

# Visualização - 3D

Uéliton Freitas

Universidade Católica Dom Bosco - UCDB

*freitas.ueliton@gmail.com*

18 de setembro de 2014

# Sumário

- 1 Introdução
- 2 Viewing Pipeline 3D
- 3 Parâmetros de Coordenadas de Visão 3D
- 4 Transformação das Coordenadas do Mundo para as de Visão
- 5 Transformação de Projeções
  - Projeções Ortogonais
  - Projeções Perspectivas
- 6 Algoritmos de Recorte 3D

# Introdução

## Visualização

- As funções de visualização processam a descrição dos objetos por meio de vários procedimentos a fim de projetar a visão do objeto na superfície do dispositivo de saída.
- Mas há outras rotinas que são específicas do 3D.
  - Rotinas de projeção.
  - Identificação de partes visuais da cena.
  - Efeitos de Luz.

# Introdução

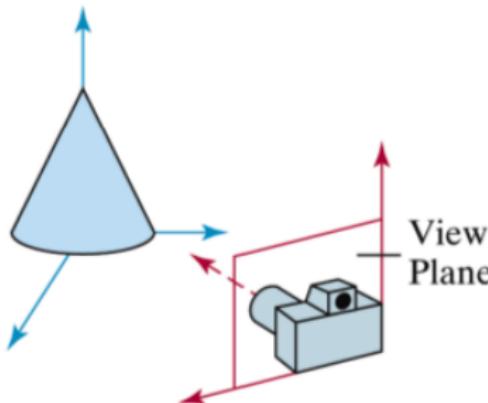
## Visualização

- As funções de visualização processam a descrição dos objetos por meio de vários procedimentos a fim de projetar a visão do objeto na superfície do dispositivo de saída.
- Alguns destes procedimentos são parecidos com o Pipeline de visualização 2D
  - Rotinas de recorte.
- Mas há outras rotinas que são específicas do 3D.
  - Rotinas de projeção.
  - Identificação de partes visuais da cena.
  - Efeitos de Luz.

# Introdução

## Visualização de uma Cena 3D

- Primeiramente, para obter uma visão de uma cena 3D descritas nas **coordenadas do mundo**, é necessário definir um sistema de referência para os parâmetros de visão (Câmera).
  - Definir a posição e orientação do **plano de visão** ou **plano de projeção**.



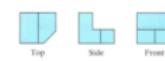
# Introdução

## Projeções

- É possível escolher diferentes métodos para projetar uma cena de visão.
  - **Projeção Paralela** - projeta objetos ao longo de linhas paralelas (Usado para desenhos arquitetônicos).
  - **Projeção de Perspectiva** - projeta os pontos de um objeto ao longo de caminhos convergentes produzindo cenas mais realísticas (objetos longe do observador ficam menores).



(a) Projeção Paralela



(a) Projeção Paralela



(b) Projeção Perspectiva

# Introdução

## Profundidade

- São raras as exceções em que a profundidade não é importante para composição de uma cena 3D.
- A profundidade explicita frente e trás do objeto.

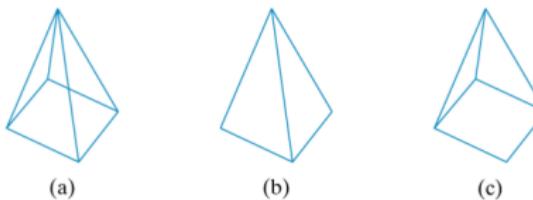
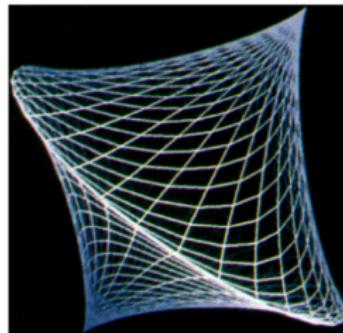


Figura : A Figura a possui problemas de visualização devido a falta de informação de profundidade

# Introdução

## Identificando Linhas e Superfícies Visíveis

- Uma forma simples de resolver o problema de profundidade é de objetos aramados (wire-frames) é variar o brilho das linhas.
  - Linhas mais próximas da posição de visão possuem maior brilho.



# Introdução

## Cenas Wire-Frame

- Há vários métodos para definir a profundidade de um objeto wire-frame.
  - Cores diferentes para linhas visíveis e não visíveis.
  - Mostrar linhas não visíveis como linhas pontilhadas.

## Cenas Realísticas

- As partes dos objetos que não são vistas são completamente eliminadas.
  - Os pixels da tela terão informações apenas das cores da superfície da frente.

# Introdução

## Rendering de Superfície

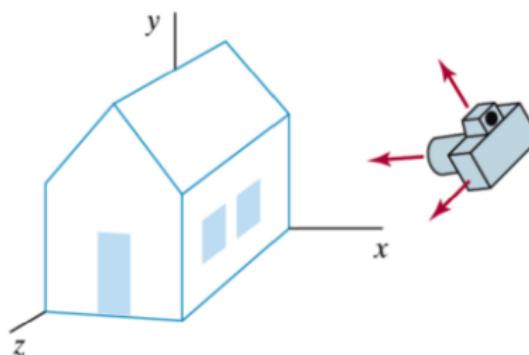
- Efeitos realísticos das cenas são objetos usando efeitos de iluminação
  - Define-se a luz do ambiente.
  - Define-se a cor e posição das fontes de luz.
- Também são definidos o material que os objetos são constituídos.
  - Transparentes, rugosos, opacos, reflexivos, etc.



# Introdução

## Criando uma Imagem

- O processo para criar uma imagem em computação gráfica em uma cena 3D é semelhante a tirar uma foto.
  - Define-se a posição de visão da câmera.
  - Define-se a orientação da câmera.
    - Como a câmera estará apontada a partir da posição de visão.
    - Como a câmera vai rotacionar definindo a posição **up**.



# Introdução

## Criando uma Imagem

- Há algumas semelhanças entre o *Pipeline da Viewing 2D e 3D*.
  - Uma **viewport 2D** é utilizada para posicionar a visão projetada no dispositivo de saída.
  - Uma janela de recorte 2D é utilizada para selecionar o que será visto na cena e mapeado para viewport.

# Introdução

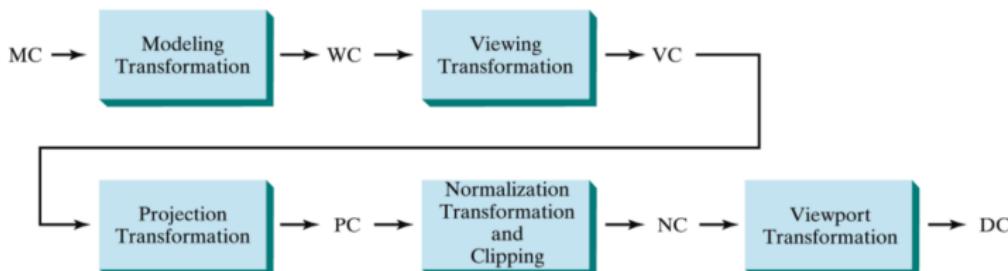
## Criando uma Imagem

- Há algumas semelhanças entre o *Pipeline da Viewing 2D e 3D*.
  - Uma **viewport 2D** é utilizada para posicionar a visão projetada no dispositivo de saída.
  - Uma janela de recorte 2D é utilizada para selecionar o que será visto na cena e mapeado para viewport.
- Porém há algumas diferenças
  - A janela de recorte é posicionada sobre um plano de visão.
  - A cena é recortada considerando um volume no espaço (volume de recorte) usando planos de recorte.

