Computer Graphics

Lecture 7

General Perspective Cameras

Orthographic Cameras

Announcements

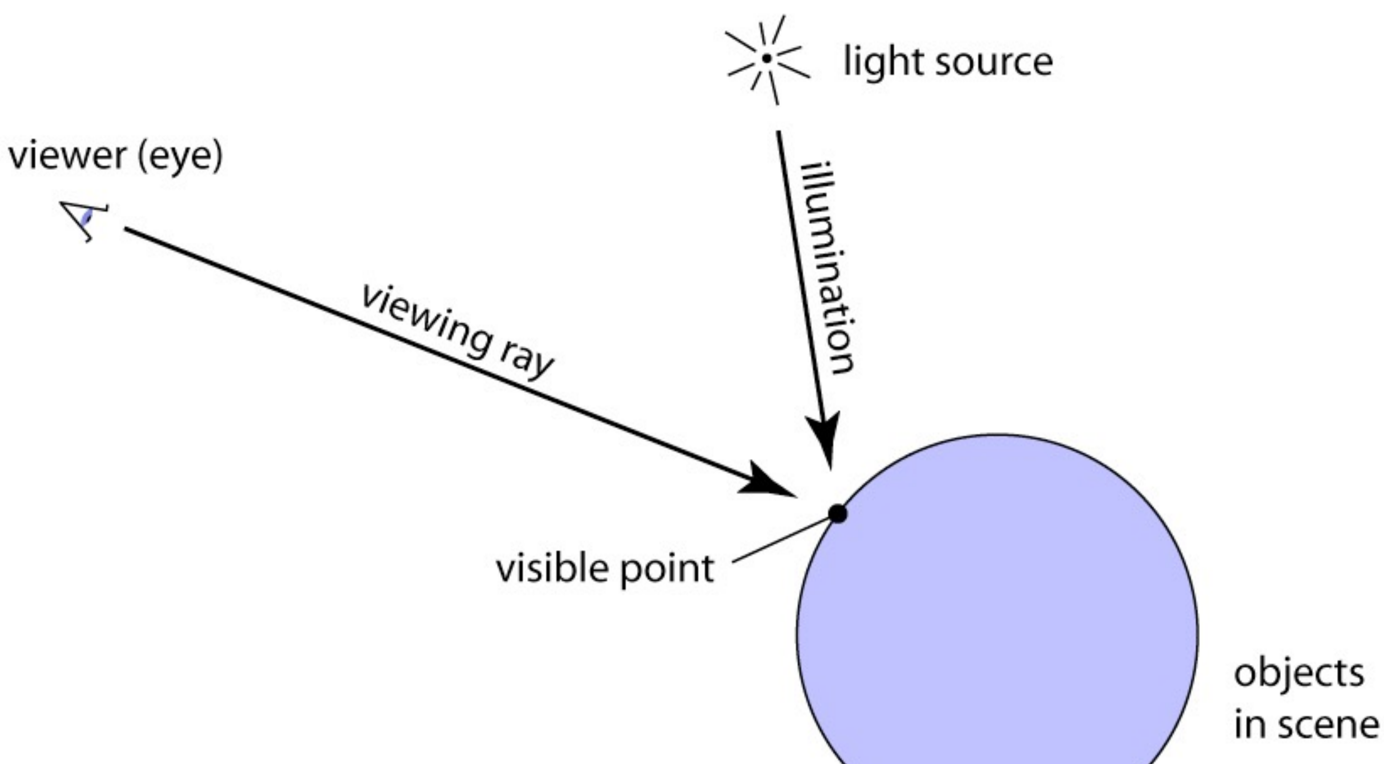
- Grading turnaround target: 1 week
  - It's not realistic to grade HW[i] before A[i] deadline.
  - But you can check your math with classmates (esp. after the HW[i] deadline)
  - And, this is graphics: if you did the math wrong, the results will (probably?) look wrong!

Goals

- Know how to generate viewing rays for *general* perspective and orthographic cameras
- Know how to construct a camera basis given **eye**, **view**, and **up** vectors.
- Be aware of some common members of the perspective and orthographic families of projections.

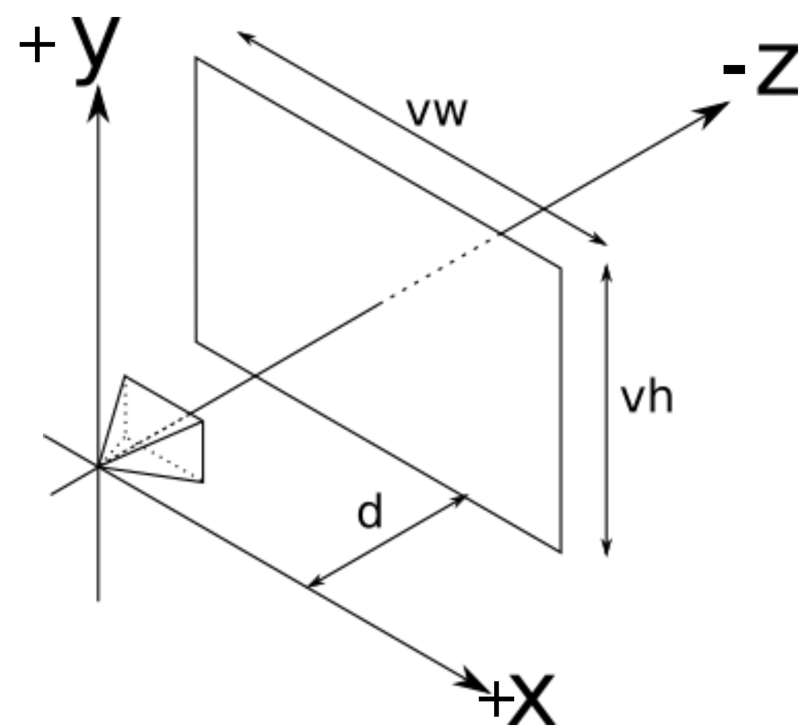
Ray Tracing: Pseudocode

```
for each pixel:
    generate a viewing ray for the pixel
    find the closest object it intersects
    determine the color of the object
```



A "canonical" camera

- Eye is at the origin (0, 0, 0)
- Looking down the **negative** z axis
- Viewport is aligned with the xy plane
- vh = vw = 1
- d = 1



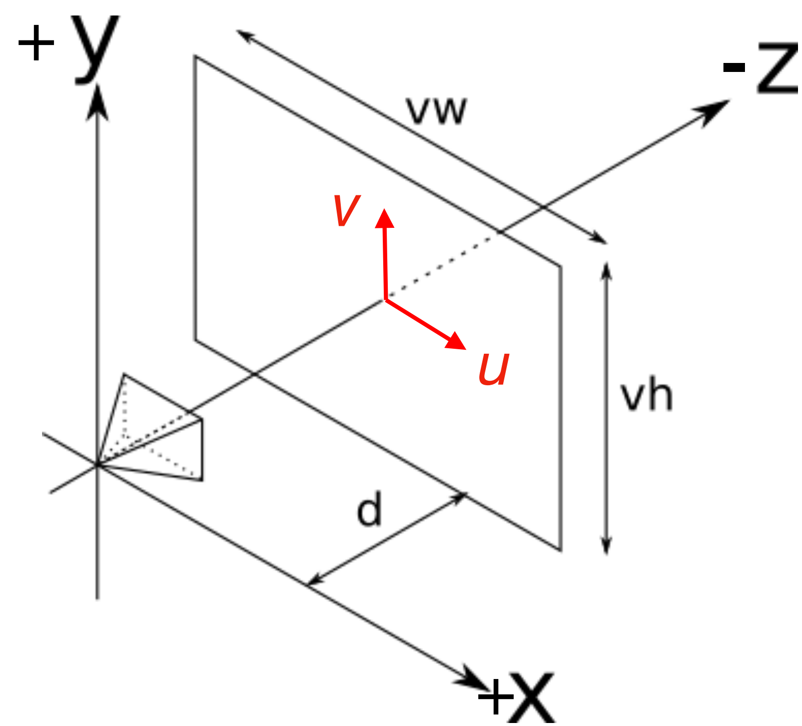
Canonical Perspective Camera: Viewing Rays

$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

Origin (**p**): (0, 0, 0)
Direction (**d**): (u, v, -1)

- Eye is at the origin (0, 0, 0)
- Looking down the **negative** z axis
- Viewport is aligned with the xy plane
- vh = vw = 1
- d = 1



More General Cameras

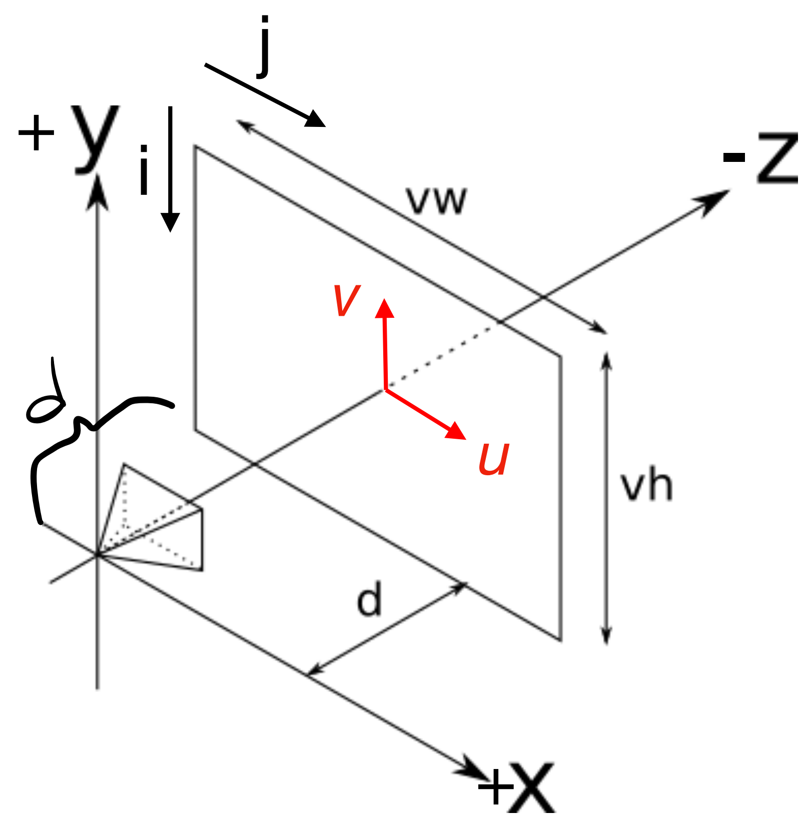
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

