# Ray Casting

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

# Announcements

- Assignment 1 is due tomorrow at midnight (12 May)
- First tutorial zoom recording is uploaded (sorry it’s not all of it, I started recording halfway through)

<u>Reminders:</u>

- Tutorials are on Fridays and are used to ask questions about the concepts, assignments, and course in general
- Use the github issues page to ask questions about the assignments.
    - There will be a TA monitoring the page

# Clarifications

## Over operator from wikipedia

(https://en.wikipedia.org/wiki/Alpha_compositing):

$$C_o = \frac{C_a \alpha_a + C_b \alpha_b (1 - \alpha_a)}{\alpha_a + \alpha_b (1 - \alpha_a)}$$

or

$$c_o = c_a + c_b (1 - \alpha_a),$$

the RGB components represent the emission of the object or pixel

$$\alpha_o = \frac{c_o}{C_o} = \alpha_a + \alpha_b (1 - \alpha_a)$$

the alpha represents the occlusion

## For further reading:

https://keithp.com/~keithp/porterduff/p253-porter.pdf

https://tomforsyth1000.github.io/blog.wiki.html#%5B%5BPremultiplied+alpha%5D%5D

# Clarifications

"Normal" alpha-blending munges together two physically separate effects - the amount of light this layer of rendering lets through from behind, and the amount of light this layer adds to the image. Instead, it keeps the two related - you can't have a surface that adds a bunch of light to a scene without it also masking off what is behind it. Physically, this makes no sense - just because you add a bunch of photons into a scene, doesn't mean all the other photons are blocked. Premultiplied alpha fixes this by keeping the two concepts separate - the blocking is done by the alpha channel, the addition of light is done by the colour channel. This is not just a neat concept, it's really useful in practice.

$$C_o = \frac{C_a \alpha_a + C_b \alpha_b (1 - \alpha_a)}{\alpha_a + \alpha_b (1 - \alpha_a)}$$

or

$$c_o = c_a + c_b (1 - \alpha_a),$$

$$\alpha_o = \frac{c_o}{C_o} = \alpha_a + \alpha_b (1 - \alpha_a)$$

https://tomforsyth1000.github.io/blog.wiki.html#%5B%5BPremultiplied+alpha%5D%5D

# Any Questions?

# Today: Ray Casting

The Ray Casting Algorithm

Introduction to Rays

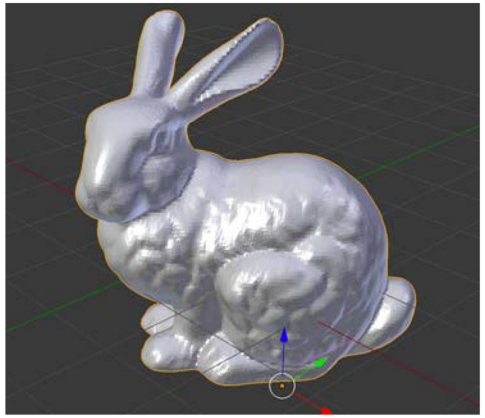
The Camera

Ray-Object Intersection

- Ray-Plane Intersection
- Ray-Sphere Intersection
- Ray-Triangle Intersection

# Photography

Input:



Objects

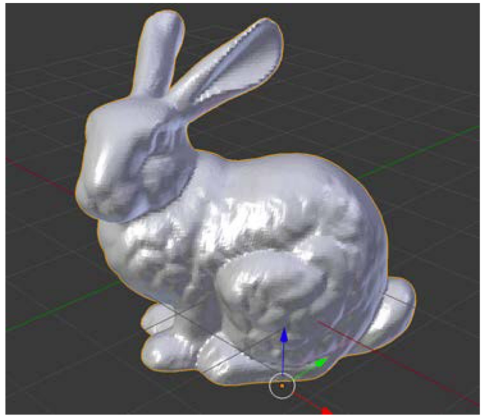
Lights

Output:

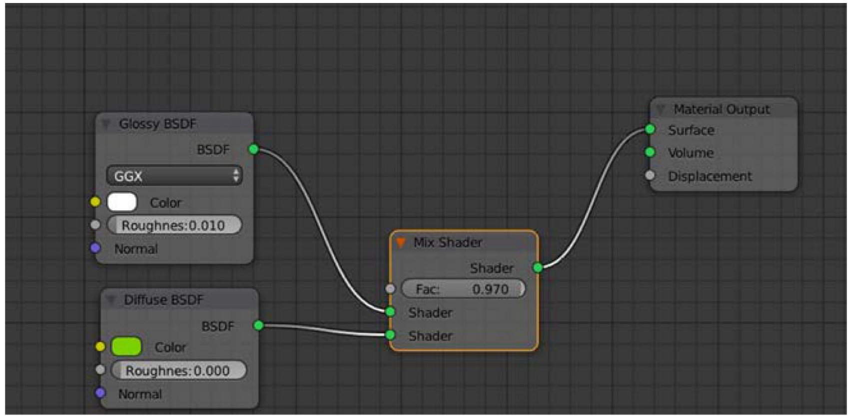
$$I\left(x,y\right)$$

# Rendering

Input:



Objects

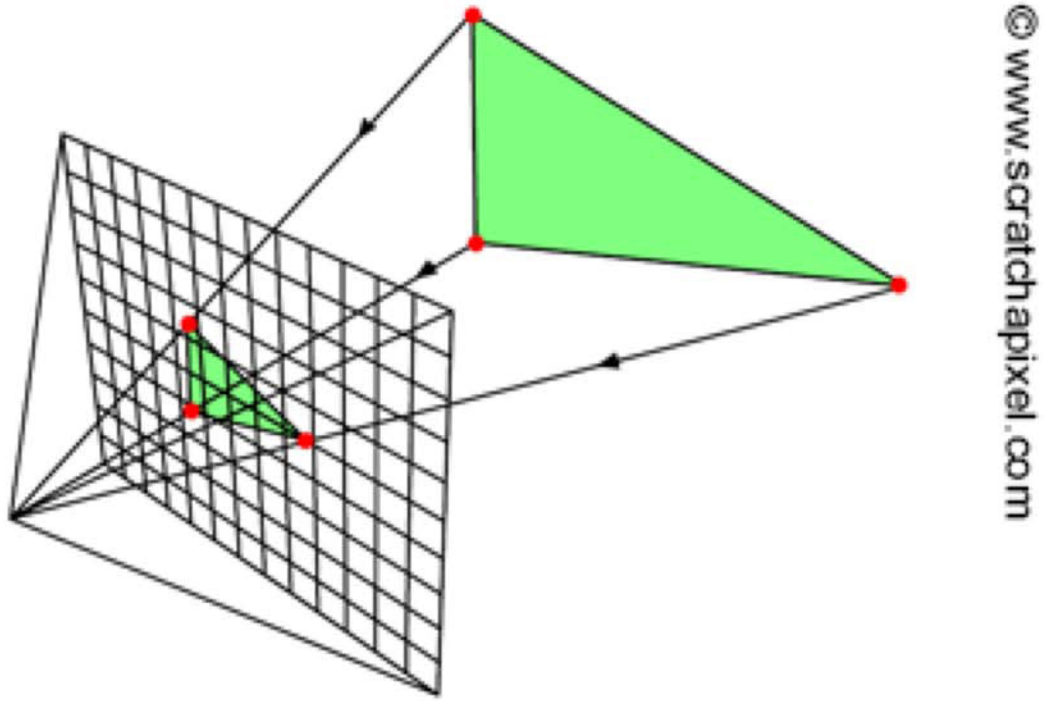



Materials

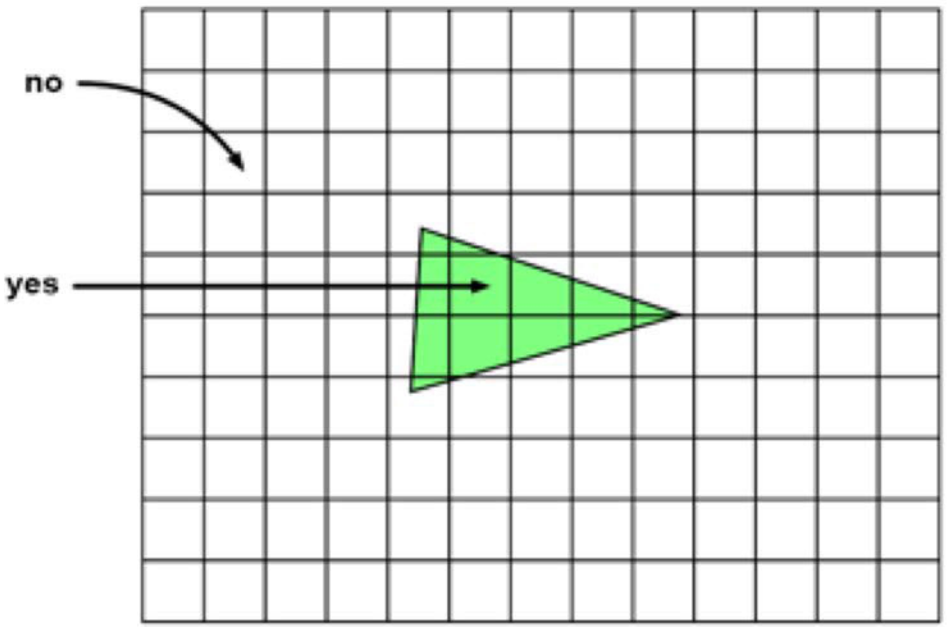
Output:

$I\left(x, y\right)$

# Rasterization




1. Project Vertices to Image Plane




2. Turn on pixels inside triangle

https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation

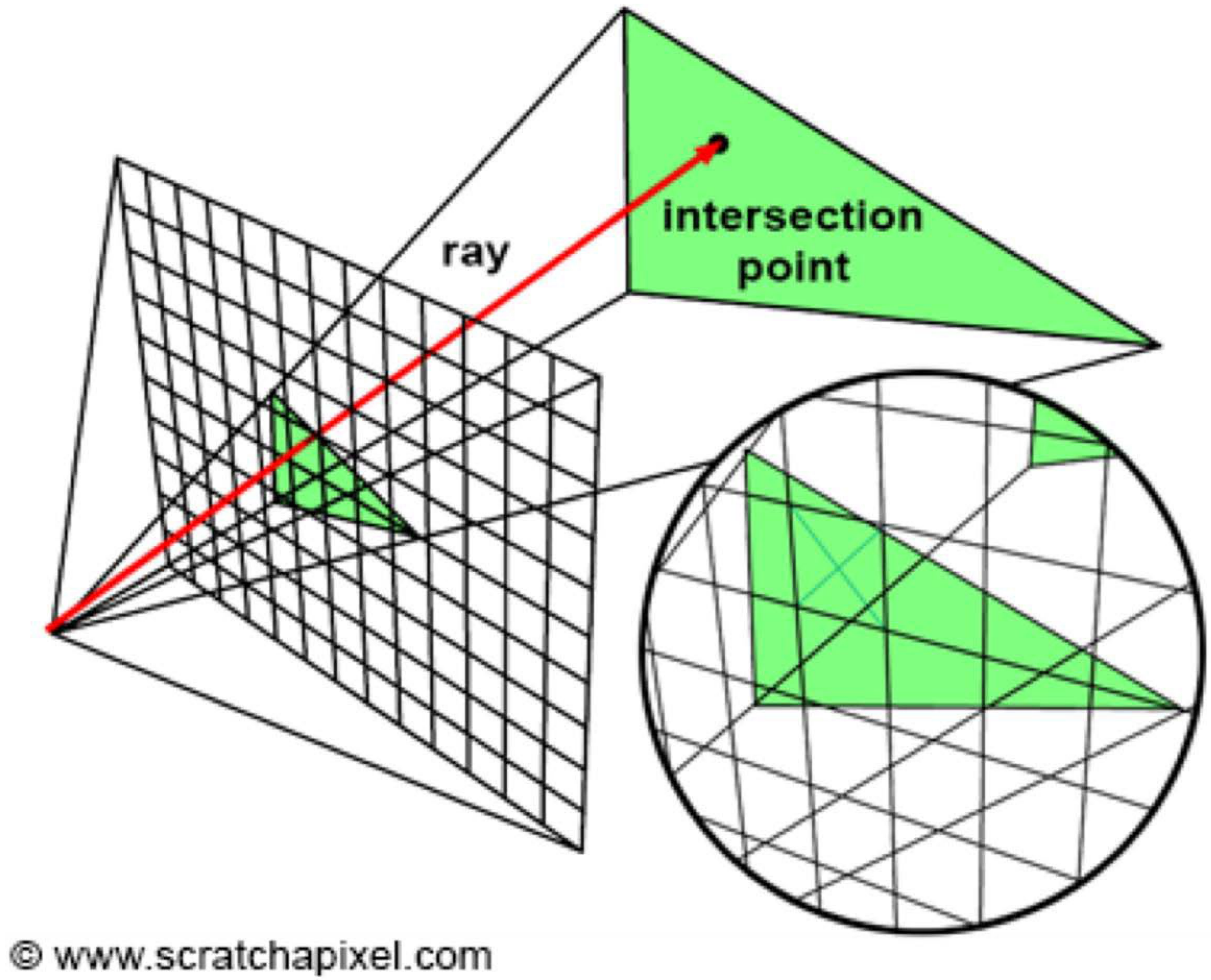

Fortnite | Epic Games

# Rasterization

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


operations can be done quickly on the GPU!

