Advanced Ambient Occlusion, Shadow and Anti-Aliasing in NVIDIA GameWorks

Jerry Cao - DevTech Engineer
jecao@nvidia.com
Overview

- VXAO
- HRTS
- TXAA 3.0
- ANSEL
- HDR Display
Voxel Ambient Occlusion

- VXGI is NVIDIA’s new real-time global illumination solution
  - Works by voxelizing geometry on every frame
  - Produces approximate but realistic looking diffuse and specular GI
  - Still too resource intensive to use in mainstream games

- VXAO is a special mode of operation of VXGI
  - Throw away all the lighting information, keep only occlusion
  - Works much faster
  - Integration is significantly simpler
Screen-Space AO Issues

- View dependence
  - Lack of occlusion behind foreground objects
  - Errors and unstable results near screen edges

- Locality
  - Only a small volume around a surface contributes to AO

- Blurriness
  - Computing a complete solution for every pixel would be too expensive
Why VXAO is Better Than SSAO

- It has none of the aforementioned issues
- World-space solution
  - An occluder can be behind something else
  - Even behind the viewer (which means less aggressive culling is required)
- Uses Voxel Cone Tracing approach
  - Occluders can be far from the surface under consideration
  - Their information will be sampled from a very coarse voxel LOD
  - Using a large AO distance has very little effect on performance
Overview of voxel cone tracing

- Transform the scene into voxels that encode opacity
  - Then downsample the opacity map
  - Opacity gives us information to compute occlusion

- Gather light by tracing cones through the opacity
Voxelization

- The process of converting a mesh into a voxel representation
- Treat it as 3D space rasterization

A binary voxel representation of an object with color information
In a real demo

- Voxelization

VXAO Result
Example: HBAO+ Channel
Example: VXAO Channel
Image Quality Differences #1

- Ground under the tank
- Bottom part of the tracks
- Blurriness

HBAO+  VXAO
Image Quality Differences #2

HBAO+  
VXAO  

HBAO+  
VXAO
HBAO+ with Color
VXAO with Color
Handling Dynamic Scenes

- Voxel representation is very expensive to construct or update?
  - Wrong!
  - It takes about 1 ms to voxelize a 300K triangle scene on a GTX 980

- Most of voxel data can be preserved between frames
  - Pass a list of AABBs for geometry that has changed
  - Those regions will be cleared and updated, everything else stays
### Performance on GTX 980

<table>
<thead>
<tr>
<th>Pass</th>
<th>Time</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VXAO: Full scene voxelization</td>
<td>1.0 ms</td>
<td>300K triangles</td>
</tr>
<tr>
<td>VXAO: Voxel post-processing</td>
<td>1.4 ms</td>
<td>128³ clip-map, 5 LODs</td>
</tr>
<tr>
<td>VXAO: Cone tracing, Interpolation</td>
<td>1.6 ms</td>
<td>1920x1080</td>
</tr>
<tr>
<td>VXAO overall</td>
<td>4.0 ms</td>
<td></td>
</tr>
<tr>
<td>HBAO+ overall</td>
<td>1.2 ms</td>
<td>1920x1080 Blur radius 4, normal channel supplied</td>
</tr>
</tbody>
</table>

HBAO+ is about 2-4x faster than VXAO on Maxwell GPUs, but loses in quality.
Performance with Partial Updates

For mostly static scenes, voxelization and post-processing cost can be significantly reduced

<table>
<thead>
<tr>
<th>Voxelized geometry percentage</th>
<th>30%</th>
<th>40%</th>
<th>63%</th>
<th>83%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voxelization time, ms</td>
<td>0.23</td>
<td>0.28</td>
<td>0.45</td>
<td>0.63</td>
<td>1.01</td>
</tr>
<tr>
<td>Voxel post-processing time, ms</td>
<td>1.02</td>
<td>1.06</td>
<td>1.08</td>
<td>1.22</td>
<td>1.49</td>
</tr>
<tr>
<td>Total time relative to full voxelization</td>
<td>50%</td>
<td>54%</td>
<td>61%</td>
<td>74%</td>
<td>100%</td>
</tr>
</tbody>
</table>

In 100% case, some large floor and wall triangles add significant contribution. Actual numbers strongly depend on the specific scene (Sponza Atrium in this case)
Hybrid Ray-Traced Shadows

Combine ray tracing with traditional shadow map algorithm

Benefits
- Razor sharp anti-aliased hard shadows
- Super soft shadows
- HRTS will blend between the two result automatically
Issues in Shadow Map

- Acne
- Peter-panning
- Aliasing

Endless tuning to alleviate the above…
Peter-panning
Aliasing
Traditional Spatial Hierarchy

- Typical solutions
  - KD-Tree
  - Bounding Volume Hierarchy
  - Uniform Grid

- Not practical
  - Reconstructing or updating is expensive
  - Tree traversal is inherently slow
Ray Traced Shadow Map Construction

- Prim Buffer - Triangle vertices
- Prim Indices Map - Prim buffer indices of triangles
- Prim Count Map - # of tris per texel
- Raytrace triangles in a later pass
Conservative Rasterization

