3D Graphics

NỘI DUNG

- Các phép biến đổi 3D
- Mô hình 3D
- Phép chiếu
- Quan sát đối tượng 3D
- Biểu diễn đường, mặt cong

Biến đổi 3D

- Các phép biến đổi hình học cơ sở
- Phép biến đổi ngược
- Kết hợp các phép biến đổi

Hệ toạ độ thuần nhất

Phép biến đổi affine 3D biến điểm P(x,y,z) thành điểm Q(x', y', z') có dạng:

$$Q = P.M$$

Trong đó, ma trận biến đổi M có dạng:

$$M = \begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ tr_x & tr_y & tr_z & 1 \end{pmatrix}$$

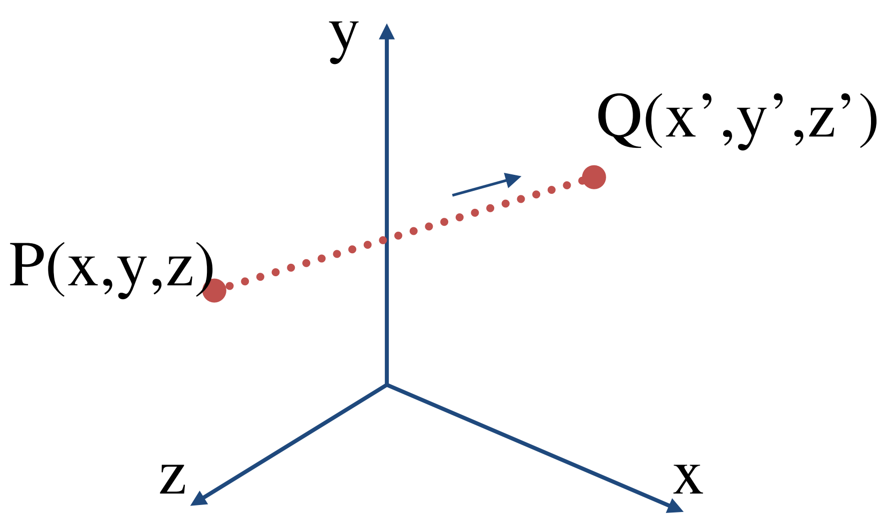
Quay , tỉ lệ

Tịnh tiến

Các phép biến đổi hình học cơ sở

Ma trận của phép biến đổi tịnh tiến:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix}$$



Các phép biến đổi hình học cơ sở

Đối xứng qua mặt phẳng xy: $M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Đối xứng qua mặt phẳng xz: $M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Đối xứng qua mặt phẳng yz: $M = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Các phép biến đổi hình học cơ sở

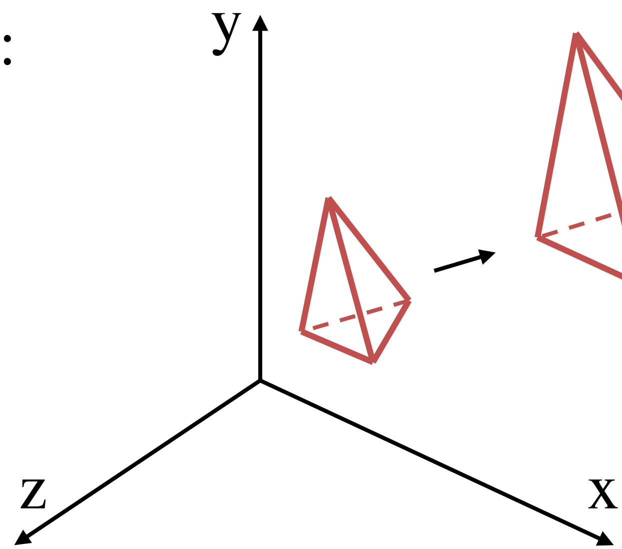
Đối xứng qua qua gốc tọa độ:

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

Các phép biến đổi hình học cơ sở

Ma trận của phép biến đổi tỉ lệ là:

$$M = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



s_x, s_y, s_z là các hệ số tỉ lệ

Khi $s_x = s_y = s_z = s$ ta có phép biến đổi đồng dạng.

Trong phép biến đổi trên, gốc toạ độ O sẽ có ảnh là chính nó. Khi đó O là tâm của phép biến đổi.

Các phép biến đổi hình học cơ sở

Phép biến đổi tỉ lệ theo tâm (x_0, y_0, z_0) được mô tả bằng dãy ba phép biến đổi sau:

- Tịnh tiến tâm (x_0, y_0, z_0) về gốc toạ độ.
- Biến đổi tỉ lệ có tâm ở gốc toạ độ.
- Tịnh tiến ngược tâm từ gốc toạ độ về lại vị trí ban đầu.

Ma trận của phép biến đổi theo tâm (x_0, y_0, z_0) như sau:

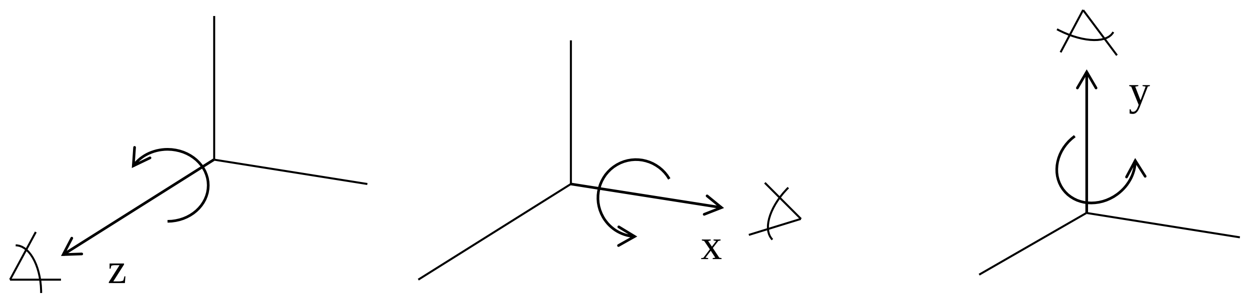
$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{pmatrix} \times \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & z_0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ (1-s_x)x_0 & (1-s_y)y_0 & (1-s_z)z_0 & 1 \end{pmatrix}$$

Các phép biến đổi hình học cơ sở

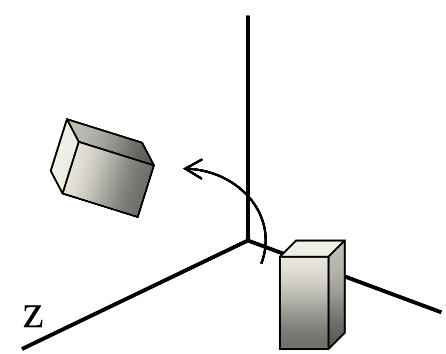
Quay quanh một trục toạ độ

- Qui ước: Quay ngược chiều kim đồng hồ theo trục sẽ tạo thành góc dương nếu nhìn về gốc tọa độ từ nửa trục dương.



Quay quanh trục z:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$



$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Các phép biến đổi hình học cơ sở

Quay quanh một trục toạ độ

Các ma trận biểu diễn phép quay quanh trục x, y, z một góc α lần lượt là R(x, α), R(y, α), R(z, α) như sau:

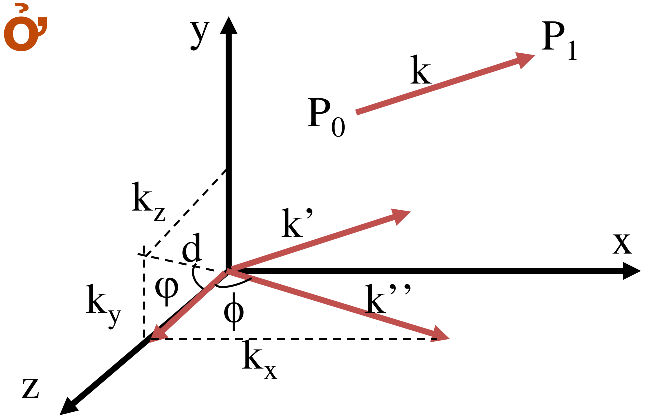
Quay quanh trục x:
$$R(x,\alpha)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ 0 & -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Quay quanh trục y:
$$R(y,\alpha)=\begin{pmatrix}\cos\alpha & 0 & -\sin\alpha & 0\\ 0 & 1 & 0 & 0\\ \sin\alpha & 0 & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Quay quanh trục z:
$$R(z,\alpha)=\begin{pmatrix}\cos\alpha & \sin\alpha & 0 & 0\\ -\sin\alpha & \cos\alpha & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Các phép biến đổi hình học cơ sở

Quay quanh một trục bất kỳ



Giả sử trục quay được biểu diễn bởi đường thẳng k đi qua 2 điểm P_0 và P_1.

Để thực hiện phép quay quanh k, ta thực hiện một chuỗi các thao tác sau:

-Thực hiện một số phép tịnh tiến, quay để k trùng trục z như sau:

+Tịnh tiến k về gốc toạ độ (thành trục k') với ma trận biến đổi là Tr(-P_0).

+Quay quanh trục x một góc φ để đặt k'lên mặt phẳng xy (thành trục k'') với ma trận biến đổi là: R(x, φ) .

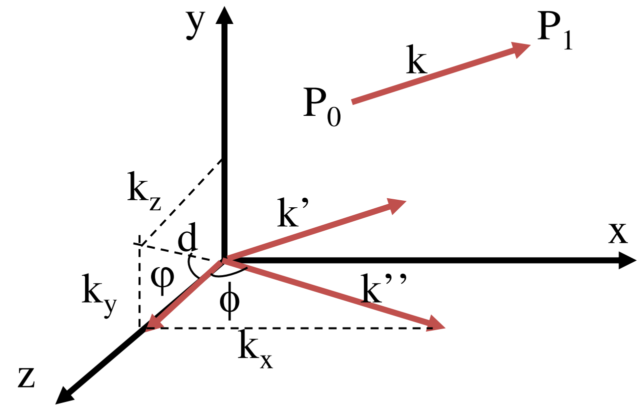
+Quay quanh trục y một góc θ để đưa k''về trục z với mt biến đổi là: R(y,-θ).

-Quay quanh trục z một góc α với ma trận biến đổi là: R(z, α).

-Thực hiện ngược lại các phép biến đổi sao cho k trở về vị trí ban đầu với các ma trận biến đổi lần lượt là: R(y,θ), R(x,-φ), Tr(P_0).

Các phép biến đổi hình học cơ sở

Quay quanh một trục bất kỳ



Vậy phép quay một điểm quanh trục k bất kỳ với một góc α được phân tích thành các chuỗi biến đổi sau:

$$Tr(-P_0).R(x, \varphi).R(y,-\theta). R(z, \alpha).R(y,\theta), R(x, -\varphi), Tr(P_0).$$

Trong đó góc quay φ, θ được xác định dựa trên cơ sở chiếu k' lên mặt phẳng yz, ta có:

$$k=P_0P_1\ ;$$

$$\cos(\varphi)=k_z/d\ ;\quad \sin(\varphi)=k_y/d$$

$$\cos(\theta)=k_z/k\ ;\quad \sin(\theta)=k_x/k$$

Phép biến đổi ngược

Tất cả các phép biến đổi đều có ma trận nghịch đảo

-Ma trận nghịch đảo của phép tịnh tiến có được bằng cách thay các hệ số t_x, t_y, t_z bằng $-t_x$, $-t_y$, $-t_z$.

-Ma trận nghịch đảo của phép biến đổi tỉ lệ có được bằng cách thay các hệ số s_x, s_y, s_z bằng $1/s_x$, $1/s_y$, $1/s_z$

-Ma trận nghịch đảo của phép quay có được bằng cách thay góc α bằng $-\alpha$.

Kết hợp các phép biến đổi

Tương tự như trong trường hợp biến đổi 2D.

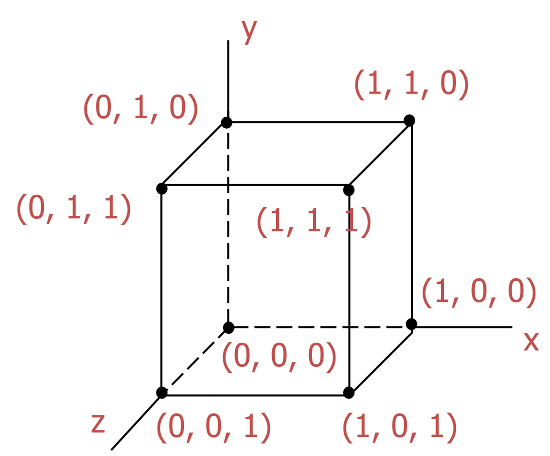
Nếu M_1 bến đổi P thành P'và M_2 biến đổi P' thành Q thì $M=M_1M_2$ sẽ biến đổi P thành Q.

⇨Ma trận của phép biến đổi kết hợp có thể được tính từ tích các ma trận của các phép biến đổi thành phần.

Bài tập

1. Một hình chóp A(0, 0, 0), B(1, 0, 0), C(0, 1, 0) và D(0, 0, 1) được xoay một góc 45^0 quanh đoạn thẳng L được xác định theo hướng **V**= **j** + **k** và đi qua đỉnh C. Xác định tọa độ các đỉnh sau phép xoay.

2. Tìm các tọa độ mới của khối vuông đơn vị như hình bên đây, sau khi xoay quanh một trục xác định bởi điểm A(2, 1, 0) và B(3, 3, 1). Góc xoay là 90^0 ngược chiều kim đồng hồ.



Mô hình 3D

- Mô hình khung kết nối
- Mô hình các mặt đa giác

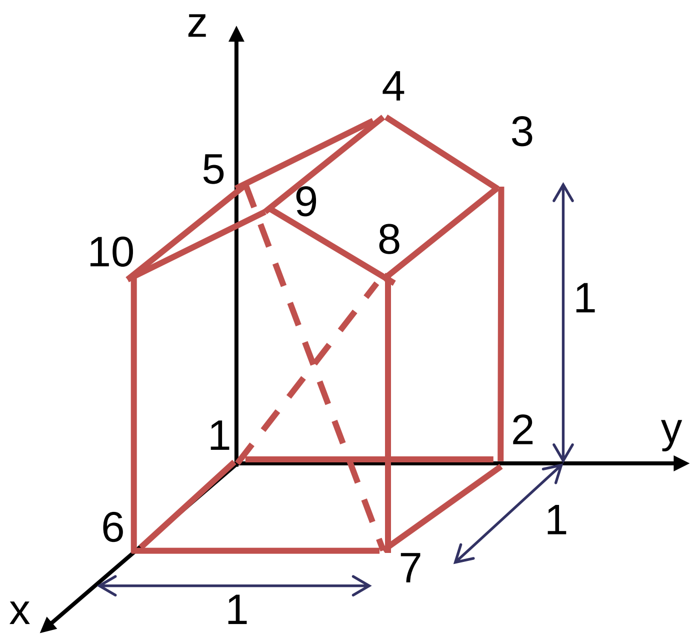
Mô hình khung kết nối

Mô hình khung kết nối (WF-WireFrame model) thể hiện hình dáng của một đối tượng 3D bởi 2 danh sách:

☞**Danh sách các đỉnh** (vertices): lưu toạ độ các đỉnh.