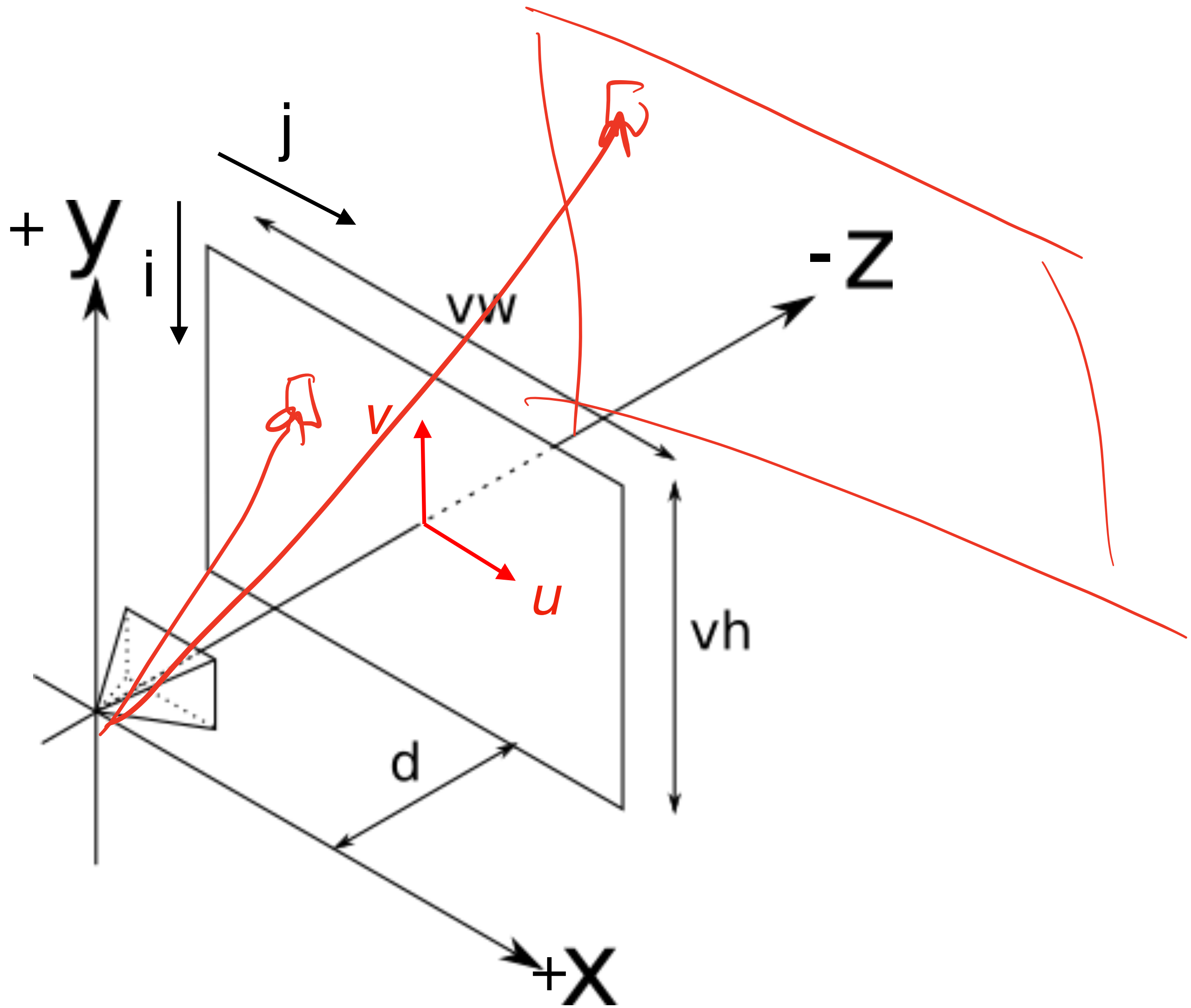
Origin (**p**): (0, 0, 0)
Direction (**d**): (*u*, *v*, -d)

Let's break some assumptions!

- **d = 1**
- vh = vw = 1
- Eye is at the origin (0, 0, 0)
- Looking down the **negative** z axis



d != 1



More General Cameras

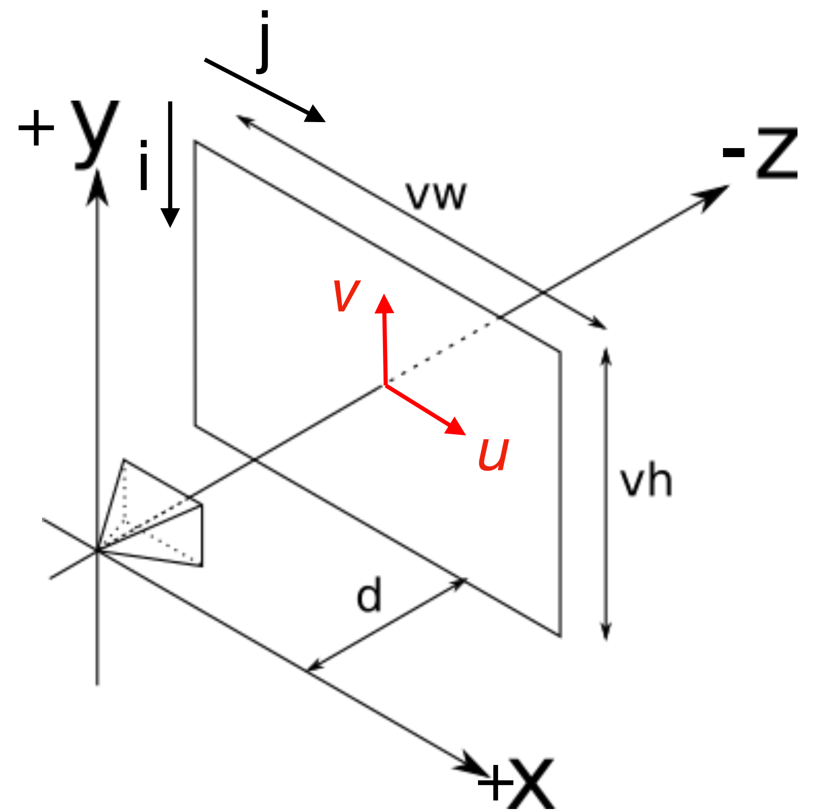
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

Origin (**p**): (0, 0, 0)
Direction (**d**): (*u*, *v*, -d)

Let's break some assumptions!

- **d = 1**
- vh = vw = 1
- Eye is at the origin (0, 0, 0)
- Looking down the **negative** z axis

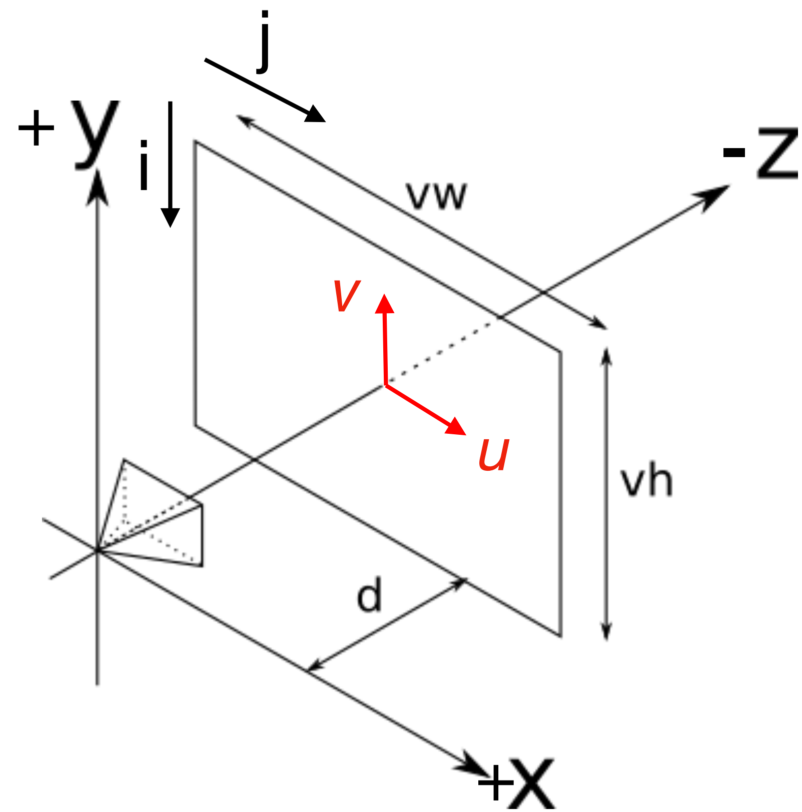


More General Cameras

$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

Origin (**p**): (0, 0, 0)
Direction (**d**): (*u*, *v*, -1)



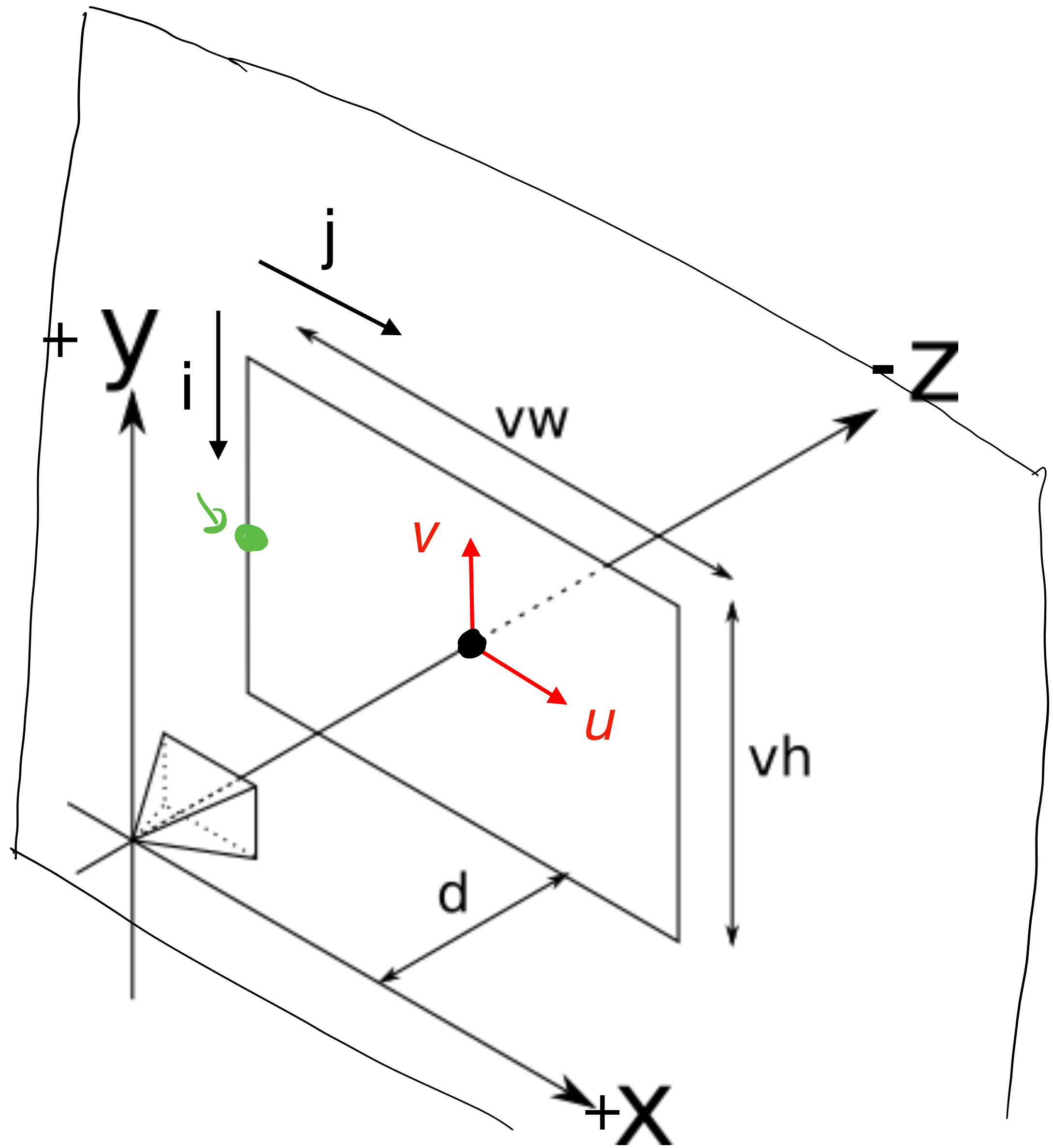
Let's break some assumptions!

