CS 112 - Ray Tracing
Illumination is not accurate

Cannot capture the effects of refraction, transparency and translucency accurately.
Direct and Indirect Illumination

- Capture only direct illumination
  - Light coming directly from light
- Light bounces from other objects in the scene
Ray Tracing

- Start from the light and find how each ray is getting reflected to from different objects to reach the viewer
  - Exponentially complex problem
- Reverse operation
  - Start from the viewer and see how a particular ray has traveled
Ray Tracing

- Casts one ray per pixel
- Casts a bunch of rays in the scene
- Find out how the ray traverses
Recursive Ray Tracing

- Ray hits an object at P
  - Cast a shadow ray S from P to each light
  - If shadow ray does not intersect any other object, calculate direct illumination from light $I_L$
  - Cast a reflected ray R from P and find its contribution, $I_R$
  - Cast a refracted ray T from P and find its contribution, $I_T$
  - $C = w_L I_L + w_R I_R + w_T I_T$
How to stop the recursion?

- If a ray has travelled beyond
  - A threshold distance
  - A threshold number of hops
  - Energy is the ray has fallen beyond a certain threshold
Intersections (Ray-Sphere)

- $|P-P_c|^2 - r^2 = 0$
- $P = P_0 + t(P_1-P_0)$
- $|P_0-P_c + t(P_1-P_0)|^2 - r^2 = 0$

Will give you a quadratic equation to solve for $t$
Ray-Triangle Intersection

- $V_0, V_1, V_2$ - Triangle
  - $V_0 + u(V_1-V_0) + v(V_2-V_0)$

- $P_0, P_1$ - Ray
  - $P_0 + t(P_1-P_0)$

Intersection point $I$, such that

- $V_0 + u(V_1-V_0) + v(V_2-V_0) = P_0 + t(P_1-P_0)$
- $u(V_1-V_0) + v(V_2-V_0) + t(P_0-P_1) = P_0-V_0$
- $uA + vB + tC = D$
### Ray Triangle Intersection

Given a ray defined by its direction vector $(u, v, t)$ and a triangle defined by its vertices $(A_x, A_y, A_z), (B_x, B_y, B_z), (C_x, C_y, C_z)$, the intersection point can be found using the following equation:

\[
\begin{bmatrix}
A_x & B_x & C_x \\
A_y & B_y & C_y \\
A_z & B_z & C_z \\
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
t \\
\end{bmatrix}
=
\begin{bmatrix}
D_x \\
D_y \\
D_z \\
\end{bmatrix}
\]

To solve for $(u, v, t)$, we can invert the matrix on the left:

\[
\begin{bmatrix}
u \\
v \\
t \\
\end{bmatrix} =
\begin{bmatrix}
A_x & B_x & C_x \\
A_y & B_y & C_y \\
A_z & B_z & C_z \\
\end{bmatrix}^{-1}
\begin{bmatrix}
D_x \\
D_y \\
D_z \\
\end{bmatrix}
\]
Antialiasing

- Shoot more than one ray through the pixels
  - Super-sampling
- Average their contribution
  - Filtering