

Computação Gráfica

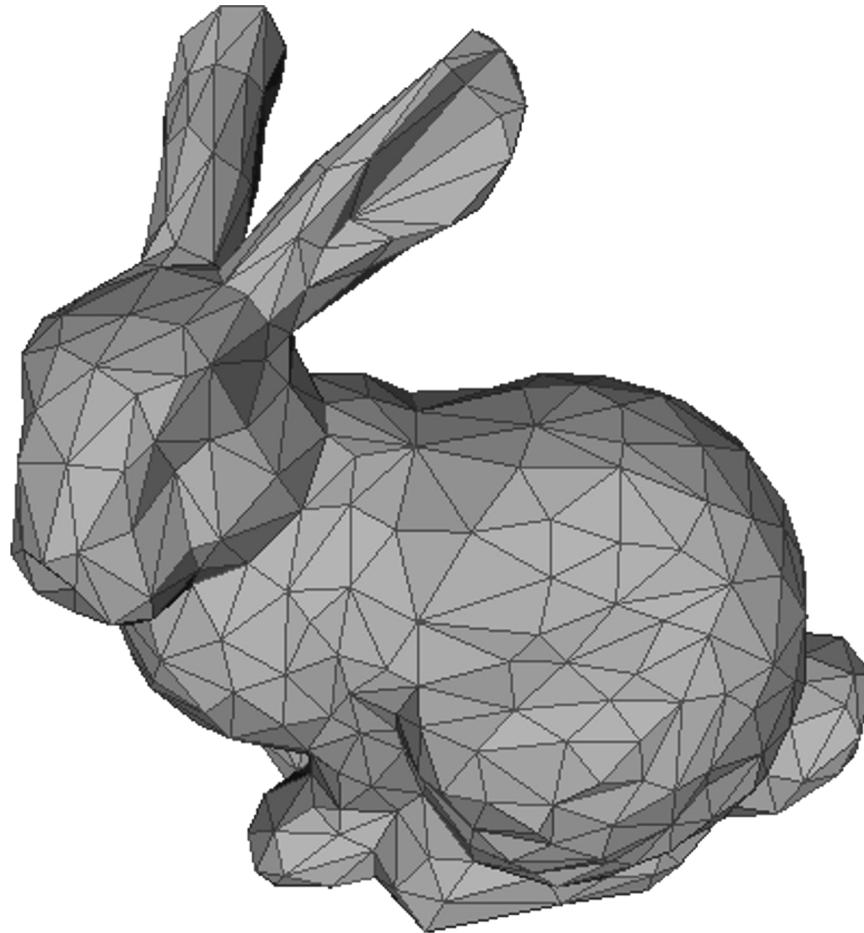
Aula 8: Geometrias e Grafo de Cena

Computação Gráfica

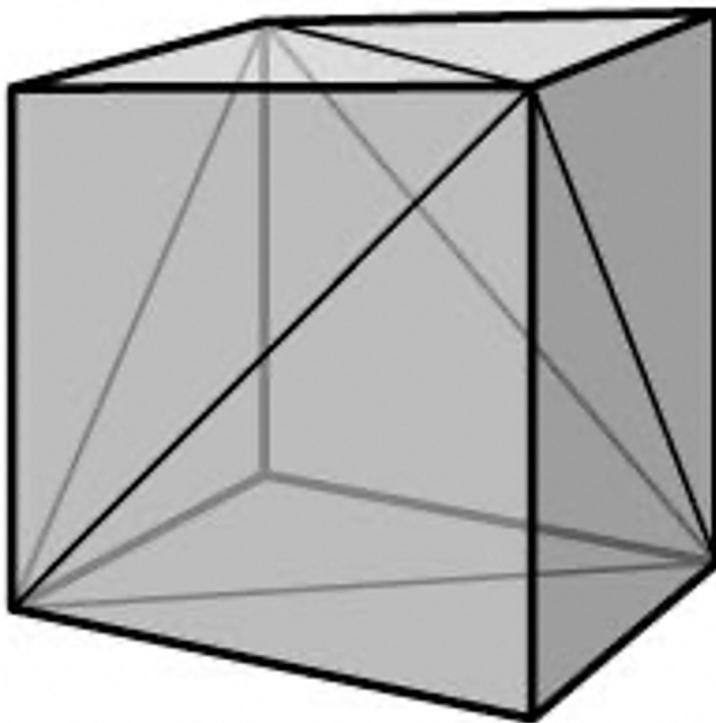
Malhas Poligonais



Malha de Triângulos



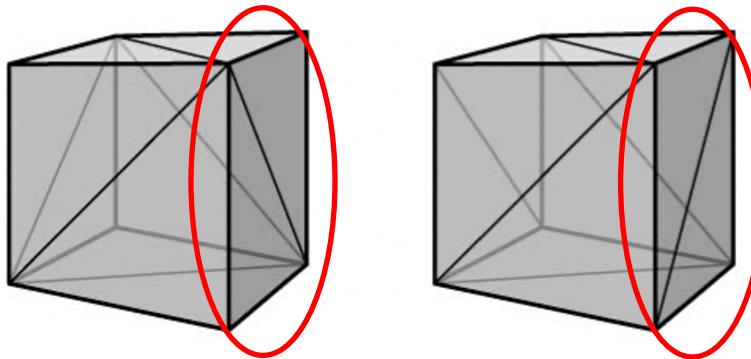
Pequena Malha de Triângulos



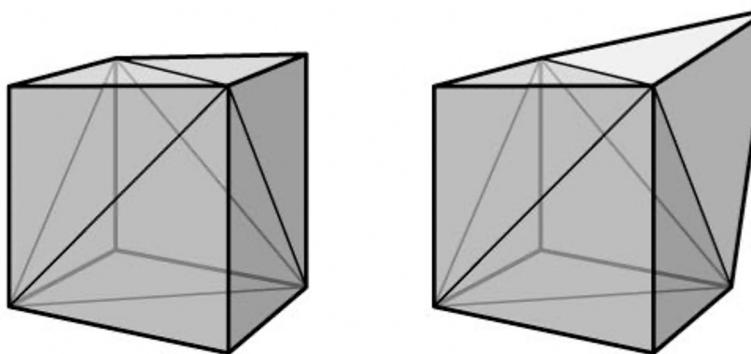
8 vértices, 12 triângulos

Topologia versus Geometria

Mesma geometria, diferente topologia da malha

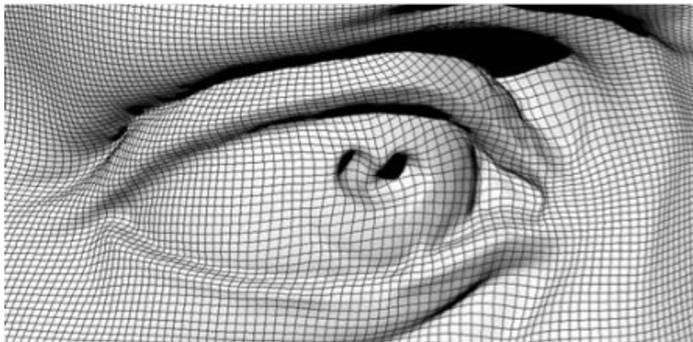
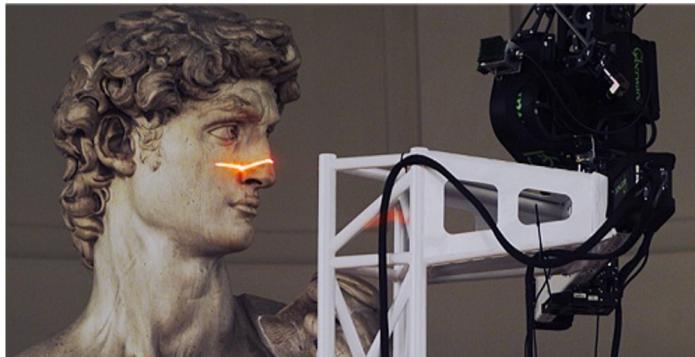


Mesma topologia, diferente geometria



Grande Malha de Trângulos

David
Digital Michelangelo Project
28.184.526 vértices
56.230.343 triângulos



Malha Gigantesca de Triângulos

Google Earth

Malha reconstruída de imagens de satélite e aéreas

Trilhões de triângulos



Data SIO, NOAA, U.S. Navy, NGA, GEBCO

Data LDEO-Columbia, NSF, NOAA

Data CSUMB SFML, CA OPC

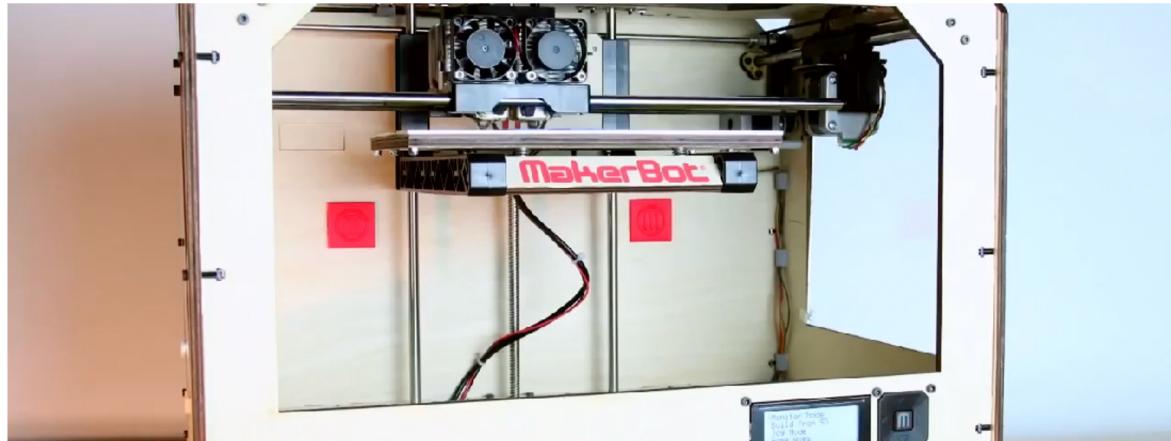
Data MBARI

Google earth

Processamento Digital de Geometria



Scanner 3D



Impressão 3D

Pipeline de Processamento Geométrico



Escaneamento

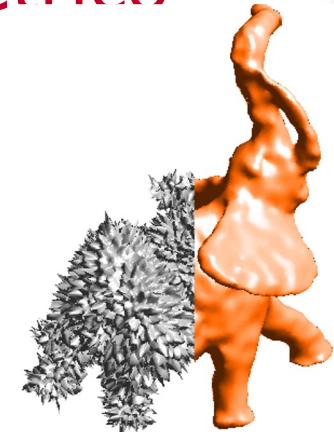
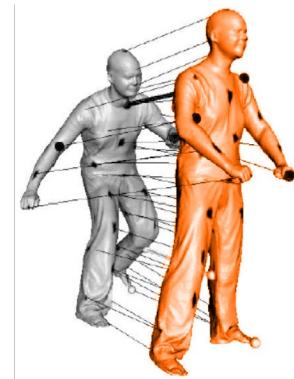
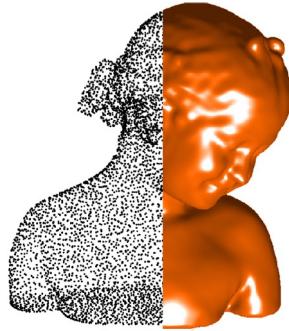


