NVIDIA Vulkan Update

- NVIDIA driver stack updates
- Libraries
- Tools
Vulkan 1.1!

- Released March 7th!
  - NVIDIA Vulkan 1.1 developer drivers out the same day

- Big ticket items:
  - Subgroups (SM6.0+)
  - Explicit Multi-GPU: support for AFR/SFR, VR applications
  - Multi-view
  - Cross-API / Cross-process synchronization primitives
  - Various “quality of life” improvements and other minor features
Subgroups

- Efficient cross-thread communication primitives
  - Exchange data between invocations of a warp/subgroup
  - Lower latency than shared memory
  - Can be used in graphics shaders
- Vulkan 1.1 supports SM6+ functionality and more

![Diagram showing shuffle, shuffle up, shuffle down, and xor operations involving indexes and any-to-any data exchange.](Image)
Meanwhile, at NVIDIA...

- **EXT_sampler_filter_minmax:**
  - Useful for voxelization algorithms, sparse textures.

- **NV_fragment_coverage_to_color:**
  - Output sample coverage information.

- **EXT_conservative_raster**
  - Cover all the pixels! Faster voxelization, raytraced shadow maps.

- **EXT_depth_range_unrestricted**
  - Depth can go to infinity! (But not beyond.)
NVIDIA keeps busy...

- **EXT_post_depth_coverage**
  - Control coverage from fragment stage.

- **EXT_shader_viewport_index_layer** (subset of NV_viewport_array2)
  - Pick render target layer in vertex shader

- **EXT_sample_locations**
  - Programmable sample locations.

- **NV_fill_rectangle**
  - Improved UI rendering performance, reduces fragment overhead for full-screen passes

- ...
Implementation Limits

- Developers sometimes run into our implementation-defined limits
  - Desire to have lots of descriptors bound comes up frequently
- NVIDIA has so far exposed our actual hardware limits
- We’re now relaxing them
  - Implementation will handle spilling transparently where needed
  - Note that performance may degrade: existing thresholds were designed to avoid spills

maxPerStageDescriptorSamplers 4000 -> 1048576
maxPerStageDescriptorUniformBuffers 12 -> 15
maxPerStageDescriptorStorageBuffers 4096 -> 1048576
maxPerStageDescriptorSampledImages 16384 -> 1048576
maxPerStageDescriptorStorageImages 16384 -> 1048576
maxPerStageDescriptorInputAttachments 8 -> 1048576
maxPerStageResources 53268 -> 4294967295
maxDescriptorSetSamplers 4000 -> 1048576
maxDescriptorSetUniformBuffers 72 -> 90
maxDescriptorSetUniformBuffersDynamic 72 -> 15
maxDescriptorSetStorageBuffers 4096 -> 1048576
maxDescriptorSetSampledImages 98304 -> 1048576
maxDescriptorSetStorageImages 98304 -> 1048576
maxDescriptorSetStorageImages 8 -> 1048576
Shader Compiler Improvements

Original compiler path

- Original bring-up path for driver
- Leveraged NVIDIA’s OpenGL shader compiler (decade+ of refinements)
Shader Compiler Improvements

Original compiler path

- Original bring-up path for driver
- Leveraged NVIDIA’s OpenGL shader compiler (decade+ of refinements)
- However...
  - SPIR-V is not GLSL, translation confused the optimizer
  - Many pathological edge cases for shader performance
  - Slow compile times, high memory usage
Shader Compiler Improvements

Brand new compiler stack!

- Very simple translation between SPIR-V and NVVM (LLVM-based representation)
- Leverages modern compiler improvements in LLVM
- NVVM compiler stack shared across DX12 and Vulkan
Shader Compiler Improvements

What to expect?

- Faster shader compilation --- 3x speedup on average
- Roughly ~50% reduction in memory footprint
- Stable runtime shader performance, less pathological cases
- Expected to ship with R396 drivers (~1 month away)

- VK_NV_glsl_shader being deprecated
  - New compiler does not speak GLSL
  - Extension will be disabled soon after compiler transition
Best Practices

- Not many anti-patterns in shipping applications
  - Vulkan was designed to avoid such things --- seems to be working so far!

