SIGGRAPH2011 VANCOUVER

Filtering Approaches for Real-Time Anti-Aliasing

http://www.iryoku.com/aacourse/

SIGGRAPH2011 VANCOUVER

Filtering Approaches for Real-Time Anti-Aliasing

A Directionally Adaptive Edge Anti-Aliasing Filter

Jason Yang
jasonc.yang@amd.com

AMD

SIGGRAPH2011 VANCOUVER

Current Hardware Approach: Multisample Anti-Aliasing (MSAA)

No AA

Stair stepping

SIGGRAPH2011 VANCOUVER

Current Hardware Approach: Multisample Anti-Aliasing (MSAA)

- Color gradations limited by the number of hardware samples

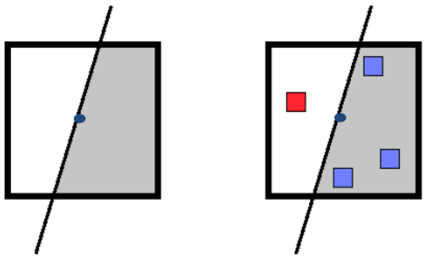
8x AA

In this example at most 8 gradations (not including no coverage).

SIGGRAPH2011 VANCOUVER

MSAA Overview

- Estimate primitive pixel coverage by testing at sample points
- Calculate a single color value per pixel per primitive (usually at the center of the pixel or at the centroid of the covered samples) and assign it to all covered samples



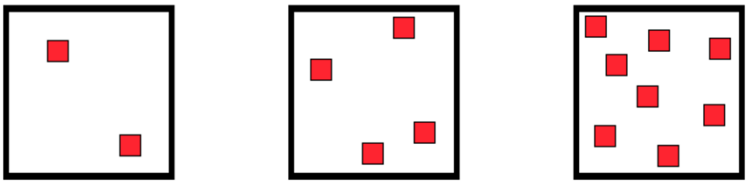
This pixel is considered ¾ filled by primitive even though it's about ½ filled

4 samples at most 4 levels of gradation (5 including non covered)

SIGGRAPH2011 VANCOUVER

Traditional MSAA

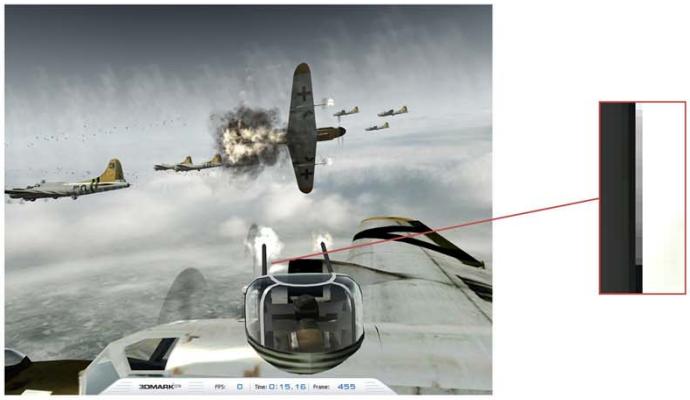
- Speeds up rendering by not calculating values for each sample separately
- Non-uniform sampling grid is used to improve pixel coverage estimate



SIGGRAPH2011 VANCOUVER

Traditional MSAA

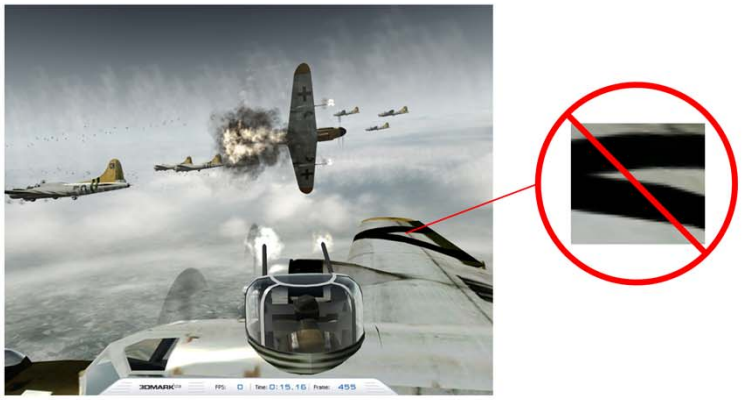
- Primary goal is anti-aliasing of primitive and Z edges



HWAA only deals with geometry edges, not textures.

