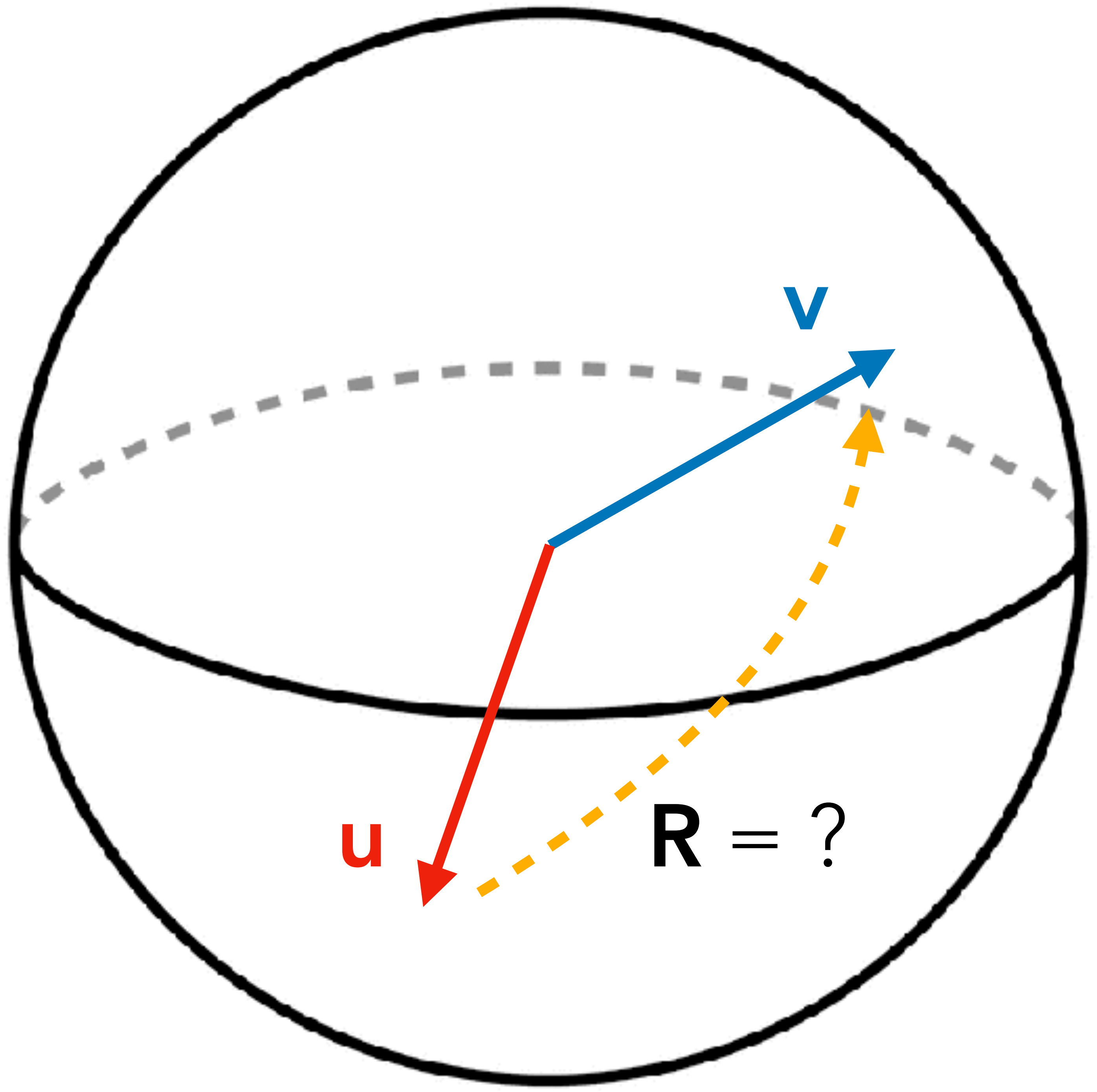
COL781: Computer Graphics

6. Perspective Projection

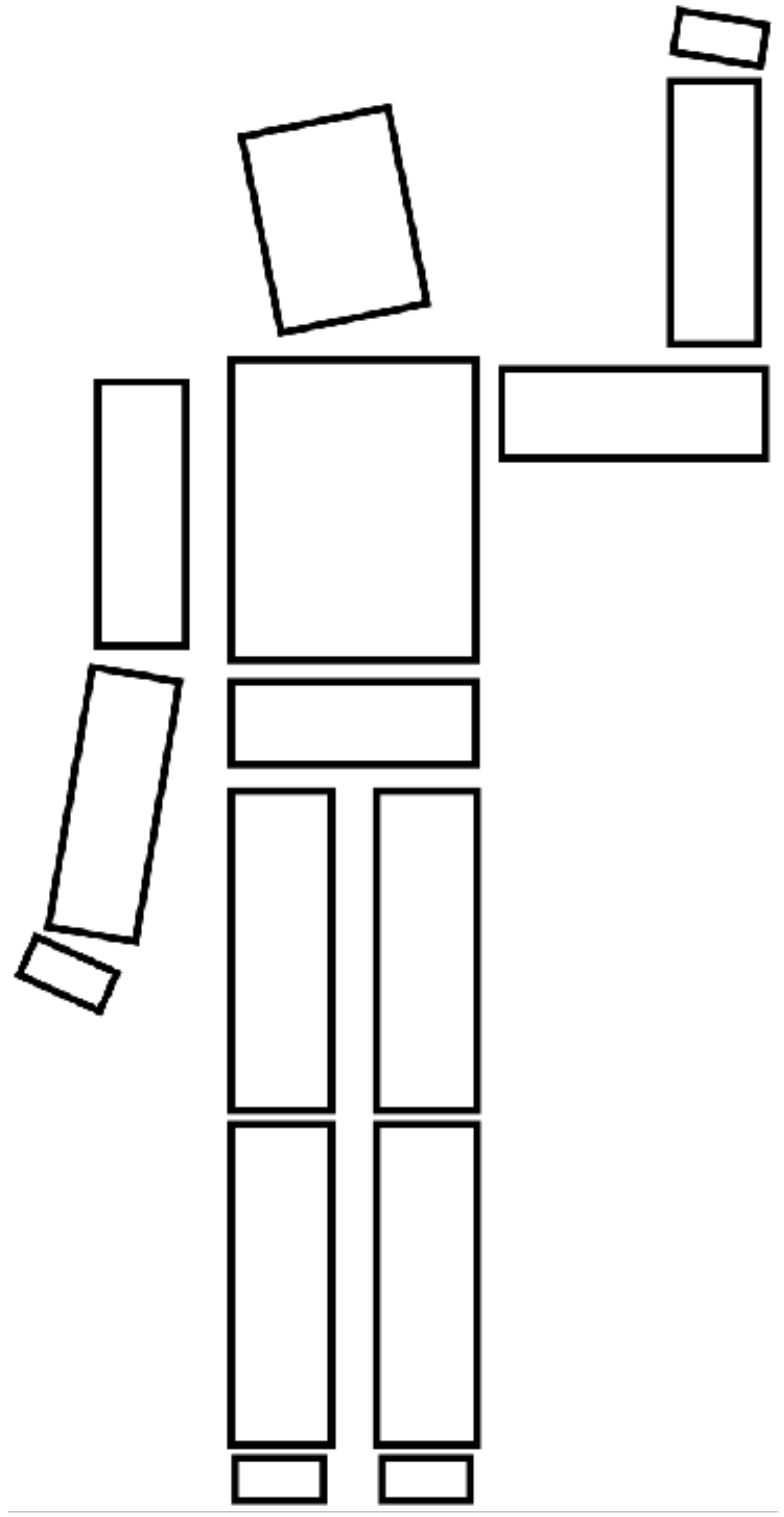
Luciano Testoni

Last class's homework

Given unit vectors **u** and **v**, find a way to construct a rotation matrix **R** which maps **u** to **v**, i.e. **Ru** = **v**. Is it unique, or are there many different such rotations?

