

Vulkan SDK Update

Siggraph Asia 2024 - Tokyo

Spencer Fricke

LunarG, Inc



About Me

- Been at LunarG for 2+ years
- Vulkan Validation Layer tech lead
- Previously lived in Japan for 2 years
 - Now located permanently back in the USA
- Active member of Vulkan Working Group
- Help maintain various other parts of ecosystem
 - Vulkan-Guide
 - SPIRV-Visualizer
 - SPIRV-Guide
 - SPIRV-Reflect

What drives the Vulkan ecosystem?

What drives the Vulkan ecosystem?

All of you!

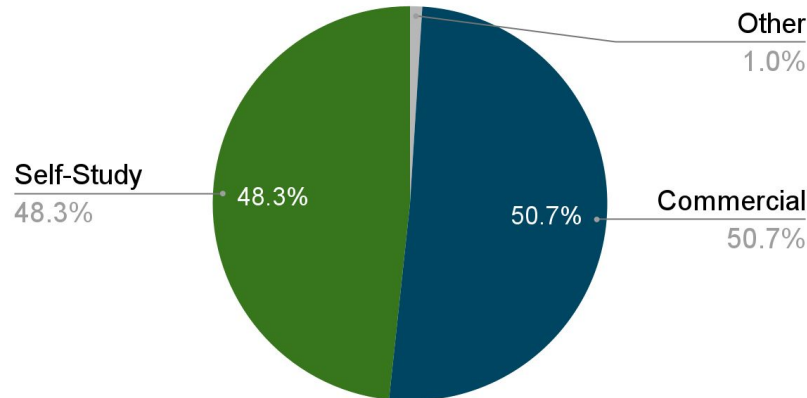
Vulkan Developer Ecosystem Survey

Helps drives LunarG Ecosystem Priorities!

What we heard -

- Shader tool chain - needs dev & maintenance
 - 60%+ glsl -> SPIR-V
 - 20% use DXC
- Validation Layers
 - Increase coverage
 - Readability / interpretation of error messages
 - Improve performance
- MoltenVK
 - Move forward faster
- Difficult Tasks
 - Identifying driver defects
 - Debugging layer issues
 - Debugging install & configuration issues

275 Respondents



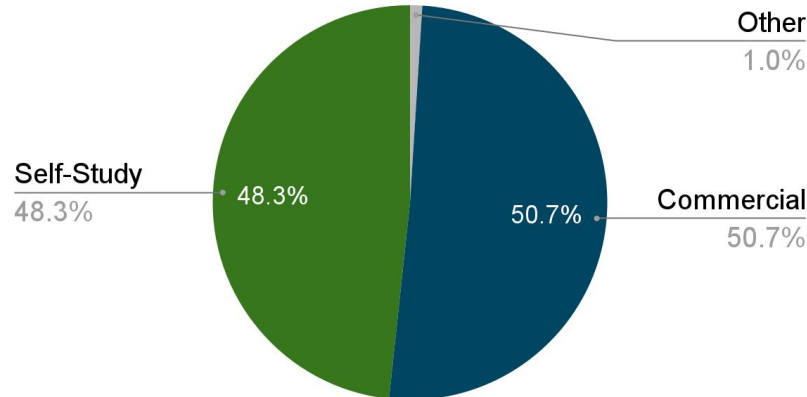
Vulkan Developer Ecosystem Survey

Helps drives LunarG Ecosystem Priorities!

