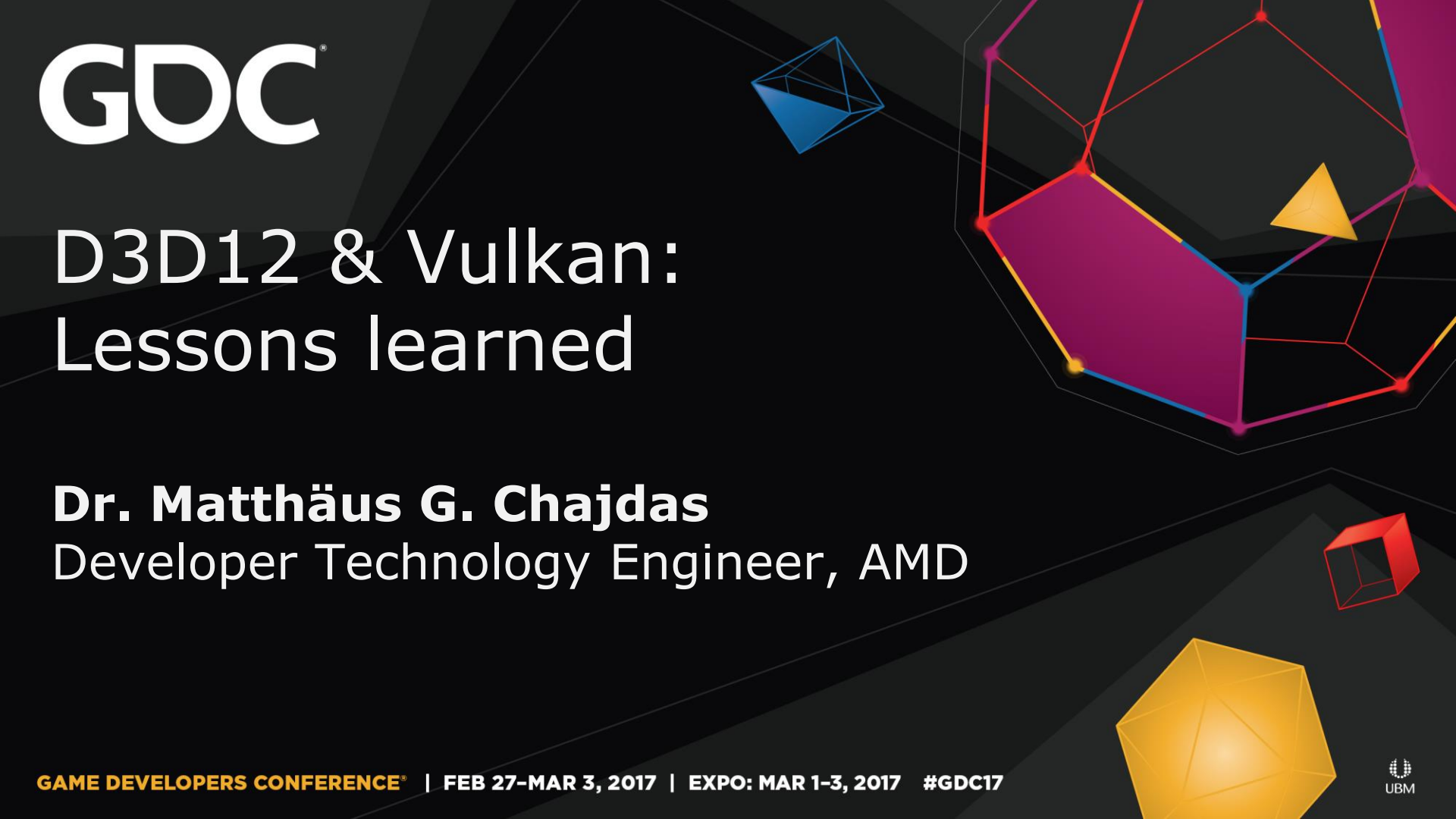


D3D12 & Vulkan: Lessons learned

Dr. Matthäus G. Chajdas
Developer Technology Engineer, AMD

D3D12 – What's new?

- DXIL
- DXGI & UWP updates
- Root Signature 1.1
- Shader cache
- GPU validation
- PIX

D3D12 / DXIL

- DXBC gets replaced by DXIL
- Language based on LLVM IR
- New open-source frontend based on Clang (**dxc**)

D3D12 / DXIL

```
float4 PSMain(PSInput input) : SV_TARGET
{
        return input.color;
}
```

```
define void @PSMain() {
  %1 = call float @dx.op.loadInput.f32(i32 4, i32 1, i32 0, i8 0, i32 undef)
    ; LoadInput(inputSigId,rowIndex,colIndex,gsVertexAxis)
  %2 = call float @dx.op.loadInput.f32(i32 4, i32 1, i32 0, i8 1, i32 undef)
    ; LoadInput(inputSigId,rowIndex,colIndex,gsVertexAxis)
  %3 = call float @dx.op.loadInput.f32(i32 4, i32 1, i32 0, i8 2, i32 undef)
    ; LoadInput(inputSigId,rowIndex,colIndex,gsVertexAxis)
  %4 = call float @dx.op.loadInput.f32(i32 4, i32 1, i32 0, i8 3, i32 undef)
    ; LoadInput(inputSigId,rowIndex,colIndex,gsVertexAxis)
  call void @dx.op.storeOutput.f32(i32 5, i32 0, i32 0, i8 0, float %1)
    ; StoreOutput(outputtSigId,rowIndex,colIndex,value)
  call void @dx.op.storeOutput.f32(i32 5, i32 0, i32 0, i8 1, float %2)
    ; StoreOutput(outputtSigId,rowIndex,colIndex,value)
  call void @dx.op.storeOutput.f32(i32 5, i32 0, i32 0, i8 2, float %3)
    ; StoreOutput(outputtSigId,rowIndex,colIndex,value)
  call void @dx.op.storeOutput.f32(i32 5, i32 0, i32 0, i8 3, float %4)
    ; StoreOutput(outputtSigId,rowIndex,colIndex,value)
  ret void
}
```

D3D12 / Updated DXGI

- First-class support for variable refresh rate displays - *-Sync
- HDR support

D3D12 / UWP

- Lots of the initial UWP limitations have been lifted
- Capabilities now on par with normal Win32

