

# Computação Gráfica

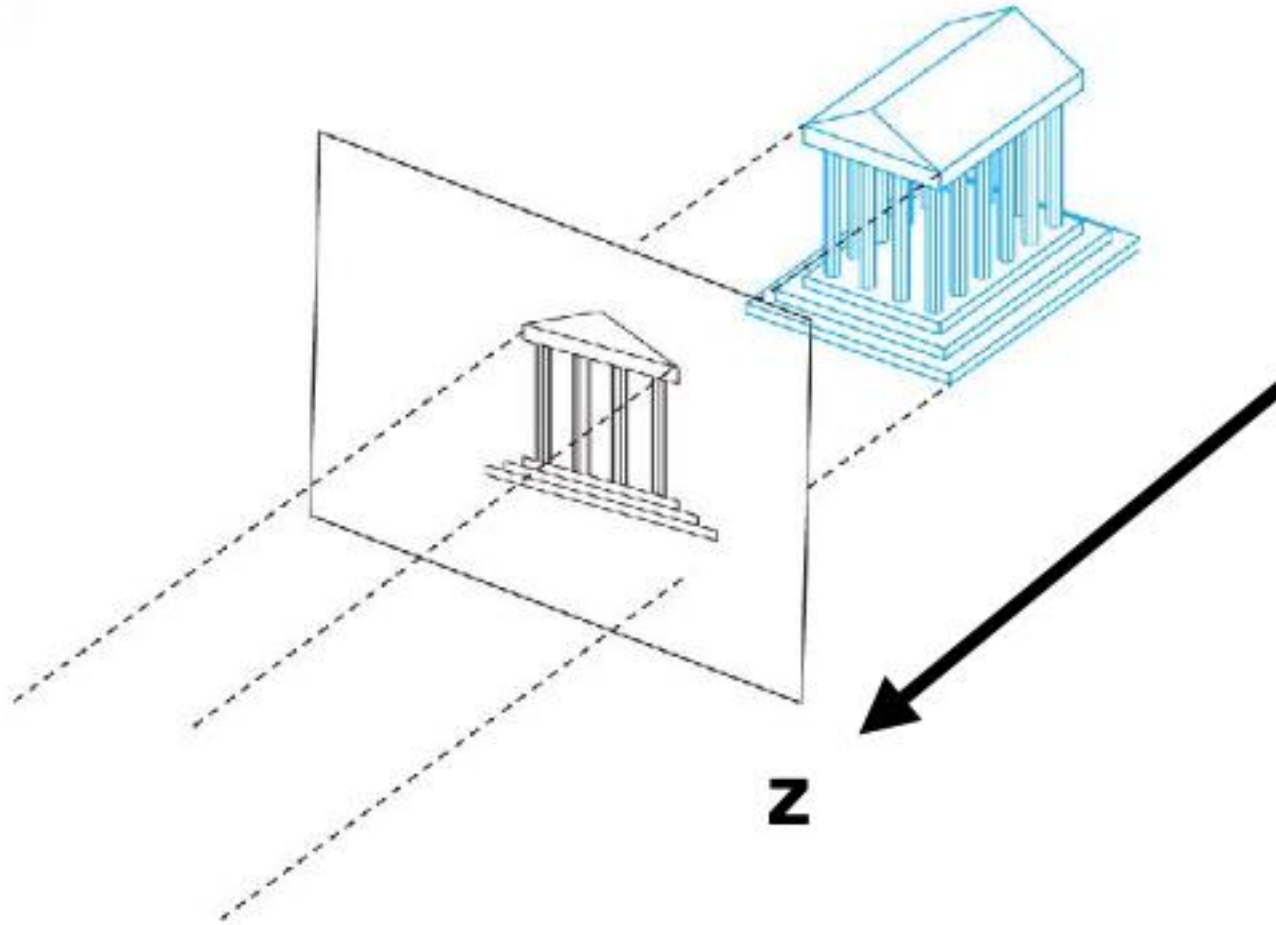
Aula 6: Projeções

Kahoot

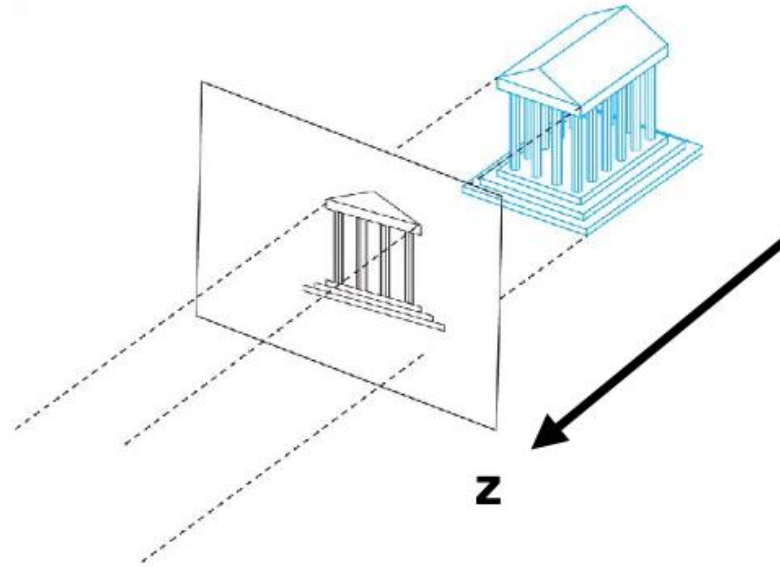
# Kahoot!

Entrar em Kahoot.it : <https://kahoot.it/>

# Do espaço da câmera (3D) para Plano de Imagem (2D)



# Do espaço da câmera (3D) para Plano de Imagem (2D)



Como transformar do espaço da câmera 3D em um plano de imagem 2D?

Uma opção: projeção ortográfica (basta excluir  $z$ )

Útil, para, por exemplo, para desenhos de engenharia

Mas será que é conveniente para tudo?

# Perspectiva



Créditos: [jefflynchphoto.com](http://jefflynchphoto.com)



# Perspectiva na Arte



Paradiesgärtlein; c. 1410

# Perspectiva da Idade Média para Renascentismo

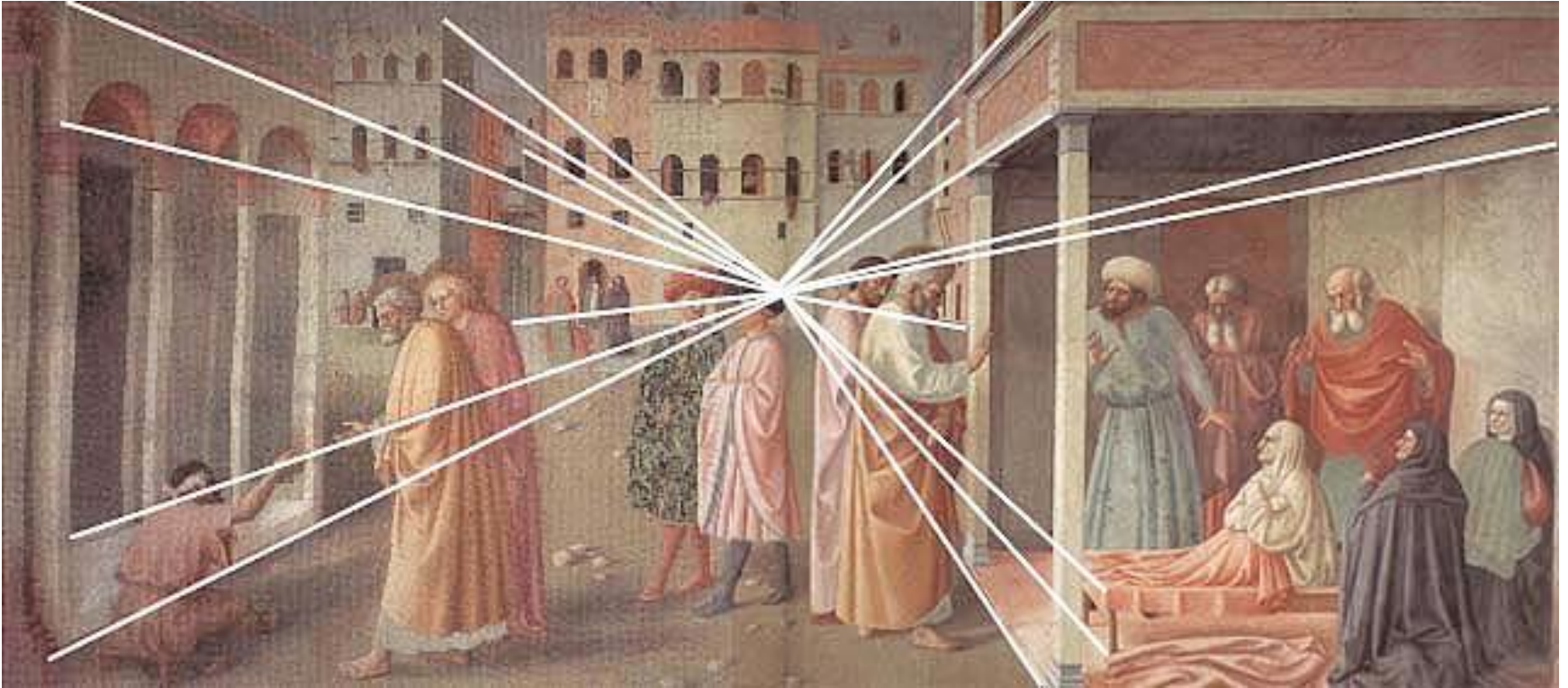


Cristo davanti a Caifa; por Giotto; c.1305

**"As vigas do teto convergem de maneira convincente, mas a projeção geométrica expõe a falha em convergir com precisão."**



