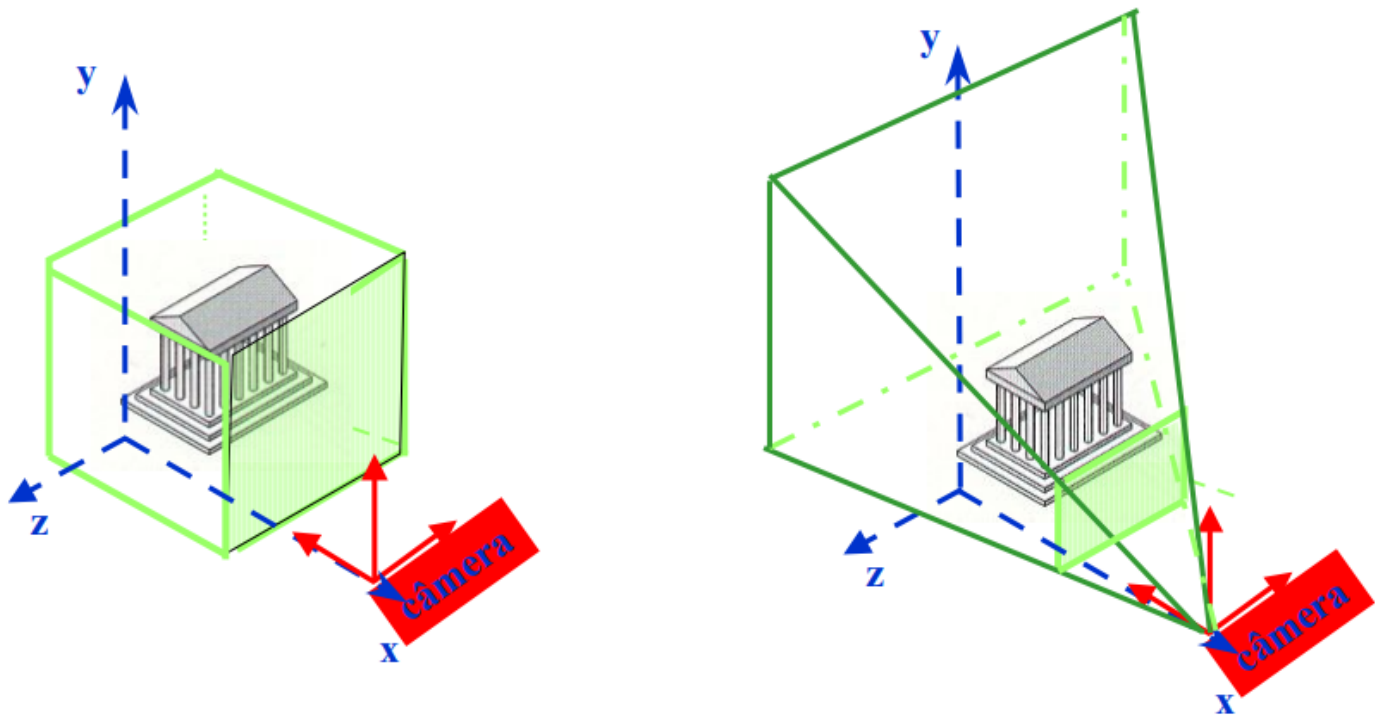


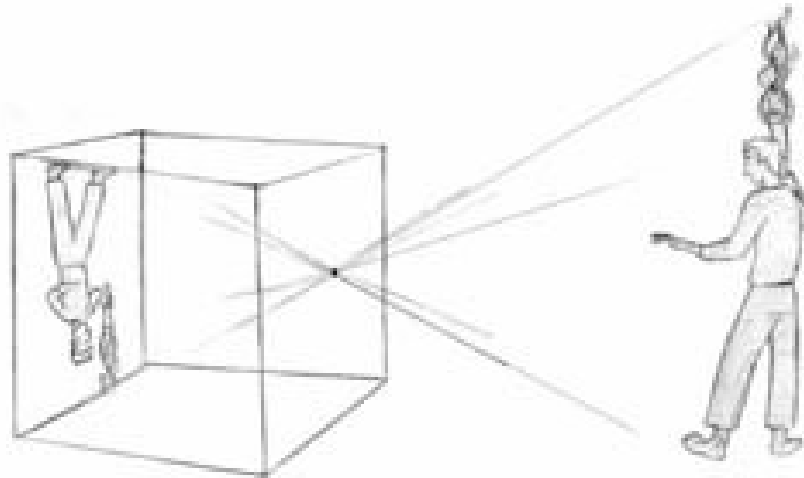
# Visualização 3D

- Se comparado com o processo de visualização 2D a visualização de objetos 3D é mais complexa porque os dispositivos gráficos existentes são adequados à apresentação de imagens planas (2D).
- Solução: Analogia com **câmera** pinhole



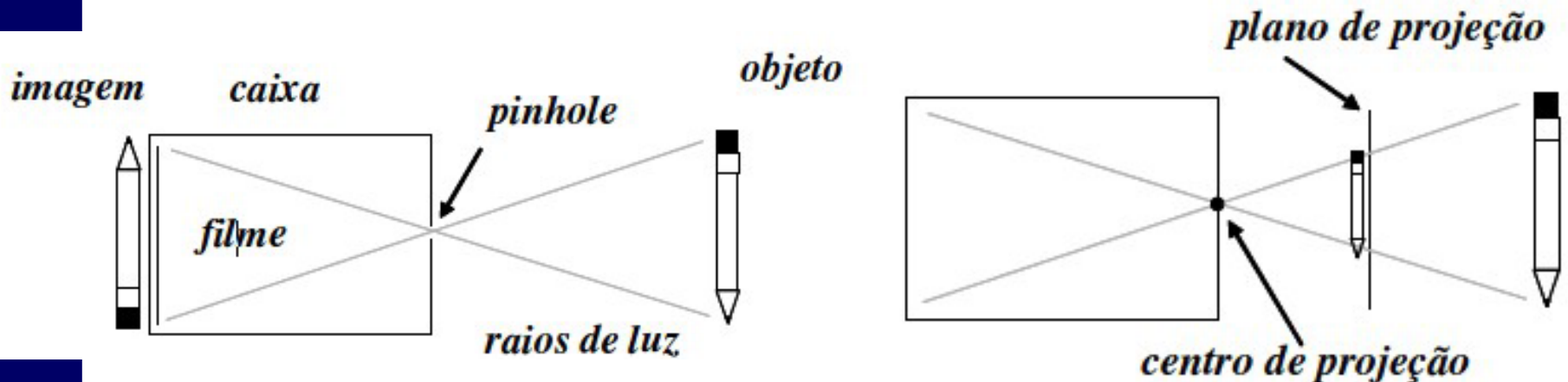
# Visualização 3D

- Os raios de luz viajam em linha reta.
- Assim, a cor de um ponto na imagem projetada (**projeção**) no plano corresponde ao ponto de interseção da reta que passa por este ponto e pelo orifício da **câmera** com o objeto iluminado.



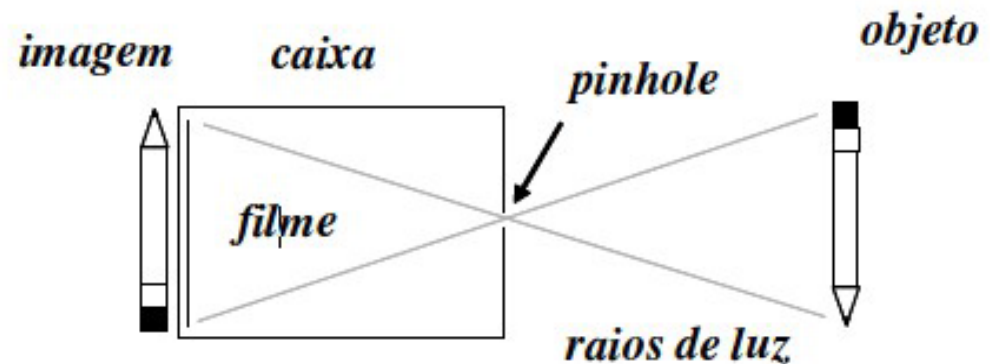
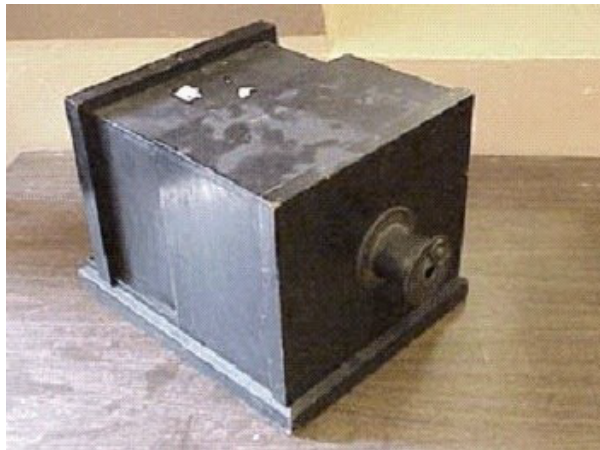
# Visualização 3D

- A imagem projetada é invertida na câmera pinhole.
- Matematicamente podemos obter a mesma imagem sem a inversão, se o plano de projeção estiver entre o **centro de projeção** (orifício) e o objeto.