Processamento



Impressão

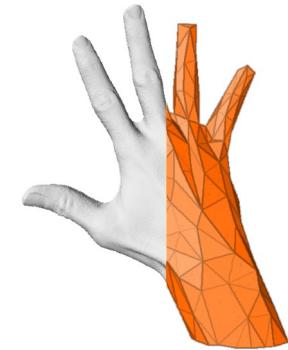
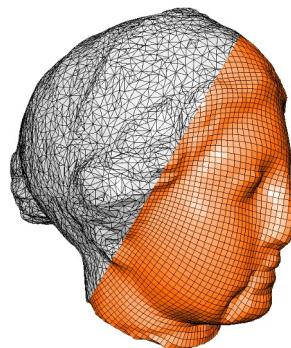
Mais Tarefas de Processamento Geométrico



Reconstrução 3D

Analise de Formas

Filtragem



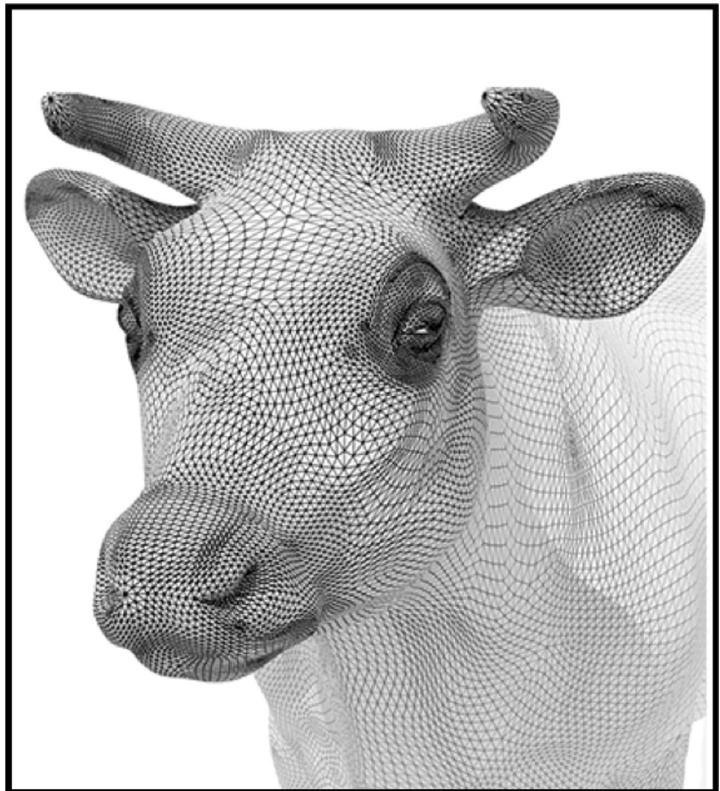
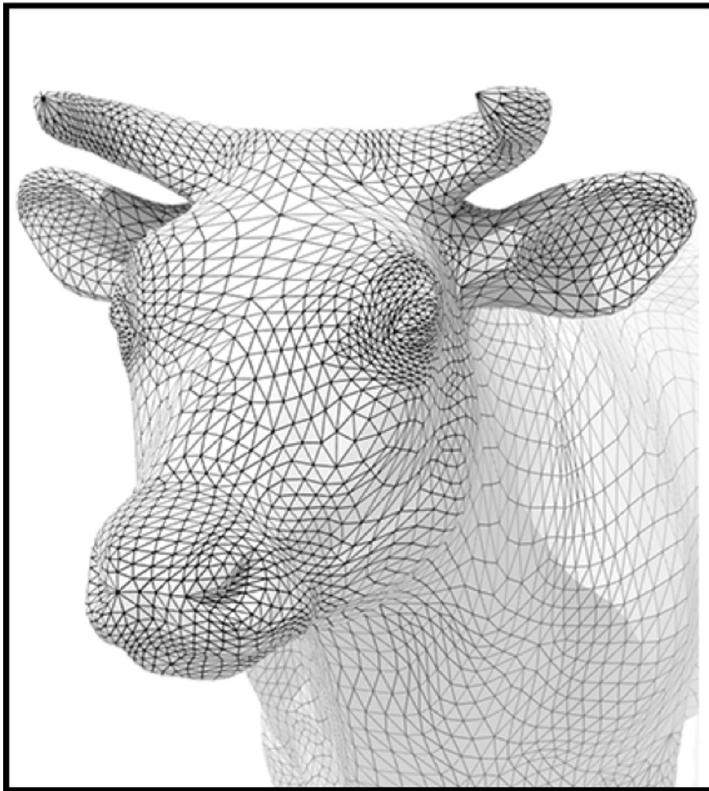
Remeshing

Parametrização

Compressão

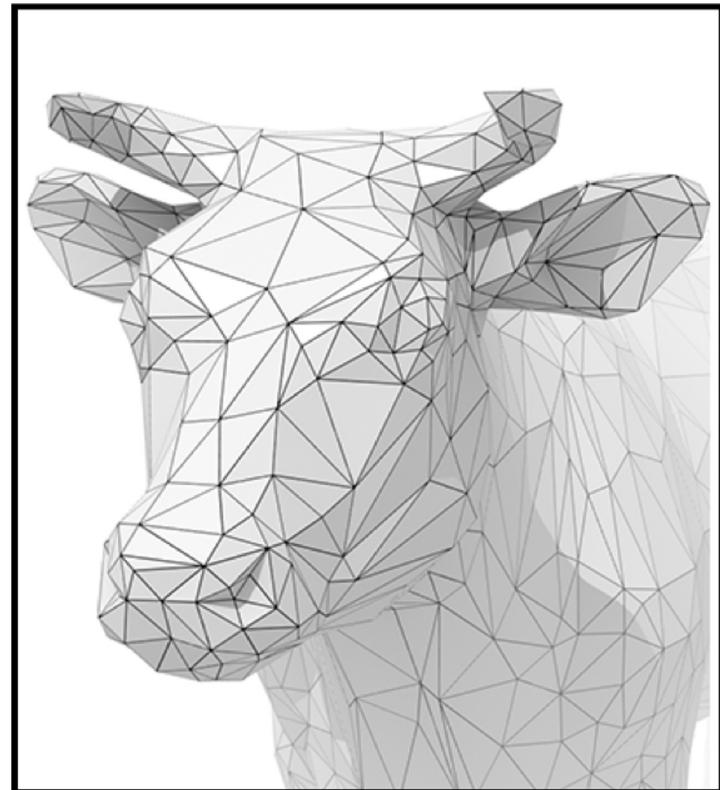
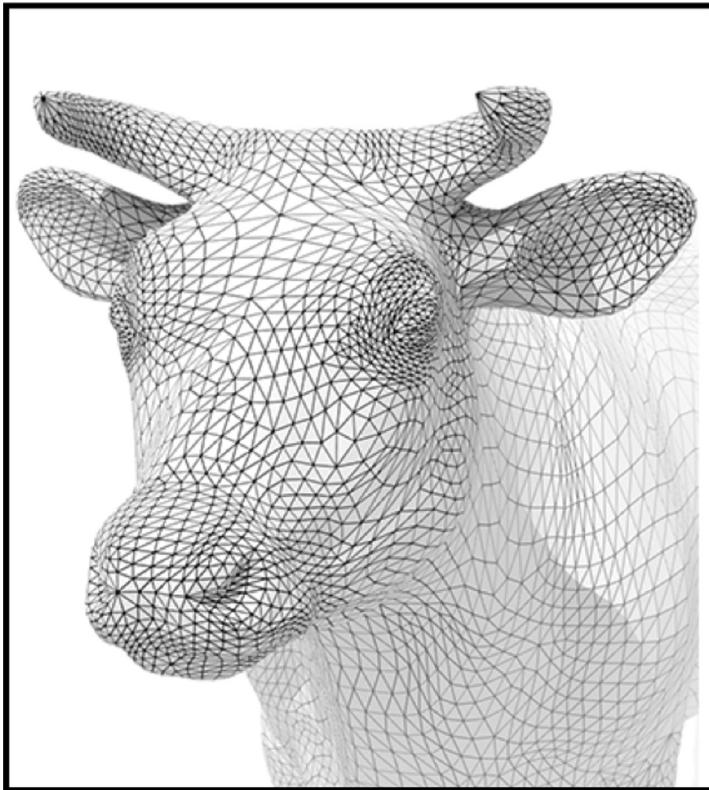
Insper

Upsampling (Refinando Malha)



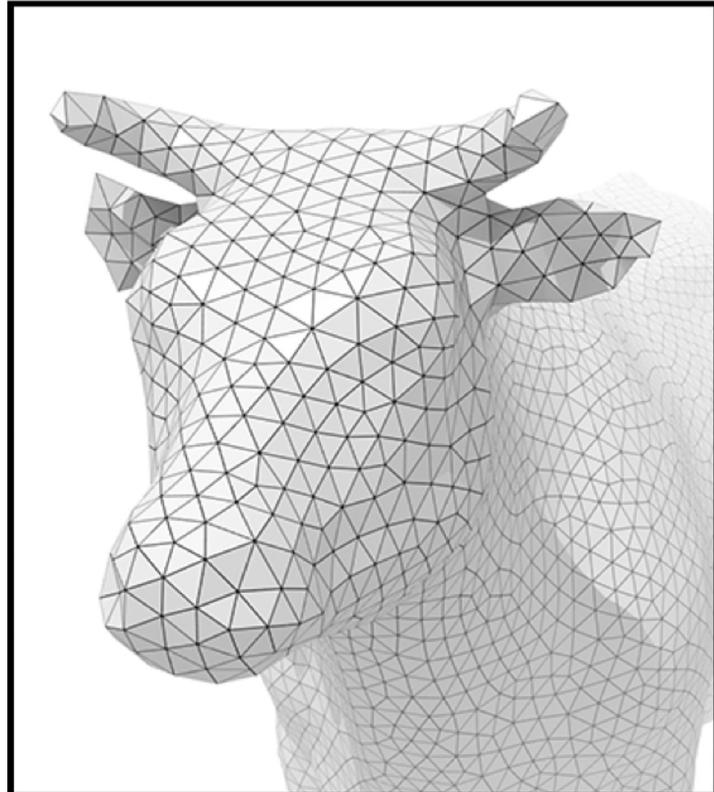
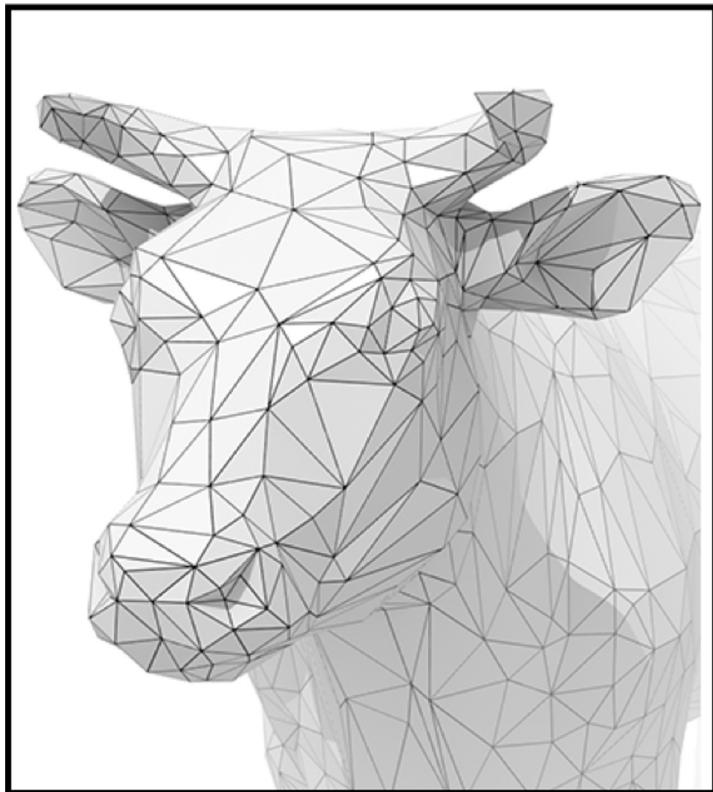
Subdivisão : melhorando resolução interpolando os pontos