# Introdução

## Vieing Pipeline 3D

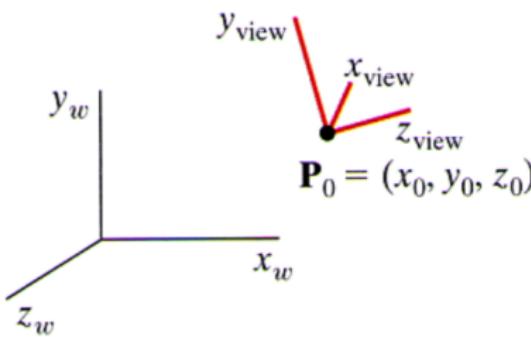
- O **Plano de Visão, Janela de Recorte, a Posição da Visão e os Planos de Recorte** são definidos nas **coordenadas do mundo**.



# Parâmetros de Coordenadas de Visão 3D

## Parâmetros de Coordenadas de Visão 3D

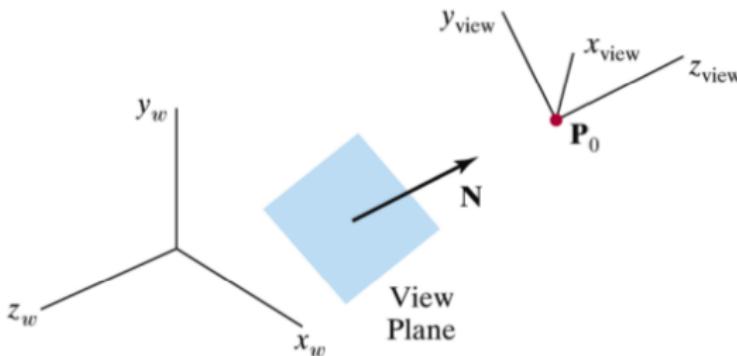
- Para estabelecer um parâmetro de coordenadas 3D é necessário:
  - A origem do sistema  $\mathbf{P}_0 = (x_0, y_0, z_0)$  chamado de **Ponto de Visão**(Onde o observador ou câmera se encontra).
  - O vetor **View up V**, que define a direção do  $y_{view}$ .
  - Uma segunda direção para um dos eixos de orientação. Normalmente o  $z_{view}$  que representa a orientação do eixo de visão.



# Parâmetros de Coordenadas de Visão 3D

## Vetor Paralelo ao Plano

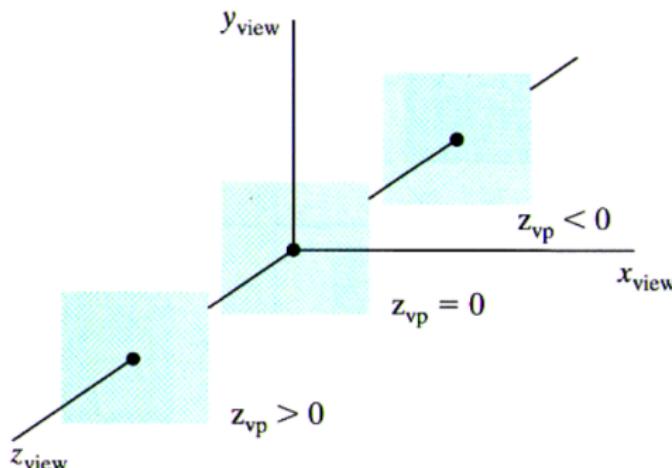
- Geralmente o plano de visão é dado pelo eixo  $z_{view}$  e o plano de projeção é formado a partir de um plano perpendicular ao eixo  $z$ .
  - Assim a orientação do plano de projeção e a direção projetiva do eixo  $z_{view}$  são dados por um vetor normal  $\mathbf{N}$  ao **Plano de Projeção**.



# Parâmetros de Coordenadas de Visão 3D

## Vetor Paralelo ao Plano

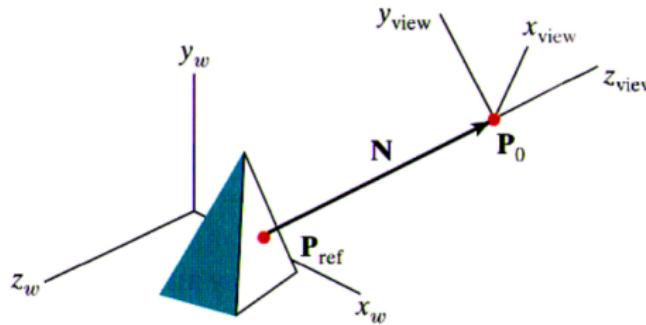
- Um escalar é utilizado para definir a movimentação do plano em um ponto  $z_{vp}$  ao longo do eixo  $z_{view}$ .
- O plano de visão é sempre paralelo ao plano  $x_{view} y_{view}$ .



# Parâmetros de Coordenadas de Visão 3D

## Vetor Paralelo ao Plano

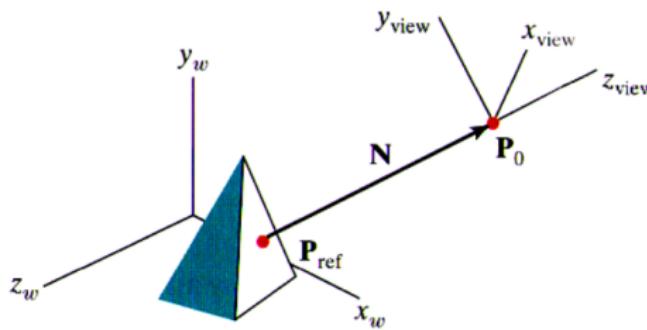
- O vetor normal  $\mathbf{N}$  pode ser obtido de várias formas:
  - A direção de  $\mathbf{N}$  pode ser obtida a partir de um ponto  $\mathbf{P}_{ref}$  até o ponto de origem  $\mathbf{P}_0$  (O inverso também é válido).
    - Neste caso o vetor  $\mathbf{N}$  é denominado **look-at point**, com a direção oposta a visão de  $\mathbf{N}$ .



# Vetor View Up

## Vetor Paralelo ao Plano

- Uma vez definida a posição e orientação do vetor  $\mathbf{N}$ , é necessário encontrar o vetor **view up**  $\mathbf{V}$  que mostra a direção do eixo  $y_{view}$ .
- Normalmente  $\mathbf{V}$  é definido selecionado uma posição relativa a origem do sistema de coordenadas do mundo.



