

Ray Casting



Some Slides/Images adapted from Marschner and Shirley

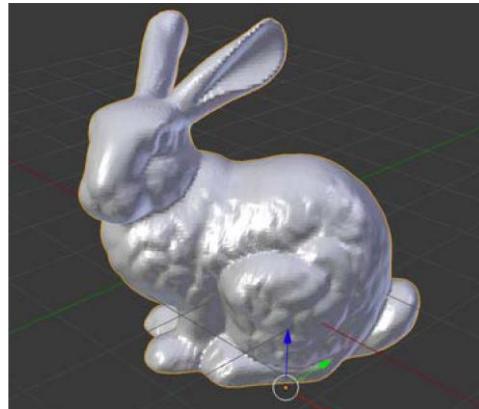
Topics

- Rasterization vs. Ray Casting
- The Ray Casting Algorithm
- Introduction to Rays
- The Camera
- Ray-Object Intersection
 - Ray-Plane Intersection
 - Ray-Sphere Intersection
 - Ray-Triangle Intersection



Photography

Input:



Objects



Lights



Camera

Output:

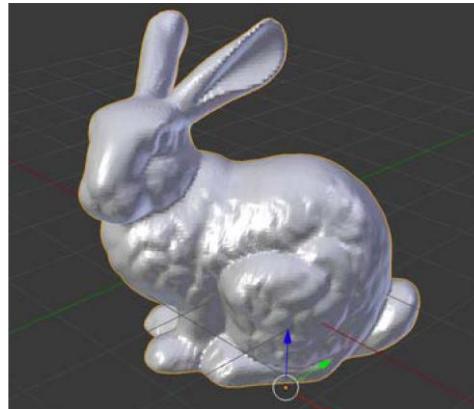


$$I(x, y)$$

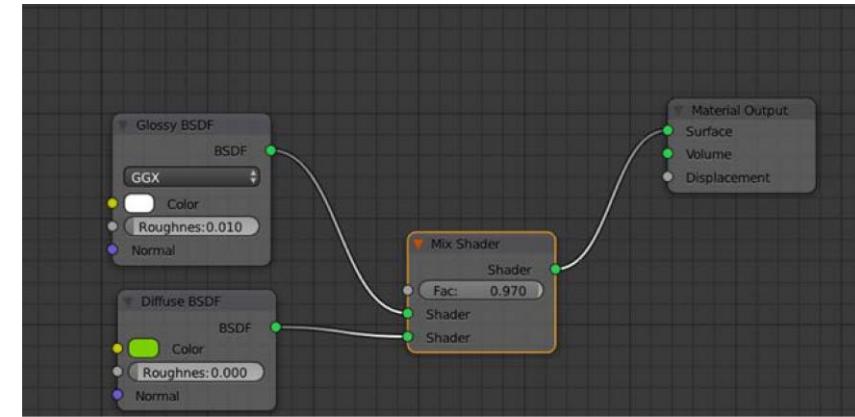


Rendering

Input:

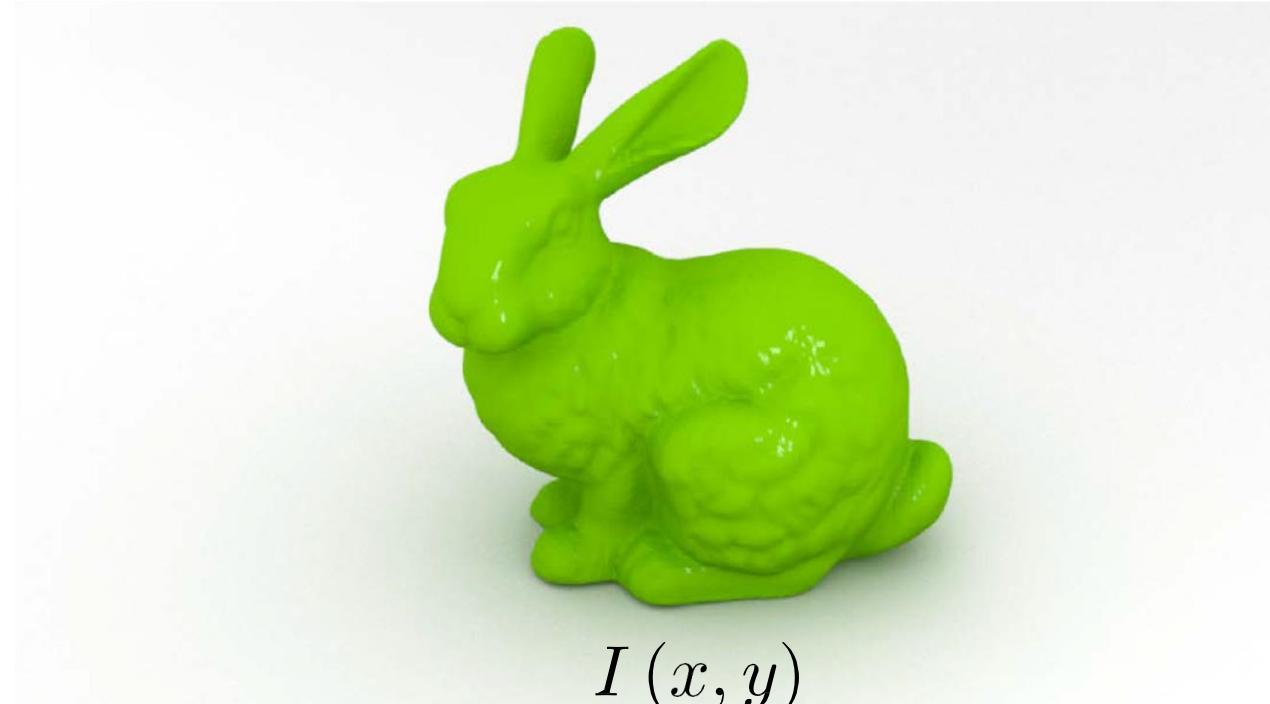


Objects



Materials

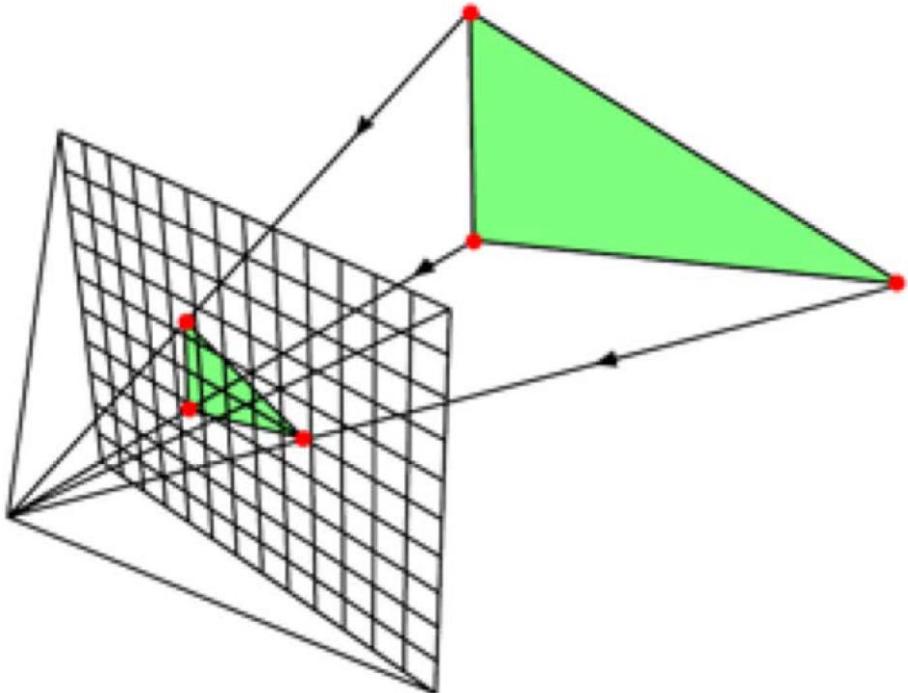
Output:



$$I(x, y)$$

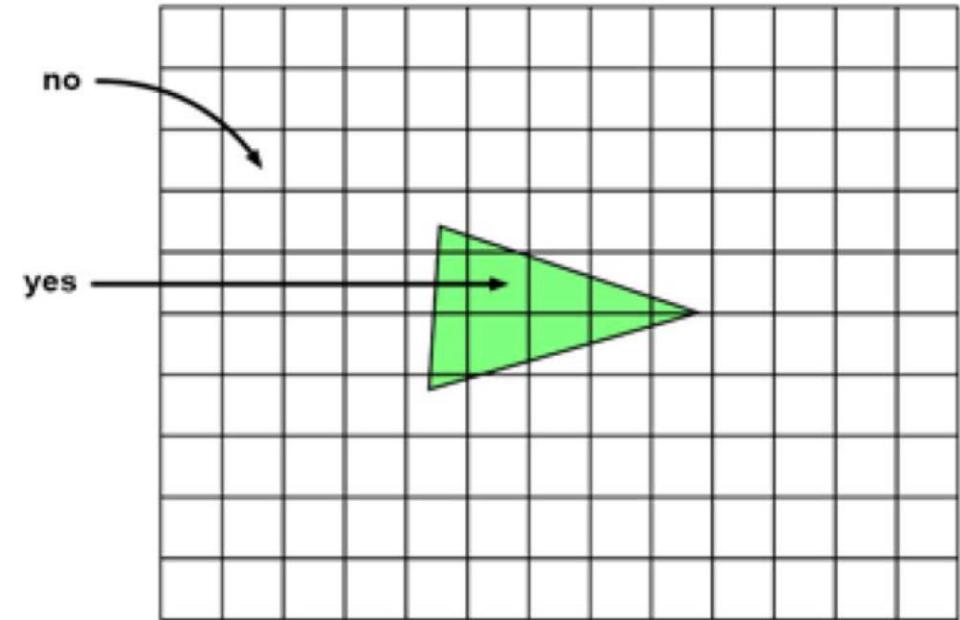


Rasterization



1. Project Vertices to Image Plane

© www.scratchapixel.com



2. Turn on pixels inside triangle

zvarownik
13878049a9f142c2b036e49f198e2149

- WARIOR_GAMING_57
- YZx_Vulka
- danielek185
- zvarownik

NW 285 300 330 345 N 15 30 60 75 NE E

danielek185



0	5
0	0
0	0



△ Przytrzymaj



Rasterization

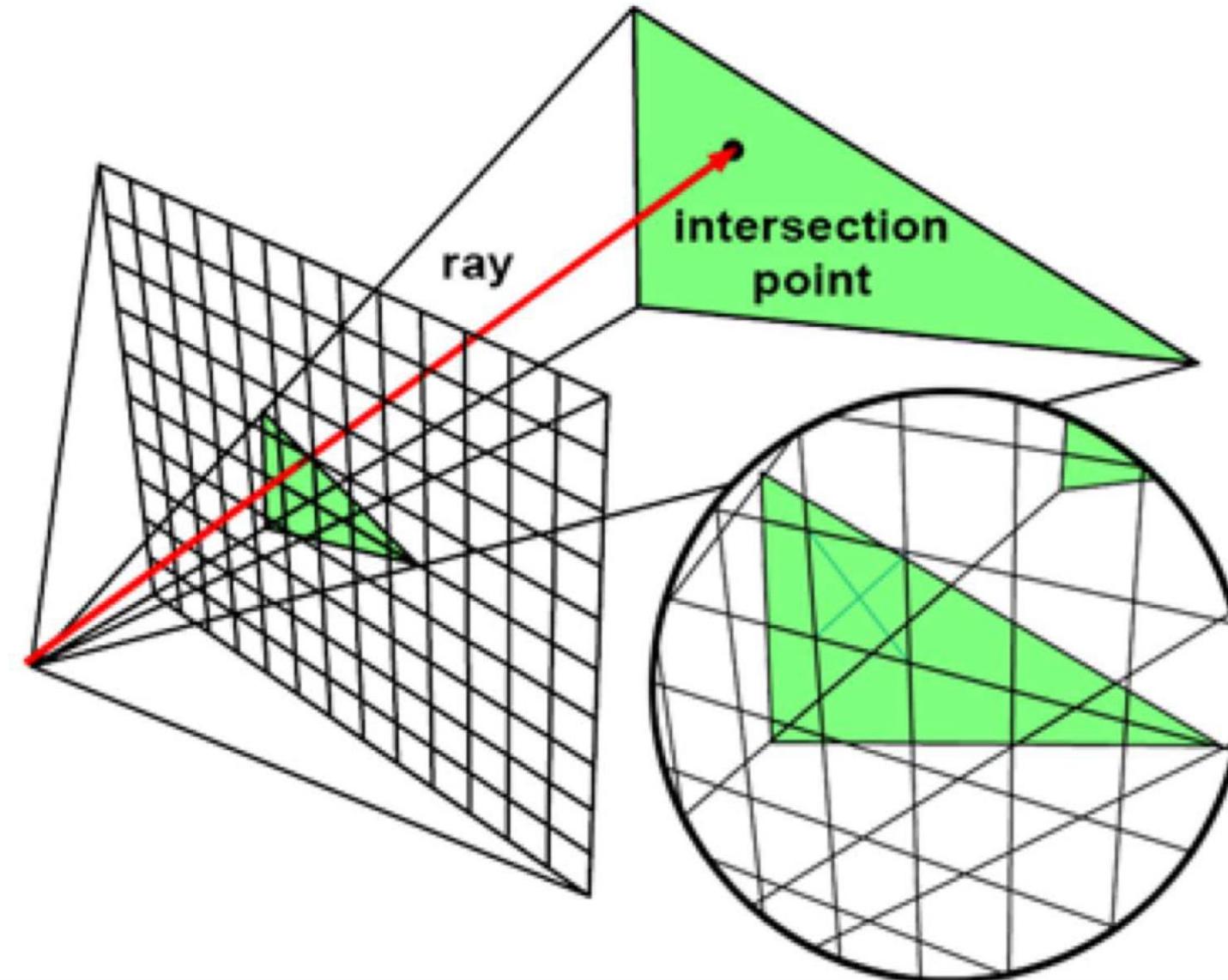
```
for each scene object {  
    for each image pixel{  
        if (object affects pixel) {  
            do something  
        }  
    }  
}
```



operations can be done
quickly on the GPU!



Ray Casting





Ray Casting

```
for each image pixel {  
    generate a ray  
    for each scene object {  
        if (ray intersects object) {  
            set pixel color  
        }  
    }  
}
```



Ray Casting vs. Rasterization

Ray Casting

```
for each image pixel {  
    for each scene object {  
        ...  
    }  
}
```

Rasterization

```
for each scene object {  
    for each image pixel {  
        ...  
    }  
}
```