Hardware Support
- DX12
- NVAPI

Software Implementation
Shadow Map Algorithm
Hybrid Ray Traced Shadow
Hybrid Approach

- Combine ray-traced shadow with conventional soft shadows
- Use an advanced filtering technique such as PCSS
- Use blocker distance to compute a lerp factor
- As blocker distance approaches 0, ray-traced result is prevalent
Lerp Factor Visualization

- \( L = \text{saturate}( BD / WSS \times PHS ) \)
- \( L \): Lerp factor
- \( BD \): Blocker distance (from ray origin)
- \( WSS \): World space scale - chosen based upon model
- \( PHS \): Desired percentage of hard shadow

- \( FS = \text{lerp}( RTS, PCSS, L ) \)
- \( FS \): Final shadow result
- \( RTS \): Ray traced shadow result (0 or 1)
- \( PCSS \): PCSS+ shadow result (0 to 1)
Hybrid Ray Traced + PCSS

SM = 3K x 3K (36 MB)
PM = 1K x 1K x 32 (128 MB)
Performance

- Prims: ~10K
- Shadow Map: 3K x 3K (36 MB)
- Primitive Map: 1K x 1K x 32 (128 MB)
- Primitive Buffer: ~360K
- Shadow Buffer: 1920 x 1080

<table>
<thead>
<tr>
<th></th>
<th>GTX 980</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Map + HW CR</td>
<td>0.4</td>
</tr>
<tr>
<td>Primitive Map + SW CR</td>
<td>0.5</td>
</tr>
<tr>
<td>Ray Trace</td>
<td>0.4</td>
</tr>
<tr>
<td>PCSS</td>
<td>1.3</td>
</tr>
<tr>
<td>PCSS + Ray Trace</td>
<td>1.8</td>
</tr>
</tbody>
</table>
Performance

- Prims: ~65K
- Shadow Map: 3K x 3K (36 MB)
- Primitive Map: 1K x 1K x 64 (256 MB)
- Primitive Buffer: ~2.2 MB
- Shadow Buffer: 1920 x 1080

<table>
<thead>
<tr>
<th></th>
<th>GTX 980</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Map + HW CR</td>
<td>0.5</td>
</tr>
<tr>
<td>Primitive Map + SW CR</td>
<td>0.7</td>
</tr>
<tr>
<td>Ray Trace</td>
<td>0.7</td>
</tr>
<tr>
<td>PCSS</td>
<td>1.3</td>
</tr>
<tr>
<td>PCSS + Ray Trace</td>
<td>2.8</td>
</tr>
</tbody>
</table>
Performance

- Prims: ~240K
- Shadow Map: 3K x 3K (36 MB)
- Primitive Map: 1K x 1K x 64 (256 MB)
- Primitive Buffer: ~8.2 MB
- Shadow Buffer: 1920 x 1080

<table>
<thead>
<tr>
<th></th>
<th>GTX 980</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Map + HW CR</td>
<td>3.4</td>
</tr>
<tr>
<td>Primitive Map + SW CR</td>
<td>4.1</td>
</tr>
<tr>
<td>Ray Trace</td>
<td>1.0</td>
</tr>
<tr>
<td>PCSS</td>
<td>1.3</td>
</tr>
<tr>
<td>PCSS + Ray Trace</td>
<td>3.4</td>
</tr>
</tbody>
</table>
Next Generation TXAA

- Time varying sample locations (PSL)
- Simpler filtering
  - No Gaussian Blur anymore, no over blurred image
  - Faster
- All MSAA mode
  - 8x MSAA mode supported
  - 1x MSAA or NonAA mode supported
TXAA 3.0 core algorithm

TXAA 3.0 = [MSAA] + Temporal AA + MFAA

Setup/Restore PSL Patterns

Base Pass

Motion Vector
Scene Color
Prior Color

TXAA Resolve

Resolved Color

Post Processing

Texture

Rendering Pass
Exponential Sum

The math formula

\[ TXAA_n = \alpha MSAA_n + (1 - \alpha)TXAA_{n-1} \]

A little mathematic background:

\[ MSAA_n = MSAA_{n-k} \]

*True for static image, \( k \) is the number of patterns used*

\[ TXAA_n = \frac{\alpha}{1-(1-\alpha)^k} \sum_{t=0}^{k-1} (1 - \alpha)^t MSAA_{n-t} \]

\[ \lim_{\alpha \to 0} \left( \frac{\alpha}{1-(1-\alpha)^k} \sum_{t=0}^{k-1} (1 - \alpha)^t MSAA_{n-t} \right) = \frac{1}{k} \sum_{t=0}^{k-1} MSAA_{n-t} \]

*Ideal result, at least for static image*

Unfortunately, there are issues:

\[ \alpha \] will never be zero

The formula doesn’t work if \( \alpha \) depends on \( MSAA_n \)

The first equation may be false for dynamic scene
Color Clipping

- To get rid of ghost issue

- An AABB in YCgCo space is constructed by neighbor pixels
  
  \[- \text{Min} = \min( c, n, s, w, e ) \]
  
  \[- \text{Max} = \max( c, n, s, w, e ) \]

