Vulkanised 2021 - Your Questions Answered

At Vulkanised 2021 we had an exciting line-up of Vulkan experts sharing their knowledge and experience. During the webinar attendees posted a wide range of great questions to the tutorial leaders, speakers and our panel of experts. These questions were answered online, and we hope the following summary of these discussions provides a useful resource for both attendees, and the wider Vulkan community.

Continue the Discussions: You'll find thriving Vulkan discussions on the forums and social media platforms below. It's a great way to get involved!

- Slack | GitHub | Discord | Stack Overflow | Twitter | Reddit | Khronos Forums
- Subscribe to the Vulkan Newsletter to receive the latest news and events.
- Check out the Vulkan Youtube Channel
- Visit the Vulkan Website

Day 1 - Tutorial Questions and Answers

The questions relate to the two tutorials that LunarG hosted:

- Ensure Correct Vulkan Synchronization by Using Synchronization Validation
- Building a Vulkan Layer in Symbiose Within the Vulkan Ecosystem

All answers provided by LunarG

1. **Are pipeline barriers the same as memory barriers?**
    a. **Answer:** Pipeline barriers allow the definition of several types of barriers. Pipeline barriers can define execution barriers (good for write-after-read), memory barriers (applying to all memory), buffer barriers (applying to a region of a given buffer), or image barriers (applying to a subresource range).

2. **Is it still interesting to use the original sync features or is it recommended to use sync2 from now on?**
    a. **Answer:** Sync2 is more explicit and specific, and allows the implementation more opportunity for the most efficient synchronization.

       Not all implementations currently support it, however there is a Sync2 layer allowing applications to use the Sync2 interface even where the implementation doesn't support it.

3. **What about write-write hazards in transfer commands? i.e. when there are 2 copies into the same buffer, unsynchronized.**
    a. **Answer:** They are only hazards if they write the same regions of the buffer. If they do write the same regions, then that is a hazard and requires a barrier to have predictable results.

4. **Do pipeline barriers synchronize only within a command buffer?**
    a. **Answer:** Pipeline barriers synchronize the hardware as the command buffer executes and once it completes the results are visible to other queues, main memory, etc. but fences and semaphores are usually still needed to synchronize with other queues or the host

6. **If I write a two subpasses in a renderpass and their srcStageMasks are BOTTON_OF_PIPE and their dstStageMask are TOP_OF_PIPE, will the draw commands in subpass 1 be executed after all commands in subpass 0 finished?**
    a. **Answer:** That will define an execution barrier between the two subpasses, but may not be sufficient depending on the memory usage in the two subpasses.

8. **The specification states that with VK_SHARING_MODE_EXCLUSIVE, ignoring the QFO barrier and just using a "normal" barrier leads to undefined content of the resource. On desktop, I did some tests and the content was always preserved. Is there a specific situation where this fails (e.g., for specifically testing for such an issue in an abstraction layer)?**
    a. **Answer:** Access to a resource after a QFO transfer from the source QF is undefined. Access before a QFO in the dest QF is undefined.

       Currently, validation doesn't complain... but it should.

9. **Is there any tradeoff when using barriers to synchronize render passes instead of subpass dependencies on desktop, if you have a single subpass per render pass?**
    a. **Answer:** The renderpass subpass dependencies give the implementation more information to allow more efficient synchronization. So using self-dependencies, when needed and unavoidable, is typically the recommendation.

10. **So for subpasses, should we also use pipeline barriers in addition to using BOTTOM/TOP_OF_PIPE or are just the pipeline barriers sufficient to order them? What's the best way to order my subpasses and if I also use memory barriers to say post processing how performat will it be?**
    a. **Answer:** You should avoid using 'wait for everything' barriers if possible, and prefer using subpass dependencies to barrier commands if possible.

11. **Did I get this right? Barriers generally affect commands on queue-level, i.e., also outside of a single command buffer's scope---but synchronization validation works only inside a single command buffer and hence, it would miss cross-command buffer hazards.**
    a. **Answer:** The current implementation is only command buffer scope. Work is in progress to expand coverage and catch these types of errors. This will be covered later.

12. **Do validation layers catch incorrect use of pipeline barriers? Such as in the case of indirect dispatch followed by an indexed indirect draw count? Where srcAccessMask & dstAccessMask are passed incorrectly.**
    a. **Answer 1:** Yes, parameter checks for pipeline barriers are part of core validation.
    b. **Answer 2:** Sync validation catches the cases where the parameters of the barrier are valid, but is not sufficient to synchronize your commands.

