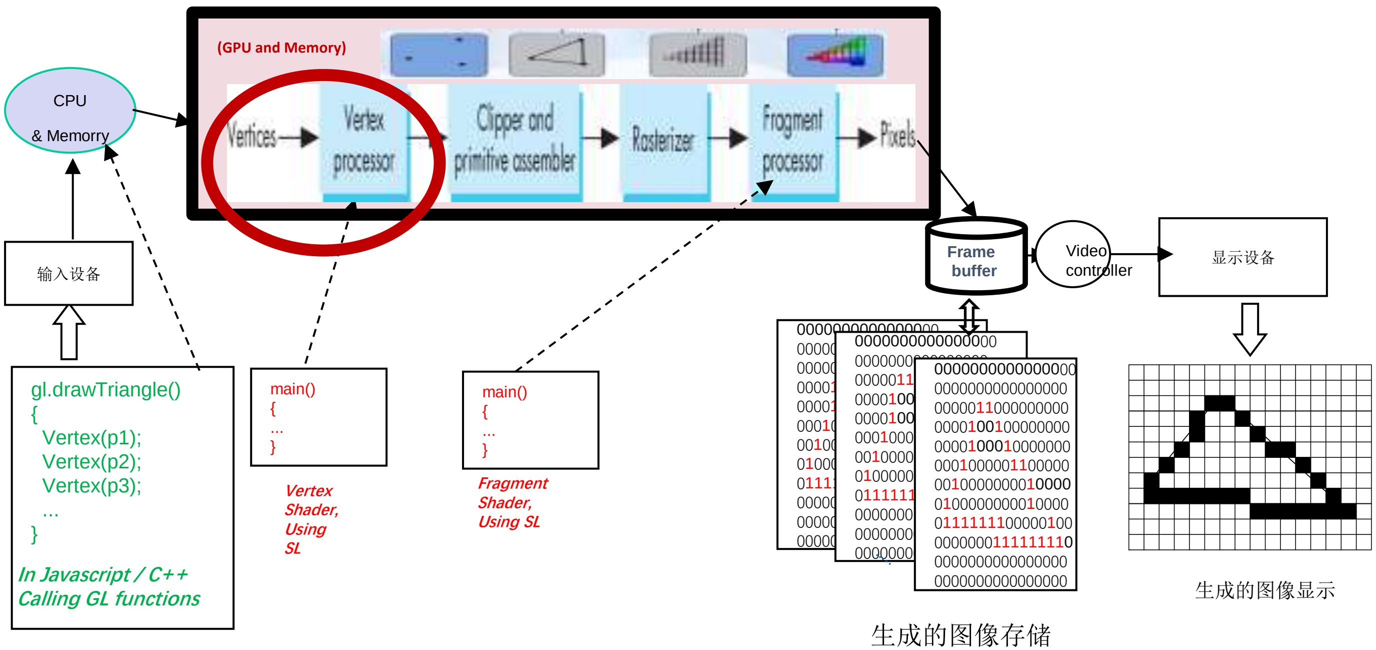
The University of New Mexico

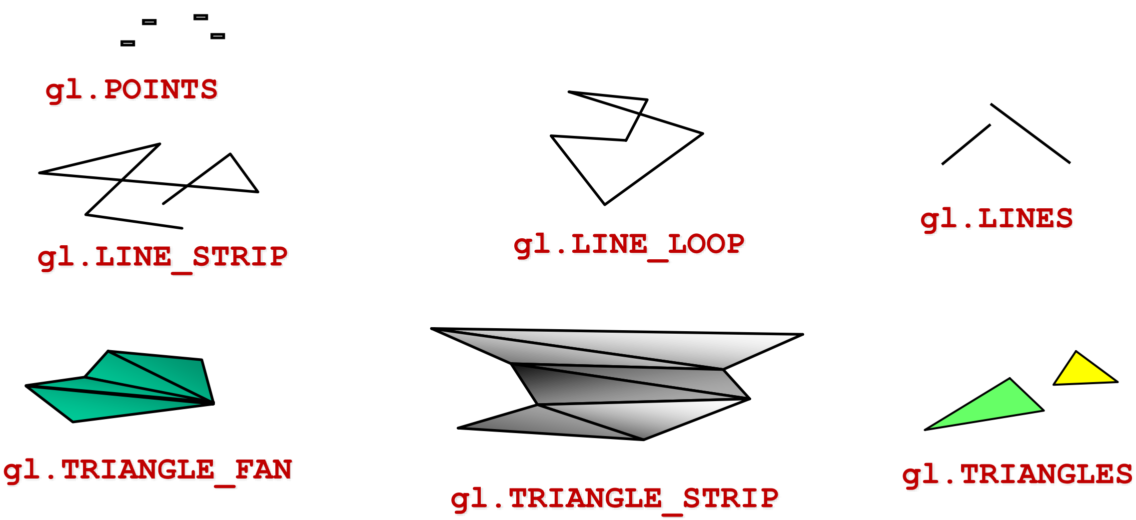
# Recap

- ➢The Programmable Rendering Pipeline 可编程渲染管线的架构
  - ➢ **Performance is achieved by using GPU rather than CPU,GPU does all rendering,**
  - ➢ **Programmer Control GPU through shaders**
    - Application's job (run at CPU) is to send data to GPU
    - Vertext Shader run at per vertex, fragment shader run at per fragment concurrently



# Recap(cont.)

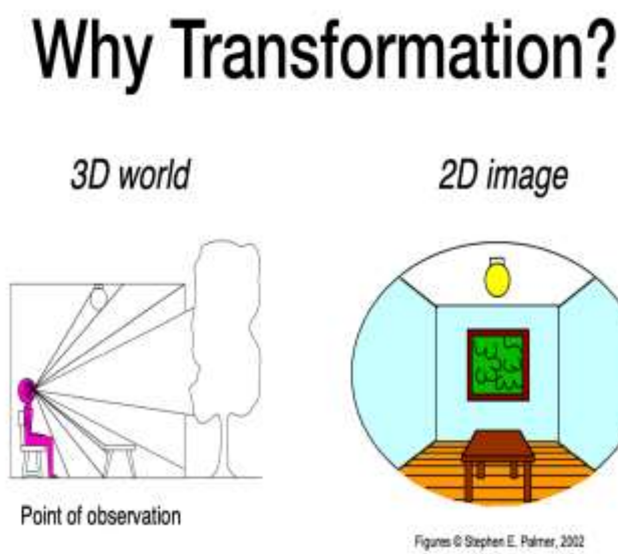
- Basic Geometry Primitive
  - **Point**
  - **Line segment**
  - **polygon(mesh, polygon)**

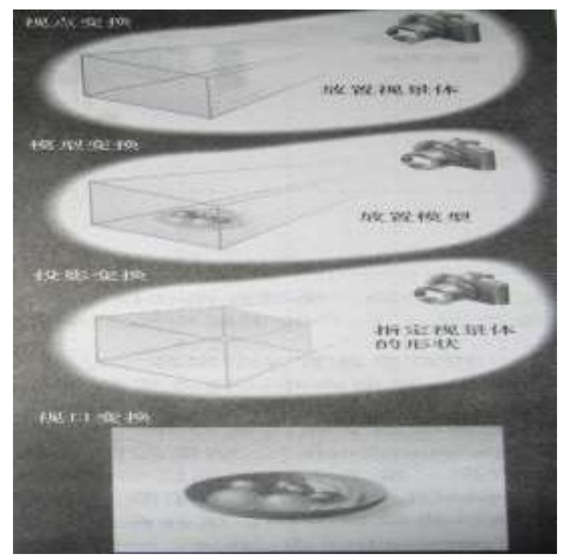



  - 思考:如何在物理空间中表示顶点和向量?

# Question

- 对象(顶点/向量)在物理空间中的表示Representation ？
  - 齐次坐标
- 对象(顶点/向量)的空间位置表示的变换Transformation?
  - 仿射变换



# Outline

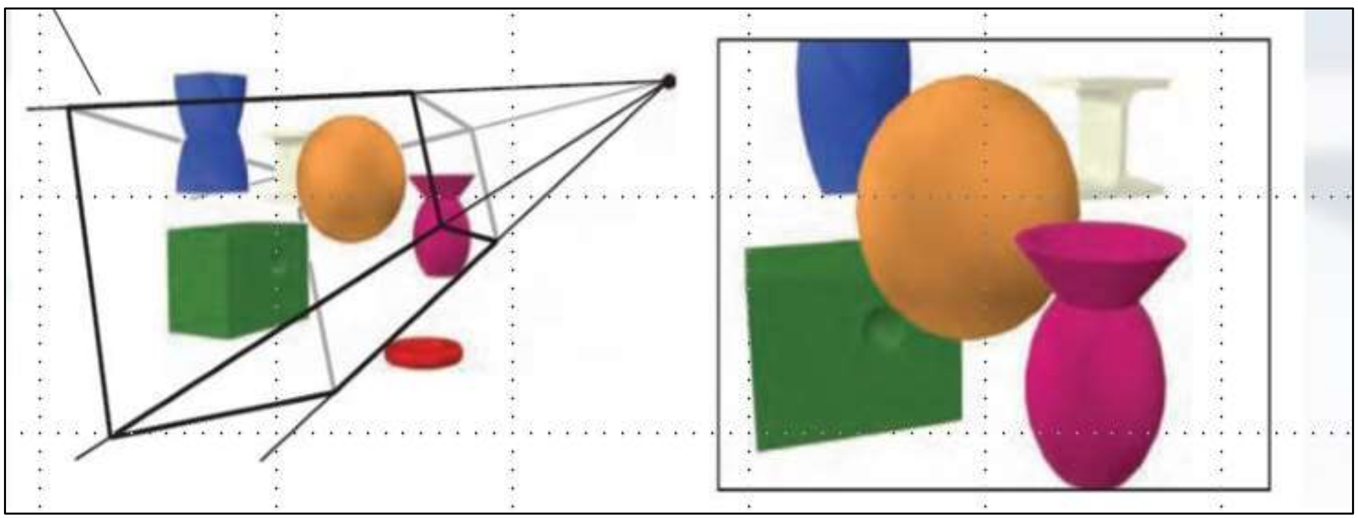
- **Representation（表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - ……

# Representation

- Need a frame of reference to relate points and objects to our physical world.（需要一个“参考框架”来将点和物体与我们的物理世界联系起来）

