MAC420/5744: Introdução à Computação Gráfica

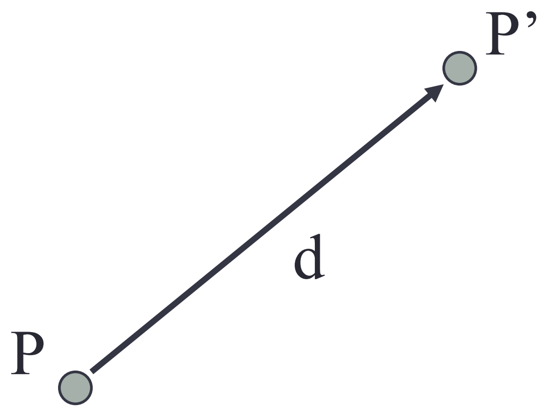
Marcel P. Jackowski

mjack@ime.usp.br

Aula #8: Transformações

Translação

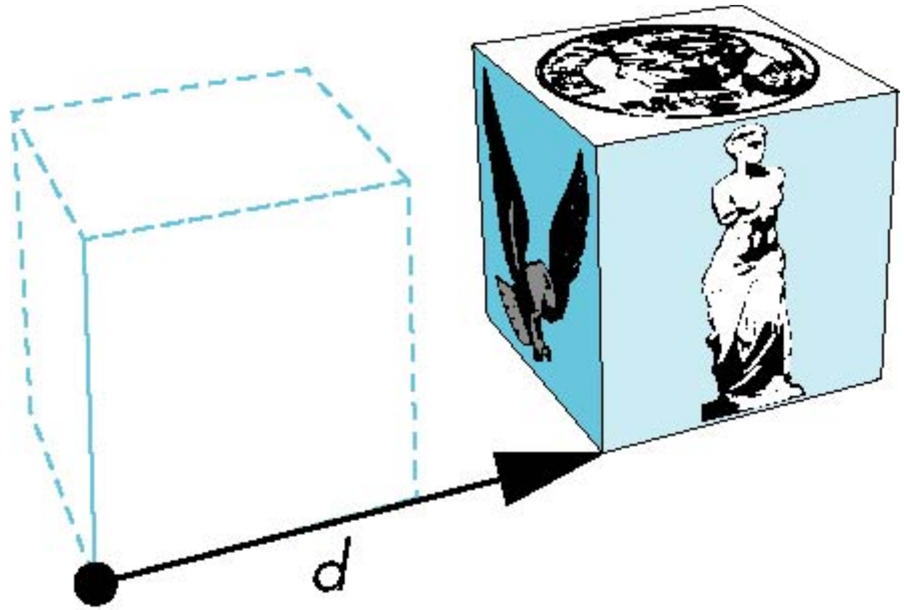
- Mover um ponto para uma nova posição



- Translação determinada pelo vetor d
  - P'=P+d
- Quantos graus de liberdade ?

Movendo vários pontos

objeto



cada ponto é transladado
pelo mesmo vetor

Coordenadas homogêneas

$\mathbf{p} = [\ x\ y\ z\ 1\ ]^T$

$\mathbf{p}' = [\ x'\ y'\ z'\ 1\ ]^T$

$\mathbf{t} = [\ t_x\ t_y\ t_z\ 0\ ]^T$

Então $\mathbf{p}' = \mathbf{p} + \mathbf{t}$ ou

$x' = x + t_x$

$y' = y + t_y$

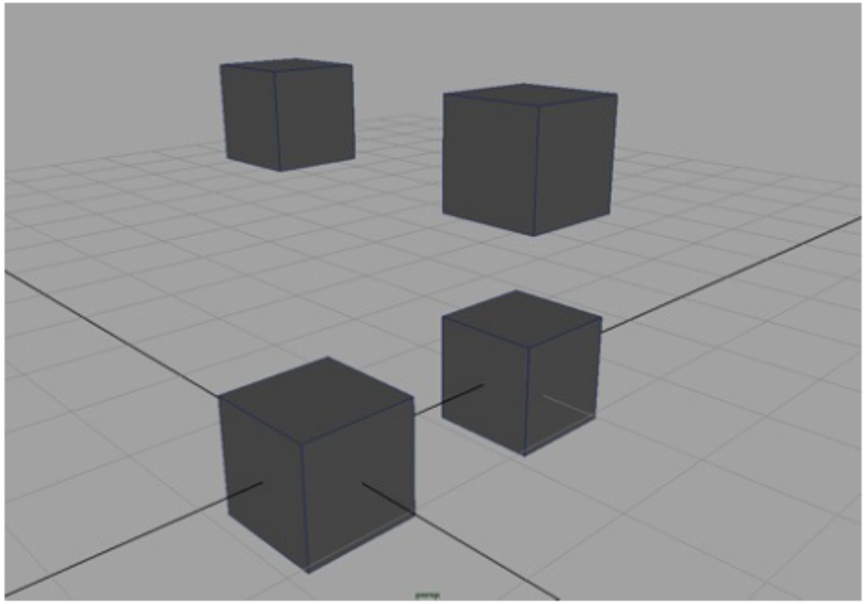
$z' = z + t_z$

note que esta expressão está em 4D e representa a operação ponto = ponto + vetor

Matriz de translação

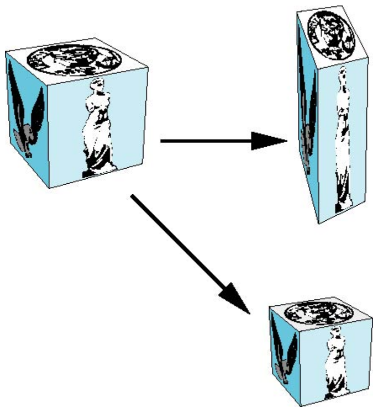
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

