CSE 413
(Computer Graphics)
Camera Transformations and Projection

Iyolita Islam

Department of Computer Science and Engineering
Military Institute of Science and Technology

Last Updated: September 28, 2020

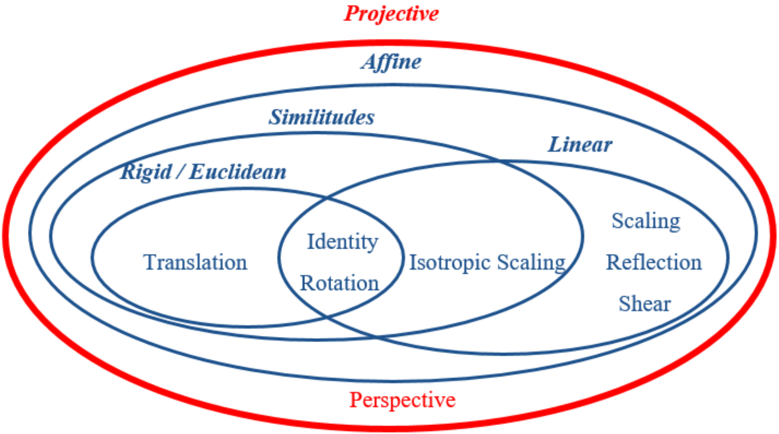
Unintentional Mistakes

Best efforts have been exercised in order to keep the slides error-free, the preparer does not assume any responsibility for any unintentional mistakes. The text books must be consulted by the user to check veracity of the information presented.

Outline

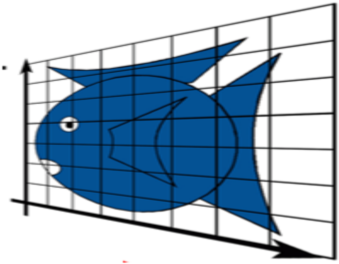
Projection

- The next step of MODELING transformation is VIEW transformation.



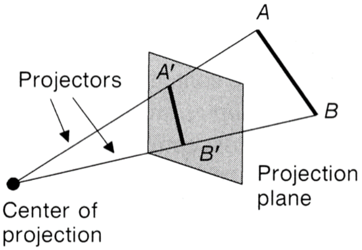
Projection

- In general, projections transform points in a coordinate system of dimension n into points in a coordinate system of dimension less than n.
- We shall limit ourselves to the projection from *3D to 2D*.
- We will deal with planar geometric projections where:
  - The projection is onto a plane rather than a curved surface
  - The projectors are straight lines rather than curves
- Projection preserves lines.



Projection

- Projection from 3D to 2D is defined by straight projection rays (projectors) emanating from the *center of projection*, passing through each point of the object, and intersecting the *projection plane* to form a projection.



Planer Geometric Projection

According to the center of projection there are two types of projections:

- Perspective Projection : if distance to center of projection is finite
- Parallel Projection : if distance to center of projection is infinite

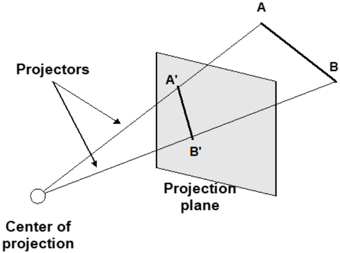


Figure: Perspective Projection

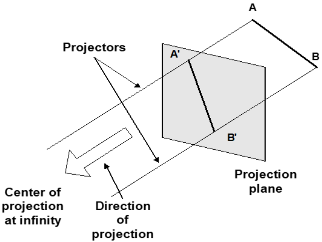
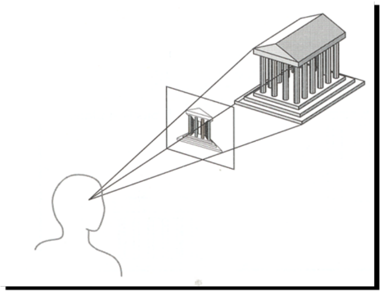


Figure: Parallel Projection

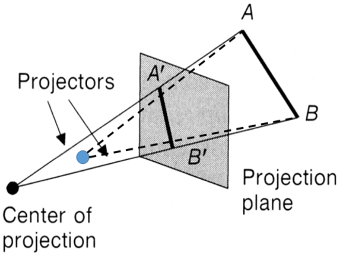
Perspective Projection

- Visual effect of perspective projection is similar to human visual system.
- Parallel lines do not in general project to parallel lines
- Angles only remain intact for faces parallel to projection plane.

