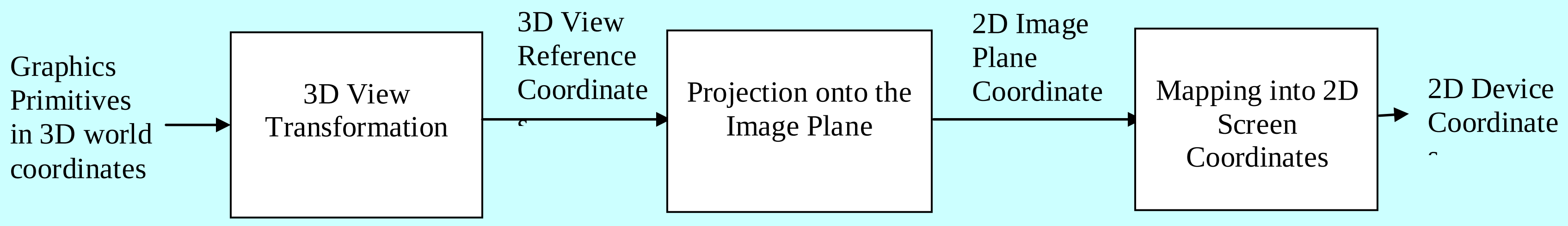
Chapter 6: Viewing in 3D

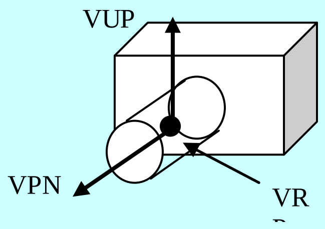
- **Conceptual Model of the 3D Viewing Process:**


Graphics Primitives in 3D world coordinates
3D View Transformation
3D View Reference Coordinate
s
Projection onto the Image Plane
2D Image Plane Coordinate
Mapping into 2D Screen Coordinates
2D Device Coordinate
s

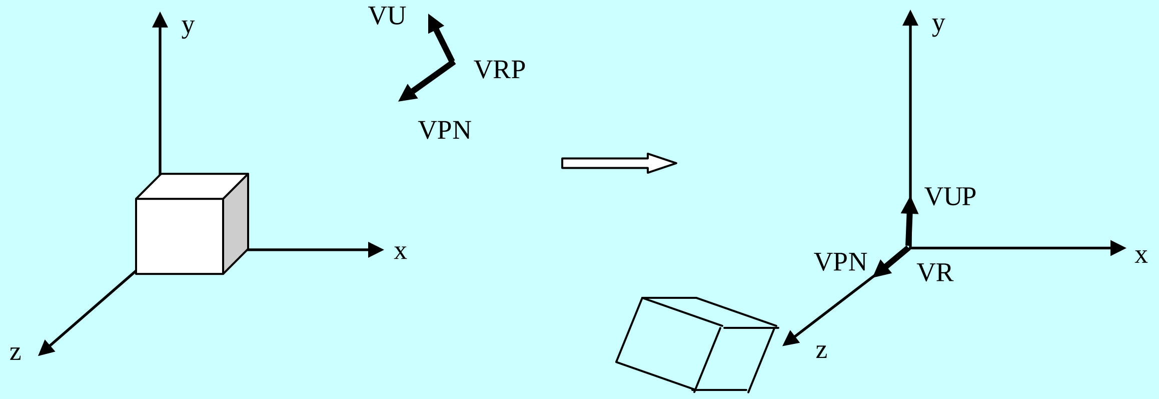
- **Specifying an Arbitrary 3D View**

Given a set of objects in a scene, we need to set up the “camera” at a proper location and proper orientation in order to generate a desired picture. The standard parameters for setting up the “camera” include:

- View Reference Point (VRP): the viewing position;
- View Plane Normal (VPN): a (unit) normal vector perpendicular to the image plane; you may think this as the axis of the camera lens;
- View Up Vector (VUP): the orientation around the axis of the camera lens.