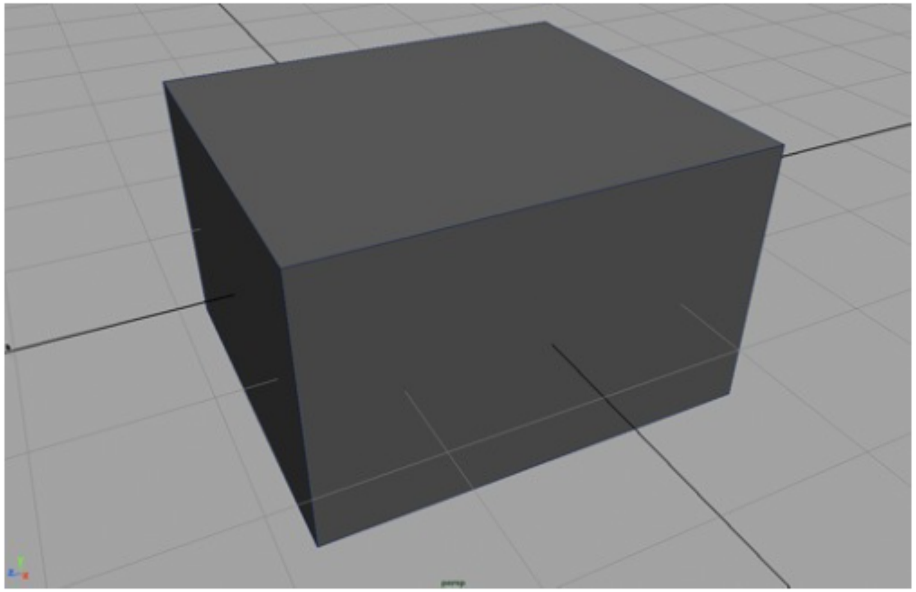
p'= Tp



Matriz de escala

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





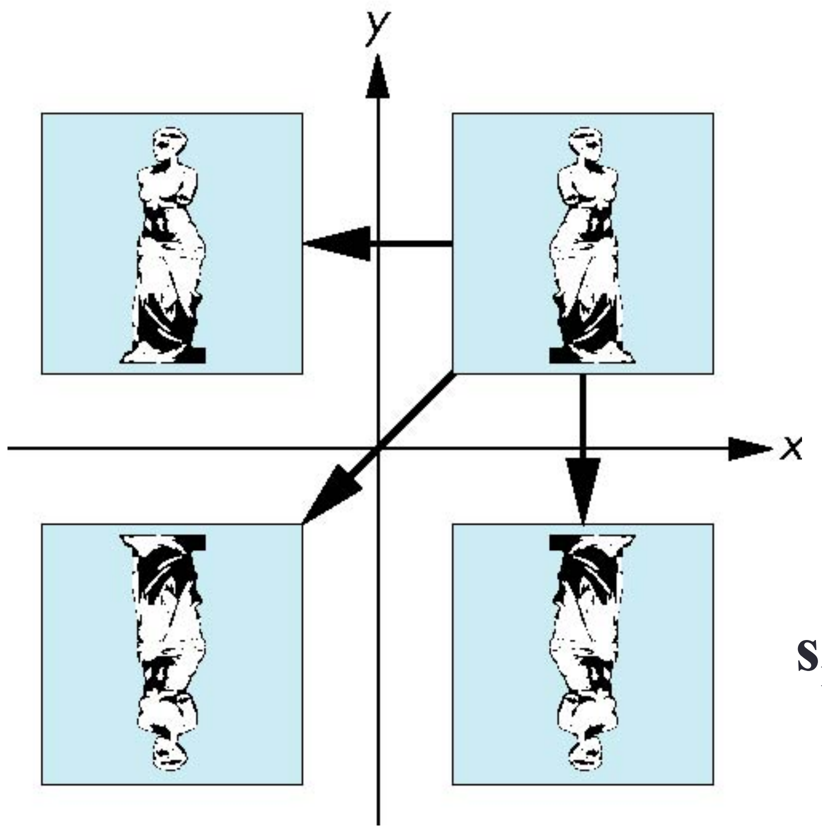
O que acontecerá se a origem do nosso referencial não for o ponto (0, 0, 0) ?

Reflexão

Correspondente a fatores negativos de escala

$s_x = -1 \; s_y = 1$

original



$s_x = -1 \; s_y = -1$

$s_x = 1 \; s_y = -1$

Inversas

- Inversa da translação (T^{-1}):

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

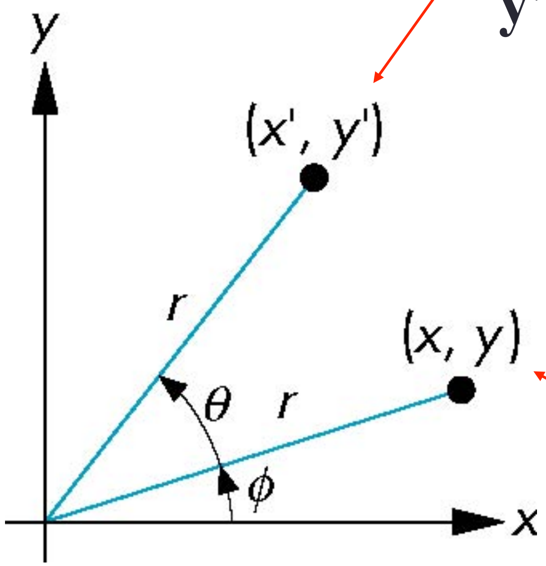
- Inversa da escala (S^{-1}):

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Rotação em 2D

- Considere uma rotação sob a origem em θ graus
  - Raio permanece o mesmo, mas o ângulo é acrescido de θ

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



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

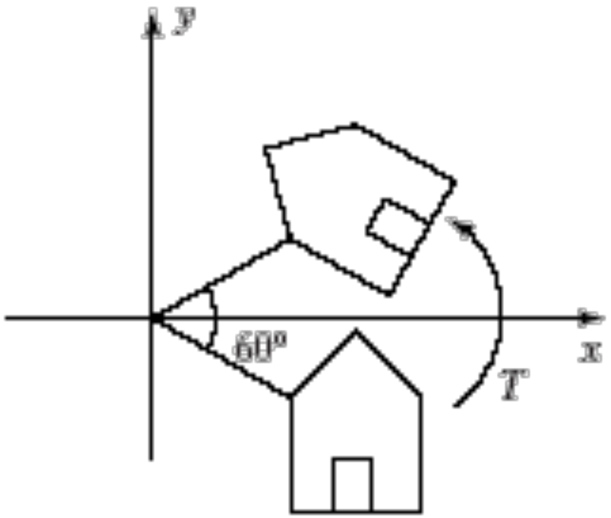
$$x = r\cos\phi$$
$$y = r\sin\phi$$

$$cos(\theta + \Phi) = cos(\theta)\,cos(\Phi) - sin(\theta)\,sin(\Phi)$$
$$sin(\theta + \Phi) = sin(\theta)\,cos(\Phi) + cos(\theta)\,sin(\Phi)$$

Matriz de rotação

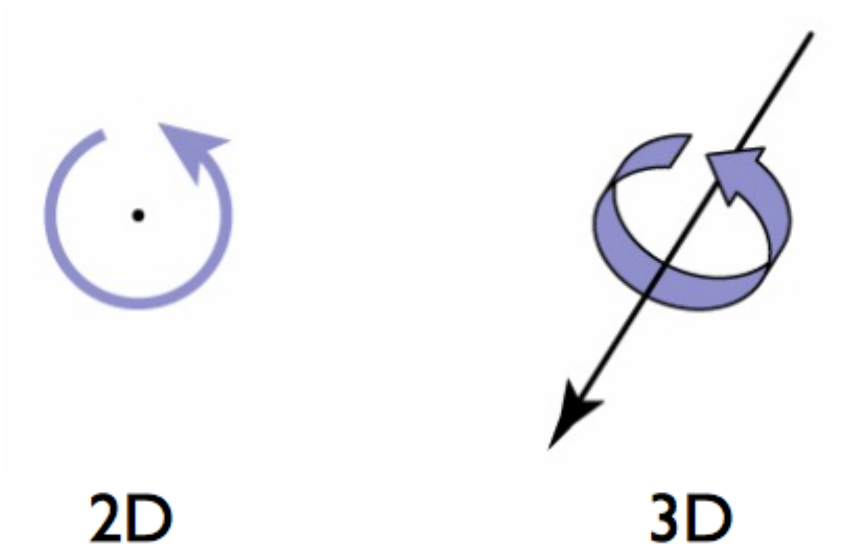
- Sentido antihorário, sob a origem, ângulo θ