# Câmera Virtual

- Uma câmera virtual deve representar a essência da geometria de uma câmera pinhole



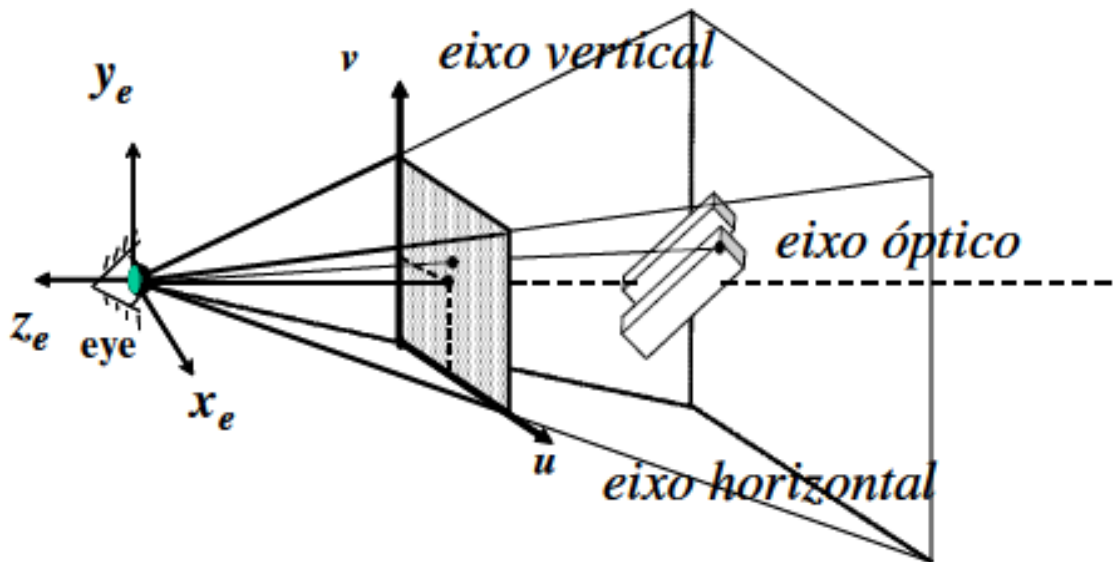
# Câmera Virtual

---

- Uma câmera virtual terá um **centro de projeção**, um **eixo óptico** e um retângulo (**plano de projeção**) onde se forma uma imagem.
- O eixo óptico é perpendicular a este retângulo e o intercepta no seu centro
- O tamanho do retângulo e a sua distância ao centro de projeção definem a abertura da câmera ou fov (field of vision).

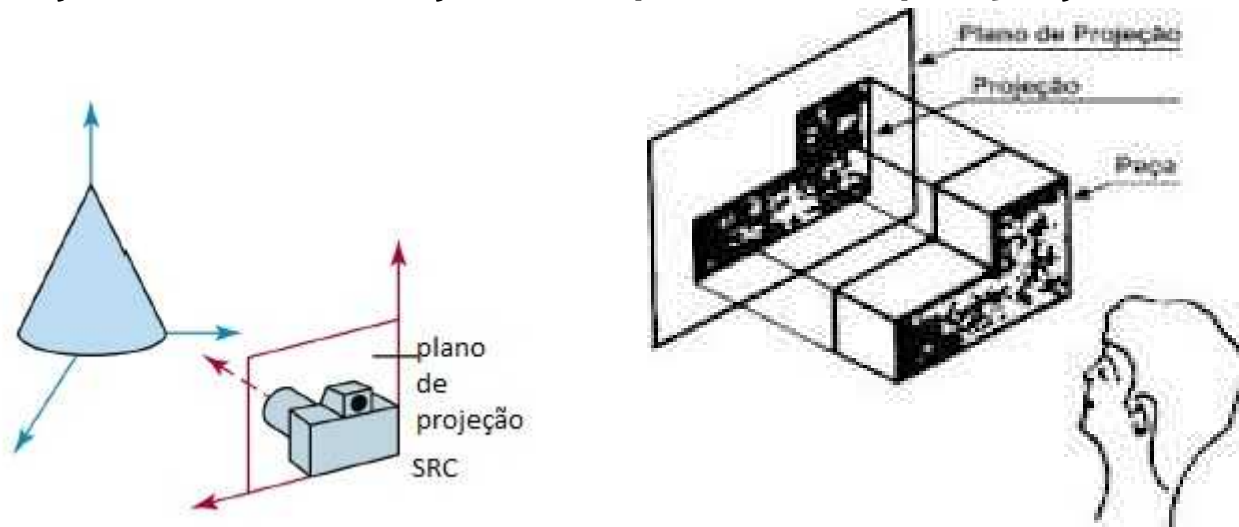
# Câmera Virtual

- Os parâmetros de uma câmera são: a posição do centro de projeção (**eye**), um ponto para onde a câmera esteja apontando (**center**) e um vetor que indique a direção “para cima” da câmera (**up**).



# Visualização de Objetos 3D

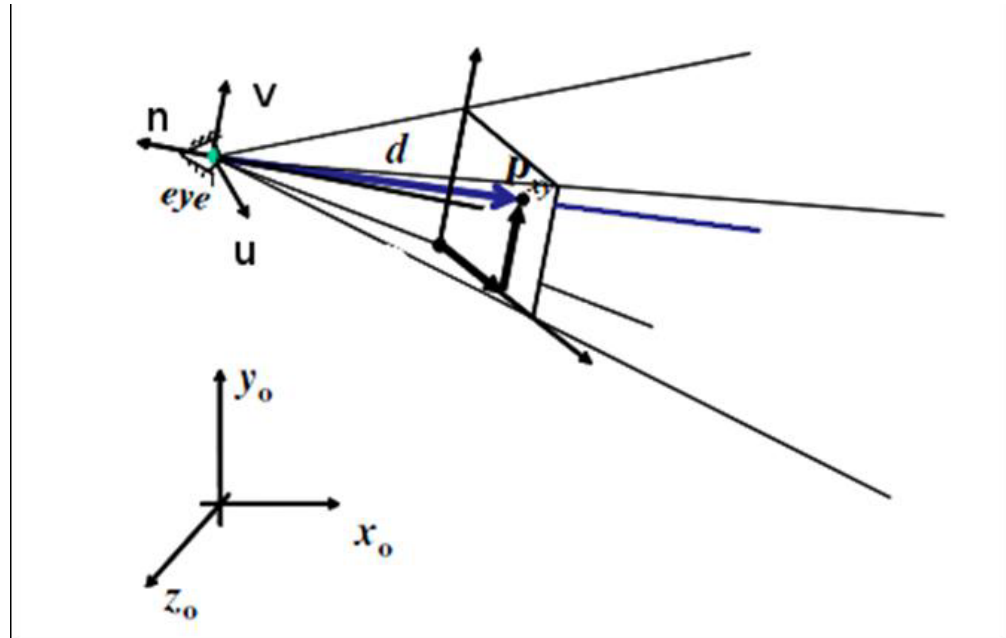
- Objeto gráfico descrito no SRU (mão direita ou mão esquerda)
- Sistema de referência da câmera
  - Posição e orientação do plano de projeção



Antes de projetar o objeto 3D é preciso obter suas coordenadas em relação ao SRC e então projetá-las.

# Câmera Virtual

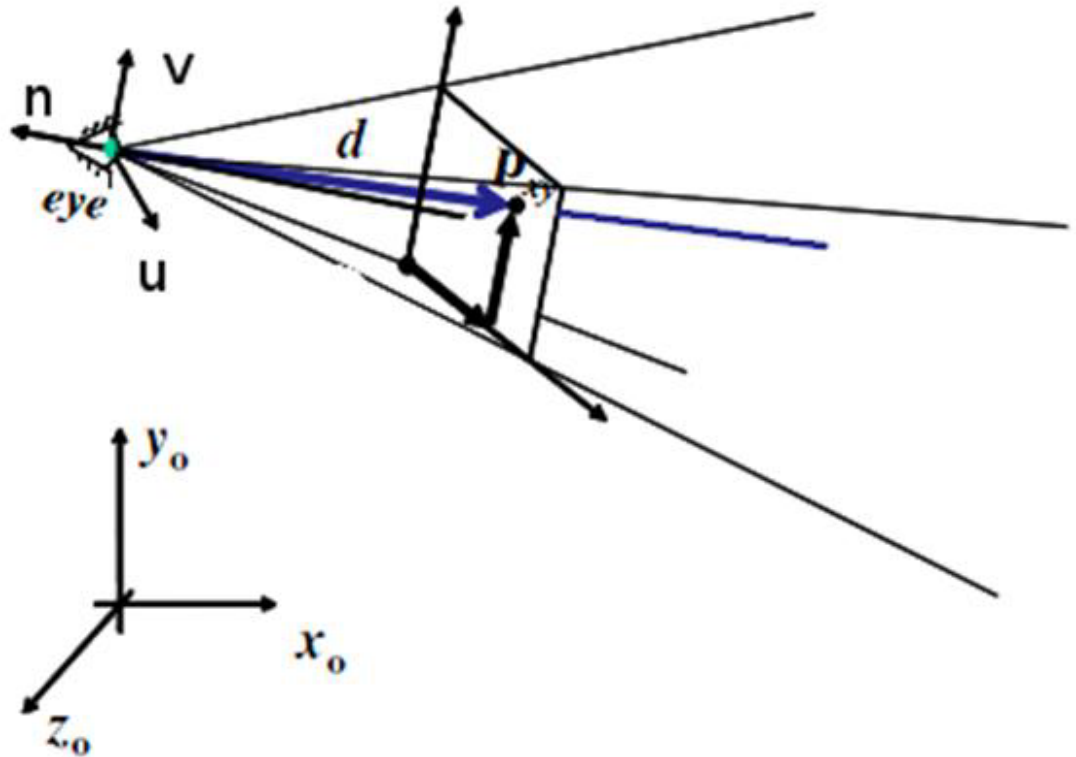
- Sistema de coordenadas da câmera:  
Coordenadas do observador (eye), ponto de visada (look) e vetor up.
- $N = \text{eye} - \text{look}$
- $U = \text{up} \times N$
- $V = N \times U$





