# Transformações Geométricas 3D

SCC0250 - Computação Gráfica

Prof. Rosane Minghim
*rminghim@icmc.usp.br*
P.A.E. Nicolas Roque *nicolas.rsantos1@gmail.com*

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
baseado no material de anos anteriores, vários autores

5 de abril de 2017

# Sumário

# Sumário

# Introdução

- Métodos para transformações geométricas 3D são extensões de métodos 2D, incluindo a coordenada $z$

- A translação e a escala são simples adaptações, mas a rotação é mais complexa
  - Em 2D somente são consideradas rotações em torno de um eixo perpendicular ao plano $xy$, em 3D pode-se pegar qualquer orientação espacial para o eixo de rotação

- Uma posição 3D expressa em coordenadas homogêneas é representada usando vetores coluna de $4$ elementos, portanto as transformações 3D são matrizes $4 \times 4$

# Sumário

# Sumário

# Translação 3D

- Um objeto é movimentado adicionando-se *offsets* a cada uma das três direções Cartesianas

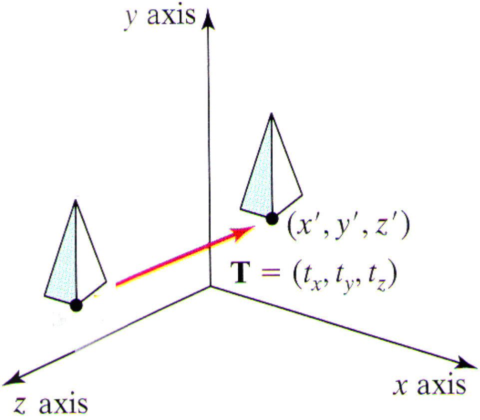
$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

- Representando matricialmente usando coordenadas homogêneas, temos

$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

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

# Translação 3D



# Translação 3D Inversa

- A translação inversa 3D é dada de forma semelhante a 2D, negando os *offsets* de translação

$$\mathbf{T}^{-1}(t_x, t_y, t_z) = \mathbf{T}(-t_x, -t_y, -t_z)$$

$$\mathbf{T}^{-1}(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Sumário

# Escala 3D

- A matriz de escala 3D é uma simples extensão da 2D, incluindo a variável $z$

- Considerando os fatores de escala $s_x > 0$, $s_y > 0$ e $s_z > 0$, temos

$$x' = x \cdot s_x$$

$$y' = y \cdot s_y$$

$$z' = z \cdot s_z$$

# Escala 3D

- Que definem a transformação

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

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

# Escala 3D

- Essa definição de escala muda a posição do objeto com relação a origem das coordenadas
  - Valores $> 1$ afastam da origem
  - Valores $< 1$ aproximam da origem

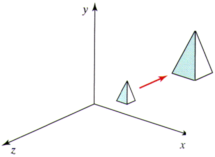



Figura: Dobrar o tamanho de um objeto também afasta o mesmo da origem

- Se $s_x = s_y = s_z$, então temos uma **escala uniforme**, caso contrário o objeto apresenta **escala diferencial**

# Escala 3D

- Para se evitar esse problema podemos definir a escala com relação a uma posição fixa $(x_f, y_f, z_f)$
  1. Translado o ponto fixo para a origem
  2. Aplico a transformação de escala
  3. Translado o ponto fixo de volta a sua posição original

$$\mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S}(s_x, s_y, s_z) \cdot \mathbf{T}(-x_f, -y_f, -z_f)$$

$$\begin{bmatrix} s_x & 0 & 0 & (1 - s_x)x_f \\ 0 & s_y & 0 & (1 - s_y)y_f \\ 0 & 0 & s_z & (1 - s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Escala Inversa 3D

- A matriz de escala inversa 3D é obtida trocando os fatores de escala por seus opostos

$$\mathbf{T}^{-1}(s_x, s_y, s_z) = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 & (1 - \frac{1}{s_x})x_f \\ 0 & \frac{1}{s_y} & 0 & (1 - \frac{1}{s_y})y_f \\ 0 & 0 & \frac{1}{s_z} & (1 - \frac{1}{s_z})z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