# Ray Casting




https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation

# 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
        }
    }
}
```

# Basic Components of Ray Casting

Ray

Camera

Intersection Tests

# Basic Components of Ray Casting

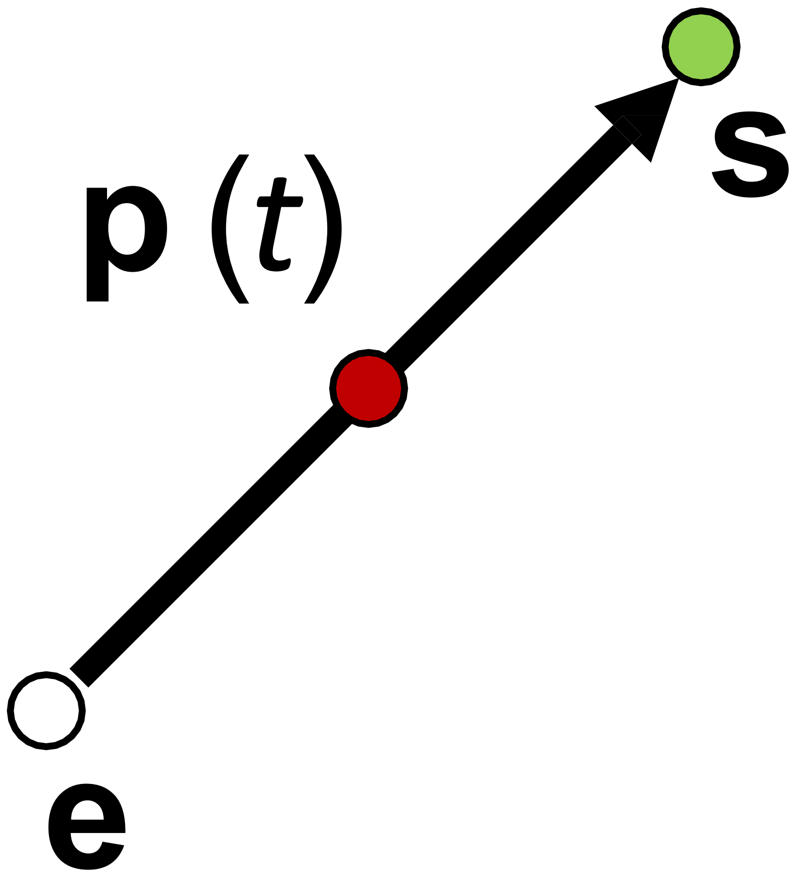
Ray

Camera

Intersection Tests

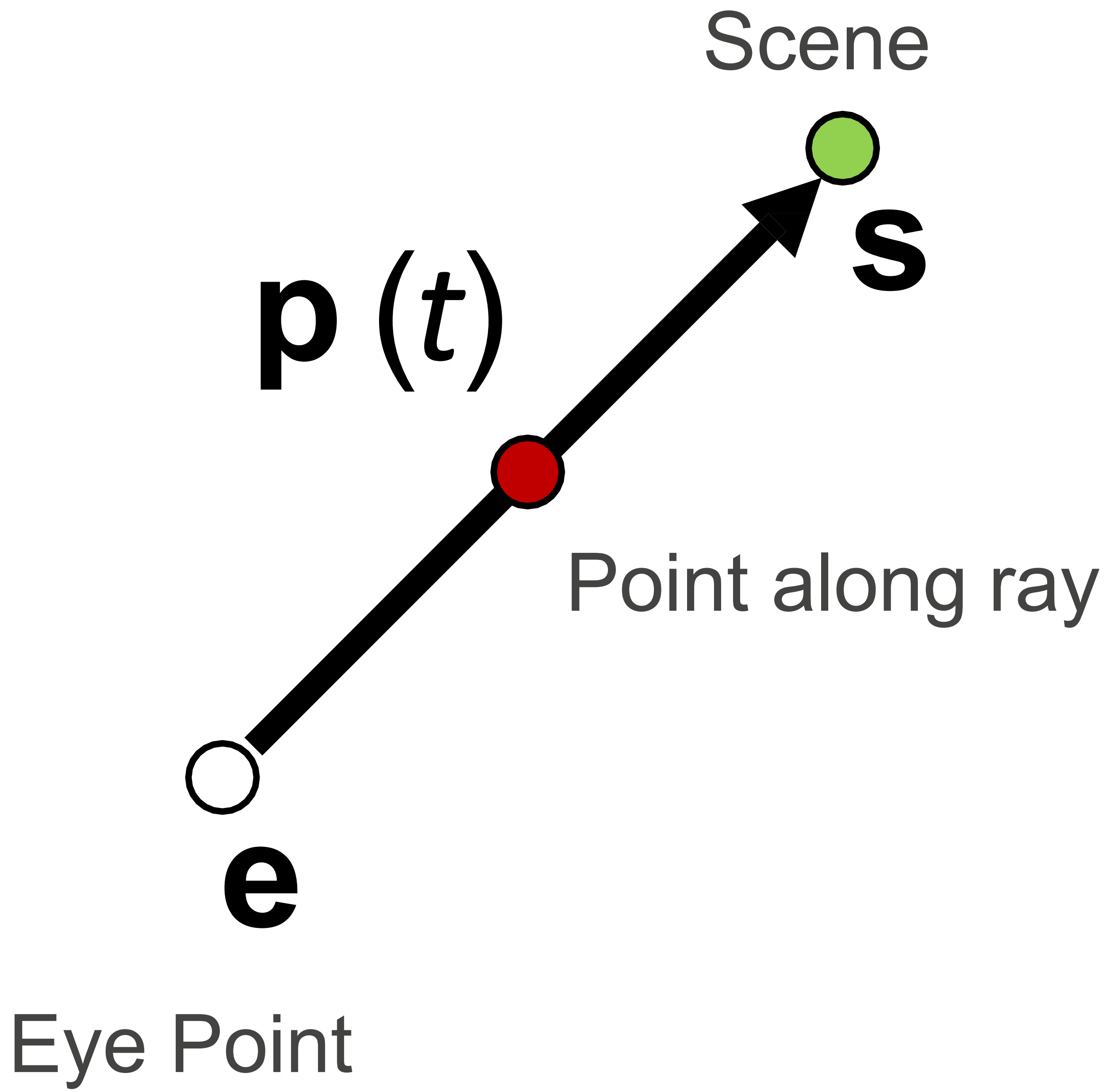
# The Ray

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



# The Ray

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

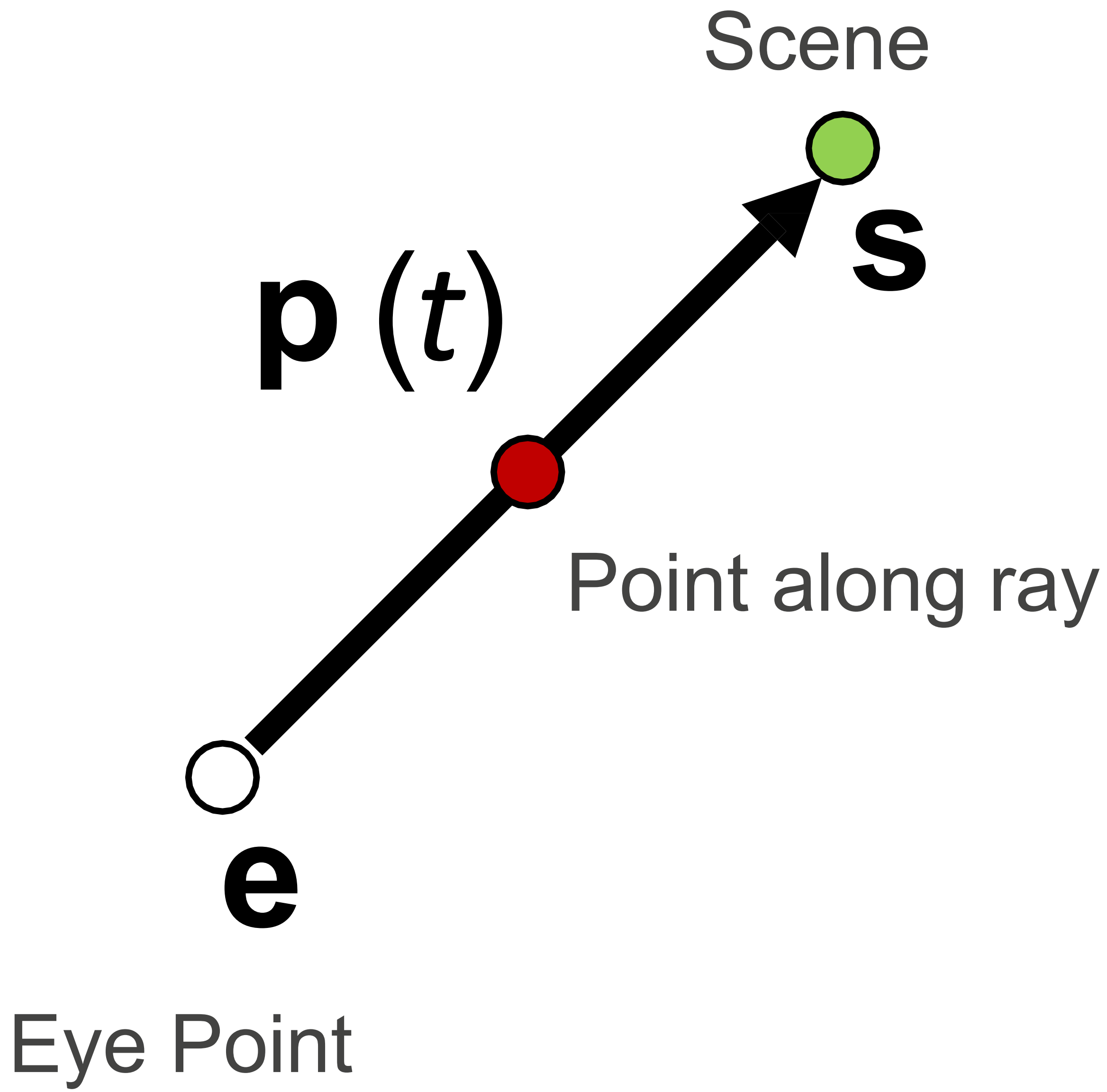


# The Ray

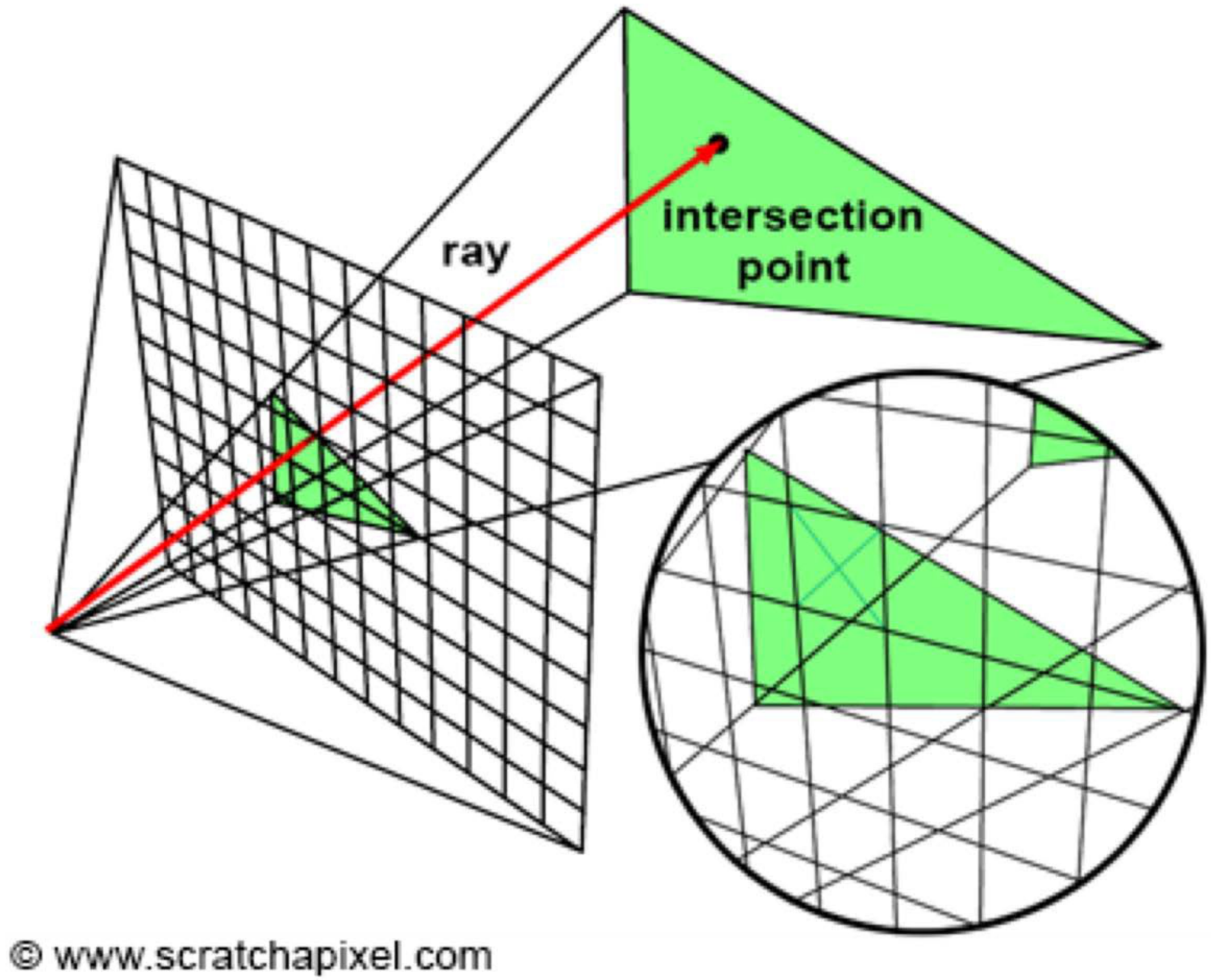
$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

$$\mathbf{p}(0) = \mathbf{e}$$