☞**Danh sách các cạnh** (edges): lưu 2 đỉnh đầu và cuối của từng cạnh.

Mô hình khung kết nối



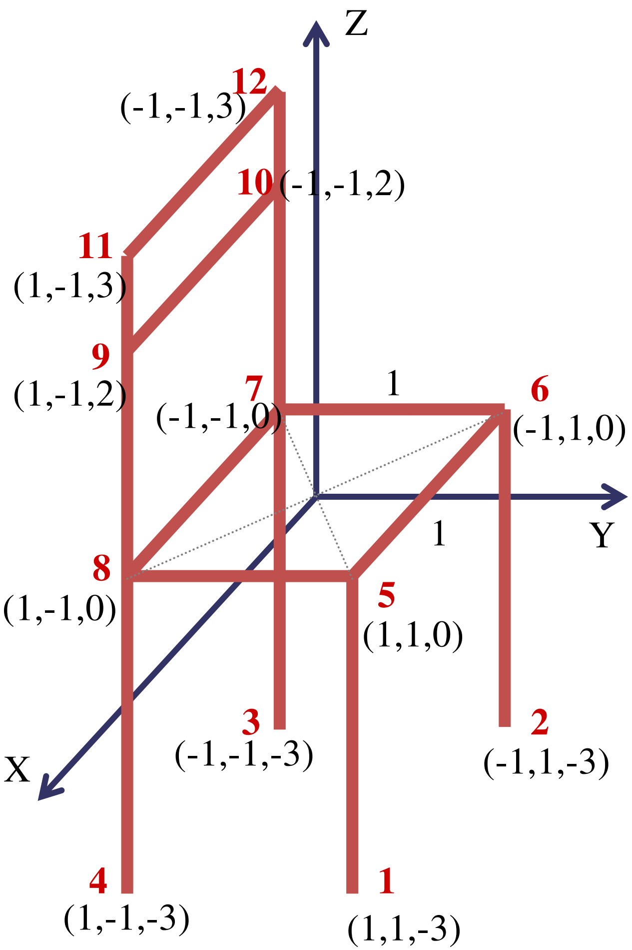
Danh sách đỉnh

Đỉnh	x	y	z
1	0	0	0
2	0	1	0
3	0	1	1
4	0	0.5	1.5
5	0	0	1
6	1	0	0
7	1	1	0
8	1	1	1
9	1	0.5	1.5
10	1	0	1

Danh sách cạnh

Cạnh	Đầu	Cuối
1	1	2
2	2	3
3	3	4
4	4	5
5	5	1
6	6	7
7	7	8
8	8	9
9	9	10
10	10	6
11	1	6
12	2	7
13	3	8
14	4	9
15	5	10
16	2	5
17	1	3

Mô hình khung kết nối



Danh sách đỉnh			
Đỉnh	X	Y	Z
1	1	1	-3
2	-1	1	-3
3	-1	-1	-3
4	1	-1	-3
5	1	1	0
6	-1	1	0
7	-1	-1	0
8	1	-1	0
9	1	-1	2
10	-1	-1	2
11	1	-1	3
12	-1	-1	3

Danh sách cạnh		
Cạnh	Đầu	Cuối
1	1	5
2	2	6
3	3	12
4	4	11
5	5	6
6	6	7
7	7	8
8	8	5
9	9	10
10	11	12

Mô hình khung kết nối

```
typedef struct 3DPoint{
        int x;   int y;   int z;
};
typedef struct EdgeType{
        int beginP;      int endP;
};
typedef struct WireFrame{
        int             numVertex, numEdge;
        3DPoint         vertex[MAX];
        EdgeType        edge[MAX];
};
```

Mô hình mặt đa giác

Mô hình các mặt đa giác (Polygon Mesh model) thể hiện hình dáng của một đối tượng 3D bởi 2 danh sách:

☞**Danh sách các đỉnh**: lưu toạ độ các đỉnh.

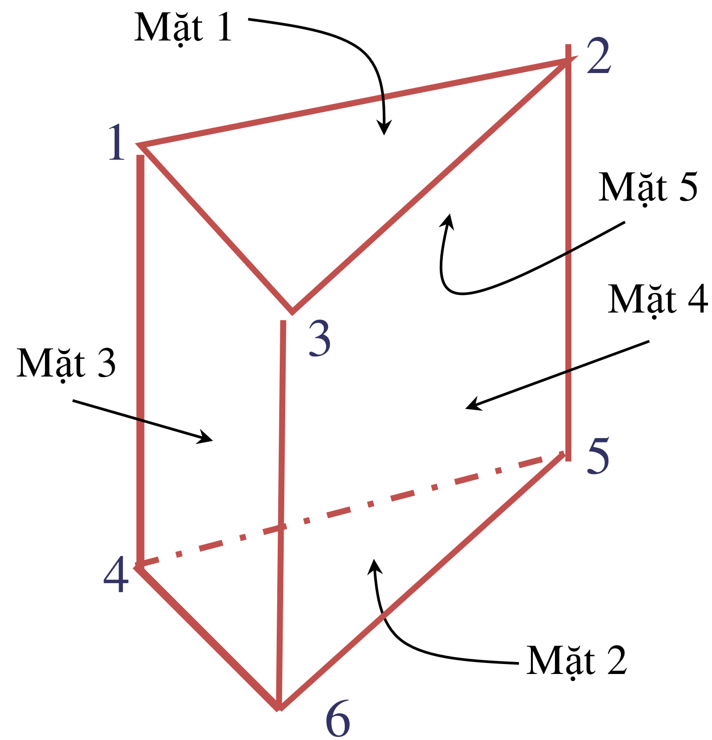
☞**Danh sách các mặt**: lưu thứ tự các đỉnh tạo nên mặt đó.

Mô hình mặt đa giác

Ví dụ: Mô tả vật thể như trong hình vẽ sau:

Danh sách đỉnh			
Đỉnh	X	Y	Z
1	x1	y1	z1
2	x2	y2	z2
3	x3	y3	z3
4	x4	y4	z4
5	x5	y5	z5
6	x6	y6	z6

Danh sách mặt	
mặt	
1	1,3,2
2	4,5,6
3	1,4,6,3
4	3,6,5,2
5	1,2,5,4



Mô hình mặt đa giác

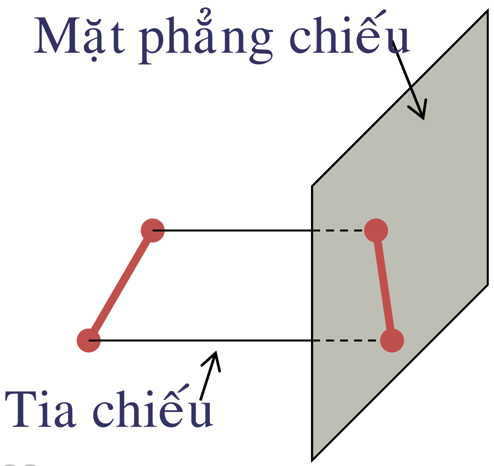
```
typedef struct 3DPoint{
        int x;   int y;   int z;
};
typedef struct  FaceType{
        int         nFace;
        int         indexFace[MAX];
};
typedef struct FaceModel{
        int                 numVertex, numFace;
        3DPoint         vertex[MAX];
        FaceType        face[MAX];
};
```

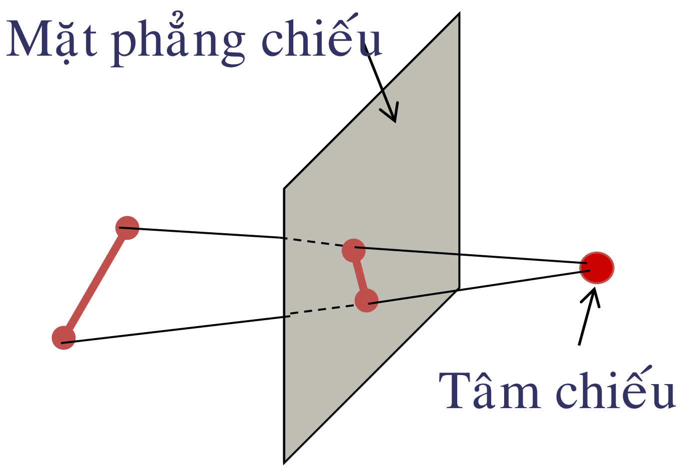
Phép chiếu

- **Chiếu** (Projection) là biến đổi hệ tọa độ n-chiều sang hệ tọa độ m-chiều, trong đó m<n.
  - Trong đồ họa máy tính thường sử dụng biến đổi 3D -> 2D
- Các khái niệm liên quan
  - **Tia chiếu**: đi qua các điểm trên đối tượng đến mặt phẳng để tạo ảnh 2D
  - **Mặt phẳng chiếu**: nơi hình thành ảnh 2D của các đối tượng 3D

...Phép chiếu

- Hai nhóm phép chiếu đối tượng 3D sang 2D cơ bản
  - **Chiếu song song** (parallel projection)
    - Chiếu các điểm trên đối tượng theo đường song song
    - Sử dụng nhiều trong đồ họa máy tính, vẽ kỹ thuật
  - **Chiếu phối cảnh** (perspective projection)
    - Chiếu các điểm trên đối tượng theo đường hội tụ đến tâm chiếu
    - Sử dụng nhiều trong các trò chơi (cảm giác thực hơn)

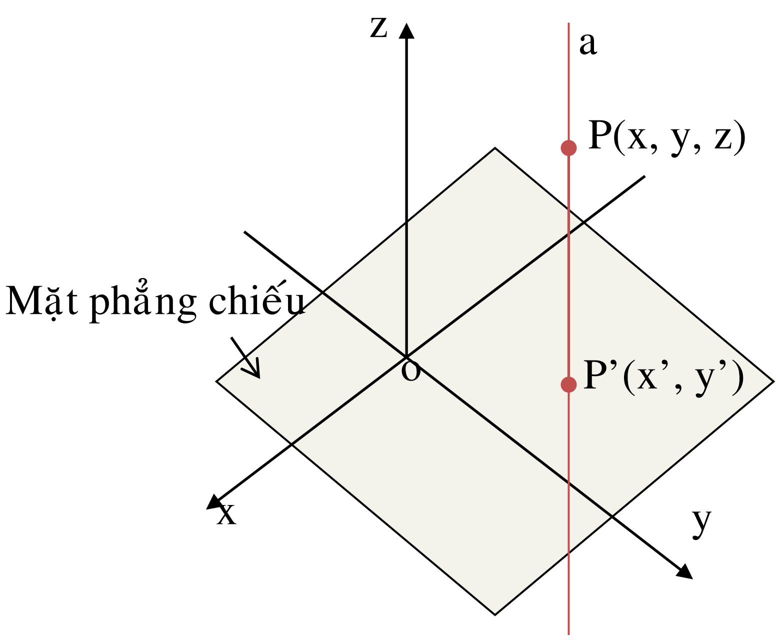


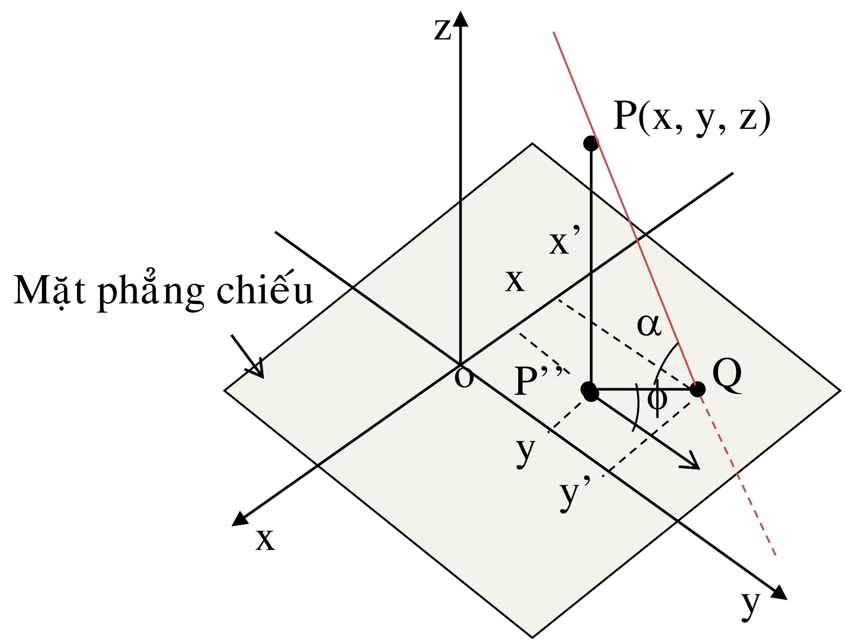


Phép chiếu song song (Parallel projective)

Khi hướng của tia chiếu vuông góc với mặt phẳng chiếu ta có **phép chiếu trực giao** (orthographic projection).

Ngược lại, ta có **phép chiếu xiên** (oblique projection).