What we heard -

- Shader tool chain - needs dev & maintenance
 - 60%+ glsl -> SPIR-V
 - 20% use DXC
- Validation Layers
 - Increase coverage
 - Readability / interpretation of error messages
 - Improve performance
- MoltenVK
 - Move forward faster
- Difficult Tasks
 - Identifying driver defects
 - Debugging layer issues
 - Debugging install & configuration issues

275 Respondents



2025 Ecosystem Survey Coming in February!

 Windows

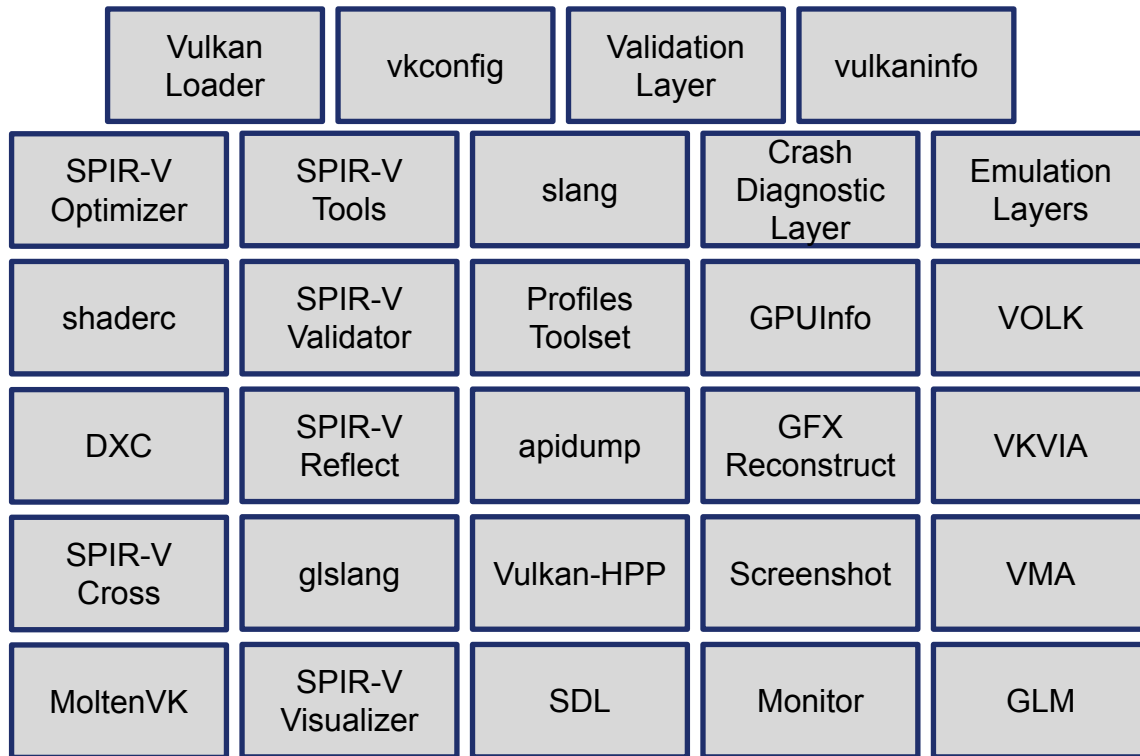
 Windows on **arm**



The Vulkan SDK

Benefits

- Pre-built
- Curated
- Integrated
- System Installation
- vkconfig ready for use
- License Registry



Delivered by LunarG in close coordination with the Khronos Vulkan working group

SDK Enhancements in 2024

2024

- Addition of Slang compiler (beta)
- Crash Diagnostic Layer
- Windows 11 on ARM Vulkan SDK
- Synchronization validation for Timeline Semaphore
- iOS support added to macOS Vulkan SDK
- Profiles enhancements
- VP_KHR_roadmap_2024
- VK_EXT_layer_settings API

SDK Enhancements in 2024

2024

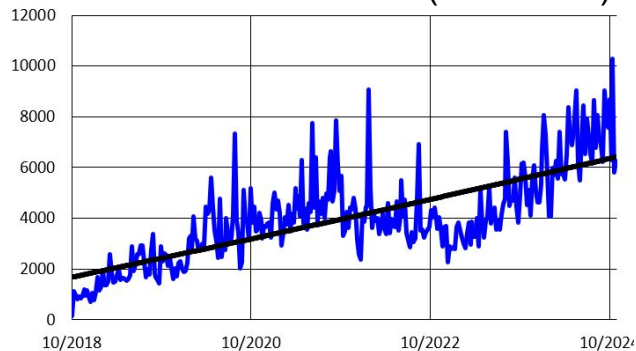
- Addition of Slang compiler (beta)
- Crash Diagnostic Layer
- Windows 11 on ARM Vulkan SDK
- Synchronization validation for Timeline Semaphore
- iOS support added to macOS Vulkan SDK
- Profiles enhancements
- VP_KHR_roadmap_2024
- VK_EXT_layer_settings API

