

Pipeline de renderização

Transformações
Modelagem

Iluminação
(*Shading*)

Transformação
Câmera

Recorte

Projeção

Rasterização

Visibilidade

A história até aqui...

Adaptação e melhoramentos de uma aula sobre o mesmo assunto (MIT - EECS 6.837 Durand and Cutler)

Transformações
Modelagem

Iluminação
(*Shading*)

Transformação
Câmera

Recorte

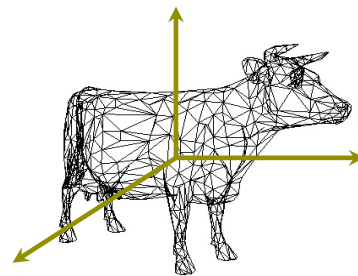
Projeção

Rasterização

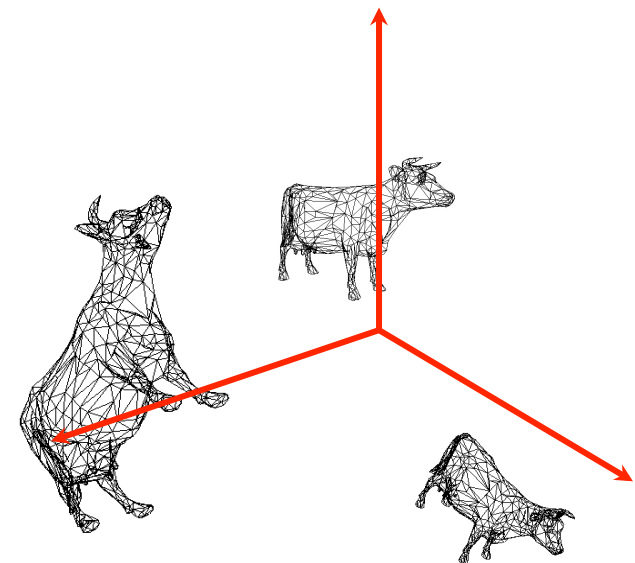
Visibilidade

✓ Objetos definidos no seu próprio sistema de coordenadas

✓ Transformações de modelagem orientam os modelos geométricos num sistema comum de coordenadas (UNIVERSO)



Coordenadas Objeto



Coordenadas Universo

Transformações
Modelagem

Iluminação
(*Shading*)

Transformação
Câmera

Recorte

Projeção

Rasterização

Visibilidade

✓ Vértices iluminados de acordo com as propriedades geométricas e de material

✓ Modelo de Iluminação Local



Transformações
Modelagem

Iluminação
(*Shading*)

Transformação
Câmera

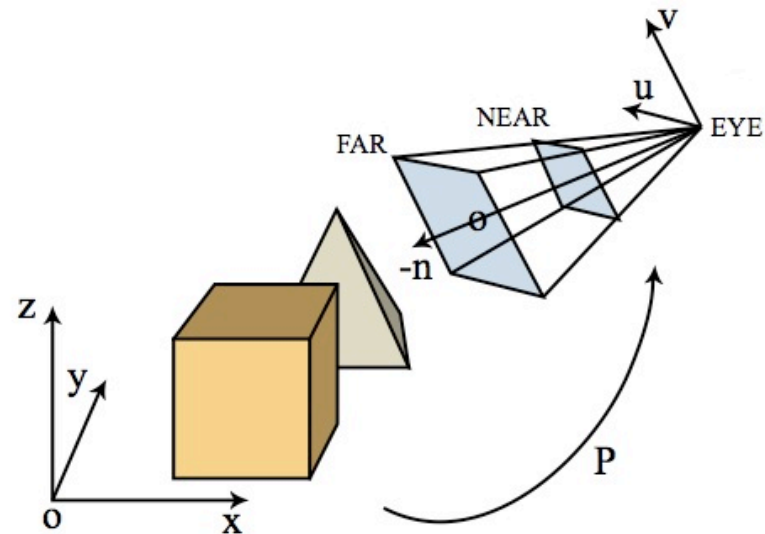
Recorte

Projeção

Rasterização

Visibilidade

✓ Mapeamento de coordenadas de Universo para câmera



Transformações
Modelagem

Iluminação
(*Shading*)

Transformação
Câmera

Recorte

Projeção

Rasterização

Visibilidade

✓ Escolha da projeção: perspectiva ou ortográfica

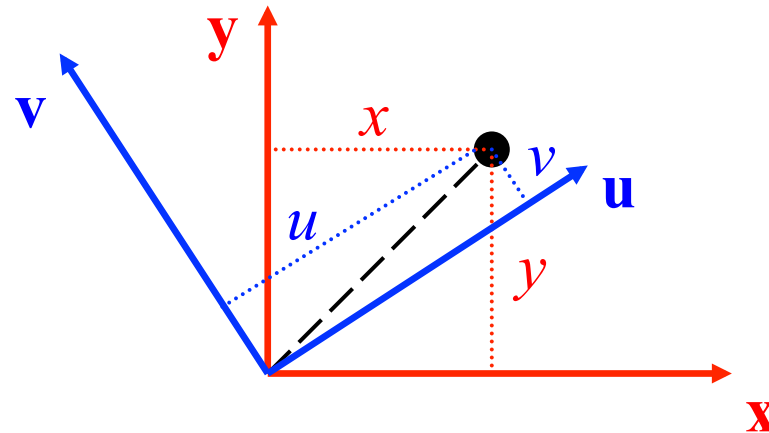


*Já vimos Recorte 2D em aulas anteriores. Precisamos apenas ver a extensão para 3D

Transformação de Câmera

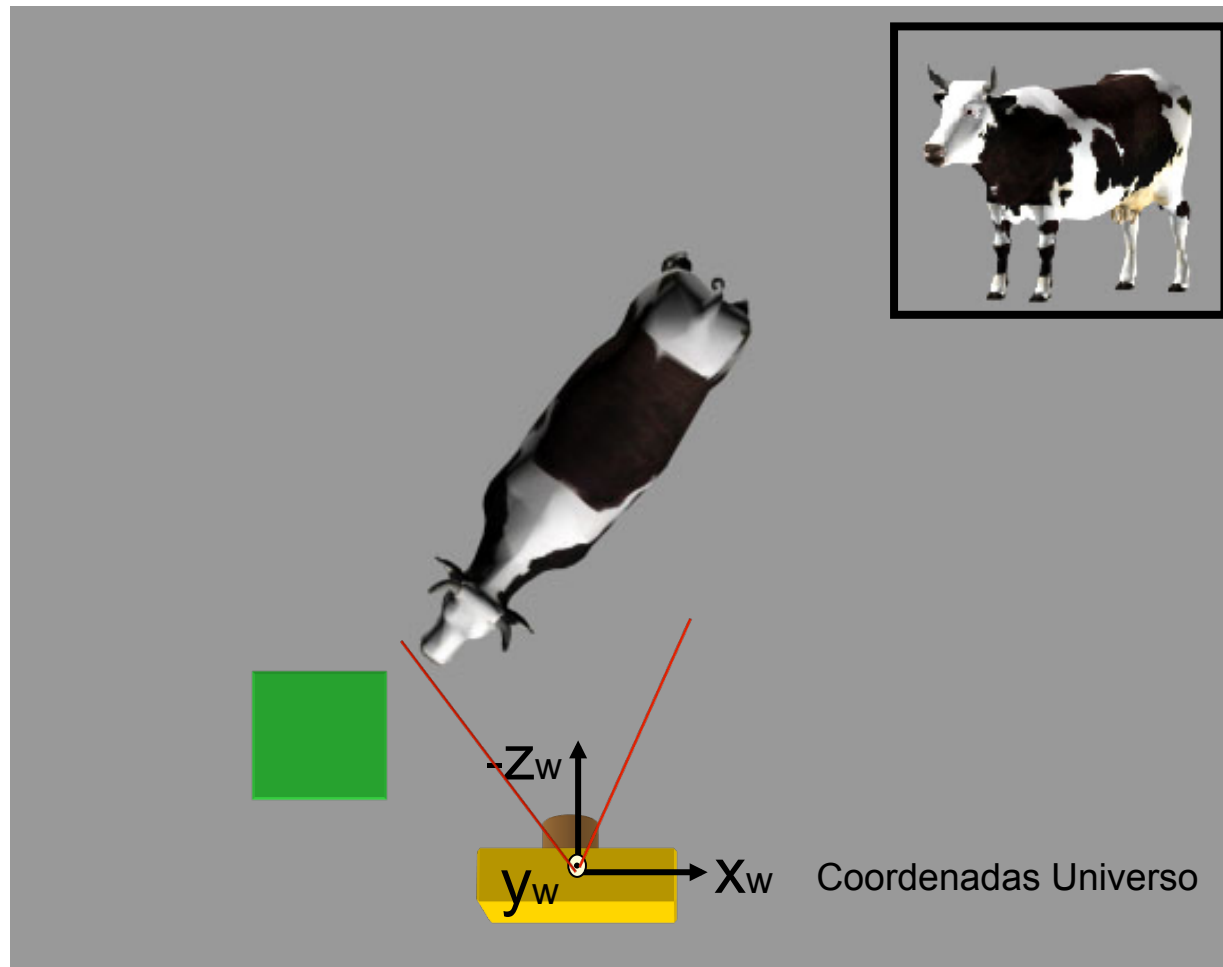
- Transformar as coordenadas, ou seja, dado os sistemas de coordenadas xyz e uvn e um ponto $p = (x, y, z)$ encontrar $p = (u, v, n)$

Em 2D esquematicamente:



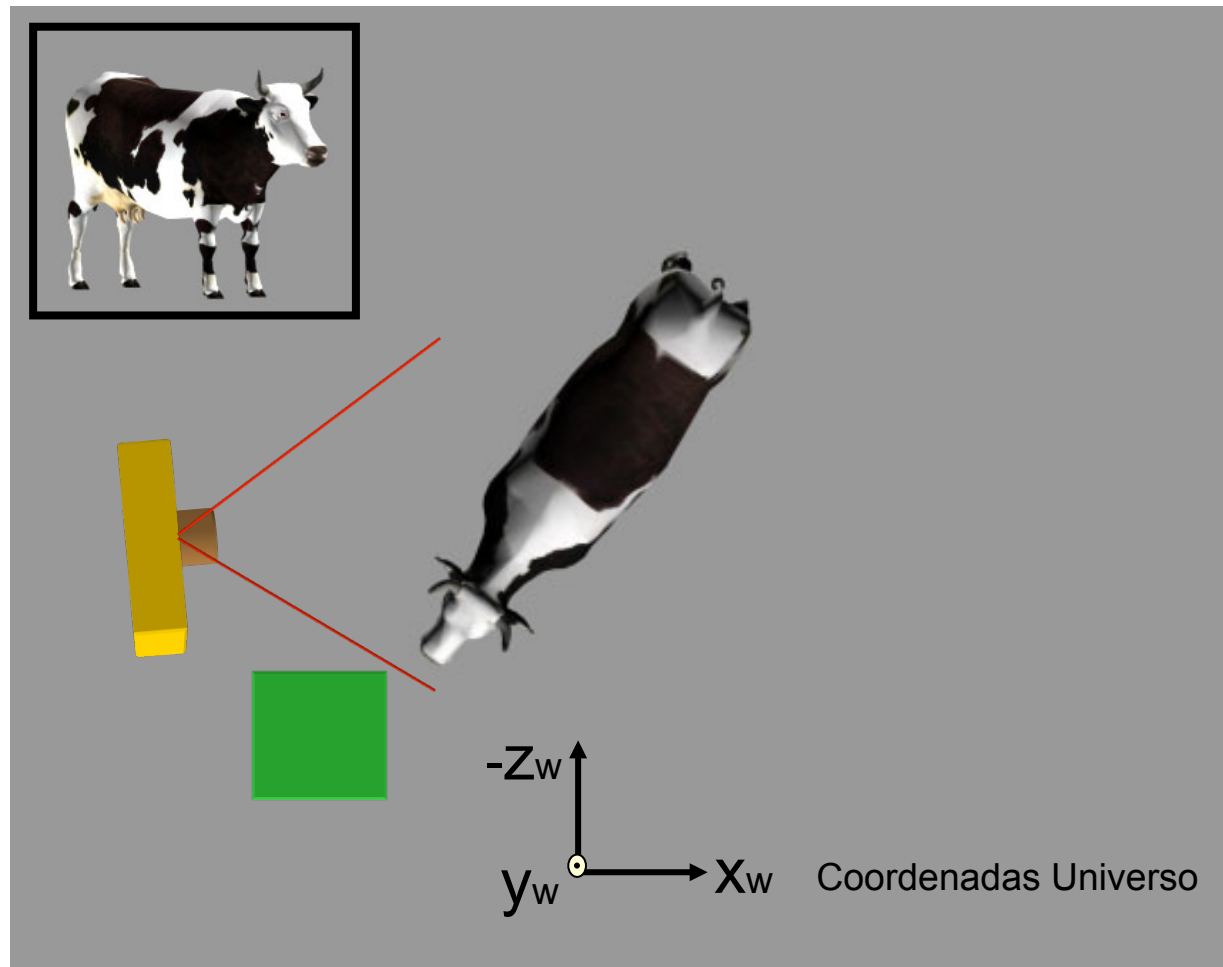
A posição espacial do ponto não muda. Suas coordenadas são expressas em outro sistema

Mudando a posição do observador



Modificado de M.M. Oliveira

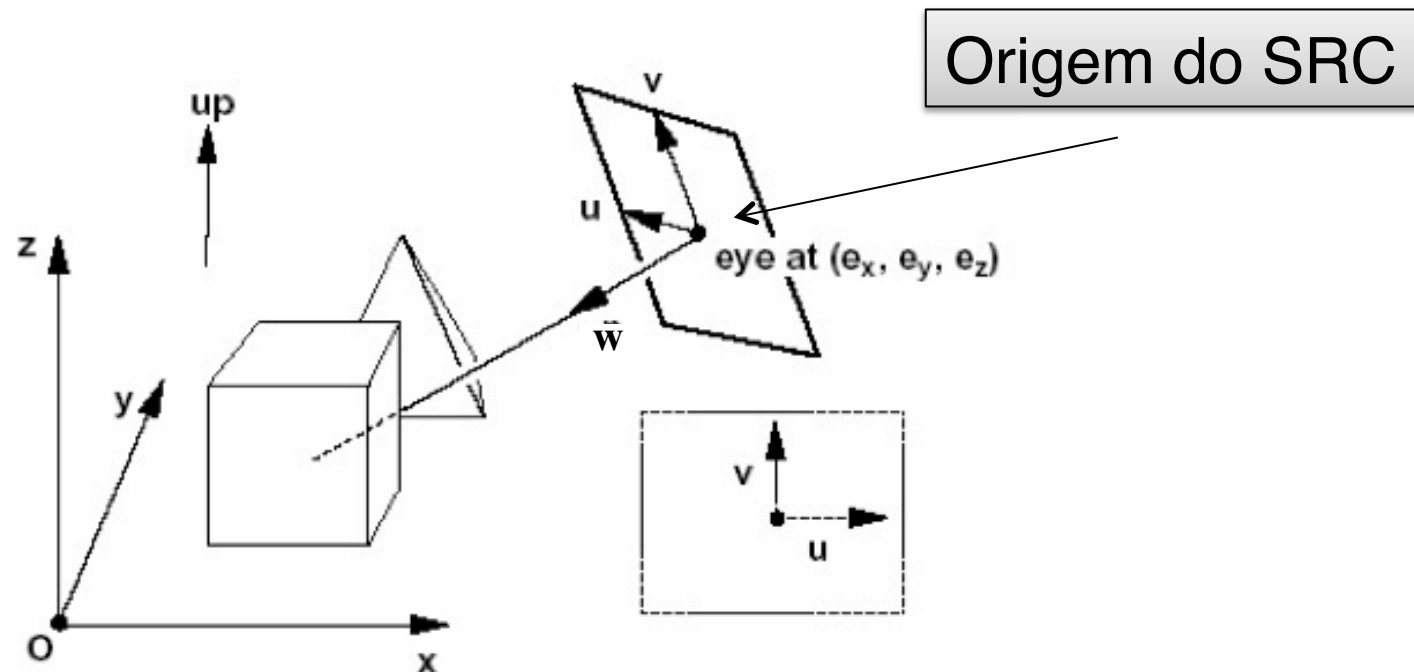
Mudando a posição do observador



Modificado de M.M. Oliveira

Sist. Referência Câmera (SRC)

