



# Viewing Pipeline 2D/3D

---

Maria Cristina F. de Oliveira  
Rosane Minghim  
2008



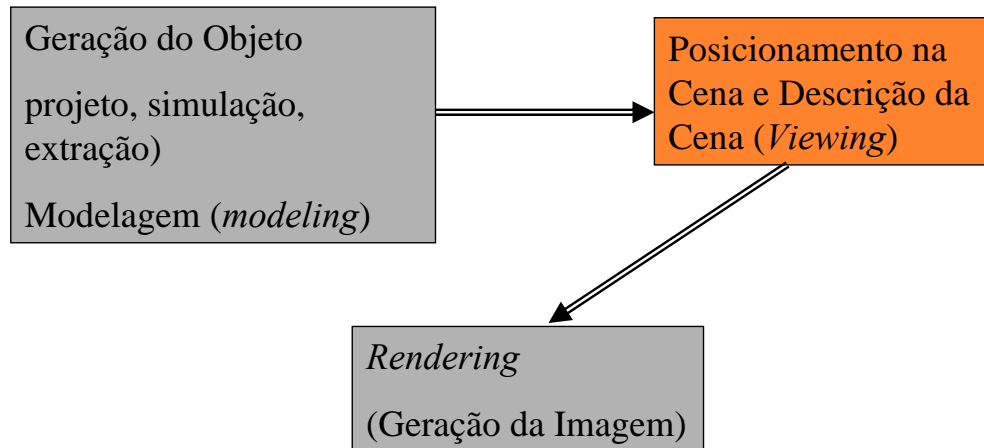
## Viewing Pipeline 2D

---

- Processo de determinar quais objetos da cena serão exibidos na tela, e como
- Transformação da cena, definida no sistema de coordenadas do mundo (WCS, ou SRU), para um sistema de coordenadas de observação, normalizado
  - VCS – *viewing coordinate system*, ou SRV – sistema de referência de observação (ou de visualização)
- E depois para o sistema de coordenadas do dispositivo
  - Visão geral do *pipeline*: v. Hearn & Baker, Fig. 6-2

## Viewing (onde estamos no pipeline)

### ■ Pipeline:



3

## Viewing Pipeline 2D

- No SRU, define "janela" ("*window*") de interesse
- Mapeia para janela normalizada no SRV ("*viewport*")
- Transformação "*window-viewport*": aplicada a todos os objetos contidos na janela de interesse
- Tudo o que está fora da "*window*" é descartado (*clipping*, ou recorte)

4

# Viewing Pipeline 2D

## ■ Dado

- *window* retangular, alinhada aos eixos principais do SRU, coord's  $w_{min}, w_{max}$
- *viewport* coord's  $v_{min}, v_{max}$

## ■ Buscamos a transformação que mapeia as coordenadas de um ponto $(w_p, w_p)_{SRU}$ no ponto $(v_p, v_p)_{SRV}$ . Duas abordagens:

- Sequência de transformações que 'alinha' a *window* com a *viewport* (translação + escala + translação inversa)
- Regra de três que mantém as proporções relativas dos objetos em ambas *window* e *viewport*

5

# OpenGL

```
glOrtho2D(GLfloat xwmin, GLfloat xwmax,  
          GLfloat ywmin, GLfloat ywmax);
```

Define janela de visualização em 2D

CIE =  $w_{min} = (xw_{min}, yw_{min})$

CSD =  $w_{max} = (xw_{max}, yw_{max})$

```
glViewport(GLint xvmin, GLint Width, GLint  
          yvmin, GLint Height);
```

CIE em  $xv_{min}, yv_{min}$

Width – largura viewport

Height – altura viewport

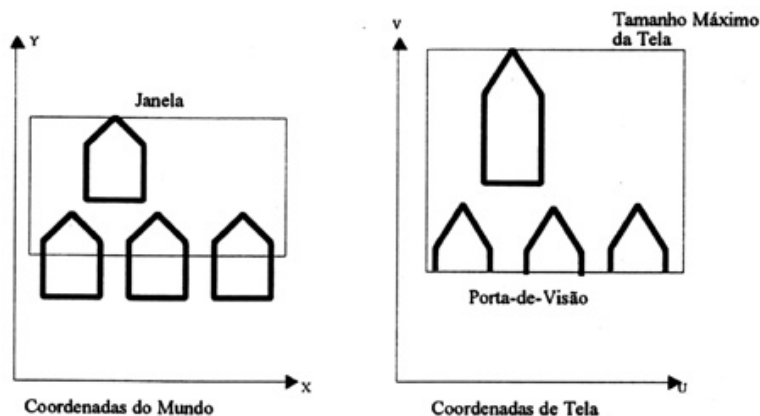
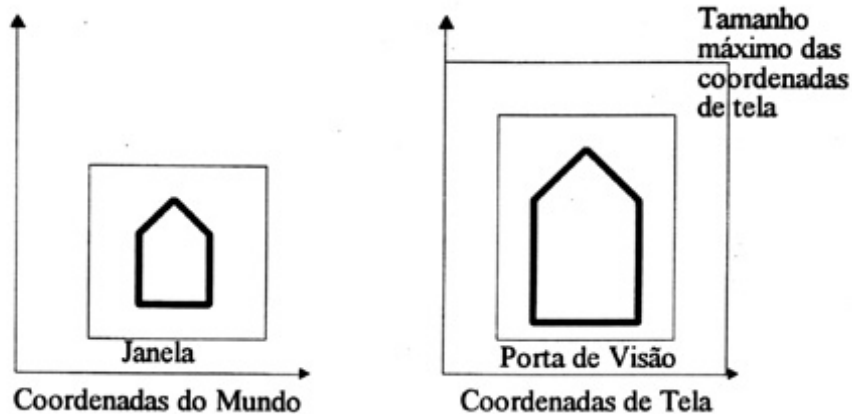
6

