Matemática e Física para Jogos
Unidade 01
Conceitos Básicos

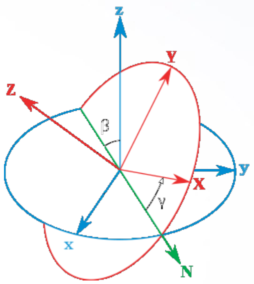
Matrizes de Transformação

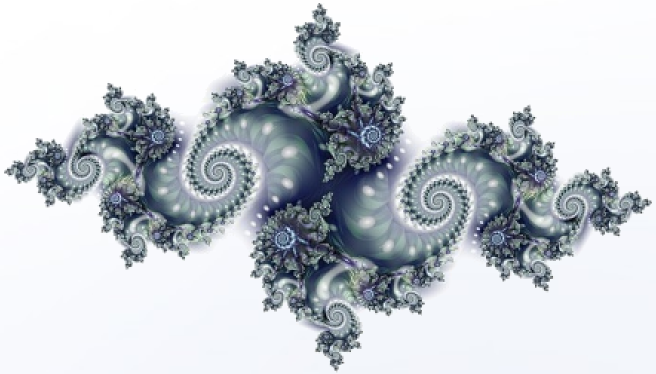
Gilvan Maia
gilvanmaia@virtual.ufc.br
Professor Adjunto

Instituto UFC Virtual
Universidade Federal do Ceará

Unidade 01
Conceitos Básicos

- Representação numérica no computador
- Funções
- **Vetores**
- **Matrizes**
- Sistemas de Coordenadas
- Quatérnios





Coordenadas Homogêneas

- Os vetores vistos são, no máximo, 3D
- As arquiteturas de hardware gráfico, assim como as APIs mais difundidas, usam vetores 4D
  - Introdução da coordenada W
- Essa coordenada especial permite expressar pontos e direções com a mesma representação
  - Quando $w = 1$, temos um **vetor posição**
  - Quando $\boldsymbol{w = 0}$, o vetor é uma **direção**
    - Significado especial na formulação dessa coordenada

Caso 2D: inserindo *w*

- Suponha um vetor em 2D

$$v^t = [x \quad y]$$

- Esse vetor pode ser reescrito usando coordenadas homogêneas

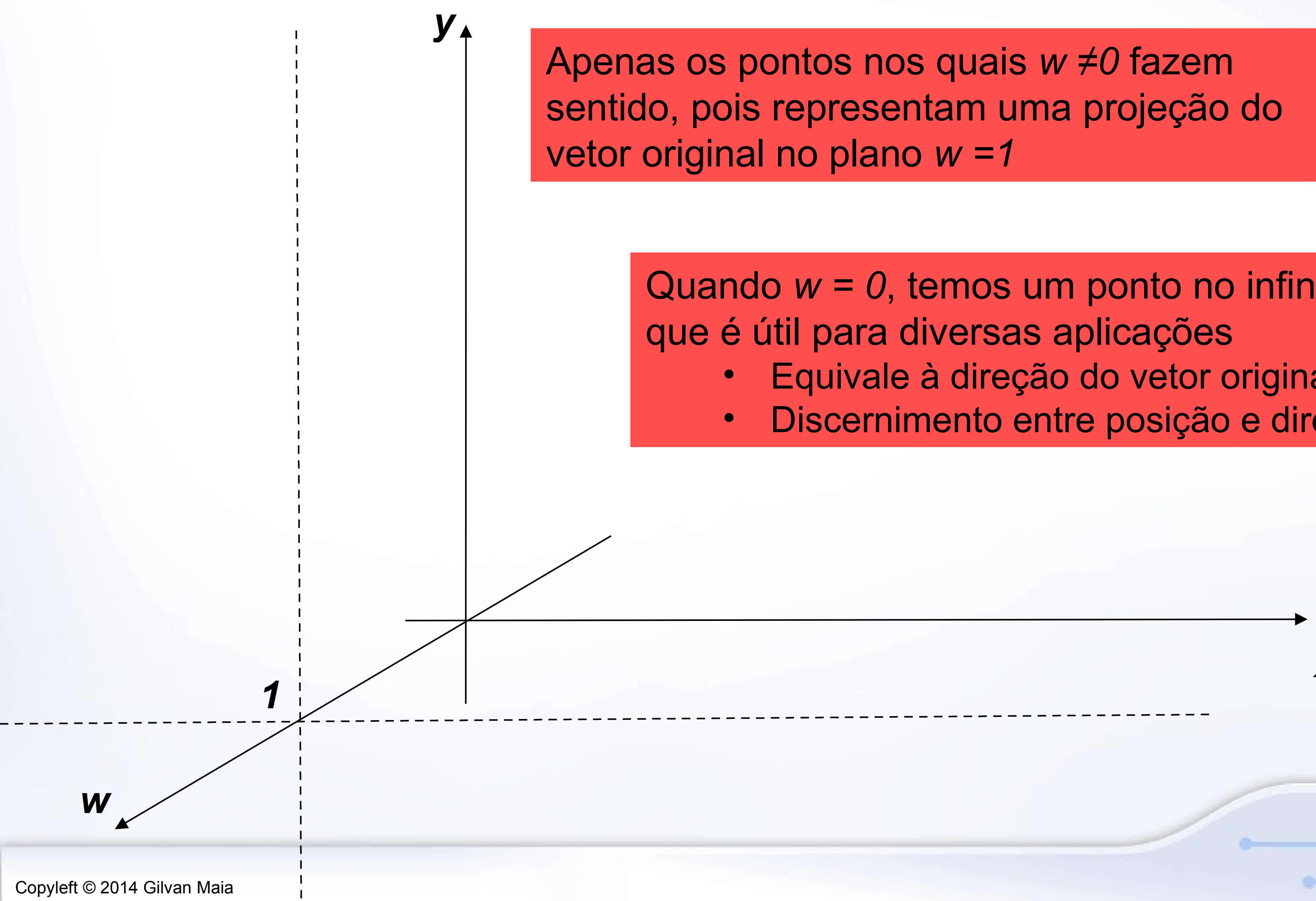
$$v^t = [x \quad y \quad w]$$

- **v** é projetado de volta para 2D usando duas operações
  - Divisão das coordenadas por **w**
    - Caso $w \neq 0$
  - Descartar a coordenada **w**

