Компьютерная графика

Лекция 15: рендеринг текста, bitmap-шрифты, векторные шрифты, (M)SDF-шрифты, чем заняться дальше

2023

Рендеринг текста

- Текст – очень сложная штука

Рендеринг текста

- Текст – очень сложная штука
- Слева направо (латиница, кириллица), справа налево (арабский, иврит), сверху вниз (хангыль, традиционное японское и китайское письмо, старомонгольское письмо)

Рендеринг текста

- Текст – очень сложная штука
- Слева направо (латиница, кириллица), справа налево (арабский, иврит), сверху вниз (хангыль, традиционное японское и китайское письмо, старомонгольское письмо)
- Иероглифы (кандзи), слоговое письмо (катакана, хирагана), консонантно-слоговое письмо (индийская, эфиопская письменность), консонантное письмо (арабский), консонантно-вокалическое письмо (латиница, кириллица)

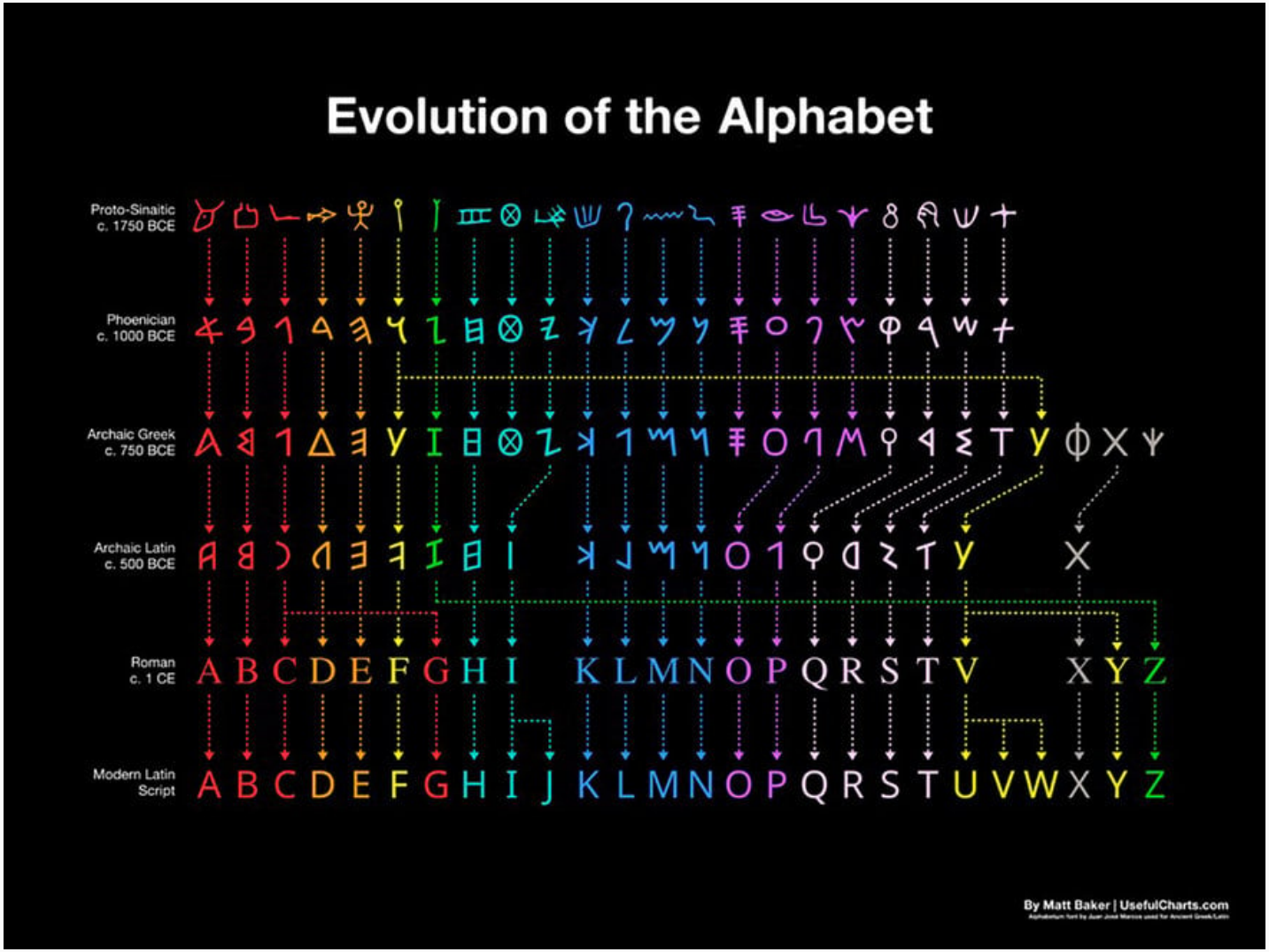
Рендеринг текста

- Текст – очень сложная штука
- Слева направо (латиница, кириллица), справа налево (арабский, иврит), сверху вниз (хангыль, традиционное японское и китайское письмо, старомонгольское письмо)
- Иероглифы (кандзи), слоговое письмо (катакана, хирагана), консонантно-слоговое письмо (индийская, эфиопская письменность), консонантное письмо (арабский), консонантно-вокалическое письмо (латиница, кириллица)
- Одна *графема* может представлять один или несколько звуков/слогов/объектов

