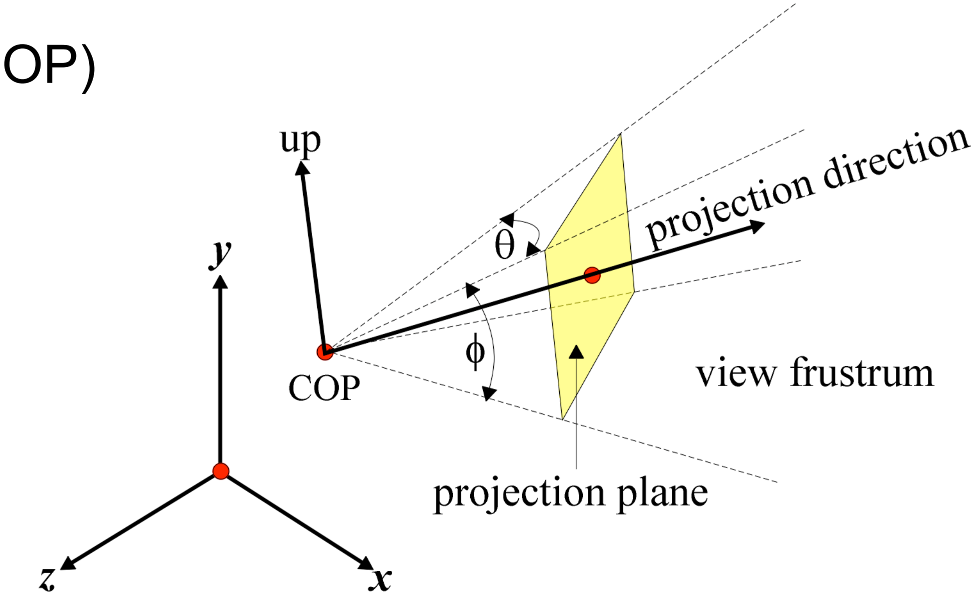
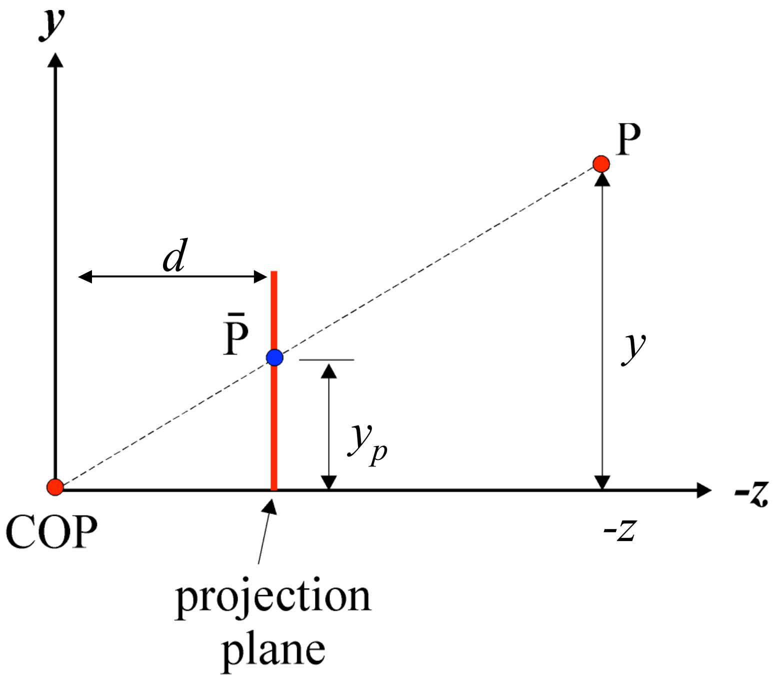
Perspective Projections

- Perspective projections are more complex and exhibit *fore-shortening* (parallel appear to converge at points).
- Parameters:
  - □ centre of projection (COP)
  - □ field of view (θ,ϕ)
  - □ projection direction
  - □ up direction



Perspective Projections

Consider a perspective projection with the viewpoint at the origin and a viewing direction oriented along the positive -***z*** axis and the view-plane located at *z* = -*d*



