

*SIGGRAPH 2016 Open Problems in Real Time Rendering Course – 26 July 2016 – Anaheim, CA*

# **Peering Through a Glass, Darkly** at the Future of Real-Time Transparency

**Morgan McGuire**

NVIDIA & Williams College

The short films were great last night at the Electronic Theatre,

# 2016 Electronic Theatre



Cosmos Laundromat (Blender Foundation)



The Legend of the Crab-Phare (Supinfocom)



Natural Attraction (Epic Scapes)



Moana (Disney)

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 2

and they were a catalog of phenomena that we can't render robustly in real-time today:

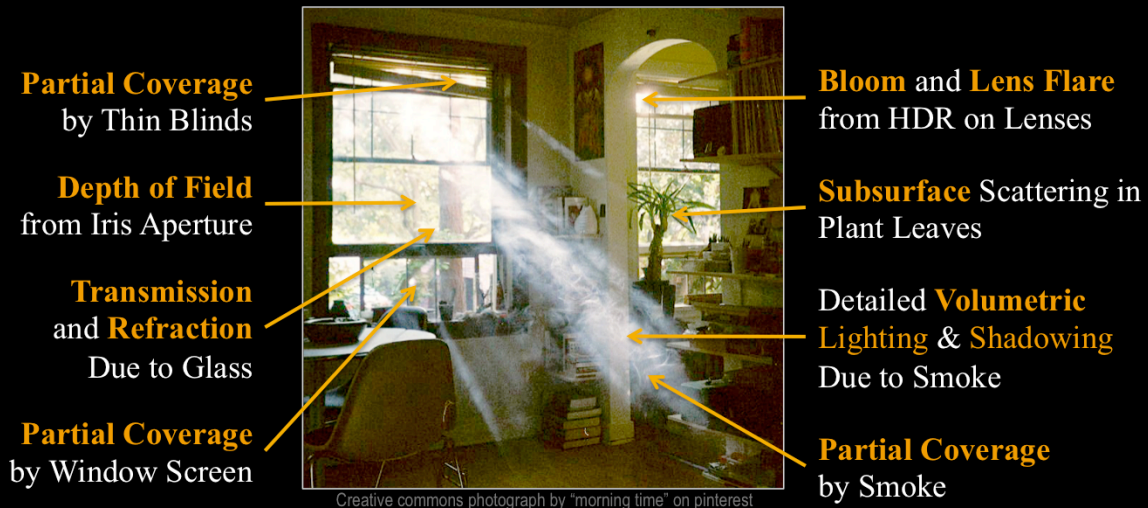
- subpixel translucent hair and grass,
- volumetric lighting on fog and smoke
- multiple scattering in clouds and rain
- correct refraction, diffusion and extinction in water and glass...

and of course, pixel-perfect subsurface scattering, motion blur, bloom, lens flare, and depth of field.

Now, a game studio with a strong art team can approximate any of these for a specific scene, but there's no real-time engine in the world where I can model clouds or water and get a correct, robust result in comparable to what even a graphics student's first OFFLINE path tracer can render for transparency. So, transparency is clearly a hard, open problem for real-time rendering. Let's take a closer look at some phenomena...



# Some “Transparency” Effects



SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 3

Here's a real photograph. We see many different effects that are all grouped under transparency...

# More “Transparency”



Transparent  
Shadows



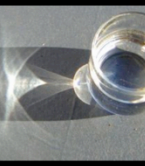
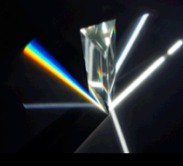
Multiple  
Scattering



Colored  
Transmission



Chromatic Abberation



Rainbows



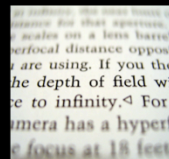
Refraction



Transparent  
Emissives



Volumetric  
Light & Shadow



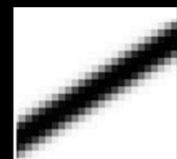
Depth of Field



Motion Blur



Thin & Perforated  
Objects



Edge  
Antialiasing

*All images from Wikimedia Commons except \* marketing image from <http://www.dachauer-spezialglas.de/praezisionsoptik/optische-filter/>*

*SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 4*

By now you should have pretty good intuition for the scope of what we casually call transparency. Let's try and define it more formally....

Real-Time “Transparency” Effects =

multiple primitives contribute to one sample (?)

...actually, only two underlying phenomena:

● SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 5

For all of the effects that we just looked at,

“transparency effects are those where multiple primitives contribute to one sample”

That’s not very exciting. It also isn’t very specific, and doesn’t relate to physically based rendering or even to sampling theory in a meaningful way.

The problem is that there are two, similar modeling scenarios going on here, which are often conflated.

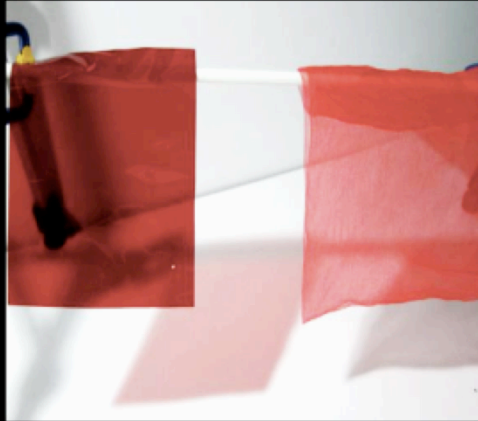
If we separate them, then we’ll see that all of the phenomena we observe arise directly from one or the other....

## Transmission

(Light passing through medium)

Refraction, diffusion, extinction, chromatic aberration, colored shadows, colored background, etc.

*Red over blue = black*



## Partial Coverage

(Varying binary coverage within a pixel)

Gray shadow. No diffusion or refraction. No coloring of background. Includes depth of field and motion blur.

*Red over blue = purple*

Image from McGuire and Enderton, Colored Stochastic Shadow Maps, I3D'11

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 6

Transmission models materials like this red gel on the left. Note that it casts a red shadow.

Partial coverage models materials like this sheer red scarf on the right. It is actually composed of many tiny opaque threads that have binary coverage, but within a pixel there are so many threads that it appears as fractional coverage. Note that it casts a gray shadow.

A simple way to know which of these you're dealing with is to ask the thought experiment of what you'd expect a red transparent object in front of a blue object to look like. If the answer is black, which is what would happen with glass or a gel, then you have transmission. If the answer is purple, which is what you'd see for a fine metal screen or smoke, then you have partial coverage.

Here's what we're going to cover in the next 40 minutes...



# Topics

## Framing the Problem:

- **Transmission** is real, and *is* rendering
- **Partial coverage** is artificial and trades one problem for another

## Framing the Solution:

- Transparency **Strategies** & Leading **Approximations**
- Future **research agenda**

⌚ SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 7

I'm going to spend the first half of the talk building a deeper understanding of the physics of transmission and computer science of partial coverage.

Once we understand the problem we're trying to solve, then we'll look at a set of strategies for tackling one or both of these core phenomena.

Finally, I'll propose short and long term research and development agendas for improving real-time transparency.

The slides from this talk are online and contain additional information, including a large bibliography of real-time transparency publications.

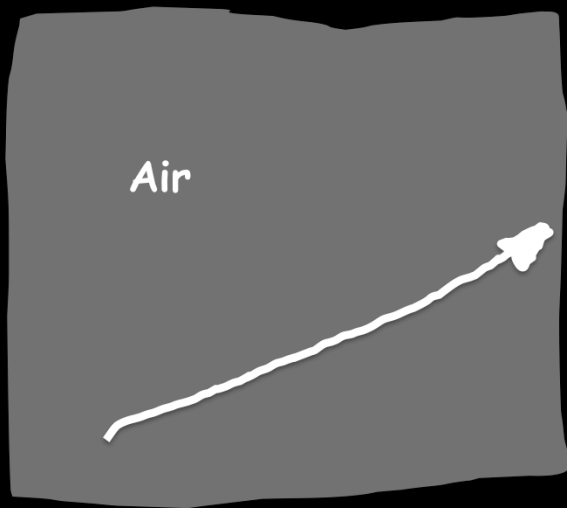
# Transmission

*A phenomenon and open problem from nature*

# Ray Optics of Transparency\*

\* Rendering typically assumes steady-state light transport and ignores polarization, phase, interference/diffraction, and most frequency-dependent phenomena. Transparency is hard enough, so let's assume those simplifications largely will be maintained in the future for entertainment applications.

# Ray Optics of Transparency



SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 10

Ok, here's our ray so we can model some optics.

We're concerned with what happens to the light initially travelling along it. That ray is in an **optically homogeneous medium**, such as "air" (at a fixed temperature, pressure, and composition).

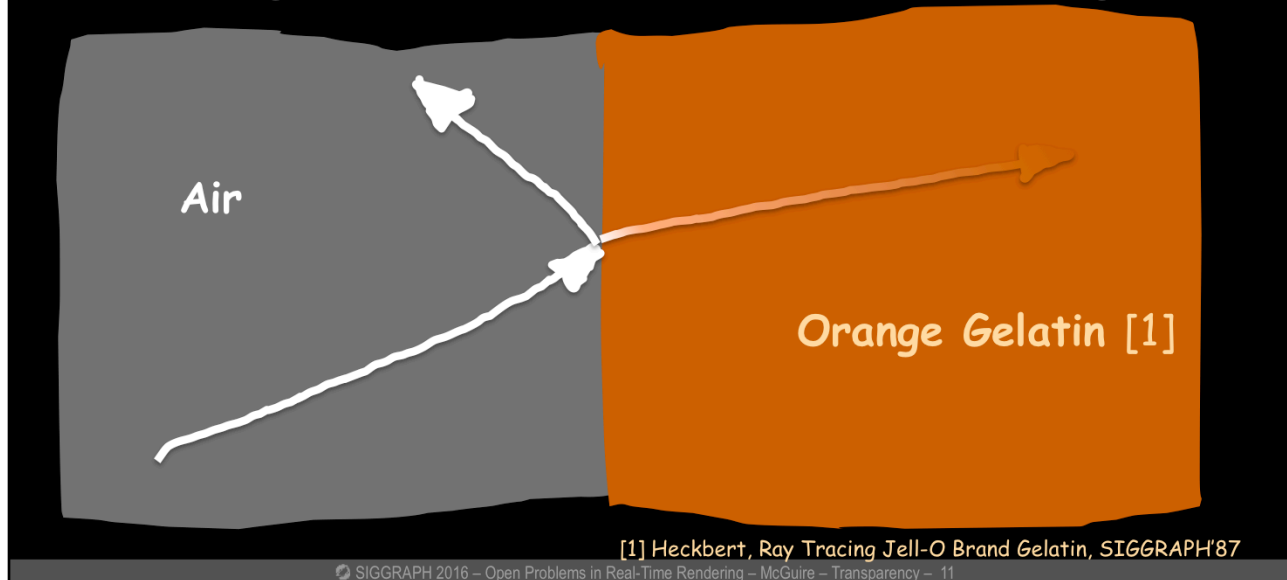
The medium has some refractive index. In fact, that is the definition of an optically homogeneous medium: the refractive index is uniform.

When the ray intersects a medium with a different refractive index, we call the interface between them a **surface**.

Let's insert another medium, such as...orange jell-o



# Ray Optics of Transparency



(If you're reading these slides after the talk...the Heckbert citation is a real paper, but that paper is itself a SIGGRAPH community in-joke)

Now, as you know, when a ray of light travels from air into orange jell-o, four interesting phenomena occur:

- First, all of the non-orange light is quickly absorbed in the jell-o
- Second, even the orange light is absorbed if the jell-o is deep enough
- Third, the light changes direction at the surface between air and jell-o due to refraction.
- Fourth, some of the light is reflected back into the air

You've know about the refractive index and Snell's law since high school, and if you're up to date on physically-based rendering you probably even know about Fresnel's laws...but let me point out that refractive indices are a little more *complex* than what we learned in high school...

# Refractive Index

(For a homogeneous medium)

Complex refractive index

Extinction coefficient (causes absorption)

$$\underline{\eta}(\lambda) = \eta(\lambda) + i\kappa(\lambda)$$

Refractive index: relative speed of light (causes refraction and aberration)

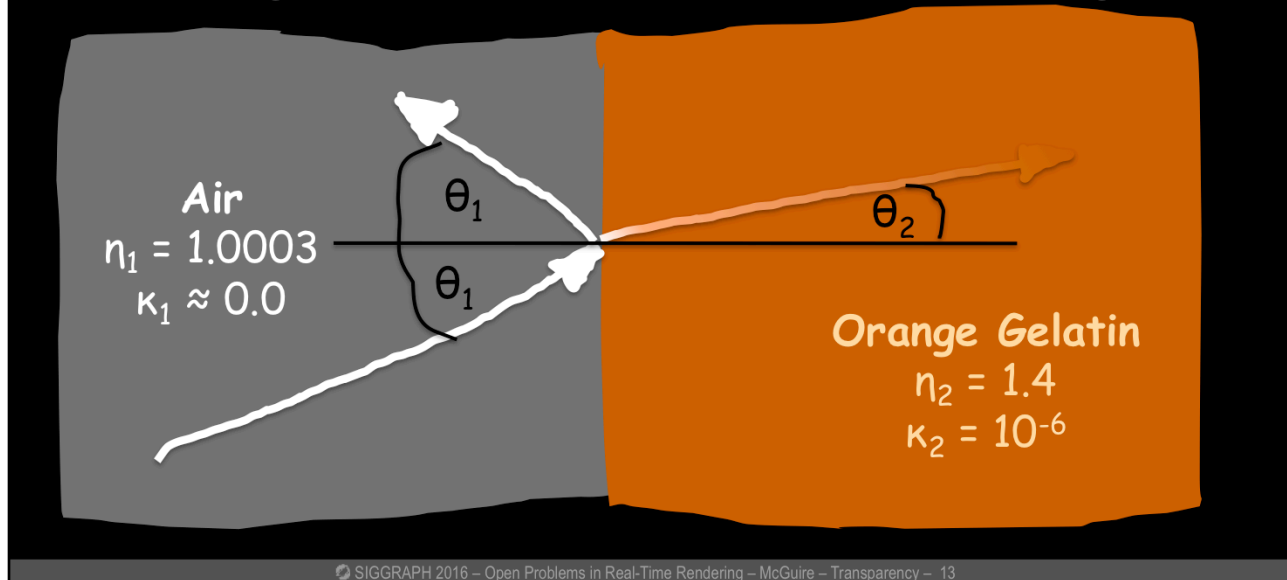
See: Hecht, *Optics*, p. 128, 4th Edition, Addison-Wesley, 2002

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 12

So, when I said “a homogeneous medium as a uniform refractive index”, I meant the *complex* refractive index, which includes an extinction coefficient tucked away in the imaginary part. That extinction coefficient describes how quickly light is absorbed over distance.

