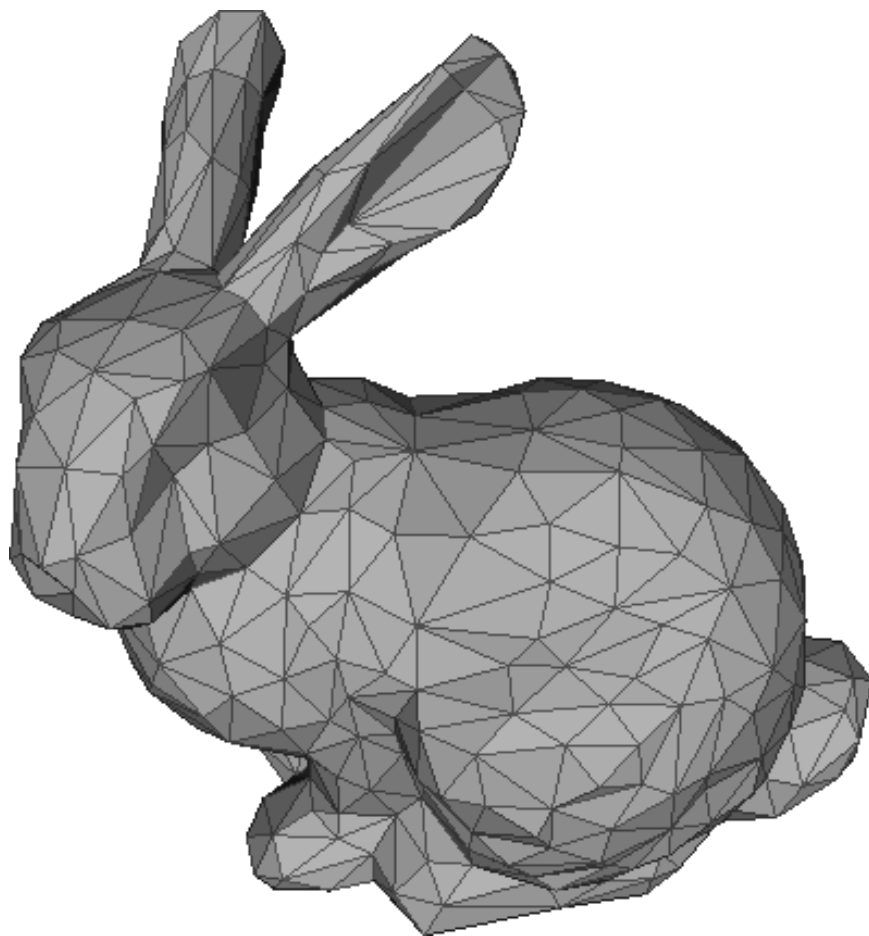


# Computação Gráfica

Aula 7: Malhas Básicas e Grafo de Cena

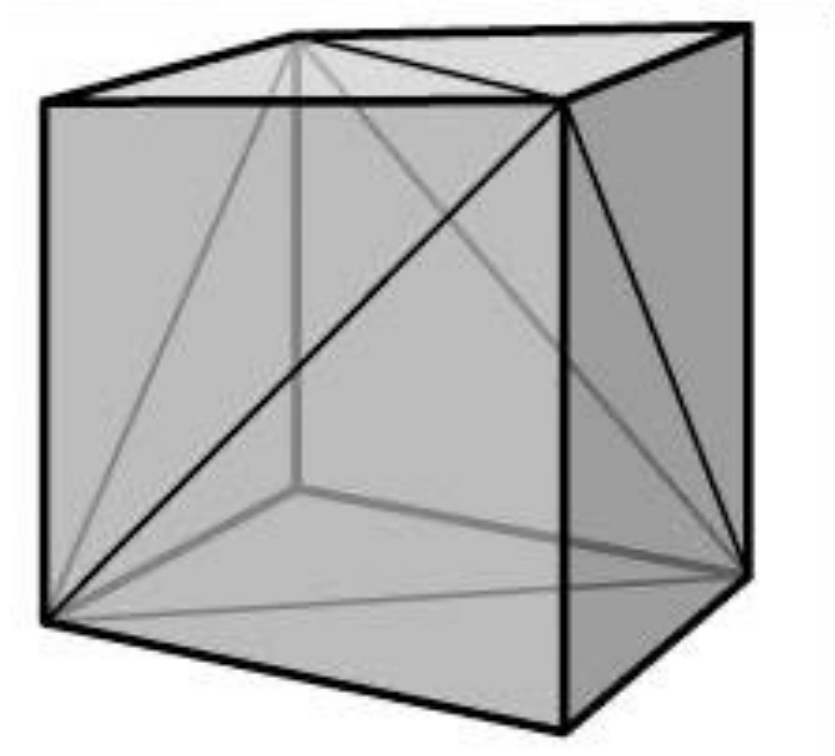
# Malha de Triângulos



**Stanford Bunny (low poly)**

original: <https://graphics.stanford.edu/data/3Dscanrep/>

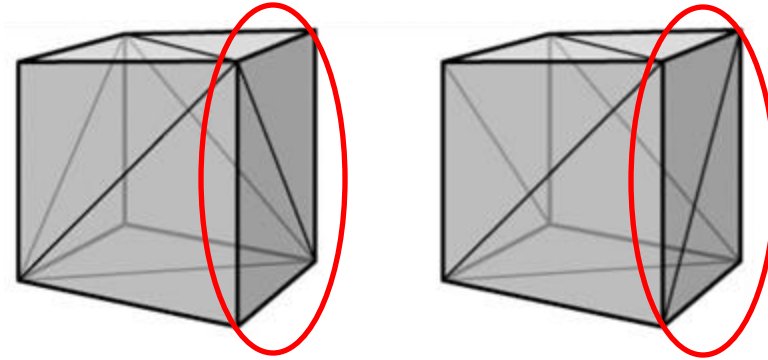
# Malha de Triângulos Pequena



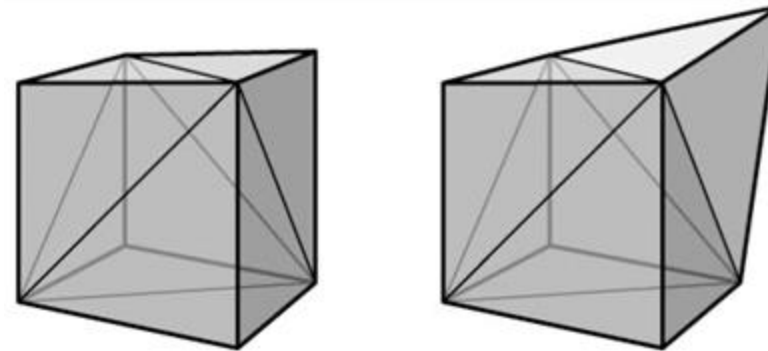
**8 vértices, 12 triângulos**

# Topologia versus Geometria

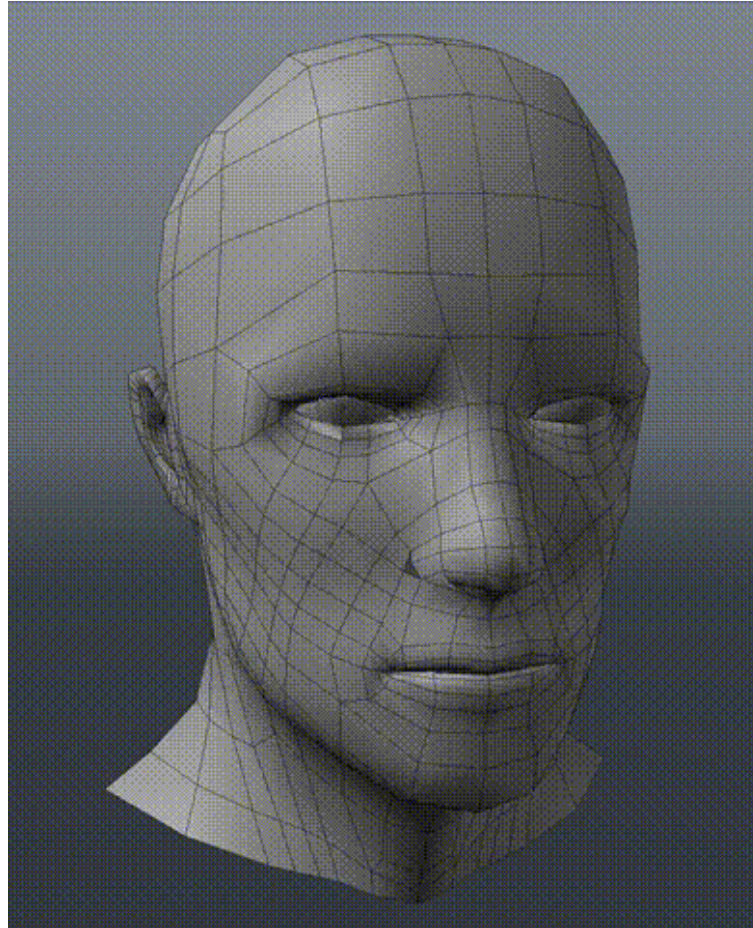
Mesma geometria, diferente topologia da malha



Mesma topologia, diferente geometria



# Blend Shapes



# Malha de Triângulos Grande

## Digital Michelangelo Project

### The statue

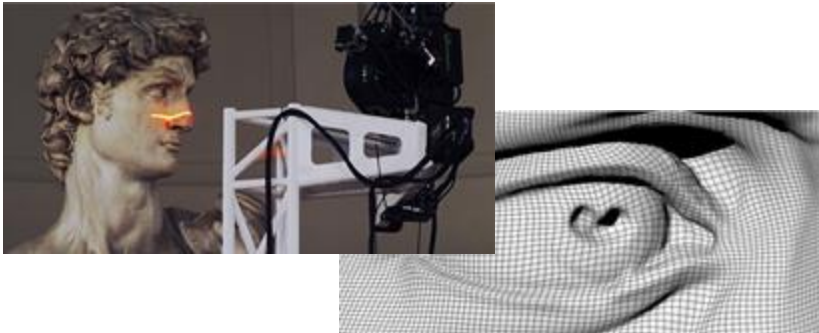
height without pedestal	517 cm
surface area	19 m <sup>2</sup>
volume	2.2 m <sup>3</sup>
weight	5,800 kg

### Our raw dataset

number of polygons	2 billion
number of color images	7,000
losslessly compressed size	32 GB

### Other statistics

total size of scanning team	22 people
staffing in the museum	3 people (on average)
time spent scanning	360 hours over 30 days
man-hours scanning	1,080
man-hours post-processing	1,500 (so far)