- Biggest concern: **use dedicated allocations** for large resources
  - Improves stability when under memory pressure
  - Can be faster in certain cases (and will never be slower)
  - Either flavor (KHR_dedicated_allocation/NV_dedicated_allocation) will work
  - Now core in Vulkan 1.1
Display driver stopped responding and has recovered
Display driver NVIDIA Windows Kernel Mode Driver, Version 497.66.04 stopped responding and has successfully recovered.
How do I debug this?
**Debugging GPU crashes**

i. Crash detected based on error code from API (CPU)

ii. Crash happened sometime in the last N frames of GPU commands...

iii. CPU call stack is likely a red-herring

**Not useful for debugging!**
NVIDIA Aftermath
Post-mortem GPU Crash Debugging

- Insert lightweight markers in the command stream
  - Can stash arbitrary app-specific data behind a marker
  - Designed such that performance impact is negligible, yet highly flexible

- After device lost, read back last marker value that the GPU executed
Debug Instrumentation with Aftermath

**Checkpoints:** Narrow in on GPU crash location WRT to command stream

**Example:**

i. Game inserts user-defined markers in the command stream (CPU)

ii. GPU signals each checkpoint once reached

iii. Last marker reached indicates GPU crash location
Aftermath for Vulkan

- Device checkpoints available soon
  - Same functionality as DX12 version

- More features to come later

```c
// VK_NV_device_diagnostic_checkpoints
typedef struct VkCheckpointDataNV {
    VkStructureType sType;
    const void* pNext;
    VkBool32 checkpointTopValid;
    void* pCheckpointTop;
    VkBool32 checkpointBottomValid;
    void* pCheckpointBottom;
} VkCheckpointDataNV;

void vkCmdSetCheckpointNV(
    VkCommandBuffer commandBuffer,
    const void* pCheckpointData);

VkResult vkGetCheckpointDataNV(
    VkQueue queue,
    VkCheckpointDataNV* pCheckpointData);
```
Flow

- Adaptive sparse voxel smoke/fire
- Vulkan support (in addition to DX11/DX12)
- New sparse framework, grid boundary restrictions removed
- Linux support
- Release coming soon
WaveWorks

- Cinematic-quality ocean simulation for interactive applications
- Rebuilt on cross-API abstraction layer
- Vulkan and DX11 now functional
- Linux support
- Lots of new features, release planned for later this year
Devtools Vulkan Update

Kyle Spagnoli
March 23rd 2018

Booth #223 - South Hall
www.nvidia.com/GDC
Introduction

- Nsight Graphics 1.0
- Vulkan specific features
- Vulkan specific road map
Nsight Graphics 1.0

- New standalone profiling, debugging, and analysis tool
- Builds upon technologies in Nsight Visual Studio Edition
- Vulkan, D3D11/12, OpenGL
- Available now!


https://developer.nvidia.com/nsight-graphics
Nsight Graphics 1.0

- Independent from Visual Studio
- Streamlined launch experience
- Targeted activities
  - Frame Debugging
  - Profiling
  - C++ Capture Export
  - Tracing
Frame Debugger
Frame Debugger - Overview

- Pause an application for live “online” analysis
- Multiple sub-windows offering different tools
- Scrub through important events
- Examine mid-frame state

NOTE: All screenshots from WaveWorks for Vulkan
Frame Debugger - Getting Started

1. Connect to process
2. Target Platform
3. Activity
4. Launch
5. Capture for Live Analysis
Frame Debugger - Event View

- Captured event stream
- Organize by command buffer, thread, queue
- Powerful filter, sort, and search systems
- Hierarchical range markers via VK_EXT_debug_marker
- Object names via VK_EXT_debug_marker
Frame Debugger - Scrubber

- Timeline view of your scene
  - Event, GPU, or CPU time scale
- Multi-queue and multi-thread aware
- Range markers via VK_EXT_debug_marker and heuristics
- Synchronization indicators
Frame Debugger - Current Target

- Visualize incremental per-draw state of your render targets
- Color, depth, stencil
- Optional overlays
- Pixel history
  - More later...
Frame Debugger - API Inspector

- Per-pipeline stage information
- See bound descriptor sets and their associated data
- Images with thumbnails
- Uniform buffers with data at-a-glance
- Reflection information if SPIRV has annotations
Frame Debugger - Descriptor Views

- Listing of all descriptors used in scene
- Sort and search by pools and layouts
- At a glance contents of each descriptor
Frame Debugger - Shaders

- Per-pipeline and per-shader information
- View shader SPIRV source
- View as GLSL or HLSL via translation
Frame Debugger - Geometry

- 3D geometry of current draw calls
- Highly configurable
  - Select attributes for position, color, and normal
  - Multiple shading modes
- Reflection information from SPIRV annotations
Frame Debugger - Resources