# Parâmetros de Coordenadas de Visão 3D

## Vetor View Up

- O vetor **V** pode ser definido em qualquer direção exceto paralela ao vetor **N**.
  - Uma forma conveniente é definir o **V** como sendo paralelo ao eixo *y*.  $\mathbf{V} = (0, 1, 0)$ .
  - Se **V** não for perpendicular a **N**, rotinas de visão podem ser aplicadas para ajustar(projetar) o vetor de modo que seja.
  - A projeção do vetor **V** em **N** pode ser dada por:

$$\text{proj}_{V_{imp}, N} = V_{proj} = \frac{\mathbf{V} \cdot \mathbf{N}}{(\|\mathbf{N}\|)^2} \cdot \mathbf{N}$$

- E o vetor **V<sub>ajust</sub>** pode ser encontrado utilizando soma de vetores:

$$V_{ajust} = V_{imp} - V_{proj}$$

# Parâmetros de Coordenadas de Visão 3D

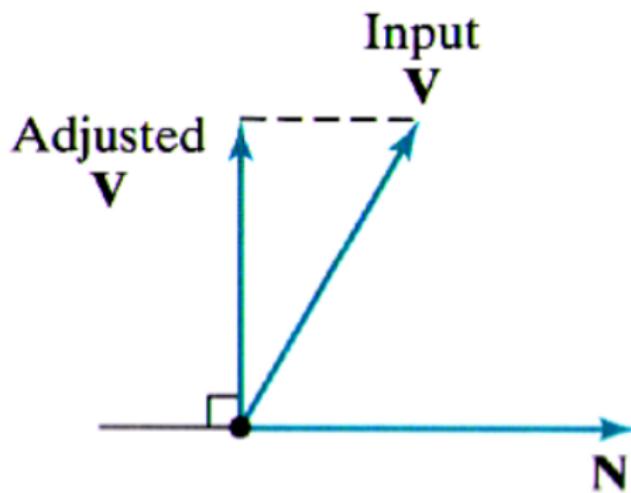


Figura : Ajuste do vetor **view up** para torna-lo perpendicular a **N**

# Sistema de Coordenadas de Visão $u \times n$

## Sistema de Coordenadas de Visão $u \times n$

- Com o vetor normal **N** definido, assim como o vetor view up **V**, basta apenas encontrar a direção positiva do eixo  $x_{view}$ .
  - A direção de  $x_{view}$  é representada por um vetor **U** obtido a partir do produto vetorial de **N** e **V**
  - O produto vetorial entre **N** e **U** também pode ser utilizado para ajustar **V** no eixo  $y_{view}$ .

# Sistema de Coordenadas de Visão $u \times n$

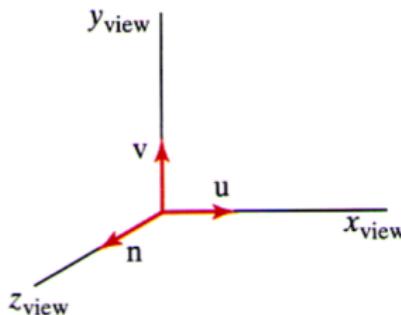
## Sistema de Coordenadas de Visão $u \times n$

- Para se obter o sistema de coordenadas **uvn** é necessário:

$$n = \frac{N}{|N|} = (n_x, n_y, n_z)$$

$$u = \frac{V \times n}{|V \times n|} = (u_x, u_y, u_z)$$

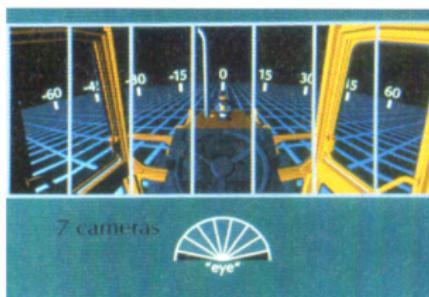
$$v = n \times u = (v_x, v_y, v_z)$$



# Gerando Efeitos de Visão 3D

## Gerando Efeitos de Visão 3D

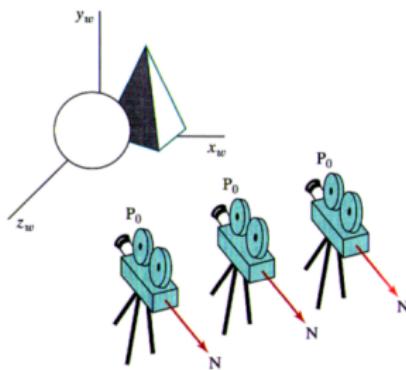
- Variando alguns parâmetros de visão é possível obter vários efeitos 3D.
  - De uma posição fixa é possível variar **N** de forma que seja possível observar objetos ao redor da posição.
  - Variar **N** para obter uma cena composta de múltiplas visões de uma posição fixa da câmera.
    - Lembrando que para cada posição de **N** é necessário ajustar os vetores dos eixos restantes mantendo a regra da mão direita.



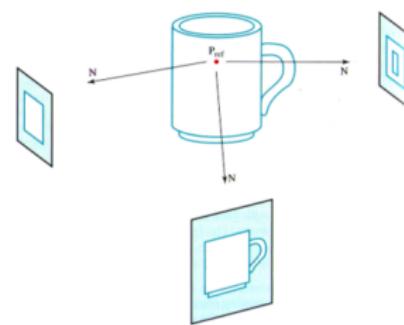
# Gerando Efeitos de Visão 3D

## Gerando Efeitos de Visão 3D

- Efeitos de movimento da câmera (pam) podem ser obtidos fixando  $\mathbf{N}$  e modificando a posição da câmera.
- Para mostrar diferentes visões de um objeto podemos mover o ponto de visão ao redor do objeto.



(a) Efeito de pam



(b) Visões diferentes

# Transformação das Coordenadas do Mundo para as de Visão

## Transformação das Coordenadas do Mundo para as de Visão

- No **Viewing Pipeline 3D** o próximo passo a ser executado após a cena ser montada é transferir as coordenadas dos objetos para para o sistema de coordenadas de visão.
  - Há uma sobreposição do sistema de coordenadas de visão sobre o sistema de coordenadas do mundo.
- Esta conversão é dada por:
  - ① Translada-se a origem do sistema de visão para a origem do sistema de coordenadas do mundo.
  - ② Rotaciona os eixos  $x_{view}$ ,  $y_{view}$  e  $z_{view}$  para deixa-los alinhados com os eixos  $x_{wc}$ ,  $y_{wc}$  e  $z_{wc}$ .

# Transformação das Coordenadas do Mundo para as de Visão

## Transformação das Coordenadas do Mundo para as de Visão

- Se a origem do sistema de visão for em  $P_0(x_0, y_0, z_0)$  a matriz de translação  $\mathbf{T}$  será:

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

# Transformação das Coordenadas do Mundo para as de Visão

## Transformação das Coordenadas do Mundo para as de Visão

- A Matriz de Rotação poder ser obtida por meio dos vetores  $\mathbf{u} = (u_x, u_y, u_z)$ ,  $\mathbf{v} = (v_x, v_y, v_z)$  e  $\mathbf{n} = (n_x, n_y, n_z)$ :

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

# Transformação das Coordenadas do Mundo para as de Visão

## Transformação das Coordenadas do Mundo para as de Visão

- Portanto a matriz de transformação é:

$$\mathbf{M}_{WC,VC} = R \cdot T = \begin{bmatrix} u_x & u_y & u_z & -u \cdot P_0 \\ v_x & v_y & v_z & -v \cdot P_0 \\ n_x & n_y & n_z & -z \cdot P_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Transformações de Projeção

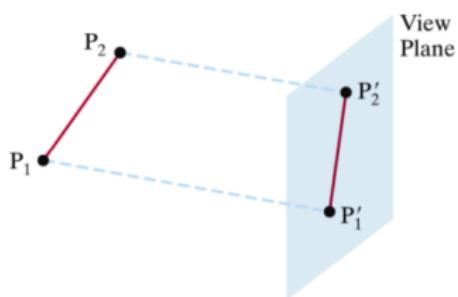
## Transformações de Projeção

- Após a transformação para as coordenadas de visão, o próximo passo do Viewing Pipeline 3D é a projeção no plano de projeção.