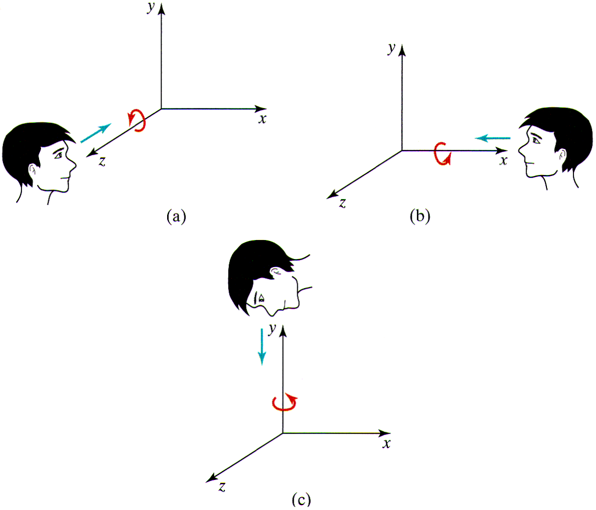
# Sumário

# Rotação 3D

- É possível rodar um objeto ao redor de qualquer eixo no espaço 3D, porém, as rotações mais fáceis são executadas ao redor dos eixos de coordenadas Cartesianas
  - É possível combinar rotações em tornos dos eixos Cartesianos para se obter rotações em torno de qualquer eixo no espaço

- Por convenção, ângulos positivos produzem rotações no sentido anti-horário

# Rotação 3D



# Rotação 3D nos Eixos Coordenados

- Uma rotação 2D é facilmente extendida para uma rotação 3D ao redor do eixo $z$

$$x' = x\cos\theta - y\,\mathrm{sen}\,\theta$$

$$y' = x\,\mathrm{sen}\,\theta + y\cos\theta$$

$$z' = z$$

- Na forma matricial usando coordenadas homogêneas

$$\mathbf{P}' = \mathbf{R_z}(\theta)\cdot\mathbf{P}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\mathrm{sen}\,\theta & 0 & 0 \\ \mathrm{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Rotação 3D nos Eixos Coordenados

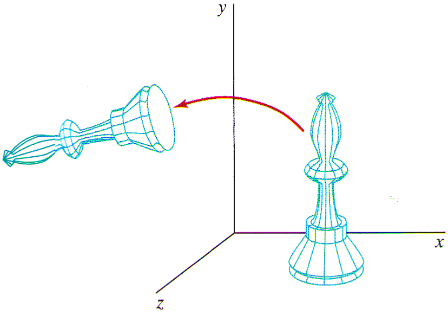
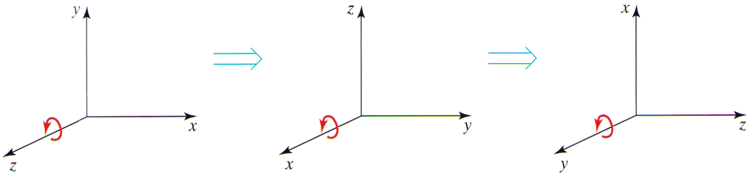



Figura: Rotação de um objeto em torno do eixo-$z$

# Rotação 3D nos Eixos Coordenados

- As transformação de rotação para os outros eixos de coordenadas podem ser obtidas por meio de uma permutação cíclica das coordenadas $x$, $y$ e $z$

$$x \to y \to z \to x$$



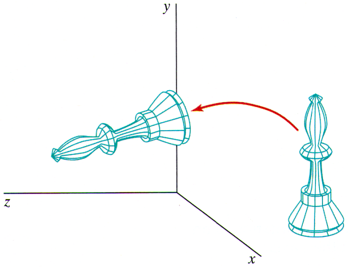
# Rotação 3D nos Eixos Coordenados

- Considerando essa permutação e substituindo na equação da rotação 3D, compomos a rotação em torno do eixo-$x$