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Matrizes de rotação

- Uma rotação em 2D é realizada ao redor de um ponto
- Uma rotação em 3D é realizada ao redor de um eixo
  - Rotações em 3D são realizadas em torno de retas (eixos), não simplesmente pontos
  - Muito mais possibilidades que em 2D



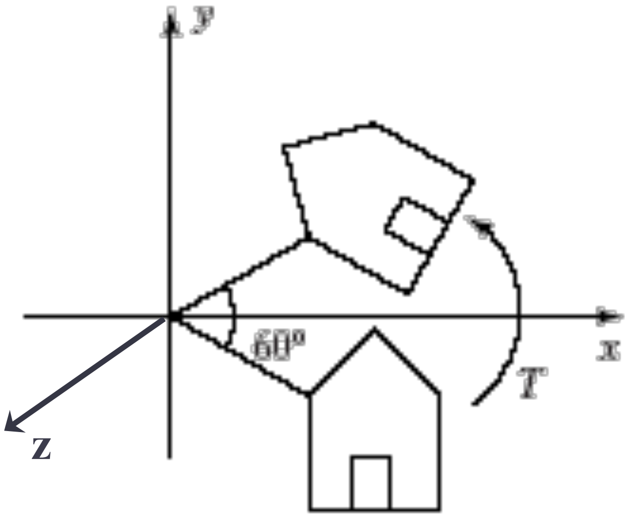
Rotação em 3D

- Rotação sob o eixo z em 3D mantem a coordenada z dos pontos
- Equivalente a uma rotação em planos onde z é constante

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

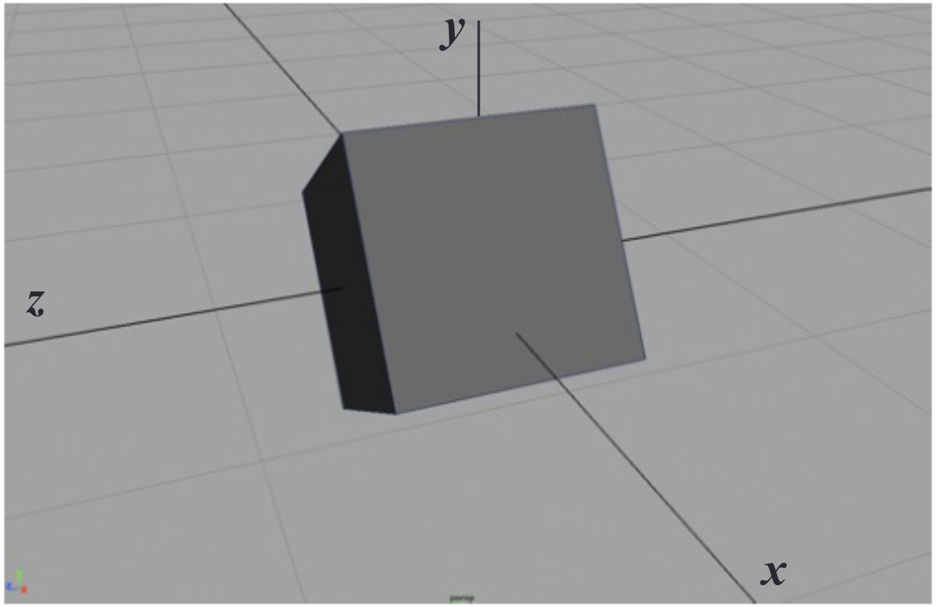
- Em coordenadas homogêneas:

$$p' = R_z(\theta)p$$



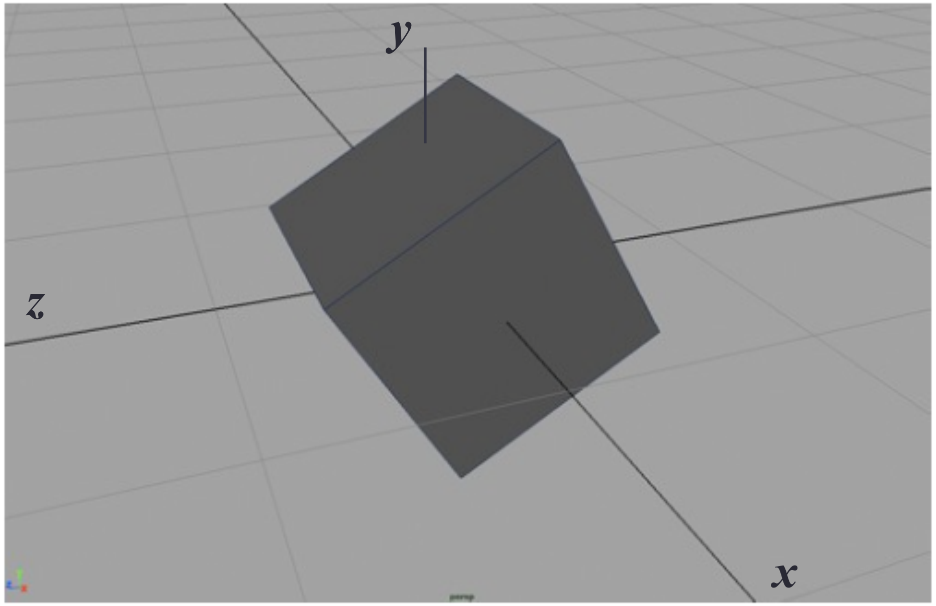
Matriz de rotação no eixo z

$$\mathbf{R} = \mathbf{R}_z(\theta) = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



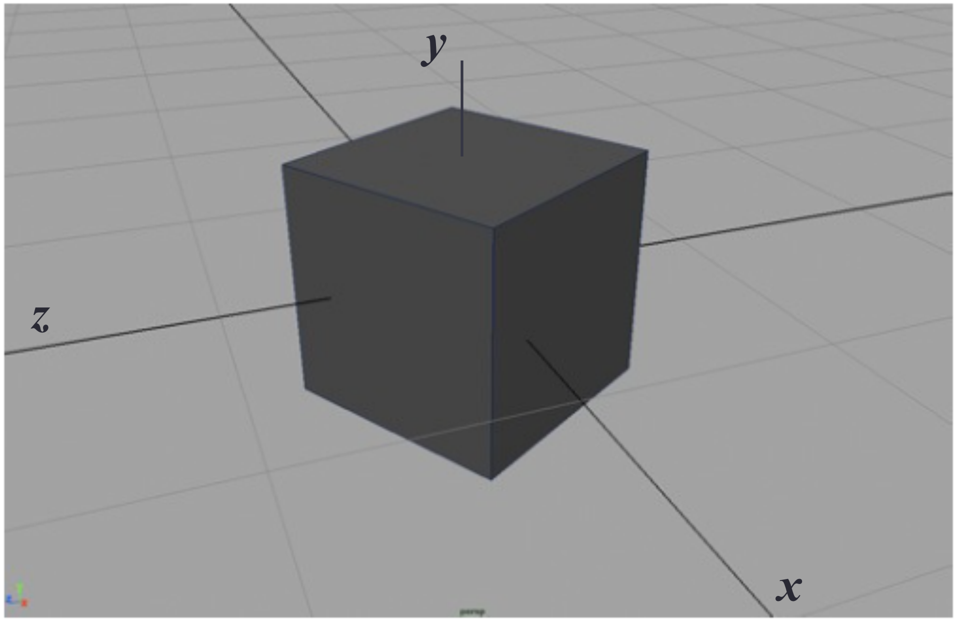
Matriz de rotação no eixo x

$$\mathbf{R} = \mathbf{R_x}(\theta) = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Matriz de rotação no eixo y

