Pathways of light and colour

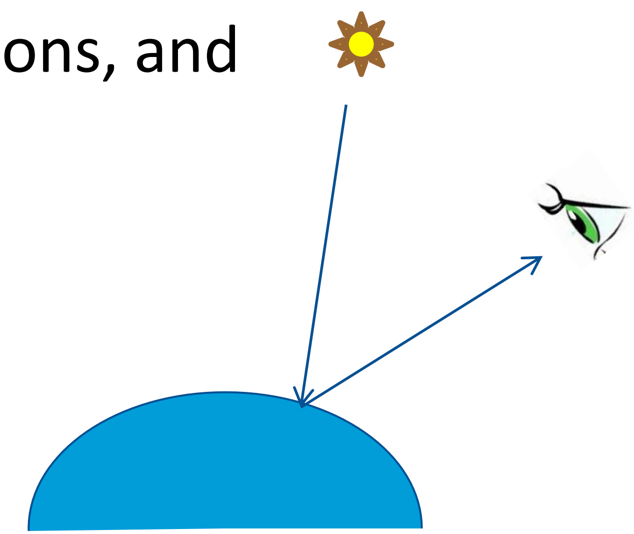
8 Ray Tracing

Department of Computer Science and Software Engineering
University of Canterbury, New Zealand.

R. Mukundan (mukundan@canterbury.ac.nz)

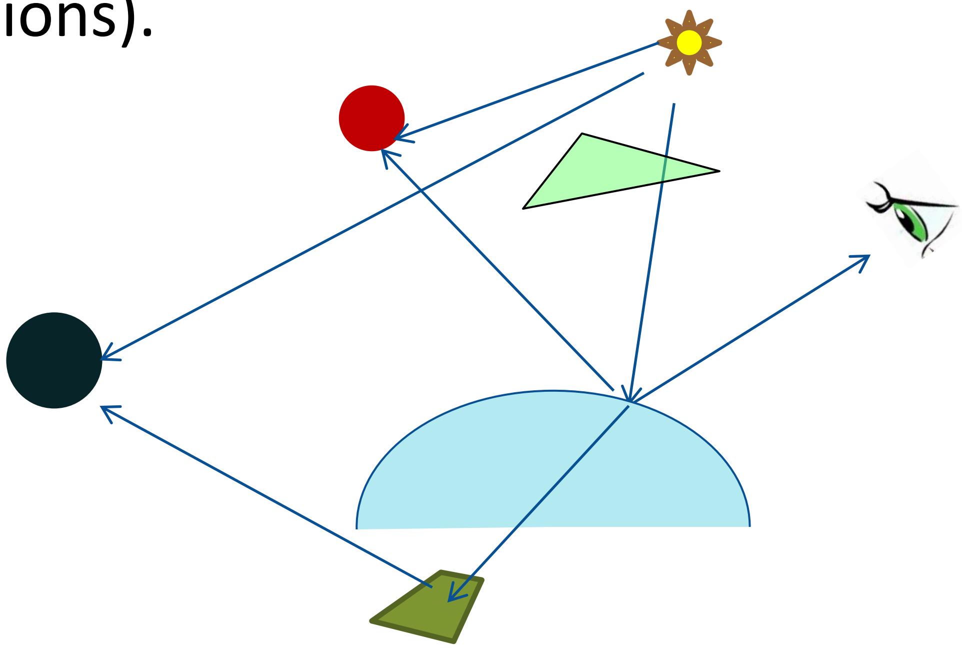
Local Illumination Model

- A local illumination model considers only light travelling from a light source to a surface and then reflected off the surface to the eye.
  - Requires only the light source coordinates, local surface geometry and the material characteristics at a vertex.
  - Suitable for the hardware pipeline (OpenGL, Direct3D)
- Does not consider occlusions, reflections, and transmittance of light.



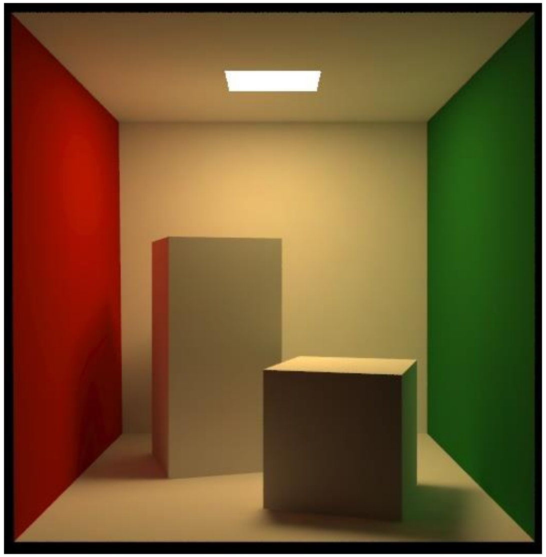
Global Illumination

- The illumination at a given point is a combination of the light received from a source and the light reflected from other surfaces in the scene.
- Considers the effects of occlusions (shadows) surface reflections, light transmission through a medium (direct transmittance and refractions).



Global Illumination Methods

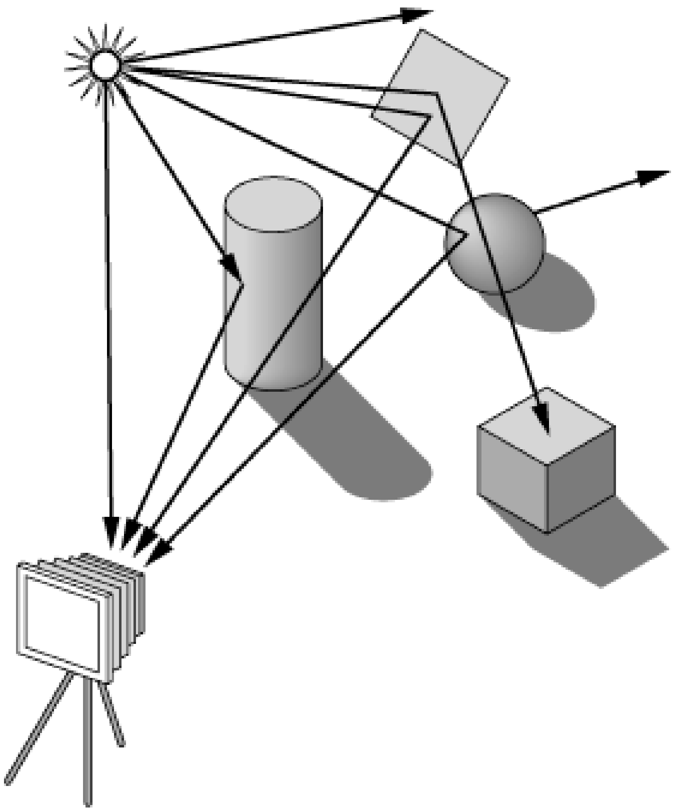
- Radiosity
  - A scene illumination can be considered as an equilibrium state for radiant energy transfers between surface elements.
  - Gives good results for diffuse illumination, but specularity is not handled.
    - Useful for modelling area light sources, colour bleeding, soft shadows.



'Forward' Ray Tracing

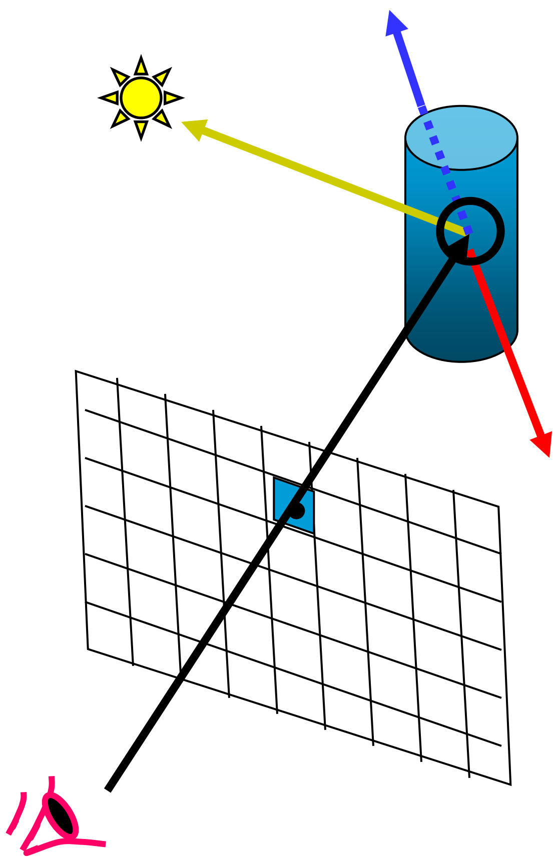
Trace rays from a light source.

- No specific limit for ray directions.
- Hardly any rays reach the eye.
- Computationally infeasible
- But a hybrid scheme called *photon mapping* can be used
  - Forward ray trace from light source
  - Record photon hits
  - Then, to render image:
    - Backward trace from eye
    - Get colour from photon hit density



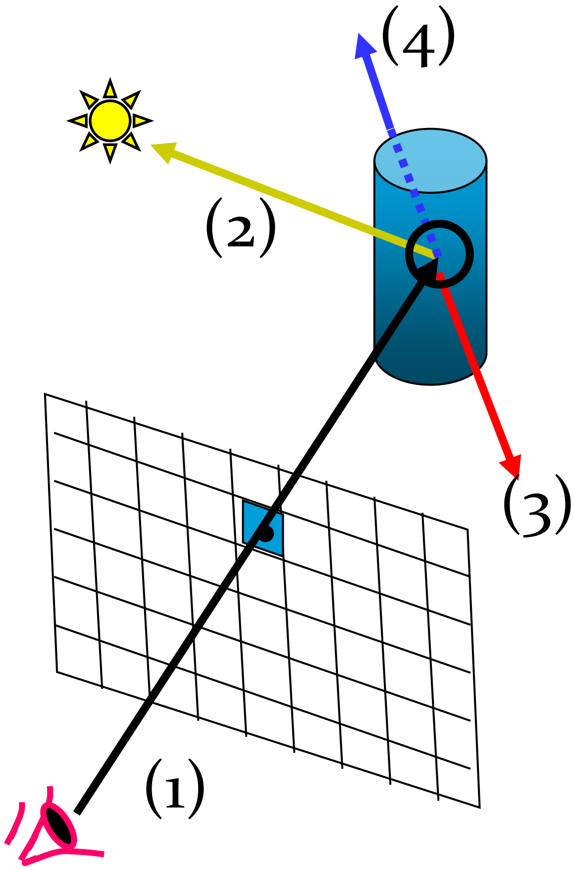
Ray Tracing (Backward Ray Tracing)

