Interactive Image-Space Techniques for Approximating Caustics

Chris Wyman *Scott Davis*
The University of Iowa

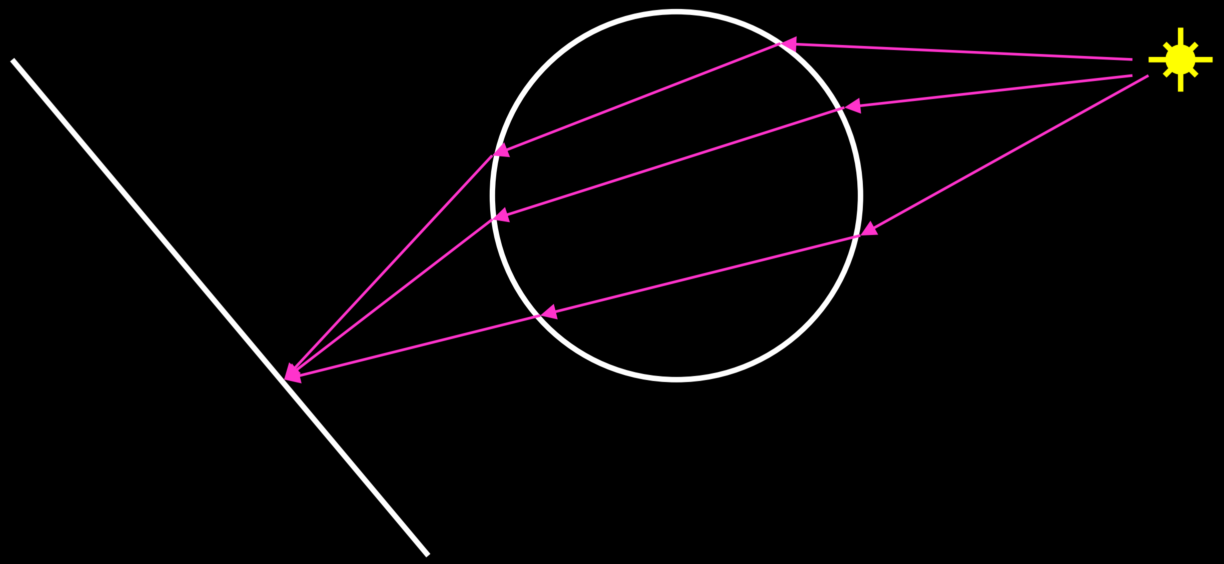
The University of Iowa

Motivation

- Realistic rendering
  - – Anytime a specular object reflects / refracts

What is a Caustic?

- Focusing of light
  - – Key idea: a point “sees” light multiple times



Challenge for Interactivity

- High frequency
  - Need lots of photons
  - Interpolation leads to blurring
- Photons *converge* & *diverge*
  - Rasterization likes coherent rays
- Complex data structures
  - E.g., kd-trees or 3D grids
  - Not GPU friendly

THE UNIVERSITY OF IOWA

Previous Work

- Particle tracing
  - Path tracing [Kajiya86]
  - Photon mapping [Jensen96]
  - Parallel CPUs for interactivity [Wald02]
  - With hardware acceleration [Purcell03,Woop05]
  - Tracing through object samples [Wand03]
- Beam tracing
  - For underwater caustics [Watt90]
  - Underwater caustic volumes with scattering [Nishita94]
  - Hardware accelerated caustic volumes [Iwasaki02]
  - Warped caustic volumes [Ernst05]

Previous Work

- Results depend on specular model
  - Planar geometry [Diefenbach97]
  - Virtual geometry [Ofek98, Schmidt03]
  - Pseudo-refraction [Oliveira00, Hurley00, Lindholm01]
  - Light field reflections [Yu05]
  - Image-space refractions [Wyman05]
- Better specular models gives better caustics
  - Reflection & refraction an important part of caustics research

