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

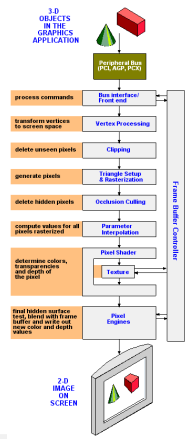
BÀI GIẢNG
ĐỒ HỌA MÁY TÍNH

Lưu hành nội bộ
Đà nẵng, 2023

Chương 2. BIẾN ĐỔI ĐỒ HỌA MÁY TÍNH

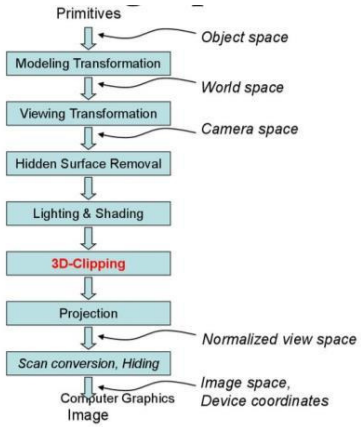
Quá trình hiển thị đồ họa 3D

- Các đối tượng trong thế giới thực phần lớn là các đối tượng 3D
- Thiết bị đồ họa hiển thị chỉ 2D.
- Do vậy, muốn có hình ảnh 3D ta cần phải giả lập trên thiết bị.
- Thực hiện chuyển đổi từng bước => Hình ảnh sẽ được hình thành từ từ, chi tiết hơn.



Các bước chính trong quá trình hiển thị đồ họa 3D

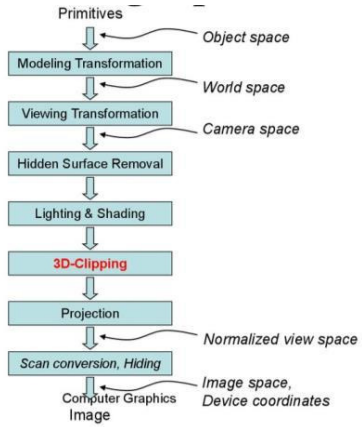
1) Modeling Transformation: Biến đổi mô hình Các phép ánh xạ tọa độ điểm hay vector thành tọa độ hay vector khác
2) Viewing Transformation: Biến đổi góc nhìn
3) Hidden Surfaces: Khử mặt khuất
4) Lighting & Shadow: Chiếu sáng và tô bóng đối tượng.
5) 3D Clipping: Xén đối tượng
6) Projection Transformation: Biến đổi hình chiếu
7) Biến đổi khung nhìn
8) Rasterization: Chuyển sang dạng pixel.
9) Display: Hiển thị hình ảnh trên màn hình 2D



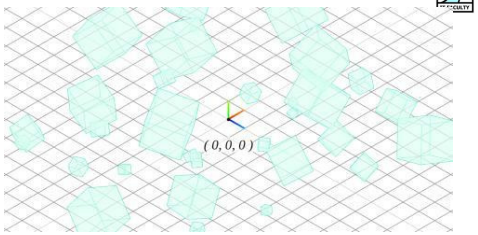
Các bước chính trong quá trình hiển thị đồ họa 3D

1) *Modelling transformation:*

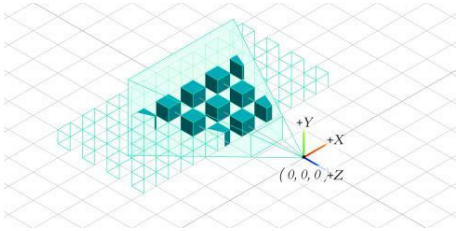
- Mỗi đối tượng được mô tả trong một hệ tọa độ riêng được gọi là hệ tọa độ đối tượng/Hệ tọa độ cục bộ.
- Thực hiện biến đổi từ hệ tọa độ đối tượng sang hệ tọa độ thế giới thực.
- Khi biểu diễn đối tượng, ta có thể chọn gốc tọa độ và đơn vị đo lường sao cho việc biểu diễn là thuận lợi nhất.
- Thường chuẩn hóa kích thước của đối tượng khi biểu diễn.



Các khối hình chữ nhật trong world space



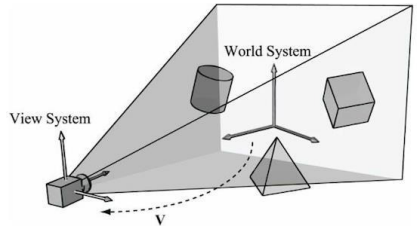
Biểu diễn các đối tượng trong camera space



Các bước chính trong quá trình hiển thị đồ họa 3D

2) Viewing Transformation

- Chuyển từ không gian thế giới thực (world space) sang không gian quan sát (eye/camera space).
- Thực hiện một phép biến đổi hệ tọa độ để đặt vị trí quan sát (viewing position) về gốc tọa độ và mặt phẳng quan sát (viewing plane) về một vị trí mong muốn.
- Hình ảnh hiển thị phụ thuộc vào vị trí quan sát và góc nhìn.
- Hệ qui chiếu có gốc đặt tại vị trí quan sát và phù hợp với hướng nhìn sẽ thuận tiện cho quá trình xử lý và hiển thị.



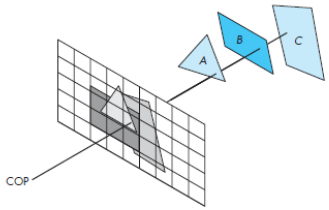
Các bước chính trong quá trình hiển thị đồ họa 3D

3) *Hidden Surface Removal:*

- Loại bỏ các đối tượng không nhìn thấy được (Trivial Rejection)
- Nhằm lược bỏ bớt các đối tượng không cần thiết do đó giảm chi phí xử lý.

