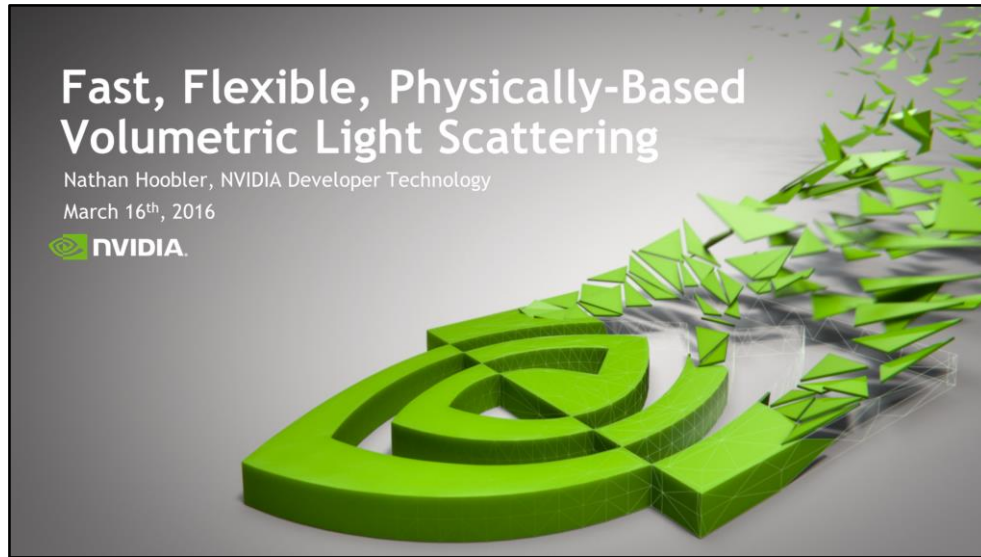


Fast, Flexible, Physically-Based Volumetric Light Scattering

Nathan Hoobler, NVIDIA Developer Technology

March 16th, 2016









Polygonal Volumetric Lighting

Fast, Flexible, and Physically-based

Fast: Scalable performance on all hardware

Flexible: Minimally invasive to integrate into existing engines

Physically-based: Easy mapping to real-world phenomenon

Source will be available on GitHub to registered developers



gameworks.nvidia.com

5 

Fast:

- Tunable quality knobs to provide best performance for a variety of platforms
- Scales well with large numbers of emitters

Flexible:

- Requires minimal modification of existing pipeline
- Works anywhere you can provide a shadow map and a scene depth
- Flexible lighting model to accommodate specific pipeline quirks (complex falloff functions, etc.)

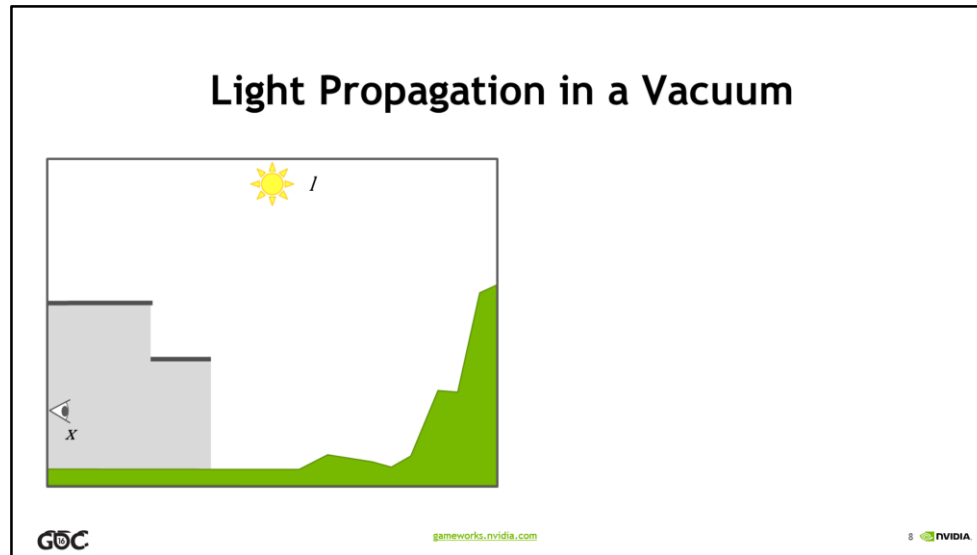
Physically-based:

- Lighting model derived from physical integrals, making it suitable for integration into physically-based renderers
- Media properties derived from real-world models, making it straightforward to get a 'realistic' starting point
- ... but still flexible enough to give artists a simple interface for tweaking

AGENDA

Background & Motivation
Algorithm Overview
Integration into *Fallout 4*

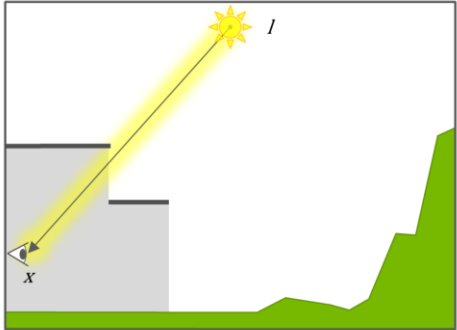
Background & Motivation



Basic scene - Light, World, and Viewer looking at lit and shadowed areas

Light Propagation in a Vacuum

Directly Visible Light-Source



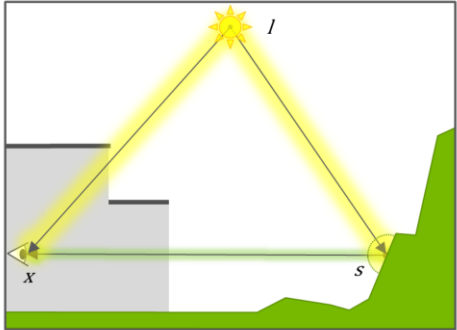
Direct Radiance:
 $L_D = L(\omega_x)$

GDC
gameworks.nvidia.com
9 NVIDIA



Light directly visible by the viewer

Light Propagation in a Vacuum

Direct Illumination



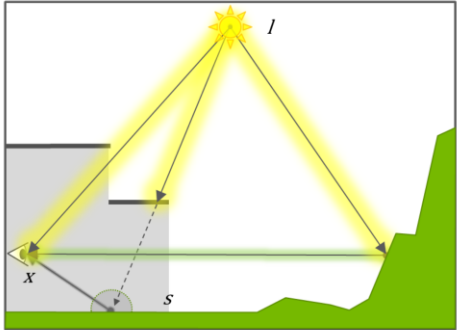
Direct Radiance:
 $L_D = L(\omega_s)\rho_s(l, s, \omega_x)$