- Traces ray outwards from the eye to objects and light sources. The opposite of reality.
- Use secondary rays to determine shadows and to model mirror reflection and refraction
- Easy way to get many global illumination effects
- But:
  - Doesn’t handle diffuse inter-reflections
    - e.g. “colour bleed” from red wall to adjacent white wall
  - Computationally expensive



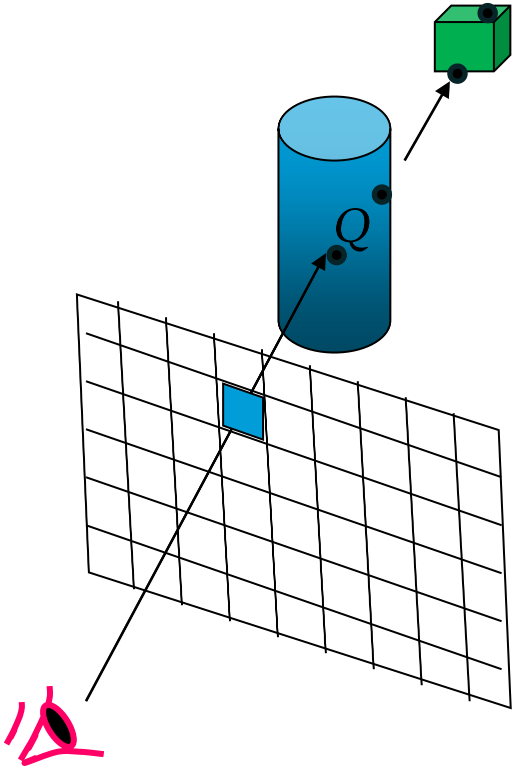
Ray Tracing: Basic Steps

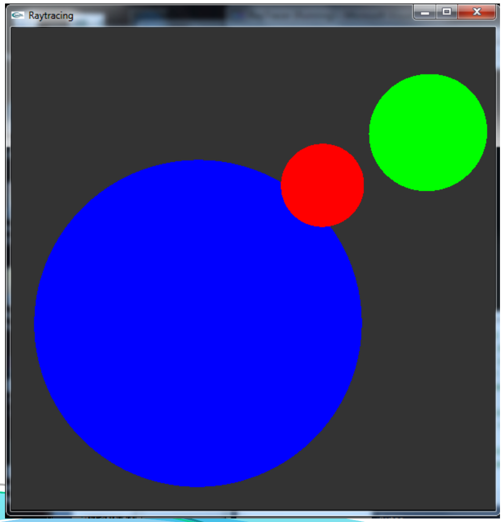
- Define a "virtual screen" and eye position
- Trace rays from eye through each "pixel" (1)
- Compute the closest point of intersection (Ray casting)
- Apply an illumination model at the point of intersection.
- Trace a shadow ray to determine if the point is in shadow (2)
- If the surface is reflective, trace a reflected ray (3)
- If the surface is refractive, trace a refraction ray (4)
- Add contributions from (3) and (4).



Ray Casting

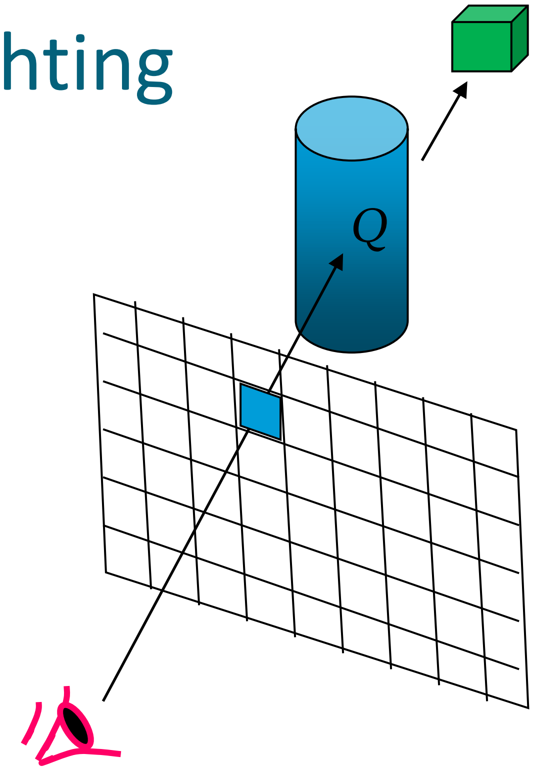
- Ray tracing without secondary rays.
- Trace a ray from the view point (called the primary ray) through each "pixel" on the image plane
  - Test each surface to determine if it is intersected by the ray.
  - Compute the points of intersection on each primary ray.
  - Get the point of intersection *Q* that is closest to the eye.
  - Use the colour of the object on which *Q* lies as pixel colour.





Ray Casting + Phong Lighting

- Still without secondary rays.
- At the point of intersection *Q*, recompute the colour value using an illumination model.



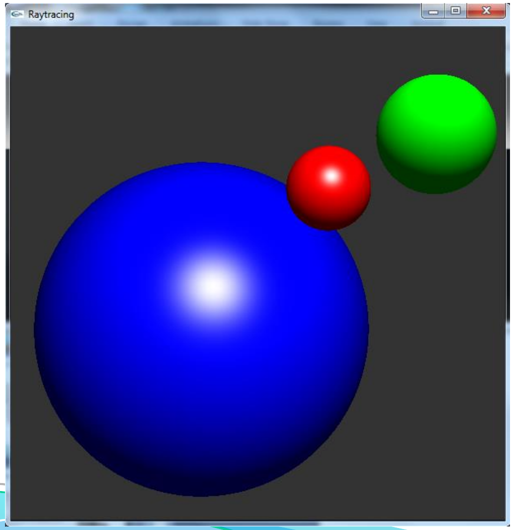
- Simplified Phong Illumination:

B = Background Color = (0.2, 0.2, 0.2)

M = Material Color (ambient and diffuse)

Light’s diffuse color = Light’s specular color = Material’s specular color = (1, 1, 1)

Col = BM + M $(\boldsymbol{L}\bullet\boldsymbol{N})$ + (1, 1, 1) $(\boldsymbol{R}\bullet\boldsymbol{V})^f$



Shadow Ray

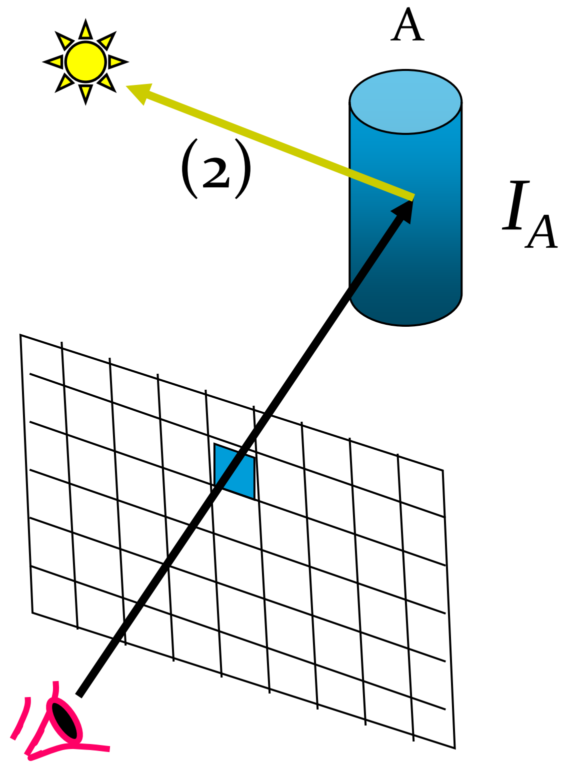
- Trace a ray from the point of intersection towards the light source (2)
- Colour I_A of object *A* determined using Phong model (previous slide)
- If the shadow ray hits an object, and if the point of intersection is between the light source and the object *A*, then apply only the ambient contribution of light to *Q*.

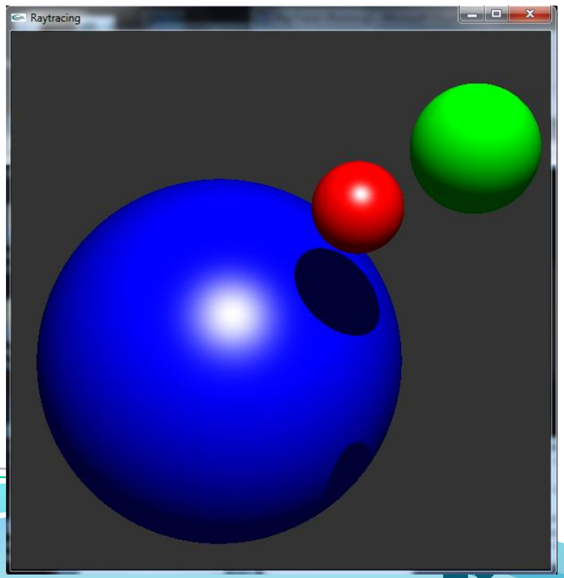
If *Q* is in shadow

$$I_A = \text{BM}$$

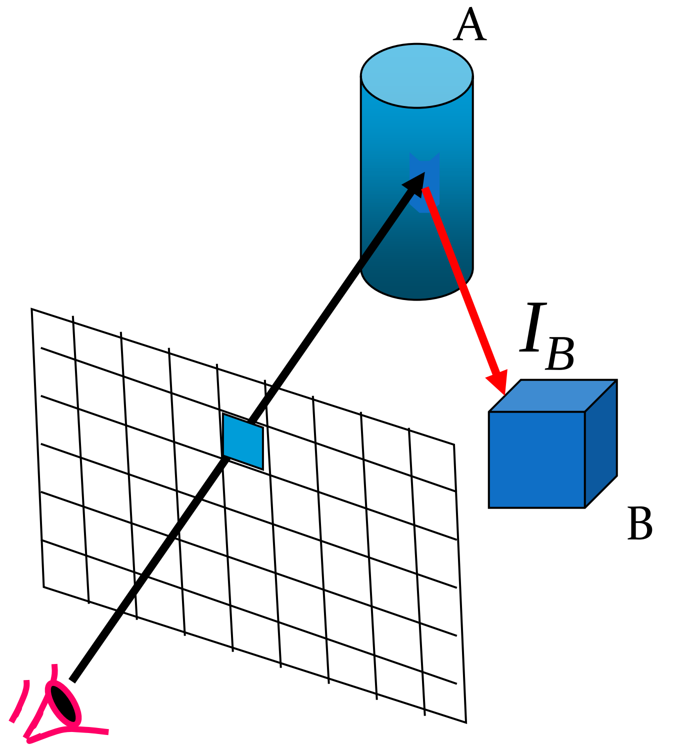
Else

$$I_A = \text{BM} + \text{M}\,(\boldsymbol{L}\bullet\boldsymbol{N}) + (1, 1, 1)\,(\boldsymbol{R}\bullet\boldsymbol{V})^f$$



