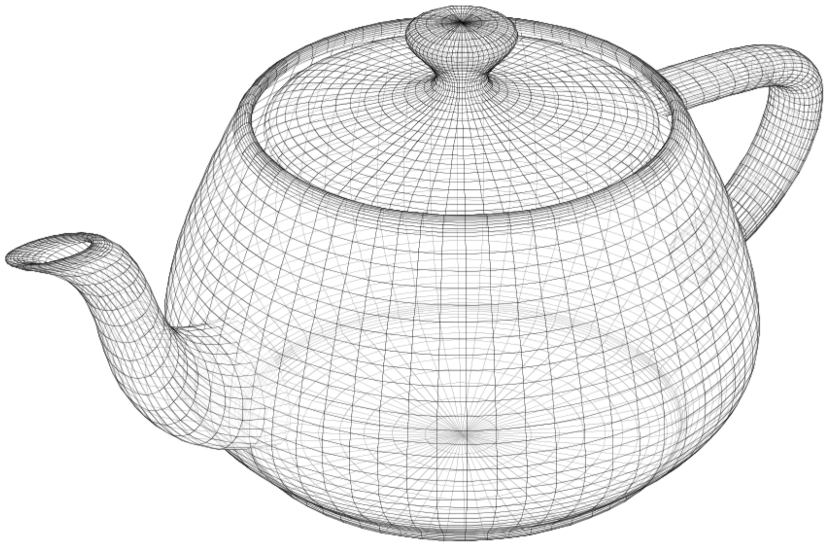
A Camera Model



Production Computer Graphics

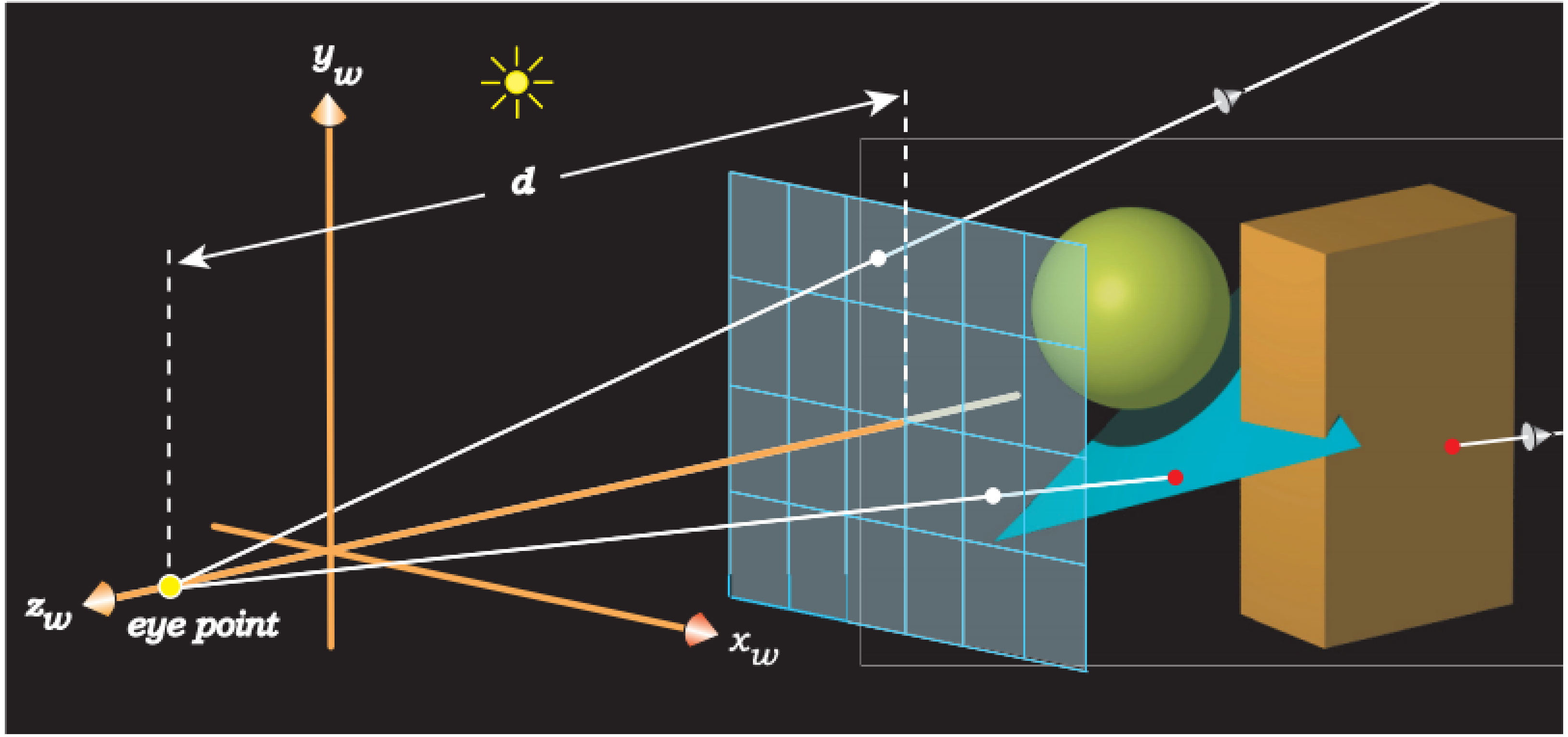
Eric Shaffer

ILLINOIS

Ray Tracing with Axis-Aligned Perspective Viewing

We will use a view direction along the negative z-axis

- …if we trace rays from an eyepoint, we get perspective projection
  - in orthographic projection the rays have different origins but the same direction
  - in perspective projection the rays have the same origin and different directions