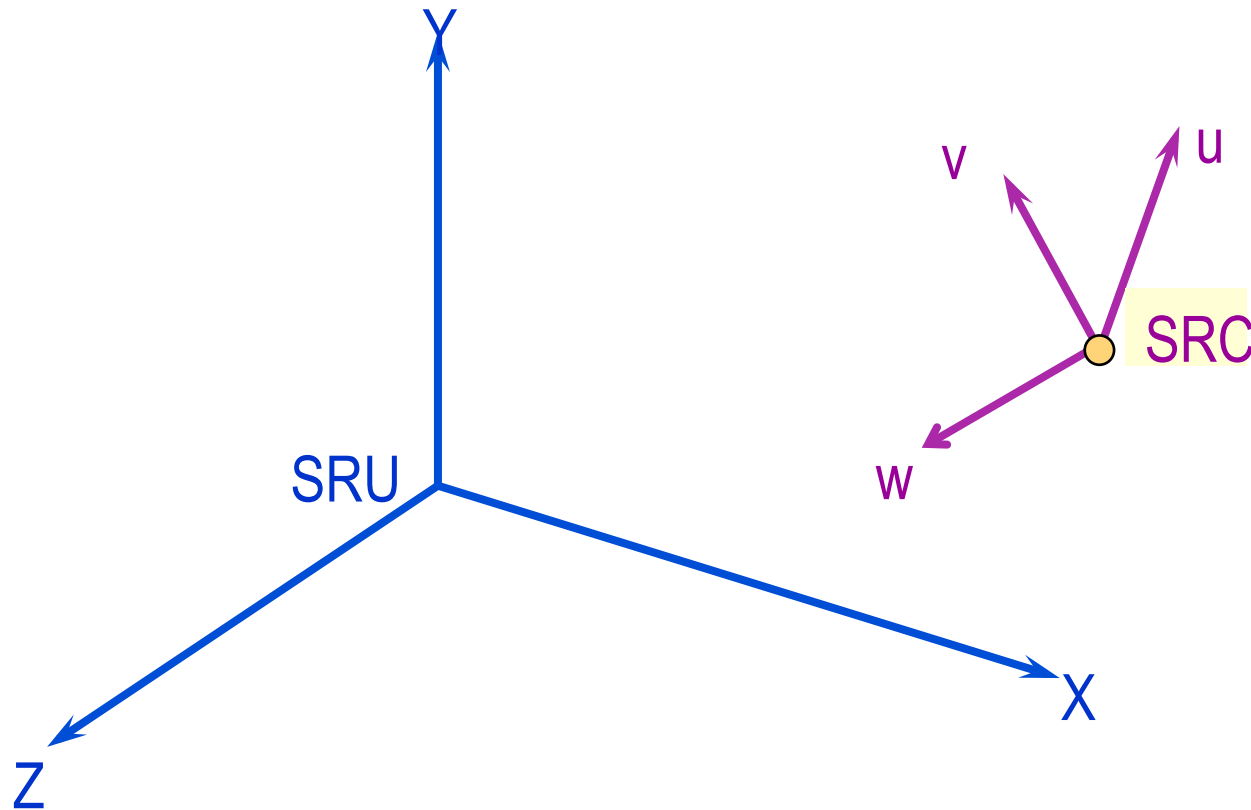
- Envolve uma translação à origem do novo sistema e uma mudança de base ortogonal



Definindo o SRC

- Precisamos especificar:
 - Origem do SRC: ponto qualquer no espaço
 - Direção para onde a câmera está apontando
 - Direção “para cima”: up vector
- A partir destas três informações um sistema de coordenadas ortogonal pode ser estabelecido

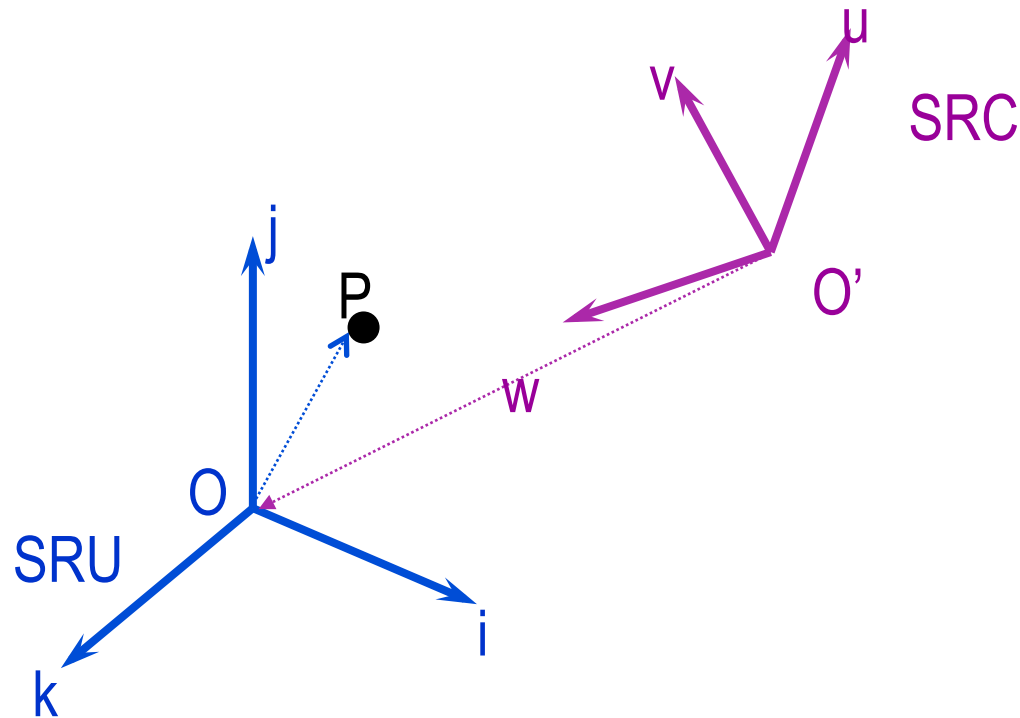
Sistema de referência da câmera (SRC)



IMPORTANTE:

SRU é “mão **direita**” e o SRC é “mão **esquerda**”

Transformação SRU \rightarrow SRC



$$P = O' + x_c u + y_c v + z_c w = (x, y, z)_{SRC}$$

$$P = O + x_u i + y_u j + z_u k = (x, y, z)_{SRU}$$

Transformação SRU \rightarrow SRC

$$P = O + x_u i + y_u j + z_u k = (x, y, z)_{\text{SRU}}$$

$$P = O + [i \ j \ k] \begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix}$$

$$P = O' + x_c u + y_c v + z_c w = (x, y, z)_{\text{SRC}}$$

$$P = \underbrace{[i \ j \ k] \begin{bmatrix} x_{o'} \\ y_{o'} \\ z_{o'} \end{bmatrix}}_{O'} + [u \ v \ w] \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Expressando a base do SRC em relação ao SRU

$$\mathbf{u} = a_{11}\mathbf{i} + a_{21}\mathbf{j} + a_{31}\mathbf{k}$$

$$\mathbf{v} = a_{12}\mathbf{i} + a_{22}\mathbf{j} + a_{32}\mathbf{k}$$

$$\mathbf{w} = a_{13}\mathbf{i} + a_{23}\mathbf{j} + a_{33}\mathbf{k}$$

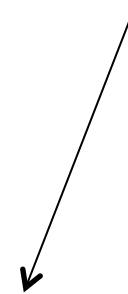
$$[\mathbf{u} \ \mathbf{v} \ \mathbf{w}] = [\mathbf{i} \ \mathbf{j} \ \mathbf{k}] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$[\mathbf{u} \ \mathbf{v} \ \mathbf{w}] = [\mathbf{i} \ \mathbf{j} \ \mathbf{k}] \mathbf{B}$$

\mathbf{B} = Matriz mudança de Base

Transformando...

$$O + [i \ j \ k] \begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = [i \ j \ k] \begin{bmatrix} x_{o'} \\ y_{o'} \\ z_{o'} \end{bmatrix} + [u \ v \ w] \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$



$$[i \ j \ k] \begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = [i \ j \ k] \begin{bmatrix} x_{o'} \\ y_{o'} \\ z_{o'} \end{bmatrix} + [i \ j \ k] \mathbf{B} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Transformando...

$$\begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = \begin{bmatrix} x_{o'} \\ y_{o'} \\ z_{o'} \end{bmatrix} + \mathbf{B} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

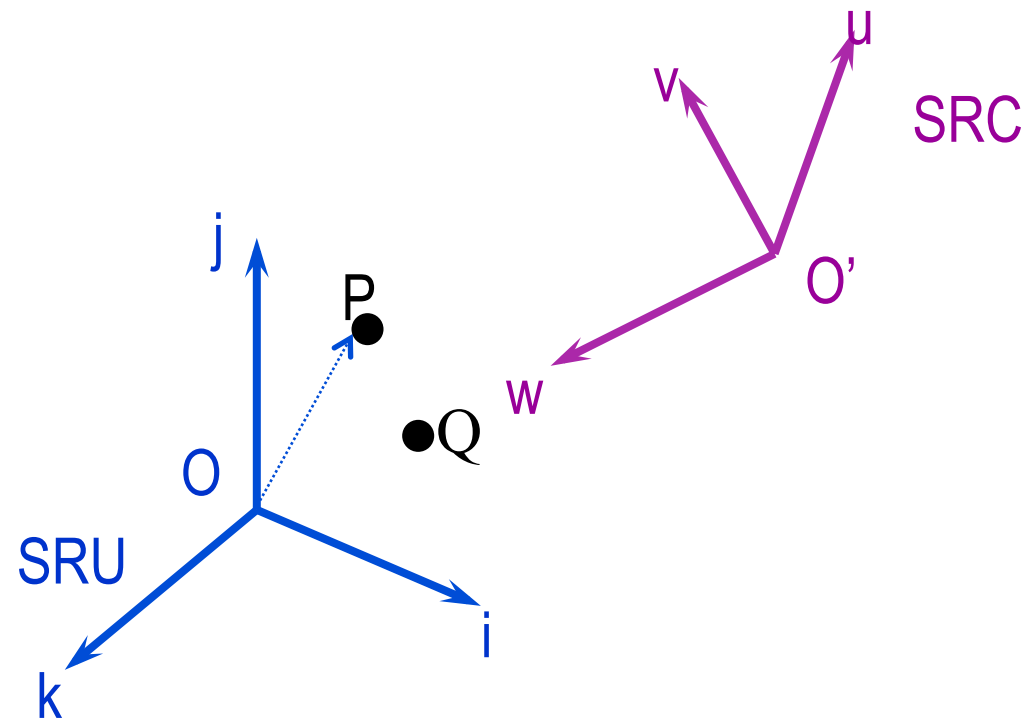
$$\begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} - \begin{bmatrix} x_{o'} \\ y_{o'} \\ z_{o'} \end{bmatrix} = \mathbf{B} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Finalmente...

$$B^{-1} \begin{bmatrix} x_u - x_{o'} \\ y_u - y_{o'} \\ z_u - z_{o'} \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

- Considerando propriedades das bases vetoriais em uso:
 - $B^{-1} = B^T$

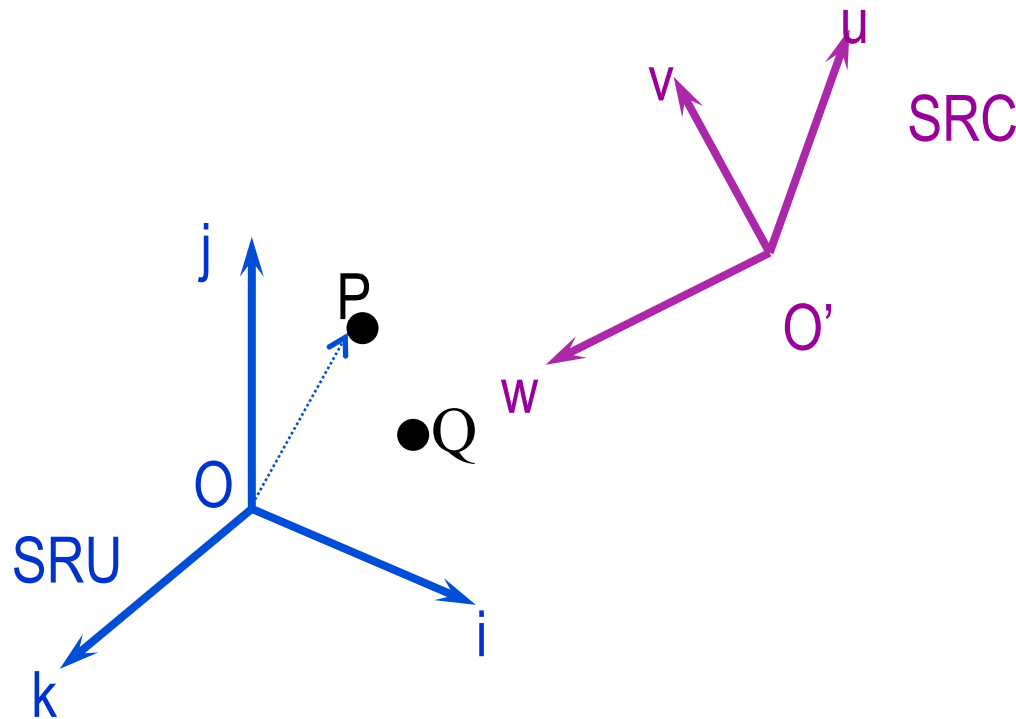
Como calcular a matriz B?



- Dados
 - up vector
 - O'
 - Alvo (O ou outro ponto qualquer Q)

Como calcular a matriz B?