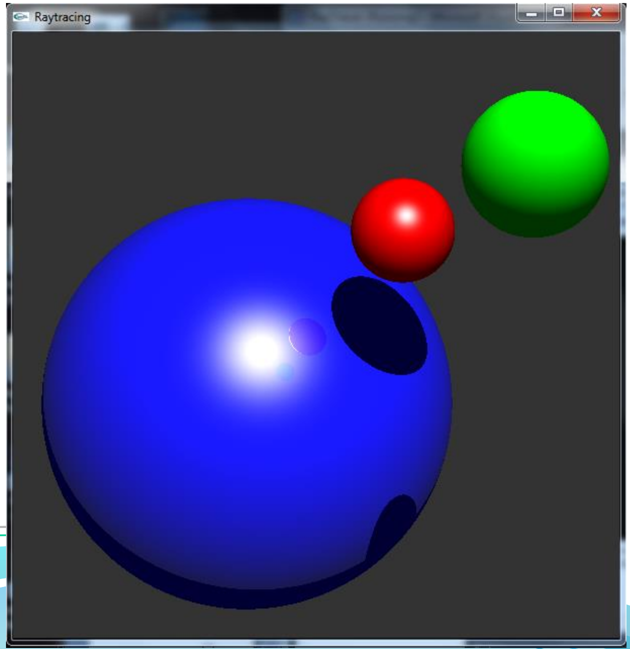


Reflections

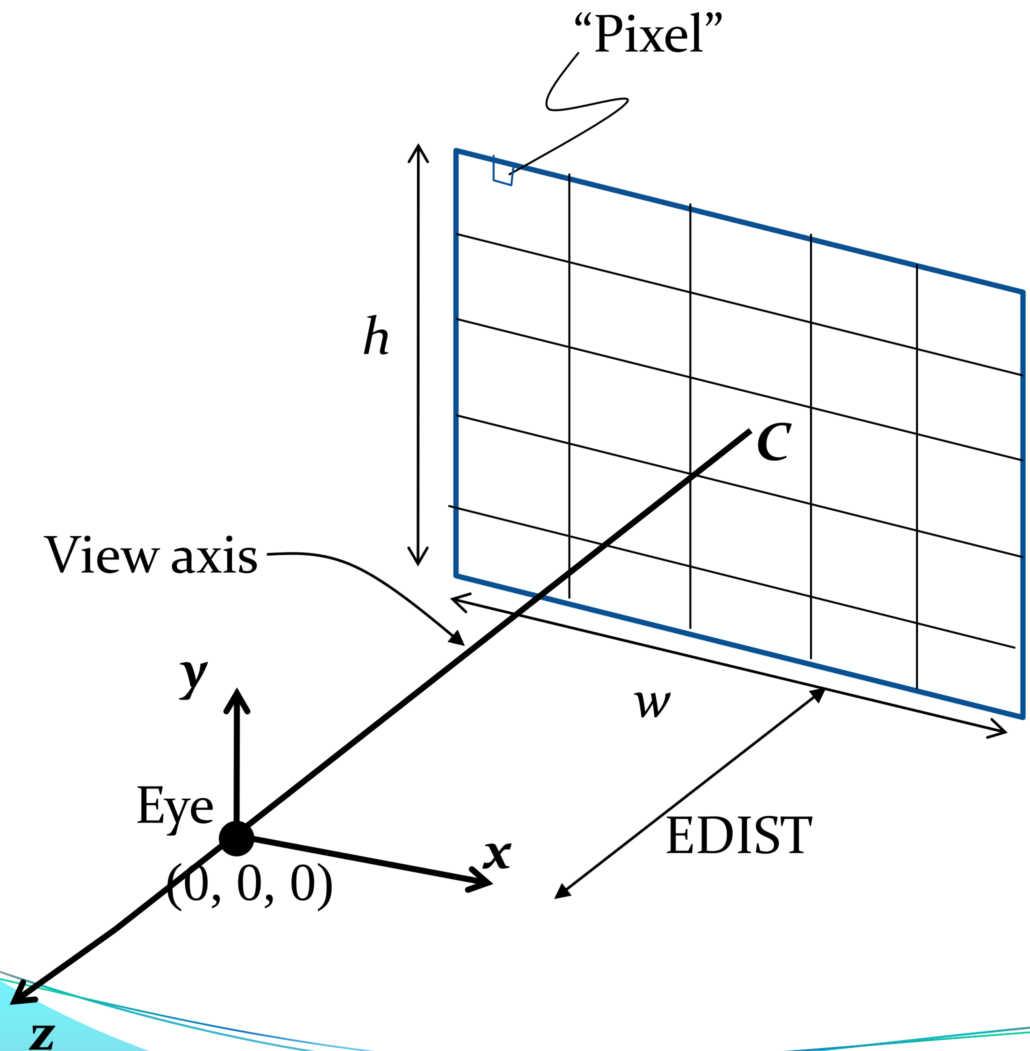


- If the surface is reflective, then a secondary ray along the direction of reflection is traced.
- If this secondary ray meets a surface at a point with intensity I_B , then $\rho_r I_B$ is added to the pixel color
  - ρ_r is a scale factor (<1), called the coeff. of reflection
  - ρ_r represents how much of colour I_B is reflected on the surface A.
- The colour of the pixel is now

$$I = I_A + \rho_r I_B$$

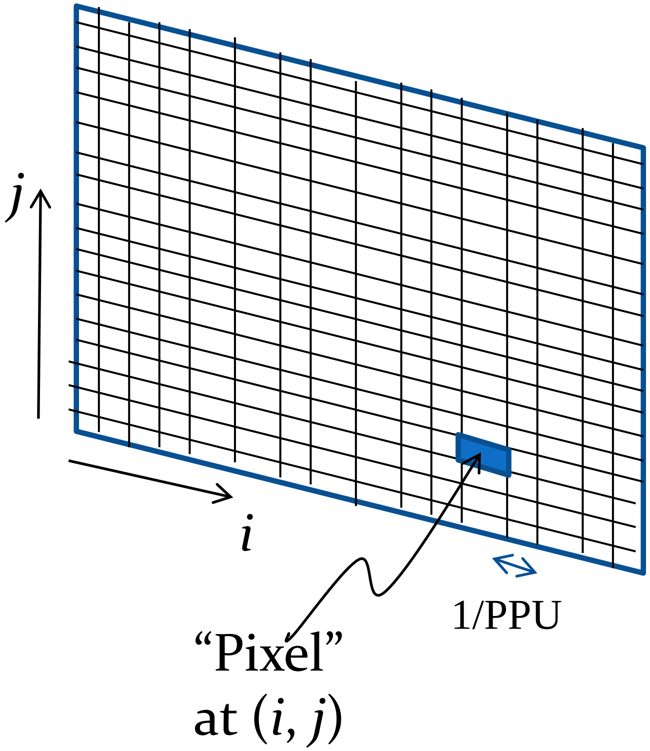


Ray Tracing Setup



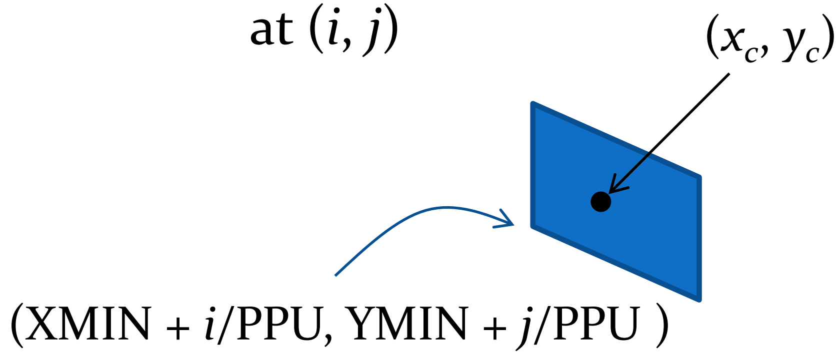
w = Width of screen in world units (eg. 5 units).
h = Height of screen in world units (eg. 5 units).
EDIST = Eye dist in world units (eg. 10 units)
PPU = Pixels per unit (eg. 100)
XMIN = $-w/2$; XMAX = $w/2$;
YMIN = $-h/2$; YMAX = $h/2$;

