LunarG improved the debugging functionality of the Vulkan API by creating a new extension: VK_EXT_debug_utils. We developed a comprehensive tutorial on the use of the new extension. Read on for more details.

The Vulkan API is now two years old, and as with all things, it is showing areas that require improvement. Debugging is one of the areas where we can make small changes that produce a large benefit for the Vulkan community. After soliciting input from IHVs and several game companies, and reviewing feedback from GitHub users, we decided to improve the debugging functionality exposed by both VK_EXT_debug_report and VK_EXT_debug_marker. As we investigated, we decided that replacing the extension was the proper decision going forward instead of trying to shoehorn new functionality into the existing extensions. The changes resulted in the creation of a new extension: VK_EXT_debug_utils.

Why the New Debug Extension?

The Vulkan Working Group received feedback from developers at several software companies asking for more information from each debug message to help them isolate the trigger in their own code. Validation messages created a special concern since an application can create multiple Vulkan objects, and only one of those objects may be handled incorrectly.

To help software developers isolate issues more efficiently, LunarG decided to combine the functionality of VK_EXT_debug_report and VK_EXT_debug_marker to produce more useful debug messages. However, while attempting to coordinate the work between these two separate extensions, we recognized a fundamental problem. VK_EXT_debug_report is an instance extension, while VK_EXT_debug_marker is a device extension, and there is no easy and clean way to indicate that functionality in an instance extension is dependent upon a device extension being present and enabled. To simplify things, we decided we could just define a new instance extension that supplied all the necessary items in one place.

We also expanded the information that is returned to a user's debug callback. This change could have been made with the old extensions, but it would require adding items to the pNext chain of most structures. While doable, it added more complexity than we thought was worthwhile since every debug callback would have to care about the pNext chain. Of course, we still may add functionality to the new pNext chain in the future.

Finally, the VK_EXT_debug_report extension uses a special internal enumeration to track object types, VkDebugReportObjectTypeEXT. This enumeration was supported for a time and even used by the VK_EXT_debug_marker extension. However, the latest versions of the Vulkan spec replace this structure with a new core object type enumeration, VkObjectType. Due to this spec change, Khronos decided to stop expanding VkDebugReportObjectTypeEXT and instead support adding new enumeration values only to VkObjectType. Consequently, the VkDebugReportObjectTypeEXT enumeration will grow stale over time.

With all these factors under consideration, LunarG decided to create this new Vulkan debug utility from scratch.

Read the Tutorial

LunarG created a comprehensive tutorial to share the benefits of the new extension and how to use it. We also provided application usage examples for your review. If you are interested in learning more about the new Vulkan Debug Utilities extension, check out the tutorial.

Click here to view the tutorial describing the new extension.

New extension available in LunarG's Vulkan SDK

This new Vulkan Debug Utilities extension can be found in the LunarG SDK, available on the **LunarXchange website**.

Info about LunarG, Inc.

LunarG's software engineering experts create innovative products and services for open source and commercial customers using leading-edge 3D graphics and compute technologies, including OpenGL, Vulkan, OpenXR, and SPIR-V. We have strengths in performance analysis and tuning, runtime and tools development, shader optimizations, driver development, and porting engines and applications to Vulkan. Our software engineering team is based in Fort Collins, Colorado. LunarG was founded by software experts who are passionate about 3D graphics.

For more information about LunarG, check out our **website**.

By Erika Johnson | May 18th, 2018 | News