Basic Components of Ray Casting

- Ray
- Camera
- Intersection Tests

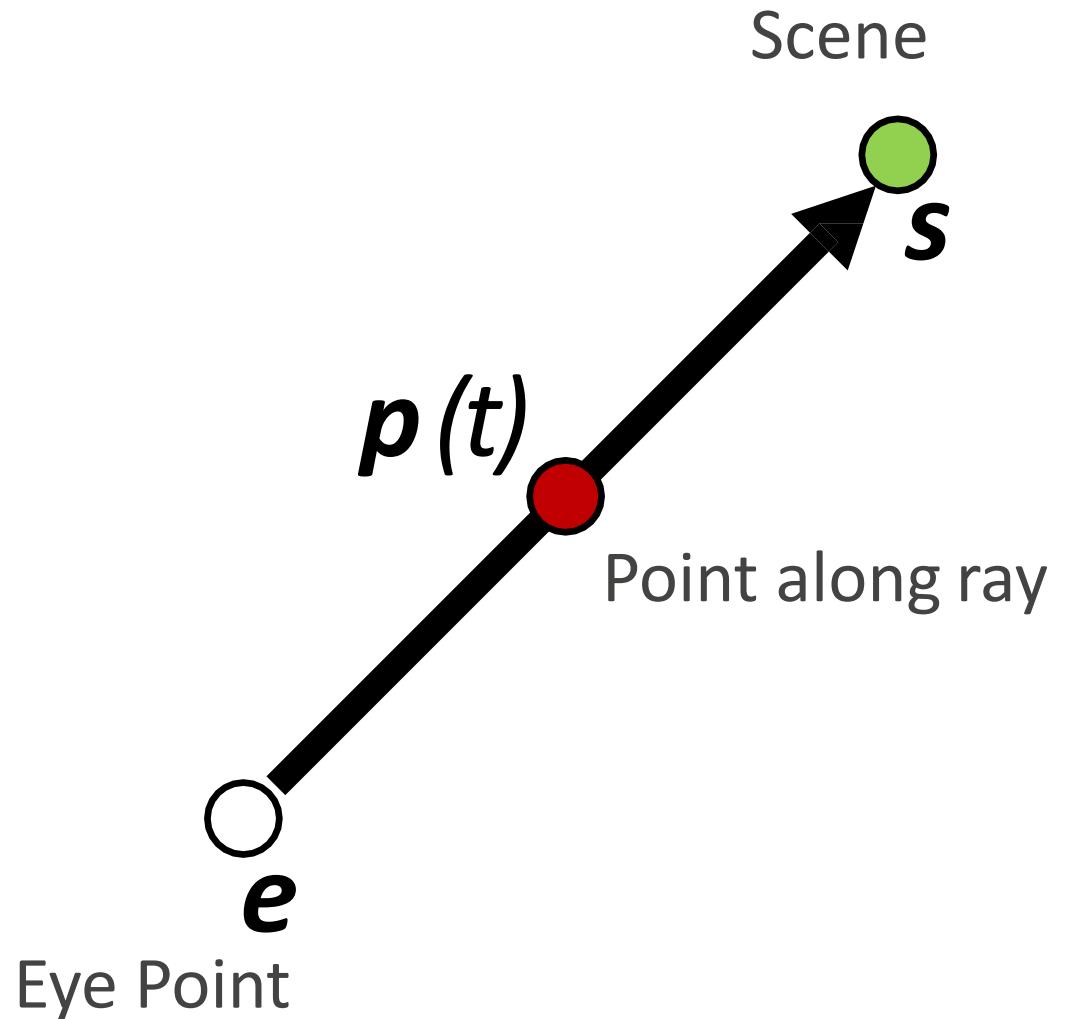


Ray

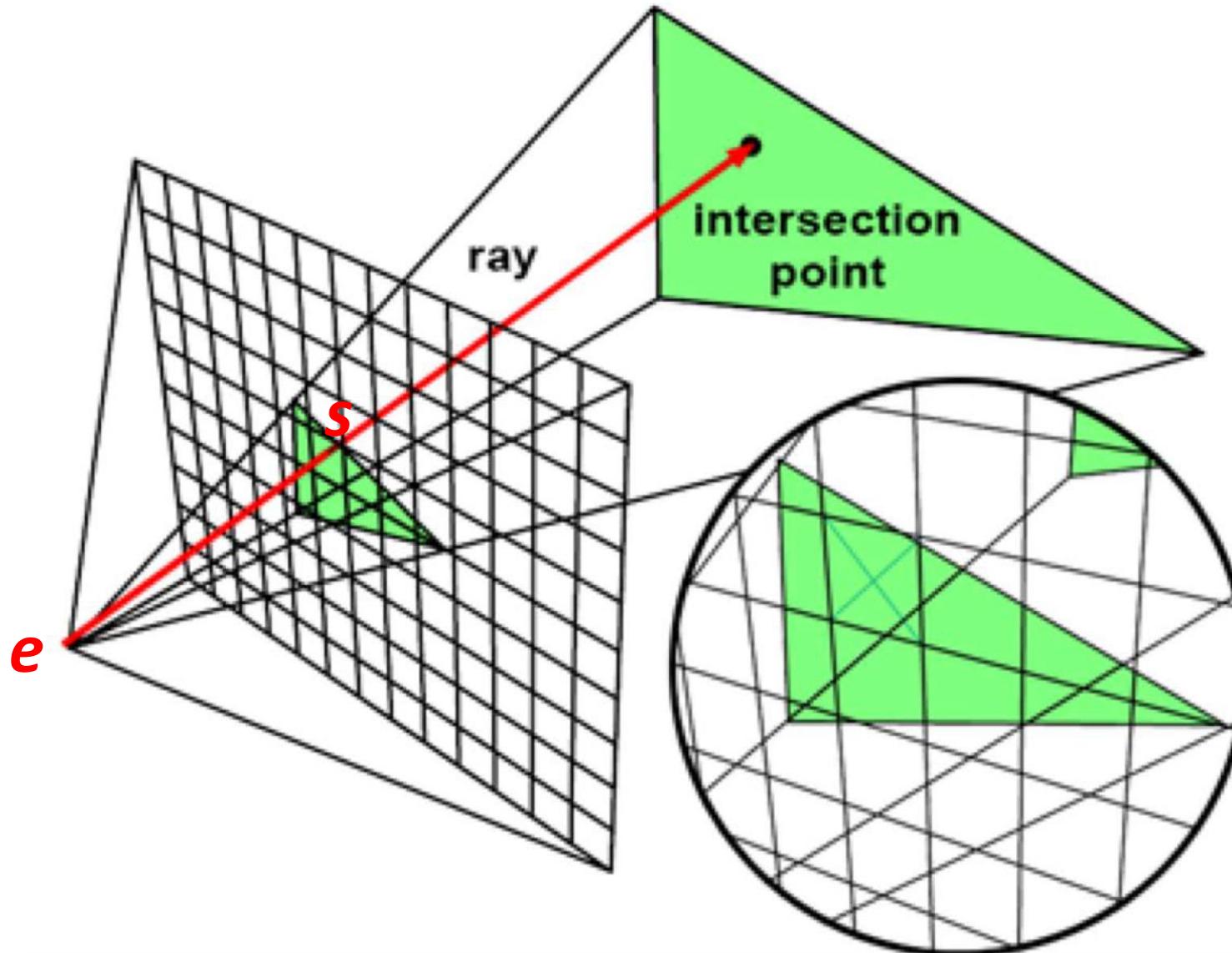
$$p(t) = e + t(s - e)$$

$$p(0) = e$$

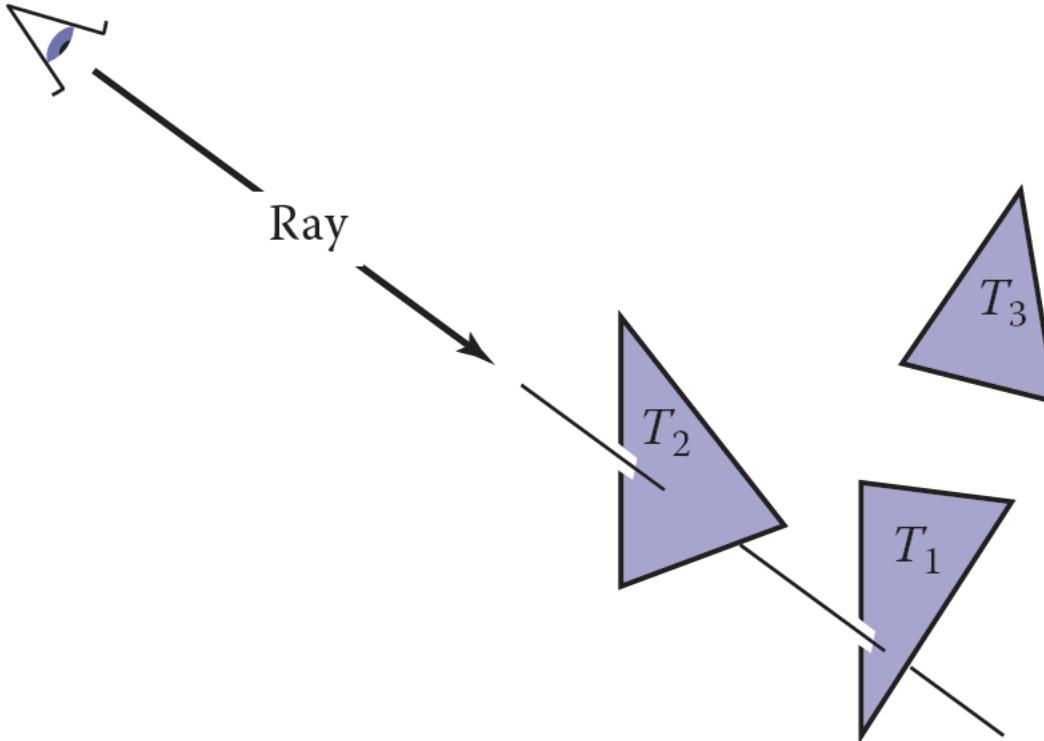
$$p(1) = s$$



Ray Casting



Ray Casting

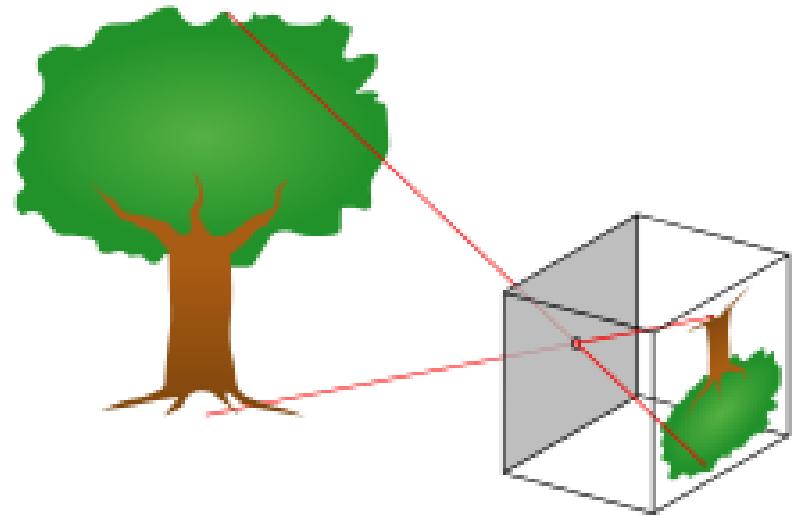


Basic Components of Ray Casting

- Ray
- Camera
- Intersection Tests

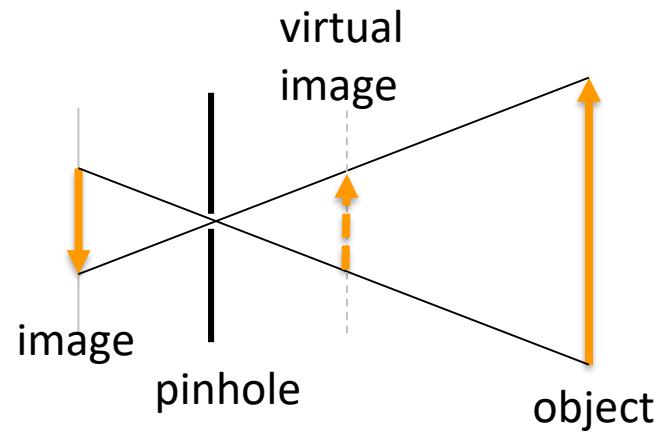


Camera model: camera obscura

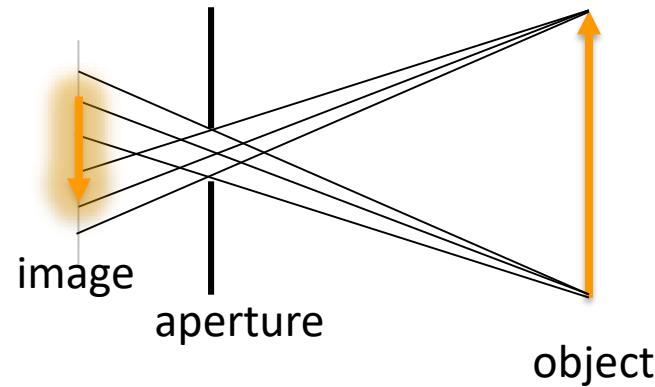


Camera model

Ideal pinhole camera

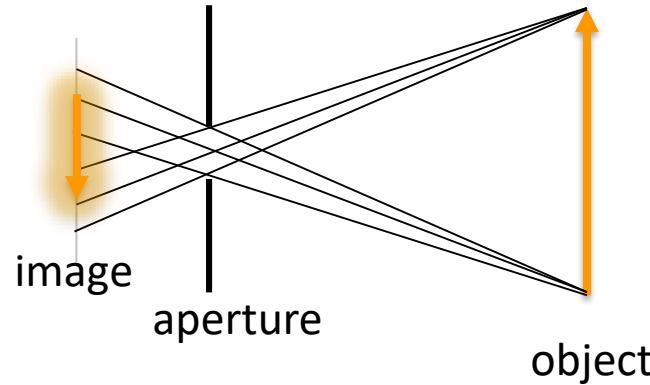


Real pinhole camera

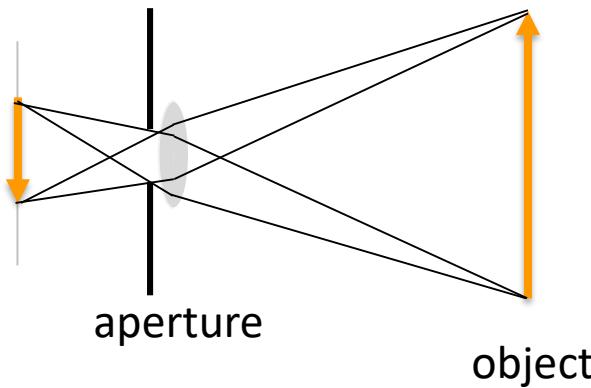


Camera model

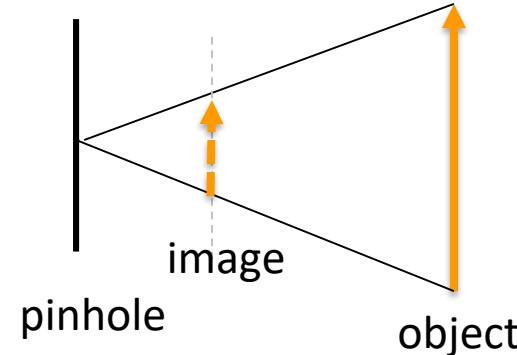
Real pinhole camera



Camera with a lens



Virtual camera

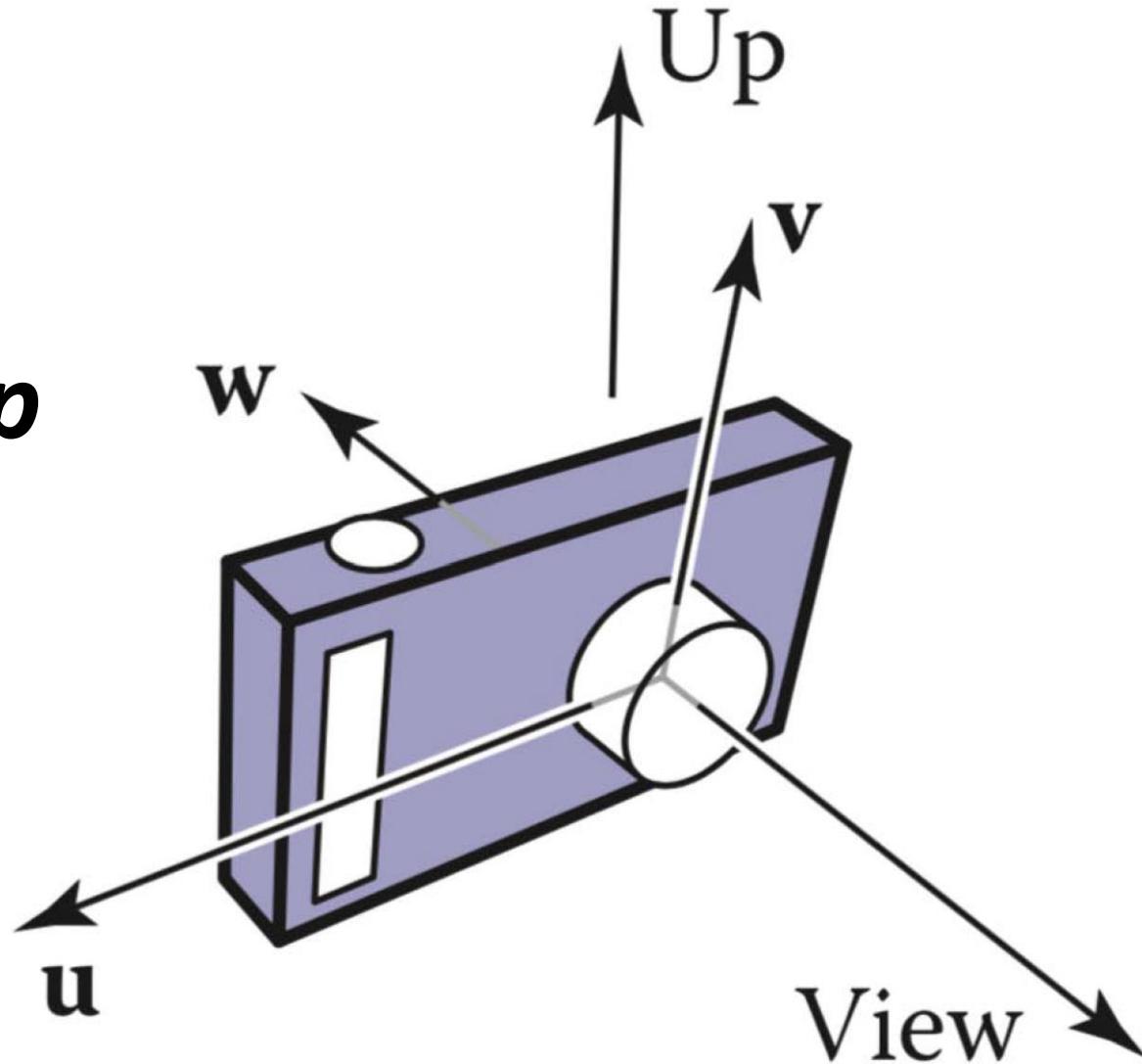


Camera

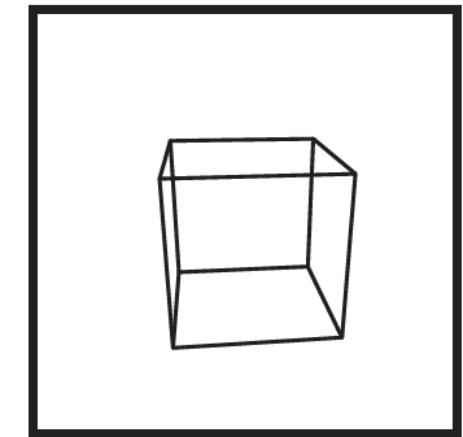
$$w = -\mathbf{view}$$

$$u = \mathbf{view} \times \mathbf{up}$$

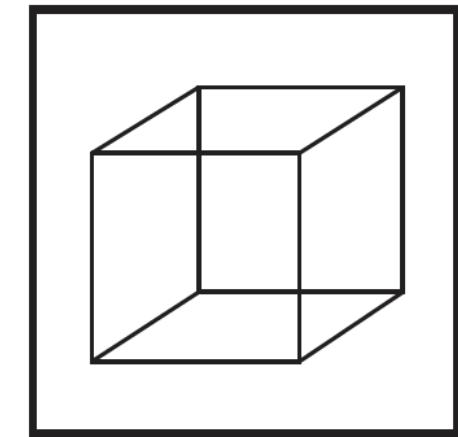
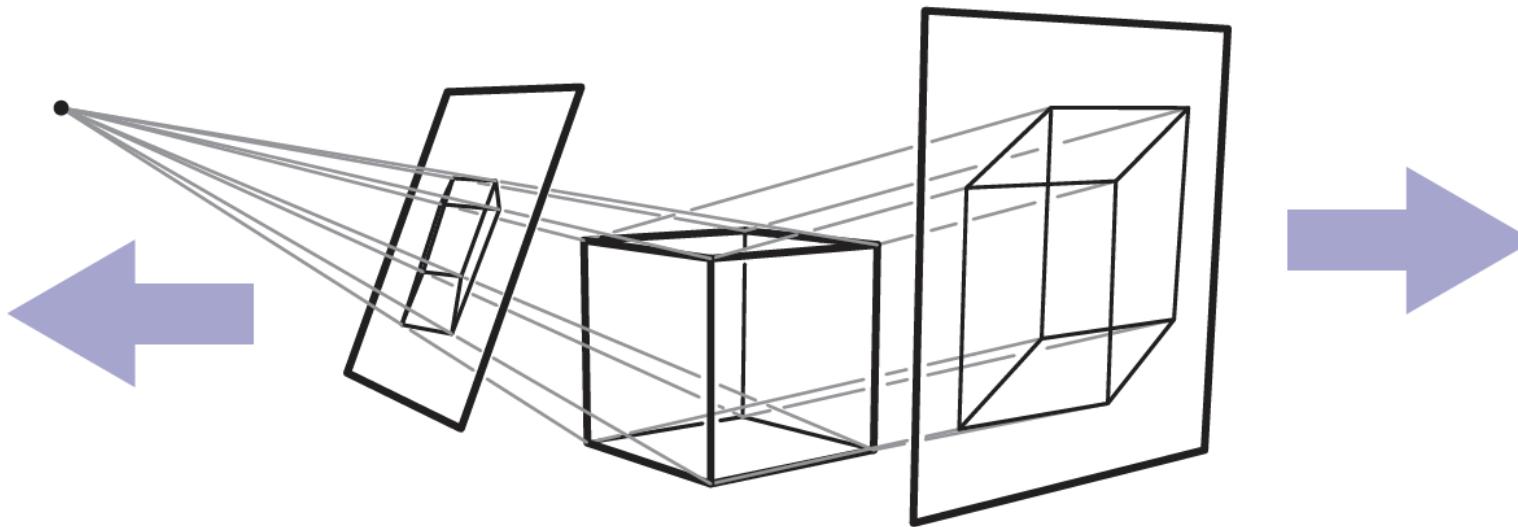
$$v = w \times u$$