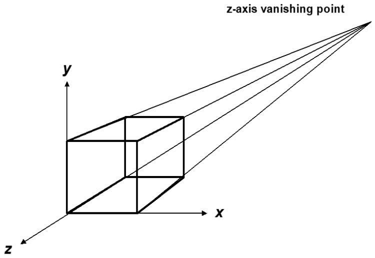


Perspective Projection

Perspective foreshortening: The farther an object is from COP the smaller it appears.



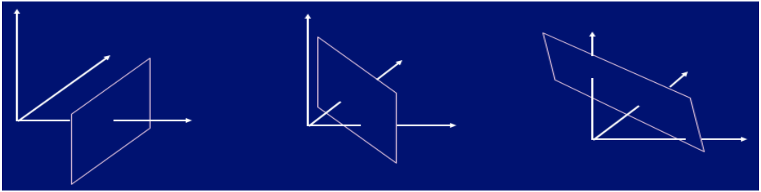
Vanishing Points: Any set of parallel lines not parallel to the view plane appear to meet at some point. There are an infinite number of these, 1 for each of the infinite amount of directions line can be oriented



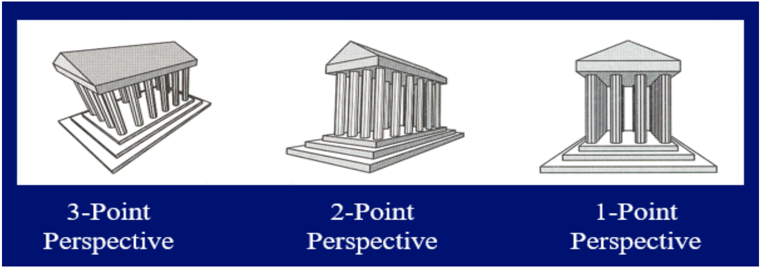
Vanishing Point

If a set of lines are parallel to one of the three axes, the vanishing point is called an axis vanishing point (Principal Vanishing Point).
There are at most 3 such points, corresponding to the number of axes cut by the projection plane

- One axis vanishing point: One principle axis cut by projection plane.
- Two axis vanishing points: Two principle axes cut by projection plane.
- Three axis vanishing points: Three principle axes cut by projection plane.

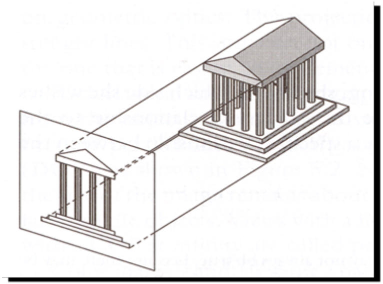


Vanishing Point



Parallel Projection

- Less realistic view because of no foreshortening
- Parallel lines remain parallel.
- Angles only remain intact for faces parallel to projection plane.



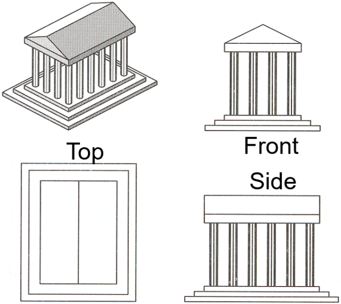
Parallel Projection

Two principal types of Parallel Projection:

- Orthographic : Direction of projection (DOP) = normal to the projection plane.
- Oblique : Direction of projection (DOP) != normal to the projection plane.

