Transparency

UCSD CSE 168
Rendering
Tzu-Mao Li
Today: rendering transparent objects in real-time, with rasterization!

https://jcgt.org/published/0002/02/09/
Rendering a single transparent object

**color C**

- 100% opacity
- 60% opacity

**quiz:** how do we compute the color here?
Rendering a single transparent object

quiz: how do we compute the color here?

\[0.6 \times C + 0.4 \times \text{background}\]
Rendering two transparent objects

**quiz 1:** which one is at the front?
Rendering two transparent objects

**quiz 1**: which one is at the front?

**quiz 2**: how do we compute the color in the middle?
Rendering two transparent objects

**quiz 1:** which one is at the front?

**quiz 2:** how do we compute the color in the middle?

\[ C_1 \alpha_1 + C_2 (1 - \alpha_1)\alpha_2 + \text{background} \times (1 - \alpha_1)(1 - \alpha_2) \]
Alpha blending

\[ C_1 \alpha_1 + C_2 (1 - \alpha_1)\alpha_2 + C_3 (1 - \alpha_1)(1 - \alpha_2)\alpha_3 + \ldots + \text{background} (1 - \alpha_1)(1 - \alpha_2)\ldots \]
Alpha blending

\[ C_1 \alpha_1 + C_2 (1 - \alpha_1) \alpha_2 + C_3 (1 - \alpha_1)(1 - \alpha_2) \alpha_3 + \ldots + \text{background} (1 - \alpha_1)(1 - \alpha_2) \ldots \]

\( C_i \alpha_i \) is often called the “premultiplied alpha”
Is alpha blending commutative?

**quiz:** will the color change if I swap the order of the two quads?
Alpha blending is not commutative!
Is alpha blending associative?
Alpha blending is associative
How do we render these transparent objects using rasterization?
Option 1: depth sorting

1. sort all the primitives based on depth
2. render them one-by-one using alpha blending

quiz: what can go wrong?

\[
C_1 \cdot \alpha_1 +
C_2 \cdot (1 - \alpha_1) \cdot \alpha_2 +
C_3 \cdot (1 - \alpha_1)(1 - \alpha_2) \cdot \alpha_3 +
\ldots +
\text{background} \cdot (1 - \alpha_1)(1 - \alpha_2)\ldots
\]
Depth sorting has troubles with cyclic order & interpenetration

cyclic order
interpenetration
A potential solution: split the primitives

**quiz:** what is the downside?

cyclic order

interpenetration
A better solution: depth peeling

idea: render multiple times
A better solution: depth peeling

idea: render multiple times

1st pass: find out the front-most color/depth using z-buffer
A better solution: depth peeling

idea: render multiple times

1st pass: find out the front-most color/depth using z-buffer
2nd pass: using the z-buffer from the 1st pass, find out the second front-most color/depth, alpha blend with first
A better solution: depth peeling

idea: render multiple times

1st pass: find out the front-most color/depth using z-buffer
2nd pass: using the z-buffer from the 1st pass, find out the second front-most color/depth, alpha blend with first
3rd pass: keep doing this
A better solution: depth peeling

idea: render multiple times

1st pass: find out the front-most color/depth using z-buffer
2nd pass: using the z-buffer from the 1st pass, find out the second front-most color/depth, alpha blend with first
3rd pass: keep doing this

quiz 1: how many z-buffers do we need to keep in memory?
A better solution: depth peeling

idea: render multiple times

1st pass: find out the front-most color/depth using z-buffer
2nd pass: using the z-buffer from the 1st pass, find out the second front-most color/depth, alpha blend with first
3rd pass: keep doing this

quiz 1: how many z-buffers do we need to keep in memory?
quiz 2: how many passes do we need?
Depth peeling is accurate but slow

need to render a scene hundreds of time
to get the correct result
Modern solution:
create an array/linked-list per-pixel on-the-fly

- turn off Z-testing for transparent objects
- collect all fragments during rendering
- sort them and alpha blend

**quiz**: how would you implement this?

a.k.a. A-buffer [Carpenter 1984]
Linked-list construction

in practice, limit the maximum fragments per-pixel to a small number, e.g., 8
Linked-lists/arrays are usually much faster than depth peeling for complex scenes

deepth peeling: 45 FPS
linked list: 338 FPS

2010

Real-Time Concurrent Linked List Construction on the GPU
Jason C. Yang, Justin Hensley, Holger Grin, and Nicolas Thibierge
Order Independent Transparency
In OpenGL 4.x