 gameworks.nvidia.com
10 

Light that is reflected by the scene towards the viewer



Light Propagation in a Vacuum

Shadows

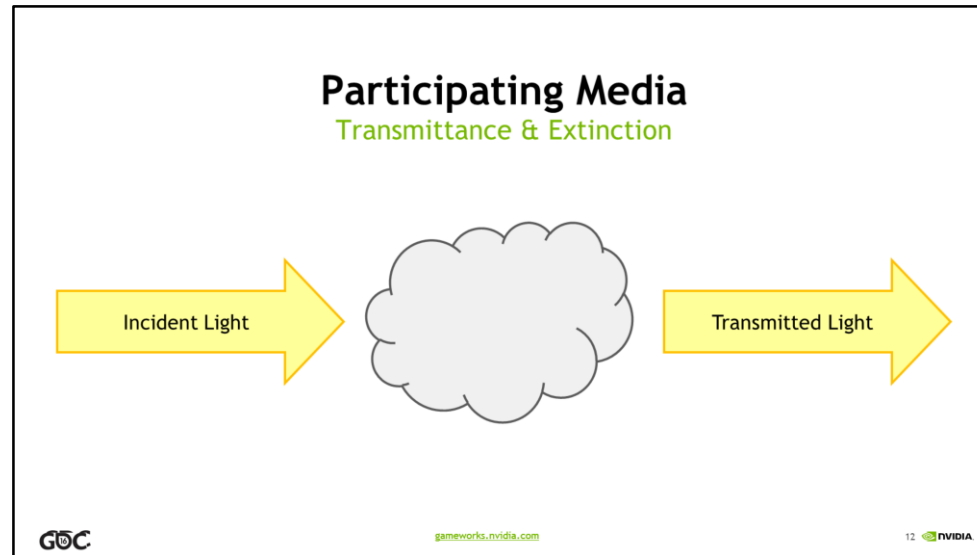


Direct Radiance:

$$L_D = L(\omega_s)\rho_s(l, s, \omega_x)V(s, l)$$

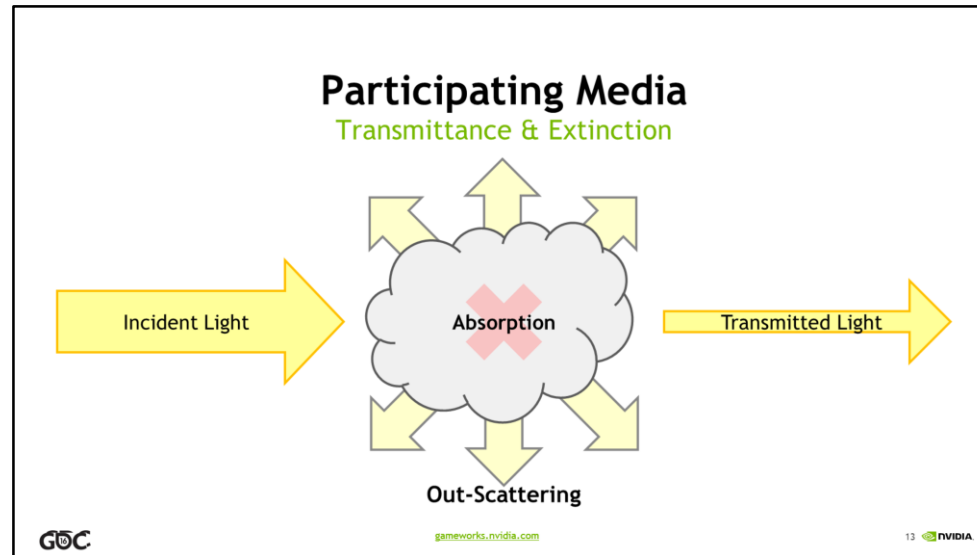

gameworks.nvidia.com
11 

Points in the scene where the light is not directly visible



This is all in a vacuum - the air affects the light too

Not just theoretical: scattering is an important visual cue for distance, grounds environment, adds 'weight' to shadows.



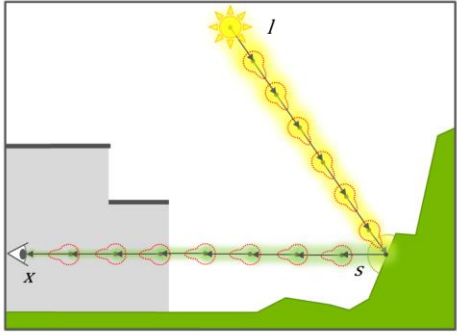
Light is attenuated in two ways

- Absorption
- Scattering

Together, the net effect is called “extinction”



Participating Media

Transmittance & Extinction



Direct Radiance:
 $L_D = L(\omega_s)\rho_s(l, s, \omega_x)V(s, l)$

Transmitted Radiance:
 $L_T = L_D T(l, s)T(s, x)$


gameworks.nvidia.com
14 

We want to know how much light traveling along a path makes it to the end point (Transmittance)
 We care about the transmittance from L to S, then from S to X.

Participating Media

Beer-Lambert Law

Transmittance:

$$T = 1 - \frac{\phi_s}{\phi_i} * \frac{\phi_a}{\phi_i} = 1 - \frac{\phi_{ex}}{\phi_i}$$

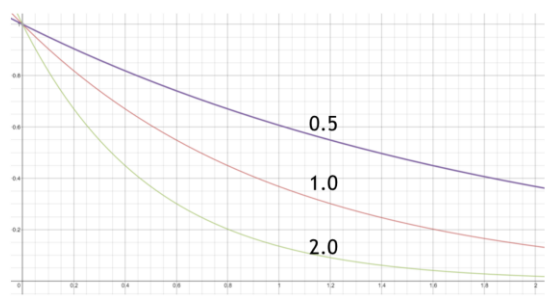
$$T(a, b) = e^{-\tau||b-a||}$$

Optical Thickness:

$$\tau_x = -\ln\left(1 - \frac{\phi_x}{\phi_i}\right)$$

$$\tau_{ex} = \tau_s + \tau_a$$

Transmittance vs Distance



gameworks.nvidia.com 15 NVIDIA

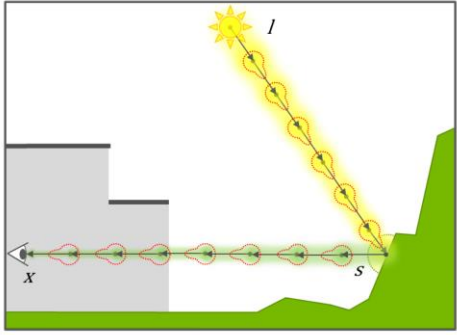
Transmittance: Ratio of power that passes out along a given ray to the power coming in along that ray over a given distance

Beer-Lambert Law: Solution to the differential equation gives us an exponential relative to the distance and thickness

Optical Thickness: Scale factor that tells how quickly the attenuation happens.