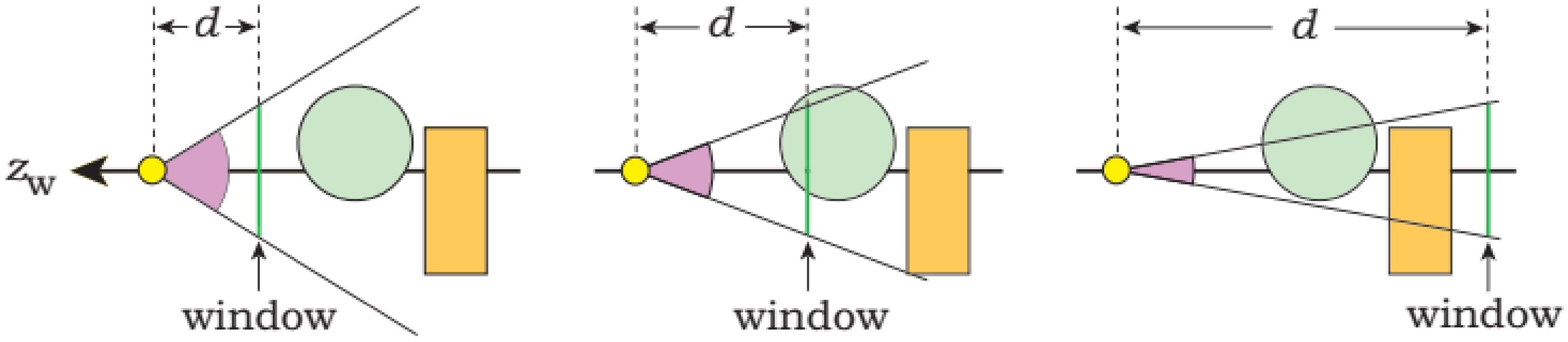


ILLINOIS

Perspective Effects

What camera action does varying d have?

- Increasing d (distance to view plane) decreases field of view
- Increasing d zooms into the scene

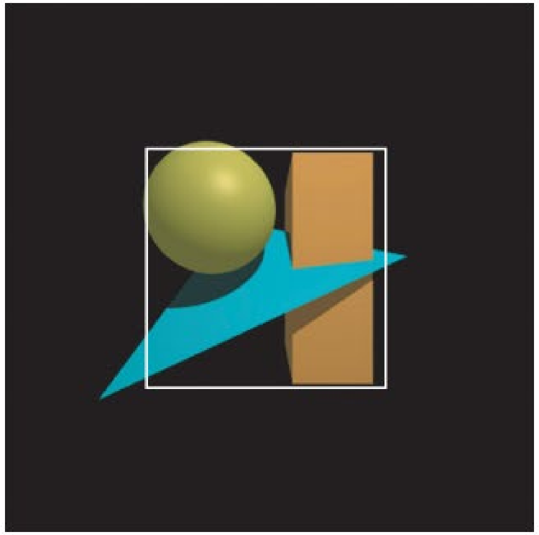


ILLINOIS

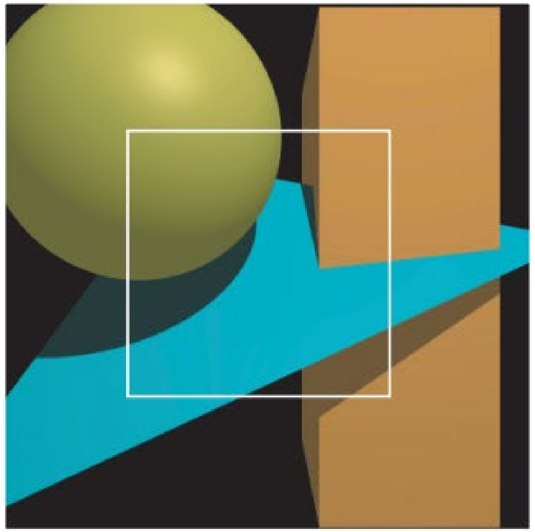
Perspective Effects

What camera action does varying d have?

- Increasing d (distance to view plane) decreases field of view
- Increasing d zooms into the scene



d= 200



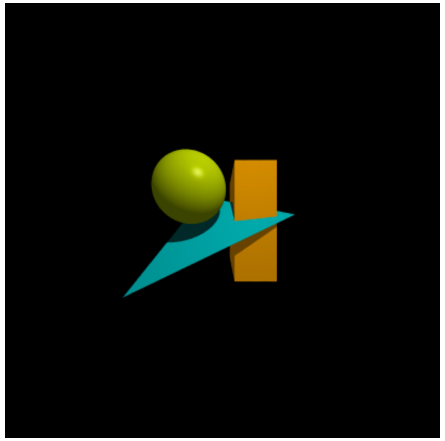
d=400

d=1000

Perspective Effects

What happens when we fix d and vary the eyepoint?

- No longer just a zoom…it also changes the projection
- Objects will be increasingly distorted as eye gets closer to objects in the scene



Orthographic Zoom Effect

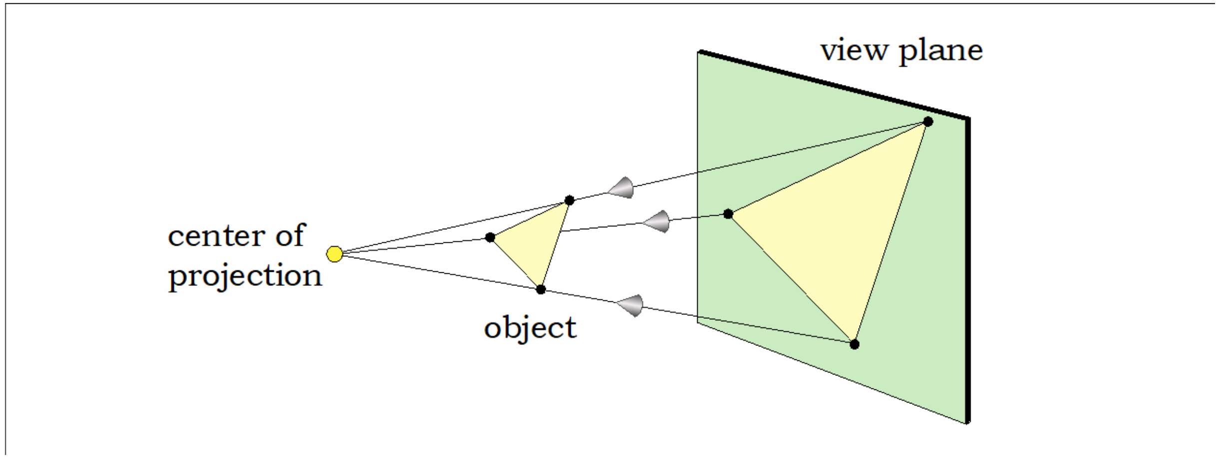
Is it possible to zoom when using an orthographic projection? Yes!