- **View Transformation Matrix**



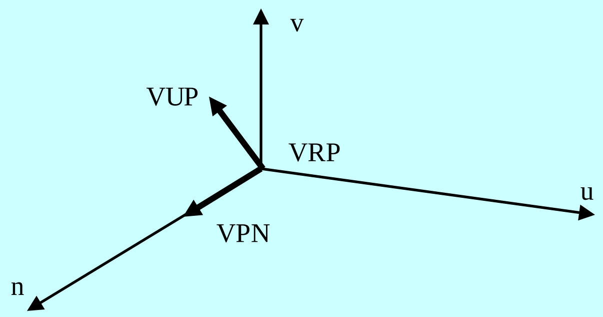
Given a scene and a set of camera parameters, the objective of 3D view transformation is to bring the VRP to the origin, VPN along the positive z direction, and VUP on the y-z plane. The same transformation is applied to all the objects in the scene as well, such that the relative relationship between the camera and the objects remains the same.

You may notice that this transformation is essentially identical to the 3D transformation example given in Chapter 5, with the following relationship between the parameters:

- P1 <-----> VRP
- P2 <-----> VRP + VPN
- P3 <-----> VRP + VUP

Therefore, for a given set of camera parameters, VRP, VPN, and VUP, we first convert them into P1, P2, and P3; then we can construct the view transformation matrix following the steps shown in Chapter 5.

- **3D View Reference Coordinate (VRC) system**



Using VPN and VUP, we can define a local coordinate system - VRC. VRC is generally considered as the camera coordinate system. It is defined as follows (assuming that VPN and VUP are nor co-linear):

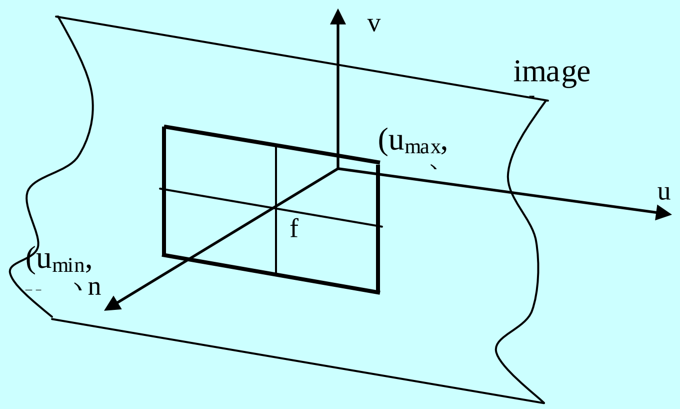
- u = VUP X VPN / |VUP X VPN|
- v = VPN X u / |VPN X u|
- n = VPN / |VPN|

where “X” denotes cross-product. Therefore, VRC is the 3D coordinate system: (u, v, n). Notice that, VUP is usually not required to be perpendicular to VPN (but, cannot be co-linear). VUP lies on v-n plane.

NOTE: Sometimes, we still use (x, y, z) to represent the camera coordinates. This is because after the 3D view transformation, the camera will sit at the origin with a standard orientation. In this case, (u,v,n) aligns with the world coordinate system (x,y,z). However, you must distinguish it from the 3D world coordinates before the view transformation.

- **View Window**

The image plane (or view plane) is perpendicular to axis n (or parallel to u-v plane) and placed at f (the focal length) on the n axis:

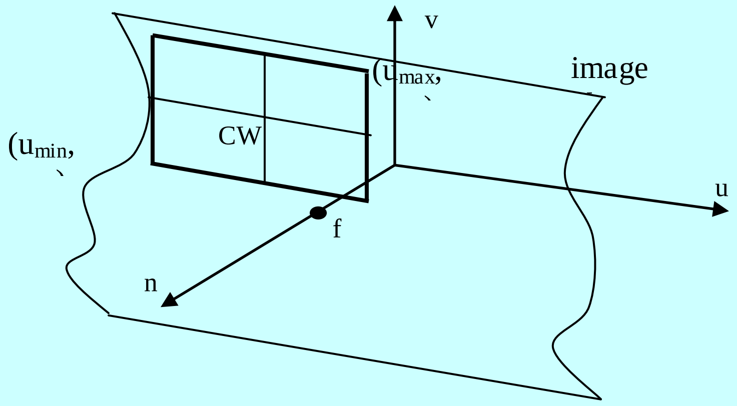


A rectangular window, called view window, is defined on the image plane. The 3D coordinates of the lower-left and upper-right corners of the view window are (u_{min}, v_{min}, f) and (u_{max}, v_{max}, f) respectively. Because the n coordinate is always f on the image plane, we often ignor the last coordinate when write a coordinate on the view window. Therefore, a view window is often specified simply by (u_{min}, v_{min}) and (u_{max}, v_{max}), and the n coordinate is implicitly defined as f.

