

Vulkan SDK Update - 日本語版

Siggraph Asia 2024 - Tokyo

Spencer Fricke

LunarG, Inc



オリジナルの英語スライド

<https://www.lunarg.com/wp-content/uploads/2024/12/SDK-Update-Siggraph-Asia.pdf>

Spencer Frickeは誰ですか？

- LunarGで2年以上働いています
- Vulkan Validation Layersのテックリードです
- 以前は日本に2年間在住
 - 現在はアメリカに永住
 - 今でも毎日日本語を勉強している！
- ワーキンググループ正会員です
- 責任者
 - Vulkan-Guide
 - SPIRV-Visualizer
 - SPIRV-Guide
 - SPIRV-Reflect

Vulkanエコシステムを動かすものは...

Vulkanエコシステムを動かすものは...

あなたの意見！

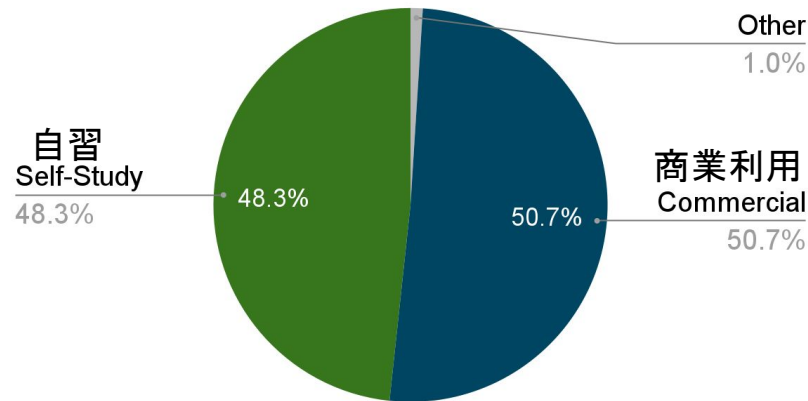
Vulkan開発者エコシステムのアンケート

LunarGの優先順位を決めるのに役立つ！

2024年の結果 -

- シェーダー・ツール・チェーンの改善
 - 60%+ glsl -> SPIR-V
 - 20% DXC を使う
- Validation Layers
 - チェック項目を増やす
 - 読みやすいエラーメッセージ
 - パフォーマンスの向上
- MoltenVK
 - 進捗の迅速化
- 難しい
 - ドライバーの不具合の特定
 - レイヤーの問題のデバッグ
 - インストールと設定の問題のデバッグ

アンケートに回答
275 Respondents



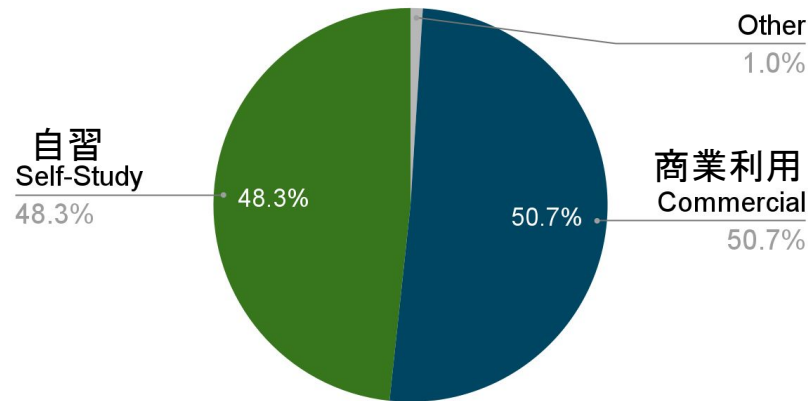
Vulkan開発者エコシステムのアンケート

LunarGの優先順位を決めるのに役立つ！

2024年の結果 -

- シェーダー・ツール・チェーンの改善
 - 60%+ glsl -> SPIR-V
 - 20% DXC を使う
- Validation Layers
 - チェック項目を増やす
 - 読みやすいエラーメッセージ
 - パフォーマンスの向上
- MoltenVK
 - 進捗の迅速化
- 難しい
 - ドライバーの不具合の特定
 - レイヤーの問題のデバッグ
 - インストールと設定の問題のデバッグ

