Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Pipeline de Visualização 3D

André Tavares da Silva

andre.silva@udesc.br

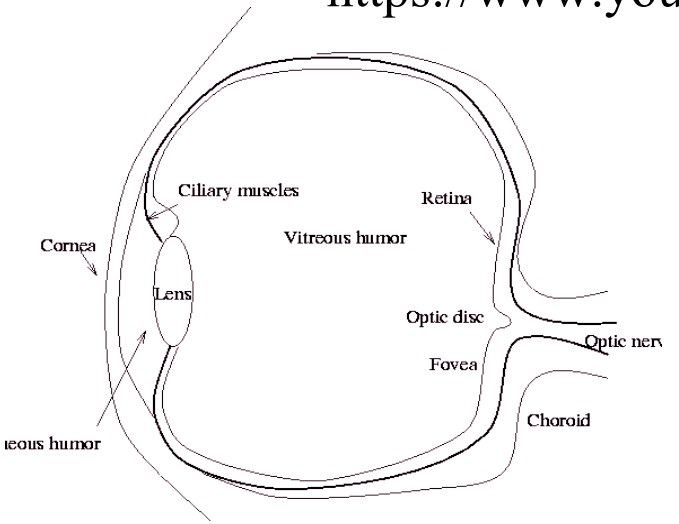
Capítulo 5 de “Foley”
Capítulo 2 de Azevedo e Conci

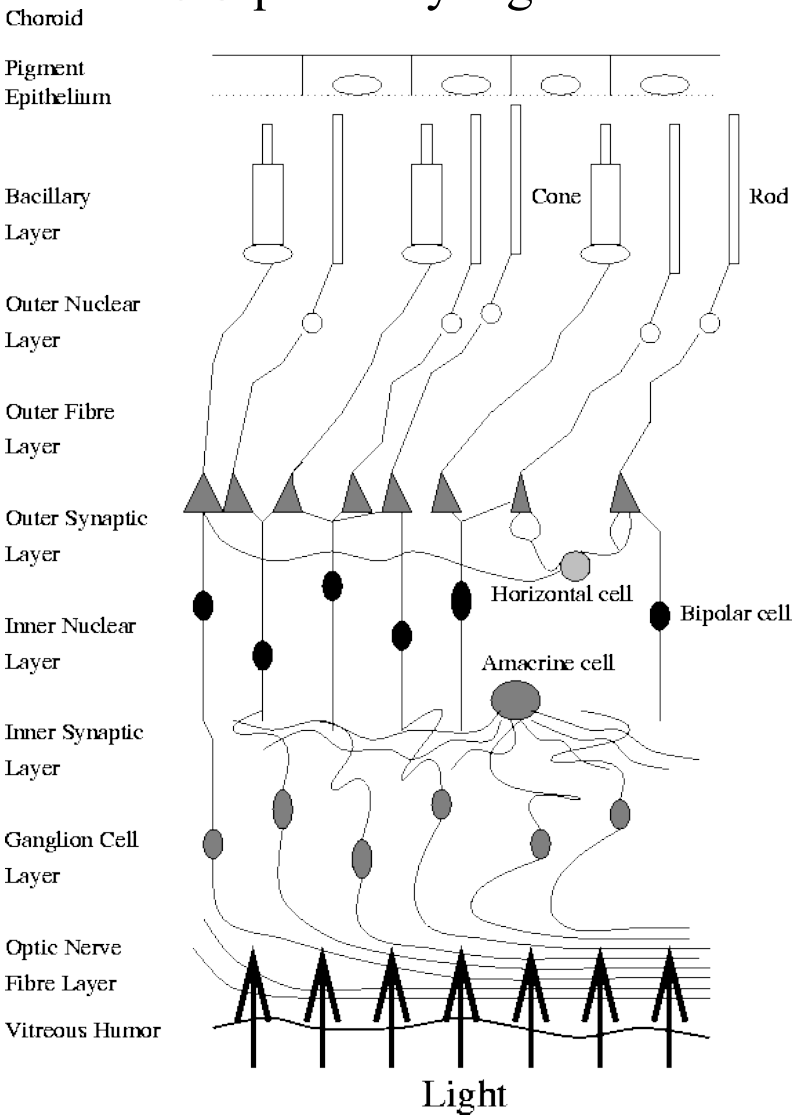
Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

