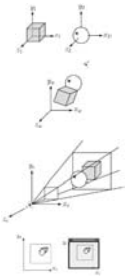
3D Viewing

- 3D viewing process
- parallel projections
- perspective projections

Mapping 3D to 2D

- 3D scene models, 2D images
  - ⇒ we need a method to project 3D to 2D
- Fundamental approach: planar projection:
  - define a plane in 3D space
  - project scene onto this plane
  - map it to 2D viewport
- But:
  - how to define plane
  - how to define mapping onto plane

3D Viewing Process



3D Model

Modeling transformations

World space

View transformations

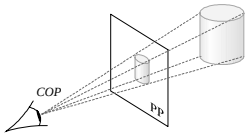
Viewing Reference Space

Projection

Screen space

Projections

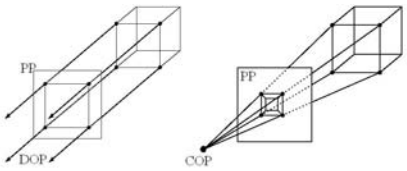
- Mapping points from 3D space to the **projection plane (PP)** along projectors emanating from the **center of projection (COP)**



- Two basic types of projections:
  - Perspective: finite distance from COP to PP
  - Parallel: infinite distance from COP to PP

Parallel and Perspective Projection

Parallel **Perspective**



Parallel vs. Perspective Projection

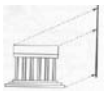
- Parallel projection:
  - \+ good for exact measurement (engineering, maps, …)
  - \+ parallel lines remain parallel
  - \- less realistic looking
  - \- angles preserved only on faces parallel to the PP

- Perspective projection:
  - \+ sizes varies inversely with distance (looks realistic)
  - \- distance and angles are not preserved
  - \- parallel lines don't remain parallel

Parallel Projections

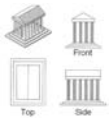
- We have a **direction of projection (DOP)** instead of COP
- There are two types:
  - **Orthographic projection:** DOP perpendicular to PP

  - **Oblique projection:** DOP not perpendicular to PP



Parallel Projection :Orthographic

- PP perpendicular to a principal axis:
  - front elevation
  - top (plan) elevation
  - side elevation



- Axonometric: PP not perpendicular to a principal axis

Parallel Projection: Orthographic (2)

- e.g. front elevation:
  projection plane is XY-plane (z = 0 plane)

$$M = \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Parallel Projection: Orthographic (3)

projection plane is not normal to any coordinate axis
rotated by an arbitrary angle

e.g.

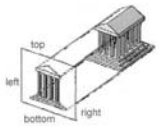
$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Parallel Projection: Orthographic (4)

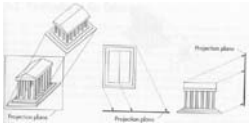
- OpenGL:

```
glOrtho (left, right, bottom, top, near, far);
```

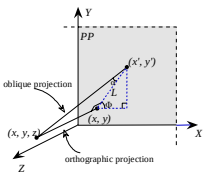
  - assumes image plane perpendicular to z axis (xy-plane)
  - project points *(x, y, z)* to *(x, y)*
  - also defines viewport mapping:



Parallel Projection: Oblique



- projectors are parallel
- direction of projection is not normal to projection plane
- two angles of interest:
  - angle between DOP and PP (α)
  - angle of projection in PP (Φ)



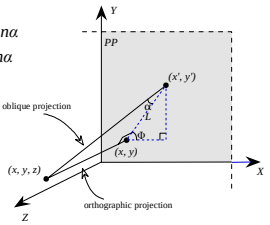
Parallel Projection: Oblique (2)

$L = z / \tan\alpha$

$x' = x + L\cos\Phi = x + z\cos\Phi / \tan\alpha$

$y' = y + L\sin\Phi = y + z\sin\Phi / \tan\alpha$

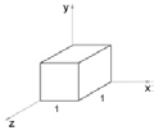
$$M = \begin{bmatrix} 1 & 0 & \cos\Phi/\tan\alpha & 0 \\ 0 & 1 & \sin\Phi/\tan\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Parallel Projection: Oblique (3)