4) *Lighting & Shading:*

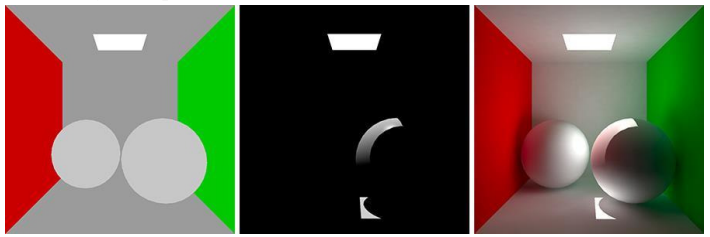
- Chiếu sáng các đối tượng (Illumination).
- Gán cho các đối tượng màu sắc dựa trên các đặc tính của các chất liệu và các nguồn sáng.
- Các mô hình chiếu sáng và tạo bóng : constant-intensity, Interpolate,...



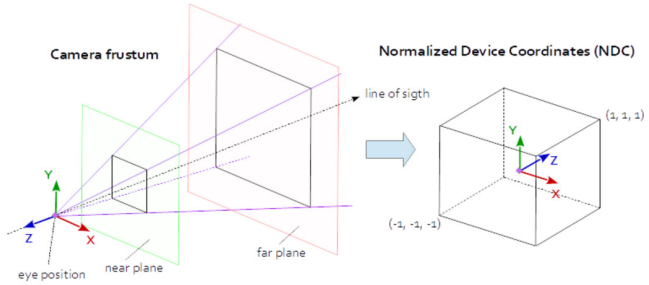
Các bước chính trong quá trình hiển thị đồ họa 3D

4) *Lighting & Shading:*

- Chiếu sáng các đối tượng (Illumination).
- Gán cho các đối tượng màu sắc dựa trên các đặc tính của các chất liệu và các nguồn sáng.
- Các mô hình chiếu sáng và tạo bóng : constant-intensity, Interpolate,...

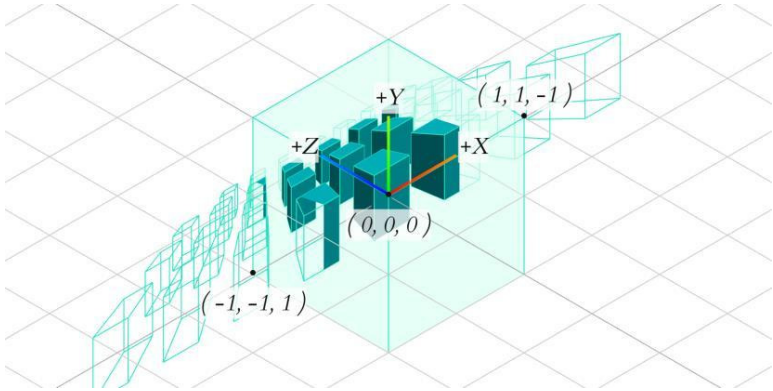


Các bước chính trong quá trình hiển thị đồ họa 3D



5) 3D Clipping: Loại bỏ phần nằm ngoài viewing frusturn.

- Thực hiện việc xén đối tượng trong cảnh để cảnh nằm gọn trong một phần không gian hình chóp cụt giới hạn vùng quan sát mà ta gọi là viewing frustum.
- Vùng quan sát viewing frustum có trục trùng với tia nhìn, kích thước giới hạn tùy theo đối tượng quan sát.

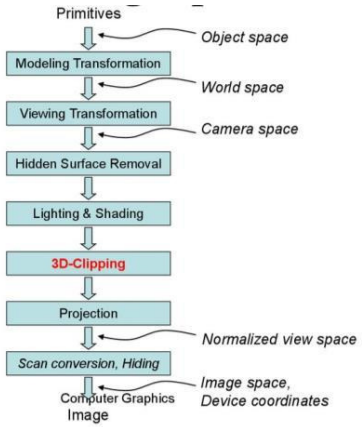


3D Clipping

Các bước chính trong quá trình hiển thị đồ họa 3D

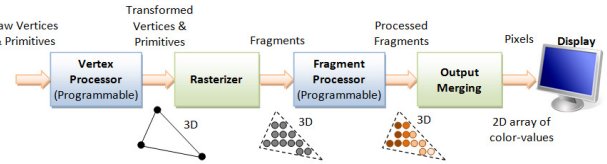
6) Projection transformation

- Chiếu từ không gian quan sát (*eye space*) xuống không gian màn hình (*screen space*).
- Thực hiện việc chiếu cảnh 3D từ không gian quan sát xuống không gian màn hình.
- Phân loại: Chiếu song song, Chiếu phối cảnh
- Thực hiện chiếu đồng thời khử mặt khuất để có thể nhận được hình ảnh trung thực.
- Khử mặt khuất cho phép xác định vị trí (x,y) trên màn hình thuộc về đối tượng nào trong cảnh.



