Filtering Approaches for Real-Time Anti-Aliasing

http://www.iryoku.com/aacourse/
Final version with in-depth commentaries is available at
http://and.intercon.ru/
Directionally Localized Anti-Aliasing
Agenda

- Aliasing & Anti-Aliasing
- Alternative Solutions
- Exploration
- DLAA
- PS3 & X360
The Idea Is ...

Blur Edges Along Their Directions
Blurred. Done!
Aliasing

Signal Processing
Indistinguishable signals when sampled
Artifact of reconstruction

Graphics
Pixel "noise"
Edge jaggies
Anti-Aliasing I

-song Reduce Higher-Frequencies
-song Oversample And "Blur"
  -Temporal in audio
  -Spatial in optics
-song No Perfect Filter Exists
  -Sampling theory
  -Sharp (aliased) vs soft (anti-aliased)
Anti-Aliasing II

Texture
Mip-mapping

Shading
Specular, rim lighting
Avoid manually

Geometry Edges
Multi-sampling (MSAA)
Custom solutions
MSAA

- Good Quality
- Partial Super Sampling
  - At least depth
- Deferred Rendering Unfriendly
- Costly On Consoles
  - Directly and indirectly
Alternatives

- Screen-Space Filtering
  - Perception based
  - Hide jaggies
  - Morphological AA (MLAA)

- Temporal (Crysis 2, Halo)

- Edge-Based AA
MLAA

🔗 Morphological Anti-Aliasing (Intel)
  Reconstruct original geometry
  Re-blend neighbors

🔗 CPU Friendly
  The Saboteur
  GoW3 (4ms / 5 SPUs)
  XBox360 GPU (> 3.7 ms)
Edge-Based

🌐 XBox360 SDK Sample
  Render one-pixel wide polygons
  Texcoord as pixel coverage
  Re-blend neighbors
Could Not Use

瘁 MLAA
  Unstable
  Tough on X360
瘁 Edge-Based
  Extra GPU cost on PS3
瘁 Temporal
  Dynamic resolution adjustment in TFU2
  Motion vs resolution
"Ideal" AA Filter

- Multi-Platform
  - GPU, SPU
  - Reliable in production
- Temporally Stable
- Perception Based
  - Hide jaggies
- Good Quality For Low Cost
What If …

Create Pixel Coverage-Like Look
Fresnel Term Based

- $(N \cdot V)^n$
- Re-Blend
- Curved Surfaces Only
- Hard To Control
Depth Based Gradients

- Find Edge Gradients
  - Depth box-blur
  - Adjust levels locally
- Re-Blend
- Flat Surfaces
Depth Re-Sampling

- Render Alternative Depth
  Rotated 2\textsuperscript{nd} z-pre pass
  Or 4x MSAA for depth
- Compute Pixel Coverage
  Remap depth value
- Re-Blend
DLAA Prototyping

_suspend
Photoshop
  Layers vs Pixels
  Hard to do complex things
  Easy to implement IF works :)

_filter / Other / Custom
  Basic 5x5 convolution
  Blurs, Edges, etc...
DLAA Prototyping II

erà Blur Vertically
DLAA Prototyping III

- Blur Vertically
- Find Vertical Edges
DLAA Prototyping

- Blur Vertically
- Find Vertical Edges
- Build Edge Mask

\[
\text{saturate}(\ abs(x) \cdot a - b)\]

DLAA Prototyping

 agua

- Blur Vertically
- Find Vertical Edges
- Build Edge Mask
  \[ \text{saturation}(\text{abs}(x) \cdot a - b) \]
- Blend With Original Layer
- Same Horizontally
Short Edges Only
Two Cases

5-Pixel Wide

16-Pixel Wide
Long Edge Detection

Take High-Pass Mask
Long Edge Detection

- Take High-Pass Mask
- Blur
Long Edge Detection III

- Take High-Pass Mask
- Blur
- Adjust Contrast
Long Edge Detection

- Take High-Pass Mask
- Blur
- Adjust Contrast
- Apply Long-Edge Filter
  Where it's needed
Long Edge Filtering