# Viewing Pipeline 2D

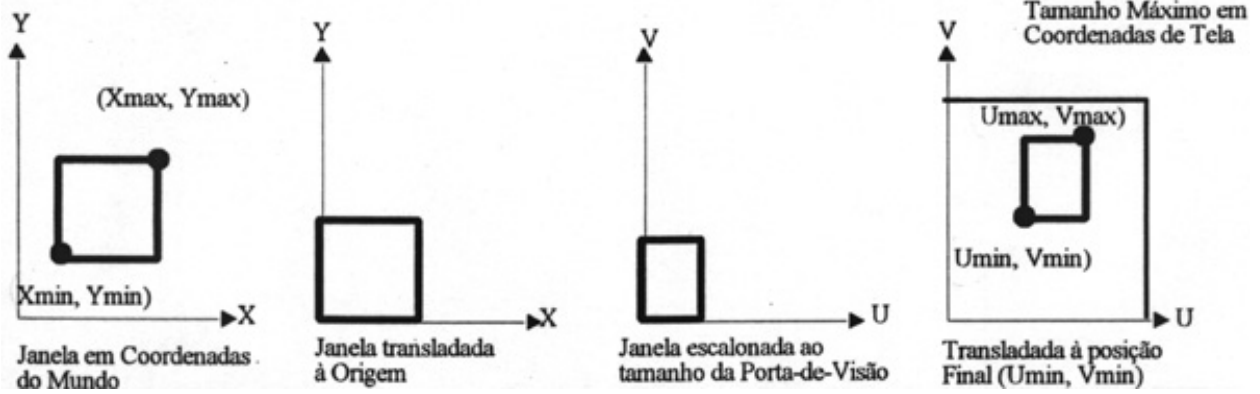
## ■ Observe que

- Mudando a posição da *viewport* pode-se exibir a mesma cena em posições diferentes do dispositivo
- Mudando o tamanho da *viewport* pode-se alterar o tamanho e as proporções dos objetos exibidos
  - *Zoom in/out*: obtido mapeando-se sucessivamente windows de tamanhos distintos (menores/maiores) em *viewport* de tamanho fixo
  - *Pan*: obtido movendo uma window de tamanho fixo na cena
- *Viewport* normalizada: quadrado de dimensão unitária, com canto inferior esquerdo (cie) na origem do SRV

7



8



9

## Bibliografia

- Hearn & Baker, Computer Graphics C Version, Cap. 6
- Apostila CG – Transformações 2D

10

# Viewing Pipeline 3D

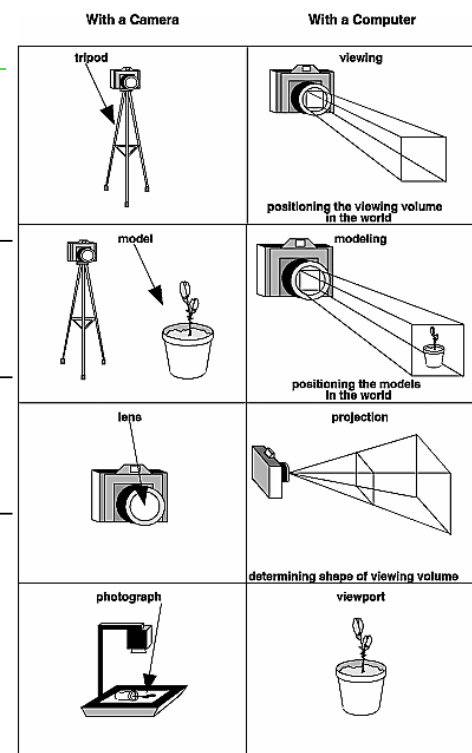
- No caso 3D, o *pipeline* requer:
  - A definição de um volume de interesse na cena 3D (SRU)
  - O mapeamento de seu conteúdo para o SRV (transformação de visualização)
  - A projeção do conteúdo do volume de interesse em um plano (transformação de projeção)
  - Mapeamento da janela resultante na *viewport* normalizada e depois para coordenadas do dispositivo

11

## Viewing Pipeline 3D: Analogia Câmera

Observação:	posiciona câmera	posiciona volume de observação
Cena:	posiciona modelo	posiciona modelo
Projeção:	escolhe lentes	escolhe formato vv
<i>Viewport</i> :	escolhe tamanho foto	escolhe porção da tela

fonte: curso CG Arizona State University,  
Dianne Hansford





# Viewing Pipeline 3D: Analogia Câmera

- Imaginamos um observador que vê a cena através das lentes de uma câmera virtual
  - “fotógrafo” pode definir a posição da câmera, sua orientação e ponto focal, abertura da lente...
  - câmera real obtém uma projeção de parte da cena em um plano de imagem 2D (o filme)
- Analogamente, a imagem obtida da cena sintética depende de vários parâmetros que determinam como esta é projetada para formar a imagem 2D no monitor
  - posição da câmera, orientação e ponto focal, tipo de projeção, posição dos “planos de recorte” (*clipping planes*), ...

