Request for Quotations

Vulkan Tutorials

September 2016

1. Background

Vulkan provides high-efficiency, cross-platform access to graphics and compute on modern GPUs used in a wide variety of devices from PCs and consoles to mobile phones and embedded platforms. This ground-up design, complementing the OpenGL® and OpenGL ES™ 3D APIs, provides applications direct control over GPU acceleration for maximized performance and predictability with minimized CPU overhead and efficient multi-threaded performance.

In order to educate the developer community how to write efficient applications with the Vulkan API, Khronos wishes to create a series of tutorials and teaching materials.

2. Goals

The **primary** goal of this project is to create a series of **Vulkan tutorials** that demonstrate efficient use of the core Vulkan API across a range of GPU architectures. The tutorials must consist of example code and associated documentation. It should be possible to guide an audience of experienced 3D graphics developers through a given tutorial in a classroom environment within an hour. The documentation included with the tutorials should be sufficiently detailed that an individual can guide themselves through the series without supervision.

The **secondary** goal is to implement makefiles and/or a makefile generation system (such as Cmake) to enable users to build and deploy the tutorials to a variety of operating systems including Windows, Linux & Android.

The **tertiary** goal is to write an automated testing framework to determine if each tutorial renders correctly on a given platform (e.g. frame 0 golden image comparison). As development milestones are reached, this framework will be used by Khronos members to determine if the tutorials are compatible with their platforms.

The **quaternary** goal is to produce presentation material that will enable the series to be taught in a classroom environment.

Upon project completion the tutorial series will be made available under the standard Khronos open source free-use license for members and the general public to educate developers how to write efficient Vulkan applications. In the future Khronos may extend the series with additional tutorials covering advanced topics.

3. Scope

Audience

Users of the material are expected to be familiar with at least one of the following 3D graphics APIs:

- OpenGL
- OpenGL ES
- DirectX
- Metal

Technologies

The following technologies must be used:

- C++
- Vulkan-Hpp
- glTF
- glm

Readability is of the utmost importance. C++ features that can improve readability, such as range-based for loops, may be used as long as they are compatible with the compilers listed in the "Target operating systems & tool chains" section.

The series may have dependencies on other tools and libraries, such as GLSLangValidator, glTF exporters/importers and CMake, but they should be kept to a minimum.

3rd party libraries that abstract Vulkan functionality may be considered appropriate if their use is justifiable, i.e. if the concepts they abstract will still be adequately explained and their introduction will make the examples easier for the audience to follow.

Target operating systems & tool chains

The examples must be compatible with the following operating system versions:

- Windows
  - Windows 7
  - Windows 10
- Linux
  - Ubuntu 14.04 LTS
  - Ubuntu 16.04 LTS
- Android
  - Nougat

The examples must be compatible with the following IDEs/compilers:

- Windows
  - Visual Studio 2013
  - Visual Studio 2015
- Linux
  - gcc 4.8.4
  - clang 3.5
- Android
  - NDK r12b

Tutorial Content

In addition to the fundamental concepts of the Vulkan API such as devices, command buffers and graphics pipelines, the tutorials must cover the following:

- Window setup
- Layers
  - Introduction to the official Khronos layers
  - How to configure and enable
- Texturing
- Off-screen renders
- Synchronization primitives
- Keeping the GPU busy
  - Best practices for pipeline barriers, UBO multi-buffering etc.
- Secondary command buffers
  - For example, rendering the main scene to one secondary command buffer and the UI to another
- Pipeline cache objects

- Multi-threaded command buffer generation
- Sub-passes
- Queue priorities
- Compute

All topics that are essential for efficient use of the API - such as correct and efficient usage of synchronization primitives - must be introduced as early as possible. Each tutorial should build on the concepts and performance recommendations introduced in all previous exercises.

The series must include a tutorial that renders a textured mesh loaded from a glTF file. Subsequent examples that introduce more complex topics, such as multi-threaded command buffer generation, should build on the glTF example. Beyond this outline, the vendor is responsible for proposing the structure of the series.

4. Deliverables, Methodology and Acceptance Criteria

Methodology

Each tutorial should be implemented in a single .cpp file with minimal header dependencies. Custom and 3rd party libraries that abstract Vulkan functionality are allowed, provided the concepts are either clearly explained in a prior example (i.e. example *n* introduces concept *X*, example *n+1* uses a library to simplify the setup & use of *X*), or a suitable justification can be given to the audience (and the Vulkan working group) why all examples access the functionality via abstraction.