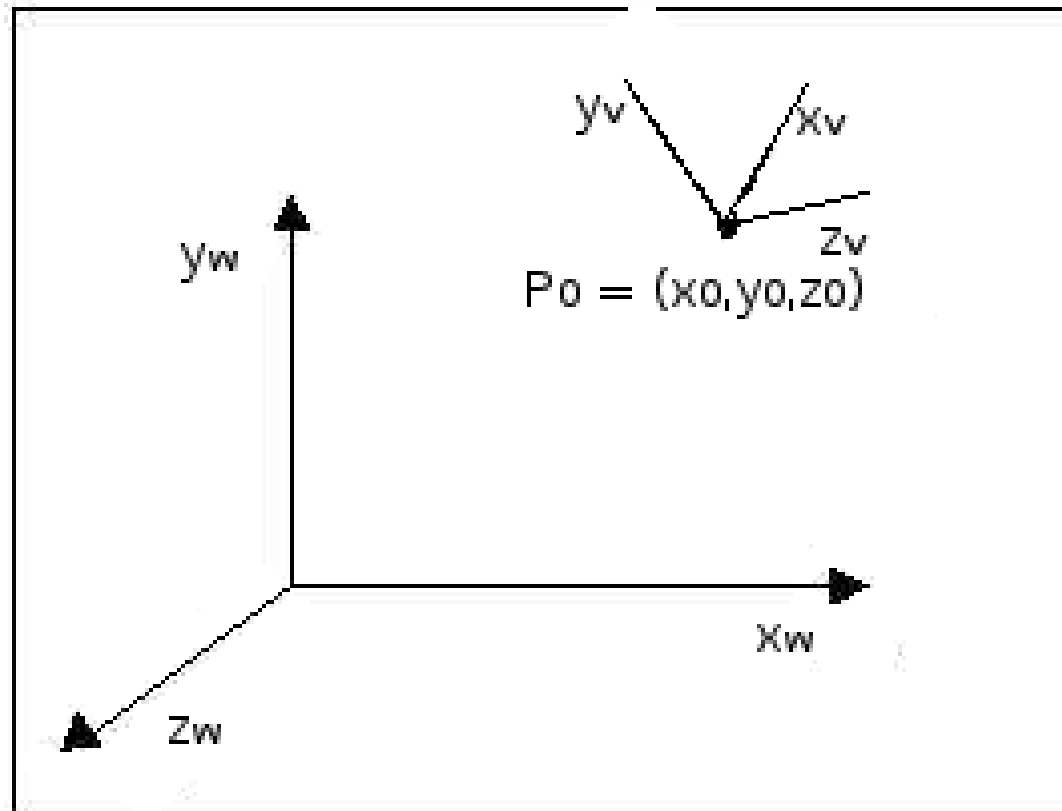
# Câmera Virtual

- Para um sistema de coordenadas é conveniente vetores normalizados.
- $\mathbf{n} = \mathbf{N}/|\mathbf{N}|$
- $\mathbf{u} = \mathbf{U}/|\mathbf{U}|$
- $\mathbf{v} = \mathbf{V}/|\mathbf{V}|$



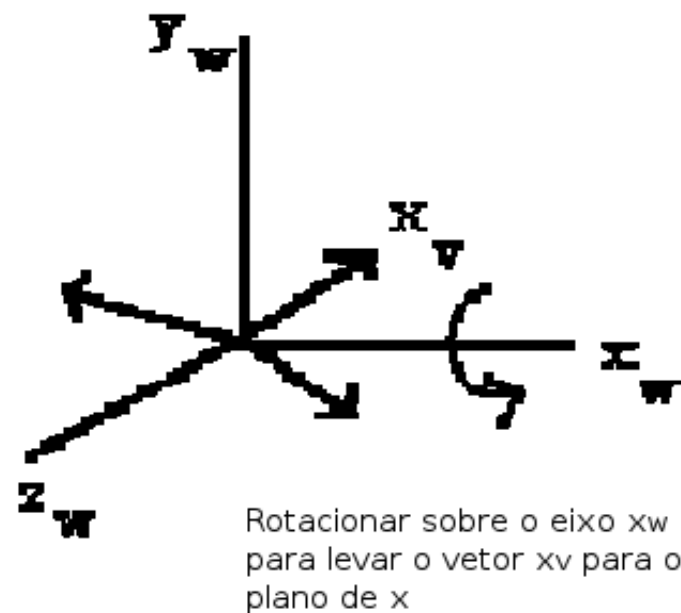
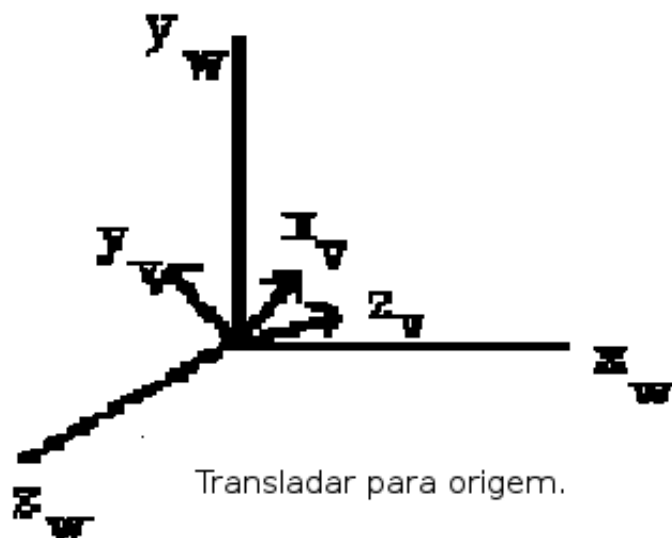
# Câmera Virtual

- Alinhando o sistema de coordenadas de visualização com o sistema de coordenadas do mundo



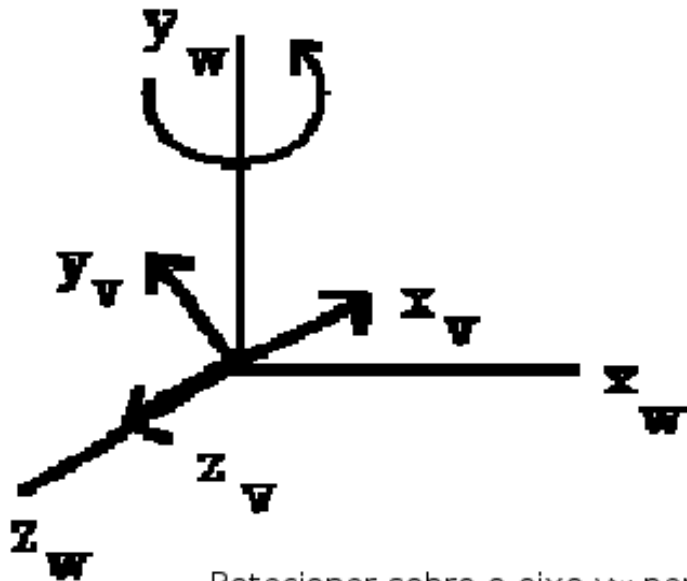
# Câmera Virtual 1/2

- Transladar para Origem do World
- Rotacionar em torno do  $X_w$

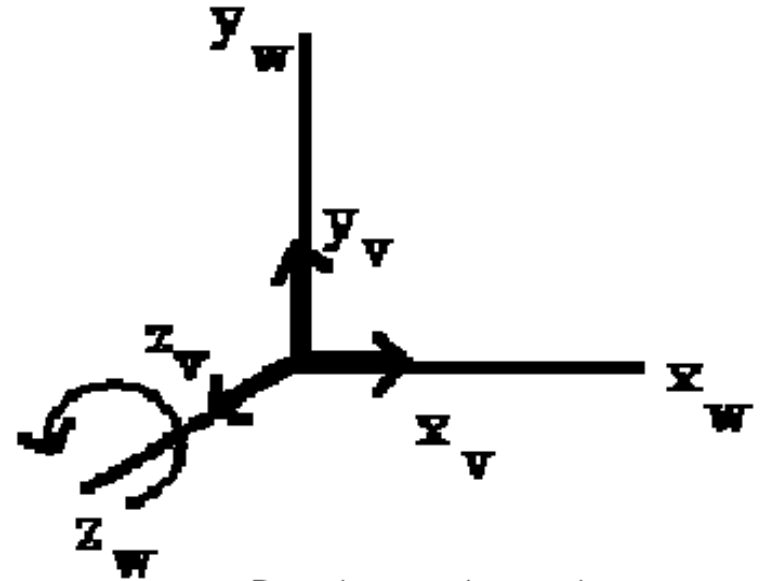


# Câmera Virtual – 2/2

- Rotacionar em torno do  $Y_w$  e  $Z_w$



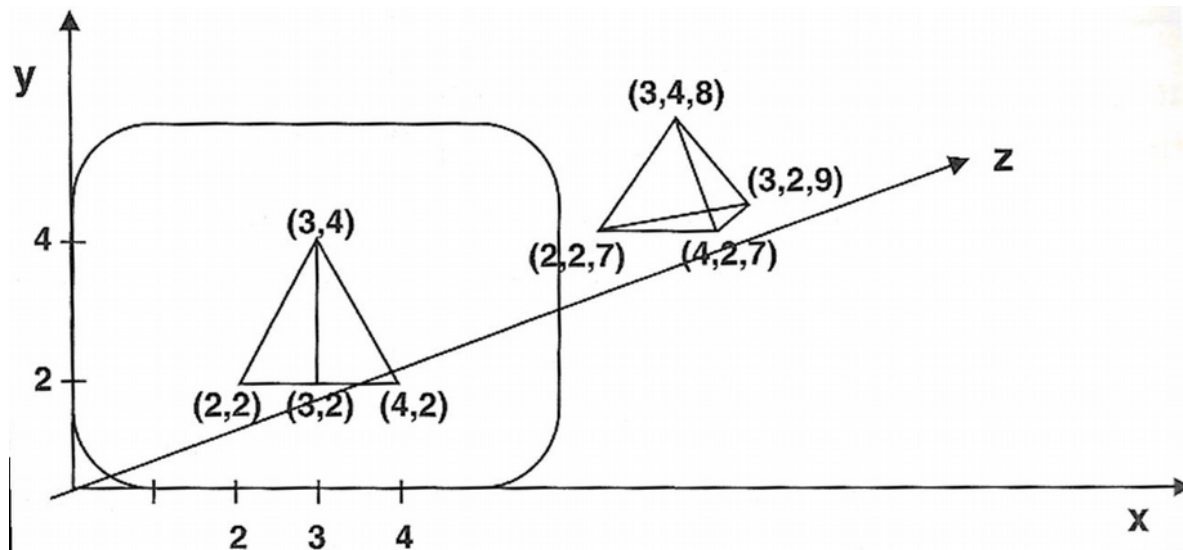
Rotacionar sobre o eixo  $y_w$  para  
levar o eixo  $y_v$  para o plano de  $y_w$



