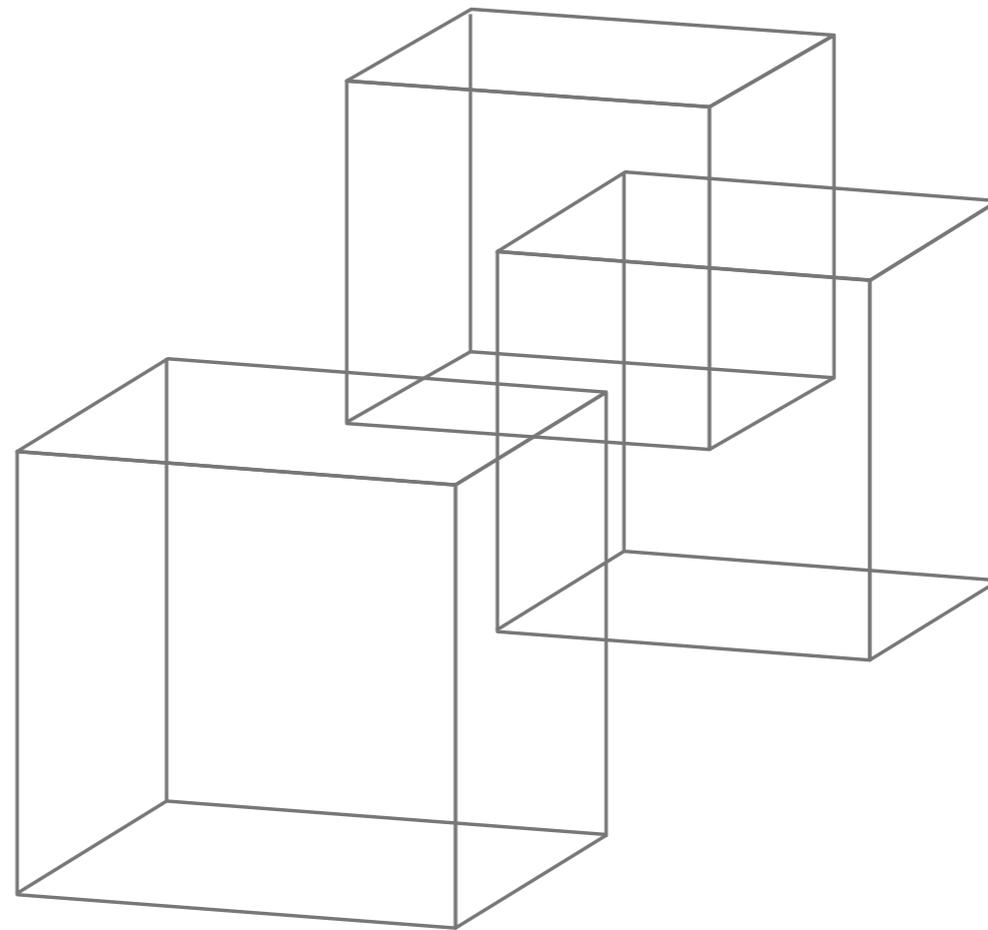


# Hidden Line Hidden Surface Removal

# HLHSR

## Necessidade de Algoritmos para Hidden-Line Hidden-Surface Removal\*



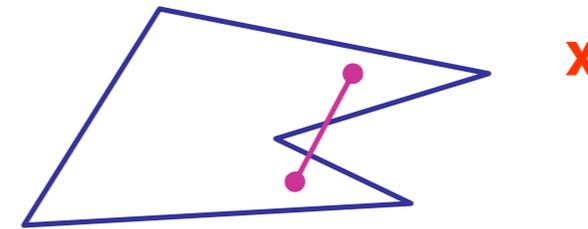
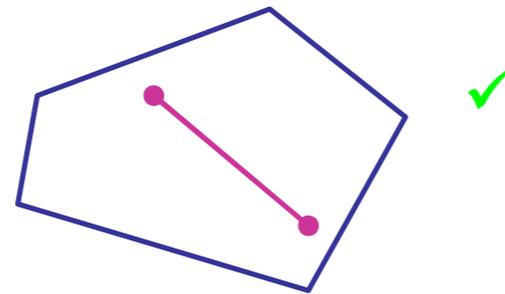
\* Remoção de partes ocultas (linhas e/ou superfícies)

*M. Próspero*

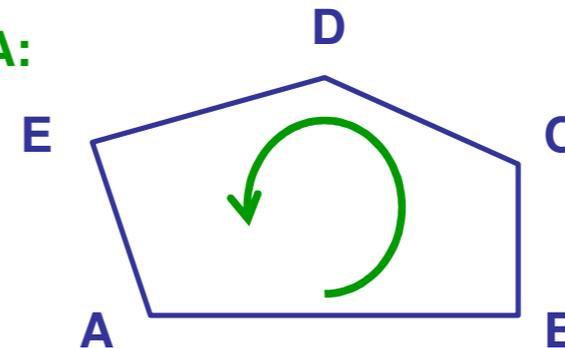
# Considerações Preliminares (I)

## POLÍGONO CONVEXO:

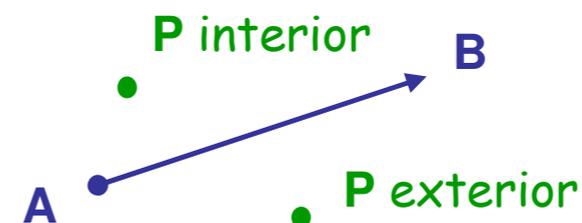
Aquele em que uma linha unindo dois quaisquer pontos interiores ao polígono esteja nele totalmente contida.