Rendering Caustics

- Usually a two-pass process:
  – Shoot photons from the light
    - Record final locations
    - Store in data structure for easy access
  – Render from the eye
    - At visible surface points find “nearby” photons
    - Weight and average nearby photons
    - Add to direct lighting

Our Algorithm

- Maintains standard two-pass process
  - Splits gather into 2 passes

1. Use GPU to find photons hit locations
2. Determine photon locality
   - Three options:
     - Render “caustic triangles” onto surfaces *(see paper)*
     - Render photons into light-space buffer
     - Render photons directly in eye-space
3. Render from eye

THE UNIVERSITY OF IOWA

Implementation

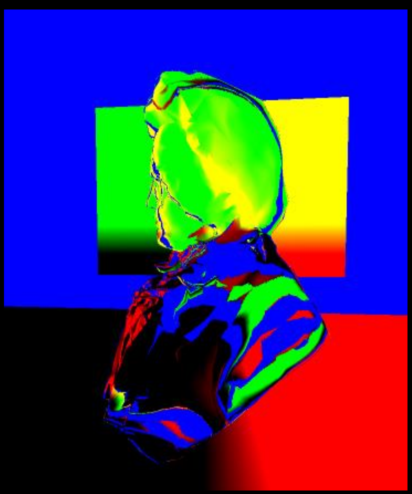
1. Render photons from light view
   - Store photon positions and depth map
   - (Optional) store photon direction, attenuation, surface normal
2. Render photons (as points)
   - Splat photons or perform image-space Gaussian blur
   - Draw into light-space map or directly in eye-space
3. Render from eye
   - Project light-space map like a shadow map

Photon Shooting

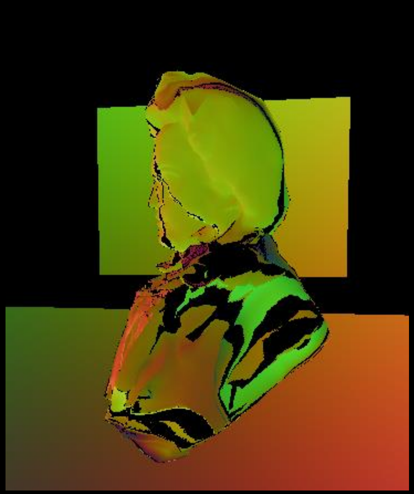
- Render from light
  - Use any existing reflection or refraction technique
  - Similar to shadow mapping

Standard Rendering From Light's View

Depth Map



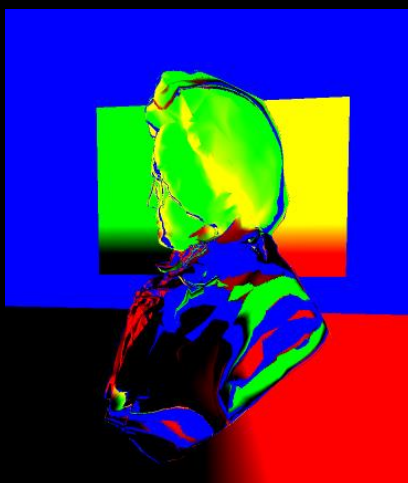
Final Photon Locations



Incident Photon Direction

THE UNIVERSITY OF IOWA

Photon Gathering



Use photon data in vertex shader, render points at photon locations

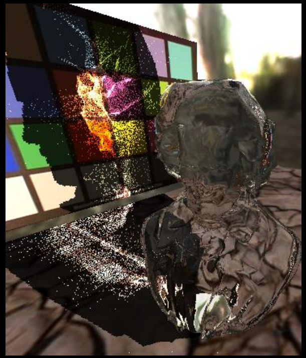
Light Space

Eye Space

The University of Iowa

Photon Gathering

- Directly visualizing photons is noisy
  - Cannot feasibly shoot enough
  - Photons are ideal points
    - Want to find irradiance (over area of a visible pixel)
  - Average over "nearby" photons