13

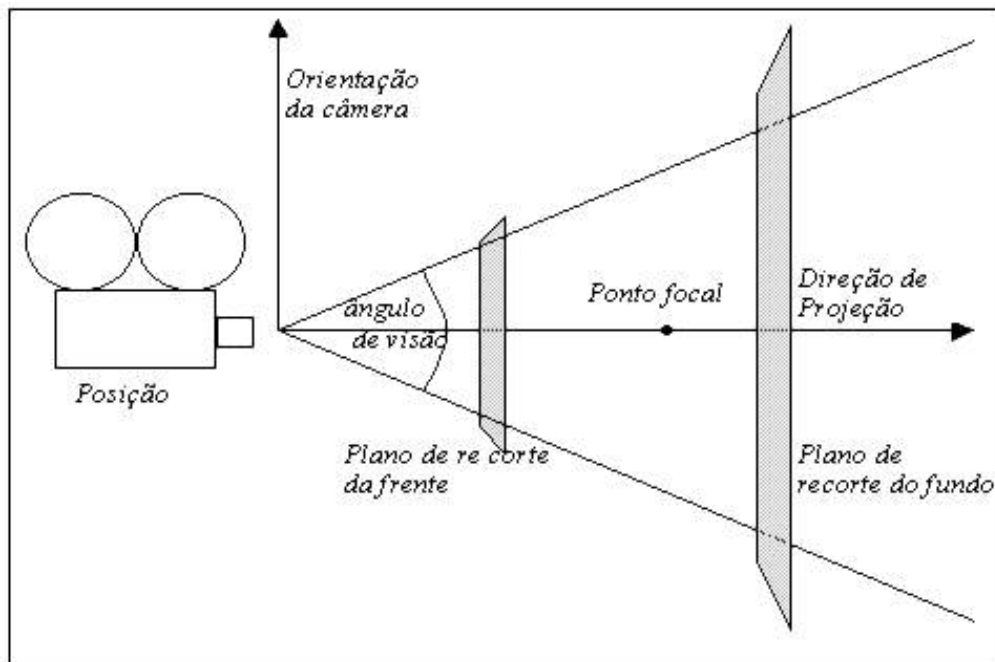


# Viewing Pipeline 3D: Analogia Câmera

- Três parâmetros definem completamente a câmera
  - Posição: aonde a câmera está
  - Ponto focal: para onde ela está apontando
  - Orientação: controlada pela posição, ponto focal, e um vetor denominado *view up*
- Outros parâmetros
  - Direção de projeção: vetor que vai da posição da câmera ao ponto focal
  - Plano da imagem: plano no qual a cena será projetada, contém o ponto focal e, tipicamente, é perpendicular ao vetor direção de projeção

14

# Viewing Pipeline 3D: Analogia Câmera



Fonte Figura: Schröder, The Visualization Toolkit, 1998

15

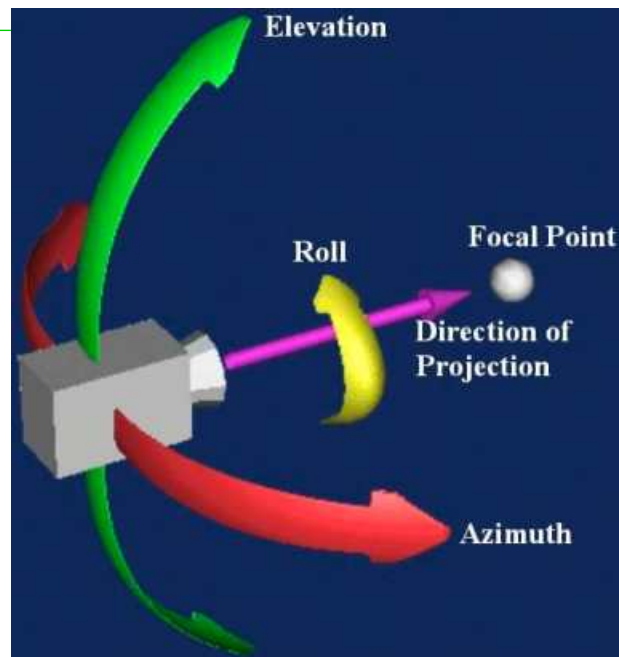
# Viewing Pipeline 3D: Analogia Câmera

- O método de projeção controla como os objetos da cena (atores) são mapeados no plano de imagem
  - Projeção ortográfica, ou paralela: processo de mapeamento assume a câmera no infinito, i.e., os raios de luz que atingem a câmera são paralelos ao vetor de projeção
  - Projeção perspectiva: os raios convergem para o ponto de observação, ou centro da projeção. Nesse caso, é necessário determinar o ângulo de visão da câmera
  - Os planos de recorte delimitam a região de interesse na cena
    - Anterior: elimina objetos muito próximos da câmera
    - Posterior: elimina objetos muito distantes

16



# Manipulação da Câmera



Fonte Figura: Schröder, The Visualization Toolkit, 1998

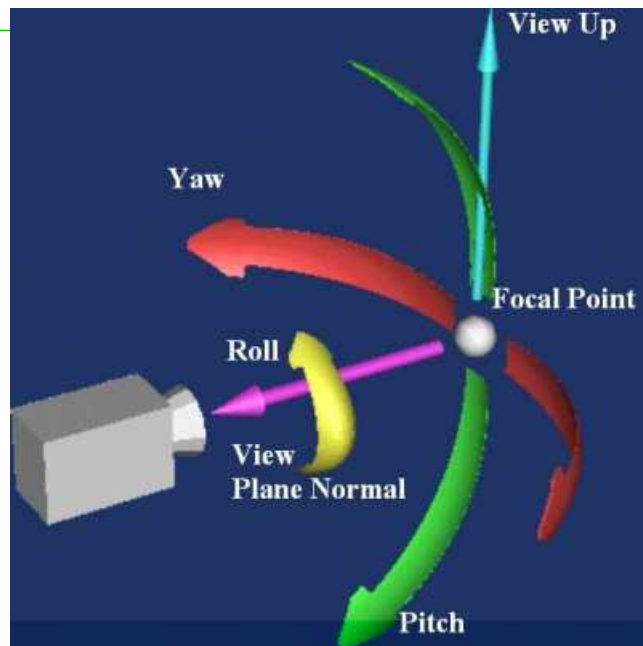
17

# Manipulação da Câmera

- *Azimuth*: rotaciona a posição da câmera ao redor do seu vetor *view up*, com centro no ponto focal
- *Elevation*: rotaciona a posição ao redor do vetor dado pelo produto vetorial entre os vetores *view up* e direção de projeção, com centro no ponto focal
- *Roll (Twist)*: rotaciona o vetor *view up* em torno do vetor normal ao plano de projeção