All of these depend on wavelength, which is why we see chromatic aberration in lenses, prisms separate colors, and there are rainbows.

# Ray Optics of Transparency



So here's our model again, with proper coefficients. We're ignoring wavelength for the moment.

I have no source for the extinction coefficient of orange jell-o is, but working backwards from observation, it must be about  $10^{-6}$  because you can see through a few centimeters of jell-o.

Once we know these coefficients, we actually have all of the parameters needed for rendering. Longstanding physics models tell us how much light is reflected and refracted and in which directions, as well as how the light is absorbed as it travels through the medium.

The reflection direction is trivial—it is the geometric mirror reflection direction. So, those two angle measures labelled "theta 1" are the same.

The refraction direction is given by Snell's law, which relates the angle measure labelled theta 2 to theta 1...

# Snell's Law

$$\frac{\sin \theta_2}{\sin \theta_1} = \frac{\eta_1}{\eta_2}$$

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 14

So, the refraction angle depends solely on the refractive index.

What about the amount of reflection and refraction? That's given by the Fresnel Reflection Coefficient

$$\frac{\sin \theta_2}{\sin \theta_1} = \frac{\eta_1}{\eta_2}$$



# Fresnel Reflection Coefficient

(For a dielectric interface)

**Surfaces transmit more head-on and reflect more at glancing angles**

$$F(\theta_1, \theta_2) = \frac{1}{2} \left[ \frac{\eta_2 \cos \theta_1 - \eta_1 \cos \theta_2}{\eta_2 \cos \theta_1 + \eta_1 \cos \theta_2} \right]^2 + \frac{1}{2} \left[ \frac{\eta_1 \cos \theta_1 - \eta_2 \cos \theta_2}{\eta_1 \cos \theta_1 + \eta_2 \cos \theta_2} \right]^2$$

$$F(\theta_1) \approx F_0 + (1 - F_0)(1 - \max(0, \cos \theta_1))^5$$

Schlick, A customizable reflectance model for everyday rendering, EGSR'93

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 15

Which is somewhat complicated and depends on the chemical properties of the surface...for jell-o and air it looks like this.

Note that it depends solely on the refractive indices and the angles...which also depend only on the refractive indices.

Christophe Schlick approximated the equation with the simpler one below, which is what is typically used in rendering. It is written in terms of the reflection at normal incidence...

$$F(\theta_1, \theta_2) = \frac{1}{2} \left[ \frac{\eta_2 \cos \theta_1 - \eta_1 \cos \theta_2}{\eta_2 \cos \theta_1 + \eta_1 \cos \theta_2} \right]^2 + \frac{1}{2} \left[ \frac{\eta_1 \cos \theta_1 - \eta_2 \cos \theta_2}{\eta_1 \cos \theta_1 + \eta_2 \cos \theta_2} \right]^2$$

$$F(\theta) \approx F_0 + (1 - F_0)(1 - \max(0, \cos \theta))^5$$

# Fresnel Coefficient

(For a dielectric interface)

$$F_0(\eta_1, \eta_2) \approx \left( \frac{\eta_2 - \eta_1}{\eta_2 + \eta_1} \right)^2$$

$0 < F_0 < 1$ : Fresnel reflection at normal incidence. Larger if the surface reflects a lot when you look straight down at it.

 SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 16

which depends only on the refractive indices.

Finally, how much light is absorbed in the medium? That's given by the Beer-Lambert law

# Beer-Lambert Law

Light falls off exponentially with distance in a homogeneous medium

$L$ : Radiance, the amount of light along a ray.

$\kappa$ : Extinction coefficient

$$L(X, \hat{\omega}) = L(Y, \hat{\omega}) \exp \left( \frac{-4\pi\kappa(\lambda)}{\lambda} ||Y - X|| \right)$$

$X, Y$ : two points in the same medium

$\lambda$ : Wavelength, the spectral “color”

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 17

Which says that transmission decreases exponentially with distance.

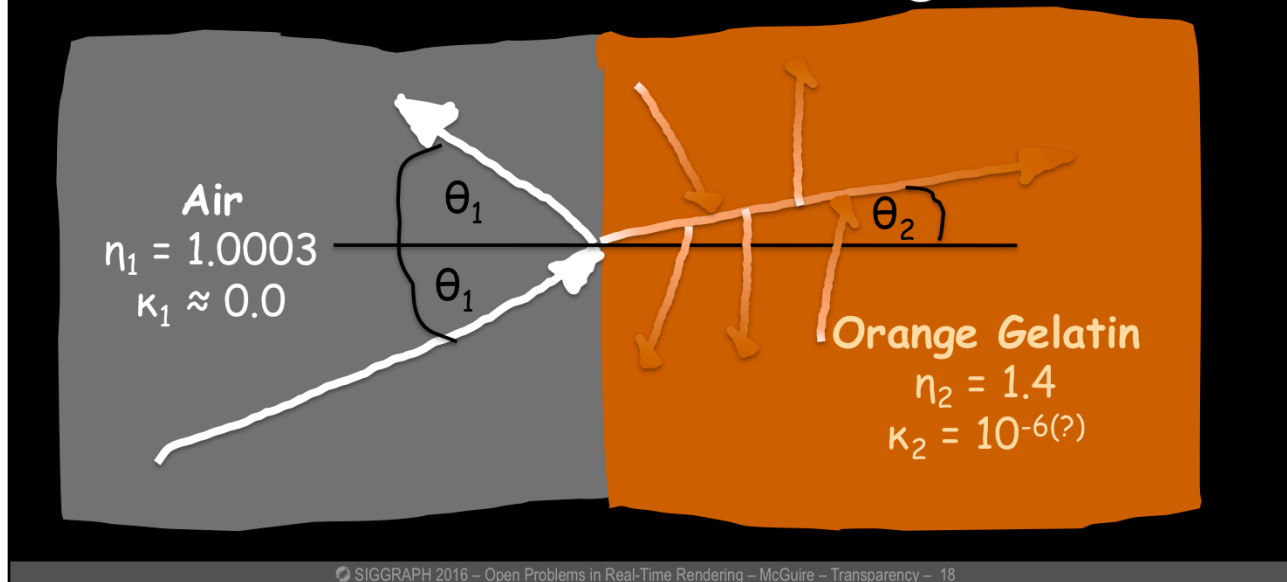
The exponent depends on the extinction coefficient, which was the imaginary part of the complex index of refraction.

There’s one important simplification here that will eventually cause us some trouble. We’ve assumed a *single scattering* model...in physics, the extinction coefficient accounts not only for absorbed light but light that is diffusely scattered within the medium. Let’s set that multiple scattering aside however.

---


$$L(X, \hat{\omega}) = L(Y, \hat{\omega}) \exp \left( \frac{-4\pi\kappa(\lambda)}{\lambda} ||Y - X|| \right)$$

# In/Out-Scattering

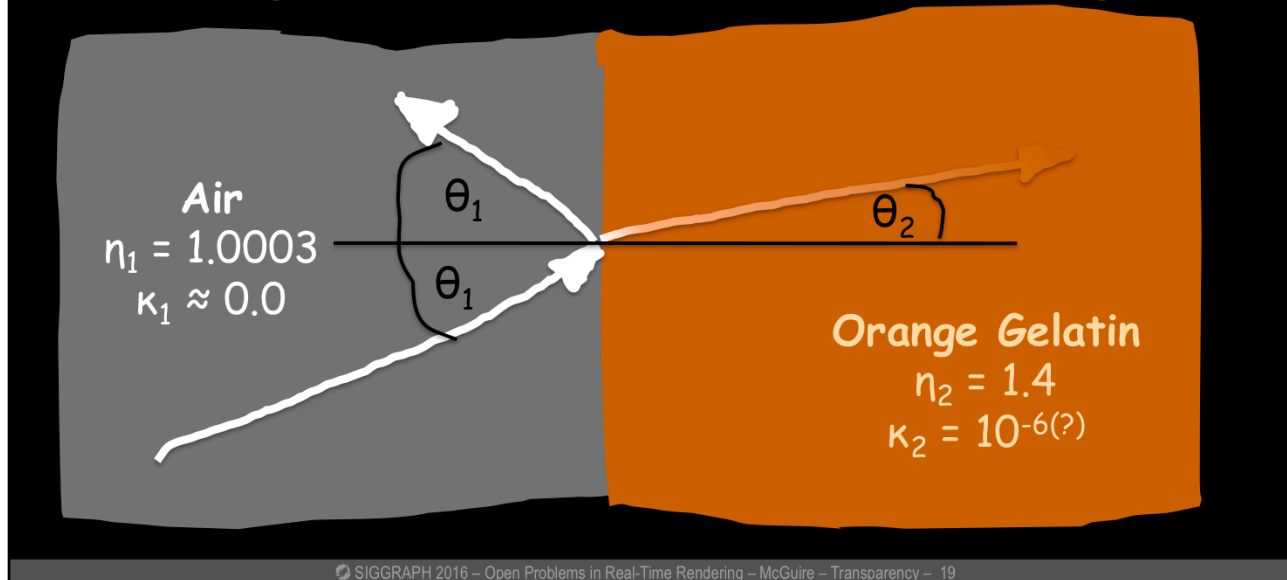


The extinction (kappa) term also takes into account that some light is scattered **out** from the ray based on the molecular and chemical structure of the medium. And of course, the light scattered out from other paths has a chance to be scattered **in** to this ray. So, there's some diffusion...if you look through orange jell-o, you'd expect the image to be a little blurry.

I'm going to set this aside for a moment to keep the story simple, but diffusion is well understood in physics and something that can be well-modeled by the best physically best renderers.



# Ray Optics of Transparency

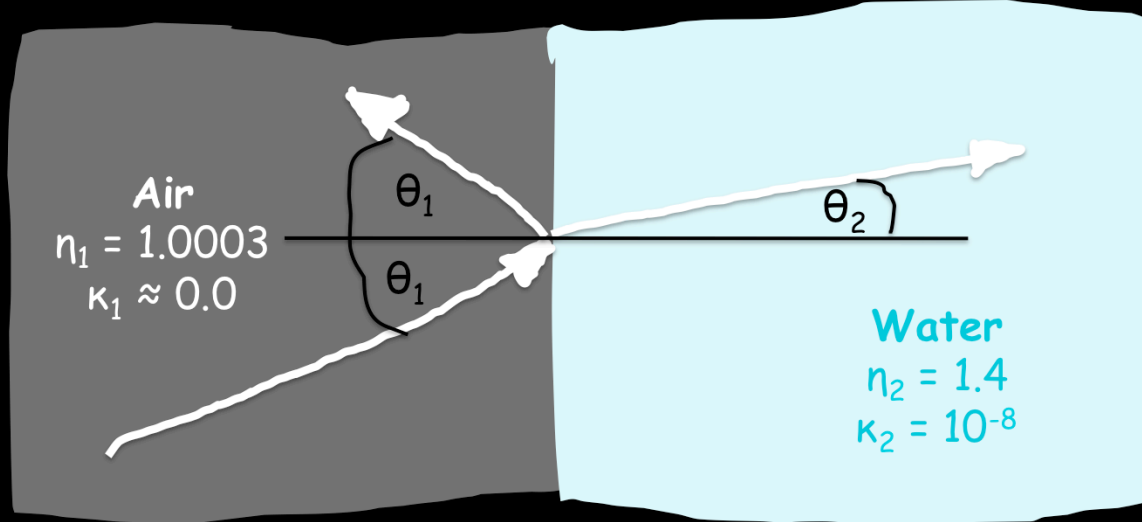


The key point of this entire derivation was that we can model the entire interaction of light and matter based solely on the complex index of refraction of media and knowing where their boundaries are.

Except for one point in the Fresnel model where I assumed the media were dielectrics, I didn't say anything about what the materials were. (and there's a similar set of equations for conductors, I just didn't show it.)

There was no notion of "opaque", "shiny", "mirror", "transparent", or any of the other usual computer graphics abstractions. That's really important. It means that the model I just described still holds when I change materials...

