CMT107 Visual Computing

II.2 Viewing

Xianfang Sun

School of Computer Science & Informatics
Cardiff University

Overview

- Projection
  - Parallel projection
  - Perspective projection
- OpenGL viewing

Viewing Transformations

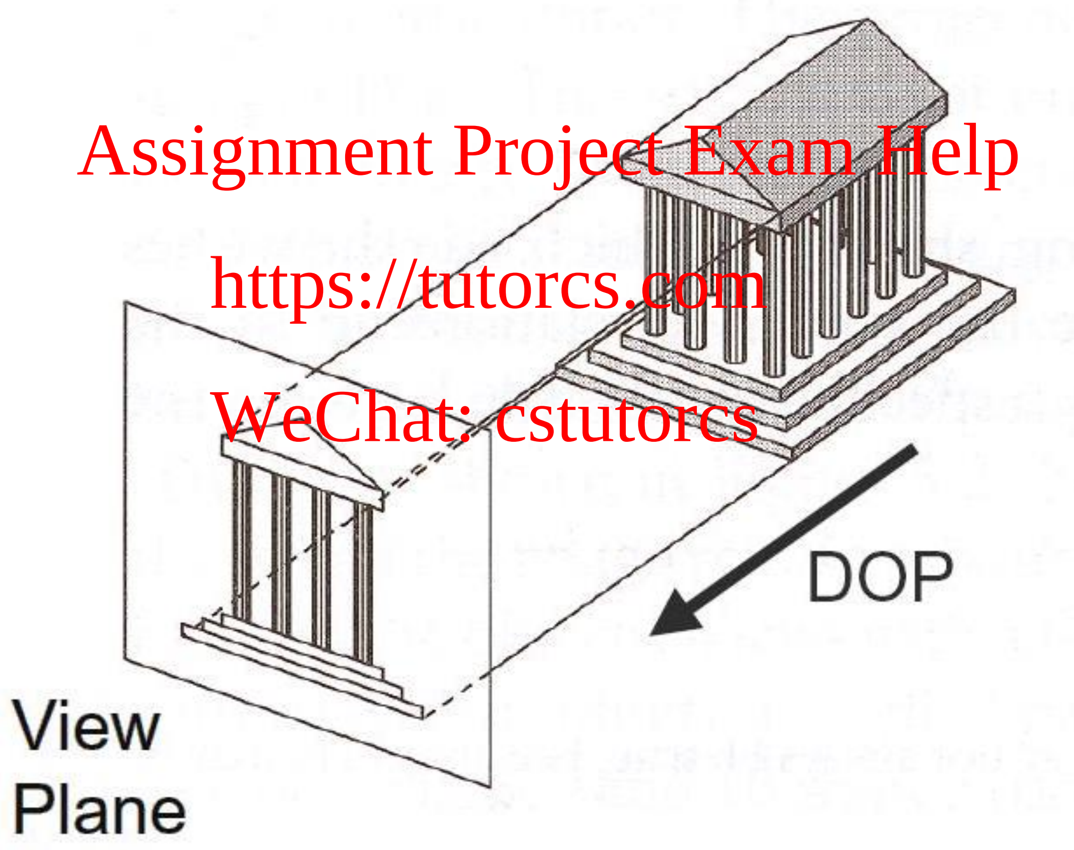
- *Viewing transformations*:
  - *Camera transformation*: 3D world coordinates to 3D camera coordinates
  - *Projection transformation*: Define a viewing volume, and transform 3D camera coordinates onto the view plane
  - *Viewport transformation*: The image on the view plane is translated and scaled to be fitted in the viewport on the screen

Projection

- General definition
  - Transform points in nD space to mD space, $n > m$
- In computer graphics:
  - Map 3D camera coordinates to 2D view plane coordinates
  - Also map depth to a specific range ([0, 1], related to viewing volume)

Parallel Projection

- Centre of projection is at *infinity*
- Direction of projection (DOP) is the *same* for all points