  *For example, where is a point?*

  - *Can’t answer without **a reference system(参照系统)***



The University of New Mexico

# Outline

- **Representation（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Transformations（变换的应用）**
  - Non-standard transformation非标变换
  - Cumulative transformation累积变换
  - ……

The University of New Mexico

# Vector space, Coordinate System, Change of Coordinate

## ➢Coordinate Systems（坐标系统）

- Consider a basis $v_1, v_2, \ldots, v_n$, ,
- A vector is written $v=\alpha_1 v_1+ \alpha_2 v_2 +\ldots+\alpha_n v_n$
- The list of scalars $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$is the <u>representation of v</u> with respect to the given basis,
- write the representation as a row or column array of scalars

$$\boldsymbol{a} =[\alpha_1 \ \ \alpha_2 \ \ \ldots \ \alpha_n]^T = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ \alpha_n \end{bmatrix}$$

*Example:* $v=2v_1+3v_2-4v_3$

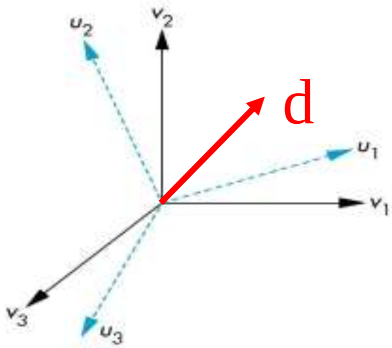
$\boldsymbol{a}=[2 \ \ 3 \ -4]^T$

*Note that this representation is with respect to a particular basis*

# Vector space, Coordinate System, Change of Coordinate(cont.)

- **Change of Coordinate Systems(坐标系转换）**
  - Consider two representations of the same vector d with respect to two different bases.

The representations are：




$$\mathbf{a}=[\alpha_1\ \alpha_2\ \alpha_3]$$

$$\mathbf{b}=[\beta_1\ \beta_2\ \beta_3]$$

where

$$d=\alpha_1 v_1+\alpha_2 v_2+\alpha_3 v_3=[\alpha_1\ \alpha_2\ \alpha_3]\,[v_1\ v_2\ v_3]^T$$

$$d=\beta_1 u_1+\beta_2 u_2+\beta_3 u_3=[\beta_1\ \beta_2\ \beta_3]\,[u_1\ u_2\ u_3]^T$$

The University of New Mexico

# Vector space, Coordinate System, Change of Coordinate(cont.)

- **Change of Coordinate Systems(cont.)**

➢ **Representing second basis in terms of first:**

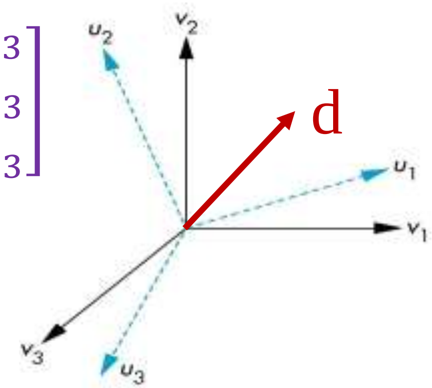
*Each of the basis vectors, u1,u2, u3, are vectors that can be represented in terms of the first basis v1,v2,v3*

➢ *已知:* $d=\alpha_1 v_1+\alpha_2 v_2+\alpha_3 v_3=[\alpha_1\ \alpha_2\ \alpha_3][v_1\ v_2\ v_3]^T$ $\quad \mathbf{a}=[\alpha_1\ \alpha_2\ \alpha_3],\ \mathbf{v}=[v_1\ v_2\ v_3]^T$

$d=\beta_1 u_1+\beta_2 u_2+\beta_3 u_3=[\beta_1\ \beta_2\ \beta_3][u_1\ u_2\ u_3]^T$ $\quad \mathbf{b}=[\beta_1\ \beta_2\ \beta_3],\ u=[u_1\ u_2\ u_3]^T$

Let:
$$u_1=\gamma_{11}v_1+\gamma_{12}v_2+\gamma_{13}v_3$$
$$u_2=\gamma_{21}v_1+\gamma_{22}v_2+\gamma_{23}v_3$$
$$u_3=\gamma_{31}v_1+\gamma_{32}v_2+\gamma_{33}v_3$$

$$\mathbf{M}=\begin{bmatrix}\gamma_{11} & \gamma_{12} & \gamma_{13}\\ \gamma_{21} & \gamma_{22} & \gamma_{23}\\ \gamma_{31} & \gamma_{32} & \gamma_{33}\end{bmatrix}$$




**Proof:** for $\mathbf{d}=\mathbf{bu}$, **and** $\mathbf{u}=\mathbf{Mv}$

so: $\mathbf{d}=\mathbf{b}(\mathbf{Mv})=\mathbf{bMv}$, and: $\mathbf{d}=\mathbf{av}$

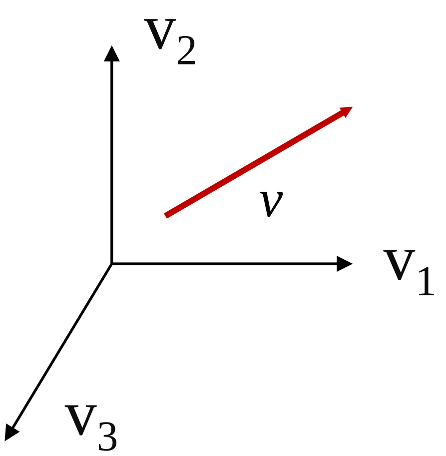
=> $\mathbf{a}=\mathbf{b}\,\mathbf{M}$ *// here a , b are <u>row</u> vector representation*

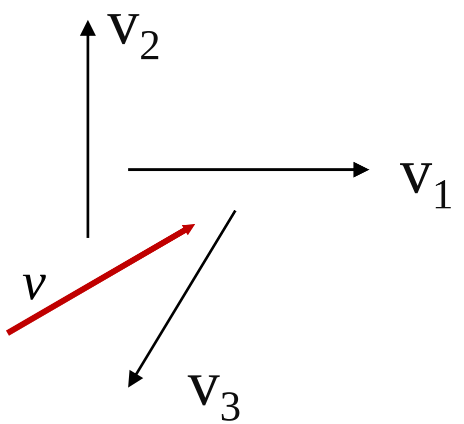
*or:* $\mathbf{b}=\mathbf{a}*\mathbf{M}^{-1}$

# Vector space, Coordinate System, Change of Coordinate(cont.)

- Example: $v=2v_1+3v_2+4v_3$ **v向量表示为 a**=[2 3 4]$^T$
  - The two case ,Which one is correct ?
    - **Both are**, *because vectors have no fixed location*

(两种情况下的v具有相同表示，因为向量没有固定位置！)



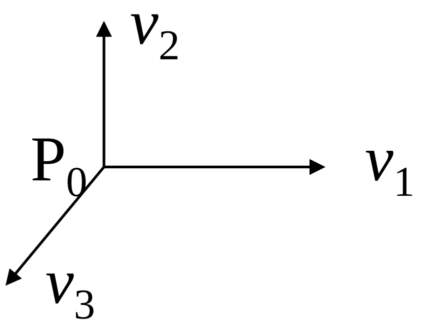



# Outline

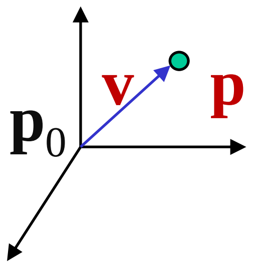
- Representation（对象在物理空间的表示）
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - **Homogeneous Coordinate**
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - Eg3:Instance transformation实例变换

# Affine space, Frames System, Change of Frame

- Frames(标架)
  - If we work in an affine space(仿射空间), can add a single point: the origin(原点), to the basis vectors(基向量), to form a frame(标架)
  - Frame(标架) determined by ($P_0$, $v_1$, $v_2$, $v_3$),




  - *Within this frame,*
    - Every **vector(向量)** *can be written as* $\boldsymbol{v} = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
    - Every **point (点)** *can be written as* $P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n$



The University of New Mexico

# Affine space, Frames System, Change of Frame(cont.)

## ➢Use Frame Will Confusing Points and Vectors

Consider the **point** and the **vector,**

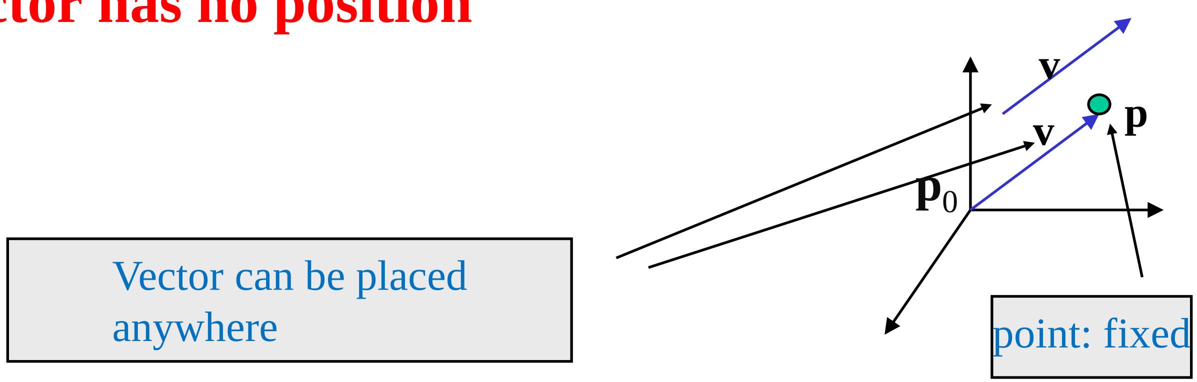
They appear to have the similar representations

$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n$ $\qquad$ $\mathbf{p}=[\beta_1\ \beta_2\ \beta_3]$

$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$ $\qquad$ $\mathbf{v}=[\alpha_1\ \alpha_2\ \alpha_3]$

## what confuses the point with the vector?

### ➢A vector has no position



# Affine space, Frames System, Change of Frame(cont.)

The University of New Mexico

- **Redefine Frame Representation**
  - **引入齐次坐标表示:Homogeneous Coordinate Representation**

If we define $0 \bullet P = \mathbf{0}$ and $1 \bullet P = P$ ,Then

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + 0 \bullet P_0 = [\alpha_1\ \alpha_2\ \alpha_3\ 0\ ]\,[v_1\ v_2\ v_3\ P_0]^T$$

$$P = \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 + 1 \bullet P_0 = [\beta_1\ \beta_2\ \beta_3\ 1\ ]\,[v_1\ v_2\ v_3\ P_0]^T$$

