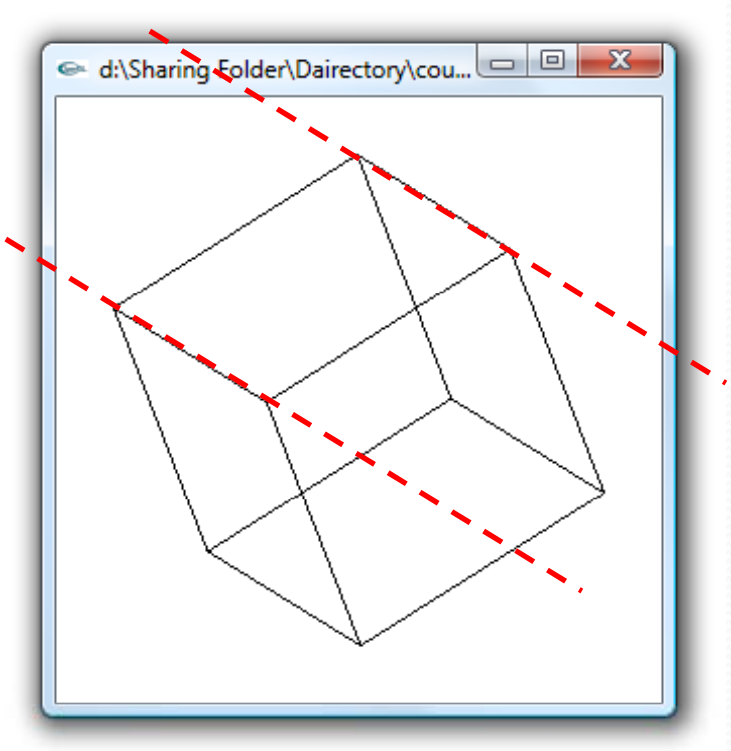
Viewing

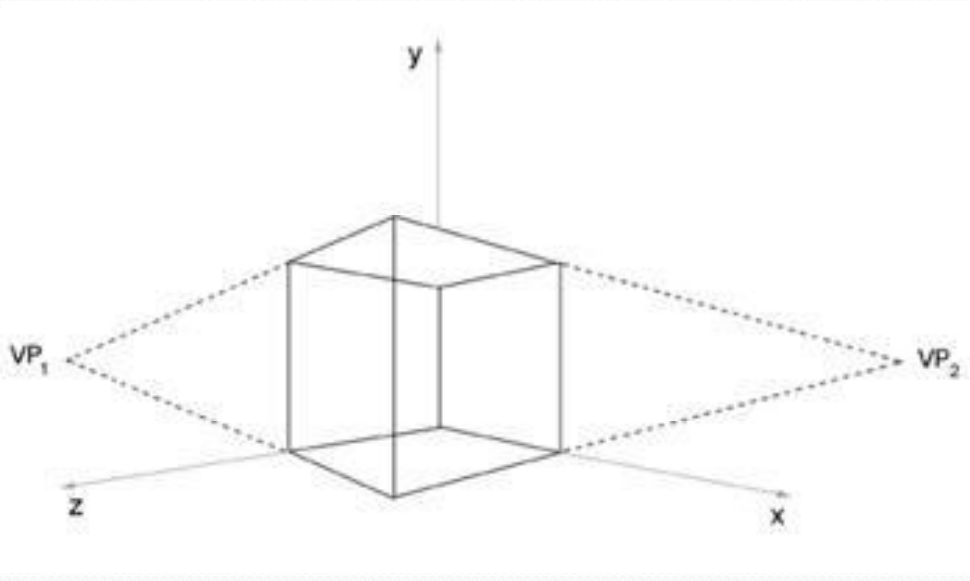
"Let's put things into prospective"

Something is not very right?!

- Parallel Projection

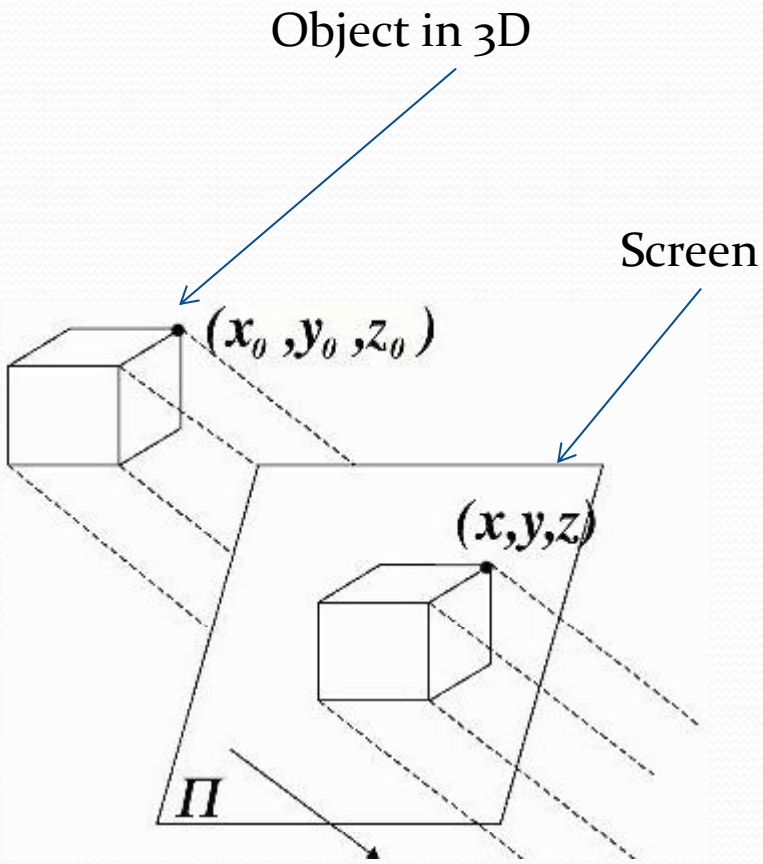


- Perspective Projection

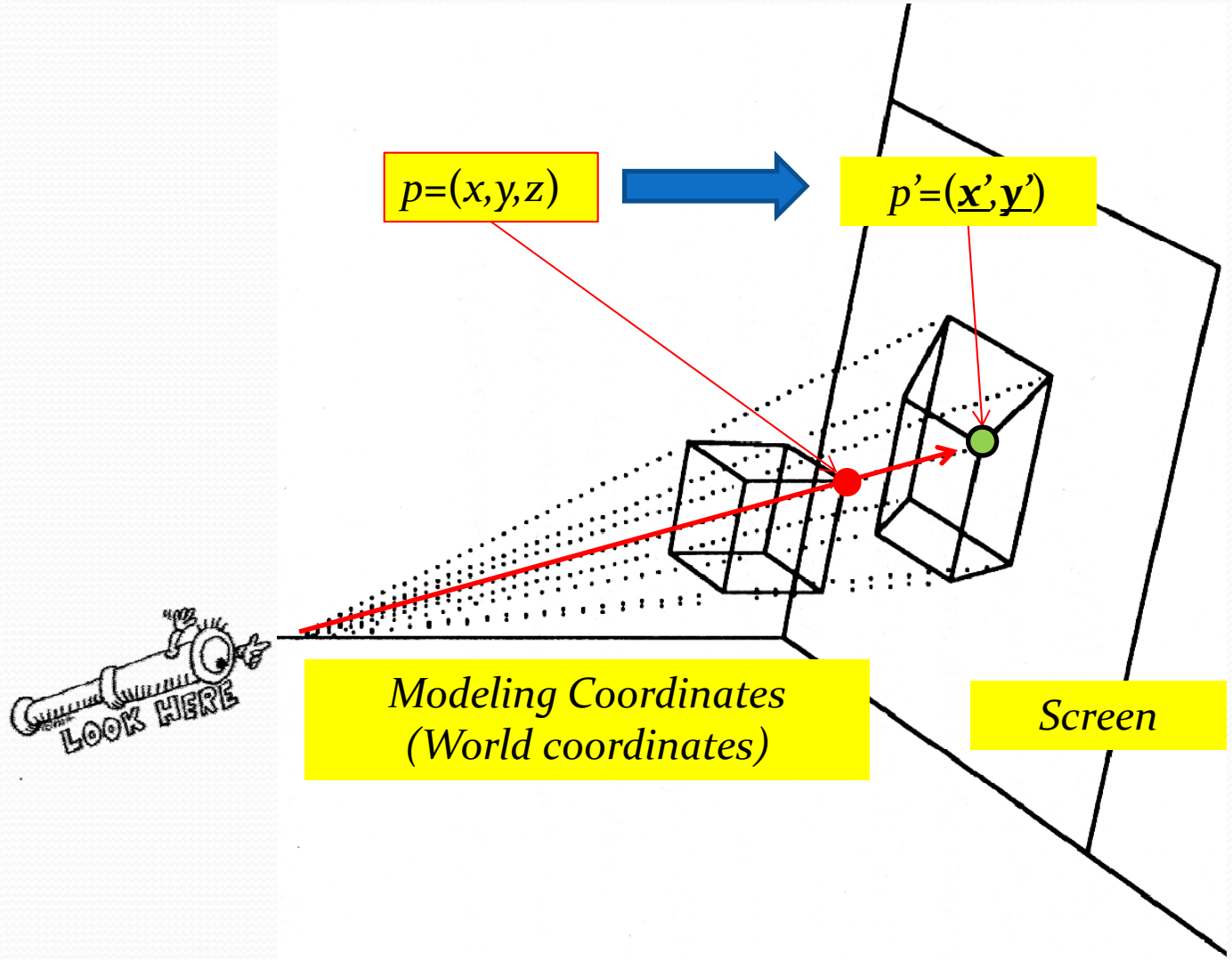


Current Stage

- Objects are transformed in 3D
  - As in 2D, after the reference frame is changed
- Although objects are in 3D, OpenGL just take the first (x,y) coordinates to draw on the screen
  - E.g. in the diagram, the position drawn on the screen is : $(x,y) = (x_0, y_0)$

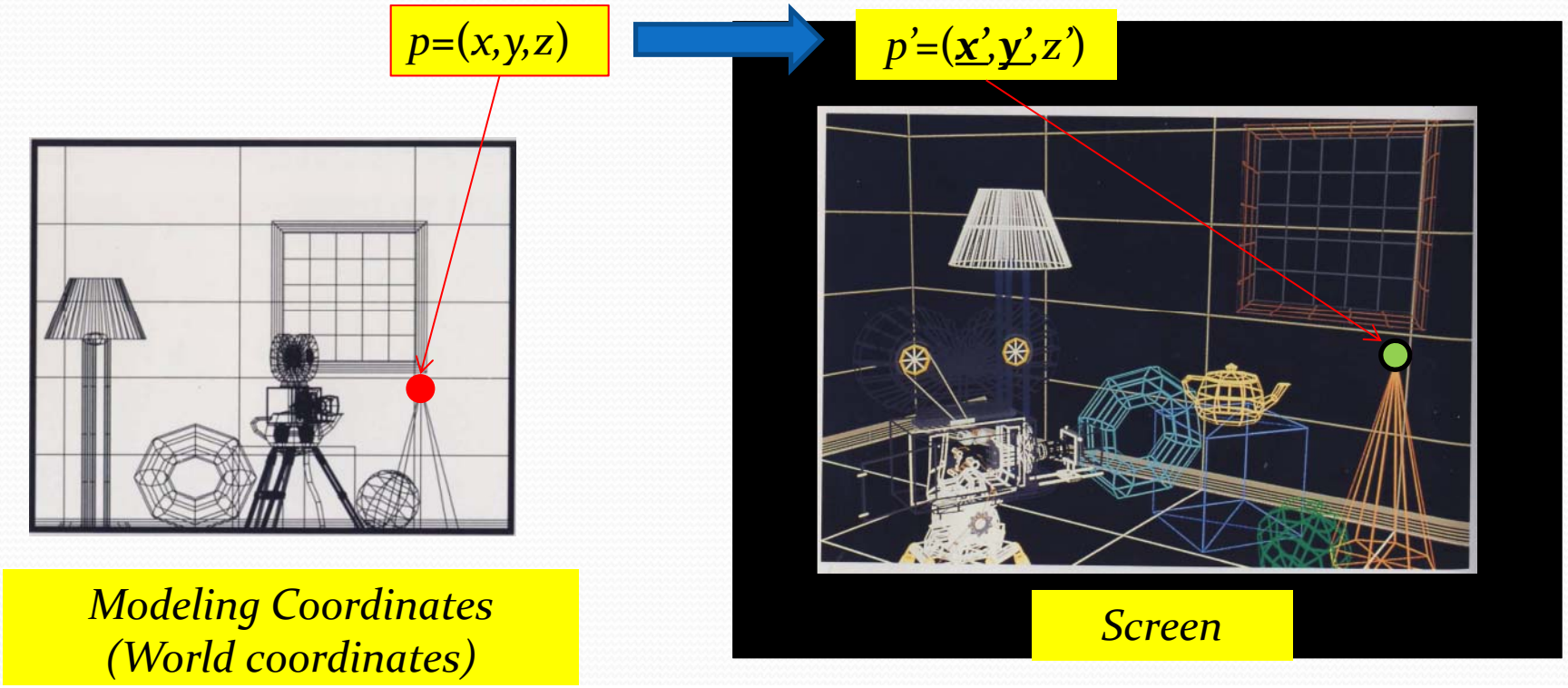


How to draw a cube?