Parallel Projection Matrix

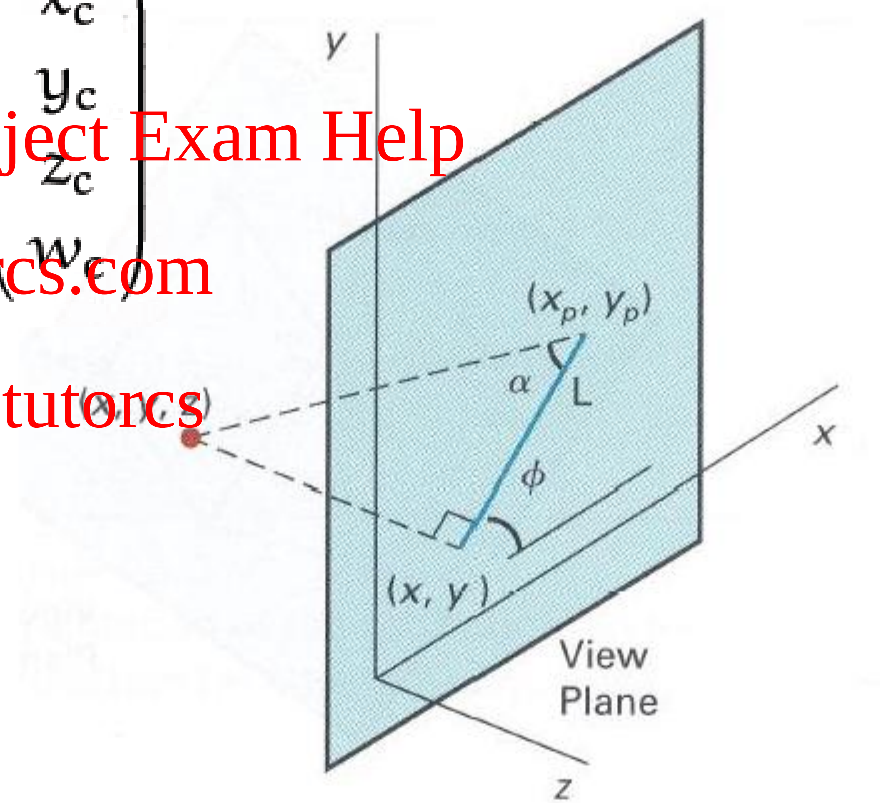
- General parallel projection transformation (defined by α, ϕ)
  - Orthogonal (orthographic) projection for $\alpha = 90°$

$$
\begin{pmatrix} x_p \\ y_p \\ z_p \\ w_p \end{pmatrix} = \begin{pmatrix} 1 & 0 & -L_1 \cos\phi & 0 \\ 0 & 1 & -L_1 \sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix}
$$

$$z_p = 0 \qquad \tan\alpha = \frac{z_c}{L}$$

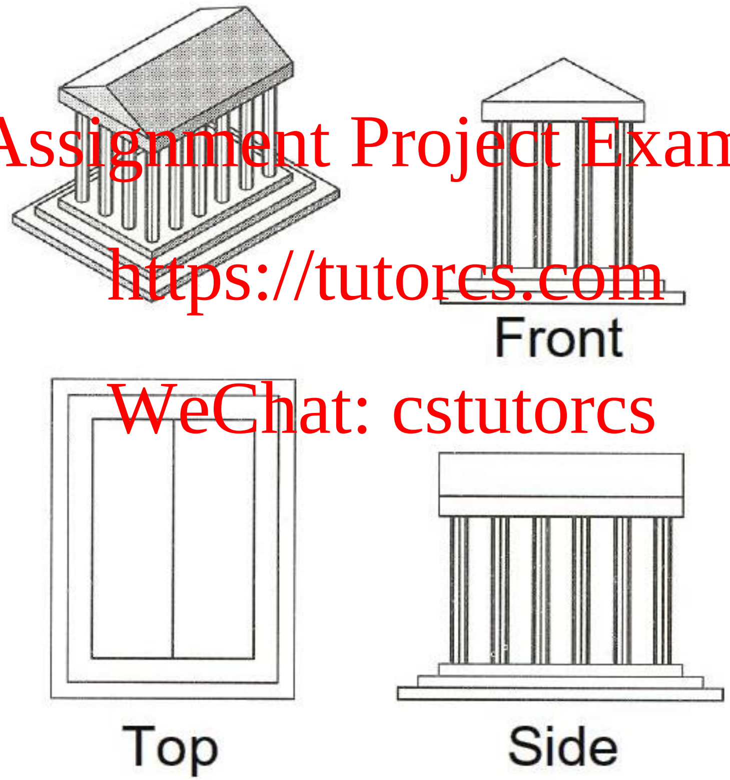
$$L_1 = \frac{1}{\tan\alpha} \quad (\text{for } \alpha \neq 90°)$$

$$L_1 = 0 \quad (\text{for } \alpha = 90°)$$



Orthographic Projection

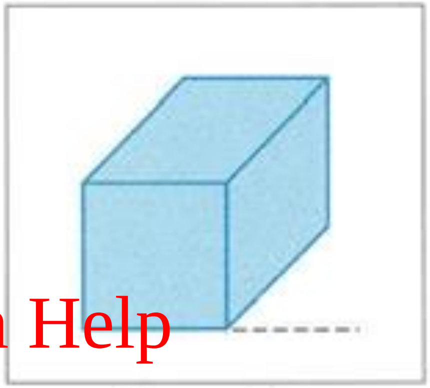
- Direction of projection *orthogonal* to view plane
  - Points with the same (x, y) coordinates will project at the same point on the view plane