$$v^t = \left[\frac{x}{w} \quad \frac{y}{w} \quad 1\right]$$

$$v^t = \left[\frac{x}{w} \quad \frac{y}{w}\right]$$

Caso 2D: entendendo *w*

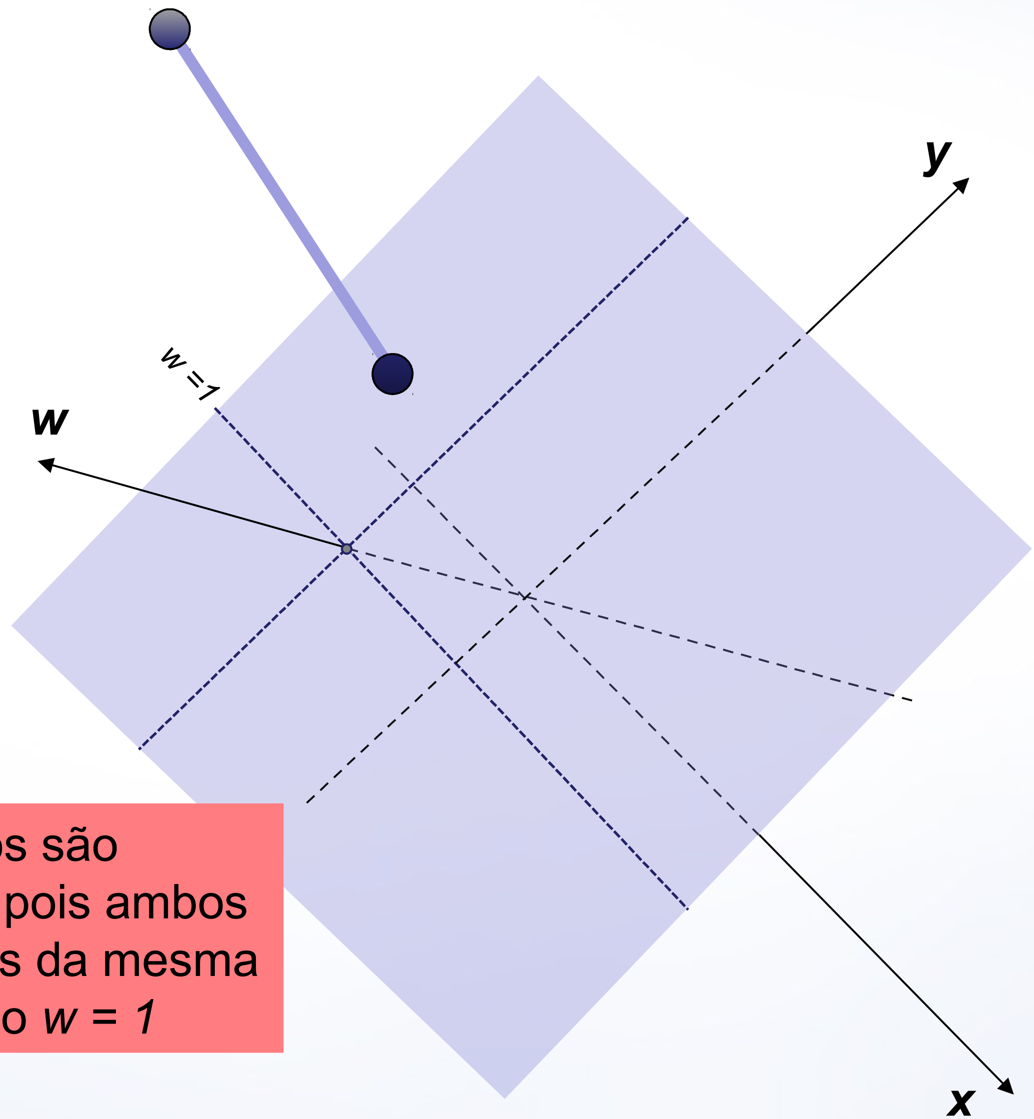


Apenas os pontos nos quais $w \neq 0$ fazem sentido, pois representam uma projeção do vetor original no plano $w = 1$

Quando $w = 0$, temos um ponto no infinito, o que é útil para diversas aplicações

- Equivale à direção do vetor original
- Discernimento entre posição e direção

Caso 2D: entendendo *w*



Com efeito, existem infinitas representações do vetor (x,y) em coordenadas homogêneas

Os dois pontos são equivalentes, pois ambos são projetados da mesma forma no plano *w = 1*

Caso 3D: entendendo *w*

- A coordenada homogênea define geometria de forma ***projetiva***
  - Pontos no infinito são chamados de “impróprios”

- Esse artifício é extremamente útil para uso de matrizes em Computação Gráfica
  - Translação
    - Operação consiste na **soma** de coordenadas
    - Uma matriz convencional é inadequada para isso
  - Projeção de 3D para 2D
    - Divisão com base nos valores das coordenadas

Representação de Vetores

- Vetores são grafados como matrizes-colunas

- Vetores são grafados como matrizes-colunas

$$v = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- A seguinte notação também pode ser usada

- A seguinte notação também pode ser usada

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

Porque Matrizes?

- Expressam transformações lineares
- São simples de se implementar
  - Adaptam-se bem ao *hardware convencional*
- Arcabouço **unificado** para lidar com vários problemas de sintetização de imagens
  - Posicionamento de modelos
  - Mapeamento de modelos para o espaço da câmera
  - Projeção de primitivas geométricas 3D em imagens 2D
  - Animação por sistema de ossos (*skeleton*)
  - ....

Matrizes de Transformação

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Multiplicação Matricial: notação

$$\mathbf{Av} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

Matrizes e Vetores

Quando *w=1,* indicamos se tratar de uma **posição**

$$\mathbf{Av} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix}$$

Direções, como os vetores normais aos vértices, usam um valor *w = 0*

Matrizes e Vetores

$$\mathbf{u} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \\ u_w \end{bmatrix} = \begin{bmatrix} a_{11}v_x + a_{12}v_y + a_{13}v_z + a_{14}v_w \\ a_{21}v_x + a_{22}v_y + a_{23}v_z + a_{24}v_w \\ a_{31}v_x + a_{32}v_y + a_{33}v_z + a_{34}v_w \\ a_{41}v_x + a_{42}v_y + a_{43}v_z + a_{44}v_w \end{bmatrix}$$

Matrizes e Vetores

- O vetor **u**' 3D é obtido pela divisão de **u** pela sua coordenada homogênea
  - Apenas quando $u_w \neq 0$
  - Caso contrário a coordenada torna-se infinita ou NaN

$$\mathbf{u}' = \frac{\mathbf{u}}{u_w} = \begin{bmatrix} u_x / u_w \\ u_y / u_w \\ u_z / u_w \\ 1 \end{bmatrix}$$

Matrizes e OpenGL

Notação matemática

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Código C/C++

- As matrizes são armazenadas em um *array* de 16 elementos
  - Os valores são passados coluna a coluna
  - É como se usássemos a *transposta* da matriz A
- Exemplo

```
double a[16] = {
    a11, a21, a31, a41,
    a12, a22, a32, a42,
    a13, a23, a33, a43,
    a14, a24, a34, a44 };
```

Matrizes e OpenGL

Escrita de código coerente em C/C++

- Tenha em mente o seguinte leiaute

```
double a[16] = {...};
```

$$\mathbf{A} = \begin{bmatrix} a[0] & a[4] & a[8] & a[12] \\ a[1] & a[5] & a[9] & a[13] \\ a[2] & a[6] & a[10] & a[14] \\ a[3] & a[7] & a[11] & a[15] \end{bmatrix}$$

Transformações Básicas

Matriz Identidade

- Representa a transformação mais básica
  - Elemento neutro da multiplicação
    - Preserva todas as coordenadas de vetores e matrizes

$$\mathbf{v} = \mathbf{I}\mathbf{v} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix} = \begin{bmatrix} 1v_x + 0 + 0 + 0 \\ 0 + 1v_y + 0 + 0 \\ 0 + 0 + 1v_z + 0 \\ 0 + 0 + 0 + 1v_w \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

Matriz de Translação

Translação

Matriz de Translação

- Denotada por $T(\Delta x, \Delta y, \Delta z)$

Denotada por

$$\begin{bmatrix} u_x \\ u_y \\ u_z \\ 1 \end{bmatrix} = \mathbf{T}(\Delta x, \Delta y, \Delta z) \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix} = \begin{bmatrix} v_x + \Delta x \\ v_y + \Delta y \\ v_z + \Delta z \\ 1 \end{bmatrix}$$

Matriz de Translação

- Dedução a partir da seguinte equação

$$\begin{bmatrix} v_x + \Delta x \\ v_y + \Delta y \\ v_z + \Delta z \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}v_x + a_{12}v_y + a_{13}v_z + a_{14} \\ a_{21}v_x + a_{22}v_y + a_{23}v_z + a_{24} \\ a_{31}v_x + a_{32}v_y + a_{33}v_z + a_{34} \\ a_{41}v_x + a_{42}v_y + a_{43}v_z + a_{44} \end{bmatrix}$$

Não se confunda: esse é um vetor-coluna!