13. **Using VkValidationFeaturesEXT we can enable synchronization programmatically. John talked about synchronization validation being costly, and that either it should be enabled or the rest of validation should be enabled. I'm not sure how I would *disable* all other synchronization. Does setting VK_VALIDATION_FEATURE_DISABLE_ALL_EXT and enabling VK_VALIDATION_FEATURE_ENABLE_SYNCHRONIZATION_VALIDATION_EXT do the job?**
    a. **Answer:** The easiest way would be to use vkconfig's Synchronization preset. Which does this:

khronos_validation.disables = VK_VALIDATION_FEATURE_DISABLE_UNIQUE_HANDLES_EXT,VK_VALIDATION_FEATURE_DISABLE_OBJECT_LIFETIMES_EXT,VK_VALIDATION_FEATURE_DISABLE_API_PARAMETERS_EXT,VK_VALIDATION_FEATURE_DISABLE_CORE_CHECKS_EXT

khronos_validation.enables = VK_VALIDATION_FEATURE_ENABLE_SYNCHRONIZATION_VALIDATION_EXT

14. **If subpass dependencies with dst and src as VK_SUBPASS_EXTERNAL are provided for a renderpass, and I issue render commands using the render pass in two separate command buffers, then in the 3rd command buffer, we wait for these two cmd buffers using queue submit wait semaphore list. Will these renderpasses get synced at 3rd queue submit or after 1st render pass?**
    a. **Answer:** If the first 2 command buffers are submitted in 1 vkQueueSubmit*() call with a semaphore signal, they would both execute before the semaphore wait in the 3rd cmd buffer would complete

15. **Does validation support split barriers? (barriers via VkEvent)**
    a. **Answer:** Yes, vkEvent barriers are supported.

16. **What's the difference when we use volk instead of statically linking with vulkan.dll?**
    a. **Answer 1:** Volk will dynamically load vulkan.dll (or libvulkan.so on linux), then use vkGetInstanceProcAddr to query all Vulkan function pointers.

       Statically linking to vulkan.dll means letting the operating system's dynamic linker link the Vulkan functions an app uses to the functions definitions in the vulkan.dll (which then call into layers and drivers).

    b. **Answer 2:** Additionally, Volk is really a 'boilerplate reducer'. You can totally implement all of its functionality really easily. The difference is that it's a lot of boilerplate code (loading each function then storing it in a variable).

       It is recommended to load device function pointers with vkGetDeviceProcAddr, which Volk does for you, but it isn't necessary to use volk to do this. Both the Volk readme and the Vulkan Loader & Layer Interface document describe in more detail why you should prefer loading device functions. (links:

       https://github.com/zeux/volk

https://github.com/KhronosGroup/Vulkan-Loader/blob/master/loader/LoaderAndLayerInterface.md)

18. **I see device simulation layer is there. Does it allow a gfxreconstruct capture that has been captured on an Android device to be replayed on a desktop?**
    a. **Answer:** No. This isn't the function of devsim. Devsim does not do any actual emulation of "other hardware". This white paper is a very good write-up on devsim and how to use it: https://www.lunarg.com/wp-content/uploads/2021/09/Enhanced-Devsim-15Sept2021.pdf

19. **Is there any oversight from Khronos Group on implicit layers installed with some applications to ensure they don't create any issue that may be hard to detect?**
    a. **Answer:** There's no direct oversight, although we're working on defining clear behavior requirements for implicit layers in the near future. Validation of layers gets to be a real difficult problem as you have to test it against many different combinations. To help, we've added new debugging options to the Vulkan loader "VK_LOADER_DEBUG" environment variable to debug layer and implementation issues that can arise. If you set "VK_LOADER_DEBUG=layer", you will see what layers are used and if they are implicit, what environment variable is required to disable them. This is also turned on by setting "VK_LOADER_DEBUG=all".

20. **Can the “frame capture/replay” be used for performance analysis? I.e. figuring out how much extra time your engine used to actually produce the frame.**
    a. **Answer 1:** Frame capture/replay is hooked up to GfxReconstruct. So it's really intended to capture a frame or frames for replay from an application. You could try to do performance analysis on that frame, but it will be slightly different than your actual game timing. We do use these kinds of captures internally for performance analysis of various things like Validation Layers, but it's only because it produces a consistent replay state. For me, if you really want performance analysis, I'd suggest a hardware tool like NSight which I believe can break things down. Then you could use GfxReconstruct for individual frames, or just loop your application on problem frames.
    b. **Answer 2:** Playback is single threaded, and the CPU side will be driven by the playback engine performance, so there will be significant differences. Some elements will be the same, for example GPU side stalls based on over-aggressive synchronization operations.
    c. **Answer 3:** A common use of GfxReconstruct traces external to us is if an application has a bug on a certain driver, then you can capture the frames and

pass them along to a HW driver vendor for them to investigate rendering issues without having to send them your whole application.