- $w = \text{normalize}(O'-O)$
- $u = w \times (\text{up vector})$
- $v = u \times w$

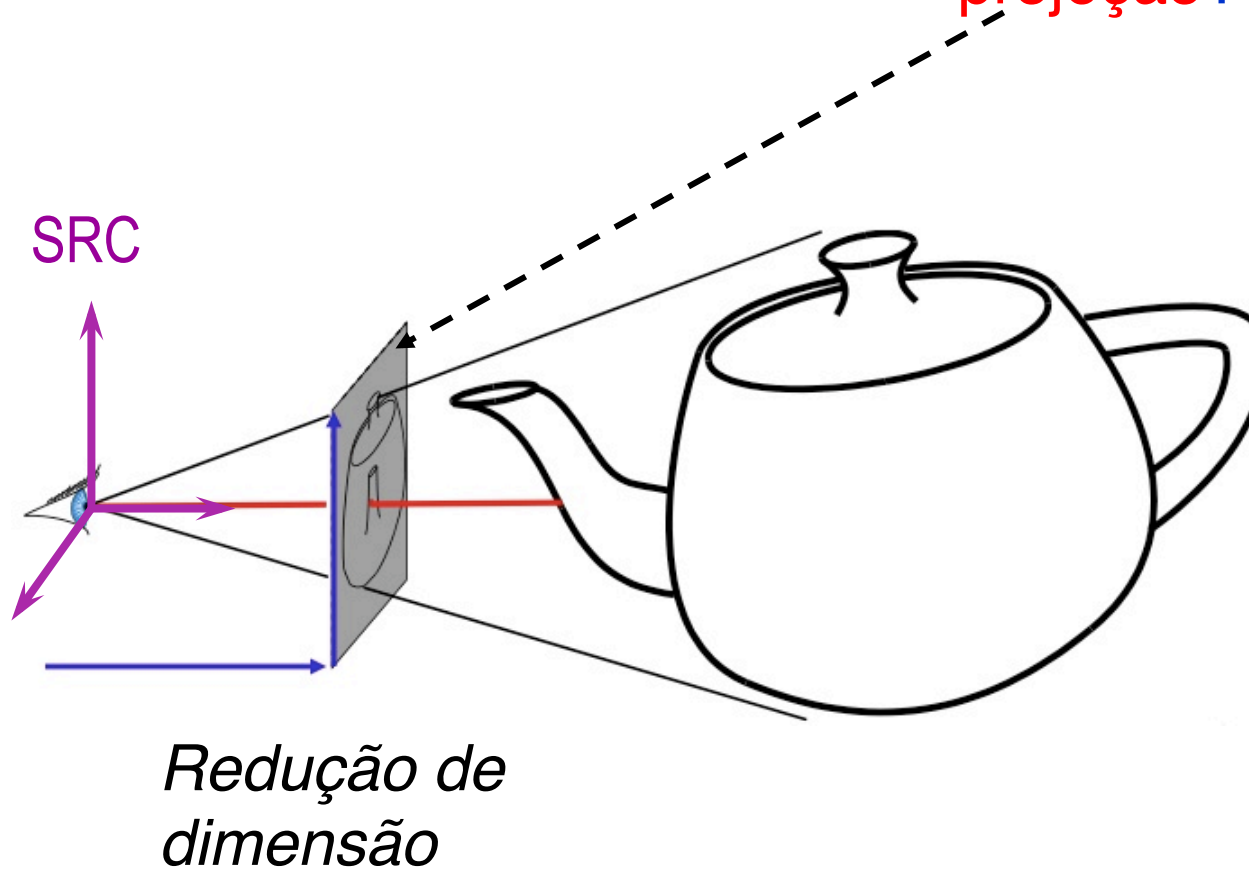


$$\begin{pmatrix} u_i & u_j & u_k \\ v_i & v_j & v_k \\ w_i & w_j & w_k \end{pmatrix}$$

Todos normalizados

Projeções

- Como é obtida a imagem do objeto no plano de projeção?

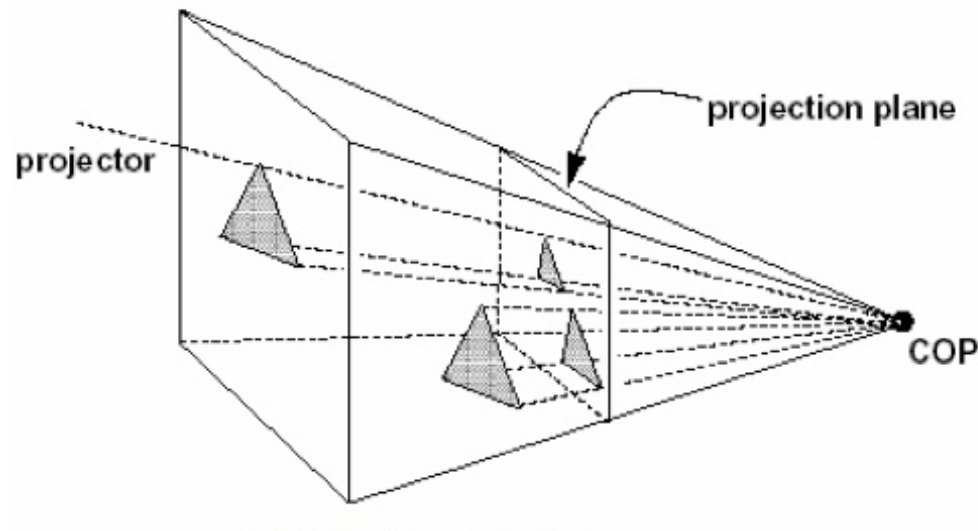


Projeções

- Genericamente, projeções transformam pontos em um sistema de coordenadas com N dimensões em pontos num sistema com dimensão menor do que N
- Para nós interessa apenas o caso de $3D \rightarrow 2D$

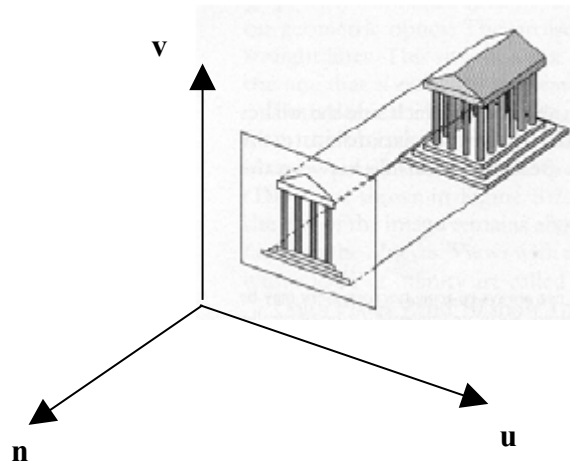
Projeções

- Uso de linhas de projeção, denominadas *raios de projeção*
- Origem em um *centro de projeção* (COP) e passam por cada parte do objeto (no nosso caso vértices) interseccionando um *plano de projeção* onde finalmente tem-se as coordenadas 2D



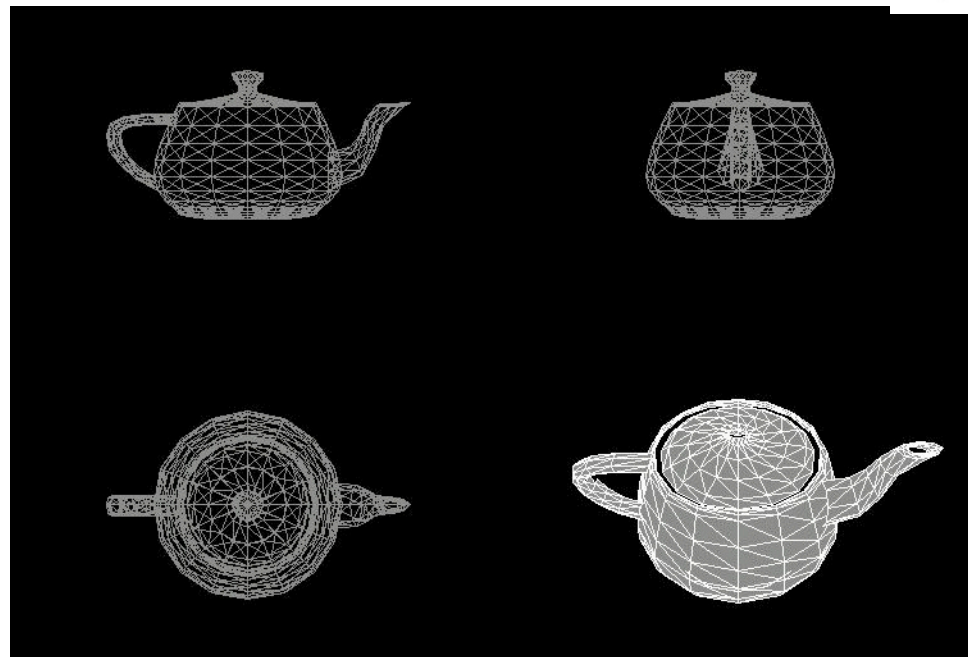
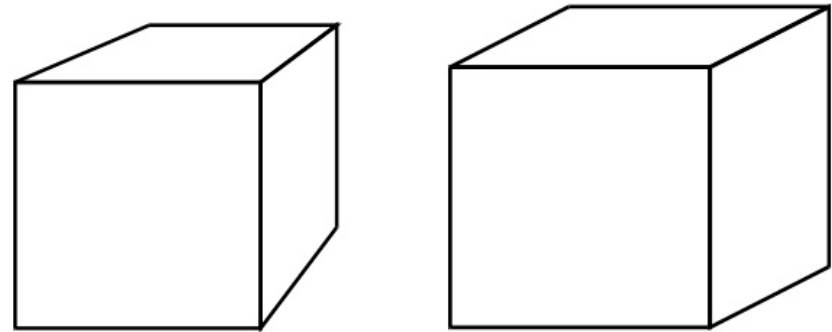
Tipos de Projeções

- Em CG 2 tipos: perspectiva e ortográfica
- Diferença: na projeção ortográfica o COP está no infinito, logo os raios de projeção são paralelos uns aos outros



Tipos de Projeções

- Diferença Visual



Qual é qual?

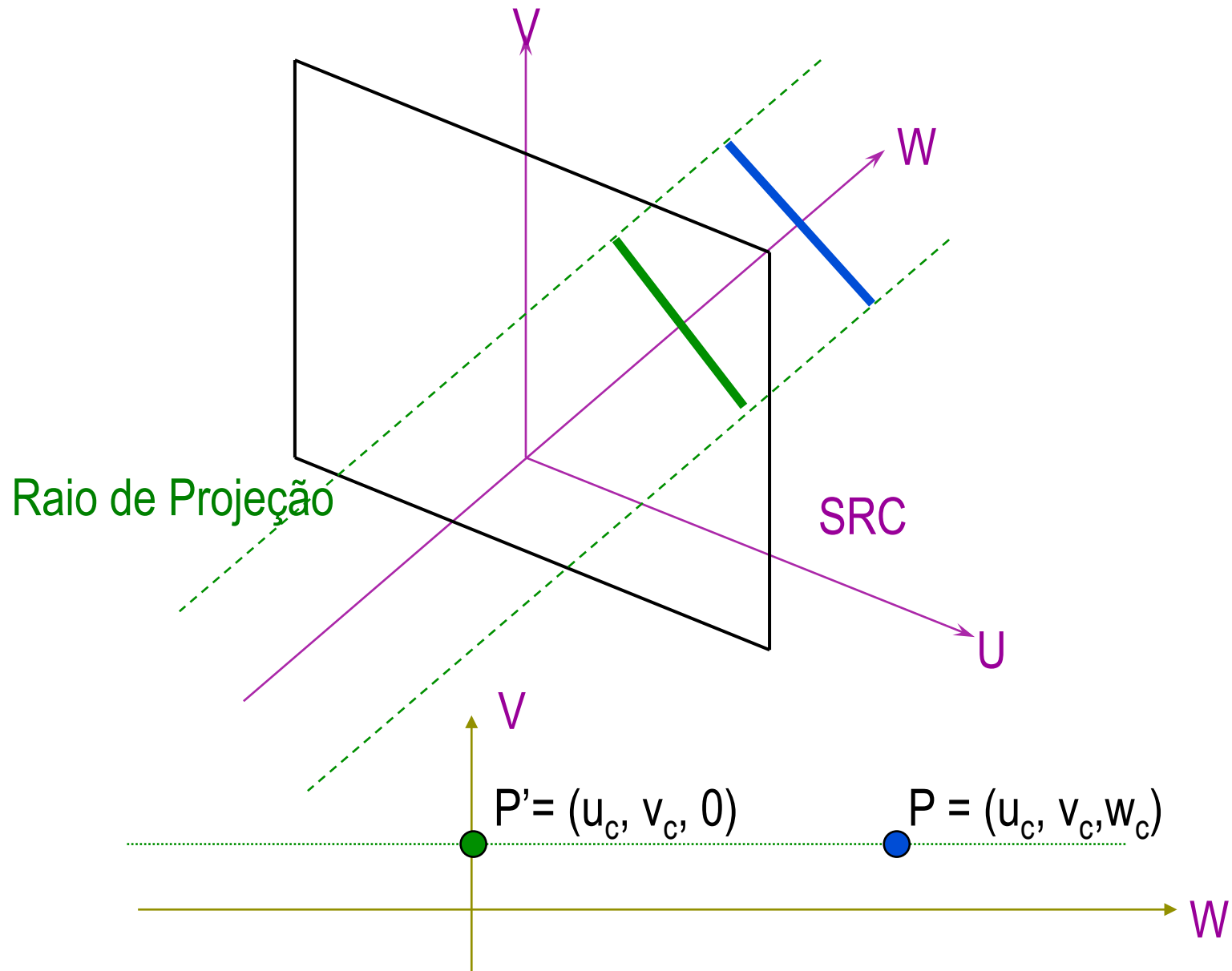
3 vistas ortográficas e uma perspectiva

Perspectiva nas Artes

- Primeira pintura em perspectiva
 - *Trinity with the Virgin, St. John and Donors*
 - Masaccio, 1427



