

How to learn Vulkan

<https://www.jeremyong.com/c++/vulkan/graphics/rendering/2018/03/26/how-to-learn-vulkan/>

Why you learning Vulkan?

1. Vulkan performs better
2. Vulkan provide a more viable cross-platform solution(MoltenSDK is open sourced)
3. Vulkan is of course shiny things
4. Vulkan can be multithreaded programming
5. Vulkan is less about graphics, and more about GPU drivers at its core.
 1. Vulkan not provide how to cascaded shadow map works
 2. Vulkan is not screen-space-relections
 3. Vulkan is not how indirect lighting is done.
6. Vulkan is very broad and deep but it is logical
7. Learning Vulkan means jumping into a very large codebase.
8. Vulkan is best by using C++ instead of interpreted languages, why?
 1. The overhead of the extra function calls will add up and potentially offset the performance gains.

How to learn Vulkan?

How to adjust your mentality to best maximize your chance for success?

1. Don't discouraged if getting to this draw takes 5 to 10 times longer than you're used to.
 1. To bolster performances, as everything is opt-in
2. You need to constantly backtrack and review.
 1. To wonder why something is necessary?
 2. To wonder how something was done?
 3. Graphics pipeline is extremely deep, it's easy to lose the big picture once you get stuck in the weeds.
 1. You need to proactively pause and remind yourself what you have done and accomplished in order to get to where you currently are.
3. Many existing frameworks and engines proved existing functionality to Vulkan, make abstraction choices to be compatible with Vulkan.
4. Refer to the spec early and often.
 1. It in SDK docs folder.
 2. Useful information in the Vulkan Spec to be quite high and well worth the time invested.

Preliminaries

1. How the graphics pipeline works
 1. <https://fgiesen.wordpress.com/2011/07/09/a-trip-through-the-graphics-pipeline-2011-index/>
2. Vulkan in 30 minutes
 1. <https://renderdoc.org/vulkan-in-30-minutes.html>
3. Vulkan API without secrets
 1. <https://www.intel.com/content/www/us/en/developer/articles/training/api-without-secrets-introduction-to-vulkan-part-1.html>
4. Vulkan Tutorial
 1. <https://vulkan-tutorial.com/>
5. Vulkan Spec
 1. <https://registry.khronos.org/vulkan/specs/1.1/html/vkspec.html>
6. Vulkan example:
 1. <https://github.com/SaschaWillems/Vulkan>
7. Awesome Vulkan
 1. <https://github.com/vinjn/awesome-vulkan>
8. Vulkan Synchronization Primer
 1. <https://www.jeremyong.com/vulkan/graphics/rendering/2018/11/22/vulkan-synchronization-primer/>


Vulkan Mental Model

Import concepts

1. Shaders
2. Render passes and pipelines
3. Memory
4. Synchronization

Important concepts in Vulkan Mental Model

Shaders

1. Various layout options
 1. **location** specifier
 1. To recognize the inputs to the vertex shader corresponding to vertex buffers and vertex attributes
 2. **binding** and **set** specifiers, and the **push_constant**
 3. Some layout variables refer to arrays of data.
 1. **noise_textures** samplers
 2. **4 uniform buffer** objects
 4. Understand what all those mean?
 5. Understand which Vulkan calls correspond to which?
 2. API call(i.e. **VkWriteDescriptorSet**) related to descriptor binding to the location
 1. **Argus named as a buffer/uniform/sampler2D passed in API call will correspond to one of the layout options(binding,set,array index**
- 
2. Where to find the mapping between API and shaders? No such mappings, it's your own business to dig it out
3. Reading Docs, read word like “binding” “set” consider how you would access them in a shader
 4. Don't afraid do trying something new.
 5. Even if you're wrong, the experience doing so will be useful.