Representa o produto da matriz A**v**

Matriz de Translação

- Dedução a partir da seguinte equação

$$\begin{bmatrix} v_x + \Delta x \\ v_y + \Delta y \\ v_z + \Delta z \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}v_x + a_{12}v_y + a_{13}v_z + a_{14} \\ a_{21}v_x + a_{22}v_y + a_{23}v_z + a_{24} \\ a_{31}v_x + a_{32}v_y + a_{33}v_z + a_{34} \\ a_{41}v_x + a_{42}v_y + a_{43}v_z + a_{44} \end{bmatrix}$$

Matriz de Translação

$$\begin{bmatrix} v_x + \Delta x \\ v_y + \Delta y \\ v_z + \Delta z \\ 1 \end{bmatrix} = \begin{bmatrix} 1v_x + 0v_y + 0v_z + \Delta x \\ 0v_x + 1v_y + 0v_z + \Delta y \\ 0v_x + 0v_y + 1v_z + \Delta z \\ 0v_x + 0v_y + 0v_z + 1 \end{bmatrix}$$

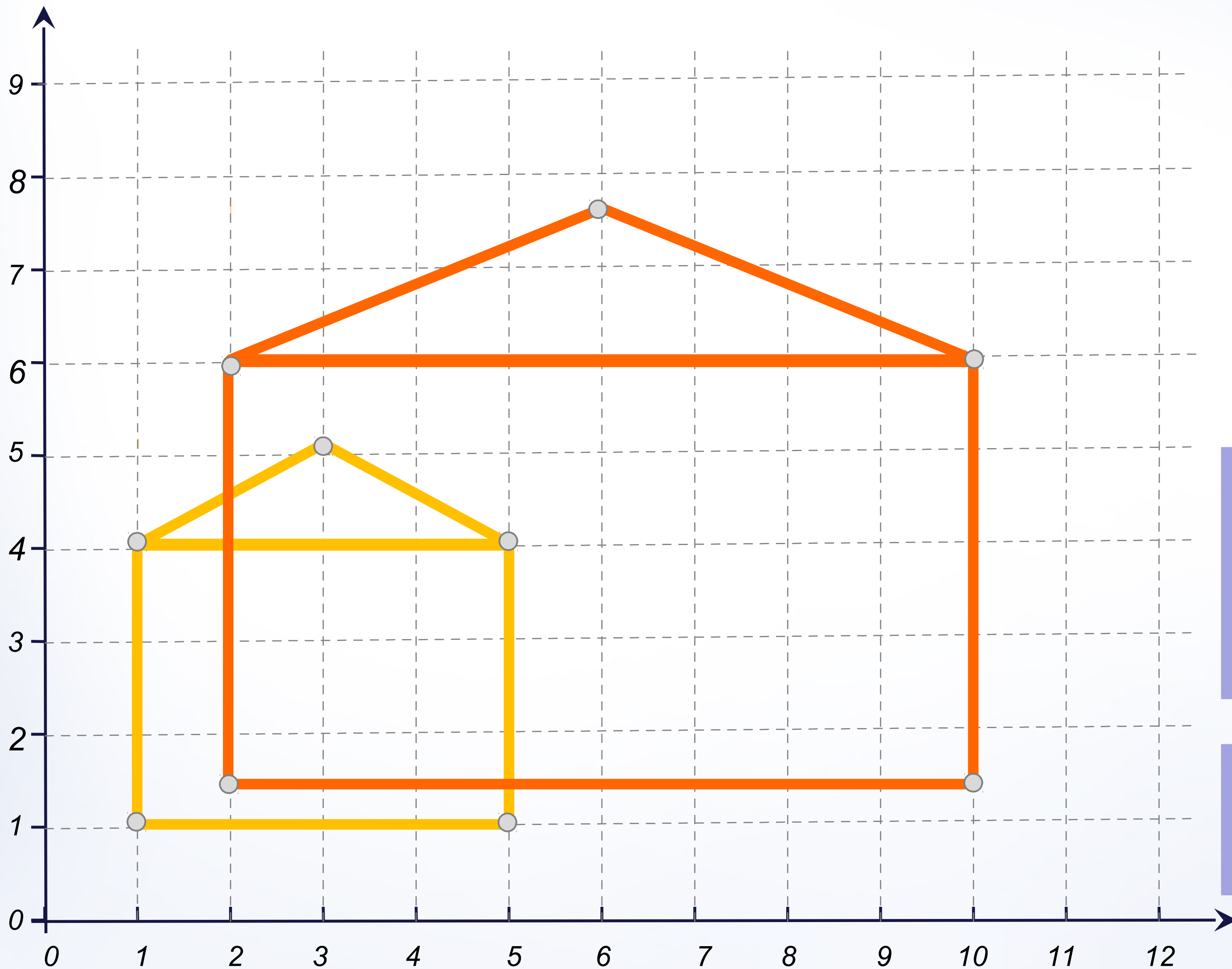
Matriz de Translação

- Assim, temos

$$\mathbf{T}(\Delta x, \Delta y, \Delta z) = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de Escala

Escala



A coordenada X está **2 vezes maior**

A coordenada Y está **1,5 vezes maior**

Escalar significa **multiplicar** cada coordenada por um **fator**

Importante:
a mesma operação é feita para todos os vértices

Quando os fatores são **idênticos** para todos os eixos, a escala é dita **uniforme**

Matriz de Escala

- A mais óbvia, pois segue a identidade

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

Matriz de Rotação

Rotação

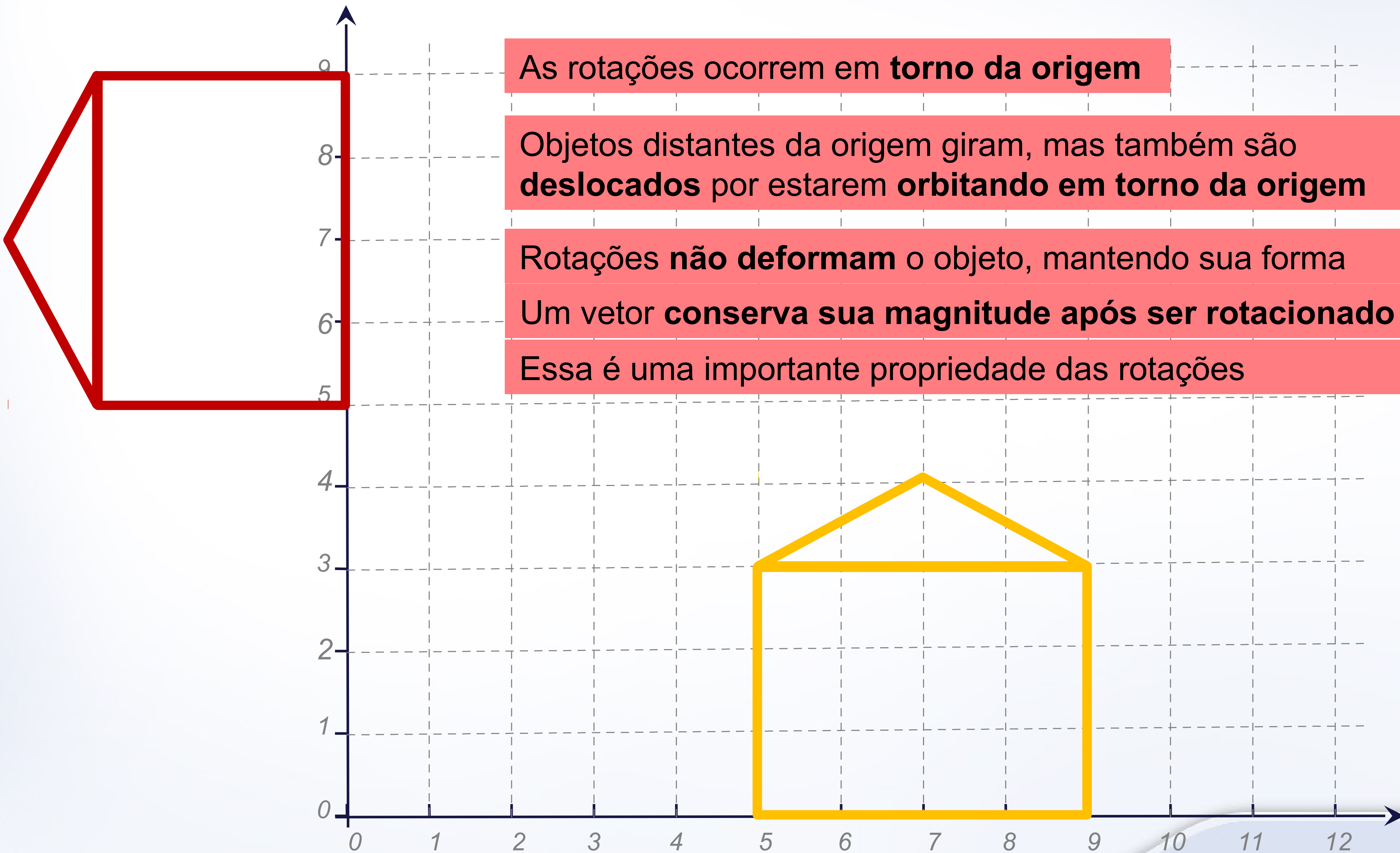
As rotações ocorrem em **torno da origem**

