Computação Gráfica
Módulo III – Geometria

UniverCidade - Prof. Ismael H F Santos

Considerações Gerais

- Objetivo: *Discutir os principais conceitos eos princípios básicos dos Sistemas Gráficos e a Programação em OpenGL.*
- A quem se destina : *Alunos e Profissionais que desejem aprofundar seus conhecimentos sobre Computação Grafica e suas aplicações.*

Bibliografia

- *Computação Gráfica Volume 1. Jonas Gomes e Luiz Velho. Instituto de Matemática Pura e Aplicada – IMPA.*
- *Introdução a Computação Gráfica - Paulo Roma*
  - http://www.lcg.ufrj.br/compgraf1/downloads/apostila.pdf
  - http://www.lcg.ufrj.br/compgraf1/downloads/apostila.ps.gz
- *Notas do Curso ministrado na Universidade de Maryland pelo Prof. David Mount*
  - ftp://ftp.cs.umd.edu/pub/faculty/mount/427/427lects.ps.gz
  - http://www.lcg.ufrj.br/~esperanc/CG/427lects.ps.gz
- *Apostila Fundamentos da Imagem Digital – Antonio Scuri*
- *Computer Graphics: Principles and Practice, Second Edition. James Foley, Andries van Dam, Steven Feiner, John Hughes. Addison-Wesley.*
- *OpenGL Programming Guide, 2nd Edition. Mason Woo, Jackie Neider, Tom Davis. Addison Wesley.*

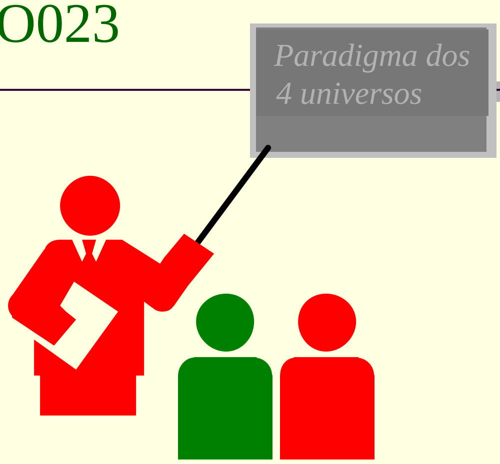
Bibliografia OpenGL

- OpenGL® Programming Guide, 2nd Edition. Mason Woo, Jackie Neider, Tom Davis. Addison Wesley.
  - http://www.lcg.ufrj.br/redbook
- Manual de referência online
  - http://www.lcg.ufrj.br/opengl
- Sítio oficial do OpenGL
  - www.opengl.org

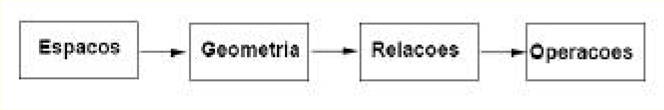
Ementa

- **Introdução ao OpenGL**
  - *Geometria*
    - Exemplos de Transformações 2D
    - Fórmulas e cálculos das transformações 2D
    - Usando matriz de transformação (por que?)
    - Coordenadas Homogêneas
    - Concatenação de transformações
    - Transformações 3D
  - *Projeções*
  - *Histórico*

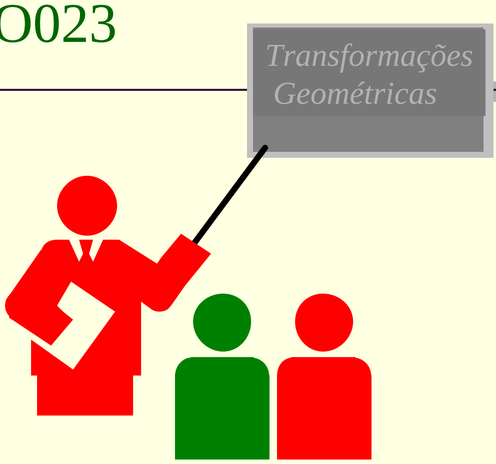
CG – CO023



Transformações Geométricas



CG – CO023

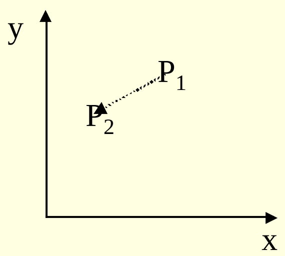


Exemplos de Transformações 2D

- Translação
- Escala
  - uniforme
  - não uniforme
- Rebatimento por um eixo (espelhamento)
- Troca de eixos
- Rotação pela origem

Translação

Translation of a point is simply vector addition

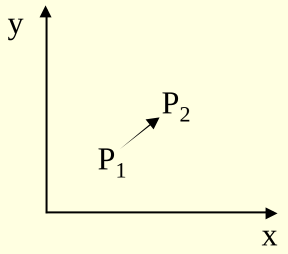


$$P_2 = P_1 + \begin{pmatrix} -2 \\ -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix} + \begin{pmatrix} -2 \\ -1 \end{pmatrix}$$

Escala

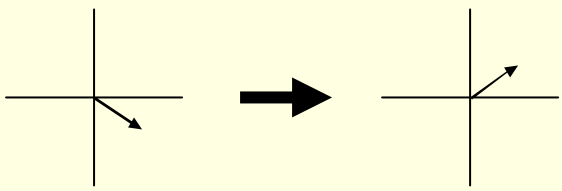
Scaling <u>about the origin</u> is scalar multiplication



$$\begin{bmatrix} 2 \\ 2 \end{bmatrix} = 2 * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

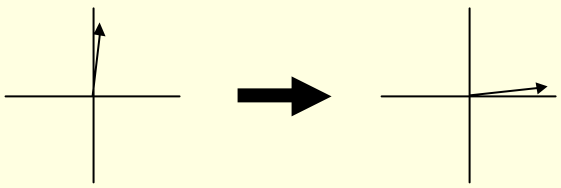
$$P_2 = P_1 * 2$$

Rebatimento por um eixo (espelhamento)



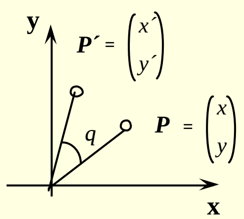
x = x
y = -y

Troca de Eixos



x = y
y = x

Rotação pela Origem



$x' = x.cos\ q - y.sen\ q$
$y' = x.sen\ q + y.cos\ q$

Por que usar Matriz nas Transformações?

- Todas as transformações podem ser efetuadas através da multiplicação de matrizes (usando coordenadas homogêneas).
- As transformações podem ser aninhadas e resolvidas de modo a haver apenas uma matriz de multiplicação a ser aplicada.
- A característica descrita acima se torna muito importante quando a mesma seqüência de transformações deve ser aplicada para diversos pontos.

Multiplicação de Matrizes

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

onde:

$$x' = a\,x + b\,y$$

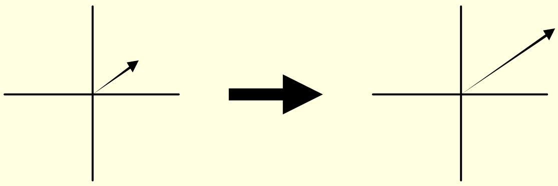
$$y' = c\,x + d\,y$$

Uniform Scaling

$$S = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}, \quad P = \begin{bmatrix} x \\ y \end{bmatrix}, \quad M = S\,P$$

$$\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s\,x \\ s\,y \end{bmatrix}$$

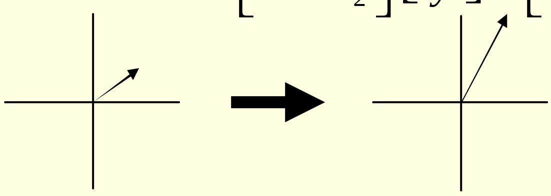
$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad M = s\,I\,P, \quad M = s \begin{bmatrix} x \\ y \end{bmatrix}$$



Non-uniform Scaling

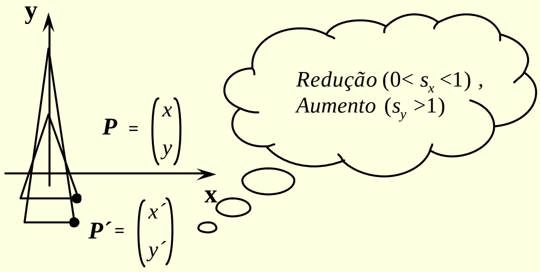
$$S = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \qquad P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$M = S\,P \qquad \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_1\,x \\ s_2\,y \end{bmatrix}$$



Note orientation shift in line

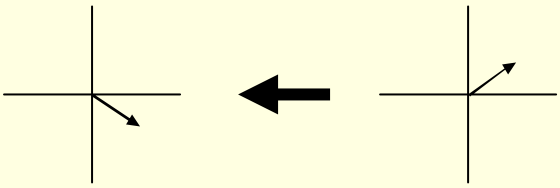
Non-uniform Scaling



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Flip an Axis...