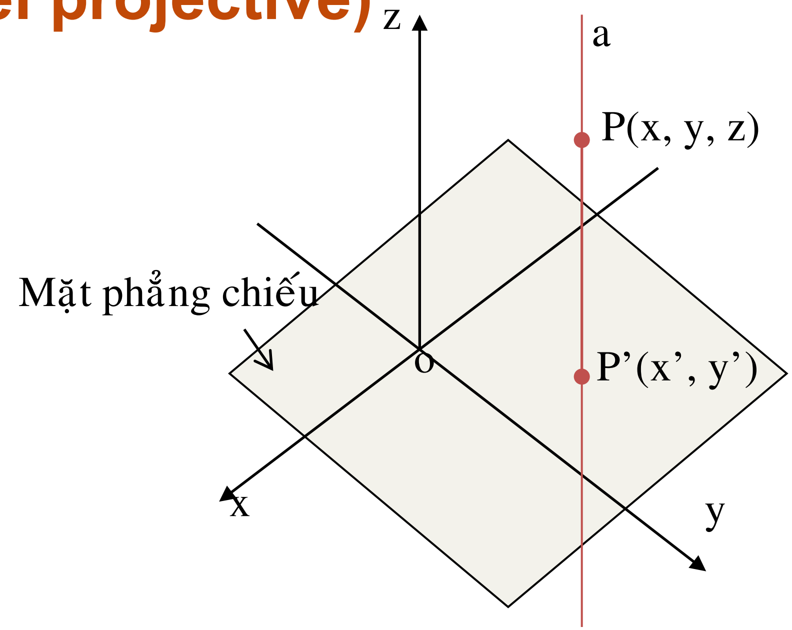


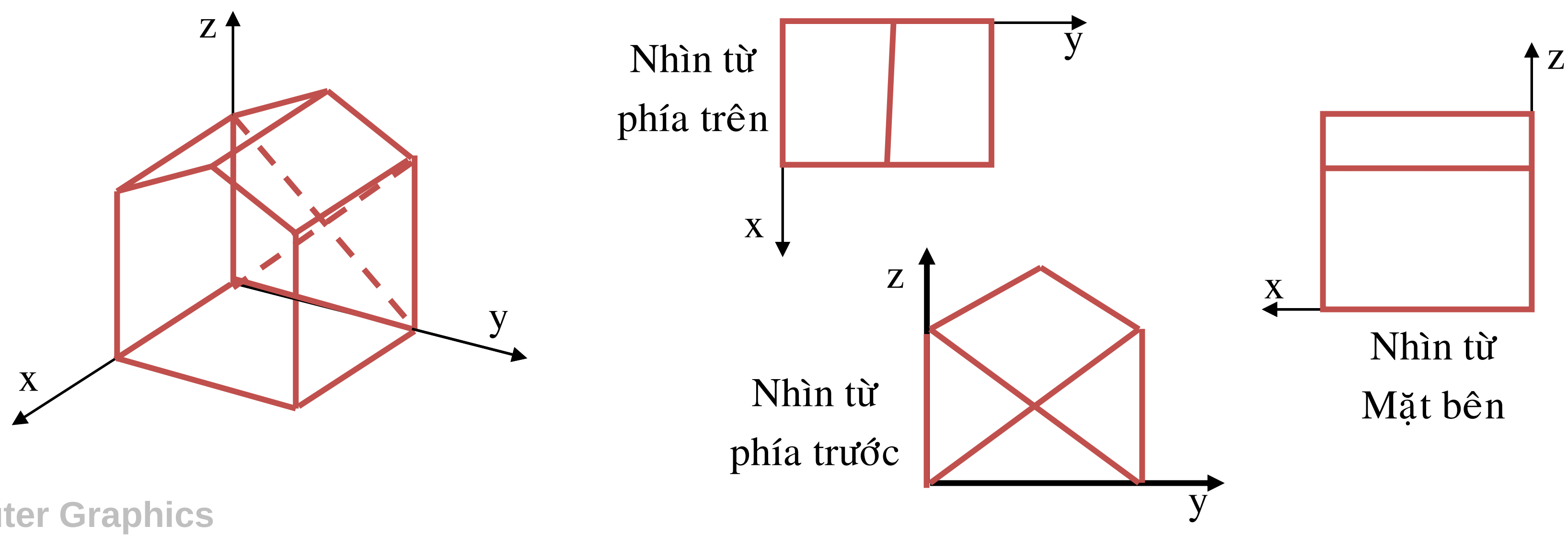


Phép chiếu song song (Parallel projective)

Phép chiếu trực giao

để chiếu điểm P(x, y, z) lên mặt phẳng chiếu thành P'(x', y'), cách đơn giản là bỏ đi thành phần z.





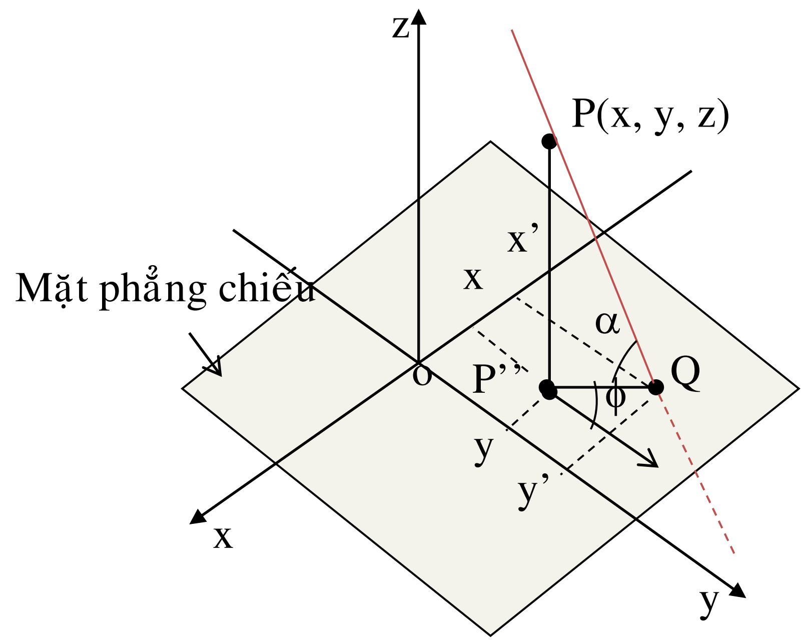
Phép chiếu song song (Parallel projective)

Phép chiếu xiên

điểm P(x,y,z) sẽ có ảnh là điểm P’.

Trong đó: P’’ là hình chiếu của P qua phép chiếu trực giao. α là góc tạo bởi tia chiếu và P’P’’và ϕ là góc tạo bởi P’P’’với trục y.

Biết P, α, ϕ ta có thể xác định được điểm chếu P’(x’,y’,z’).



☞ Phép chiếu song song bảo toàn được mối quan hệ giữa các chiều của đối tượng. Tuy nhiên phép chiếu song song không cho một biểu diễn thực của đối tượng ba chiều.

Phép chiếu song song (Parallel projective)

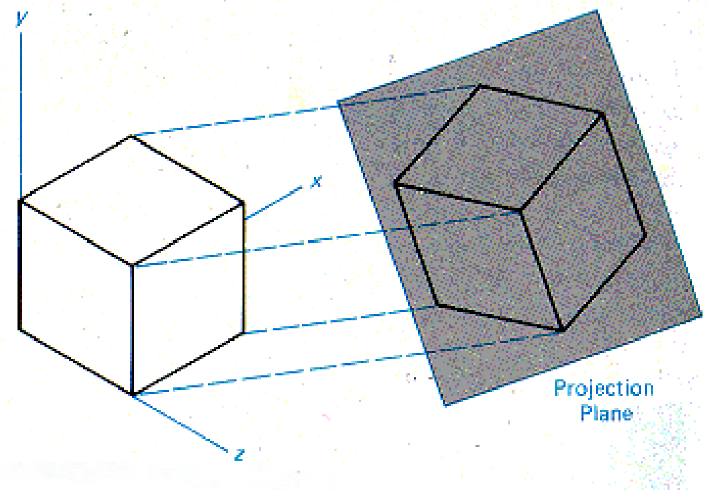
Chiếu xiên:

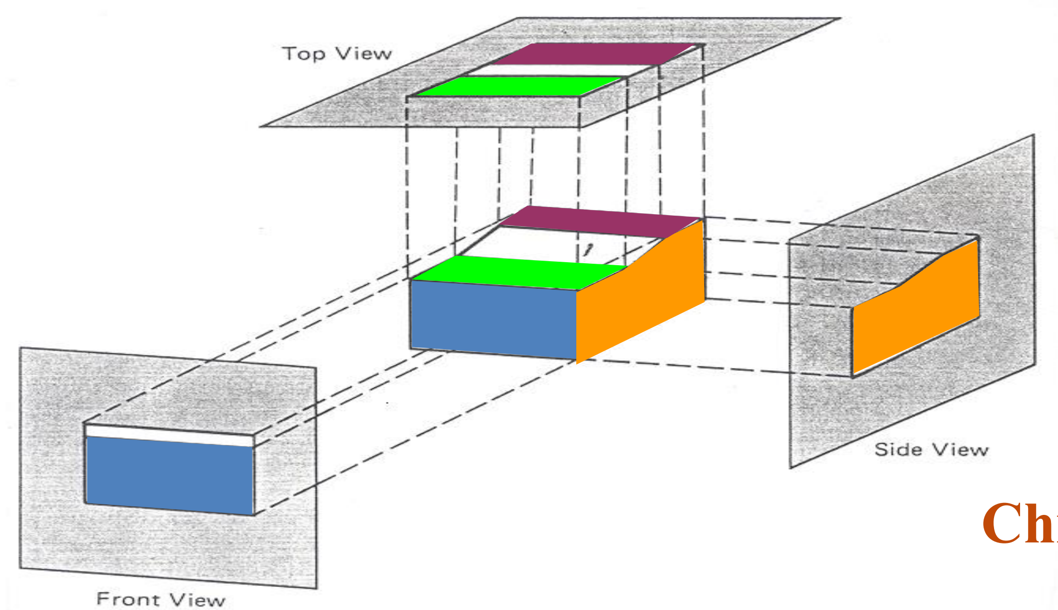
Chiếu lên mặt xy

xp=x

yp=y

z =0

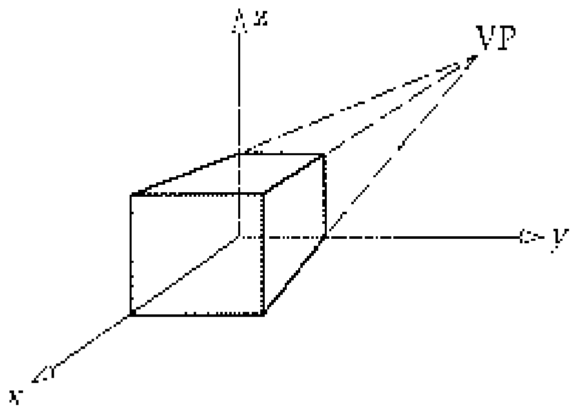




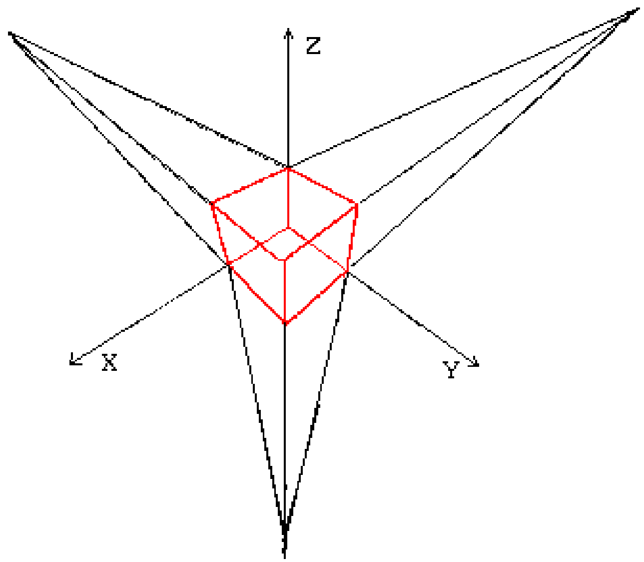
Chiếu trực giao

Phép chiếu phối cảnh (Perspective projection)

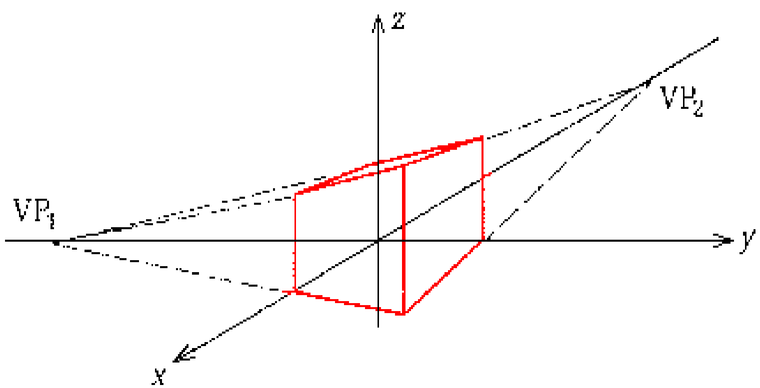
- Các tia chiếu gặp nhau tại tâm chiếu (vanishing point)



1 tâm chiếu: Mặt chiếu song song với hai trục tọa độ



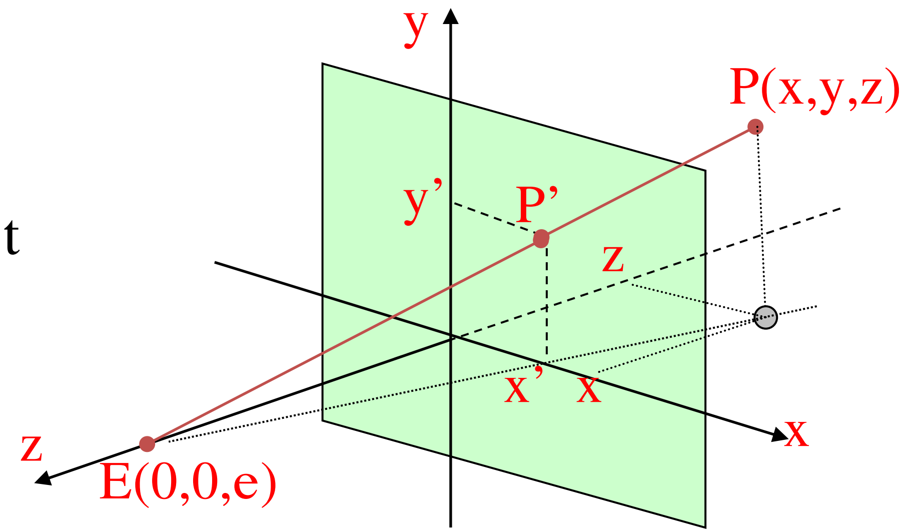
3 tâm chiếu: Mặt chiếu không song song với bất kỳ trục tọa độ nào



2 tâm chiếu: Mặt chiếu song song với một trục tọa độ

Phép chiếu phối cảnh (Perspective projection)

Các tia chiếu hội tụ về một điểm duy nhất gọi là mắt nhìn.
Phép chiếu phụ thuộc vào vị trí tương đối của mắt nhìn và mặt phẳng quan sát.

