CS 563 Advanced Topics in Computer Graphics

Realistic Transparency

by Nik Deapen
- Fresnel Equations
- Color Filtering
- Implementation
- Some Topics
- Photon Mapping and Caustics
In Chapter 27, \( kr \) and \( kt \) were constant. Now they depend on the incidence angle and the relative dielectric constants.

\[
\begin{align*}
    r_{||} &= \frac{\eta \cos \theta_i - \cos \theta_t}{\eta \cos \theta_i - \cos \theta_t} \\
    r_i &= \frac{\cos \theta_i - \eta \cos \theta_t}{\cos \theta_i - \eta \cos \theta_t} \\
    \eta &= \frac{\eta_{in}}{\eta_{out}} \\
    k_r &= \frac{r_{||}^2 + r_i^2}{2} \\
    k_t &= 1 - k_r
\end{align*}
\]
Simple vs Dielectric (Fisheye)
- When you shoot from the direction of the normal (Normal Incidence)

\[
\begin{align*}
    r_{||} &= \frac{\eta - 1}{\eta + 1} \\
    r_{\perp} &= -\frac{\eta - 1}{\eta + 1} \\
    k_r &= \frac{\eta - 1^2}{\eta + 1^2} \\
    k_t &= \frac{4\eta}{n + 1^2}
\end{align*}
\]

- When perpendicular to the normal (Grazing Incidence)
  - \(kr = 1\)
  - \(kt = 0\)
Total Internal Reflection

- Fresnel Equations Not Valid Here
  - $k_r = 1$
  - $k_t = 0$

![Graph showing total internal reflection](image-url)
- Radiance Attenuation

\[ \frac{dL}{L} = -\sigma dx \]

\[ L(d) = L_0 e^{-\sigma d} \]

\[ L(d) = c_f^d L_0 \]

- Cf = Color Filter

- No Color Filter -> RGB = (1,1,1)
- Green Tint Color Filter -> (.9,1,.9)
  - note - .9 depends on your coordinate system
Color Filtered Spheres
- Implement DielectricTransparentMaterial
- Heirarchy
  - IMaterial
  - Abstract Material
    - AbstractRetransmittedMaterial
      - AbstractReflectiveMaterial
        - PerfectSpecular
        - GlossySpecular
    - AbstractTransparentMaterial
      - SimpleTransparentMaterial
      - DielectricTransparentMaterial
  - Phong
  - Matte
- **Memory**
  - $n_{in}$

- **Function**
  - **Hit**
    - Obtain - Ray->$n_{out}$
    - Compute Fresnel Reflectance and Transmission
    - Compute Reflected Ray and Transmitted Ray (ch27)
    - Compute Color
      - Trace the Transimttance
        - Set Ray->$n_{out}$ to $n_{in}$
        - Return ColorTraced * Color Filter$^\text{distance to hit point}$
      - Trace the Reflectance
        - Normal Perfect Specular Trace
      - Combine the Colors with $kd$ and $kf$
Triangle Meshes
Boxes and Glass Pane
(Skipping Most Theory)
Glass of Water (or Beer)

- Cannot model separately
  - Not with his framework
- Why?
  - Need a perfect transition to Glass and Water
    - to give η
- Model as a single compound object
  - Cylinders and Disk
  - Cylinders and Part Tori
- Fishbowls modeled the Same Way
Glass of Water
- Algorithm
- Photons
- Photon Emission
- Photon Tracing
- Storage and Retrieval
2-Pass Algorithm

1. Send the Photons out from the Lights
2. Trace the scene, gather photons to compute radiance flux at each point
What is a Photon?

- **Position**
  - not tied to an object
- **Color**
  - Color of the photon (usually white)
- **Angle**
  - direction the photon was traveling

```c
struct photon {
  float x, y, z;   // position
  char p[4];       // color packed as 4 chars
  char phi, theta; // compressed incidence dir
  short flag;      // flag used by kd tree

  // (given by Jensen)
};
```
Photon Emission

- Point Lights
  - Pick a Random Direction
    - Not as easy as picking a (rho, theta)

- Area Lights
  - Pick a Random Point
  - Pick a Random Direction (Cosine Distribution)

- Projection Maps
  - Yes or No Projections
  - Specific Projections
Photon Tracing

- Same as Ray Tracing
  - (and global illumination)

- Bouncing
  - If ray hits a non-reflective object it can bounce
    - in a direction given by the objects BRDF
    - with a probability \( k \) given by the BRDF
      - the power of the reflectance
      - this makes all the photons have the same power
Photon Storage and Retrieval (KD Tree)

- **Construction**
  - After all the photons have been traced
  - **Algorithm**
    - Take a midpoint of all the photons in a direction
    - Split the tree at this point
    - Recursively iterate until some depth
      - depends on how many photons you want per bucket

- **Retrieval**
  - **Algorithm**
    - Get the bin with the given hit point
    - if all the n closest are in the bin
      - return those
    - Get all the bins around it (8)
      - return the n closest from all 27
Radiance Estimate

- **Collection**
  - Sphere – N Closest within a sphere
  - Disk – N Closest within a disk
    - (with a given normal)

- **Filtering**
  - Cone
    - \( \frac{d}{kr} \)
      - \( d = \) distance
      - \( r = \) maximum distance

- **Gaussian**
  - <<enter really complex Gaussian formula here>>
- Color Filtering
- Ray Marching
  - Adaptive
    - $dx = \frac{\log(\text{rand}())}{\sigma(\text{density}(x))}$
- Volume Photon Map
- Exclude Direct Illumination
- Ray Tracing From the Ground Up
  - Kevin Suffern
- Realistic Image Synthesis Using Photon Mapping
  - Henrik Wann Jensen