- Feedback color is clipped against the AABB

A simple demonstration in 2D space, in TXAA it works similarly in YCgCo space
Programmable Sample Location

- Enable graphics programmer to set up customized sample locations for each sample
- Enhance AA quality at minimal extra overhead

Quality of AA algorithm depends on the number of sample patterns available, 4 or 8 samples patterns work fine
Disable PSL if Necessary

- Shadow map generation pass
- Reflection map
- UI Rendering
- Post-Processing
TXAA 3.0 Integration

- Easy to integrate, only three interfaces:
  - `GFSDK_TXAA_DX11_InitializeContext(&txaaContext, device)`
    - Initialize TXAA 3.0 context, it will fail on devices prior to Kepler
  - `GFSDK_TXAA_DX11_ResolveFromMotionVectors(&resolveParameters, &motion)`
    - Perform TXAA 3.0 resolve pass
  - `GFSDK_TXAA_DX11_ReleaseContext(&txaaContext)`
    - Release TXAA 3.0 context

- Misc:
  - TXAA 3.0 also provides several debugging features
  - Besides the library itself, there is also an utility library providing some helper interfaces
    - Edge detection
    - Motion vector generation from depth and camera information
TXAA Performance

- Comparing with other AA techniques, the overhead in TXAA can almost be neglected

- TXAA only has one draw call, relatively cheap
  - On a GTX 980, it costs less than 1ms
  - Even with 4x MSAA
ANSEL Integration

- Ease of integration
- Overlay UI Controls
- Witness integration is ~40 lines of code
- Witcher 3 integration is ~150 lines of code
Free Camera
Super Resolution
ANSEL POST-PROCESS SHADER

GAME ENGINE → BUFFERS (COLOR BUFFER, Z BUFFER, G BUFFER, ...) → POST PROCESS SHADER → FINAL DISPLAY IMAGE
Loads of Effects!

- Color curves
- Sketch
- Color space transformation
  - Hue shift
  - Desaturation
  - Brightness
- Contrast
- Film grain
- Bloom
- Lens flares
- Anamorphic glare
- Distortion effects
  - Map distortion ("heathaze")
  - Fisheye
  - Explosion
- Color aberration
- Tonemapping schemes
  - Haarm-Peter Duiker
  - Reinhard
  - Hable (Uncharted2)
- Lens dirt
- Lightshafts
- Vignette
- Gamma correction
- Convolution filters
  - Sharpening
  - Edge detection
  - Blur
  - Hipass/lowpass
- FXAA
- Sepia tone
- Halftone
- Deband
- Denoise
360 photos
Stereo capture

STEREO IMAGE

Left Eye  Right Eye
Ansel Coming to games soon
Ansel Coming to games soon
Why HDR Display

- LDR display offers much limited range of gamut

- Without HDR display
  - Highlight looks flatter, can’t be distinguished from diffuse white
  - Highlight compromise on saturation
  - Shadows get crushed toward black

- HDR Display
  - Higher luminance. (1000+nits)
  - Wider gamut.
LDR Displayed Image
CIE Chromaticity Diagram

- It reveals the gamut that is visible to human
- Colors with the highest saturation are at the boundary
- Any interior color can be generated by interpolating different set of colors
sRGB vs BT.2020/Rec.2020

- sRGB
- BT.2020/Rec.2020
Does the old way work?

- Over compressed image
- Lose bright values
- Dim values appears shiny
  - Middle-Grey appears white
Can we drop Tonemapping?

Since we have HDR display now, is tone mapping still necessary?

Yes

- The best displays come nowhere close to the range of luminance in real world
- The first generation HDR compliant will generate 1000 nits of luminance
- BT. 2020 only supports luminance up to 10000 nits

<table>
<thead>
<tr>
<th>Condition</th>
<th>Luminance (cd/m², or nits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun at horizon</td>
<td>600000</td>
</tr>
<tr>
<td>60-watt light bulb</td>
<td>120000</td>
</tr>
<tr>
<td>Clear sky</td>
<td>8000</td>
</tr>
<tr>
<td>Typical office</td>
<td>100-1000</td>
</tr>
<tr>
<td>LDR display</td>
<td>80-100</td>
</tr>
<tr>
<td>Cloudy moonlight</td>
<td>0.25</td>
</tr>
</tbody>
</table>
A new Tone Mapping

- Tone mapping in Academy Color Encoding System (ACES)
  - Sigmoid-style curve
  - Performs on each color channel independently
- We have already implemented the algorithm for you
UI Composition

Used to be done in sRGB space after 3D Rendering

It works the same way on HDR display

Hints:
- Don’t use color with maximum luminance in UI components
- Add a scaling factor
- Be careful with alpha blending, use a simple tonemapper (Reinhard)
Physically Based Shading

- It will generate colors with a large range of luminance
Be Careful with Light Geometry Proxy

- Light geometry proxy may be dimmer than reflected light.
- It used to work fine in LDR monitor since both reach maximum brightness.
LDR vs HDR

- Left: Image rendered on LDR display
- Right: Image rendered on HDR TV and captured with a real camera
THANKS