アンケートに回答
275 Respondents

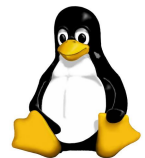


2025 年のエコシステムのアンケートは 2 月に実施されます！

Windows

Windows on arm

Vulkan SDK について



利点

- バイナリはビルド済み
- 厳選バージョン
- システムに統合
- システムのインストール
- vkconfigの使用準備完了
- ライセンスレジストリ

Vulkan Loader	vkconfig	Validation Layer	vulkaninfo	
SPIR-V Optimizer	SPIR-V Tools	slang	Crash Diagnostic Layer	Emulation Layers
shaderc	SPIR-V Validator	Profiles Toolset	GPUInfo	VOLK
DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKZIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

KhronosのVulkanワーキンググループとの緊密な連携のもと、LunarGによって提供される。

2024年のSDKの機能強化

2024年

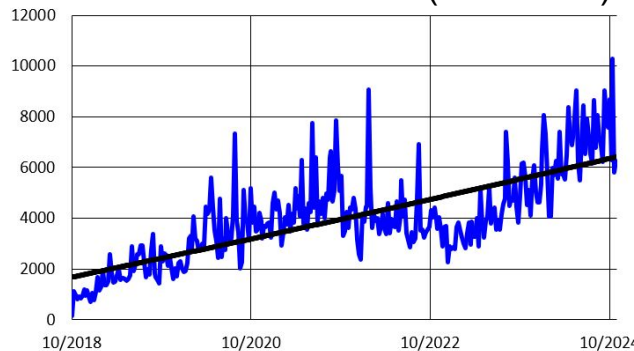
- Slangコンパイラの追加(ベータ版)
- Crash Diagnostic Layer (クラッシュ診断レイヤ)
- Windows 11 on ARM Vulkan SDK
- Synchronization validation for Timeline Semaphore
- macOS Vulkan SDKにiOSサポートを追加
- Profiles enhancements
- VP_KHR_roadmap_2024
- VK_EXT_layer_settings API

1月にリリース予定

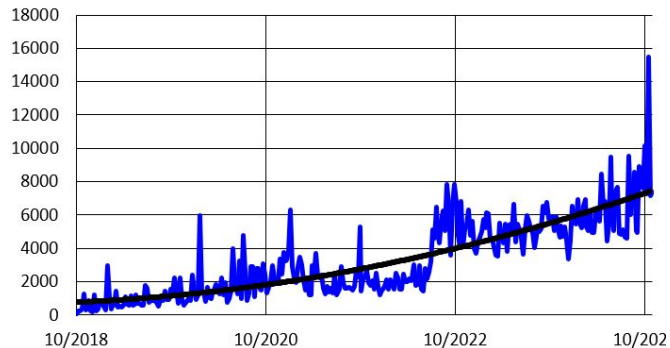
- Vulkan 1.4のサポート
- vkconfig3
- Windows Vulkan SDKのインストールに伴うVulkanランタイム(ローダー)の自動更新

SDK ダウンロード

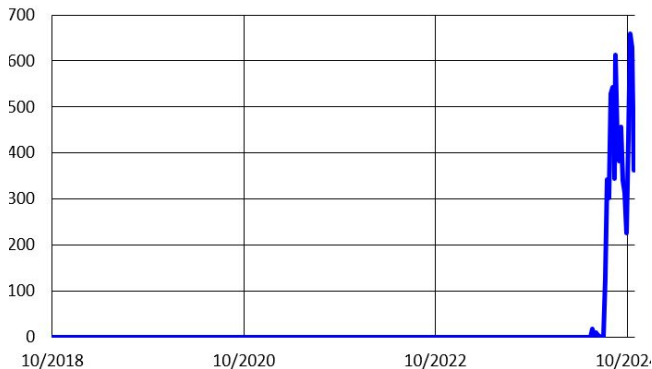
Linux SDK (~6000/wk)