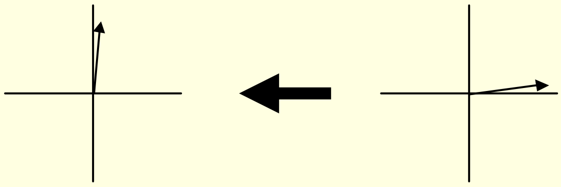
$$\begin{bmatrix} x \\ -y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



What does this do to appearance of objects?

Swap Axes

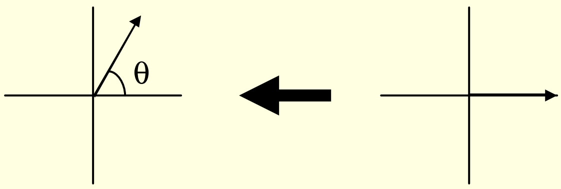
$$\begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



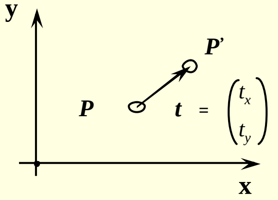
Rotate by θ

$$M = R\,P,\; R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix},\; P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta)\,x - \sin(\theta)\,y \\ \sin(\theta)\,x + \cos(\theta)\,y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Transformações Geométricas (Translação)

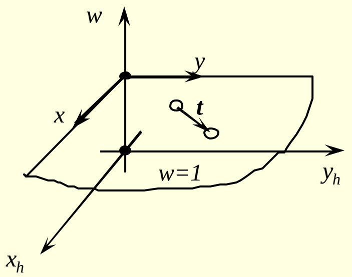


$$\mathbf{P'} = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$ ⇦ *Não pode ser escrito na forma* ☒

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$ ⇦ *Ruim para implementação* ☒

Vantagens das coordenadas homogêneas (Translação)

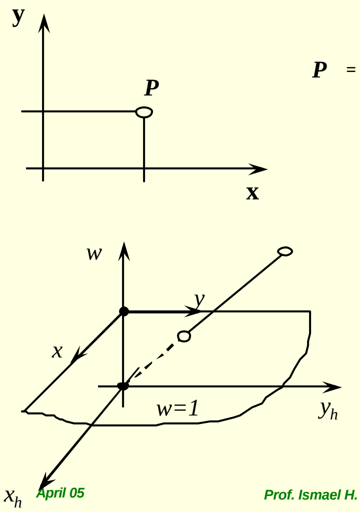


$$\mathbf{P'} = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

[T]

Matrizde Translação

Coordenadas homogêneas



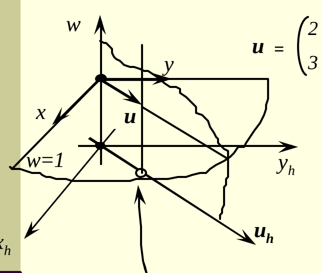
$$P = \begin{pmatrix} x \\ y \end{pmatrix} \cong \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \cong \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} x_h \\ y_h \\ w \end{bmatrix}$$

$$x = x_h / w \qquad y = y_h / w \qquad w > 0$$

Ex.:

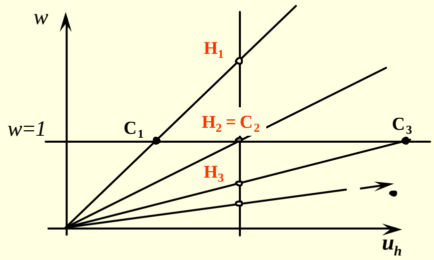
$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \cong \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \cong \begin{bmatrix} 6 \\ 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 9 \\ 6 \\ 3 \end{bmatrix}$$

Vantagens das coordenadas homogêneas (pontos no infinito)



$$u = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$\begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} = \begin{pmatrix} ? \\ ? \end{pmatrix}$$



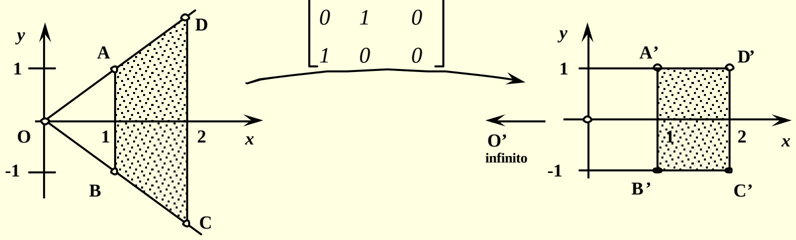
$$H_1 = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix} \quad H_2 = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \quad H_3 = \begin{bmatrix} 2 \\ 3 \\ 1/2 \end{bmatrix} \quad H_4 = \begin{bmatrix} 2 \\ 3 \\ 1/4 \end{bmatrix} \quad \ldots \quad \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}$$

$$C_1 = \begin{pmatrix} 1 \\ 1.5 \end{pmatrix} \quad C_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad C_3 = \begin{pmatrix} 4 \\ 6 \end{pmatrix} \quad C_4 = \begin{pmatrix} 8 \\ 12 \end{pmatrix}$$

infinito na direção (2,3)

Vantagens das coordenadas homogêneas (pontos no infinito, exemplo)

$$\begin{bmatrix} 3 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$



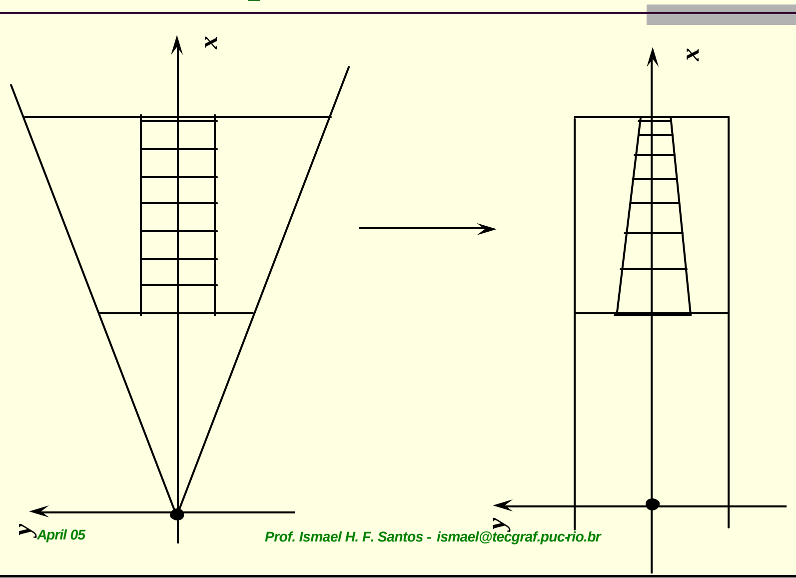
$$\begin{bmatrix} 3 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$ ☺

$$\begin{bmatrix} 3 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$ ☺

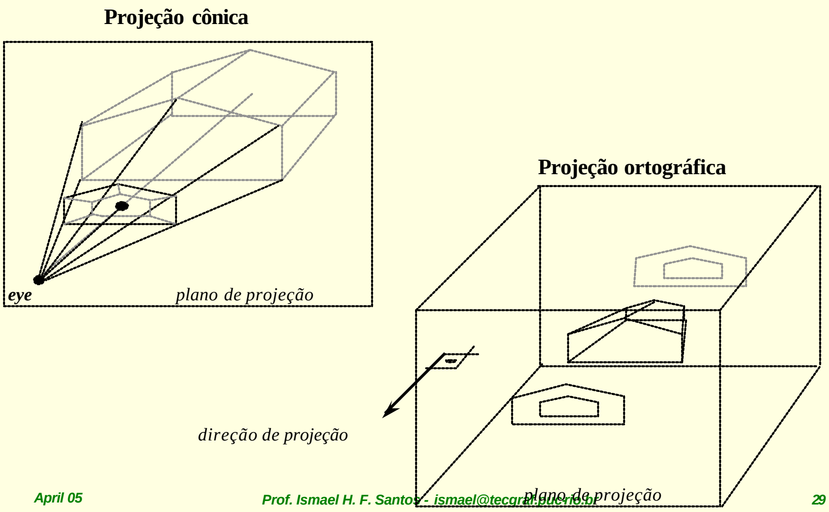
$$\begin{bmatrix} 3 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$ ☺

