TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐHĐN
KHOA CÔNG NGHỆ THÔNG TIN

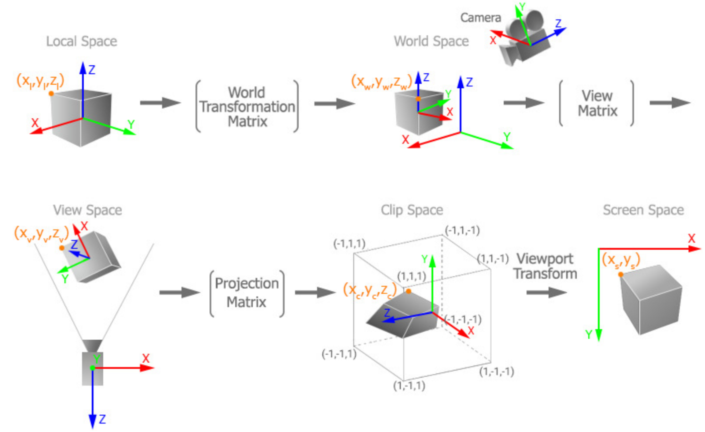
BÀI GIẢNG
ĐỒ HỌA MÁY TÍNH

Lưu hành nội bộ

Đà nẵng, 2023

Chương 2. BIẾN ĐỔI ĐỒ HỌA MÁY TÍNH

Chương 3. CÁC PHÉP BIẾN ĐỔI TRONG ĐỒ HỌA

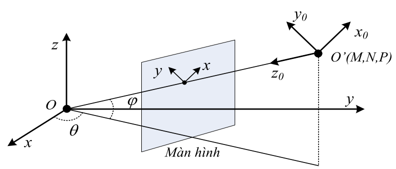


Chương 4. ĐỒ HỌA BA CHIỀU

4.1 Biến đổi hệ quan sát/góc nhìn 3D (VIEW TRANSFORMATION)

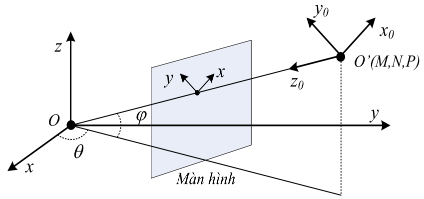
Quan sát một vật thể trong không gian bằng hai cách:

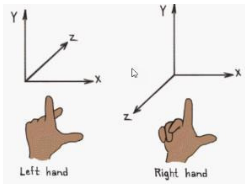
- Điểm nhìn đứng yên, vật thể di động;
- Vật thể đứng yên, điểm nhìn di động => chọn



Nguyên tắc biến đổi hệ quan sát:

- Vật thể được chiếu lên hệ $O(x, y, z)$ theo quy tắc bàn tay phải;
- Mắt phải đặt tại gốc hệ tọa độ $O'(x_0, y_0, z_0)$ theo quy tắc bàn tay trái;
- Mặt phẳng chiếu là mặt phẳng vuông góc với đường thẳng OO';
- Trục z_0 của hệ quan sát $O'(x_0, y_0, z_0)$ phải hướng đến gốc O.
- Biến đổi một điểm $P(x, y, z)$ trong hệ tọa độ $O(x, y, z)$ thành điểm $P'(x_0, y_0, z_0)$ trong hệ tọa độ quan sát $O'(x_0, y_0, z_0)$.
- Chiếu lên tọa độ trên mặt phẳng quan sát (màn hình).



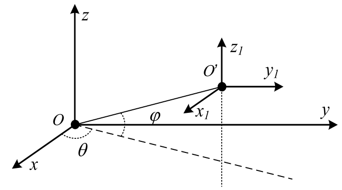


4.1.1 Các bước biến đổi hệ quan sát

Bước 1: Xét trong hệ tọa độ $O(x, y, z)$, điểm O' có tọa độ (M, N, P). Tịnh tiến hệ tọa độ $O(x, y, z)$ theo vector OO' như sau:

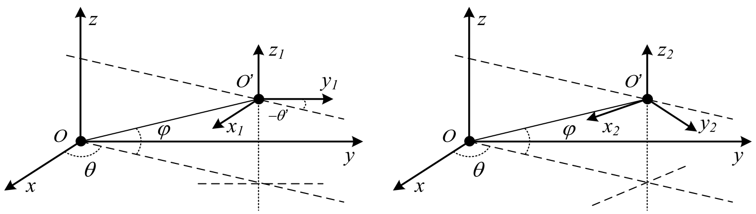
$$[M_T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -M & -N & -P & 1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -r.\cos(\theta)\cos(\varphi) & -r.\sin(\theta)\sin(\varphi) & -r.\sin(\varphi) & 1 \end{bmatrix}$$

☞ Hệ tọa độ **$O(x, y, z)$** biến đổi thành hệ tọa độ **$O_1(x_1, y_1, z_1)$**.



Hình 4.9. Tịnh tiến hệ tọa độ $O(x, y, z)$ theo vector OO'thành hệ tọa độ $O1(x_1, y_1, z_1)$.

Bước 2: Quay hệ **$O_1(x_1, y_1, z_1)$** một góc $-\theta'$ quanh trục z_1 theo chiều kim đồng hồ,
Trong đó $\theta' = 90^0 - \theta$.
Phép quay này làm cho trục âm của y_1 cắt trục z.



Hình 4.10. Quay hệ tọa độ (x_1, y_1, z_1) quanh trục z_1 thành hệ tọa độ $O_2(x_2, y_2, z_2)$

Gọi $[M_R]_z$ là ma trận tổng quát của phép quay quanh trục z_1.

Do quay góc $-\theta'$ quanh hệ trục nên dùng ma trận nghịch đảo $[M_R]_z^{-1}$.

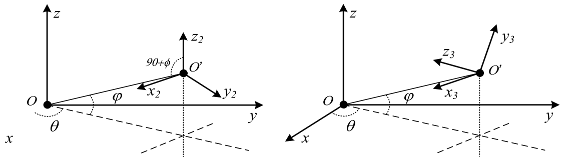
$$[M_R]_z^{-1}=\begin{bmatrix} \cos(a) & \sin(a) & 0 & 0 \\ \sin(a) & \cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thay $a = -\theta'$ vào ma trận trên :

$$[M_R]_z^{-1}=\begin{bmatrix} \sin(\theta) & \cos(\theta) & 0 & 0 \\ -\cos(\theta) & \sin(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Khi đó, hệ tọa độ **$O_1(x_1, y_1, z_1)$** biến đổi thành hệ tọa độ **$O_2(x_2, y_2, z_2)$**.

Bước 3: Quay hệ **$O_2(x_2, y_2, z_2)$** một góc ($90^0 + \varphi$) quanh trục x_2.
Phép biến đổi này sẽ làm cho trục z_2 hướng đến gốc tọa độ O.



Hình 4.11. Quay hệ tọa độ (x_2, y_2, z_2) quanh trục x_2

$$[M_R]_x^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(a) & -sin(a) & 0 \\ 0 & sin(a) & cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -sin(\varphi) & -cos(\varphi) & 0 \\ 0 & cos(\varphi) & -sin(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Thay góc $a = 90^o + \varphi$

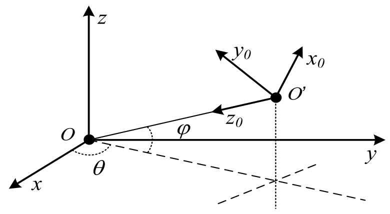
>> Hệ tọa độ **$O_2(x_2, y_2, z_2)$** biến đổi thành hệ tọa độ **$O_3(x_3, y_3, z_3)$.**

Bước 3: Biến đổi hệ tọa độ **$O_3(x_3, y_3, z_3)$** thành hệ gián tiếp theo quy tắc bàn tay trái.

Thực hiện đổi hướng trục x_3 bằng cách đổi dấu các phần tử ứng với các hoành độ trong ma trận biến đổi. Ta nhận được ma trận:

$$D = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Khi đó, hệ **$O_3(x_3, y_3, z_3)$** biến đổi thành hệ **$O(x_0, y_0, z_0)$**.



