CT4201/EC4215: Computer Graphics

Ray Tracing

BOCHANG MOON

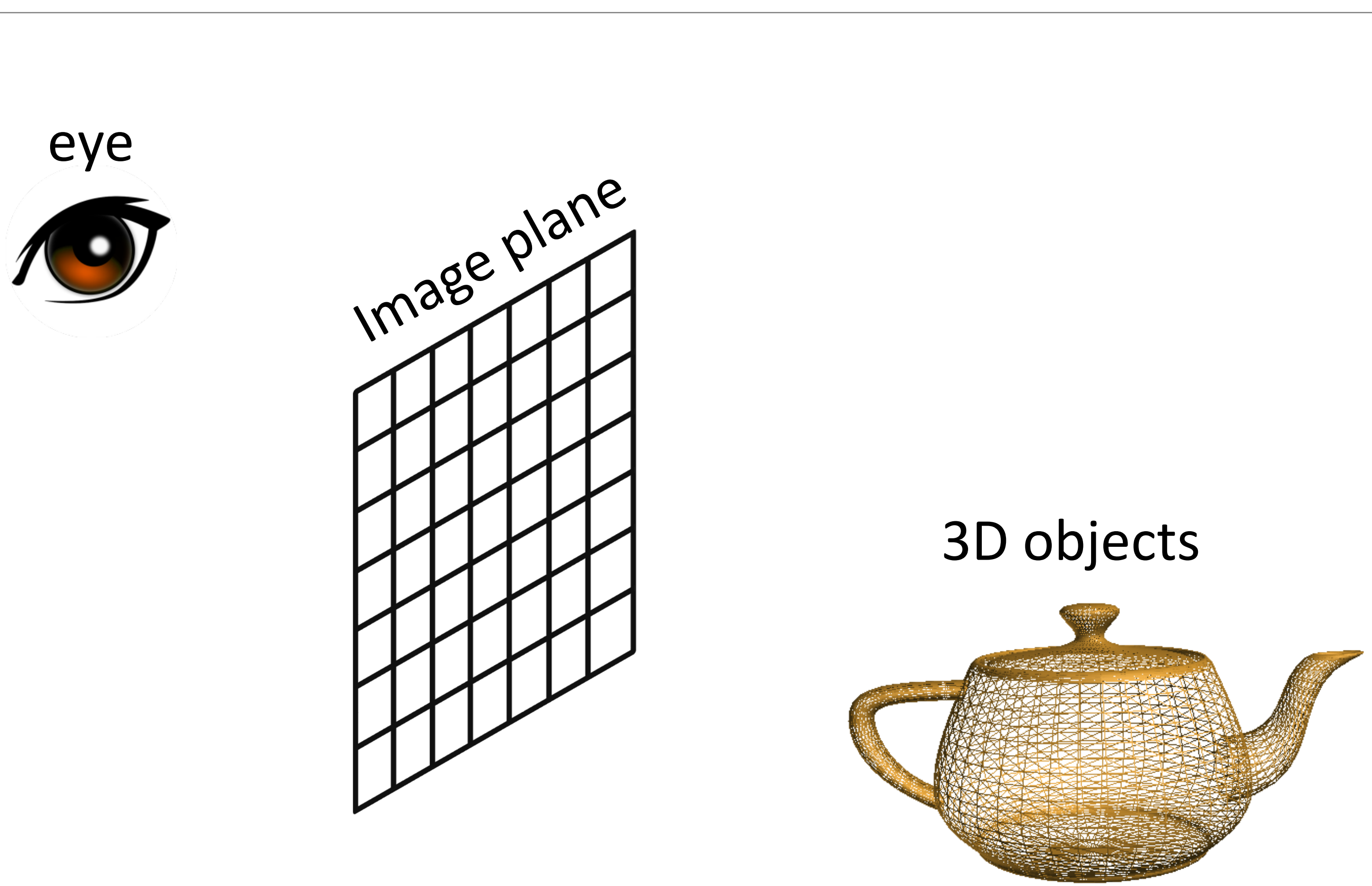
Ray Tracing

- A rendering technique:
  - Produce a 2D image from a scene (models)

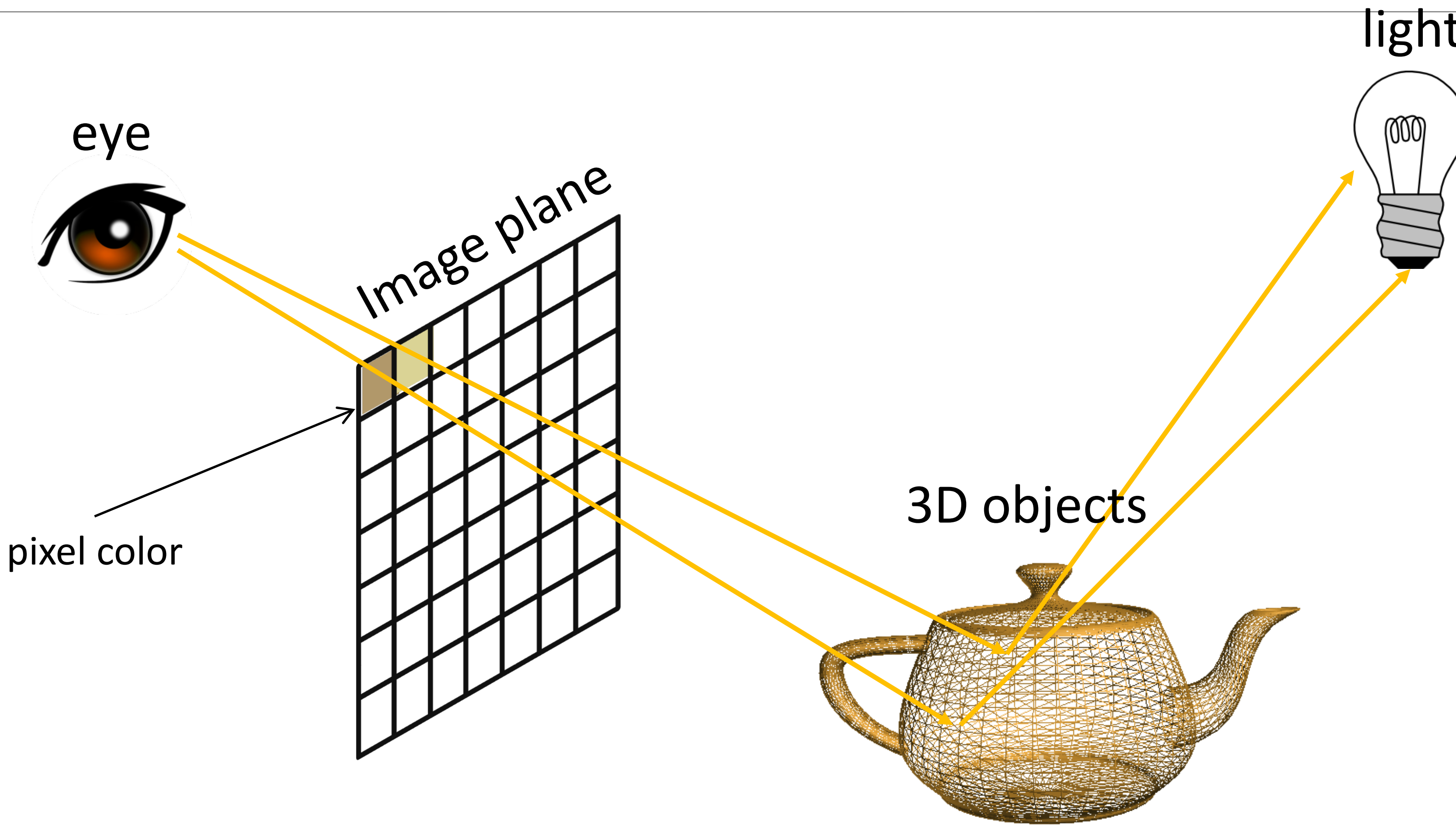
- Image-order rendering:
  - Loop over pixels to decide pixel colors