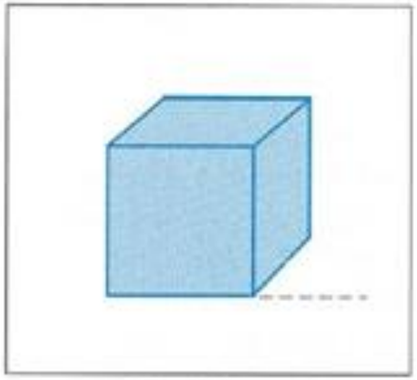


- Applications: for exact scaling the object like CAD etc

Oblique Projection

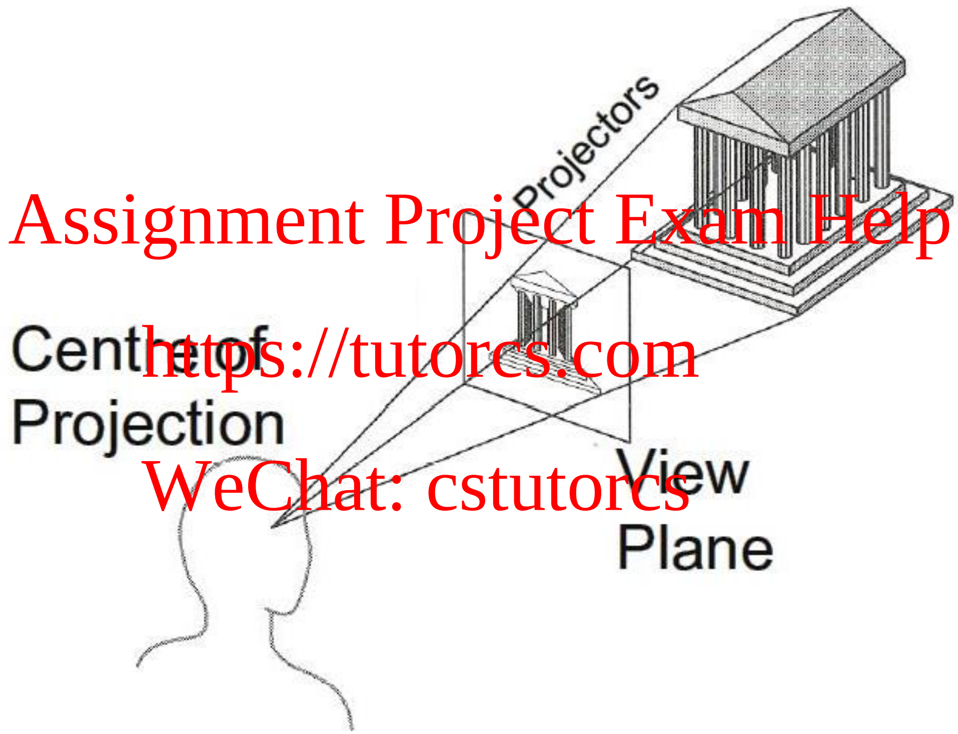
- Direction of projection *not orthogonal* to view plane
  - For cavalier projection ($\alpha = 45°$), two points with the same (x, y) coordinates will keep their distance on the view plane
  - For cabinet projection ($\alpha = 63.4°$), two points with the same (x, y) coordinates will half their distance on the view plane
- Applications: for technical drawing and illustration like in furniture, or architecture, etc.