- **the homogeneous coordinate representation**

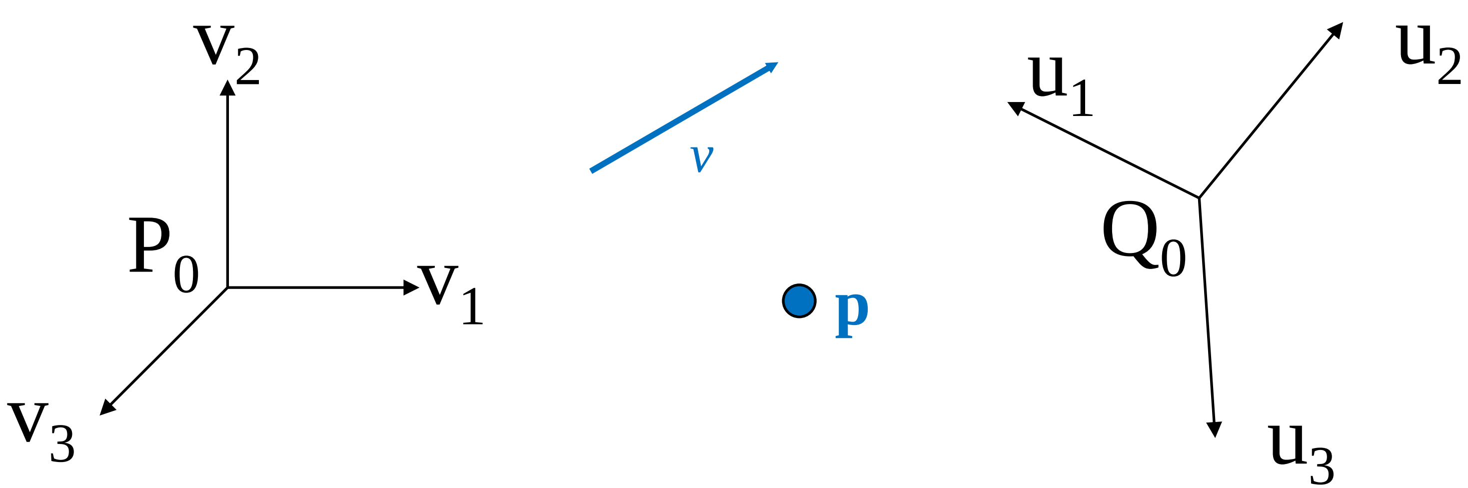
**Vectors:** $\mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3\ \mathbf{0}\ ]^{\mathbf{T}}$

**Points:** $\mathbf{p} = [\beta_1\ \beta_2\ \beta_3\ \mathbf{1}\ ]^{\mathbf{T}}$

The University of New Mexico

# Affine space, Frames System, Change of Frame(cont.)

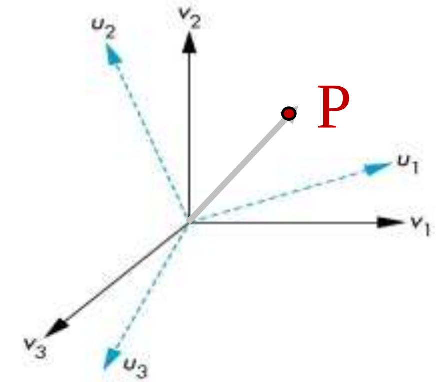
- **Change of Frames（标架的转换）**
  - Consider two frames: $(P_0, v_1, v_2, v_3)$ , $(Q_0, u_1, u_2, u_3)$
    - Any point or vector can be represented in either frame
    - can represent $Q_0, u_1, u_2, u_3$ in terms of $P_0, v_1, v_2, v_3$



# Affine space, Frames System, Change of Frame(cont.)

## **Change of Frames** in homogeneous coordinates（**cont.)**

### - Representing One Frame in Terms of the Other




- Extending what we did with change of bases

Let: $u_1 = \gamma_{11}v_1+\gamma_{12}v_2+\gamma_{13}v_3$

$u_2 = \gamma_{21}v_1+\gamma_{22}v_2+\gamma_{23}v_3$

$u_3 = \gamma_{31}v_1+\gamma_{32}v_2+\gamma_{33}v_3$

$Q_0 = \gamma_{41}v_1+\gamma_{42}v_2+\gamma_{43}v_3+\gamma_{44}P_0$

$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

- Defining a 4 x 4 matrix, The matrix **M** is 4 x 4 and specifies an affine transformation(仿射变换) in homogeneous coordinates(齐次坐标表示)
- Within the two frames, any **point** or **vector** has a representation of the same form, where $\alpha_4=\beta_4=1$ for points and $\alpha_4=\beta_4=0$ for vectors .

  $\boldsymbol{a}=[\alpha_1\ \alpha_2\ \ \alpha_3\ \alpha_4]$ in the first frame $v=(V_1,V_2,\ V_3,\ P_0)^T$

  $b=[\beta_1\ \beta_2\ \ \beta_3\ \beta_4]$ in the second frame $u=(u_1,u_2,u_3,Q_0)^T$

**Proof**: for **P** = **bu** , **and** **u** =**Mv** **//d is point or vector**

so: **P** = **b**(**Mv**) = **bMv**, and: **P** = **av**

=> $\boldsymbol{a=b\,M}$ ***// here a , b are row vector representation***

# Outline

- **Representation（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - **Homogeneous Coordinate**
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - ……

# Homogeneous Coordinate

The University of New Mexico

## ➢What is Homogeneous coordinates(齐次坐标)

- 齐次坐标是一种数学工具，用于投影几何中，通过将一个原本是n维的向量表示为一个n+1维向量，以简化几何变换和操作.
- 齐次坐标的概念由奥古斯特·费迪南德·莫比乌斯在1827年引入，并在多个领域中得到广泛应用，包括电脑图形学和3D电脑视觉。

- **三维空间的齐次坐标(Xh，Yh，Zh，h)**
  - ➢h=0 表示"向量vector"
  - ➢h!=0 表示 "点 point"
- **三维笛卡尔坐标**（X, Y, Z）**和齐次坐标($X_h$，$Y_h$，$Z_h$，h）的转换**
  - ➢$X= X_h / h$ , $Y=X_h / h$ ， $Z=Z_h / h$

  Example1: 齐次坐标为 (20,15,10,5) ，(16,12, 8, 4)，(4,3, 2,1)的点，它们的笛卡尔坐标都是（ 4，3 , 2 ）
- **齐次坐标表示不是唯一的！**
  - **当h=1时，该点的表示称为"规格化齐次坐标"**
  - **如无特殊说明，一般图形顶点都采用规格化齐次坐标来表示！**

# Homogeneous Coordinate(cont.)

- **Why Homogeneous coordinates ?**
  - **Homogeneous coordinates are key to all computer graphics systems:**
    - Hardware pipeline works with 4 dimensional representations . All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4 x 4 matrices
    - **简化几何变换：通过增加一个维度，齐次坐标能够简化三维空间中的点、线、面的表达方式和旋转、平移等操作。**
      - **例如，二维平面上的点可以用齐次坐标(x,y,1)(*x*,*y*,1)表示，这样可以直接利用矩阵运算进行变换，而无需分别处理平移和旋转。**

# Outline

- **Representation*（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate*
- **Transformations*（变换）**
  - Five Standard Transformation（五种标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - ……

The University of New Mexico

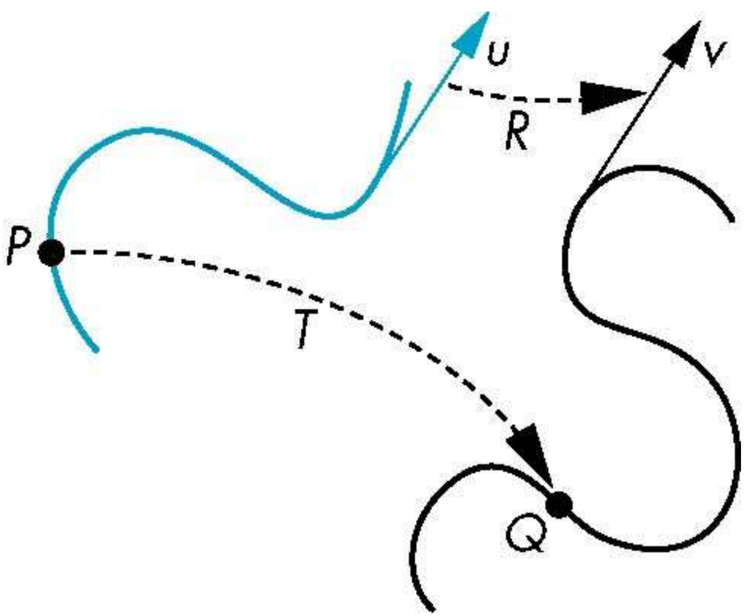
# Transformations*（变换）

## ➢What is Transformation?

- ➢*maps points to other points* and/or *maps vectors to other vectors*

*例如:把点P的表示，通过“T变换”，变为点Q的表示,*

*把向量u的表示，通过“R变换”，变为 向量v的表示,*

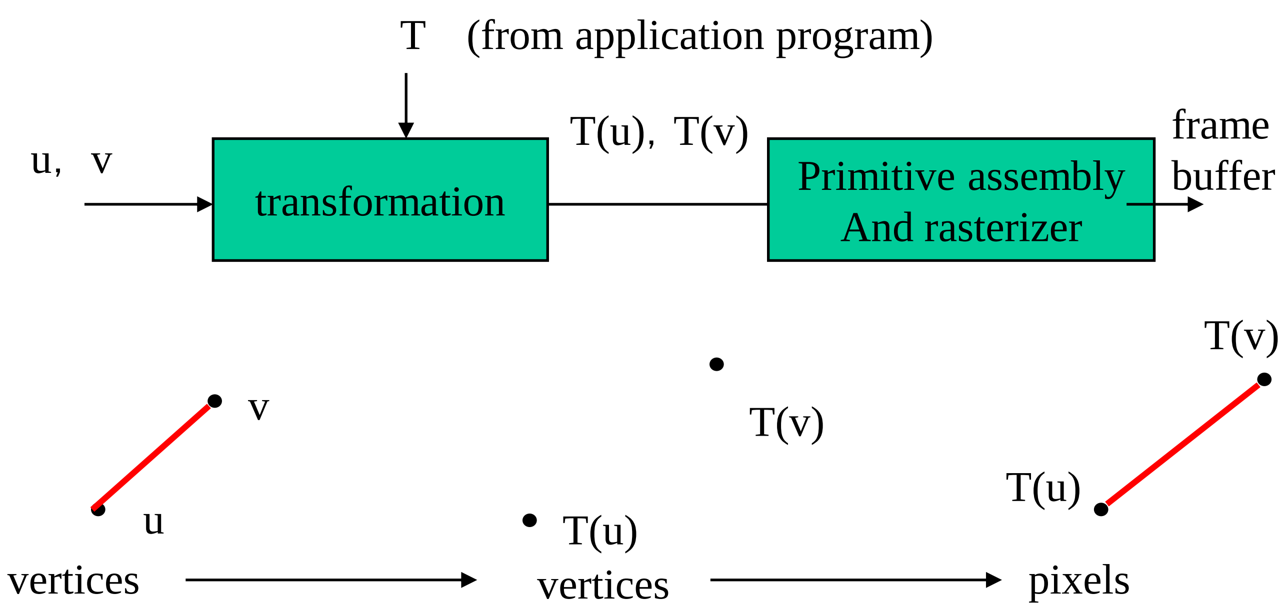



**v=R(u)**

**Q=T(P)**

# Transformations in Graphics

- Why Transformation?
  - 只需要对顶点作变换，就可以实现对点、线段以及多边形面图元的变换.
    - 例如：下面对线段的变换：



# Outline

- **Representation*（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate*
- **Transformations*（变换）**
  - Five Standard Transformation（五种标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - ……