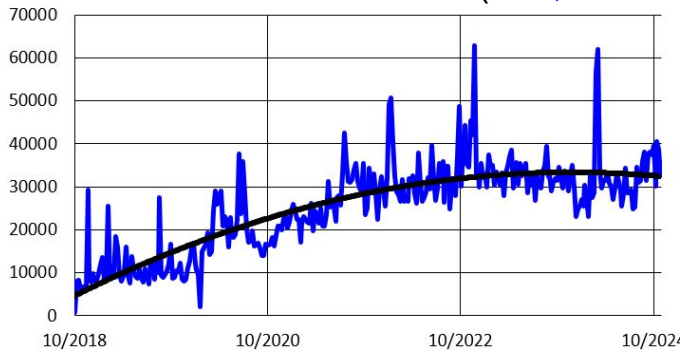
Mac SDK (~6000/wk)



Windows 11 on ARM SDK



Windows x86 SDK (~31,000/wk)



Trends -

- Win x86 - 停滞
- Linux - 減速
- Mac - 成長
- Win ARM - 早すぎる

★ 大きな変動による
多項式トレンドライン

Vulkan SDKコンポーネント の一部の更新

GFXReconstruct (ジー・エフ・エックス・リーコンストラクト)

主な結果

- 中間結果を表示するための描画リソースを取得できるようになった
- 1回のセッション内の複数のキャプチャ間で初期化データを共有し、ファイルサイズを削減できるようになりました。
- OpenXR実験のブランチ

作業が必要

- キャプチャーのパフォーマンス向上
- アンドロイドの機能と安定性
- レイトレーシングのサポート

Validation Layers (検証レイヤ)

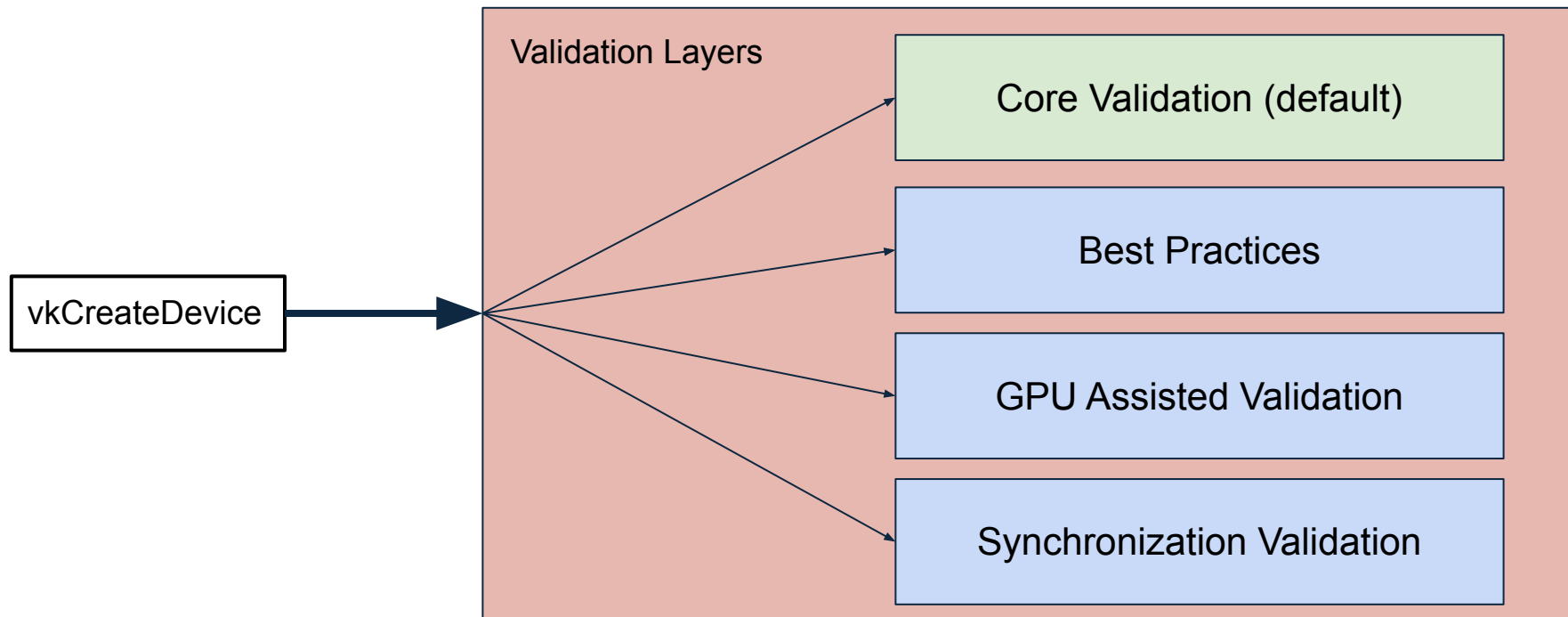
主な結果

- エラーメッセージの改善
- 拡張機能の追加
- 多くの問題を解決
- **クラッシュゼロ**
- **誤検出ゼロ**
 - すべてのエラーを本当のエラーと見なしたい

