**Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки**

# Лабораторна робота №2

з дисципліни

«Комп’ютерна графіка та мультимедіа»

Виконав:

студент групи ІП-05

Гапій Денис Едуардович

номер у списку групи: 5

Перевірив:

Родіонов П. Ю.

Київ 2022

## Тема: «Вступ до комп’ютерної графіки»

**Мета**: навчитися налаштовувати середовище API Vulkan для програмування комп’ютерної графіки.

## Контрольні запитання:

1. Теоретичні засади функціонування API Vulkan.

   Vulkan розроблено з нуля для роботи на різноманітних платформах, починаючи від мобільних телефонів і планшетів і закінчуючи ігровими консолями та високоякісними настільними комп’ютерами. Базовий дизайн API є багаторівневим або, ми можемо сказати, модульним, тому він дає змогу створити загальну, але розширювану архітектуру для перевірки коду, налагодження та профілювання, не впливаючи на продуктивність.

2. Можливості практичного застосування API Vulkan.

   Для 3D-графіки та обчислювальних програм. Теоретично Vulkan можна використовувати в апаратному забезпеченні паралельних обчислень, для керування десятками мільярдів ядер графічного процесора, у крихітних переносних пристроях та іграшкових безпілотниках, у 3D-принтерах, автомобілях, комплектах віртуальної реальності та майже в будь-якому іншому із сумісним графічним процесором.

3. Порівняльна характеристика OpenGL та API Vulkan.

   Враховуючи основні принципи, використані в розробці Vulkan, — його застосування має принести перевагу в швидкодії в порівнянні з OpenGL, шляхом більш ефективного використання GPU, завдяки меншому навантаженню на ЦП. Також в плюси можна записати: чудова міжплатформенна підтримка, краща підтримка багатопоточних процесорів, і трохи агностицизму ОС.

| OpenGL | Вулкан |
| --- | --- |
| Спочатку створено для графічних робочих станцій із прямими рендерерами, розділеною пам'яттю. | Краще підходить для сучасних платформ, включаючи мобільні платформи з уніфікованою пам'яттю та підтримкою мозаїчного відтворення. |
| Драйвер виконує перевірку стану, відстеження залежностей, перевірку помилок. Це може обмежити та рандомізувати продуктивність. | Програма має прямий і передбачуваний контроль над GPU через явний API. |
| Застаріла модель потоків не дозволяє створювати графічні команди паралельно з виконанням команд. | API, розроблений для багатоядерних і багатопоточних платформ. Кілька командних буферів можна створювати паралельно. |
| Вибір API може бути складним, синтаксис розвивався протягом двадцяти років. | Видалення застарілих вимог спрощує дизайн API, спрощує інструкції з використання, зменшує розмір специфікації. |
| Компілятор мови шейдерів є частиною драйвера, і він підтримує лише GLSL. Джерело шейдера потрібно надіслати. | SPIR-V — це нова мета компілятора, що забезпечує гнучкість і надійність мови зовнішнього кінця. |
| Розробники повинні враховувати різницю між постачальниками. | Завдяки простішому API та загальномовним інтерфейсам більш суворе тестування збільшить сумісність між постачальниками. |

Рис. 1. Порівняння

4. Сутність та призначення технології SPIR-V.

   SPIR-V — це формат обміну двійковим проміжним представленням , який використовується для взаємодії з різнорідною машиною. SPIR-V виражає операції, які можна розбити відповідно до ISA , знайденого на GPU, FPGA, DSP, CPU тощо. Найпоширеніші випадки використання SPIR-V – це використання для етапів графічного шейдера та обчислювальних ядер у інші API, такі як Vulkan, OpenGL і OpenCL.

   • достатньо низького рівня, щоб обійти весь текстовий аналіз, залежність від однієї мови високого рівня та уникнути проблем спричинених з переносимістю зручності мов високого рівня та

   • достатньо високо-рівнева, щоб не залежати від пристрою та не втрачати інформацію, необхідну для максимальної продуктивності цільового пристрою.

5. Необхідні компоненти та кроки для налаштування роботи API Vulkan.