21. **If I understand the diagram correctly, the Vulkan Configurator is writing out two text files. Where in the file system are those files being written to? Is the Configurator setting any environment variables as well?**
    a. **Answer 1:** Yes, VkConfig writes out two files: the override layer and the layer settings file. On Linux the settings file is written to $USER/.local/share/vulkan/settings.d, and the layer is written to $USER/.local/share/vulkan/implicit_layer.d.
    b. **Answer 2:** On windows, we store files in the following paths:

       %HOME%\AppData\Local\LunarG\vkconfig\override\VkLayerOverride.json tells a Vulkan application which layers to use

       %HOME%\AppData\Local\LunarG\vkconfig\override\vk_layer_settings.txt tells Vulkan layers which settings to use

       In addition, Windows system creates registry entries in the following locations:

       HKEY_CURRENT_USER\Software\Khronos\Vulkan\ImplicitLayers will have an entry that points to the JSON file above

       HKEY_CURRENT_USER\Software\Khronos\Vulkan\Settings will have an entry that points to the text file above

       HKEY_CURRENT_USER\Software\LunarG\vkconfig stores the application settings for vkconfig

22. **If you are using Configurator to launch an application, will it still pass on the messages to the app's vkDebugUtilsMessengerCallback?**
    a. **Answer:** Yes, messages still are sent to the callback

23. **Do you have plans on adding some verification mechanism to the loader so that the app can control what implicit layers are loaded? (DLL signing or something like this).**

    Many modern games have anti-cheat protection which blocks third-party libraries. They block all VK layers because there's no way to check whether it's a harmless layer or some cheat disguised as a harmless layer.

    a. **Answer 1:** This is a tough question. We haven't really thought about signing. That is something we can look into but won't be anytime soon.

However, game engines/applications will always be able to control their environment, and even if we implemented signing, it doesn't mean they would unblock those layers.

b. **Answer 2:** Currently, apps can disable implicit layers by setting the "disable_environment" variable which all implicit layers are required to have.

   Some implicit layers don't actually get enabled all the time because those layers have an "enable_environment" variable which has to be set for the layer to be active. Renderdoc, Steam overlay, & various situationally dependent layers will not be enabled unless the env-var is set.

24. **Does GFXReconstruct support write watch? I've been using that in my D3D12 APITrace project recently and it seems much faster.**

    a. **Answer:**

       commit c3c5be7338a477e0655d4573b2e66afbd3a25954[m

       Author: Dustin Graves <dustin@lunarg.com>

       Date: Tue Oct 22 18:49:36 2019 -0600

       Detect modified pages with GetWriteWatch

       When capturing with GFXRECON_PAGE_GUARD_EXTERNAL_MEMORY=1 on Windows, use the Win32 GetWriteWatch() function to obtain a list of modified pages. Replaces use of VirtualProtect() to generate exceptions on page write to detect modifications to mapped memory.

       Change-Id: I4cf7130895761fcd4d563217b53bdb0b4f8a40c6

25. **(Than write watch)**

    a. **Answer:** Yes

26. ***page guard**

    a. **Answer:** Yes

27. **What kind of operation can the layers perform ? I'm having trouble understanding what layers can and can't do, if I feel like something I want to implement can be a layer.**

    a. **Answer 1:** There is a lot you can do, and little you can't do. That doesn't mean a layer is the right solution for every problem. Without knowing the problem I can't say whether a layer is the right choice or not.

b. **Answer 2:** You can look here for some info: https://github.com/KhronosGroup/Vulkan-Loader/blob/master/loader/LoaderAndLayerInterface.md#layers.

28. **We had bad crashes from enabling Vulkan validation layers and debug markers on some platforms. Returned pointers seemed to have turned into handles that were sequential (1,2,3,4,..) instead of the actual pointers. When handed back to the Vulkan API/driver, these handles would then crash. This required disabling debug markers when using validation layers. I was told this was standard behavior for layers. Is this improved now?**

    a. **Answer:** I haven't heard of crashes about this in a while. However, we do wrap handles because there are certain handles that can be re-used in some situations which the Validation layers want to differentiate. That being said, the only reason you would get a crash is if you have validation layers enabled, but they're not wrapping/unwrapping the handle for certain extensions. Then, when your application would call those extension entry-points, they would pass down the "wrapped" handle as if it was the "unwrapped" handle. If you can reproduce, please create an issue on our Validation Layer repo.

Day 2 - Technical Talks and ‘Ask the Experts’

These questions and answers relate to the following individual presentations and panel discussion.

- Vulkan Fast Forward - New features and Directions
- Case Study: The NAP Framework and Vulkan
- Vulkan SDK and Ecosystem Updates
- The New Optimized ASTC texture Compressor (astcenc)
- Shader Compilation Pipeline in Baldur's Gate 3
- Vulkan Portability and MoltenVK - Layering Vulkan Over Metal.
- Android GPU Inspector 2.0 - Profiling at a Vulkan render pass level
- BasisU Texture Compression + KTX2
- Ask the Vulkan Experts - Open Q&A Session

Answers provided by the speakers and various Khronos members.

1. **We now have synchronization2 and also RenderPass2, do you expect to keep the numbers going up as the API evolves or will the "oldest" versions deprecate with time?**
    a. **Answer - Tom Olson, Arm:** Our commitment per the Vulkan spec, 1.x is always backward compatible back to 1.0. So we will never get rid of the old style synchronization. At 2.0, all bets are off. We do recommend usage of the synchronization2 API. When you can move to synchronization2 is not dependent upon when it is available on all devices. LunarG created an emulation layer for the synchronization2 API that can be used if the underlying hardware does not support the extension.

