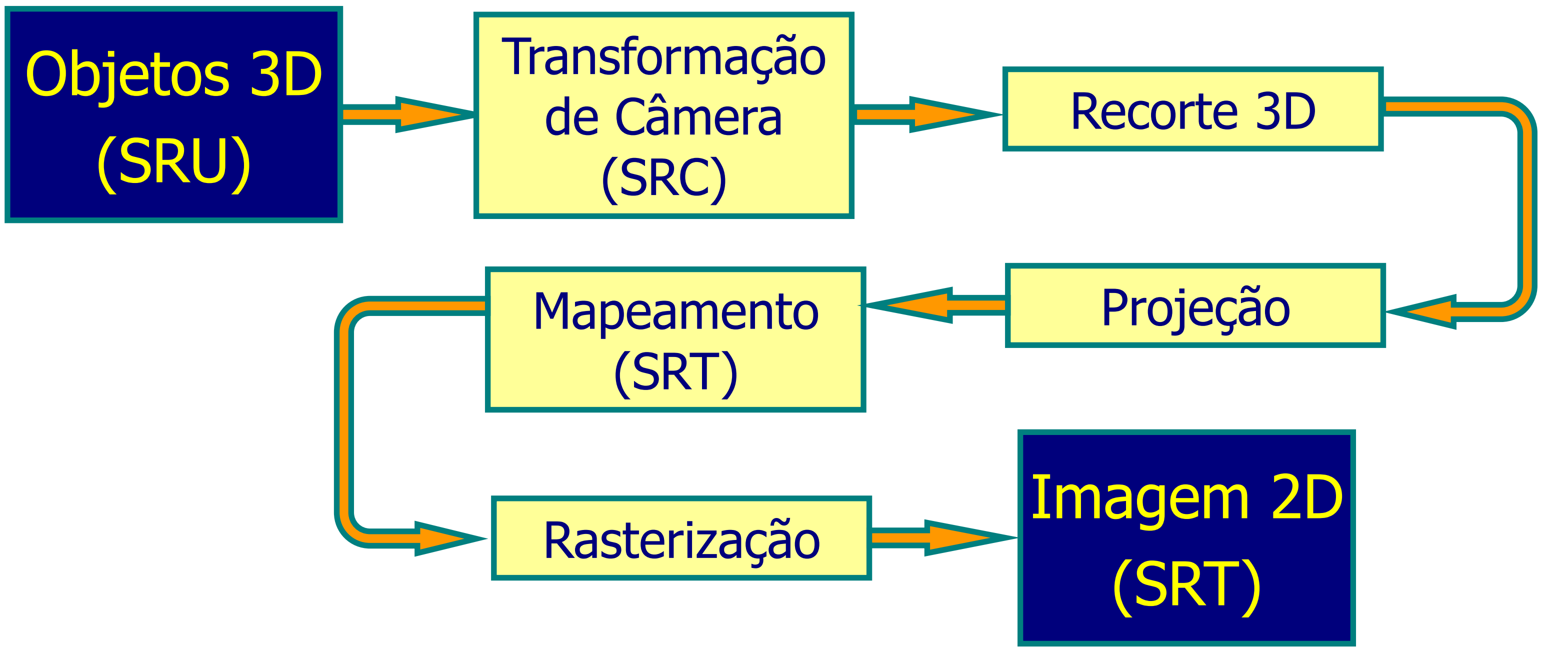
VISUALIZAÇÃO EM 3D

Adair Santa Catarina
Curso de Ciência da Computação
Unioeste – Campus de Cascavel – PR

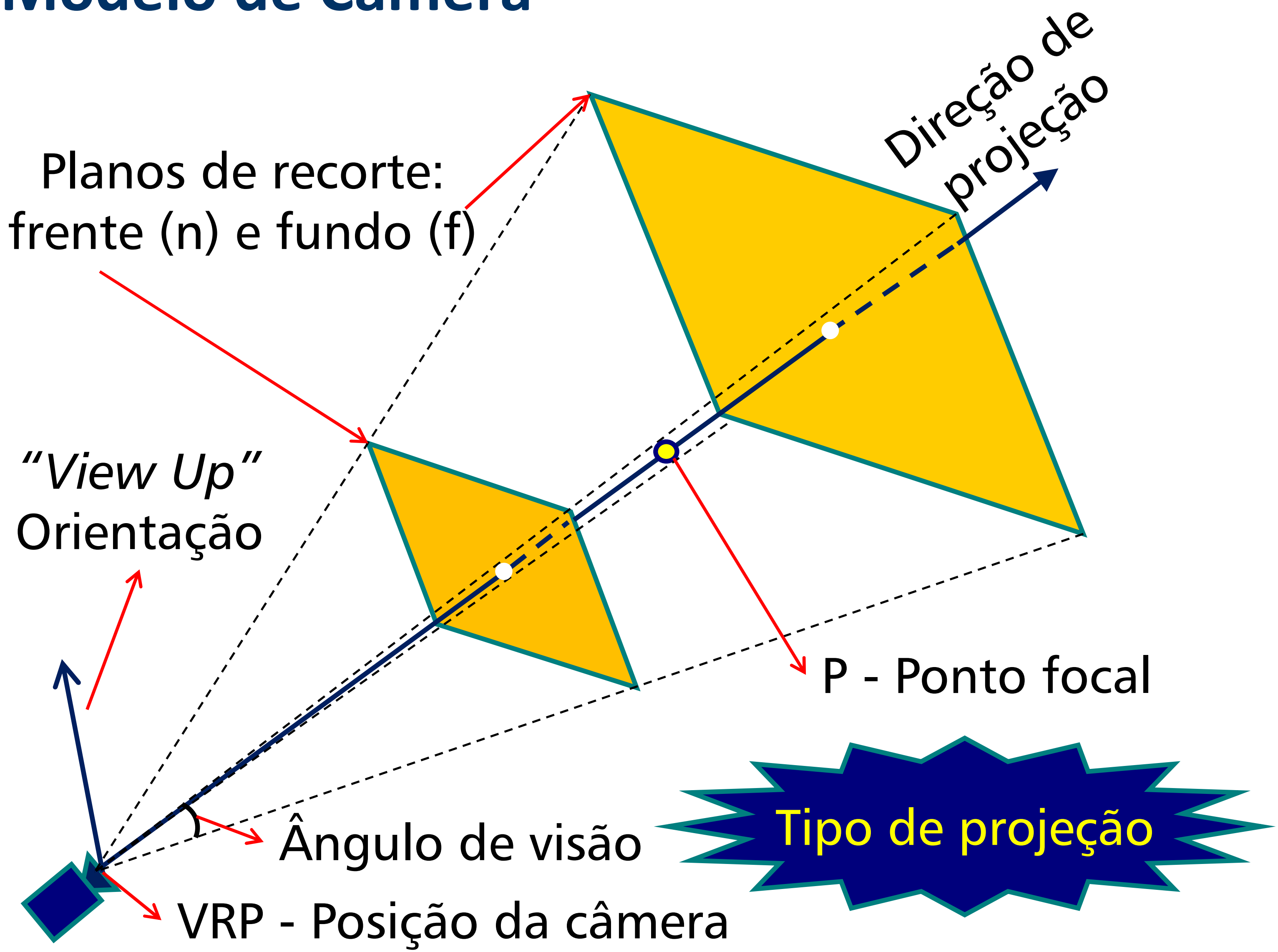
Jan/2021

Pipeline de Visualização

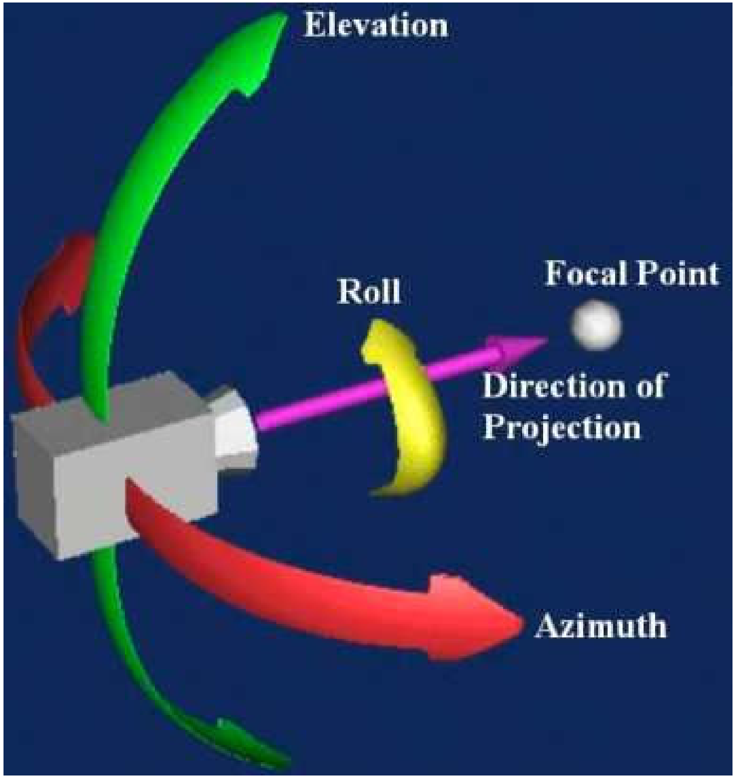
Corresponde a uma sequência de operações realizadas sobre os objetos 3D da cena para desenhá-los corretamente em uma tela 2D.

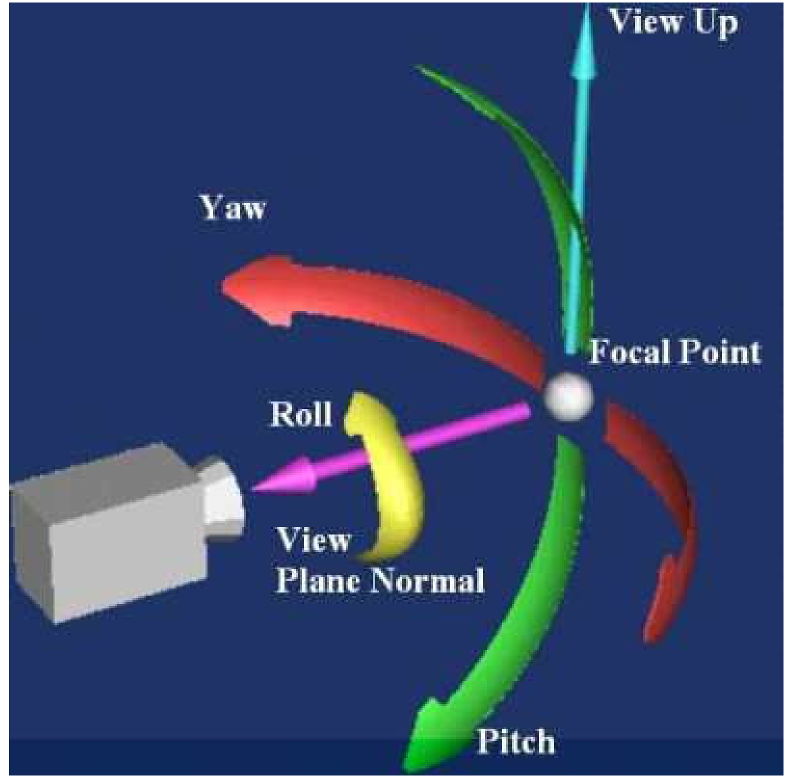


O Modelo de Câmera



Movimentos de Câmera



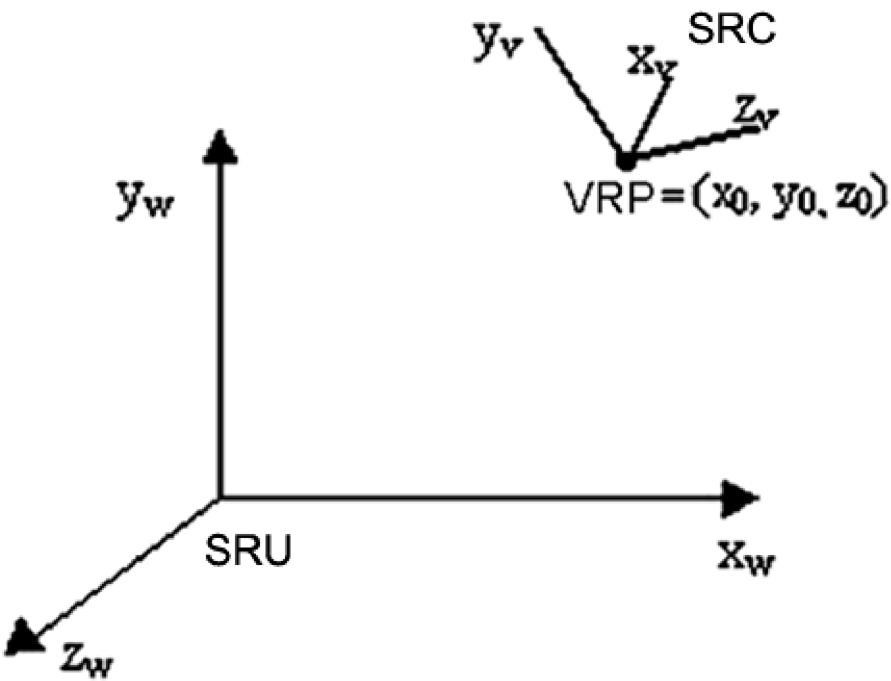


- Dolly (in, out): modifica a distância da câmera;
- Zoom (in, out): altera o ângulo de visão.

Sistema de Referência da Câmera (SRC)

Primeira etapa do *pipeline* de visualização:

Transformação de Câmera

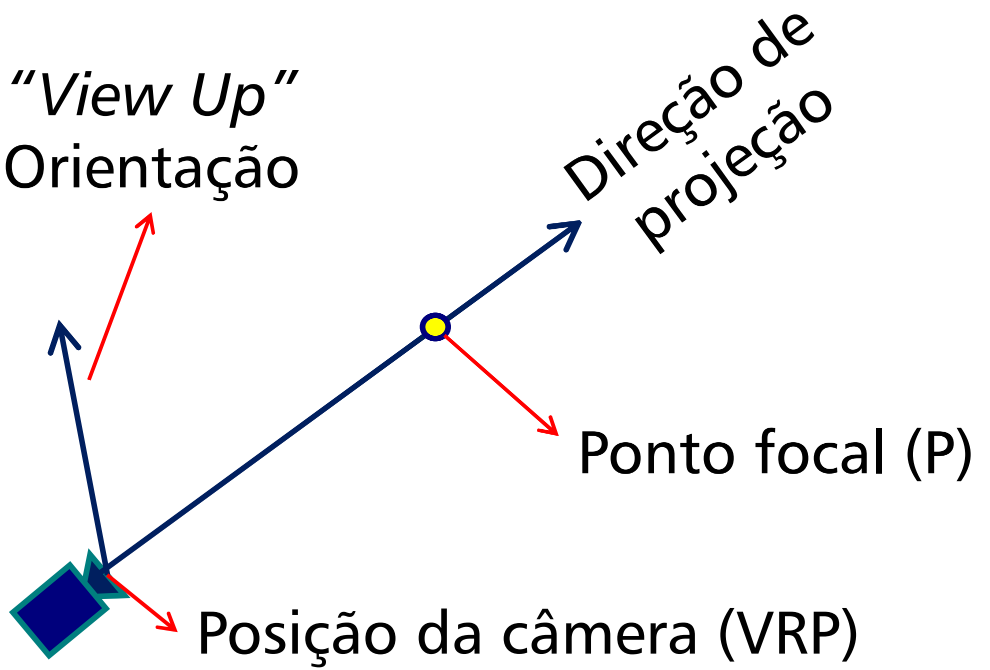


Objetivo

Converter as coordenadas dos objetos do Sistema de Referência do Universo (SRU) para o SRC.

Especificação do SRC

Definir a posição da câmera (VRP), o ponto focal (P) e o vetor de orientação da câmera *(View Up)*.



Transformação de SRU para o SRC

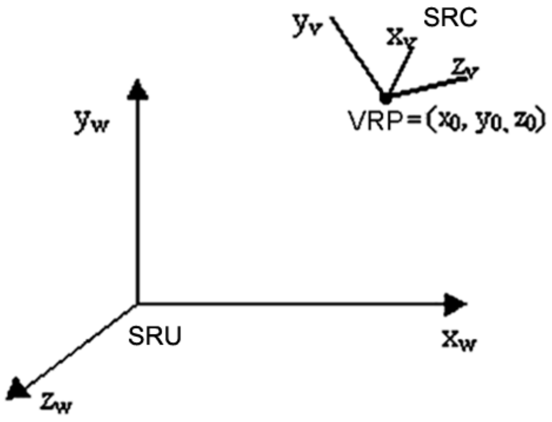
Equivale a uma transformação que superpõe o SRC (mão-direita) ao SRU, usando transformações geométricas de translação e rotação.

Processo em 2 etapas:

1) Transladar o VRP para a origem do SRU

$$T(-x_0, -y_0, -z_0)$$

2) Aplicar rotações para alinhar os eixos do SRC (x_v, y_v, z_v) com os eixos do SRU (x_w, y_w, z_w).



Transformação de SRU para o SRC

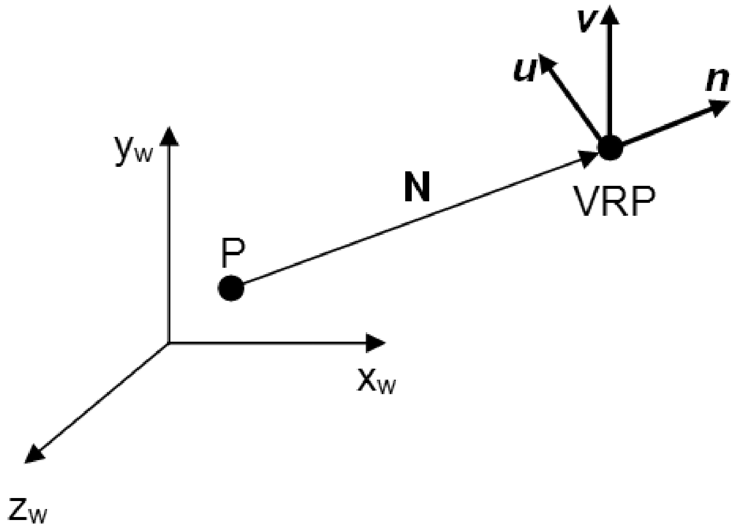
Como determinar as rotações que alinham os eixos do SRC e do SRU?

Processo em 3 etapas:

1) Determinar o vetor ***n***;

2) Determinar o vetor ***v***;

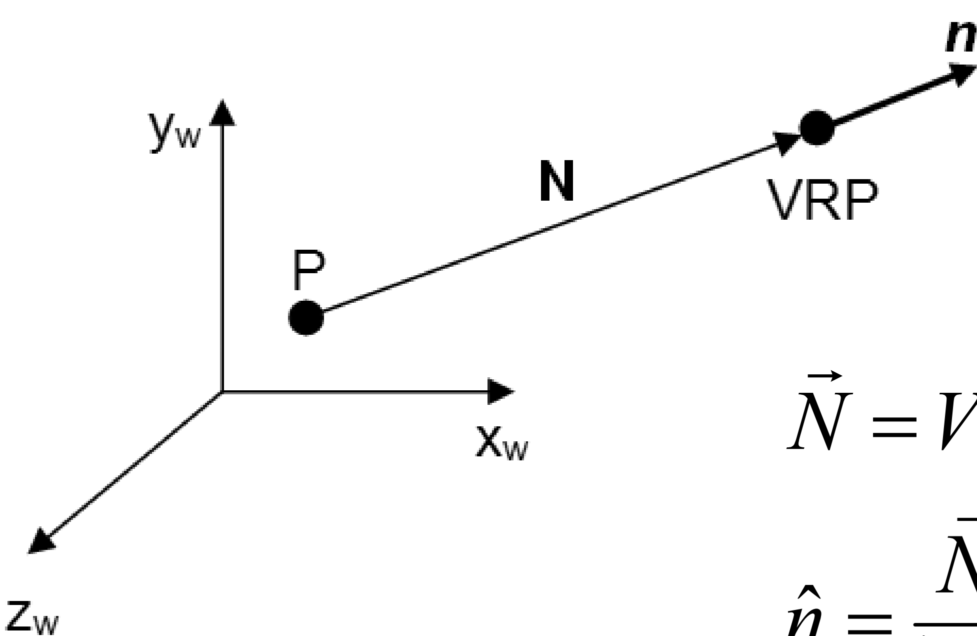
3) Determinar o vetor ***u***.



u, ***v*** e ***n*** formam a base ortonormal do SRC.

