NVIDIA'S VR INSIGHTS

Ingo Esser, Robert Menzel, 10/10/2018
Motivation

Multi-View Rendering (new in Turing)

Variable Rate Shading (new in Turing)

OpenGL multicast 2 (new extension version)
MOTIVATION
GRAPHICS PIPELINE

VR Workloads
GRAPHICS PIPELINE

VR Workloads

249M Pix/s
N vertices
30 Hz
(4K display)

792M Pix/s
2N vertices
90 Hz
(Vive Pro /w oversampling)
GRAPHICS PIPELINE

VR Workloads

249M Pix/s
N vertices
30 Hz
(4K display)

792M Pix/s
2N vertices
90 Hz
(Vive Pro /w oversampling)

- Application
- Driver
- Geometric Pipeline
- Rasterization Fragment Shader

6x
6x
6x
3x
NVIDIA VRWORKS
Comprehensive SDK for VR Developers

<table>
<thead>
<tr>
<th>GRAPHICS</th>
<th>HEADSET</th>
<th>PROFESSIONAL</th>
<th>SIMULATION</th>
<th>VIDEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE PASS STEREO</td>
<td>CONTEXT PRIORITY</td>
<td>WARP &amp; BLEND</td>
<td>VRWORKS AUDIO</td>
<td>VRWORKS 360 VIDEO</td>
</tr>
<tr>
<td>MULTI-VIEW RENDERING</td>
<td>DIRECT MODE</td>
<td>SYNCHRONIZATION</td>
<td>PHYSX</td>
<td>GPU DIRECT FOR VIDEO</td>
</tr>
<tr>
<td>MULTURES SHADING</td>
<td>FRONT BUFFER RENDERING</td>
<td>GPU AFFINITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARIABLE RATE SHADING</td>
<td>VR SLI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENS MATCHED SHADING</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MULTI-VIEW RENDERING
TWO PASS STEREO RENDERING

2 Full Geometry Passes

Left Eye (Pass 1)

Right Eye (Pass 2)
TWO PASS RENDERING
Mono to Stereo

Workload in all steps of the pipeline double.

Getting CPU bound fast, especially in CAD!
SINGLE-PASS-STEREO
1 Pass on Pascal

Left Eye

Right Eye
SINGLE-PASS-STEREO
Mono to Stereo

Cut CPU time in half
Cut VTG processing (nearly) in half
No change in raster & shading

DX: NVAPI

Vulkan: VK_KHR_multiview & VK_NVX_multiview_per_view_attributes

OpenGL: GL_NV_stereo_view_rendering
SINGLE-PASS-STEROO

Limitations

Two views only
Assume co-planar screens: Only change X-axis
MULTI-VIEW RENDERING
Next Generation Single-Pass-Stereo

Left Eye
Right Eye
**MULTI-VIEW RENDERING**

**Turing**

Up to 4 arbitrary views in hardware.

Up to 32 arbitrary views in software.

Vulkan: VK_KHR_multiview

OpenGL: GL_OVR_multiview & GL_OVR_multiview2
MULTI-VIEW RENDERING

Pre-Turing

Up to 32 arbitrary views in software.
Still significant reduction in CPU overhead.
Reduces number of code paths.

Vulkan: VK_KHR_multiview
OpenGL: GL_OVR_multiview & GL_OVR_multiview2
MULTI-VIEW RENDERING

Non-VR Use-cases

One pass Cascaded Shadow Maps
MULTI-VIEW RENDERING

Example: OpenGL

Render to multiple layers (just like Single-Pass-Stereo)

Provide data for all views to Vertex Shader

Handle view dependent operations via new built-in gl_ViewID_OVR

Minimize number of varyings dependent on gl_ViewID_OVR!
MULTI-VIEW RENDERING

Example: OpenGL

```cpp
mat4 modelViewProjection = viewProjMatrix[gl_ViewID_OVR] * model;
gl_Position = modelViewProjection * vertex_pos;
```
MULTI-VIEW RENDERING

Example: OpenGL

```cpp
mat4 modelViewProjection = viewProjMatrix[0] * model;
gl_Position = modelViewProjection * vertex_pos;
gl_Position /= gl_Position.w;

if (gl_ViewID_OVR == 1) {
    mat4 modelViewProjection2 = viewProjMatrix[1] * model;
    vec4 pos = modelViewProjection2 * vertex_pos;
    gl_Position.x = pos.x / pos.w; // hint that only X depends on the viewID to mimic SPS
}
```
MULTI-VIEW RENDERING

Recap

Reduces geometric load and CPU overhead

More flexible than SPS

Software fallback for pre-Turing GPUs

Performance boost depends on number of view dependent attributes

DX11: NVAPI / DX12: View Instancing API

Vulkan: VK_KHR_multiview

OpenGL: GL_OVR_multiview & GL_OVR_multiview2
VARIABLE RATE SHADING
VARIABLE RATE SHADING

