Компьютерная графика

Лекция 6: Ошибки OpenGL, расширения OpenGL, blending, освещение

2021

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур
  - `glTexParameteri` устанавливающая mag filter в что-то отличное от `GL_LINEAR` или `GL_NEAREST`

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур
  - `glTexParameteri` устанавливающая mag filter в что-то отличное от `GL_LINEAR` или `GL_NEAREST`
  - `glBindBuffer` с ID буффера, который не был получен через `glGenBuffers`

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур
  - `glTexParameteri` устанавливающая mag filter в что-то отличное от `GL_LINEAR` или `GL_NEAREST`
  - `glBindBuffer` с ID буффера, который не был получен через `glGenBuffers`
  - etc.

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур
  - `glTexParameteri` устанавливающая mag filter в что-то отличное от `GL_LINEAR` или `GL_NEAREST`
  - `glBindBuffer` с ID буффера, который не был получен через `glGenBuffers`
  - etc.
- В таких случаях генерируется ошибка как особое значение типа `GLenum`

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур
  - `glTexParameteri` устанавливающая mag filter в что-то отличное от `GL_LINEAR` или `GL_NEAREST`
  - `glBindBuffer` с ID буффера, который не был получен через `glGenBuffers`
  - etc.
- В таких случаях генерируется ошибка как особое значение типа `GLenum`
- Как правило, вызвавшая ошибку операция не выполняется

Ошибки OpenGL

- Часто драйвер может понять, что в определённом OpenGL-вызове содержится ошибка
  - `glTexImage2D` с параметром `target`, не являющимся одним из типов текстур
  - `glTexParameteri` устанавливающая mag filter в что-то отличное от `GL_LINEAR` или `GL_NEAREST`
  - `glBindBuffer` с ID буффера, который не был получен через `glGenBuffers`
  - etc.
- В таких случаях генерируется ошибка как особое значение типа `GLenum`
- Как правило, вызвавшая ошибку операция не выполняется
- `glGetError()` - получить значение ошибки, если она была

Ошибки OpenGL

Возможные значения ошибок:

- `GL_NO_ERROR` - ошибки нет

Ошибки OpenGL

Возможные значения ошибок:

- `GL_NO_ERROR` - ошибки нет
- `GL_INVALID_ENUM` - недопустимое значения перечисления (например, `GL_TEXTURE_2D` как первый параметр `glBindBuffer`)

Ошибки OpenGL

Возможные значения ошибок:

- `GL_NO_ERROR` - ошибки нет
- `GL_INVALID_ENUM` - недопустимое значения перечисления (например, `GL_TEXTURE_2D` как первый параметр `glBindBuffer`)
- `GL_INVALID_VALUE` - недопустимое значение числового аргумента (например, отрицательная ширина текстуры в `glTexImage2D`)

Ошибки OpenGL

Возможные значения ошибок:

- `GL_NO_ERROR` - ошибки нет
- `GL_INVALID_ENUM` - недопустимое значения перечисления (например, `GL_TEXTURE_2D` как первый параметр `glBindBuffer`)
- `GL_INVALID_VALUE` - недопустимое значение числового аргумента (например, отрицательная ширина текстуры в `glTexImage2D`)
- `GL_INVALID_OPERATION` - недопустимая операция (например, `glUniform1i` при отсутствии текущей шейдерной программы)

Ошибки OpenGL

Возможные значения ошибок:

- `GL_NO_ERROR` - ошибки нет
- `GL_INVALID_ENUM` - недопустимое значения перечисления (например, `GL_TEXTURE_2D` как первый параметр `glBindBuffer`)
- `GL_INVALID_VALUE` - недопустимое значение числового аргумента (например, отрицательная ширина текстуры в `glTexImage2D`)
- `GL_INVALID_OPERATION` - недопустимая операция (например, `glUniform1i` при отсутствии текущей шейдерной программы)
- `GL_INVALID_FRAMEBUFFER_OPERATION` - операция рисования/чтения пикселей с невалидным framebuffer'ом (о них мы поговорим позже)

Ошибки OpenGL

Возможные значения ошибок:

