

Компьютерная графика

Лекция 14: рендеринг текста, bitmap-шрифты, векторные шрифты, SDF-шрифты, volume rendering, volume slicing, raymarching

2021

Рендеринг текста

- ▶ Абстрактный текст

Рендеринг текста

- ▶ Абстрактный текст
- ▶ + кодировка \Rightarrow машинное представление текста

Рендеринг текста

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

Рендеринг текста

- ▶ Абстрактный текст
- ▶ + кодировка \Rightarrow машинное представление текста
- ▶ + шрифт + настройки шейпинга (shaping) \Rightarrow набор глифов (изображений символов) и их координат
- ▶ \Rightarrow нарисованный текст

Кодировки

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

Кодировки

- ▶ Описывают машинное представление текста, т.е. соответствие последовательностей символов и последовательностей бит
- ▶ ASCII: 7 бит (обычно дополняется нулевым старшим битом до 8 бит), первые 32 символа - управляющие (`\r`, `\n`, `tab`, ...), остальные 96 - буквы английского алфавита (большие и маленькие) и прочие символы (различные скобки, арифметические операции, пунктуация, пробел, ...)

Кодировки

- ▶ Описывают машинное представление текста, т.е. соответствие последовательностей символов и последовательностей бит
- ▶ ASCII: 7 бит (обычно дополняется нулевым старшим битом до 8 бит), первые 32 символа - управляющие (`\r`, `\n`, `tab`, ...), 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

- ▶ Описывают машинное представление текста, т.е. соответствие последовательностей символов и последовательностей бит
- ▶ ASCII: 7 бит (обычно дополняется нулевым старшим битом до 8 бит), первые 32 символа - управляющие (\r, \n, tab, ...), 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
- ▶ Огромное количество в основном 8-битных кодировок для разных алфавитов и систем:
 - ▶ 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
 - ▶ etc.

Кодировки

- ▶ Описывают машинное представление текста, т.е. соответствие последовательностей символов и последовательностей бит
- ▶ ASCII: 7 бит (обычно дополняется нулевым старшим битом до 8 бит), первые 32 символа - управляющие (\r, \n, tab, ...), 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
- ▶ Огромное количество в основном 8-битных кодировок для разных алфавитов и систем:
 - ▶ 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
 - ▶ etc.
- ▶ Unicode-кодировки

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:
 - ▶ UTF-8: от 1 до 4 байт на символ, совпадает с ASCII в диапазоне 0..7Fh, самая распространённая сегодня кодировка (95% интернета)

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:
 - ▶ 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

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:
 - ▶ 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

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:
 - ▶ 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

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:
 - ▶ 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)

Unicode

- ▶ Unicode - стандарт, описывающий соответствие символов целочисленным кодам в диапазоне 0..10FFFFh исключая D800h..DFFFh (используется для суррогатных пар в UTF-16; всего 1112064 символов), и рекомендации по их интерпретации и визуализации
- ▶ На сегодняшний день описывает 144697 символа
- ▶ Unicode-кодировки:
 - ▶ 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)
- ▶ N.B.: один символ unicode **не соответствует** одному видимому символу (*графеме*)

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:
 - ▶ Битмар-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:
 - ▶ Битмап-шрифты: глиф - готовое изображение (bitmap)
 - ▶ Векторные шрифты: глиф описывается как геометрическая фигура
 - ▶ SDF-шрифты: глиф описывается с помощью *signed distance field* (SDF)

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
 - ▶ SDF-шрифты: глиф описывается с помощью *signed distance field* (SDF)
- ▶ Современные форматы шрифтов (.ttf - TrueType, .otf - OpenType) - векторные, описывают границу глифа как набор отрезков и квадратичных кривых Безье (т.е. 2-ого порядка)

Шрифты

- ▶ Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования
- ▶ Виды шрифтов:
 - ▶ Битмар-шрифты: глиф - готовое изображение (bitmap)
 - ▶ Векторные шрифты: глиф описывается как геометрическая фигура
 - ▶ SDF-шрифты: глиф описывается с помощью *signed distance field* (SDF)
- ▶ Современные форматы шрифтов (.ttf - TrueType, .otf - OpenType) - векторные, описывают границу глифа как набор отрезков и квадратичных кривых Безье (т.е. 2-ого порядка)
- ▶ Битмар и SDF шрифты часто строятся по векторным шрифтам

Шрифты

- ▶ Содержит набор *глифов* (изображений символов в каком-либо виде) и правил их использования
- ▶ Виды шрифтов:
 - ▶ Битмар-шрифты: глиф - готовое изображение (bitmap)
 - ▶ Векторные шрифты: глиф описывается как геометрическая фигура
 - ▶ SDF-шрифты: глиф описывается с помощью *signed distance field* (SDF)
- ▶ Современные форматы шрифтов (.ttf - TrueType, .otf - OpenType) - векторные, описывают границу глифа как набор отрезков и квадратичных кривых Безье (т.е. 2-ого порядка)
- ▶ Битмар и SDF шрифты часто строятся по векторным шрифтам
- ▶ FreeType - самая распространённая библиотека для чтения векторных шрифтов; умеет растеризовать в битмар и (с версии 2.11.0, июль 2021) в SDF