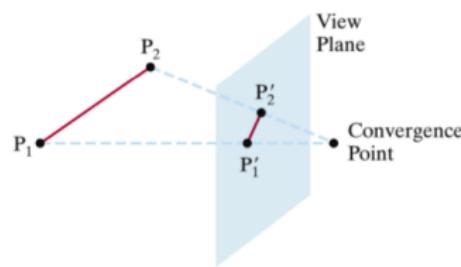
## Pacotes Gráficos

- Em geral os pacotes gráficos suportam:
  - **Projeção Paralela:** as coordenadas são transferidas para o plano de projeção ao longo de linhas paralelas.
  - **Projeção Perspectiva:** as coordenadas são transferidas para um ponto convergindo para um ponto.

# Transformações de Projeção



(c) Projeção Paralela



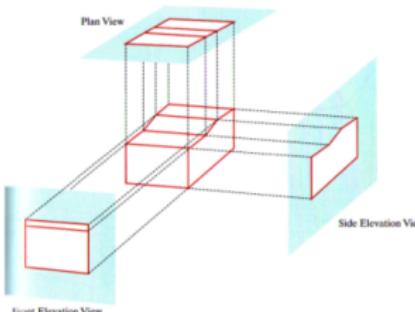
(d) Projeção Perspectiva

## Projeções Ortogonais

# Projeções Ortogonais ou Ortográficas

## Projeções Ortogonais

- Transformam as descrições dos objetos utilizando um plano de projeção ao longo das linhas paralelas ao vetor **N**.
- É utilizada para visão frontal, lateral e superior dos objetos.
- Preserva o tamanho e ângulos dos objetos. Devido a isso este tipo de projeção é principalmente utilizada em programas arquitetônicos.



## Projeções Ortogonais

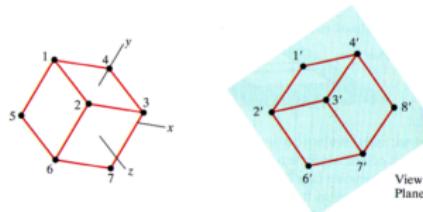
## Projeções Ortogonais ou Ortográficas

## Projeções Axonométricas

- É a projeção ortogonal que mostra mais de uma face do objeto.

## Projeções Axonométricas Isométrica

- A projeção **Isométrica** é a projeção Axonométrica mais comum.
  - Elá consiste em alinhar o plano de projeção de forma a intersectar cada eixo coordenado no qual o objeto é definido a mesma distância da origem.



## Projeções Ortogonais

## Projeções Ortogonais ou Ortográficas

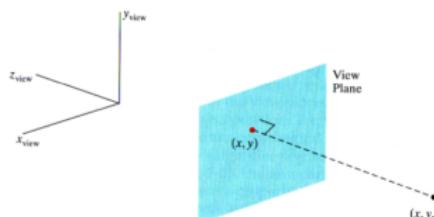
## Coordenadas de Projeções Ortogonais

- Com a direção da projeção sendo paralela ao eixo  $z_{view}$ , as equações para as transformações de projeção ortogonal em uma posição  $(x, y, z)$  são:

$$x_p = x$$

$$y_p = y$$

- O valor de  $z$  é armazenado para futuras procedimentos para determinar a visibilidade.

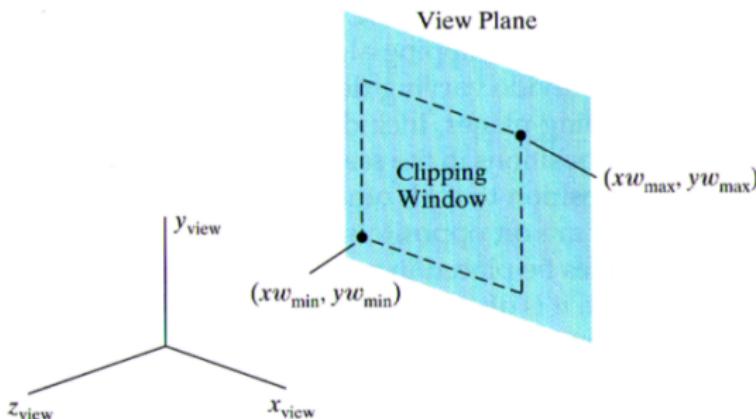


## Projeções Ortogonais

## Projeções Ortogonais ou Ortográficas

## Janela de Recorte e Volume de Projeção Ortogonal

- Para determinar o quando da cena aparecerá, uma janela de recorte é então utilizada.
  - É necessário determinar os limites da janela de projeção sobre o plano formado pelos eixos  $x_{view} \times y_{view}$ , ou seja, com as arestas paralelas aos mesmos.

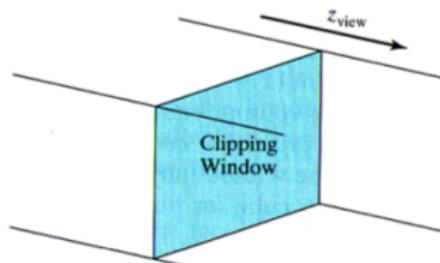


## Projeções Ortogonais

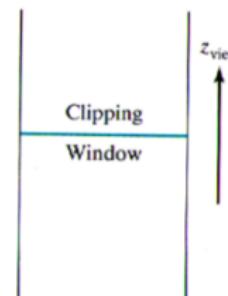
## Projeções Ortogonais ou Ortográficas

## Janela de Recorte e Volume de Projeção Ortogonal

- As arestas da **Janela de Recorte** especificam os valores de  $x$  e  $y$  que serão mostrados na cena, formando assim, o **VOLUME de Visão de Projeção Ortogonal**.



Side View  
(a)



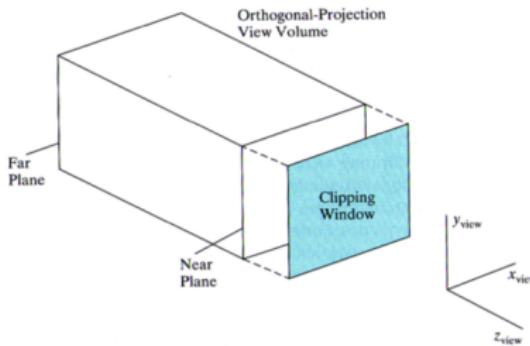
Top View  
(b)

## Projeções Ortogonais

## Projeções Ortogonais ou Ortográficas

## Janela de Recorte e Volume de Projeção Ortogonal

- Para limitar a extensão do volume de projeção dois planos de fronteira, denominados **Planos de Recorte Near/Far** são utilizados paralelamente aos planos de visão.
  - Permite eliminar objetos que estão na frente ou atrás de uma parte da cena.
  - Com a direção de visão ao longo do eixo negativo de  $z_{view}$ , temos  $z_{far} < z_{near}$ .