# Perspectiva no Renascentismo



Guarigione dello storpio e resurrezione di Tabita; por Masolino; 1424

**"O primeiro uso conhecido de convergência central precisa foi por Masolino em 1425."**

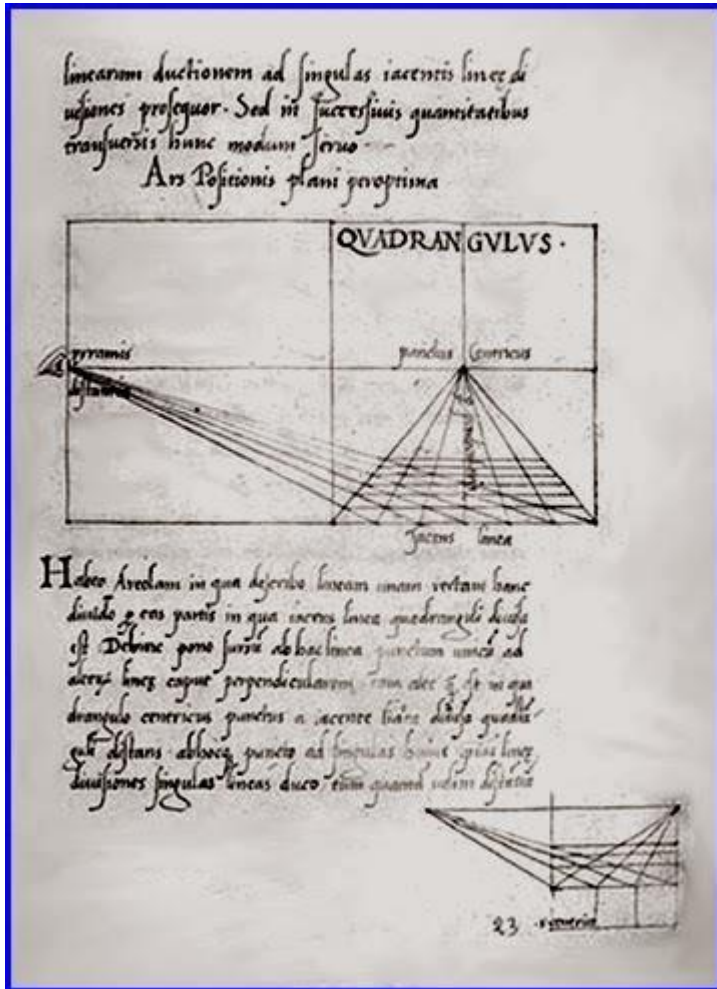


# De pictura

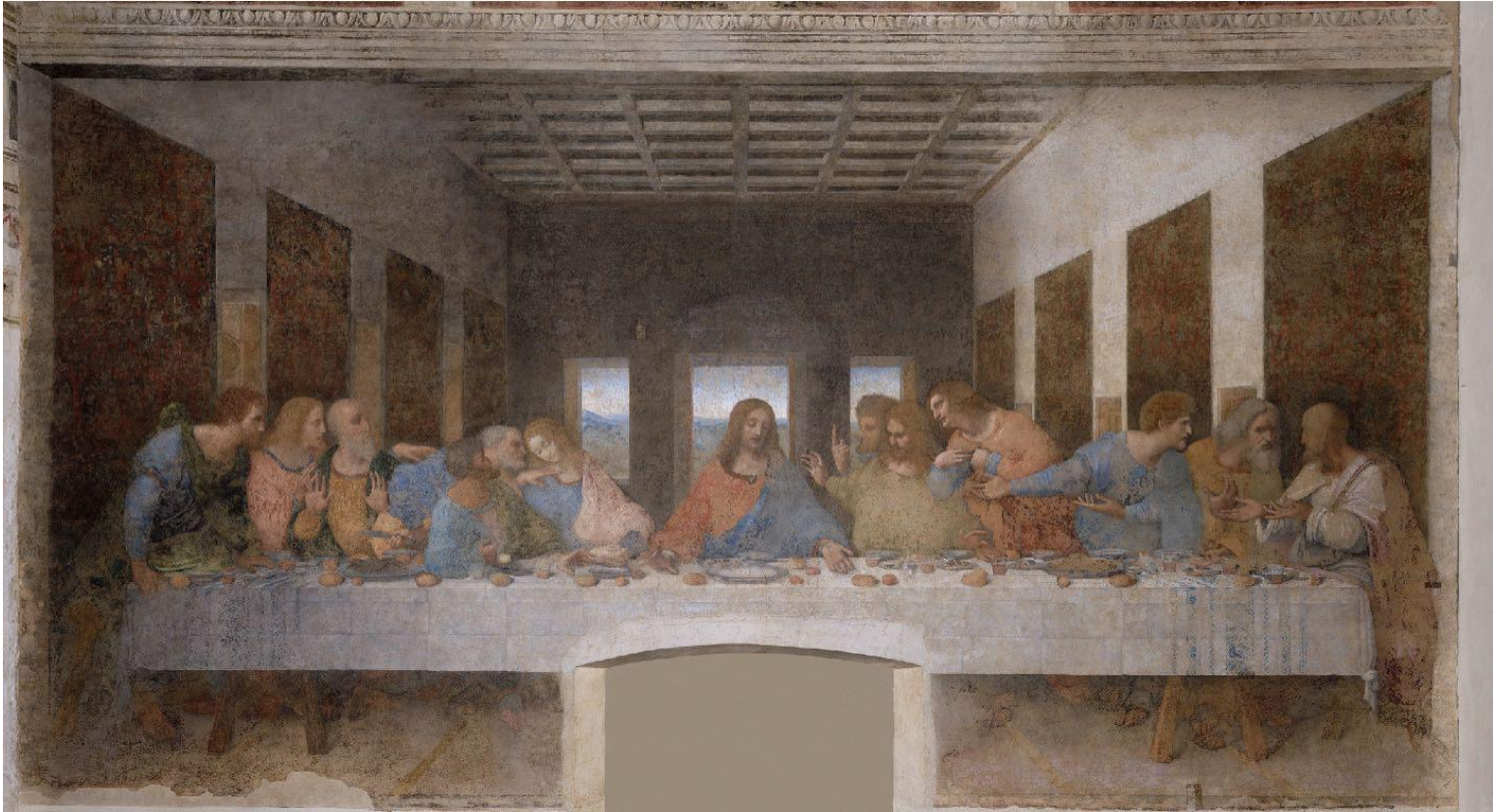
Leon Battista Alberti escreveu *De pictura* pela primeira vez em 1435, no qual descreveu, em palavras e ilustrações, os princípios geométricos da perspectiva na pintura. No ano seguinte, ele escreveu uma versão no dialeto toscano, *Della pittura*.

Hoje, apenas 20 manuscritos latinos e três manuscritos toscanos sobrevivem. Foi publicado pela primeira vez na imprensa em Basel em 15XX.

Fonte: <https://euclidinflorence.wordpress.com/exhibition-catalog/albertis-de-pittura/>



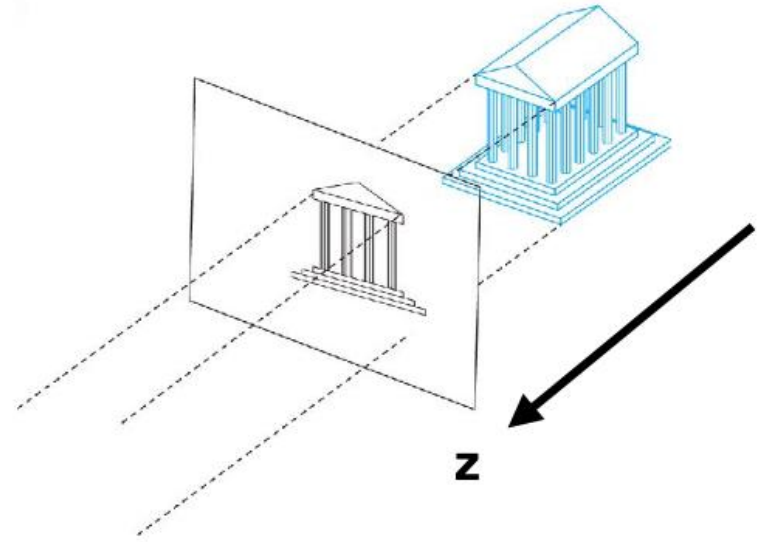
# Perspectiva na Arte



A Última Ceia; por Leonardo da Vinci; 1499

# Perspectiva na Arte

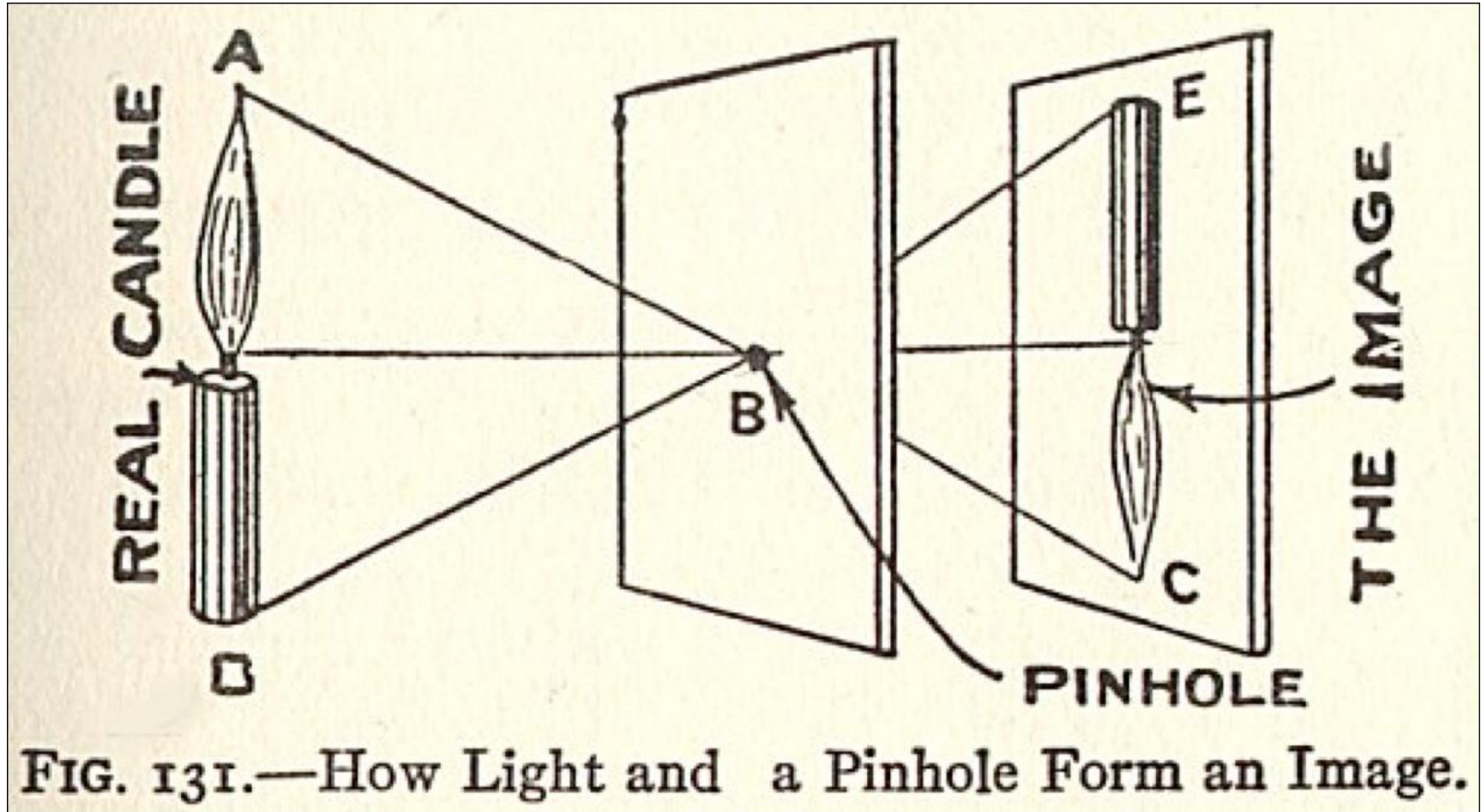
Uma projeção ortográfica teria o mesmo resultado na obra da última ceia de Leonardo da Vinci?