2. **RenderDoc isn't focused on gpu timing and doesn't really have a macOS build. AGI was the first tool that did this with Vulkan, but is only available on one out of 6 of my Android devices with Android 11. This need for cross-platform Vulkan perf and frame analysis tools is something missing. Metal GPU capture has in-depth capture, timings, counters, etc. AGI was great for cross-thread performance analysis akin to Xcode Instruments.**
    a. **Answer 1:** That's a good point. Performance profiling is heavily vendor/device-specific, so a cross-vendor and cross-platform profiler for Vulkan is a big task.

       The VK_KHR_performance_query may be a good base for such a profiler, but isn't broadly supported yet.

    b. **Answer 2:** Hi Sascha! I really appreciate all your samples and write ups on Vulkan. AGI uses perfetto to display the timings so that might be one fast path. Simply being able to dump vs/fs perf timings and see them visually with names and hover timings, and see them across multiple frames would really be useful. That's most of the insight from AGI and Instruments.
    c. **Answer 3:** I recommend filling a bug/feature request. While that might seem like a standard answer; when/where tools aren't working is exactly the kind of information that is used to determine how to improve the tools. It's hard to provide a more complete answer in a live Q/A session; but these are very important issues to raise.

3. **May I know the trends of OpenCL and Vulkan, which one should be used for computing tasks in the future?**
    a. **Answer 1 - Ralph Potter, Samsung:** Both are used, OpenCL is older and more widely used. Because it is older, OpenCL will have more documentation and examples. If you're looking to learn from scratch, that might be a better starting place. If you're looking for Android support, OpenCL is not officially supported on Android as far as I can remember. I know we are looking at compute on purely Vulkan for a few applications.
    b. **Answer 2:** As Mark said, it depends on your goals. Vulkan lets you reach a lot more devices and platforms, if that is important to you. The price of that is that it's harder to use and you are likely to be less productive initially. I will say that from the Vulkan Working Group's point of view, making Vulkan work well for compute is very important. But TBH that is mostly driven by the trend in game engines toward doing more and more work in compute shaders. We're paying less attention to traditional scientific computing, so if you are working

c. **Answer 3 - Ralph Potter, Samsung:** OpenCL is very established on a wide range of devices, some of which don't support Vulkan. OpenCL is a more accessible API than Vulkan, it gives you less control, but you may be productive more quickly. Having said this, we do spend a lot of time on Vulkan compute as well. OpenCL is not going away, it may be the correct choice for you. If you are trying to combine graphics with compute, maybe Vulkan compute is the better route. There isn't a "one size fits all" answer.
d. **Answer 4:** OpenCL also has an expensive context swap that compute shaders don't if you have raster/compute interop.

   Follow-up Question: In terms of availability on platforms, which is the better choice, OpenCL or Vulkan?

   i. **Answer - Tom Olson, Arm:** Vulkan is certainly more available on mobile. Although there are multiple implementations of OpenCL shipping, OpenCL is not officially supported in Android. You can expect support for computing in Vulkan to improve over time.

4. **Do you expect Vulkan to live with OpenGL or to completely replace it in the next few years as the main cross platform graphics API, even in courses, where Vulkan can be more complicated than OpenGL to learn the basis of computer graphics?**

   a. **Answer 1 - Tom Olson, Arm:** For a first course in graphics, Vulkan introduces a lot of issues that just get in the way of learning. I think the web APIs may offer an interesting direction for teaching basic concepts. Vulkan would be good for a second course with a real-time focus.

5. **Can we completely trust GLSLangValidator for SPIRV to GLSL conversion -> slight modification to shaders <- later GLSL to SPIRV and deploy shaders**

   a. **Answer 1:** I don't know of any issues here. I would recommend trying it and if an issue occurs file an issue on GitHub.
   b. **Answer 2:** We use GLSLangValidator internally and haven't run into problems with trusting it. Issues are well addressed in a reasonable time frame. I would recommend using it.

