The development & optimization of the DX12 version of *King Of Wushu*

Lv Wenwei
Engine Technology Expert
Snail Game

Yang Xueqing
Devtech Engineer
NVIDIA
King Of Wushu

• First 3D martial arts MOBA

• With top real-time graphics effects

• First DirectX12 multi-platform online game in China

• Latest NVIDIA GameWorks Integration

• Xbox One & PlayStation 4 version have been released in China, PC version is coming soon in this year
DX12 Porting
From DirectX11 to DirectX12

• 2 senior graphics engineers, 6 weeks

• Override DX11 interfaces with DX12 APIs

• Manage rendering states with hash map, storing and searching dynamically in real-time

• Assign an unique ID to each resource and state to generate Hash value, if the resource is released, the ID can be reassigned

• Every resource has its bit width in the Hash value, and has ID upper limit

• SDKLayer(Debug Runtime) helps you to find potential rendering issues as early as possible
PipelineStateObject Management

- State settings based on DX11 are cached and delayed until the draw call is executed
  1. RasterizerState
  2. BlendState
  3. DepthStencilState
  4. InputLayout
  5. Shader

- Speedup the generation of PSOs by getting the cached PSO via GetCachedBlob
Sampler Management

• Sampler management

1. One sampler heap can store 2048 Samplers at most

2. Group the 16 samplers of each shader and generate hash value for each group, so there are at most 128 sampler groups in one frame

3. Set the same sampler to the fixed slot, e.g. NormalMap, ShadowMap

4. Have to switch heaps if you want more sampler groups
View Management

• Shader Resource View management
  1. One view heap can store 1M view descriptors at most
  2. With the same management to the samplers, max 16 SRVs are used in one draw call

• Constant Buffer View management
  1. Instead of descriptor table, we use root descriptor. Because constant buffers are dynamic, their GPU address changes frequently
  2. Static CB can be managed with descriptor table, but the slots should be fixed
  3. 256 bytes align is required for the constant buffers used in each shader stage
  4. Call map before set to avoid iteratively accessing
Command List

• API calling in Command List is not thread-safe
• The submit in the Command Queue is thread-safe
• Generate multiple command lists to avoid rendering stall
• Sync the command lists with fences, using the frame ID as the expected value
• Driver and run-time don’t provide extra threads for building and committing render commands asynchronously
• Store the D3D11 commands in the command buffer, send it to a work thread and execute it at the end of each frame. you need to reinterpret D3D11 commands to D3D12 commands in the work thread
Summary

- Developers have more options with the open memory management mode
- Flexible resource binding reduces the requirement of bandwidth
- Powerful and low overhead command submit pipeline is helpful to customized parallelized rendering
Integration of NVIDIA GameWorks & GPU New Features
TXAA Integration

- Reuse the graphics pipeline of CE3 MSAA mode

- Need velocity map in resolve pass due to temporal AA
  - Use the velocity map generated by motion blur
  - Need to decode the packed data to float16

- Record the TXAA result of previous frame

- Set programmable sample locations via NVAPI
HBAO+ Integration

• Need to transform the view space normals into world normals

• Convert the depths to view space

• Be aware of the texture format in MSAA mode
Fast GS Integration

- Render the shadow maps in one pass
- Combine the visible objects lists of each LOD
- Use the coarsest view frustum that contains all the view frustums of each LOD
- Store shadow maps in a texture array
  - Rendered as a render target array
  - Render LOD of shadow maps by setting different viewports
  - Render the render target array with fast GS
Summary

• Besides the GameWorks & features above, we also have integrated HairWorks & Clothing
• Saved lots of development cost by using GameWorks
• GameWorks makes game more competitive in the market
• NVIDIA provides reliable technical supports
• Take full advantage of the newest GPU features
  • Vendors fully understand their products
NVIDIA GameWorks and GPU new features in *King Of Wushu*
Agenda

- **GameWorks - VisualFX**
  - PostWorks : FXAA3.0
  - ShadowWorks : HBAO+
  - HairWorks

- **GameWorks – PhysX**
  - Clothing

- **Maxwell features**
  - Multi-Projection Acceleration
    - Fast Geometry Shader (Fast GS)
    - Fast Viewport Multi-casting
TXAA 3.0

No AA  8x MSAA  TXAA
What is TXAA

• Temporal AA mixed with MSAA

• Replaces MSAA Resolve

• Provides higher quality resolve filter
  – Better than the default MSAA box filter
What’s new TXAA 3.0

• More user controls
  – Control of the reconstruction filter that’s used
  – Per-pixel control of AA application

• Higher AA quality & faster perf
  – Maxwell feature: Programmable sample locations
Programmable sample locations

- Sample locations fully programmable
Programmable sample locations

- High-level MSAA quality at low-level MSAA cost
Programmable sample locations

- Interleaved sample positions
  - 16x sample locations can be tiled to a set of pixels
  - Higher AA quality