# Malha de Triângulos Gigantesca

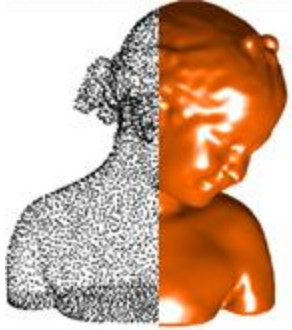
Google Earth

Malha reconstruída de imagens de satélite e aéreas

Trilhões de triângulos



# Mais Tarefas de Processamento Geométrico



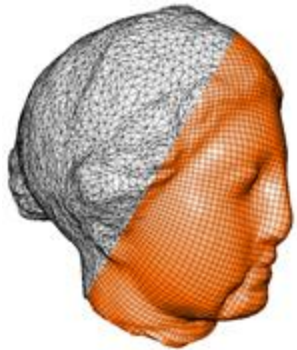
Reconstrução 3D



Análise de Formas



Filtragem



Remeshing



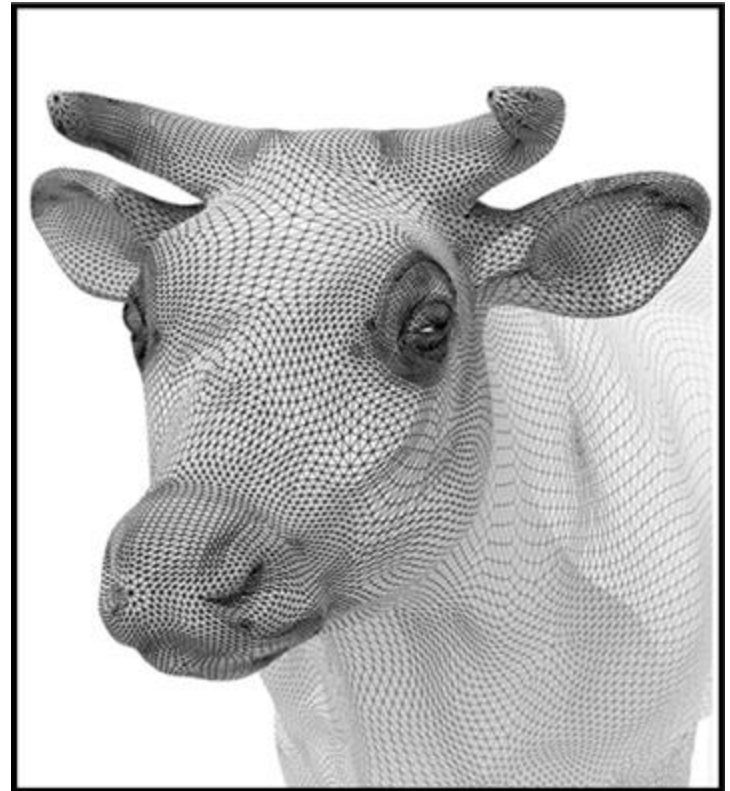
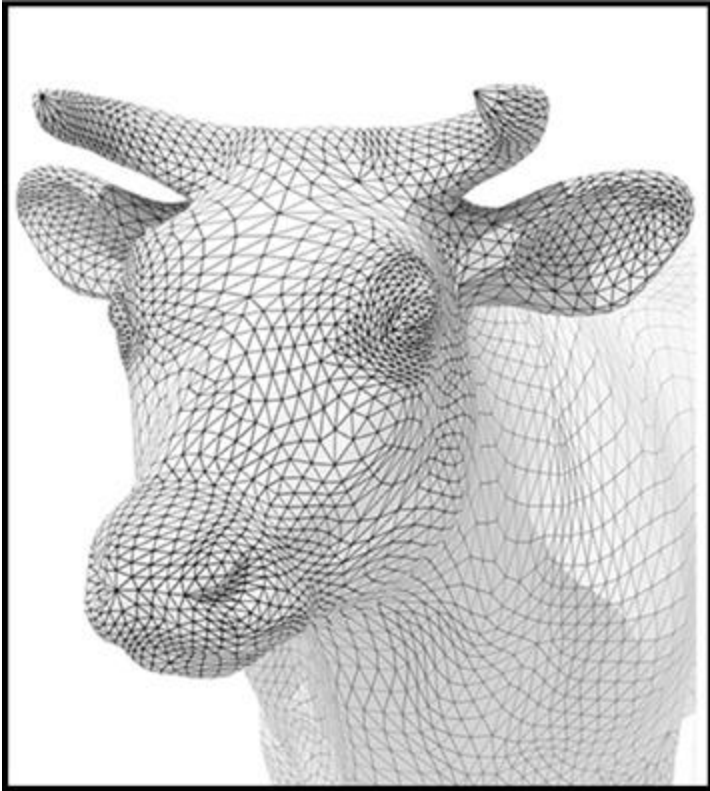
Parametrização



Compressão

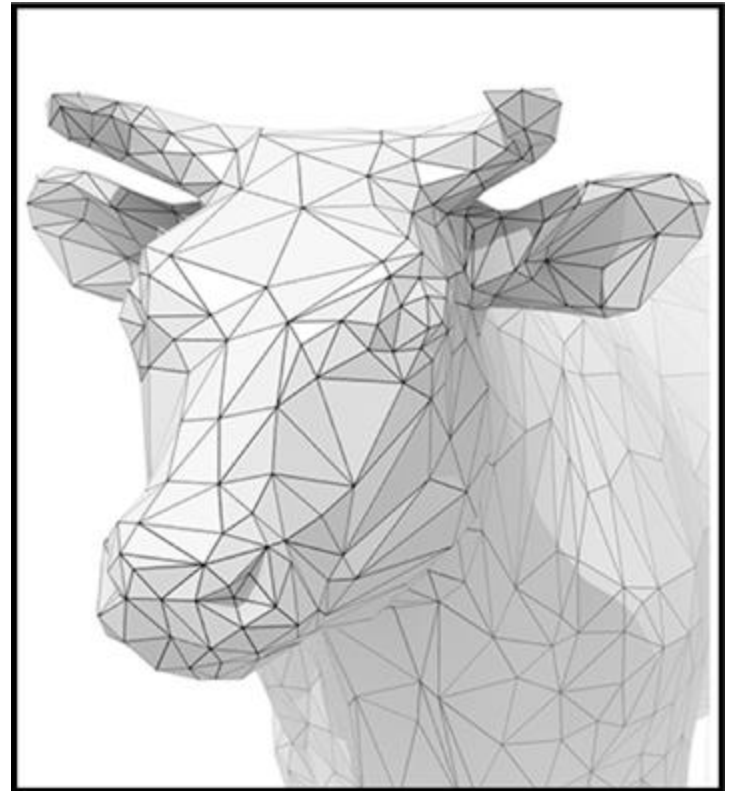
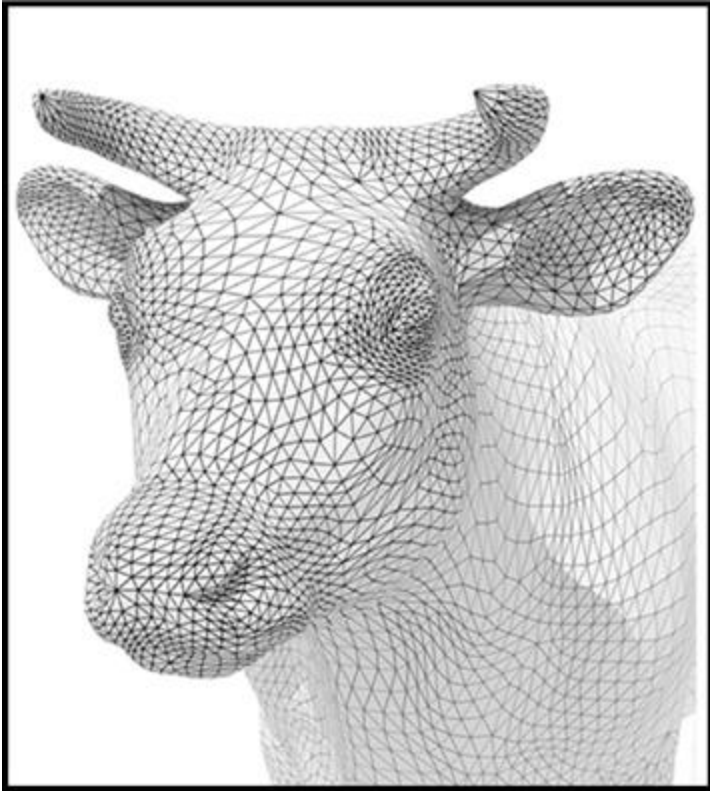


# Upsampling (Refinando Malha)



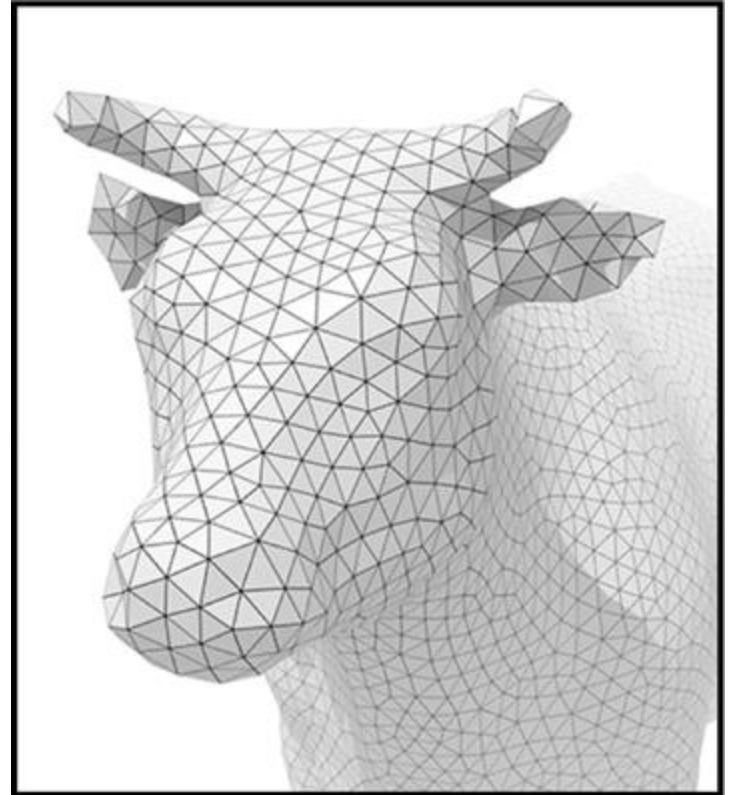
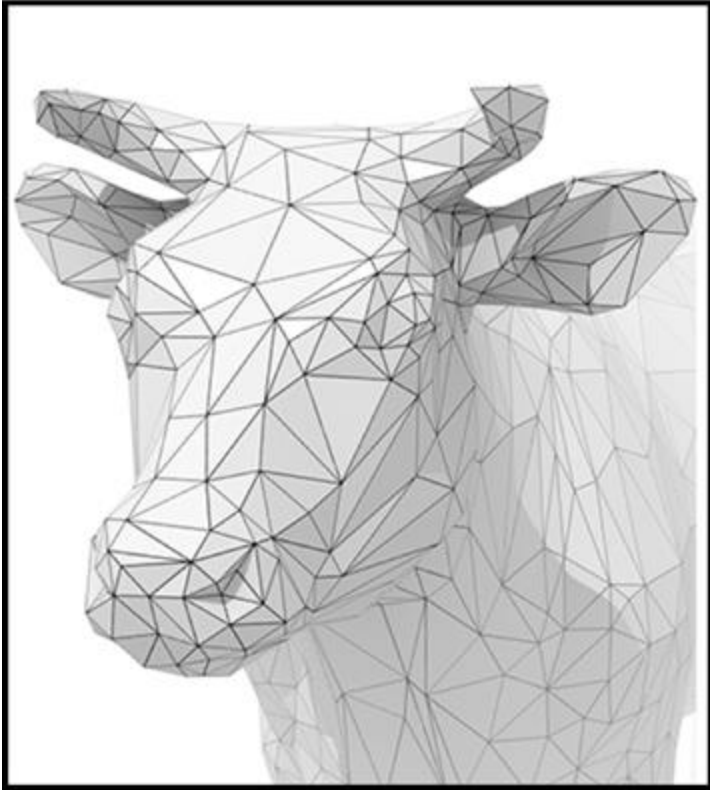
Subdivisão : melhorando resolução interpolando os pontos