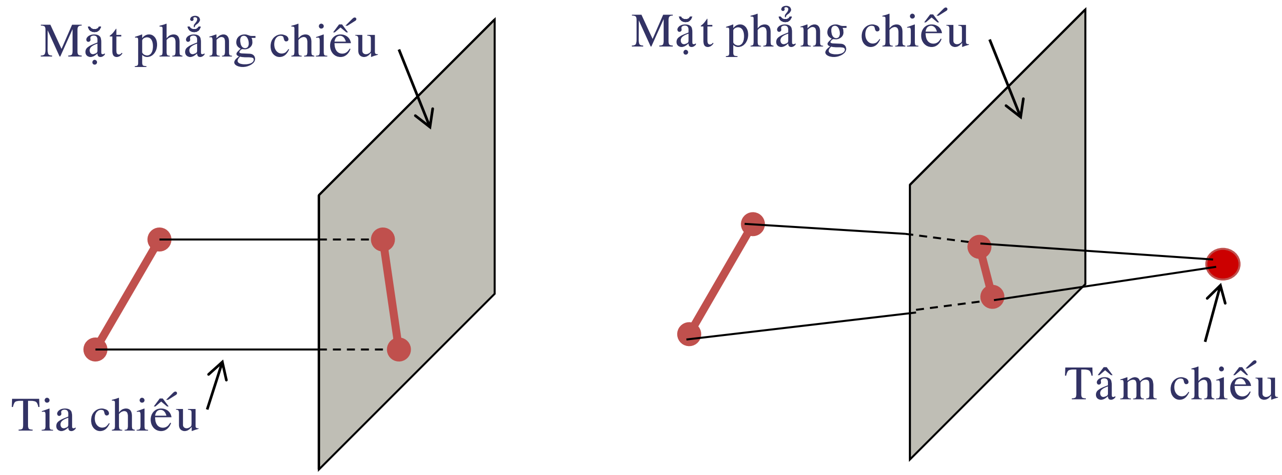


Giả sử mặt phẳng đượcđặt tại z=0, tâm phép chiếu được đặt trên trục z với khoảng cách e và P nằm trước mắt nhìn

Ta có: $\frac{x'}{x} = \frac{e}{e + (-z)} \Rightarrow x' = \frac{x}{1 - z/e}$

tương tự $\frac{y'}{y} = \frac{e}{e + (-z)} \Rightarrow y' = \frac{y}{1 - z/e}$ và $z' = 0$

Phép chiếu phối cảnh (Perspective projection)



- **Nhận xét**:
  - ☞ Phép chiếu phối cảnh không giữ nguyên hình dạng của vật thể.
  - ☞ Vật ở trước mặt phẳng chiếu thì được phóng lớn, sau mặt phẳng chiếu thì bị thu nhỏ. Vât ở xa thì trông nhỏ, ở gần thì trông lớn.
  - ☞ Ta có thể xem phép chiếu song song như là một phép chiếu phối cảnh nhưng có tâm chiếu ở xa vô cực

Quan sát đối tượng 3D

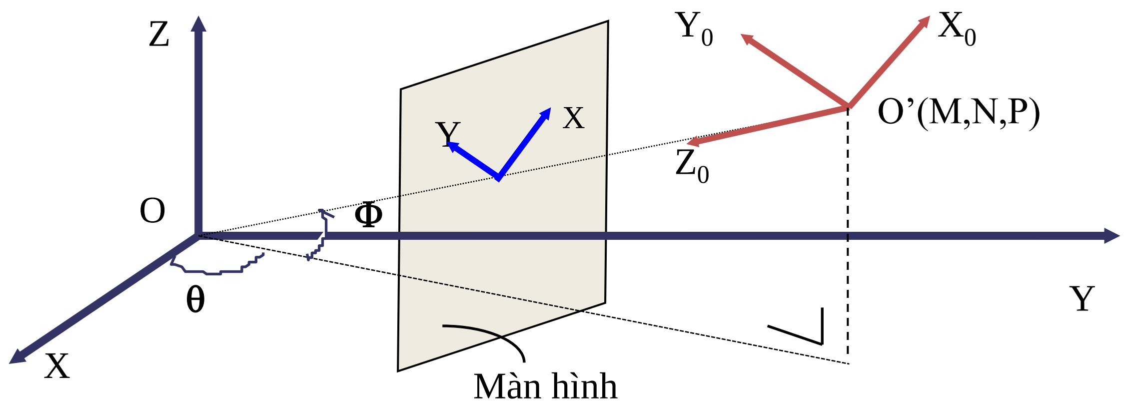
Khi mô tả việc quan sát một vật thể trong không gian ta cần lưu ý :

☞ Vật thể được chiếu lên một hệ theo quy tắc bàn tay phải (O,X,Y,Z)

☞ Mắt nằm ở gốc của một hệ theo quy tắc bàn tay trái (O',X_0,Y_0,Z_0)

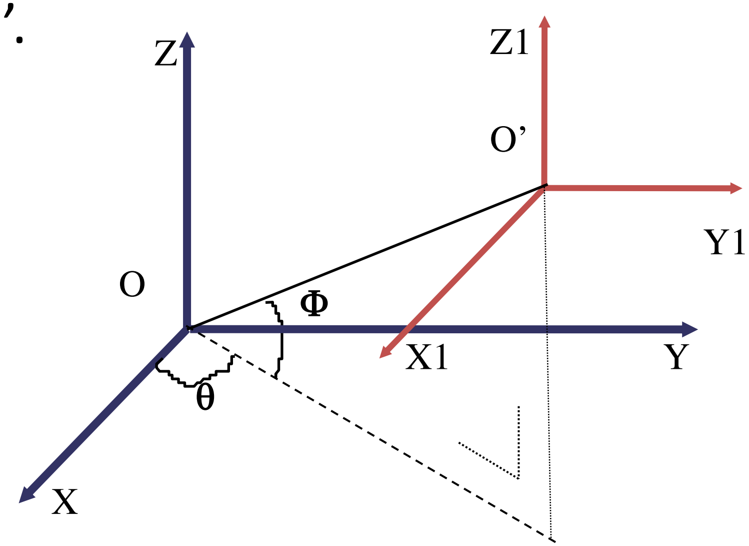
☞ Mặt phẳng chiếu vuông góc với đường thẳng OO'

☞Trục Z_0 của hệ toạ độ thứ 2 hướng vào gốc O. Hệ toạ độ thứ hai có tên là hệ toạ độ quan sát.



☞phép biến đổi một điểm P(x,y,z) trong hệ toạ độ thứ nhất sang P'(x0,y0,z0) trong hệ toạ độ thứ hai rồi chuyển sang toạ độ trên mặt phẳng quan sát ?

• **Bước 1**: Tịnh tiến gốc O thành O'.
Ma trận của phép tịnh tiến
(Lấy nghịch đảo):

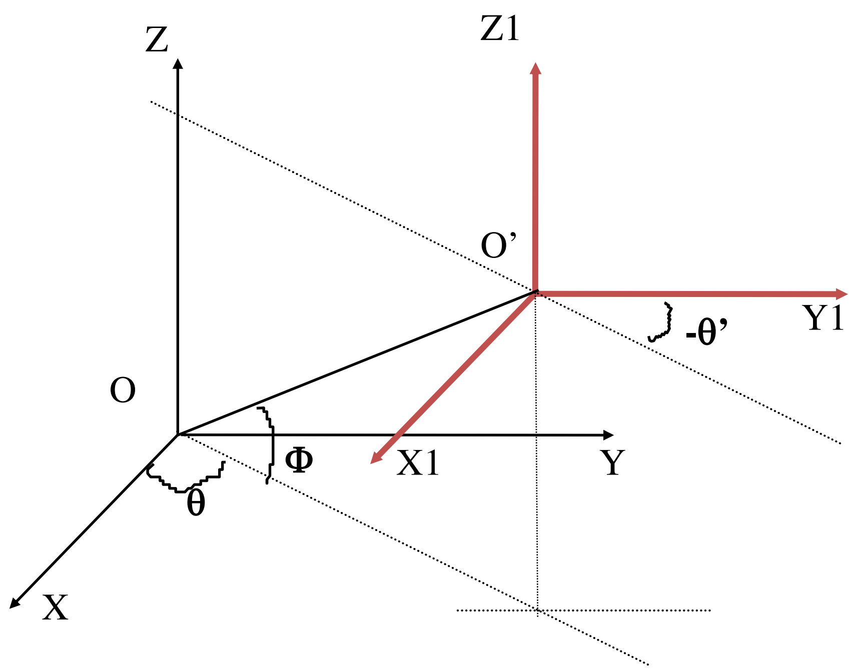


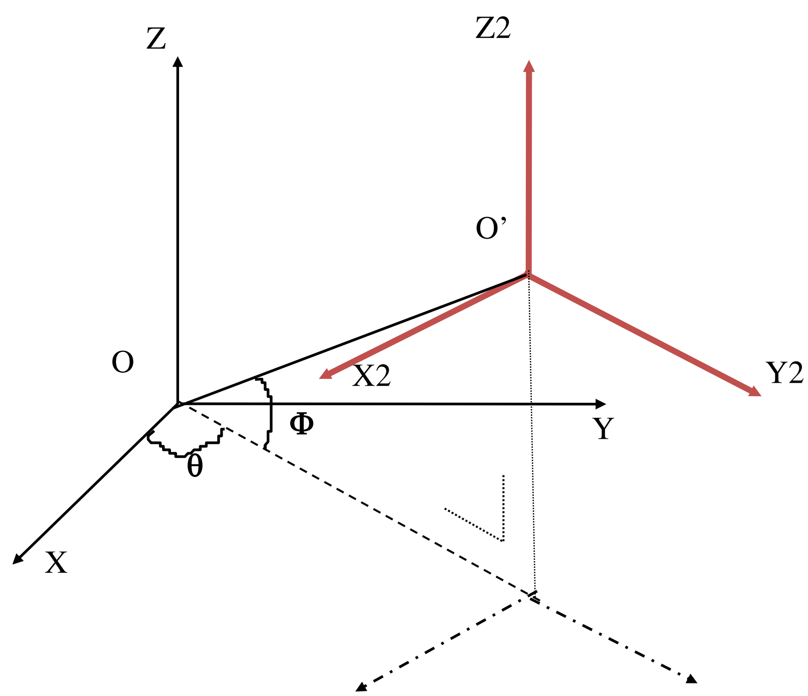
$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -M & -N & -P & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -R.Cos(\theta).Cos(\phi) & -R.Sin(\theta).Cos(\phi) & -R.Sin(\phi) & 1 \end{pmatrix}$$

☞Hệ (X,Y,Z) biến đổi thành hệ (X1,Y1,Z1).

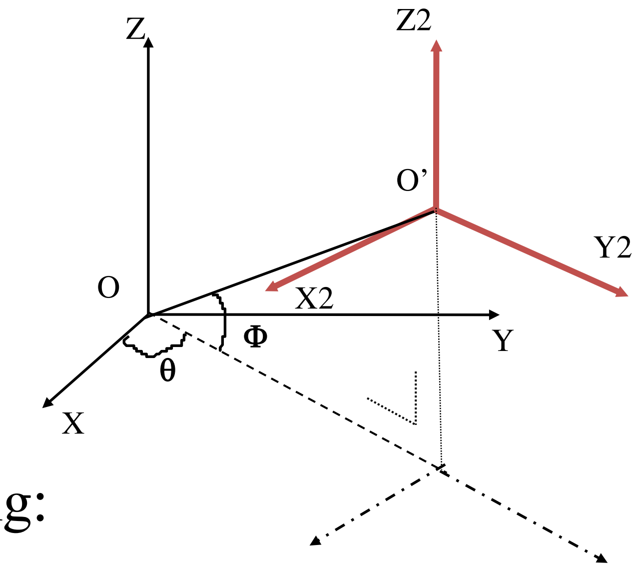
•**Bước 2**: Quay hệ (X1,Y1,Z1) một góc **-θ‘ (θ‘=90⁰ - θ)** quanh trục Z1 theo chiều kim đồng hồ. Phép quay này làm cho trục âm của Y1 cắt trục Z.





•**Bước 2**: Gọi Rz là ma trận tổng quát của phép quay quanh trục Z. Vì đây là phép quay hệ trục nên phải dùng ma trận nghịch đảo $R^{-1}{}_z$.

$$\mathbf{R}^{-1}{}_z = \begin{pmatrix} Cos(a) & -Sin(a) & 0 & 0 \\ Sin(a) & Cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Thay a = **-θ'** . Theo các phép toán lượng giác:

Sin(**-θ'**) = -Sin(**θ'**) = -Sin(90^0 - **θ) = -Cos(θ)**

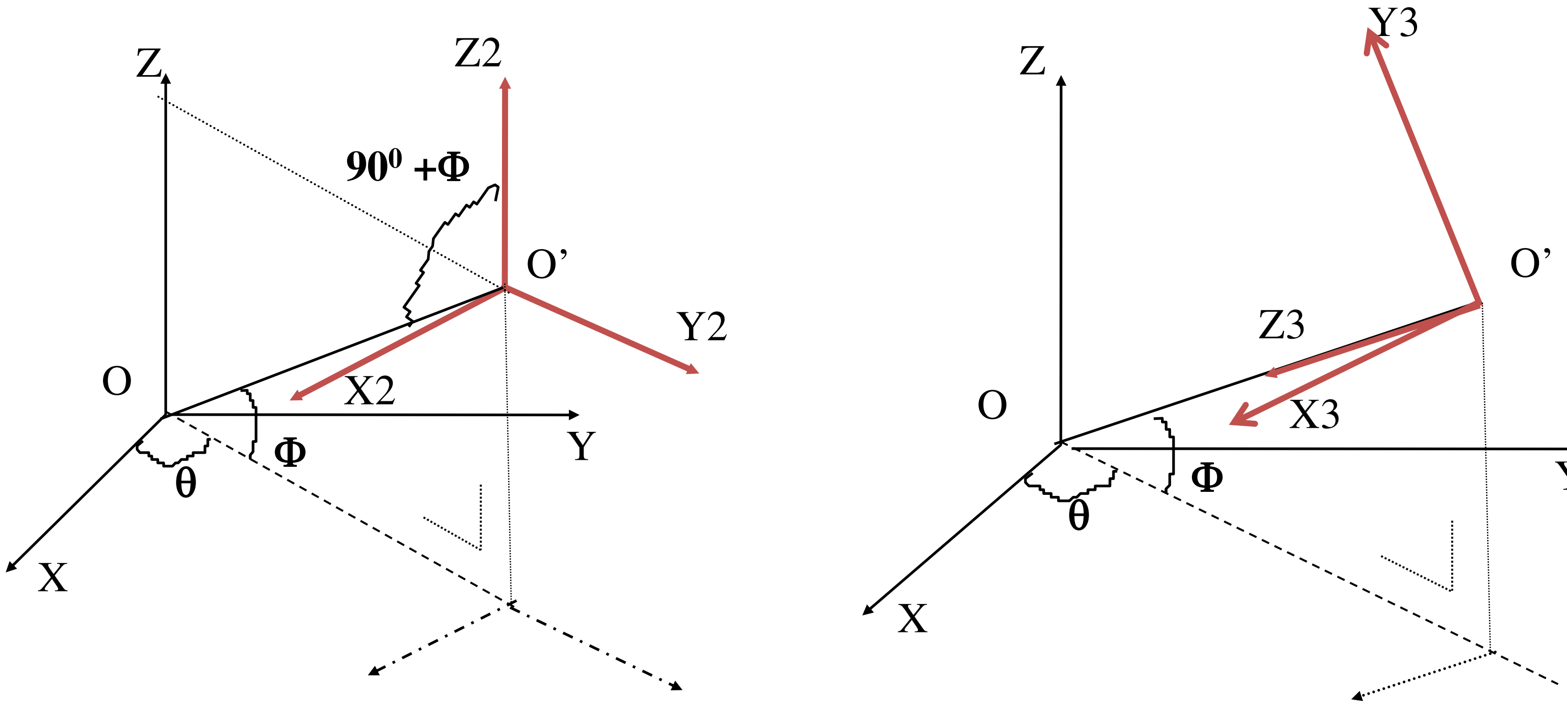
Cos(**-θ'**) = Cos(**θ'**) = Cos(90^0**-θ**) = Sin(**θ**)

Nên ma trận của phép quay tìm được sẽ có dạng:

$$\mathbf{B} = \begin{pmatrix} Sin(\theta) & Cos(\theta) & 0 & 0 \\ -Cos(\theta) & Sin(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

☞hệ (X1,Y1,Z1) biến đổi thành hệ (X2,Y2,Z2).