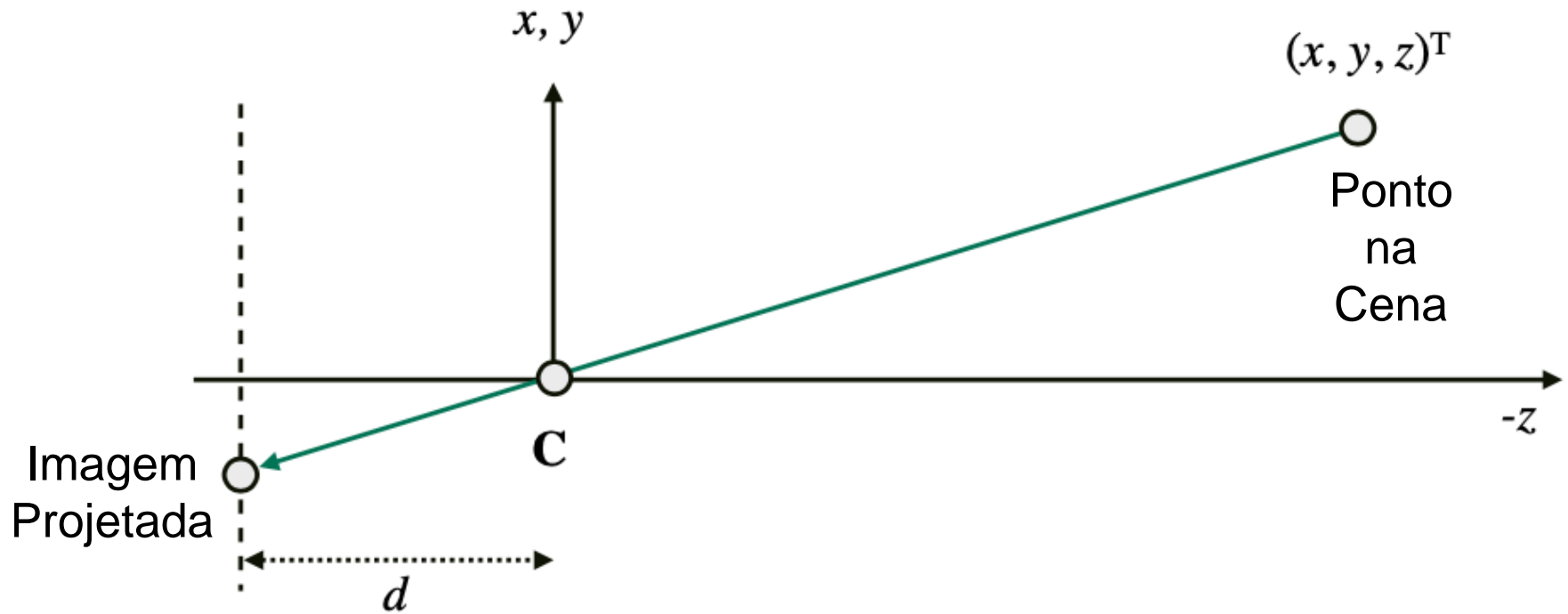
**NÃO**



# Relembrando Câmera Pinhole

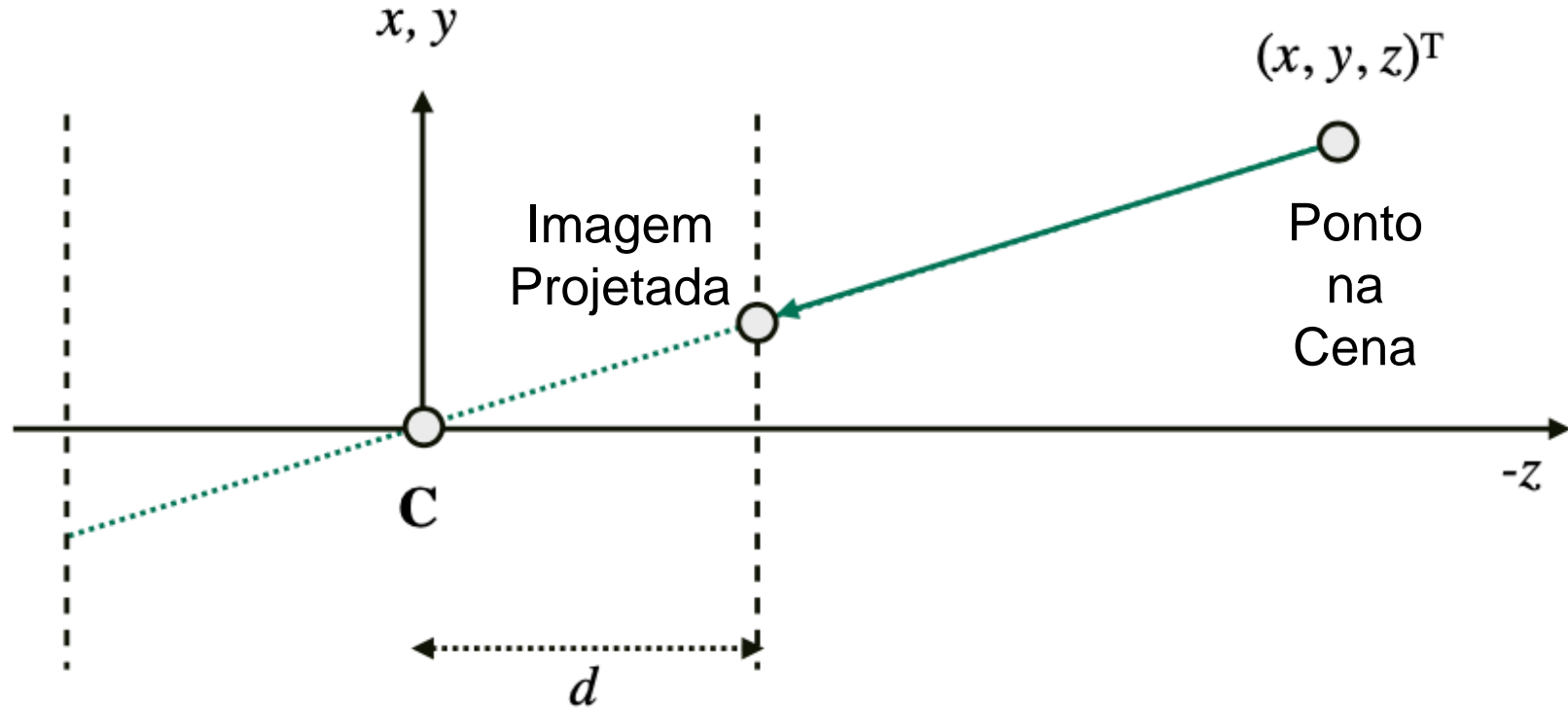


# Relembrando Câmera Pinhole



A imagem vai estar invertida (como numa câmera pinhole real)

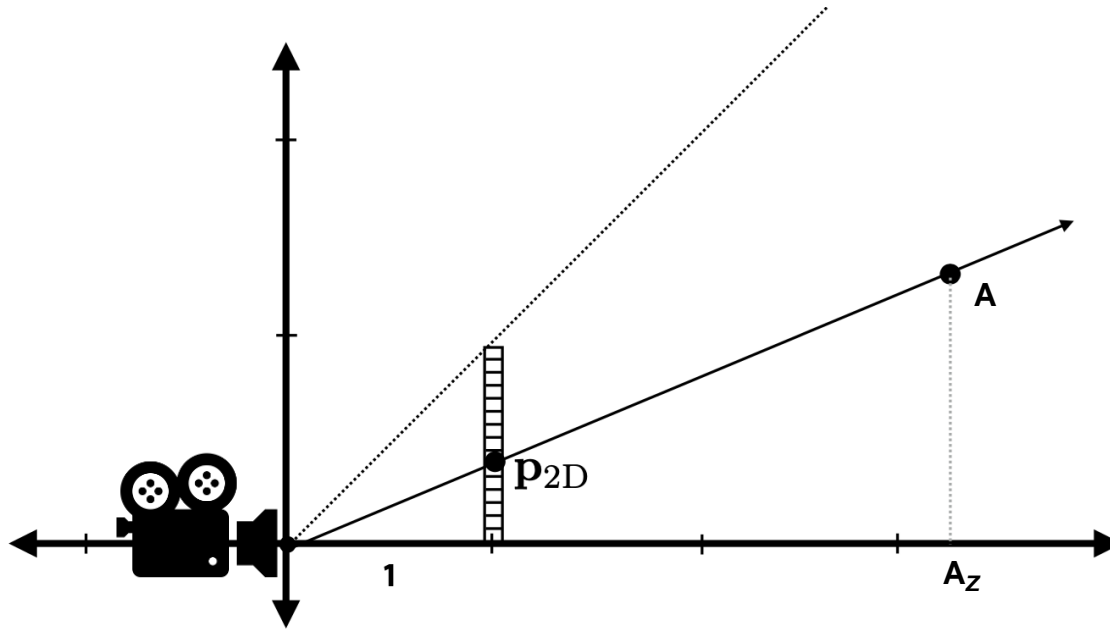
# Transformação Projetiva de Câmera Pinhole