$$y' = y \cos\theta - z \operatorname{sen}\theta$$

$$z' = y \operatorname{sen}\theta + z \cos\theta$$

$$x' = x$$



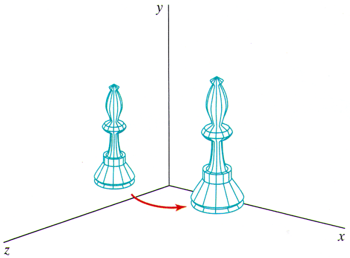
# Rotação 3D nos Eixos Coordenados

- O mesmo ocorrendo para se obter as equações para rotação em torno do eixo-$y$

$$z' = z \cos\theta - x \operatorname{sen}\theta$$

$$x' = z \operatorname{sen}\theta + x \cos\theta$$

$$y' = y$$



# Rotação 3D nos Eixos Coordenados

- Portanto as matrizes de rotação em torno dos eixos $x$ e $y$ são, respectivamente

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\operatorname{sen}\theta & 0 \\ 0 & \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \operatorname{sen}\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Rotação 3D Inversa

- A inversa de uma rotação é obtida trocando $\theta$ por $-\theta$

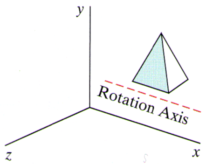
- Como somente o sinal do seno é alterado, a inversa pode ser obtida trocando as linhas pelas colunas, isto é $\mathbf{R}^{-1} = \mathbf{R}^T$

# Rotação 3D Geral

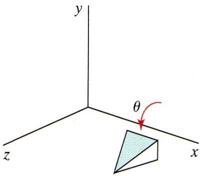
- A rotação em torno de qualquer eixo pode ser obtida como a combinação de rotações e translações

- No caso especial quando o eixo de rotação é paralelo a algum eixo de coordenadas, obtemos a rotação desejada fazendo
  1. Translado o objeto de forma que o eixo de rotação coincida com o eixo paralelo de coordenadas
  2. Executo a rotação
  3. Translado o objeto de forma que o eixo de rotação é movido de volta a posição original

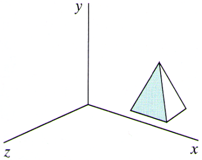
# Rotação 3D Geral



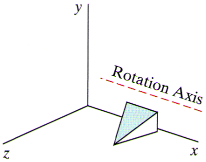

(a)
Original Position of Object




(c)
Rotate Object Through Angle $\theta$




(b)
Translate Rotation Axis onto $x$ Axis




(d)
Translate Rotation
Axis to Original Position

# Rotação 3D Geral

- Essa sequencia de transformação sobre um ponto $P$ é

$$\mathbf{P}' = \mathbf{T}^{-1} \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T} \cdot \mathbf{P}$$

- Ou seja, a matriz composta de rotação é

$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T}$$

- Que é a mesma forma da matriz de rotação 2D quando o eixo de rotação (ortogonal ao plano $xy$) não coincide com a origem

# Rotação 3D Geral

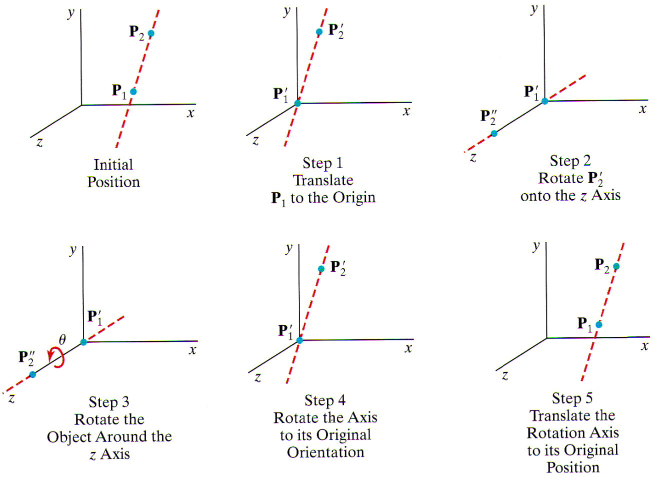
- Quando o eixo de rotação não é paralelo aos eixos de coordenadas, algumas transformações adicionais são necessárias
  - Também são necessárias rotações para alinhar o eixo de rotação com o eixo de coordenadas escolhido e para trazer de volta o eixo de rotação para a posição original

