NVIDIA

Booth #801, West Hall

nvidia.com/siggraph2018

ADAPTIVE TEMPORAL ANTIALIASING

Adam Marrs and Rahul Sathe, August 14th 2018

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**

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

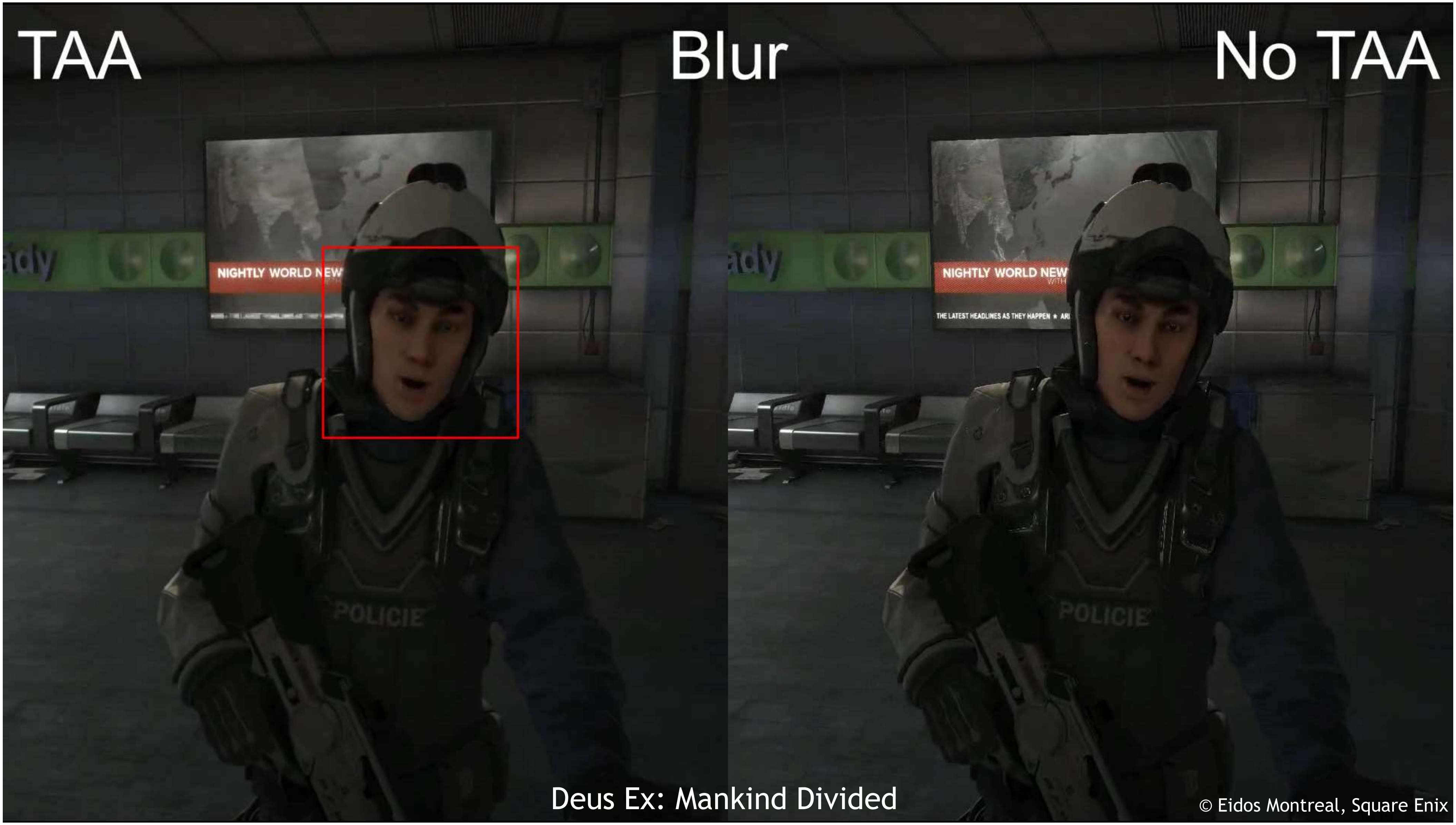
Multisampling: MSAA, CSAA, SBAA [Salvi and Vidimce 2012], SRAA [Chajdas et al. 2011]

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

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

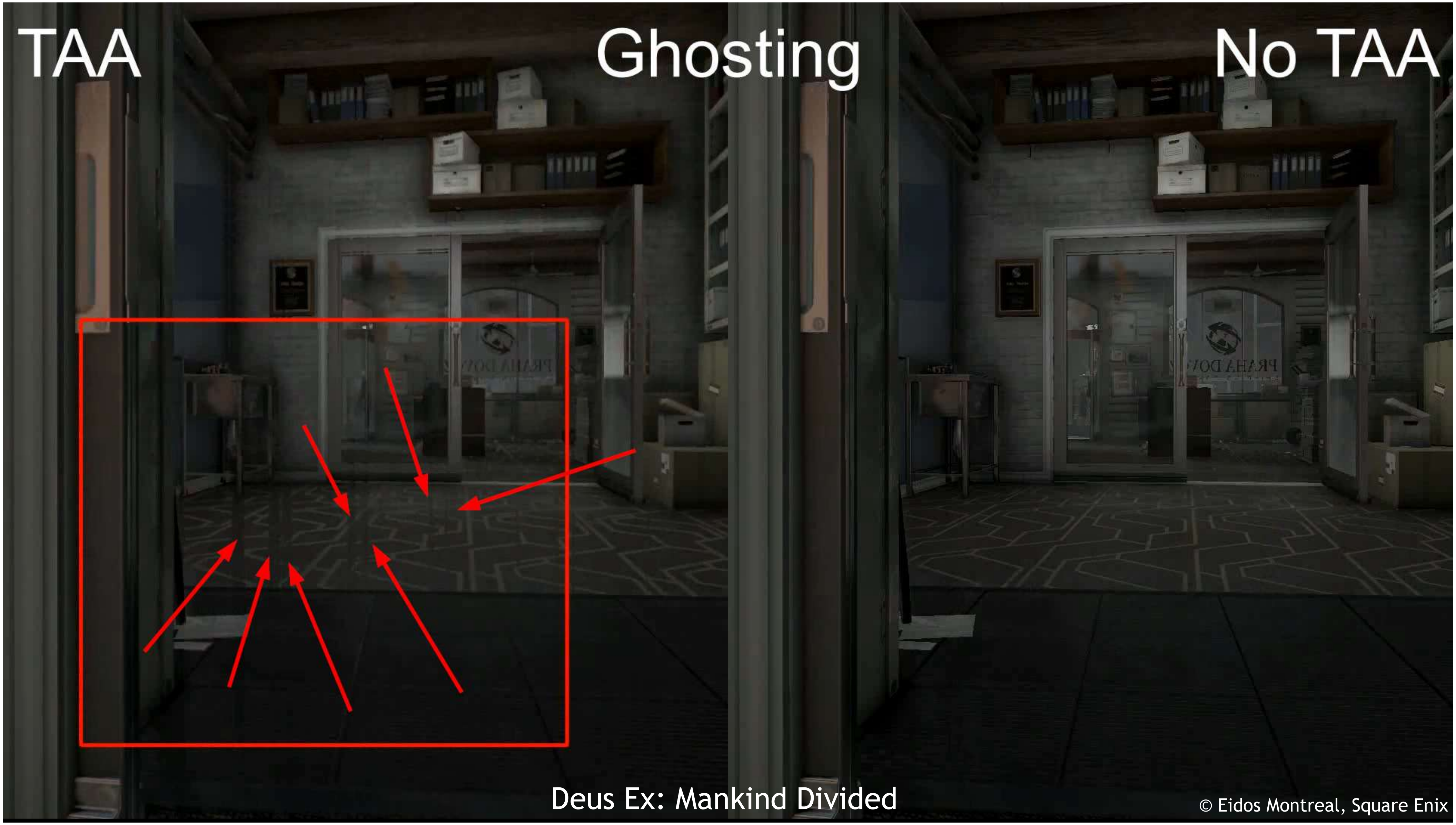
Time: SMAA [Jimenez et al. 2012], TAA [Yang el al] [Karis 2014]

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
NIGHTLY WORLD NEW
POLICIE
NIGHTLY WORLD NEW
THE LATEST HEADLINES AS THEY HAPPEN
POLICIE
Deus Ex: Mankind Divided
© Eidos Montreal, Square Enix

TAA

No TAA


TAA
Ghosting
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

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

NVIDIA Turing Architecture, RTX, and Microsoft DXR enable **full interoperability** between ray and raster rendering on the GPU for the first time

Finding A Practical Hybrid Algorithm

Redefining AA

Goal: find the pixels that will benefit most from supersampling

Goal: leverage Turing's RTCores to accelerate ray tracing and adaptively improve results

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!


