Game Developers Conference

Applying DirectX* Sampler Feedback: Texture Space Shading and Streaming with DirectStorage*

John Gierach, Graphics Driver Performance Lead
Allen Hux, Graphics Software Architect
Intel Corporation

Agenda

- Overview
- Texture Space Shading
- UL 3DMark* Sampler Feedback Feature Test
- Mip Region Size
- Conclusion & Call to Action
- References

D3D12 Sampler Feedback Background

What is feedback?

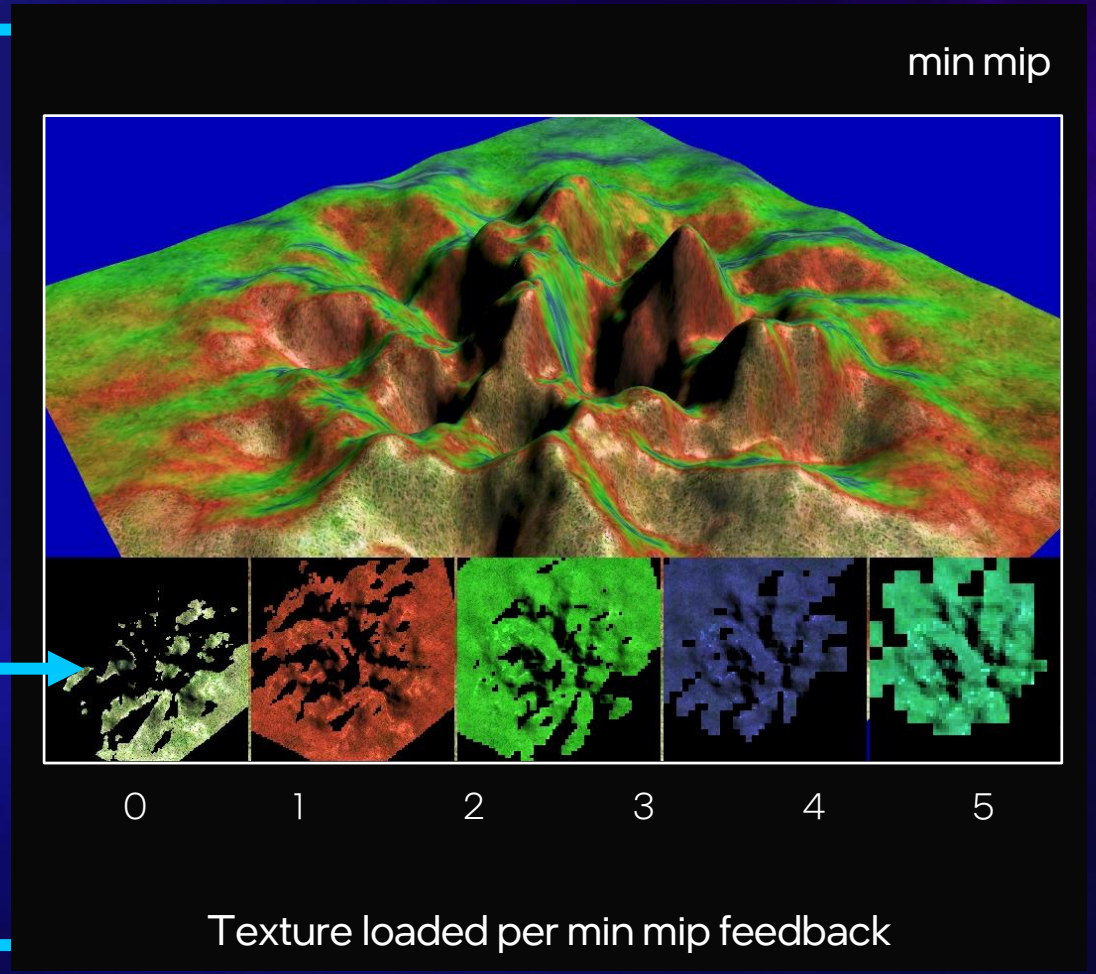
- The reverse of texture sampling: which texels were read?
- Efficiently determine what the hardware did
- Pair “feedback” texture with your “real” texture asset

There are two types of Sampler Feedback:

- Mip Region Used
- Min Mip Feedback

Mip Region Used

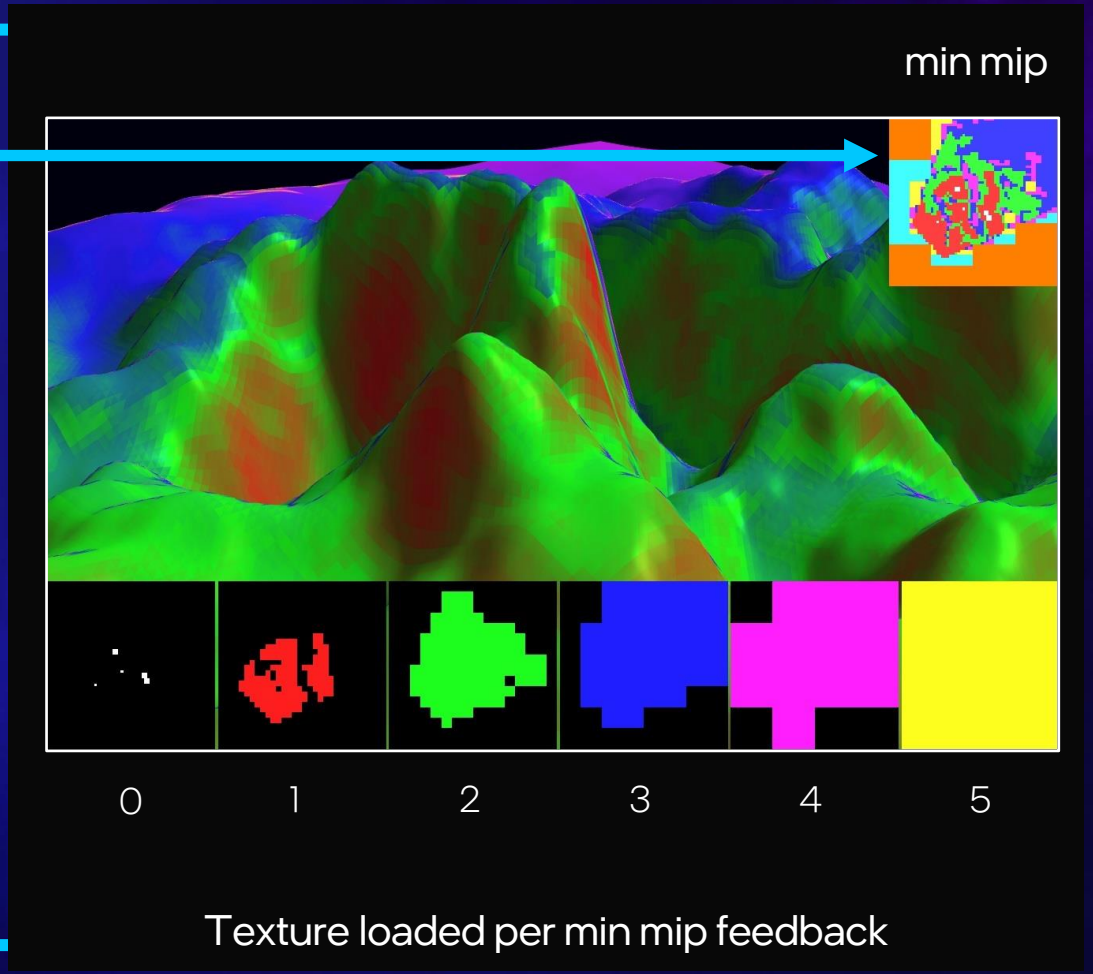
- Feedback per mip region within a mip
- Texel value = 0xFF if any texel in region touched
- Good for texture space shading



Mip Mip Feedback

Min Mip Feedback

- Texel value = min mip sampled for a mip region
- Stay tuned for Allen's talk for all the details



Agenda

- Overview
- Texture Space Shading
- UL 3DMark* Sampler Feedback Feature Test
- Mip Region Size
- Conclusion & Call to Action
- References

Texture Space Shading

Pass 1:

Perform expensive lighting calculations and store them in textures

Pass 2:

Rasterize scene while using lit textures from Pass 1

Why Use Texture Space Shading?