# Rotação 3D Geral

- Dado o eixo de rotação e o ângulo de rotação, isso pode ser feito como
  1. Transladar o objeto de forma que o eixo de rotação passe pela origem do sistema de coordenadas
  2. Rotacionar o objeto para que o eixo de rotação coincida com um dos eixos de coordenadas
  3. Realizar a rotação sobre o eixo de coordenadas escolhido
  4. Aplicar a rotação inversa para trazer o eixo de rotação para sua orientação original
  5. Aplicar a translação inversa para trazer o eixo de rotação para sua posição espacial original

- Por conveniência, o eixo de coordenadas escolhido para o alinhamento normalmente é o eixo-$z$

# Rotação 3D Geral



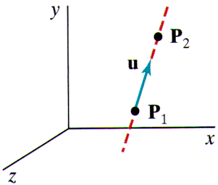
# Rotação 3D Geral

- Assumindo que o eixo de rotação é definido por dois pontos ($\mathbf{P_2}$ para $\mathbf{P_1}$) e que a rotação se dá em sentido anti-horário em relação a esse eixo, podemos calcular suas componentes como

$$\mathbf{V} = \mathbf{P_2} - \mathbf{P_1} = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

- E o vetor unitário do eixo de rotação é

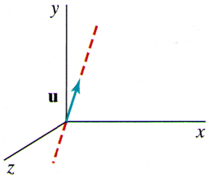
$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a, b, c)$$



# Rotação 3D Geral

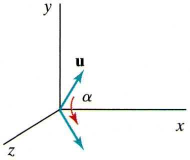
- O primeiro passo da sequencia de rotação é definir uma matriz de translação para reposicionar o eixo de rotação de forma que esse passe pela origem
  - Como a rotação se dá no sentido anti-horário, movemos o ponto $\mathbf{P_1}$ para a origem, ou seja

$$\begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

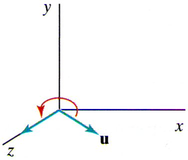


# Rotação 3D Geral

- Após isso, encontramos a transformação que coloca o eixo de rotação sobre o eixo $z$
  - Existem várias maneiras de se realizar esse alinhamento, por exemplo, primeiro rotacionamos sobre o eixo $x$, depois sobre o eixo $y$
  - A rotação sobre o eixo $x$ define o vetor $\mathbf{u}$ no plano $xz$, e a rotação no eixo $y$ rotaciona $\mathbf{u}$ até sobrepor o eixo $z$



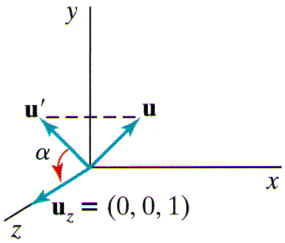

(a)




(b)

# Rotação 3D Geral

- A rotação em torno do eixo $x$ pode ser definida determinando os senos e cossenos do ângulo de rotação necessário para projetar $\mathbf{u}$ no plano $xz$
- Esse ângulo de rotação ($\alpha$) é o ângulo entre a projeção de $\mathbf{u}$ no plano $yz$ com o eixo $z$ positivo



# Rotação 3D Geral

- Se a projeção de $\mathbf{u}$ no plano $yz$ for $\mathbf{u}' = (0, b, c)$, então o cosseno do ângulo de rotação $\alpha$ pode ser determinado a partir do produto escalar de $\mathbf{u}'$ com o vetor unitário $\mathbf{u_z}$ ao longo do eixo $z$

$$\cos\alpha = \frac{\mathbf{u}' \cdot \mathbf{u_z}}{|\mathbf{u}'||\mathbf{u_z}|} = \frac{c}{d}$$