$$\mathbf{p}(1) = \mathbf{s}$$

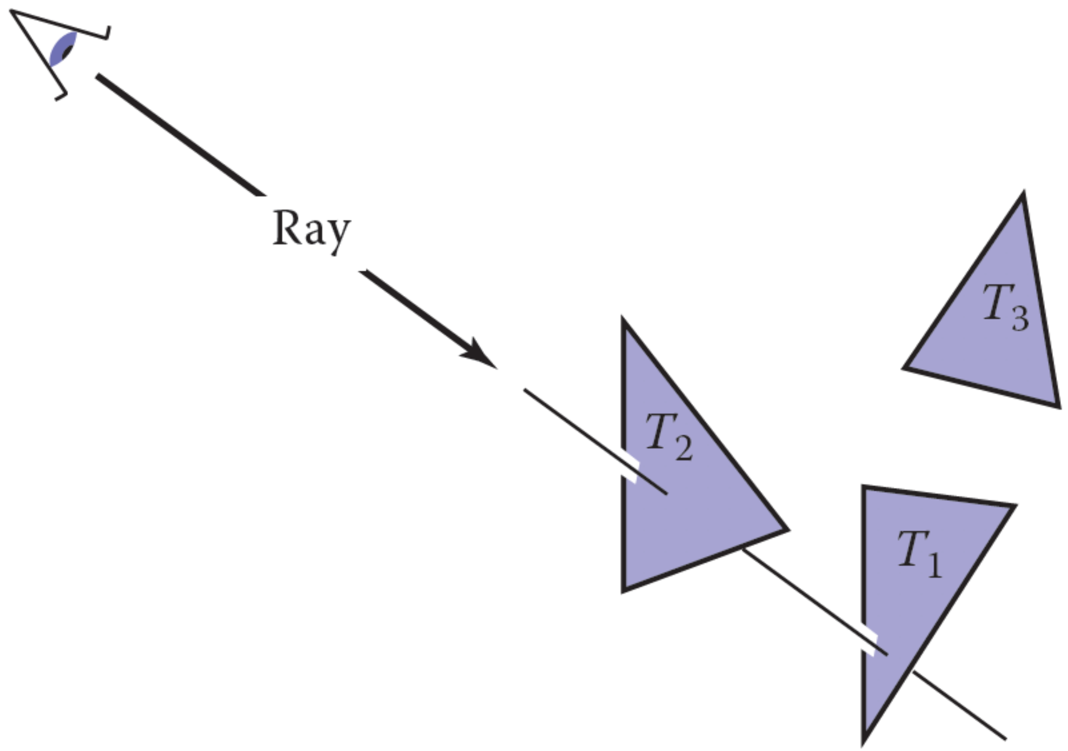


# Ray Casting




https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation

# Ray Casting



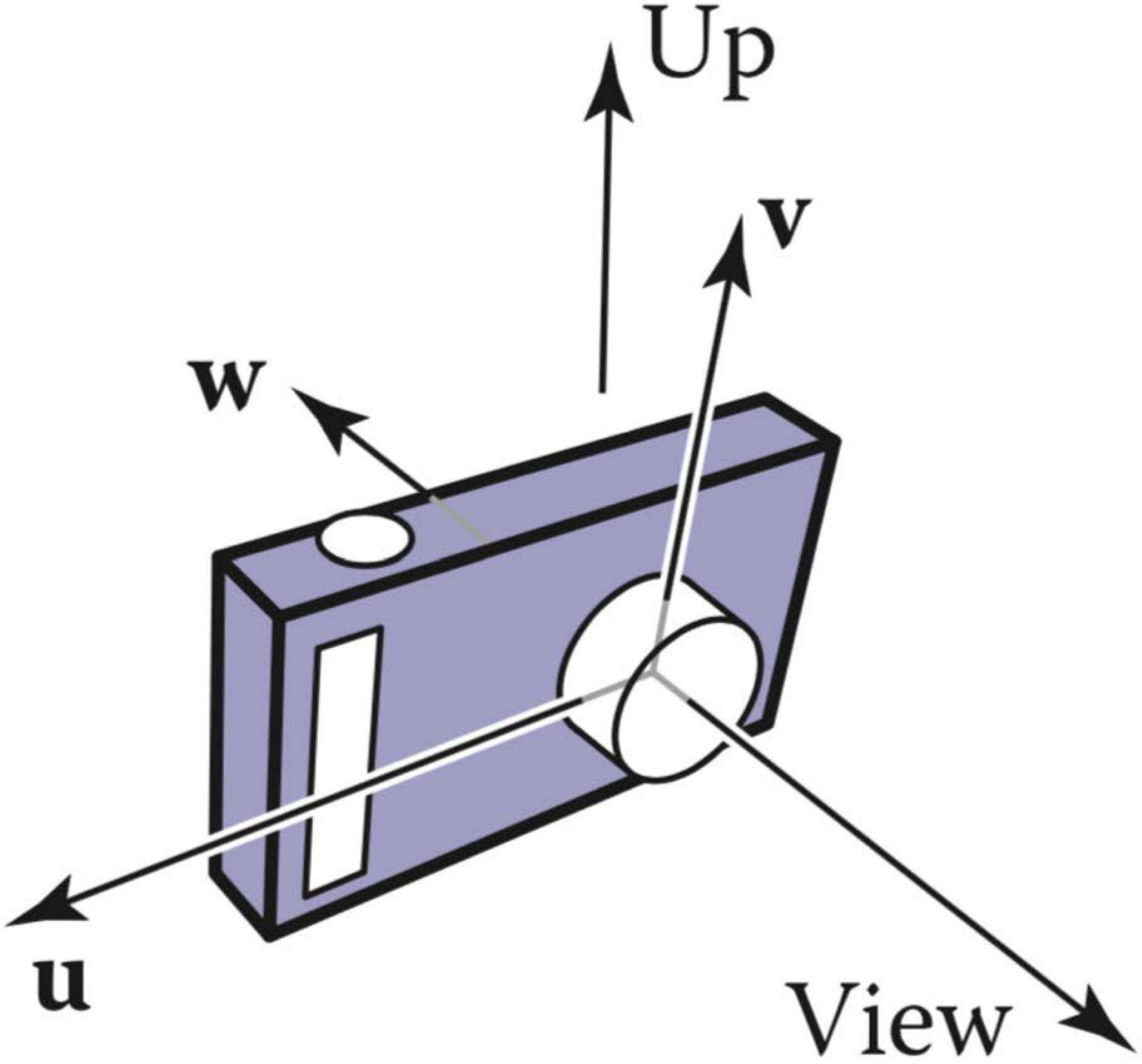
# Basic Components of Ray Casting

Ray

Camera

Intersection Tests

# The Camera

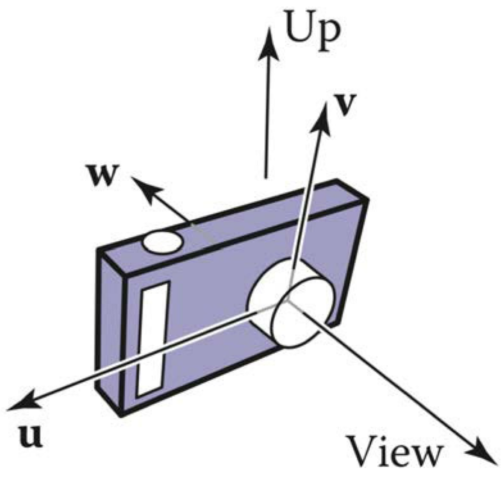


# The Camera

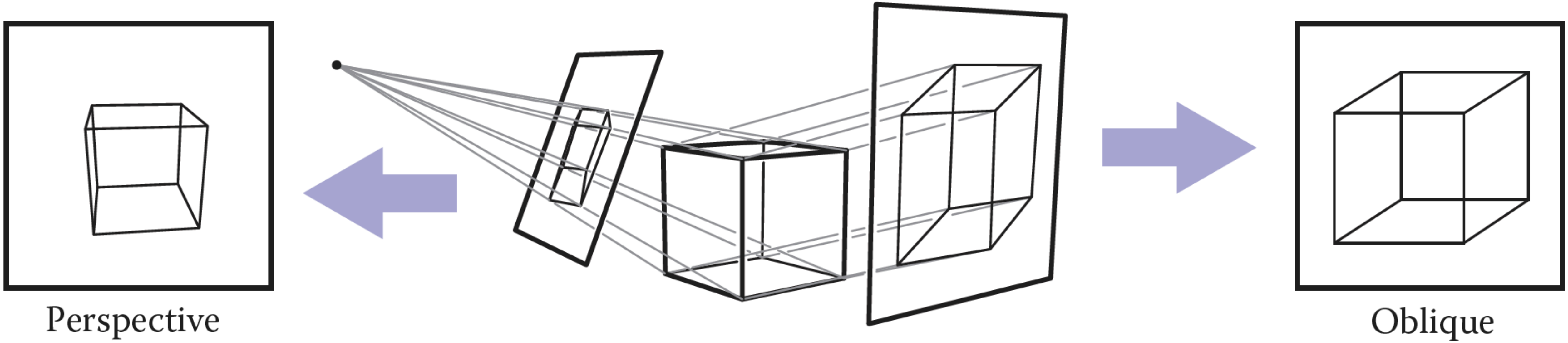
$$\mathbf{w} = -\frac{\text{View}}{\|\text{View}\|}$$

$$\mathbf{u} = \text{View} \times \text{Up}$$

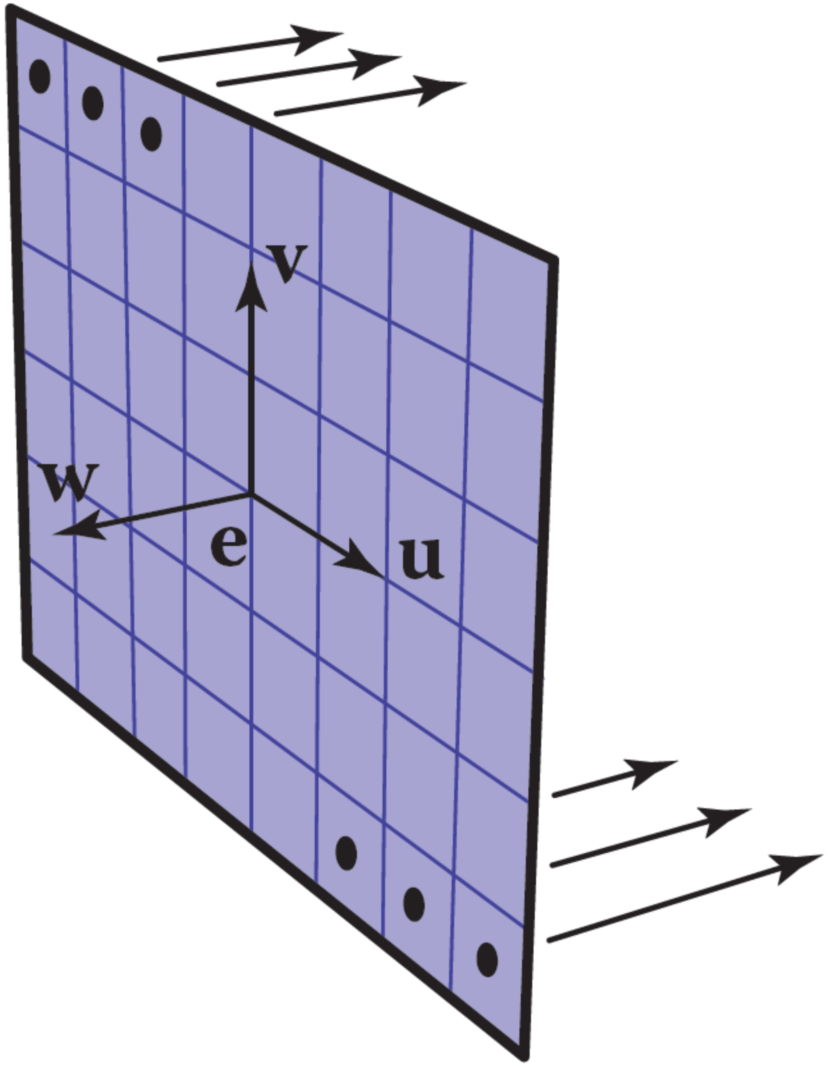
$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$



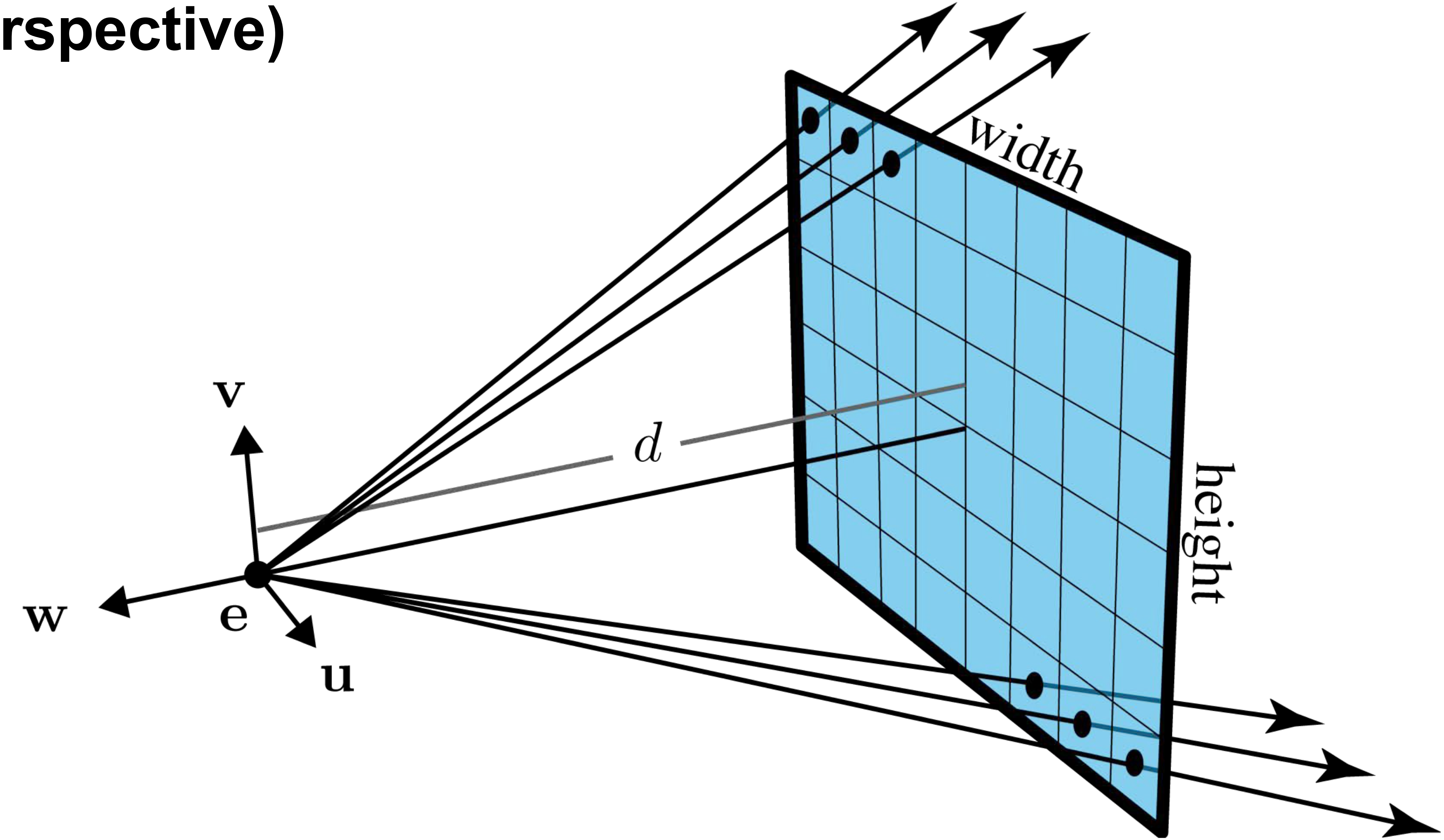
# Orthographic v Perspective Projection



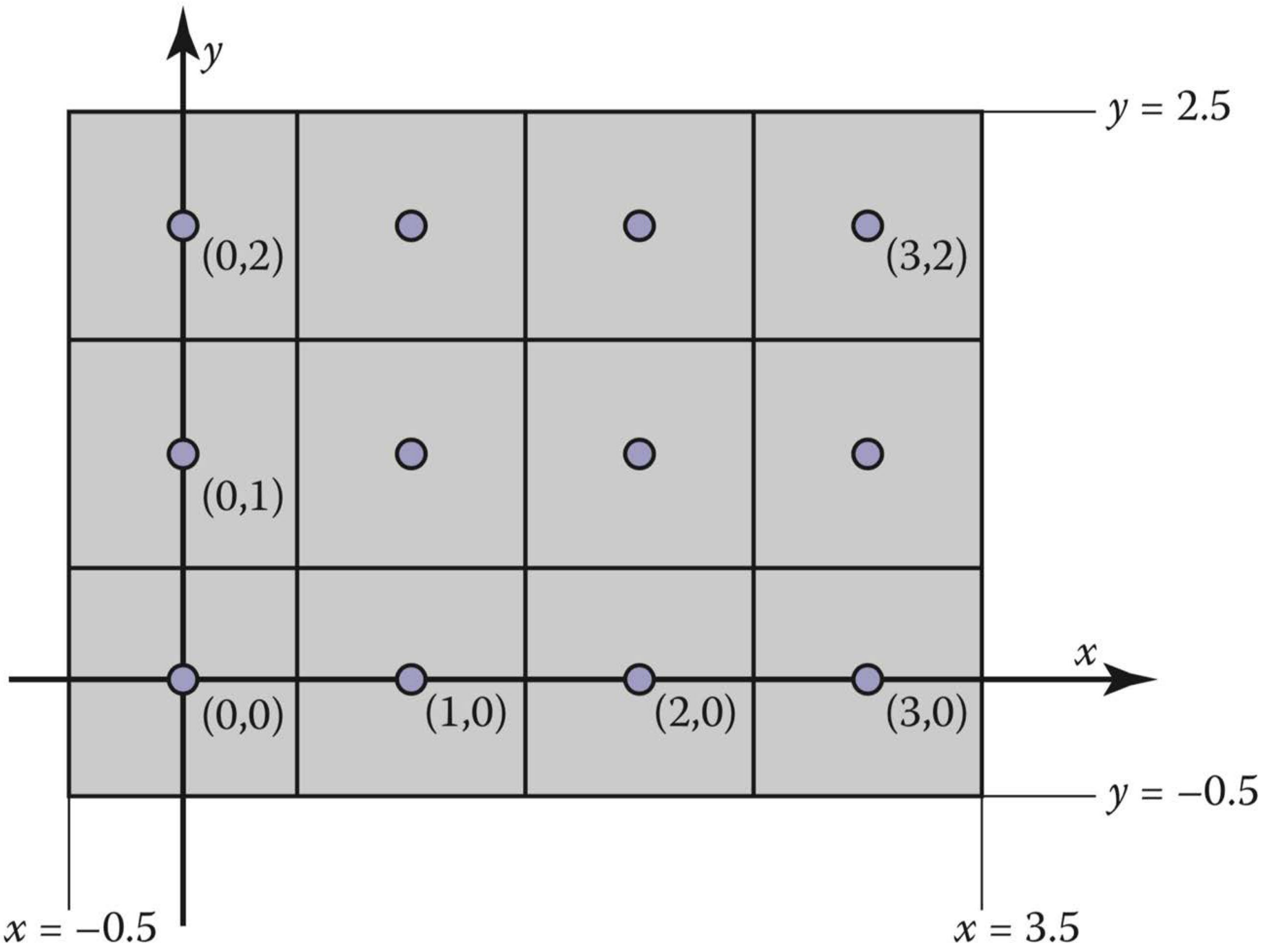