The University of New Mexico

# Affine Transformation

## ➢Affine Transformations（仿射变换）

In Euclidean geometry, an **affine transformation is a geometric transformation**

**That preserves lines and parallelism,but not ecessarily distances and angles.**

- Every affine transformation preserves lines（仿射变换具有保线性:变换前是直线的，变换后依然是直线；变换前是平行线的，变换后依然是平行线）
- Every linear transformation is equivalent to a change in frames（每个线性变换等价于一个框架改变）
- However, an affine transformation has only 12 degrees of freedom because 4 of the elements in the matrix are fixed , and are a subset of all possible 4 x 4 linear transformations（每个仿射变换，对应于一个4*4矩阵，但仅有12个自由度）

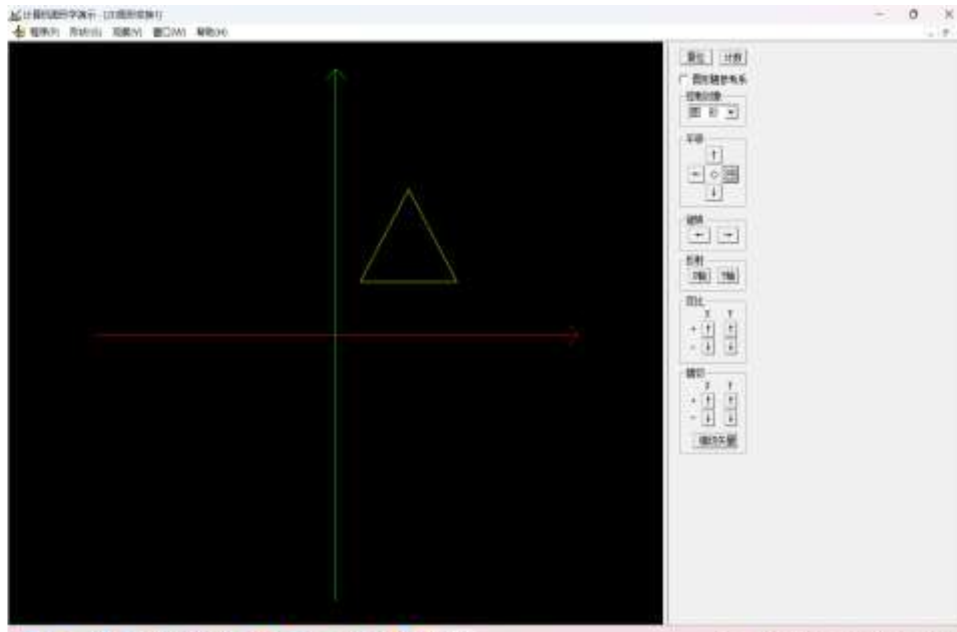
◆以下是采用“齐次坐标”并且用“行向量”表示顶点时的变换矩阵(框架改变矩阵)，

$a=b\,M$ **//变换矩阵M将齐次坐标 b转换为齐次坐标 a，这里a,b是1*4的行向量**

$$\mathbf{M}=\begin{bmatrix}\gamma_{11} & \gamma_{12} & \gamma_{13} & 0\\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0\\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0\\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1\end{bmatrix}$$

# Standard Transformations*

- Introduce standard transformations（标准变换） and
- <u>Derive homogeneous coordinate transformation matrices</u>
  - Rotation
  - Translation
  - Scaling
  - Reflection
  - Shear



<u>演示</u>

（标准变换：以原点和坐标轴为参考点或参考轴的变换）

# Standard Transformations*
# 1）Translation平移

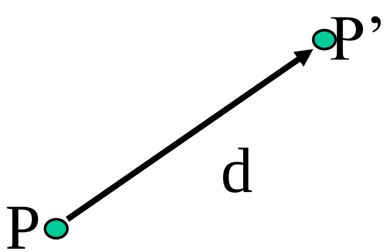
- Move (translate, displace) a point to a new location
- Displacement(移动）determined by a vector d
  But when we move many points，there is usually only one way
- Translation: every point displaced  by same vector

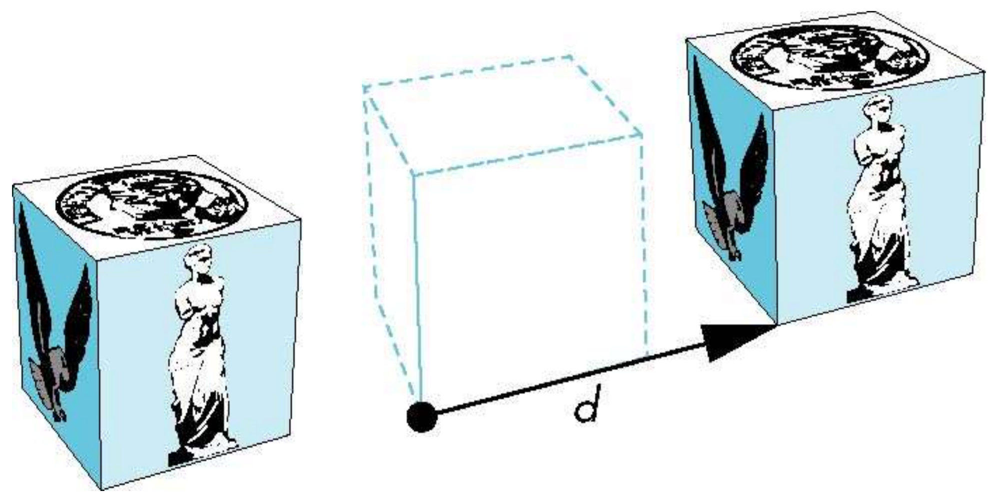
$$P' = P + d$$

$$x' = x + d_x$$

$$y' = y + d_y$$

$$z' = z + d_z$$

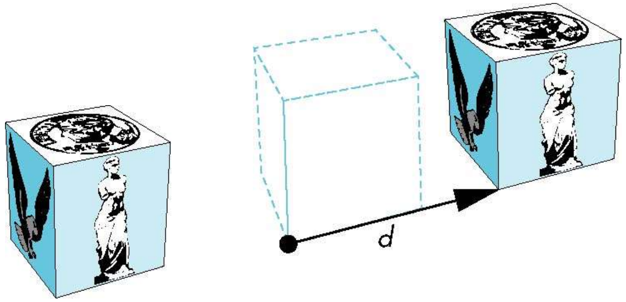


# Standard Transformations*
# 1）Translation平移（cont.)

- Using the homogeneous coordinate representation in some frame,
  Hence **P'** = P + **d** or

$$\begin{aligned} x' &= 1*x + 0*y + 0*z + d_x *1 \\ y' &= 0*x + 1*y + 0*z + d_y *1 \\ z' &= 0*x + 0*y + 1*z + d_z *1 \\ 1 &= 0*x + 0*y + 0*z + 1 *1 \end{aligned}$$



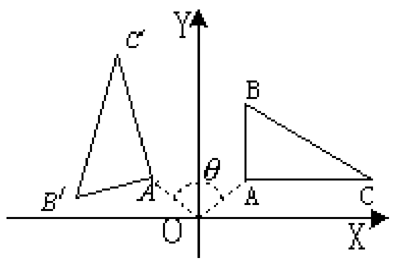

- express translation using a 4 x 4 matrix **T** in homogeneous coordinates

$$\mathbf{P'} = \mathbf{T}\,\mathbf{P} \quad \text{where} \quad \mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 注意：P',P,d 都是列向量表示，T矩阵对P进行“左乘”
- This form in homogeneous coordinates is better for implementation!

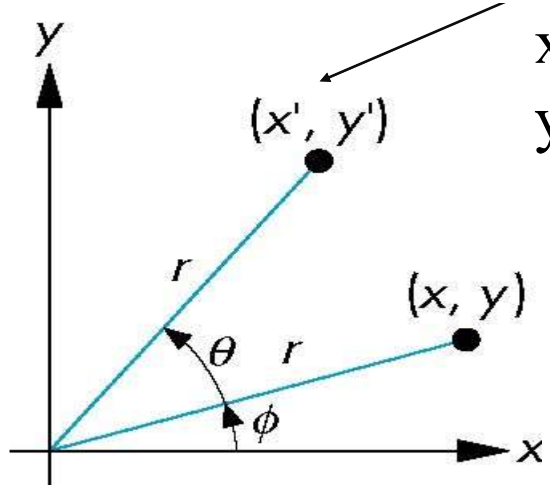
# Standard Transformations*
# 2）Rotation旋转

- 二维平面上，绕原点的旋转
  - Consider rotation about the origin by θ degrees，radius stays the same, angle increases by $\theta$
  - *注意：逆时针方向旋转时，角度为正；顺时针方向旋转时，角度为负。*




相对原点旋转 θ角

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




$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

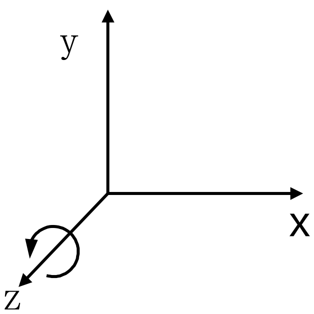
$$x = r\cos\phi$$
$$y = r\sin\phi$$

# Standard Transformations*

## 2）Rotation（cont.)

- 三维旋转，绕Z轴旋转
  - Rotation about z axis in three dimensions leaves all points with the same z, equivalent to rotation in two dimensions in planes of constant z

$$
\begin{aligned}
x' &= x*\cos\theta - y*\sin\theta + z*0 + 1*0 \\
y' &= x*\sin\theta + y*\cos\theta + z*0 + 1*0 \\
z' &= x*0 + y*0 + z*1 + 1*0 \\
1 &= x*0 + y*0 + z*0 + 1*1
\end{aligned}
$$




- in homogeneous coordinates

**$P' = R_Z(\theta)P$,**

$$
R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The University of New Mexico

# Standard Transformations*

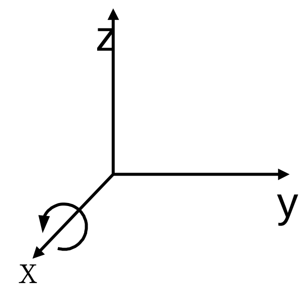
# 2）Rotation(cont.)