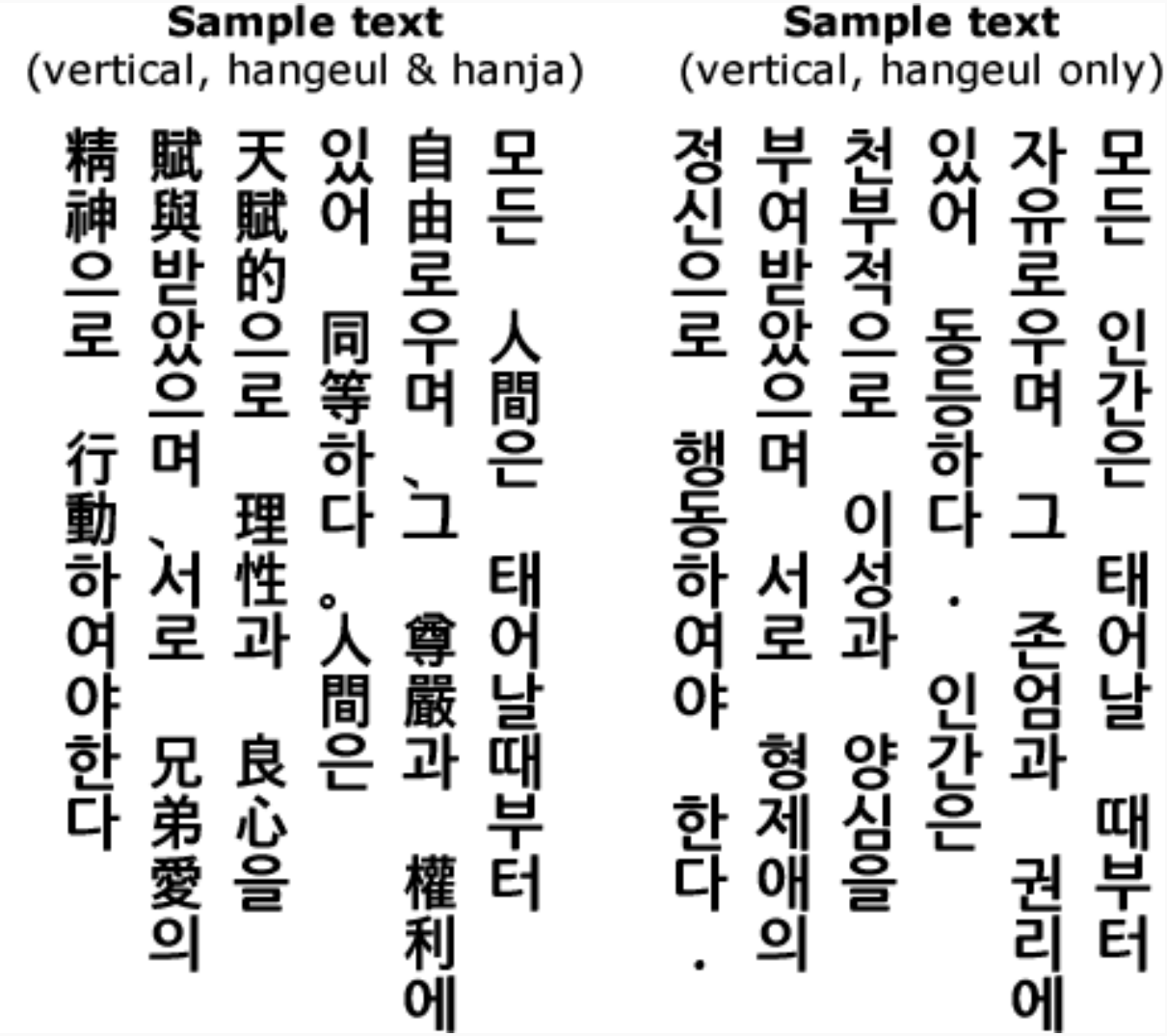
Рендеринг текста

- Текст – очень сложная штука
- Слева направо (латиница, кириллица), справа налево (арабский, иврит), сверху вниз (хангыль, традиционное японское и китайское письмо, старомонгольское письмо)
- Иероглифы (кандзи), слоговое письмо (катакана, хирагана), консонантно-слоговое письмо (индийская, эфиопская письменность), консонантное письмо (арабский), консонантно-вокалическое письмо (латиница, кириллица)
- Одна *графема* может представлять один или несколько звуков/слогов/объектов
- Могут быть сложные правила по соединению символов между собой (арабский, лигатуры в латинице), дополнения к символам (диакритика)

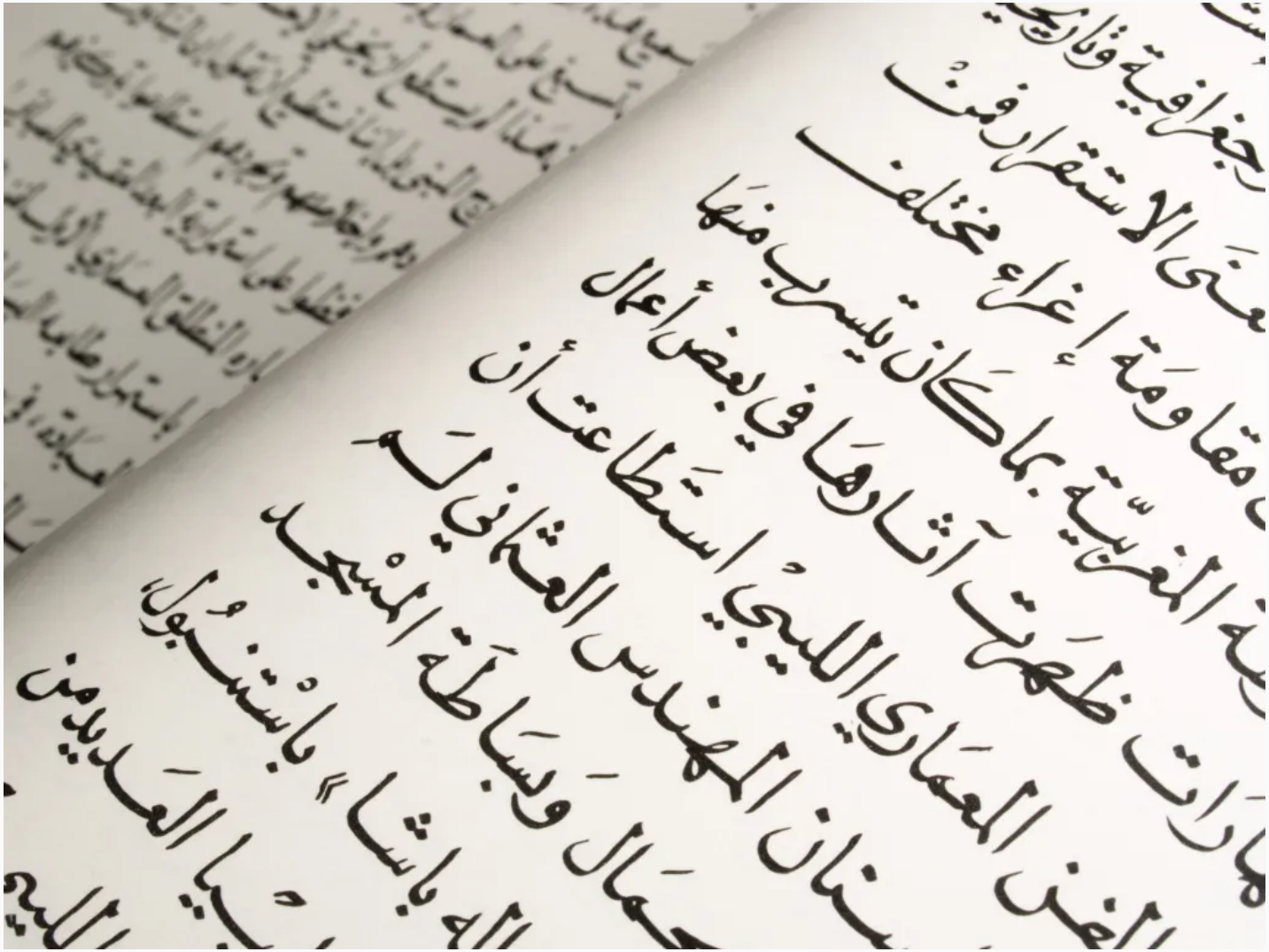
Алфавиты


Evolution of the Alphabet
Proto-Sinaitic
c. 1750 BCE
Phoenician
c. 1000 BCE
Archaic Greek
c. 750 BCE
Archaic Latin
c. 500 BCE
Roman
c. 1 CE
A B C D E F G H I K L M N O P Q R S T V X Y Z
Modern Latin
Script
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
By Matt Baker | UsefulCharts.com

Корейская письменность



Арабская письменность



Этапы рендеринга текста

- Абстрактный текст

Этапы рендеринга текста

- Абстрактный текст
- + *кодировка* $\Longrightarrow$ машинное представление текста

Этапы рендеринга текста

- Абстрактный текст
- + *кодировка* $\Longrightarrow$ машинное представление текста
- + *шрифт* + настройки *шейпинга (shaping)* $\Longrightarrow$ набор глифов (изображений символов) и их координат

Этапы рендеринга текста

- Абстрактный текст
- + *кодировка* $\Longrightarrow$ машинное представление текста
- + *шрифт* + настройки *шейпинга (shaping)* $\Longrightarrow$ набор глифов (изображений символов) и их координат
- + *алгоритм* рендеринга $\Longrightarrow$ нарисованный текст

Кодировки

- Описывают машинное представление текста, т.е. соответствие последовательностей символов и последовательностей бит

Кодировки: ASCII

- 7 бит (обычно дополняется нулевым старшим битом до 8 бит), первые 32 символа - управляющие (`\r`, `\n`, `\t`, ...), остальные 96 – буквы английского алфавита (большие и маленькие) и прочие символы (различные скобки, арифметические операции, пунктуация, пробел, ...)

Кодировки: ASCII

- 7 бит (обычно дополняется нулевым старшим битом до 8 бит), первые 32 символа - управляющие (`\r`, `\n`, `\t`, ...), остальные 96 – буквы английского алфавита (большие и маленькие) и прочие символы (различные скобки, арифметические операции, пунктуация, пробел, ...)
- Многие кодировки совпадают с ASCII в диапазоне 0-127 или 32-127

Кодировки: 8-битные

- **ISO/IEC 8859** – 15 разных вариантов (ISO/IEC 8859-5 для русского языка)

Кодировки: 8-битные

- **ISO/IEC 8859** – 15 разных вариантов (ISO/IEC 8859-5 для русского языка)
- **Code page XXX** – много разных кодировок для DOS (Code page 866 для русского языка)

Кодировки: 8-битные

- **ISO/IEC 8859** – 15 разных вариантов (ISO/IEC 8859-5 для русского языка)
- **Code page XXX** – много разных кодировок для DOS (Code page 866 для русского языка)
- **Windows code pages** (Windows-1251 для русского языка)

Кодировки: 8-битные

- **ISO/IEC 8859** – 15 разных вариантов (ISO/IEC 8859-5 для русского языка)
- **Code page XXX** – много разных кодировок для DOS (Code page 866 для русского языка)
- **Windows code pages** (Windows-1251 для русского языка)
- **KOI-8** и вариации – для русского языка

Кодировки: 8-битные

- **ISO/IEC 8859** – 15 разных вариантов (ISO/IEC 8859-5 для русского языка)
- **Code page XXX** – много разных кодировок для DOS (Code page 866 для русского языка)
- **Windows code pages** (Windows-1251 для русского языка)
- **KOI-8** и вариации – для русского языка
- И т.д.