## POLÍGONO (CONVEXO) COM ORIENTAÇÃO POSITIVA:



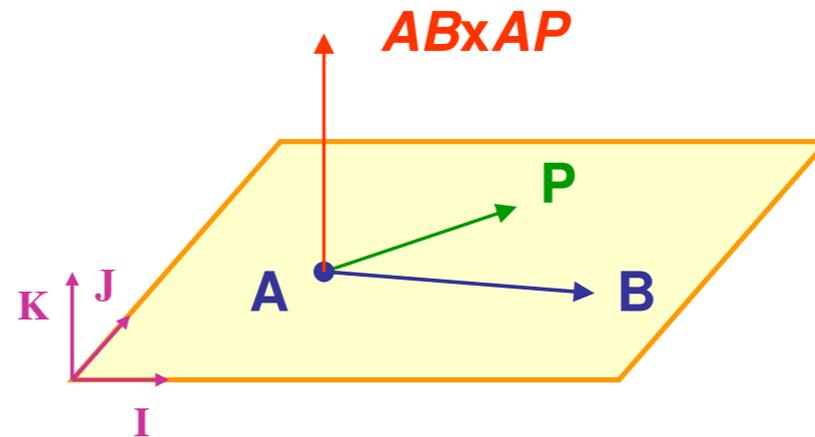
Um ponto interior a um polígono convexo com orientação positiva fica à esquerda de todas as arestas do mesmo.



M. Próspero

# Considerações Preliminares (II)

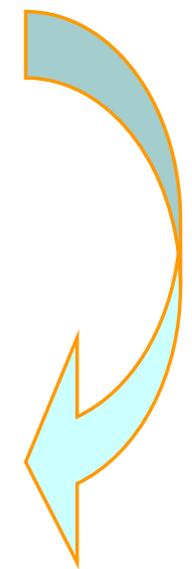
Determinação da posição de um ponto **P** via produto externo de dois vetores:



$$AB \times AP = \begin{vmatrix} \mathbf{I} & \mathbf{J} & \mathbf{K} \\ (x_B - x_A) & (y_B - y_A) & 0 \\ (x_P - x_A) & (y_P - y_A) & 0 \end{vmatrix} = ((x_B - x_A)(y_P - y_A) - (x_P - x_A)(y_B - y_A)) \mathbf{K} = k \mathbf{K}$$

$k > 0 \Rightarrow \mathbf{P}$  à esquerda de  $AB$

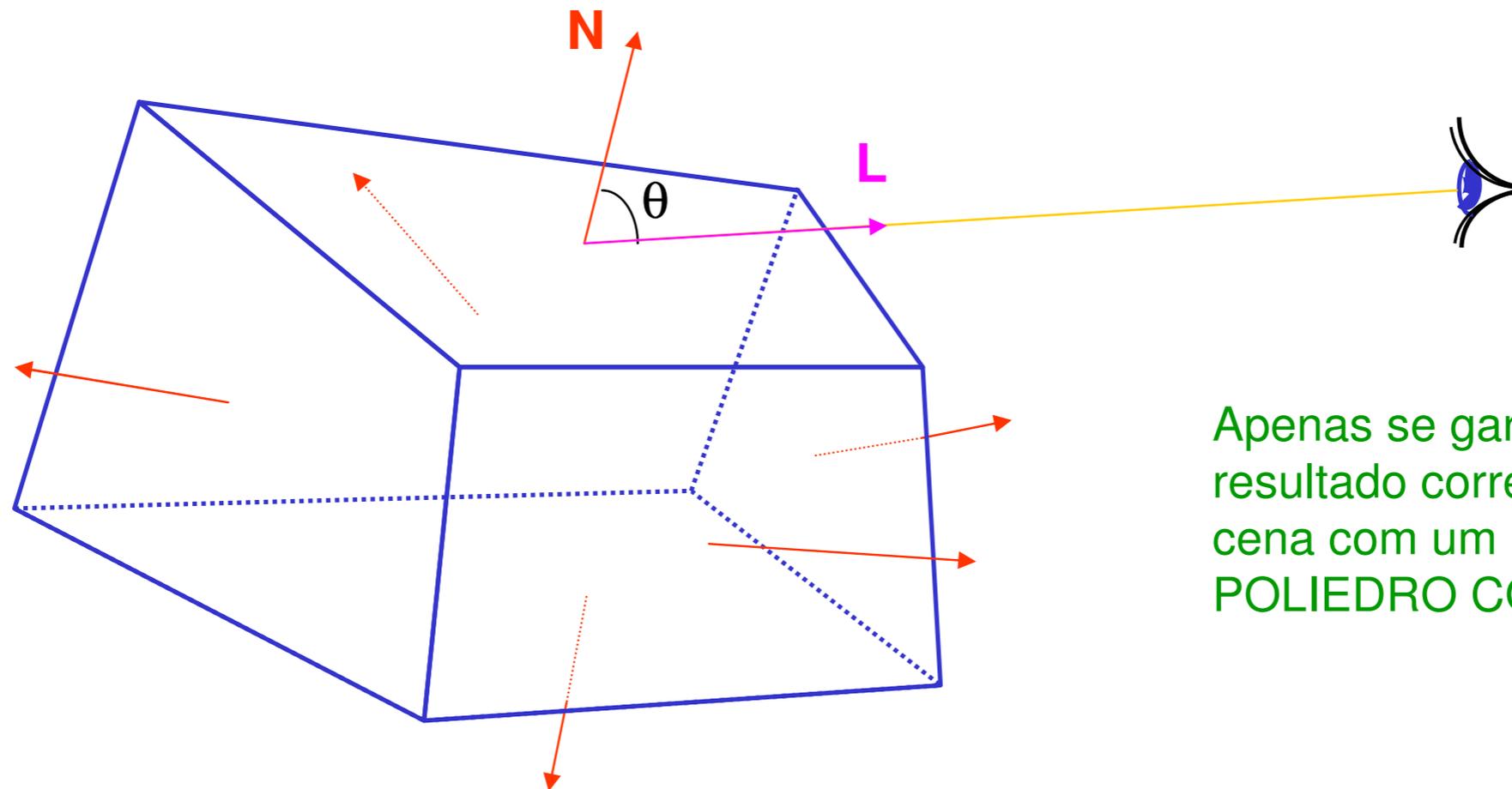
$k < 0 \Rightarrow \mathbf{P}$  à direita de  $AB$



M. Próspero

# Método do Produto Interno

## Test for Culling Polygon Faces



Apenas se garante um resultado correto numa cena com um e um só POLIEDRO CONVEXO

A superfície não é visível  $\longrightarrow 90^\circ \leq \theta \leq 180^\circ$

