CMSC427 fall 2017
Global illumination – intro
Ray tracing and radiosity
So far – local illumination

- One triangle, one light at a time
- Object rendering *into* image – what pixel sees me?
- Opaque surfaces
- No shadows, no crosstalk between facets
Now—global illumination +

- Scene oriented – consider all triangles and lights
- Image oriented rendering – what object can pixel see?
- Translucent and transparent surfaces, refraction
- Shadows, color bleeding
Ray Tracing and Radiosity

- General concepts
  - What advantages do they have?
  - How can you spot a ray-traced or radiosityed image?
- How they work
  - Ray tracing overview
  - Radiosity overview
- Other approaches: path tracing, ray marching
Ray tracing

- Reflections, refractions
- Sharp shadows
- Partial physical model
- Point lights
- Tracing single rays
Radiosity

- Soft shadows
- Better physics
- Extended lights
- Integrating over extended areas
Ray tracing principles
The Rays

- Concepts:
  - View (primary) ray
  - Secondary rays
    - Shadow ray (to all lights)
  - Reflection ray
  - Refraction ray
The rays again
Recursive rays (rinse, repeat)
Refraction

- Depends on ratio of speed of light between two materials
Tweaking ray tracing

- One is not enough
  - One primary ray can “barely” miss an object
- Stochastic or random ray tracing
  - multiple, random primary rays out of a pixel
- Speeding it up
  - Speed up intersection calculations by data structure
Radiosity principles

- Global vs. local computations
The Cornell box
Radiosity vs. ray tracing

- Ray tracing - sample rays for each light, reflection and refraction
- Radiosity - integration over all rays on patch
- Iterate until light solutions are stable
Patch computations
Ray tracing details

- The basic recursive algorithm
- Casting the primary ray
- Intersecting with an object: sphere (circle), triangle
- Computing refraction ray
Ray tracing algorithm

```java
for (int j = 0; j < imageHeight; ++j) {
    for (int i = 0; i < imageWidth; ++i) {
        Ray primaryRay = new Ray(i, j);
        color[i][j] = rayTrace(primaryRay, 0);
    }
}

rayTrace(ray, generation)
if (generation > maxGen) return backgrdColor;
hitPt = intersect(ray, objectList);
if (hitPt == null) return backgroundColor;
c = accumulateLights(hitPt);
if (reflective(hitPt)) {
    reflectRay = reflect(ray, hitPt)
    c += trace(reflectRay) 
} 
if (refractive(hitPt)) {
    refractRay = refract(ray, hitPt)
    c += trace(refractRay) 
}
return c
```
Casting first ray

\[ p(t) = \text{eye} + t(\text{imagePt} - \text{eye}) \quad \text{with } t \in [0, \infty) \]

imagePt is \((x', y', -d)\), eye is at origin
Intersecting with first object

For each object:
- Compute hit time $t$ when ray hits object
- Find object with smallest $t$ – that is hit point

Spheres are easiest!
Refraction

http://en.wikipedia.org/wiki/Refraction

• Light rays that travel from one medium to another are bent

• To the viewer, object at location $x$ appears to be at location $y$
Index of refraction

http://en.wikipedia.org/wiki/Refractive_index

- Speed of light depends on medium
  - Speed of light in vacuum $c$
  - Speed of light in medium $v$
- Index of refraction $n = \frac{c}{v}$
  - Air 1.00029
  - Water 1.33
  - Acrylic glass 1.49
- “Change in phase velocity leads to bending of light rays”
Snell’s law


- Ratio of sines of angle of incidence $\theta_1$ and refraction $\theta_2$ is equal to opposite ratio of indices of refraction $n_1, n_2$

$$\frac{\sin \theta_1}{\sin \theta_2} = \frac{n_2}{n_1}$$

- Vector form in 3D

$$\mathbf{r} = \frac{n_1}{n_2} \mathbf{v} + \left( \frac{n_1}{n_2} \cos \theta_1 + \cos \theta_2 \right) \mathbf{n}$$

- Viewing, refracted direction $\mathbf{v}, \mathbf{r}$ (P,Q)

- Normal vector $\mathbf{n}$
Total internal reflection

http://en.wikipedia.org/wiki/Total_internal_reflection

- Angle of refracted ray

\[ \theta_2 = \arcsin \left( \theta_1 \frac{n_1}{n_2} \right) \]

- Critical angle

\[ \theta_c = \frac{n_2}{n_1} \]

- If \( \theta_1 = \theta_c \) we get \( \theta_2 = \pi/2 \) refracted ray is parallel to interface

- If \( \theta_1 > \theta_c \) we have total internal reflection (light ray does not cross interface between media)
Fresnel equations

- Fresnel equations are relatively complex to evaluate
- In graphics, often use Schlick’s approximation
- Ratio $F$ between reflected and refracted light
- Indices of refraction $n_1, n_2$

\[
F = f + (1 - f)(1 - \mathbf{v} \cdot \mathbf{n})^5
\]

\[
f = \frac{(1.0 - \frac{n_1}{n_2})^2}{(1.0 + \frac{n_1}{n_2})^2}
\]
Newer algorithms

- Path tracing
  - A multisampled, randomized version of ray tracing to approx. radiosity
- Ray marching
  - A “binary search” approach to finding the hit point for complex shapes when no closed form exists
This image is ...?