- **Bước 3**: Quay hệ (X2,Y2,Z2) một góc $90^0 + \Phi$ quanh trục X2. Phép biến đổi này sẽ làm cho trục Z2 hướng đến gốc O.



- **Bước 3**:

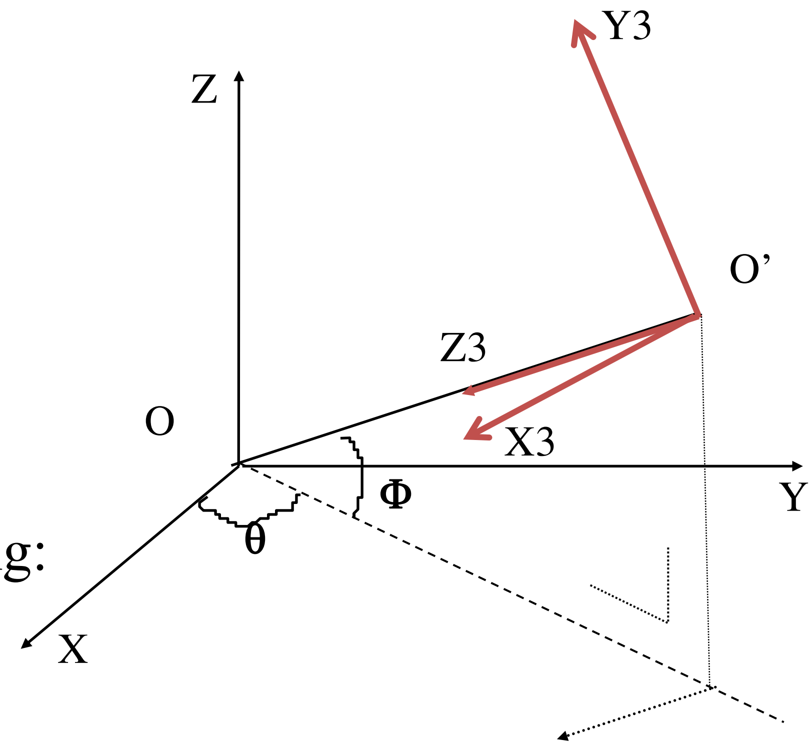
Ta có:

$$R^{-1}x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(a) & -Sin(a) & 0 \\ 0 & Sin(a) & Cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Thay góc a = 900 + **Φ** ,

ta có: Cos(900 + **Φ**) = -Sin(**Φ**)

và Sin(900 + **Φ**) = Cos(**Φ**)



Nên ma trận tìm được sẽ có dạng:

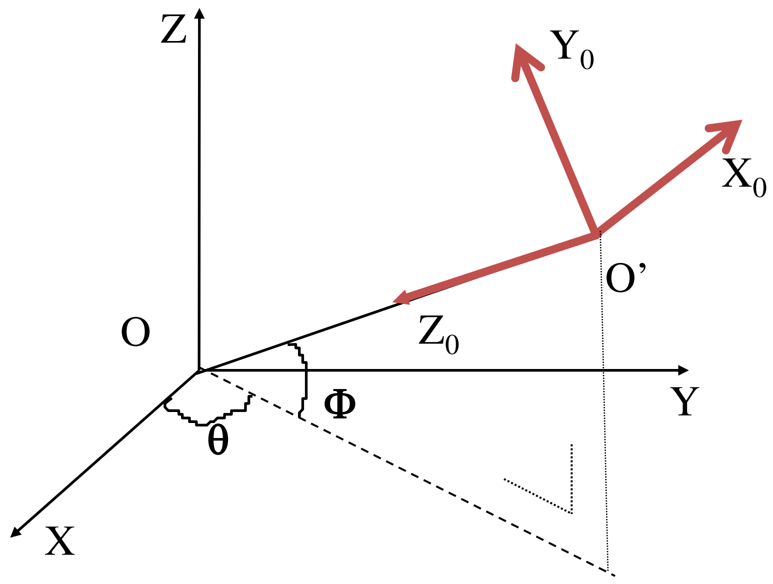
$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -Sin(\phi) & -Cos(\phi) & 0 \\ 0 & Cos(\phi) & -Sin(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

☞ hệ (X2,Y2,Z2) biến đổi thành hệ (X3,Y2,Z3).

- **Bước 4**: Biến đổi hệ trực tiếp (X3,Y3,Z3) thành hệ gián tiếp.

Đổi hướng trục X3 bằng cách đổi dấu các phần tử của cột X.
Ta nhận được ma trận:

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



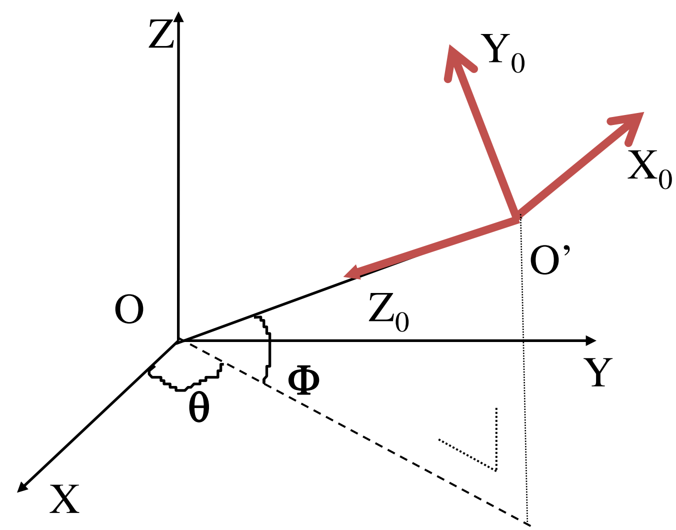
☞ hệ (X3,Y3,Z3) biến đổi thành hệ (X_0,Y_0,Z_0).

- **TÓM LẠI**

Một điểm trong không gian biểu diễn trong hệ quan sát có dạng:

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

Gọi T = A.B.C.D, ta tính được:



$$\mathbf{T} = \begin{pmatrix} -\sin(\theta) & -Cos(\theta).Sin(\phi) & -Cos(\theta).Cos(\phi) & 0 \\ Cos(\theta) & -Sin(\theta).Sin(\phi) & -Sin(\theta).Cos(\phi) & 0 \\ 0 & Cos(\phi) & -Sin(\phi) & 0 \\ 0 & 0 & R & 1 \end{pmatrix}$$

Vậy:

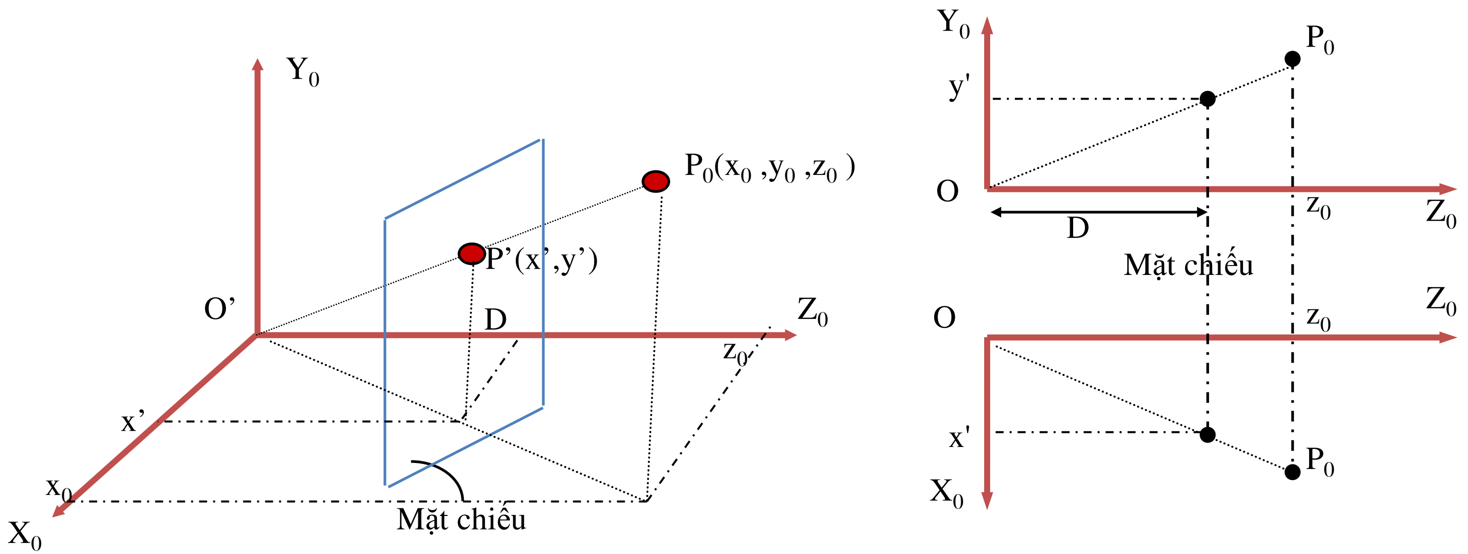
$$x_0 = -x.Sin(\theta) + y.Cos(\theta)$$

$$y_0 = -x.Cos(\theta).Sin(\Phi) - y.Sin(\theta).Sin(\Phi) + z.Cos(\Phi)$$

$$z_0 = -x.Cos(\theta).Cos(\Phi) - y.Sin(\theta).Cos(\Phi) - z.Sin(\Phi) + R$$

Chiếu ảnh trong hệ quan sát lên màn hình:

- Phép chiếu phối cảnh



Gọi D là khoảng cách từ mặt phẳng chiếu đến mắt(gốc tọa độ).

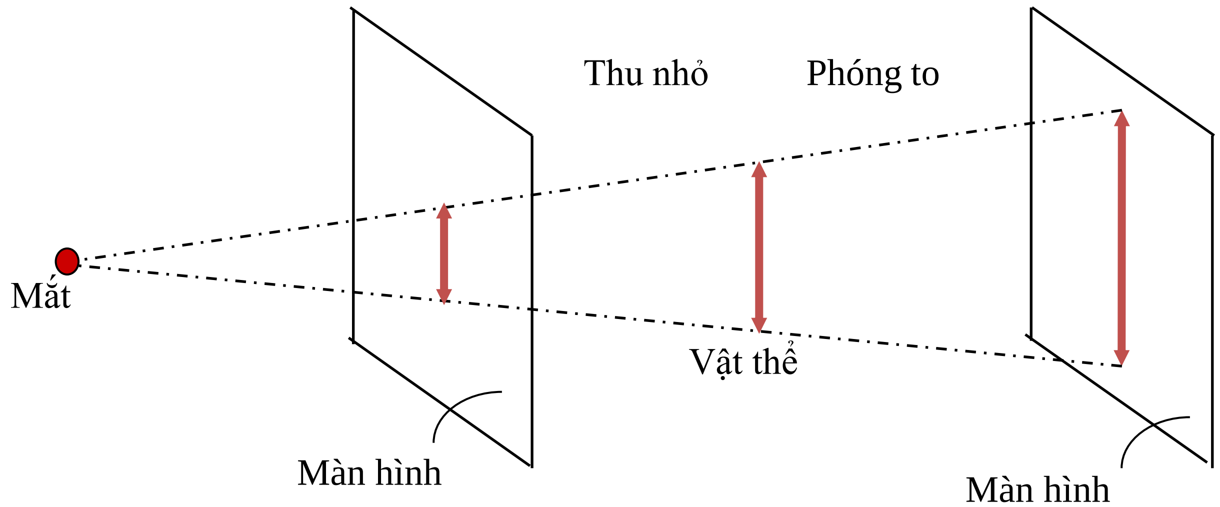
Ta có: $x'/x_0 = D/z_0$ và $y'/y_0 = D/z_0$

hay $x' = x_0.D/z_0$ và $y' = y_0.D/z_0$

Chiếu ảnh của hệ quan sát lên màn hình:

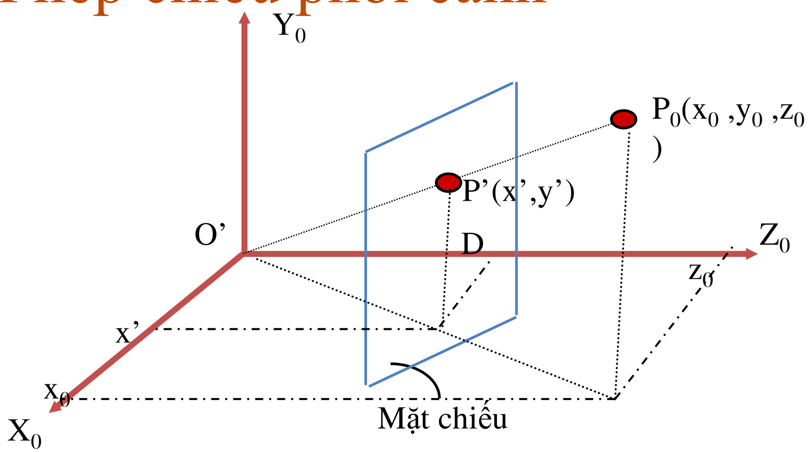
- Phép chiếu phối cảnh

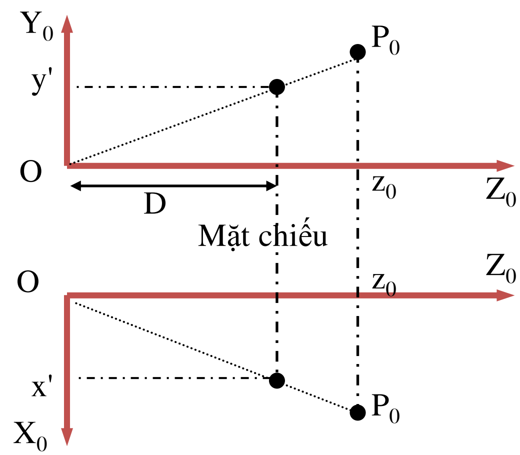
Nếu z>D thì ảnh thu nhỏ (hội tụ), ngược lại ảnh được phóng lớn.