Obtenção da Base Ortonormal do SRC

1) Calculando o vetor ***n***:



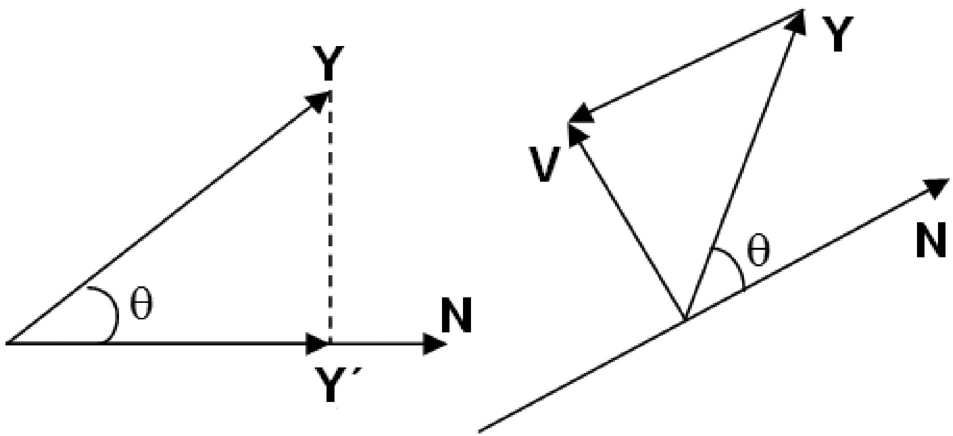
$$\vec{N} = VRP - P$$

$$\hat{n} = \frac{\vec{N}}{\left|\vec{N}\right|} = \left(n_1, n_2, n_3\right)$$

Obtenção da Base Ortonormal do SRC

2) Calculando o vetor ***v***:

Projetar um vetor ***Y*** qualquer sobre o plano de projeção obtendo o vetor ***V***.



$$\vec{Y} = (0,1,0)$$

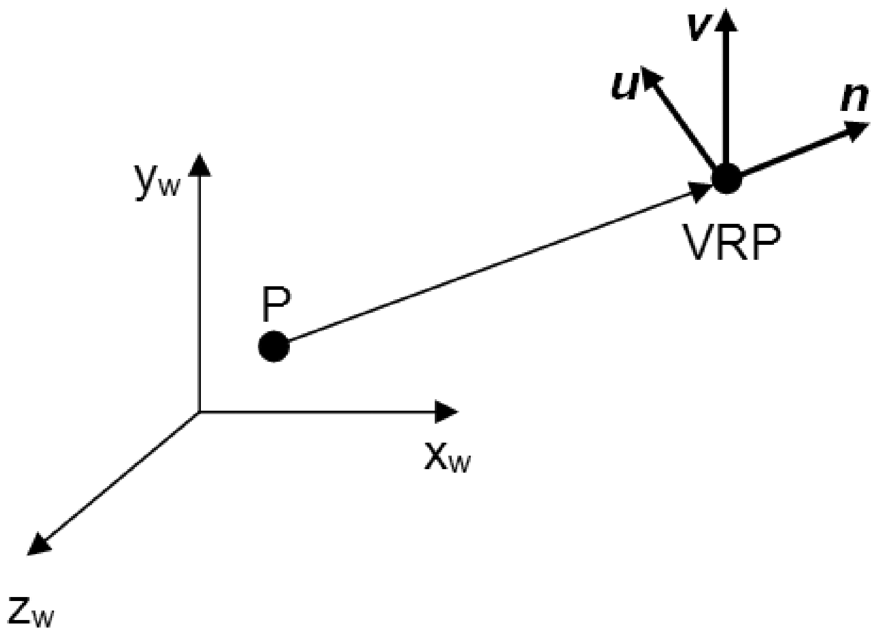
$$\vec{Y}' = \frac{\vec{N}\cdot\vec{Y}}{\left|\vec{N}\right|}\cdot\frac{\vec{N}}{\left|\vec{N}\right|}$$

$$\vec{Y}' = \left(\vec{Y}\cdot\hat{n}\right)\cdot\hat{n}$$

$$\vec{V} = \vec{Y} - \left(\vec{Y}\cdot\hat{n}\right)\cdot\hat{n} \qquad \hat{v} = \frac{\vec{V}}{\left|\vec{V}\right|} = \left(v_1, v_2, v_3\right)$$

Obtenção da Base Ortonormal do SRC

3) Calculando o vetor ***u***:



$$\vec{U} = \vec{V} \times \vec{N}$$

$$\hat{u} = \frac{\vec{U}}{\left|\vec{U}\right|} = \left(u_1, u_2, u_3\right)$$

ou

$$\hat{u} = \hat{v} \times \hat{n}$$

Matriz de Transformação $M_{SRU,SRC}$

A base ortonormal do SRC define a matriz de rotações que alinha os eixos do SRC aos eixos do SRU.

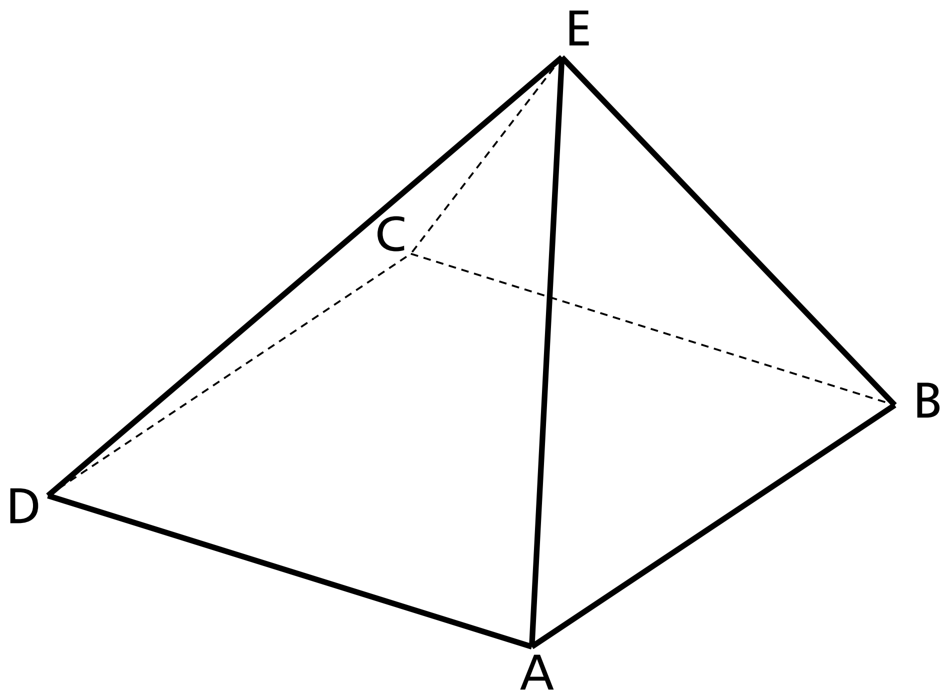
$$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Concatenando as matrizes, tem-se:

$$M_{SRU,SRC} = R \cdot T = \begin{bmatrix} u_1 & u_2 & u_3 & -VRP \cdot \hat{u} \\ v_1 & v_2 & v_3 & -VRP \cdot \hat{v} \\ n_1 & n_2 & n_3 & -VRP \cdot \hat{n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Exercício 01 – Conversão SRU, SRC

O objeto abaixo está descrito no SRU. Considerando que a câmera está posicionada em VRP = (50, 15, 30) e o ponto focal P = (20, 6, 15), converta as coordenadas do objeto para o SRC.



A = (30, 2, 25)

B = (35, 4, 20)

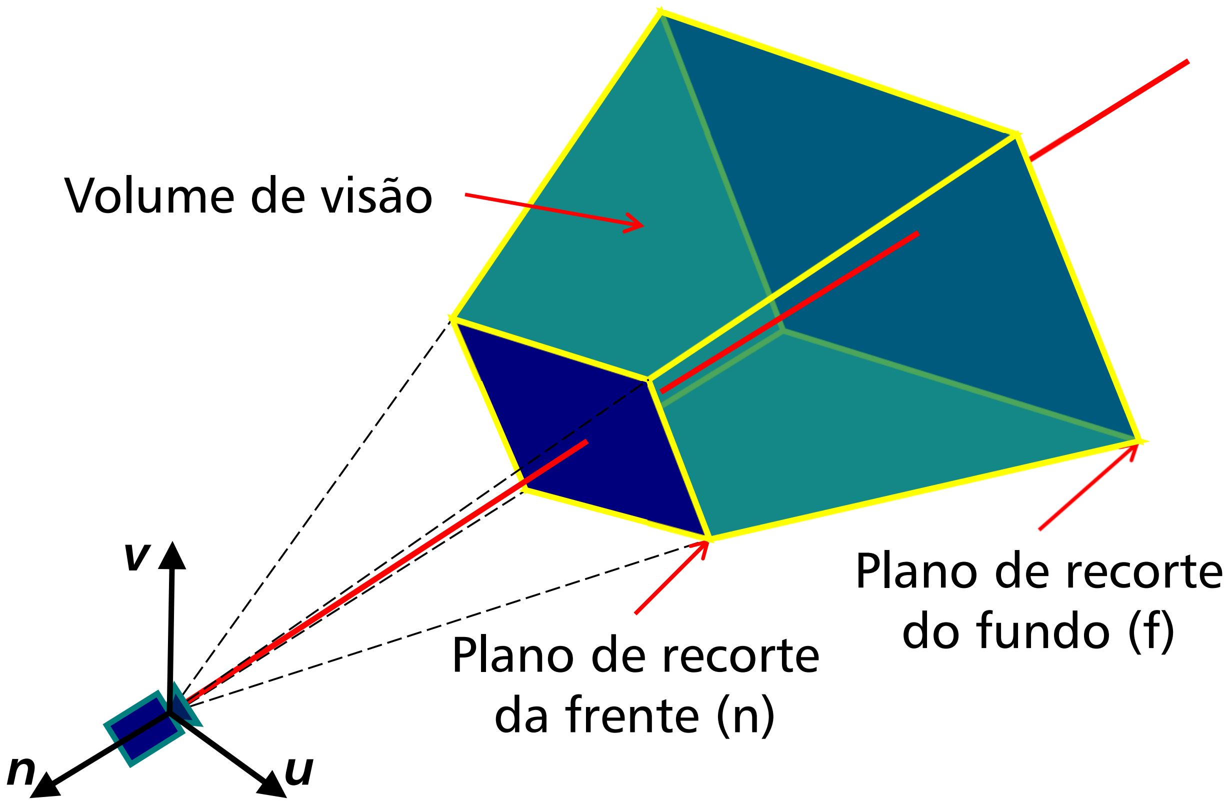
C = (25, 3, 18)

D = (20, 1, 23)

E = (30, 10, 22.5)

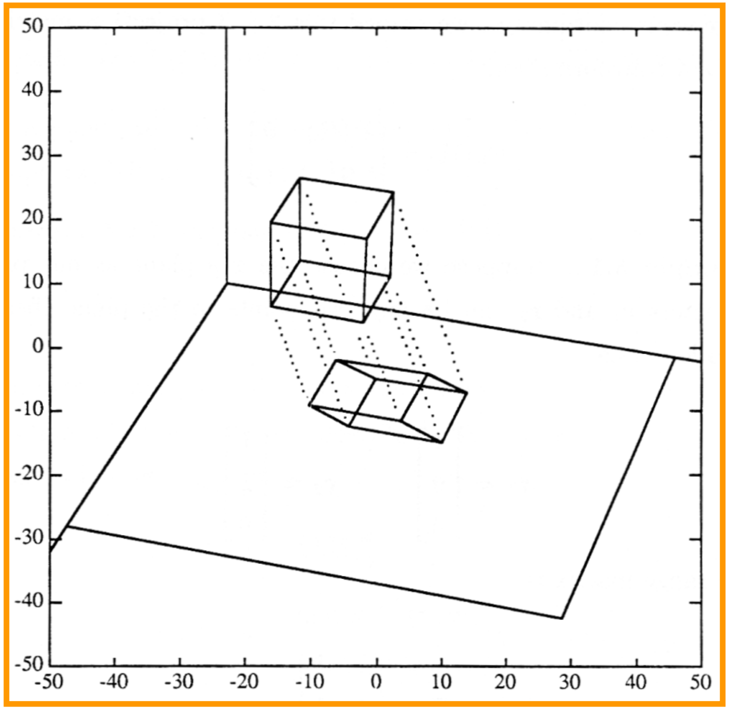
Recorte 3D

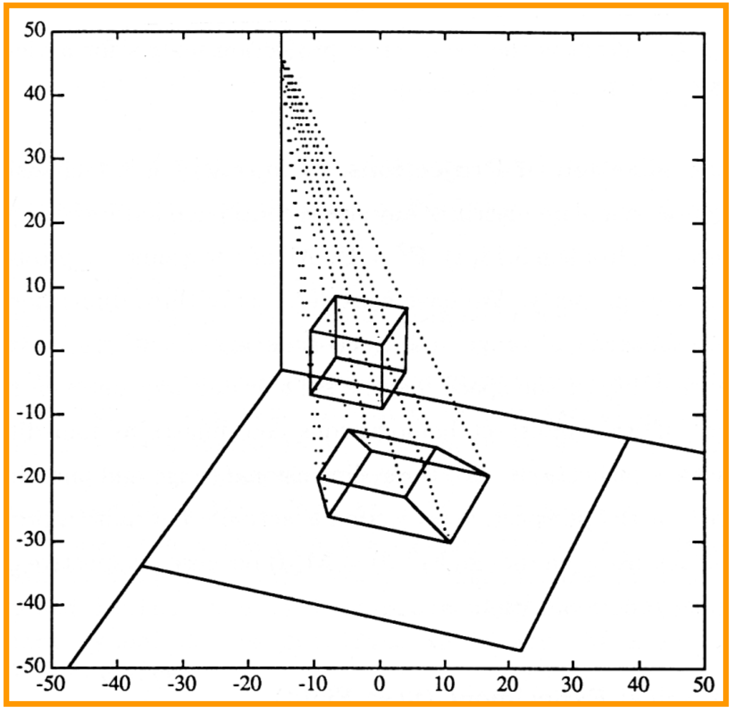
A transformação de recorte 3D visa eliminar os objetos que estão fora do volume de visão (z > n ou z < f).



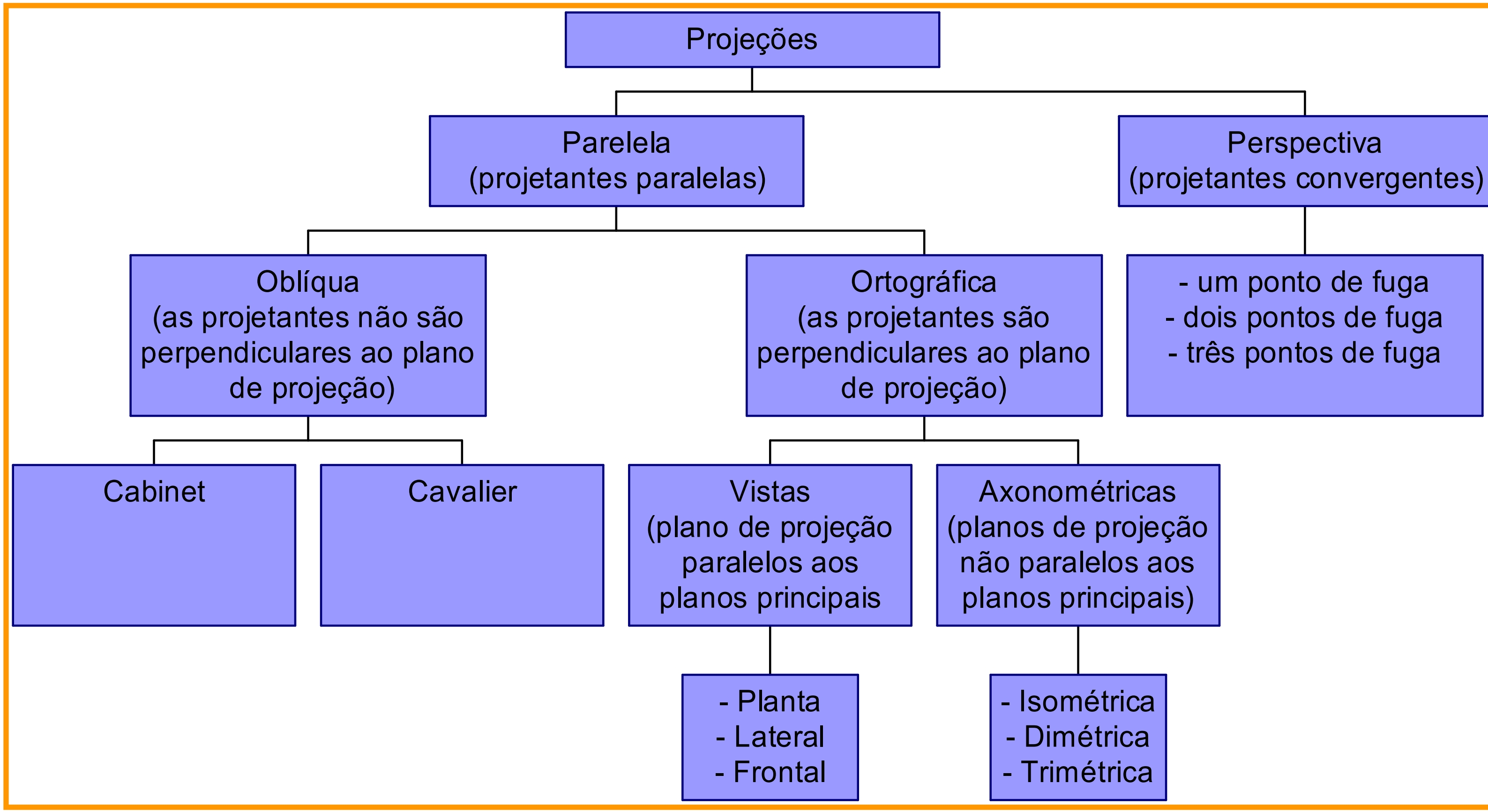
Projeções

Projeção é o processo que possibilita representar objetos tridimensionais (3D) em meios bidimensionais (2D), que são os dispositivos de exibição utilizados nos computadores.



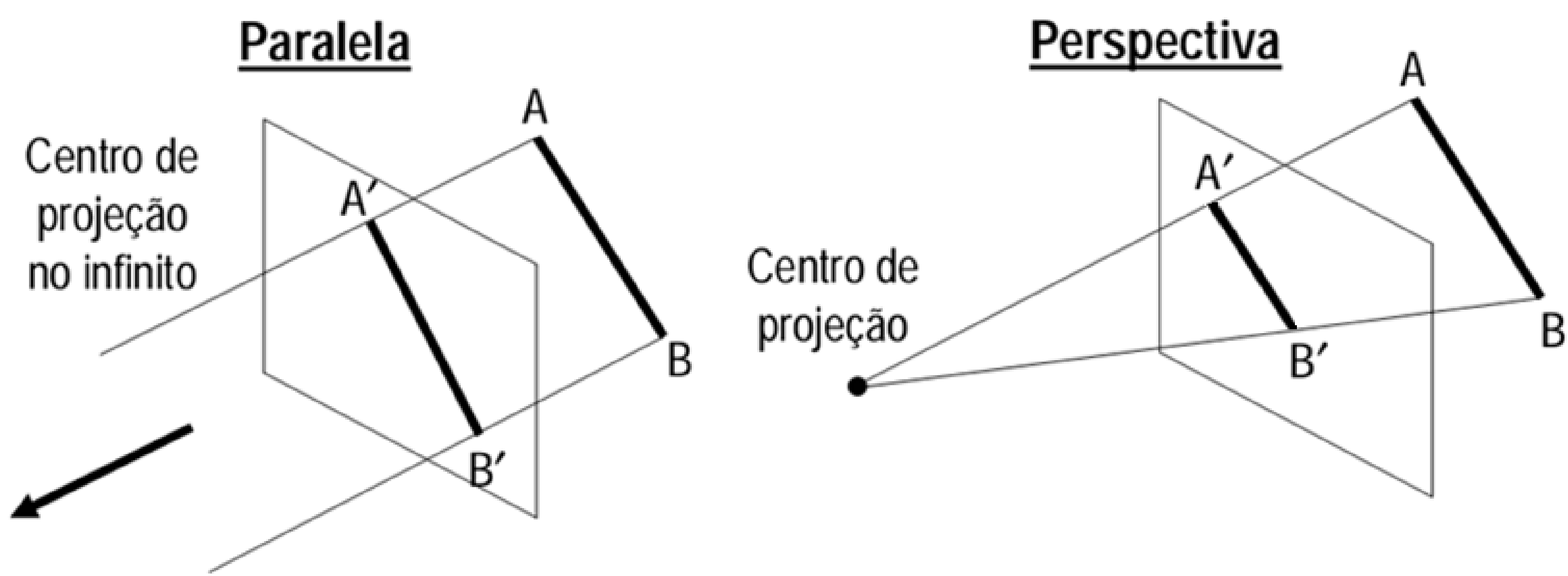


Taxonomia das Projeções



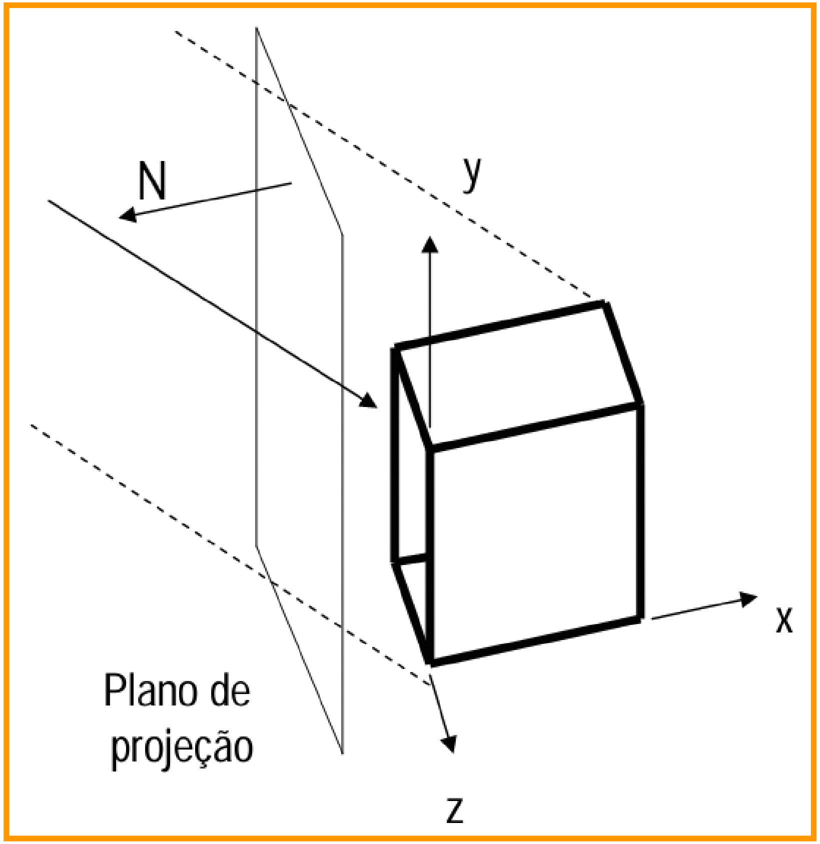
Projeção Paralela x Perspectiva

- **Paralela:** as projetantes que incidem no plano de projeção são paralelas entre si;
- **Perspectiva:** as projetantes convergem no centro de projeção.