The purpose of the view window is do clipping. The contents being projected onto the image plane inside the view window will be mapped onto a region of a display device (e.g. a screen, or a printer). Any part of the 3D world that is projected onto the image plane outside the view window will be clipped out.

- **Center of Window (CW)**: the middle point of the view window.

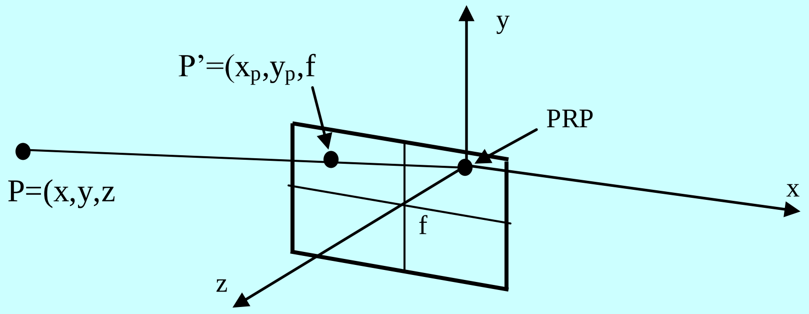
  Note: In most cases, CW is on the n axis. However, it is allowed to be located at any place on the image plane:



Perspective Projections

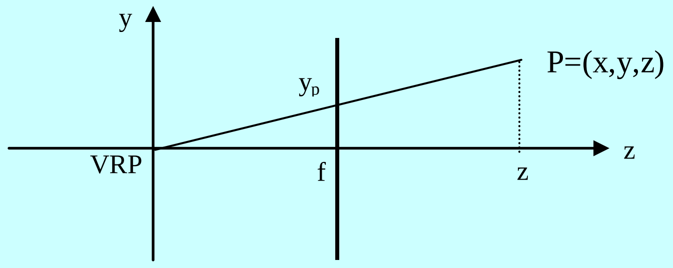
- **Projection Reference Point** (PRP)

  In case of perspective rojection, PRP is the center of projection, i.e. the origin of the VRC:



All projection rays pass through PRP. Given a point P=(x,y,z) in the scene, the ray passing through P and PRP intersects with the image plane at P'=(x_p,y_p,z_p), where z_p=f.

To calculate the project point P', let's look at the following side view of the above diagram:



Consider the two similar triangles: ∇(VRP-f-y_p) and ∇ (VRP-z-P). We have:

$$\frac{y_p}{f} = \frac{y}{z}. \qquad (1.a)$$

Similarly, we also have:

$$\frac{x_p}{f} = \frac{x}{z}. \qquad (1.b)$$

Therefore,

$$\begin{cases} x_p = \dfrac{f \cdot x}{z} = \dfrac{x}{z/f} \\ y_p = \dfrac{f \cdot y}{z} = \dfrac{y}{z/f} \end{cases} \qquad (2)$$

The calculation of (2) can be expressed as a 4X4 matrix, called perspective projection matrix:

$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix}.$$

Example: applying M_{per} to P=(x,y,z) yields:

$$M_{per} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/f \end{bmatrix}.$$

Notice that the resulting vector is a homogeneous coordinate. By normalizing it, i.e. dividing each element by the last element of the homogeneous coordinate, we obtain:

$$\begin{bmatrix} x \\ y \\ z \\ z/f \end{bmatrix} = \begin{bmatrix} \frac{x}{z/f} \\ \frac{y}{z/f} \\ f \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}.$$

Note: In some other graphics textbooks and in some graphics systems, the image plane is placed at z=0, and the center of projection at z=-f. The perspective projection matrix following this definition will be slightly different from the one derived in this textbook.