Rotacionar sobre o eixo  $z_w$  para  
levar o eixo  $z_v$  para o plano de  $z_w$

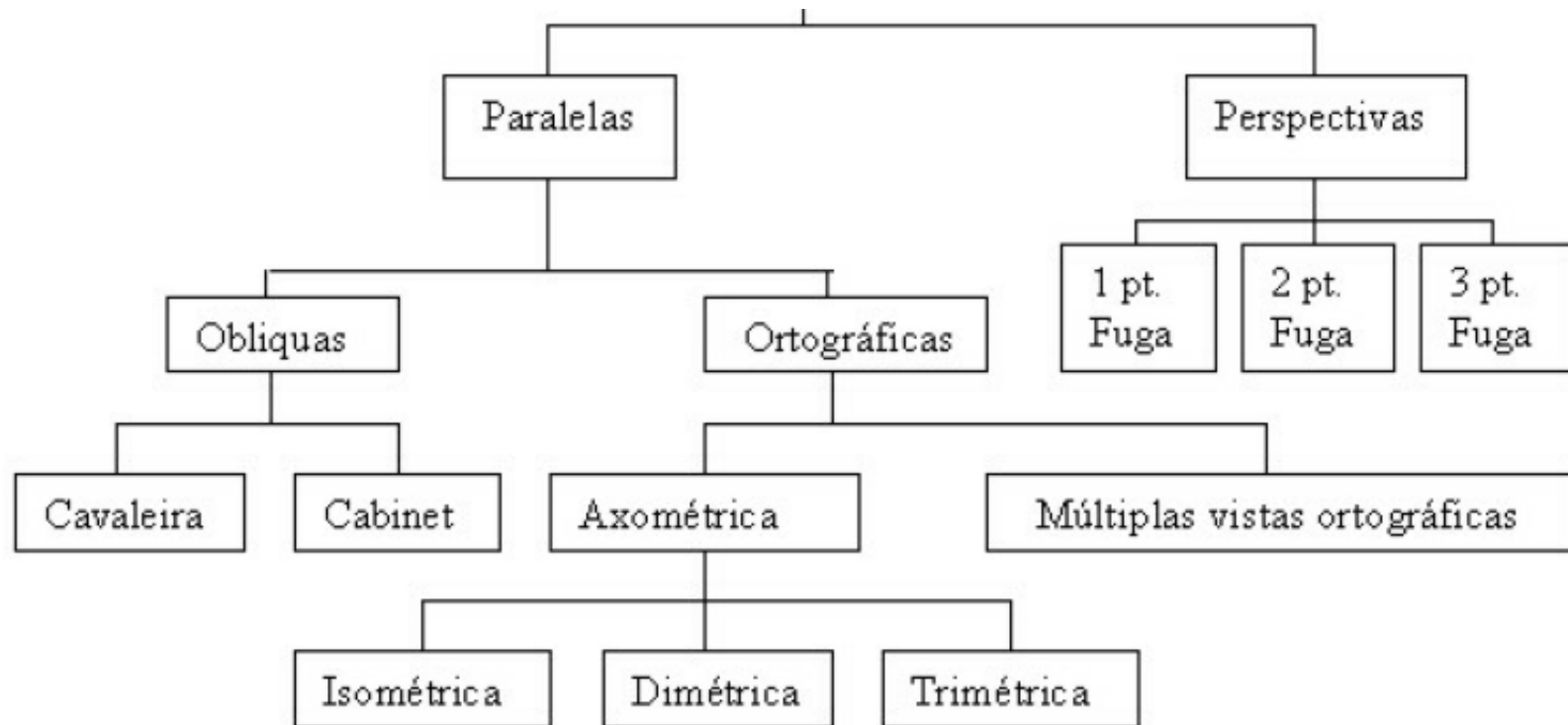
# Projeção

- **Projeções** geométricas permitem a visualização bidimensional de objetos tridimensionais
- Para gerar uma imagem de um objeto 3D, precisamos converter as coordenadas 3D em coordenadas 2D, que correspondam a uma visão do objeto de uma posição



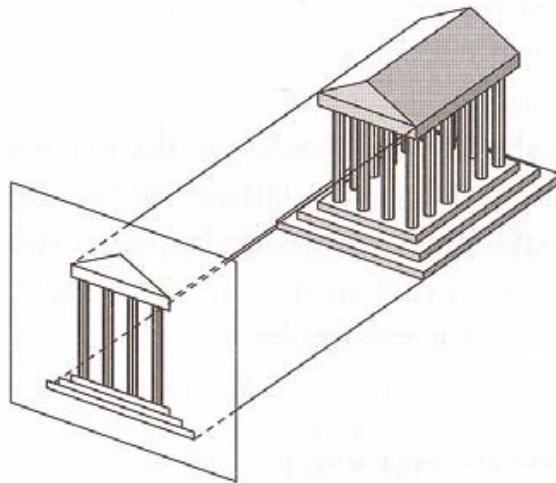
# Projeção

- As classificações das projeções dependem das relações entre o centro de projeção, o plano de projeção e as linhas ou raios de projeção.



# Projeção Paralela

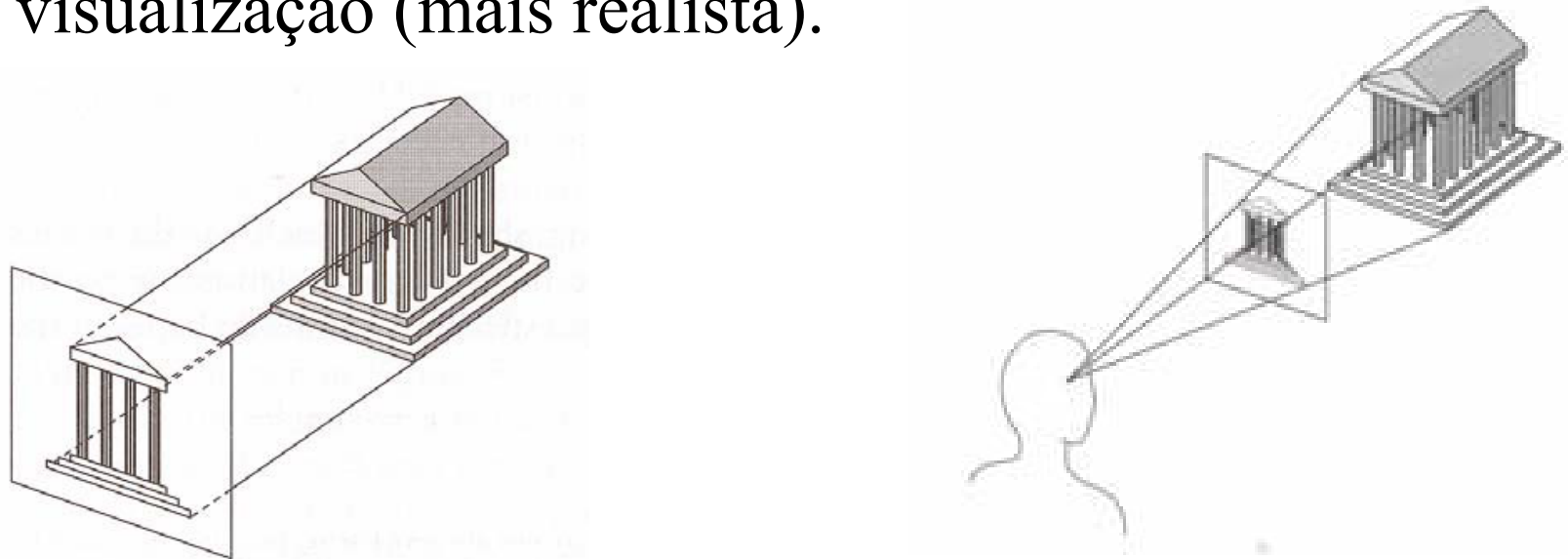
- Um método para visualizar um objeto em um plano de exibição é projetar pontos na superfície do objeto ao longo de linhas paralelas.
- Esta técnica usada em engenharia e arquitetura representa um objeto com ponto de vista que mostra as dimensões exatas do objeto.



Centro de projeção no infinito

# Projeção Perspectiva

- Um método para visualizar um objeto em um plano de exibição é projetar ao longo de caminhos convergentes
- Este processo faz com que objetos mais longe da posição de visualização sejam exibidos mais pequenos do que objetos que estão mais perto da posição de visualização (mais realista).