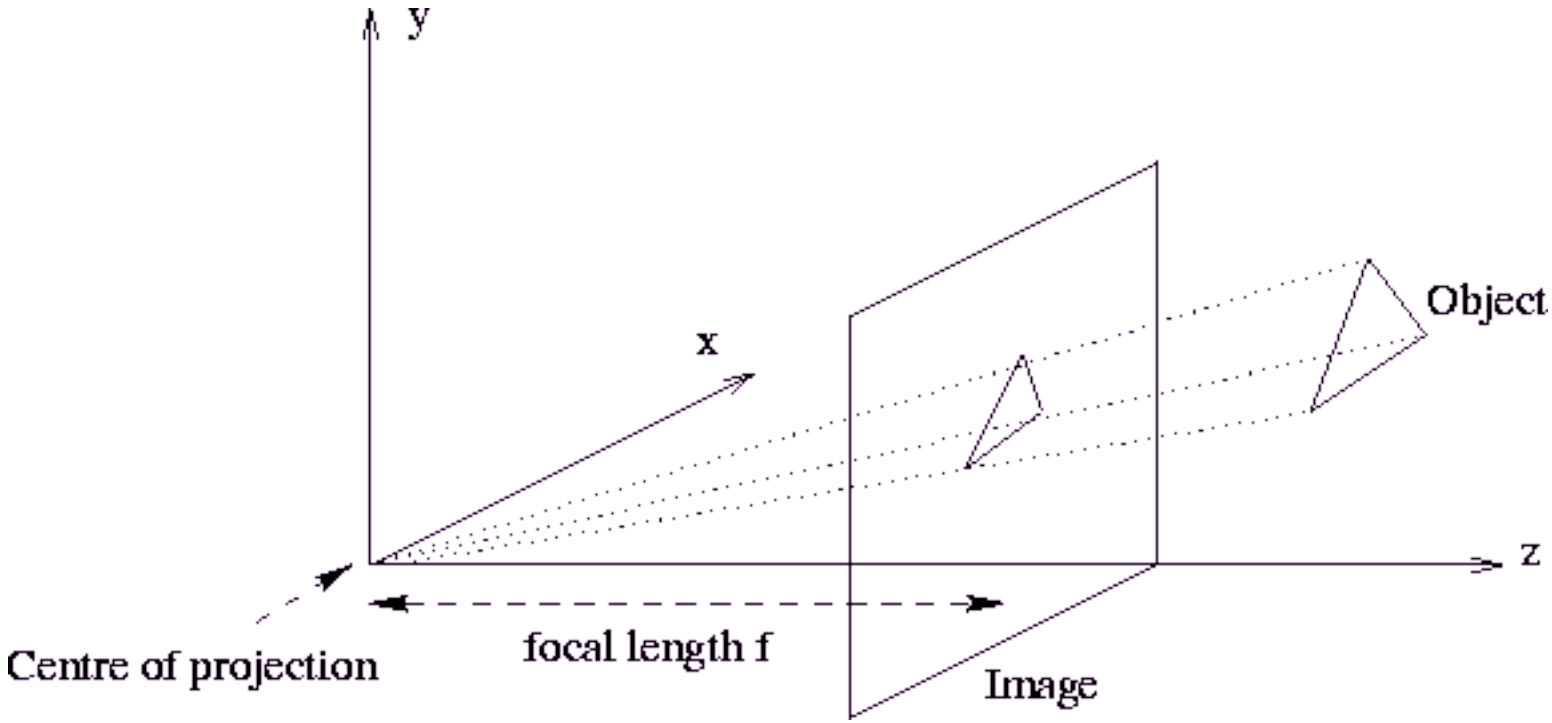
Processo de Visualização

https://www.youtube.com/watch?v=OGqAM2Mykng



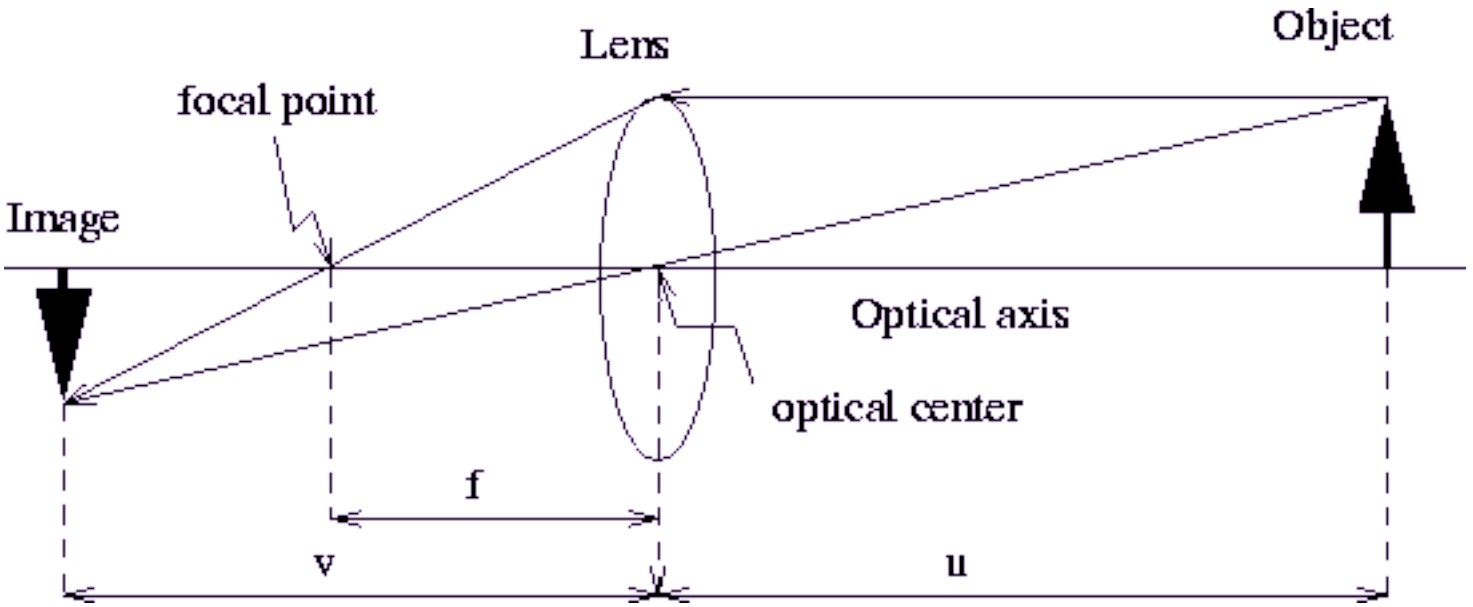


Processo de Visualização



Modelo *pinhole*

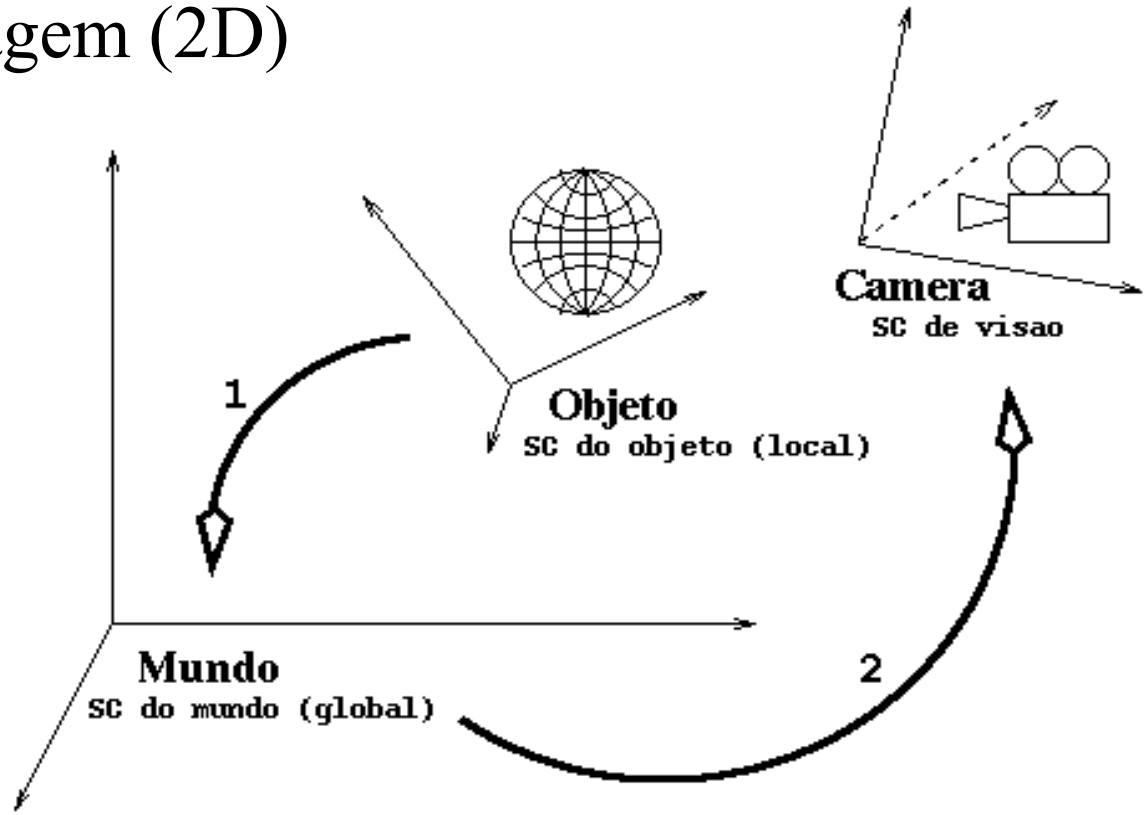
Processo de Visualização



Modelo de câmera simplificado

Processo de Visualização 3D

1) Transforma a forma da Observação (3D)
2) Transformam coordenadas do mundo para coordenadas de imagem (2D)



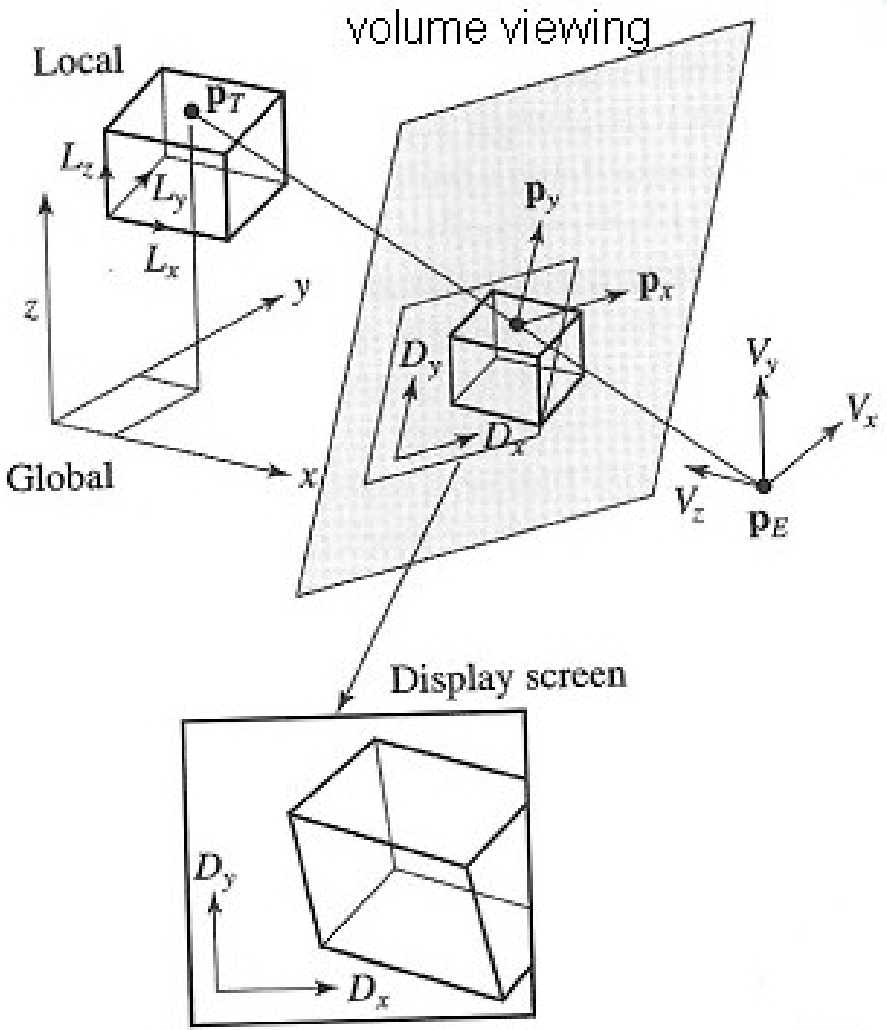
Processo de Visualização 3D simplificado

Plano de Visualização ou
Plano de Projeção

Ponto de Observação ou
Posição da Câmera Virtual

Área de Visualização no
Plano de Projeção ou *Window*

Sistemas:
- Do Mundo/De Referencia
- Do Objeto
- Da Câmera



Visualização 2D x 3D

- O que muda?
  - Uma dimensão a mais: profundidade (Z)
  - Cena 3D e dispositivo de exibição 2D

Objetos 2D

- Pontos 2D
- Polígonos (Triângulo, retângulo, circulo, ...)
- Objetos compostos por um polígono

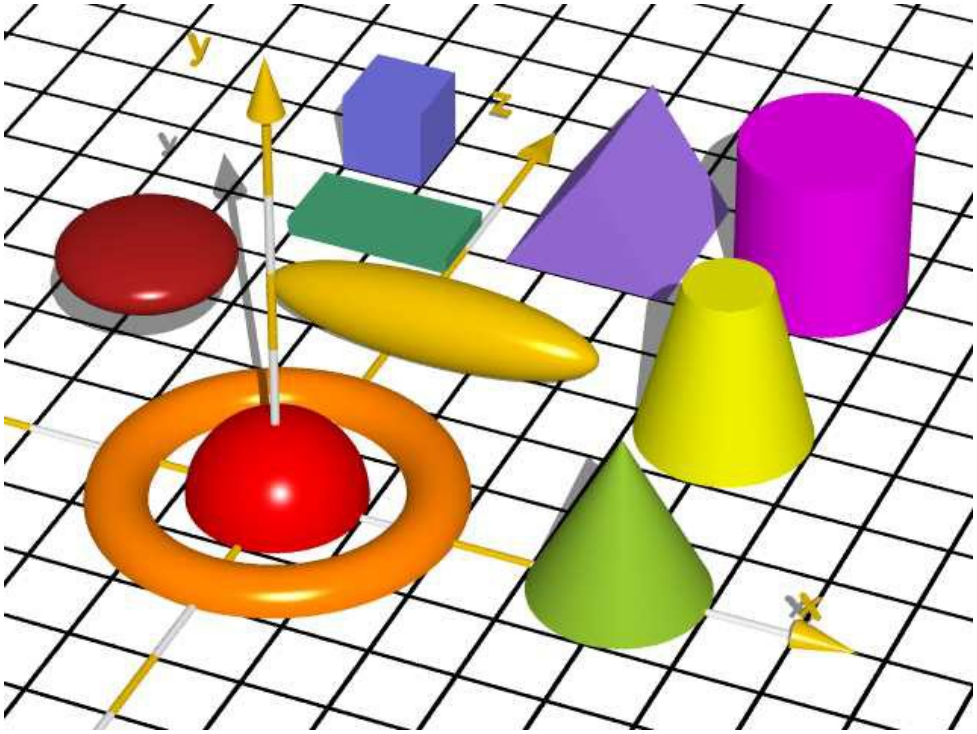
Plano: largura, altura (x,y)