$$\begin{bmatrix} 3 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$ ☺

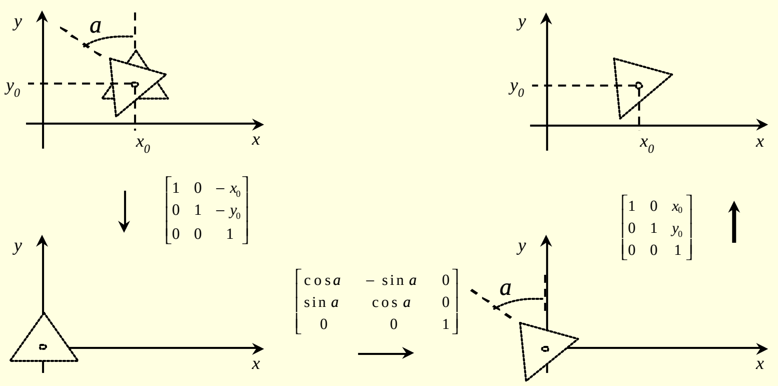
Efeito de profundidade



Simplificação da projeção cônica

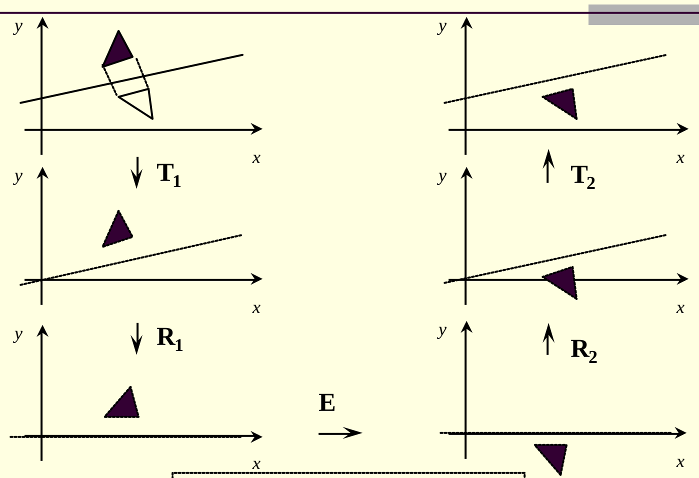


Concatenação



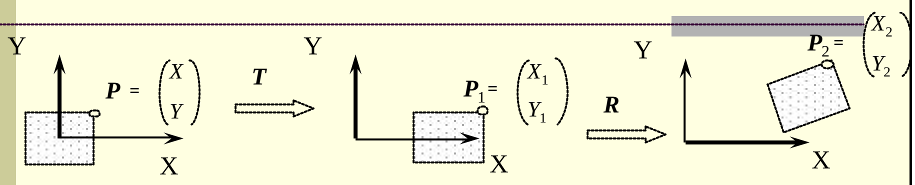
$$\begin{Bmatrix} x' \\ y' \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

Concatenação de Transformações

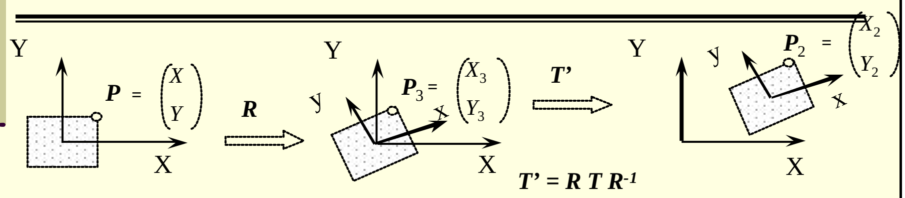


$P' = T_2\ R_2\ E\ R_1\ T_1\ P$

Composição com sistema local/móvel

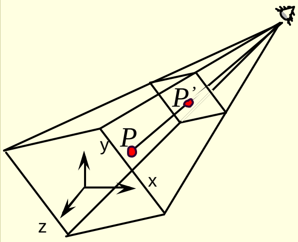


$P_1 = T\,P$ e $P_2 = R\,P_1$ $\Rightarrow$ $P_2 = R\,T\,P$



$P_3 = R\,P$ e $P_2 = T'\,P_3 \Rightarrow P_2 = R\,T\,R^{-1}\,R\,P$ ou $P_2 = R\,T\,P$

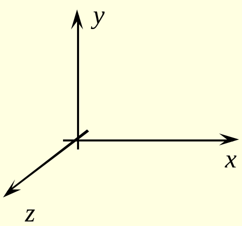
Geometria Projetiva e Coordenadas Homogêneas em 3D



$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x_h/w \\ y_h/w \\ z_h/w \end{pmatrix} \qquad P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

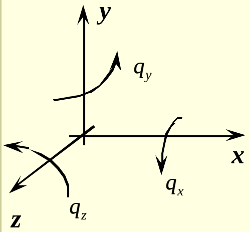
Transformações em 3D (translações e escalas)



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformações em 3D (Rotações)



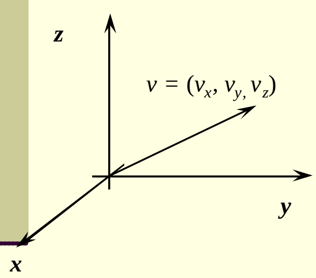
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q_x & -\text{sen}\, q_x & 0 \\ 0 & \text{sen}\, q_x & \cos q_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos q_y & 0 & \text{sen}\, q_y & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}\, q_y & 0 & \cos q_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos q_z & -\text{sen}\, q_x & 0 & 0 \\ \text{sen}\, q_x & \cos q_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

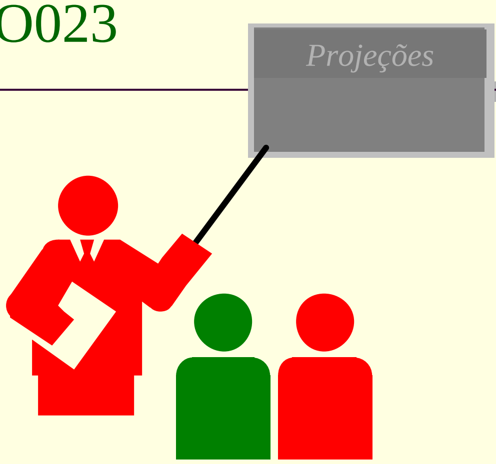
Transformações em 3D (rotação em torno de um eixo qualquer)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

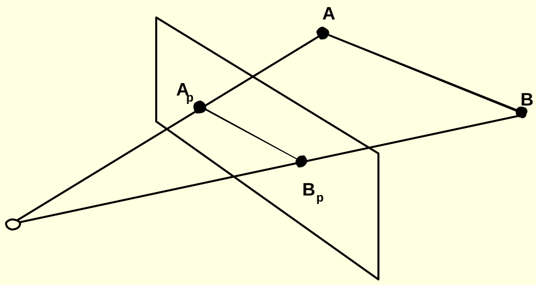


$$\begin{aligned}
m_{11} &= v_x^2 + \cos q\ (1 - v_x^2) \\
m_{12} &= v_x v_y (1 - \cos q\ ) - v_z\, \text{sen}\, q \\
m_{13} &= v_z v_x (1 - \cos q\ ) + v_y\, \text{sen}\, q \\
m_{21} &= v_x v_y (1 - \cos q\ ) + v_z\, \text{sen}\, q \\
m_{22} &= v_y^2 + \cos q\ (1 - v_y^2) \\
m_{23} &= v_y v_z (1 - \cos q\ ) - v_x\, \text{sen}\, q \\
m_{31} &= v_x v_z\ (1 - \cos q\ ) - v_y\, \text{sen}\, q \\
m_{32} &= v_y v_z (1 - \cos q\ ) + v_x\, \text{sen}\, q \\
m_{22} &= v_z^2 + \cos q\ (1 - v_z^2)
\end{aligned}$$

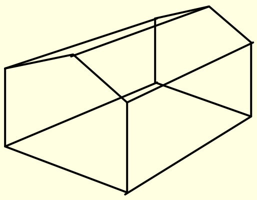
CG – CO023



Projeções Planas Cônicas

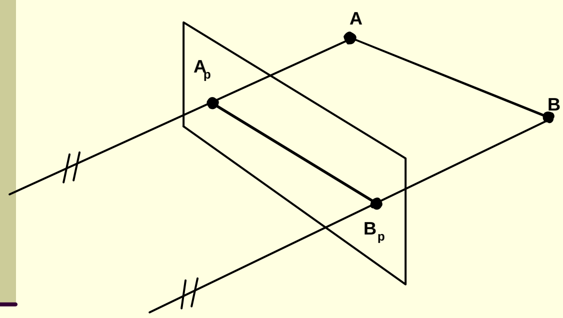


☺ *realista*



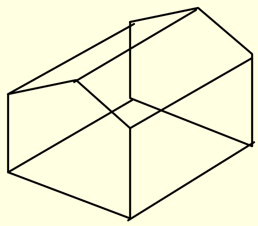
☹ *não preserva escala*
☹ *não preserva ângulos*

