Computação Gráfica I

Professor:

Anselmo Montenegro
www.ic.uff.br/~anselmo

Conteúdo:

- Transformações geométricas no espaço

Transformações geométricas no espaço: *Introdução*

- As transformações geométricas são operações fundamentais para a modelagem, visualização e interação com objetos gráficos 3D.

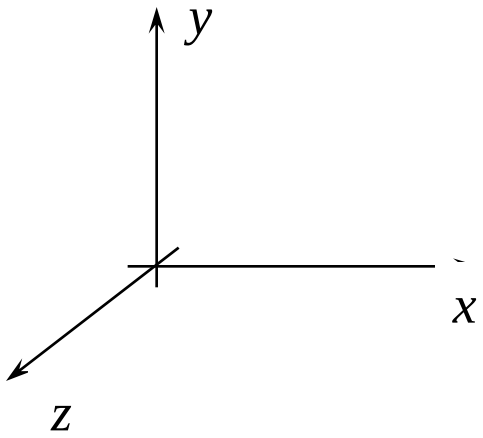
- Descreveremos os seguintes tópicos:
  - Escalas, rotações e translações no espaço.
  - Esquemas para *representação de orientações*.
  - *Composição de transformações*:
    - Instanciação de objetos.
    - Hierarquia.

Transformações geométricas espaço: *Introdução*

- Transformações de escala, rotação e translação são fundamentais para a *criação de cenas compostas por diversos objetos*.

- As matrizes de translação e escala são de fato uma simples extensão das matrizes de transformação definidas no plano.

Transformações geométricas no espaço:
Translações e escalas



Translação

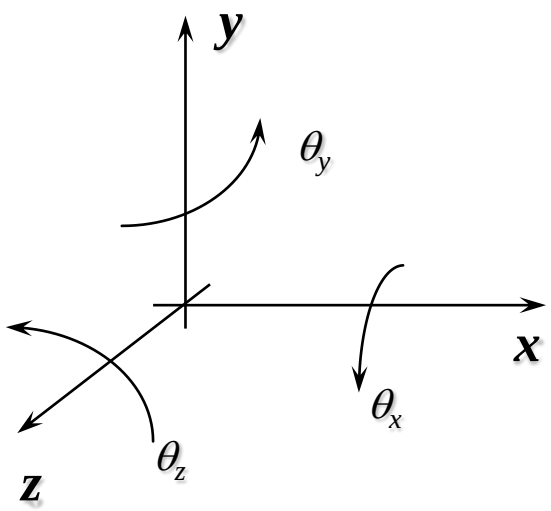
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \left[\begin{array}{ccc:c} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Escala

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \left[\begin{array}{ccc:c} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações geométricas no espaço: *Rotações*

- As operações de rotação no espaço são mais complexas do que no plano.
- Uma extensão natural é definirmos a rotação de um objeto a partir da *rotação em torno dos eixos cartesianos*.

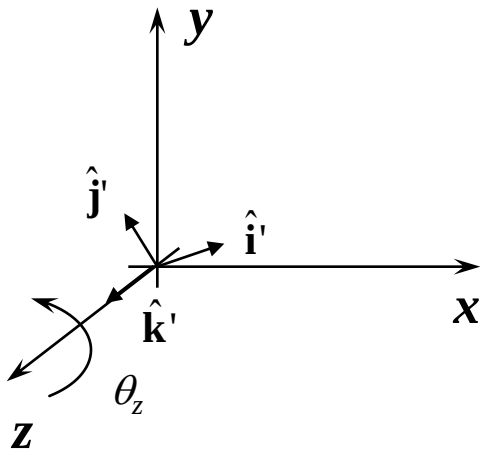


Transformações geométricas no espaço: *Rotações*

- Podemos facilmente definir as matrizes de rotação em cada eixo.

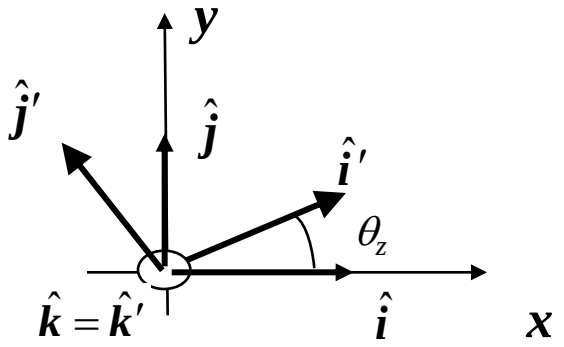
- As *colunas* de uma matriz de rotação em torno de um certo eixo cartesiano são dados pela *transformação dos vetores da base canônica*.

Transformações geométricas no espaço:
Rotações no eixo z



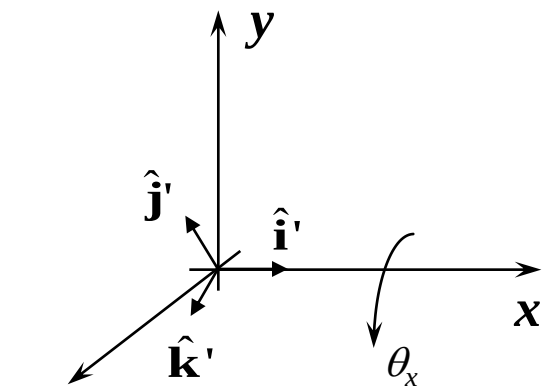
$$\hat{i}' = \begin{bmatrix} \cos\theta_z \\ \sin\theta_z \\ 0 \end{bmatrix} \quad \hat{j}' = \begin{bmatrix} -\sin\theta_z \\ \cos\theta_z \\ 0 \end{bmatrix} \quad \hat{k}' = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

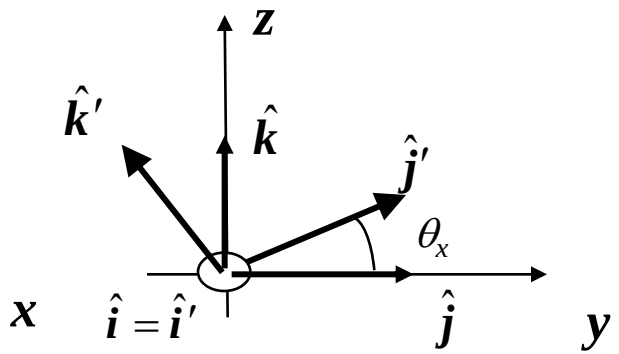
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações geométricas no espaço:
Rotações no eixo x



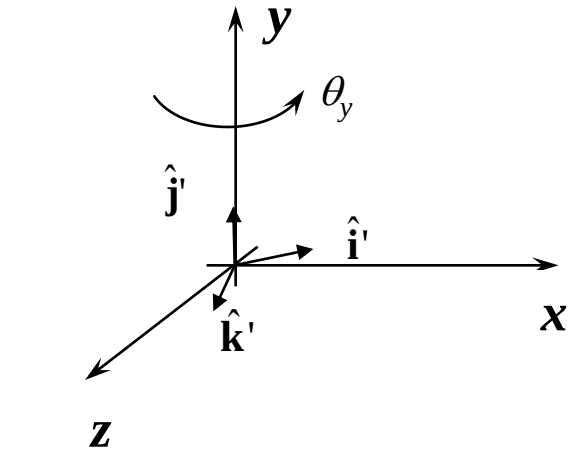


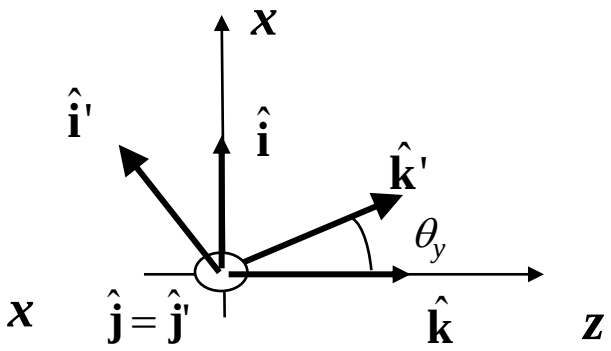
$$\hat{i}' = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \hat{j}' = \begin{bmatrix} 0 \\ \cos\theta_x \\ \sin\theta_x \end{bmatrix} \qquad \hat{k}' = \begin{bmatrix} 0 \\ -\sin\theta_x \\ \cos\theta_x \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações geométricas no espaço:
Rotações no eixo y



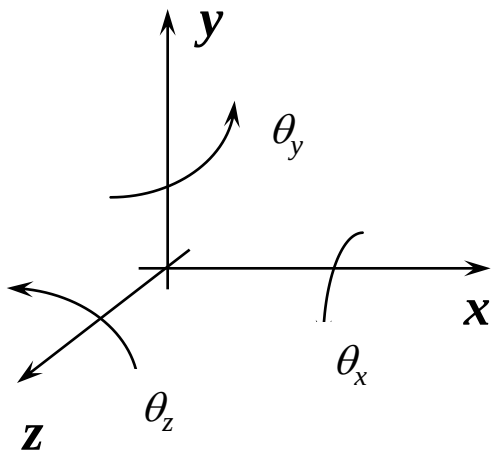


$$i' = \begin{bmatrix} \cos\theta_y \\ 0 \\ -\sin\theta y \end{bmatrix} \quad \hat{j}' = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \hat{k}' = \begin{bmatrix} \sin\theta_y \\ 1 \\ \cos\theta y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações geométricas no espaço:
Rotações nos eixos cartesianos



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações geométricas no espaço: *matrizes de transformação*

- Em geral, uma *matriz de transformação em coordenadas homogêneas* tem a seguinte estrutura:

$$\begin{bmatrix} M & T \\ s & 1 \end{bmatrix}$$

- Se a matriz M e o vetor s tem dimensões 2x2 e 1x2, respectivamente, então a transformação ocorre no *plano homogêneo*.
- Se as dimensões forem 3x3 e 3x1, respectivamente, então a transformação ocorre no *espaço homogêneo*.

Transformações geométricas no espaço: *matrizes de transformação*

- A forma matricial homogênea pode representar
  a) Transformações lineares.
  b) Translações
  c) Transformações afins.

$$\begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} I & T \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} M & T \\ 0 & 1 \end{bmatrix}$$

(a) (b) (c)

Transformações geométricas no espaço: *matrizes de transformação*

- A matriz que representa uma *transformação afim* representa uma transformação linear seguida de uma translação:

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

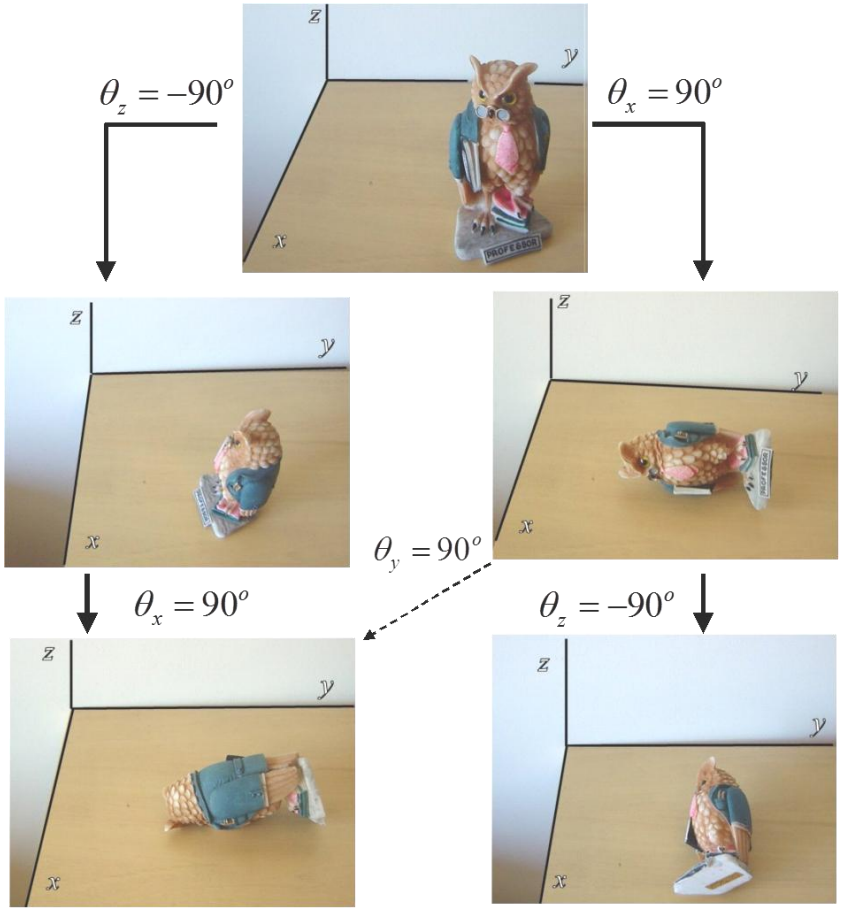
- Caso invertêssemos a ordem teríamos

$$\begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} M & MT \\ 0 & 1 \end{bmatrix}$$

Transformações geométricas no espaço: *matrizes de transformação*

- Como podemos ver, a *ordem influi no resultado*.
- Na segunda ordem, a translação não pode ser lida diretamente da última coluna última matriz,.
- Como a última linha é o vetor $[0..0\ 1]$ então *estas matrizes mantêm os pontos no plano* $w = 1$.
- Nas transformações projetivas, que serão vistas mais tarde, a última linha assume outros valores e os pontos podem ser deslocados do plano $w=1$.