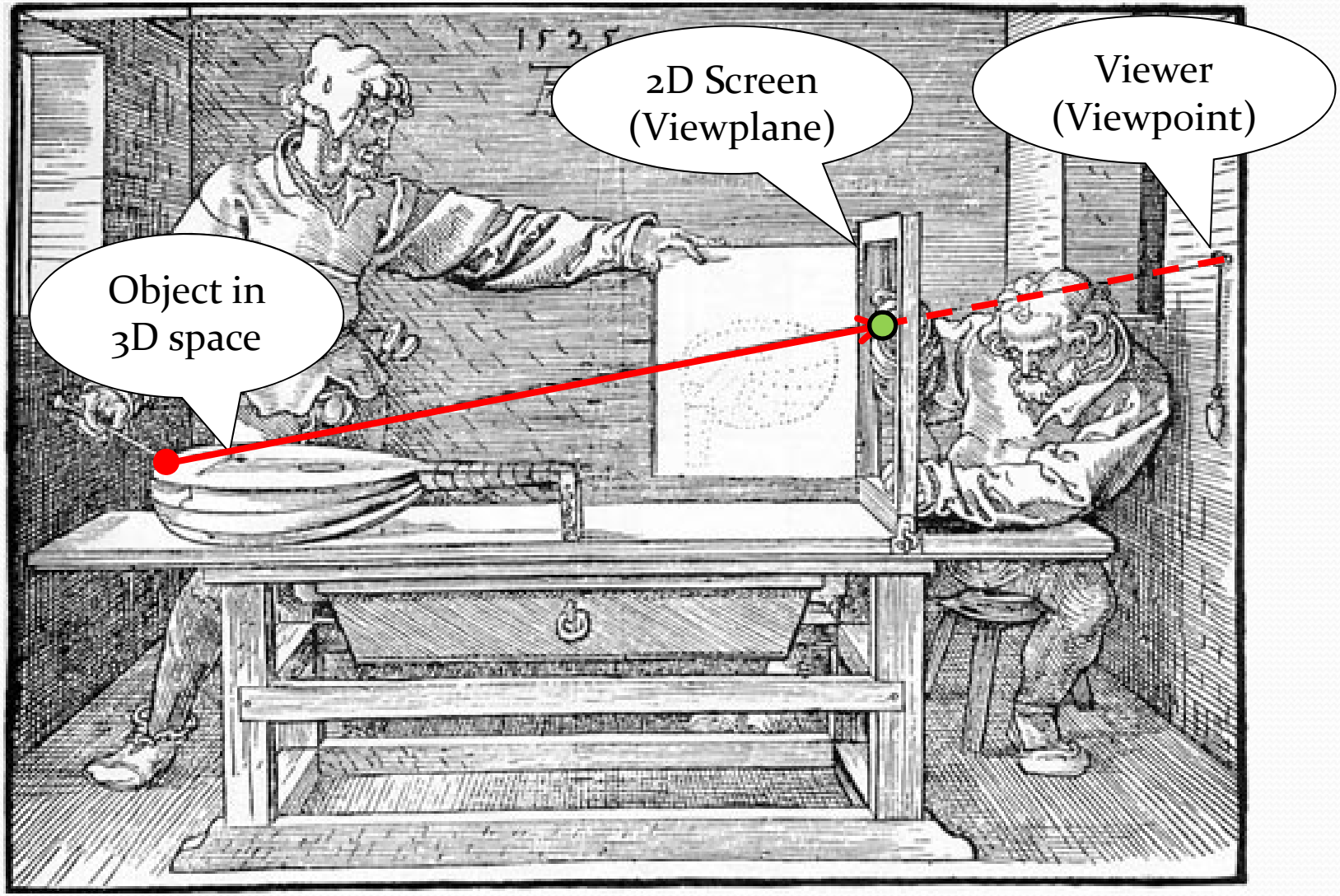


Transformation/Viewing

- After we specified the vertices positions (x,y,z) of the objects in the world coordinates
- Compute its position on the screen coordinates (x',y')

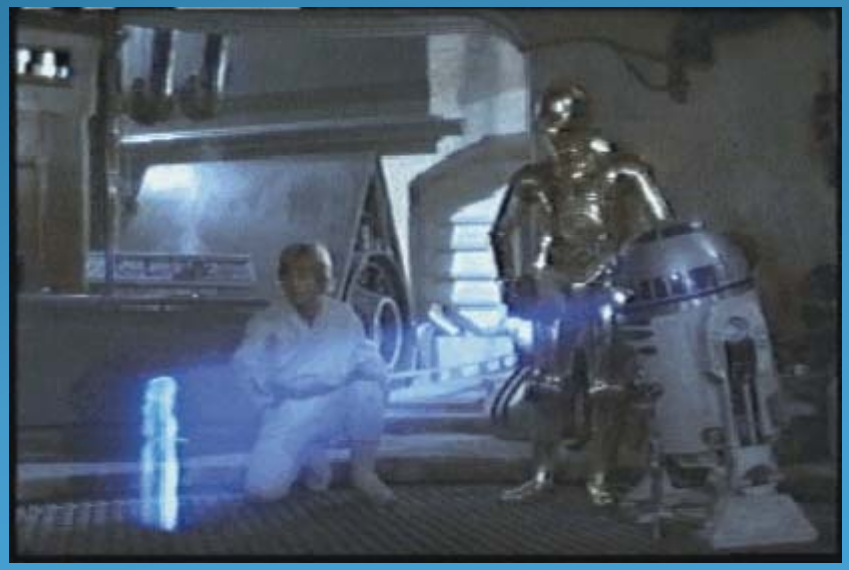


Perspective Projection



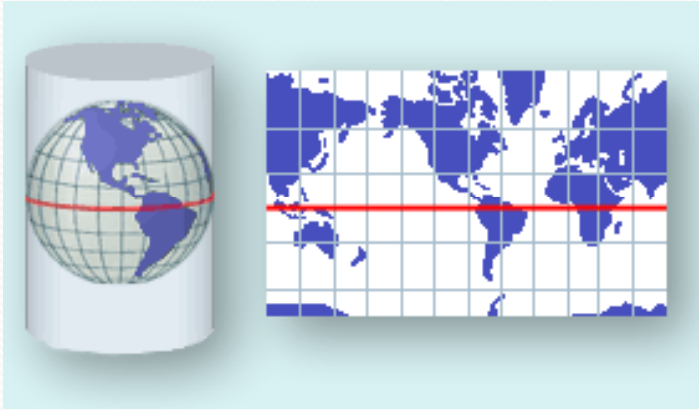
Two Types of Projections

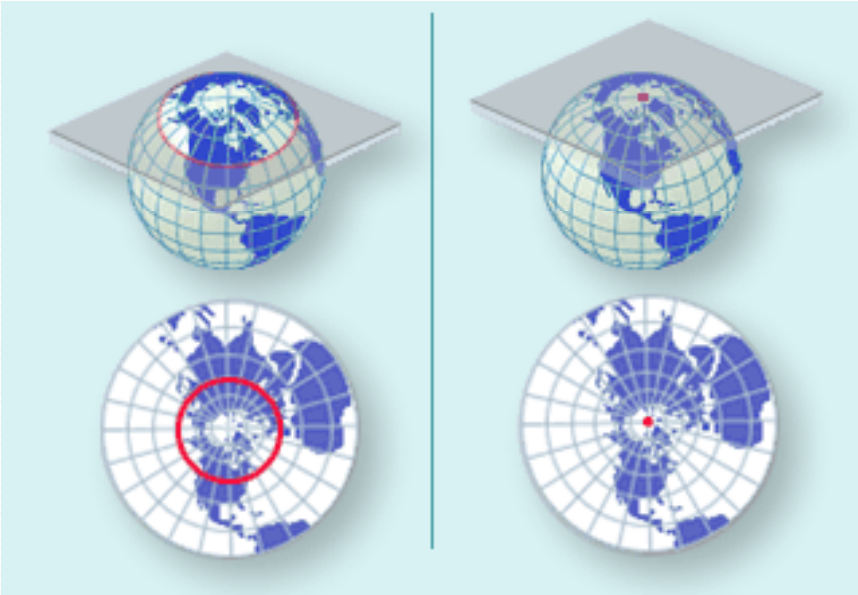
Orthographic vs Perspective

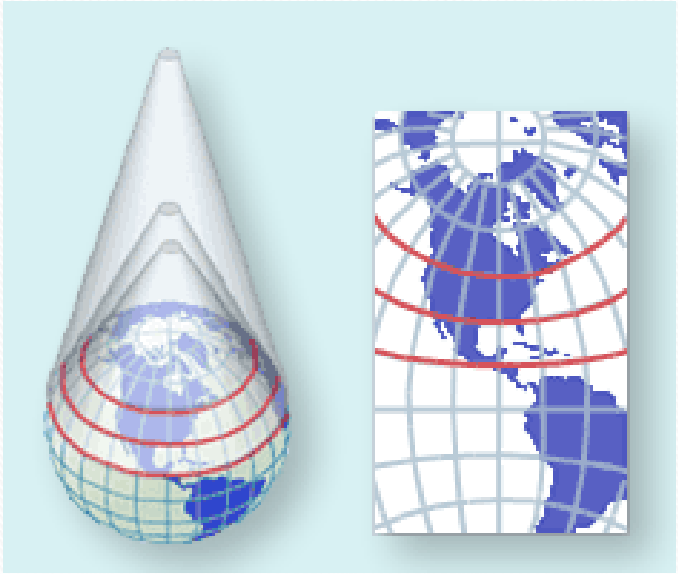


Map Projections

- Mainly 3 basic types
  - Cylindrical
  - Conic
  - Azimuthal

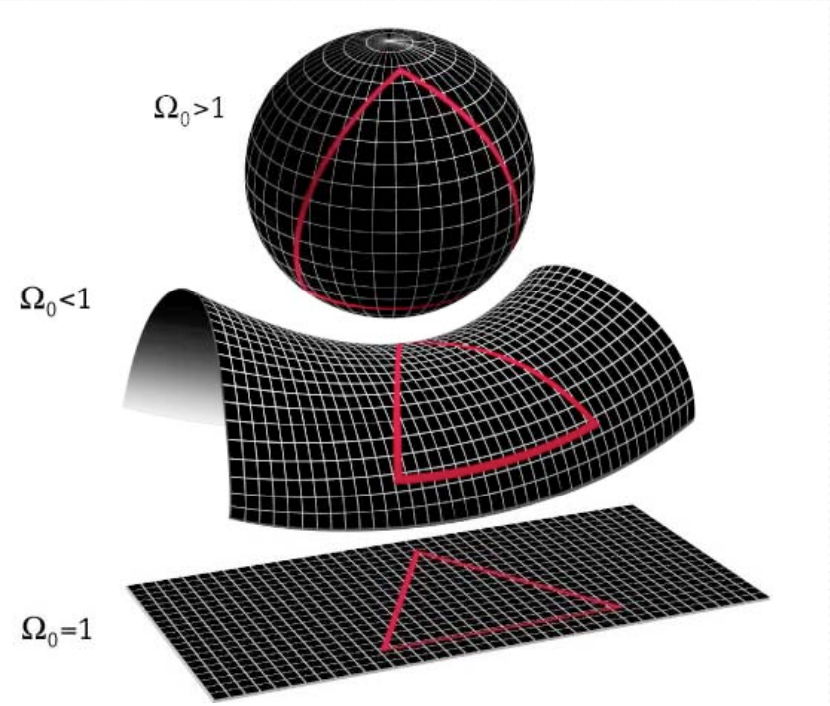






Definition of Projection

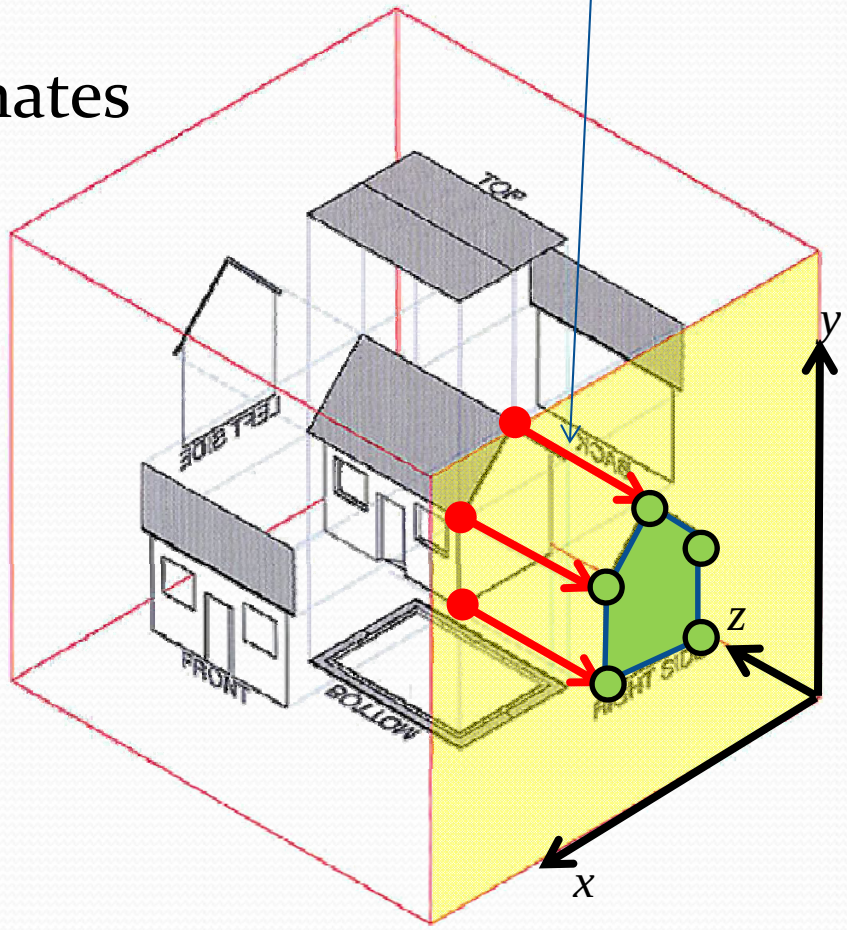
- The transformation of an “image” (points, lines, etc) from one ***space*** to another
  - The “spaces” could be anything other than “planes”
- However, we focus on projecting the **3D space** onto a **2D plane**