Coming in January

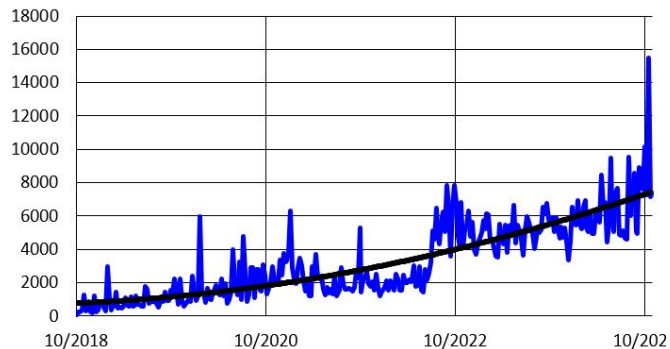
- Vulkan 1.4 support
- vkconfig3
- Automatic update of the Vulkan Runtime (Loader) with the Windows Vulkan SDK installation

SDK Downloads

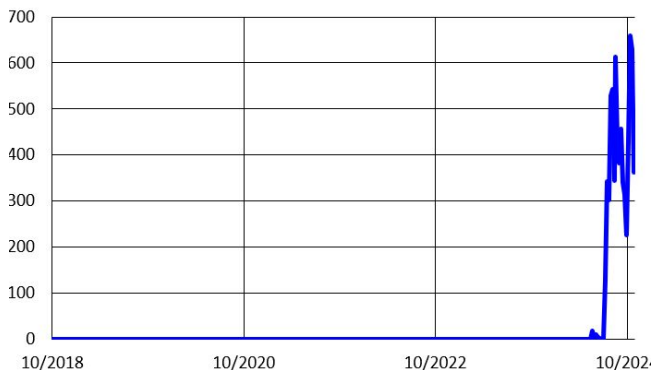
Linux SDK (~6000/wk)



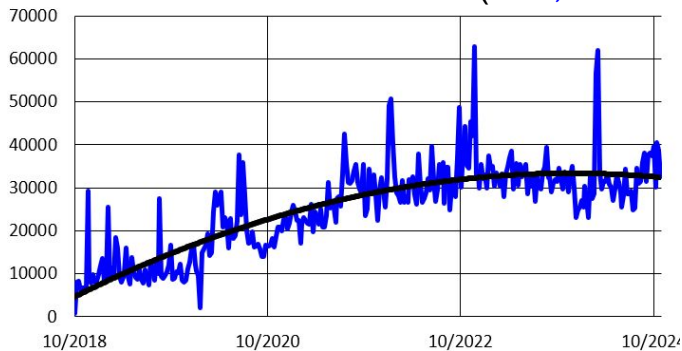
Mac SDK (~6000/wk)



Windows 11 on ARM SDK



Windows x86 SDK (~31,000/wk)



Trends -

- Win x86 - Plateau
- Linux - Slowing
- Mac - Growing
- Win ARM - Too early

★ Polynomial trendlines due to large fluctuations

Updates for some of the Vulkan SDK components

GFXReconstruct

Key Results

- Can now get draw resources for showing intermediate results
- Can now share initialization data among multiple captures in one session to reduce file sizes
- Experimental OpenXR branch

Needs Work

- Capture overhead improvement
- Android functionality and stability
- Ray-tracing support