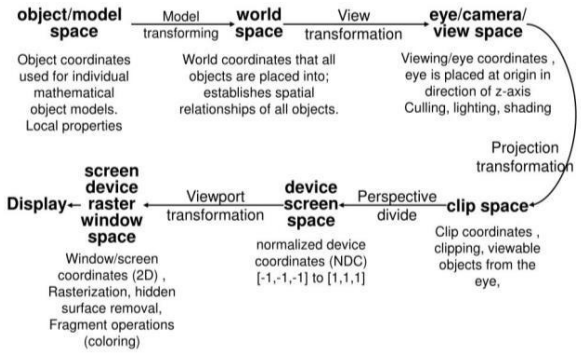
Các bước chính trong quá trình hiển thị đồ họa 3D

7) Chuyển đổi tượng sang dạng pixel (Rasterization)

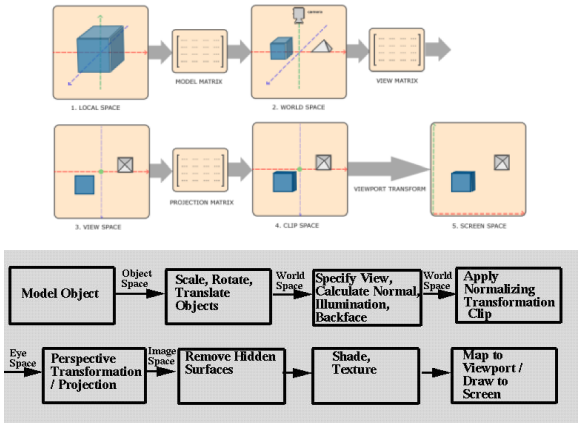


8) Hiển thị đối tượng (Display)

2.1 Các bước biến đổi mô hình và biến đổi hệ tọa độ



2.1.1 Các bước biến đổi mô hình

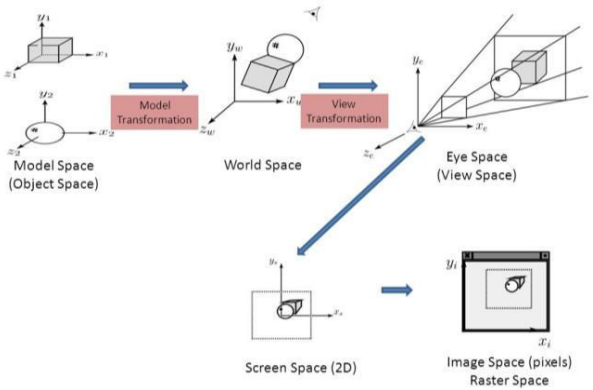


2.1.2 Quá trình biến đổi hệ tọa độ

- Hệ tọa độ đối tượng/mô hình (object/model coordinates):

Khi thiết kế đối tượng, thường gốc tọa độ đặt tại tâm của đối tượng.

Các tọa độ được biến đổi sang không gian toàn cảnh với biến đổi mô hình (modeling transformation, world transformation) với ma trận $M_{object \rightarrow world}$

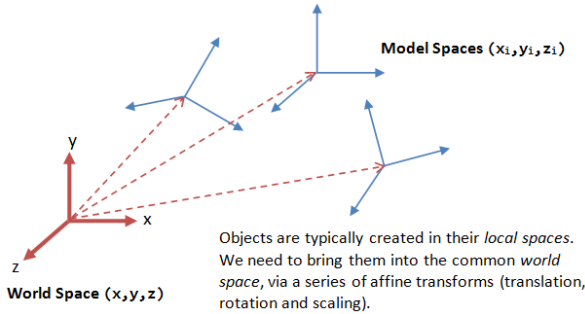


- Hệ tọa độ thế giới thực (world coordinates):

Các đối tượng được đặt vào vị trí trong cảnh do người lập trình quy định.

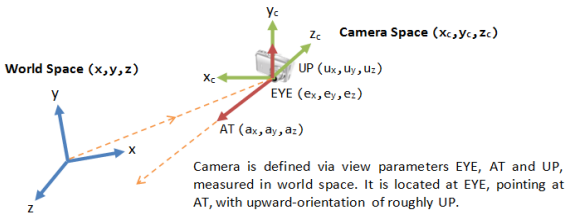
Cảnh phải được nhìn thấy bởi người xem dưới góc độ quan sát của họ.

Thực hiện bởi biến đổi view hay camera (viewing/camera transformation), với ma trận $M_{world \rightarrow view}$



- Hệ tọa độ quan sát (*view/eye/camera coordinates*):
  - Gốc tọa độ đặt tại mắt người quan sát
  - Một *hướng* ngắm (nhìn vào tâm điểm)
  - Một hướng lên trên (up)
  - Một hướng khởi tạo để qui định mắt người quan sát hướng về phía nào trong không gian toàn cảnh).

Đối tượng 3D được hiển thị trên màn ảnh 2D. Do đó cần có biến đổi phép chiếu với ma trận $M_{projection}$



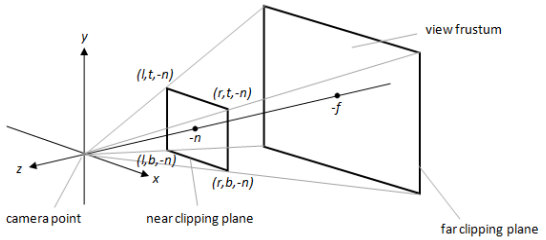
Camera is defined via view parameters EYE, AT and UP, measured in world space. It is located at EYE, pointing at AT, with upward-orientation of roughly UP.

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.