For a fixed resolution you can adjust zoom by changing the pixel size ***s***

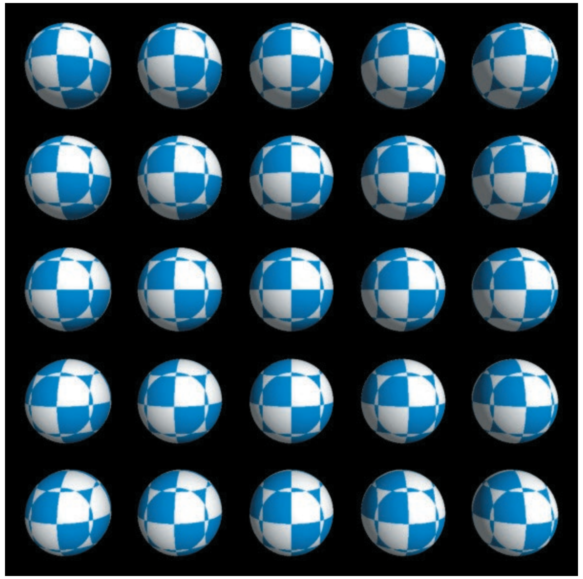
You could also keep ***s*** fixed and increase the resolution.

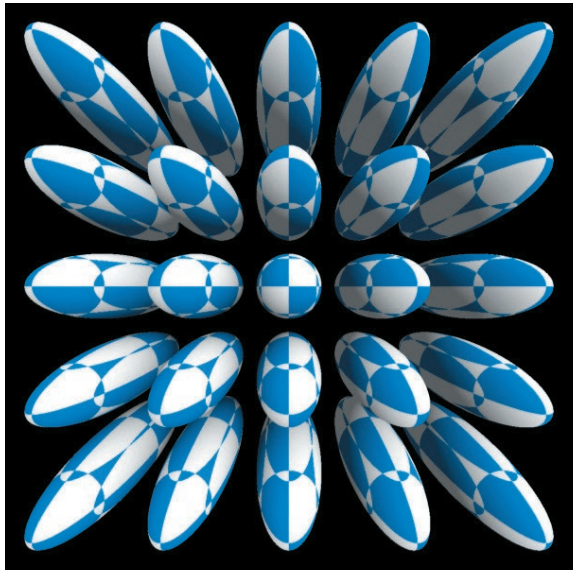
Perspective Effects

What if the object is between the viewplane and eyepoint?