18

# Manipulação da Câmera



Fonte Figura: Schröder, The Visualization Toolkit, 1998

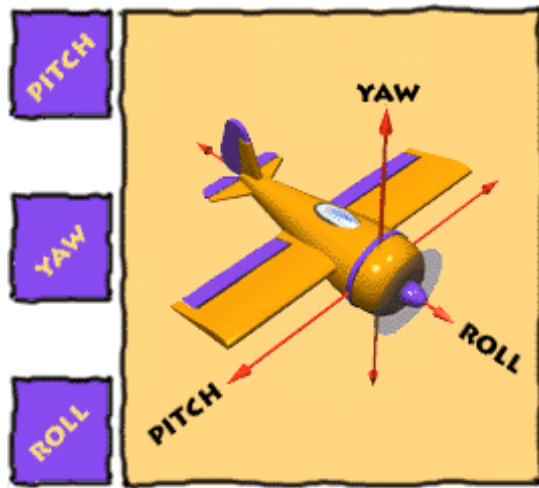
19

# Manipulação da Câmera

- *Yaw*: rotaciona o ponto focal em torno do vetor *view up*, com centro na posição da câmera
- *Pitch*: rotaciona o ponto focal ao redor do vetor dado pelo produto vetorial entre o vetor *view up* e o vetor direção de projeção, com centro na posição da câmera
- *Dolly (in, out)*: move a posição ao longo da direção de projeção (mais próximo ou mais distante do ponto focal)
- *Zoom (in, out)*: altera o ângulo de visão, de modo que uma região maior ou menor da cena fique potencialmente visível

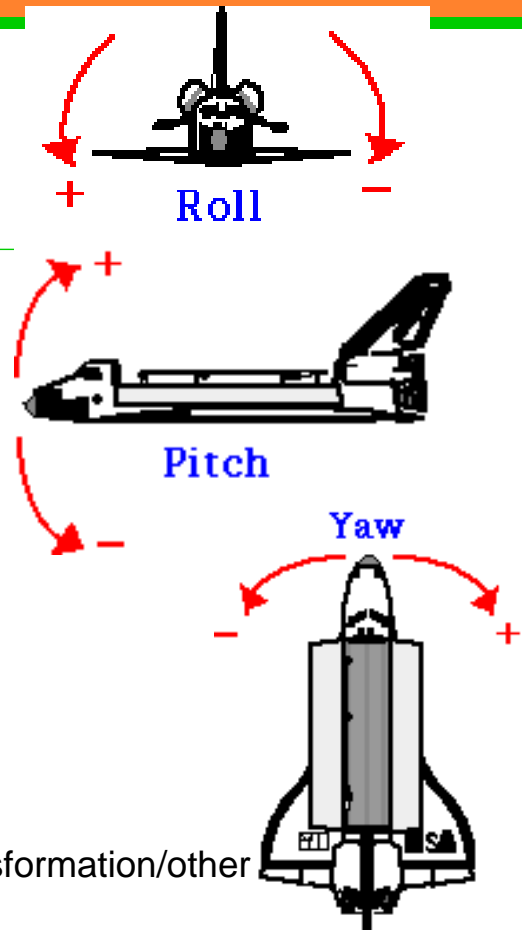
20

## Outra visão



Fonte:

<http://escience.anu.edu.au/lecture/cg/Transformation/otherRotation.en.html>



## Viewing Pipeline 3D

- V. Figura 12.2, Hearn & Baker
- Retomando: o *pipeline* requer a transformação da cena especificada no SRU para o SRV (ou VCS)
  - O SRV descreve a cena como vista pela câmera...
  - O primeiro passo nesse processo consiste em especificar o SRV. Como?
    - Necessário especificar origem e os três eixos de referência...

# Especificação do SRV

- Origem do sistema

- Posição da câmera (VRP: View Reference Point, ou PRO)

- Associados à câmera:

- Vetor direção de projeção (**N**), que dá a direção do ponto focal, e vetor view-up (**V**), que indica o 'lado de cima' da câmera (ambos devem ser perpendiculares entre si!)
- Plano de imagem, no qual a cena 3D será projetada, perpendicular ao vetor direção de projeção

- Eixos:

- eixo z associado ao vetor direção de projeção, eixo y associado ao vetor view-up, eixo x...

23

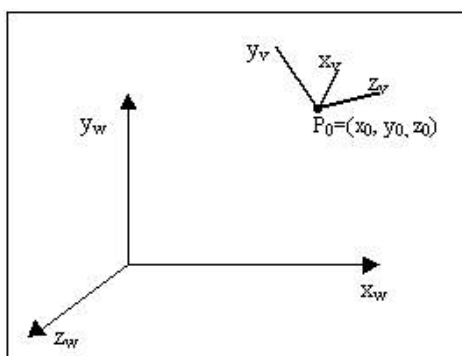
# Especificação do SRV

- Dados os vetores **N** e **V**, os vetores unitários podem ser calculados como indicado ao lado

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{N}|} = (n_1, n_2, n_3)$$

$$\mathbf{u} = \frac{\mathbf{V} \times \mathbf{N}}{|\mathbf{V} \times \mathbf{N}|} = (u_1, u_2, u_3)$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u} = (v_1, v_2, v_3)$$