- Hệ tọa độ clip (*clip coordinates*):

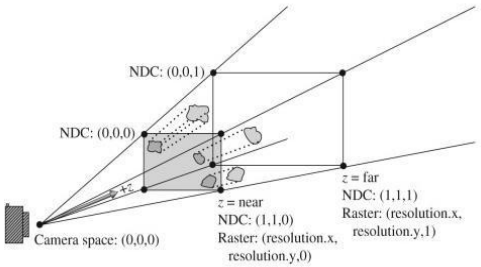
Biến đổi phép chiếu thực hiện cắt và giới hạn cảnh xem vào một hình chóp cụt, thường xác định bởi: góc nhìn theo chiều thẳng đứng (fovy - field of view in y direction), tỉ lệ màn hình (hay cửa sổ chương trình), tỉ lệ này xác định góc nhìn ngang, mặt phẳng clip gần và mặt phẳng clip xa.

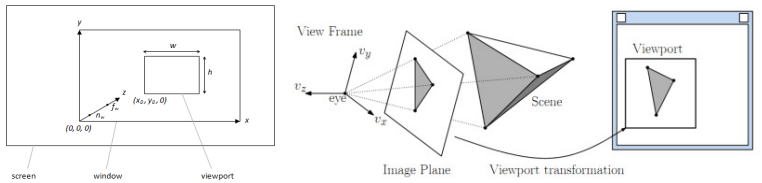
Phép chiếu phải ánh xạ tọa độ camera vào tọa độ màn ảnh trong các khoảng [*-1,1*].



- Hệ tọa độ thiết bị chuẩn hóa (Normalized Device Ccoordinates - NDC):

OpenGL thực hiện chuẩn hóa tọa độ bằng cách chia các tọa độ thành phần của biến *gl_Position* bởi thành phần thứ tư của nó là *gl_Position.w*. Tại đây, tất cả các tọa độ *x, y, x* đều nằm trong khoảng [-1,1], được gọi là tọa độ thiết bị chuẩn hóa.

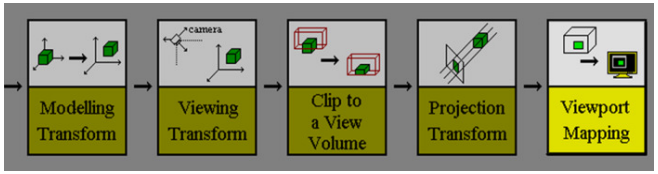




- Hệ tọa độ màn ảnh/cửa sổ (screen/window coordinates): Tọa độ cuối cùng là tọa độ màn ảnh hay cửa sổ, với biến đổi cuối cùng là biến đổi khung nhìn (viewport transformation) => Ma trận biến đổi.

3D Model ⇒World Space ⇒ Camera ⇒ Projection

2.2 Overview OpenGL viewing pipeline

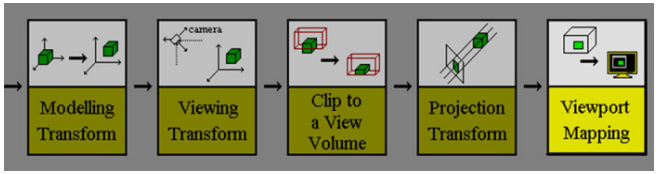


1) Modelview matrix – Projection matrix using:

glMatrixMode(GL_MODELVIEW): *Model transform, View transform*

- glTranslate()
- glRotate()
- glScale()
- glLoadMatrix()
- glMultMatrix()
- gluLookAt(x0, y0, z0, xref, yref, zref, Vx, Vy, Vz)

Overview OpenGL viewing pipeline



4) Projection matrix:

glMatrixMode (GL_PROJECTION): *Projection transform, Clipping, Perspective divide*

- gluOrtho2D (xwmin, xwmax, ywmin, ywmax)
- gluPerspective (theta, aspect, dnear, dfar)
- glFrustum (xwmin, xwmax, ywmin, ywmax, dnear, dfar)

5) Viewport matrix:

- glViewport(xPos, yPos, xSize, ySize)
- glutInitWindowSize(*width, height*)
- glutInitWindowPosition(*x, y*)
- lDepthRangef()

Chương 3. CÁC PHÉP BIẾN ĐỔI TRONG ĐỒ HỌA

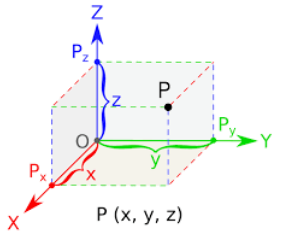
3.1 Các phép biến đổi đồ họa 2D

3.2 Tọa độ đồng nhất (Homogeneous)

- Cho phép các phép biến đổi Affine có thể được biểu diễn dễ dàng bằng một ma trận.
- Giúp cho việc tính toán có thể thực hiện trong không gian xạ ảnh, giống như là hệ tọa độ Descartes trong không gian Euclide

- Trong phép biến đổi hình học Affine, tọa độ điểm được mô tả dưới ma trận [x^* y^* h]. Trong đó $x = x^*/h$, $y = y^*/h$, $z = z^*/h$ và h là một số thực tuỳ ý
- Cho phép biểu diễn hợp nhất của các phép biến đổi dưới phép nhân ma trận, hỗ trợ cho việc xử lý bằng cả phần cứng và phần mềm: kết hợp với cả các phép biến đổi đặc biệt không tuyến tính khác (non-affine) phép chiếu phối cảnh (Perspective projections), bends, tapers...