Perspective Distortion

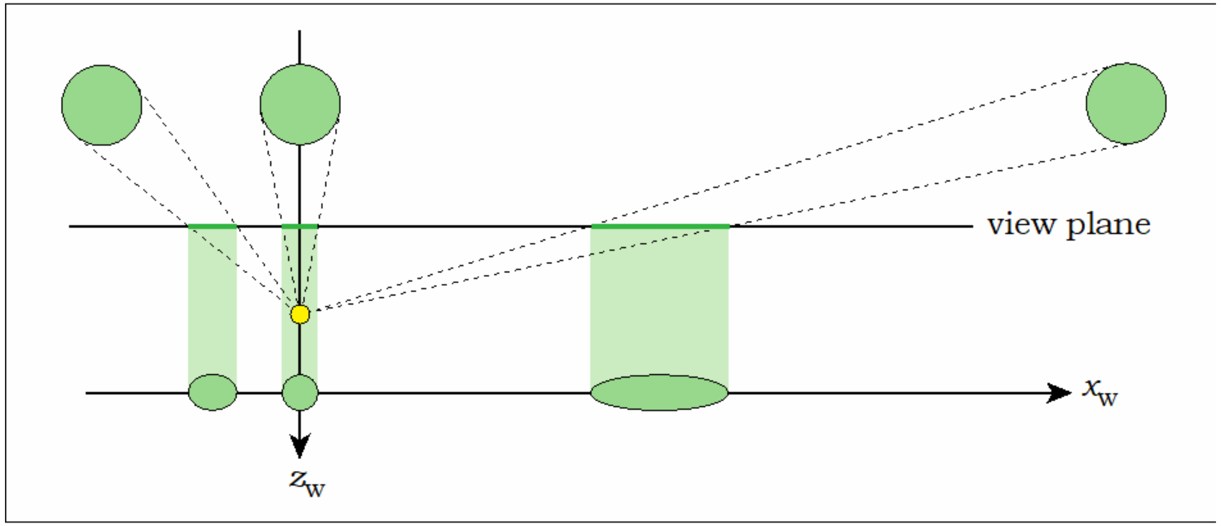




Not a bug – behaving as expected

As spheres get farther from the z-axis, more distortion

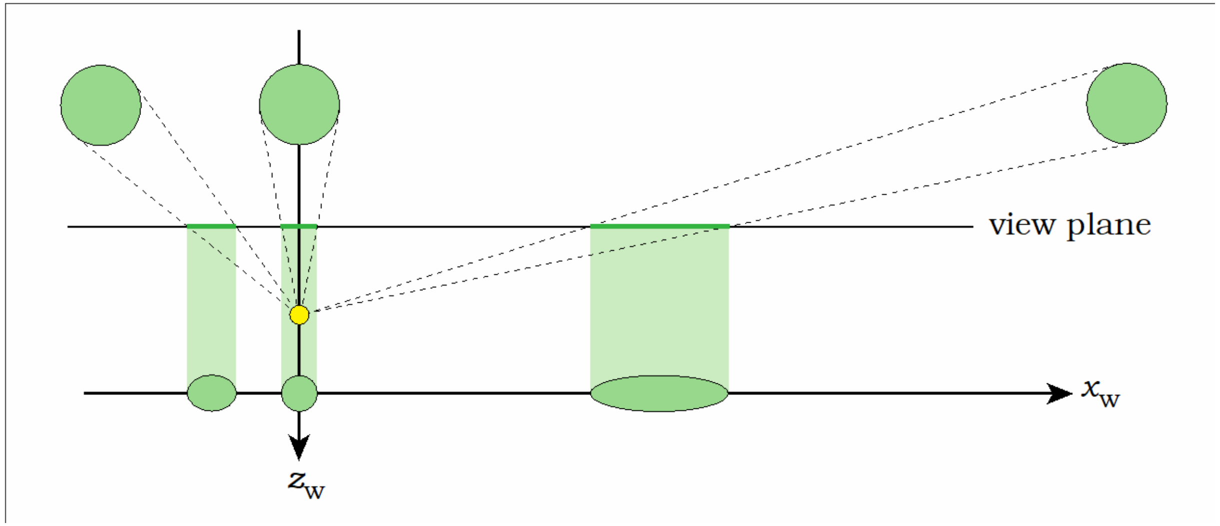
- CoP is on z-axis
- Z-axis is orthogonal to view plane



Distortion can be “fixed” by moving eyepoint back away from scene objects.

Perspective Distortion – Why Don’t We See It?

- Caused by the view plane being flat
- Our retinas are spherical
  - So we don’t see it