# Generating Rays (Orthographic)

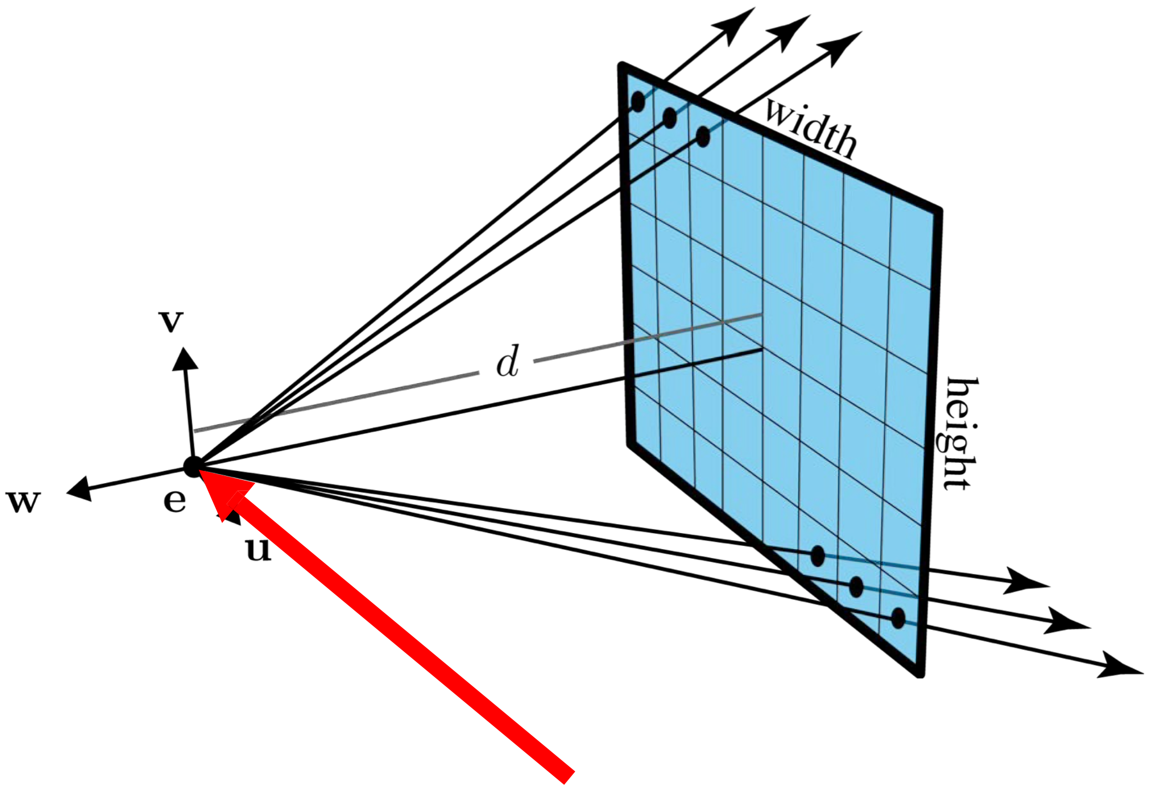


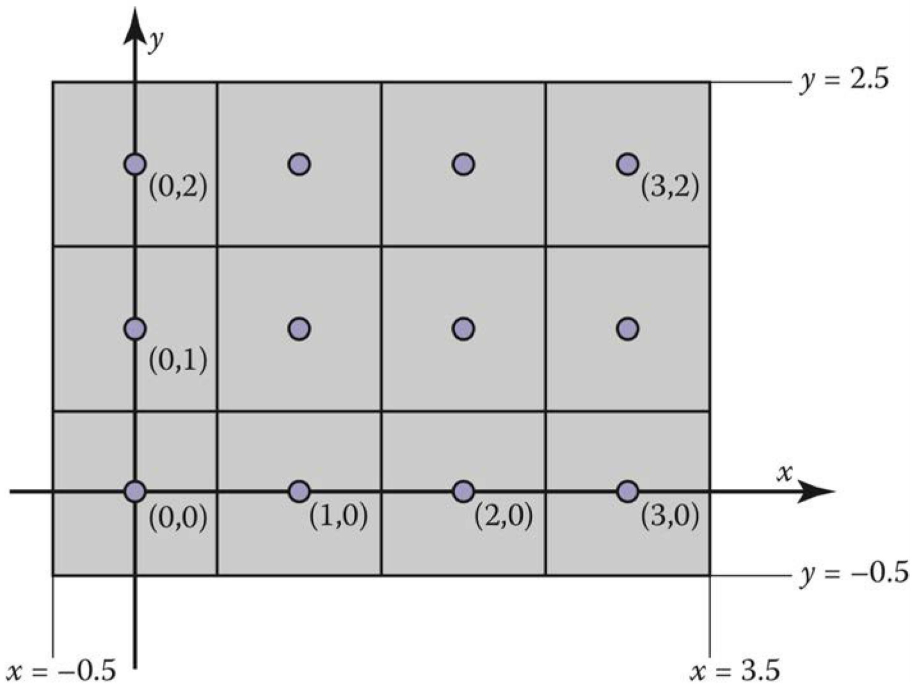
# Generating Rays (Perspective)



# Recall: Standard Pixel Coordinate System

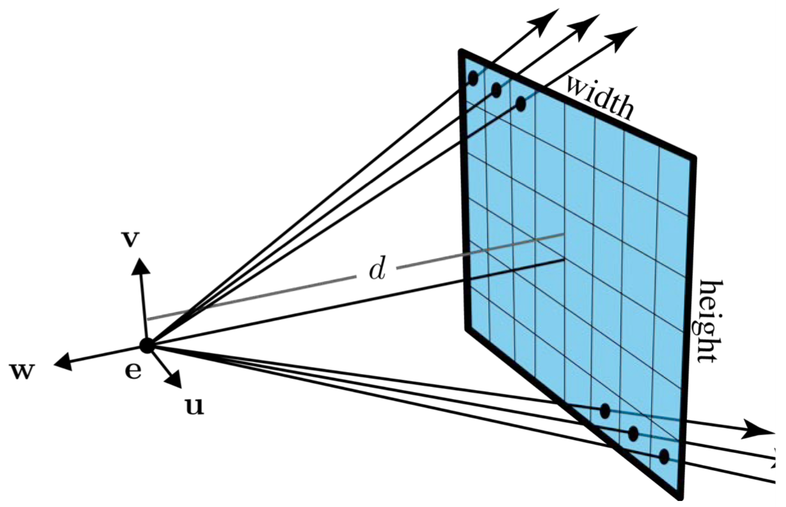




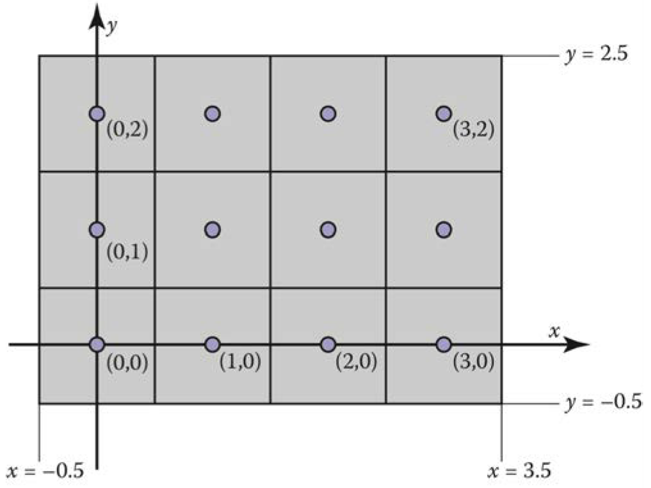




Origin of camera frame (the eye)

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




Camera space




Pixel space

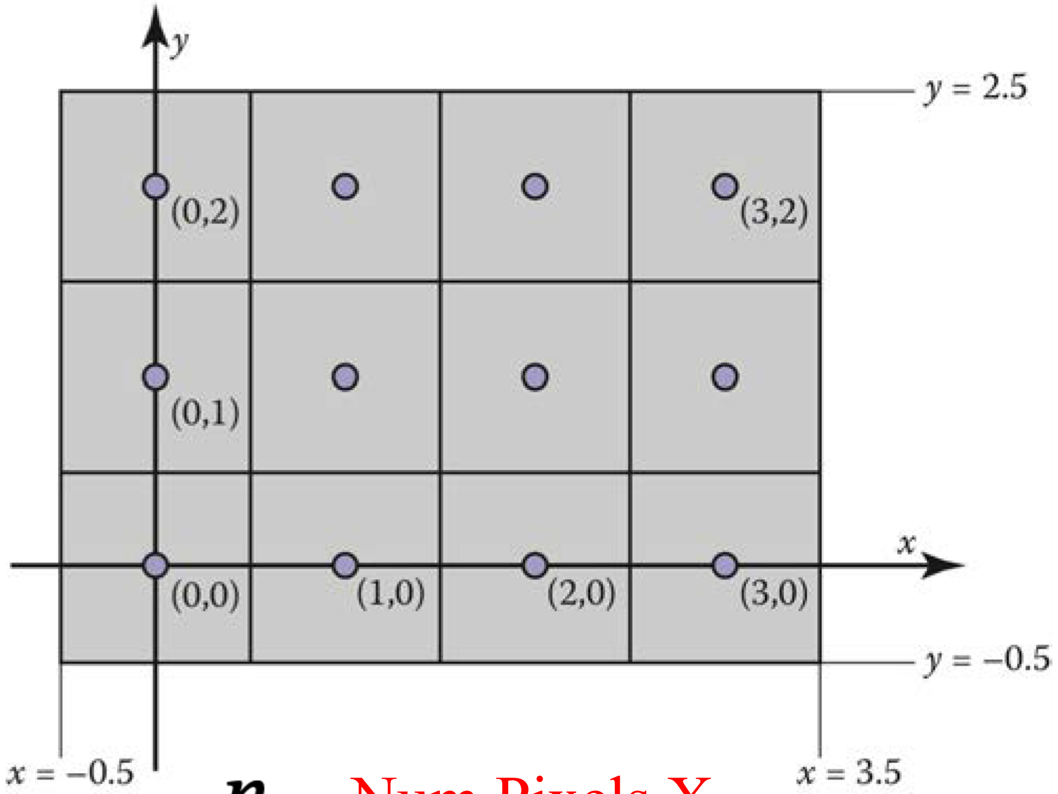
Bottom Left Corner $(i, j)$ : ?

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

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

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

$n_x$ Num Pixels Y




$n_y$ Num Pixels X

