In This Tutorial...

→Introduction
→Transparency effects and applications
→Anti-Aliasing impact in the final image
→Why combine Transparency with Anti-Aliasing?

In This Tutorial...

→Transparency
→Understanding and solving Transparency
→High-quality and high-performance Transparency Techniques

In This Tutorial...

→Anti-Aliasing
→Understanding and solving the Aliasing problem
→High-quality and high-performance Anti-Aliasing Techniques

In This Tutorial.

→Applications
→How to combine Transparency and Anti-Aliasing
→Applications for Transparency with Anti-Aliasing
→Conclusions
→Future Work

Outline

→Introduction

→Transparency Problem and Basic Approach
→Raster-based Transparency Techniques

→AA Problem and Basic Approaches
→AA Techniques

→Applications
→Conclusions
Stochastic Transparency, NVIDIA 2010
Effects

G-Buffer, 2007
Translucency

Jimenez, Siggraph Course 2011
Aliased Anti-Aliased

Aliased Anti-Aliased 4xMSAA
Transparency + Anti-Aliasing

- Both refer to visibility computation
  - Fragment coverage over pixel area
  - Fragment visibility along depth
- Could both be combined into a single visibility representation?

Outline

- Introduction
- Transparency Problem and Basic Approach
- Raster-based Transparency Techniques
  - AA Problem and Basic Approaches
  - AA Techniques
- Applications
- Conclusions
Introduction

→ Transparency Problem and Basic Approach
→ Raster-based Transparency Techniques
→ AA Problem and Basic Approaches
→ AA Techniques
→ Applications
→ Conclusions

Outline

→ Transparency Problem and Basic Approach
→ Transparency Fundamentals
→ Transparency in Practice
→ Primitive Sorting
→ Raster-based Transparency Techniques
→ Buffer-Based
→ Depth Peeling
→ Sort Independent
→ Stochastic

Outline

→ Transparency Fundamentals

Transparency Fundamentals

• How do we perceive transparency?

• How we render transparency
  • How much light from the fragment arrives in the eye?

• Blending equations [Porter and Duff, 1984]
  – Back-to-Front
    \[ C'_{\text{acc}} = (1 - \alpha_i)C_{\text{acc}} + \alpha_iC_i \]
  – A new fragment reduces visibility from the accumulated color

\[ C_{\text{acc}} \quad \text{Accumulated color} \]
\[ C_i \quad \text{Color from fragment } i \]
\[ a_{\text{acc}} \quad \text{Accumulated opacity} \]
\[ \alpha_i \quad \text{Opacity from fragment } i \]

Transparency Fundamentals

• Opacity (\(\alpha\)):
  – how much light the surface transmits?

• Visibility:
  – how much of the transmitted light the eye can see?

Transparency Fundamentals

• How do we perceive transparency?

• How much light from the fragment arrives in the eye?

• Blending equations [Porter and Duff, 1984]
  – Back-to-Front
    \[ C'_{\text{acc}} = (1 - \alpha_i)C_{\text{acc}} + \alpha_iC_i \]
  – A new fragment reduces visibility from the accumulated color

\[ C_{\text{acc}} \quad \text{Accumulated color} \]
\[ C_i \quad \text{Color from fragment } i \]
\[ a_{\text{acc}} \quad \text{Accumulated opacity} \]
\[ \alpha_i \quad \text{Opacity from fragment } i \]
**Transparency Fundamentals**

- Blending equations [Porter and Duff, 1984]
  - Front-to-Back
    \[ C_{\text{acc}}' = C_{\text{acc}}(1 - \alpha_{\text{acc}}) + \alpha_{\text{acc}}C_i \]
  - A new fragment has its visibility reduces by the accumulated opacity
    \[ C_{\text{acc}} \quad \text{Accumulated color} \]
    \[ C_i \quad \text{Color from fragment } i \]
    \[ \alpha_{\text{acc}} \quad \text{Accumulated opacity} \]
    \[ \alpha_i \quad \text{Opacity from fragment } i \]