Projeções Planas Paralelas



☹ *pouco realista*

☺ *preserva paralelismo*
☺ *possui escala conhecida*



April 05

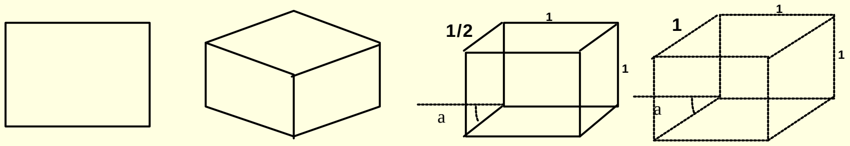
Classificação das projeções planas

- **Paralelas**
  - **ortográficas** dp // n
    - **plantas**
    - **elevações**
    - **iso-métrica**
  - **oblíquas** dp **não é paralela a** n
    - **cavaleiras**
    - ***cabinet***
- **Cônicas**
  - **1 pto de fuga**
  - **2 ptos de fuga**
  - **3 ptos de fuga**

April 05

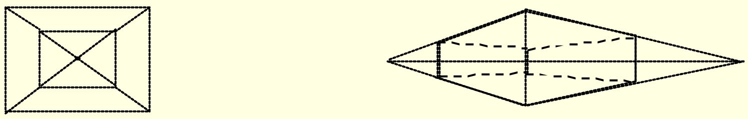
Projeções de um cubo

- **Paralelas**



planta ou elevação **iso-métrica** **Cabinete (a =45 ou 90)** **Cavaleira (a =45 ou 90)**

- **Cônicas**

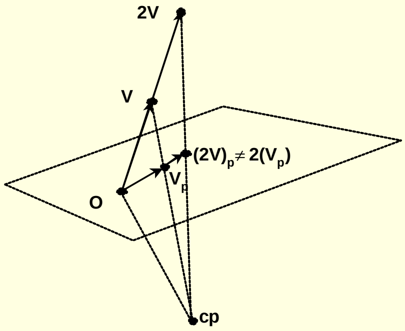


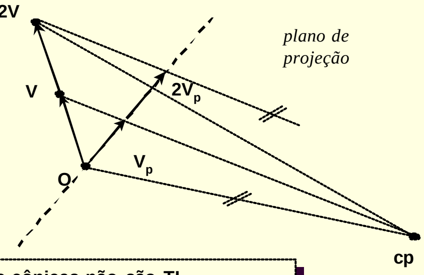
1 pto de fuga **2 ptos de fuga**

Projeção plana é uma transformação linear?

Ou seja:

T(P+Q) = T(P)+T(Q) e
T(a P) = a T(P) ?

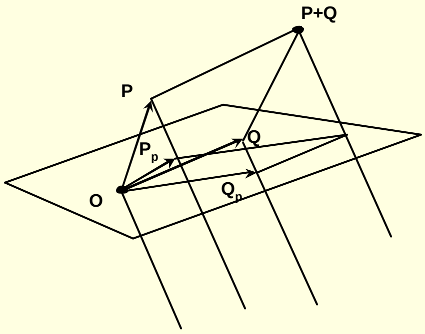




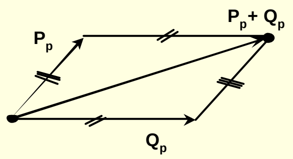
Projeções cônicas não são TL, paralelas podem ser.

Projeção plana paralela é uma transformação linear?

T(P+Q) = T(P)+T(Q) ?



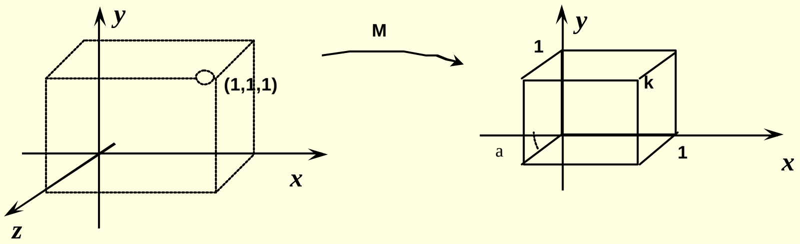
retas paralelas
projetam em paralelas



T(0) = 0 ?

Projeção paralela em plano que passa pela origem é uma transformação linear

Matrizes de projeções Cavaleiras e Cabinetes



$Â^3$ $Â^2$

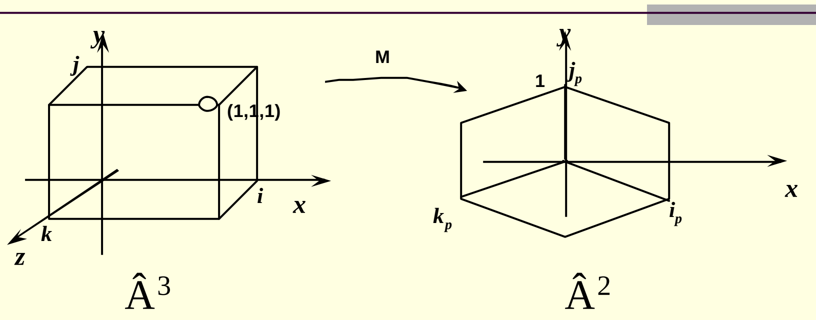
T(1,0,0) = (1,0)

T(0,1,0) = (0,1)

T(0,0,1) = (-k cos a , -k sin a)

$$M = \begin{vmatrix} 1 & 0 & -k.\cos(a) \\ 0 & 1 & -k.\sin(a) \end{vmatrix}$$

Matrizes de projeções pseudo-isométricas



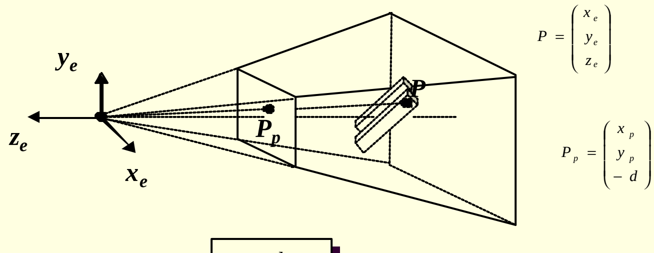
T(1,0,0) = (cos 30 ,-sin 30)

T(0,1,0) = (0,1)

T(0,0,1) = (-cos 30, -sin 30)

$$M = \begin{vmatrix} \cos 30 & 0 & -\cos 30 \\ -\sin 30 & 1 & -\sin 30 \end{vmatrix}$$

Projeção cônica simples



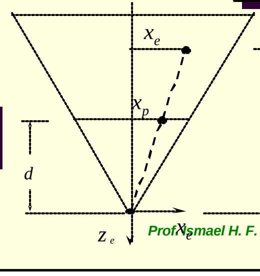
$$P = \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix}$$

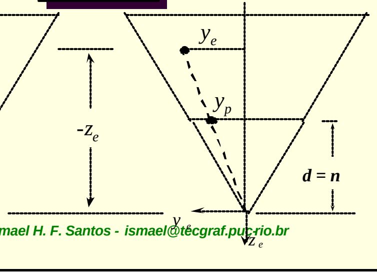
$$P_p = \begin{pmatrix} x_p \\ y_p \\ -d \end{pmatrix}$$

$z_p = -d$

$$\frac{x_p}{x_e} = \frac{d}{-z_e}$$

$$x_p = \frac{d}{-z_e}\, x_e$$

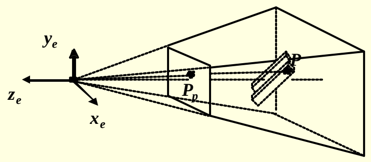




$$\frac{y_p}{y_e} = \frac{d}{-z_e}$$

$$y_p = \frac{d}{-z_e}\, y_e$$

Projeção cônica simples



$$x_p = \frac{d}{-z_e}\, x_e$$

$$y_p = \frac{d}{-z_e}\, y_e$$

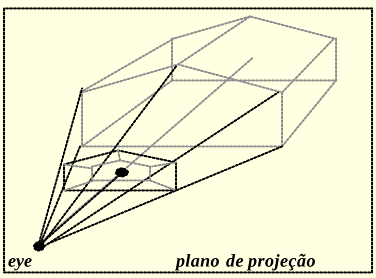
$$z_p = -d$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} d\,x_e \\ d\,y_e \\ d\,z_e \\ -z_e \end{bmatrix} = \begin{bmatrix} (d/-z_e)\,x_e \\ (d/-z_e)\,y_e \\ -d \\ 1 \end{bmatrix}$$