# Downsampling (Simplificando a Malha)



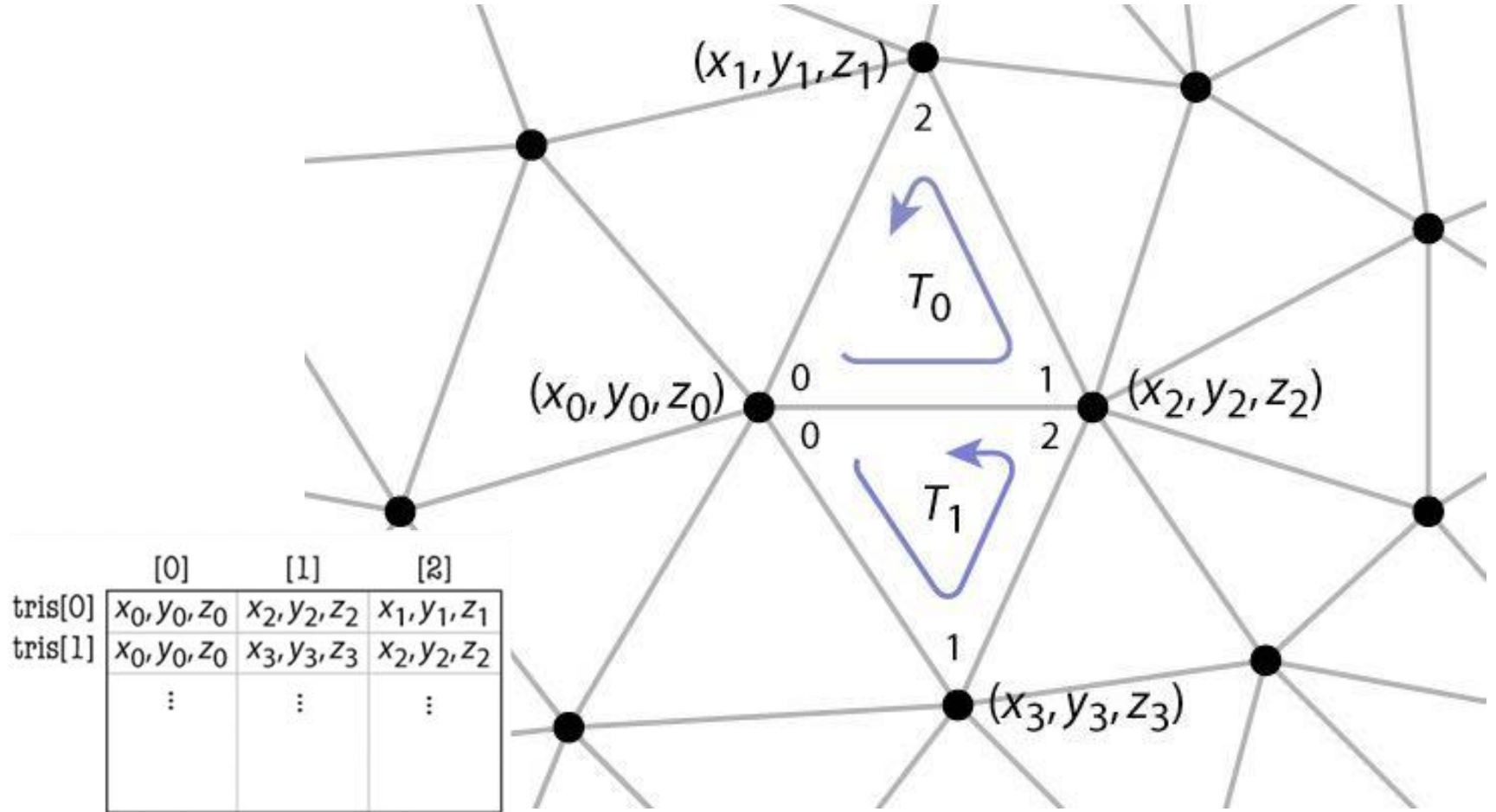
Diminuindo Pontos : tentando manter a forma original

# Homogeneizando Malha



Ajustando Pontos para melhorar qualidade (possivelmente)

# Lista de Triângulos

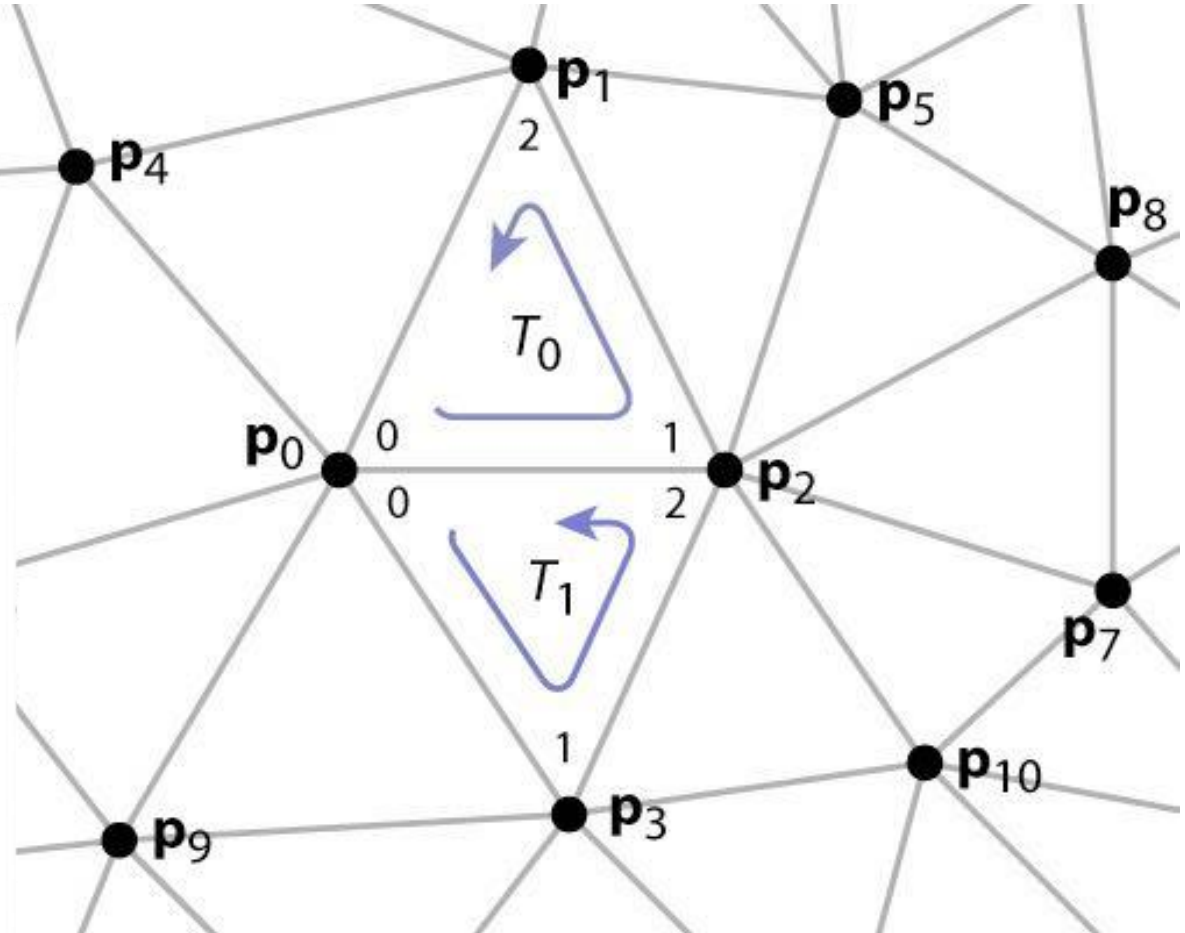




# Lista de Pontos e sua conexão por índices

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

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
	$\vdots$





# Comparação

## Triângulos

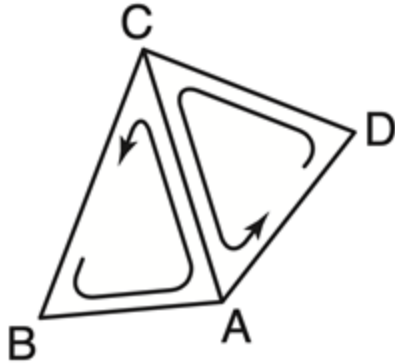
- + Simples
- Muita Informação Redundante

## Pontos e Triângulos

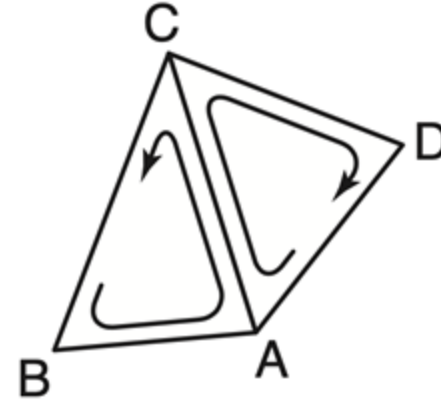
- + Compartilhamento de vértices reduz consumo de memória
- + Garante integridade da malha  
(alterar um vértice, altera para todos os polígonos)

# Validade Topológica: Consistência da Orientação

Orientação consistente

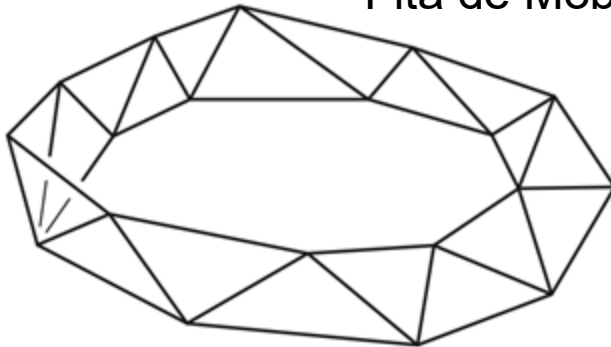


Orientação inconsistente



Não orientável

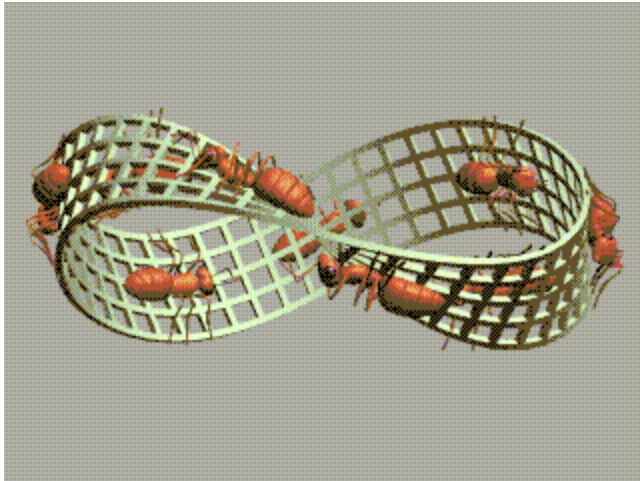
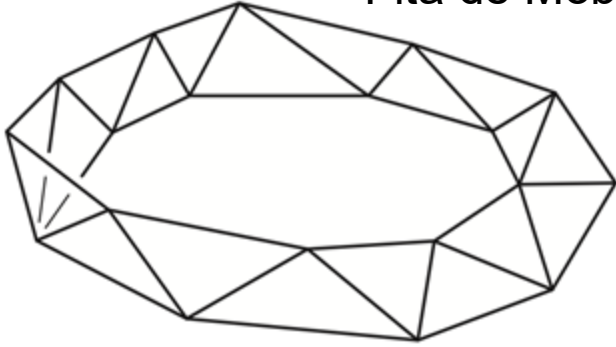
Fita de Möbius