Traditional MSAA

- Texture resampling performed elsewhere (mipmapping)
  - High frequency texture details (e.g. text) are lost in the MSAA color buffer and cannot be easily post-processed further



Motivation

- Can we use the GPU's shader processing power and flexibility for better edge anti-aliasing (AA)?
- Goal
  - Improve primitive edge appearance (vs. "standard" MSAA processing) using the same number of samples and better software filtering algorithms

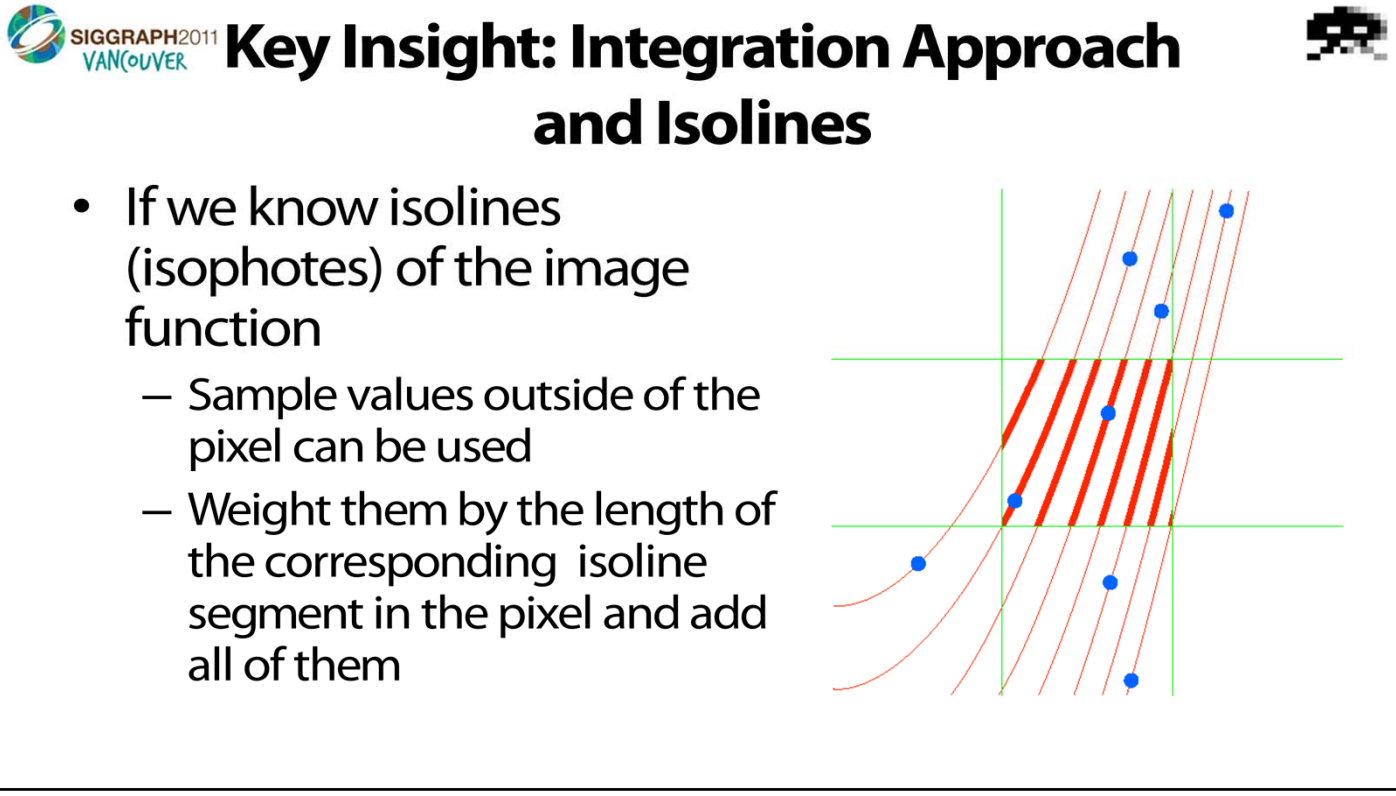
SIGGRAPH2011 VANCOUVER

Motivation

- Can we use the GPU's shader processing power and flexibility for better edge anti-aliasing (AA)?

Our New Filter

Edge-Detect Custom Filter AA – using 8x MSAA coverage samples can achieve up to 24 gradations of color.