Projeção Ortográfica

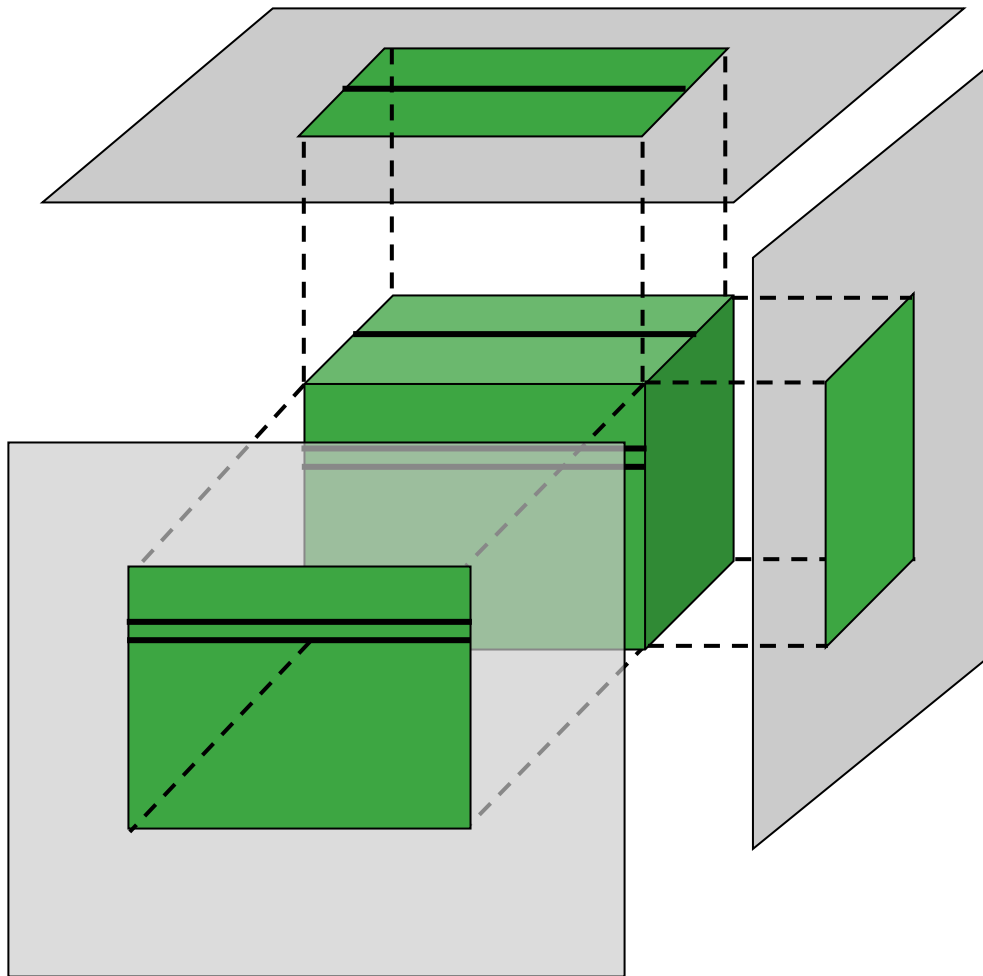


Obtendo Coordenadas de Projeção

- Precisamos “livrar-nos” de uma dimensão
- Para projeções ortográficas é simples:
 - Basta descartarmos a coordenada equivalente a w no SRC (assumindo o plano de projeção em $w = 0$)

$$\begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Vistas ortográficas



- Mais comuns
 - Front-elevation
 - Side-elevation
 - Plan-elevation
- Direção de projeção paralela a um dos eixos principais (x, y, z)
- Plano de projeção perpendicular ao eixo

Projeções ortográficas

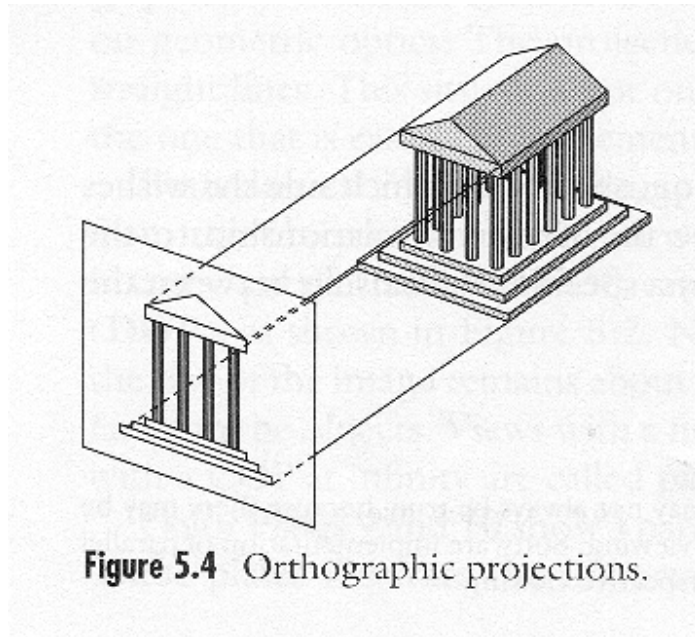


Figure 5.4 Orthographic projections.

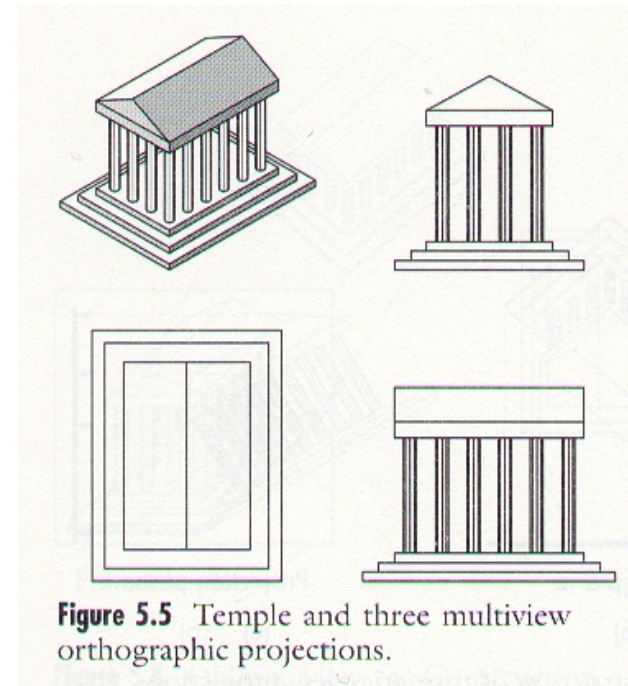
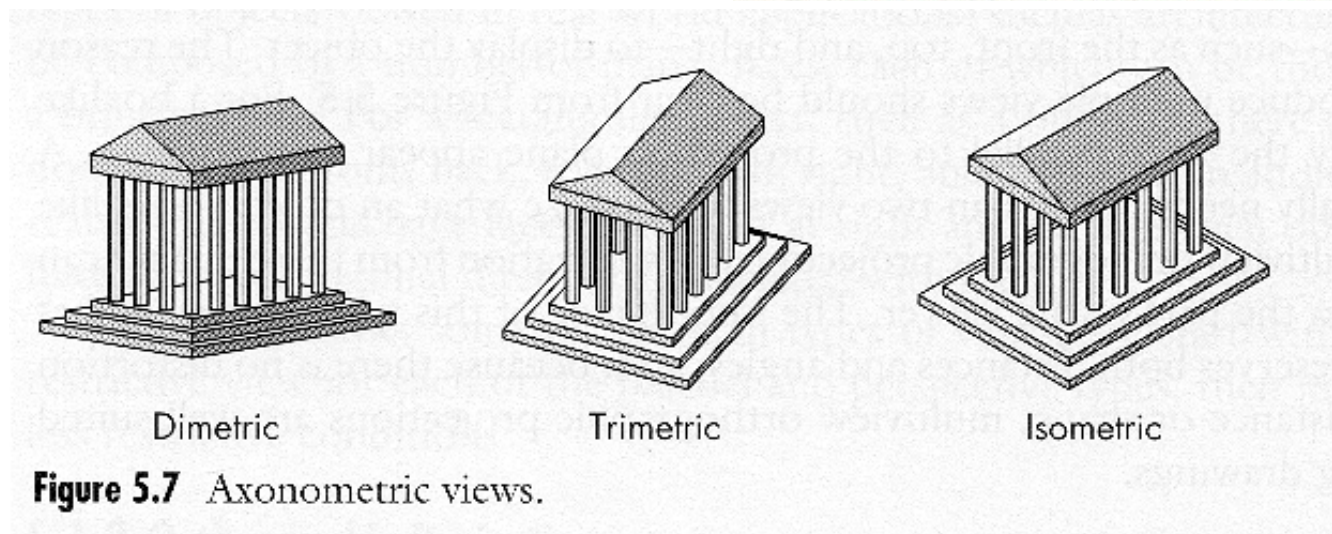


Figure 5.5 Temple and three multiview orthographic projections.



Dimetric

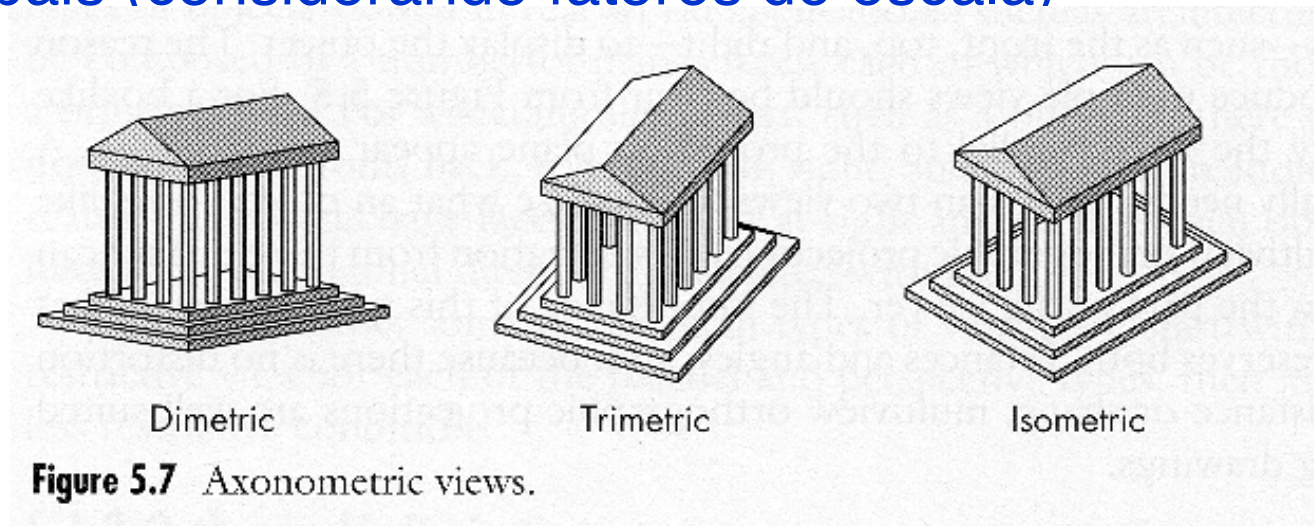
Trimetric

Isometric

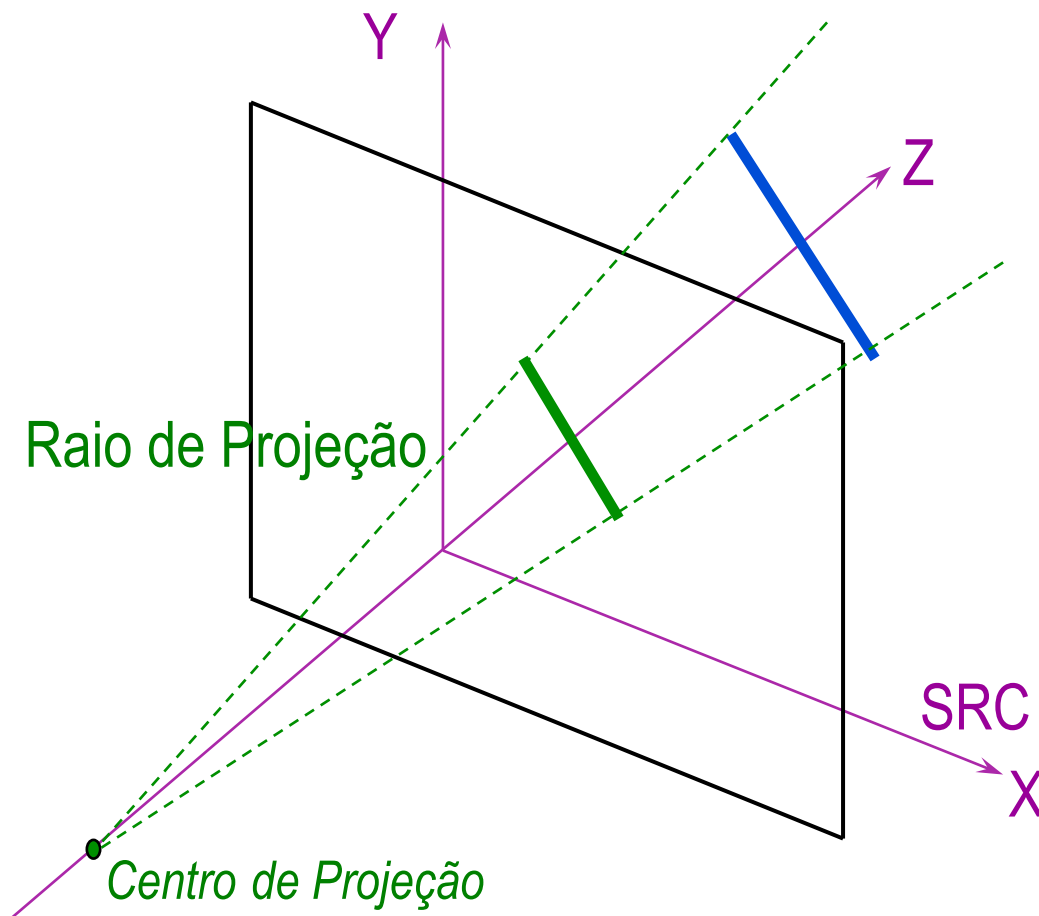
Figure 5.7 Axonometric views.

Projeções ortográficas axonométricas

- Plano de projeção NÃO é perpendicular a um dos eixos principais
- Amostra várias faces do objeto ao mesmo tempo
- É preservado o paralelismo entre as linhas
- Não são preservados ângulos entre as linhas
- Distâncias podem ser medidas ao longo dos eixos principais (considerando fatores de escala)



Projeção perspectiva



Projeção perspectiva

- **Propriedades:**
 - tamanho da projeção de um objeto varia inversamente com a distância ao centro de projeção
 - Linhas paralelas, em geral, não são projetadas paralelamente
 - Ângulos só são preservados nas faces paralelas ao plano de projeção
 - Distâncias não são preservadas

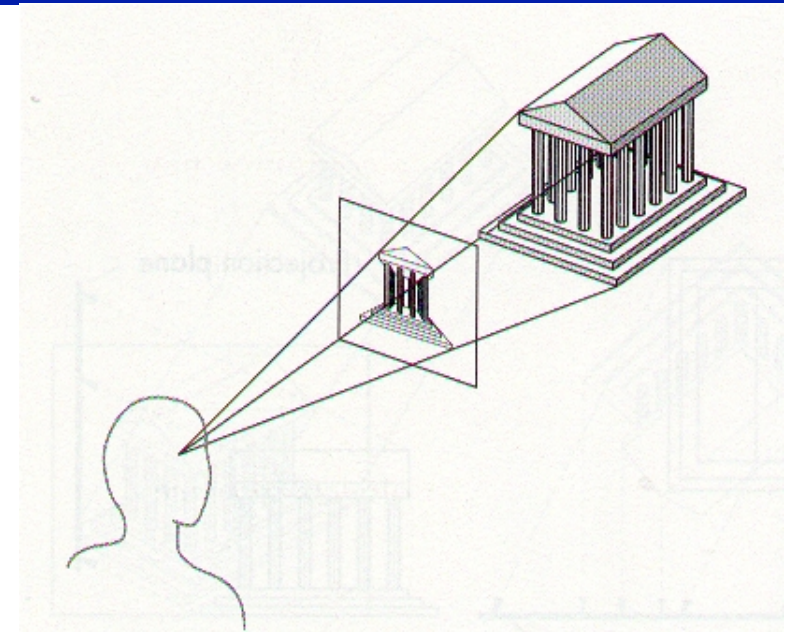


Figure 5.9 Perspective viewing.

