2. Морзе Н. «STEM: проблеми та перспективи», Київський Університет імені Бориса Грінченка 19 серпня 2016р. - [Електронний ресурс]. Режим доступу: https://www.slideshare.net/ippo-kubg/stem-65590054.

3. Яськів С. «Що таке система освіти STEAM і чому це гарантія успішної кар'єри» - [Електронний ресурс]. Режим доступу: https://impactlab.media/2019/03/18/shho-take-stem/.

ОГЛЯД VULKAN APPLICATION PROGRAMMING INTERFACE (API), ЙОГО ОСОБЛИВОСТІ ТА ХАРАКТЕРИСТИКИ

Сеньків Арсен Ігрович
магістрант спеціальності 014.09 Середня освіта (Інформатика),
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
senkiv_ai@fizmat.tnpu.edu.ua
Струк Оксана Олегівна
кандидат фізико-математичних наук, доцент кафедри інформатики та методики її навчання,
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
oksana.struk@gmail.com

Vulkan API розробила Khronous Group, синтаксис даної API дуже подібний до OpenGL. У всьому API використовується префікс vk. Наприклад функції виглядають так vkDoSomething (…), імена структур чи Handle-ів: VkSomething, а всі константні вираження (макроси, макровиклики та елементи переліку): VK_SOMETHING. Також є функції, що мають особливий вигляд, до них добавляється префікс Cmd: vkCmdJustDo (…).

Писати на Vulkan можна як на C, так і на C++, звичайно другий варіант буде зручнішим. Є (і будуть створюватись) порти на інші мови [1].

Початок роботи та основні поняття. Vulkan розділяє два поняття – це пристрій (device) і хост (host). Пристрій буде виконувати всі команди, що йому відправлені, а хост буде їх відправляти. Фактично, наш додаток і є хост.

Для роботи з Vulkan знадобиться ідентифікатор (Handle) його екземпляру (instance), можливо не один, а також на пристрій (device), знову ж таки, не завжди може вистачати одного.

Vulkan у додаток можна завантажити динамічно. В SDK (розробили LunarG), якщо був оголошений макрос VK_NO_PROTOTYPES при завантаженні бібліотеки Vulkan власноруч (не використовуючи linker, а певними засобами в коді), то перш за все буде потрібна функція vkGetInstanceProcAddr, за допомогою якої можна дізнатися адреси основних функцій Vulkan – ті які працюють без екземпляра, включаючи функцію його створення, і функції, які працюють з екземпляром, включаючи його деструктор та функцію створення пристрою. Після створення пристрою можна отримати функції, які працюють з ним (а також з його дочірніми об`єктами) через vkGetDeviceProcAddr.

В Vulkan потрібно заповнити даними певну структуру, щоб створити об'єкт. В ньому це все працює приблизно таким чином: якщо дані заздалегідь підготовленні – то з ними можна працювати часто і з високою продуктивністю. В

інформацію про екземпляр можна помістити інформацію про ваш додаток, версії рушія, версії використаного API та іншу інформацію [2].

Шари та розширення. В голому Vulkan немає серйозних перевірок на вхідні дані. Коли він щось робить, то він це зробить в будь-якому випадку, навіть якщо це призведе до краху додатку, драйвера чи відеокарти. Це зроблено для продуктивності. Але можна підключити шари що будуть перевіряти дані і розширення до екземпляру чи пристрою, якщо треба [2].

Шари (layers). В основному, призначення шарів – перевіряти вхідні дані на помилки і відслідковувати роботу Vulkan. Працюють вони дуже просто: припустимо, викликаємо функцію, і вона потрапляє в верхній шар, раніше заданого при створенні пристрою або екземпляру. Він перевіряє все на правильність, після цього передає виклик у наступний шар. Так буде, поки справа не дійде до ядра Vulkan [2].

Розширення (extensions). Розширення розширюють роботу Vulkan додатковим функціоналом. Наприклад, одне розширення (debug report) буде виводити помилки (і не тільки) з усіх шарів. Для цього потрібно буде вказати необхідну Callback-функцію. Цей Callback може дорого обійтися, особливо якщо виводити всю отриману інформацію прямо в консоль. Після обробки повідомлення, можна вказати, чи передавати виклик функції далі (в наступний шар) чи ні. З їх допомогою можна уникнути критичних помилок [2].

Пристрій. Vulkan розділяє поняття фізичного пристрою і логічного. Фізичним пристроєм може бути ваша відеокарта або процесор, що підтримує графіку. Логічний пристрій створюється на основі фізичного: збирається інформація про фізичний пристрій, вибирається потрібне, готується інша необхідна інформація, тоді створюється пристрій. Може бути кілька логічних пристроїв на основі одного фізичного, але об’єднувати фізичні пристрої для єдиної роботи поки що, не можна. Інформація, яку потрібно зібрати це підтримувані формати, пам’ять, можливості та сімейства черг.