Transformações geométricas no espaço: *rotações e orientações*



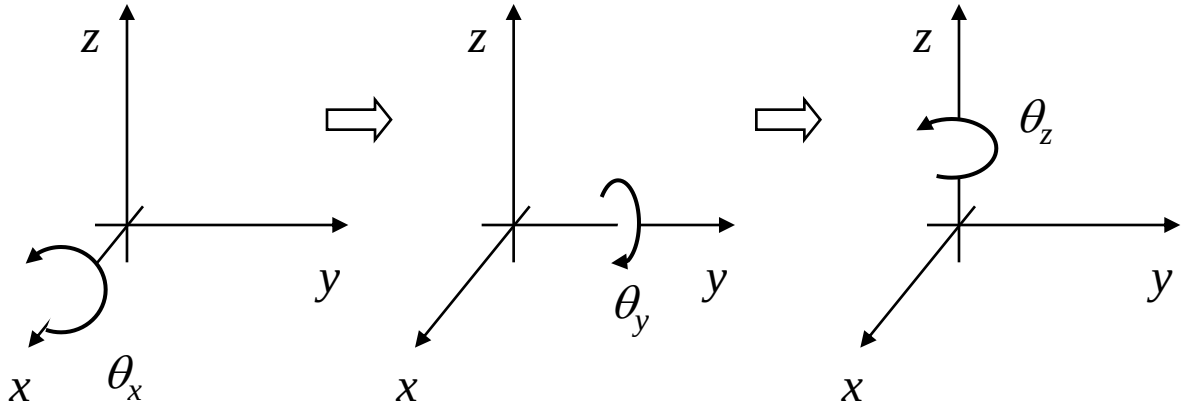
Transformações geométricas no espaço: *Ângulos de Euler*

- Já que as *rotações não comutam devemos adotar uma ordem específica*.

- Esta forma de representar orientações é denominada *Ângulos de Euler*.

- Na literatura de aeronáutica estas rotações são chamadas de
  - Roll – giro em torno do eixo longitudinal
  - Pitch – ângulo de ataque.
  - Yaw – giro em torno do eixo vertical.

Transformações geométricas no espaço: *ângulos de Euler*



$$\mathbf{R}(\theta_x,\theta_y,\theta_z)=\begin{bmatrix} c_y c_z & c_y s_z & -s_y & 0 \\ s_x s_y c_z - c_x s_z & s_x s_y s_z + c_x c_z & s_x c_y & 0 \\ c_x s_y c_z + s_x s_z & c_x s_y s_z - s_x c_z & c_x c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$s_x = \sin(\theta_x),\ s_y = \sin(\theta_y),\ s_z = \sin(\theta_z)$$

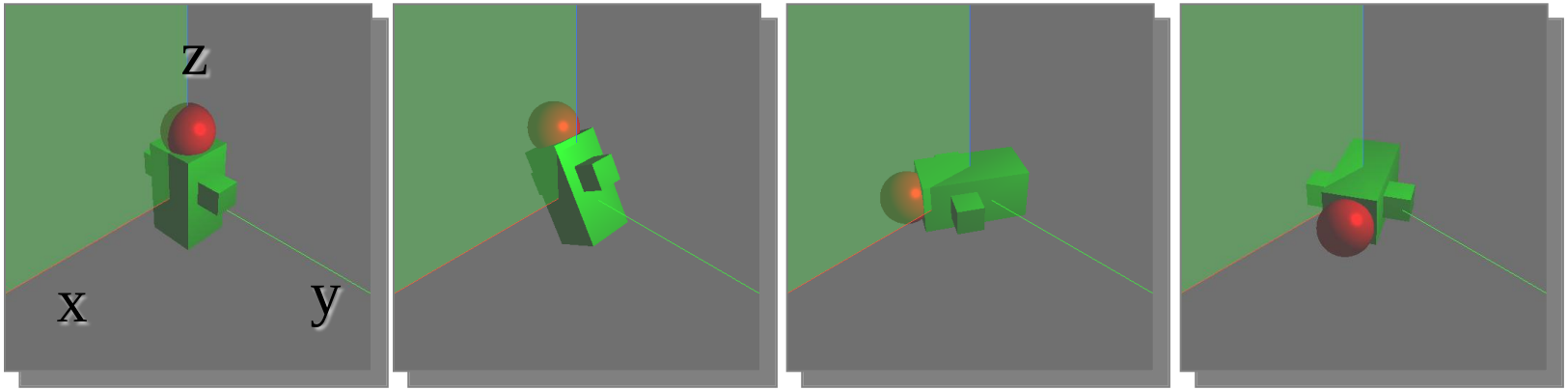
$$c_x = \cos(\theta_x),\ c_y = \cos(\theta_y),\ c_z = \cos(\theta_z)$$

Transformações geométricas no espaço: *Ângulos de Euler*

- Problemas:
  - *Gimbal lock*: perda de graus de liberdade em certas configurações.
  - Não são parâmetros adequados para interpolações.

Transformações geométricas no espaço: *ângulos de Euler – Gimbal Lock*

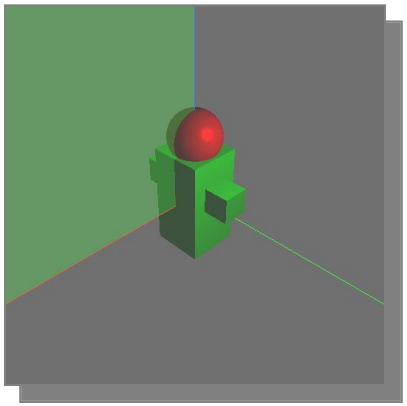
- Animador deseja rodar o boneco de lado (θ_x= 30°) graus, inclinálo para frente (θ_y= 90°) e levantar seu braço esquerdo.
- A última operação não será possível.



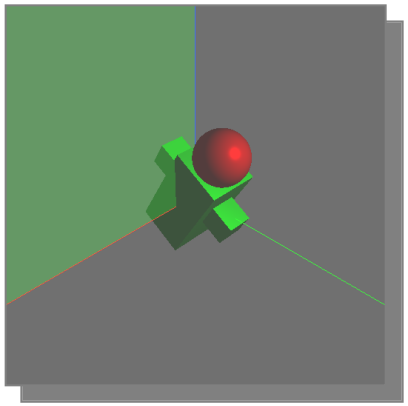
original | θ_x=30° | θ_x=30° θ_y=90° | θ_x=30° θ_y=90° θ_z=60°

Transformações geométricas no espaço: *ângulos de Euler – Gimbal Lock*

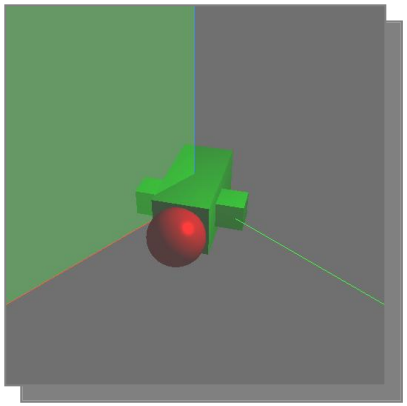
- Mesmo resultado obtido apenas com rotações no eixo x e y.



original



θ_x=-60°



θ_x=-60°

θ_y=90°

Transformações geométricas no espaço: *ângulos de Euler*

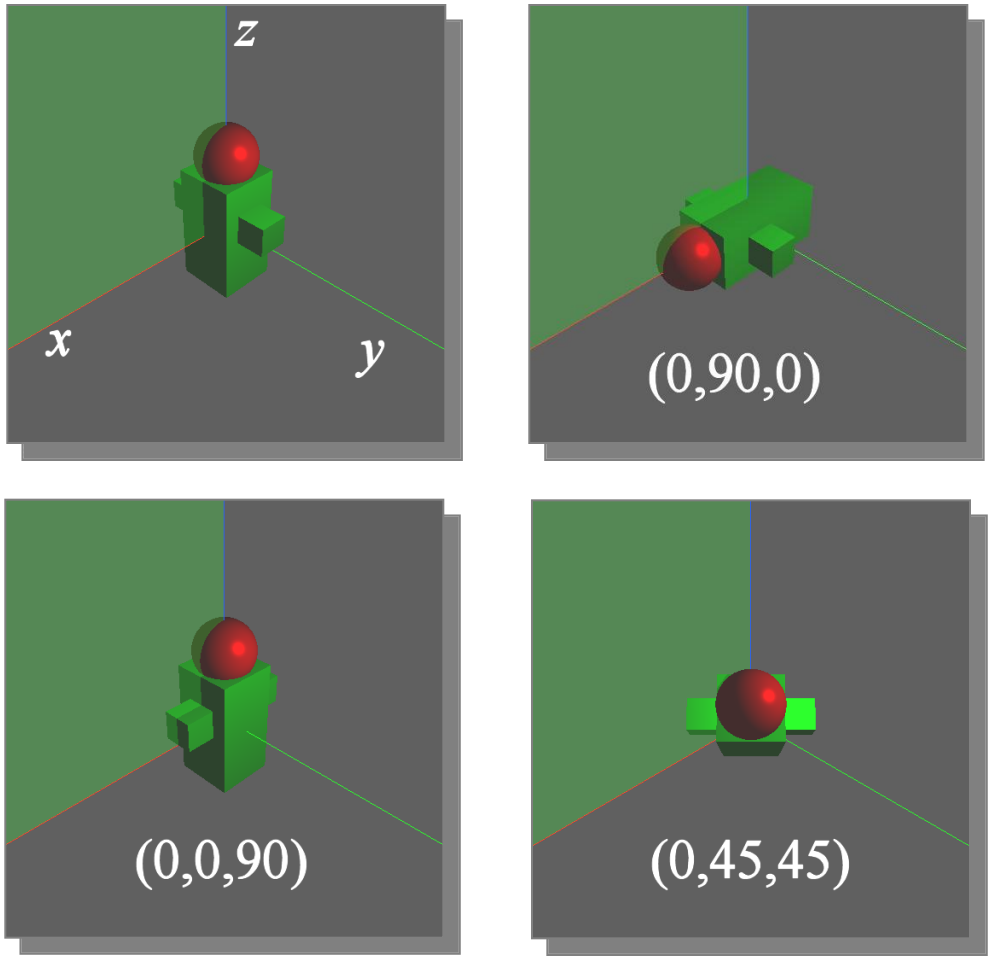
$$\mathbf{R}(\theta_x,\theta_y,\theta_z)=\begin{bmatrix} c_y c_z & c_y s_z & -s_y & 0 \\ s_x s_y c_z - c_x s_z & s_x s_y s_z + c_x c_z & s_x c_y & 0 \\ c_x s_y c_z + s_x s_z & c_x s_y s_z - s_x c_z & c_x c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}(\theta_x,90^o,\theta_z)=\begin{bmatrix} 0 & 0 & -1 & 0 \\ s_x c_z - c_x s_z & s_x s_z + c_x c_z & 0 & 0 \\ c_x c_z + s_x s_z & c_x s_z - s_x c_z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ \sin(\theta_x-\theta_z) & \cos(\theta_x-\theta_z) & 0 & 0 \\ \cos(\theta_x-\theta_z) & \sin(\theta_x-\theta_z) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Apesar de especificarmos 2 parâmetros só temos 1 grau de liberdade