Não está mais invertida



# Projeção Perspectiva Básica (2D)



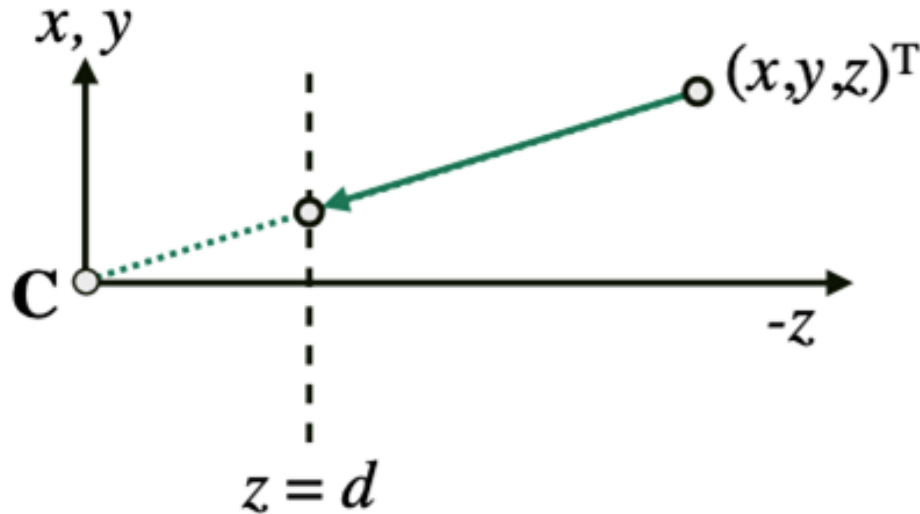
Resultado da projeção perspectiva desejada (ponto 2D)

$$P_{2D} = \begin{bmatrix} \frac{A_x}{A_z} \\ \frac{A_y}{A_z} \end{bmatrix}$$

# Transformações Projetivas

Projeção em perspectiva padrão:

- Centro de projeção:  $(0, 0, 0)^T$
- Plano de imagem em  $z = d$



$$proj \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d \cdot \frac{x}{z} \\ d \cdot \frac{y}{z} \end{pmatrix}$$

# Transformações Projetivas

Projeção em perspectiva padrão:

- Centro de projeção:  $(0, 0, 0)^T$
- Plano de imagem em  $z = d$

Projeção perspectiva:

- Precisa de divisão por  $z$
- Como representar por matriz?

$$proj \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d \cdot \frac{x}{z} \\ d \cdot \frac{y}{z} \end{pmatrix}$$

Coordenadas homogêneas!



# Projeção em Coordenadas Homogêneas (3D)

$$\mathbf{p} = \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} \longleftrightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$

$$\mathbf{q} = \mathbf{M} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} \longleftrightarrow \begin{pmatrix} xd/z \\ yd/z \\ d \\ 1 \end{pmatrix}$$

Termo  
diferente de  
zero na linha  
final.

termo final  
que não é  
mais 1 (um)

# Perspectiva: posição da câmera + ângulo de visão



Canon EF Lens Work III

Nesta sequência, o ângulo de visão diminui à medida que a distância da pessoa aumenta, para assim manter o tamanho da pessoa na imagem.

Observe a mudança dramática na perspectiva do fundo.

# Composição Perspectiva

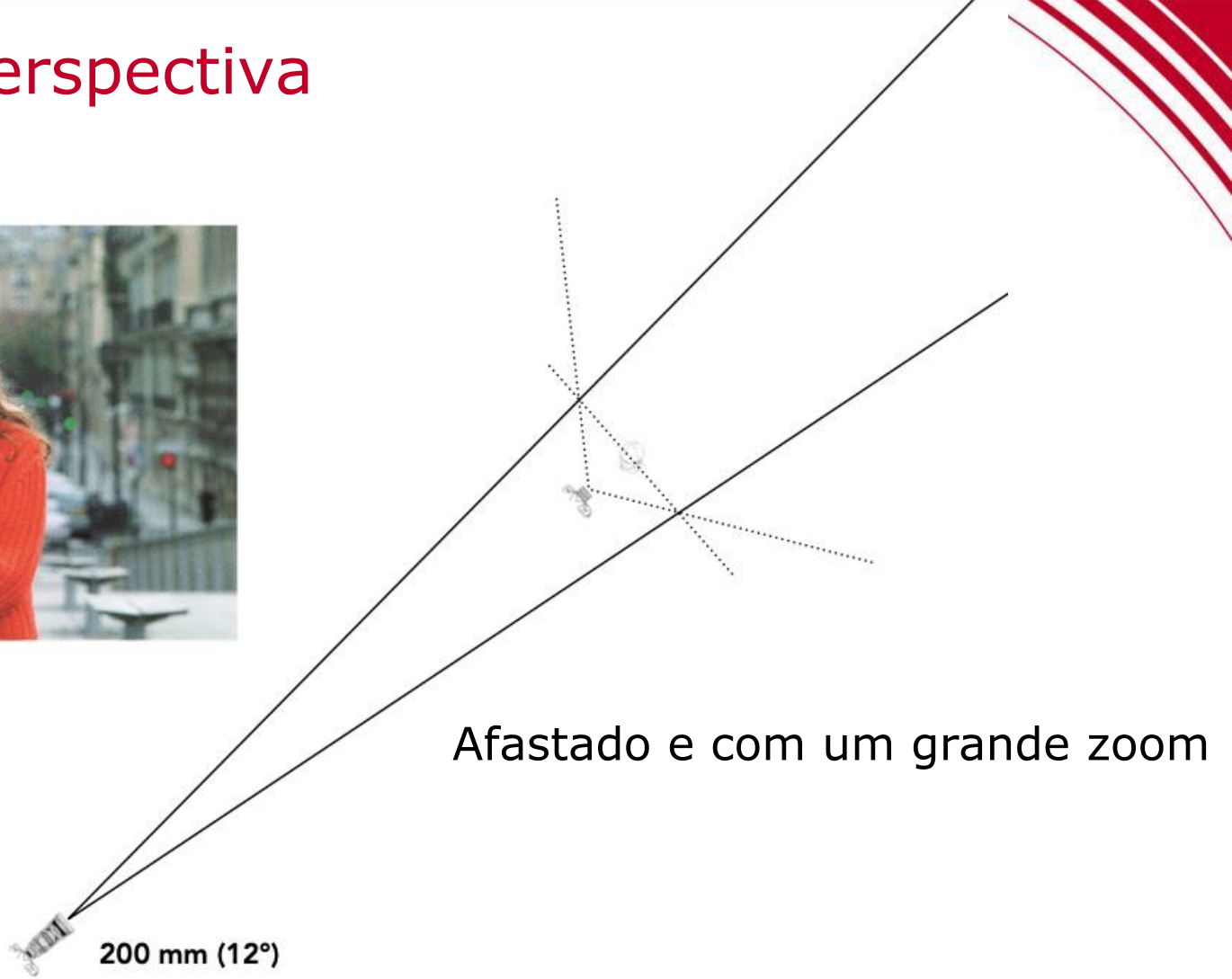


16 mm (110°)

