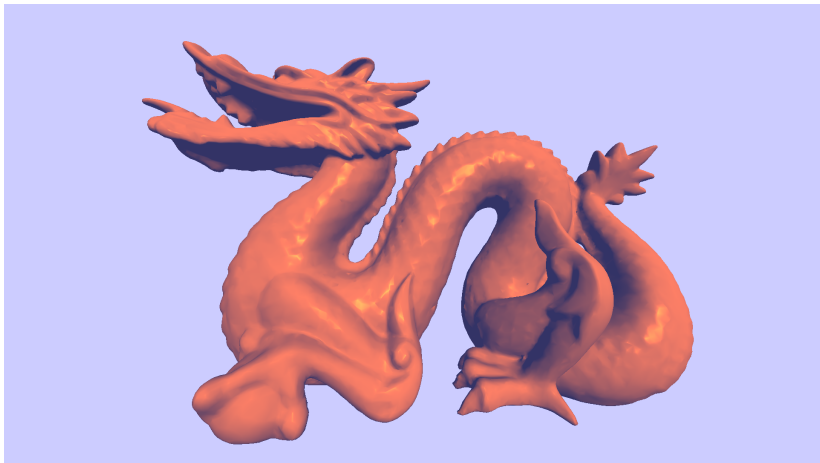


Компьютерная графика

Практика 7: Рендеринг в текстуру, пост-обработка

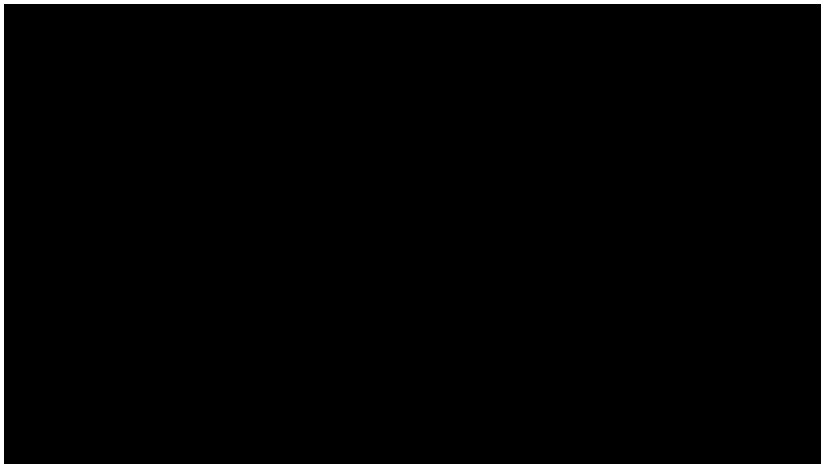
2021



Задание 1

Создаём и настраиваем framebuffer

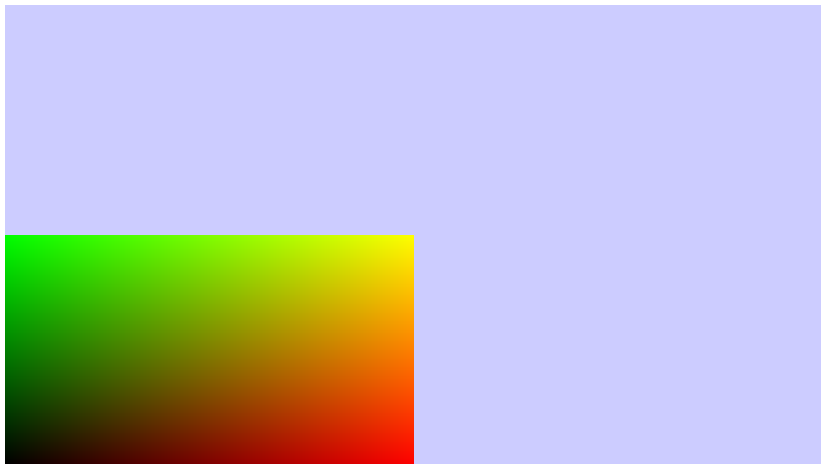
- ▶ Создаём текстуру для цветового буфера: min/mag фильтры – `GL_NEAREST`, размеры – `width/2` на `height/2`, формат – `GL_RGBA8`, в качестве данных можно передать `nullptr`
- ▶ Создаём renderbuffer для буфера глубины: `glGenRenderbuffers`, `glBindRenderbuffer`, `glRenderbufferStorage`, формат – `GL_DEPTH_COMPONENT24`, размеры – как у текстуры
- ▶ Создаём framebuffer: `glGenFramebuffers`
- ▶ Присоединяем к нему цвет и глубину: `glBindFramebuffer` (`target = GL_DRAW_FRAMEBUFFER`), `glFramebufferTexture` с параметром `attachment = GL_COLOR_ATTACHMENT0`, `glFramebufferRenderbuffer` с параметром `attachment = GL_DEPTH_ATTACHMENT`
- ▶ Проверяем, что фреймбуффер настроен правильно: `glCheckFramebufferStatus` должна вернуть `GL_FRAMEBUFFER_COMPLETE`
- ▶ Текстуре и renderbuffer'у нужно обновлять размер при изменении размера экрана: там, где обрабатывается `SDL_WINDOWEVENT_RESIZED`, нужно добавить `glBindTexture`, `glTexImage2D`, `glBindRenderbuffer`, `glRenderbufferStorage`