Vingadores: Ultimato

# Validade Topológica: Consistência da Orientação

Fita de Möbius



M. C. Escher - Möbius Strip II (1963) esper

# X3D

- triangleSet (já vimos)
- triangleStripSet
- indexedTriangleStripSet
- indexedFaceSet

# TriangleStripSet

Um **TriangleStripSet** representa uma forma geométrica 3D composta por faixas de triângulos. O campo **stripCount** descreve quantos vértices devem ser usados em cada faixa do **Coordinate**. As coordenadas são atribuídas a cada faixa pegando os vértices **stripCount[i]** do campo de coordenadas, onde i é um índice sequencial de stripCount.

```
TriangleStripSet : X3DComposedGeometryNode {  
  MFNode [in,out] attrib [] [X3DVertexAttributeNode]  
  SFNode [in,out] color NULL [X3DColorNode]  
  SFNode [in,out] coord NULL [X3DCoordinateNode]  
  SFNode [in,out] fogCoord NULL [FogCoordinate]  
  SFNode [in,out] metadata NULL [X3DMetadataObject]  
  SFNode [in,out] normal NULL [X3DNormalNode]  
  MFInt32 [in,out] stripCount [] [3,∞)  
  SFNode [in,out] texCoord NULL [X3DTextureCoordinateNode]  
  SFBool [] ccw TRUE  
  SFBool [] colorPerVertex TRUE  
  SFBool [] normalPerVertex TRUE  
  SFBool [] solid TRUE  
}
```



# TriangleStripSet (exemplo)

<Shape>

<TriangleStripSet stripCount='13'>

<Coordinate point='

```
-4.0 -1.0 -0.5 -> P0  
-4.5 -2.0 -0.5 -> P1  
-3.0 -0.5 0.0 -> P2  
-2.5 -1.5 -0.5 -> P3  
-2.0 -0.5 -1.0 -> P4  
-1.5 -1.5 -0.5 -> P5  
-0.5 0.5 -0.5 -> P6  
0.0 0.0 0.0 -> P7  
1.0 0.5 -0.5 -> P8  
1.5 -2.0 -1.0 -> P9  
2.5 -2.0 -0.5 -> P10  
2.5 -2.5 -0.5 -> P11  
3.5 -2.0 -1.0 -> P12
```

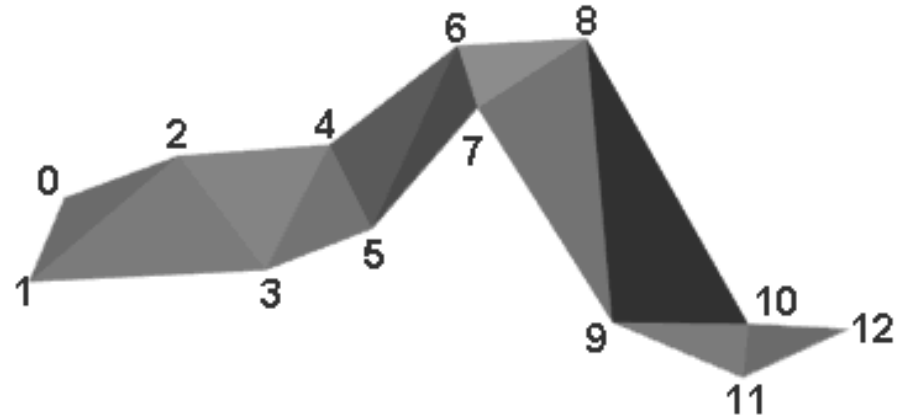
'/>

</TriangleStripSet>

<Appearance> <Material emissiveColor='0.5 0.5 0.5' /> </Appearance>

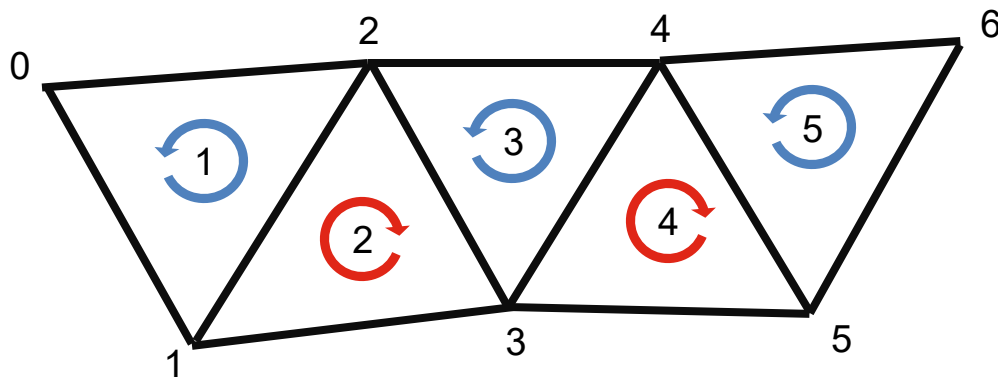
</Shape>

Os vértices são conectados de 3 em 3. Assim o primeiro triângulo será ligando os vértices (0, 1, 2), o segundo (1, 2, 3) e assim por diante, até chegar a contagem definida em stripCount. Perceba que stripCount é uma lista.



# Cuidado

A direção de montagem dos triângulos precisa ser constantemente alternada, senão a detecção de triângulos irá falhar



Um dos truques é inverter a ordem de conexão, de modo que, nos triângulos pares, a conexão de dois vértices seja invertida.

Por exemplo: conecte (0,1,2), depois (1,3,2), depois (2,3,4) e assim por diante.

# IndexedTriangleStripSet

Um **IndexedTriangleStripSet** representa uma forma 3D composta de um conjunto de triângulos em forma de uma tira, são usados nos índices do campo **index** para especificar como montar faixa de triângulos. Um índice de "-1" indica que a faixa atual terminou e a próxima começa.

```
IndexedTriangleStripSet : X3DComposedGeometryNode {  
  MFInt32    [in]      set_index      []      [0,∞) or -1  
  MFNode     [in,out]  attrib          []      [X3DVertexAttributeNode]  
  SFNode     [in,out]  color           NULL    [X3DColorNode]  
  SFNode    [in,out]  coord          NULL    [X3DCoordinateNode]  
  SFNode     [in,out]  fogCoord        NULL    [FogCoordinate]  
  SFNode     [in,out]  metadata        NULL    [X3DMetadataObject]  
  SFNode     [in,out]  normal          NULL    [X3DNormalNode]  
  SFNode     [in,out]  texCoord        NULL    [X3DTextureCoordinateNode]  
  SFBool     []        ccw             TRUE  
  SFBool     []        colorPerVertex TRUE  
  SFBool     []        normalPerVertex TRUE  
  SFBool     []        solid           TRUE  
  MFInt32   []        index         []      [0,∞) or -1  
}
```

# IndexedTriangleStripSet (exemplo)

```
<Shape>
```

```
<IndexedTriangleStripSet
```

```
  index=' 1 3 9 5 7 -1 -> F0  
          0 1 9 -1 -> F1  
          1 2 3 -1 -> F2  
          3 4 5 -1 -> F3  
          5 6 7 -1 -> F4  
          7 8 9 -1 -> F5
```

```
'>
```

```
<Coordinate point='
```

```
  0.0   1.0   0.0 -> P0  
 -0.23  0.32  0.0 -> P1  
 -0.95  0.30  0.0 -> P2  
 -0.38 -0.12  0.0 -> P3  
 -0.58 -0.80  0.0 -> P4  
 -0.0   -0.4   0.0 -> P5  
  0.58 -0.80  0.0 -> P6  
  0.38 -0.12  0.0 -> P7  
  0.95  0.30  0.0 -> P8  
  0.23  0.32  0.0 -> P9
```

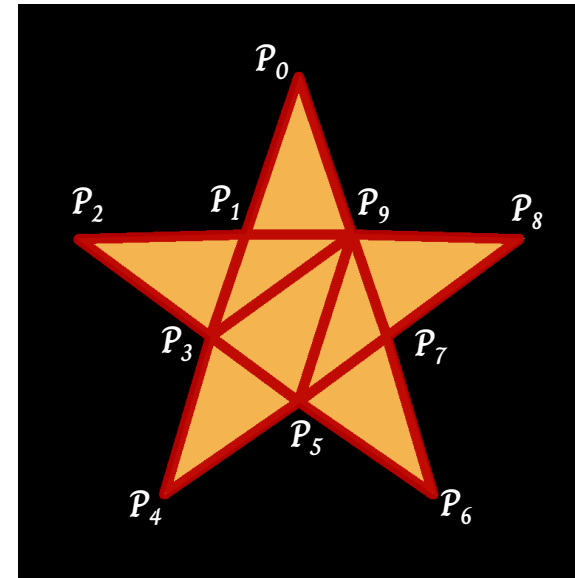
```
'/>
```

```
</IndexedTriangleStripSet>
```

```
<Appearance> <Material emissiveColor='0.5 0.5 0.5' /> </Appearance>
```

```
</Shape>
```

Os vértices são conectados seguindo a ordem definida no campo index. Ligando de 3 em 3 vértices até encontrar um valor -1. Outras listas de índices podem aparecer na sequência.



# IndexedFaceSet

Um **IndexedFaceSet** representa uma forma 3D composta por um conjunto de polígonos. Os índices do campo **coordIndex** especificam como os vértices devem ser conectados para formar as faces.

```
IndexedFaceSet : X3DComposedGeometryNode {
  MFInt32      [in]      set_colorIndex
  MFInt32      [in]      set_coordIndex
  MFInt32      [in]      set_normalIndex
  MFInt32      [in]      set_texCoordIndex
  MFNode[in,out] attrib   []      [X3DVertexAttributeNode]
  SFNode[in,out] color    NULL    [X3DColorNode]
  SFNode[in,out] coord    NULL    [X3DCoordinateNode]
  SFNode[in,out] fogCoord NULL    [FogCoordinate]
  SFNode[in,out] metadata NULL    [X3DMetadataObject]
  SFNode[in,out] normal   NULL    [X3DNormalNode]
  SFNode[in,out] texCoord NULL    [X3DTextureCoordinateNode]
  SFBool []      ccw      TRUE
  MFInt32 []      colorIndex []      [0,∞) or -1
  SFBool []      colorPerVertex TRUE
  SFBool []      convex    TRUE
  MFInt32 []      coordIndex []      [0,∞) or -1
  SFFloat []      creaseAngle 0      [0,∞)
  MFInt32 []      normalIndex  []      [0,∞) or -1
  SFBool []      normalPerVertex TRUE
  SFBool []      solid        TRUE
  MFInt32 []      texCoordIndex []      [-1,∞)
}
```



# IndexedFaceSet

O nó IndexedFaceSet representa uma forma 3D composta por faces (polígonos) construídas a partir de vértices listados no campo coord.

O IndexedFaceSet utiliza os índices em seu campo coordIndex para especificar as faces poligonais, indexando nas coordenadas do nó Coordinate. Um índice de -1 indica que a face atual foi finalizada e que a próxima começa.