Rasterization

Pixel
VARIABLE RATE SHADING

Rasterization

Pixel

Sampling position
VARIABLE RATE SHADING

Rasterization

Pixels: 40
Samples rastered: 40
F.Shader invocations*: 40

* (not counting helper threads)
VARIABLE RATE SHADING

Multi Sampling Rasterization

Pixels: 44
Samples rastered: 69
F.Shader invocations: 44
VARIABLE RATE SHADING

Multi Sampling Rasterization

Shading result stored for one sampling position
Shading result stored for two sampling position

Pixels: 44
Samples rastered: 69
F.Shader invocations: 44
VARIABLE RATE SHADING

- Pixel
- Sampling position
- Fragment Shader Invocation
VARIABLE RATE SHADING
VARIABLE RATE SHADING

Shading result stored for one pixel

Shading result stored for two pixels

Shading result stored for four pixels
VARIABLE RATE SHADING

Pixels: 40
Samples rastered: 40
F.Shader invocations: 14
VARIABLE RATE SHADING

1x1 Shading Rate

Pixels: 477
Samples rastered: 477
F.Shader invocations: 477
VARIABLE RATE SHADING

2x2 Shading Rate
Pixels: 477
Samples rastered: 477
F.Shader invocations: 128
VARIABLE RATE SHADING

4x4 Shading Rate
Pixels: 477
Samples rastered: 477
F.Shader invocations: 42
VARIABLE RATE SHADING
VARIABLE RATE SHADING

1x1 Shading Rate

4x4 Shading Rate

2x2 Shading Rate

1x1 Shading Rate

2x2 Shading Rate

2x2 Shading Rate
VARIABLE RATE SHADING

Shading Rate Lookup

1x1 Shading Rate

4x4 Shading Rate

2x2 Shading Rate

1x1 Shading Rate

2x2 Shading Rate

2x2 Shading Rate
VARIABLE RATE SHADING

Shading Rate Lookup

Framebuffer

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Palette

1x1 Shading Rate
4x4 Shading Rate
2x2 Shading Rate
2x4 Shading Rate
VARIABLE RATE SHADING

Shading Rate Lookup

Framebuffer

Shading Rate Image (8 bit integer)

Palette (16 entries)
VARIABLE RATE SHADING

Shading Modes

GL_SHADE_NO_PIXELS_NV
GL_SHADE_1X1_PIXELS_NV
GL_SHADE_1X2_PIXELS_NV
GL_SHADE_2X1_PIXELS_NV
GL_SHADE_2X2_PIXELS_NV
GL_SHADE_2X4_PIXELS_NV
GL_SHADE_4X2_PIXELS_NV
GL_SHADE_4X4_PIXELS_NV
VARIABLE RATE SHADING

Shading Modes: Multi-Sample Framebuffers

GL_SHADE_2_SAMPLES_NV  GL_SHADE_4_SAMPLES_NV  GL_SHADE_8_SAMPLES_NV  GL_SHADE_16_SAMPLES_NV
VARIABLE RATE SHADING

Foveated Rendering
VARIABLE RATE SHADING
Foveated Rendering

Foveation pattern in Shading Rate Image
For layered rendering (e.g. Multi-View Rendering): Use texture array for SRI
VARIABLE RATE SHADING

Per Object Shading Rate
VARIABLE RATE SHADING

Per Object Shading Rate

Two Viewports:
Both span full framebuffer
Each has own Shading Rate Palette
Select matching viewport in VTG Shader
VARIABLE RATE SHADING

Layered Framebuffer

Shading Rate Lookup

Shading Rate Image Array (8 bit integer)

Per Viewport Palette (16 entries)
VARIABLE RATE SHADING

Varying Extrapolation
VARIABLE RATE SHADING

Varying Extrapolation
VARIABLE RATE SHADING
Varying Extrapolation
VARIABLE RATE SHADING

Varying Extrapolation

Varyings are interpolated in the Pixel center
VARIABLE RATE SHADING

Varying Extrapolation

which means extrapolation for some (but just a small amount)
VARIABLE RATE SHADING

Varying Extrapolation

unless they are defined as centroid
VARIABLE RATE SHADING

Varying Extrapolation
VARIABLE RATE SHADING
Varying Extrapolation

Varyings are interpolated in the coarse pixel center

Significantly more extrapolation compared to MSAA:
Use centroid to avoid artifacts!
VARIABLE RATE SHADING
Various Use-Cases

Foveated Rendering
Content Adaptive Shading
Motion Adaptive Shading
VARIABLE RATE SHADING

Recap

Reduces Fragment load

Allows to tailor workload to needs

Fine-grained control over shading rate

Performance boost depends on shading complexity and triangle size

Vulkan: VK_NV_shading_rate_image

OpenGL: GL_NV_shading_rate_image

DX11: NVAPI
VR SLI
(GL MULTICAST 2)
VR SLI
Crash course

Left view data
Geometry
Materials
Right view data

L R

R

R
VR SLI
Scaling 1 vs 2 GPUs

App Left

GPU L

App Right

GPU R

App Both

GPU L

GPU R

Copy

Time: GPU L + GPU R

Time: GPU + Copy

\[ Scaling = \frac{2 \times GPU}{GPU + Copy} \]
**VR SLI**

Scaling determined by workload and copy time

\[
Scaling = \frac{2 \times \text{GPU}}{\text{GPU} + \text{Copy}}
\]