- Without complex data structure:
  - Find nearby photons in image space!
    - [Gunther04] proposed for distributed ray traced caustics

THE UNIVERSITY OF IOWA

Photon Gathering

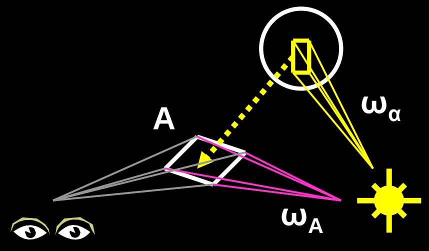
- For real-world distance ε to search:
  - Some NxN region contains all photons within ε
  - Region also contains extra photons
  - Used stored position to ignore extra photons
  - We used a fixed 7x7 region
    - Equivalent to a fixed-size nearest neighbor query

- Or splat photons into image-space
  - Use alpha-blending
  - Not discussed in paper… But faster!

Intensity Determination

- Gather radiance over small area
  - Assumes all units physically based
  - OpenGL / DirectX apps usually use arbitrary units
- Idea: Caustics see the light multiple times
  - Count how many times photons hit a point
  - Use the *ratio* at each pixel:
    - (# photons w/caustics) / (# photons w/o caustics)
  - Be aware: Not all photons are equal!

Intensity Determination



- Each photon has energy:

$$E_\alpha = (\omega_\alpha / 4\pi)\, E_L$$

  – Based on solid angle (ω_α) photon covers

- Each eye-view pixel corresponds to region A
  – This region subtends some solid angle (ω_A) seen from the light
  – Without caustics, A would receive energy:

$$E_A = (\omega_A / 4\pi)\, E_L$$

- Total energy can then be computed (a ratio):

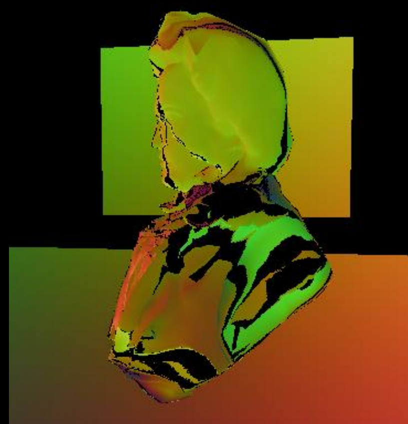
$$E_{total} = \sum_{\alpha_i \text{ in } A} (E_{\alpha_i} / E_A)$$

(See paper for computing ω_A)

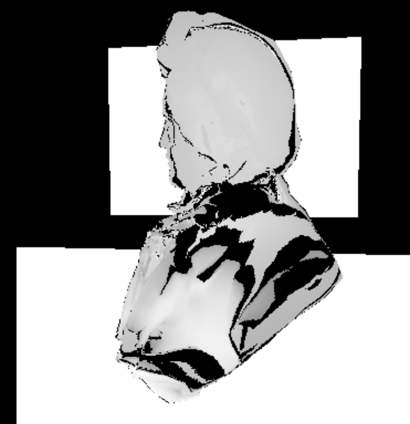
THE UNIVERSITY OF IOWA

Non-Diffuse Receivers

- Irradiance not always enough!
  - Specular surfaces get different contribution from each photon
  - When splatting or gathering photons, account for direction



Incident photon direction

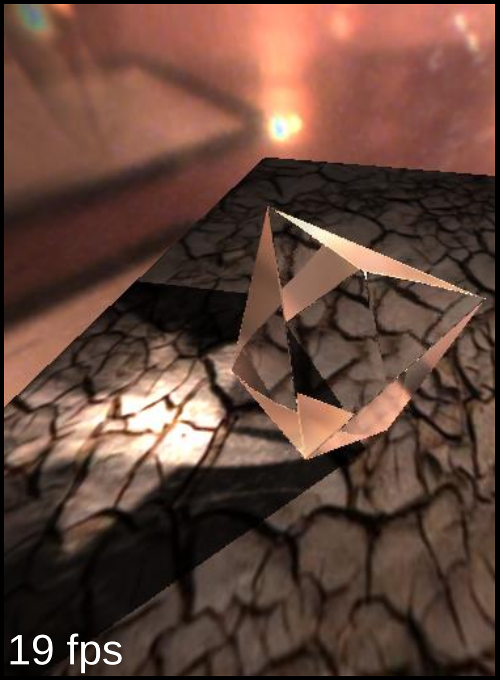


Photon attenuation
(Fresnel or volumetric)

