МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3
по курсу “Компьютерная графика”

Основы 3D графики

Выполнил: В. Н. Шиширин
Группа: М8О-310Б-23
Преподаватель: В. Д. Бахарев

Москва, 2025

Условие

В этой лабораторной работе вам предстоит загрузить изображение из файла, создать текстуру из изображения и, используя текстурные координаты, а также сэмплеры, наложить текстуру на объекты

- Загрузить изображение из файла (желательно, чтобы ширина и высота изображения имели значение степени двойки, сами знаете зачем)
- Создать текстуру из изображения (использовать veekay::graphics::Texture)
- Создать объект сэмплера и задать разумные параметры сэмплирования
- Записать в набор дескрипторов новую привязку (дескриптор – изображение+сэмплер), чтобы его увидел фрагментный шейдер
- Вершины объектов должны содержать текстурные координаты
- Фрагментный шейдер должен сэмплировать текстуру, используя текстурные координаты, которые были переданы из вершинного шейдера
- Реализовать использование разных текстур (VkImageView) и сэмплеров (VkSampler) моделями на сцене (с помощью использования разных наборов дескрипторов для каждой текстуры или набора текстур, то есть “материала”)

Метод решения

Задача: загрузить изображение, создать текстуру и сэмплер, добавить в вершины UV-координаты и реализовать сэмплинг текстур в фрагментном шейдере; обеспечить возможность использования разных текстур/самплеров для разных материалов через отдельные наборы дескрипторов.

Математика и логика:

- UV-координаты: каждой вершине добавляется атрибут uv = (u, v) в диапазоне [0,1]. Вершинный шейдер передаёт uv во фрагментный

шейдер; аппаратная интерполяция линейно интерполирует uv по треугольнику.

- Вычисление цвета: фрагментный шейдер читает базовый цвет из текстуры через вызов типа color = texture(sampler, uv) и комбинирует его с параметрами материала (albedo, модификаторы освещения).
- Фильтрация и мипмапы: при масштабировании/уменьшении текстуры используются режимы фильтрации — ближайший (nearest), билинейный (linear) и режимы mipmap (nearest/mipmap_linear) для предотвращения aliasing; для улучшения качества можно использовать anisotropic filtering. Рекомендуется генерировать mipmaps (либо на CPU при загрузке, либо посредством vkCmdBlitImage) — по этой причине ширина/высота изображения часто выбираются степенью двойки.
- Адресация: режимы адресации (repeat, clamp_to_edge, mirrored_repeat) определяют поведение при uv вне [0,1]; выбирается в настройках семплера в зависимости от задачи.
- Несколько текстур/материалов: разные модели связываются с разными VkImageView+VkSampler через отдельные descriptor sets или разные слоты в material descriptor set; в шейдере выбирается соответствующий сэмплер/текстура по дескриптору материала.

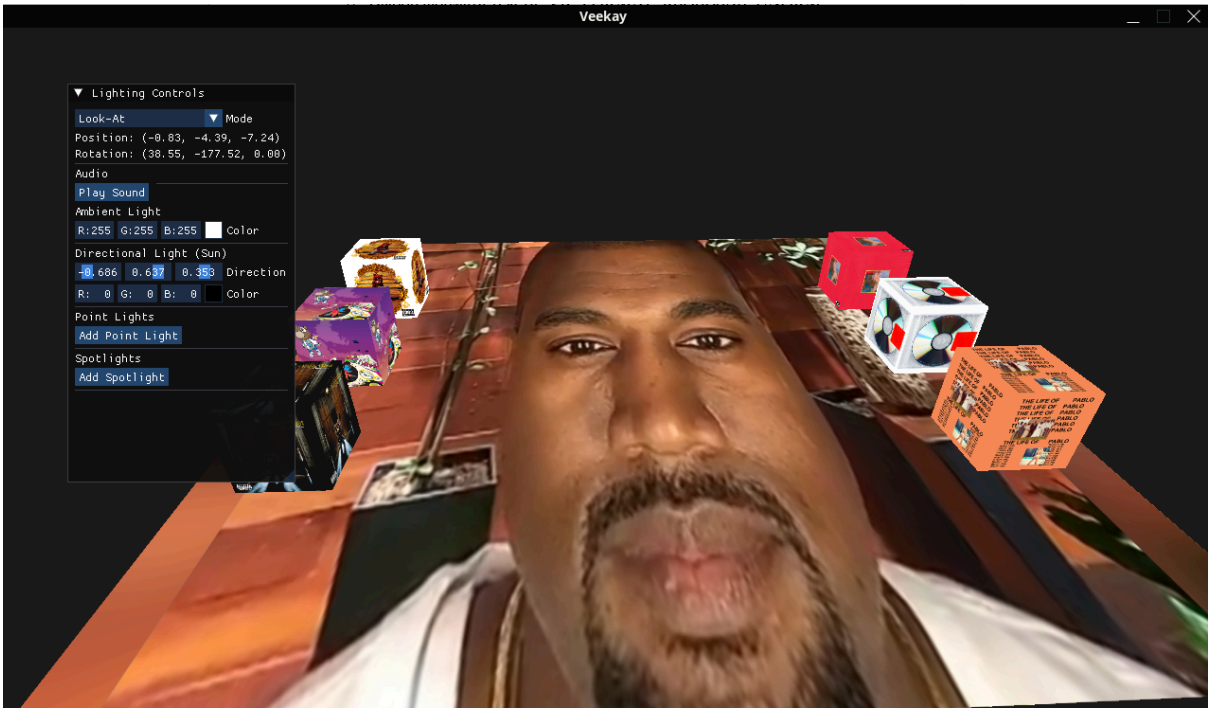
Ключевые Vulkan-объекты и шаги (перечисление):