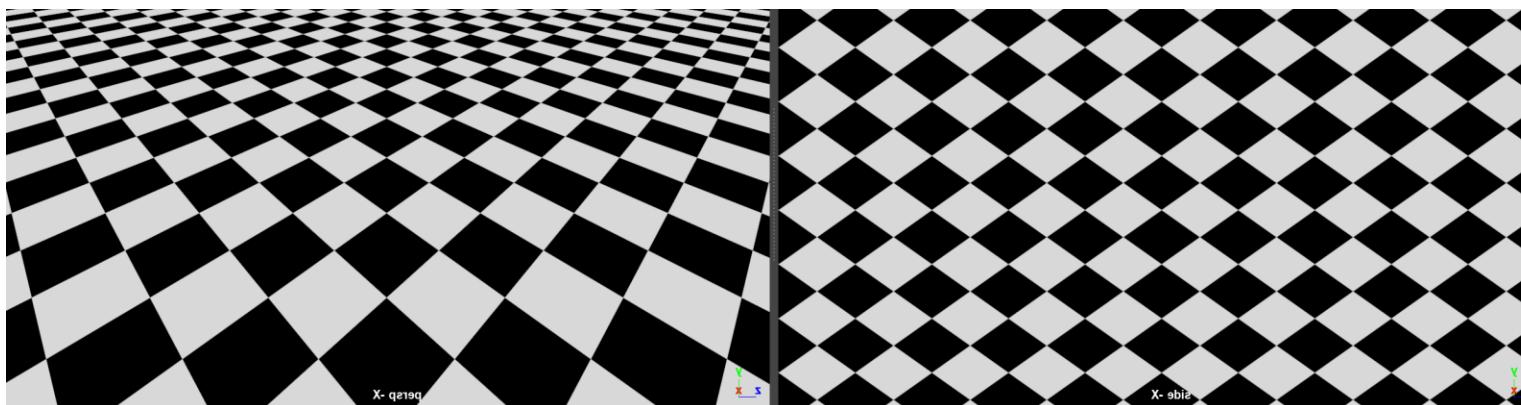
Orthographic v Perspective Projection



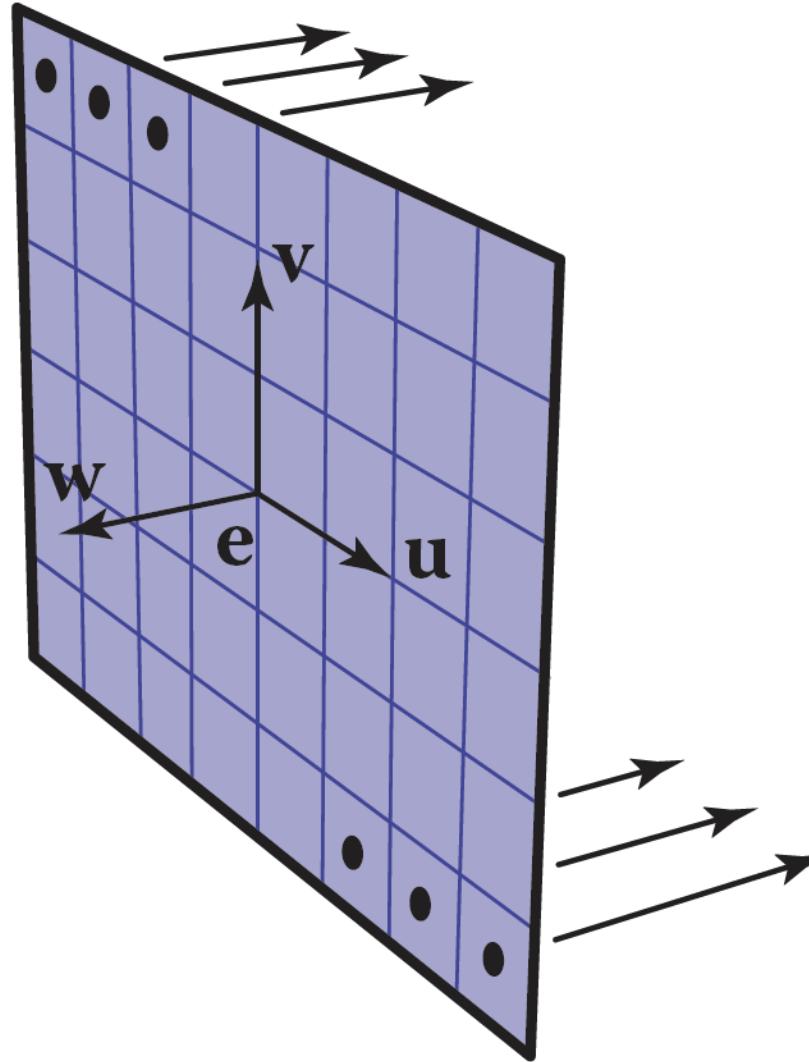
Perspective



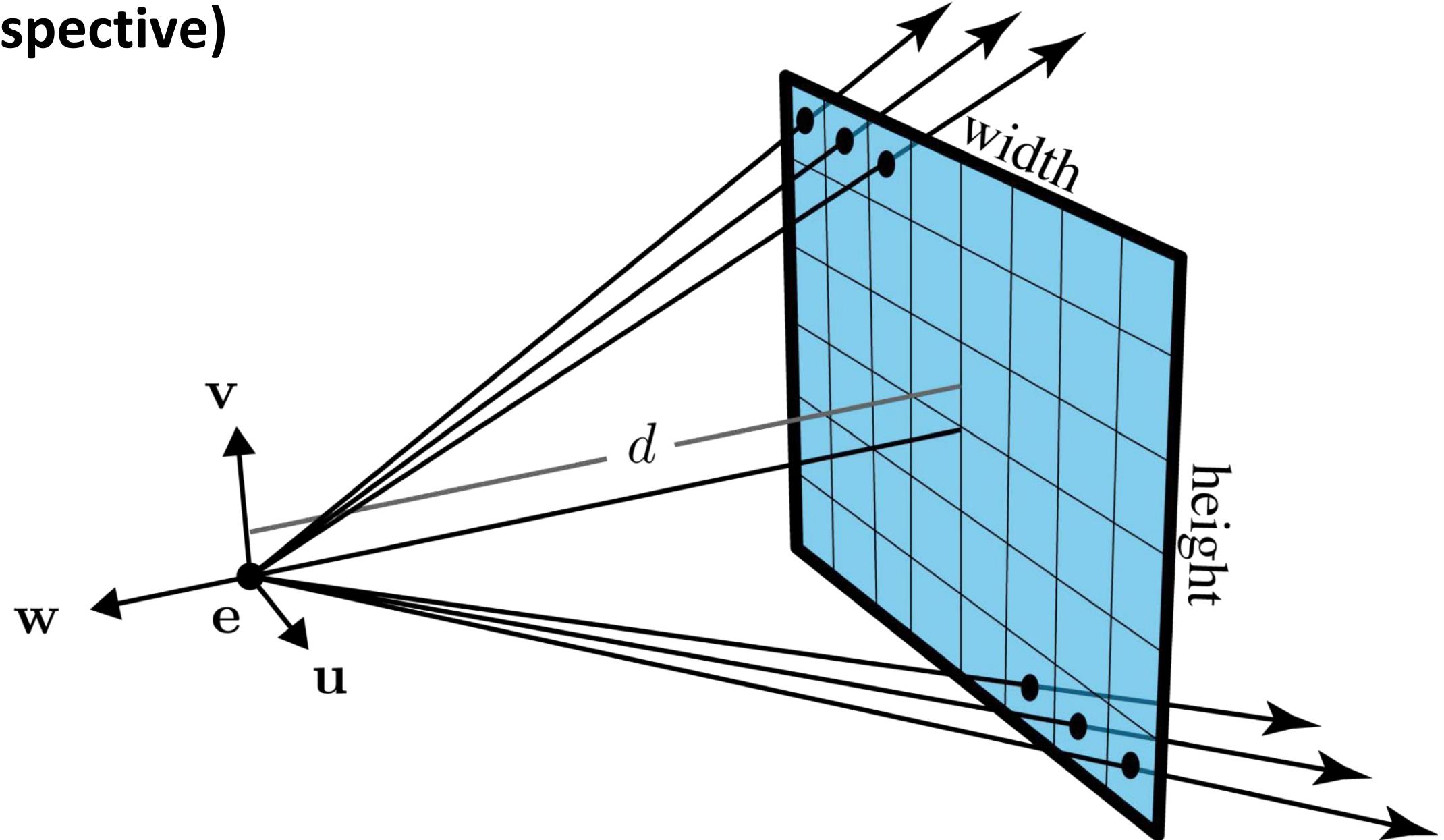
Oblique



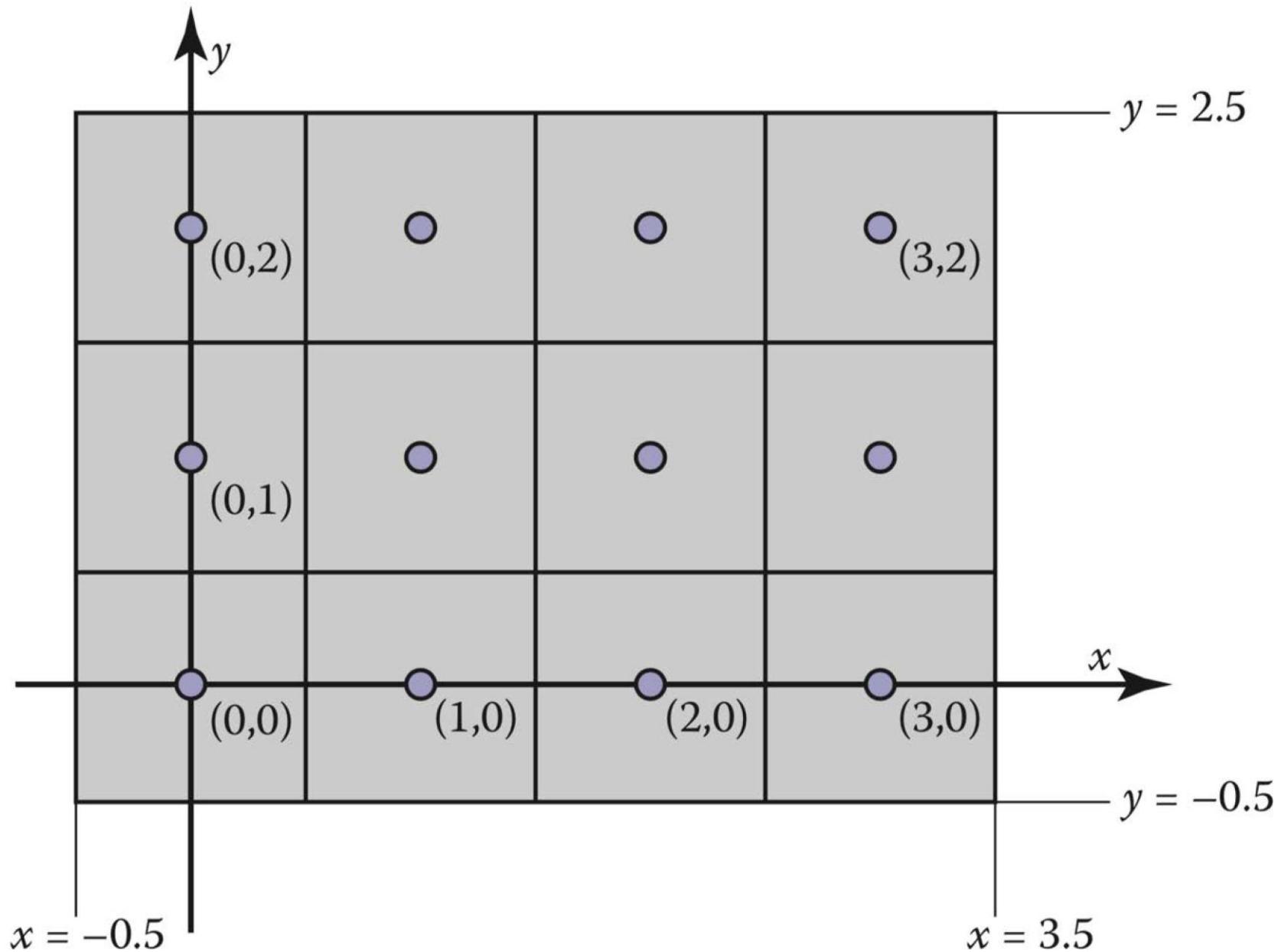
Generating Rays (Orthographic)

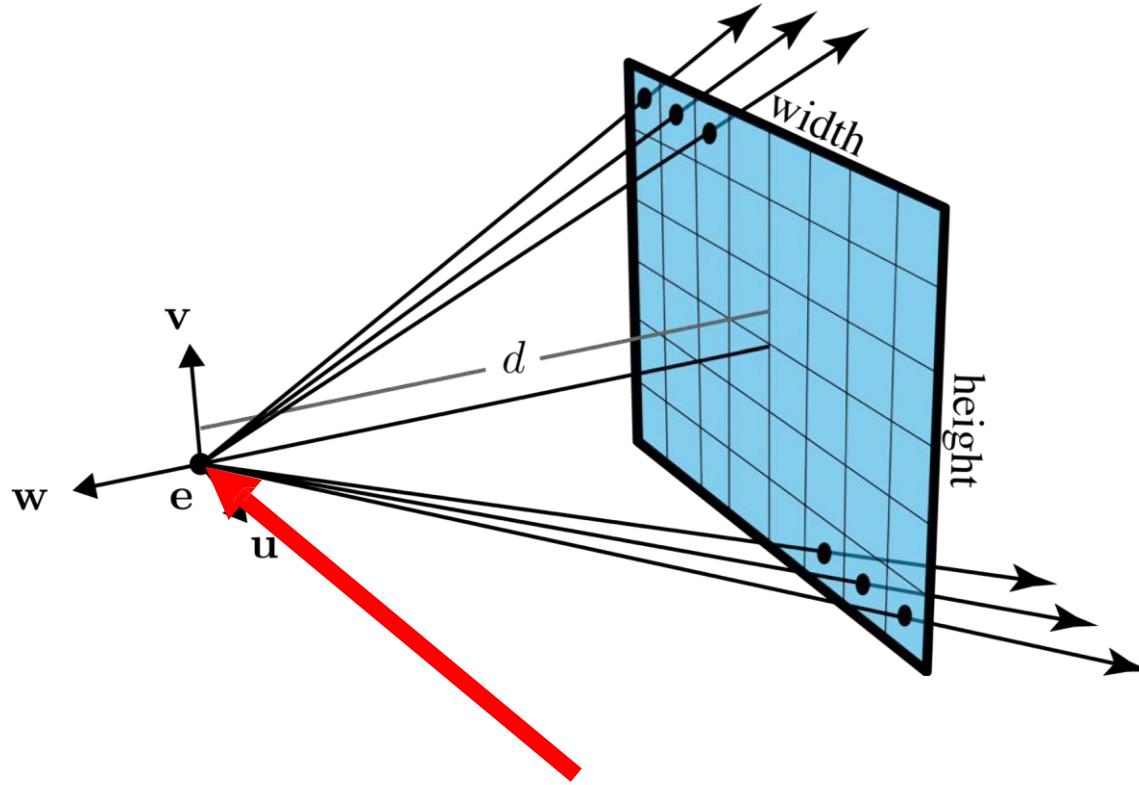


Generating Rays (Perspective)



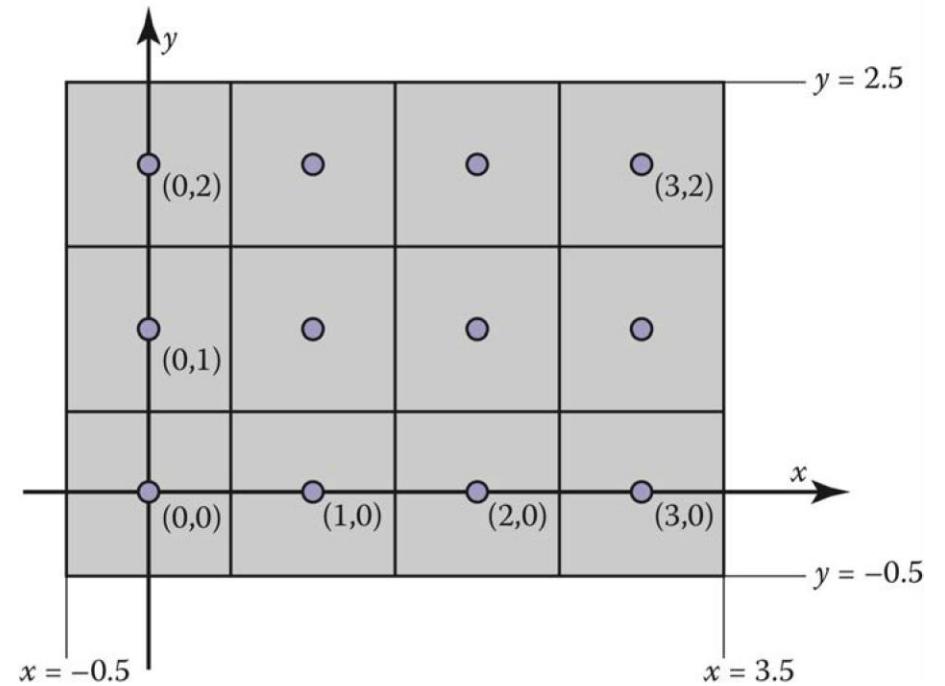
Recall: Standard Pixel Coordinate System

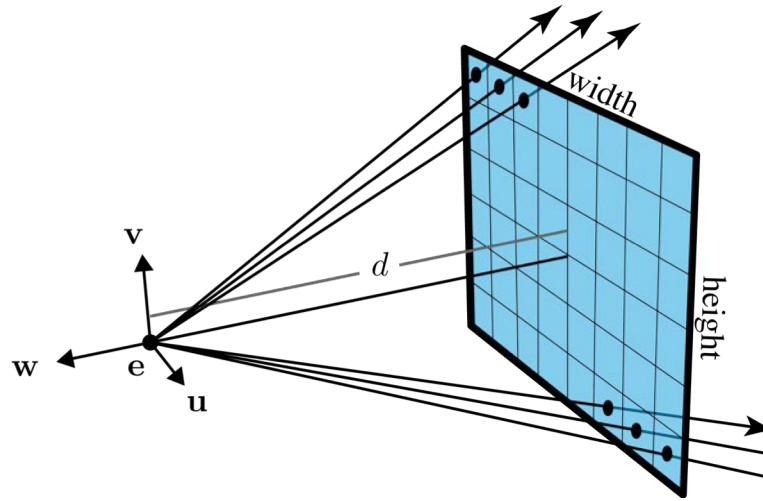




Origin of camera frame (the eye)

What are the (u, v) coordinates for pixel (i, j) in the camera frame?





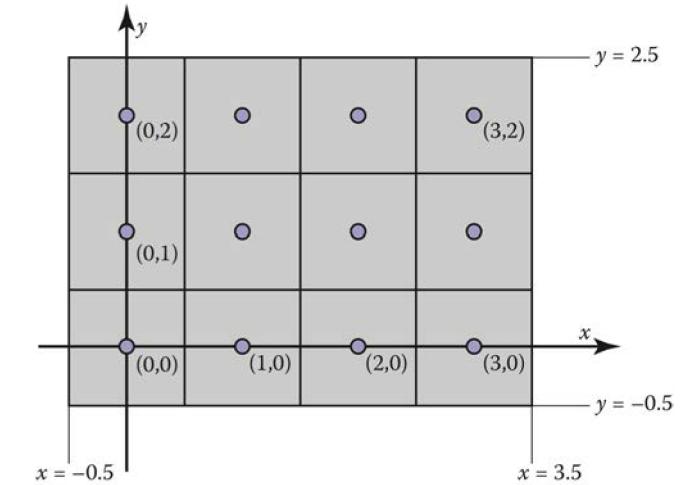
Camera space

Bottom Left Corner $(i, j) : ?$

Top Right Corner $(i, j) : ?$

Bottom Left Corner $(u, v) : ?$

Top Right Corner $(u, v) : ?$



Pixel space



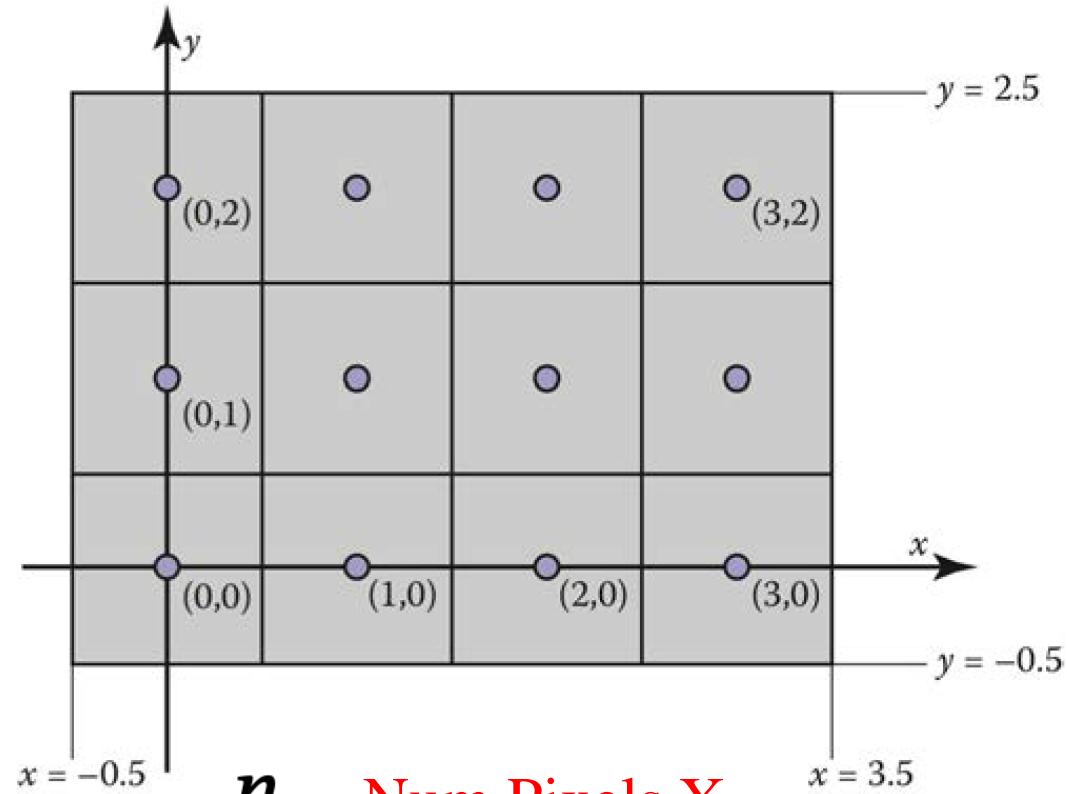
Bottom Left Corner (i, j) : $(-\frac{1}{2}, -\frac{1}{2})$