Шейпинг

- ▶ Процесс преобразования последовательности символов в набор отпозиционированных глифов

Шейпинг

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

Шейпинг

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

Шейпинг

- ▶ Процесс преобразования последовательности символов в набор отпозиционированных глифов
- ▶ Может включать в себя:
 - ▶ Настройки шейпинга: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
 - ▶ Hinting: применение дополнительных преобразований к векторному глифу в зависимости от разрешения

Шейпинг

- ▶ Процесс преобразования последовательности символов в набор отпозиционированных глифов
- ▶ Может включать в себя:
 - ▶ Настройки шейпинга: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
 - ▶ Hinting: применение дополнительных преобразований к векторному глифу в зависимости от разрешения
 - ▶ Kerning: изменение расстояния между соседними глифами

Шейпинг

- ▶ Процесс преобразования последовательности символов в набор отпозиционированных глифов
- ▶ Может включать в себя:
 - ▶ Настройки шейпинга: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
 - ▶ Hinting: применение дополнительных преобразований к векторному глифу в зависимости от разрешения
 - ▶ Kerning: изменение расстояния между соседними глифами
 - ▶ Лигатуры: последовательность несвязанных символов, представленная одним глифом (ff, fi, <=>)

Шейпинг

- ▶ Процесс преобразования последовательности символов в набор отпозиционированных глифов
- ▶ Может включать в себя:
 - ▶ Настройки шейпинга: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
 - ▶ Hinting: применение дополнительных преобразований к векторному глифу в зависимости от разрешения
 - ▶ Kerning: изменение расстояния между соседними глифами
 - ▶ Лигатуры: последовательность несвязанных символов, представленная одним глифом (ff, fi, <=>)
- ▶ Для простых моноширинных шрифтов шейпинг может сводиться к расположению глифов на равных расстояниях друг от друга

Шейпинг

- ▶ Процесс преобразования последовательности символов в набор отпозиционированных глифов
- ▶ Может включать в себя:
 - ▶ Настройки шейпинга: направление (слева-направо, справа-налево, сверху-вниз, снизу-вверх), размер шрифта, межбуквенное расстояние, стиль (жирный, курсив, и т.п.)
 - ▶ Hinting: применение дополнительных преобразований к векторному глифу в зависимости от разрешения
 - ▶ Kerning: изменение расстояния между соседними глифами
 - ▶ Лигатуры: последовательность несвязанных символов, представленная одним глифом (ff, fi, <=>)
- ▶ 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
- ▶ FreeType позволяет сделать шейпинг, но хуже, чем harfbuzz

abcfgop AO *abcfgop*
abcfgop AO *abcfgop*

維基百科
維基百科國際
維基百科
維基百科國際

abcfgop

abcfgop

Kerning

AV Wa
No kerning

AV Wa
Kerning applied

$AE \rightarrow \text{Æ}$	$ij \rightarrow \text{ij}$
$ae \rightarrow \text{æ}$	$st \rightarrow \text{ſt}$
$OE \rightarrow \text{Œ}$	$ft \rightarrow \text{ft}$
$oe \rightarrow \text{œ}$	$et \rightarrow \text{\&}$
$ff \rightarrow \text{ff}$	$fs \rightarrow \text{\beta}$
$fi \rightarrow \text{fi}$	$ffi \rightarrow \text{ffi}$

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
- ▶ Содержит 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
- ▶ Содержит информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- ▶ Плохо ведёт себя при масштабировании (как увеличении, так и уменьшении), bitmap'ы не особо помогают

Рендеринг bitmap-шрифтов

- ▶ Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- ▶ Содержит информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- ▶ Плохо ведёт себя при масштабировании (как увеличении, так и уменьшении), bitmap'ы не особо помогают
- ▶ Очень прост в реализации

Рендеринг bitmap-шрифтов

- ▶ Обычно представлены в виде texture atlas: одна текстура, содержащая все глифы шрифта
- ▶ Содержит информацию о расположении глифов в текстуре (текстурные координаты левого верхнего и правого нижнего пикселя)
- ▶ Плохо ведёт себя при масштабировании (как увеличении, так и уменьшении), bitmap'ы не особо помогают
- ▶ Очень прост в реализации
- ▶ Часто используется для дебажного текста, инди-игр, и т.п.

