
[Richard Zakia]

Computer Graphics

Lecture 6
Projections
More General Cameras

Announcements

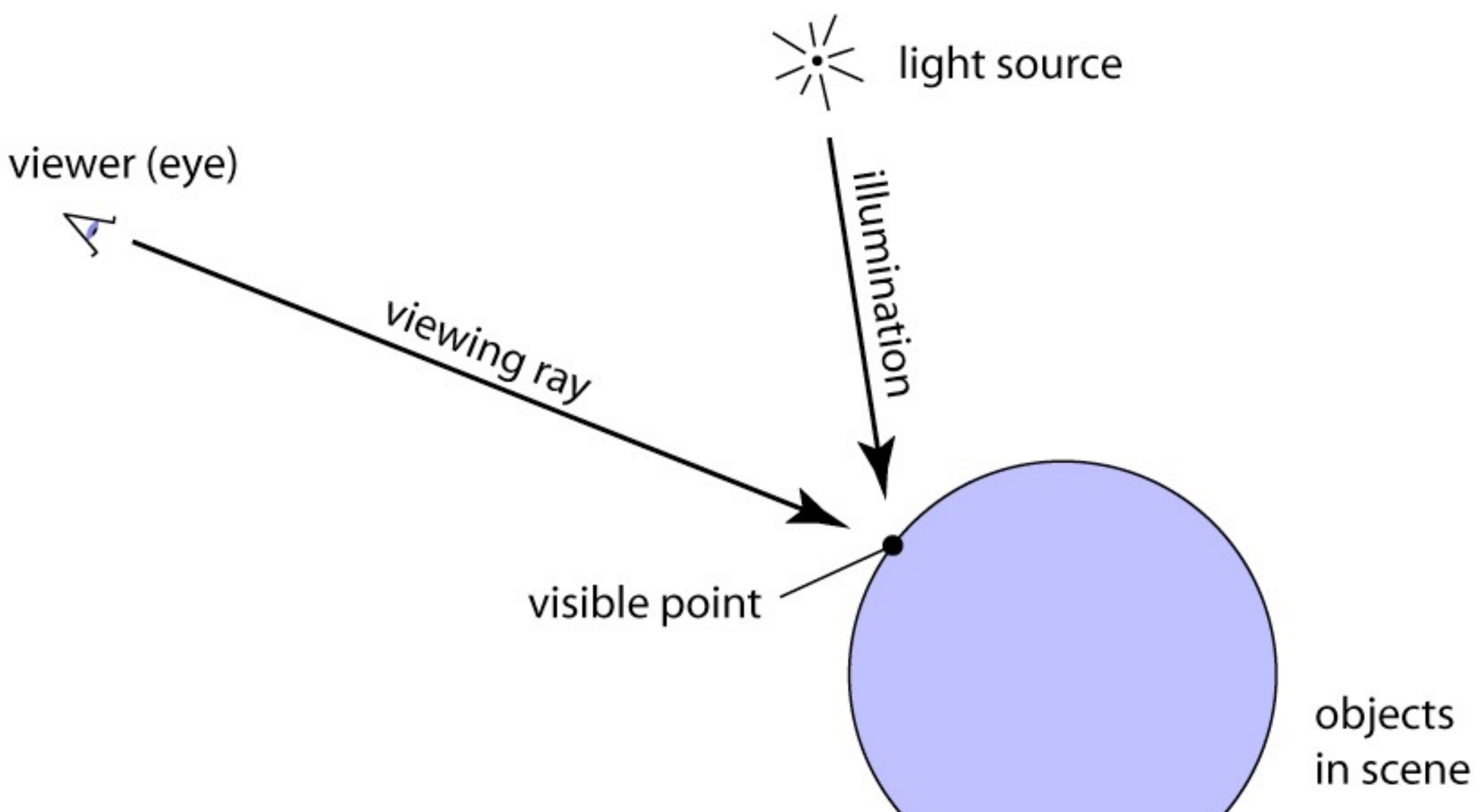
- CS Department TGIF - today at 4pm! Details: https://cs.wwu.edu/tgif-department-social
- HW1 is out - due Monday 1/25
    - #3 will help with A1
- A1 - if you don't have a partner yet, meet me in the In-Class Voice channel on Discord directly after class.

Announcements

- No class Monday - MLK Day
- I'll hold office hours Tuesday 10-11am to replace Monday's.
- Today's office hours will be cut short: 2:00-2:30.
- Tuesday's lecture - live, no videos
    - You may want to review Chapter 2.2 on quadratic equations.

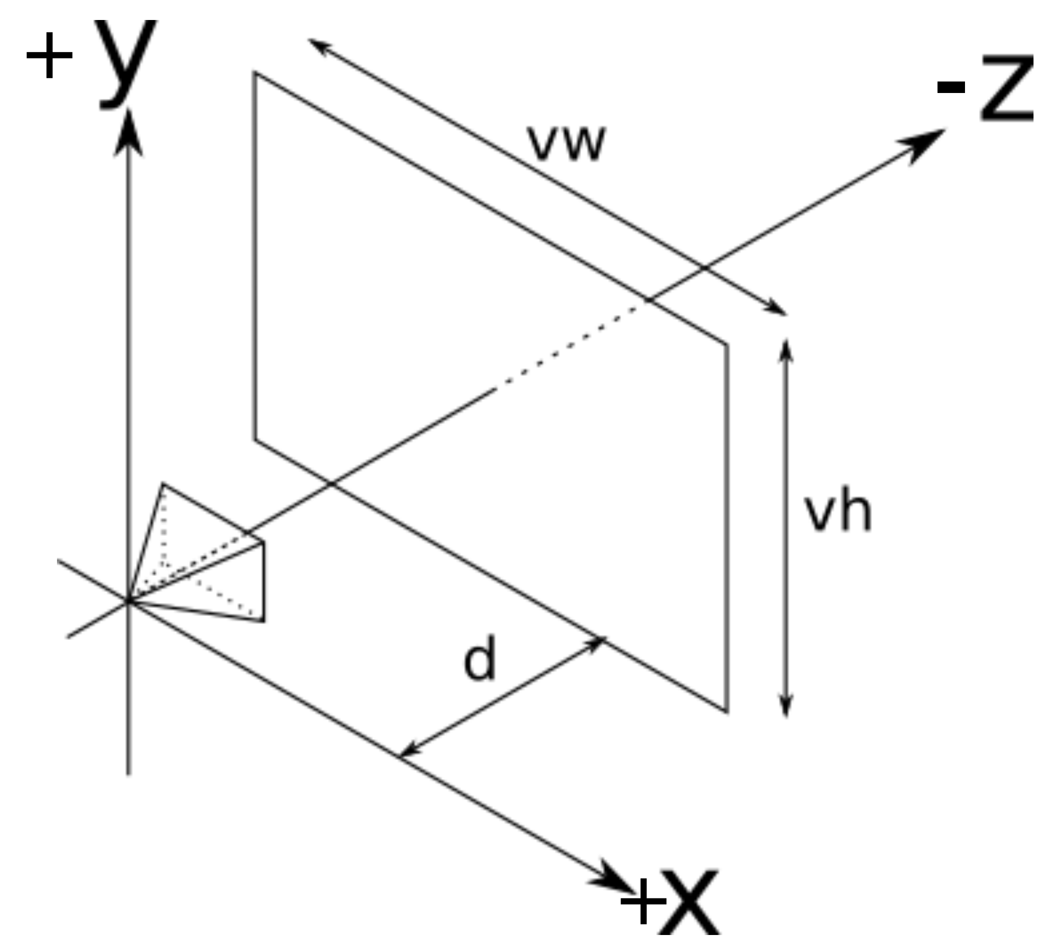
Ray Tracing: Pseudocode

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



A "canonical" perspective 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



More General Cameras

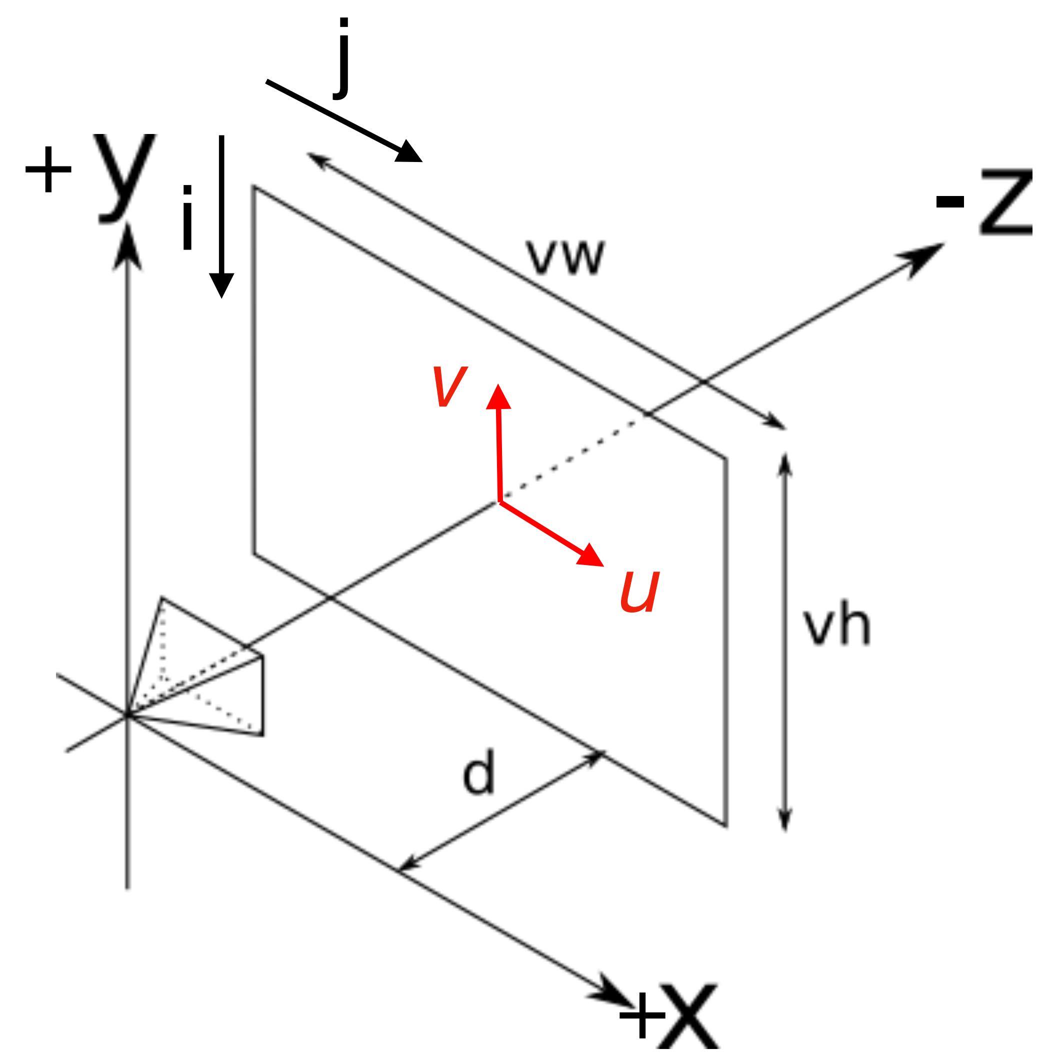
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