- d = 1
- **vh = vw = 1**
- Eye is at the origin (0, 0, 0)
- Looking down the **negative** z axis

vw != vw != 1

$$u = \left(\frac{j - \frac{1}{2}}{W} - \frac{1}{2}\right) vw$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right) \cdot vh$$



Origin (**p**): (0, 0, 0)

Direction (**d**): (*u*, *v*, -1)

More General Cameras

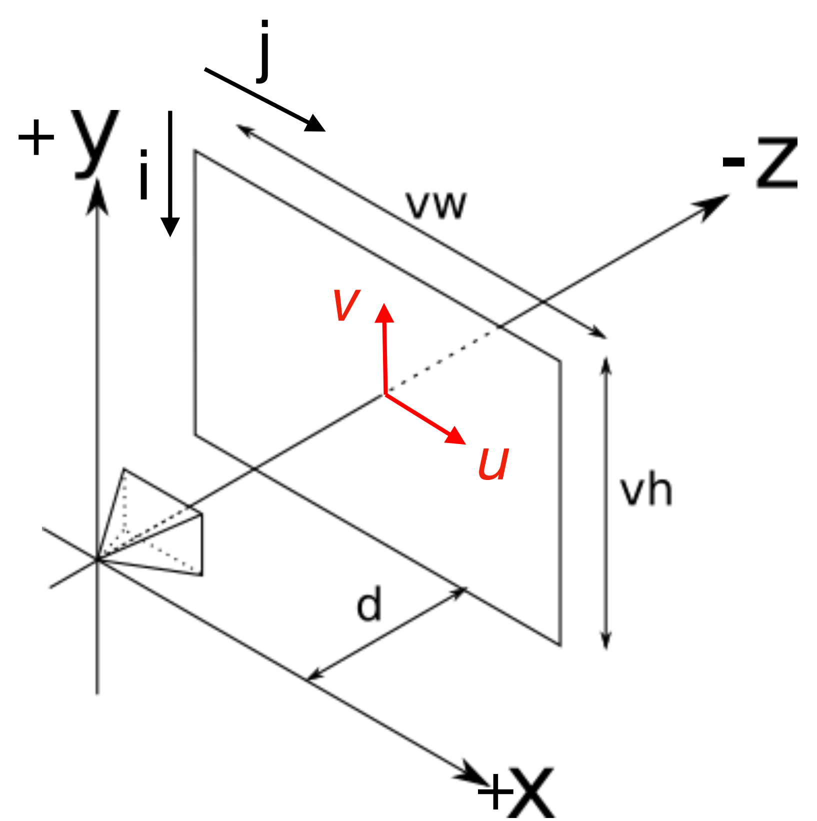
$$u = \left(\frac{j - \frac{1}{2}}{W} - \frac{1}{2} \right) * vw$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2} \right) * vh$$

Origin (**p**): (0, 0, 0)
Direction (**d**): (*u*, *v*, -1)

Let's break some assumptions!

- d = 1
- **vh = vw = 1**
- Eye is at the origin (0, 0, 0)
- Looking down the **negative** z axis



More General Cameras

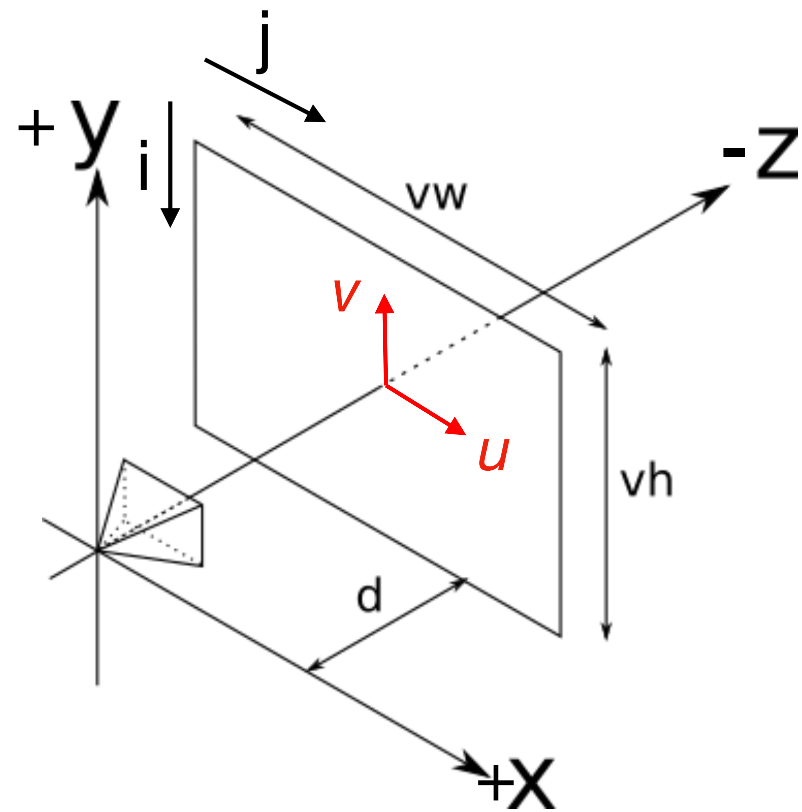
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

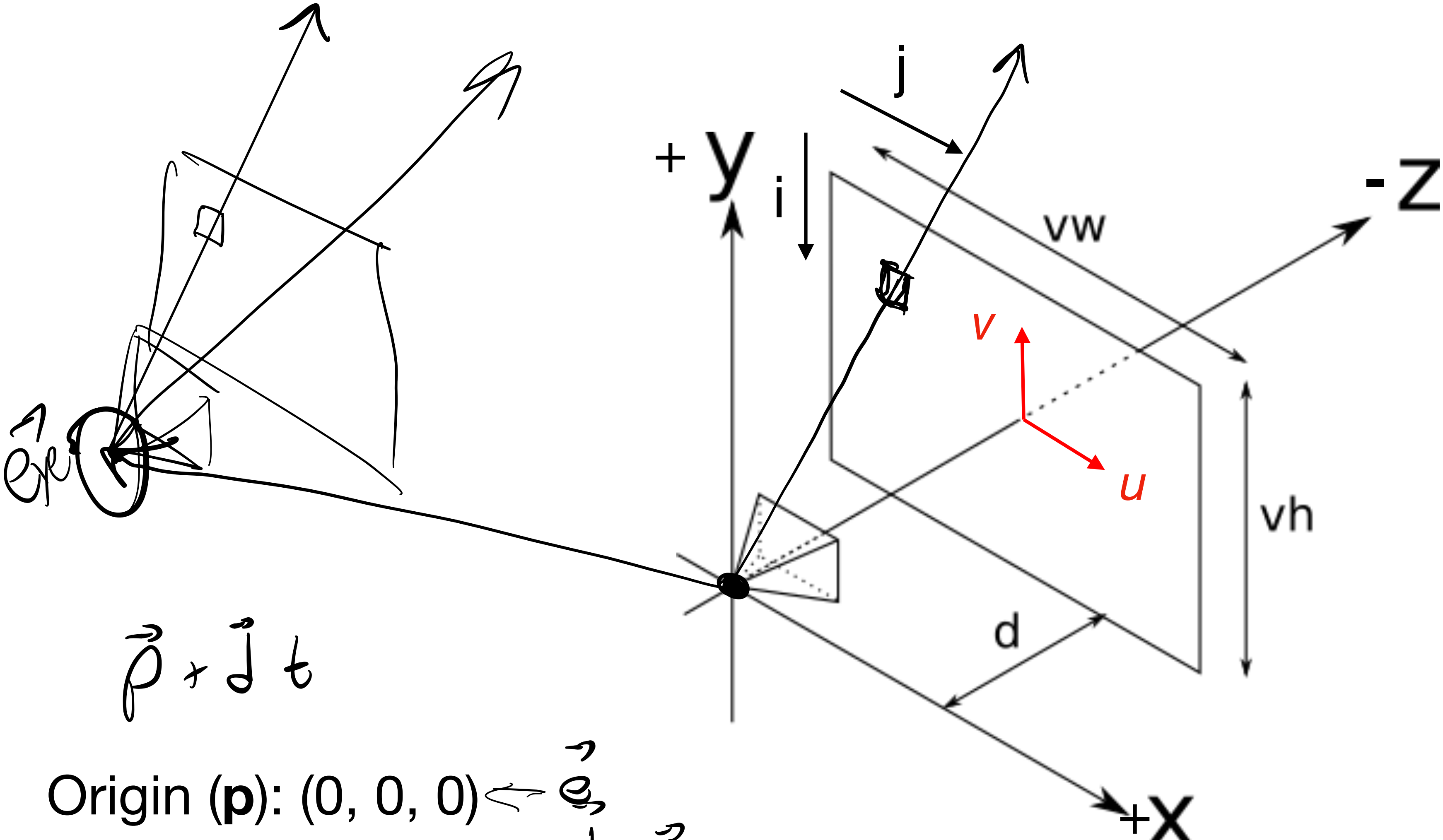
Origin (**p**): (0, 0, 0)
Direction (**d**): (*u*, *v*, -1)

Let's break some assumptions!