Objetos 3D

- Pontos 3D
- Polígonos (Cubo, Paralelepípedo, Esfera ...)
- Objetos compostos por polígonos

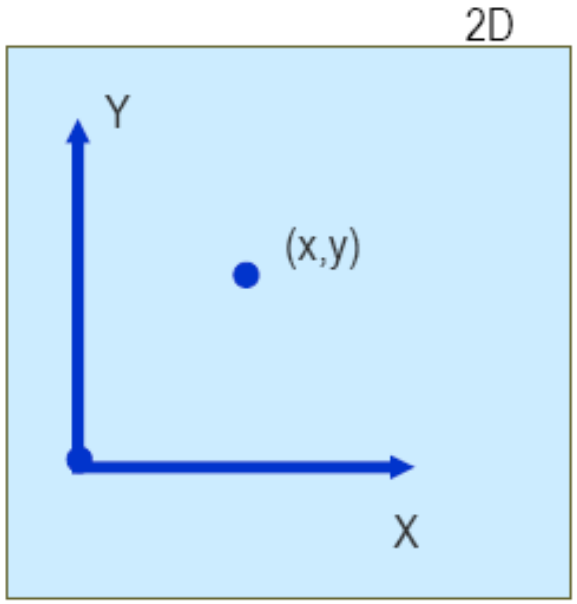
Espaço: largura, altura e profundidade (x,y,z)