D3D12 / Root Signature 1.1

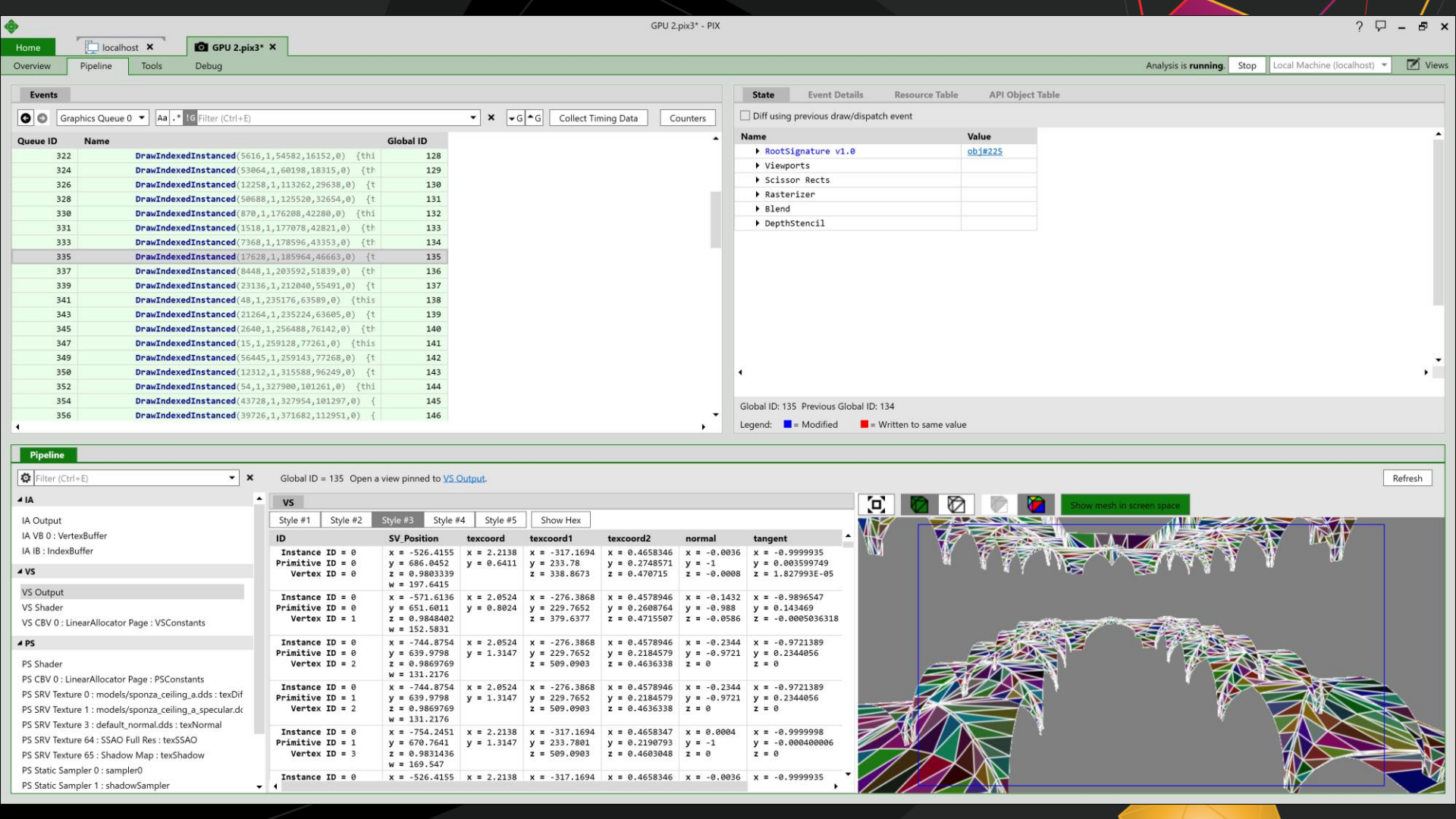
- Allows you to tell the driver that descriptors won’t change
- Can allow some optimizations in the future
- Nice to have, but nothing critical

D3D12 / Shader cache

- The shader cache was not sufficient in D3D12 RTM
- Big improvements in the “Anniversary” edition – now usable
- Note: Drivers may have yet another shader cache!

D3D12 / GPU validation

- Checks descriptors at draw time
- Discovers various hard-to-find bugs (stale descriptors, etc.)
- Rather slow - run over night for regression testing


GPU 2.pix3* - PIX
Home
localhost
GPU 2.pix3*
Overview
Pipeline
Tools
Debug
Analysis is running.
Stop
Local Machine (localhost)
Views
Events
Graphics Queue 0
Filter (Ctrl+E)
Collect Timing Data
Counters
Queue ID
Name
Global ID
322 DrawIndexedInstanced(5616,1,54582,16152,0) {thi 128
324 DrawIndexedInstanced(53064,1,60198,18315,0) {th 129
326 DrawIndexedInstanced(12258,1,113262,29638,0) {t 130
328 DrawIndexedInstanced(50688,1,125520,32654,0) {t 131
330 DrawIndexedInstanced(870,1,176208,42280,0) {thi 132
331 DrawIndexedInstanced(1518,1,177078,42821,0) {th 133
333 DrawIndexedInstanced(7368,1,178596,43353,0) {th 134
335 DrawIndexedInstanced(17628,1,185964,46663,0) {t 135
337 DrawIndexedInstanced(8448,1,203592,51839,0) {th 136
339 DrawIndexedInstanced(23136,1,212040,55491,0) {t 137
341 DrawIndexedInstanced(48,1,235176,63589,0) {this 138
343 DrawIndexedInstanced(21264,1,235224,63605,0) {t 139
345 DrawIndexedInstanced(2640,1,256488,76142,0) {th 140
347 DrawIndexedInstanced(15,1,259128,77261,0) {this 141
349 DrawIndexedInstanced(56445,1,259143,77268,0) {t 142
350 DrawIndexedInstanced(12312,1,315588,96249,0) {t 143
352 DrawIndexedInstanced(54,1,327900,101261,0) {thi 144
354 DrawIndexedInstanced(43728,1,327954,101297,0) { 145
356 DrawIndexedInstanced(39726,1,371682,112951,0) { 146
State
Event Details
Resource Table
API Object Table
Diff using previous draw/dispatch event
Name
Value
RootSignature v1.0
obj#225
Viewports
Scissor Rects
Rasterizer
Blend
DepthStencil
Global ID: 135 Previous Global ID: 134
Legend: = Modified = Written to same value
Pipeline
Filter (Ctrl+E)
Global ID = 135 Open a view pinned to VS Output.
Refresh
IA
IA Output
IA VB 0 : VertexBuffer
IA IB : IndexBuffer
VS
VS Output
VS Shader
VS CBV 0 : LinearAllocator Page : VSConstants
PS
PS Shader
PS CBV 0 : LinearAllocator Page : PSConstants
PS SRV Texture 0 : models/sponza_ceiling_a.dds : texDif
PS SRV Texture 1 : models/sponza_ceiling_a_specular.dc
PS SRV Texture 3 : default_normal.dds : texNormal
PS SRV Texture 64 : SSAO Full Res : texSSAO
PS SRV Texture 65 : Shadow Map : texShadow
PS Static Sampler 0 : sampler0
PS Static Sampler 1 : shadowSampler
VS
Style #1
Style #2
Style #3
Style #4
Style #5
Show Hex
ID
SV_Position
texcoord
texcoord1
texcoord2
normal
tangent
Instance ID = 0 Primitive ID = 0 Vertex ID = 0 x = -526.4155 y = 686.0452 z = 0.9803339 w = 197.6415 x = 2.2138 y = 0.6411 x = -317.1694 y = 233.78 z = 338.8673 x = 0.4658346 y = 0.2748571 z = 0.470715 x = -0.0036 y = -1 z = -0.0008 x = -0.9999935 y = 0.003599749 z = 1.827993E-05
Instance ID = 0 Primitive ID = 0 Vertex ID = 1 x = -571.6136 y = 651.6011 z = 0.9848402 w = 152.5831 x = 2.0524 y = 0.8024 x = -276.3868 y = 229.7652 z = 379.6377 x = 0.4578946 y = 0.2608764 z = 0.4715507 x = -0.1432 y = -0.988 z = -0.0586 x = -0.9896547 y = 0.143469 z = -0.0005036318
Instance ID = 0 Primitive ID = 0 Vertex ID = 2 x = -744.8754 y = 639.9798 z = 0.9869769 w = 131.2176 x = 2.0524 y = 1.3147 x = -276.3868 y = 229.7652 z = 509.0903 x = 0.4578946 y = 0.2184579 z = 0.4636338 x = -0.2344 y = -0.9721 z = 0 x = -0.9721389 y = 0.2344056 z = 0
Instance ID = 0 Primitive ID = 1 Vertex ID = 2 x = -744.8754 y = 639.9798 z = 0.9869769 w = 131.2176 x = 2.0524 y = 1.3147 x = -276.3868 y = 229.7652 z = 509.0903 x = 0.4578946 y = 0.2184579 z = 0.4636338 x = -0.2344 y = -0.9721 z = 0 x = -0.9721389 y = 0.2344056 z = 0
Instance ID = 0 Primitive ID = 1 Vertex ID = 3 x = -754.2451 y = 670.7641 z = 0.9831436 w = 169.547 x = 2.2138 y = 1.3147 x = -317.1694 y = 233.7801 z = 509.0903 x = 0.4658347 y = 0.2190793 z = 0.4603048 x = 0.0004 y = -1 z = 0 x = -0.9999998 y = -0.000400006 z = 0
Instance ID = 0 x = -526.4155 x = 2.2138 x = -317.1694 x = 0.4658346 x = -0.0036 x = -0.9999935
Show mesh in screen space

