SIGGRAPH2005

An Approximate Image-Space Approach for Interactive Refraction

Chris Wyman

The University of Iowa

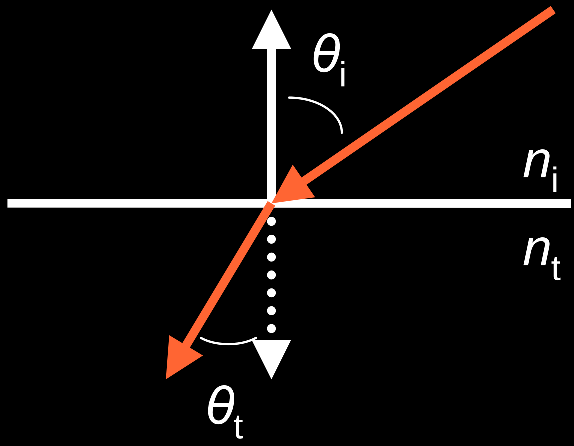
Rendering Refraction

SIGGRAPH2005

- Governed by Snell's Law:

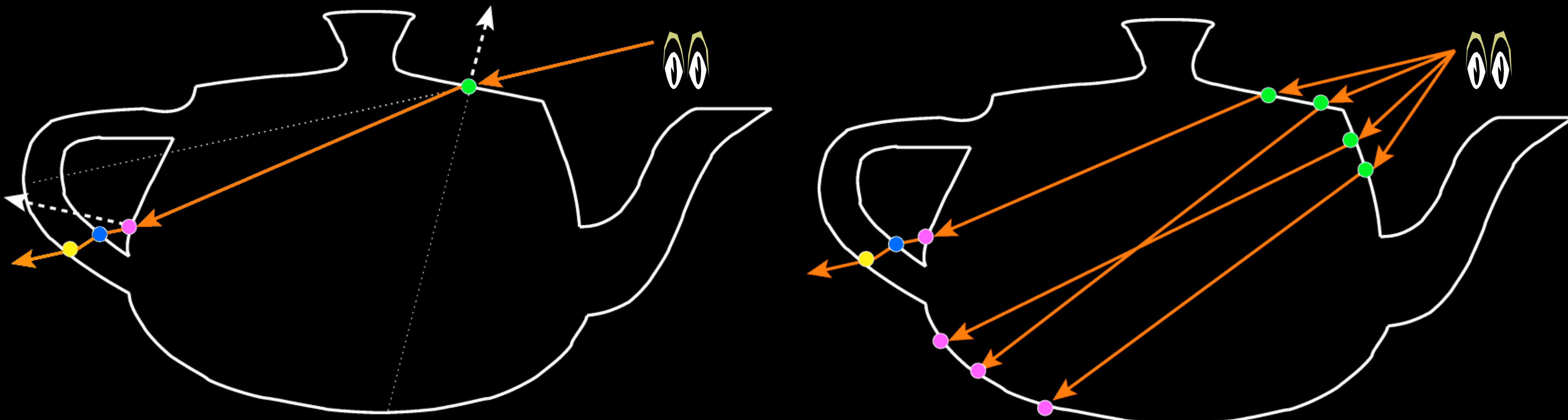
$$n_i \sin \theta_i = n_t \sin \theta_t$$

- To compute, need:
  - Incident direction
  - Surface normal
  - Indices of refraction



- What makes interactive refractions hard?
  - After refraction, rays no longer coherent

Ray Tracing versus Rasterization



- Ray tracing follows one ray at a time
  - Coherency not an issue
- Rasterization examines > 1 ray in parallel
  - Refracted rays not coherent, have different origins, and vary in behavior