- Onde $d$ é a magnitude de $\mathbf{u}'$, isto é

$$d = \sqrt{b^2 + c^2}$$

# Rotação 3D Geral

- Similarmente é possível determinar o seno de $\alpha$ igualando a forma independente de coordenadas do produto vetorial

$$\mathbf{u}' \times \mathbf{u_z} = \mathbf{u_x} |\mathbf{u}'||\mathbf{u_z}| \operatorname{sen} \alpha$$

- Com a sua forma Cartesiana

$$\mathbf{u}' \times \mathbf{u_z} = \mathbf{u_x} \cdot b$$

$$\mathbf{u}' \times \mathbf{u_z} = \mathbf{u_x} |\mathbf{u}'||\mathbf{u_z}| \operatorname{sen} \alpha = \mathbf{u_x} \cdot b$$

- Como $|\mathbf{u_z}| = 1$ e $|\mathbf{u}'| = d$, então

$$\operatorname{sen} \alpha = \frac{b}{d}$$

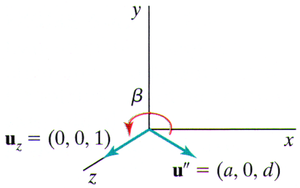
# Rotação 3D Geral

- Com os senos de cossenos de $\alpha$ determinados, podemos definir a matriz para a rotação $\mathbf{u}$ sobre o eixo $x$ no plano $xz$

$$\mathbf{R_x}(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Rotação 3D Geral

- O próximo passo é determinar a matriz de rotação que vai rotacionar o vetor unitário $\mathbf{u}''$ (resultante da rotação anterior) no plano $xz$ em torno do eixo $y$ até sobrepor o eixo $z$
  - Como $\mathbf{u} = (a, b, c)$, então $\mathbf{u}'' = (a, 0, d)$ pois a rotação em torno do eixo $x$ não altera a coordenada $x$, a coordenada $y$ é zerada pela projeção no plano $xz$ e a coordenada $z = d$ porque $|\mathbf{u}''| = |\mathbf{u}|$



# Rotação 3D Geral

- Com isso podemos novamente encontrar os senos e cossenos do ângulo $\beta$ fazendo

$$\cos\beta = \frac{\mathbf{u''} \cdot \mathbf{u_z}}{|\mathbf{u''}||\mathbf{u_z}|}$$

- Como $|\mathbf{u_z}| = |\mathbf{u''}| = 1$

$$\cos\beta = d$$

# Rotação 3D Geral

- Igualando a forma independente de coordenadas do produto vetorial

$$\mathbf{u}'' \times \mathbf{u_z} = \mathbf{u_y} |\mathbf{u}''||\mathbf{u_z}| \operatorname{sen} \beta$$

- Com a forma Cartesiana

$$\mathbf{u}'' \times \mathbf{u_z} = \mathbf{u_y} \cdot (-a)$$

$$\mathbf{u}'' \times \mathbf{u_z} = \mathbf{u_y} |\mathbf{u}''||\mathbf{u_z}| \operatorname{sen} \beta = \mathbf{u_y} \cdot (-a)$$

- Temos

$$\operatorname{sen} \beta = -a$$

# Rotação 3D Geral

- Portanto, a matriz de rotação de $\mathbf{u}''$ sobre o eixo $y$ é

$$\mathbf{R_y}(\beta) = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Rotação 3D Geral

- Com essas rotação em $\alpha$ e $\beta$ nós alinhamos o eixo de rotação sobre o eixo $z$, então agora a rotação de um ângulo $\theta$ pode ser aplicada