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

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



More General Cameras

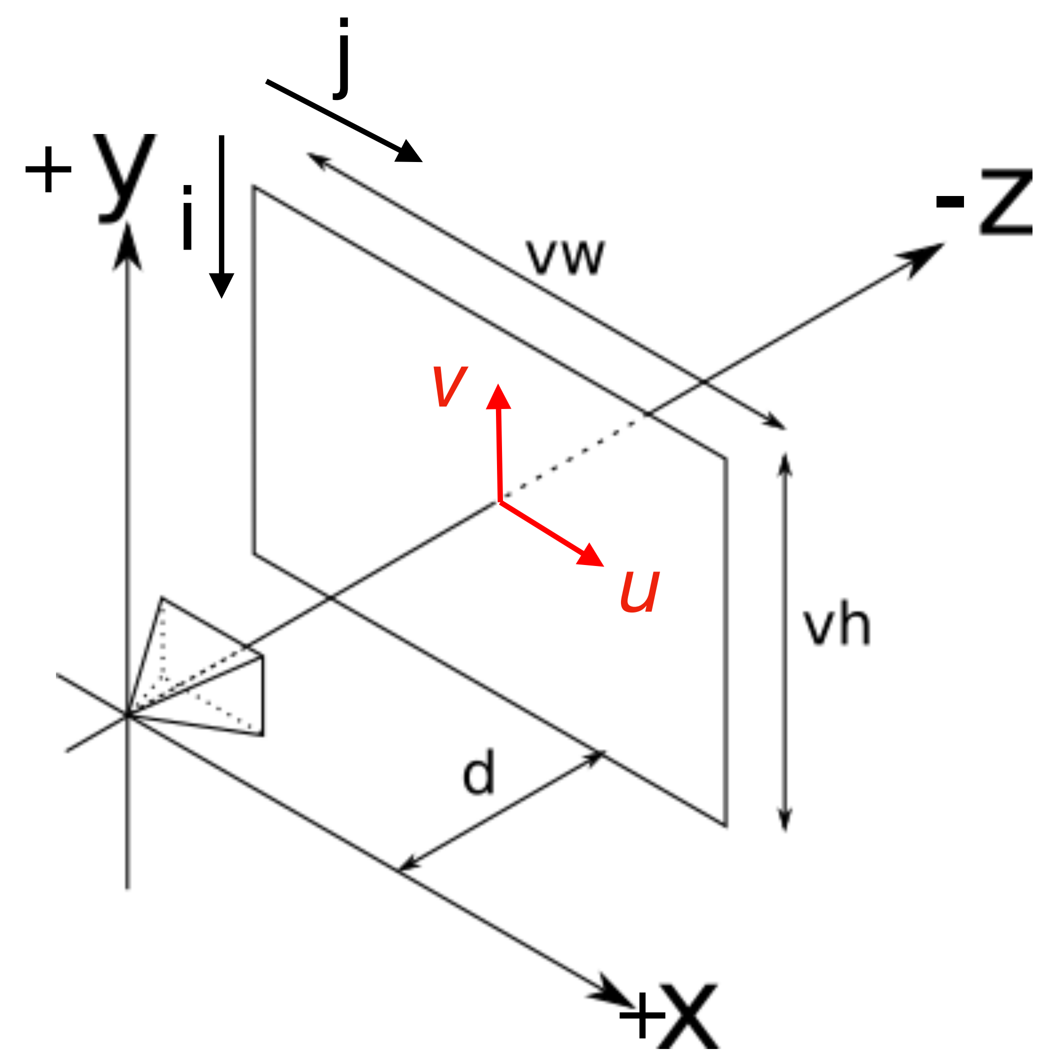
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

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

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

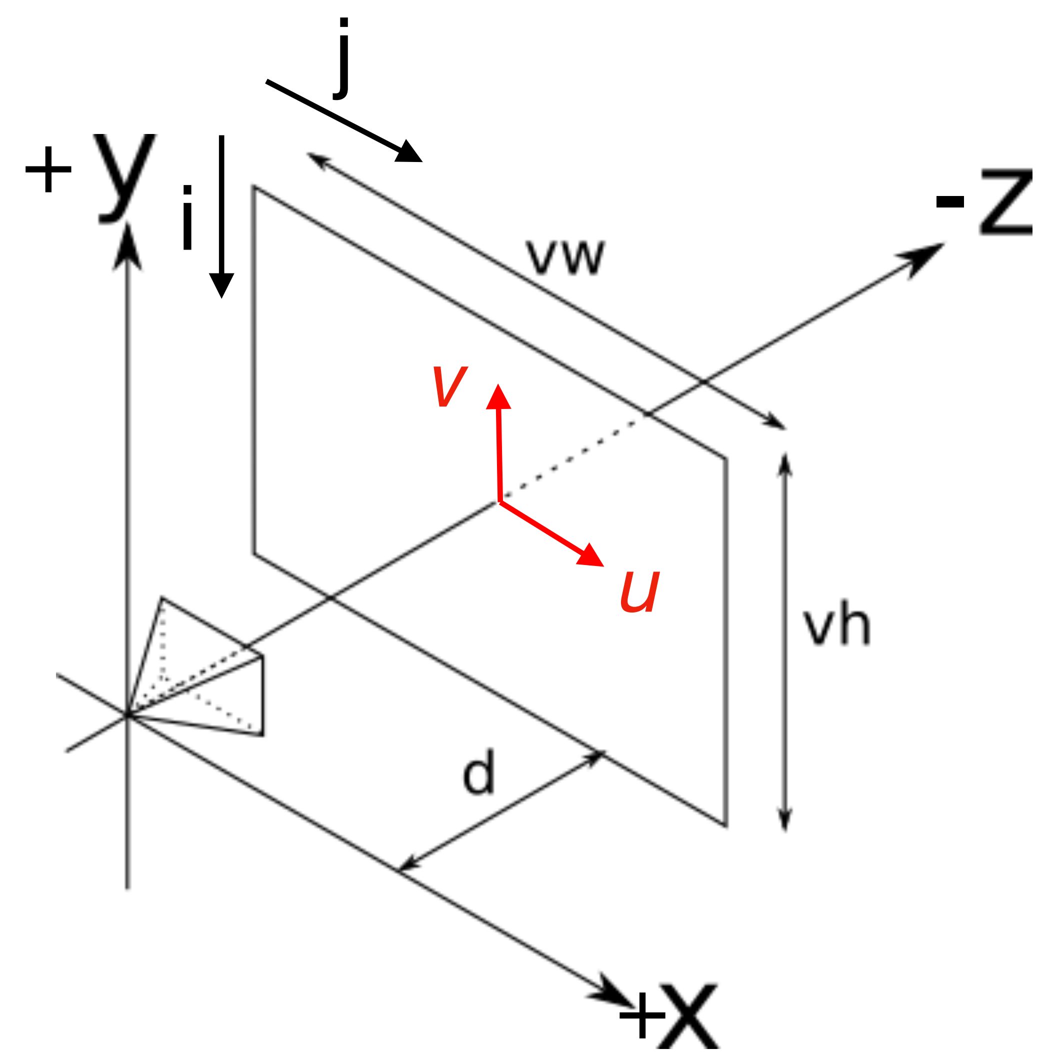
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