Skip redundant lighting calculations

- Reuse within the same frame (e.g. VR rendering)
- Reuse across frames

Shading rate decoupled from rasterization rate

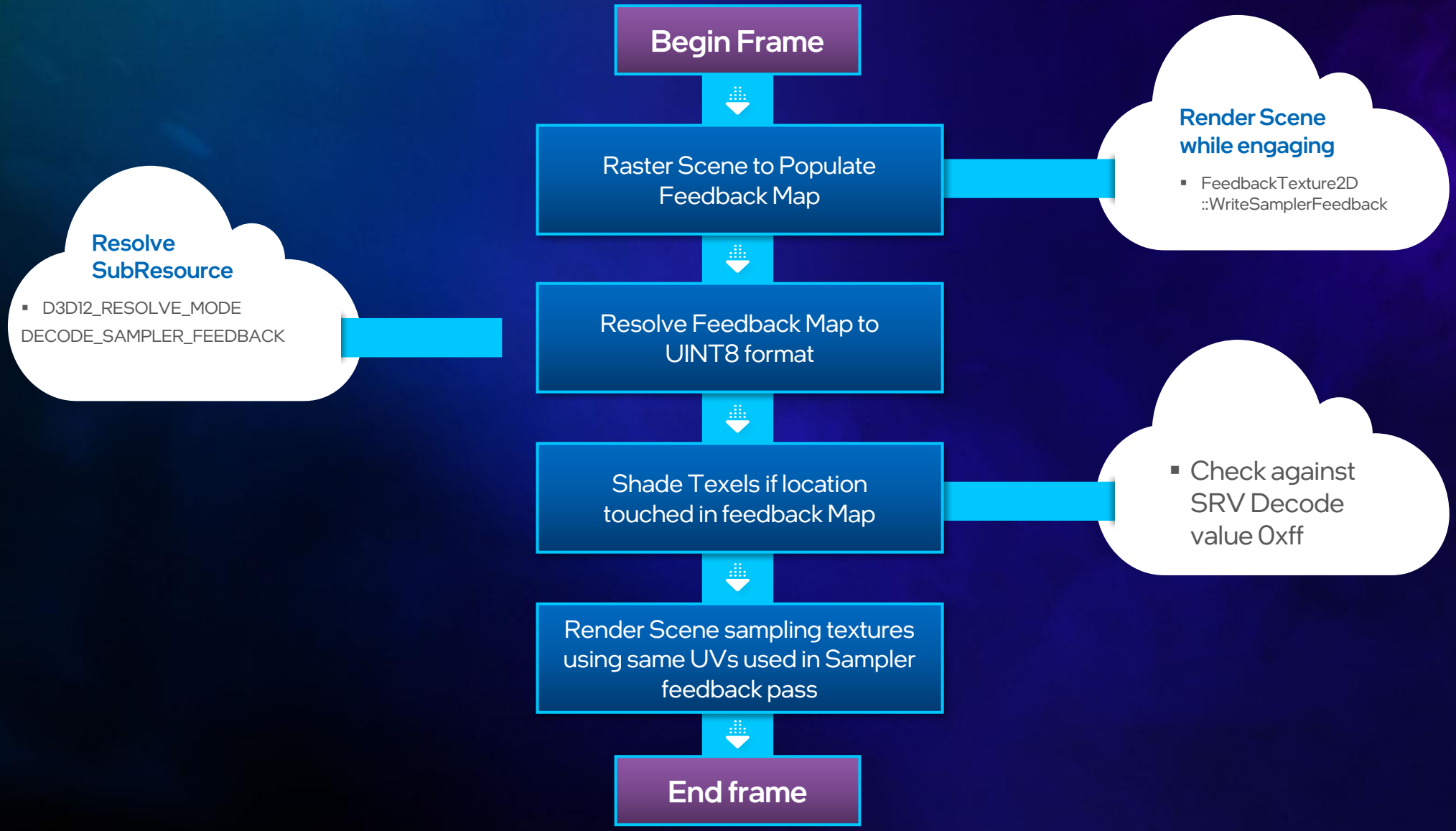
- Performance versus quality adjustable with sampler bias

Remove shimmer artifacts rendering from far objects

How Does Sampler Feedback Help?

- Less Texels Shaded == Better Performance
- Sampler feedback will tell us which texels will be sampled during rasterization
- Only shade texels that will be sampled during rasterization

DX12 Sampler Feedback Flow



Agenda

- Overview
- Texture Space Shading
- UL 3DMark* Sampler Feedback Feature Test
- Mip Region Size
- Conclusion & Call to Action
- References

UL 3DMark* Sampler Feedback Feature Test

UL 3DMark* Sampler Feedback Feature Test

Feature test designed to benchmark sampler feedback performance

Implements 2 modes:

- Sampler Feedback
- Software emulated sampler feedback

Intel Gen 11 results:

- 23% net workload benefit using Sampler Feedback
- Sampler Feedback pass 3.1x faster than emulated path

intel

Workload Design

Sampler Feedback using "Deferred" Approach

Resource Initialization

Create Sampler Feedback Resource

- Dxgi_format_sampler_feedback_mip_region_used_opaque
- Mip Region Size 8 x 8 x 1
- Mip Count 5

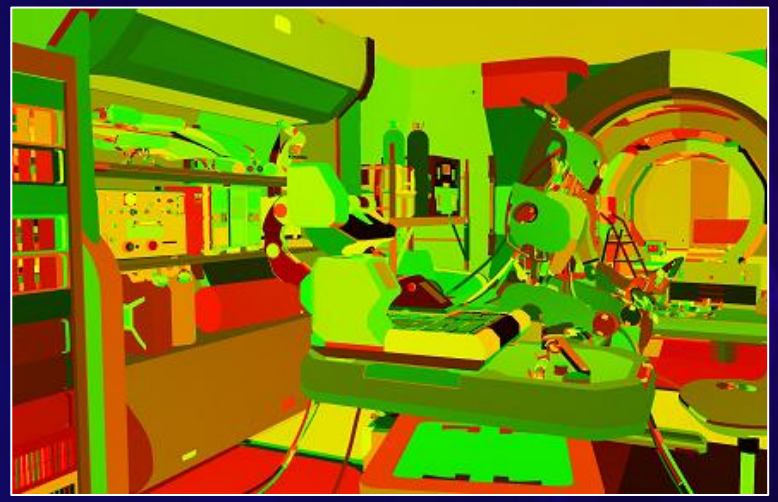
Create Paired Resource

Create Feedback View

- CreateSamplerFeedbackUnorderedAccessView
- Maps to FeedbackTexture2D in HLSL

Collect Final UVs and Gradients

- Rasterize all scene geometry
- Depth test enabled
- Depth write enabled
- Write UV to render target
- Write Gradient ddx(UV), ddy(UV) to render target



Engage Sampler Feedback

Full screen pixel shader pass

Load UVs and gradients

Call Write Sampler Feedback Grad with inputs

- Store results to FeedbackTexture2D object

Performance tip: Application can stochastically skip WriteSamplerFeedbackGrad calls.

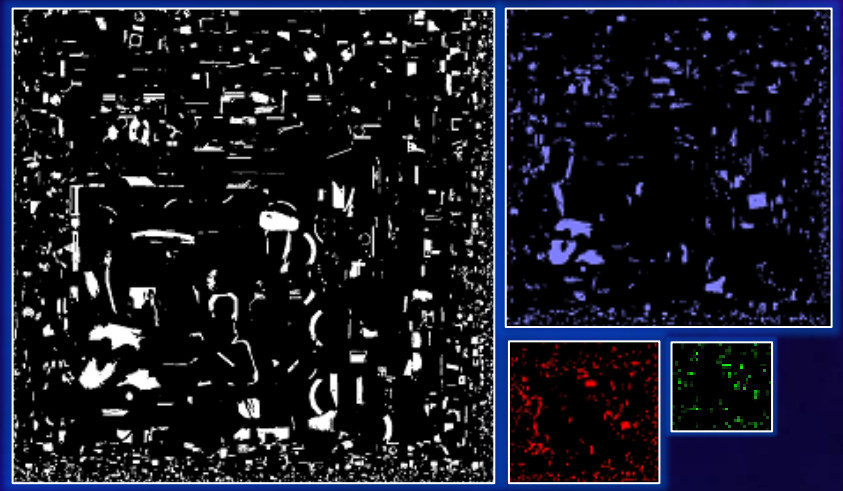
*Ensure proper image quality when trying this

Resolve Sampler Feedback

Call ResolveSubResourceRegion with
D3D12_RESOLVE_MODE_DECODE_SAMPLER_FEEDBACK

After resolve, touched feedback texels will have 0xFF

Images on right visualize mips touched



Performance tip:

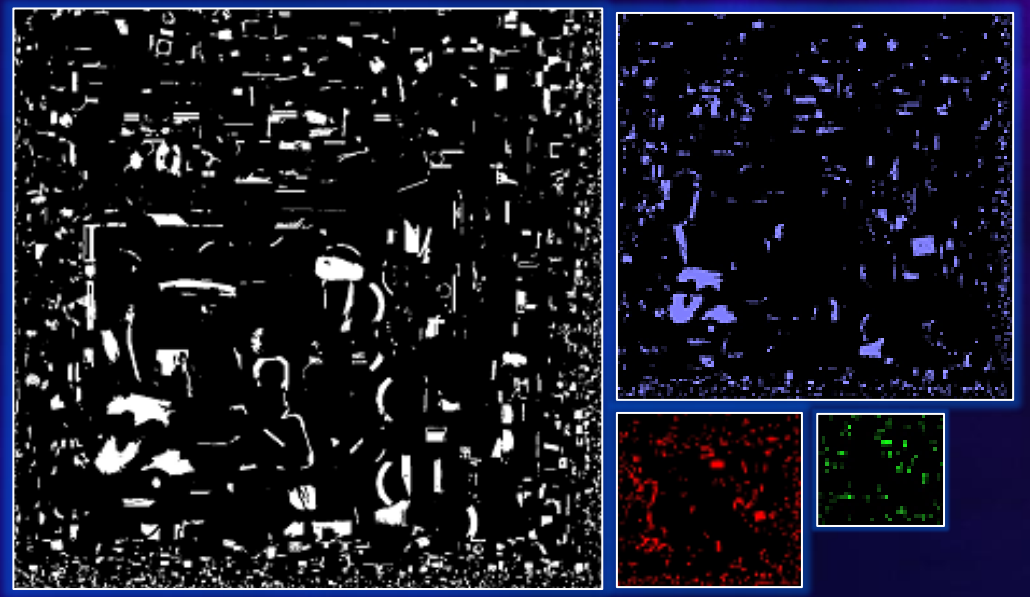
- Batch barriers for transitions to/from resolve states
- Resolve entire mip chain in one ResolveSubResourceRegion call with sub resource index UINT_MAX

Compaction

Goal: Only dispatch compute shader threads for regions that need to get texel shaded

Build data for Execute Indirect:

- Thread group count
- Pixel XY offset per thead group

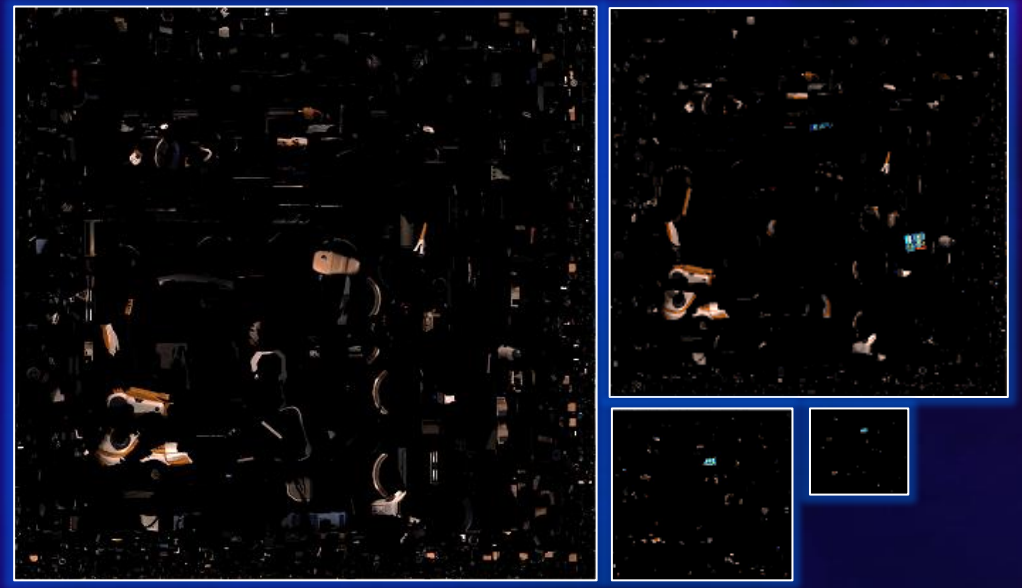


Texture Space Shading

Perform shading for all touched texels in feedback map

Implemented using ExecuteIndirect

Performance tip:
Use results from higher level mips if available to save costly lighting calculations



Render Final Output

- Full screen pixel shader pass
- Sample shaded texels
- Use SampleGrad with same parameters as Sampler Feedback pass
- Tone map

Special thanks to our partners @ UL for developing this workload!

Feature Test coming Q3' 2021

Visit https://benchmarks.ul.com/3dmark for more information!

Agenda

- Overview
- Texture Space Shading
- 3DMark* Sampler Feedback Feature Test
- Mip Region Size
- Conclusion & Call to Action
- References

Mip Region Size

Mip Region Size will map a texel in the feedback map to a region in the paired texture

Different Mip Region sizes will change the performance

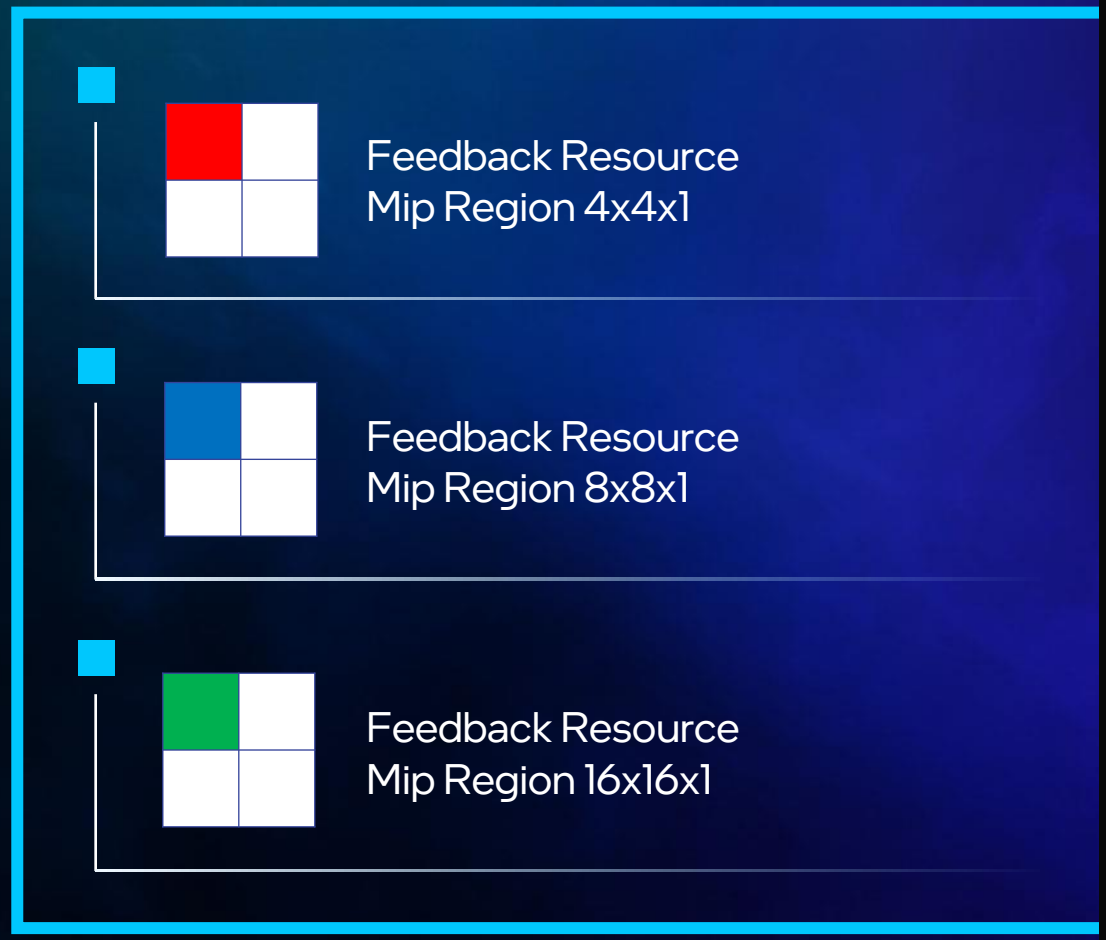
Smaller Mip Region results in finer granularity of a mip region used.