Top Right Corner (i, j) : $(n_x - \frac{1}{2}, n_y - \frac{1}{2})$

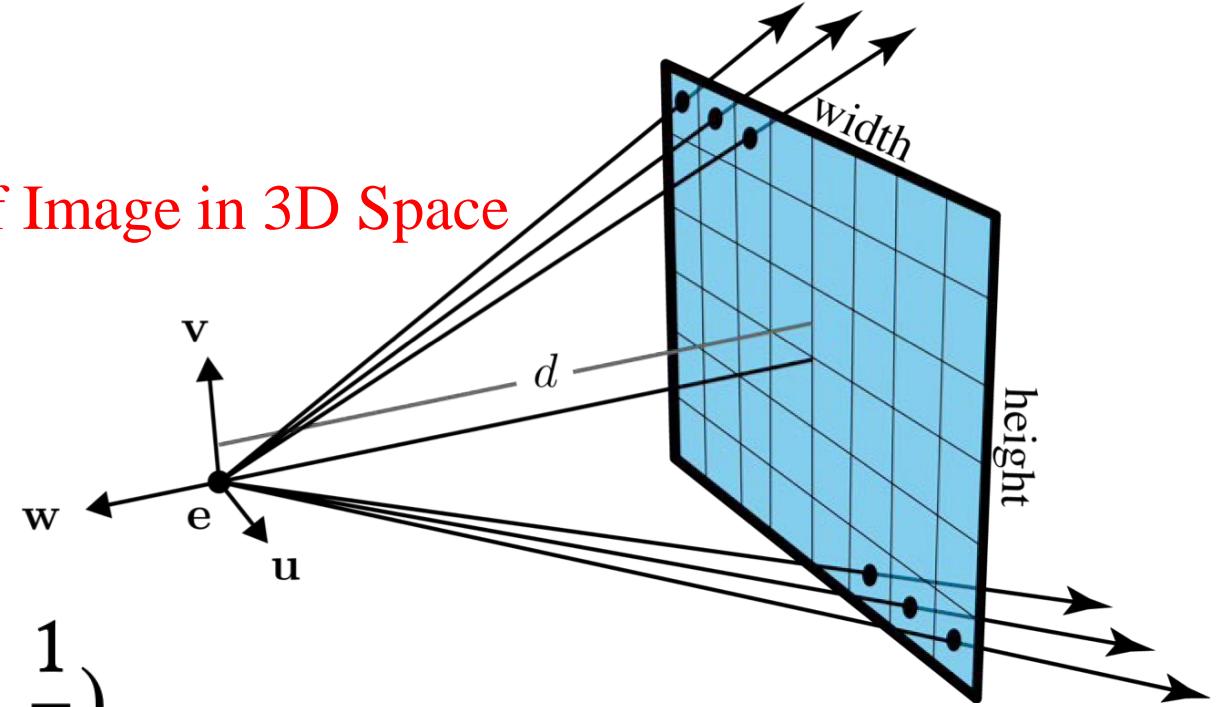
Bottom Left Corner (u, v) :

Top Right Corner (u, v) :

n_y Num Pixels Y



Physical Width of Image in 3D Space



Physical Height of Image in 3D Space

$$\text{Bottom Left Corner } (i, j) : \left(-\frac{1}{2}, -\frac{1}{2} \right)$$

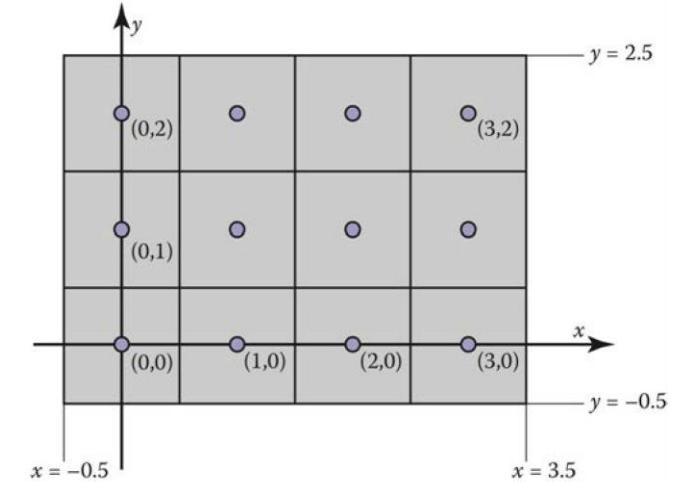
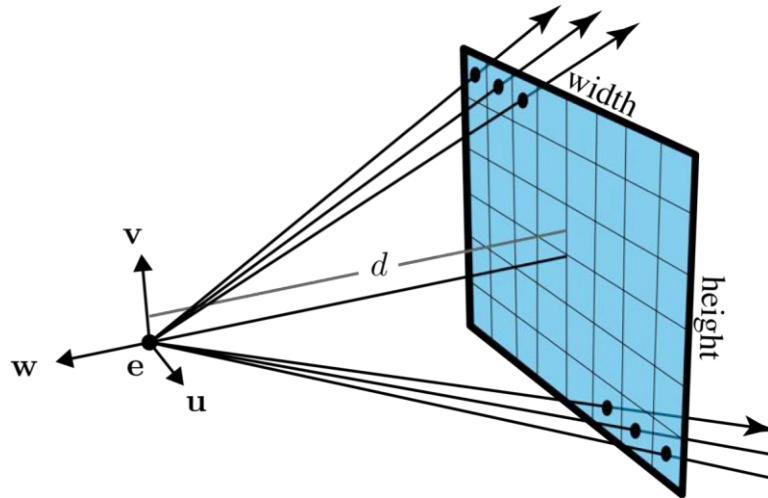
$$\text{Top Right Corner } (i, j) : \left(n_x - \frac{1}{2}, n_y - \frac{1}{2} \right)$$

$$\text{Bottom Left Corner } (u, v) : \left(-\frac{\text{width}}{2}, -\frac{\text{height}}{2} \right)$$

$$\text{Top Right Corner } (u, v) : \left(\frac{\text{width}}{2}, \frac{\text{height}}{2} \right)$$



pixel at position (i, j) in the raster image has the position:



Physical Width of Image in 3D Space

$$u = \frac{\text{width}}{n_x} \cdot \left(i + \frac{1}{2} \right) - \frac{\text{width}}{2}$$

Num Pixels X $\rightarrow n_x$

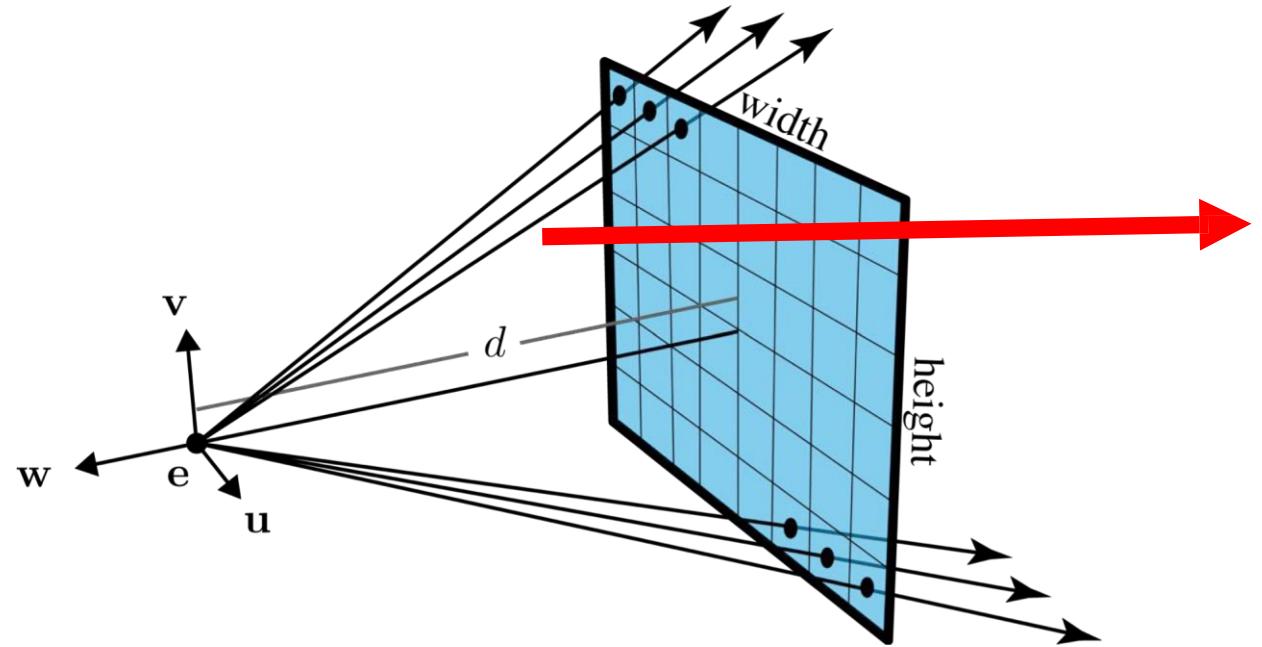
Physical Height of Image in 3D Space

$$v = \frac{\text{height}}{n_y} \cdot \left(j + \frac{1}{2} \right) - \frac{\text{height}}{2}$$

Num Pixels Y $\rightarrow n_y$



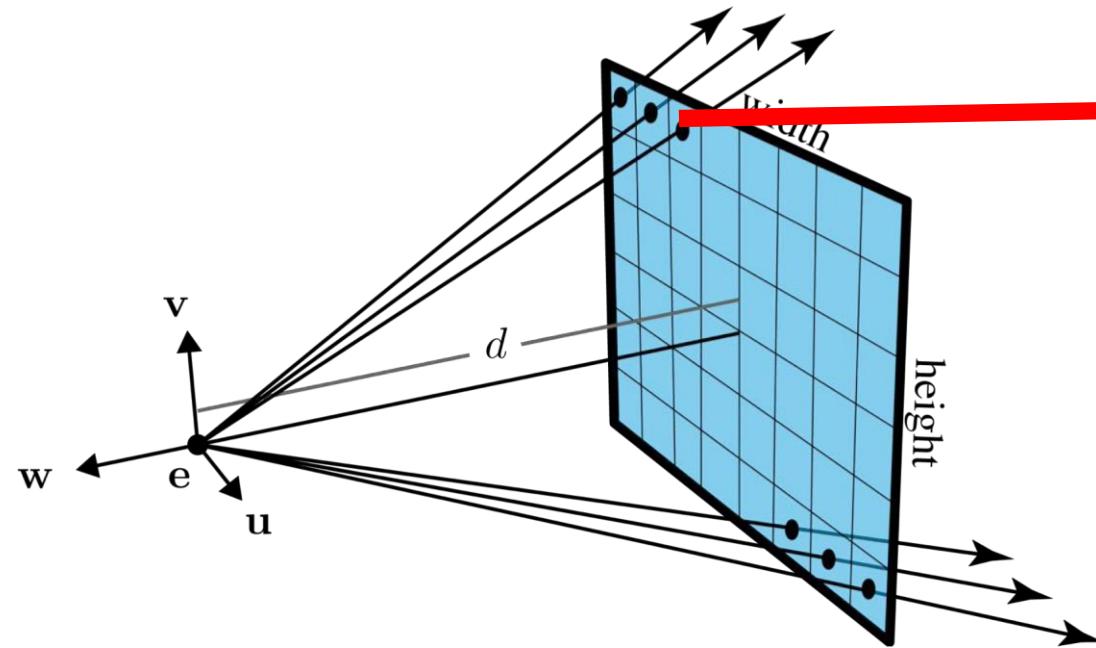
Ray Equation in Camera Space



$$p(t) = e + t(s - e)$$



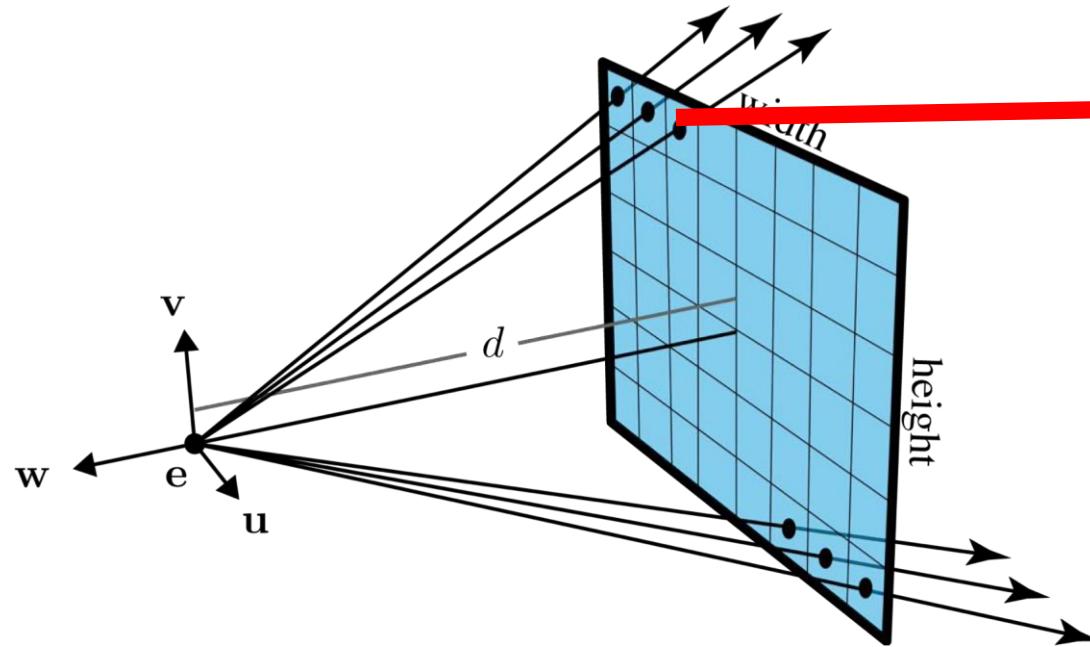
Ray Equation in Camera Space



$$p(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \left(\begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$



Ray Equation in Camera Space



$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

$$\mathbf{p}(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \left(\begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

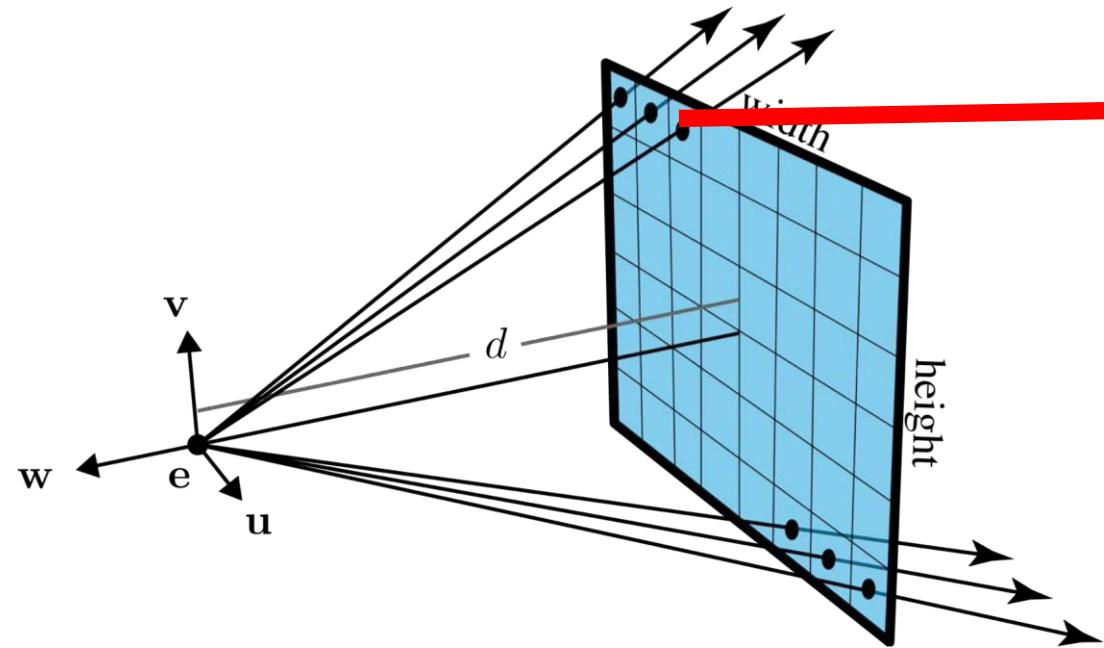
$$\mathbf{p}(t) = t \begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix}$$

$$u = \frac{\text{width}}{n_x} \cdot \left(i + \frac{1}{2} \right) - \frac{\text{width}}{2}$$

$$v = \frac{\text{height}}{n_y} \cdot \left(j + \frac{1}{2} \right) - \frac{\text{height}}{2}$$



Ray Equation in World Space

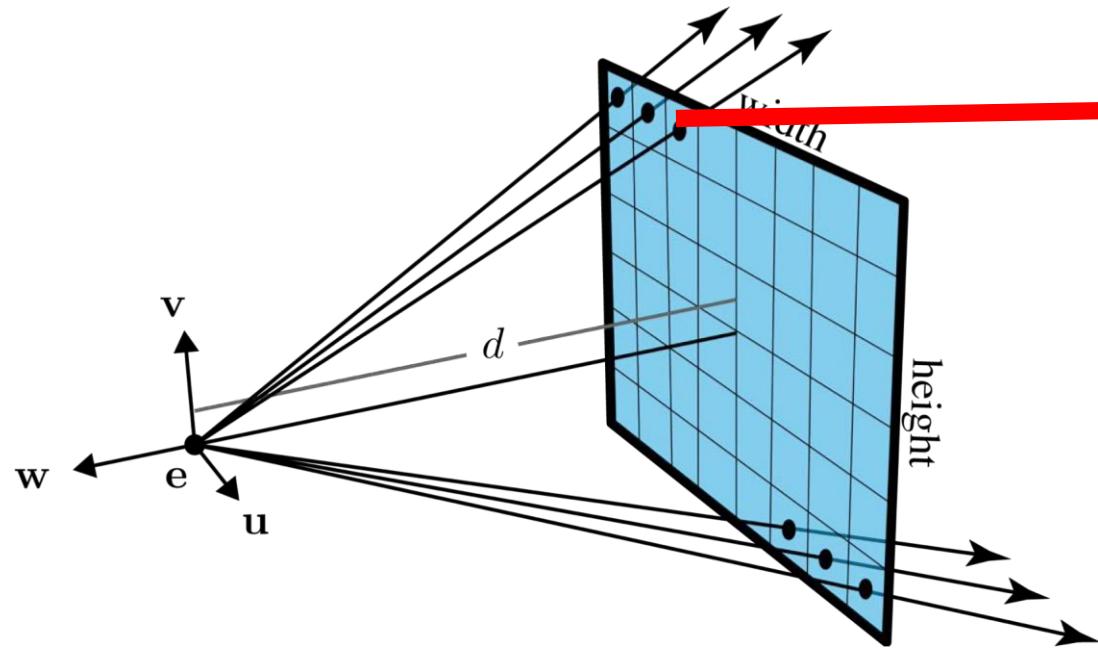


$$p(t) = e + t(s - e)$$

$$p(t) = e + t(u(i)u + v(j)v - dw)$$



Ray Equation in World Space



$$p(t) = e + t(s - e)$$

$$p(t) = e + t \begin{bmatrix} u & v & w \end{bmatrix} \begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix}$$

Camera Transformation Matrix



Ray Casting

```
for each pixel in the image {  
    Generate a ray  
    for each object in the scene {  
        if (Intersect ray with object) {  
            Set pixel colour  
        }  
    }  
}
```