Validation Layers

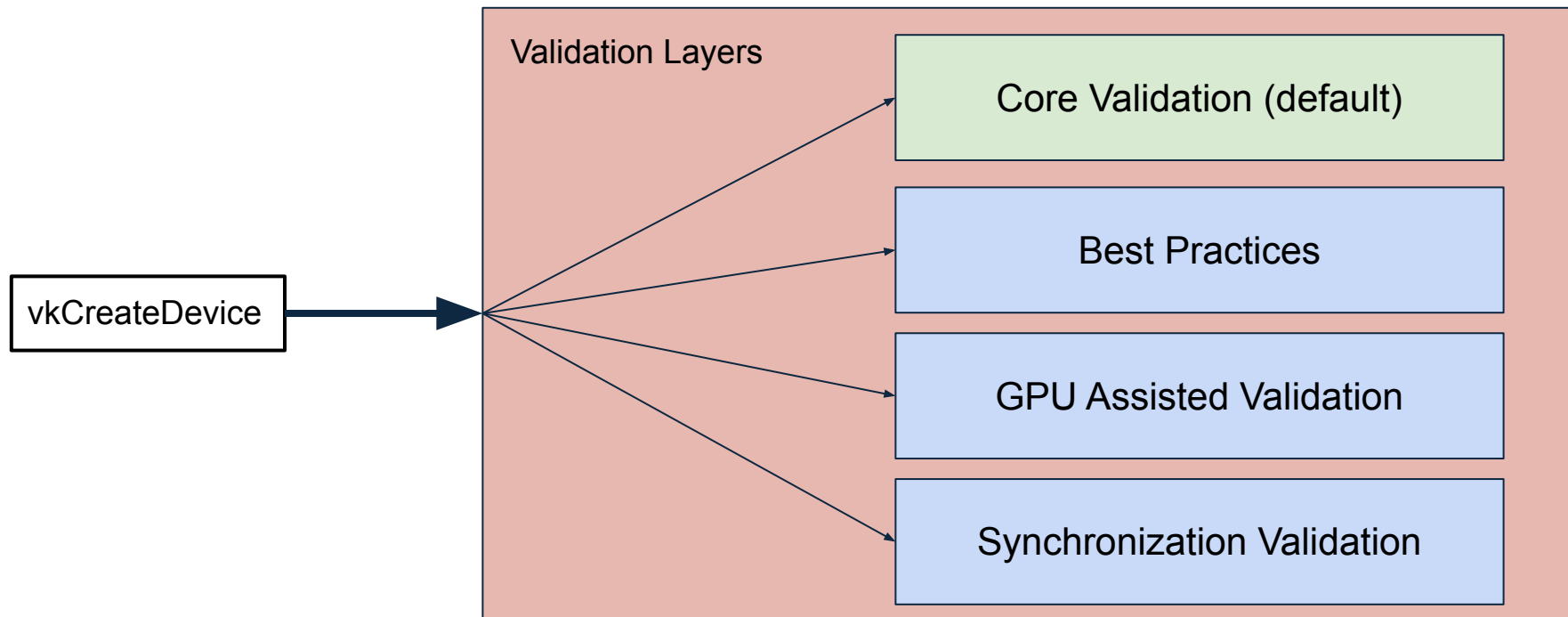
Key Results

- Improved error messages
- More extensions added
- Many issues closed
- **Zero crashing focus**
- **Zero False Positives focused**

Needs Work

- Many Open Issues
- Always new extensions
- Continue performance tuning
- Best practices
 - Lack dedicated engineer
- Error reporting
 - Consistency, completeness, format
 - **Please raise an issue if you find a confusing error message!**

Validation Layers



GPU-AV (GPU Assisted Validation)

Key Results

- Dedicated engineer!
- Performance improvements
- More accurate results
- New functionality added
- Better error messages

Needs Work

- Still can be painfully slow
- Majority of missing validation checks require GPU-AV
- Errors can still be hard to know where they came from