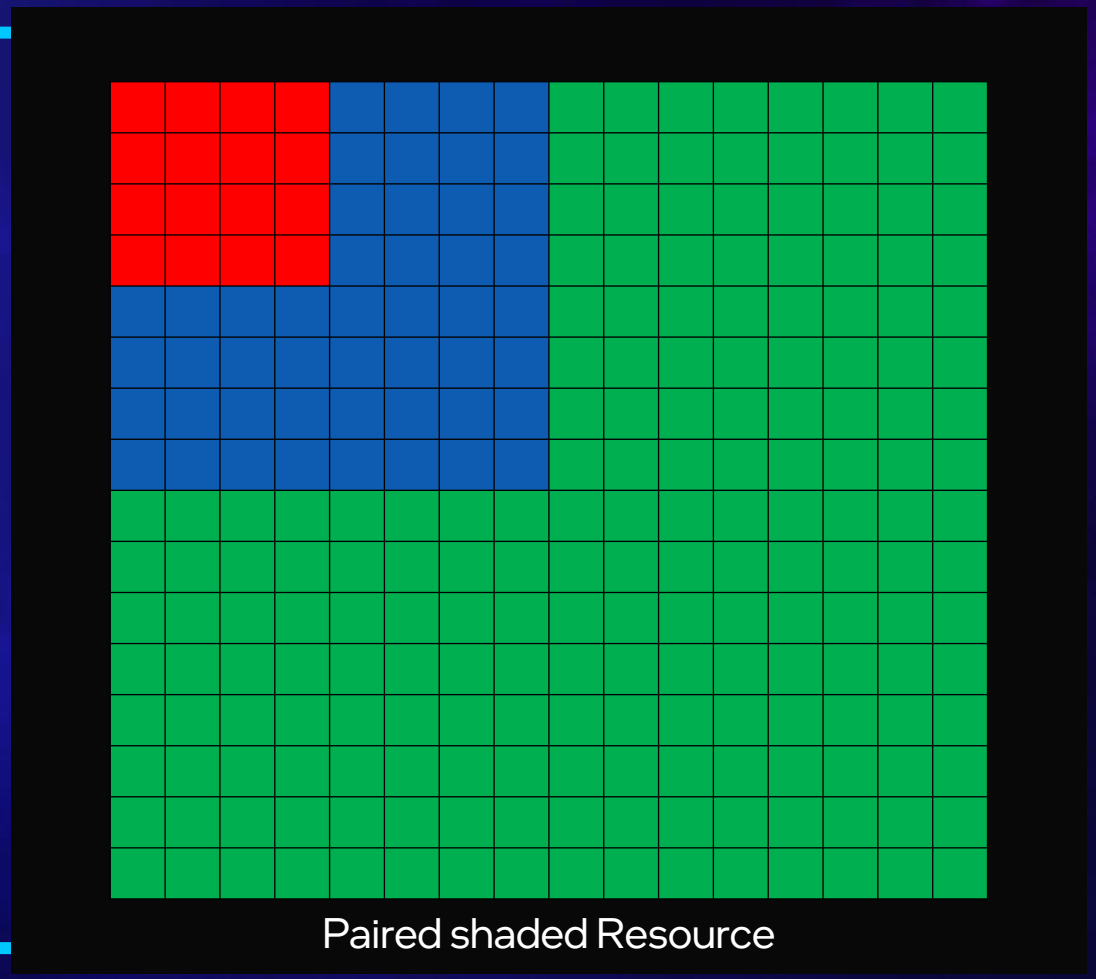
Smaller Mip Region will result in a larger feedback resource. Which will have:

- Higher cost for clears
- Higher cost for resolves
- Higher bandwidth cost
- Potentially less shaded texels

Example Data to follow!

Mip Region Example





Mip Region Size Performance Characteristics

Agenda

- Overview
- Texture Space Shading
- 3DMark* Sampler Feedback Feature Test
- Mip Region Size
- Conclusion & Call to Action
- References

Summary & Call to Action

Intel Gen11 processors support Sampler Feedback

Begin developing with Sampler Feedback feature today!

We can't wait to see how innovative developers will use the feature!

https://benchmarks.ul.com/3dmark
https://store.steampowered.com/app/223850/3DMark/

Thank You!

Up Next:
Sampler Feedback Streaming with Microsoft Direct Storage*

Sampler Feedback Streaming with DirectStorage* for Windows*

Allen Hux, Intel

Agenda

- Asset Streaming Opportunity
- D3D12 Sampler Feedback Background
- D3D12 Reserved Resources
- Connecting DirectStorage*
- Results
- Conclusion & Call to Action
- References

Asset Streaming Vision

We can draw scenes using assets that, together, far exceed physical memory if we stream just what's needed per frame.

D3D12 Sampler Feedback identifies what to stream

DirectStorage* for Windows makes streaming simple and efficient

Build Previously Impossible Scenes

- **1000** objects
  350MB texture for each (16k x 16k bc7)
  no texture re-use
- **350 GB :** total memory for assets
  230 MB : physical memory used
- **0.06%** resident (230MB/350GB)

Textures courtesy Hubble

Agenda

- Asset Streaming Opportunity
- D3D12 Sampler Feedback Background
- D3D12 Reserved Resources
- Connecting DirectStorage*
- Results
- Conclusion & Call to Action
- References

D3D12 Reserved Resources

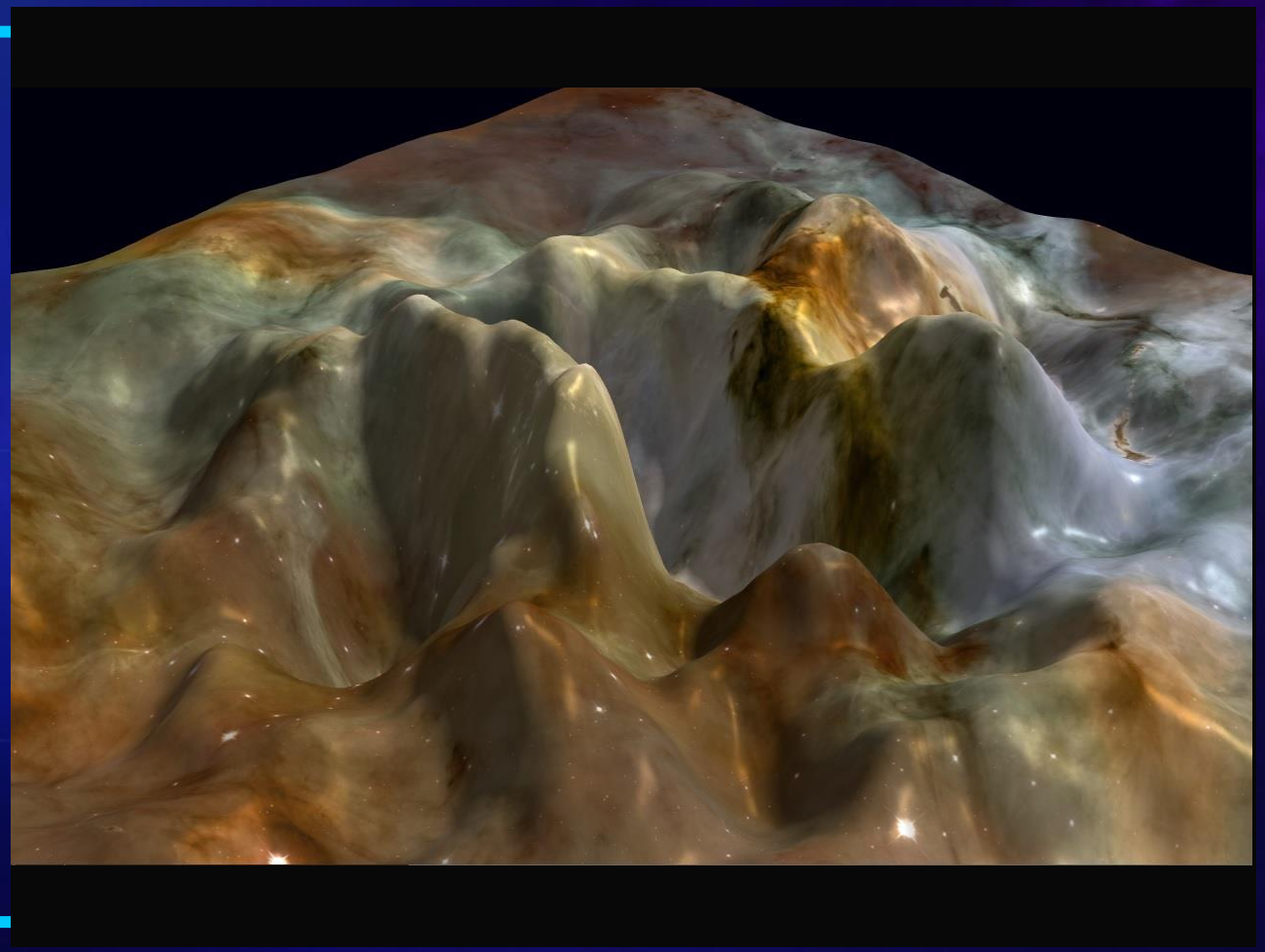
Easy memory management for massive assets

- High-performance virtual memory for textures
- Allows partial residency, sub-mip granularity
- 64KB Tiles, dimension a function of texture format
- Tiles from multiple resources in 1 or more heaps