Черги (queue) та сімейства черг (queue family). Пристрій може робити наступні 4 речі: малювати графіку, проводити різні обчислення, копіювати дані і працювати з sparse memory management. Ці можливості представлені у вигляді сімейств черг: кожна родина підтримує певні можливості, якщо ідентичні сімейства були розділені, Vulkan представить їх як одне сімейство.

Після того, як було обрано потрібний (потрібні) сімейство(а), з них можна отримати черги. Черги – це місце, куди будуть надходити команди для пристрою. Черг і сімейств не багато. У NVIDIA зазвичай 1 сімейство з усіма можливостями на 16 черг. Після закінчення підбору родин і кількості черг, можна створювати пристрій.

Команди, їх виконання і синхронізація. Всі команди для пристрою ставляться в спеціальний контейнер – командний буфер. Тобто не існує жодної функції в Vulkan, яка змусила б пристрій зробити що-небудь відразу. При завершенні операції керування повертається додатку. Є тільки функції заповнення командного буфера певними командами (наприклад, намалювати що-небудь або

скопіювати зображення). Тільки після запису командного буфера на хості, його можна відправити в чергу, яка знаходиться в пристрої.

Командний буфер буває двох видів: первинний і вторинний. Первинний відправляється прямо в чергу. Вторинний же не може бути відправлений – він запускається в первинному. В ньому записуються команди в такому ж порядку, в якому були викликані функції. У чергу вони надходять в такому ж порядку. А ось здійсняться вони можуть майже «хаотичному» порядку. Щоб не було повного хаосу в додатку, розробники Vulkan передбачили засоби синхронізації.

Найважливіше: хост не очікує завершення виконання команд і командних буферів. Після відправлення командних буферів в чергу, відразу додаток отримує керування.

Є 4 примітиви синхронізації: паркан (fence), семафор (semaphore), подія (event) і бар'єр (barrier).

- **Паркан –** найпростіший метод синхронізації – він дозволяє хосту очікувати виконання певних речей. Наприклад, завершення виконання командного буфера.
- **Семафор –** спосіб синхронізації всередині пристрою. Його стан не можна переглянути чи почекати його на хості, не можна чекати його всередині командного буфера, але можемо вказати, який семафор повинен подати сигнал при завершенні виконання всіх команд буфера, і який семафор чекати перед тим, як почати виконання команд в буфері. Тільки чекати буде не весь буфер, а його певна стадія.
- **Події –** елемент «тонкого» налаштування. Подати сигнал можна як з хоста, так і з пристрою, чекати можна також і на пристрої, і на хості. Подія визначає залежність двох сетів команд (до і після) в командному буфері.
- **Бар'єр** – можна використати тільки в пристрої, точніше – в командному буфері, оголошуючи залежності першого і другого сету команд.

Конвеєри. Нижче показані два конвеєри Vulkan: (рис. 1).

Тобто у Vulkan є два конвеєра: графічний і обчислювальний. За допомогою графічного, можна малювати, а обчислювального – обчислювати. Результати обчислень можуть потім відправиться в графічний конвеєр. Так можна з легкістю заощадити час на системі частинок, наприклад.

Змінити порядок або змінити самі стадії конвеєра не можна. Виняток становлять програмовані стадії (шейдери).

Для конвеєра можна створити кеш, який можна використати в інших конвеєрах і навіть після перезапуску програми.

Конвеєр необхідно налаштувати і асоціювати з командним буфером, перш ніж останній буде використовувати команди конвеєра.

Render Pass, графічний конвеєр і фреймбуфер. Для того, щоб використовувати команди відтворення, потрібен графічний конвеєр. У графічному конвеєрі необхідно вказати render pass, який містить інформацію про під-проходи (subpass), їх залежність один від одного і їх вкладення (attachment). Вкладення – інформація про зображення, яке буде використовуватися у framebuffer-ах. Framebuffer створюється спеціально для певного render pass. Щоб почати прохід,

потрібно вказати сам прохід і framebuffer. Після початку проходу можна малювати. Після того, як малювання завершено, можна завершити прохід.