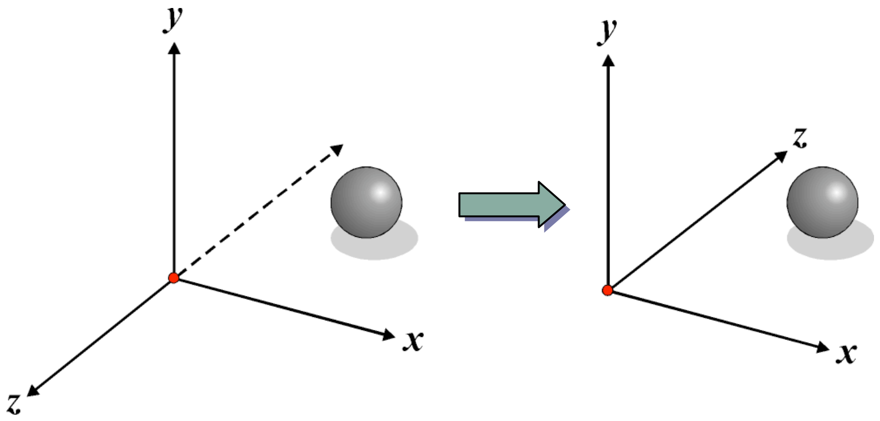
$$\frac{y}{z} = \frac{y_P}{d} \Rightarrow y_P = \frac{y}{z/d}$$

a similar construction for $x_p \Rightarrow$

$$\begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{x}{z/d} \\ \dfrac{y}{z/d} \\ -d \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} x \\ y \\ -z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

divide by homogenous ordinate to map back to 3D space

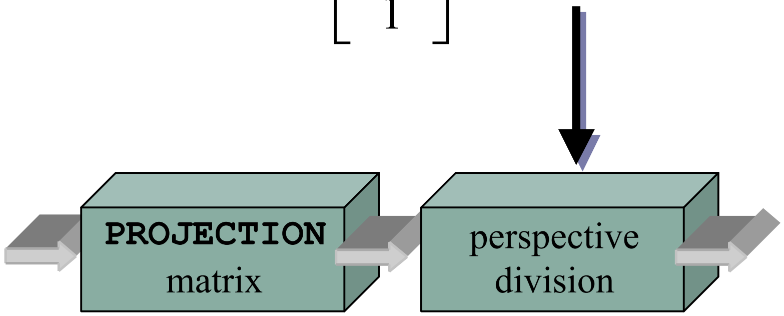
Perspective Projections Details



$$\begin{bmatrix} x \\ y \\ -z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Flip ***z*** to transform to a left handed co-ordinate system ⇒ increasing *z* values mean increasing distance from the viewer.

$$\begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{x}{z/d} \\ \dfrac{y}{z/d} \\ -d \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} x \\ y \\ -z \\ z/d \end{bmatrix}$$