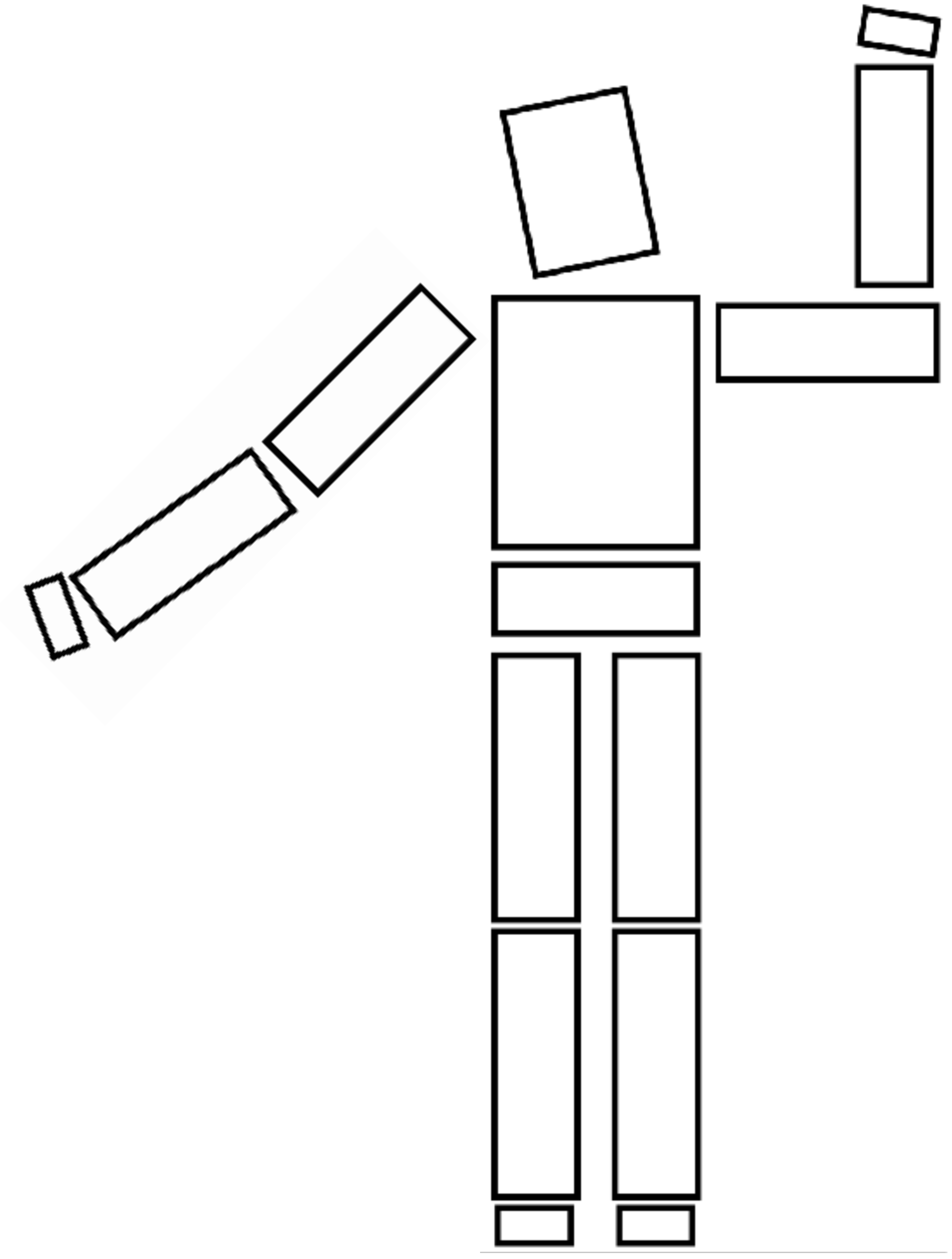


Hierarchical transformations

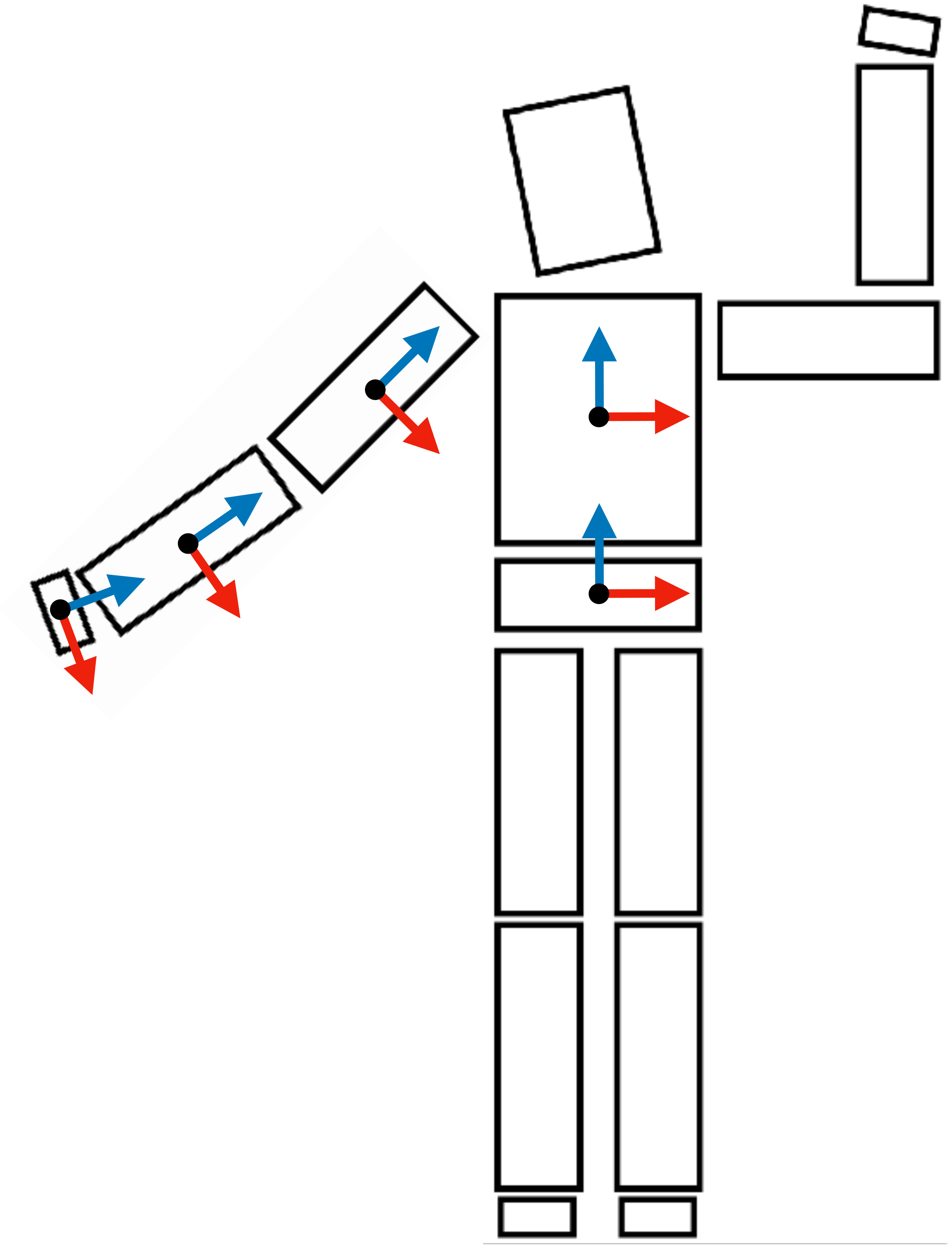


hip
- chest
  - head
  - left upper arm
    - left lower arm
      - left hand
  - right upper arm
    - right lower arm
      - right hand
- left upper leg
  - …
- …

Hierarchical transformations



- hip
  - chest
    - head
    - **left upper arm**
      - **left lower arm**
        - **left hand**
    - right upper arm
      - right lower arm
        - right hand
  - left upper leg
    - …
  - …



Shapes are specified in the corresponding part's local coordinate frame

Each part's transformation is relative to its parent

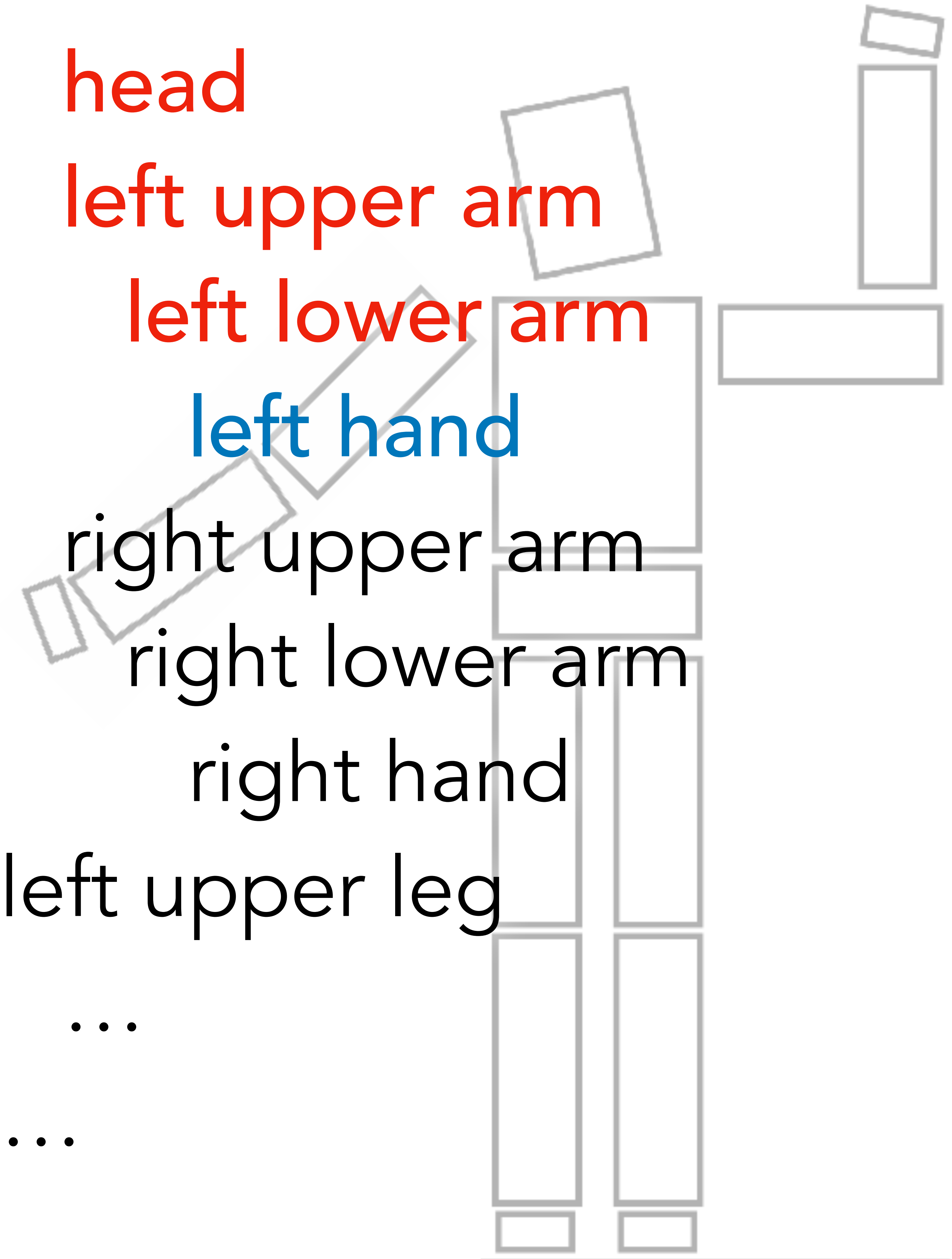
Given point in left hand frame,

```
Vec3 world_point
   = world_from_hip * hip_from_chest
   * chest_from_ularm * uarm_from_llarm
   * llarm_from_lhand * lhand_point
```

or simply

```
Mat3x3 world_from_lhand
   = world_from_hip * hip_from_chest
   * chest_from_ularm * uarm_from_llarm
   * llarm_from_lhand
```

hip
  chest
    head
    left upper arm
      left lower arm
        left hand
    right upper arm
      right lower arm
        right hand
  left upper leg
    …
  …



```
Mat3x3 world_from_lhand
   = world_from_hip * hip_from_chest
   * chest_from_ularm * uarm_from_llarm
   * llarm_from_lhand

Mat3x3 world_from_lhand
   = world_from_llarm * llarm_from_lhand
```

Going down the tree:

Push parent's matrix on stack

Multiply child's matrix on right

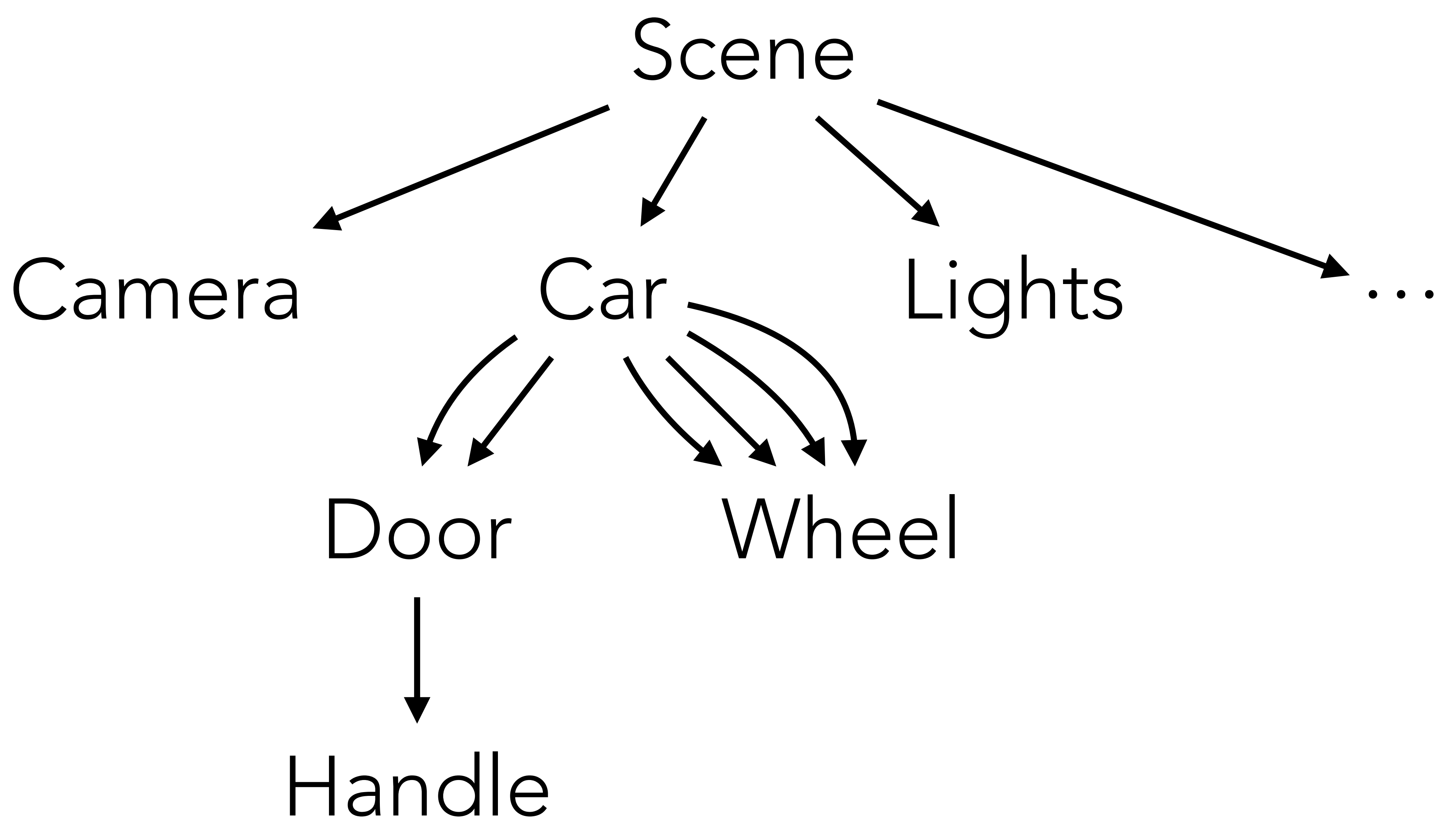
Going back up: Pop parent's matrix from stack