Hình 4.12. Biến đổi hệ $O_3(x_3, y_3, z_3)$ thành hệ quan sát Camera

- Điểm $P'(x_0, y_0, z_0, 1)$ trong hệ tọa độ quan sát $O'(x_0, y_0, z_0, 1)$ được biểu diễn thông qua điểm $P(x, y, z, 1)$ trong hệ tọa độ $O(x, y, z, 1)$ dựa theo thứ tự bốn phép biến đổi:

$$(x_0, y_0, z_0, 1) = (x, y, z, 1).A.B.C.D = (x, y, z, 1).M$$

$$M = \begin{bmatrix} -sin(\theta) & -cos(\theta).sin(\varphi) & -cos(\theta).cos(\varphi) & 0 \\ cos(\theta) & -sin(\theta).sin(\varphi) & -sin(\theta).cos(\varphi) & 0 \\ 0 & cos(\varphi) & -sin(\varphi) & 0 \\ 0 & 0 & r & 1 \end{bmatrix}$$

Vậy:

$$x_0 = -x.sin(\theta) + y.cos(\theta)$$
$$y_0 = -x.cos(\theta).sin(\varphi) - y.sin(\theta).sin(\varphi) + z.cos(\varphi)$$
$$z_0 = -x.cos(\theta).cos(\varphi) - y.sin(\theta).cos(\varphi) - z.sin(\varphi) + r.$$

4.2 Building a transformation Matrix

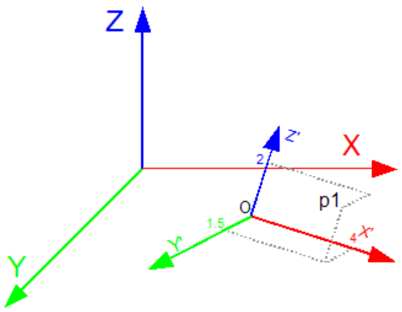
- Local Coordinate System
- Calculation for a11
- Calculation for a12
- Inverse matrix
- Adjugated matrix
- Determinant of A , or |A|
- Inverse matrix result

4.2.1 *Local Coordinate System*

We need to use a new (local) axis :

- ***Camera transform*** : The camera defines a new local axis, with the location of the camera as a new origin, the view direction of the camera as Z axis (corresponds to the Z-buffer)
- ***Normal mapping*** : Incoming light (a vector) is transformed to the local axis of one triangle of the model. This makes it possible to use the normal map texture for the light calculation.

We want to define a matrix for the following situation (Fig).



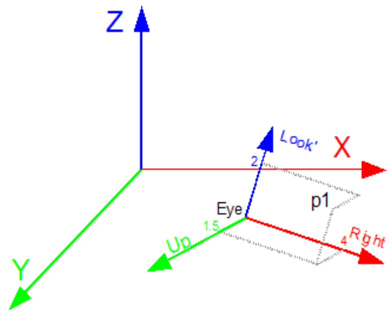
We need to calculate this matrix for a camera transformation, we could also use another notation for the new axises :

Given a point P_1 with coordinates in the world axis [x, y, z] and we wish to know the coordinates of this point in the new (local) axis [x', y' , z'].

Note that the 3 vector ***Right***, ***Up*** and ***Look*** are vectors expressed in the World Coordinate System.

The ***Eye*** position is also a coordinate in the World Coordinate System.

The new coordinate system is an orthogonal coordinate system.



We can write the following :

$$|x \quad y \quad z| = x' * \overrightarrow{Right} + y' * \overrightarrow{Up} + z' * \overrightarrow{Look} + Eye$$

Right, Up and Look are vector (3 components) so we can write the previous equation as a matrix :

$$\lfloor x \quad y \quad z \quad 1 \rfloor = \lfloor x' \quad y' \quad z' \quad 1 \rfloor \times \begin{bmatrix} Right_x & Right_y & Right_z & 0 \\ Up_x & Up_y & Up_z & 0 \\ Look_x & Look_y & Look_z & 0 \\ Eye.x & Eye.y & Eye.z & 1 \end{bmatrix}$$

The following elements are expressed in the original coordinate system (**World Coordinate Syste**m) :

- *x, y, z*
- *Right, Up, Look*
- *Eye*

The coordinate ***(x' y' z')*** is the new coordinate, expressed in the **Local Coordinate System**.

To calculate a point ***p(x', y', z')*** we need to use the properties of inverse matrices :

$$v = p * A$$
$$\rightarrow v * A^{-1} = p * A * A^{-1}$$
$$\rightarrow v * A^{-1} = p$$

Using the properties of vector cross products to calculate an inverse matrix.

The inverse of a 4x4 matrix A is given by :

$$A^{-1} = \frac{\begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix}}{|A|}$$

or : A^{-1} = adj (A) / det (A)

The calculation for the adjugated matrix is shown for 2 elements (a_{11} and a_{12}).

- *Calculation for a_{11}:* To calculate a_{11} we need to calculate the value for a determinant in the original matrix. De determinant is formed by leaving out row 1 and column 1 from the original matrix.

determinant

$$A^{-1}=\frac{\begin{bmatrix} \boxed{a_{11}} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix}}{|A|} \qquad A=\begin{bmatrix} Right_x & Right_y & Right_z & 0 \\ Up_x & Up_y & Up_z & 0 \\ Look_x & Look_y & Look_z & 0 \\ Eye_x & Eye_y & Eye_z & 1 \end{bmatrix}$$

a_{11} is :

$$a_{11}=\begin{vmatrix} Up_y & Up_z & 0 \\ Look_y & Look_z & 0 \\ Eye_y & Eye_z & 1 \end{vmatrix}$$

>> Calculate the determinant for a 3x3 matrix

$a_{11} = Up_y * Look_z - Up_z * Look_y$

- Calculation for a_{12}: a_{12} is the element in the **second** row and **first** column. The determinant is formed by removing the **first** row and **second** column from the original matrix :

rij 1 kolom 2 wegnemen

$$A^{-1}=\frac{\begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ \boxed{a_{12}} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix}}{|A|} \qquad A=\begin{bmatrix} Right_x & Right_y & Right_z & 0 \\ Up_x & Up_y & Up_z & 0 \\ Look_x & Look_y & Look_z & 0 \\ Eye_x & Eye_y & Eye_z & 1 \end{bmatrix}$$

a_{12} is :

$$a_{12}=(-1)^{1+2}\begin{vmatrix} Up_x & Up_z & 0 \\ Look_x & Look_z & 0 \\ Eye_x & Eye_z & 1 \end{vmatrix}$$

$$a_{12}=Up_z * Look_x - Up_x * Look_z$$

- **Inverse matrix**

Adjugated matrix - Applying the formula to every cell of the adjugated matrix yields :

$$\frac{\begin{bmatrix} Up_y*Look_z-Up_z*Look_y & -\begin{vmatrix} Right_y & Right_z & 0 \\ Look_y & Look_z & 0 \\ Eye_y & Eye_z & 1 \end{vmatrix} & \begin{vmatrix} Right_y & Right_z & 0 \\ Up_y & Up_z & 0 \\ Eye_y & Eye_z & 1 \end{vmatrix} & -\begin{vmatrix} Right_y & Right_z & 0 \\ Up_y & Up_z & 0 \\ Look_y & Look_z & 0 \end{vmatrix} \\ Up_z*Look_x-Up_x*Look_z & \begin{vmatrix} Right_x & Right_z & 0 \\ Look_x & Look_z & 0 \\ Eye_x & Eye_z & 1 \end{vmatrix} & -\begin{vmatrix} Right_x & Right_z & 0 \\ Up_x & Up_z & 0 \\ Eye_x & Eye_z & 1 \end{vmatrix} & \begin{vmatrix} Right_x & Right_z & 0 \\ Up_x & Up_z & 0 \\ Look_x & Look_z & 0 \end{vmatrix} \\ \begin{vmatrix} Up_x & Up_y & 0 \\ Look_x & Look_y & 0 \\ Eye_x & Eye_y & 1 \end{vmatrix} & -\begin{vmatrix} Right_x & Right_y & 0 \\ Look_x & Look_y & 0 \\ Eye_x & Eye_y & 1 \end{vmatrix} & \begin{vmatrix} Right_x & Right_y & 0 \\ Up_x & Up_y & 0 \\ Eye_x & Eye_y & 1 \end{vmatrix} & -\begin{vmatrix} Right_x & Right_y & 0 \\ Up_x & Up_y & 0 \\ Look_x & Look_y & 0 \end{vmatrix} \\ -\begin{vmatrix} Up_x & Up_y & Up_z \\ Look_x & Look_y & Look_z \\ Eye_x & Eye_y & Eye_z \end{vmatrix} & \begin{vmatrix} Right_x & Right_y & Right_z \\ Look_x & Look_y & Look_z \\ Eye_x & Eye_y & Eye_z \end{vmatrix} & -\begin{vmatrix} Right_x & Right_y & Right_z \\ Up_x & Up_y & Up_z \\ Eye_x & Eye_y & Eye_z \end{vmatrix} & \begin{vmatrix} Right_x & Right_y & Right_z \\ Up_x & Up_y & Up_z \\ Look_x & Look_y & Look_z \end{vmatrix} \end{bmatrix}}{|A|}$$

