

# Компьютерная графика

## Практика 15: MSDF-текст

2022



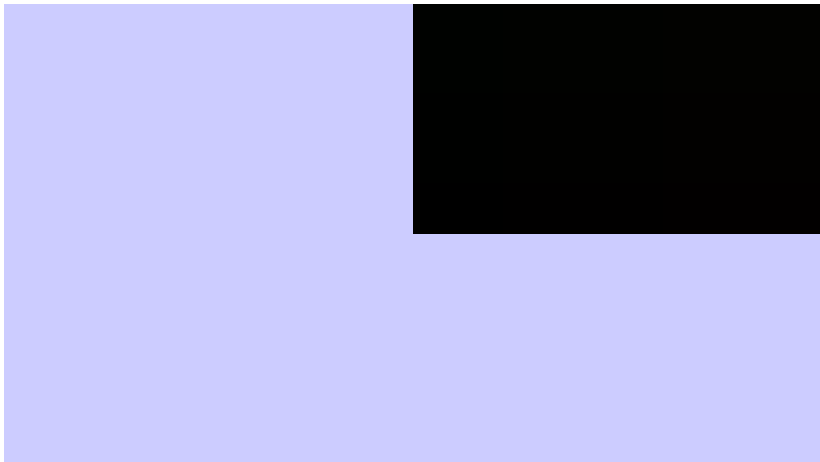
- ▶ В этой практике есть текст "Hello, world!" по умолчанию, но можно его стирать и печатать свой прямо в программе :)

# Задание 1

Рисуем треугольник

- ▶ Заводим структуру вершины с полями `vec2 position` и `vec2 texcoord`
- ▶ Заводим VAO + VBO для них, настраиваем атрибуты
- ▶ Используем атрибуты в шейдерной программе:  
`gl_Position = vec4(position, 0.0, 1.0)`, в качестве цвета выводим `vec4(texcoord, 0.0, 1.0)`
- ▶ Инициализируем VBO данными при создании: треугольник с координатами (0,0), (100,0) и (0,100) и текстурными координатами (0,0), (1,0) и (0,1)
- ▶ Рисуем этот треугольник (координаты сильно выходят за диапазон `[-1..1]`, так что мы увидим только угол треугольника)

# Задание 1



## Задание 2

### Настраиваем матрицу

- ▶ Заводим матрицу `transform`, которая переводит из экранных координат в OpenGL-ные:  
 $[0, width] \times [height, 0] \mapsto [-1, 1] \times [-1, 1]$
- ▶ Передаём эту матрицу в качестве значения `uniform-переменной transform`
- ▶ В шейдере применяем к вершине матрицу `transform` (сама матрица там уже есть)
- ▶ N.B. экранная Y-координата идёт **сверху вниз**, а OpenGL-ная – **снизу вверх**!
- ▶ N.B. треугольник должен появиться в верхнем левом углу и быть размером ровно 100 пикселей

## Задание 2



## Задание 3

Генерируем глифы

- ▶ Убираем заполнение VBO на старте
- ▶ Вместо этого в цикле рисования в случае, если флаг `text_changed` имеет значение `true`, генерируем новый массив вершин:
  - ▶ Заводим координаты 'пера' `vec2 pen(0.0)`, это точка отсчёта для текущего символа
  - ▶ Проходимся по всем буквам переменной `text`, находим соответствующий глиф в `font.glyphs`
  - ▶ Для каждой буквы генерируем прямоугольник из 4 вершин (разбивая вручную на 2 треугольника, т.е. в итоге 6 вершин)
  - ▶ Координаты вершины – `[glyph.xoffset .. glyph.xoffset + glyph.width] + pen.x`, аналогично для `Y`
  - ▶ Текстурные координаты вершины – `[glyph.x .. glyph.x + glyph.width] / texture_width`, аналогично для `Y`
  - ▶ После каждого символа нужно сдвинуть перо по `X` на `glyph.advance`
- ▶ Загружаем эти вершины в VBO, запоминаем количество вершин, и очищаем флаг `text_changed`



## Задание 3



## Задание 4

Выводим MSDF-глифы

- ▶ Передаём значение `font.sdf_scale` в новую uniform-переменную `float sdf_scale`
- ▶ Заводим uniform-переменную для текстуры шрифта `sampler2D sdf_texture` (она уже выставлена для нулевого texture unit'a, ничего дополнительно делать не нужно)
- ▶ Выводим буквы чёрного цвета с прозрачностью, посчитанной через SDF (см. слайды лекции)

## Задание 4

Hello, world!

## Задание 5

### Центрируем текст

- ▶ При обновлении текста вычисляем bounding box всех вершин (т.е. максимальные и минимальные  $X$  и  $Y$  координаты)
- ▶ Дополняем матрицу transform так, чтобы центр текста был в центре экрана

## Задание 5

Hello, world!

## Задание 6

Увеличиваем текст

- ▶ Дополняем матрицу `transform` так, чтобы буквы стали больше (примерно в 5-6 раз, не принципиально)
- ▶ Сглаживание на границе букв не учитывает растяжение и будет размытым
- ▶ Чтобы сделать чёткое сглаживание, вместо значения 0.5 в функции `smoothstep` используем величину  $\text{length}(\text{vec2}(\text{dFdx}(\text{sdfValue}), \text{dFdy}(\text{sdfValue}))) / \text{sqrt}(2.0)$

Hello, world!

## Задание 7\*

Добавляем обводку текста

- ▶ Дорабатываем шейдер, чтобы у чёрного текста появилась белая обводка
- ▶ Параметры подберите на свой вкус, главное – чтобы при изменении размеров экрана всё ещё выглядело красиво :)



Hello, world!