Participating Media

Transmittance & Extinction

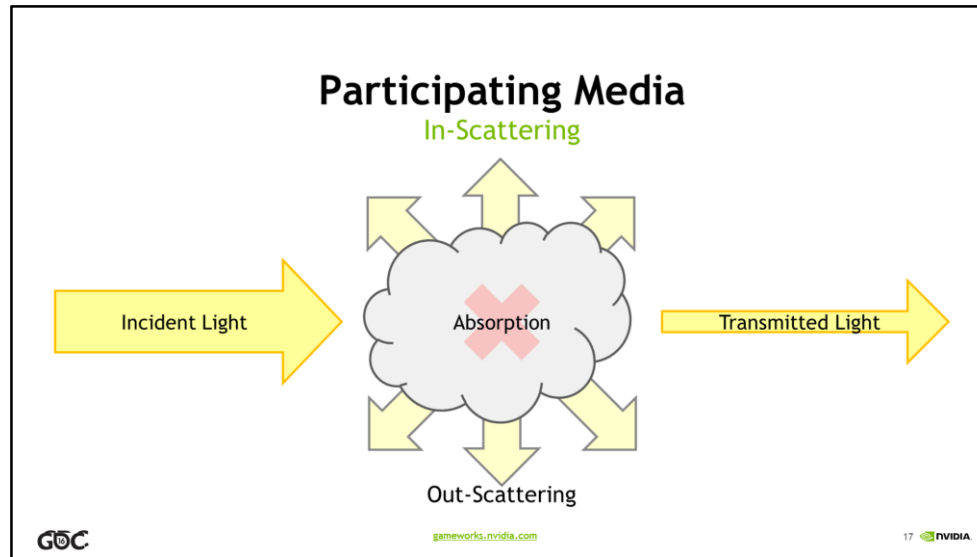


Direct Radiance:
 $L_D = L(\omega_s)\rho_s(l, s, \omega_x)V(s, l)$

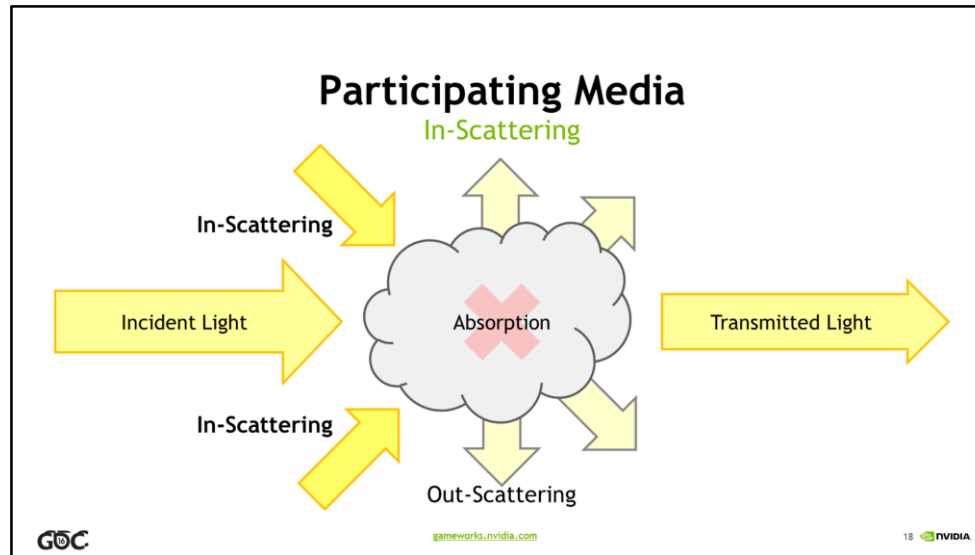
Transmitted Radiance:
 $L_T = L_D e^{-\tau_{ex}(\bar{l}s + \bar{s}x)}$

Using the Beer-Lambert law and a given optical thickness and distance, the transmittance becomes an exponential term.



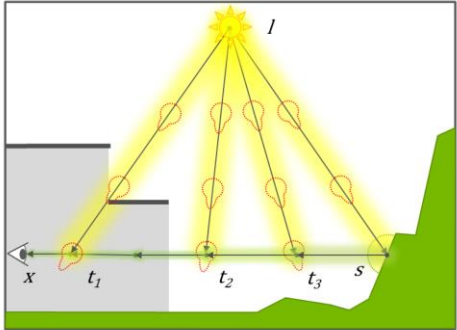
When light is scattered, it exits in a different direction from where it came in.



This means we don't only care about light traveling along the ray from the scene to the eye, but also the sum of all the light crossing that path and gets scattered towards the eye as well.

Participating Media



In-Scattering



Direct Radiance:
 $L_D = L(\omega_s)\rho_s(l, s, \omega_x)V(s, l)$

Transmitted Radiance:
 $L_T = L_D e^{-\tau_{ex}(\vec{l}s + \vec{s}x)}$

In-Scattered Radiance:
 $L_S = \int_0^{\vec{s}x} L(t, l)V(t, l)\rho_m(\vec{t}l, \omega_x) e^{-\tau_{ex}(\vec{l}t + \vec{t}x)} dt$


gameworks.nvidia.com
19 

The in-scattered light is the result of an integral over the view vector that accounts for the amount of light reaching that point, the angle between the viewer and the light direction, and the scattering distribution.

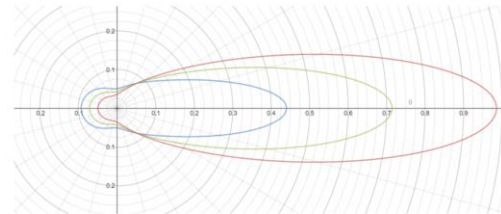
Phase Functions

Overview

Volume analog of BRDF

Directional distribution of energy relative to incident direction

Determined by the ratio of medium particles and their size relative to the light



Phase function for atmosphere on a clear day



gameworks.nvidia.com

20 

The Phase Function is a distribution of how light is scattered relative to the incoming direction. Determined by the individual particles in the medium and their proportions relative to one another and the light itself.

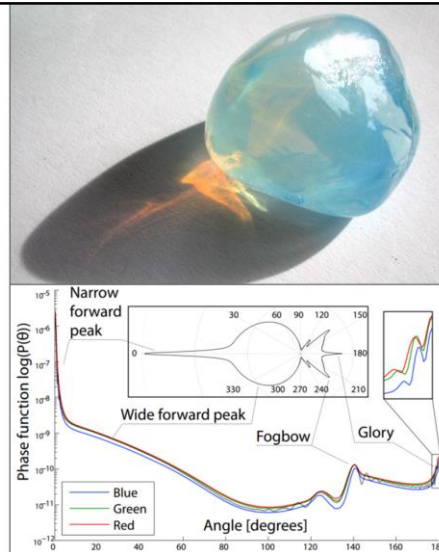
Phase Functions

Complex Phenomenon

Can be Wavelength-dependent

Can become arbitrarily complex depending on atmospheric conditions

We can approximate common conditions with simple models



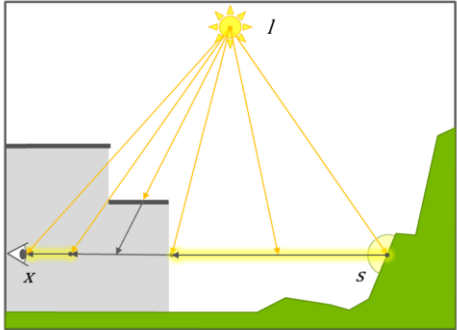
In the real world this can be incredibly complex due to things like irregular particle shapes (like ice crystals) or wavelength level effects.

For thin atmosphere (not clouds) and small distances (scenes, not whole atmospheres) we can approximate all this with simpler functions.

Algorithm Overview



Interval Integration

Overview



In-Scattered Radiance:

$$L_S = \int_0^{\overline{s\bar{x}}} L(t, l) V(t, l) \rho_m(\overline{tl}, \omega_x) e^{-\tau_{ex}(\overline{tl} + \overline{t\bar{x}})} dt$$

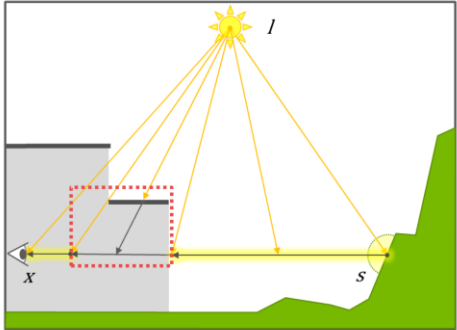
 GDC
gameworks.nvidia.com
23  NVIDIA

In theory we just need to solve the in-scattering integral along each view vector.

- Analytical solutions
- Ray-Marching = numerical integration



Interval Integration

Non-Constant Visibility



In-Scattered Radiance:

$$L_S = \int_0^{\bar{s}_x} L(t, \vec{l}) V(t, \vec{l}) \rho_m(\vec{t}, \omega_x) e^{-\tau_{ex}(\vec{t} + \vec{t}_x)} dt$$

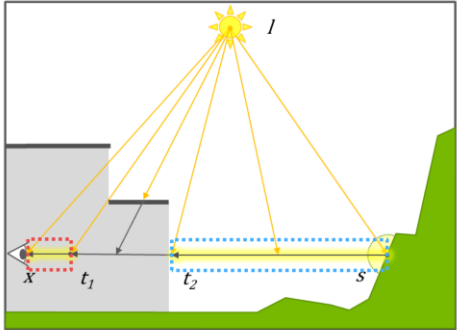
 GDC
gameworks.nvidia.com
24  NVIDIA

This gets complicated by the visibility term, which can be arbitrary. Some sections of the path may not be lit.