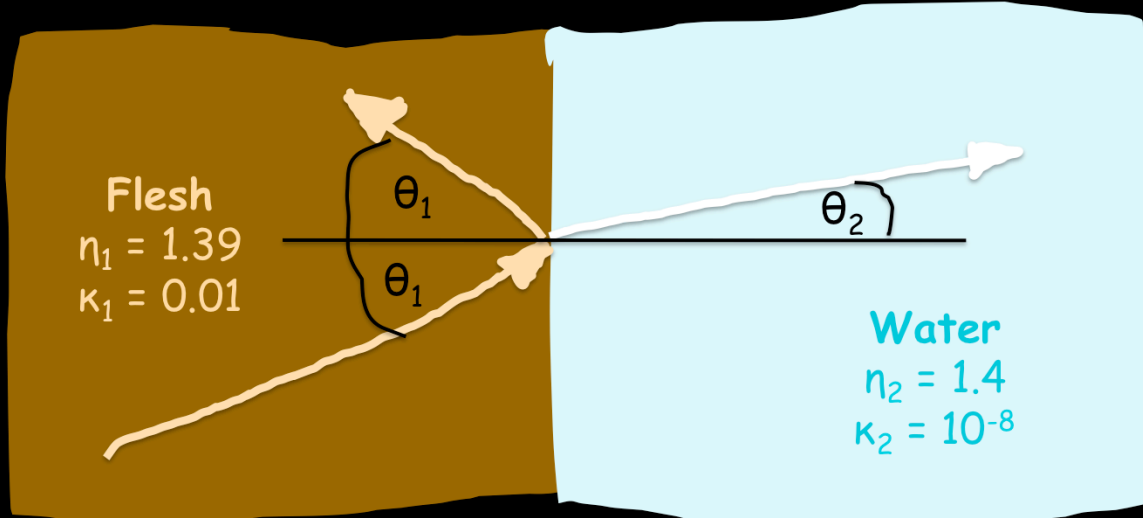
# Ray Optics of Transparency



Hale and Querry, Optical constants of water in the 200-nm to

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – 200- $\mu\text{m}$  wavelength region. *Applied Optics*, 1973

# Ray Optics of Transparency?



Ding et al., Refractive indices of human skin tissues at eight wavelengths and estimated dispersion relations between 300 and 1600 nm, *Physics in Medicine and Biology*, 2006

Transparency – 21

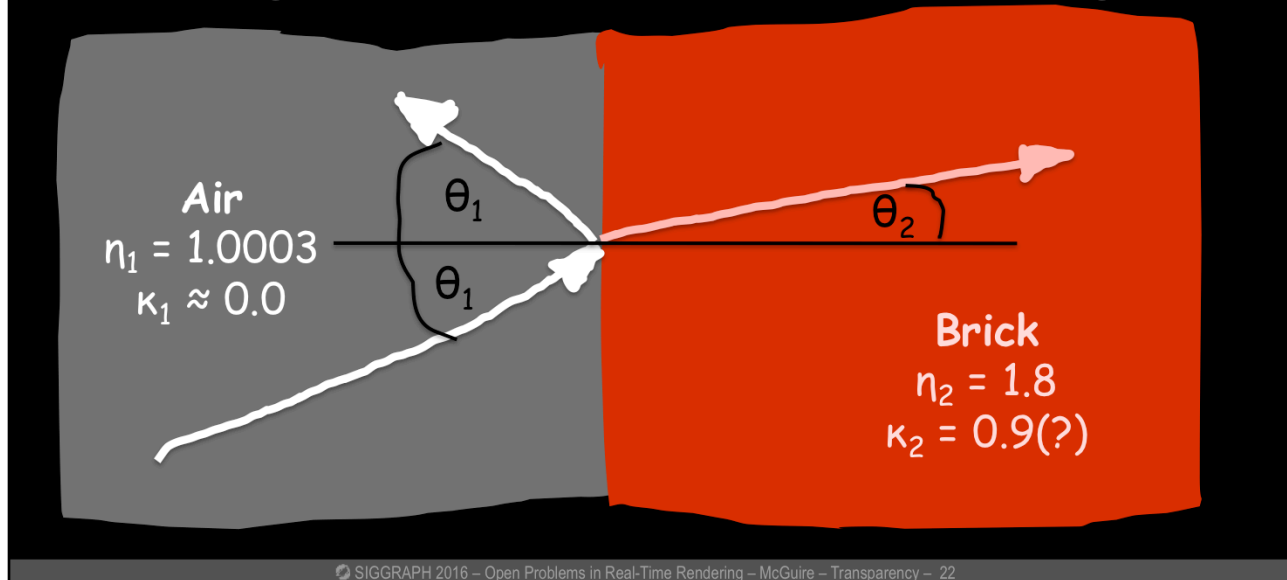
Wait, can I send light through skin? Yes... The extinction coefficient is just relatively large.

Through an ear lobe, eyelid, or finger you observe significant transmitted light.

(By the way, skin is the kind of material where a multiple scattering model is useful)

Ok, let's keep going. How about .... Brick?

# Ray Optics of Transparency?!



What? I can apply the transmission model to a brick? Bricks are obviously opaque.

Well, you *can* see through a brick if it is thin enough. Bricks just absorb visible light really quickly. They have high extinction coefficients.

They also have high real refractive indices...a brick lens will refract pretty severely.

The faulty conclusion that a brick will block light is based on two assumptions:

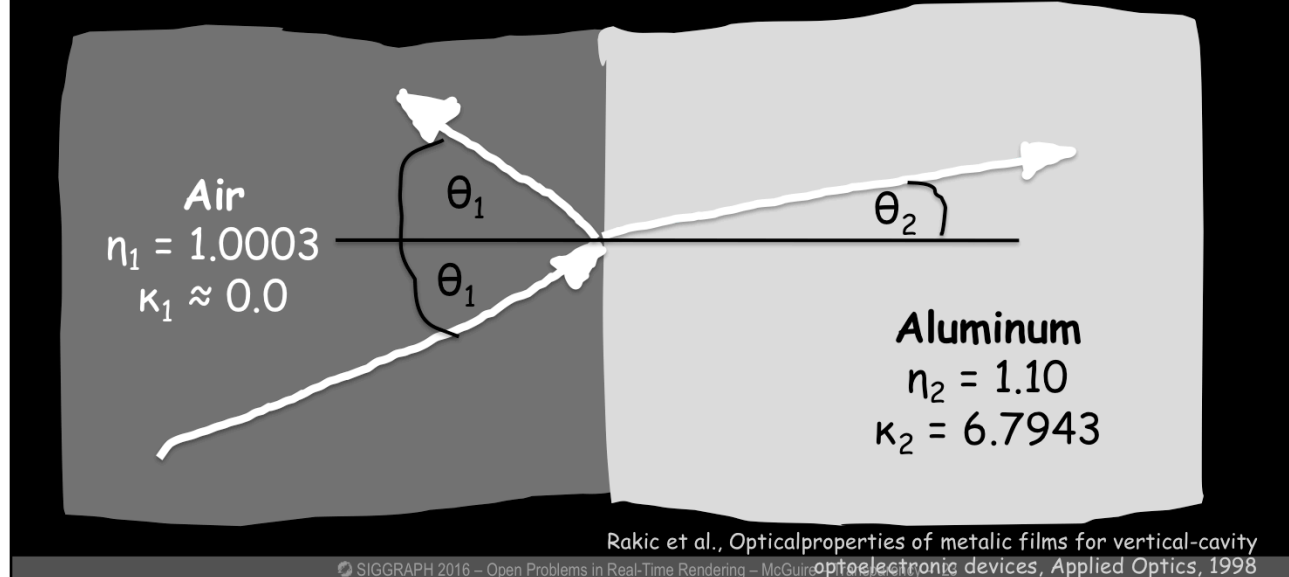
- That the brick is thick enough that almost all light is absorbed
- That we're using visible wavelengths of light...the model applies to all wavelengths, and if you hit a brick with microwaves, radio waves, or X-rays, they penetrate very differently than visible wavelengths but they're all just "light". Remember, bricks aren't just red—they have a color spectrum outside what we can see

(Actually, I don't know what the extinction coefficient of brick is at visible wavelengths. But there's a lot of experimental data on building materials at microwave and radio wave frequencies:

<https://www.brown.edu/research/labs/mittleman/sites/brown.edu.research.labs.mittleman/files/uploads/JansenJIMT.pdf>)

I can even do this:

# Ray Optics of Transparency?!!!



So, the point is...that's all. I'm not just telling you about transparency. I just described the geometric and physical model of all computer rendering:

# Ray Optics of **Rendering**

- Model optically homogeneous media parameterized by **complex refractive index**
- Apply **Beer-Lambert** within a medium
- Compute **reflection** and **refraction** by applying **Fresnel** and **Snell** at **interfaces**

[and use “some model” of diffusion/**multiple scattering** within a medium]

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 24

That's all of computer graphics.

And all of the other fancy effects fall out of this same model.

For example, lens flare, bloom, and chromatic aberration occur because of these reflections and refraction in the lenses of a camera.

There are even three algorithms for correctly rendering from this model...

# Complete Rendering Algorithms

- Metropolis Light Transport [Veach97]
  - Path Tracing with Photon Mapping [Jensen96]
  - Bidirectional Path Tracing [Lafortune93]
- 
- BRDF, BTDF, BSDF [Torrance67, Blinn77, Torrance82...]
  - BSSSRDF [Jensen01]
  - Photon Beams [Jarosz08]

🔗 SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 25

Unfortunately, they're all really slow. These will probably never be real-time algorithms for complex scenes.

[click] Even with some aggregate models to speed convergence for heterogeneous volumes...

[click] offline film rendering still approximates and essentially fakes transport in clouds, skin, and other complex materials. So, we're probably not going to directly apply these algorithms to real-time rendering.

# Transmission Conclusions

- A large part of the “open problem in transparency” is... truly **physically-based rendering**
- **Offline algorithms exist** to render correctly, given correct scene models
- The current versions of those algorithms likely **won't ever scale** to real-time for media with **mid-range extinction coefficients** ( $\kappa$ ) and **high diffusion**



That was a lot of math, very quickly!  
Read the slides at your own pace at  
<http://advances.realtimerendering.com/>

A more detailed explanation and all equations  
appear in *The Graphics Codex*  
<http://graphicscodex.com>

# Topics

## Framing the Problem:

- **Transmission** is real, and *is* rendering
- ➔ • **Partial coverage** is artificial and trades one problem for another

## Framing the Solution:

- Transparency **Strategies** & Leading **Approximations**
- Future **research agenda**

⌚ SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 28

Now that we understand transmission, let's look at the other half of the problem...  
partial coverage

# Transmission:

## *A Phenomenon from Nature*

- Under the ray optics model\*, rays travel through piecewise-homogeneous **media**
  - Homogeneous = constant refractive index (speed) and extinction coefficient (absorption). Both are functions of frequency (“spectral color”)
- Radiance absorbed exponentially with distance, according to the Beer-Lambert model. It also in- and out-scatters from the ray due to diffusion “(multiple scattering)”
- A **surface** is just the interface between two different media. At a surface:
  - Fraction (determined by Fresnel, etc.) of the light ray reflects in the mirror direction
  - Fraction (ditto.) continues, at a direction given by Snell
- Actually, that covers all of rendering! Everything else you’ve heard of is a special case or approximation of this general scenario.
  - “Opaque” surfaces just have really high reflection and absorption rates for visible light. Recall that those are functions of the refractive index...which is why X-rays see through “opaque” materials and “clear” sunscreen blocks UV light
  - Glossy reflection is just mirror reflection on surfaces with complex geometry
  - Subsurface scattering arises from transmission into partly-absorbing medium with lots of substructure
- Cameras create some high-level effects because of transmission in the lenses
  - A camera’s objective contains multiple physical lenses, each of which is imperfect. There’s a tiny amount of diffusion and reflection, which is not noticable unless you’re looking at something really bright: lens flare and bloom
  - Wide aperture creates “depth of field” z-region where points blur less than one pixel and so appear sharp; outside of that, defocus is obvious. This also creates vignetting
  - Because of frequency-dependence, chromatic aberration is evident at edges of the frame: each lens is a curvy prism
- Offline Metropolis Light Transport, Path Tracing with Photon Mapping, and Bidirectional Path Tracing are the only complete rendering algorithms.
  - They model the light scattering described here to arbitrary accuracy...but can be *really* slow
  - When there are a lot of surfaces (“participating medium”), convergence is too slow to use. Even film companies fake clouds.
  - Photon beams are one technique for improving convergence
  - BSSSDFs approximate sub-surface case
  - Ray marching approximates the “participating medium” case
- There is no comparable real-time complete rendering solution. Instead, renderers approximate each of the effects on this page independently and phenomenologically (that is, they are hacks). That’s fragile and requires a lot of art direction to avoid surprising visual failures.

\* Under a full quantum model, these effects are much more complicated. At that level, we can’t even model transmission and reflection without also modeling diffraction/interference and polarization. But let’s set that aside: we have enough open problems.

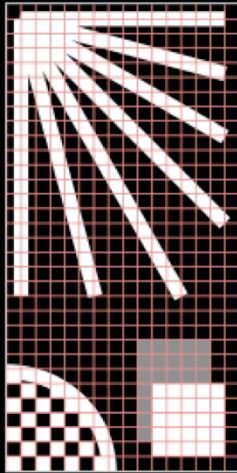
# Partial Coverage ( $\alpha$ )

A phenomenon and open problem  
we created to reduce ray aliasing

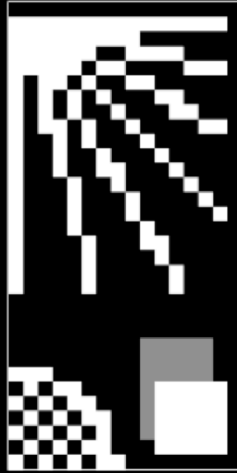
SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 30

All modern graphics is based on sampling light along rays.  
This includes GPU rasterization, which is an efficient way of amortizing the cost of sampling triangles with a bundle of rays.

# Binary Coverage Causes Aliasing



Geometry



1 ray/pixel



16,384 rays/pixel

Image from Chajdas et al., Subpixel reconstruction antialiasing, I3D'11  
SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 31

Now, a ray either intersects a triangle or it doesn't. Coverage is fundamentally a binary property. We can model a scene using triangles with parts cut out using a binary alpha mask, but the coverage is still binary. This is physically correct—rays don't sort-of interact with surfaces, they either do or don't.

In this diagram, the red lines show the pixel grid and the white objects are the geometry that we'd like to render.

The problem with sampling along rays is that it causes **aliasing** when we don't use enough rays to measure the subpixel coverage within a pixel. This is why the center image has jagged edges.

For a pixel that is partially covered by several surfaces, we'd need a lot of rays to average the 0's and 1's to the correct ratios...

The idea of partial coverage is that we want the image on the right, but we only want to pay for a small number of rays at each pixel

# Binary Coverage Causes Aliasing

- Real ray-triangle coverage is binary
  - Hit:  $\alpha = 1$
  - Miss:  $\alpha = 0$
- Estimating pixel values with rays causes aliasing
  - noise, flickering, and blur

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 32

Now, a ray either intersects a triangle or it doesn't. Coverage is fundamentally a binary property. We can model a scene using triangles with parts cut out using a binary alpha mask, but the coverage is still binary. This is physically correct—rays don't sort-of interact with surfaces, they either do or don't.

The problem with sampling along rays is that it causes **aliasing** when we don't use enough rays to measure the subpixel coverage within a pixel.

For a pixel that is partially covered by several surfaces, we'd need a lot of rays to average the 0's and 1's to the correct ratios...

# Aliasing Solution: Partial Coverage

- **Partial coverage  $0 < \alpha < 1$**  [Porter84, Smith96, Glassner15]
  - Prefilter coverage for many rays to reduce noise
  - Assumes objects have uncorrelated subpixel positions
  - Assumes visibility is independent of shading
- Arises from:
  - Edge antialiasing
  - Subpixel primitives
  - MIP-mapped binary  $\alpha$  textures
  - Depth of field (rays with different origins)
  - Motion blur (rays in time)
  - Artists repurposing  $\alpha$  to simulate transmission!