- Typical render resolution for Vive
  1512 x 1680 (per eye)

- Copy time over PCIe (@6GB/s)
  1.5ms

- Max scaling with 11ms frame time

\[
\frac{2 \times 9.5\text{ms}}{9.5\text{ms} + 1.5\text{ms}} = 1.72
\]
**VR SLI**

Higher resolutions limit scalability

*Scaling* = \( \frac{2 \times GPU}{GPU + Copy} \)

Vive Pro render resolution

2016 x 2240

Copy time over PCIe (@6GB/s)

2.8ms

Max scaling with 11ms frame time

\[
\frac{2 \times 8.2ms}{8.2ms + 2.8ms} = 1.49
\]
VR SLI

Improve scaling using NVLink

Copy times can hurt scaling with higher resolutions

NVLink on dual Quadro GP100: 4x faster than PCIe 3.0

Copy time for Vive Pro (2016 x 2240): 0.7ms

Max scaling with 11ms frame time

\[
\frac{2 \times 10.3ms}{10.3ms + 0.7ms} = 1.87
\]

NVLink is used automatically if present

NVLink speed measured with 2 bridges, copy via OpenGL multicast, single frame of HTC Vive, on HP Z840 workstation
VR SLI
Single-Input HMD

StarVR HMD image courtesy of Starbreeze
VR SLI
Dual-Input HMD

StarVR HMD image courtesy of Starbreeze
OPENGL VR SLI: MULTICAST 2

Feedback on Multicast led to new functionality

Command & data broadcast
BufferSubData to specific GPU
CopyImageSubData & CopyBufferSubData
GPU-GPU Framebuffer Blit
Global barrier & directed sync functions
GPU Masks
Per-GPU sample locations
Per-GPU queries

Dynamic Multicast toggle (WGL_NV_multigpu_context)
GPU_ID built-in in GLSL shader
Per-GPU viewports & scissors
Texture & Buffer upload mask
Asynchronous copies
New extension WGL_NV_multigpu_context: Request SLI mode per context

No need to restart application

Possible to share resources between contexts
New extension WGL_NV_multigpu_context: Request SLI mode per context

No need to restart application

Possible to share resources between contexts

On toggle:

Clean up per-GPU resources

Keep scene data

Alternate Frame Rendering (AFR)
MULTICAST 2

GPU ID built-in: gl_DeviceIndex

Multicast v1 required per-GPU uploads

Larger code changes in some renderers

Add shader built-in: gl_DeviceIndex

Upload all views to all GPUs

Use per-GPU data in shaders

Renderer can remain unchanged

Just modify shaders instead
MULTICAST 2
Per-GPU Viewports & Scissors

Add new function to set viewports and scissors per GPU

```c
glMulticastViewportArrayvNVX( ... );
```

```c
glMulticastScissorArrayvNVX( ... );
```

Per-GPU Lens Matched Shading
MULTICAST 2
Per-GPU Viewports & Scissors

Add new function to set viewports and scissors per GPU

```c
glMulticastViewportArrayvNVX( ... );

// Per-GPU Lens Matched Shading

```

Per-GPU Multi Resolution Shading
MULTICAST 2

Per-GPU Viewports & Scissors

Add new function to set viewports and scissors per GPU

```c
glMulticastViewportArrayvNVX( ... );

glMulticastScissorArrayvNVX( ... );
```

Per-GPU Lens Matched Shading

Per-GPU Multi Resolution Shading

Easily set up Split Frame Rendering (SFR)
Multicast provides per-GPU buffer uploads

Asymmetrical functionality wrt texture upload functions

Add new mask function to modify texture & buffer uploads

\[ \text{glUploadGpuMaskNVX( GLbitfield mask );} \]

Useful for simpler per-GPU texture streaming

Conserve PCIe bandwidth
MULTICAST 2
Asynchronous Copies

Multicast copies stall source GPU while copy takes place

Easy to use because of implicit synchronization

New copy functions do not stall, but also need more synchronization

```cpp
glAsyncCopyBufferSubDataNVX( ... );
glAsyncCopyImageSubDataNVX( ... );
```

Copy while both GPUs can continue rendering

Allows for more complex rendering algorithms
VR SLI

Recap

VR SLI covers a wide variety of workloads

Almost perfect load balancing between left/right eye and two GPUs

Copy overhead and view independent workloads limit scaling

NVLink can help improve scaling

OpenGL: `GL_NV_gpu_multicast` / `GL_NV_gpu_multicast2`

Vulkan: `VK_KHR_device_group` (core in VK 1.1)

DX11: NVAPI
TRY IT OUT!
..and more information

NVIDIA VRWorks SDK provides OpenGL, Direct3D & Vulkan samples

developer.nvidia.com/vrworks

More detail in our previous GTC talks:

2018 - S8695 - NVIDIA VR Update
2017 - S7191 - Vulkan Technology Update
2016 - S6338 - VR Multi GPU Acceleration Featuring Autodesk VRED
2015 - S5668 - VR Direct: How NVIDIA Technology Is Improving The VR Experience