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

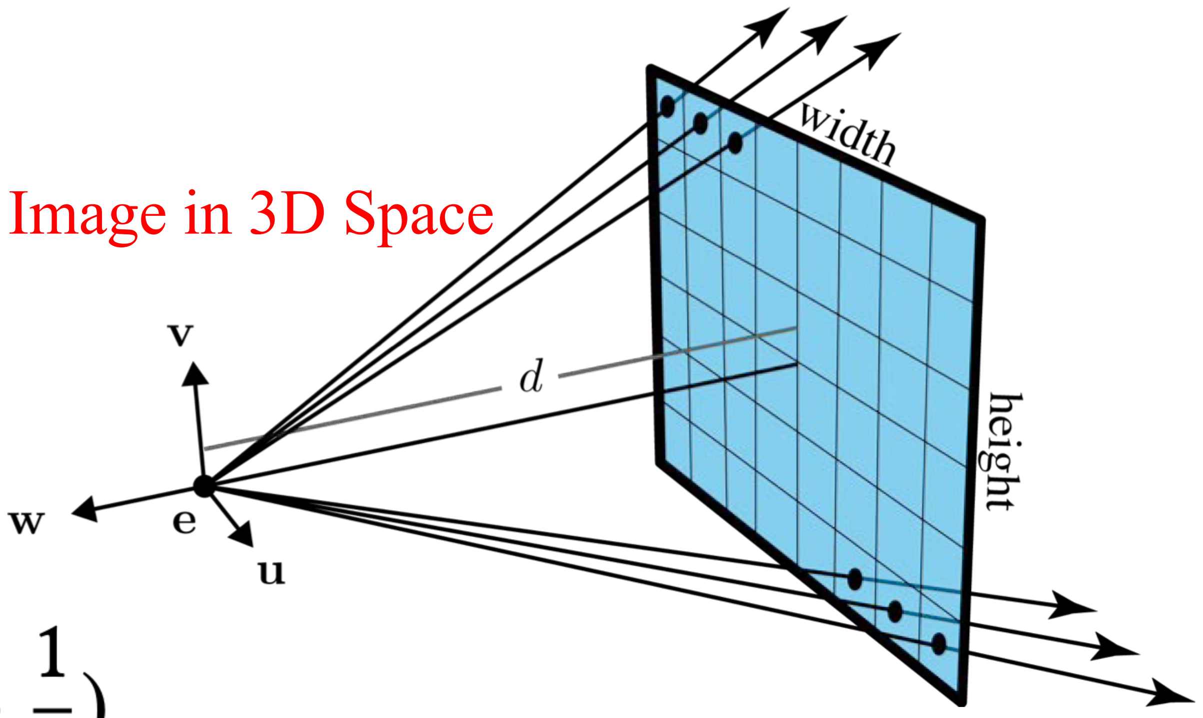
Top Right Corner $(i, j)$ : $(n_x - \frac{1}{2}, n_y - \frac{1}{2})$

Bottom Left Corner $(u, v)$ :

Top Right Corner $(u, v)$ :

Physical Width of Image in 3D Space

Physical Height of Image in 3D Space



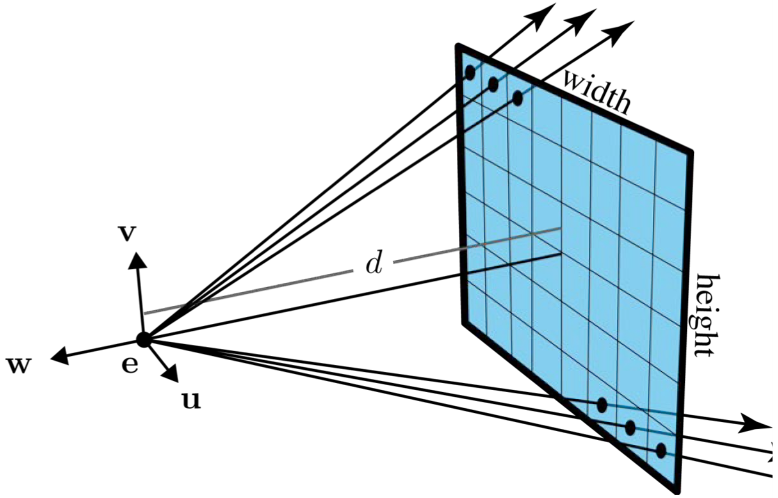

Bottom Left Corner $(i, j)$ : $\left(-\frac{1}{2}, -\frac{1}{2}\right)$

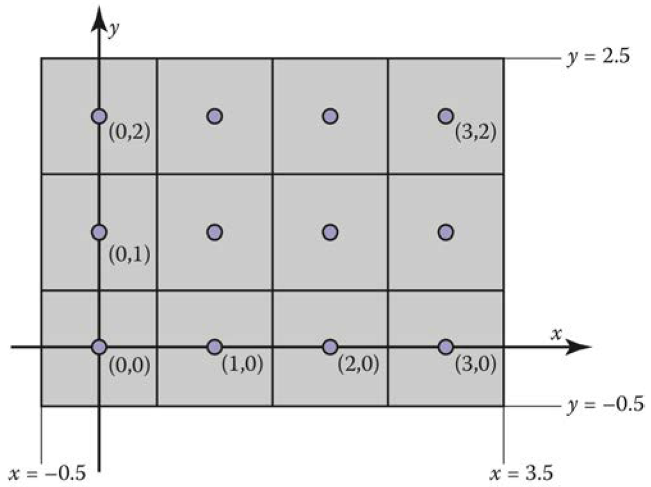
Top Right Corner $(i, j)$ : $\left(n_x - \frac{1}{2}, n_y - \frac{1}{2}\right)$

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

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

Num Pixels X

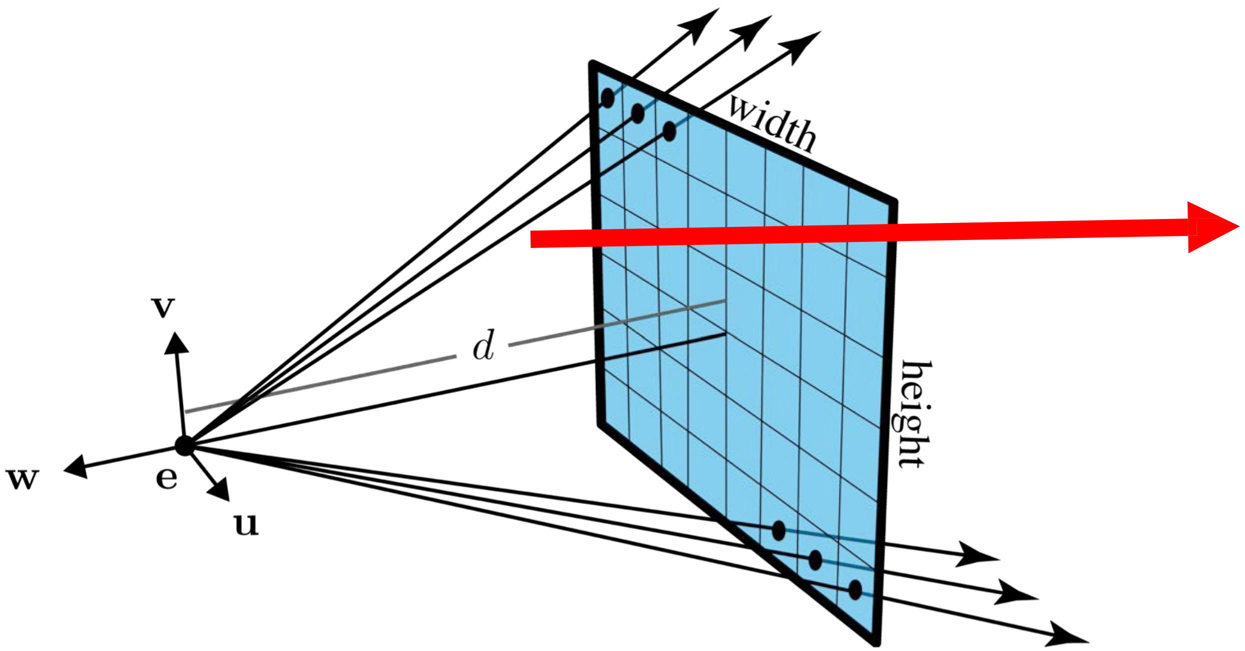