- Biểu diễn đồng nhất $P = (wP_x, wP_y, wP_z, w)$, các điểm nằm trên cùng tia, điểm ở vô cùng có w=0.
- Tọa độ thông thường → Tọa độ đồng nhất (thêm 1)
- Tọa độ đồng nhất → Tọa độ thông thường (chia cho [illegible] đi thành phần tọa độ thứ 4)



Cách chuyển đổi:

Cho điểm $P = (P_x, P_y, P_z)$, Biểu diễn tọa độ đồng nhất của P: $(P_x, P_y, P_z, 1)$

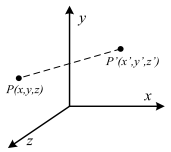
Cho vector $v = (v_x, v_y, v_z)$, Biểu diễn tọa độ đồng nhất của v: $(v_x, v_y, v_z, 0)$

Ví dụ: P(*3, 6, 2, 3*) → tọa độ thông thường là (*1, 2, 2/3*)

3.3 Các phép biến đổi đồ họa 3D (MODEL TRANSFORMATION)

3.3.1 Phép tịnh tiến

- Biến đổi điểm $P(x, y, z, 1) \rightarrow P'(x', y', z', 1)$ thông qua (t_x, t_y, t_z):



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [M_T] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Hình 3.14. Phép tịnh tiến 3D

- t_x, t_y, t_z :biểu diễn giá trị tịnh tiến theo các hướng x, y, z.

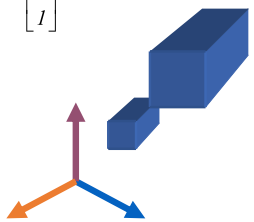
3.3.2 Phép biến đổi tỉ lệ

Điểm $P(x, y, z, 1)$ được biến đổi tỉ lệ thành $P'(x', y', z', 1)$:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [M_S] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Nếu $s_x \neq s_y \neq s_z$: biến đổi méo.

Nếu $s_x = s_y = s_z$: biến đổi đồng dạng



Dùng phép tỉ lệ có ma trận biến đổi đồng dạng như sau:

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

Biến đổi ma trận để cho các tọa độ (x, y, z) thành các tọa độ Descartes chuẩn.

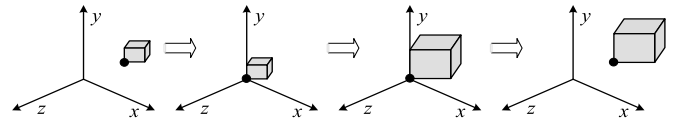
Biển diễn lại ma trận trên:

$$\begin{bmatrix} x \\ y \\ z \\ s \end{bmatrix} = \begin{bmatrix} x/s \\ y/s \\ z/s \\ 1 \end{bmatrix}$$

Nếu $s > 1$ thì sẽ giảm kích thước đối tượng và ngược lại.

Các bước biến đổi tỉ lệ một đối tượng:

- Tịnh tiến đối tượng sao cho điểm xác định trùng với điểm gốc.
- Biến đổi tỉ lệ đối tượng theo tọa độ gốc;
- Dịch chuyển ngược đối tượng về vị trí cũ.



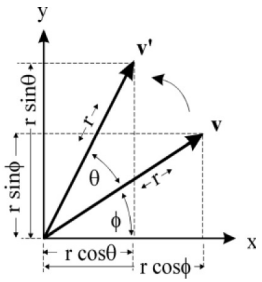
a) Vị trí gốc *b) Tịnh tiến* *c) Tỉ lệ* *d) Tịnh tiến ngược*

Hình 3.15. Phép biến đổi tỉ lệ tại một điểm trên đối tượng

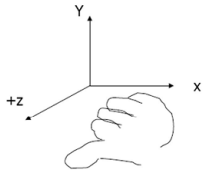
$$[M] = [M_T]^{-1}[M_S][M_T] = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3.6.3. *Phép quay 3D*

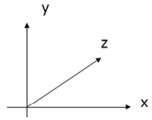
- Quay quanh các trục *Ox*, *Oy*, *Oz* hoặc quanh một trục bất kì
- Tịnh tiến kết hợp với phép quay quanh trục tọa độ.
- Xét phép quay cơ sở theo chiều dương (ngược chiều kim đồng hồ) trong hệ trục tọa độ theo quy tắc bàn tay phải.



x × y = z; y × z = x; z × x = y;



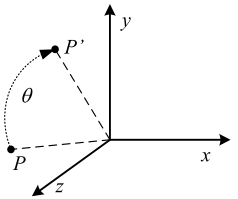
Right hand coordinate system



Left hand coordinate system
Not used in this class and
Not in OpenGL

3.6.3.1. *Phép quay quanh trục Ox*

Điểm $P(x, y, z, 1)$ khi quay quanh trục Ox một góc $\theta \rightarrow P'(x', y', z', 1)$

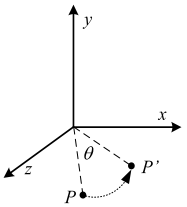


$$x' = x$$
$$y' = y.\cos\theta - z.\sin\theta$$
$$z' = y.\sin\theta + z\cos\theta$$
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [M_R]_X^\theta . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Hình 3.16. Phép quay quanh trục Ox

3.6.3.2. *Phép quay quanh trục Oy*

Điểm $P(x, y, z, 1)$ khi quay quanh trục Oy một góc $\theta \rightarrow P'(x', y', z', 1)$

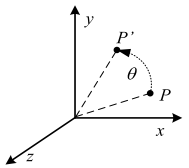


$$x' = x.\cos\theta + z.\sin\theta$$
$$y' = y$$
$$z' = -x.\sin\theta + z.\cos\theta$$
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [M_R]_Y^\theta . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Hình 3.17. Phép quay quanh trục Oy

3.6.3.3. *Phép quay quanh trục Oz*

Điểm $P(x, y, z, 1)$ khi quay quanh trục Oz một góc $\theta \rightarrow P'(x', y', z', 1)$



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

Hình 3.18. Phép quay quanh trục Oz

Ví dụ 01: Tổng hợp các phép biến đổi.