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 33

Partial coverage was introduced early in the history of computer graphics as a way to describe the statistics of unmodeled subpixel geometric structure. The models of partial coverage in place today date to Porter and Duff's 1984 compositing paper.

Partial coverage allows us to reduce or eliminate sampling noise when rendering all of the situations listed on this slide, and are an indispensable part of today's real-time renderers. Here's an example of how they work:

# “Over” Compositing

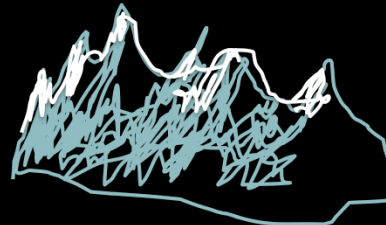
$$L(C, \hat{\omega}) = L(A, \hat{\omega})\alpha_A + L(B, \hat{\omega})(1 - \alpha_A)$$



C: Composite seen  
by the eye



A: Foreground tree



B: Background mountain

Porter and Duff, Compositing digital images, SIGGRAPH'84

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 34

On the left is my eye, or a camera, looking rightwards. In line with a sampling point on the image plane are a pine tree and then a mountain on the far right.

Under **binary** coverage, a light ray from the mountain either hits the tree leaves and is blocked, or passes through to my eye. So, a point on the image plane has a value determined by one object or the other but never both.

Under **partial** coverage, the renderer estimates how much of the entire pixel was covered by tree leaves. The remainder is then covered by the mountain.

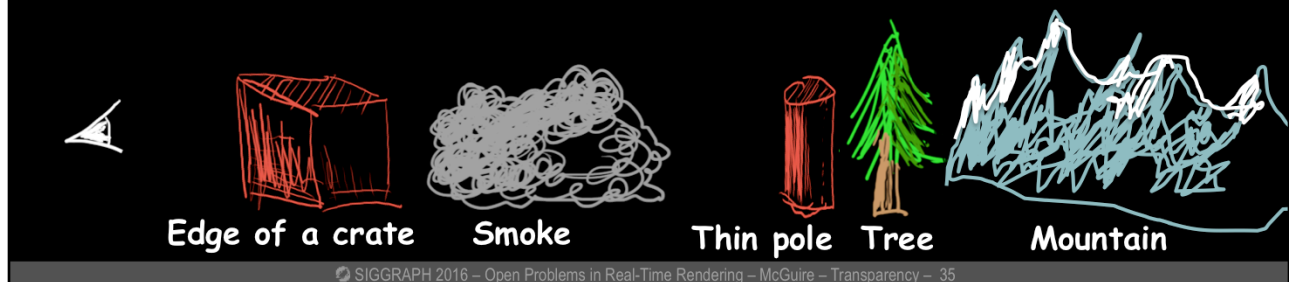
Now, the value of the pixel is a combination of the tree at A and the mountain at B, based on the partial coverage alpha by the tree. The “OVER” compositing operator models this. There are other operators that you’d use for other ways of considering the problem, but they’re equivalent.

Here’s the challenge for real-time rendering. Imagine that I have a scene with more than a single tree in it...



# Compositing is Order-Dependent

$$L_C = L_A\alpha_A + (1 - \alpha_A)(L_B\alpha_B + (1 - \alpha_B)(L_D\alpha_D + \dots))$$

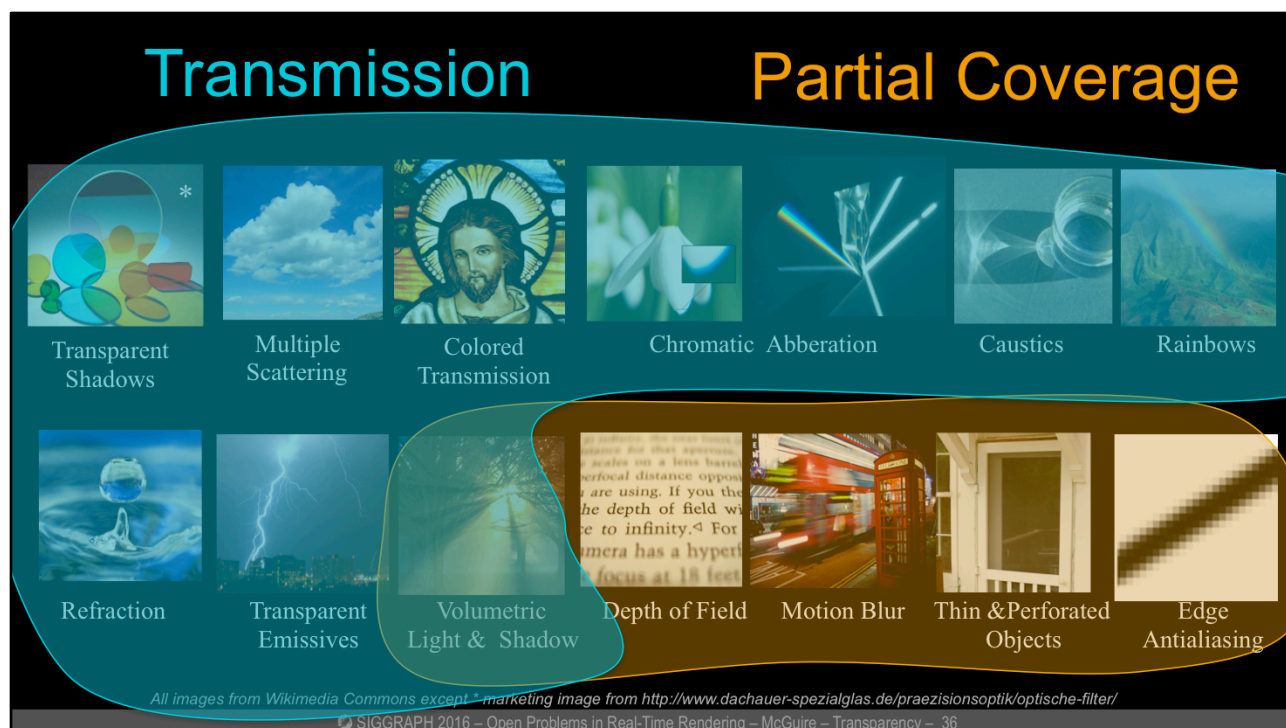


A ray tracer would encounter each of these objects in order and could simply apply the OVER operator. It doesn't care about partial vs. binary coverage.

But a rasterizer processes object primitives, not rays. That was fine under binary coverage because each sample was covered by a single object...we could write those objects to the frame buffer and the depth buffer in any order and let which ever one was closest at the end of the frame color the pixel.

Under partial coverage, we have two choices for perfect compositing, both of which are undesirable:

1. Save every single primitive that affects a sample, and then sort and composite them at the end. This is called an A buffer and takes too much space to use.
2. Sort the primitives and then composite them as they are rendered. This is slow, and impossible in the general case.



Volumetric light and shadowing occurs under both transmission (due to in and out scattering) and partial coverage (due to lots of small particles).

Depth of field is funny...it is a partial coverage phenomenon that is caused by transmission through a lens.

“Can’t I just use alpha for transmission as well as partial coverage?”

This is art. You can use anything you want to achieve the look that you want.

But if you *want* your glass, fog, and water to look realistic, then...no. Alpha isn’t enough.

You *can* use transmission to model partial coverage, but I don’t think it is a good idea. Do you really want to figure out the extinction coefficient and thickness that will exactly equal the coverage of a telephone wire through a given pixel?

# Partial Coverage:

## *A Phenomenon We Created*

- Coverage is not natively a transparency problem: an object either interacts with a light ray, or it doesn't
- All mainstream renderers:
  - sample along **rays** (rasterization is an efficient way of checking a lot of rays against one triangle at once)
  - output **pixel** values (eventually)
- **Aliasing** results if we sample too few rays to measure a pixel's value
  - Camera visibility: "edge aliasing"
  - Light visibility: "shadow aliasing"
  - Orientation: "specular aliasing"
  - Material: "texture aliasing"
  - Motion: "temporal aliasing"
- Can blur the signal, losing some correctness to avoid flicker and jaggies
  - MIP-mapping, PCF shadow filtering, MLAA & FXAA antialiasing
- "Alpha" is a model of sub-pixel coverage for blurring many binary visibility samples into a single continuous value
  - MIP-mapped mask texture
  - GPU numerical coverage mask to alpha, e.g. `glEnable(GL_LINE_SMOOTH)`
  - Text antialiasing
  - Research analytic coverage
  - Motion blur and depth of field
  - CAD/DCC/game nonphotorealistic hidden surface views
- Turns partial coverage *into* a transparency problem
- Not the same as transmission:
  - Example of 99% "transparent" glass and a 1% coverage ( $\alpha = 0.01$ ) surface with Fresnel effects
  - Can't model wavelength-specificity with partial coverage
  - Statistical independence assumption of coverage (see Porter & Duff, Smith & Blinn)
  - Not a real-world phenomenon! This is an artifact of our ray-based renderers
  - Historically used to approximate glass and other effects

# Topics

## Framing the Problem:

- **Transmission** is real, and *is* rendering
- **Partial coverage** is artificial and trades one problem for another

## Framing the Solution:



- Constraints
- Transparency **Strategies** & Leading **Approximations**
- Future **research agenda**

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 38

We have built some understanding of what the transparency problem is and we know how to solve it for offline rendering.  
Now let's look at real-time constraints and approaches to a real-time solution.

# Ideal Solution

- **General-purpose**
  - E.g., look through a double-pane window at a smoky room with a fish tank in it...different media composite correctly
  - Transparent shadows consistent with visible transparency
- Support for *non-physical rendering*
  - Hidden surface views, artistically lit clouds, magic/sci-fi FX
- **Robust** (predictable)
  - No light/dark leaks, energy conserving, temporally stable
- Minimal compromises
  - *All* glass reflects, *all* fog has multiple scattering, etc.

# Elephants in the Room

- **Deferred shading**

- “Deferred” renderers actually make forward passes for transparents
- $k$ -buffer of deferred samples?...AGAA [Crassin15]

- **Post-processed effects**

- Motion blur, depth of field, temporal AA, screen-space reflections, ambient occlusion
- Need depth, normal, motion vector, etc. at every sample
- Don’t work well on forward-rendered transparents or ray tracing [McGuire15, Salvi16, El Garawany16, Gonzalez-Ochoa16]
- E.g., edges of tree leaves, hair, glass windows on cars under bridge

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 40

I’m just going to throw this out there as the first rarely-acknowledged challenge in real-time transparency. Modern renderers are fundamentally designed around z-buffers in a way that assumes no transparency.

**Deferred shading** assumes ONE point to shade at each sample location on screen. Transparency necessarily has multiple points.

**Post-processed effects** assume one surface per sample.

# Game Production Constraints

- Huge **range** of performance and GPU features: mobile, current-gen (DX11), DX12, and 10x variation on current PC
- Can **engineer** per-platform
- Too expensive to have custom **art** per platform
- Transparency budget is 0.5 - 4 ms/frame

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 41

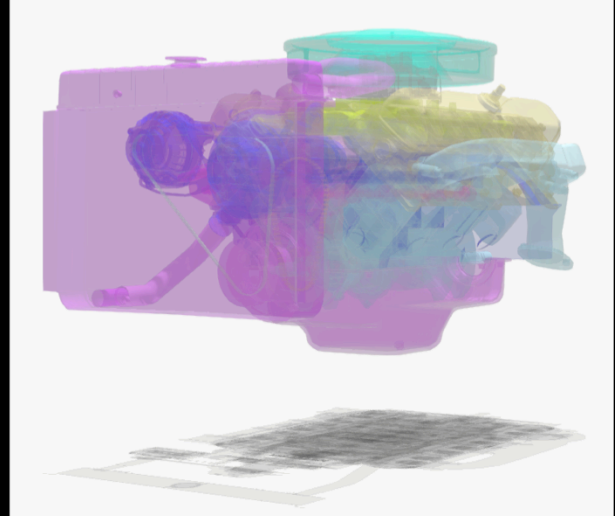
Game development targets a wide range of platforms. Projects are dominated by art, so it is usually ok to spend more engineering time to have platform-specific implementations, but it is not acceptable to have to re-model or re-tune art assets for each platform. Nobody can afford to ship a game with one strategy for transparency on PC and a different strategy on Xbox One...so a multiplatform title or engine is likely to adopt the lower-end Xbox One's constraints even on PC.

There's also more to real-time rendering than Games...

# Digital Content Creation



[Salvi14]



[McGuire16]

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 42

Content creation applications, like those by Autodesk, are the greatest consumers of real-time transparency algorithms. They have to render models with realistic glass for architects, as well as subpixel wireframes and hidden surfaces.

Here, robustness is essential, and there's no room for an artist to tune the materials and insert hints. The algorithms need to just work correctly...which is what game developers *want* anyway.