Orthographic Projection

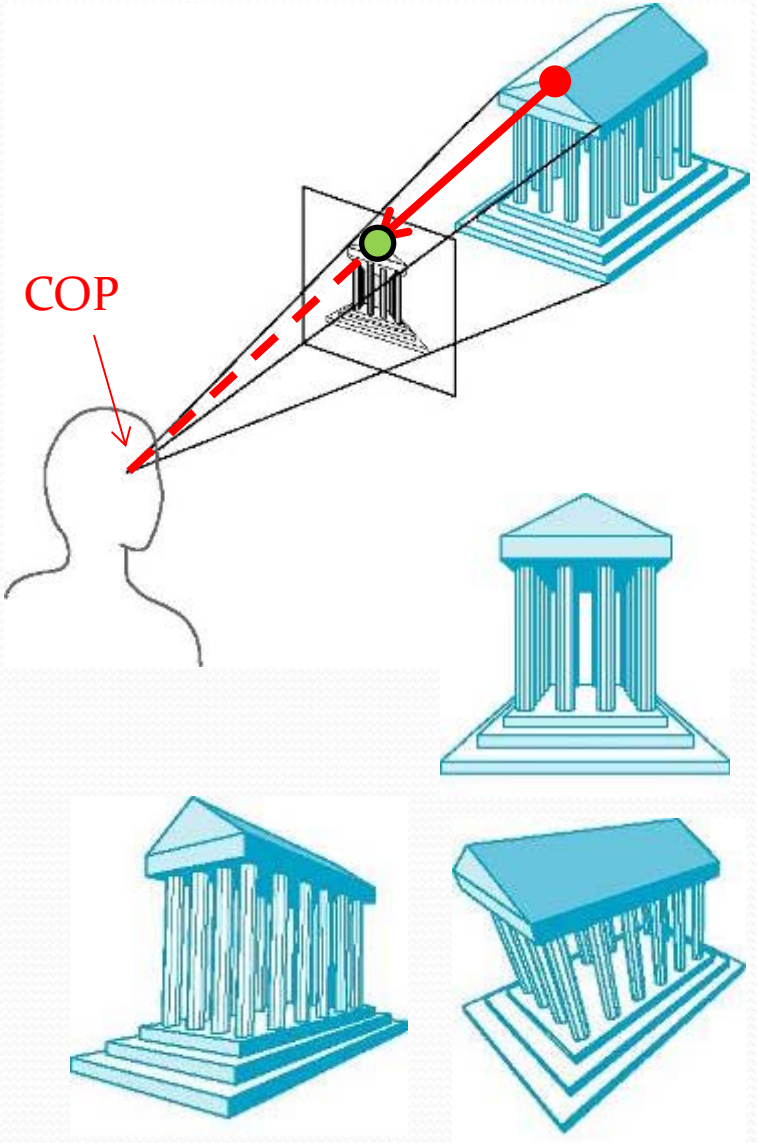
- A ***parallel*** projection
- Just ignore one of the coordinates
  - E.g. taking *x* and *y* only
- Good for
  - Graphic designs
  - Construction drawings
- However, not "realistic"
  - Not what human eyes see

These arrows that bring the points from 3D to 2D are called "*projectors*".

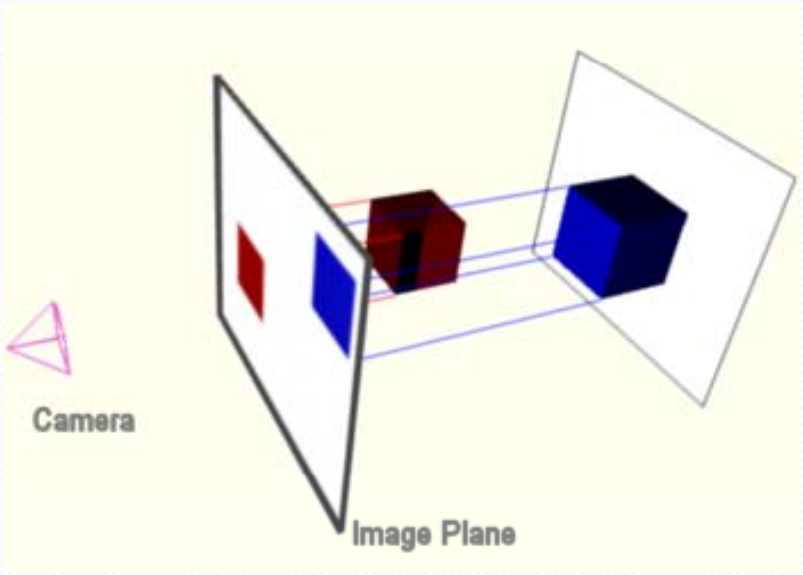


Perspective Projection

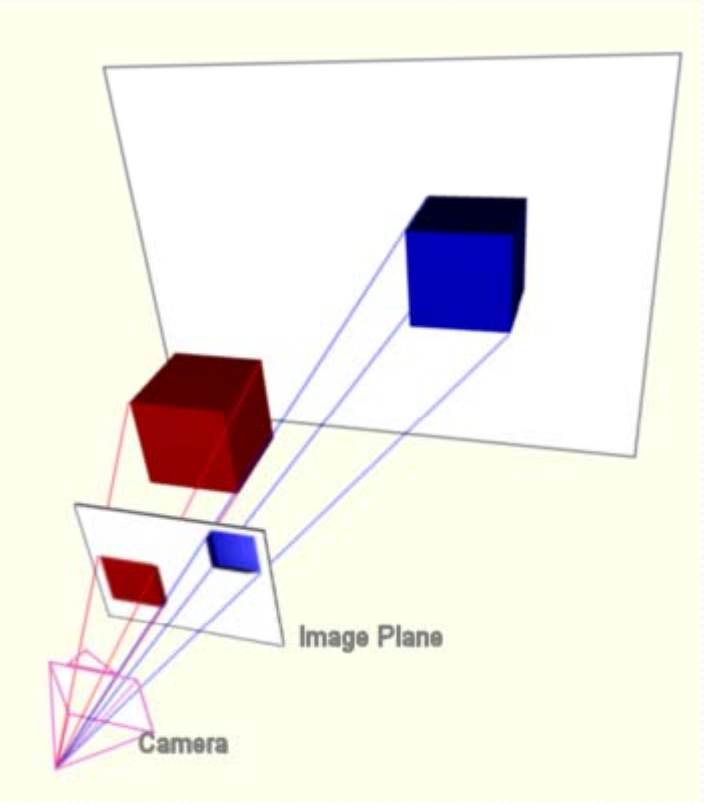
- Projectors converge at center of projection
- Further objects are smaller, nearer objects are bigger
  - This looks more realistic, our eyes (and most cameras) work in a similar fashion.
- Equal distances along a line are not projected into equal distances (foreshortening)
- Angles preserved only in planes parallel to the projection plane