Scene graph

Usually the entire scene is represented as a tree / DAG!

Nodes may contain geometry or other content

Edges contain transformations



Why a DAG? So we can reuse the same geometry multiple times: **instancing**

So far we know:

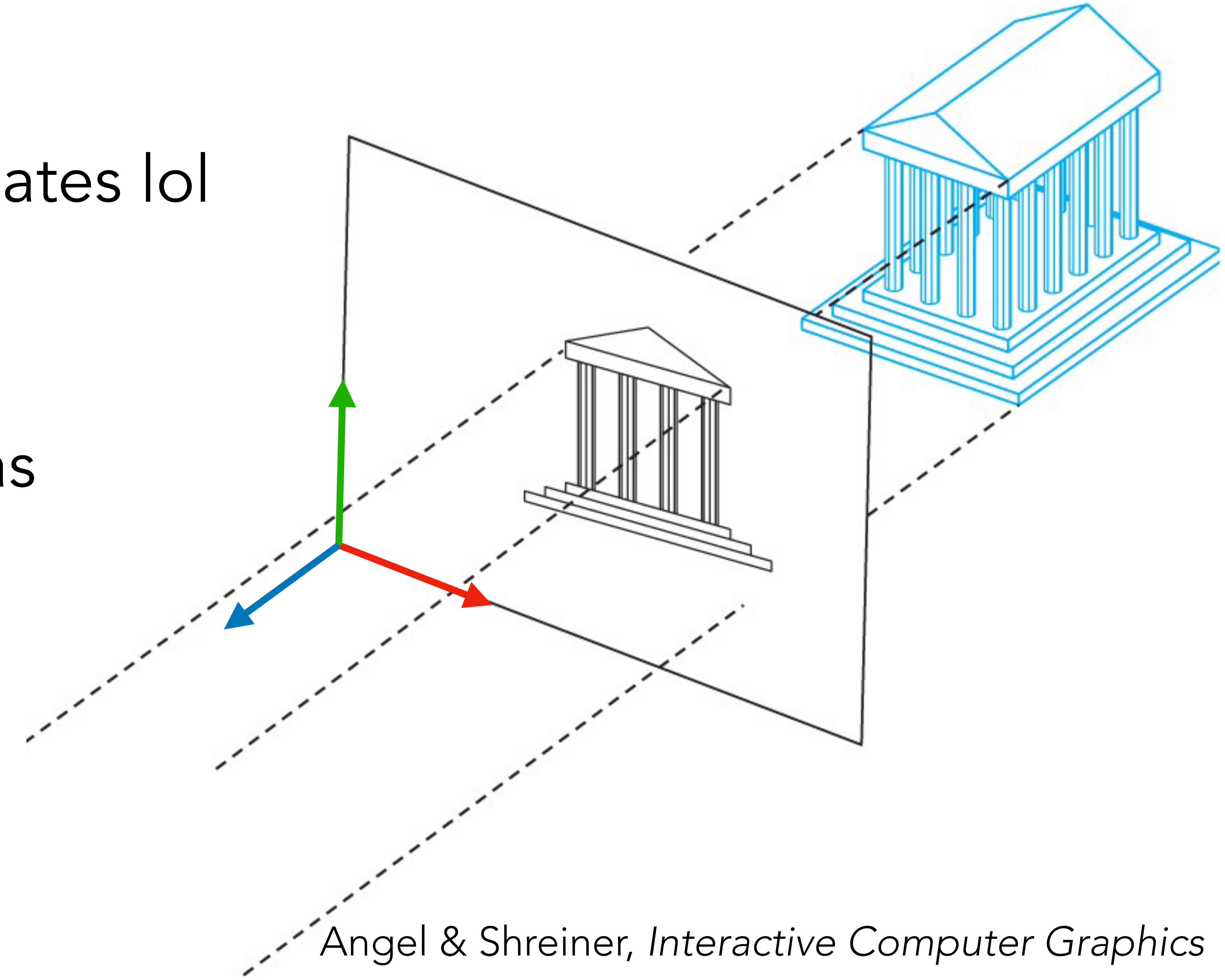
- How to draw 2D shapes
- How to transform 2D and 3D shapes

Today: How to draw 3D shapes on a 2D screen?

Parallel projection

Easy way: Just drop one of the coordinates lol

- Useful for engineering drawings
- Doesn't match how eyes and cameras actually see things!



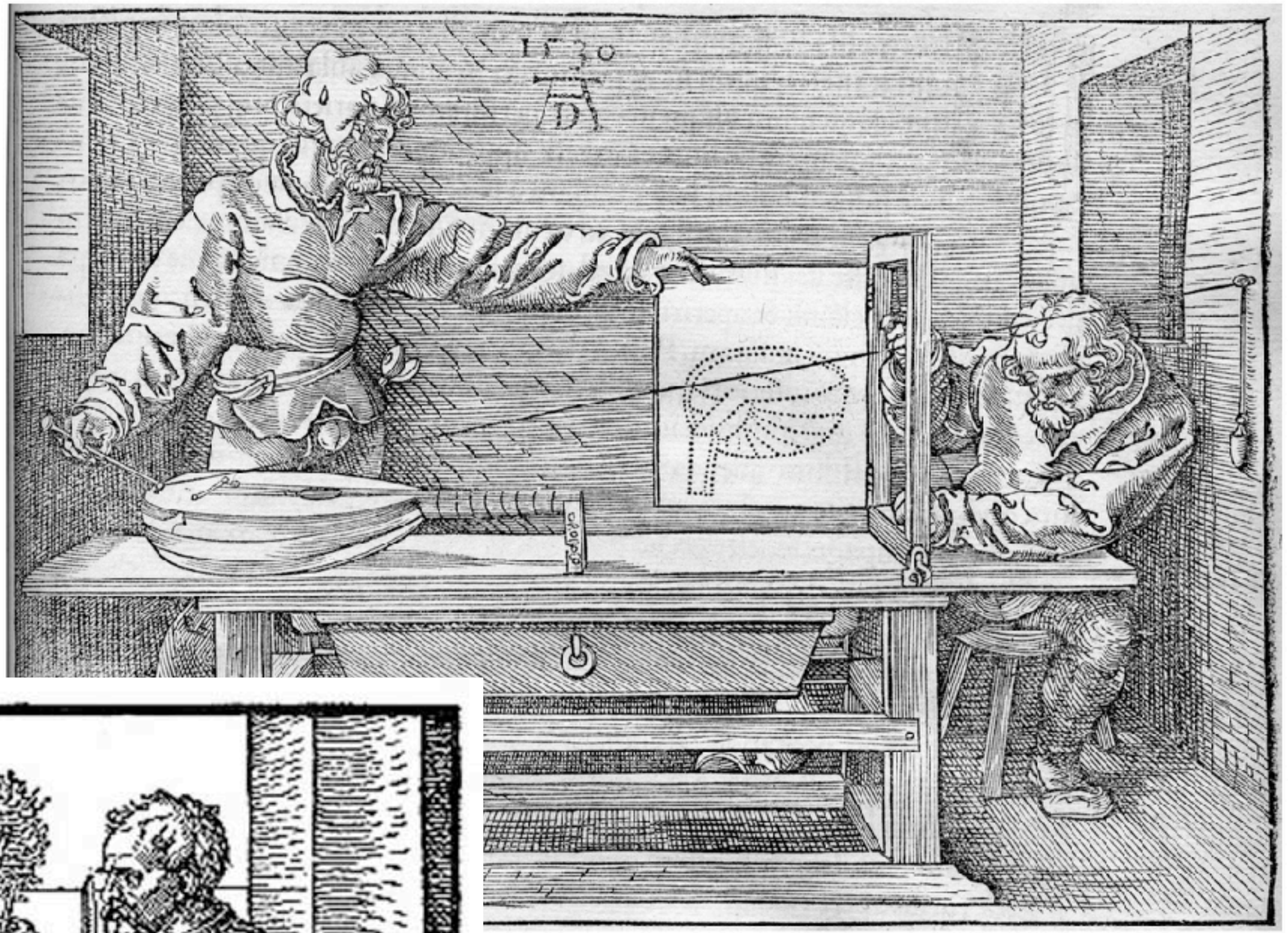
Angel & Shreiner, *Interactive Computer Graphics*