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

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



More General Cameras

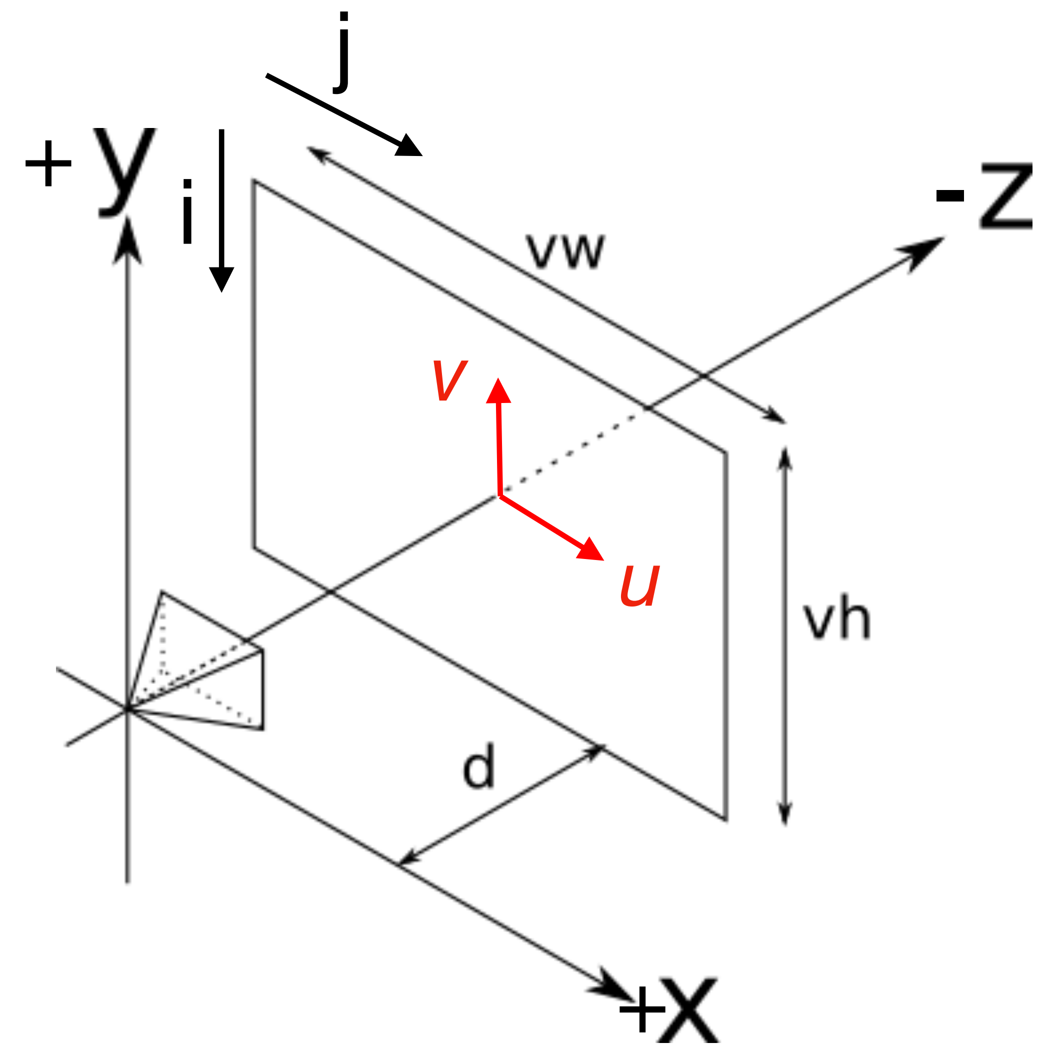
$$u = \left(\frac{j - \frac{1}{2}}{W} - \frac{1}{2}\right) * vw$$

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

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

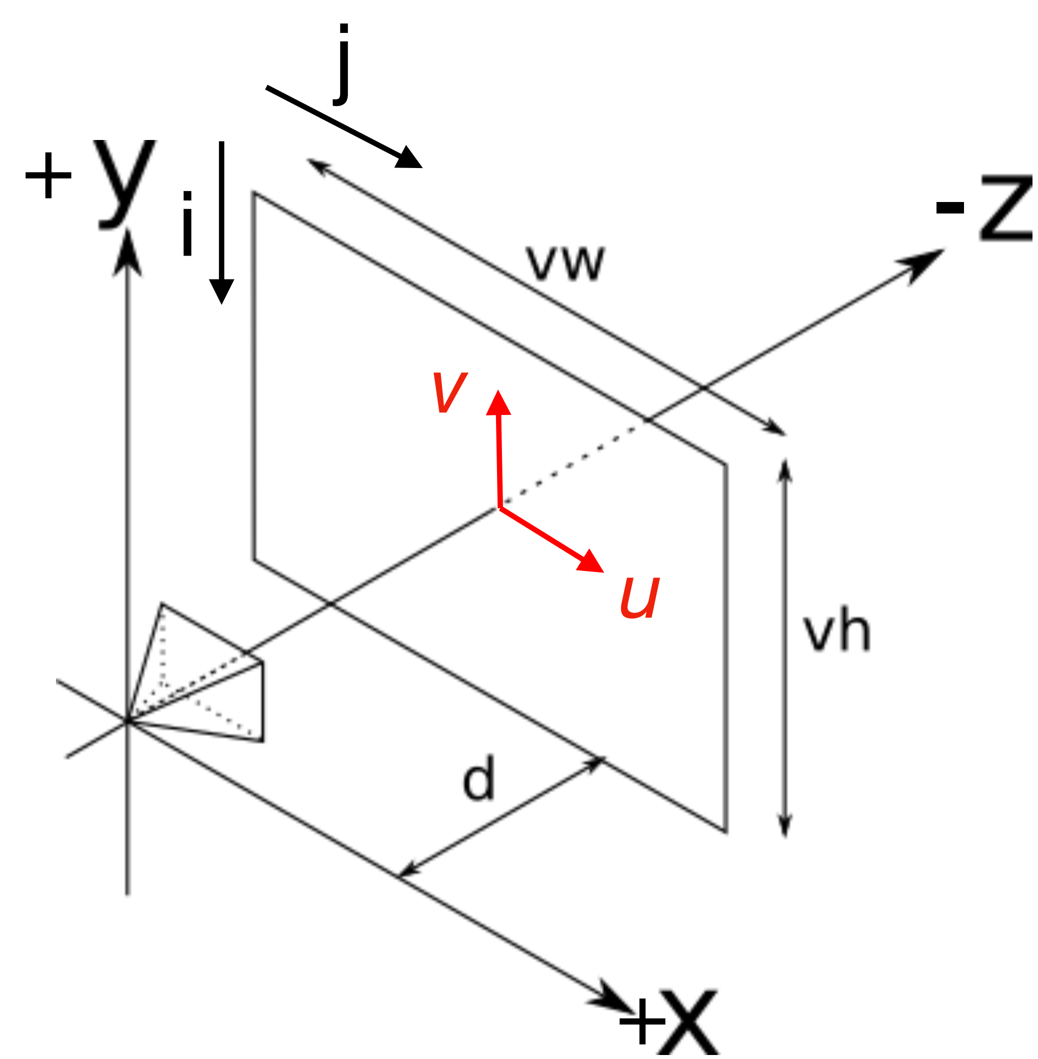


More General Cameras

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

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

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

More General Cameras

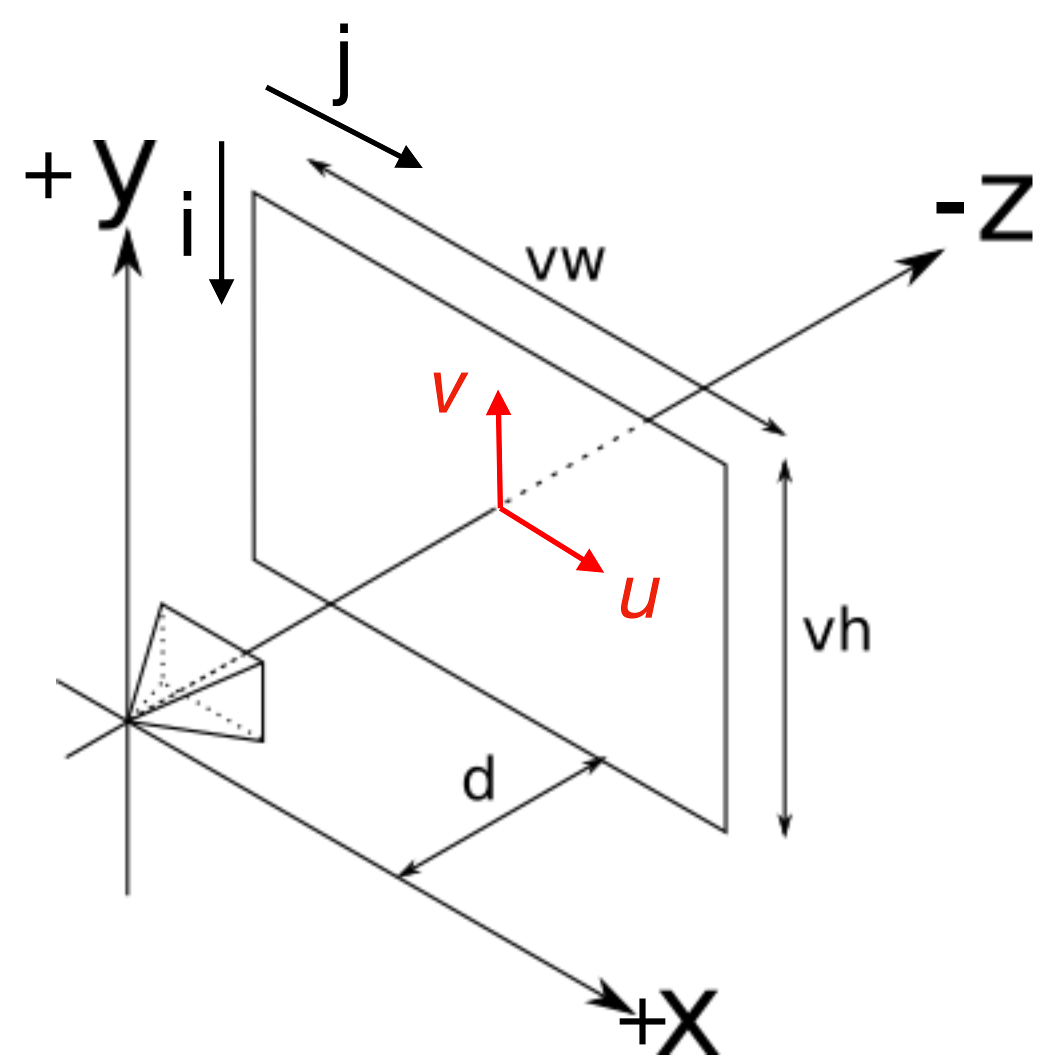
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