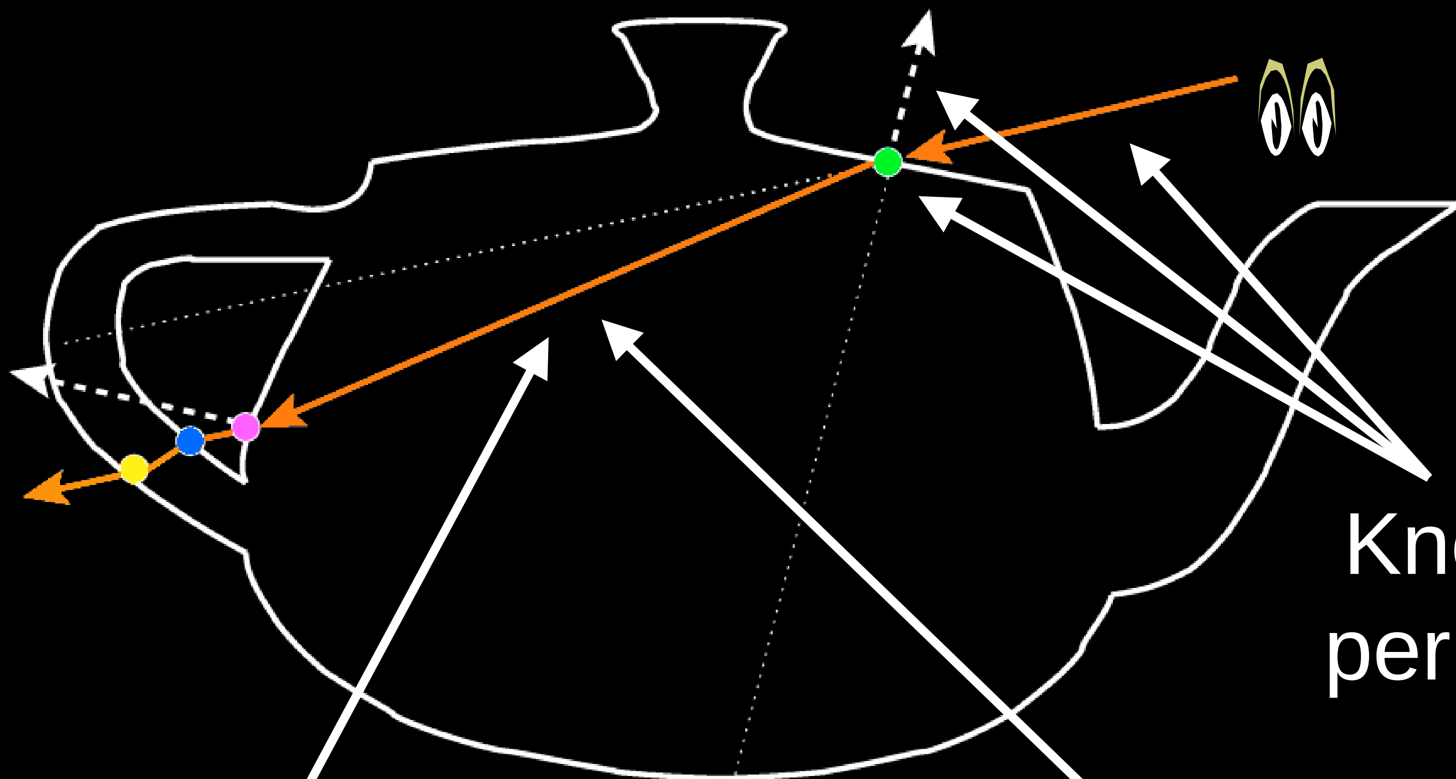
Previous Interactive Work

SIGGRAPH2005

- Ray tracing [Parker 99, Wald 02, Purcell 03]
- Light fields [Heidrich 99]
- Rasterization-based methods
  - Thickness parameter [Kay 78]
  - Multi-pass approach [Diefenbach 97]
  - One pseudo-refraction [Oliveira 00, Hurley 00, Lindholm 01]
  - Virtual geometry [Schmidt 03]
  - GPU vertex computation [Guy 04]