D3D12 / PIX

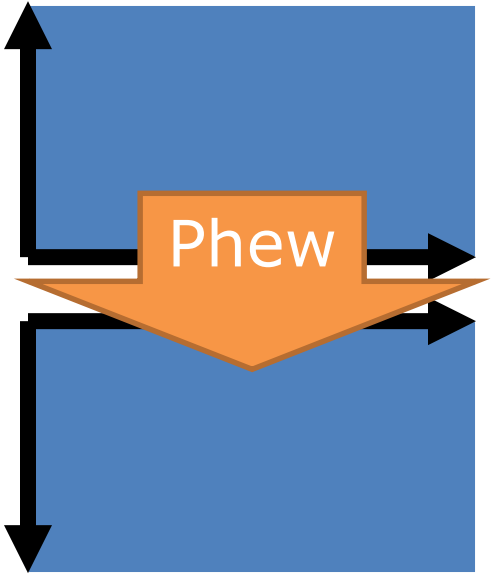
- Alternative to RenderDoc
- More than just a debugger
  - Profiling
  - Easy access to shader ISA

Vulkan – What's new?

- KHR_maintenance1
- EXT_shader_subgroup
- KHR_get_physical_device_properties2
- KHR_shader_draw_parameters
- And many (>**20**) more …

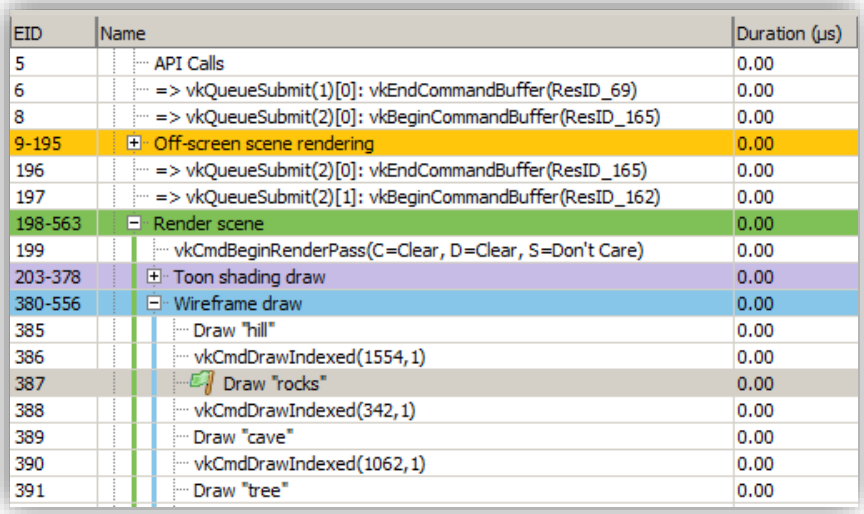
Vulkan / Usability

- `KHR_maintenance1`
- Window origin fix (`AMD_negative_viewport_height`)
- Various other small fixes



Vulkan / Usability

- VK_EXT_debug_marker
  - Markup scene just like in D3D with annotations
  - Supported by tools!



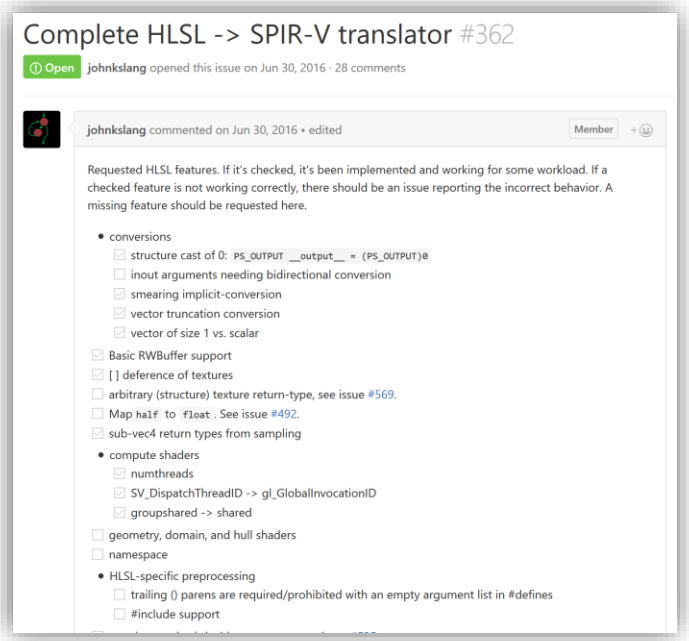
EID	Name	Duration (µs)
5	API Calls	0.00
6	=> vkQueueSubmit(1)[0]: vkEndCommandBuffer(ResID_69)	0.00
8	=> vkQueueSubmit(2)[0]: vkBeginCommandBuffer(ResID_165)	0.00
9-195	⊞ Off-screen scene rendering	0.00
196	=> vkQueueSubmit(2)[0]: vkEndCommandBuffer(ResID_165)	0.00
197	=> vkQueueSubmit(2)[1]: vkBeginCommandBuffer(ResID_162)	0.00
198-563	⊟ Render scene	0.00
199	vkCmdBeginRenderPass(C=Clear, D=Clear, S=Don't Care)	0.00
203-378	⊞ Toon shading draw	0.00
380-556	⊟ Wireframe draw	0.00
385	Draw "hill"	0.00
386	vkCmdDrawIndexed(1554,1)	0.00
387	Draw "rocks"	0.00
388	vkCmdDrawIndexed(342,1)	0.00
389	Draw "cave"	0.00
390	vkCmdDrawIndexed(1062,1)	0.00
391	Draw "tree"	0.00

Vulkan / Porting

- `VK_AMD_draw_indirect_count`
  - Multi-draw-indirect with **count from buffer**
  - Feature-parity with OpenGL
- `KHR_shader_draw_parameters`
  - `gl_drawID`, `gl_BaseVertex`, `gl_BaseInstance`
  - Again, feature parity

Vulkan / Porting

- glslang accepts HLSL now
- Already usable for many real-world shaders!



Complete HLSL -> SPIR-V translator #362

Open johnkslang opened this issue on Jun 30, 2016 · 28 comments

johnkslang commented on Jun 30, 2016 • edited — Member

Requested HLSL features. If it's checked, it's been implemented and working for some workload. If a checked feature is not working correctly, there should be an issue reporting the incorrect behavior. A missing feature should be requested here.