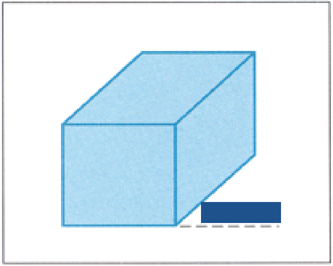
Orthographic Parallel Projection

- Direction of projection is perpendicular to view plane

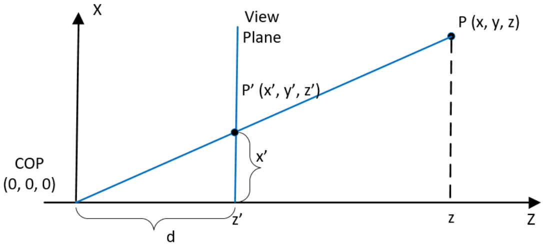


Oblique Parallel Projection

- Direction of projection is not perpendicular to view plane



Perspective Projection Matrix



- The projected point of P on XY plane is P'.
- △POZ and △P'OZ' are similar.

  Here, z' = d

  $\Rightarrow \frac{x'}{x} = \frac{z'}{z} \Rightarrow$ x' $= \frac{z'x}{z} = \frac{dx}{z}$

  $\Rightarrow$ x' $= \frac{dx}{z}$

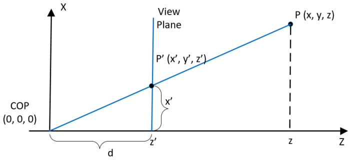
  same, y' $= \frac{dy}{z}$

Perspective Projection Matrix

- projection is not linear in Cartesian coordinate system. So, the resultant matrix will be calculated in Homogeneous coordinate system (4X4 matrix) .

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

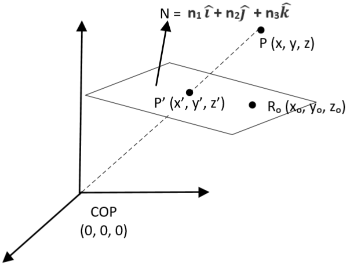
$$\Rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ dz \\ z \end{bmatrix} = \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ d \\ 1 \end{bmatrix}$$



Projection - Example

Schaum's Outline, Problem 7.3

- Camera at (0, 0, 0) and projection plane is given in point normal form (R_o and N).
  $R_o = (x_o, y_o, z_o)$
  $N = n_1\hat{i} + n_2\hat{j} + n_3\hat{k}$
- P' is the projection of P on the plane.



- P and P' are on the same line. So, performing scaling,
  $\Rightarrow \alpha\vec{OP} = \vec{OP'}$
  $\Rightarrow \alpha x = x'$
  Same for $\alpha y = y'$ and $\alpha z = z'$

- $(P' - R_o).N = 0$
  $\Rightarrow x' n_1 + y' n_2 + z' n_3 = x_0 n_1 + y_0 n_2 + z_0 n_3$
  $\Rightarrow \alpha x n_1 + y n_2 + z n_3 = d_0$ [$R_0.N$ is a constant]
  $\Rightarrow \alpha = \frac{d_0}{xn_1+yn_2+zn_3}$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{d_0 x}{xn_1+yn_2+zn_3} \\ \frac{d_0 y}{xn_1+yn_2+zn_3} \\ \frac{d_0 z}{xn_1+yn_2+zn_3} \\ 1 \end{bmatrix} = \begin{bmatrix} d_0 x \\ d_0 y \\ d_0 z \\ xn_1 + yn_2 + zn_3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} d_0 & 0 & 0 & 0 \\ 0 & d_0 & 0 & 0 \\ 0 & 0 & d_0 & 0 \\ n_1 & n_2 & n_3 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Projection - Example

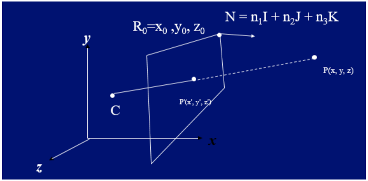
Schaum's Outline, Problem 7.4

- Camera at (0, 0, 0).
- Projection plane is z = d.
- P' is the projection of P on the plane.