Color Bleeding
Long Edge Filtering

- Color Bleeding
- Luminosity Blending Mode
- Blurred luminance As Target
  Find local pixel that matches it
Noise Level Estimation

Exclude Noisy Regions

Have long vertical and horizontal edges

$||H_{hf} - V_{hf}|| > \lambda$
Gradient Levels Comparison

- no AA
- MLAA
- DLAA
Visual Results
Reflections Anti-Aliasing

no reflection AA

reflection DLAA
Execution Results @ 720p

- XBox360: $2.2 \pm 0.2 \text{ ms}$
- PlayStation3: $1.6 \pm 0.3 \text{ ms}$ (5 SPUs)

Project Time

<table>
<thead>
<tr>
<th>Task</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>8 weeks (part time)</td>
</tr>
<tr>
<td>X360</td>
<td>2 weeks</td>
</tr>
<tr>
<td>PS3 (SPU)</td>
<td>&gt; 3 weeks</td>
</tr>
</tbody>
</table>
Implementation Strategies

Execution Time
- Reuse samples
- Reject as much work as possible
- Balance pipelines

Memory Usage
- Reuse textures and buffers
- Pack data by usage

Global Pipeline Optimizations
Work Rejection

難しい

Pre-Process
- Find long edge regions
- High-pass around long edges
- Resolve

Process
- Short edges
- Short and long edges (~10-20 %)
- Resolve
Long Edge Estimation I

 предлагается найти длинные осевые ребра непосредственно

 при пониженном разрешении (например, с помощью HDR-снижения)
Long Edge Estimation II

- Transfer Into Hi-Z (4x4 pixel blocks)
  - 4x MSAA trick
- Flip Hi-Z Test With Depth Trick
  - Using D3DHIZFUNC

![mask](image1)

![dilated Hi-Z](image2)
High-Pass Filter

- 5 Bi-Linear Samples
- Around Long Edges Only
- Store In Alpha
Short Edges

⚠️ Low And High-Pass Filters
Reuse vertical and horizontal samples

⚠️ Normalized Blending Coefficients
\[ t_h = \frac{\lambda \cdot L(\text{edge}_h) - \varepsilon}{L(\text{blur}_h)} \]
L(x) - intensity function

⚠️ Re-Blend
\[ c = \text{lerp}(c, \text{blur}_h, \text{saturate}(t_h)) \]
Long Edges

Sparse Sampling On GPU
  Reuse short samples
  Extra 4 bi-linear samples

Discard If Horizontal And Vertical
  [branch] based on blurred high-pass
Long Edges

缩小灰度图像，使之与模糊强度匹配的局部像素

\[
\text{blurred}_{\text{lum}} = \text{linterp} (\ X_{\text{lum}}, \ Y_{\text{lum}}, \ t )
\]

\[
\text{color} = \text{linterp} (\ X, \ Y, \ t )
\]
Long Edges III

Find Local Pixel That Matches Blurred Intensity

\[
\text{blurred}_{\text{lum}} = \text{lerp}( X_{\text{lum}}, Y_{\text{lum}}, t )
\]

\[
\text{color} = \text{lerp}( X, Y, t )
\]

Two Search Cases
Long Edges IV

Find Local Pixel That Matches Blurred Intensity
\[
\text{blurred}_{\text{lum}} = \text{lerp}(X_{\text{lum}}, Y_{\text{lum}}, t)
\]
\[
\text{color} = \text{lerp}(X, Y, t)
\]

Two Search Cases
Top and bottom neighbors

Re-Blend
Based on longEdgeMask
## Typical SPU Code

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Ctx</th>
<th>Locs</th>
<th>Data Pipeline</th>
<th>3rd Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>002</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>2219 86534, 537, 1</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>004</td>
<td>2410 54642, 543, 2</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td>2410 54651, 576, 19</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>006</td>
<td>2423 841, 542, 600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>012</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>013</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>016</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>018</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>019</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SPU Post Processing