ROAD
TO
HOME
EXIT
No AA


EXIT


Sparse RT
EXIT
Post ATAA
Final Frame


ATAA
FXAA
TAA

Adaptive Temporal Antialiasing

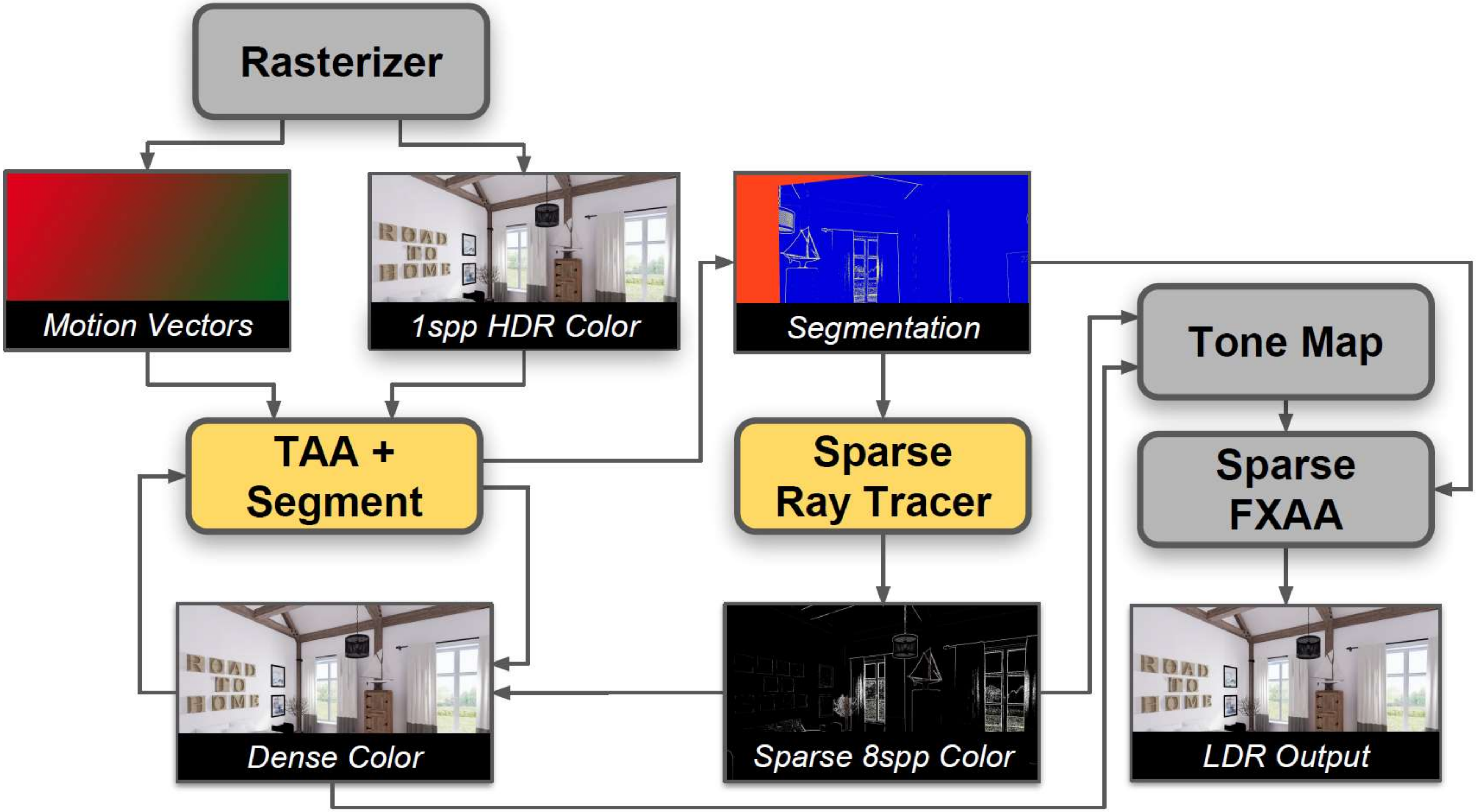
Implementation Details

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)



Rasterizer
Motion Vectors
1spp HDR Color
Segmentation
TAA +
Segment
Sparse
Ray Tracer
Tone Map
Sparse
FXAA
Dense Color
Sparse 8spp Color
LDR Output

Adaptive Temporal Antialiasing

Implementation Details

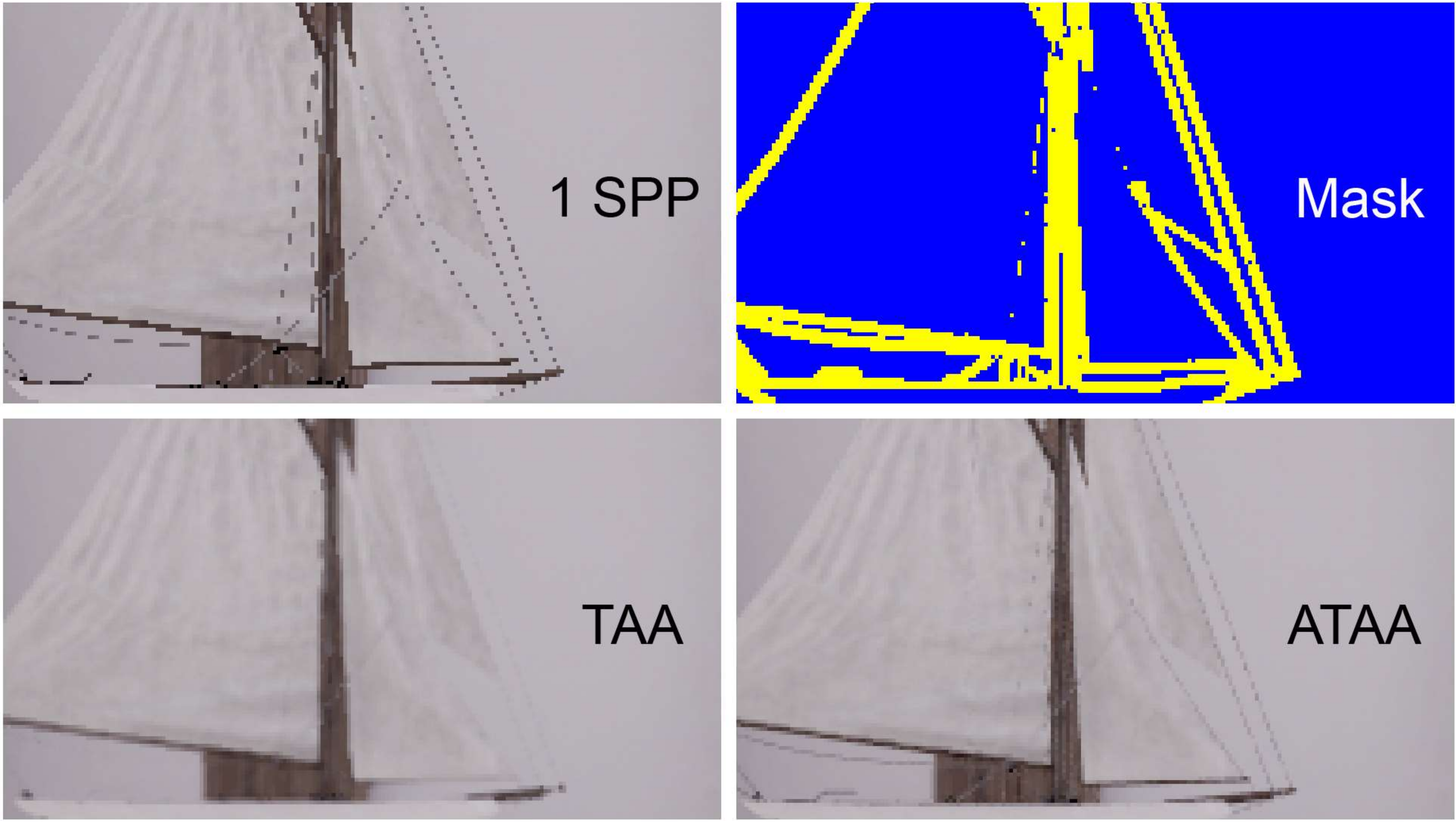
TAA failure detection (segmentation) is a combination of criteria:

- 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

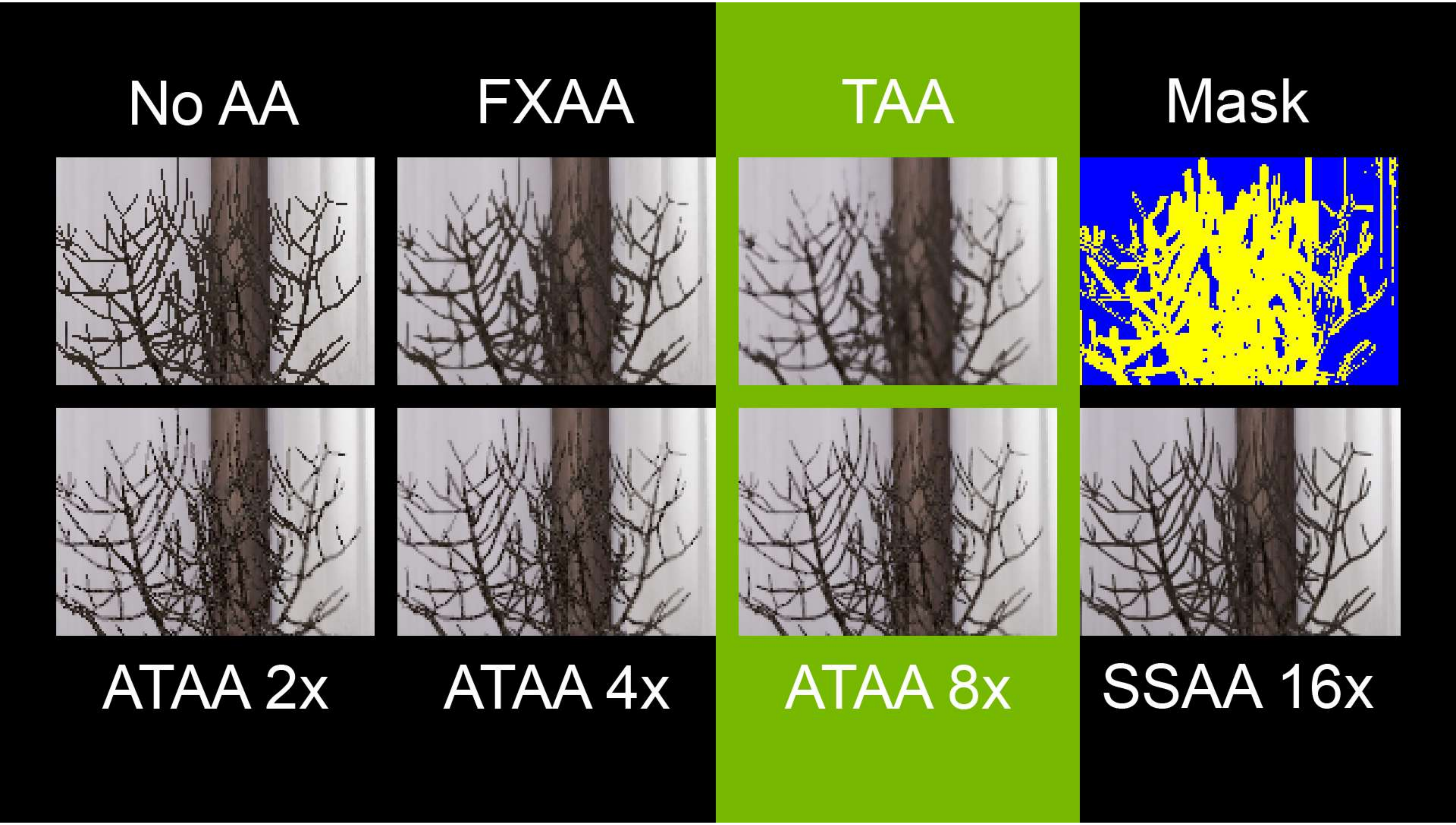
Frames are almost always dominated by TAA-classified (blue) pixels