- `GL_NO_ERROR` - ошибки нет
- `GL_INVALID_ENUM` - недопустимое значения перечисления (например, `GL_TEXTURE_2D` как первый параметр `glBindBuffer`)
- `GL_INVALID_VALUE` - недопустимое значение числового аргумента (например, отрицательная ширина текстуры в `glTexImage2D`)
- `GL_INVALID_OPERATION` - недопустимая операция (например, `glUniform1i` при отсутствии текущей шейдерной программы)
- `GL_INVALID_FRAMEBUFFER_OPERATION` - операция рисования/чтения пикселей с невалидным framebuffer'ом (о них мы поговорим позже)
- `GL_OUT_OF_MEMORY` - закончилась память на GPU (может быть вызвана любой OpenGL-командой, обычно не обрабатывается)

Ошибки OpenGL

- Документация к каждой функции описывает все возможные ошибки, которые эта функция может вызвать, и в каких случаях

Ошибки OpenGL

- Документация к каждой функции описывает все возможные ошибки, которые эта функция может вызвать, и в каких случаях
  - N.B. `GL_OUT_OF_MEMORY` нигде не указан, потому что может появиться где угодно

Ошибки OpenGL

- Документация к каждой функции описывает все возможные ошибки, которые эта функция может вызвать, и в каких случаях
  - N.B. `GL_OUT_OF_MEMORY` нигде не указан, потому что может появиться где угодно
- Не все ошибки покрываются этим механизмом (например: `glDrawArrays` с правильно настроенным VAO но с пустым VBO с вершинами)

Ошибки OpenGL

- Документация к каждой функции описывает все возможные ошибки, которые эта функция может вызвать, и в каких случаях
  - N.B. `GL_OUT_OF_MEMORY` нигде не указан, потому что может появиться где угодно
- Не все ошибки покрываются этим механизмом (например: `glDrawArrays` с правильно настроенным VAO но с пустым VBO с вершинами)
- Нет информации о том, какая именно функция вызвала ошибку (любая из тех, что были вызваны до `glGetError`)

Ошибки OpenGL

- Документация к каждой функции описывает все возможные ошибки, которые эта функция может вызвать, и в каких случаях
  - N.B. `GL_OUT_OF_MEMORY` нигде не указан, потому что может появиться где угодно
- Не все ошибки покрываются этим механизмом (например: `glDrawArrays` с правильно настроенным VAO но с пустым VBO с вершинами)
- Нет информации о том, какая именно функция вызвала ошибку (любая из тех, что были вызваны до `glGetError`)
  - $\Rightarrow$ Придётся расставлять проверку после каждого OpenGL-вызова

Ошибки OpenGL

- Драйвер может хранить один флаг ошибки, и пропускать все последующие ошибки до вызова `glGetError`

Ошибки OpenGL

- Драйвер может хранить один флаг ошибки, и пропускать все последующие ошибки до вызова `glGetError`
- Драйвер может хранить несколько флагов, и `glGetError` возвращает и очищает любой из них

Ошибки OpenGL

- Драйвер может хранить один флаг ошибки, и пропускать все последующие ошибки до вызова `glGetError`
- Драйвер может хранить несколько флагов, и `glGetError` возвращает и очищает любой из них
- $\Rightarrow$ Чтобы очистить ошибки, нужно вызывать `glGetError` в цикле:

```
while (glGetError() != GL_NO_ERROR);
```

Ошибки OpenGL

- OpenGL 4.3+ (или расширение `GL_ARB_debug_output`): более удобный механизм, `glDebugMessageCallback`

Ошибки OpenGL

- OpenGL 4.3+ (или расширение `GL_ARB_debug_output`): более удобный механизм, `glDebugMessageCallback`
- khronos.org/opengl/wiki/OpenGL_Error
- docs.gl/gl3/glGetError

Расширения OpenGL

- Предоставляют дополнительную функциональность за рамками возможностей конкретной версии OpenGL

Расширения OpenGL

- Предоставляют дополнительную функциональность за рамками возможностей конкретной версии OpenGL, например
  - Конкретный производитель выпустил новую функциональность