Transformações geométricas no espaço: *ângulos de Euler – interpolação*

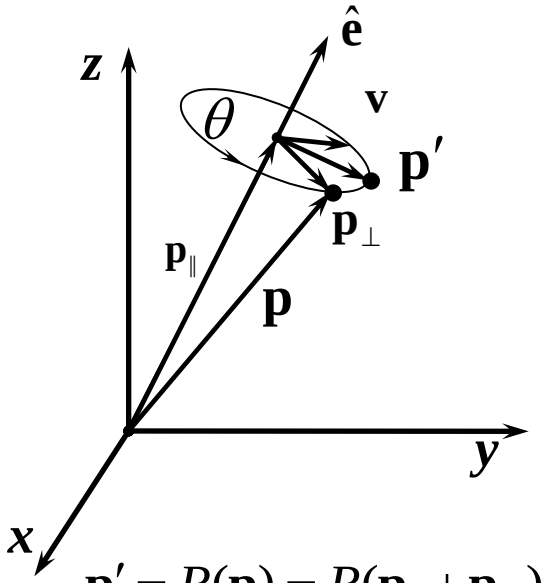
- Interpolação entre as orientações (0,90,0) e (0,90,0).
- Note que em uma interpolação mais natural a cabeça não sairia tanto do plano xz.

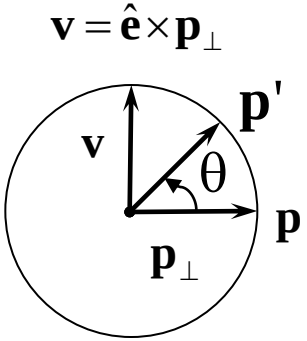


Transformações geométricas no espaço: *outras formas de se expecificar orientações*

- Rotações em torno de um eixo:
  - Euler provou em 1775 que dadas duas posições rotacionadas de um objeto, é sempre possível levar uma posição a outra através de uma *rotação de um ângulo em torno de um eixo*.
  - Esta rotação tem a mesmo comportamento que a interpolação de duas posições através de um segmento de reta que os une.
  - Sai da primeira posição indo para a segunda sem oscilações.

Transformações geométricas no espaço: *rotação em torno de um eixo ê*





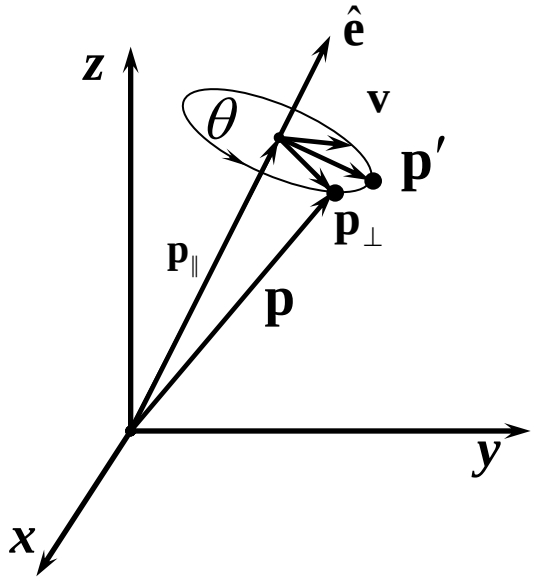
$$\mathbf{p}' = R(\mathbf{p}) = R(\mathbf{p}_{\parallel} + \mathbf{p}_{\perp})$$

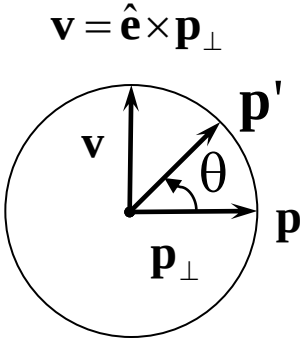
$$\mathbf{p}' = R(\mathbf{p}_{\parallel}) + R(\mathbf{p}_{\perp})$$

$$\mathbf{p}' = \mathbf{p}_{\parallel} + (\cos\theta)\mathbf{p}_{\perp} + (sen\,\theta)\mathbf{v}$$

$$\mathbf{p}' = (\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}} + (\cos\theta)(\mathbf{p} - (\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}}) + (sen\,\theta)(\hat{\mathbf{e}}\times\mathbf{p})$$

Transformações geométricas no espaço: *rotação em torno de um eixo ê*





$$\mathbf{p}' = (\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}} + (\cos\theta)(\mathbf{p} - (\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}}) + (sen\,\theta)(\hat{\mathbf{e}}\times\mathbf{p})$$

$$\mathbf{p}' = (\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}} + (\cos\theta)\mathbf{p} - (\cos\theta)(\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}} + (sen\,\theta)(\hat{\mathbf{e}}\times\mathbf{p})$$

$$\mathbf{p}' = (\cos\theta)\mathbf{p} + (1-\cos\theta)(\hat{\mathbf{e}}\cdot\mathbf{p})\hat{\mathbf{e}} + (sen\,\theta)(\hat{\mathbf{e}}\times\mathbf{p})$$

Transformações geométricas no espaço: *rotação em torno de um eixo ê*

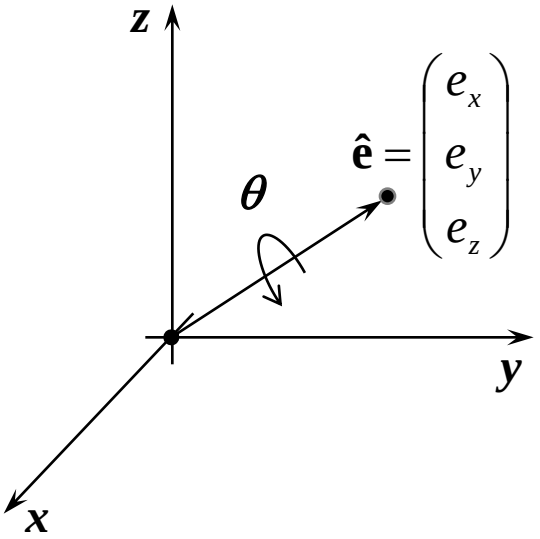
$$\begin{pmatrix} m_{11} \\ m_{21} \\ m_{22} \end{pmatrix} = \cos\theta \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + (1-\cos\theta) e_x \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} + \sin\theta \begin{pmatrix} 0 \\ e_z \\ -e_y \end{pmatrix} = \begin{pmatrix} \cos\theta + (1-\cos\theta) e_x^2 \\ e_x e_y (1-\cos\theta) + e_z \sin\theta \\ e_x e_z (1-\cos\theta) - e_y \sin\theta \end{pmatrix}$$

$$\begin{pmatrix} m_{12} \\ m_{22} \\ m_{32} \end{pmatrix} = \cos\theta \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + (1-\cos\theta) e_y \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} + \sin\theta \begin{pmatrix} -e_z \\ 0 \\ e_x \end{pmatrix} = \begin{pmatrix} e_y e_x (1-\cos\theta) - e_z \sin\theta \\ \cos\theta + (1-\cos\theta) e_y^2 \\ e_y e_z (1-\cos\theta) - e_x \sin\theta \end{pmatrix}$$

$$\begin{pmatrix} m_{13} \\ m_{23} \\ m_{33} \end{pmatrix} = \cos\theta \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + (1-\cos\theta) e_z \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} + \sin\theta \begin{pmatrix} e_y \\ -e_x \\ 0 \end{pmatrix} = \begin{pmatrix} e_z e_x (1-\cos\theta) + e_y \sin\theta \\ e_z e_y (1-\cos\theta) - e_x \sin\theta \\ \cos\theta + (1-\cos\theta) e_z^2 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

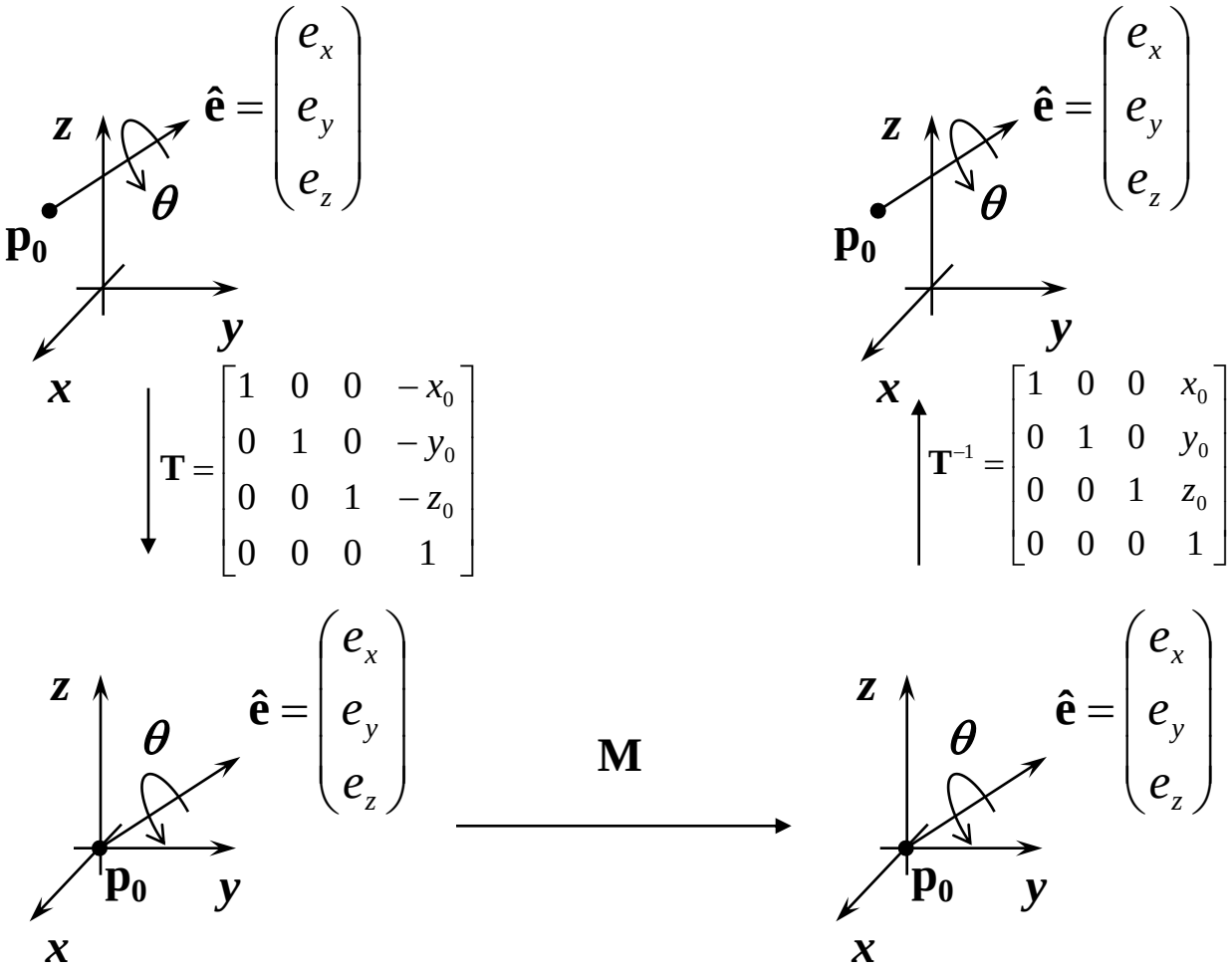
Transformações geométricas no espaço: *rotação em torno de um eixo ê*



$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\mathbf{M} = \begin{bmatrix} \cos\theta + (1-\cos\theta)e_x^2 & e_y e_x (1-\cos\theta) - e_z \sin\theta & e_z e_x (1-\cos\theta) + e_y \sin\theta & 0 \\ e_x e_y (1-\cos\theta) + e_z \sin\theta & \cos\theta + (1-\cos\theta)e_y^2 & e_z e_y (1-\cos\theta) - e_x \sin\theta & 0 \\ e_x e_z (1-\cos\theta) - e_y \sin\theta & e_y e_z (1-\cos\theta) - e_x \sin\theta & \cos\theta + (1-\cos\theta)e_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformações geométricas no espaço:
rotação em torno de um eixo ê fora da origem



Transformações geométricas no espaço: *quaternios*

- A especificação de orientações através de rotações em torno de um eixo *não fornece uma álgebra simples* para as diversas operações necessárias para animação.