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

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

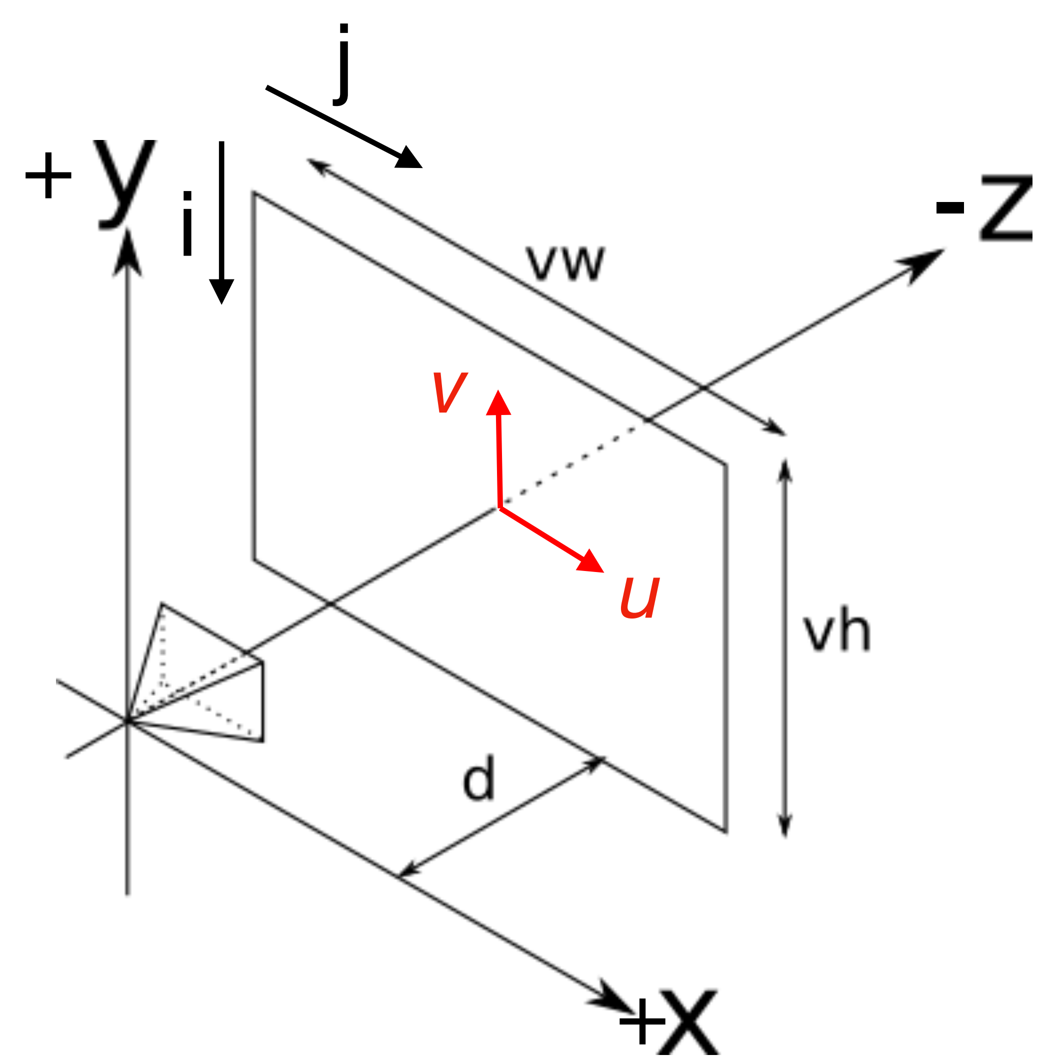
$$u = \frac{j - \frac{1}{2}}{W} - \frac{1}{2}$$

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

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

Let's break some assumptions!

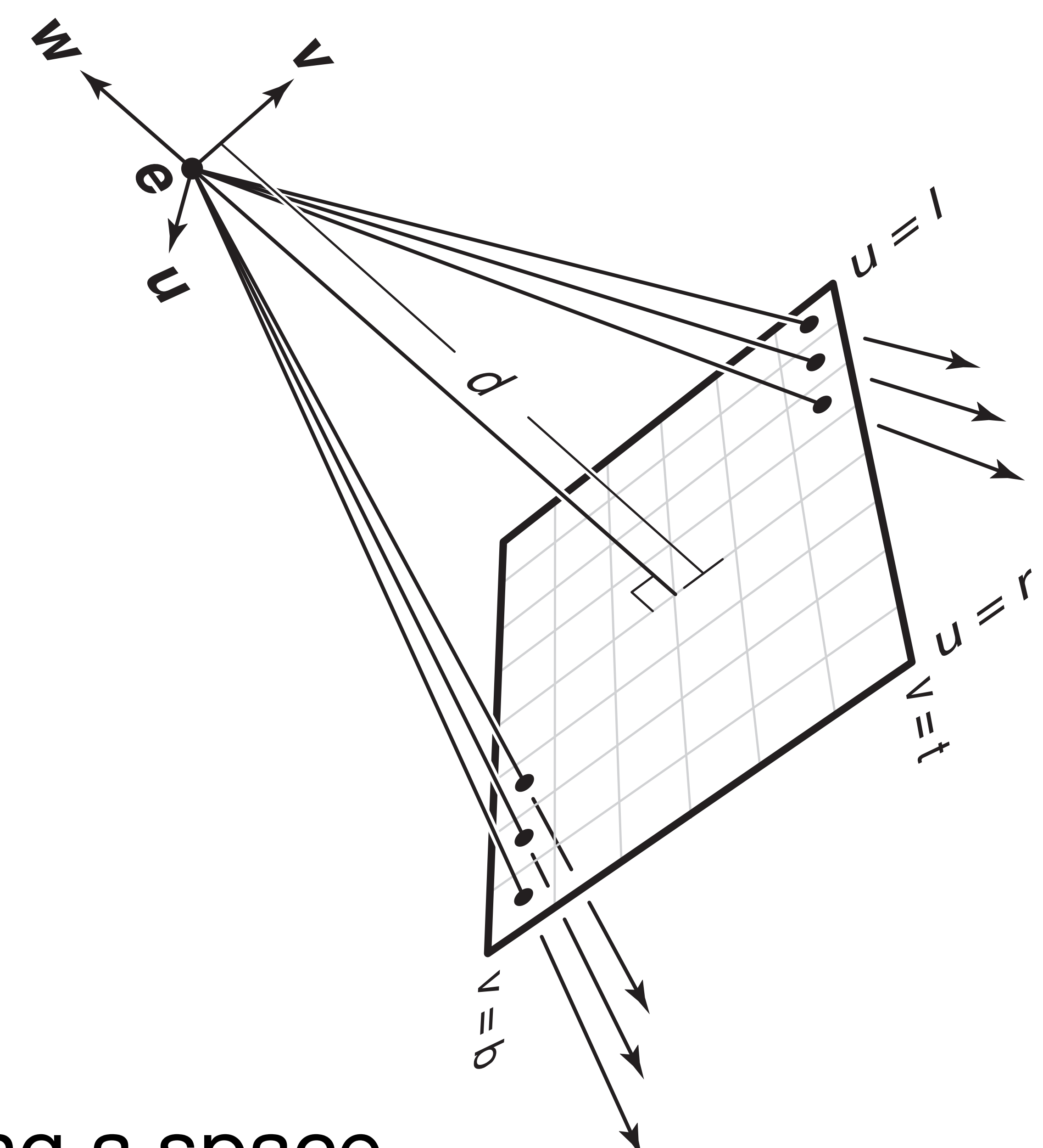
- d = 1
- vh = vw = 1
- Eye is at the origin (0, 0, 0)
- **Looking down the negative z axis**



What if I want to point the camera somewhere else?

The key idea:

1. Find a **basis** where the camera *is* in canonical pose.
2. Change basis back to familiar x-y-z coordinates.



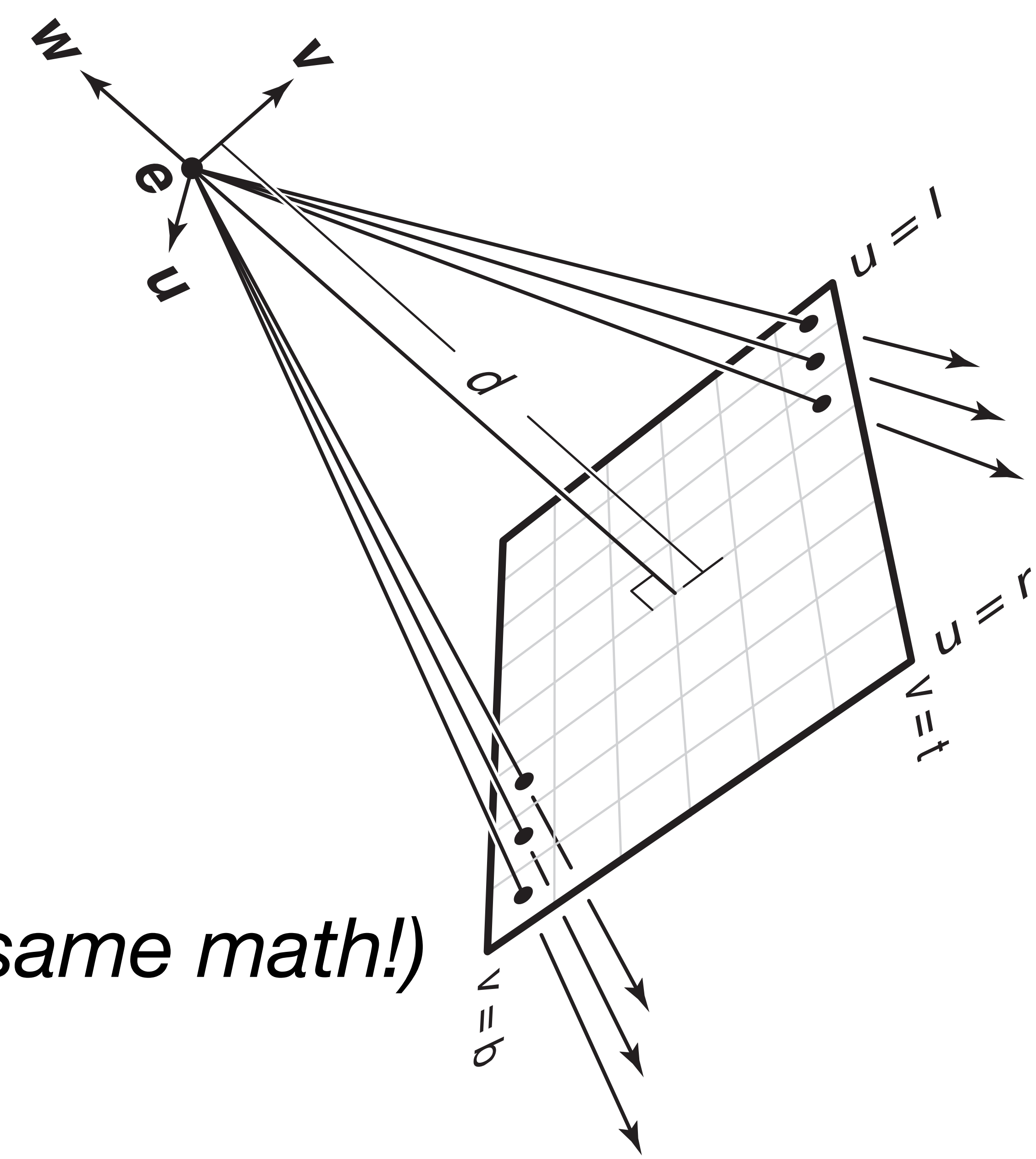
Pedantic side note:

- a **basis** is a set of vectors spanning a space.
- a **frame** is a basis plus an **origin** point.

What if I want to point the camera somewhere else?

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

- **e** is the position of the eye
- **u** points right from **e**
- **v** points up from **e**
- **w** points back from **e**

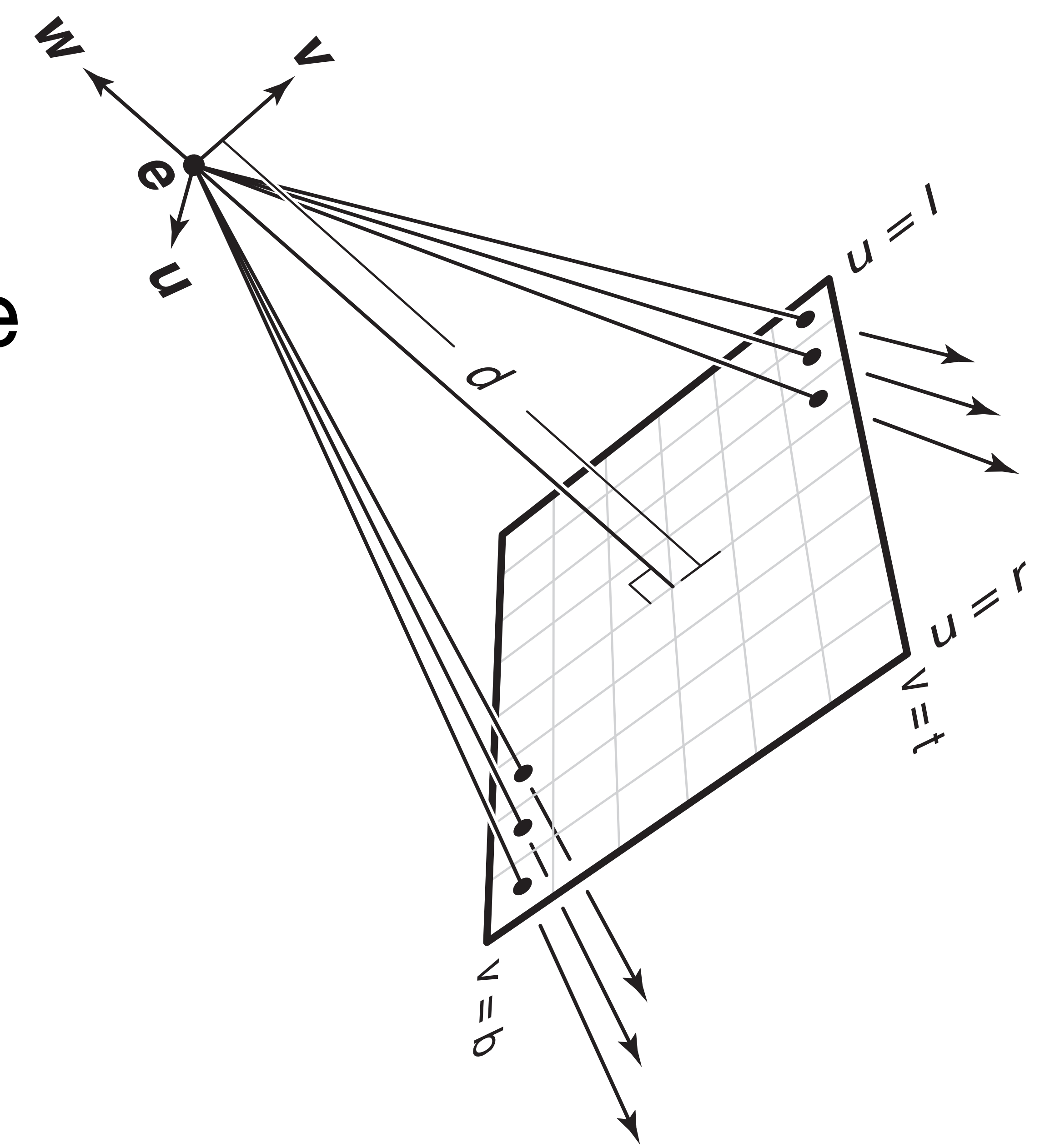


Given this, we can generate a viewing ray as follows:

1. Turn (i,j) into *u, v* instead of x, y *(same math!)*
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

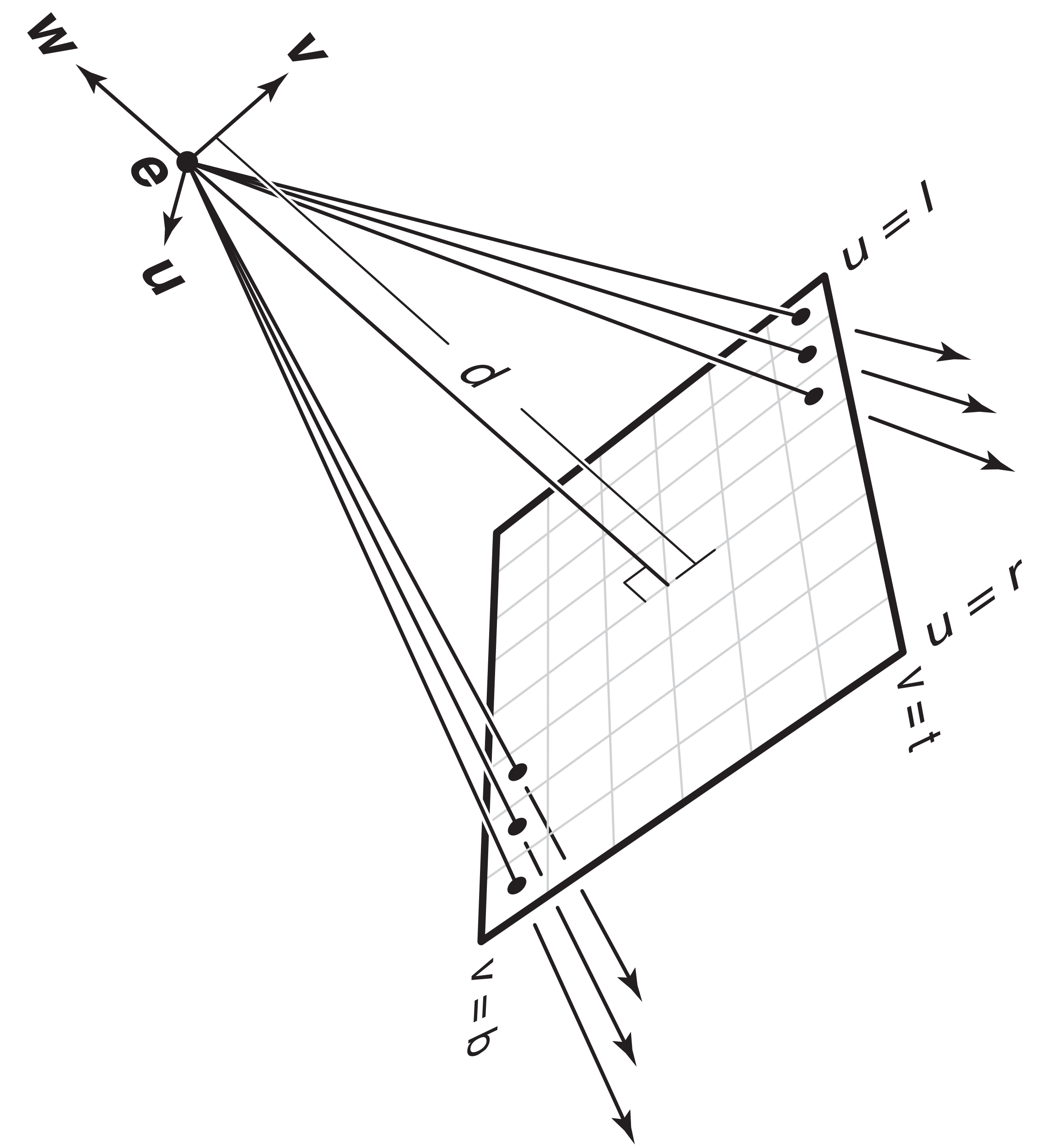
- Ask the modeler to specify **e, u, v, w** : makes the math simple, but not very intuitive modeling.