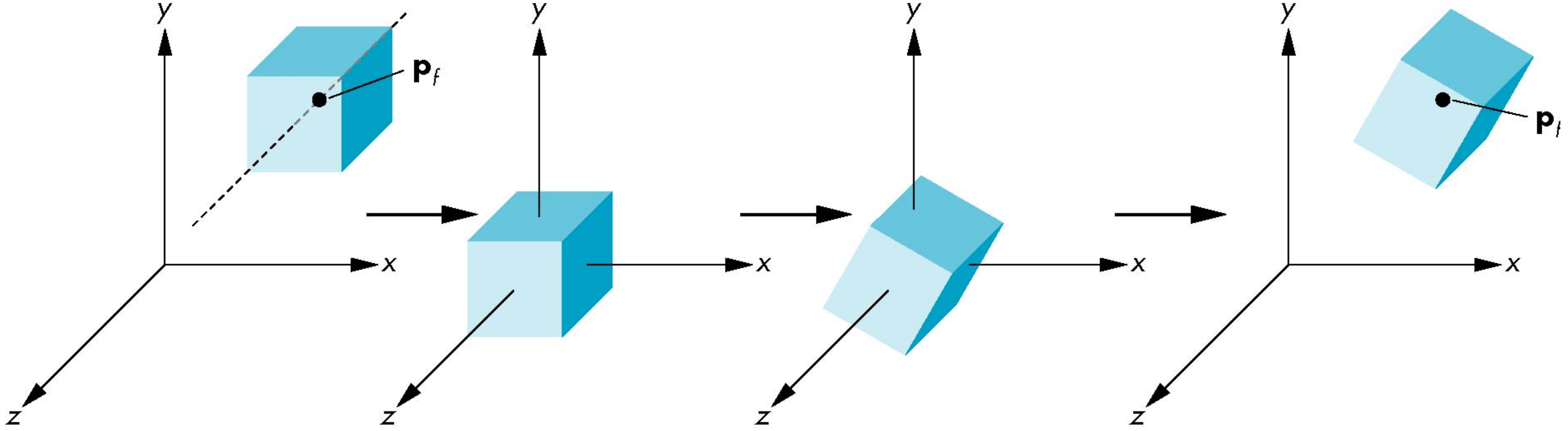
$$\mathbf{R} = \mathbf{R_y}(\theta) = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Rotação em torno de ponto arbitrário

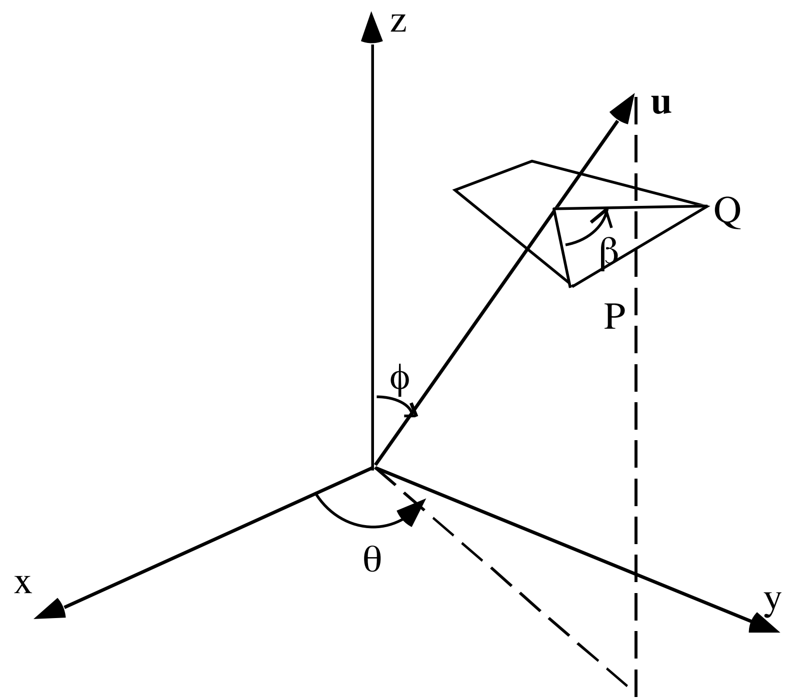
- Move-se p_f para a origem
- Rotaciona
- Move-se p_f para a sua localização original

$$\mathbf{M = T(p_f)\ R(\theta)\ T(-p_f)}$$



Rotação em torno de um eixo arbitrário

- Desejamos rotacionar u para transformar P em Q
- Decompor a transformação em passos:
  - Duas rotações para alinhar u com o eixo z.
  - Rotacionar β sob o eixo z
  - Desfazer as duas primeiras rotações.



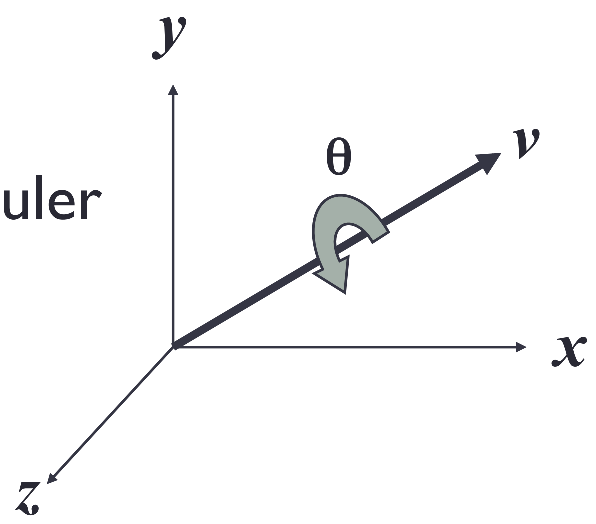
$$R_u(\beta) = R_z(\theta)\, R_y(\Phi)\, R_z(\beta)\, R_y(-\Phi)\, R_z(-\theta)$$

Rotações arbitrárias

- Qualquer rotação em 3D em torno de um eixo (que passa pela origem) pode ser obtido através do produto de 5 matrizes com ângulos de Euler apropriados.
- Isso implica que somente três valores são necessários para especificar qualquer rotação!

$$\mathbf{R}(\theta) = \mathbf{R}_z(\theta_z)\,\mathbf{R}_y(\theta_y)\,\mathbf{R}_x(\theta_x)$$

