Game Developers Conference

Real-Time Animated Translucency

Greg James, NVIDIA
Simon Green, NVIDIA

Audience & Goals

- Programmers
- Designers
- Technical Artists

- Review current methods
  - Being used in upcoming titles
- Future directions

Introduction

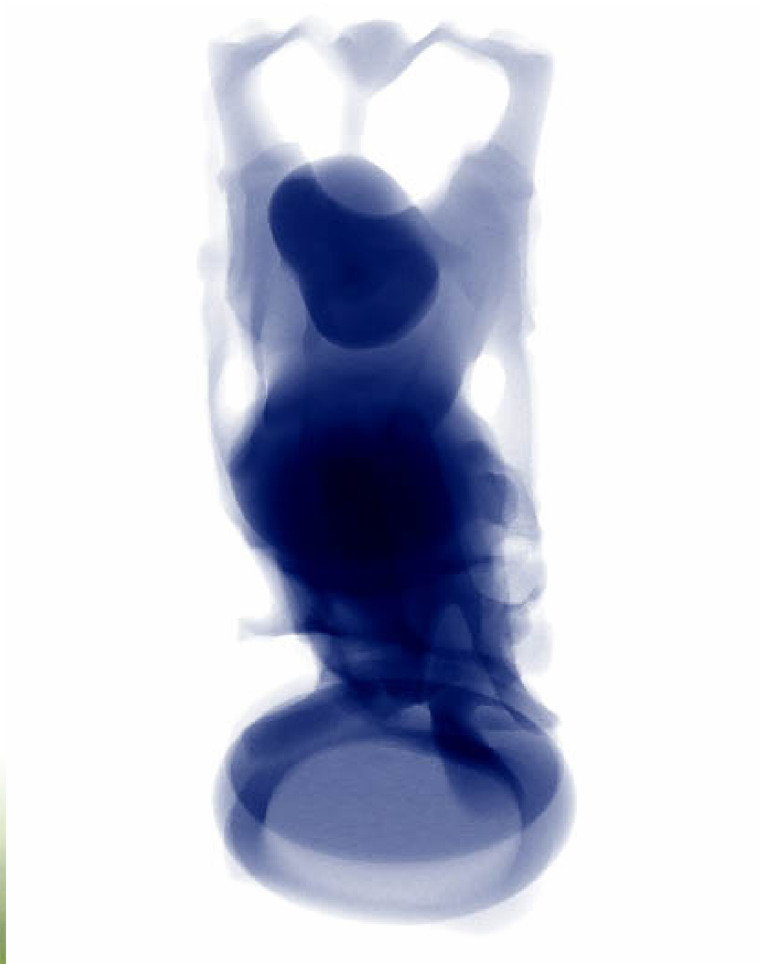
- Basics of translucency & scattering
- Focus on visual appearance, not physics
- Techniques:
  - Atmospheric light scattering (Hoffman & Preetham)
  - Pre-computed radiance transfer (P.P. Sloan)
  - Polygon hulls as thick volumes
  - Lighting model tricks
  - Depth-map based scattering
  - Texture-space diffusion

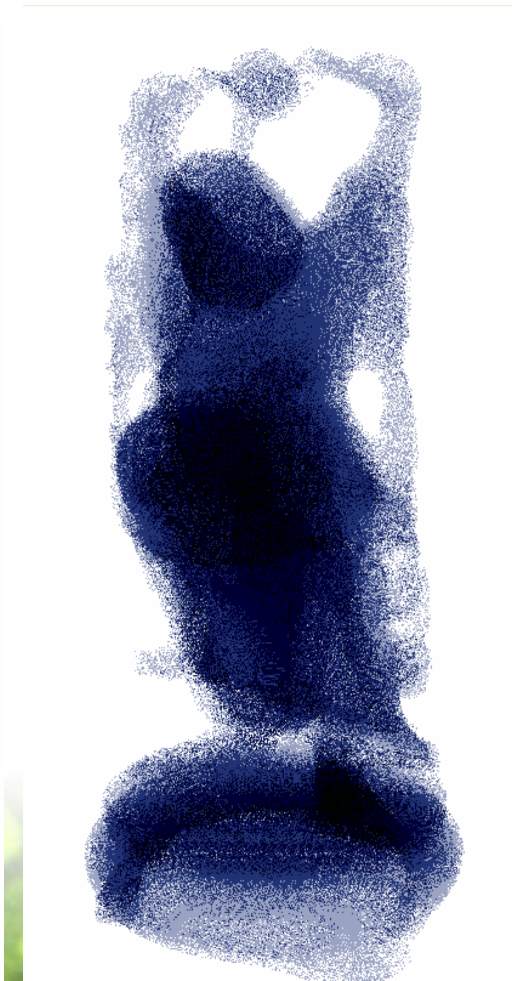
Teaser Images

- Volume fogs
- Trivial to animate the fog and the scene

Teaser Images

- X-ray effect





Teaser Images

Light transmission

No light transmission

Game Developers Conference

EVOLVE GDC>04

Teaser Image: Skin Diffusion

No Diffusion

Subsurface Diffusion

NVIDIA.

Translucency and Scattering

- All materials are translucent
  - Depends on light wavelength
- Light penetrates all surfaces to some degree
  - Different wavelengths have
    - Different penetration depths
    - Different falloff vs. depth
- If not absorbed or reflected, the light might scatter and exit somewhere else

Why Use Translucency?

- Subtle effect, but powerfull visual cue
- Translucent objects are more pleasing, interesting, and realistic
- Particularly significant for skin shading
  - Simon will talk about this
- Routinely used in movies - Harry Potter, The Hulk, Matrix Revolutions

Game Developers Conference

Translucency Basics

- Optical Properties
  - Absorption (probability vs. distance)
  - Scattering (probability vs. distance & angle)
  - Impedance changes (reflection and refraction)
    - Optical impedance determines the index of refraction
- Everything absorbs and scatters
  - fluids, solids, gasses, even pure clean air
- Opacity, transparency, and translucency
  - Vary in the probability of absorption & scattering

Opactiy

- High probability of absorption & scattering
- Light takes short paths
- Light comes from surface, not interior

Transparency

- Low probability of absorption and scattering

Images courtesty of Leigh Van Der Byl

Translucency

- Low probability of absorption
- High probability of scattering

Leigh Van Der Byl

Real-Time Attitude

- Get the look. Forget the math
  - See Hoffman & Preetham for good scattering math
- Various techniques
  - Depth-map rendering for thickness & scattering
  - Texture-space diffusion
- Requirements
  - Artist friendly, content friendly
  - Fast as blazes
  - Fallbacks
  - Animate-able lighting and self-shadowing

Depth Maps

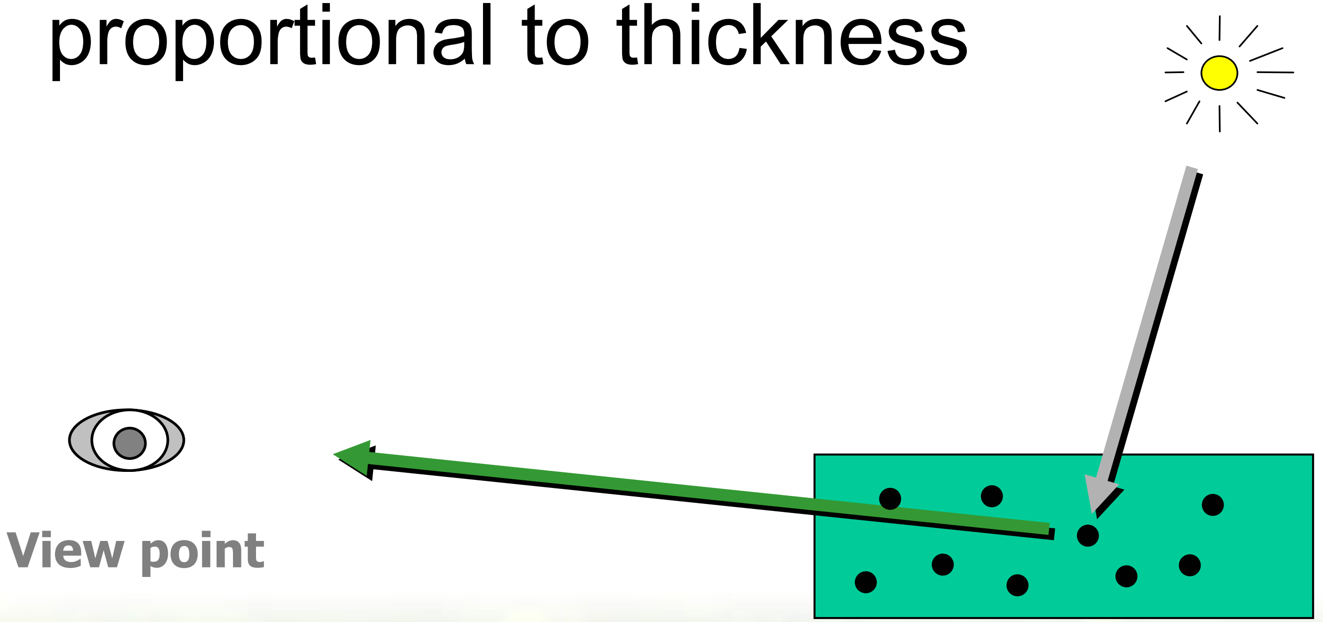
- Fog is an ordinary polygon model
- Render-to-texture passes used to calculate distance through fog object
- ps.1.3
- ps.2.0 is faster
- ps.3.0 is faster++