作業が必要

- 多くの未解決問題
- 常に新しい拡張機能
- パフォーマンス
- Best practices (ベストプラクティス)
 - 専任エンジニアの不足
- エラーメッセージ
 - 一貫性、完全性、形式
 - **紛らわしいエラーメッセージを見つけた場合は、問題を提起してください！**

Validation Layers



GPU-AV (GPU Assisted Validation)

主な結果

- 専属エンジニア!
- パフォーマンスの向上
- より正確な結果
- 新機能の追加
- より良いエラーメッセージ

作業が必要

- まだまだ遅い
- 欠落している検証チェックの大部分はGPU-AVを必要とする
- エラーがどこから来たのか、まだわかりにくいことがある

Synchronization Validation

主な結果

- タイムラインセマフォのサポート
(1.3.296で追加)
- 多くのGithubの問題を解決！
- 1月のSDKのエラーレポートが改善されました

作業が必要

- パフォーマンス
- メモリエイリアシングのサポート
- より良いテスト
- シェーダー内のアクセスを追跡する
GPU-AV統合

これは冗談です。
英語の「sync」と「sink」は
同じ発音です。



[ChatGPTから生成]

Vulkan Configurator 3 (vkconfig3)

- **Vulkan Configurator 2** - 2024年5月からメンテナンスモードになっています SDK
- **Vulkan Configurator 3** - 目標は2025年1月にSDKを公開すること
 - Vulkan Loader設定ファイルの実装:
 - UIを使用したVulkan開発者による全レイヤーの制御
 - UIを使用したVulkan Loaderのロギング
 - 実行可能なレイヤーごとの設定
 - 特定のレイヤー バージョンを選択しやすく
 - ユースケースごとのタブを使用したUIの再設計
 - 環境変数と複数のオプションを使用したアプリケーションランチャーの改善
 - 診断タブの追加 (Vulkanインストールのチェック)
 - コマンドラインを実行および制御するためのレイヤー設定タイプの追加

Vulkan Configurator 3 - UIの例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