$\theta_x\ \theta_y\ \theta_z$ são chamados de ângulos de Euler

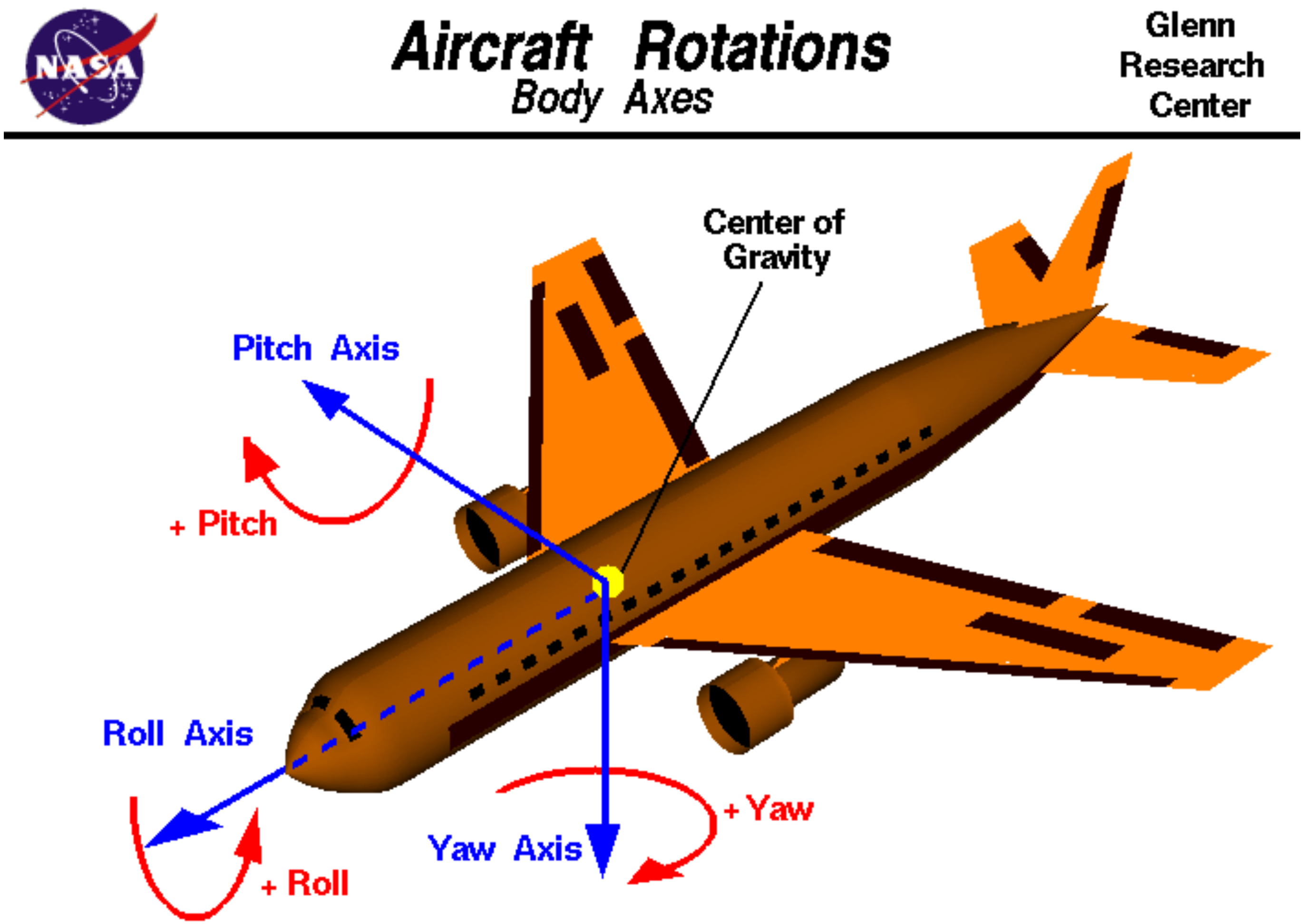


Note that rotations do not commute
We can use rotations in another order but with different angles

Ângulos de Euler

- Um objeto pode ser orientado arbitrariamente
- Ângulos de Euler acumulam 3 rotações nos eixos de coordenadas
- Recebem diferentes nomes dependendo do contexto de aplicação
  - "Heading, elevation, bank": aeronaves
  - "Yaw, pitch, roll": aeronaves
  - "Pan, tilt, roll": câmeras

Roll, pitch, and yaw



Rotações com Euler

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)$$

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

$$c_i = \cos(\theta_i)$$

$$s_i = \sin(\theta_i)$$

Inversas

- Podemos calcular inversas das matrizes de rotação através de fórmulas genéricas, mas observamos que
  - Rotação: $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$
- Satisfeita por qualquer matriz de rotação:

$$\cos(-\theta) = \cos(\theta)$$

$$\sin(-\theta) = -\sin(\theta)$$

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

Propriedades

- Translações: parte linear é a matriz identidade
- Escalas: parte linear é uma matriz diagonal
- Rotações: parte linear é ortogonal
  - A transposta é igual a sua inversa
  - As colunas de R são mutualmente ortonormais: $RR^T=R^TR=I$
  - E o determinante de R é 1.0 [det(R) = 1]

Parte linear **Parte afim**

$$T = \begin{bmatrix} a & b & c & j \\ d & e & f & k \\ g & h & i & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composição

- Podemos criar matrizes de transformação arbitrárias multiplicando matrizes de rotação, translação e escala
- Já que a mesma transformação é aplicada para muitos vértices, o custo de formar uma matriz $\mathbf{M}=\mathbf{ABCD}$ é insignificante comparado ao custo de calcular $\mathbf{Mp}$ para muitos vertices $\mathbf{p}$
- A parte mais difícil é como formar uma transformação a partir das especificações da aplicação

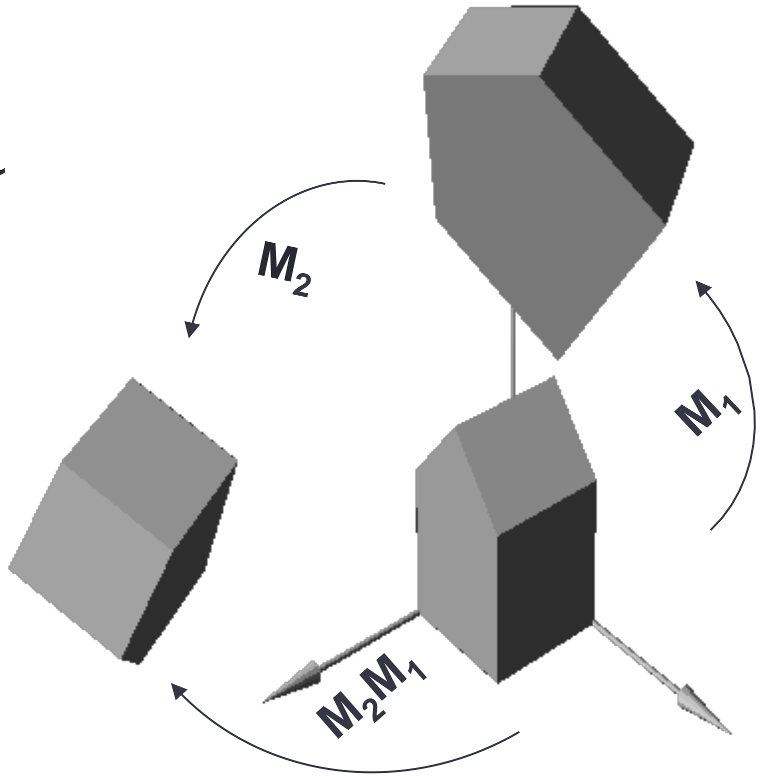
Ordem das transformações

- Note que a matriz mais à direita é a primeira a ser aplicada
- Todas relações abaixo são matematicamente equivalentes:

$$p' = ABCp = (AB)(Cp) = A(BC)p = A(B(Cp))$$
$$p' = Mp$$
$$M = ABC$$

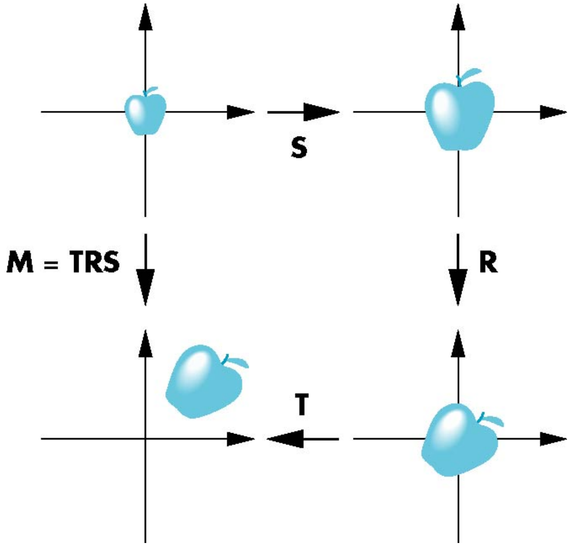
Exemplo

- A casa é primeiramente transformada por $\mathbf{M}_1$, e novamente transformada por $\mathbf{M}_2$.
- O resultado é o mesmo se transformada uma única vez por $\mathbf{M}_2\mathbf{M}_1$.



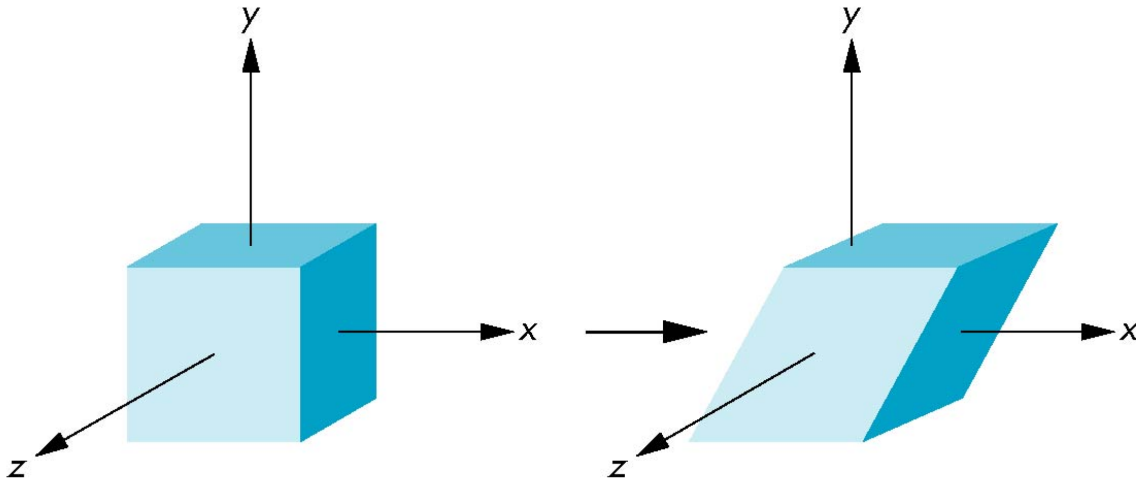
Instanciamento

- Na fase de modelagem, é comum desenharmos com um único objeto, centralizado na origem, orientado nos eixos canônicos, de tamanho unitário
- Aplicaremos uma transformação de instanciamento para
  - Redimensionar
  - Orientar
  - Posicionar



Cisalhamento

- Útil na confecção de mais uma transformação básica
- Equivalente à forças aplicadas em direções opostas

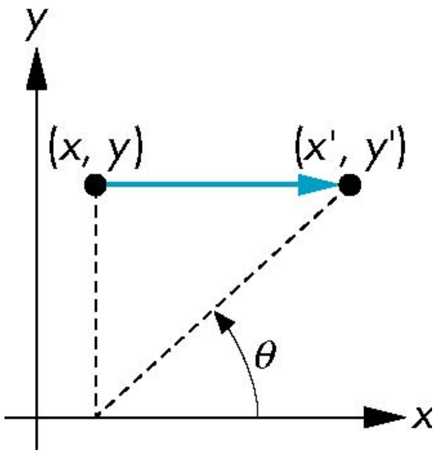


Matriz de cisalhamento

- Considere um cisalhamento ao longo do eixo x

$$x' = x + y \cot\theta$$
$$y' = y$$
$$z' = z$$

$$\mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

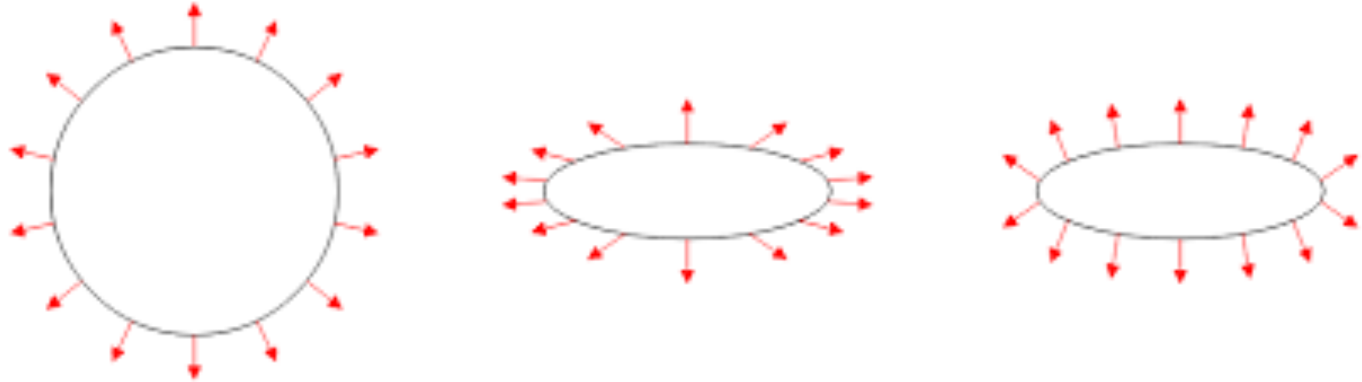


Propriedades

- Translações não alteram a área ou volume.
- Para uma escala pura, a nova área é $S_x S_y$ vezes a área original
- Para um cisalhamento em um único eixo, a nova área é a mesma que área original.
- Em 3D, o resultado é análogo.