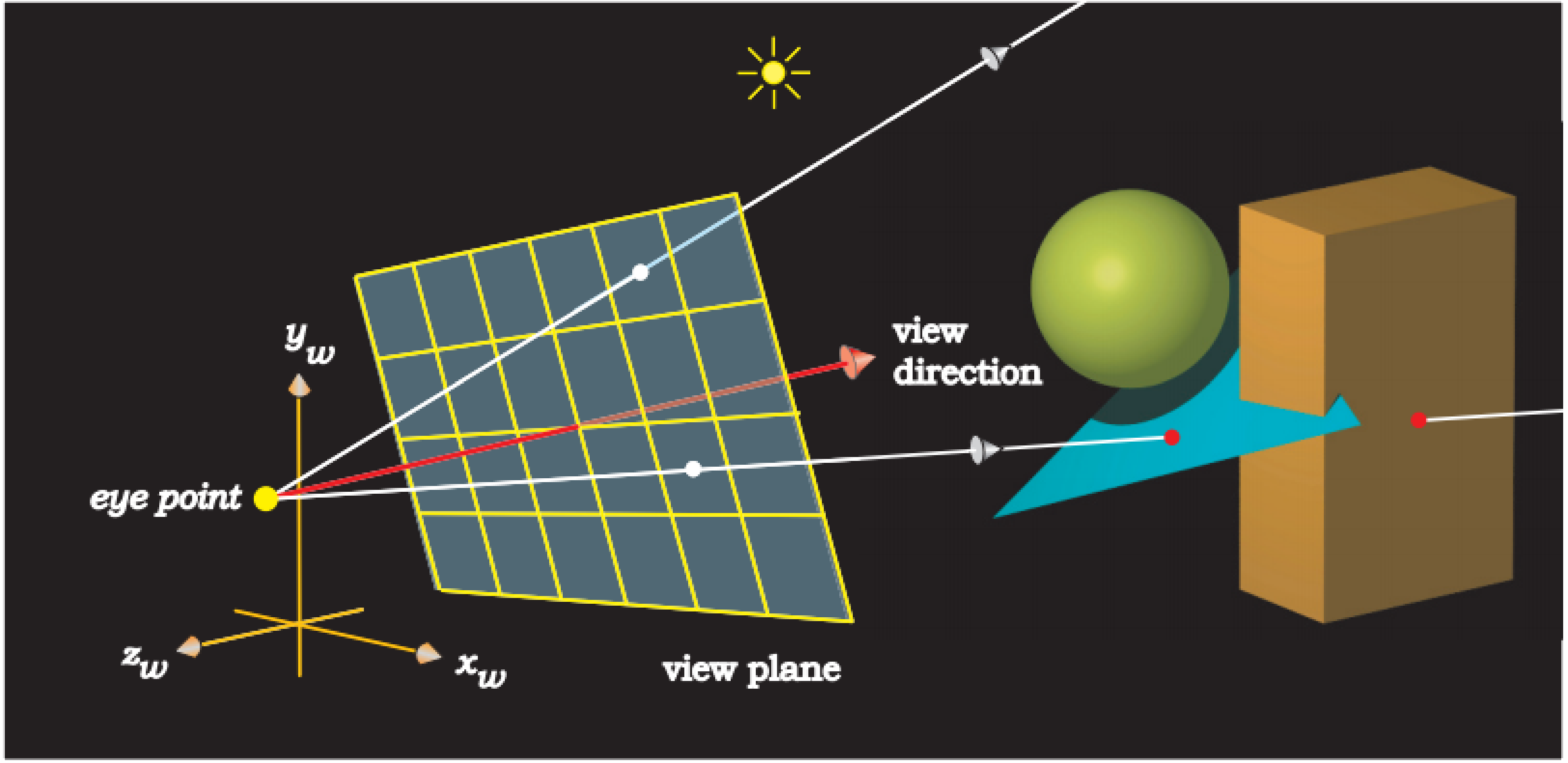


ILLINOIS

Movable Camera

We would like to add the ability to see a scene

- from any point
- in any direction

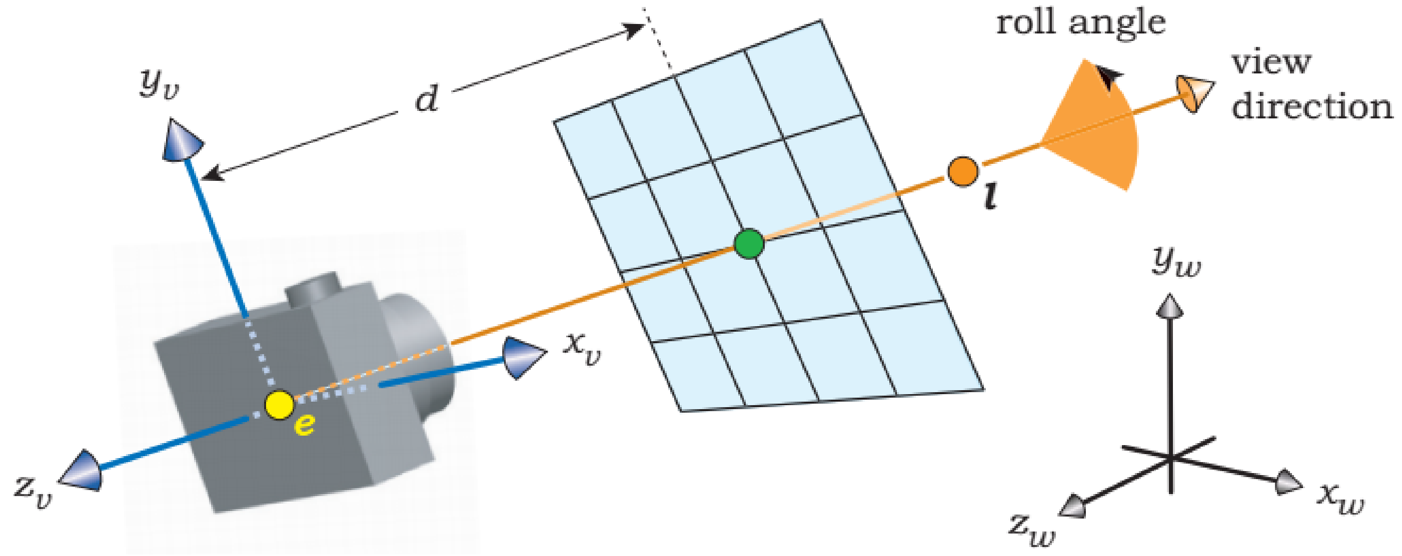


Turns out to be easy…just need to figure out how to generate the origin and direction of rays through the view plane…

Virtual Pinhole Camera

We have the following parameters

- Eyepoint – a (x,y,z) position in world space
- View direction – a (x,y,z) vector in world space
- Up – (x,y,z) world space vector describing the up direction of camera
  - roll around the view direction
- d …also need the distance to the view plane from eyepoint



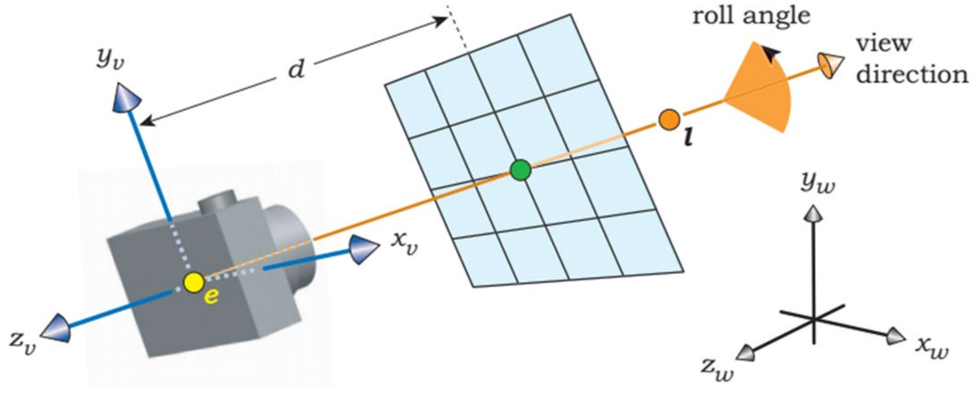
Axis-Aligned Perspective Viewing

- For perspective viewing, eyepoint e will be ray origin
  - Ray direction will be $p - e$ where p is a point in a pixel
- In world space with the eye at (0,0,0) looking down –z pixel centers are

$$p = \left(s\left(c - \frac{h_{res}}{2} + \frac{1}{2}\right), \mathrm{s}\left(r - \frac{v_{res}}{2} + \frac{1}{2}\right), -\mathrm{d}\right)$$