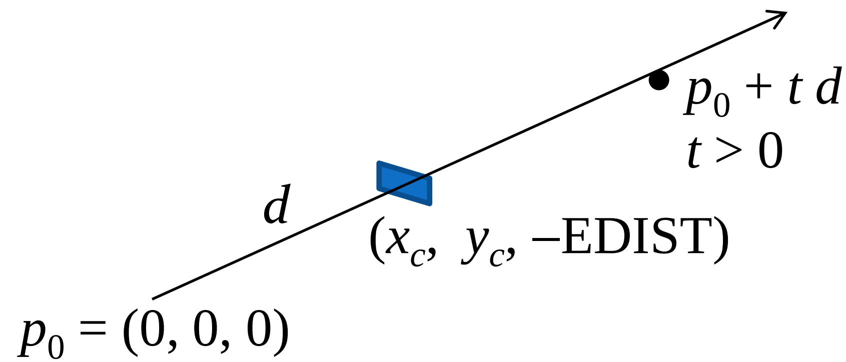
Primary Ray



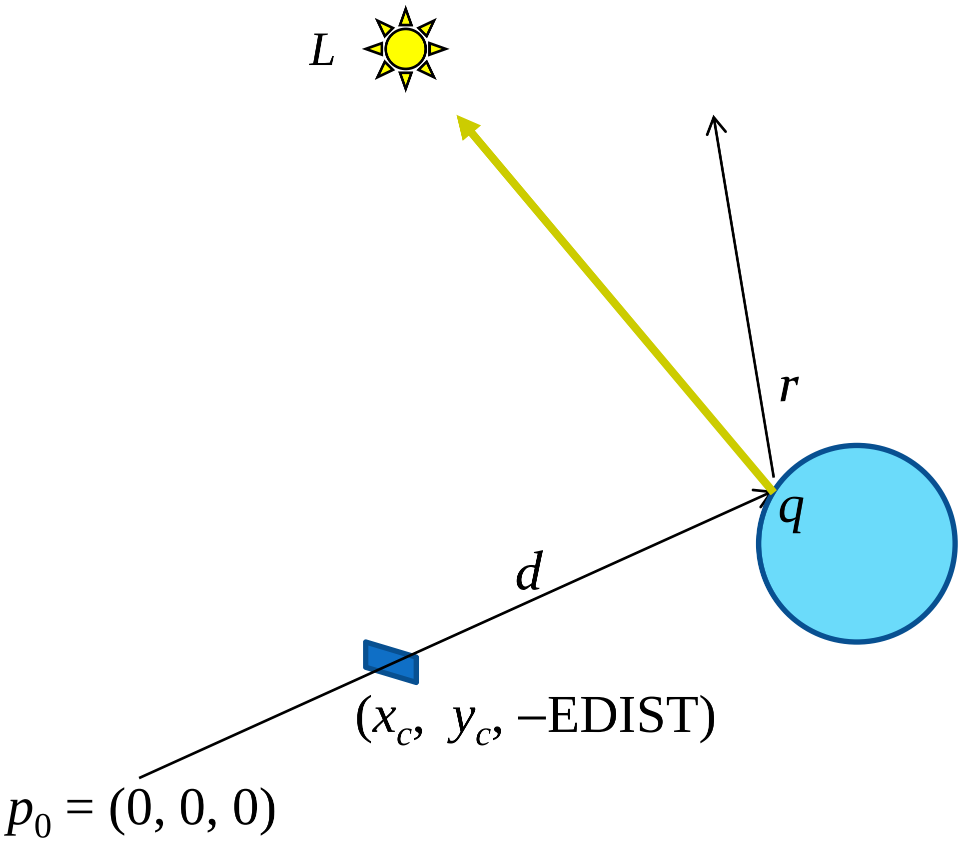
i : 0...w*PPU–1
j: 0...h*PPU–1.

Primary ray:
Position p_0 = (0, 0, 0)
Direction d = $(x_c, y_c, -\text{EDIST})$
(Always normalize direction!)





Secondary Rays



Shadow ray:

Position = q

Direction = $L - q$ normalized.

Reflection ray:

Position = q

Direction:

$r = -2(d \cdot n)n + d$ normalized.

- Every ray is specified using its point of origin, and a unit vector denoting its direction.
- Any point on the ray can be represented by a single parameter $t > 0$. The distance of the point from the source of the ray is t.

Ray-plane intersection

- Given a point a on a plane, and the normal vector n of the plane, the plane's equation can be written as

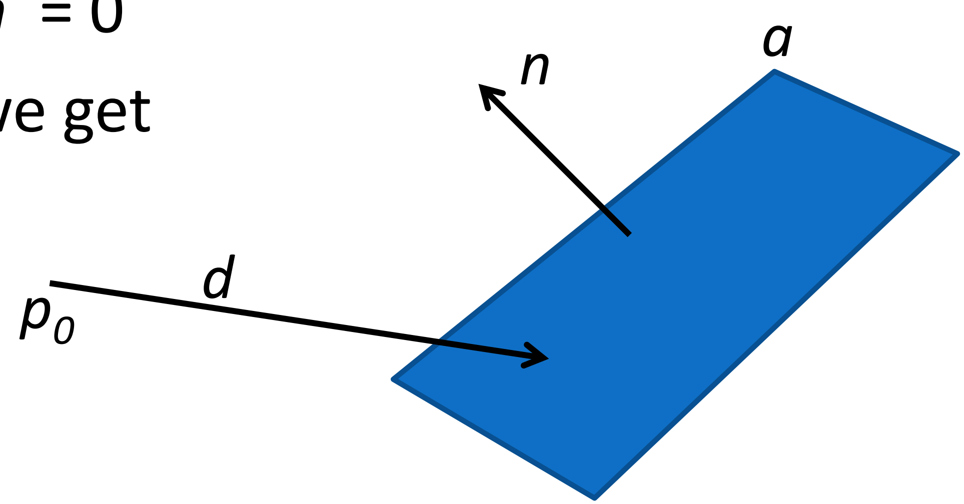
  $(p - a) \bullet n = 0$

- A ray is given by the equation

  $p = p_0 + t\,d.$

- At the point of intersection, both equations are true. Therefore, $(p_0 + t\,d - a) \bullet n = 0$
- From the above equation, we get

$$t = \frac{(a - p_0) \bullet n}{d \bullet n}$$



Ray-sphere intersection

- Equation of a sphere centred at C with radius r is

  $(p - C) \bullet (p - C) = r^2$

- Consider a ray given by the equation

  $p = p_0 + t\,d.$

- At the point of intersection, both equations are true.

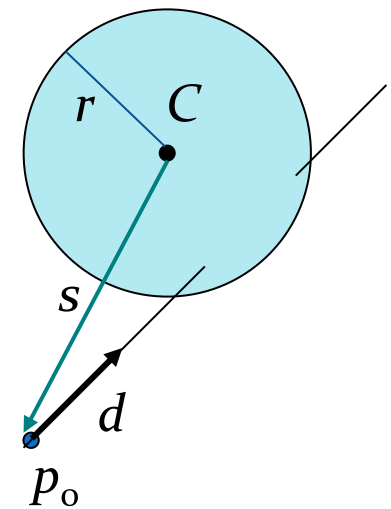
Therefore, $(p_0 + t\,d - C) \bullet (p_0 + t\,d - C) = r^2$

$(s + t\,d) \bullet (s + t\,d) = r^2$, where $s = p_0 - C$

$(d \bullet d)\, t^2 + 2\,(s \bullet d)\, t + (s \bullet s) - r^2 = 0.$

Since d is a unit vector, we get

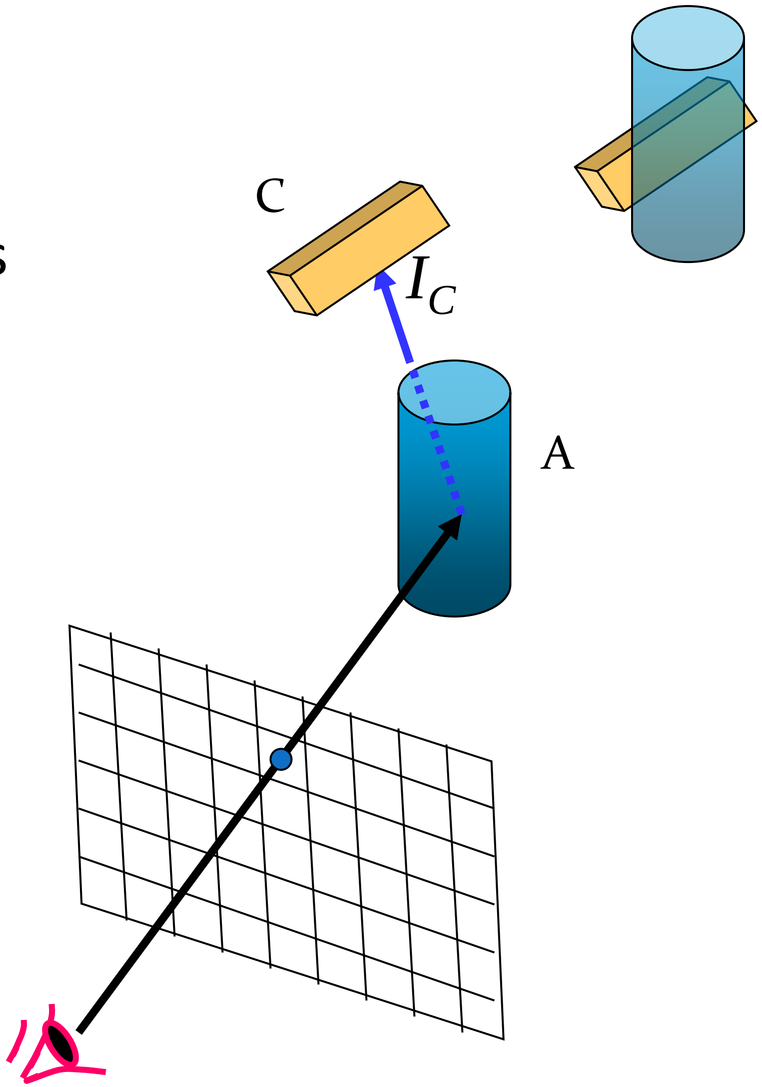
$$t = -(s \bullet d) \pm \sqrt{(s \bullet d)^2 - (s \bullet s) + r^2}$$