Christoph Kubisch - ckubisch@nvidia.com
Approximate solution: approximate transmittance

\[
\text{transmittance} = \prod_{i} (1 - \alpha_i)
\]

c.f. moment shadow map
Approximate solution: approximate transmittance

c.f. moment shadow map

transmittance = \prod_{i} (1 - \alpha_i)

first pass: approximate the transmittance using a smooth function (usually a Fourier basis)
second pass: render using the approximated transmittance
Approximate v.s. reference

quiz: which one is approximated?
Approximate v.s. reference

moment-based, 10 ms

depth peeling, 123 ms

Moment-Based Order-Independent Transparency

CEDRICK MÜNSTERMANN, University of Bonn, Germany
STEFAN KRUMPEN, University of Bonn, Germany
REINHARD KLEIN, University of Bonn, Germany
CHRISTOPH PETERS, Karlsruhe Institute of Technology, Germany and University of Bonn, Germany
Approximate solution: weighted, blended transparency

idea 1: rewrite the alpha blending equation to make it commutative
idea 2: use depth of the fragment to introduce weighting

\[
\frac{\sum_i C_i w(z_i, \alpha_i)}{\sum_i \alpha_i w(z, \alpha_i)} \left( 1 - \prod_i (1 - \alpha_i) \right) + \text{background} \times \prod_i (1 - \alpha_i)
\]
Approximate solution: weighted, blended transparency

\[
\begin{align*}
\text{eq. 7} & : w(z, \alpha) = \alpha \cdot \max \left[ 10^{-2}, \min \left[ 3 \times 10^3, \frac{10}{10^{-5} + (|z|/100)^6} \right] \right] \\
\text{eq. 8} & : w(z, \alpha) = \alpha \cdot \max \left[ 10^{-2}, \min \left[ 3 \times 10^3, \frac{10}{10^{-5} + (|z|/100)^6} \right] \right] \\
\text{eq. 9} & : w(z, \alpha) = \alpha \cdot \max \left[ 10^{-2}, \min \left[ 3 \times 10^3, \frac{0.03}{10^{-5} + (|z|/200)^4} \right] \right] \\
\text{eq. 10} & : w(z, \alpha) = \alpha \cdot \max \left[ 10^{-2}, 3 \times 10^3 \cdot (1 - d(z))^5 \right]
\end{align*}
\]
Approximate solution: weighted, blended transparency

quiz: which one is approximated?
Approximate solution: weighted, blended transparency

**quiz:** which one is approximated?

- approximated
- reference
Approximate solution: weighted, blended transparency

weighted-blended transparency 5.5ms

moment-based, 10 ms

depth peeling, 123 ms

Weighted Blended Order-Independent Transparency
Volumes are usually represented as billboards in video games

https://research.activision.com/content/dam/atvi/activision/atvi-touchui/research/tech-reports/docs/PracticalOT.pdf
Can we handle more complex volumes?

https://en.wikipedia.org/wiki/Smoke
What is the alpha blending equation if we have uncountably infinite primitives?
Taking $n$ to the limit of infinity...

$$\sum_{i}^{n} C_{i} \alpha_{i} \prod_{j}^{i} (1 - \alpha_{j})$$

$\downarrow$

$n \rightarrow \infty$
Taking $n$ to the limit of infinity...

\[ \sum_{i}^{n} C_{i} \alpha_{i} \prod_{j}^{i} (1 - \alpha_{j}) \]

\[ \downarrow \quad n \rightarrow \infty \]

\[ \int_{0}^{T} C(t') \alpha(t') \exp \left( - \int_{0}^{t'} \alpha(t) dt \right) dt' \]
Taking $n$ to the limit of infinity…

$$\sum_{i}^{n} C_i \alpha_i \prod_{j} (1 - \alpha_j)$$

$n \rightarrow \infty$

volume rendering equation

$$\int_{0}^{T} C(t') \alpha(t') \exp \left( - \int_{0}^{t'} \alpha(t) dt \right) dt'$$
Raymarching algorithm

$\alpha$ is usually stored in a 3D grid

use alpha blending to get the final color!
Often used in medical imaging!
Volume rendering is used extensively in recent computer vision

these are all volumes!

https://www.matthewtancik.com/nerf
Can be extended to handle multiple bounces!
Next: light baking

https://unity.com/how-to/advanced/optimize-lighting-mobile-games