Próximo e com um pequeno zoom

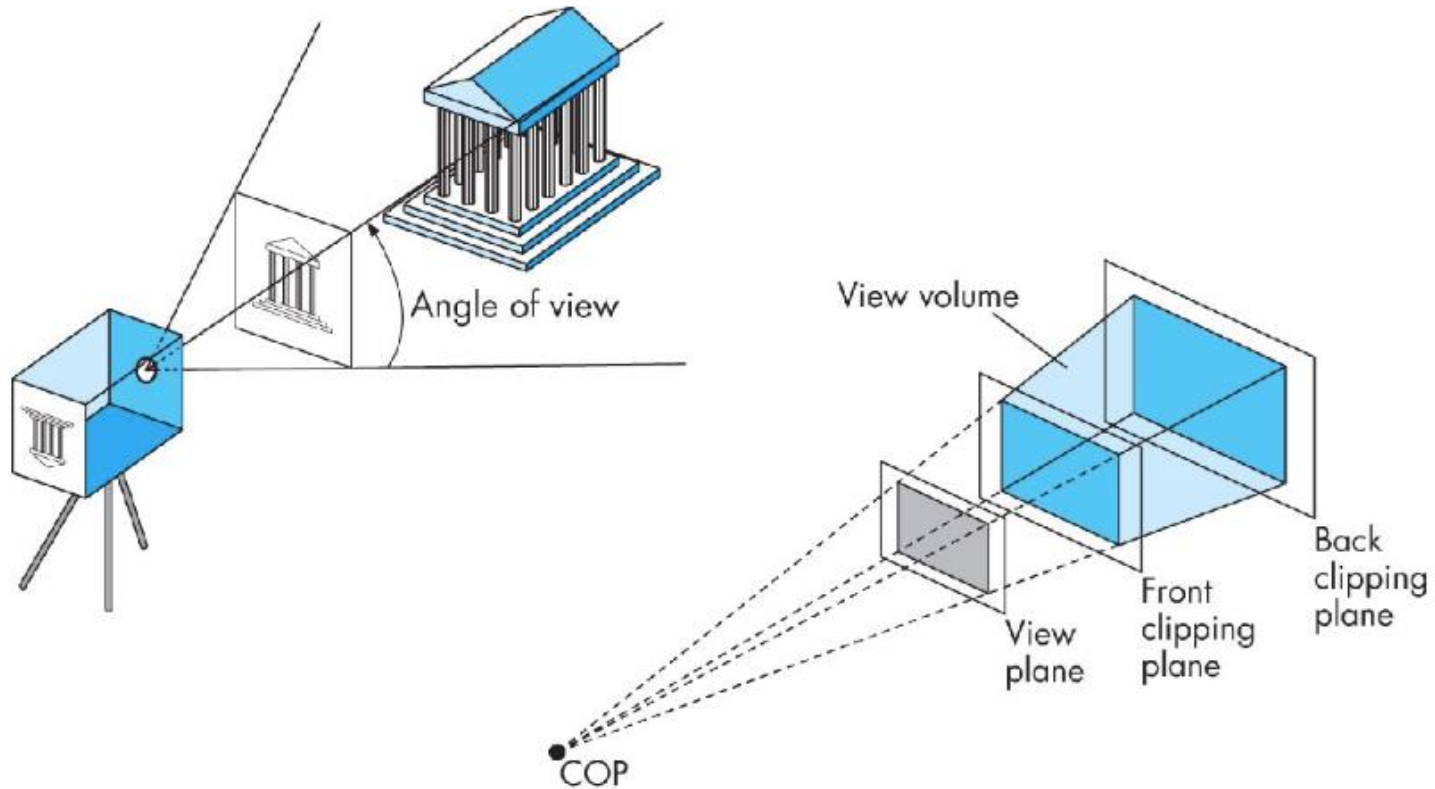


# Composição Perspectiva



Afastado e com um grande zoom

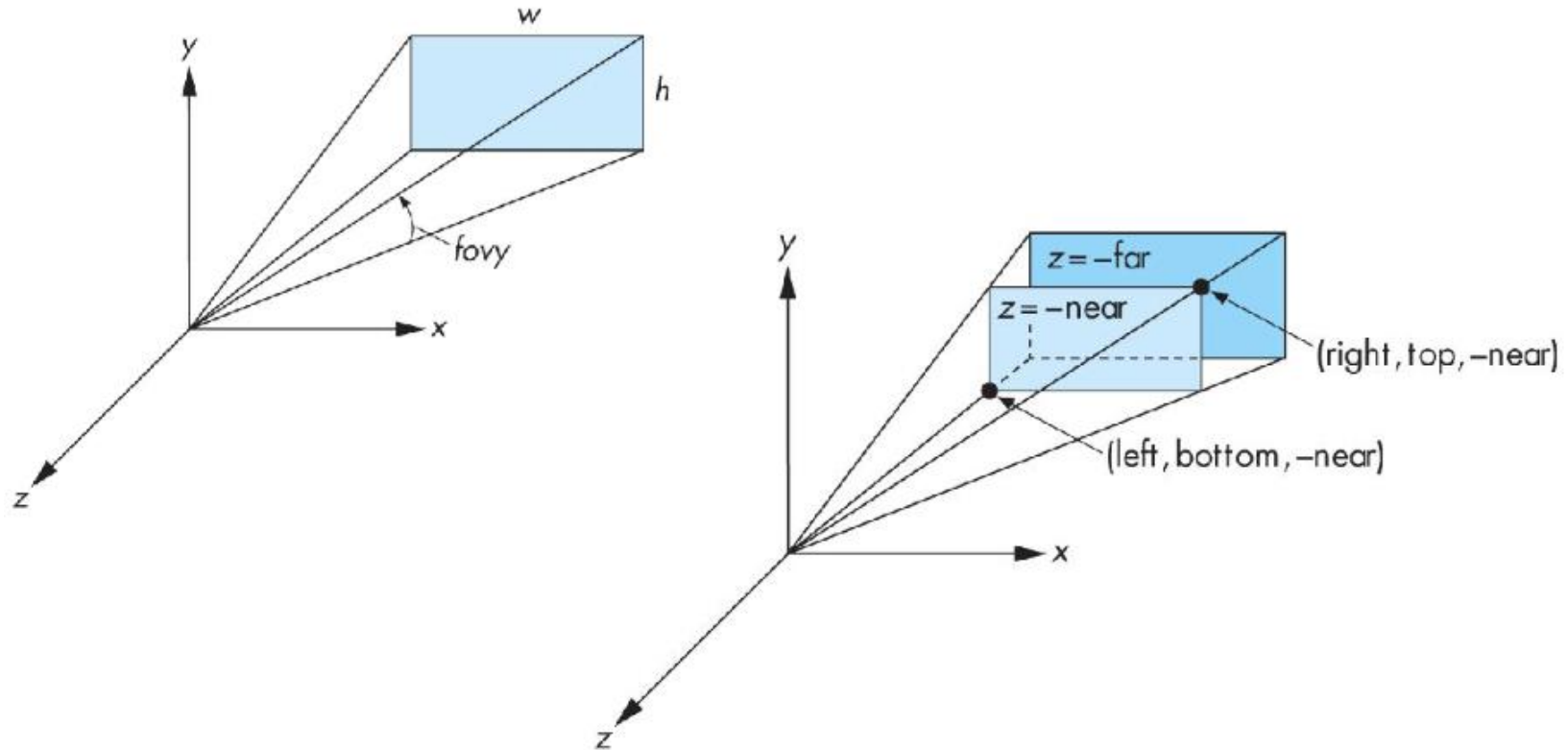
# Especificando a Projeção Perspectiva



COP = Center Of Projection

do livro: Angel and Shreiner, Interactive Computer Graphics

# Especificando o Volume de Visão Perspectiva



# Especificando o Volume de Visão Perspectiva

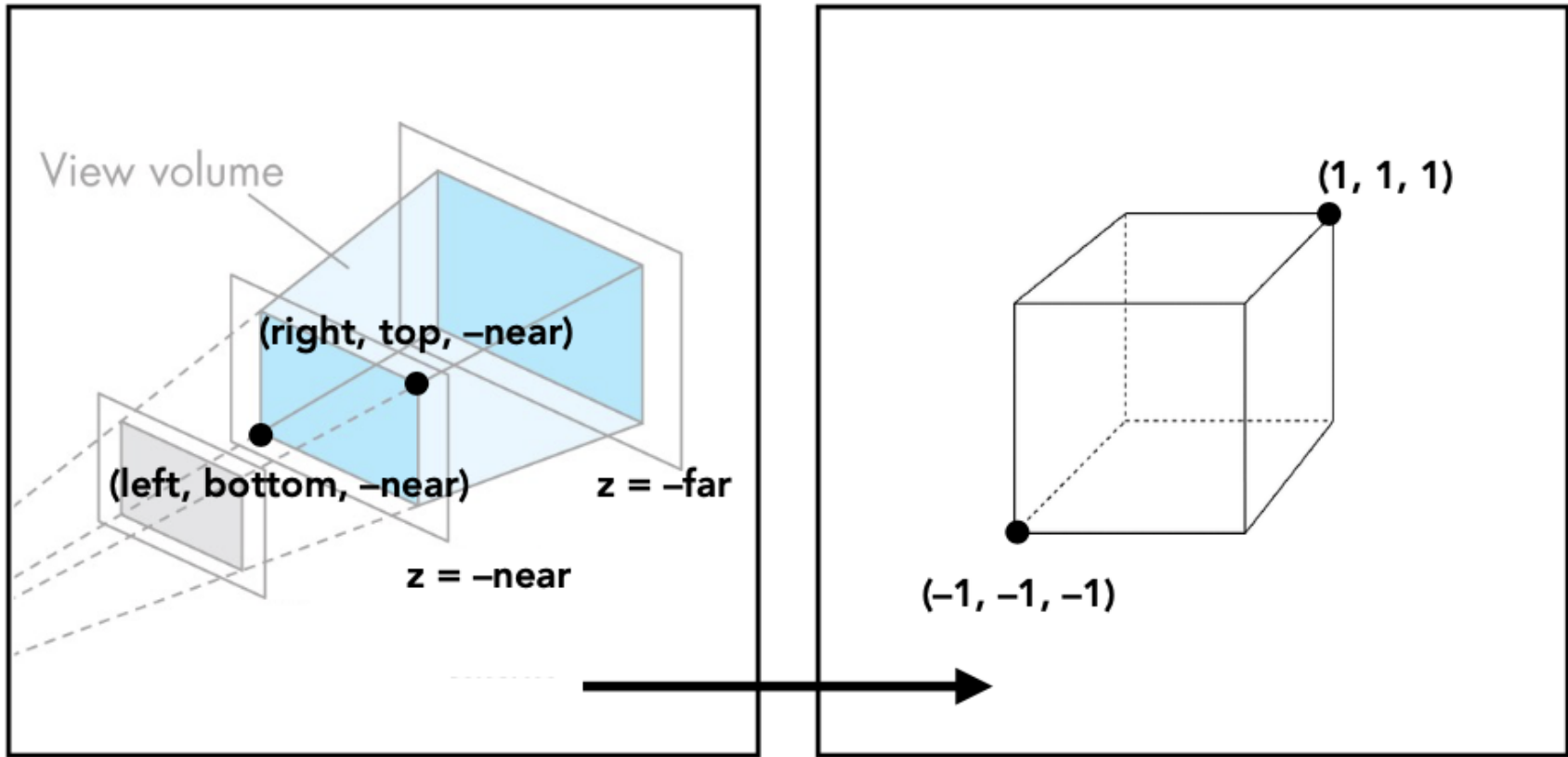
Parametrizado por

- fovy: campo de visão angular vertical (eixo y)
- razão de aspecto: largura / altura do campo de visão
- near: profundidade do plano de corte próximo
- far: profundidade do plano de recorte distante

Quantidades derivadas

- $\text{top} = \text{near} * \tan(\text{fovy})$
- $\text{bottom} = -\text{top}$
- $\text{right} = \text{top} * \text{aspect}$
- $\text{left} = -\text{right}$

# Implementação de Projeção Perspectiva



Coordenadas de câmera (view)

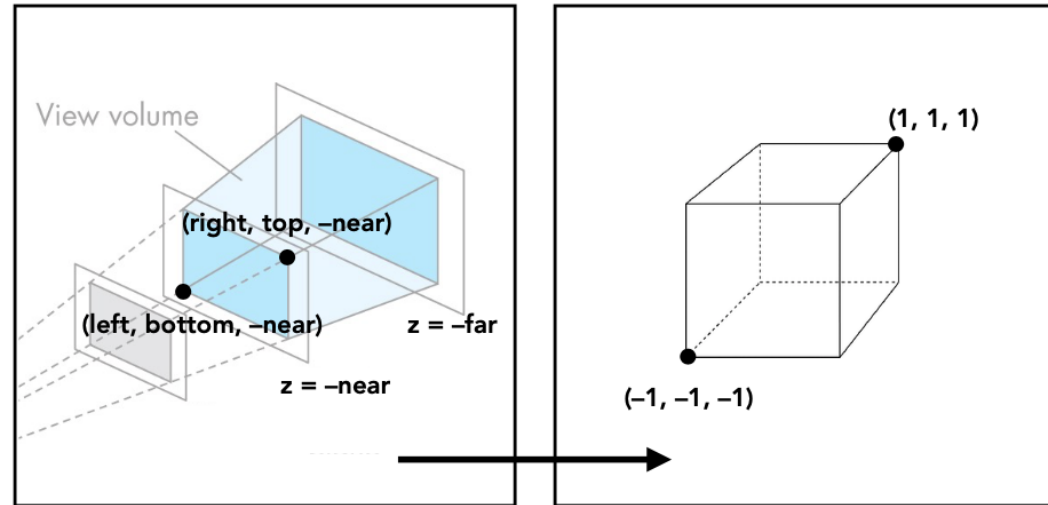
Normalized Device Coords (NDC)

já iremos para as coordenadas da tela



# Transformações Perspectivas

- Não precisa ser simétrica em relação ao eixo z, porém, para simplificar, assumimos por enquanto que é simétrica
- A transformação preservará as informações de profundidade