Observations

SIGGRAPH2005



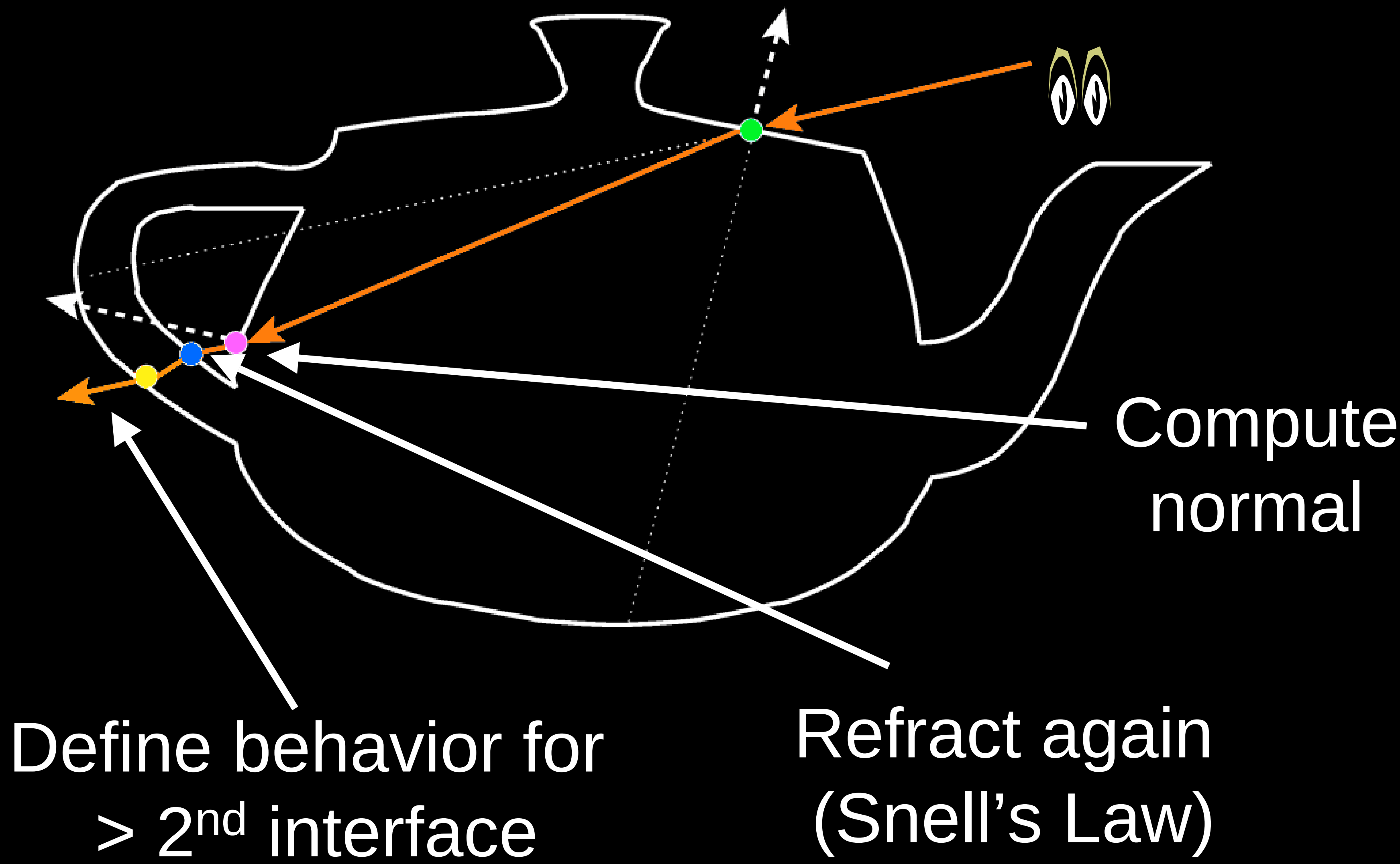
Known per pixel

To find 2nd point, need length of refracted ray

Can compute direction at each pixel (Snell's Law)

Observations

SIGGRAPH2005



Problems to Solve

- Computing distance between 1st & 2nd refraction points
- Finding normal at second refraction
- Defining behavior for > 2 interfaces
  - For additional refractions
  - For total internal reflection

Steps to Refract

SIGGRAPH2005

1) Use data at front surfaces for 1st refraction
   - Viewing direction
   - Surface normal

+

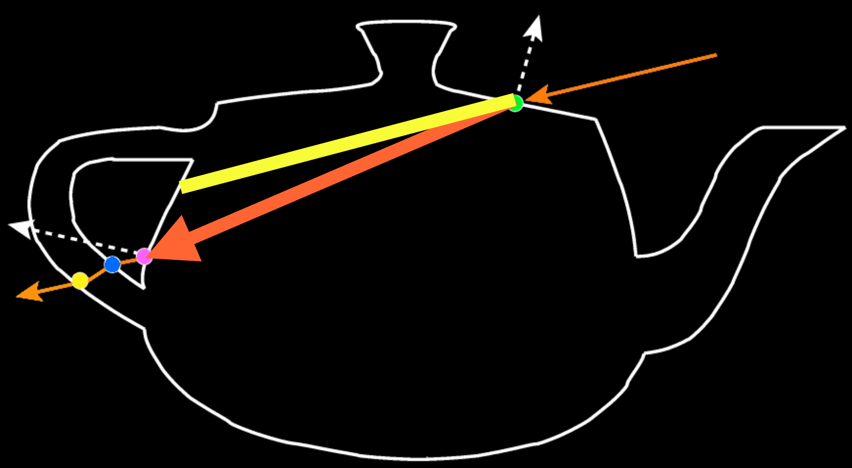
+ indices of refraction =

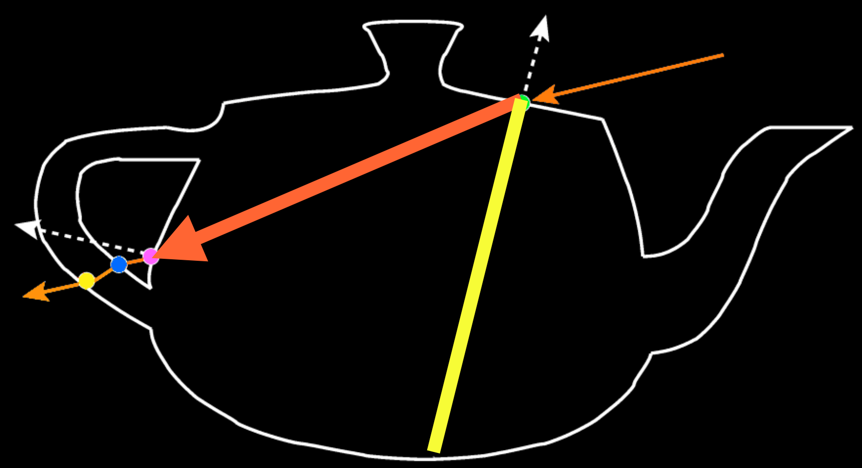
Steps to Refract

SIGGRAPH2005

2) Find distance to second refraction location

- Intersecting geometry directly often too expensive
- Need approximation easy to compute on GPU
- Two come to mind d_V and d_N