Perspective

Jeff Lynch

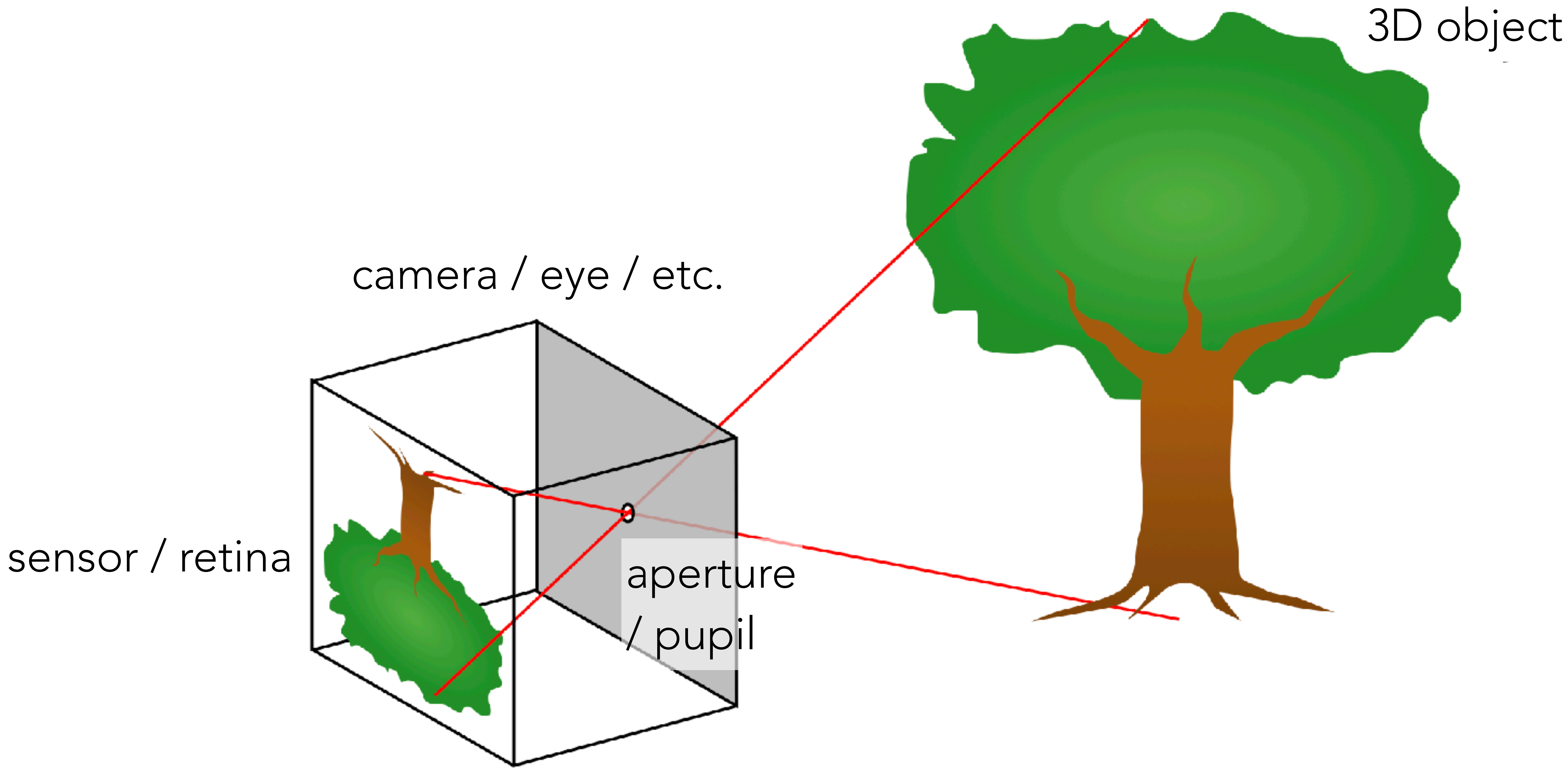
Algorithmic drawing in the 1500s

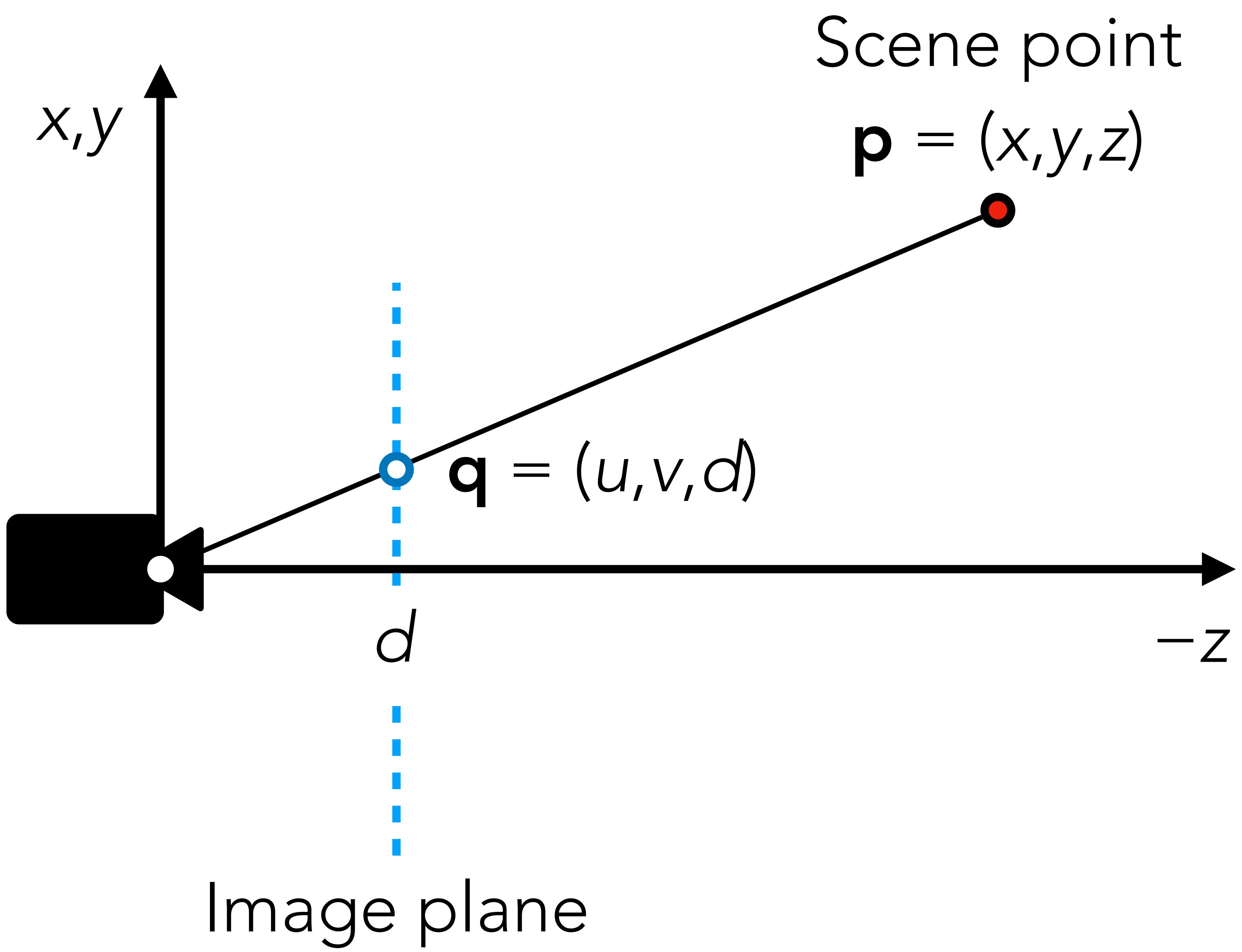
A point is drawn where the ray from the viewpoint meets the image plane.



Albrecht Dürer

Pinhole camera model





Assume camera is at the origin, pointing in the direction $-z$.

Where is the point **p** projected to?

$$\frac{x}{z} = \frac{u}{d}$$

$$u = \frac{xd}{z}$$

Similarly $v = yd/z$

(W.l.o.g., let's take $d = -1$)

What if the camera is not at the origin and/or not looking along $-z$?

Just change to a coordinate system in which it is.

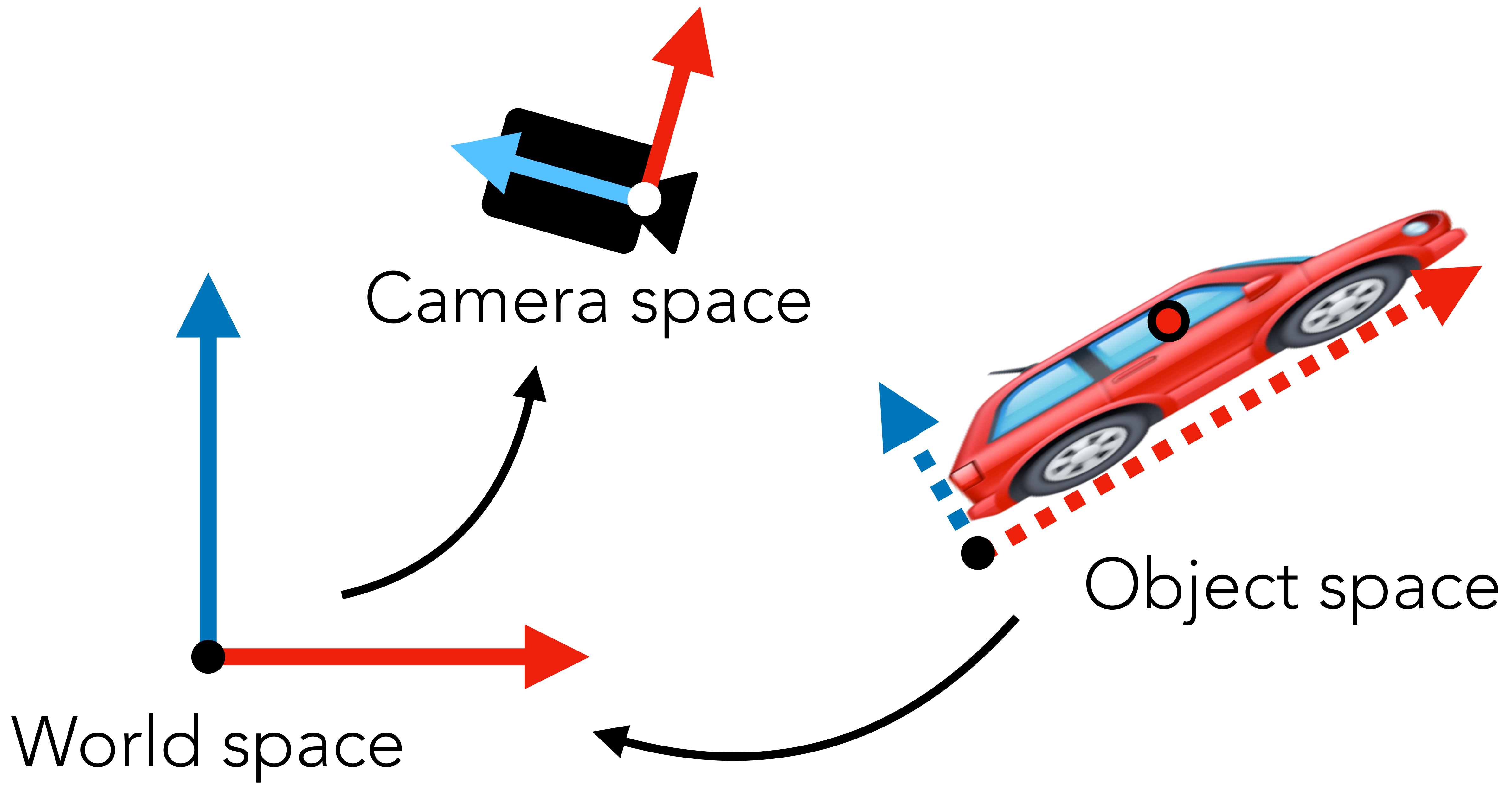
Viewing transformation

Usually, user specifies:

- center of projection $\mathbf{c}$
- target point $\mathbf{t}$ or view vector $\mathbf{v} = (\mathbf{t}-\mathbf{c})/\|\mathbf{t}-\mathbf{c}\|$
- "up vector" $\mathbf{u}$