Ray marching has trouble because you need a lot of steps to capture occlusion features.

Interval Integration




Sum of Intervals



In-Scattered Radiance:

$$L_{S'}'(t) = L(t, l) \rho_m(\vec{l}, \omega_x) e^{-\tau_{ex}(\vec{l}, \vec{l}x)}$$

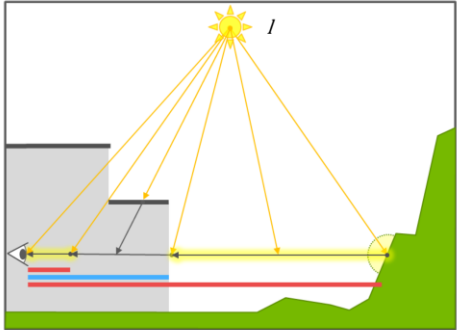
$$L_S = \int_x^{t_1} L_{S'}'(t) dt + \int_{t_2}^s L_{S'}'(t) dt$$

However, we observe that visibility is binary, so we can express the integral as the sum of integrals over the lit intervals and eliminate the visibility term.

Interval Integration

Sum & Difference of Intervals





In-Scattered Radiance:

$$L_{S'}'(t) = L(t, l) \rho_m(\vec{l}, \omega_x) e^{-\tau_{ex}(\vec{l} + \vec{l}_x)}$$

$$L_{S'}(d) = \int_0^d L_{S'}'(t) dt$$

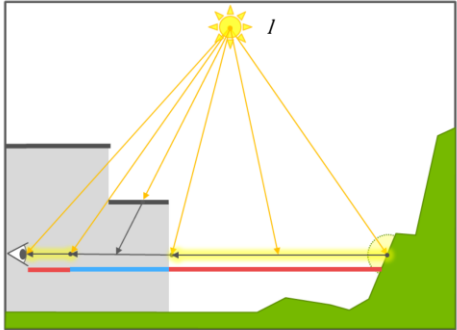
$$L_S = L_{S'}(t_1) - L_{S'}(t_2) + L_{S'}(s)$$


gameworks.nvidia.com
26 

We also see that the intervals can be expressed as the sum and differences a set of integrals starting at the eye.

Interval Integration

Sum & Difference of Intervals





In-Scattered Radiance:

$$L_{S'}'(t) = L(t, l) \rho_m(\vec{l}, \omega_x) e^{-\tau_{ex}(\bar{l} + \bar{l}x)}$$

$$L_{S'}(d) = \int_0^d L_{S'}'(t) dt$$

$$L_S = L_{S'}(t_1) - L_{S'}(t_2) + L_{S'}(s)$$

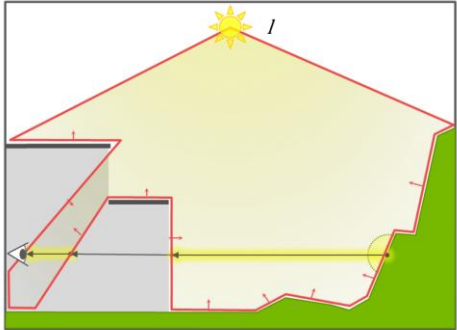
$$L_S = \sum_{n \in G_f} L_{S'}(t_n) - \sum_{m \in G_b} L_{S'}(t_m)$$


gameworks.nvidia.com
27 

We want some way to automatically figure out where these intervals are, rather than trying to sample the lighting function repeatedly and hoping we get it right as with ray marching.

Interval Integration



Intervals from Mesh



In-Scattered Radiance:

$$L_S = \sum_{n \in G_f} L_{S'}(t_n) - \sum_{m \in G_b} L_{S'}(t_m)$$

Use front and back faces of light volume as integration intervals


gameworks.nvidia.com
28 

We invert the process, by using the depth data stored in a light's shadow map to create a mesh that fills the lit volume. The intervals we add correspond to front faces, and the intervals we want to subtract correspond to back faces.



Here's a lit scene with no volumetric lighting.



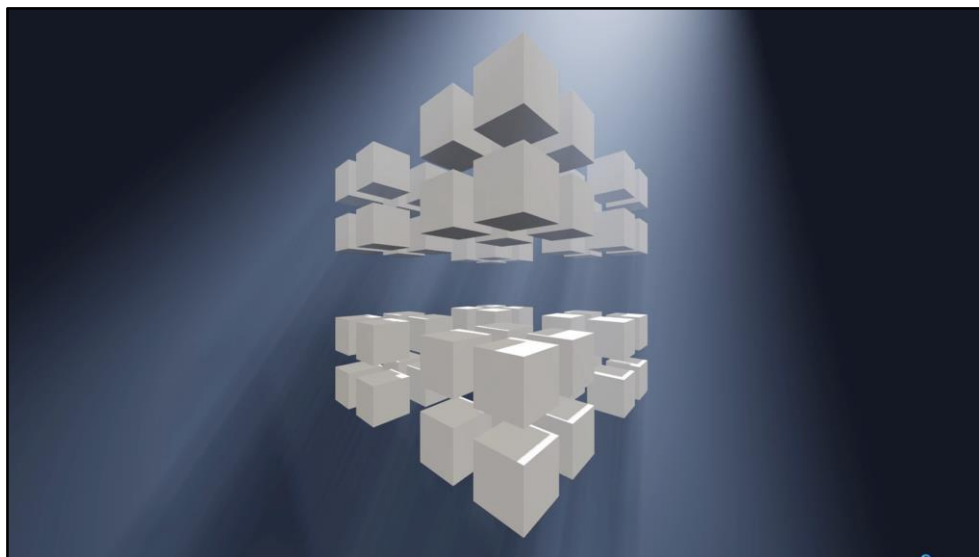
Here is the extruded volume laid over the image. We create a mesh that fills the light's frustum in clip space, tessellating the far plane and using the shadow map to displace the tessellated vertices causing it to match the scene geometry.



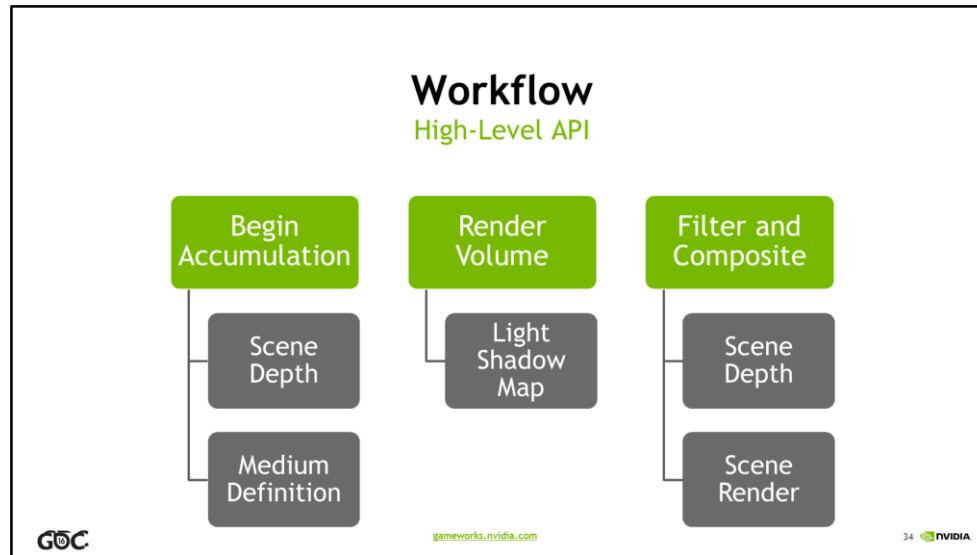
Here's the computed inscattering. In the pixel shader we compute the integral to the fragment depth, and then use blending to add or subtract the integral to an accumulation buffer.