# Topics

## Framing the Problem:

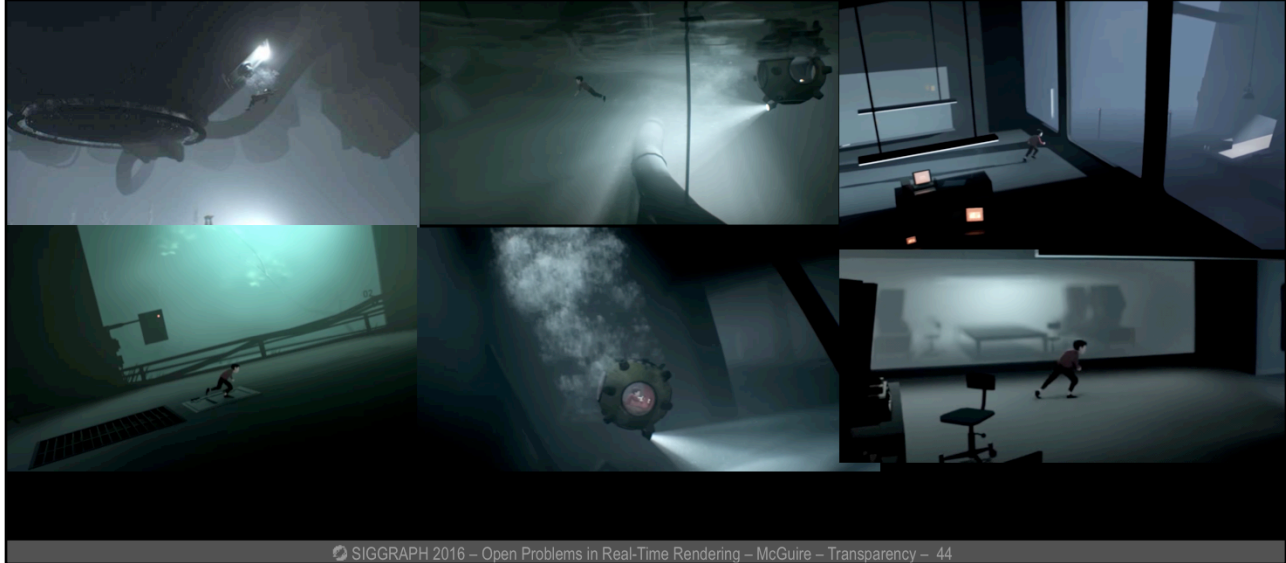
- **Transmission** is real, and *is* rendering
- **Partial coverage** is artificial and trades one problem for another

## Framing the Solution:

- Constraints
- ➔ • Transparency **Strategies** & Leading **Approximations**
- Future **research agenda**

I'm going to categories today's strategies, but first,

# Playdead's "Inside" (2016)



There is no question that the game with the best transparency today is Playdead's "Inside", which was released three weeks ago.

The game is full of overlapping water, bubbles and debris in water, fog, smoke, fire, and glass, with diffusion, refraction, and volumetric shadows. There's also zero aliasing in this game...partial coverage is handled perfectly.

However, I suspect that most of what's going on here is really masterful art direction and modeling for the fixed camera and specific scenes. This is how good we want all real-time transparency to look, but this probably isn't a practical way for us to achieve it in most applications.

The bibliography for this course (which won't be shown on screen) contains hundreds of citations.

The recent GDC and SIGGRAPH Advances presentations are where you should look for today's best practices. I'm largely NOT going to summarize those.

Instead, for the next ten minutes I will focus on some forward-looking techniques, which often aren't quite ready for deployment today. These are what I'd start from when inventing new techniques for games now in preproduction, and which might to hold the seeds for future research of a comprehensive transparency solution.

# Strategies

- Sorted Transparency
- Ray Tracing
- Low-Resolution Visibility Function (OIT)
  - $k$ -buffer, Fourier, weighted-blended
- Stochastic Transparency
- Volumetric (Voxels)
- Special cases and Post-Processing

 SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 45

Explain: Two problems, some techniques for one or the other, some for both

# Sorted Transparency

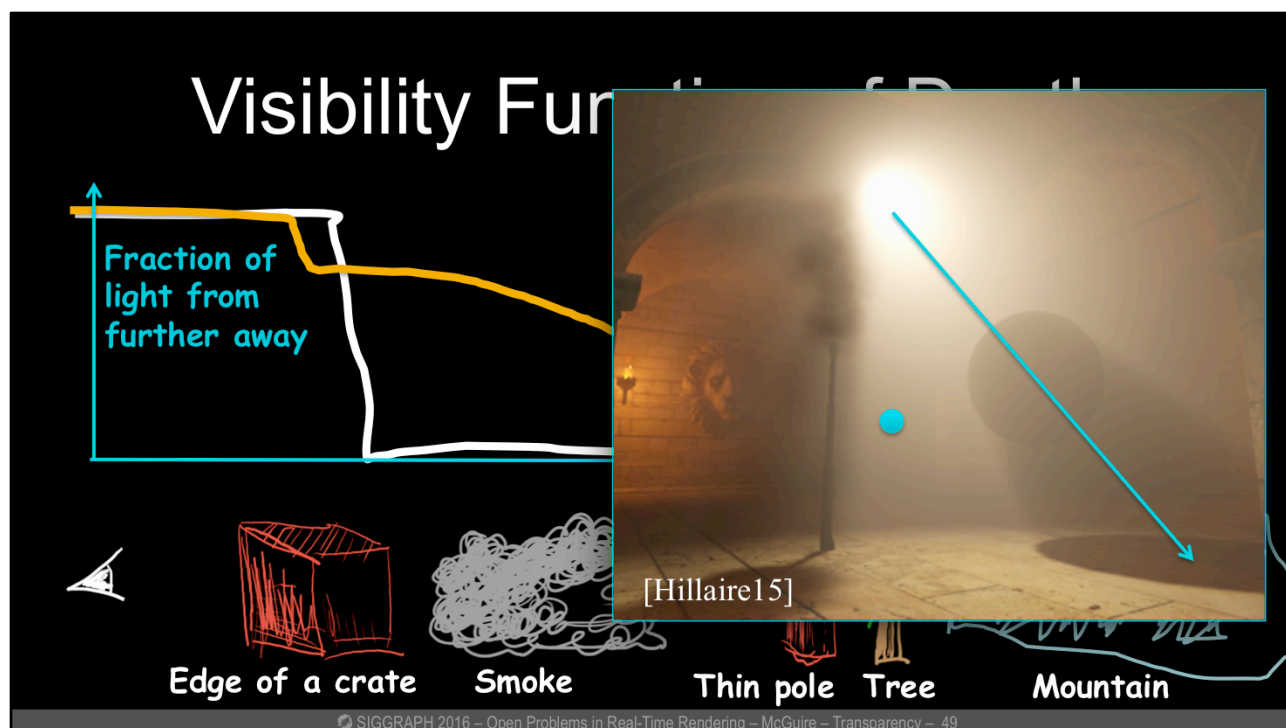
- Can handle **coverage and transmission**
- Useless for shadows and indirect light
- Inefficient on modern architectures
- Dominates modern practice  
 (“forward transparent pass”)

# Ray Tracing / Ray Marching

- Bulletproof **transmission** strategy
- Handles most **partial coverage** well
- Works for indirect light and shadows
- **OptiX/Embree/FireRays** real-time implementations
  - Too slow for *full* game rendering, but within 10x
  - A second CPU/GPU could trace one path per pixel at 720p/30Hz for modern game assets right now

# Low-Resolution Models of Visibility

 SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 48



## Visibility function

...issue: do you use the SAME approximation for eye-paths and light paths? E.g., visible samples and shadows?

Many errors can be concealed when making approximations along eye light paths because you're seeing the ray end-on and only noticing the accumulation [click]...but when we consider volumetric lighting, you can see this function *from the side* and can't cheat it so easily...[click]

# Low-Resolution Models of Visibility

- Deep shadow maps [Lokovic00]
- $k$ -buffers, blended OIT, Fourier opacity maps

 SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 50

Lokovic and Veach's deep shadow maps introduced the notion that



# $k$ -Buffers

- Consistency Principle: submitting primitives in the same order produces the same result, but it is not purely “z-order independent”
- Modern implementations: atomics or interlock
- $k = 1$ : Store min depth, choose over/under blending based on depth test [Salvi14]



[Maule13] Atomic  $k$ -buffer partial coverage

● SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 52

Atomics-only k-buffer like approach: accumulate and sort, then composite. Works on current consoles, faster than pixel sync on some platforms.



[Salvi14] Pixel shader interlock  $k$ -buffer partial coverage

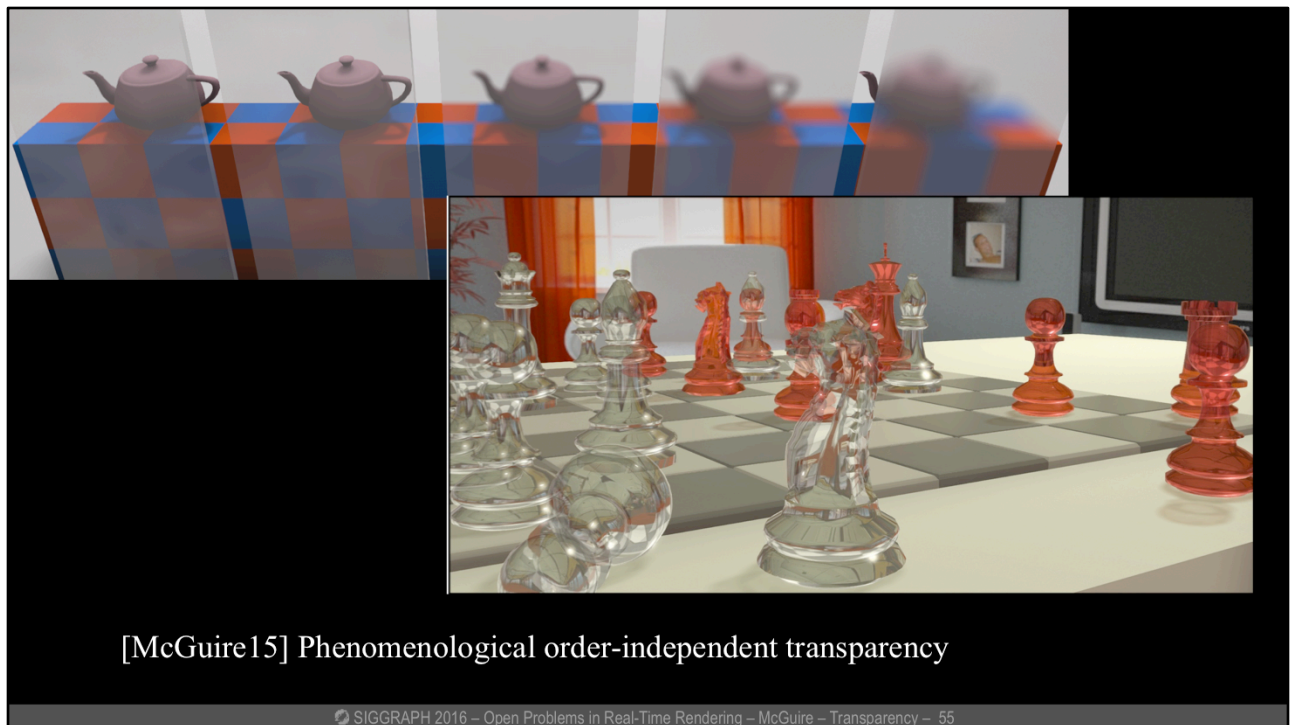
SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 53

More sophisticated and modern version of Maule...

I think this is the best current solution for hair.

# Order-Independent Blending

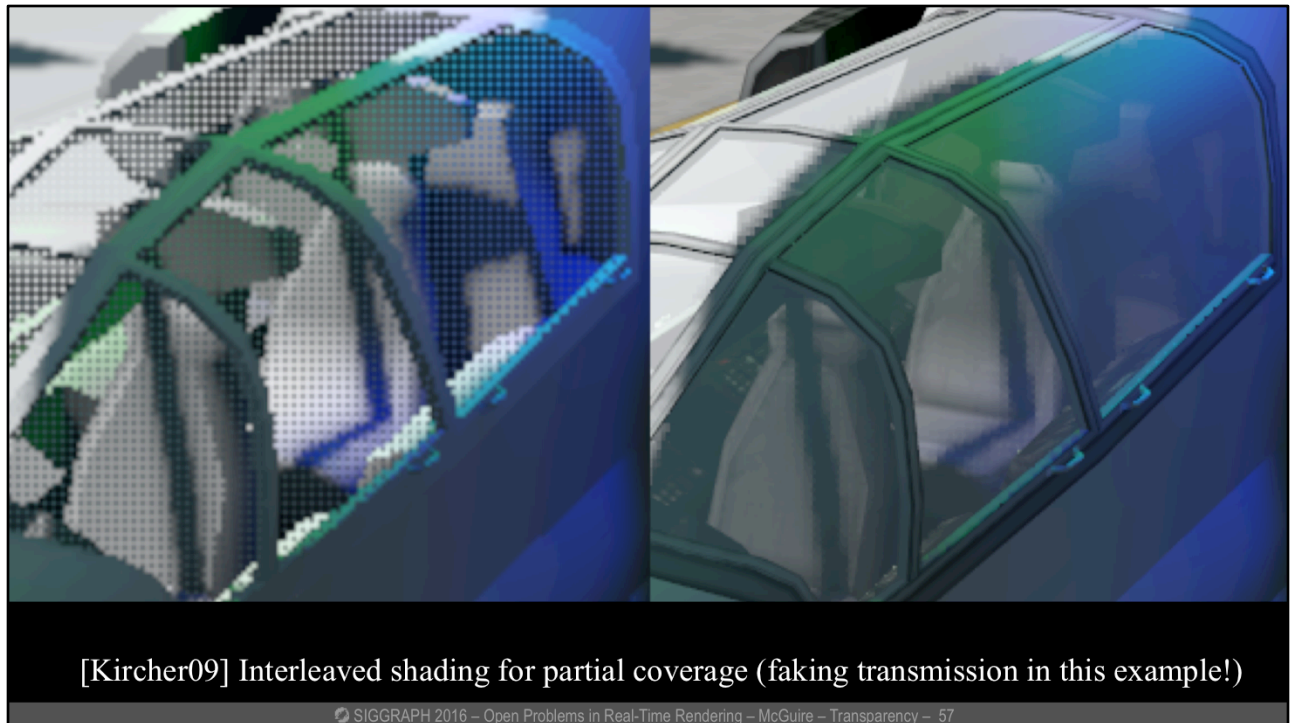
- Completely hallucinate the **visibility function** [Meshkin07]
- Makes **colored transmission** and **partial coverage** practical on current hardware [McGuire15]
- Allows some screen-space tricks for **refraction** and **diffusion**



Console-friendly order-indepdenent transparency with no atomics or pixel sync.  
Also fake refraction, diffusion,