Cada face do IndexedFaceSet deve ter:

- **pelo menos três vértices não coincidentes;**
- **vértices que definem um polígono coplanar;**
- **vértices não possuem auto-intersecção.**

# Exemplo IndexedFaceSet

<Shape>

<IndexedFaceSet coordIndex='0 1 2 3 4 5 6 7 -1'>

<Coordinate point='

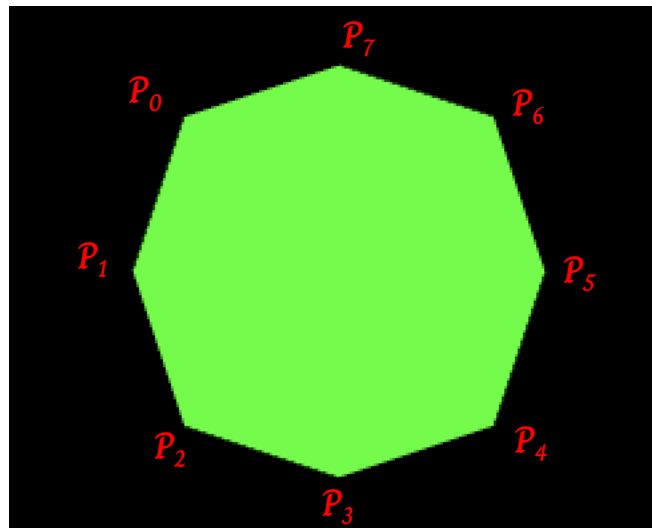
-3	3	0	-> $P_0$
-4	0	0	-> $P_1$
-3	-3	0	-> $P_2$
0	-4	0	-> $P_3$
3	-3	0	-> $P_4$
4	0	0	-> $P_5$
3	3	0	-> $P_6$
0	4	0	-> $P_7$

'/>

</IndexedFaceSet>

<Appearance> <Material emissiveColor='0 1 0' /> </Appearance>

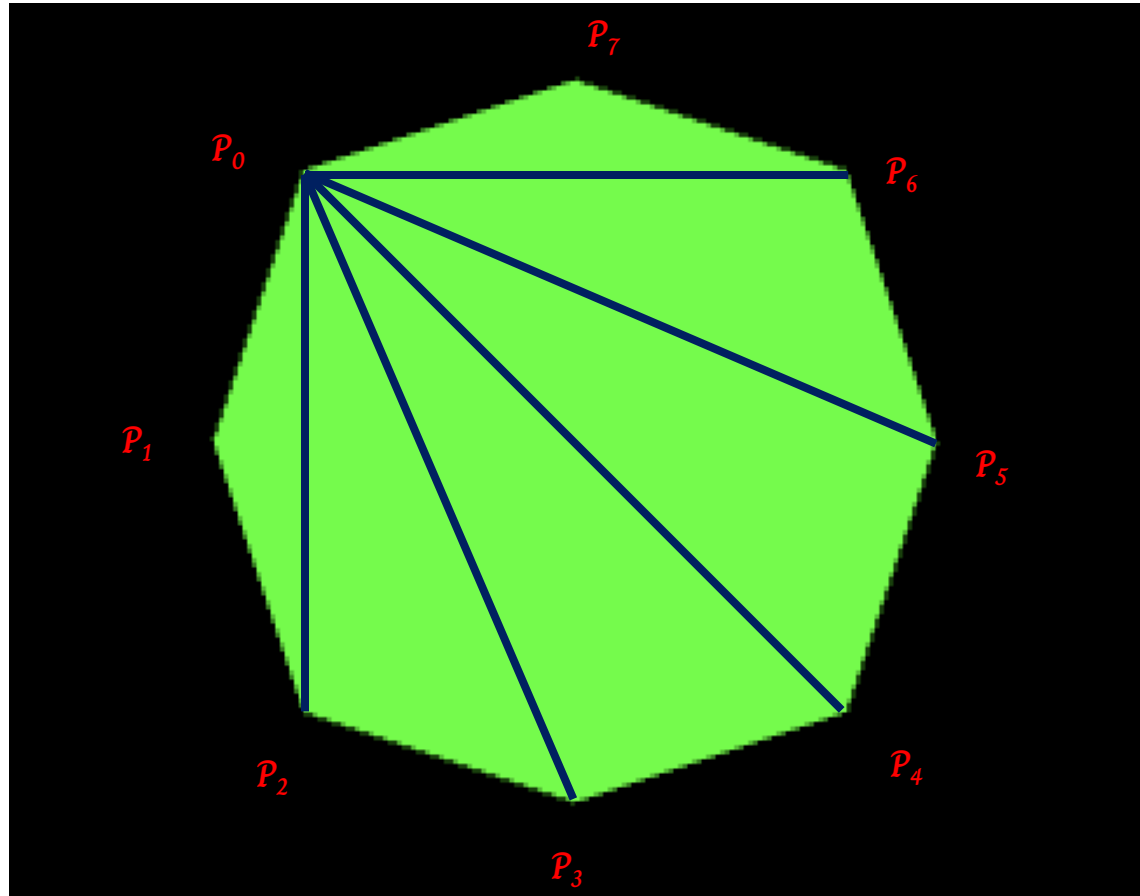
</Shape>



Os pontos são o contorno da superfície.

# Exemplo IndexedFaceSet

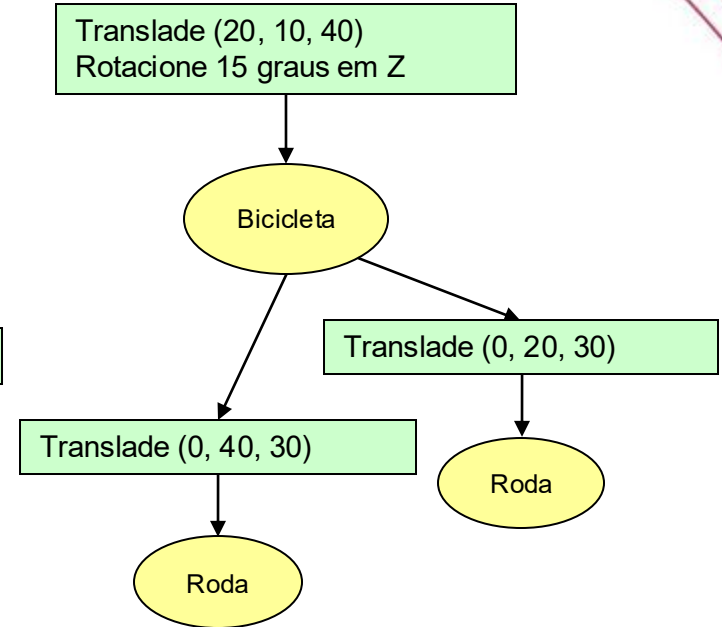
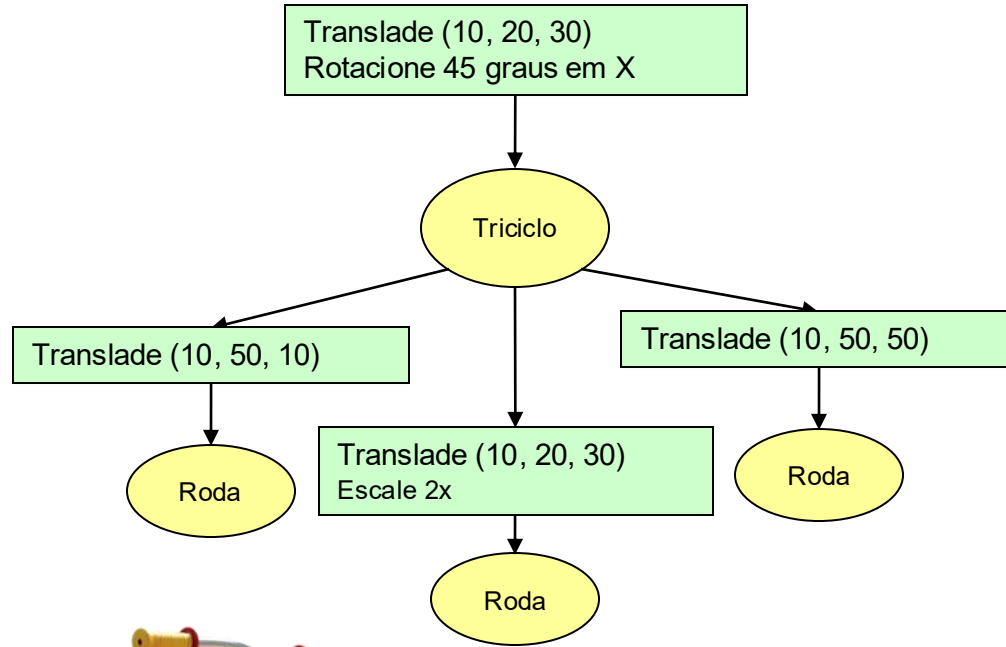
Possível solução para tecer geometria



# Computação Gráfica

## Grafo de Cena

# Grafo de Cena

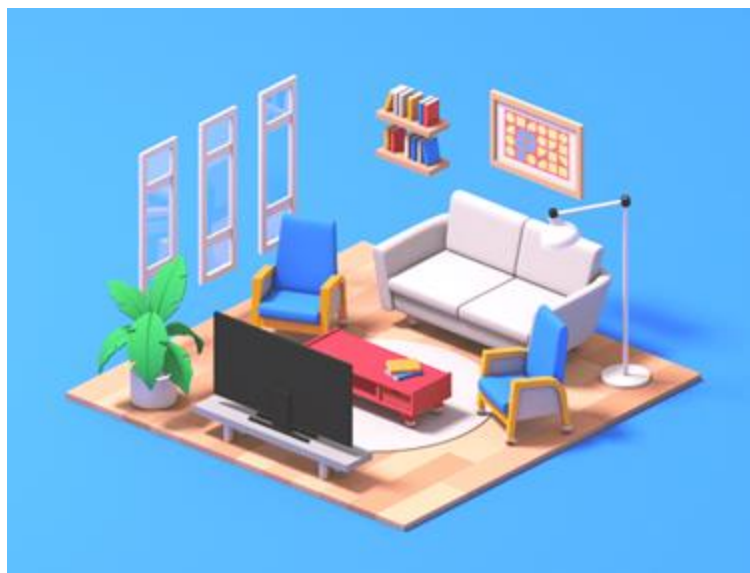
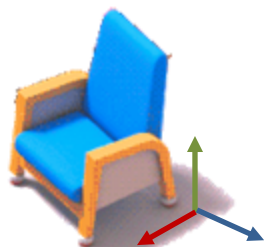




