Filtering Approaches for Real-Time Anti-Aliasing

http://www.iryoku.com/aacourse/
Filtering Approaches for Real-Time Anti-Aliasing

Anti-Aliasing Methods in CryENGINE 3

Tiago Sousa
R&D Principal Graphics Engineer
Crytek

tiago@crytek.com
CryENGINE 3 AA Requirements

- Orthogonal and general solutions
  - No per-platform AA solution
- Play nice with HDR/Deferred techniques
- Sub-pixel accuracy is important for us
  - Schimering was the biggest offender on Crysis 1 and 2 levels
  - Crysis had immensely aliased assets: alpha tested/tiny sub-pixel details
  - HDR makes it even worse, big range of lighting contrast/color variation
- Low memory footprint
- Cost less than 2 ms on low end GPUs
  - Every ms counts for consoles
Multitude of workarounds and added complexity
Each platform with its own solution, workarounds and optimizations
Performance cost

For current Gen, situation doesn’t look promising...

Minimal G-Buffer
- RGBA8 World Space BF Normals + Glossiness
- Readback Z24S8 Depth + Stencil for tagging interior areas

Deferred lighting buffers
- Xbox 360: 2x RGB10F resolved to RGB10A2 for performance/memory
- PS3: 2x RGBA8 encoded in RGBK
- PC: 2x FP16

Scene buffer
- X360: RGB10F resolved to FP16 for performance
- PS3: RGBA8 encoded in RGBK for opaque, FP16 for transparents
- PC: FP16

X360: 28 MB; PS3: 31.5MB; PC: 49 MB at 720p, 110 MB at 1080p and so on.
The Quest for AA Alternatives
Increasingly popularity since Crysis 1 and KZ2

Aproximated as directional blur along screen space velocity vector

Delta from prev/cur screen space position, per-pixel or per vertex

Main benefit: Less noticeable aliasing during movement

Added bonus: Great cinematic look

User for all platforms
A-Buffer SSAA [Haeberli90]

- Add sub-pixel jitter to camera frustum
- Brute force: Render scene multiple times
  - N sub samples ↔ N scene renders
- Robust and best quality
  - Also more uses besides SSAA (TSSAA/DOF/Soft-Shadows)
- Base concept used for our techniques

- Problem: Cannot afford render scene multiple times (yet)
  - Great for reference/marketing quality shots though

Alternatively, the usual approach of rendering higher resolution and downscale
Distribute A-Buffer SSAA Overframes

• Running at 60 fps?
  – Add sub-pixel jitter to camera frustum every frame
  – Store previous/current frame and linear blend them
  – Light-speed 2x SSAA: ~0.5 ms on current consoles
  – 2 frames ⇔ 2x SSAA, 4 frames ⇔ 4x SSAA, etc

• But... not many reach 60 fps on consoles
  – Lower fps results in extremely noticeable image ghosting

Used in few older 60 fps games (DMC4?) , but who was first?

And even at 60 fps, still noticeable ghosting
Strafing is the worst case, we’ll use if for images
Minimizing Artifacts

• Improving blending: Reprojection
  – Velocity vector fetches from previous frame sub-sample target
  – Exactly same as in TAA (but single tap)

• Deformable geometry slightly more expensive to handle
  – Output pixel velocity into a render target
  – Could not afford for vegetation

• Problem: Disoccluded regions ghosting
Using reprojection
Minimizing Artifacts (2)

- Disable blending if $||V|| > 0$?
  - Very rare the case when player not moving
  - And we still want AA during camera movement
- Weighting using color/edge tagging?
  - Sub-pixel/hi frequency detail results in noticeable schimering
- Reprojection range clamping
  - Pixel weight proportional to reprojection limit
    - Eg: $f\text{BlendW} = \text{saturate}(1 - (f\text{VLen} / f\text{VMaxLen}))$
  - Coarse Depth stored in sub-sample buffer alpha channel
    - Mask out if $f\text{VLen} > f\text{MaxVThreshold}$ and $f\text{CurrD} > f\text{PrevD}$

Disable accumulation if $||V|| > 0$?
  - Very rare the case when player not moving
  - And we still want AA during camera movement

Weighting using color?
  - Sub-pixel/hi frequency detail results in noticeable schimering