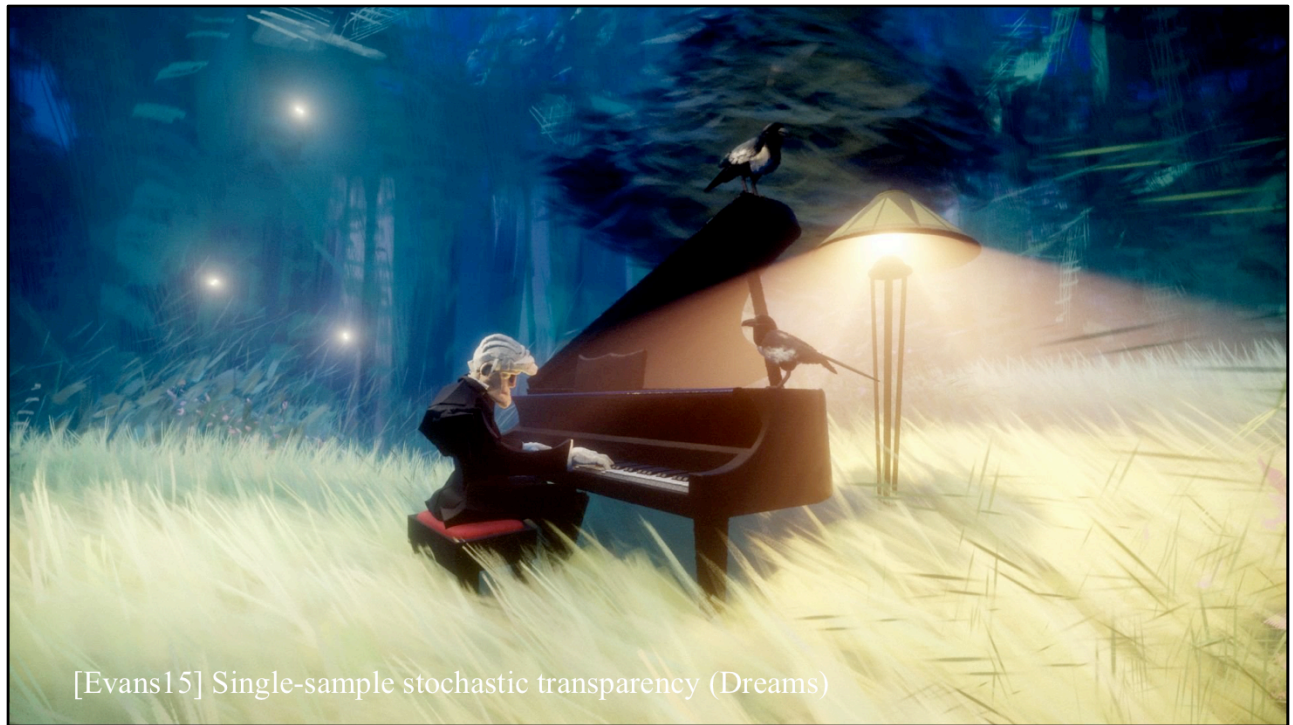
# Stochastic Transparency

- Bulletproof **coverage** strategy
- Works for indirect light and shadows
- Typically requires high-rate MSAA
- Inferred lighting for deferred shading [Kircher09]
- 1spp + temporal antialiasing [Evans15]



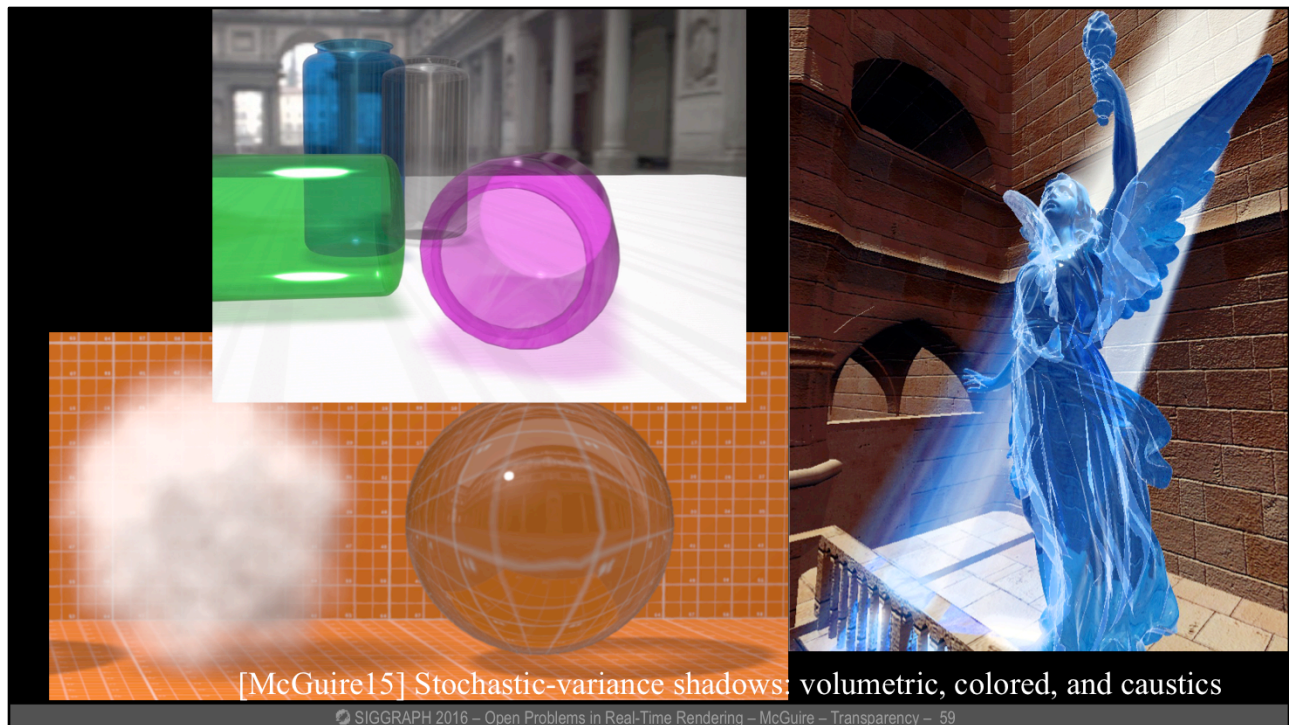
Very popular for game hidden-surface rendering, LOD transitions, and some particle effects.



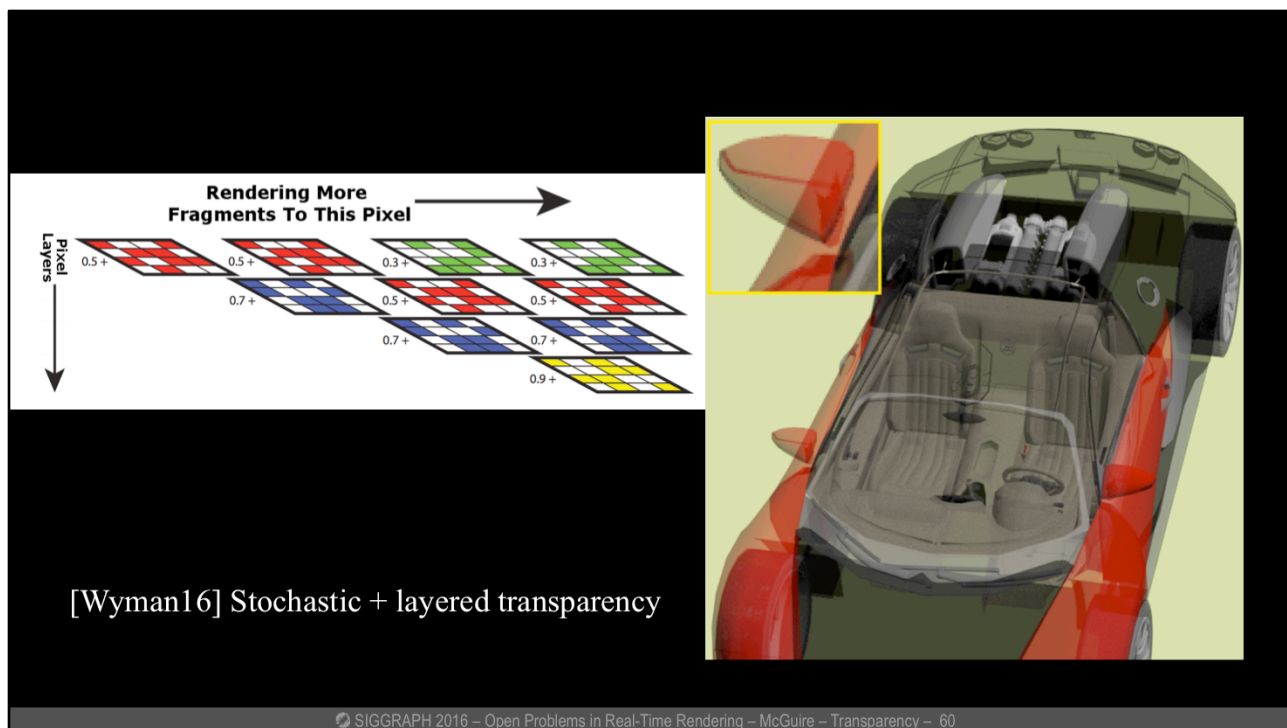


[Evans15] Single-sample stochastic transparency (Dreams)





Colored shadows, caustics, and volumetric lighting and shadows using a stochastic variance shadow map



Valuable in two ways: new way of thinking about partial coverage algorithms, and a new proof-of-concept algorithm.  
 I think that the notion of mixing stochastic with k-buffers is a possible path to stochastic transparency that also supports physically-based transmission

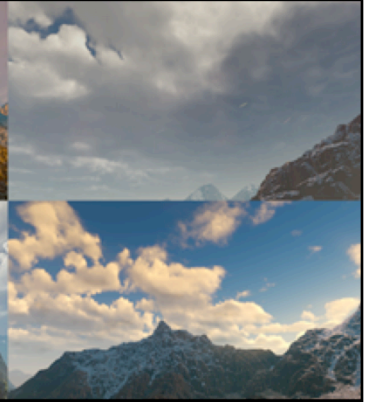
# Volumetric



[Tatarchuk15] Destiny



[Schneider15] Horizon: Zero Dawn





[Evans15] Analytic volume (Dreams)

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 63

Stochastic transparency on objects, also using epipolar-style ray march integration for uniform volumetrics with some noise



Shadow map -> extrude light volumes, assume uniform density medium and solve the integral





Ray marching at low resolution. Cleverness in compositing with other transparents by using a very low resolution voxel representation of coverage

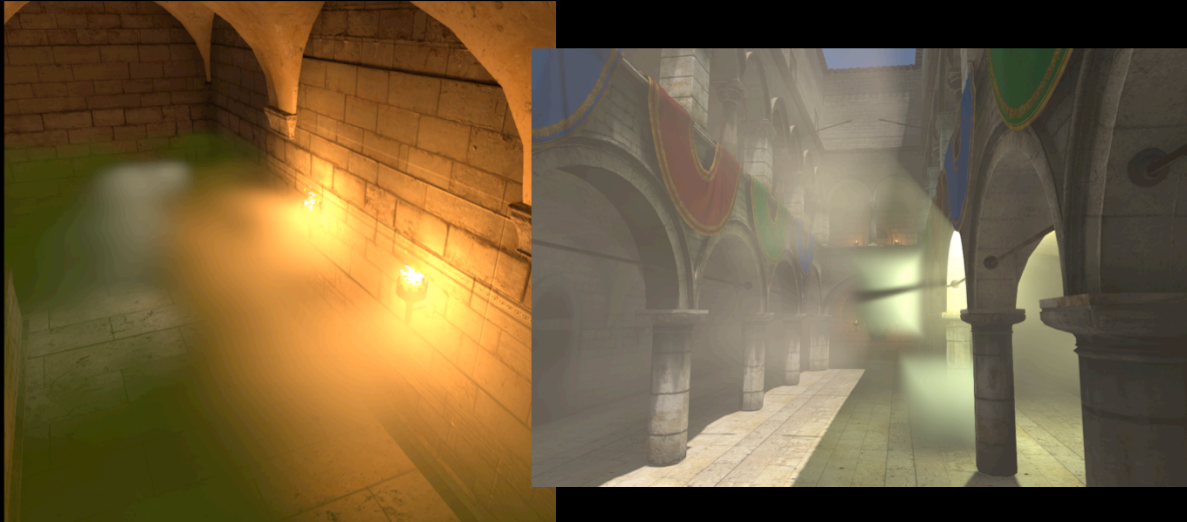


Camera-space ray marching with stable sampling depths. The idea is to compute at low resolution along the z-axis, but make those samples relatively coherent between frames so that the aliasing is at least consistent over time.

Bowles showed a simple proof of concept at SIGGRAPH last year. I think there's a lot of potential here...ray marching is the one high-quality method that we know rendering of dense transmissive media like fog, and this gives a large speedup while concealing the major error source.



[Hillaire15] volumetric (Frostbite)



SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 67

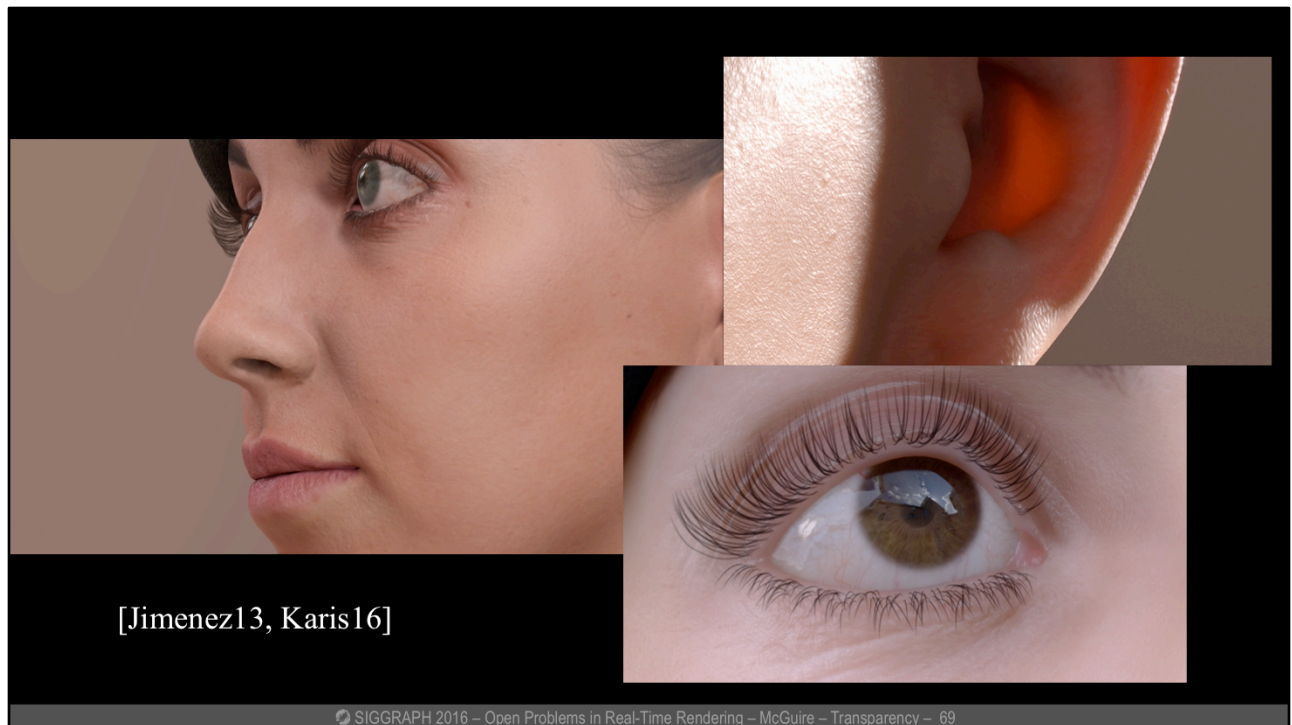
Scattering, emission, extinction, albedo... (this is describing materials that contain lots of small particle, so are inhomogeneous) stored in camera-space voxels. Primarily on fog volumes with noise textures, but also some particles.