Construct orthonormal basis

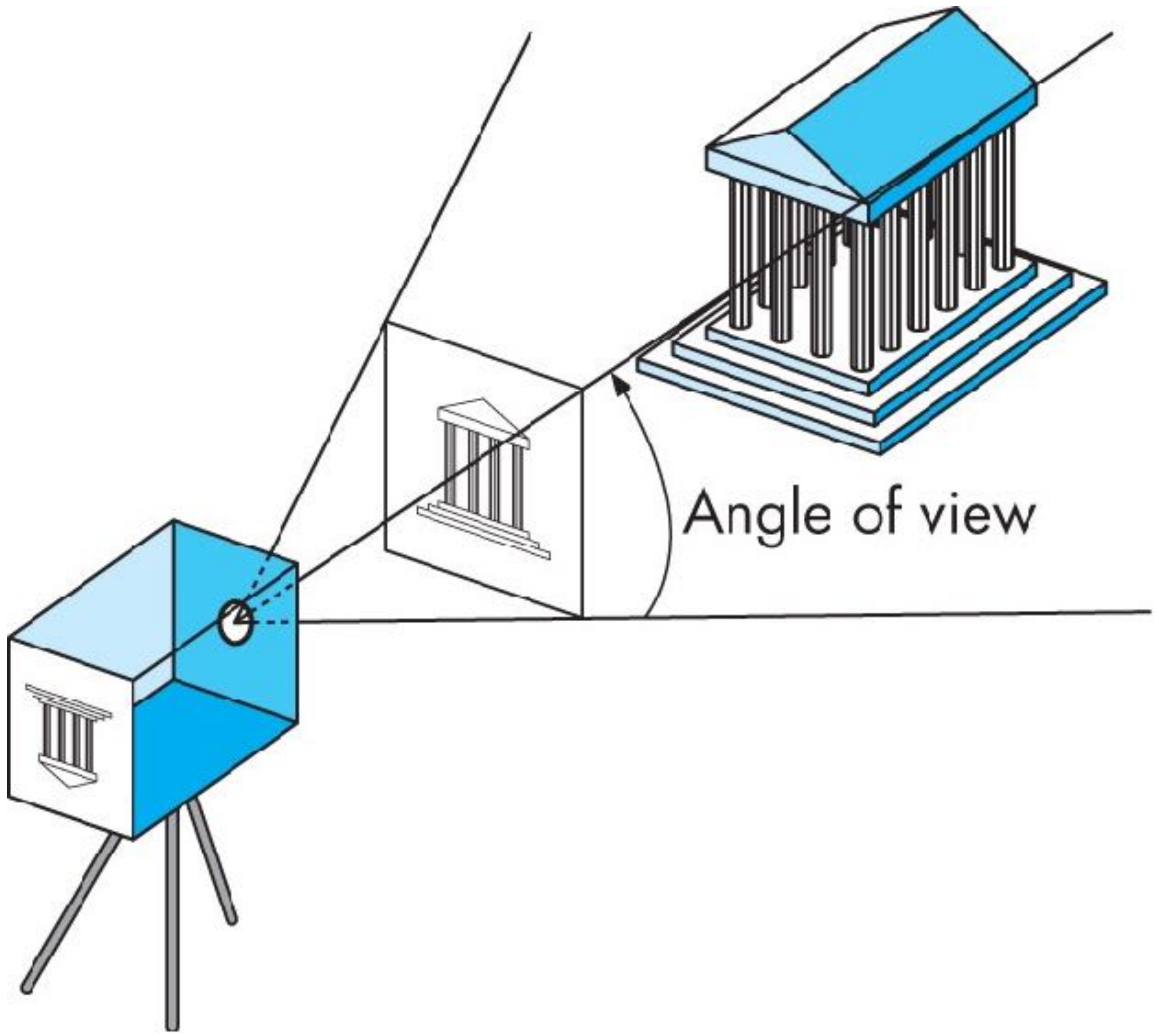
$$\mathbf{e}_2 = (\mathbf{v}\times\mathbf{u})/\|\mathbf{v}\times\mathbf{u}\|$$
$$\mathbf{e}_1 = \mathbf{v}\times\mathbf{e}_2$$
$$\mathbf{e}_3 = -\mathbf{v}$$



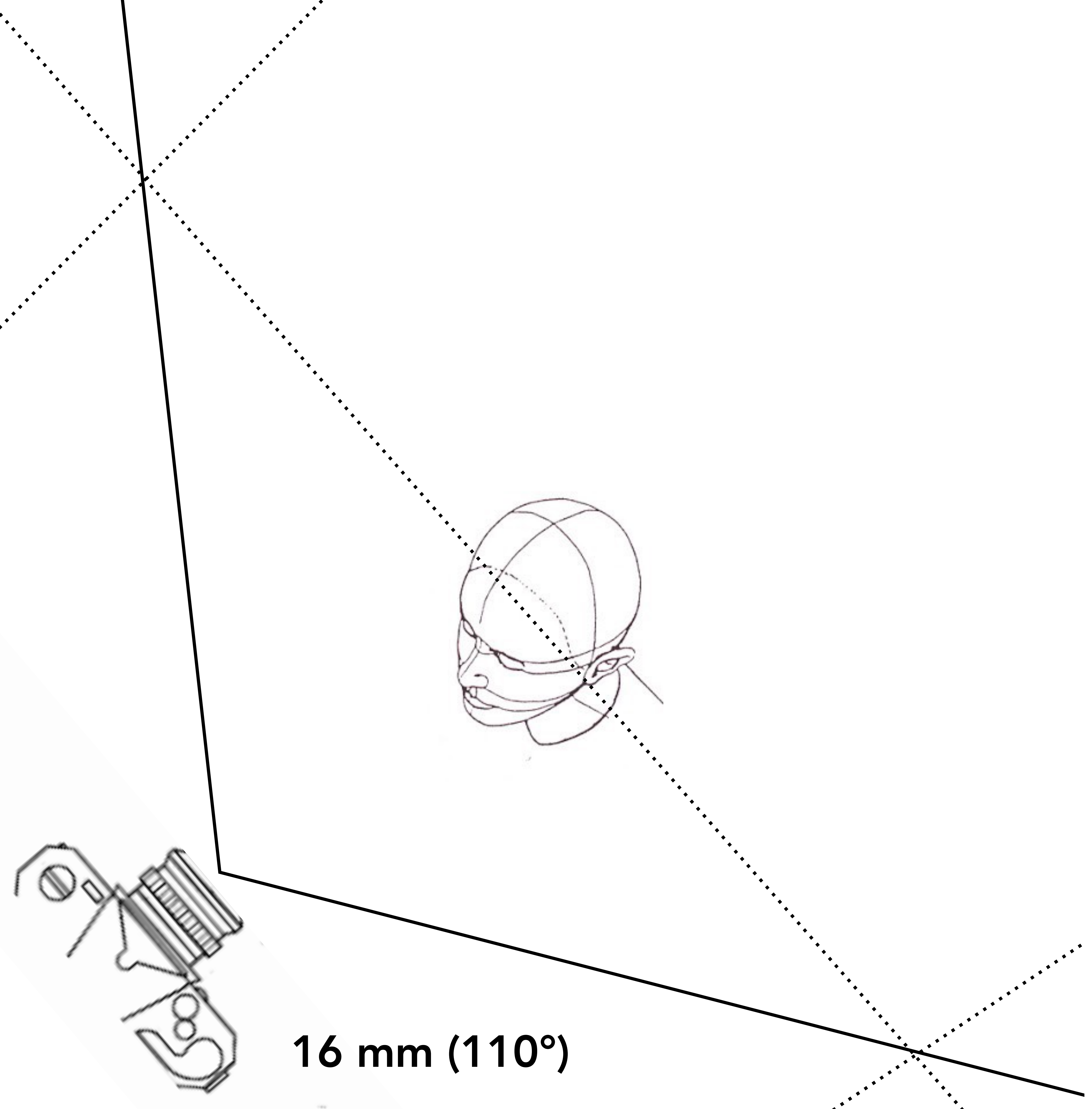
Camera → world: $\mathbf{M} = [\mathbf{e}_1 \ \ \mathbf{e}_2 \ \ \mathbf{e}_3 \ \ \mathbf{c}]$

World → camera: $\mathbf{M}^{-1}$

Once point is in camera space, projected point = $\begin{bmatrix} xd/z \\ yd/z \end{bmatrix}$

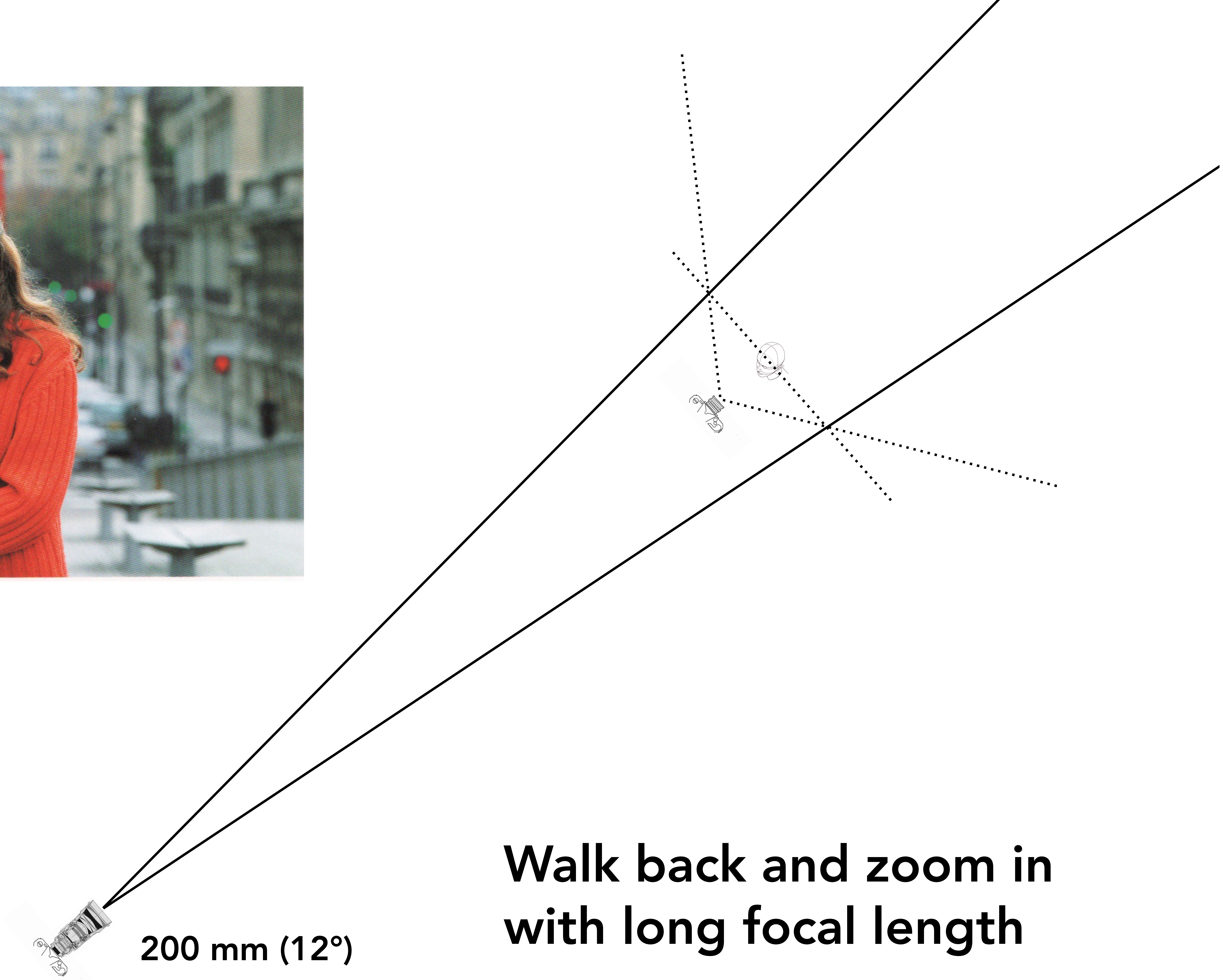


Canon EF Lens Work III



16 mm (110°)