Objetos 3D

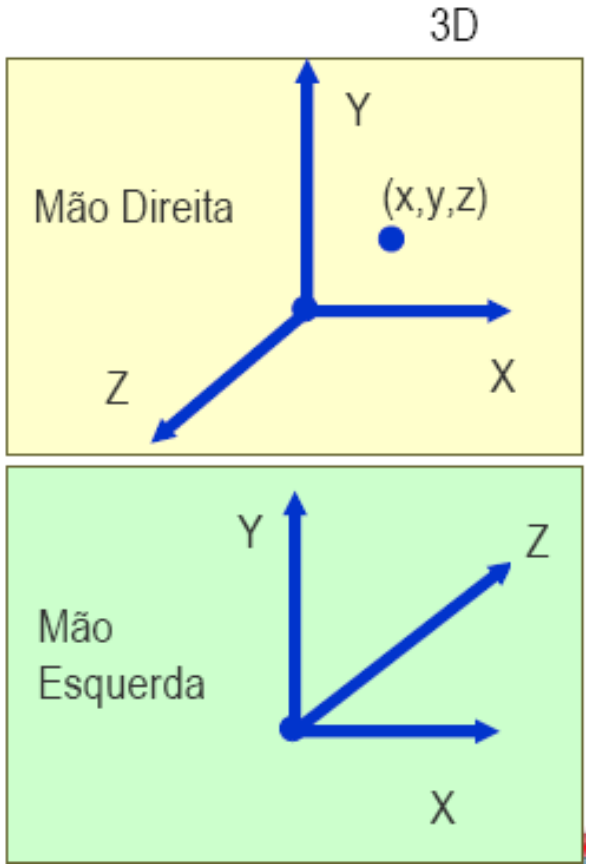


- Paralelepípedo
- Pirâmide
- Cone
- Esfera
- Prisma
- Torus

2D → 3D

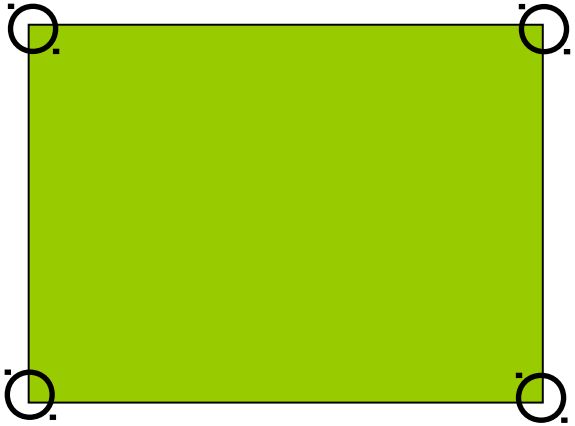


Plano

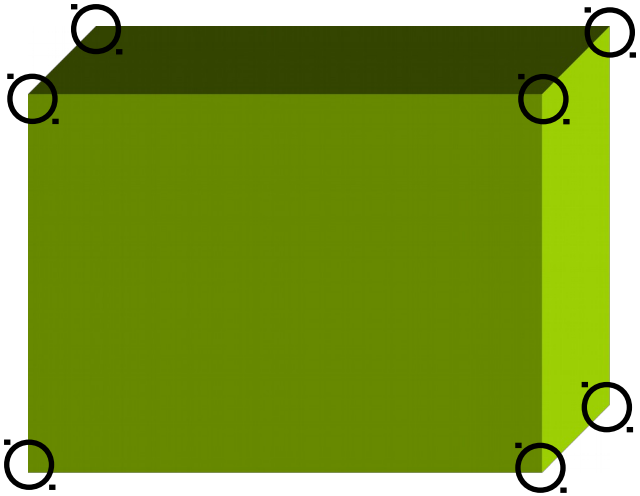


Espaço

2D → 3D



2D

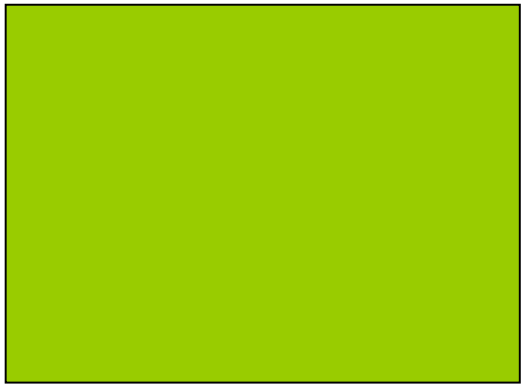


3D

Vértices

2D → 3D

Uma face



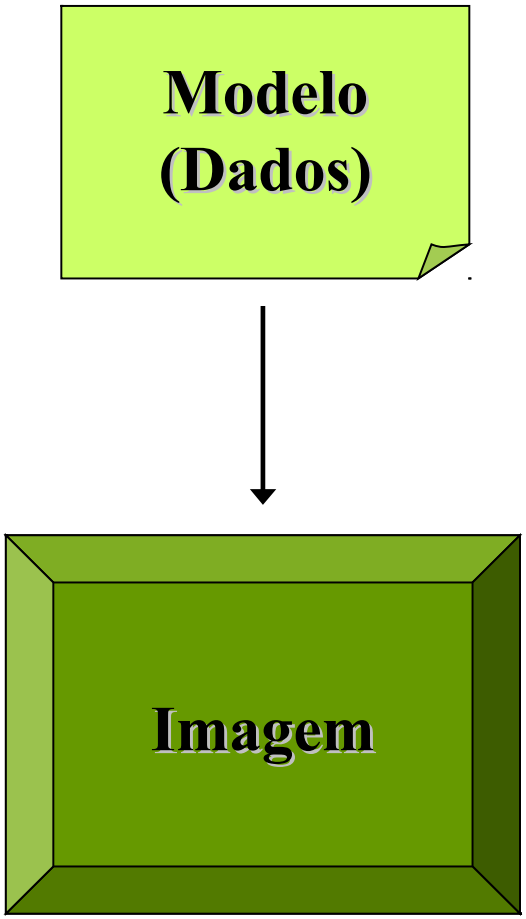
2D

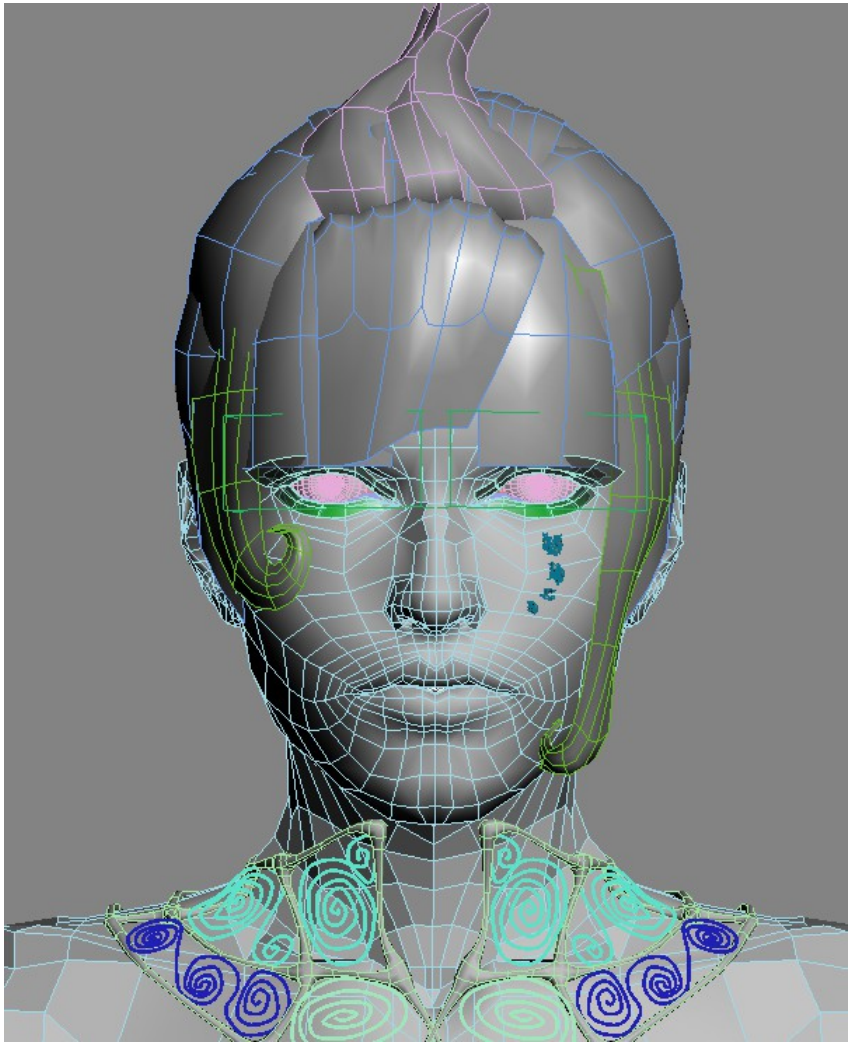
Várias faces

3D

Faces

Pipeline de Visualização 3D





Modelo 3D

Imagem

Etapas do Pipeline de Visualização

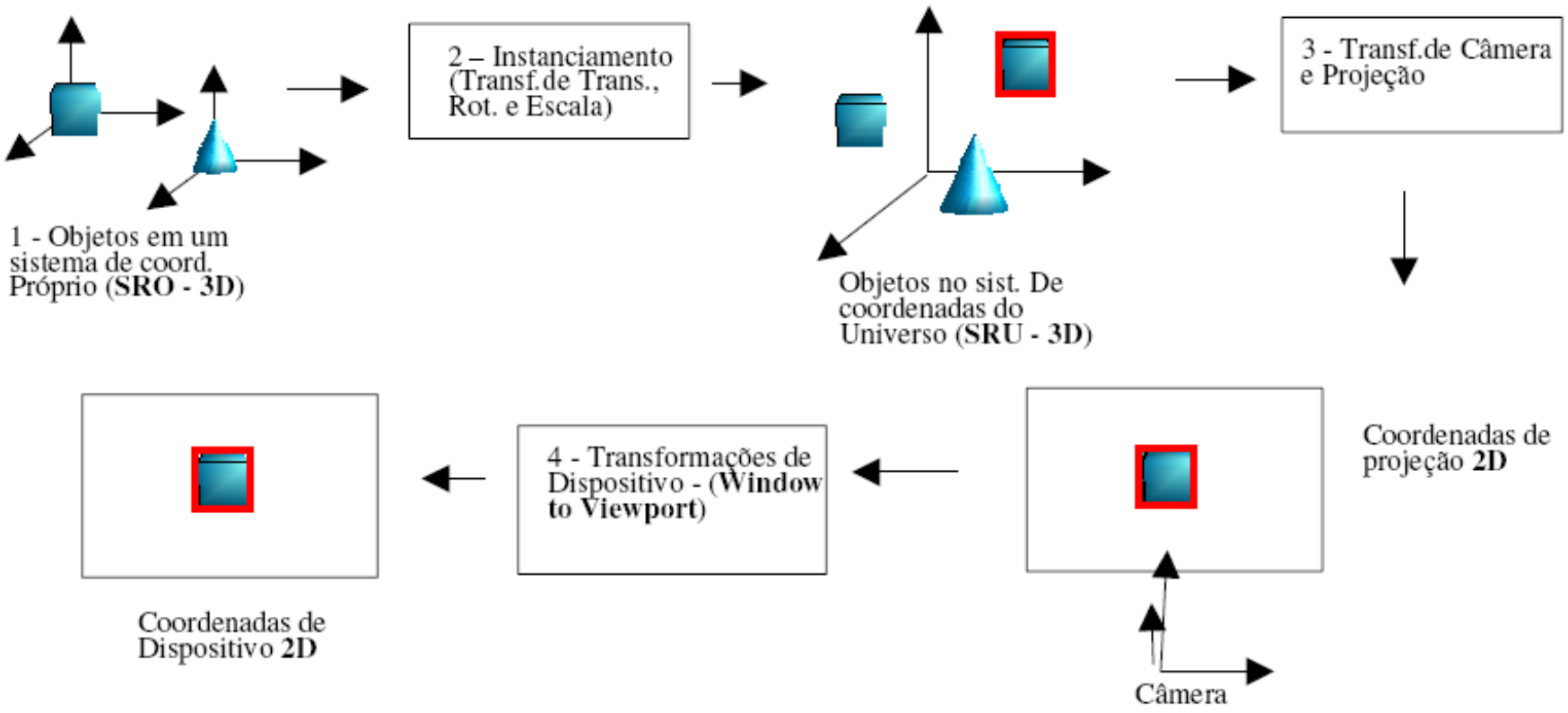
Etapas 2D

- Instanciamento
- Recorte 2D
- Mapeamento

Etapas 3D

- Instanciamento
- Transformações de câmera e projeção
- Recorte 3D
- Projeção
- Mapeamento

Pipeline de Visualização 3D



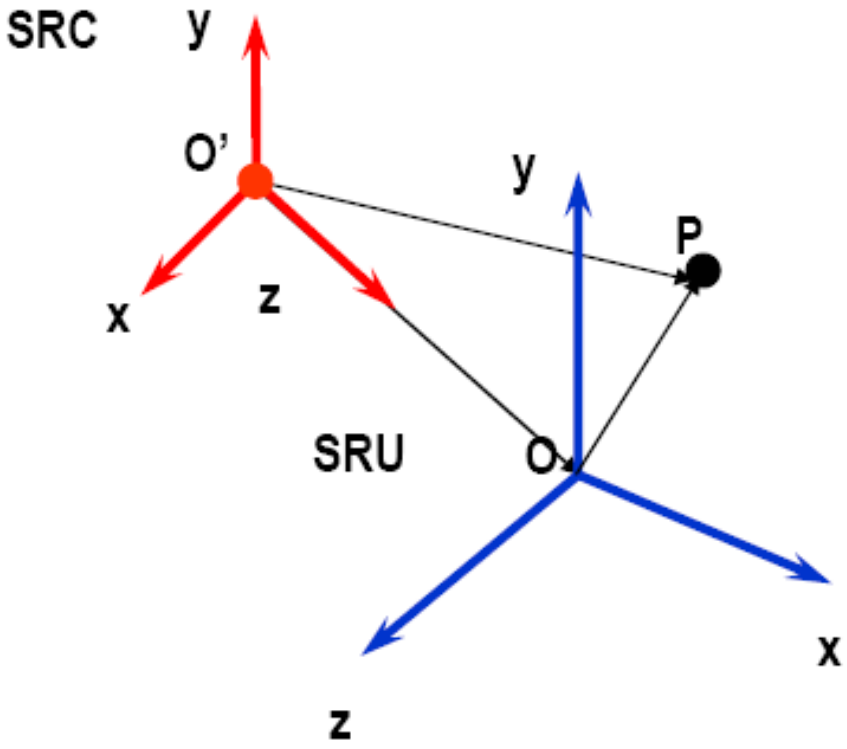
Aspectos Envolvidos

- Posicionamento da câmera
  - Observador
- Orientação da câmera
  - Direção da projeção
- Definição do tipo de projeção
  - Paralela ou Perspectiva
- Definição da região de interesse SRU
  - Prisma ou tronco de pirâmide

Câmera Virtual/Sintética

- Metáfora para representar o que é visto.
- A câmera pode ser movimentada para qualquer posição em qualquer sentido.
- A câmera será o nosso programa, que transforma as informações dos objetos em uma imagem.