Volume Fog Technique

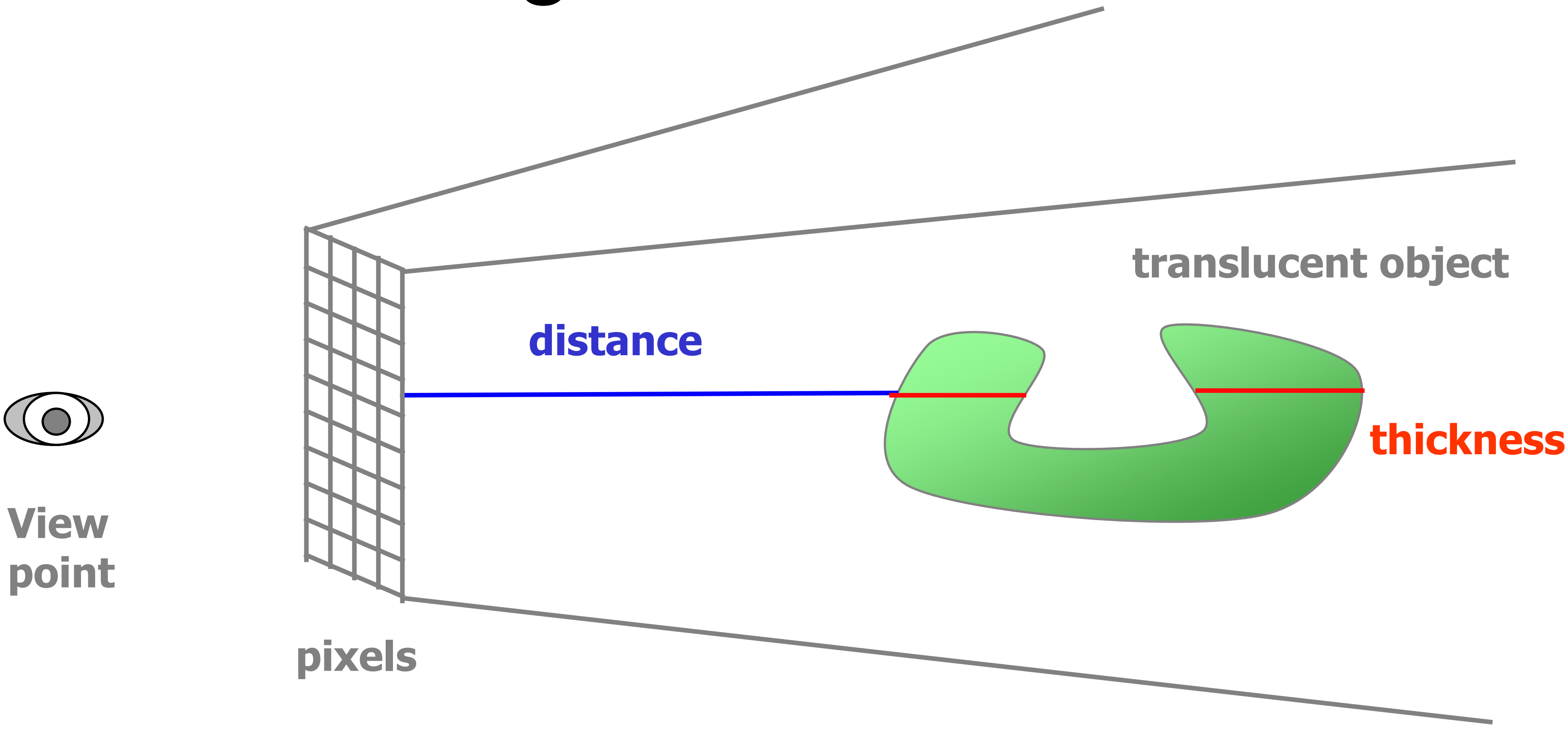
- Inspired by Microsoft's "Volume Fog" DXSDK demo (Dan Baker)
- Inspired by [Mech01]
- Compute thickness through ordinary polygon objects from camera's P.O.V.
  - Render the depths of an object's front and back faces
- Derive color from thickness
- Great method for single scattering

Single Scattering

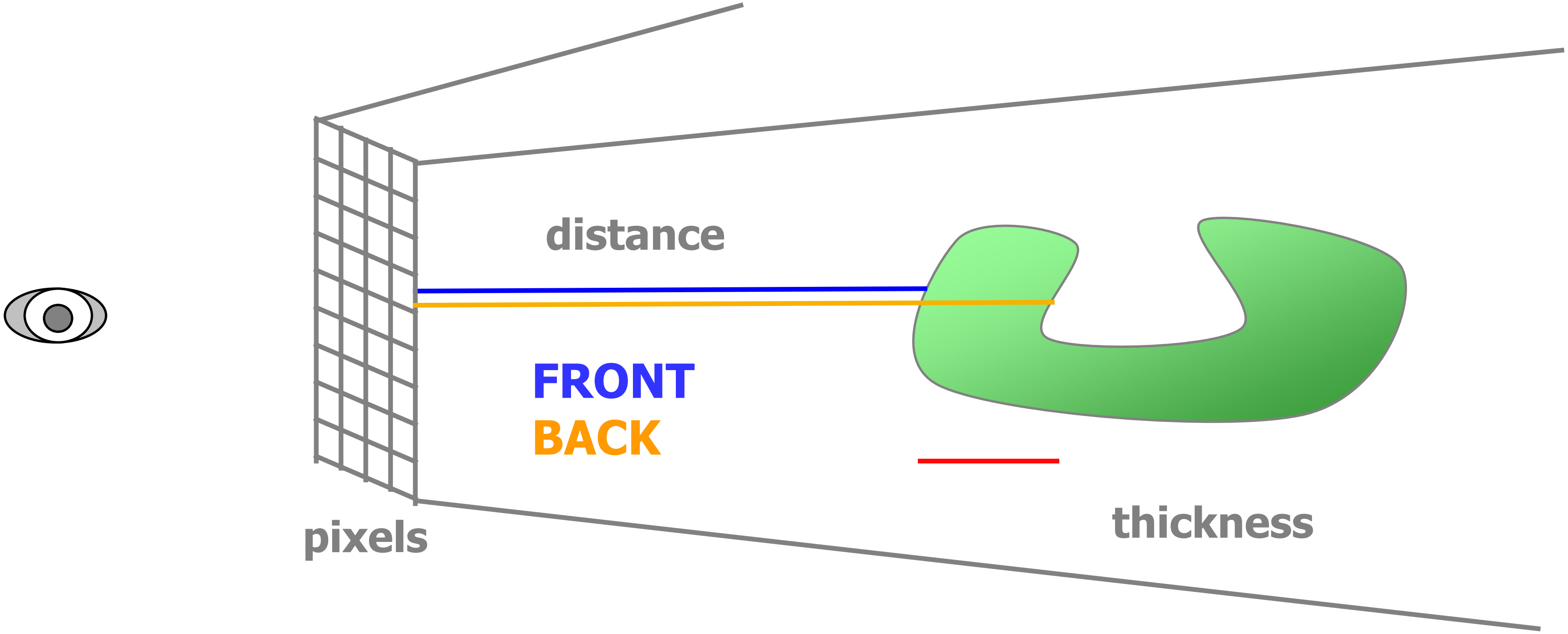
- Light bounces once from source to eye
- Light contribution from scattering is proportional to thickness