Synchronization Validation

Key Results

- Timeline Semaphore support (added in 1.3.296)
- Many Github issues closed!
- Improved error reporting in the January 1.4 SDK

Needs Work

- Performance tuning
- Continue improving error reporting
- Support for memory aliasing
- Better testing
- **GPU-AV integration to track accesses inside shaders**



[Generated from ChatGPT]

Vulkan Configurator 3 (vkconfig3)

- **Vulkan Configurator 2** - in maintenance mode since May 2024 SDK
- **Vulkan Configurator 3** - Targeting January 2025 SDK
 - Implementing Vulkan Loader settings file:
 - Vulkan developer control of all layers using the UI
 - Vulkan Loader logging using the UI
 - Per-executable layer configurations
 - Easier to select a specific layer version
 - Redesigning UI using a tab per use case
 - Improve application launcher with environment variables and multiple options
 - Add a diagnostic tab (checking Vulkan installation)
 - Adding a layer setting type to run and control command lines

Vulkan Configurator 3 - UI Example 1

Vulkan Configurator 3.0.0-20240906 <ACTIVE>

Diagnostic Applications Layers Configurations Preferences Help

Per-Application: `${VULKAN_SDK}\Bin\vkcube.exe`

Layers Mode: Layers Controlled by Vulkan Configurator

API dump
 Frame Capture
 Portability
 Validation

Loader Messages: Errors Warnings Informations Debug Layers Drivers

Layers Views: All Available Layers

Execute Closer to the Vulkan Application

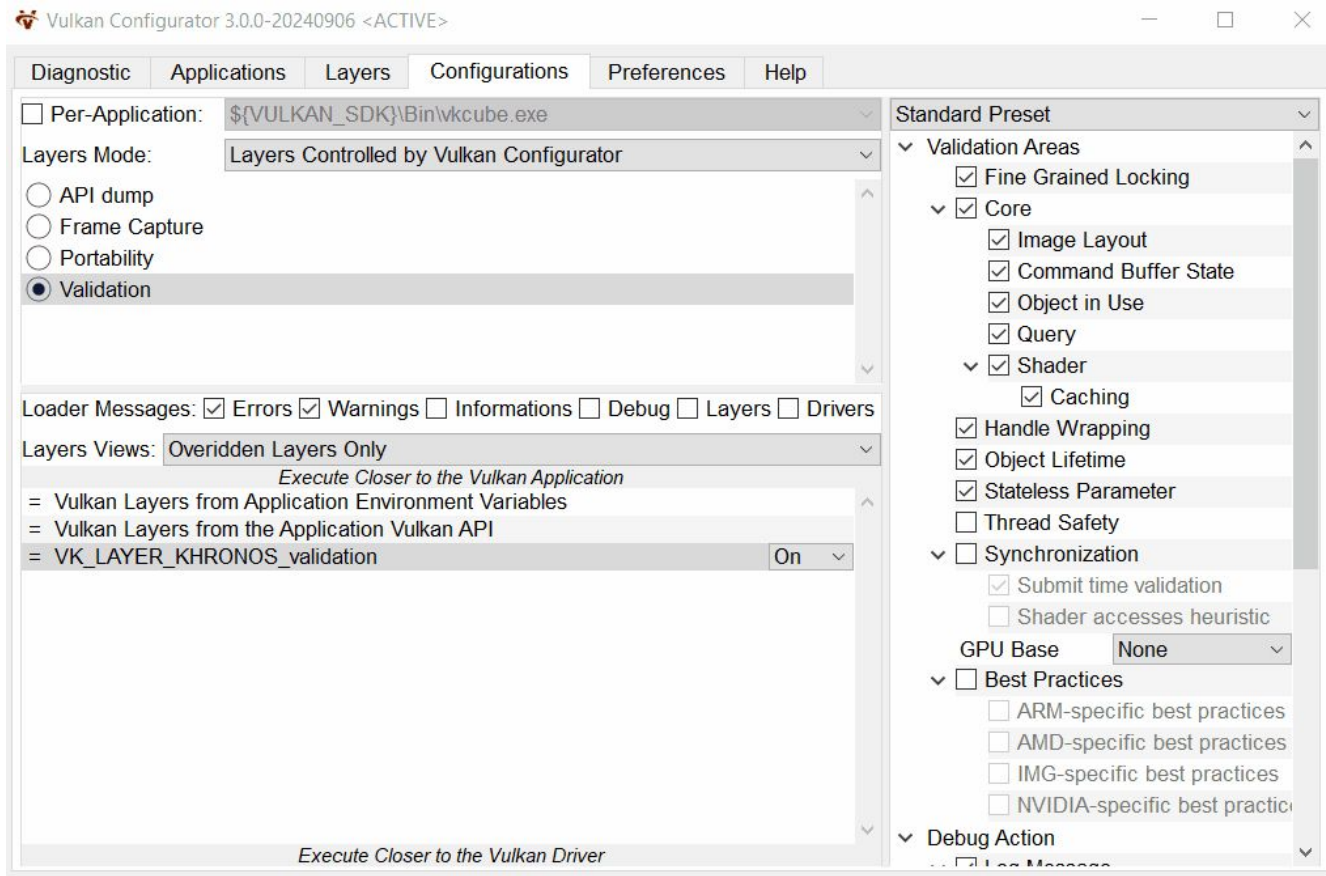
- = Vulkan Layers from Application Environment Variables
- = Vulkan Layers from the Application Vulkan API
- = VK_LAYER_NV_optimus Latest Auto
- = **VK_LAYER_KHRONOS_validation Latest On**
- = VK_LAYER_KHRONOS_profiles Latest Auto
- = VK_LAYER_KHRONOS_shader_object Latest Auto
- = VK_LAYER_KHRONOS_synchronization2 Latest Auto
- = VK_LAYER_LUNARG_api_dump Latest Auto
- = VK_LAYER_LUNARG_crash_diagnostics (ALPHA) Latest Auto
- = VK_LAYER_LUNARG_gfxreconstruct Latest Auto
- = VK_LAYER_LUNARG_monitor Latest Auto
- = VK_LAYER_LUNARG_screenshot Latest Auto