or :

$$\frac{\begin{bmatrix} Up_y*Look_z-Up_z*Look_y & Right_z*Look_y-Right_y*Look_z & Right_y*Up_z-Right_z*Up_y & 0 \\ Up_z*Look_x-Up_x*Look_z & Right_x*Look_z-Right_z*Look_x & Right_z*Up_x-Right_x*Up_z & 0 \\ Up_x*Look_y-Up_y*Look_x & Right_y*Look_x-Right_x*Look_y & Right_x*Up_y-Right_y*Up_x & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}}{|A|}$$

This matrix can be simplified by using the properties of orthogonal vectors.

The cross product of 2 of the axises has the third axis as a result. So :

$$\overrightarrow{Right}\times\overrightarrow{Up}=\overrightarrow{Look}$$
$$\overrightarrow{Up}\times\overrightarrow{Look}=\overrightarrow{Right}$$
$$\overrightarrow{Look}\times\overrightarrow{Right}=\overrightarrow{Up}$$

A number of possible simplifications (derived from the cross product)

$Right_x = Up_y * Look_z - Up_z * Look_y$

$Up_y = Right_x * Look_z - Right_z * Look_x$

...

This simplifies the matrix to :

$$A^{-1}=\frac{\begin{bmatrix} Right_x & Up_x & Look_x & 0 \\ Right_y & Up_y & Look_y & 0 \\ Right_z & Up_z & Look_z & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}}{|A|}$$

As an example T_x is calculated , T_y and Tz are similar.

T_y and T_z are calculated as :

$$T_y=-\overrightarrow{Eye}\cdot\overrightarrow{Up}$$
$$T_z=-\overrightarrow{Eye}\cdot\overrightarrow{Look}$$

Again vector cross products are used to simplifiy the equations :

$$T_x=-\begin{vmatrix} Up_x & Up_y & Up_z \\ Look_x & Look_y & Look_z \\ Eye_x & Eye_y & Eye_z \end{vmatrix}$$

$$T_x=\begin{matrix} -Eye_x*(Up_y*Look_z-Up_z*Look_y) \\ -Eye_y*(Up_z*Look_x-Up_x*Look_z) \\ -Eye_z*(Up_x*Look_z-Up_z*Look_x) \end{matrix}$$

$$T_x=-Eye_x*Right_x-Eye_y*Right_y-Eye_z*Right_z$$

$$T_x=-\overrightarrow{Eye}\cdot\overrightarrow{Right}$$

- Determinant of A , or |A|

The last step is the calculation of the determinant of A :

$$|A|=\begin{vmatrix} Right_x & Right_y & Right_z & 0 \\ Up_x & Up_y & Up_z & 0 \\ Look_x & Look_y & Look_z & 0 \\ Eye_x & Eye_y & Eye_z & 1 \end{vmatrix}$$

$$|A|=1*\begin{vmatrix} Right_x & Right_y & Right_z \\ Up_x & Up_y & Up_z \\ Look_x & Look_y & Look_z \end{vmatrix}$$

$$|A|=\begin{matrix} +Right_x*(Up_y*Look_z-Up_z*Look_y) \\ +Right_y*(Up_z*Look_x-Up_x*Look_z) \\ +Right_z*(Up_x*Look_y-Up_y*Look_x) \end{matrix}$$

>>This is a property of orthogonal vectors of length 1.

$$|A|=\begin{matrix} +Right_x*Right_x \\ +Right_y*Right_y \\ +Right_z*Right_z \end{matrix}$$

$$|A|=\|\overrightarrow{Right}\|^2=1$$

- **Inverse matrix result**

Finally we have a matrix that can transform a vertex from the World Coordinate System to the new Local Coordinate System :

$$A^{-1}=\begin{bmatrix} Right_x & Up_x & Look_x & 0 \\ Right_y & Up_y & Look_y & 0 \\ Right_z & Up_z & Look_z & 0 \\ -\overrightarrow{Eye}\cdot\overrightarrow{Right} & -\overrightarrow{Eye}\cdot\overrightarrow{Up} & -\overrightarrow{Eye}\cdot\overrightarrow{Look} & 1 \end{bmatrix}$$

4.2.2 Example of View Transformation

Let (1, 1, 1) be the view reference point.

Let (0, 0, 0) be the look-at point.

Let (0, 1, 0) be the up vector.

Translate	Rotate about Oy	Rotate about Ox
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{2}/\sqrt{3} & 1/\sqrt{3} & 0 \\ 0 & -1/\sqrt{3} & \sqrt{2}/\sqrt{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotate about z: Identity

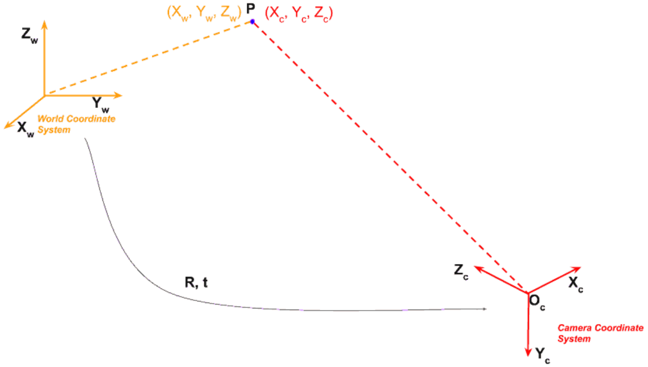
Reflection:

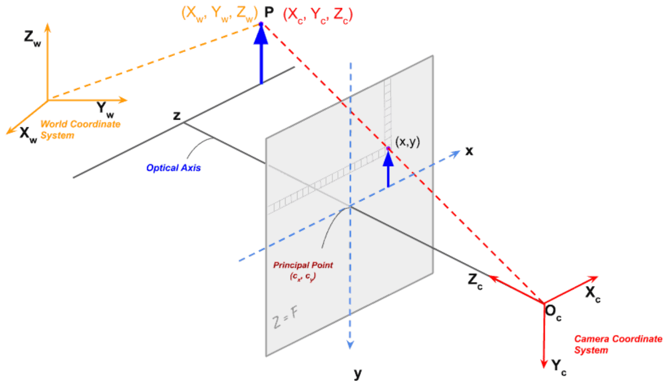
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

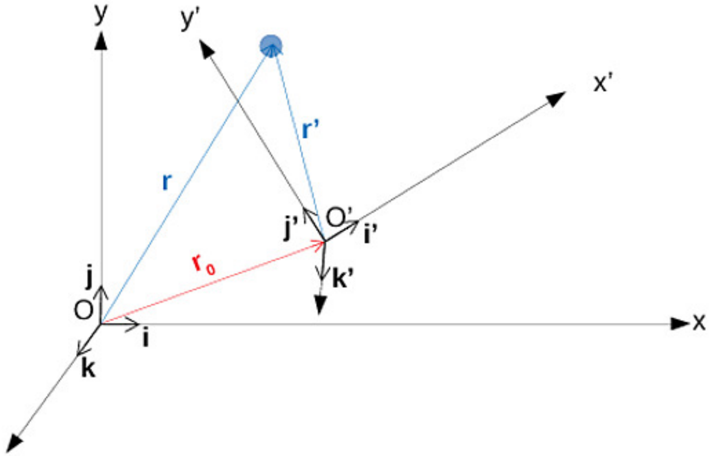
Composite transform V:

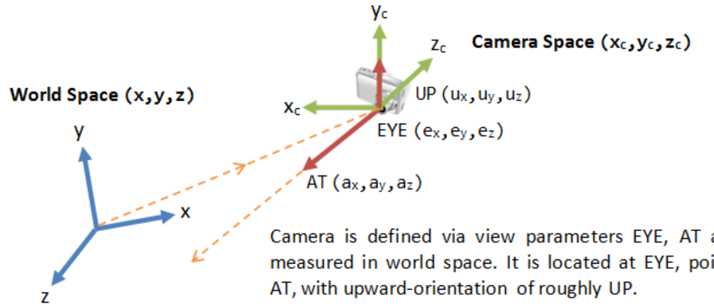
$$\begin{bmatrix} \sqrt{2}/2 & -1/\sqrt{6} & -1/\sqrt{3} & 0 \\ 0 & 2/\sqrt{6} & -1/\sqrt{3} & 0 \\ -\sqrt{2}/2 & -1/\sqrt{6} & -1/\sqrt{3} & 0 \\ 0 & 0 & \sqrt{3} & 1 \end{bmatrix}$$

4.2.3 Camera Look-At setup







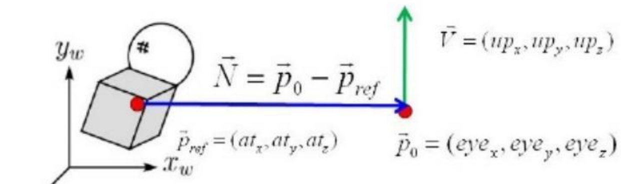


Camera is defined via view parameters EYE, AT and UP, measured in world space. It is located at EYE, pointing at AT, with upward-orientation of roughly UP.

In the Camera space, camera is located at origin, pointing at $-z_c$, with upward-orientation of y_c.
z_c is opposite of AT, y_c is roughly UP.

Mapping from world to eye coordinates

gluLookAt ($eye_x, eye_y, eye_z, at_x, at_y, at_z, up_x, up_y, up_z$)



$$\vec{n} = \frac{\vec{N}}{|\vec{N}|} \qquad \vec{u} = \frac{\vec{V} \times \vec{n}}{|\vec{V}|} \qquad \vec{v} = \vec{n} \times \vec{u}$$

H&B equation (10-1)

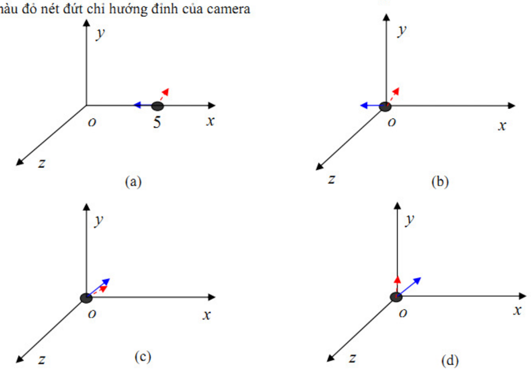
Ví dụ 2.3: Lệnh gluLookAt(5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, -1.0); được thay thế tương đương bởi các câu lệnh sau

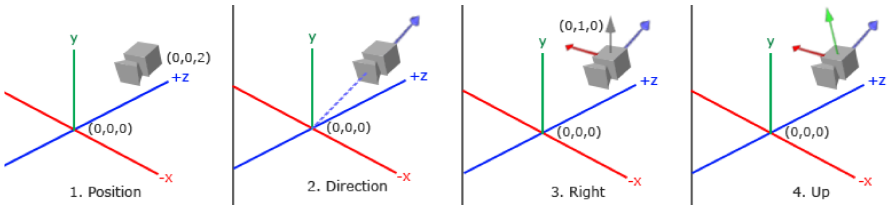
`glRotatef(45, 0.0, 0.0, 1.0);`*//Quay một góc 45 độ quanh trục oz đưa hướng đỉnh camera về hướng dương trục oy*

`glRotatef(-90, 0.0, 1.0, 0.0);`*//Quay một góc -90 độ quanh trục oy đưa hướng nhìn camera về hướng âm trục oz*

`glTranslatef(-5.0, 0.0, 0.0);`*//Đưa camera về gốc tọa độ*

Chú ý: Thứ tự thực hiện các lệnh theo thứ tự ngược lại thứ tự của các câu lệnh. Quá trình được minh họa trên hình 2.7, mũi tên màu xanh chỉ hướng nhìn của camera và mũi tên màu đỏ nét đứt chỉ hướng đỉnh của camera



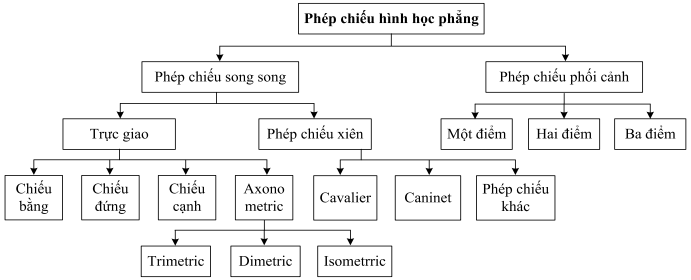


- Ví dụ : OpenGL Example

```
void SetUpViewing()
{
        // The viewport isn't a matrix, it's just state...
        glViewport( 0, 0, window_width, window_height );

        // Set up camera->screen transformation first
        glMatrixMode( GL_PROJECTION );
        glLoadIdentity();
        gluPerspective( 60, 1, 1, 1000 ); // fov, aspect, near, far


        // Set up the model->camera transformation
        glMatrixMode( GL_MODELVIEW );
        gluLookAt( 3, 3, 2,   // eye point
                   0, 0, 0,   // look at point
                   0, 0, 1 ); // up vector
        glRotatef( theta, 0, 0, 1 ); // rotate the model
        glScalef( zoom, zoom, zoom ); // scale the model
}
```

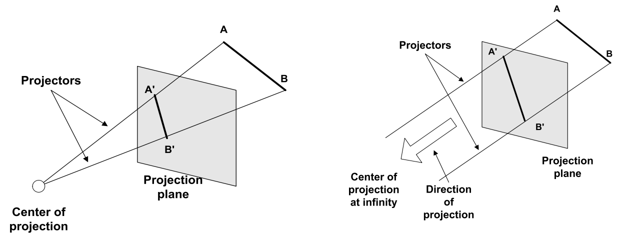
4.3 Phép chiếu (PROJECTION TRANSFORMATION)

- *Phép chiếu* dùng để chiếu các đối tượng *3D* lên bề mặt *2D*.
- *Hình chiếu* là ảnh của một đối tượng trên một mặt phẳng chiếu.
- Phân loại các phép chiếu (Hình 4.1):



Hình 4.1. Phân loại các phép chiếu hình học phẳng

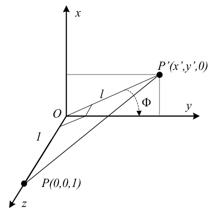
- Phép chiếu song song: tâm chiếu đặt ở vô cực sao cho các tia chiếu song song với nhau;
- Phép chiếu phối cảnh: tâm chiếu đặt cách mặt phẳng chiếu một khoảng cách hữu hạn.



4.3.1 Phép chiếu song song

4.3.1.1 Phép chiếu xiên (Oblique Projection)

- Các tia chiếu song song từ tâm chiếu ở ∞ tạo với mặt phẳng chiếu một góc chiếu xiên.
- Khoảng cách *l*: hệ số rút ngắn của mọi đường thẳng *z = 0*
- φ: là góc giữa hình chiếu và trục nằm ngang:

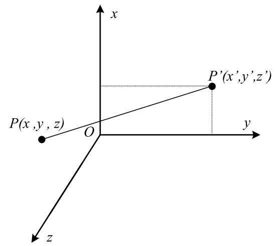


Hình 4.2. Phép chiếu xiên một điểm lên Z = 0

Ma trận phép chiếu xiên:

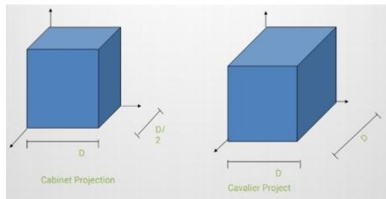
$$[M_{ISO}] = [M_{TILT}].\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\varphi & l\sin\varphi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Phép chiếu xiên (Oblique Projection)



Hình 4.2. Phép chiếu xiên một điểm lên Z = 0

$$x' = l.\cos\varphi$$
$$y' = l.\sin\varphi$$

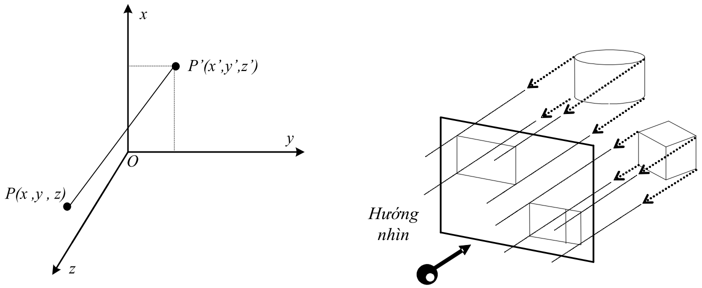


- *l = 1* : các đường thẳng vuông góc với mặt phẳng chiếu sẽ giữ nguyên độ dài thật của chúng => phép chiếu xiên đều (*cavalier*).
- *l = ½* : chiều dài hình chiếu của các đường thẳng vuông góc với mặt phẳng chiếu sẽ bằng ½ độ dài thật của nó => phép chiếu cân (*cabinet*). Giá trị φ thường từ 30^0 đến 45^0.