- d = 1
- vh = vw = 1
- **Eye is at the origin (0, 0, 0)**
- Looking down the **negative** z axis



e != (0, 0, 0)



Origin (**p**): (0, 0, 0) ← $\vec{e}$

Direction (**d**): (*u*, *v*, -1) ← $\vec{d} + \vec{e}$

More General Cameras

$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

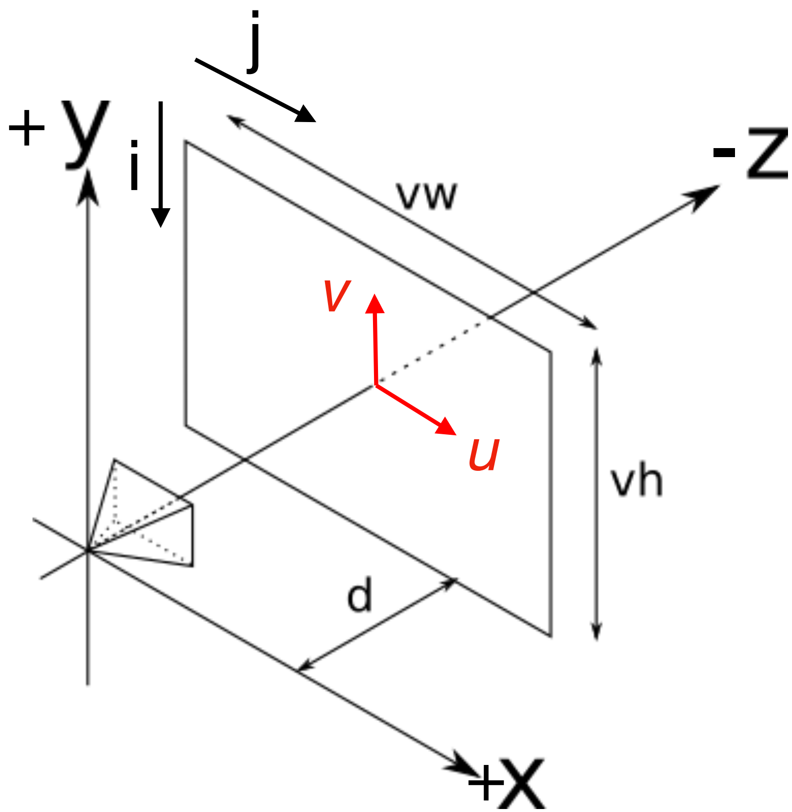
$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

Origin (**p**): (e_x, e_y, e_z)

Direction (**d**): $(u, v, -1)$

Let's break some assumptions!

- d = 1
- vh = vw = 1
- **Eye is at the origin (0, 0, 0)**
- Looking down the **negative** z axis

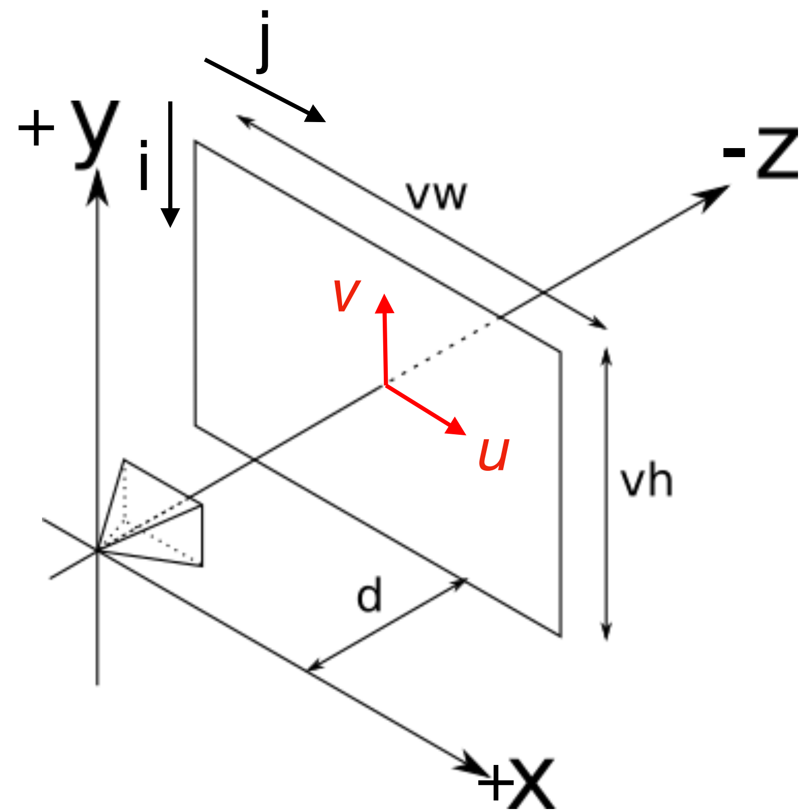


More General Cameras

$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

$$v = -\left(\frac{i - \frac{1}{2}}{H} - \frac{1}{2}\right)$$

Origin (**p**): (0, 0, 0)
Direction (**d**): (*u*, *v*, -1)

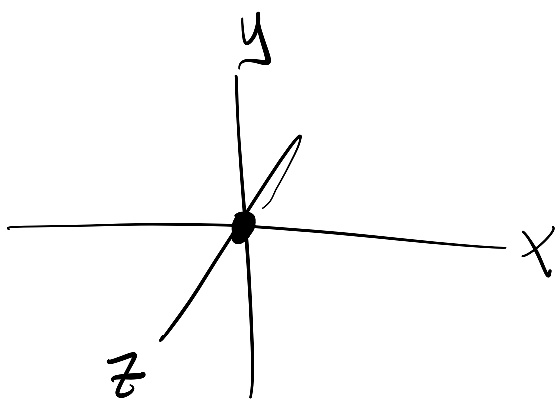


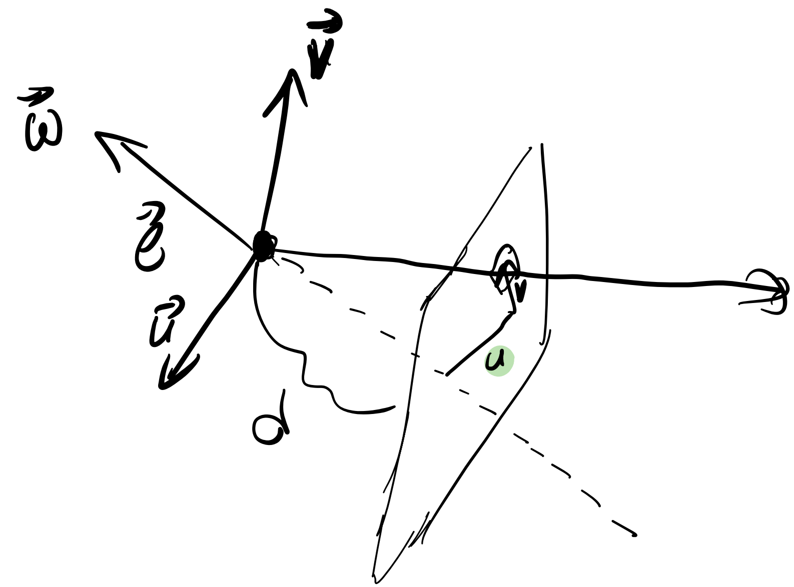
Let's break some assumptions!

- d = 1
- vh = vw = 1
- Eye is at the origin (0, 0, 0)
- **Looking down the negative z axis**

Change of Basis

Reminder: 3B1B video, and Section 2.4.5 - Orthonormal Bases and Coordinate Frames





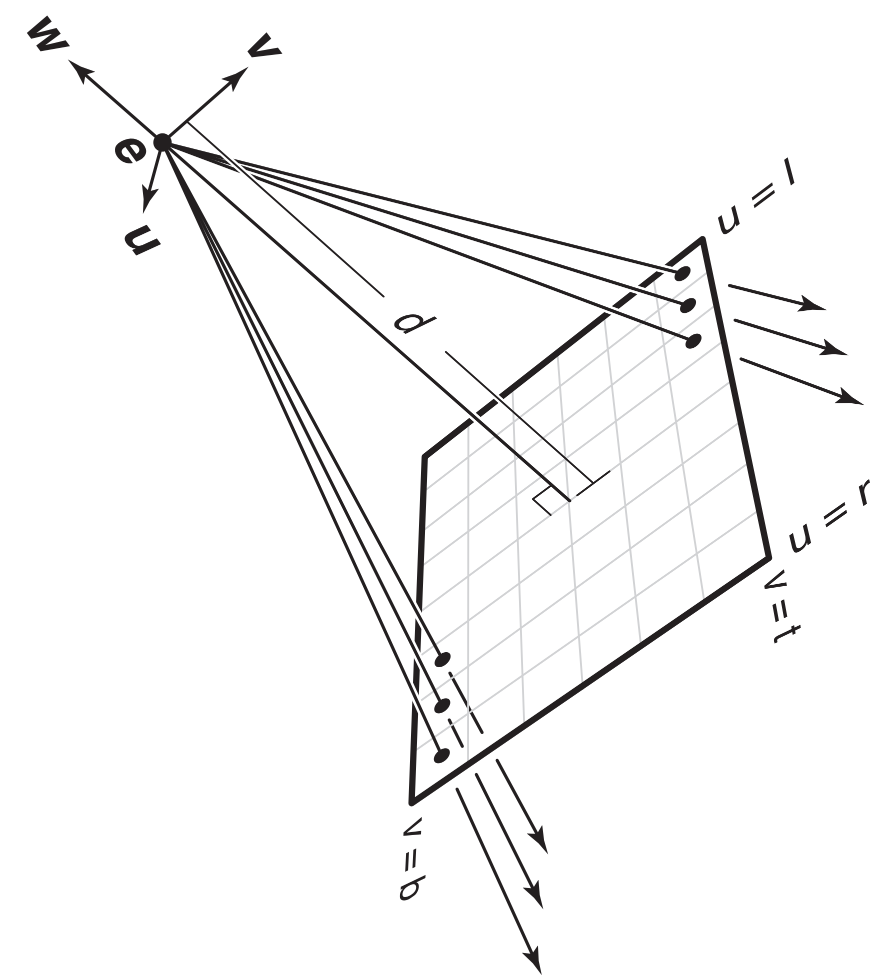
$$\vec{p} = \vec{e}$$

$$\vec{d} = u \cdot \vec{u} + v \cdot \vec{v} + -d \cdot \vec{w}$$

If I want to put the camera somewhere else?