PROJECTION matrix

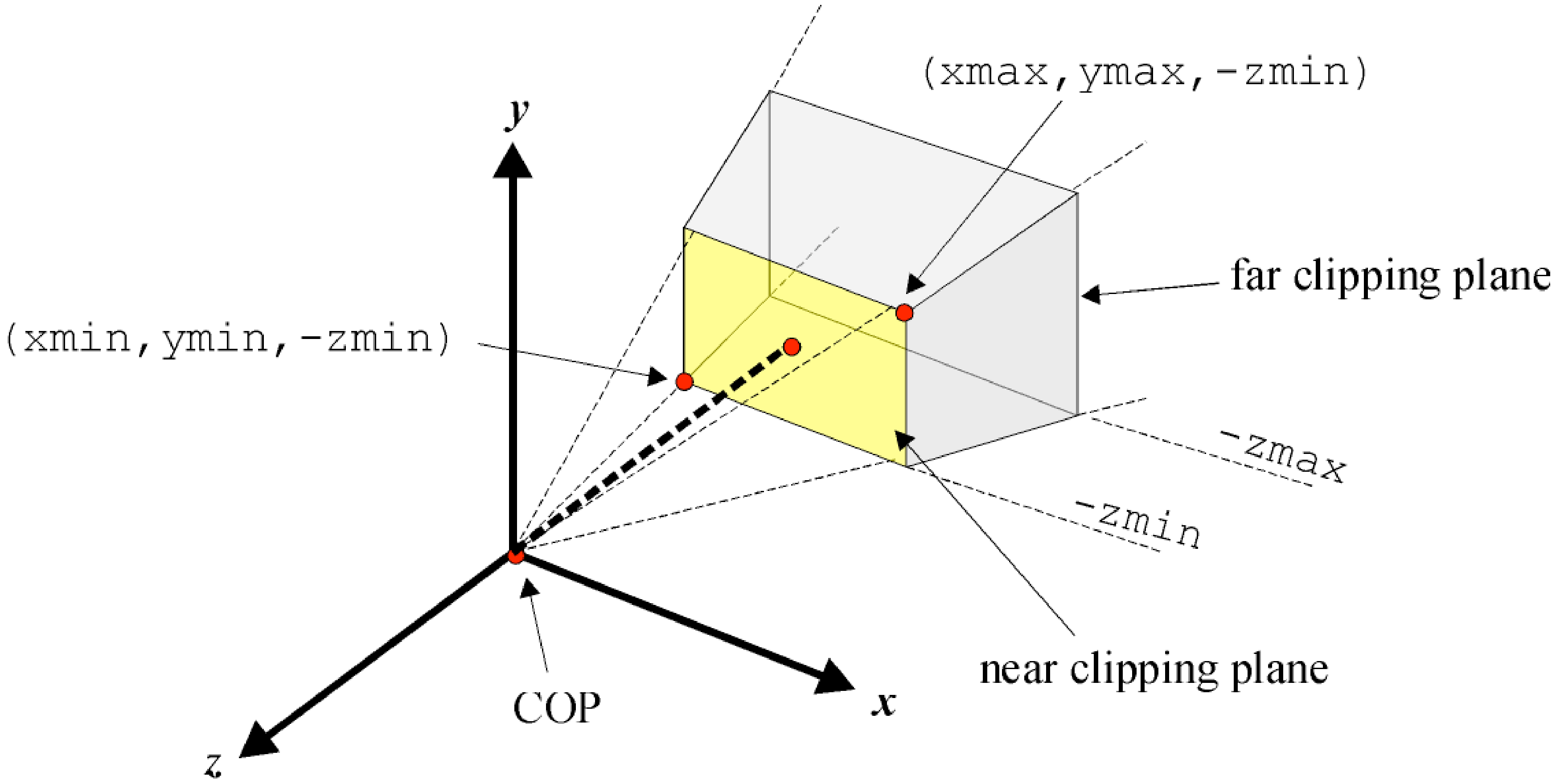
perspective division

Perspective Projection

- Depending on the application we can use different mechanisms to specify a perspective view.
- Example: the *field of view* angles may be derived if the distance to the viewing plane is known.
- Example: the viewing direction may be obtained if a point in the scene is identified that we wish to look at.
- OpenGL supports this by providing different methods of specifying the perspective view:
  - **`gluLookAt`**, **`glFrustrum`** and **`gluPerspective`**

Perspective Projections

```
glFrustrum(xmin, xmax, ymin, ymax, zmin, zmax);
```



glFrustrum

- Note that all points on the line defined by (xmin,ymin,-zmin) and COP are mapped to the *lower left* point on the viewport.
- Also all points on the line defined by (xmax,ymax,-zmin) and COP are mapped to the upper right corner of the viewport.
- The viewing direction is always parallel to ***-z***
- It is not necessary to have a *symmetric frustrum* like:

```
glFrustrum(-1.0, 1.0, -1.0, 1.0, 5.0, 50.0);
```