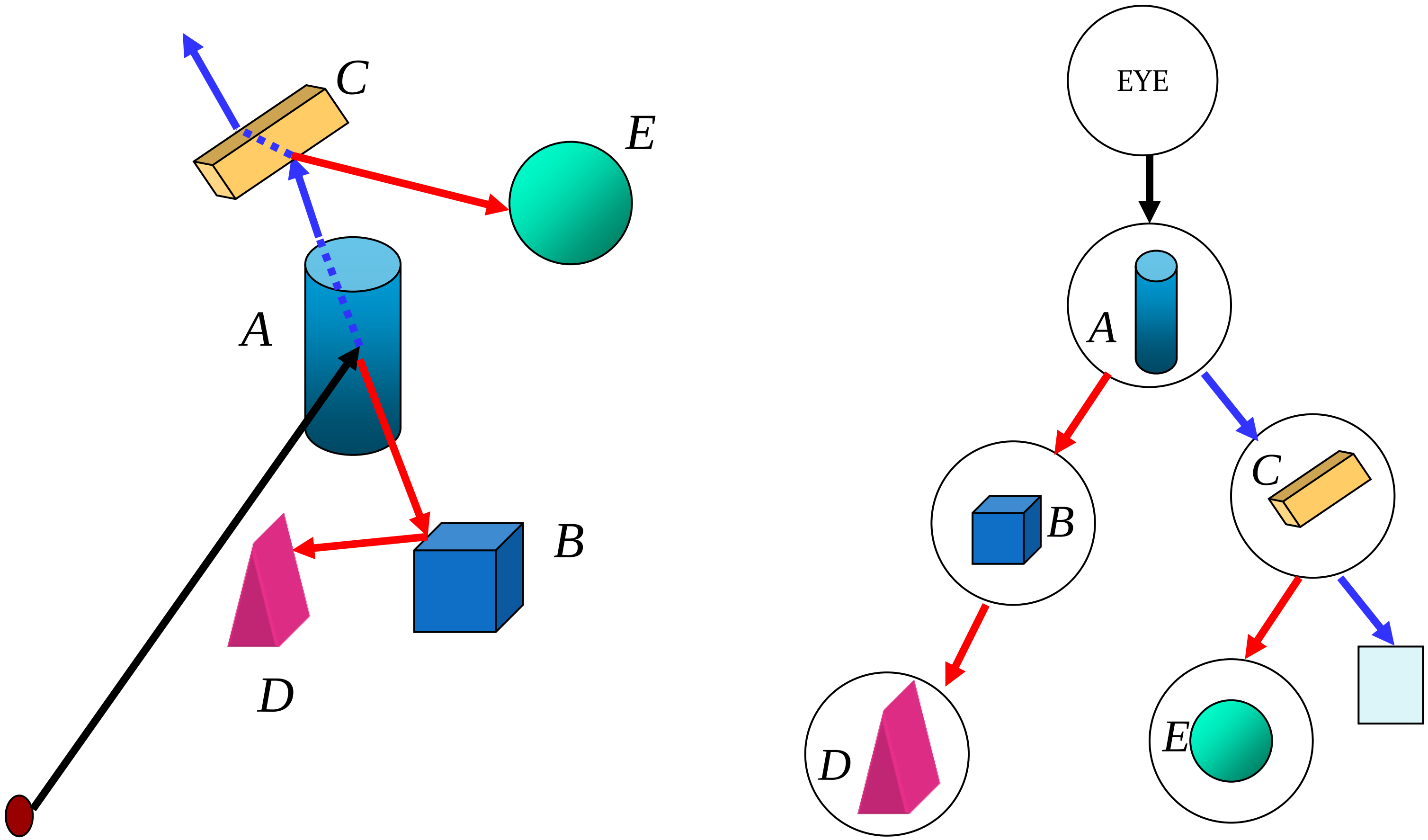
Refractions

- If the surface is transparent/translucent, a secondary ray along the direction of refraction is traced.
- If this secondary ray meets a surface at a point with intensity I_C, then $\rho_t I_C$ is added to the pixel color.
  - ρ_t is a scale factor (<1), called the coeff. of transmission
- The colour of the pixel is now

$$I = I_A + \rho_c I_c$$



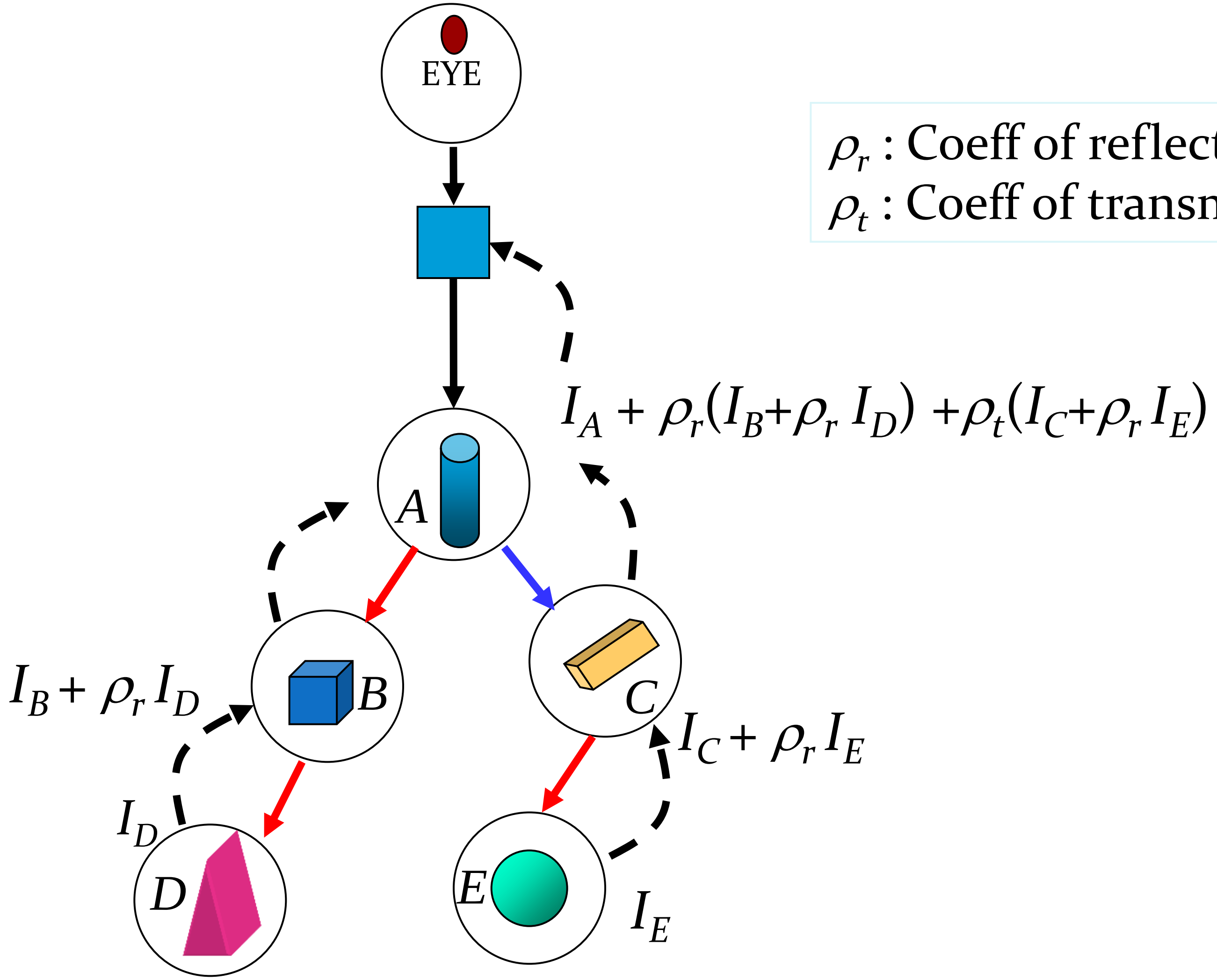
Binary Ray-tracing Tree



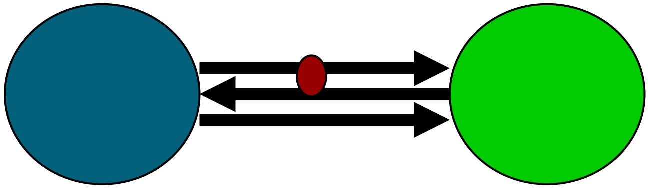
- Left branches represent reflections
- Right branches represent transmission paths