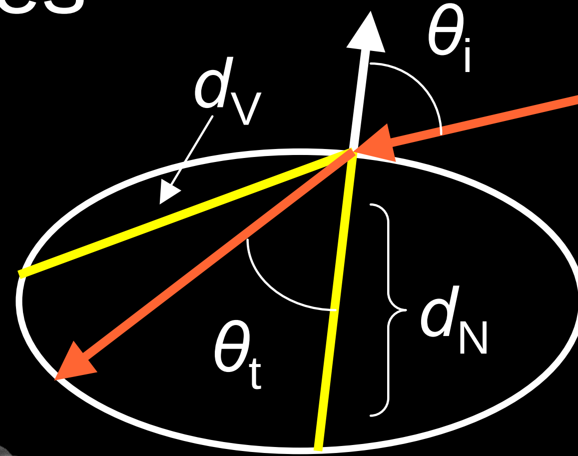


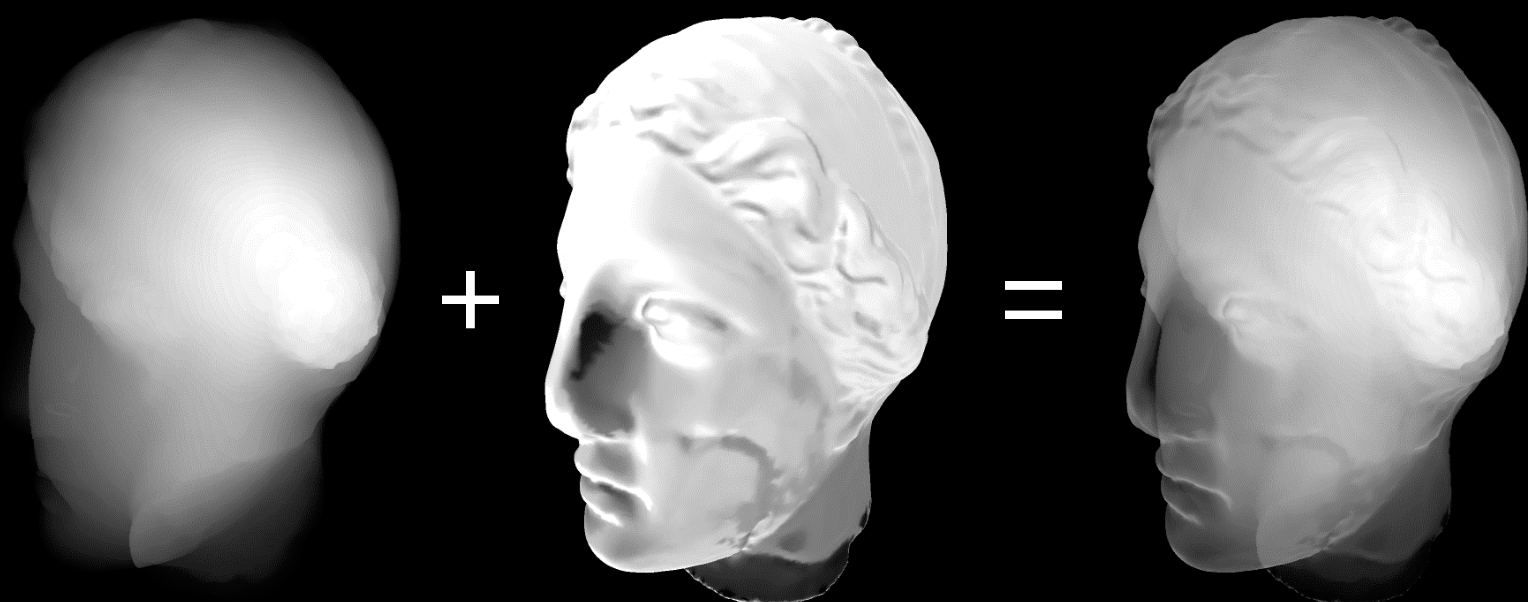


Length of 1st Refracted Ray

- Interpolate between two distances
  - Linear interpolation based on angle

$$\tilde{d} = \frac{\theta_t}{\theta_i} d_{\vec{V}} + \left(1 - \frac{\theta_t}{\theta_i}\right) d_{\vec{N}}$$