Кодировки: unicode

- **Unicode** – стандарт, описывающий соответствие абстрактных символов целочисленным кодам (*code points*) в диапазоне `0x0..0x10FFFFh` исключая `0xD800h..0xDFFFh` для суррогатных пар в UTF-16 (итого 1112064 code point'a), и рекомендации по их интерпретации и визуализации

Кодировки: unicode

- **Unicode** – стандарт, описывающий соответствие абстрактных символов целочисленным кодам (*code points*) в диапазоне `0x0..0x10FFFFh` исключая `0xD800h..0xDFFFh` для суррогатных пар в UTF-16 (итого 1112064 code point'a), и рекомендации по их интерпретации и визуализации
- На сегодняшний день описывает 154998 символов (в прошлом году было 149186)

Кодировки: unicode

- **Unicode** – стандарт, описывающий соответствие абстрактных символов целочисленным кодам (*code points*) в диапазоне `0x0..0x10FFFFh` исключая `0xD800h..0xDFFFh` для суррогатных пар в UTF-16 (итого 1112064 code point'a), и рекомендации по их интерпретации и визуализации
- На сегодняшний день описывает 154998 символов (в прошлом году было 149186)
- Сам unicode – **не кодировка**, а таблица соответствия, но есть основанные на нём кодировки

Кодировки: unicode

- UTF-8: от 1 до 4 байт на символ, совпадает с ASCII в диапазоне `0..7Fh`, самая распространённая сегодня кодировка (95% интернета)

Кодировки: unicode

- UTF-8: от 1 до 4 байт на символ, совпадает с ASCII в диапазоне `0..7Fh`, самая распространённая сегодня кодировка (95% интернета)
- UCS-2: устаревшая, 2 байта на символ, не поддерживает весь unicode

Кодировки: unicode

- **UTF-8**: от 1 до 4 байт на символ, совпадает с ASCII в диапазоне `0..7Fh`, самая распространённая сегодня кодировка (95% интернета)
- **UCS-2**: устаревшая, 2 байта на символ, не поддерживает весь unicode
- **UTF-16**: 2 или 4 байта на символ

Кодировки: unicode

- **UTF-8**: от 1 до 4 байт на символ, совпадает с ASCII в диапазоне `0..7Fh`, самая распространённая сегодня кодировка (95% интернета)
- **UCS-2**: устаревшая, 2 байта на символ, не поддерживает весь unicode
- **UTF-16**: 2 или 4 байта на символ
- **UTF-32**: 4 байта на символ

Кодировки: unicode

- UTF-8: от 1 до 4 байт на символ, совпадает с ASCII в диапазоне `0..7Fh`, самая распространённая сегодня кодировка (95% интернета)
- UCS-2: устаревшая, 2 байта на символ, не поддерживает весь unicode
- UTF-16: 2 или 4 байта на символ
- UTF-32: 4 байта на символ
- GB 18030: специальная кодировка для китайских иероглифов (но тоже поддерживает весь unicode)

Кодировки: unicode

- В кодировках UTF-8 и UTF-16 символ занимает *переменное* количество байт – в этой кодировке сложнее двигаться по тексту на целое число code point'ов

Кодировки: unicode

- В кодировках UTF-8 и UTF-16 символ занимает *переменное* количество байт – в этой кодировке сложнее двигаться по тексту на целое число code point'ов
- UTF-16 часто используется в WinAPI как альтернатива более старым функциям, поддерживавшим только 8-битные кодировки

Кодировки: unicode

- В кодировках UTF-8 и UTF-16 символ занимает *переменное* количество байт – в этой кодировке сложнее двигаться по тексту на целое число code point'ов
- UTF-16 часто используется в WinAPI как альтернатива более старым функциям, поддерживавшим только 8-битные кодировки
- **Моё мнение**: лучше хранить текст в UTF-8, редактировать его в UTF-32, и конвертировать в UTF-16 только для передачи в WinAPI :)

Code points, глифы и графемы

- **Code point** – один unicode элемент (абстрактный символ)

Code points, глифы и графемы

- **Code point** – один unicode элемент (абстрактный символ)
- **Глиф** – одно изображение в шрифте

Code points, глифы и графемы

- **Code point** – один unicode элемент (абстрактный символ)
- **Глиф** – одно изображение в шрифте
- **Графема** – один визуальный символ (один или несколько глифов)

Code points, глифы и графемы

- **Code point** – один unicode элемент (абстрактный символ)
- **Глиф** – одно изображение в шрифте
- **Графема** – один визуальный символ (один или несколько глифов)
- В общем случае один code point не соответствует одному глифу или одной графеме

Code points, глифы и графемы

- **Code point** – один unicode элемент (абстрактный символ)
- **Глиф** – одно изображение в шрифте
- **Графема** – один визуальный символ (один или несколько глифов)
- В общем случае один code point **не соответствует** одному глифу или одной графеме
- Примеры:
  - Два символа `ff` могут быть представлены двумя глифами или одним глифом (лигатурой) ff

Code points, глифы и графемы

- **Code point** – один unicode элемент (абстрактный символ)
- **Глиф** – одно изображение в шрифте
- **Графема** – один визуальный символ (один или несколько глифов)
- В общем случае один code point не соответствует одному глифу или одной графеме
- Примеры:
  - Два символа `ff` могут быть представлены двумя глифами или одним глифом (лигатурой) ff
  - Символ `ò` может быть одним или двумя code point'ами и одним или двумя (`o` + `` ` ``) глифами, но считается одной графемой

Шрифты

- Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования

Шрифты

- Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования
- Виды шрифтов:

Шрифты

- Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования
- Виды шрифтов:
  - **Bitmap-шрифты**: глиф – готовое изображение (bitmap)

Шрифты

- Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования
- Виды шрифтов:
  - **Bitmap-шрифты**: глиф – готовое изображение (bitmap)
  - **Векторные шрифты**: глиф описывается как геометрическая фигура

Шрифты

- Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования
- Виды шрифтов:
  - **Bitmap-шрифты**: глиф – готовое изображение (bitmap)
  - **Векторные шрифты**: глиф описывается как геометрическая фигура
  - **(M)SDF-шрифты**: глиф описывается с помощью *signed distance field* (SDF)

Шрифты

- Современные форматы шрифтов (`.ttf` – TrueType, `.otf` – OpenType) – векторные, описывают границу глифа как набор отрезков и квадратичных кривых Безье

Шрифты

- Современные форматы шрифтов (`.ttf` – TrueType, `.otf` – OpenType) – векторные, описывают границу глифа как набор отрезков и квадратичных кривых Безье
- Bitmap и SDF шрифты часто строятся по векторным шрифтам

Шрифты

- Современные форматы шрифтов (`.ttf` – TrueType, `.otf` – OpenType) – векторные, описывают границу глифа как набор отрезков и квадратичных кривых Безье
- Bitmap и SDF шрифты часто строятся по векторным шрифтам
- `FreeType` – самая распространённая библиотека для чтения векторных шрифтов; умеет растеризовать в bitmap и (с версии 2.11.0, июль 2021) в SDF

Шейпинг (shaping)

- Процесс преобразования последовательности символов в набор отпозиционированных глифов

Шейпинг (shaping)

- Процесс преобразования последовательности символов в набор отпозиционированных глифов
- Может включать в себя:

Шейпинг (shaping)

- Процесс преобразования последовательности символов в набор отпозиционированных глифов
- Может включать в себя:
  - Настройки: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)

Шейпинг (shaping)

- Процесс преобразования последовательности символов в набор отпозиционированных глифов
- Может включать в себя:
  - Настройки: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
  - Hinting: сдвиг глифов, чтобы они были лучше выровнены по пиксельной сетке

Шейпинг (shaping)

- Процесс преобразования последовательности символов в набор отпозиционированных глифов
- Может включать в себя:
  - Настройки: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
  - Hinting: сдвиг глифов, чтобы они были лучше выровнены по пиксельной сетке
  - Kerning: изменение расстояния между соседними глифами для лучшего восприятия