- conversions
  - [x] structure cast of 0: `PS_OUTPUT __output__ = (PS_OUTPUT)0`
  - [ ] inout arguments needing bidirectional conversion
  - [x] smearing implicit-conversion
  - [x] vector truncation conversion
  - [x] vector of size 1 vs. scalar
- [x] Basic RWBuffer support
- [x] [ ] deference of textures
- [ ] arbitrary (structure) texture return-type, see issue #569.
- [ ] Map `half` to `float`. See issue #492.
- [x] sub-vec4 return types from sampling
- compute shaders
  - [x] numthreads
  - [x] SV_DispatchThreadID -> gl_GlobalInvocationID
  - [x] groupshared -> shared
- [ ] geometry, domain, and hull shaders
- [ ] namespace
- HLSL-specific preprocessing
  - [ ] trailing () parens are required/prohibited with an empty argument list in #defines
  - [ ] #include support

Vulkan extensions / Performance

- `VK_AMD_rasterization_order`
  - Relaxed rasterization order
  - A stepping stone towards more declarative rendering

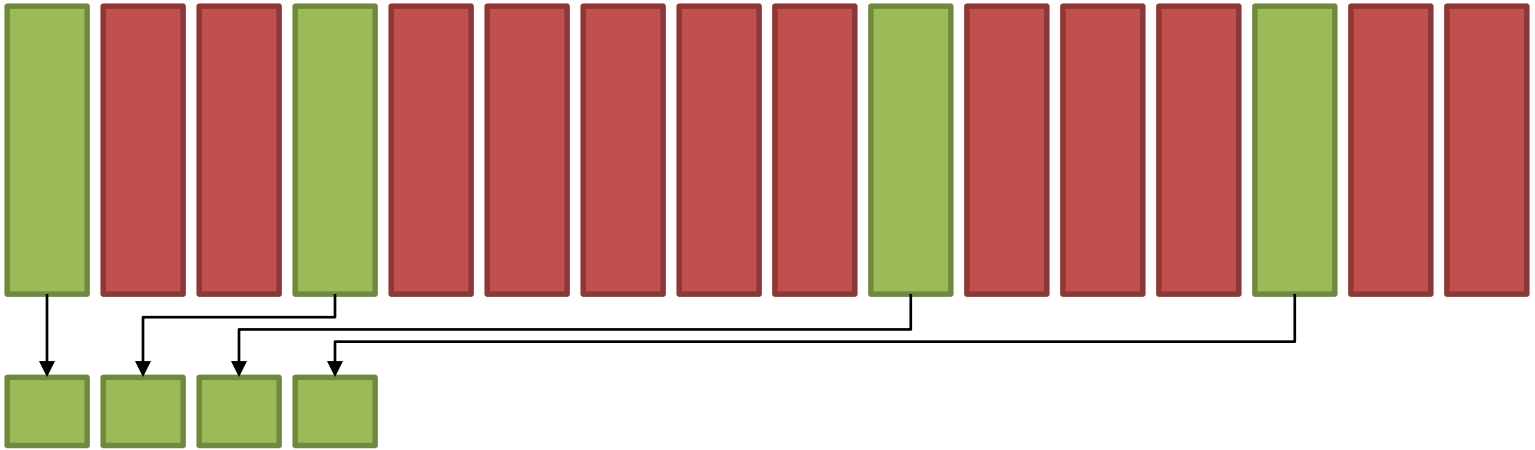
What's new – D3D12 & Vulkan

- Wave-wide
- FP16

Vulkan & D3D12 / Wave-wide

- Wave-wide instructions are now core in both APIs
  - Shader Model 6.0 for HLSL
  - `SPV_KHR_shader_ballot`, `EXT_shader_subgroup_*` for SPIR-V
- Console-like programing everywhere!

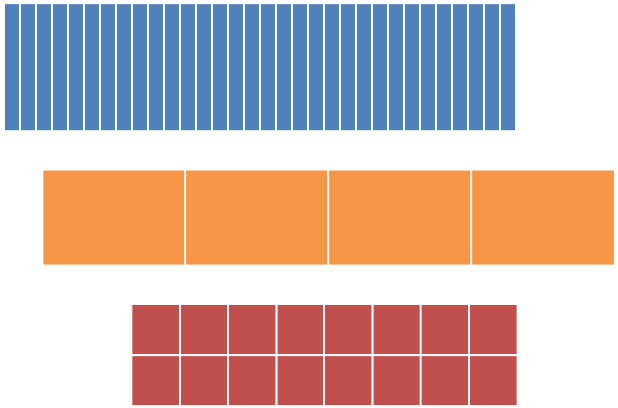
Vulkan & D3D12 / Wave-wide



Compact wave-wide using a wave-wide prefix sum: Now in SM6 and SPIR-V!

Vulkan & D3D12 / Wave-wide

- Use the right atomics at the right level



Wavefront - **Intrinsic**

Threadgroup - **Local memory**

Dispatch - **Global memory**

Vulkan & D3D12 / Wave-wide

- Your data is wave-uniform but your shader compiler doesn't know it
- Express it now in SM6 and SPIR-V!
  - `readFirstLane`
  - `WaveReadFirstLane`

Vulkan & D3D12 / Wave-wide

- Another typical use:
  - Iterate over light sources
  - Tell compiler light index is uniform wave-wide
  - Data goes into SGPR instead of VGPR
  - Profit!

Vulkan & D3D12 / FP16

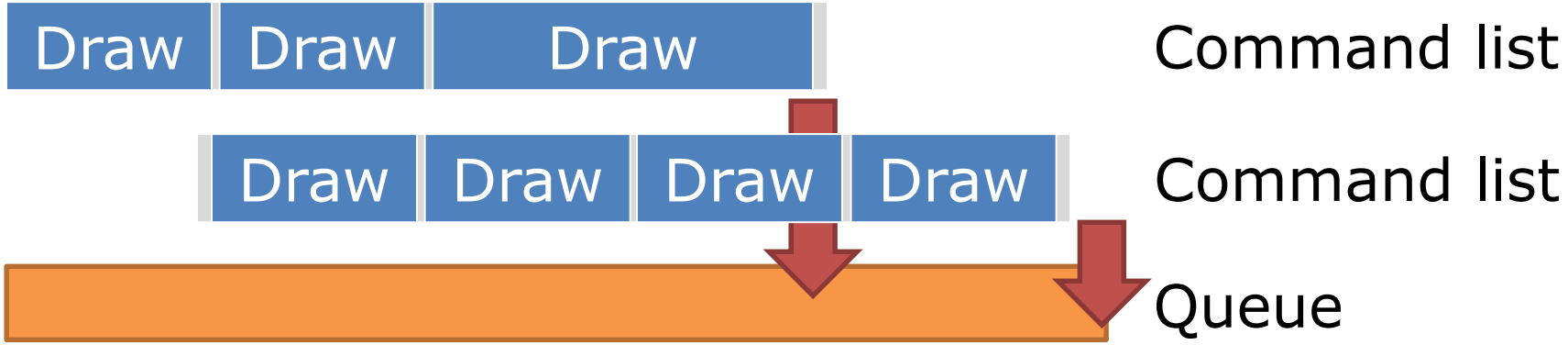
- Same benefits on PC as on console
  - Reduced register count (and LDS usage!)
  - Better performance
- Simplifies porting between console & mobile

Dawn of a new era

- Peak performance requires new concepts!

Command lists

- Separate **recording** from **submission**
- Much higher throughput!