Transformações de vetores normais

- Transforming surface normals
  - differences of points (and therefore tangents) transform OK
  - normals do not --> use inverse transpose matrix



have: $\mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$

want: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T X \mathbf{n} = 0$

so set $X = (M^T)^{-1}$

then: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$

Transformações em 3D

- Vetores e pontos em 3D
- Transformação linear afim

$$\vec{V} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix} \qquad P = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \qquad T = \begin{bmatrix} a & b & c & j \\ d & e & f & k \\ g & h & i & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformações rígidas

- Não modificam a forma (dimensões / ângulos) do objeto
- São compostas de uma rotação e uma translação

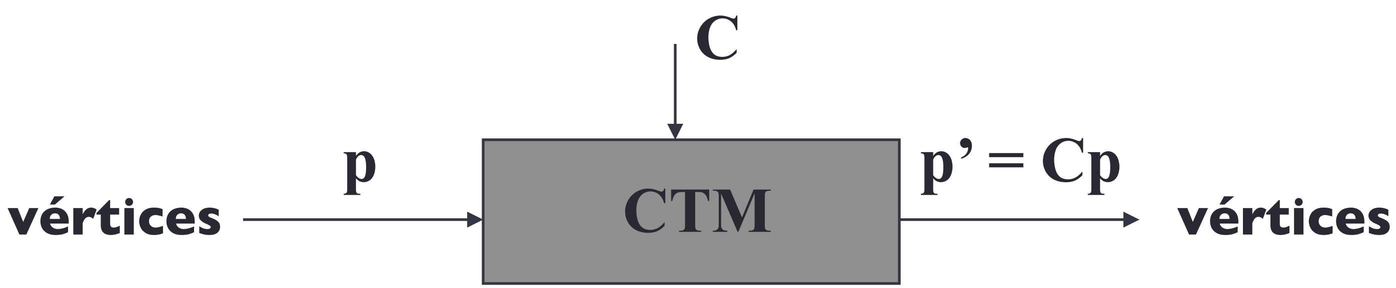
$$T = \begin{bmatrix} \overbrace{\boxed{\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}}}^{\textbf{Rotação}} & \overbrace{\boxed{\begin{matrix} j \\ k \\ l \end{matrix}}}^{\textbf{Translação}} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

Transformações a partir de pontos

- Transformações afins em 2D possuem 6 graus de liberdade (DOFs)
  - Triângulos mapeiam em triângulos
- Transformações afins em 3D possuem 12 graus de liberdade
  - Tetraedros mapeiam em tetraedros

Matriz de transformação

- Conceptualmente existe uma matriz de transformação corrente (CTM) que utilizamos para aplicar em todos os vértices
- Ela é normalmente definida na aplicação e passada para os shaders



Matriz model-view

- In WebGL, the model-view matrix is used to
  - Position the camera
  - Can be done by rotations and translations but is often easier to use the `lookAt()` function in `MV.js`
- The projection matrix is used to define the view volume and to select a camera lens
- Although these matrices are no longer part of the OpenGL state, it is usually a good strategy to create them in our own applications

$$\mathbf{q} = \mathbf{P}\,\mathbf{M}_V\,\mathbf{p}$$

Matrizes arbitrárias

- Can load and multiply by matrices defined in the application program
- Matrices are stored as one dimensional array of 16 elements by MV.js but can be treated as 4 x 4 matrices in row major order
- OpenGL wants column major data
  - **gl.uniformMatrix4f** has a parameter for automatic transpose.
  - **flatten** function converts to column major order which is required by WebGL functions

Pilhas de matrizes

- In many situations we want to save transformation matrices for use later
  - Traversing hierarchical data structures
- Pre 3.1 OpenGL maintained stacks for each type of matrix
- Easy to create the same functionality in JS
  - push and pop are part of Array object

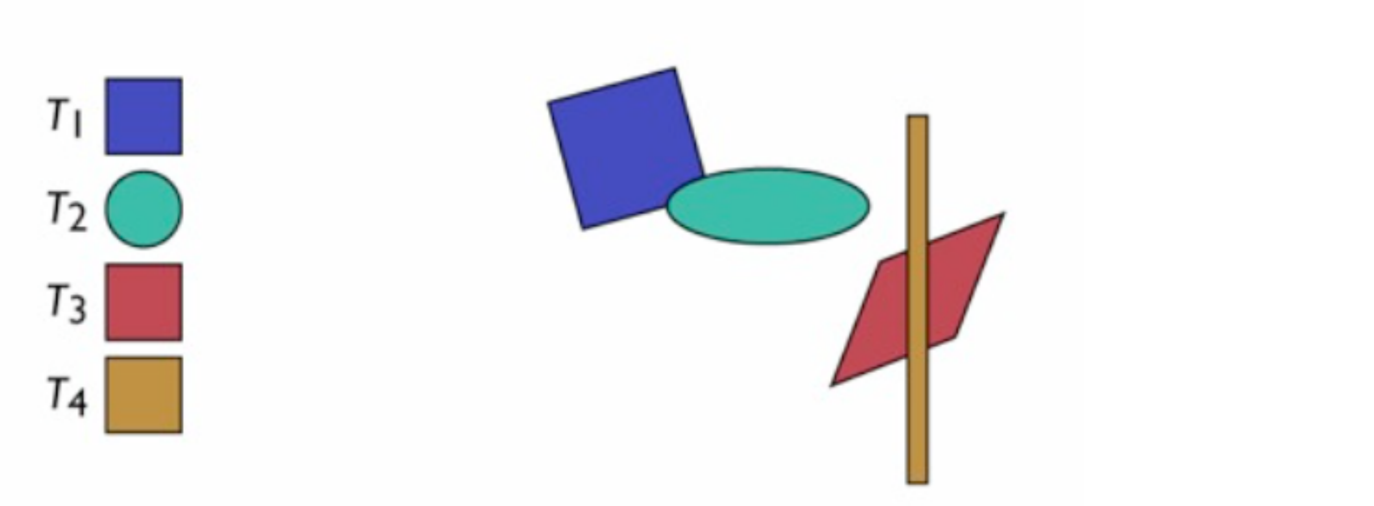
```
var stack = [ ]
stack.push(modelViewMatrix);
modelViewMatrix = stack.pop();
```

Onde aplicaremos as transformações ?