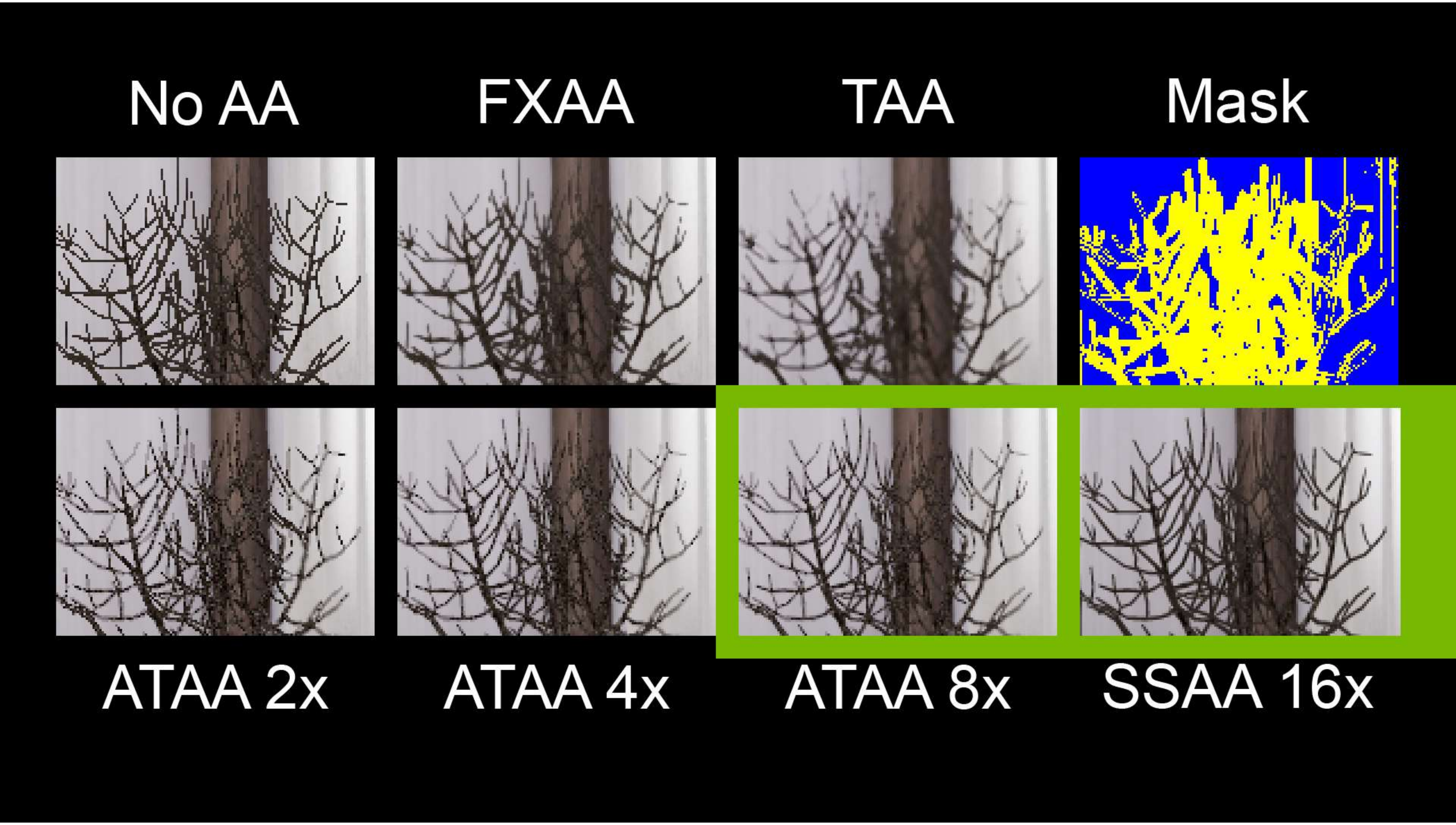

ROAD
TO
HOME
EXIT
TAA

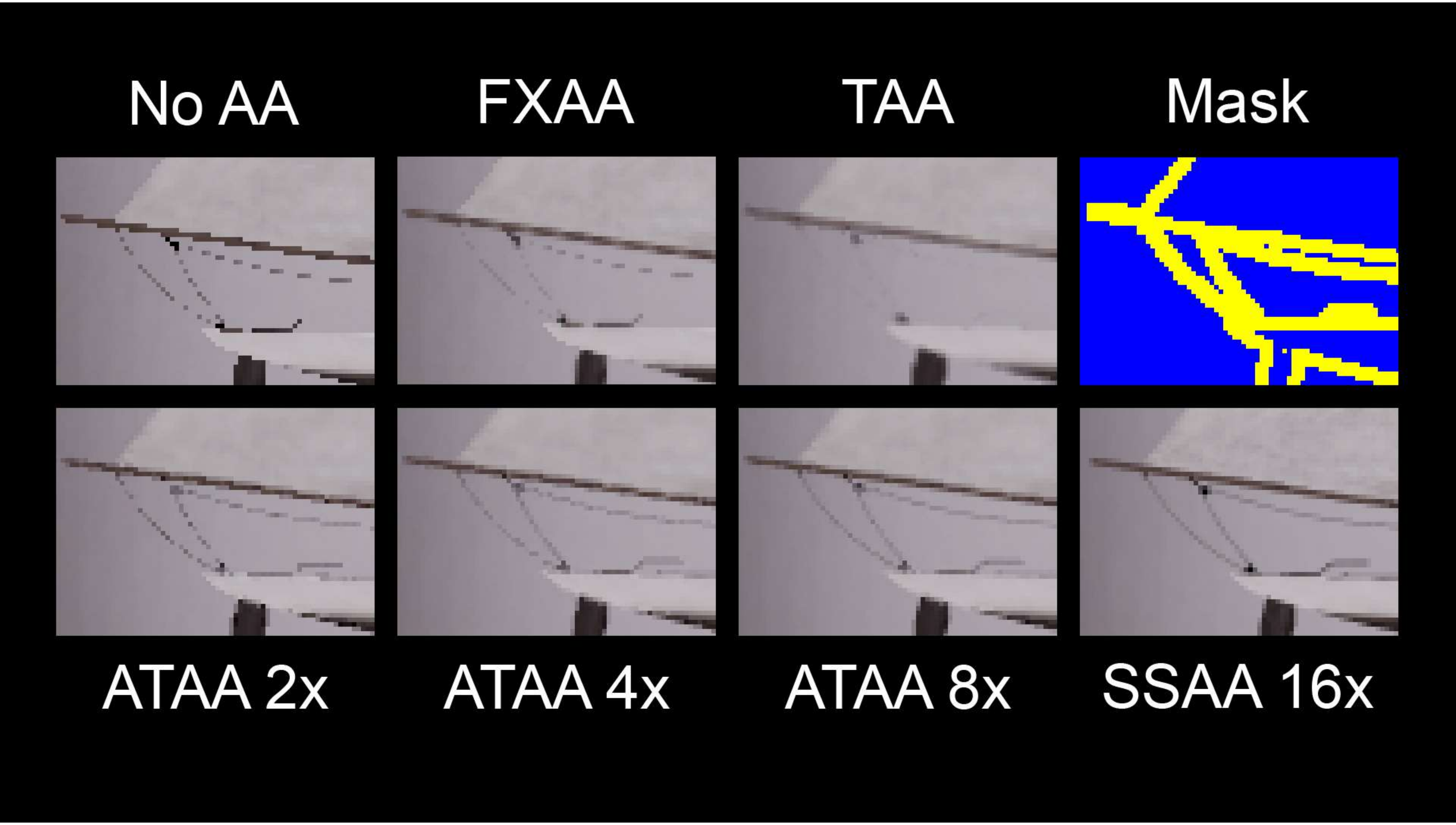

ROAD
TO
HOME
EXIT
ATAA

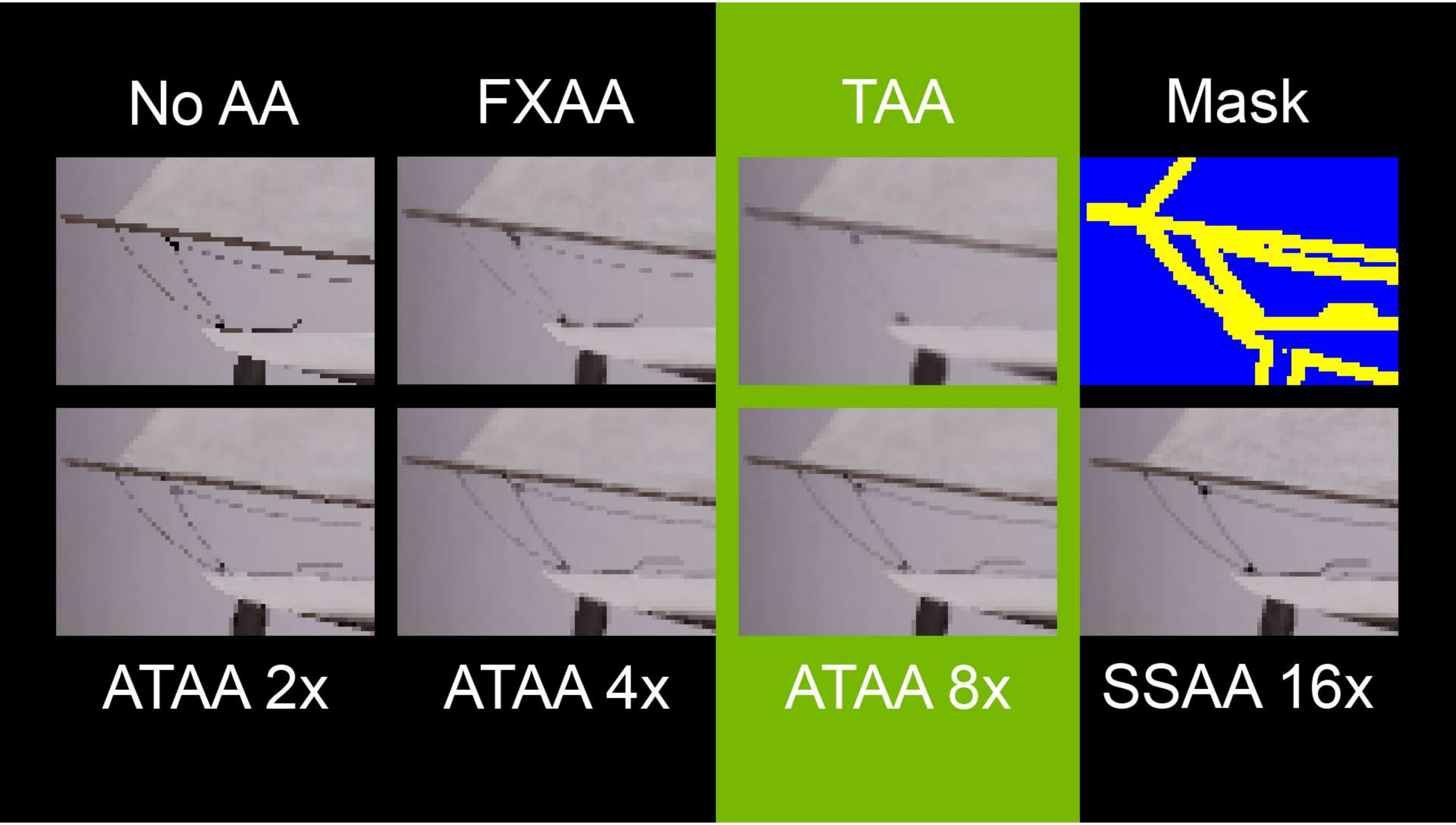

1 SPP
Mask
TAA
ATAA

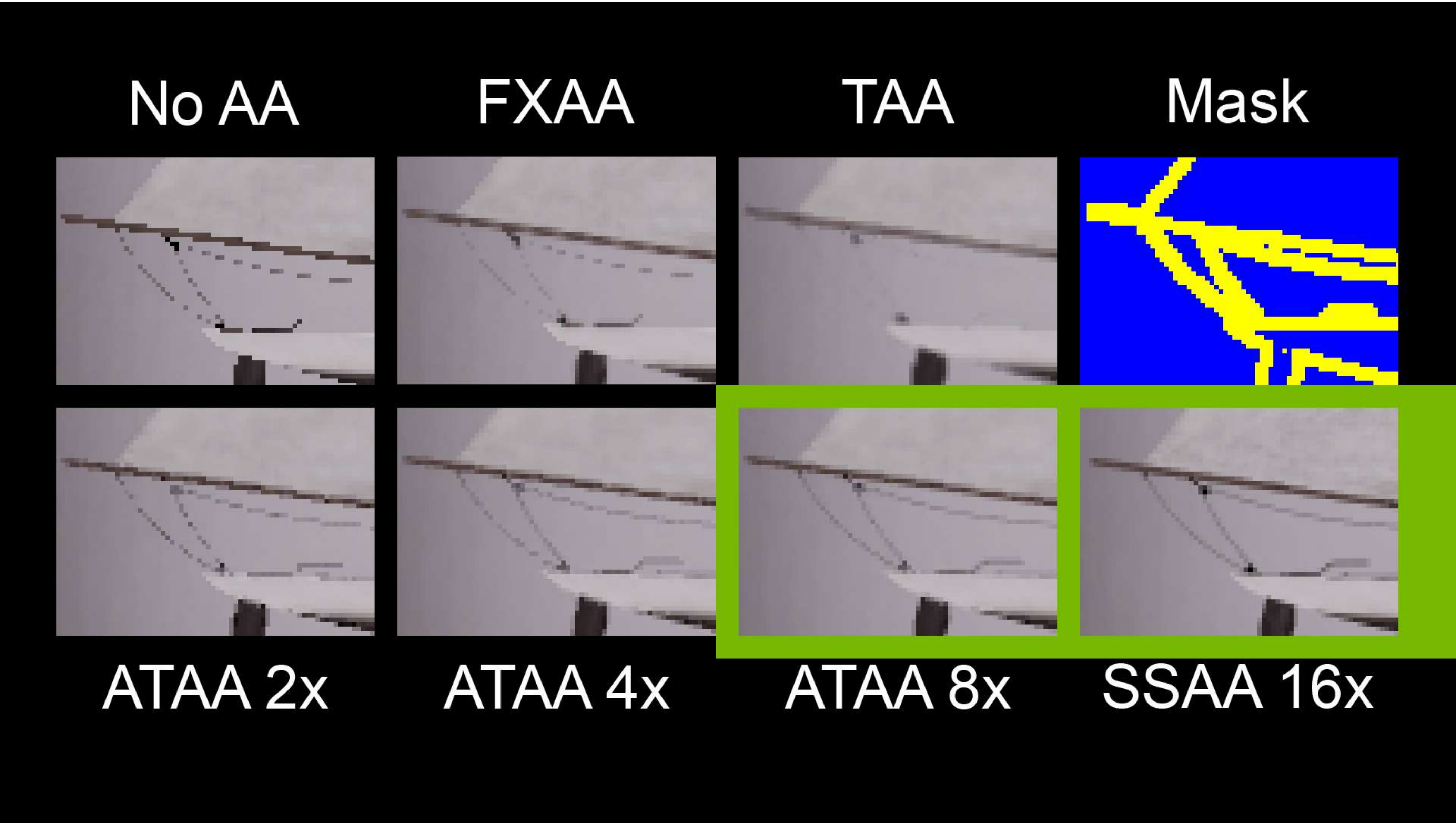

No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x

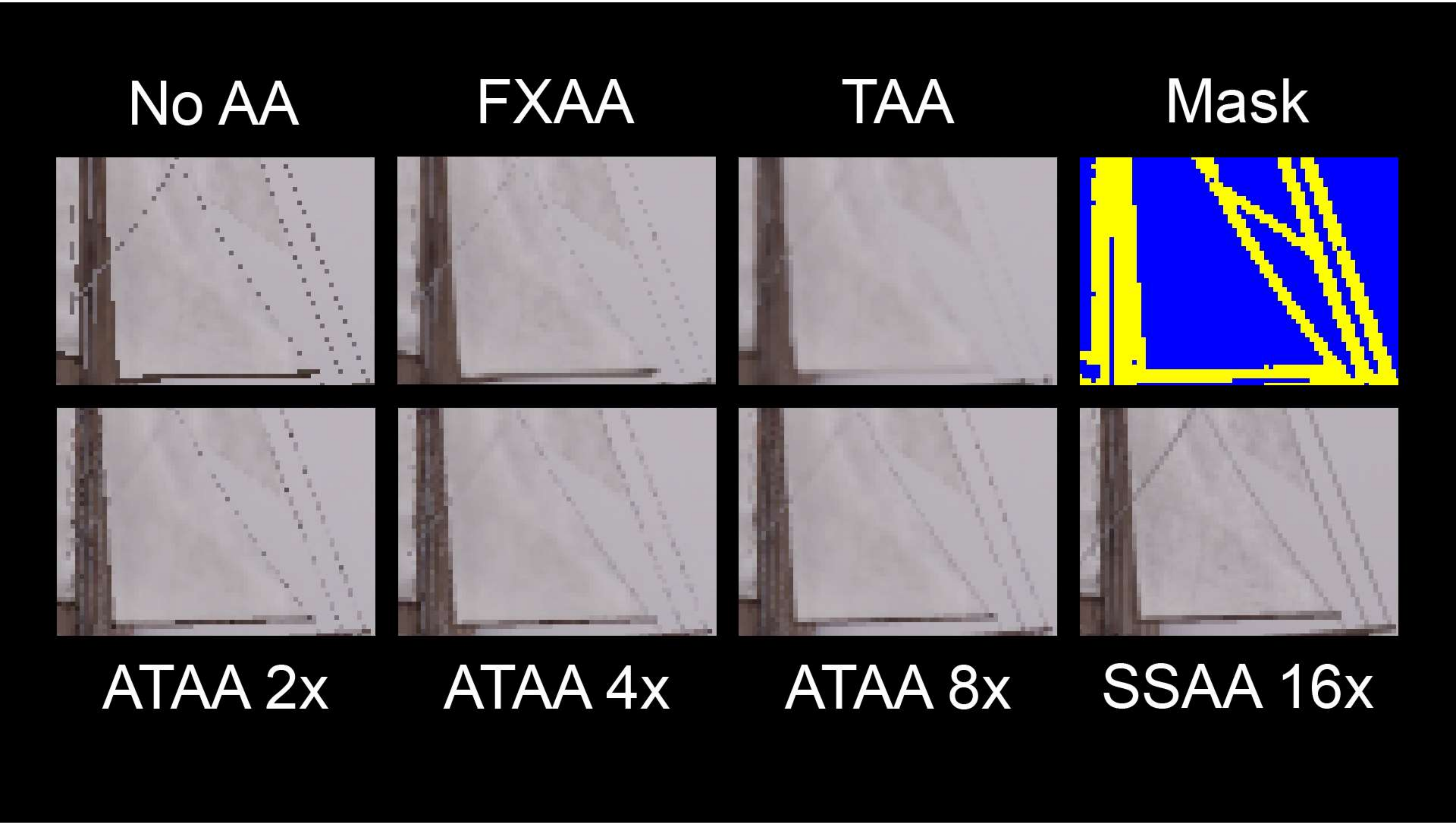

No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x

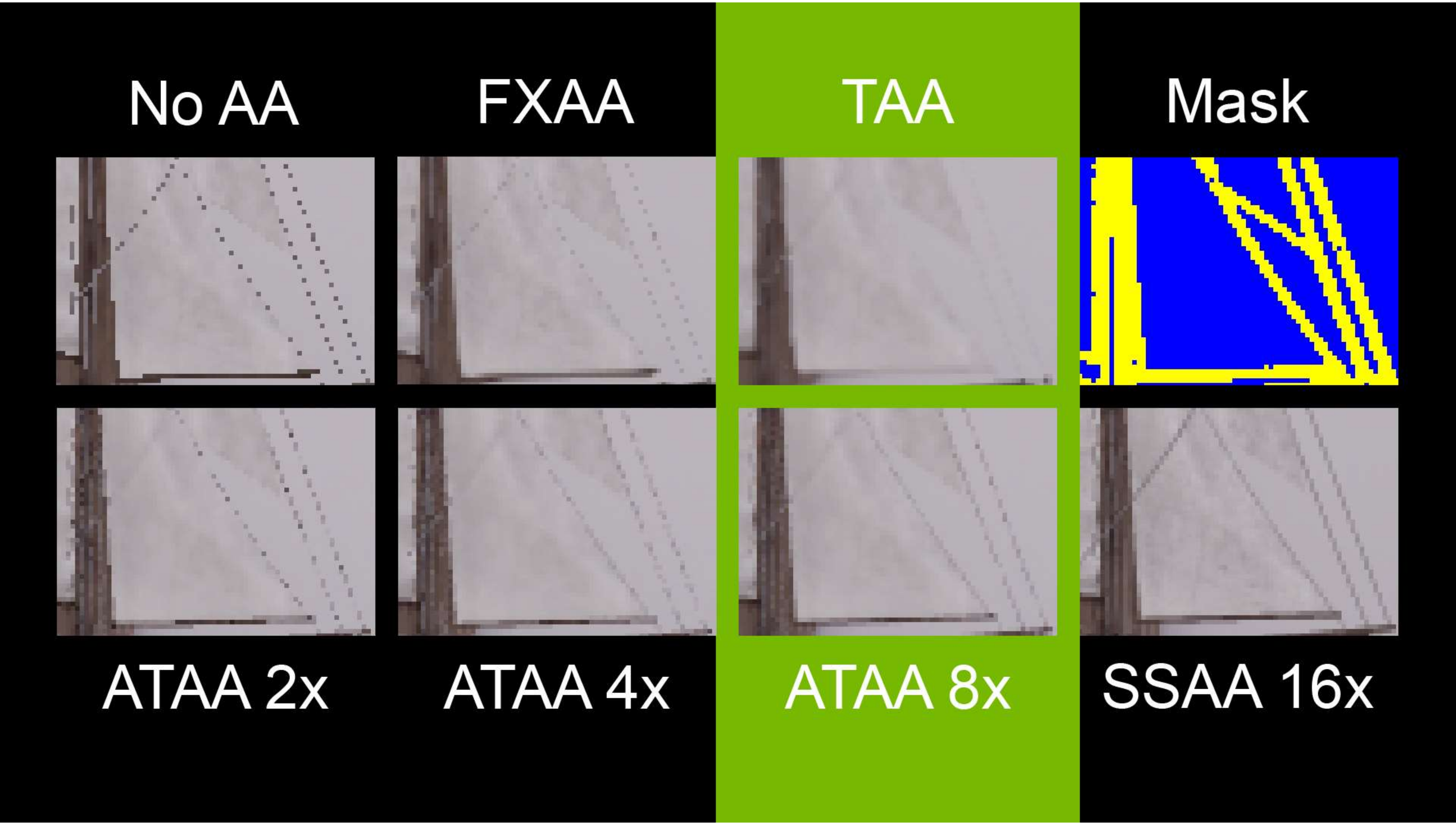