The camera's pose is defined by a **coordinate frame:**

- **u** points right from the **eye**
- **v** points up from the eye
- **w** points back from the eye



Problem 1: Give the viewing ray for pixel (i, j) given **e**, **u**, **v**, **w**, *u*, and *v.*

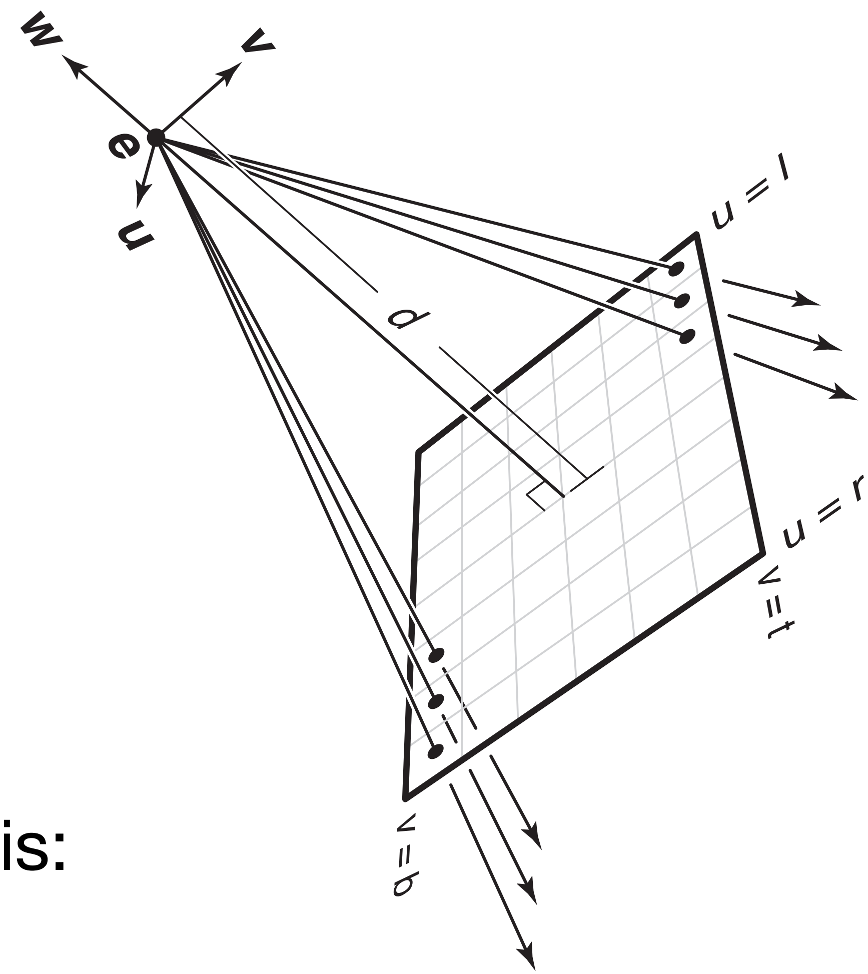
If I want to put the camera somewhere else?

The camera's pose is defined by a **coordinate frame:**

- **u** points right from the **eye**
- **v** points up from the eye
- **w** points back from the eye

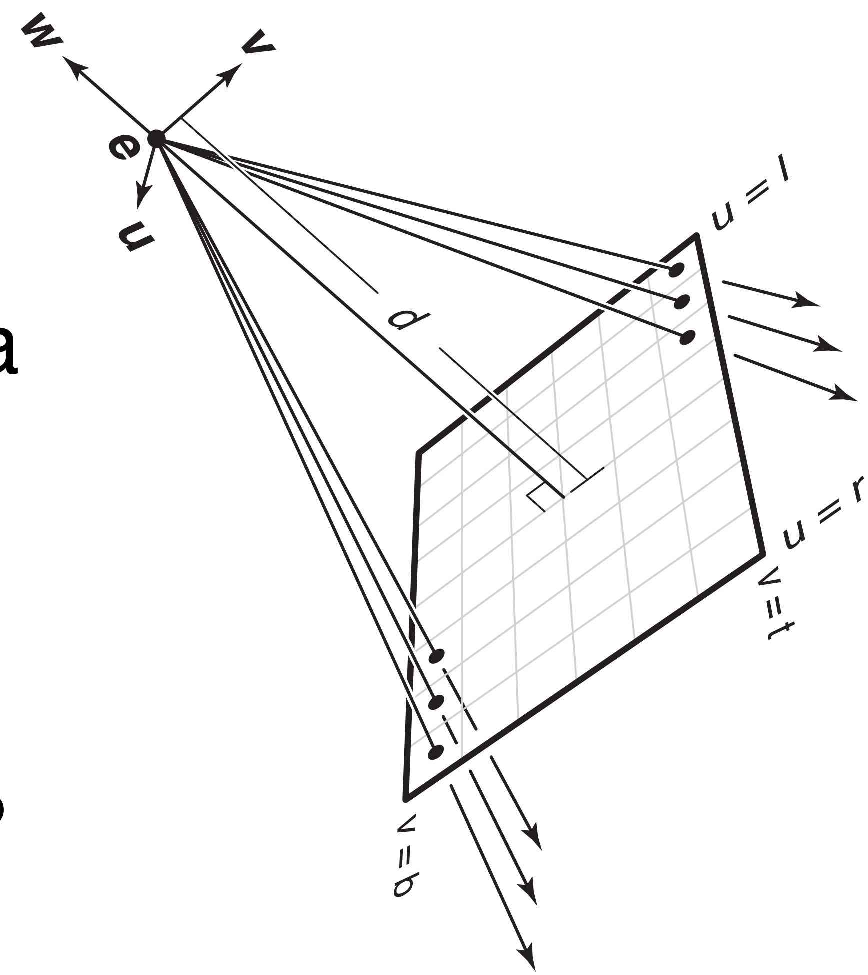
Given this, we can generate a viewing ray as follows:

1. Turn (i,j) into *u, v* as before
2. Viewing ray in (x, y, z) world is:
   origin = **eye**
   direction = $u * \mathbf{u} + v * \mathbf{v} + -d * \mathbf{w}$



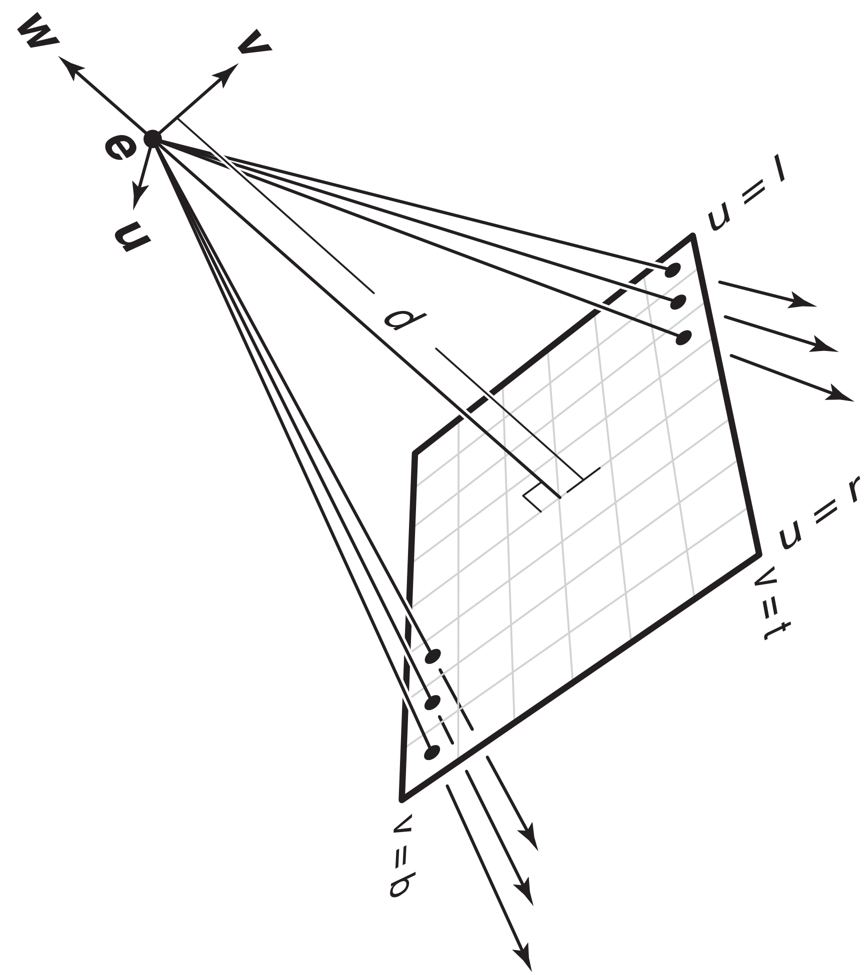
Creating A Camera Basis

- **e, u, v, w** : simple math, but not very intuitive
- Can we position a camera based on:
    - eye
    - view direction or point?



Creating A Camera Basis

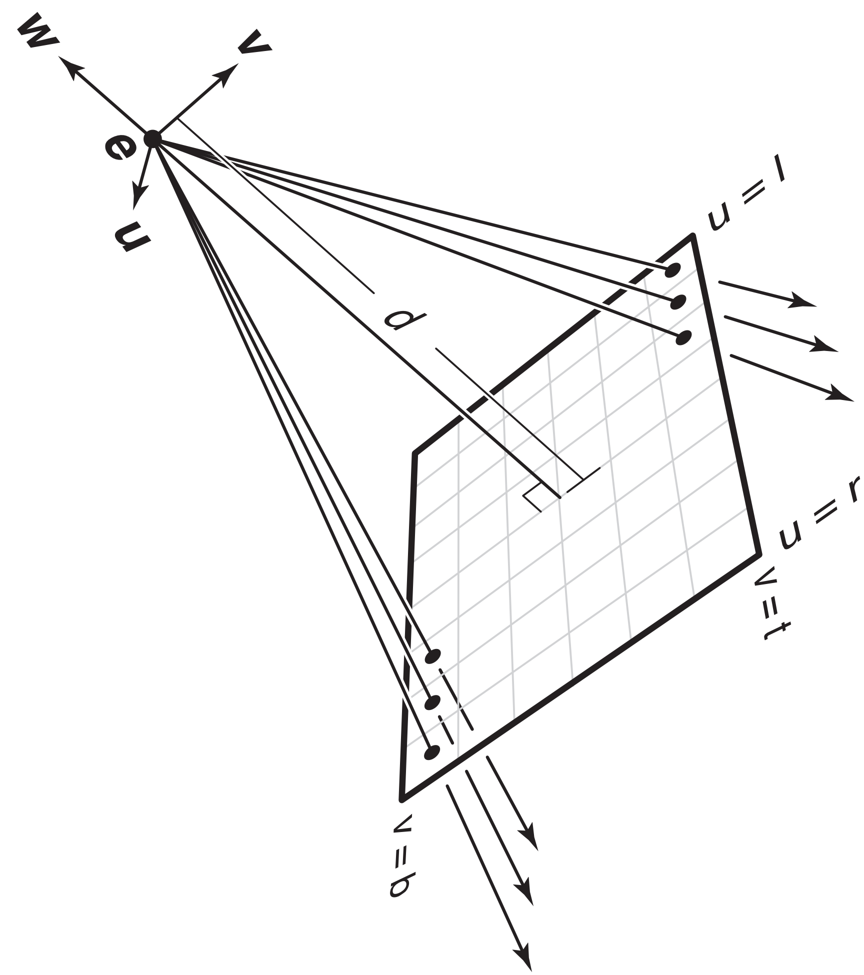
- **eye** - position of eye
- **view** direction - direction camera is looking
- **up** vector - points "up" in the scene, but not necessarily in image space.