- Завантаження Vulkan API (SDK)
- Завантаження потрібних нам бібліотек GML, GLFW
- Створення проекту у вашому IDE.
- Додати шляхи заголовків бібліотек у властивостях проекту.
- Додати шляхи бібліотек до залежностей у властивостях проекту.
- Налаштувати розрядність застосунку та версію компілятора c++.

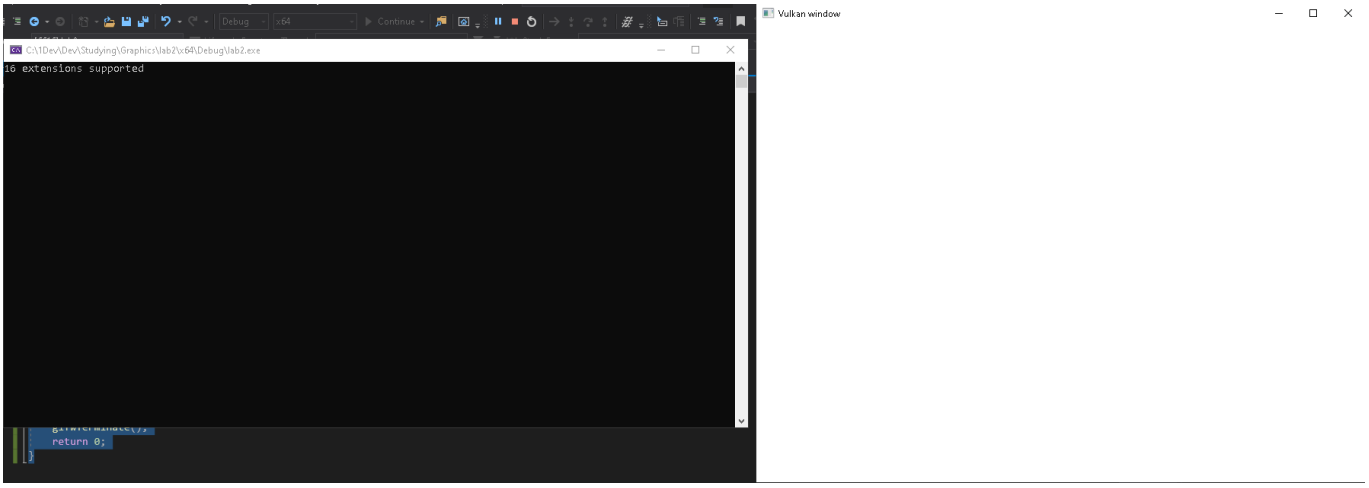
## Реалізація прикладу:

Використаний код:

```
#define GLFW_INCLUDE_VULKAN

#include <GLFW/glfw3.h>

#define GLM_FORCE_RADIANS

#define GLM_FORCE_DEPTH_ZERO_TO_ONE

#include <glm/vec4.hpp>

#include <glm/mat4x4.hpp>

#include <iostream>


int main() {

    glfwInit();

    glfwWindowHint(GLFW_CLIENT_API, GLFW_NO_API);

    GLFWwindow* window = glfwCreateWindow(800, 600, "Vulkan
window", nullptr, nullptr);

    uint32_t extensionCount = 0;

    vkEnumerateInstanceExtensionProperties(nullptr,
&extensionCount, nullptr);
```

```
    std::cout << extensionCount << " extensions
supported\n";
    glm::mat4 matrix;
    glm::vec4 vec;
    auto test = matrix * vec;
    while (!glfwWindowShouldClose(window)) {
        glfwPollEvents();
    }

    glfwDestroyWindow(window);
    glfwTerminate();
    return 0;
}
```

Результат запуску коду:




Вивід к-сті доповнень у консоль, та пусте вікно створене за допомогою GLFW.

## Висновок:

Я успішно налаштував середовище API Vulkan для програмування

комп’ютерної графіки та опанував усі потрібні для цього інструменти та залежності.

Завантажив та підключив до проекту залежності у вигляді бібліотек: GLFW - для створення вікна, яка, на щастя, підтримується не лише на Windows, а й на Linux та MacOS ; GLM — це бібліотека, розроблена для використання операцій лінійної алгебри з графічними API.

Реалізувавши запропонований викладачем приклад, створив порожній застосунок, для майбутнього відображення результатів наших шейдерів.

Частково проаналізував та засвоїв відмінності між Vulkan та OpenGL.