Two Types of Projection

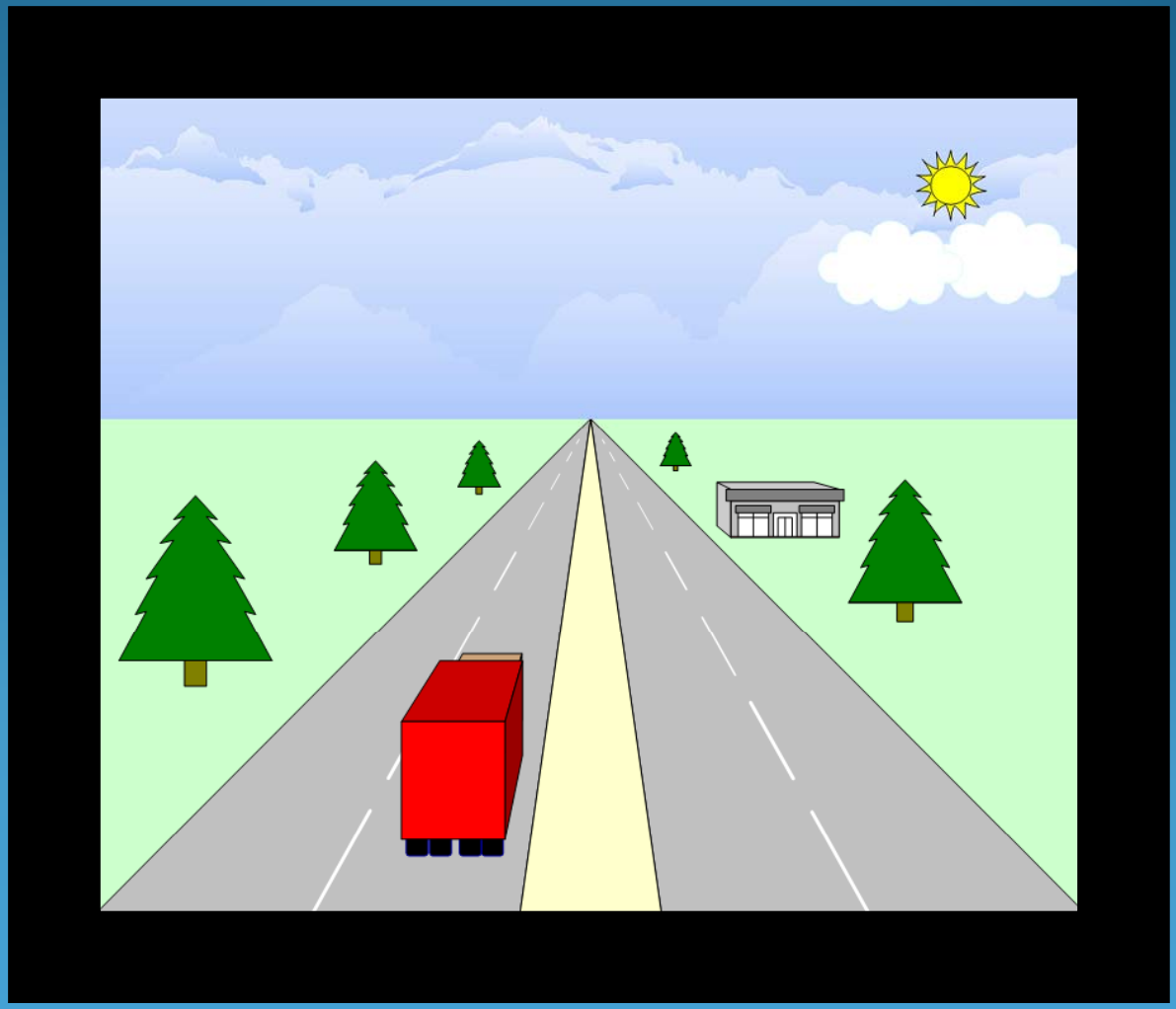


Parallel Projection

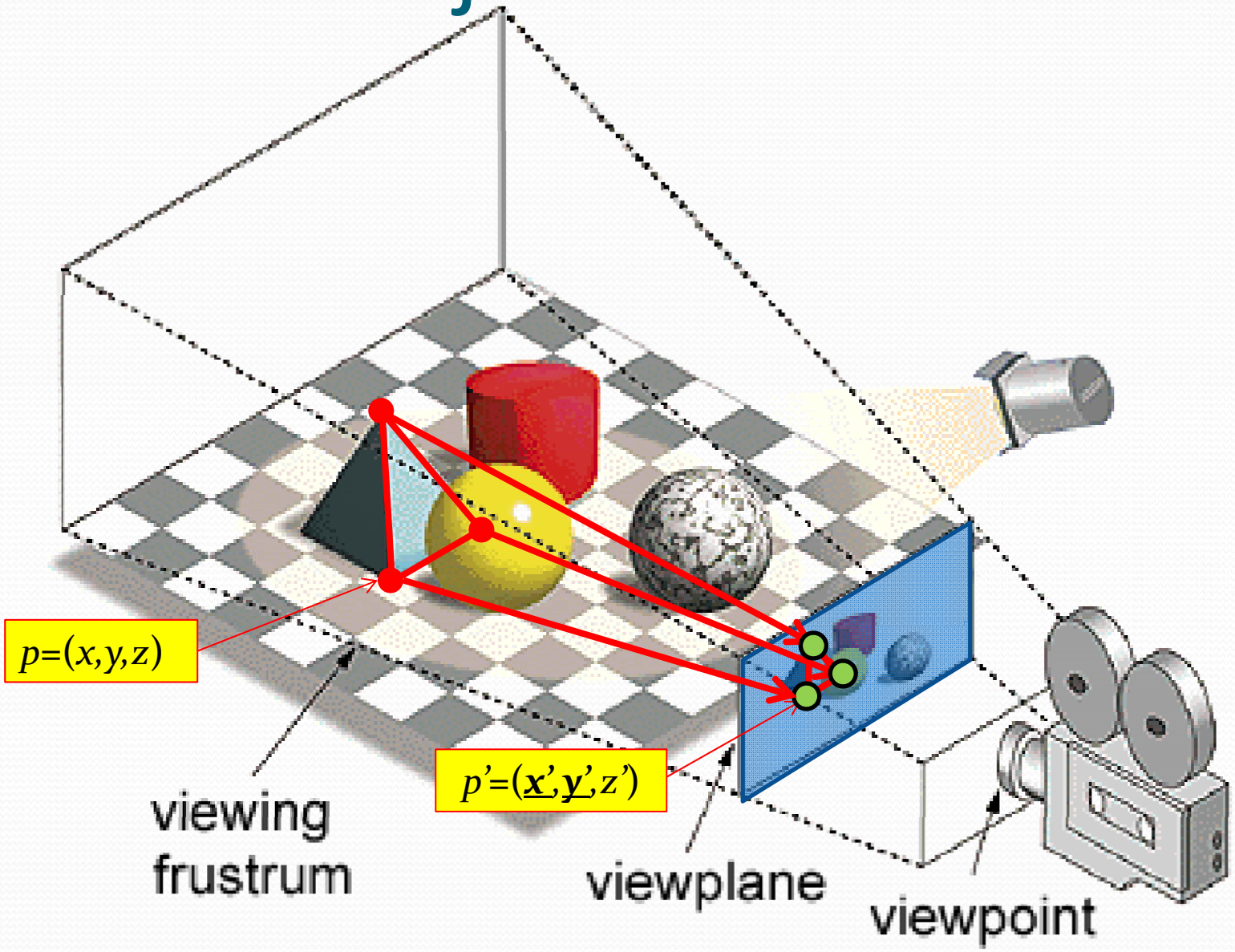


Perspective Projection

Perspective Projection

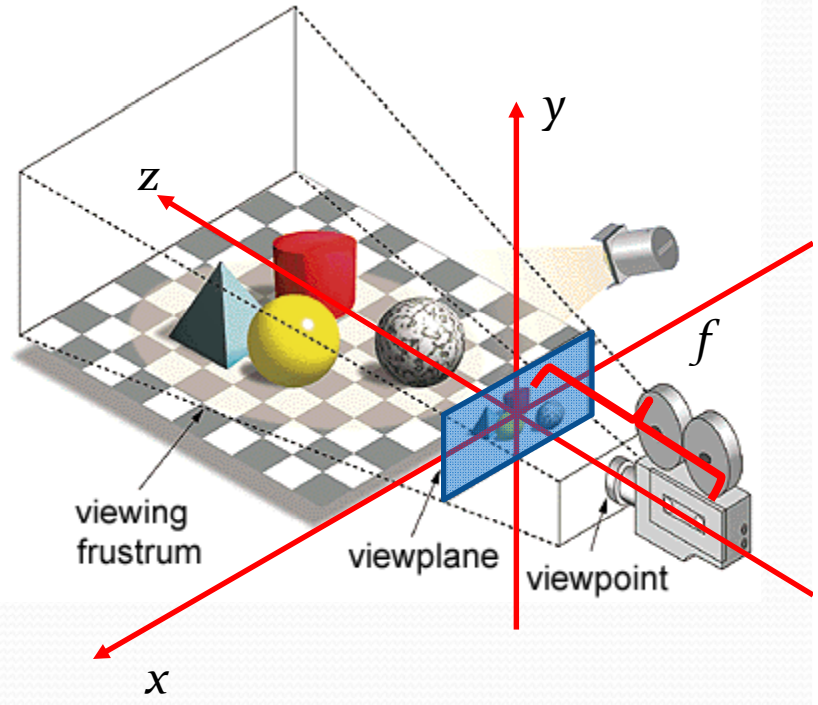


Perspective Projection

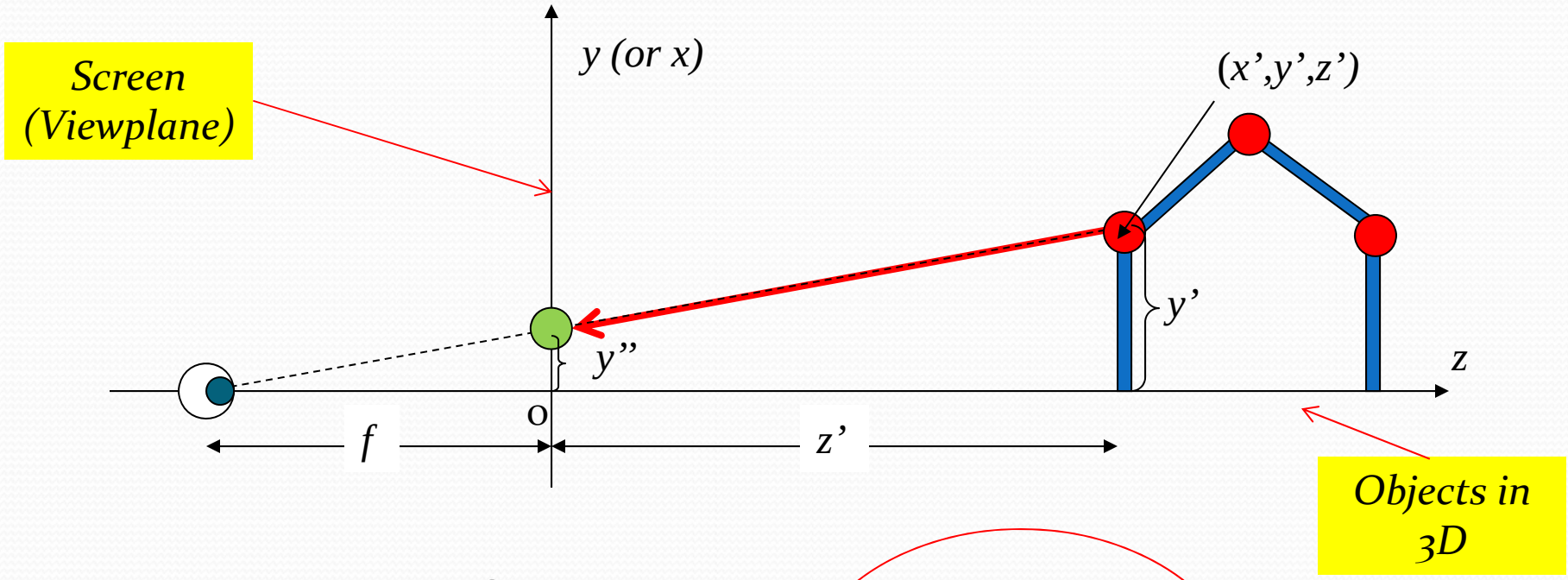


Setting Up the "Stage"

- Assuming the viewpoint has
  - Position: (0,0,-*f*) for *f* > 0
  - It's "up vector" is (0,1,0)
    - Facing upwards as the *y*-axis
  - "Looking direction" vector is (0,0,1)
    - Towards the positive direction of the *z*-axis
- The viewplane is the *xy*-plane



Perspective Transformation



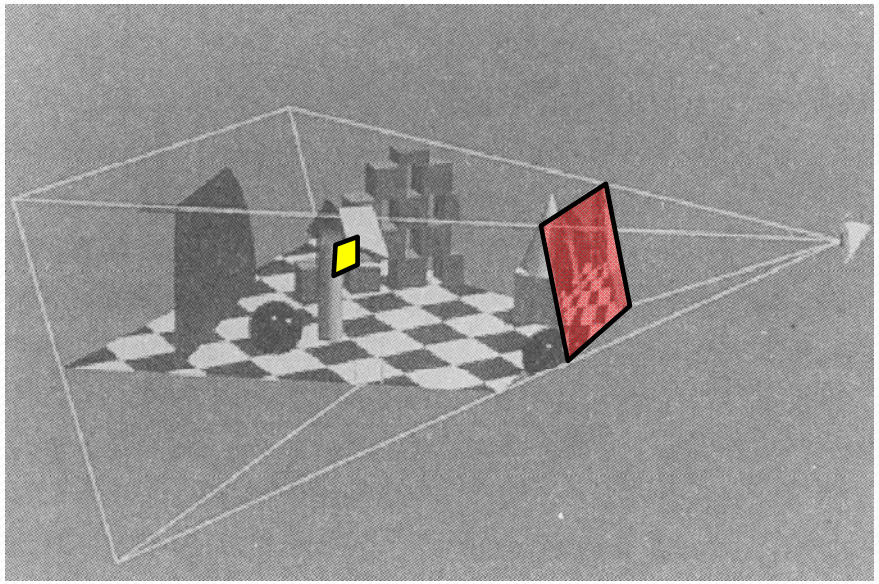
$$\frac{y''}{y'} = \frac{f}{f + z'} \quad \Leftrightarrow \quad y'' = \frac{y' f}{f + z'}$$

Perspective Transformation Matrix

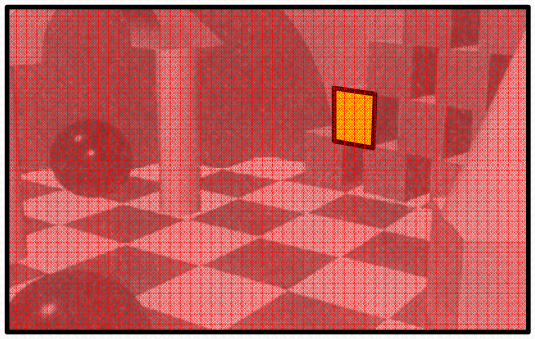
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ \frac{z'+f}{f} \end{pmatrix} \Rightarrow \begin{pmatrix} \frac{x'f}{z'+f} \\ \frac{y'f}{z'+f} \\ \frac{z'f}{z'+f} \end{pmatrix}$$

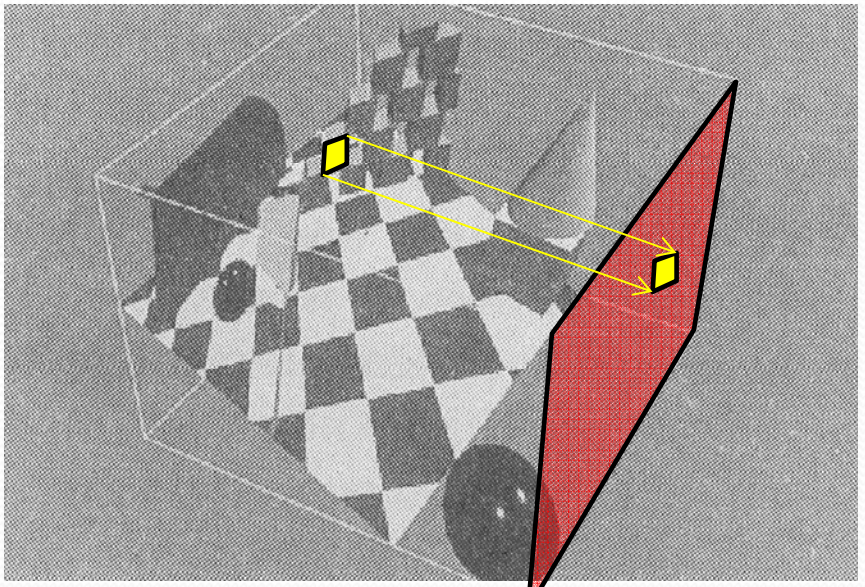
Perspective Projection Matrix: P_f

- Note that after the projection, objects are still “3D”
  - z-values are NOT zero
  - Distorted, e.g. a regular cube has no more right angles