Cavalier
(DOP at 45°)


Cabinet
(DOP at 63.4°)

Perspective Projection

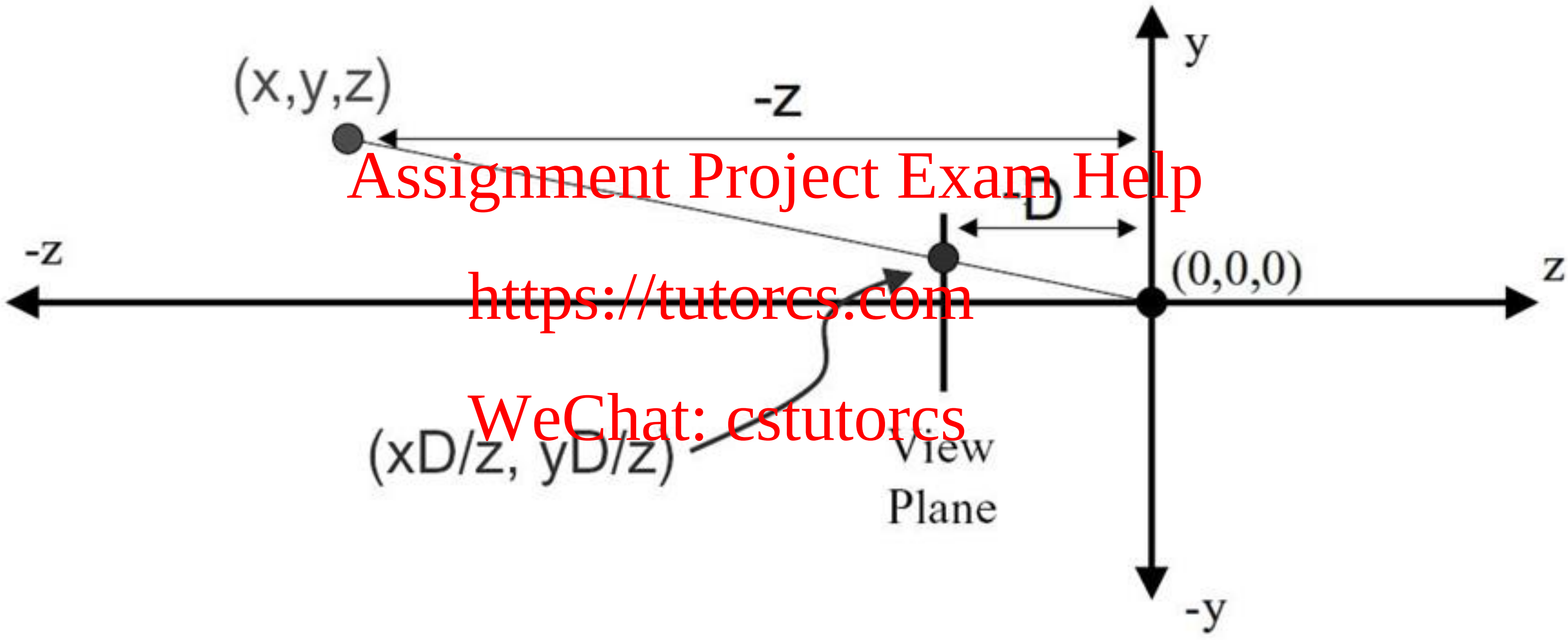
- Map points onto view plane along projectors emanating from centre of projection



- Application : for art drawings, especially for representing large scenes.

Perspective Projection

- Compute 2D coordinates from 3D coordinates using *similar triangles*



$$\frac{y_c}{z_c} = \frac{y_p}{D} \qquad \frac{x_c}{z_c} = \frac{x_p}{D} \qquad \text{for } D < 0$$

Perspective Projection Matrix

- 4×4 homogeneous coordinates matrix representation

$$
\begin{aligned}
x_p &= x_c D / z_c & & & x'_p &= x_c \\
y_p &= y_c D / z_c & &\rightarrow & y'_p &= y_c \\
z_p &= D & & & z'_p &= z_c \\
w_p &= 1 & & & w'_p &= z_c / D
\end{aligned}
$$

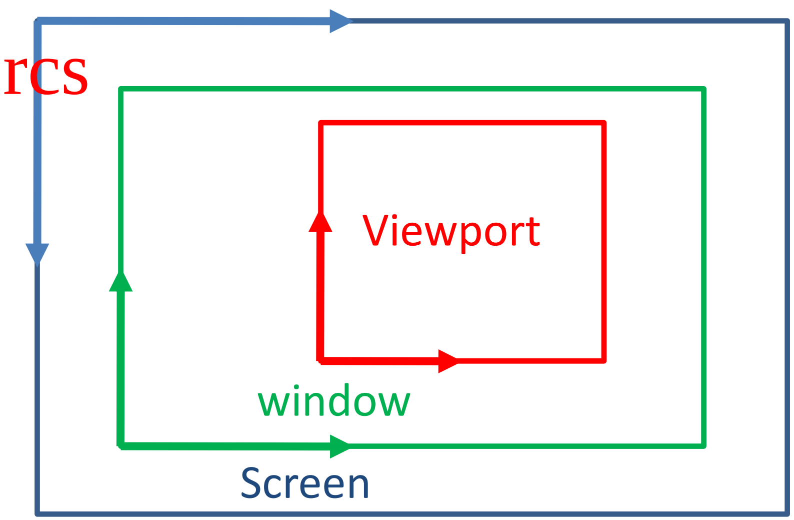
$$
\begin{pmatrix} x_p \\ y_p \\ z_p \\ w_p \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix}
$$