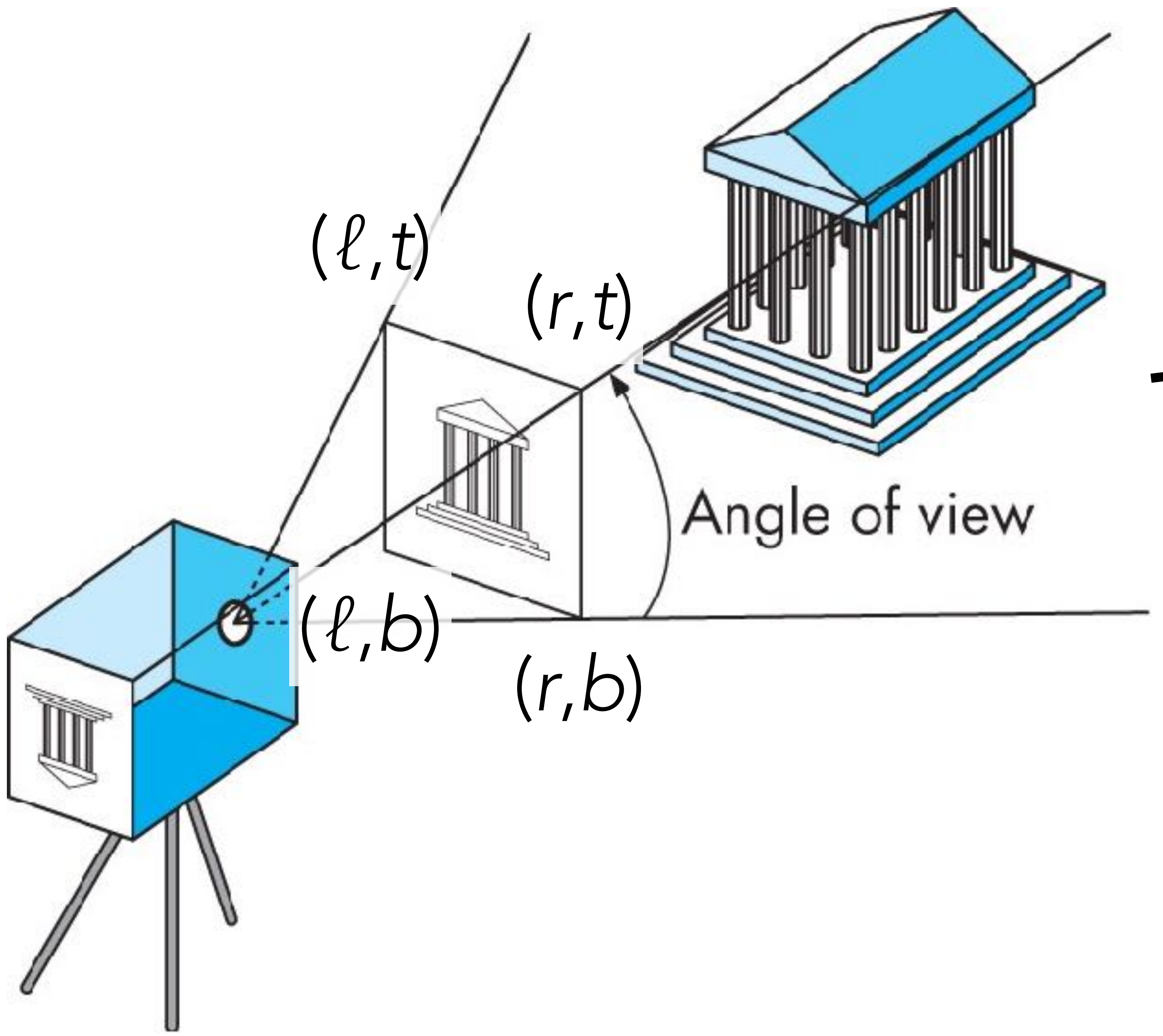
Up close and zoomed wide with short focal length



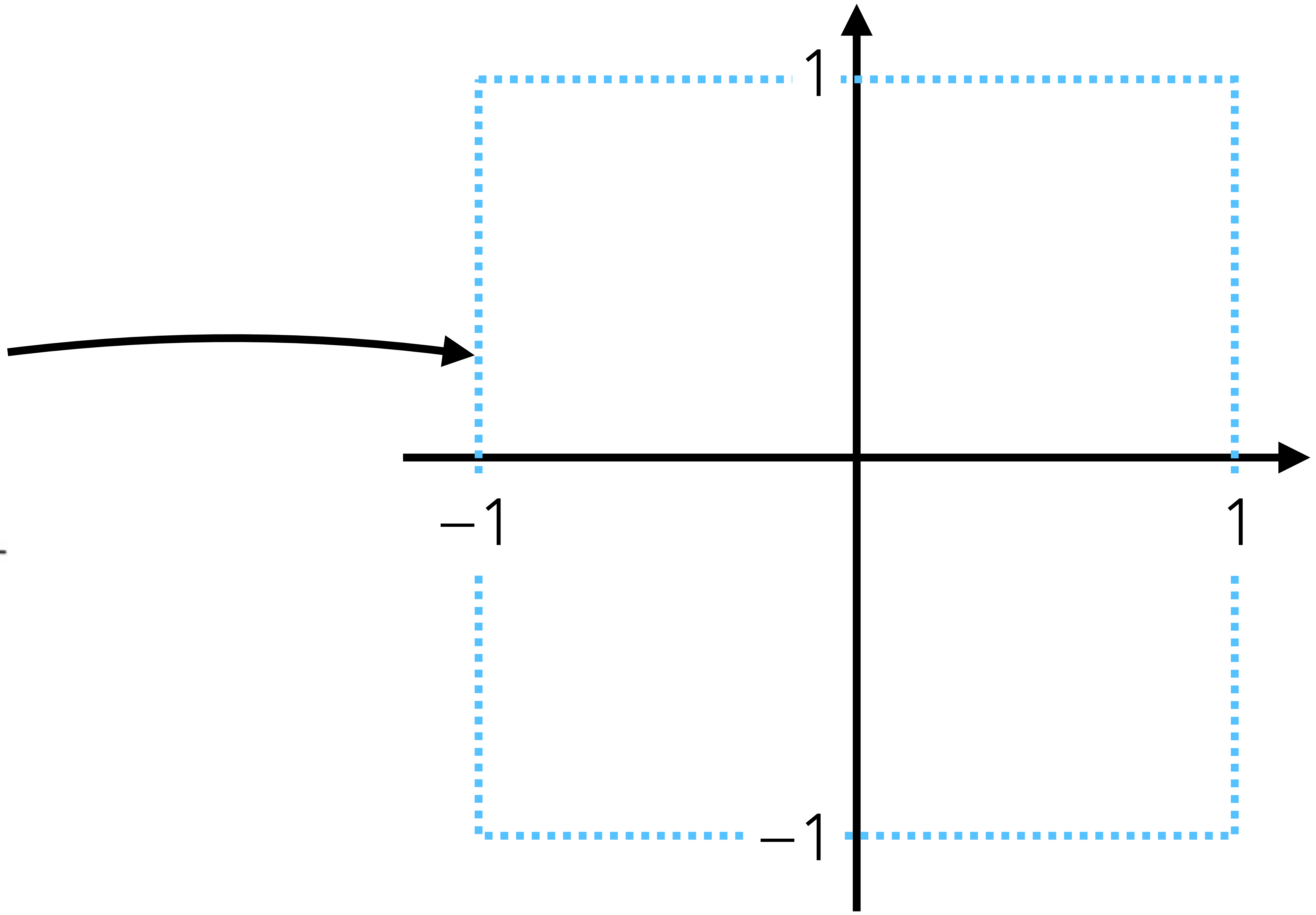
200 mm (12°)

Walk back and zoom in with long focal length

Ren Ng



Coordinates after perspective division



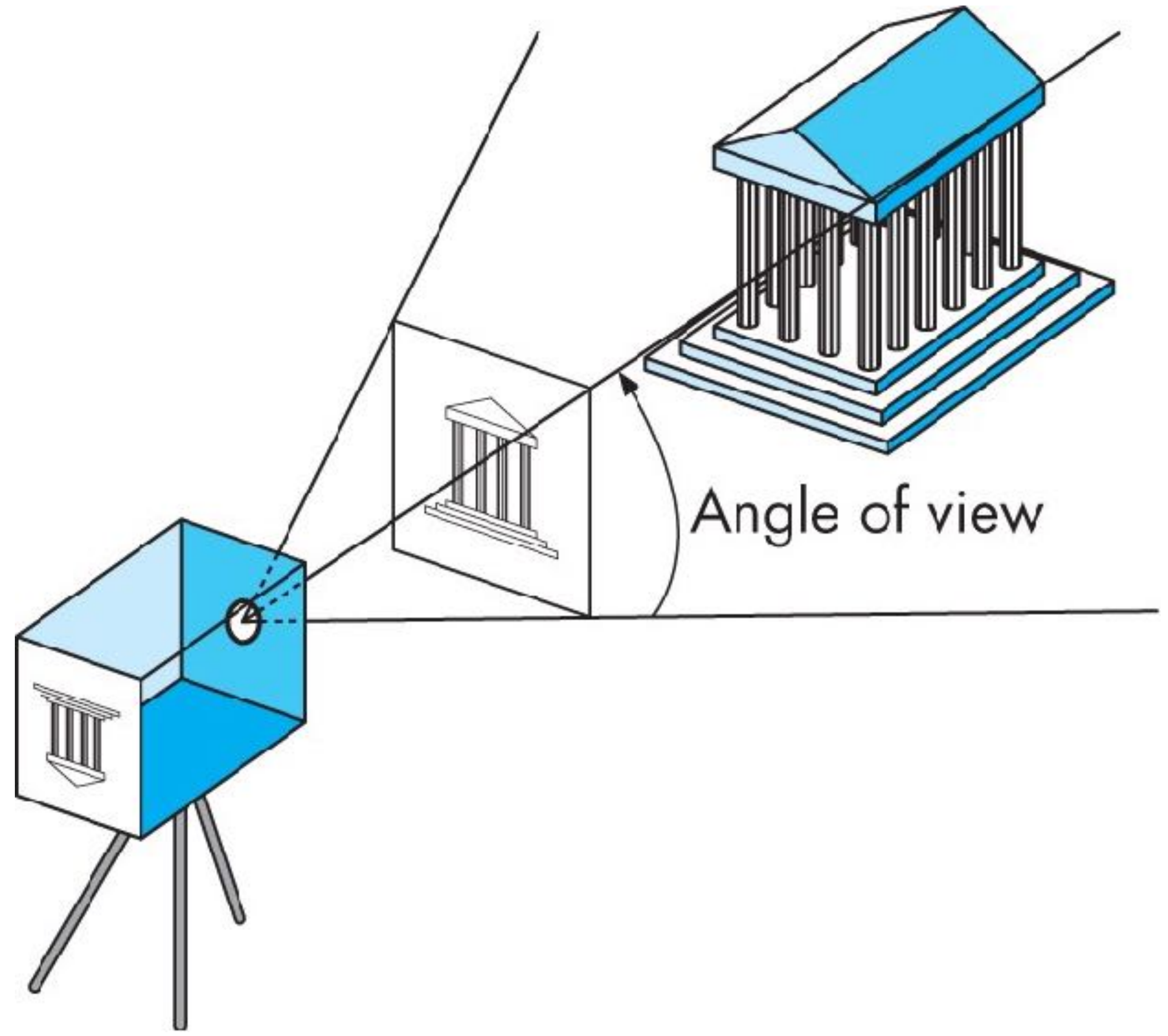
"Normalized device coordinates"

(Actually there's a bit more in NDC… Will correct later!)