Objetos distantes da origem giram, mas também são **deslocados** por estarem **orbitando em torno da origem**

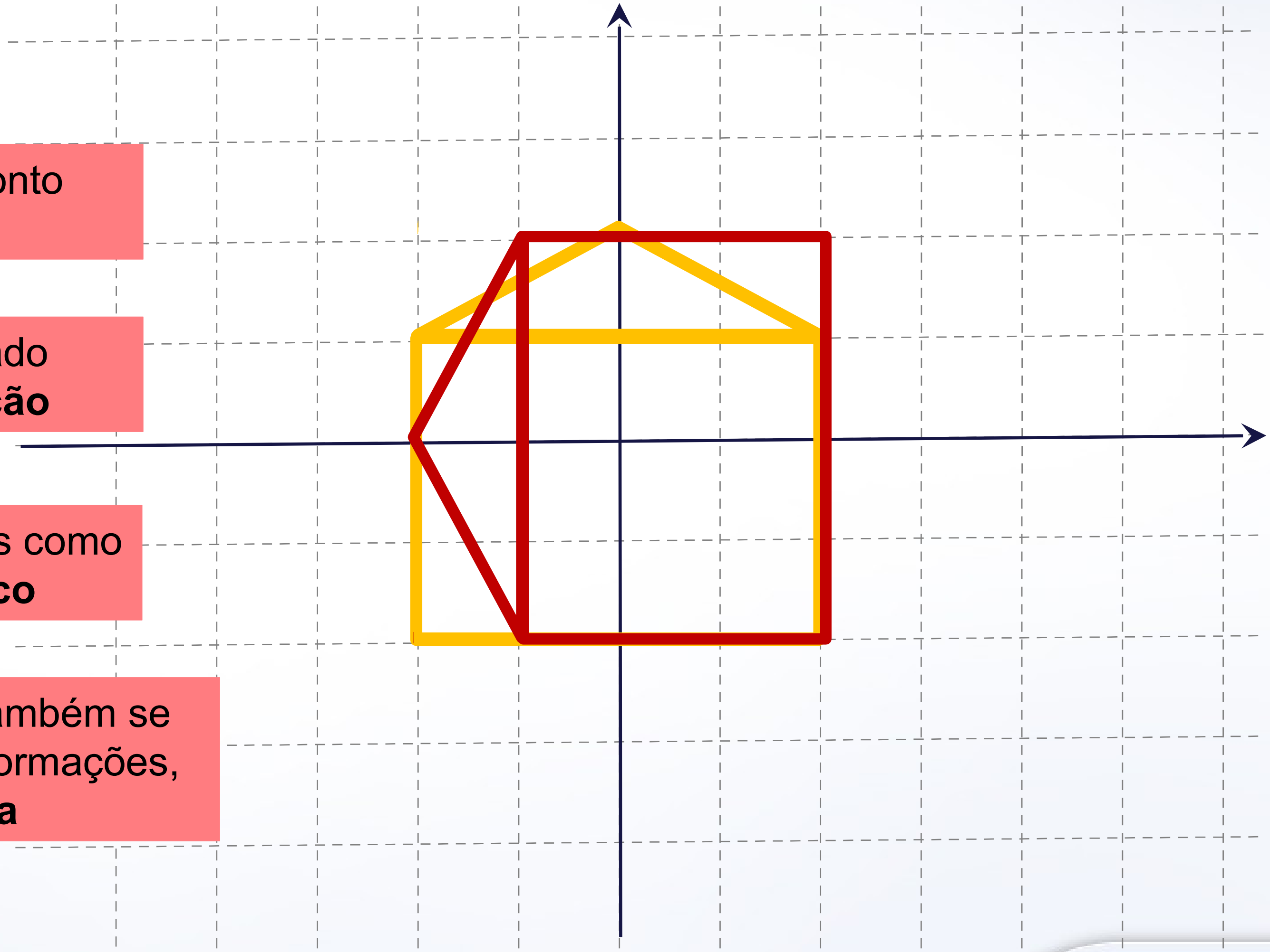
Rotações **não deformam** o objeto, mantendo sua forma

Um vetor **conserva sua magnitude após ser rotacionado**

Essa é uma importante propriedade das rotações



Rotação: centro na origem



A origem é o único ponto invariante à rotação

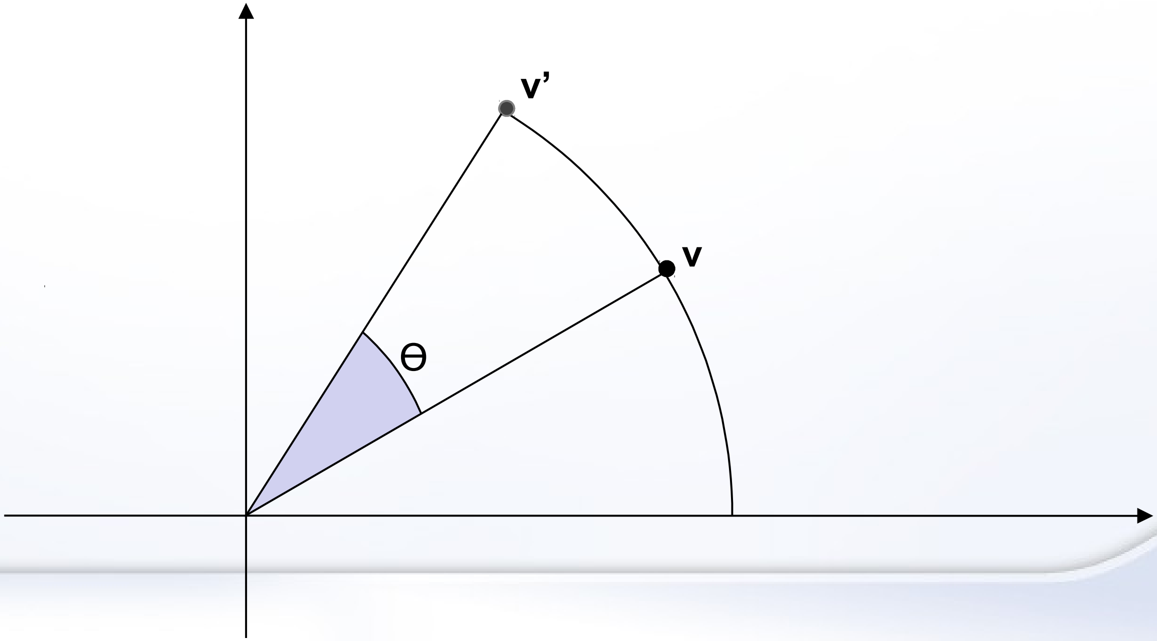
Tal ponto é denominado **pivô da transformação**

Mais adiante veremos como usar um **pivô genérico**

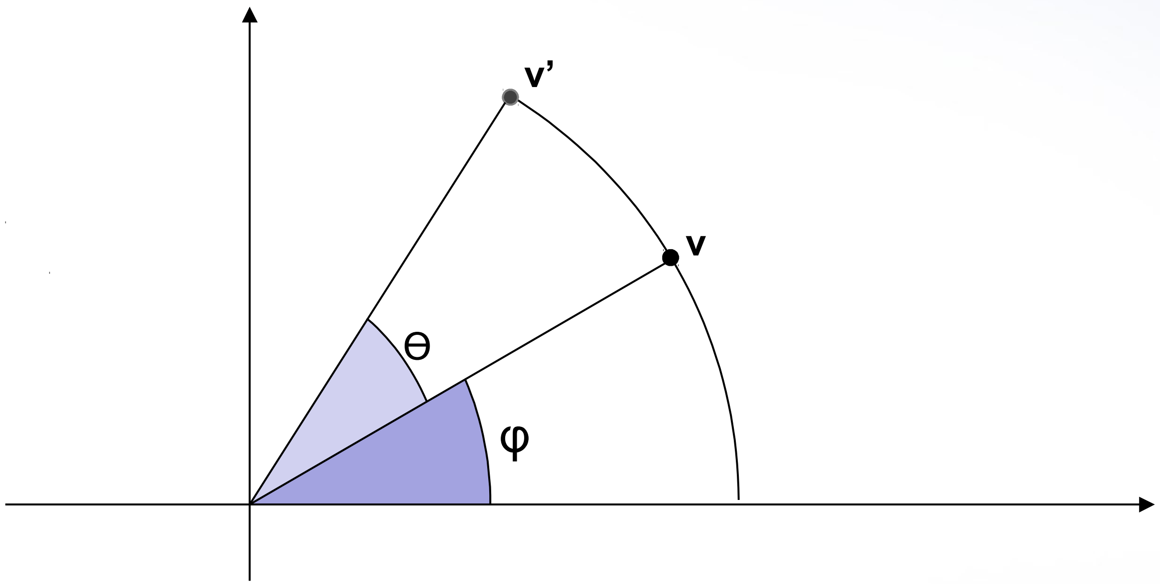
O conceito de **pivô** também se aplica a outras transformações, em particular à **escala**