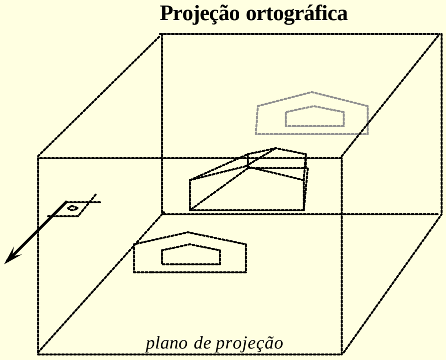
w $\div\, w$

Simplificação da projeção cônica

Projeção cônica

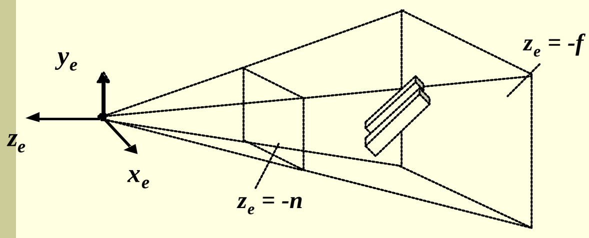


Projeção ortográfica



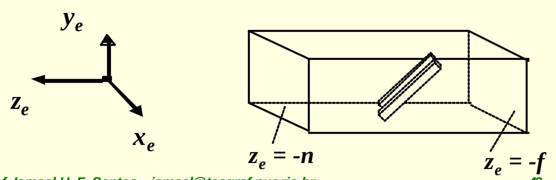
direção de projeção

Distorce o frustum de visão para o espaço da tela



$d = n$

$$[\,P\,] = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



Uma equação para profundidade

$$z_e = a - \frac{b}{z}$$

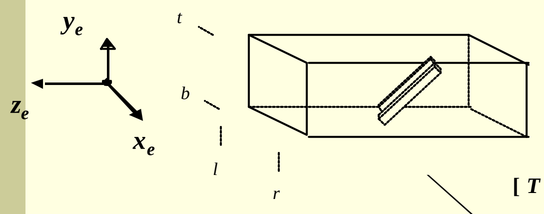
Ptos no near (z=-n): $$-n = a + \frac{b}{n}$$

Ptos no far (z=-f): $$-f = a + \frac{b}{f}$$

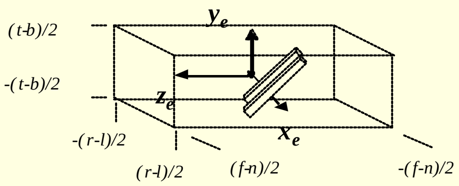
$$\Rightarrow \begin{cases} a = f + n \\ b = f \cdot n \end{cases}$$

$$[\,P\,] = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & n \bullet f \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

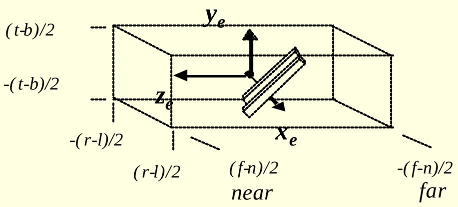
Translada o paralelepípedo de visão para origem



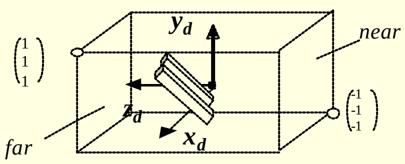
$$[T] = \begin{bmatrix} 1 & 0 & 0 & -(r+l)/2 \\ 0 & 1 & 0 & -(t+b)/2 \\ 0 & 0 & 1 & +(f+n)/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Escala o paralelepípedo de visão no cubo [-1,1]x[-1,1]x[-1,1]



$$[S] = \begin{bmatrix} 2/(r-l) & 0 & 0 & 0 \\ 0 & 2/(t-b) & 0 & 0 \\ 0 & 0 & -2/(f-n) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Matriz Frustum do OpenGL

$$[\,P\,] = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & n \bullet f \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$[\,T\,] = \begin{bmatrix} 1 & 0 & 0 & -(r+l)/2 \\ 0 & 1 & 0 & -(t+b)/2 \\ 0 & 0 & 1 & +(f+n)/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[S] = \begin{bmatrix} 2/(r-l) & 0 & 0 & 0 \\ 0 & 2/(t-b) & 0 & 0 \\ 0 & 0 & -2/(f-n) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

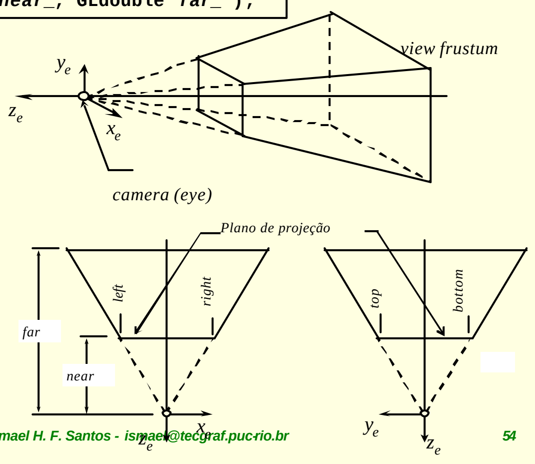
$$[\,S\,T\,P\,] = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

OpenGL Spec

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

Projeção Cônica (Frustum)

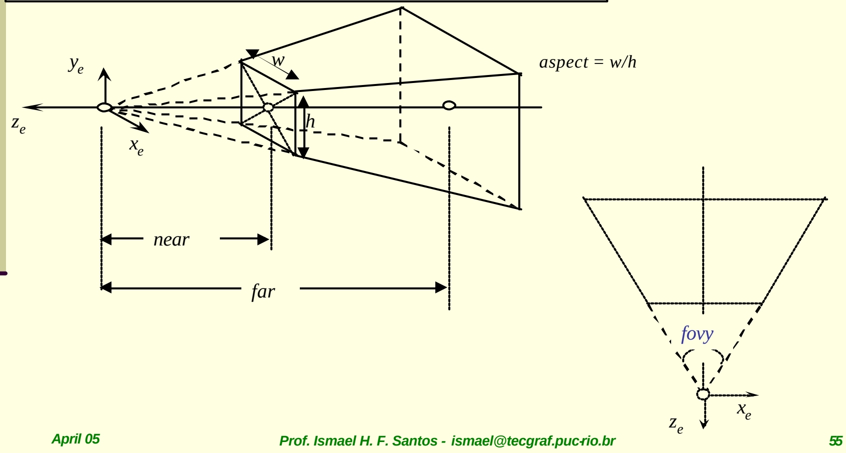
```
void glFrustum( GLdouble left, GLdouble right,
                GLdouble bottom, GLdouble top,
                GLdouble near_, GLdouble far_ );
```



Obs.: *near* e *far* são distâncias(> 0)

Projeção Cônica (Perspective)

```
void glPerspective( GLdouble fovy, GLdouble aspect,
                    GLdouble near_, GLdouble far_ );
```



Matriz Ortho do OpenGL

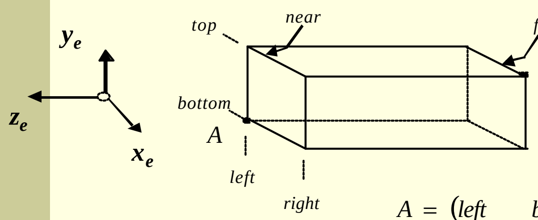
$$[\,T\,] = \begin{bmatrix} 1 & 0 & 0 & -(r+l)/2 \\ 0 & 1 & 0 & -(t+b)/2 \\ 0 & 0 & 1 & +(f+n)/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

OpenGL Spec

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

$$[S] = \begin{bmatrix} 2/(r-l) & 0 & 0 & 0 \\ 0 & 2/(t-b) & 0 & 0 \\ 0 & 0 & -2/(f-n) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projeção Paralela (Ortho)



$B = (right \quad top \quad -far)$

$A = (left \quad bottom \quad -near)$

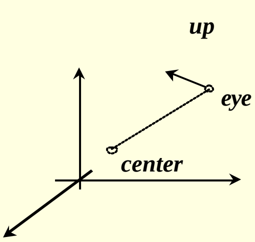
```
void glOrtho( GLdouble left, GLdouble right,
              GLdouble bottom, GLdouble top,
              GLdouble near_, GLdouble far_ );
```

```
void gluOrtho2D( GLdouble left, GLdouble right,
                 GLdouble bottom, GLdouble top );
```

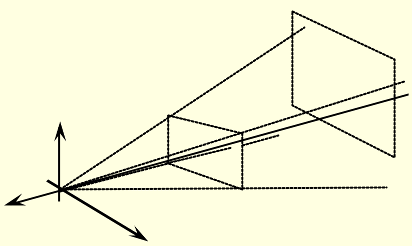

Glu Look At

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

Dados: ***eye***, ***ref***, ***up*** *(definem o sistema de coordenadas do olho)*
Determine a matriz que leva do sistema de Coordenadas dos Objetos para o sistema de Coordenadas do Olho