Creating A Camera Basis

- **eye** - position of eye
- **view** direction - direction camera is looking
- **up** vector - points "up" in the scene, but not necessarily in image space.

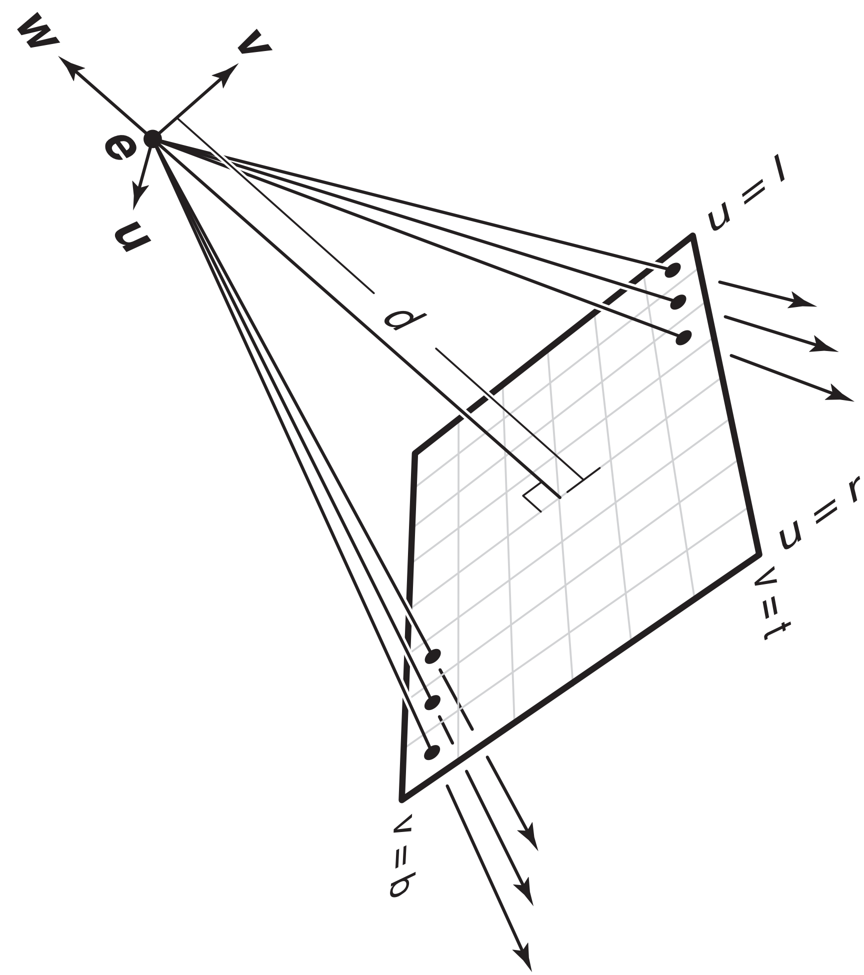
  Why do we need this?
  To eliminate "roll" ambiguity,
  up *should be orthogonal to* **u**



Creating A Camera Basis

- **eye** - position of eye
- **view** direction - direction camera is looking
- **up** vector - points "up" in the scene, but not necessarily in image space.

Why do we need this?
To eliminate "roll" ambiguity,
up *should be orthogonal to* **u**



Problem 2: Compute the coordinate frame (**e**, **u**, **v**, **w**) for a camera given its **eye**, **view**, and **up** vectors

Creating a Camera Basis

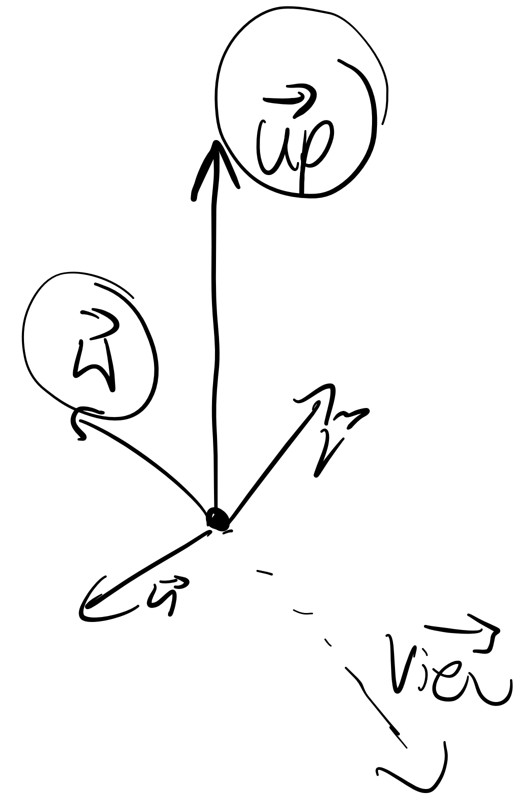
Given **eye**, **view**, and **up**:

1. $\vec{e} = \vec{eye}$

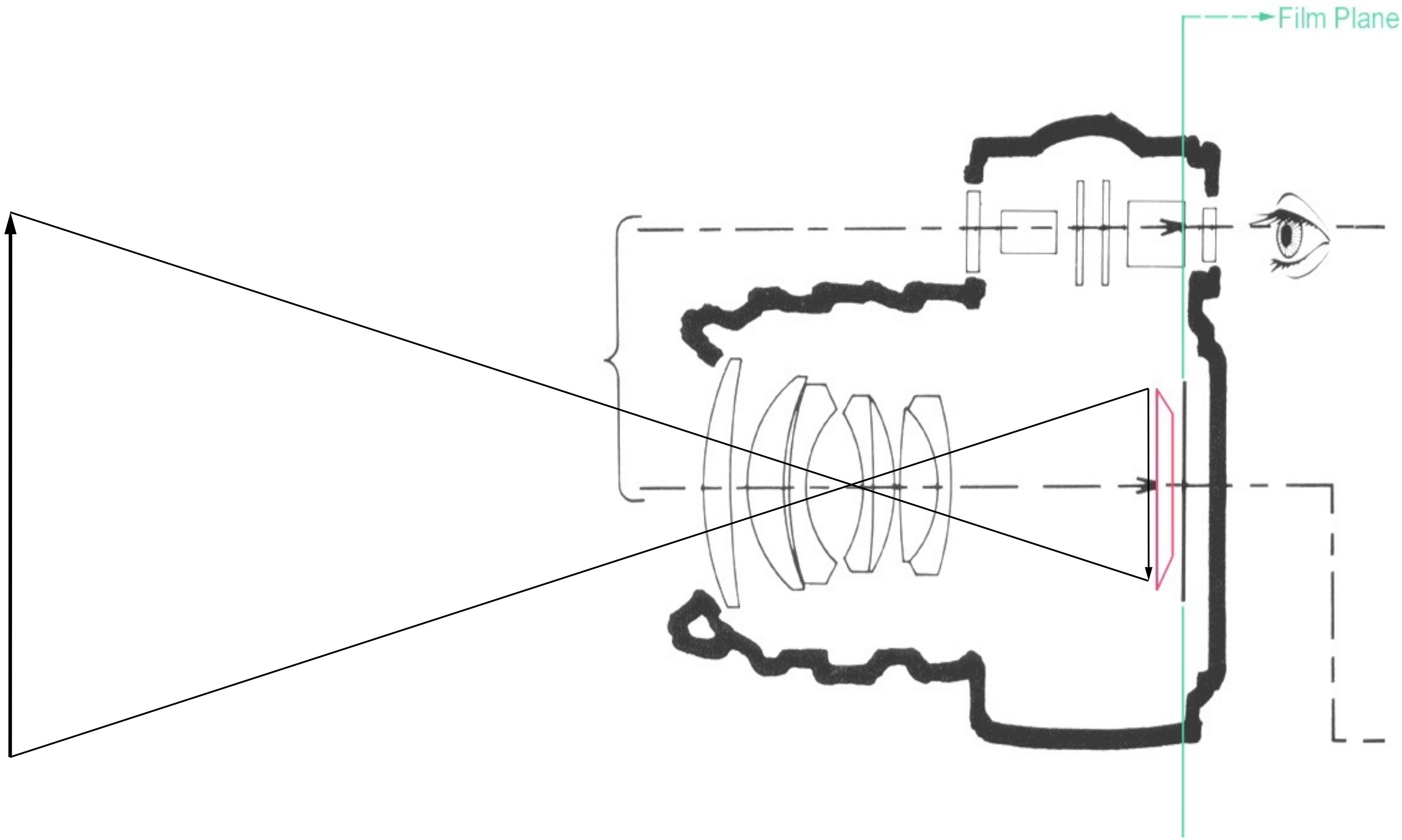
3. $\vec{u} = \vec{up} \times \vec{w}$

4. $\vec{v} = \vec{w} \times \vec{u}$

2. $\vec{w} = -\vec{view}$



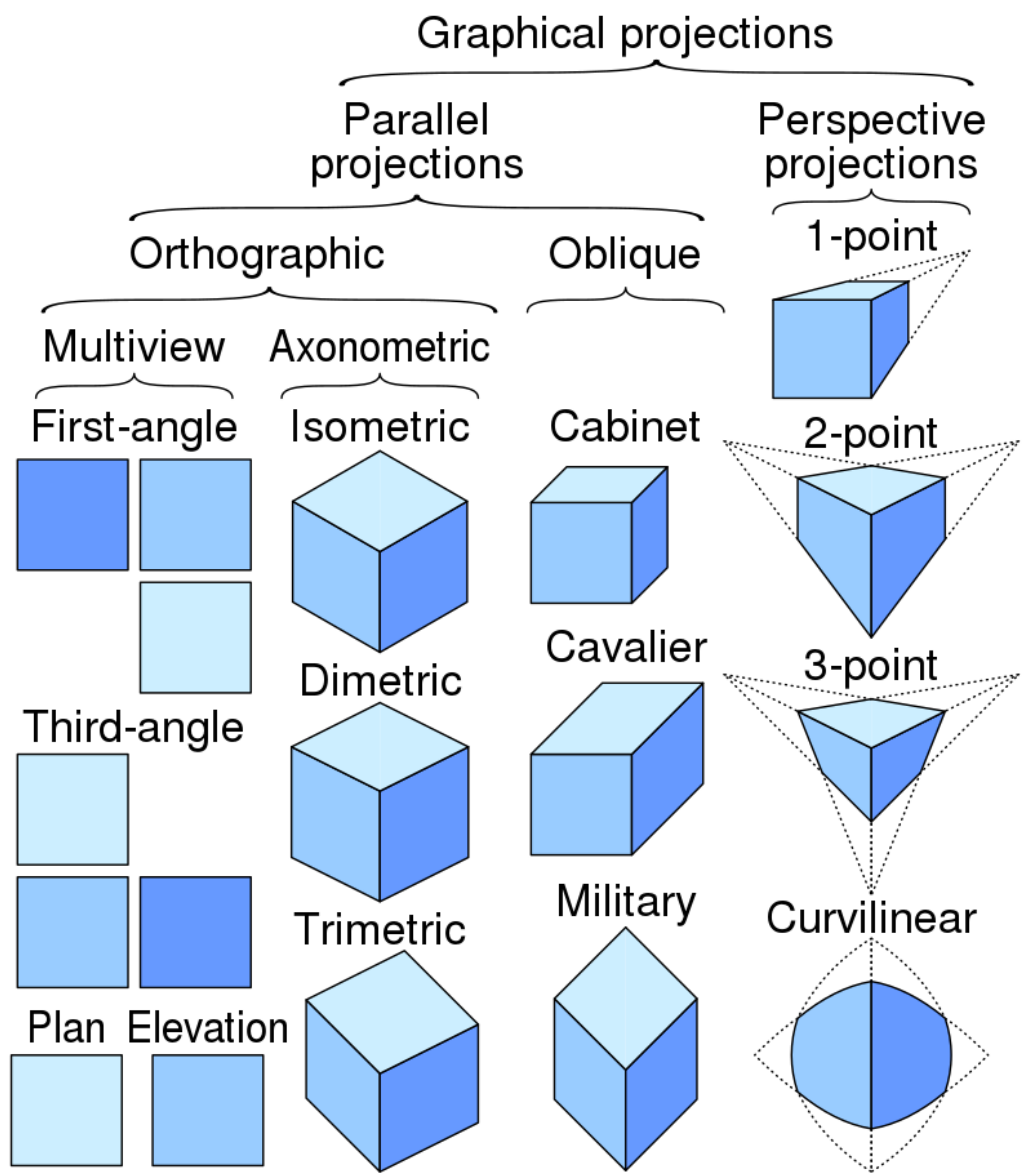
Perspective Cameras: IRL



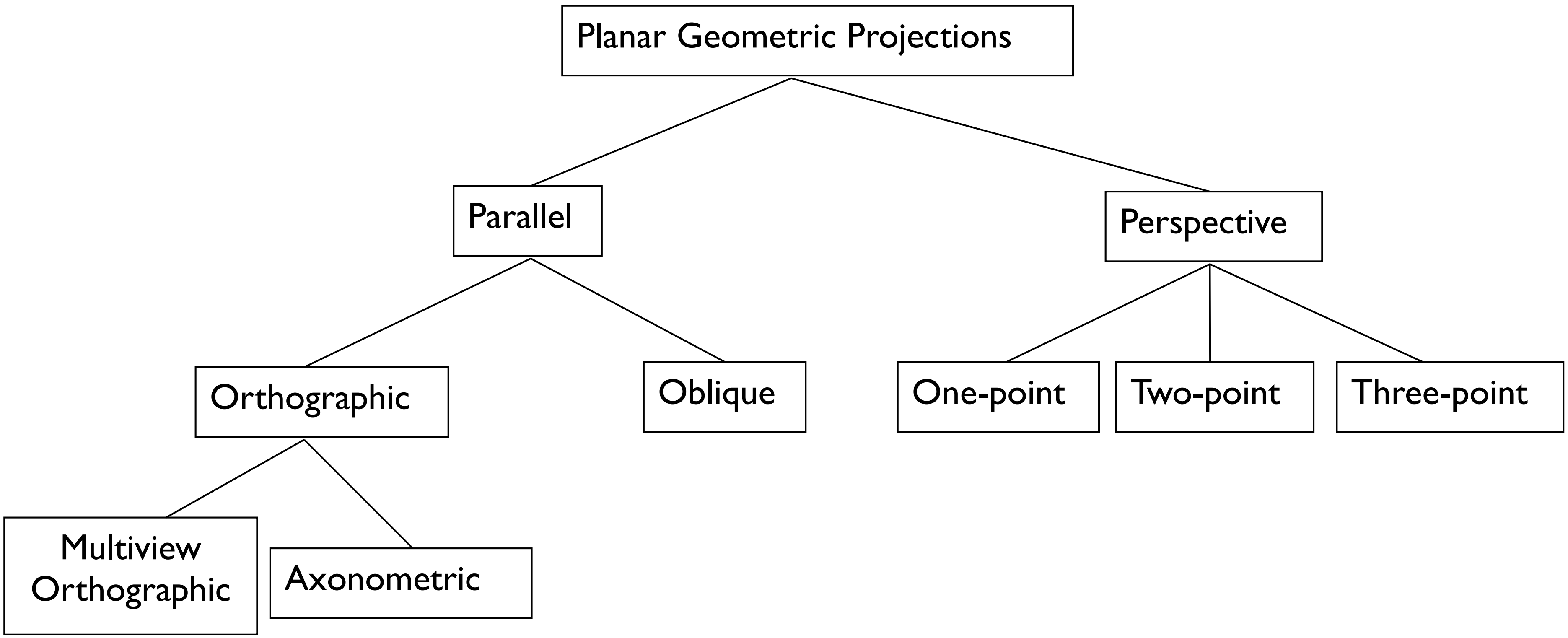
Perspective Cameras: IR(ish)L

- Thin lens model

Classical Projections: Taxonomy



Classical Projections: Taxonomy

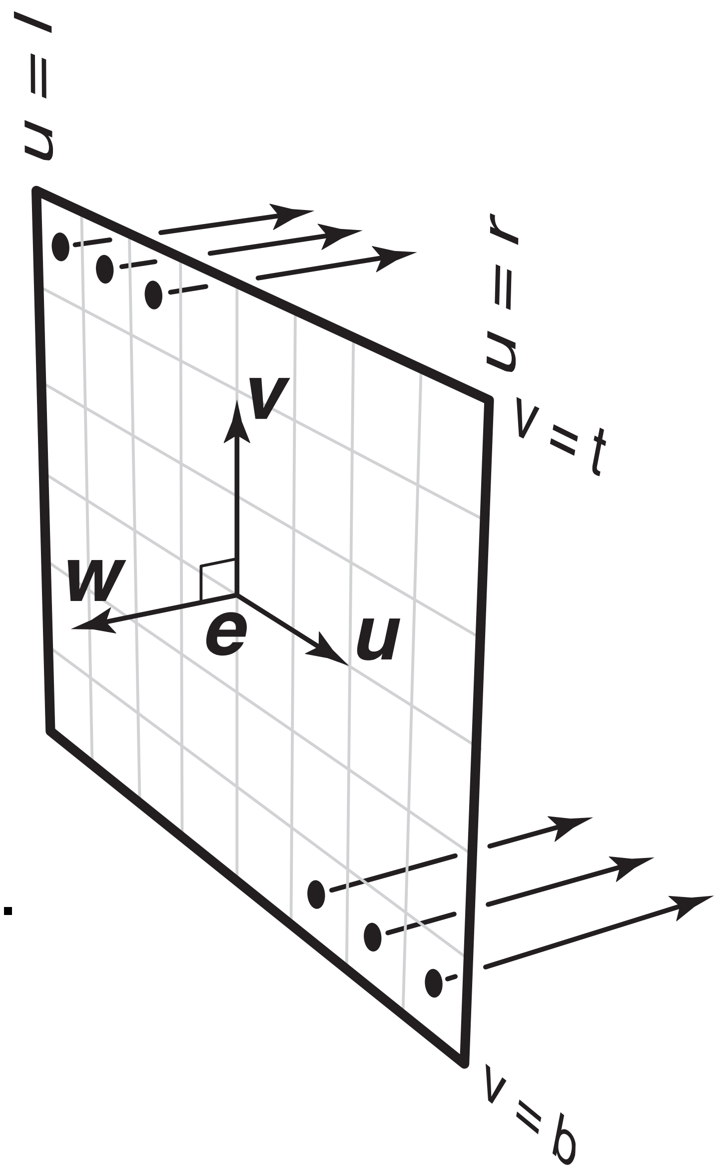


Parallel Projections

- Parallel viewing rays
- Ray origins from pixels
- Camera origin (eye) is on the image plane

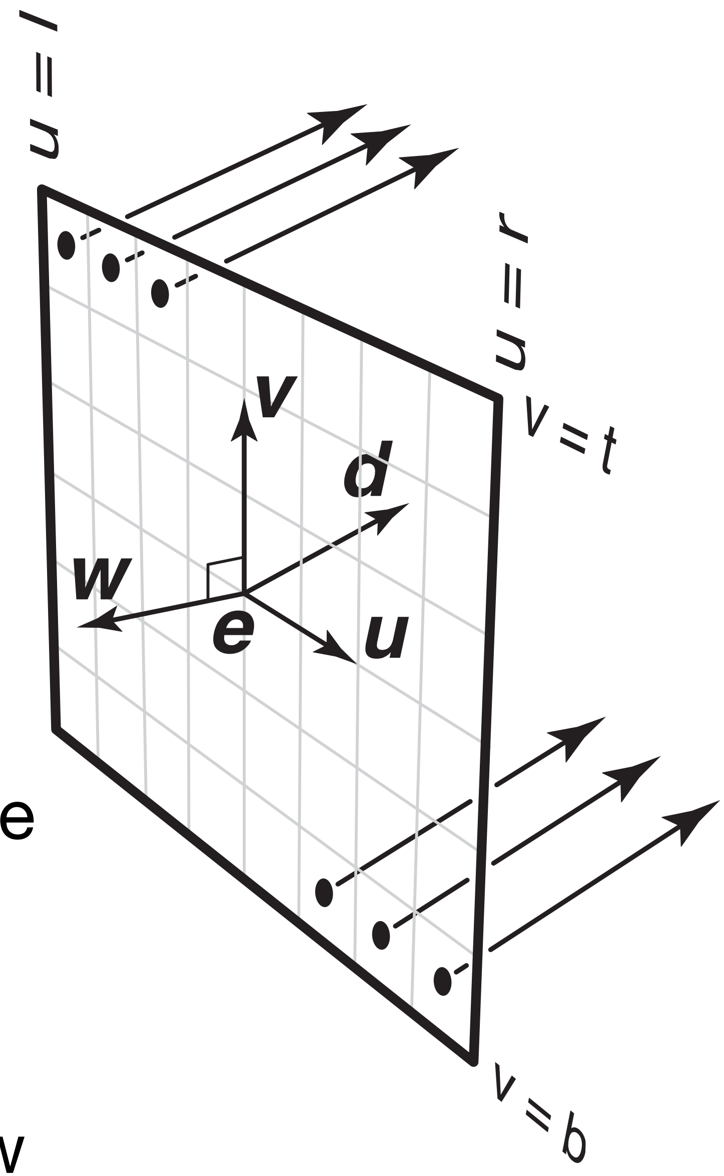
Orthographic: viewing rays are perpendicular to projection plane.