Example – Pixel Space

width = 2

height = 1

$n_x = 50$

$n_y = 25$

Pixel space

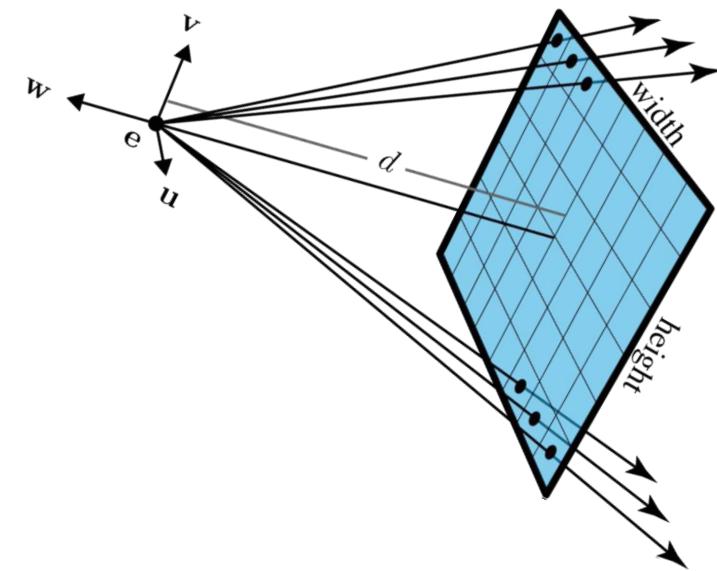
25 pixels {



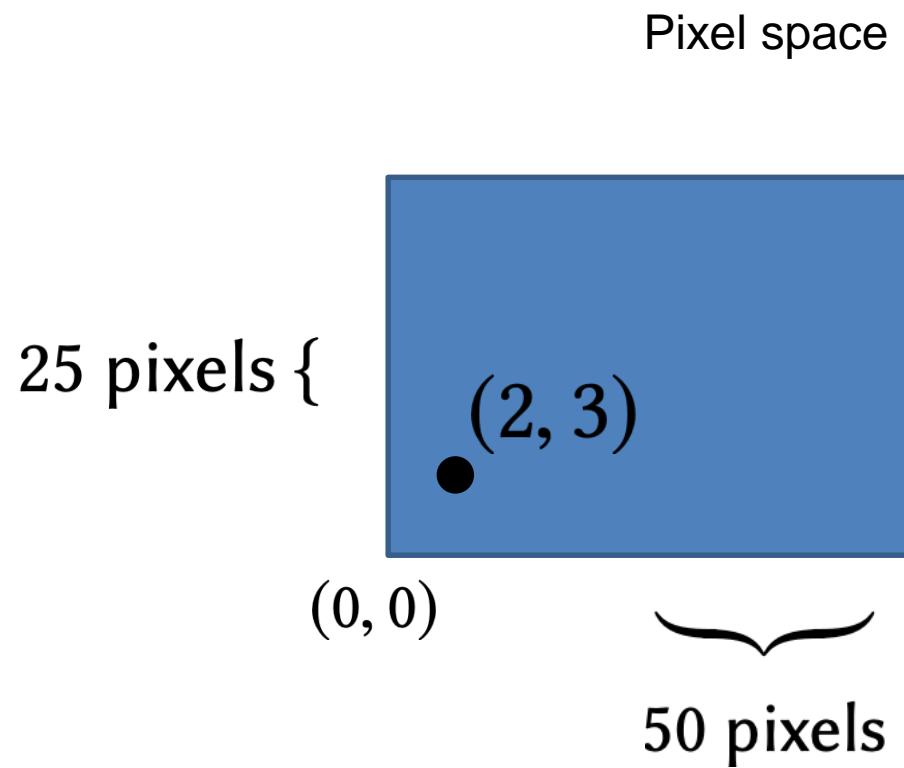
(0, 0)



50 pixels



Example – Pixel Space



width = 2

height = 1

$n_x = 50$

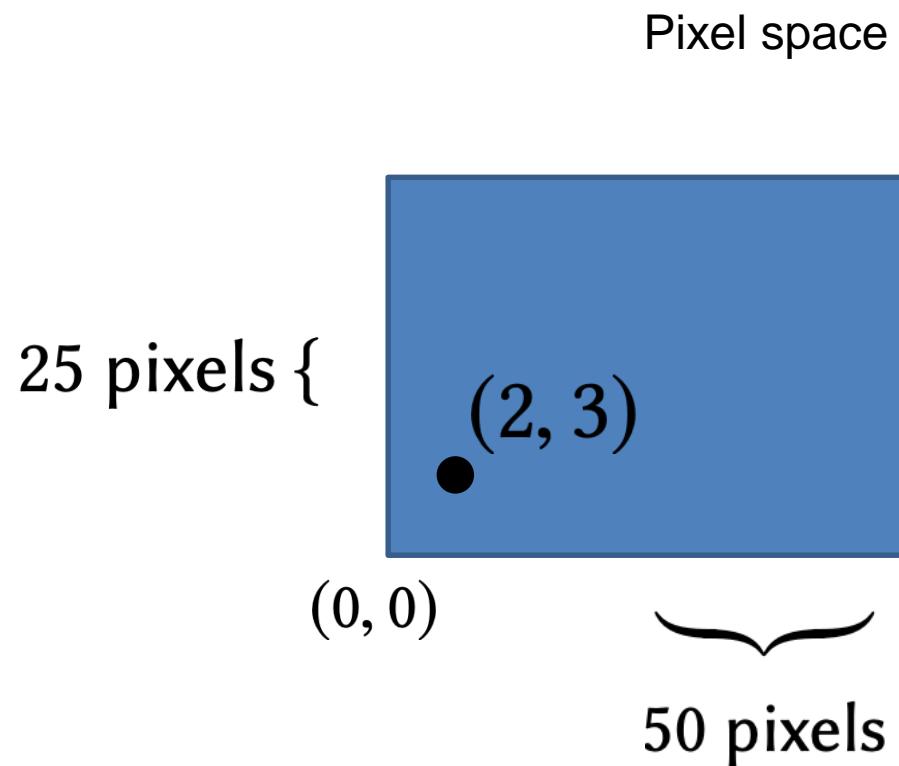
$n_y = 25$

$$u = \frac{\text{width}}{n_x} \cdot \left(i + \frac{1}{2} \right) - \frac{\text{width}}{2}$$

$$v = \frac{\text{height}}{n_y} \cdot \left(j + \frac{1}{2} \right) - \frac{\text{height}}{2}$$



Example – Pixel Space



width = 2

height = 1

$n_x = 50$

$n_y = 25$

$$u = \frac{2}{50} \cdot \left(2 + \frac{1}{2} \right) - \frac{2}{2}$$

$$v = \frac{1}{25} \cdot \left(3 + \frac{1}{2} \right) - \frac{1}{2}$$



Example – Pixel Space

$\text{width} = 2$

$\text{height} = 1$

$n_x = 50$

$n_y = 25$

25 pixels {



(0, 0)



50 pixels

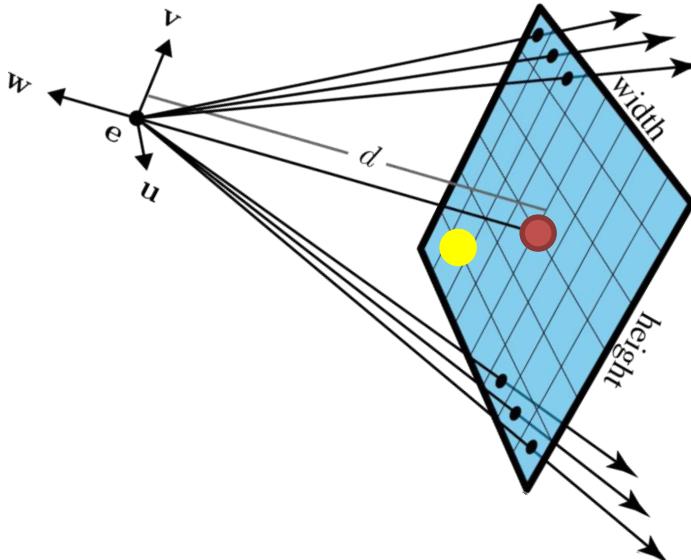
$u = -0.9$

$v = -0.36$



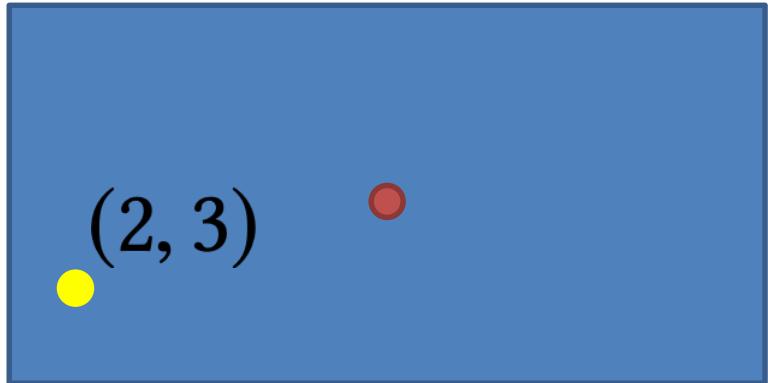
Example – Pixel Space

Camera space



Pixel space

25 pixels {



$(0, 0)$



50 pixels

$\text{width} = 2$

$\text{height} = 1$

$n_x = 50$

$n_y = 25$

$u = -0.9$

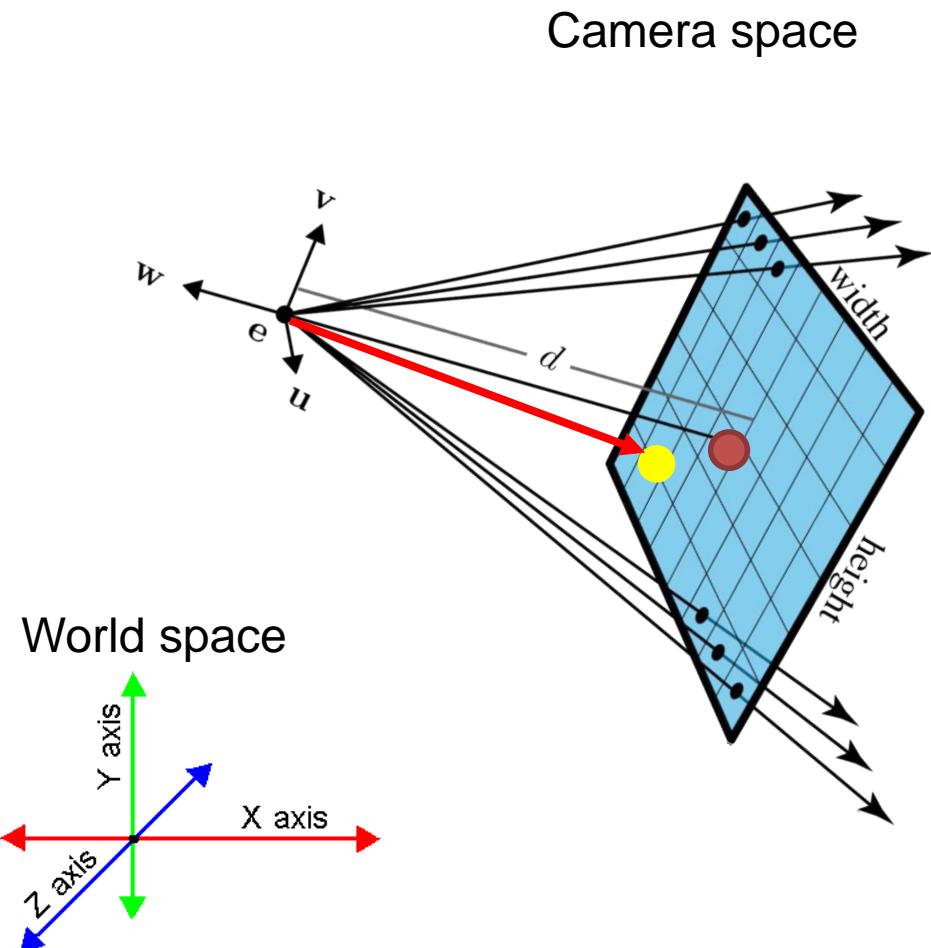
$v = -0.36$



Example – Camera Space

u, v, w

are the basis vectors for camera space



$$d = 10$$

d is the distance from the viewpoint to the image plane (focal length)

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

our ray equation



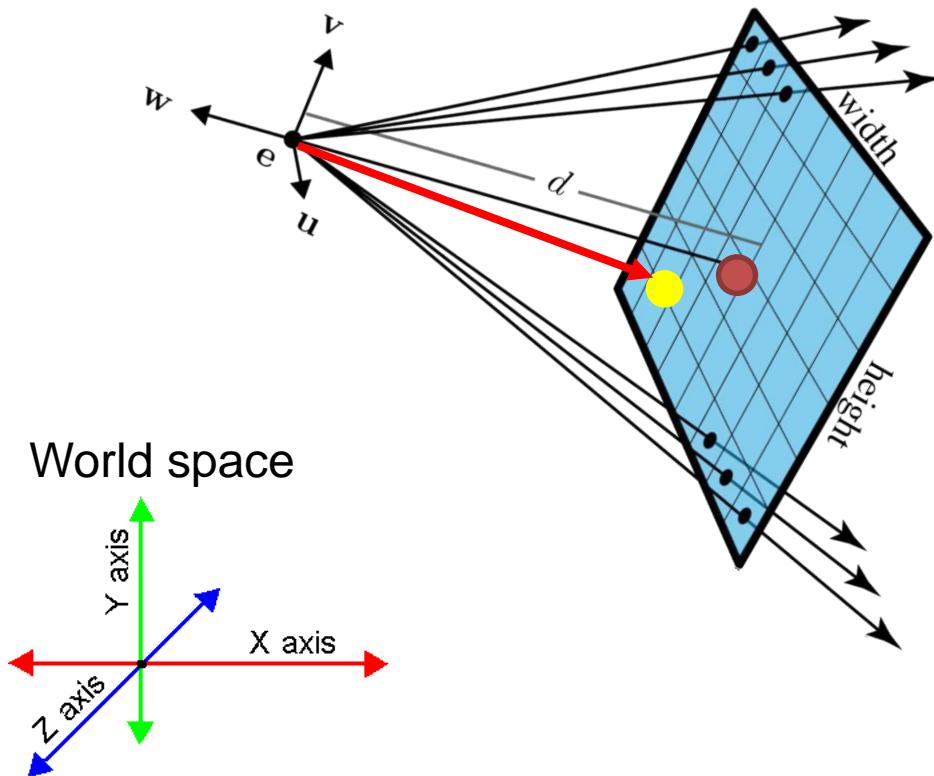
Example – Camera Space

u, v, w

are the basis vectors for camera space

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

our ray equation

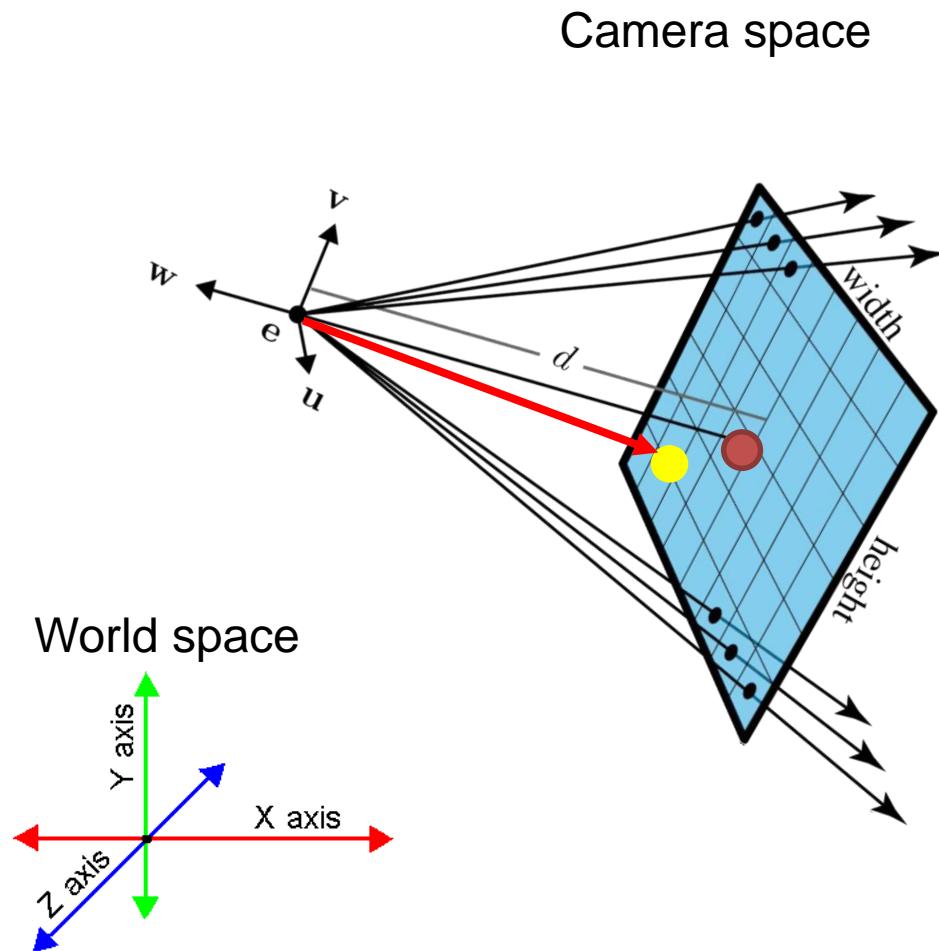


$$\mathbf{p}(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \left(\begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$



$d = 10$

Example – Camera Space



u, v, w

are the basis vectors for camera space

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

our ray equation

for $(i, j) = (2, 3)$

$$u(2) = -0.9, v(3) = -0.36$$

$$\mathbf{p}(t) = t \begin{bmatrix} -0.9 \\ -0.36 \\ -10 \end{bmatrix}$$



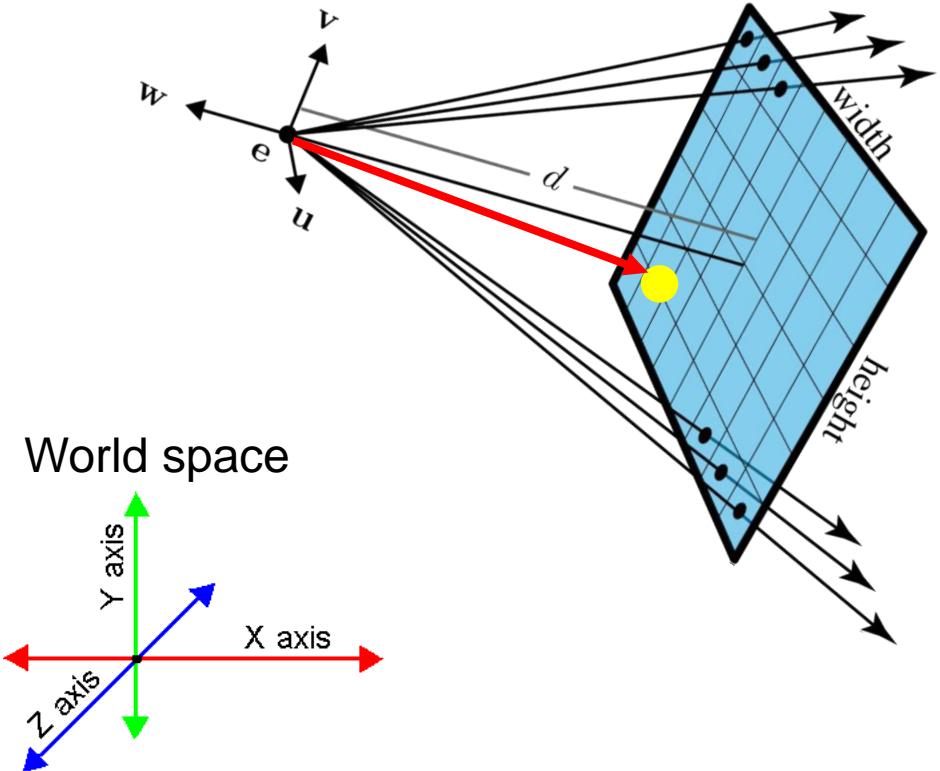
Example – World Space

u, v, w

are the basis vectors for camera space

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

our ray equation



$$\mathbf{e} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\mathbf{p}(t) = \mathbf{e} + t(u(i)\mathbf{u} + v(j)\mathbf{v} - d\mathbf{w})$$