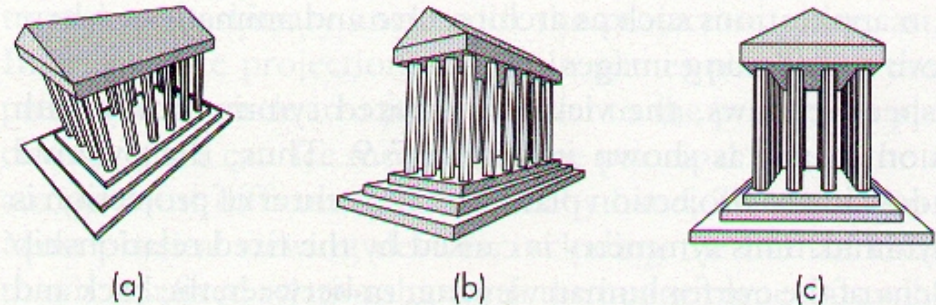
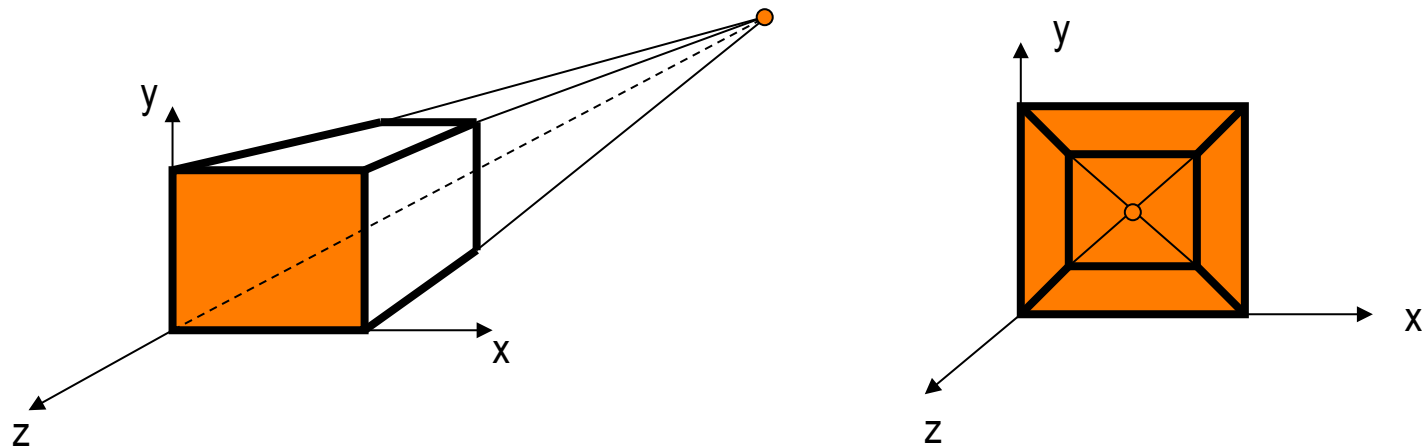


Figure 5.10 Classical perspective views: The (a) three-, (b) two-, and (c) one-point perspectives.

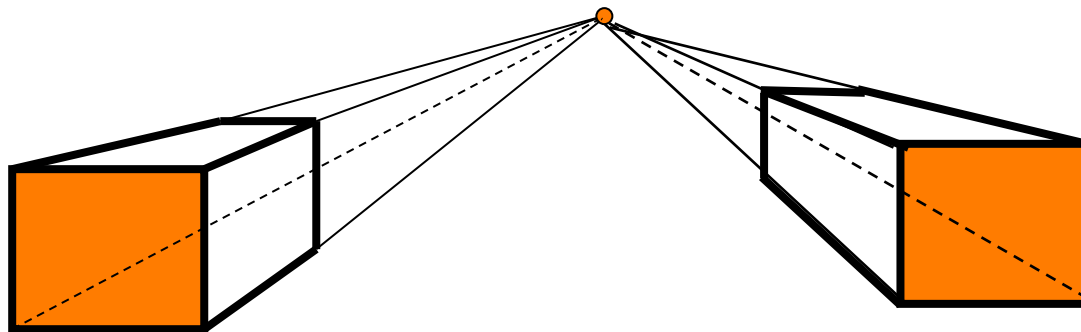
Projeção perspectiva

- Linhas paralelas a um eixo principal convergem para o ponto de fuga de um eixo (onde o eixo intercepta o plano de projeção)
 - Perspectiva é classificada conforme o número de pontos de fuga
 - Corresponde ao número de eixos interceptados pelo plano de projeção

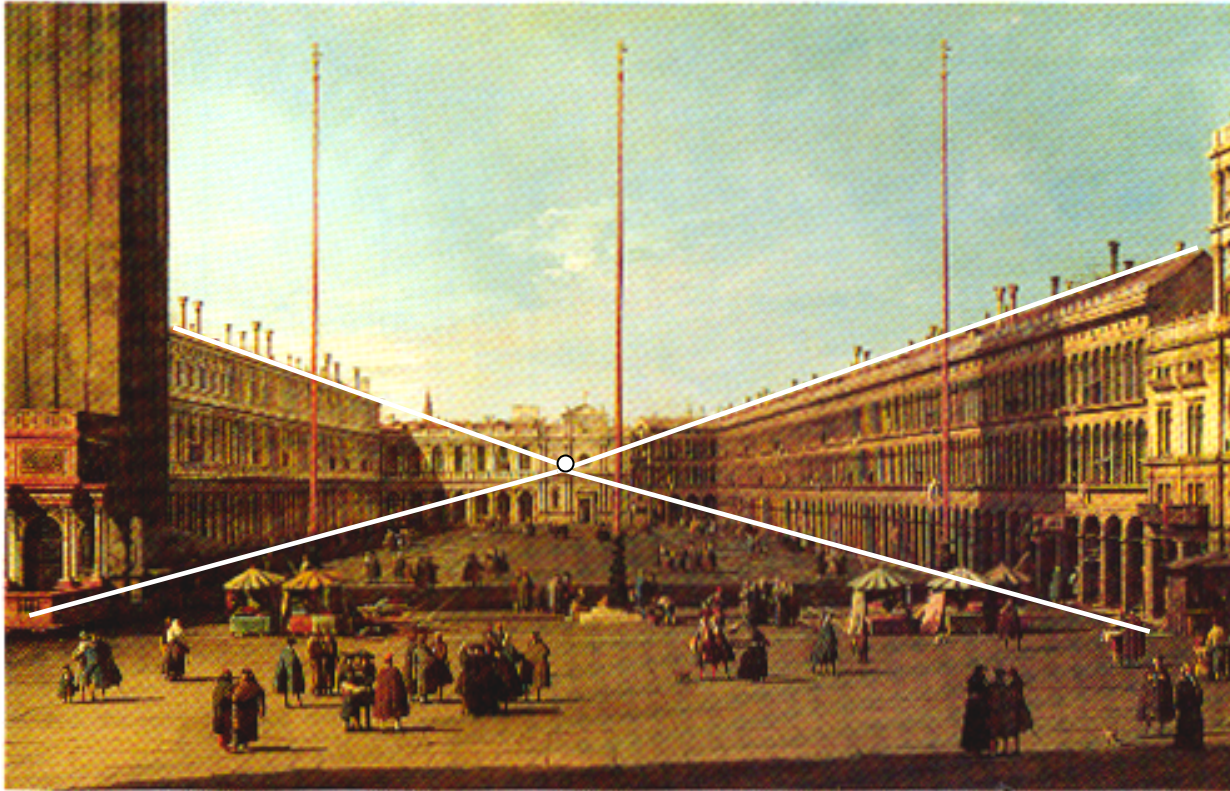


1-point perspective

- Plano de projeção corta apenas um eixo

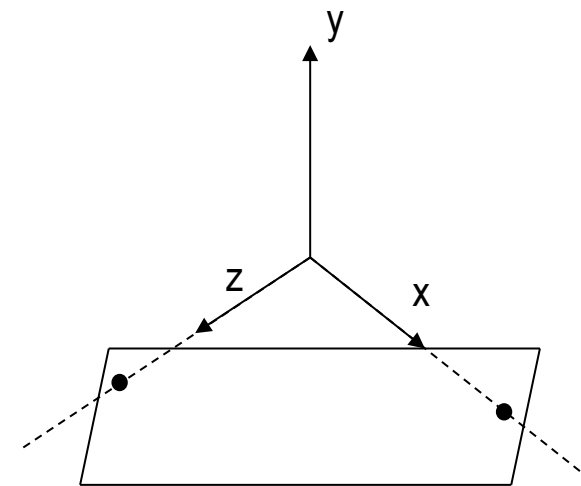
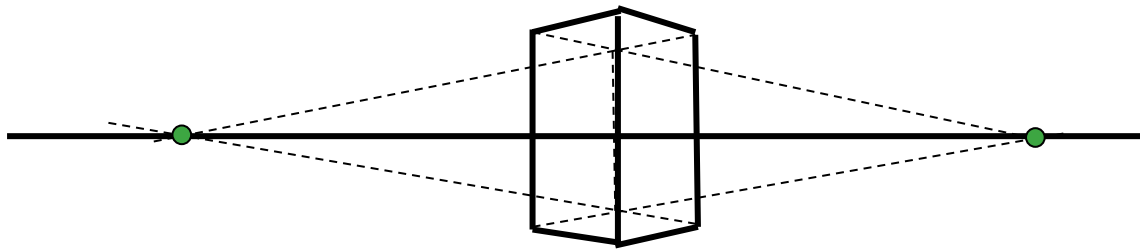


1-point perspective



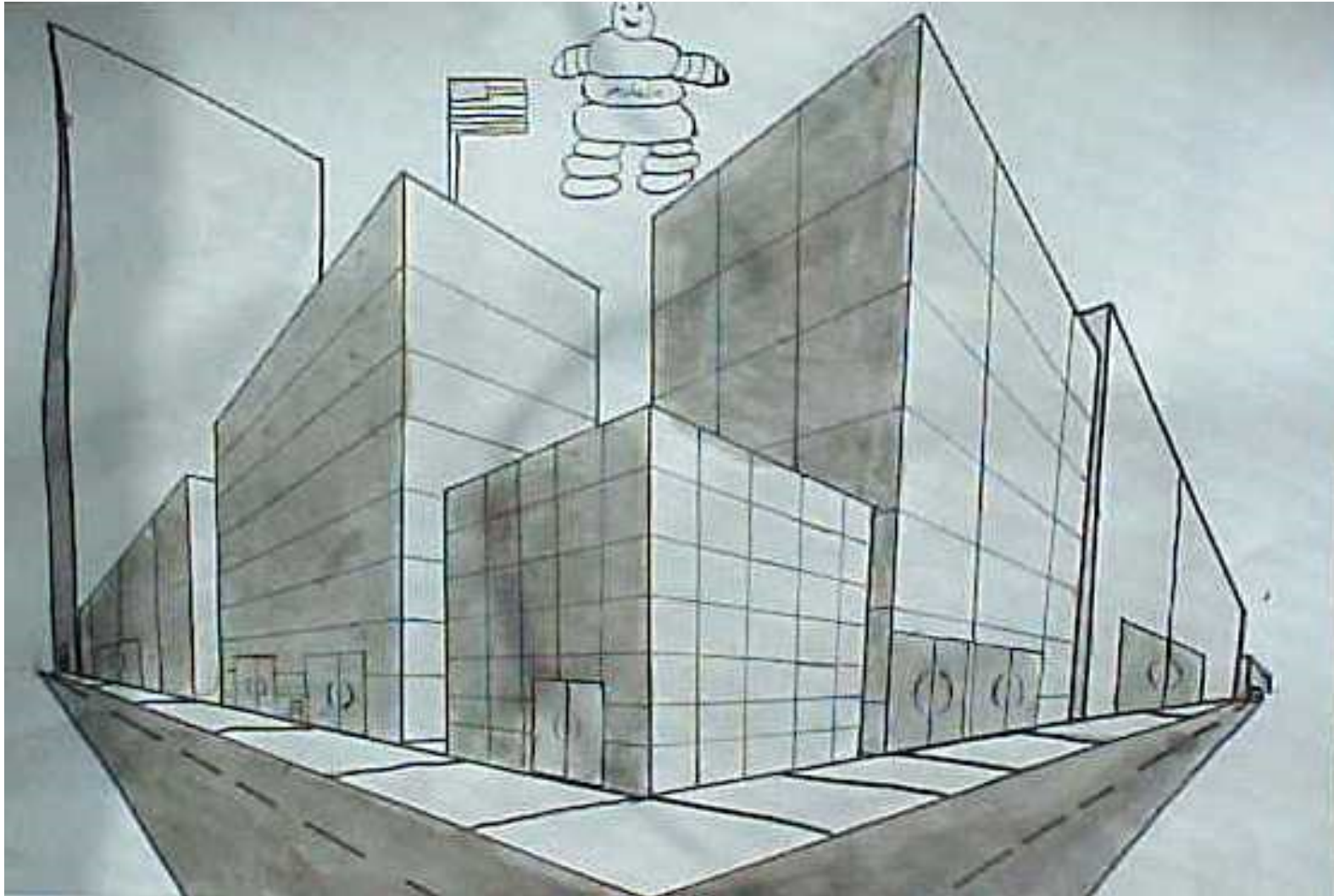
- A painting (*The Piazza of St. Mark, Venice*) done by Canaletto in 1735-45 in one-point perspective.