Шейпинг (shaping)

- Процесс преобразования последовательности символов в набор отпозиционированных глифов
- Может включать в себя:
  - Настройки: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
  - Hinting: сдвиг глифов, чтобы они были лучше выровнены по пиксельной сетке
  - Kerning: изменение расстояния между соседними глифами для лучшего восприятия
  - Лигатуры: последовательности несвязанных символов, представленные одним глифом (ff, fi, <=>)

Шейпинг (shaping)

- Шейпинг не занимается расстановкой строк и абзацев, а также двунаправленным (bidirectional) текстом (идущим как слева-направо, так и справа-налево)

Шейпинг (shaping)

- Шейпинг не занимается расстановкой строк и абзацев, а также двунаправленным (bidirectional) текстом (идущим как слева-направо, так и справа-налево)
- Для простых моноширинных шрифтов шейпинг может сводиться к расположению глифов на равных расстояниях друг от друга

Шейпинг (shaping)

- Шейпинг не занимается расстановкой строк и абзацев, а также двунаправленным (bidirectional) текстом (идущим как слева-направо, так и справа-налево)
- Для простых моноширинных шрифтов шейпинг может сводиться к расположению глифов на равных расстояниях друг от друга
- `harfbuzz` – одна из самых распространённых библиотек для шейпинга текста, используется всеми на свете

Шейпинг (shaping)

- Шейпинг не занимается расстановкой строк и абзацев, а также двунаправленным (bidirectional) текстом (идущим как слева-направо, так и справа-налево)
- Для простых моноширинных шрифтов шейпинг может сводиться к расположению глифов на равных расстояниях друг от друга
- `harfbuzz` – одна из самых распространённых библиотек для шейпинга текста, используется всеми на свете
- `FreeType` позволяет сделать шейпинг, но хуже, чем harfbuzz

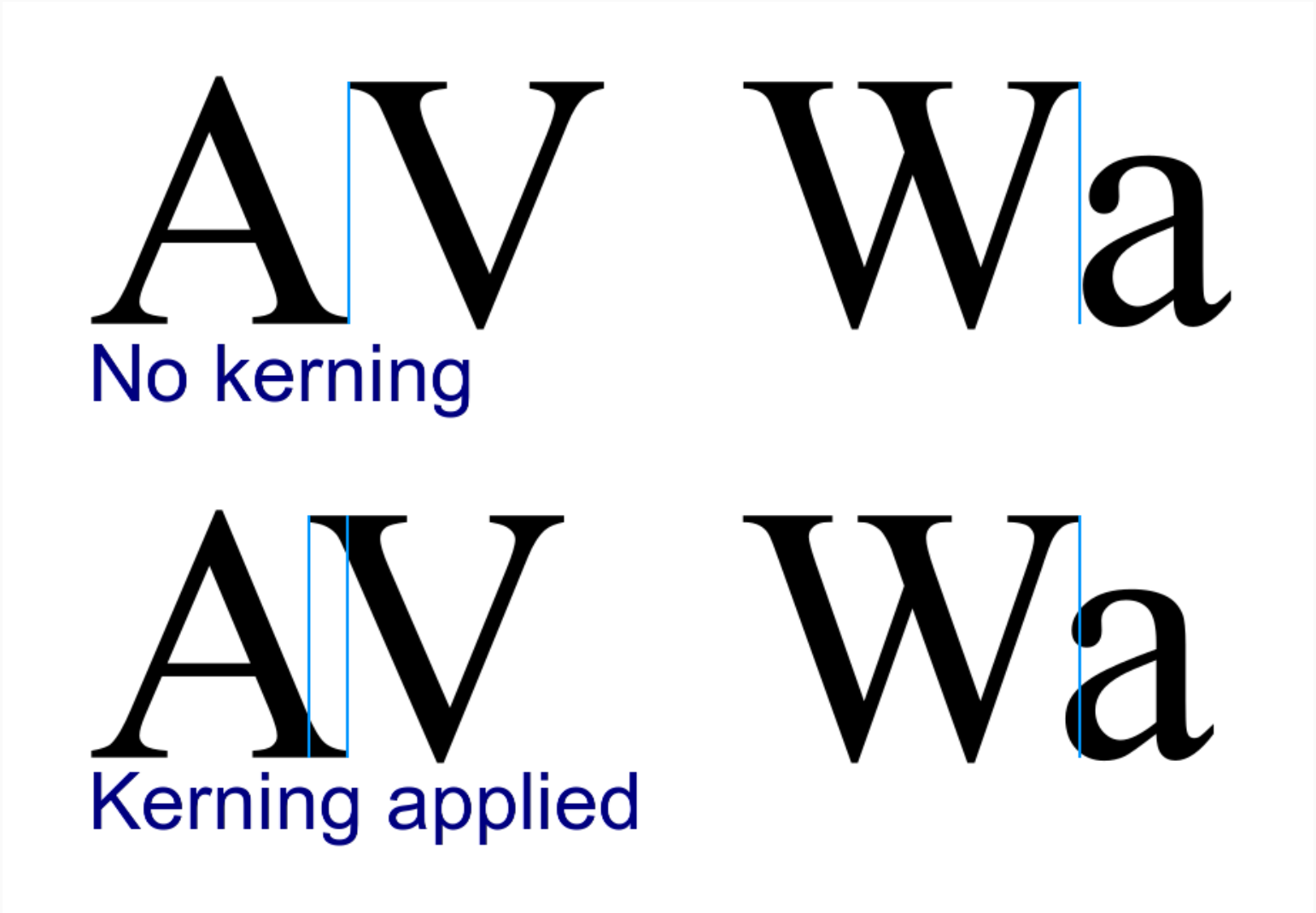
Шейпинг (shaping)

- Шейпинг не занимается расстановкой строк и абзацев, а также двунаправленным (bidirectional) текстом (идущим как слева-направо, так и справа-налево)
- Для простых моноширинных шрифтов шейпинг может сводиться к расположению глифов на равных расстояниях друг от друга
- `harfbuzz` – одна из самых распространённых библиотек для шейпинга текста, используется всеми на свете
- `FreeType` позволяет сделать шейпинг, но хуже, чем harfbuzz
- `FriBidi` – бибилиотека для двунаправленного текста