- ▶ Картинка будет чёрной, так как теперь мы ничего не рисуем в default framebuffer



Задание 2

Рисуем в текстуру

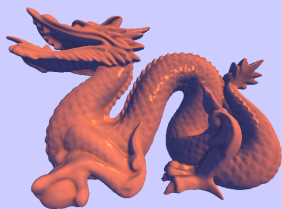
- ▶ Перед `glClear` устанавливаем наш framebuffer текущим для рисования (`glBindFramebuffer`) и настраиваем `glViewport`
- ▶ После рисования основной модели, перед рисованием прямоугольника делаем текущим для рисования основной framebuffer (`id = 0`), настраиваем `glViewport` и очищаем цветовой буфер и буфер глубины (`glClear`)
- ▶ Раскомментируем рисование прямоугольника: пока он всё ещё выведется просто с цветовым градиентом, так как шейдер мы не меняли
- ▶ Дракон виден не будет, так как он нарисовался в другой фреймбуфер



Задание 3

Выводим нарисованную текстуру

- ▶ Добавляем в шейдер для прямоугольника текстуру, которую мы нарисовали с помощью framebuffer'a:
`uniform sampler2D render_result;`
- ▶ В `out_color` пишем значение пикселя из текстуры
- ▶ Устанавливаем значение соответствующей uniform-переменной в 0 (`glGetUniformLocation`, `glUniform1i`)
- ▶ Перед рисованием прямоугольника устанавливаем текстуру текущей для texture unit 0 (`glActiveTexture`, `glBindTexture`)