Rendering Thickness Per-Pixel

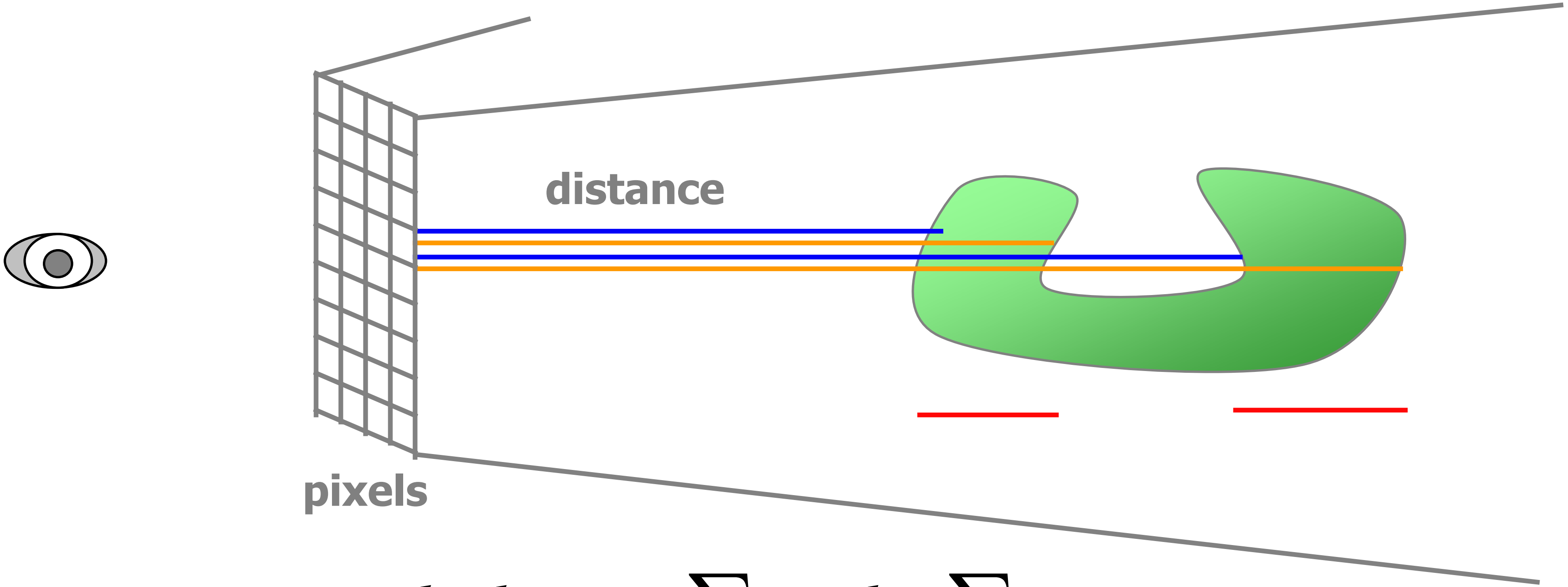


Thickness From Distances



THICKNESS = BACK - FRONT

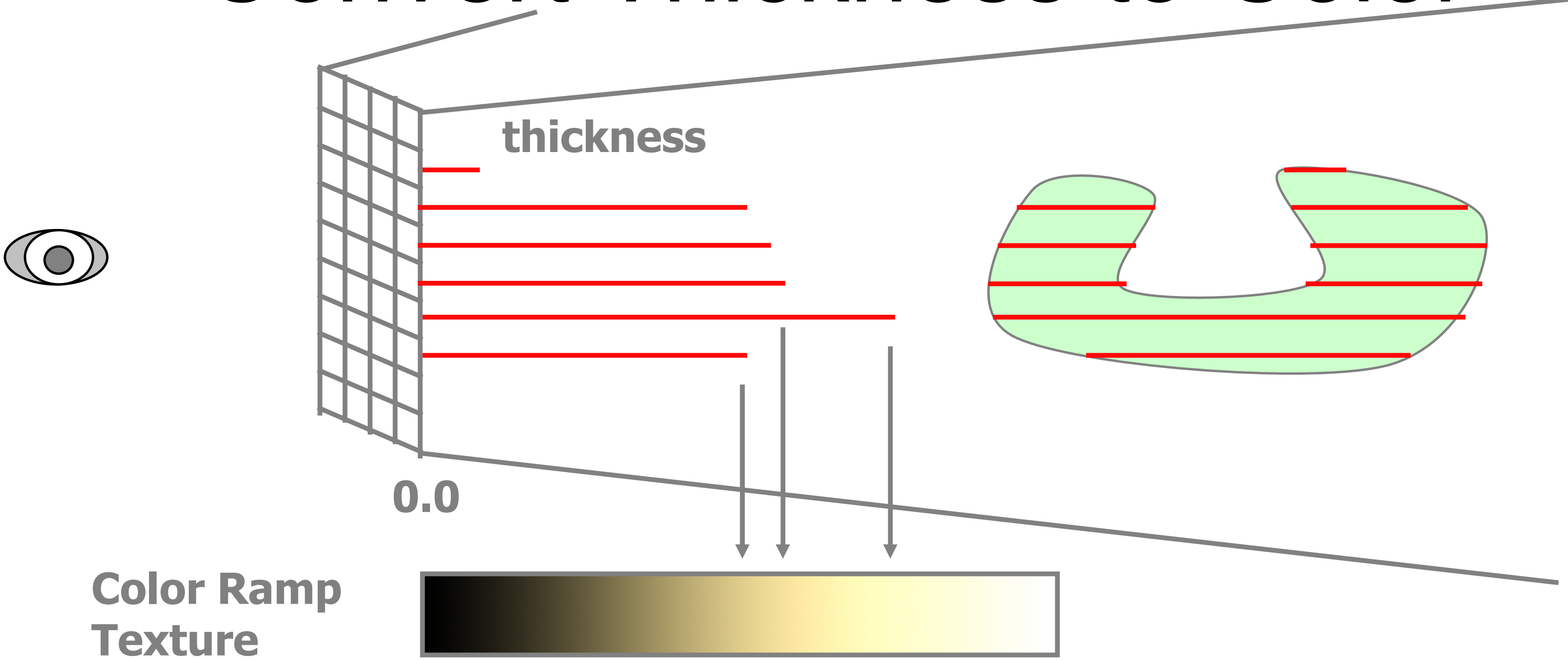
Rendering Thickness Per-Pixel



$$Thickness = \sum Back - \sum Front$$

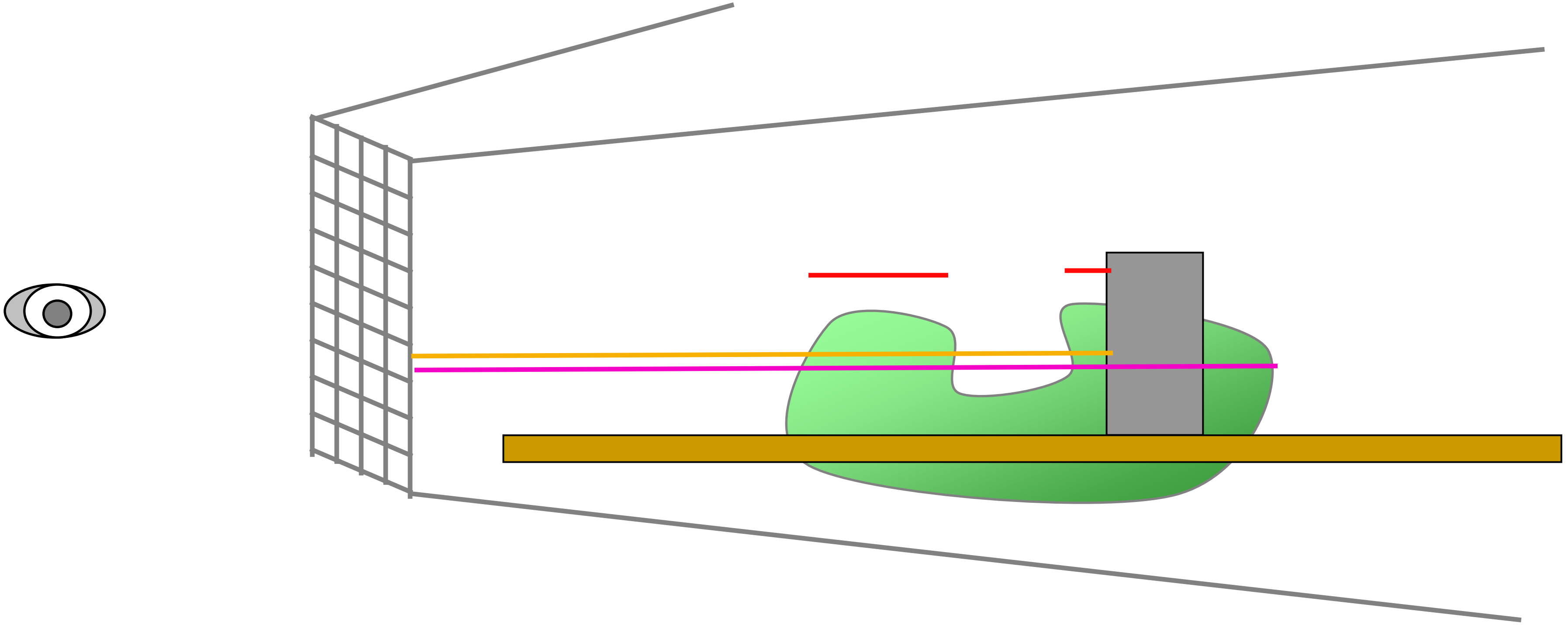
- Thickness for any uniform density object is easy
- No Z-Buffer. Use additive blending

Convert Thickness to Color



- Thickness * scale ➔ TexCoord.x
- Color ramp texture: Artistic or math
- Easy to control the look

What About Intersection?

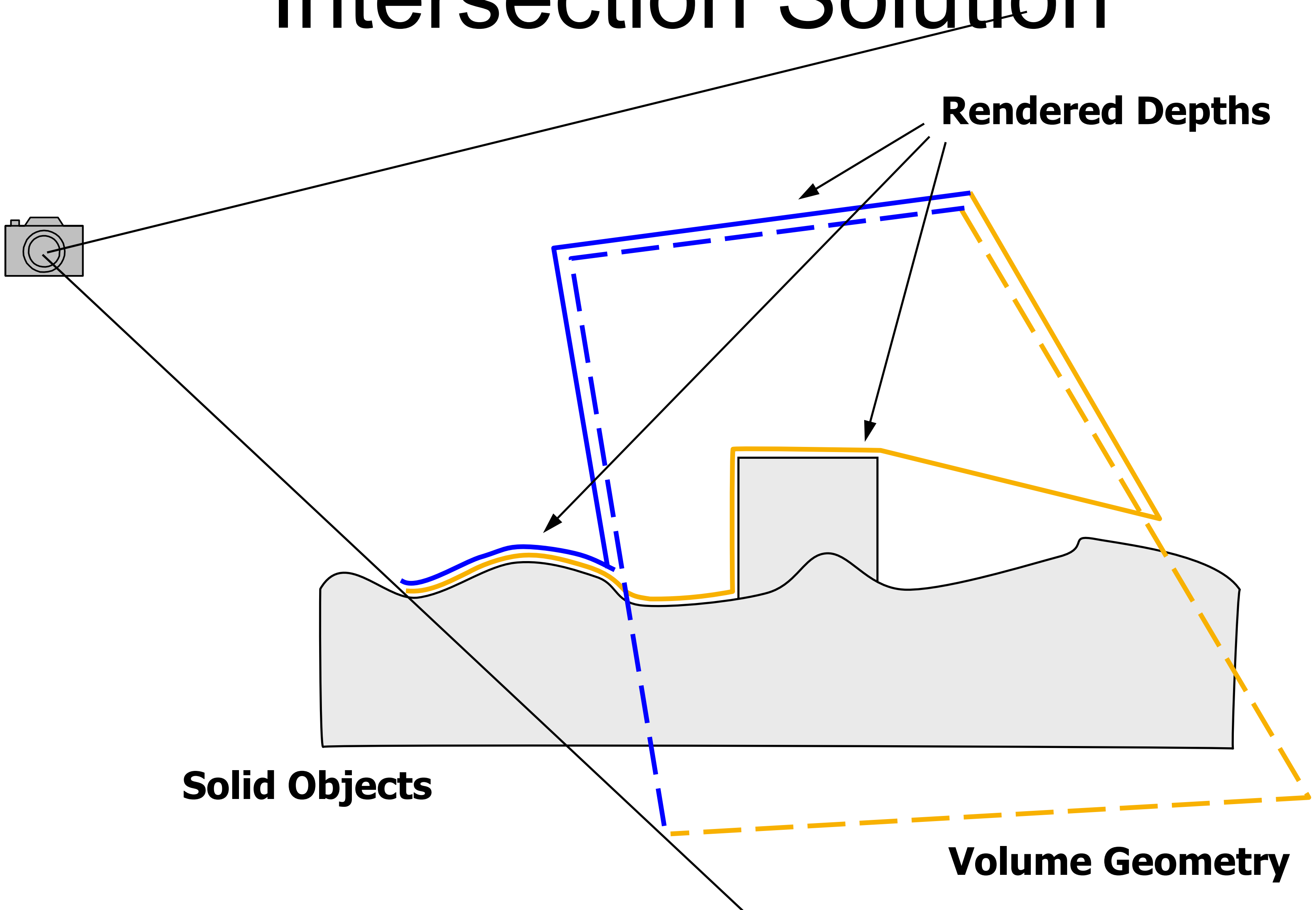


- Need depth to solid object
- Not depth to volume object faces

Intersection Solution

- Need depth of nearest solid object
  - Render it to a texture
  - Read the texture in a pixel shader
- As you render each of the volume object’s faces
  - Pixel shader outputs lesser of
    - Depth of volume object triangle being drawn
    - Solid object depth (from texture) at pixel being drawn
  - Disable depth testing
  - Additive blend the output depth into the framebuffer

Intersection Solution



Intersection Method Advantages

Advantages

- Does not require stencil
- Does not require multi-pass

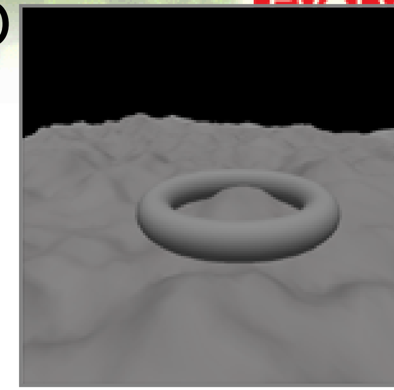
Disadvantages

- Must render depth of
  - Anything intersecting the volumes
  - Anything that can occlude the volumes
    - Can be avoided depending on the scene