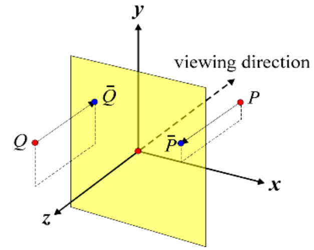
4.3.1.2 Phép chiếu trực giao (Orthographic projection)

Mặt phẳng chiếu thường là các mặt phẳng tọa độ: *Oxy, Oyz, Ozx*

P(x, y, z) chiếu trên mặt phẳng *Oxy|Oyz|Ozx => P'(x', y', z')*.



Hình 4.3. Phép chiếu vuông góc



$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \Rightarrow \bar{P} = \mathbf{M}P \text{ where } \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Các ma trận phép phép chiếu trực giao :

a) Ma trận phép chiếu trên các mặt phẳng Oxy (*Z = 0)*

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

b) Ma trận phép chiếu trên các mặt phẳng Oxz (*Y = 0)*

$$M_Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Ma trận phép chiếu trên các mặt phẳng Oyz (*X = 0)*

$$M_X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

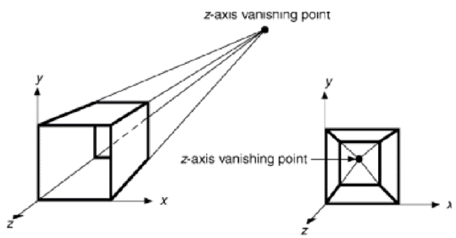
- Nếu đối tượng có các mặt của không song song với một trong các mặt phẳng tọa độ

$\Rightarrow$ Phép chiếu trực giao không mang lại hình dạng thật của vật thể.

$\Rightarrow$ Dùng các phép biến đổi hình học (quay, dịch chuyển, ...) để mặt phẳng chính của đối tượng trùng với một mặt phẳng tọa độ.

4.3.2 Phép chiếu phối cảnh (Perspective Projection)

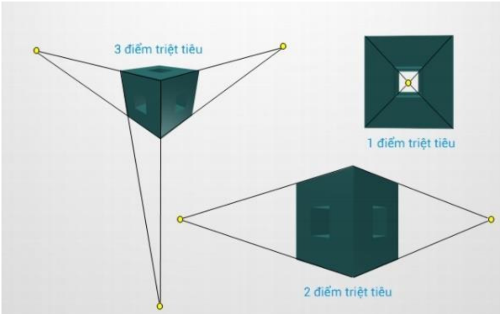
- Các tia chiếu xuất phát từ 1 điểm gọi là tâm chiếu.
- Kích thước đối tượng sẽ nhỏ dần khi tâm chiếu lùi xa khỏi mặt phẳng chiếu.
- Phép chiếu phối cảnh tạo ra hiệu ứng về luật xa gần tạo cảm giác về độ sâu của đối tượng trong thế giới thực cho phép quan sát mô hình thực.
- Các đoạn thẳng song song của mô hình 3D sau phép chiếu hội tụ tại 1 điểm gọi là điểm triệt tiêu (*vanishing point*).

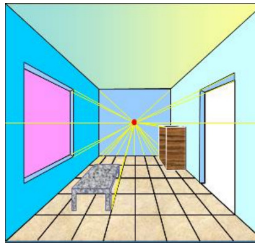


Phân loại phép chiếu phối cảnh:

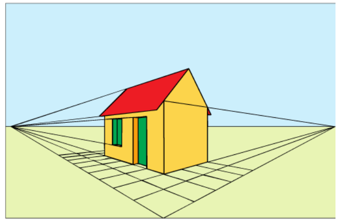
- Dựa vào tâm chiếu - Centre Of Projection (COP)
- Dựa vào mặt phẳng chiếu (projection plane)

Ba dạng phép chiếu phối cảnh: *một điểm, hai điểm và ba điểm triệt tiêu*

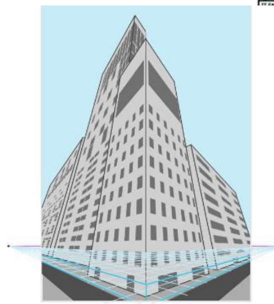




Hình chiếu 1 điểm tụ/triệt tiêu

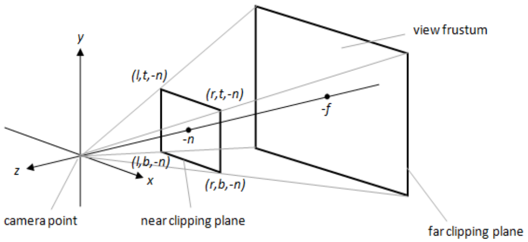


Hình chiếu phối cảnh 2 điểm tụ của ngôi nhà.

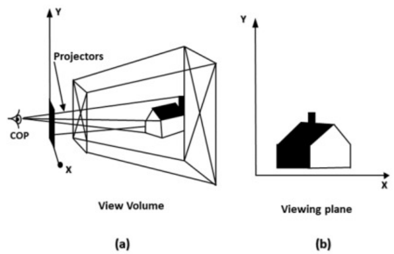


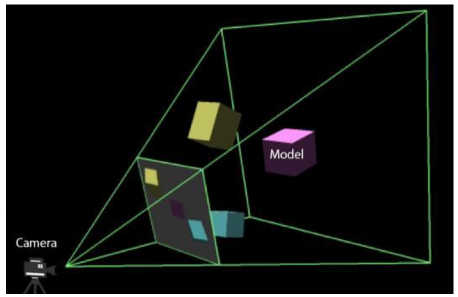
Hình chiếu 3 điểm tụ/triệt tiêu

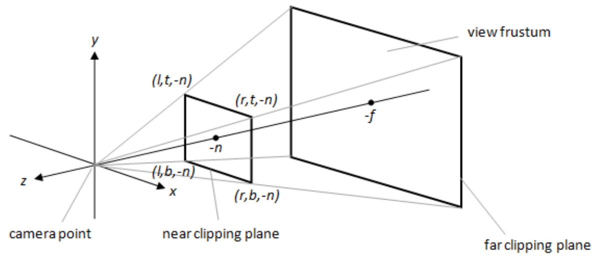
4.4 Xây dựng ma trận phép chiếu phối cảnh tổng quát



- Chiếu đỉnh của đối tượng trong camera space xuống mặt clip gần (*near clipping plane*).
- Mặt phẳng clip gần và mặt clip xa (*near/far clipping plane*) là hai mặt phẳng vuông góc với trục *Oz*, cách camera khoảng cách *n, f* tương ứng, về phía màn ảnh (theo hướng nhìn của người xem, ngược với hướng của trục *Oz*).
- Cảnh xem được giới hạn trong một hình chóp cụt (*view frustum*).