- Back-to-Front blending
  \[ C_{\text{acc}}' = (1 - \alpha_i)C_{\text{acc}} + \alpha_iC_i \]

- (0.5, 0, 0, 0.5) = (1 - 0.5)(0,0,0,0) + 0.5Red
  \[ C_{\text{acc}}' = (1 - \alpha_i)C_{\text{acc}} + \alpha_iC_i \]

- (0.25, 0, 0.5, 0.75) = (1 - 0.5)(0.5, 0, 0, 0.5) + 0.5Blue
  \[ C_{\text{acc}}' = (1 - \alpha_i)C_{\text{acc}} + \alpha_iC_i \]

- (0.125, 0.5, 0.25, 0.875) = (1 - 0.5)(0.25, 0.5, 0.5, 0.75) + 0.5Green
  \[ C_{\text{acc}}' = (1 - \alpha_i)C_{\text{acc}} + \alpha_iC_i \]
**Transparency Fundamentals**

- Front-to-back blending

\[
(0.0, 0.5, 0.0, 0.0) + (1 - 0.5) \times \text{Green} = \text{Green} \\
C'_{\text{acc}} = C_{\text{acc}} + (1 - \alpha_{\text{acc}}) \alpha_i C_i
\]

\[
(0.0, 0.5, 0.25, 0.0, 0.75) + (1 - 0.75) \times \text{Red} = \text{Red} \\
C'_{\text{acc}} = C_{\text{acc}} + (1 - \alpha_{\text{acc}}) \alpha_i C_i
\]

- Out-of-order fragments
  - Wrong visibility computation

\[
(0.5, 0.0, 0.0, 0.0, 0.5) + (1 - 0.5) \times \text{Red} = \text{Red} \\
C'_{\text{acc}} = C_{\text{acc}} + (1 - \alpha_{\text{acc}}) \alpha_i C_i
\]

\[
(0.5, 0.25, 0.0, 0.5) + (1 - 0.5) \times \text{Green} = \text{Green} \\
C'_{\text{acc}} = C_{\text{acc}} + (1 - \alpha_{\text{acc}}) \alpha_i C_i
\]
Transparency Fundamentals

- Summary
  - Back-to-Front
  - Front-to-Back
  - Out-of-Order
    - FTB blending
    - BTF blending

Outline

→ Transparency Problem and Basic Approach
→ Transparency Fundamentals
→ Transparency in Practice
→ Primitive Sorting
→ Raster-based Transparency Techniques
  → Buffer-Based
  → Depth Peeling
  → Sort Independent
  → Stochastic

Transparency in Practice

- Depth Test Enabled
  - Incorrect fragment discard

Correct Transparency

- Depth Test Disabled
- Depth-Sorted-Order rendering
  - Primitive sorting
    - Assuming no interpenetration
  - Correct visibility results

Primitive Sorting

X

All fragments have the same depth-sorted order from their primitive.

Fragment Sorting

The fragments are depth-sorted individually.
Outline

→ Transparency Problem and Basic Approach
→ Transparency Fundaments
→ Transparency in Practice
→ Primitive Sorting

→ Raster-based Transparency Techniques
→ Buffer-Based
→ Depth Peeling
→ Sort Independent
→ Stochastic

Primitive Sorting

• Interpenetration Issue

Primitive Sorting

• Interpenetration Issue

Primitive Sorting

• Interpenetration
• Spatial data structures
  – BSP
  – Kd-Tree
  – Octree
  – ...
Outline

→ Transparency Problem and Basic Approach
→ Transparency Fundamentals
→ Transparency in Practice
→ Primitive Sorting

→ Raster-based Transparency Techniques
→ Buffer-Based
→ Depth Peeling
→ Sort Independent
→ Stochastic

Buffer-based

1. Store fragments
2. Sort fragments
3. Blend fragments

\[ C'_{\text{acc}} = C_{\text{acc}} + (1 - a_{\text{acc}})C_{\text{frag}}a_{\text{frag}} \]