- Also consider photon attenuation

THE UNIVERSITY OF IOWA

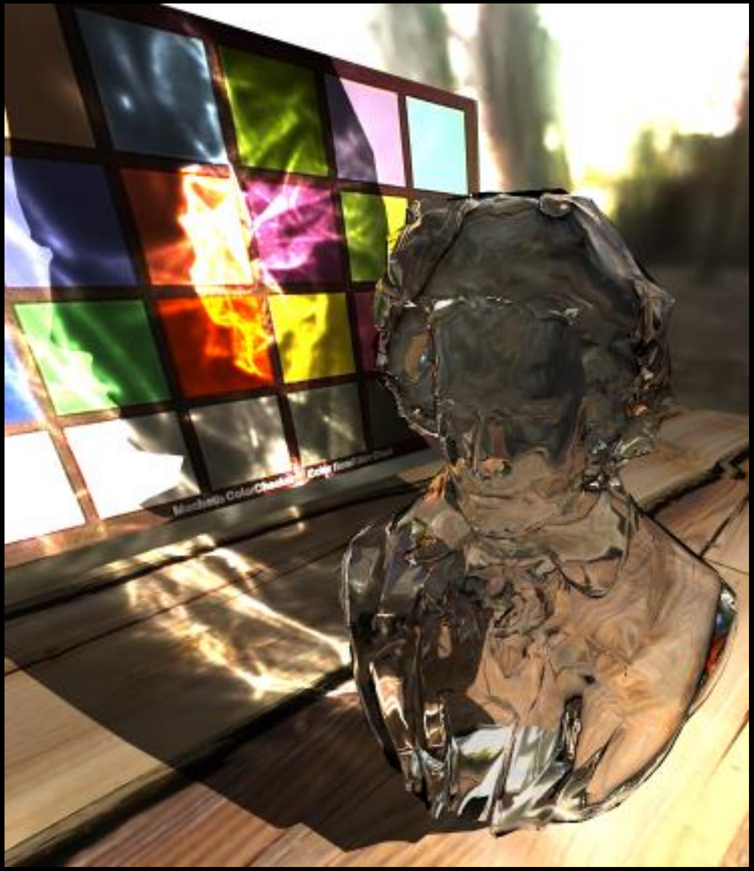
Results



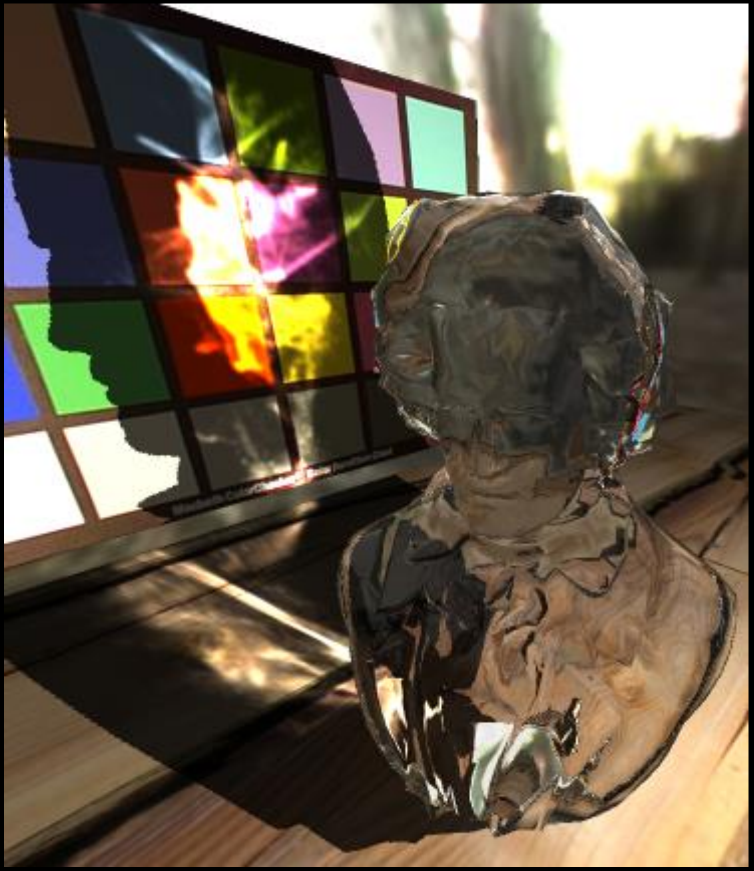
Rendering speed dominated by photon count, as each photon is rendered as a vertex.