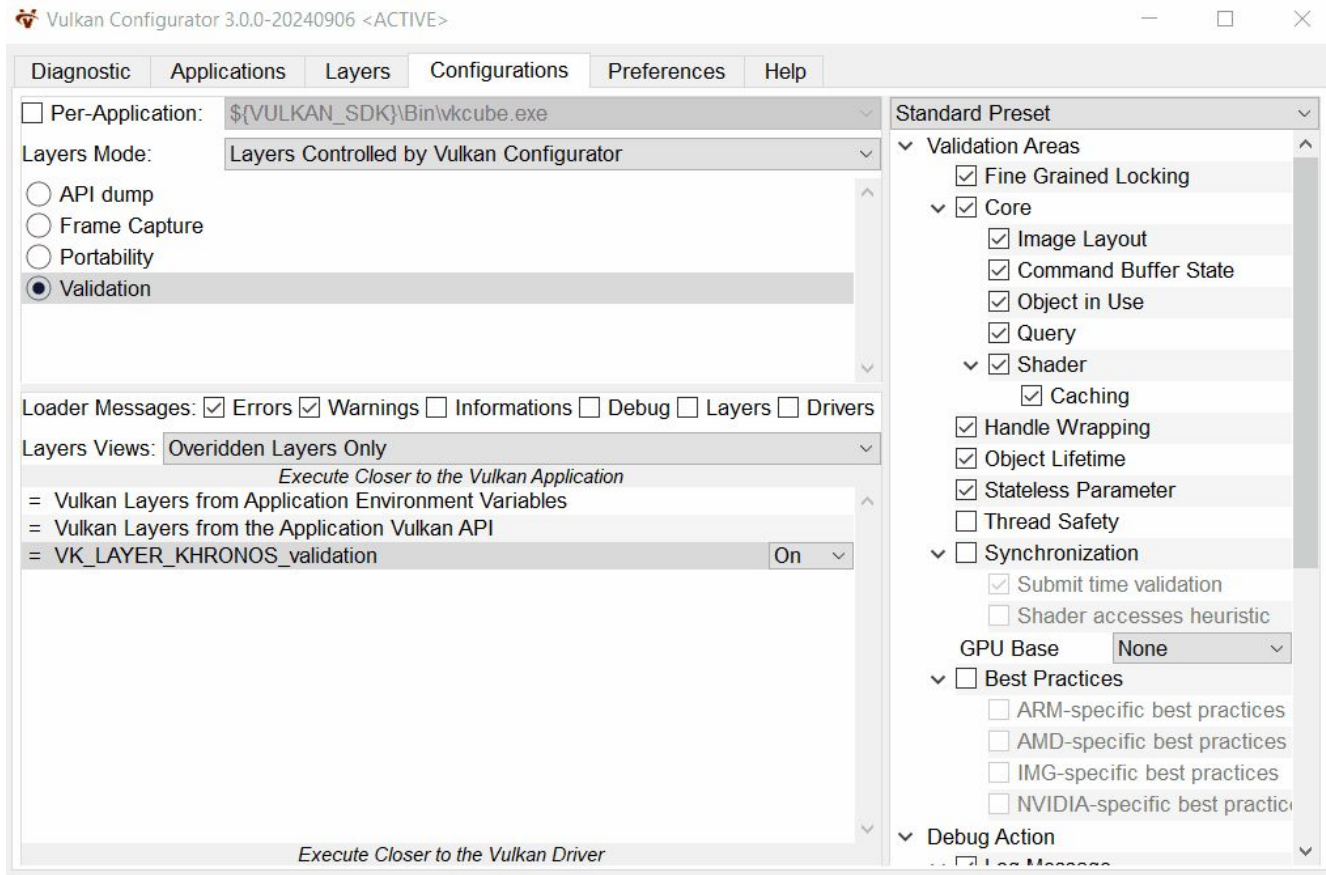
- = Vulkan Layers from Application Environment Variables
- = Vulkan Layers from the Application Vulkan API
- = VK_LAYER_NV_optimus Latest Auto
- = **VK_LAYER_KHRONOS_validation Latest On**
- = VK_LAYER_KHRONOS_profiles Latest Auto
- = VK_LAYER_KHRONOS_shader_object Latest Auto
- = VK_LAYER_KHRONOS_synchronization2 Latest Auto
- = VK_LAYER_LUNARG_api_dump Latest Auto
- = VK_LAYER_LUNARG_crash_diagnostic (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の例2



Crash Diagnostic Layer (クラッシュ診断レイヤー)

- GPUのハングやクラッシュの原因を突き止め、特定する
 - 別名 VK_ERROR_DEVICE_LOST
- コマンドバッファ内のコマンドを追跡して情報を取得する
- ダンプファイルの生成
- ユーザーからの強い要望
 - デバイス・ロスト・エラーのデバッグが非常に困難！
- まだ開発初期
 - フィードバックをお待ちしています！

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 - ユースケース

- **Roadmap profiles:** Vulkanデバイスの将来の方向性に関するガイダンスを示す。
 - 例: Khronos Roadmap 2024
- **Platform profiles:** プラットフォーム上で利用可能なVulkanサポートを表します。
 - 例: Android Baseline 2021
- **Engine profiles:** エンジンのいくつかのレンダリングコードパスの要件を表します。
 - 例: Unreal EngineのVP_UE_Vulkan_SM6_RT
- **Device profiles:** Vulkanデバイスのための単一のVulkanドライバのVulkanサポートを表現します。
 - 例: [GPUinfo.org](https://gpuinfo.org/) のレポート
- **Architecture profiles:** GPUのクラスのVulkanサポートを表現します。
 - 例: D3D12 Feature Level 12.1