- At a glance view of all images and buffers
- Objects names via VK_EXT_debug_marker extension
- Powerful search and filters
- Revision and consumption tracking
- Links to backing device memory
Frame Debugger - Device Memory

- At a glance view of all device memory objects
- Listing of bound resources
- Memory of each sub-resource
- Object layout map

NOTE: This example is only one resource per-memory region.
C++ Capture
C++ Capture - Overview

- Export an application frame as a free standing executable built from C++ source
- Previously called “Serialization” in Nsight VSE
- Debug / profile / edit & experiment with source
C++ Capture - Code Output

- Great for rapid “hack and slash” changes
- Human readable code

```cpp
// Create VkImageView uid_40

BEGIN_DATA_SCOPE();

static VkImageViewCreateInfo VkImageViewCreateInfo_temp_14[1] = { VkImageViewCreateInfo{
    /* sType = */ VK_STRUCTURE_TYPE_IMAGE_VIEW_CREATE_INFO,
    /* pNext = */ nullptr,
    /* flags = */ VkImageViewCreateFlags(0),
    /* image = */ VkImage_uid_37,
    /* viewType = */ VK_IMAGE_VIEW_TYPE_2D,
    /* format = */ VK_FORMAT_R8G8B8A8_UNORM,
    /* components = */ VkComponentMapping{
        /* r = */ VK_COMPONENT_SWIZZLE_R,
        /* g = */ VK_COMPONENT_SWIZZLE_G,
        /* b = */ VK_COMPONENT_SWIZZLE_B,
        /* a = */ VK_COMPONENT_SWIZZLE_A,
    },
    /* subresourceRange = */ VkImageSubresourceRange{
        /* aspectMask = */ VkImageAspectFlags(VK_IMAGE_ASPECT_COLOR_BIT),
        /* baseMipLevel = */ 0,
        /* levelCount = */ 1,
        /* baseArrayLayer = */ 0,
        /* layerCount = */ 1
    }},

NV_THROW_IF(vkCreateImageView(VkDevice_uid_16, VkImageViewCreateInfo_temp_14, nullptr, &VkImageView_uid_40) != VK_SUCCESS, "A Vulkan API call was unsuccessful");
```
C++ Capture - Vulkan Capabilities

- Respects the acquire / record / submit / present model
  - Can’t naively loop frames
  - Need to record command buffers based on results of `vkAcquireNextImageKHR`

- Best-effort to support replay on different hardware from capture
  - Dynamically patch up code
  - Some cases are impossible (e.g. missing major feature or extension)
C++ Capture - .nsight-gfxcppcap File

- Additional metafile generated with C++ export
- Screenshot, system information, statistics, etc.
- Ability to build (#1) and launch (#2) from within the GUI
- Easily debug, profile, etc. later
Vulkan Tools Roadmap
Vulkan Tools Roadmap

- **2018 - 1st Half**
  - Vulkan 1.1 support
  - Linux support*
  - Android support
  - Shader statistics
  - Shader editing
  - C++ export improvements

- **2018 - 2nd Half**
  - Hardware profiling
  - GPU trace
  - Pixel history

- **Beyond...**
  - Support future extensions and core updates

---

*Already available in the NVIDIA Linux Graphics Debugger 2.3*
Roadmap - Shader Statistics

- Low level information about each pipeline and shader object
- Estimated cycles
- Register counts
- LMem counts

NOTE: Currently supported for D3D & OpenGL
Roadmap - Shader Editing

- Edit shader source and see changes reflected live in running application
- Quickly toggle between shader sets
- Edit as SPIRV, GLSL, or HLSL

NOTE: Currently supported for D3D & OpenGL
Roadmap - Hardware Profiling

- Low level performance metrics on a range and event basis
- Find performance limiting hardware units
- Identify optimization opportunities

NOTE: Currently supported for D3D & OpenGL
Roadmap - GPU Trace

- Low level GPU utilization
- Identify and optimize asynchronous compute opportunities

NOTE: Currently supported for D3D12 in preview
Road Map - Pixel History

- Trace the life of a pixel in a render target
- Draws, clears, and blits that contribute to final output pixel value
- Detailed information about failed fragments
  - Back face culling, depth, etc.

NOTE: Currently supported for D3D & OpenGL
Thank you!

- Nvidia Nsight Graphics 1.0 now available
- Nvidia Linux Graphics Debugger 2.3 (with Vulkan support) now available
- Live demos @ Booth 233 in South Hall
- We are hiring!

NVIDIA.com > Developers > GameWorks > Tools

https://developer.nvidia.com/nsight-graphics