The University of Iowa

Results



Photon Map



Our Approach
(Note: Only refracted photons traced)

Demo

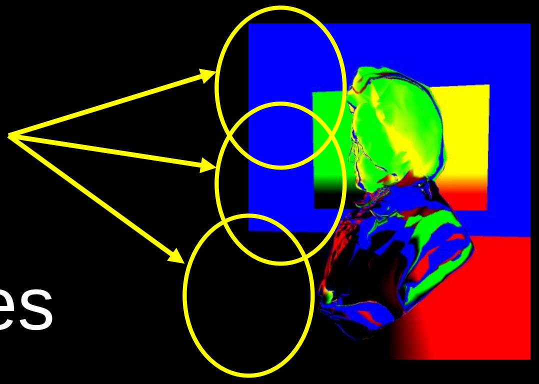
- 1.83 GHz Intel Core Duo
- nVidia GeForce Go 7800, 256 MB

- Framerates:
  - 10 – 30 fps
  - 512^2 photons (about 1/3 hit refractors)
  - 512^2 image resolution
  - Models 1,000 – 100,000 triangles

- Gathering photons over multiple frames reduces noise

Limitations

- Noise
  - Under sampling
  - Cannot gather from dynamically sized nearby regions
  - Coherency between frames difficult for dynamic scenes
- Inherited problems from specular techniques
  - Ray - b/g object intersection speed a factor for us
- "Wasted" photons
  - Not all photon buffer pixels useful
  - Expensive waste
- Limited to point light sources

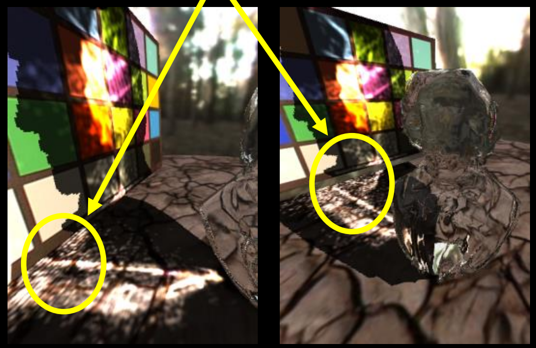


THE UNIVERSITY OF IOWA

Conclusions

- Adapts photon mapping for GPUs
  - Image-space gather instead of a search
- Physically based
  - Approximation errors come from refraction technique
- Can gather in either eye- or light-space
  - Tradeoff: blur ↔ noise
  - Tradeoff: aliasing ↔ varying gather with viewpoint

Suggest: gather in light-space



Future Work

- Reducing noise
  - Adaptive photon buffer?
  - Perspective photon buffer?
  - Dynamically sized nearby neighbor search?
    - Maintain crisp caustics while reducing noise
    - Use fewer photons without temporal coherency issues
    - Good preliminary results
- Importance sampling
  - Eliminate “wasted” photons

Questions?