- Object-order rendering:
  - Iterate objects and compute some pixel colors related to each object

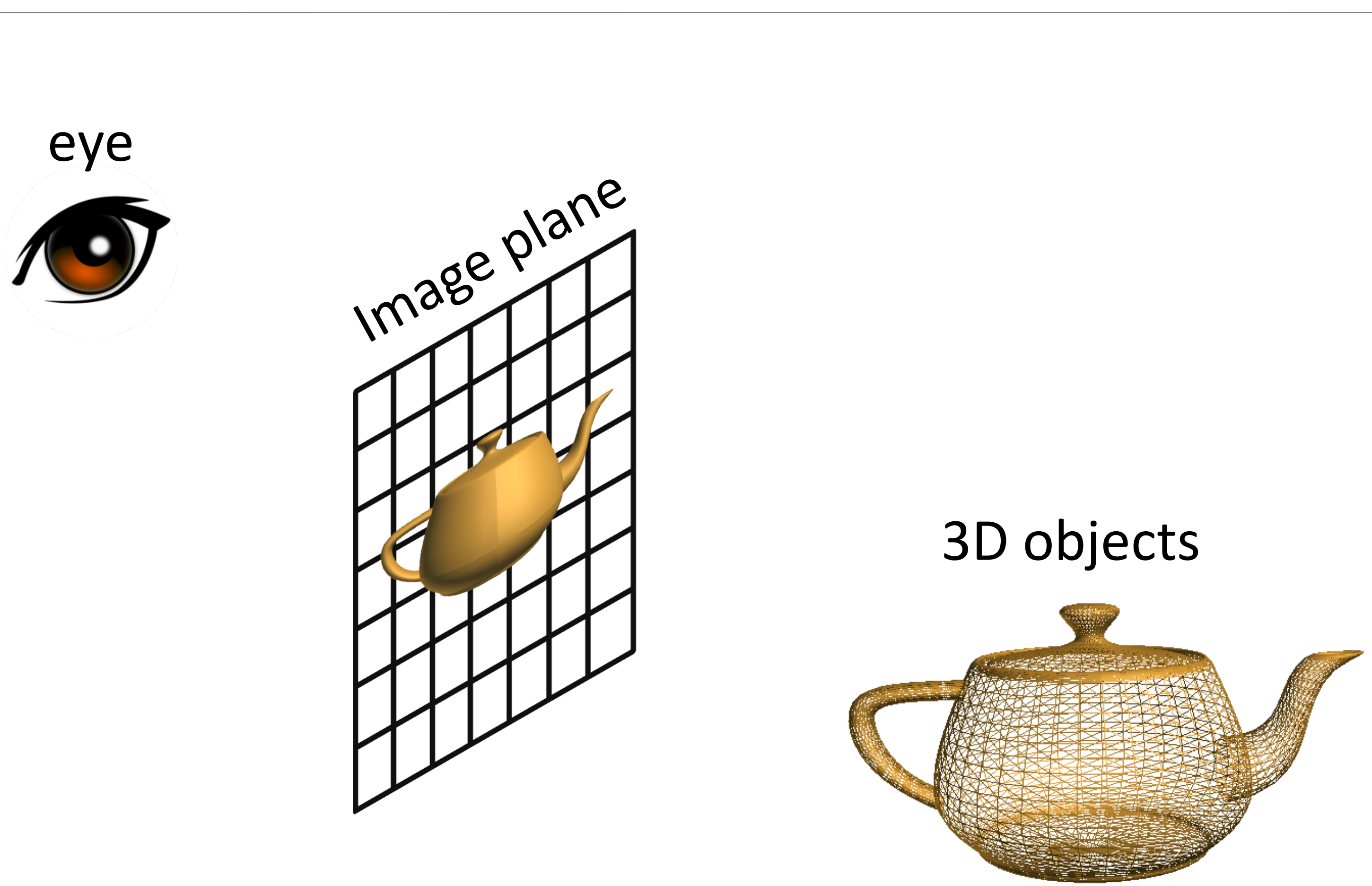
Rendering



Ray Tracing

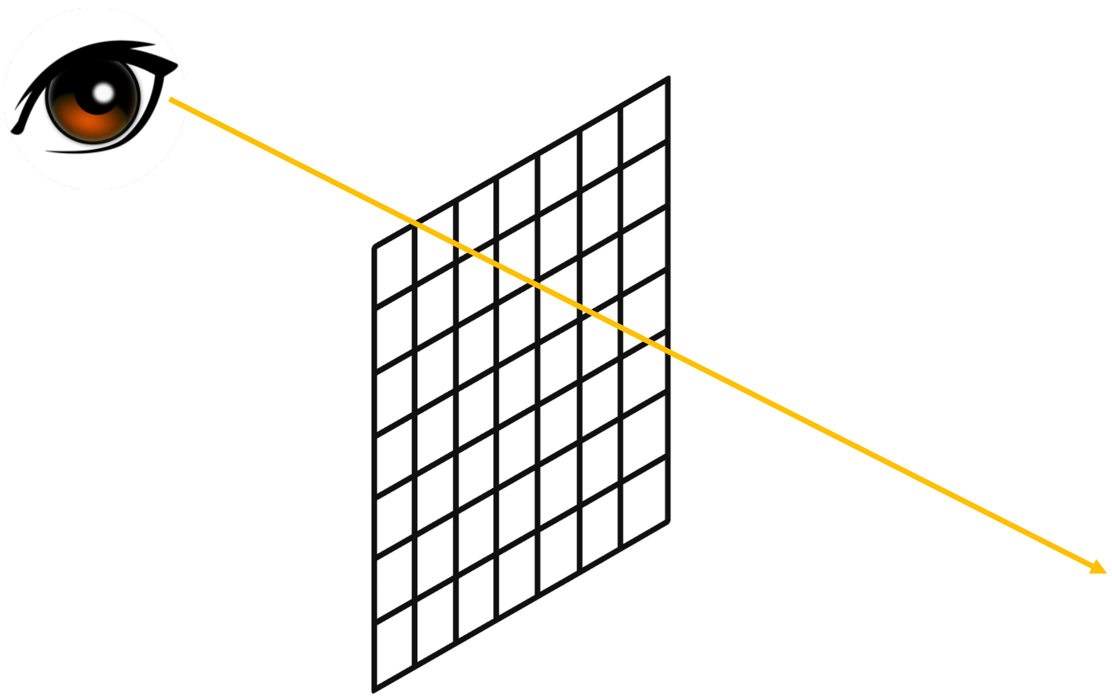