<table>
<thead>
<tr>
<th>AA</th>
<th>Sample Locations</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>16xAA</td>
<td>1x1 pixels</td>
<td>16</td>
</tr>
<tr>
<td>8xAA</td>
<td>1x2 pixels</td>
<td>16</td>
</tr>
<tr>
<td>4xAA</td>
<td>2x2 pixels</td>
<td>16</td>
</tr>
<tr>
<td>2xAA</td>
<td>4x2 pixels</td>
<td>16</td>
</tr>
<tr>
<td>1xAA</td>
<td>4x4 pixels</td>
<td>16</td>
</tr>
</tbody>
</table>
AA Off
4xMSAA
TXAA 3.0
(4xTXAA+PSL)
HBAO+
HBAO+ Design Goals

• Look better than the HBAO algorithm
  – HBAO suffers from over-occlusion behind thin objects

• Better efficiency than the HBAO algorithm
  – Minimize the math ops / TEX sample
  – Interleaved rendering, have the highest possible texture cache hit rate

• Full-res SSAO, not half-res
  – Rendering SSAO in half-res tends to cause bad flickering on thin geometry (e.g. alpha-tested surfaces)

• Easy to integrate
HBAO

Over-occlusion & flickering
HBAO+ No visual issues
Fixed Sampling Pattern

For each sample, adjacent pixels fetching adjacent texels

⇒ Good spatial locality 😊
Random Sampling Pattern

For each sample, adjacent pixels fetching far-apart texels

⇒ Poor spatial locality 😞
Jittered Sampling Pattern

For each sample, adjacent pixels fetching sectored texels

→ Better spatial locality

... but as kernel size increases, sector size increases too 😅
Interleaved Rendering

Render each sampling pattern **separately**, using **downsampled** input textures
Step 1: Deinterleave Input

1 Draw call with 4xMRTs

Full-Resolution
Input Texture

Width = W
Height = H

Half-Resolution
2D Texture Array

Width = iDivUp(W, 2)
Height = iDivUp(H, 2)
Step 2: Jitter-Free Sampling

Input: Texture Array A (slices 0, 1, 2, 3)

Output: Texture Array B (slices 0, 1, 2, 3)
Step2 : Jitter-Free Sampling

1. Constant jitter value per draw call
   ➔ better per-sample locality

2. Low-res input texture per draw call
   ➔ less memory bandwidth needed
Step 3: Interleave Results

1 Draw call
With 1 Tex2DArray fetch per pixel
4x4 Interleaving

4x4 jitter textures are commonly used for jittering large sparse filters

Can use a 4x4 interleaving pipeline

1. **Deinterleaving:** 2 Draw calls with 8xMRTs
2. **Sampling:** 16 Draw calls
3. **Interleaving:** 1 Draw call
Multi-Projection Acceleration
Fast GS vs Regular GS

- Fast GS is a special kind of geometry shader
- Fast GS cannot "create" new primitives
- Fast GS saves the cost of the geometry expansion
Fast Viewport Multi-casting

Geometry Pipeline

Viewport Mask

Viewport 1

Viewport 2

Viewport N

...
Use Cases

• Where we only use the geometry shader stage to set per-primitive attributes, instead of changing the primitive topology itself.

• Cube-Map rendering

• Voxelization

• Multi-resolution rendering (for VR)

• Cascaded Shadow Maps
Implementing CSM with Fast GS

- Generate all of the shadow maps in a single rendering pass, save CPU overhead
- Render the shadow maps with a coarsest view frustum which contains all the frustums of each LOD, setting different view ports for different LOD

LOD=2

LOD=1

LOD=0
Implementing CSM with Fast GS

- Do view frustum culling in the GS
  - Cast the primitives to desired viewports by setting bits in the viewport mask
  - Primitive is killed if viewport mask equals 0
Implementing CSM with Fast GS
HairWorks & Clothing
Summary
Summary

PhysX
- PARTICLES
- DESTRUCTION
- CLOTHING
- FLEX

VisualFX
- FACEWORKS
- HAIRWORKS
- TURBULENCE
- FLAMEWORKS

OptiX
- INTERACTIVE RAY TRACING
- AMBIENT OCCLUSION
- PROCEDURAL SURFACES
- LIGHT BAKING

Samples
- SOFT SHADOWS
- PARTICLE SHADOWS
- MOTION BLUR
- TERRAIN TESSELLATION
Summary

• NVIDIA GameWorks
  – SDKs of efficient high-quality graphics & physics effects
  – Samples, documentation & tutorials
  – Developer tools
  – Making game developing easier
    https://developer.nvidia.com/gameworks

• New GPU hardware features
  – More optimization approaches available
  – More new algorithms
Thank you!
Questions?

youngy@nvidia.com
CGDC 2015 中国游戏开发者大会
分享智慧 LET US SHARE