- Para isso existe uma estrutura matemática mais adequada denominada *quatérnios*.

- No século 18, W. R. Hamilton propôs os *quatérnios* como uma extensão em quatro dimensões para os números complexos.

Transformações geométricas no espaço: *quaternios*

- *Quatérnios* podem representar rotações e orientações no espaço tridimensional.
- Existem diferentes notações para *quatérnios*
- Uma possível forma é representá-los conforme abaixo:
  - $\mathbf{q} = (q_v, q_w) = iq_x + jq_y + kq_z + q_w = \mathbf{q_v} + q_w$
  - $q_v = iq_x + jq_y + kq_z = (q_x, q_y, q_z)$
  - $i^2 = j^2 = k^2 = -1$ and $ij = -ji = k$, $jk = -kj = i$, $ki = -ik = -j$

Transformações geométricas no espaço: *quaternions – operações fundamentais*

- **Adição: q** + **r** = $(q_v,q_w)+(r_v,r_w)$

  Multiplicação: qr = $(q_v \times r_v + r_w q_v + q_w r_v,\ q_w r_w - q_v . r_v)$, onde "." indica produto escalar e "x" produto vetorial); Obs.: qr ≠ rq

- **Conjugado:** $q^* = (q_v,q_w)^* = (-q_v,q_w)$
- **Norma:** $n(q)^2 = qq^* = q^*q = q_v . q_v + q_w^2 = qx^2 + qy^2 + qz^2 + qw^2$

- **Inverso:** $q^{-1} = q^* / N(q)^2$

  Identidade:
  - (0,0,0,1) - multiplicação
  - (0,0,0,0) – adição
- Outras operações podem ser derivadas das operações básicas.

Transformações geométricas no espaço: *quaternios unitários*

- Um quatérnio $\mathbf{q} = (q_v, q_w)$ é *unitário* se $n(\mathbf{q}) = 1$.
- Pode-se escrever um quatérnio unitário como

$$q = (\sin\phi \mathbf{u_q}, \cos\phi) = \sin\phi \mathbf{u_q} + \cos\phi$$

- onde $\mathbf{u_q}$ é um vetor 3d tal que $||\mathbf{u_q}||=1$
- *Quatérnios unitários* são perfeitamente apropriados para representar rotações e orientações.

Transformações geométricas no espaço: *quaternions e rotações*

- Pode ser provado que a rotação de um vetor v por um *quatérnio unitário* e´ dado por:

$$\mathbf{p´ = q\ p\ q^*} = q\ p\ q^{-1}\text{ , onde } p = (p_x\ p_y\ p_z\ p_w)^T$$

- Dados dois quatérnios unitários **q** e **r**. A concatenação da aplicação de **q** sobre **p** seguida de **r** é dada por:

$$\mathbf{r\ (q\ p\ q^*)\ r^* = (r\ q)\ p\ (rq)^* = c\ p\ c^*}$$

Transformações geométricas no espaço: *quaternions e API's gráficas*

- O modo *retained* do Direct3D e o XNA suportam *quatérnio*.
- OpenGL não fornece suporte direto a *quatérnio*.
- Como resultado é necessário converter orientações em *quatérnio* para outra forma de representação.

Transformações geométricas no espaço: *quaternions e API's gráficas*

- Tanto *OpenGL* quanto *Direct3D* fornecem modos de se especificar orientações via matrizes.

- Logo, a conversão *quatérnio-matriz* é necessária.

- Tal conversão é necessário se for necessário importar orientações de sistemas gráficos que não adotam *quatérnio* como o *LightWave*.

Transformações geométricas no espaço: conversão *ângulo e eixo -quaternio*

- A conversão da forma *ângulo e eixo* para quatérnio é simples.

- Utiliza duas operações trigonométricas e operações de divisão e multiplicação.

$$q = [\cos(Q/2), \sin(Q/2)v] ,$$

- onde Q um ângulo e v um eixo.

Transformações geométricas no espaço: conversão *ângulos de Euler - quaternion*

- A conversão de ângulos de Euler para quatérnio segue um padrão similar.
- É necessário apenas ficar atento quanto a ordem.
- Supondo a forma *yaw*, *pitch* e *roll* temos a seqüência de *quaternios*:

  - $q = q_{yaw}\ qp_{itch}\ q_{roll}$ onde:

    - $q_{pitch} = [\cos(y/2), (\sin(y/2), 0, 0)]$
    - $q_{yaw} = [\cos(q/2), (0, \sin(q/2), 0)]$
    - $q_{roll} = [\cos(f/2), (0, 0, \sin(f/2)]$

Transformações geométricas no espaço: conversão *quaternio - matriz*

- Um quatérnio **q** pode ser convertido em uma matriz **Mq** conforme a expressão abaixo:

$$M^q = \begin{pmatrix} 1-s(q_y^2+q_z^2) & s(q_xq_y-q_wq_z) & s(q_xq_z+q_wq_y) & 0 \\ s(q_xq_y+q_wq_z) & 1-s(q_x^2+q_z^2) & s(q_yq_z-q_wq_x) & 0 \\ s(q_xq_z-q_wq_y) & s(q_yq_z+q_wq_x) & 1-s(q_x^2+q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Onde s = 2n(**q**). Para **q** unitário M^q reduz-se a

$$M^q = \begin{pmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y-q_wq_z) & 2(q_xq_z+q_wq_y) & 0 \\ 2(q_xq_y+q_wq_z) & 1-2(q_x^2+q_z^2) & 2(q_yq_z-q_wq_x) & 0 \\ 2(q_xq_z-q_wq_y) & 2(q_yq_z+q_wq_x) & 1-s(q_x^2+q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformações geométricas no espaço: *conversão matriz -quaternio*

- A conversão matriz-quatérnio requer um conjunto de passos.
- A chave para a conversão surge das diferenças entre elementos da matriz anterior:

$$M^q = \begin{pmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y-q_wq_z) & 2(q_xq_z+q_wq_y) & 0 \\ 2(q_xq_y+q_wq_z) & 1-2(q_x^2+q_z^2) & 2(q_yq_z-q_wq_x) & 0 \\ 2(q_xq_z-q_wq_y) & 2(q_yq_z+q_wq_x) & 1-s(q_x^2+q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$m_{21}^q - m_{12}^q = 4q_wq_x$$
$$m_{02}^q - m_{20}^q = 4q_wq_y$$
$$m_{10}^q - m_{01}^q = 4q_wq_z$$

- Conhecendo-se q_w, é possível determinar o quatérnio $\mathbf{q} = (q_x,q_y,q_z,q_w)$

Transformações geométricas no espaço: *conversão matriz -quatérnio*

- O traço de M^q é dado pela soma dos elementos da diagonal:

$$tr(M^q) = 4 - 2s(q_x^2 + q_y^2 + q_z^2) = 4\left(1 - \frac{q_x^2 + q_y^2 + q_z^2}{q_x^2 + q_y^2 + q_z^2 + q_w^2}\right) = \frac{4q_w^2}{n(\hat{q})}$$

- Com efeito, a conversão para um quatérnio unitário é dada por:

$$q_w = \frac{1}{2}\sqrt{tr(M^q)} \quad q_x = \frac{m_{21}^q - m_{12}^q}{4q_w} \quad q_y = \frac{m_{02}^q - m_{20}^q}{4q_w} \quad q_z = \frac{m_{10}^q - m_{01}^q}{4q_w}$$

Transformações geométricas no espaço: *interpolação de quaternios*

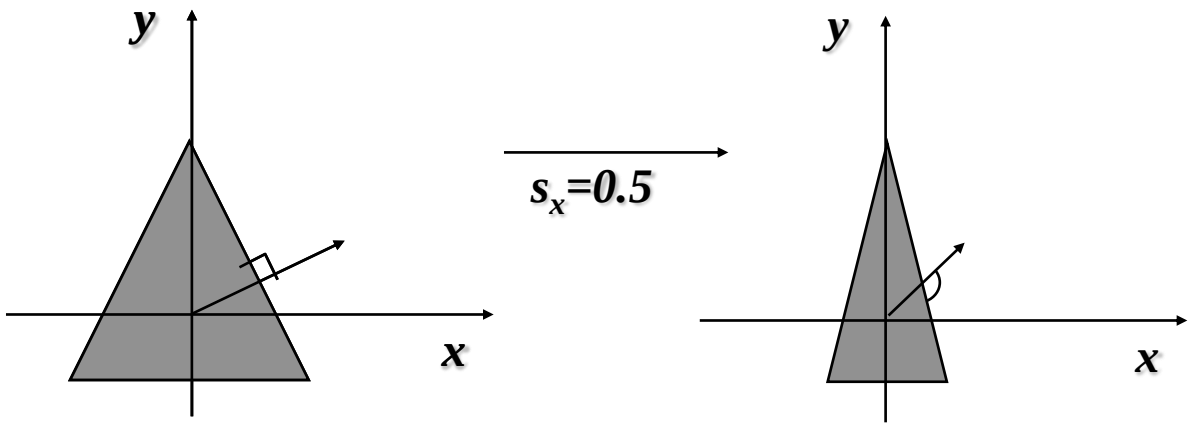
- A interpolação linear esférica entre dois quatérnios **q** e **r,** dado um parâmetro $t \in [0,1]$, calcula a interpolação entre dois quatérnios através da seguinte expressão:
  - **s**(**q**,**r**,t)=(**r**$\mathbf{q}^{-1}$)^t**q**
- Implementações em software utilizam a seguinte fórmula:
  - **s**(**q**,**r**,t)=slerp(**q**,**r**,t) = $\frac{\sin(\phi(1-t))}{\sin\phi}\hat{q} + \frac{\sin(\phi t)}{\sin\phi}\hat{r}$

Transformações geométricas no espaço: Transformações em *OpenGL*

- Translação
  - *glTranslate{fd}(TYPE x, TYPE y, TYPE z);*
- Rotação de *angle* graus em torno de um eixo (x,y,z).
  - *glRotate{fd}(TYPE angle,TYPE x, TYPE y, TYPE z);*
- Escala
  - *glScale{fd}(TYPE sx, TYPE sy, TYPE sz);*