24

## Conversão SRU->SRV

- Temos 2 espaços vetoriais (sist. coordenadas) em  $\mathbb{R}^3$ , definidos por duas bases ortonormais
  - SRU, espaço  $x_w, y_w, z_w$  (**i, j, k**)
  - SRV, espaço  $x_v, y_v, z_v$  (**u, v, n**)
- Queremos inicialmente coincidir as origens. Isso se faz trasladando a origem do  $S_v$  para a origem do  $S_w$  (lembrando transf. Entre sistemas de coordenadas). Sendo **P**<sub>0</sub> as coordenadas da origem de  $S_v$  em  $S_w$

25

## Conversão SRU->SRV

- Matriz de translação:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Queremos obter a matriz de rotação **R** que alinha os 2 sistemas, i.e., transforma de um espaço vetorial para o outro (p.ex. de  $x_v, y_v, z_v$  para  $x_w, y_w, z_w$ )

26

## Conversão SRU->SRV

- Matriz de rotação pode ser formada diretamente a partir de dois vetores, já que eles definem uma base ortonormal (e uma matriz ortogonal)

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

27

## Lembrando...

- Matriz  $\mathbf{R}$  é ortogonal:
  - Cada linha descreve um vetor unitário e os vetores são mutuamente ortogonais: definem uma base ortonormal
  - Analogamente, as colunas da matriz também definem uma base ortonormal
  - Na verdade, dada qualquer base ortonormal, a matriz cujas linhas (ou colunas) são formadas pelos seus vetores é ortogonal

28

# Lembrando...

- Conseqüentemente:

- R é normalizada

- a soma dos quadrados dos elementos em qqr linha/coluna é 1

- R é ortogonal

- produto escalar de qqr par de linhas ou colunas é zero
- inversa de R é igual à sua transposta

29

## Conversão SRU->SRV

- A matriz completa de transformação é

$$\mathbf{M}_{wc,vc} = \mathbf{R} \cdot \mathbf{T} = \begin{bmatrix} u_x & u_y & u_z & -u\mathbf{P}_0 \\ v_x & v_y & v_z & -v\mathbf{P}_0 \\ n_x & n_y & n_z & -n\mathbf{P}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Essa matriz, aplicada ao SRV alinha os eixos  $x_v, y_v, z_v$  ( $\mathbf{u}, \mathbf{v}, \mathbf{n}$ ) do SRV aos eixos  $x_w, y_w, z_w$  ( $\mathbf{i}, \mathbf{j}, \mathbf{k}$ ) do SRU
- A componente de translação alinha as origens

30

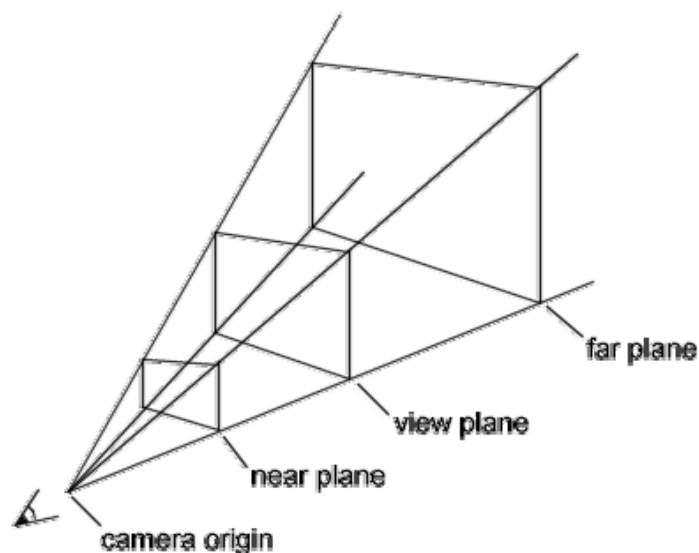
# Transformação de Projeção

- Tendo a cena descrita no SRV, o próximo passo no *pipeline* consiste em projetar o conteúdo do volume de visualização no plano de imagem
  - Volume de visualização: 'viewing frustum': define a região de interesse na cena
  - Antes da projeção é aplicado um processo de 'recorte' (*clipping*), em que as partes dos objetos que estão fora do VF são descartadas
  - Recorte 3D – em relação aos planos de recorte (*clipping planes*)

31

## *Viewing Frustum*

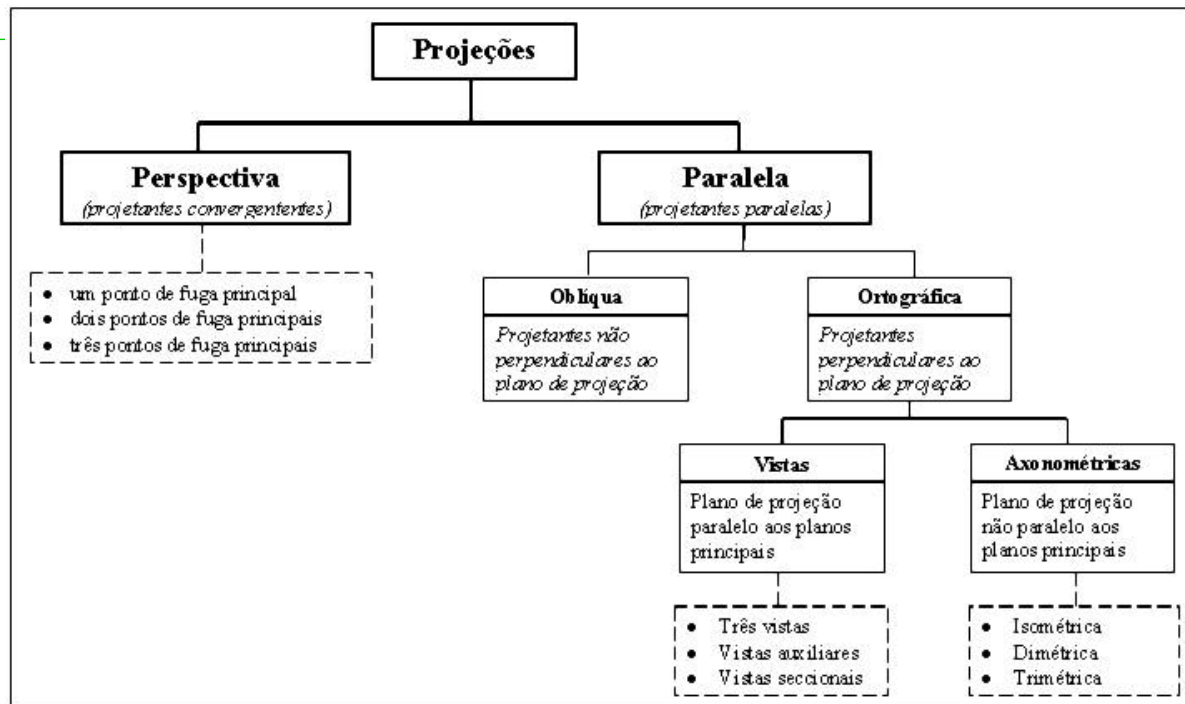
Volume de visualização, projeção perspectiva