ID3D12Device::CreateReservedResource

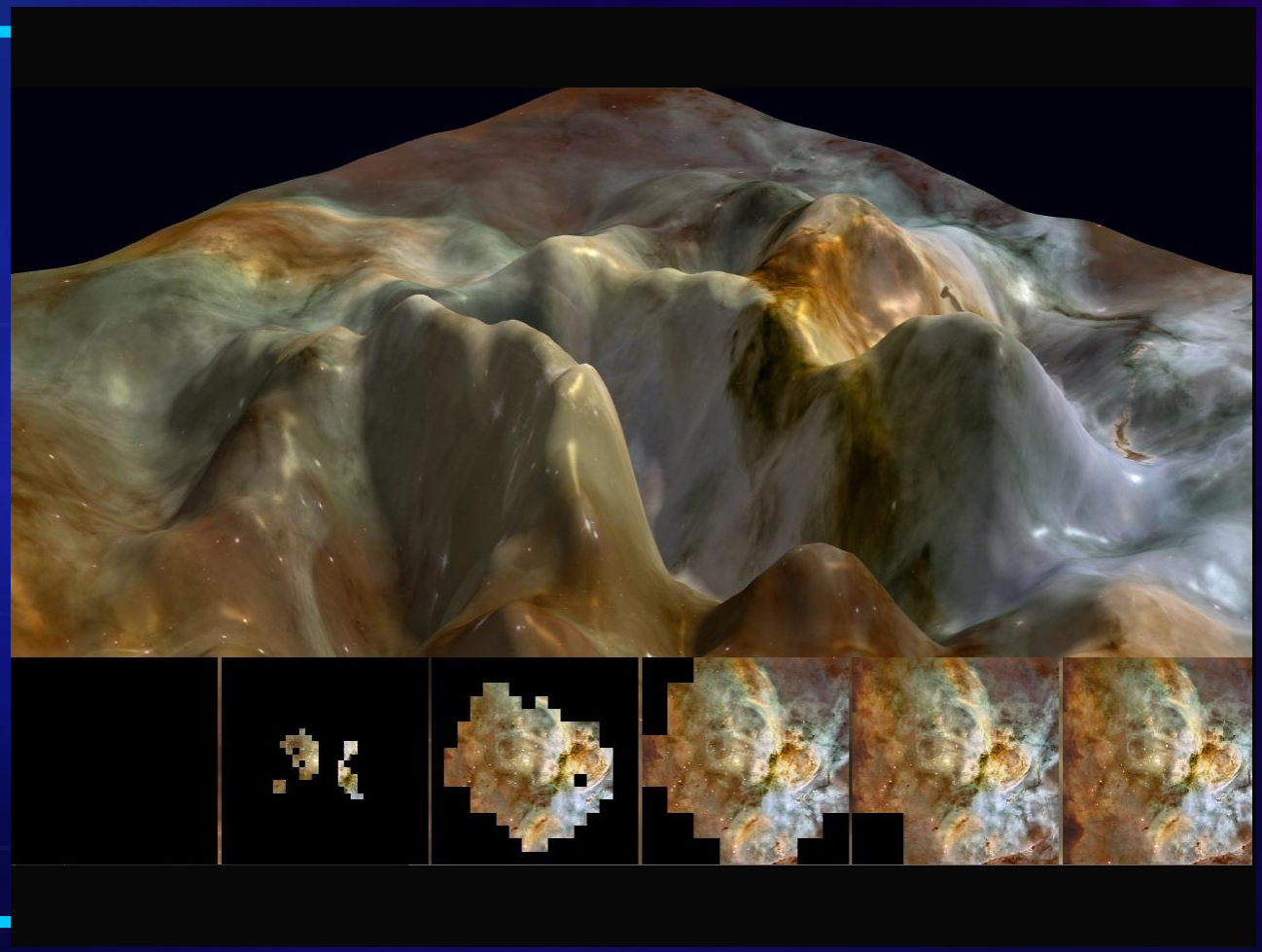
Example: Texture on Terrain

- Example of a reserved resource
- This texture is only partially loaded



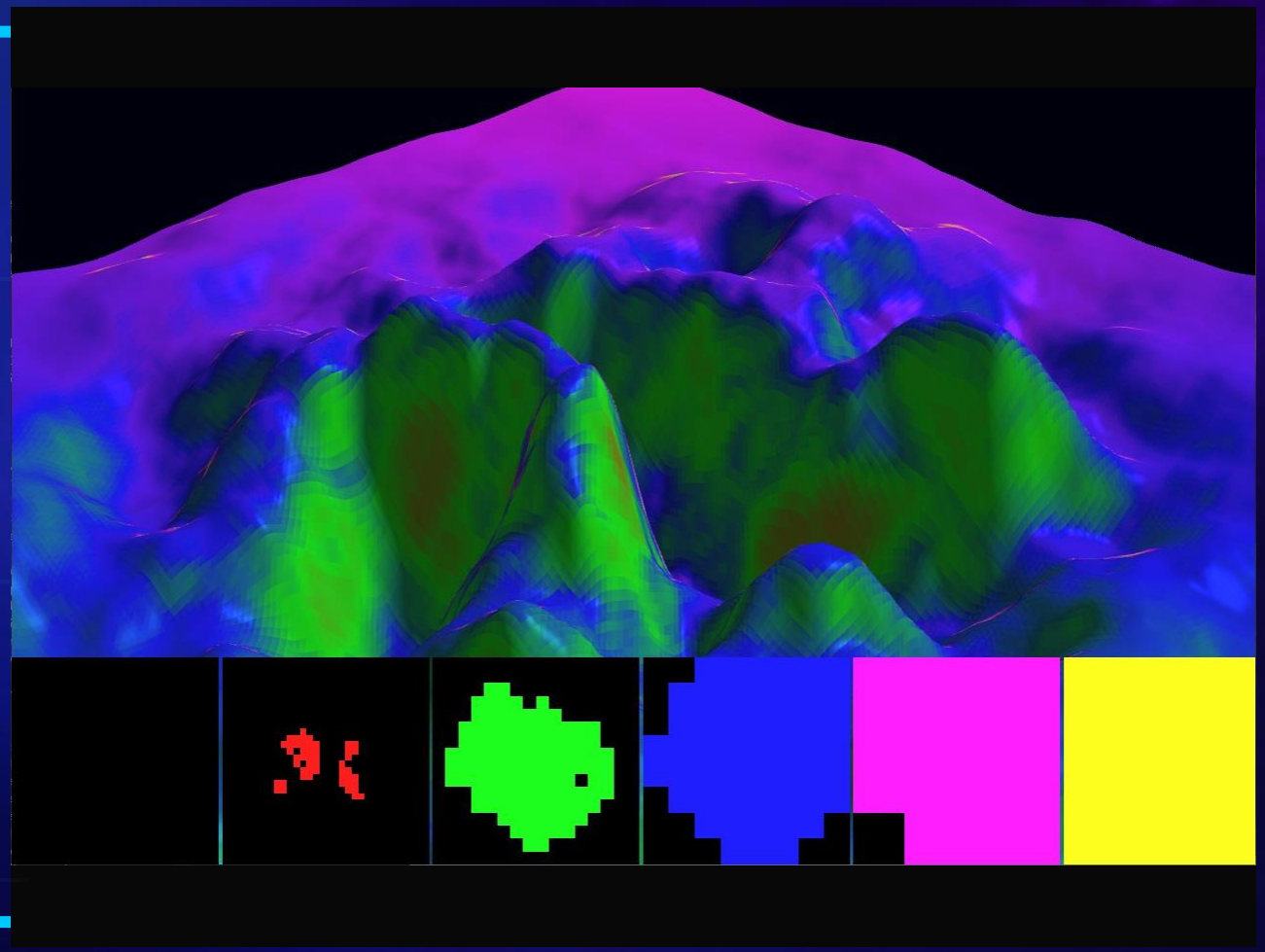
Example: Texture on Terrain

- Now showing the mips
- No tiles of mip 0 loaded
- mip 1 partially loaded



Example: Texture on Terrain

- Set color = mip level
- Can more clearly see how tiles correspond to the visible texture
- In demo, all tiles (for 350GB or assets) fit within a single 1GB heap



Agenda

- Asset Streaming Opportunity
- D3D12 Sampler Feedback Background
- D3D12 Reserved Resources
- Connecting DirectStorage*
- Results
- Conclusion & Call to Action
- References

D3D12 Sampler Feedback Background

What is feedback?