If there's an edge there will be isolines, so can use samples outside of the pixel.

Isoline Evaluation in Three Channels

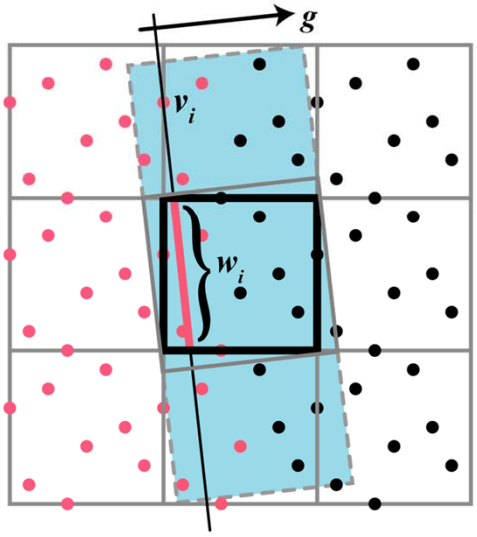
- Need to find isolines
  - But only in the case of low curvature (as this is the case near the actual primitive edges)
  - Then we can model them as a straight lines
- Fit a plane into three channels to compute a gradient estimate
  - Use linear approximation
  - Solve least squares problem
  - Please see [Iourcha HPG2009] for more details

Effectively doing a linear fit of the RGB samples – least squares problem.

Iourcha, K., Yang, J., and Pomianowski, A. 2009. A Directionally Adaptive Edge Anti-Aliasing Filter. *High Performance Graphics*.

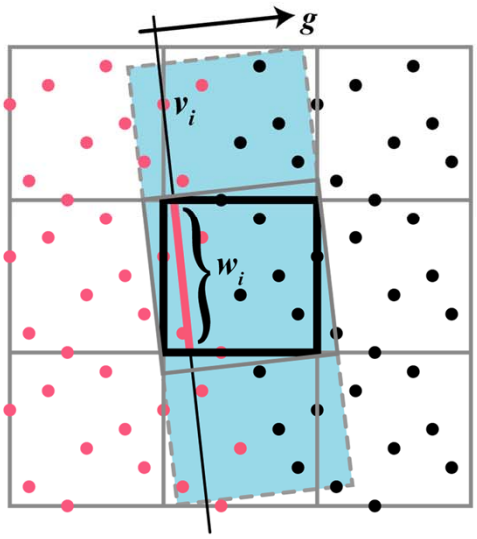
Filter Computation (Integration)

- Construct a square around the pixel, with two sides orthogonal to ***g***
- Extend the rectangle, in the direction orthogonal to ***g*** until it meets the 3x3 pixel boundary



Filter Computation (Integration)

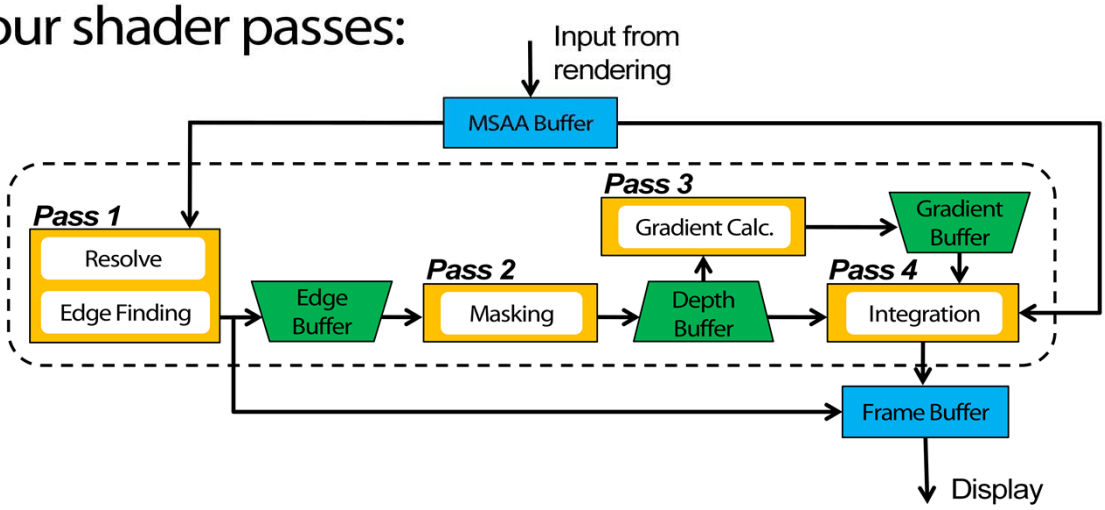
- For every sample $\boldsymbol{v_i}$, length of the segment of the line passing though $\boldsymbol{v_i}$ and orthogonal to $\boldsymbol{g}$ inscribed by the pixel is its weight w_i
- Calculate the weighted sum of all samples in the rectangle