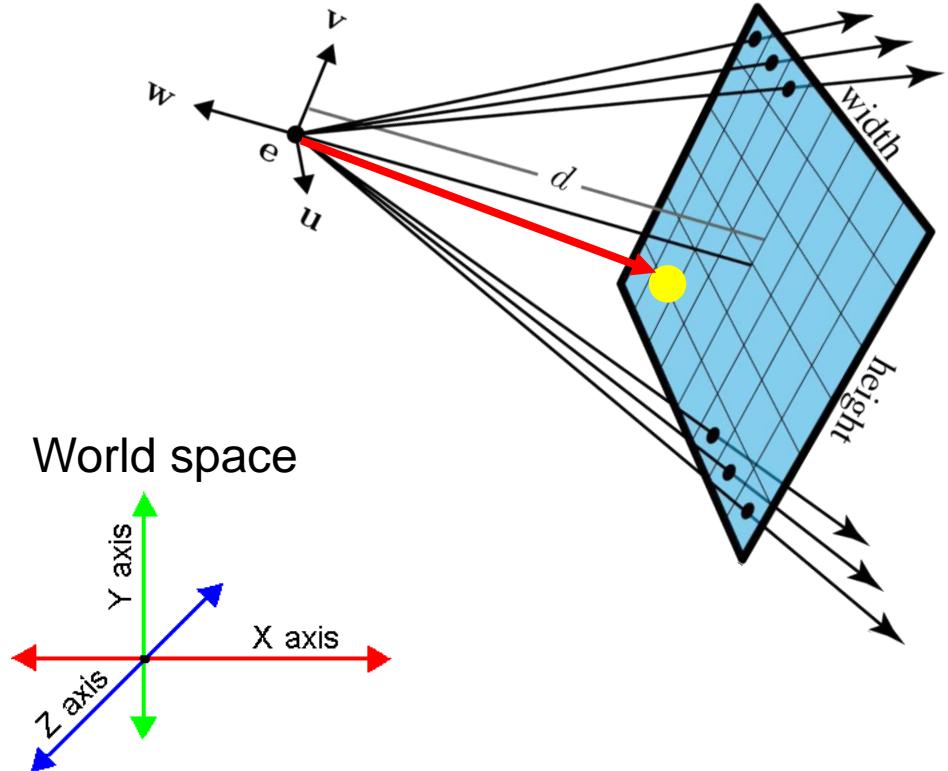
Example – World Space

u, v, w

are the basis vectors for camera space

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

our ray equation



$$\mathbf{e} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\mathbf{p}(t) = \mathbf{e} + t \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{bmatrix} \begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix}$$



Example – World Space

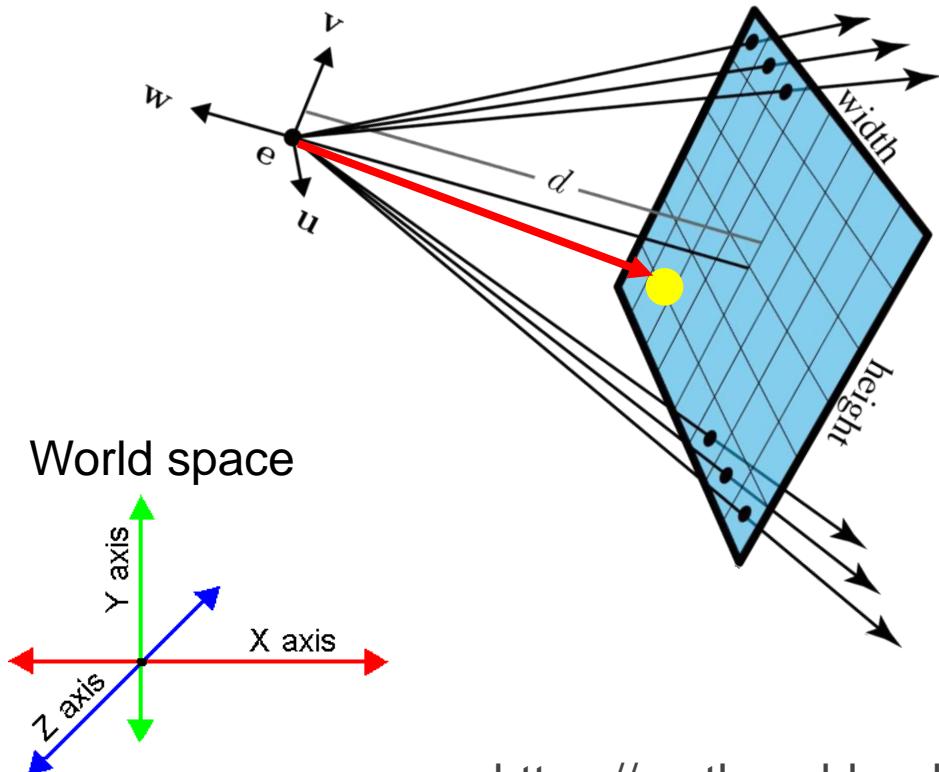
$$\mathbf{e} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

u, v, w

are the basis vectors for camera space

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

our ray equation



$$\mathbf{p}(t) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} + t \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -0.9 \\ -0.36 \\ -10 \end{bmatrix}$$