Cavalier

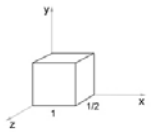
- α = 45 (arctan(1)): DOP makes 45 angle with PP
- lines perpendicular to PP are not foreshortened
- all sides of a cube maintain equal length
- + edges can be measured dir
- - object look too elongated



Parallel Projection: Oblique (4)

Cabinet

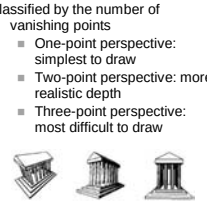
- α = 63.4 (arctan(2)): DOP makes 63.4 angle with PP
- lines perpendicular to PP are foreshortened by one-half
- appear more realistic than cavalier, because of the reduction of length in lines perpendicular to the PP



Perspective Drawing

Classified by the number of vanishing points

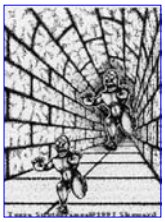
- One-point perspective: simplest to draw
- Two-point perspective: more realistic depth
- Three-point perspective: most difficult to draw



- All three types are equally simple in computer graphics: defines by the orientation of projection plane

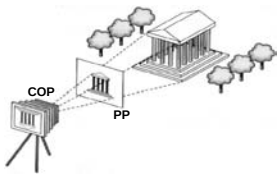
Perspective Projection

- What did the perspective projection do to Z?

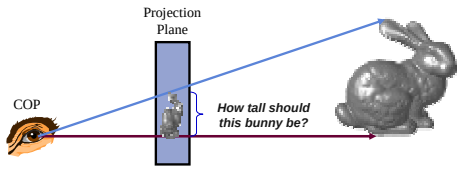


Perspective Projection (2)

- The image is the intersection of the projectors through object and the projection plane

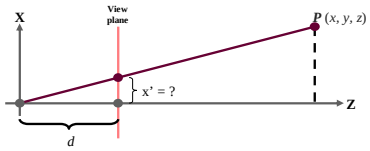


Perspective Projection (3)



Perspective Projection (4)

Viewing from top: what is x'?
hint: find the similar triangles



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

Perspective Projection (5)

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}$$

$$x' = \frac{d \cdot x}{z} = \frac{x}{z/d}, \quad y' = \frac{d \cdot y}{z} = \frac{y}{z/d}, \quad z = d$$

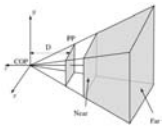
$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x' \\ y \\ z \\ 1 \end{bmatrix} \quad \textbf{OR} \quad \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

when:
- center of projection at z = 0
- projection plane at distance d

when
- center of projection at z = -d
- projection plane at z = 0 (xy-plane)

Perspective Projection: Viewing Volume

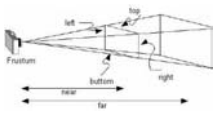
- Sides of the viewport define an infinite pyramid where COP is at apex
- Adding the clipping planes we get a trucated pyramid called a frustum
- Can think of this as the viewing volume:
  - nothing outside is visible
  - projection maps this volume to a rectangular window



Perspective Projection: OpenGL

- defining frustum by (*left*, *right*, *top*, *bottom*) coordinates
- near & far clipping planes (customarily)
- things nearer than *near* and farther than *far* won't be drawn

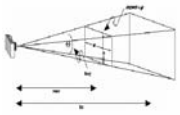
```
glFrustum (left, right, bottom, top, near, far);
```



Perspective Projection: OpenGL (2)

- view port is always centered about z axis
- field of view y (fovy): the angle made by the sides of the frustum along y axis
- aspect: aspect ratio of the view port (width/height)

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



Perspective Projection: OpenGL (3)

- So far so good, but limited:
  - COP (focal point) at the origin
  - image plane parallel to xy-plane

√√ we can transform world (viewing transformation)

√√ or in more general: we can parameterize camera by three vectors:
  - lookFrom: position of COP: (ex, ey, ez)
  - lookAt: the position camera is pointing at: (ax, ay, az)
  - vUp: vertical axis of image (ux, uy, uz)

```
gluLookAt (ex, ey, ez, ax, ay, az, ux, uy, uz)
```


Summary: Planar Geometry Projections