Z\(^3\): An Economical Hardware Technique for High-Quality Antialiasing and Order-Independent Transparency

Norm Jouppi - Compaq WRL
Chun-Fa Chang - UNC Chapel Hill
Order-Dependent Transparency

Pixel color: [Red]
Order-Dependent Transparency

Viewer → Pixel center

Pixel color:
Order-Dependent Transparency

![Diagram of viewer and pixel colors]

Viewer → Pixel center

Pixel color: □
Order-Independent (O-I) Transparency

- Render transparent & opaque objects in any order
  - “Don’t want to sort primitives”
  - Can’t sort subpixels
- Useful with textures (e.g., trees) and compositing
- David Kirk, NVIDIA, 1998 Eurographics/SIGGRAPH keynote: Order-independent transparency is an “unsolved problem and opportunity for high-quality high-performance 3D graphics on a PC”
A-buffer Methods for O-I Transparency

- Developed by Carpenter as a software technique and enhanced by others
- Object-based algorithm:
  - Keep a list of each visible sub-pixel fragment
  - Compute final pixel from fragment list
- Implemented by some in hardware
  - Dynamic storage allocation in hardware!
- Doesn’t correctly antialias interpenetrating objects
How Can Aliasing Be Eliminated?

- Five main hardware methods:
  - Higher resolution monitors (impractical, expensive)
  - Blending (only works for lines)
  - Accumulation buffer (slow)
  - A-buffer methods
  - Supersampling
Supersampling

- Transparent to the user:
  - Render the image at higher resolution
  - Filter down to screen resolution
  - Requires more memory and time
  - Need at least 4X resolution in x and y to look significantly better

- Correctly handles interpenetrating opaque objects
**Sparse Supersampling**

- Have larger sample array, but sparsely populated
  - For various $n$, consider $n$ samples on $nxn$ grid
- Can give more intensity steps with fewer samples
  - Better images
  - Less time and storage
8x8 Sparse Supersampling

- One sample per row and column
- Near horizontal and vertical edges look better
  - 9 intensity steps with 8 samples
- Some other angles aren’t as good
  - Diagonals already look better due to screen and eyes
- Looks almost as good as full 8x8
- Used in SGI’s Infinite Reality
Problems with Sparse Supersampling

- Doesn’t support order-independent transparency
  - Sorting doesn’t fix interpenetrating transparency
- Uses too much memory capacity and bandwidth
- Wastes resources: Much of the sample point data is similar to data for other sample points
  - Often only a few objects are visible within a pixel
  - Use a single color for an entire fragment
  - Use a more compact Z representation
Compact Z Representations

- 4 options:
  - Single Z at pixel center
  - Zmin and Zmax
  - Centroid adjusted Z
  - Z, Zdx, and Zdy

- WARNING:
  Correct subpixel visibility calculations are more important than “correct” antialiasing of subpixels
Visibility Errors

[Diagrams showing eye pixel centers and A, B points with Zmin and Zmax labels]
The Z3 Data Structure

![Diagram of the Z3 Data Structure]
Z^3 Algorithm

- 4 stages for processing a new fragment:
  - Occlusion check
  - If pass, insert fragment in Z order
  - Compute final pixel color
  - If too many fragments to fit, merge two fragments
Z³ Block Diagram

Incoming fragment

Occlusion check (including subpixel Z computation)

New fragment insertion

Pixel color computation

Fragment compression

Stored fragments

Frame buffer memory

Pixel color

Compressed fragments
Occlusion Check Stage

- Read in existing fragments sorted by center Z
- Expand Z gradients and Z values into per-sample Z values
- Standard occlusion test per sample
- Totally occluded fragments are deleted
**Fragment Insertion Stage**

- If not occluded, insert new fragment in proper place in sorted fragment list
Final Color Computation

- Only if new fragment is not fully occluded
- Compute per-sample ordering of fragment pairs
  - Produce a “swap vector” between fragments
- Accumulate fragments for each sample
- Average each sample into the final pixel color
Fragment Merging

- Only if we have too many fragments
- Compute Z distance between pairs of fragments
- Merge closest pair of fragments
- Merges fragments from the same surface first, etc.
Limited Precision Z-slopes

- Compact 8-bit format:
  - 1-bit sign
  - 5-bit exponent
    - Can represent $2^{31}$ to 0
    - Covers entire range of 24-bit Z value
  - 3-bit mantissa (one bit is hidden)
- With rounding max error is 1 part in 9
  - .1001 is rounded up to .101
  - .1000111… is rounded down to .100
Maximum Potential Errors
Due to Compact Slope Format

Note: Tilted surface must be drawn first to produce error
Reduced Precision Slopes:  
The Bottom Line

- May misplace edges by a sample point
  - Pixel could be 15/16 instead of 14/16
- Traditional 4X sparse supersampling is worse
  - Pixel limited to 3/4 or 4/4 instead of 14/16
Pixel Complexity of Opaque Test Case
Complexity of Transparent Test Case
Original Aliased Image
2x2 “Antialiasing”
Low-end SGI Infinite Reality
High-end Infinite Reality
16x16 Sparse Z3
Sum of Squares of Per-Pixel Errors for Opaque Test Case vs. # of Fragments

![Graph showing the sum of squares of per-pixel errors for different fragment counts. The graph compares Full, Sparse, 4X Z3, 8X Z3, 16X Z3, and 32X Z3 configurations. The x-axis represents the number of fragments stored at each pixel, while the y-axis represents the error, scaled in scientific notation. The graph shows a decrease in error as the number of fragments increases.]
Video

Also available at
Errors for Transparent Test Case vs. # of Fragments

![Graph showing errors for different fragment counts for 4X Z3, 8X Z3, and 16X Z3. The y-axis represents error, and the x-axis represents the number of fragments stored at each pixel. The graph shows a steady increase in error with an increase in the number of fragments.](image)
# Additional Memory Requirements

<table>
<thead>
<tr>
<th>Screen size</th>
<th>2-fragment pixels</th>
<th>4-fragment pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024x768</td>
<td>14MB</td>
<td>28MB</td>
</tr>
<tr>
<td>1280x1024</td>
<td>24MB</td>
<td>48MB</td>
</tr>
<tr>
<td>1600x1200</td>
<td>35MB</td>
<td>70MB</td>
</tr>
<tr>
<td>1920x1200</td>
<td>41MB</td>
<td>82MB</td>
</tr>
</tbody>
</table>

Note: Memory is currently about 75¢ a MB
Memory gets 4X cheaper every 3 years
Conclusions

- Z³ provides high-quality antialiasing and order-independent transparency at small additional cost
- Easy to implement due to fixed per-pixel storage
- Large numbers of sample points (e.g., 16) feasible
- Correctly antialiases interpenetrating surfaces, even if they are transparent (unlike A-buffer)
- Z³’s smaller memory requirements mean higher performance for a given memory bandwidth