Downsampling (Simplificando a Malha)



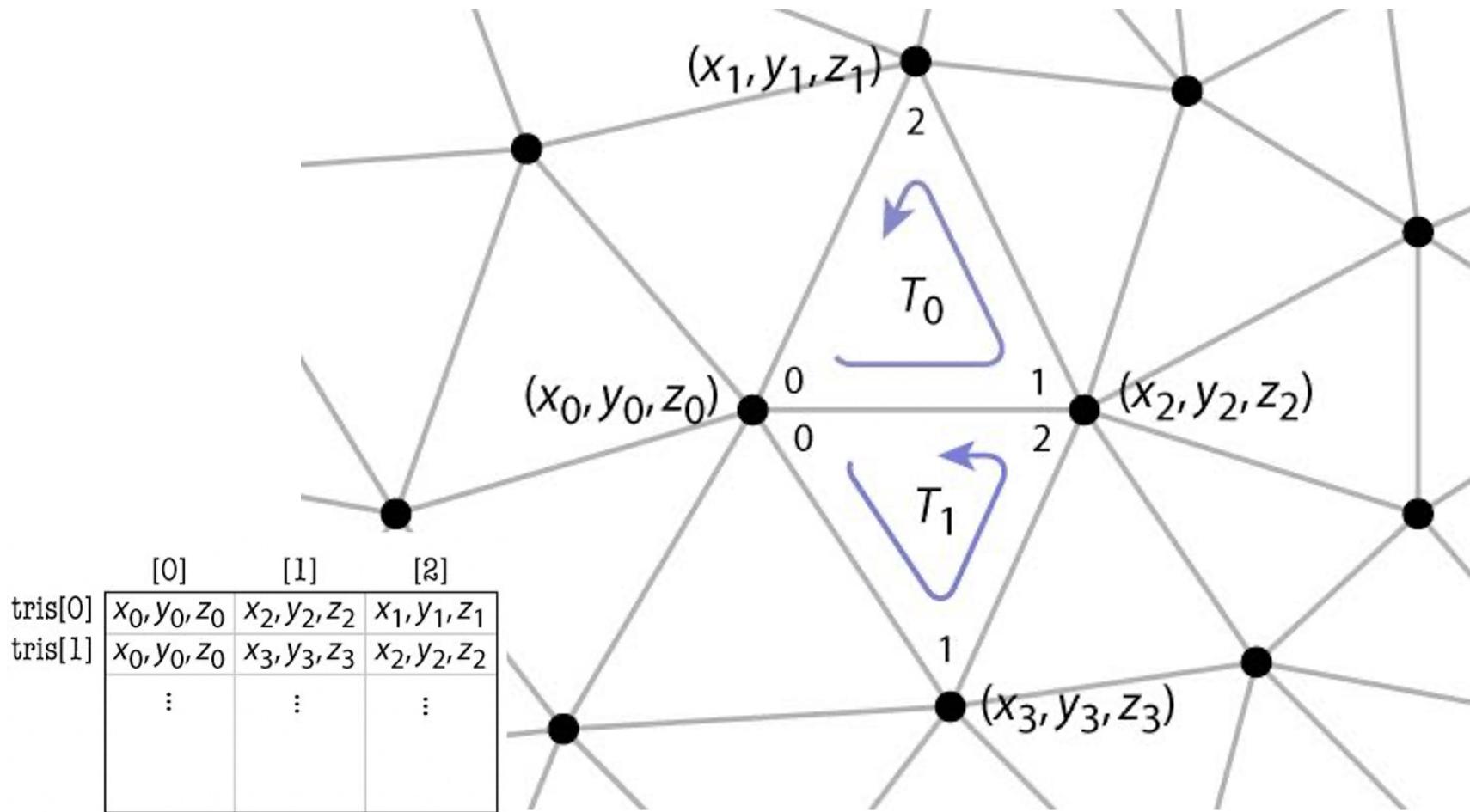
Diminuindo Pontos : tentando manter a forma original

Homogeneizando Malha



Ajustando Pontos para melhorar qualidade (possivelmente)

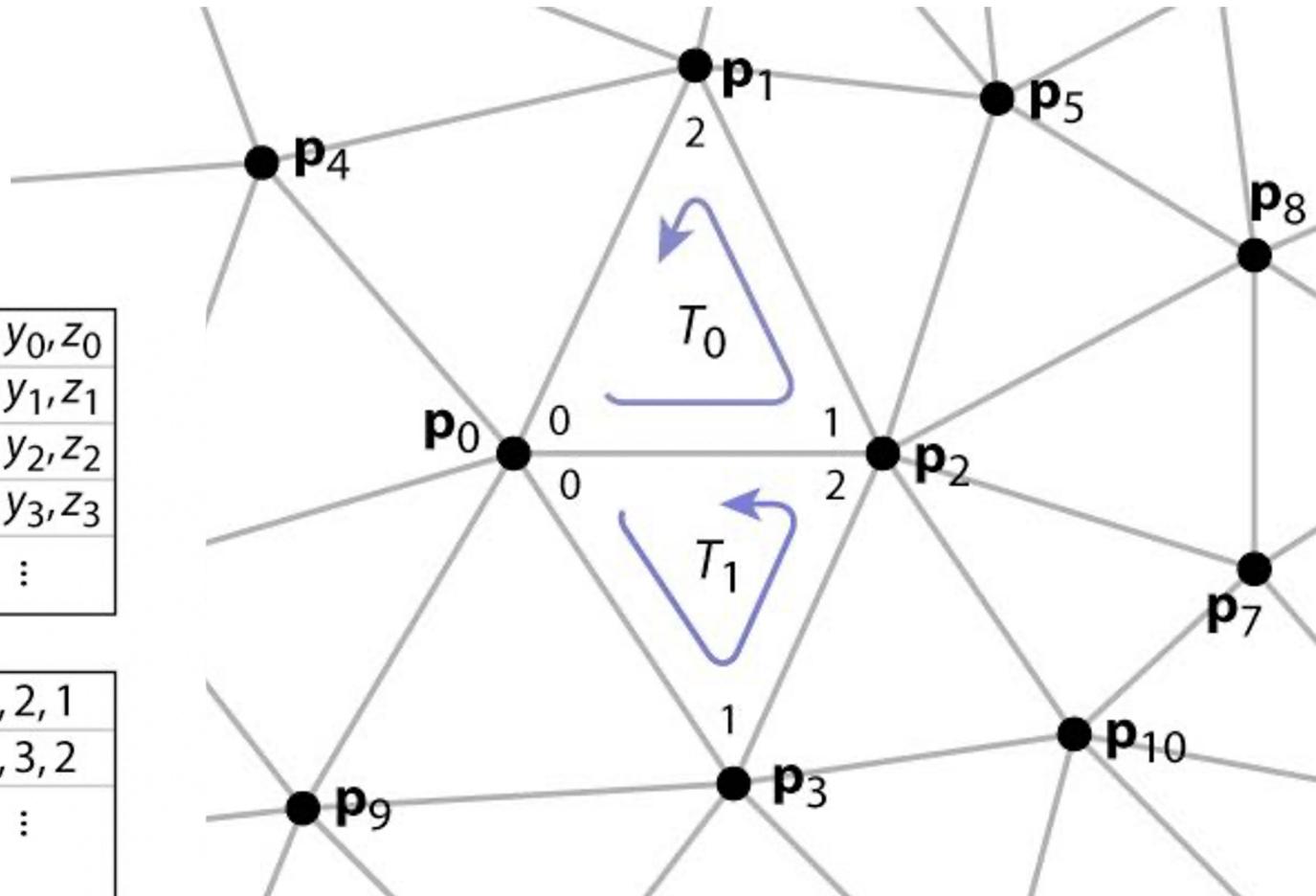
Lista de Triângulos



Lista de Pontos e sua conexão por índices

verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
	x_2, y_2, z_2
	x_3, y_3, z_3
:	

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
:	



Comparação

Triângulos

- + Simples
- Muita Informação Redundante

Pontos e Triângulos

- + Compartilhamento de vértices reduz consumo de memória
- + Garante integridade da malha
 - (alterar um vértice, altera para todos os polígonos)

Informação da Topologia da Malha

Aplicações:

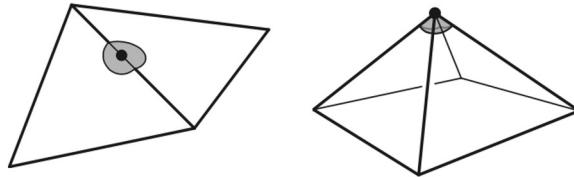
- Acesso aos vizinhos em tempo constante
ex: cálculo de normais da superfície, subdivisões
- Edição da Geometria
ex: adicionando/removendo vértices, faces, arestas, etc

Solução: Estrutura de Dados Topológicas

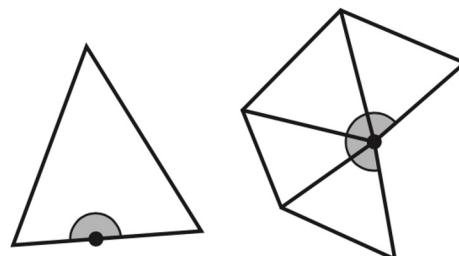
Validade Topológica: Manifold

Definição: um manifold 2D é uma superfície que, quando cortada em qualquer ponto com uma pequena esfera, sempre produz um (e somente um) disco contínuo (ou meio disco nas bordas).

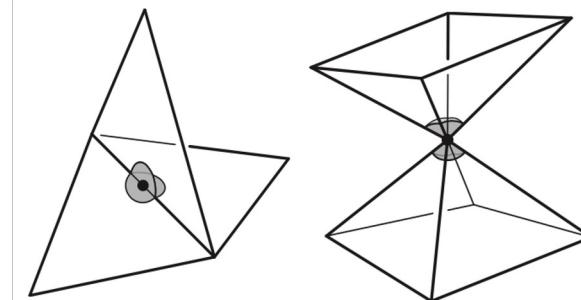
Manifold



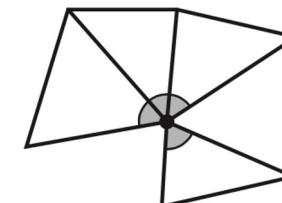
nas bordas



Não Manifold



nas bordas



Validade Topológica: Manifold

Definição: um manifold 2D é uma superfície que, quando cortada em qualquer ponto com uma pequena esfera, sempre produz um disco (ou meio disco nas bordas).