2-point perspective

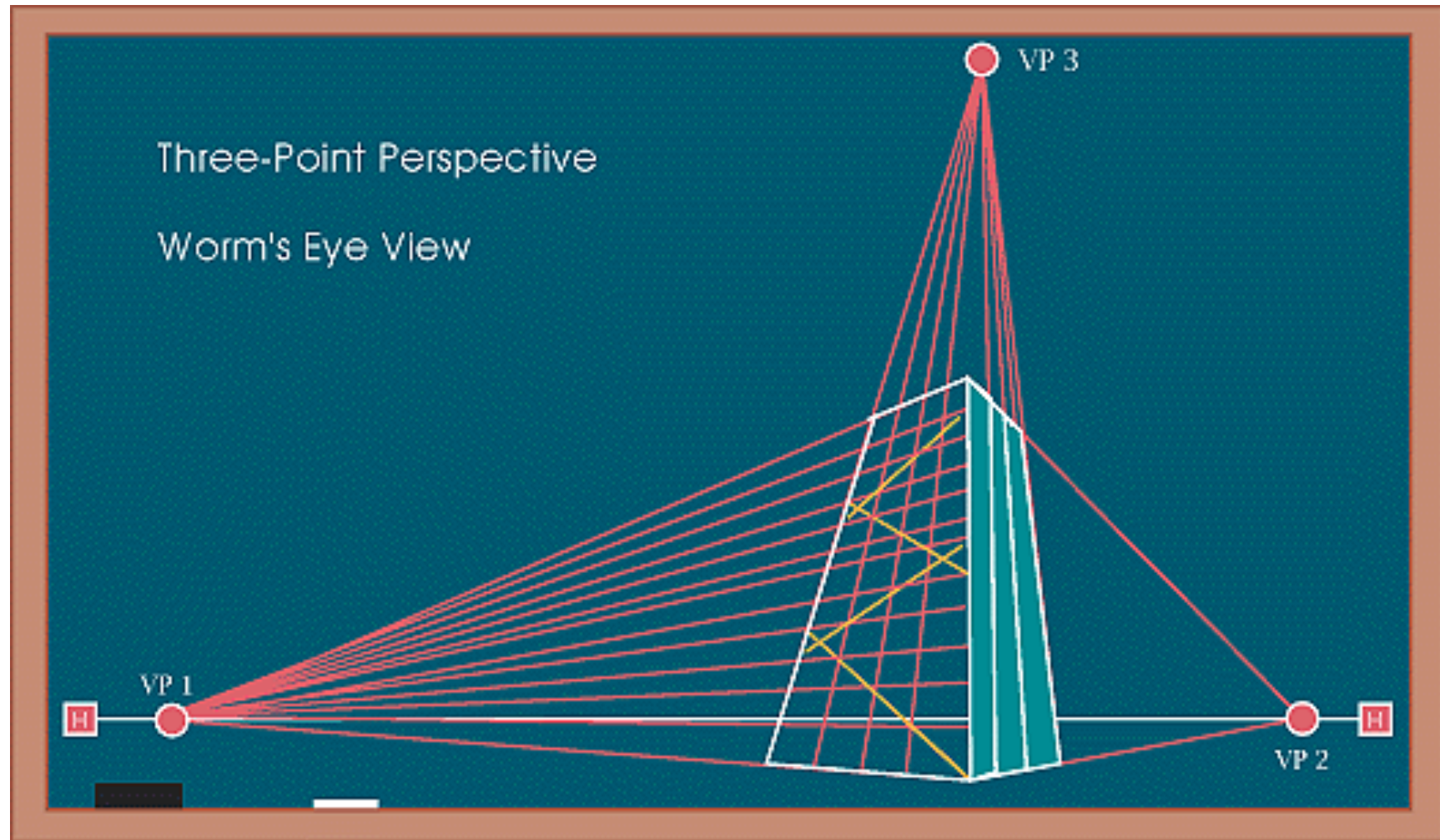


Plano de projeção

2-point perspective



3-point perspective

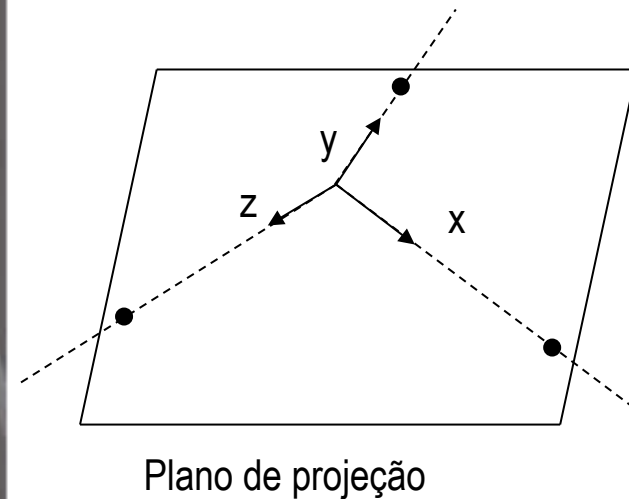


3-point perspective

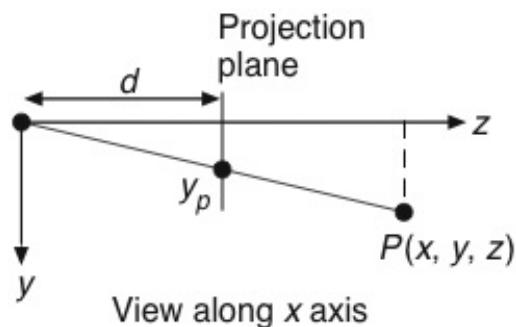
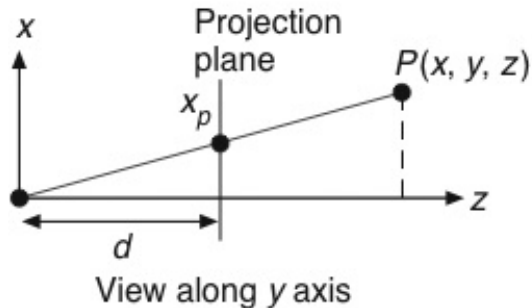
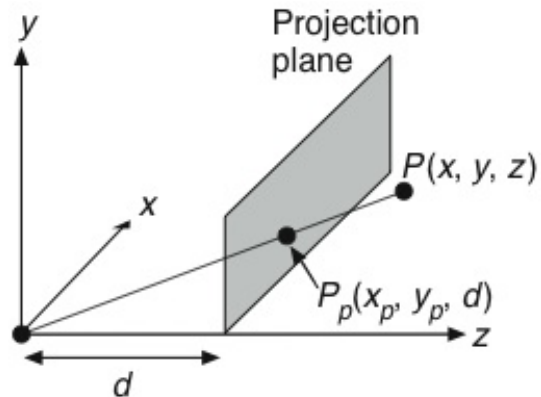
- City Night, 1926
 - Georgia O'Keefe



- Acrescenta pouco em relação a perspectiva com 2 pontos de fuga



Obtendo Coordenadas de Projeção

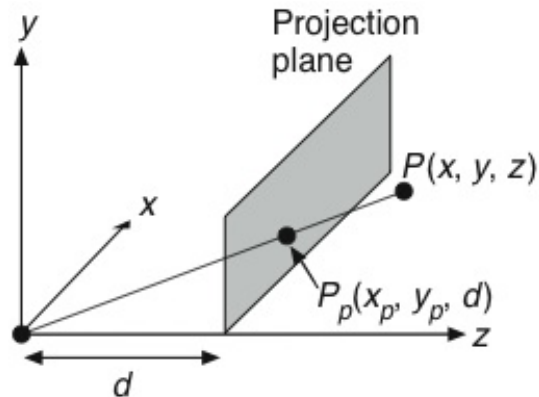


- Para projeções perspectivas?
- Assumindo que o plano de projeção encontra-se perpendicular ao eixo **w** do SRC e a uma distância **d**
- Por semelhança de triângulos temos:

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

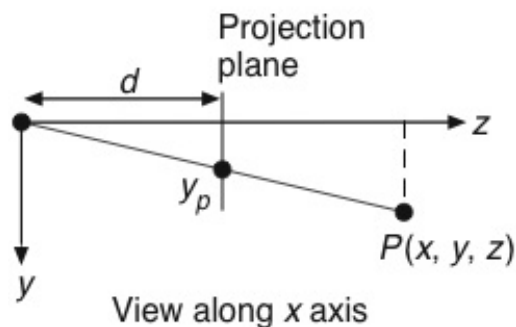
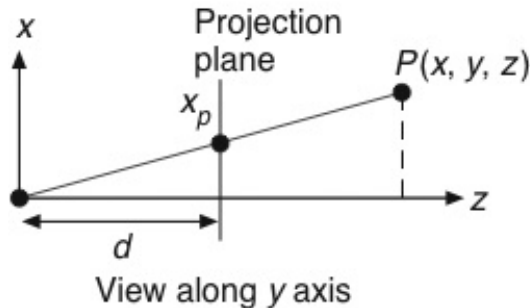
$$\frac{y_p}{d} = \frac{y}{z}$$

Obtendo Coordenadas de Projeção



$$x_p = \frac{d \cdot x}{z} = \frac{x}{z/d} \quad y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}$$

matricialmente:



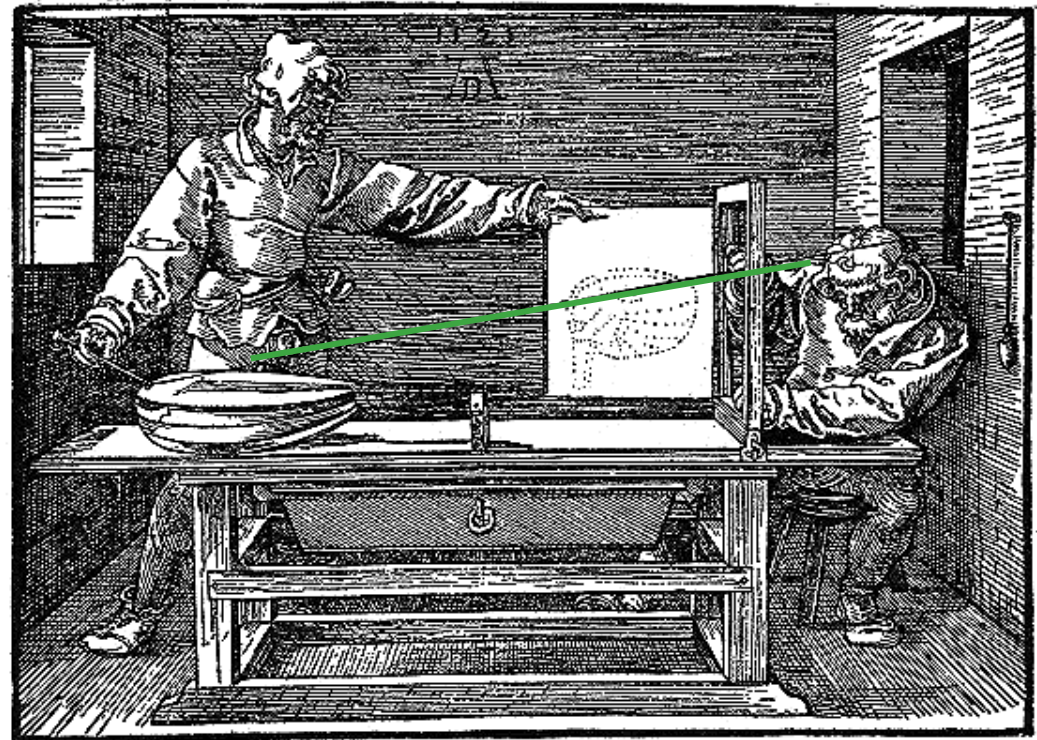
homogenize

Matriz de Projeção Perspectiva

$$\begin{pmatrix} x * d / z \\ y * d / z \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z / d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

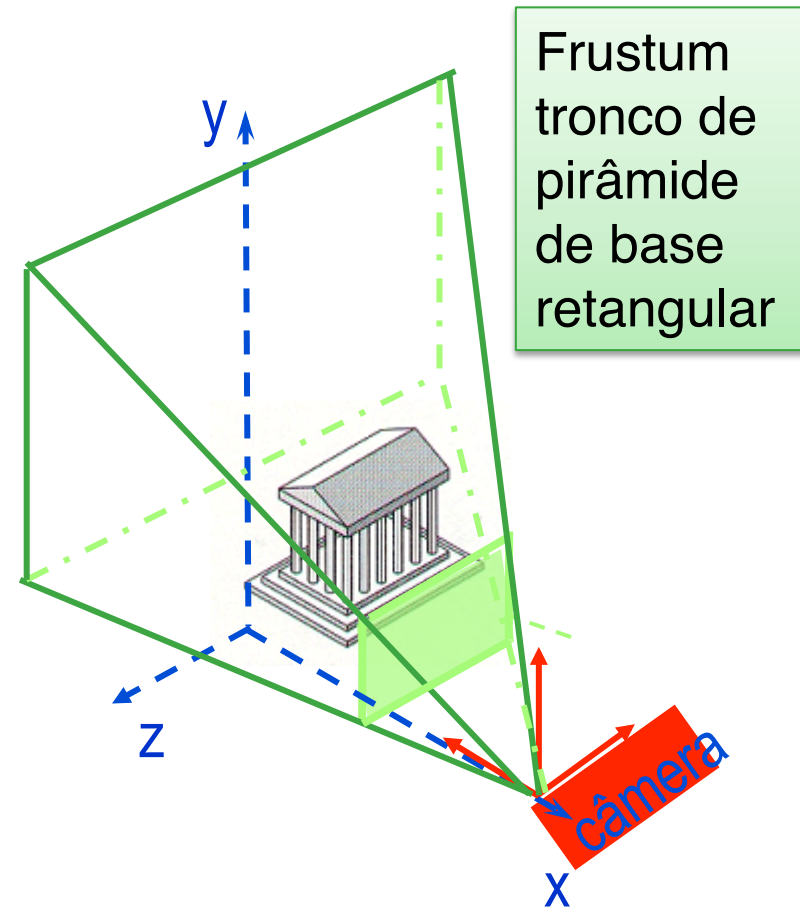
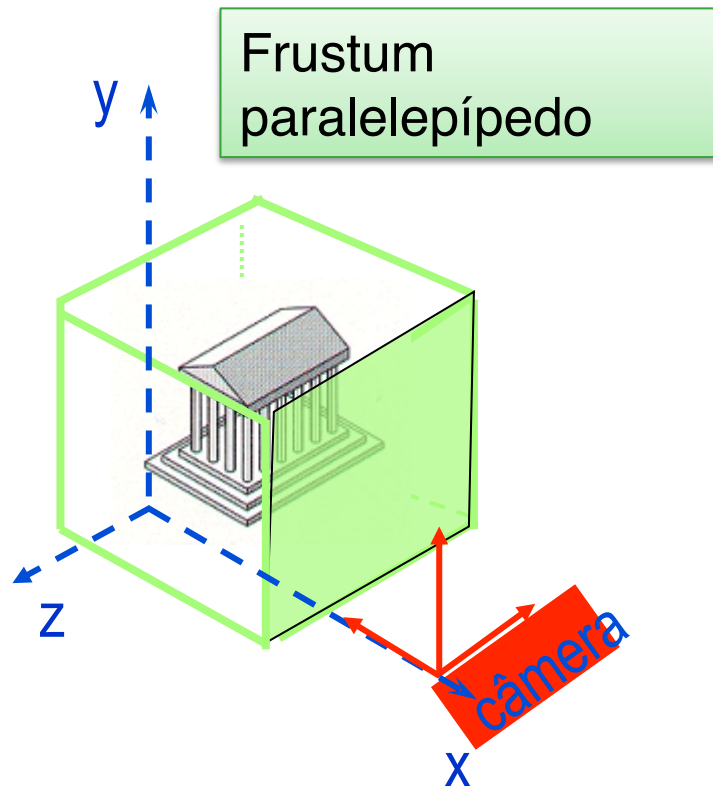
Volume de Visualização

- Precisamos definir uma área do espaço que será considerada
- Somente os objetos dentro do volume serão considerados
- Depende da **PROJEÇÃO**



Albrecht Durer doing perspective projections in 1525.

