High Quality Temporal Supersampling

Brian Karis  (@BrianKaris)
Context

- Unreal Engine 4’s primary anti-aliasing solution
  - Referred to as Temporal AA in the engine
- First used in the UE4 Infiltrator tech demo
- Several major revisions since then
- Still ongoing work
UE4 renderer

- Deferred shading
- Physically based
- HDR
No AA
Problem

- Horribly aliased input
- Both geometric and shading aliasing
- Mostly from subpixel features
- Want temporal stability
MSAA?

- Too expensive with deferred
  - Don’t want to shade more than once per pixel
- Doesn’t affect shading aliasing
  - More significant aliasing inside triangles than at their edges
Spatial filter?

- MLAA, FXAA, SMAA, etc.
- Essentially edge finding, reduces stair stepping
  - Primarily not a stair stepping problem
- No knowledge of subpixel features
- Not temporally stable
  - Even on simple stair stepping

[Reshetov09], [Lottes11], [Jimenez12]
Specular Lobe filtering?

- Toksvig, LEAN, vMF, etc.
- Filters shading input to prevent subpixel shading output
- Difficult to pre-filter everything
  - Geometric features are major contributor
  - Often no existing unique roughness map
  - Procedural texturing
  - Still aliases
- Screen space filter aliases
  - Misses subpixel features

[Hi12], [Neubelt13], [Mittring12]
Temporal filtering

- Distribute samples over multiple frames
- I’ve had great success with this in the past
  - SSAO
  - SSR
- Replaced spatial filter
  - Higher quality
  - Cheaper
- Do the same with supersampling?
Step 1: Static scene
Jittering

- Adjust projection matrix

```cpp
ProjMatrix[2][0] += ( SampleX * 2.0f - 1.0f ) / ViewRect.Width();
ProjMatrix[2][1] += ( SampleY * 2.0f - 1.0f ) / ViewRect.Height();
```

Regular grid
Sample pattern

- Want a low discrepancy progressive sequence
  - No clustering in either space or time
- Halton (2,3) worked well enough
  - Better than any HW MSAA sample ordering
Moving average

• Simple moving average
  – Not enough samples
  – \( n = 2 \) practical for color
  – \( n = 5 \) if luma only

• Exponential moving average
  – Nearly infinite number of samples with fixed storage

\[
s_t = \alpha x_t + (1 - \alpha)s_{t-1}
\]

\[
s_t = \frac{1}{n} \sum_{k=0}^{n-1} x_{t-k}
\]
Exponential smoothing

- When $\alpha$ is small exponential $\approx$ simple

$$s_t = \alpha x_t + (1 - \alpha)s_{t-1} = \alpha \sum_{k=0}^{\infty} (1 - \alpha)^k x_{t-k}$$

$$x_t = x_{t-n} \Rightarrow s_t = \frac{\alpha}{1 - (1 - \alpha)^n} \sum_{k=0}^{n-1} (1 - \alpha)^k x_{t-k}$$

$$\lim_{\alpha \to 0} \frac{\alpha}{1 - (1 - \alpha)^n} \sum_{k=0}^{n-1} (1 - \alpha)^k x_{t-k} = \frac{1}{n} \sum_{k=0}^{n-1} x_{t-k}$$
When to average?

• **Before tone mapping**
  – The physically correct location
  – Bright values dominate
  – Aliases badly with limited # of samples

• **After tone mapping**
  – All post filters flicker
  – Aliased input → aliased output
Straightforward tone map solution

- Hybrid of before and after
  - Apply before all post
  - Tone map input
  - Accumulate samples
  - Reverse tone map output

- Same AA quality as after tone mapping

- Provides AAed input to post processing chain
  - No more flickering bloom
Better tone map solution

- Tone mapping desaturates bright pixels
- Weight samples instead based on luminance
  - Maintains chroma
  - Perceptually closer to ground truth
- No need to store the weight
  - Rederive weight
  - Saves GPRs
- See my blog post: [Karis13]

\[
\text{weight} = \frac{1}{1 + \text{luma}}
\]

\[
T(\text{color}) = \frac{\text{color}}{1 + \text{luma}}
\]

\[
T^{-1}(\text{color}) = \frac{\text{color}}{1 - \text{luma}}
\]
Tone map
Tone map vs Luma weight

Luma weight
Reconstruction filter

- Box filter is not stable under motion
- PRMan anti-aliasing guide
- Gaussian fit to Blackman-Harris 3.3
  - Support is ~2 pixels wide

\[ W(x) = e^{-2.29x^2} \]
Step 2: Dynamic scene
Reprojection

- History for current pixel may be elsewhere on screen
  - May not exist at all
- Use same velocity buffer calculation as motion blur
- Remember to remove jitter
Velocity accuracy

• Need velocity (motion vectors) for everything
  – Motion without correct velocity will smear
• Accuracy is super important
  – Minor imprecision will streak a static image
  – 16:16 RG velocity buffer
• Can be tricky
  – Procedural animation
  – Scrolling textures
  – Almost opaque translucent objects
Motion on edges