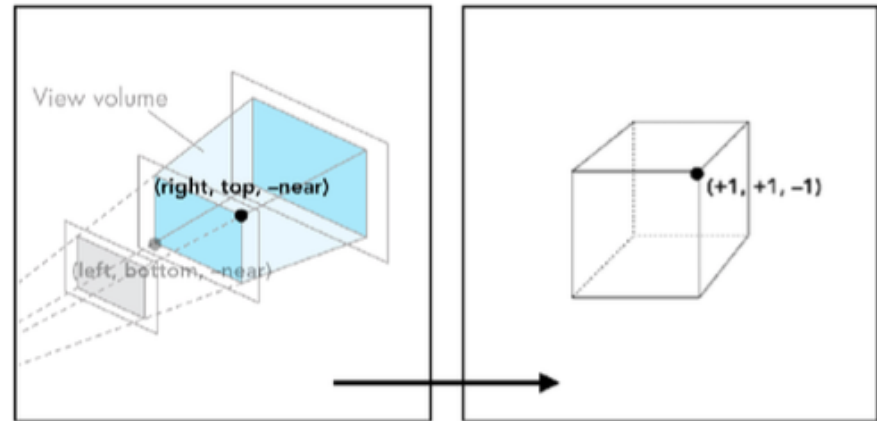


# Matriz de Transformação Perspectiva

$$\mathbf{P} = \begin{bmatrix} \frac{near}{right} & 0 & 0 & 0 \\ 0 & \frac{near}{top} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{-2far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

# Exemplo com a Matriz Perspectiva

$$\begin{aligned}
 \mathbf{P} &= \begin{bmatrix} \frac{\text{near}}{\text{right}} & 0 & 0 & 0 \\ 0 & \frac{\text{near}}{\text{top}} & 0 & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & \frac{-2\text{far}*\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \text{right} \\ \text{top} \\ -\text{near} \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} \text{near} \\ \text{near} \\ \text{near} * \frac{\text{far}+\text{near}}{\text{far}-\text{near}} - \frac{2\text{far}*\text{near}}{\text{far}-\text{near}} \\ \text{near} \end{bmatrix} \\
 &= \begin{bmatrix} 1 \\ 1 \\ \frac{-\text{far}+\text{near}}{\text{far}-\text{near}} \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}
 \end{aligned}$$



# Divisão Homogênea (Homogeneous Divide)

Fatalmente o quarto valor no vetor dos pontos (valor  $w$ ) após a multiplicação da matriz de projeção perspectiva não será mais um (1). Faça a divisão de todos os elementos por  $w$  para que o quarto valor volte a ser um (1).

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \\ 1 \end{bmatrix}$$

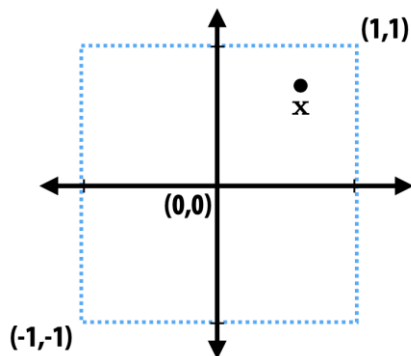


# Transformações para Tela

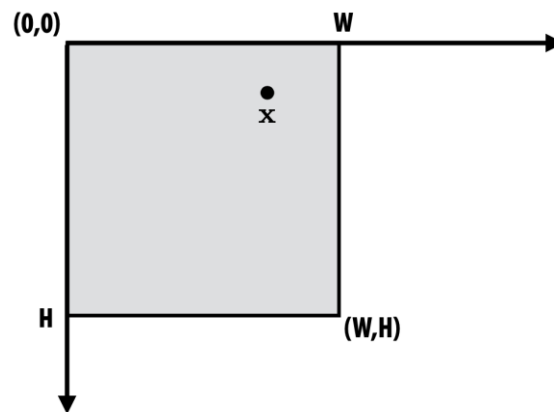
## Screen Transformation

Converter pontos no espaço de coordenadas normalizado da câmera para coordenadas de pixel da tela

sistema normalizado da câmera



Espaço de coordenadas da tela (imagem de  $w \times h$ )



Todos os pontos dentro da região  $(-1,-1)$  a  $(1,1)$  estão na tela

$(1, 1)$  no espaço da câmera mapeia para  $(W, 0)$  no espaço da tela

$(-1, -1)$  no espaço da câmera mapeia para  $(0, H)$  no espaço de tela

Perceba que é preciso espelhar em Y, transladar por  $(1, 1)$  e escalar para  $(W/2, H/2)$

# Transformações para Tela

## Screen Transformation

Qual matriz espelha em Y, transladar por (1, 1) e escalar para (W/2, H/2) ?

$$\begin{array}{ccc} \text{Escala para (W/2, H/2)} & \text{Translate por (1,1)} & \text{Espelha em Y} \\ \begin{bmatrix} \frac{W}{2} & 0 & 0 & 1 \\ 0 & \frac{H}{2} & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \cdot \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \boxed{\begin{bmatrix} \frac{W}{2} & 0 & 0 & \frac{W}{2} \\ 0 & -\frac{H}{2} & 0 & \frac{H}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \end{array}$$

Uma projeção ortogonal do objeto no plano x-y do espaço da câmera (Camera Space) resultará na imagem esperada.

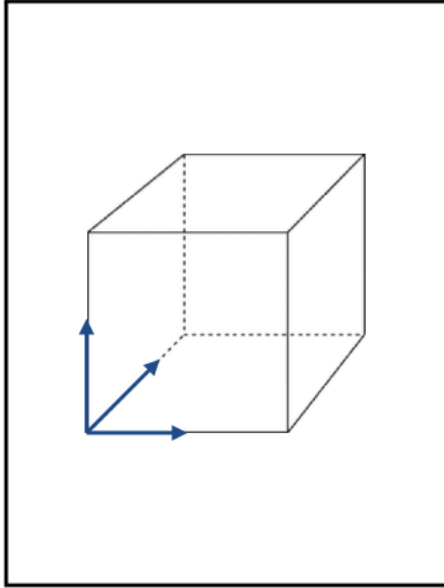
Contudo perceba que quando vamos do espaço da câmera (Camera Space) para o espaço da tela (Screen Space) ainda temos o valor do Z (profundidade), ou seja, podemos dizer o que espaço da tela é um espaço 3D. Essa informação de profundidade será usada no futuro.

# Recapitulando Transformações

Sistemas coordenadas:

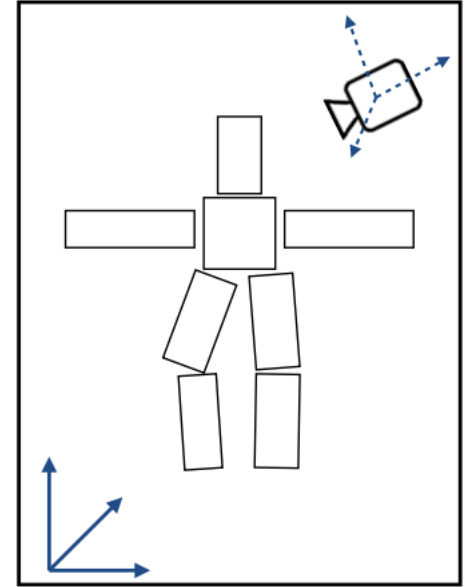
- Coordenadas do objeto (model 3D)
  - Aplicar transformações do modelo ...
- Coordenadas do mundo (world/cena 3D)
  - Aplicar transformação de visualização ...
- Coordenadas da câmera (view/visão 3D)
  - Aplicar transformação de projeção perspectiva
- Coordenadas normalizadas (NDC 3D)
  - Aplicar transformação de tela 2D ...
- Coordenadas da tela (screen 2D)