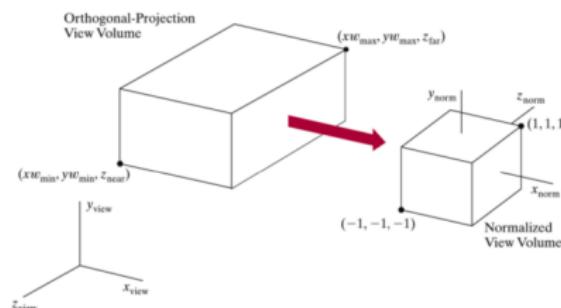


## Projeções Ortogonais

## Projeções Ortogonais ou Ortográficas

## Transformações de Normalização para Projeção Ortogonal

- Qualquer posição  $(x, y, z)$  em uma projeção ortogonal pode ser mapeada para  $(x, y)$ , as coordenadas dentro do volume de visão são as coordenadas de projeção, assim elas podem ser mapeadas para o **volume de visão normalizado** sem precisar ser reprojetadas.



**Figura :** Transformações de normalização de um sistema de referência baseado na regra da mão esquerda.

## Projeções Ortogonais

## Projeções Ortogonais ou Ortográficas

## Transformações de Normalização para Projeção Ortogonal

- A transformação de normalização é semelhante a obtida em 2D, com a adição da coordenada  $z$  normalizada no intervalo  $z_{near}$  a  $z_{far}$  para -1 e 1:

$$\mathbf{M}_{ortho,norm} = R \cdot T$$

$$= \begin{bmatrix} \frac{2}{xw_{max} - xw_{min}} & 0 & 0 & -\frac{xw_{max} + xw_{min}}{xw_{max} - xw_{min}} \\ 0 & \frac{2}{yw_{max} - yw_{min}} & 0 & -\frac{yw_{max} + yw_{min}}{yw_{max} - yw_{min}} \\ 0 & 0 & \frac{-2}{z_{near} - z_{far}} & -\frac{z_{far} + z_{near}}{z_{far} - z_{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Projeções Ortogonais

# Projeções Ortogonais ou Ortográficas

## Transformações de Normalização para Projeção Ortogonal

- A multiplicação desta matriz com a matriz que transforma as coordenadas do mundo em coordenadas de visão produz a transformação correta para se obter as coordenadas corretas para se obter as coordenadas normalizadas da projeção ortogonal.

$$\mathbf{M}_{ortho,norm} \cdot \mathbf{M}_{WC,VC}$$

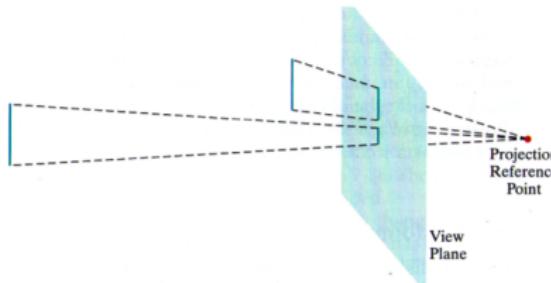
- Com a normalização operações como a de recorte e identificação se superfícies visíveis podem ser feitas de modo mais eficiente.

## Projeções Perspectivas

# Projeções Perspectivas

## Projeções Perspectivas

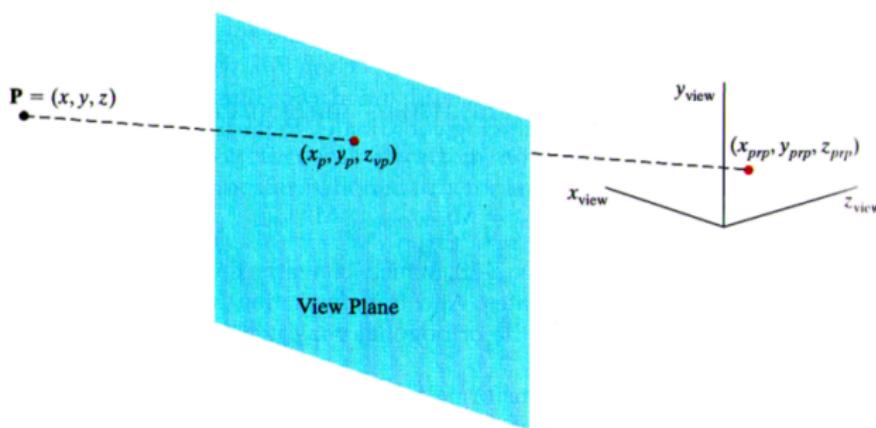
- Para um maior realismo nas cenas, quando comparado a perspectiva, temos que levar em consideração que os raios de luz refletidos na cena possuem caminhos convergentes.
- Uma aproximação desta característica pode ser feita projetando os objetos ao plano de visão ao longo de caminhos convergentes a uma posição chamada **ponto de referência de projeção (centro da projeção)**.



# Transformações de Coordenadas de Projeção Perspectiva

## Transformações de Coordenadas de Projeção Perspectiva

- Algumas bibliotecas gráficas permitem escolher o ponto de projeção  $(x_{ppr}, y_{ppr}, z_{ppr})$



# Transformações de Coordenadas de Projeção Perspectiva

## Transformações de Coordenadas de Projeção Perspectiva

- Considerando que a projeção do ponto  $(x, y, z)$  intersecta o plano de projeção na posição  $(x_{prp}, y_{prp}, z_{prp})$ , podemos descrever qualquer ponto ao longo desta linha de projeção como sendo:

$$x' = x + t(x_{prp} - x)$$

$$y' = y + t(y_{prp} - y)$$

$$z' = z + t(z_{prp} - z)$$

$$0 \leq t \leq 1$$

- No plano de visão  $z' = z_{vp}$ , então podemos encontrar  $t$ :

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

# Transformações de Coordenadas de Projeção Perspectiva

## Transformações de Coordenadas de Projeção Perspectiva

- Substituindo o valor de  $t$  nas equações de  $x'$  e  $y'$  obtemos as equações de projeção perspectiva.

# Transformações de Coordenadas de Projeção Perspectiva

## Posição do Ponto de Projeção

- Geralmente o ponto de projeção está entre o plano de projeção e a cena, mas existem outras posições possíveis (menos sobre o plano de projeção).