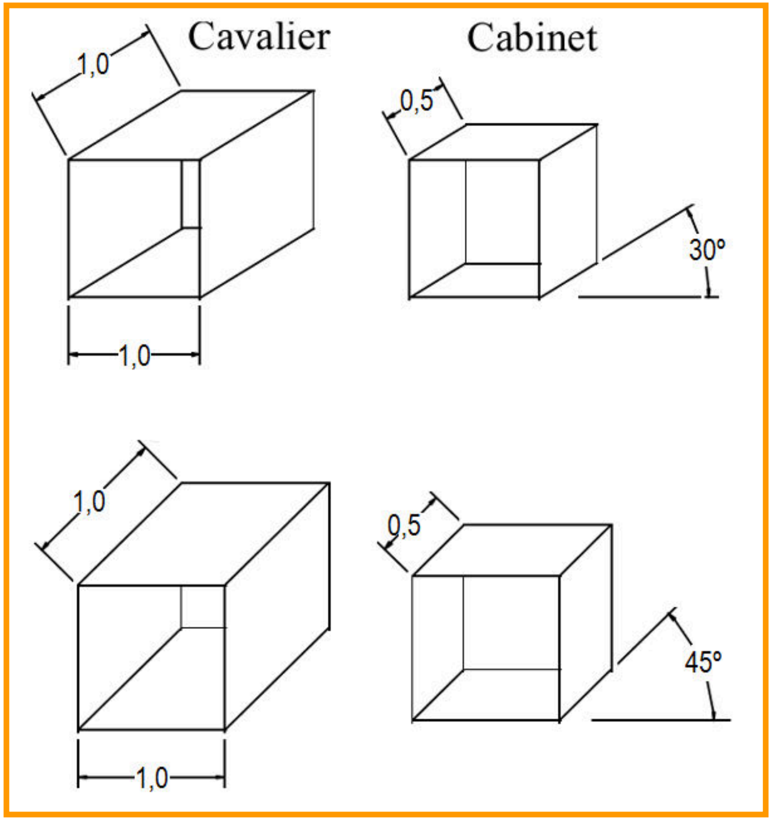
Projeções Paralelas Oblíquas

Nas projeções oblíquas as projetantes não incidem perpendicularmente ao plano de projeção.



Projeções Oblíquas – Cabinet x Cavalier

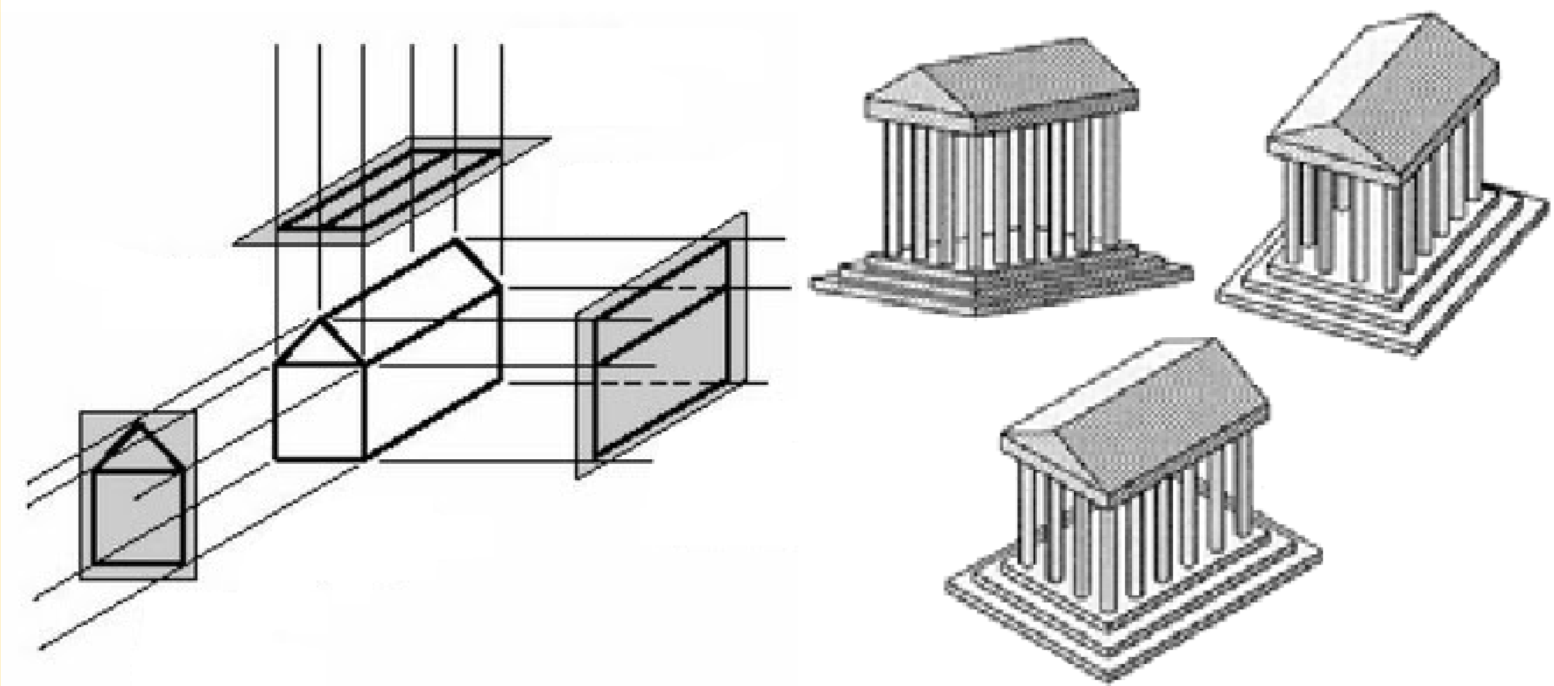
Nas projeções oblíquas as dimensões das arestas paralelas ao plano de projeção são equivalentes às dimensões reais do objeto.



Na projeção Cabinet há um encolhimento na profundidade para corrigir a distorção da projeção oblíqua.

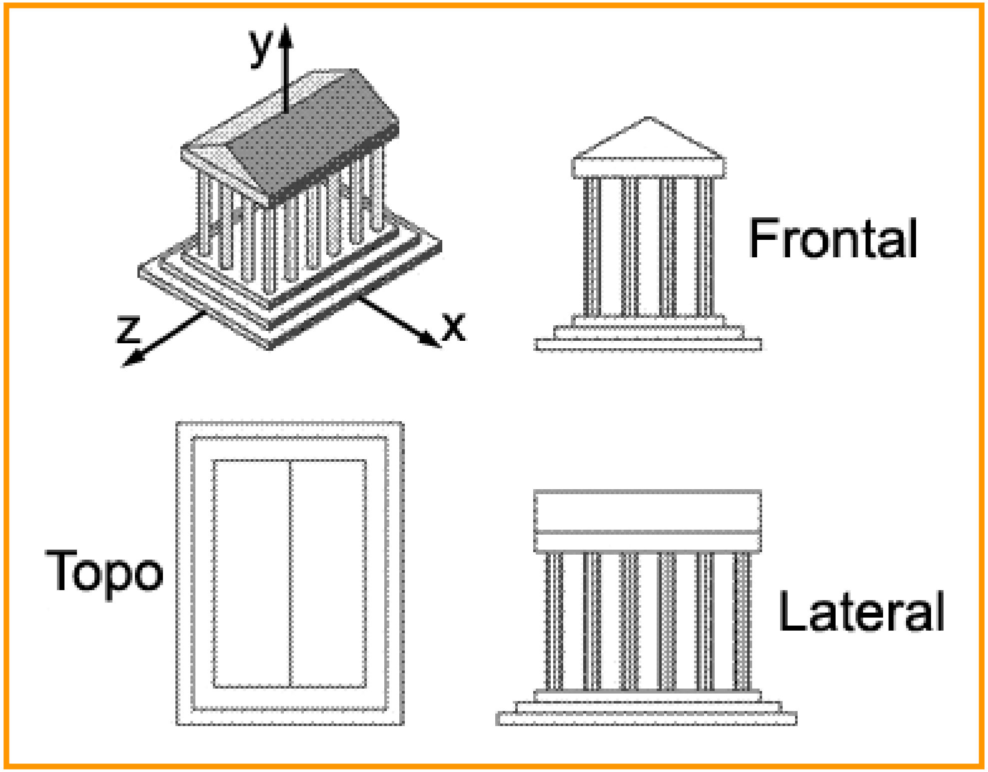
Projeções Paralelas Ortográficas

Nas projeções ortográficas as projetantes incidem perpendicularmente ao plano de projeção.



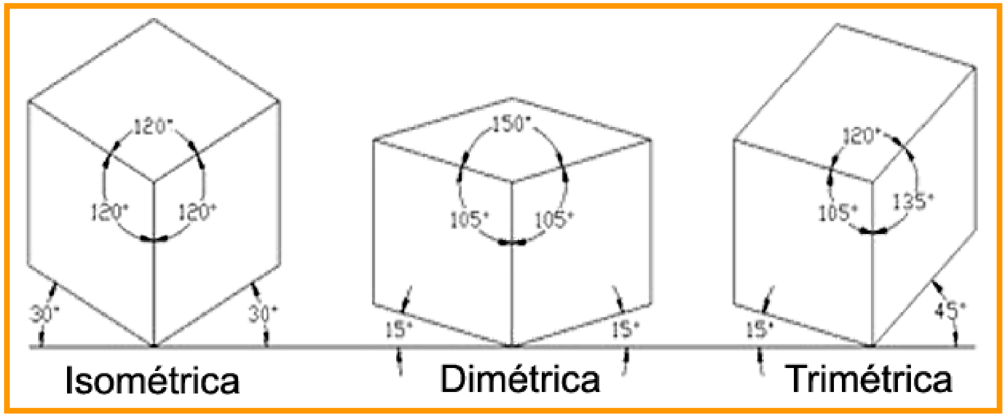
Projeções Ortográficas – Vistas

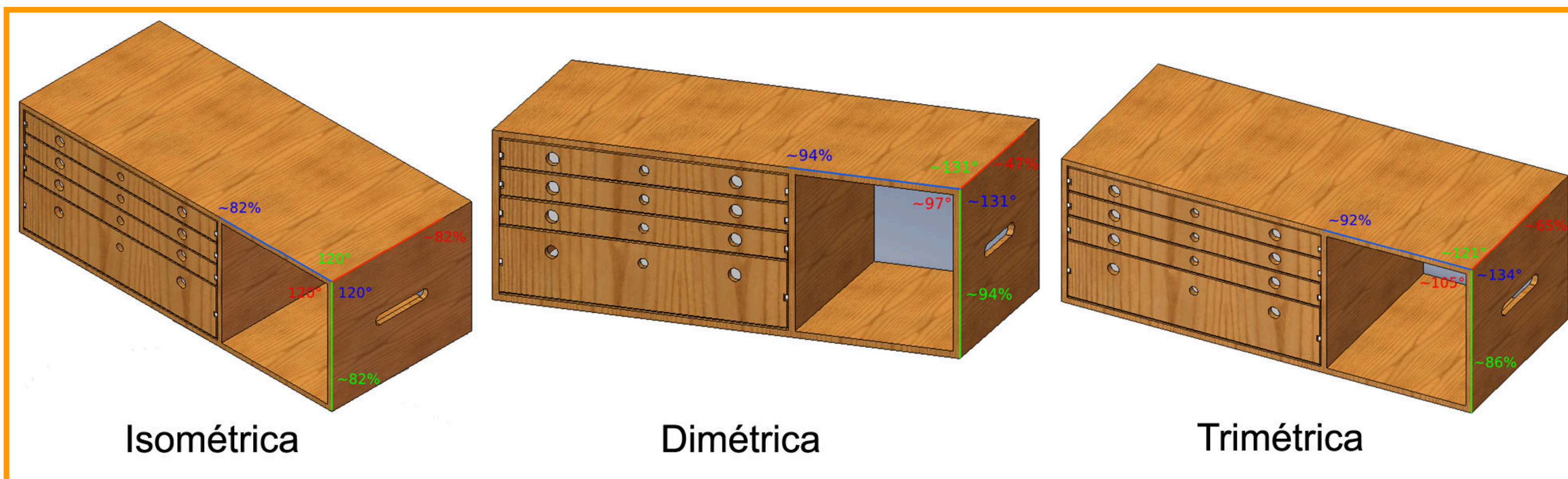
Nas vistas ortográficas a normal do plano de projeção e as projetantes coincidem com os eixos do sistema cartesiano.



Projeções Ortográficas – Axonométricas

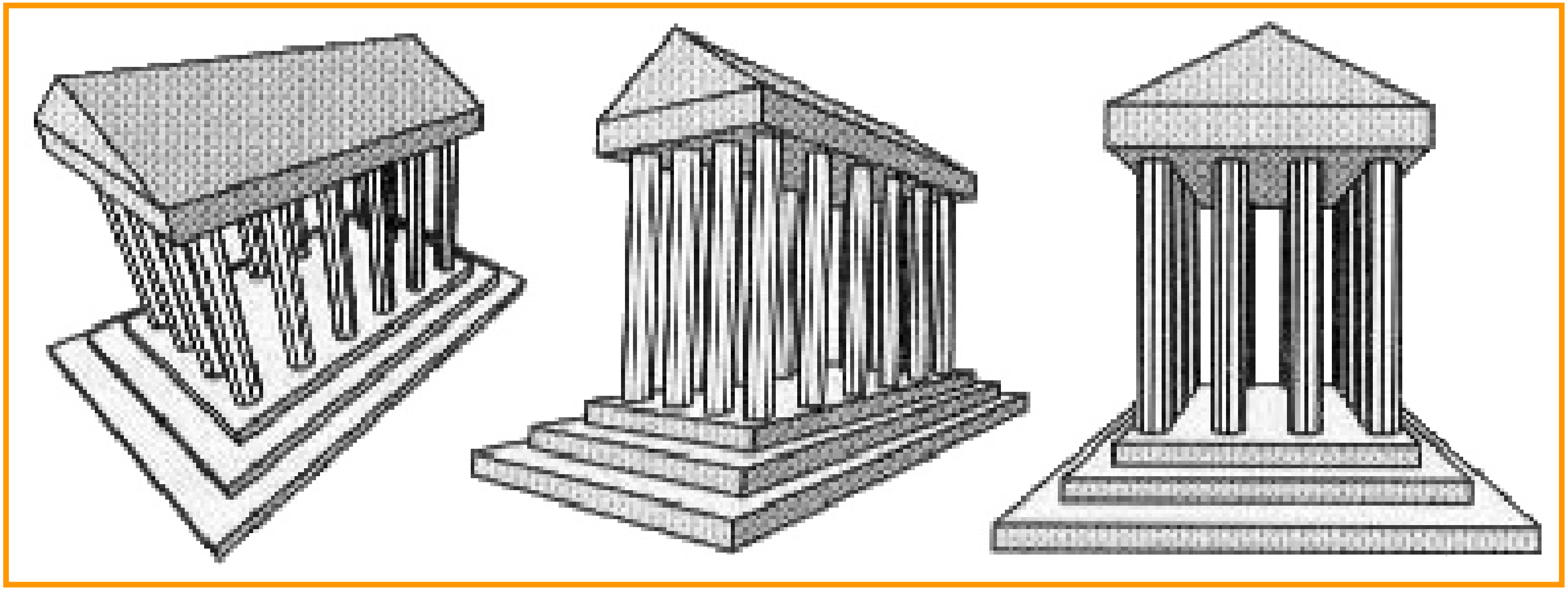
Os planos principais do objeto são oblíquos em relação ao plano de projeção.





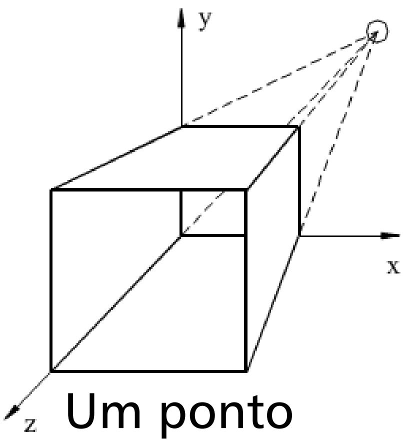
Projeções em Perspectiva

- As arestas dos objetos convergem para os pontos de fuga;
- Ocorrência do encurtamento perspectivo.

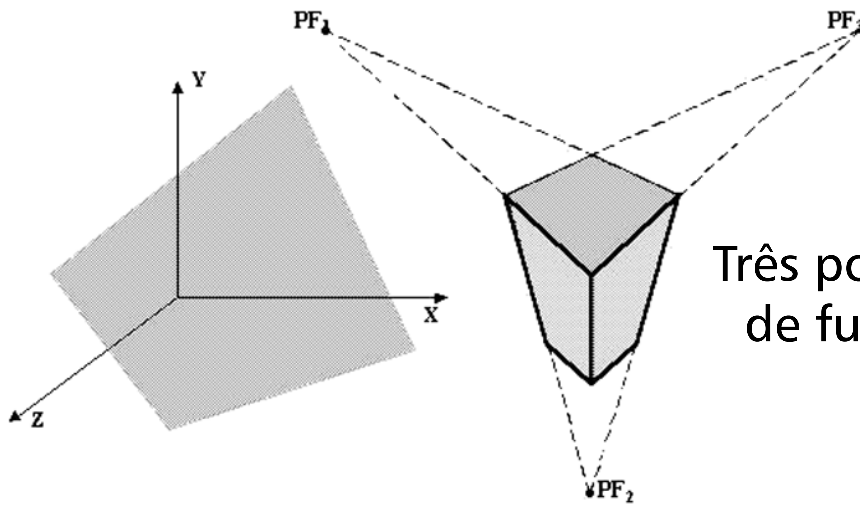


Pontos de Fuga: um, dois e três

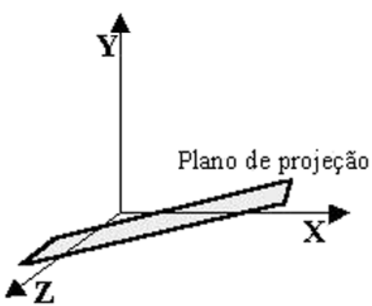
O número de eixos interceptados pelo plano de projeção determina o número de pontos de fuga.