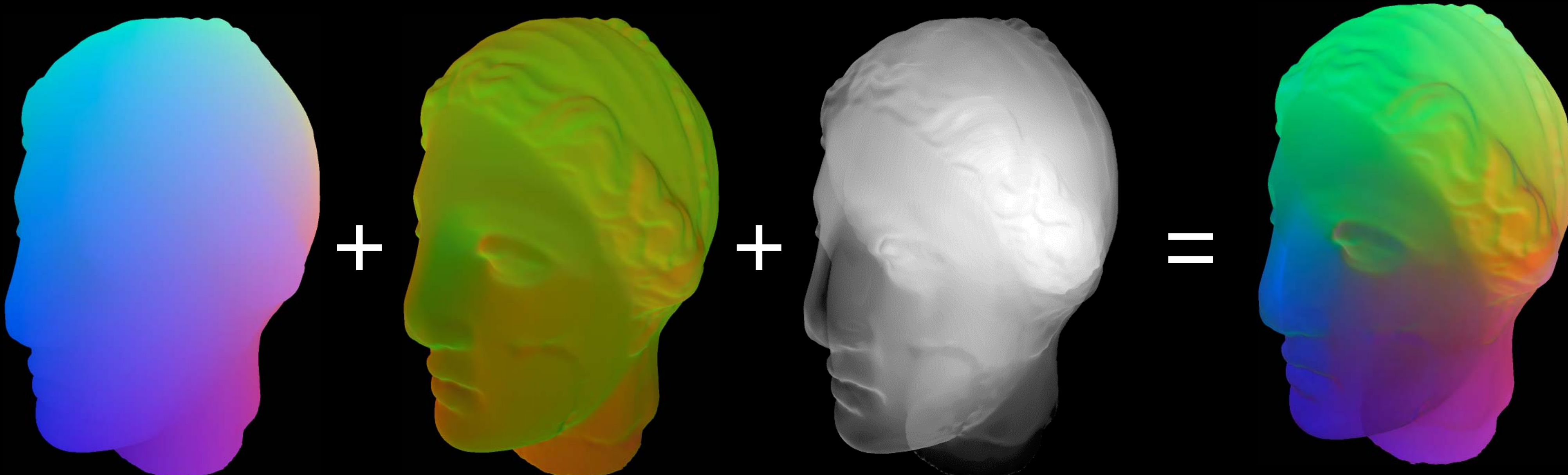


Steps to Refract

SIGGRAPH2005

3) Approximate 2nd refraction location, using

- First refraction location
- Transmitted direction
- Approximate length of refracted ray

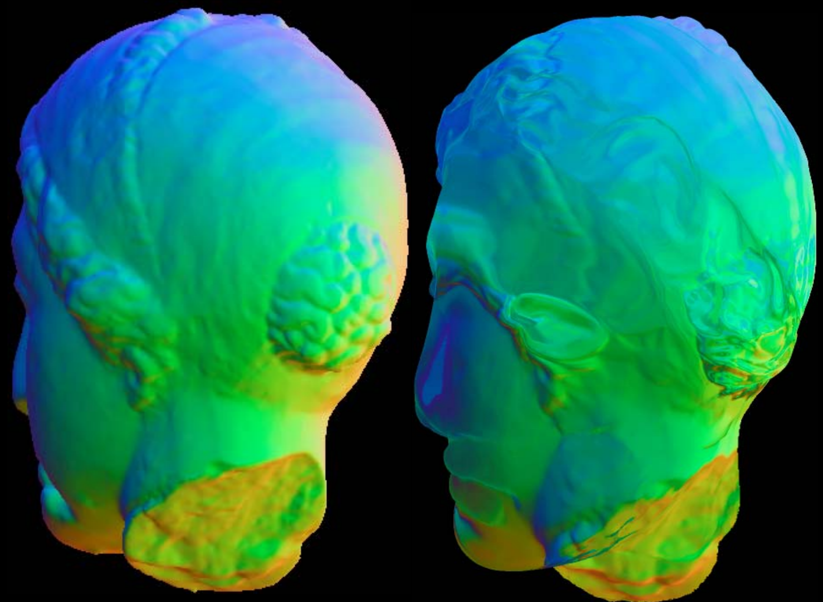


Steps to Refract

SIGGRAPH2005

4) Find surface normal at 2nd refraction
   - Hard to access geometry information on GPU
   - Unless it is in a texture!