6. **I’m trying to use float16_t and finding it difficult. Many GLSL constants cast to half, missing float16_int8, 16bit_storagte and or 16bit_storage inputOutput16/uniformConstants16 ext support on Nvidia/Adreno and many desktop devices. No real samples of how to write around all of these issues, and an assumption that all extensions are present and fully true for all 5 booleans. Transpiled MSL from mediump doesn't translate to half from mediump. Existing compile options do force all "Relaxed Precision" math in fp16 without analysis of precision. And no easy interop from mediump to float16_t.**
    a. **Answer 1:** Adreno currently crashes in driver on all float16_t usage with "unsupported rounding mode" on RTZ commands. Mali has full 16bit_storage support. MSL doesn't have any of this, half and float just work and makes shader writing simple. And GLSL's need to cast doesn't work with #define half mediump float, since it's not a true type.
    b. **Follow-up question:** I don't feel like Khronos has really dogfooded half support, and it was missing from Vulkan 1.0. My ask is for more realistic examples of how to write shaders that utilize this. This also points to the limits of Vulkan not having a designated shader language, and hacking GLSL has limits. Reading transpiled spri-v also strips all comments from the original source code.
        i. **Answer 2:** Thanks Alec, that's good input. Ideally you'd raise the lack of samples issue at the Samples repository, though I will remember that you've asked. The question of GLSL future, we'll try to cover in the panel. Oh, and: the Adreno issue runs into the problem that mobile GPU drivers are hard to update. If the behavior is a bug, it's quite possible that Qualcomm has fixed it, but the fix isn't available on the devices you're working with. Unfortunately that's outside our control, though I know Google is trying to fix it. We will certainly take an action to look at CTS coverage for half types.
    c. **Follow-up question:** I don't really care about the API (Vulkan, DX, Metal) as much as I do making shader writing and transport simpler. There is a lot more that can be done on the Vulkan compile tools to improve this, and encourage fp16 usage. It's clear Adreno is running float16_t code since it breaks the driver despite exposing the extension, but will indeed fix this in their next driver. But we can't now use that construct since we also need inputOutput16 support. Nvidia is also missing this extension. They encourage mediump, but then we don't get half in MSL.
        i. **Answer 3:** At some point, we should be transporting materials and shaders in GLTF3?, and so more transportable (and performant) shader languages are an important step towards that.

7. **What do we mean by the synchronization layer emulates the sync2 API?**
    a. **Answer:** There is a layer that can be enabled that translates the Sync2 calls into Sync1 calls if the underlying implementation doesn't support Sync2.

8. **Any plans for something similar to vkconfig for the Android ecosystem? I had so many issues related to random Samsung devices saying that my shader has issues without any further info. A debug tool can be very helpful. There is support for the debug layer already in Android I believe, but using that requires us to place the debug layers in certain locations and also set environment variables. An app-like thing would really be helpful to configure these.**
    a. **Answer:** https://developer.android.com/ndk/guides/graphics/validation-layer talks about how to use validtion on Android. We also create binaries for every SDK release which you can find here https://github.com/KhronosGroup/Vulkan-ValidationLayers/releases

9. **We now have Debug Printf, this is nice, is there any progress on a **debug assert?****
    a. **Answer 1: Mark Young, LunarG** There is currently no plan or progress on this.
    b. **Follow up question:** @Mark Y, Can you expand on that? Is there a particular reason? Given that this feature was requested almost immediately after the announcement of Debug Printf, and I keep seeing it popup when talking about Debug Printf, I don't imagine community *want* is the issue.
        i. **Answer 2 - Mark Young, LunarG:** Perhaps we are not understanding your request. However we believe that the assert feature you are interested in is not an enhancement to debug Printf but rather a new feature to spirv-tools. You say this was requested after the release of debug printf. Did you submit a github issue? I would highly recommend getting this enhancement request into a github repository (probably spirv-tools)
    c. **Answer 3 - Tom Olson, Arm:** Assert is tricky to even define on a massively multithreaded engine that is only loosely coupled to the timeline of your application. If one fragment thread asserts, what happens to others? When does the application find out about it? But that's not to say it isn't a good idea. It's sort of related to the "current hot topic" I talked about, of how to debug DEVICE_LOST situations. If we are able to create a way to do something sensible to capture and report when the GPU hits a memory fault, it becomes thinkable to turn that into an SWI from a shader - and then you have

something like assert. So I would say, it could happen, but there are sub-problems we have to solve first.

10. **Old captures of gfxreconstruct are broken with new tools or drivers. Is there any effort to fix backward compatibility issues?**
    a. **Answer:** Yes, we are aware that a driver update can sometimes break a previously made trace. It is in the design goal to handle these situations but I won't claim that all the issues are currently addressed! It would help if you can submit an issue on the GFXReconstruct github. Also note that GFXReconstruct is still at a 0.9x release version. We consider version 1.0 the time when we can be more disciplined about these backward compatibility issues and will be working to get it to 1.0 functionality/quality over the next several months.

11. **What do you add to your Vulkan application to use the Vulkan configurator files?**
    a. **Answer 1:** VkConfig works outside of your application with the Vulkan loader and any layers. You can watch this: https://www.lunarg.com/introducing-the-new-vulkan-configurator-vkconfig/#:~:text=The%20Vulkan%20Configurator%2C%20also%20called,system's%20Vulkan%20implicit%20layer%20configuration.
    b. **Answer 2:** Install the SDK and run vkconfig. No changes to your application needed.

12. **Is the ASTC texture format renderable, or can it only be used for input textures?**
    a. **Answer 1:** Like all other compressed formats, ASTC is not renderable
    b. **Answer 2:** You can alias the format as a non compressed format (e.g. UINT) and write to it that way, but you need to handle compression of blocks yourself.