Before Perspective Transformation



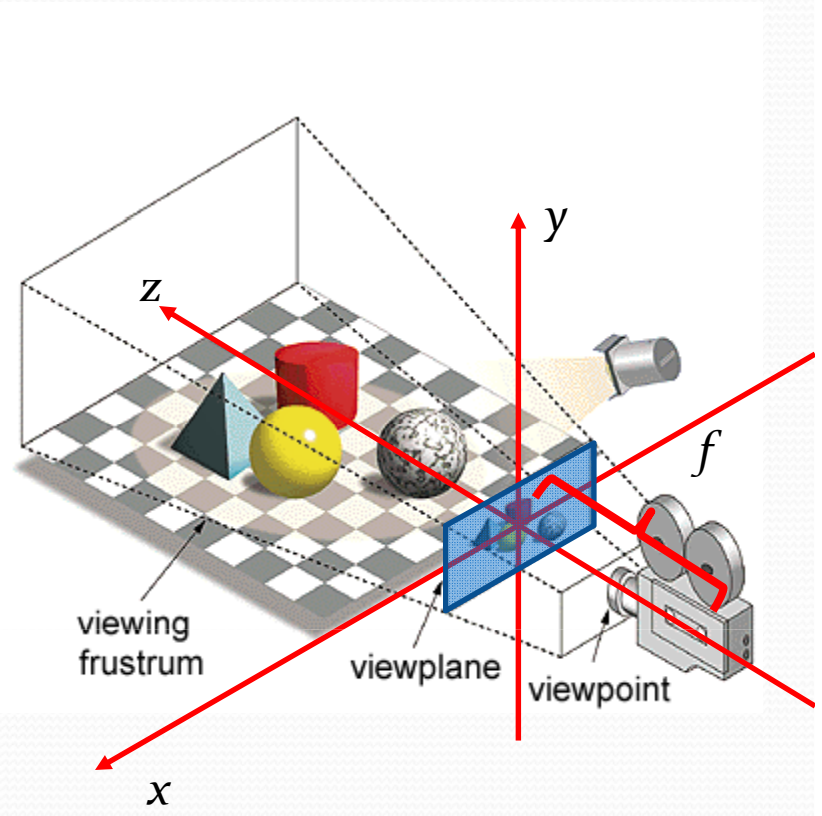


After Perspective Transformation

Summary

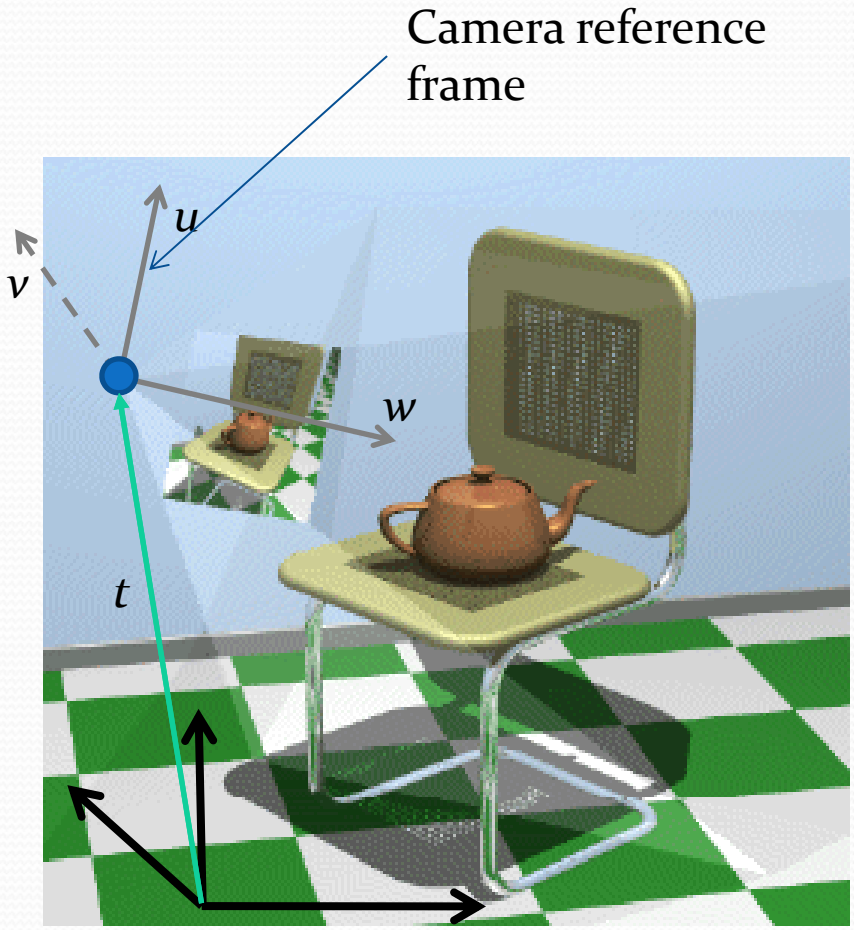
- The view point
  - Position: (0,0,-*f*) for *f* > o
  - It's "up vector" is (0,1,0)
    - Facing upwards as the *y*-axis
  - "Looking at" vector is (0,0,1)
    - Towards the positive direction of the *z*-axis
- The view plane is the *xy*-plane
- Then the projection matrix is

$$P_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 1 \end{pmatrix}$$

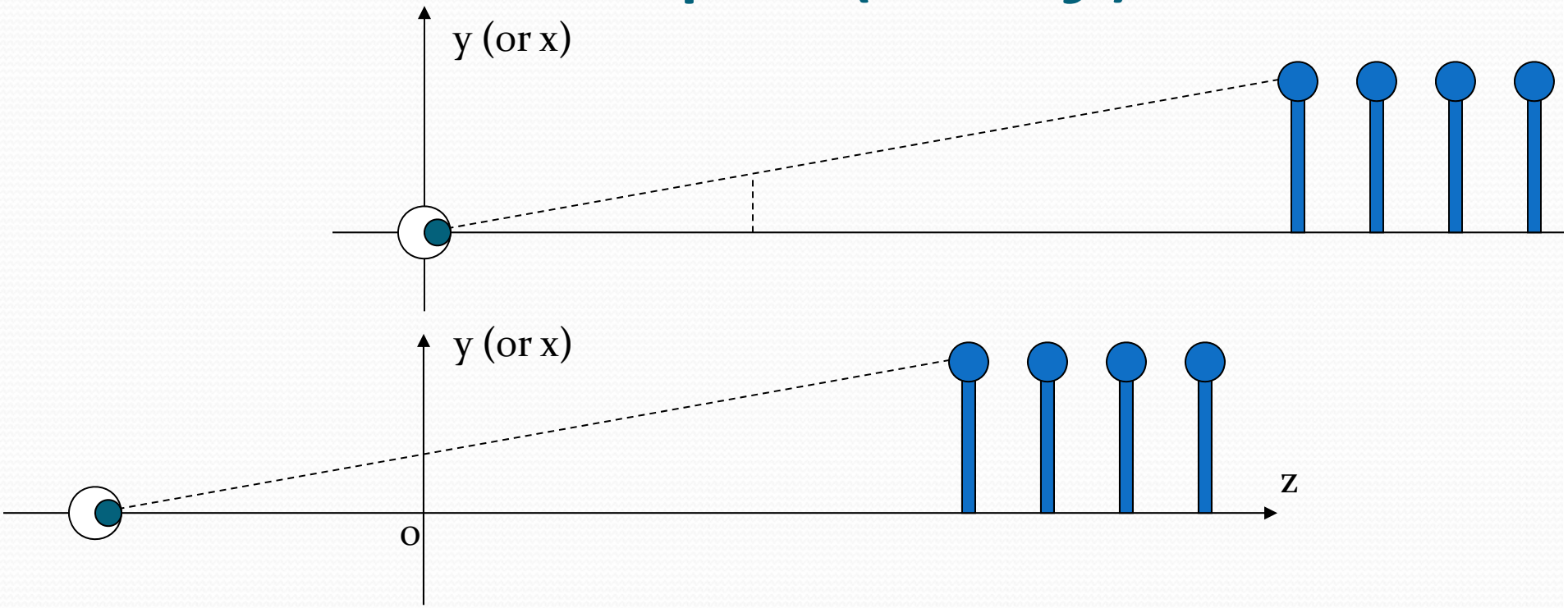


However, the Camera can be at ANYWHERE, namely, not at (0,0,-f)

- Let the camera be
  - At position: t
  - Look-at direction: w
  - An UP vector: u
  - "left" vector: $v = u \times w$
  - Assuming w, u and v are unit vectors
- Steps
  - Nothing new: move your reference frame to the "camera reference frame"
  - Let call this transformation M_c



One More Step: $T(0,0,-f)$



- All in all $$p' = P_f T(0,0,-f) M_c p$$
  - And we draw the x and y coordinates of p' on the screen

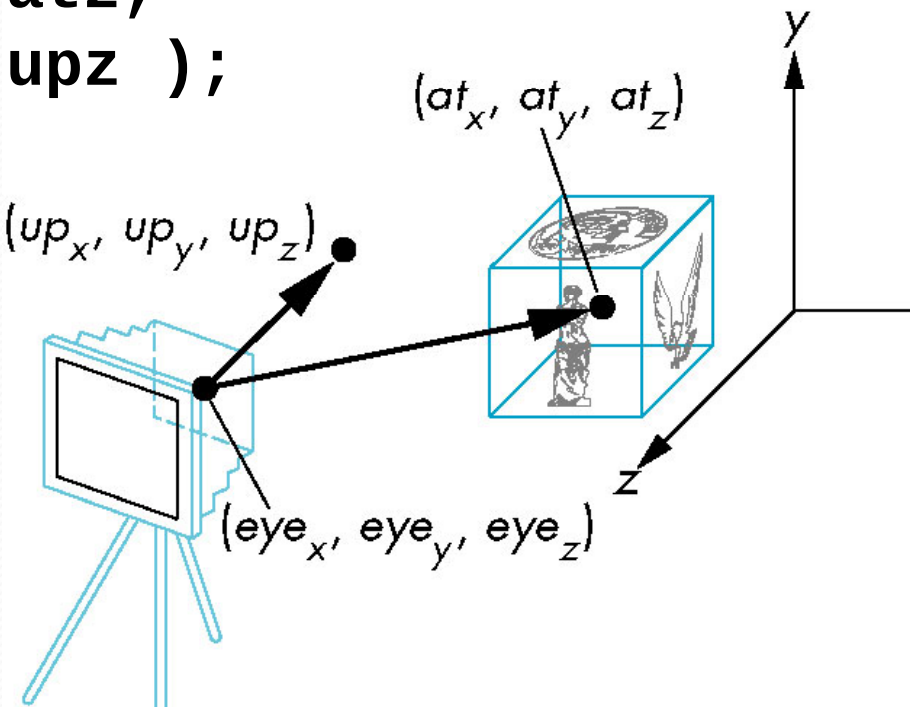
OpenGL Viewing

OpenGL Projection: Step 1 (Moving to the camera reference frame)

- First, when you are still in

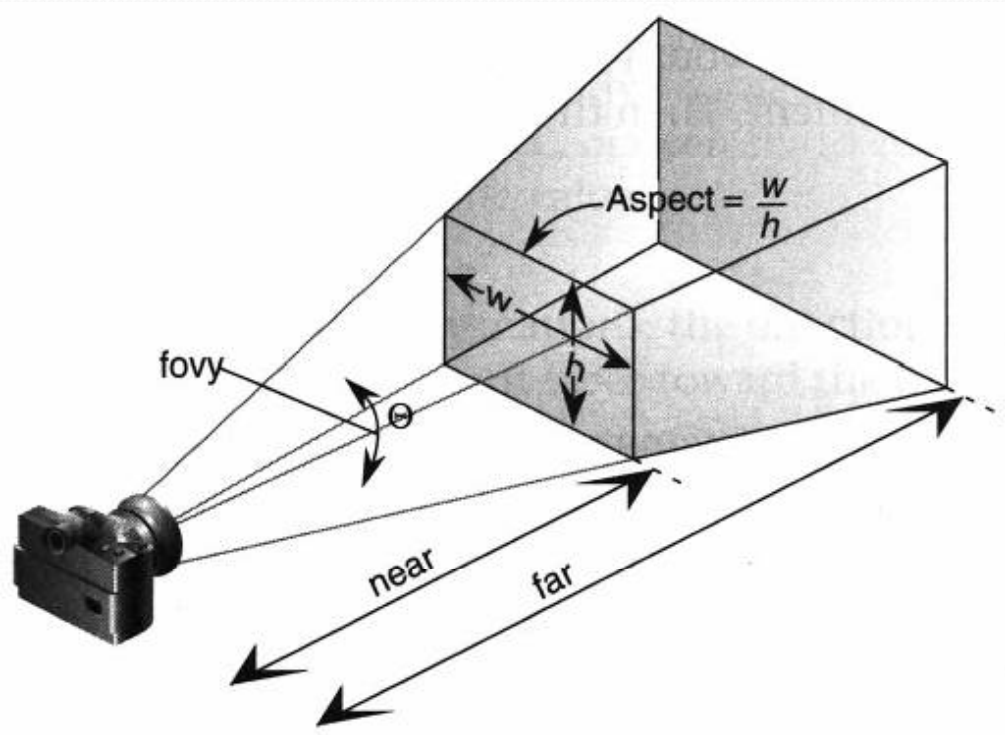
```
glMatrixMode(GL_MODELVIEW);
gluLookAt( eyex, eyey, eyez,
           atx, aty, atz,
           upx, upy, upz );
```

Note that it's not the "look-at direction/vector"