Posicionamento da Câmera

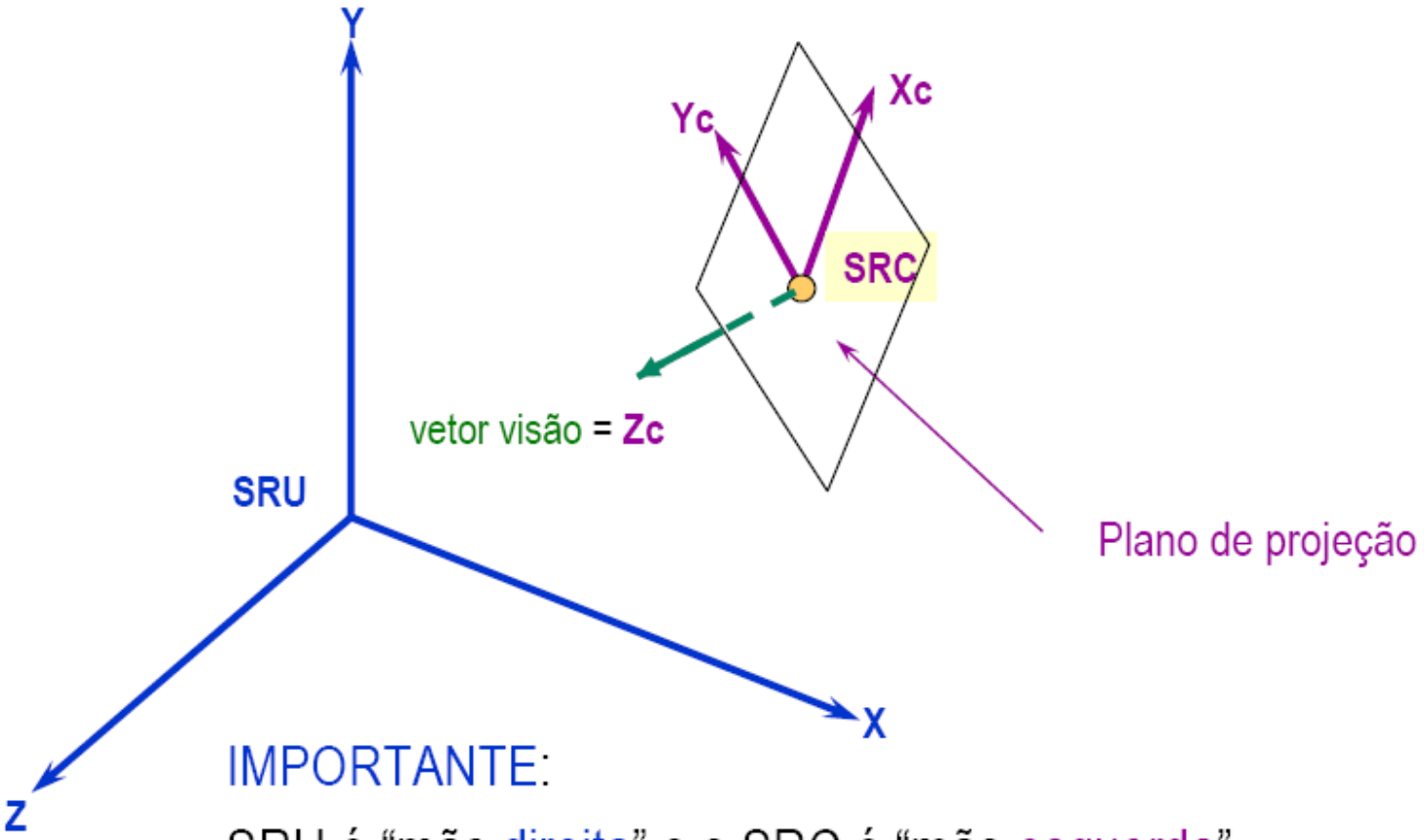


$P = (x, y, z)_{SRU}$

transformação de coordenadas

$P = (x', y', z')_{SRC}$

Orientação da Câmera



IMPORTANTE:

SRU é "mão direita" e o SRC é "mão esquerda"

Enfatizando...

- SRC é “Mão Esquerda”
  - Z cresce com a distância da câmera
- SRU é “Mão Direita”

Parâmetros da Visualização

- Câmera Sintética (3D)
  - SRC (Sistema de Referência da Câmera)
    - Posição de Observação (3 DOF)
    - Direção de Observação (3 DOF)
  - Ponto Focal (1 DOF)
- Planos de Corte
  - View Frustrum
- Tipo de Projeção (2D)

Projeção

- Estudada desde o século XIV.
- Transforma os pontos de um sistema de **N** dimensões em pontos de um sistema de **M** dimensões, onde **M** < **N**.
- Nosso caso: **3D → 2D**.

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Projeção

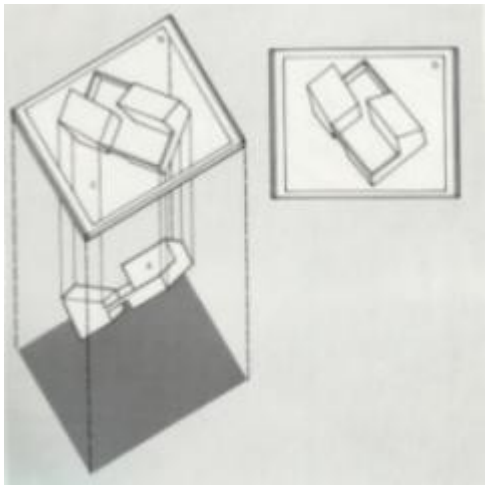
Albrecht Dürer

Projeção

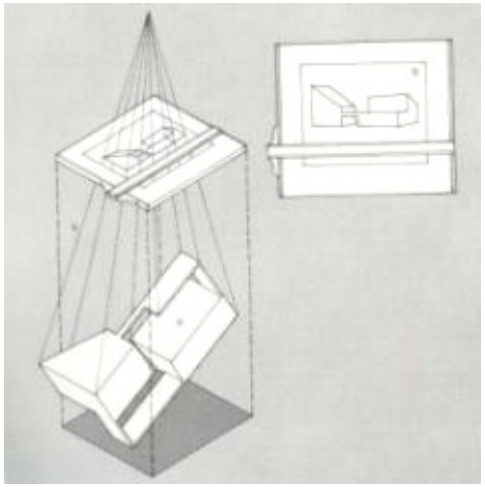
- Projeção de um objeto no espaço através de linhas de projeção (**raios de projeção**), que têm origem em um **centro de projeção** e passam pelos pontos do objeto interseccionando um **plano de projeção**, onde se forma a **imagem bidimensional**.

Tipos de Projeção

Paralela



Perspectiva

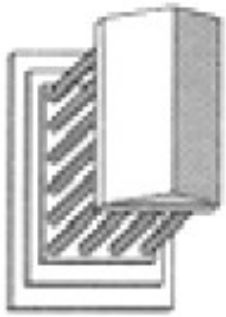


- Diferença entre as projeções: na paralela os raios de projeção são paralelos entre si enquanto na projeção perspectiva eles convergem para um ponto.

Tipos de Projeção

Front elevation

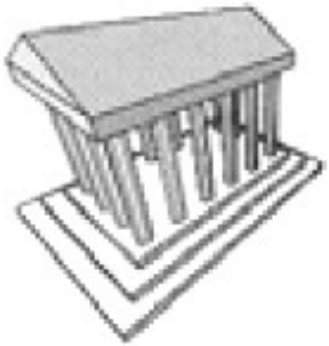
Elevation oblique



Plan oblique

Isometric

One-point perspective



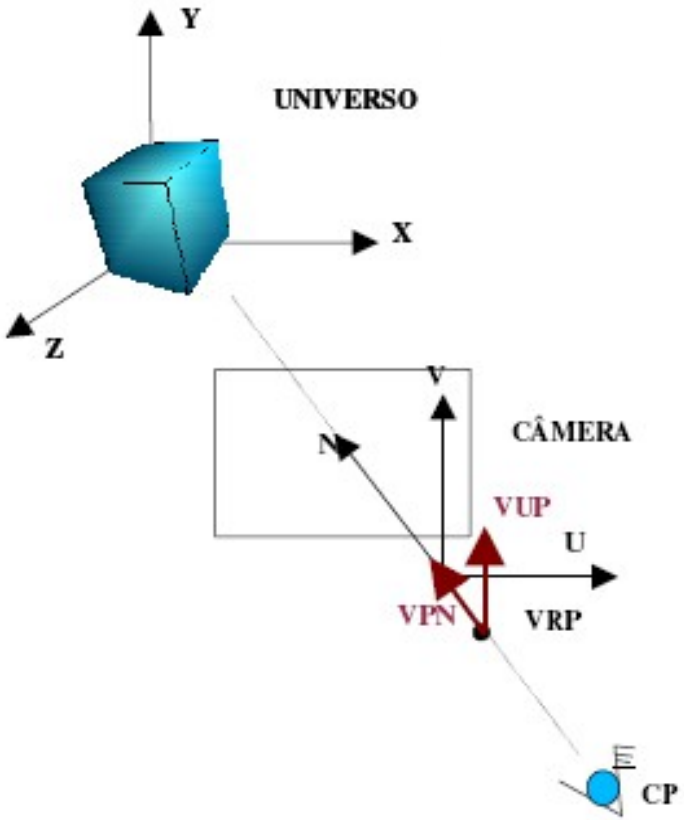
Three-point perspective

Projeções

- O efeito da projeção perspectiva é semelhante ao sistema visual humano.

- A projeção perspectiva é mais **realista**, mas não é útil quando precisamos medir as dimensões de um objeto.

Sistema de Referência da Câmera

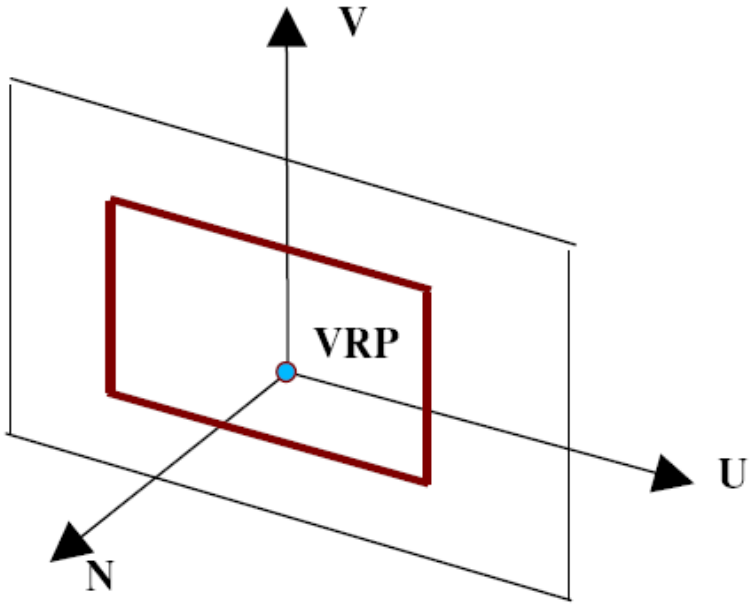


- Plano de projeção (posição e orientação)
- Janela de Visualização
- Posição do olho
  (centro de projeção)

- VPN(v, u, n) - View Plane Normal
- VUP indica onde é para cima
- VRP indica a posição da câmera
- VPR + VPN : Plano de Projeção

Sistema de Referência da Câmera

- Após a definição do SRC, podemos definir a janela de visualização (*viewport*).



Sistema de Referência da Câmera

- Definida a janela, definimos o centro de projeção.
- PRP: Ponto de referência da projeção. Dado em coordenadas do SRC e usualmente definido ao longo do eixo n.

Volume de Visualização

- Por último é necessária a definição da área do espaço que será visualizada (*View Volume*).
- Somente os objetos dentro desta área serão visualizados.