13. **I'm curious how we transfer ASTC 3D textures with 3D blocks (Android) vs. 3D textures with 2D blocks (iOS)? There seems to be disagreement about support for 3D blocks in the HDR spec. GLTF2 only has KTX2 support for Basis, but I want to store final assets with ETC/ASTC/BC format in glTF2 and KTX2 format.**
    a. **Answer 1:** There are two modes for supporting 3D textures in the API (and separate extensions for both). Sliced 3D textures, which are basically stacks

of 2D textures, and volumetric 3D with true 3D blocks. Mali supports both, but there isn't wide adoption for volumetric blocks from other IHVs.

IIUC, there will be some differences between the two (sliced 3D reconstructs using bilinear interpolation between slices, volumetric 3D uses simplex interpolation between the slices). However, not sure exactly what issue you are referring to. Happy to look offline - just drop me a mail with the details.

b. **Answer 2:** We have now added ATSCEnc as a backend for the Khronos toKTX utility (currently LDR only, but HDR in the works).
c. **Answer 3:** GLTF2 only supports 2D KTX2 files with basis, so it hasn't had to address the 3D block vs. 2D stack issue yet. This has ramifications for Basis transcode and how to store textures for portability. If we could always just assume stacked ASTC then that could be the transport format.
d. **Answer 4:** Practically, if you want portability, stacked 3D is probably the way to go. I'm not even sure if volumetric 3D has a Vulkan extension at the moment.

18. **In Gfxrecon captures, some captures work across platforms, like the capture captured on Android also works on desktop but there is some problem when the resolution of the device is quite high and when playing on desktop the entire screen is taken up and we aren't even able to minimize it. A way to just be able to dump a screenshot without the window would really be helpful. Also, a way to adjust the viewport or window without cutting the output would be helpful.**
    a. **Answer 1:** The best place for feature requests is at the project page https://github.com/LunarG/gfxreconstruct
    b. **Answer 2:** It would be very helpful if you could post your request to the GFXReconstruct repository issues. https://github.com/LunarG/gfxreconstruct/issues/588
    c. **Follow-up Comment:** I understand that all the issues cannot be addressed immediately, just wanted to point out that it would really be helpful if this request is addressed.

19. **As an ecosystem question, are any language improvements to GLSL planned in the future? E.g. better user defined type support, like operator/built-in-function overloading, so custom types, like quaternions, are easier to create and use? It seems that no functional *language* improvements have been made to GLSL itself since the release of Vulkan, despite a plethora of GIT issues requesting such improvements.**
    a. **Answer - Tom Olson, Arm:** A fair point. GLSL is being actively maintained, but no one has been driving it forward. The OpenGL / GL ES group technically owns it, but they are in maintenance mode and not likely to add new features.

Vulkan is the most active user, but Vulkan made a deliberate choice to define its interfaces in terms of SPIR-V, so that developers can use any language that maps to that. We think that was the right choice and are not going to change it. But we do *care* about GLSL and want to see it move forward. This is under high-level discussion within Khronos right now.

20. If bindings are auto-assigned, do you have an unique Pipeline Layout for each shader?

a. **Answer:** Yes and no! VK_DESCRIPTOR API is still available and works. We generally prefer to have auto-assigned bindings for engine shaders (that's not a lot of them and they are executed rarely compared to materials ones) and predictable well-define bindings for material's shaders (there are a lot of them and they are generated anyway)

21. What is the performance tradeoff while using MoltenVK over native Vulkan drivers?

a. **Answer 1:** This is a complicated question to answer because it depends on how the app is using Vulkan features and how those features map to Metal.

   The best answer is to advise you to try to run your app on the platform you want to target, and see if your app performs well enough for your needs.

   MoltenVK is used satisfactorily by a number of AAA titles.

b. **Answer 2:** It would be useful to gather those numbers from an Intel-based mac that dual-boots into Windows. It would run the Dota2 benchmark on macOS and on Windows and compare the numbers directly.

22. What is the overhead of using MoltenVK versus Metal directly?

a. **Answer 1 - Bill Hollings, Brenwill:** This is a complicated question to answer because it depends on how the app is using Vulkan features and how those features map to Metal.

   The best answer is to advise you to try to run your app on the platform you want to target, and see if your app performs well enough for your needs.

   MoltenVK is used satisfactorily by a number of AAA titles.

b. **Answer 2:** We measured this in gfx-portability a few years ago, while benchmarking Dota2. Of all the time spent in Vulkan implementation, roughly 75% was attributed to the Metal driver. So one can estimate the CPU overhead to be 25-30%.