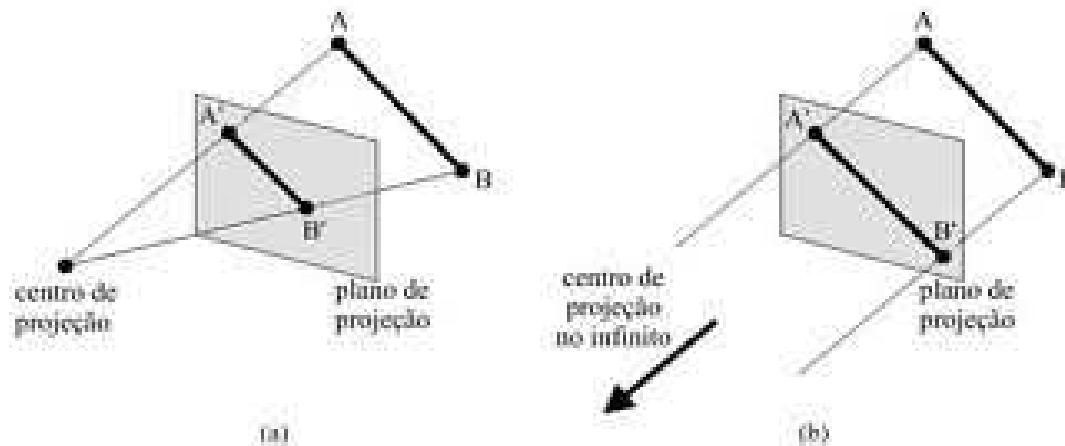
# Projeção

- **Projeção Paralela**

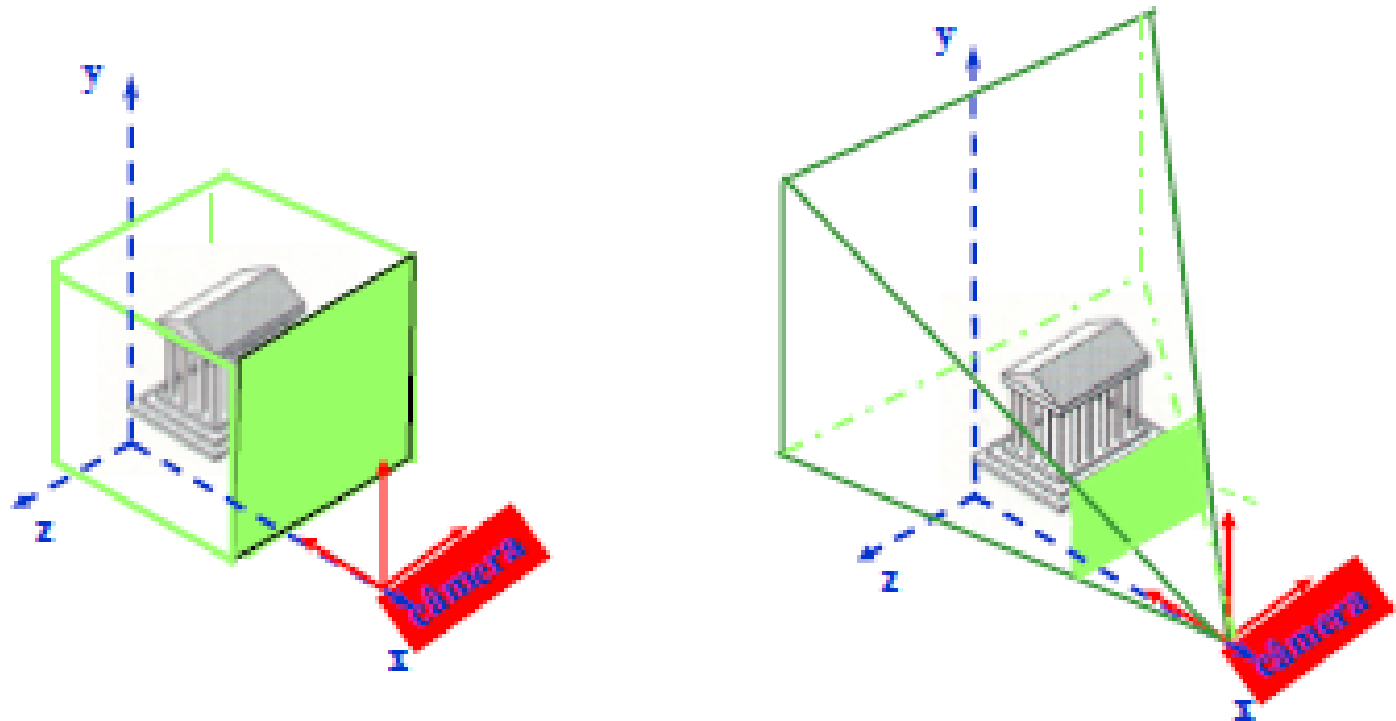
- projeta os pontos de um objeto ao longo de linhas paralelas
- Usado para desenhos arquitetônicos e de engenharia

- **Projeção Perspectiva**

- projeta os pontos de um objeto ao longo de caminhos convergentes
- Gera cenas mais realísticas (objetos longe da posição de visão são mostrados menores)



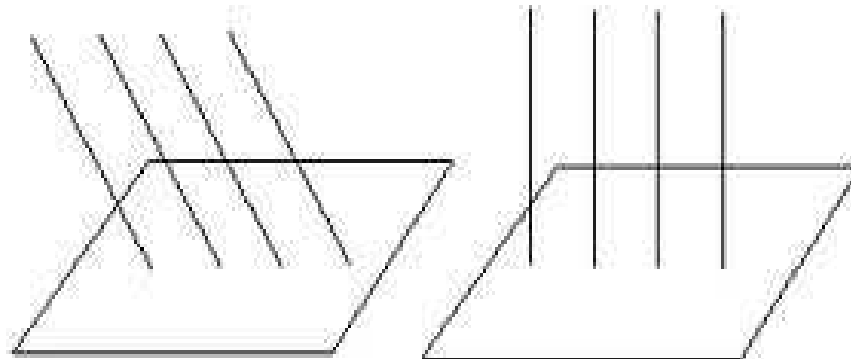
# Projeção



- Projeção ortográfica x projeção perspectiva

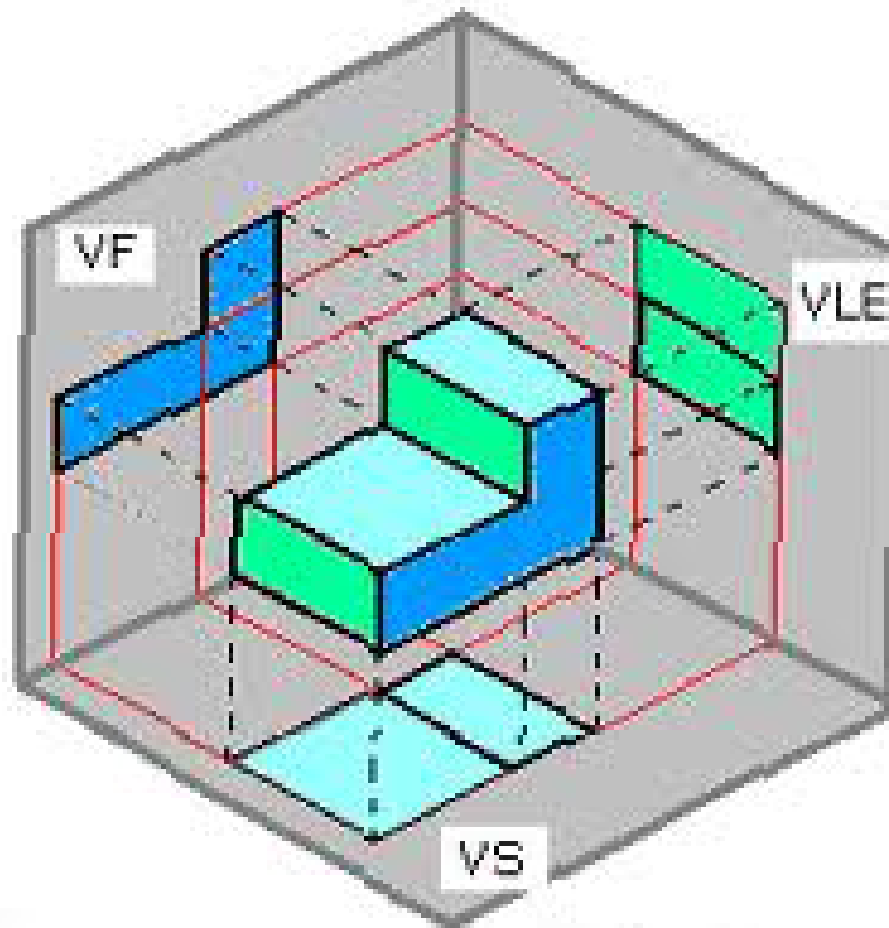
# Projeção Paralela

- A projeção paralela preserva retas paralelas; ângulos são preservados apenas em plano paralelos ao plano de projeção
- Projeção Ortográfica
  - Projetantes perpendiculares ao plano de projeção
- Projeção Oblíqua
  - Projetantes não perpendiculares ao plano de projeção