SIGGRAPH2011 VANCOUVER

Bringing It All Together: Sample Implementation

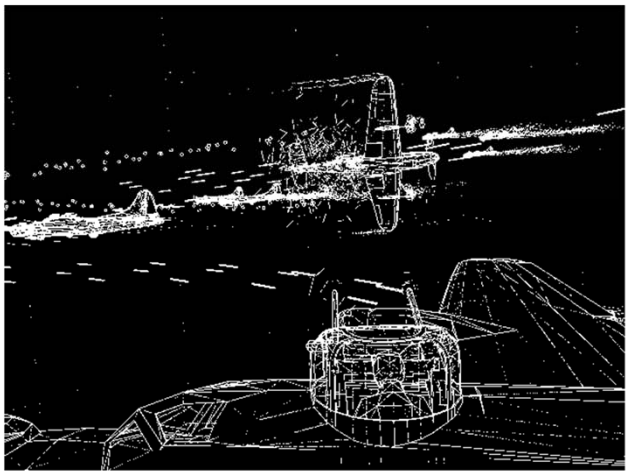
- Implemented using MS DirectX 10.1
- Four shader passes:



SIGGRAPH2011 VANCOUVER

Pass 1

- Identify edge pixels using the MSAA buffer.
  - Partial coverage == edge
  - High color contrast == edge of interest
- Seed the frame buffer by performing a standard resolve at each pixel



You know there's an edge of the samples are not fully covered.

SIGGRAPH2011 VANCOUVER

Pass 2

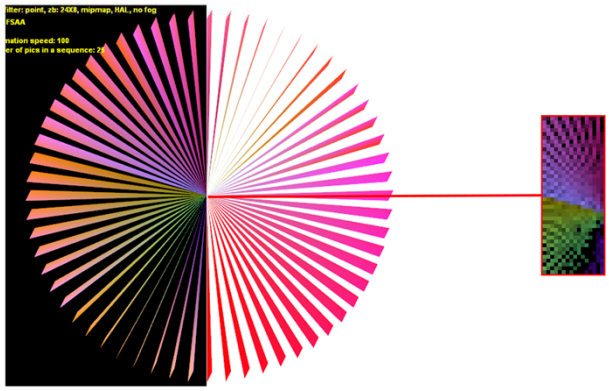
- Mask out candidate pixels using edge patterns
- Computed in shader vs. tables
  - We use 3x3 pixel patterns

Notice elimination of pixels from masking

SIGGRAPH2011 VANCOUVER

Masking

- For instance, if all 3x3 pixels are "edge" ones, there is no long dominating edge, and we do not want to smooth-out high-frequency in this case, etc.

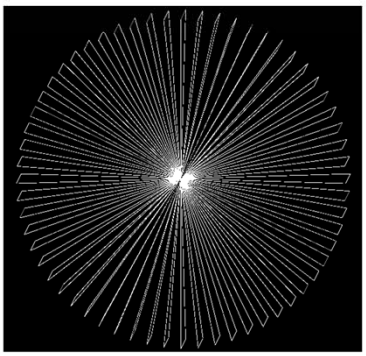


Aids in performance and look. Don't want to overfilter corners.