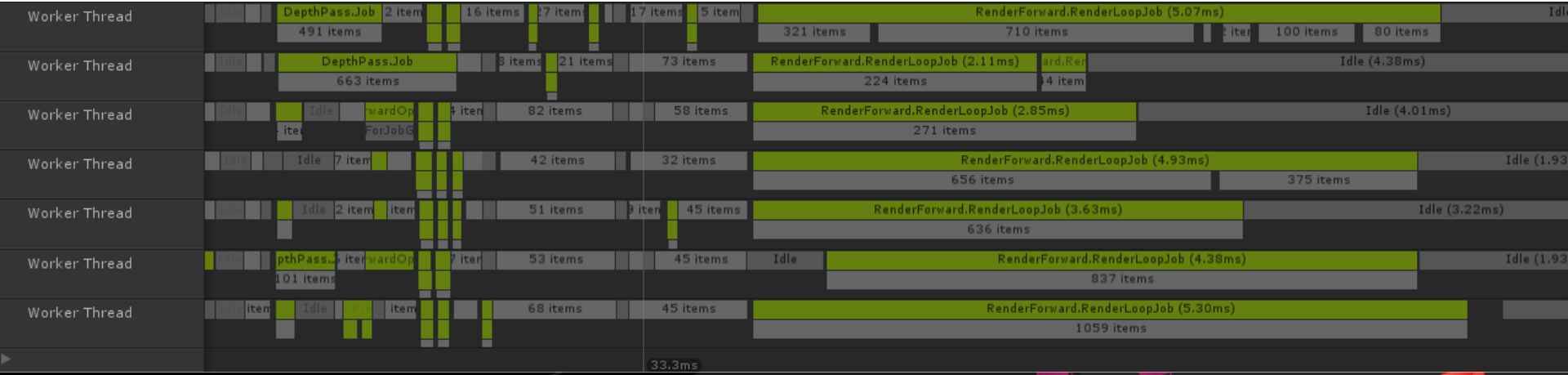


Engine evolution / Multithreading

- It's not just Ashes any more ☺
- Engines are getting ready for *massive* multithreading

Engine evolution / Multithreading

- Here's Unity firing up all cores!

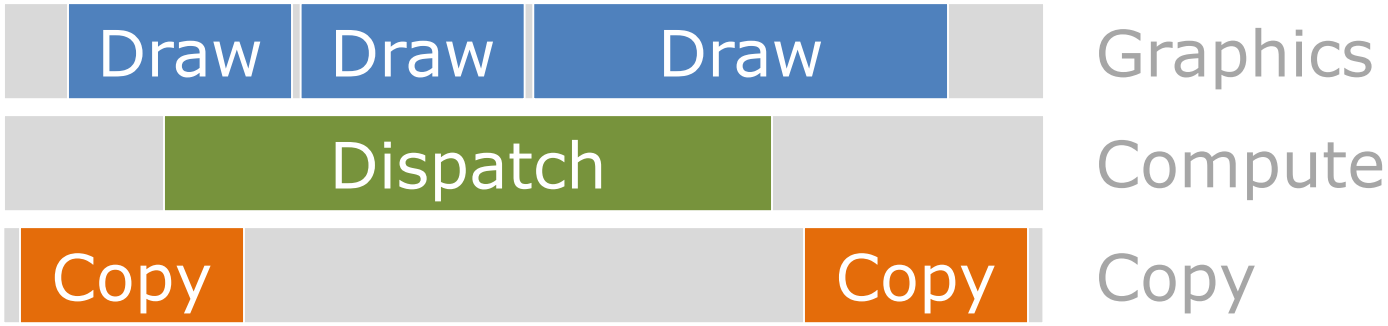


Engine evolution / Multithreading

- Plan for >10 threads
  - Increase draw call count – high/ultra settings on *new APIs only*
  - Or: Cut latency! Twitchy 144 fps games, anyone?

Graphics, compute, copy queues

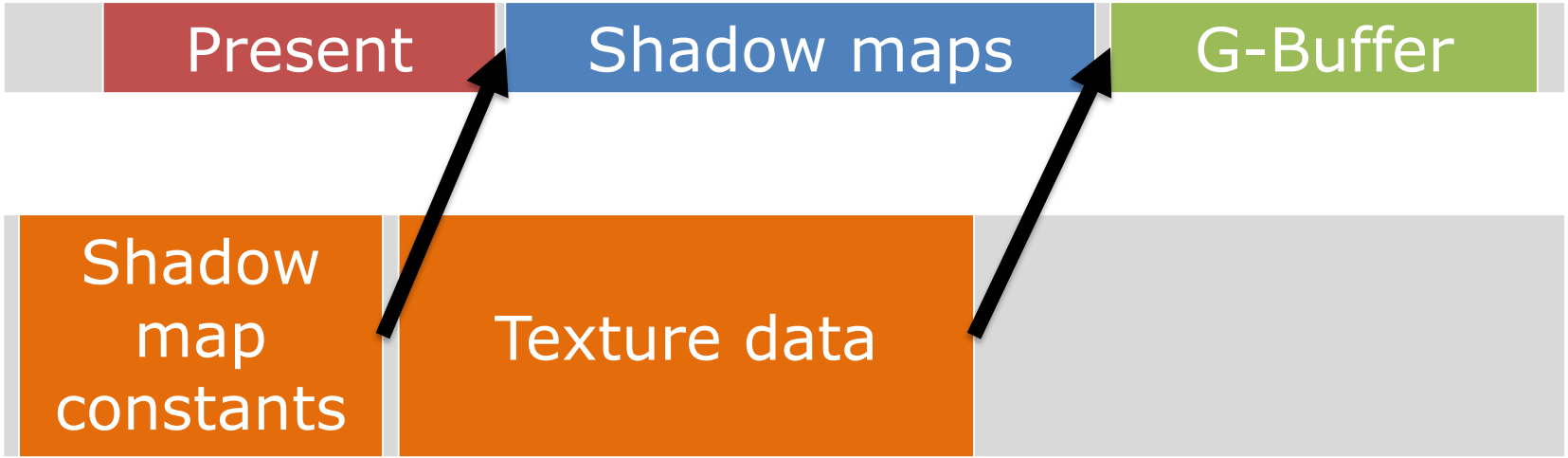
- Schedule independent work on independent queues
- Fill up the whole GPU



Lessons learned / Copy queue

- Copy queue is low-latency, low-speed, but it's **separate hardware**
  - Copy queue is **optimized for transfer over PCIe®**, not for GPU local copies
  - For PCIe®, it is the **fastest way** to transfer data
  - Avoid waiting on copy queue from graphics/compute
  - Ideal use of copy queue is streaming data over a few frames
- Some games still don't use it …
  - **Multi-millisecond-savings** are common
  - If you go from CPU to GPU or back, the copy queue is the queue of choice!