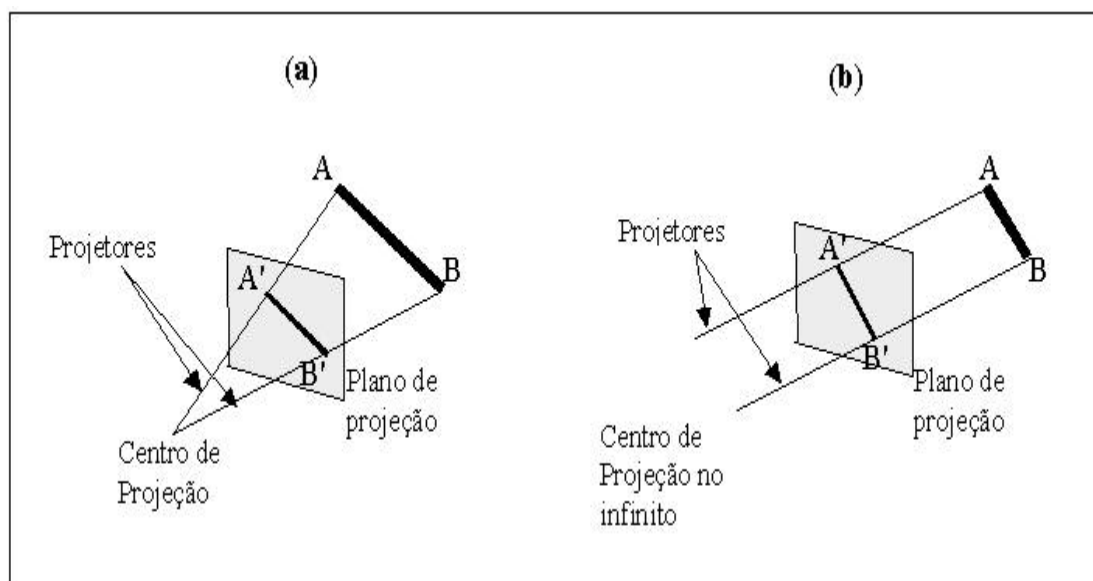
32



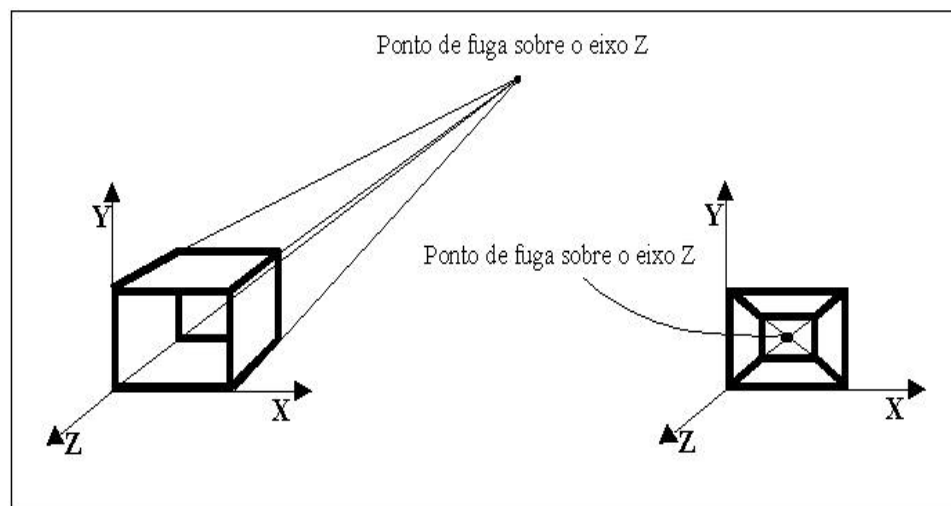
# Taxonomia das projeções



## Projeções paralela e perspectiva

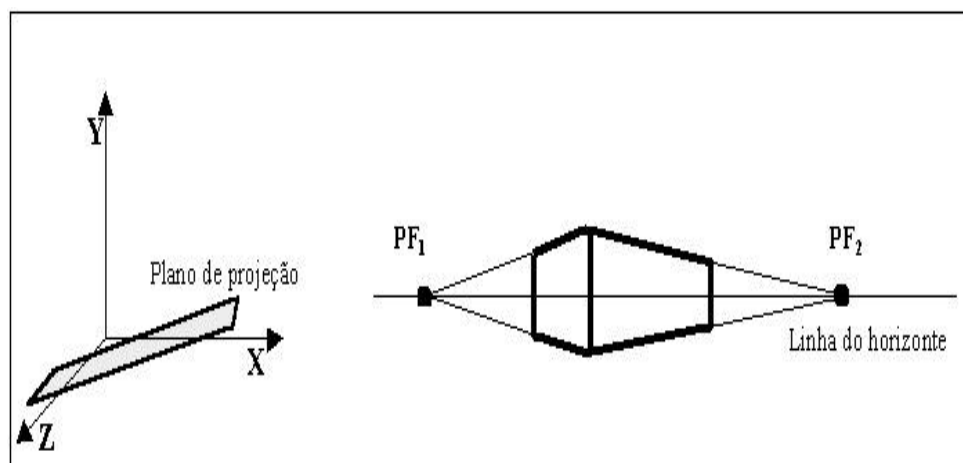


## Projeção perspectiva um ponto de fuga



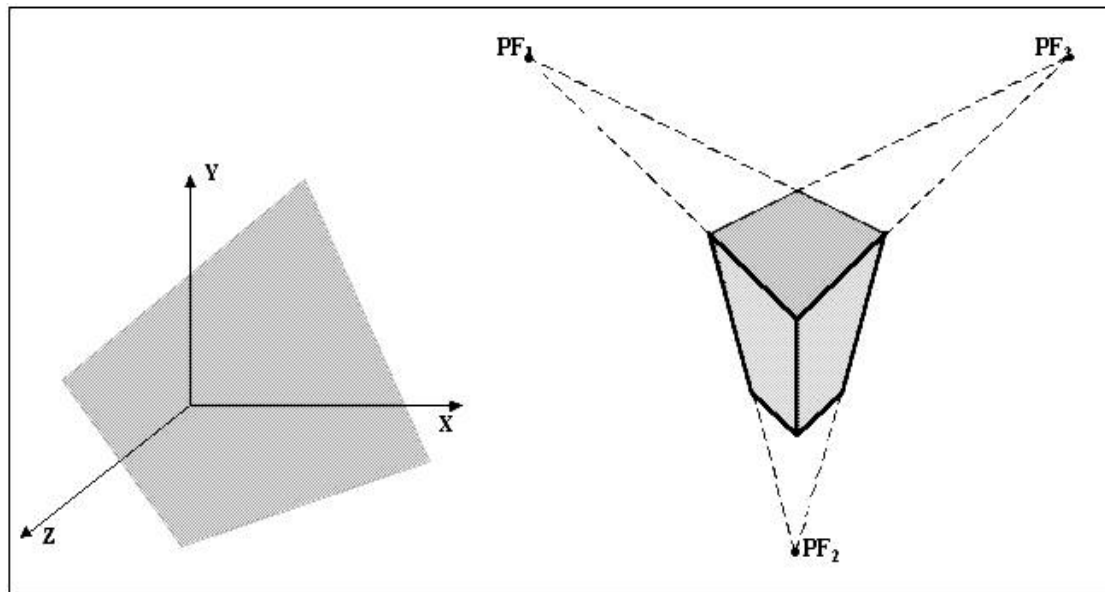
35

## Projeção perspectiva dois pontos de fuga



36

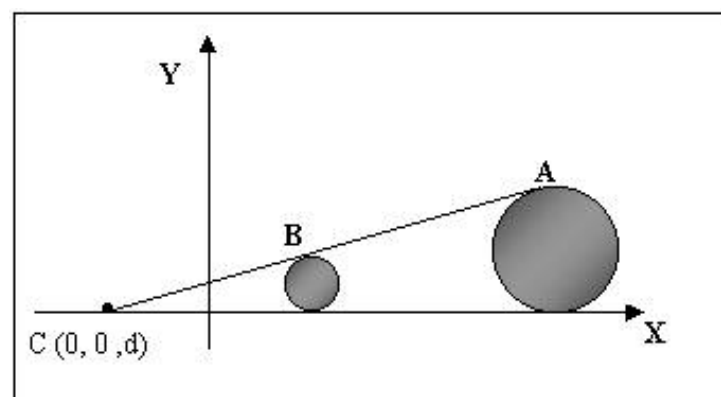
## Projeção perspectiva três pontos de fuga



37

## Características da Perspectiva

- Encurtamento perspectivo
  - Objetos ficam menores a medida que se distanciam do centro de projeção