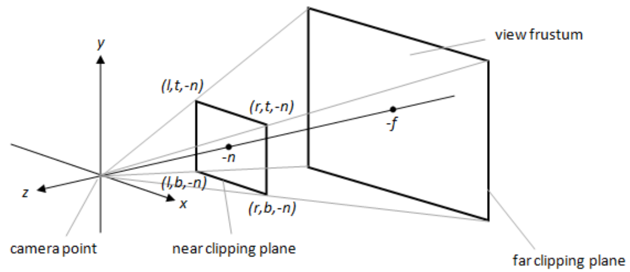






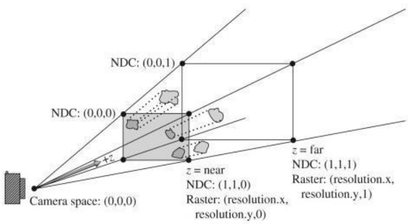
Hình chữ nhật của mặt phẳng clip gần có:

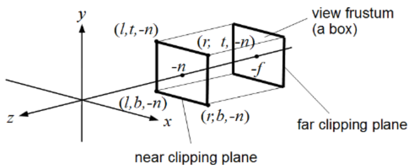
- Tọa độ: $-n$ (near) trên trục z.
- Đỉnh góc trên, bên trái của hình chữ nhật có tọa độ theo trục Ox là *l (left)*, theo Oy là *t* (top).
- Đỉnh góc trên, bên phải có tọa độ ngang là *r* (right), có tọa độ theo chiều cao là *t* (top).
- Đỉnh góc dưới, bên trái có tọa độ ngang là *l* (left), có tọa độ theo chiều cao là *b (bottom)*.
- Đỉnh góc dưới, bên phải có tọa độ theo chiều ngang là *r*, có tọa độ theo chiều cao là *b*.



- Tất cả các tọa độ này là trong camera space: Tọa độ gốc ≡ Tọa độ Camera
- Biến đổi camera chỉ đổi cơ sở và tịnh tiến, không có scale như trong biến đổi mô hình.

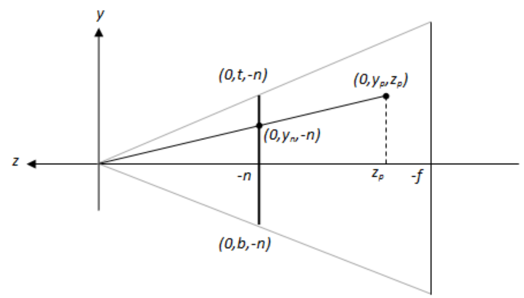
Ánh xạ các tọa độ x, y vào các khoảng [-1,1]





- Biến đổi phép chiếu sẽ giới hạn cảnh xem trong hình chóp cụt view, các cảnh bên ngoài sẽ được cắt bỏ.
- Chiếu ánh xạ các tọa độ (*x, y, z*) vào trong các khoảng [*-1,1*] và loại bỏ các giá trị bên ngoài.

4.4.1 Xác định tọa độ y_n trong mặt phẳng Oyz



- Chiếu một điểm P(x_p, y_p, z_p) (điểm biểu diễn bởi vector (x_p, y_p, z_p)) xuống mặt phẳng clip gần: lấy giao điểm của mặt phẳng clip gần với đường thẳng đi qua điểm đó và gốc tọa độ.
- Gọi giao điểm này là $P^*(x_n, y_n, z_n)$. Ta có $z_n = -n$.

- Chiếu không gian xuống mặt phẳng *Oyz*:

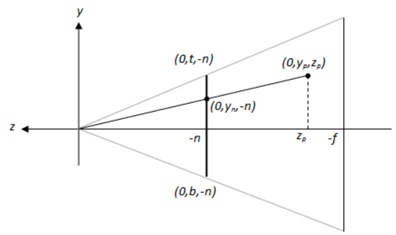
  Hình chiếu tương ứng của điểm P (x_p, y_p, z_p) là $(0, y_p, z_p)$

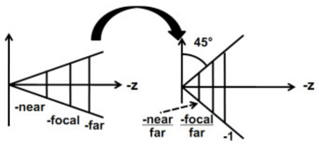
  Hình chiếu tương ứng của P*(x_n, y_n, $-n$) là $(0, y_n, -n)$.

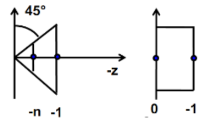
Ta có:

$$\frac{y_n}{y_p} = \frac{-n}{z_p} \Rightarrow y_n = \frac{-ny_p}{z_p}$$

- Khi $y_n = top$, giá trị ánh xạ mong muốn là $y_p' = 1$
- Khi $y_n = bottom$, giá trị ánh xạ mong muốn là $y_p' = -1$







Biểu diễn $y_{p'}$ dưới dạng một ánh xạ tuyến tính theo y_n :

$y_{p'} = A*y_n + B$

Ta có hệ phương trình tuyến tính:

$$\begin{cases} A*t + B = 1 \\ A*b + B = -1 \end{cases} \quad Suy\ ra \quad \begin{cases} A = \dfrac{2}{t-b} \\ B = 1 - A*t = 1 - \dfrac{2t}{t-b} = \dfrac{-(t+b)}{t-b} \end{cases}$$

Vậy:

$$y_{p'} = \frac{2y_n}{t-b} - \frac{t+b}{t-b} = -\frac{2ny_p}{z_p(t-b)} - \frac{t+b}{t-b}$$

$$y_{p'} = \frac{1}{-z_p}\left(\frac{2n}{t-b}y_p + \frac{t+b}{t-b}z_p\right)$$

4.4.2 Xác định tọa độ x_n trong mặt phẳng Oxz

Phân tích tương tự đối với tọa độ *x*, ta được $x_{p'}$:

$$x_{p'} = \frac{1}{-z_p}\left(\frac{2n}{r-l}x_p + \frac{r+l}{r-l}z_p\right)$$

4.4.3 Xác định tọa độ z_n

Ánh xạ tọa độ z vào khoảng [-1,1]

Khi $z_p = -f$, giá trị ánh xạ mong muốn là $z_p' = 1$

Khi $z_p = -n$, giá trị ánh xạ mong muốn là $z_p' = -1$

Ánh xạ cho z_p' tuyến tính trực tiếp theo biến z_p sẽ không thuận tiện cho biểu diễn ánh xạ theo phép nhân ma trận sau này => Ta ánh xạ tới z_p' theo biến $1/z_p$:

$$z_{p'} = A\frac{1}{z_p} + B$$

Kết quả có hệ phương trình tuyến tính:

$$\begin{cases} -A\dfrac{1}{f} + B = 1 \\ -A\dfrac{1}{n} + B = -1 \end{cases} \quad Suy\ ra \quad \begin{cases} A = \dfrac{2fn}{f-n} \\ B = 1 + \dfrac{2fn}{f} * \dfrac{1}{f-n} = 1 + \dfrac{2n}{f-n} = \dfrac{f+n}{t-n} \end{cases}$$

$$z_{p'} = \frac{1}{-z_p}\left(-\frac{f+n}{f-n}z_p - \frac{2nf}{f-n}\right)$$

>>

Từ các kết quả trên ta có

$$x_{p'} = \frac{1}{-z_p}\left(\frac{2n}{r-l}x_p + \frac{r+l}{r-l}z_p\right)$$

$$y_{p'} = \frac{1}{-z_p}\left(\frac{2n}{t-b}y_p + \frac{t+b}{t-b}z_p\right)$$

$$z_{p'} = \frac{1}{-z_p}\left(-\frac{f+n}{f-n}z_p - \frac{2nf}{f-n}\right)$$

Biểu diễn dạng tích ma trận như sau:

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p}\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

>> Ma trận phép chiếu phối cảnh tổng quát:

$$M_{projection} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p} M_{projection}\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} -z_p x_{p'} \\ -z_p y_{p'} \\ -z_p z_{p'} \\ -z_p \end{bmatrix} = M_{projection}\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

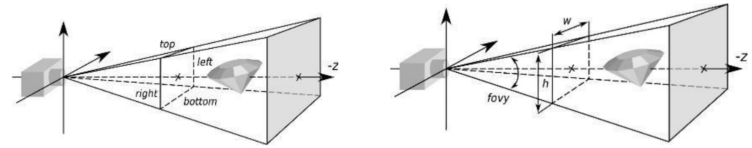
>> Khi nhân bên trái tọa độ camera với ma trận M_{pro}, kết quả được tọa độ vế trái, tọa độ này được gọi là tọa độ clip:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} -z_p x_{p'} \\ -z_p y_{p'} \\ -z_p z_{p'} \\ -z_p \end{bmatrix}$$

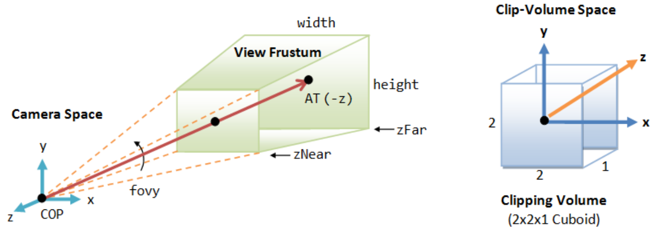
- Perspective Projection
  The call *glFrustum(l, r, b, t, n, f)* generates a matrix R, where

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \text{ and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$

4.5 Xây dựng ma trận phép chiếu theo góc nhìn (fovy)



- Ma trận phép chiếu trong thực tế là ma trận phép chiếu theo góc nhìn, là trường hợp đặc biệt của ma trận phép chiếu phối cảnh.
- Hình chữ nhật của mặt phẳng clip gần (và của mặt phẳng clip xa) được bố trí đối xứng thì tâm nằm trên trục z.
  Do đó: $l + r = t + b = 0$ ($l = -r$, $t = -b$)
  Ngoài ra:
  $$(t-b)\frac{1}{n} = \tan\left(\frac{fovy}{2}\right)$$
  Với $fovy$ là góc nhìn theo chiều thẳng đứng.