Transformações geométricas no espaço: Transformações em *OpenGL*

- Para transformarmos um certo objeto poligonal basta aplicar a matriz de transformação em cada um dos seus vértices.
- Por outro lado, no caso geral, *as normais destes objetos não seguem a mesma transformação*.
- Exemplo:



Transformações geométricas no espaço: Transformações em *OpenGL*

- Considere o plano com equação

$$n^T p = \begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Se incluirmos a matriz identidade $I = M^{-1}M$ não alteramos a equação abaixo:

$$n^T p = \begin{bmatrix} a & b & c & d \end{bmatrix} M^{-1} M \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações geométricas no espaço: Transformações em *OpenGL*

- A equação do plano transformado $n'.p'=0$ é

$$\begin{bmatrix} a & b & c & d \end{bmatrix} M^{-1} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = 0$$

- Logo, temos que a normal transformada é

$$n' = \begin{bmatrix} a' \\ b' \\ c' \\ d' \end{bmatrix} = M^{-T} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = M^{-T} n$$

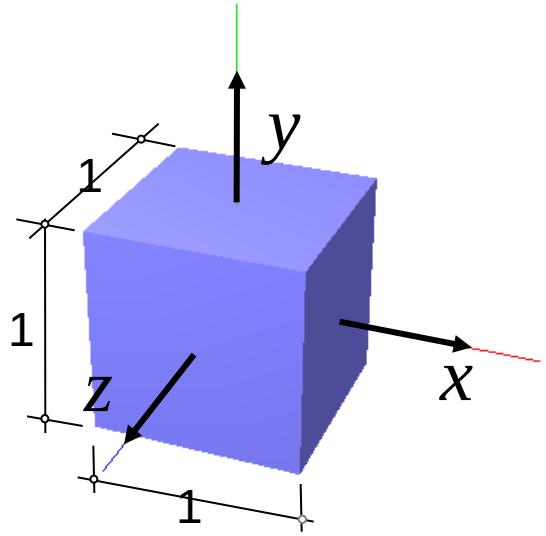
Transformações geométricas no espaço: Transformações em *OpenGL*

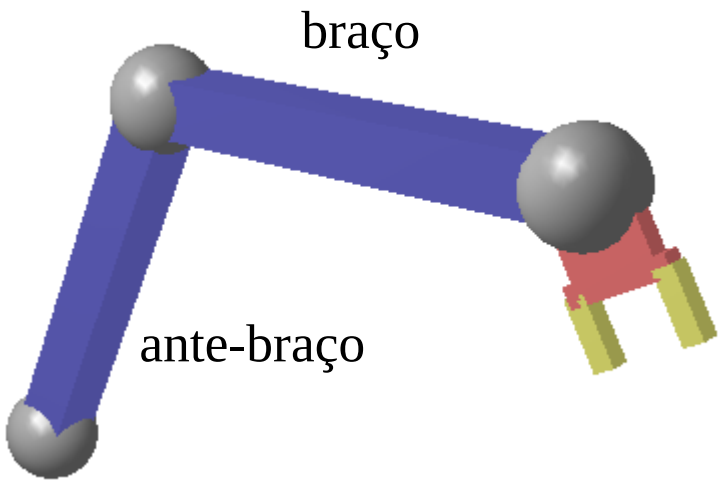
- O processo de instanciação de objeto permite a *especificação de modelos complexos através de modelos padrão simples e transformações geométricas*.

- É fundamental para a descrição de *objetos compostos de várias partes*, principalmente quando há *vínculo* entre as mesmas.

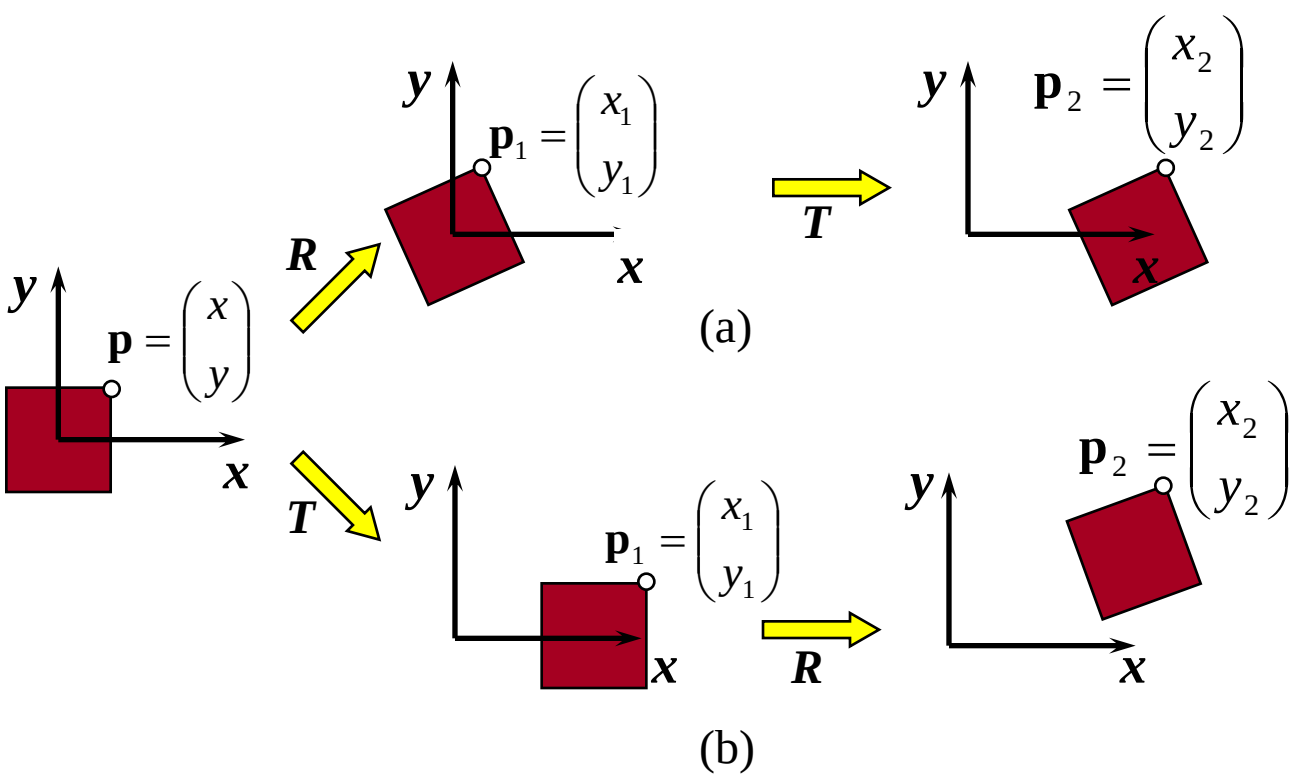
- Devemos primeiramente esclarecer a questão de ordem e interpretação das transformações geométricas compostas.

Transformações geométricas no espaço: Transformações em *OpenGL*





Transformações geométricas no espaço: Transformações em *OpenGL*



Transformações geométricas no espaço: Transformações em *OpenGL*

- Às vezes é difícil especificar transformações geométricas nos casos em que a existe *dependência* entre a posição das partes de um objeto composto.

- Nestas situações é conveniente adotar uma outra interpretação geométrica para as transformações compostas.

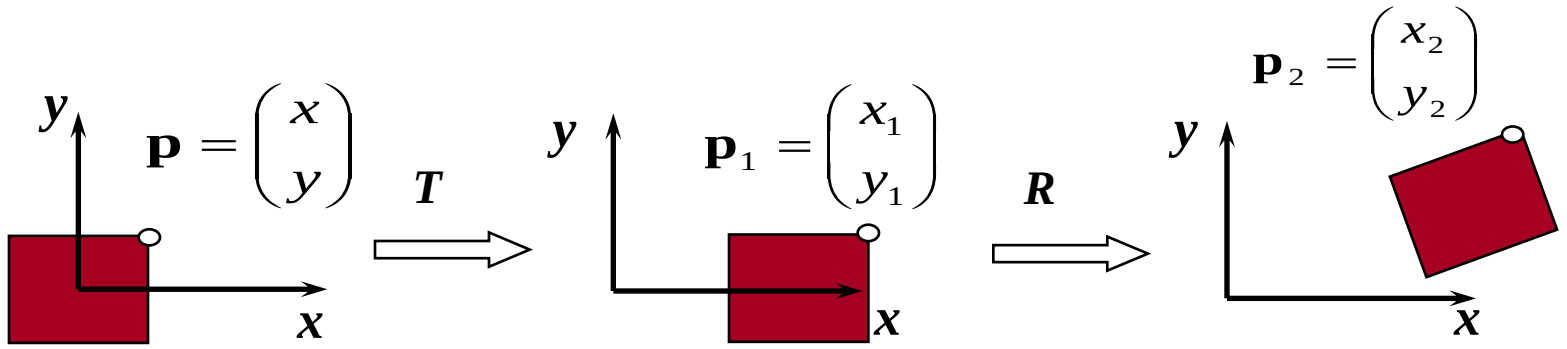
Transformações geométricas no espaço: Transformações em *OpenGL*

- Ao invés de considerarmos que as transformações ocorrem nos objetos, *consideramos que elas ocorrem em um sistema de eixos locais* que rodam e transladam.

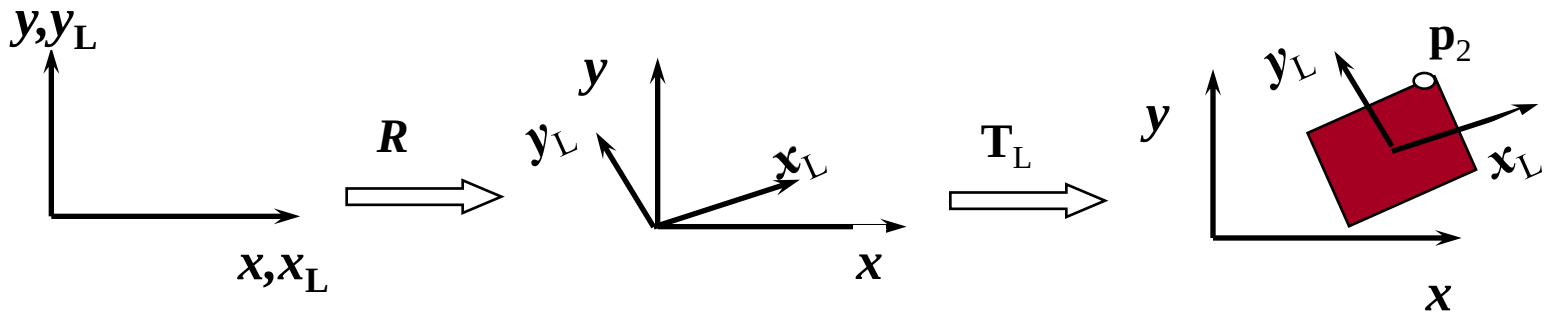
- A idéia é que os eixos locais inicialmente coincidem com o sistema de referência global.

- A cada rotação e translação, *um dado eixo local muda de posição e/ou orientação*.