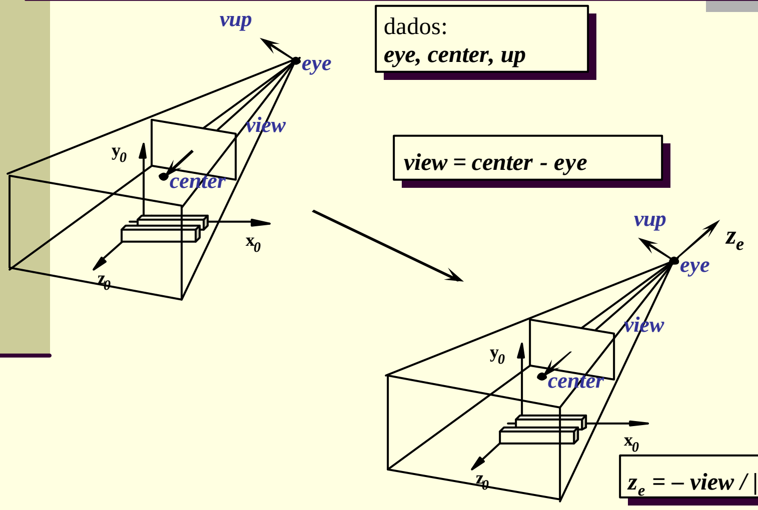


Coordenadas dos Objetos



Coordenadas do Olho

Calcula o sistema - $x_e\ y_e\ z_e$

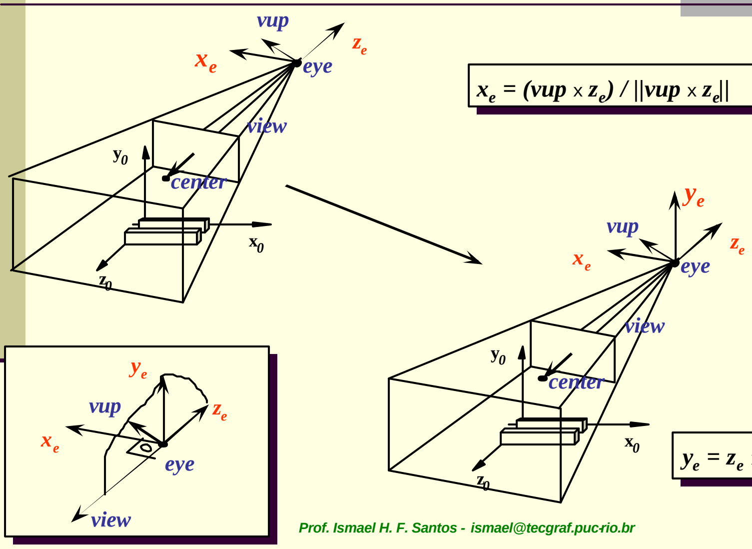


dados:
eye, center, up

view = center - eye

$z_e = -\, view \,/\, \|view\|$

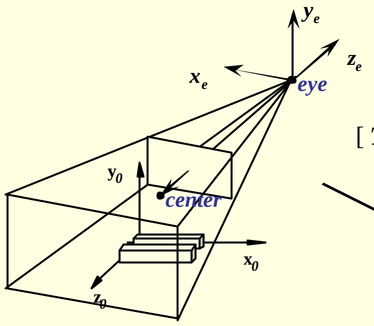
Calcula o sistema - $x_e\ y_e\ z_e$



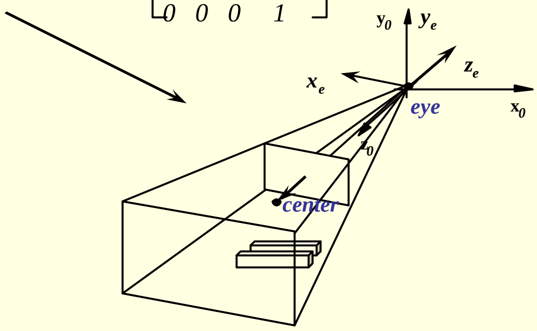
$x_e = (vup \times z_e) \,/\, \|vup \times z_e\|$

$y_e = z_e \times x_e$

Translada o *eye* para origem

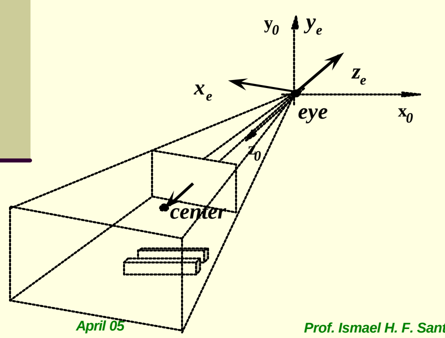


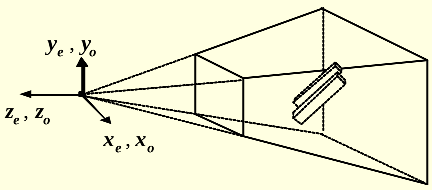
$$[\boldsymbol{T}] = \begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Roda $x_e\, y_e\, z_e$ para $x_o\, y_o\, z_o$

$$[\boldsymbol{R}] = \begin{bmatrix} x_{ex} & x_{ey} & x_{ez} & 0 \\ y_{ex} & y_{ey} & y_{ez} & 0 \\ z_{ex} & z_{ey} & z_{ez} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$





Matriz Look At do OpenGL

$$[\,T\,] = \begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$z_e = -\, view \,/\, \|view\|$

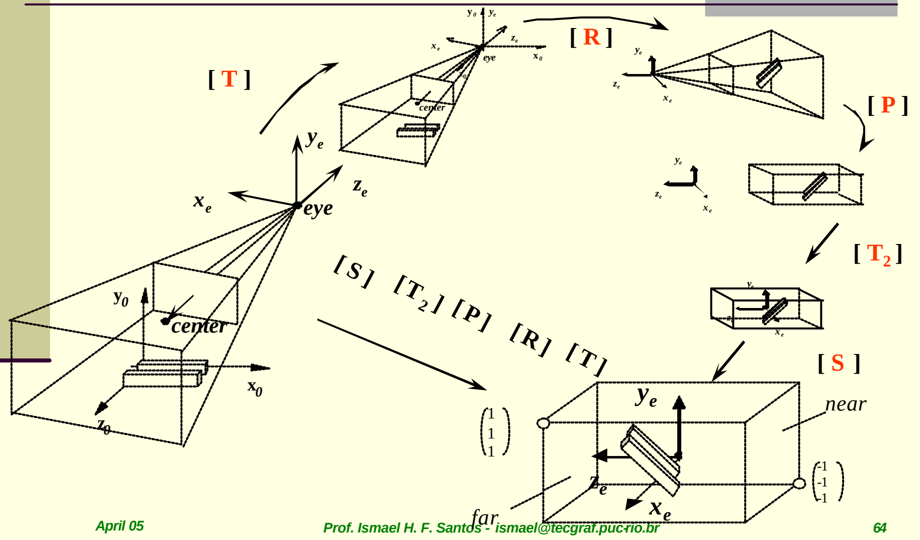
$x_e = (vup \times z_e) \,/\, \|vup \times z_e\|$

$y_e = z_e \times x_e$

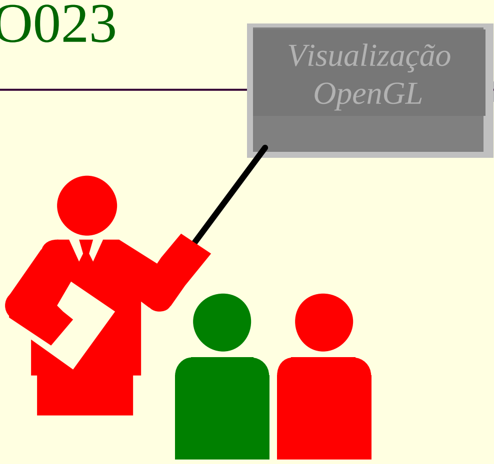
$[\,R\,T\,] = ?$

$$[\,R\,] = \begin{bmatrix} x_{ex} & x_{ey} & x_{ez} & 0 \\ y_{ex} & y_{ey} & y_{ez} & 0 \\ z_{ex} & z_{ey} & z_{ez} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Concatenação das transformações