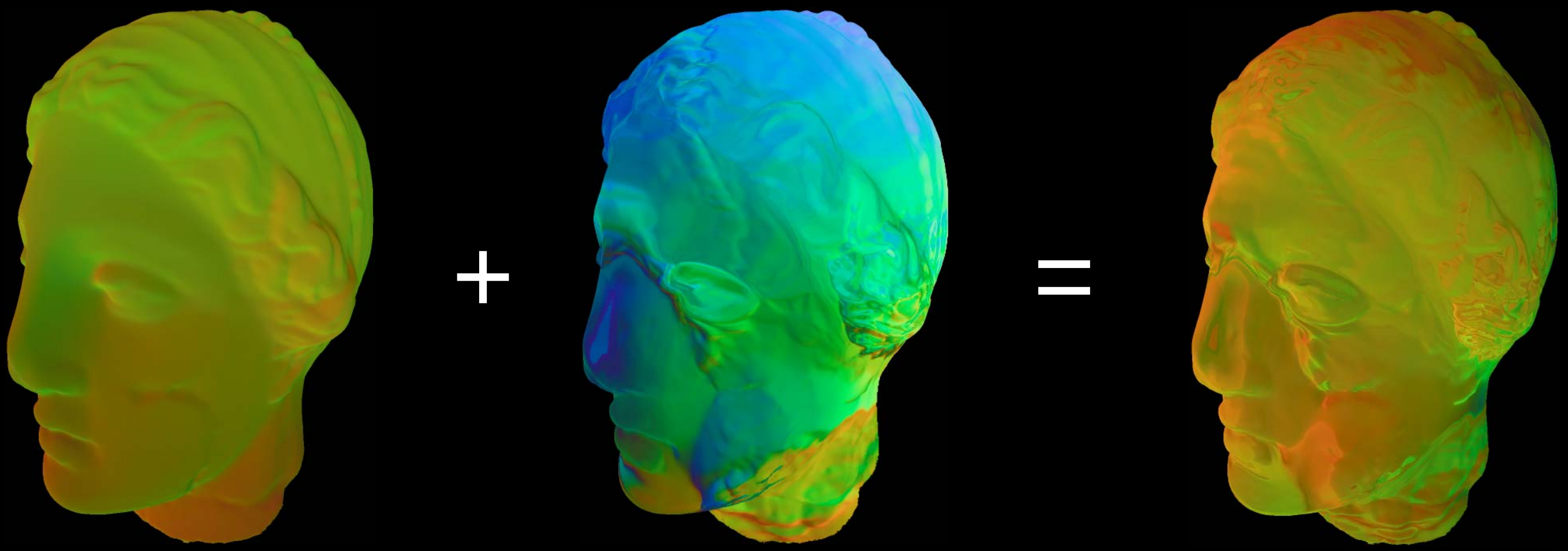
- Draw back faces to texture
  - Use normal as color
  - Project 2nd refraction position into buffer to find normal