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

# Rotação 3D Geral

- Assim, a matriz de rotação completa sobre um eixo arbitrário fica

$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R_x}^{-1}(\alpha) \cdot \mathbf{R_y}^{-1}(\beta) \cdot \mathbf{R_z}(\theta) \cdot \mathbf{R_y}(\beta) \cdot \mathbf{R_x}(\alpha) \cdot \mathbf{T}$$

# Rotação 3D Geral

- Uma forma menos intuitiva de obter a matriz de rotação composta $\mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)$ é lembrando que a matriz para qualquer sequencia de rotações 3D é da forma

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Onde a matriz $3 \times 3$ superior é ortonormal

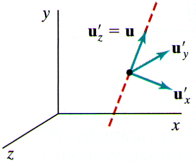
# Rotação 3D Geral

- Portanto podemos definir um sistema de coordenadas locais com um eixo alinhado ao eixo de rotação, e os vetores unitários para os três eixos de coordenadas são usados para construir a matriz de rotação
- Assumindo que o eixo de rotação não é paralelo a qualquer eixo de coordenadas, esse vetores poderiam ser calculados como

$$\mathbf{u}'_\mathbf{z} = \mathbf{u} = (u'_{z1}, u'_{z2}, u'_{z3})$$

$$\mathbf{u}'_\mathbf{y} = \frac{\mathbf{u} \times \mathbf{u_x}}{|\mathbf{u} \times \mathbf{u_x}|} = (u'_{y1}, u'_{y2}, u'_{y3})$$

$$\mathbf{u}'_\mathbf{x} = \mathbf{u}'_\mathbf{y} \times \mathbf{u}'_\mathbf{z} = (u'_{x1}, u'_{x2}, u'_{x3})$$



# Rotação 3D Geral

- Então a matriz buscada $\mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)$ fica

$$\mathbf{R} = \begin{bmatrix} u'_{x1} & u'_{x2} & u'_{x3} & 0 \\ u'_{y1} & u'_{y2} & u'_{y3} & 0 \\ u'_{z1} & u'_{z2} & u'_{z3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Que transforma os vetores unitários $\mathbf{u'_x}$, $\mathbf{u'_y}$ e $\mathbf{u'_z}$ nos eixos $x$, $y$ e $z$, alinhando o eixo de rotação com o eixo $z$, porque $\mathbf{u'_z} = \mathbf{u}$

# Sumário

# Compondo Transformações 3D

- Assim como nas transformações 2D, as transformações 3D são compostas multiplicando matrizes

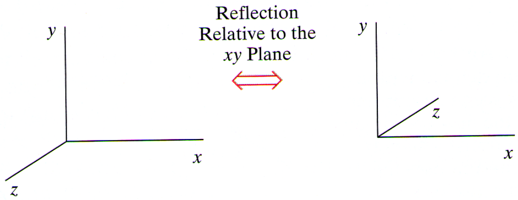
- Novamente a tranformações mais a direita será a primeira a ser aplicada, e é necessário observar se a API gráfica utilizada é pós- ou pré-multiplicada

# Sumário

# Sumário

# Reflexão 3D

- É semelhante a reflexão 2D: rotação de $180^0$ sobre um eixo (plano) de rotação
- Quando o plano de rotação é um plano coordenado ($xy$, $xz$ ou $yz$), essa transformação pode ser vista como uma conversão entre um sistema orientado com a mão-esquerda e um orientado com a mão-direita (ou vice-versa)



# Reflexão 3D

- Essa conversão entre um sistema orientado pela mão-direita, para um orientado pela mão-esquerda é obtido trocando o sinal da coordenada $z$, mantendo as coordenadas $x$ e $y$ (reflexão relativa ao plano $xy$)

$$\mathbf{M_{z_{reflect}}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- As reflexões relativas ao planos $yz$ e $xz$ são obtidas de forma semelhante