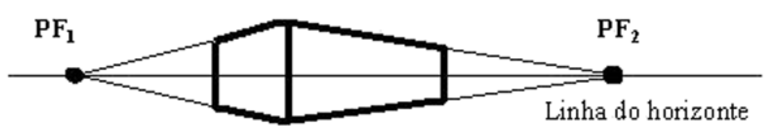


Um ponto de fuga



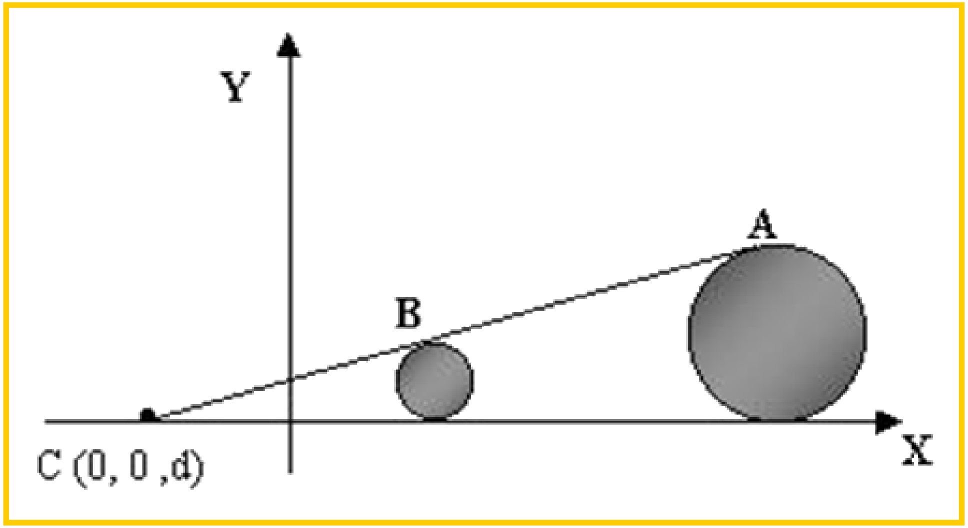
Três ponto de fuga



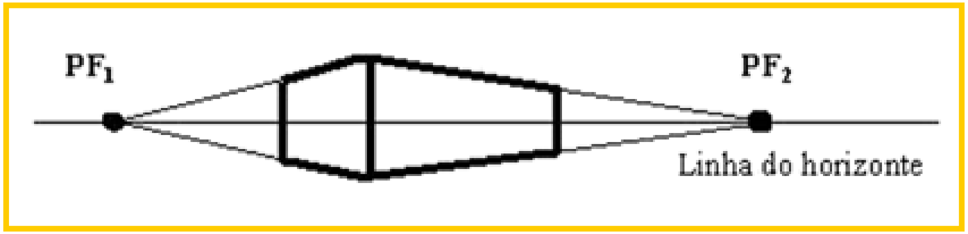


Dois pontos de fuga

Anomalias da Projeção em Perspectiva

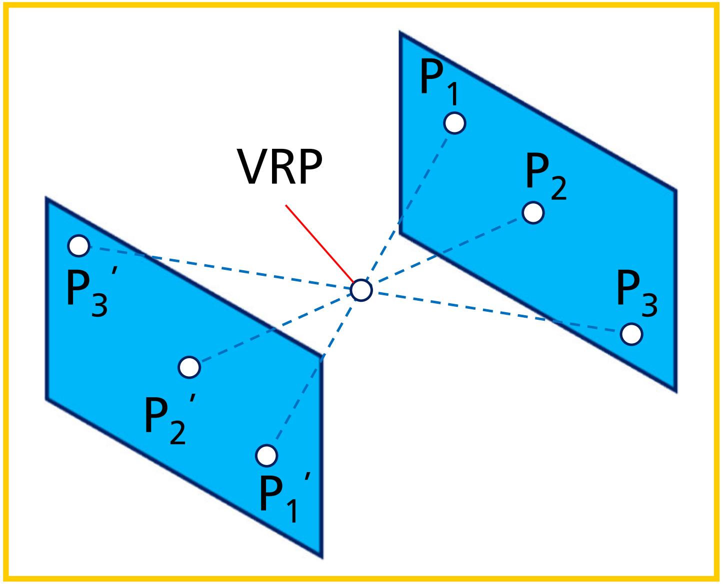


Encurtamento perspectivo

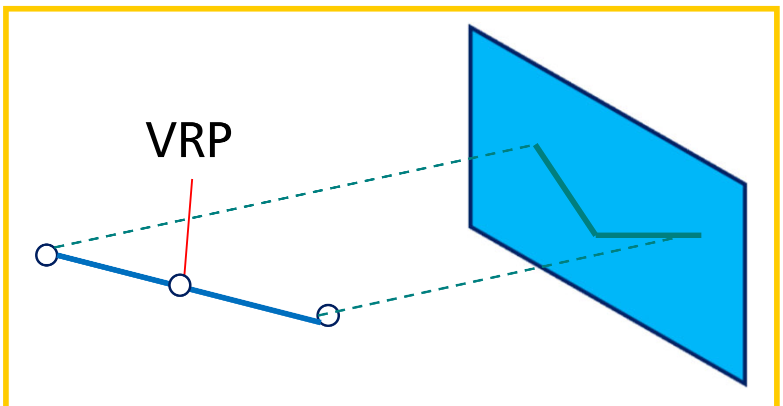


Pontos de fuga

Anomalias da Projeção em Perspectiva



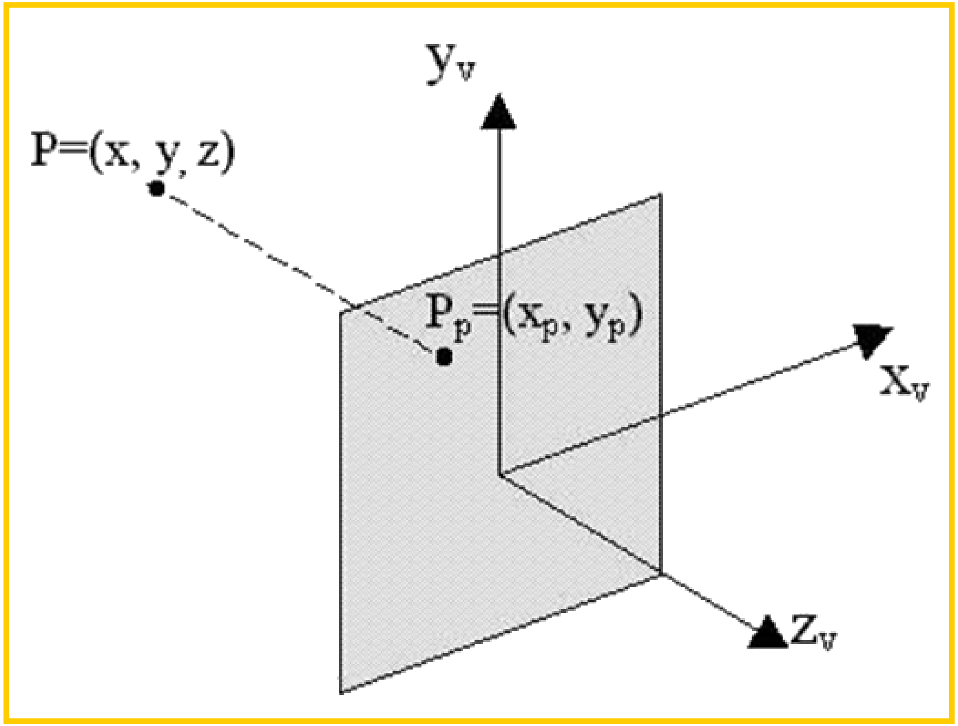
Confusão visual



Distorção topológica

Matemática das Projeções Ortográficas

O plano de projeção é perpendicular à direção de projeção.



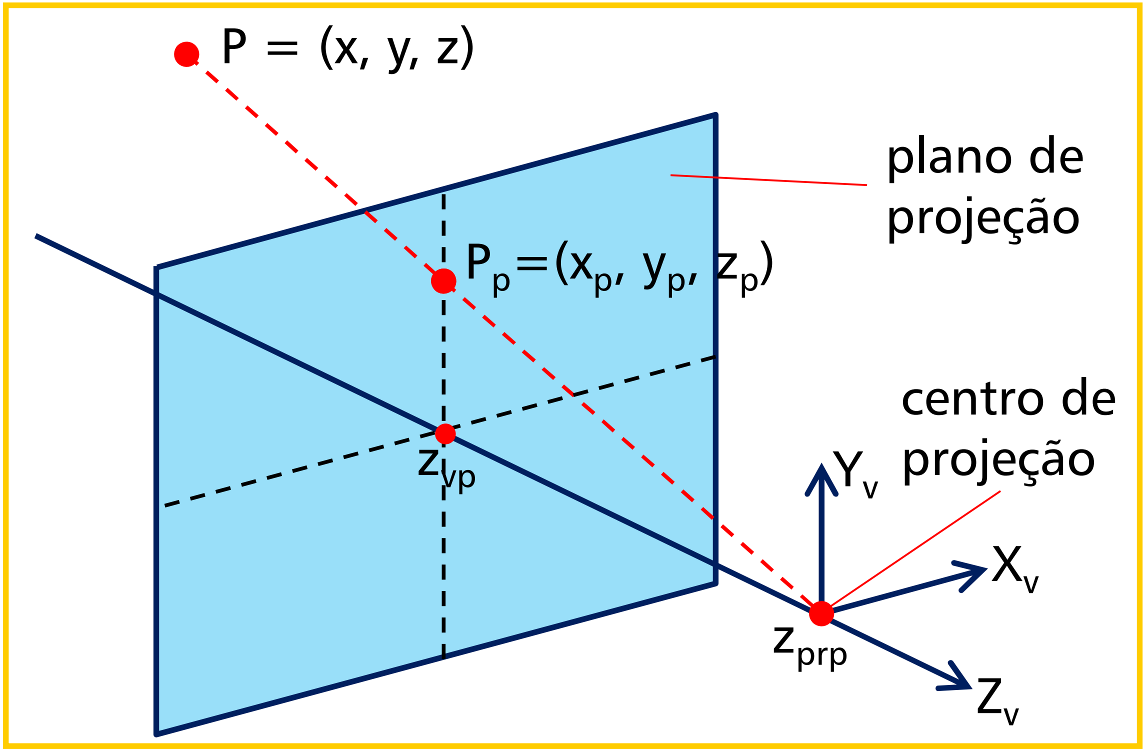
Vista Frontal

$$x_p = x, \quad y_p = y$$

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

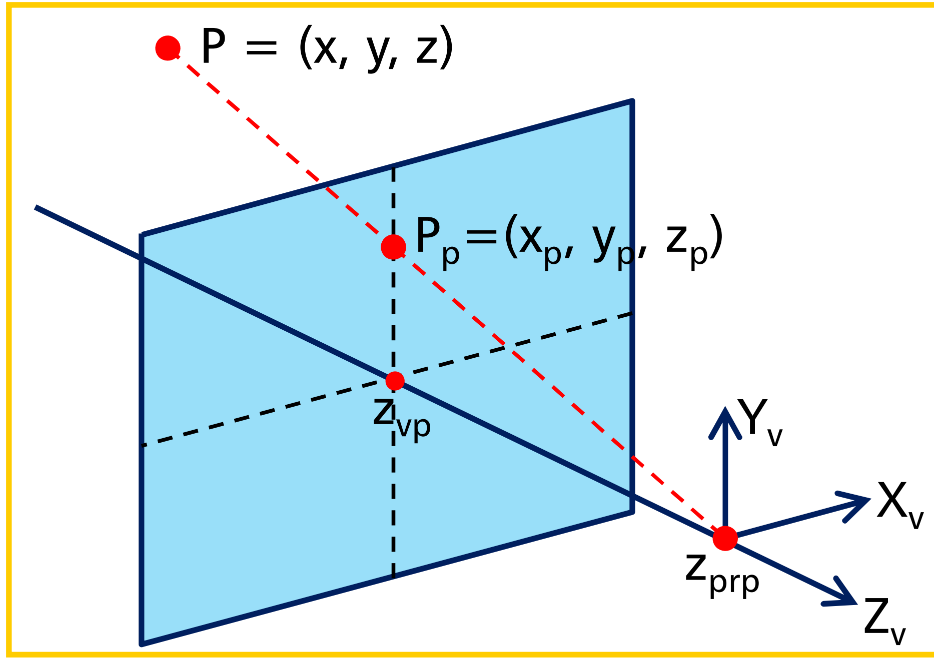
Matemática das Projeções em Perspectiva

Calcular as coordenadas dos objetos 3D, ao longo das projetantes, na interseção com o plano de projeção.



Matemática das Projeções em Perspectiva

Qualquer ponto P_p, ao longo de uma projetante, pode ser escrito como:



$$x_p = x - xu$$

$$y_p = y - yu$$

$$z_p = z - (z - z_{prp}) \cdot u$$

$$com \ \ u = [0, 1]$$

Quanto u = 0 → P = (x, y, z)

Quando u = 1 → P_p = (0, 0, z_{prp})

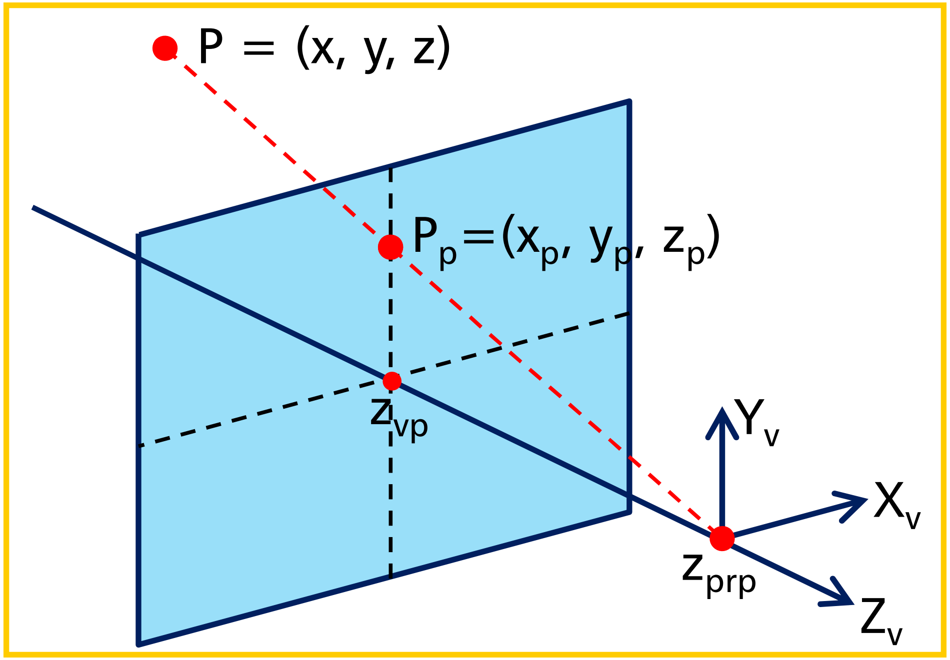
Matemática das Projeções em Perspectiva

No plano de projeção sabemos que $z_p = z_{vp}$, então:

$$z_p = z - (z - z_{prp}) \cdot u$$

$$z_{vp} = z - (z - z_{prp}) \cdot u$$

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



Substituindo *u* em x_p e y_p, temos:

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) \quad e \quad y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right)$$

$$\boxed{z_{prp} - z_{vp} = d_p}$$

Matemática das Projeções em Perspectiva

Usando coordenadas homogêneas, podemos escrever a transformação perspectiva na forma matricial.

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

O fator homogêneo é:

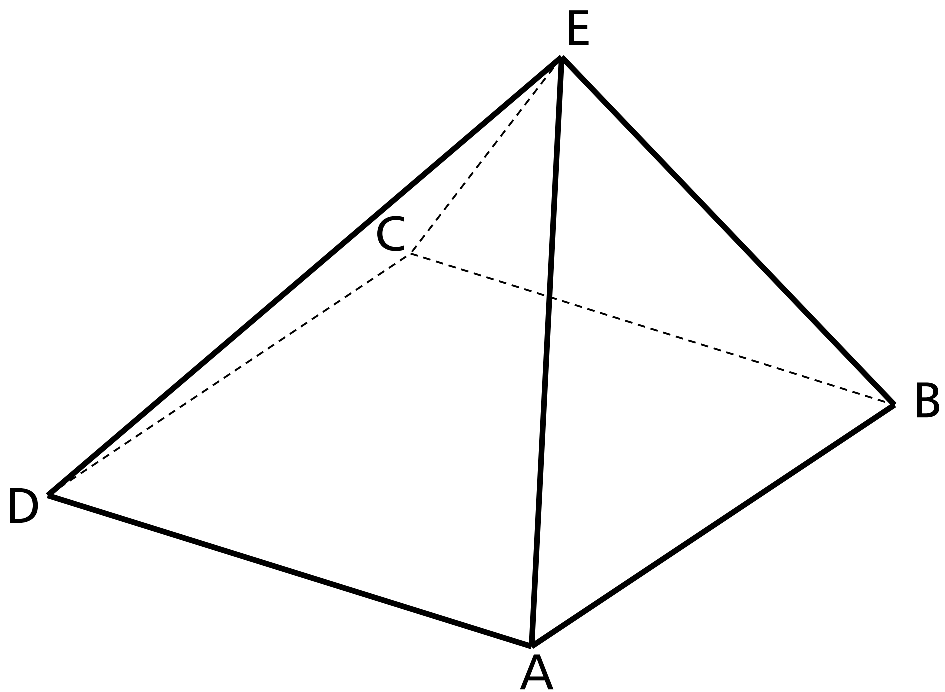
$$h = \frac{z_{prp} - z}{d_p}$$

Portanto:

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

Exercício 02 – Projeções

Considerando o exercício 01, obtenhas as coordenadas daquele objeto em projeção paralela ortográfica e perspectiva.



$A = (30, 2, 25)$

$B = (35, 4, 20)$

$C = (25, 3, 18)$

$D = (20, 1, 23)$

$E = (30, 10, 22.5)$

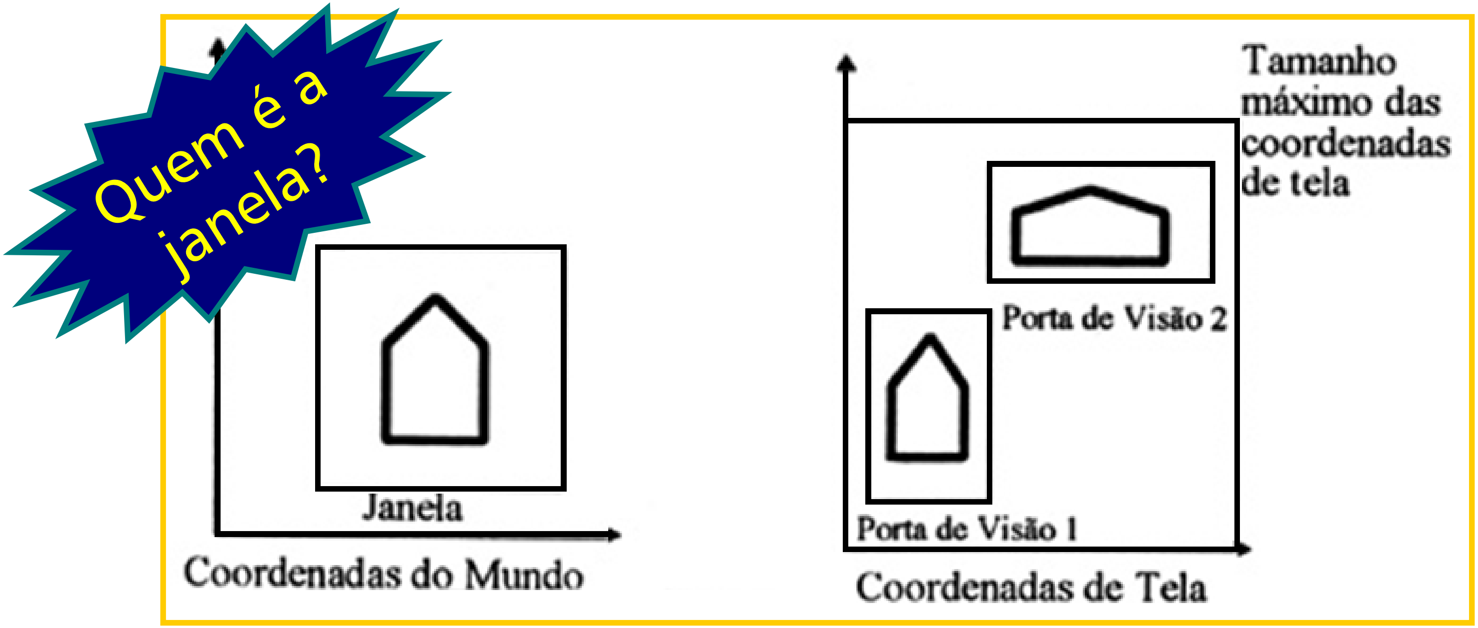
$VRP = (50, 15, 30)$

$P = (20, 6, 15)$

$d_p = 17$

Mapeamento para o SRT

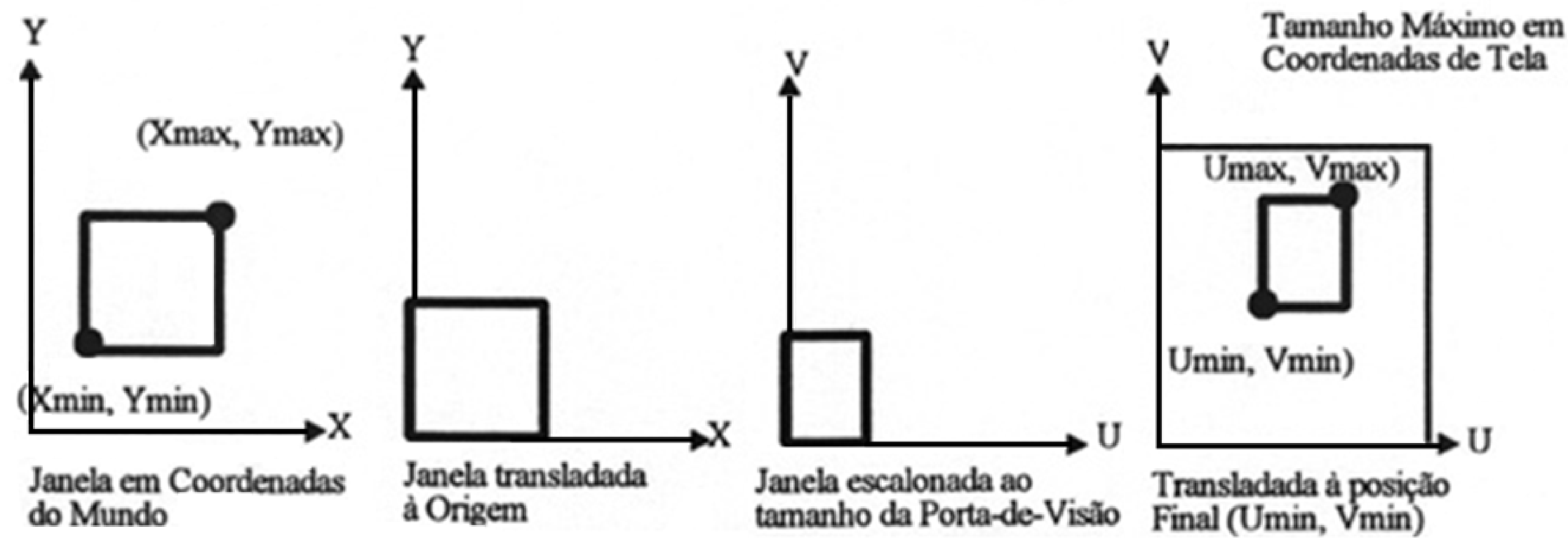
- A janela corresponde a uma porção visível do mundo;
- Porta de visão é a região da tela correspondente.



Se janela e porta de visão não possuem a mesma razão largura/altura, então ocorrem deformações nos objetos.