Caracterização da Rotação

- Sejam v e v' dois pontos
  - v e v' são equidistantes da origem
  - v' é obtido por meio de uma rotação de v em torno da origem
  - Seja o ângulo ϴ



Caracterização da Rotação



$$\boldsymbol{v}^t = (v_x \quad v_y) = (r.cos\varphi \quad r.sen\varphi)$$

$$\boldsymbol{v}'^t = (r.\cos(\varphi + \theta) \quad r.sen(\varphi + \theta))$$

$$\boldsymbol{v}'^t = (r(\cos\varphi\cos\theta - \mathrm{sen}\varphi\mathrm{sen}\theta) \quad r(\cos\varphi\mathrm{sen}\theta + \mathrm{sen}\varphi\cos\theta))$$

$$\boldsymbol{v}'^t = ((r\cos\varphi\cos\theta - r\mathrm{sen}\varphi\mathrm{sen}\theta) \quad (r\cos\varphi\mathrm{sen}\theta + r\mathrm{sen}\varphi\cos\theta))$$

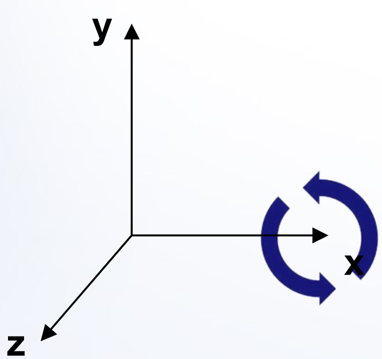
$$\boldsymbol{v}'^t = (\, v_x\cos\theta - v_y\,\mathrm{sen}\theta \quad v_x\,\mathrm{sen}\theta + v_y\,cos\theta \quad)$$

Forma matricial simplificada

$$\begin{pmatrix} v'_x \\ v'_y \end{pmatrix} = \begin{pmatrix} cos\theta & -sen\theta \\ sen\theta & cos\theta \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

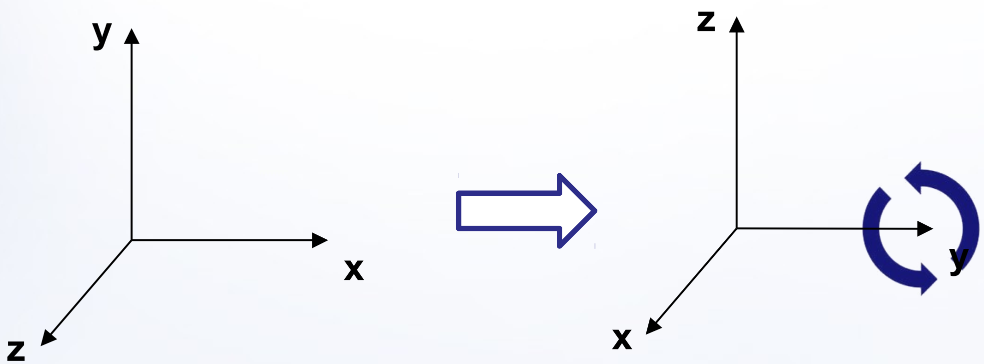
Matriz de Rotação

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



Matriz de Rotação: Eixo X

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



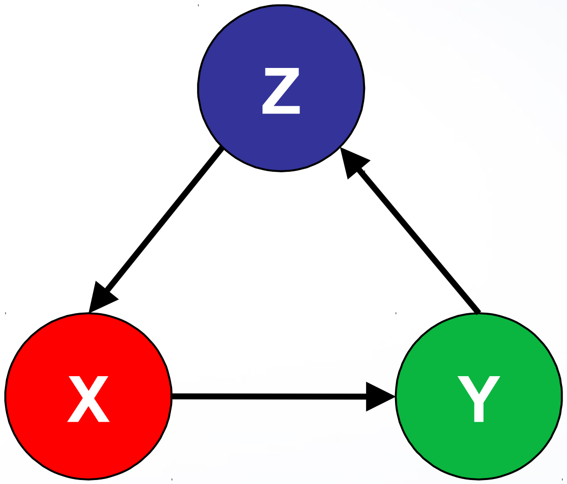
Matriz de Rotação: Eixo Y

Porque o sinal está invertido?

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

Entendendo a Rotação em Y

- Rotação de 90º em Z
  - Transforma o eixo X em Y
- Rotação de 90º em X
  - Transforma o eixo Y em Z
- Rotação de 90º em Y
  - Transforma o eixo Z em X



- Temos um ciclo consistente de rotações

  X → Y → Z → ***X***

Matrizes como Bases Vetoriais

Matrizes e Bases Vetoriais

- Podemos reescrever **A** usando blocos
  - Toda coluna de **A** é um vetor

$$\mathbf{v}' = \mathbf{A}\mathbf{v} = \begin{bmatrix} \vec{\mathbf{a}} & \vec{\mathbf{b}} & \vec{\mathbf{c}} & \vec{\mathbf{d}} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

a é um vetor, ou seja, uma *matriz-coluna* com quatro coordenadas

A notação em bloco é *conveniente* para expressar transformações de forma ***sucinta***

Essa notação facilita cálculos e facilita a caracterização de várias transformações

As fórmulas para as inversas de matrizes compostas, por exemplo, geralmente são deduzidas usando a notação em blocos

Matrizes e Bases Vetoriais

- O produto **A.v** pode ser reescrito como

$$\mathbf{v}' = \mathbf{A}\mathbf{v} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ a_w & b_w & c_w & d_w \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

Matrizes e Bases Vetoriais

- Expandindo a expressão, obtemos

$$\mathbf{v}' = \begin{bmatrix} a_x v_x + b_x v_y + c_x v_z + d_x v_w \\ a_y v_x + b_y v_y + c_y v_z + d_y v_w \\ a_z v_x + b_z v_y + c_z v_z + d_z v_w \\ a_w v_x + b_w v_y + c_w v_z + d_w v_w \end{bmatrix}$$

Matrizes e Bases Vetoriais

- Expandindo a expressão, obtemos

$$\mathbf{v}' = v_x \begin{bmatrix} a_x \\ a_y \\ a_z \\ a_w \end{bmatrix} + v_y \begin{bmatrix} b_x \\ b_y \\ b_z \\ b_w \end{bmatrix} + v_z \begin{bmatrix} c_x \\ c_y \\ c_z \\ c_w \end{bmatrix} + v_w \begin{bmatrix} d_x \\ d_y \\ d_z \\ d_w \end{bmatrix}$$

$$\mathbf{v}' = v_x \mathbf{a} + v_y \mathbf{b} + v_z \mathbf{c} + v_w \mathbf{d}$$

Perceba que os vetores **a**, **b**, **c** e **d** são uma base vetorial para o vetor **v**, o que fundamentalmente produz o vetor **v’** final

Matrizes Ortogonais

- Quando **a**, **b**, **c** e **d** são uma base *ortonormal*
  - Os vetores **a**, **b**, **c** e **d** são unitários
  - Esses vetores são todos *perpendiculares entre si*
    - O produto escalar ***a.b*** é igual zero
    - O produto escalar ***a.c*** é igual zero
    - O produto escalar ***a.d*** é igual zero
    - O produto escalar ***b.c*** é igual zero
    - O produto escalar ***b.d*** é igual zero
    - O produto escalar ***c.d*** é igual zero
- A inversa da matriz ortogonal é a sua transposta
  - Toda matriz de rotação é ortogonal
  - Isso permite inverter rotações *muito facilmente*