- Better: position a camera based on:
    - eye
    - view direction or point?

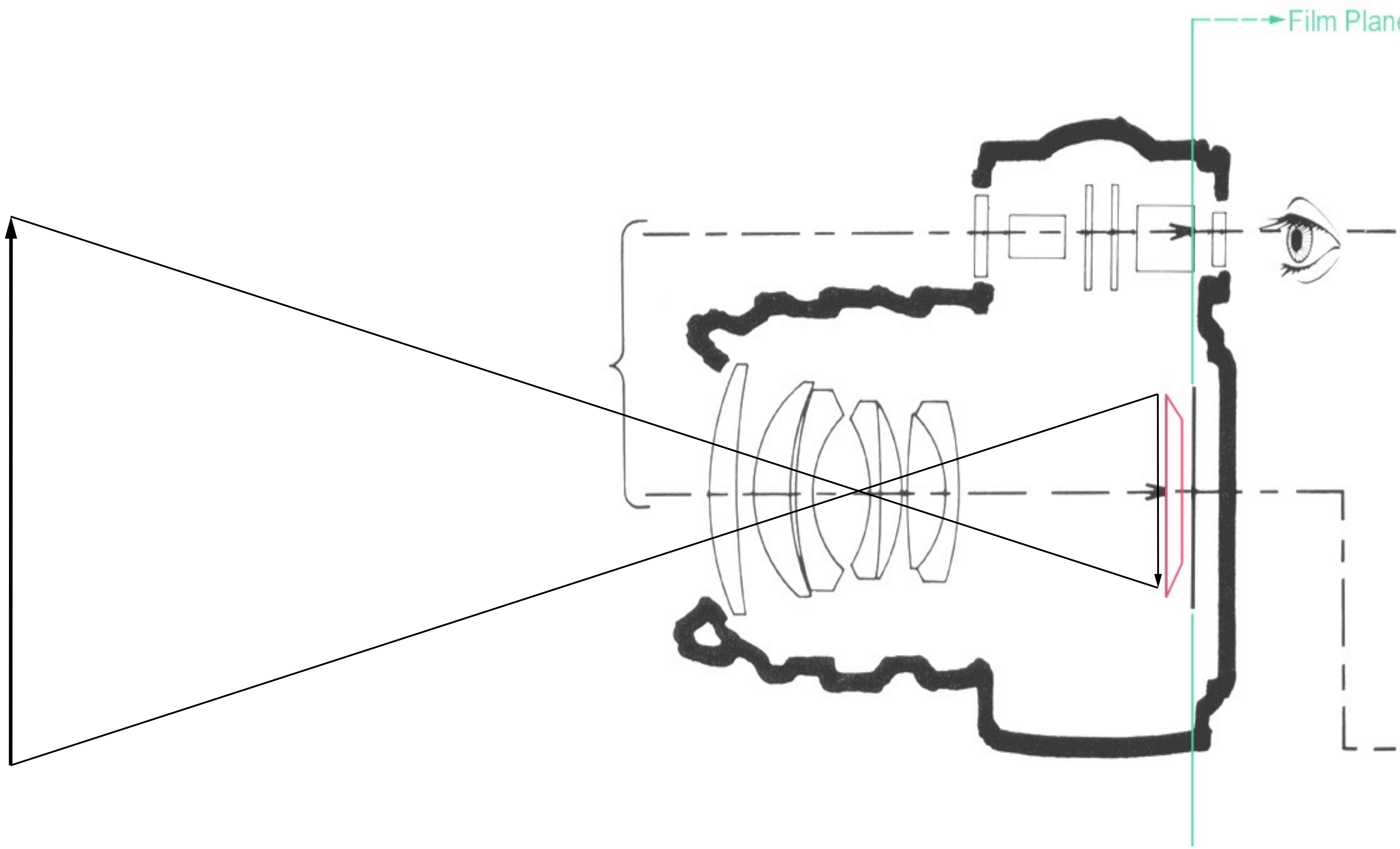
Creating A Camera Basis

- **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 Basis

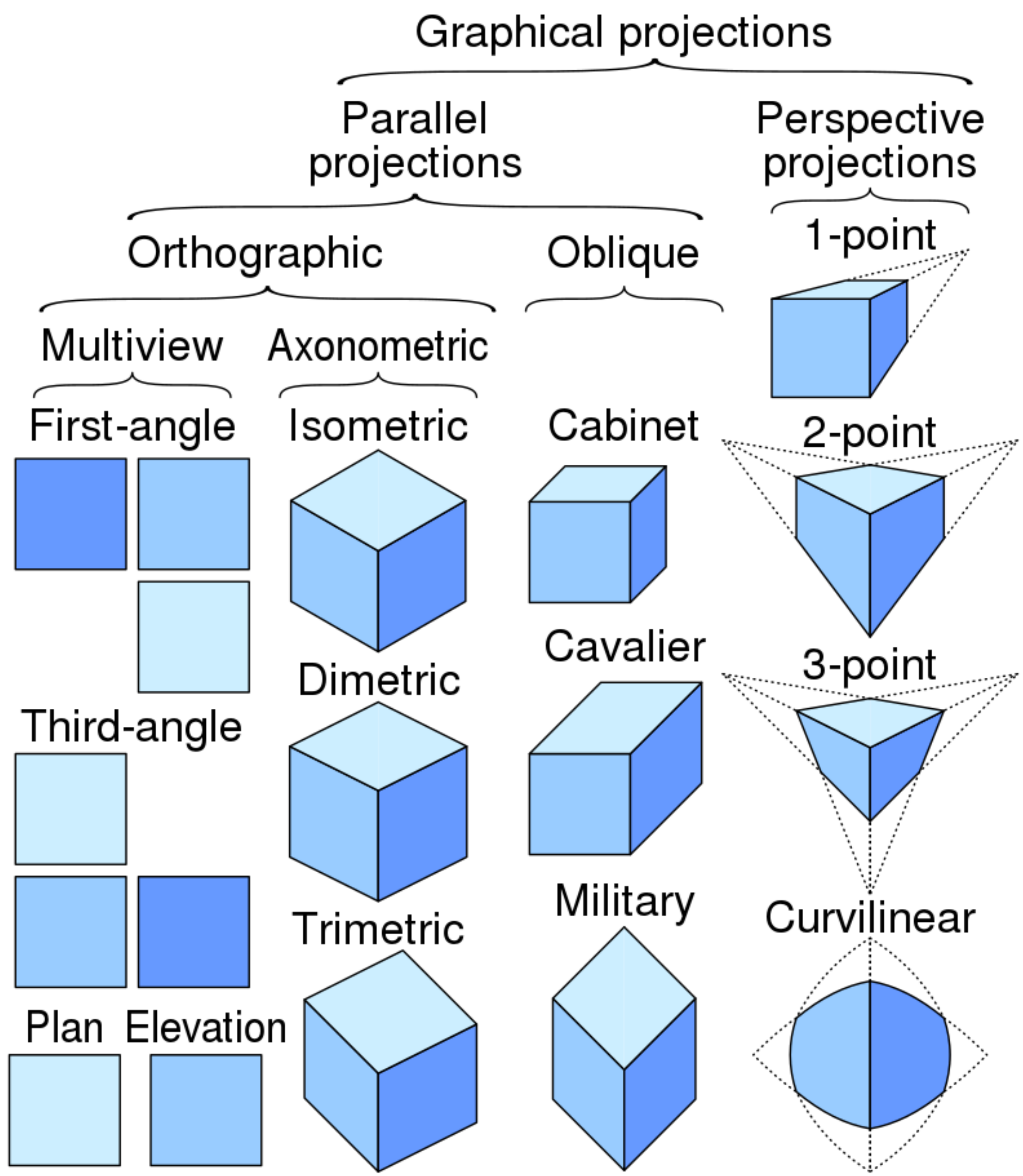
Perspective Cameras: IRL



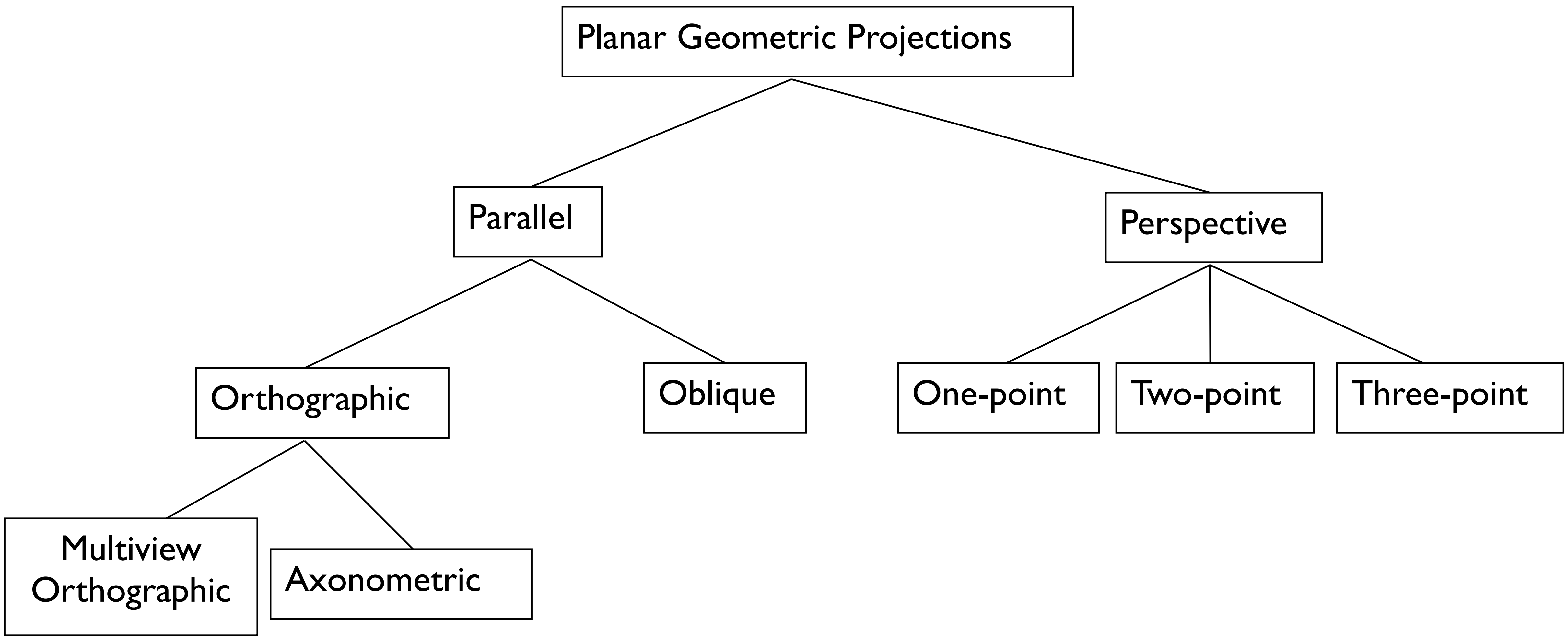