Execute Closer to the Vulkan Driver

User-Defined Settings

- Validation Areas
 - Fine Grained Locking
 - Core
 - Image Layout
 - Command Buffer State
 - Object in Use
 - Query
 - Shader
 - Caching
 - Handle Wrapping
 - Object Lifetime
 - Stateless Parameter
 - Thread Safety
- Synchronization
 - Submit time validation
 - Shader accesses heuristic
- GPU Base: None
- Best Practices
 - ARM-specific best practices
 - AMD-specific best practices
 - IMG-specific best practices
 - NVIDIA-specific best practices
- Debug Action

Vulkan Configurator 3 - UI Example 2



Crash Diagnostic Layer

- Track down and identify the cause of GPU hangs and crashes
 - aka `VK_ERROR_DEVICE_LOST`
- Instruments command buffers with completion checkpoints
- Generates a dump file
- Strong user demand
 - Debugging Device Lost errors very difficult!
- Still in early development
 - **We would love to get your feedback!**

For more information

https://www.youtube.com/watch?v=h5Ty-o8_pWE



Introduction to the Crash Diagnostic Layer

Jeremy Gebben
Senior Graphics Software Engineer
LunarG, Inc



Vulkan Profiles - Use Cases

- **Roadmap profiles:** express guidance on the future direction of Vulkan devices.
 - Eg: Khronos Roadmap 2024
- **Platform profiles:** express the Vulkan support available on a platform.
 - Eg: Android Baseline 2021
- **Engine profiles:** express some rendering code paths requirements of an engine.
 - Eg: VP_UE_Vulkan_SM6_RT in Unreal Engine.
- **Device profiles:** express the Vulkan support of a single Vulkan driver for a Vulkan device.
 - Eg: [GPUinfo.org reports](https://gpuinfo.org/reports)
- **Architecture profiles:** express the Vulkan support of a class of GPUs.
 - Eg: D3D12 Feature Level 12.1