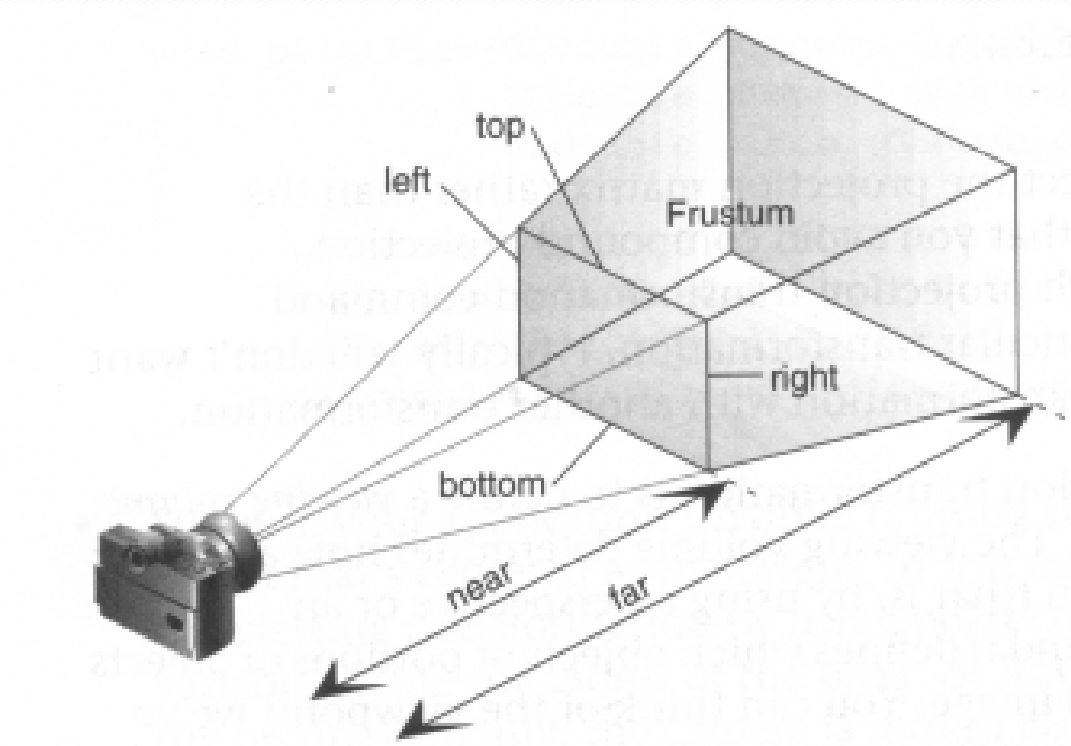
OpenGL Projection: Step 2

- Then change it to the viewing matrix mode
  - **`glMatrixMode(GL_PROJECTION);`**
    `glLoadIdentity();`
    `gluPerspective(fovy,aspect,near,far );`

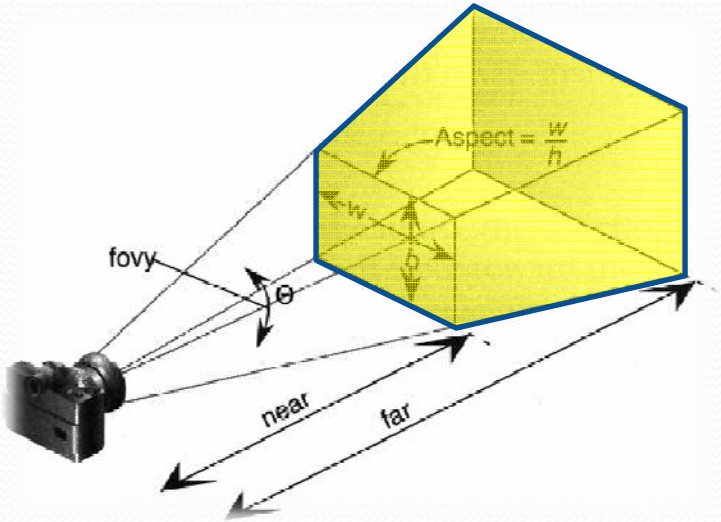


OpenGL Projection: An Alternate Step 2

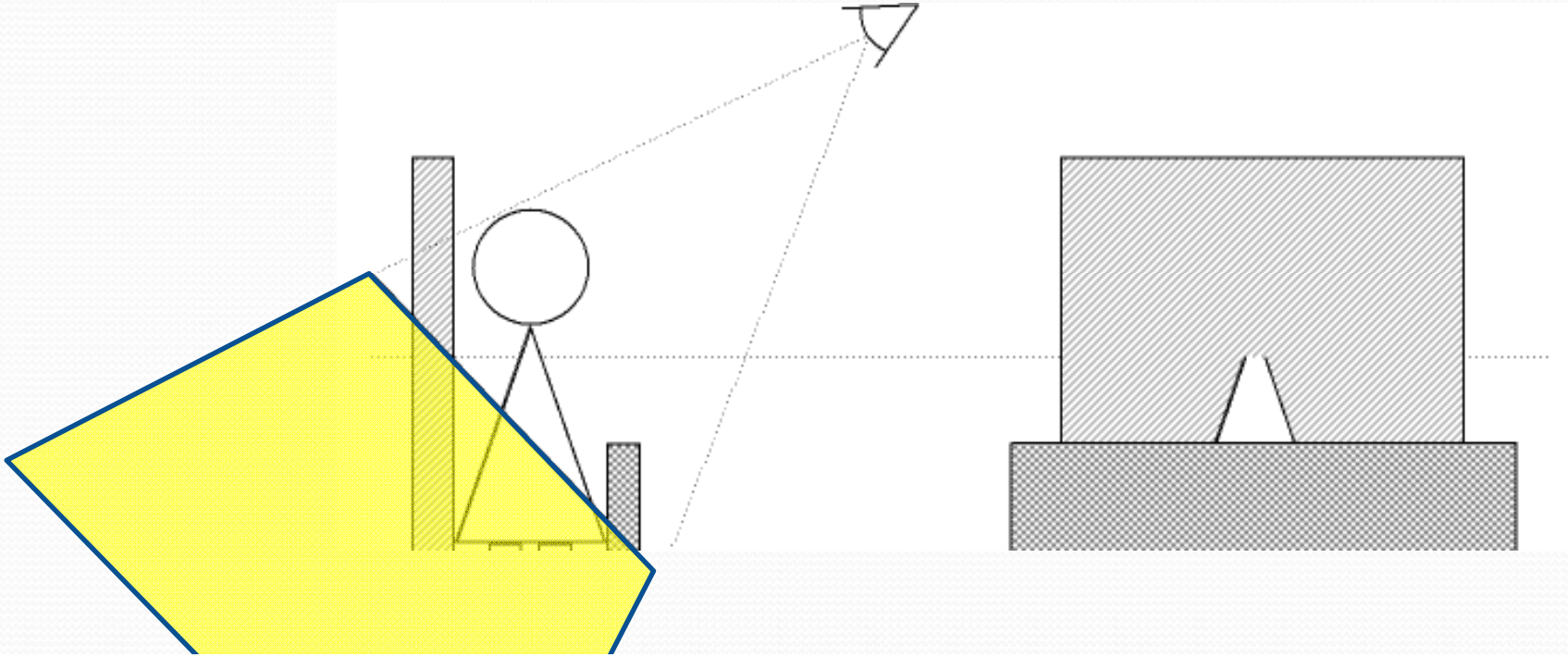
- Another way other than **`gluPerspective`**
  - **`glFrustum(left,right,bottom,top,near,far);`**



Viewing Frustum



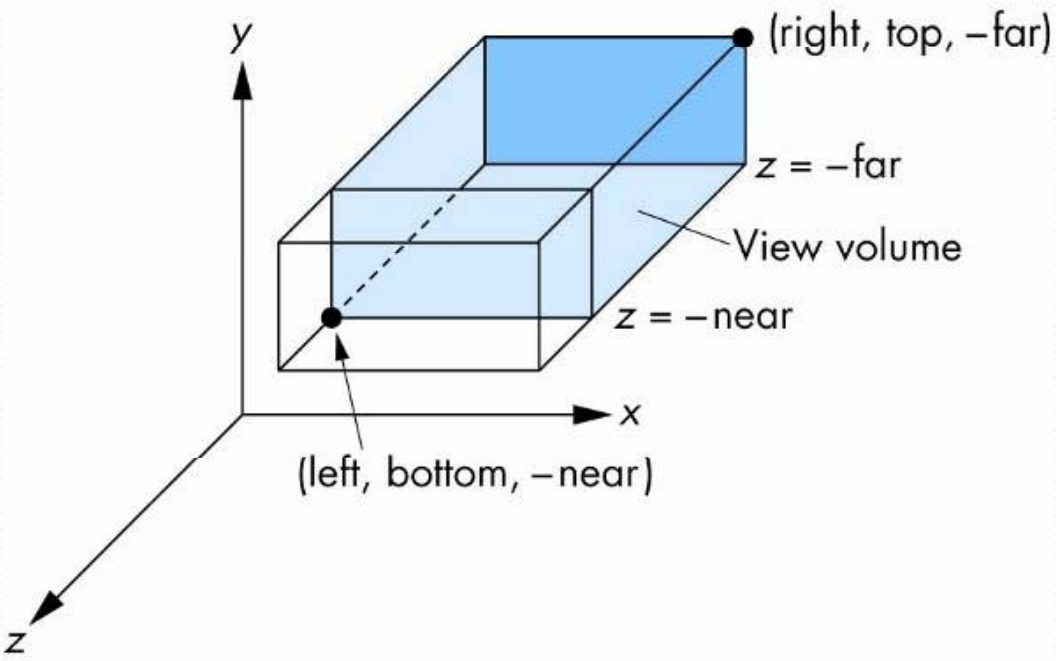
- Objects outside the frustum
  - will be clipped
  - Clipping must be done **before** the perspective projection



Well, if you don't want to "put things in perspective"

- OpenGL Orthographic Projection
- Specified by defining a view using
  - `glOrtho( left, right, bottom, top, near, far );`

- Objects outside
  - Will be clipped



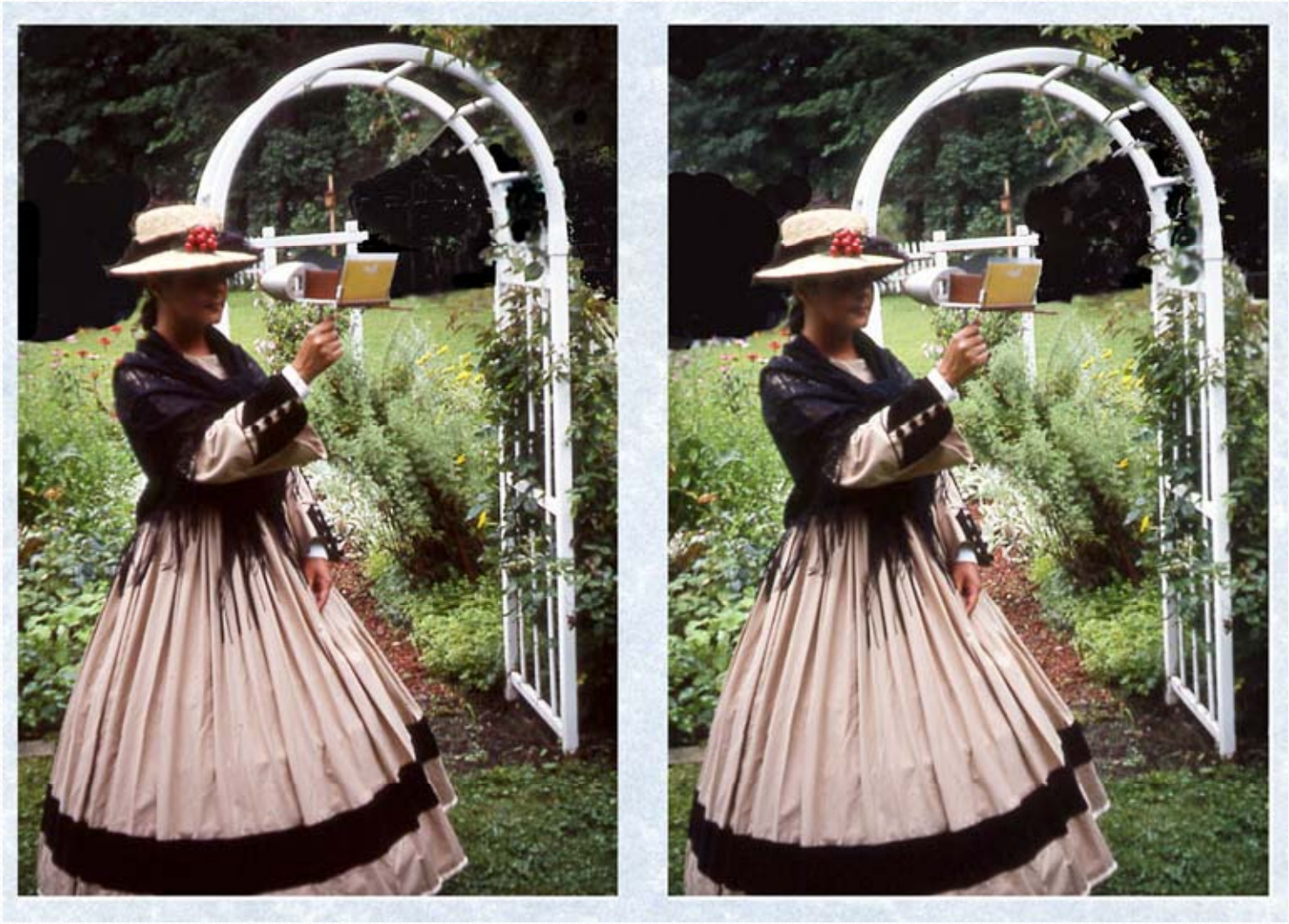
Some Tricks on Projections

How does Matrix make the freeze and rotate animation?