Perspective Cameras: IR(ish)L

- Thin lens model

Classical Projections: Taxonomy

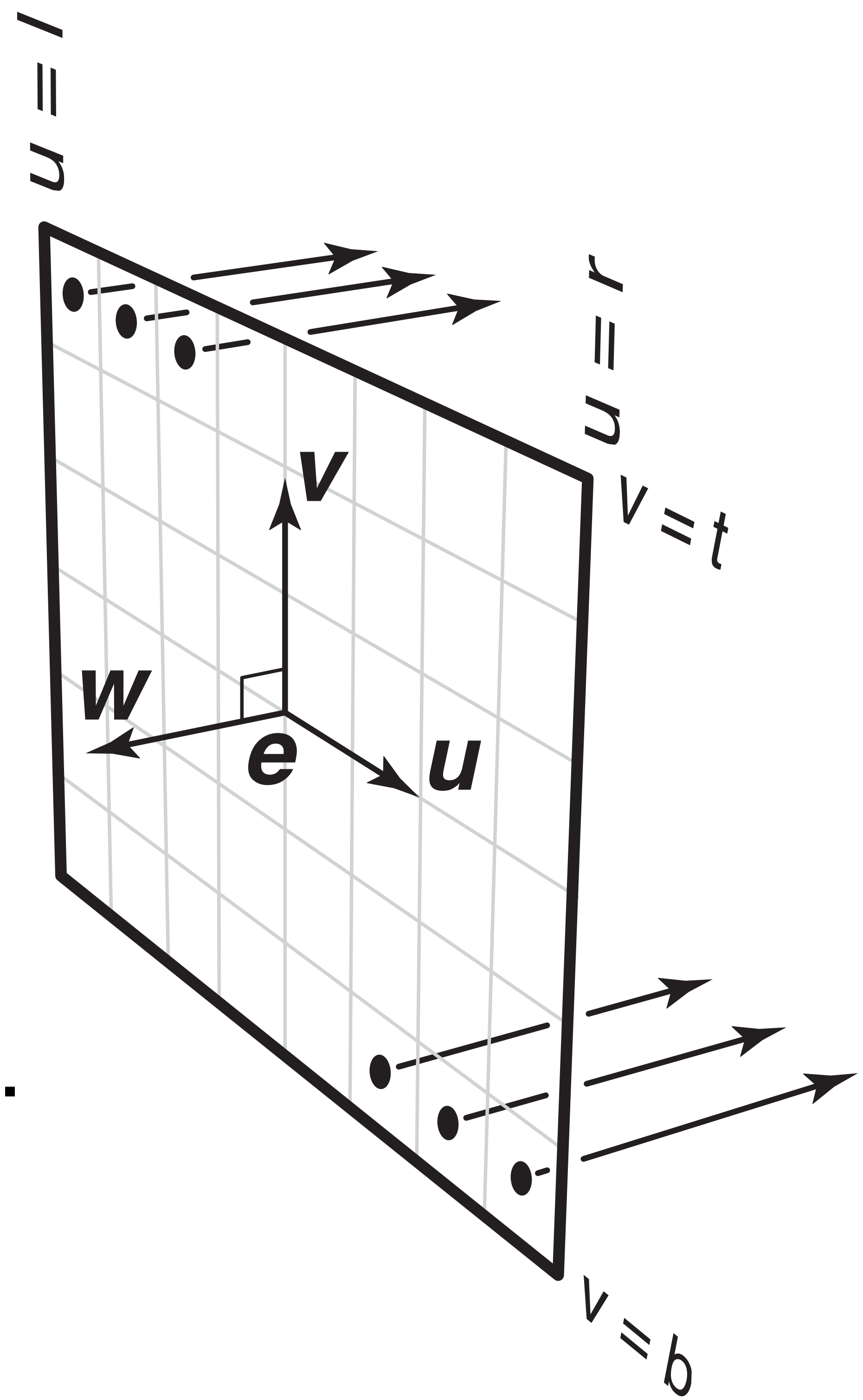


Classical Projections: Taxonomy



Parallel Projections

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

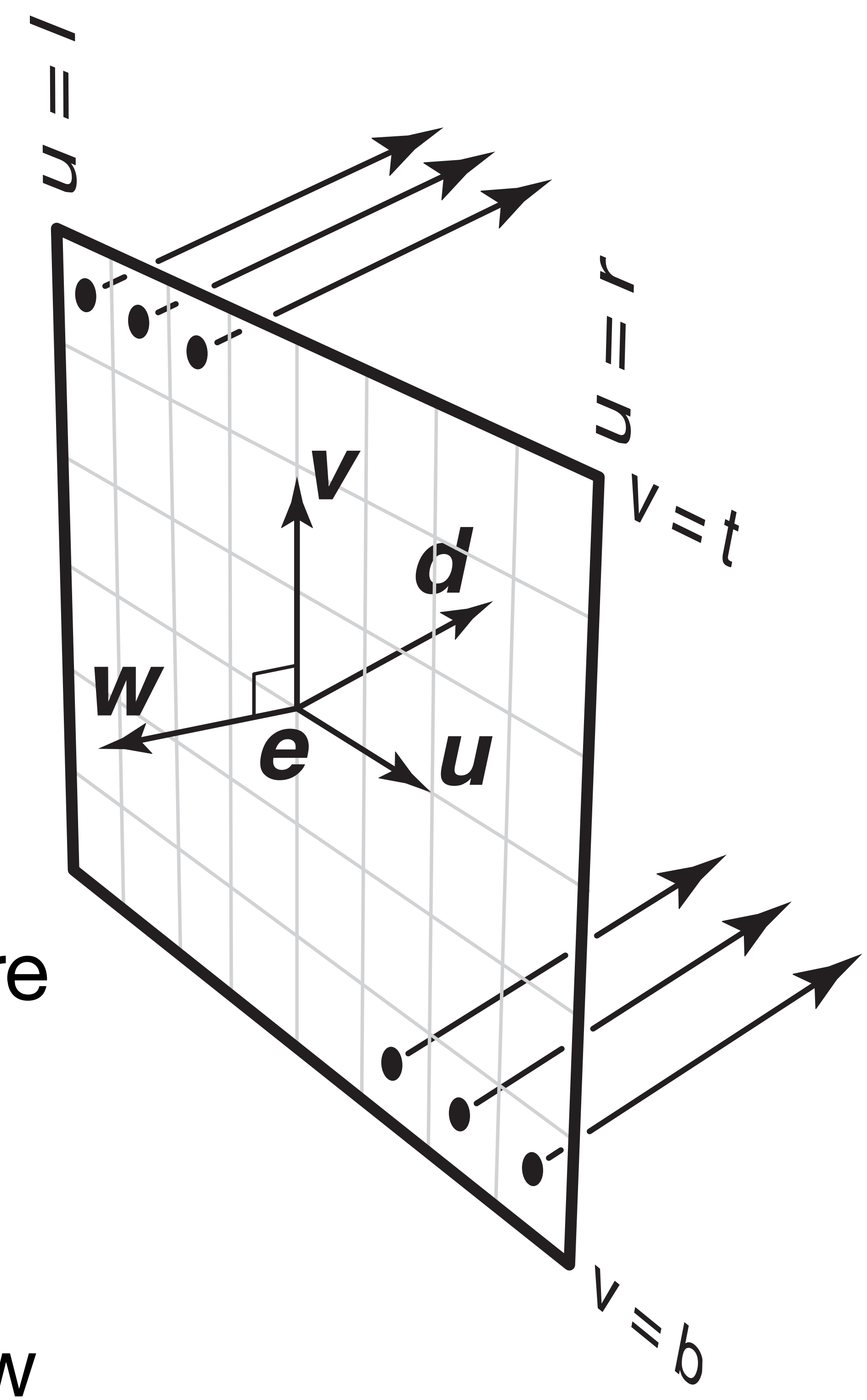


Orthographic: viewing rays are perpendicular to projection plane.

