УДК 004.92:004.428 DOI: https://doi.org/10.30837/IYF.IIS.2024.293

ОПТИМАЛЬНЕ ВИКОРИСТАННЯ OPENGL ТА VULKAN

Цапко Б. В.
Науковий керівник – к.т.н., доцент Чуприна А. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: bohdan.tsapko@nure.ua

The purpose of this work is to make the optimal choice between the OpenGL and Vulkan APIs depending on the needs of developers to obtain the required performance characteristics based on the implementation of similar optimization methods in both APIs. The methods of graphic modeling, their implementation and methods of reproduction, various approaches to the optimization of graphic modeling for certain methods of visualization and processing of graphic objects are considered. Techniques such as using shaders, optimizing textures and buffers have been thoroughly reviewed and compared in terms of performance and how they are implemented in the OpenGL and Vulkan APIs.

Для моделювання будь-якої графічної сцени потрібно потужне графічне обладнання, а також API для її розробки. Саме вибір останнього та підхід до його використання грає вирішальну роль у відтворенні графіки за допомогою графічного обладнання. І відповідає за це саме розробник програми, в якої він хоче використати свої вміння розроблювати графіку.

На сьогодні є безліч інтерфейсів програмування застосунків (API) за допомогою для моделювання графічних моделей чи об’єктів. Велика роль відводиться саме розробнику, бо він пише код, реалізує логіку в програмі та оптимізує її певними функціями або способи, які пропонує обраний API. Але якщо інший інтерфейс пропонує схожий функціонал, але іншу реалізацію, то одне і теж моделювання такої самої сцени може дати різну статистику про рендеринг сцени, тобто швидкість моделювання одного кадру в секунду, зайнятість відеопам’яті тощо.

Таким чином, правильним рішенням є обрати такий API, який дає найбільшу продуктивність. Але для досягнення цієї мети потрібно докласти неабиякі зусилля, бо такі інтерфейси моделювання комп’ютерної графіки хоч і дають більше продуктивності, вони потребують від розробника більшого розуміння та досвіду в розробці рушіїв з використанням певного графічного інтерфейсу через те, що вони стають більш низькорівневими [1].

А чи варто обирати важкий для розуміння API для моделювання невеликих сцен або коли просто відтворюється звичайне вікно з кнопками? Це питання лежить на плечах розробника(ів), бо тільки вони визначають, чи потрібна для власного проекту висока оптимізація, чи через його невелике навантаження необхідність у ній зникає.

Одними із таких прикладів є інтерфейси OpenGL та Vulkan, перший з яких вже поступається другому в кількості можливостей та функціоналу, який можна використати для розробки графічних об’єктів. Можливо, це дійсно так, бо розробники API Vulkan, у певному розумінні, хотіли, щоб він став логічним продовженням OpenGL [2].

Наведені інтерфейси є двома популярними API для рендерингу графіки у відеоіграх та інших графічних застосунках, проте мають ряд відмінностей і застосовуються в різних сценаріях.

По-перше, OpenGL має відносно простий і зрозумілий API, на відмінну від Vulkan. Це може допомогти розробникам скоротити час розробки або навчання та підвищити продуктивність, особливо для менших проектів.

По-друге, ефективність та продуктивність у Vulkan більша, ніж в OpenGL, бо перший розроблений для високопродуктивних програм, щоб забезпечувати більше контролю над конвеєром графічного процесору і зменшити навантаження на процесор, що є критичним для вимогливих додатків, таких як ігри високого класу.

По-третє, OpenGL існує протягом багатьох років і має велику базу досвідчених розробників та ігор. Багато проектів інтегрують OpenGL у свої рішення і перехід на новий API може вимагати значних зусиль та витрат. Хоча, якщо використовувати його як базу для навчання з майбутньою перспективою перейти до Vulkan, а не інтегрувати у вже існуючий проект, це може бути для декого навпаки – перспективою.

Можемо зробити висновок, що Vulkan є більш низькорівневим та ефективним графічним API порівняно з OpenGL. Отже, останній більше не потрібен і його повністю замінив Vulkan? Насправді це не так. Хоч це і старіша технологія, вона все ще широко використовується у вже існуючих програмах і має велику спільноту розробників та користувачів [3].

Наше дослідження вирішує наведену проблему – коли потрібно використовувати OpenGL, а коли Vulkan. В залежності від вмісту розроблювального додатку: велика кількість текстур, розміщуваних об’єктів, графічних ефектів або можливостей; розробники віддаватимуть перевагу тому чи іншому API, якщо для них є пріоритетом мати менше навантаження на графічний або центральний процесор, менше витраченого часу або грошей на розробку, більшу кількість кадрів в секунду тощо.

Список використаних джерел:

1. Game engine basics. GDQuest. URL: https://www.gdquest.com/tutorial/getting-started/learn-to/game-engine-basics/ (дата звернення: 09.02.2024).

2. OpenGL vs Vulkan. That One Game Dev. URL: https://thatonegamedev.com/cpp/opengl-vs-vulkan/ (дата звернення: 09.02.2024).