38



## Características da Perspectiva

- Pontos de Fuga
  - Retas não paralelas ao plano de projeção parecem se interceptar em um ponto no horizonte
- Confusão Visual
  - Objetos situados atrás do centro de projeção são projetados de cima para baixo e de trás para a frente
- Distorção Topológica
  - Pontos contidos no plano que contém o centro de projeção e é paralelo ao plano de projeção são projetados no infinito

39



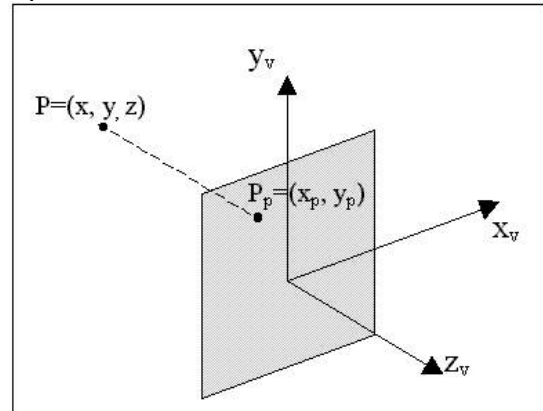
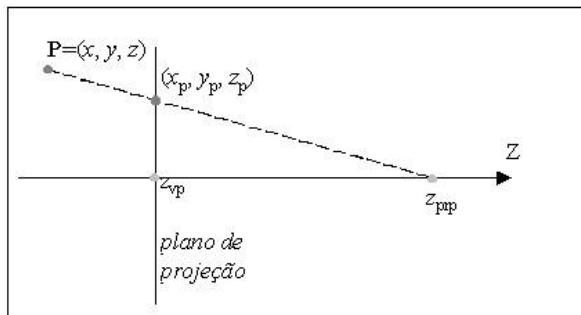
## Transformação de Projeção