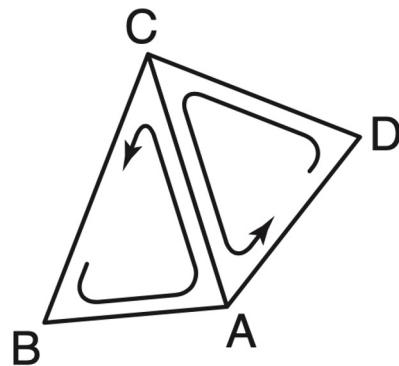
Se a malha é manifold, podemos contar com propriedades úteis:

- Uma aresta conecta com exatamente duas faces
- Uma aresta conecta com exatamente dois vértices
- Uma face consiste em um contorno de arestas e vértices
- Um vértice consiste em um contorno de arestas e faces

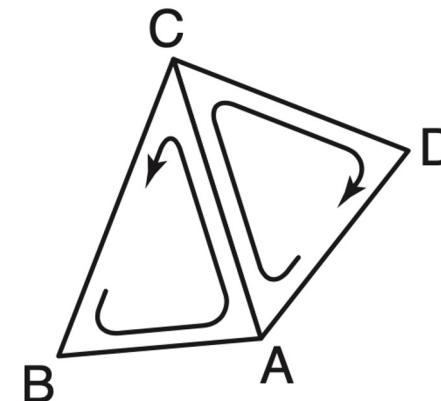
A fórmula do poliedro de Euler é válida: $\#f - \#a + \#v = 2$
(para uma superfície topologicamente equivalente a uma esfera)
(Verifique em um cubo: $6 - 12 + 8 = 2$)

Validade Topológica: Consistência da Orientação

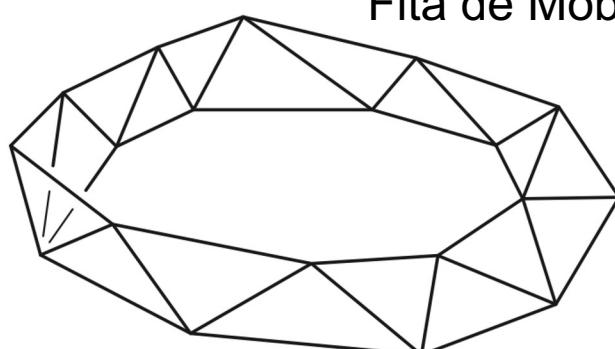
Orientação consistente



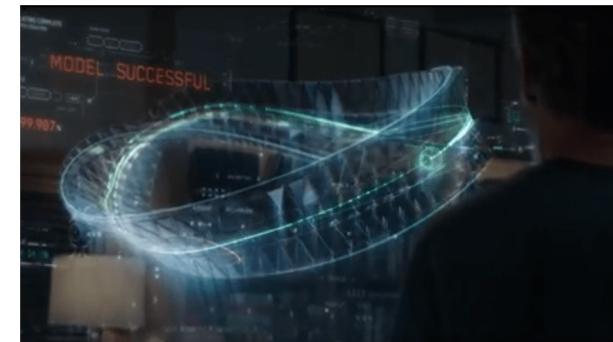
Orientação inconsistente



Não orientável



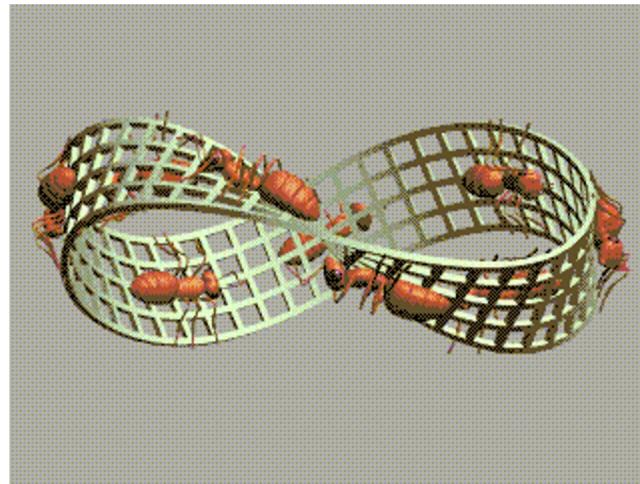
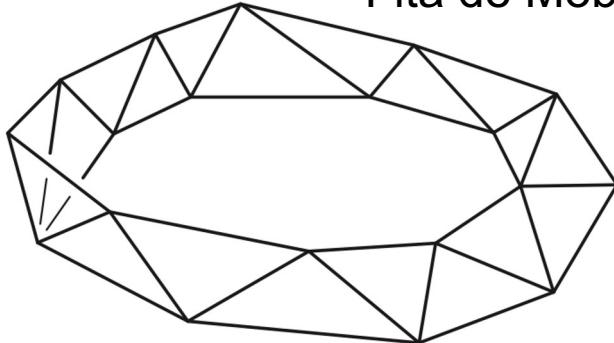
Fita de Möbius



Vingadores: Ultimato

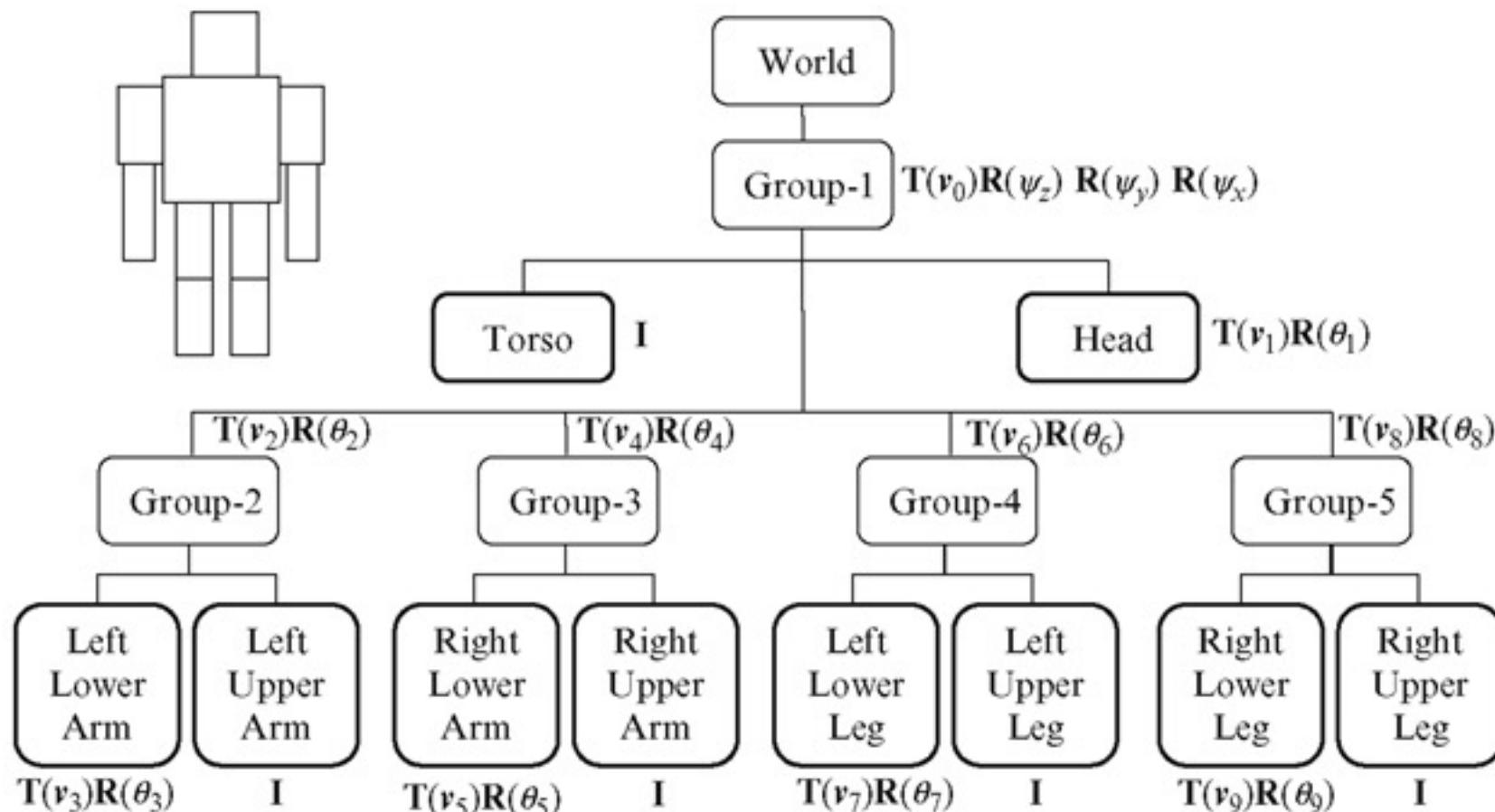
Validade Topológica: Consistência da Orientação

Fita de Möbius



M. C. Escher - Möbius Strip II (1963) *sper*

Grafo de Cena (Scene Graph)



Grafo de Cena (parece, mas não é uma árvore)

- Um Grafo Acíclico Dirigido "directed acyclic graph (DAG)"

Repetição:

- Uma cena pode conter várias cópias (instâncias) de um mesmo objeto
- O modelo (Objeto 3D) pode usar várias cópias de uma parte de outro modelo

Resultados

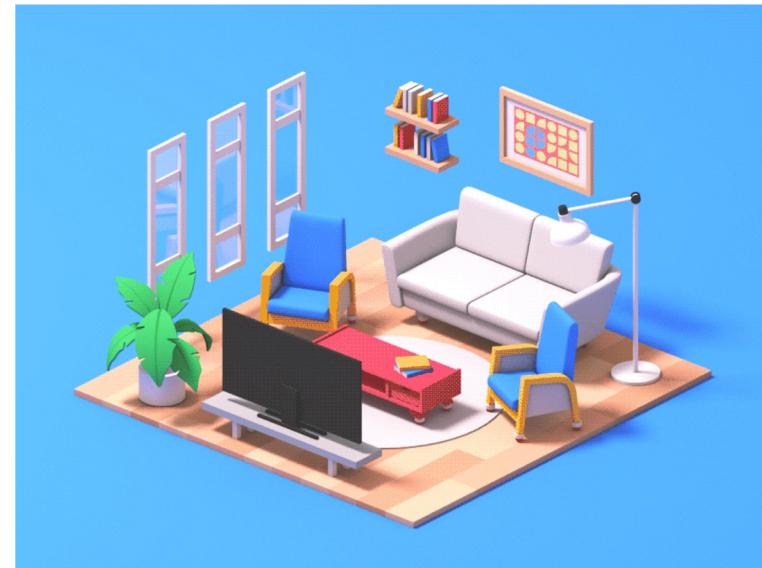
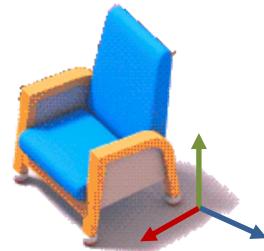
- Um nó (instâncias) pode possuir vários pais
- Ao se alterar uma cópia, essa se propaga para todas instâncias
- No *traversal* o mesmo objeto será desenhado várias vezes nas diferentes coordenadas