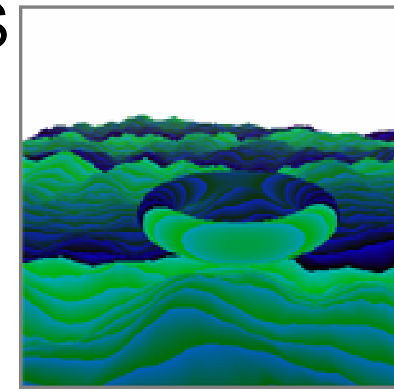
Steps: Pixel Shader 2.0

1. Render solid objects to backbuffer
   - Ordinary rendering
2. Render depth of solid objects that might intersect the fog volumes
   - To ARGB8 texture, “S”
   - RGB-encoded depth. High precision!
3. Render fog volume backfaces
   - To ARGB8 texture, “B”
   - Additive blend to sum depths
   - Sample texture “S” for intersection

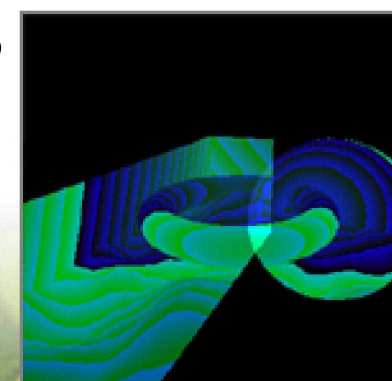
O



S

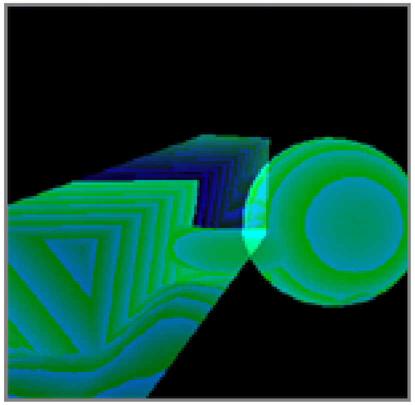


B

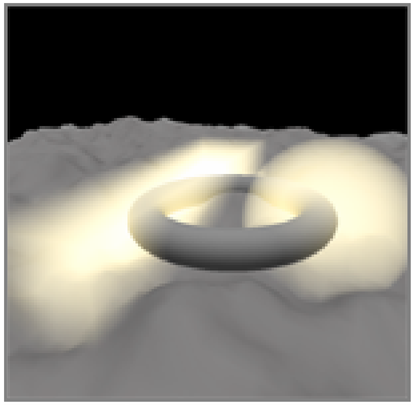


Steps: PS.2.0 contd.

F



4. Render fog volume front faces
   - To ARGB8 texture, “F”
   - Additive blend to sum depths
   - Sample texture “S” for intersections
5. Render quad over backbuffer
   - Samples “B” and “F”
   - Computes thickness at each pixel
   - Converts thickness to color using fog color ramp texture
   - Blends color to the scene
   - 5 instruction ps.2.0 shader



Final

PS.3.0 HW Improvements

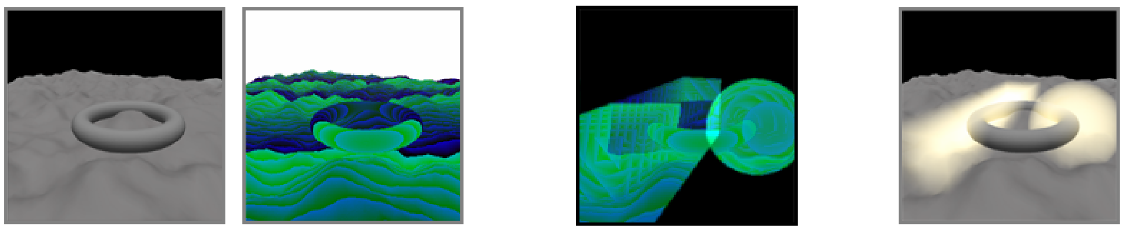
- Front / back facing register
- Multiple Render Targets (MRT)
- Floating-point framebuffer blending

- Fewer passes
- Fewer render-target textures

PS.3.0 vs. PS.2.0

MRT

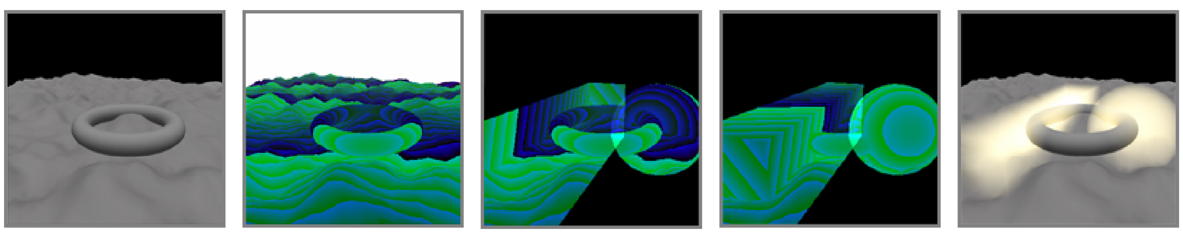
ps.3.0 HW



a. RT-Tex RT-Tex 'O' c. 'F' and 'B' F/B register d. Backbuffer

3 Passes

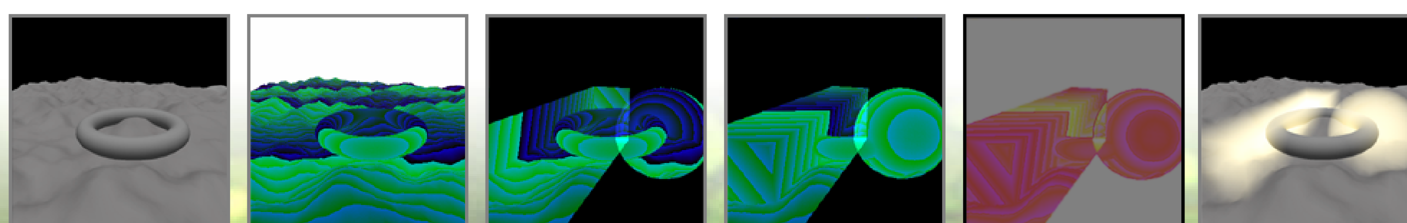
ps.2.0 HW



a. Backbuffer b. Texture 'O' c. Texture 'B' d. Texture 'F' e.

5 Passes

ps.1.3 HW



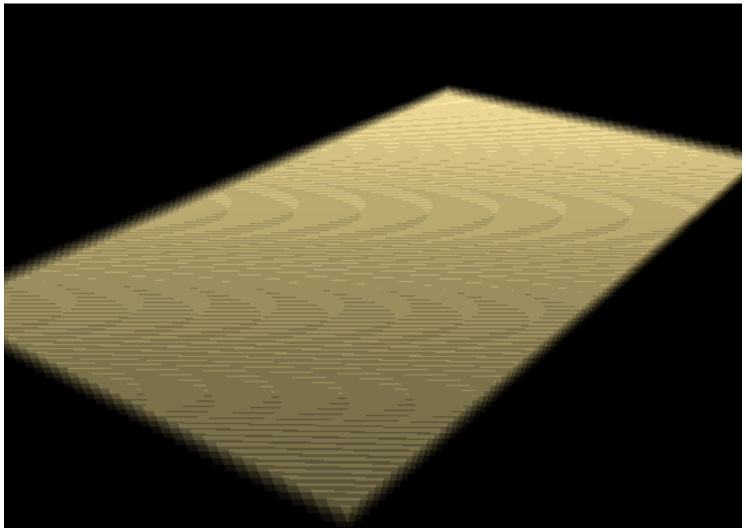
6

Volume Fog Technique

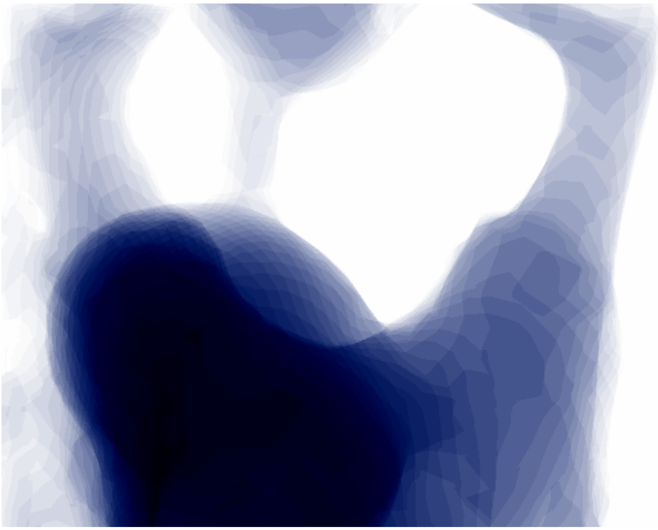
- NV demo improvements
  - Higher precision: 12, 15, 18, 21-bit depth
  - Precision vs. depth complexity tradeoff
  - High precision decode & depth compare
  - Dithering eliminates depth aliasing
  - No banding, even with deep view frustum
  - Simple, complete intersection handling for any shapes

Importance of Dithering

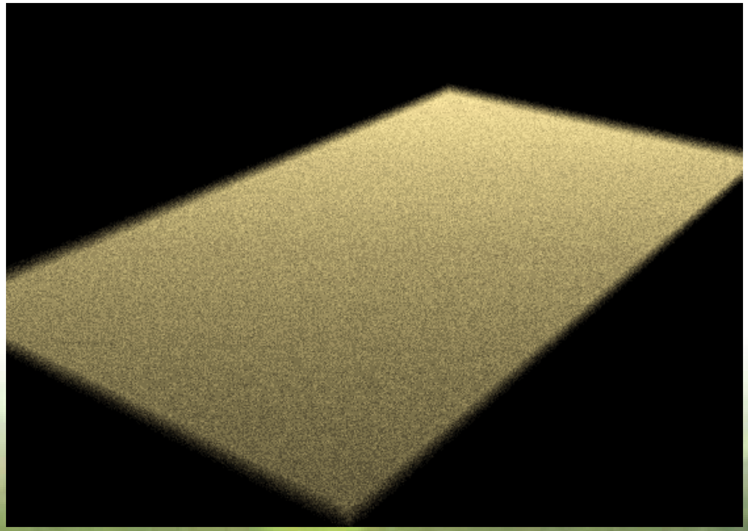
a.



