Ray Tracing

CS116B

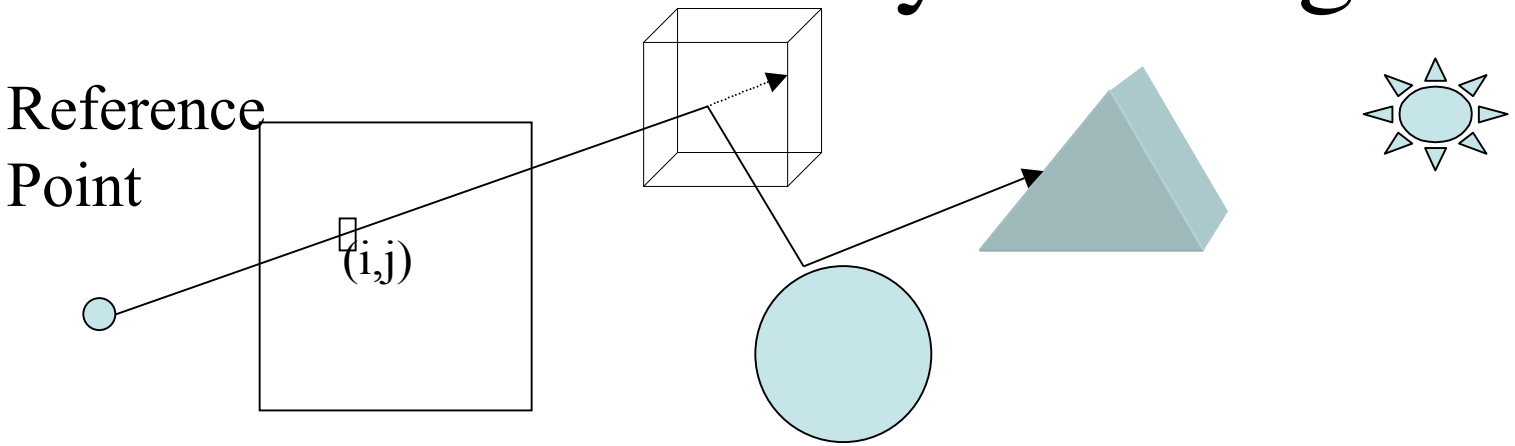
Chris Pollett

Apr 20, 2004.

Outline

- Basic Ray-Tracing
- Intersections

Basic Ray-Tracing



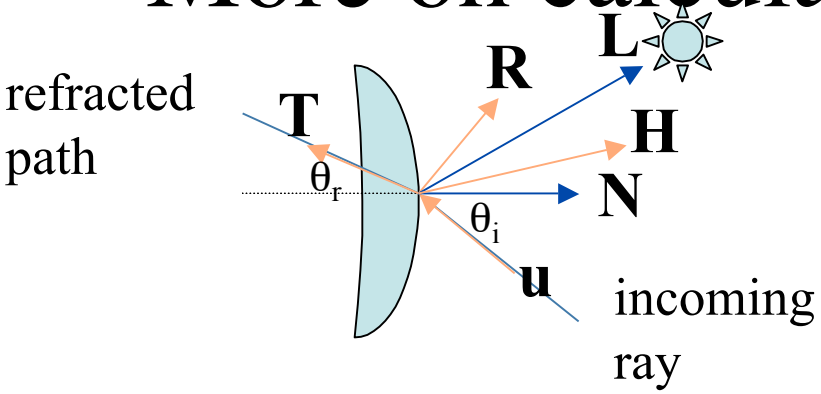
Projection Plane

- Have a set up like in the above picture.
- Shoot rays from a reference point through centers of each pixel of the projection plane.
- Find the nearest surface in scene hit by a given ray.
- Calculate how it would be lit in our lighting model
- Then calculate secondary reflection and refraction rays and see what surfaces they hit and add their value, appropriately attenuated, to the value that we just calculated.