Chiếu ảnh của hệ quan sát lên màn hình:

- Phép chiếu phối cảnh





Gọi D là khoảng cách từ mặt phẳng chiếu đến mắt(gốc tọa độ).

Ta có: $x'/x_0 = D/z_0$ và $y'/y_0 = D/z_0$

hay $x' = x_0 .D/z_0$ và $y' = y_0 .D/z_0$

- Phép chiếu song song

Xem như tâm chiếu ở xa vô cực, ta có:

$\mathbf{x' = x_0}$ và $\mathbf{y' = y_0}$

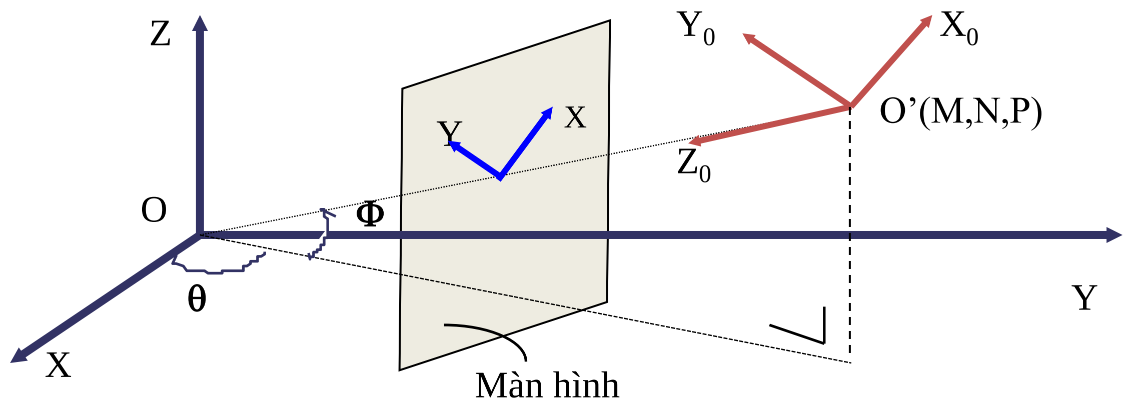
...Quan sát đối tượng 3D

- 4 giá trị ảnh hưởng đến phép chiếu vật thể 3D:

  + Các góc **θ** , **Φ**

  + Khoảng cách R từ O đến O'

  + Khoảng cách D từ O' đến mặt phẳng quan sát.



- Như vậy:

  -Tăng giảm **θ** sẽ quay vật thể trong mặt phẳng (XY).

  -Tăng giảm **Φ** sẽ quay vật thể lên xuống.

  -Tăng giảm R để quan sát vật từ xa hay gần.

  -Tăng giảm D để phóng to hay thu nhỏ ảnh.

…Quan sát đối tượng 3D

```
3DPoint        Chieu(3DPoint  P){
      //1: chieu phoi canh,      0:  chieu song song
      3DPoint   T;      float      x0,y0,z0, ap,bt;
      ap=M_PI*anpha/180;      bt=M_PI*beta/180;
      x0=-P.x*sin(ap)+P.y*cos(ap);
      y0=-P.x*cos(ap)*cos(bt)-P.y*sin(ap)*sin(bt)+P.z*cos(bt);
      if (PhepChieu){
         z0=-P.x*cos(ap)*cos(bt)-P.y*sin(ap)*cos(bt)-P.z*sin(bt)+R;
         T.x=(int)(x0*D/z0)+xc;         T.y=(int)(y0*D/z0)+yc;
      }
      else {
            T.x=(int)(x0*D)+yc;        T.y=(int)(y0*D)+yc;
      }
      return T;
}
```

…Quan sát đối tượng 3D

```
void Vedoituong3D(WireFrame w)
{       3DPoint       D1,D2;
        int     i;
        for (i=1;i<=w.socanh;i++)
        {       D1=Chieu(w.dinh[w.canh[i].DD]);
                D2=Chieu(w.dinh[w.canh[i].DC]);
                line(D1.x,D1.y,D2.x,D2.y);
        }
}
```

Biểu diễn đường, mặt cong

- Đường cong

- Mặt cong

Đường cong

- Cách biểu diễn
  - Đường cong bất kỳ có thể biểu diễn bới ma trận điểm
    - Cần số lượng điểm vô cùng lớn để biểu diễn chính xác hình dạng
  - Sử dụng hàm đa thức để thể hiện hình dạng đường cong
    - Dạng tổng quát của hàm đa thức

$$p(x) = a_n x_n^n + a_{n-1} x_{n-1}^{n-1} + \ldots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

n – nguyên dương, a_0, a_1, ..., a_n là số thực

- Đa thức thuận tiện cho tính toán bằng máy tính
- Trong đồ họa đòi hỏi xác định tiếp tuyến, pháp tuyến cho đường cong. Đa thức dễ dàng tính vi phân.

Đường cong

Phương trình tham số của đường thẳng

Gọi M(x,y) là một điểm thuộc đường thẳng AB.

Ta có: $\overrightarrow{AM} = t.\overrightarrow{AB}$

$$\Leftrightarrow \begin{cases} x - x_A = t.(x_B - x_A) \\ y - y_A = t.(y_B - y_A) \end{cases} \Leftrightarrow \begin{cases} x = (1-t).x_A + t.x_B = X(t) \\ y = (1-t).y_A + t.y_B = Y(t) \end{cases}$$

Hay M=(1-t).A+t.B

Khi t=0 thì M≡A

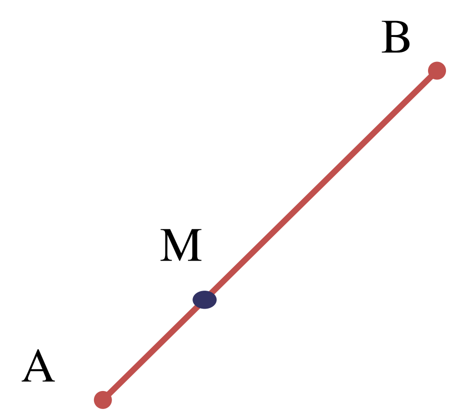
t=1 thì M≡B

Nếu $0 \le t \le 1$ thì M thuộc đoạn AB

Phương trình tham số đoạn AB là:

P(t)=(1-t).A+t.B với $0 \le t \le 1$

Hay P(t) = (X(t), Y(t))



Đường cong Bezier

<u>Bài toán</u>: Cho n+1 điểm p_0, p_1, p_2,....,p_n được gọi là các điểm kiểm soát (điểm điều khiển). Xây dựng đường cong trơn đi qua 2 điểm p và pn được giới hạn trong bao lồi do n+1 điểm trên tạo ra.

Thuật toán Casteljau

Để xây dựng đường cong P(t), ta dựa trên một dãy các điểm cho trước rồi tạo ra giá trị P(t) ứng với mỗi giá trị t nào đó.

Phương pháp này tạo ra đường cong dựa trên một dãy các bước nội suy tuyến tính hay ***nội suy khoảng giữa*** (In-Betweening).

Đường cong Bezier

Thuật toán Casteljau

Với 3 điểm P_0 , P_1 , P_2 có thể xây dựng một Parabol nội suy từ 3 điểm này bằng cách chọn một giá trị t ∈ [0, 1] rồi chia đoạn P_0P_1 theo tỉ lệ t, ta được điểm P_0^1 trên P_0P_1. Tương tự, chia tiếp P_1P_2 cũng theo tỉ lệ t, ta được P_1^1. Nối P_0^1 và P_1^1, lại lấy điểm trên $P_0^1P_1^1$ chia theo tỉ lệ t, được P_0^2. Tập điểm P_0^2 chính là đường cong p(t).

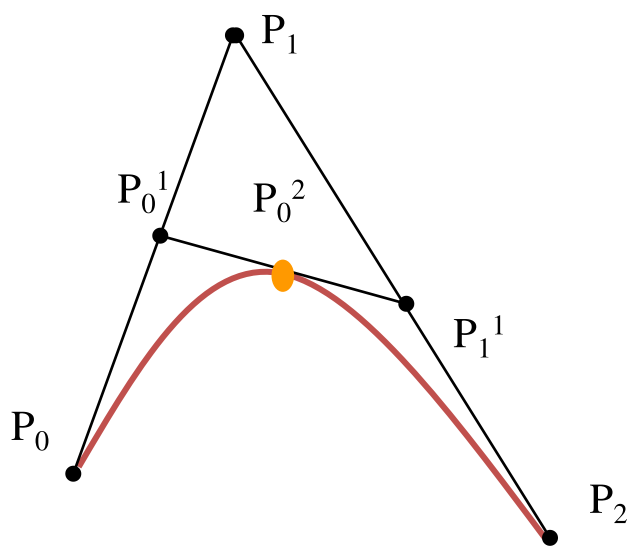
Biểu diễn bằng phương trình:

$P_0^1(t) = (1-t).P_0 + t.P_1$ (1)

$P_1^1(t) = (1-t).P_1 + t.P_2$ (2)

$P_0^2(t) = (1-t).P_0^1 + t.P_1^1$ (3)

Trong đó t ∈ [0, 1]



Thay (1), (2) vào (3) ta được:

$$\mathbf{P(t) = P_0^2(t) = (1-t)^2.P_0 + 2t.(1-t).P_1 + t^2.P_2}$$

Đây là một đường cong bậc 2 theo t nên nó là một Parabol.

Đường cong Bezier

☞Thuật toán Casteljau cho (n+1) điểm kiểm soát:

Giả sử ta có tập điểm: $P_0, P_1, P_2, ..., P_n$

Với mỗi giá trị t cho trước, ta tạo ra điểm $P_i^r(t)$ ở thế hệ thứ r, từ thế hệ thứ (r - 1) trước đó, ta có:

$$P_i^r(t) = (1-t).P_i^{r-1}(t) + t.P_{i+1}^{r-1}(t) \qquad (\, r=0,1,...,n \text{ và } i=0,...,n-r)$$

Thế hệ cuối cùng:

$$\boxed{P_0^n(t) = \sum_{i=0}^{n} P_i.(1-t)^{n-i}.t^i.C_n^i}$$

là **đường cong Bezier** của các điểm $P_0, P_1, P_2, ..., P_n$

Các điểm P_i, i=0,1,...,n gọi là các **điểm kiểm soát** (điểm Bezier).

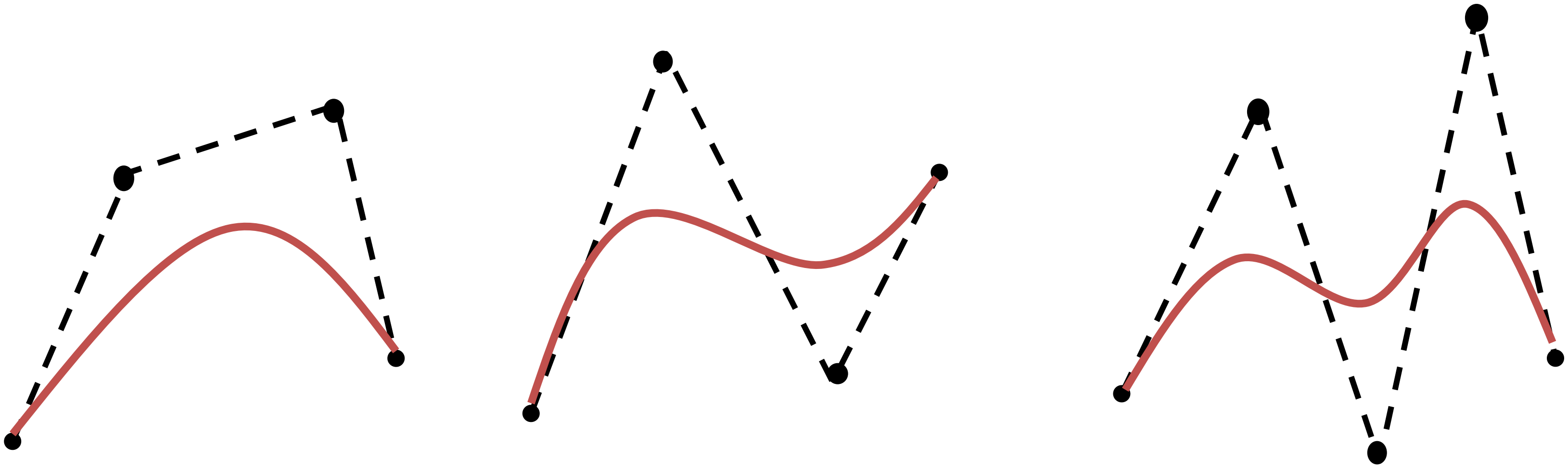
Đa giác tạo bởi các điểm kiểm soát gọi là **đa giác kiểm soát (**đa giác Bezier).