# Grafo de Cena

## Vantagens:

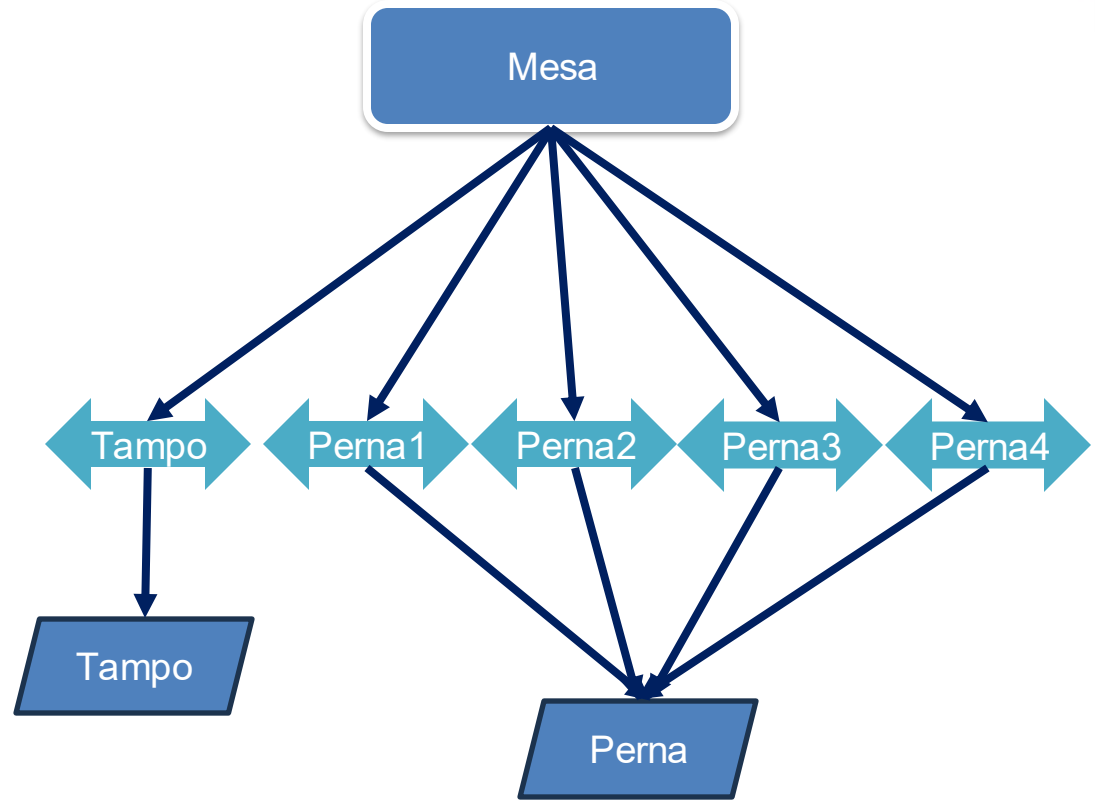
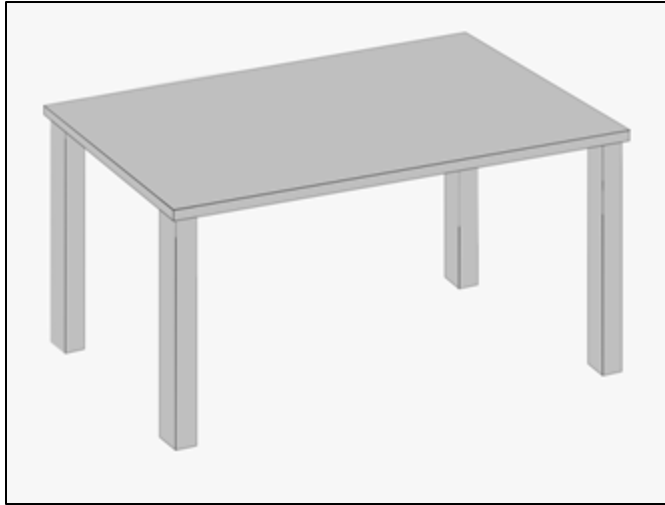
- Permite definições de objetos nos seus sistemas de coordenadas
- Permite o uso de objetos várias vezes numa mesma cena
- Permite processamento pela hierarquia
- Permite animações de articulações de forma simples



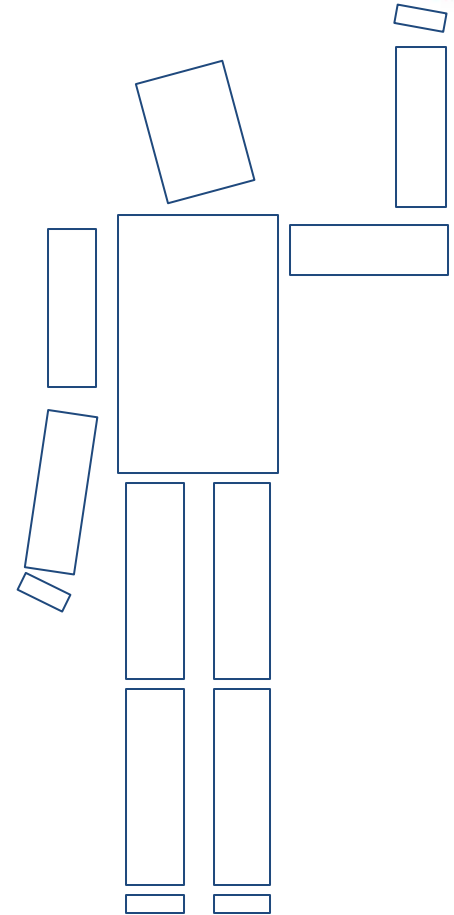
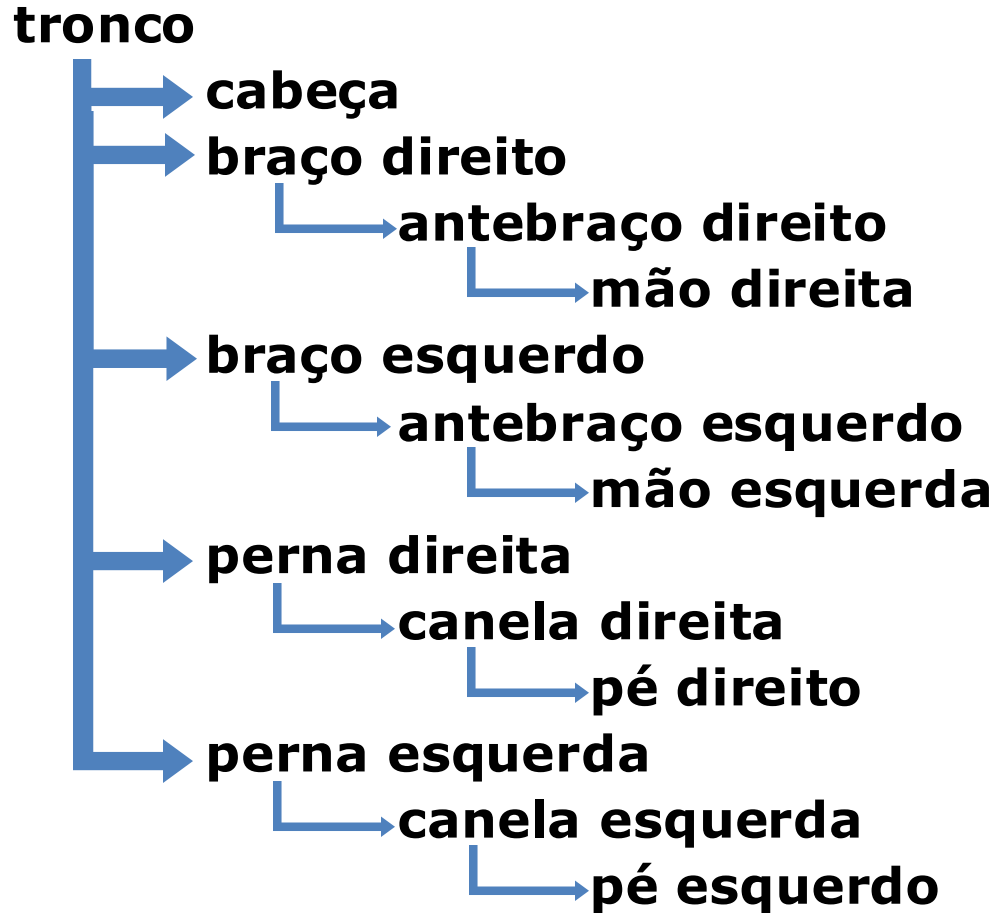
# Representação Hierárquica

- Cada grupo pode conter subgrupos e/ou objetos geométricos.
- Cada grupo possui uma transformação relativa ao seu grupo pai.
- A transformação em um nó folha corresponde à concatenação de todas as transformações ao longo do caminho do nó raiz até a folha.
- A alteração da transformação de um grupo afeta todos os nós subsequentes.
- O grafo permite edição de alto nível: ao modificar apenas um nó, todos os nós relacionados são atualizados.

# Instanciação – partes do modelo

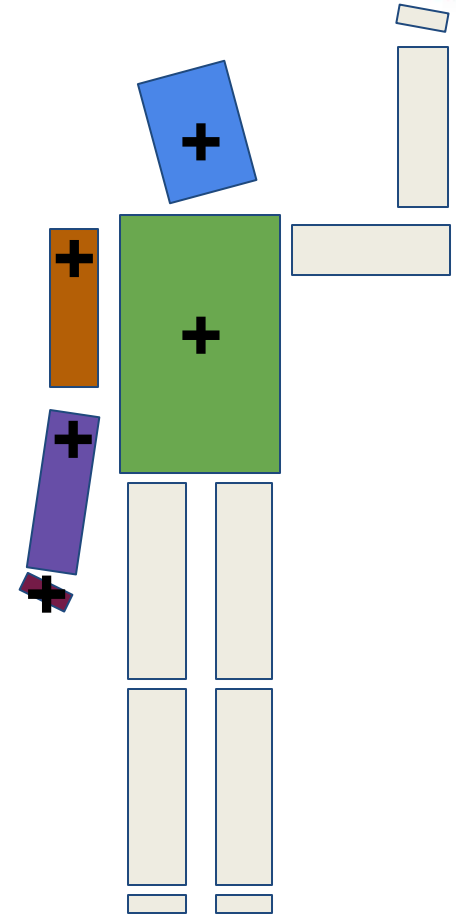
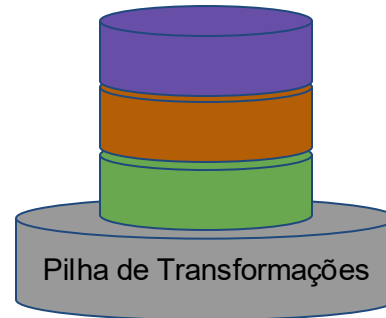