# Projeção Paralela Ortográfica - Vistas

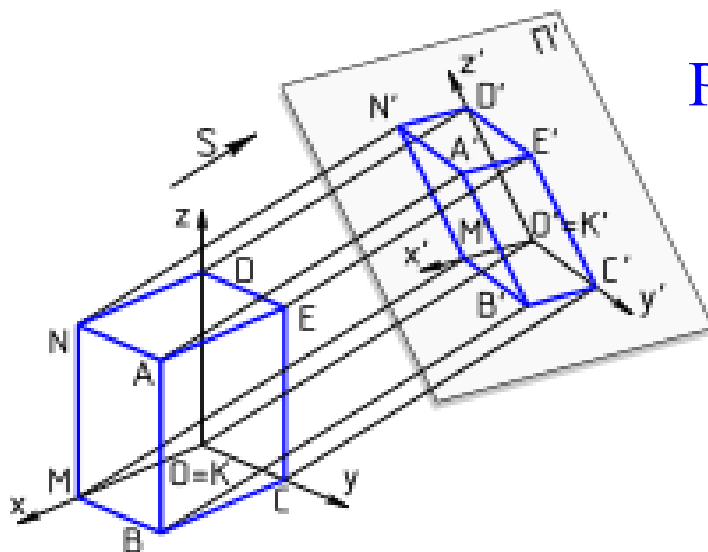
## Projeção Paralela ortográfica vistas



Plano de projeção paralelo a um dos planos principais

# Projeção Paralela Ortográfica - Axonométrica

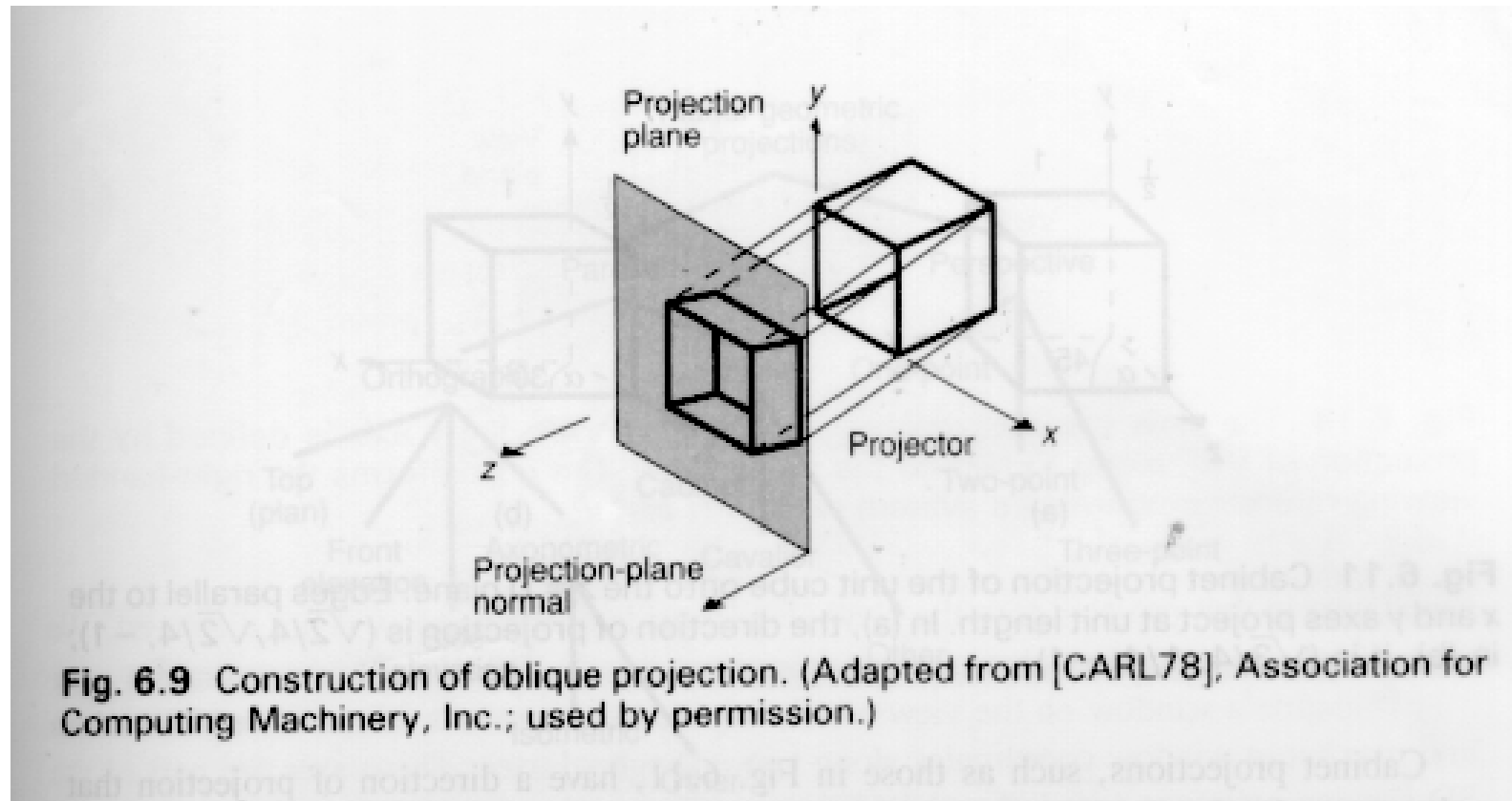
- Projeção ortogonal que mostra mais de uma face do objeto
- Retas paralelas são projetadas em retas paralelas, mas os ângulos não são preservados.
- Projeção axonométrica isométrica:
  - O plano de projeção é alinhado para intersectar cada eixo à mesma distância da origem ( $120^\circ$ )



Projeção axonométrica

# Projeção Oblíqua

- Cavalier (45 graus)
- Cabinet (63.4 graus)

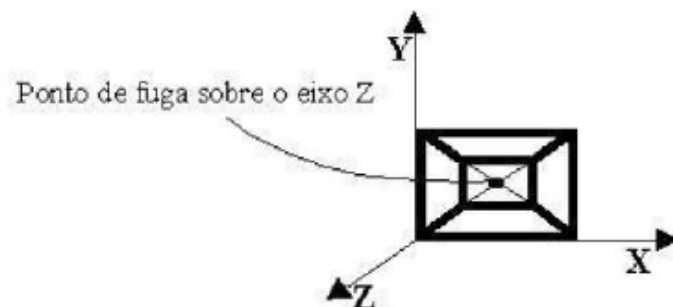
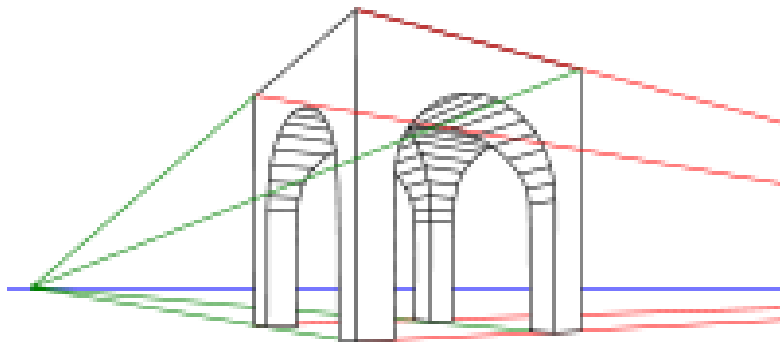


# Projeção Perspectiva

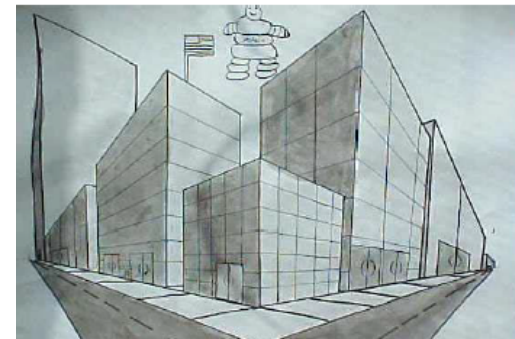
- Centro de projeção é um ponto do espaço
- Simula a projeção do olho humano
- O objeto é deformado de forma inversamente proporcional à distância ao centro de projeção
- Não permitem medidas diretas
- Objetos mais distantes parecem menores
- Retas paralelas se encontram em um ponto (ponto de fuga)
- Pode ter 1, 2 ou 3 pontos de fuga

# Projeção Perspectiva

- Os desenhos em perspectiva são caracterizados pela mudança do comprimento e pelos pontos de fuga
- Pontos de fuga: No de eixos que o plano de projeção corta.
- Se a projeção é com 1 ponto de fuga então o plano de projeção corta  $z$  e as linhas paralelas a  $x$  e  $y$  não convergem



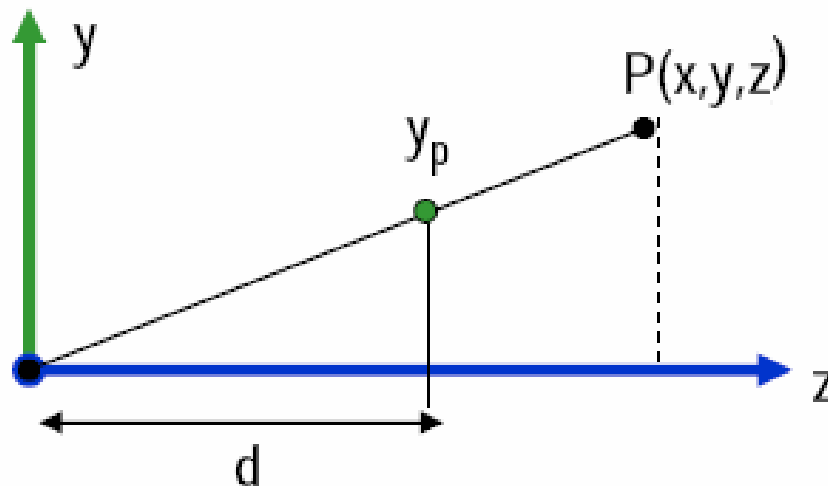
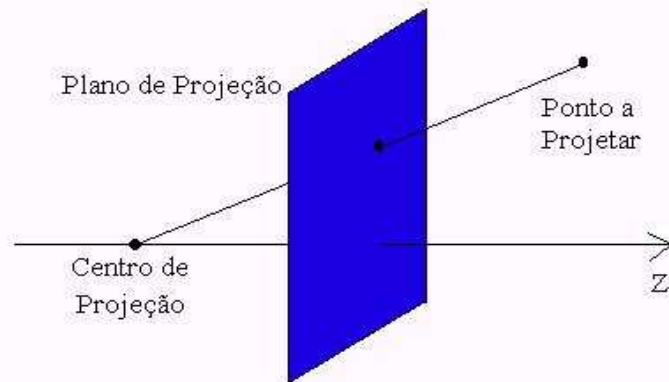
1 ponto  
de fuga