Ray Casting



Some Slides/Images adapted from Marschner and Shirley and David Levin

Basic Components of Ray Casting

Ray

Camera

Intersection Tests

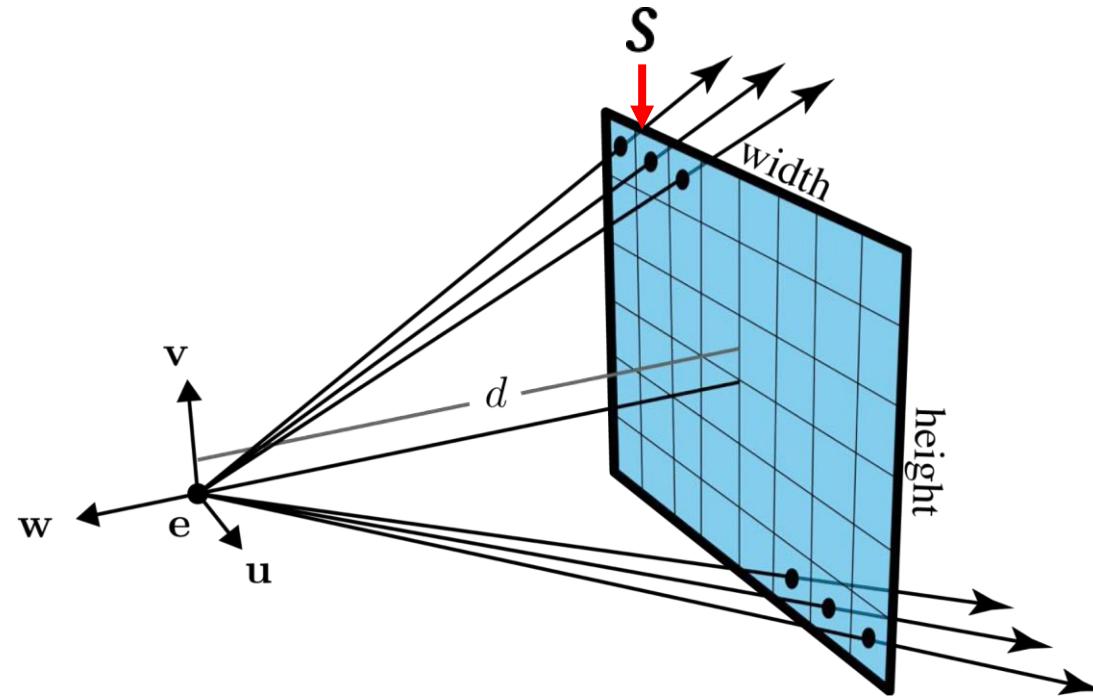


Intersection Tests

- Plane
- Sphere
- Triangle



Ray Equation

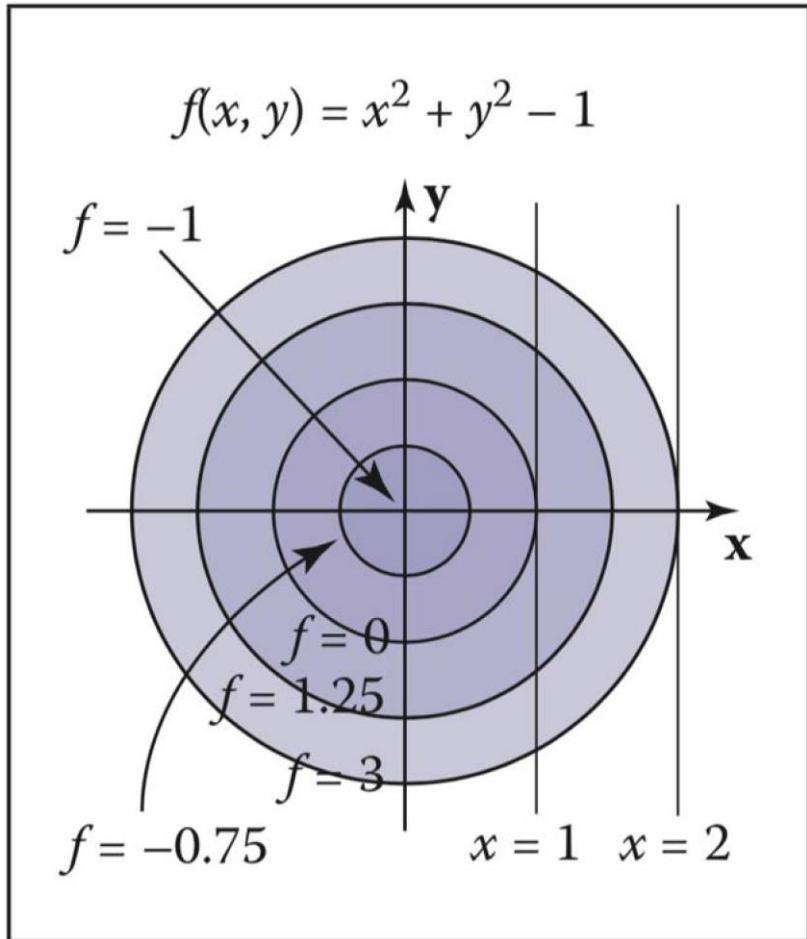


$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$



Aside: Types of Surface

Implicit Surface



Parametric Surface

$$x = r \cos \phi \sin \theta,$$

$$y = r \sin \phi \sin \theta,$$

$$z = r \cos \theta.$$

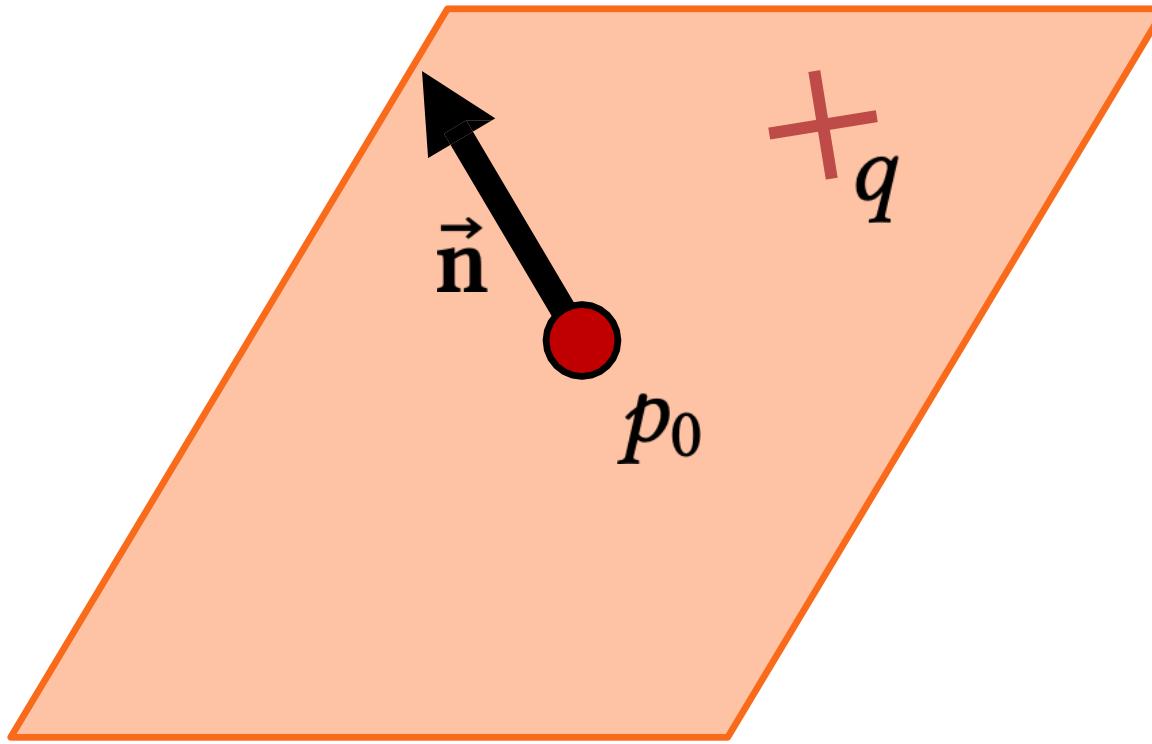


Intersection Tests

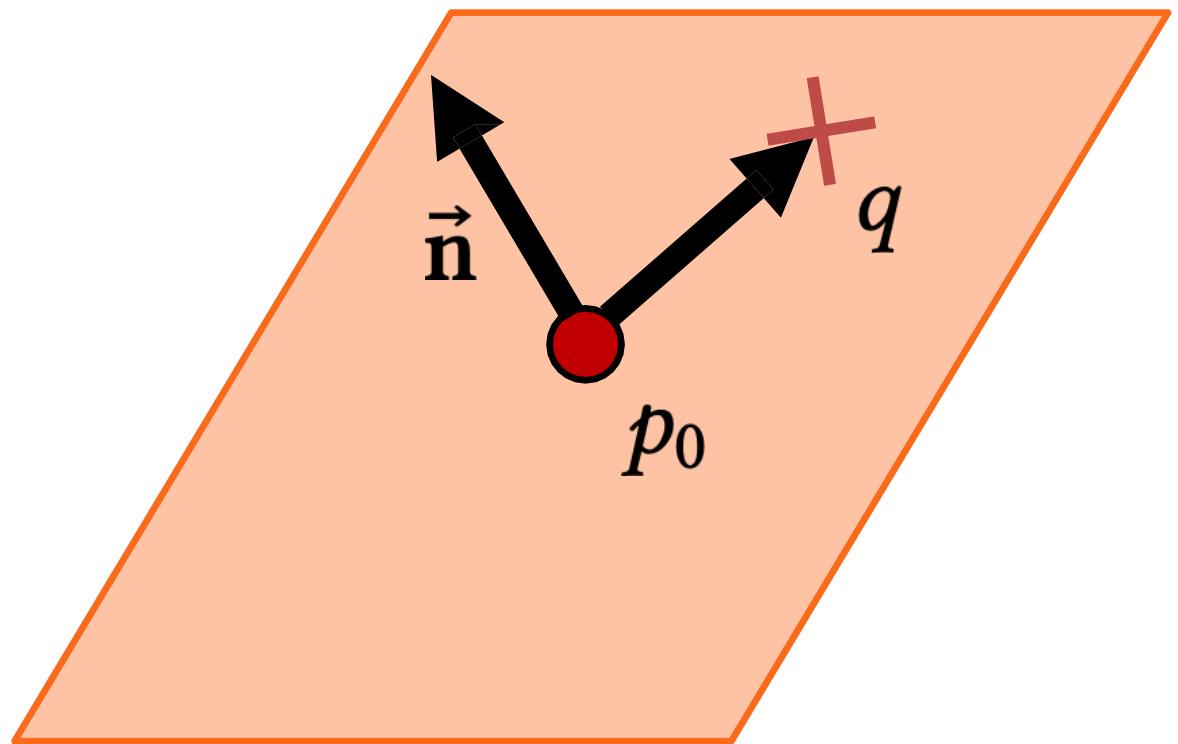
- Plane
- Sphere
- Triangle



Plane Equation



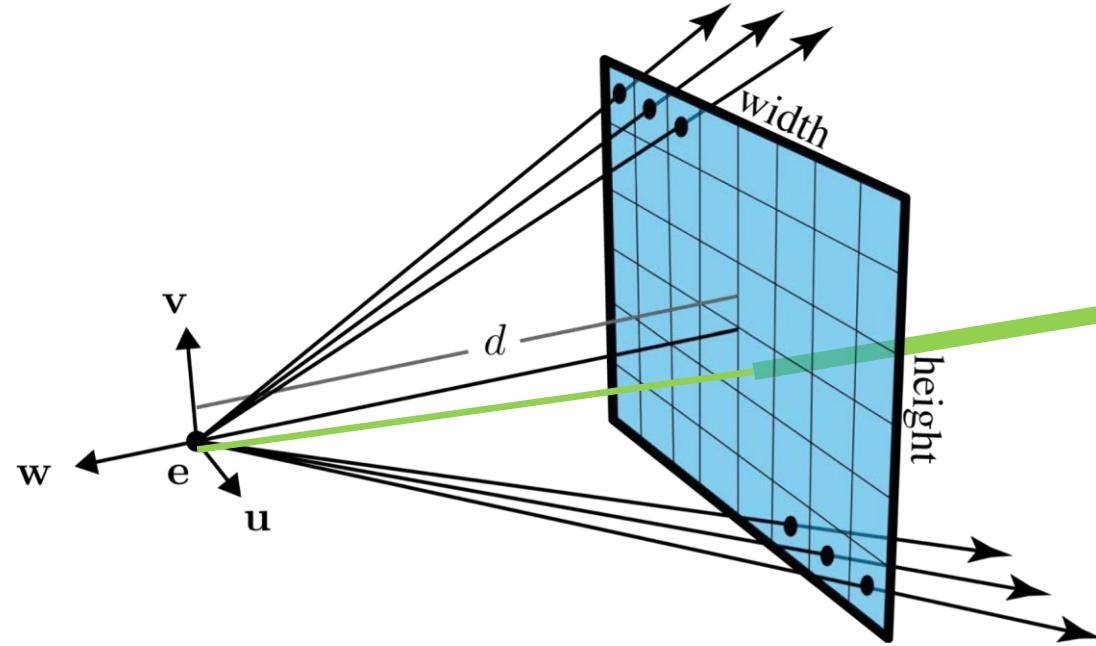
Plane Equation



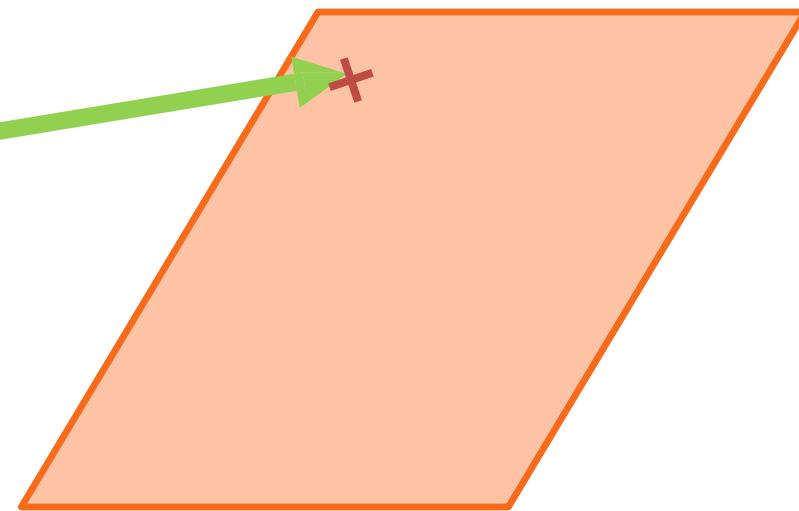
$$\vec{n} \cdot (q - p_0) = 0$$



Ray-Plane Intersection



Intersection

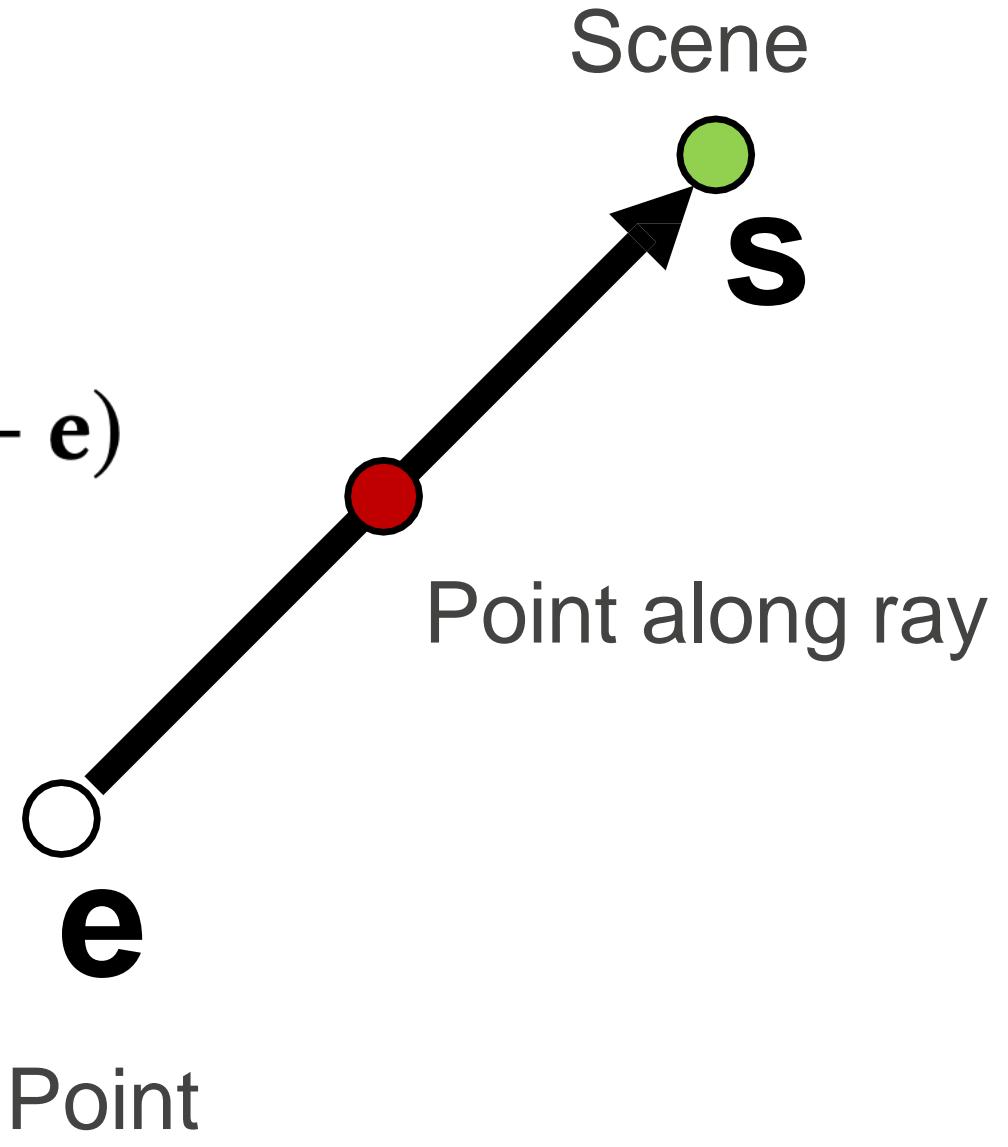


Ray-Plane Intersection

Ray equation

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

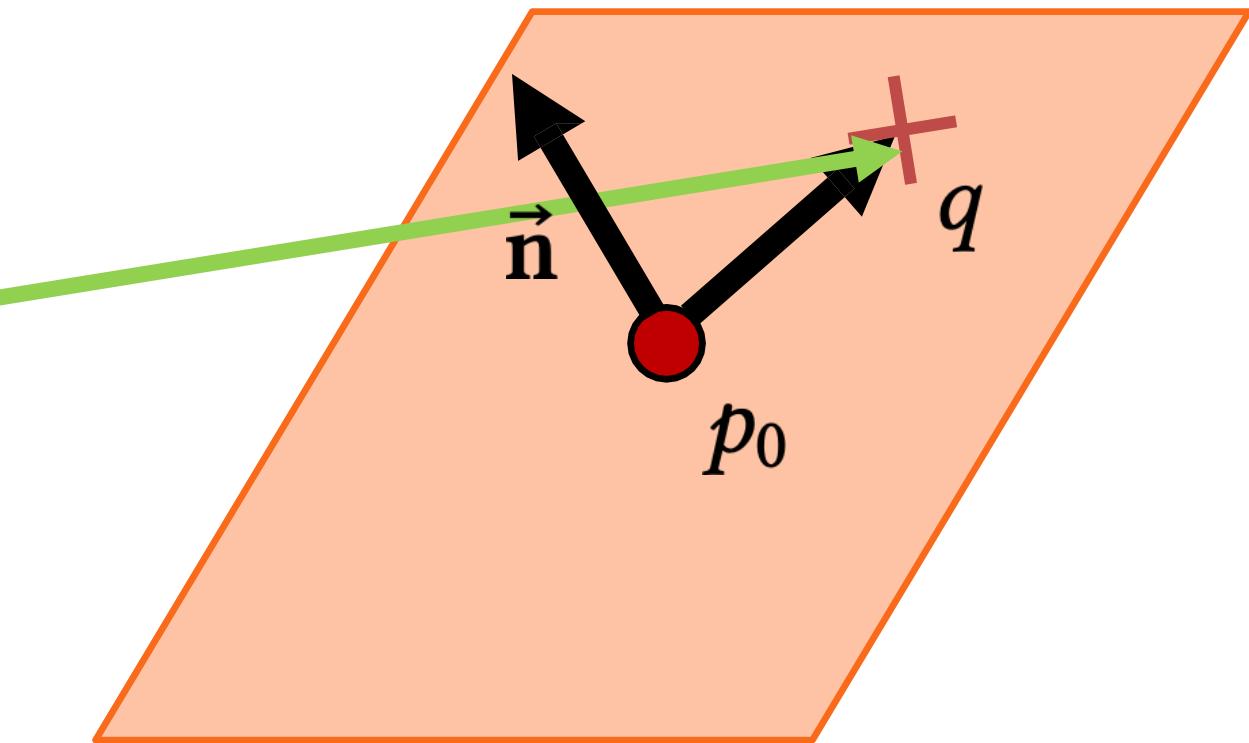
$$\mathbf{p}(t) = \mathbf{e} + t\vec{\mathbf{d}}$$



Eye Point



Plane Equation



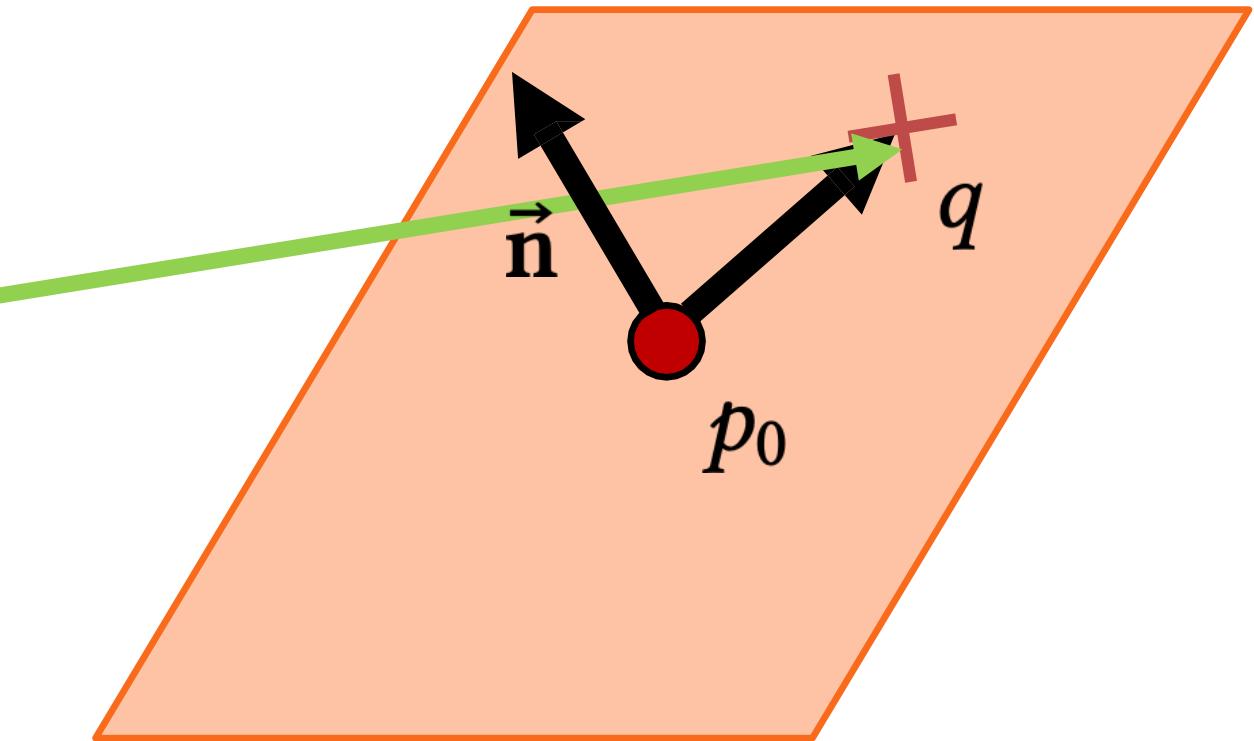
Plane equation
Substitute ray equation into it

$$\vec{n} \cdot (\mathbf{p}(t) - \mathbf{p}_0) = 0$$

$$\vec{n} \cdot ((\mathbf{e} + t\vec{\mathbf{d}}) - \mathbf{p}_0) = 0$$



Plane Equation



Plane equation
Substitute ray equation into it

$$\vec{n} \cdot (\mathbf{p}(t) - p_0) = 0$$

$$\vec{n} \cdot ((\mathbf{e} + t\vec{d}) - p_0) = 0$$

Solve for t

$$t = \frac{-\vec{n} \cdot (\mathbf{e} - p_0)}{\vec{n} \cdot \vec{d}}$$

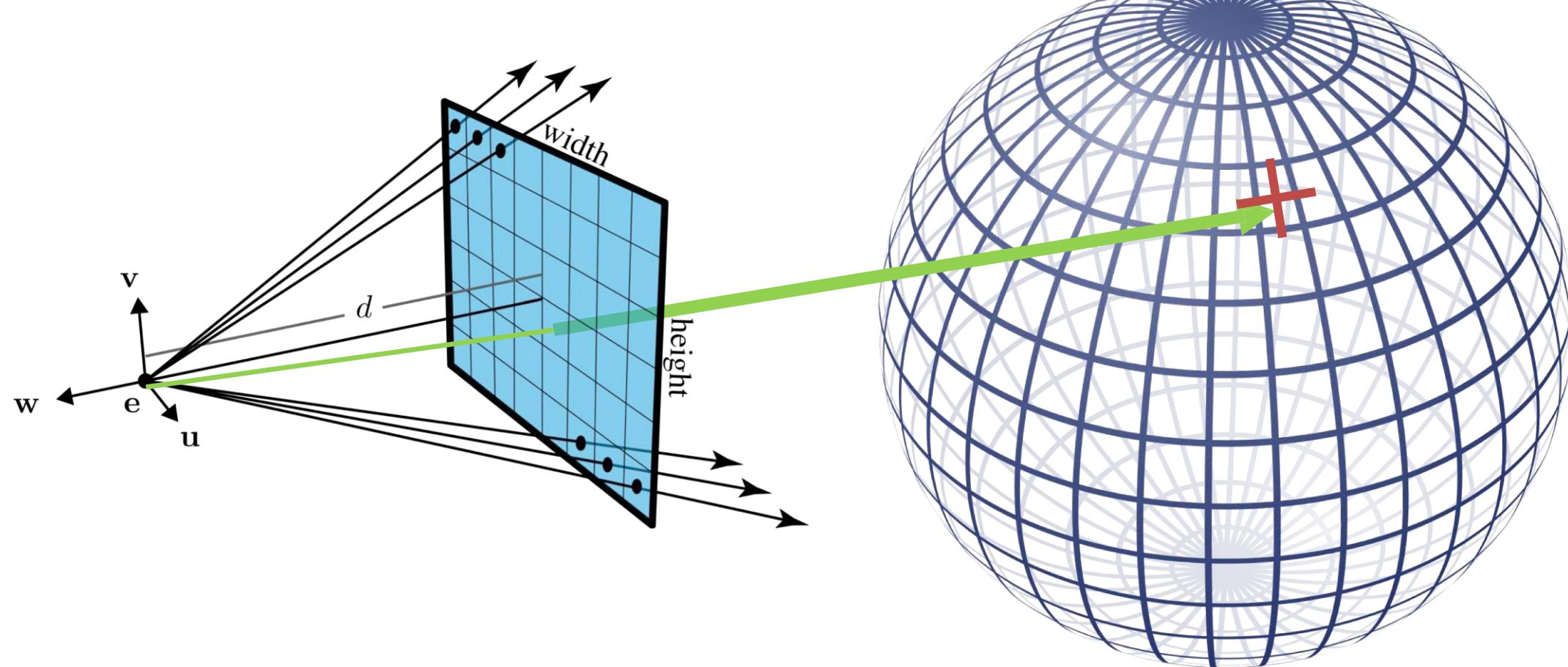


Intersection Tests

- Plane
- Sphere
- Triangle



Ray-Sphere Intersection



Implicit Equation of a Sphere

$$(q - \mathbf{c}) \cdot (q - \mathbf{c}) - r^2 = 0$$

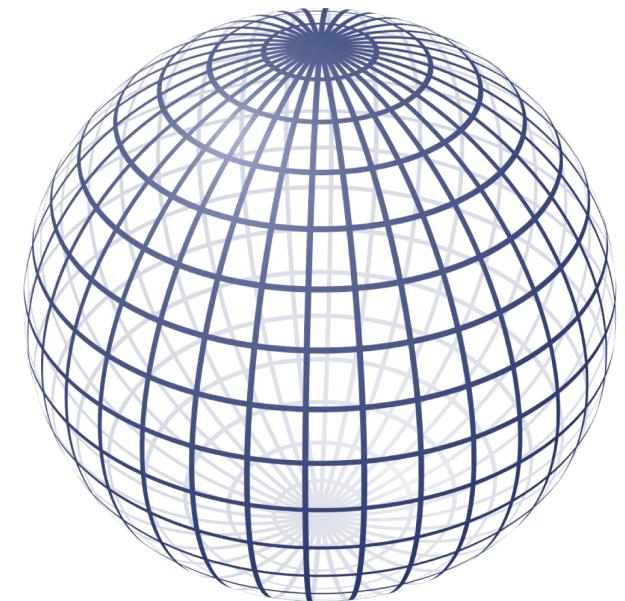
Sphere centered at \mathbf{c} with radius r



Ray-Sphere Intersection

Substitute ray equation into implicit equation for sphere

$$(\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) \cdot (\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) - r^2 = 0$$



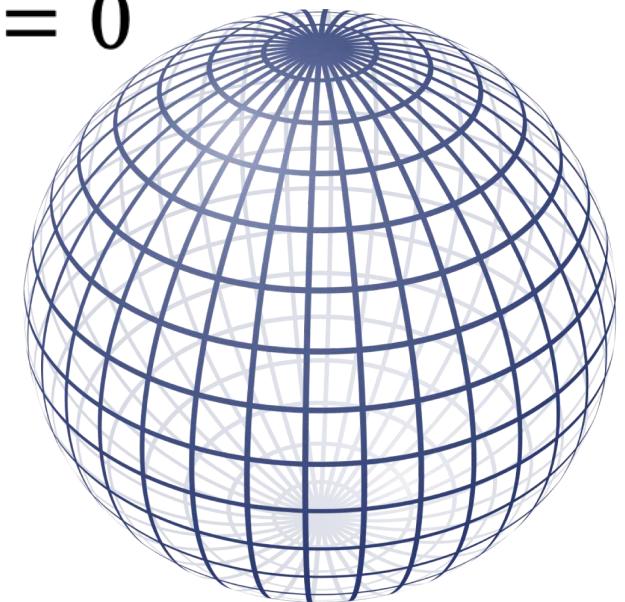
Ray-Sphere Intersection

Substitute ray equation into implicit equation for sphere

$$(\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) \cdot (\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) - r^2 = 0$$

Rearrange

$$(\vec{\mathbf{d}} \cdot \vec{\mathbf{d}})t^2 + 2\vec{\mathbf{d}} \cdot (\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2 = 0$$



Ray-Sphere Intersection

Substitute ray equation into implicit equation for sphere

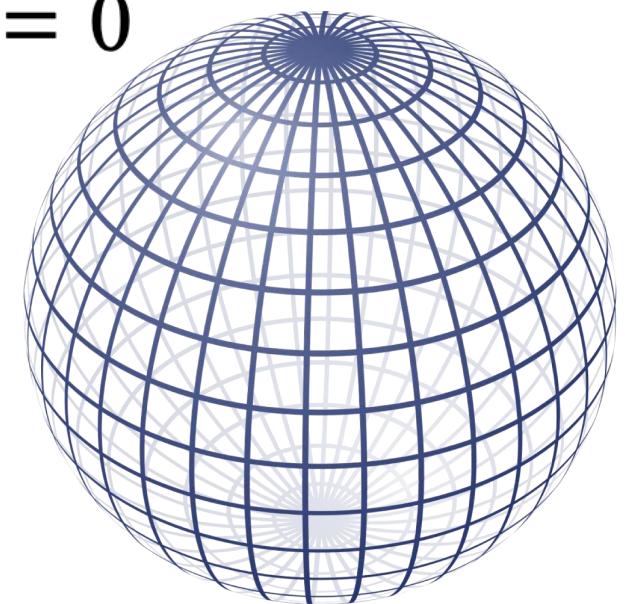
$$(\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) \cdot (\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) - r^2 = 0$$

Rearrange

$$(\vec{\mathbf{d}} \cdot \vec{\mathbf{d}})t^2 + 2\vec{\mathbf{d}} \cdot (\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2 = 0$$

Looks familiar...

$$At^2 + Bt + C = 0$$



Ray-Sphere Intersection

Substitute ray equation into implicit equation for sphere

$$(\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) \cdot (\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) - r^2 = 0$$

Rearrange

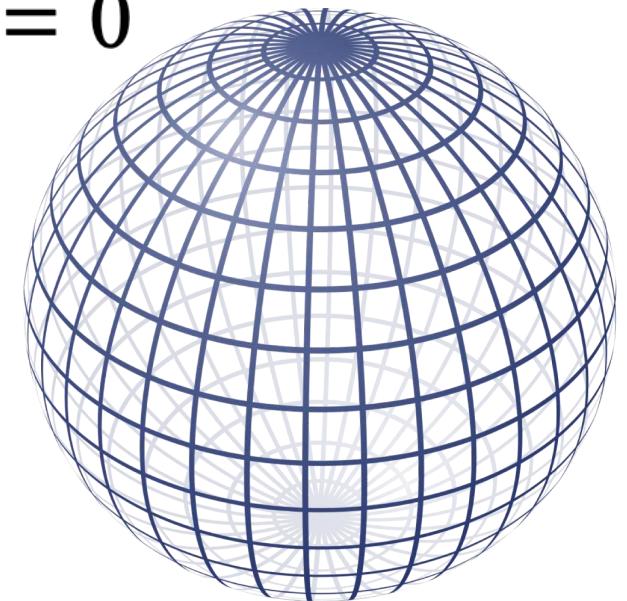
$$(\vec{\mathbf{d}} \cdot \vec{\mathbf{d}})t^2 + 2\vec{\mathbf{d}} \cdot (\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2 = 0$$

Looks familiar...

$$At^2 + Bt + C = 0$$

It's a quadratic! (can use the quadratic equation)

Hint: the discriminant tells us what kinds of roots the equation has.