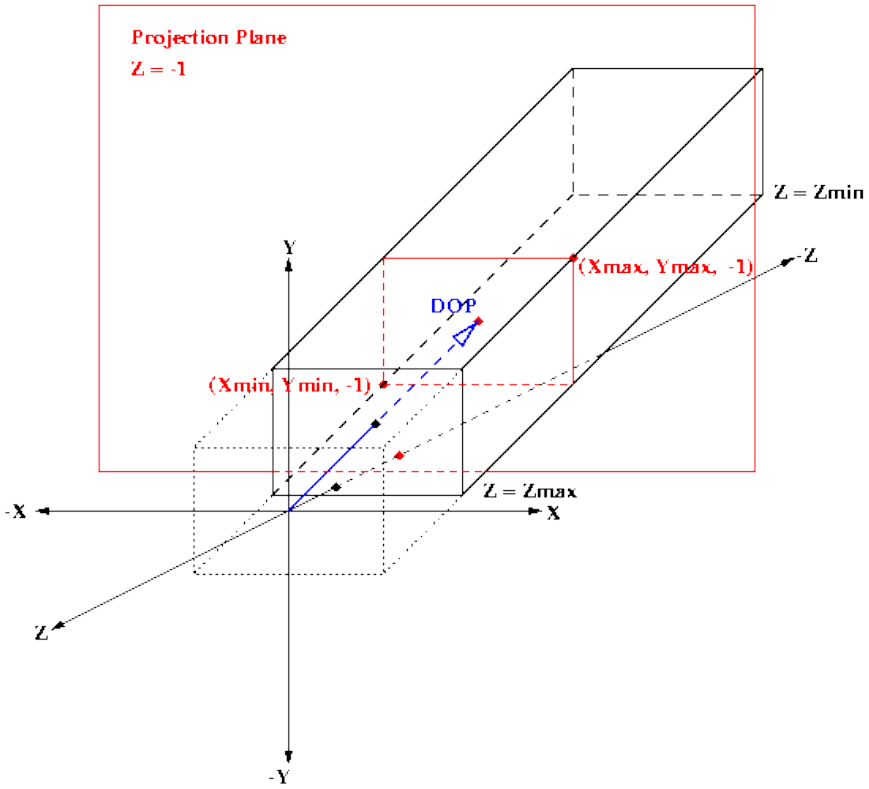
Do Plano Projetivo para a Tela

(Foley et al 1996: cap 5.4)

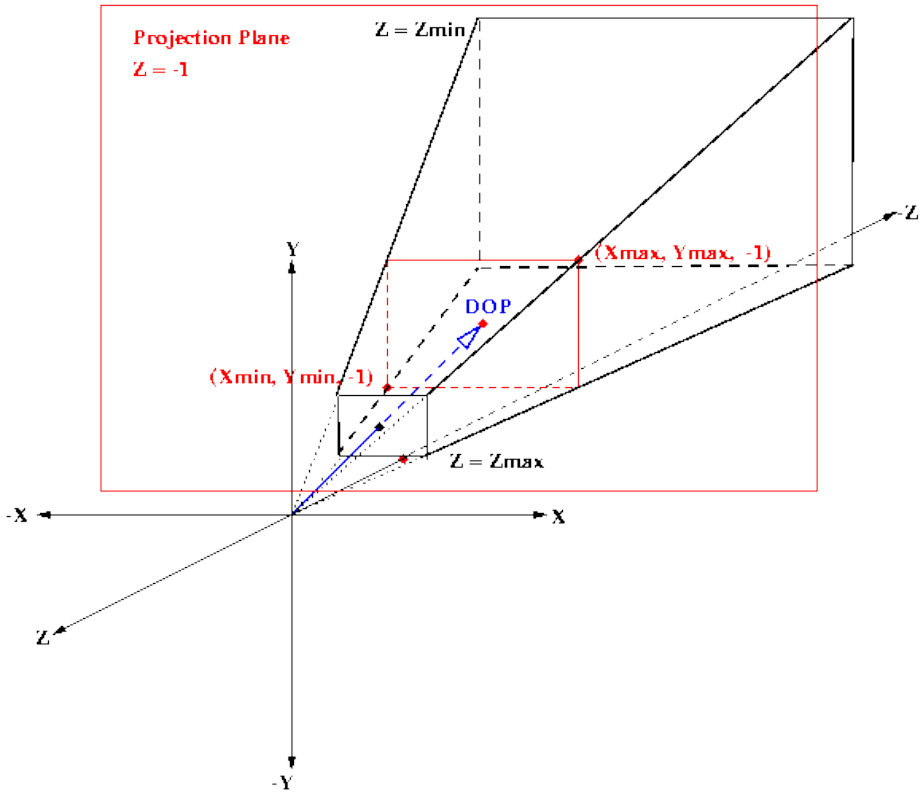
- Nem tudo que é projetado é visto
- Objetos MUITO distantes não são vistos
- Objetos MUITO próximos não são vistos
- Objetos Acima/Abaixo não são vistos
- Objetos À Esquerda/Direita não são vistos

Volume de Visualização

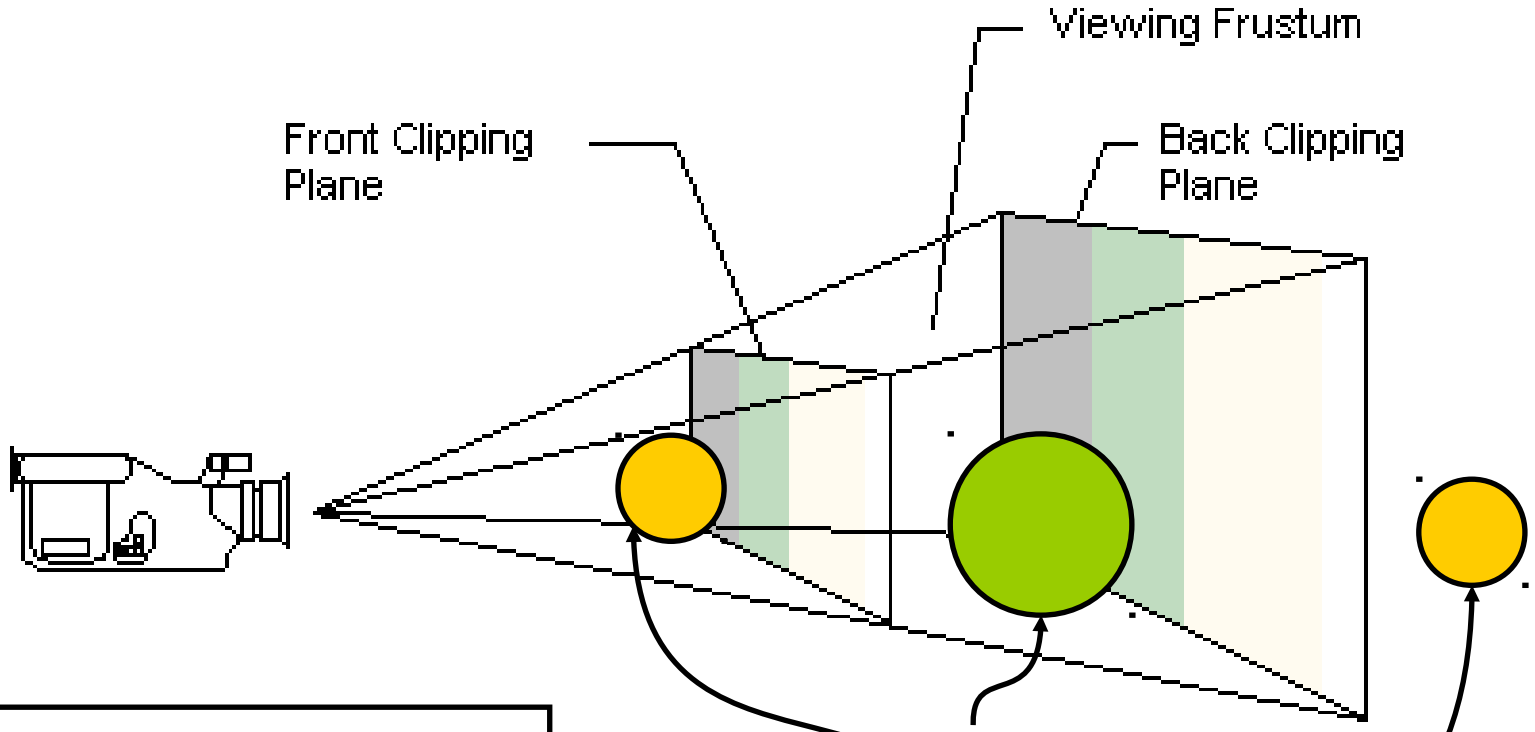
Parallel Projection Frustum



Perspective Projection Frustum

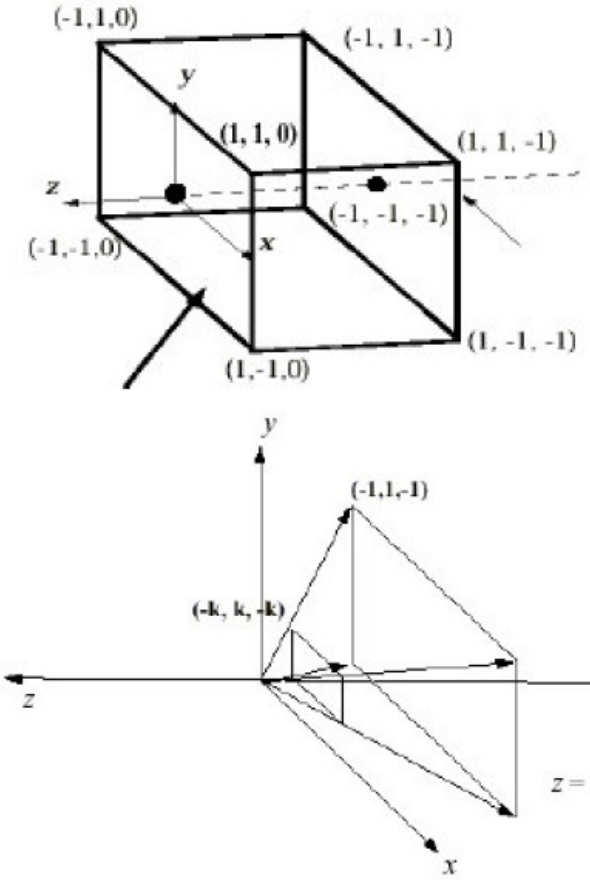


Volume de Visualização



E se existir um objeto interseccionando os planos de *clipping*?
Recortar

Coordenadas Normalizadas



- Planos para o volume da projeção paralela:

  x:{-1,1}; y:{-1,1}; z:{-1,1}

- Para o volume da projeção perspectiva:

  x:{z,-z}; y:{z,-z}; z:{$-z_{min}$,-1}

Recorte 3D

- O recorte 3D é feito através de uma extensão do algoritmo de recorte 2D.
- O algoritmo de Cohen-Suterland classifica as extremidades das linhas de acordo com sua posição no espaço em relação ao volume de visualização.
- No algoritmo 2D eram utilizados 4 bits, em 3D são utilizados 6 bits.