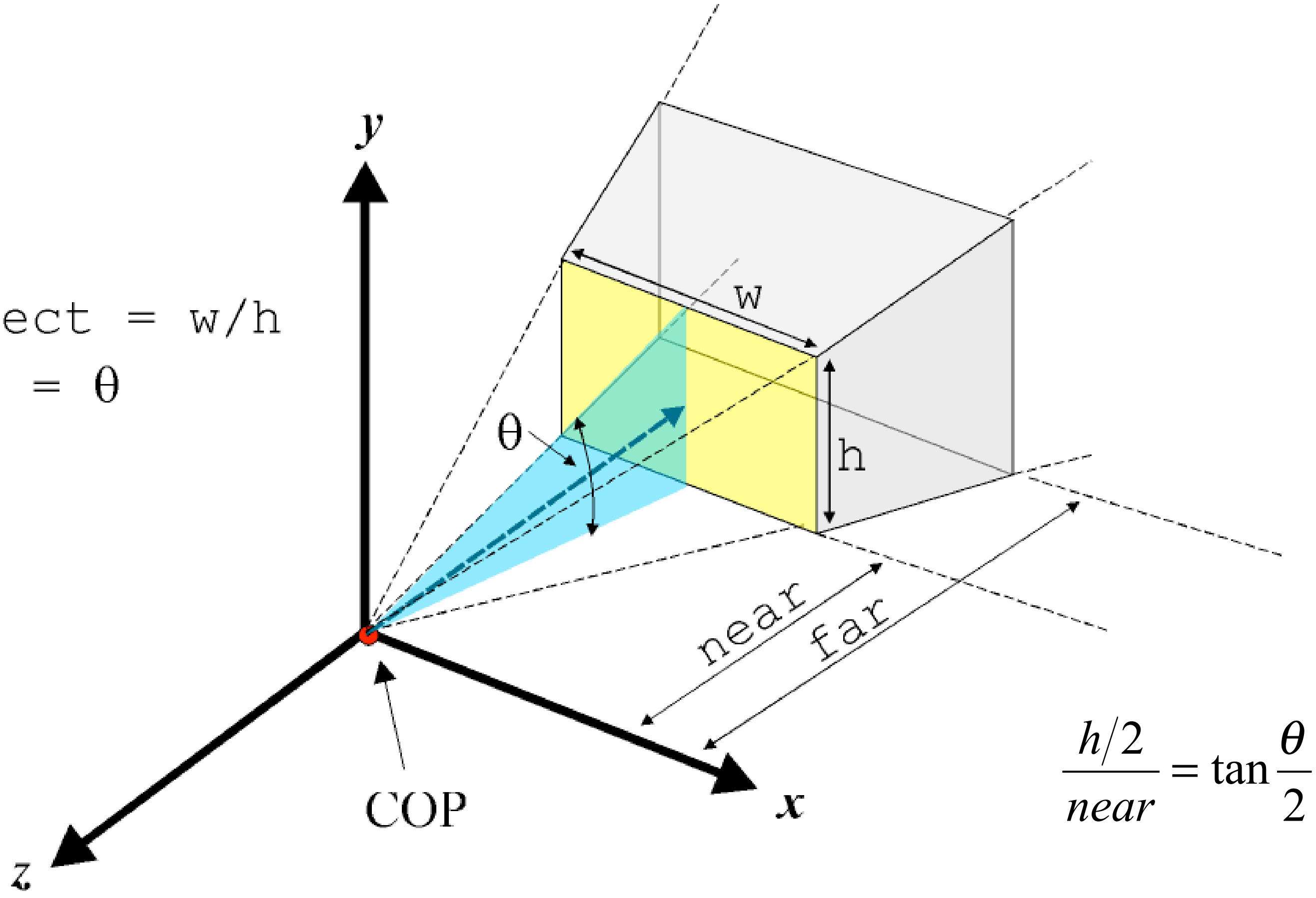
- Non symmetric frustrums introduce *obliqueness* into the projection.
- zmin and zmax are specified as <u>positive</u> distances along ***-z***

