Computer Graphics

Lecture 8

Diffuse Shading
Mirror Reflection
Shadows
Announcements

• Some A1 stats:

  • Time since release: one week
  • Time to deadline: 5 days
  • Office hours remaining before deadline: 2
  • Questions in office hours so far: 1
  • Groups that hadn’t created their github repo as of this morning: 8
A0 Artifact results are in...

- A 2-way tie for first place!
First Place

Alex Ayala!
Also First Place

Raiden Van Bronkhorst!
Ray Tracing: Pseudocode

for each pixel:
  → generate a viewing ray for the pixel
  → find the closest object it intersects
  determine the color of the object
Ray Tracing: Pseudocode

for each pixel:
    generate a viewing ray for the pixel
    find the closest object it intersects
    determine the color of the object
Shading

What does the color of a pixel depend on?

- Material
- Color of object
- Light
- Intensity of light
direction of light from surface
occlusions between surface and light?
surface normal
transparency/translucency of stuff shining thru?
Shading

What does the color of a pixel depend on?

Hint: think beyond matte gray spheres.

Shading

What does the color of a pixel depend on?

- surface normal
- surface properties (color, shininess, ...)
- eye direction
- light direction (for each light)
Shading

What does the color of a pixel depend on?

- surface normal  \textit{stored in or calculated from object}
- surface properties (color, shininess, ...)  \textit{stored in object}
- eye direction  \textit{calculated from viewing ray and intersection point}
- light direction (for each light)  \textit{calculated from light and intersection point}
function ray_intersect(ray, sphere, tmin, tmax):
    • Use last lecture's math to find +/- t
    • If no real solutions, return nothing
    • Otherwise, return closest t that lies between tmin and tmax
    • Also return info needed for shading - store in a HitRecord struct.

In A2: t, intersection point, normal, texture coordinate, object
Light Sources

- Where does light come from?
- Two simple kinds of sources:
  - point source: defined by a 3D position
  - directional source: defined by a 3D direction vector
  - ...many other possibilities!
Point and Directional Lights
Problems 1-2: Eye Direction and Light Direction

Given a viewing ray \((p + td)\) and the \(t\) at which it intersects a surface, find a unit vector giving the direction from the surface towards the viewer.
Light Sources: Exercise

Given a ray \((\mathbf{p} + td)\) and the \(t\) at which it intersects a surface, calculate a unit vector giving the direction from the surface towards:

- a point light source at position \(\mathbf{S}\)
- a directional light source with direction \(\ell\)
Diffuse (Lambertian) Reflection

- On a *diffuse* surface, light scatters uniformly in all directions.
- No dependence on view direction.
- Many surfaces are approximately diffuse:
  - matte painted surfaces, projector screens,
  - anything that doesn't look "shiny"
Diffuse (Lambertian) Reflection

\[ L_d \propto \cos \theta = \hat{n} \cdot \hat{l} \]
Diffuse (Lambertian) Reflection

The top face of a cube receives some amount of light.

Rotated 60°, the same face receives half the light.

Light per unit area is proportional to \[ \cos \theta = \vec{n} \cdot \vec{l} \]

Highly recommended reading: https://ciechanow.ski/lights-and-shadows/
Diffuse (Lambertian) Shading

- The full model:

\[ L_d = \begin{bmatrix} k_d \end{bmatrix} I \max(0, \n \cdot \ell) \]

- why max?

\[ L_d = \begin{bmatrix} k_d \end{bmatrix} I \max(0, \n \cdot \ell) \]
Diffuse (Lambertian) Shading

\[ L_d = k_d I \max(\vec{n} \cdot \vec{l}) \]

For colored objects, \( k_d \) is a 3-vector of R, G, and B reflectances.
Let's talk shinies.
Let's talk shinies.

How does a mirror interact with light?
Problems 3-4: Diffuse and Mirror Reflection
Mirror Reflection

What does a camera see when it looks at a mirror?

Can we do this using the tools we already have?

(Problem 4)
Mirror Reflection

What does a camera see when it looks at a mirror?

Can we do this using the tools we already have?

\texttt{find\_intersection(ray, scene)}
Mirror Reflection

What does a camera see when it looks at a mirror?

compute \( r \)
ray = Ray(\( x \), \( r \))
find_intersection(ray, scene)
Recursion!?

traceray(ray, scene):
    t, rec = find_intersection(ray, scene)
    if rec.obj is a mirror:
        compute \( r \), the reflection direction
        mirror_ray = Ray(rec.x, r)
        return traceray(mirror_ray, scene)
    # other cases, ...