Ray Tracing



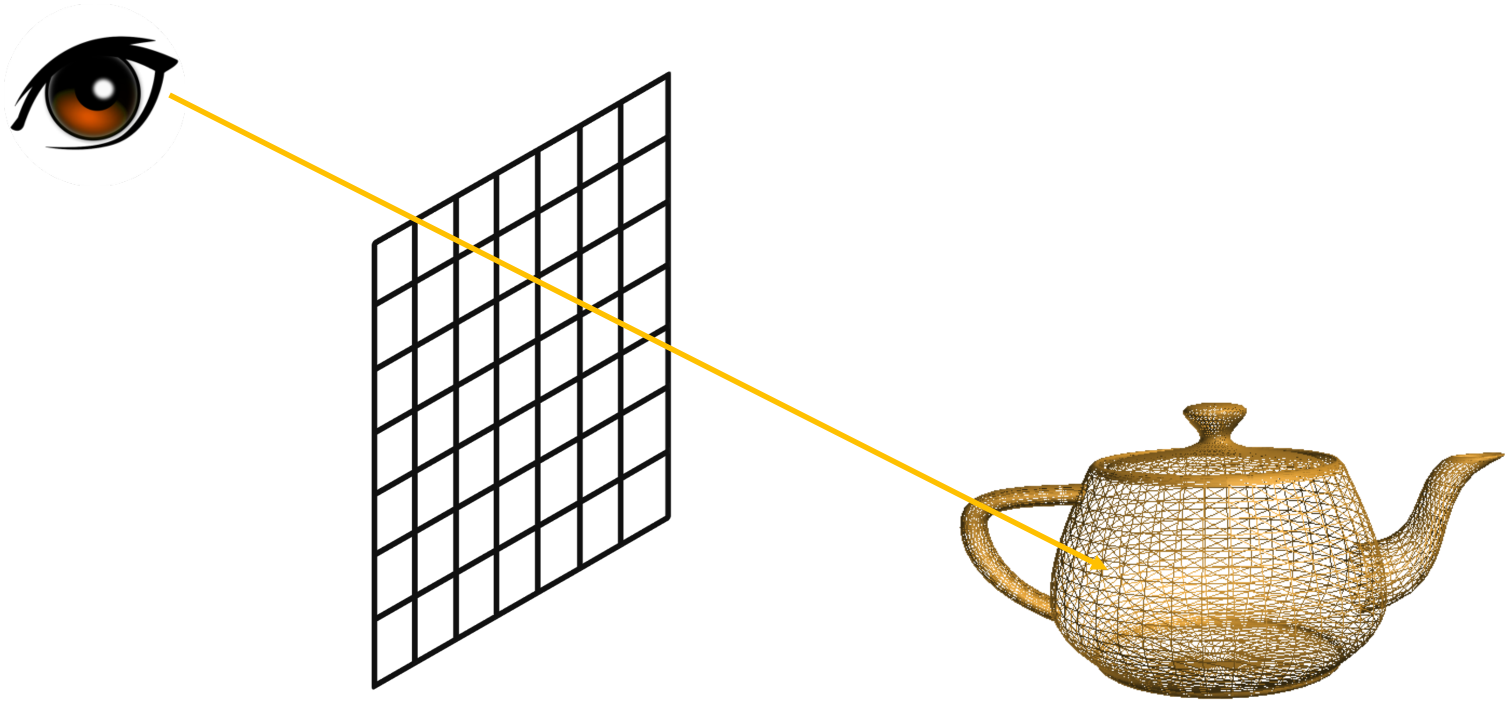
Basic Ray Tracer

- Ray generation
  - Compute the origin and direction of a ray per pixel, by considering the camera and image plane