Choose transformation so that points in field of view fall inside $[-1,1] \times [-1,1]$

Puzzle:

What is the maximum possible angle of view in perspective projection?



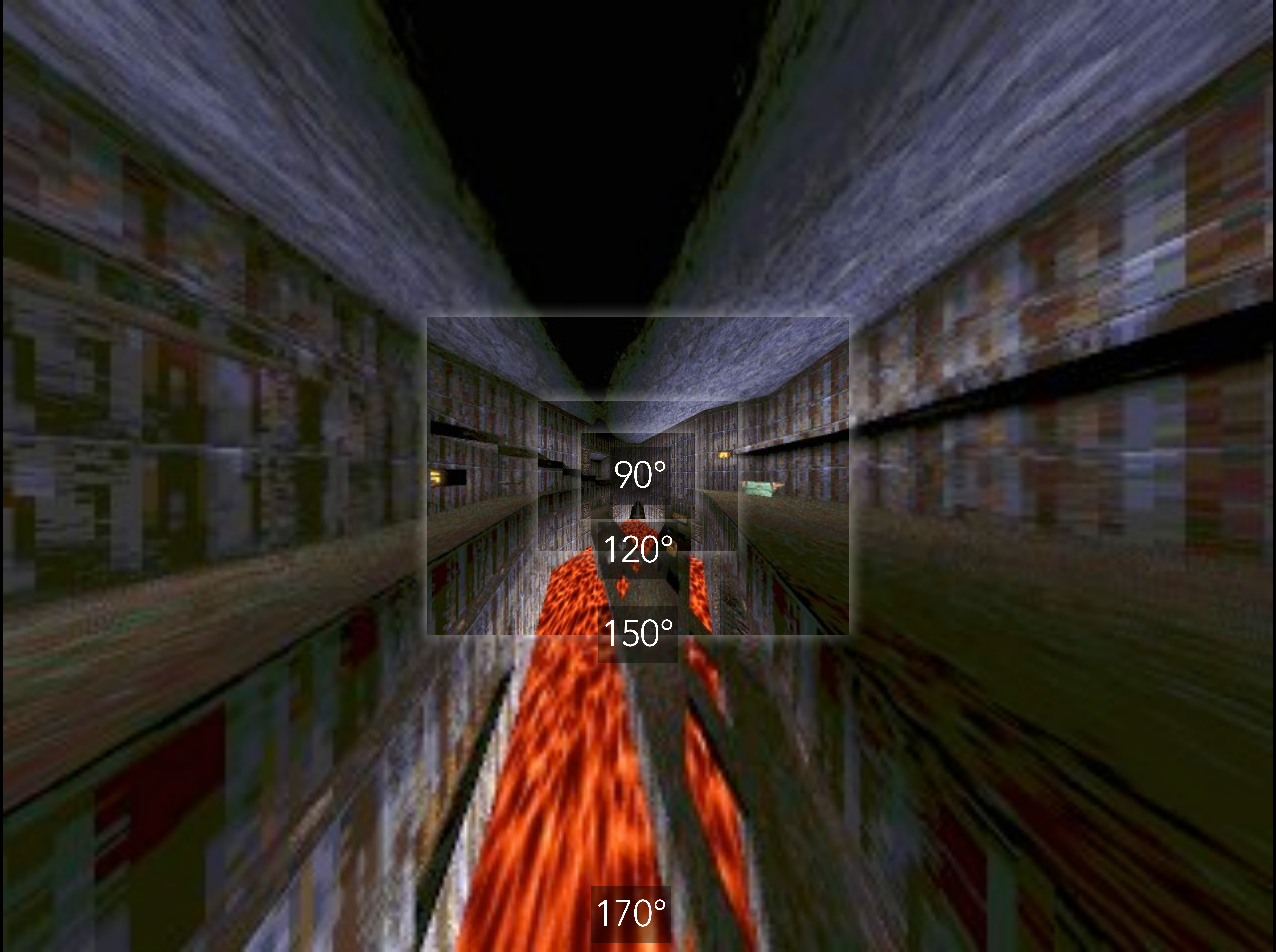
Why does no graphics application or game let you set your angle of view to anything remotely close to it?