Transformação Janela – Porta de Visão

- Uma matriz de TG composta por 3 passos;
- Translação, Escala, Translação.



$$T\left(-x_{\min}, -y_{\min}, 0\right) \quad S\left(\frac{u_{\max}-u_{\min}}{x_{\max}-x_{\min}}, \frac{v_{\max}-v_{\min}}{y_{\max}-y_{\min}}, 1\right) \quad T\left(u_{\min}, v_{\min}, 0\right)$$

Transformação Janela – Porta de Visão

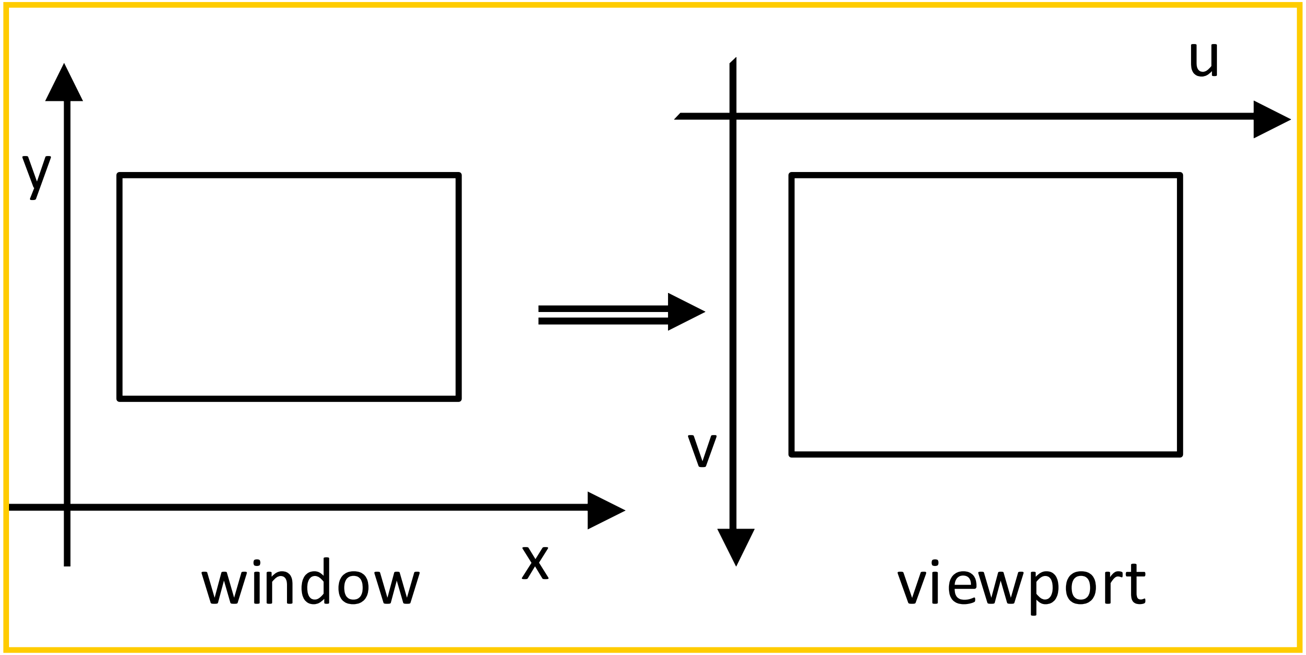
Concatenando as 3 transformações temos M_{jp}.

$$M_{jp} = T\left(u_{\min}, v_{\min}, 0\right) \cdot S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}, 1\right) \cdot T\left(-x_{\min}, -y_{\min}, 0\right)$$

$$M_{jp} = \begin{bmatrix} \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} & 0 & 0 & -x_{\min} \cdot \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} + u_{\min} \\ 0 & \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} & 0 & -y_{\min} \cdot \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} + v_{\min} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

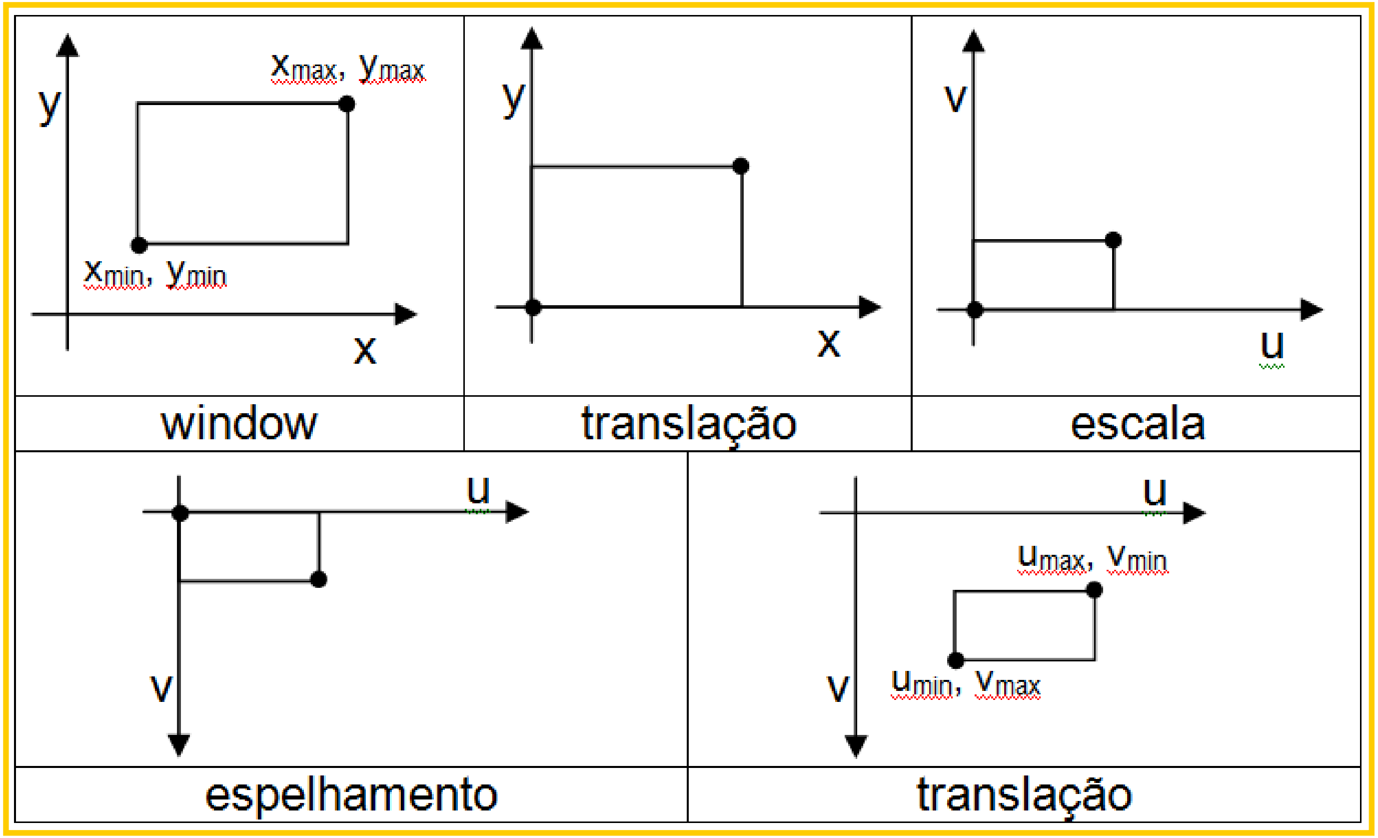
Transformação Janela – Porta de Visão

E se a viewport tem eixo v em sentido oposto ao eixo y?



Transformação Janela – Porta de Visão

Em algum momento será necessário espelhar o objeto em relação ao eixo Ox.



Transformação Janela – Porta de Visão

Concatenando as 4 transformações temos:

$$M_{jp} = T\left(u_{\min}, v_{\max}, 0\right) \cdot E\left(\overrightarrow{Ox}\right) \cdot S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}, 1\right) \cdot T\left(-x_{\min}, -y_{\min}, 0\right)$$

$$M_{jp} = \begin{bmatrix} \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} & 0 & 0 & -x_{\min} \cdot \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} + u_{\min} \\ 0 & \frac{v_{\min} - v_{\max}}{y_{\max} - y_{\min}} & 0 & y_{\min} \cdot \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} + v_{\max} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Concatenando as matrizes do *Pipeline*

Para reduzir a quantidade de operações matemáticas devemos concatenar as matrizes do *Pipeline* na ordem correta.

$$M_{SRU,SRT} = M_{jp} \cdot M_{proj} \cdot M_{SRU,SRC}$$

Assim, a transformação de um ponto no SRU diretamente para o SRT é obtida por:

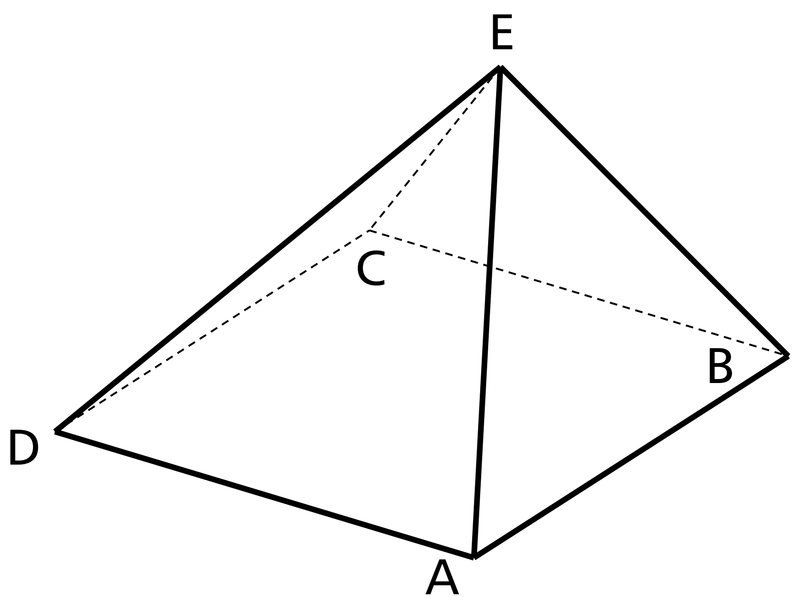
$$P' = M_{SRU,SRT} \cdot P_{SRU} \Rightarrow \begin{bmatrix} x_h \\ y_h \\ z' \\ h \end{bmatrix} = M_{SRU,SRT} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$x_{SRT} = \frac{x_h}{h}$$

$$y_{SRT} = \frac{y_h}{h}$$

Exercício 03 – Mapeamento para o SRT

Concatene as matrizes do *pipeline* e mapeie o objeto do exercício 1 para coordenadas do SRT, considerando os parâmetros abaixo.



VRP = (50, 15, 30)
P = (20, 6, 15)
d_p = 17

Janela:
- Largura = 16
- Altura = 10

Porta de visão:
- (umin, vmin) = (0, 0)
- (umax, vmax) = (320, 240)

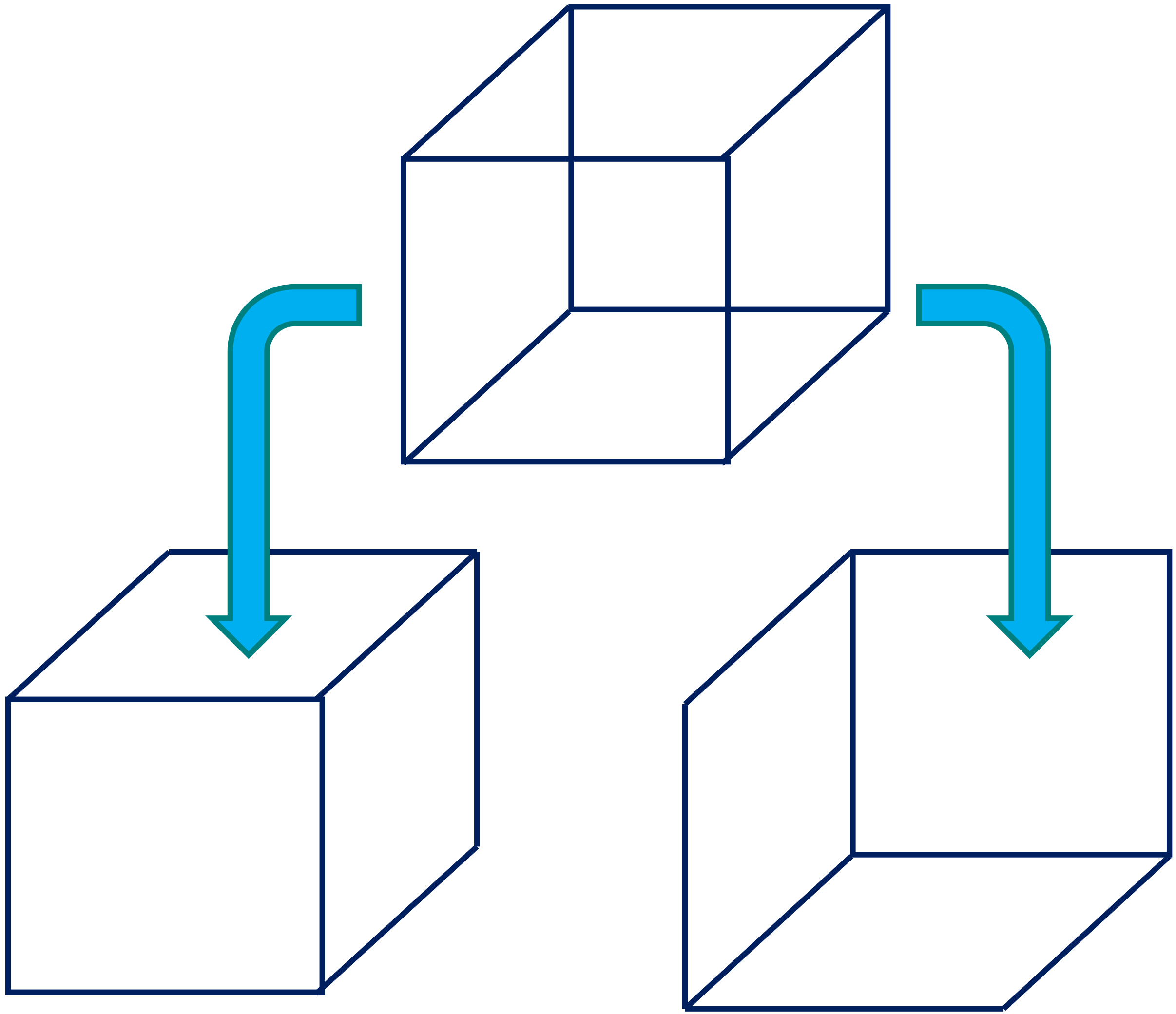
Visibilidade

O processo de determinação da visibilidade objetiva determinar quais objetos são visíveis a partir da posição da câmera.

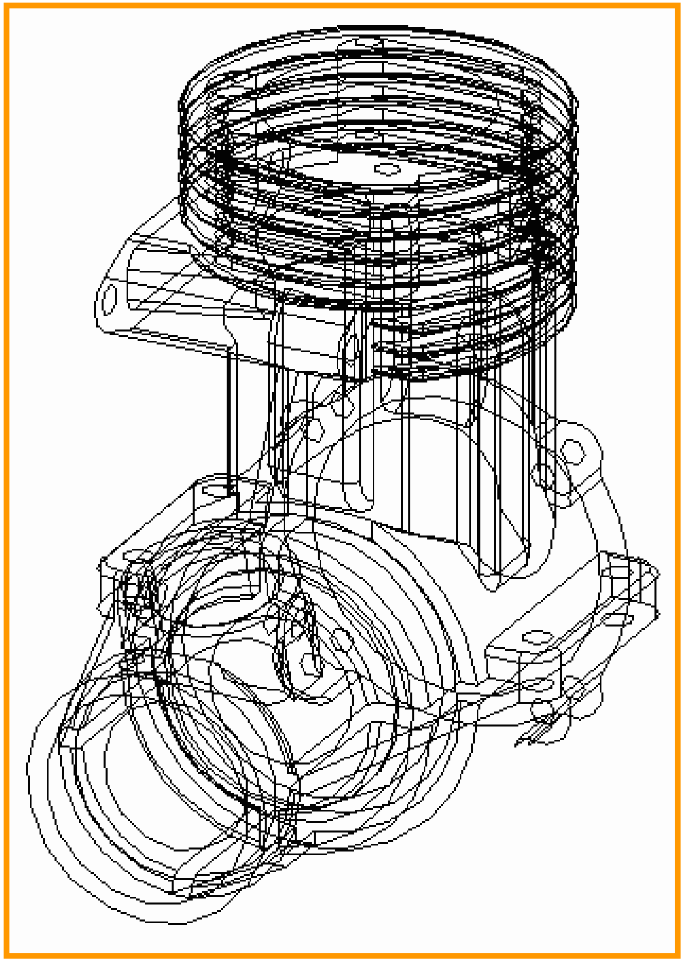
Objetos não visíveis não precisam ser desenhados, reduzindo o processamento necessário para se criar uma cena sintética.

A solução eficiente do problema da visibilidade é um dos principais passos para obtenção de imagens realistas.

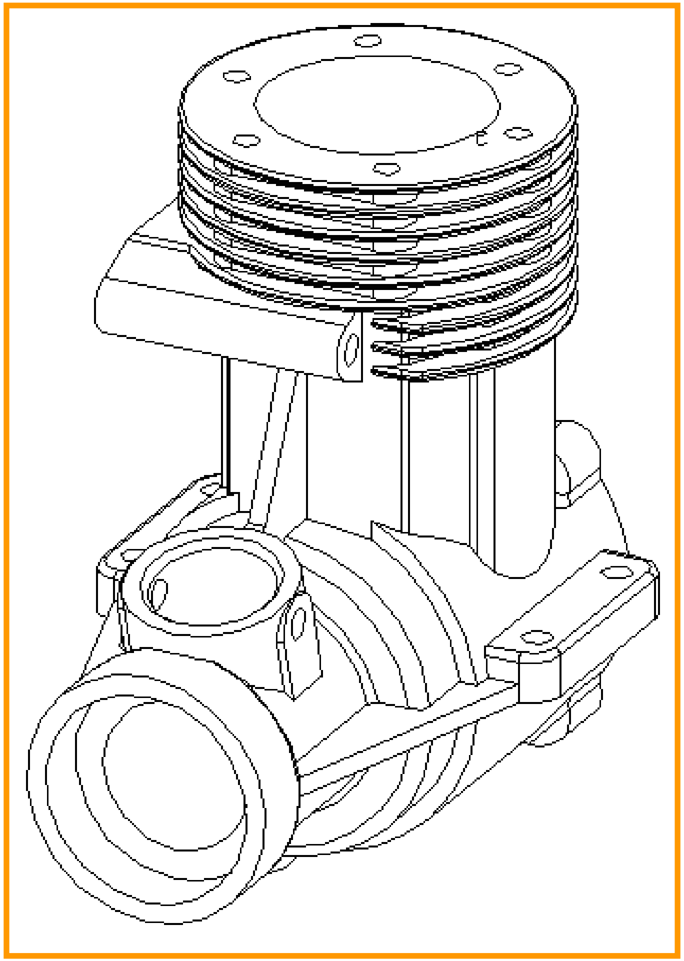
Visibilidade – Confusão Visual



Visibilidade – Confusão Visual



Sem ocultação de linhas



Com ocultação de linhas

Determinação da Visibilidade

Duas soluções:

- Representação por um subconjunto de curvas, extraídas dos contornos dos objetos.
- Representação do sólido através de uma série de faces conectadas apropriadamente.

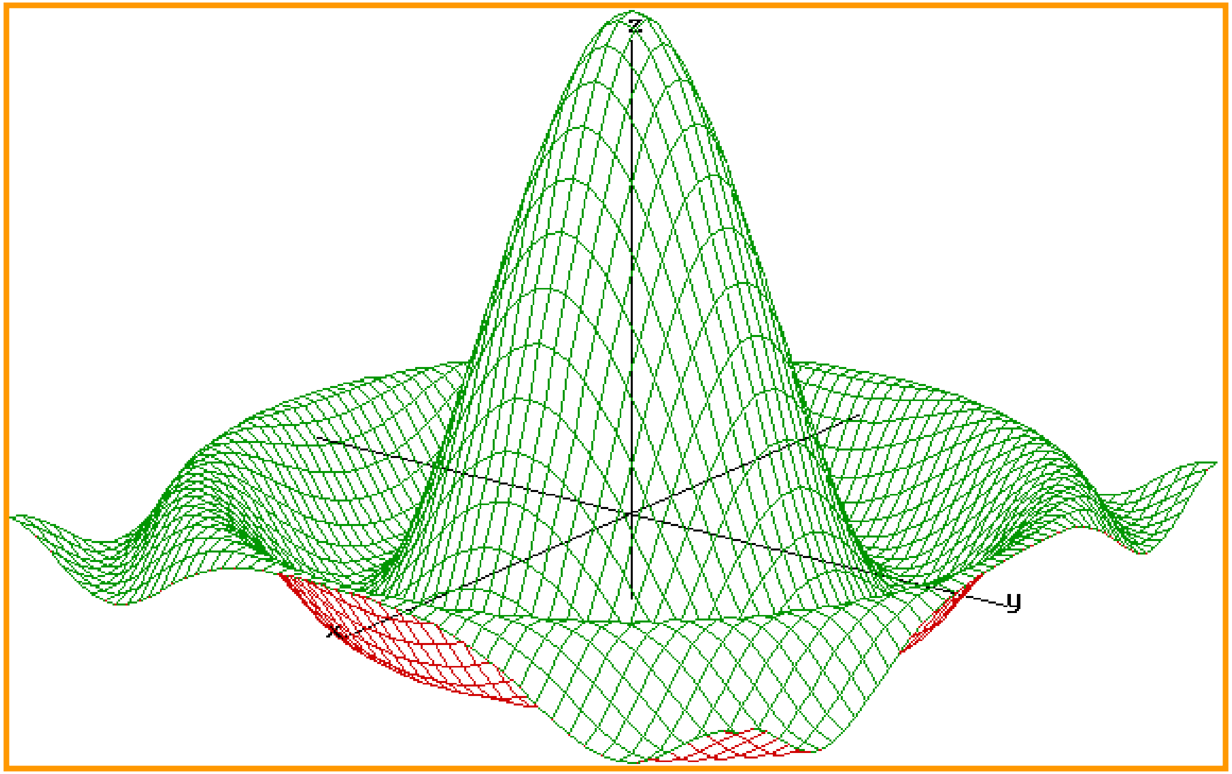
Há várias maneiras para resolver este problema:

- Algumas são simples, mas usam mais memória;
- Outras usam menos memória, mas são lentas;
- Outras são rápidas, mas limitadas.