Unbounded Buffers

- Store all fragments

→ A-buffer

True Linked Lists

Carpenter, 1984
Unbounded Buffers
• Store all fragments
  → A-buffer
  True Linked Lists
  → PPLL
  GPU Linked Lists
  Emulation

Fixed-Size Buffers
→ 3D buffers
Stores all attributes

→ Z^3
→ k-buffer

Buffer-Based Summary
• Store all fragments
  – Precise compositing of fragments
  – Unbounded amounts of memory

• Fixed storage
  – Faster but approximate
    • Not all fragments considered in each pixel
    • Approximate visibility contributions

Outline
→ Transparency Problem and Basic Approach
→ Transparency Fundamentals
→ Transparency in Practice
→ Primitive Sorting
→ Raster-based Transparency Techniques
→ Buffer-Based
  → Depth Peeling
  → Sort Independent
  → Stochastic
**Depth Peeling**

- Store the farthest unblended layer — 1st geometry pass

<table>
<thead>
<tr>
<th>Geometric Primitives</th>
<th>Eye</th>
<th>Depth</th>
<th>Transparent</th>
<th>Opaque</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-Blend with the opaque layer

**Depth Peeling**

- Store the farthest unblended layer — 2nd geometry pass

<table>
<thead>
<tr>
<th>Geometric Primitives</th>
<th>Eye</th>
<th>Depth</th>
<th>Transparent</th>
<th>Opaque</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Depth Peeling**

- Compose layers over several geometry steps — 2nd geometry pass

<table>
<thead>
<tr>
<th>Geometric Primitives</th>
<th>Eye</th>
<th>Depth</th>
<th>Transparent</th>
<th>Opaque</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Depth Peeling**

- Compose layers over several geometry steps
  - 2nd geometry pass

<table>
<thead>
<tr>
<th>Geometric Primitives</th>
<th>Eye</th>
<th>Depth</th>
<th>Transparent</th>
<th>Opaque</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Blend with the opaque layer

End of geometry pass

---

**Depth Peeling**

- Compose layers over several geometry steps
  - 3rd geometry pass

<table>
<thead>
<tr>
<th>Geometric Primitives</th>
<th>Eye</th>
<th>Depth</th>
<th>Transparent</th>
<th>Opaque</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Blend with the opaque layer

End of geometry pass
**Dual Depth Peeling**

- Compose two layers per geometry step
  - Front-most and back-most

**Depth-Peeling Summary**

- Fixed and small amount of memory
- Slow geometry multipass

**Outline**

→ Transparency Problem and Basic Approach
→ Transparency Fundamentals
→ Transparency in Practice
→ Primitive Sorting
→ Raster-based Transparency Techniques
  → Buffer-Based
  → Depth Peeling
→ Sort Independent
→ Stochastic

**Sort-Independent Transparency**

- 2 Geometry passes
  - Alpha accumulation
  - Good approximation for low alpha
  - Inaccurate for high alpha

**Sort-Independent Transparency**

- 2 Geometry passes
  - Alpha accumulation
  - Good approximation for low alpha
  - Inaccurate for high alpha
Outline

→ Transparency Problem and Basic Approach
→ Transparency Fundamentals
→ Transparency in Practice
→ Primitive Sorting
→ Raster-based Transparency Techniques
→ Buffer-Based
→ Depth Peeling
→ Sort Independent
→ Stochastic

Stochastic Transparency

• Super resolution screen-door transparency
  — Alpha and visibility translate to sample individual probability of coverage

Stochastic Transparency
Stochastic Transparency

- MSAA samples to represent visibility
  - Pre-passes to estimate visibility
- Small number of samples per pixel leads to noise

Outline

→ Introduction
→ Transparency Problem and Basic Approach
→ Raster-based Transparency Techniques
→ AA Problem and Basic Approaches
→ AA Techniques
→ Applications
→ Conclusions
The Aliasing Problem

- I want to draw a shape...
  
  \[
  \text{Thin primitives}
  \]

- But what appears doesn’t look like my shape!
  
  \[
  \text{Single sample per pixel result}
  \]

The Aliasing Problem

- Why doesn’t it look like my shape?
  
  \[
  \text{Thin primitive}
  \]

- Thin primitives
  
  \[
  \text{Single sample per pixel result}
  \]
The Aliasing Problem

- Single-Sample Problem: Temporal Aliasing

![Temporal Aliasing Example](image)

The Aliasing Problem

- If only I could estimate the pixels coverage...