Задание 4

Выводим текстуру несколько раз

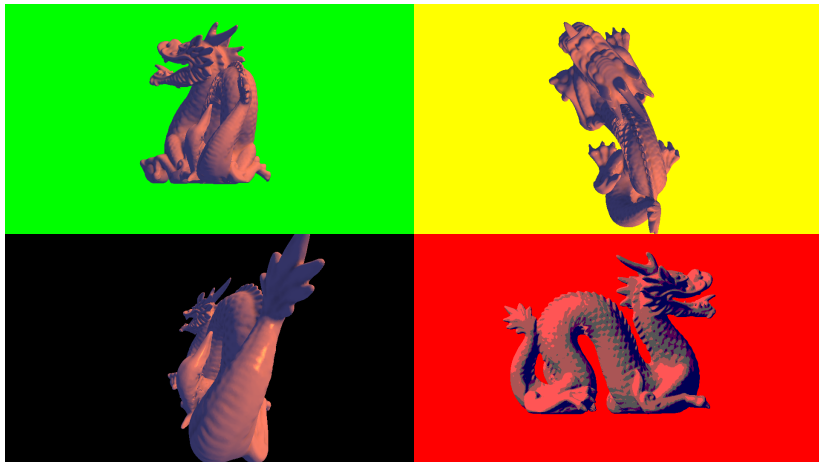
- ▶ Заносим весь рендеринг (**не** обработку событий SDL и **не** функцию `SDL_GL_SwapWindow`) в цикл, повторяющий рендеринг 4 раза
- ▶ В зависимости от номера повторения меняем uniform-переменные `center` и `size`, чтобы картинка нарисовалась во всех четырёх углах экрана
- ▶ В зависимости от номера повторения меняем `glClearColor` (цвета выберите сами)
- ▶ В зависимости от номера повторения меняем проекцию:
 - ▶ 0 – перспективная проекция, ничего не меняем
 - ▶ 1 – ортографическая проекция на плоскость XY
 - ▶ 2 – ортографическая проекция на плоскость XZ
 - ▶ 3 – ортографическая проекция на плоскость YZ
- ▶ Конкретное соответствие номеров и проекций не так важно, главное – увидеть три ортографических проекции на разные плоскости и одну перспективную
- ▶ Ортографическую проекцию можно сделать с помощью `projection = glm::ortho(...)` (**учитывая aspect ratio экрана!**) и настроив `view` какими-нибудь вращениями



Задание 5

Делаем color grading

- ▶ Добавляем во фрагментный шейдер пост-обработки (с которым рисуется прямоугольник) uniform-переменную `uniform int mode;` и устанавливаем её в цикле рендеринга в номер картинки (0..3)
- ▶ При `mode == 1` преобразуем цвет как `color = floor(color * 4.0) / 3.0`
- ▶ При остальных значениях `mode` пока выводим просто текстуру без искажений

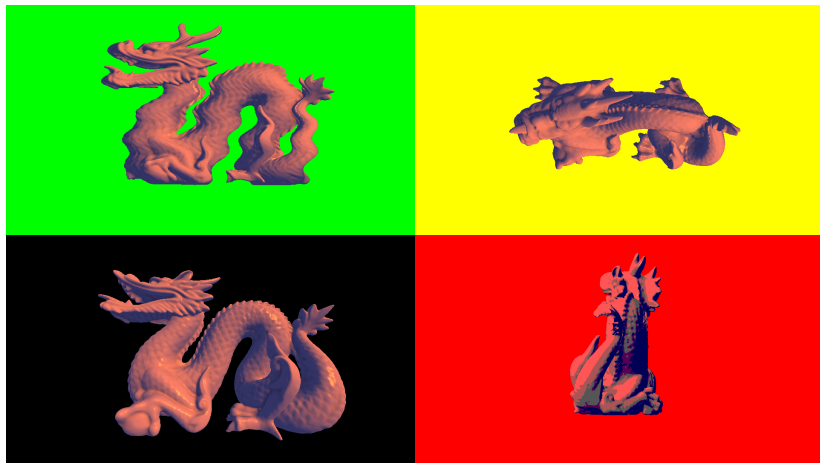


Задание 6

Искажаем изображение

- ▶ При `mode == 2` искажаем изображение: прибавляем к текстурной координате что-нибудь зависящее от времени и координаты, например

```
texcoord + vec2(sin(texcoord.y * 50.0 + time)  
                * 0.01, 0.0)
```



Задание 7*

Делаем гауссово размытие

- ▶ При `mode == 3` делаем гауссово размытие, например, с $N = 7$ и `radius = 5.0` (код и описание есть в слайдах лекции)
- ▶ N.B.: у вас получится $(2N + 1)^2$ обращений к текстуре