$$u = \frac{\text{width}}{n_x} \cdot \left(i + \frac{1}{2}\right) - \frac{\text{width}}{2}$$

Physical Height of Image in 3D Space

Num Pixels Y

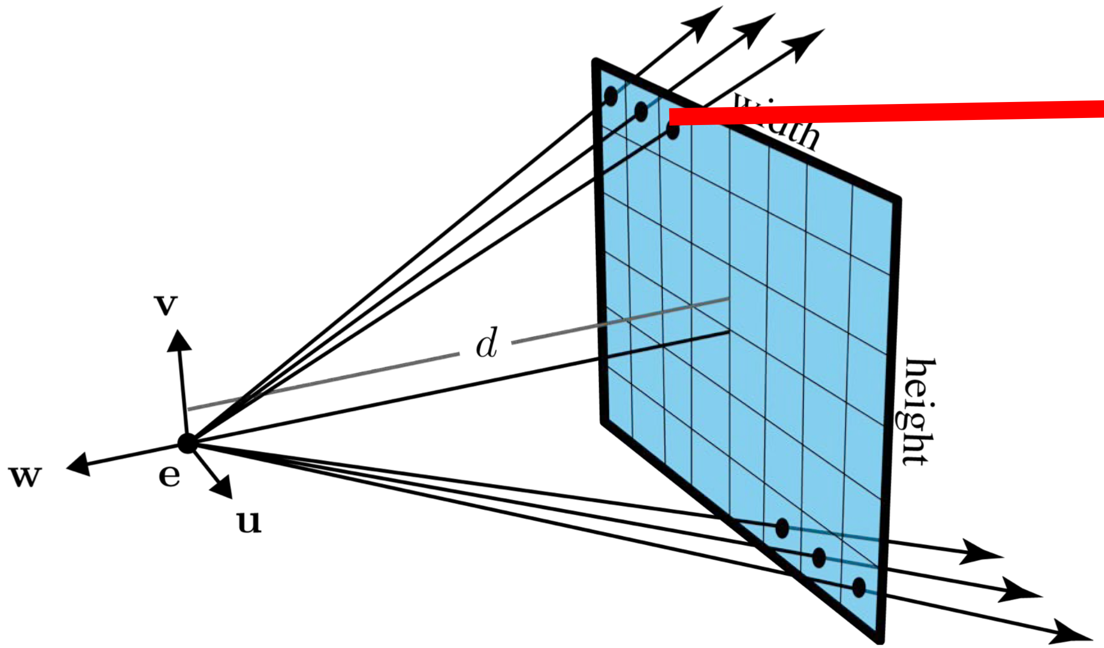
$$v = \frac{\text{height}}{n_y} \cdot \left(j + \frac{1}{2}\right) - \frac{\text{height}}{2}$$

# Ray Equation in Camera Space




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

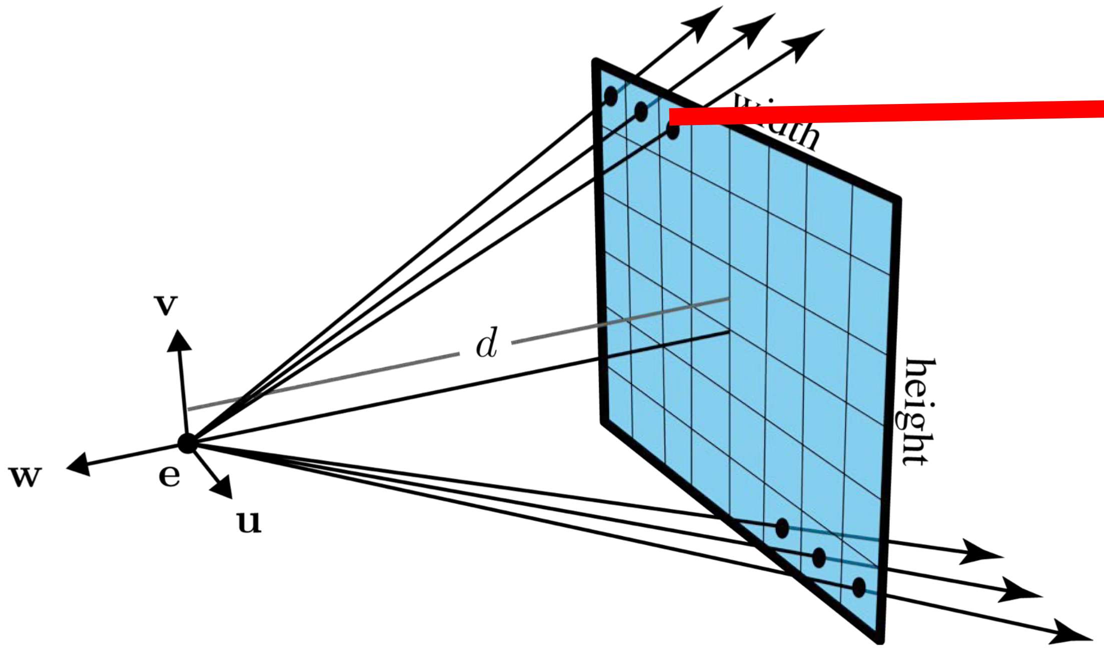
# Ray Equation in Camera Space




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

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

# Ray Equation in Camera Space




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

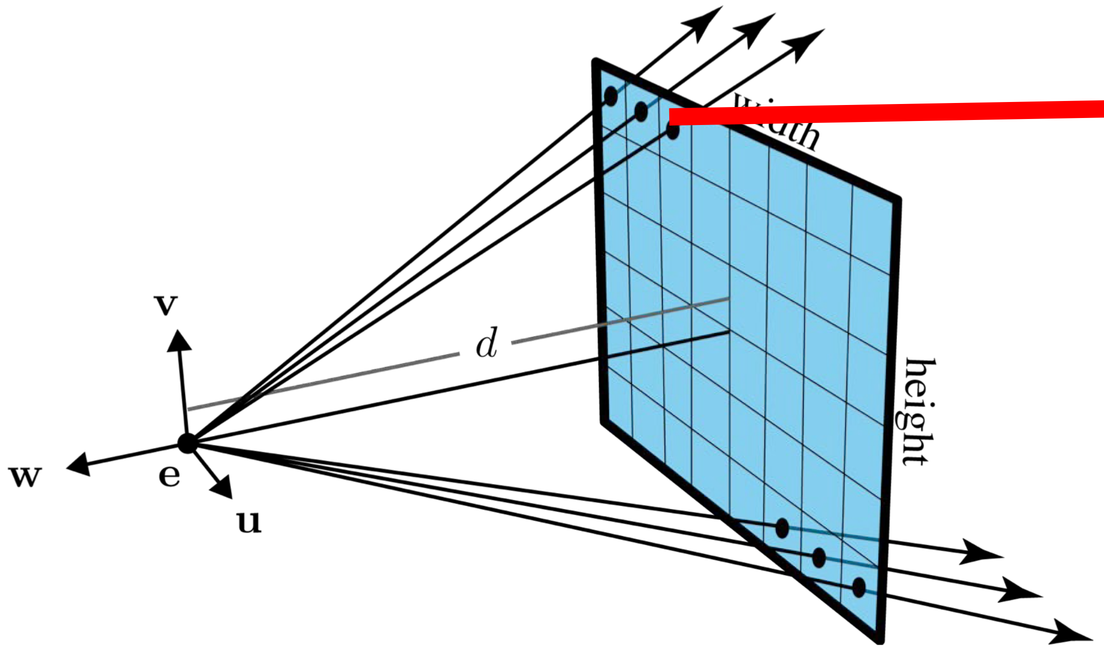