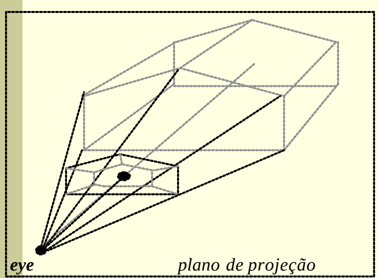


CG – CO023

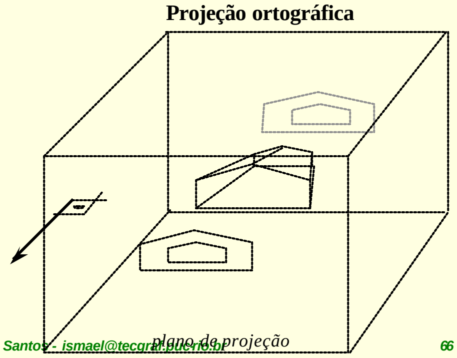


Simplificação da projeção cônica

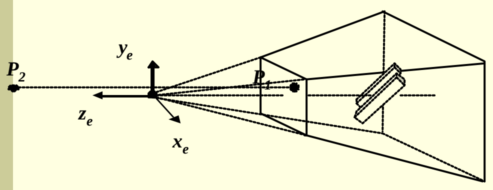
Projeção cônica



Projeção ortográfica



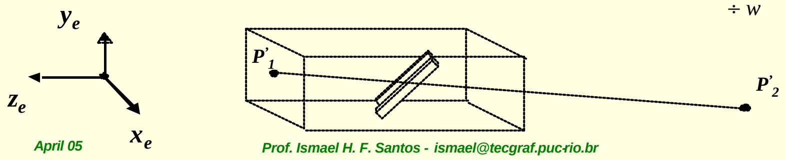
Problema do *clipping*



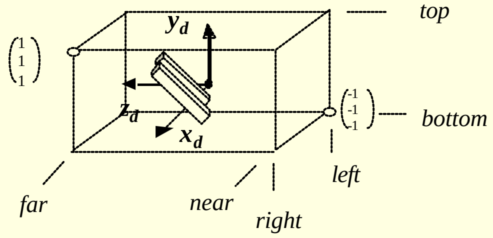
$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & n \bullet f \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -n \\ 1 \end{bmatrix} = \begin{bmatrix} n \\ n \\ -n^2 \\ n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -n \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & n \bullet f \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} n \\ n \\ n^2+2nf \\ -n \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -n-2f \\ 1 \end{bmatrix}$$

$\div w$

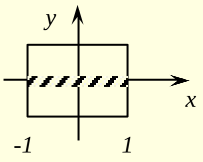


Clipping em coordenadas homogêneas



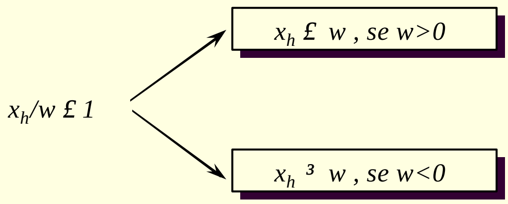
x Î [left, right]	-1 £ x £ 1	-1 £ x_h/w £ 1
y Î [bottom, top]	-1 £ y £ 1	-1 £ y_h/w £ 1
z Î [near, far]	-1 £ z £ 1	-1 £ z_h/w £ 1

Clipping em coordenadas homogêneas



x Î [*left, right*]

-1 £ x_h/w £ 1

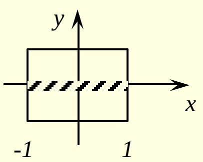


OpenGL Spec

Primitives are clipped to the *clip volume*. In clip coordinates, the *view volume* is defined by

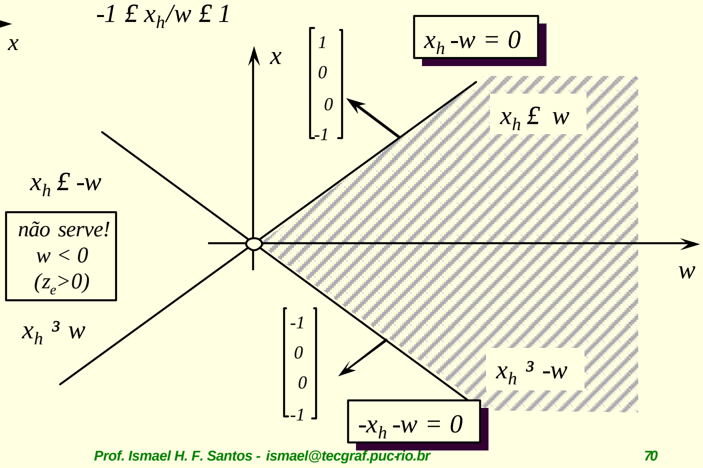
$$-w_c \leq x_c \leq w_c$$
$$-w_c \leq y_c \leq w_c \quad .$$
$$-w_c \leq z_c \leq w_c$$

Clipping em coordenadas homogêneas



x Î [*left, right*]

-1 £ x_h/w £ 1



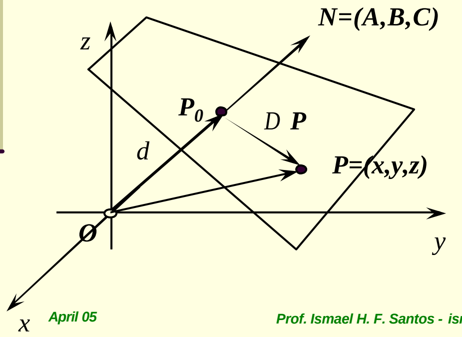
Equação de um plano

$N.P = Ax + By + Cz$

$N.P = N.(P_0 + D\ P) = N.P_0 = d$

$(N.D P = 0)$

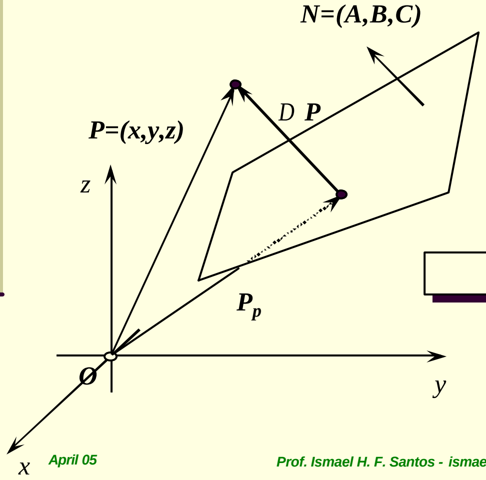
$d = Ax + By + Cz$



$Ax + By + Cz + D = 0$

$(A, B, C) = N$
e
$D = -d = N.(-P_0)$

Distância de um ponto a um plano



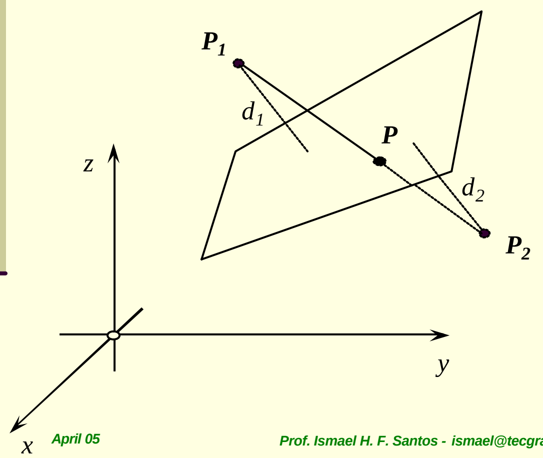
$N.P = Ax + By + Cz$

$N.P = N.(P_p + D\ P)$

$N.P = d + N.D\ P$

$N.D\ P = Ax + By + Cz + D$

Interseção de reta com plano



$d_1 = | Ax_1 + By_1 + Cz_1 + D |$

$d_2 = | Ax_2 + By_2 + Cz_2 + D |$

$$P = \frac{d_1 \, P_2 + d_2 \, P_1}{d_1 + d_2}$$

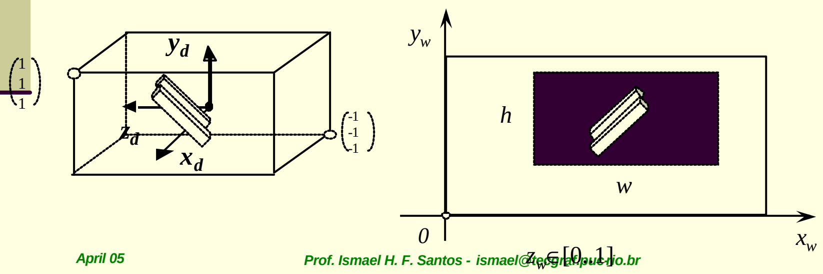
Cálculo das distâncias

```
/* ====================  Distance  =====================
**
** This function computes and returns the distance between a
** point and a plane. Normal points toward out.
*/
double Distance(double x, double y, double z, double w, int plane )
{
 switch( plane )
 {
  case 0: return( -w - x );
  case 1: return( -w + x );
  case 2: return( -w - y );
  case 3: return( -w + y );
  case 4: return( -w - z );
  case 5: return( -w + z );
 }
 return( 0.0 );
}
```

Transformação para o Viewport

```
void glViewport(GLint x0, GLint y0,
                GLsizei width, GLsizei height );
```

$$x_w = x_0 + w*(x_d - (-1)) / 2$$
$$y_w = y_0 + h*(y_d - (-1)) / 2$$
$$z_w = z_d / 2 + 1/2$$