Steps to Refract

5) Refract 2nd time by applying Snell's Law

- First refraction ray
- Projected normal at second intersection point

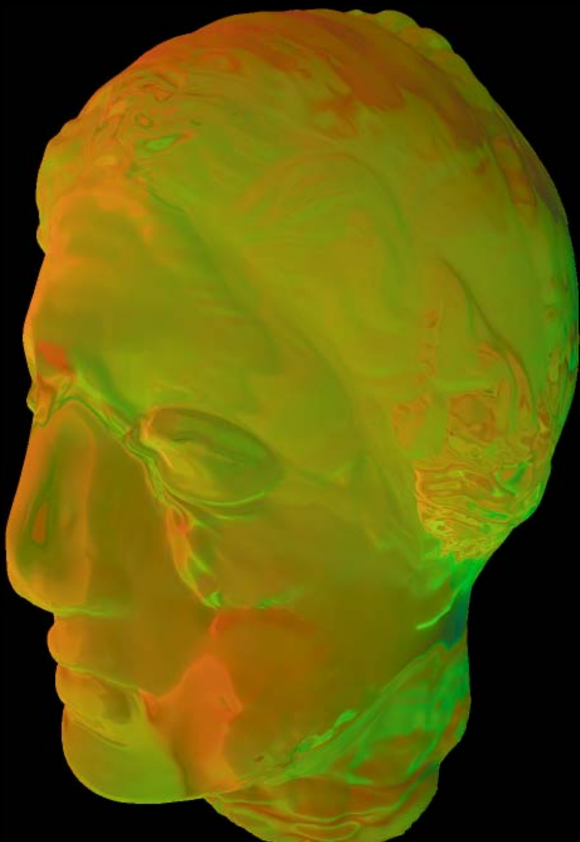


Steps to Refract

SIGGRAPH2005

6) Use 2nd refraction, index into environment

– Ignore additional refractions



+

=

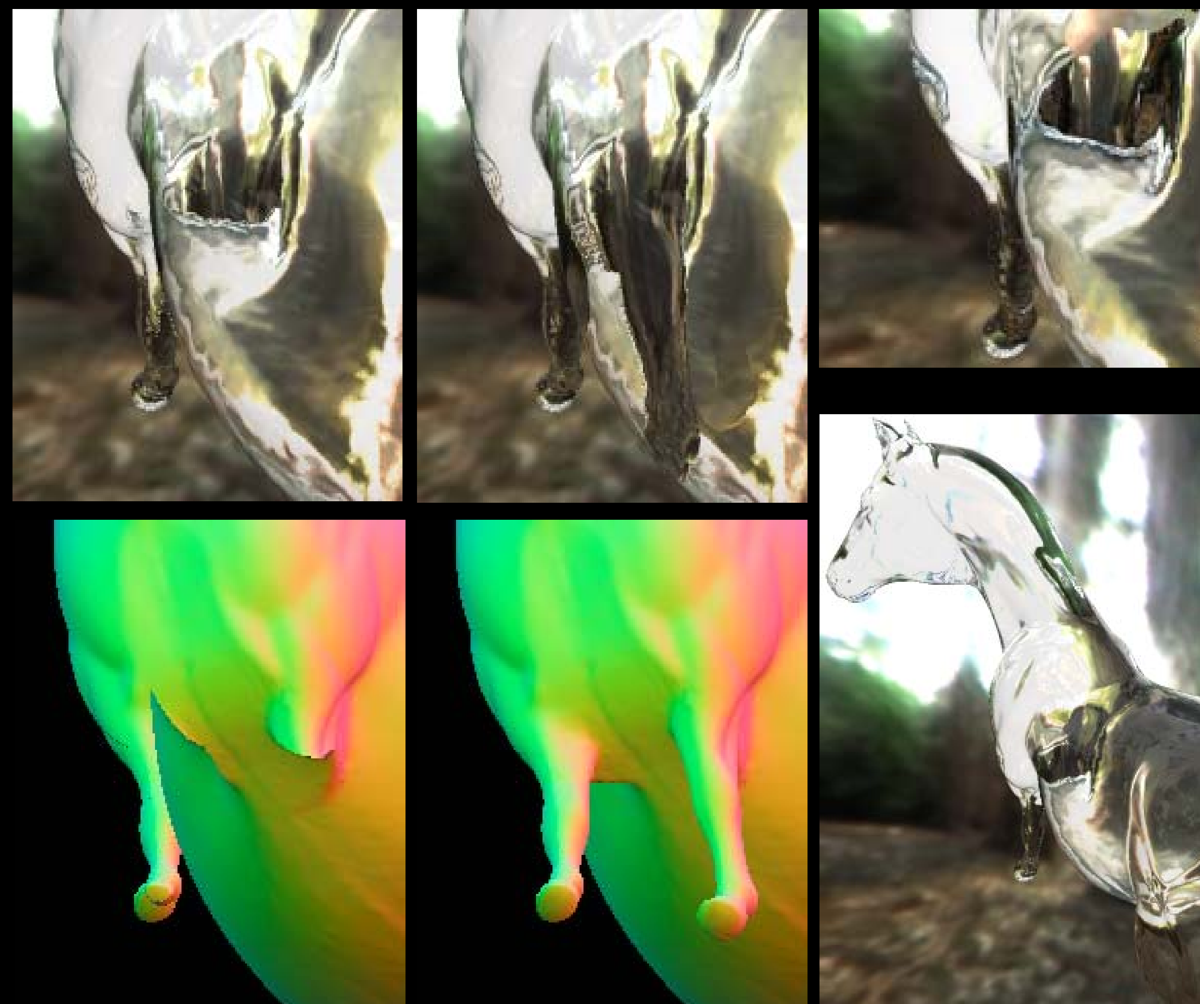
Problems

SIGGRAPH2005

- Standard texture problems
- Distance approx. is *smooth*!
  – For complex objects, should have discontinuities
  – But index into texture…
- Depth complexity > 2
- Total internal reflection
  – Must deal with this

High Depth Complexity

- Problem:
  - Normal map only stores one surface
  - Refracting surface could be in front or behind stored point



Total Internal Reflection

- Three possible methods

Continue straight

Refract (alter angle)

Reflect

Implementation

SIGGRAPH2005

- Split these 6 steps into two online passes
  - Precompute d_N for each refractive model
  - 1^{st} draws back facing normals & z-buffer
  - 2^{nd} does the rest of the work
- Developed on AGP GeForce 6800, 128 MB
  - Probably could run on older cards with work
- Times running on P4 3.0 GHz, 2 GB

Results (at 1024^2)