Đường cong Bezier

☞Thuật toán Casteljau cho (n+1) điểm kiểm soát:

Thế hệ cuối cùng:

$$P_0^n(t) = \sum_{i=0}^{n} P_i.(1-t)^{n-i}.t^i.C_n^i$$



Đường cong Bezier

Đường cong Bezier dựa trên (n+1) điểm kiểm soát P_0 ,P_1 , ...,P_n được cho bởi công thức:

$$P_0^n(t) = P(t) = \sum_{k=0}^{n} P_k . B_k^n(t)$$

Trong đó: P(t) là một điểm trong mặt phẳng hoặc trong không gian.

$B_k^n(t)$ gọi là đa thức Bernstein, được cho bởi công thức:

$$B_k^n(t) = C_n^k.(1-t)^{n-k}.t^k = \frac{n!}{k!.(n-k)!}.(1-t)^{n-k}.t^k$$ với $n \geq k$

Mỗi đa thức Bernstein có bậc là n. Thông thường ta còn gọi các $B_k^n(t)$ là các **hàm trộn** (blending function).

Đường cong Bezier

Đường cong Bezier dựa trên (n+1) điểm kiểm soát P_0 ,P_1 , ...,P_n được cho bởi công thức:

$$P_0^n(t) = P(t) = \sum_{k=0}^{n} P_k.B_k^n(t)$$

Tương tự, đối với mặt Bezier ta có phương trình sau:

$$P(u,v) = \sum_{i=0}^{m} \sum_{k=0}^{n} P_{i,k}.B_i^m(u)B_k^n(v)$$

Trong trường hợp này, khối đa diện kiểm soát sẽ có (m+1).(n+1) đỉnh.

Đường cong Bezier

Để tạo ra một đường cong Bezier từ một dãy các điểm kiểm soát ta áp dụng phương pháp lấy mẫu hàm p(t) ở các giá trị cách đều nhau của tham số t, ví dụ có thể lấy $t_i = i/m$, $i=0,1,...,m$.

Khi đó ta sẽ được các điểm $P(t_i)$ từ công thức Bezier. Nối các điểm này bằng các đoạn thẳng ta sẽ được đường cong Bezier gần đúng.

Đường cong Bezier

Minh họa việc vẽ đường cong Bezier trong mặt phẳng:

```
typedef  struct {
        int x;      int y ;
}CPoint;
int GiaiThua( int n){
        if (n == 0)  return 1;
        else      return n*GiaiThua(n - 1);
}
float HamMu(float a, int n){
        if (n==0)  return 1;
        else      return   a*HamMu(a,n-1);
}
```

Đường cong Bezier

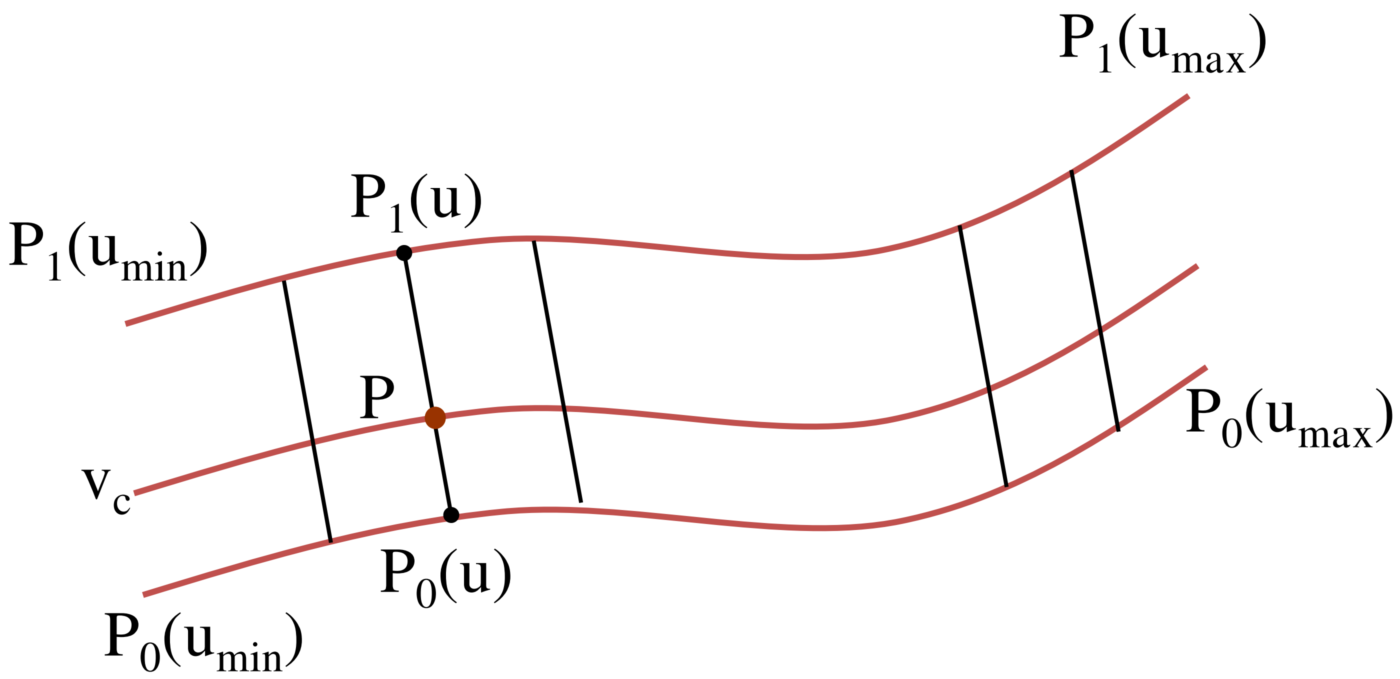
```
float BernStein(float t,int n, int k){
        float ckn,kq;
        ckn = GiaiThua(n)/(GiaiThua(k)*GiaiThua(n - k));
        kq = ckn*HamMu(1 - t,n - k)*HamMu(t,k);
        return kq;
}

CPoint TPt(CPoint P[ ],float t, int n){
        CPoint Pt;     float B;      int k;
        Pt.x=0;        Pt.y=0;
        for (k = 0; k<=n; k++){
                B = BernStein(t,n,k);
                Pt.x = Pt.x + P[k].x*B;   Pt.y = Pt.y + P[k].y*B;
        }
        return Pt;
}
```

Đường cong Bezier

```
void DrawBezier( int n, CPoint P[ ]){
        CPoint Pt;              float dt,t,m;
        int i;
        t=0;    m=100;          dt=1/m;
        moveto(P[0].x,P[0].y);
        for(i = 1; i<= m ; i++){
                Pt=TPt(P,t,n);
                lineto(Pt.x,Pt.y);
                t = t + dt;
        }
        lineto(P[n].x,P[n].y);
}
```

Mặt có quy tắc

là mặt được tạo bằng cách quét một đường thẳng trong không gian theo một cách nào đó.



Mặt có quy tắc

Phương trình tổng quát:

Gọi P_0P_1 là đoạn thẳng sinh ra mặt có quy tắc, và $P \in P_0P_1$

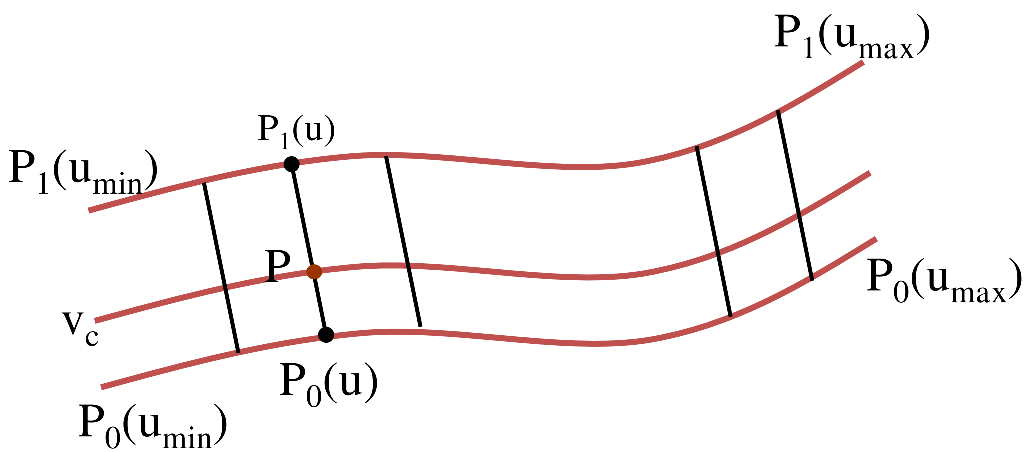
Ta có: $P(v)=(1-v)P_0+v.P_1$ với $v\in[0,1]$

Mà P_0 di chuyển trên đường $p_0(u)$

Mà P_1 di chuyển trên đường $p_1(u)$

Nên P thuộc mặt có quy tắc: $\mathbf{P(u,v)=(1-v).p_0(u)+v.P_1(u)}$

Với $u_{min}\le u\le u_{max}$ và $0\le v\le 1$



Mặt có quy tắc

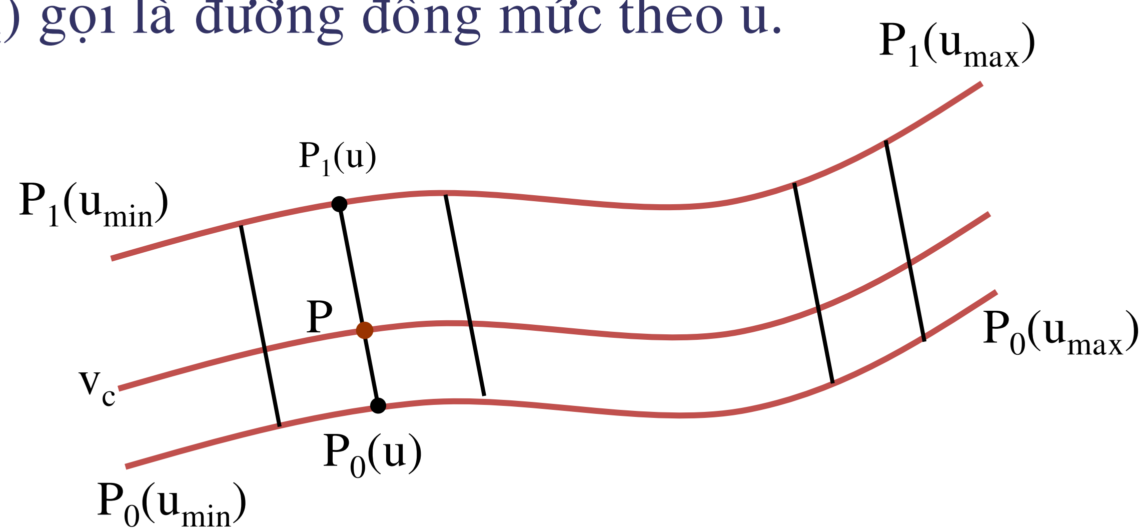
Đặc biệt:

Nếu v=0 và u thay đổi thì P di chuyển trên đường $p_0(u)$

Nếu v=1 và u thay đổi thì P di chuyển trên đường $p_1(u)$

Nếu $v=v_c$ và u thay đổi thì P di chuyển trên cung song song với $p_0(u_{min})$ $p_0(u_{max})$ gọi là đường đồng mức theo v.

Nếu $u=u_c$ và v thay đổi thì P di chuyển trên đoạn thẳng $p_0(u_c)$ $p_1(u_c)$ gọi là đường đồng mức theo u.



…Biểu diễn đường, mặt cong

Mặt có quy tắc

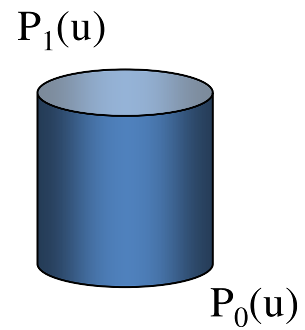
<u>Ví dụ</u>:

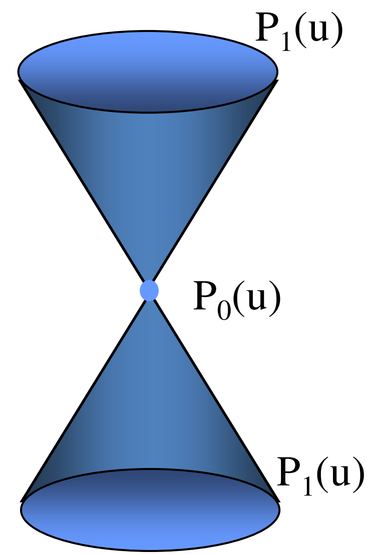
Mặt trụ: $p_0(u)$ là đáy dưới

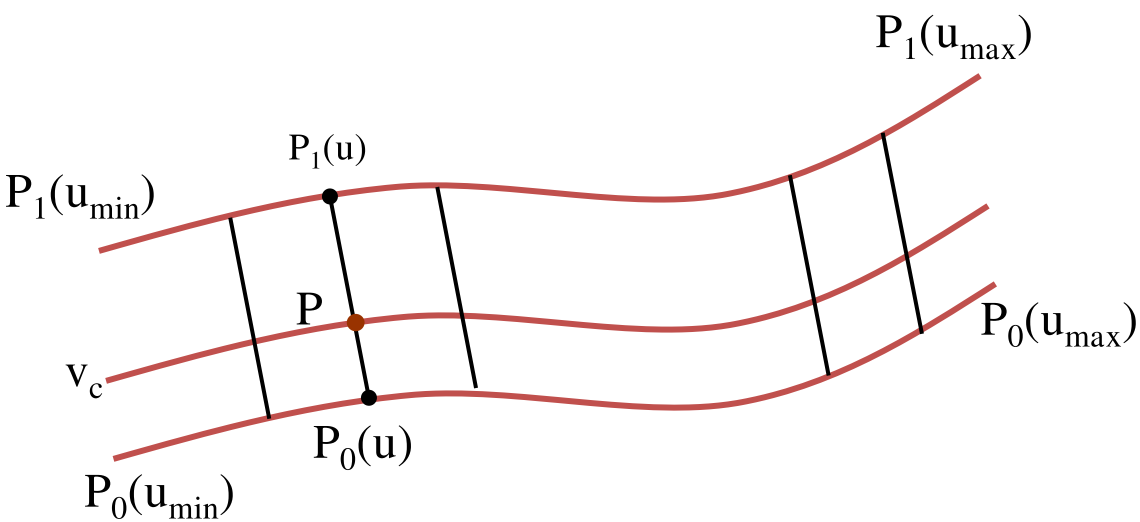
$P_1(u)$ là đáy trên

Mặt nón: $p_0(u)$ là 1 điểm

$P_1(u)$ là đường tròn







Mặt có quy tắc

Mặt trụ (Cylinder)

Mặt trụ là mặt được tạo ra khi một đường thẳng (đường sinh) được quét dọc theo một đườg cong $p_0(u)$ (đường chuẩn). Đường cong $p_0(u)$ nằm trên một mặt phẳng nào đó.