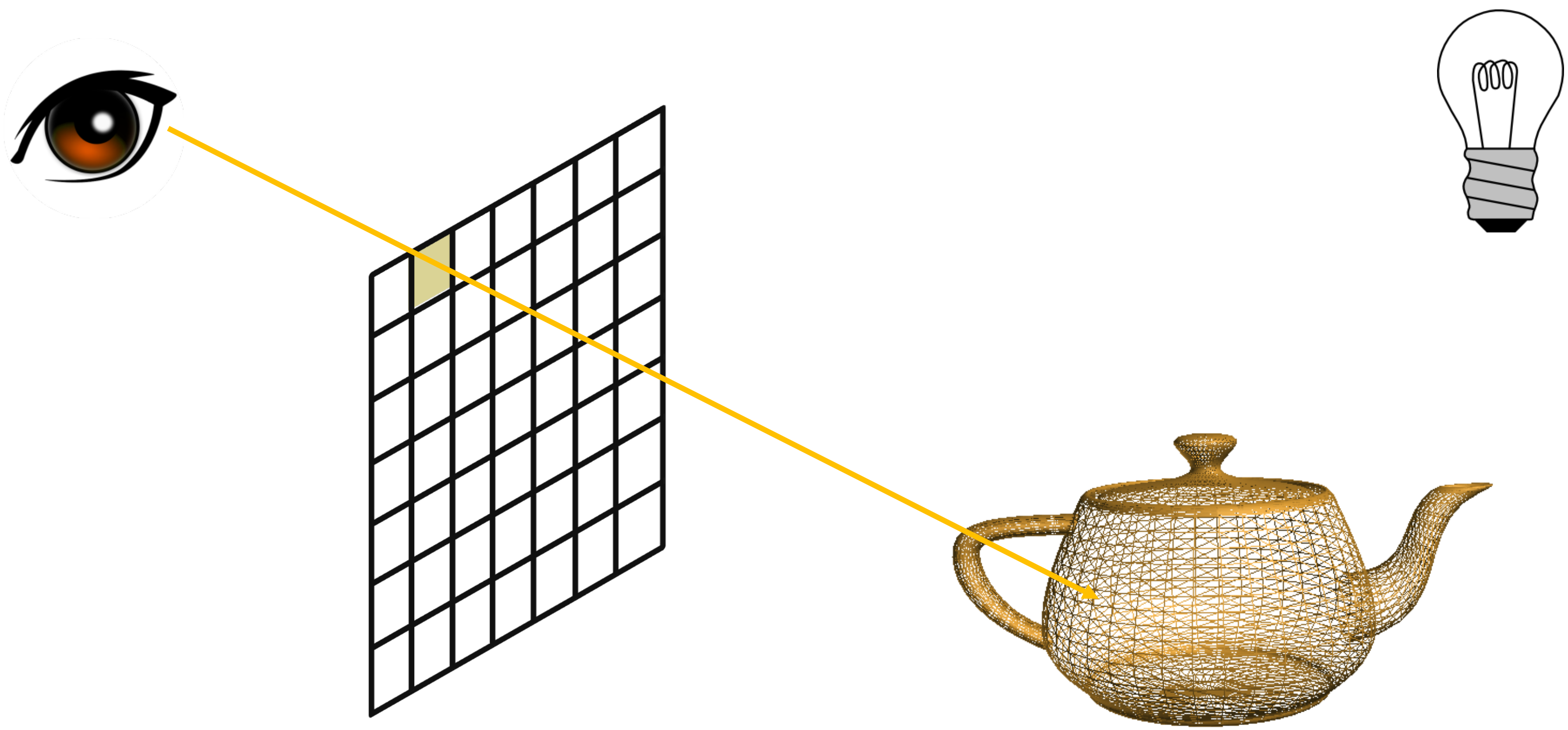
Basic Ray Tracer

- Ray intersection
  - Find the closest intersection point between the ray and objects



Basic Ray Tracer

- Shading
  - Compute the pixel color using the geometry, material, and lights at the intersection point



Basic Ray Tracer

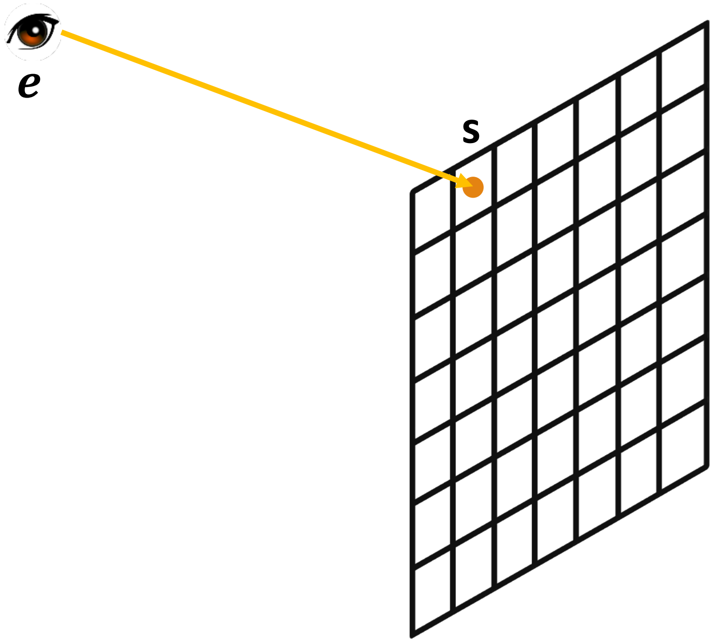
- For each pixel do
  - Compute a primary ray (viewing ray)
  - Find the closest intersection point between the ray and a scene
  - Determine a pixel color

Primary Ray Generation

- Mathematical representation for a ray
  - 3D parametric line: $\boldsymbol{p}(t) = \boldsymbol{e} + t(\boldsymbol{s} - \boldsymbol{e})$

- Properties
  - $\boldsymbol{p}(0) = \boldsymbol{e}, \quad \boldsymbol{p}(1) = \boldsymbol{s}$
  - $\boldsymbol{p}(t_1)$ is closer to the eye than $\boldsymbol{p}(t_2)$ when $0 < t_1 < t_2$
  - When $t < 0, \boldsymbol{p}(t)$ is behind the eye
  - $\boldsymbol{e}$ is a given value

- Q. How can we compute **s**?