Transformações geométricas no espaço: Transformações em *OpenGL*

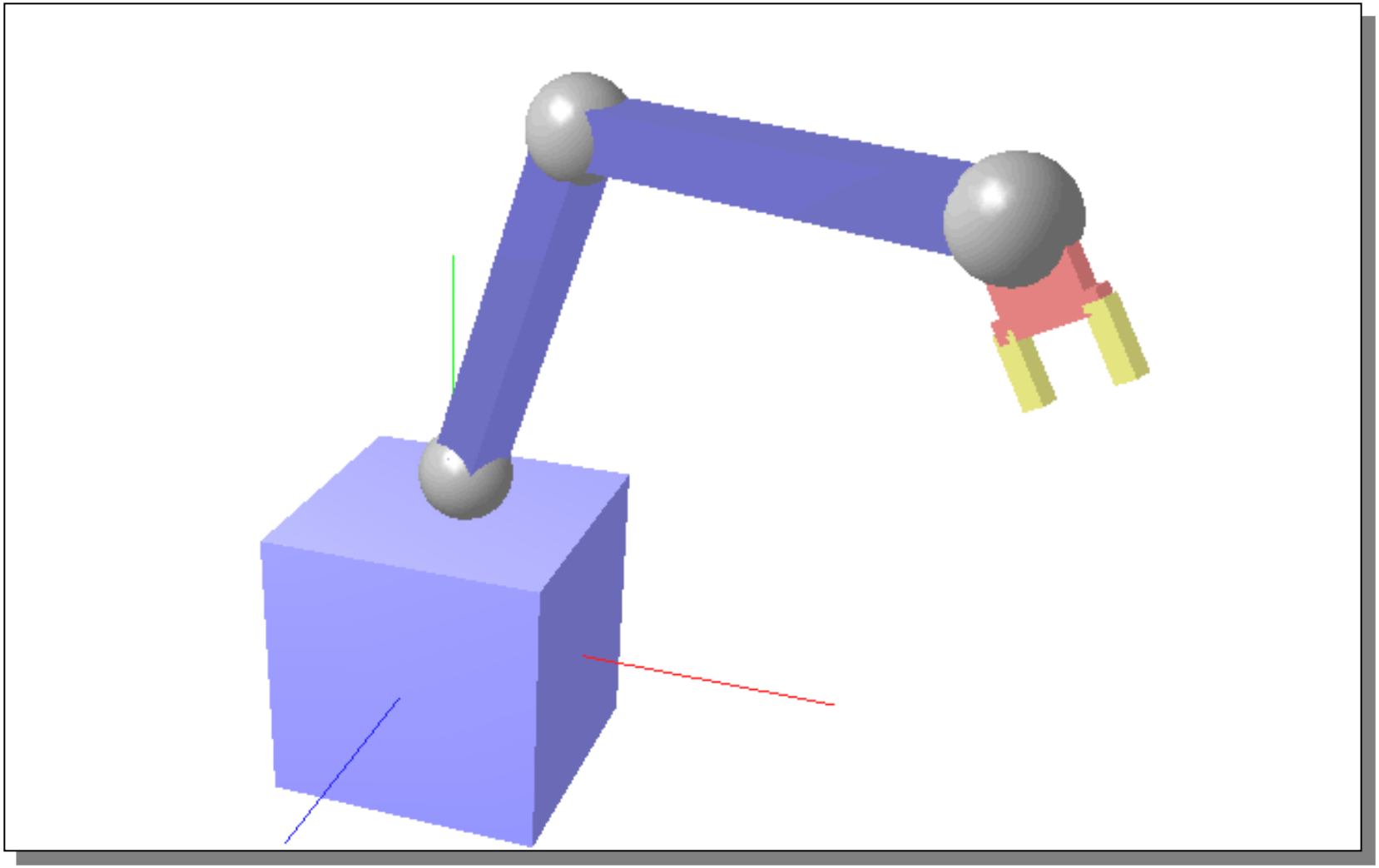


$\mathbf{p}_1 = \mathbf{T}\,\mathbf{p}$ e $\mathbf{p}_2 = \mathbf{R}\,\mathbf{p}_1 \Rightarrow \mathbf{p}_2 = \mathbf{R}\,\mathbf{T}\,\mathbf{p}$

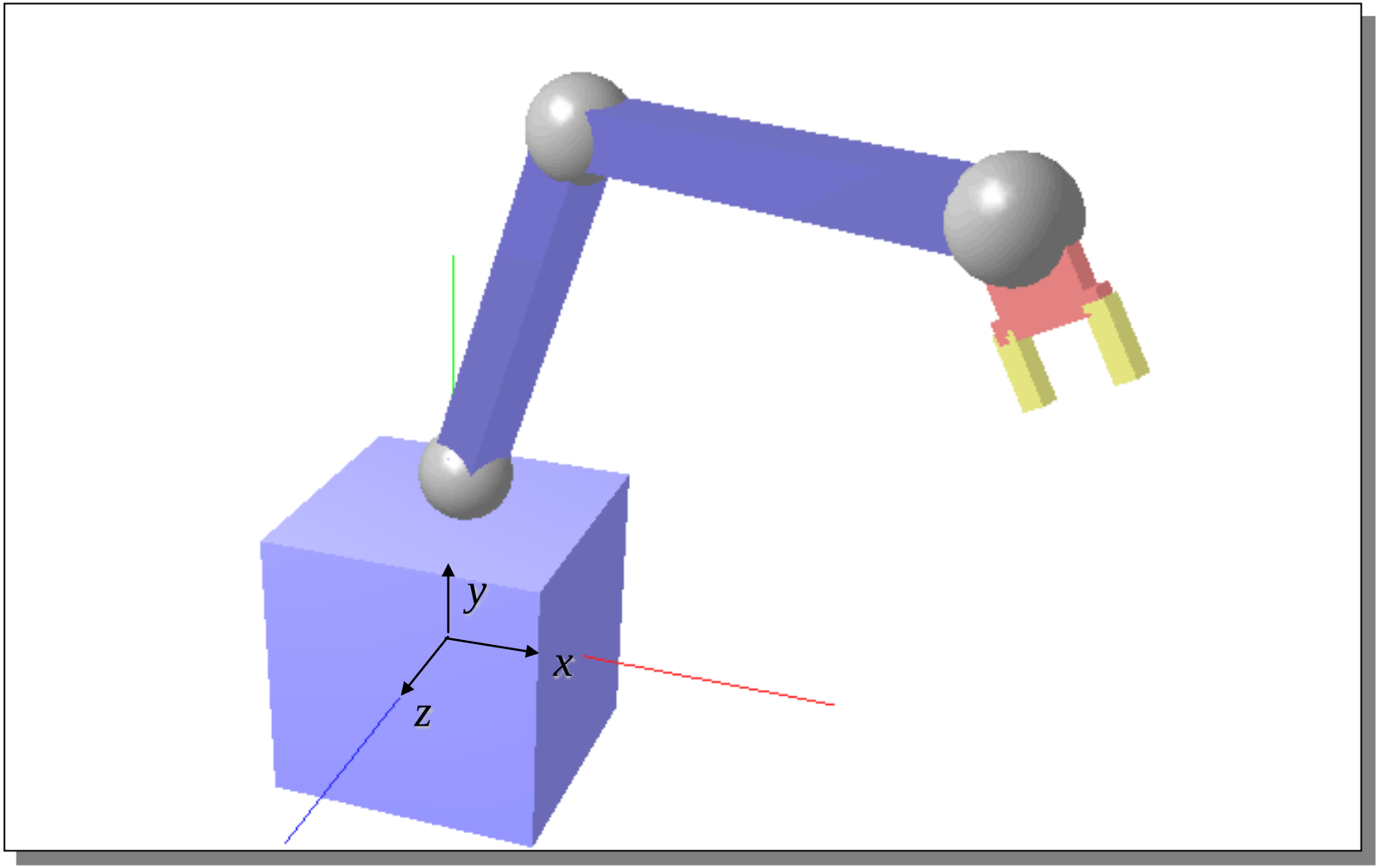


$\mathbf{p}_2 = \mathbf{T}_L\mathbf{R}\,\mathbf{p}$, $\mathbf{T}_L = \mathbf{R}\mathbf{T}\mathbf{R}^{-1} \Rightarrow \mathbf{p}_2 = \mathbf{R}\,\mathbf{T}\,\mathbf{R}^{-1}\,\mathbf{R}\,\mathbf{p}$ ou $\mathbf{p}_2 = \mathbf{R}\,\mathbf{T}\,\mathbf{p}$

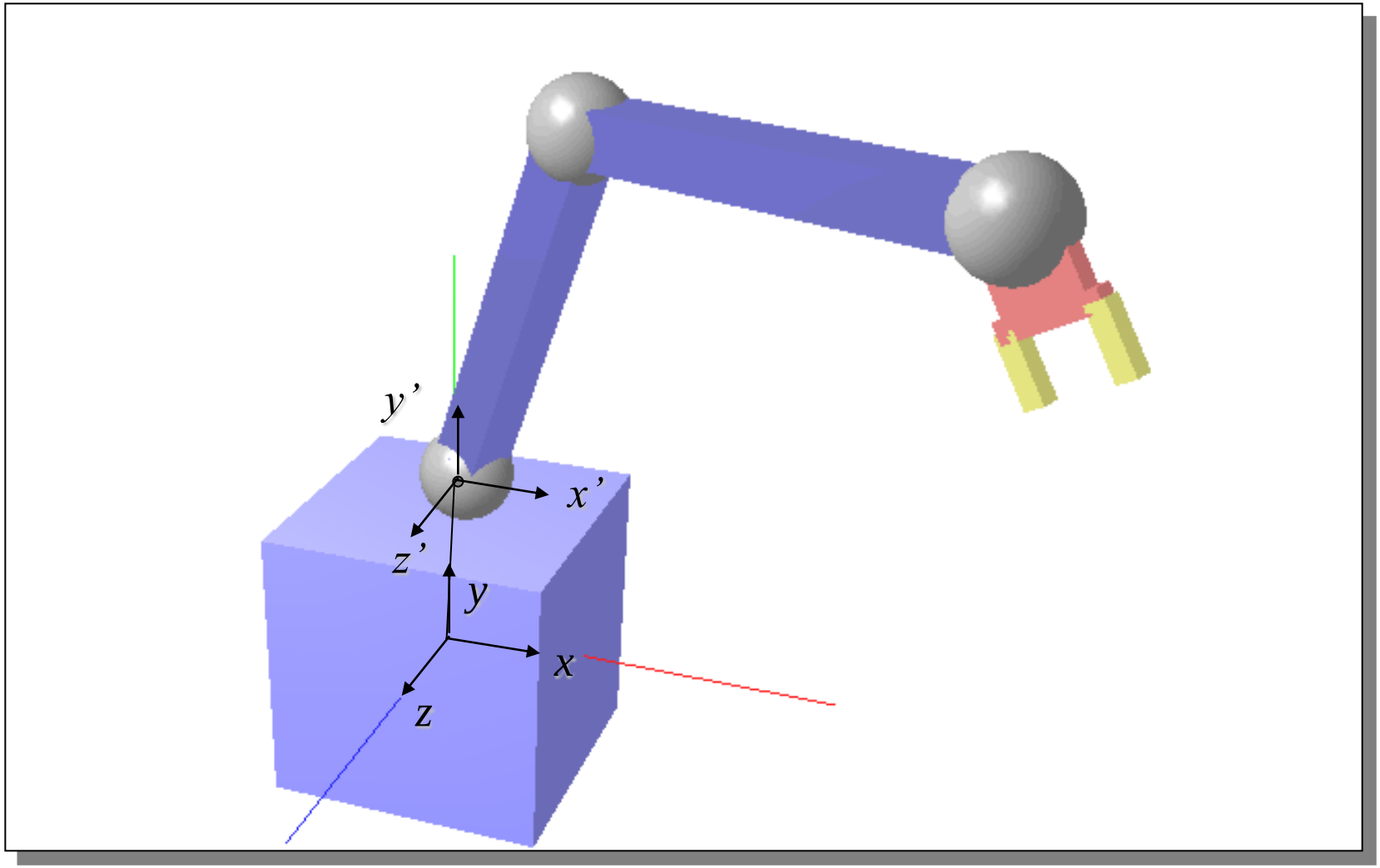
Transformações geométricas no espaço: Transformações em *OpenGL*