23. **What would be the best course of action when I'm perf limited by MoltenVK? Vulkan->Metal command encoding? Is the buffer prefilling option on a worker thread the right way to go?**
    a. **Answer 1:** MoltenVK doesn't do much during command recording. All the meat happens at submission. So you may try doing queue submissions on a separate thread, if MoltenVK doesn't do this internally already.
    b. **Follow-up question:** Yeah without that option enabled, it does encode in submit indeed, but as I understand, its queues need to be externally synchronized, so I can't submit on multiple threads simultaneously.
    c. **Answer 2:** You can file a feature request on MoltenVK to record command buffers "live" as you submit Vulkan commands. That would allow proper multi-threaded recording. ("gfx-portability" has that as default behavior)

24. **There was guidance provided in the "Android GPU Inspector" session that we should "split position attributes". I'm unclear on what that means in practice. Separate vertex buffer from other attributes? Or is this a reference to interleaved attributes perhaps?**
    a. **Answer 1:** Mobile GPUs will often use just the position attribute to figure out what geometry is visible, and only need the remaining attributes for geometry that is actually visible. Densely packing position (by putting the other attributes in a separate buffer) is often a performance win
    b. **Answer 2:** Also this approach improves efficiency for depth-only passes (e.g. shadow maps), where only position is needed.
    c. **Answer 3:** Also ray tracing requires all positions to be stored in float3 in a single buffer, so I've stopped compressing those.
25. **There are lots of Vulkan resources for 3D, but are there equivalent resources for those of us doing 2D/CAD type things? I.e., drawing pixel based things at vertex coordinates, lines with miters and joins, drawing cubic paths which result in lots of degenerate triangles, Vulkan equivalents to NV_path_rendering, shaders to do font rendering).**
    a. **Answer 1:** Loop-Blinn rendering is quite complex in terms of how you process your curve data. Neither Microsoft nor Nvidia ever open-sourced that very amazing extension so it could be implemented on any other platforms. But there's a lot more that should be done to draw 2D SVG and vector data quickly in 3D. We are still drawing text from atlases on GPUs.
    b. **Follow-up question:** Agreed. This is precisely why I was asking. These things really aren't something that end users really have the ability to deal with. Mark Kilgard's Siggraph 2020 Polar Stroking presentation is a good example--complex, with lots of degenerate triangles, not something end users should be assembling. https://developer.nvidia.com/siggraph/2020/video/sig03-vid
    c. **Answer 2 - Tom Olson, Arm:** Granted it's not something you want to code up over the weekend to replace your texture-based glyph atlas, but it isn't really hardware functionality either. That puts it pretty much out of scope for the Vulkan API. As a rule, if something can (at least in principle) be done by the app just as well as the driver could do it, we won't put it into Vulkan. If it's possible but just too complex for the app, it belongs in middleware.

26. **Hi Sascha! I really love www.gpuinfo.org. Pete Harris pointed me to this. It would be nice to be able to search by GPU (f.e. Mali G76) there. There are so many phone names, but only a few dominant Mali and Adreno devices on Android.**
    a. **Answer 1 - Sascha Willems, Independent:** Thanks for the praise. That's a great suggestion. In the early days, I only displayed GPU names and you could search for them, but moved to displaying device names.

       I'll see how I can add this back into the database, so you'll be able to search for GPU names again :)

29. **What is being done on the portability initiative regarding PlayStation? News about it always seems to avoid referring to Sony consoles.**
    a. **Answer - Dzmitry Malyshau, Mozilla:** Nothing so far, unfortunately
30. **Not specific to Vulkan at all, so.. sort of off topic, but perhaps Piers could volunteer any insight into the future of SPIR-V support from vendors with OpenGL? (Support and functionality across vendors seems quite patchy, last I tried).**
    a. **Answer 1 - Piers Daniell, NVIDIA:** OpenGL 4.6 supports SPIRV 1.0 and it could be expanded to support later versions of SPIRV with extensions. But currently there is baseline SPIRV support in OpenGL
31. **Proprietary APIs have very nice beginner friendly SDKs, like DirectXTK and SceneKit. Are middleware engines the only viable option for beginners on Vulkan?**
    a. **Follow up question:** Related to the question, do you expect Vulkan to live with OpenGL or completely replace it (even with Vulkan being much more difficult; especially for learning graphics)?
    b. **Answer 1 - Karen Ghavam, LunarG:** Why doesn't the SDK have more tutorial and learning materials? It was a conscious decision to have the SDK only include tools for developing applications and to not be a tool for teaching Vulkan. Instead there are many tutorials, samples, etc. available out in the ecosystem for learning Vulkan.
    c. **Answer 2 - Tom Olson, Arm:** Khronos does have a DevRel organization (one person part time), so now we have the KhronosGroup/Vulkan-Samples repository which we didn't have before and we would like to expand that. We realize we are not up to the level of the closed, proprietary APIs. Long term it is a goal to improve the learning tools, whether it is done directly by Khronos or by other parties. Is Vulkan a great API to learn graphics from? No, not for learning basic rendering. I would not teach a beginning graphics course and use Vulkan. I might introduce it at the end of the course to get folks thinking about GPUs. OpenGL continues to exist and is being maintained (thank you Piers), but we are not adding a lot of new functionality to OpenGL.

d. **Answer 3 - Ralph Potter, Samsung:** The right choice of API for learning or development is going to be heavily dependent upon what you are trying to do. At Samsung we would love to see many Vulkan AAA titles but OpenGLES is not going away. If what you need is a quick way to get some triangles on the screen, OpenGL may be a better choice. Longer term, we are seeing many things being layered on top of Vulkan. Down the line it is very possible that your OpenGLES is calling Vulkan anyway. But the correct choice of API for learning and developing apps is very dependent upon how much you are willing to invest and what types of devices you are targeting.