## ➢Rotation about x and y axes

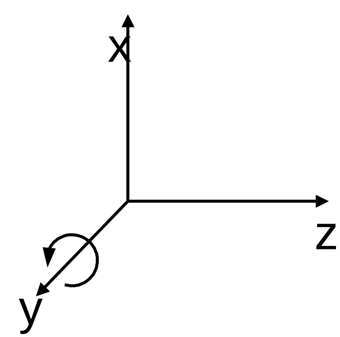
Same argument as for rotation about *z* axis

- For rotation about $x$ axis, $x$ is unchanged
- For rotation about $y$ axis, $y$ is unchanged

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



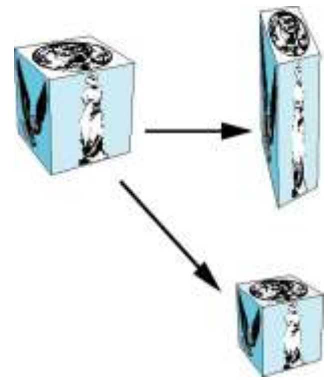

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

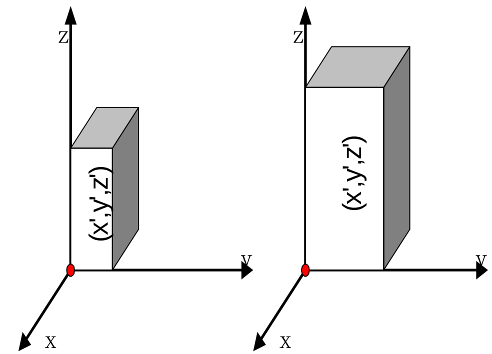


# Standard Transformations*
## 3) Scaling缩放/变比

- Expand or contract along each coordinate axis and fixed point of origin

$$
\begin{aligned}
x' &= s_x * x \\
y' &= s_y * y \\
z' &= s_z * z \\
1 &= 1*1
\end{aligned}
$$






- in homogeneous coordinates
  - **P'=S P**

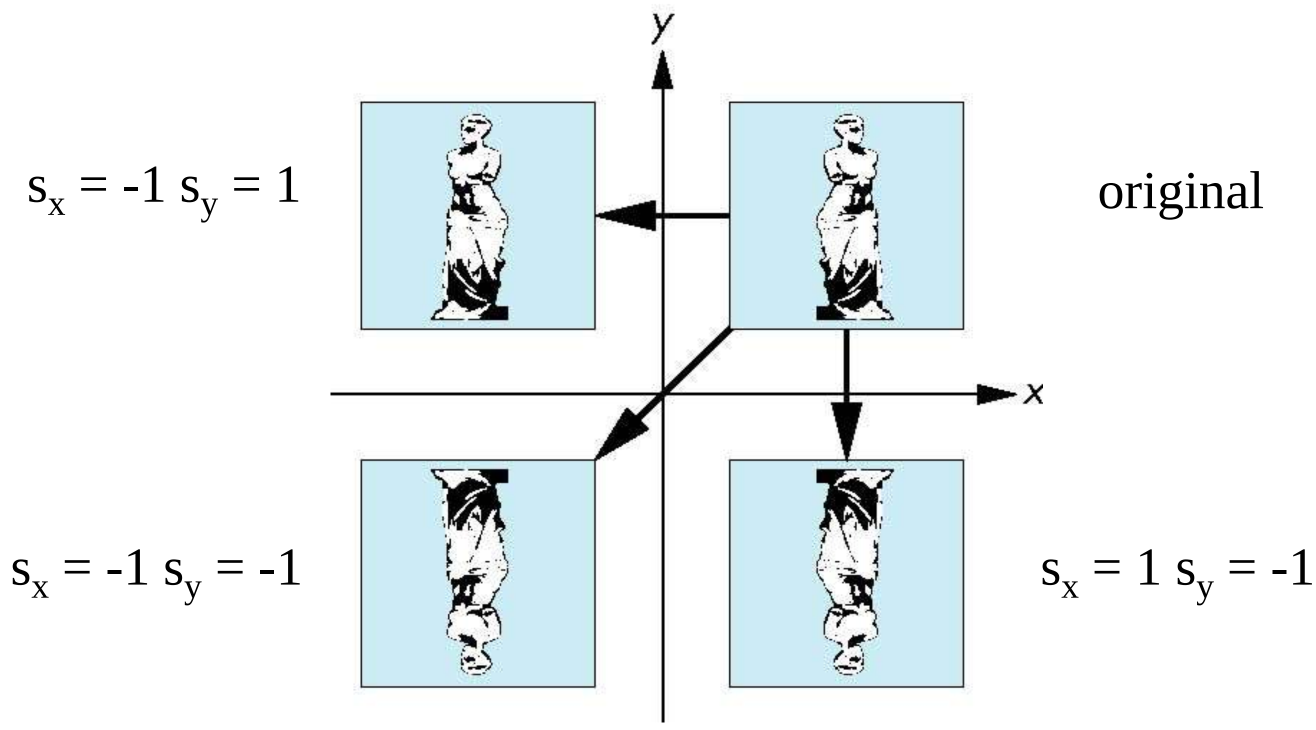
$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 注意：缩放时，是参考“原点”，沿着X，Y，Z轴进行缩放的！

# Standard Transformations*
# 4）Reflection反射/对称

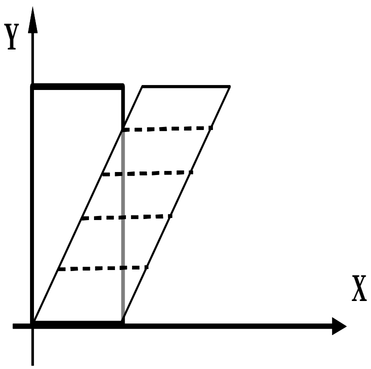
- 反射：参照坐标轴或原点进行
  - corresponds to **negative scale factors（负1）**

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



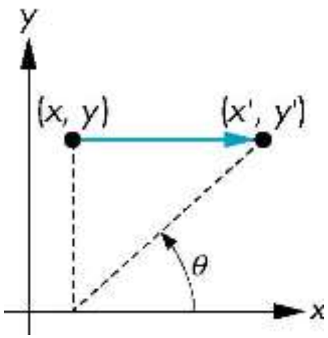
# Standard Transformations*
# 5)Shear-错切

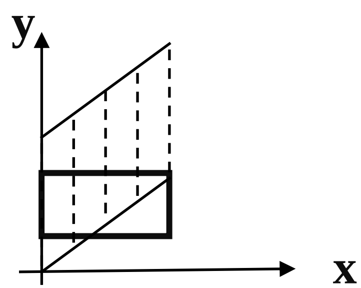
- 二维空间中，用齐次坐标表示的错切
  - $P' = Shear(sh_x, sh_y)P$ $\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$




x依y轴的错切：

- $x' = x + sh_x \cdot y$
- $y' = y$
- $1 = 1$




*Y*依X轴的错切

- $x_1 = x$
- $y_1 = sh_y \cdot x + y$
- $1 = 1$

# Standard Transformations*
# 5)Shear-错切(cont.)

- 三维空间中，用齐次坐标表示的错切P’ =Shear($sh_x$ , $sh_y$)P
  - 依赖轴:坐标保持不变的轴
  - 方向轴:坐标“参照依赖轴”呈线性变化的轴

X为依赖轴：

$$x' = x$$
$$y' = dx + y$$
$$z' = gx + z$$

Y为依赖轴：

$$x' = x + by$$
$$y' = y$$
$$z' = hy + z$$

z为依赖轴：

$$x' = x + cz$$
$$y' = y + fz$$
$$z' = z$$

$$T_{SHx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ d & 1 & 0 & 0 \\ g & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{SHy} = \begin{bmatrix} 1 & b & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & h & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{SHz} = \begin{bmatrix} 1 & 0 & c & 0 \\ 0 & 1 & f & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Standard Transformations*

The University of New Mexico

➢总结用齐次坐标表示变换形式如下：

■用齐次坐标且"行向量"表示： **P'= P *M** **//右乘变换矩阵**

$$\mathbf{p'}=[x'\ y'\ z'\ 1] \quad \mathbf{p}=[\ x\ y\ z\ 1] \qquad M_{4*4}=\begin{bmatrix} sx & d & g & 0 \\ b & sy & h & 0 \\ c & f & sz & 0 \\ \mathrm{tx} & ty & tz & 1 \end{bmatrix}$$

■用齐次坐标且"列向量"表示： **P'=M* P** **//左乘变换矩阵**

$$\mathbf{p'}=\begin{bmatrix} \mathrm{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \mathbf{1} \end{bmatrix} \qquad M_{4*4}=\begin{bmatrix} sx & b & c & tx \\ d & sy & f & ty \\ g & h & sz & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{p}=\begin{bmatrix} \mathrm{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \mathbf{1} \end{bmatrix}$$

The University of New Mexico

# Outline

- **Representation（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - ……

# Concatenation Transformation

- Concatenation Transformation级联变换： 是多个标准变换的串联形式, 变换矩阵的累乘！
  - 采用齐次坐标表示，可将平移变换表示为矩阵乘，统一了变换的表示
  - 任意变换可表示为多个标准变换累乘(级联换矩阵M),作用于初始数据P0