Economiza memória e tempo

Grafo de Cena

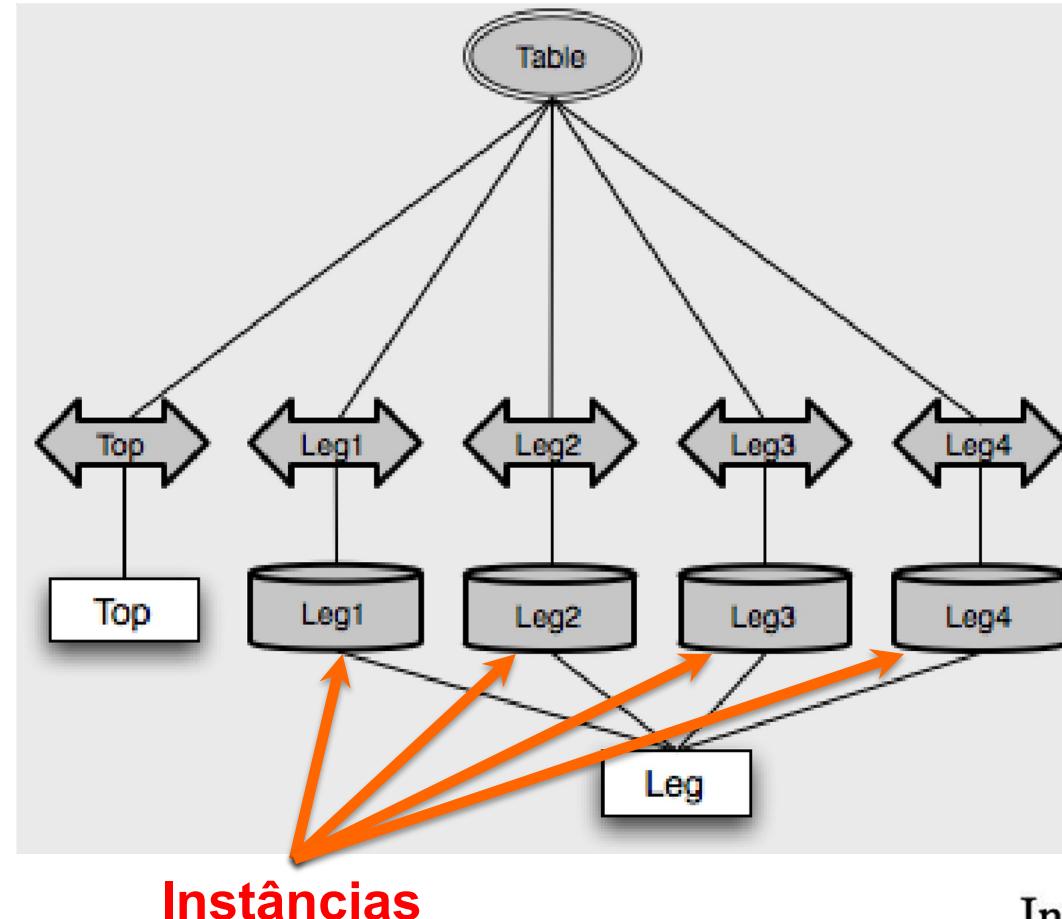
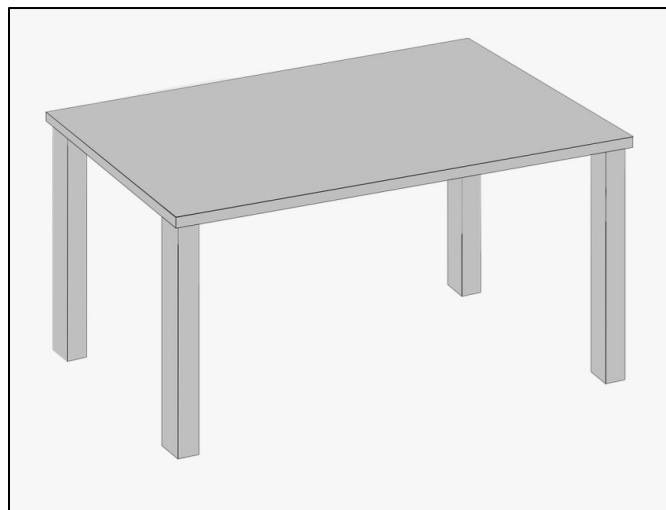
Vantagens:

- Permite definições de objetos nos seus sistemas de coordenadas
- Permite o uso de objetos várias vezes numa mesma cena
- Permite processamento pela hierarquia
- Permite animações de articulações de forma simples

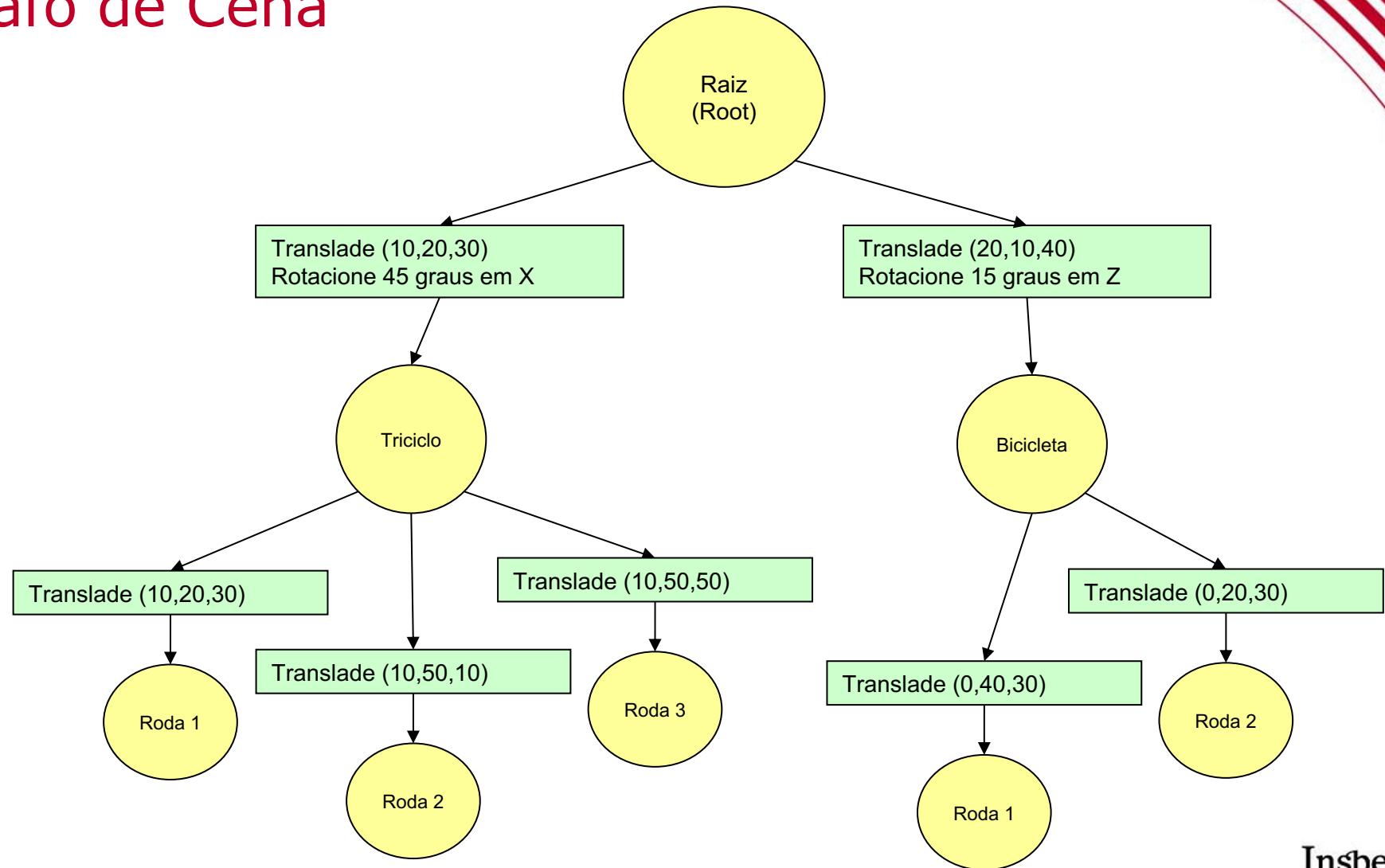


por: Guillaume Kurkdjian

Instanciação – partes do modelo



Grafo de Cena



Para que serve esta organização?

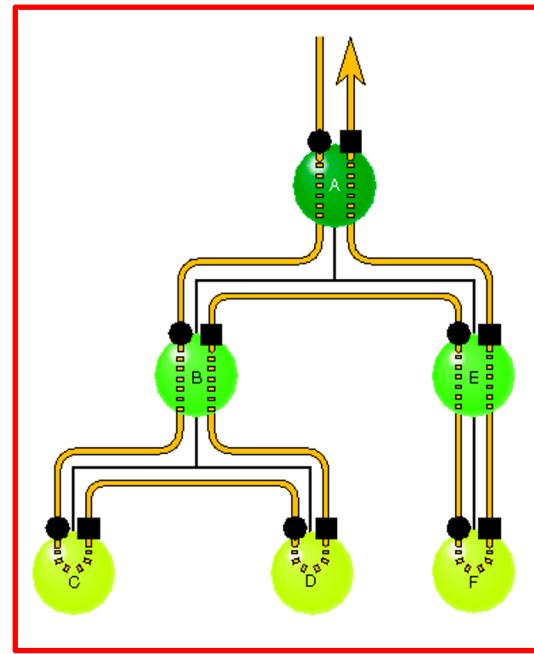
Os nós podem possuir filhos sendo que as transformações nos pais são repassadas para os filhos.



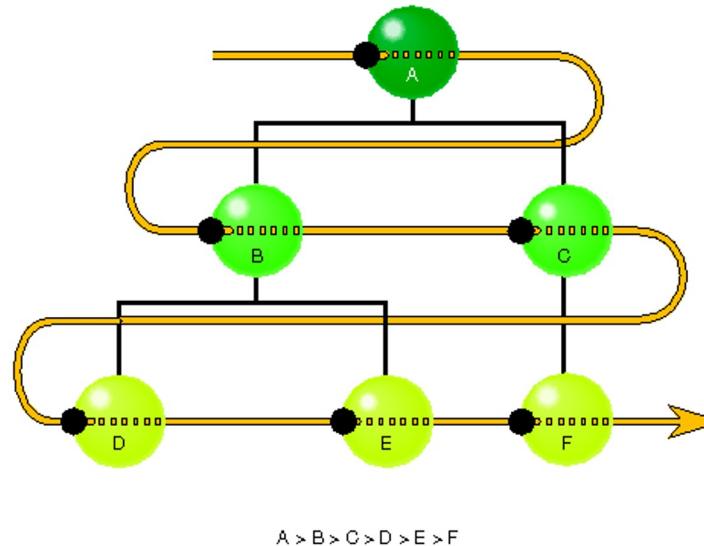
Traversal

O *traversal* vai aplicando operações nos nós do grafo conforme visita recursivamente cada um deles.