Расширения OpenGL

- Предоставляют дополнительную функциональность за рамками возможностей конкретной версии OpenGL, например
  - Конкретный производитель выпустил новую функциональность
  - Фунцкиональность доступна на большинстве реализаций OpenGL, но ещё не успела войти в новую версию OpenGL

Расширения OpenGL

- Предоставляют дополнительную функциональность за рамками возможностей конкретной версии OpenGL, например
  - Конкретный производитель выпустил новую функциональность
  - Фунцкиональность доступна на большинстве реализаций OpenGL, но ещё не успела войти в новую версию OpenGL
  - Функциональность доступна в новой версии OpenGL, но крайне распространена и хочется ей пользоваться из старой версии

Расширения OpenGL

- Делятся на
  - ARB (*Architectural Review Board*) - функциональность, которая (скорее всего) войдёт в следующий стандарт
  - EXT - широко распространённая функциональность
  - NV - расширение от Nvidia (возможно, доступно и на других видеокартах)
  - AMD - расширение от AMD (возможно, доступно и на других видеокартах)
  - APPLE - расширение от APPLE (возможно, доступно и на других видеокартах)
  - И т.д.

Расширения OpenGL

- Расширение - набор констант и функций (как и конкретная версия OpenGL)

Расширения OpenGL

- Расширение - набор констант и функций (как и конкретная версия OpenGL)
- Функции нужно загружать, так же, как и функции самого OpenGL

Расширения OpenGL

- Расширение - набор констант и функций (как и конкретная версия OpenGL)
- Функции нужно загружать, так же, как и функции самого OpenGL
- Константы и перечисления заканчиваются на суффикс в зависимости от типа расширения:
  - `ARB_debug_output`: `glDebugMessageCallbackARB`, `GL_DEBUG_SEVERITY_HIGH_ARB`
  - `EXT_texture_filter_anisotropic`: `GL_TEXTURE_MAX_ANISOTROPY_EXT`
  - `NV_texture_barrier`: `glTextureBarrierNV`

Расширения OpenGL

- Расширение - набор констант и функций (как и конкретная версия OpenGL)
- Функции нужно загружать, так же, как и функции самого OpenGL
- Константы и перечисления заканчиваются на суффикс в зависимости от типа расширения:
  - `ARB_debug_output`: `glDebugMessageCallbackARB`, `GL_DEBUG_SEVERITY_HIGH_ARB`
  - `EXT_texture_filter_anisotropic`: `GL_TEXTURE_MAX_ANISOTROPY_EXT`
  - `NV_texture_barrier`: `glTextureBarrierNV`
- Исключение: core extensions - предоставляют функциональность новых версий OpenGL в старых версиях OpenGL
  - Имеют тип `ARB`
  - Не имеют суффиксов в названиях функций и констант

Расширения OpenGL

- Получить список поддерживаемых расширений:

```
GLint numExtensions;
glGetIntegerv(GL_NUM_EXTENSIONS, numExtensions);
for (GLint i = 0; i < numExtensions; ++i) {
    std::cout << glGetStringi(GL_EXTENSIONS, i) << "\n";
}
```

Расширения OpenGL

- Получить список поддерживаемых расширений:

```
GLint numExtensions;
glGetIntegerv(GL_NUM_EXTENSIONS, numExtensions);
for (GLint i = 0; i < numExtensions; ++i) {
    std::cout << glGetStringi(GL_EXTENSIONS, i) << "\n";
}
```

- GLEW загружает их автоматически:

```
if (GLEW_EXT_texture_filter_anisotropic) {
    std::cout << "Anisotropic filtering is supported\n";
}
```

Расширения OpenGL

- khronos.org/opengl/wiki/OpenGL_Extension
- khronos.org/registry/OpenGL/index_gl.php - список всех зарегистрированных расширений