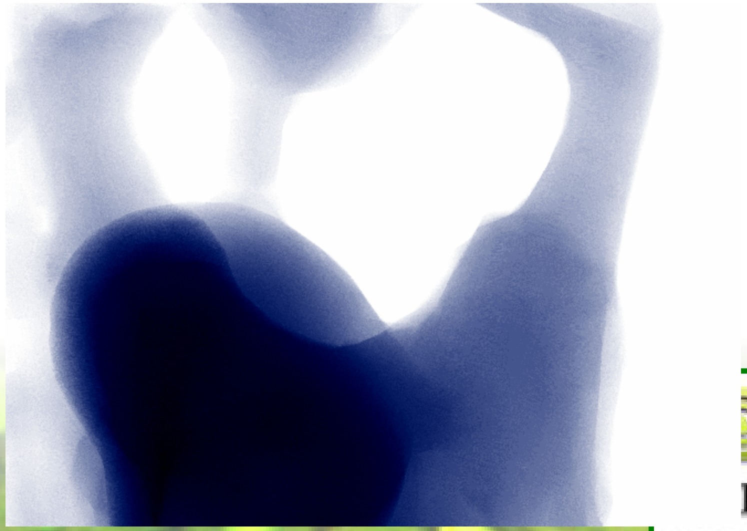
b.



c.



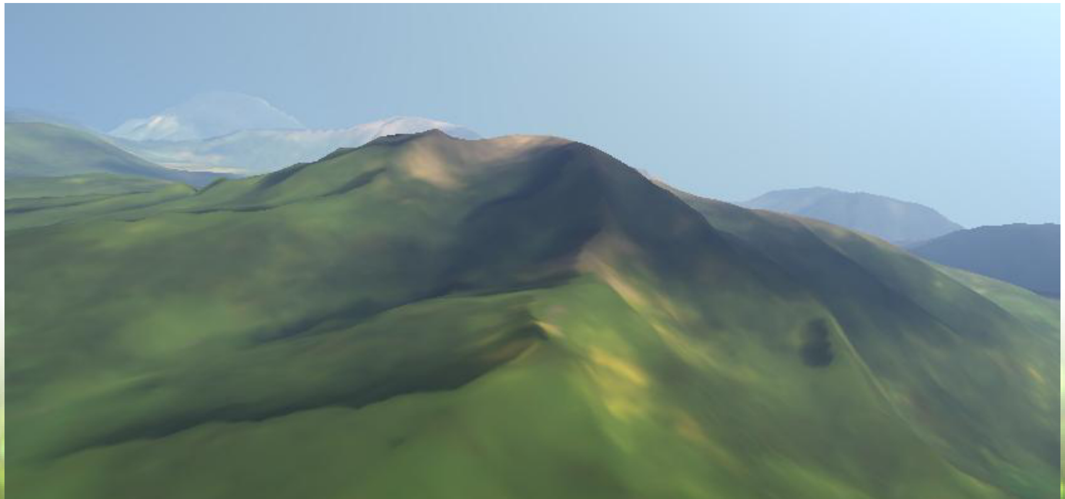
d.



Fancier Scattering

- We used a texture to convert thickness to color

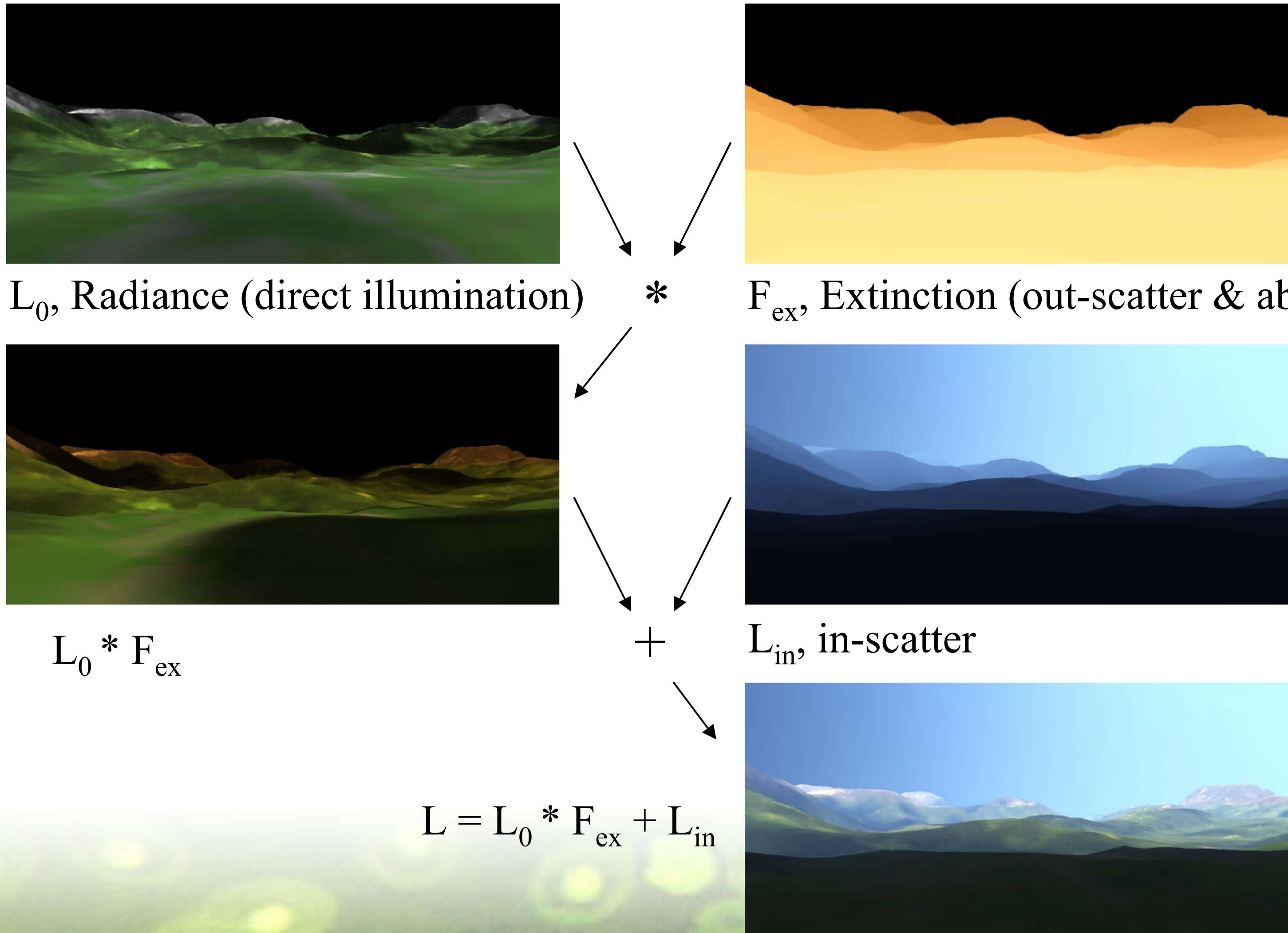
- Could use math to describe light scattering
- Hoffman & Preetham atmospheric scattering



Real-Time Translucent Atmosphere

- Hoffman & Preetham
- Rayleigh & Mie scattering in vertex shader

Atmospheric Scattering Terms



Images courtesy of Hoffman & Preetham

Game Developers Conference

EVOLVE GDC>04

GREG'S REFERENCE SLIDES

- **

NVIDIA.

Non- and Near-Real-Time Methods

- Faster monte carlo simulation
  - H.W. Jensen, J. Buhler, Siggraph'02, p. 576
- BSSRDF
  - C. Hery (ILM), Jensen, et. al.
- Pre-computed Radiance Transfer (PRT)
  - P.P. Sloan, MSFT
  - New work at GDC 2004 MSFT Developer Day (Tues.) on animating the parameters
  - Animation is tough in SH-basis

Issues with SH-Basis PRT

- Illumination from sources at infinity
  - Environment map
  - Must be pre-processed to encode in SH basis
  - How to get occlusion from local dynamic objects?
    - trees, walls, other occluders
- Self-shadowing and large motions
  - Animation transforms have high-frequency effects on lighting and self-shadowing
  - Accounting for high frequencies with N animation parameters leads to a data explosion

PRT for Sub-Surface Only

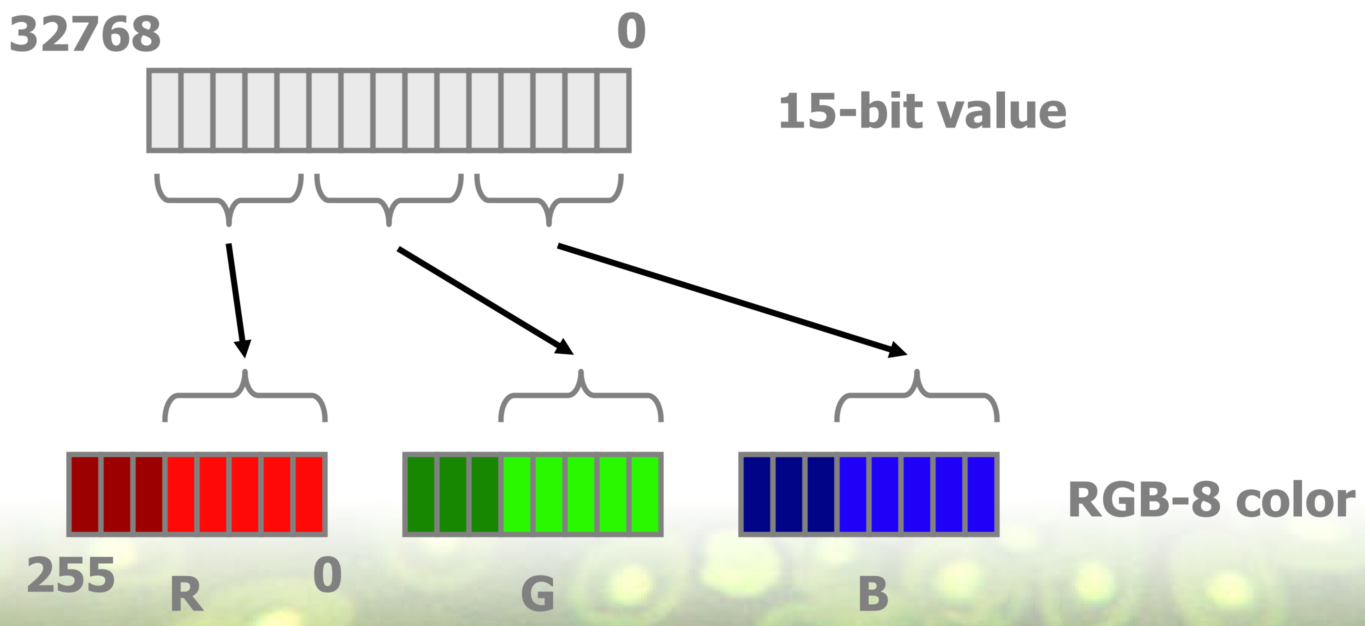
- Separate PRT for self-shadowing from PRT for sub-surface
- Incident radiance has high-frequency changes under animation
  – SH basis is undesireable – slow encode
- Sub-surface light has low frequencies
- Simple ‘gather’ of incident L (reduce resolution) and polynomial mapping to sub-surface contribution ?

