

**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»**

Студент: Г. А. Ермеков  
Преподаватель: В. Д. Бахарев  
Группа: М8О-301Б-23  
Дата:  
Оценка:  
Подпись:

**Москва, 2025**

## **Условие**

**Задача:** 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.

# 1 Метод решения

Для реализации поставленной задачи была доработана программа на языке C++ с использованием графического API Vulkan.

## 1 Загрузка изображения

Для загрузки изображения из файла была использована библиотека lodepng. Изображение загружается в массив байтов, который затем передается в конструктор класса текстуры.

```
1 || std::vector<unsigned char> image;
2 || unsigned width, height;
3 || unsigned error = lodepng::decode(image, width, height, "assets/lenna.png");
```

## 2 Создание текстуры

Класс veeekay::graphics::Texture (в файле source/graphics.cpp) инкапсулирует создание VkImage, выделение памяти и загрузку данных. Конструктор принимает указатель на пиксели и выполняет следующие действия:

1. Создает VkImage с нужными параметрами (размер, формат).
2. Выделяет и привязывает память (vkAllocateMemory, vkBindImageMemory).
3. Копирует данные пикселей из временного (staging) буфера в изображение, выполняя необходимые переходы раскладки памяти (layout transitions) с помощью vkCmdPipelineBarrier.
4. Создает VkImageView для доступа к изображению из шейдеров.

## 3 Сэмплер

Для выборки текстелей из текстуры в шейдере был создан объект сэмплера VkSampler. Настроен линейный фильтр (VK\_FILTER\_LINEAR) для минификации и магнификации, а также режим повторения текстуры (VK\_SAMPLER\_ADDRESS\_MODE\_REPEAT).

## 4 Дескрипторы

Для передачи текстуры и сэмплера в шейдер был обновлен макет дескрипторов (VkDescriptorSetLayout). Добавлена привязка (binding 2) типа VK\_DESCRIPTOR\_TYPE\_COMBINED\_IMAGE\_SAMPLER. Обновление дескрипторного набора (vkUpdateDescriptorSets) связывает созданный VkImageView и VkSampler с соответствующей привязкой.

## 5 Шейдеры

**Вершинный шейдер** (`shader.vert`) принимает текстурные координаты в атрибуте вершины (location 2) и передает их во фрагментный шейдер.

**Фрагментный шейдер** (`shader.frag`) принимает интерполированные текстурные координаты и использует сэмплер для получения цвета пикселя:

```
1 || layout (binding = 2) uniform sampler2D texSampler;
2 || //
3 || vec3 base_color = texture(texSampler, f_uv).rgb * albedo_color;
```

Полученный цвет текстуры затем используется в расчетах освещения.

## 2 Результаты

В ходе выполнения лабораторной работы были изучены механизмы работы с текстурами в графическом API Vulkan.

Были реализованы следующие этапы:

- Загрузка изображения формата PNG с диска в оперативную память.
- Создание объектов Vulkan: `VkImage`, `VkImageView`, `VkSampler`.
- Настройка барьеров памяти для корректного перевода изображения в оптимальный для чтения шейдером формат (`VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL`).
- Модификация графического конвейера для передачи текстурных координат и сэмплеров в шейдеры.

В результате на 3D-объекты сцены (кубы и плоскость) корректно накладывается текстура. Текстура корректно взаимодействует с моделью освещения, модулируя базовый цвет объектов.

В ходе выполнения лабораторной работы был успешно реализован механизм работы с текстурами в Vulkan. Изучены основные концепции работы с изображениями в графическом API: создание объектов изображений, управление памятью, настройка сэмплеров и передача данных в шейдеры. Полученные знания позволяют создавать визуально более привлекательные 3D-сцены с использованием текстур, что является важным шагом в освоении современной графической разработки.

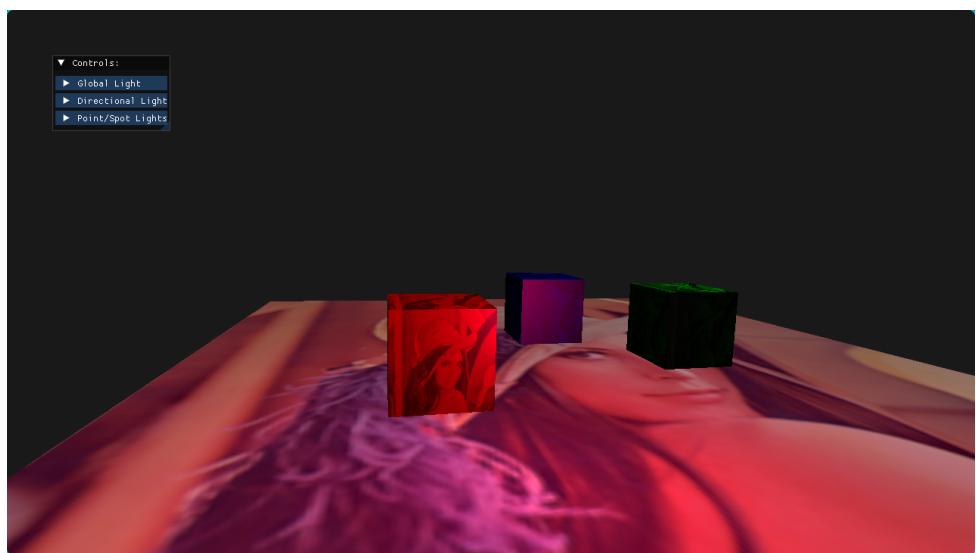


Рис. 1: Результат применения текстуры к объектам сцены