The combined result

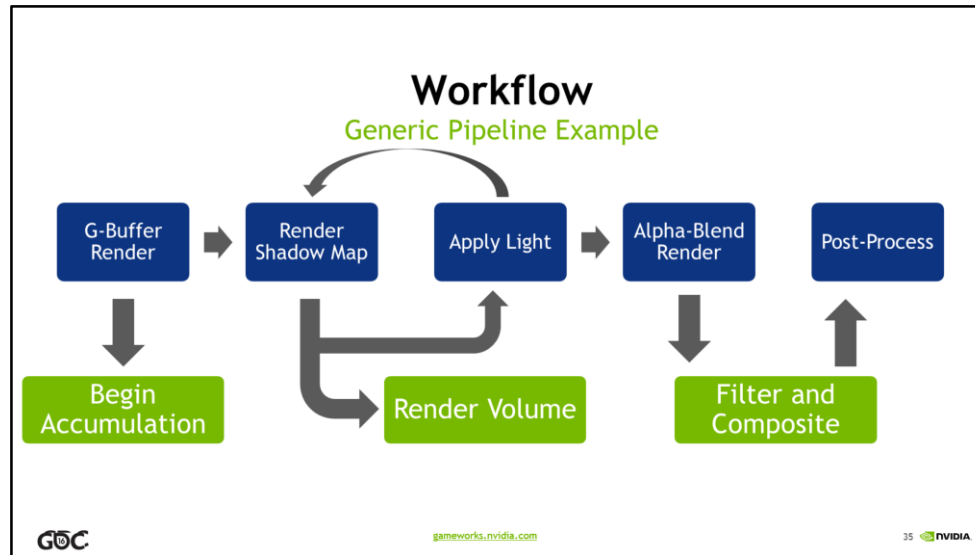


Animation showing the process in action



Three main hooks to the API.

The library needs a scene depth buffer and media description, then shadow map information and light descriptions for each light we want to accumulate.



Assumes deferred renderer

Only issue with forward renderer is that a depth pre-pass is needed

We want to accumulate after alpha-blended objects are rendered

Medium Specification

Multiple Phase Terms

Density - Optical depth of that term in <RGB>

Phase Function - Directional scattering distribution (applied per-channel)

Phase Parameters - Eccentricity (for Henyey-Greenstein)

Absorption - Optical depth of light absorption <RGB>

Terms are summed together per-channel to produce composite medium



gameworks.nvidia.com

36 

A medium is described by an absorption coefficient and a set of phase terms

We support different phase terms to express different media, then add them together for a composite volume

Medium Specification

Rayleigh Scattering

Particles much smaller than wavelength of light (O₂, N₂, etc.)

Generally constant at a given altitude

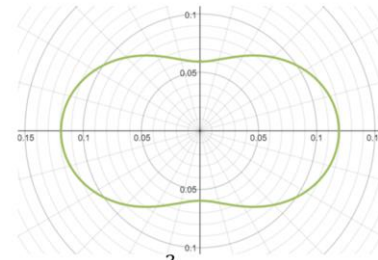
Wavelength dependent

Relative optical depth for CIE-RGB:

$$R: 0.596 \times 10^{-5}$$

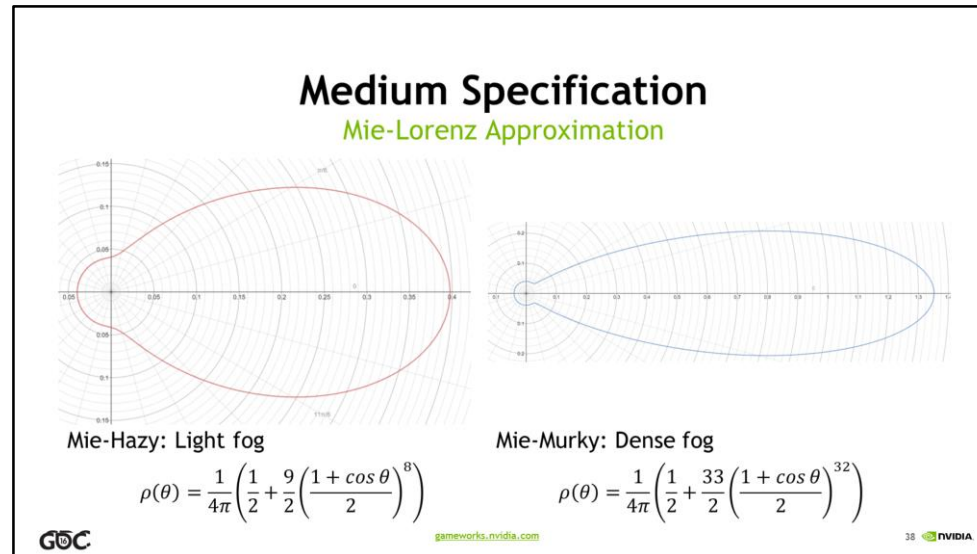
$$G: 1.324 \times 10^{-5}$$

$$B: 3.310 \times 10^{-5}$$



$$\rho(\theta) = \frac{3}{16\pi} (1 + \cos^2(\theta))$$

Rayleigh scattering describes pure “clean” air (particles much smaller than light wavelengths)



Mie scattering covers larger particles which have a large forward scattering component.

Medium Specification

Henye-Greenstein Function

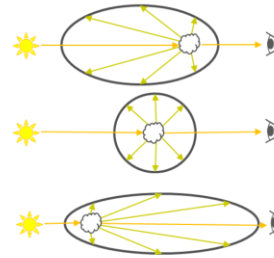
Tunable phase function with a specified eccentricity $g = (-1, 1)$

$g < 0$: back-scattering

$g = 0$: isotropic scattering

$g > 0$: forward scattering

Multiple terms can be combined to approximate complex functions



Henye-Greenstein scattering is useful for modeling arbitrary phase functions.
Sum multiple terms with different G values.

Volume Rendering

Overview

Convert light description + shadow map into geometry

Solve Integral at each intersection and add or subtract based on facing

Different solvers based on light type

Directional Light - Analytical Solution

Omnidirectional Light - Look-up Texture

Spot Light - Look-up Texture or Numerical Integration



gameworks.nvidia.com

40 

When lights are rendered we generate a mesh based on the Light/Shadow map description, then render it with a pixel shader that evaluates the integral based on the type of light.