2 pontos  
de fuga



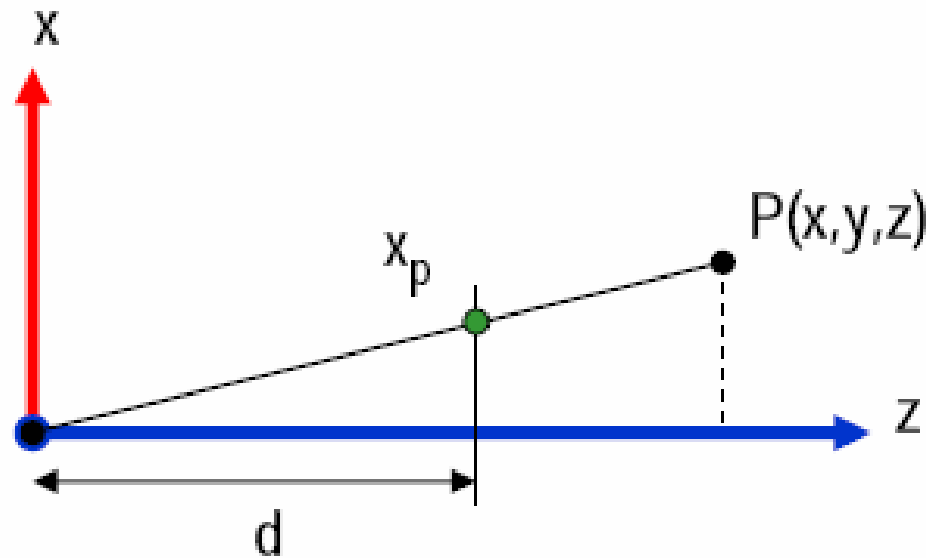
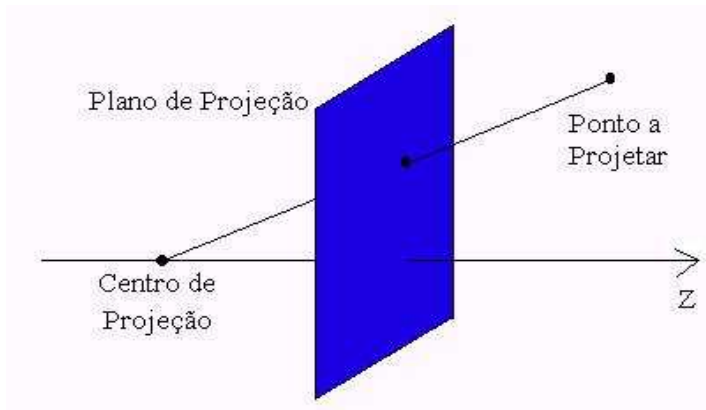
# Projeção Perspectiva (y')



$$\frac{y_p}{d} = \frac{y}{z}$$

$$y_p = \frac{yd}{z} = \frac{y}{z/d}$$

# Projeção Perspectiva (x')



$$\frac{x_p}{d} = \frac{x}{z}$$

$$x_p = \frac{xd}{z} = \frac{x}{z/d}$$

- Após efetuar a mudança de coordenadas, do mundo real para as coordenadas da câmera, a próxima etapa é efetuar a projeção desejada, ou seja, aplicar uma matriz que transforma os pontos .

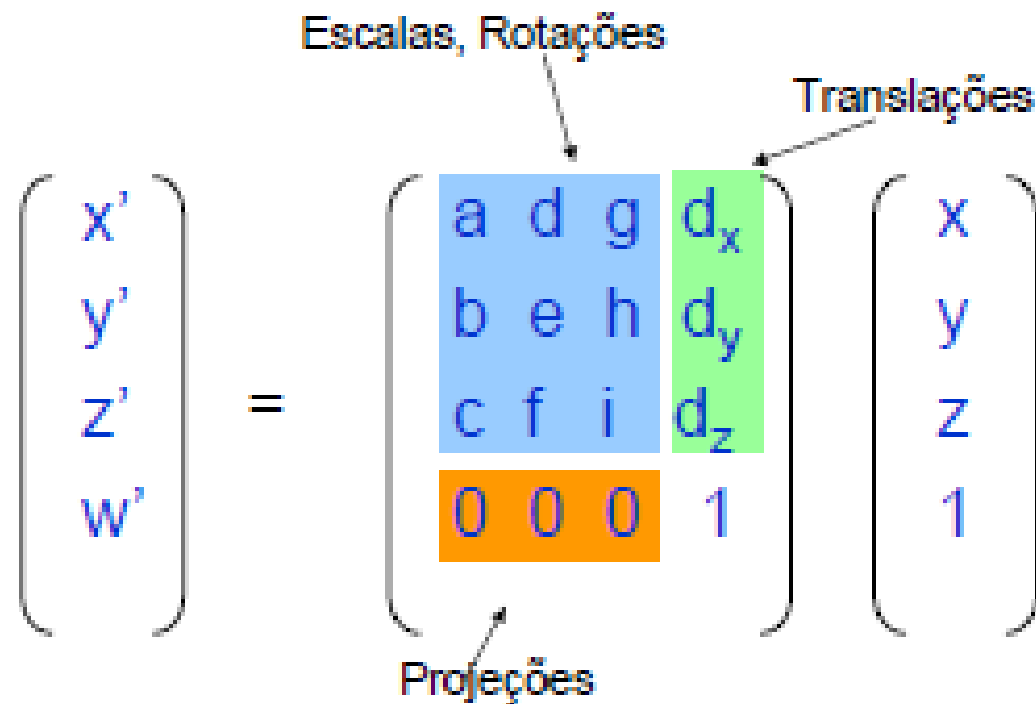
# Projeção Perspectiva

- A matriz que efetua a projeção perspectiva é dada por

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} x/(z/d) \\ y/(z/d) \\ z/(z/d) \\ 1 \end{bmatrix}$$

# Projeção Perspectiva



The diagram illustrates the structure of a perspective projection matrix. It shows a 4x4 matrix being multiplied by a 4x1 column vector. The 4x4 matrix is partitioned into three colored regions: a blue 3x3 block for scaling and rotation, a green 3x1 column for translation, and an orange 1x3 row for projection. The 4x1 vector contains the coordinates x, y, z, and 1. Arrows point from the labels to their respective parts in the matrix.

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} a & d & g & d_x \\ b & e & h & d_y \\ c & f & i & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Escalas, Rotações

Translações

Projeções

A projeção perspectiva pode ser incorporada na matriz de instanciamento (pipeline de visualização)

# Processo de Visualização 3D

---

1. Transformações de Instanciamento 3D
2. Recorte contra o volume de visualização
3. Determinação de partes visíveis da cena
4. Projeção

5. Recorte dos objetos contra a Window
6. Mapeamento Window-Viewport
7. Conversão de Varredura

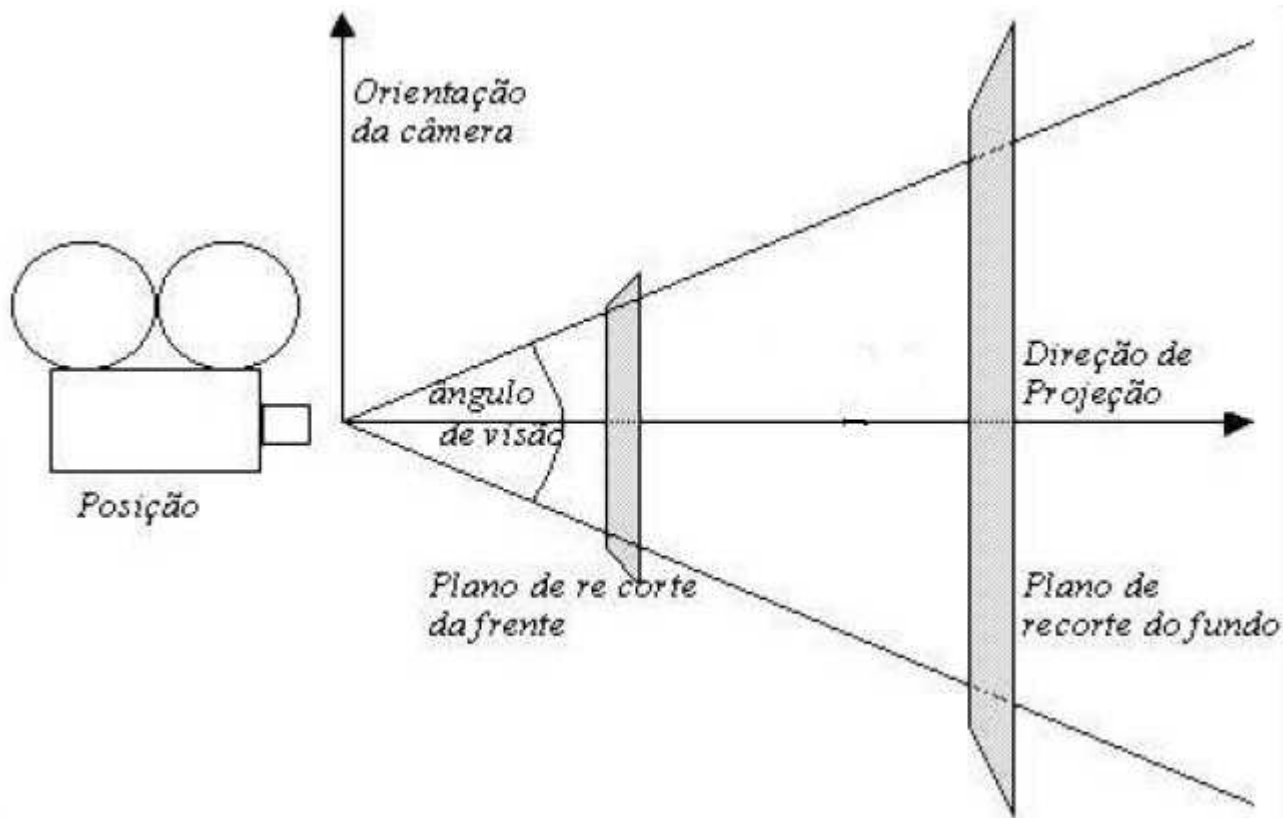
Visualização  
2D

# Transformações de Instanciamento 3D

---

- São aplicadas a todos os pontos da geometria do objeto
- Transformações no SRU
  - Escala
  - Rotação
  - Translação

