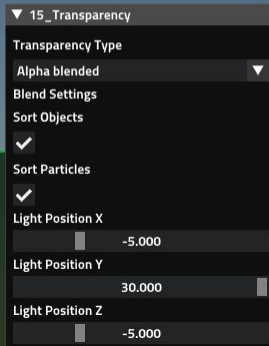




Transparency unit test

CPU Time: 0.546000 ms
GPU 0.421627 ms
Frame Time: 0.840000 ms
-----GPU Times-----
Draw Skybox - 0.107685 ms
Render opaque geometry - 0.071631 ms
Render transparent geometry - 0.230108 ms



Alpha blended transparency

This sample uses basic alpha blending. It sorts objects based on the distance from the camera and renders them back to front. It does the same for the particle systems and the particles. It doesn't sort the particle systems against the other objects as they're rendered using a different draw call.

Pros:

- Cheapest of the techniques evaluated.
- Works on all hardware.

Cons:

- All transparent geometry needs to be sorted otherwise you get hideous artifacts (Particle systems)
- For best result needs to be sorted per triangle or even per pixel.
- Sorting increases CPU usage.

Reference:

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-10-transparency/>

CPU Time: 0.290000 ms
GPU 0.682932 ms
Frame Time: 0.806000 ms
-----GPU Times-----
Draw Skybox - 0.108972 ms
Render opaque geometry - 0.071684 ms
Render transparent geometry (WBOIT) - 0.348323 ms
Composite WBOIT buffers - 0.123373 ms

▼ 15_Transparency

Transparency Type
(WBOIT) Weighted blended order indepe ▼

Blend Settings

Color Resistance
1.000

Range Adjustment
0.280

Depth Range
200.000

Ordering Strength
4.000

Underflow Limit
0.010

Overflow Limit
3000.000

Reset

Light Settings

Light Position X
-5.000

Light Position Y
30.000

Light Position Z
-5.000

Weighted blended order independent transparency

This sample uses order independent transparency. Transparent surfaces can be rendered out of order. The shader blends the colors based on a weight relative to the distance to the camera and an exact coverage of the background (opaque) objects is calculated. In a composite pass the blended color and the coverage are rendered on top of the opaque geometry.

Pros:

- Easy to implement.
- Works well on old hardware.
- Very fast relative to other order independent techniques.
- Handles objects that are (100%, 50%] transparent pretty well.

Cons:

- Doesn't handle (50%, 0%] transparent objects well.
- Lots of magic numbers.
- Needs to be tweaked per scene.

Reference:

<http://casual-effects.blogspot.com/2014/03/weighted-blended-order-independent.html>

<http://casual-effects.blogspot.com/2015/03/implemented-weighted-blended-order.html>

CPU Time: 0.297000 ms
GPU 0.717381 ms
Frame Time: 0.845000 ms
-----GPU Times-----
Draw Skybox - 0.109324 ms
Render opaque geometry - 0.072412 ms
Render transparent geometry (WBOIT) - 0.376298 ms
Composite WBOIT buffers - 0.328414 ms

▼ 15_Transparency

Transparency Type
(WBOIT) Weighted blended order indepe ▼

Opacity Sensitivity
3.000

Weight Bias
5.000

Precision Scalar
10000.000

Maximum Weight
20.000

Maximum Color Value
1000.000

Additive Sensitivity
10.000

Emissive Sensitivity
0.490

Reset

Light Position X
-5.000

Light Position Y
30.000

Light Position Z
-5.000

Weighted blended order independent transparency by Volition

This sample uses weighted, blended, order independent transparency with the changes proposed by Volition in their GDC 2018 presentation. Blending of emissive transparent objects has been improved by adding a render target for the additive intensity. In the original technique emissive object lose their intensity relative to their transparency. The also changed to weight function to reduce the chance at under/overflow of the framebuffers at FP16 per pixel.

Pros:

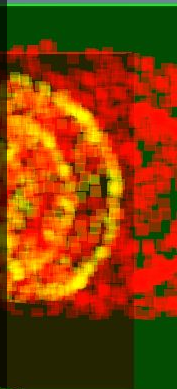
- Easy to implement.
- Works well on old hardware.
- Very fast relative to other order independent techniques.
- Handles objects that are (100%, 50%] transparent pretty well.

Cons:

- Doesn't handle (50%, 0%] transparent objects well.
- Lots of magic numbers.
- In our sample it looks worse than the original technique.
- Does not maintain correct color value for a single layer of transparency.

Reference:

<https://www.gdcvault.com/play/1025400/Rendering-Technology-in-Agents-of>



CPU Time: 0.272000 ms
GPU 2.920503 ms
Frame Time: 3.174000 ms
-----GPU Times-----
Draw Skybox - 0.108691 ms
Render opaque geometry - 0.072741 ms
Clear AOIT buffers - 0.203228 ms
Render transparent geometry (AOIT) - 2.232011 ms
▼ 15_Transparency

Transparency Type
(AOIT) Adaptive order independent trans ▼
Light Position X
-5.000
Light Position Y
30.000
Light Position Z
-5.000

Adaptive order independent transparency

This sample uses order independent transparency. It stores the color of the 4 closest surfaces per pixel. Using Raster Order Views the surfaces can be sorted from front to back without artifacts due to race conditions. If more than 4 transparent layers overlap on the same pixel the 4th color is blended together with the other layers.

Pros:

- Correct result for up to 4 transparent layers per pixel.
- Best result of the evaluated techniques.

Cons:

- DX12 only and only on GPUs that support it.
- Performance scales badly with the number of overlapping transparent layers.

Reference:

<https://software.intel.com/en-us/articles/oit-approximation-with-pixel-synchronization-update-2014>
<https://software.intel.com/en-us/gamedev/articles/rasterizer-order-views-101-a-primer>

Phenomenological Transparency

We added Phenomenological transparency[1][2] to our Transparency unit test. Phenomenological Transparency is an evolution of Weighted Blended Order Independent Transparency that has been combined with other techniques to achieve effects caused by transparent surfaces in a scene. It adds translucency, refraction, diffusion, colored shadows and caustics.

Pros:

- All effects can be used independent of one another.
- All effects are order independent.
- Most effects come at negligible performance costs.

Cons:

- Colored shadows have a high performance cost and don't look great unless aggressively blurred. Which would decrease performance even more.
- Refraction has a lot of artifacts because of missing data.
- Refraction doesn't work with backface culling.

Reference:

<https://casual-effects.com/research/McGuire2017Transparency/McGuire2017Transparency.pdf>

<https://casual-effects.com/g3d/www/index.html>