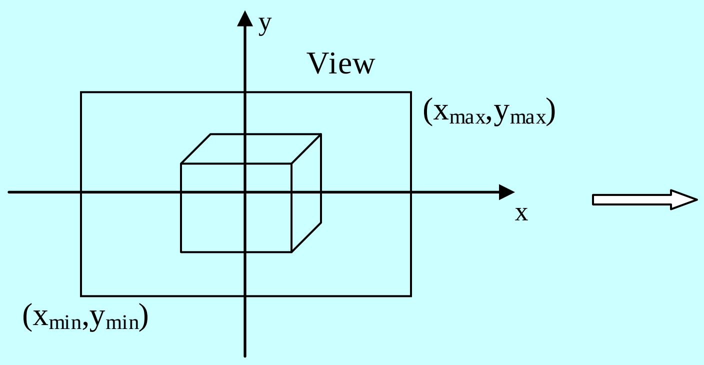
- **View Port Mapping**

  - Graphics primitives are specified in the 3D world coordinate systems.
  - After the 3D view transformation, they are in camera coordinate system now, but it is still a 3D point.
  - After the perspective projection, the point P'=(x_p,y_p,z_p) is still a 3D point in the camera coordinate system. However, z_p=f.
  - To display P' on a screen, P' has to be mapped into a pixel indices in screen (device) coordinates.
  - Furthermore, you may want to map:

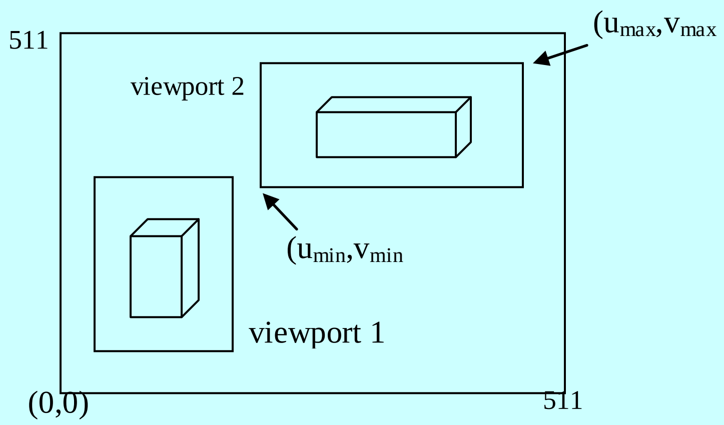
1. P’ to several different devices (e.g. scarren, ploter, etc.); or
2. the view window not to the entire screen, but to a subregion (such as a display window on the screen).

- These considerations lead to a two-step mapping method:
  1. Map the camera coordinates into a device independent coordinates, often called virtual coordinates, first;
  2. Map a virtual coordinates into a desired device coordinates.

- For simplicity (as did in the textbook), we skip the virtual coordinates.



View Window on the Image



Display

This viewport mapping can be accomplished in three steps:

1. Translate the lower-left corner of the View Window - (x_{min},y_{min}) - to the origin: $T(-x_{min}, -y_{min})$;
2. Scale the View Window size to the Viewport size, i.e.

   width: $(x_{max} - x_{min})$ <-----> $(u_{max} - u_{min})$

   height: $(y_{max} - y_{min})$ <-----> $(v_{max} - v_{min})$

   or using a scaling matrix:

$$S(s_x, s_y) = S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}\right).$$

3. Finally, translate the origin to the lower-left corner of the viewport by the translation: $T(u_{\min}, v_{\min})$.

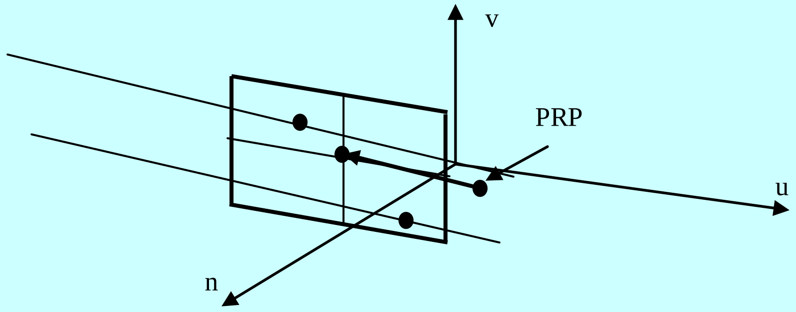
The composite matrix for viewport mapping is:

$$M_{uv} = T(u_{\min}, v_{\min}) \cdot S(s_x, s_y) \cdot T(-x_{\min}, -y_{\min})$$
$$= \cdots = \begin{bmatrix} s_x & 0 & -x_{\min} s_x + u_{\min} \\ 0 & s_y & -y_{\min} s_y + v_{\min} \\ 0 & 0 & 1 \end{bmatrix}.$$

Parallel Projections

- **Direction of Projection** (DOP)

In case of parallel projection, a DOP vector is defined, such that all the projection rays are parallel to it. You directly specify a vector as DOP. However, DOP is often specified by the vector from PRP to CW:

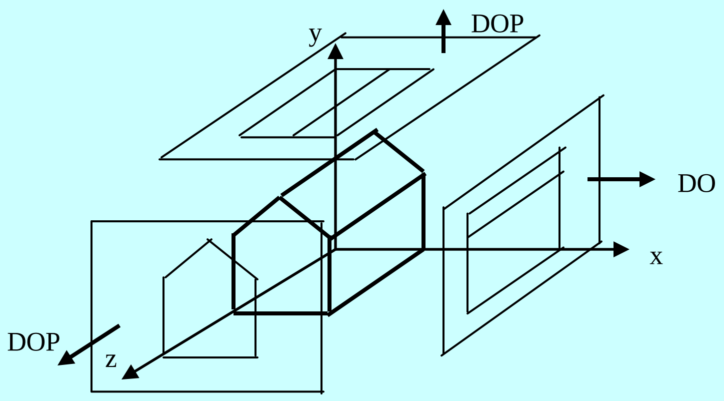


Parallel projections can broadly divided into two categaries:

- Orthographic paralell projection - DOP is perpendicular to the projection (i.e. image) plane;
- Oblique parallel projection - DOP is not perpendicular to the projection (i.e. image) plane.

- **Orthographic Projection**

  A simple, but most useful type, is the **Front**, **Top**, and **Side evaluations** (or views). That is, each projection plane is perpendicular to one of x-, y-, or z-axis. In other words, the DOP is along one the x-, y-, or z-axis:



Each of these projection can be easily obtained by dropping one the coordinate from a 3D point (x,y,z).

Example: **Front projection**: the projection plane is perpendicular to the z-axis. Becaue a project ray is parallel to DOP (i.e. the z-axis in this case), any point, P=(x,y,z), in the scene will have the same (x,y) coordinate on the projection plane, and z is dropped.

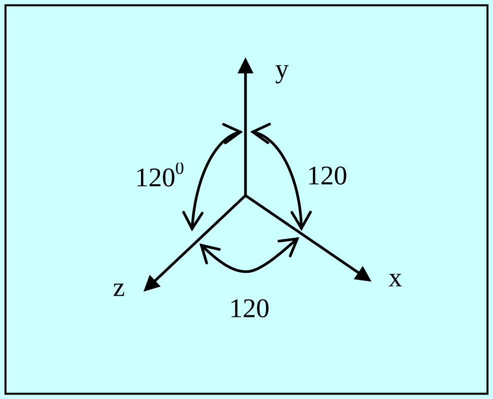
If the projection plane is not perpendicular to anyone of the principal axes, it is called **Axonometric Orthographic Projection**.

Let the DOP (i.e. the normal of the projection plane) be (dx, dy, dz). If the following condition is satisfied:

$$|dx| = |dy| = |dz|,$$

it is called **Isometric Projection,** a particular type of Axonometric Orthographic Projection. It has some useful properties:

- All three axes are equally forshortened on the projection plane;
- The projection of three axes will have equal angles between each other.



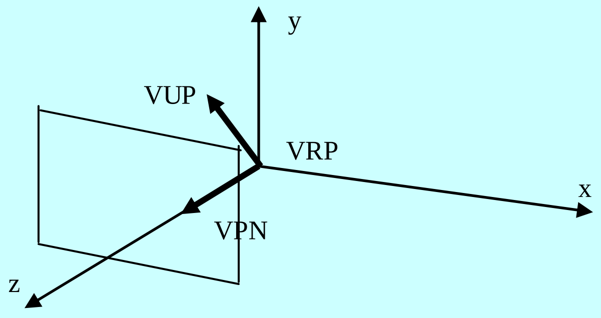
- **Oblique Projections**

The DOP is not perpendicular to the projection plane anymore.

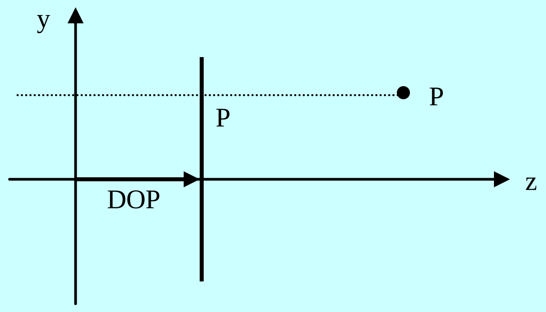
- If the DOP makes a 45^0 angle with the projection plane, it is called **Cavalier projection**.
- If the DOP makes a 63.4^0 angle with the projection plane, it is called **Cabinet projection**.