# Esqueleto – Representação Hierárquica



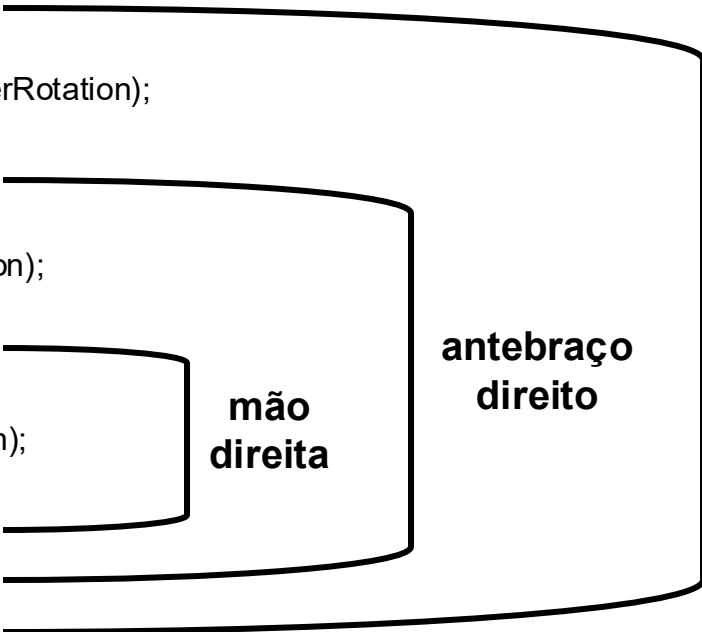
# Implementando Representação Hierárquica

```
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

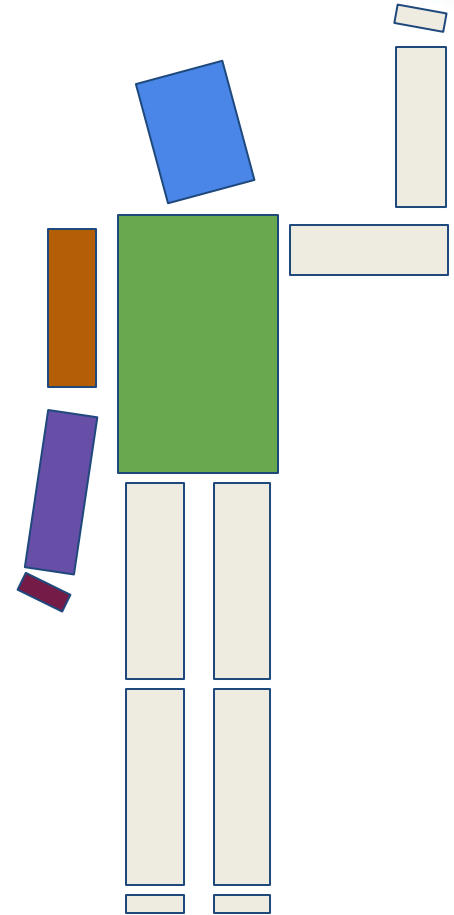
```
translate(0, 5);  
drawTorso();  
pushmatrix(); // armazena a matriz de transformação atual na pilha  
translate(0, 5); // matriz de transformação é atualizada pela translação  
rotate(headRotation); // matriz de transformação é atualizada pela rotação  
drawHead();  
popmatrix(); // recupera matriz 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();
```



**mão direita**

**antebraço direito**

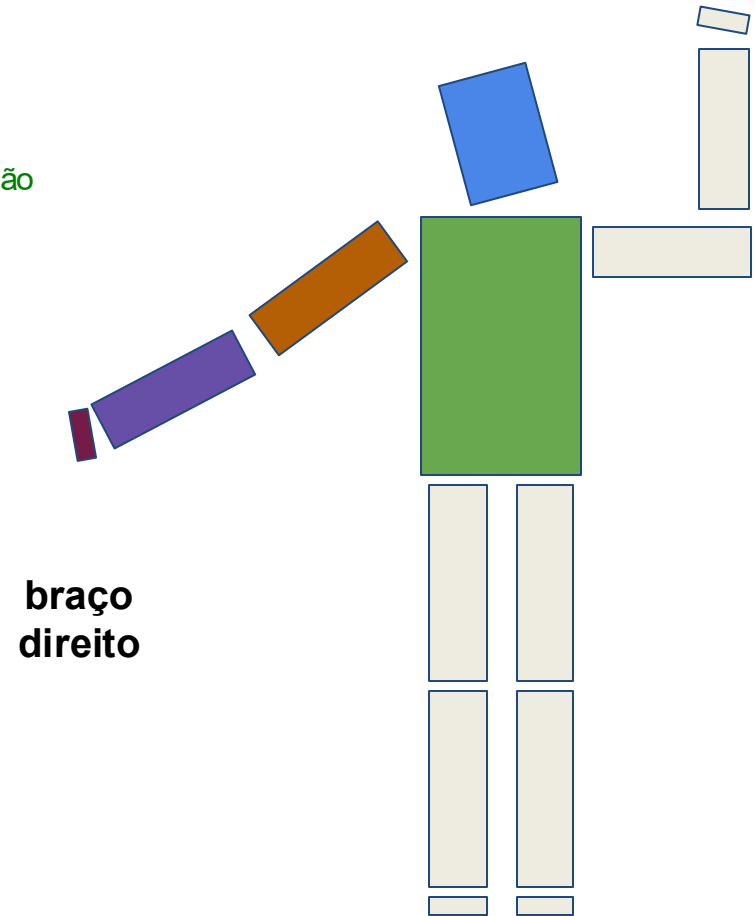
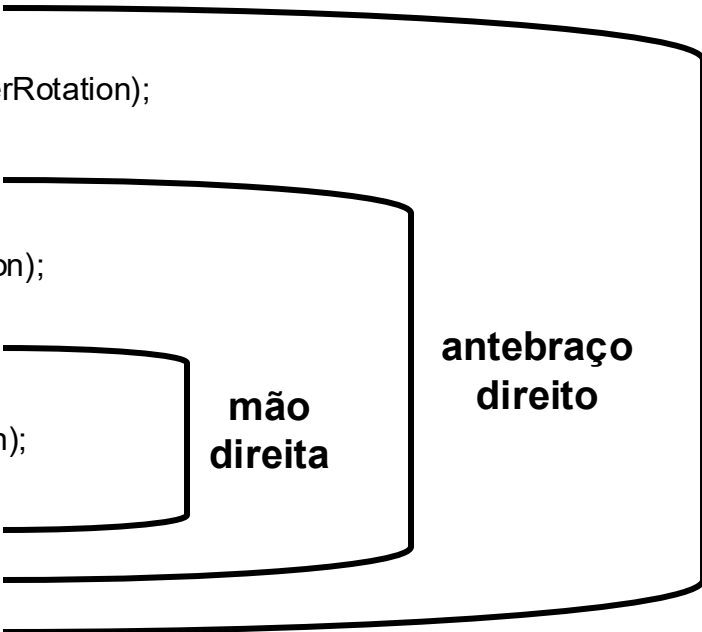
**braço direito**





# Implementando Representação Hierárquica

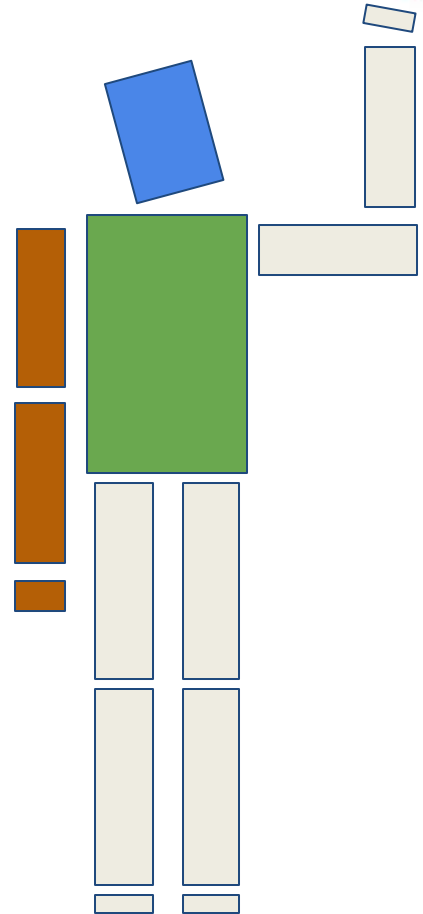
```
translate(0, 5);  
drawTorso();  
pushmatrix(); // armazena a matriz de transformação atual na pilha  
translate(0, 5); // matriz de transformação é atualizada pela translação  
rotate(headRotation); // matriz de transformação é atualizada pela rotação  
drawHead();  
popmatrix(); // recupera matriz 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();
```



# Cena em X3D

```
<Transform>
  <Shape> <!-- Tronco -->
    <Box size="2 3 1"/>
    <Appearance><Material emissiveColor='0 1 0' /></Appearance>
  </Shape>
  <Transform translation="0 2.5 0" rotation="0 0 1 0.4">
    <Shape> <!-- Cabeça -->
      <Box size="1 1 1"/>
      <Appearance><Material emissiveColor='0 0 1' /></Appearance>
    </Shape>
  </Transform>
  <Transform translation="-1.6 0.3 0">
    <Shape> <!-- Braço -->
      <Box size="0.5 1.8 0.5"/>
      <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
    </Shape>
    <Transform translation="0.0 -2.0 0">
      <Shape> <!-- Antebraço -->
        <Box size="0.5 1.8 0.5"/>
        <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
      </Shape>
      <Transform translation="0.0 -1.2 0">
        <Shape> <!-- Mão -->
          <Box size="0.5 0.2 0.5"/>
          <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
        </Shape>
      </Transform>
    </Transform>
  </Transform>
</Transform>
```

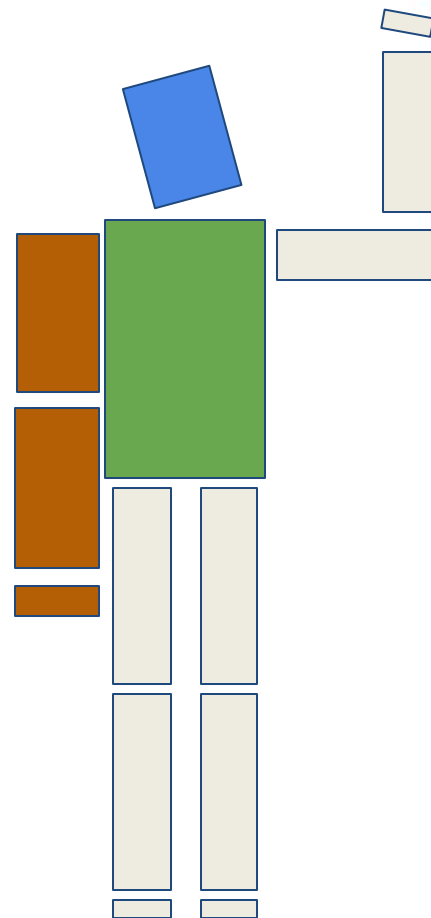
**E se quisermos um braço mais musculoso?**



# Cena em X3D

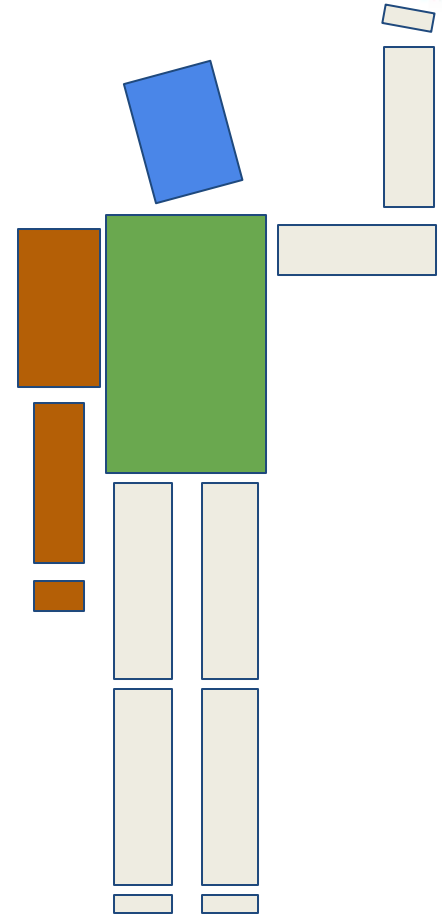
```
<Transform>
  <Shape> <!-- Tronco -->
    <Box size="2 3 1"/>
    <Appearance><Material emissiveColor='0 1 0' /></Appearance>
  </Shape>
  <Transform translation="0 2.5 0" rotation="0 0 1 0.4">
    <Shape> <!-- Cabeça -->
      <Box size="1 1 1"/>
      <Appearance><Material emissiveColor='0 0 1' /></Appearance>
    </Shape>
  </Transform>
  <Transform translation="-1.6 0.3 0" scale="1.5 1 1.5">
    <Shape> <!-- Braço -->
      <Box size="0.5 1.8 0.5"/>
      <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
    </Shape>
    <Transform translation="0.0 -2.0 0">
      <Shape> <!-- Antebraço -->
        <Box size="0.5 1.8 0.5"/>
        <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
      </Shape>
      <Transform translation="0.0 -1.2 0">
        <Shape> <!-- Mão -->
          <Box size="0.5 0.2 0.5"/>
          <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
        </Shape>
      </Transform>
    </Transform>
  </Transform>
</Transform>
```

Como resolver para só o braço?



# Cena em X3D

