“...the hardest problem in computer graphics is finding something that hasn’t already been done. That is your challenge.”

Jim Blinn, 1987
The State of (Anti)aliasing in Real Time

Problem Space

Games limited to a low *fixed sample counts* (~1spp) at modest resolutions (< 4K)

**Result:** primary surfaces are *undersampled* and have *unbounded error* when material, geometric, or shading features exist between samples

Aliasing due to undersampling manifests as *jagged edges, spatial noise, and flickering*

**Constraints:** 8x supersampled image quality without ghosting/blurring on a real-time budget
The State of (Anti)aliasing in Real Time

**TAA All Day**

**Supersampling (SSAA):** cost linearly proportional to the number of samples while only improving quality with the square root

**Multisampling:** MSAA, CSAA, SBAA [Salvi and Vidimce 2012], SRAA [Chajdas et al. 2011]

**Aggregation:** DCAA [Wang et al. 2015], AGAA [Crassin et al. 2016]

**Spatial:** MLAA [Reshetov 2009], FXAA [Lottes 2009]

**Time:** SMAA [Jimenez et al. 2012], TAA [Yang et al] [Karis 2014]

*Current Best Practice:* employ many strategies simultaneously, hand tune by artists [Pettineo 2015, Pedersen 2016], rely on TAA for the best !/$ solution
TAA

Blur

No TAA

Deus Ex: Mankind Divided

© Eidos Montreal, Square Enix
Finding A Practical Hybrid Algorithm
Redefining AA

Offline ray tracers use highly adaptive sample counts to resolve aliasing and bound error.

Some real-time work in this area [Hollander et al. 2013], but still must rasterize all geometry.

Previous hybrid ray and raster algorithms were impractical due to HW architectures & APIs.

DXR and RTX enable full interoperability between ray and raster rendering on the GPU.
Finding A Practical Hybrid Algorithm

Redefining AA

*Naively* ray tracing every pixel is *too expensive*

Find the pixels that will benefit most from supersampling, using the existing 1spp input

**Goal:** harness strengths of TAA while addressing its failures simply and unequivocally

**Goal:** work within the constraints of conventional game engines
Adaptive Temporal Antialiasing

Core Idea

We *efficiently* combine ray and raster, leverage *adaptive* sampling in the context of TAA.

Step 1: Run TAA

Step 2: Compute a segmentation mask of where TAA fails, and why

Step 3: Replace complex post-failure TAA heuristics with robust alternatives: *ray tracing*

Step 4: Use segmentation mask to guide ray tracing adaptivity

Step 5: Enjoy!
Adaptive Temporal Antialiasing

Implementation Details

Unreal Engine 4 extended with DXR API support, running on NVIDIA RTX

TAA fullscreen post-process extended to compute and output segmentation mask

Sparse ray tracing in DXR Ray Generation shaders, dispatched as separate fullscreen post-process pass before tonemapping

Each primary ray casts a single shadow ray to the sun’s directional light source (hard shadows)
Adaptive Temporal Antialiasing

Implementation Details

TAA failure detection (segmentation) is a combination of criteria:

- Motion Vectors
- Segmentation History, single frame look-back (was this pixel marked as ATAA?)
- Luminance, temporal change within a pixel neighborhood
- Depth, 3x3 edge-detecting Sobel filter

Frames are almost always dominated by TAA-classified (blue) pixels
<table>
<thead>
<tr>
<th>Method</th>
<th>Image 1</th>
<th>Image 2</th>
<th>Image 3</th>
<th>Image 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No AA</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>FXAA</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>TAA</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>Mask</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
<tr>
<td>ATAA 2x</td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
<tr>
<td>ATAA 4x</td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
<tr>
<td>ATAA 8x</td>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td><img src="image28.png" alt="Image" /></td>
</tr>
<tr>
<td>SSAA 16x</td>
<td><img src="image29.png" alt="Image" /></td>
<td><img src="image30.png" alt="Image" /></td>
<td><img src="image31.png" alt="Image" /></td>
<td><img src="image32.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Adaptive Temporal Antialiasing

Nota Bene!

Lighting and shading methods between ray and raster must match!

Shadow Rays vs. Shadow Maps, Reflection Cubemaps vs. Reflection Rays

Be aware of pre-AA screen-space algorithms (DoF, Lens Flare, SSAO)

Denoising of area light contributions can make your life tricky
Performance
Titan V

1920x1080 resolution, 107,881 pixels selected for RT, 5.2% of total image resolution

Trace, Material Evaluation, Dynamic Lighting, Reflection Probe, 1 Shadow Ray

<table>
<thead>
<tr>
<th>Variant</th>
<th>Rays</th>
<th>GPU Time (ms)</th>
<th>Optimized GPU Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATAA 8x</td>
<td>1,693,280</td>
<td>18.40</td>
<td>16.85</td>
</tr>
<tr>
<td>ATAA 4x</td>
<td>846,640</td>
<td>9.30</td>
<td>8.55</td>
</tr>
<tr>
<td>ATAA 2x</td>
<td>423,320</td>
<td>4.60</td>
<td>4.28</td>
</tr>
</tbody>
</table>

Performance figures reported in milliseconds (ms)
Conclusions

Demonstrated a practical hybrid AA solution in a production game engine for the first time

The adaptive hybrid strategy injects the advantages of best-quality offline AA strategies while avoiding the limitations of existing best performance real-time methods

Real-time even on first-generation ray tracing commodity hardware and software

Just the beginning: of real-time hybrid raster + adaptive ray tracing
Adaptive Temporal Antialiasing

Future Work

**Performance:** dominated by the ray trace. How to handle about denoising approaches?

DXR is still an “experimental” feature of Windows 10!

Mainstream GPUs do not yet exist that support the DXR API

RT ecosystem is still evolving: wide-spread deployment in games not yet recommended
Adaptive Temporal Antialiasing

Future Work

**Texture LoD:** no forward-difference derivatives in DXR, how to evaluate texture mipmap levels in arbitrary material graphs is an open problem

**Improve Sampling & Filtering:** casting rays in static 8x, 4x, or 2x MSAA n-rooks patterns

**Segmentation:** limited by the 1spp raster input. Conservative raster to the rescue?

**Improved Adaptivity:** enforcing fixed per frame ray budgets and per-pixel ray adaptivity
Acknowledgements

Coauthors: Josef Spjut, Holger Gruen, Rahul Sathe, and Morgan McGuire

Ignacio llamas, Edward Liu, and the entire NVIDIA ray tracing team

Epic Games
@acmarrs