- The reverse of texture sampling: which texels were read?
- Efficiently determine what the hardware did

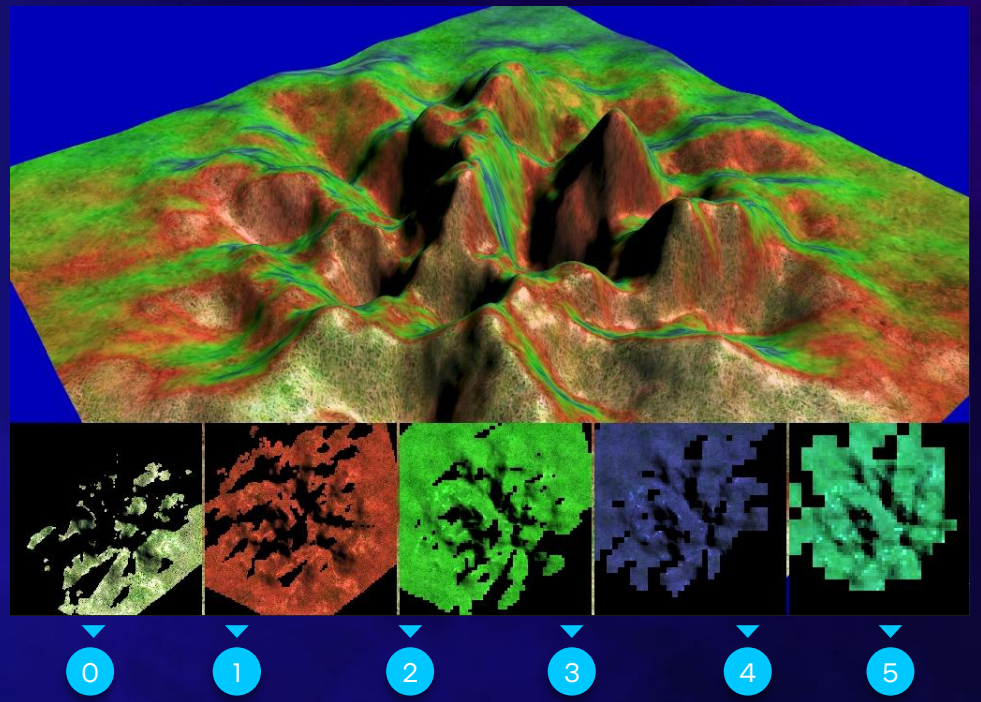
Sampler feedback resources are lower-resolution

Finest Granularity is 4x4

Two Kinds of Feedback

Mip Region Used

- multiple mip layers
- texel value = 0xff if any texel sampled
- good for texture space shading



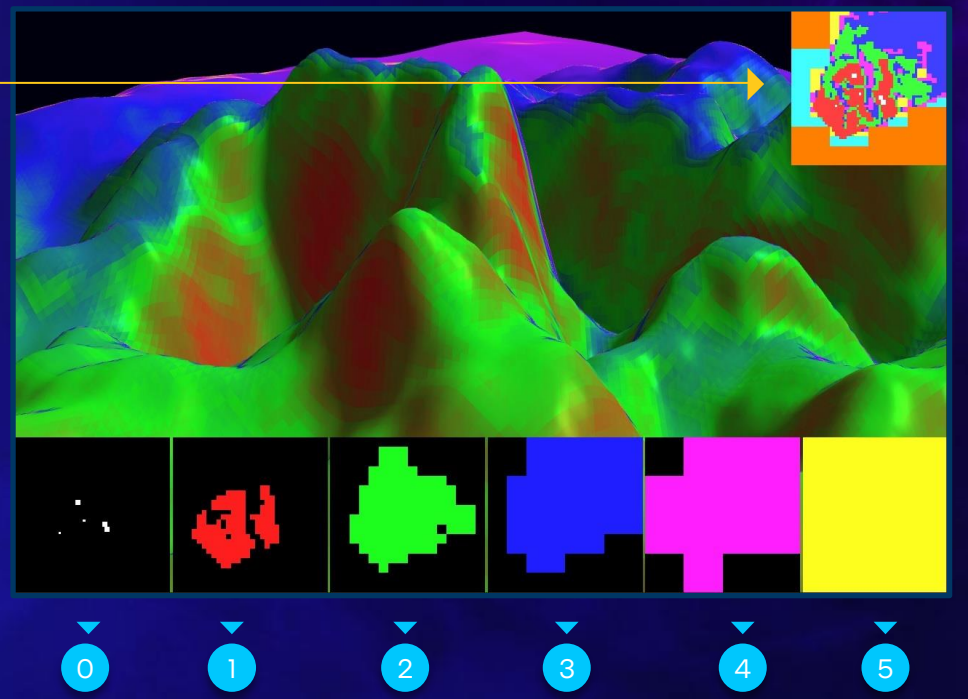
Two Kinds of Feedback

Mip Region Used

- multiple mip layers
- texel value = 0xff if any texel sampled
- good for texture space shading

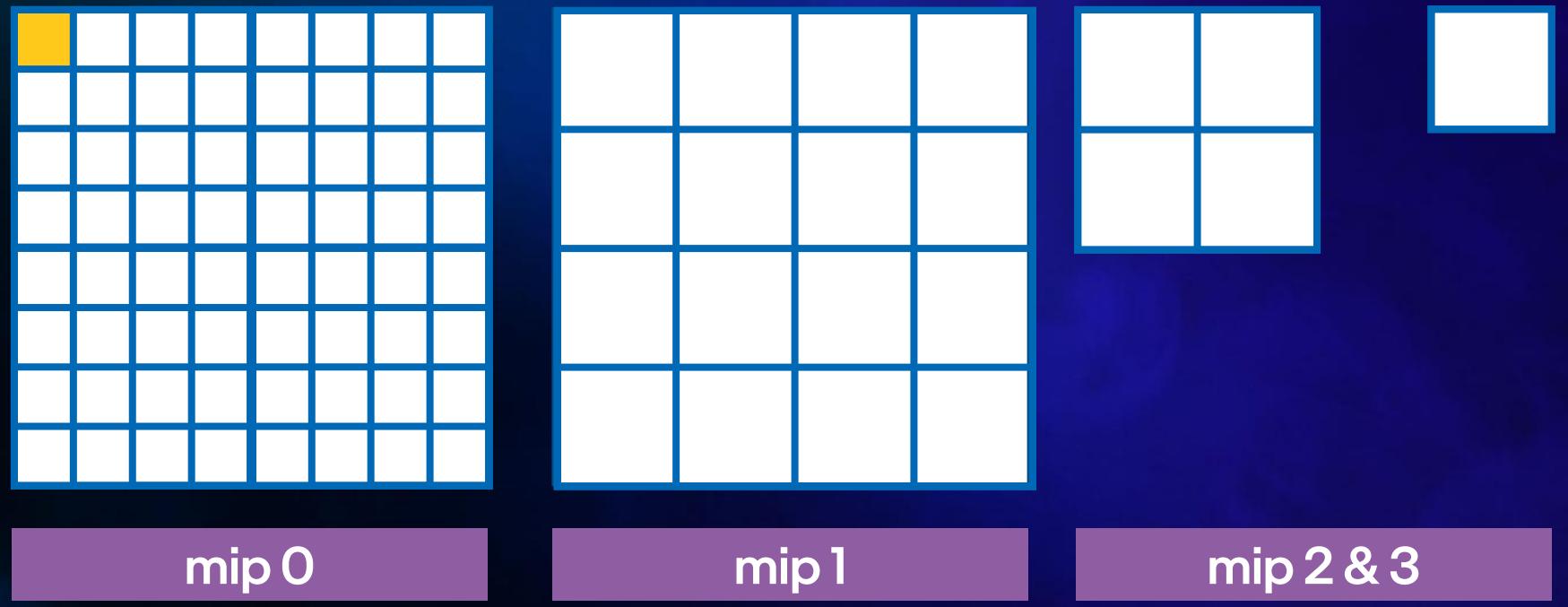
Min Mip Feedback

- single-layer
- texel value = min mip sampled



Min Mip Feedback Example

- Consider a 4x4 min-mip map, region size 4x4
- Sample the top left texel of mip 0 (orange)



Feedback

0	FF
FF	FF

Min Mip Map

- Feedback answers the question: was it sampled?
- Min mip map answers the question: is it resident?