Inversa da Matriz Ortogonal

- A inversa da matriz ortogonal é a sua transposta
  - Toda matriz de rotação é ortogonal
  - Isso permite inverter rotações *muito facilmente*

$$\mathbf{A}^t\mathbf{A} = \begin{bmatrix} \vec{\mathbf{a}}^t \\ \vec{\mathbf{b}}^t \\ \vec{\mathbf{c}}^t \\ \vec{\mathbf{d}}^t \end{bmatrix} \begin{bmatrix} \vec{\mathbf{a}} & \vec{\mathbf{b}} & \vec{\mathbf{c}} & \vec{\mathbf{d}} \end{bmatrix}$$

$$\mathbf{A}^t\mathbf{A} = \begin{bmatrix} \vec{\mathbf{a}}^t\vec{\mathbf{a}} & \vec{\mathbf{a}}^t\vec{\mathbf{b}} & \vec{\mathbf{a}}^t\vec{\mathbf{c}} & \vec{\mathbf{a}}^t\vec{\mathbf{d}} \\ \vec{\mathbf{b}}^t\vec{\mathbf{a}} & \vec{\mathbf{b}}^t\vec{\mathbf{b}} & \vec{\mathbf{b}}^t\vec{\mathbf{c}} & \vec{\mathbf{b}}^t\vec{\mathbf{d}} \\ \vec{\mathbf{c}}^t\vec{\mathbf{a}} & \vec{\mathbf{c}}^t\vec{\mathbf{b}} & \vec{\mathbf{c}}^t\vec{\mathbf{c}} & \vec{\mathbf{c}}^t\vec{\mathbf{d}} \\ \vec{\mathbf{d}}^t\vec{\mathbf{a}} & \vec{\mathbf{d}}^t\vec{\mathbf{b}} & \vec{\mathbf{d}}^t\vec{\mathbf{c}} & \vec{\mathbf{d}}^t\vec{\mathbf{d}} \end{bmatrix}$$

Inversa da Matriz Ortogonal

$$\mathbf{A}^t\mathbf{A} = \begin{bmatrix} \vec{\mathbf{a}}^t\vec{\mathbf{a}} & \cancel{\vec{\mathbf{a}}^t\vec{\mathbf{b}}} & \cancel{\vec{\mathbf{a}}^t\vec{\mathbf{c}}} & \cancel{\vec{\mathbf{a}}^t\vec{\mathbf{d}}} \\ \cancel{\vec{\mathbf{b}}^t\vec{\mathbf{a}}} & \vec{\mathbf{b}}^t\vec{\mathbf{b}} & \cancel{\vec{\mathbf{b}}^t\vec{\mathbf{c}}} & \cancel{\vec{\mathbf{b}}^t\vec{\mathbf{d}}} \\ \cancel{\vec{\mathbf{c}}^t\vec{\mathbf{a}}} & \cancel{\vec{\mathbf{c}}^t\vec{\mathbf{b}}} & \vec{\mathbf{c}}^t\vec{\mathbf{c}} & \cancel{\vec{\mathbf{c}}^t\vec{\mathbf{d}}} \\ \cancel{\vec{\mathbf{d}}^t\vec{\mathbf{a}}} & \cancel{\vec{\mathbf{d}}^t\vec{\mathbf{b}}} & \cancel{\vec{\mathbf{d}}^t\vec{\mathbf{c}}} & \vec{\mathbf{d}}^t\vec{\mathbf{d}} \end{bmatrix}$$

É fácil mostrar que **A.A**^{t} = **I**

$$\mathbf{A}^t\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{I}$$

Logo, as inversas **A**$^{-1}$ = **A**^{t} e (**A**^{t})$^{-1}$= **A** são triviais

Rotação Sobre um Eixo Arbitrário

Rotação Sobre um Eixo Arbitrário

- Tomemos o vetor **u** como eixo desejado para a rotação
  - A ideia é transformar **u** no eixo Z usando rotações
    - Isso simplifica a rotação
  - Após isso, aplica-se a rotação em Z
    - Note-se que o eixo da rotação não é alterado
    - No momento, **u** está alinhado com Z
  - Por fim, o eixo Z é rotacionado de volta em **u**
    - Isso traz **u** volta à sua configuração original
    - Apesar de parecer complicado, basta conhecer

Rotação Sobre um Eixo Arbitrário

- Cinco passos
  1. Rotacione **u** em torno de X até que atinja o plano XZ
     - Obtemos assim um vetor **u’**
  2. Rotacione **u’** em torno de Y até coincidir com Z
  3. Realize a rotação sobre Z
  4. Desfaça o passo 2, trazendo o eixo Z para **u’**
  5. Desfaça o passo 1, trazendo **u’** para **u**

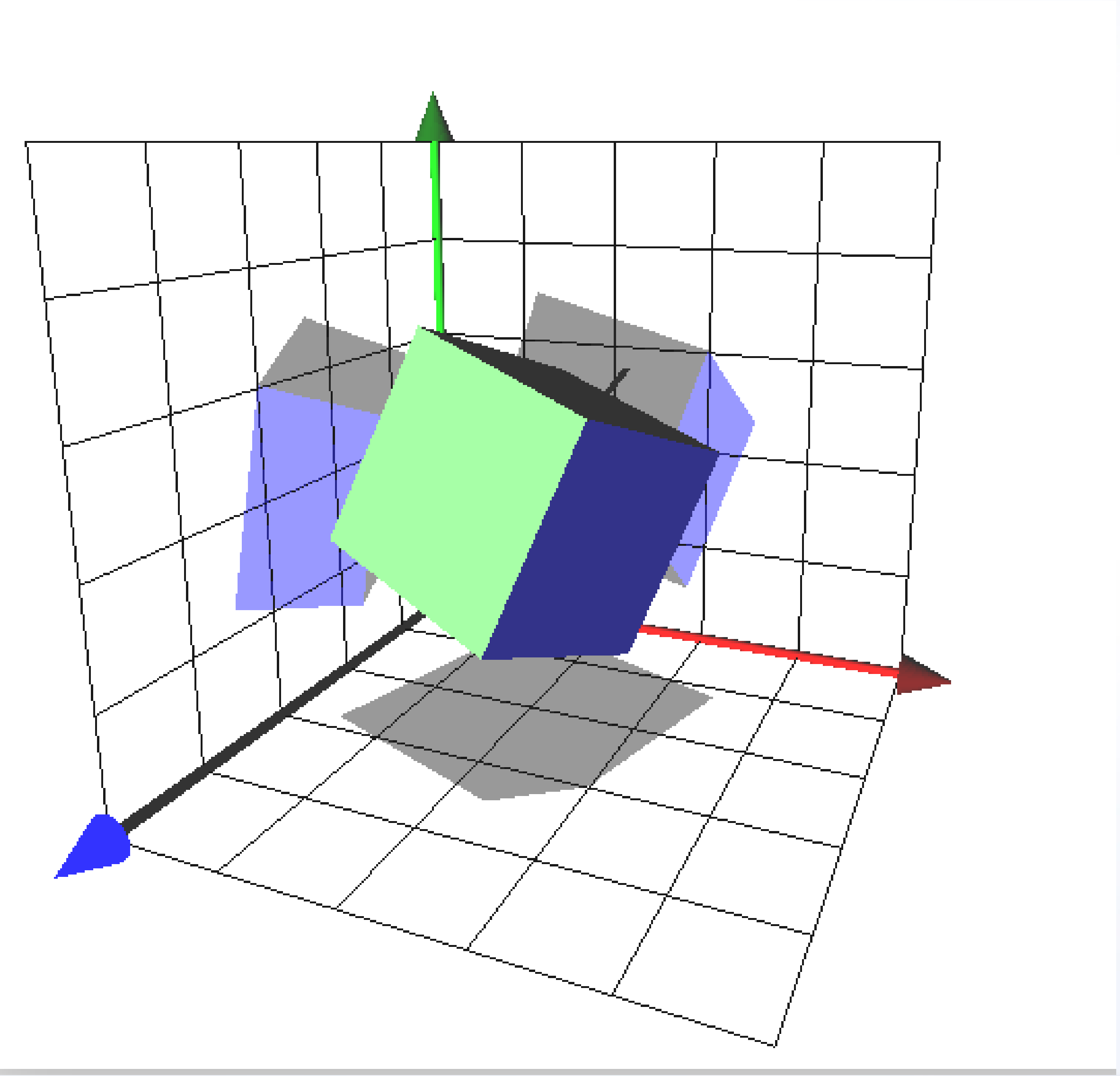
Rotação Sobre um Eixo Arbitrário

- Representação matricial
  - As matrizes são aplicadas da direita para a esquerda

$$R_u(\theta) = R_x^{-1}(\theta_1) R_y^{-1}(\theta_2) R_z(\theta) R_y(\theta_2) R_x(\theta_1)$$