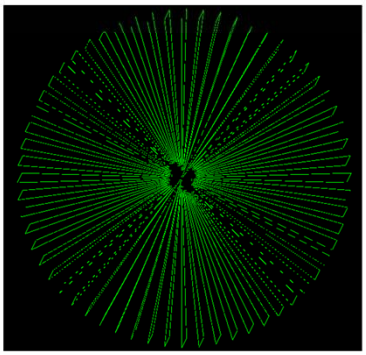
SIGGRAPH2011 VANCOUVER

Masking

- Can help avoid excessive corner smoothing
- Reduces processing time



Found Edges

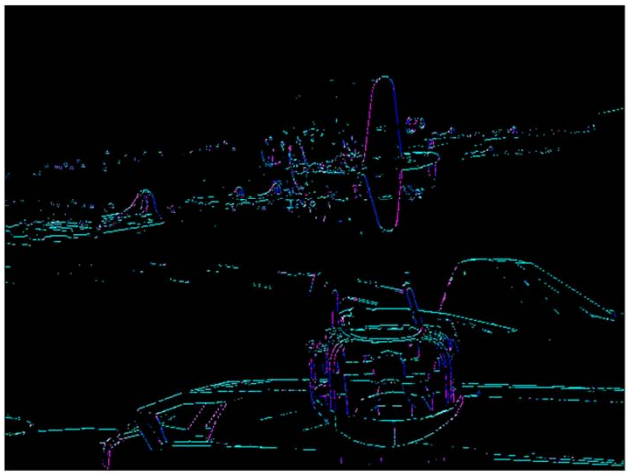


After Masking

SIGGRAPH2011 VANCOUVER

Pass 3

- Compute "gradients"



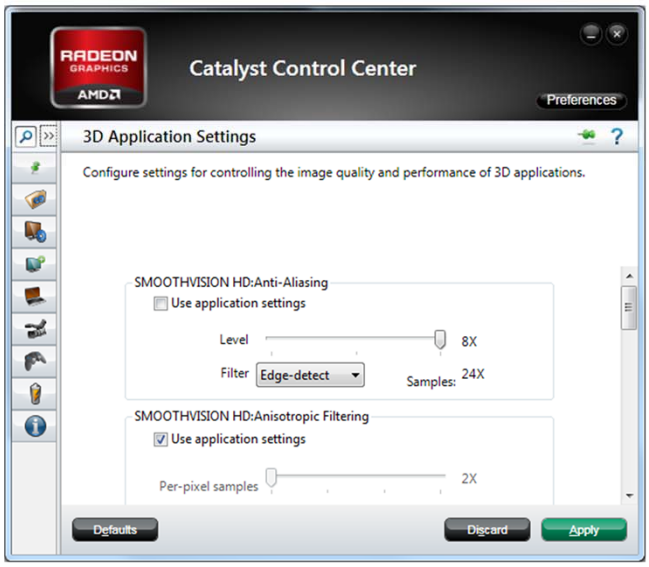
Do the least squares here.

Pass 4

- Calculate the final frame buffer color for the pixels from Pass 3 using the presented integration method with input samples from a 3x3 pixel neighborhood
- Integration and weights are computed in shader
- All other pixels were already filtered in Pass 1

Conclusion

- Runs in real-time
- No HW or SW modifications needed
- Go outside of the pixel
- Use GPU for more complex processing



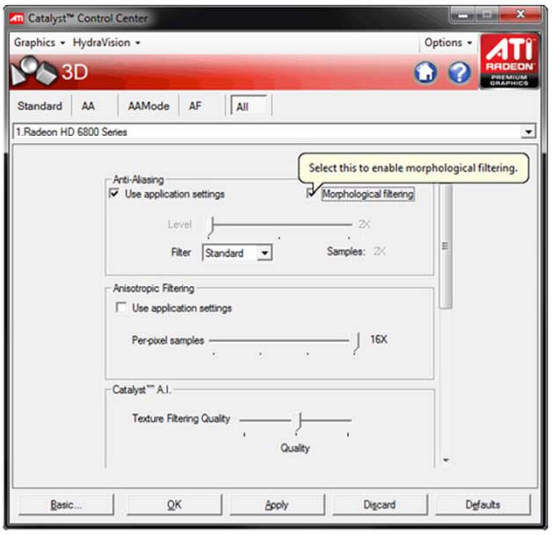
8x = 24x

4x = 12x

2x not very interesting

One Problem

- Still relies on HW samples
- Problem for deferred rendering
  - Lost geometry data
  - Morphological Anti-Aliasing…



Future?

- The world will move to new hardware sometime soon
- Interesting DX11 and GL4.x features
  - More flexible AA
  - Run shaders at sample rate
  - Compute shader – interesting data structures and better performance

References

- jasonc.yang@amd.com
- http://developer.amd.com/gpu_assets/AA-HPG09.pdf
- http://sites.amd.com/us/game/technology/Pages/morphological-aa.aspx

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2011 Advanced Micro Devices, Inc. All rights reserved.