Hoje, pela redução de custo, as técnicas baseadas no uso intensivo de memória são mais utilizadas.

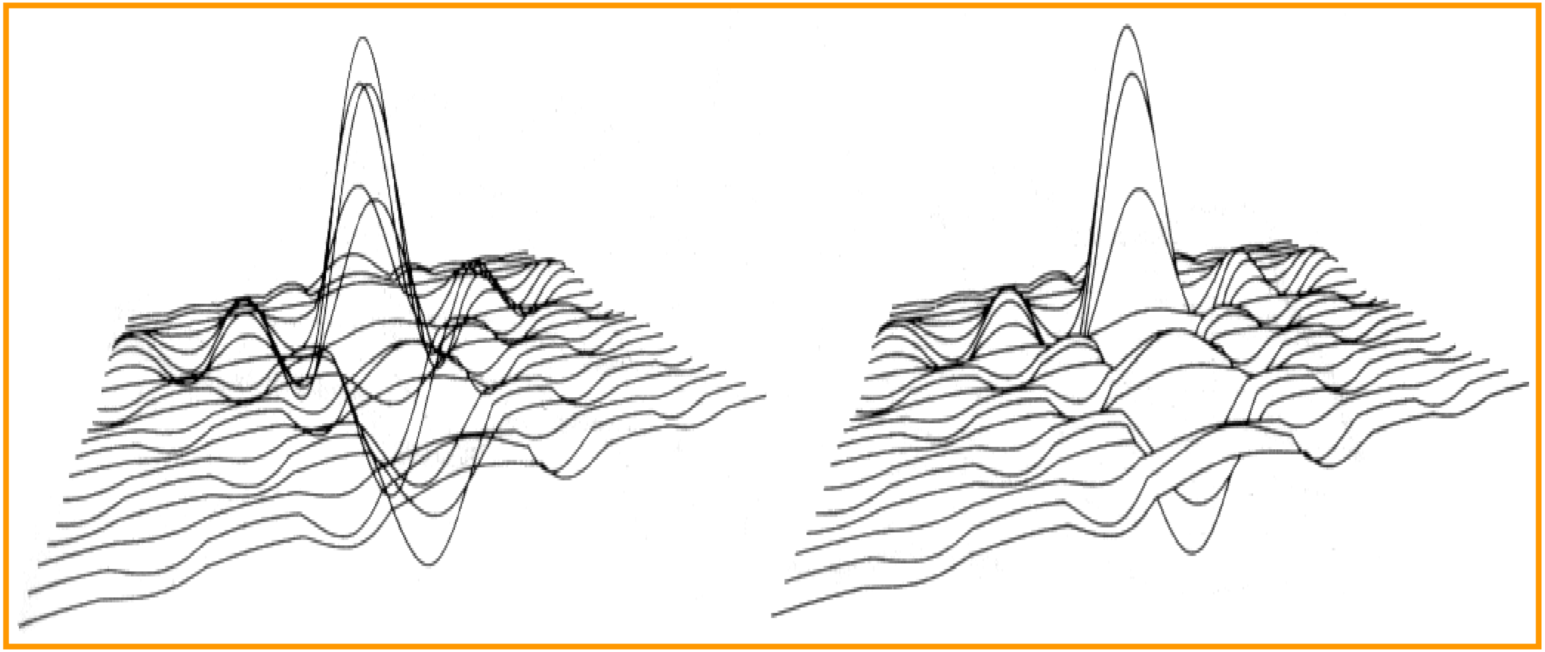
Ocultação de Linhas

Uma aplicação usual na computação gráfica é o desenho de funções contínuas a duas variáveis, como ***z=f(x, y)***.



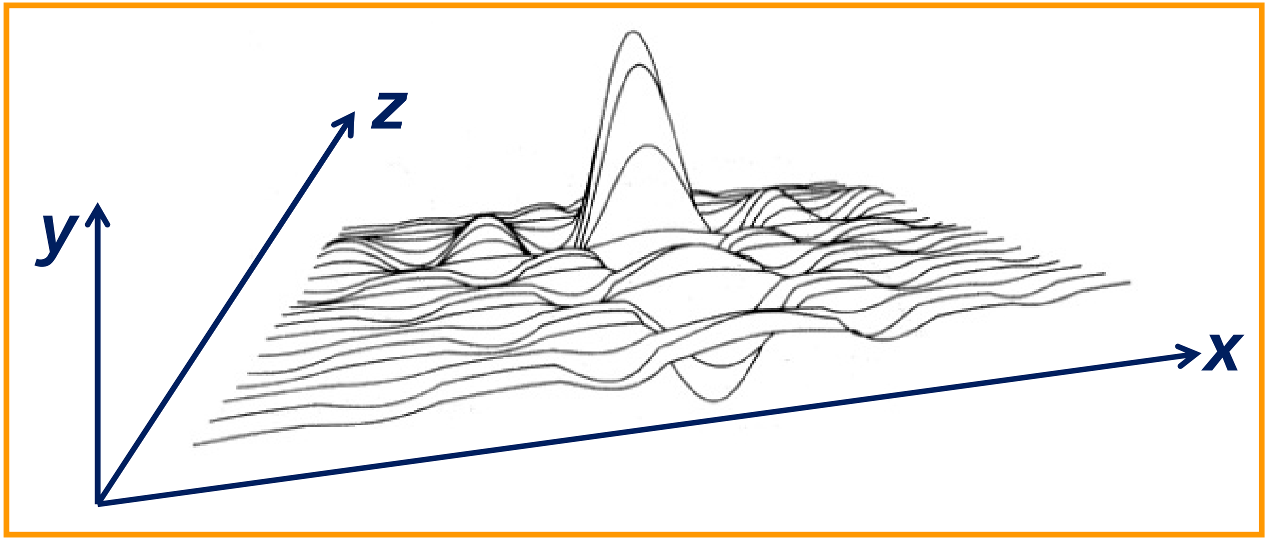
Ocultação de Linhas

Este é um caso especial de problema de ocultação de superfícies, para o qual existe solução específica e eficaz.



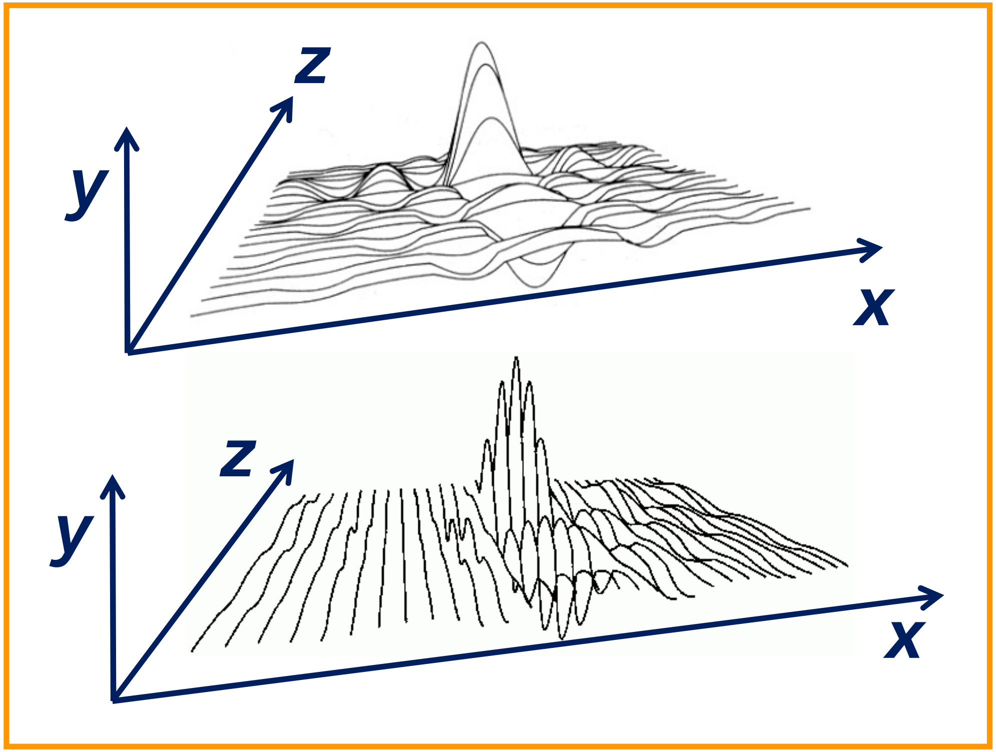
Ocultação de Linhas – Solução

- Uma função ***y = f(x, z)*** pode ser aproximada por uma matriz ***m x n*** de valores ***y***;
- Cada ponto da matriz é a altura de um ponto ***(x, z)*** do reticulado que é a base do gráfico;
- ***x*** → colunas; ***z*** → linhas.



Ocultação de Linhas – Solução

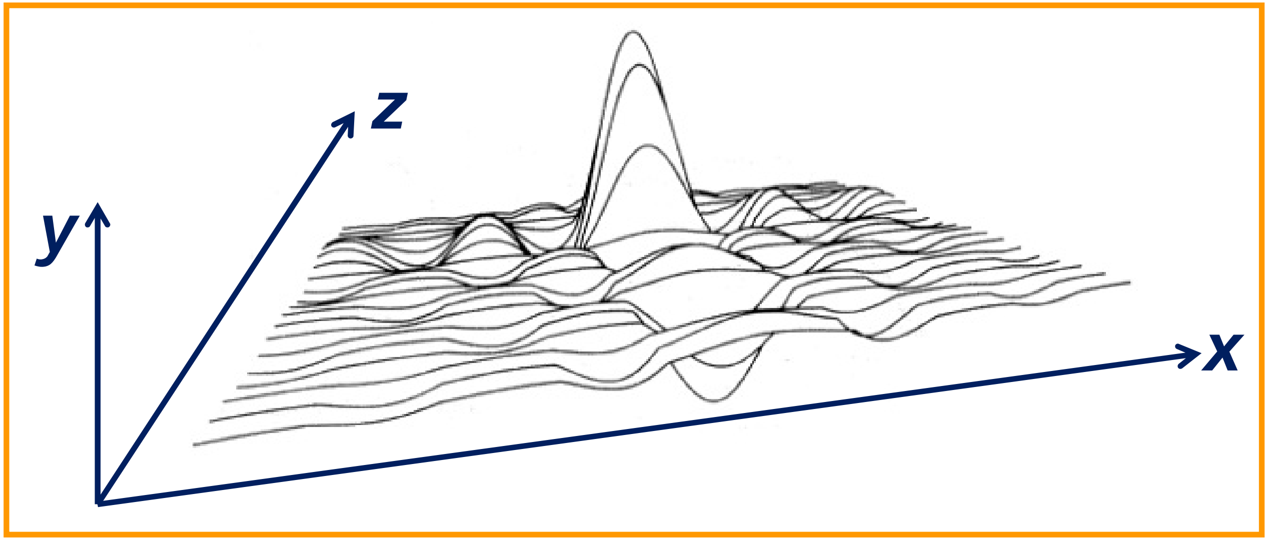
Para uma representação *wireframe* devemos desenhar dois conjuntos de *polylines*. Um conjunto de linhas com ***z*** fixos e outro de colunas com ***x*** fixos.



Ocultação de Linhas – Solução

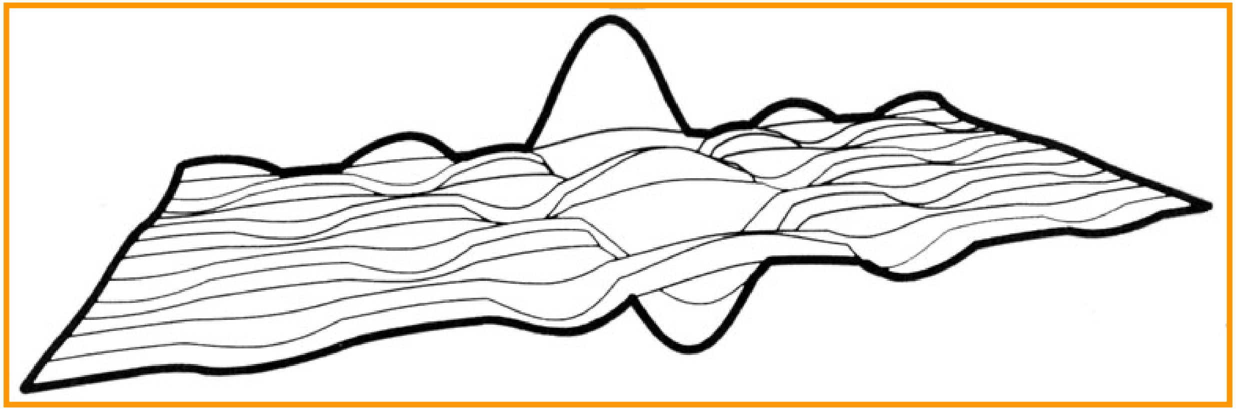
O algoritmo de ocultação de linhas deve suprimir todas as partes da superfície que estão atrás de outras partes.

Consideremos o problema de desenhar somente as *polylines* com z constantes. A *polyline* mais próxima do observador está sobre uma aresta.



Ocultação de Linhas – Solução

Desenhar as *polylines* de coordenada *z* constante na ordem "frente-para-fundo", pois aquelas mais afastadas não ocultam as mais próximas do observador.



Usando a silhueta de controle, desenhar apenas as partes de cada *polyline* que não são ocultadas por partes desenhadas anteriormente.

Algoritmo da Linha Horizontal

Utiliza uma estrutura de dados para representar o mínimo e o máximo da coordenada ***y*** para cada coordenada ***x*** da silhueta.

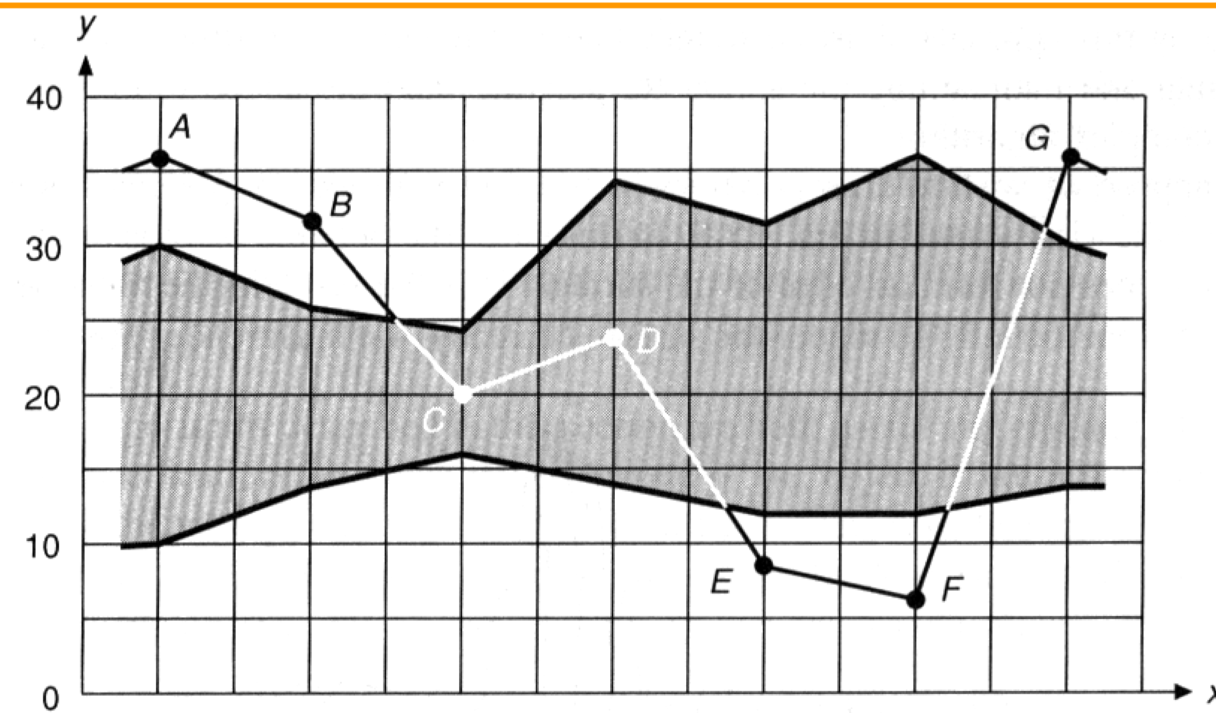
Abordagem imagem-precisão

Dois vetores (*YMIN* e *YMAX*) para armazenar os limites da silhueta, em coordenadas de projeção, para um conjunto finito de coordenadas ***x***. Para aumentar a precisão aumentamos o conjunto de coordenadas ***x***.

Abordagem objeto-precisão

Substituição dos vetores *YMIN* e *YMAX* por duas *polylines* (listas). As partes desenhadas (visíveis) são inseridas nas *polylines*. É mais precisa e complexa.

Abordagem Imagem-Precisão



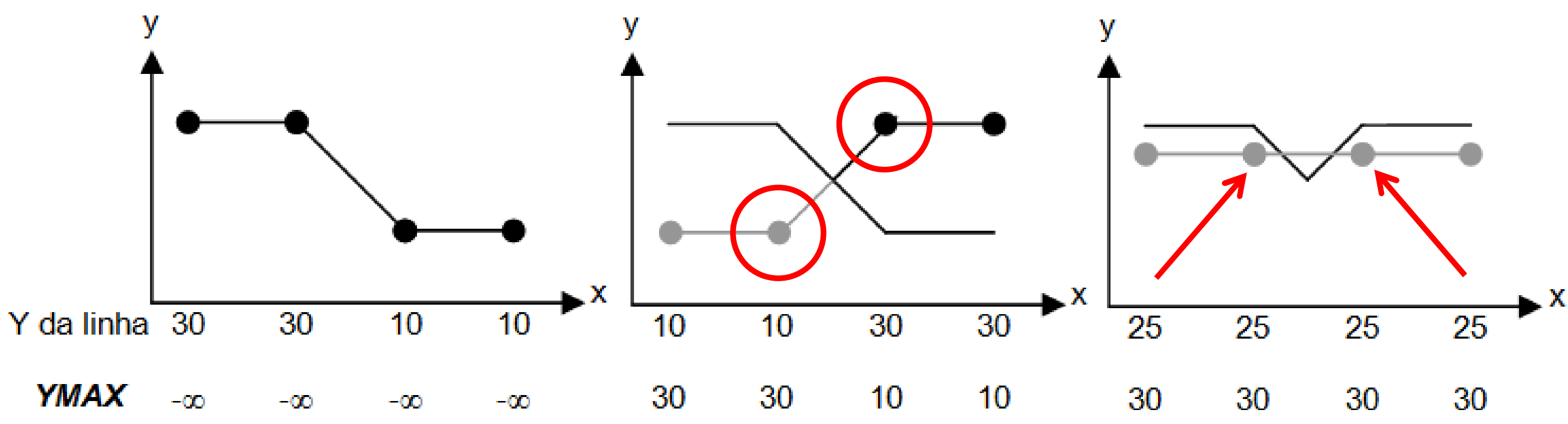
New *YMAX*	36	34	32	26	24	29	34	33	32	34	36	33	36
New *YMIN*	10	12	14	15	16	15	14	13	8	7	6	13	14
Line's *Y*	**36**	34	**32**	26	**20**	22	**24**	16	**8**	7	**6**	21	**36**
Old *YMAX*	30	28	26	25	24	29	34	33	32	34	36	33	30
Old *YMIN*	10	12	14	15	16	15	14	13	12	12	12	13	14

Acima: A, B, G. **Abaixo:** E, F **Entre:** C, D

Visíveis: AB, EF **Invisíveis:** CD **Parciais:** BC, DE, FG

Problemas da Abordagem Imagem-Precisão

Problemas de *aliasing*



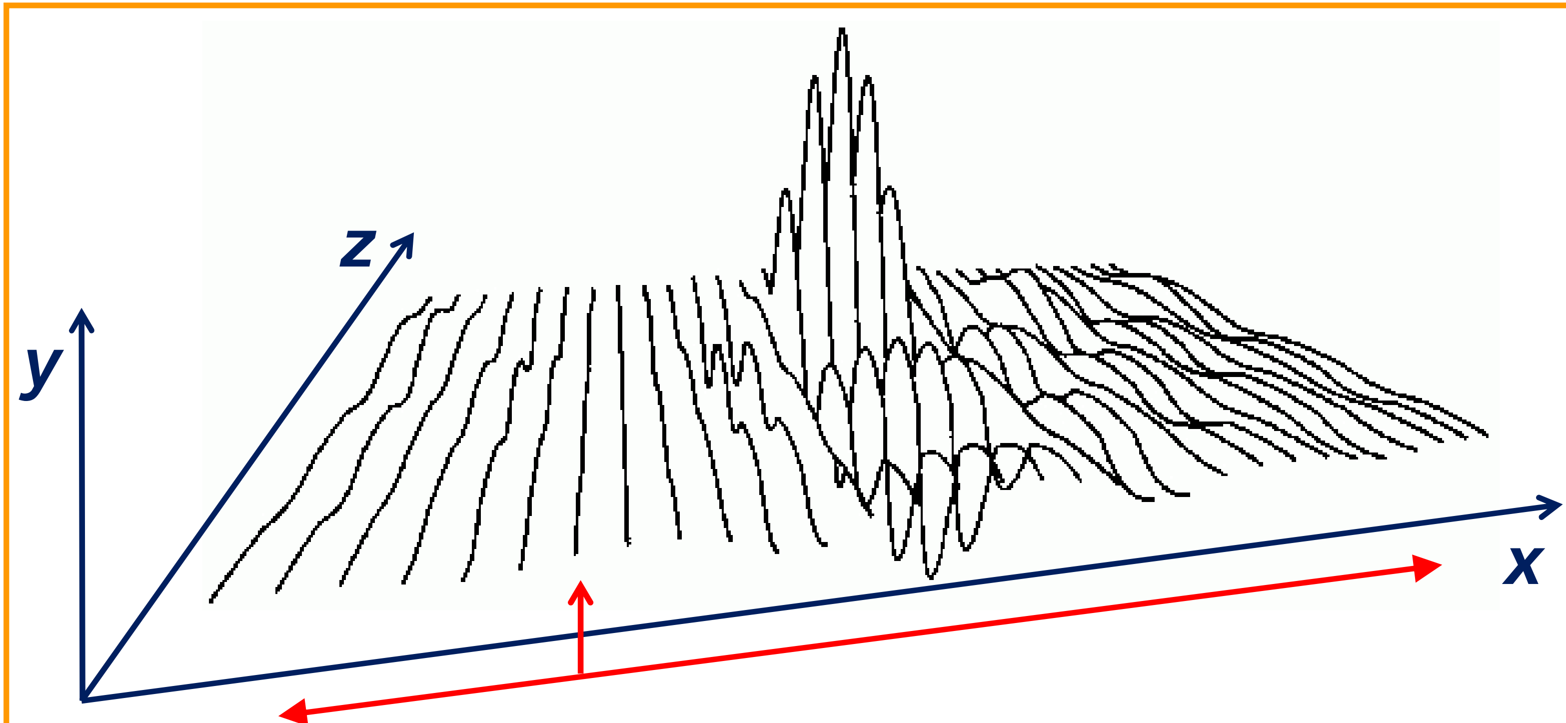
Vértices em lados opostos da silhueta (***YMAX***). Calcular o ponto de interseção e traçar o segmento visível.