- PRP: *Projection Reference Point*
  - o centro de projeção...
  - Alguns sistemas assumem que coincide com a posição da câmera (a origem do SRV)
- Problema
  - determinar as coordenadas  $(x_p, y_p, z_p)$  do ponto  $P = (x, y, z)$  projetado no plano de projeção (Figura)

40

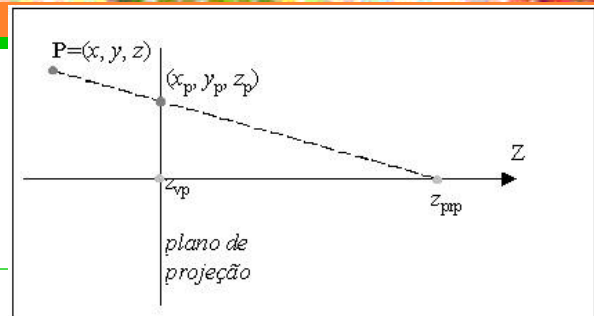
# Transformação de Projeção

- Suponha o centro de projeção posicionado em  $z_{prp}$ , um ponto no eixo  $z_v$ , e que o plano de projeção, normal ao eixo  $z_v$ , está posicionado em  $z_{vp}$ ,



41

## Transformação de Projeção



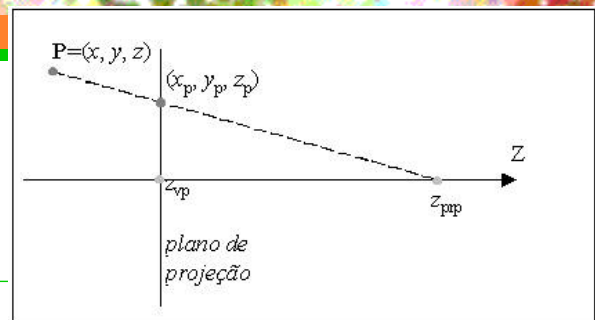
- Coordenadas projetadas  $(x', y', z')$  de um ponto  $(x, y, z)$  ao longo da linha de projeção

$$\begin{aligned} x' &= x - x * u \\ y' &= y - y * u \\ z' &= z - (z - z_{prp}) * u, u \in [0, 1] \end{aligned}$$

- Para  $u = 0$  estamos em  $P = (x, y, z)$ , para  $u = 1$  temos o centro de projeção  $(0, 0, z_{prp})$ .
- No plano de projeção:  $z' = z_{vp}$ . Podemos resolver  $z'$  para obter o valor de  $u$  nessa posição...

42

# Transformação de Projeção



- Valor de  $u$  no plano de projeção:

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$

- Substituir nas eqs. de  $x'$  e  $y'$
- $d_p$ : distância do plano de projeção ao centro de projeção, i.e.,

$$d_p = z_{vp} - z$$

43

# Transformação de Projeção

- Substituindo nas eqs. de  $x'$  e  $y'$

$$x_p = x \left( \frac{z_{prp} - z_{vp}}{z_{prp} - z} \right) = x \left( \frac{d_p}{z_{prp} - z} \right)$$

$$y_p = y \left( \frac{z_{prp} - z_{vp}}{z_{prp} - z} \right) = y \left( \frac{d_p}{z_{prp} - z} \right)$$

$$w_p = w \left( \frac{z_{prp} - z_{vp}}{z_{prp} - z} \right) = w \left( \frac{d_p}{z_{prp} - z} \right)$$

44

- Fator homogêneo:

$$h = \frac{z_{prp} - z}{d_p}$$

- Normalizar em relação a  $w = 1$  (dividir por  $h$ ) para obter as coordenadas projetadas no plano:

$$x_p = \frac{x_h}{h}, \quad y_p = \frac{y_h}{h}$$

45

## Transformação de Projeção

- Na forma matricial homogênea

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

46

# Transformação de Projeção

## ■ Observações:

- Valor original da coordenada  $z$  (no VCS) deve ser mantido para uso posterior por algoritmos de remoção de superfícies ocultas
- Centro de projeção não precisa necessariamente estar posicionado ao longo do eixo  $z_v$ . Eqs. podem ser generalizadas para considerar o centro um ponto qualquer
- Alguns pacotes gráficos (e nós tb.!) assumem  $z_{prp} = 0$ , i.e., centro de projeção coincide com origem do VCS
- Casos especiais: plano de projeção coincide com plano  $x_v y_v$ , i.e.,  $z_{vp} = 0$  (e  $d_p = z_{prp}$ )

47

# Projeções Paralelas

- No caso de projeções ortográficas, matrizes de transformação são triviais
- Ex. projeção em plano paralelo a  $x_v y_v$  (VCS):

$$M_{ortgraf} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

48





# Paralela vs. Perspectiva



## ■ Projeção perspectiva

- Tamanho varia inversamente com distância: aparência realística
- Distâncias e ângulos não são preservados
- Linhas paralelas não são preservadas


## ■ Projeção paralela

- Boa para medidas exatas
- Linhas paralelas são preservadas
- Ângulos não são preservados
- Aparência menos realística

49



# Bibliografia

- 
- Capítulo 6 da apostila
  - Cap. 12 Hearn & Baker
  - Cap. 2 Conci e Azevedo
  - <http://escience.anu.edu.au/lecture/cg/Transformation/index.en.html>
  - Curso CG da ACM (link na pág. GBDI)
  - ...

50