- **Implementation of Parallel Projections**

The camera model will be the same as before, i.e. VRP, VPN, and VUP. After the standard 3D view transformation, we are at the following situation:

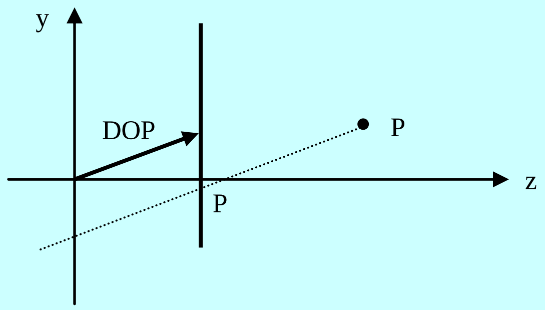


For **Orthographic projections**, the DOP is perpendicular to the image plane, or in other words, it is parallel to z-axis now. Let's look at the side view:



Given a point P = (x,y,z) in the scene, when it is projected onto the image plane, the projection point P' = (x,y). We can see that it can be simply obtained by dropping the z coordinate from P because the projection ray is parallel to the z-axis. Unlike the perspective projection, the projection plane can be placed at any position along z and the projection result will be the same.

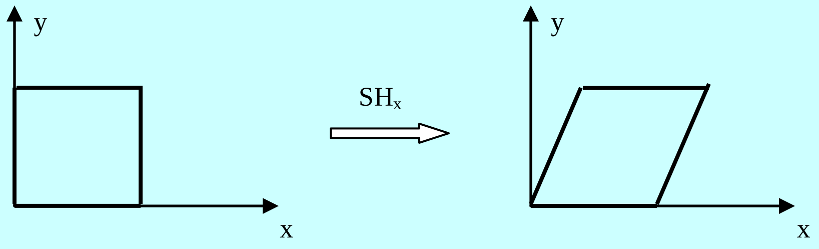
For **Oblique projections**, the DOP will at certain angle (other than 90^0) to the image plane:



We may use a projection ray that is passing through a point P in the scene and parallel to DOP to find the projection point P' on the image plane. However, a standard way is to apply a transformation, called **shear transformation**, to bring DOP onto the z-axis, at the same time maintain the image plane to be perpendicular to the z-axis, such that the situation is reduced to the orthographic situation.

- **Shear Transformations in 2D**:

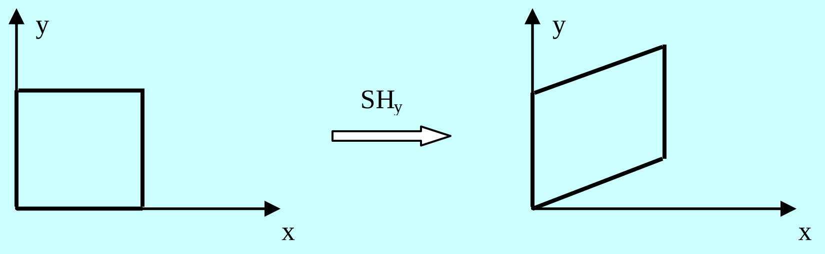
  - **Along x-axis:**



Under shear transformation along x-axis, the y coordinate of any point is not changing. The x coordinate is translated for certain amount depending on the corresponding y coordinate of the point. Shear transformation can be implemented by a 3X3 matrix in homogeneous coordinates:

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad \text{E.g.} \quad SH_x \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + ay \\ y \\ 1 \end{bmatrix}.$$

- **Along y-axis:**



Similarly, we have the shear transformation matrix along y-axis:

$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

- **Shear Transformation in 3D:**

In 3D case, you may perform shear operation in both x and y directions simultaneously. The shear transformation matrix for 3D case is a simple extension from 2D cases. In case, z coordinate is not changing, and x and coordinates will be displaced for certain amounts respectively depending on the corresponding z coordinate of a point.