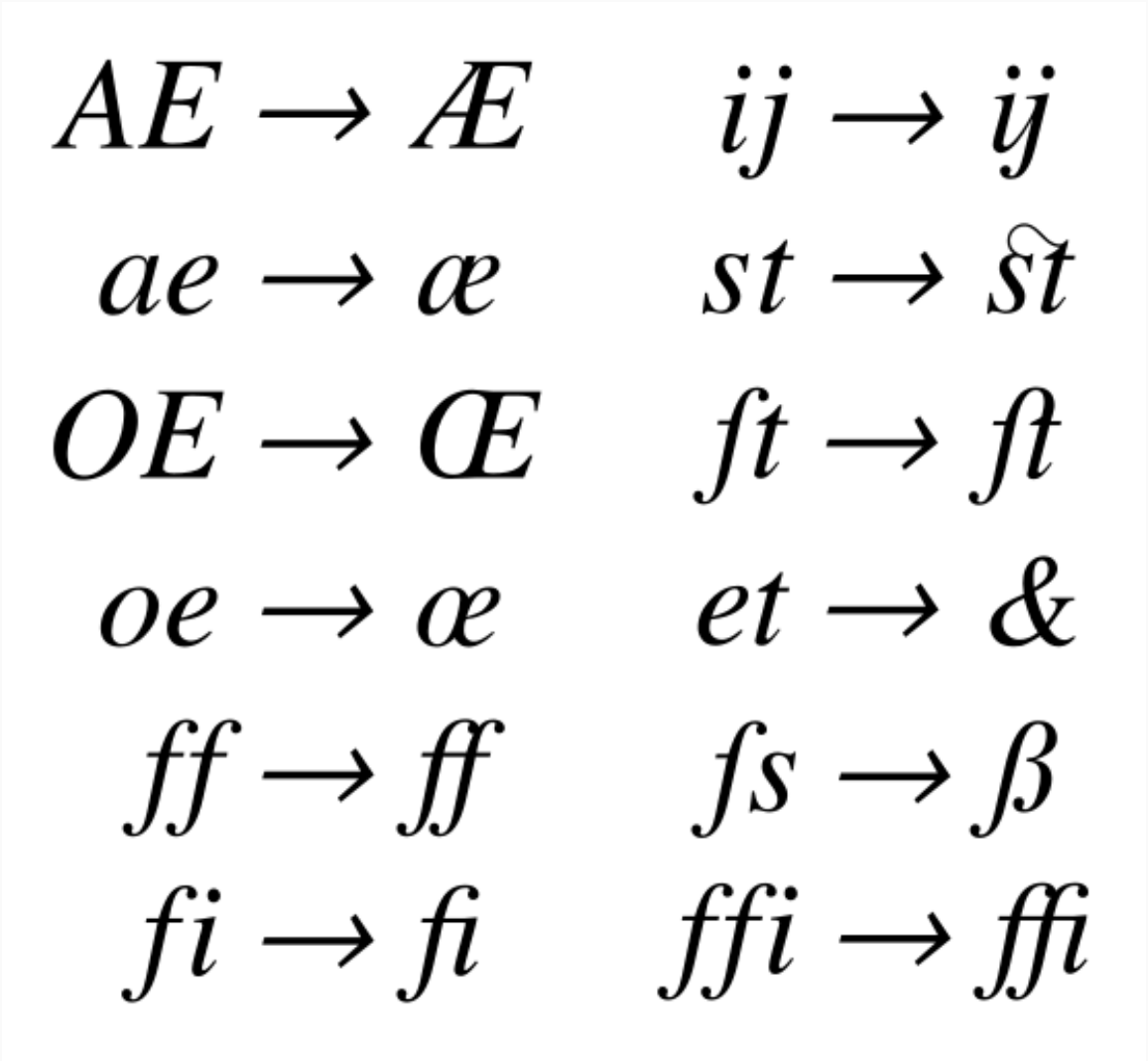
Hinting


abcfgop AO abcfgop
維基百科
維基百科國際
abcfgop AO abcfgop
維基百科
維基百科國際
abcfgop
abcfgop

Kerning



Лигатуры



Рендеринг bitmap-шрифтов

- Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта

Рендеринг bitmap-шрифтов

- Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- Содержат информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)

Рендеринг bitmap-шрифтов

- Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- Содержат информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- Глифы рисуются как текстурированные прямоугольники

Рендеринг bitmap-шрифтов

- Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- Содержат информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- Глифы рисуются как текстурированные прямоугольники
- Плохо ведут себя при масштабировнии (как увеличении, так и уменьшении), mipmap'ы не особо помогают

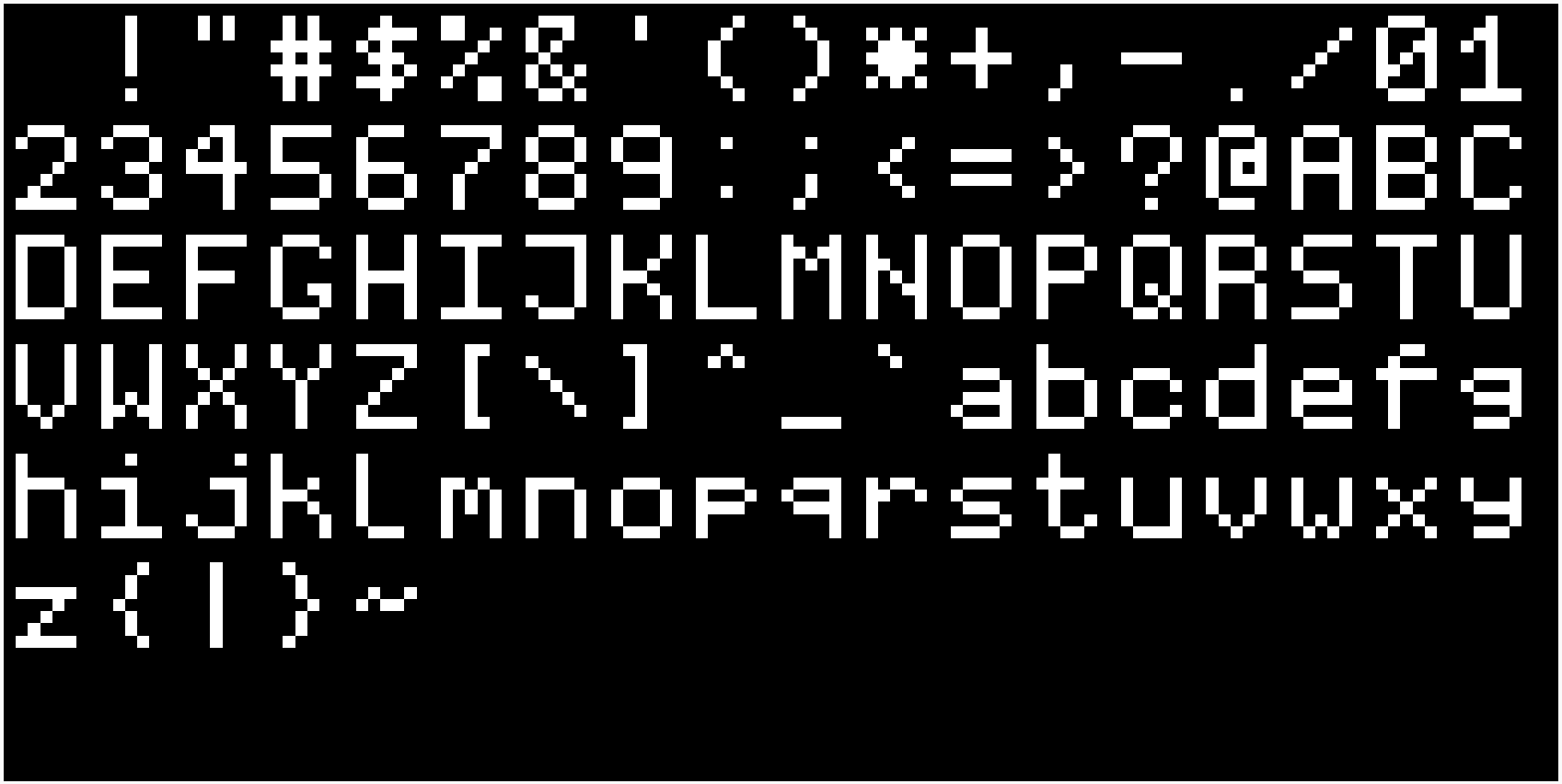
Рендеринг bitmap-шрифтов

- Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- Содержат информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- Глифы рисуются как текстурированные прямоугольники
- Плохо ведут себя при масштабировнии (как увеличении, так и уменьшении), mipmap'ы не особо помогают
- Очень просты в реализации

Рендеринг bitmap-шрифтов

- Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- Содержат информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- Глифы рисуются как текстурированные прямоугольники
- Плохо ведут себя при масштабировнии (как увеличении, так и уменьшении), mipmap'ы не особо помогают
- Очень просты в реализации
- Часто используются для дебажного текста, инди-игр, и т.п.

Bitmap-шрифт



Bitmap-шрифт: описание в коде

```
struct bitmap_font
{
  GLuint texture_id;

  struct glyph
  {
    vec2 top_left;
    vec2 bottom_right;
  };

  std::unordered_map<std::char32_t, glyph> glyphs;
};
```

Bitmap-шрифт: фрагментный шейдер

```
uniform sampler2D font_texture;
uniform vec3 text_color;

in vec2 texcoord;

layout (location = 0) out vec4 out_color;

void main()
{
  float alpha = texture(font_texture, texcoord).r;
  out_color = vec4(text_color, alpha);
}
```

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников
  - Запаковка фигур в текстуру + шейдер, который честно вычисляет площадь пересечения фигуры и пикселя

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников
  - Запаковка фигур в текстуру + шейдер, который честно вычисляет площадь пересечения фигуры и пикселя
  - Полигональная аппроксимация глифа (рисуется с использованием stencil буфера) + треугольник со специальным шейдером для каждой кривой Безье

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников
  - Запаковка фигур в текстуру + шейдер, который честно вычисляет площадь пересечения фигуры и пикселя
  - Полигональная аппроксимация глифа (рисуется с использованием stencil буфера) + треугольник со специальным шейдером для каждой кривой Безье
  - Slug algorithm (запатентован)