Nếu một biến đổi mô hình bao gồm cả phép quay, phép tỉ lệ và phép tịnh tiến thì:

$M_{object \to world} = M_{translation} M_{scaling} M_{rotation}$

Ví dụ 02: Quay điểm P(0.5, 0.5, 0) một góc $\alpha = 90^o$ quanh trục Oz.

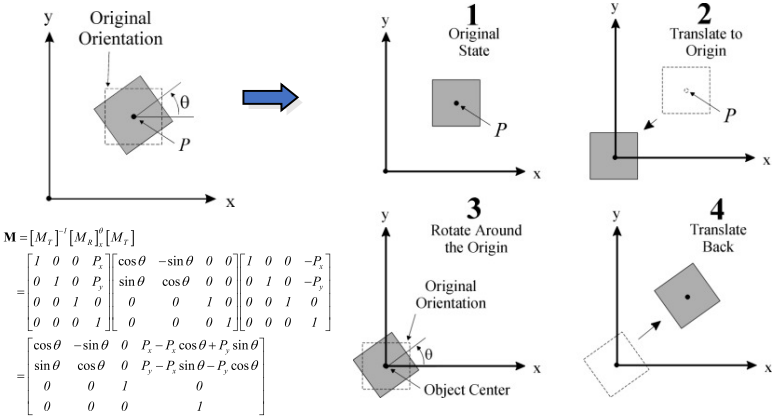
$$\begin{bmatrix} \cos\left(90 \cdot \frac{\pi}{180}\right) & -\sin\left(90 \cdot \frac{\pi}{180}\right) & 0 & 0 \\ \sin\left(90 \cdot \frac{\pi}{180}\right) & \cos\left(90 \cdot \frac{\pi}{180}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix}$$

Kết quả được điểm P'(-0.5, 0.5, 0)

Dịch chuyển điểm P(-0.5, 0.5, 0) theo vectơ T(0.1, -0.2, 0.5)

$$\begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & -0.2 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.4 \\ 0.3 \\ 0.5 \\ 1 \end{bmatrix}$$

Ví dụ 03: Quay một đối tượng quanh tâm (x, y) góc θ



$$\mathbf{M} = [M_T]^{-1} [M_R]_z^{\theta} [M_T]$$

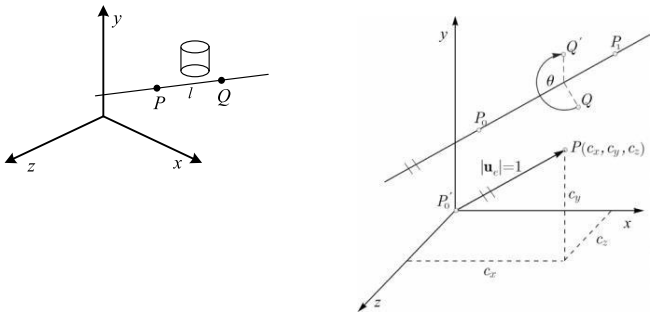
$$= \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 & P_x - P_x\cos\theta + P_y \sin\theta \\ \sin\theta & \cos\theta & 0 & P_y - P_y \sin\theta - P_y \cos\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.6.3.4. Phép quay quanh trục bất kì

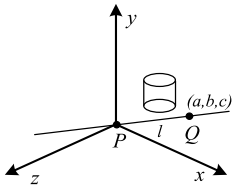
Cho điểm $P_0(x_0, y_0, z_0)$, $P_1(x_1, y_1, z_1)$.

Quay một điểm Q một góc θ quanh trục là đường thẳng P_0P_1 theo hướng nhìn từ $P_1 \to P_0$

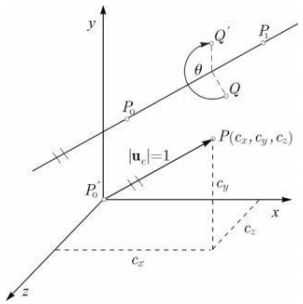


Hình 3.19. Quay quanh trục là đường thẳng PQ

Bước 1: Tịnh tiến điểm P trùng với gốc O: ma trận tịnh tiến $[M_T]$

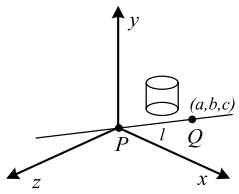


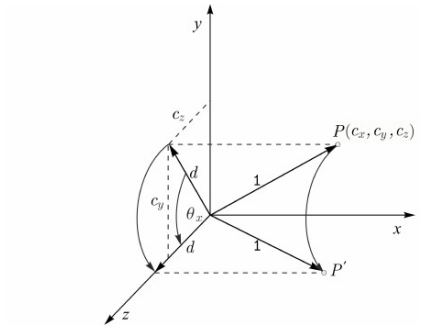
$$[M_T] = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Hình 3.20. Tịnh tiến trục PQ về gốc tọa độ

Bước 2: Quay *PQ* quanh trục *Ox* một góc *θx* để *PQ* nằm trên mặt *xz*: ma trận quay $[M_R]_X^{\theta x}$



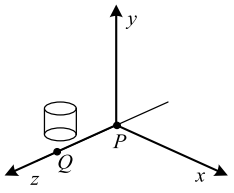


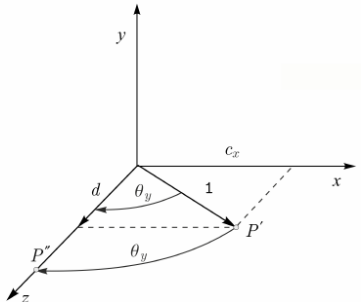
$$[M_R]_X^{\theta_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c_z}{d} & \frac{-c_y}{d} & 0 \\ 0 & \frac{c_y}{d} & \frac{c_z}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$d = \sqrt{c_y^{\,2} + c_z^{\,2}}$$