![Aliasing Coverage Example](image)

Outline

- AA Problem and Basic Approaches
  - The Aliasing Problem
  - Single Sample Problem
- Super Sampling
  - Sample Distribution
- AA Techniques
  - FSAA
  - IAA
  - GBAA

Single Sample Problem

- Rendering into a normal resolution

![Center Sampling Example](image)

Single Sample Problem

- Rendering into a normal resolution
  - Center sampling
  - Aliased result

![Aliased Result Example](image)
Outline

→ AA Problem and Basic Approaches
  → The Aliasing Problem
  → Single Sample Problem
→ Super Sampling
→ Sample Distribution
→ AA Techniques
  → FSAA
  → IAA
  → GBAA

Super Sampling

• Rendering into 4x-higher resolution
  – Center sampling

Super Sampling

• Rendering into 4x-higher resolution
  – Center sampling high-resolution result

Super Sampling

• Rendering into 4x-higher resolution
  – 2x2 box filter
  – Down-filtered result
Central Sampling x Super Sampling

* Thin Primitives
* Rasterization process

**Result in the display**

Central Sampling x Super Sampling

* Temporal Aliasing

<table>
<thead>
<tr>
<th>Frame</th>
<th>Center sampling</th>
<th>4x super sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>+</td>
<td>0%</td>
</tr>
<tr>
<td>i+1</td>
<td>+</td>
<td>100%</td>
</tr>
</tbody>
</table>

Outline

- AA Problem and Basic Approaches
- The Aliasing Problem
- Single Sample Problem
- Super Sampling
- Sample Distribution

AA Techniques
- FSAA
- IAA
- GBAA

Sample Distribution

* Critical Angle

Critical angle:
- Nearly horizontal and nearly vertical edges
- Critical angle rotated:
  - Less disturbing

Sample Distribution

* Regularity

Good = pixel area well covered by samples
Arbitrary = pixel coverage by samples may be good may be not
Cheap = low computational cost
Costly = high computational cost
Outline

→ AA Problem and Basic Approaches
  → The Aliasing Problem
  → Single Sample Problem
  → Super Sampling
  → Sample Distribution
→ AA Techniques
  → FSAA
  → IAA
  → GBAA

Full-Scene Anti-Aliasing

• Input: 3D scene
  → No selection
  → Entire scene is processed
  → More than one sample per pixel
  → Buffers
  → Filtering
  → Downsampling
• Output: anti-aliased image

Super Sampling Anti-Aliasing

• Per-sample color evaluation

  All attributes are computed individually per sample

Multi-Sampling Anti-Aliasing

• Single color evaluation
  → Once per pixel

  Shaded color computed at the pixel center
  Color replicated to the samples. Depth is computed individually per sample

Coverage-Sample Anti-Aliasing

• Also EQAA
• Single color evaluation
• Coverage-only and regular samples
  Shaded color computed at the pixel center
  Color replicated to the samples. Depth is computed individually per sample
  Boolean samples encoding coverage. No attribute stored

Directionally-Adaptive Edge AA

• Single color evaluation
  Shaded color computed at the pixel center
  Isolines approximation
  Color replicated to the samples. Depth is computed individually per sample
**Subpixel Reconstruction AA**

- Single color evaluation
- Samples reconstruction

![Diagram of subpixel reconstruction AA]

**A-Buffer**

- Single color evaluation
- Boolean coverage mask
- Max and min depth values

![Diagram of A-Buffer]