Perspective vs. Parallel Projection

- Perspective projection
  - Size varies inversely with distance – looks realistic
  - Distance and angles are not (in general) preserved
  - Parallel lines do not (in general) remain parallel
- Parallel projection
  - Good for exact measurements
  - Parallel lines remain parallel
  - Angles are not (in general) preserved
  - Less realistic looking

Viewport on Screen

- Coordinate systems on display:
  - Screen coordinate system: Origin at the upper-left corner of the screen, x direction from left to right, and y direction from top to bottom
  - Window coordinate system: Origin at the lower-left corner of the window, x direction from left to right, and y direction from bottom to top
  - Viewport: The rectangular region in the window where the image is drawn. Defined on window coordinate system by (x_0, y_0, w, h)



Viewport Transformation

- The whole image on the view plane are mapped on the whole viewport (by scaling and translating)
- To avoid distortion, the aspect ratio of the viewport should be equal to the aspect ratio of the viewing volume
  - aspect ratio: The ratio of the width to the height of a rectangle area (w/h)

OpenGL Projection

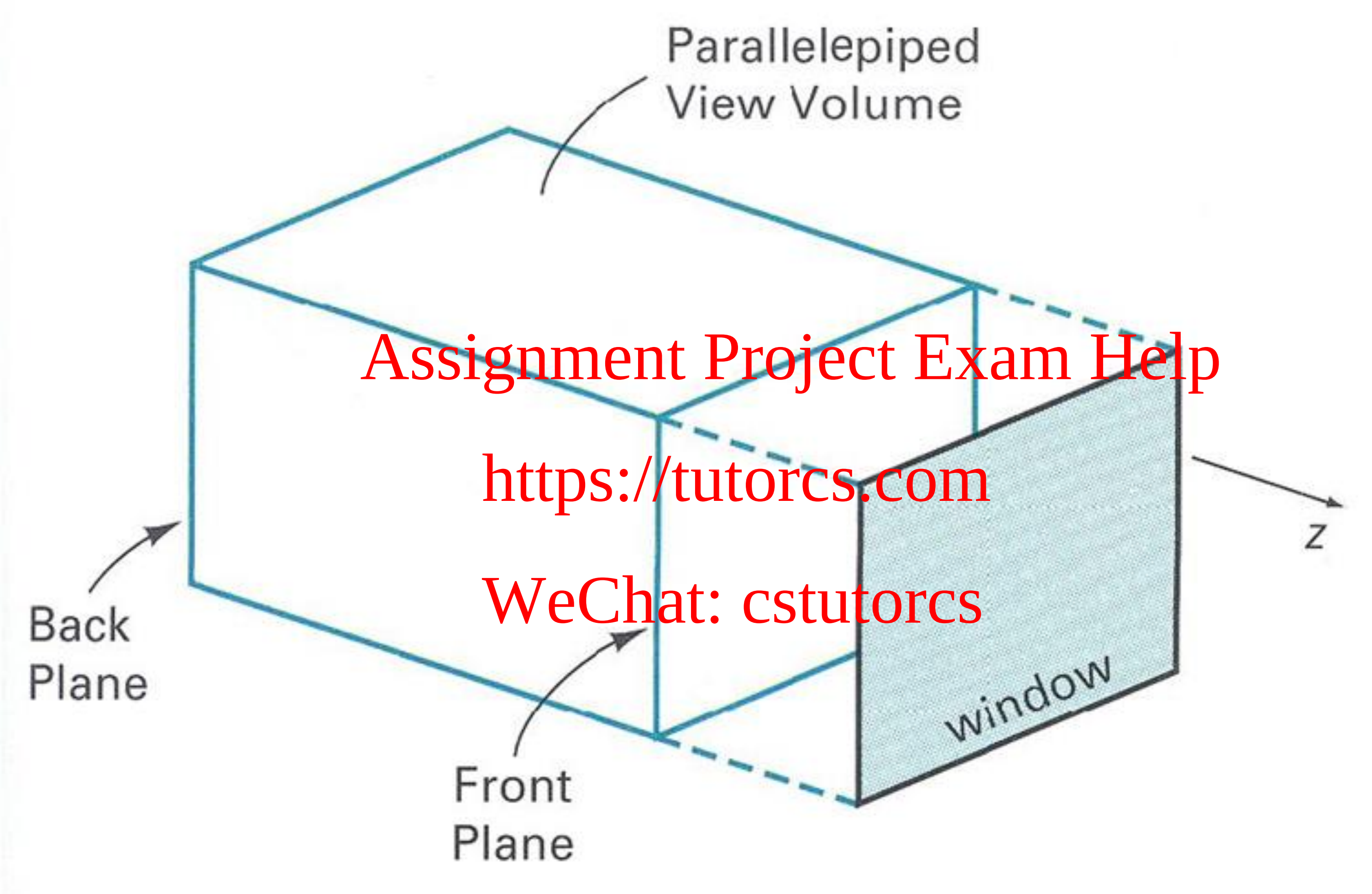
- Actual projection is set by projection matrix
- Projection matrix specifies parallel or perspective projection parameters
- Projection matrix is essentially defined by selecting a viewing volume (the region camera can see)