- Software Pipelining
  - Hide latency
- Balance Even And Odd Instructions
- Stream Processing
- Tiled RSX Surfaces
  - 0.3 ms to copy from VRAM
  - Partial untiling with DMA
DLAA On SPUs

- No Need to Handle Overlaps
- Short Edges
  - Byte operations $\rightarrow$ 4 RGBA pixels / clk
  - $(1\ 2\ 1) = \text{AVGB}(\ \text{AVGB}(\ l,\ c),\ \text{AVGB}(\ c,\ r))$
  - $\|x - y\| = \text{ABSDB}(x, y)$
- Long Edges
  - $\text{blur}(x) = \sum f(x + dx)$
  - $\text{blur}(x + 1) = \text{blur}(x) - f(x - r) + f(x + 1 + r)$
DLAA On SPUs

Quick Luminance

\[ \text{SUMB} (G, R, G, B) \rightarrow 0.25 R + 0.5 G + 0.25 B \]

Quick Saturate

\[ \text{CFLTU} \ x, \ x, \ 32; \ \text{CUFLT} \ x, \ x, \ 32 \]

Quick Interpolation

\[ r = \text{lerp}(x, y, t) \]

\[ \text{FS} \ r, \ Y, \ X \quad \text{SHUFB} \ X, \ x, \ _, \ _ \]

\[ \text{FMA} \ r, \ t, \ r, \ X \quad \text{SHUFB} \ Y, \ y, \ _, \ _ \]
## Typical SPU Code

<table>
<thead>
<tr>
<th>Line</th>
<th>Labels</th>
<th>Even Pipeline</th>
<th>Odd Pipeline</th>
<th>Gdb Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>004</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>012</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>013</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>016</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>018</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The table and diagram are visual representations of code and pipeline information, but the specifics are not readable from the image.
# Efficient SPU Code

| Game Codes | Game Iden
tation | Game Name | Game Type | Game Rating |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>Hello World</td>
<td>World</td>
<td>World</td>
<td>World</td>
</tr>
<tr>
<td>004</td>
<td>Game 1</td>
<td>Game 1</td>
<td>Game 1</td>
<td>Game 1</td>
</tr>
<tr>
<td>005</td>
<td>Game 2</td>
<td>Game 2</td>
<td>Game 2</td>
<td>Game 2</td>
</tr>
<tr>
<td>006</td>
<td>Game 3</td>
<td>Game 3</td>
<td>Game 3</td>
<td>Game 3</td>
</tr>
<tr>
<td>007</td>
<td>Game 4</td>
<td>Game 4</td>
<td>Game 4</td>
<td>Game 4</td>
</tr>
<tr>
<td>008</td>
<td>Game 5</td>
<td>Game 5</td>
<td>Game 5</td>
<td>Game 5</td>
</tr>
</tbody>
</table>

*Note: This table is a simplified representation of the actual content in the document.*

---

**Table:**

<table>
<thead>
<tr>
<th>Category</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game 1</td>
<td>123</td>
</tr>
<tr>
<td>Game 2</td>
<td>456</td>
</tr>
<tr>
<td>Game 3</td>
<td>789</td>
</tr>
<tr>
<td>Game 4</td>
<td>012</td>
</tr>
<tr>
<td>Game 5</td>
<td>345</td>
</tr>
</tbody>
</table>

**Header:**

- **Category:** A list of categories for the values.
- **Values:** Actual values corresponding to each category.

---

**Figure:**

A figure showing a graph or chart related to the data presented in the table.
Conclusion

DLAA

- Xbox360: $2.2 \pm 0.2 \text{ ms}$
- PlayStation3: $1.6 \pm 0.3 \text{ ms}$ (5 SPUs)

End Of Console Life Cycle
- Every millisecond counts
- Tricks are inevitable
- Different solutions & different thinking
Acknowledgments

Szymon Swistun
Ruslan Abdikeev
Axel Wefers
Jerome Scholler
Tom Madams

Anti-Aliasing Community
Thank You
Thank You

Questions ?

dandreev@LucasArts.com