✓好处：无需计算和保存中间过程产生的顶点数据Pi，可以减少数据量极大的顶点数据的存储和传输，从而大大提高顶点变换的效率！

$$\begin{cases} P_2 = M_1^1 \cdot P_1 + M_2^1 \\ P_3 = M_1^2 \cdot P_2 + M_2^2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ P_n = M_1^n \cdot P_{n-1} + M_2^n \end{cases} \xrightarrow{\text{采用齐次坐标}} \begin{cases} P_2 = M_1 P_1 \\ P_3 = M_2 P_2 \\ \cdots\cdots\cdots\cdots \\ P_n = M_n P_{n-1} \end{cases} \xrightarrow{\text{级联}} \boxed{\begin{aligned} P_n &= M_{n-1} * \ldots * M_2 * M_1 * P_0 \\ &= M * P_0 \end{aligned}}$$

# Concatenation Transformation(cont.)

- 级联变换中矩阵乘的顺序Concatenation Order
  - many references use <u>column matrices</u> to represent points，note that “matrix on the right is the first applied” （采用列矩阵表示点，注意最右边矩阵是第1个应用的矩阵）

- Mathematically, the following are equivalent

$\mathbf{p}' = \mathbf{ABCp} = \mathbf{A}(\mathbf{B}(\mathbf{Cp}))$ //当列向量表示时，累乘顺序是左乘

$\mathbf{p}'^{T} = \mathbf{p}^{T}\mathbf{C}^{T}\mathbf{B}^{T}\mathbf{A}^{T}$ //行向量表示时：累乘顺序是右乘（万琳课件用之）

# Concatenation Transformation(cont.)

The University of New Mexico

- The difficult part is *"how to **form** M"*

*a desired Concatenation transformation is from the specifications in the application"*

$$P_n = M * P_0$$

$$M = M_n * \ldots * M_2 * M_1$$

# Outline

- **Representation（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - ......

# Non-standard transformation

The University of New Mexico

## •补充:Inverses Transformation 逆变换

Although we could compute inverse matrices by general formulas, we can use simple geometric observations

- Translation: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathbf{T}(-d_x, -d_y, -d_z)$
- Rotation: $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$, $\mathbf{R}^{-1}(\theta) = \mathbf{R}^{T}(\theta)$
  - Holds for any rotation matrix
  - Note that since $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta)=-\sin(\theta)$
- Scaling: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$

# Non-standard transformation（cont.)

The University of New Mexico

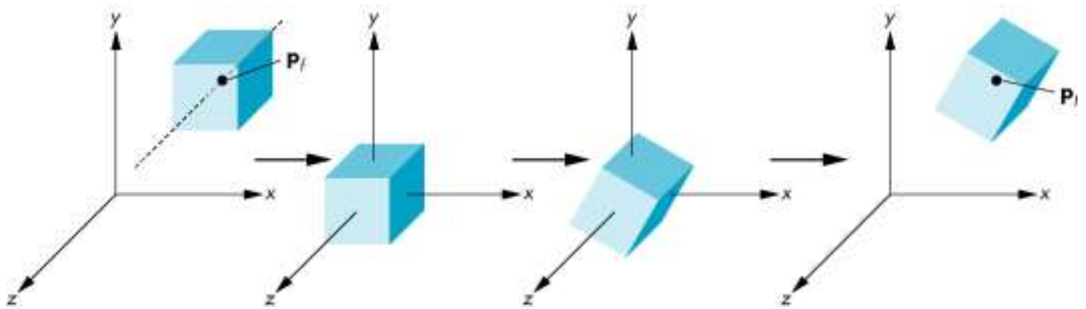
## ex1: 绕任意点（非原点），且Z轴方向的旋转

**Rotation About a Fixed Point other than the Origin**

➢**该变换表示为多个标准变换的级联变换矩阵：**

$\mathbf{M} = \mathbf{T}(p_f)\ \mathbf{Rz}(\theta)\ \mathbf{T}(-p_f)$ //列向量表示：左乘顺序

1. Move fixed point to origin $\mathbf{T}(-p_f)$
2. Rotate $\theta$ $\mathbf{Rz}(\theta)$
3. Move fixed point back $\mathbf{T}(p_f)$



# Non-standard transformation (cont.)

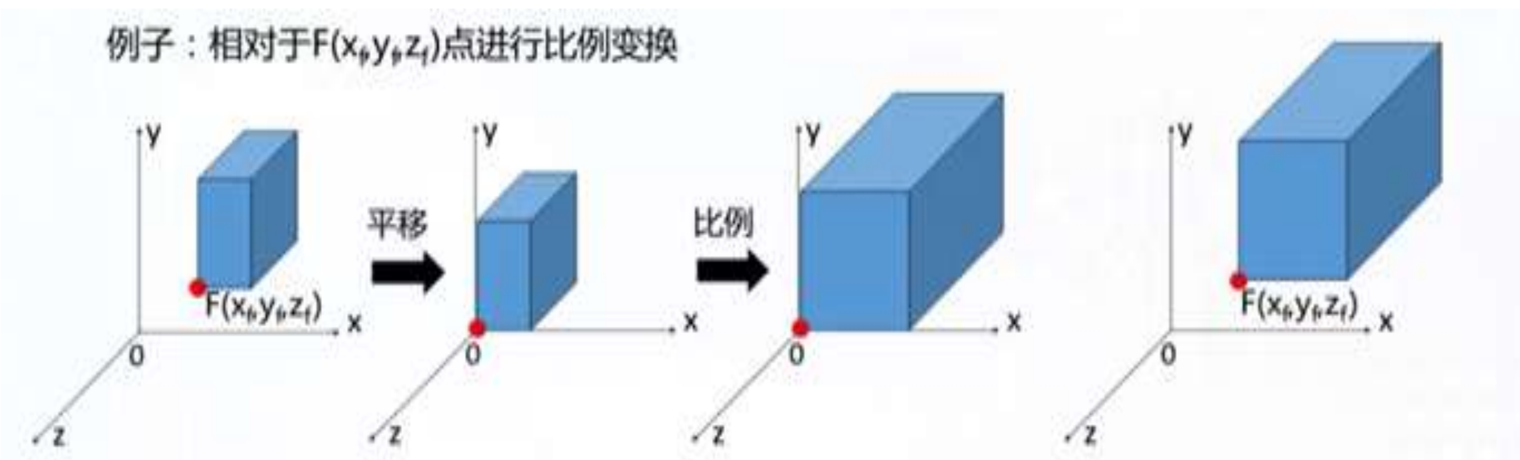
The University of New Mexico

**Ex2: 依任意点的缩放(Scale About a Fixed Point other than the Origin)**

➢**该变换表示为多个标准变换矩阵累乘的级联矩阵:**

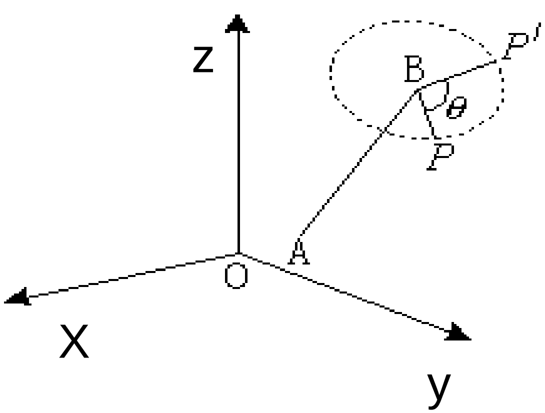
**M** = **T**($X_f$, $Y_f$, $Z_f$) **S**(Sx,Sy,Sz) **T**($-X_f$, $-Y_f$, $-Z_f$) //列向量表示:左乘

1. Move fixed point to origin **T**($-X_f$, $-Y_f$, $-Z_f$)
2. Scale **S**(Sx,Sy,Sz)
3. Move fixed point back **T**($X_f$, $Y_f$, $Z_f$)



# Non-standard transformation（cont.)

- **Ex3:绕任意轴的旋转Rotation Transformation Around Any Axis**
- 如空间一点P($x_p$, $y_p$, $z_p$)绕AB轴旋转角到P$'$($x_p'$, $y_p'$, $z_p'$)，求Rab



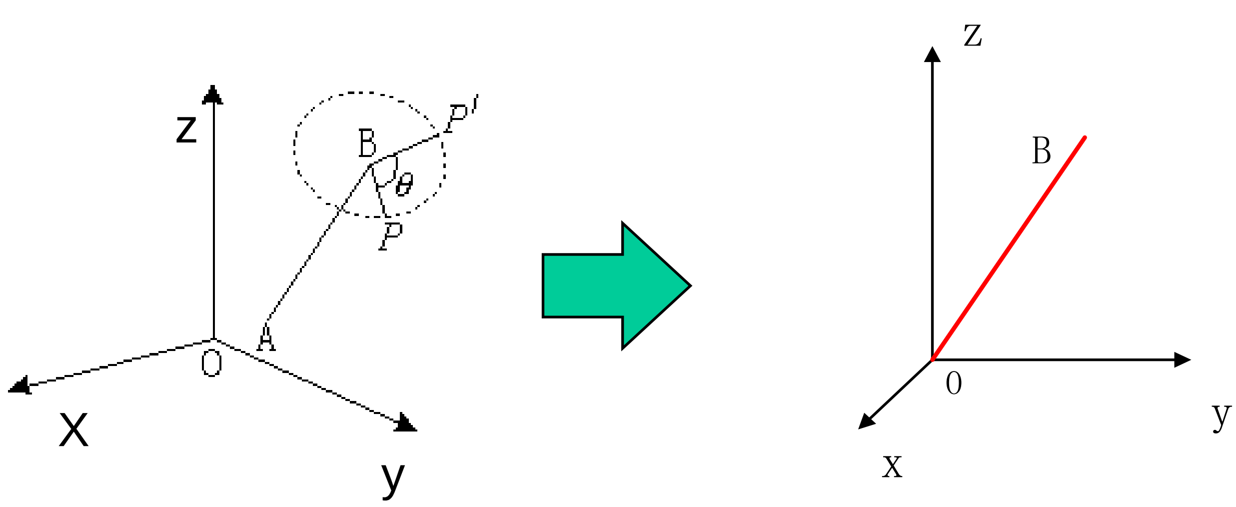

$$\begin{bmatrix} x_p' \\ y_p' \\ z_p' \\ 1 \end{bmatrix} = R_{ab}(\theta) \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