Ambos os vértices estão abaixo de ***YMAX***, portanto o segmento é considerado não visível.

Solução: Abordagem objeto-precisão

Algoritmo da Linha Horizontal

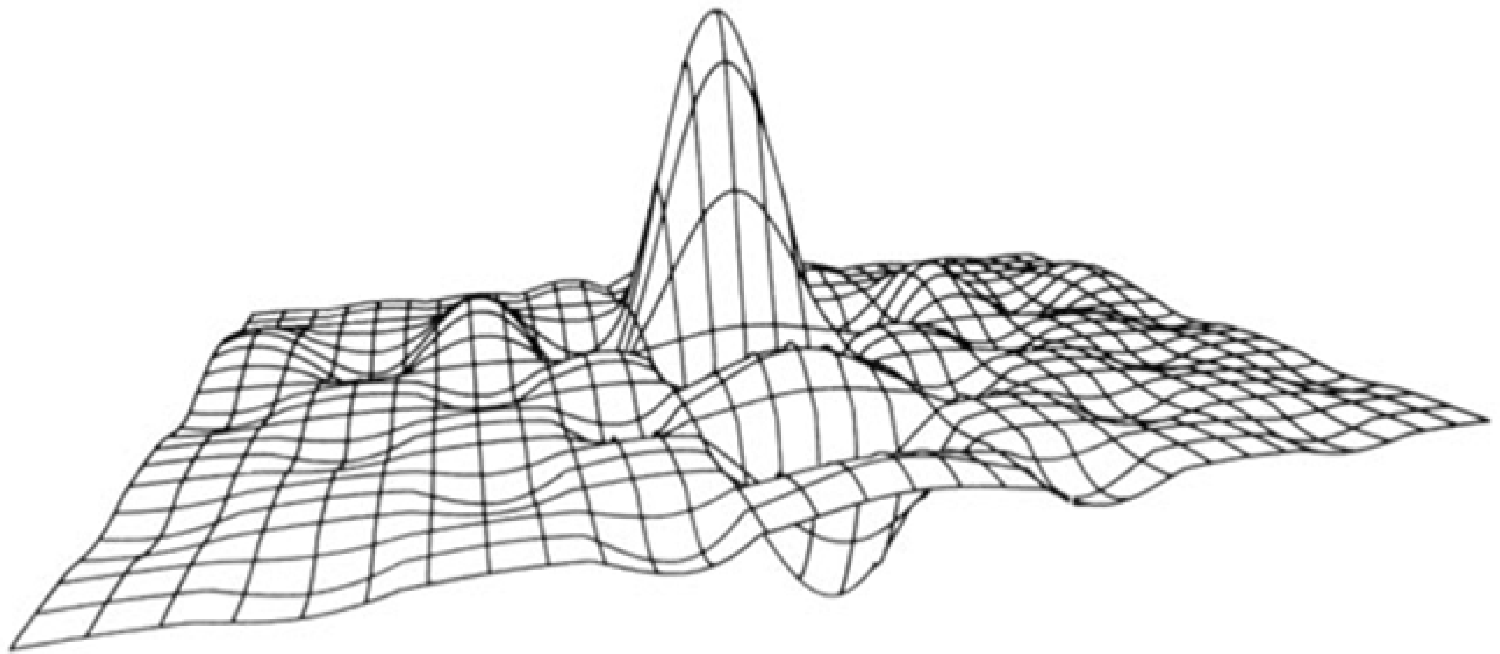
Desenhar as *polylines* com coordenada ***x*** constante.



Começar pela polyline mais próxima do observador que, neste caso, não forma uma aresta.

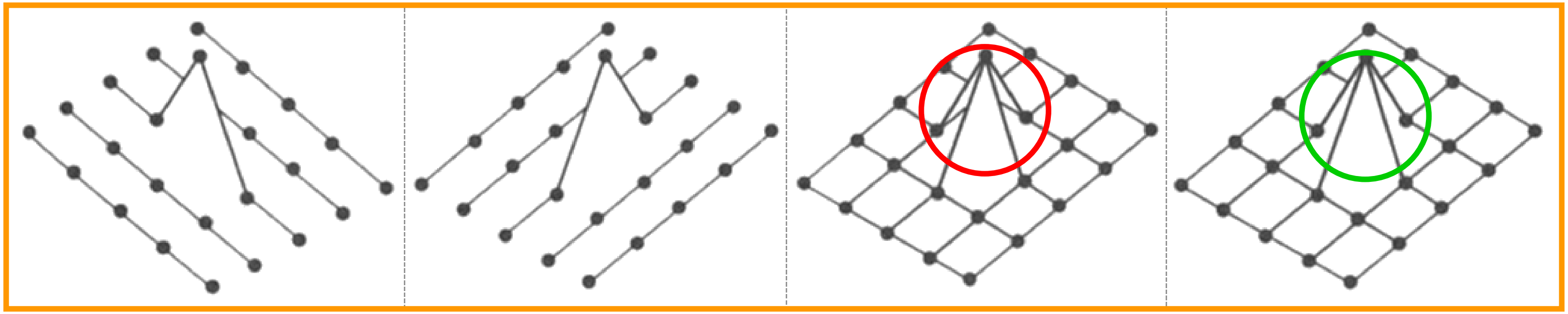
Algoritmo da Linha Horizontal

Sobreposição das *polylines*, as com coordenadas ***z*** constantes e as com coordenadas ***x*** constantes.



Algoritmo da Linha Horizontal – Problemas

Um conjunto de *polylines* não oculta as *polylines* do outro conjunto.



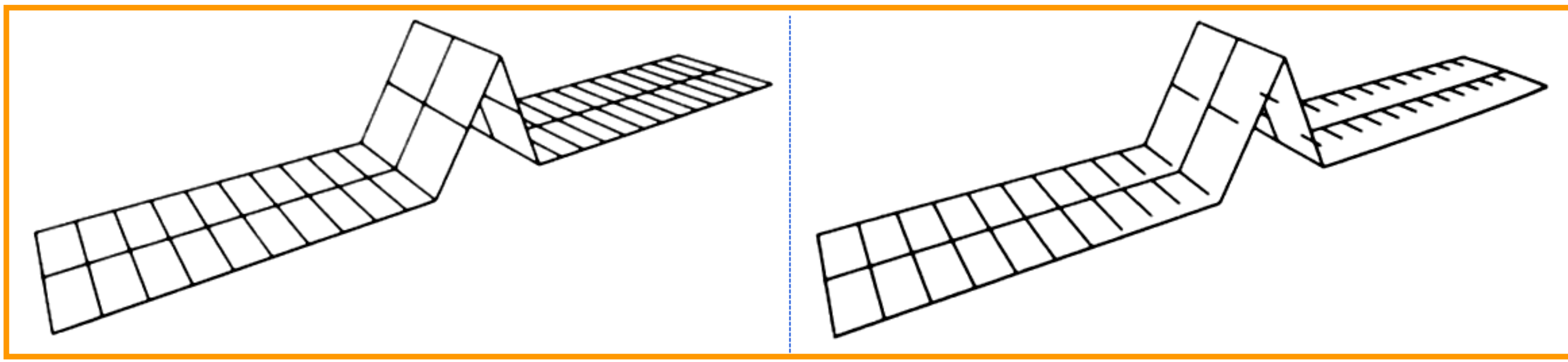
Solução

O conjunto de *polylines* mais próximas do paralelismo com o plano de projeção é processado na ordem frente-fundo (as com ***z*** constantes, no exemplo). Entre duas *polylines* com ***z*** constantes e subsequentes são traçados os segmentos das *polylines* com ***x*** constantes.

Compartilhar a estrutura de dados da silhueta.

Ordem de Processamento

O conjunto de segmentos das polylines com x constantes também devem ser processados na ordem correta (frente-fundo e esquerda/direita).



Ordem correta

Ordem incorreta

Visibilidade de Superfícies

Dois algoritmos básicos

para cada pixel da imagem faça{

- determinar o objeto mais próximo do observador que é visível pela projeção através do pixel;
- plotar o pixel na cor apropriada.

}

Algoritmo imagem-precisão

Algoritmo objeto-precisão

para cada objeto no mundo faça{

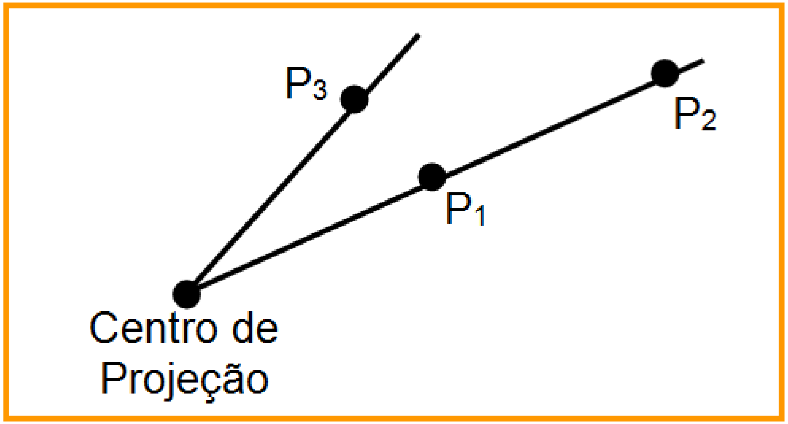
- determinar quais partes do objeto cuja visão não está ocultada por partes do mesmo ou de outro objeto;
- plotar estas partes na cor apropriada.

}

Diferenças entre os Algoritmos Básicos

Imagem-precisão	Objeto-precisão
Usa a resolução da tela;	Independente da resolução da tela;
Processa cada pixel para determinar a visibilidade;	Processa os objetos para determinar quais partes são visíveis;
Mudança no tamanho da tela implica em recálculos.	Mudança no tamanho da tela implica só no redesenho da cena.

Visibilidade – Projeção Paralela x Perspectiva



$P_1 = (x1, y1, z1)$ oculta $P_2 = (x2, y2, z2)$?

Sim. Se ambos os pontos estiverem na mesma linha de visada.

Em projeção paralela quando:

$(x_1 = x_2)$ e $(y_1 = y_2)$

Em projeção perspectiva quando:

$(x_1/z_1 = x_2/z_2)$ e $(y_1/z_1 = y_2/z_2)$

Paralela

Perspectiva Canônica

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+z_{min}} & \frac{-z_{min}}{1+z_{min}} \\ 0 & 0 & -1 & 0 \end{bmatrix}, z_{min} = \frac{n}{f} \neq -1$$

Visibilidade por Prioridade

Princípio

Quando o objeto ***A*** bloqueia a visão do objeto ***B***, com ambos na mesma linha de visada do observador, então ***B*** está mais distante que ***A***.

Calcular as distâncias entre os objetos e o observador. Ordenar os objetos de acordo com a distância calculada, apresentando-os na tela em ordem decrescente.

Algoritmo do Pintor

Cálculo da Distância – Algoritmo do Pintor

Calcular a distância euclidiana entre o observador (VRP) e o centróide do objeto (CO).

$$D = \sqrt{(VRP_x - CO_x)^2 + (VRP_y - CO\,y)^2 + (VRP_z - CO_z)^2}$$

Considerando que o observador está na origem do SRC, temos:

$$D = \sqrt{CO_x^2 + CO_y^2 + CO_z^2}$$

Cálculo da Distância – Algoritmo do Pintor

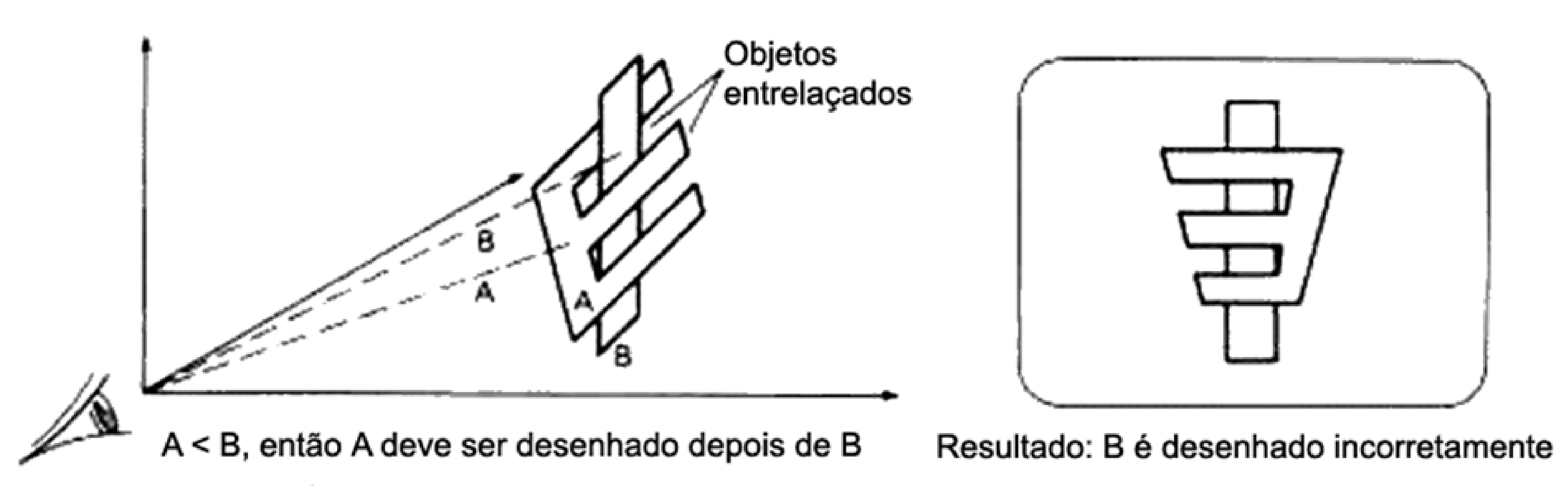
A radiciação é computacionalmente cara; então é preferível utilizar expressões trigonométricas para calcular a distância.

$$h = \frac{CO_x}{\cos\left(arctg\left(\frac{CO_y}{CO_x}\right)\right)}$$

$$D = \frac{CO_x}{\cos\left(arctg\left(\frac{CO_x}{h}\right)\right)}$$

Algoritmo do Pintor – Problemas

A ordenação pela distância aos centróides apresenta limitações.

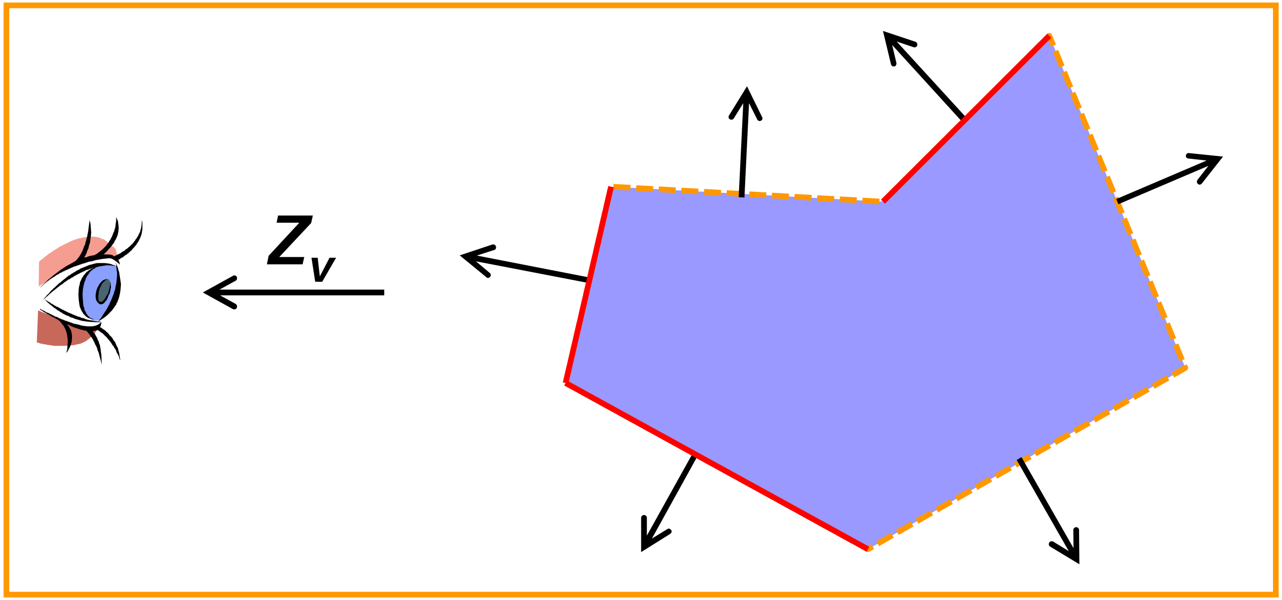


Solução

Dividir um objeto côncavo em vários objetos convexos.

Visibilidade pelo Cálculo da Normal

Ao observarmos uma superfície não podemos observar seu lado oposto.



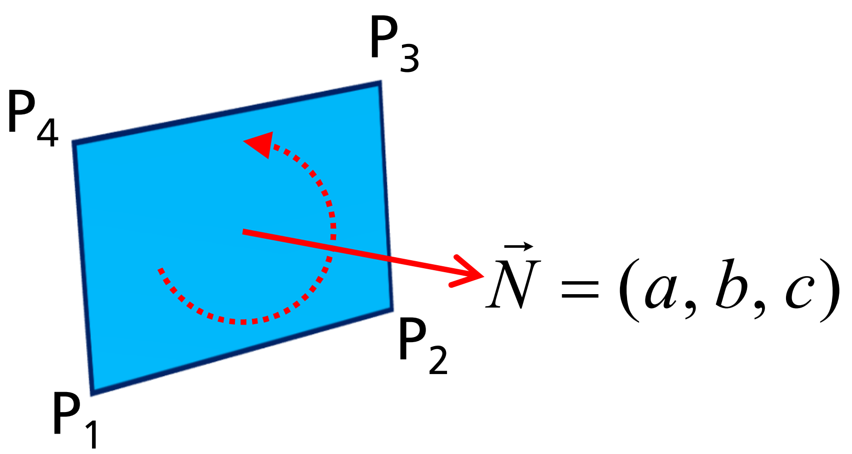
O lado visível de uma superfície é aquele cuja normal está voltada para o lado onde se encontra o observador.

Determinando a Equação de um Plano

Para obtermos o vetor normal a uma superfície determinamos a equação do plano que contém esta superfície.

$$a \cdot x + b \cdot y + c \cdot z + d = 0$$

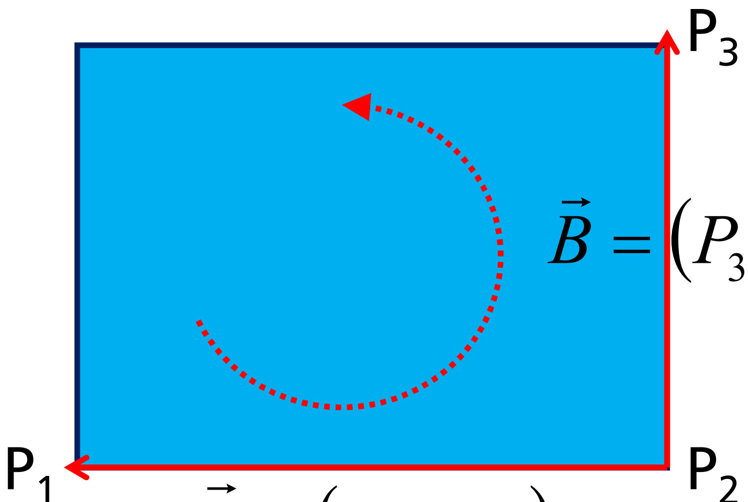
O vetor normal ao plano é dado por:



$$\vec{N} = (a, b, c)$$

Determinando a Equação de um Plano

Dados 3 vértices (P_1, P_2 e P_3), em sentido anti-horário, que pertencem a superfície, temos:



$$\vec{B} = (P_3 - P_2)$$

$$\vec{A} = (P_1 - P_2)$$

Portanto:

$$\vec{N} = \vec{B} \times \vec{A}$$

$$\vec{N} = \begin{vmatrix} i & j & k \\ B_x & B_y & B_z \\ A_x & A_y & A_z \end{vmatrix} = (a, b, c)$$

Determinando a Equação de um Plano

De outra forma:

$$\vec{N} = \begin{vmatrix} i & j & k \\ (P_3x - P_2x) & (P_3y - P_2y) & (P_3z - P_2z) \\ (P_1x - P_2x) & (P_1y - P_2y) & (P_1z - P_2z) \end{vmatrix} = (a, b, c)$$

$$a = (P_3y - P_2y)\cdot(P_1z - P_2z) - (P_1y - P_2y)\cdot(P_3z - P_2z)$$
$$b = (P_3z - P_2z)\cdot(P_1x - P_2x) - (P_1z - P_2z)\cdot(P_3x - P_2x)$$
$$c = (P_3x - P_2x)\cdot(P_1y - P_2y) - (P_1x - P_2x)\cdot(P_3y - P_2y)$$

Qual a Posição Relativa do Observador?

