color

for each channel we'll approximate the color at the intersection point as the sum of five terms
- emission
- ambient reflection
- diffuse reflection
- specular reflection
- specular transmission

first a recap

light from source hits surface

some is absorbed and released as heat

first a recap

light from source hits surface

some is diffusely reflected uniformly in all directions

first a recap

light from source hits surface

some is specularly reflected at or near the angle of reflection

first a recap

light from source hits surface

some is transmitted through surface
transmission

light from source hits surface
some is transmitted through surface

what do we see?

specular transmission

light from source hits surface
reflected light hits surface and some is transmitted through surface

specular reflection

light from source hits surface
reflected light hits surface and some is specularly reflected

specular reflection

light from source hits surface
reflected light hits surface and some is diffusely reflected but diffusely reflected light occurs from all (non-occluded) directions

We do not consider inter-object diffuse reflections

specular reflection & transmission

light from source hits surface
We will recursively compute specular reflection and transmission
**color: simple ray casting**

For each channel, we'll approximate the color at the intersection point as the sum of five terms:
- emission
- ambient reflection
- diffuse reflection
- specular reflection
- specular transmission (no contribution)

---

**ray tracing**

- simple ray casting
- recursive ray tracing
- cheap tricks
- optimizations

---

**global effects**

- shadows
- specular reflection
- specular transmission

---

**ray tracing**

- cast ray through pixel into scene
- find closest intersection (if any)
- compute luminance at intersection
  - direct illumination (no occlusions)
  - indirect reflection

---

**color: ray tracing**

For each channel, we'll approximate the color at the intersection point as the sum of five terms:
- emission
- ambient reflection
- diffuse reflection (check for shadows)
- specular reflection (check for shadows)
- recursive term

---

**recursive rays**

- cast ray
- specular reflection
- shadow
- specular transmission
occlusion (shadows)

Light \( L \) is occluded if the ray \( R' = (P, -\mathbf{n}) \) intersects some object in the scene.

\[ R' \]

occlusion: implementation

Light \( L \) is occluded if the ray \( (P, -\mathbf{n}) \) intersects some object in the scene.

\[ \text{offset } R' \text{ slightly so it doesn't intersect at } P \]

reflections

- Cast ray reflected at \( P \) into scene
- Find closest intersection point \( P' \) (if any)
- Compute color at \( P' \)
- Scale by \((1-k_{\text{trans}}) \) \( \times \) msr\([g, b]\) and add to color at \( P \)

transmission

- Cast ray transmitted at \( P \) into scene
- Find closest intersection point \( P' \) (if any)
- Compute color at \( P' \)
- Scale by \( k_{\text{trans}} \) \( \times \) msr\([g, b]\) and add to color at \( P \)

refraction - Snell's law

\[ \text{incoming ray (} P_0, \mathbf{v} \text{)} \]

\[ \text{transmitted ray (} P, \mathbf{v}' \text{)} \]

thin surface refraction

\[ \text{ignore Snell's law} \]

\[ \text{incoming ray (} P_0, \mathbf{v} \text{)} \]

\[ \text{transmitted ray (} P, \mathbf{v} \text{)} \]
thick surface refraction

 incoming ray \( (P_0, v_{in}) \)

\[ \theta_{in} \]

refractive index \( n_{in} \)

\[ \theta_{in} \] satisfies:

\[ n_{in} \sin \theta_{in} = n_{out} \sin \theta_{out} \]

\[ \sin \theta_{out} = \beta \sin \theta_{in} \]

where \( \beta = \frac{n_{in}}{n_{out}} \)

provided \( 0 \leq \beta \sin \theta_{in} \leq 1 \)

What if \( \beta \sin \theta_{in} < 1 \)?

\[ \theta_{out} > 90^\circ : \text{no transmission} \]

\[ \theta_{out} \]

refractive index \( n_{out} \)

\[ \theta_{out} \]

transmitted ray \( (P, v_{out}) \)

\[ v_{out} = \cos \theta_{out} \times (n) + \sin \theta_{out} \times s \]

\[ \sin \theta_{out} = \beta \sin \theta_{in} \]

\[ \cos \theta_{out} = (1-\sin^2 \theta_{out})^{1/2} = (1-\beta^2 \sin^2 \theta_{in})^{1/2} \]

\[ s = v_{in} - (n \times n) \times n = v_{in} + \cos(\theta_{in}) \times n \]

thick surface refraction

\[ \theta_{in} \]