No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x

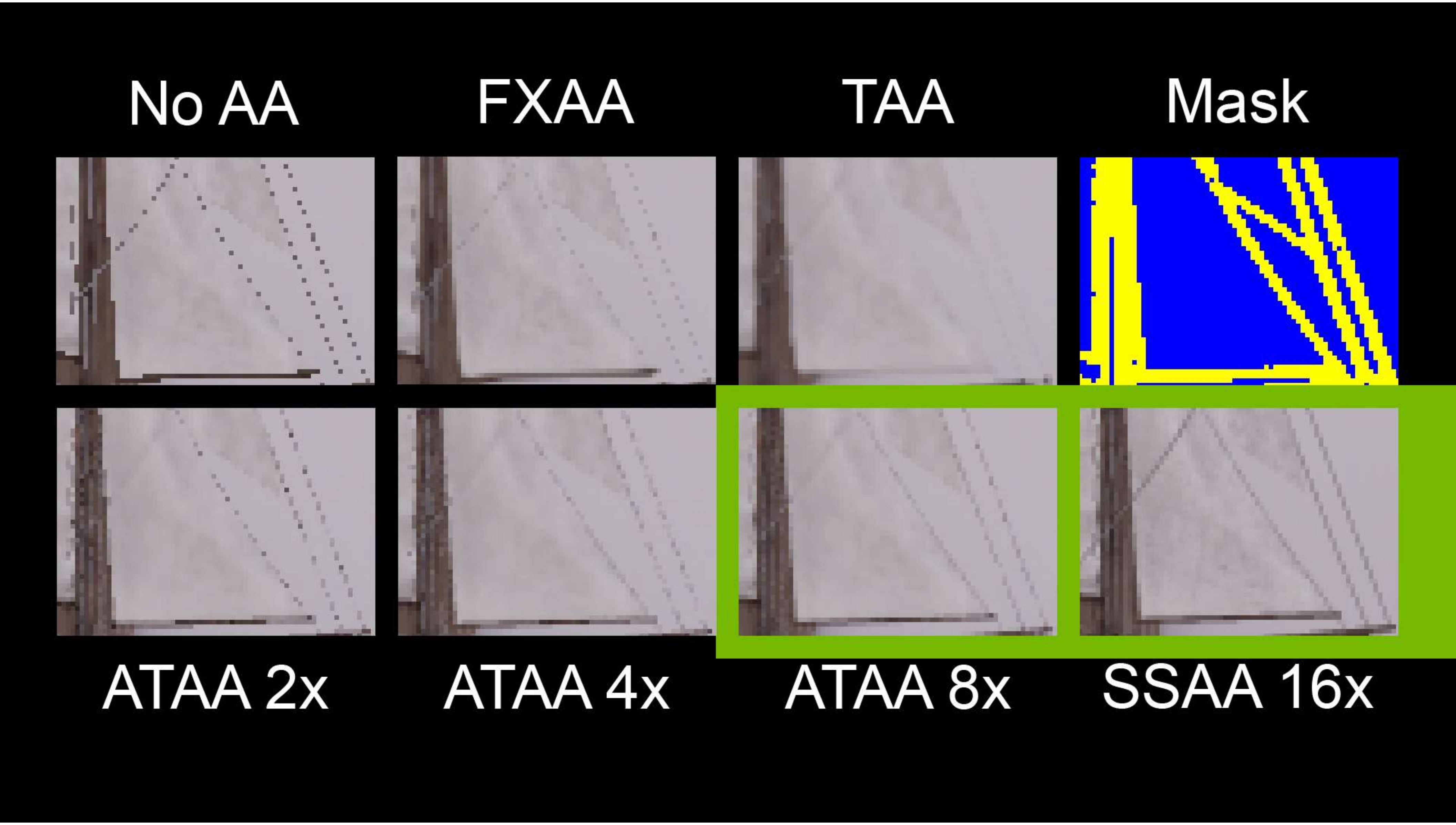

No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x

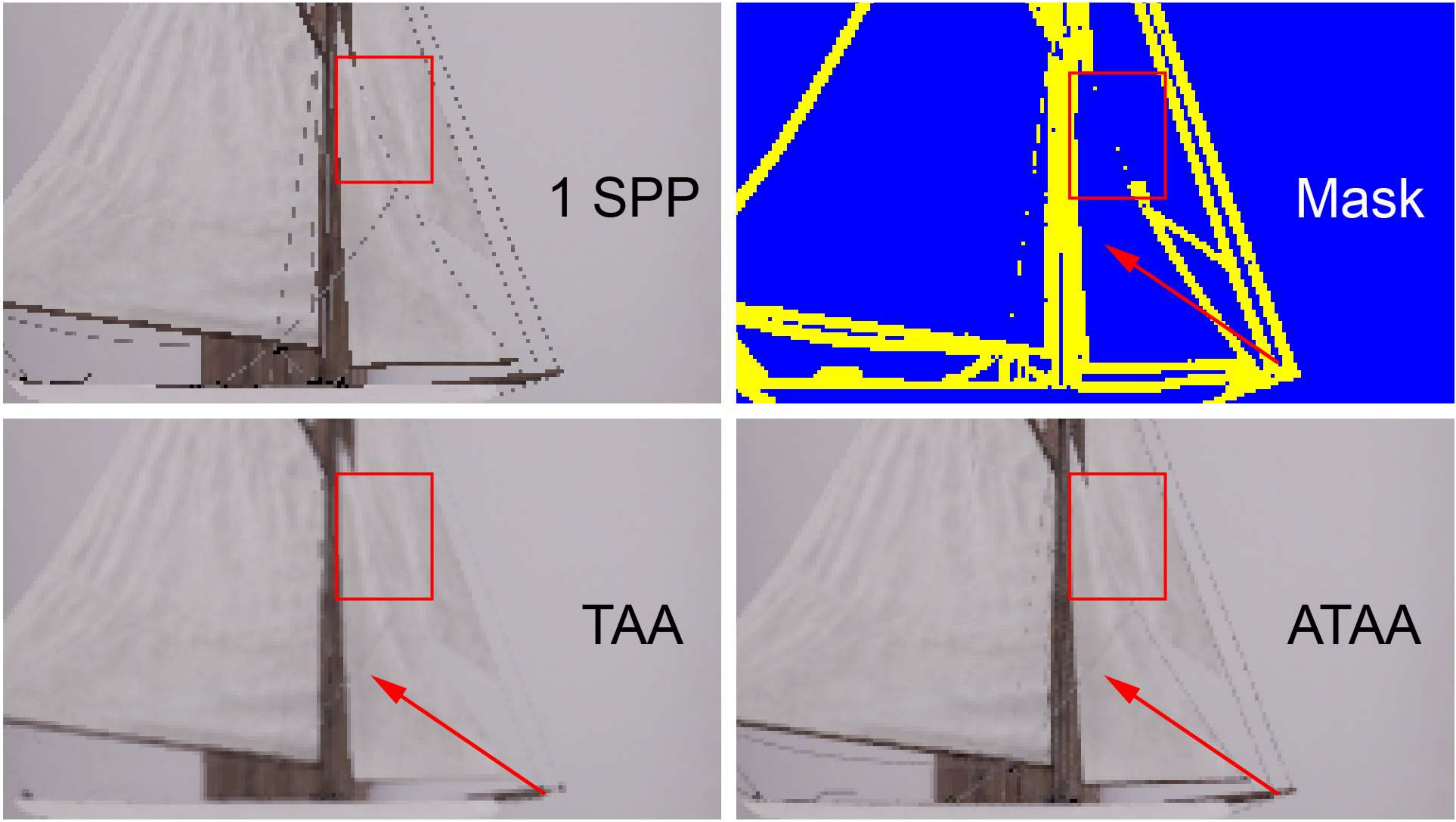

No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x


No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x


No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x


No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x


No AA
FXAA
TAA
Mask
ATAA 2x
ATAA 4x
ATAA 8x
SSAA 16x


1 SPP
Mask
TAA
ATAA

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

Variant	Rays	Titan V (Volta) GPU Time (ms)
ATAA 8x	1,693,280	**16.85**
ATAA 4x	846,640	8.55
ATAA 2x	423,320	4.28

Performance figures reported in milliseconds (ms)

Performance
Quadro RTX 6000 vs. 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

Variant	Rays	Titan V (Volta)	Quadro RTX 6000 (Turing)	Speedup
ATAA 8x	1,693,280	16.85	5.58	**3x**
ATAA 4x	846,640	8.55	2.83	**3x**
ATAA 2x	423,320	4.28	1.45	**3x**

UE4 Total Frame Time (with ATAA 8x) : 9.8ms

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

Incredible performance speedup from Turing's RTCores

Game-Ready: with costs as low as 1.45ms on Turing in UE4, ATAA on Turing is poised to reinvent AA