- Загрузка изображения и создание текстуры:
  - Функция: loadTexture / veekay::graphics::Texture (создаёт VkImage, выделяет память, формирует VkImageView в формате, например, VK_FORMAT_R8G8B8A8_UNORM).
  - Используется staging buffer (host visible VkBuffer) → vkCmdCopyBufferToImage → переходы layout (VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL → VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL).
  - (Опционально) генерация mipmaps через vkCmdBlitImage или заранее подготовленные уровни.
- Семплер:
  - Создание VkSampler (параметры: magFilter/minFilter, mipmapMode, addressModeU/V/W, maxAnisotropy, compareEnable=false).
  - Функция/операция: createSampler / vkCreateSampler.
- Дескрипторы:

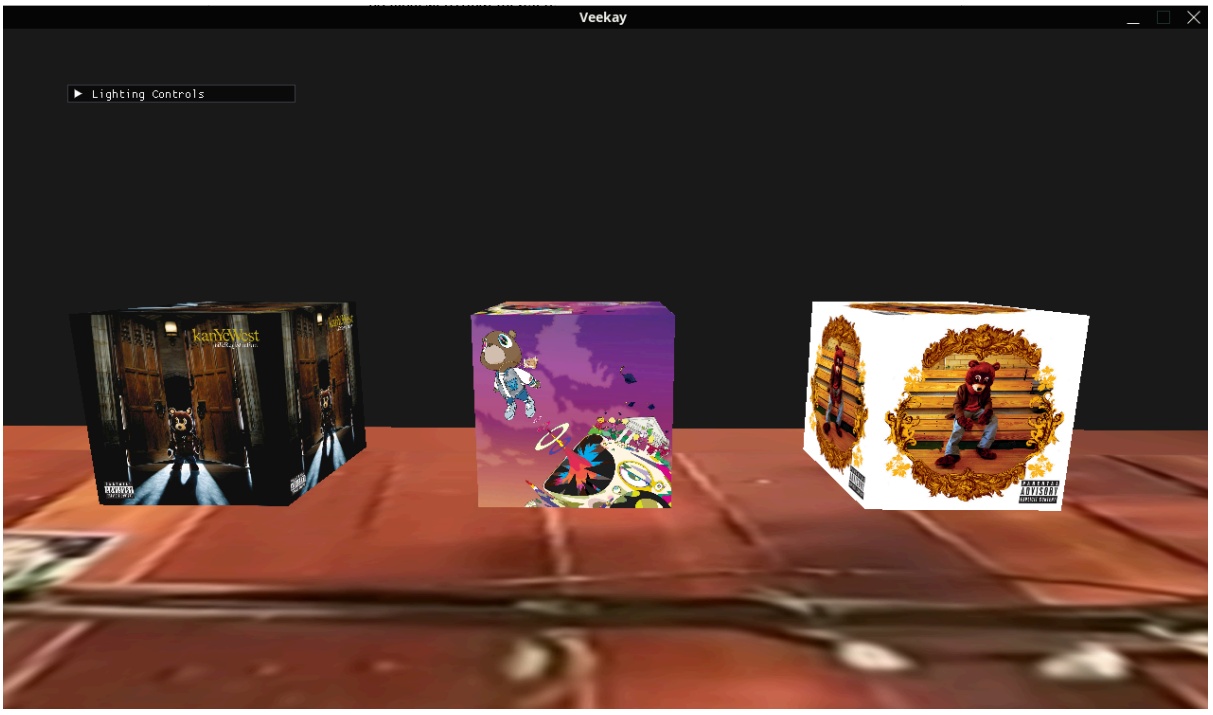
    - VkDescriptorSetLayout содержит биндинг типа VK_DESCRIPTOR_TYPE_COMBINED_IMAGE_SAMPLER для текстуры+сэмплера.
    - VkDescriptorPool / VkDescriptorSet: allocate и vkUpdateDescriptorSets (передаётся VkDescriptorImageInfo с imageView и sampler).
    - Для нескольких материалов — отдельные descriptor sets (material_descriptor_sets) или массивы дескрипторов.
- Вершинный формат:
    - VkVertexInputAttributeDescription включает атрибут для UV (VK_FORMAT_R32G32_SFLOAT) и соответствующий offset в структуре Vertex.
    - Vertex buffer хранит position, normal, uv (и другие атрибуты при необходимости).
- Шейдеры:
    - Вершинный шейдер (shaders/shader.vert): получает uv и передаёт во фрагментный шейдер.
    - Фрагментный шейдер (shaders/shader.frag): объявляет uniform sampler2D (или array/struct для нескольких текстур) и выполняет color = texture(sampler, uv).
- Команды рендеринга и обновление данных:
    - Перед рендером CPU заполняет staging buffer данными изображения и обновляет descriptor sets (vkUpdateDescriptorSets).
    - В рендер-процессе: vkCmdBindDescriptorSets, vkCmdBindVertexBuffers, vkCmdDrawIndexed.
- Дополнительно:
    - Выбор формата текста: VK_FORMAT_R8G8B8A8_UNORM типичен для 8-бит RGBA.
    - Параметры семплера для качества: minFilter = VK_FILTER_LINEAR, magFilter = VK_FILTER_LINEAR, mipmapMode = VK_SAMPLER_MIPMAP_MODE_LINEAR, addressMode = VK_SAMPLER_ADDRESS_MODE_REPEAT, maxAnisotropy = 16 (при поддержке).
    - Для разных моделей/материалов — поддержка нескольких VkImageView/VkSampler и организация material descriptor sets.