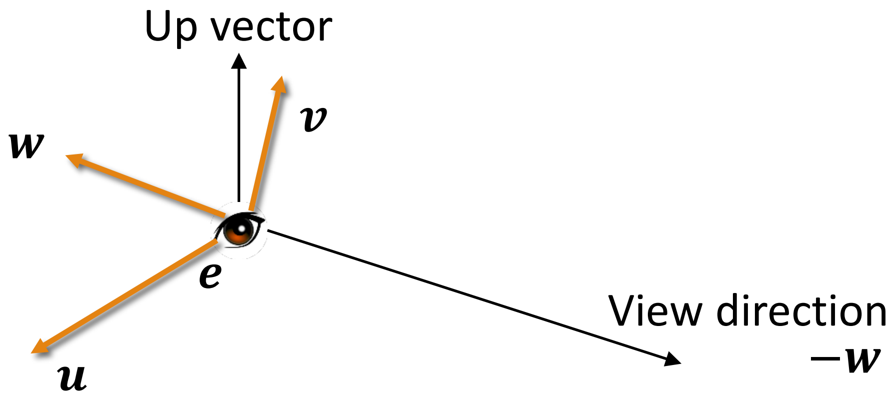


Primary Ray Generation

- Mathematical representation for a ray
  - 3D parametric line: $\boldsymbol{p}(t) = \boldsymbol{e} + t(\boldsymbol{s} - \boldsymbol{e})$

- $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ forms a right-handed coordinate system

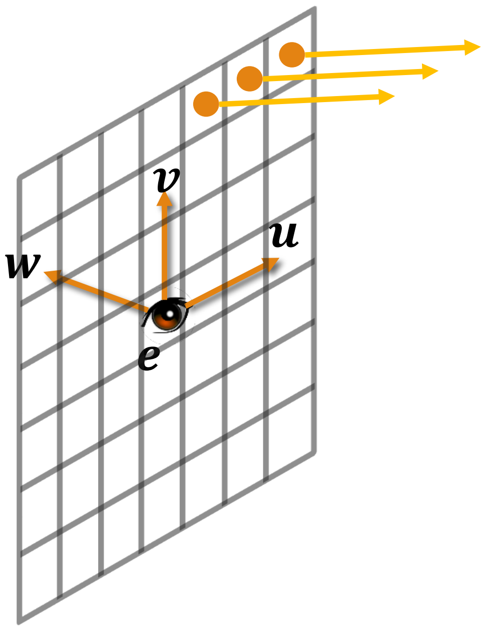
- Two kinds of views
  - Orthographic view
  - Perspective view



Orthographic Views

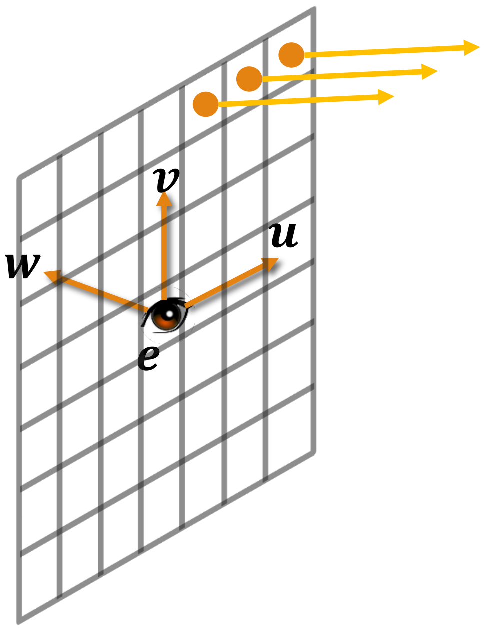
- All primary rays have the same direction, ***-w***
- The primary ray starts on the image plane defined by ***e, u, v***
- The image plane is defined with four numbers:
  - l, r: positions of left and right edges of the image plane
  - b, t: positions of bottom and top edges

- To make an image with $n_x \times n_y$
  - Pixels are spaced as the following:
    - $\frac{r-l}{n_x}$ horizontally, $\frac{t-b}{n_y}$ vertically
- Position (α, β) in the image plane is corresponding to a pixel (i, j) in the raster image:
  - $\alpha = l + \frac{(r-l)(i+0.5)}{n_x}$
  - $\beta = b + \frac{(t-b)(j+0.5)}{n_y}$
  - (α, β) are the coordinates of the pixel's position on the image plane



Orthographic Views

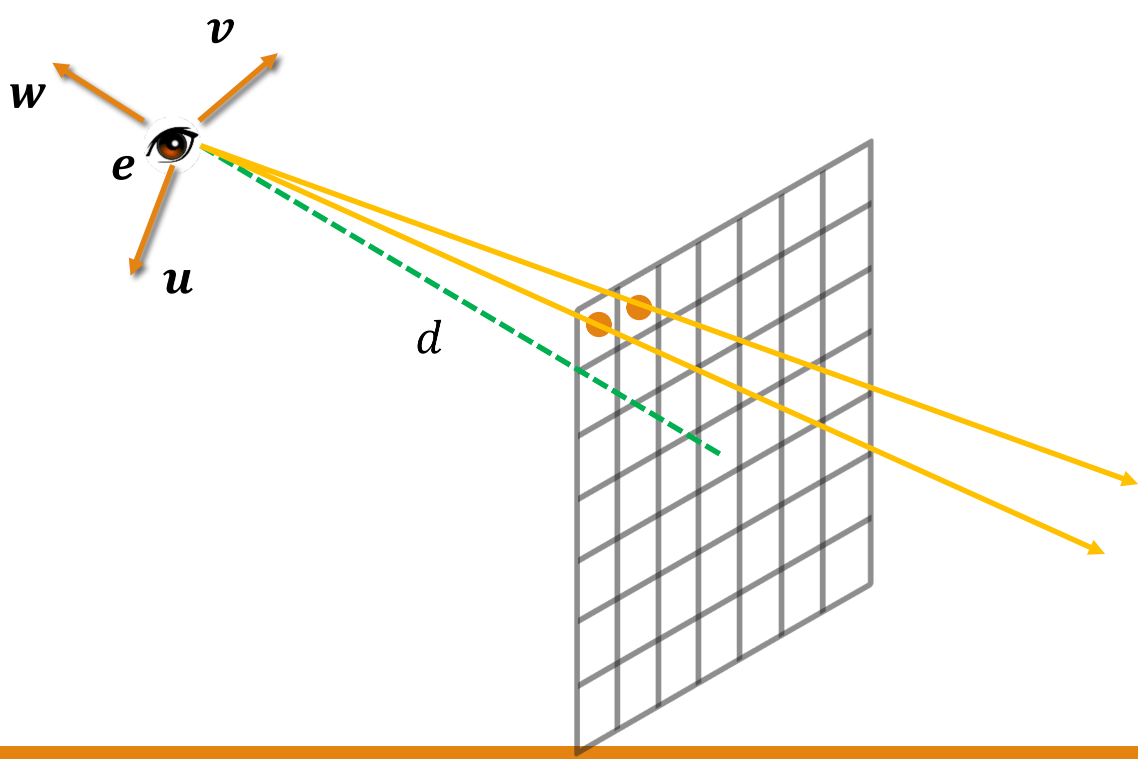
- Procedure to generate orthographic viewing rays
  - Compute α and β
    - $\alpha = l + \frac{(r-l)(i+0.5)}{n_x}$
    - $\beta = b + \frac{(t-b)(j+0.5)}{n_y}$
- $ray.direction \coloneqq -\boldsymbol{w}$
- $ray.origin \coloneqq \boldsymbol{e} + \alpha\boldsymbol{u} + \beta\boldsymbol{v}$