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты

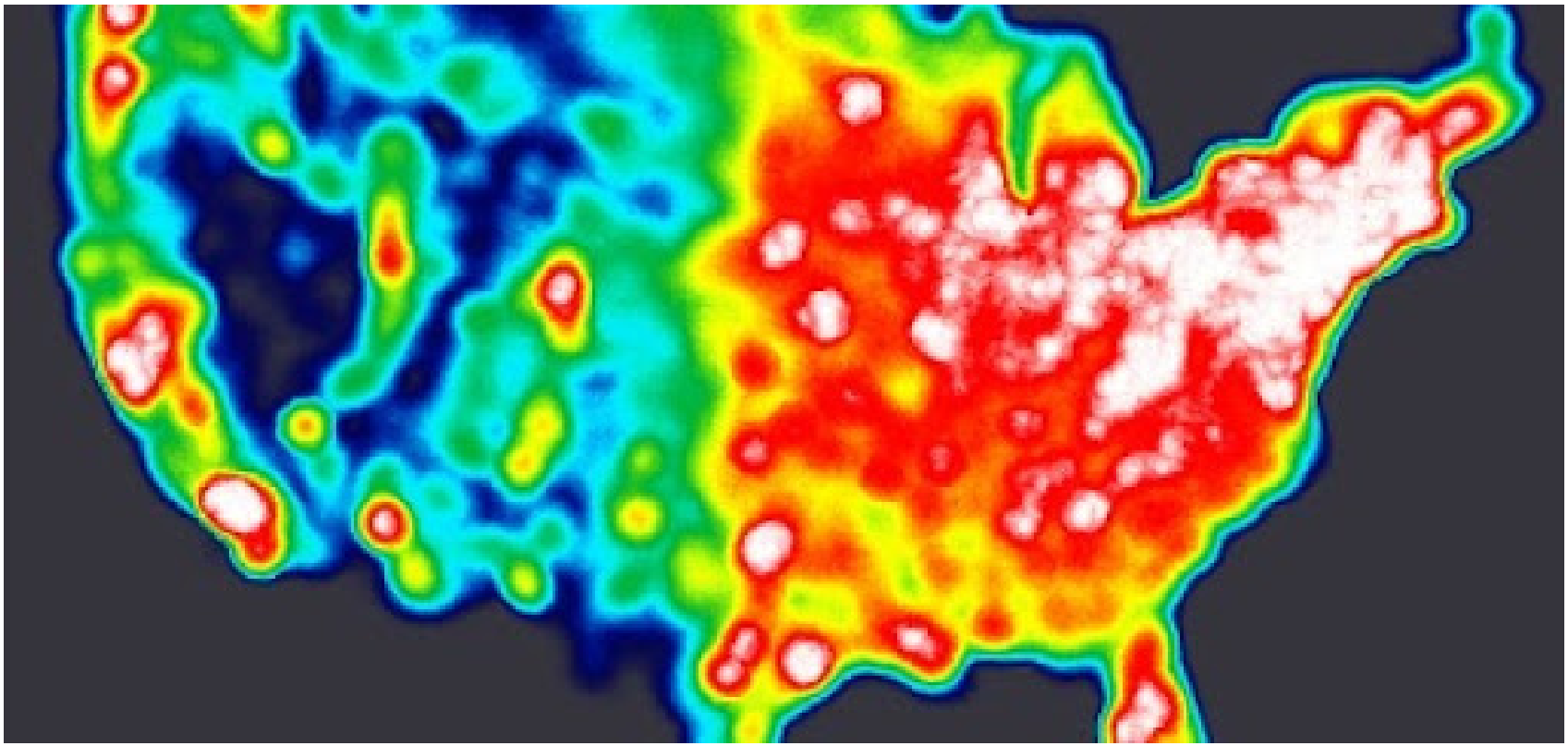
Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты
  - Алгоритмы освещения и теней

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты
  - Алгоритмы освещения и теней
  - Heatmaps