Vulkan Profiles - Tool Set for Developers

- For more information about
 - How to create a Vulkan Profile
 - Vulkan Profiles Layer
 - Vulkan Profiles API Library



"Better Vulkan Application Deployment thanks to Vulkan Profiles"

- Can now use GLSL, HLSL, and SPIR-V on Compiler Explorer (godbolt)
- <https://www.lunarg.com/lunarg-3d-graphics-engineer-adds-glsl-spir-v-as-inputs-to-compiler-explorer/>

The image shows the Compiler Explorer interface with two editor panes. The left pane shows the GLSL source code, and the right pane shows the corresponding SPIR-V IR.

```
1 // This will be consumed as a .glsl file and needs the sta
2 // -S comp --target-env vulkan1.1
3 #version 450
4 layout(set = 0, binding = 0, rgba8) readonly uniform image
5 layout(set = 0, binding = 1, std430) buffer SSB0 {
6     ivec2 coords;
7     vec4 data;
8 };
9
10 void main() {
11     data = imageLoad(myImage, coords);
12     data = data * 3.0f;
13 }
14
```

```
27     %int_0 = OpConstant %int 0
28     %_ptr_StorageBuffer_v2int = OpTypePointer StorageBuffer %
29     %_ptr_StorageBuffer_v4float = OpTypePointer StorageBuffer
30     %float_3 = OpConstant %float 3
31     %main = OpFunction %void None %4
32     %6 = OpLabel
33     %18 = OpLoad %15 %myImage
34     %21 = OpAccessChain %_ptr_StorageBuffer_v2int %_
35     %22 = OpLoad %v2int %21
36     %23 = OpImageRead %v4float %18 %22
37     %25 = OpAccessChain %_ptr_StorageBuffer_v4float %
38     OpStore %25 %23
39     %26 = OpAccessChain %_ptr_StorageBuffer_v4float %
40     %27 = OpLoad %v4float %26
41     %29 = OpVectorTimesScalar %v4float %27 %float_3
42     %30 = OpAccessChain %_ptr_StorageBuffer_v4float %
43     OpStore %30 %29
44     OpReturn
```


BREAKING NEWS!

- Can
- [https://github.com/KhronosGroup/compiler-explorer/](https://github.com/KhronosGroup/compiler-explorer)



```
GLSL source #1 ✎  
A ▾ Save/Load  
1 // This will compile and run  
2 // -S compiler-explorer/...  
3 #version 430  
4 layout(setof 1) inout vec4  
5 layout(setof 1) inout ivec2  
6   ivec2 ivec2;  
7   vec4  vec4;  
8 };  
9  
10 void main()  
11 {  
12     data = data;  
13 }  
14
```

The screenshot shows a GitHub pull request page for the repository 'compiler-explorer'. The title of the pull request is 'Add preliminary Slang support #7151'. It is opened by 'spencer-lunarg' and aims to merge 2 commits into the 'main' branch from the 'spencer-lunarg:spencer-lunarg-slang-1' branch. The page shows 10 conversations, 2 commits, 12 checks, and 11 files changed. A comment from 'spencer-lunarg' is visible, dated 3 days ago. The comment text is as follows:

Part of [#2331](#) and similar to [my GLSL change](#)

[Slang](#) is a GPU focused shading language that has been worked on for years. Last week [The Khronos Group](#) will now be running the project as open governance. The latest [Vulkan SDK](#) has also added a build of `Slangc` (slang compiler).

This change adds support for Slang (the language) as a front end with `Slangc` (the compiler) as the only compiler. Slang can be used for things like GLSL/HLSL, but that is for a future PR.

I am in contacts with people on the Slang development and plan to support things for Slang as well as the other GPU related shading languages

We need your help!

LUNAR)G[®]