A superfície é visível  $\longrightarrow 0^\circ \leq \theta < 90^\circ \longrightarrow 0 < \cos \theta \leq 1$

$$\cos \theta = \frac{L \cdot N}{|L| |N|}$$

em que

$$L \cdot N = x_L x_N + y_L y_N + z_L z_N$$

M. Próspero

# Método do Produto Interno

## Cálculo dos vetores normais:

Polígonos com orientação positiva quando observados do exterior do poliedro

Tomando 3 vértices consecutivos:

$$N = (V_2 - V_1) \times (V_3 - V_2)$$

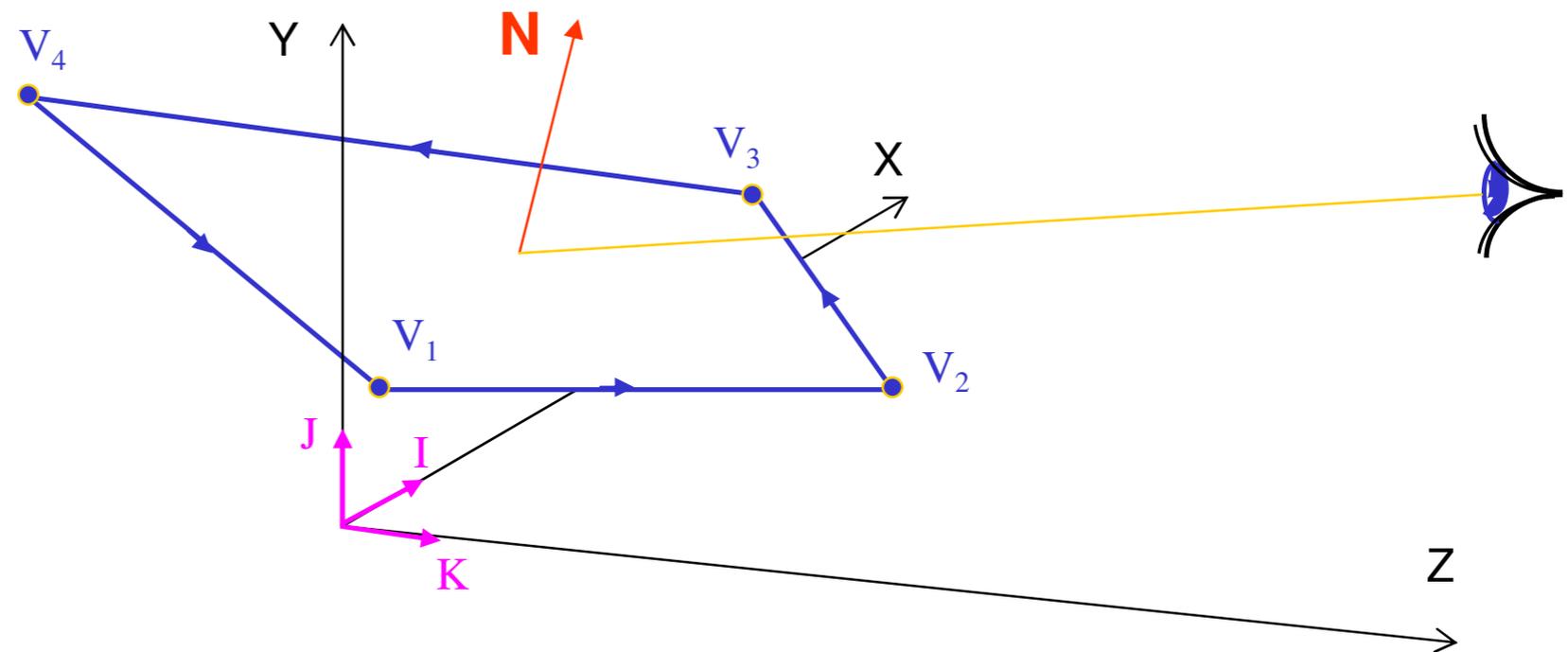
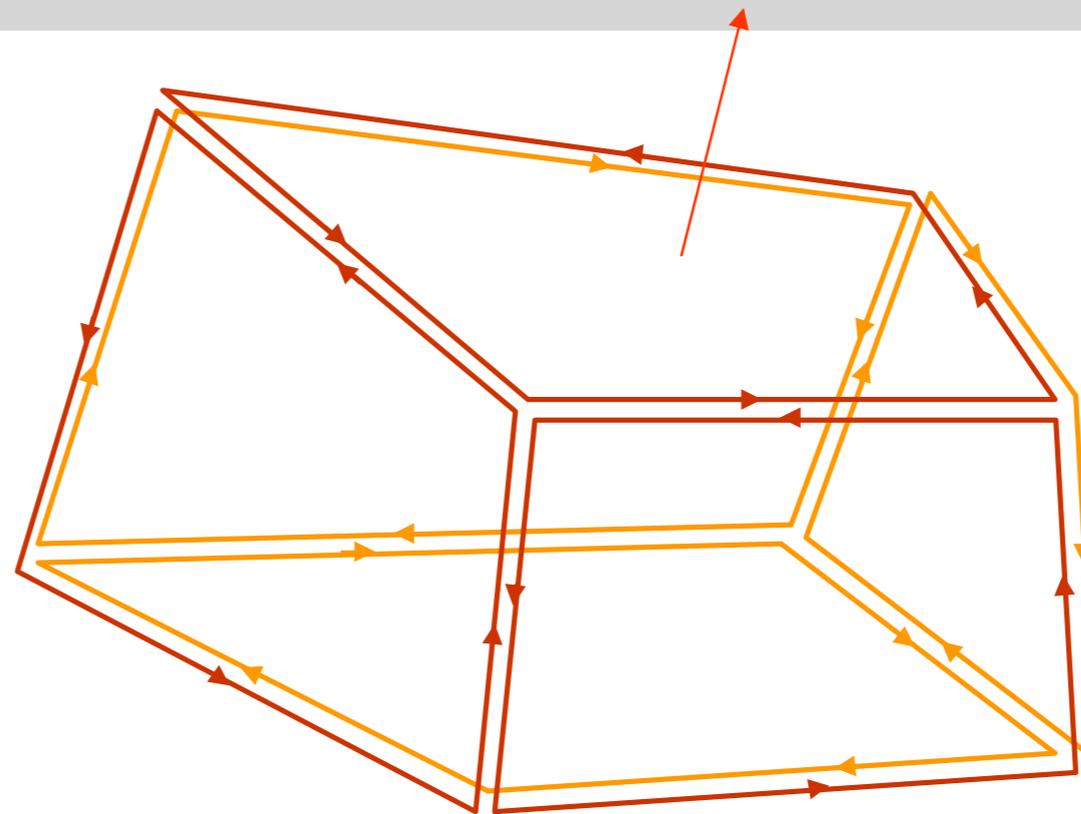
$$= \begin{vmatrix} \mathbf{I} & \mathbf{J} & \mathbf{K} \\ x_{21} & y_{21} & z_{21} \\ x_{32} & y_{32} & z_{32} \end{vmatrix}$$