refractive index \( n_{in} \)

\[ \theta_{in} \]

transmitted ray \( (P, v_{out}) \)

\[ v_{out} = \beta \cos \theta_{in} - (1-\beta^2 \sin^2 \theta_{in})^{1/2} \times n + \beta \times v_{in} \]

thick surface refraction

\[ \theta_{in} \]

refractive index \( n_{in} \)

\[ \theta_{in} \]

transmitted ray \( (P, v_{out}) \)

\[ v_{out} = \beta \cos \theta_{in} - (1-\beta^2 \sin^2 \theta_{in})^{1/2} \times n + \beta \times v_{in} \]

stopping conditions

recurse until:

a) maximum recursive depth specified by user is reached

b) contribution to color is less than user specified bound

- cast new ray from \( P \) into scene
- find closest intersection point \( P' \) (if any)
- compute color at \( P' \)
- scale and add to color at \( P \)

implementation issues

offset new ray slightly to make sure you don't find \( P \) again!!

- cast new ray from \( P \) into scene
- find closest intersection point \( P' \) (if any)
- compute color at \( P' \)
- scale and add to color at \( P \)
Ray Tracing

- Simple ray casting
- Recursive ray tracing
- Cheap tricks
- Optimizations

Cheap Tricks

- Texture mapping
- Procedural texture mapping
- Bump mapping
- Transparency mapping
- Depth of field
- Lens effects
- Jittering
- Soft shadows

Texture Mapping

- "Glue" image to surface

Texture Mapping

\[ f(p) : \text{coordinates in the texture map corresponding to surface point } p \]
texture mapping triangle
Input specifies texture coordinates of triangle vertices

triangle: parametric form
A point p on the triangle T can be uniquely represented as
\( q = t_0 + \beta u + \gamma w \) where \( \beta \geq 0, \gamma \geq 0, \beta + \gamma \leq 1 \)

computing \( f(p) \)

compute \( f(u) = f(t_1) - f(t_0) \) and \( f(w) = f(t_2) - f(t_0) \)

computing \( f(p) \)

compute \( f(p) = f(t_0) + \beta f(u) + \gamma f(w) \)

What is \( f(p) \)?

Need to resample! For your ray tracer use bilinear interpolation.
using the color

- Surface
- Texture (digital image)

r,g,b of texture at f(p):
1. Use as pixel color
2. Use as diffuse and specular coefficient of surface at p
3. Use as diffuse coefficient of surface at p

DO THIS!

texture mapping sphere

- Example: Latitude/Longitude

parameterization

A point p on the surface can be represented relative to c, r, t, u

\[ p - c = \alpha r + \beta t + \gamma u \]

where
\[ \alpha = \cos(\theta_p) \cos(\phi_p) \]
\[ \beta = \cos(\theta_p) \sin(\phi_p) \]
\[ \gamma = \sin(\theta_p) \]

\( \theta_p, \phi_p \)

parameterization

\( f(p) = \left( \frac{w_r \phi_p}{360}, \frac{h_t \theta_p}{360} \right) \) where the angles are in degrees and \( w, h \) are the image width and height in pixels

texture mapping sphere
as before

• resample image
• use color as ...

problems

• given p compute $\phi$, $\theta$
  you can do that!
• poles
  test for pole and use default texture coordinate
• seams
  • use good textures
  • overlap & blend or mix
  • don’t look there
  • 3d textures

3d textures

use stack of images
how do we generate these images?

cheap tricks

• texture mapping
• procedural texture mapping
• bump mapping
• transparency mapping
• depth of field
• lens effects
• jittering
• soft shadows

procedural texture mapping

• procedure returns a texture color for any point in 3d space (note this is not an image stack)

• sample to find texture for surface
procedural textures

• advantages
  - don’t need to find a mapping from a (complex) 3d surface to a 2d texture image
  - concise representation of texture
• disadvantages
  - ad hoc techniques cannot duplicate photographs

perlin noise – 1D example

step 1: generate discrete noise function with specified length, amplitude, sampling frequency

example: length=8, amplitude = 3, sampling frequency is 7 Hz

step 2: interpolate with smoothing

step 3: repeat with various amplitudes/frequencies

step 4: add together

perlin noise – 2D example

for more info see Perlin Noise link on proj2 web site

cheap tricks

• texture mapping
• procedural texture mapping
• bump mapping
• transparency mapping
• depth of field
• lens effects
• jittering
• soft shadows