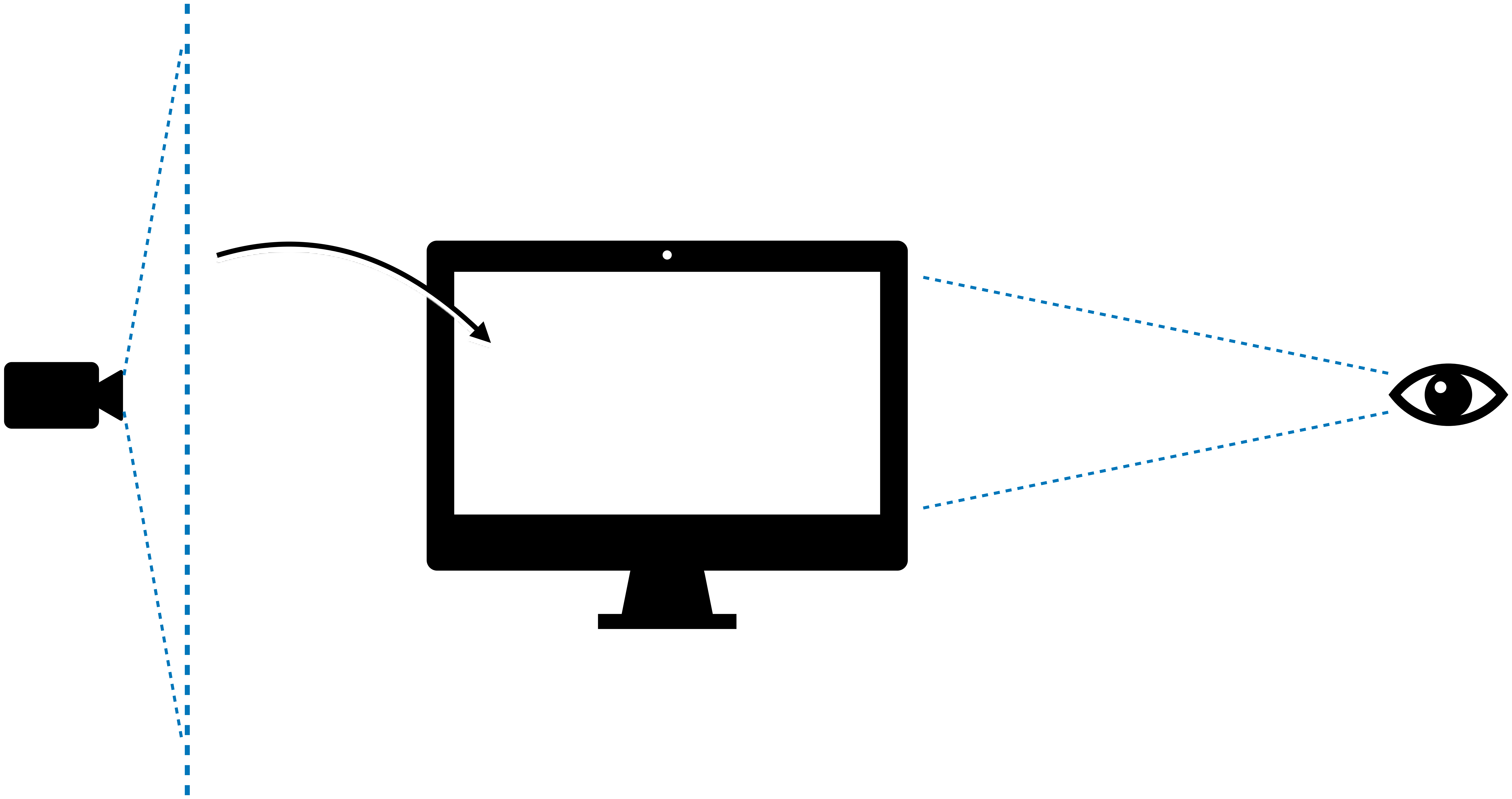
Wouter van Oortmerssen

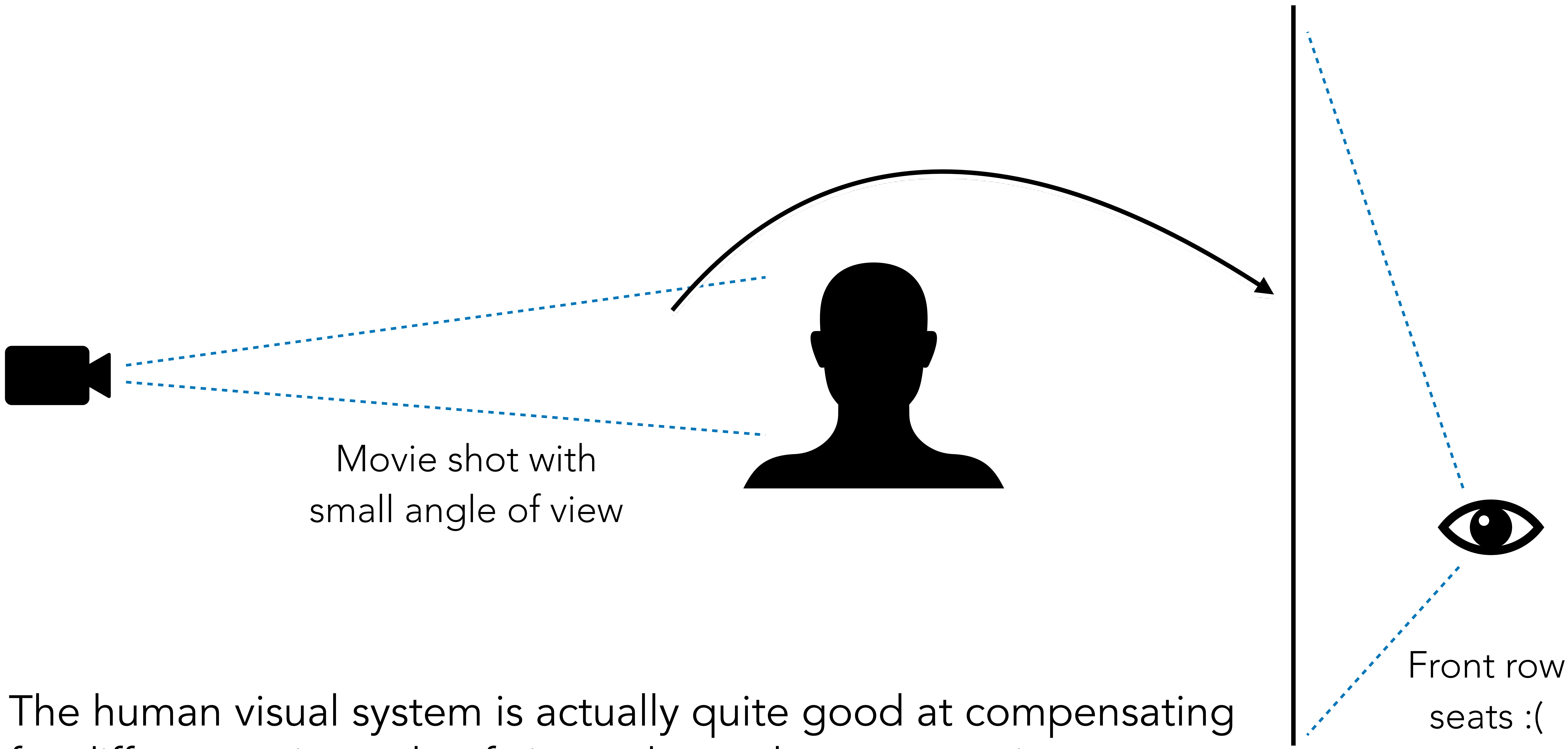
Wouter van Oortmerssen

Wouter van Oortmerssen

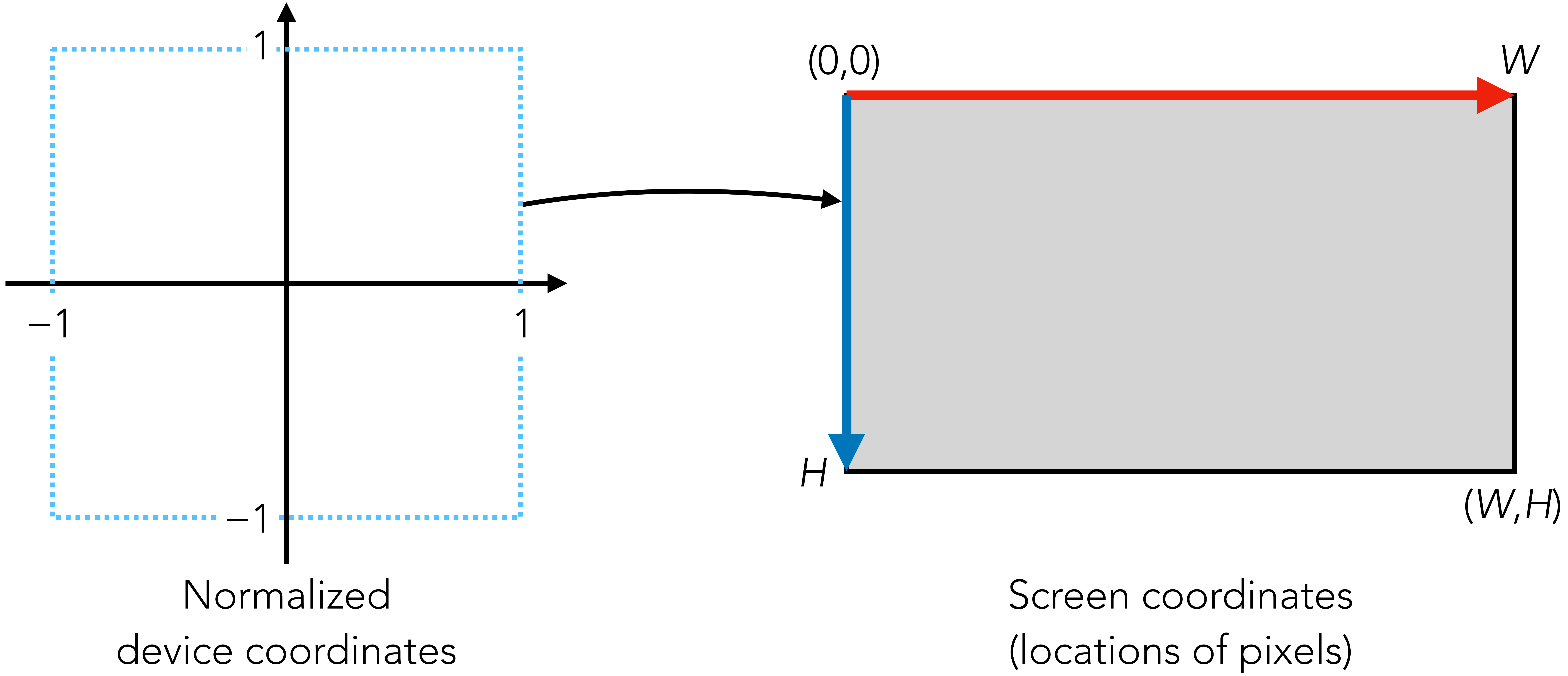
Wouter van Oortmerssen


90°
120°
150°
170°

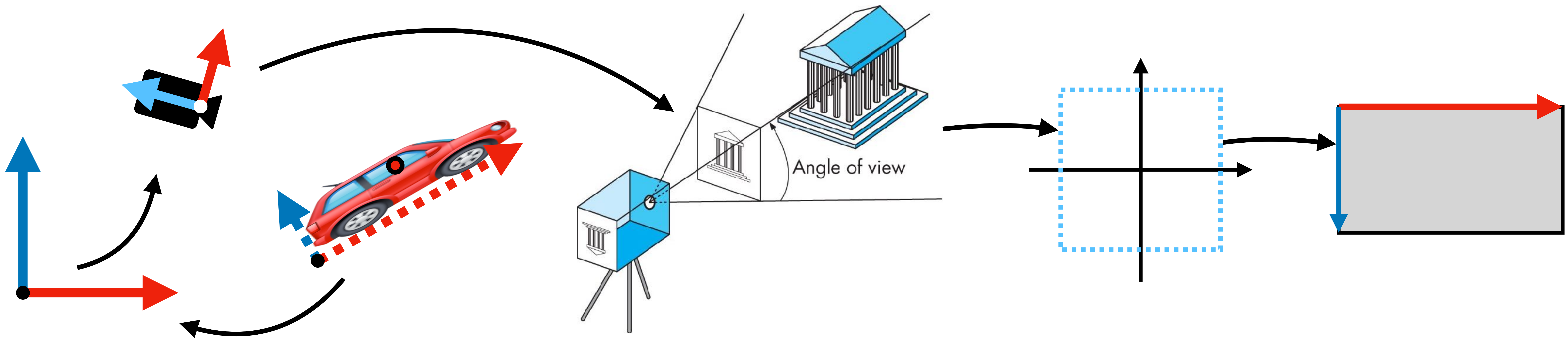




The human visual system is actually quite good at compensating for differences in angle of view… but only up to a point.



And finally, we can rasterize our triangles!



- Object space → world space
- World space → camera space
- Camera space → projection plane (division by *z*)
- Projection plane → NDC
- NDC → screen coordinates

Two problems:

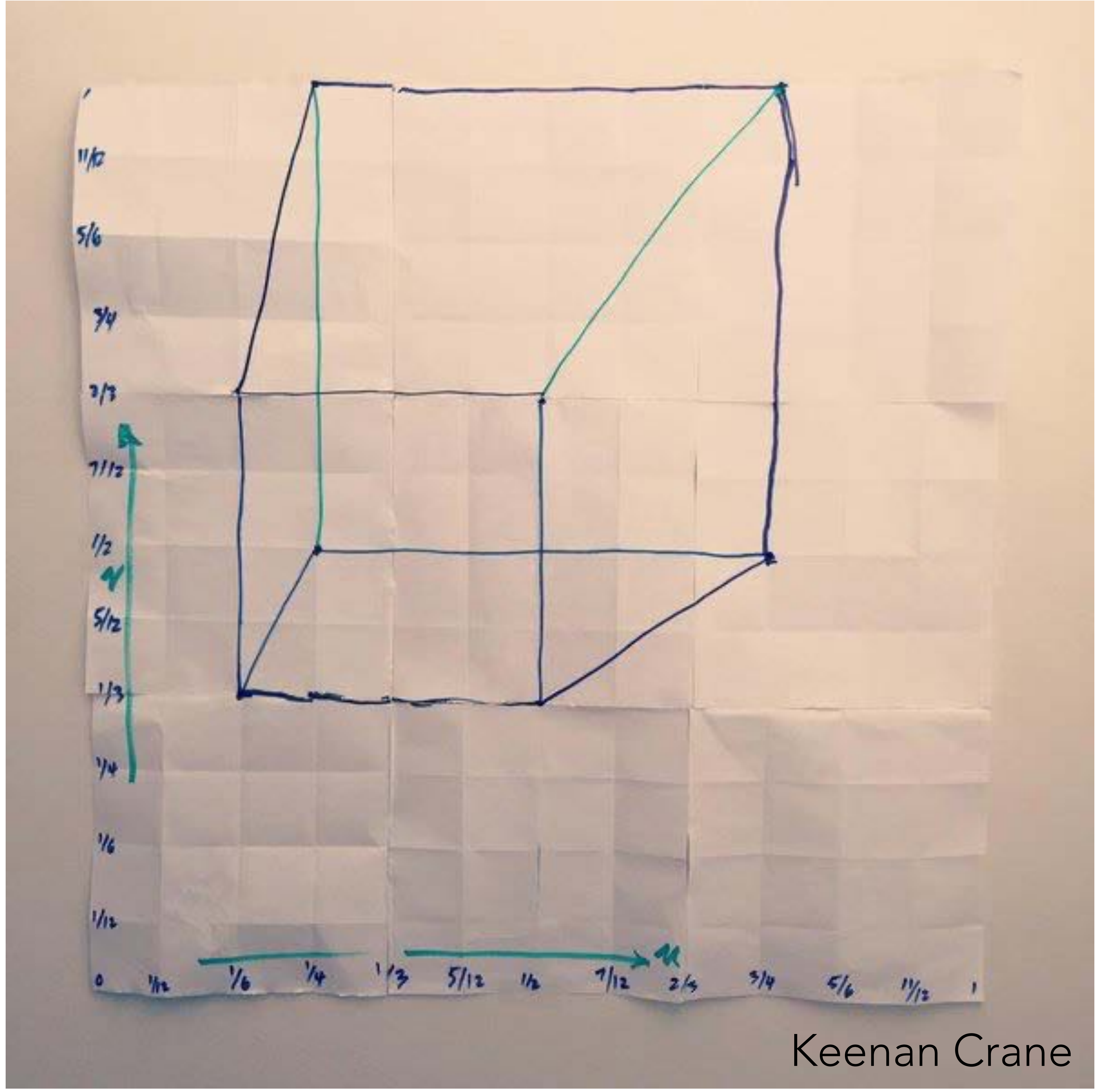
- Every step is a matrix, except perspective division.
- Final result has lost depth information (the *z* coordinate): don't know which points are in front of which

Homework exercise: DIY 3D GFX

Draw a cube! (manually, or with Excel, or using a plotting library)

Start with vertices at (±1,±1,±1), translate somewhere along −z, maybe apply some rotation, then draw the projected points and join them with edges.

Translated by (2,3,−5), no rotation



Keenan Crane