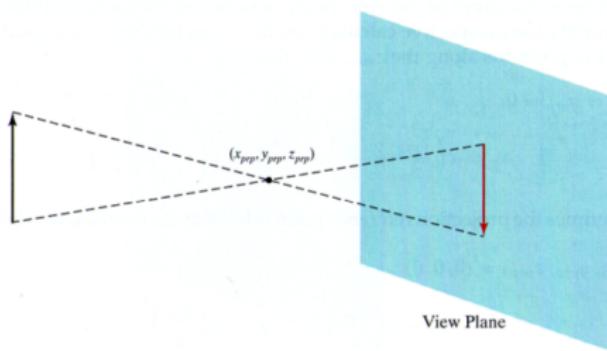
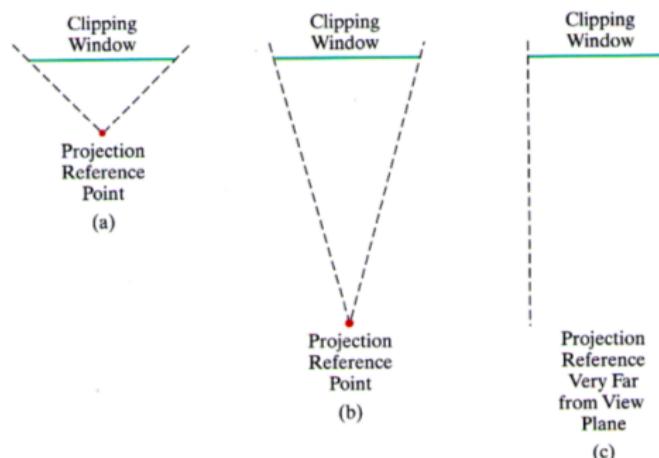


Figura : Os objetos da cena será invertidos caso o ponto de referência esteja a frente do plano.

# Transformações de Coordenadas de Projeção Perspectiva

## Posição do Ponto de Projeção

- Os efeitos de perspectiva dependem da posição em que o centro de projeção está em relação ao plano de visão.



**Figura :** Quanto mais perto o centro de projeção está do plano, maiores serão os objetos próximos ao plano.

# Transformações de Coordenadas de Projeção Perspectiva

## Pontos de Fuga na Projeção Perspectiva

- Linhas paralelas entre si e ao plano de projeção são projetadas como linhas paralelas.
- Linhas paralelas que não são paralelas ao plano de projeção são projetadas em linhas convergentes.
- O ponto em que as Linhas convergem é denominado **Ponto de Fuga**.



# Transformações de Coordenadas de Projeção Perspectiva

## Pontos de Fuga na Projeção Perspectiva

- Os conjuntos de linhas que são paralelas ao algum dos eixos principais dos objetos levam a composição de **pontos de fuga principais**.
  - É possível controlar os número de pontos de fuga, que podem ser 1,2 ou 3. Para tal controle basta mudar a orientação do plano de projeção.
  - O número de pontos de fuga principais é igual a quantidade de eixo principais do objeto que intersectam o plano de projeção.

## Projeções Perspectivas

## Transformações de Coordenadas de Projeção Perspectiva

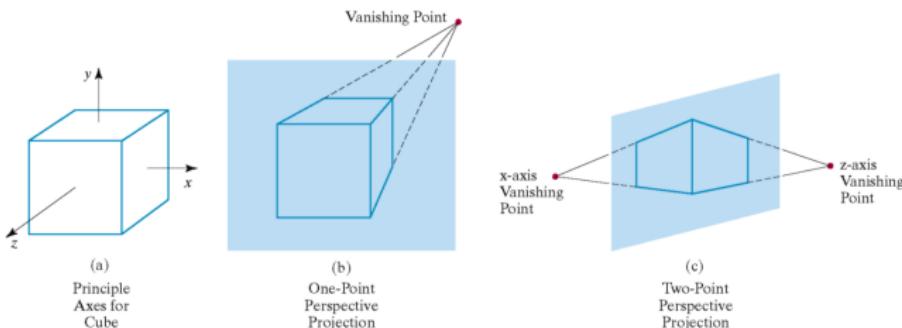
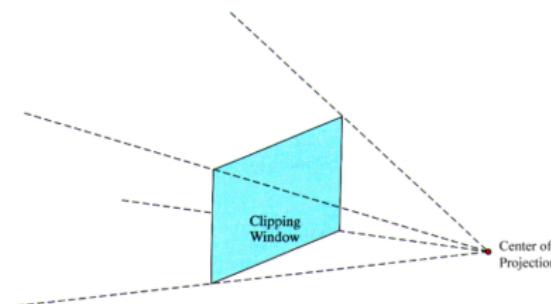


Figura : Em (b) o plano  $xy$  é paralelo ao plano(apenas  $z$  é intersectado), e em (c) os eixos  $x$  e  $y$  são intersectados.

# Transformações de Coordenadas de Projeção Perspectiva

## Volume de Projeção Perspectiva

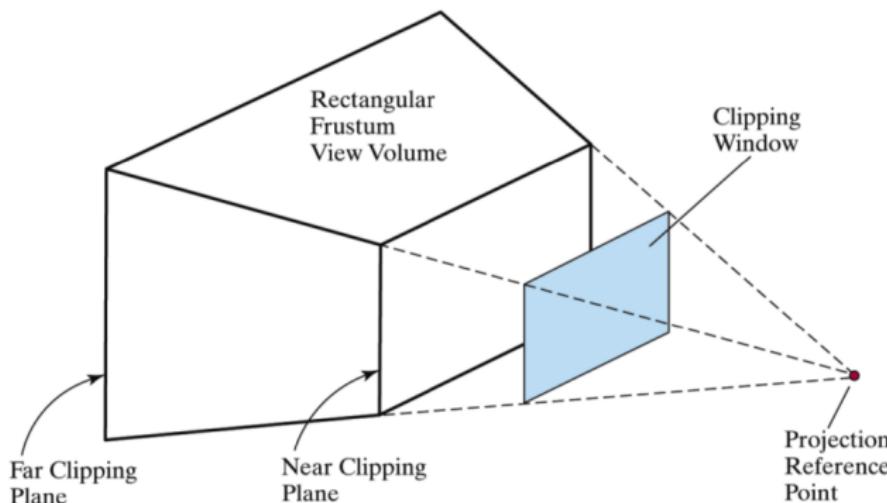
- Na projeção em perspectiva, o volume de visão é definido por uma pirâmide infinita com o ápice no ponto que corresponde ao centro de projeção. Esta pirâmide é normalmente chamada de **Pirâmide de Visão**.
  - Objetos estão fora dos limites da pirâmide são excluídos das rotinas de recortes.



# Transformações de Coordenadas de Projeção Perspectiva

## Volume de Projeção Perspectiva

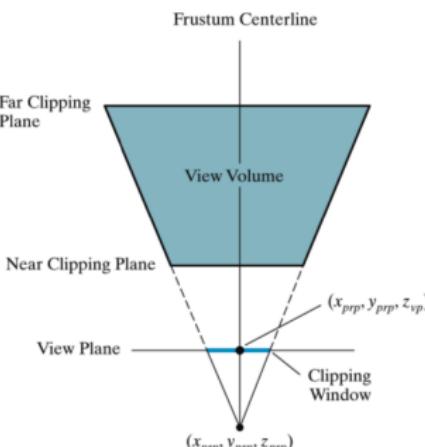
- Adicionando os planos de recorte near/far perpendiculares ao eixo  $z_{view}$  a pirâmide de visão é truncada resultando em um tronco de pirâmide denominado **Frustum**.



