Rendering Faster and Better with VRWorks

Cem Cebenoyan
Director of Engineering, NVIDIA
Talk Overview

VRWorks Features
  Context Priority, Direct Mode, VR SLI
  Multi-Res Shading, Lens Matched Shading, Single Pass Stereo

UnrealEngine 4 Integration
How is VR rendering different?
How is VR rendering different?

High framerate, low latency

High FPS, low latency

Stereo Rendering

Lens Distortion
NVIDIA VRWorks

COMPREHENSIVE SDK FOR VR DEVELOPERS
Context Priority and Direct Mode
Context priority for asynchronous time warp
Reduces latency from head rotation
Pascal Graphics Preemption

Improves responsiveness
Timewarp with Maxwell

- GPU
- BUFFER
- TIMEWARP
- Flip
- Scanout
- Flash backlight
- Latency
- Conservative Preemption Request
Timewarp with Pascal

Pascal Preemption Request

GPU

TIMEWARP

Flip

Scanout

Flash backlight

Latency
Direct Mode
Plug-and-play Compatibility Via Direct Interface with the VR Headset
VR SLI
VR SLI

Two eyes...two GPUs!
“Normal” SLI

GPUs render alternate frames

CPU  |  N  |  N+1
---   |----|----
GPU 0 |  N  |
GPU 1 |  N+1 |
Display |  N  |  N+1

Latency
VR SLI

Each GPU renders one eye—lower latency

CPU

GPU 0

GPU 1

Display

Latency

N

N+1

N_L

N+1_L

N_R

N+1_R

N

N+1
VR SLI

GPU affinity masking: full control

Left eye rendering

Shadow maps, GPU physics, etc.

Right eye rendering

\[
\text{UINT SetGPUMask(in UINT GPUMask);}
\]
VR SLI

Broadcasting reduces CPU overhead

Render scene once
VR SLI

Per-GPU constant buffers, viewports, scissors

```
NvAPI_Status VSSetConstantBuffers(
    [in] ID3D11DeviceContext *pContext,
    [in] UINT GPUMask,
    [in] UINT StartSlot,
    [in] UINT NumBuffers,
);
```

```
void glLGPUNamedBufferSubDataNVX ( 
    GLbitfield gpuMask, 
    GLuint buffer, 
    GLintptr offset, 
    GLsizeiptr size, 
    const GLvoid *data );
```
VR SLI

Cross-GPU data transfer via PCI Express

```c
void glGPUCopyImageSubDataNVX ( 
    GLuint sourceGpu, 
    GLbitfield destinationGpuMask, 
    GLuint srcName, 
    GLuint srcTarget, 
    GLint srcLevel, 
    GLint srcX, 
    GLint srcY, 
    GLint srcZ, 
    GLuint dstName, 
    GLuint dstTarget, 
    GLint dstLevel, 
    GLint dstX, 
    GLint dstY, 
    GLint dstZ, 
    GLsizei width, 
    GLsizei height, 
    GLsizei depth);

NvAPI_Status CopySubresourceRegion( 
    [in] ID3D11DeviceContext *pContext, 
    [in] ID3D11Resource *pDstResource, 
    [in] UINT DstSubresource, 
    [in] UINT DstGPUIndex, 
    [in] UINT DstX, 
    [in] UINT DstY, 
    [in] UINT DstZ, 
    [in] ID3D11Resource *pSrcResource, 
    [in] UINT SrcSubresource, 
    [in] UINT SrcGPUIndex, 
    [in] const D3D11_BOX *pSrcBox, 
    [in, optional] UINT ExtendedFlags = 0 );
```
Multi-Resolution Shading
VR headset optics

Distortion and counter-distortion
VR headset optics

Distortion and counter-distortion
Distorted rendering

Render normally, then resample
Distorted rendering

Over-rendering the outskirts
Multi-resolution shading
Subdivide the image, and shrink the outskirts
Multi-resolution shading

Fast viewport broadcast on NVIDIA Maxwell GPUs

Geometry Pipeline → Viewport 1 → Viewport 2 → ... → Viewport N
Lens Matched Shading

RENDERS TO A LENS CORRECTED SURFACE
Lens Matched Shading

RENDERS TO A LENS CORRECTED SURFACE
TRADITIONAL STEREO RENDERING

REQUIRES 2 GEOMETRY PASSES

Left Eye (Pass 1)  Right Eye (Pass 2)
Traditional Stereo Rendering

Double the full pipeline rendering cost

Many engines now adding support for Instanced Stereo Rendering
Improves on CPU side cost of stereo submission
Doesn’t help with GPU cost
SINGLE PASS STEREO