- Properties
  - Same direction for all rays
  - Different origins for rays

Perspective Views

- All rays have the same origin, ***e***, but have different directions
- The image plane is placed with a distance, *d*, in front of ***e***
  - *d: image plane distance (called the focal length)*
- Procedure to generate perspective viewing rays
  - Compute α and β
    - $\alpha = l + \frac{(r-l)(i+0.5)}{n_x}$
    - $\beta = b + \frac{(t-b)(j+0.5)}{n_y}$
- $ray.direction \coloneqq -d\boldsymbol{w} + \alpha\boldsymbol{u} + \beta\boldsymbol{v}$
- $ray.origin \coloneqq \boldsymbol{e}$



Intersection between Ray and Object

- Generated ray: $\mathbf{p}(\mathrm{t}) = \boldsymbol{e} + t\boldsymbol{d}$
- The next task is to find the closest intersection point between a ray and objects
  - i.e., need to find a t in the interval $[t_0, t_1]$ (e.g., $[0, +\infty]$)
- Objects
  - Sphere
  - Triangle
  - Multiple objects

Intersection between Ray and Sphere

- Ray: $\boldsymbol{p}(t) = \boldsymbol{e} + t\boldsymbol{d}$
- Implicit surface: $f(\boldsymbol{p}) = 0$
- Intersection points should satisfy both equations
  - $f(\boldsymbol{p}(t)) = f(\boldsymbol{e} + t\boldsymbol{d}) = 0$

- Let's define a sphere with center $c = (x_c, y_c, z_c)$ and radius *r*
  - $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - r^2 = 0$
  - $(\boldsymbol{p} - \boldsymbol{c}) \cdot (\boldsymbol{p} - \boldsymbol{c}) - r^2 = 0$ (vector form)
  - A point p that satisfies this equation is on the sphere

- By plug-in the parametric ray equation,
  - $(\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{c}) \cdot (\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{c}) - r^2 = 0$
  - By rearranging terms with respect to *t* (unknown value):
  - $(\boldsymbol{d} \cdot \boldsymbol{d})t^2 + 2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{c})t + (\boldsymbol{e} - \boldsymbol{c}) \cdot (\boldsymbol{e} - \boldsymbol{c}) - r^2 = 0$

Intersection between Ray and Sphere

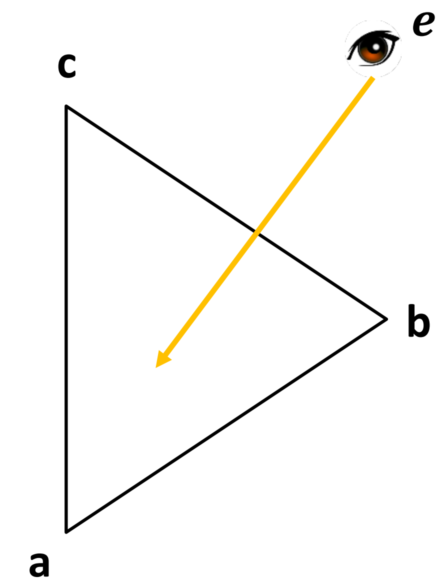
- A quadratic equation in t
  - $(\boldsymbol{d} \cdot \boldsymbol{d})t^2 + 2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{c})t + (\boldsymbol{e} - \boldsymbol{c}) \cdot (\boldsymbol{e} - \boldsymbol{c}) - r^2 = 0$

- The solutions for $at^2 + bt + c = 0$
  - $t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
  - $b^2 - 4ac$ (called discriminant)
    - When $b^2 - 4ac < 0$, there is no solution (the ray does not intersect with the sphere)
    - When $b^2 - 4ac = 0$, a solution exists (the ray touches the sphere)
    - When $b^2 - 4ac > 0$, two solutions exist (the ray enters and leaves the sphere)

- $t = \frac{-\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{c}) \pm \sqrt{\left(\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{c})\right)^2 - (\boldsymbol{d} \cdot \boldsymbol{d})\left((\boldsymbol{e} - \boldsymbol{c}) \cdot (\boldsymbol{e} - \boldsymbol{c}) - r^2\right)}}{(\boldsymbol{d} \cdot \boldsymbol{d})}$