Reprojection range clamping? (used for Crysis 2)
  - Pixel weight proportional to reprojection limit
    - Eg: $f\text{BlendW} = \text{saturate}(1 - (f\text{VLen} / f\text{VMaxLen}))$
  - Coarse Depth stored in sub-sample buffer alpha channel
    - Mask out if $f\text{VLen} > f\text{MaxVThreshold}$ and $f\text{CurrD} > f\text{PrevD}$
  - 8 bit insufficient to cover large range, limited to nearby
Clamped reprojection (used in Crysis 2)
Minimizing Artifacts (3)

- Store $|V|$ in sub-sample buffer alpha channel
  - Weight: $\frac{\text{abs}(f\text{PrevLenV} - f\text{CurrLenV})}{f\text{MaxLen}}$
Clamped reprojection + Velocity weighting
float fDepth = GetLinearDepth(sDepth, tcBase.xy);
float4 vPosWS = WorldViewPos.xyz + IN.vCam.xyz * fDepth;

float4 vPrevPos = mul(mViewProjPrev, float4(vPosWS, 1.0 ));
vPrevPos /= vPrevPos.w;

float2 vVelocity = vPrevPos.xy - tcBase.xy;
half4 cObjVelocityParams = tex2D(sObjVelocity, tcBase.xy);
half2 vObjVelocity = DecodeMotionVector(cObjVelocityParams);

vVelocity = cObjVelocityParams.w? vObjVelocity : vVelocity;
float fVLenSq = dot(vVelocity.xy, vVelocity.xy) + 1e-6f;

vVelocity /= fVLenSq;

half4 cCurr = tex2D(sCurrFrame, tcBase.xy);
half4 cPrev = tex2D(sPrevFrame, tcBase.xy + vVelocity * min(fVLenSq, fMaxLen));

half fBlendW = 0.5 - 0.5 * saturate(fVLenSq / fMaxLen);
fBlendW *= saturate(1 - (abs(cCurr.a - cPrev.a) * fWeightScale ));

OUT.Color = lerp(cCurr, cPrev, fBlendW);
2x Quincunx SSAA

- Improving quality with 2 sub-samples
  - Bilinear fetch to one of sub-samples
  - "Aproximate" 4x SSAA
2x Quincunx SSAA

No AA  2x SSAA  2x Quincunx SSAA
Distributed A-Buffer SSAA: Caveats

- Not temporally stable
  - No AA on disoccluded regions
  - Input signal changes (color/lighting), no robust solution yet

- Alpha blending problematic
  - Withouth OIT, only possible to handle correctly for first hit
  - Additional overhead

- Multi-GPU
  - Additional frame latency to address
  - For Crysis 2, we switched to Nvidia’s FXAA when in MGPU
    - Schimering was again, biggest complain from MGPU users
Future Work

- SSAA combo with post processed AA
  - Maybe similarly to DLAA: horizontal/vertical edges, blend taps
    - This means at least 4 additional taps
  - AA on disoccluded regions
Distributed A-Buffer SSAA: Current Results

Far from perfect, but:

• Orthogonal
• Sub-pixel accuracy
  – Shader anti-aliasing bonus
• 2x Quincunx SSAA: 1 ms for consoles
  – 0.2 ms at 1080p on pc’s
  – 2x SSAA + edge AA: 1.7 ms
  – 4x SSAA + edge AA: 2.2 ms
  – 3 MB additional memory footprint

Missing pretty pictures
Acknowledgements

- Nick Kasyan, Nicolas Schulz, Vaclav Kyba, Michael Kopietz, Carsten Wenzel, Vladimir Kajalin, Andrey Konich, Ivo Zoltan Frey
- Jorge Jimenez, Diego Guitierrez, Naty Hoffman
- And to the entire Crytek team
Further Readings


Green, S “Stupid OpenGL Shader Tricks”, 2003


Swoboda, M “Deferred Rendering in FrameRanger”, 2009

Yang, G et al “Amortized Super Sampling”, 2010


Sousa, T., Kasyan, N. and Schulz, N. “Secrets of the CryENGINE 3 Technology”, 2011
Questions?

tiago@crytek.com

twitter: crytek_tiago
If time allows
Bonus: Marketing Screenshots

- Always some trickery
  - On CryENGINE 2 rendered multiple tiles at big resolutions and downsampled to get SSAA

- On CryENGINE 3 distributed SSAA with many samples
  - Random sub-pixel jitter
  - Almost perfect SSAA
  - All Crysis 2 marketing shots used this variation