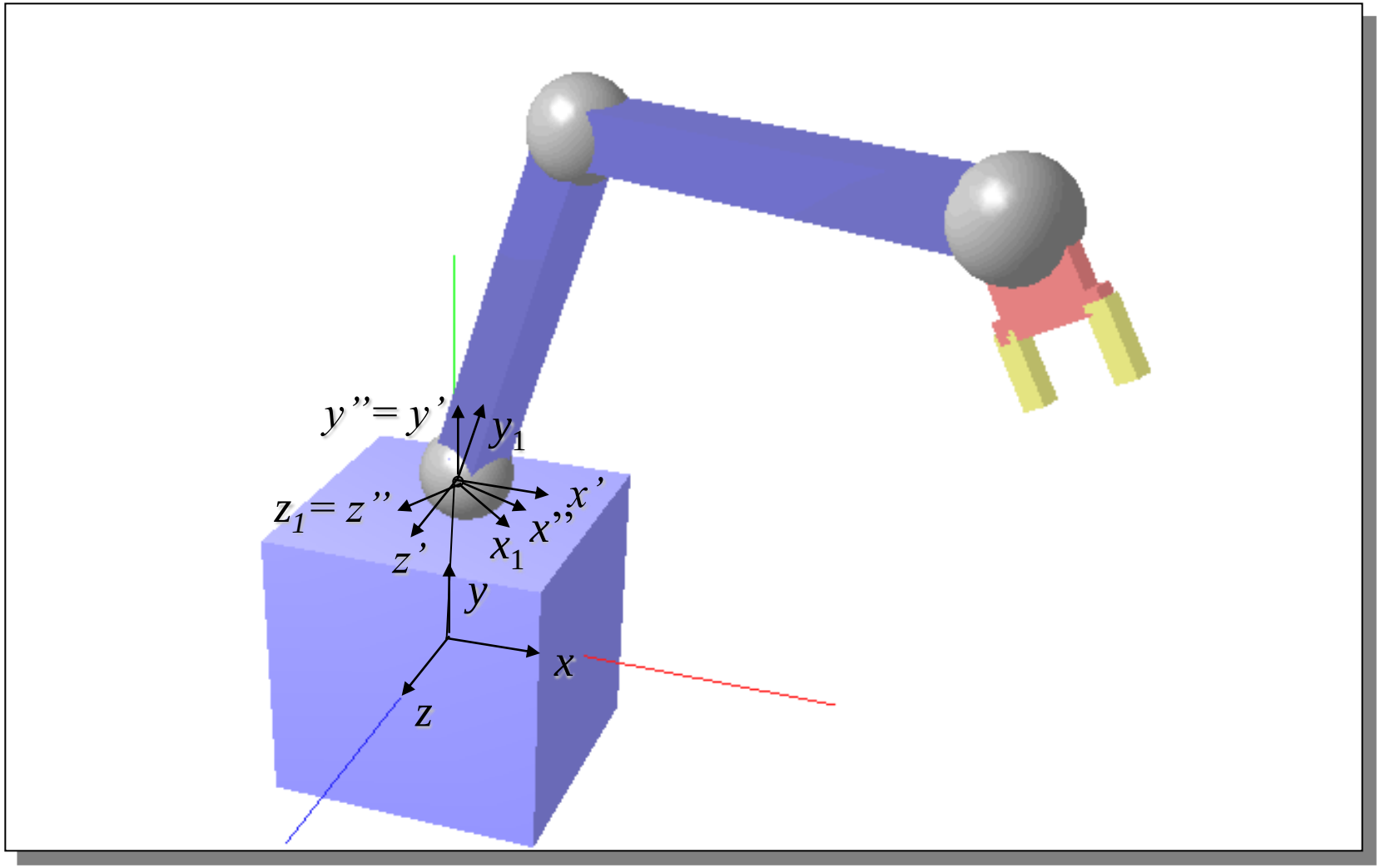
Transformações geométricas no espaço: Transformações em *OpenGL*



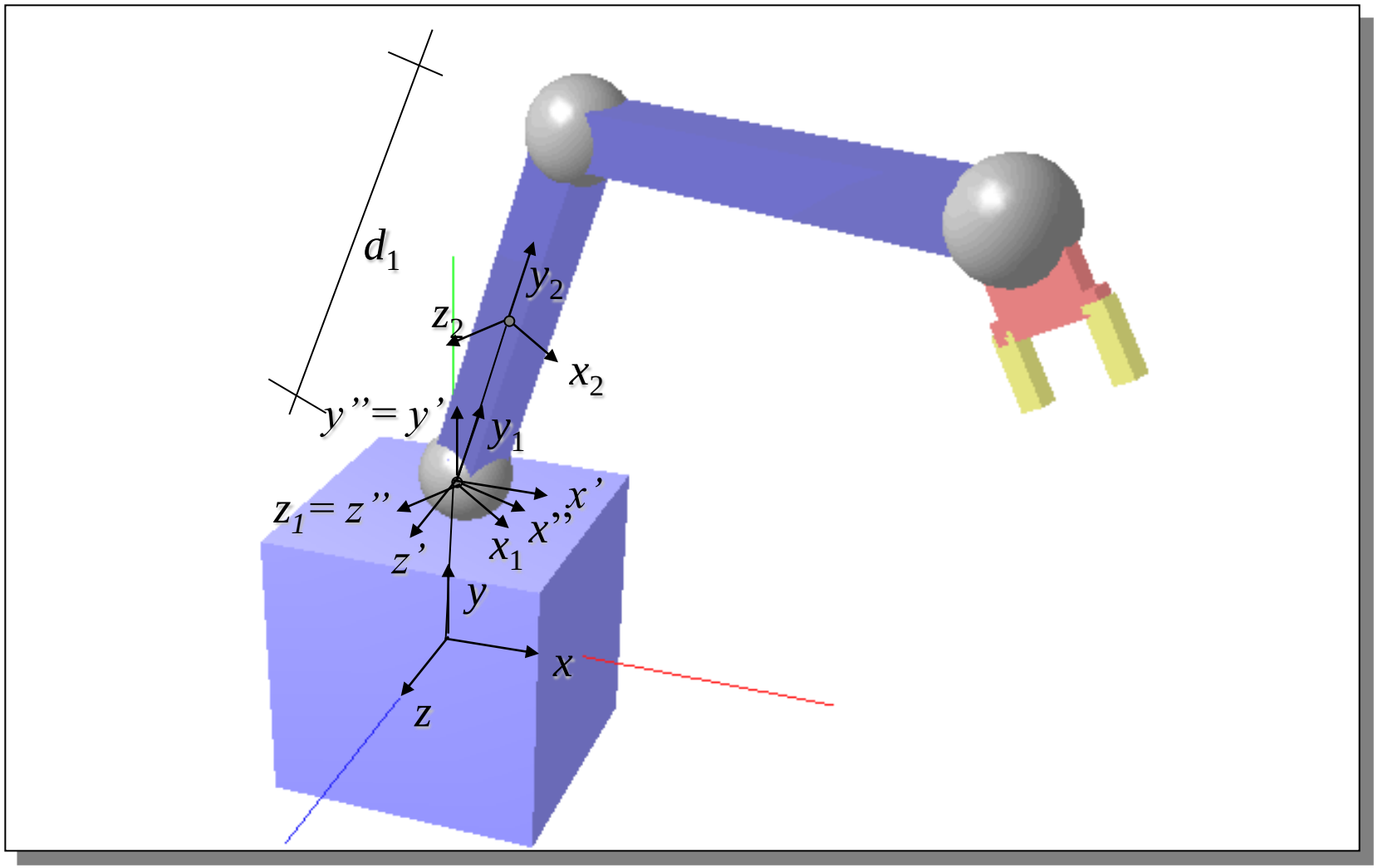
Transformações geométricas no espaço: Transformações em *OpenGL*



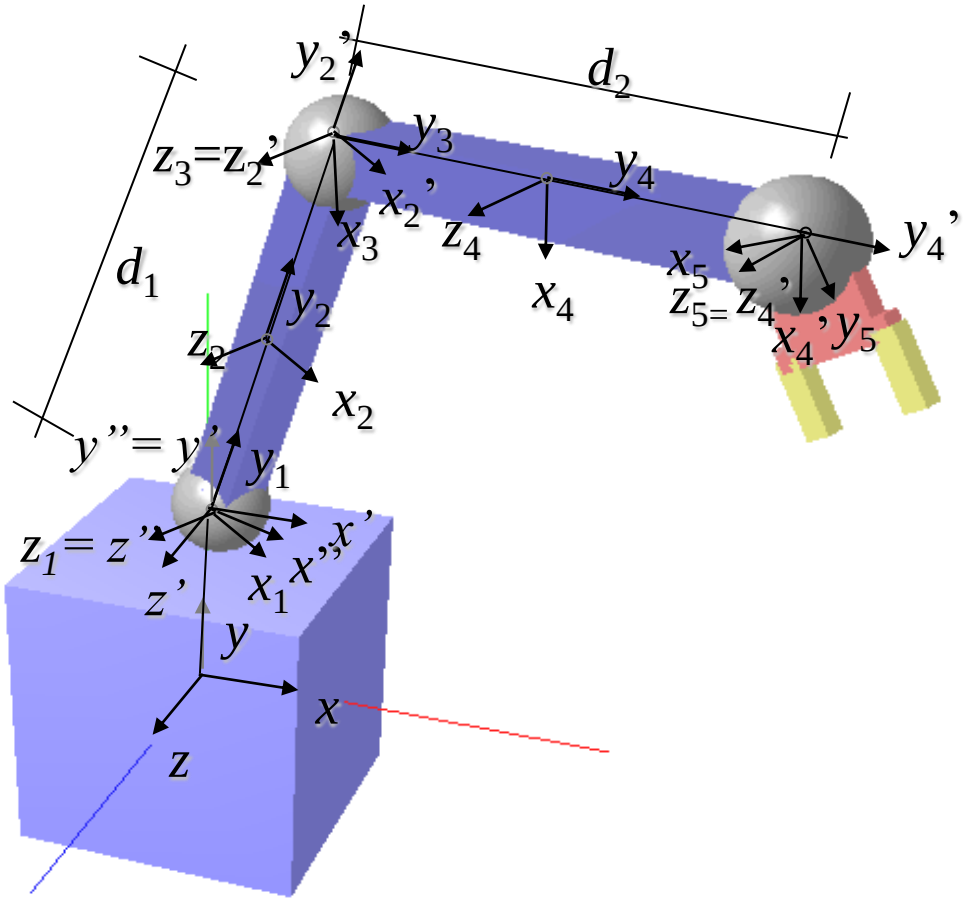
Transformações geométricas no espaço: Transformações em *OpenGL*