Volume Rendering

Directional Light

$$L_{S'}(d) = L(d, l) \rho_m(\bar{x}l, \omega_x) \frac{1 - e^{-\tau_{ex}d}}{\tau_{ex}}$$

With some assumptions we can simplify the integral for directional light

- **Constant Direction** (parallel light-source)
- **Constant Power** (Light distance >> medium depth)

Reduces to analytic function evaluated in pixel-shader



Directional lights can be simplified to an analytic solution if we make some assumptions

Volume Rendering

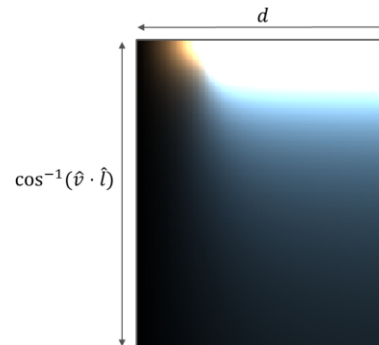
Omnidirectional Light

Local lights too complex for analytical solutions

All terms can be expressed in with respect to direction and distance

Create a Look-up texture:

- Map to 2D space
- evaluate differential
- Numerically integrate with CS



GDC

gameworks.nvidia.com

42 NVIDIA

Omnidirectional lights are solved using a look-up texture we generate that contains a numerically integrated solution.

Volume Rendering

Spot Lights

Spotlights are worse case than omnidirectional lights

Angular falloff complicates integral

Interval needs to be clamped to cone intersection

Can be simplified

- No Falloff
- Fixed Falloff
- Variable Falloff



GDC

gameworks.nvidia.com

43 NVIDIA

Spotlights are just omni-lights with an angle restriction, but that makes the integral much harder
Broken into subclasses and solved separately

Volume Rendering

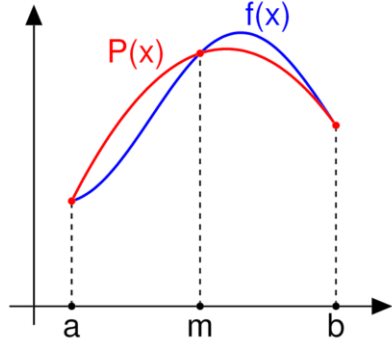
Numerical Integration



No Falloff: Treat as point light

Fixed Falloff: Use modified point LUT

For other cases, integrate in the PS

- 1) Do cone intersection as before
- 2) Numerically integrate (Newton-Cotes)
- 3) Use the result as the integral




gameworks.nvidia.com
44 

No Falloff is very fast, just use the same look-up technique as omni lights and clamp interval to cone intersection

Fixed falloff uses a special case of the integral where the spotlight power $N=1$ that lets us reduce the integral to the sum of three 2D lookups

Spotlights with arbitrary falloff power can be solved using numerical integration over the interval

We use Newton-Coates, which approximates the function as the sum of polynomial intervals. More expensive, but still fairly fast.

Apply Lighting

MSAA/Temporal Resolve

Use MSAA when down-sampling

- Preserves shading rate savings
- Improves edge quality

Resolve MSAA accumulation buffer before compositing

If temporal sampling, feed resolved accumulation through TAA system



gameworks.nvidia.com

45 

All the results are summed into an accumulation texture.

We might use MSAA or temporal AA to improve the rendering quality, so we have to resolve the accumulation buffer before final compositing.

Apply Lighting

Composite Results

Bilateral upsampling: helpful at $\frac{1}{4}$ -resolution, situational at $\frac{1}{2}$ -resolution

Additively blend resolved/filtered output with scene

Use dual-source blending to attenuate based on medium+depth



gameworks.nvidia.com

46 

For the actual compositing, we simply upsample the resolved accumulated inscatter and additively blend it with the scene output.

We have bilateral filtering in place for the upsampling, but it anecdotally it seems like the slight blur of bilinear is preferable to the hard edge of a bilateral for half-resolution. At quarter-resolution the blur gets wider, so bilateral becomes preferable.

The composite pass also supports attenuating the transmitted light from the scene, but this is optional as the application may already have a fogging component baked into the framebuffer.

Integration into *Fallout 4*

Fallout 4 Integration

Feb: Development/Integration Begins

March: Artists authoring environments

April-June: Console Ports

July-August: Optimization (PC+Console)

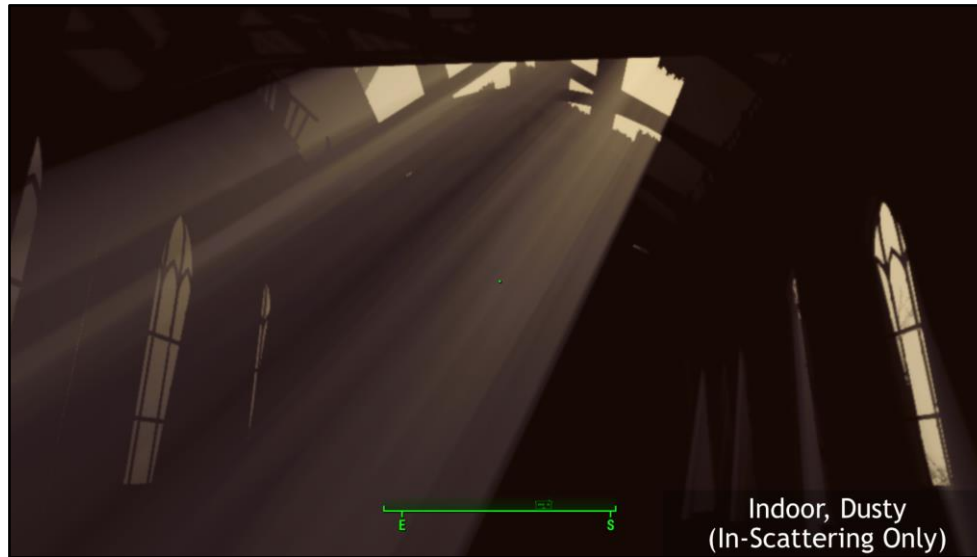
Dedicated NVIDIA QA resources





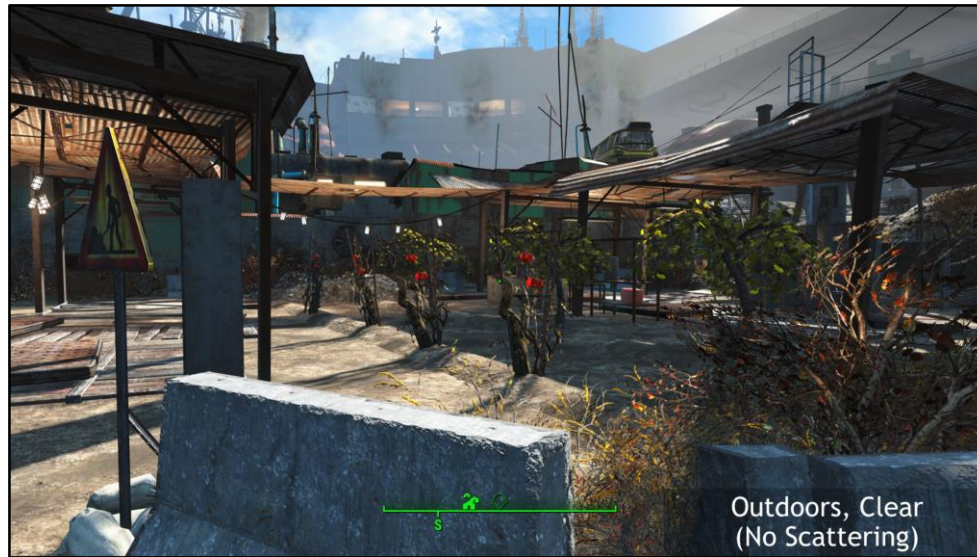
Indoor scene with dusty atmosphere, light shafts coming through a hole in the roof

This is the kind of effect that would be hand-made by artists, but is generated automatically



Inscattering-only view

Note we capture the fine details around the opening in a way that would be hard for ray marchers, and the edges of the volume are crisp and clean.