Volume de Visualização



- A definição do tipo de projeção, define também o tipo de Volume de Visualização (ou *Frustum*, em Inglês)

Transformações
Modelagem

Iluminação
(*Shading*)

Transformação
Câmera

Recorte

Projeção

Rasterização

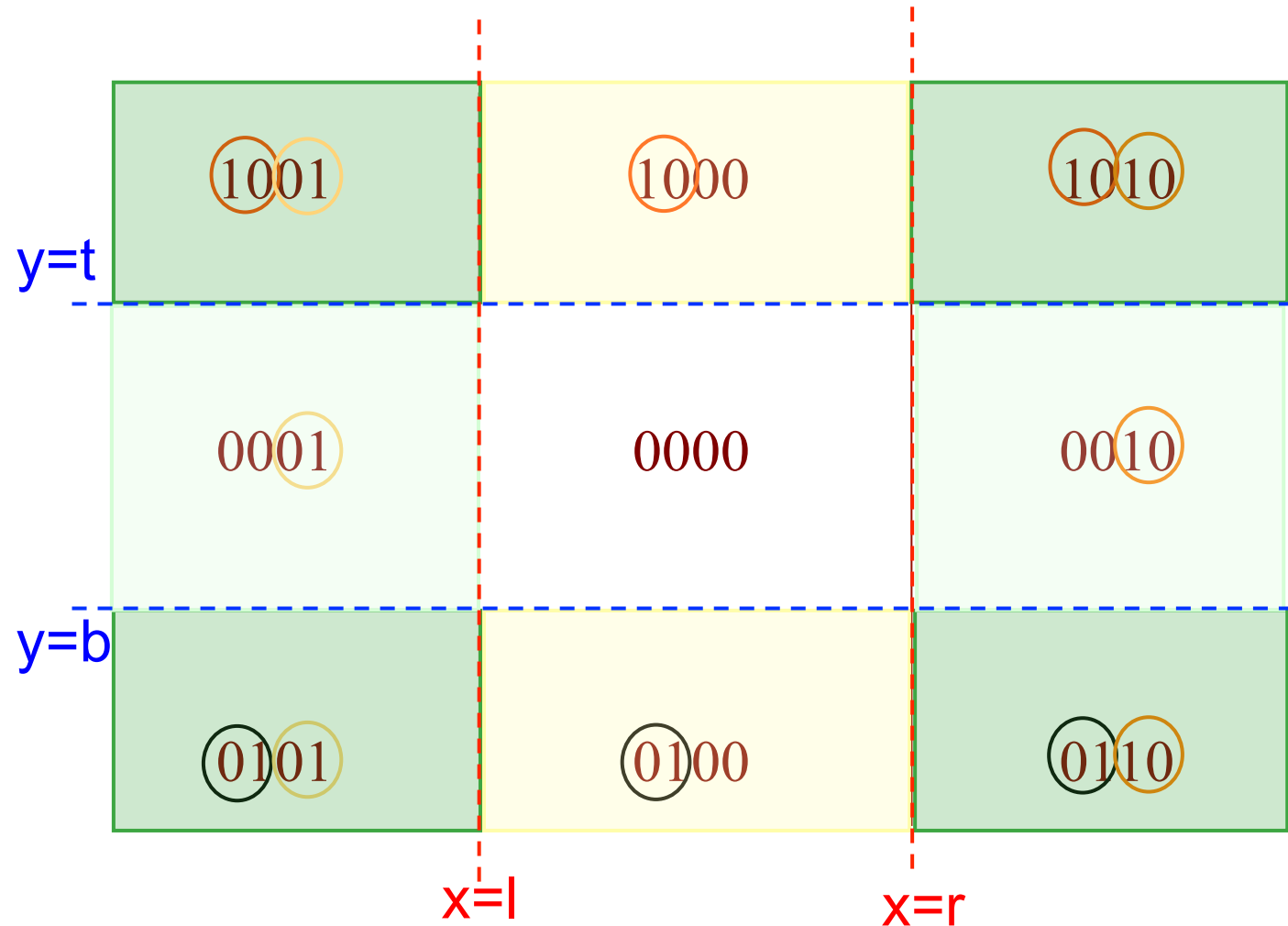
Visibilidade

E como fica o recorte 3D?

Adaptação e melhoramentos de uma aula sobre o mesmo
assunto (MIT - EECS 6.837 Durand and Cutler)

Algoritmo de Cohen-Sutherland 2D

- Maneira eficiente de determinar os diferentes casos
- Divide a região em 9 subespaços
- Atribuição de códigos aos espaços

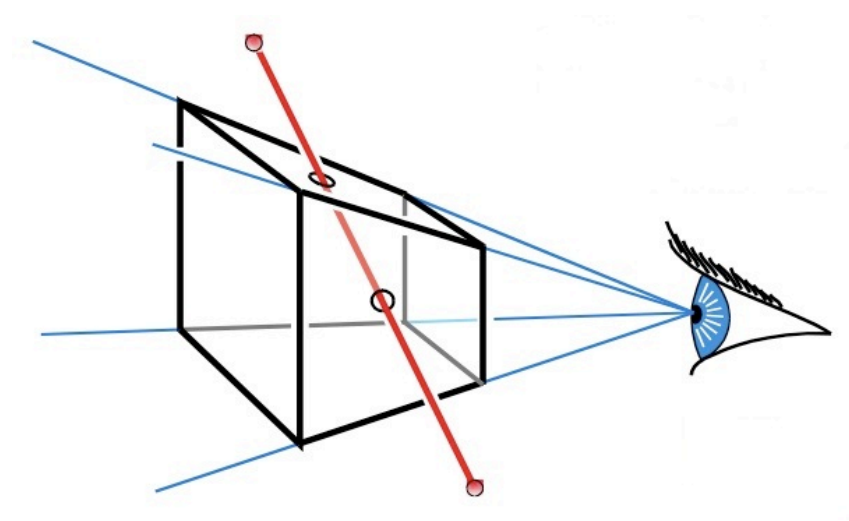


Cohen-Sutherland 2D - Outcodes

If $y > t \rightarrow$ seta primeiro bit em 1
If $y < b \rightarrow$ seta segundo bit em 1
If $x > r \rightarrow$ seta terceiro bit em 1
If $x < l \rightarrow$ seta quarto bit em 1

Recorte 3D

- Extensão de Cohen-Sutherland para 3D
- 6 outcodes ao invés de 4
 - bit 1: ponto está acima do volume
 - bit 2: ponto está abaixo do volume
 - bit 3: ponto está à direita do volume
 - bit 4: ponto está à esquerda do volume
 - bit 5: ponto está atrás do volume
 - bit 6: ponto está à frente do volume
- Os casos permanecem os mesmos



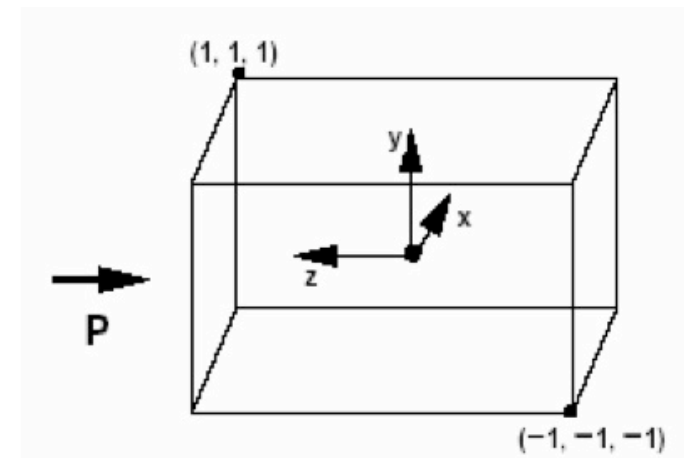
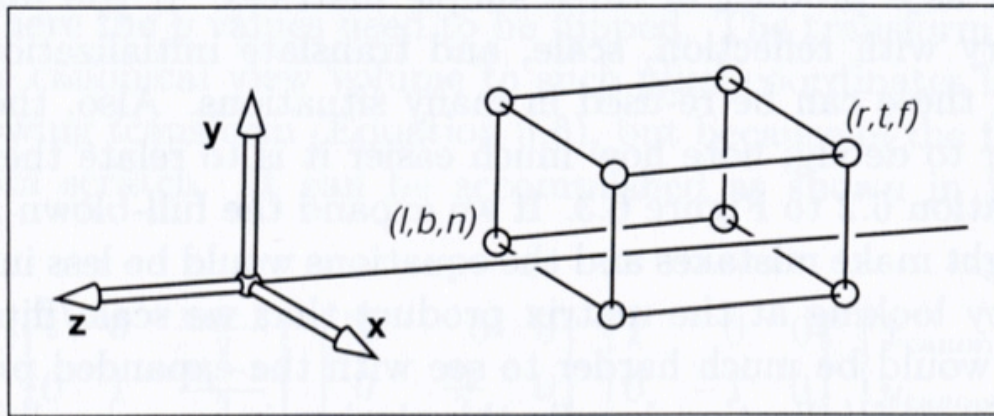
Coordenadas Normalizadas

MAS ANTES DE RECORTAR...

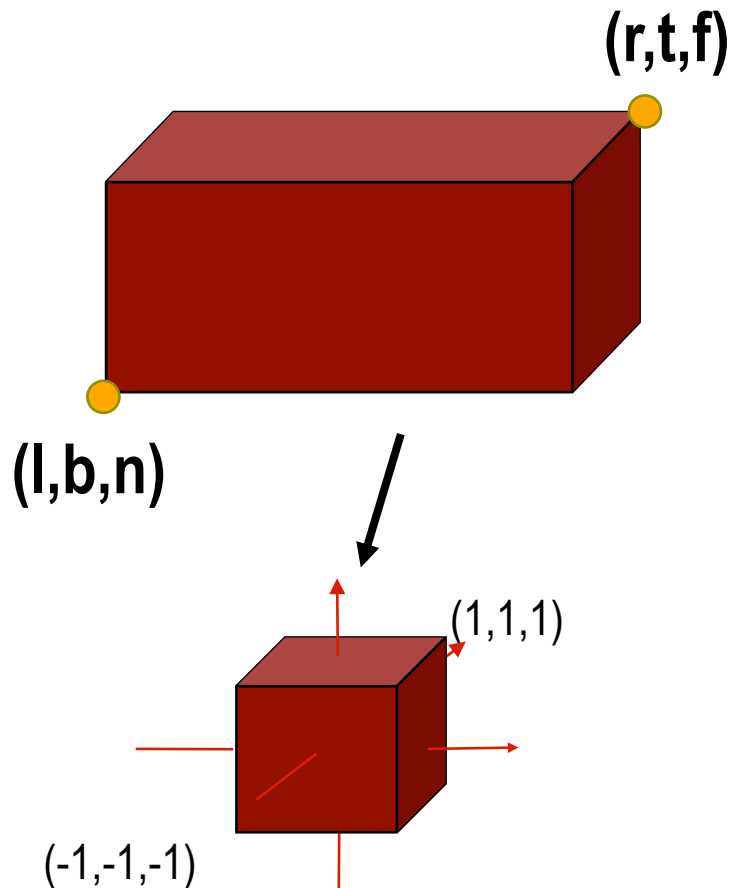
- Vantagens
 - Recorte mais eficiente para um volume retangular, alinhado com os eixos (*Volume-padrão*)
 - *Frustum* tem geometria arbitrária
- Desvantagens
 - Como todos os pontos são transformados antes do recorte, provavelmente estaremos transformando pontos que posteriormente serão eliminados pelo recorte



Coordenadas Normalizadas - Projeção Ortográfica

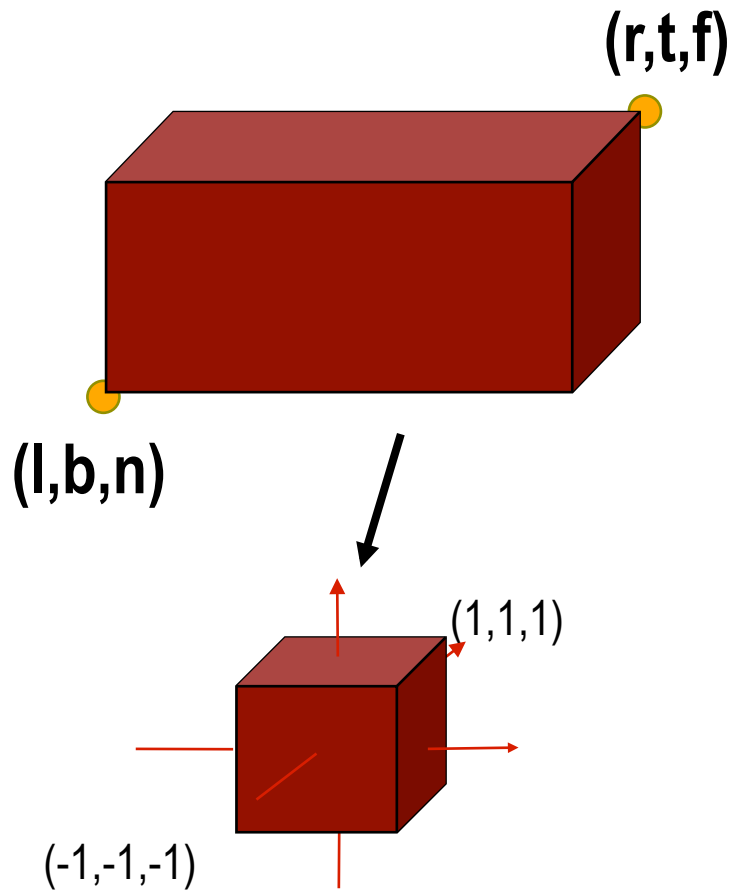


Transformação para o volume canônico de projeção ortográfica



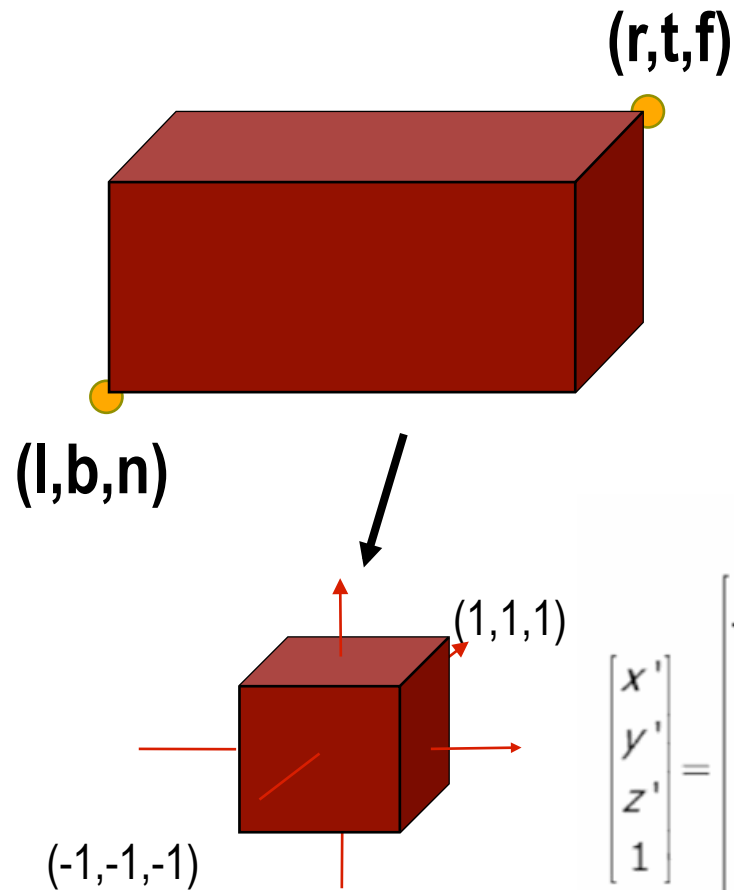
- Translação (T)
 - o ponto central do volume de visualização deve ser levado para a origem
 - $Dx = - (r + l) / (r - l)$
 - $Dy = - (t + b) / (t - b)$
 - $Dz = - (f + n) / (f - n)$
 - Isto é necessário porque, não obrigatoriamente r e l , b e t são centrados na origem!