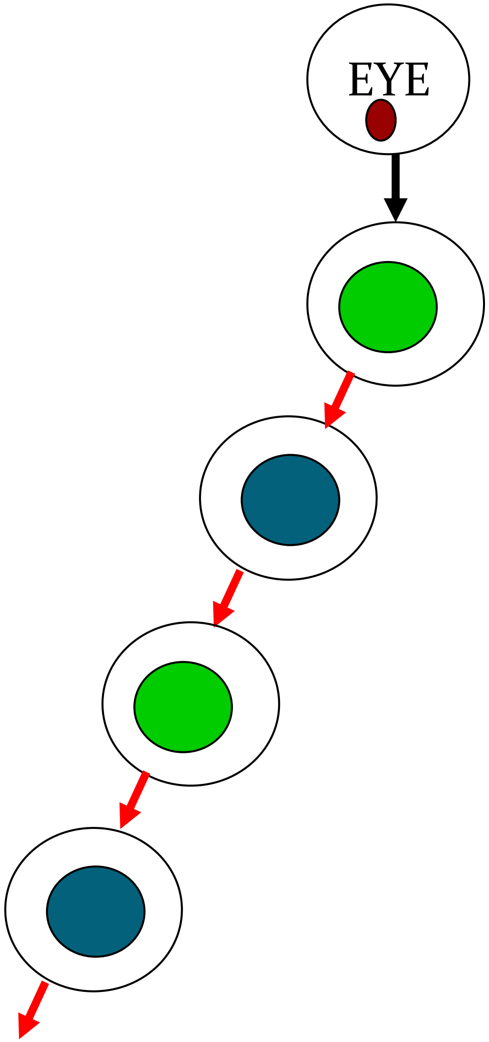
Accumulation of Intensities

ρ_r : Coeff of reflection
ρ_t : Coeff of transmission



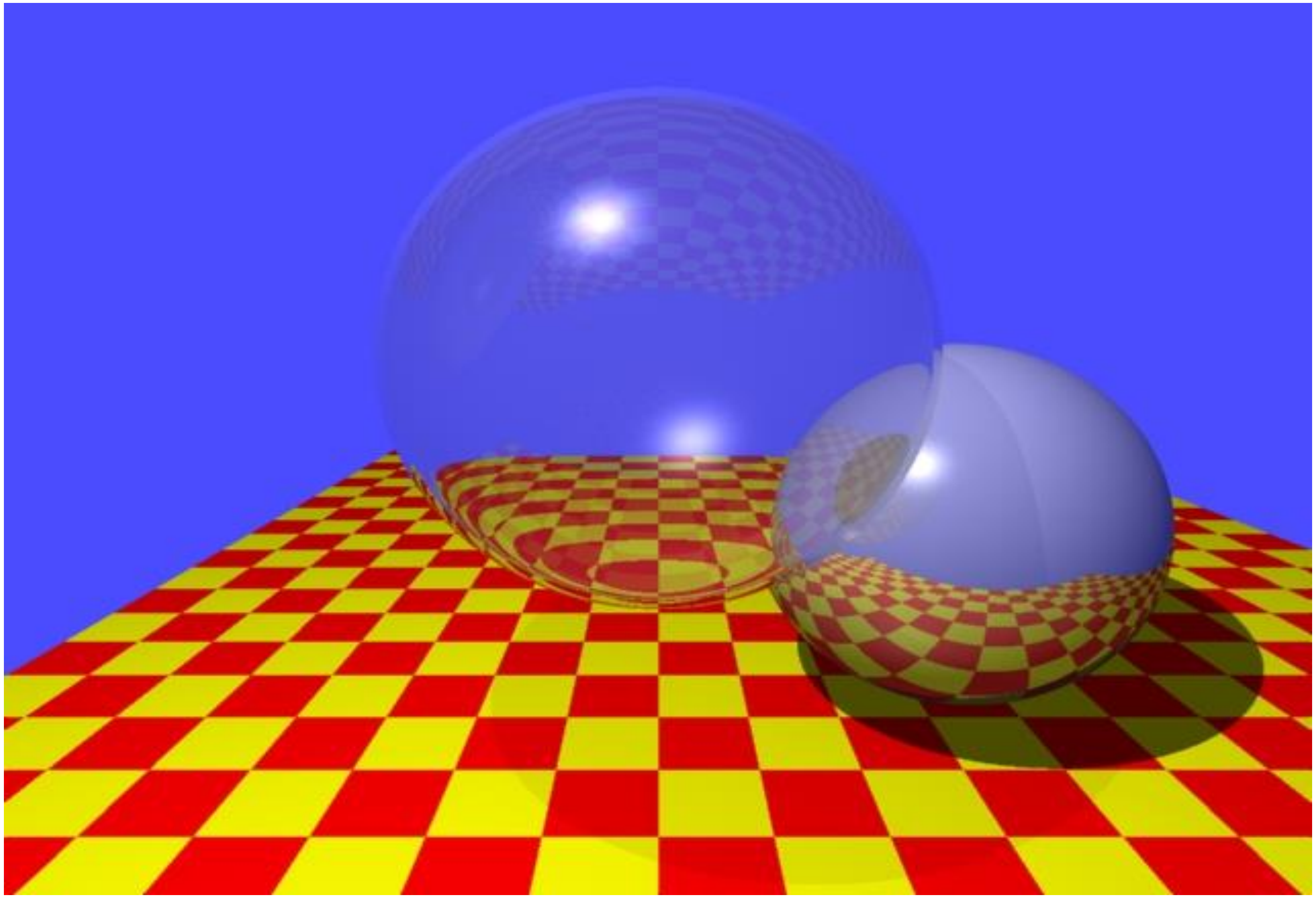
Multiple Reflections





Must limit recursion depth

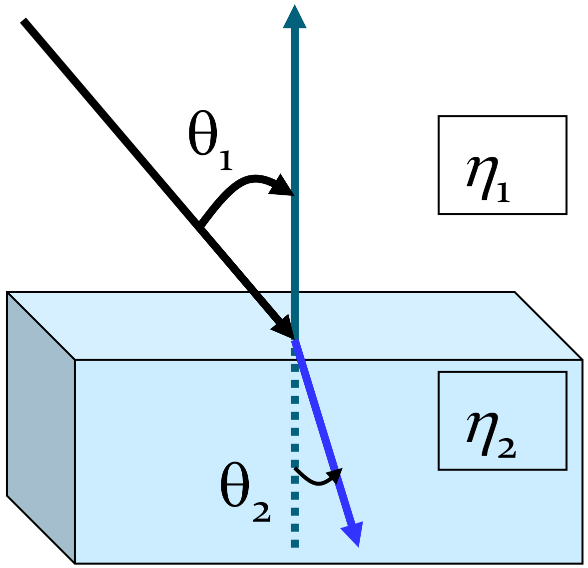
Ray Tracing: Example



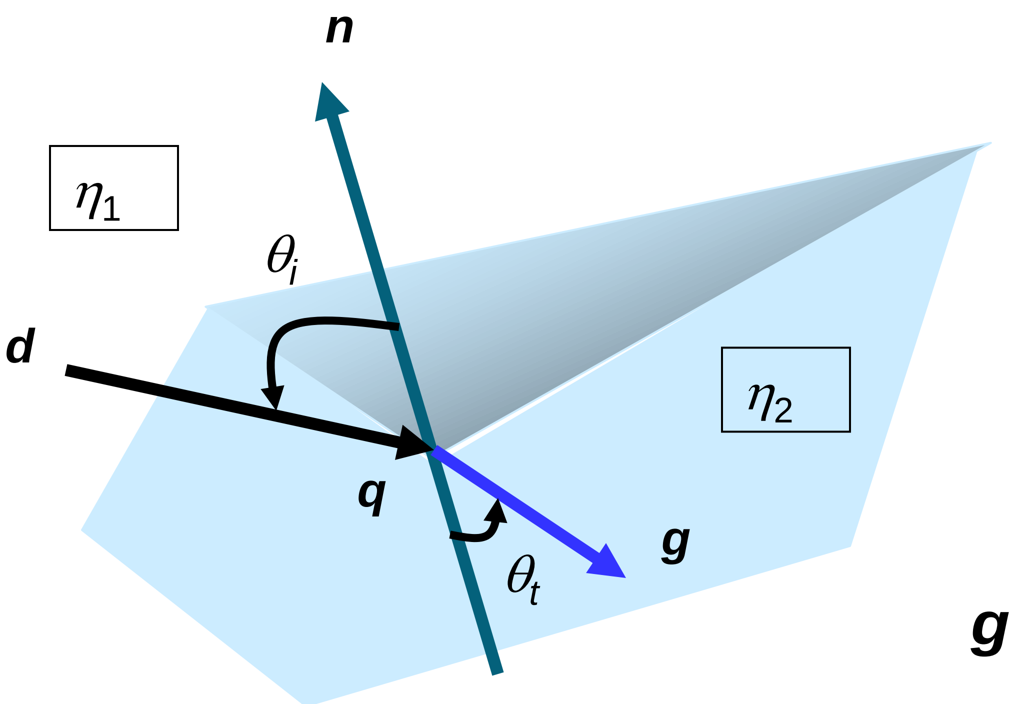
Index of Refraction

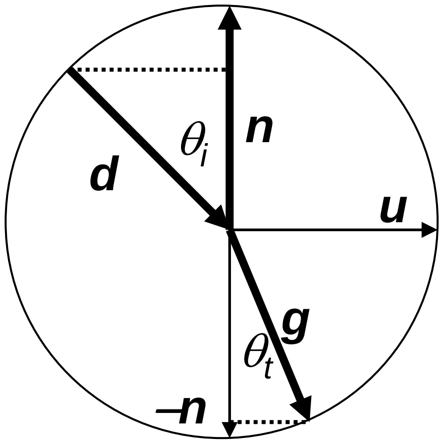
- Light travels at speed c/η in a medium with index of refraction η.
- Common values of index of refraction:
  - Air 1.
  - Water 1.33
  - Glass 1.5
  - Diamond 2.4

- Snell's Law of Refraction:
  - $\eta_1 \sin\theta_1 = \eta_2 \sin\theta_2$



Refracted Ray



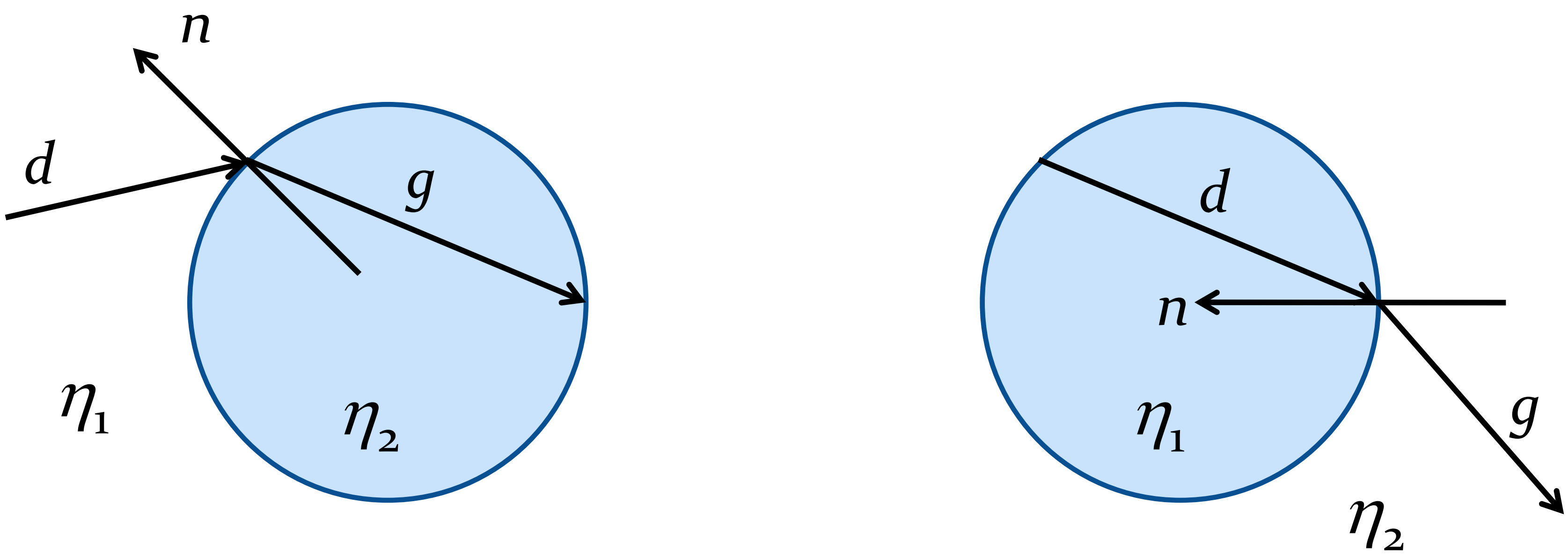


$$\boxed{\begin{aligned} \boldsymbol{u}\sin\theta_i - \boldsymbol{n}\cos\theta_i &= \boldsymbol{d} \\ \boldsymbol{u}\sin\theta_t - \boldsymbol{n}\cos\theta_t &= \boldsymbol{g} \end{aligned}}$$

$$\boldsymbol{g} = \left(\frac{\eta_1}{\eta_2}\right)\boldsymbol{d} - \left(\frac{\eta_1}{\eta_2}(\boldsymbol{d.n}) + \cos\theta_t\right)\boldsymbol{n}$$