Результаты

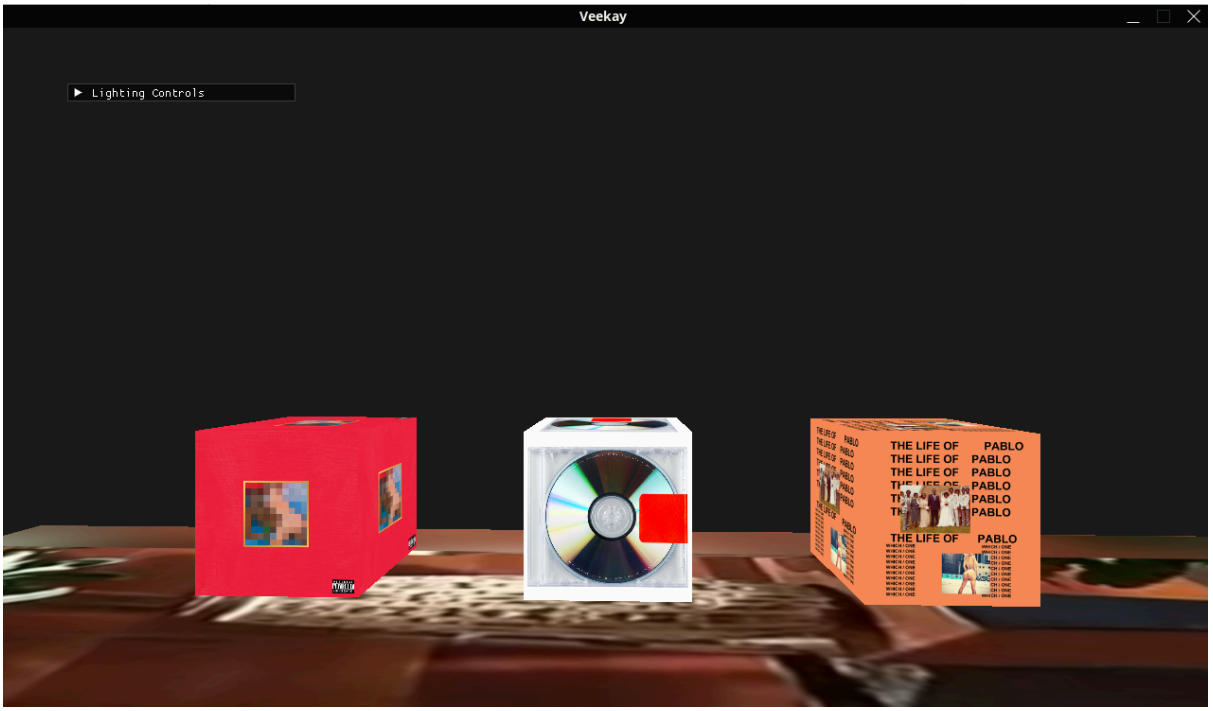
Добавленные текстуры на объекты в сцене



Текстуры на объектах в виде кубов



Текстуры на объектах в виде кубов



Выводы

Я научился загружать изображения и создавать из них текстуры (VkImage/VkImageView) с корректной конфигурацией семплера (фильтрация, режим адресации, mipmaps). Я научился добавлять UV-координаты в вершины, передавать их в шейдеры и делать сэмплинг в фрагментном шейдере (texture(sampler, uv)), а также связывать разные текстуры и сэмплеры с моделями через descriptor sets для реализации материалов.