For arbitrary camera position and direction we can

- generate p in local camera space
- convert p to world space

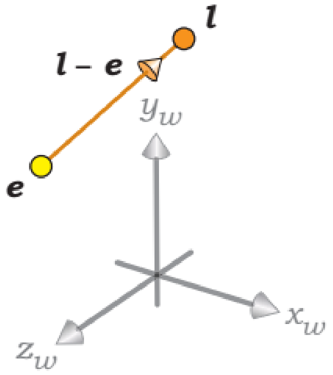


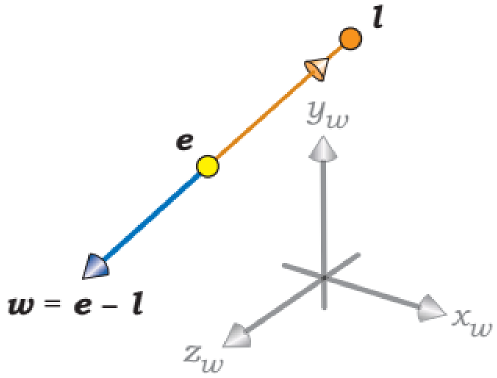
ILLINOIS

Constructing a Local Euclidean Frame for the Camera

Frame = an origin plus an orthonormal basis

- Camera frame origin = eyepoint
- Orthonormal basis (ONB) = ***u*, *v* ,*w***

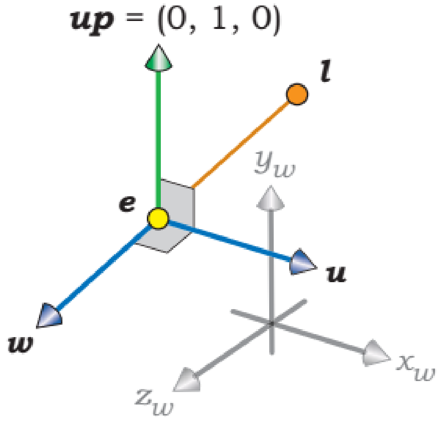


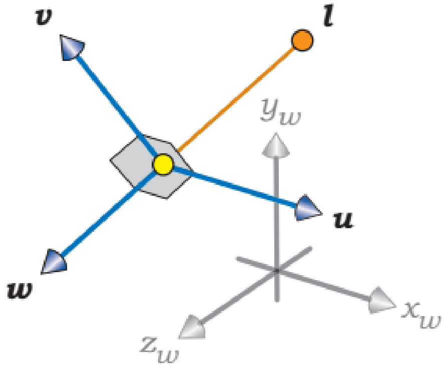


$$\boldsymbol{w} = \frac{(\boldsymbol{e}-\boldsymbol{l})}{\|\boldsymbol{e}-\boldsymbol{l}\|}$$

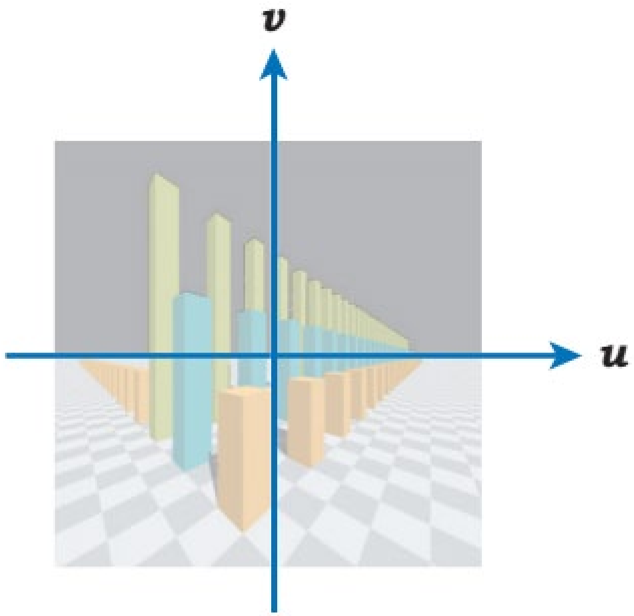
$$\boldsymbol{u} = \frac{(\boldsymbol{up} \times \boldsymbol{w})}{\|\boldsymbol{up} \times \boldsymbol{w}\|}$$

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





Primary Ray Calculation



Direction **u** is parallel to pixel rows

Direction **v** is parallel to pixel columns

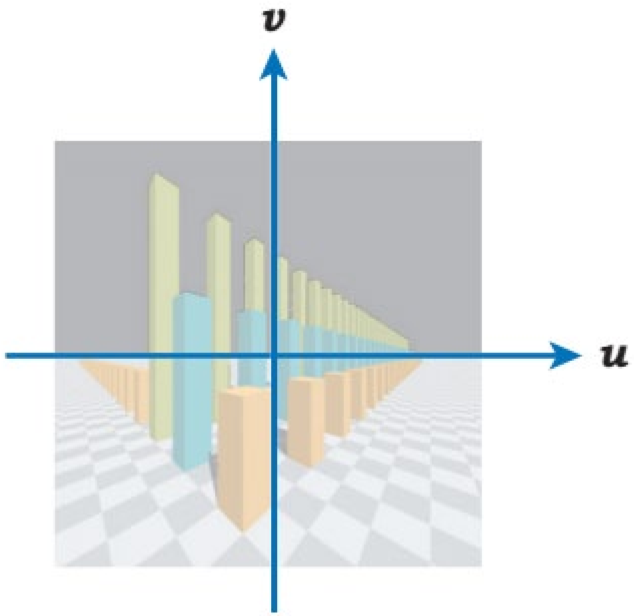
Compute point in pixel like before:

δ_x, δ_y is a random offset in the pixel square generated by the anti-aliasing method of your choice

$$p = \left(s\left(c - \frac{h_{res}}{2} + \delta_x\right), s\left(r - \frac{v_{res}}{2} + \delta_y\right), -\mathrm{d}\right)$$

Keep in mind…we are doing this in local camera space

Primary Ray Calculation



Direction **u** is parallel to pixel rows

Direction **v** is parallel to pixel columns

Compute point in pixel like before:

δ_x, δ_y is a random offset in the pixel square generated by the anti-aliasing method of your choice