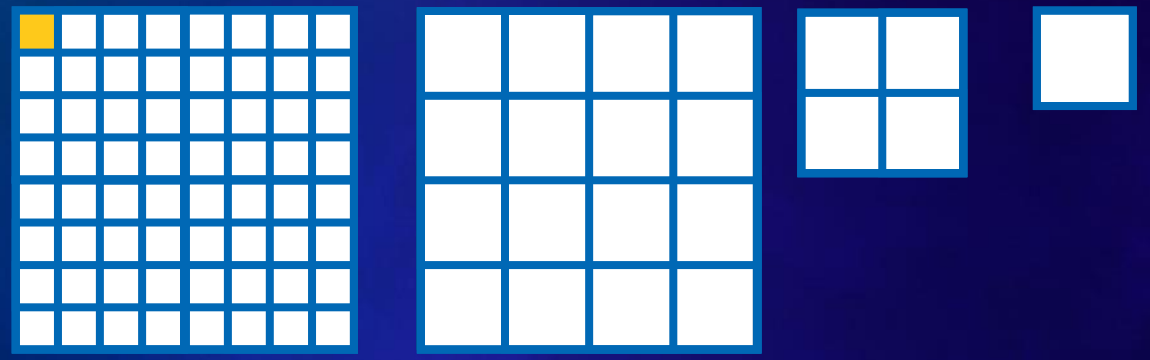
- Idea: if we load everything at & below region, no artifacts

  e.g. if mip 1 was sampled, trilinear/aniso will also sample layer 2

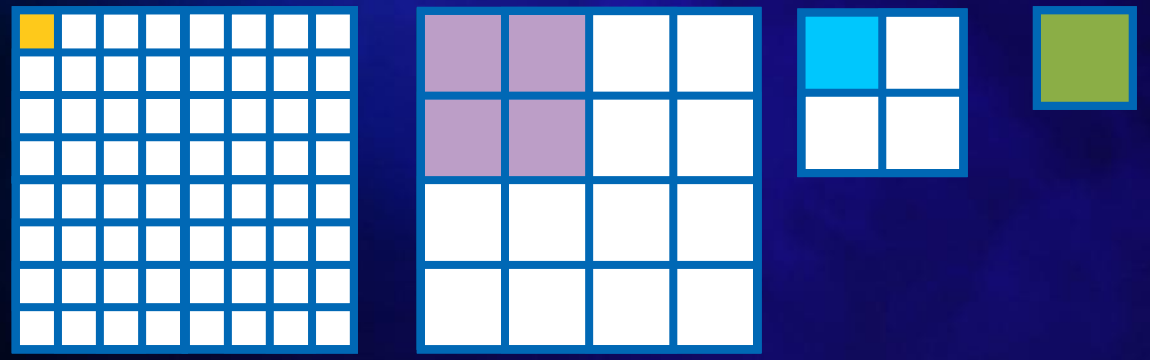
A min mip map can be created from min mip feedback

Example: Building a Min Mip Map

Sampler read orange region



Conforming texture must contain these regions



Min Mip Feedback

0	FF
FF	FF

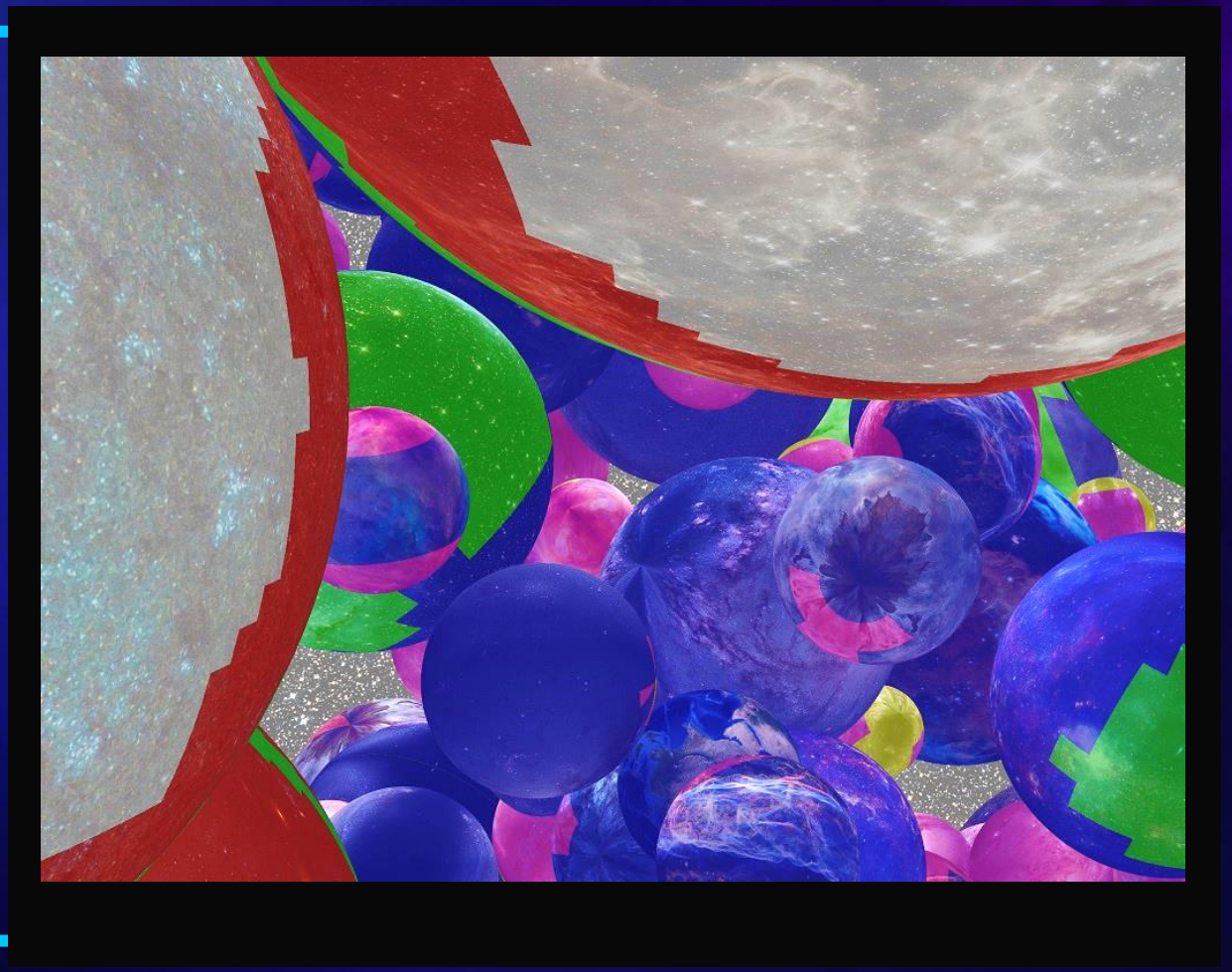
Min Mip Map

0	3
3	3

Sampler Feedback + Reserved Resources

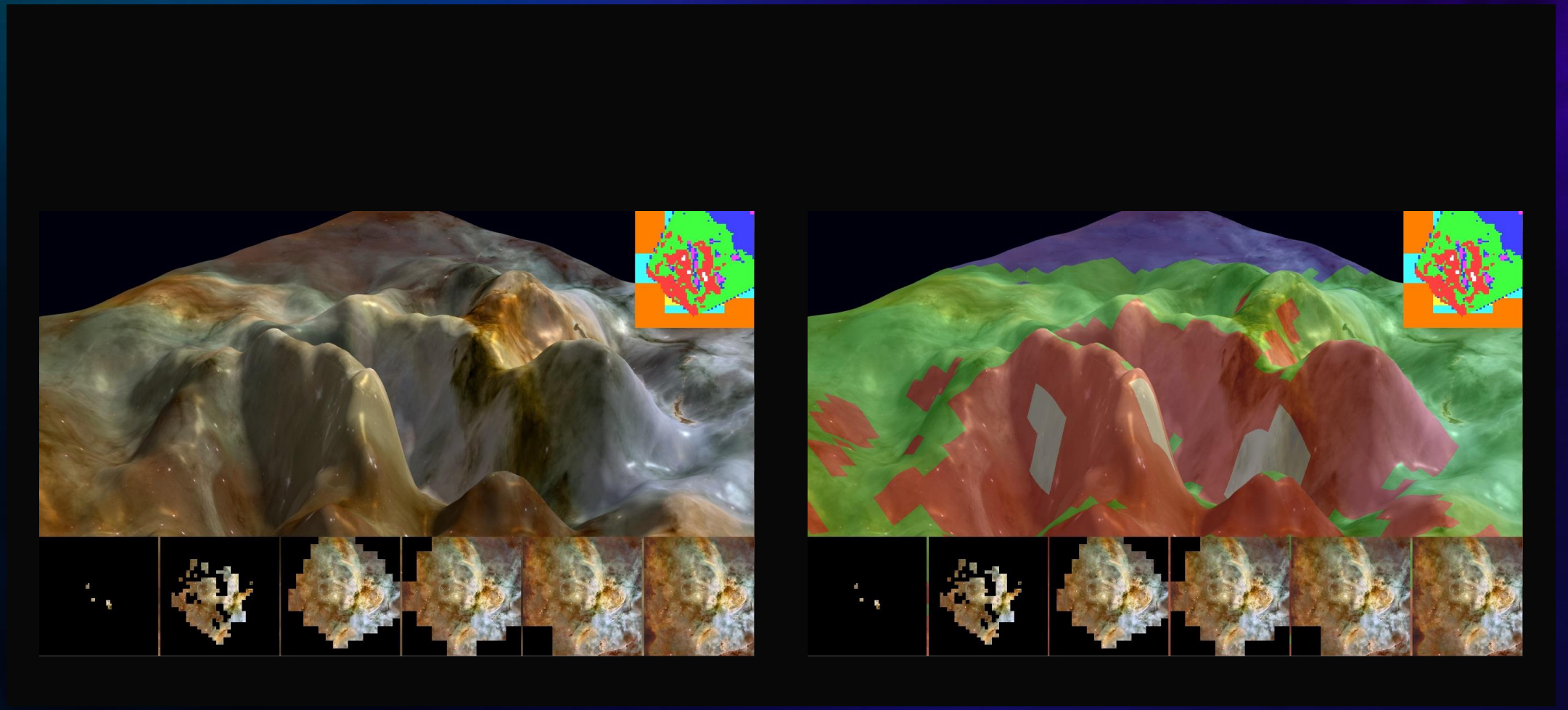
- **Set Sampler Feedback region size to reserved resource tile size**
  - e.g. 256 x 256 for BC7
- **Sampler Feedback min mip map tells you which tiles to load**
  - e.g. all tiles at and below mip 3 in a particular region
- **Sampler Feedback Resource is very small:**
  4KB for 16kx16k BC7