- Moving silhouette edges lose AA
  - Smooth AAed edge doesn’t move with object
  - Effectively an aliased mask in the velocity buffer

- Dilate velocity
  - Take front most velocity
Ghosting
Ghosting

- Depth compare?
  - All samples don’t share same depth
- Velocity weighting?
  - Shading changes
  - Translucency

[Nehab07], [Sousa11]
Neighborhood clamping

- New kid in town!
  - [Lottes11], [Malan12]
- Restrict history to the range of current frame’s local neighborhood
  - Assumes AA result is blend of neighbors
  - Clamp with min/max of 3x3 neighborhood
Neighborhood clamping artifacts
Shaped neighborhood clamp

- Simple clamp to min/max of 8 neighbors results in 3x3 box artifacts
- Want min/max to appear filtered
  - Round out the shape
- Solution: average 2 neighborhood’s min/max
YCoCg box

- Basic min/max is an AABB in RGB space
- Ideally use convex hull of neighborhood colors
  - Too expensive
- Orient box in luma direction
  - Luma has high local contrast
  - Chroma typically doesn’t
Clip instead of clamp

- Constrain to a blend of history and neighborhood average
- Clip line segment to box
- Colors don’t collect in box corners like clamping does
Basic min/max RGB clamp
Clipped to shaped YCoCg box
Translucency

• Translucency is a poor fit for temporal
  – Single history
  – Single velocity
• Ideally render translucency separate and composite
  – Can’t unjitter depth buffer to compare against
• Possible solution: 4xMSAA depth prepass
  – Alternate which sample to shade
Our translucency solution

- “Responsive AA” material flag
- Sets stencil when rendering translucency
- Temporal AA pass tests stencil and uses minimal feedback
  - Unfortunately need >0 feedback to prevent visible jittering
- Only useful for small particles like sparks
  - Neighborhood clamping handles the rest
Temporal AA is a firewall

Visibility samples

GBuffer  Lighting  Translucency

Spatial filters

TAA  Bloom  DOF  Post

Depth

Temporal AA
Temporal AA is a firewall

Visibility samples

Spatial filters

GBuffer ➔ Lighting ➔ Translucency ➔ TAA ➔ Bloom ➔ Post

DOF Setup ➔ TAA ➔ DOF

Temporal AA
Flickering

- Camera is static but some pixels flicker
- Missing subpixel feature’s history gets clamped
  - Often vertical or horizontal lines due to coherent jitter
- Clamping is an instantaneous impulse
- This leads to saw tooth waves which appear as flickering

Bright edge missing in one frame
Basic anti-flickering idea

- Bias towards impulse frame
  - Shrinks amplitude of wave
- Reduce exponential smoothing blend factor
  - Reduces recovery from impulses
- Only where needed
  - Overly blurry results if done everywhere

Bright edge missing in one frame
First anti-flickering attempt

- Store historical variance data in alpha channel
  - Remember clamping events
  - Reduce blend factor and recover over time

- Responsiveness issues
  - Can result in ghosting or blurring
  - Can bias towards aliased result
Our current anti-flickering solution

- Reduce blend factor when history is near clamping
  - Will happen after clamp events
  - Memory specific to event
  - Doesn’t require additional storage

- Not completely solved
  - Extremely difficult!
  - Impossible to solve multiple opposing clamps
Blurring: filter kernel

- Mipmap bias all textures
  - Incorrect derivatives for supersampling
- If low contrast then reduce filter kernel size
  - Technically aliases but looks fine
- Can add additional post sharpen filter
  - Mitchell 4.0 filter’s negative lobes are >1 pixel away
Blurring: reprojection diffusion

- Could use back and forth error compensation
  - Haven’t had good results
- Could store history at higher resolution
  - Really expensive
- When reprojecting outside pixel reduce filter size and feedback

[Valient14], [Yang09]
Noise filter

- Not its original purpose
  - Really nice side effect
- Used for SSR and SSAO
  - Stochastic sampling works pretty well
  - Doesn’t cost anything extra
  - Almost perfect mirror reflections with only 16 ray march steps
Many more potential applications

- Stochastic transparency
- Single sample anisotropic specular IBL
- Soft shadows
- Reduced steps for ray casting
  - Parallax occlusion mapping
  - Volumetric lighting
- Path tracing?
- VR?
Future directions

• Combine spatial and temporal

• Separate translucency
  – Visibility and shading sample disconnect

• Different jitter per pixel
  – Custom MSAA sample placement

• More complete motion vectors
  – Translucency
  – Motion estimation
Conclusion

- Temporal supersampling is production ready
  - High quality
  - High performance
- Needs a lot of perceptual tuning
Thanks

- Timothy Lottes, co-inventor
- Epic
  - Rendering team
Full source code available!

unrealengine.com

$19/mo + 5%

Epic is hiring!
References