Nếu tỉ lệ màn hình là *a* (*a* = *width/height*) thì:

$$(r-l)\frac{2}{n} = a*(t-b)\frac{2}{n} = a*\tan\left(\frac{fovy}{2}\right)$$

Gọi $d = 1/\tan\left(\frac{fovy}{2}\right)$. Ta có: $\frac{2*n}{r-l} = \frac{d}{a}$ $\qquad \frac{2*n}{t-b} = d$

Ma trận phép chiếu phối cảnh tổng quát:

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p} M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \qquad \begin{bmatrix} -z_p x_{p'} \\ -z_p y_{p'} \\ -z_p z_{p'} \\ -z_p \end{bmatrix} = M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

$$M_{projection} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Ma trận phép chiếu theo góc nhìn (fovy):

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p} M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \qquad \begin{bmatrix} -z_p x_{p'} \\ -z_p y_{p'} \\ -z_p z_{p'} \\ -z_p \end{bmatrix} = M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

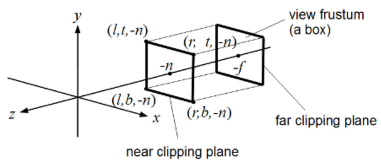
$$M_{projection} = \begin{bmatrix} \frac{d}{a} & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

4.5.1 Phép chiếu trực giao (orthographic projection)

- Các tọa độ *x, y* kết quả không phụ thuộc vào tọa độ *z* đầu vào.
- Tuy nhiên, tọa độ z vẫn được xử lý và được ánh xạ vào khoảng [-1,1].
- Hình chóp cụt trở thành hình hộp.
- Tọa độ clip cũng là tọa độ thiết bị chuẩn hóa.

Ma trận phép chiếu trực giao:

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p} M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$



Ma trận phép chiếu phối cảnh tổng quát:

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p} M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \qquad \begin{bmatrix} -z_p x_{p'} \\ -z_p y_{p'} \\ -z_p z_{p'} \\ -z_p \end{bmatrix} = M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

$$M_{projection} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Ma trận phép chiếu trực giao:

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \frac{1}{-z_p} M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \qquad \begin{bmatrix} -z_p x_{p'} \\ -z_p y_{p'} \\ -z_p z_{p'} \\ -z_p \end{bmatrix} = M_{projection} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

$$M_{projection} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Orthographic Projection:
  The call *glOrtho(l, r, b, t, n, f)* generates R, where

$$R = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & \frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & \frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & \frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } R^{-1} = \begin{bmatrix} \frac{r-l}{2} & 0 & 0 & \frac{r+l}{2} \\ 0 & \frac{t-b}{2} & 0 & \frac{t+b}{2} \\ 0 & 0 & \frac{f-n}{-2} & \frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.5.2 Các kỹ thuật đặc biệt tạo hình chiếu phối cảnh

- Các phép chiếu phối cảnh ở trên thường không tạo ra hình chiếu tương ứng với đối tượng.
- Để quan sát được nhiều mặt, thực hiện các phép tịnh tiến và quay kết hợp trước khi thực hiện phép chiếu phối cảnh một tâm chiếu.

4.5.3 Thể tích nhìn chuẩn

Sau khi biến đổi phối cảnh ⇒ Thực hiện tịnh tiến, co dãn thể tích nhìn để được thể tích nhìn chuẩn. Ma trận biến đổi $M_{projection}$:

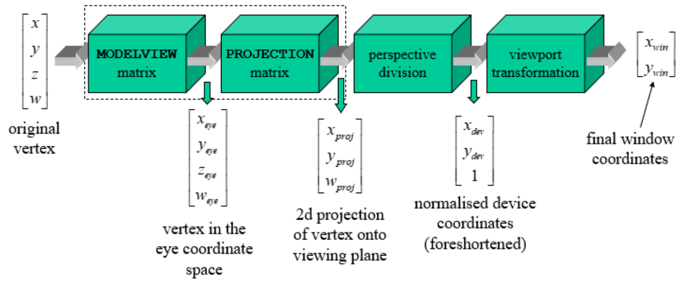
$$M_{projection} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

- Hàm *glFrustum(left, right, bott, top, N, F)* tạo ra ma trận $M_{projection}$.
- Hàm *gluPerspective(viewAngle, aspect, N, F)* cũng tạo ra ma trận $M_{projection}$ bằng cách tính:

$$top = N \tan\left(\frac{\pi}{180} viewAngle / 2\right)$$

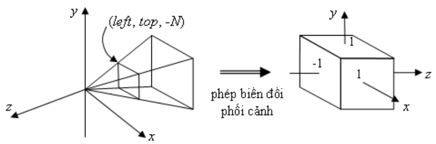
$bott = -top$
$right = top \times aspect$
$left = -right$

4.6 Phép chiếu trong OpenGL



- Trong OpenGL, tọa độ clip được đặt vào biến *gl_Position*, là một vector 4 chiều (kiểu vec4 trong OpenGL). Thành phần thứ tư là gl_Position.w sẽ chứa giá trị w_c ở trên.
- OpenGL thực hiện phép chia phối cảnh bằng cách chia các thành phần của *gl_Position* cho *gl_Position.w*, và kết quả được tọa độ thiết bị chuẩn hóa.

$$\begin{bmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \\ 1 \end{bmatrix} = \begin{bmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \\ w_c/w_c \end{bmatrix}$$

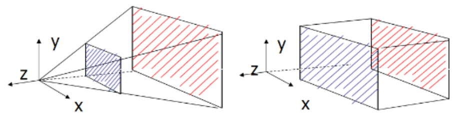


- Tọa độ thiết bị chuẩn hóa mới là tọa độ trong khoảng [-1,1], làm tọa độ đầu vào cho biến đổi viewport sau này.
- Thể tích nhìn chuẩn không phụ thuộc camera, dễ dàng cho cắt xén.

4.6.1 Ma trận phép chiếu

GL_MODELVIEW: Object->Camera
GL_PROJECTION: Camera->Screen
glViewport(0,0,w,h): Screen->Device

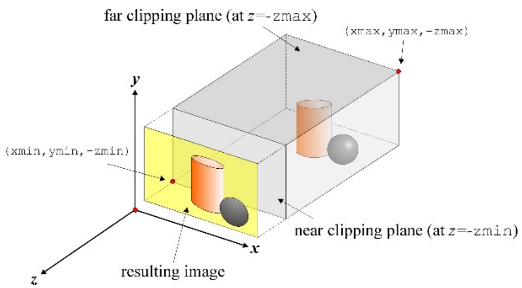
Trước khi thực hiện các thao tác chiếu, cần gọi 2 hàm
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
Ma trận hiện hành tương ứng phép chiếu này



Perspective: **gluPerspective()** Parallel: **glOrtho()**

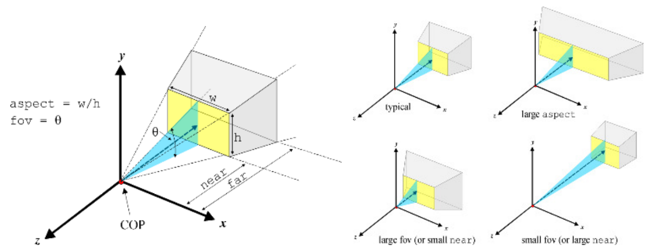
4.6.2 Chiếu song song

glOrtho(xmin, xmax, ymin, ymax, zmin, zmax);