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников
  - Запаковка фигур в текстуру + шейдер, который честно вычисляет площадь пересечения фигуры и пикселя
  - Полигональная аппроксимация глифа (рисуется с использованием stencil буфера) + треугольник со специальным шейдером для каждой кривой Безье
  - Slug algorithm (запатентован)
- Обычно легко переносит масштабирование

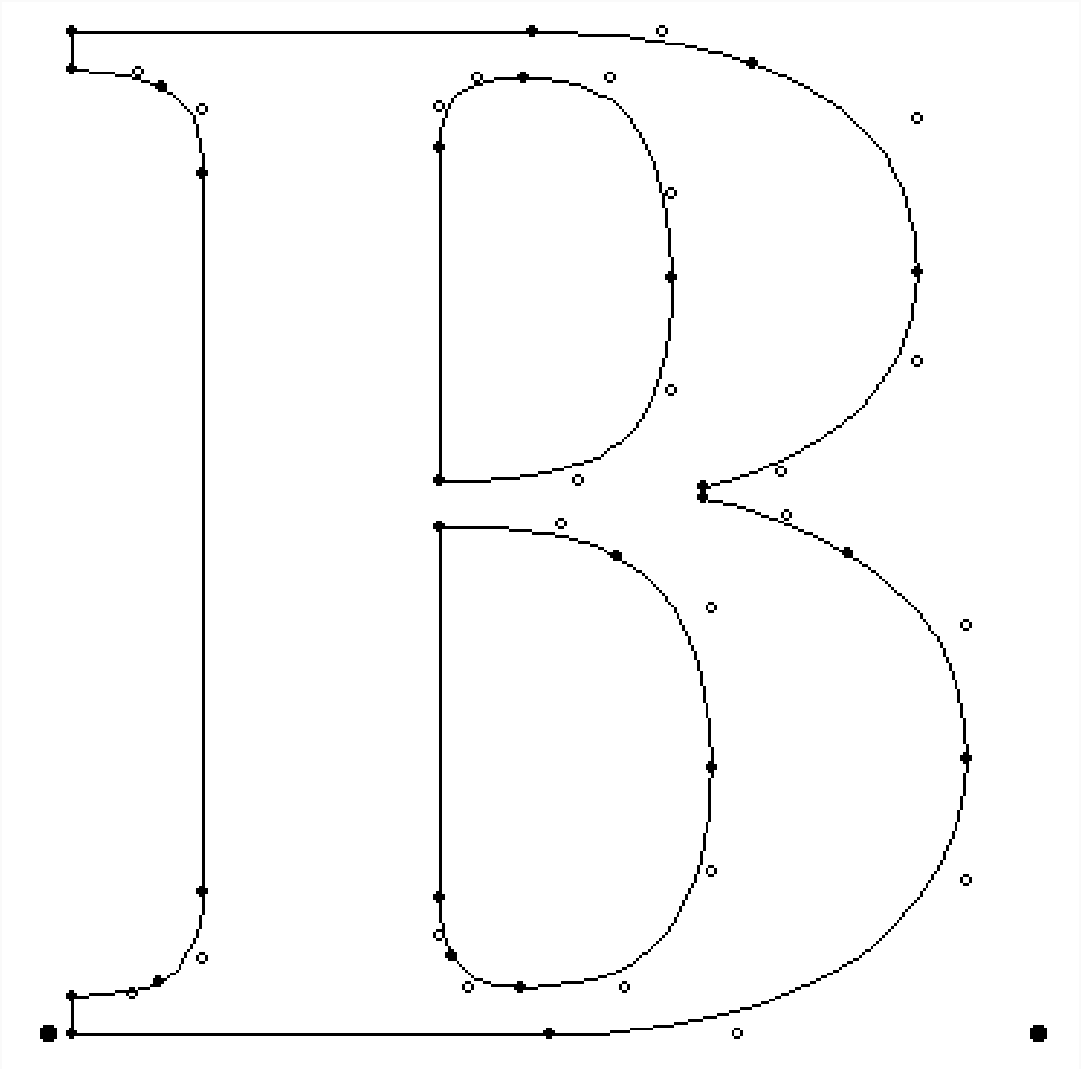
Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников
  - Запаковка фигур в текстуру + шейдер, который честно вычисляет площадь пересечения фигуры и пикселя
  - Полигональная аппроксимация глифа (рисуется с использованием stencil буфера) + треугольник со специальным шейдером для каждой кривой Безье
  - Slug algorithm (запатентован)
- Обычно легко переносит масштабирование
- Сложен в реализации

Рендеринг векторных шрифтов

- Глиф описывается как набор геометрических фигур (фигура может описывать ‘дырку’ в другой фигуре, как дырка в букве ‘O’), граница фигуры – набор отрезков и квадратичных кривых Безье
- Много разных способов рендеринга:
  - Аппроксимация набором треугольников
  - Запаковка фигур в текстуру + шейдер, который честно вычисляет площадь пересечения фигуры и пикселя
  - Полигональная аппроксимация глифа (рисуется с использованием stencil буфера) + треугольник со специальным шейдером для каждой кривой Безье
  - Slug algorithm (запатентован)
- Обычно легко переносит масштабирование
- Сложен в реализации
- Используется для текста максимально возможного качества

Векторный глиф



Slug algorithm

Signed distance field (SDF)

- Описание двумерного или трёхмерного объекта/фигуры *функцией расстояния* до границы объекта

Signed distance field (SDF)

- Описание двумерного или трёхмерного объекта/фигуры *функцией расстояния* до границы объекта
- Обычно положительна снаружи объекта и отрицательна внутри (поэтому *signed*), $f(p) = 0$ – граница объекта

Signed distance field (SDF)

- Описание двумерного или трёхмерного объекта/фигуры *функцией расстояния* до границы объекта
- Обычно положительна снаружи объекта и отрицательна внутри (поэтому *signed*), $f(p) = 0$ – граница объекта
- SDF может быть представлена явной формулой (напр. $f(p) = \|p - O\| - R$ – расстояние до сферы радиуса R с центром в точке O), комбинацией более простых SDF, или текстурой

Signed distance field (SDF)

- Описание двумерного или трёхмерного объекта/фигуры *функцией расстояния* до границы объекта
- Обычно положительна снаружи объекта и отрицательна внутри (поэтому *signed*), $f(p) = 0$ – граница объекта
- SDF может быть представлена явной формулой (напр. $f(p) = \|p - O\| - R$ – расстояние до сферы радиуса R с центром в точке O), комбинацией более простых SDF, или текстурой
- SDF-сцены часто используются для экспериментального рендеринга и удобны для raymarching'а

Signed distance field (SDF)

- SDF-сцены часто используются для экспериментального рендеринга и удобны для raymarching’a

Signed distance field (SDF)

- SDF-сцены часто используются для экспериментального рендеринга и удобны для raymarching'a
- Часто, использующиеся на практике функции не являются SDF, и удовлетворяют более слабым условиям (например, $f(p)$ не больше расстояния до объекта, или $|\nabla f| \leq L$)

Signed distance field (SDF)

- SDF-сцены часто используются для экспериментального рендеринга и удобны для raymarching'a
- Часто, использующиеся на практике функции не являются SDF, и удовлетворяют более слабым условиям (например, $f(p)$ не больше расстояния до объекта, или $|\nabla f| \leq L$)
- Блог Inigo Quilez (`iquilezles.org`) – один из лучших ресурсов про SDF и рендеринг с их помощью

Рендеринг SDF-шрифтов (Valve, 2007)

- Описывается так же, как bitmap-шрифт, но текстура хранит значения SDF для глифов

Рендеринг SDF-шрифтов (Valve, 2007)

- Описывается так же, как bitmap-шрифт, но текстура хранит значения SDF для глифов
- Фрагментный шейдер читает значение SDF из текстуры шрифта: если оно меньше 0, то пиксель находится внутри глифа (e.g. чёрный пиксель), иначе – нет (e.g. прозрачный пиксель)

Рендеринг SDF-шрифтов (Valve, 2007)

- Описывается так же, как bitmap-шрифт, но текстура хранит значения SDF для глифов
- Фрагментный шейдер читает значение SDF из текстуры шрифта: если оно меньше 0, то пиксель находится внутри глифа (e.g. чёрный пиксель), иначе – нет (e.g. прозрачный пиксель)
- Прост в реализации

Рендеринг SDF-шрифтов (Valve, 2007)