Dados podem ser empilhados para algum uso no retorno a um nó.

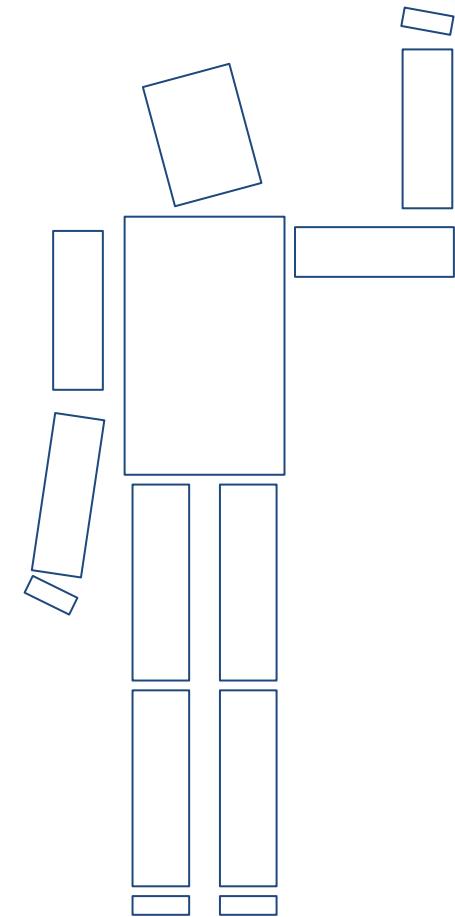
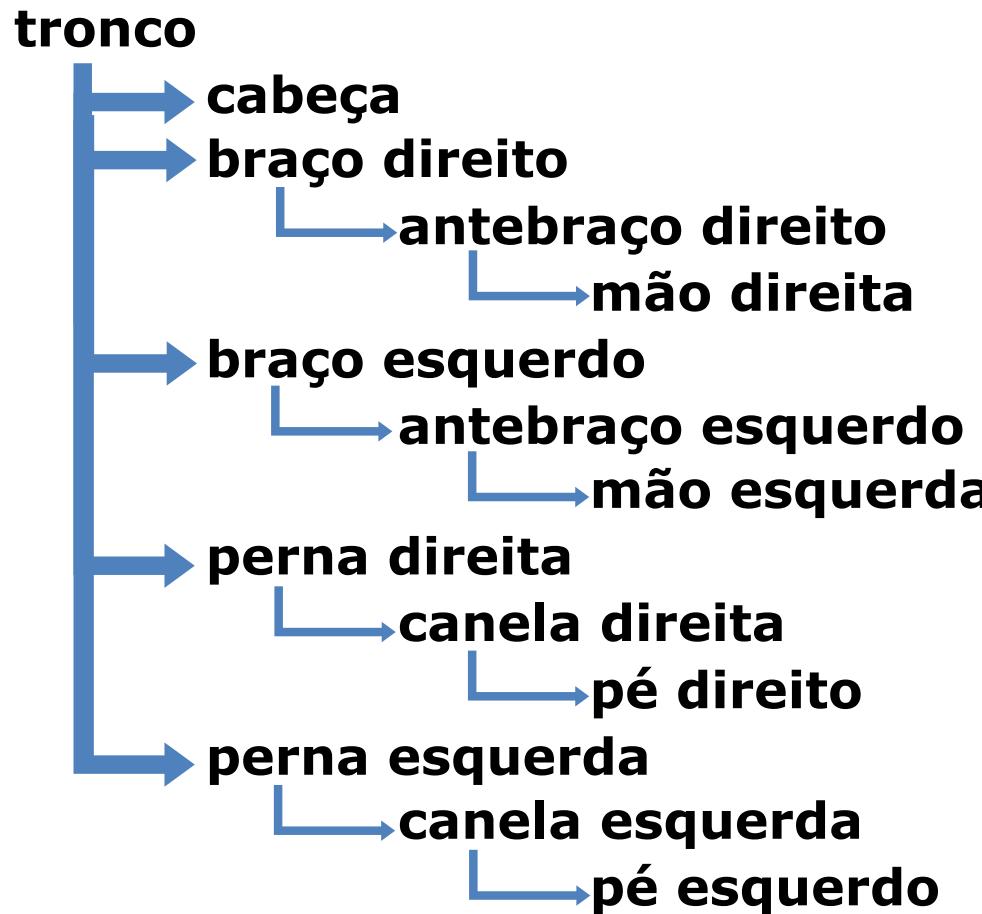


$A > B > C > D > B > E > F > E > A$



$A > B > C > D > E > F$

Esqueleto – Representação Hierárquica

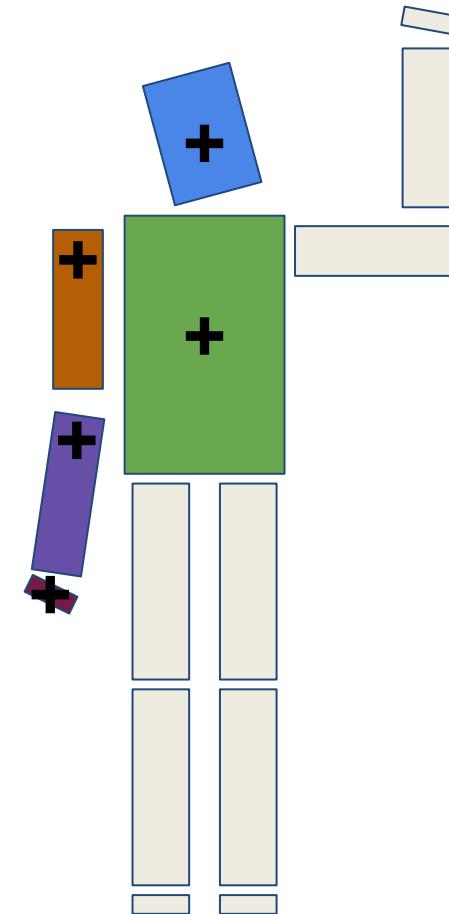
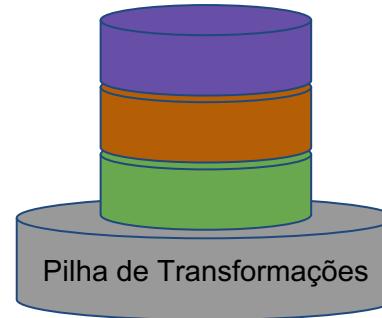


Representação Hierárquica

- Cada grupo contém subgrupos e/ou objetos geométricos
- Cada grupo possui uma transformação associada ao grupo pai
- A transformação no nó folha é a concatenação de todas as transformações no caminho do nó raiz até a folha
- Alterar a transformação de um grupo afeta todas as partes subsequentes
- Permite edição de alto nível simples, alterando apenas um nó, alteramos todos os relacionados

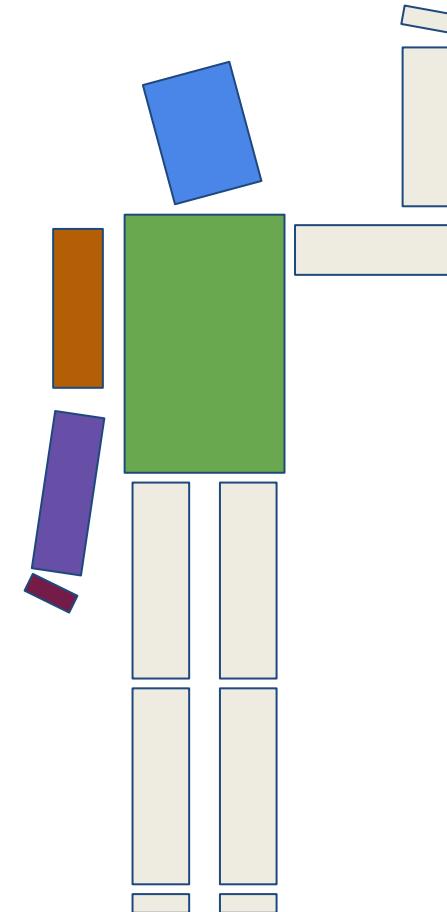
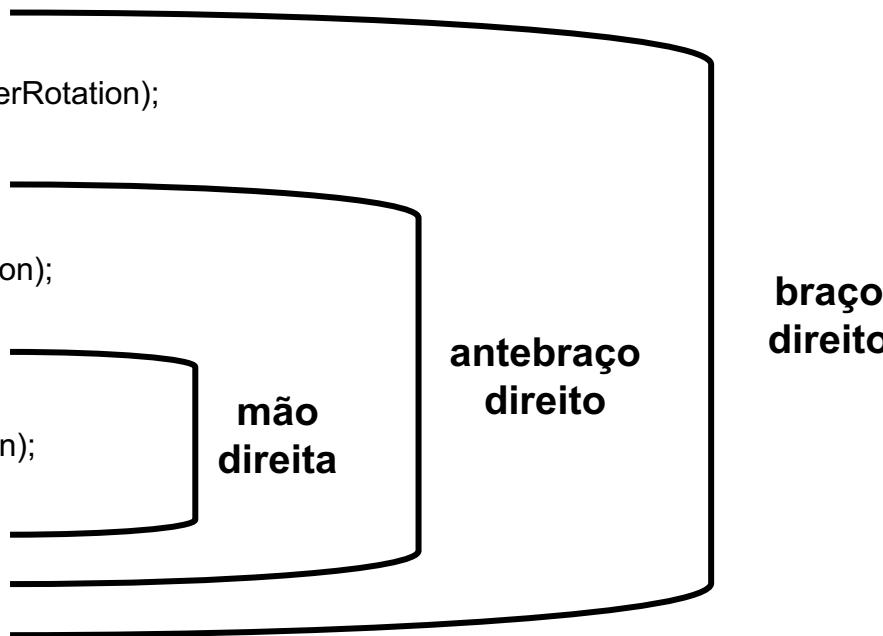
Implementando Representação Hierárquica

```
identity();
drawTorso();
pushmatrix(); // armazena a matriz de transformação atual na pilha
translate(0, 3); // matriz de transformação é atualizada pela translação
rotate(headRotation); // matriz de transformação é atualizada pela rotação
drawHead();
popmatrix(); // recupera matriz de transformação armazenada na pilha
pushmatrix();
translate(-2, 3);
rotate(rightShoulderRotation);
drawUpperArm();
pushmatrix();
translate(0, -3);
rotate(elbowRotation);
drawLowerArm();
pushmatrix();
translate(0, -3);
rotate(wristRotation);
drawHand();
popmatrix();
popmatrix();
popmatrix();
```