Transformação para o volume canônico de projeção ortográfica



- Escala (S)
 - o volume canônico tem dimensões $2 \times 2 \times 2$
 - $S_x = 2 / (r - l)$
 - $S_y = 2 / (t - b)$
 - $S_z = 2 / (f - n)$
- Volume canônico
 - $P' = S.T.P$

Transformação para volume canônico de projeção ortográfica

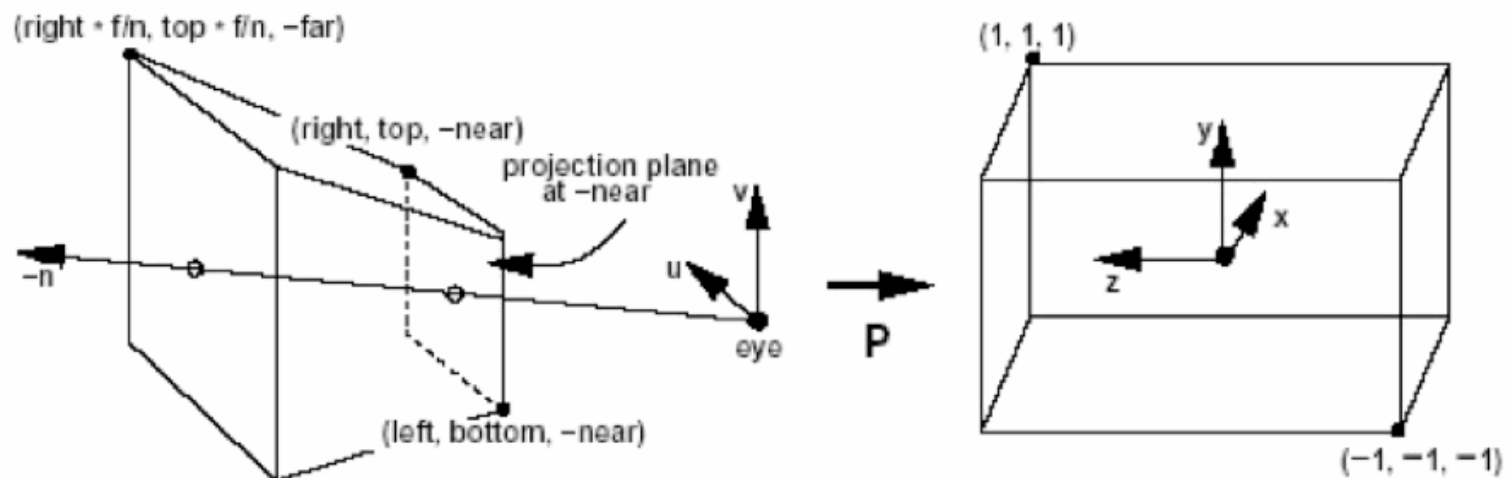


- Volume canônico
 - Origem deve ser (0,0,0)
 - Lado deve ter tamanho 2
 - $z_{min} = -near$
 - $z_{max} = -far$
- Matriz composta

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & \frac{-(\text{right} + \text{left})}{\text{right} - \text{left}} \\ 0 & \frac{2}{\text{bottom} - \text{top}} & 0 & \frac{-(\text{bottom} + \text{top})}{\text{bottom} - \text{top}} \\ 0 & 0 & \frac{2}{\text{far} - \text{near}} & \frac{-(\text{far} + \text{near})}{\text{far} - \text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Coordenadas Normalizadas

Proiecção Perspectiva



matriz de conversão ↷

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \frac{2 \cdot \text{near}}{\text{right} - \text{left}} & 0 & \frac{-(\text{right} + \text{left})}{\text{right} - \text{left}} & 0 \\ 0 & \frac{2 \cdot \text{near}}{\text{bottom} - \text{top}} & \frac{-(\text{bottom} + \text{top})}{\text{bottom} - \text{top}} & 0 \\ 0 & 0 & \frac{\text{far} + \text{near}}{\text{far} - \text{near}} & \frac{-2 \cdot \text{near} \cdot \text{far}}{\text{far} - \text{near}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Visualização 3D em OpenGL

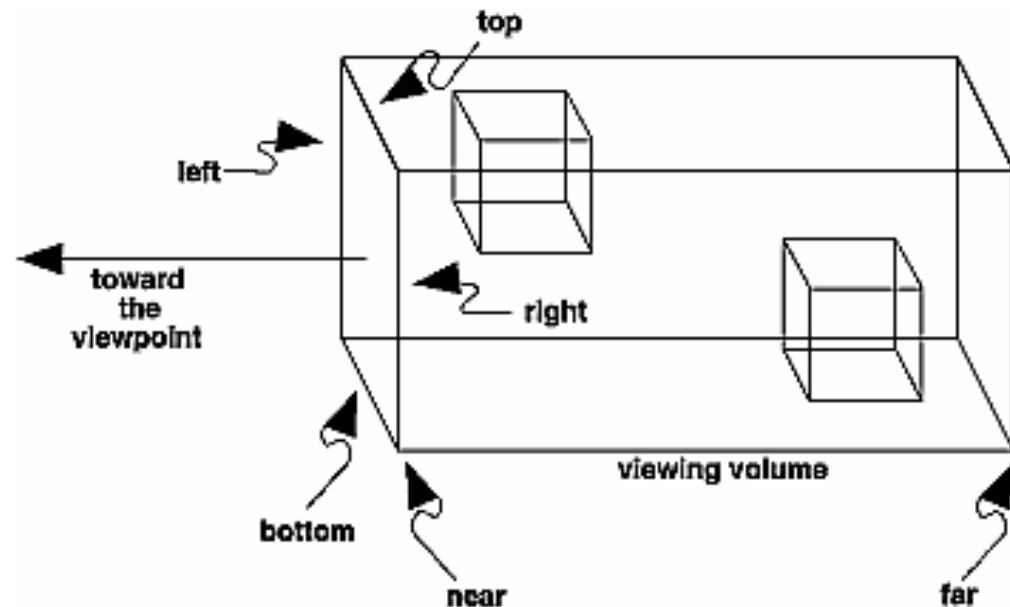
- Posição default da câmera
 - na origem (0,0,0)
 - orientada para o eixo -z
- Volume default para projeção ortográfica
 - 2 x 2 x 2
 - objetos atrás do observador também são projetados

Visualização (2D) em OpenGL

- Utilizando a GLU
 - `gluOrtho2D (left, right, bottom, top);`
 - todos os parâmetros são double
- Esta função chama
 - `glOrtho (left, right, bottom, top, near, far);`
 - `near = -1.0` e `far = 1.0`
 - especificando limites do volume, distâncias do observador no SRU
- Não se especifica posição do observador
 - O observador está em (0,0,0) olhando para a direção do eixo z negativo

Em OpenGL – Ortográfica 3D

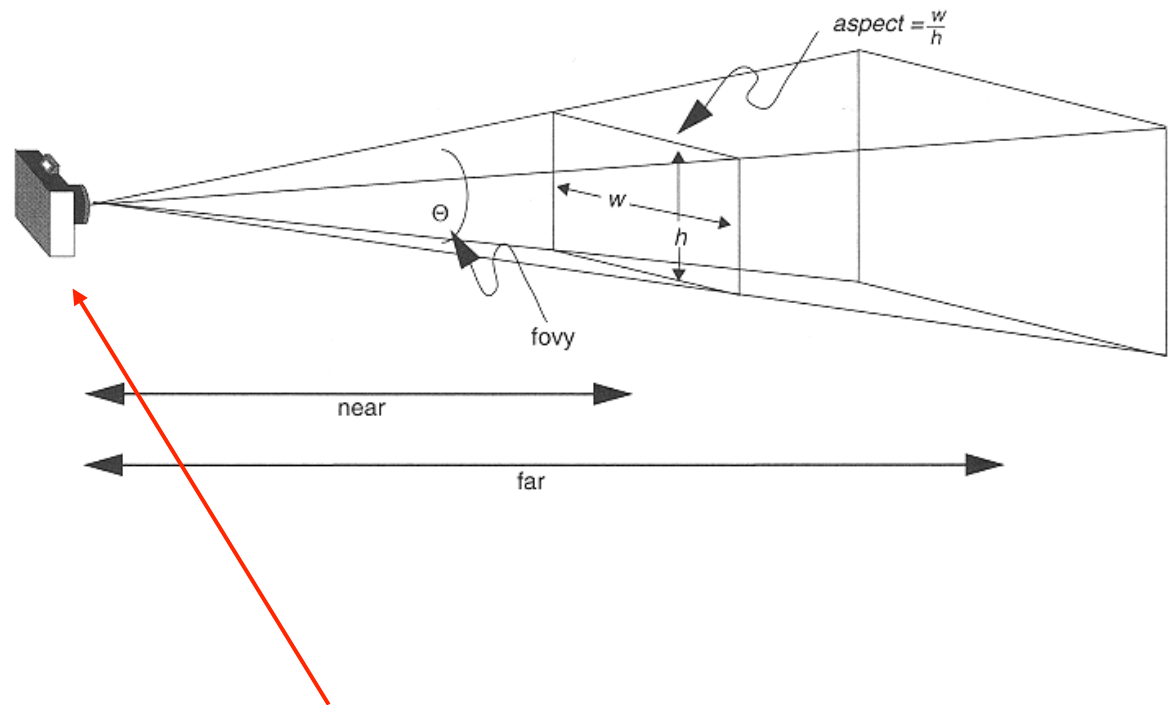
```
void glOrtho(  
    GLdouble left,  
    GLdouble right,  
    GLdouble bottom,  
    GLdouble top,  
    GLdouble near,  
    GLdouble far  
);
```



Em OpenGL - Perspectiva

Qual o *up vector* neste caso?

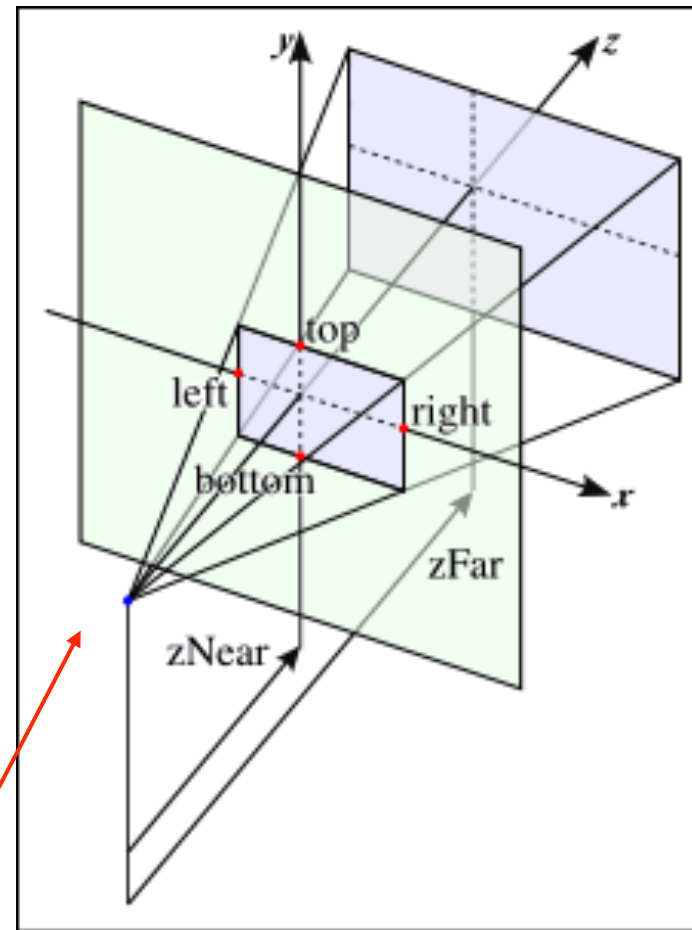
```
void gluPerspective(  
    GLdouble fovy,  
    GLdouble aspect,  
    GLdouble zNear,  
    GLdouble zFar  
) ;
```



A posição é definida anteriormente
com `glTranslate`

Em OpenGL - Perspectiva

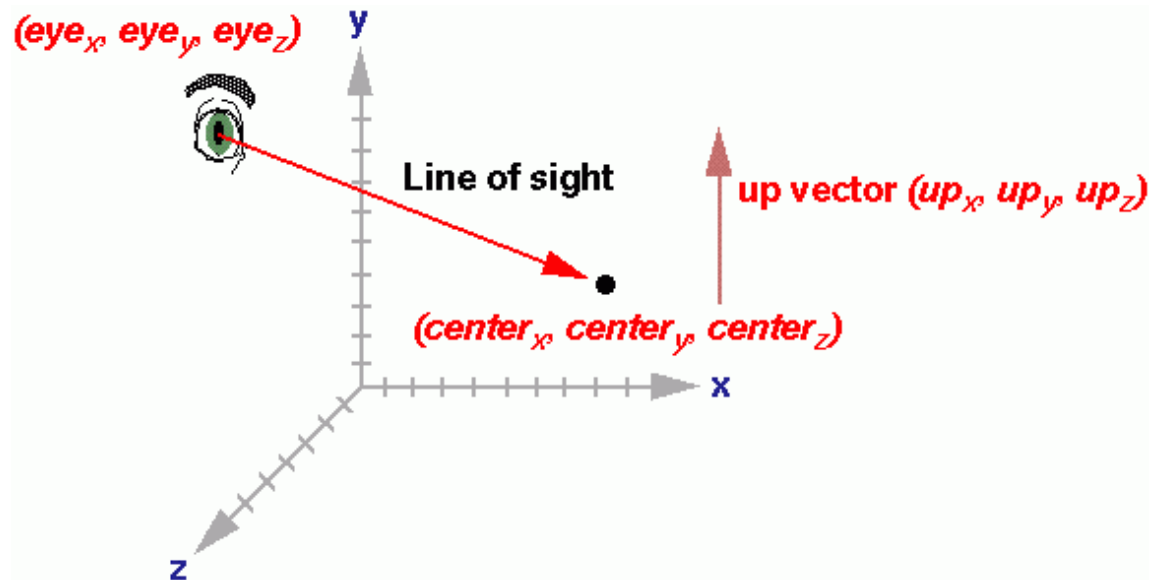
```
void glFrustum(  
    GLdouble left,  
    GLdouble right,  
    GLdouble bottom,  
    GLdouble top,  
    GLdouble near,  
    GLdouble far  
);
```



A posição é definida anteriormente
com `glTranslate`

Em OpenGL - Perspectiva

```
void gluLookAt(  
    GLdouble eyex,  
    GLdouble eyey,  
    GLdouble eyez,  
    GLdouble centerx,  
    GLdouble centery,  
    GLdouble centerz,  
    GLdouble upx,  
    GLdouble upy,  
    GLdouble upz,  
);
```



A posição é diretamente definida