- Описывается так же, как bitmap-шрифт, но текстура хранит значения SDF для глифов
- Фрагментный шейдер читает значение SDF из текстуры шрифта: если оно меньше 0, то пиксель находится внутри глифа (e.g. чёрный пиксель), иначе – нет (e.g. прозрачный пиксель)
- Прост в реализации
- Требует чуть больше места под глифы, но менее требователен к разрешению

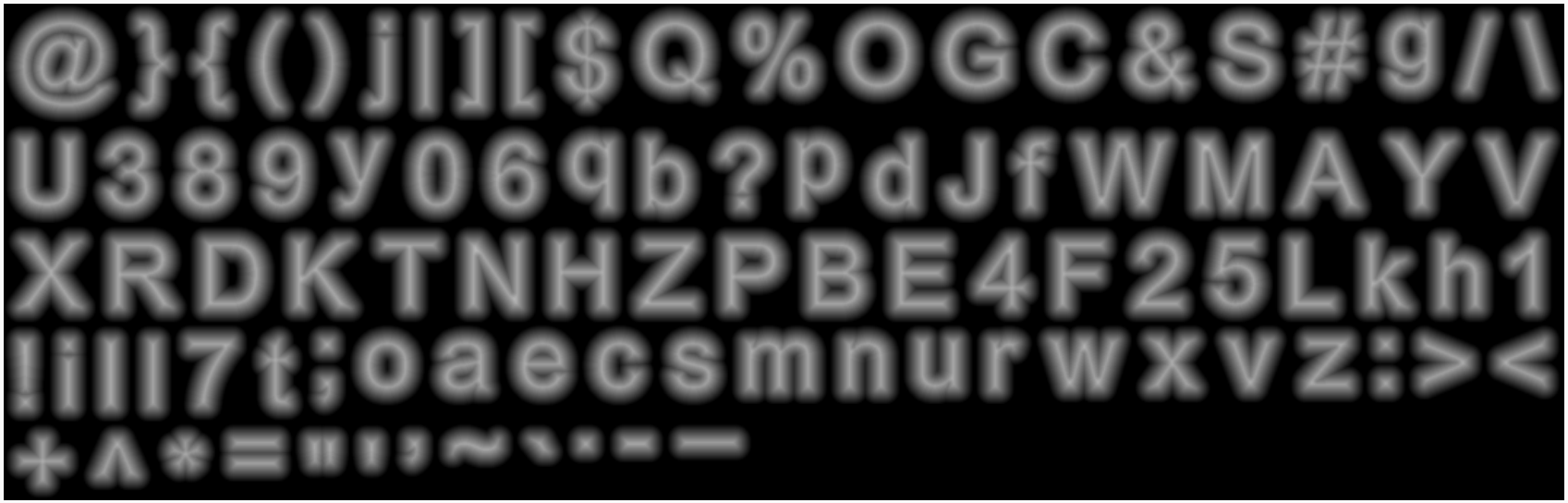
Рендеринг SDF-шрифтов (Valve, 2007)

- Описывается так же, как bitmap-шрифт, но текстура хранит значения SDF для глифов
- Фрагментный шейдер читает значение SDF из текстуры шрифта: если оно меньше 0, то пиксель находится внутри глифа (e.g. чёрный пиксель), иначе – нет (e.g. прозрачный пиксель)
- Прост в реализации
- Требует чуть больше места под глифы, но менее требователен к разрешению
- Неплохо масштабируется (есть артефакты, но менее серьёзные, чем для bitmap-шрифтов)

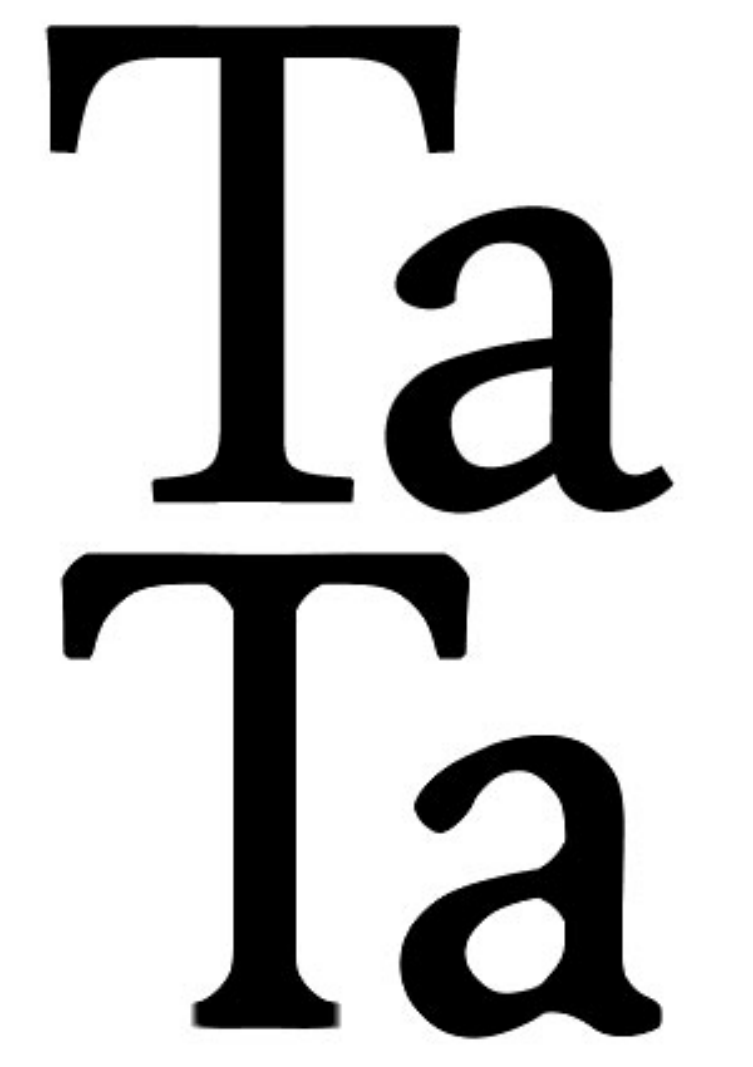
Рендеринг SDF-шрифтов (Valve, 2007)

- Описывается так же, как bitmap-шрифт, но текстура хранит значения SDF для глифов
- Фрагментный шейдер читает значение SDF из текстуры шрифта: если оно меньше 0, то пиксель находится внутри глифа (e.g. чёрный пиксель), иначе – нет (e.g. прозрачный пиксель)
- Прост в реализации
- Требует чуть больше места под глифы, но менее требователен к разрешению
- Неплохо масштабируется (есть артефакты, но менее серьёзные, чем для bitmap-шрифтов)
- Один из самых популярных способов рендеринга шрифтов

SDF-шрифт



SDF-шрифт: артефакты при magnification



SDF-текстура

- Тестуры позволяют хранить значения от 0 до 1

SDF-текстура

- Тестуры позволяют хранить значения от 0 до 1
- Мы хотим хранить произвольные, но не очень большие по модулю числа (расстояние в пикселях)

SDF-текстура

- Тестуры позволяют хранить значения от 0 до 1
- Мы хотим хранить произвольные, но не очень большие по модулю числа (расстояние в пикселях)
- $\Longrightarrow$ В текстуре придётся хранить что-то в духе $0.5 + sdf/scale$, где $scale$ – максимальное представимое расстояние

SDF-текстура

- Тестуры позволяют хранить значения от 0 до 1
- Мы хотим хранить произвольные, но не очень большие по модулю числа (расстояние в пикселях)
- $\Longrightarrow$ В текстуре придётся хранить что-то в духе $0.5 + sdf\,/scale$, где $scale$ – максимальное представимое расстояние
- $\Longrightarrow$ При чтении из текстуры нужно применять обратное преобразование $(sdf - 0.5) \cdot scale$

SDF-текстура

- Тестуры позволяют хранить значения от 0 до 1
- Мы хотим хранить произвольные, но не очень большие по модулю числа (расстояние в пикселях)
- $\Longrightarrow$ В текстуре придётся хранить что-то в духе $0.5 + sdf\,/scale$, где $scale$ – максимальное представимое расстояние
- $\Longrightarrow$ При чтении из текстуры нужно применять обратное преобразование $(sdf - 0.5) \cdot scale$
- Для трёхмерного текста лучше включить для этой текстуры анизотропную фильтрацию, чтобы текст хорошо выглядел ‘сбоку’