Heatmap



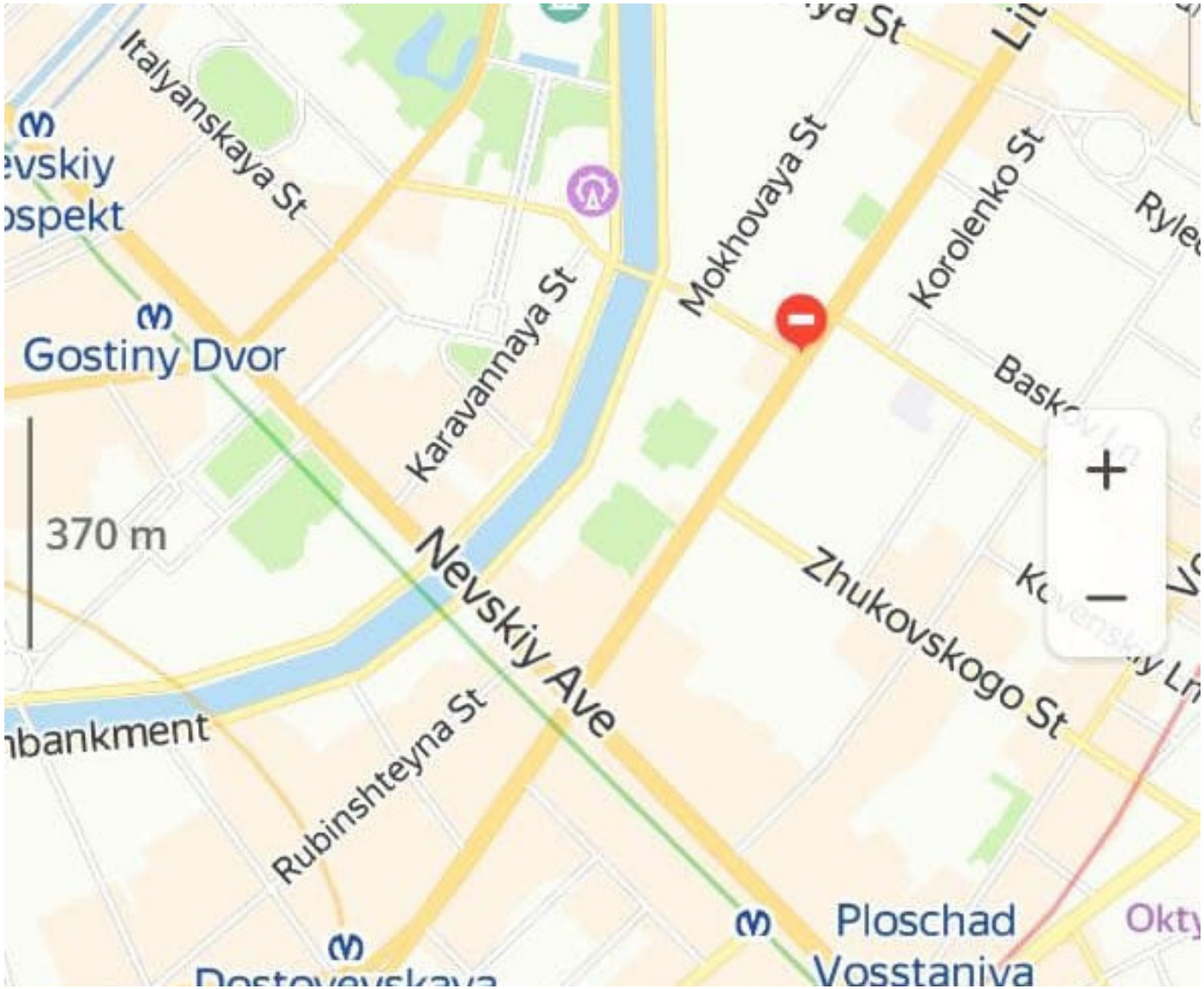
Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты
  - Алгоритмы освещения и теней
  - Heatmaps

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты
  - Алгоритмы освещения и теней
  - Heatmaps
  - Ручное сглаживание

Yandex maps


Italyanskaya St
Gostiny Dvor
Karavannaya St
Mokhovaya St
Korolenko St
370 m
Nevskiy Ave
Zhukovskogo St
Rubinshteyna St
Ploschad
Vosstaniya

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты
  - Алгоритмы освещения и теней
  - Heatmaps
  - Ручное сглаживание

Blending (смешивание)

- Результат фрагментного шейдера (`layout (location = 0) vec4 out_color;`) записывается в пиксель экрана, стирая то, что было в нём до этого
- Иногда мы хотим "смешать"результат и пиксель экрана, и записать результат смешивания
  - Полупрозрачные объекты
  - Алгоритмы освещения и теней
  - Heatmaps
  - Ручное сглаживание
  - И т.д.