$$\mathbf{p}(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \left( \begin{bmatrix} u(i) \\ v(j) \\ -d \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

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

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

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

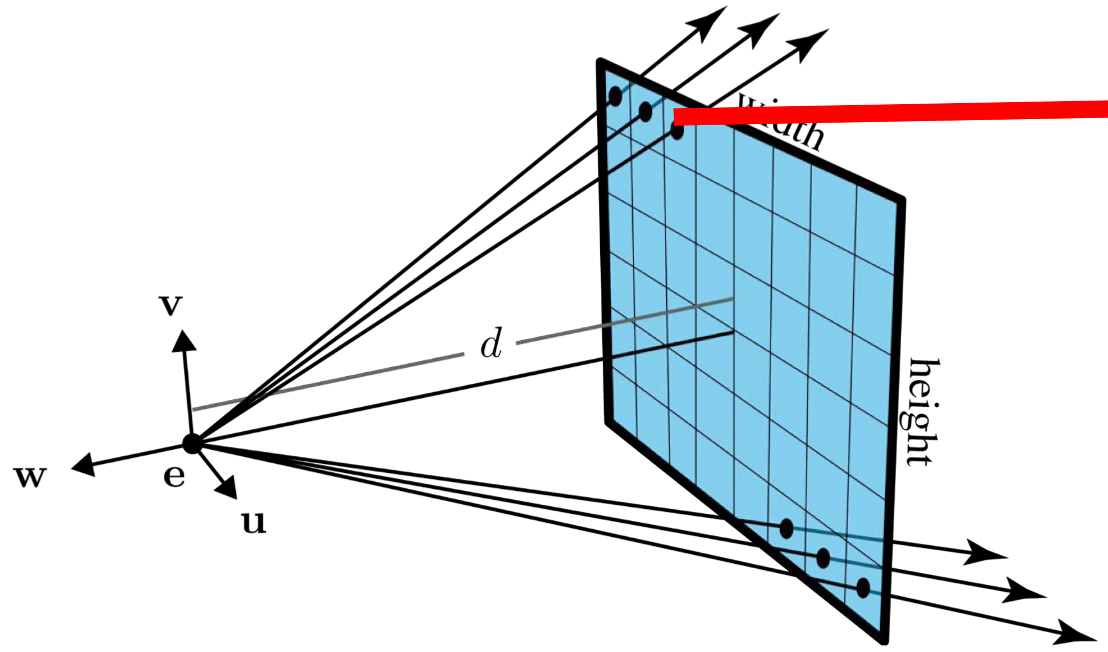
# Ray Equation in World Space




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

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

# Ray Equation in World Space




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

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

*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
        }
    }
}
```

# Next class

Ray Intersections