Transformações geométricas no espaço: Transformações em *OpenGL*



Transformações geométricas no espaço: Transformações em *OpenGL*

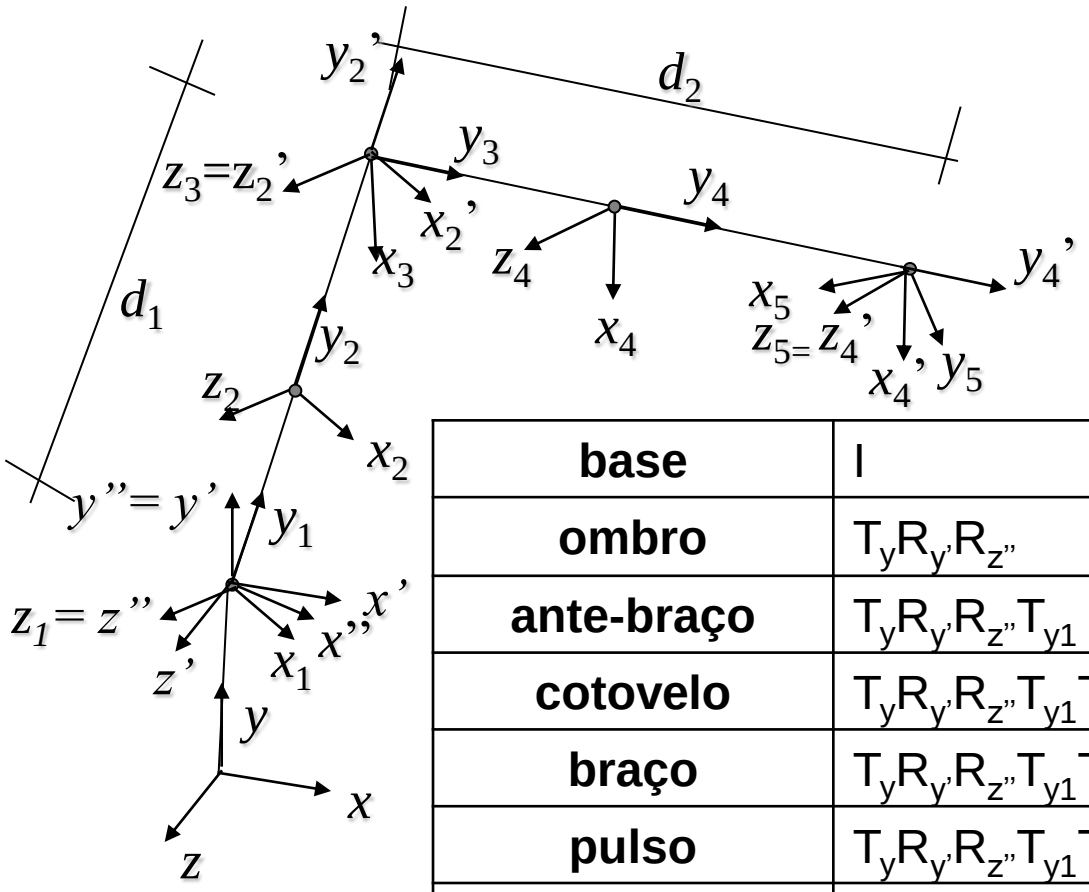



```
Desenha a base (em xyz);
Translada em y;
Roda em y';
Roda em z'';
Desenha o ombro (em x1y1z1)
Translada em y1 de d1/2;
Desenha o ante-braço (em
x2y2z2);
Translada em y2 de d1/2;
Roda em z2'
Desenha o cotovelo (em x3y3z3);
Translada em y3 de d2/2;
Desenha o braço (em x4y4z4);
Translada em y4 de d2/2;
Roda em z4';
Desenha o pulso (em x5y5z5);
Translada em y5
Desenha a mão (em x6y6z6);
```

Transformações geométricas no espaço: Transformações em *OpenGL*

```
glMatrixMode(GL_MODELVIEW);    /* transformações do modelo           */
glLoadIdentity( );             /* carrega a identidade como corrente */
desenhaBase( );                /* em xyz                             */
glTranslatef(0,d0,0.);         /* translada em y                     */
glRotatef(angy0, 0.,1.,0.);    /* roda em y'                         */
glRotatef(angz0, 0.,0.,1.);    /* roda em z''                        */
desenhaOmbro( );               /* em x1y1z1                          */
glTranslatef(0.,d1/2,0.);      /* translada em y1                    */
desenhaAnteBraco( );           /* em x2y2z2                          */
glTranslatef(0.,d1/2,0.);      /* translada em y2                    */
glRotatef(angz1, 0.,0.,1.);    /* roda em z2´                         */
desenhaCotovelo( );            /* em x3y3z3                          */
glTranslatef(0.,d2/2,0.);      /* translada em y3                    */
desenhaBraco( );               /* em x4y4z4                          */
glTranslatef(0.,d2/2,0.);      /* translada em y4                    */
glRotatef(rz5, 0.,0.,1.);      /* roda em z4'                        */
desenhaPulso( );               /* em x5y5z5                          */
glTranslatef(0.,d3,0.0);       /* translada em y5                    */
desenhaMao( );                 /* em x6y6z6                          */
```

Transformações geométricas no espaço: Transformações em *OpenGL*



base	I
ombro	$T_y R_{y'} R_{z''}$
ante-braço	$T_y R_{y'} R_{z''} T_{y1}$
cotovelo	$T_y R_{y'} R_{z''} T_{y1} T_{y2} R_{z2'}$
braço	$T_y R_{y'} R_{z''} T_{y1} T_{y2} R_{z2'} T_{y3}$
pulso	$T_y R_{y'} R_{z''} T_{y1} T_{y2} R_{z2'} T_{y3} T_{y4} R_{z4'}$
mão	$T_y R_{y'} R_{z''} T_{y1} T_{y2} R_{z2'} T_{y3} T_{y4} R_{z4'} T_{y5}$

Transformações geométricas no espaço: Transformações em *OpenGL*

- Sistemas como o OpenGL utilizam o conceito de *matriz corrente*.

- Desta forma, apenas uma matriz é responsável por realizar as transformações geométricas.

- Esta matriz é denominada *matriz de modelagem e visualização* (*model view matrix*).

Transformações geométricas no espaço: Transformações em *OpenGL*

- Quando fornecemos uma nova matriz M, ela é multiplicada pela esquerda pela matriz corrente C. Isto é $C_{nova} = CM$.

- Geometricamente isto significa que *a transformação descrita por M ocorrerá primeiro que a transformação dada por C*.

- Isto é bastante conveniente para o esquema de interpretação do processo de instanciação baseado em *eixos locais*.

Transformações geométricas no espaço: Transformações em *OpenGL*

- Existem situações em que o processo de instanciação não é descrito por uma seqüência de transformações com estrutura linear.

- É comum por exemplo, encontrarmos objetos que são descritos por *seqüências de transformação estruturadas em forma de árvore*.

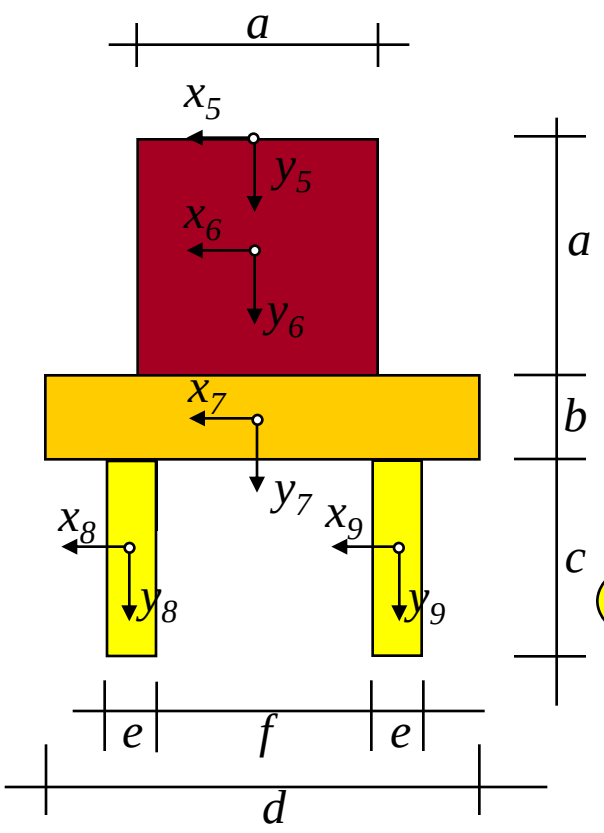
- Nestes casos é definida uma *hierarquia* sobre o conjunto de transformações e partes do objeto que especificam o modelo.

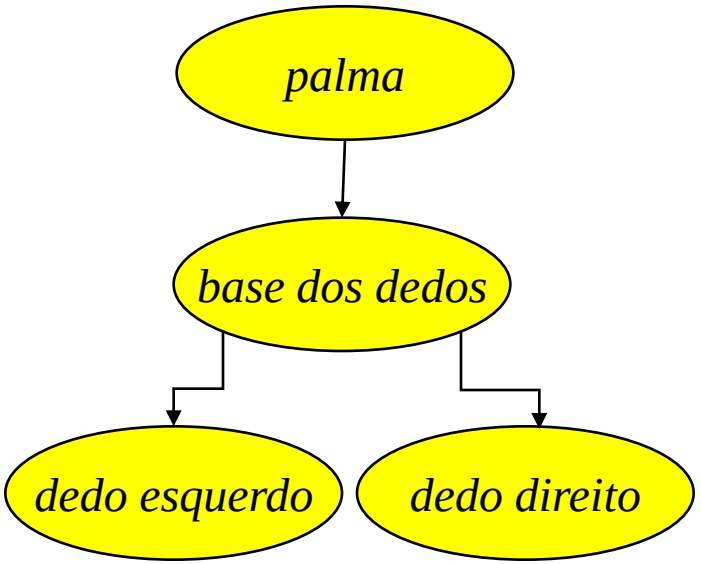
Transformações geométricas no espaço: Transformações em *OpenGL*

- Nestes casos, ao término do percorrimento de um dos ramos, é necessário *recuperar a matriz* do nó quando primeiro chegamos a ele.

- Por exemplo, podemos definir os dedos direito e esquerdo da mão de um robô a partir da base da mão.

Transformações geométricas no espaço: Transformações em *OpenGL*





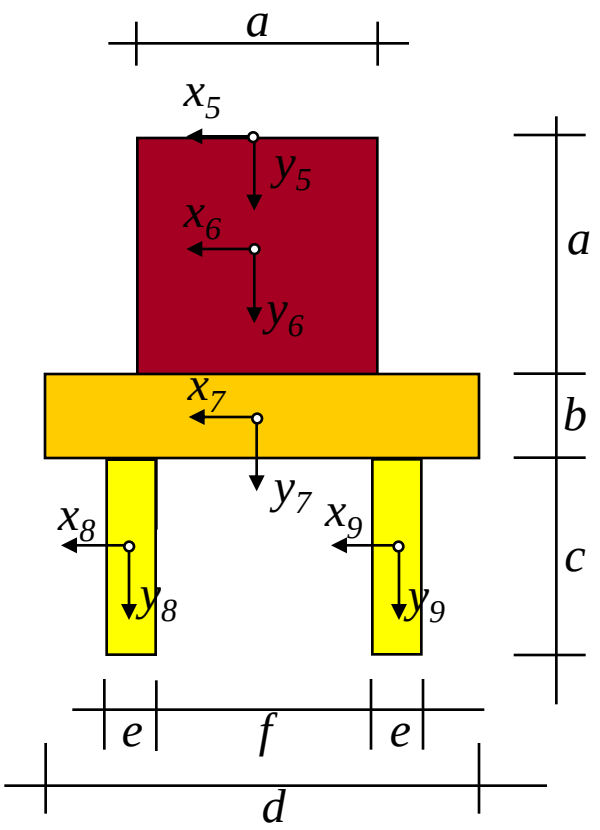
Transformações geométricas no espaço: Transformações em *OpenGL*

- Sistemas como o OpenGL implementam uma *estrutura de pilha para matrizes de transformação*.

- Desta forma, é possível percorrer a árvore saltando e recuperando as matrizes dos nós pai através de instruções *pop* e *push*.

- Com o mecanismo de pilha, podemos garantir que uma função retorna sem alterar o estado corrente das transformações.

Transformações geométricas no espaço: Transformações em *OpenGL*




```
void desenhaDedos(float b,float c, float f, float f )
{
  /* dedo esquerdo */

  glPushMatrix();      /* Salva matriz corrente C0 */
    glTranslatef((f+e)/2,(b+c)/2,0.);  /* C=CTesq  */
    glScalef(e,c,e);                   /* C=CS     */
    glutSolidCube(1.0);
  glPopMatrix();       /* Recupera da pilha C=C0   */

  /* dedo direito */

  glPushMatrix();      /* Salva matriz corrente C0 */
    glTranslatef((f+e)/2,(b+c)/2,0.);  /* C=CTdir  */
    glScalef(e,c,e);                   /* C=CS     */
    glutSolidCube(1.0);
  glPopMatrix();       /* Recupera da pilha C=C0   */

}
```

Transformações geométricas no espaço: Transformações em *OpenGL - exemplo*