Gọi d là đường sinh, d=const.

$p_0(u)$ là đáy dưới

$P_1(u)$ là đáy trên

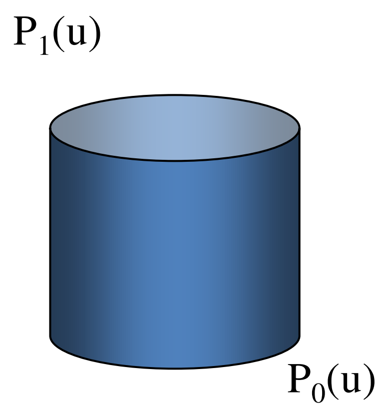
Suy ra: $d = P_1(u) - P_0(u)$

Ta có: $p(u,v) = (1-v).P_0(u)+v.\,P_1(u)$

$= P_0(u)+(P_1(u)-P_0(u)).v$

$= P_0(u)+d.v$

Vậy $p(u,v) = P_0(u)+d.v$



…Biểu diễn đường, mặt cong

Mặt có quy tắc

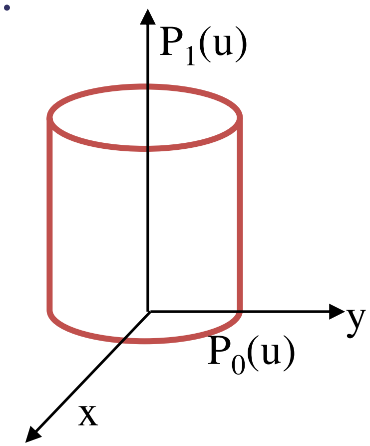
Mặt trụ (Cylinder)

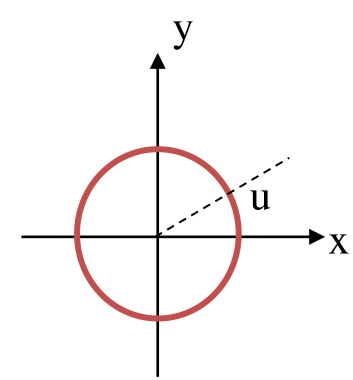
Pt mặt trụ: $p(u,v) = P_0(u)+d.v$

Dạng quen thuộc của mặt trụ là mặt trụ tròn: Trong mặt phẳng xOy, lấy $P_0(u)$ là đường tròn tâm O bán kính r .

Ta có: $d=(0,0,h)$

$P_0(u)=(r.\cos(u), r.\sin(u), 0)$





vậy: $\begin{cases} X(u,v) = r.\cos(u) \\ Y(u,v) = r.\sin(u) \\ Z(u,v) = h.v \end{cases}$ với $\begin{cases} 0 \le v \le 1 \\ 0 \le u \le 2\pi \end{cases}$

Mặt có quy tắc

```
void   DrawCylinder(float R, float h){
       Point3D P;          Point2D P1;
       double   Delta_U,Delta_V,u,v;
       Delta_U = 0.06;  Delta_V = 0.03;
       for (u=0; u<2*M_PI; u+=Delta_U)
       { for (v=0; v<1; v+=Delta_V)
               {        P.x = R*cos(u) ;
                        P.y = R*sin(u) ;
                        P.z = v*h;
                        P1 = Chieu(KieuChieu,P);
                        putpixel(xc+P1.x,yc+P1.y,WHITE);
               }
       }
}
```

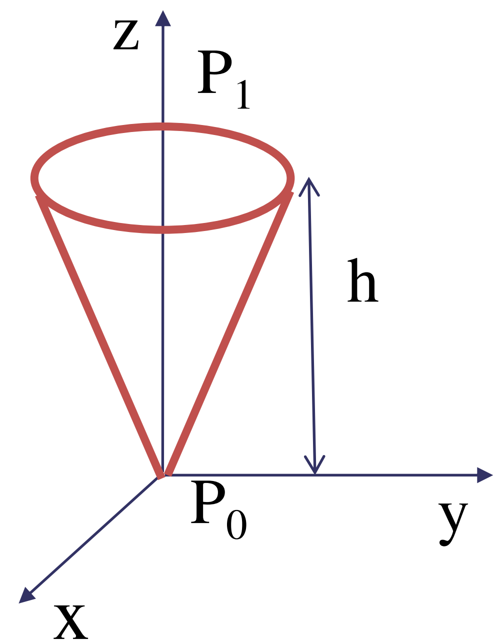
Mặt nón(Cone)

Mặt nón là mặt được tạo ra khi một đường thẳng di chuyển dọc theo một đường cong phẳng cho trước. Các đường thẳng luôn đi qua một điểm cố định gọi là đỉnh của mặt nón

$p_0(u)$ trùng với gốc toạ độ O

$P_1(u)$ là đường tròn tâm (0,0,h) bán kinh r.

$P_1(u)=(r.\cos(u), r.\sin(u), h)$



Ta có: $\mathbf{p(u,v)= (1\text{-}v).P_0(u)+v.\ P_1(u)= v.P_1(u)}$

Vậy $\mathbf{p(u,v) = v.P_1(u)}$

Hay: $\begin{cases} X(u,v) = v.r.\cos(u) \\ Y(u,v) = v.r.\sin(u) \\ Z(u,v) = v.h \end{cases}$ với $\begin{cases} 0 \le v \le 1 \\ 0 \le u \le 2\pi \end{cases}$

Mặt nón(Cone)

```
void DrawCone(float R, float h)
{       Point3D P;      Point2D P1;   double Delta_U,Delta_V,u,v;
        Delta_U = 0.03;  Delta_V = 0.1;
        for (u=0; u<2*M_PI ;u+=Delta_U)
        { for (v=0; v<1; v+=Delta_V)
                {P.x = v*R*cos(u);
                P.y = v*R*sin(u);
                P.z = v*h;
                P1 = Chieu(KieuChieu,P);
                putpixel(xc+P1.x,yc+P1.y,WHITE);
                }
        }
}
```

Mặt tròn xoay

ược tạo bằng cách quay đường cong C quanh một trục nào đó.

Giả sử: Đường cong C thuộc mặt phẳng xOz,

Trục quay là trục z

Gọi M(x, y, z) là một điểm thuộc C.

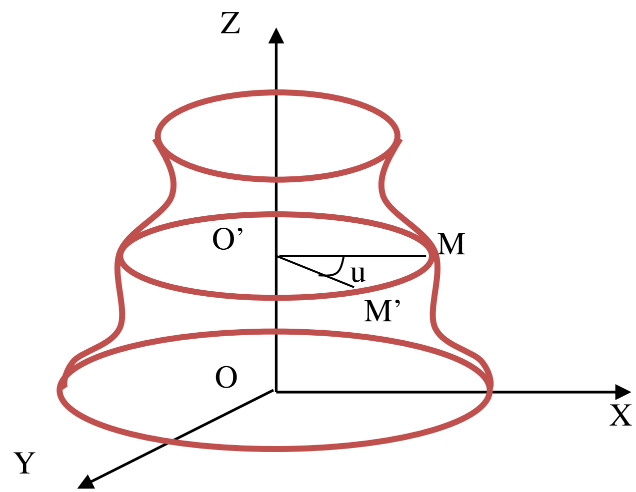
Ta có M(X(v),0,Z(v))

Cho M quay quanh trục z một góc u ⇨M thuộc mặt phẳng z=Z(v)

Gọi ảnh của M qua phép quay là M'(x, y, z)

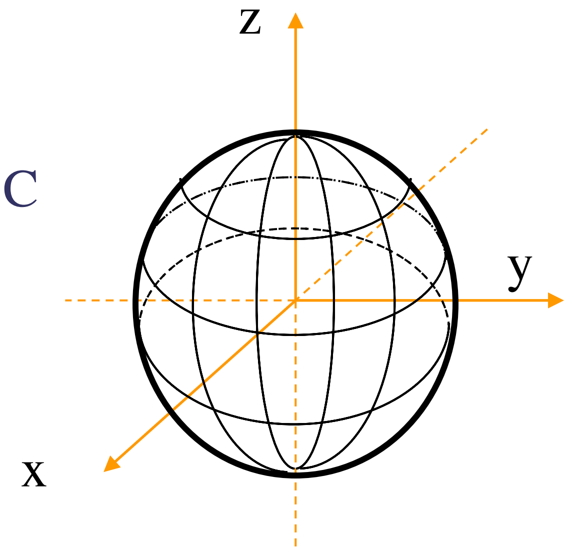
⇨Tính M'(x,y,z) qua X(v), Z(v) và u.

Ta có: O'M=O'M'=X(v)



Hay: $\begin{cases} x = X(v).\cos(u) \\ y = X(v).\sin(u) \\ z = Z(v) \end{cases}$ với $\begin{cases} v_{min} \le v \le v_{max} \\ 0 \le u \le 2\pi \end{cases}$

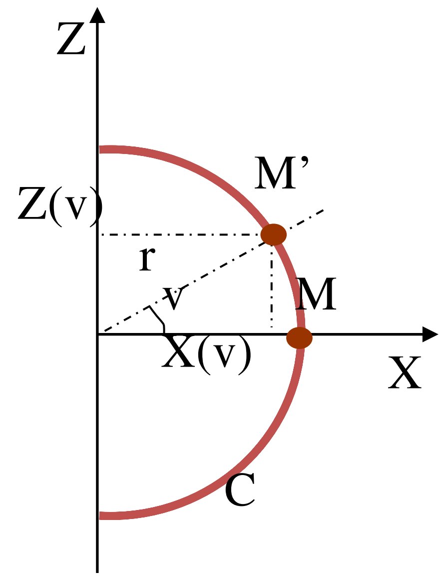
Mặt tròn xoay



Mặt cầu: Với mặt cầu tâm O bán kính r thì C là ½ đường tròn trong mặt phẳng xOz.

Ta có: X(v)=r.cos(v);

Z(v)=r.sin(v)



$$\begin{cases} X(u,v) = r.\cos(u).\cos(v) \\ Y(u,v) = r.\sin(u).\cos(v) \\ Z(u,v) = r.\sin(v) \end{cases}$$

với $\begin{cases} -\pi/2 \le v \le \pi/2 \\ 0 \le u \le 2\pi \end{cases}$

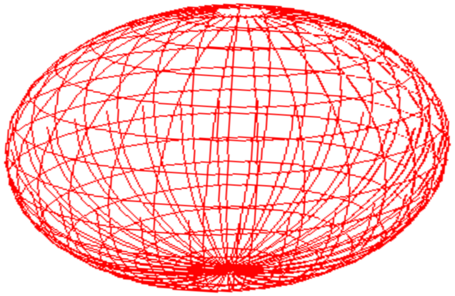
Mặt tròn xoay

Mặt cầu:

```
void DrawSphere(float R){
        Point3D P;   Point2D P1;  double Delta_U,Delta_V,u,v,Pi_2;
        Pi_2 = M_PI/2;   Delta_U = 0.1;    Delta_V = 0.1;
        for (v=-Pi_2; v<Pi_2 ;v+=Delta_V)
        { for (u=0; u<2*M_PI; u+=Delta_U)
                {        P.x = R*cos(u)*cos(v);
                         P.y = R*sin(u)*cos(v);
                         P.z = R*sin(v);
                         P1 = Chieu(KieuChieu,P);
                         putpixel(xc+P1.x,yc+P1.y,GREEN);
                }
        }
}
```

Mặt tròn xoay

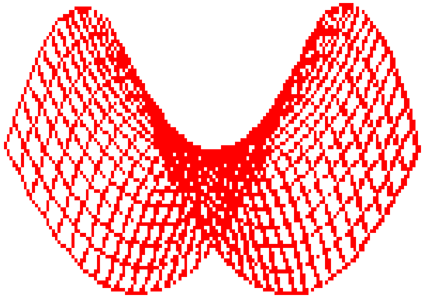
Mặt Ellipsoid:



Phương trình: $\begin{cases} X(u,v) = R_x.\cos(u).\cos(v) \\ Y(u,v) = R_y.\sin(u).\cos(v) \\ Z(u,v) = R_z.\sin(v) \end{cases}$

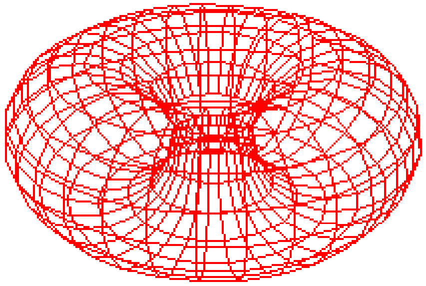
với $\begin{cases} -\pi/2 \le v \le \pi/2 \\ 0 \le u \le 2\pi \end{cases}$

Mặt Hypeboloid:



Phương trình: $\begin{cases} X(u,v) = u \\ Y(u,v) = v \\ Z(u,v) = u^2 - v^2 \\ -1 \le u, v \le 1 \end{cases}$

Mặt xuyến:



Phương trình: $\begin{cases} X(u,v) = (R + a.\cos(v)).\cos(u) \\ Y(u,v) = (R + a.\cos(v)).\sin(u) \\ Z(u,v) = a.\sin(v) \\ 0 \le u \le 2\pi; \quad -\pi/2 \le v \le \pi/2 \end{cases}$

Mặt tròn xoay

Phương pháp chính ở đây là vẽ các đường đồng mức theo u và v.

Để vẽ một đường đồng mức u tại giá trị u' khi v chạy từ V_{Min} đến V_{Max} ta làm như sau:

-Tạo một tập hợp các giá trị v[i] $\in$ [V_{Min} ,V_{Max}], xác định vị trí P[i] = (X(u',v[i]), Y(u',v[i]), Z(u',v[i])).

-Chiếu từng điểm P[i] lên mặt phẳng.

-Vẽ các đường gấp khúc dựa trên các điểm 2D P'[i].

…Biểu diễn đường, mặt cong

Mặt tròn xoay

TÓM LẠI: vẽ một mặt cong, ta thực hiện các bước sau

-Nhập các hệ số của phương trình mặt:
a, b, c, d, U_{min}, U_{max}, V_{min}, V_{max}.

-Tính các hàm 2 biến: X(u,v), Y(u,v), Z(u,v).

-Khởi tạo phép chiếu: Song song/Phối cảnh.

-Vẽ họ đường cong u.

-Vẽ họ đường cong v.

Thank You…!