How does 3D Movies Work?

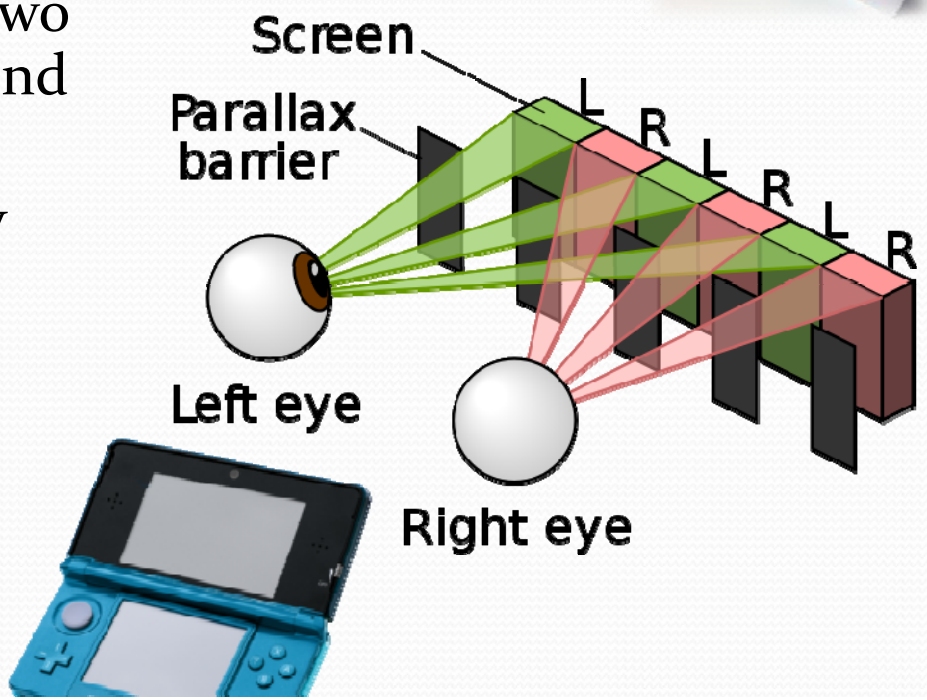
- In normal daily life, we have two eyes, and actually each of them perceive a slightly different image
- Then our brain processes these two image in real time to reconstruct a 3D object in our mind
- 3D cinema/TV try to deliver two different images into your eyes
  - So the movies have to be filmed by two viewpoints
    - For CG movies that is easy
    - For real life movies, double lens

Stereographic Photography



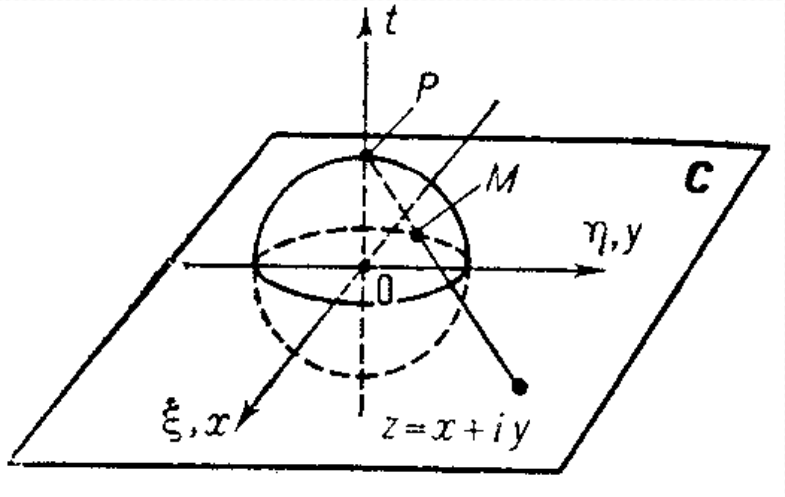
Delivery of Two Different Images

- Anaglyph
  - Red and Blue channels
- Eclipse method
  - Goggle shutters left and right synchronizing with the screen
- Polarization systems
  - Two different images are by two different direction of waves and filtered by the goggle
- Interference filter technology
  - Different wave lengths
- Autostereoscopy
  - No goggle needed
  - Waves are shot with two slightly different directions



Other Projections

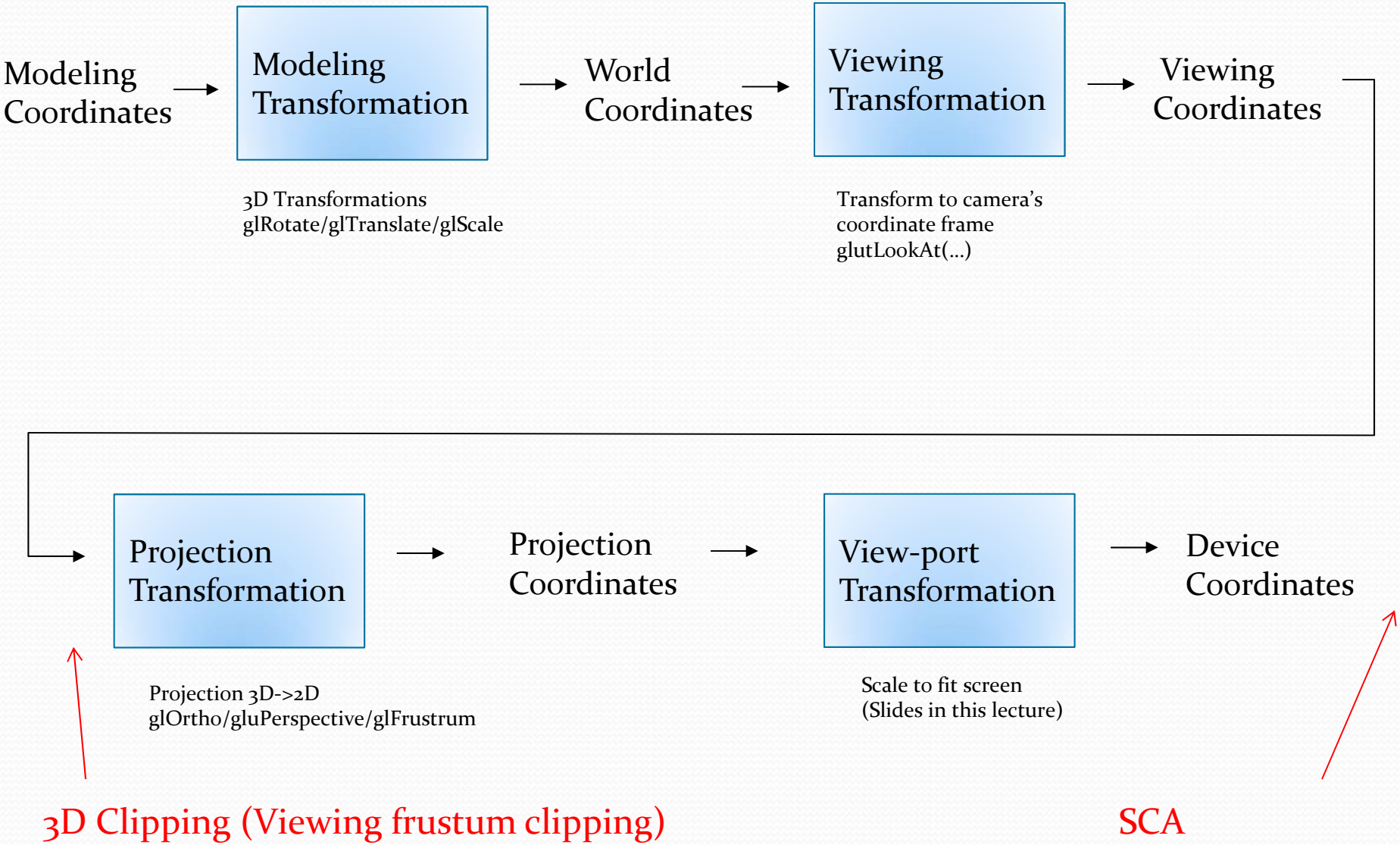
Stereographic Projection



Graphics Pipelines

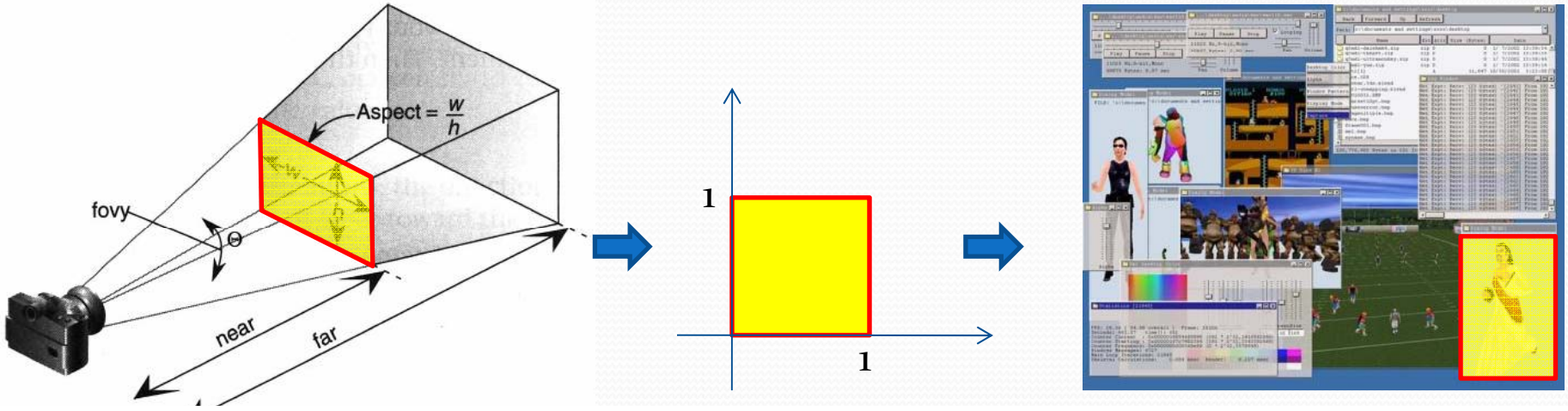
The overall real-time rendering process

Graphic Pipeline



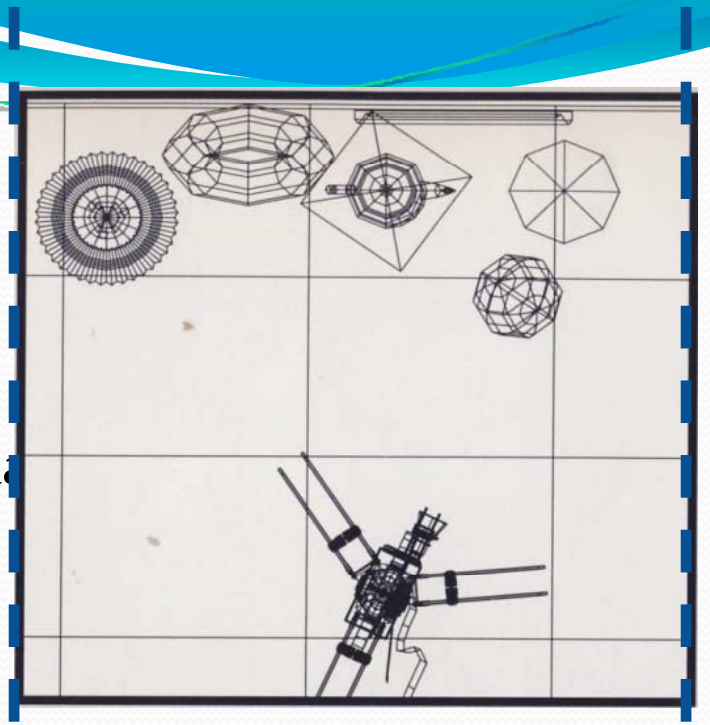
View-port Transformation

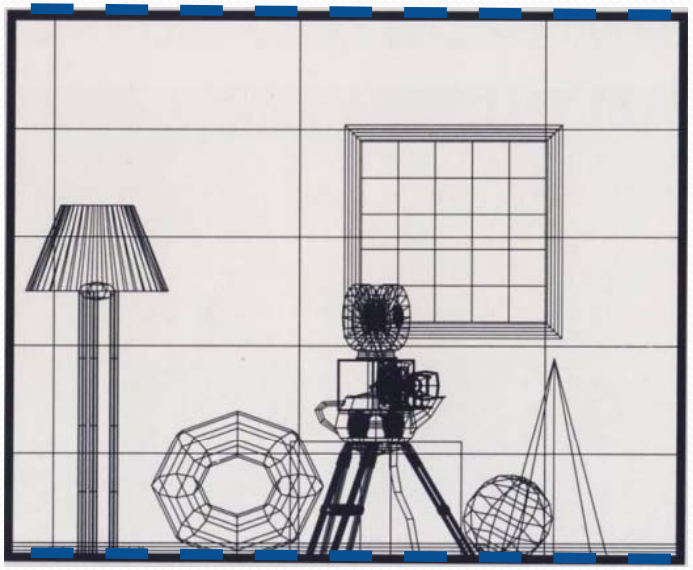
- Screens can be reshape, panned, etc.
- After projection, OpenGL maps your screen to the unit-square
  - Normalized Device Coordinates
- Then map the square to your device, e.g. a window
  - **Scale to the window size then translate**