RENDERS LEFT & RIGHT EYE IN ONE GEOMETRY PASS
Demo!
Multi-Res in UnrealEngine 4
Multi-resolution shading

Unreal Engine integration

We’ve integrated multi-res in UE 4.10, 4.11, and 4.12

Currently limited support for post effects with multi-res

Available on GitHub

https://github.com/NvPhysX/UnrealEngine/tree/MultiRes-4.10

Pascal features coming soon
UE4 integration 4.10

https://github.com/NvPhysX/UnrealEngine/tree/MultiRes-4.10

Postprocessing passes supported:

- Bloom & lens flare
- Tonemapping
- Temporal AA
- Reflection environment (not screen-space reflection)
- Height fog
- Decals
- Distortion/refraction
UE4 integration 4.11 / 4.12


Adds postprocessing passes supported:

- SSAO
- Screenspace Reflections

Supports Instanced Stereo Rendering

Major refactoring
Multi-resolution shading

Performance

UE4 Infiltrator demo: +30% to +40% FPS
@ approximate VR render res

Everest VR seeing up to +40%

Best when pixel-bound
We’ve seen ~50% perf boosts
UE4 integration

Main points

Direct engine integration of Multi-res shading

RHI support for multiple viewports & scissors

FastGS plumbing

Post-processing shaders
Rendering passes

Geometry passes: add FastGS; render with 9 viewports & scissors
  Base pass, decals, deferred lights, depth prepass, distortion, shadow projection, translucency, velocity

Image-space passes: remap linear/multi-res UVs as needed
  Decals, deferred lights, postprocessing, shadow projection, tiled deferred lights
Image-space passes

Two common cases

Reconstruct 3D position from 2D+depth
   Map 2D position to linear before reconstructing

Large filter kernels
   Map sample point to linear, apply offset, map back to multi-res
UV remapping
Shadow projection

Example of reconstructing 3D position

Before:

```cpp
float2 ScreenUV = float2(SVPos.xy * Frame.BufferSizeAndInvSize.zw);
float SceneW = CalcSceneDepth(ScreenUV);
float2 ScreenPosition = (ScreenUV.xy - Frame.ScreenPositionScaleBias.wz) / Frame.ScreenPositionScaleBias.xy;
float4 ShadowPosition = mul(float4(ScreenPosition.xy * SceneW,SceneW,1), ScreenToShadowMatrix);
```
Shadow projection

Example of reconstructing 3D position

After:

```c
float2 ScreenUV = float2( SVPos.xy * Frame.BufferSizeAndInvSize.zw );
float SceneW = CalcSceneDepth( ScreenUV );

// Remap for warped viewports, no-op for regular viewports
float2 LinearScreenUV = MultiResMapRenderTargetMultiResToLinear(ScreenUV);

float2 ScreenPosition = (LinearScreenUV.xy - View.ScreenPositionScaleBias.wz) / View.ScreenPositionScaleBias.xy;
float4 ShadowPosition = mul(float4(ScreenPosition.xy * SceneW,SceneW,1), ScreenToShadowMatrix);
```
Shadow projection

Example of reconstructing 3D position

After:

```c
float2 ScreenUV = float2( SVPos.xy * Frame.BufferSizeAndInvSize.zw );
float SceneW = CalcSceneDepth( ScreenUV );

// Remap for warped viewports, no-op for regular viewports
float2 LinearScreenUV = MultiResMapRenderTargetMultiResToLinear(ScreenUV);

float2 ScreenPosition = (LinearScreenUV.xy - View.ScreenPositionScaleBias.wz) / View.ScreenPositionScaleBias.xy;
float4 ShadowPosition = mul(float4(ScreenPosition.xy * SceneW,SceneW,1), ScreenToShadowMatrix);
```
Shadow projection

Example of reconstructing 3D position

After:

```c
float2 ScreenUV = float2( SVPos.xy * Frame.BufferSizeAndInvSize.zw );
float SceneW = CalcSceneDepth( ScreenUV );

// Remap for warped viewports, no-op for regular viewports
float2 LinearScreenUV = MultiResMapRenderTargetMultiResToLinear(ScreenUV);

float2 ScreenPosition = (LinearScreenUV.xy - View.ScreenPositionScaleBias.wz) / View.ScreenPositionScaleBias.xy;
float4 ShadowPosition = mul(float4(ScreenPosition.xy * SceneW,SceneW,1), ScreenToShadowMatrix);
```
Demo Time

Demo UE4!
Questions?
cem@nvidia.com

http://developer.nvidia.com/vrworks