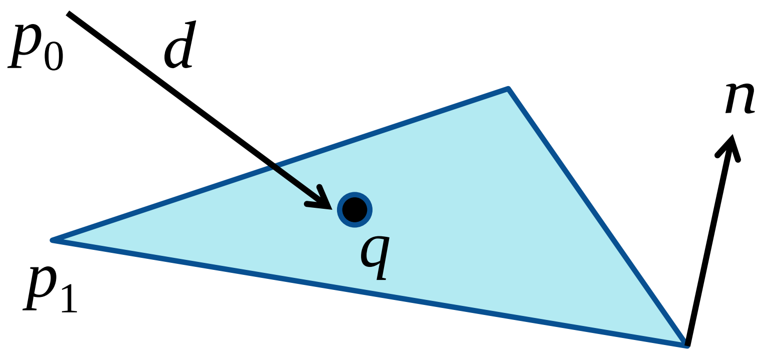
$$\cos\theta_t = \sqrt{\left(1 - \left(\frac{\eta_1}{\eta_2}\right)^2 \left(1 - (\boldsymbol{d.n})^2\right)\right)}$$

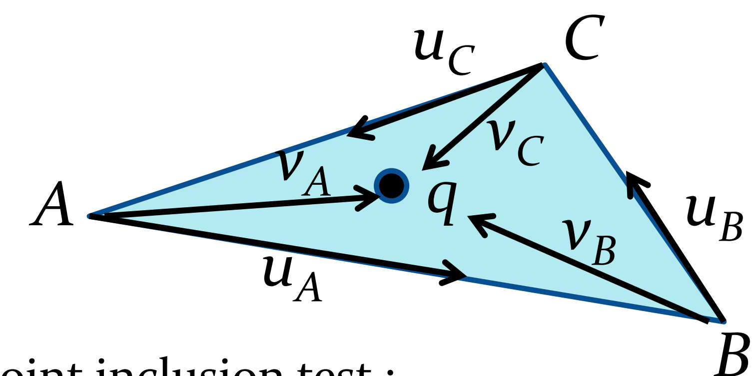
Sphere Refraction



Ray intersection with polygonal objects

- Use the plane equation of each triangle to get the point of intersection with the ray.
- Check if the point of intersection lies within the triangle.





Intersection :

$$t = \frac{(p_1 - p_0)\cdot n}{d\cdot n}$$

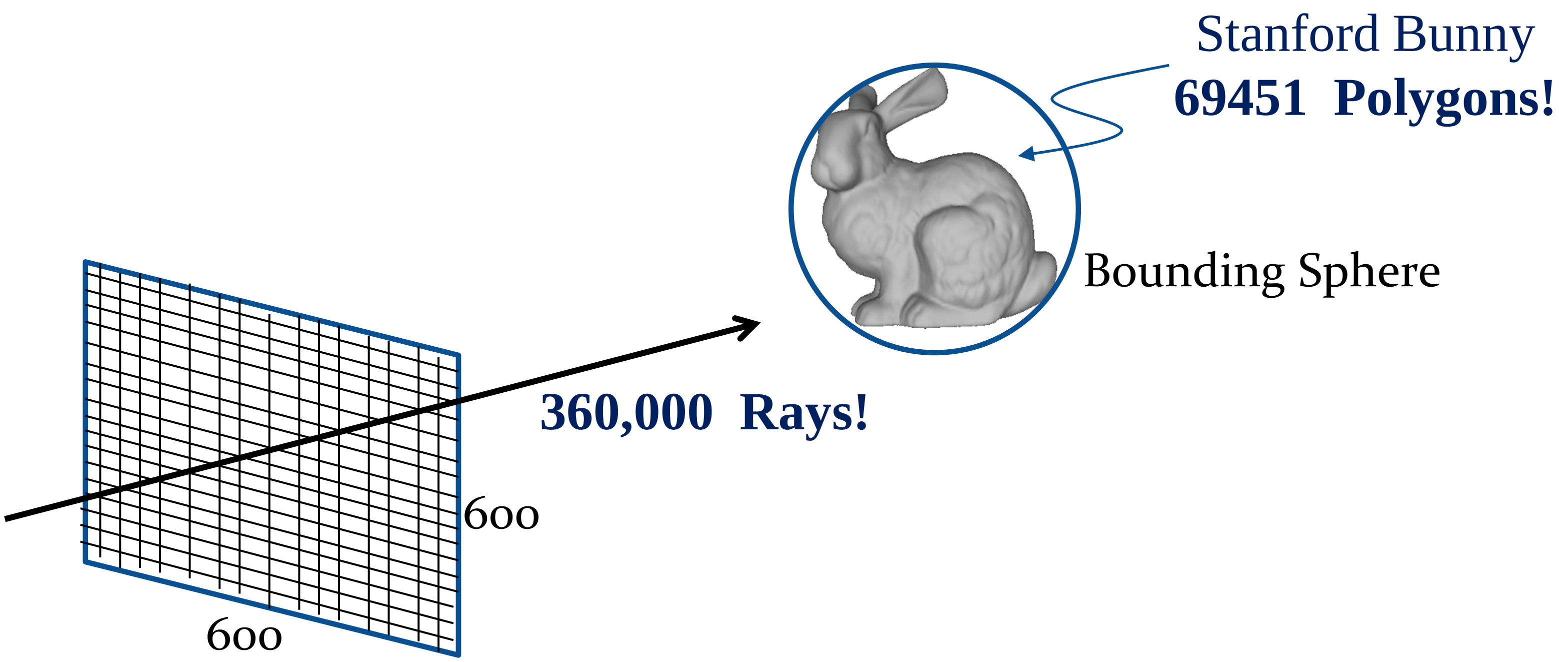
$$q = p_0 + \ t\, d$$

Point inclusion test :

If the cross products $(u_A \times v_A)$, $(u_B \times v_B)$, $(u_C \times v_C)$ have the same sign, then the point q is inside the triangle.

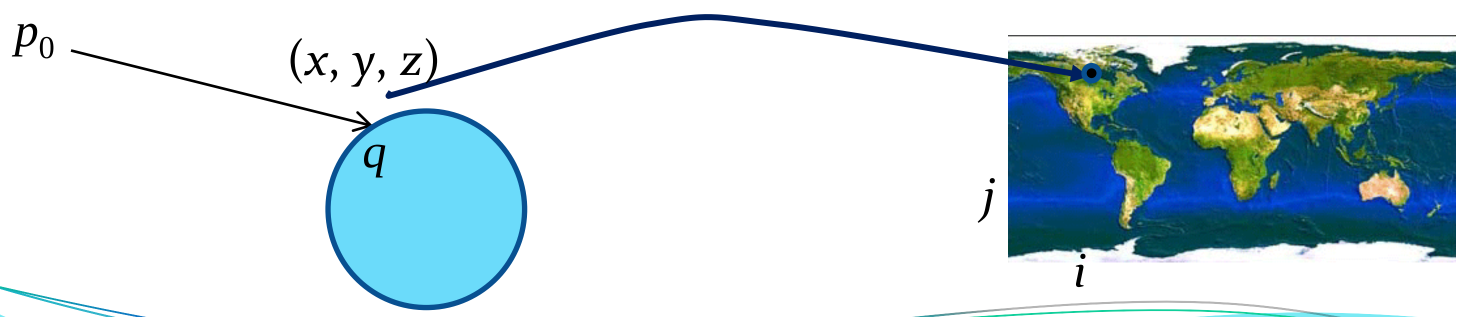
Ray intersection with polygonal objects

- Complex polygonal objects will require a large amount of ray-triangle intersection tests.
- Bounding volume hierarchies and spatial subdivision methods (*k*d-Trees, Octrees) are used to reduce the number of ray-primitive intersection tests.



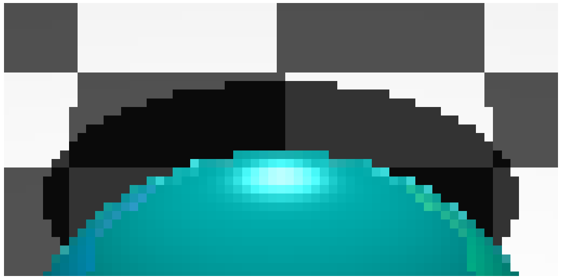
Texture Mapping

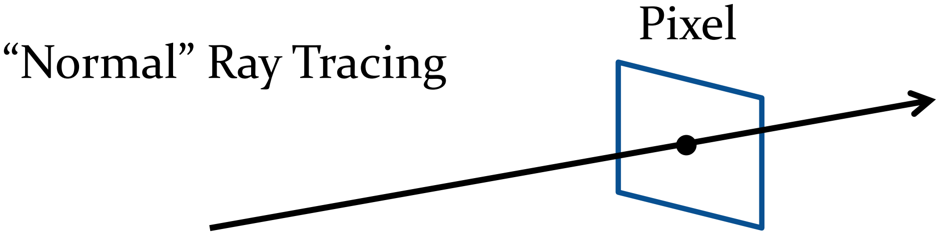
- 2D texturing:
  - Similar to OpenGL texturing ~~(introduced later in this course)~~, but does not use texture coordinates or texture memory.
  - Map the coordinates of the point of intersection (*x*, *y*, *z*) to image coordinates, and assign the colour of the pixel to that point.
- 3D texturing:
  - Define a function to map (*x*, *y*, *z*) coordinates to (*r*, *g*, *b*) colour values.