**FSAA Techniques Summary**

- SSAA
- MSAA
- CSAA
- DAA
- SRAA

**FSAA Summary**

- **Pros**
  - High image quality
  - Temporal stability
  - Subpixel anti-aliasing

- **Cons**
  - High memory consumption

**Outline**

→ AA Problem and Basic Approaches
  → The Aliasing Problem
  → Single Sample Problem
  → Super Sampling
  → Sample Distribution

→ AA Techniques
  → FSAA
  → IA AA
  → GBAA

**Image Post-Processing AA**

- **Input:** digital image
  - Search for aliased pixels
  - Selection of aliased pixels
  - Intelligent blur of aliased pixels

- **Output:** anti-aliased image
Morphological Anti-Aliasing

- Search for aliased patterns
  - Create list of found patterns
- Rebuild approximated geometry
- Blur

Practical MLAA (Jimenez's)

- Parallel MLAA
  - MLAA in real time using GPU filtering
    - Search for edge distances with filtering
      - Both pixels have edge
      - Start

Fast approxXimate AA

- Search for aliased pixels
  - Local luma contrast
    - 4-neighbors
- Directional blur filter
  - Perpendicular to luma gradient

Directionally Localized AA

- Photoshop prototype
  - Filtering sequence

Image Post-Processing Summary

- Pros
  - Very fast
  - Low resource consumption
- Cons
  - Does not adequately handle subpixels
  - Temporal instability

Outline

- AA Problem and Basic Approaches
  - The Aliasing Problem
  - Single Sample Problem
  - Super Sampling
  - Sample Distribution
- AA Techniques
  - FSAA
  - IAA
  - GBAA
**Geometric Anti-Aliasing**

- Input: 3D scene
  - Extra information to edge pixels
  - Post-processing blur
- Output: anti-aliased image

**Distance-to-Edge Anti-Aliasing**

- Pixel distance to triangle edge
  - Distance approximated by color
    - Each vertex receives one channel setup
    - Interpolation
  - Distance-guided blur
    - Only if smaller than one pixel

**Geometry-Buffer Anti-Aliasing**

- Pixel distance to triangle edge
  - Computed analytically
    - Encoded into RGB texture
    - Major direction
- Distance-guided blur
  - Only if less than half pixel

**Geometric Anti-Aliasing Summary**

- Pros
  - Image quality
  - Temporal stability
- Cons
  - Does not resolve thin primitives

**Outline**

- Introduction
- Transparency Problem and Basic Approach
  - Raster-based Transparency Techniques
- AA Problem and Basic Approaches
  - AA Techniques
    - FSAA → Image Post-Processing → Geometric AA
- Applications
- Conclusions
Outline

→Introduction
→Transparency Problem and Basic Approach
→Raster-based Transparency Techniques
→AA Problem and Basic Approaches
→AA Techniques
→Applications
→Conclusions

AA using Hardware for Transparency

• Coverage is used to weight alpha
  – Order-dependent
  – Less memory
  
  

  MSAA
  
  AA-OIT

• Combining them with Deferred Shading is even more challenging

Transparency using Hardware for AA

• Multisampled texture used to store fragments
  – Post-processed sorting and blending
  – Uniform alpha
    • Channel alpha used to store depth

AA for Transparent Scenes

• FSAA
  – High quality
  – High memory consumption
• GAA
  – High quality
  – Geometric borders only
• IAA
  – Low quality
  – Low memory consumption

Conclusions

• OfT and AA are still open problems
  – Combining them is challenging
  – Combining them with Deferred Shading is even more challenging

• There is no default solution
  – Application dependent
    • Quality x Performance x Memory
Future...

• OIT and AA with Deferred Shading
  • Deferred Shading
    – G-buffers
    – Shading as a post-processing stage
  • +OIT
    – Overburden G-buffers
  • +AA
    – Overburden G-buffers
    – Decoupling of sample from fragment
      • MSAA impractical