- Points inside the viewing volume are projected into a cube of edge length 2 (x, y, and z all range from -1 to 1)

  - Depths are maps of the z coordinate to the range [0, 1]

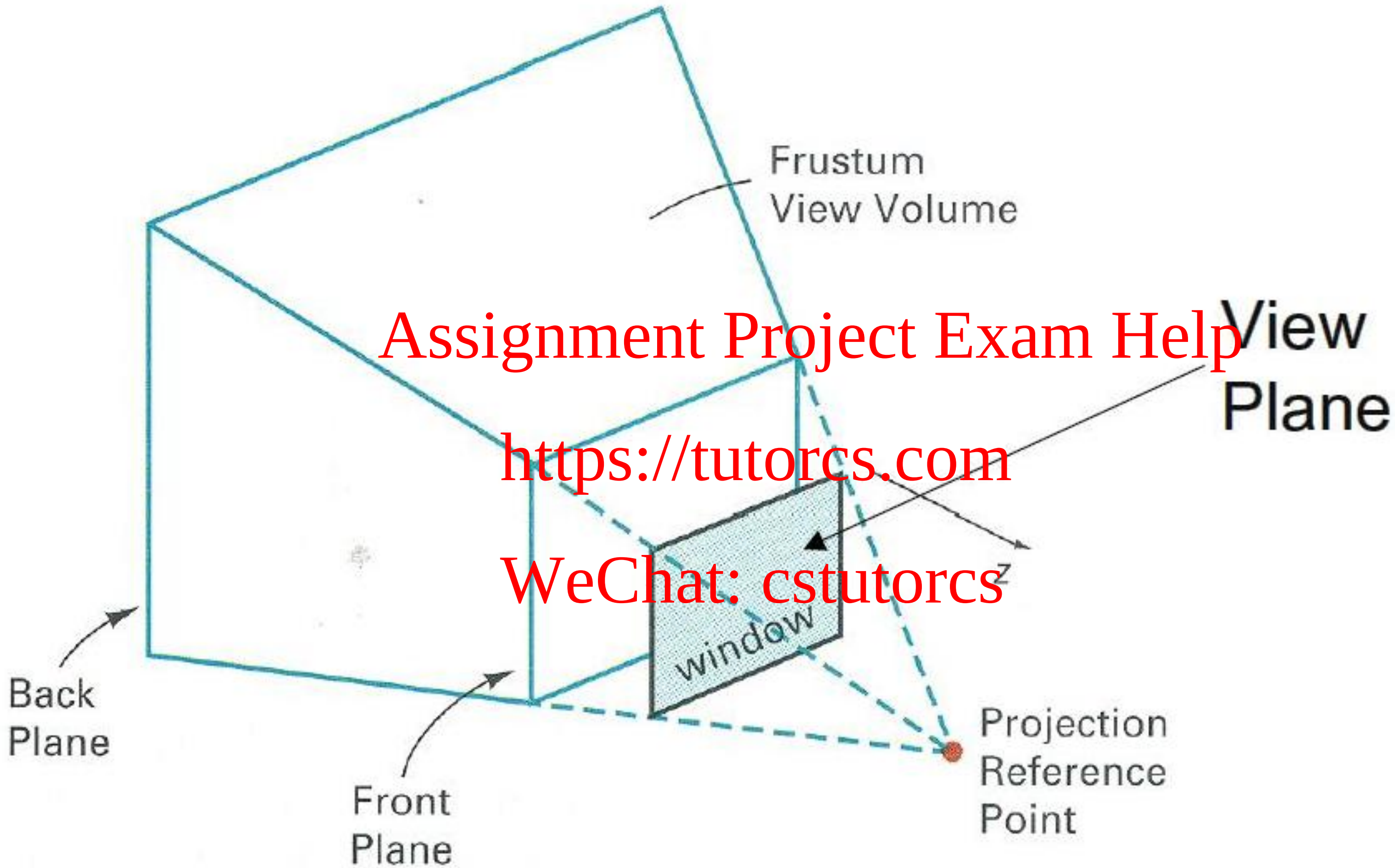
- Orthographic and perspective projections are implemented in class Transform, simulating the projection functions in OpenGL fixed-function pipeline

