

Компьютерная графика

Практика 5: Текстуры

2021



Задание 1

Загружаем модель cow, как в предыдущей практике:

- ▶ Создаём VAO, VBO, EBO
- ▶ Загружаем вершины в VBO
- ▶ Загружаем индексы в EBO
- ▶ Настраиваем атрибуты в VAO (пока только первые два)
- ▶ Связываем индексы с VAO
- ▶ Рисуем с помощью `glDrawElements`
- ▶ N.B. модель можно крутить и двигать стрелочками



Задание 2

Добавляем текстурные координаты

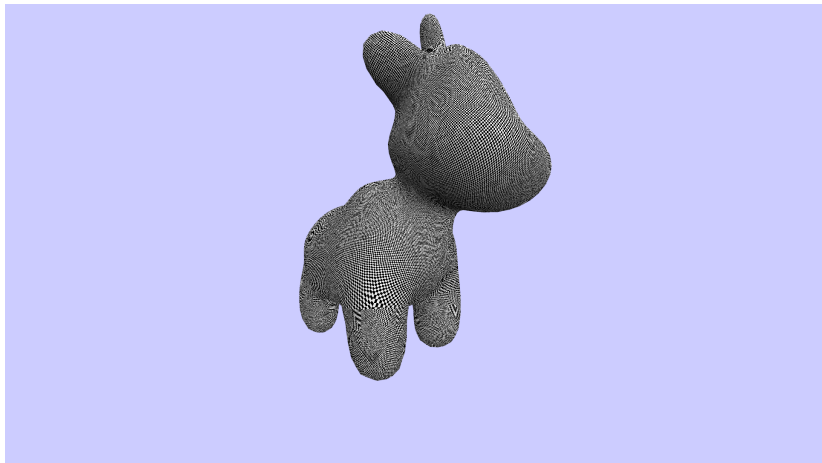
- ▶ Описываем новый (index=2) атрибут для VAO
- ▶ Добавляем входной атрибут типа `vec2` в вершинном шейдере, и передаём его без изменений во фрагментный
- ▶ Во фрагментном шейдере используем текстурные координаты в качестве цвета (альбедо), например `albedo = vec3(texcoord, 0.0)`



Задание 3

Создаём и используем текстуру шахматной раскраски

- ▶ Создаём объект текстуры типа `GL_TEXTURE_2D`
- ▶ Выставляем для него `min` и `mag` фильтры в `GL_NEAREST` (`glTexParameteri`)
- ▶ Выбираем размер текстуры (лучше довольно большой, в духе 512 или 1024)
- ▶ В качестве данных загружаем пиксели шахматной раскраски
 - ▶ Пиксели удобнее всего хранить как `std::uint32_t`
 - ▶ Создать массив пикселей можно как
`std::vector<std::uint32_t> pixels(size * size);`
 - ▶ Заполнить его чёрными `0xFF000000u` и белыми `0xFFFFFFFFu` пикселями в порядке шахматной раскраски
 - ▶ Для `glTexImage2D` параметры
`internalFormat = GL_RGBA8, format = GL_RGBA,`
`type = GL_UNSIGNED_BYTE`
- ▶ Во фрагментном шейдере добавляем `uniform`-переменную типа `sampler2D` и берём из неё цвет (`albedo`) функцией `texture`
- ▶ В значение `uniform`-переменной записываем 0 (`glUniform1i`)
- ▶ Перед рисованием делаем нашу текстуру текущей для `texture unit`'а номер 0 (`glActiveTexture + glBindTexture`)

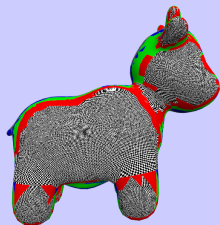


- ▶ Если подвигать и покрутить модель, будет видный сильный *муар* – эффект, возникающий, когда значения сигнала (в нашем случае – текстура) читаются с частотой меньшей, чем частота самого сигнала (в нашем случае частоту определяют пиксели экрана)

Задание 4

Вручную добавляем mipmap-уровни

- ▶ Меняем min-фильтр текстуры на `GL_NEAREST_MIPMAP_NEAREST`
- ▶ После загрузки данных текстуры вызываем `glGenerateMipmap`, чтобы сгенерировались mipmap-уровни
- ▶ Перезаписываем 1-ый mipmap-уровень монохромной картинкой красного цвета
 - ▶ Размер по обеим осям должен быть ровно в 2 раза меньше, чем размер исходного изображения
- ▶ Перезаписываем 2-ый mipmap-уровень монохромной картинкой зелёного цвета
 - ▶ Размер по обеим осям должен быть ровно в 4 раза меньше, чем размер исходного изображения
- ▶ Перезаписываем 3-ый mipmap-уровень монохромной картинкой синего цвета
 - ▶ Размер по обеим осям должен быть ровно в 8 раз меньше, чем размер исходного изображения
- ▶ Подвигайте модель, чтобы увидеть, как меняется выбранный mipmap-уровень



Задание 5

Загружаем настоящую текстуру

- ▶ Создаём *новую* текстуру
- ▶ Настраиваем для неё mag и min фильтры в `GL_LINEAR` и `GL_LINEAR_MIPMAP_LINEAR` соответственно
- ▶ Загружаем её данные из файла с помощью функции `stbi_load`, путь до изображения есть в переменной `cow_texture_path`, число каналов `desired_channels` указывать равным 4
- ▶ Загружаем эти данные на GPU с помощью `glTexImage2D` и не забываем сгенерировать mipmapс
- ▶ Очищаем данные на CPU с помощью `stbi_image_free`
- ▶ Используем для этой текстуры texture unit с номером 1, шейдер на этом этапе меняться не должен



Задание 6*

Крутим текстуру по модели

- ▶ Передаём текущее время в шейдер в качестве uniform-переменной
- ▶ Сдвигаем текстурные координаты на какое-то зависящее от времени значение