em que

$$x_{ji} = x_j - x_i$$

$$y_{ji} = y_j - y_i$$

$$z_{ji} = z_j - z_i$$



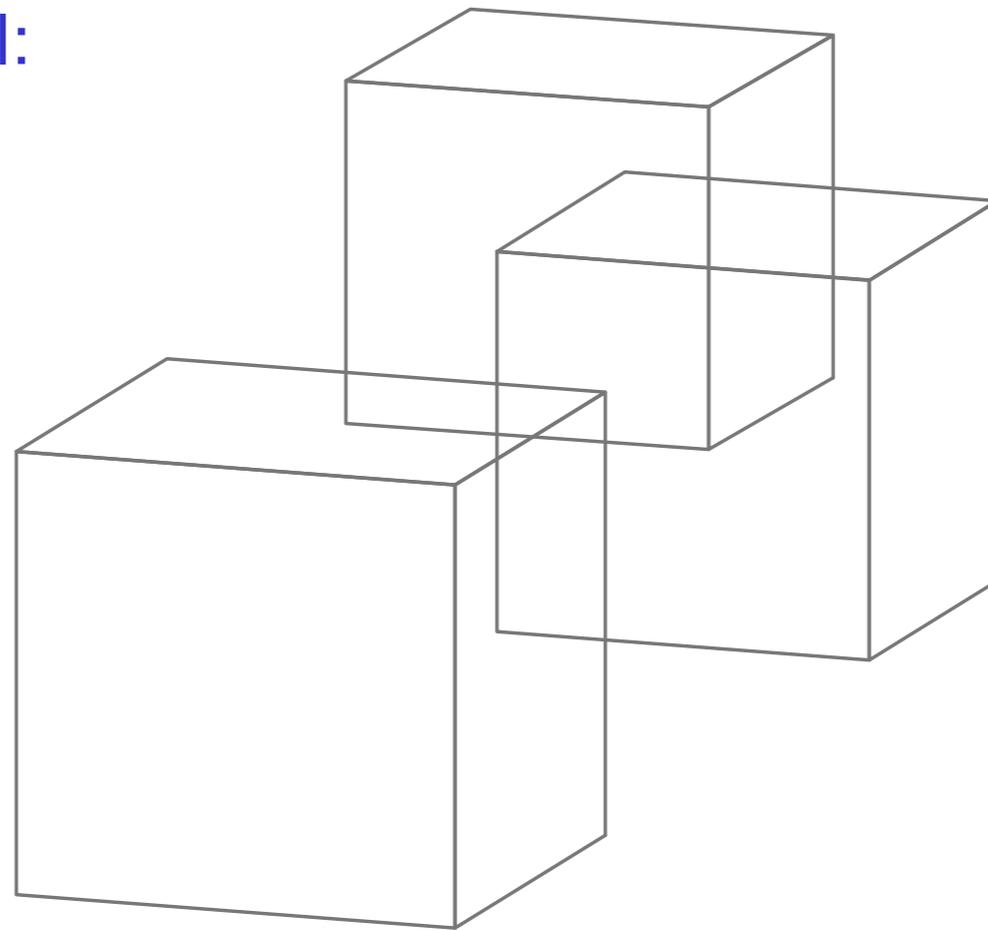
**OBS.:** Na **projeção ortogonal** bastará investigar o sinal de uma das componentes de  $N$  (ou seja: para a projeção em  $XY$ , se  $N_z > 0$  então o polígono será visível)