4.6.3 Chiếu phối cảnh (Perspective Projection)

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



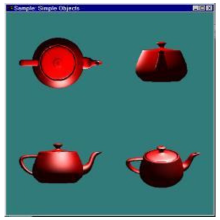
$$\frac{h/2}{near} = \tan\frac{\theta}{2} \Rightarrow h = 2\,near\,\tan\frac{\theta}{2}$$

Ví dụ: Minh họa phép chiếu trong OpenGL

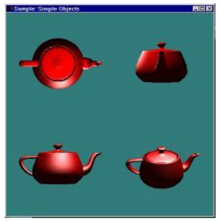
```
void myDisplay() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION); // set up projection
    glLoadIdentity();
    gluLookAt( ... ); // set up camera frame
    gluPerspective(fovy, aspect, near, far);
    // or glFrustum(...)
    // or glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 50.0);

    glMatrixMode(GL_MODELVIEW);
    drawModel(); // draw everything
    glutSwapBuffers();
}
```


Ví dụ minh họa:

```
// top left: top view
glViewport(0, win_height/2, win_width/2, win_height/2);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 50.0);
gluLookAt(0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glCallList(object);

// top right: right view
glViewport(win_width/2, win_height/2, win_width/2, win_height/2);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 50.0);
gluLookAt(5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glCallList(object);

// bottom left: front view
glViewport(0, 0, win_width/2, win_height/2);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 50.0);
gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glCallList(object);
```

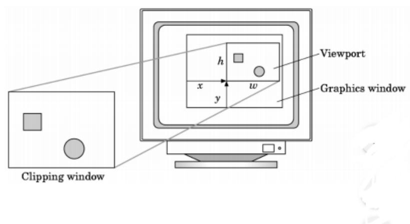



```
// bottom right: rotating perspective view
glViewport(win_width/2, 0, win_width/2, win_height/2);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(70.0, 1.0, 1, 50);
gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(30.0, 1.0, 0.0, 0.0);
glRotatef(Angle, 0.0, 1.0, 0.0);
glCallList(object);
```

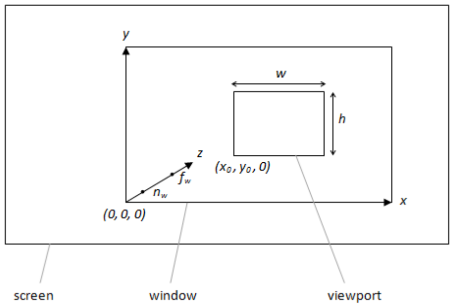


4.7 Biến đổi khung nhìn (VIEWPORT TRANSFORMATION)

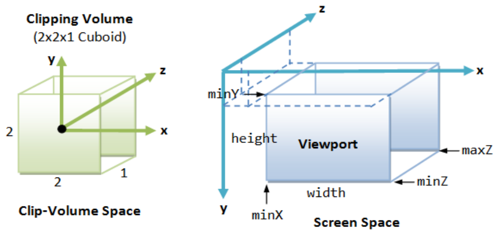
- Sau khi áp dụng ma trận biến đổi phép chiếu, được tọa độ clip.
- Tọa độ clip là tọa độ thiết bị chuẩn hóa trong khoảng [-1,1].
- Tọa độ thiết bị chuẩn hóa là tọa độ logic, cần phải được biến đổi sang tọa độ cửa sổ để hiển thị trong một màn hình cụ thể.
- Khung nhìn có thể là toàn bộ cửa sổ (mặc định) hoặc là một hình chữ nhật bộ phận của cửa sổ.



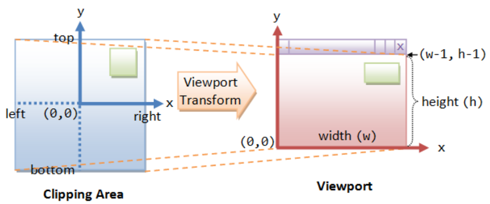
- Gốc tọa độ đặt tại góc dưới bên trái của cửa sổ, trục x hướng phải, trục y hướng lên trên.
- Tọa độ x, y xác định vị trí của một điểm ảnh,
- Trục z mang tính qui ước, mặc định hướng vào trong màn hình. Tọa độ z được dùng để xử lý buffer độ sâu để qui định điểm ảnh được hiển thị nếu nó gần nhất tới người xem, và không nếu đã có một điểm ảnh gần hơn.
- Góc dưới bên trái của khung nhìn được xác định trên cửa sổ tại tọa độ $x = x_0$ và $y = y_0$, cả hai tính theo đơn vị là điểm ảnh (pixel).
- *w:* Chiều rộng khung nhìn; *h:* chiều cao
    - Thang độ sâu mặc định đi từ n_w tới f_w, trong đó n_w là ánh xạ của mặt phẳng clip gần và f_w là ánh xạ của mặt phẳng clip xa, được đặt bởi hàm *glDepthRangef(nw, fw);*
    - Giá trị mặc định là $n_w = 0.0f$ và $f_w = 1.0f$.



- Giá trị mặc định của thang độ sâu thường là đủ dùng, và hàm đặt viewport chỉ liên quan đến không gian 2D của màn ảnh: *glViewport(x_0, y_0, w, h);*
- Độ phân giải buffer độ sâu và liên quan tới kiến trúc không gian camera:



Hai hàm *glViewport()* và *glDepthRangef()* sử dụng dụng ma trận biến đổi $M_{viewport}$:



- Ma trận biến đổi $M_{viewport}$:

$$M_{viewport} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & x_0 + \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & y_0 + \frac{h}{2} \\ 0 & 0 & \frac{f_w - n_w}{2} & \frac{f_w + n_w}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gọi z_{ndc} là thành phần z của tọa độ thiết bị chuẩn hóa và z_w là thành phần z của tọa độ cửa sổ, ta có:

$$z_w = \frac{f_w - n_w}{2} z_{ndc} + \frac{f_w + n_w}{2}$$

Gọi z_e là thành phần z của tọa độ camera, trong biến đổi phép chiếu, ta tính được:

$$z_{ndc} = \frac{2nf}{z_e(f-n)} + \frac{f+n}{f-n}$$

Thay z_{ndc} vào biểu thức tính z_w

$$z_w = \frac{nf(f_w - n_w)}{z_e(f-n)} + \frac{(f+n)(f_w - n_w)}{2(f-n)} + \frac{f_w + n_w}{2}$$

$$z_w = \frac{nf(f_w - n_w)}{z_e(f-n)} + \frac{ff_w - nn_w}{f-n}$$

Biến đổi đẳng thức trên để biểu diễn z_e như một hàm của z_w

$$z_w(f-n) = \frac{nf(f_w - n_w)}{z_e} + ff_w - nn_w$$

$$\frac{nf(f_w - n_w)}{z_e} = z_w(f-n) - ff_w + nn_w$$

$$z_e = \frac{nf(f_w - n_w)}{z_w(f-n) - ff_w + nn_w}$$

Giả sử buffer độ sâu chứa dữ liệu dạng điểm cố định (fixed point) với k bits, hệ số định tỉ lệ (scale) $1/s$ trong đó $s = 2k - 1$, thì độ sâu z_{wi} trong buffer độ sâu được tính: $z_{wi} = \lfloor s z_w \rfloor$

Giá trị tính toán thực tế của z_w là z_{wi}/s, thay giá trị này cho z_w trong công thức tính z_e:

$$z_e = \frac{nf(f_w - n_w)}{\frac{z_{wi}}{s}(f-n) - ff_w + nn_w}$$

Với giá trị mặc định $n_w = 0$ và $f_w = 1$:

$$z_e = \frac{nf}{\frac{z_{wi}}{s}(f-n) - f},$$

mặt khác công thức tính z_w theo z_e trở thành

$$z_w = \frac{nf}{z_e(f-n)} + \frac{f}{f-n}$$

Suy ra

$$z_{wi} = \lfloor s z_w \rfloor = \lfloor s\left(\frac{nf}{z_e(f-n)} + \frac{f}{f-n}\right) \rfloor$$