Perspective Projections

```
gluPerspective(fov, aspect, near, far);
```

aspect = w/h

fov = θ

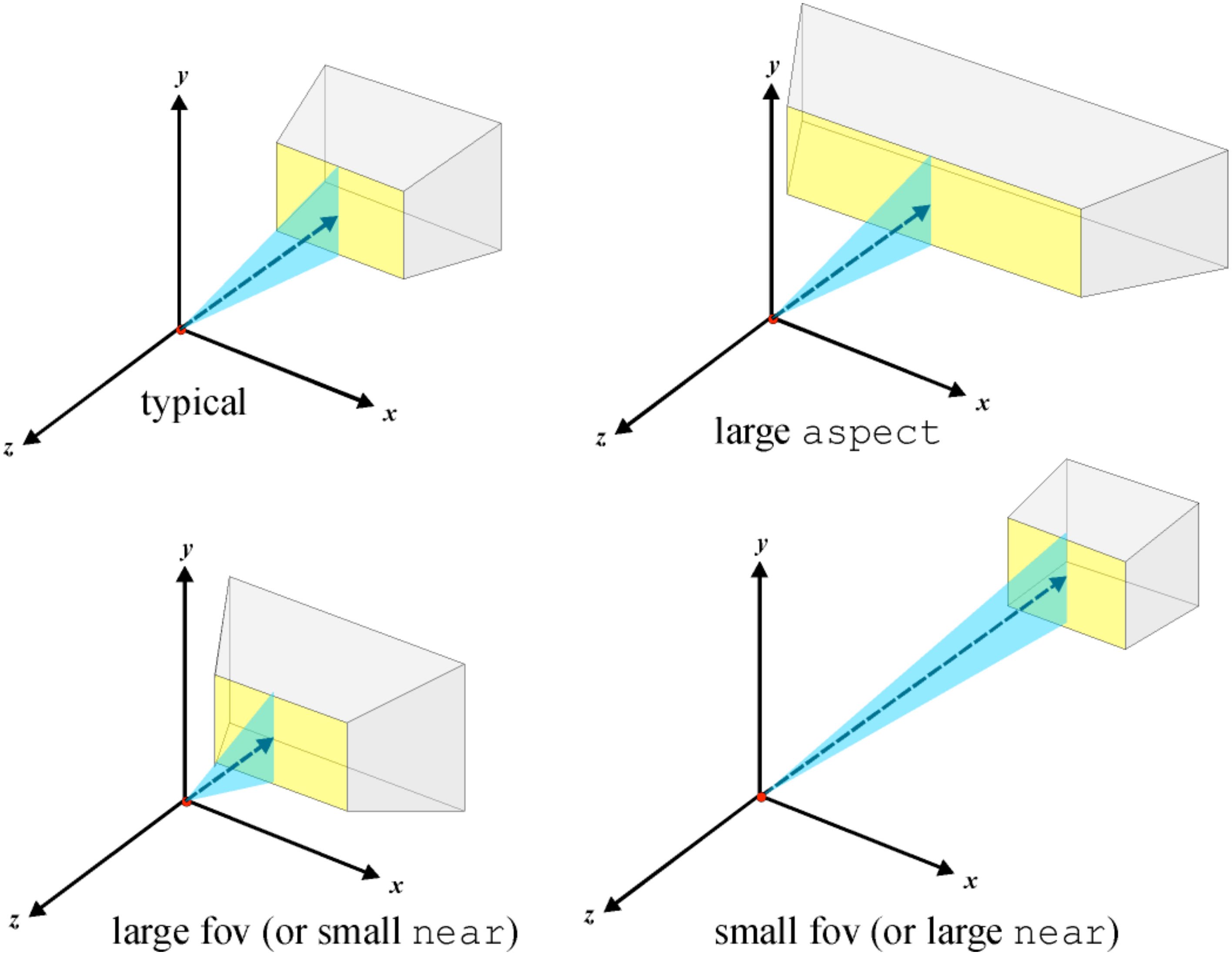


$$\frac{h/2}{near} = \tan\frac{\theta}{2} \Rightarrow h = 2\,near\tan\frac{\theta}{2}$$

`gluPerspective`

- A utility function to simplify the specification of perspective views.
- Only allows creation of *symmetric frustrums*.
- Viewpoint is at the origin and the viewing direction is the ***-z*** axis.
- The *field of view* angle, `fov`, must be in the range [0..180]
- `aspect` allows the creation of a view frustrum that matches the *aspect ratio* of the viewport to eliminate distortion.

Perspective Projections



typical

large aspect

large fov (or small near)

small fov (or large near)