- Where ?
  - in application to vertices
  - in vertex shader: send MV matrix
  - in vertex shader: send angles
- Choice between second and third unclear
- Do we do trigonometry once in CPU or for every vertex in shader?
  - GPUs have trig functions hardwired in silicon

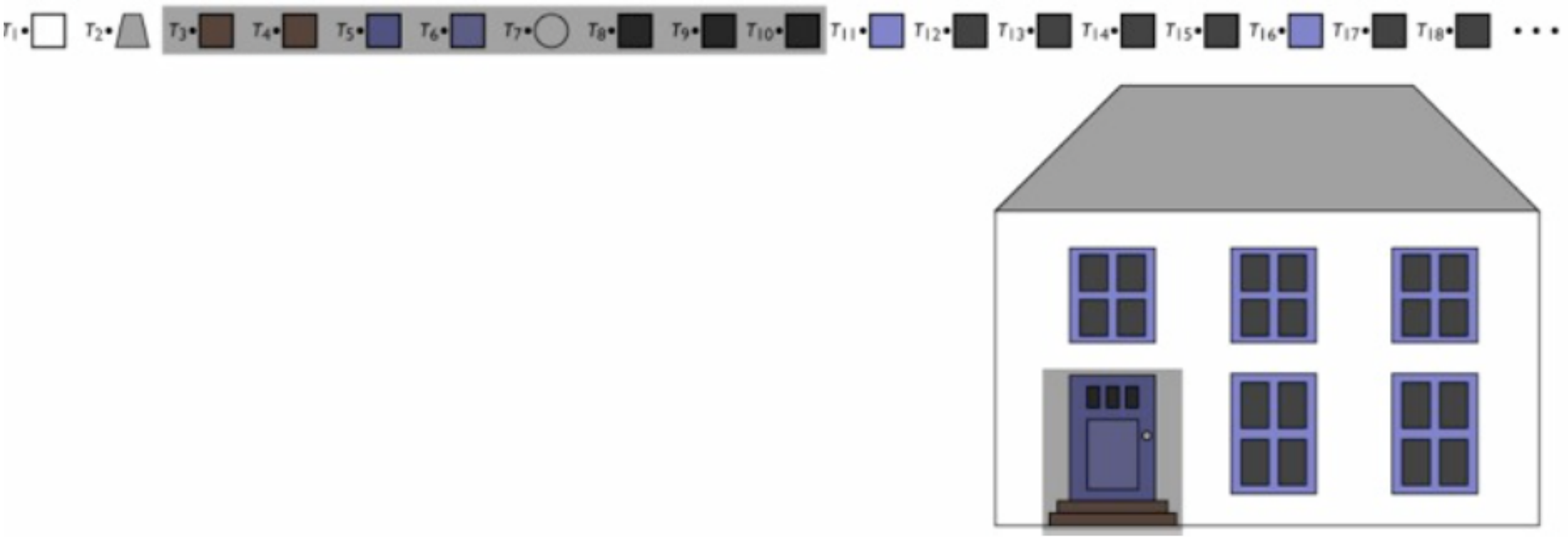
Estruturas de dados

- Representing a drawing (“scene”)
- List of objects
- Transform for each object
  - can use minimal primitives: ellipse is transformed circle
  - transform applies to points of object



Exemplo

- Can represent drawing with flat list
  - but editing operations require updating many transforms

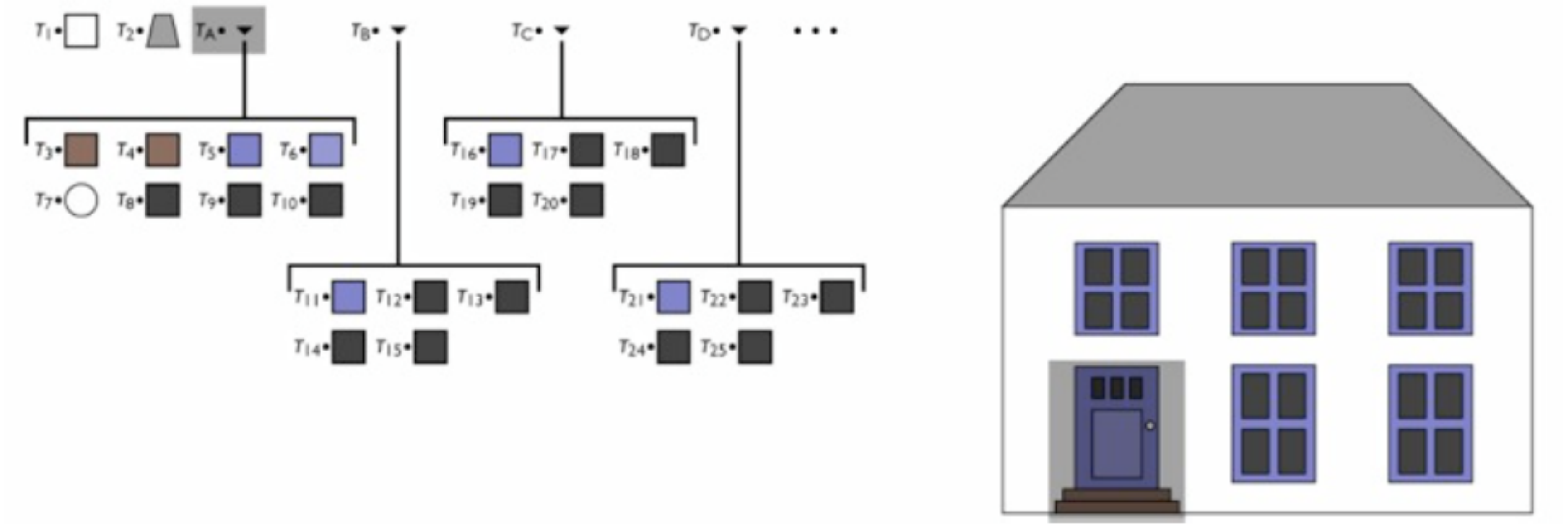


Grupos de objetos

- Treat a set of objects as one
- Introduce new object type: group
  - contains list of references to member objects
- This makes the model into a tree
  - interior nodes = groups
  - leaf nodes = objects
  - edges = membership of object in group

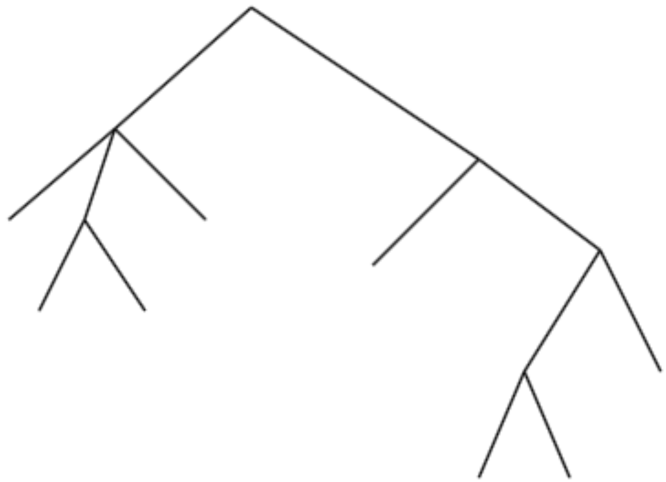
Exemplo

- Add group as a new object type
  - lets the data structure reflect the drawing structure
  - enables high-level editing by changing just one node



"Scene graph"

- A name given to various kinds of graph structures (nodes connected together) used to represent scenes
- Simplest form: tree
  - just saw this
  - every node has one parent
  - leaf nodes are identified with objects in the scene



Tarefa

- Leitura livro-texto
  - Shirley and Marschner. Fundamentals of Computer Graphics, CRC Press, 3rd Ed. 2010
  - Capítulo 6
- Exercício-programa 1
- Prova #1(10/4)
  - Capítulos 1-6