Hình 3.21. Quay PQ quanh trục Ox

Bước 3: Quay *PQ* quanh trục *Oy* một góc *-θy* để *PQ* nằm trên trục *Oz:* ma trận quay $[M_R]_Y^{-\theta y}$

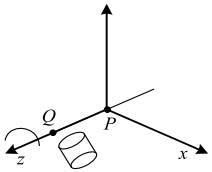


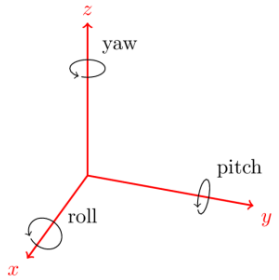


$$[M_R]_Y^{-\theta_y} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{d}{l} & 0 & \frac{-c_x}{l} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{c_x}{l} & 0 & \frac{d}{l} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hình 3.21. Quay trục PQ quanh trục Ox, Oy để cùng hướng với trục Oz

Bước 4: Quay quanh trục *Oz* một góc *θz*

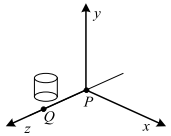




$$[M_R]_Z^{\theta z} = \begin{bmatrix} \cos\theta z & -\sin\theta z & 0 & 0 \\ \sin\theta z & \cos\theta z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hình 3.22. Quay quanh trục Oz một góc θ.

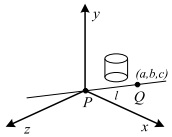
Bước 5: Quay lại *PQ* ngược lại quanh trục *Oy* một góc *θy*



$$[M_R]_Y^{\theta_y} = \begin{bmatrix} \cos\theta y & 0 & \sin\theta y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta y & 0 & \cos\theta y & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{d}{l} & 0 & \frac{-cx}{l} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{cx}{l} & 0 & \frac{d}{l} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hình 3.21. Quay ngược PQ quanh trục Oy

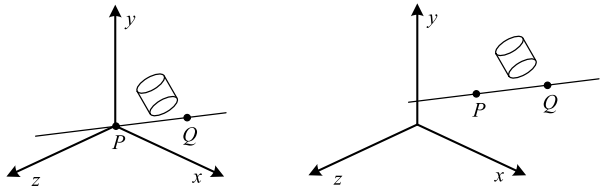
Bước 6: Quay *PQ* ngược lại quanh trục *Ox* một góc *-θx*



$$[M_R]_X^{\theta_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta x & -\sin\theta x & 0 \\ 0 & \sin\theta x & \cos\theta x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{cz}{d} & \frac{-cy}{d} & 0 \\ 0 & \frac{cy}{d} & \frac{cz}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hình 3.21. Quay ngược PQ quanh trục Ox

Bước 8: Tịnh tiến *PQ* về lại vị trí ban đầu: ma trận tịnh tiến $[M_T]^{-1}$



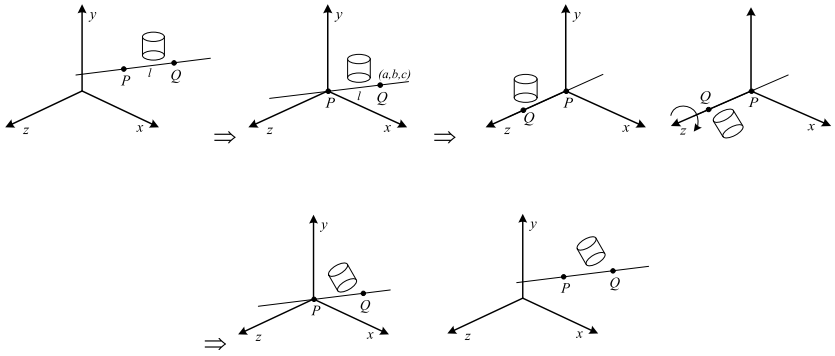
a) Quay trục PQ trở về hướng ban đầu b) Tịnh tiến PQ trở về vị trí cũ

Hình 3.23. Quay và tịnh tiến trục PQ về vị trí ban đầu

$$[M_T] = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ma trận biến hình là tích các ma trận thành phần.

$$[M_R]_{PQ}^{\theta} = [M_T]^{-1}[M_R]_X^{\theta x}[M_R]_Y^{-\theta y}[M_R]_Z^{\theta x}[M_R]_Y^{-\theta y}[M_R]_X^{\theta x}[M_T]$$



Ví dụ: Cho P_0(6, -2, 0), P_1(12, 8, 0). Quay điểm Q(10, 6, 0, 1) một góc 60° quanh trục P_0P_1 theo hướng từ $P_0 \rightarrow P_1$.

Ta có:

cx = 12 – 6 = 6; cy = 8 - (-2) = 10; cz = 0 – 0 = 0.

d = sqrt(10*10 + 0*0) = 10;

l = sqrt(6*6 + 10*10+0*0) = 11.6619.