Blending (смешивание)

- Включается/выключается: `glEnable(GL_BLEND)`

Blending (смешивание)

- Включается/выключается: `glEnable(GL_BLEND)`
- *src* - результат фрагментного шейдера
- *dst* - пиксель на экране
- Уравнение blending'а:

$$dst \leftarrow f(C_{src} \cdot src, C_{dst} \cdot dst)$$

Blending (смешивание)

- Включается/выключается: `glEnable(GL_BLEND)`
- *src* - результат фрагментного шейдера
- *dst* - пиксель на экране
- Уравнение blending'а:

$$dst \leftarrow f(C_{src} \cdot src, C_{dst} \cdot dst)$$

- f - настраиваемая функция
- C_{src} и C_{dst} - настраиваемые веса

Blending (смешивание)

- Настройка функции f: `glBlendEquation(GLenum equation)`:

Blending (смешивание)

- Настройка функции f: `glBlendEquation(GLenum equation)`:
  - `GL_FUNC_ADD`: $f(src, dst) = src + dst$
  - `GL_FUNC_SUBTRACT`: $f(src, dst) = src - dst$
  - `GL_FUNC_REVERSE_SUBTRACT`: $f(src, dst) = dst - src$
  - `GL_MIN`: $f(src, dst) = \min(src, dst)$
  - `GL_MAX`: $f(src, dst) = \max(src, dst)$

Blending (смешивание)

- Настройка весов C_{src} и C_{dst}:
  `glBlendFunc(GLenum src, GLenum dst)`:

Blending (смешивание)

- Настройка весов C_{src} и C_{dst}:
  `glBlendFunc(GLenum src, GLenum dst)`:
  - `GL_ZERO`: $C = 0$
  - `GL_ONE`: $C = 1$
  - `GL_SRC_COLOR`: $C = src$
  - `GL_ONE_MINUS_SRC_COLOR`: $C = 1 - src$
  - `GL_SRC_ALPHA`: $C = src_A$
  - `GL_ONE_MINUS_SRC_ALPHA`: $C = 1 - src_A$
  - И т.д.

Blending (смешивание)

- Обычно $src = (R, G, B, A)$ означает, что цвет имеет полупрозрачность A
  - $A = 0$ - полностью прозрачный цвет
  - $A = 1$ - полностью непрозрачный цвет

Blending (смешивание)

- Обычно $src = (R, G, B, A)$ означает, что цвет имеет полупрозрачность A
  - $A = 0$ - полностью прозрачный цвет
  - $A = 1$ - полностью непрозрачный цвет
- Для этого нужна такая формула смешивания:

$$dst \leftarrow src_A \cdot src + (1 - src_A) \cdot dst$$

Blending (смешивание)

- Обычно $src = (R, G, B, A)$ означает, что цвет имеет полупрозрачность A
  - $A = 0$ - полностью прозрачный цвет
  - $A = 1$ - полностью непрозрачный цвет
- Для этого нужна такая формула смешивания:

$$dst \leftarrow src_A \cdot src + (1 - src_A) \cdot dst$$

- Этому соответствует настройка

```
glBlendEquation(GL_FUNC_ADD);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Blending (смешивание)

- Обычно $src = (R, G, B, A)$ означает, что цвет имеет полупрозрачность A
  - $A = 0$ - полностью прозрачный цвет
  - $A = 1$ - полностью непрозрачный цвет
- Для этого нужна такая формула смешивания:

$$dst \leftarrow src_A \cdot src + (1 - src_A) \cdot dst$$

- Этому соответствует настройка

```
glBlendEquation(GL_FUNC_ADD);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

- Это самый типичный способ использовать blending

Blending (смешивание)

- khronos.org/opengl/wiki/Blending
- docs.gl/gl3/glBlendFunc
- docs.gl/gl3/glBlendEquation
- opengl-tutorial.org/intermediate-tutorials/tutorial-10-transparency
- learnopengl.com/Advanced-OpenGL/Blending