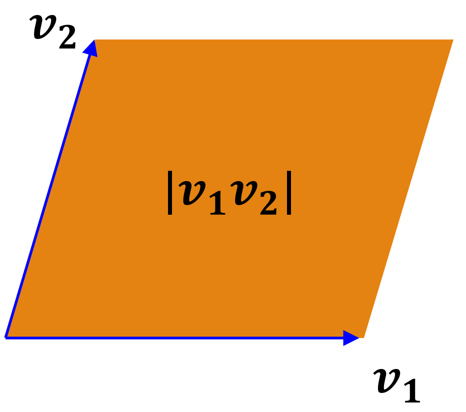
Intersection between Ray and Triangle

- Ray: $\mathbf{p}(\mathrm{t}) = \boldsymbol{e} + t\boldsymbol{d}$
- Intersection point:
  - $\boldsymbol{e} + t\boldsymbol{d} = \boldsymbol{a} + \beta(\boldsymbol{b} - \boldsymbol{a}) + \gamma(\boldsymbol{c} - \boldsymbol{a})$
- Solving the equation for t, β, γ:
  - $x_e + tx_d = x_a + \beta(x_b - x_a) + r(x_c - x_a)$
  - $y_e + ty_d = y_a + \beta(y_b - y_a) + r(y_c - y_a)$
  - $z_e + tz_d = z_a + \beta(z_b - z_a) + r(z_c - z_a)$
  - Can be rewritten:
  - $\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_e \\ y_a - y_e \\ z_a - z_e \end{bmatrix}$



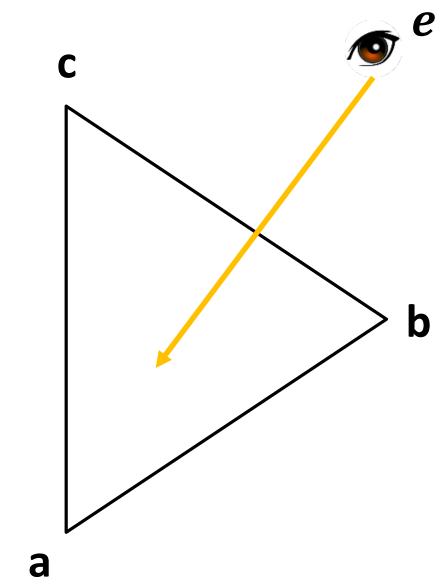
Intersection between Ray and Triangle

- Cramer's rule can be utilized to solve the 3 x 3 linear system
  - $$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$
  - $$x = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{|A|}, \quad y = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{|A|}, \quad z = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{|A|}$$
  - where $|A| = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$, $|.|$ is the determinant
  - $$|A| = a_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix} - b_1 \begin{vmatrix} a_2 & c_2 \\ a_3 & c_3 \end{vmatrix} + c_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}$$
  - $$\begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix} = b_2 c_3 - c_2 b_3$$



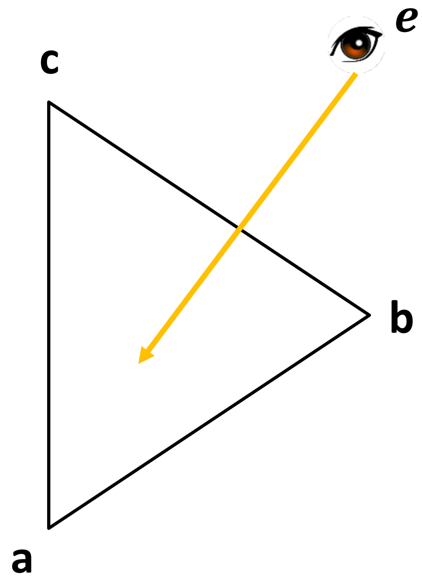
Intersection between Ray and Triangle

- $$\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_e \\ y_a - y_e \\ z_a - z_e \end{bmatrix}$$
- $$\beta = \frac{\begin{vmatrix} x_a - x_e & x_a - x_c & x_d \\ y_a - y_e & y_a - y_c & y_d \\ z_a - z_e & z_a - z_c & z_d \end{vmatrix}}{|A|}$$
- $$\gamma = \frac{\begin{vmatrix} x_a - x_b & x_a - x_e & x_d \\ y_a - y_b & y_a - y_e & y_d \\ z_a - z_b & z_a - z_e & z_d \end{vmatrix}}{|A|}$$
- $$t = \frac{\begin{vmatrix} x_a - x_b & x_a - x_c & x_a - x_e \\ y_a - y_b & y_a - y_c & y_a - y_e \\ z_a - z_b & z_a - z_c & z_a - z_e \end{vmatrix}}{|A|}$$
- where $|A| = \begin{vmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{vmatrix}$



Intersection between Ray and Triangle

- Procedure (with early termination) for finding the intersection:
  - Input: a ray, vertex a, b, c, interval $[t_0, t_1]$
  - Compute t
  - If $(t < t_0)$ or $(t > t_1)$ then
    - return false
  - Compute γ
  - If $(\gamma < 0)$ or $(\gamma > 1)$ then
    - return false
  - Compute β
  - If $(\beta < 0)$ or $(\beta > 1 - \gamma)$ then
    - return false
  - return true



Intersection between Ray and Objects

- Procedure for finding the *closest* intersection:
  - hit = false
  - For each object o do
    - If (o is intersected with the ray at a parameter t and $t \in [t_0, t_1]$) then
      - hit = true
      - store some information (e.g., o, normal, etc.) for shading
      - $t_1 = t$
  - return hit