# Recapitulando Transformações



**Coordenadas do Objeto**

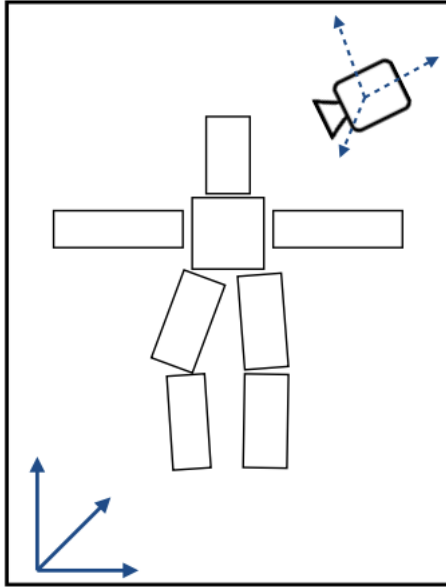
→  
**Modeling  
transforms**



**Coordenadas do Mundo**

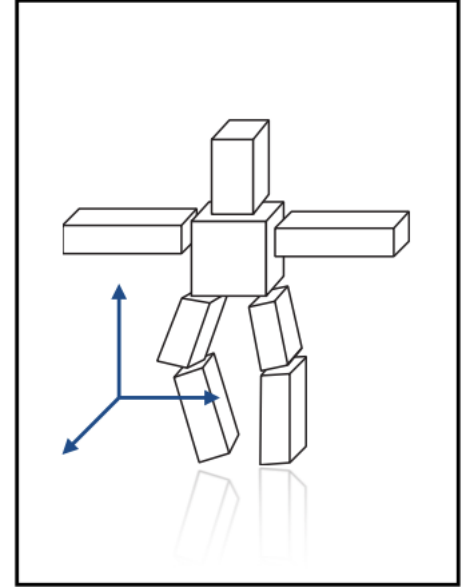


# Recapitulando Transformações



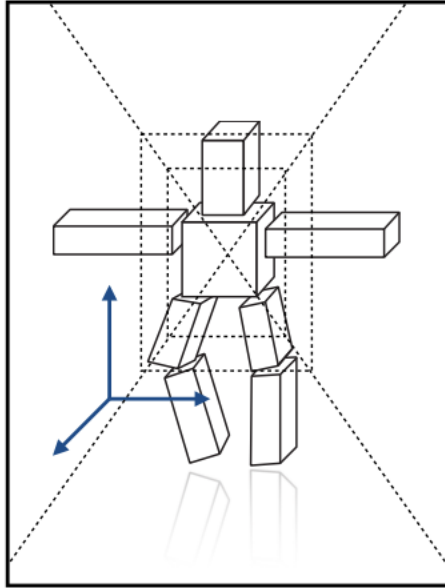
**Coordenadas do Mundo**

→  
**Viewing  
transform**



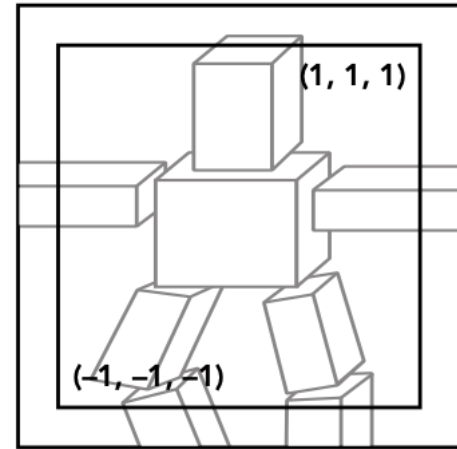
**Coordenadas da Câmera**

# Recapitulando Transformações



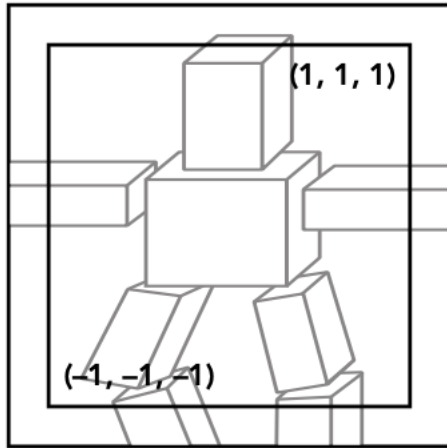
**Coordenadas da Câmera**

→  
**Perspective  
projection  
and  
homogeneous  
divide**



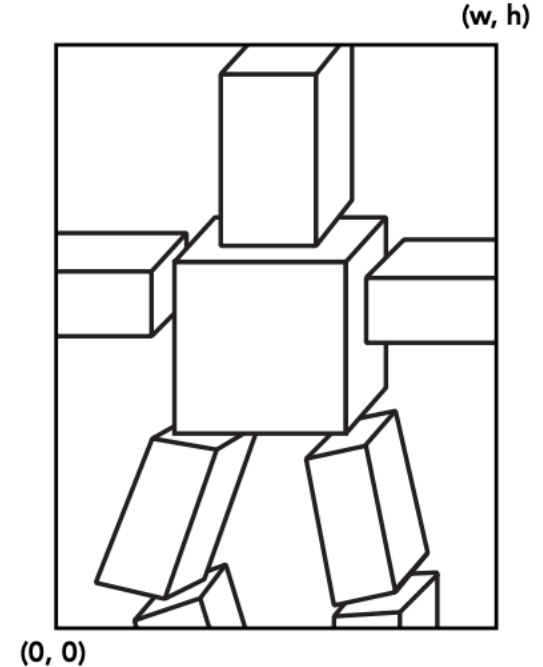
**Coordenadas Normalizadas  
de Câmera "Device" (NDC)**

# Recapitulando Transformações



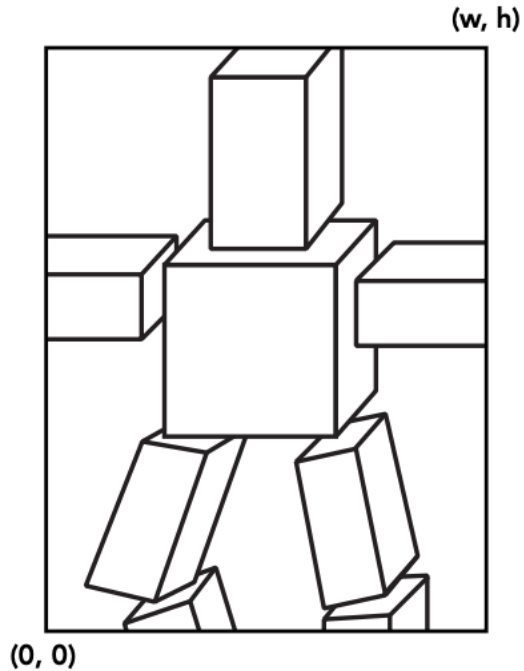
**Coordenadas Normalizadas  
de Câmera "Device" (NDC)**

→  
**Screen  
transform**

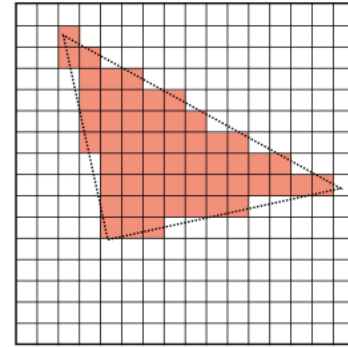


**Coordenadas da Tela**

# Recapitulando Transformações



→  
**Rasterização**



**Imagem**

**Coordenadas da Tela**



# X3D Continuação

- Transform (mas somente um por vez)
- Viewpoint
- Coordinate
- TriangleSet
- IndexedTriangleStripSet
- TriangleStripSet
- Box

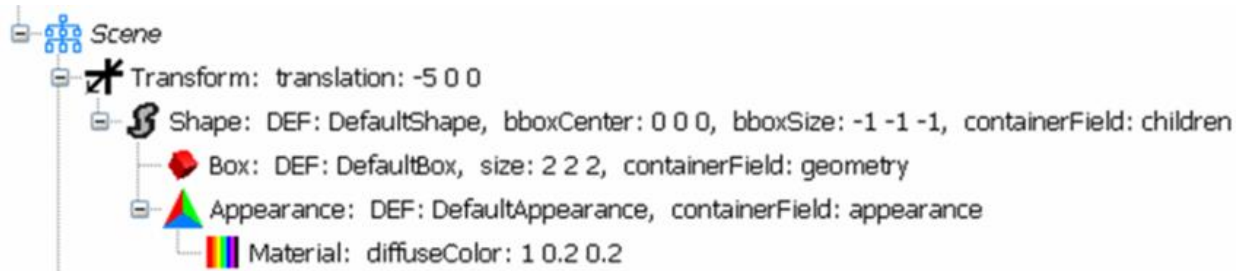


# Transform

Transform : X3DGroupingNode {

MFNode	[in]	addChildren		[X3DChildNode]
MFNode	[in]	removeChildren		[X3DChildNode]
SFVec3f	[in,out]	center	0 0 0	(-∞,∞)
<b>MFNode</b>	<b>[in,out]</b>	<b>children</b>	<b>[]</b>	<b>[X3DChildNode]</b>
SFNode	[in,out]	metadata	NULL	[X3DMetadataObject]
<b>SFRotation</b>	<b>[in,out]</b>	<b>rotation</b>	<b>0 0 1 0</b>	<b>[-1,1] or (-∞,∞)</b>
<b>SFVec3f</b>	<b>[in,out]</b>	<b>scale</b>	<b>1 1 1</b>	<b>(0,∞)</b>
SFRotation	[in,out]	scaleOrientation	0 0 1 0	[-1,1] or (-∞,∞)
<b>SFVec3f</b>	<b>[in,out]</b>	<b>translation</b>	<b>0 0 0</b>	<b>(-∞,∞)</b>
SFVec3f	[]	bboxCenter	0 0 0	(-∞,∞)
SFVec3f	[]	bboxSize	-1 -1 -1	[0,∞) or -1 -1 -1

}



# Transform

Ordem de execução das operações de Transformação quando mais de uma definida no mesmo nó Transform:

1. Escala
2. Rotação
3. Translação

```
<Transform scale="2 2 2" rotation="0 1 0 1.5708" translation="1 1 0">
```

=

```
<Transform translation="1 1 0">  
  <Transform rotation="0 1 0 1.5708">  
    <Transform scale="2 2 2">
```

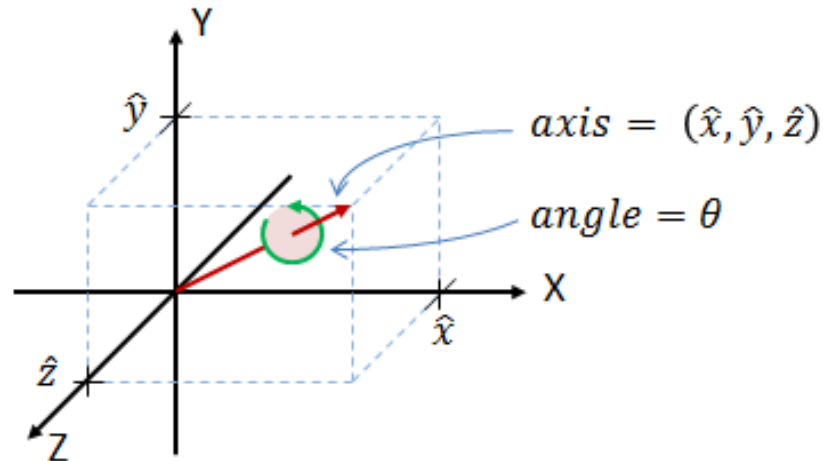
**Obs: Não necessário implementar hierarquia de Transforms para essa fase do projeto**

# Em especial: SFRotation

O **SFRotation** é um vetor usado para definir rotações dos objetos.

Os três primeiros valores especificam um vetor de eixo normalizado sobre o qual a rotação ocorre, de modo que os três primeiros valores devem estar dentro do intervalo  $[-1 \dots +1]$  para representar um vetor de unidade normalizado.

O quarto valor especifica a quantidade de rotação pela regra da mão direita em torno desse eixo em radianos.



# Viewpoint

O nó **Viewpoint** define um ponto de vista que fornece uma câmera virtual em perspectiva da cena. O valor de **fieldOfView** representa o ângulo de visão mínimo em radianos em qualquer eixo de direção perpendicular à visão deste ponto de vista. O campo de visão deve ser maior que zero e menor que  $\pi$ .