M. Próspero

# Método do Produto Interno

## Aplicação do método de *Culling* de faces para Hidden-Line Removal

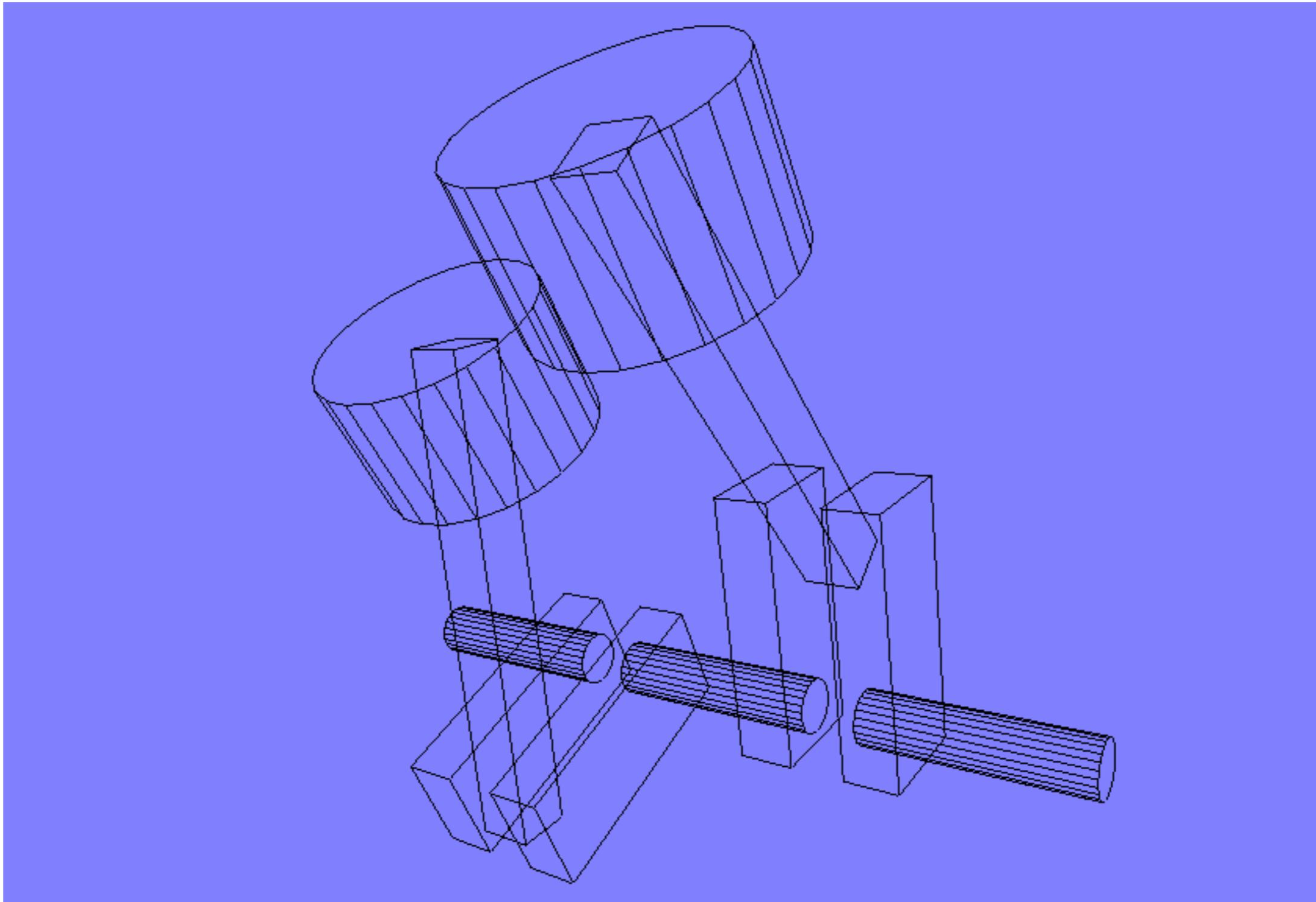
Resultado final:



Em OpenGL aplica-se o método usando `glEnable(GL_CULL_FACE)`,  
podendo definir-se o tipo de faces a ocultar (ie, *front* ou *back*) via `glCullFace()`

M. Próspero

# Exemplo



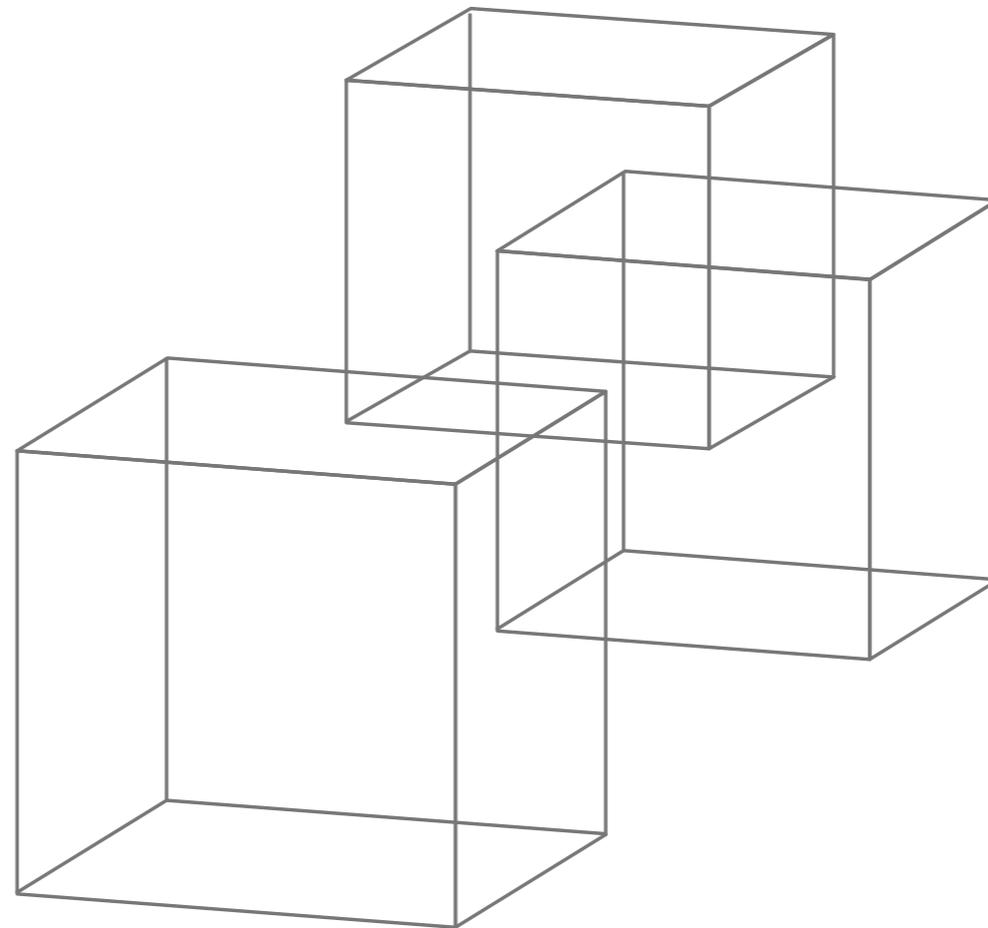
Objetos em VRML com *culling ON*

M. Próspero

# Algoritmo de Z-Buffer

Algoritmo mais geral e mais adequado a HSR

Situação inicial de um exemplo de aplicação:

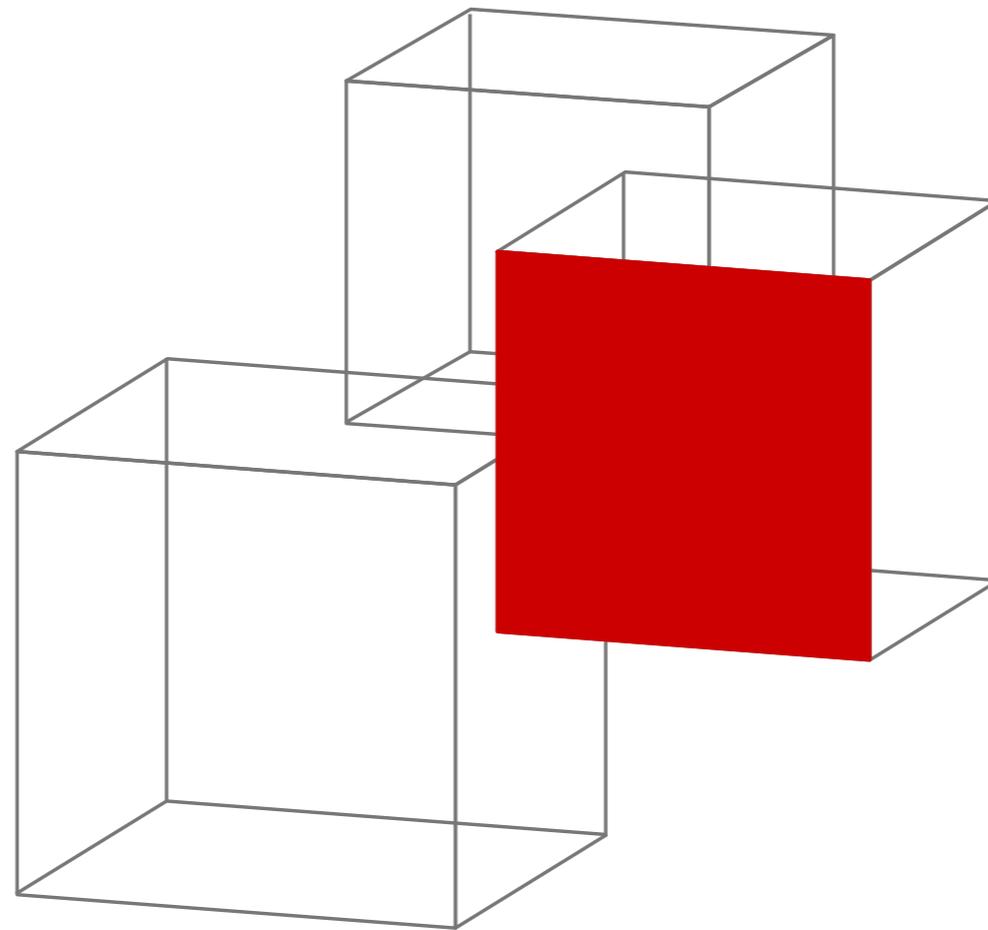


*M. Próspero*

# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

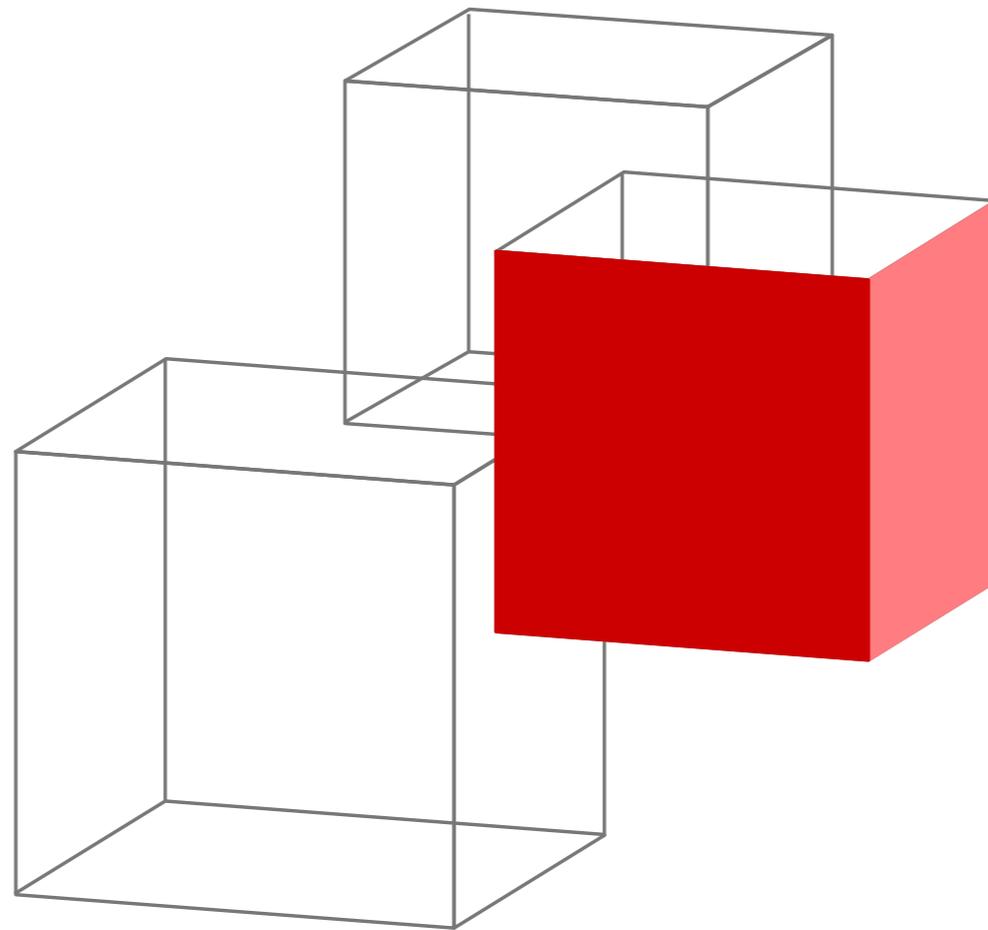
Embora os resultados intermédios possam ser diferentes, a ordem de processamento dos polígonos é irrelevante para o resultado final.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