Basic Ray Tracer

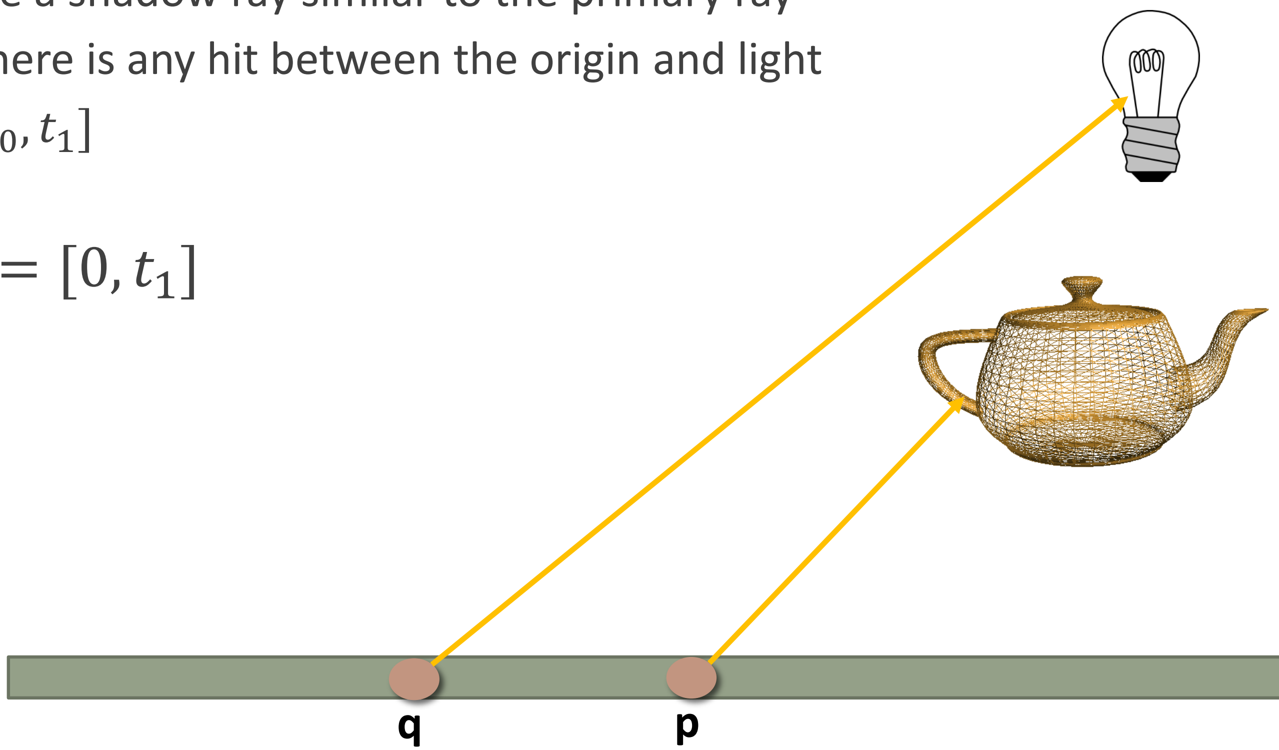
- For each pixel do
  - Compute a primary ray (viewing ray)
  - Find the closest intersection point between the ray and a scene
  - Determine a pixel color
    - e.g., we can apply the Phong illumination model here

Basic Ray Tracer

- For each pixel do
  - Compute a primary ray (viewing ray)
  - If (ray intersects an object with $t \in [0, \infty)$) then
    - Compute a hit record that contains some information (normal, materials, …)
    - Evaluate an illumination model and set a pixel color
  - Else
    - Set a pixel color to background color

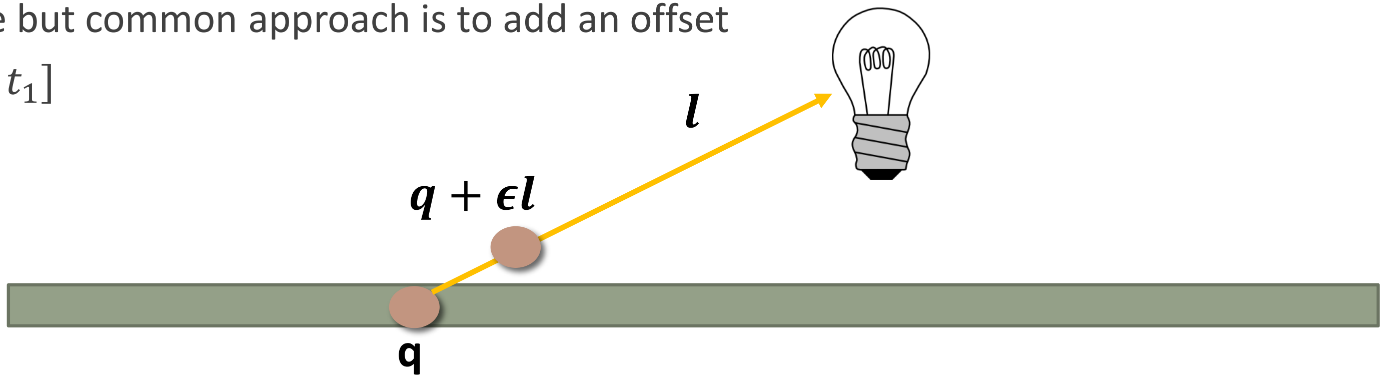
Shadows

- Assume there are two intersection points, p and q
  - p is in shadow, but q is not in shadow
- Rays to determine whether or not the point is in shadow are *shadow rays*
  - Generate a shadow ray similar to the primary ray
  - Check there is any hit between the origin and light
    - $t = [t_0, t_1]$
  - e.g., $t = [0, t_1]$



Shadows

- Assume there are two intersection points, p and q
  - p is in shadow, but q is not in shadow
- Rays to determine whether or not the point is in shadow are *shadow rays*
  - Generate a shadow ray similar to the primary ray
  - Check there is any hit between the origin and light
    - $t = [t_0, t_1]$
  - Due to numerical issues, the shadow ray can intersect the surface on which the point lies
    - A naïve but common approach is to add an offset
    - $t = [\epsilon, t_1]$



Shadows

- Pseudocode to implement shadows (based on the Phong illumination)
- Input: a ray $\boldsymbol{e} + t\boldsymbol{d}$, $[t_0 = 0, t_1 = \infty]$
- If (there is a hit between the ray and objects) then
  - $\boldsymbol{p} = \boldsymbol{e} + t\boldsymbol{d}$ // p is the closest intersection from e
  - $\boldsymbol{color}\ \boldsymbol{c} = (\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})$
  - If (there is no hit between the shadow ray and a light) then
    - $\boldsymbol{c} = \boldsymbol{c} + \boldsymbol{k_a L_a} + \boldsymbol{L_d k_d} \max(0, \boldsymbol{n} \cdot \boldsymbol{l}) + \boldsymbol{L_s k_s} \max(0, \boldsymbol{r} \cdot \boldsymbol{v})^s$
  - return c
- Else
  - return background color

Some History of Ray Tracing

- Rene Descartes (1637) used ray tracing to explain the phenomena of rainbow
- In rendering, the ray casting was presented by Arthur Appel (1968)
  - Ray casting (discussed so far) tends to be interchangeable to ray tracing
  - Ray tracing generates additional rays (e.g., secondary rays) to simulate global illumination effects
  - Ray tracing becomes popular due to the Whitted's paper (1980)
    - T. Whitted. An improved illumination model for shading display. Communications of the ACM, 23(6):343–349, 1980

Further Readings

- Chapter 4