Copy queue

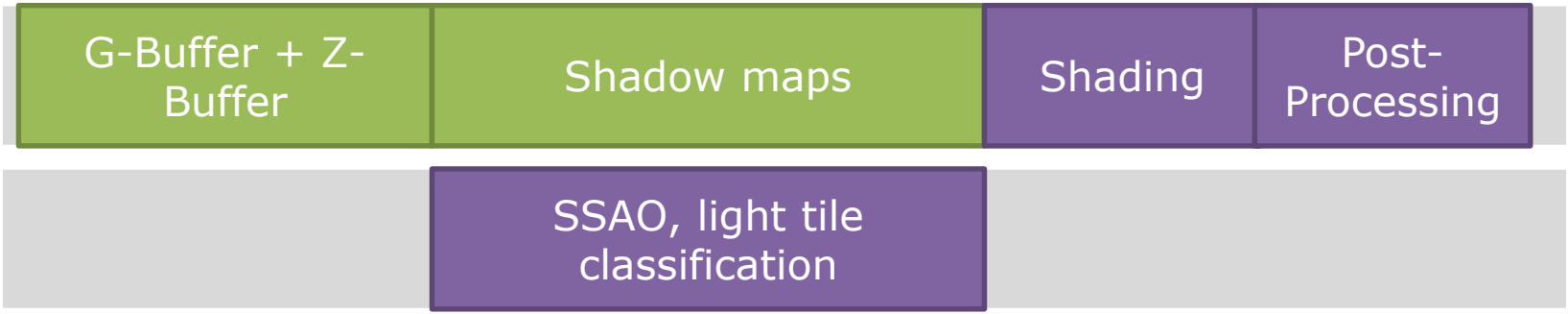

Present
Shadow maps
G-Buffer
Shadow map constants
Texture data

Lessons learned / Copy queue

- Use to it upload all your buffers (constants, index buffers, etc.)
- Use it to defragment memory

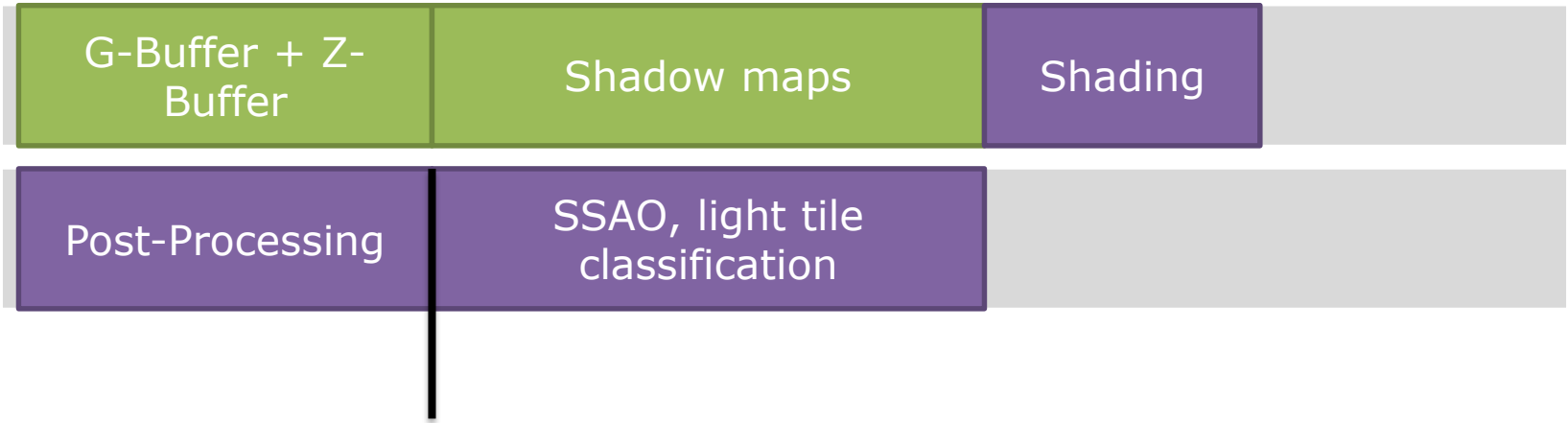
Lessons learned / Async compute

- Most games right now



Lessons learned / Async compute

- Best performance & production proven!



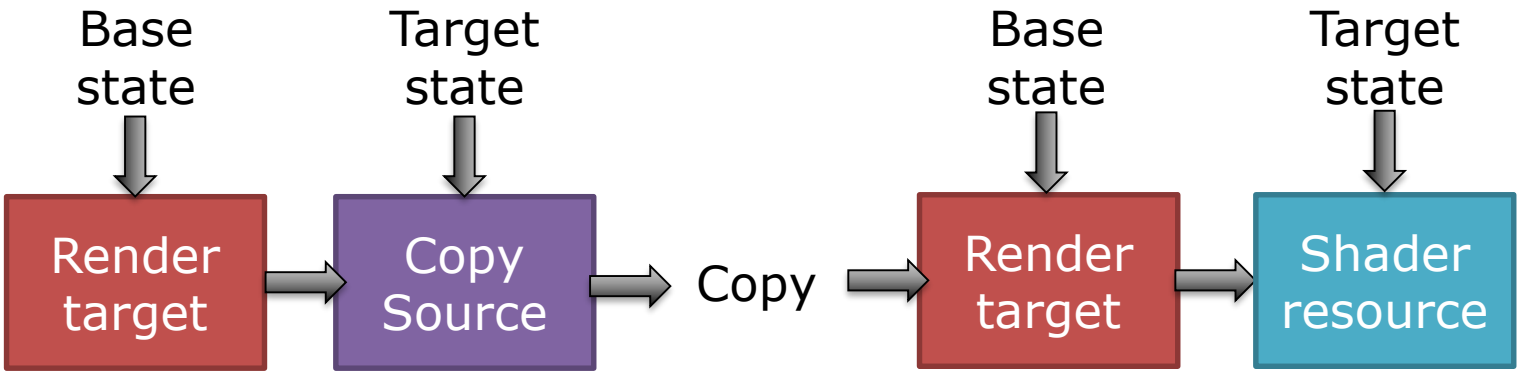
Lessons learned / Barriers

- Barriers are **hard**
- Most issues come from **retrofitting** engines
- New engine designs handle them **much more robustly**

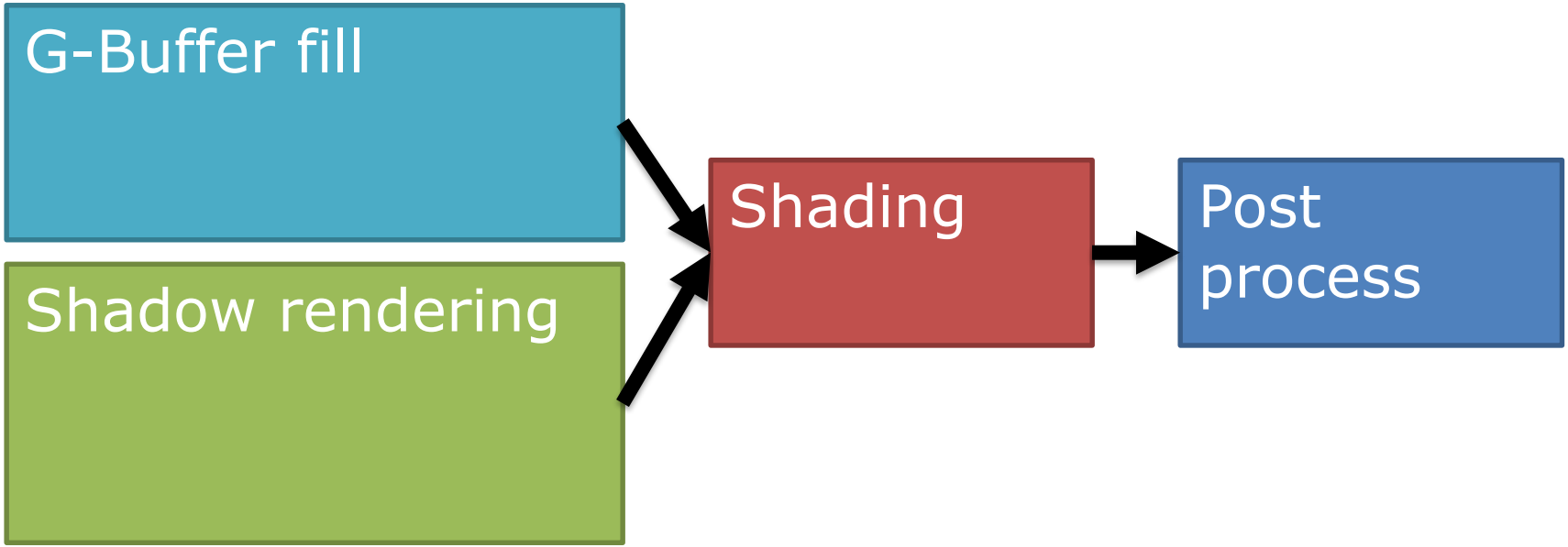
Lessons Learned / Barriers

- Missing barriers
  - Validation layer helps
  - No longer a big issue
- Missing batching
- The „base state“ problem