Scattering Characteristics

- $B_{sc}(\theta)$ is the probability of scattering
  - Depends on angle, θ
- Rayleigh scattering
  - Why the sky is blue
  - Particle size < wavelength of light
  - Electron orbits make it wavelength dependent
- Mie scattering
  - Why smoke is smokey
  - Particles > wavelength of light
  - Depends on particle absorption & reflectance
  - Complex probability of scattering, $B_{sc}(\theta)$

RGB-Encoding

- Encode and Sum high-precision numbers stored as A8R8G8B8 colors
- Use if no float framebuffer blending



Radomir Mech Helicopter

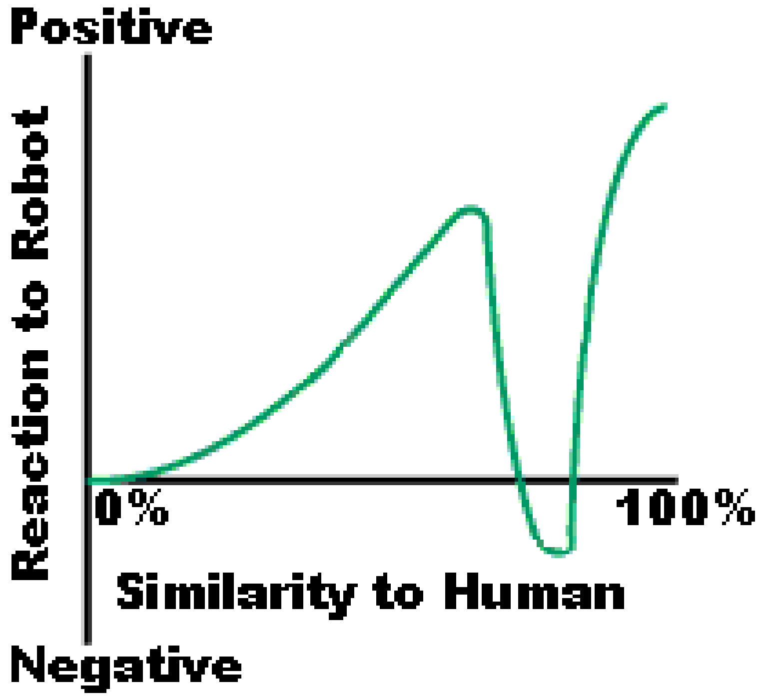
BEGIN SIMON’S SECTION

- **

Other Scattering Techniques

- Why scatter?
- 3 Techniques:
  - Lighting model tricks
  - Depth-map based scattering
  - Texture-space diffusion

The Uncanny Valley



Coined by Japanese roboticist Doctor Masahiro Mori

What Does Scattering Look Like?

- Softens overall effect of lighting
  - small surface details are less visible
  - light bleeds from light areas into shadows
- Attenuation
  - the further light travels through the material, the more of it gets absorbed and diffused
- Color shift
  - the color of the exiting light is affected by sub-surface material

BRDF

From: Jensen et al "A Practical Model for Subsurface Light Transport"

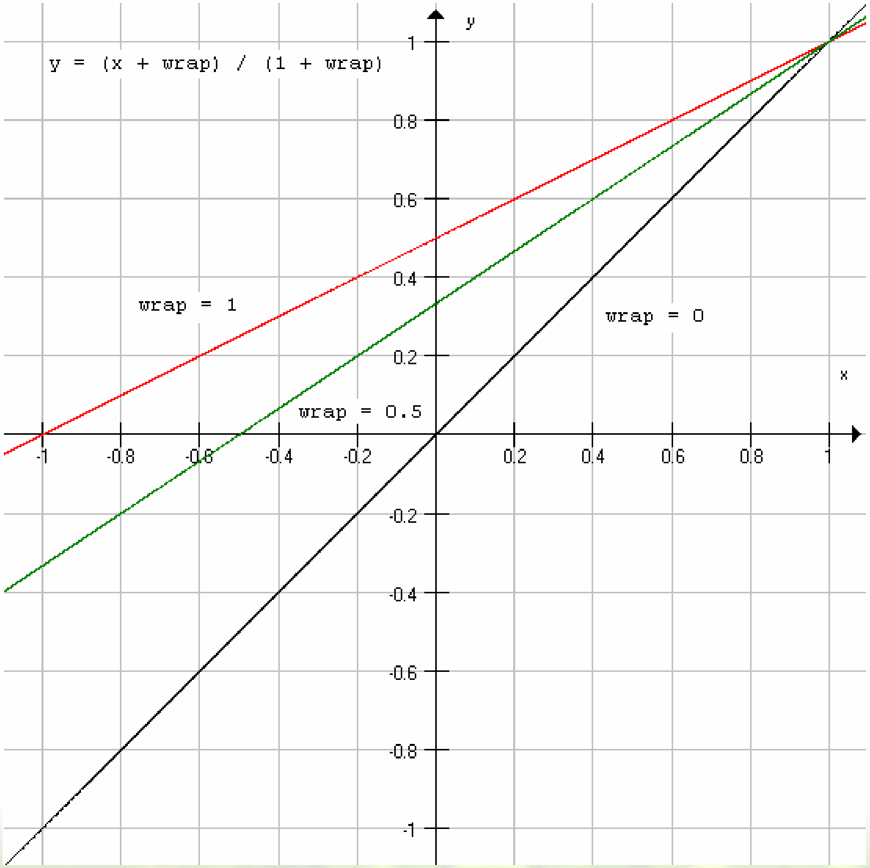
BSSRDF

From: Jensen et al "A Practical Model for Subsurface Light Transport"

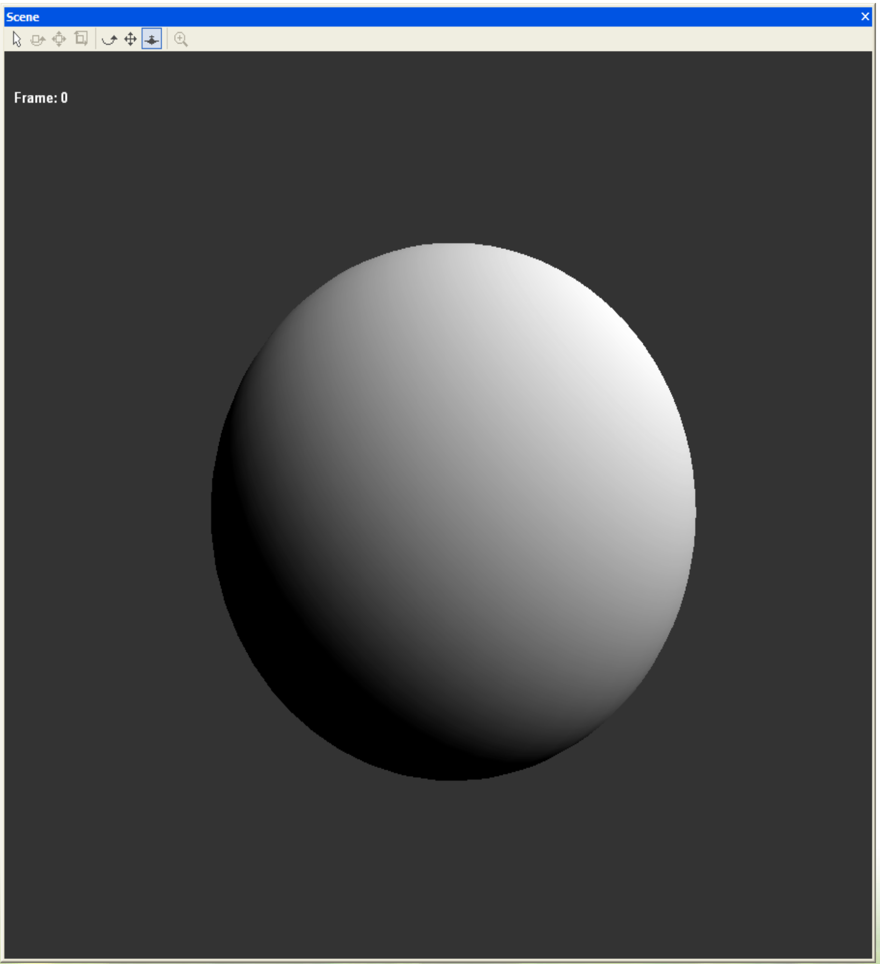
Lighting Function Tricks

- Very few diffuse surfaces actually obey Lambert's law – e.g. the moon
- "Wrap" lighting is a simple modification of the normal Lambert diffuse function
- **diffuse = (dot(L, N) + wrap) / (1 + wrap)**
- Causes lighting to "wrap" around object beyond the normal 90 degrees
- Can bake function into texture map
- Means less ambient, fill lighting is required