# Non-standard transformation（cont.)

- **Ex: Rotation Transformation Around Any Axis（cont.)**
  - **Step1: 平移A点到原点**

    **将AB平移,使A点与坐标原点重合，得到OB**

    OB的方向数设为(a,b,c), $a^2+b^2+c^2=|OB|^2$




(1)

# Non-standard transformation(cont.)

The University of New Mexico

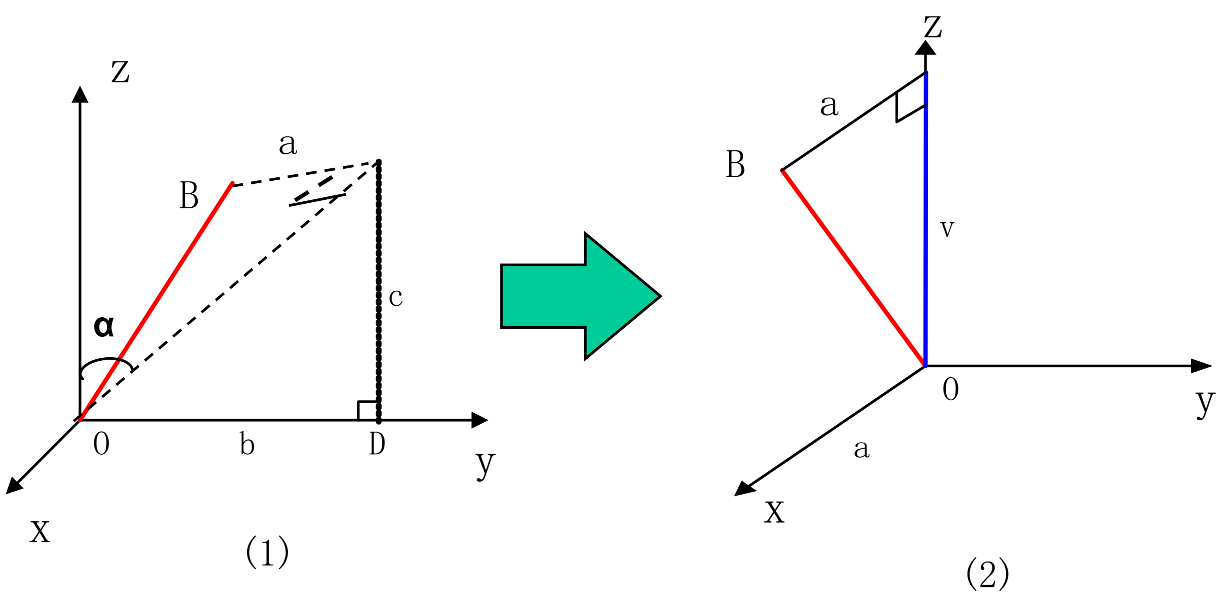
- **Ex: Rotation Transformation Around Any Axis(cont.)**

➢Step2: OB经绕X轴逆时针旋转到xoz面上

**将OB绕x轴逆时针旋转α角，则OB旋转到XOZ平面上**

**α角：*将OB投影到YOZ面后的投影线和Z轴的夹角，也是OB旋转的角度。***

***v值: $v^2=b^2+c^2$***



# Non-standard transformation(cont.)

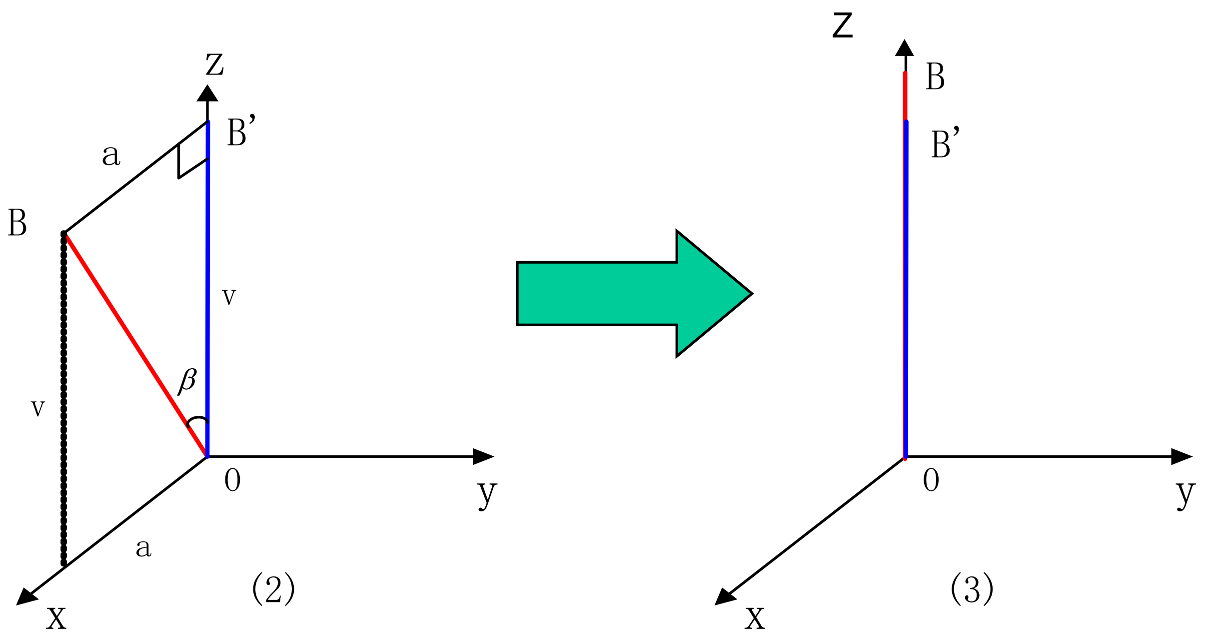
The University of New Mexico

- **Ex: Rotation Transformation Around Any Axis(cont.)**

➢Step3：**将OB绕Y轴顺时针旋转到Z轴上**

**将OB绕Y轴顺时针旋转β角，则OB旋转到Z轴上**

**β角：*将OB绕Y轴顺时针旋转的角度*** ***here:*** $v^2=b^2+c^2$

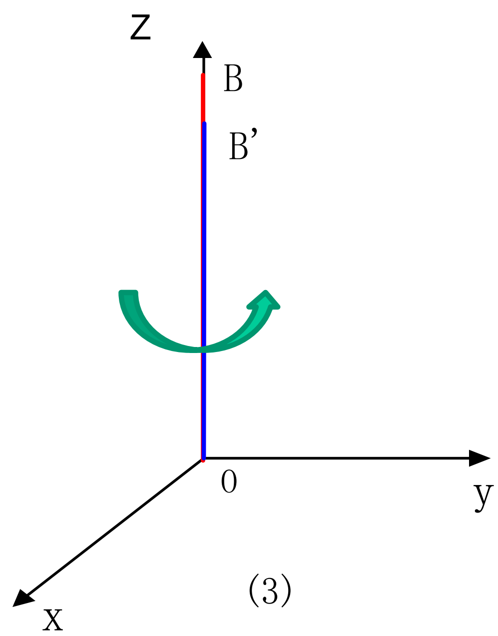


# Non-standard transformation(cont.)

The University of New Mexico

- **Ex: Rotation Transformation Around Any Axis(cont.)**

➢Step4: 物体绕OB轴（Z轴）作标准旋转



# Non-standard transformation(cont.)

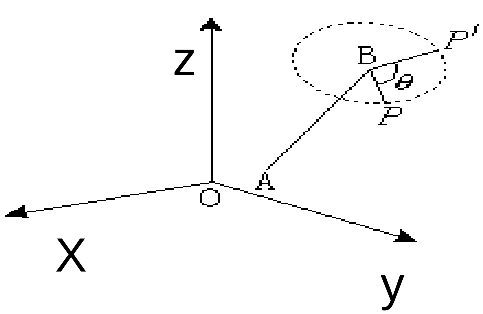
- **Ex: Rotation Transformation Around Any Axis(cont.)**

➢ Step5~Step7：逆变换

  ➢ **作前面step3, step2,step1对应的逆变换**

  ➢ 按顺序从右边到左排列这七个标准变换矩阵相乘，可求出Rab

$$T_{Rab} = T_{tA}^{-1} T_{Rx}^{-1} T_{Ry}^{-1} T_{Rz} T_{Ry} T_{Rx} T_{tA}$$




$$\begin{bmatrix} x_p' \\ y_p' \\ z_p' \\ 1 \end{bmatrix} = R_{ab}(\theta) \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

# Non-standard transformation（cont.)

➢Practice

对2D空间中的图形，请写出以下对称变换所对应的串联矩阵

- 点 (2,3)对称
- 轴 y=2对称
- 轴 y=2x对称
- 轴y=ax+b对称

- **using Homogeneous Coordinate and column vector**

  **采用齐次坐标表示和列向量（*注意：串联矩阵左乘*）**

***提示:　Non-Standard Transformation Solutions***

1. *首先，T1-把”非标准”变换条件转换为标准变换条件*
2. *然后，　T2-实施标准变换*
3. *最后，T3-逆变换回非标准条件下*

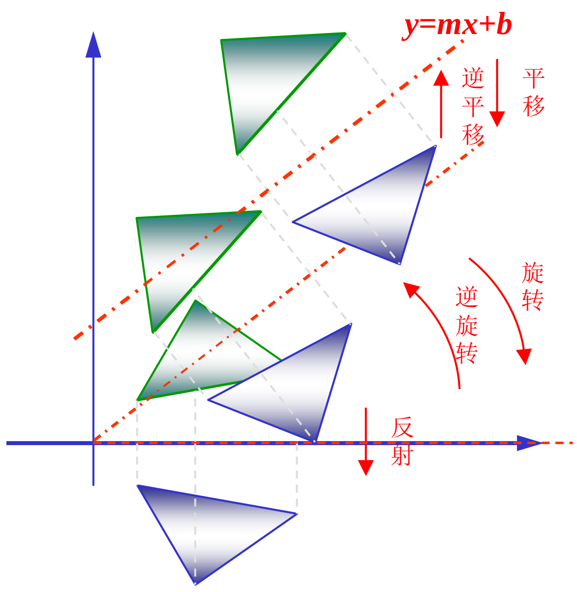
# Non-standard transformation(cont.)