Boundary Conditions

- We terminate paths through the scene if:
  - The ray intersects no surface.
  - The ray intersects a light source that is not a reflecting surface
  - The tree has reached some maximum depth.
- Let h be the height of our tree so far, at each surface point, we:
  - Check if a terminating condition has been reached.
  - Use the basic illumination model to calculate a surface intensity contribution (I_r, I_g, I_b).
  - Let (Rx,Ry,Rz) and (Tx, Ty, Tz) be the unit vectors for the reflected and transmitted rays.
  - Let (Xr, Yr, Zr) and (Xt,Yt,Zt) be the pixel locations of the surface that will be intersected by the reflected and transmitted light rays respectively.
  - Let dr and dt be the distances to these surfaces, and att(dr) and att(dt) the corresponding attenuations.
  - Calculate (Irr, Irg, Irb) = ray_trace(Rx,Ry,Rz, Xr, Yr, Zr, h-1) and (Itr, Itg, Itb) = ray_trace(Tx,Ty,Tz, Xt, Yt, Zt, h-1)
  - Return the value (I_r + att(dr) *Irr + att(dt) *Itr, I_g + att(dr) *Irg + att(dt) *Itg, I_b + att(dr) *Irb + att(dt) *Itb).

More on calculating with rays



- **T** -transmitted ray; **N** - surface normal; **u** unit vector for incoming ray = -**V**, the view vector; **R** - unit reflected ray; **L** - unit vector to light source; **H** -halfway vector between **L** and **V**.
- Ambient light is $k_a\mathbf{I}_a$; diffuse light is proportional to $k_d(\mathbf{N.L})$; and specular light is proportional to $k_s(\mathbf{H.L})^{n\_s}$.
- $\mathbf{R} = \mathbf{u} - (2\mathbf{u.N})\mathbf{N}$; $\mathbf{T} = (\eta_i/\eta_r)\mathbf{u} - (\cos\theta_r - (\eta_i/\eta_r)\cos\theta_i)\mathbf{N}$.
- η_i and η_r are the index of incidence and of refraction respectively and the formula for **T** comes from Snell's law.

More calculating with rays

- Any point on a ray can be calculated using the equation: $\mathbf{P} = \mathbf{P}_0 + s\mathbf{u}$.
- **u** depends on the pixel we are tracing through and can be calculated as $\mathbf{u} = \mathbf{P}_{pix} - \mathbf{P}_{prp} / (\mathbf{P}_{pix} - \mathbf{P}_{prp})$.
- For each surface we want to find the value for **P** where the ray intersects the surface (if it does). This involves solving a system of equations involving the ray equation and the surface equation. We'll talk more about this in the next several slides.
- Another issue is that their might be an obstructing surface between the light source and the surface point we are considering. We can send out a **shadow ray** from the surface point to the light to check for this. If we detect an opaque surface in between then we ignore that light source. (Can be lazy and not do for this homework.)

Ray-Sphere Intersections

- A sphere is given by the equation:

  $|\mathbf{P} - \mathbf{P}_c|^2 - r^2 = 0$
- Plugging in our ray equation we get:

  $|\mathbf{P}_0 - s\mathbf{u} - \mathbf{P}_c|^2 - r^2 = 0$
- Letting $\Delta\mathbf{P} = \mathbf{P}_0 - \mathbf{P}_c$, and expanding the magnitude as a square root of a dot product, we get:

  $s^2 - 2(\mathbf{u}.\Delta\mathbf{P})s + (\Delta\mathbf{P} - r^2) = 0$
- Solving this gives:

  $s = (\mathbf{u}.\Delta\mathbf{P}) +/- [(\mathbf{u}.\Delta\mathbf{P})^2 - |\Delta\mathbf{P}|^2 + r^2]^{1/2}$.
- If the equation under the square root is negative there is no intersection.

Ray-Polyhedron Intersections

- Usually embed polyhedron in a bounding sphere.
- First, check the intersection with the bounding sphere, if this fails ignore this polyhedron; otherwise, ...
- Identify front faces of the polyhedron. i.e., those satisfying: $\mathbf{u}.\mathbf{N} < 0$.
- For each such face, solve the plane equation: $\mathbf{N}.\mathbf{P} = -D$. That is, $\mathbf{N}.(\mathbf{P}_0 + s\mathbf{u}) = -D$. This gives

  $s = -(D + \mathbf{N}.\mathbf{P}_0)/\ \mathbf{u}.\mathbf{N}$.
- Finally, do an inside outside check to see if this point is inside or outside of the polygon on the polygons plane.

Reducing Object-Intersection Calculations

- Surface intersection calculations are the most time expensive component of a ray-tracer.
- We want to come up with ways to reduce the amount of time we spend on these calculations.
- One technique is to enclose groups of objects within a bounding object and test intersections with that bounding object.
- We can also create a hierarchy of such objects.