32. **Have there been any developments regarding memory usage? For example, I remember the recommendation used to be to just use 80% or so of the available GPU memory (kind of a fuzzy heuristic).**
    a. **Answer 1 - Tobias Hector, AMD:** It sort of depends upon your platform. On desktop 80% memory usage may be ok (if you are running a full screen game and nobody else is using up your memory. 80% usage on mobile would probably not be good. You always need to leave some memory available for the system to avoid constant paging in and out. Always profile on your target devices.
    b. **Answer 2 - Piers Daniell, NVIDIA:** There was an extension released recently, VK_ext_pageable_device. If you use this extension, it allows you to make the best use of advanced paging and you can allocate more memory than what is immediately available

33. **Any news on mesh shaders or another standardized compute-like way to create groups of primitives that directly go to the rasterizer?**
    a. **Answer 1 - Tom Olson, Arm:** No question, the fixed function geometry pipeline as we have known it over the years is on its last legs and we are going to be moving to a more compute-like paradigm, emitting geometry into rasterizers. Exactly what that is going to look like is TBD. Mesh and task shaders are one possibility, but it's too early to say if it is the right answer. Task and mesh shaders are pretty immature and there isn't a lot of content yet. There are some real issues in portability of implementations; as they come online you are likely to find that all of the vendors are making conflicting recommendations for best practices for performance. So I expect it to be a painful adoption process. We will continue to work on it, and we are quite interested in it. We have other things on the front burner currently but I expect we'll be coming back to "what does the future geometry pipeline need to look like?" before long.
    b. **Answer 2 - Tobias Hector, AMD:** I can't disagree with what Tom said. There is a desire to support mesh shading for DX emulation and direct porting, so you

have that ability to have feature parity. Can't communicate timelines at the moment.

34. **I want to get into Vulkan. What do you recommend to start learning Vulkan?**
    a. **Answer 1: Kris Rose, The Khronos Group** - www.vulkan.org has a bunch of resources to help you get started but in particular www.vulkan-tutorial.com is a good place to start with the basics.
    b. **Answer 2: Mark Young, LunarG** - I think there are multiple ways, but I would say online/Github is best since the API moves quickly.

       Start with an online tutorial (both https://vulkan-tutorial.com and https://vkguide.dev are good).

       After that, start looking at Vulkan Samples (https://github.com/KhronosGroup/Vulkan-Samples).

       There's many resources on Khronos Vulkan website. Even this (https://www.khronos.org/blog/beginners-guide-to-vulkan).

35. **Any plans to encourage developers to use multipass more rigorously such as in benchmarks & games?**
    a. **Answer 1 - Tom Olson, Arm**: I had in my list of "hot topics", fixing Render Passes.
    b. **Answer 2 - Tobias, Hector, AMD**: We hear from all developers that multipass can be useful on tiled based GPUs. However it is high effort for not a lot of gain in many cases. It certainly trips up porting efforts of moving existing engines into Vulkan. We are not necessarily discouraging multipass approaches, but we are looking at alternative ways of expressing them so they are simpler to use and getting it out of the way when you don't want to do multipass and at some point there is going to be a change in the ecosystem about what does multipass really mean

36. **With the Vulkan Video Extension, how related is it to the manufacturers HW accelerated APIs (ie NVEC or AMF) wrt features?**
    a. **Answer 1 - Tony Zlatinski, NVIDIA:** Vulkan Video API, like the rest of Vulkan, is a low-level thin API wrapper on top of the HW. The difference from the NVDEC/NVENC is that those Video SDKs have a number of CUDA utilities on the top of the encode/decode HW. Assuming somebody knows how to replicate those utility algorithms it would be possible to implement the same functionality on the top of Vulkan Video. NVIDIA has a long-term plan to support the same CUDA utilities on top of Vulkan Video.

I'm not familiar with AMF, but as a higher-level model API, it should be possible to add support for Vulkan Video.

b. **Answer 2 - Ahmed Abdelkhalek, AMD:** Yes, as Tony mentioned Vulkan Video is low-level and currently focuses on basic decode/encode codec-specific support for all IHVs. So feature-wise it may be considered a subset of higher-level vendor-specific APIs/SDKs like NVDEC/NVENC/AMF/etc., especially when it comes to encode where features which may be orthogonal to the specific codec are developed to enhance encoding results.

   It’s possible to develop vendor-specific extensions that extend Vulkan video to bring features up to the level of NVDEC/NVENC/AMF/etc. Similar to NVIDIA, AMD also has long term plans for this.