The University of New Mexico

## ➢ Practice(cont.)

推导关于任意轴y=ax+b的对称变换矩阵思路:

– ①平移反射轴使其经过原点;
– ②将反射轴旋转到坐标轴之一上，且进行关于坐标轴反射;
– ③利用逆旋转和平移变换将线置回原处。




- 串联矩阵M为:

M= T(0,b) •R(θ) • $F_{x轴}$• R(-θ) • T(0,-b)

1)平移 T(0,-b)
2) θ=arctg(a)，顺时针旋转 R(-θ)
3)关于X轴的反射 $F_{X轴}$
4)旋转回去R(θ)
5)平移回去T(0,b)

# Outline

- **Representation（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换

# Cumulative Transformation

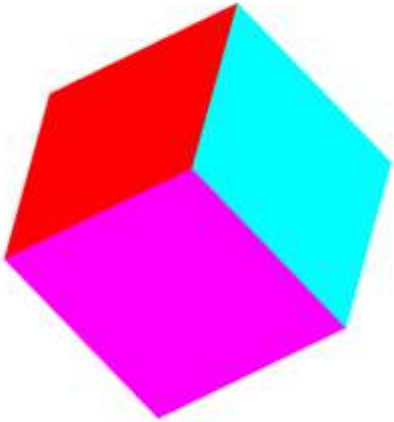
The University of New Mexico

- **累积变换的例子: Virtual trackball 虚拟跟踪球模拟**

- Smooth Rotation：From a practical standpoint, we are often want to use transformations to move and reorient an object smoothly

ref：Angel8ECode/ 04/trackball.*

  - 让物体绕过中心的任意轴进行旋转;
  - 旋转的任意轴是可以改变的，且由鼠标滑动方向来进行交互控制改变



# Cumulative Transformation(cont.)

The University of New Mexico

- **累积变换的例子: Virtual trackball 虚拟跟踪球模拟(cont.)**
- 第n帧画面中的顶点计算公式为: $P_n = R_n P_0$

```
<script id="vertex-shader" type="x-shader/x-vertex">
in  vec4 vPosition;
in  vec4 vColor;
Out  fColor;
uniform mat4 r;
void main(){
  vec4 p;
  gl_Position = r *vPosition;
  fColor = vColor;
}
```

- $R_n$是累积变换矩阵，多帧画面对应有各自的串联变换矩阵，形成变换序列$R_0, R_1, \ldots\ldots, R_n$，且存在累积效应：$R_n = \boldsymbol{R_{cur}} R_{n-1}$
- $\boldsymbol{R_{cur}}$是当前最近一次的任意轴旋转矩阵，有以下两种计算方式：
  - find the Euler angles(欧拉角)，use $\boldsymbol{R_{cur}} = \boldsymbol{R_{iz} R_{iy} R_{ix}}$
    - Not very efficient
  - ✓find the rotation axis and angle ,use $R_{cur} = R(axis_i, angle_i)$,
    - ✓Efficient
- *Quaternions（四元数法）can be more efficient than either*

# Cumulative Transformation（cont.)

The University of New Mexico

- 累积变换的例子: Virtual trackball 虚拟跟踪球模拟（cont.)
- 累积变换矩阵的实现代码：

$$R_n = \boldsymbol{R_{cur}}\, R_{n-1}$$

```
Function render(){
  gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
   //鼠标键没有放开并继续移动时，则继续计算新的旋转轴和角度，得到新的一次旋转变换
   矩阵rotate(angle, axis)，累乘后传递rotationMatrix给着色器
  if(trackballMove) {  // 这里的rotationMatrix 就是Rn  (Rn = Rcur Rn-1)
    axis = normalize(axis); //下页讲如何计算axis
    //rotationMatrix = mult(rotationMatrix, rotate(angle, axis)); //error
    rotationMatrix = mult( rotate(angle, axis),rotationMatrix); //right
    gl.uniformMatrix4fv(rotationMatrixLoc, false, flatten(rotationMatrix));
  }
  gl.drawArrays( gl.TRIANGLES, 0, NumVertices );//绘制
  requestAnimFrame( render );//动画切换帧
}
```

# Cumulative Transformation(cont.)

## Ex: Virtual trackball 虚拟跟踪球模拟(cont.)

- 当前旋转变换矩阵的实现代码: $R_{cur} = R(axis_i, angle_i)$

●ref: Angel8ECode/common\MV.js

```
/**生成绕任意轴旋转的变换矩阵*/
function rotate( angle, axis )
{
    if ( !Array.isArray(axis) ) {
        axis = [ arguments[1], arguments[2], arguments[3] ];
    }

    var v = normalize( axis );
    var x = v[0];
    var y = v[1];
    var z = v[2];

    var c = Math.cos( radians(angle) );
    var omc = 1.0 - c;
    var s = Math.sin( radians(angle) );

    var result = mat4(
        vec4( x*x*omc + c,   x*y*omc - z*s, x*z*omc + y*s, 0.0 ),
        vec4( x*y*omc + z*s, y*y*omc + c,   y*z*omc - x*s, 0.0 ),
        vec4( x*z*omc - y*s, y*z*omc + x*s, z*z*omc + c,   0.0 ),
        vec4()
    );

    return result;
}
```

- *思考: 参数angle, axis如何计算?*

# Cumulative Transformation(cont.)

The University of New Mexico

## Ex: Virtual trackball 虚拟跟踪球模拟(cont.)

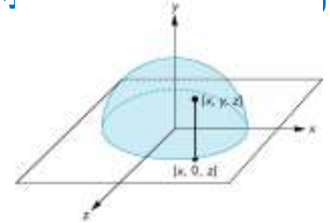
➢ $R_{cur} = R\,(axis_i, angle_i)$中参数axis 和 angle的计算方法

- 将轨迹球的位置和在正常鼠标垫上的位置联系起来，将鼠标位置p(x,z)反投影为半球坐标p(x,y,z)

$$Y=\sqrt{r^2 - x^2 - z^2},$$

if r □ |x|□ 0, r □ |z| □ 0

origin at center of ball



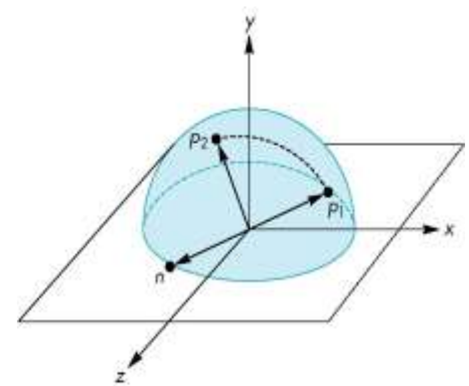
- 从鼠标上获得的两个点，将它们投射到上半球的点p1和点p2上，这些点决定了球体上的一个大圆

  - 从p1旋转到p2旋转轴axis，由原点，p1和p2决定的平面的法线表示

$$\mathbf{n} = \mathbf{p}_1 \times \mathbf{p}_2$$

  - p1,p2之间的角度angle,

$$|\sin\theta| = \frac{|\mathbf{n}|}{|\mathbf{p}_1||\mathbf{p}_2|}$$




当鼠标移动缓慢或频繁采样移动点时，

可以近似计算角度： $\theta \approx \sin\theta$

# Cumulative Transformation(cont.)

The University of New Mexico

## Ex: Virtual trackball 虚拟跟踪球模拟(cont.)

- $R_{cur} = R\,(axis_i, angle_i)$中参数axis 和 angle的实现代码

```
/*根据输入的屏幕鼠标点击位置X,Y, 将屏幕坐标2D转换为板球面上的3D坐标输出，并且单位化使得X2+Y2+Z2=1被mouseMotion调用*/
function trackballView( x,  y ) {
    var d, a;
    var v = [];
    v[0] = x;
    v[1] = y;
    d = v[0]*v[0] + v[1]*v[1];
    if (d < 1.0)
      v[2] = Math.sqrt(1.0 - d);
    else {
      v[2] = 0.0;
      a = 1.0 /  Math.sqrt(d);
      v[0] *= a;
      v[1] *= a;
    }
    return v;
}
```

```
/*mouseMotion当鼠标按下了并移动时，根据新的位置X,Y,转换为半球上三维坐标后，结合保留的上一次的三维位置，计算得出旋转轴及旋转角度，并且保留本次3D 位置。*/
function mouseMotion( x,  y)
{   var dx, dy, dz;
    var curPos = trackballView(x, y);

    if(trackingMouse) {
      dx = curPos[0] - lastPos[0];
      dy = curPos[1] - lastPos[1];
      dz = curPos[2] - lastPos[2];

      if (dx || dy || dz) {
         angle = -0.1 * Math.sqrt(dx*dx + dy*dy + dz*dz);
         axis[0] = lastPos[1]*curPos[2] - lastPos[2]*curPos[1];
         axis[1] = lastPos[2]*curPos[0] - lastPos[0]*curPos[2];
         axis[2] = lastPos[0]*curPos[1] - lastPos[1]*curPos[0];
         lastPos[0] = curPos[0];
         lastPos[1] = curPos[1];
         lastPos[2] = curPos[2];
      }
    }
    render();
}
```

# Outline

- **Representation（对象在物理空间的表示）**
  - Vector space, Coordinate System, Change of Coordinate
  - Affine space, Frames System, Change of Frame
  - Homogeneous Coordinate
- **Transformations*（变换）**
  - Five Standard Transformation（标准变换）
  - Concatenation Transformation*（串联变换）
- **Applying Concatenation Transformation（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - …

# Summary

- **Homogeneous Coordinates Representation（齐次坐标表示）**
  - Point(x,y,z,h)，顶点表示，h!=0
  - Vector(x,y,z,0) 向量表示：h=0

$$M_{4*4}=\begin{bmatrix} sx & b & c & tx \\ d & sy & f & ty \\ g & h & sz & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Affine Transformation(仿射变换,五种基本变换)：**
  - Five Standard Transformation：Translation, Rotation, Scaling, Reflection, Shear.
  - Use Homogeneous Coordinates, Transformation is a 4*4 Matrix, 12 freedom
  - Use column vector representation , affine transformation then like:
- **Concatenation Transformation（串联变换）：**

  Pn’ = Mn*…*M2*M1* P0; //Mi is Standard Transformation

  **M**=Mn*…*M2*M1; //M is Concatenation Transformation
- **Applying Concatenation Transformations（应用串联变换）**
  - Eg1:Non-standard transformation非标变换
  - Eg2:Cumulative transformation累积变换
  - **……（下次课讲顶点着色器中的MVP变换，即观察流水线中的坐标变换）**