i.e., ray direction **d** = -w



Funky Parallel Projections

- Parallel viewing rays
- Ray origins from pixels
- Camera origin (eye) is on the image plane

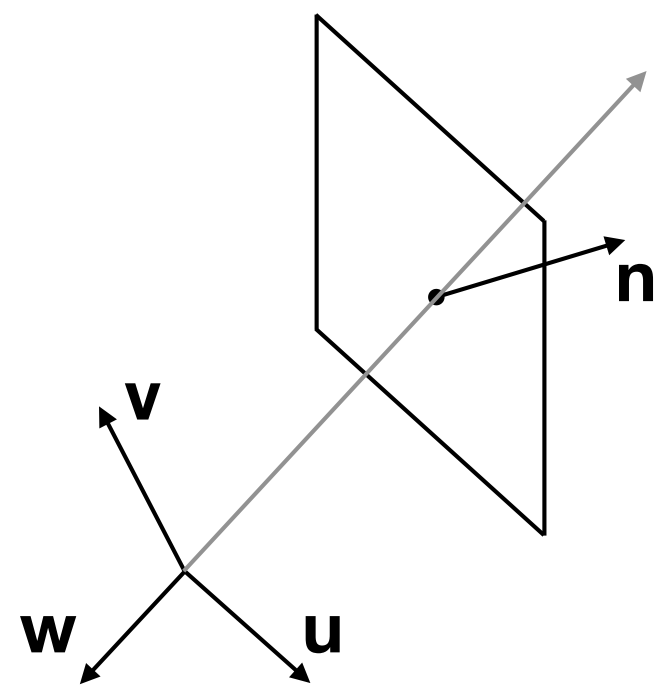


Oblique parallel: viewing rays are *not* perpendicular to projection plane.

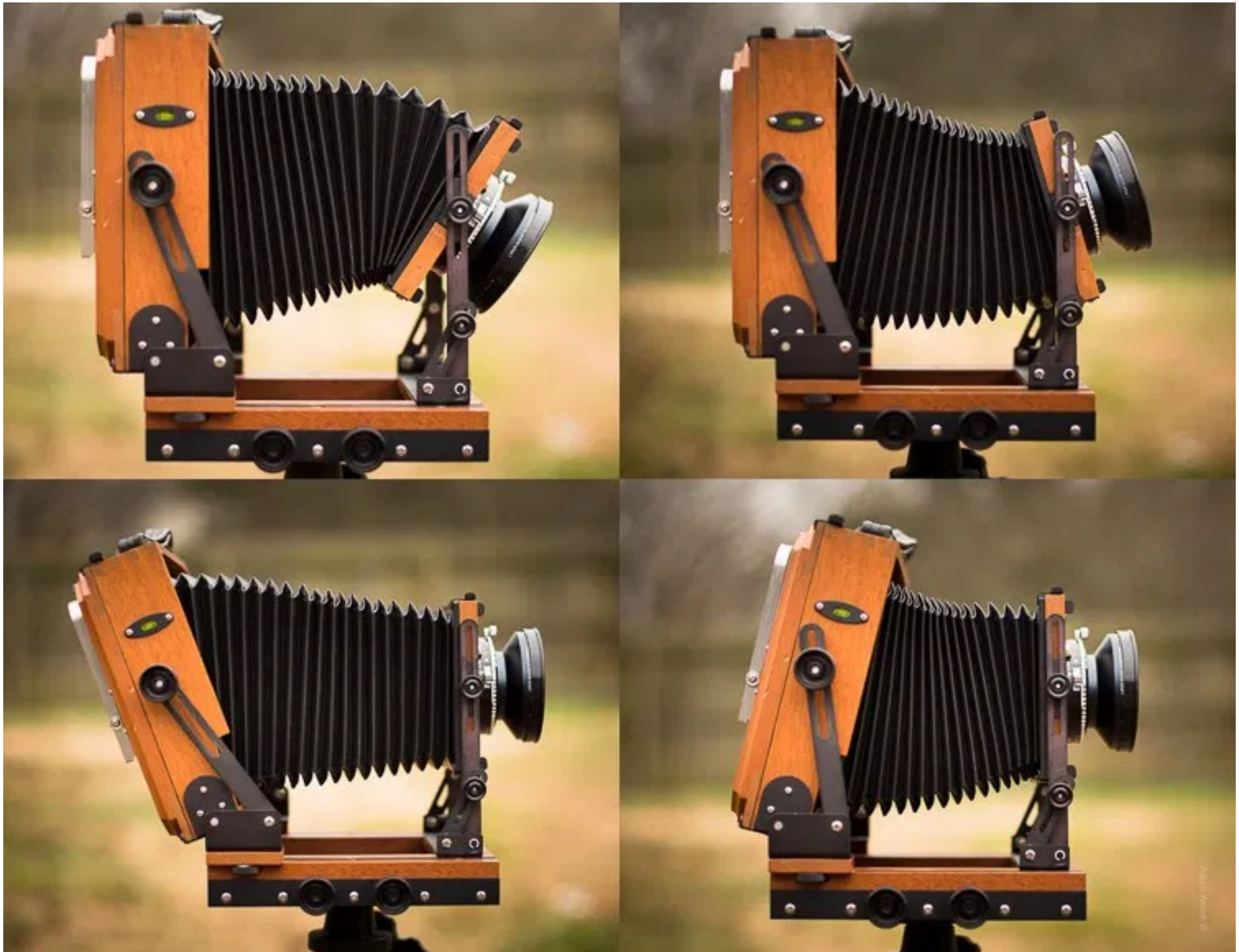
i.e., ray direction **d** differs from -w

Funky Perspective Projections

Shifted perspective: view direction **not** the same as the projection plane normal



Funky Perspective Projections: IRL

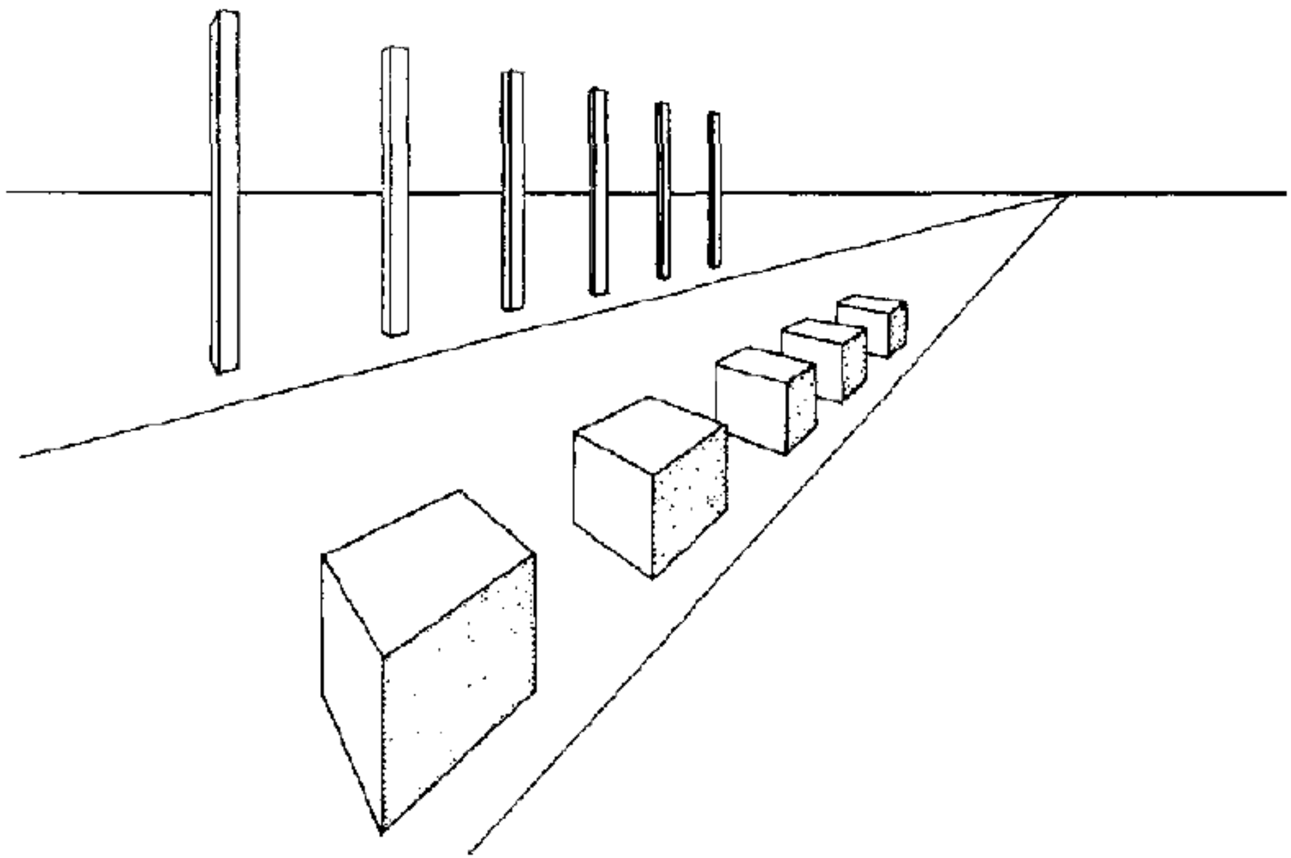


Funky Perspective Projections: IRL

... but why do we want this?

Perspective distortions

- Lengths, length ratios



[Carlbom & Paciorek 78]

"foreshortening": object size is inversely related to depth

[Philip Greenspun]

camera tilted up: converging vertical lines

[Philip Greenspun]

lens shifted up: parallel vertical lines