- Como funciona
  - As duas primeiras rotações simplificam o problema
  - A terceira rotação produz o efeito desejado
  - As duas últimas rotações desfazem as das primeiras

Visualizando cada Passo



Rotação em X

Eixo no plano YZ

Rotação em Y

Eixo em Z

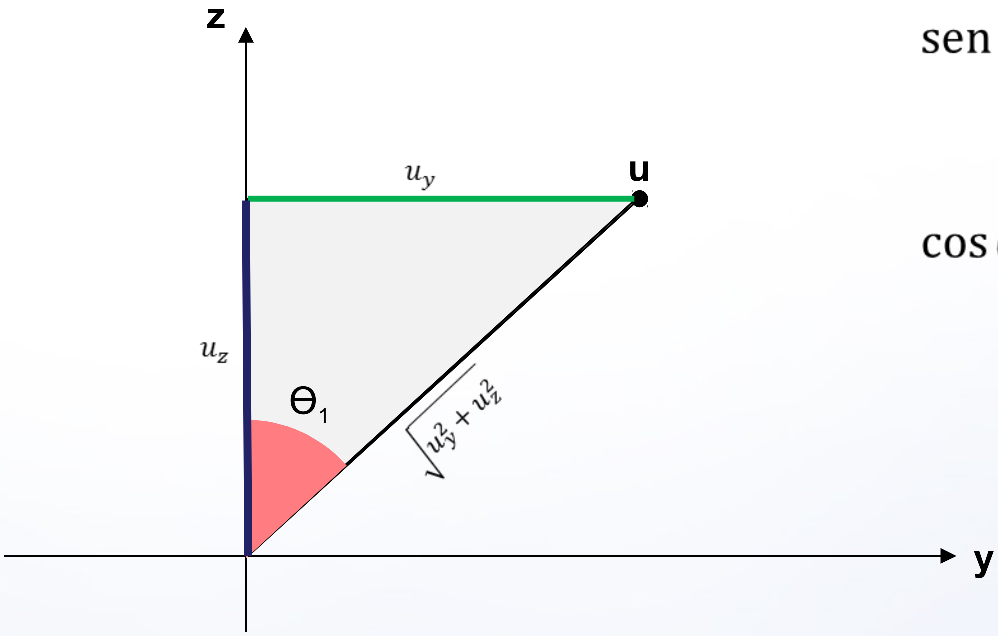
Rotação em Z

Desfazer rotação Y

Desfazer rotação X

Passo 1: Rotação sobre Eixo X

- A rotação é calculada considerando a projeção do vetor u (eixo de rotação) no plano YZ
  - A coordenada X é desconsiderada



$$\operatorname{sen}\theta_1 = \frac{u_y}{\sqrt{u_y^2 + u_z^2}}$$

$$\cos\theta_1 = \frac{u_z}{\sqrt{u_y^2 + u_z^2}}$$

Passo 1: Rotação sobre Eixo X

- Obtendo a matriz
  - Basta usar o seno e cosseno na fórmula anterior

$$\mathbf{R}_x(\theta_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & -sen(\theta_1) & 0 \\ 0 & sen(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

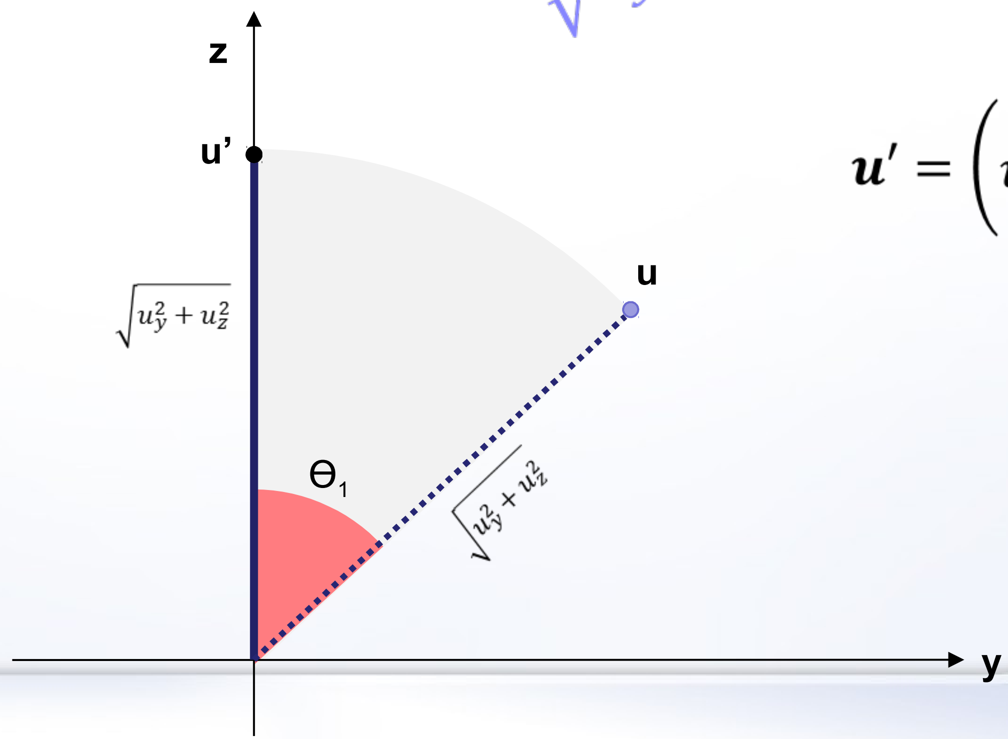
$$\operatorname{sen}\theta_1 = \frac{u_y}{\sqrt{u_y^2 + u_z^2}}$$

$$\cos\theta_1 = \frac{u_z}{\sqrt{u_y^2 + u_z^2}}$$

Passo 1: Rotação sobre Eixo X

- Após a rotação, observa-se que
  - A coordenada X não é alterada pela rotação
  - A coordenada Y torna-se 0
  - A coordenada Z torna-se $\sqrt{u_y^2 + u_z^2}$

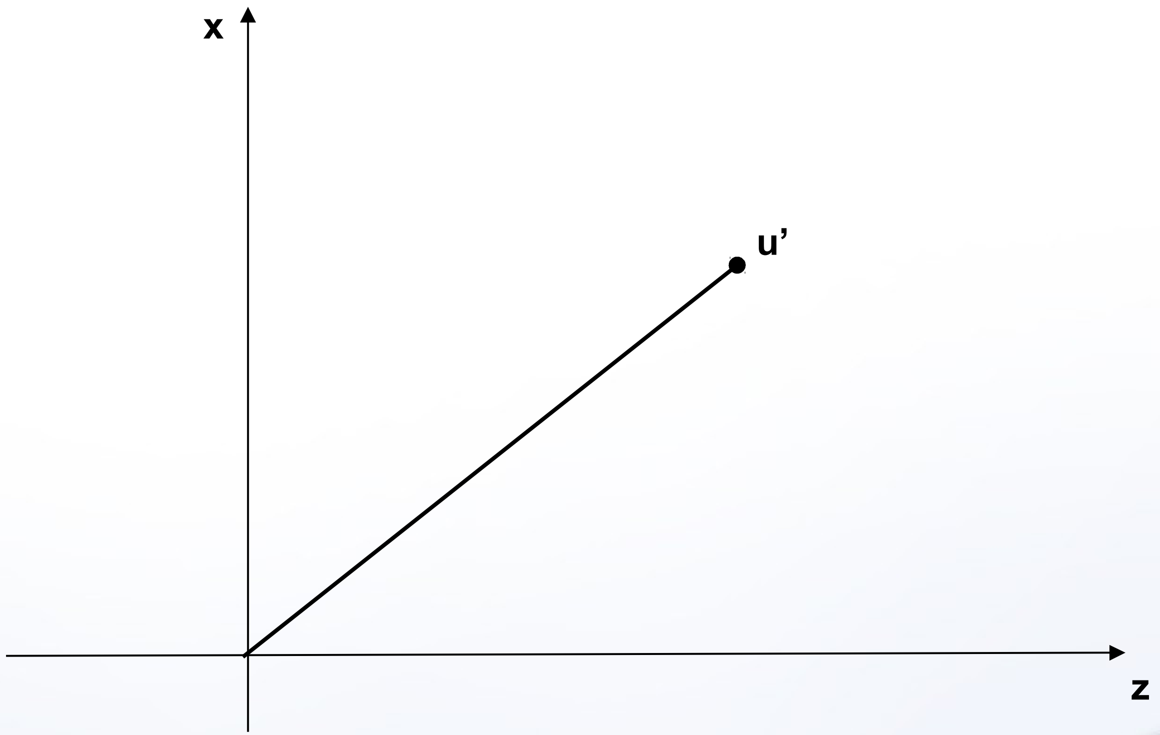
- Após a rotação, observa-se que
  - A coordenada X não é alterada pela rotação
  - A coordenada Y torna-se 0
  - A coordenada Z torna-se