Aplicando um dos $\boldsymbol{P_i}$ na equação do plano

$$a \cdot x + b \cdot y + c \cdot z + d = 0$$

Temos:

$$d = -(a \cdot P_2 x) - (b \cdot P_2 y) - (c \cdot P_2 z)$$

Substituindo a posição do observador (VRP) na equação do plano temos:

$$D = a \cdot VRP_x + b \cdot VRP_y + c \cdot VRP_z + d$$

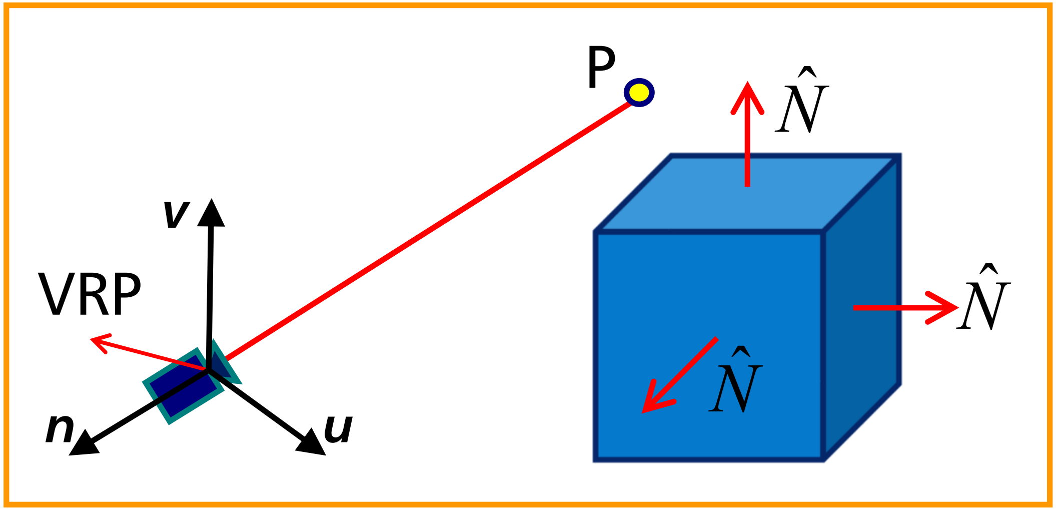
Se D = 0 → O observador é parte do plano;

Se D > 0 → O observador está à frente do plano;

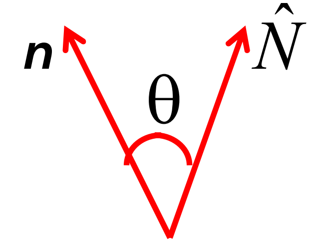
Se D < 0 → O observador está atrás do plano.

A Superfície é Visível?

A visibilidade de uma superfície depende da posição do observador (VRP) e da direção de projeção.



Interpretação



$$\cos(\theta) > 0$$

$$\text{p}/\ 0 \le \theta < \frac{\pi}{2}$$

Aplicamos o teste: $n \cdot \hat{N} > 0$

Se o teste for satisfeito então a superfície é visível.

Visibilidade pelo Cálculo da Normal

Este método é adequado para poliedros convexos fechados, como cubos, prismas, cilindros e esferas, por exemplo.

Não resolve o problema quando um objeto oculta outro objeto. Ainda é necessário ordenar as faces visíveis de acordo com a distância ao observador.

Utilizado como pré-filtro, pode eliminar mais da metade das superfícies a serem processadas em outros métodos de determinação da visibilidade, reduzindo o tempo de processamento em aproximadamente 75%.

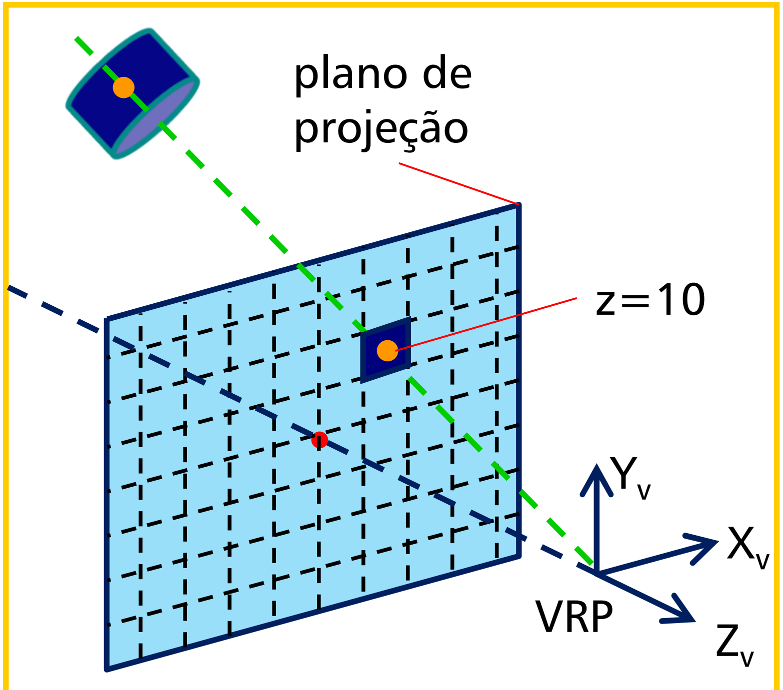
Algoritmo *Z-Buffer*

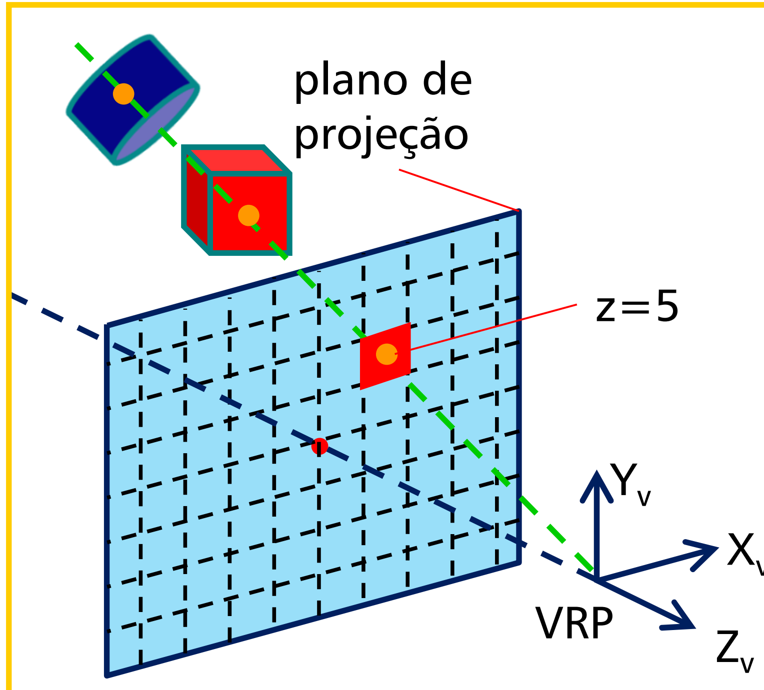
Método desenvolvido por Catmull (1974). Fácil de implementar em software ou hardware.

Implementação baseada no uso intensivo de memória. Requer dois *bufffers* de dimensões idênticas à tela de apresentação.

- O *buffer* de imagem, também chamado de rascunho, tem profundidade de cor igual à tela de apresentação. É inicializado com a cor de fundo.
- O *z-buffer* armazena a profundidade associada ao objeto visível em cada pixel. É inicializado com um valor que represente a máxima distância de um ponto ao plano de projeção (ou à câmera).

Algoritmo *Z-Buffer*





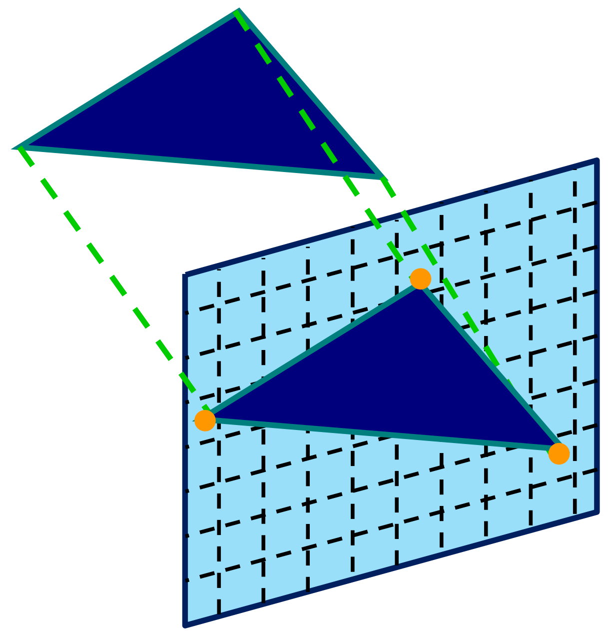
Os objetos são projetados pixel a pixel. Para cada pixel projetado verificamos sua profundidade em relação ao valor armazenado no *z-buffer*.

- Se for menor, calculamos a cor do pixel e atualizamos o buffer de imagem, bem como o *z-buffer*.

Algoritmo *Z-Buffer* Incremental

A cada novo pixel projetado, ao longo de uma superfície plana, devemos comparar sua profundidade com o valor armazenado no *z-buffer*.

Como acelerar o cálculo da profundidade do pixel?

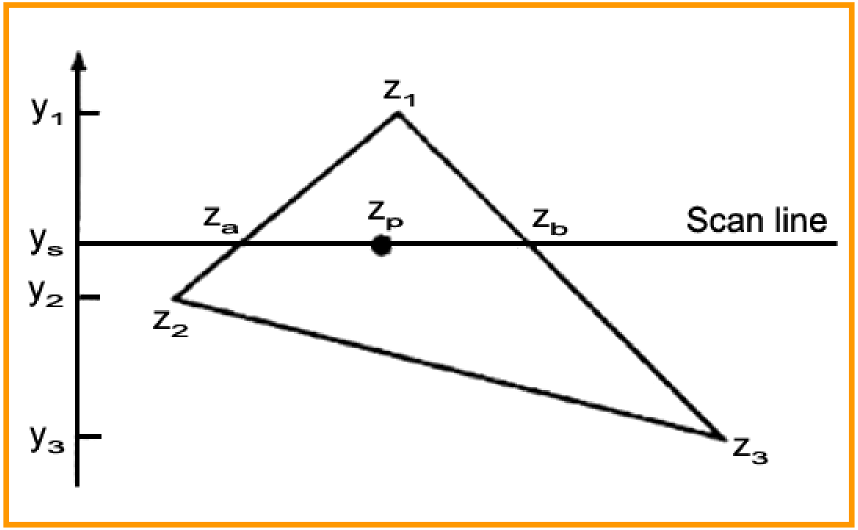


A equação do plano que contém a superfície a ser projetada é:

$$ax + by + cz + d = 0$$

Agora consideremos uma projeção paralela desta superfície ($x_p = x$, $y_p = y$).

Algoritmo *Z-Buffer* Incremental



Dada a equação do plano:

$$ax + by + cz + d = 0$$

Assim, $z_a(x_a, y_s)$ é:

$$z_a = \frac{-d - ax_a - by_s}{c}$$

e $z_{a+1}(x_a+1, y_s)$ é:

$$z_{a+1} = \frac{-d - a(x_a + 1) - by_s}{c}$$

Algoritmo *Z-Buffer* Incremental

Então, a diferença de profundidade entre z_a e z_{a+1} é:

$$z_{a+1} - z_a = \frac{-d - a(x_a + 1) - by_s}{c} - \frac{-d - ax_a - by_s}{c}$$

Ou seja, para $\Delta x = 1$:

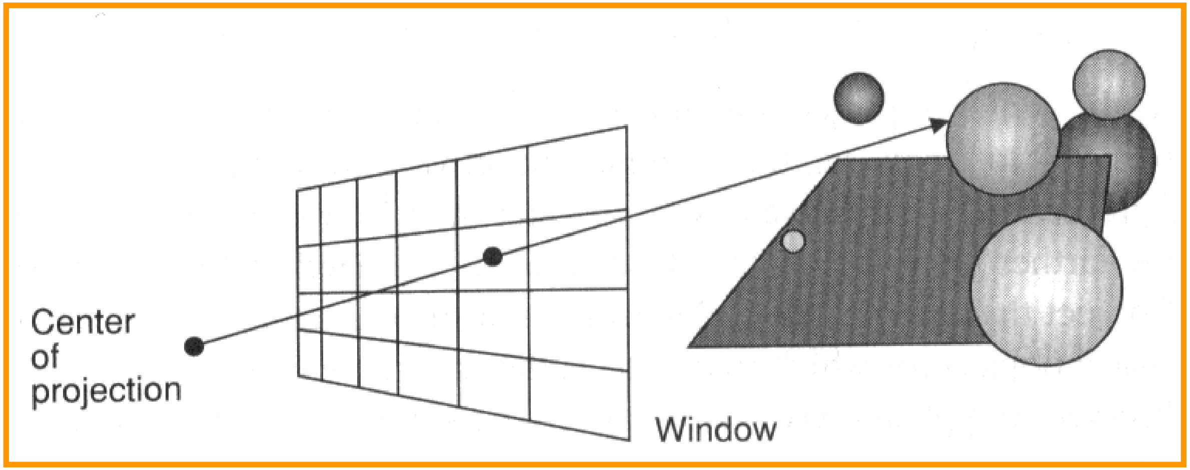
$$\Delta_z = -\frac{a}{c}$$

Assim, o cálculo completo da profundidade é realizado apenas no primeiro ponto da *scan line*. Para cada novo pixel (x_i, y_s), a profundidade será incrementalmente calculada por:

$$z_i = z_{i-1} - \frac{a}{c}$$

Ray Tracing

É um algoritmo imagem-precisão, o qual determina a visibilidade das superfícies pelo traçado de um raio de luz imaginário, que parte do olho do observador até os objetos da cena.



O pixel assume a cor do objeto interceptado que está mais próximo do centro de projeção.

Ray Tracing

O núcleo de qualquer *ray tracer* é a tarefa de determinar a interseção do raio com os objetos.

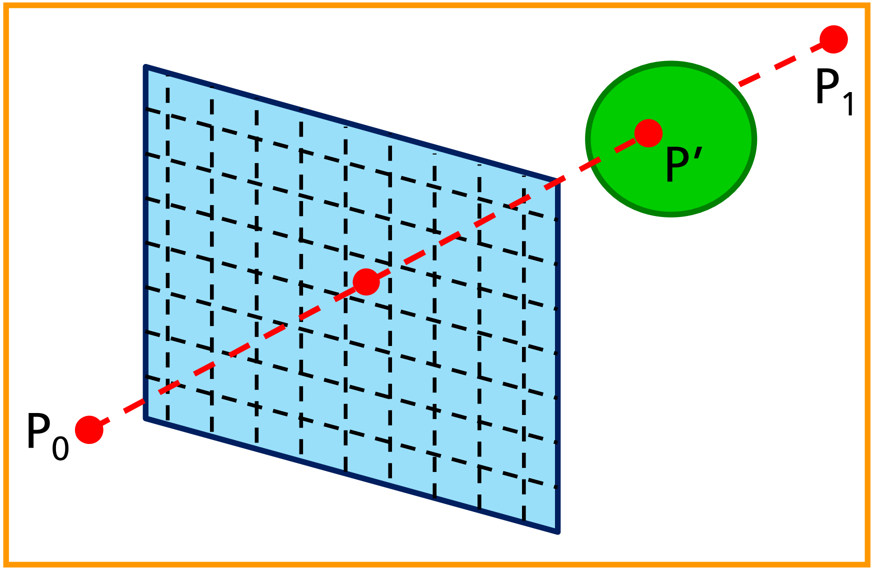
Cada ponto ***P=(x, y, z)*** ao longo do raio entre ***P_0=(x_0, y_0, z_0)*** e ***P_1=(x_1, y_1, z_1)*** pode ser escrito em função de um parâmetro ***t[0, 1]***, tal que:

$$x = x_0 + t(x_1 - x_0)$$

$$y = y_0 + t(y_1 - y_0)$$

$$z = z_0 + t(z_1 - z_0)$$

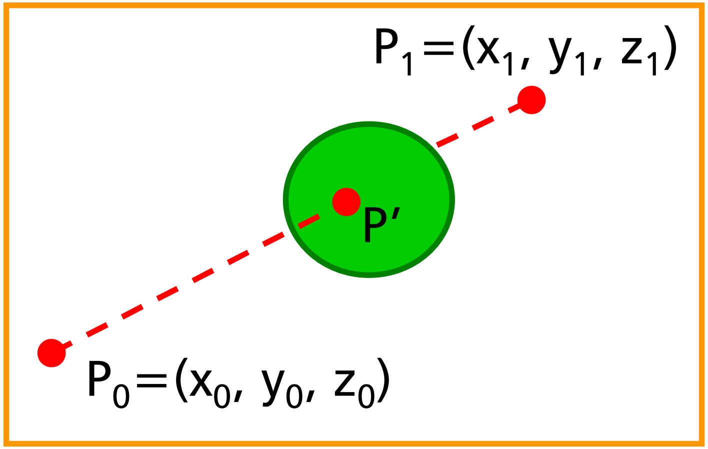
Qual a coordenada de P'?



Ray Tracing – Interseção com Esferas

Por conveniência reescrevemos as equações paramétricas:

$$x = x_0 + t \cdot \Delta x$$
$$y = y_0 + t \cdot \Delta y$$
$$z = z_0 + t \cdot \Delta z$$



Uma esfera é representada pela equação:

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$$

Onde ***(a, b, c)*** é o centro da esfera e ***r*** seu raio.

Para calcularmos a interseção entre o raio e a esfera, substituímos os valores ***x***, ***y*** e ***z*** da equação do raio na equação da esfera.

Ray Tracing – Interseção com Esferas

$$x = x_0 + t \cdot \Delta x$$
$$y = y_0 + t \cdot \Delta y$$
$$z = z_0 + t \cdot \Delta z$$

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$$

$$(\Delta x^2 + \Delta y^2 + \Delta z^2)t^2 + 2t\left[\Delta x(x_0 - a) + \Delta y(y_0 - b) + \Delta z(z_0 - c)\right] + (x_0 - a)^2 + (y_0 - b)^2 + (z_0 - c)^2 - r^2 = 0$$

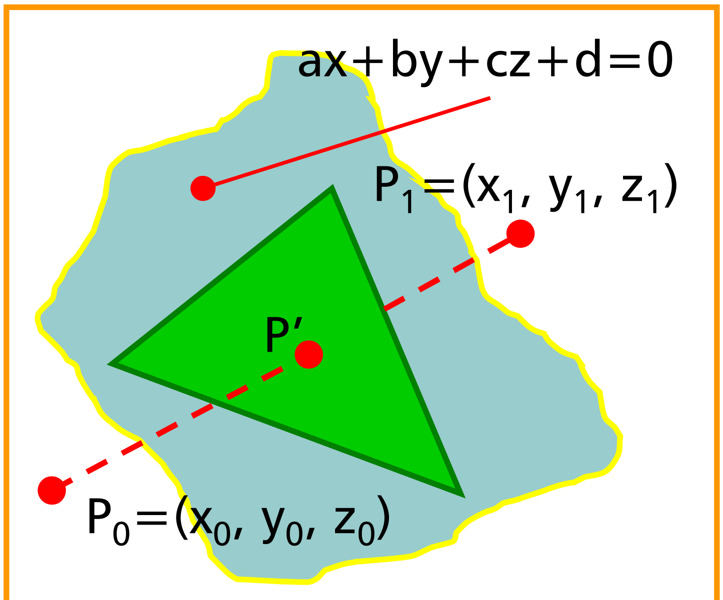
Resolvendo a equação acima podemos encontrar:

- ✓ nenhuma raiz real: o raio não intercepta a esfera;
- ✓ uma raiz real: o raio tangencia a esfera;
- ✓ duas raízes reais: o raio intercepta a esfera na raiz positiva de menor valor.

Ray Tracing – Interseção com Polígonos

Para encontrar a interseção entre o raio e um polígono primeiramente determinamos se o raio intercepta o plano do que contém o polígono.

Substituímos os valores de x, y e z da equação do raio na equação do plano que contém o polígono.



$$\left.\begin{aligned} x &= x_0 + t\cdot\Delta x \\ y &= y_0 + t\cdot\Delta y \\ z &= z_0 + t\cdot\Delta z \end{aligned}\right\} \Rightarrow ax + by + cz + d = 0$$

$$t = -\frac{(a\cdot x_0 + b\cdot y_0 + c\cdot z_0 + d)}{(a\cdot\Delta x + b\cdot\Delta y + c\cdot\Delta z)}$$

Depois, testes dentro-fora.

Ray Tracing – Desempenho

É um algoritmo computacionalmente caro. Numa tela com 1024x768 pixels e 100 polígonos serão necessários 78.643.200 cálculos de interseção.

O cálculo de interseções representa de 75% a 90% do tempo de processamento.

Usar como pré-filtro o teste de visibilidade pelo cálculo da normal.

Outra alternativa é envolver os objetos (poliedros) em uma esfera. Se a esfera envolvente é interceptada, então verificamos se há interseção com as faces do poliedro.