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

ADAPTIVE AA WITH CONSERVATIVE RASTERIZATION

Use conservative rasterization to identify “interesting” pixels

ADAPTIVE AA WITH CONSERVATIVE RASTERIZATION

Use conservative rasterization to identify “interesting” pixels

- Introduced in D3D11.3 for feature level 12 hardware

ADAPTIVE AA WITH CONSERVATIVE RASTERIZATION

Use conservative rasterization to identify “interesting” pixels

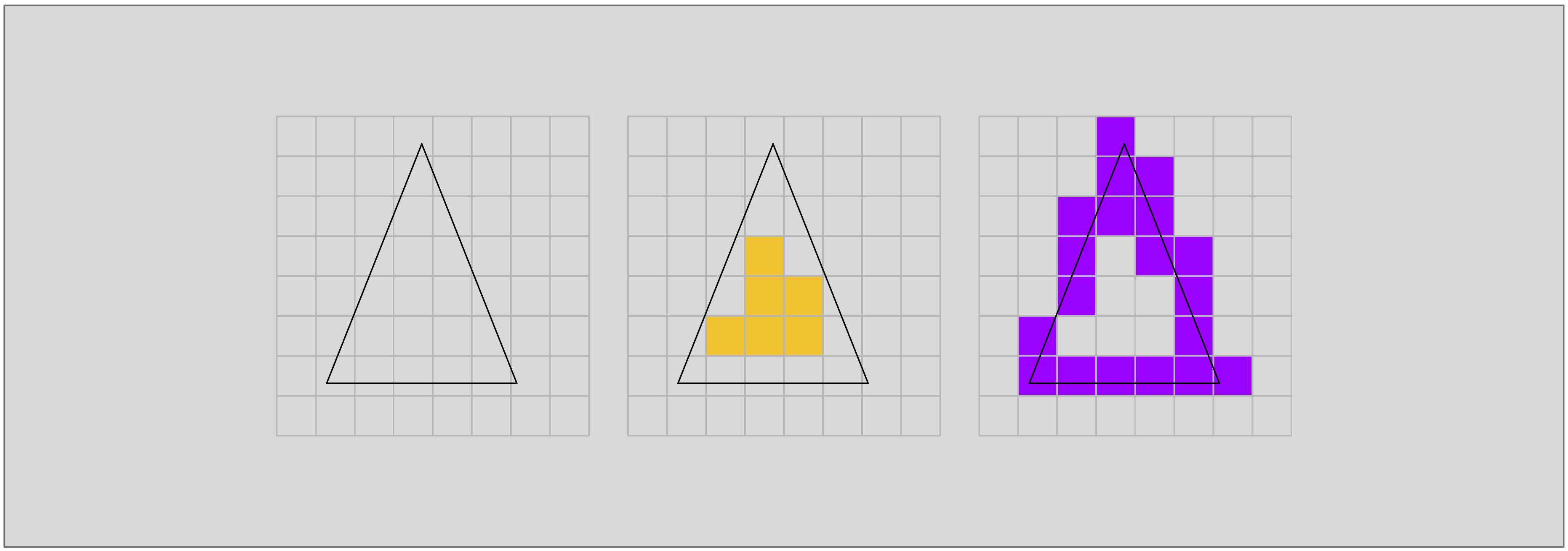
- Introduced in D3D11.3 for feature level 12 hardware
- Rasterizes pixels where the fragment intersects with the pixel extents
    - Not just covers the sample(s)

ADAPTIVE AA WITH CONSERVATIVE RASTERIZATION

Use conservative rasterization to identify “interesting” pixels

- Introduced in D3D11.3 for feature level 12 hardware
- Rasterizes pixels where the fragment intersects with the pixel extents
    - Not just covers the sample(s)
- Tier 3 allows identifying pixels that are fully and/or partially covered
    - SV_InnerCoverage

CONSERVATIVE RASTERIZATION



THE ALGORITHM

GS (Fast GS on NVIDIA)

- Calculates depth plane equation coefficients, e.g. $z = Ax + By + C$
- Calculates min/max depths of the primitive
- All calculations are done in the clip-space to avoid clipping issues

THE ALGORITHM

GS (Fast GS on NVIDIA)

- Calculates depth plane equation coefficients, e.g. $z = Ax + By + C$
- Calculates min/max depths of the primitive
- All calculations are done in the clip-space to avoid clipping issues

PS classifies the pixel as “interesting” based on `SV_InnerCoverage`

- Partially covered → Output 0x1 indicating “interesting”
- Fully covered → Output 0x0 indicating “not so interesting”

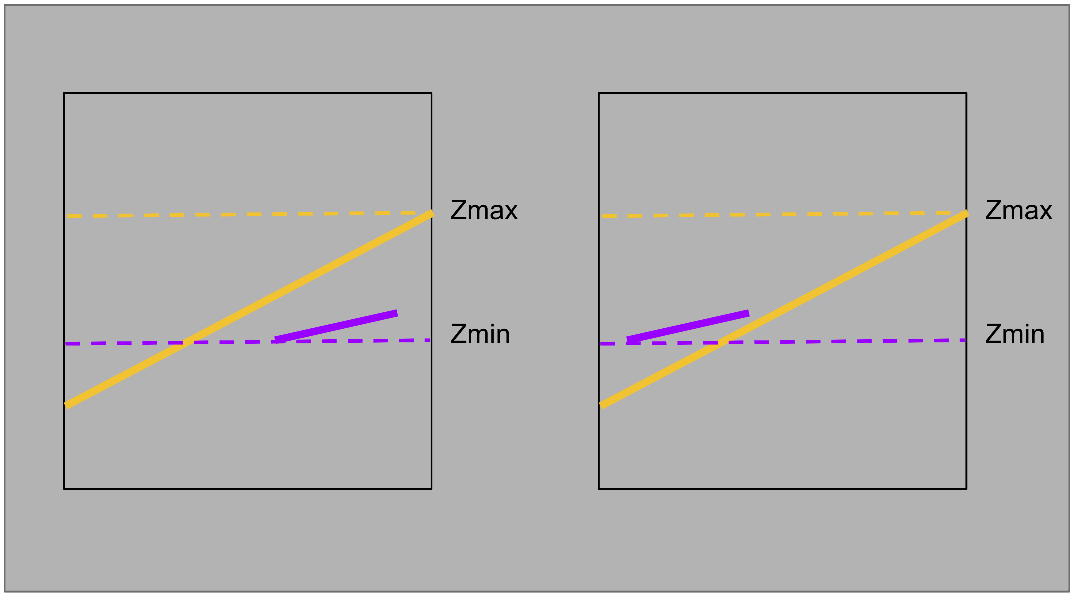
THE ALGORITHM

PS also generates the depths using the plane equation, such that

- No potentially visible (partially covered) fragment is occluded
    - Output the closest depth
- A fully covered fragment occludes as many fragments as possible conservatively
    - Output the farthest depth

PS clamps the min/max depths to the primitive depths calculated in the GS

THE ALGORITHM



Outputting conservative depths from the PS
“interesting” pixels output Zmin and full covered output Zmax

```
void main(
   in float4 i_position                 : SV_Position,
   in uint m_InnerCov                   : SV_InnerCoverage;
   in nointerpolation float3 planeCoeff : PLANE_COEFF;
   in nointerpolation float2 primDepth  : PRIM_DEPTH_MIN_MAX;
   out uint o_color                     : SV_Target0,
   out float o_depth                    : SV_Depth)
{
 // Set the defautl min and max values.
 float zMin = MAX_VAL, zMax = MIN_VAL;

 // Calculate Z at the four corners and calculate min/max of those.
 // Depth is a monotonic function, so evaluating at corner suffices,
 // unless primitive is entirely contained within a pixel OR contains
 // one of the vertices.
 CalculateZminZmax(zMin, zMax, planeCoeff, i_position);

if (i_vtx.m_InnerCov & 0x1) {
       o_color = 0x0;    // Fully covered pixel
       o_depth = zMax;   // Output the farthest depth
   } else {
       o_color = 0x1;    // Partially covered (“interesting”) pixel
       o_depth = zMin;   // Output the
   }
```

Pseudo-code
Pixel Classification Shader

DXR BASED VISIBILITY

DXR distributes the samples over the “interesting” pixels

Sample positions and number are completely programmer controlled

Rays are generated from the camera towards these samples

Ray Trace those rays against the BVH

NON-DXR BASED RAY TRACING

Pass the vertex positions from GS to PS

For “interesting” pixels

- Generate up to N rays from the eye
- Calculate the ray triangle intersection
- Output the depth/color while keeping track of the nearest depth

Resolve the final color

MORE DETAILS…

Book chapter in upcoming **GPU Zen 2** book.


NVIDIA

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