Parallel Projection Viewing Volume



H&B Figure 12.30

Perspective Projection Viewing Volume



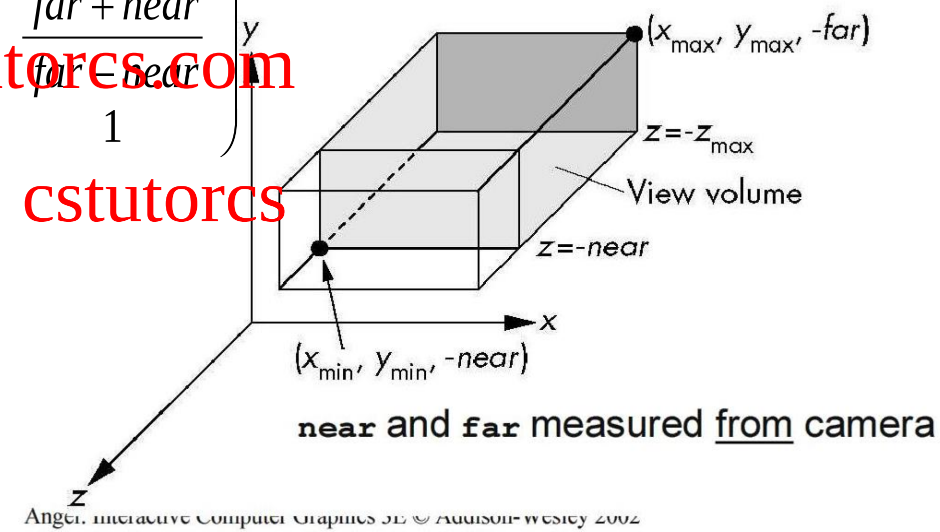
H&B Figure 12.30

Orthographic Viewing in Transform

ortho (xmin, xmax, ymin, ymax, near, far);

- Projection matrix:

$$P = \begin{pmatrix} \dfrac{2}{x_{\max} - x_{\min}} & 0 & 0 & -\dfrac{x_{\max} + x_{\min}}{x_{\max} - x_{\min}} \\ 0 & \dfrac{2}{y_{\max} - y_{\min}} & 0 & -\dfrac{y_{\max} + y_{\min}}{y_{\max} - y_{\min}} \\ 0 & 0 & \dfrac{2}{far - near} & \dfrac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