# Recorte contra o volume de visualização





# Determinação de partes visíveis da cena

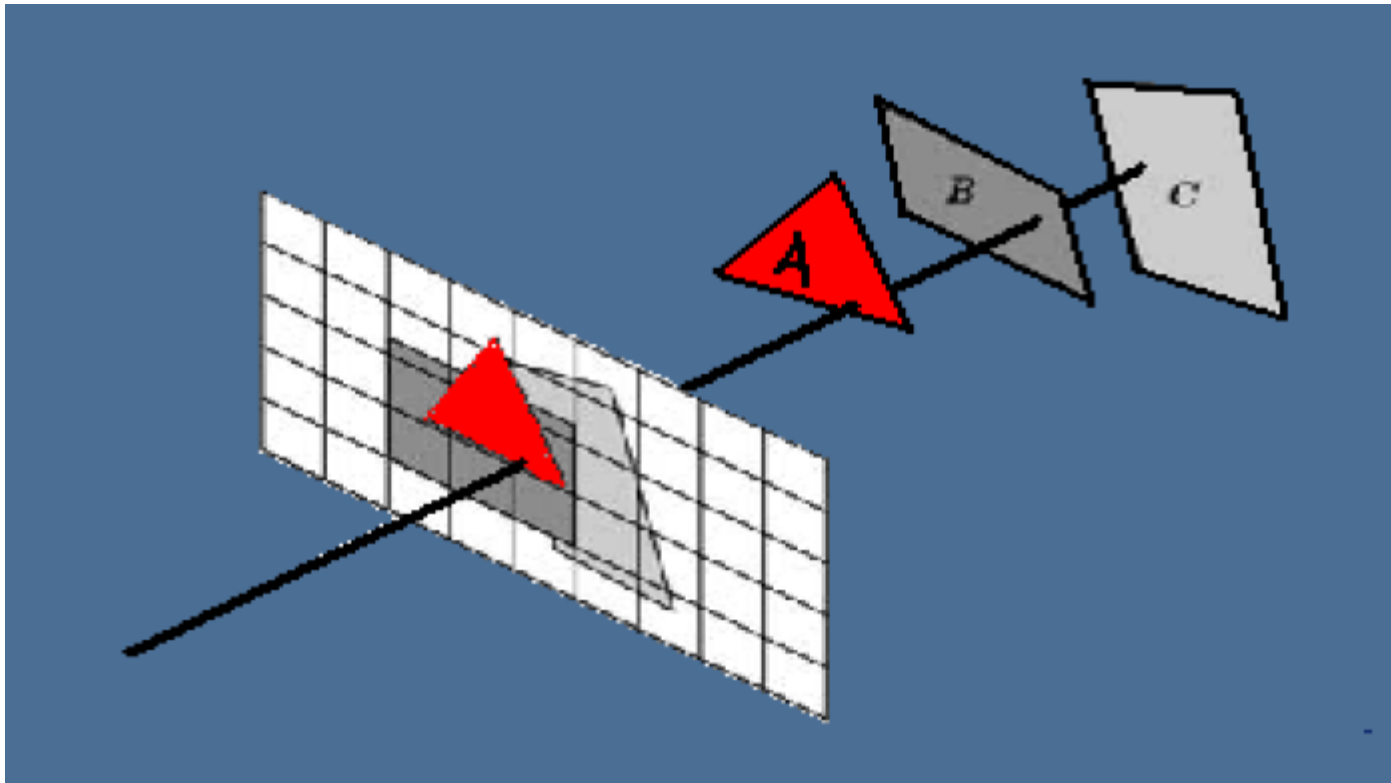
---

- Algoritmo Z-buffer
  - Mais comumente utilizado
- Algoritmo Scan-line
- Algoritmo do pintor
- Ray tracing de superfície visível
- Outros

# Z-buffer

- Para cada polígono do modelo
  - Projete o polígono na viewport
    - Para cada pixel no polígono
      - Calcule a cor do pixel (iluminação)
      - Calcule o z do pixel ( $z = x \cdot d / x_p$ )
      - Compare o z calculado com o z armazenado para o pixel no z-buffer
      - Se o z calculado é menor que o z armazenado então atualize a cor do pixel para a cor atual e o z do pixel no z-buffer para o z calculado

# Z-buffer



# A-buffer

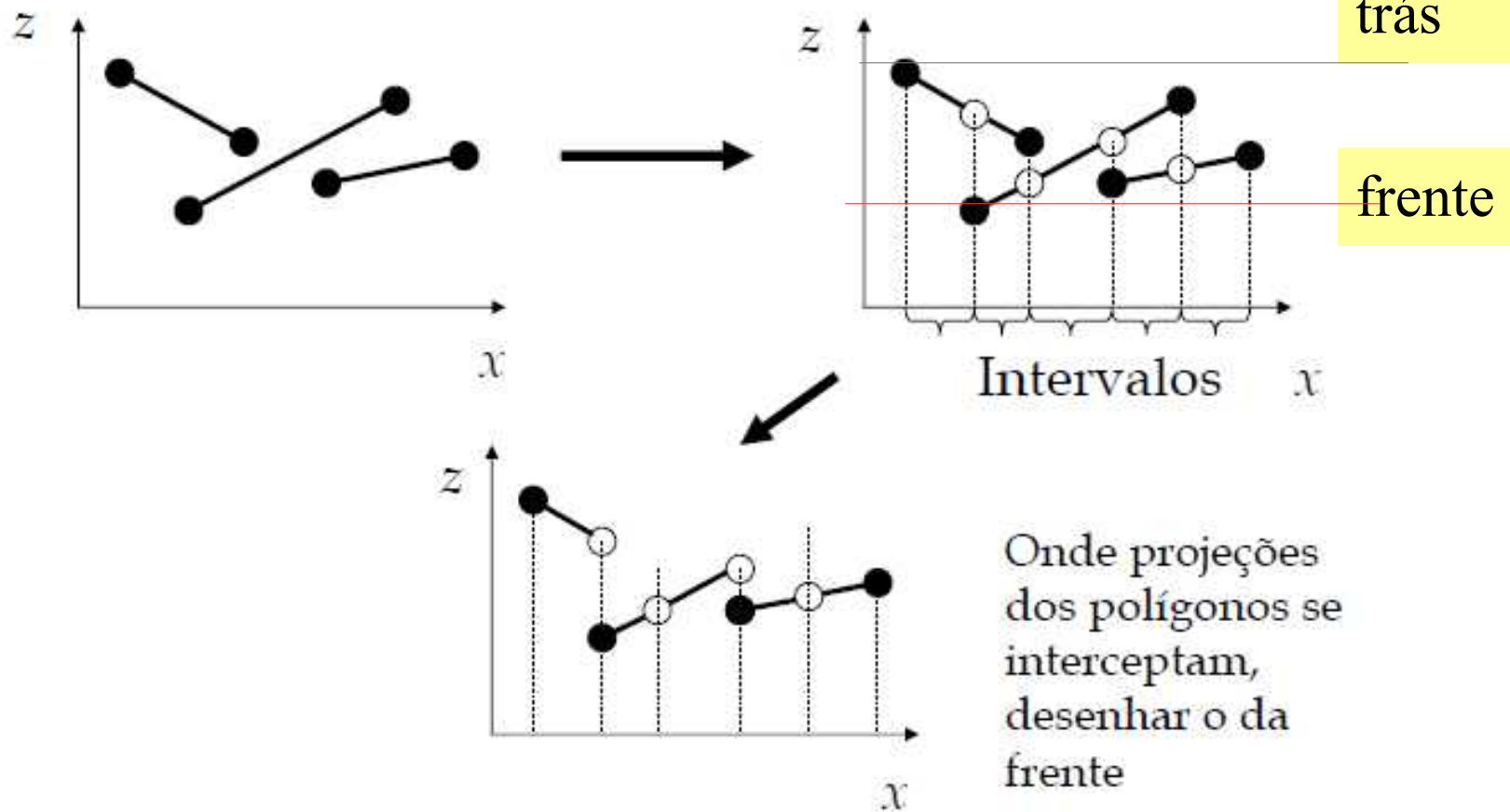
- Melhora do z-buffer para implementar transparência
- Fase 1: Polígonos são rasterizados
  - Se pixel coberto pelo polígono opaco
    - Inserir na lista removendo polígonos mais profundos
  - Se o polígono é transparente
    - Inserir na lista
- Fase 2: Geração da imagem
  - Máscaras de subpixels são misturadas para obter cor final do pixel

# Scan-line

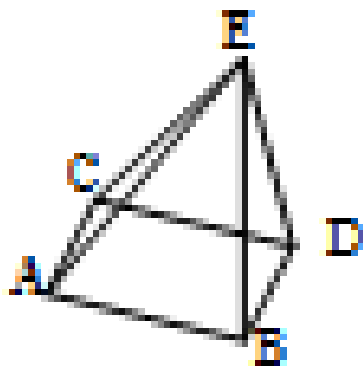
---

- Para cada linha de varredura na imagem (y)
  - Para cada pixel
    - Determine o objeto mais próximo
    - Calcule a cor do pixel
    - Pinte o pixel

# Scan-line



Faça a projeção da pirâmide abaixo, descrita no SRU, mão esquerda. Utilize  $d = 1$ .



$A = (1, 1, 1)$
$B = (3, 1, 1)$
$C = (1, 1, 3)$
$D = (3, 1, 3)$
$E = (2, 3, 2)$