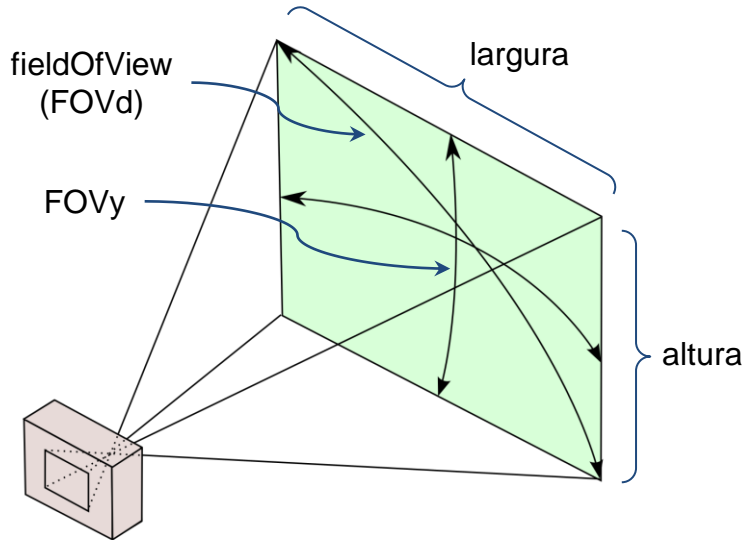


```
Viewpoint : X3DViewpointNode {  
  SFBool      [in]      set_bind  
  SFVec3f     [in,out]  centerOfRotation    0 0 0      (-∞,∞)  
  SFString    [in,out]  description          ""  
  SFFloat    [in,out]  fieldOfView          $\pi/4$       (0, $\pi$ )  
  SFBool      [in,out]  jump                 TRUE  
  SFNode      [in,out]  metadata             NULL      [X3DMetadataObject]  
  SFRotation [in,out]  orientation         0 0 1 0    [-1,1],(-∞,∞)  
  SFVec3f    [in,out]  position            0 0 10    (-∞,∞)  
  SFBool      [in,out]  retainUserOffsets    FALSE  
  SFTIME      [out]     bindTime  
  SFTIME      [out]     isBound  
}
```

# FOVy (opcional para projeto)

No X3D o fieldOfView é determinado pelo ângulo de visão mínimo de qualquer eixo de direção perpendicular à visão.

$$\frac{\text{display width}}{\text{display height}} = \frac{\tan(\text{FOV}_{horizontal}/2)}{\tan(\text{FOV}_{vertical}/2)}$$



$$\text{FOVy} = 2 \cdot \arctan \left( \tan \left( \frac{\text{FOVd}}{2} \right) \cdot \frac{\text{Altura}}{\sqrt{\text{Altura}^2 + \text{Largura}^2}} \right)$$



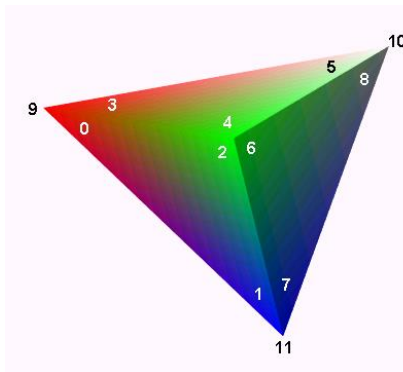
# Coordinate

O nó **Coordinate** define um conjunto de coordenadas 3D a serem usadas no campo de coordenadas de nós de geometria baseada em vértices.

```
Coordinate : X3DCoordinateNode {  
  SFNode          [in,out]  metadata NULL    [X3DMetadataObject]  
  MFVec3f         [in,out]  point   []       (-∞,∞)  
}
```

# TriangleSet

O nó **TriangleSet** representa uma geometria 3D que representa uma coleção de triângulos individuais. Cada triângulo é formado por um conjunto consecutivo de três vértices do nó **Coordinate**. Se o nó de coordenadas não contém um múltiplo de três valores de coordenadas, os vértices restantes devem ser ignorados..



```
TriangleSet : 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]  
  SFNode [in,out] texCoord NULL [X3DTextureCoordinateNode]  
  SFBool [] ccw TRUE  
  SFBool [] colorPerVertex TRUE  
  SFBool [] normalPerVertex TRUE  
  SFBool [] solid TRUE  
}
```

# 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='0 1 2 3 4 5 6 7 8 9 10 11 12 -1'>

<Coordinate point='-4.0 -1.0 -.5 -4.5 -2.0 -.5 -3.0 -.5 0 -2.5 -1.5 -.5 -2.0 -.5 -1 -1.5 -1.5 -.5 -.5 .5 -.5 0 0 0 1 .5 -.5 1.5 -2.0 -1 2.5 -2.0 -.5 2.5 -2.5 -.5 3.5 -2.0 -1.0'>

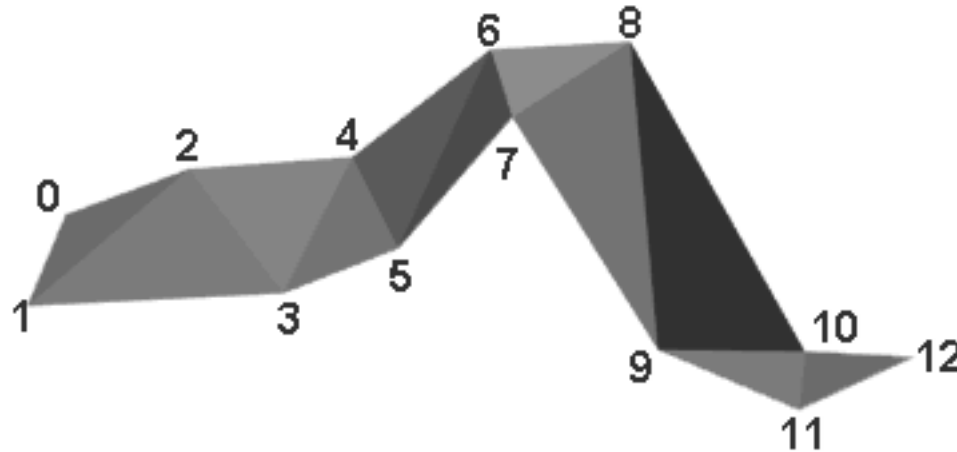
</IndexedTriangleStripSet >

<Appearance>

<Material diffuseColor='0.5 0.5 0.5'>

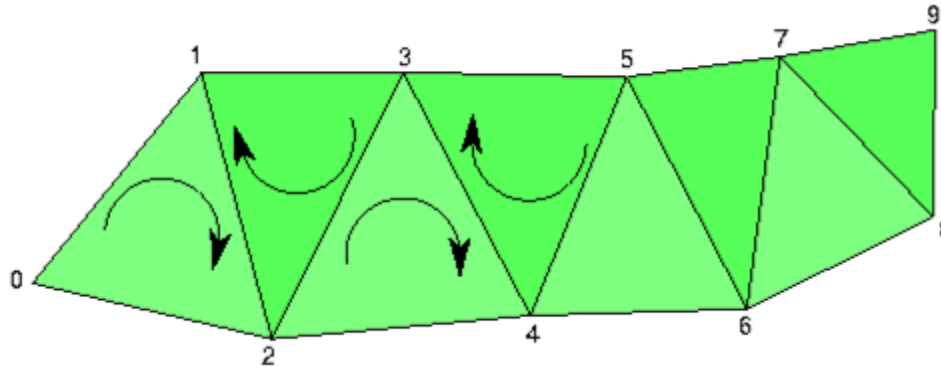
</Appearance>

</Shape>



# Cuidado

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



Fonte: <https://www.opengl.org/archives/resources/code/samples/advanced/advanced98/notes/node16.html>

# 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 -.5 -4.5 -2.0 -.5 -3.0 -.5 0 -2.5 -1.5 -.5 -2.0 -.5 -1 -1.5 -1.5 -.5 -.5 .5 -.5 0 0 0 1 .5 -.5 1.5 -2.0 -1 2.5 -2.0 -.5 2.5 -2.5 -.5 3.5 -2.0 -1.0'/'>

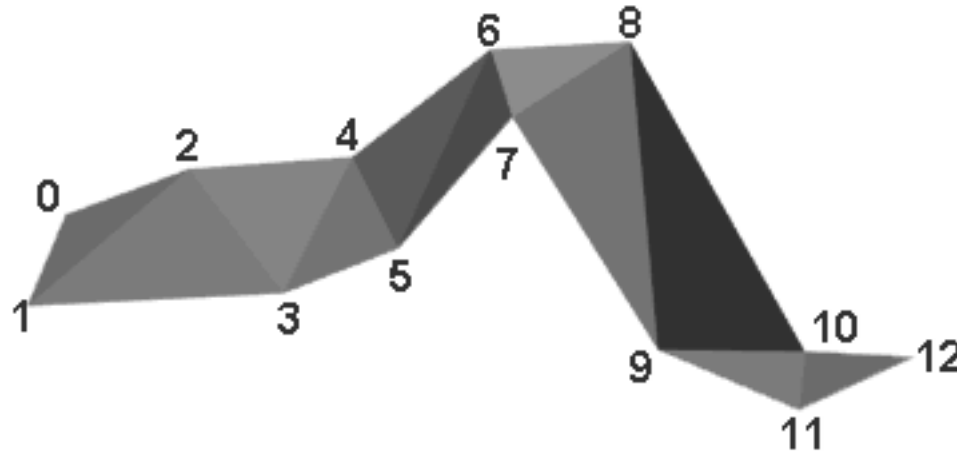
</TriangleStripSet>

<Appearance>

<Material diffuseColor='0.5 0.5 0.5'/'>

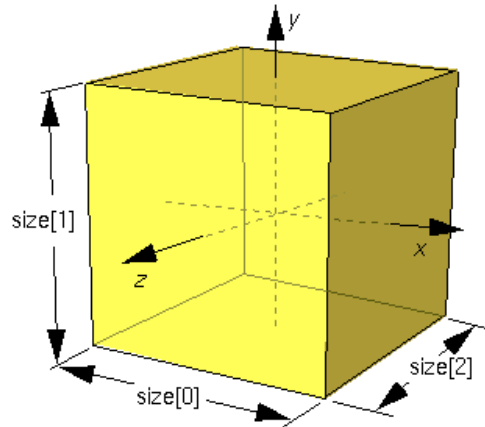
</Appearance>

</Shape>



# Box

O nó **Box** especifica uma caixa 3D paralelepípeda retangular centrada no (0, 0, 0) no sistema de coordenadas local e alinhado com os eixos de coordenadas locais. Por padrão, a caixa mede 2 unidades em cada dimensão, de -1 a +1. O campo **size** especifica as extensões da caixa ao longo dos eixos X, Y e Z, respectivamente, e cada valor do tamanho deve ser maior que zero.



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

# Projeto 1 : Segunda Parte

## **Continuar renderizador:**

**Tarefa 2:** fazer as transformadas no modelo (2,5 pontos)

**Tarefa 3:** fazer as transformadas de câmera (2,5 pontos)

**Tarefa 4:** fazer as transformadas de projeção perspectiva (2,5 pontos)

**Tarefa 5:** fazer as transformações para coordenadas da tela (2,5 ponto)

## **Base:**

<https://github.com/lpsoares/Renderizador> (atualizar)

## **Data de Entrega:**

8/3/2021 às 23:59 [Supondo mesmo repositório da fase 1.]

## **Detalhes:**

Página da Disciplina (<https://lpsoares.github.io/ComputacaoGrafica/>)

# Computação Gráfica

Luciano Soares

<lpsoares@insper.edu.br>

Fabio Orfali

<fabioo1@insper.edu.br>