Toward Accurate and Efficient Order-Independent Transparency

Overview
Correctly rendering multi-layered transparent geometry requires accumulating contributions from multiple fragments per pixel. Dynamic A-buffers (e.g., Yang et al’s [2010] per-pixel linked lists) achieve this by storing and sorting fragments on-the-fly. We introduce an improvement to recent GPU-based interactive A-buffer techniques: we decouple visibility and shading to reduce memory demands of multi-fragment rendering.

The Compact A-buffer
Existing interactive A-buffers store shading and visibility inside fragment lists, saving per primitive shading data repeatedly for multiple pixels. Decoupling storage of primitive and fragment data in our new compact A-buffer significantly reduces memory overhead. This approach resembles the decoupling proposed by Liktor and Dachschafer’s [2012] compact G-buffer.

Performance and Memory Usage
When primitive count exceeds fragment count, our compact A-buffer has a larger memory footprint. However, our compact A-buffer scales more efficiently as fragment count increases. This scaling comes at the cost of an additional layer of indirection while accessing shading data, increasing shader execution time.

Non-Optical Rendering
Accurate and efficient OIT has applications to non-optical rendering such as ballistic simulations. Particularly, optical transparency computes the light absorbed as photons pass through the environment, whereas ballistic simulation computes the energy absorbed as projectiles pass through an object [Butler and Stephens 2007].

Future Work
Our future work may examine the performance and accuracy tradeoffs between exact and approximate raster-based transparency for non-optical rendering applications. We may also compare order-independent transparency algorithms with ray tracing for ballistic simulations.

References