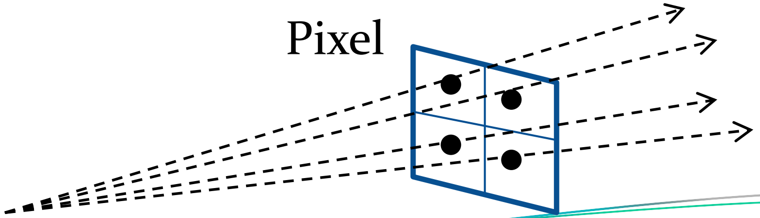
Anti-Aliasing

The ray tracing algorithm samples the light field using a finite set of rays generated through a discretized image space. This results in distortion artefacts such as jaggedness along edges of polygons and shadows.



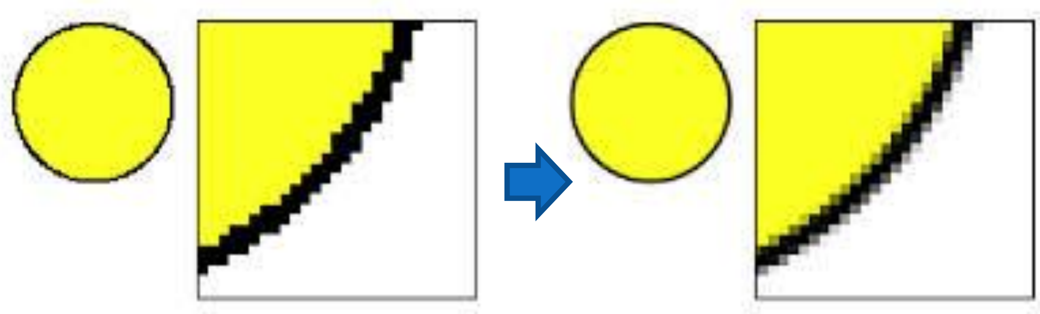


- Supersampling: Generate several rays through each square pixel (eg. divide the pixel into four equal segments) and compute the average of the colour values.



Anti-Aliasing

- Adaptive Sampling: As shown on the previous slide, each pixel is divided into four “sub-pixels”. Primary rays are generated through the centres of each sub-pixel. If the colour value along any ray varies significantly from the other three, that sub-pixel is split further into four sub-pixels, and more rays are generated through them.

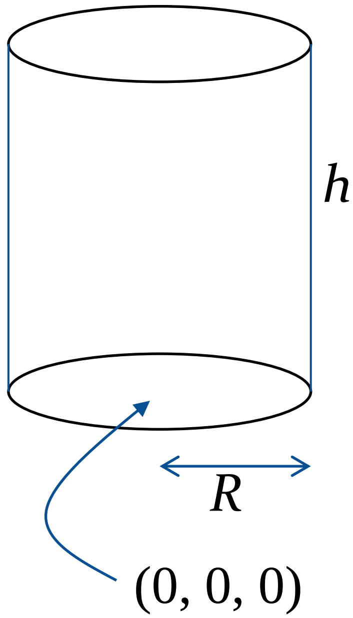


Cylinder

- A cylinder at the origin with axis along the *y*-axis, radius *R* and height *h* is given by

  $x^2 + z^2 = R^2$

  $0 \le y \le h$



- Normal vector at (x, y, z)

  (un-normalized) $n = (x, 0, z)$

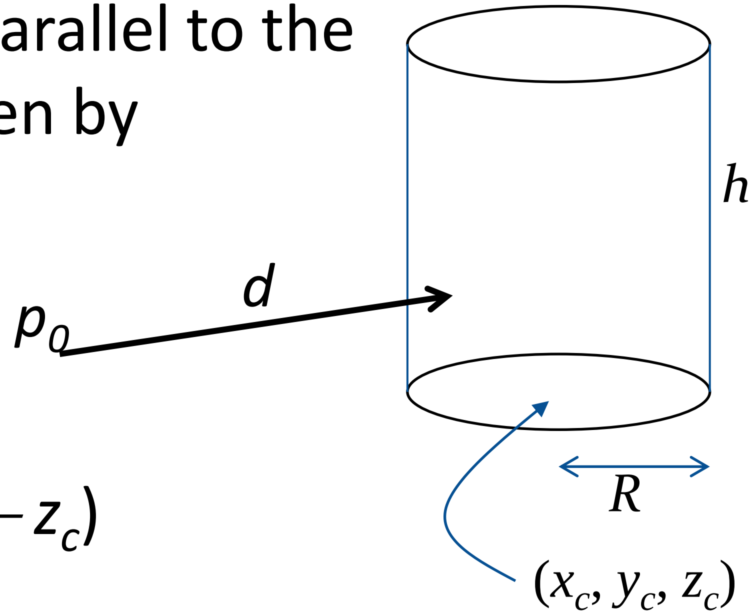
  Normalized $n = (x/R,\ 0,\ z/R)$

Ray – Cylinder Intersection

- A cylinder at (x_c, y_c, z_c), with axis parallel to the y-axis, radius R and height h is given by

$$(x - x_c)^2 + (z - z_c)^2 = R^2$$

$$0 \le (y - y_c) \le h$$



- Normal vector at (x, y, z)

(un-normalized) $n = (x - x_c,\ 0,\ z - z_c)$

- Ray equation:

$x = x_0 + d_x\, t;\qquad y = y_0 + d_y\, t;\qquad z = z_0 + d_z\, t;$

- Intersection equation:

$$t^2\left(d_x^2 + d_z^2\right) + 2t\left\{d_x\left(x_0 - x_c\right) + d_z\left(z_0 - z_c\right)\right\} + \left\{\left(x_0 - x_c\right)^2 + \left(z_0 - z_c\right)^2 - R^2\right\} = 0.$$

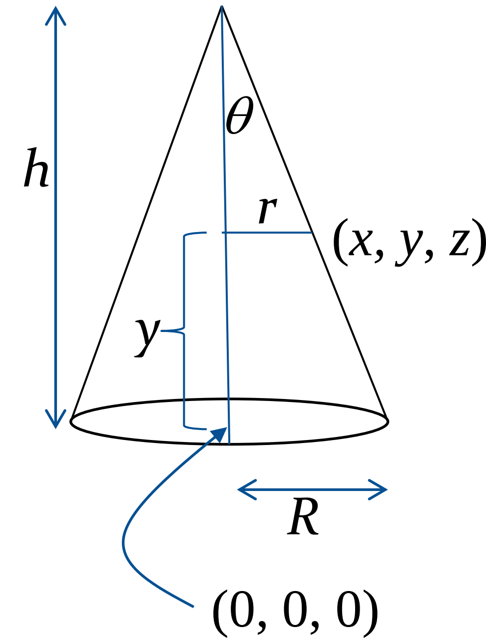
Cone

- Consider a cone with base at the origin, axis parallel to the *y*-axis, radius *R*, and height *h*:
- Important equations:

$$\tan(\theta) = R/h$$

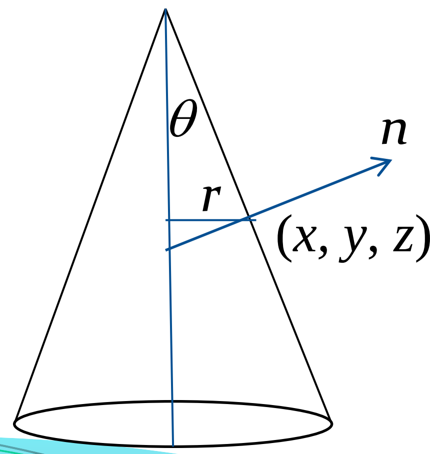
$$x^2 + z^2 = r^2$$

$$x^2 + z^2 = \left(\frac{R}{h}\right)^2 (h - y)^2$$



- Surface normal vector (un-normalized):

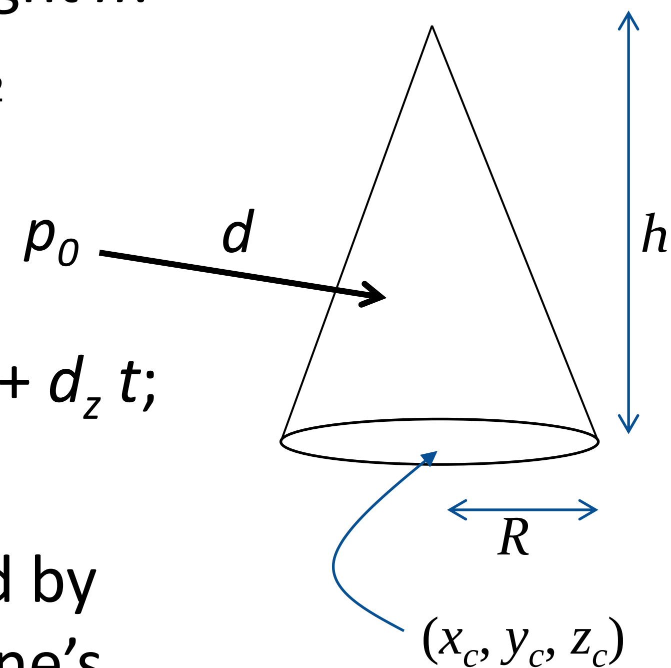
$$n = (x,\ r\tan(\theta),\ z)$$



Ray-Cone Intersection

- Equation of a cone with base at (x_c, y_c, z_c), axis parallel to the *y*-axis, radius *R*, and height *h*:

$$(x - x_c)^2 + (z - z_c)^2 = \left(\frac{R}{h}\right)^2 (h - y + y_c)^2$$



- Ray equation:

$$x = x_0 + d_x\, t; \qquad y = y_0 + d_y\, t; \qquad z = z_0 + d_z\, t;$$

- The points of intersection are obtained by substituting the ray equation in the cone's equation and solving for *t*.