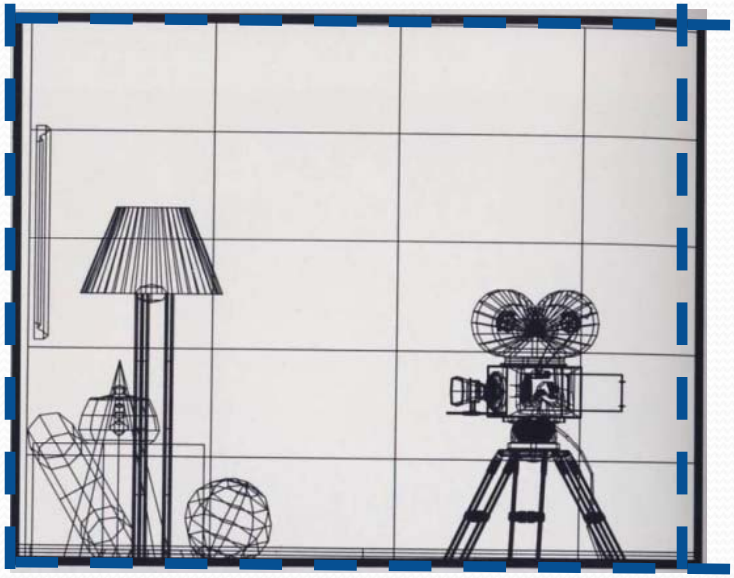


Where are we?

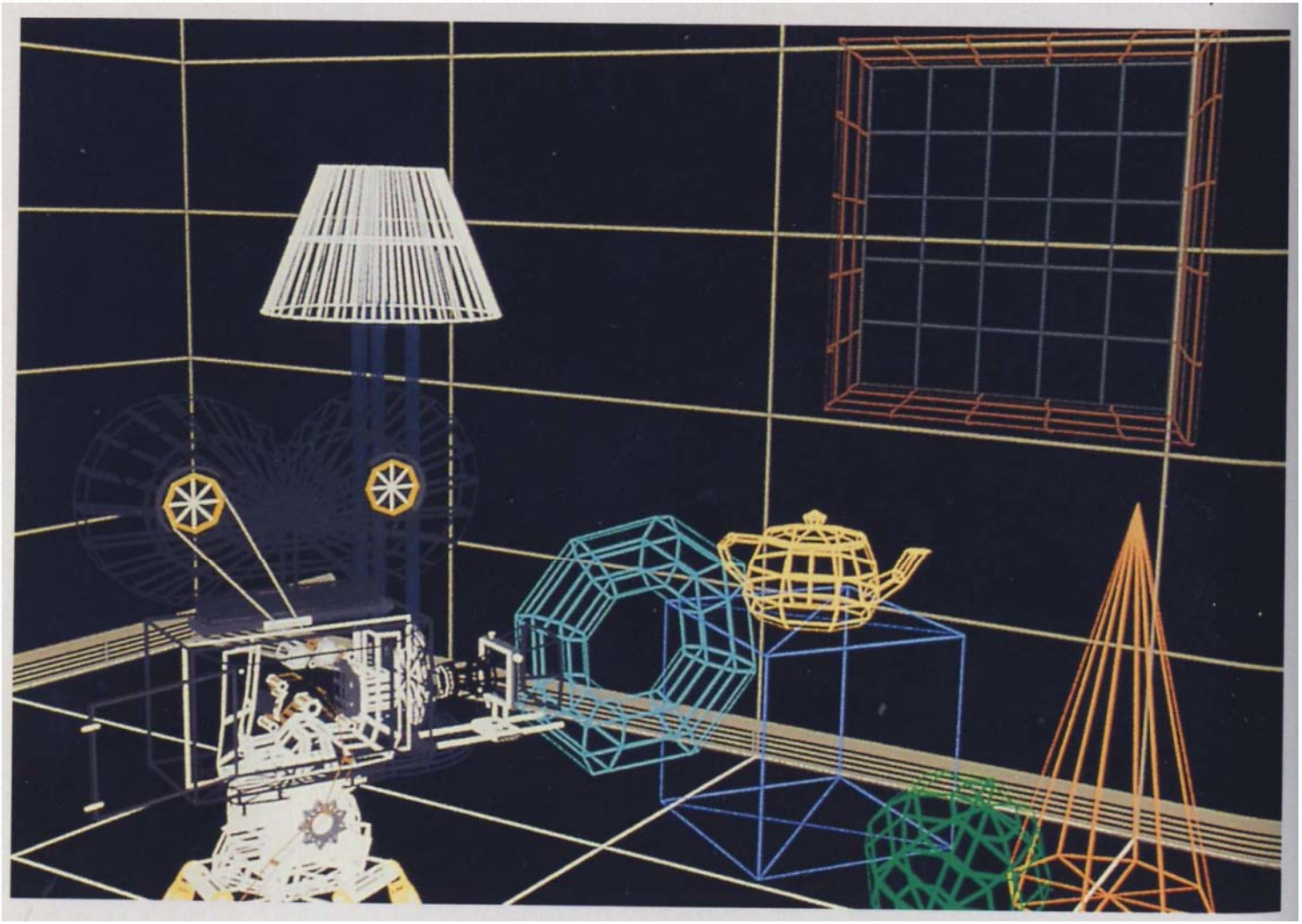
- Object modeling (M,E)
  - Object Representation
    - Using different data structure/forma to represent objects
    - Geometric models





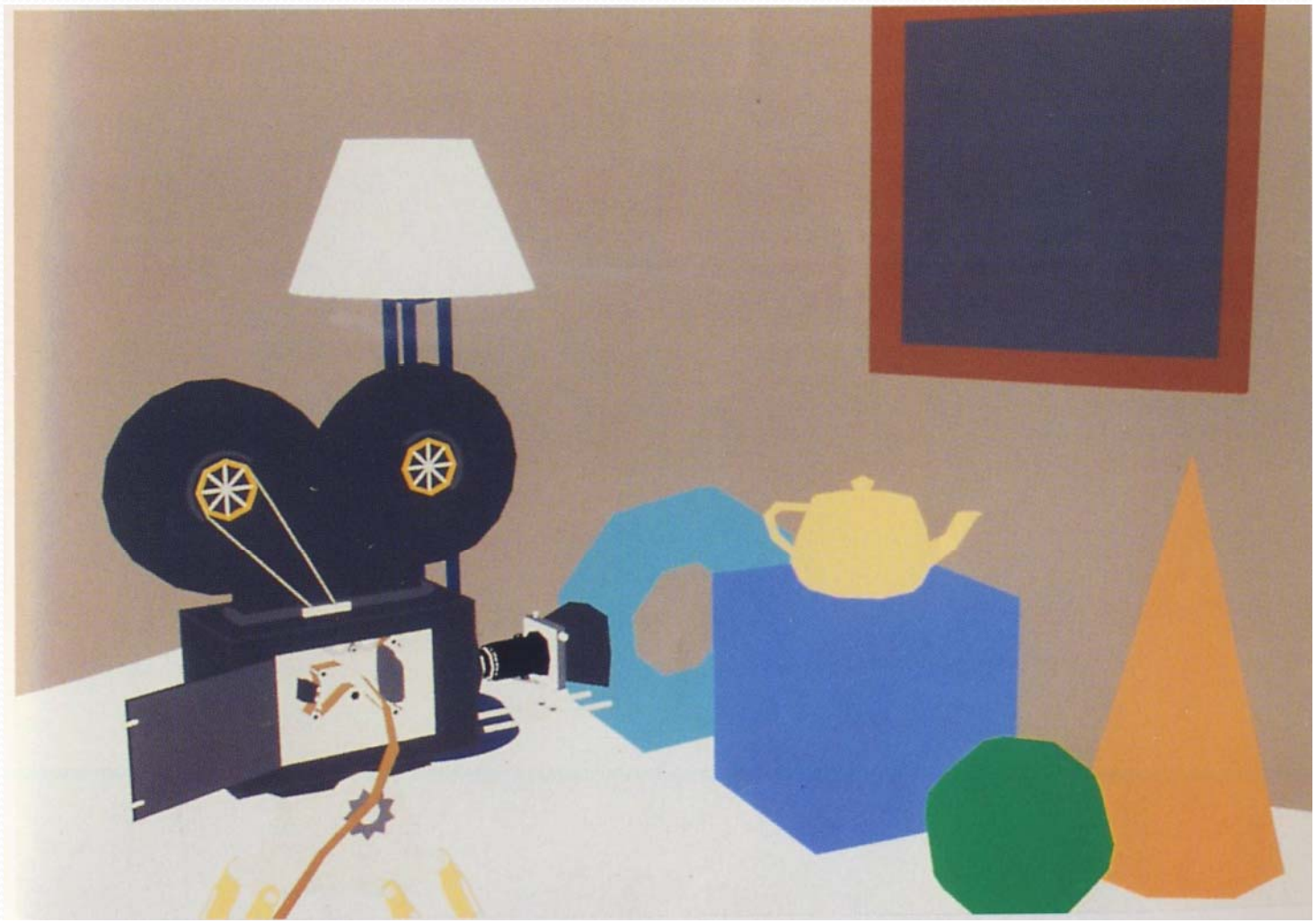


Transformation/Viewing



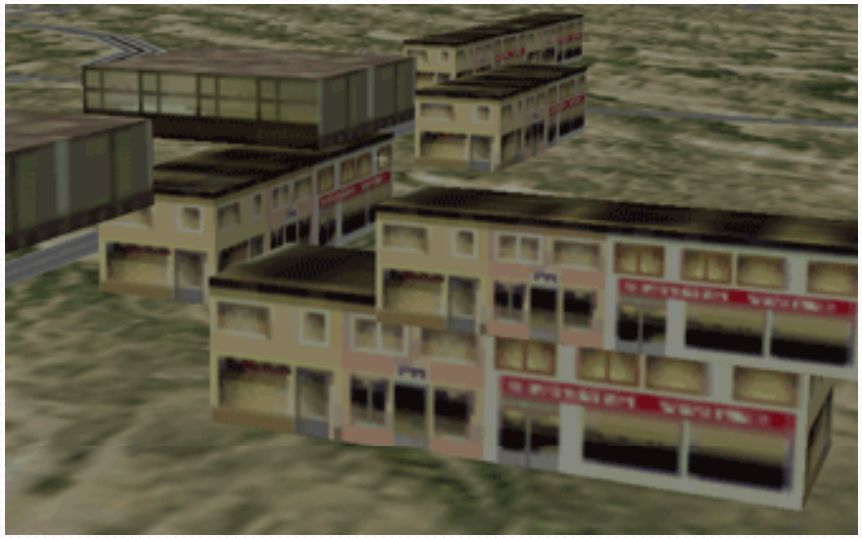
Scan Convert Algorithm

- Next lecture: But how do we know the order?
  - What is in front of/behind of what?

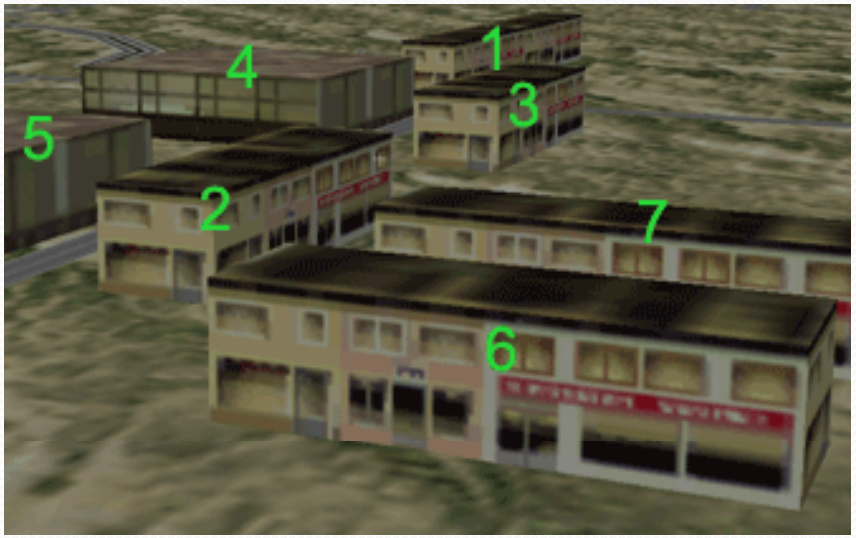


Wrong Drawing Order

- How to draw objects correctly in the order of depth?



Wrong



Correct