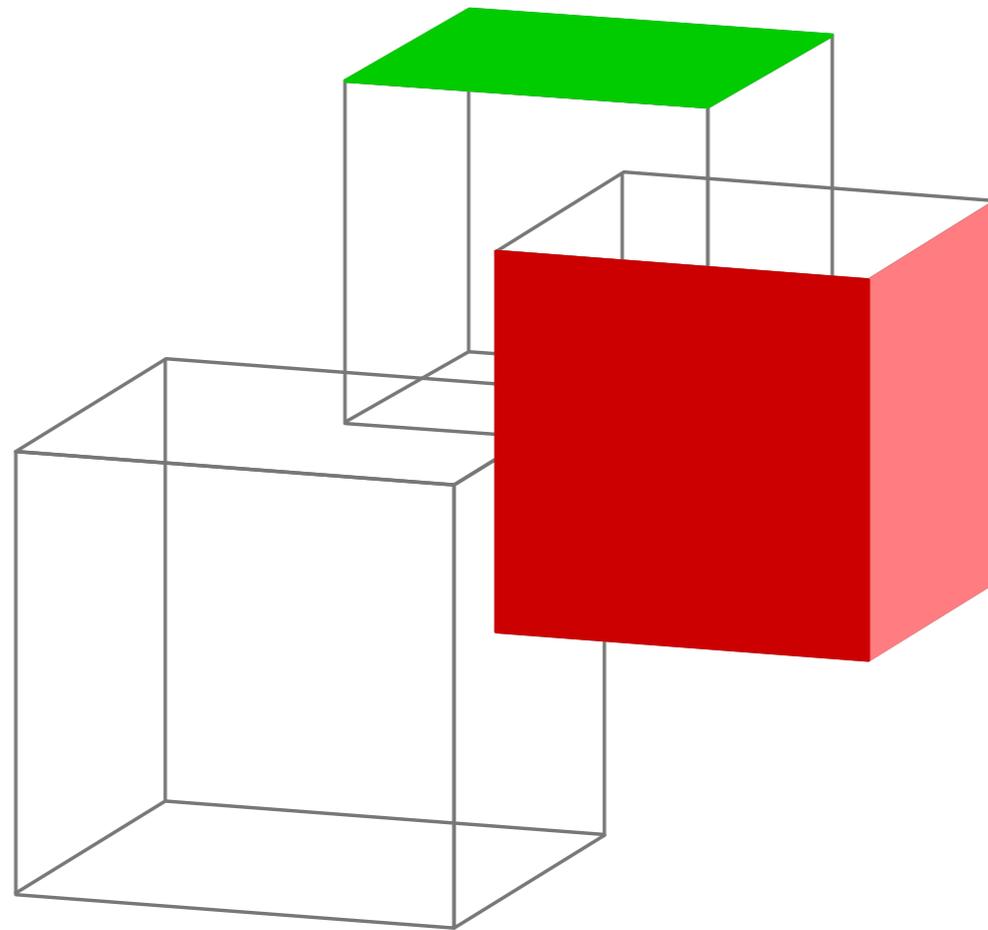
Embora os resultados intermédios possam ser diferentes, a ordem de processamento dos polígonos é irrelevante para o resultado final.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

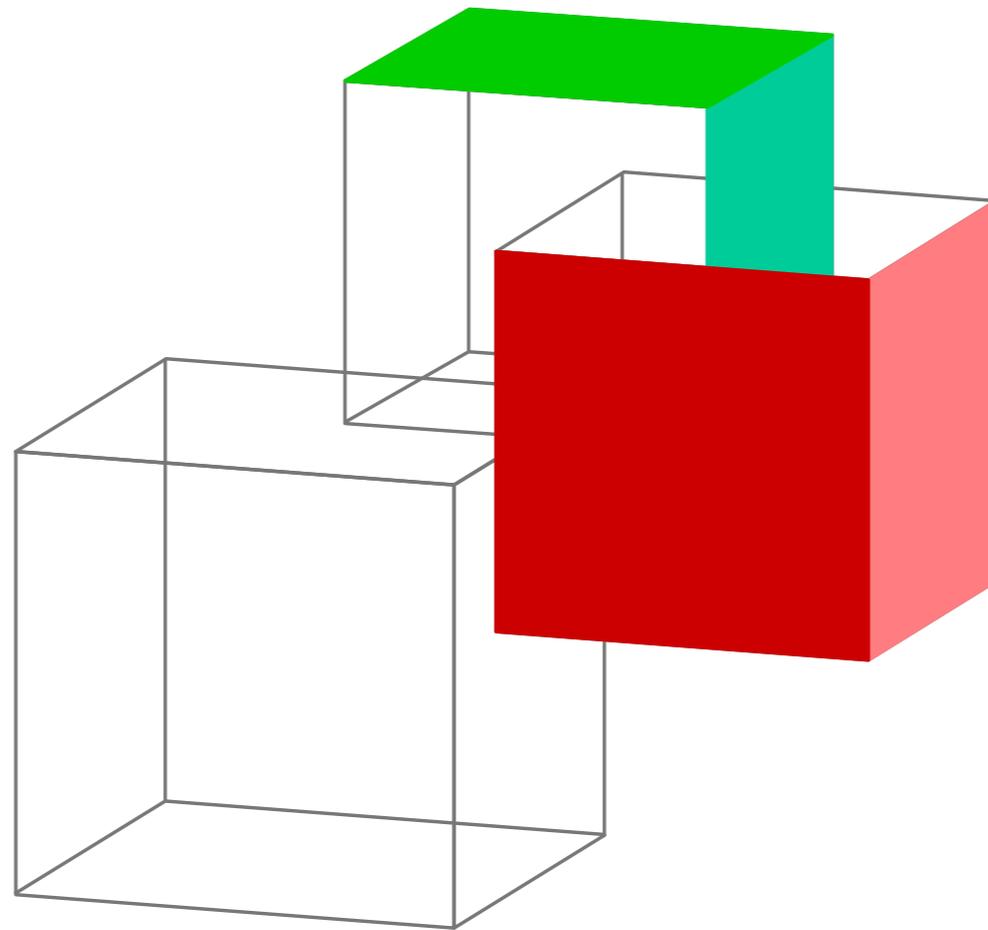
Embora os resultados intermédios possam ser diferentes, a ordem de processamento dos polígonos é irrelevante para o resultado final.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