Transformações de um vértice

OpenGL Spec

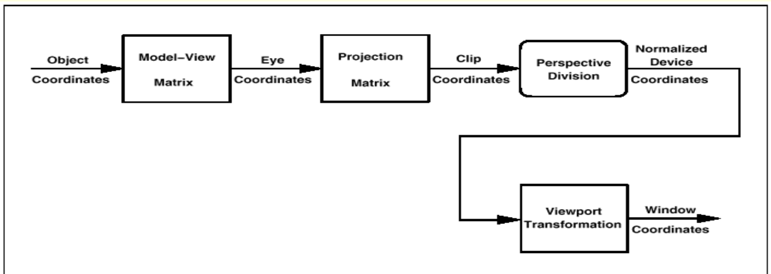
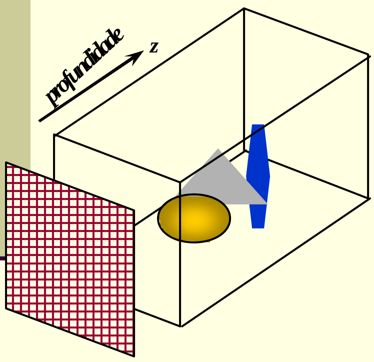
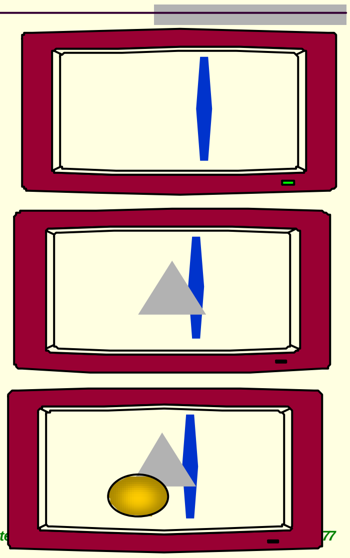


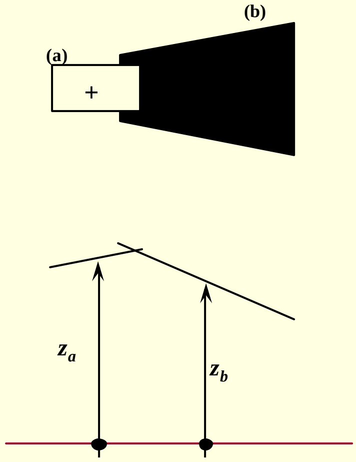
Figure 2.6. Vertex transformation sequence.

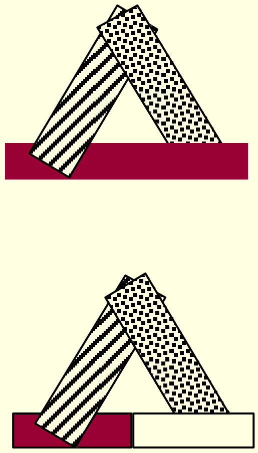
Modelo do Pintor



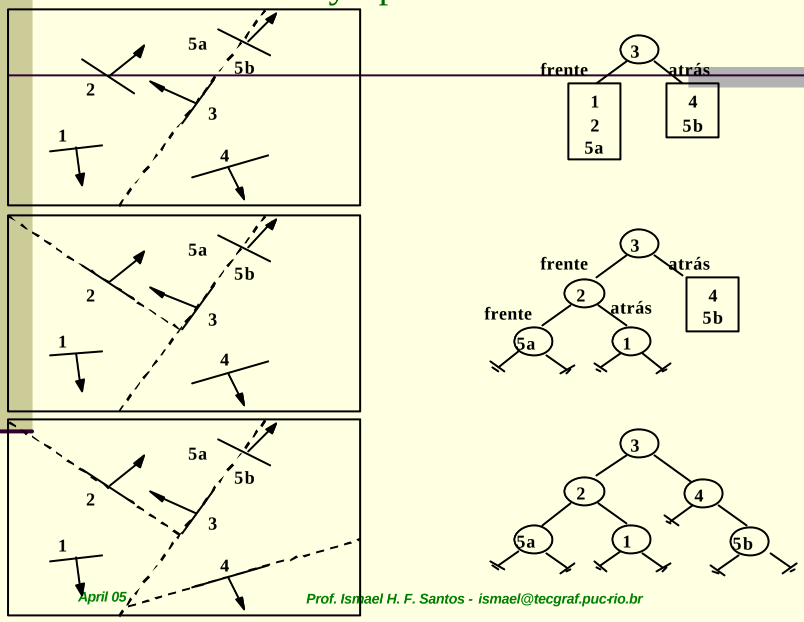


Problemas na ordenação de faces





BSP trees: Binary Space Partion Trees



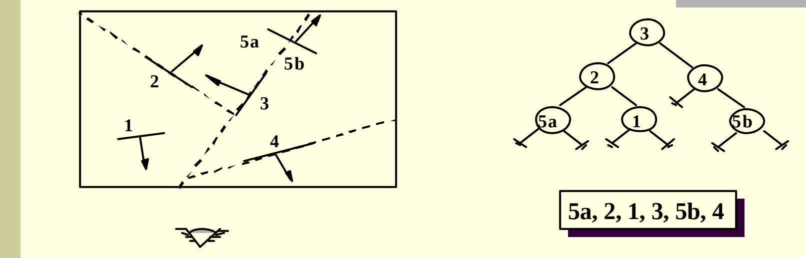
Exibição de uma BSP

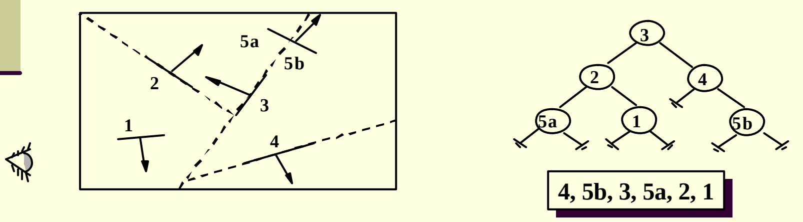
```
void bspDisplay(bspTree *tree) {
  if (arvore não é vazia) {
    if (observador está a frente da raiz) {
      bspDisplay(tree®backChild);
      DisplayPolygon(tree®root);
      bspDisplay(tree®frontChild);
    }
    else {
      bspDisplay(tree®frontChild);
      DisplayPolygon(tree®root);
      bspDisplay(tree®backChild);
    }
  }
}
```

Mostra a árvore de trás, a raiz e a árvore da frente.

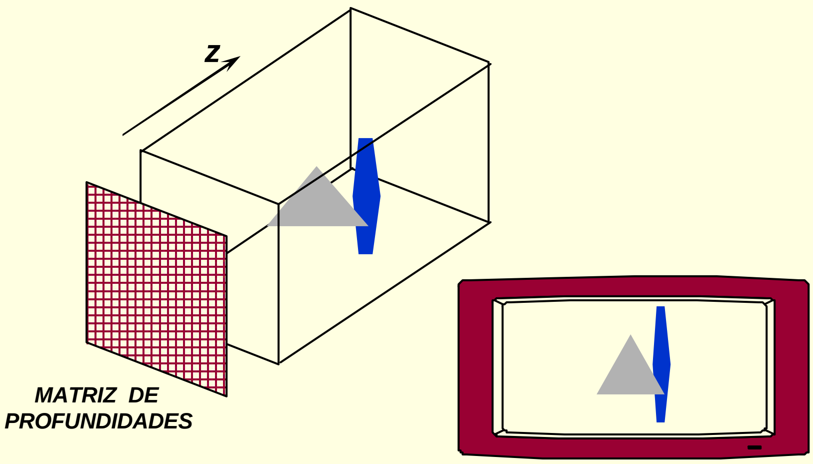
Mostra a árvore da frente, a raiz e a árvore de atrás.

BSP trees: Dois exemplos de exibição

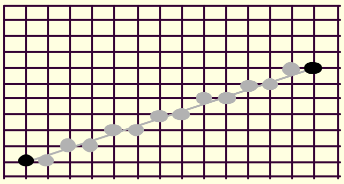


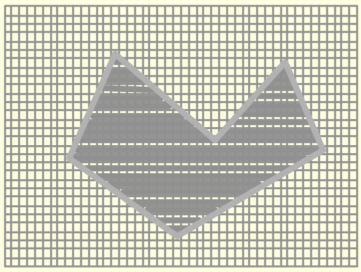


ZBuffer: idéia básica



Rasterização de Polígonos e Linhas





ZBuffer - pseudo-código

```
void ZBuffer( void) {
 int x,y;

 for (x=0; x<w; x++) {
   for (y=0;y<h; y++) {
        WritePixel(x,y, bck_color);
        WriteZ(x,y,0);
   }
 }

  for (each primitive) {
    for (each pixel in the projected primitive) {
       double pz = z coordinate of the (x,y) pixel;
       if  (pz <= ReadZ(x,y)) {
          WritePixel(x,y, color);
          WriteZ(x,y,pz);
        }
    }
  }

} /* Zbuffer */
```

```
void glEnable( GL_DEPTH_TEST );
```