# Transformações de Coordenadas de Projeção Perspectiva

## Frustum Simétrico

- Se a linha que parte do centro de projeção e cruza o centro da janela de recorte intersectando a linha central do Frustum de projeção e perspectiva.
  - Se essa linha for perpendicular ao plano de visão então o Frustum é simétrico.



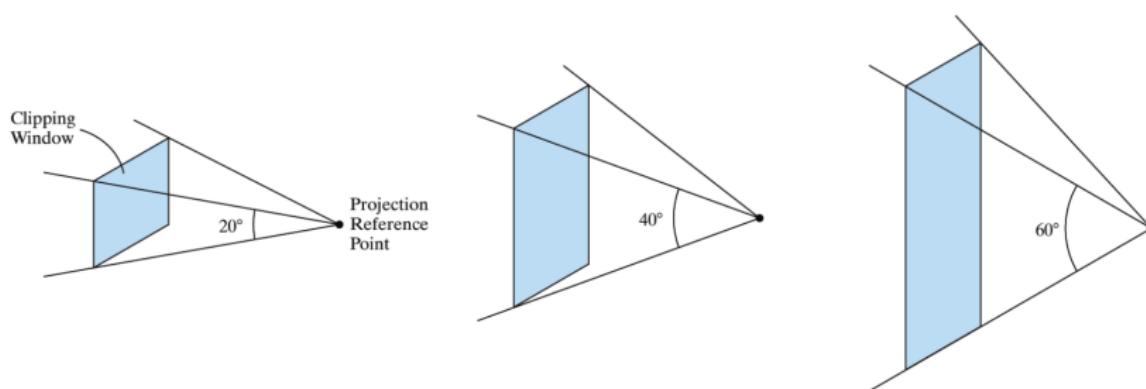
# Transformações de Coordenadas de Projeção Perspectiva

## Frustrum Simétrico

- É possível definir um ângulo de visão que influenciará no tamanho da janela de recorte.
  - Diminuir o ângulo do campo de visão diminui a janela de recorte.
    - Mover o ponto para longe do plano de visão.
    - Zoom-in de uma pequena região da cena.
  - Aumentar o ângulo do campo de visão aumenta a janela de recorte.
    - Mover o ponto para perto do plano de visão.
    - Zoom-out da cena.

## Projeções Perspectivas

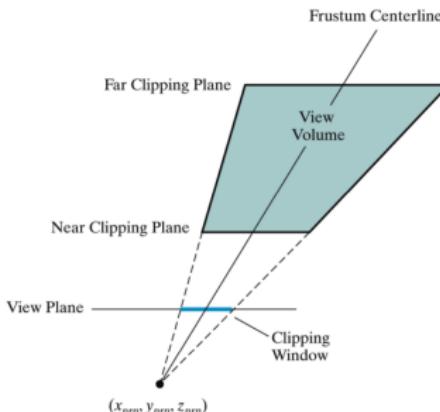
## Transformações de Coordenadas de Projeção Perspectiva



# Transformações de Coordenadas de Projeção Perspectiva

## Frustum Oblíquo

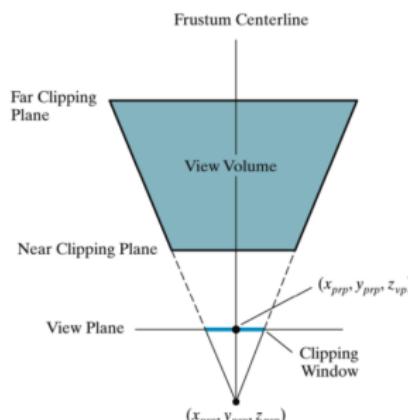
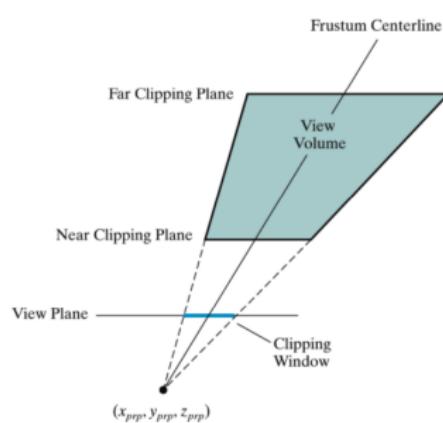
- Se a linha que parte do centro de projeção e cruza o centro da janela de recorte não intersecta a linha central do Frustum de projeção e perspectiva, então temos um **Frustum oblíquo**.
  - Então é necessário transformar este Frustum é um Frustum simétrico.



# Transformações de Coordenadas de Projeção Perspectiva

## Frustum Oblíquo

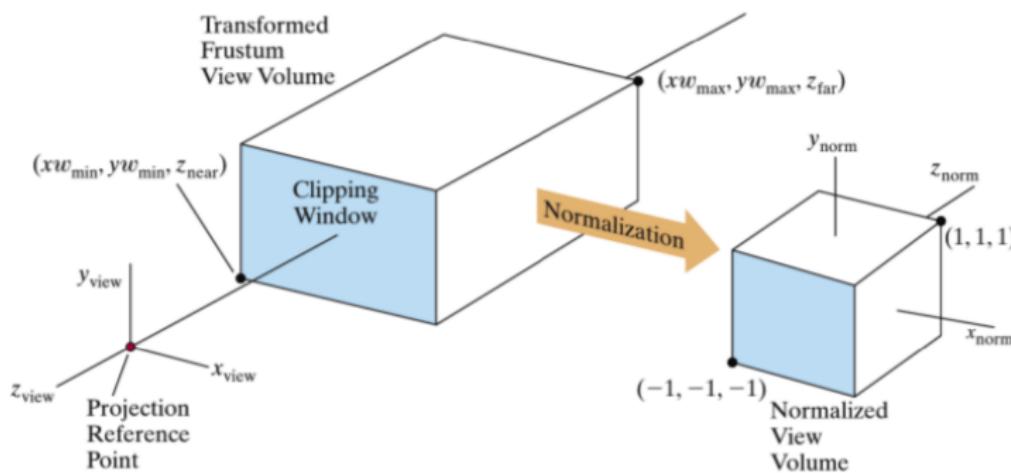
- Esta transformação pode ser obtida fazendo o cisalhamento em z.



# Transformações de Coordenadas de Projeção Perspectiva

## Normalização

- O ultimo passo da projeção perspectiva é mapear o paralelepípedo para um **volume normalizado**.



# Algoritmos de Recorte 3D

## Algoritmos de Recorte 3D

- Com o cubo normalizado os algoritmos de recorte podem ser aplicados com maior eficiência.
- Os algoritmos de recorte eliminam os objetos fora do volume de visão e processão o resto.
- São extensões dos algoritmos de recorte 2D, mas com planos como fronteiras ao invés de linhas.
- O recorte é aplicado após todas as transformações serem aplicadas.

# Algoritmos de Recorte 3D

## Algoritmos de Recorte 3D

- No pipeline 3D temos que os pontos processados utilizando as matrizes de transformações geométricas estão em uma representação homogênea.