$$\boldsymbol{u}' = \begin{pmatrix} u_x & 0 & \sqrt{u_y^2 + u_z^2} \end{pmatrix}$$

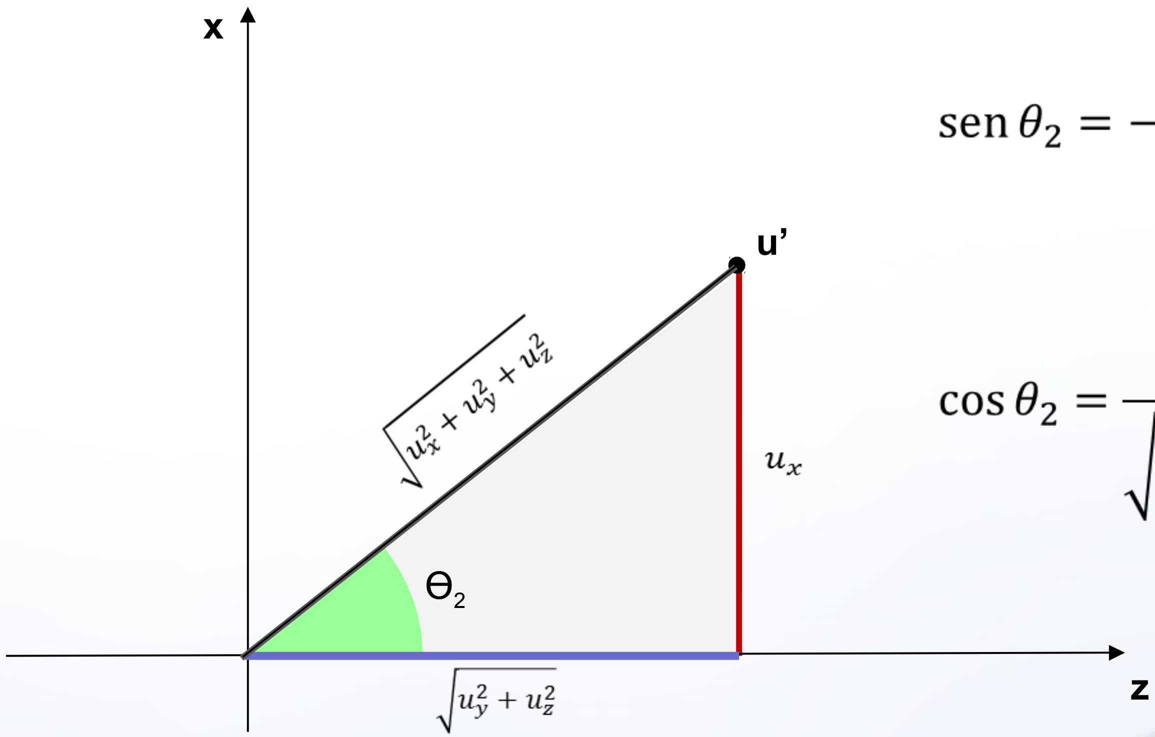
Passo 2: Rotação sobre Eixo Y

- O ponto **u’** já está posicionado no plano XZ
  - Isso significa que a coordenada Y é zero



Passo 2: Rotação sobre Eixo Y

- O objetivo é alinhar **u'** com o eixo Z
  - A rotação desejada é feita no sentido horário
  - O ângulo de rotação é *negativo*!



$$\operatorname{sen}\theta_2 = -\frac{u_x}{\sqrt{u_x^2 + u_y^2 + u_z^2}}$$

$$\cos\theta_2 = \frac{\sqrt{u_y^2 + u_z^2}}{\sqrt{u_x^2 + u_y^2 + u_z^2}}$$

Passo 2: Rotação sobre Eixo Y

- Obtendo a matriz, por substituição
  - Cuidado com os sinais dos senos!

$$\mathbf{R}_y(\theta_2) = \begin{bmatrix} \cos(\theta_2) & 0 & sen(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ -sen(\theta_2) & 0 & \cos(\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\operatorname{sen}\theta_2 = -\frac{u_x}{\sqrt{u_x^2 + u_y^2 + u_z^2}}$$

$$\cos\theta_2 = \frac{\sqrt{u_y^2 + u_z^2}}{\sqrt{u_x^2 + u_y^2 + u_z^2}}$$

Passo 3: Rotação sobre Eixo Z

- Finalmente a rotação desejada!
  - Basta usar a matriz de rotação convencional
  - Note que o eixo **u''** e Z estão alinhados
    - **u''** possui a mesma norma do eixo original **u**
    - **u''** não é afetado pela rotação

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

Passo 4: desfazendo $R_y(\theta_2)$

- Basta usar a matriz transposta
  - Toda matriz de rotação é ortogonal por definição

$$\mathbf{R}_y^{-1}(\theta_2) = \mathbf{R}_y^{T}(\theta_2) = \begin{bmatrix} \cos(\theta_2) & 0 & -sen(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ sen(\theta_2) & 0 & \cos(\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Passo 5: desfazendo $R_x(\theta_1)$

- Novamente, usamos a matriz transposta
  - Note-se que $\operatorname{sen}(-\alpha) = -(\operatorname{sen}(\alpha))$

$$\mathbf{R}_x^{-1}(\theta_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & sen(\theta_1) & 0 \\ 0 & -sen(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotação Sobre Eixo Arbitrário

- Conhecendo cada matriz, é preciso computar

- Conhecendo cada matriz, é preciso computar

$$R_u(\theta) = R_x^{-1}(\theta_1) R_y^{-1}(\theta_2) R_z(\theta) R_y(\theta_2) R_x(\theta_1)$$

- Cálculos menos complexos quando $\|\boldsymbol{u}\| = 1$
  - Assim, obtém-se a seguinte matriz

- Cálculos menos complexos quando
  - Assim, obtém-se a seguinte matriz

$$\begin{bmatrix} u_x^2(1-\cos\theta)+\cos\theta & u_xu_y(1-\cos\theta)-u_z sen\theta & u_xu_z(1-\cos\theta)+u_y sen\theta & 0 \\ u_xu_y(1-\cos\theta)+u_z sen\theta & u_y^2(1-\cos\theta)+\cos\theta & u_yu_z(1-\cos\theta)-u_x sen\theta & 0 \\ u_xu_z(1-\cos\theta)-u_y sen\theta & u_yu_z(1-\cos\theta)+u_x sen\theta & u_z^2(1-\cos\theta)+\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Outras Matrizes Úteis

Outras Matrizes Úteis

- Até agora vimos transformações básicas
  - Porém entendendo a fundo como funcionam
  - Translações, escalas e rotações
- Assuntos para os próximos slides
  - Composição de Transformações
  - Pipeline de Transformações
  - Mudanças de Base
  - Matrizes de Câmera
  - Matrizes de Projeção