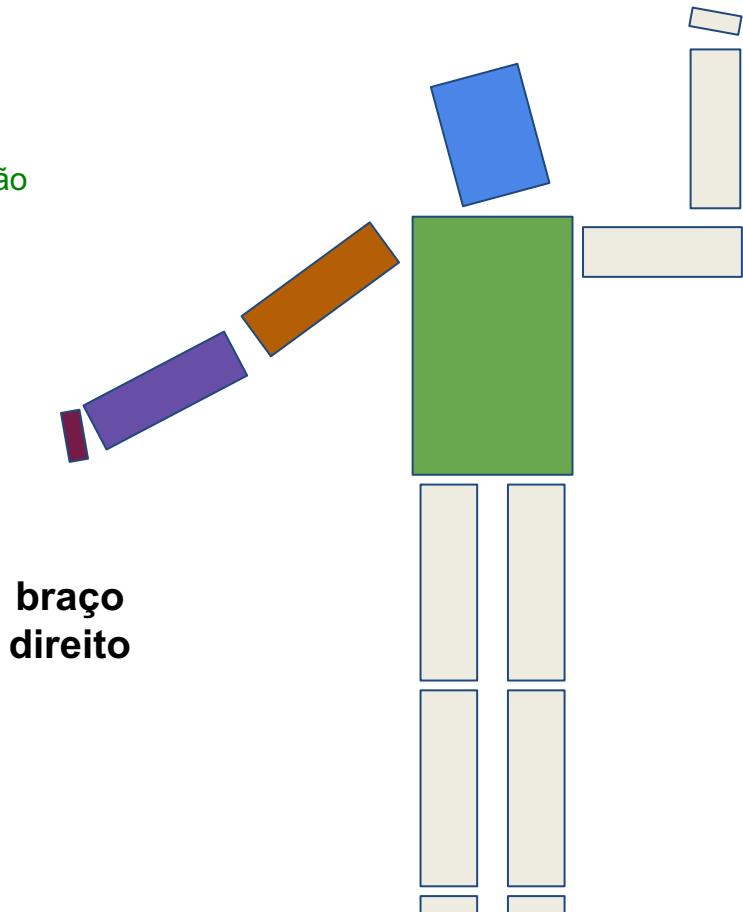
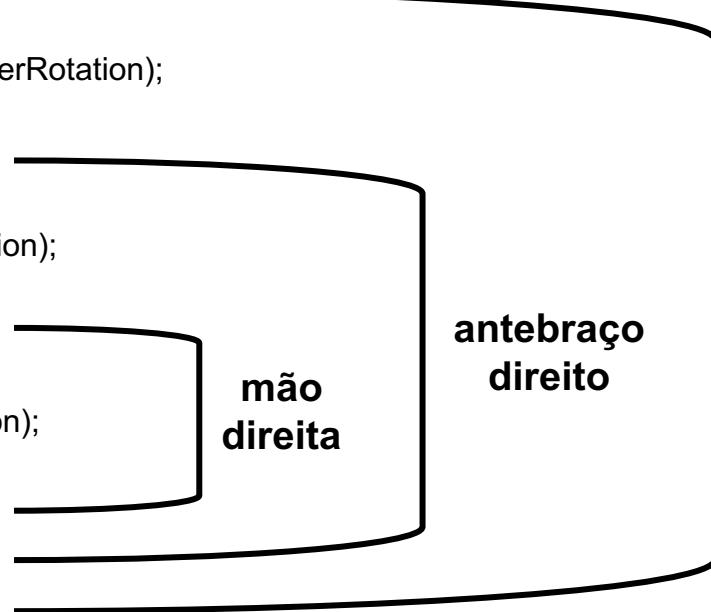
Implementando Representação Hierárquica

```
translate(0, 5);
drawTorso();
pushmatrix(); // armazena a matriz de transformação atual na pilha
translate(0, 5); // matriz de transformação é atualizada pela translação
rotate(headRotation); // matriz de transformação é atualizada pela rotação
drawHead();
popmatrix(); // recupera matriz de transformação armazenada na pilha
pushmatrix();
translate(-2, 3);
rotate(rightShoulderRotation);
drawUpperArm();
pushmatrix();
translate(0, -3);
rotate(elbowRotation);
drawLowerArm();
pushmatrix();
translate(0, -3);
rotate(wristRotation);
drawHand();
popmatrix();
popmatrix();
popmatrix();
```



Implementando Representação Hierárquica

```
translate(0, 5);
drawTorso();
pushmatrix(); // armazena a matriz de transformação atual na pilha
translate(0, 5); // matriz de transformação é atualizada pela translação
rotate(headRotation); // matriz de transformação é atualizada pela rotação
drawHead();
popmatrix(); // recupera matriz de transformação armazenada na pilha
pushmatrix();
translate(-2, 3);
→ rotate(rightShoulderRotation);
drawUpperArm();
pushmatrix();
translate(0, -3);
rotate(elbowRotation);
drawLowerArm();
pushmatrix();
translate(0, -3);
rotate(wristRotation);
drawHand();
popmatrix();
popmatrix();
popmatrix();
```



Perguntas

Referência:

O cubo vermelho foi criado na origem

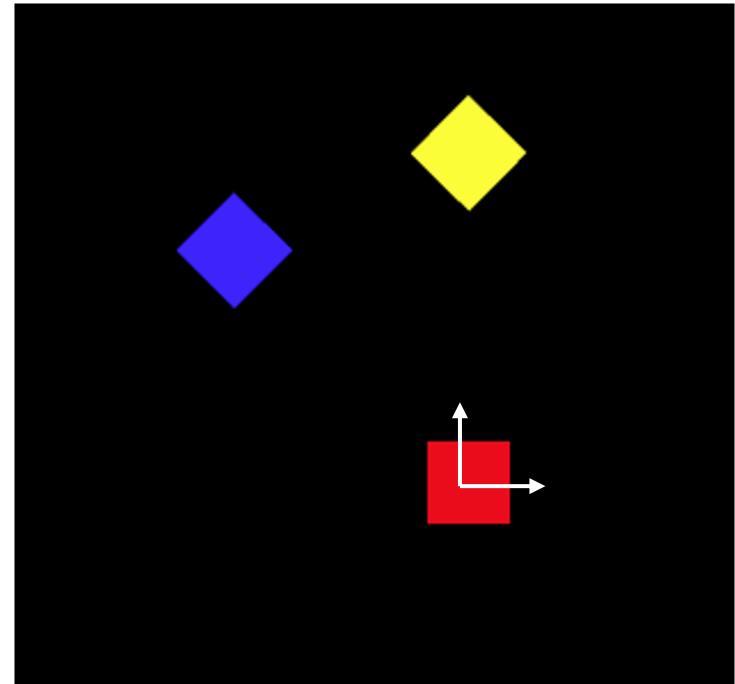
Pergunta:

Qual cubo foi:

1º rotacionado 45° e depois transladado na vertical?

1º transladado na vertical e depois rotacionado de 45°?

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.71 & -0.71 & 0 & 0 \\ 0.71 & 0.71 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Perguntas

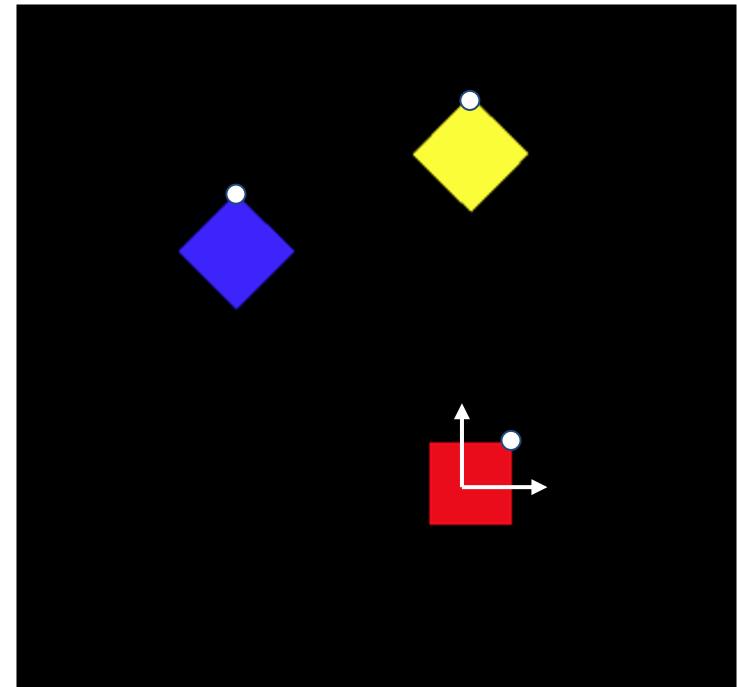
$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.71 & -0.71 & 0 & 0 \\ 0.71 & 0.71 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Primeiro Rotaciona depois Translada

$$TR = \begin{bmatrix} 0.71 & -0.71 & 0 & 0 \\ 0.71 & 0.71 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \implies \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 5.42 \\ 1 \\ 1 \end{bmatrix}$$

Primeiro Translada depois Rotaciona

$$RT = \begin{bmatrix} 0.71 & -0.71 & 0 & -2.84 \\ 0.71 & 0.71 & 0 & 2.84 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \implies \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2.84 \\ 4.26 \\ 1 \\ 1 \end{bmatrix}$$



Perguntas

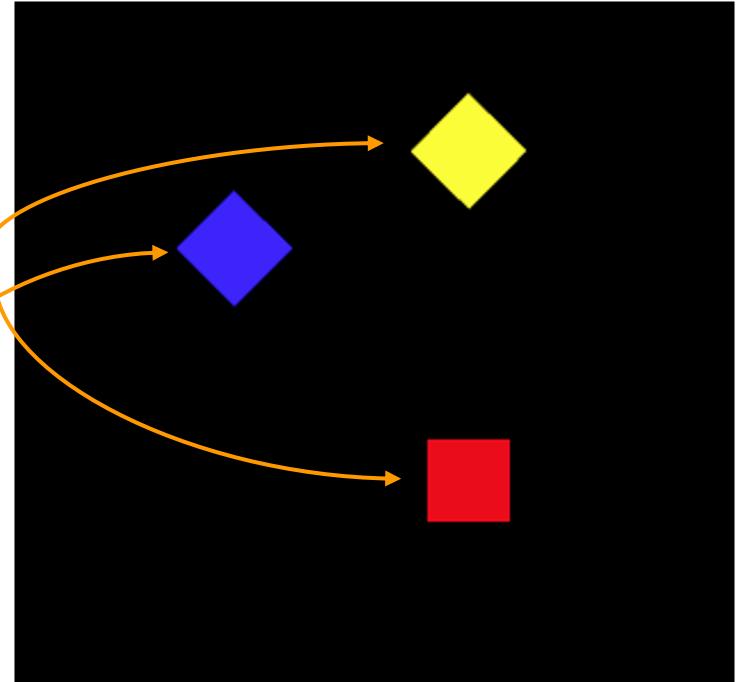
```
<Scene>
<Viewpoint position="0 0 15" fieldOfView="0.7854"/>

<Transform translation="0 0 0">
<Transform rotation="0 0 1 0">
<Shape>
<Box size="1 1 1"/>
<Appearance>
<Material diffuseColor="1 0 0"/>
</Appearance>
</Shape>
</Transform>
</Transform>

<Transform translation="0 4 0">
<Transform rotation="0 0 1 0.79">
<Shape>
<Box size="1 1 1"/>
<Appearance>
<Material diffuseColor=" " />
</Appearance>
</Shape>
</Transform>
</Transform>

<Transform rotation="0 0 1 0.79">
<Transform translation="0 4 0">
<Shape>
<Box size="1 1 1"/>
<Appearance>
<Material diffuseColor=" " />
</Appearance>
</Shape>
</Transform>
</Transform>

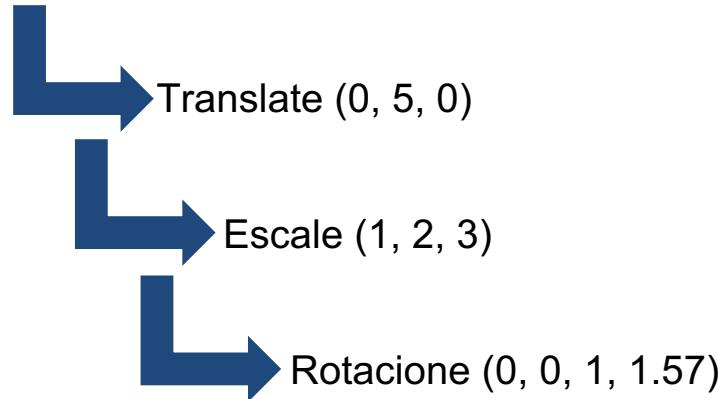
</Scene>
```



Usando Matrizes

Na prática multiplicamos as matrizes e empilhamos.