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

Projection - Example
Schaum's Outline, Problem 7.5

- Camera at (a, b, c).
- Projection point is given in point normal form. $R_o = (x_o, y_o, z_o)$
  N $= n_1\hat{i} + n_2\hat{j} + n_3\hat{k}$



- We need to,
  1. Translate the camera by (-a, -b, -c)
  2. project P
  3. Translate back camera by (a, b, c)
- So, transformation matrix $= TPT_{back}$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ n_1 & n_2 & n_3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Camera Positioning

- Let, the camera is at (0, 0, 0) point and direction is on X-axis.
  So, camera definition:
  position - (pos.x, pos.y, pos.z)
  l (looking direction) - X
  r (right direction) - Y
  u (up direction) - Z
- l, r and u are unit vectors and perpendicular to each other.
  u = r x l
  r = l x u
  l = u x r

Camera Positioning - openGL

- In openGL,
  r = X
  u = Y
  l = - Z and
  camera position (pos.x, pos.y, pos.z)
- For our own purpose, we need,
  r = X
  u = Y
  l = Z and
  camera position (0, 0, 0)
- So, we need a translation for the camera position and a rotation for the alignment.

Camera Positioning - openGL

- Translation(-pos.x, -pos.y, -pos.z)
- Rotation (for l) $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \text{v.} \begin{bmatrix} -l.x \\ -l.y \\ -l.z \end{bmatrix}$
- Rotation (for u) $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \text{v.} \begin{bmatrix} u.x \\ u.y \\ u.z \end{bmatrix}$
- Rotation (for r) $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \text{v.} \begin{bmatrix} r.x \\ r.y \\ r.z \end{bmatrix}$

Camera Positioning - openGL

- Translation(-pos.x, -pos.y, -pos.z)

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

- Rotation Resultant Matrix= vR

  $$\Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathrm{v} \begin{bmatrix} r.x & u.x & -l.x \\ r.y & u.y & -l.y \\ r.z & u.z & -l.z \end{bmatrix}$$

- I = vR

  $$\Rightarrow v = R^{-1} = R^T$$

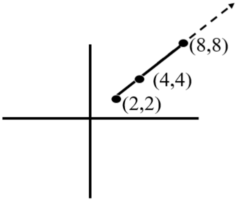
  $$\Rightarrow \mathrm{v} = \begin{bmatrix} r.x & r.y & r.z \\ u.x & u.y & u.z \\ -l.x & -l.y & -l.z \end{bmatrix}$$

Linearity

- In Cartesian co-ordinate system, translation is not linear.
  In homogeneous co-ordinate system, translation is linear.
- In 2D, Cartesian $\Rightarrow$ (x, y) ; Homogeneous $\Rightarrow$ (x, y, w)
- car (x, y) $\xrightarrow{\text{c to h}}$ hom (x, y, 1)
- hom (x, y, w) $\xrightarrow{\text{h to c}}$ cur $(\frac{x}{w}, \frac{y}{w})$

How can we show point and vector at a time in Homogeneous co-ordinate system?

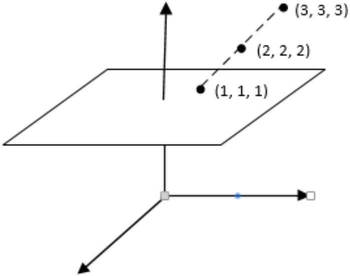
- $(4,4,w)_h = (\frac{4}{w}, \frac{4}{w})_c$
- w $= 1 \Rightarrow (4,4)_c$; w $=\frac{1}{2} \Rightarrow (8,8)_c$
- if we increase w, the point goes in a specific direction.



When w = 0, we can locate the point as infinity. So, when w = 0, we assume the (x, y, w) as a vector.
When w > 0, (x, y, w) is a point.
So, in Homogeneous co-ordinate system, we can show point and vector altogether.
From two points we can generate the vector also:
$\Rightarrow$(4, 4, 1) - (2, 2, 1) = (2, 2, 0)

Homogeneous co-ordinate system?

- In Homogeneous system, after operation among the vectors and points, if w $\neq$ 0 then it is a vector.
- If w $\neq$ 1 also, then we need to scale it to be 1.
- $(2, 2, 2)_h$ must be scaled as $(1, 1, 1)_h$.
- same for $(3, 3, 3)_h$, $(4, 4, 4)_h$; all are to be scaled as (1, 1, 1).



End of Slides