SIGGRAPH2005

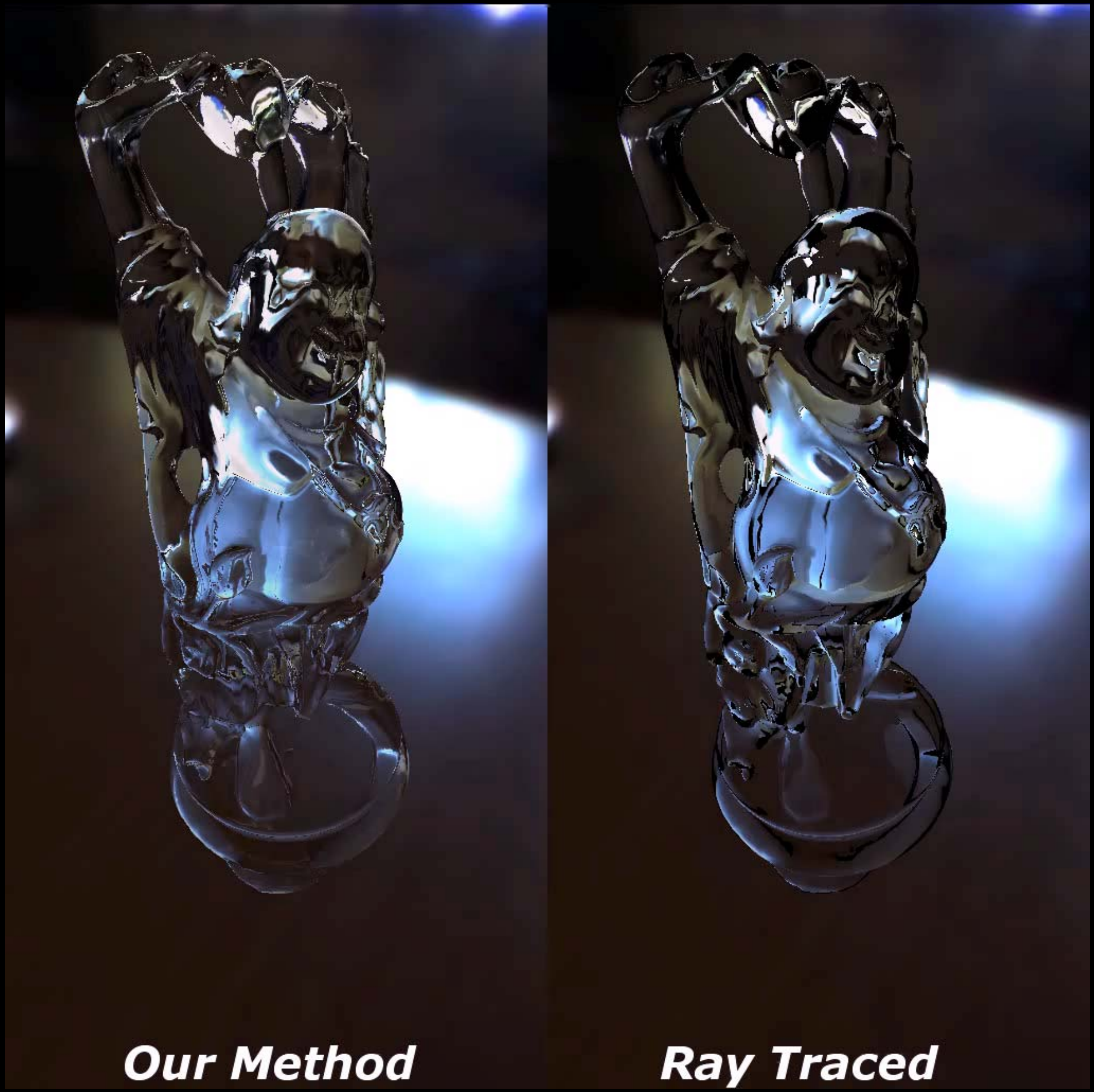
50,000 triangles

$n_t = 1.2$

One sided: 137 fps

Two sided: 53 fps

Ray traced: 122 sec



Results (at 1024^2)

SIGGRAPH2005

Venus (100k), 38 fps

$n_t = 1.3$

F-16

(4500)

149 fps

$n_t = 1.2$

Dragon

(250k)

21 fps

$n_t = 1.2$

Summary

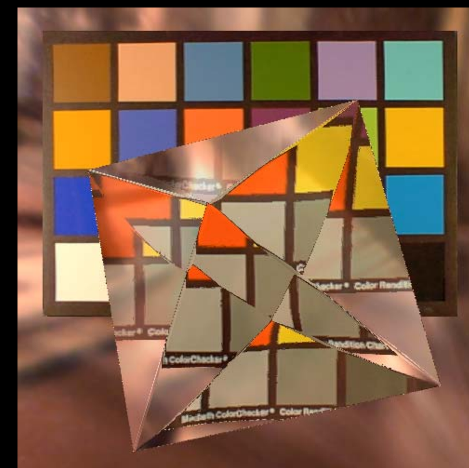
SIGGRAPH2005

- Two pass, approximate refraction
  - Easy to understand and implement
  - Fast rendering speed
  - Plausible results
  - Deformable models with less accuracy (use only d_V)
- Problems
  - Limited to two refractions
  - Infinite environments

Future Work

SIGGRAPH2005

- Refract background geometry
  - Could approach as dynamic per-pixel displacement mapping



- With background geometry, can render interactive caustics
  - Image courtesy Scott Davis

- Add more than two interfaces

Questions?

SIGGRAPH2005

Thanks!