Ma trận tổng hợp: $[M_R]_{PQ}^{\theta} = [M_T]^{-1}[M_R]_X^{\theta x}[M_R]_Y^{-\theta y}[M_R]_Z^{\theta x}[M_R]_Y^{-\theta y}[M_R]_X^{\theta x}[M_T]$

0.632	0.221	0.743	2.647
0.221	0.868	-0.446	-1.588
-0.743	0.446	0.500	5.347
0.000	0.000	0.000	1.000

Kết quả: Q*(10.294, 5.824, 0.594, 1.000)

3.4 Phép đối xứng

- Đối xứng qua mặt phẳng xOy

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

3.5 Phép biến dạng

Biến dạng theo bất kỳ trục toạ độ nào cũng bị ảnh hưởng bởi toạ độ tương ứng với hai trục còn lại. Tâm là gốc tọa độ.

$$\begin{bmatrix} 1 & h_{yx} & h_{zx} & 0 \\ h_{xy} & 1 & h_{zy} & 0 \\ h_{xz} & h_{yz} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.6 Phép biến đổi Affine với OpenGL

3.6.1 Biểu diễn ma trận

- Trong OpenGL các điểm được biểu diễn dưới hệ tọa độ thuần nhất.
- Tọa độ của một điểm 3D được thể hiện bởi $(x, y, z, w)^T$, thông thường $w = 1$
  Chú ý: *Cách biểu diễn vector điểm ở dạng cột*.
- Một phép biến đổi trên một điểm *P(x, y, z, w)* tương ứng với việc nhân ma trận cột *P* với ma trận biến đổi *M* kích thước 4x4: *P' = M.P*
- Trong mỗi bước ModelView và Projection, tại mỗi thời điểm, OpenGL đều lưu trữ một ma trận biến đổi hiện hành.
- Để thực thi bước ModelView, cần gọi hàm: *glMatrixMode(GL_MODELVIEW)*
- Để thực thi bước Projection, cần gọi hàm: *glMatrixMode(GL_PROJECTION)*

Để thiết lập ma trận biến đổi hiện hành bằng ma trận M, dùng hàm:
void **glLoadMatrix**{fd}(const TYPE *m);
Ma trận M có dạng

$$M = \begin{bmatrix} m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \\ m_4 & m_8 & m_{12} & m_{16} \end{bmatrix}$$

Sau khi thay đổi ma trận hiện hành, nhưng sau đó muốn khôi phục lại nó. Ví dụ như dời tới một điểm nào đó để vẽ khối hộp, sau đó muốn trở lại vị trí ban đầu. Để hỗ trợ các thao tác lưu trữ ma trận hiện hành, OpenGL sử dụng stack cho mỗi loại ma trận hiện hành, với các hàm sau:

- Đẩy ma trận hiện hành vào trong stack: `void glPushMatrix(void)`
- Lấy ma trận hiện hành ở đỉnh stack: `void glPopMatrix(void)`

3.6.2 Các phép biến đổi Affine trong OpenGL

a) Translation: glTranslate(dx, dy, dz);*

where [dx, dy, dz] is the translation vector.
Mmodelview = Mmodelview * T(dx, dy, dz);

Where T(dx, dy, dz) = $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

b) Rotation: glRotate(angle, x, y, z)*

glRotatef(a, 1, 0, 0);
$M_{modelview} = M_{modelview} * M_{RX}(a)$;
Where Rx(a) denote the rotation matrix about the x-axis for degree a:

$M_{RX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) & 0 \\ 0 & \sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

c) Scaling: glScale(sx, sy, sz);*

glScalef(1, 3, 1);

- Nhân ma trận M có thể được thực hiện bởi hàm: **void glMultMatrix();**

Chú ý:

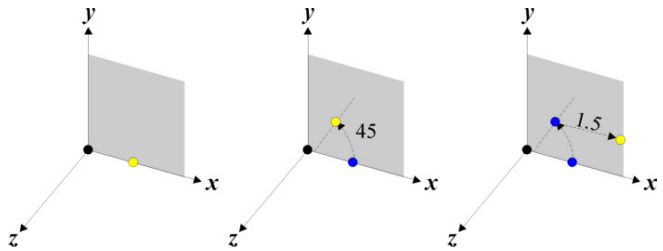
- Các thao tác biến đổi trên đều có nghĩa là lấy ma trận biến đổi hiện hành nhân với ma trận biến đổi affine cần thực hiện.
- Thứ tự thực hiện sẽ *ngược* với suy nghĩ thông thường: Lý do là tọa độ được biểu diễn dạng vector cột – Chú ý $(AB)^T = B^T A^T$)

Ví dụ: Thực hiện phép quay quanh trục Oz một góc α và tịnh tiến đi một đoạn theo vector (tr_x, tr_y, tr_z), các bước thực hiện sẽ là

Thao tác	Ma trận hiện hành
Khởi tạo glMatrixMode(GL_MODELVIEW) glLoadIdentity()	$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$
Phép tịnh tiến: glTranslatef(tr_x, tr_y, tr_z)	$\begin{bmatrix} 1 & & & tr_x \\ & 1 & & tr_y \\ & & 1 & tr_z \\ & & & 1 \end{bmatrix}$
Phép quay glRotatef(α, 0, 0, 1)	$\begin{bmatrix} \cos\alpha & -\sin\alpha & & tr_x \\ \sin\alpha & \cos\alpha & & tr_y \\ & & 1 & tr_z \\ & & & 1 \end{bmatrix}$

Ví dụ:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(1.5, 0.0, 0.0);
glRotatef(45.0, 0.0, 0.0, 1.0);
glVertex3f(1.0, 0.0, 0.0);
```



BÀI TẬP:

1) Lập trình C/C++ xây dựng các hàm sau để tích hợp trong chương trình OpenGL:

- **myTranslatef(x, y, z);**
- **myScalef(x, y, z);**
- **myRotatef(α, x, y, z);**