Integrate lighting along rays, with some tricks for in/out scattering based on knowledge of the volumes

State of the art. Desired features: fix some leaking, not need special modeling knowledge about the volumes and lights, integrate with other non-fog like material transparency.

# Faces

- Very sensitive cases
- Does not interact with other transparents
- Localized, output can be treated as an “opaque” surface



Special cases for human figures, which we're very perceptually sensitive to.

Skin inspired by d'Eon 2007...screen space blurring, red blurs further than other colors, lots of ambient occlusion darkening.

Also very careful handling of reflection and refraction in eyes.

# Post-Process Effects

- Motion Blur
- Depth of Field
- Bloom
- Lens Flare



[Jiminez14, Guertin14] Post-process motion blur (Call of Duty Advanced Warfare)



[Jiminez14] Post-processed depth of field (Call of Duty Advanced Warfare)



[Jiminez14] Post-processed bloom (Call of Duty Advanced Warfare)

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 73

# Strategies

- Ray Tracing
  - Bulletproof solution for transmission, but currently expensive and requires a different data structure than rasterization. **Embree/FireRays/OptiX** can trace one ray per pixel at 720p in about 10 ms (Imagination has a mobile hardware ray tracer, but not provisioned for full-screen)
  - Challenges with co-planar surfaces: modelling is actually hard for any transmissive surfaces
- Stochastic “screen door” [for partial coverage]
  - Bulletproof solution for coverage (including edge antialiasing, shadows, motion blur, and depth of field) only
  - High-rate MSAA or post-process antialiasing/reconstruction needed
  - Some tricks for colored transmission to fake Beer-Lambert absorption for shadows, reducing noise in the coverage case using multiple passes
  - Doesn’t help with transmission/refraction
  - Alpha-to-coverage
  - [Special cases, e.g., inferred]
  - Lspp + TAA can work for some content (Dreams PS4)
  - **Wyman HPG’16**
- Smooth models of transmission-along-a-ray
  - Sorted transparency
    - Primitive sorting is slow or impossible, especially on GPUs...and unbounded runtime issues
    - Requires changing blending modes and shaders frequently for reasonably correct combinations of partial coverage, transmission, reflection, refraction, and emissives
    - Works everywhere, state of the art in many cases!
    - Best bet for reasonable screen-space refraction
  - *k*-buffer Blending
    - Pragmatic improvement over A-buffer, Z3 buffer
    - (sometimes,  $k = 1!$ ): WBOT, over/under, etc.
    - Generally, no refraction (phenomenological tries to fake refraction)
    - Wyman recently unified with stochastic for the coverage-only case
    - State of the art
    - **Maule et al., Hybrid transparency, ISD’13** (uses atomic min, doesn’t need pixelsync; mix order independent and order dependent; bias-correction similar to stochastic)
    - $K \leq 4$  **Sab’i and Vadyanathan, Multi-layer alpha blending, ISD’14**
    - **McGuire and Mara, Phenomenological Transparency, ISD’16** (uses stochastic for shadows!)
  - Volumetric
    - Various screen, camera, light, and world space methods
    - Work well for “participating medium” cases like fog and smoke, but may lack fine detail
    - (froxel, media molecule)
    - Tend to leak through walls
    - Screen-space god rays
    - **Sebastien Hillaire’s Froxels**
    - **COD/Jimenez skin**
- **Post-processed bloom, DOF, Motion blur (see COD talk)**



	Performance	Console	Coverage	Thin Transmission	Thick Transmission	Shadows & Caustics	Correctness
Ray Tracing			✓	✓	✓	✓	✓
Voxel	✓	✓			✓	✓	✓
Stochastic	✓	✓	✓			✓	✓
<i>k</i> -buffer	✓	✓	✓		✓		✓
Weighted Blending	✓	✓	✓	✓	✓		
Sorted		✓	✓	✓			✓

**Long term** (top row)

**Short term** (bottom row)

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 75

Excepting the special case techniques, I might coarsely summarize the properties of each strategy like this.

And based on that, you can probably see why I'd call the bottom of this table better short-term strategies and the top better long term strategies.

# Topics

## Framing the Problem:

- **Transmission** is real, and *is* rendering
- **Partial coverage** is artificial and trades one problem for another

## Framing the Solution:

- Constraints
- Transparency **Strategies** & Leading **Approximations**
-  • Future **research agenda**

I'll now conclude with two slides of recommendations for how I'd approach short and long-term transparency research and development

# Two R&D Agendas

Recommendations for approaching transparency  
in three- and ten-year windows

# Shape of Solutions

- Agenda: *More* complete transparency solution
  - Consistent transmission functions for light and camera
  - No light/dark leaks
  - Refraction
  - Colored transmission
  - Mixing materials: looking through glass at smoke...and the camera in the middle of them
- Good news: Lots of new room to experiment
  - DX12 Fragment shader interlock/pixel sync now shipping on all new vendor GPUs
  - Lots of recent research results on very different techniques
  - OptiX Prime/Embree/FireRays high-performance ray tracing solutions
  - Deferred k-buffer/deeper aggregates?
- Bad news:
  - For the next 5 years, most games must still run on current (XB1 & PS4) as well as upgraded consoles (Scorpio & NEO)
    - Techniques with platform-specific engineering are ok. Platform-specific art direction and tuning are too expensive
    - Room for disruption in console market?
    - Another window for indie PC games to pull ahead technically
- 10 years: probably more film-like
  - Ray tracing + stochastic super-sampling is very attractive. Need:
    - Faster ray tracers + spatial data structure builders
    - Deferred MSAA solution
    - Compressed MSAA (return of CSAA?)
    - Much better denoising (TAA)
  - Volumetric effects without leaks or overblurring: voxels or massive particle systems
  - Deep/k-buffers buffers for screen-space post processing & compositing: Some effects will probably always be post-processed depth of field, motion blur, lens flare, bloom

• SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 78

Coda: We call rendering phenomena “effects” when we have to special-case them because the renderer doesn’t just handle them implicitly. There was a time when “highlights” and “shadows” were effects...now we take them for granted. I want to take transparency for granted.

View-model filtering for DoF done today

# Production Through 2020

- **Transmission**
  - (Diffusion, refraction, aberration, colored shadows + backgrounds, multiple scattering, glass, water, fog)
  - **Sorted blending** and **screen-space ray cast**
  - **Phenomenological** transparency [McGuire16]
- **Partial coverage**
  - (Smoke, thin objects, hidden surface, MIP-mapped alpha mask, motion blur, gray shadows)
  - DX12: **Layered G-buffers** (**k-buffered**, stochastic, AGAA) [Maule12, Salvi14, Crassin15, Wyman16]
  - Current-gen: **Phenomenological** [McGuire16]
  - Polygon and texture mask antialiasing: post-processed antialiasing (TAA, FXAA, SMAA, etc.)
- Dense media via **voxels** and particles deposited in voxels [Hillaire15, Vos14]
- Skin and eyes as special cases [Jiminez13]
- Antialiasing, Depth of Field, Bloom, Motion Blur, Vignette, and Lens Flare in **post** [Jiminez14]

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 79

If I was in preproduction on a game right now, I'd be looking to extend these techniques. There's still room between them for innovation within the constraints of current GPU architectures.

The key ideas of working with sparse sampling in depth and ensuring temporal stability even at the cost of biasing the result are really important. Wyman and Maule's hybrid ideas haven't been sufficiently explored either.

For a longer term or academic research agenda, I think the solutions might look very different, however...

# Next Decade Research Target

- Transmission: **Ray trace** triangles and ray march voxels
- Partial coverage: **Stochastic** or **layered G-buffer**
- Either 64x super-sampling or very good denoising/antialiasing filters
- **Layered** Forward+/G-buffers produced by all rendering for use in post-processing

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 80

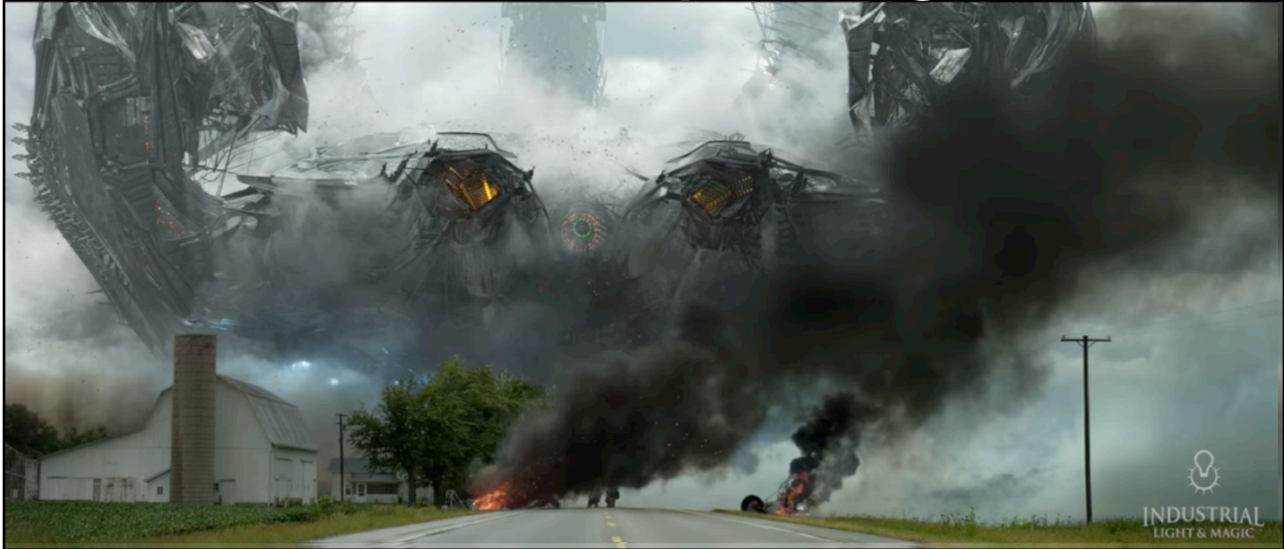
I think we're absolutely going to be ray tracing transmission (and probably sharp reflections) within a decade in production. There are a lot of open sub problems on how to get us there on both the software and hardware side, and if you look at the HPG proceedings for the past few years, you'll see that this work is already in progress.

Ray tracing and ray marching are so robust and flexible that there's no way we'll settle for a pile of hacks once they are performant. There's a lot of work to be done on denoising and upsampling the results to get the most out of every ray, however!

We might see a comeback of MSAA or supersampling to help with this. That makes partial coverage work with deferred effects...essentially, by going back to binary coverage!

...but transmission isn't ever going to work with purely stochastic methods, so I believe that just as film renderers produce multiple layers for compositing and post, we're going to produce layered G-buffers for use in post processing and compositing. While there's lots of good work to do on further tweaks to the post-processing pipeline, I wouldn't pursue doing any of those effects in-camera except maybe stochastic motion blur. Film is satisfied with the quality of post-processing these and I don't see us having higher standards.

# Film Compositing



Transformers Age of Extinction FX reel, ILM 2014 <https://www.youtube.com/watch?v=btn2tPxjisA>

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 81

# Next Decade Research Target

- Transmission: **Ray trace** triangles and ray march voxels
- Partial coverage: **Stochastic** or **layered G-buffer**
- Either 64x super-sampling or very good denoising/antialiasing filters
- **Layered** Forward+/G-buffers produced by all rendering for use in post-processing
- Denoising, Antialiasing, Bloom, Depth of Field, Motion Blur, Vignette, and Lens Flare in **post**  
(maybe stochastic motion blur...)
- *What about AR & VR **latency**?*

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 82

I think we're absolutely going to be ray tracing transmission (and probably sharp reflections) within a decade in production. There are a lot of open sub problems on how to get us there on both the software and hardware side, and if you look at the HPG proceedings for the past few years, you'll see that this work is already in progress.

Ray tracing and ray marching are so robust and flexible that there's no way we'll settle for a pile of hacks once they are performant. There's a lot of work to be done on denoising and upsampling the results to get the most out of every ray, however!

We might see a comeback of MSAA or supersampling to help with this. That makes partial coverage work with deferred effects...essentially, by going back to binary coverage!

...but transmission isn't ever going to work with purely stochastic methods, so I believe that just as film renderers produce multiple layers for compositing and post, we're going to produce layered G-buffers for use in post processing and compositing. While there's lots of good work to do on further tweaks to the post-processing pipeline, I wouldn't pursue doing any of those effects in-camera except maybe stochastic motion blur. Film is satisfied with the quality of post-processing these and I don't see us having higher standards.