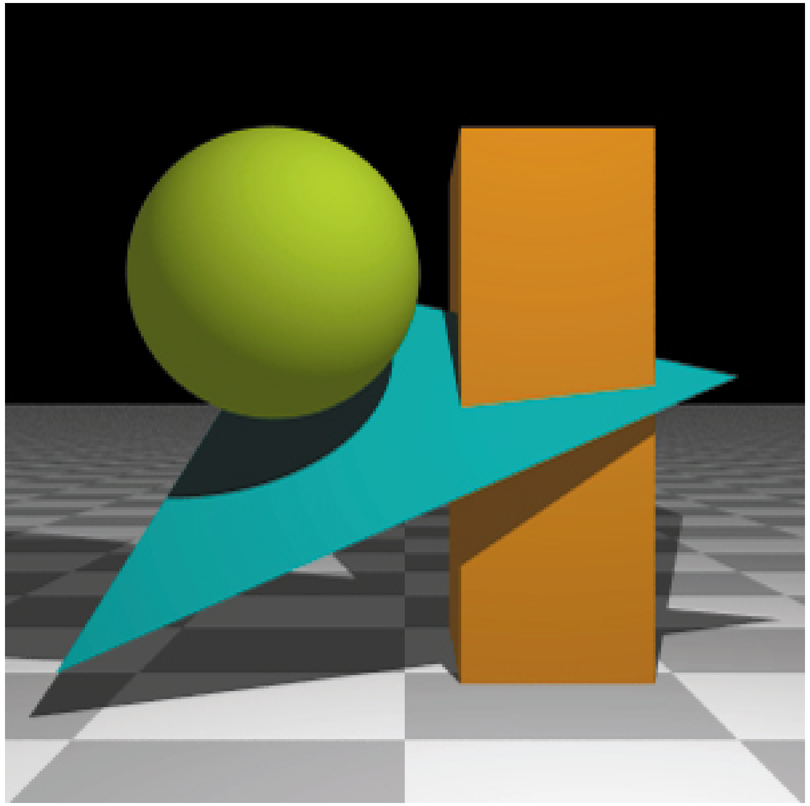
$$p_v = (x_v, y_v, z_v)$$

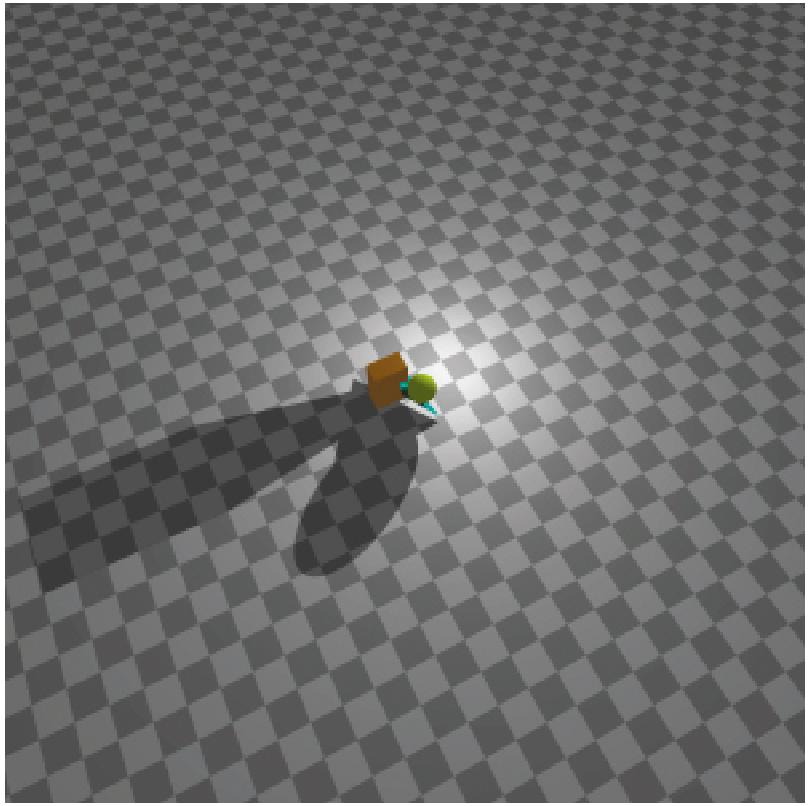
$$p_v = \left(s\left(c - \frac{h_{res}}{2} + \delta_x\right), s\left(r - \frac{v_{res}}{2} + \delta_y\right), -\mathrm{d}\right)$$

Keep in mind…we are doing this in local camera space

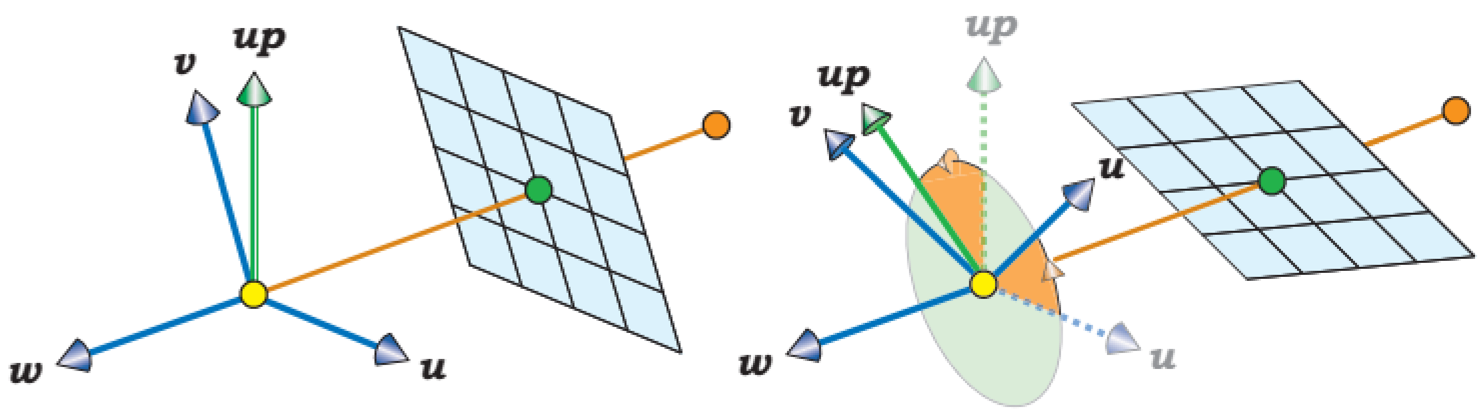
Convert to world space by $p_w = ux_v + vy_v + wz_v$