Por exemplo:



```
<Transform translation="0 5 0">  
<Transform scale="1 2 3">  
<Transform rotation="0 0 1 1.57">
```

Usando Matrizes

Na prática multiplicamos as matrizes e empilhamos.

Por exemplo:

Translação (0,5,0)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

PushMatrix()

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 5 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 2 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Escala (1,2,3)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 5 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

PushMatrix()

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 5 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 2 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

PILHA

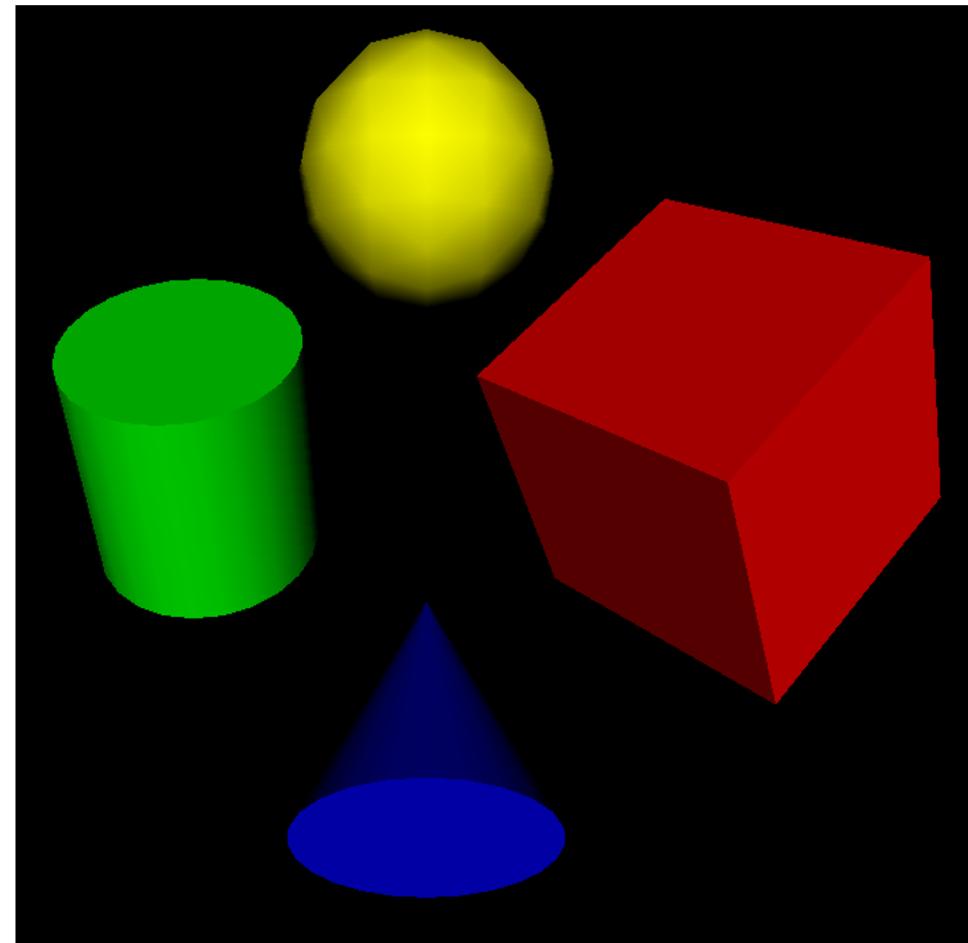
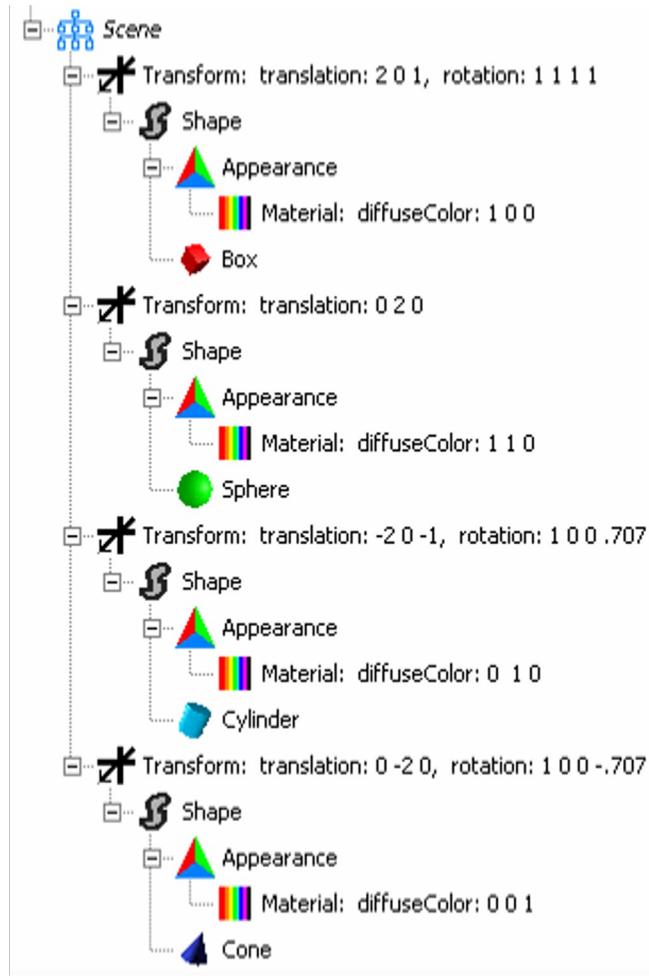
Rotação (0, 0, 1, 1.57)

PushMatrix()

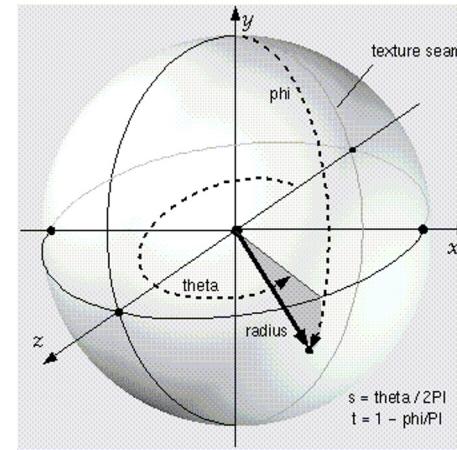
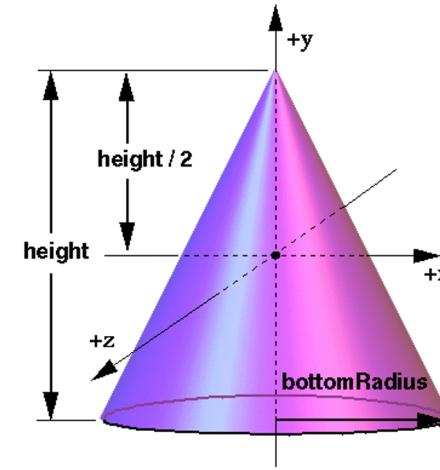
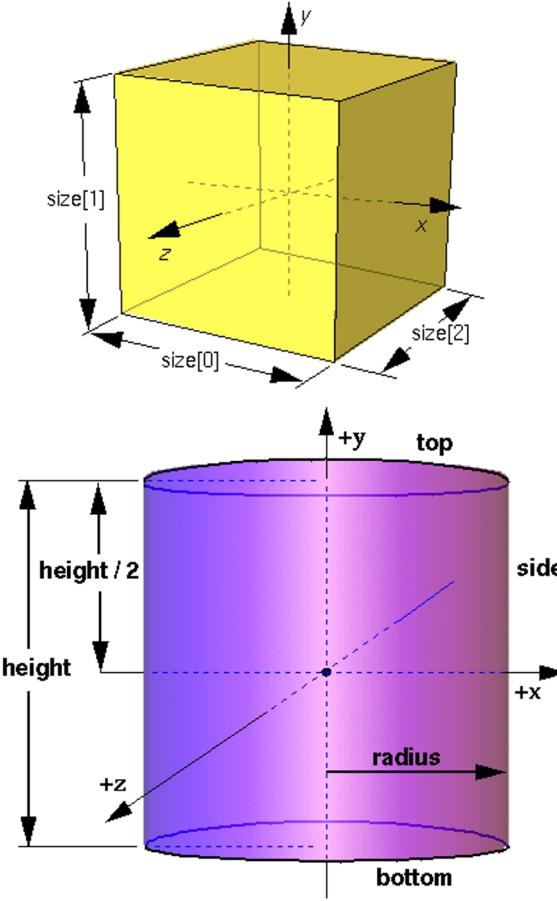
Outra Forma de Pensar

	Identidade * pontos
 Translate (0, 5, 0)	Identidade * Translação * pontos
 Escale (1, 2, 3)	Identidade * Translação * Escala * pontos
 Rotacione (0, 0, 1, 1.57)	Identidade * Translação * Escala * Rotação * pontos

Formas e Transformações (X3D-Edit)



Especificação de algumas primitivas



Especificação das primitivas

```
Box : X3DGeometryNode {  
    SFNode [in,out] metadata NULL [X3DMetadataObject]  
    SFVec3f []     size    2 2 2 (0,∞)  
    SFBool []      solid   TRUE  
}
```

```
Cylinder : X3DGeometryNode {  
    SFNode [in,out] metadata NULL [X3DMetadataObject]  
    SFBool []      bottom  TRUE  
    SFFloat []     height  2  (0,∞)  
    SFFloat []     radius  1  (0,∞)  
    SFBool []      side    TRUE  
    SFBool []      solid   TRUE  
    SFBool []      top    TRUE  
}
```

```
Cone : X3DGeometryNode {  
    SFNode [in,out] metadata NULL [X3DMetadataObject]  
    SFBool []      bottom  TRUE  
    SFFloat []     bottomRadius 1  (0,∞)  
    SFFloat []     height   2  (0,∞)  
    SFBool []      side    TRUE  
    SFBool []      solid   TRUE  
}
```

```
Sphere : X3DGeometryNode {  
    SFNode [in,out] metadata NULL [X3DMetadataObject]  
    SFFloat []     radius  1  (0,∞)  
    SFBool []      solid   TRUE  
}
```

Computação Gráfica

Luciano Soares
[<lpsoares@insper.edu.br>](mailto:lpsoares@insper.edu.br)