# Sumário

# Cisalhamento 3D

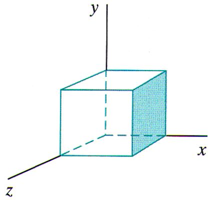
- Cisalhamento relativo aos eixos $x$ e $y$ é o mesmo que o já discutido em 2D, mas em 3D também é possível realizar o cisalhamento relativo ao eixo $z$
- O cisalhamento geral em torno do eixo-$z$, dado um ponto de referência $z_{ref}$ é produzido pela seguinte matriz

$$\mathbf{M_{z_{shear}}} = \begin{bmatrix} 1 & 0 & sh_{zx} & -sh_{zx} \cdot z_{ref} \\ 0 & 1 & sh_{zy} & -sh_{zy} \cdot z_{ref} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

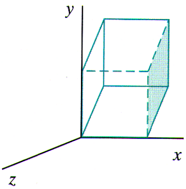
- O efeito de $sh_{zx}$ e $sh_{zy}$ é alterar os valores das coordenadas $x$ e $y$ uma quantidade proporcional a distância de $z_{ref}$, enquanto mantém a coordenada $z$ inalterada

# Cisalhamento 3D

- Exemplo de matriz de cisalhamento com $sh_{zx} = sh_{zy} = 1$ e $z_{ref} = 0$ aplicada sobre um cubo unitário




(a)




(b)

# Sumário

# Transformações Afim

- Uma transformação afim é dada pela forma

$$\begin{aligned} x' &= a_{xx}x + a_{xy}y + a_{xz}z + b_x \\ y' &= a_{yx}x + a_{yy}y + a_{yz}z + b_y \\ z' &= a_{zx}x + a_{zy}y + a_{zz}z + b_z \end{aligned}$$

- $x'$, $y'$ e $z'$ são transformações lineares das coordenadas originais $x$, $y$ e $z$

- Uma propriedade geral é que linhas paralelas são transformadas em linhas paralelas e pontos finitos são transformados em pontos finitos
- Translação, rotação, escala, reflexão e cisalhamento, ou suas combinações, são transformações afins

# Sumário

# Programação OpenGL

```
public class Renderer extends KeyAdapter implements GLEventListener{

    public Renderer() {
        this.alpha = 0;
        this.beta = 0;
        this.delta = 1;
    }

    public void init(GLAutoDrawable drawable) {
        GL gl = drawable.getGL();
        gl.glClearColor(0, 0, 0, 0); //define a cor de fundo
        gl.glEnable(GL.GL_DEPTH_TEST); //remoção de superfície oculta

        gl.glMatrixMode(GL.GL_PROJECTION); //define que a matrix é a de projeção
        gl.glLoadIdentity(); //carrega a matrix de identidade
        gl.glOrtho(-5, 5, -5, 5, -5, 5); //define uma projeção ortográfica
    }

    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int ↩
        height) {
    }

    public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, ↩
        boolean deviceChanged) {
    }

    ...

    private float alpha;
    private float beta;
    private float delta;
}
```

# Programação OpenGL

```
public class Renderer extends KeyAdapter implements GLEventListener {
    ...

    public void display(GLAutoDrawable drawable) {
        GL gl = drawable.getGL();
        GLUT glut = new GLUT();

        //limpa o buffer
        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);

        //define que a matrix é a de modelo
        gl.glMatrixMode(GL.GL_MODELVIEW);
        gl.glLoadIdentity(); //carrega matrix identidade

        //rotaciona e escala uma esfera 'aramado'
        gl.glRotatef(beta, 0, 1, 0);
        gl.glRotatef(alpha, 1, 0, 0);
        gl.glScalef(delta, delta, delta);
        gl.glColor3f(1, 1, 0);
        glut.glutWireSphere(1.0f, 20, 20);

        //desenha um 'piso' sob a esfera
        gl.glTranslatef(0, -1, 0);
        gl.glScalef(4, 0.1f, 4);
        gl.glColor3f(0, 0, 1);
        glut.glutSolidCube(1.0f);

        //força o desenho das primitivas
        gl.glFlush();
    }

    private float alpha;
    private float beta;
    private float delta;
}
```