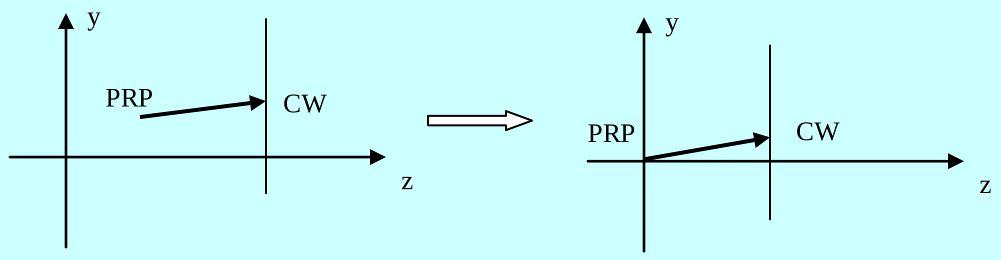
$$SH_{xy} = \begin{bmatrix} 1 & 0 & SH_x & 0 \\ 0 & 1 & SH_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Canonical View Volume and Clipping in 3D

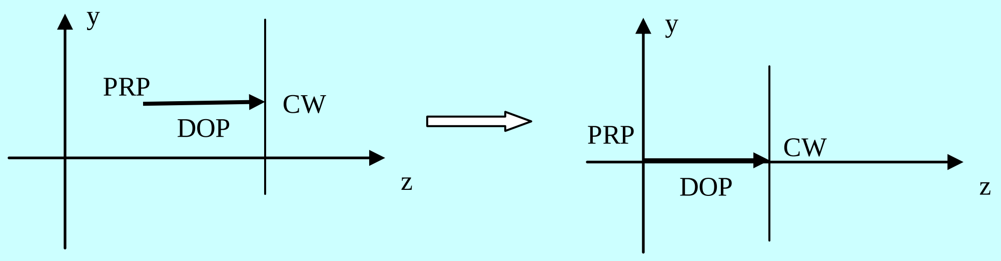
- After 3D view transformation, VRP will be at the origin. In previous discussion, we have simplified the situation to that PRP is at VRP, i.e. PRP is at the origin after the 3D view transformation.

- In general, PRP is NOT necessarily at VRP. So, before any projection, we have to translate the PRP to the origin:

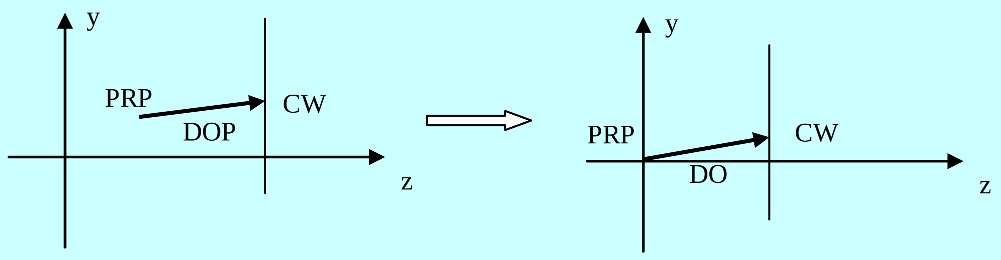
Perspective:



Orthographic:



Oblique:



- After translating PRP to the origin, CW may not be on the z-axis if the view window is defined at an arbitrary position (except the orthographic case). To bring CW onto z-axis, at the same time maintaining the image plane perpendicular to the z-axis, the shear transformation discussed above can be applied. Then, we have the standard situation: PRP is at the origin, and CW is on the z-axis. For both parallel cases, DOP is parallel to z-axis now.

- We have discussed clipping in 2D. The purpose is to ignor any (or portion) of an object that is outside of the view window. This concept can be extended to 3D. That is, you may only want to draw objects that are within certain range of distance. Or in other words, you may want to clip out any object that is outside of this range. This can be specified by a front-clipping-plane F, and a back-clipping-plane B.

- The clipping volume is then defined as:

    - **Perspective**: Planes F and B, plus four planes each of that is passing through the origin and one of the view window boundary respectively (Figure 6.42, page 226).

    - **Parallel**: Planes F and B, plus four planes each of that is parallel to z-axis and passing through one of the view window boundary respectively.

- Finally, by scaling the volume to unit size, we obtain the canonical view volume: Figure 6.43 (a) for the perspective case, and Figure 6.43 (b) for the parallel case, on page 230.

- Clipping in 3D is not performed precisely like the 2D case. In general, a bounding box or bounding sphere is computed for an object. The bounding box (or sphere) is tested against the canonical view volume. If it is entirely outside the clipping volume, it is ignored. Otherwise, the object will be rendered.

------------------------------ END ---------------------------------