Color = mip level. For some resources, not all tiles of each mip layer are are loaded.



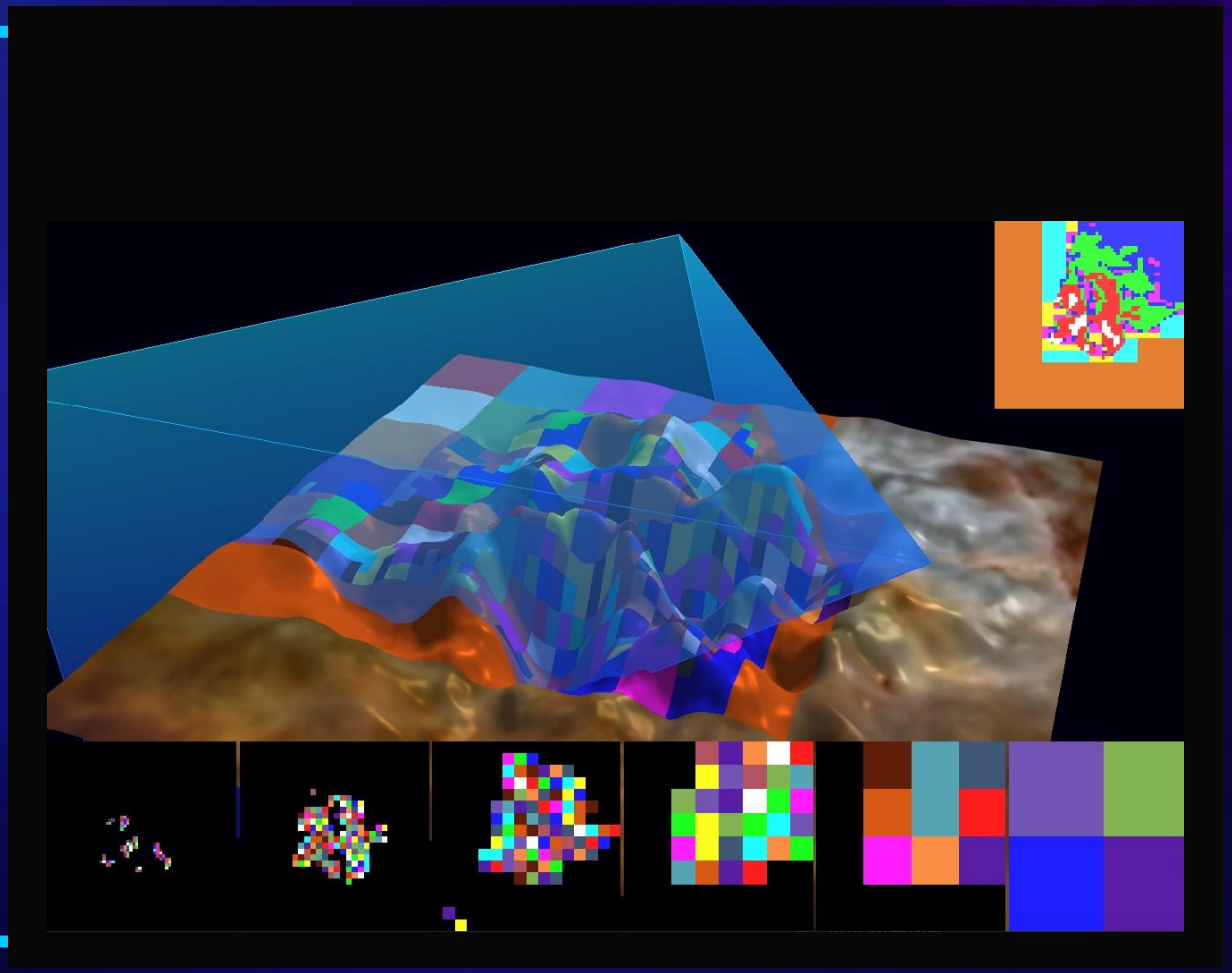
Sampler Feedback Avoids Artifacts

No cracks/seams between tiles at different mip levels



Sampler Feedback Enables Aggressive Memory Management

- Tile resolution drops with distance
- Tiles outside of view can be evicted quickly
- (blurry area is packed mips)



Agenda

- Asset Streaming Opportunity
- D3D12 Sampler Feedback Background
- D3D12 Reserved Resources
- Connecting DirectStorage*
- Results
- Conclusion & Call to Action
- References

File Streaming Background

- Sampler Feedback: what to load
- Reserved Resources: where they are loaded

How do we load the tiles?

Make Your Assets Streaming Friendly

Tile texture assets

- Want: single 64KB contiguous file reads
- DDS Textures: 64 reads per tile! (each row of BCn is 4 high)

Most disks do well with contiguous reads

Sparse reads of 64KB chunks achieve high throughput

File Streaming (for DX) is Hard

Traditionally a lot of bookkeeping

- event handles, upload buffers, copy queue, command lists, command allocators
- may have a dedicated thread to poll event handles & create copy commands

DirectX interaction is complex

- must manage upload resources (e.g one large shared or many small upload buffers)
- minimize time from start of file load to signal of DX fence

Difficult to implement with high performance

- Want: Low Latency, Maximum Bandwidth, Minimal CPU Overhead
- Especially critical for streaming applications – cannot have multi-frame delay

Streaming with DirectStorage* for Windows

DirectStorage = file loading that speaks DirectX

- Can synchronize with familiar DirectX fence objects

Replaced hundreds of lines of file upload code

- fewer kernel transitions, etc.

Faster and lower CPU overhead

- fewer kernel transitions, etc.

Easily load from disk or memory to regions, tiles, or mips

- trivial to upload from tiled asset files

Streaming with DirectStorage* for Windows

DirectStorage = file loading that speaks DirectX

- Can synchronize with familiar D3D12 fence objects

Replaced hundreds of lines of file upload code

Faster and lower CPU overhead

- fewer kernel transitions, etc.

Easily load from SSD or memory to regions, tiles, or mips

DirectStorage replaced hundreds of lines code
plus 1 dedicated CPU thread

> “We have been collaborating closely with Intel on DirectStorage for Windows, and are really excited about new experiences developers will be able to unlock with it”

- Damyan Pepper, Development Lead (DirectStorage for Windows*), Microsoft

Agenda

- Asset Streaming Opportunity
- D3D12 Sampler Feedback Background
- D3D12 Reserved Resources
- Connecting DirectStorage*
- Results
- Conclusion & Call to Action
- References

Stream Many (Very) Large Assets

- **For this scene:**
  - ~50MB/s
    <250MB physical memory

Agenda

- Asset Streaming Opportunity
- D3D12 Sampler Feedback Background
- D3D12 Reserved Resources
- Connecting DirectStorage*
- Results
- Conclusion & Call to Action
- References

Summary / Call to Action

Intel® Iris® Xe Graphics and future Intel dGPUs support Sampler Feedback

Intel systems will support DirectStorage* when available

Begin developing with Sampler Feedback today!

intel

References

- Microsoft® Sampler Feedback Specification
- DirectStorage is Coming to PC
- Sample Source Code
- Hubble Images

Legal Notices and Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates.

Your costs and results may vary.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.


intel