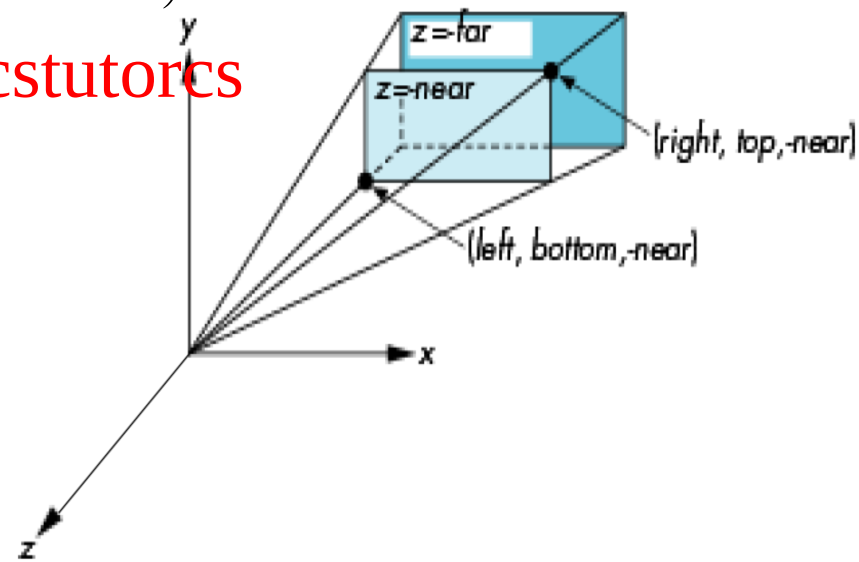
- No oblique projection is implemented

Perspective Viewing in Transform

frustum (xmin, xmax, ymin, ymax, near, far);

- Projection matrix:

$$P = \begin{pmatrix} \frac{2near}{x_{\max} - x_{\min}} & 0 & \frac{x_{\max} + x_{\min}}{x_{\max} - x_{\min}} & 0 \\ 0 & \frac{2near}{y_{\max} - y_{\min}} & \frac{y_{\max} + y_{\min}}{y_{\max} - y_{\min}} & 0 \\ 0 & 0 & \frac{far + near}{far - near} & \frac{2\, far \cdot near}{far - near} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



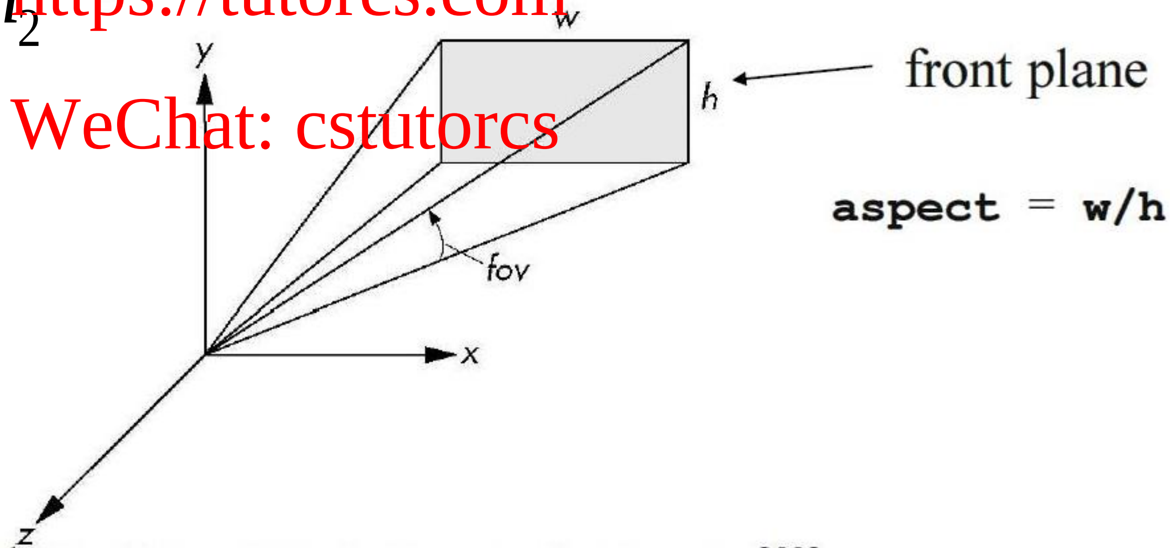
Using Field of View

- frustum not intuitive
- Better interface (for symmetric frustum):

  perspective (fovy, aspect, near, far) =
  frustum (-w2, w2, -h2, h2, near, far);

$$h_2 = near \cdot \tan(fovy/2)$$

$$w_2 = aspect \cdot h_2$$



Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

OpenGL Viewport

glViewport (x, y, width, height);

- Default value (0, 0, winWidth, winHeight)
  - winWidth and winHeight specify the size of the window
- Map points drawn on the view plane into the viewport
  - Coordinate transforming from ([-1,-1] ~[1,1]) on the camera coordinate system to ([x,y] ~[x+width,y+height]) on the window coordinate system
- When combined with perspective(), either
  - glViewport (x, y, width, height);
    perspective(fovy, width/height, near, far);
  - glViewport (x, y, width, width/aspect);
    perspective(fovy, aspect, near, far);
- Similar when combined with ortho()

Summary

- How are world coordinates transformed into camera coordinates? Why is this done?
- What is parallel projection? How is it computed?
- What is perspective projection? How is it computed?