```
<Transform>
  <Shape> <!-- Tronco -->
    <Box size="2 3 1"/>
    <Appearance><Material emissiveColor='0 1 0' /></Appearance>
  </Shape>
  <Transform translation="0 2.5 0" rotation="0 0 1 0.4">
    <Shape> <!-- Cabeça -->
      <Box size="1 1 1"/>
      <Appearance><Material emissiveColor='0 0 1' /></Appearance>
    </Shape>
  </Transform>
  <Transform translation="-1.6 0.3 0" >
    <Transform scale="1.5 1 1.5">
      <Shape> <!-- Braço -->
        <Box size="0.5 1.8 0.5"/>
        <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
      </Shape>
    </Transform>
    <Transform translation="0.0 -2.0 0">
      <Shape> <!-- Antebraço -->
        <Box size="0.5 1.8 0.5"/>
        <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
      </Shape>
      <Transform translation="0.0 -1.2 0">
        <Shape> <!-- Mão -->
          <Box size="0.5 0.2 0.5"/>
          <Appearance><Material emissiveColor='1 0.5 0' /></Appearance>
        </Shape>
      </Transform>
    </Transform>
  </Transform>
</Transform>
```



# Perguntas

## Referência:

O cubo vermelho foi criado na origem

## Pergunta:

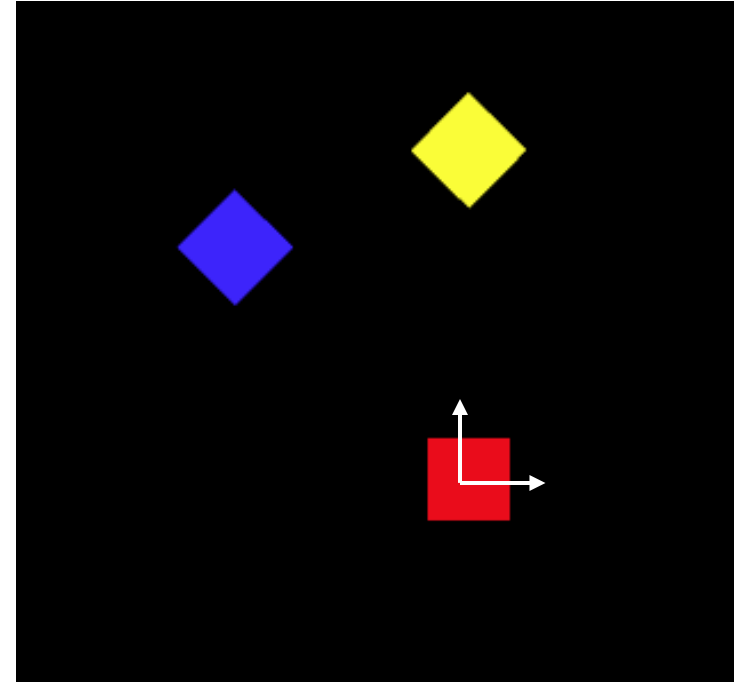
Qual cubo foi:

1º rotacionado 45° e depois transladado na vertical?

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

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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



# Perguntas

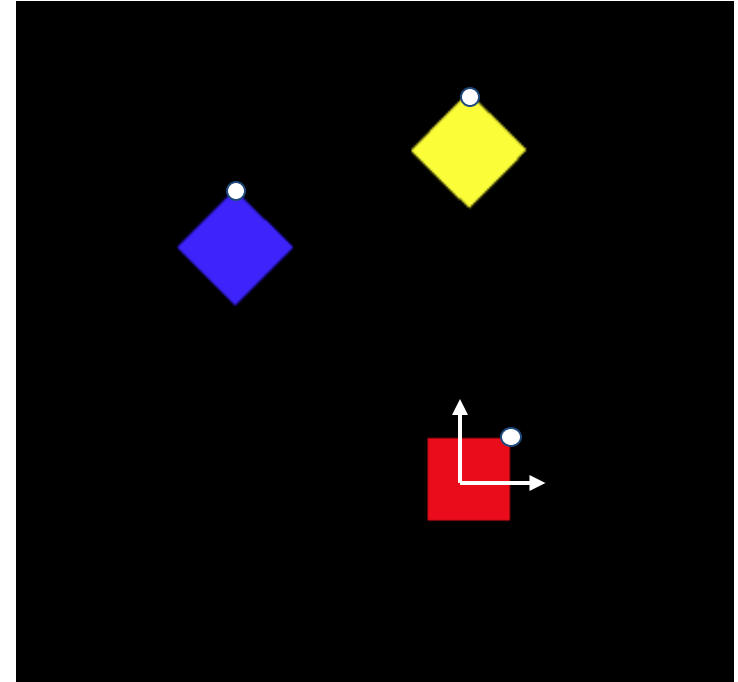
$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.71 & -0.71 & 0 & 0 \\ 0.71 & 0.71 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Primeiro Rotaciona depois Translada

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

Primeiro Translada depois Rotaciona

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





# Perguntas

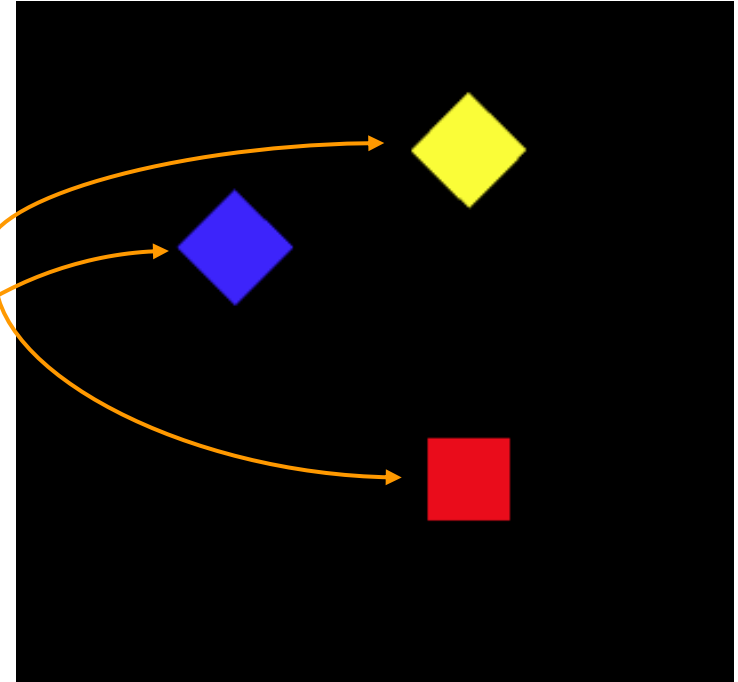
```
<Scene>  
<Viewpoint position="0 0 15" fieldOfView="0.7854"/>
```

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

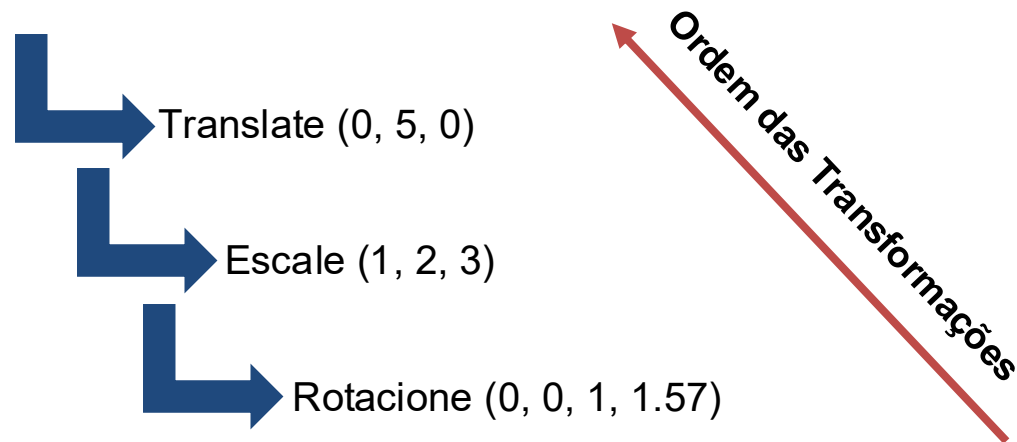
```
<Transform rotation="0 0 1 0.79">  
  <Transform translation="0 4 0">  
    <Shape>  
      <Box size="1 1 1"/>  
      <Appearance>  
        <Material emissiveColor="0 0 1"/>  
      </Appearance>  
    </Shape>  
  </Transform>  
</Transform>
```

```
</Scene>
```



# Usando Matrizes

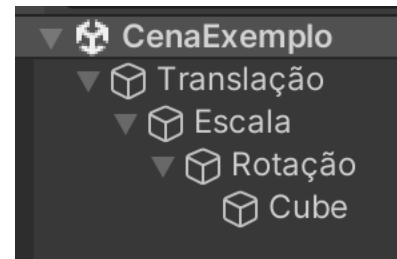
Na prática multiplicamos as matrizes e empilhamos.  
Por exemplo:



```
<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()**

**Escala (1,2,3)**

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

**PushMatrix()**

**Rotação (0, 0, 1, 1.57)**




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

**PushMatrix()**

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

PILHA

# 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

# ATIVIDADE:

Acesse o notebook no site da disciplina.

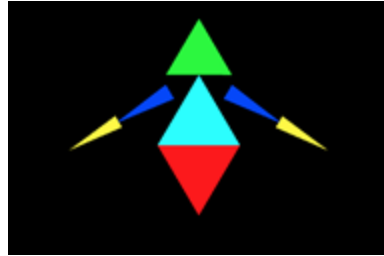
Crie uma cópia para você e realize todos os exercícios.

Voltamos em 30 minutos?

# Terceira parte do projeto 1



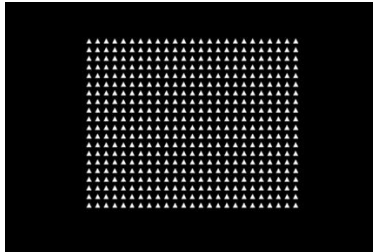
letras.x3d



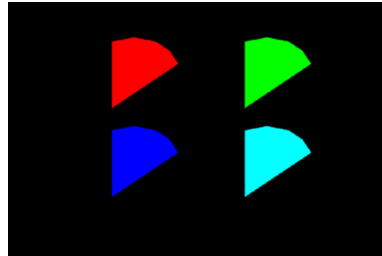
avatar.x3d



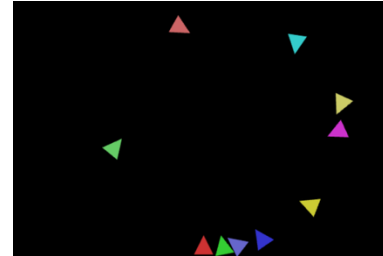
tiradetrinagulos.x3d



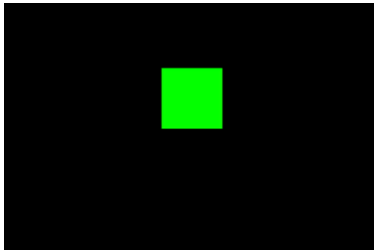
bound500.x3d



leques.x3d



girando.x3d



vertices10.x3d



estrela.x3d

<https://lpsoares.github.io/Renderizador/>



# Atualizando o Fork do Repositório

```
git remote add upstream https://github.com/lpsoares/Renderizador
```

```
git pull upstream master
```

Eventualmente:

```
git fetch upstream
```

```
git merge upstream/master
```

# Computação Gráfica

Luciano Soares

<lpsoares@insper.edu.br>

Fabio Orfali

<fabioO1@insper.edu.br>