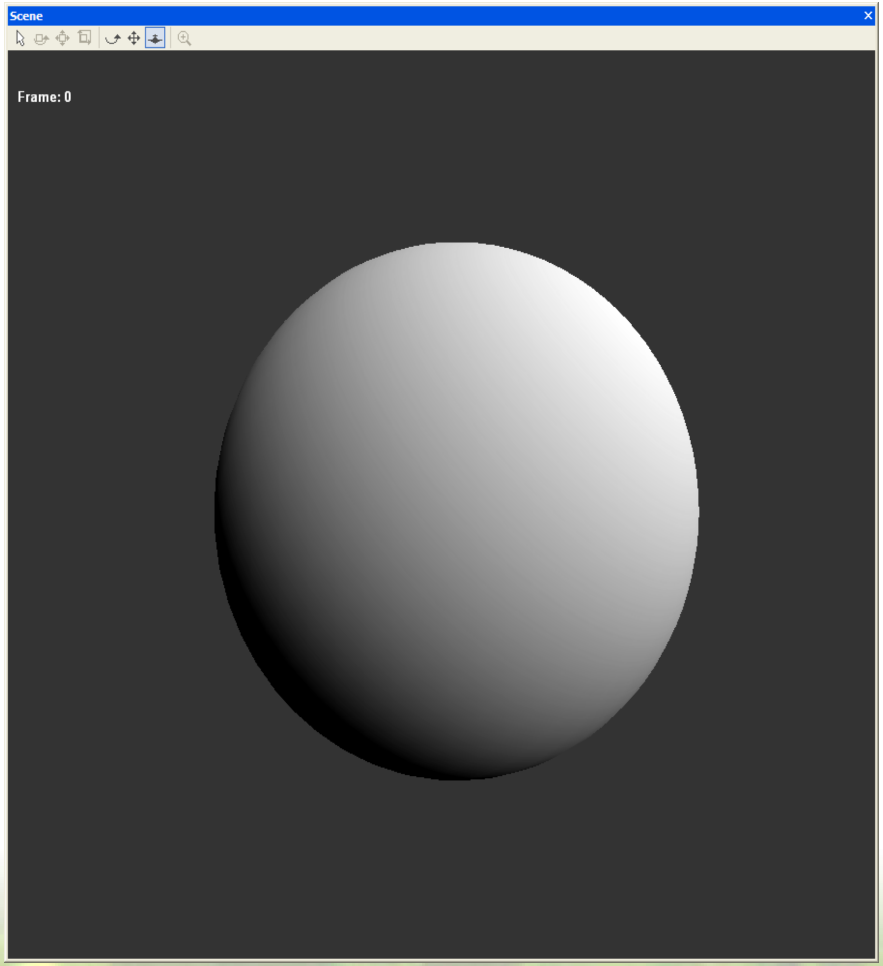
Wrap lighting function



Without Wrap Lighting



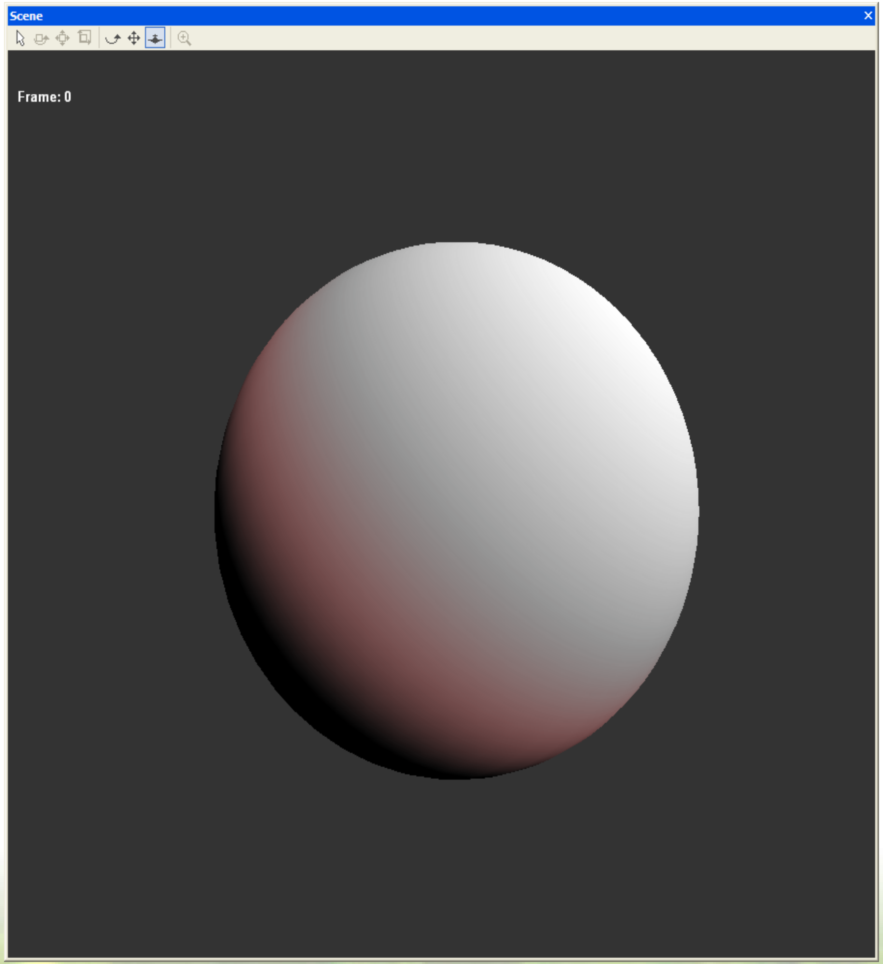
With Wrap Lighting



GameDevelopers Conference

EVOLVE GDC>04

Wrap Lighting with Color Shift



NVIDIA.

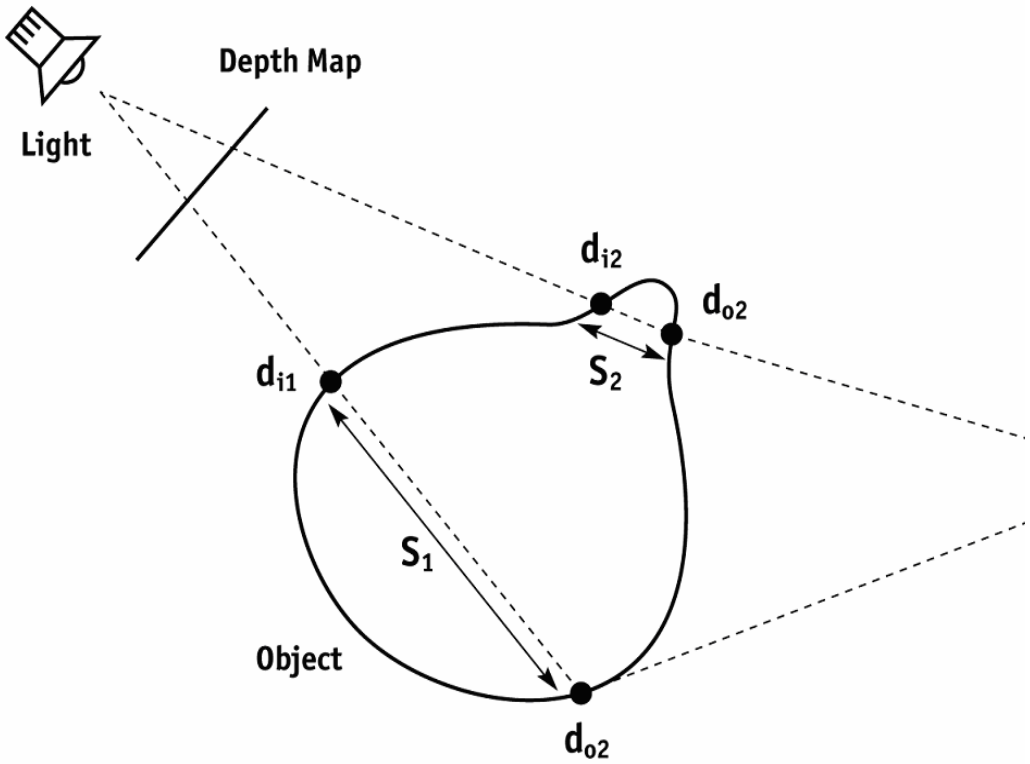
Depth-Map Based Scattering

- The distance light travels through the material is an important factor in scattering
  - The further it goes, the more of it is absorbed and scattered away
- We can use depth maps to measure this
- Very similar to Greg's technique, but from the point of view of the light
- Technique first described by Christophe Hery (ILM), see 2002/2003 Siggraph Renderman Course Notes

Depth-Map Based Scattering

- Very similar to shadow mapping
- Depth map pass:
  - Render scene from point of view of light
  - Store distance from light to texture
- Second pass:
  - Shader calculates distance of shaded point from light
  - Looks up in depth texture to get distance from light at entry point
  - Subtracts the two to get thickness

Using a Depth Map to Measure Thickness

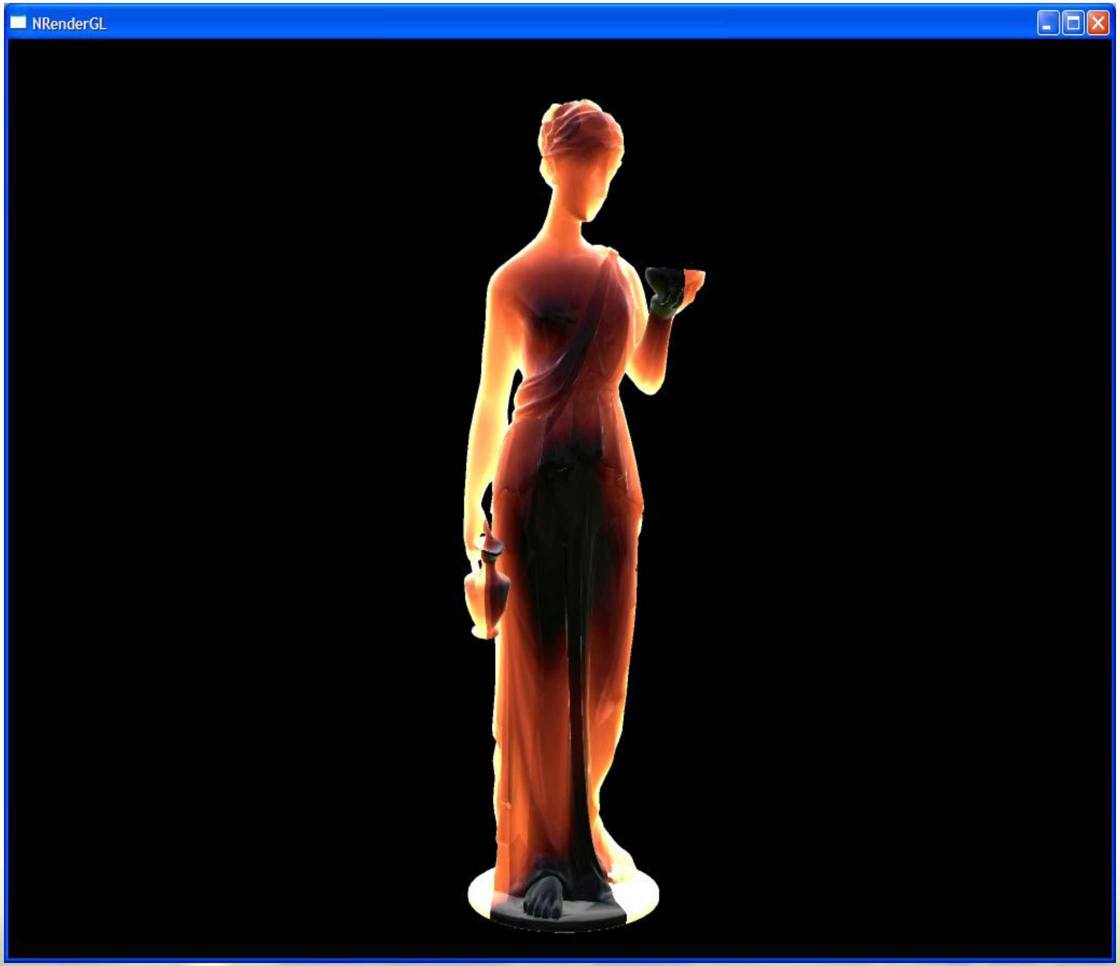


Shading

- What to do with the thickness value?
- Can use it directly to index into an artist-created 1D color table
- Intensity should fall off exponentially with distance
- Should also take into account
  - Fresnel effect at entry and exits points (requires normal at entry point)
  - Refraction
  - Color map