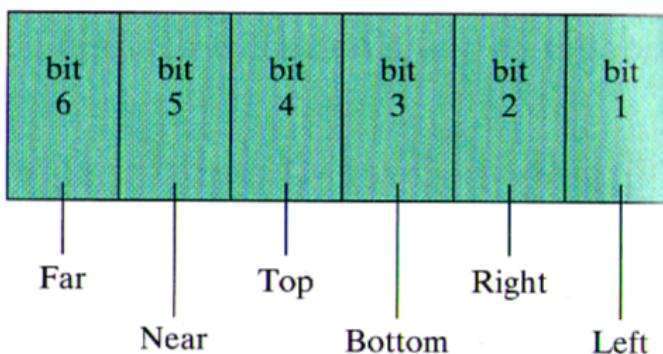
$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \mathbf{M} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Onde  $\mathbf{M}$  é a concatenação de todas as matrizes.
- Um método de recorte efetivo é aplicar as rotinas de recorte na representação homogênea.
  - Os procedimentos de recorte podem ser iguais em qualquer tipo de projeção aplicada.

# Algoritmos de Recorte 3D

## Código de Região 3D

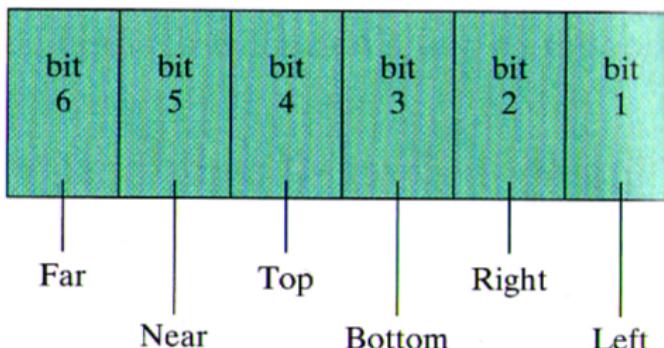
- O conceito de código de região 2D (Algoritmo de Cohen-Sutherland) pode ser estendido para representações 3D adicionando mais alguns bits.
  - Código de 6 bits.



# Algoritmos de Recorte 3D

## Código de Região 3D

- Os valores dos bits são calculados de forma semelhante ao 2D.
  - 0 está dentro de uma fronteira, 1 está fora.



# Algoritmos de Recorte 3D

## Código de Região 3D

- Considerando que estamos trabalhando com coordenadas homogêneas  $P = (x_h, y_h, z_h, h)$ , um ponto dentro do volume de visão deve satisfazer:

$$-1 \leq \frac{x_h}{h} \leq 1$$

$$-1 \leq \frac{y_h}{h} \leq 1$$

$$-1 \leq \frac{z_h}{h} \leq 1$$

# Algoritmos de Recorte 3D

## Código de Região 3D

- Com  $P = (x_h, y_h, z_h, h)$  e assumindo  $h \neq 0$  podemos calcular:
  - se  $h > 0$

$$-h \leq x_h \leq h$$

$$-h \leq y_h \leq h$$

$$-h \leq z_h \leq h$$

- se  $h < 0$

$$h \leq x_h \leq -h$$

$$h \leq y_h \leq -h$$

$$h \leq z_h \leq -h$$

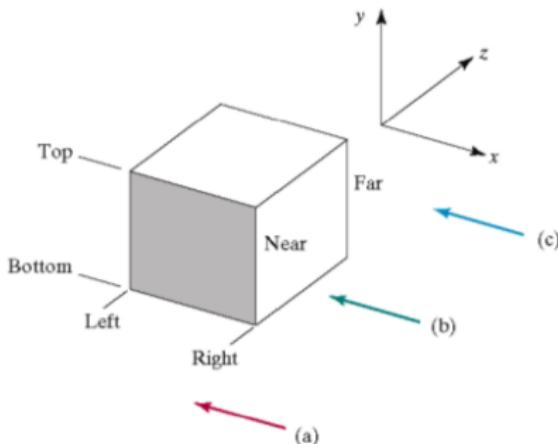
# Algoritmos de Recorte 3D

## Código de Região 3D

- Com  $h > 0$  podemos definir os valores dos bits como:

$$\text{bit}_1 = 1 \text{ se } h + x_h < 0 \text{ (esquerda)}$$

# Algoritmos de Recorte 3D



|        |        |        |
|--------|--------|--------|
| 011001 | 011000 | 011010 |
| 010001 | 010000 | 010010 |
| 010101 | 010100 | 010110 |

Region Codes  
In Front of Near Plane  
(a)

|        |        |        |
|--------|--------|--------|
| 001001 | 001000 | 001010 |
| 000001 | 000000 | 000010 |
| 000101 | 000100 | 000110 |

Region Codes  
Between Near and Far Planes  
(b)

|        |        |        |
|--------|--------|--------|
| 101001 | 101000 | 101010 |
| 100001 | 100000 | 100010 |
| 100101 | 100100 | 100110 |

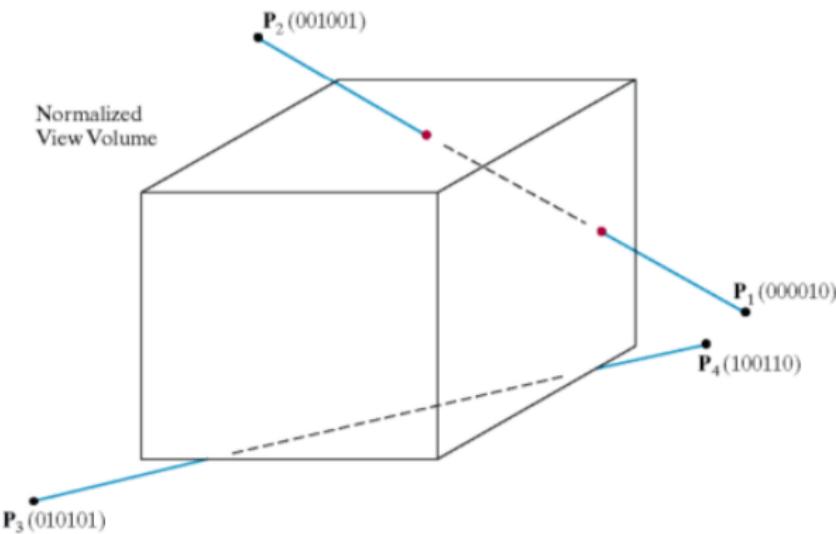
Region Codes  
Behind Far Plane  
(c)

# Algoritmos de Recorte 3D

## Recorte de Linhas

- O recorte de linha é essencialmente o mesmo que o recorte de linha 2D.
  - Se ambos os pontos finais da linhas são 000000, a linha está dentro.
  - Uma linha está dentro se a operação “**ou**” sobre seus pontos finais for igual a 000000.
  - Uma linha está fora se a operação “e” sobre seus pontos finais for diferente de 000000.

# Algoritmos de Recorte 3D



# Algoritmos de Recorte 3D

## Recorte de Polígonos 3D

- Pacotes gráficos geralmente lidam com os objetos descritos com equações lineares, sendo assim, as intersecções das arestas dos objetos com o plano de recorte são utilizadas para delimitar o objeto.
- Além disso, algumas rotinas simples como o **bounding box** podem ser utilizadas para definir se um polígono está dentro ou fora das fronteiras.