# Programação OpenGL

```java
public class Renderer extends KeyAdapter implements GLEventListener {
    ...

    @Override
    public void keyPressed(KeyEvent e) {

        switch (e.getKeyCode()) {
            case KeyEvent.VK_PAGE_UP://faz zoom-in
                delta = delta * 1.1f;
                break;
            case KeyEvent.VK_PAGE_DOWN://faz zoom-out
                delta = delta * 0.809f;
                break;
            case KeyEvent.VK_UP://gira sobre o eixo-x
                alpha = alpha - 1;
                break;
            case KeyEvent.VK_DOWN://gira sobre o eixo-x
                alpha = alpha + 1;
                break;
            case KeyEvent.VK_LEFT://gira sobre o eixo-y
                beta = beta + 1;
                break;
            case KeyEvent.VK_RIGHT://gira sobre o eixo-y
                beta = beta - 1;
                break;
        }
    }
}
```

# Programação OpenGL

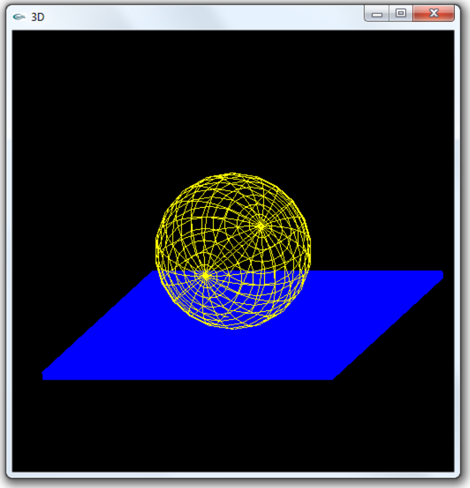
```
public static void main(String[] args) {
  //acelera o rendering
  GLCapabilities caps = new GLCapabilities();
  caps.setDoubleBuffered(true);
  caps.setHardwareAccelerated(true);

  //cria o painel e adiciona um ouvinte GLEventListener
  Renderer r = new Renderer();
  GLCanvas canvas = new GLCanvas(caps);
  canvas.addGLEventListener(r);

  //cria uma janela e adiciona o painel
  JFrame frame = new JFrame("Aplicação JOGL Simples");
  frame.addKeyListener(r);
  frame.getContentPane().add(canvas);
  frame.setSize(400, 400);
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  //inicializa o sistema e chama display() a 60 fps
  Animator animator = new FPSAnimator(canvas, 60);
  frame.setLocationRelativeTo(null);
  frame.setVisible(true);
  animator.start();
}
```

# Programação OpenGL



# Programação OpenGL

- Armazenando e restaurando transformações

```
public void display(GLAutoDrawable drawable) {
  GL gl = drawable.getGL();
  GLUT glut = new GLUT();

  //limpa o buffer
  gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);

  //define que a matriz é a de modelo
  gl.glMatrixMode(GL.GL_MODELVIEW);
  gl.glLoadIdentity();

  gl.glScalef(delta, delta, delta); //faça a escala de todos objetos

  gl.glPushMatrix(); //armazena a matriz corrente
    gl.glTranslatef(-3, 0, 0);
    gl.glRotatef(beta, 0, 1, 0);
    gl.glRotatef(alpha, 1, 0, 0);
    gl.glColor3f(1, 1, 0);
    glut.glutWireSphere(1, 20, 20);
  gl.glPopMatrix(); //restaura a matriz anterior

  gl.glPushMatrix(); //armazena a matriz corrente
    gl.glTranslatef(3, 0, 0);
    gl.glRotatef(beta, 0, 1, 0);
    gl.glRotatef(alpha, 1, 0, 0);
    gl.glColor3f(1, 0, 0);
    glut.glutWireSphere(1, 20, 20);
  gl.glPopMatrix(); //restaura a matriz anterior

  //força o desenho das primitivas
  gl.glFlush();
}
```

# Programação OpenGL