Depth-Map Scattering Example

Depth-Map Scattering Example



More Sophisticated Models

- Using a single depth map sample is cheap, but has artifacts
  - Doesn't simulate diffusion (no blurring)
  - Features from backside of model will be visible
- More sophisticated single scattering approximations march along refracted ray, taking multiple samples
  - Use phase functions to describe directions light is scattered when it hits a particle
- Multiple scattering models
  - Use diffusion approximation to simulate multiple scattering in highly scattering media such as skin

Depth-Map Based Scattering

- Disadvantages
  - Only works with convex objects, holes are not accounted for correctly (not a big problem in practice)
  - Could use Greg's technique to solve this
- Advantages
  - Works for animating objects
  - No pre-calculation necessary

Caveat – Uniform density

- nose & fingers are same thickness as ears, but ears let more light through
- need to account for what is under the surface
- more painting of maps for bone, flesh, blood
- can get to be a lot of work

Texture Space Diffusion

- One of the observed effects of subsurface scattering is a general blurring of the lighting
- Artists often use 2D tricks in Photoshop
  - Gaussian blur image, add a percentage back on top of original image
  - Sometimes called glow / bloom
- Why can't we do this in real-time?
- We can, and we can do it in UV texture space instead of screen space
- Technique first described by George Borshukov in "Realistic Human Face Rendering for "The Matrix Reloaded"

The Matrix Reloaded

Texture Space Diffusion

- Render model unwrapped to UV space
  - Render model with diffuse lighting, but using UV texture coordinates as position
  - Requires good, unique UV mapping
  - Generates 2D light-map
- Blur light-map using normal techniques
  - separable convolution, make use of bilinear filtering
  - Can blur different color channels by different amounts to simulate different mean free paths of wavelengths
  - For skin, blur red channel more than green and blue
- Render model with blurred light-map
  - shader combines with color map and regular lighting

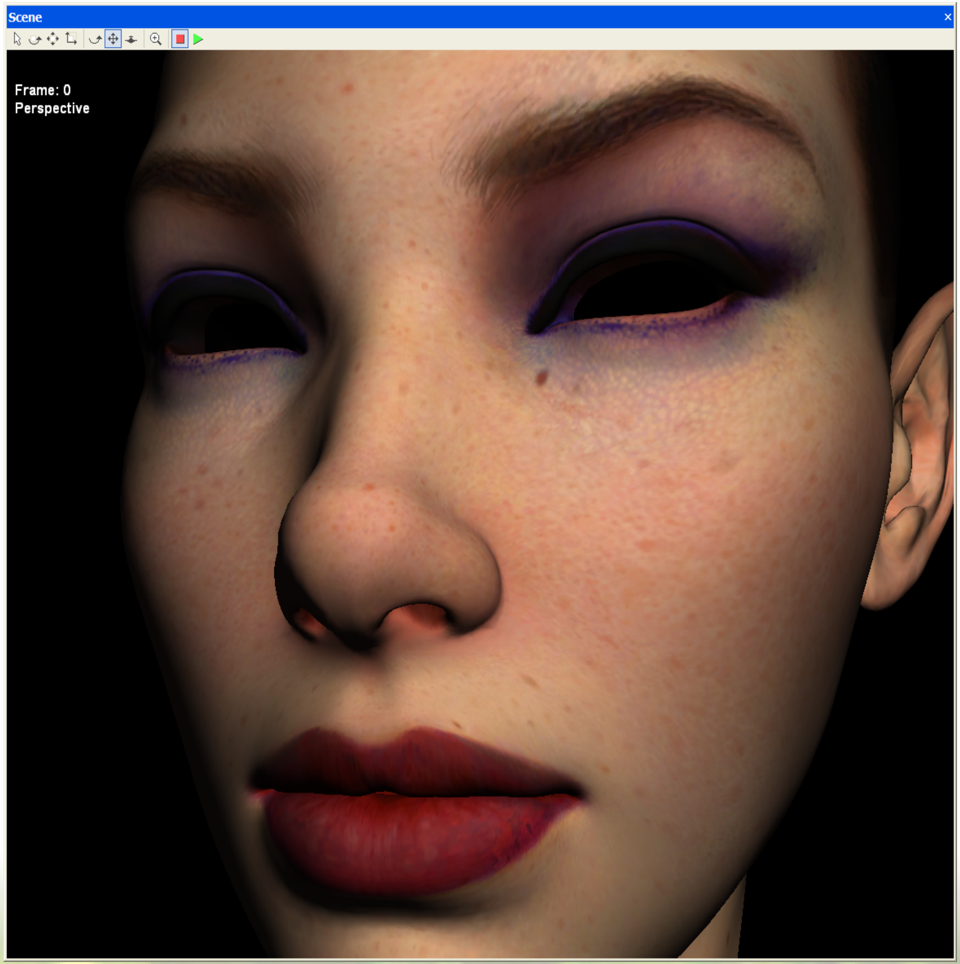
Lightmap Before Blurring

Lightmap After Blurring

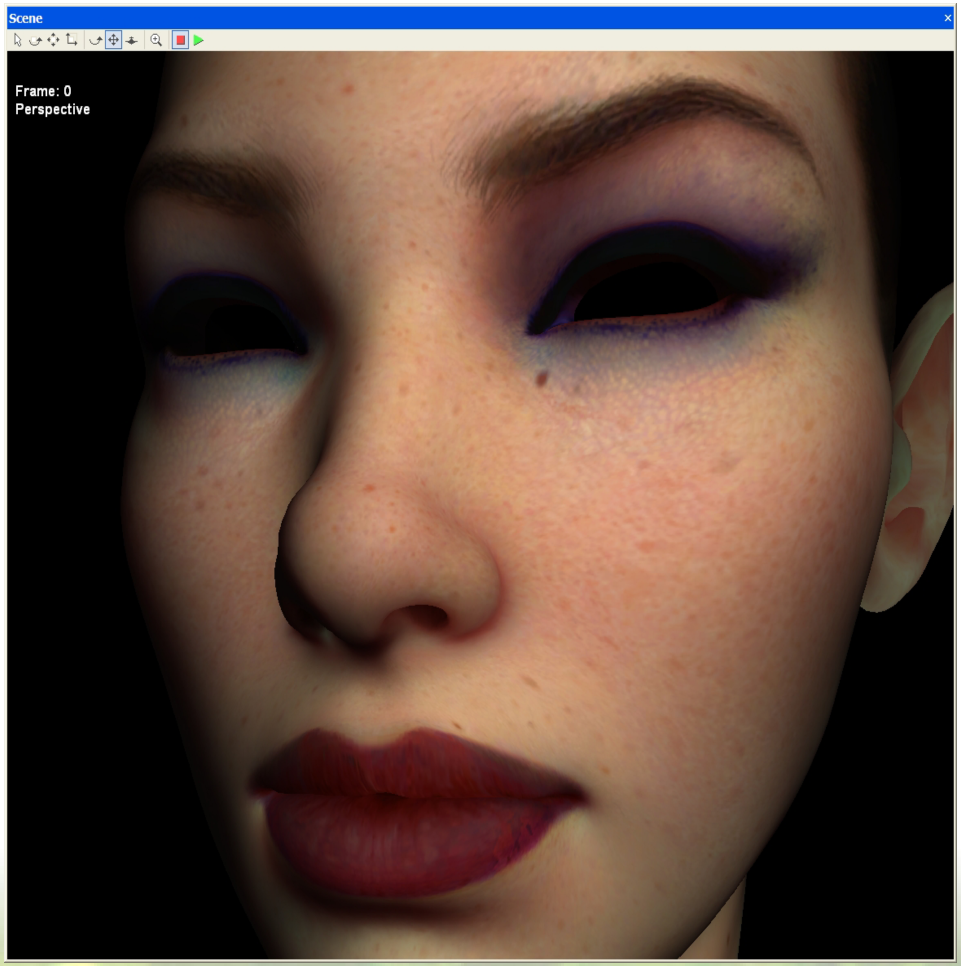
Original Lighting

With Blurred Lightmap

Dusk - No Diffusion



Dusk - With Diffusion



Future Work

- Combine depth-map technique with texture space blurring
- Use fp16 blending for measuring thickness
- Experiment with depth-peeling
- Use several color maps for different skin layers (surface, veins etc.)

Conclusion

- Scattering can help take your game characters to the next level of realism
- 90% of the look of a full BSSRDF simulation can be achieved using cheap approximations

References

- NVIDIA SDK available online at http://developer.nvidia.com
- Borshukov, George, and J. P. Lewis. 2003. “Realistic Human Face Rendering for ‘The Matrix Reloaded.’” SIGGRAPH 2003. Available online at http://www.virtualcinematography.org/
- Everitt, Cass. 2003. “Order-Independent Transparency.” Available online at http://developer.nvidia.com/view.asp?IO=order_independent_transparency[CK1]
- Hery, Christophe. 2002. “On Shadow Buffers.” Presentation available online at http://www.renderman.org/RMR/Examples/srt2002/PrmanUserGroup2002.ppt
- Hery, Christophe. 2003. “Implementing a Skin BSSRDF.” RenderMan course notes, SIGGRAPH 2003. Available online at http://www.renderman.org/RMR/Books/sig03.course09.pdf.gz
- James, Greg. 2003. “Rendering Objects as Thick Volumes.” In *ShaderX2: Shader Programming Tips & Tricks With DirectX 9*, ed. Wolfgang F. Engel. Wordware. More information available online at http://www.shaderx2.com/
- Jensen, Henrik Wann, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. 2001. “A Practical Model for Subsurface Light Transport.” *Proceedings of SIGGRAPH 2001*.
- Mech, Radomir, “Hardware-Accelerated Real-Time Rendering of Gaseous Phenomena.” Journal of Graphics Tools, 6(3):1-16, 2001. http://www.acm.org/jgt/papers/Mech01/
- Nayar, S. K., and M. Oren. 1995. “Generalization of the Lambertian Model and Implications for Machine Vision.” *International Journal of Computer Vision* 14, pp. 227–251.
- Pharr, Matt. 2001. “Layer Media for Surface Shaders.” Advanced RenderMan course notes, SIGGRAPH 2001. Available online at http://www.renderman.org/RMR/Books/sig01.course48.pdf.gz