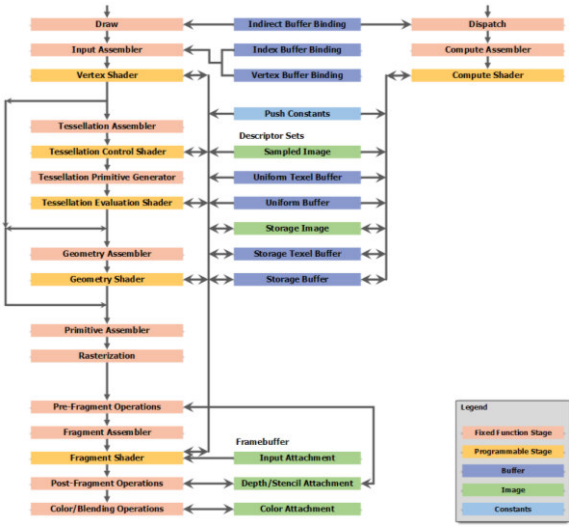

Рис. 1. Конвеєри Vulkan

Управління пам'яттю та ресурсами. Пам'ять у Vulkan розподіляється хостом (за винятком swapchain). Якщо зображення (або інші дані) потрібно помістити в пристрій – виділяється пам'ять. Спочатку створюється ресурс певних розмірів, потім запитується його вимоги до пам'яті, виділяється для нього пам'ять, ресурс асоціюється з ділянкою цієї пам'яті і тільки тоді можна копіювати в цей ресурс необхідні дані. Є пам'ять, яку можна змінити безпосередньо з хоста (host visible), є локальна пам'ять пристрою ну і також інші види пам'яті. Її вид впливає на швидкість доступу до неї. У Vulkan можна написати свій розподіл пам'яті хоста, налаштувавши Callback-функції. Але вимоги до пам'яті, це не тільки її розмір, але й її вирівнювання (alignment).

Шейдери. Vulkan підтримує 6 видів шейдерів: *вершинний, контроль тесселяції, аналіз тесселяції, геометричний, фрагментний* і *обчислювальний*. Написати їх можна на SPIR-V, а потім зібрати в байт код, який в додатку зкріпити в модуль, тобто створимо shader-модуль з цього коду. Можна написати його на звичному GLSL і потім конвертувати в SPIR-V (транслятор вже є). Також можна написати свій транслятор і навіть асемблер – багато даних викладені в OpenSource.

Вікна та дисплеї. Для того, щоб виводити щось у вікно або на екран – потрібні спеціальні розширення. Для вікон це базове розширення площини і розширення площини, специфічне для кожної з систем (win32, xlib, xcb, android, mir, wayland). Для дисплея (тобто FullScreen) потрібно розширення display, але те й інше використовують розширення swapchain.

Ланцюжок перемикань не пов'язаний з графічним конвеєром. Все досить просто. Є певний presentation engine, в якому є черга зображень. Одне зображення показується на екран, інші чекають своєї черги. Кількість зображень можна вказати власноруч. Є ще кілька режимів, які дозволять дочекатися сигналу вертикальної синхронізації.

Список використаних джерел

1. Sellers G. Vulkan Programming Guide: The Official Guide to Learning Vulkan (OpenGL) / G. Sellers, J. Kessenich.. – 480 с. – (Addison-Wesley Professional).
2. Lapinski P. Vulkan Cookbook: Work through recipes to unlock the full potential of the next generation graphics API - Vulkan: Solutions to next gen 3D graphics API / Pawel Lapinski.. – 702 с.

ПРОГРАМНІ ЗАСОБИ ДЛЯ РОЗРОБКИ 3D-МОДЕЛІ ВІРТУАЛЬНОГО ТУРУ

Тимочків Олександр Романович
магістрант спеціальності 014.09 Середня освіта (Інформатика),
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
tymochkiv_or@fizmat.tnpu.edu.ua
Генсерук Галина Романівна
кандидат педагогічних наук, доцент кафедри інформатики та методики її навчання,
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
genseruk@gmail.com

Із розвитком сучасного інформаційного суспільства неабиякої популярності набуває віртуальний туризм. Завдяки віртуальним екскурсіям, туристичні об’єкти стають доступнішими для різних категорій населення, а туристи можуть побачити те чи інше місце не виходячи з дому. На нашу думку, такі позитивні зрушення вимагають ретельного дослідження та вивчення, як і сам процес розробки й упровадження віртуальних 3D-турів.

Віртуальний тур – це реалістичне тривимірне зображення, що складається з різних тривимірних об’єктів і активних посилань-переходів (хотспотів). 3D-тур дозволяє побачити простір навколо себе і розглянути деталі навколишнього світу в найдрібніших подробицях, а також здійснити обертання і переміщення за віртуальним об’єктом. Сьогодні віртуальні технології постають однією із найбільш важливих і актуальних проблем.

Тривимірна графіка і анімація займають особливе місце серед комп’ютерних технологій. На ринку професійних програм до теперішнього часу лідирують програми комерційного поширення, але існує великий вибір і серед 3D-редакторів вільного (безкоштовного) поширення.

В процесі дослідження нами проведено характеристику редакторів 3D графіки, яку подано нижче.

Unity 3D – середовище для розробки інтерактивні ігор під операційні системи Windows, Mac, iOS, Android, Linux, Wii, Playstation, Xbox One і інших. Можливе написання сценаріїв на мовах JavaScript, C #, Boo. Розділ «Допомога» містить багато зразків з експлуатації софта. Готові додатки можна швидко протестувати в спеціальному тест-вікні. Користувач може додавати різні мультимедійні файли. У середовищі відсутні проблеми з сумісністю форматів відео, зображень і звуків, наявна функція імпорту моделей в COLLADA, FBX, DXF, 3DS.