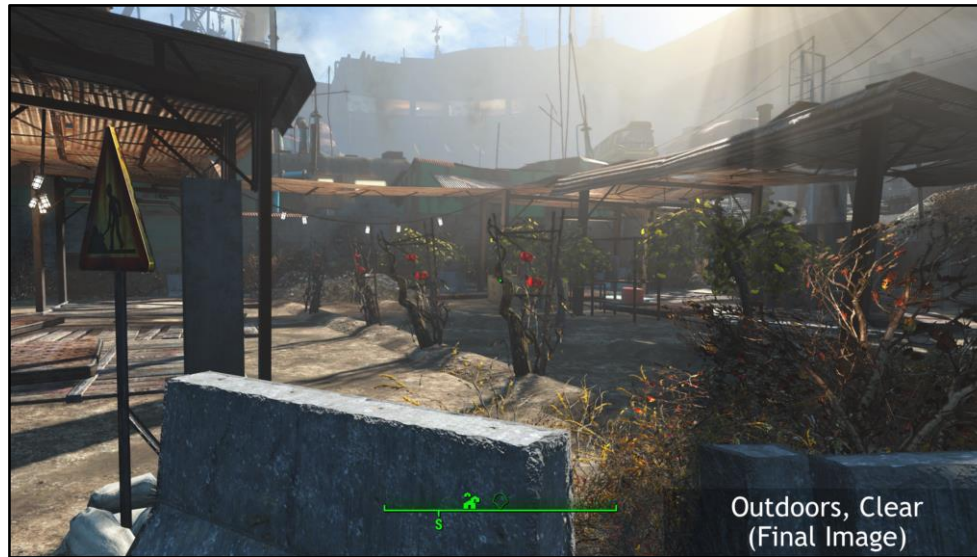


Outdoors, clear sky, mid-day. No inscattering.



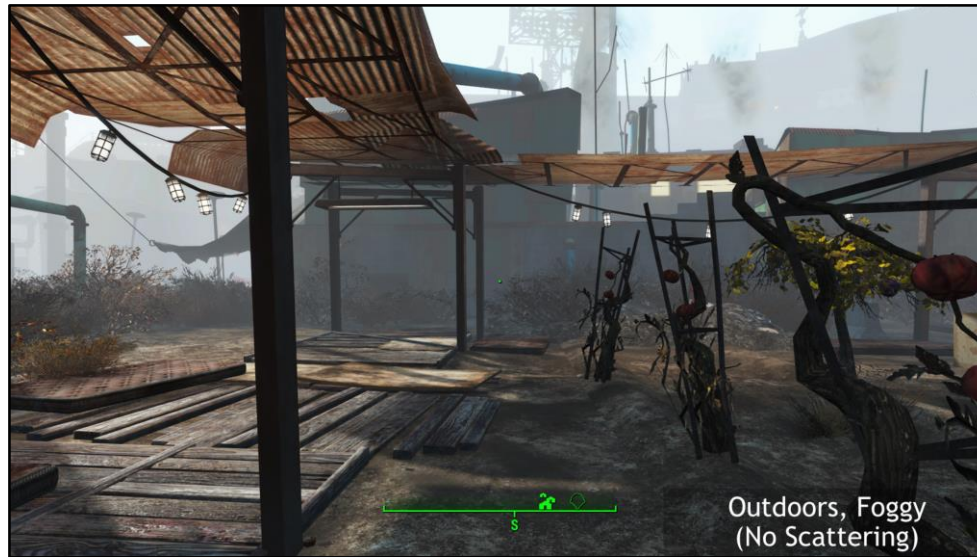
Inscattering view. Off-screen occluders cast shadows into the view.

Note the chromatic effects from rayleigh scattering tinting the aerial perspective

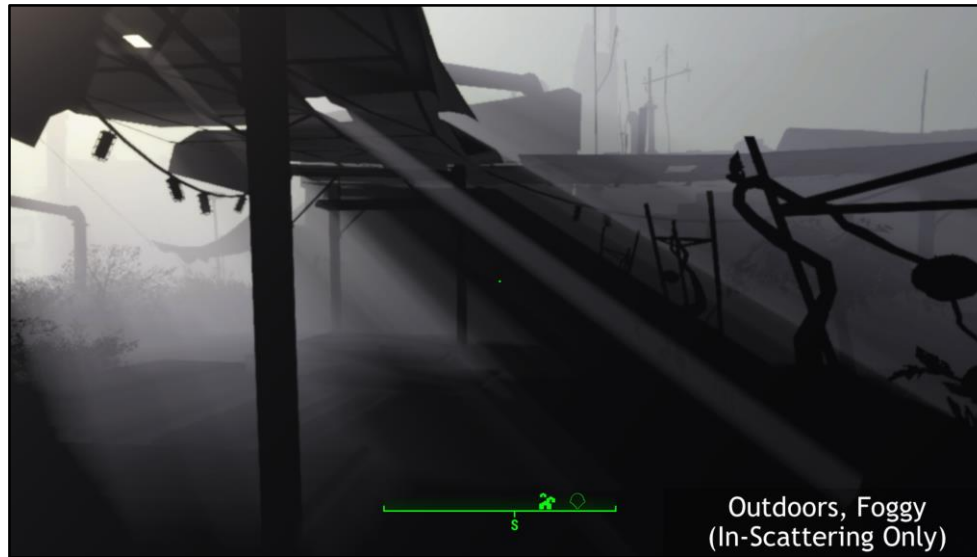


Combined result.

See how the structure shadows reduce the atmospheric perspective, giving them a sense of “mass”

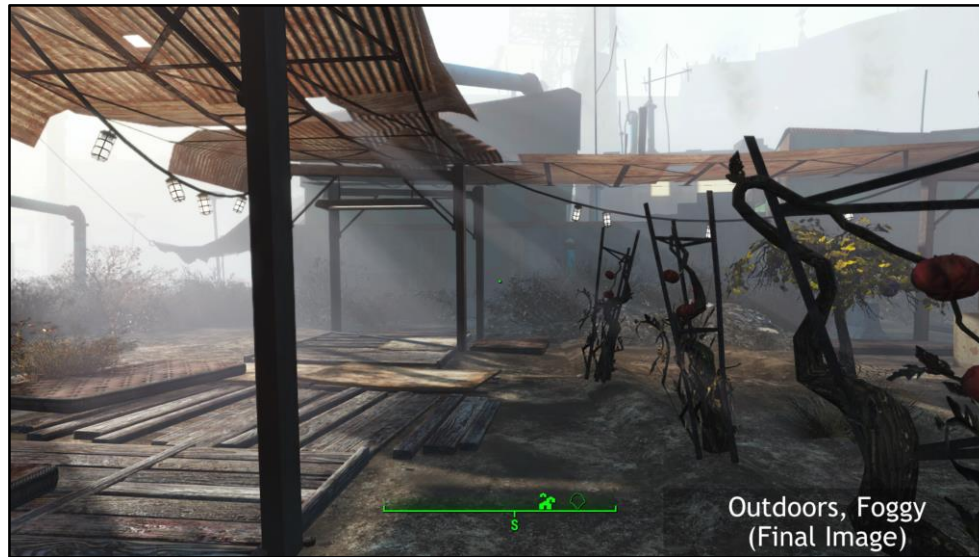


Same location, different view. Morning, misty environment. No Inscattering.



Inscatter-only.

Small features are captured well by the mesh and there are crisp, sharp edges at discontinuities.



Combined results.



Outdoors, dusty street-scape. No Inscatter.



Inscatter only.

A wide phase function from the dust gives a large 'bleed over' around the light source. Realistic bloom!

Detailed enough to capture the thin elements of the fire escape on the far side.





Combined results.

