Projections

- **axis-aligned orthographic**
- **orthographic**
- **perspective**
- **oblique**
Computing Viewing Rays

Parallel projection
same direction, different origins

Perspective projection
same origin, different directions
Ray-Triangle Intersection

```c
boolean raytri (ray r, vector p0, p1, p2, interval [t_0,t_1])
{
    compute t
    if (( t < t_0 ) or (t > t_1))
        return ( false )
    compute γ
    if ( (γ < 0 ) or (γ > 1))
        return ( false )
    compute β
    if ( ((β < 0 ) or (β+γ > 1))
        return ( false )
    return true
}
```
Point in Polygon?

• Is P in polygon?
• Cast ray from P to infinity
  – 1 crossing = inside
  – 0, 2 crossings = outside
Point in Polygon?

- Is P in concave polygon?
- Cast ray from P to infinity
  - Odd crossings = inside
  - Even crossings = outside
What Happens?
Raytracing Characteristics

• Good
  – Simple to implement
  – Minimal memory required
  – Easy to extend

• Bad
  – Aliasing
  – Computationally intensive
    • Intersections expensive (75-90% of rendering time)
    • Lots of rays
Basic Illumination Concepts

- **Terms**
  - Illumination: calculating light intensity at a point (object space; equation) based loosely on physical laws
  - Shading: algorithm for calculating intensities at pixels (image space; algorithm)

- **Objects**
  - Light sources: light-emitting
  - Other objects: light-reflecting

- **Light sources**
  - Point (special case: at infinity)
  - Area
A Simple Model

• Approximate BRDF as sum of
  – A diffuse component
  – A specular component
  – A “ambient” term
Diffuse Component

• Lambert’s Law
  – Intensity of reflected light proportional to cosine of angle between surface and incoming light direction
  – Applies to “diffuse,” “Lambertian,” or “matte” surfaces
  – Independent of viewing angle
• Use as a component of non-Lambertian surfaces
Diffuse Component

\[ k_d I (\hat{l} \cdot \hat{n}) \]

\[ \max (k_d I (\hat{l} \cdot \hat{n}), 0) \]
Diffuse Component

• Plot light leaving in a given direction:

• Plot light leaving from each point on surface
Specular Component

- Specular component is a mirror-like reflection
- Phong Illumination Model
  - A reasonable approximation for some surfaces
  - Fairly cheap to compute
- Depends on view direction
Specular Component

\[ k_s I (\hat{r} \cdot \hat{v})^p \]

\[ k_s I \max(\hat{r} \cdot \hat{v}, 0)^p \]
Specular Component

• Computing the reflected direction

\[ \hat{r} = -\hat{l} + 2(\hat{l} \cdot \hat{n})\hat{n} \]

\[ \hat{h} = \frac{\hat{l} + \hat{v}}{\|\hat{l} + \hat{v}\|} \]
Specular Component

• Plot light leaving in a given direction:

• Plot light leaving from each point on surface
Specular Component

• Specular exponent sometimes called “roughness”
Ambient Term

• Really, it's a cheap hack
• Accounts for “ambient, omnidirectional light”
• Without it everything looks like it’s in space
Summing the Parts

\[ R = k_a I + k_d I \max(\hat{l} \cdot \hat{n}, 0) + k_s I \max(\hat{r} \cdot \hat{v}, 0)^p \]

- Recall that the \( k? \) are by wavelength
  - RGB in practice
- Sum over all lights
Shadows

• What if there is an object between the surface and light?
Ray Traced Shadows

• Trace a ray
  – Start = point on surface
  – End = light source
  – t=0 at Surface, t=1 at Light
  – “Bias” to avoid surface acne

• Test
  – Bias ≤ t ≤ 1 = shadow
  – t < Bias or t > 1 = use this light
Mirror Reflection
Ray Tracing Reflection

- Viewer looking in direction $d$ sees whatever the viewer “below” the surface sees looking in direction $r$

- In the real world
  - Energy loss on the bounce
  - Loss different for different colors

- New ray
  - Start on surface, in reflection direction
Ray Traced Reflection

• Avoid looping forever
  – Stop after $n$ bounces
  – Stop when contribution to pixel gets too small
Specular vs. Mirror Reflection
Combined Specular & Mirror

- Many surfaces have both
Refraction
Front
Calculating Refraction Vector

- **Snell’s Law**
  \[ n_v \sin \theta_v = n_t \sin \theta_t \]

- **In terms of** \( \theta_t \)
  \[ \hat{t} = \hat{m} \sin \theta_t - \hat{n} \cos \theta_t \]