Recorte 3D

- Bits:
  - Bit 0: ponto acima do volume
  - Bit 1: ponto abaixo do volume
  - Bit 2: ponto direita do volume
  - Bit 3: ponto esquerda do volume
  - Bit 4: ponto atrás do volume
  - Bit 5: ponto na frente do volume

Recorte 3D

- Todos os bits em zero: trivialmente aceitos.
- AND entre os pontos diferente de zero: trivialmente recusados.
- AND entre os pontos igual a zero: recortar.

Recorte 3D

- Recorte: intersecção entre a reta e os planos de visualização.
- Como?

  Usando a representação paramétrica da reta.

  P0(x0,y0,z0) e P1(x1,y1,z1)

  $x = x0 + t(x1-x0)$

  $y = y0 + t(y1-y0)$

E para esferas?

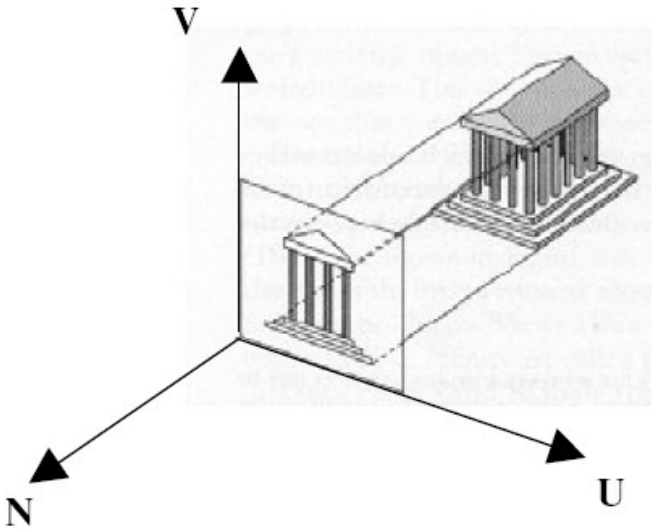
Coordenadas de Projeção

- Precisamos nos "livrar" de uma dimensão.
- Projeções Ortográficas (paralelas):
  - Basta descartar a coordenada n (assumindo que o plano de projeção em n=0).

    up = u = x

    vp = v = y

    n = 0



Coordenadas de Projeção

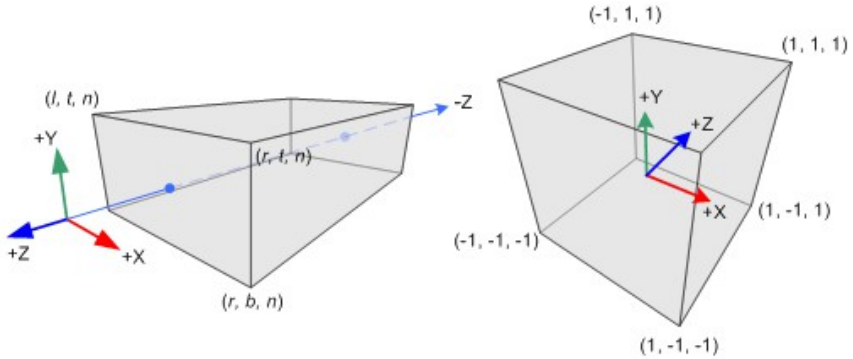
- Projeções Perspectivas:
  - Assumindo que o plano de projeção é perpendicular ao eixo N do SRC e a uma distância **d:**

```
up = u / (n / d)
vp = v / (n / d)
```

  - Existem matrizes de transformação para obter as coordenadas projetadas.

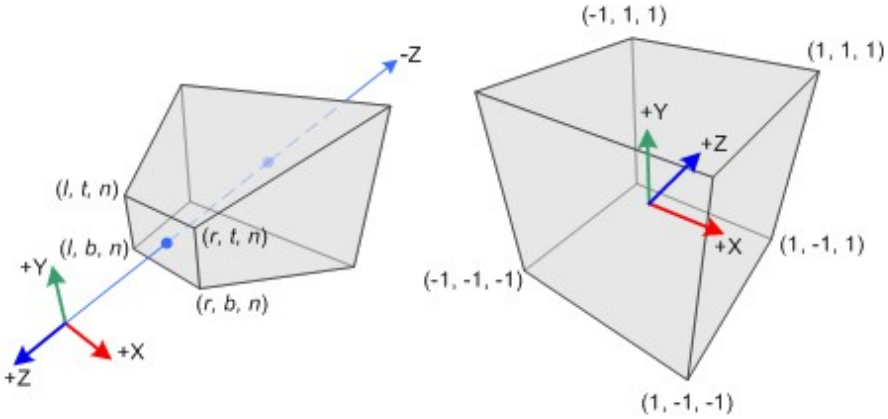
Matriz de projeção

Paralela/ortográfica



$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

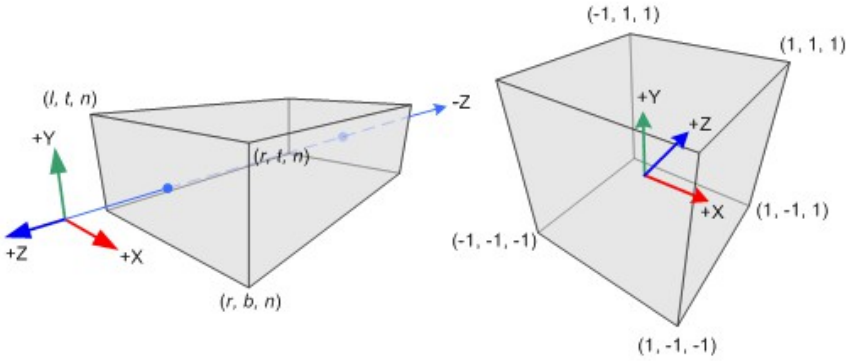
Perspectiva



$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Matriz de projeção

Paralela/ortográfica



$$\begin{pmatrix} \frac{1}{r} & 0 & 0 & 0 \\ 0 & \frac{1}{t} & 0 & 0 \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

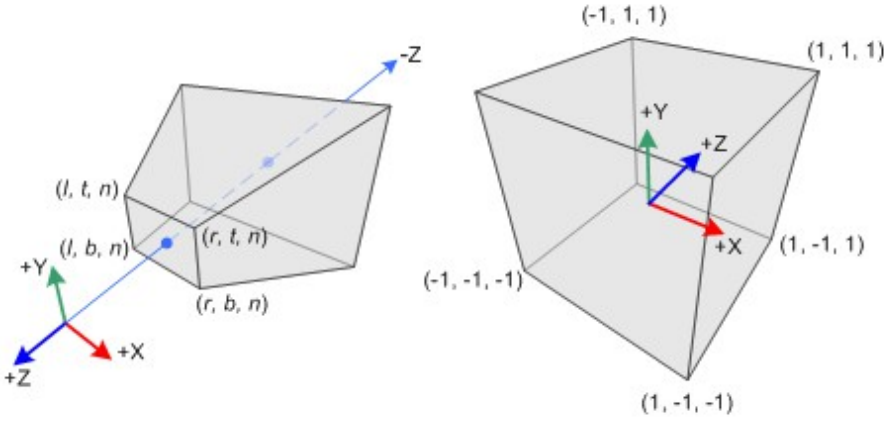
caso forem simétricos

$r = -l$

e

$t = -b$

Perspectiva

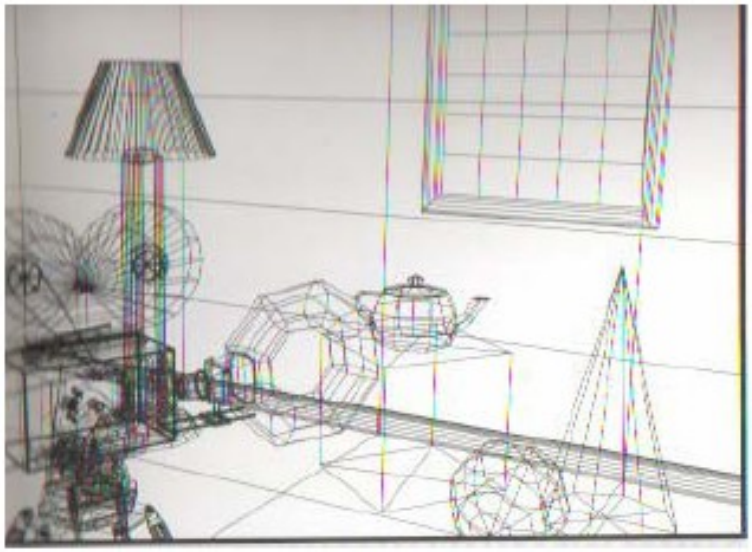


$$\begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Resumo da Câmera Sintética