We used this scene with various settings to generate perf numbers.

Gpu Performance

Concord, High Quality @ 1080p

	BEGIN	RENDER VOLUME	FILTER & COMPOSITE	TOTAL
GTX 980 Ti	0.037 ms	0.908 ms	0.264 ms	1.209 ms
GTX 970	0.049 ms	1.364 ms	0.389 ms	1.802 ms
Radeon Fury X	0.038 ms	2.581 ms	0.305 ms	2.924 ms


gameworks.nvidia.com


High Quality: Half-Resolution



Begin pass is cheap (generate media LUT and downsample depth).

Render Volume generally scales relative to the total pixel coverage of the volume rather than number of volumes. Multiple small lights on the screen shouldn't cost much.

Gpu Performance

Concord, Medium Quality @ 1080p

	BEGIN	RENDER VOLUME	FILTER & COMPOSITE	TOTAL
GTX 980 Ti	0.035 ms	0.415 ms	0.394 ms	0.844 ms
GTX 970	0.049 ms	0.610 ms	0.565 ms	1.224 ms
Radeon Fury X	0.041 ms	1.074 ms	0.636 ms	1.751 ms


gameworks.nvidia.com
61 

Medium Quality: Quarter-resolution, 4x MSAA, Bilateral Upsampling

The reduced shading rate dramatically helps the volume render pass, but MSAA allows us to keep surprisingly good quality around light edges. Filtering&Composite gets more expensive due to the additional processing to maintain quality.

INTEGRATION TIPS

- Maximize Dynamic Range
- Make sure shadow maps are consistent
- Temporal filter low-res effects separately
- Be aware of worst-case scenes

Implementation Issues

Reduced Dynamic Range Limits Contrast

Problem: Hard to get intense effects without washing out scene

Intensity proportional to light source power

Real-world effects involve 10,000:1 contrast between source and scene!

Baked-in ambient normalizes intensity between bright and dark areas

Simply increasing medium density causes wash-out

Solution: Need HDR with real adaptation between dark and bright



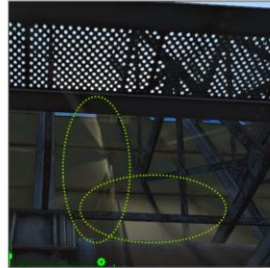
gameworks.nvidia.com

63 

Implementation Issues

Shadow Map/Scene Inconsistencies

Problem: Shadow Map inconsistencies become much more noticeable



GDC



gameworks.nvidia.com



64 NVIDIA

Implementation Issues

Shadow Map/Scene Inconsistencies

Problem: Shadow Map inconsistencies become much more noticeable

“Bug” in art, but not noticeable because surface and shadows not usually visible

May only render front-faces to shadow map, but there may not have consistent geometry/alpha masks on both sides

May not render “distant” occluders to the shadow map

May use a separate, high-detail map for certain occluders

Solution: be consistent with your shadow map contents!



Implementation Issues

Temporal Jittering Causes Flicker

Problem: Temporal AA jitter causes flickering

Temporal AA jitters to increase effective resolution, then filters to smooth

Library runs at $\frac{1}{2}$ - $\frac{1}{4}$ resolution to improve performance

A 1 px flicker at full-res could become 4x4 px in the down-sampled buffer!

Full-res temporal filter not designed to smooth artifacts that large

Solution: Added separate TAA resolve to down-sampled buffers

Implementation Issues

Perf Drop with High Frequency Occluders

Problem: Poor perf in specific views

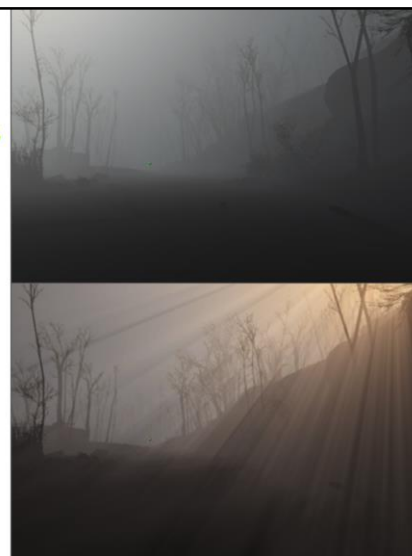
Cost proportional to total pixel coverage

Dense occluders are no problem but create overhead at low angles

Ex: Sunset through the woods

Solution: Adjust tessellation factor based on view angle

Solution: Pre-filter shadow map

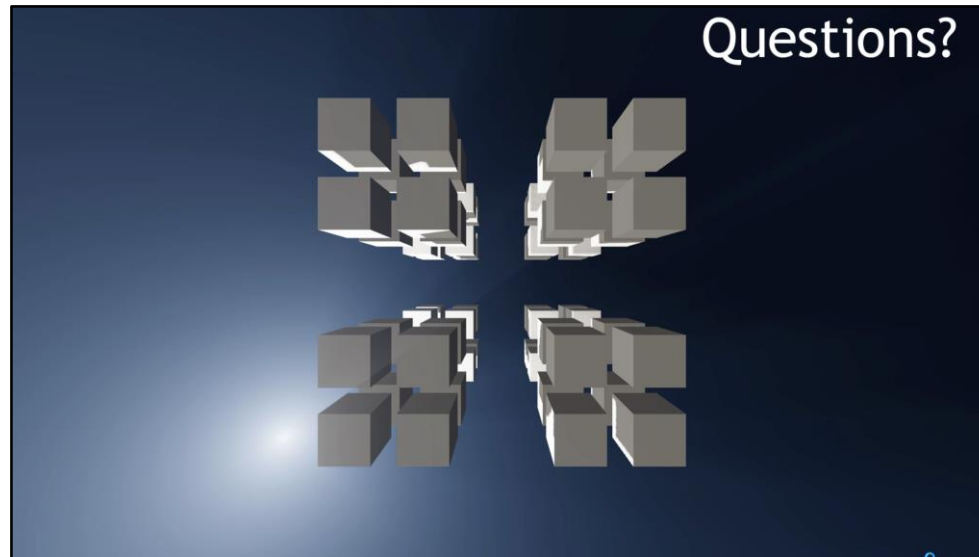


SUMMARY

ameworks Volumetric Lighting is...

- Fast enough for entire spec range
- Flexible enough for almost any engine
- Compatible with physically-based engines
- Currently available in DirectX 11 (with ports being added according to demand)

<http://developer.nvidia.com>



Look-up is small, but flexible

Function is smooth, so it approximates well

RELATED READING

Hoffman, N., and A. J. Preetham. 2002. Rendering Outdoor Scattering in Real Time. <http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/ATI-LightScattering.pdf>

Sun, B., et. al. 2005. A Practical Analytic Single Scattering Model for Real Time Rendering. <http://www.cs.columbia.edu/~bosun/sig05.htm>

Bouthors, A., Neyret, F., Lefebvre, S. Real-time realistic illumination and shading of stratiform clouds. <http://www-evasion.imag.fr/Publications/2006/BNL06/bnl06-elec.pdf>

Englehardt, T., Dachsbacher, C. 2010. Epipolar Sampling for Shadows and Crepuscular Rays in Participating Media with Single Scattering. <http://gpucomputing.net/sites/default/files/papers/5398/espmss10.pdf>

Wronski, B. 2014. Volumetric Fog: Unified, Compute Shader Based Solution to Atmospheric Scattering. <http://bartwronski.com/publications/>

Hillaire, S. 2015. Physically-based & Unified Volumetric Rendering in Frostbite. <http://www.frostbite.com/2015/08/physically-based-unified-volumetric-rendering-in-frostbite/>