- **\( \hat{m} \) term**
  \[
  \hat{m} = \frac{(\hat{n}(\hat{n} \cdot \hat{v}) - \hat{v})}{\sin \theta_v}
  \]
  \[
  \hat{m} \sin \theta_t
  = (\hat{n}(\hat{n} \cdot \hat{v}) - \hat{v}) \sin \theta_t / \sin
  = (\hat{n}(\hat{n} \cdot \hat{v}) - \hat{v}) n_v / n_t
  \]
Calculating Refraction Vector

• **Snell’s Law**
  \[ n_v \sin \theta_v = n_t \sin \theta_t \]

• **In terms of** \( \theta_t \)
  \[ \hat{t} = \hat{m} \sin \theta_t - \hat{n} \cos \theta_t \]

• **\( \hat{n} \) term**
  \[ -\hat{n} \cos \theta_t \]
  \[ = -\hat{n} \sqrt{1 - \sin^2 \theta_t} \]
  \[ = -\hat{n} \sqrt{1 - \sin^2 \theta_v \frac{n_v^2}{n_t^2}} \]
  \[ = -\hat{n} \sqrt{1 - (1 - \cos^2 \theta_v) \frac{n_v^2}{n_t^2}} \]
  \[ = -\hat{n} \sqrt{1 - (1 - (\hat{n} \cdot \hat{v})^2) \frac{n_v^2}{n_t^2}} \]
Calculating Refraction Vector

• Snell’s Law
  \[ n_v \sin \theta_v = n_t \sin \theta_t \]

• In terms of \( \theta_t \)
  \[ \hat{t} = \hat{m} \sin \theta_t - \hat{n} \cos \theta_t \]

• In terms of \( \hat{n} \) and \( \hat{v} \)
  \[ \hat{t} = (\hat{n}(\hat{n} \cdot \hat{v}) - \hat{v})n_v/n_t \]
  \[ -\hat{n} \sqrt{1 - (1 - (\hat{n} \cdot \hat{v})^2) n_v^2/n_t^2} \]
Alpha Blending

• How much makes it through
• $\alpha =$ opacity
  – How much of foreground color 0-1
• $1-\alpha =$ transparency
  – How much of background color
• Foreground$^*\alpha +$ Background$^*(1-\alpha)$
Refraction and Alpha

• Refraction = what direction
• $\alpha = \text{how much}$
  – Often approximate as a constant
  – Better: Use Fresnel

$$F = \frac{1}{2} \left( \frac{n_v \hat{n} \cdot \hat{r} + n_t \hat{n} \cdot \hat{t}}{n_v \hat{n} \cdot \hat{r} - n_t \hat{n} \cdot \hat{t}} \right)^2 + \frac{1}{2} \left( \frac{n_v \hat{n} \cdot \hat{t} + n_t \hat{n} \cdot \hat{r}}{n_v \hat{n} \cdot \hat{t} - n_t \hat{n} \cdot \hat{r}} \right)^2$$

– Schlick approximation

$$F_0 = \frac{(n_v - n_t)^2}{(n_v + n_t)^2}$$

$$F \approx F_0 + (1 - F_0)(1 - \hat{n} \cdot \hat{v})^5$$
Full Ray-Tracing

• For each pixel
  – Compute ray direction
  – Find closest surface
  – For each light
    • Shoot shadow ray
    • If not shadowed, add direct illumination
  – Shoot ray in reflection direction
  – Shoot ray in refraction direction
Dielectric

if (p is on a dielectric) then
    r = reflect (d, n)
if (d.n < 0) then
    refract (d, n, n, t)
    c = -d.n
    kr = kg = kb = 1
else
    kr = exp(-alpher * t)
    kg = exp(-alphag * t)
    kb = exp(-alphab * t)
    if (refract(d, -n, 1/n t) then
        c = t.n
    else
        return k * color(p+t*r)
R0 = (n-1)^2 / (n+1)^2
R = R0 + (1-R0)(1 - c)^5
return k(R color(p + t*r) + (1-R)color(p+t*t)
Distribution Ray Tracing
Distribution Ray Tracing

• Anti-aliasing
• Soft Shadows
• Depth of Field
• Glossy Reflection
• Motion Blur

• Turns Aliasing into Noise
Sampling
Soft Shadows
Depth of Field

Soler et al., Fourier Depth of Field, ACM TOG v28n2, April 2009
Pinhole Lens
Lens Model
Lens Model
Ray Traced DOF

- Move image plane out to focal plane
- Jitter start position within lens aperture
  - Smaller aperture = closer to pinhole
  - Larger aperture = more DOF blur
Glossy Reflection
Motion Blur

- Things move while the shutter is open
Ray Traced Motion Blur

• Include information on object motion
• Spread multiple rays per pixel across time