Bitmap-шрифт

! " # \$ % & ' () * + , - . / 0 1
2 3 4 5 6 7 8 9 : ; < = > ? @ 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 [\] ^ _ ` 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 { | } ~

Bitmap-шрифт: описание в коде

```
struct bitmap_font
{
    GLuint texture_id;

    struct glyph
    {
        vec2 top_left;
        vec2 bottom_right;
    };

    std::unordered_map<std::uint32_t, glyph_data> glyphs;
};
```

Рендеринг векторных шрифтов

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

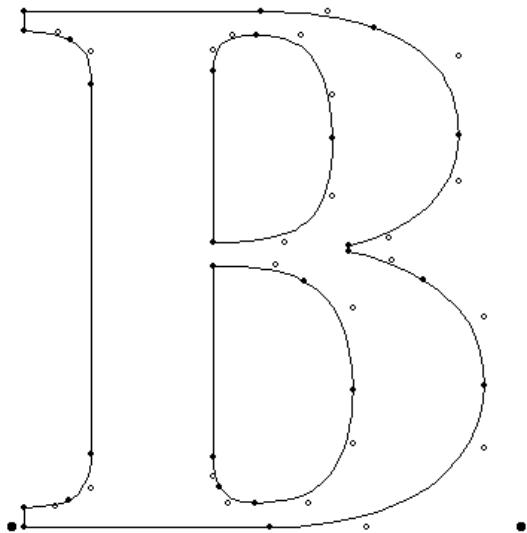
Рендеринг векторных шрифтов

- ▶ Глиф описывается как набор геометрических фигур (фигура может описывать "дырку" в другой фигуре, как дырка в букве "О"), граница фигуры - набор отрезков и квадратичных кривых Безье
- ▶ Много разных способов рендеринга:
 - ▶ Аппроксимация набором треугольников
 - ▶ Заpackовка фигур в текстуру, шейдер вычисляет площадь пересечения фигуры и пикселя
 - ▶ Полигональная аппроксимация глифа (рисуеться 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 "О"), граница фигуры - набор отрезков и квадратичных кривых Безье
- ▶ Много разных способов рендеринга:
 - ▶ Аппроксимация набором треугольников
 - ▶ Заpackовка фигур в текстуру, шейдер вычисляет площадь пересечения фигуры и пикселя
 - ▶ Полигональная аппроксимация глифа (рисуеться 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) 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) или текстурой
- ▶ SDF-сцены часто используются для экспериментального рендеринга и удобны для raymarching'a

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 (рисуем чёрный пиксель), иначе - нет (рисуем белый пиксель)
 - ▶ Обычно добавляется интерполяция от чёрного к белому в районе границы глифа для сглаживания

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 (рисуем чёрный пиксель), иначе - нет (рисуем белый пиксель)
 - ▶ Обычно добавляется интерполяция от чёрного к белому в районе границы глифа для сглаживания
- ▶ 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 (рисуем чёрный пиксель), иначе - нет (рисуем белый пиксель)
 - ▶ Обычно добавляется интерполяция от чёрного к белому в районе границы глифа для сглаживания
- ▶ Прост в реализации
- ▶ Требуется чуть больше памяти под текстуры

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 (рисуем чёрный пиксель), иначе - нет (рисуем белый пиксель)
 - ▶ Обычно добавляется интерполяция от чёрного к белому в районе границы глифа для сглаживания
- ▶ Прост в реализации
- ▶ Требуется чуть больше памяти под текстуры
- ▶ Неплохо масштабируется (бывают артефакты, но куда 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 (рисуем чёрный пиксель), иначе - нет (рисуем белый пиксель)
 - ▶ Обычно добавляется интерполяция от чёрного к белому в районе границы глифа для сглаживания
- ▶ Прост в реализации
- ▶ Требуется чуть больше памяти под текстуры
- ▶ Неплохо масштабируется (бывают артефакты, но куда менее серьёзные, чем для bitmap-шрифтов)
- ▶ Один из самых распространённых способов рендеринга шрифтов

SDF-шрифт

@}{()jll[\$Q%OGC&S#9/\

U389Y06qb?pdJfWMA YV

XRDKTNHZPBE4F25Lkh1

!lll7t;oaecsmnurwxvz:><

+^*=|||~\.-

SDF-шрифт: артефакты при magnification

Ta
Ta

Рендеринг SDF-шрифтов

- ▶ Можно легко реализовать много дополнительных эффектов:

Рендеринг SDF-шрифтов

- ▶ Можно легко реализовать много дополнительных эффектов:
 - ▶ Обводка текста другим цветом: рисуем цвет обводки, если $0 \leq f(p) \leq \varepsilon$

Рендеринг SDF-шрифтов

- ▶ Можно легко реализовать много дополнительных эффектов:
 - ▶ 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;  
  
in vec2 texcoord;  
  
layout (location = 0) out vec4 out_color;  
  
void main()  
{  
    float sdfValue = texture(sdfTexture, texcoord).r;  
    float alpha = smoothstep(-0.5, 0.5, sdfValue);  
    out_color = vec4(0.0, 0.0, 0.0, alpha);  
}
```

Рендеринг текста: ссылки

- ▶ FreeType
- ▶ harfbuzz
- ▶ Тьюториал по рендерингу bitmap-шрифтов
- ▶ Тьюториал по рендерингу SDF-шрифтов
- ▶ Один способ рендеринга векторных шрифтов
- ▶ Другой способ рендеринга векторных шрифтов
- ▶ Slug algorithm
- ▶ Slug library