Vulkan Profiles - 開発者向けツール

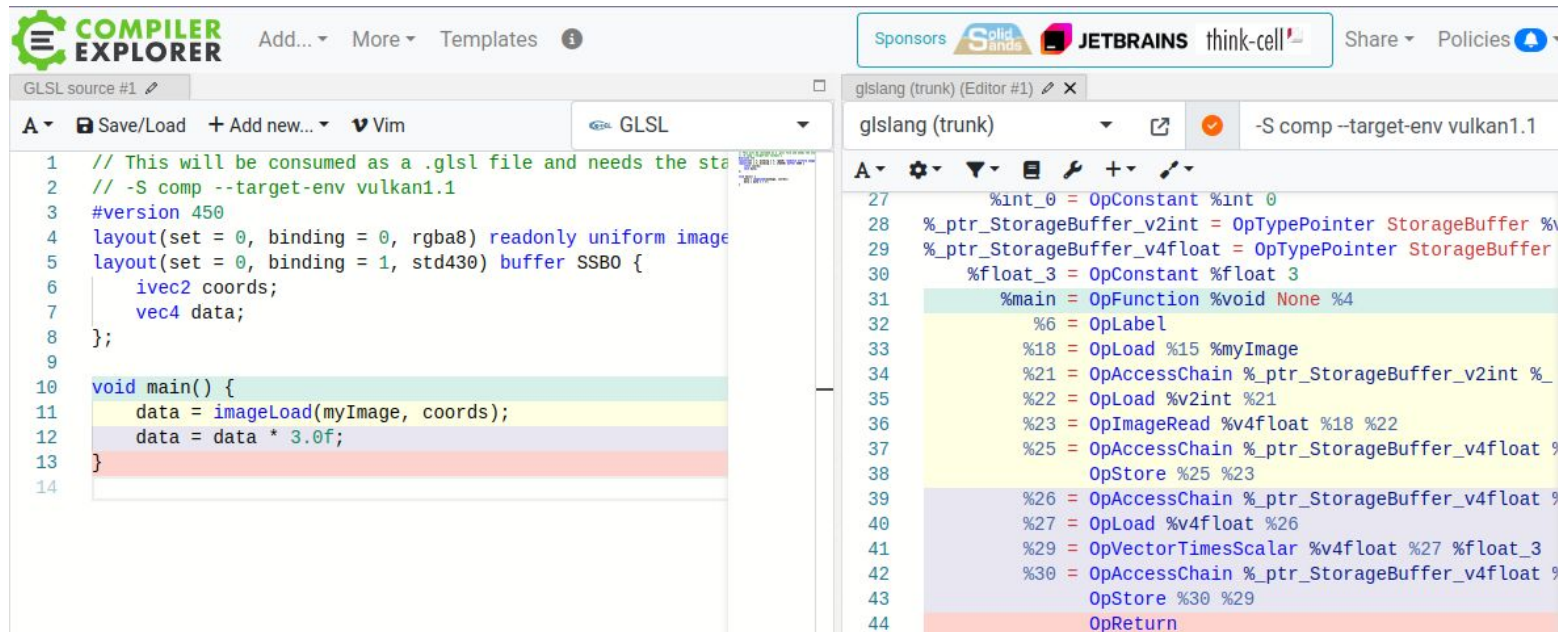
- 詳細情報
 - Vulkan Profileの作成方法
 - Vulkan Profiles Layer
 - Vulkan Profiles API Library



"Better Vulkan Application Deployment thanks to Vulkan Profiles"

(VulkanプロファイルのおかげでVulkanアプリケーションの展開が向上)

- コンパイラー・エクスプローラー(godbolt)でGLSL、HLSL、SPIR-Vが使用可能に
- <https://www.lunarg.com/lunarg-3d-graphics-engineer-adds-glsl-spir-v-as-inputs-to-compiler-explorer/>



The image shows a screenshot of the Compiler Explorer interface. The left pane displays the GLSL source code for a shader, and the right pane displays the corresponding SPIR-V intermediate representation.

```
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
```


速報 !

- コン
- <https://github.com/KhronosGroup/glslang>



GLSL source #1

Save/Load

```
1 // This will compile and run
2 // -S compiler
3 #version 430
4 layout(setof_layouts)
5 layout(setof_layouts)
6   ivec2
7   vec4 d
8 };
9
10 void main()
11   data =
12   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 submitted by 'spencer-lunarg' and aims to merge 2 commits into the 'main' branch. The pull request has 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

あなたの助けが必要です！

(Vulkanの使用経験についてご意見をお聞かせください )

LUNAR)G[®]