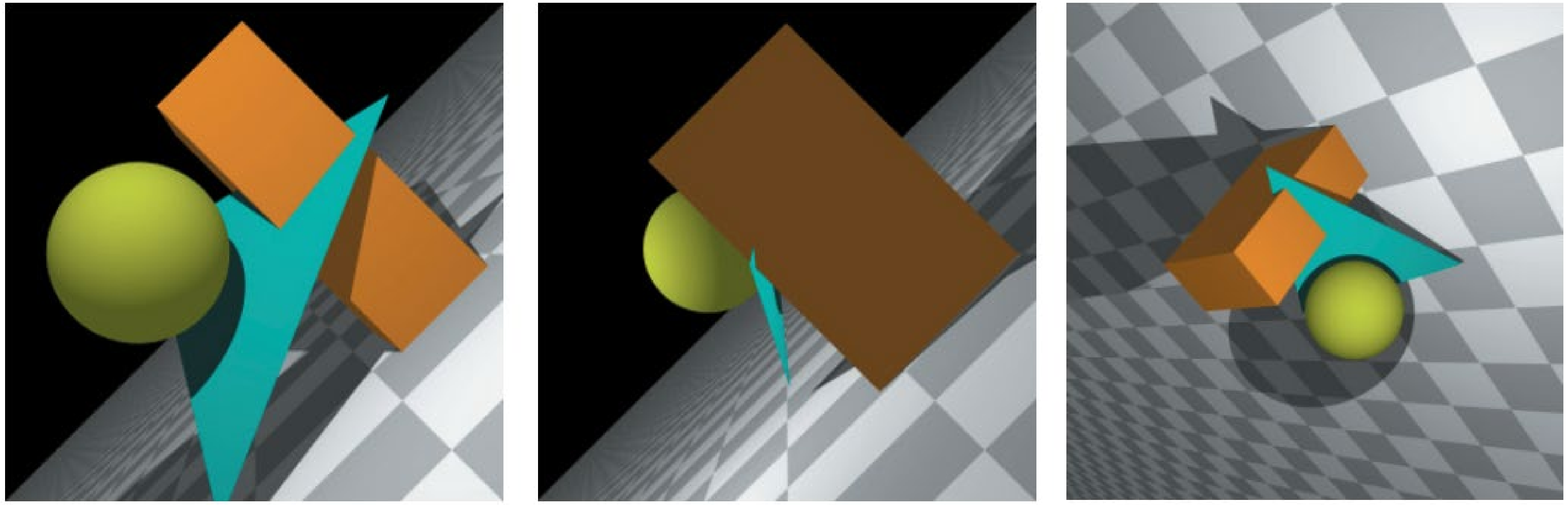
Example





Changing Up

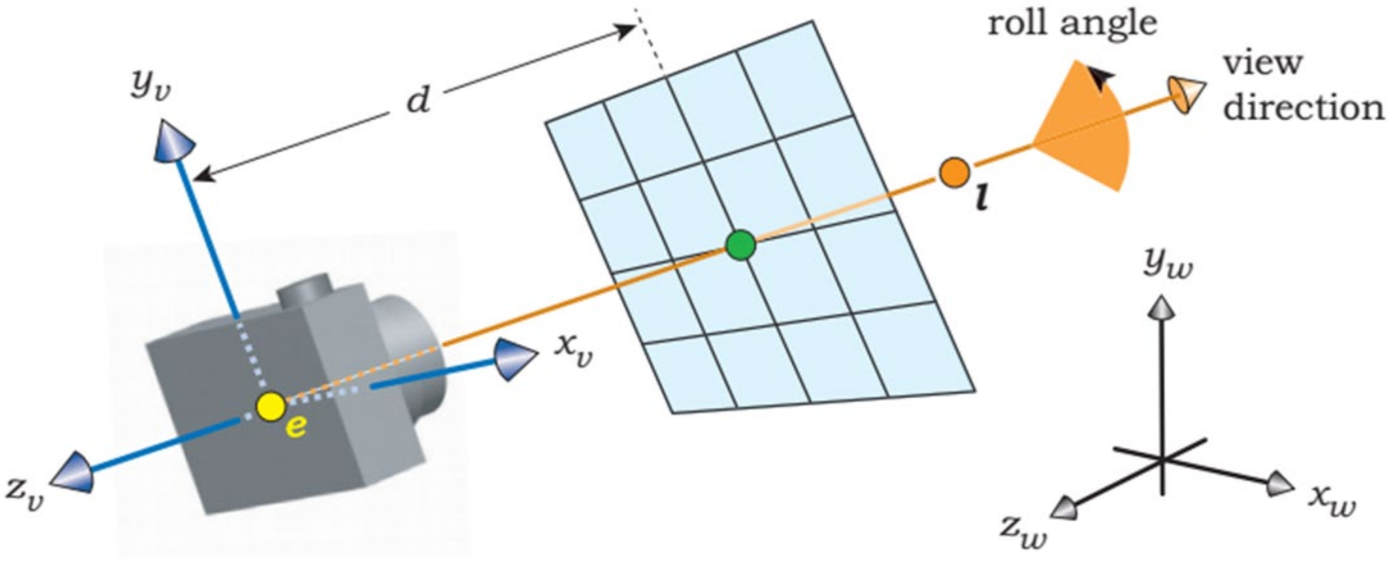




Singularity

For any viewing system, there are parameter values that can break the system

- When does this happen for the parameters we have described?
  - Up and the view direction cannot be parallel



ILLINOIS