно ли рисовать в этот фреймбуфер
- Фреймбуфер считается complete, если
  - Размеры всех его attachment'ов совпадают
  - Все attachment'ы имеют правильный формат
- Проверить completeness - `glCheckFramebufferStatus`: вернёт `GL_FRAMEBUFFER_COMPLETE` или некий код ошибки

Renderable formats

- Что такое *правильный формат*?

Renderable formats

- Что такое *правильный формат*?
- Для цветового буфера - color-renderable формат, т.е. любой цветовой формат: `GL_RED`, `GL_RGB8`, `GL_RGBA8`, `GL_RG32F`, ...

Renderable formats

- Что такое *правильный формат*?
- Для цветового буфера - color-renderable формат, т.е. любой цветовой формат: `GL_RED`, `GL_RGB8`, `GL_RGBA8`, `GL_RG32F`, ...
- Для буфера глубины - depth-renderable формат: `GL_DEPTH_COMPONENT16`, `GL_DEPTH_COMPONENT24`, `GL_DEPTH_COMPONENT32F`, `GL_DEPTH24_STENCIL8`, `GL_DEPTH32F_STENCIL8`

Renderable formats

- Что такое *правильный формат*?
- Для цветового буфера - color-renderable формат, т.е. любой цветовой формат: `GL_RED`, `GL_RGB8`, `GL_RGBA8`, `GL_RG32F`, ...
- Для буфера глубины - depth-renderable формат: `GL_DEPTH_COMPONENT16`, `GL_DEPTH_COMPONENT24`, `GL_DEPTH_COMPONENT32F`, `GL_DEPTH24_STENCIL8`, `GL_DEPTH32F_STENCIL8`
- Для stencil буфера - stencil-renderable формат: `GL_DEPTH24_STENCIL8`, `GL_DEPTH32F_STENCIL8`

Viewport

- `glViewport` настраивает преобразование из координат $[-1, 1]^2$ в пиксельные координаты

Viewport

- `glViewport` настраивает преобразование из координат $[-1, 1]^2$ в пиксельные координаты
- Никак не связан с фреймбуфером

Viewport

- `glViewport` настраивает преобразование из координат $[-1, 1]^2$ в пиксельные координаты
- Никак не связан с фреймбуфером
- Но обычно мы хотим, чтобы viewport совпадал с размером текущего фреймбуфера

Viewport

- `glViewport` настраивает преобразование из координат $[-1, 1]^2$ в пиксельные координаты
- Никак не связан с фреймбуфером
- Но обычно мы хотим, чтобы viewport совпадал с размером текущего фреймбуфера
- $\Rightarrow$ Перед рисованием с помощью фреймбуфера надо подумать о viewport'е

Viewport

- `glViewport` настраивает преобразование из координат $[-1, 1]^2$ в пиксельные координаты
- Никак не связан с фреймбуфером
- Но обычно мы хотим, чтобы viewport совпадал с размером текущего фреймбуфера
- $\Rightarrow$ Перед рисованием с помощью фреймбуфера надо подумать о viewport'е
- N.B. `glClear` игнорирует viewport

Рисование в текстуру: код

```
GLuint fbo;
glGenFramebuffers(1, &fbo);

// создаём текстуру
...

// привязываем текстуру
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, fbo);
glFramebufferTexture(GL_DRAW_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
    fbo_color_texture, 0);

// проверяем completeness
if (glCheckFramebufferStatus(GL_DRAW_FRAMEBUFFER) != GL_FRAMEBUFFER_COMPLETE)
    throw std::runtime_error("Framebuffer incomplete");

...

// Рисование в FBO
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, fbo);
glClear(GL_COLOR_BUFFER_BIT);
glViewport(0, 0, fbo_width, fbo_height);
drawSomething();

// Рисование в дефолтный фреймбуфер
// здесь можно использовать текстуру fbo_color_texture
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, 0);
glClear(GL_COLOR_BUFFER_BIT);
glViewport(0, 0, screen_width, screen_height);
drawSomethingElse();
```

Renderbuffers

- Объекты OpenGL, хранящие пиксели

Renderbuffers

- Объекты OpenGL, хранящие пиксели
- Нельзя загрузить данные, нет режимов фильтрации, нет mipmaps

Renderbuffers

- Объекты OpenGL, хранящие пиксели
- Нельзя загрузить данные, нет режимов фильтрации, нет mipmaps
- Могут быть attachment'ами для фреймбуфера, вместо текстур

Renderbuffers

- Создать/удалить: `glGenRenderbuffers/glDeleteRenderbuffers`

Renderbuffers

- Создать/удалить: `glGenRenderbuffers/glDeleteRenderbuffers`
- Сделать текущим: `glBindRenderbuffer`, target = `GL_RENDERBUFFER`

Renderbuffers

- Создать/удалить: `glGenRenderbuffers/glDeleteRenderbuffers`
- Сделать текущим: `glBindRenderbuffer`, target = `GL_RENDERBUFFER`
- Выделить память: `glRenderbufferStorage`

Renderbuffers

- Создать/удалить: `glGenRenderbuffers/glDeleteRenderbuffers`
- Сделать текущим: `glBindRenderbuffer`, target = `GL_RENDERBUFFER`
- Выделить память: `glRenderbufferStorage`
- Привязать renderbuffer к фреймбуферу: `glFramebufferRenderbuffer`

Framebuffers & renderbuffers: ссылки

- www.khronos.org/opengl/wiki/Framebuffer_Object
- www.khronos.org/opengl/wiki/Renderbuffer_Object
- opengl-tutorial.org/intermediate-tutorials/tutorial-14-render-to-texture
- learnopengl.com/Advanced-OpenGL/Framebuffers