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

Funky Parallel Projections

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

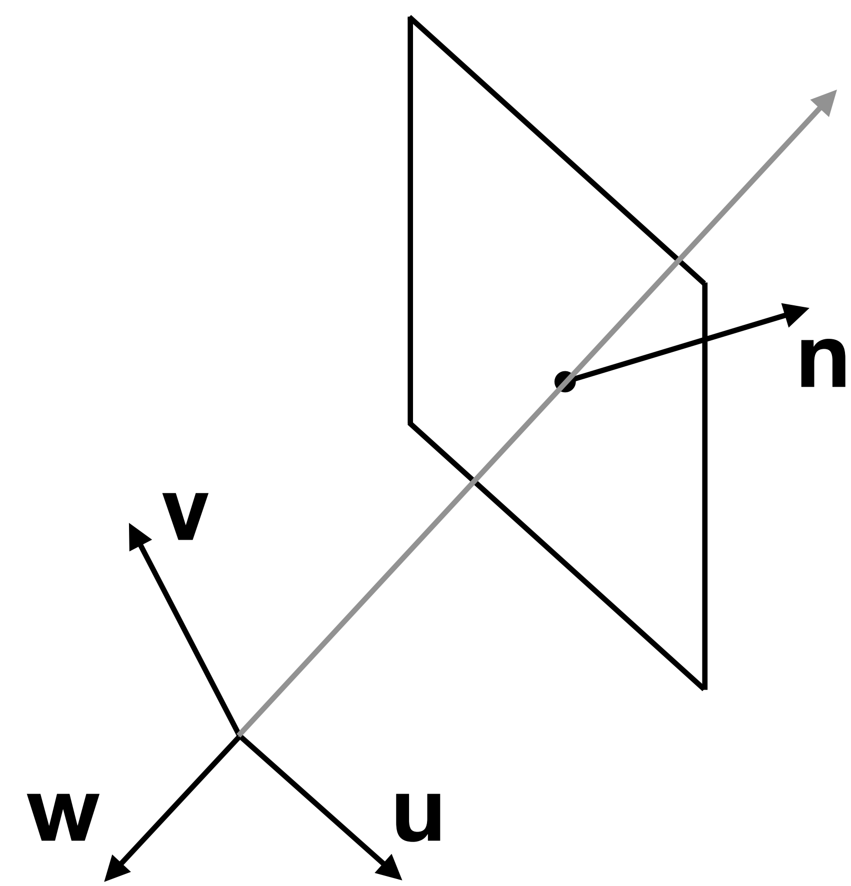


Oblique parallel: viewing rays are *not* perpendicular to projection plane.

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

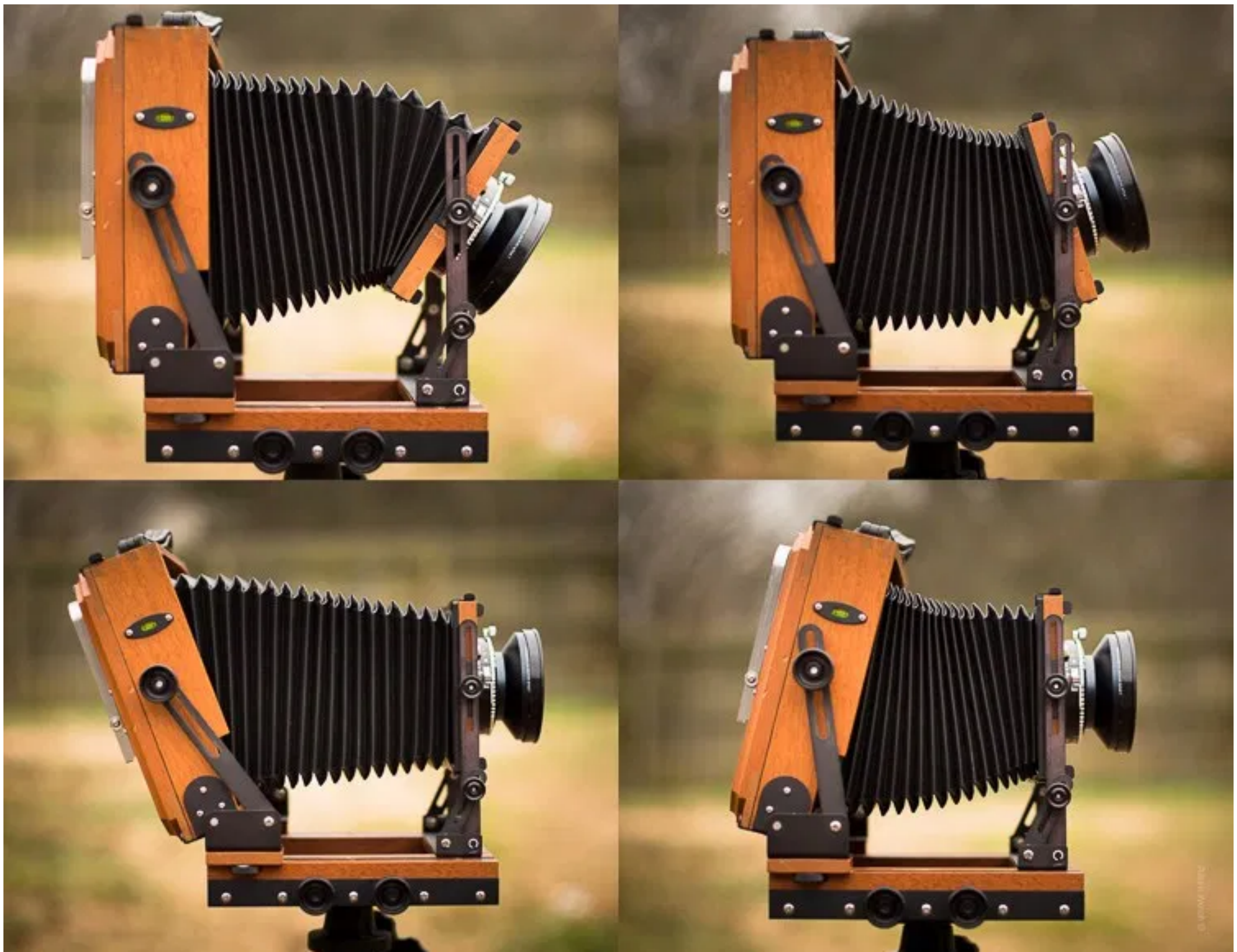
Funky Perspective Projections

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



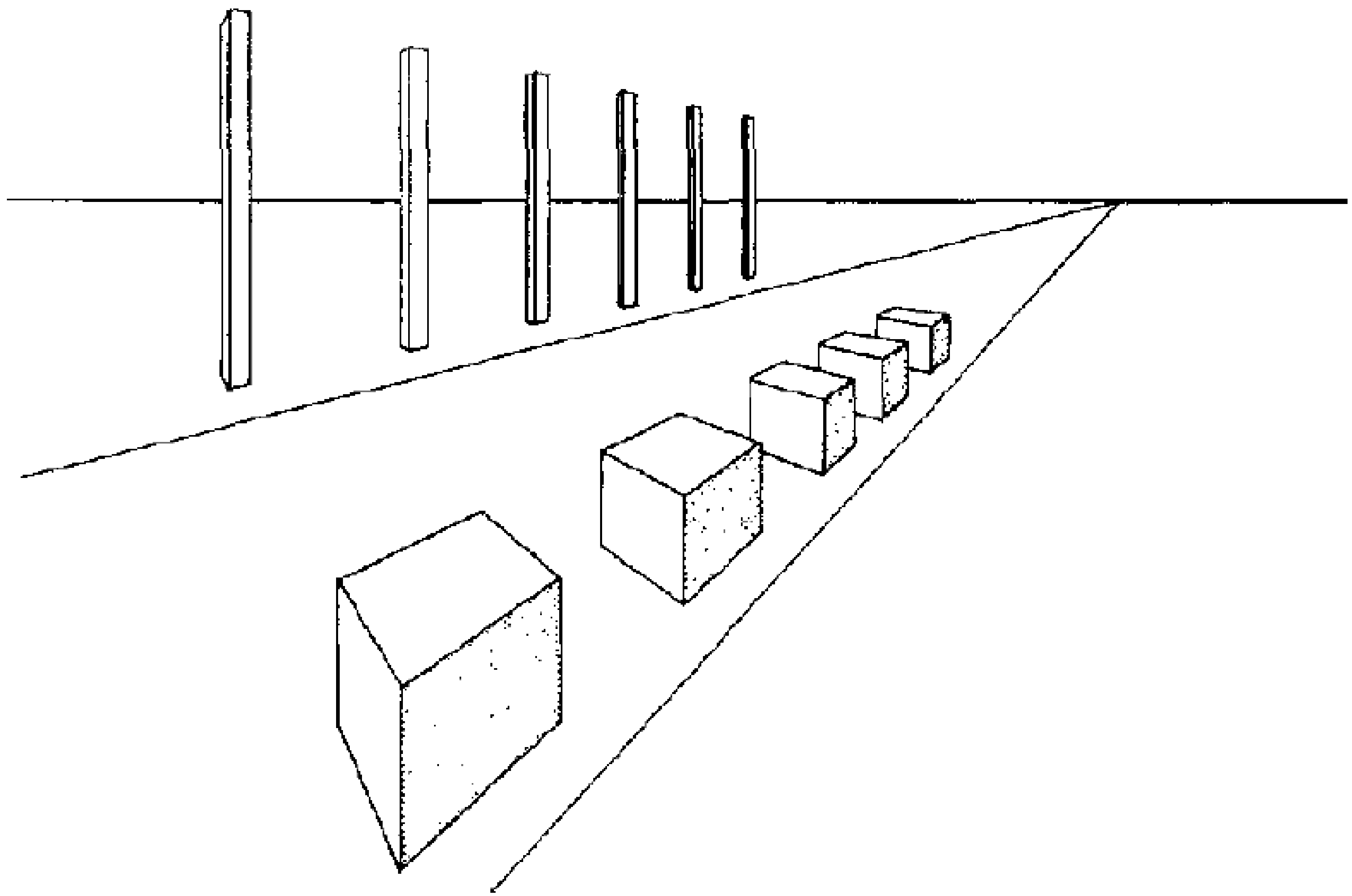
...why do we want this?

Funky Perspective Projections: IRL



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

Problems

- Create a camera basis given an **at** point, i.e., a 3D point that the camera should be looking at, instead of a **view** direction.
- Generate viewing rays for an orthographic projection, given a basis **u, v, w**.
- The "view volume" associated with a projection is the volume of 3D space that projects onto the image plane/viewport. Describe (informally) the **shape** of the view volume for an orthographic camera and a perspective camera.