Рендеринг SDF-шрифтов

- Можно легко реализовать много дополнительных эффектов:

Рендеринг SDF-шрифтов

- Можно легко реализовать много дополнительных эффектов:
  - Сглаживание: вместо жёсткой границы плавно меняем прозрачность пикселя в зависимости от значения SDF

Рендеринг SDF-шрифтов

- Можно легко реализовать много дополнительных эффектов:
  - Сглаживание: вместо жёсткой границы плавно меняем прозрачность пикселя в зависимости от значения SDF
  - Сглаживание с учётом масштаба и перспективы: по `dFfx, dFfy` можно вычислить диапазон значений SDF, чтобы сглаживание было ровно в 1 пиксель в экранных координатах

Рендеринг SDF-шрифтов

- Можно легко реализовать много дополнительных эффектов:
  - Сглаживание: вместо жёсткой границы плавно меняем прозрачность пикселя в зависимости от значения SDF
  - Сглаживание с учётом масштаба и перспективы: по `dFfx, dFfy` можно вычислить диапазон значений SDF, чтобы сглаживание было ровно в 1 пиксель в экранных координатах
  - Обводка текста другим цветом: рисуем цвет обводки, если $0 \leq f(p) \leq \varepsilon$

Рендеринг SDF-шрифтов

- Можно легко реализовать много дополнительных эффектов:
  - Сглаживание: вместо жёсткой границы плавно меняем прозрачность пикселя в зависимости от значения SDF
  - Сглаживание с учётом масштаба и перспективы: по `dFfx, dFfy` можно вычислить диапазон значений SDF, чтобы сглаживание было ровно в 1 пиксель в экранных координатах
  - Обводка текста другим цветом: рисуем цвет обводки, если $0 \leq f(p) \leq \varepsilon$
  - Псевдотрёхмерный текст: по градиенту SDF можно восстановить нормаль к глифу

SDF-шрифт с эффектами

SDF-шрифт: фрагментный шейдер

```
uniform sampler2D sdfTexture;
uniform float sdfScale;
uniform vec3 textColor;

in vec2 texcoord;

layout (location = 0) out vec4 out_color;

void main()
{
  float textureValue = texture(sdfTexture, texcoord).r;
  float sdfValue = sdfScale * (textureValue - 0.5);
  // сглаживание
  float alpha = smoothstep(-0.5, 0.5, sdfValue);
  out_color = vec4(textColor, alpha);
}
```

MSDF (Chlumský, 2015)

- У SDF-текста есть типичные артефакты: острые углы сглаживаются, из-за чего приходится брать SDF-текстуру большого разрешения

MSDF (Chlumský, 2015)

- У SDF-текста есть типичные артефакты: острые углы сглаживаются, из-за чего приходится брать SDF-текстуру большого разрешения
- Идея: билинейная интерполяция не портит прямые линии $\Longrightarrow$ представим глиф *пересечением* нескольких объектов, чтобы острые углы, представленные пересечением нескольких прямых, не сглаживались

MSDF (Chlumský, 2015)

- У SDF-текста есть типичные артефакты: острые углы сглаживаются, из-за чего приходится брать SDF-текстуру большого разрешения
- Идея: билинейная интерполяция не портит прямые линии $\Longrightarrow$ представим глиф *пересечением* нескольких объектов, чтобы острые углы, представленные пересечением нескольких прямых, не сглаживались
- В текстуре есть 4 канала (RGBA) $\Longrightarrow$ можем сохранить сразу 4 разных SDF в одной текстуре!

MSDF

- **MSDF**: Multi-channel signed distance field

MSDF

- **MSDF**: Multi-channel signed distance field
- На практике оказывается, что хватает 3 каналов, т.е. трёх различных SDF

MSDF

- **MSDF**: Multi-channel signed distance field
- На практике оказывается, что хватает 3 каналов, т.е. трёх различных SDF
- Вместо пересечения (т.е. минимума из трёх значений SDF) лучше брать *медиану* (т.е. значение посередине между двумя другими)

MSDF

- **MSDF**: Multi-channel signed distance field
- На практике оказывается, что хватает 3 каналов, т.е. трёх различных SDF
- Вместо пересечения (т.е. минимума из трёх значений SDF) лучше брать *медиану* (т.е. значение посередине между двумя другими)
- Эффекты на этих трёх каналах уже не реализовать, но для этого можно записать обычную SDF в четвёртый (альфа) канал

MSDF

- **MSDF**: Multi-channel signed distance field
- На практике оказывается, что хватает 3 каналов, т.е. трёх различных SDF
- Вместо пересечения (т.е. минимума из трёх значений SDF) лучше брать *медиану* (т.е. значение посередине между двумя другими)
- Эффекты на этих трёх каналах уже не реализовать, но для этого можно записать обычную SDF в четвёртый (альфа) канал
- Есть инструменты (программа, библиотека, сайт) для генерации таких текстур по шрифту

MSDF-шрифт

MSDF: сравнение с SDF

MSDF: пример кода

```
uniform sampler2D sdfTexture;
uniform float sdfScale;
uniform vec3 textColor;

in vec2 texcoord;

layout (location = 0) out vec4 out_color;

float median(vec3 v) {
    return max(min(v.r, v.g), min(max(v.r, v.g), v.b));
}

void main()
{
  float textureValue =
    median(texture(sdfTexture, texcoord).rgb);
  float sdfValue = sdfScale * (textureValue - 0.5);
  float alpha = smoothstep(-0.5, 0.5, sdfValue);
  out_color = vec4(textColor, alpha);
}
```

Шрифты и шейпинг: ссылки

- `FreeType`
- `harfbuzz`
- `FriBidi`

Векторные и bitmap-шрифты: ссылки

- Туториал по рендерингу bitmap-шрифтов
- Один способ рендеринга векторных шрифтов
- Другой способ рендеринга векторных шрифтов
- Slug algorithm
- Slug library

SDF и MSDF-шрифты: ссылки

- Статья от Valve про SDF-текст
- Туториал по рендерингу SDF-шрифтов
- SDF font generator
- Репозиторий с генератором MSDF-шрифтов от автора этой техники
- Shape Decomposition for Multi-channel Distance Fields (Chlumský, 2015)
- MSDF font generator

Чем заняться дальше?

- Очень много источников света
- Очень много объектов
- Очень большая сцена
- Сложные объекты
- Другие API
- Raytracing
- Статьи и конференции

Очень много источников света

- Deferred shading
- Tiled/Clustered shading
- Compute shaders (OpenGL 4.3 или расширение)
- **Линейные** и **площадные источники света** с помощью linearly transformed cosines (LTC)

Очень много объектов

- Batching
- Frustum culling
- Occlusion culling
- LOD

Очень большая сцена (e.g. планета)

- Иерархический LOD для ландшафта (например, в виде квадродерева)
- Проблемы с точностью (`float` не хватает; нужно рисовать относительно некой anchor-точки в `double`)
- Проблемы с буфером глубины (`reversed z`)

Сложные объекты

- Вода: отражение + преломление (по Френелю) + ‘туман’ или честное рассеивание в плотности воды + волны Герстнера
- Растительность (vegetation): трава, кусты, деревья (см. статья в GPU Gems, доклад про траву в Ghost of Tsushima, доклад про vegetation в Horizon Zero Dawn)
- Облака + небо (volume rendering) (см. доклад про облака в Horizon Zero Dawn)
- Сложные BRDF (см. Crash Course in BRDF Implementation)

Другие API

- OpenGL 4.0+: compute shaders, indirect rendering, direct state access
- Vulkan: `vulkan-tutorial.com`
- WebGPU: `WebGPU C++ Guide`

Raytracing + real-time GI

- Ray Tracing in One Weekend (**не** real-time)
- Vulkan raytracing tutorial
- Voxel cone tracing
- Surflet-based GI
- ReSTIR GI
- Radiance cascades

Статьи и конференции

- GPU Gems 1, 2, 3
- SIGGRAPH (e.g. 2022)
- https://www.gdcvault.com/free/
- Graphics Programming Weekly

В заключение

- Область real-time рендеринга очень активно развивается
- О рисовании любого объекта/эффекта можно найти десятки статей и даже PhD
- Есть тысячи туториалов по всему на свете
- Не бойтесь гуглить и читать непонятные статьи, со временем станет понятнее
- Не бойтесь писать мне :)