- Instanciar os objetos no universo (SRO → SRU)
- Converter o SRU para o SRC
- Aplicar a normalização
- Recorte 3D contra volume canônico
- Efetuar projeção
- Mapeamento de *window* para *viewport*

Pipeline de Visualização 3D



Wireframe

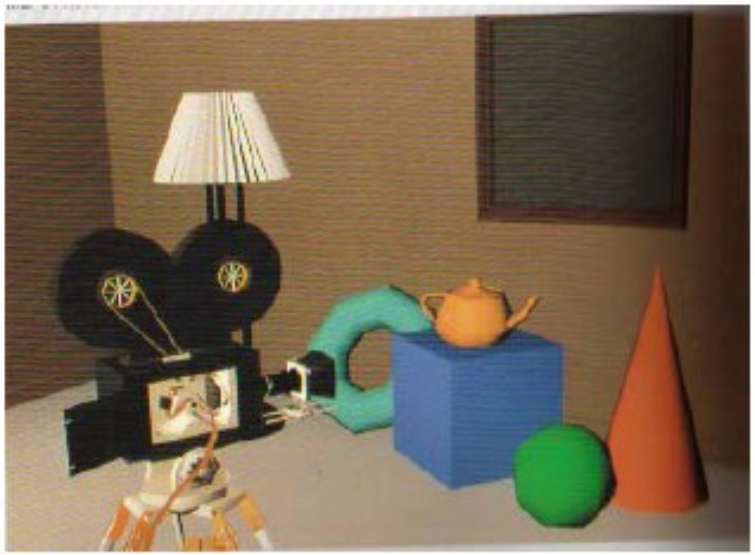
Wireframe com Cores

Pipeline de Visualização 3D

Remoção de Superfícies Ocultas

Luz Ambiente, sem consideração de orientação de faces

Pipeline de Visualização 3D



Sombreamento das faces

Sombreamento com Reflexão Especular

Pipeline de Visualização 3D

Superfícies Curvas

Mais de uma Fonte de Luz

Pipeline de Visualização 3D

Texturas

Sombras

Pipeline de Visualização 3D

Remoção de Faces Ocultas

Algoritmos de remoção de face oculta:

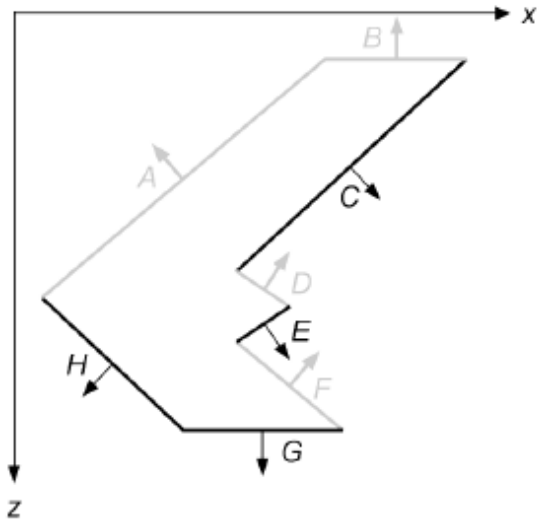
- Backface Culling
- Algoritmo do Pintor
- Z-Buffer (Depth-Buffer)

Backface Culling

- Remove os polígonos traseiros (backfaced) dos objetos.
- De cada polígono que compõe o objeto sai um vetor normal. Esse vetor indica qual dos lados é a frente do polígono.
- Testa o vetor normal de cada polígono. Se ele não aponta para o observador ele é um polígono traseiro.

Backface Culling

- Se N.V > 0, para objetos sólidos isso significa que essa face nunca será vista pelo observador
- V é um vetor que parte do olho para o objeto

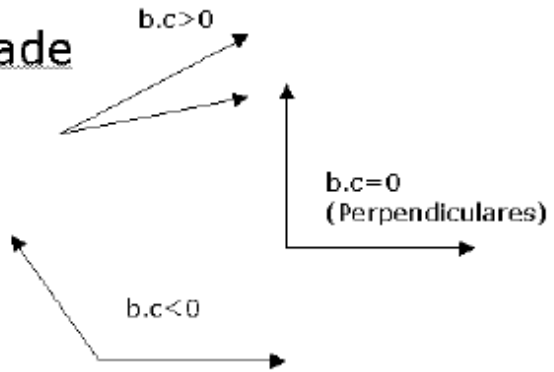


Sinal de b.c e Perpendicularidade

$\cos\theta > 0$ se $|\theta| < 90^\circ \rightarrow$ b.c > 0

$\cos\theta = 0$ se $|\theta| = 90^\circ \rightarrow$ b.c = 0

$\cos\theta < 0$ se $|\theta| > 90^\circ \rightarrow$ b.c < 0

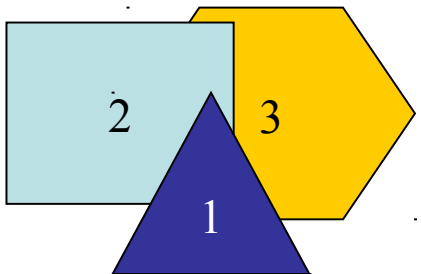


Backface Culling

- Não é uma solução completa.
- Grande velocidade.
- Cerca de metade dos polígonos são descartados.
- Aplicado geralmente com outro algoritmo de remoção de faces ocultas.
- Fácil integração com o hardware.
- Implementado em OpenGL.

Algoritmo do Pintor

- Vários objetos na cena
- Desenha os polígonos como um pintor à óleo faz: os objetos mais distantes primeiro.
  - Ordena os polígonos de acordo com o z;
  - Resolve ambiguidades onde os z´s se sobrepõe;
  - Desenhar a partir do maior z até o menor;
  - O polígono mais próximo será desenhado por último;
  - Precisa de todos os polígonos.



Z-Buffer (Depth-Buffer)

- Criado em 1974, por Catmull.
- Idéia:
  - Na etapa de rasterização considerar o z também;
  - Além do framebuffer, existe também um buffer de profundidade (depth-buffer);
  - Inicialmente todo o depth-buffer é setado para o infinito;
  - Framebuffer é setado para a cor de fundo;

Z-Buffer (Depth-Buffer)

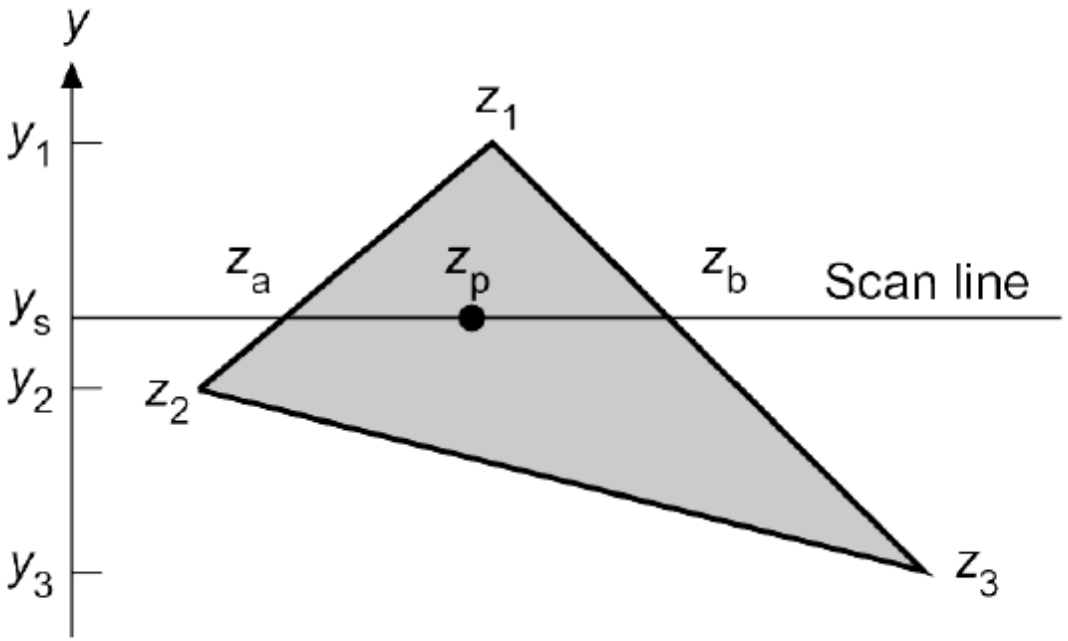
- Realizar a rasterização através da seguinte rotina:

```
WritePixel(int x, int y, float z, colour)
if ( z < zbuf[x][y] ) then
    zbuf[x][y] = z;
    frambuffer[x][y] = colour;
end
```

- O *buffer* armazena o pixel mais próximo até então em (x,y);
- Não renderiza um pixel se ele tiver profundidade maior do que z[x][y];

Z-Buffer (Depth-Buffer)

- Determinação da profundidade



$$z_a = z_1 - (z_1 - z_2)\frac{y_1 - y_s}{y_1 - y_2}$$

$$z_b = z_1 - (z_1 - z_3)\frac{y_1 - y_s}{y_1 - y_3}$$

$$z_p = z_b - (z_b - z_a)\frac{x_b - x_p}{x_b - x_a}$$

Z-Buffer (Depth-Buffer)

- Vantagens:
  - Fácil implementação;
  - Simples de implementar em hardware;
  - Pode começar antes de ter todos os polígonos da cena;
  - Objetos não necessitam ser polígonos;
  - Implementado em OpenGL;

Z-Buffer (Depth-Buffer)

- Desvantagens:
  - Consome “muita” memória;
  - Um pixel pode ser desenhado várias vezes;

Comparação

- Backface Culling: rápido, porém insuficiente;
- Algoritmo do pintor: muitos detalhes, lento;
- Z-Buffer: rápido de implementar, mas consome memória.