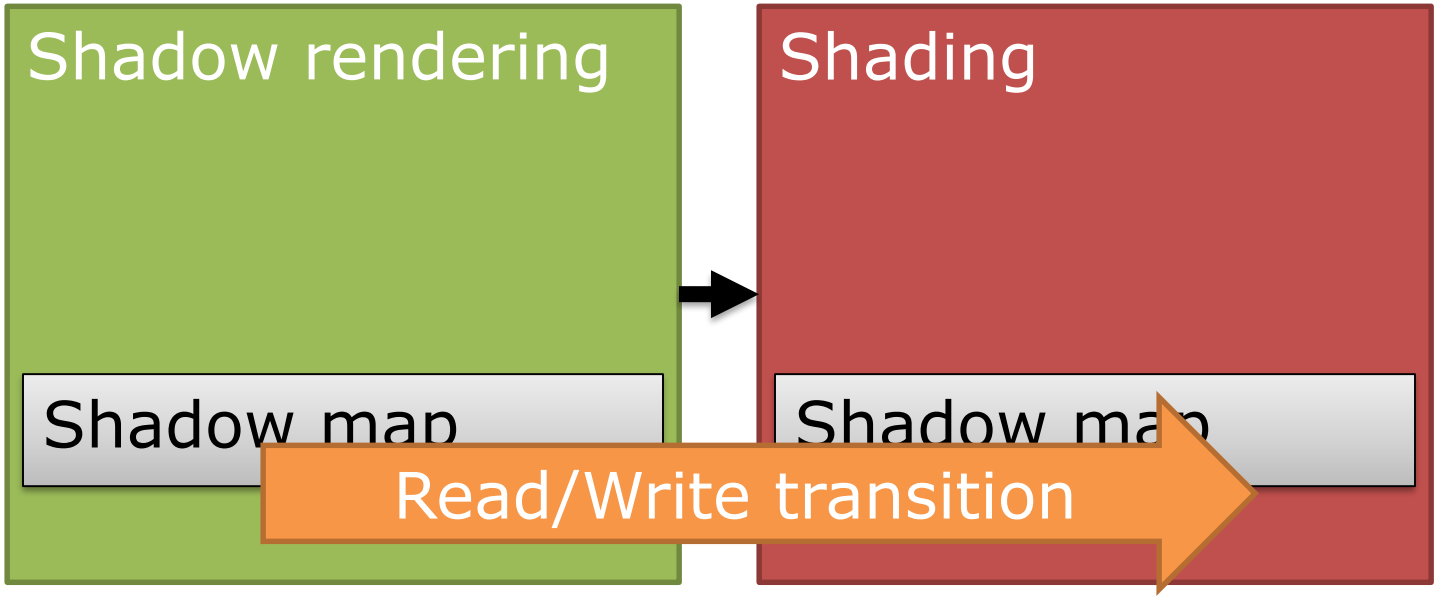
The “base state” problem



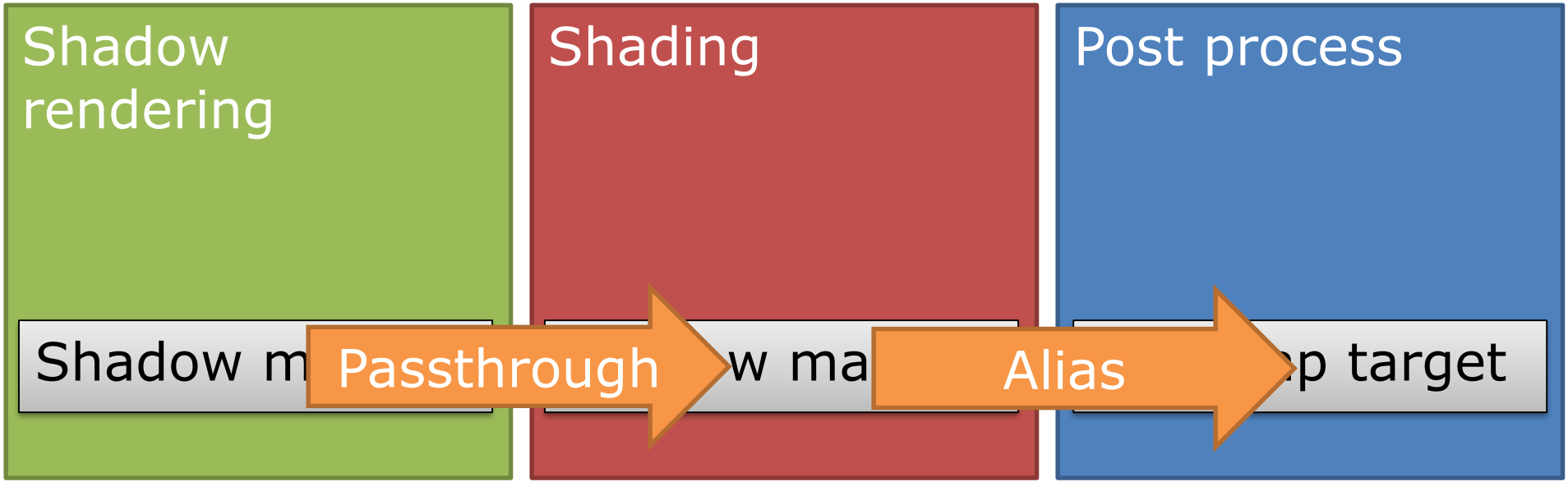
Engine evolution / Task graphs



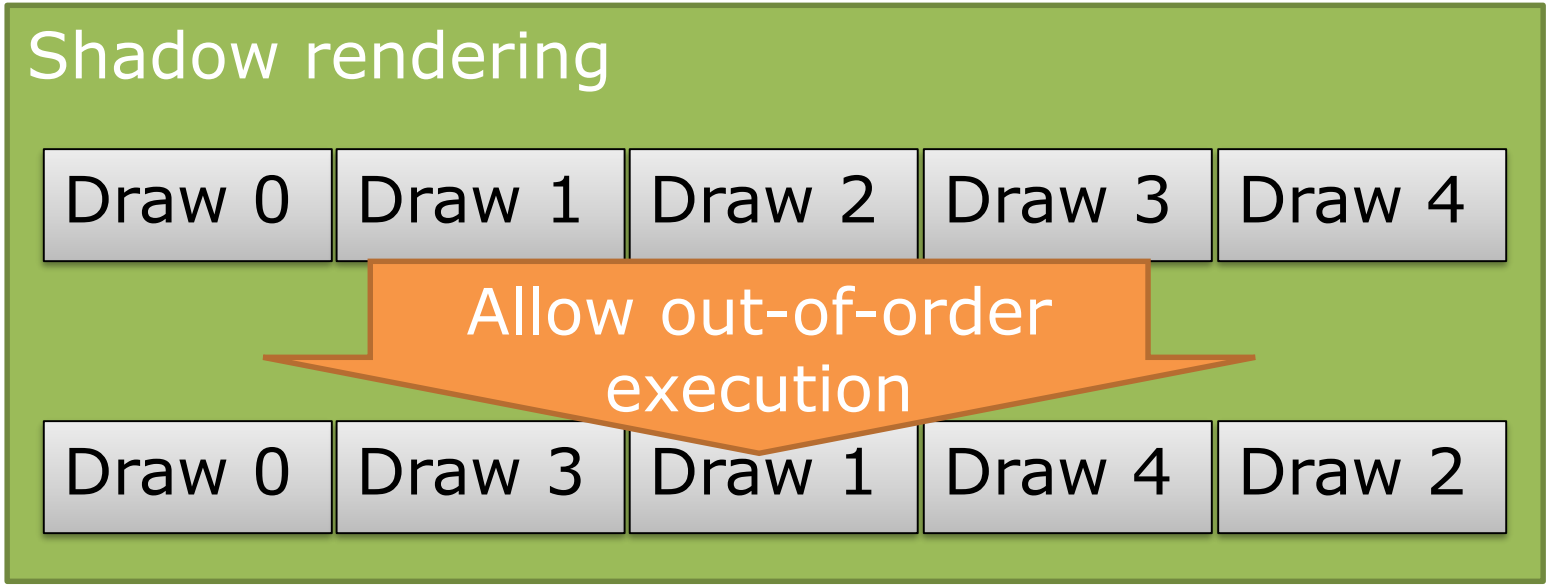
Engine evolution / Task graphs



Engine evolution / Task graphs



Engine evolution / Task graphs



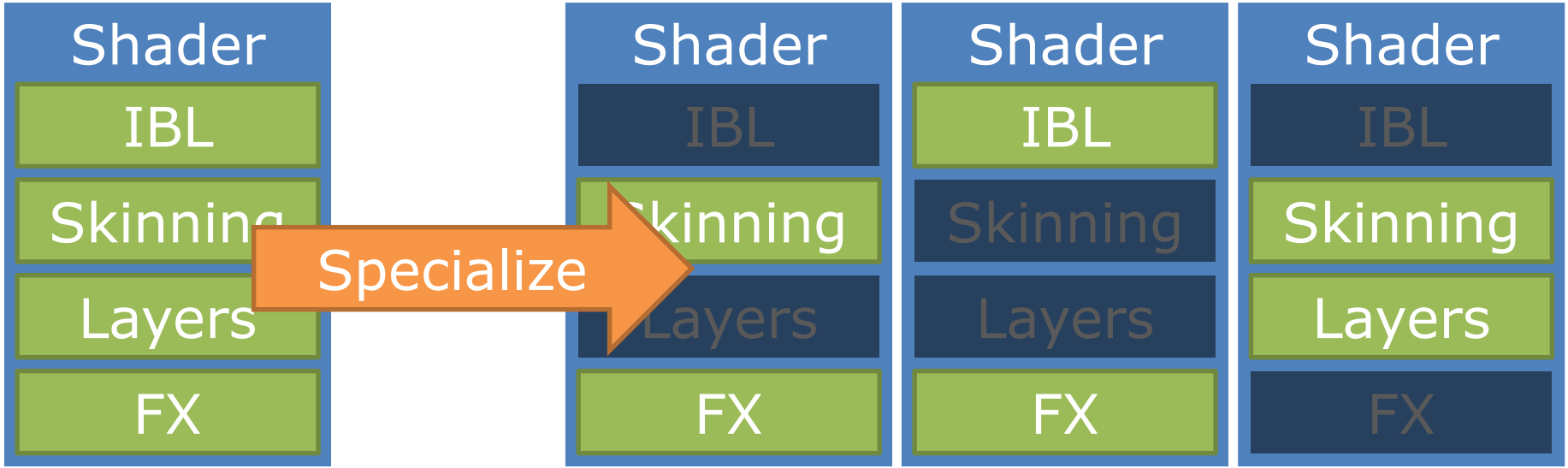
Engine evolution / Barriers

- Manual handling *doesn't cut it any more*
- Need higher level abstractions – render graphs
  - This is happening – check out the FrameGraph presentation from Frostbite!
  - Native support in Vulkan since day 1

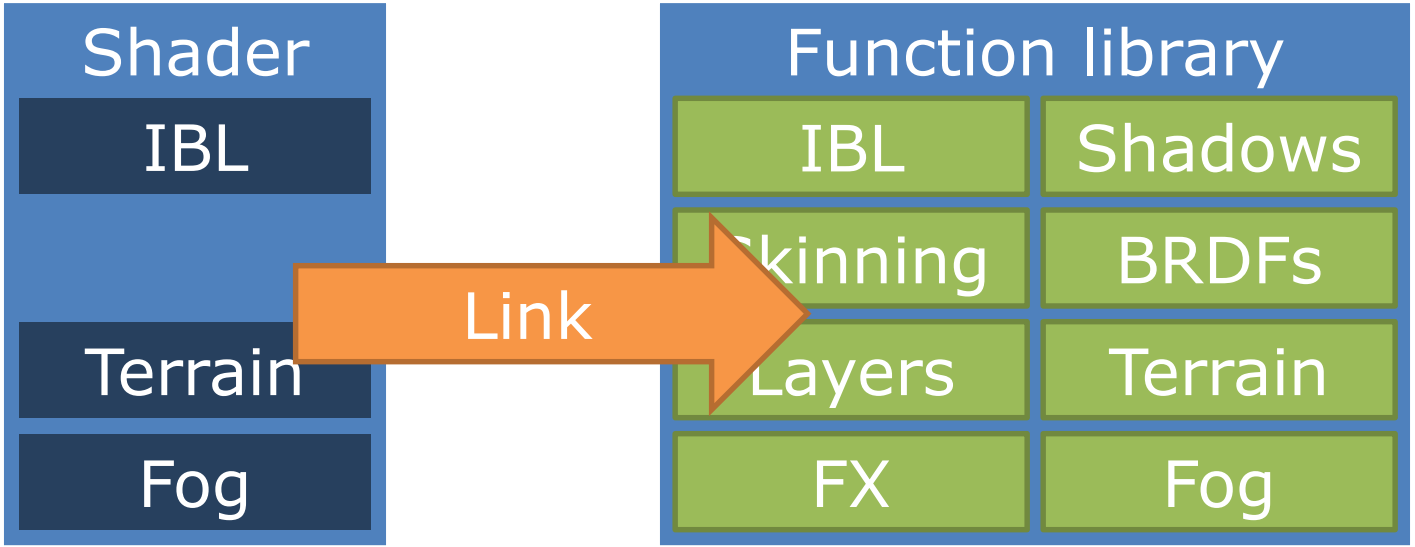
Engine evolution / Shaders

- Shader permutations are getting fewer
  - Doom has only a couple hundred total
  - More games are changing creation pipelines to prune variations earlier
- More high-level work (around compilers) is happening

Engine evolution / Shaders



Engine evolution / Shaders



Engine design

- Engines moving towards more **high-level** rendering
- APIs improve to make them easier to use
- Gamers benefit!

Open topics / Scalability

- Scalability is not solved at all yet
  - Games support old and new APIs for all settings
  - Mobile → desktop increasingly important
- New APIs *only* seems to be the path forward

A new approach to APIs

- **Strong collaboration** between ISVs, IHVs and standard bodies
- **APIs evolved** along with game engines
- Loads of changes since release to make your life easier!

Conclusion

- APIs continue to change
  - What do **you** need?
  - What would make **your** life simpler?
- Community collaboration is critical
  - Especially for shader language changes
  - It’s easy to contribute – give it a shot!

Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

AMD