# Acknowledgements

- Natalya Tatarchuk (Bungie)
- Aaron Lefohn, Chris Wyman, Marco Salvi, Ward Lopes (NVIDIA)
- Sébastien Hillaire (Frostbite)
- The computer graphics community on Twitter
- <http://www.exploratorium.edu/snacks/laser-jello>
- <http://refractiveindex.info>

[Tatarchuk15]

DESTINY

SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 83

# BIBLIOGRAPHY OF SELECTED REAL-TIME TRANSPARENCY PUBLICATIONS

**Highlighted** techniques mentioned explicitly in the talk.

# k-buffer and Peeling

- **Salvi, Temporal Supersampling, GDC 2016**
- **Salvi and Viadyanathan, Multi-layer alpha blending, I3D'14**
- Vasilakis and Fudos, k+-buffer: Fragment synchronized k-buffer, I3D'14
- Qin et al, Cone tracing for furry object rendering, TVCG, 2014
- Salvi, Pixel synchronization: solving old graphics problems with new data structures, SIGGRAPH'13 Real-Time Rendering Course
- **Maule et al., Hybrid transparency, I3D'13**
- Zhang, Memory-hazard-aware k-buffer algorithm for order-independent transparency rendering, TVCG 2013
- Salvi and Vidimče, Surface based antialiasing, I3D'12
- Knowles et al., Efficient layered fragment buffer techniques, in OpenGL Insights, 2012
- Salvi et al., Adaptive transparency, HPG'11
- Maule et al., A survey of raster-based transparency techniques, C&G, 2011
- Liu et al., Multilayer depth peeling via fragment sort, IEEE CAD'09
- Liu et al, Efficient depth peeling via bucket sort, HPG'09
- Bavoil et al., Multi-fragment effects on the GPU using the k-buffer, I3D'07
- Myers and Bavoil, Stencil routed k-buffer, NVIDIA tech report, 2007
- Mark and Proudfoot, The F-buffer: a rasterization-order FIFO buffer for multi-pass rendering, GH'01
- Everitt, Interactive Order-Independent Transparency, NVIDIA tech report 2001
- Jouppi and Chang, Z3: An economical hardware technique for high-quality antialiasing and transparency, HWWS'99
- Salesin and Stolfi, Rendering CSG models with a ZZ-buffer, SIGGRAPH'90
- Salesin and Stolfi, The ZZ-Buffer: a simple and efficient rendering algorithm with reliable antialiasing, PIXIM'89
- Carpenter, The A-buffer, an antialiased hidden surface method, SIGGRAPH 1984

# Blending

- **El Garawany, Deferred Lighting in Uncharted 4, SIGGRAPH'16 Real-Time Rendering Course**
- **Gonzalez-Ochoa, Rendering Rapids in Uncharted 4, SIGGRAPH'16 Real-Time Rendering Course**
- **McGuire and Mara, A phenomenological scattering model for order-independent transparency, I3D'16**
- Glassner, Interpreting alpha, JCGT 2015
- Tatarchuk, Applied graphics research for video games, I3D'15 Keynote
- Tatarchuk et al., Mythic science fiction in real-time: Destiny rendering engine, SIGGRAPH'13 Real-Time Rendering Course
- McGuire and Bavoil, Weighted blended order-independent transparency, JCGT 2013
- Bavoil and Myers, Order independent transparency with dual depth peeling, NVIDIA tech report 2008
- Sintorn and Assarsson, Real-time approximate sorting for self shadowing and transparency in hair rendering, I3D'08
- Filion and McNaughton, StarCraft III effects and techniques, SIGGRAPH'08 Real-Time Rendering Course
- Meshkin, Sort-independent alpha blending, GDC'07
- Gosselin et al., Real-time texture space skin rendering, GDC'04
- Smith and Blinn, Blue screen matting, SIGGRAPH'96
- Haeberli and Akeley, The accumulation buffer: hardware support for high-quality rendering, SIGGRAPH'90
- Porter and Duff, Compositing digital images, SIGGRAPH'84

# Stochastic & Coverage

- **Evans, Learning from failure, SIGGRAPH'15 Real-Time Rendering Course**
- Andersson et al., Filtered stochastic shadow mapping using a layered approach, CGF 2015
- Wyman, Exploring and expanding the continuum of OIT algorithms, HPG'16
- Wang et al., Decoupled coverage antialiasing, HPG'15
- Crassin et al., Aggregate G-buffer anti-aliasing, I3D'15
- Andersson et al., Adaptive texture space shading for stochastic rendering, CGF 2014
- Munkberg et al, Per-vertex defocus blur for stochastic rasterization, CGF'12
- Munkberg and Akenine-Moller, Hyperplane culling for stochastic rasterization, EG'12
- Akenine-Moller et al., Efficient depth of field rasterization using a tile test based on half-space culling, CGF'12
- Munkberg et al., Hierarchical stochastic motion blur rasterization, HPG'12
- Enderton et al., Stochastic transparency, TVCG 2011
- Laine et al., Clipless dual-space bounds for faster stochastic rasterization, SIGGRAPH'11
- McGuire and Enderton, Colored stochastic shadow maps, I3D'11
- Shirley et al, A local image reconstruction algorithm for stochastic rendering, I3D'11
- McGuire et al., Real-time stochastic rasterization on conventional GPU architectures, HPG' 10
- Enderton et al., Stochastic transparency, I3D'10
- Trebilco, Light indexed deferred rendering, ShaderX7, 2009
- **Kircher and Lawrance, Inferred lighting: fast dynamic lighting and shadows for opaque and translucent objects, Sandbox'09**
- Akenine-Moller et al., Stochastic rasterization using time-continuous triangles, GH'07
- Young, Coverage-sampled antialiasing, NVIDIA tech report 2006

# Volume & Participating Media (1/2)

- **Hoobler, Fast, flexible, physically-based volumetric light scattering, GDC'16**
- **Hillaire, Towards unified and physically-based volumetric lighting in Frostbite, SIGGRAPH'15 Real-Time Rendering Course**
- Schneider, The real-time volumetric cloudscape of Horizon: Zero Dawn, SIGGRAPH'15 Real-Time Rendering Course
- **Bowles and Zimmermann, A novel sampling algorithm for fast and stable volume rendering, SIGGRAPH'15 Real-Time Rendering Course**
- **Jimenez, Next generation post processing in call of duty advanced warfare, SIGGRAPH'14 Real-Time Rendering Course**
- Tatarinov and Cantlay, From terrain to godrays, GDC'14
- Glatzel, Volumetric Lighting for Many Lights in Lords of the Fallen, Digital Dragons 2014
- **Wronski, Volumetric fog: unified compute shader-based solution to atmospheric scattering, SIGGRAPH'14 Real-Time Rendering Course**
- **Valient, Reflections and volumetrics of Killzone Shadow Fall, SIGGRAPH'14 Real-Time Rendering Course**
- **Jimenez et al., Next generation character rendering, GDC'13**
- **Yusov, Practical implementation of light scattering effects using epipolar sampling and 1D min/max binary trees, GDC'13**
- Vos, Volumetric Light Effects in Killzone: Shadow Fall, GPU Pro 5, 2014
- Wyman and Dai, Imperfect voxelized shadow volumes, HPG'13
- Billeter et al., Real-time multiple scattering using light propagation volumes, I3D'12
- Evans, Two uses of voxels in LittleBigPlanet2's graphics engine, SIGGRAPH'11 Real-Time Rendering Course
- Wyman, Voxelized shadow volumes, HPG'11
- Chen et al., Real-time volumetric shadows using 1D min-max mipmaps, I3D'11

## Volume & Participating Media (2/2)

- Thiedmann et al., Voxel-based global illumination, I3D'11
- Salvi et al., Adaptive volumetric shadow maps, EGSR'10
- Kaplanyan and Dachsbacher, Cascaded light propagation volumes for real-time indirect illumination, I3D'10
- Jansen and Bavoil, Fourier opacity maps, I3D'10
- Engelhardt and Dachsbacher, Epipolar sampling for shadows and crepuscular rays in participating media with single scattering, I3D'10
- Billeter et al., Real time volumetric shadows using polygonal light volumes, HPG'10
- Toth and Umenhoffer, Real-time volumetric lighting in participating media, EG'09
- Sintorn and Assarsson, Hair self shadowing and transparency depth ordering using occupancy maps, 2009
- Hable, Fast skin shading, ShaderX7 2009
- Wyman and Ramsey, A hybrid method for interactive shadows in homogeneous media, ShaderX7 2009
- Tatarchuk et al. Advanced interactive medical visualization on the GPU, J. Par. Dist. Com. 2008
- Wyman and Ramsey, Interactive volumetric shadows in participating media with single-scattering, IRT'08
- Sousa, Crysis next gen effects, GDC'08
- d'Eon et al., Efficient rendering of human skin, EGSR'07
- Mitchell, Volumetric light scattering as a post process, GPU Gems 3, 2007
- Sun et al., A practical analytic single scattering model for real time rendering, SIGGRAPH'05
- Tatarchuk, Advances in real-time skin rendering, GDC'05
- Ernst et al., Interactive rendering of caustics using interpolated warped volumes, GI'05
- O'Neil, Accurate atmospheric scattering, GPU Gems 2, 2005
- Lokovic and Veach, Deep shadow maps, SIGGRAPH'00

© SIGGRAPH 2016 – Open Problems in Real-Time Rendering – McGuire – Transparency – 89

# Other Motion Blur & Depth of Field

- Viadyanathan et al., Layered light field reconstruction for defocus blur, ToG 34:2, 2014
- **Guertin et al., A fast and stable feature-aware motion blur filter, HPG'14**
- Wronski, Bokeh depth of field, blog post, 2014
- Munkenberg et al, Layered reconstruction for defocus and motion blur, EGSR'14
- **Sousa, CryEngine3 graphics gems, SIGGRAPH'13 Real-Time Rendering Course**
- Viadyanathan et al., Adaptive image space shading for motion and defocus blur, HPG'12
- Tzeng et al., High-quality parallel depth-of-field using line samples, HPG'12
- McGuire et al, A reconstruction filter for plausible motion blur, I3D'12
- Ragan-Kelley et al, Decoupled sampling for graphics pipelines, ToG 2011
- White and Barre-Brisebois, More performance!, SIGGRAPH'11 Real-Time Rendering Course
- Shishkovtsov and Rege, Metro 2033, GDC'10
- Padget, Efficient real-time motion blur for multiple rigid objects, ShaderX7, 2009
- Lonroth and Unger, Advanced real-time post-processing using GPGPU techniques, MS thesis, 2009
- Hammon, Practical post-process depth of field, GPU Gems 3, 2007
- Rosado, Motion blur as a post-process effect, GPU Gems 3, 2007
- Lefohn, Glift: generic data structures for graphics hardware, PhD thesis, UC Davis 2006
- Kass et al., Interactive depth of field using simulated diffusion, Pixar tech report, 2006
- Hensley et al., Fast summed-area table generation and its applications, EG'05
- Scheuermann and Tatarchuk, Advanced depth of field rendering, ShaderX3, 2004
- Riguer et al, Real-time depth of field simulation ShaderX2, 2003
- Tatarchuk, Motion Blur using geometry and shading distortion, ShaderX2, 2003



# Other Refraction

- Ganestam and Doggett, Real-time multiply recursive reflections and refractions using hybrid rendering, Visual Computer 2015
- McGuire and Mara, Efficient GPU screen-space ray tracing, JCGT 2014
- De Rousiers et al., Real-time rough refraction, I3D'11
- Muller et al., Screen space meshes, SCA'07
- Davis and Wyman, Interactive refractions with total internal reflection, GI'07
- Wyman, Interactive image-space refraction of nearby geometry, GRAPHITE'05
- Wyman, An approximate image-space approach for interactive refraction

# Ray Tracing & Related

- Amstutz et al., An evaluation of multi-hit ray traversal in a BVH using existing first-hit/any-hit kernels, JCGT 2015
- Wald et al., Embree: a kernel framework for efficient CPU ray tracing, ToG 2014
- Gribble et al., Multi-hit ray traversal, JCGT 2014
- Laine et al., Megakernels considered harmful: wavefront path tracing on GPUs, HPG'13
- Parker et al., OptiX: a general purpose ray tracing engine, SIGGRAPH'10
- McGuire and Luebke, Image space photon mapping, HPG'09
- Wyman and Nichols, Adaptive caustic maps using deferred shading, CGF 2009
- Wyman, Hierarchical caustic maps, I3D'08
- Wyman and Dachsbacher, Reducing noise in image space caustics with variable sized-splatting, JGT'08
- [there are *many* more real-time ray tracing papers!]

# Analytic

- Gribel et al., Theory and analysis of higher-order motion blur rasterization, HPG'13
- Barringer et al., High-quality curve rendering using line sampled visibility, SIGGRAPH Asia'12
- Gribel et al., High-quality spatio-temporal rendering using semi-analytic visibility, ToG 2011
- Gribel et al, Analytical motion blur rasterization using compression, HPG'10
- Coy, Simplified high quality antialiased lines, ShaderX7 2009
- Wu, Fast antialiased circle generation, Graphics Gems II, 1991
- Wu, An efficient antialiasing technique, CG 1991

# Referenced Offline Algorithms

- Jarosz et al., Progressive photon beams, SIGGRAPH Asia'11
- **Jarosz et al., The beam radiance estimate for volumetric photon mapping, EG'08**
- **Jensen et al., A practical model for subsurface light transport, SIGGRAPH'01**
- **Jensen, Realistic image synthesis using photon mapping, 2001**
- Veach, Robust Monte Carlo methods for light transport simulation, PhD Thesis, Stanford 1998
- **Veach and Guibas, Metropolis light transport, SIGGRAPH'97**
- Jensen, Global illumination using photon maps, EGSR'96
- **Veach and Guibas, Optimally combining sampling techniques for Monte Carlo rendering, SIGGRAPH'95**
- Lafortune and Willems, A theoretical framework for physically based rendering, 1994
- **Lafortune and Willems, Bidirectional path tracing, CompuGraphics'93**
- Kajiya, The rendering equation, SIGGRAPH'86
- Whitted, An improved model for shaded display, CACM 1980

[these techniques are refined and applied in hundreds of other papers]



Attribution 4.0 International

Slides copyright 2016 Morgan McGuire, [mcguire@cs.williams.edu](mailto:mcguire@cs.williams.edu)  
Reproduction and distribution permitted under the CC-BY 4.0 license  
<https://creativecommons.org/licenses/by/4.0/>

In accordance with USA Fair Use provisions for academic review and education,  
this presentation contains images copyrighted by others. The attributed authors for  
each retain their copyrights and I assert no claims on that content.