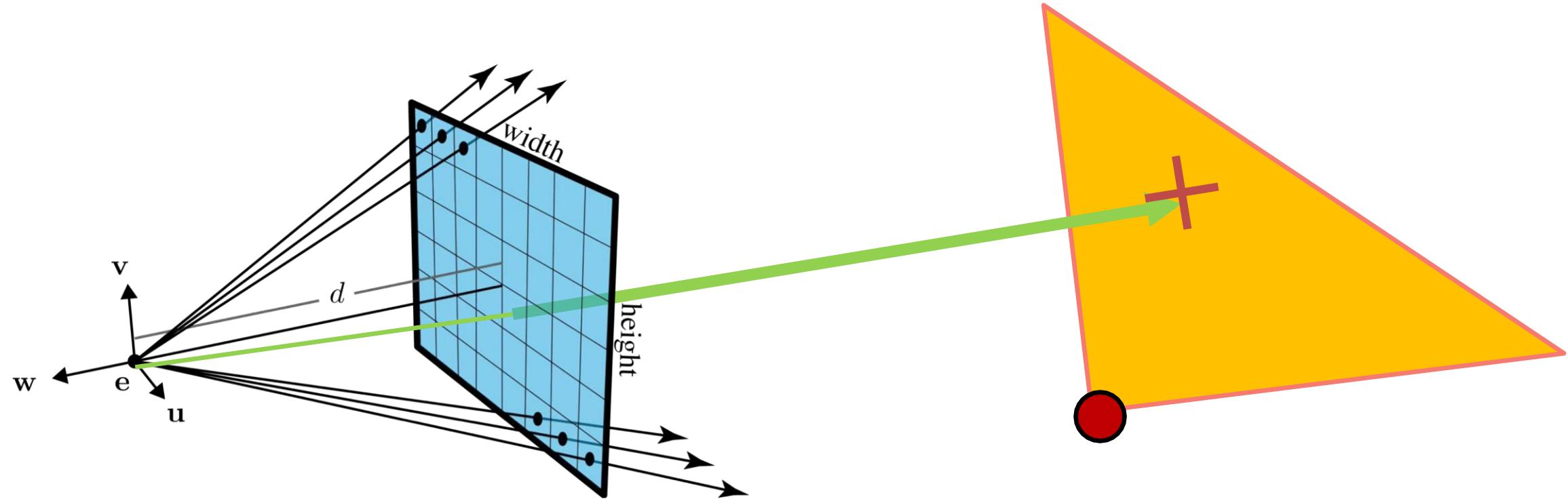


Intersection Tests

- Plane
- Sphere
- Triangle

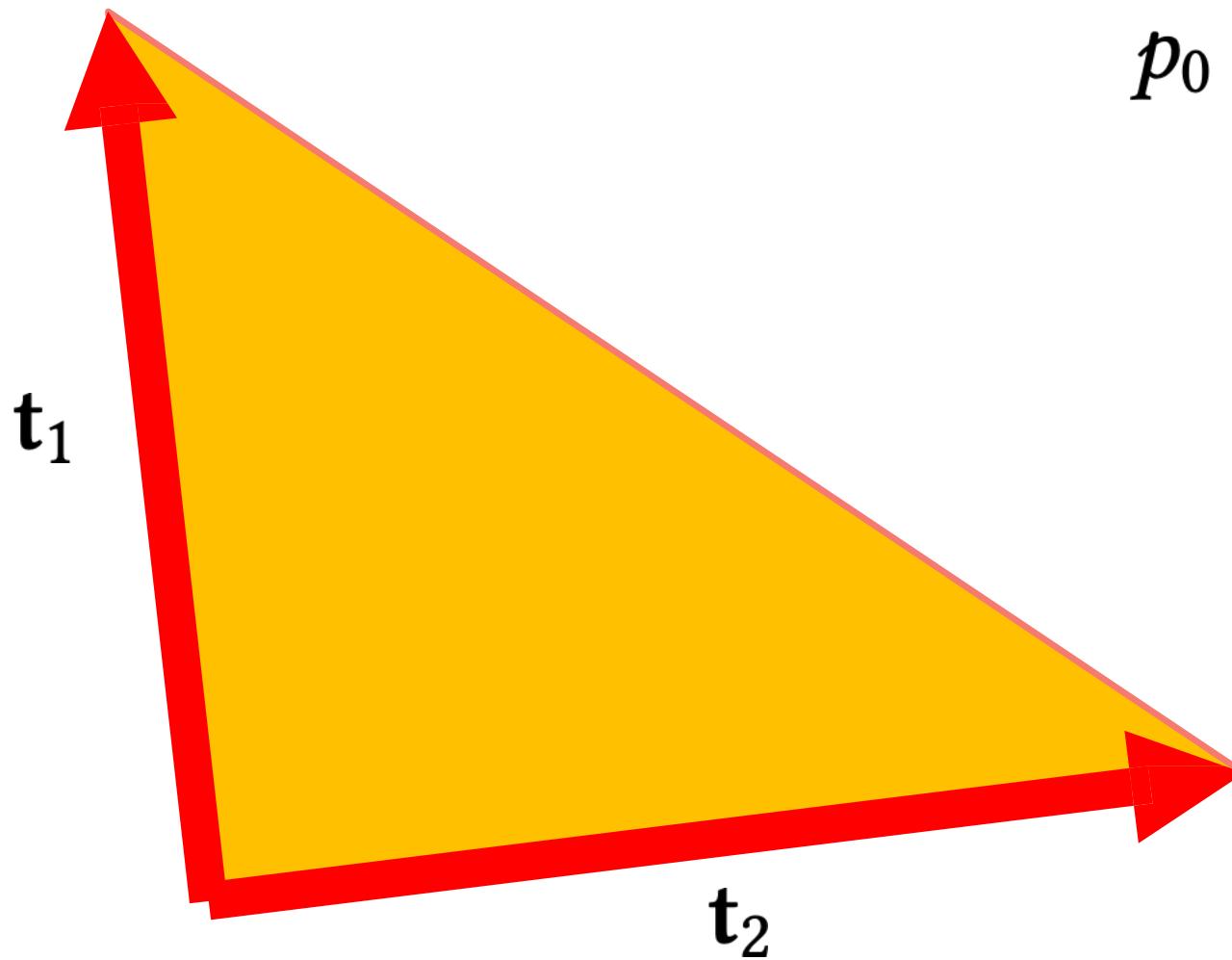


Ray-Triangle Intersection

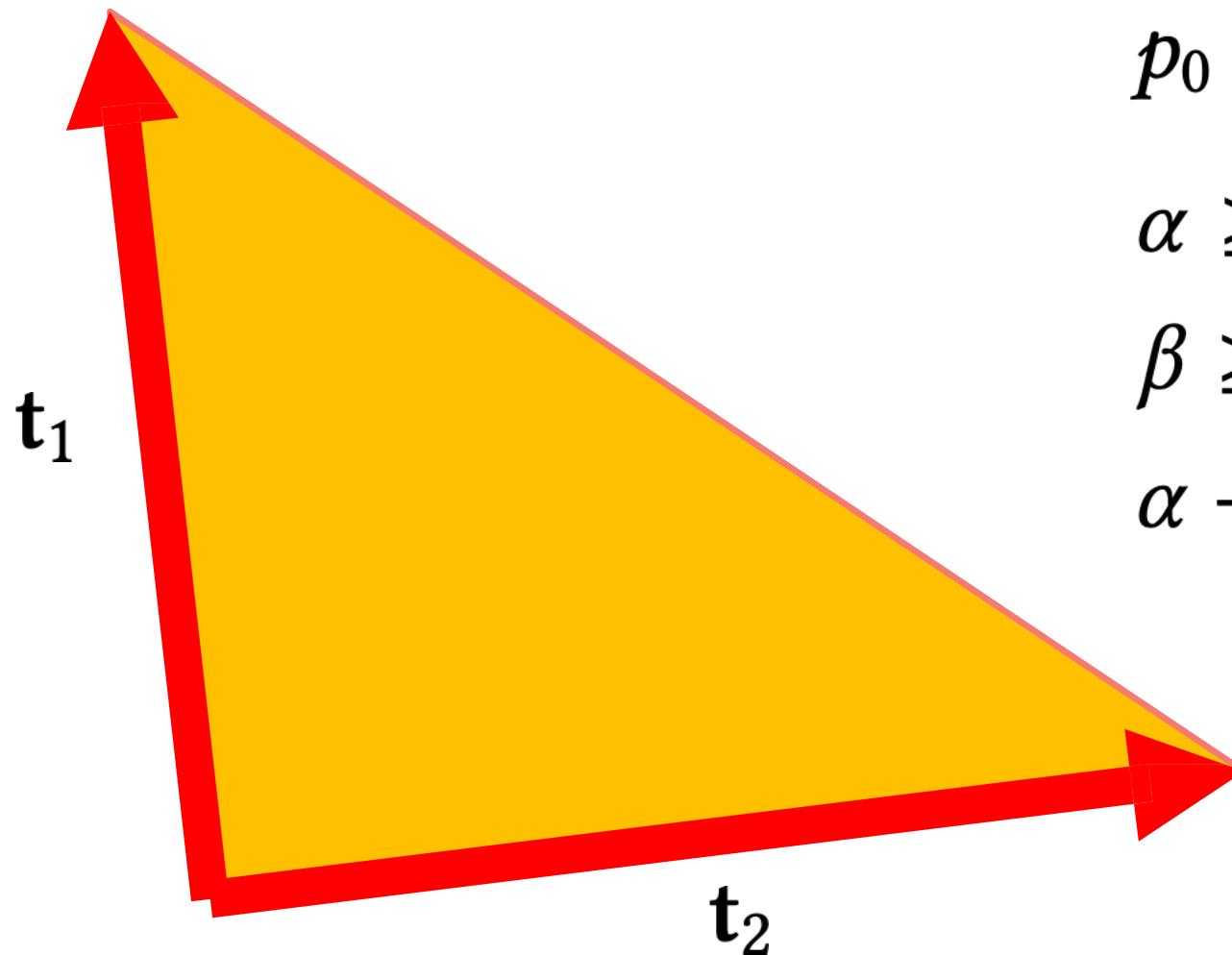


Equations for a Triangle

$$p_0 = \alpha t_1 + \beta t_2$$



Equations for a Triangle



$$p_0 = \alpha \mathbf{t}_1 + \beta \mathbf{t}_2$$

$$\alpha \geq 0$$

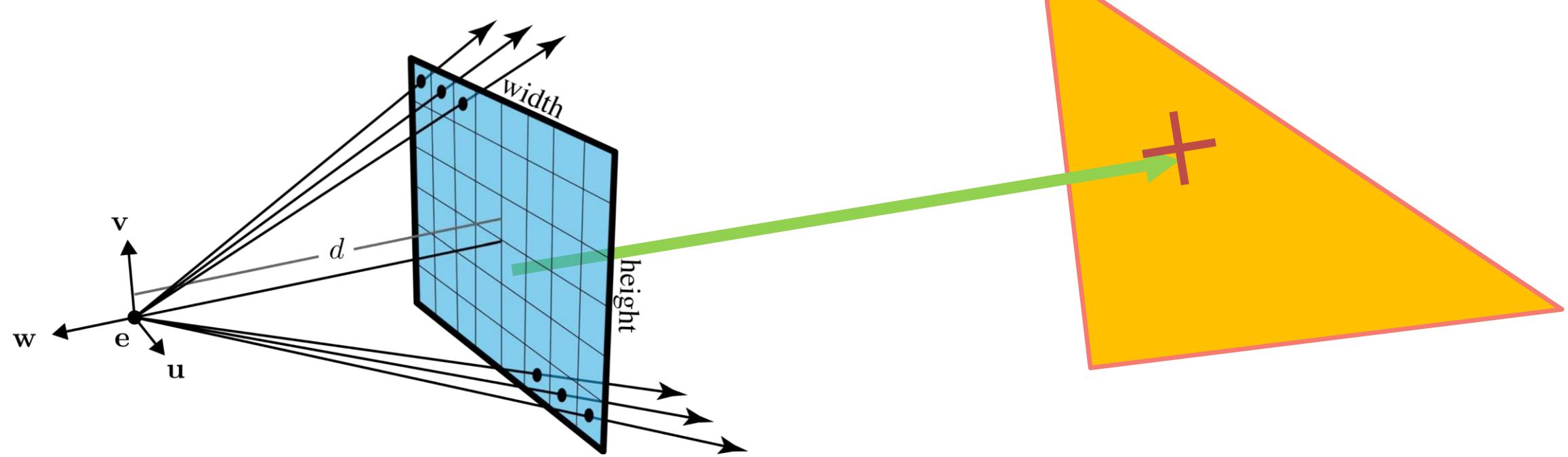
$$\beta \geq 0$$

$$\alpha + \beta \leq 1$$



Intersection with a Triangle (Parametric Surface)

Check via equating point on surface with point on ray



Intersection with a Triangle (Parametric Surface)

Check via equating point on surface with point on ray

$$\mathbf{p}(t) = \alpha \mathbf{t}_1 + \beta \mathbf{t}_2$$

$$\mathbf{e} + t\vec{\mathbf{d}} = \alpha \mathbf{t}_1 + \beta \mathbf{t}_2$$

$$\mathbf{e} = \alpha \mathbf{t}_1 + \beta \mathbf{t}_2 - t\vec{\mathbf{d}}$$



Intersection with a Triangle (Parametric Surface)

Check via equating point on surface with point on ray

$$\mathbf{e} = \alpha \mathbf{t}_1 + \beta \mathbf{t}_2 - t \vec{\mathbf{d}}$$

$$\mathbf{e} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad -\vec{\mathbf{d}}] \begin{bmatrix} \alpha \\ \beta \\ t \end{bmatrix}$$

Check values of α, β, t



Ray Casting

```
for each pixel in the image {  
    Generate a ray  
  
    for each object in the scene {  
  
        if (Intersect ray with object)  
            { Set pixel colour  
  
        }  
    }  
}
```

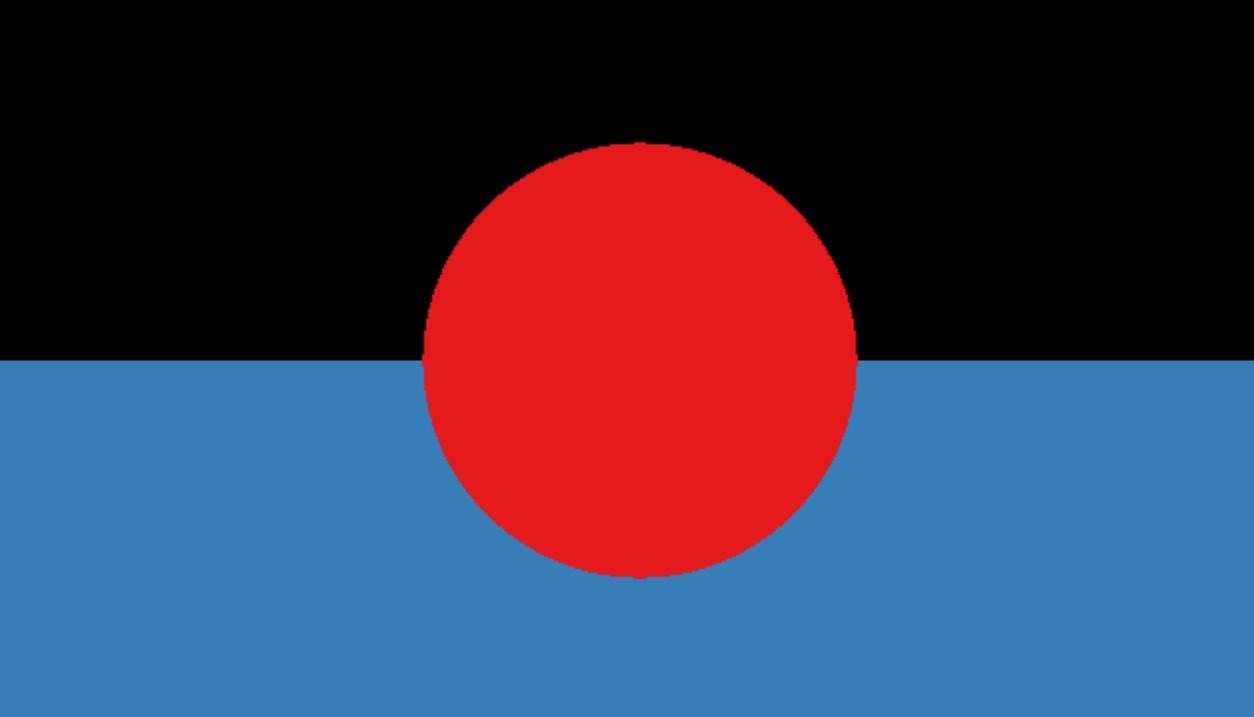


Output Type

- Object ID
- Surface Normal
- Depth



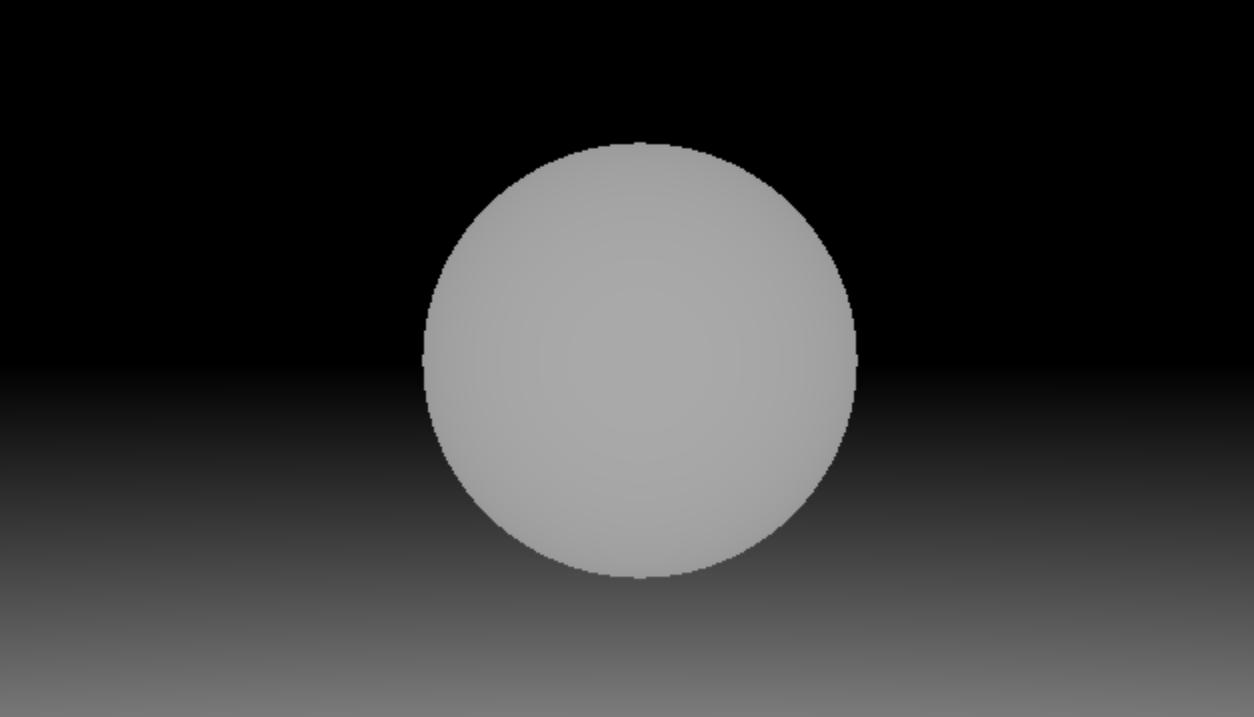
Output Type: Object ID



Output Type: Surface Normal



Output Type: Depth



Assignment #2 tasks

`src/viewing_ray.cpp`

Construct a viewing ray given a camera and subscripts to a pixel.

`src/first_hit.cpp`

Find the first (visible) hit given a ray and a collection of scene objects

`Sphere::intersect_ray` in `src/Sphere.cpp`

Intersect a sphere with a ray.

`Plane::intersect_ray` in `src/Plane.cpp`

Intersect a plane with a ray.

`Triangle::intersect_ray` in `src/Triangle.cpp`

Intersect a triangle with a ray.

`TriangleSoup::intersect_ray` in `src/TriangleSoup.cpp`

Intersect a triangle soup with a ray.



Next Week

Ray Tracing