In addition to learning how to write conformant Vulkan code, the audience must understand the importance of optimally preparing and executing work to avoid unnecessary CPU and/or GPU stalls.

Tutorials must be compatible with all conformant Vulkan drivers. Required limits from the specification should be targeted rather than implementation specific limits. Optional features and extensions should be avoided. Required extensions, such as VK_KHR_swapchain & VK_KHR_*_surface, are allowed. If any limits are exceeded or optional features/extensions are used, the tutorial must have a graceful fallback path to ensure compatibility across platforms.

All tutorials must be able to run through the Khronos validation layers without generating any errors or warnings.

The series may be derived from existing material providing the goals outlined in this document are met. The license of source material must be compatible with the standard Khronos open source free-use license.

Tutorial documentation must be written in Asciidoc and conform to the Vulkan working group's writing style.. The documentation must be stored in the same repository as the tutorial code. All documents must be readable when processed by GitHub's asciidoc interpreter, i.e. users must be able to read the material on the GitHub website once the project is publicly released.

Presentations must be stored in the PowerPoint .pptx file format.

The automated testing framework must be developed in a separate private GitLab repository. It will not be included in the public release, as it is only required by the RFQ vendor and Khronos members. As precision differences between GPU architectures may account for slight differences in rendered images, comparison tests should pass if the rendered image falls within a suitable error tolerance from its corresponding golden image.

Deliverables

- Tutorials
  - Source code
    - Exercises - tutorial skeleton structure & boilerplate
    - Solutions - completed exercises
  - Makefiles
- Documents
  - Tutorial series introduction - expected audience, goals, tutorial overview, recommended reading & online resources (hardware vendor documentation, profilers & debuggers etc.)
  - Tutorial exercises
- Presentations
  - Tutorial series introduction
  - Tutorial exercises

Tutorial code must be hosted and developed on a private Khronos GitLab git repository. Point release branches must be created as milestones are reached, for example when a new tutorial has been completed. This will give Khronos members the opportunity to run automated compatibility tests against their drivers and share pass/failure results with the RFQ vendor. The RFQ vendor will be responsible for working with IHVs to resolve failure cases. IHVs may choose to loan or gift hardware to the RFQ vendor to speed up failure case investigations.

Once the series is complete, Khronos members will ratify the resultant material to determine if it is ready for a public release.

5. Selection Schedule and Process

Khronos will follow the schedule below to select a Contractor:

1. **September 27th** – Khronos Releases RFQ;

2. **October 17th** – RFQ responses received by Khronos;

3. **October 31st** – Contractor selected and notified;

4. **November 14th** – Contract executed and start of work.

The contractor will be selected from any received bids by the Vulkan working group. The selected contractor will be required to execute the standard Khronos Membership Agreement (with membership fees waived) if they are not already a Khronos member, and execute the standard Khronos Contactors Agreement with milestones and costs entered into Exhibit B and Contractor Disclosures entered into Exhibit C.

No work shall begin, and Khronos shall be liable for no costs or expenses, until the selected contractor is in receipt of an executed contractor's agreement.

6. RFQ Responses

The RFQ responses will form the basis for detailed milestone and cost negotiations for the final contract with the selected vendor(s). Vendors are encouraged to quote for a subset of the deliverables if they feel they are able to specifically address that subset as Khronos will consider splitting this project between multiple respondents.

Please provide the following information in the format of your choice:

1. Identification of which elements of the Project Scope outlined in Section 3 on which you wish to bid;

2. A description of your familiarity with C++, Vulkan and cross-platform development (Windows, Linux and Android);

3. Proposed tutorial structure;

4. Proposed schedule to complete deliverables, including when you are available to commence work and incremental milestones;

5. Whether you are bidding a fixed cost or a T&M contract. Fixed cost responses are strongly preferred;

6. Confirmation that you are willing to work under the Khronos Contractor Agreement and execute the Khronos Membership Agreement for the duration of the project;

7. Any particular issues or risk factors that you wish to highlight;

8. Supporting materials, including background materials about your company, highlighting relevant experience and expertise for this project, personnel backgrounds (e.g., resume and GitHub profiles), in particular success in delivering projects requiring close attention to detail and strong writing and communication skills.

Responses and all subsequent communication regarding this RFQ should be sent to: vulkan-tutorials-rfq@khronos.org.