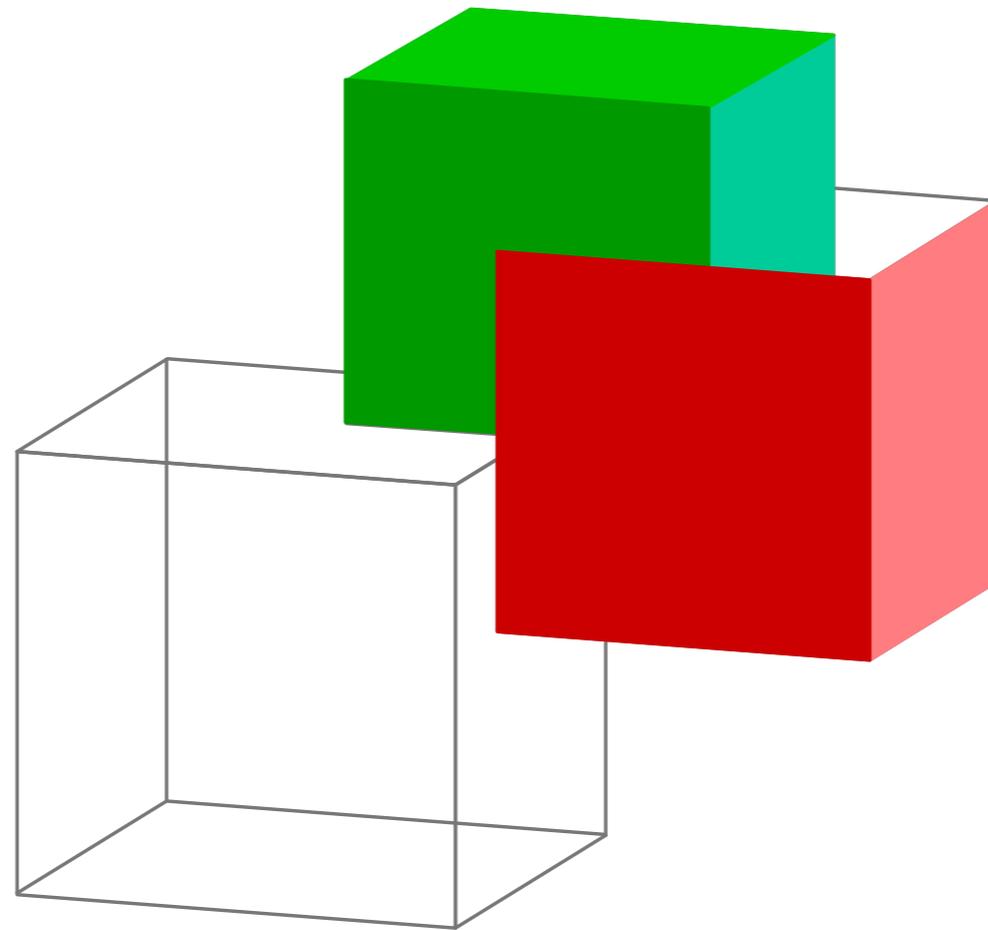
Cada pixel de um polígono só é escrito se não houver sobreposição com nenhum outro ou, caso contrário, se a sua distância ao observador for menor que a distância associada ao pixel, já escrito, pertencente a polígono tratado em fase anterior.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

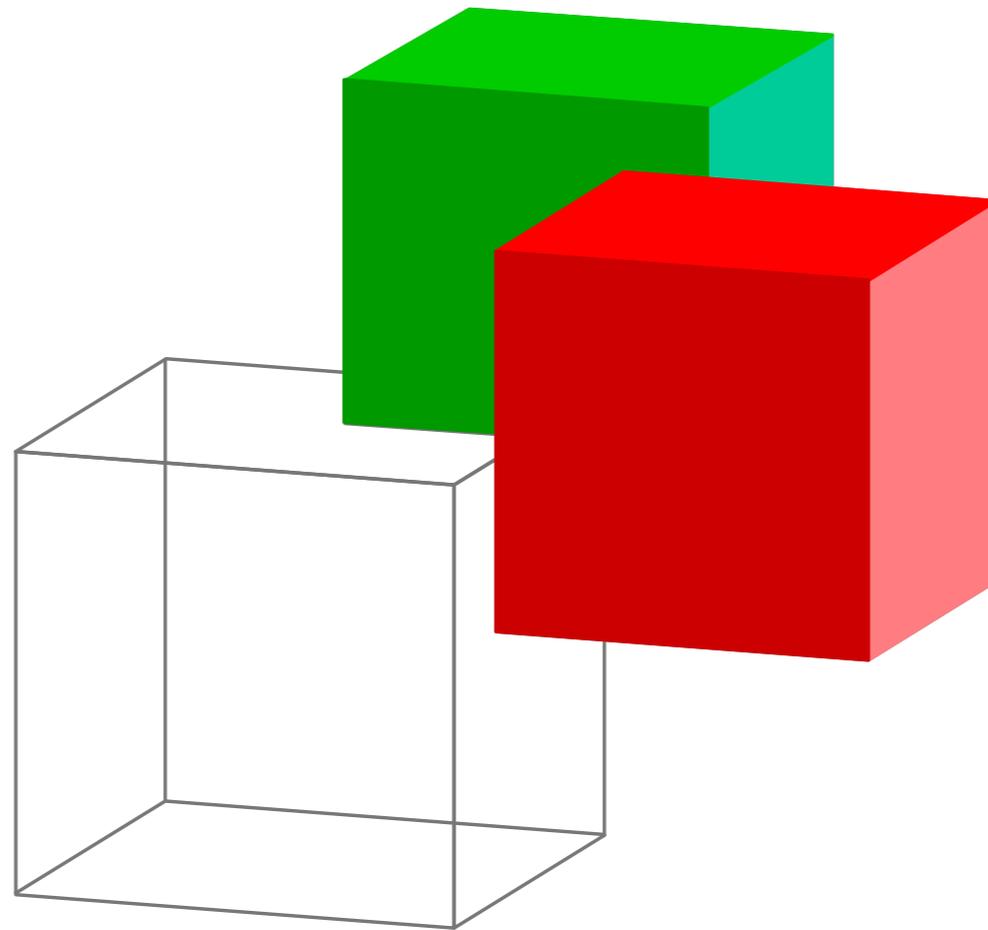
Cada pixel de um polígono só é escrito se não houver sobreposição com nenhum outro ou, caso contrário, se a sua distância ao observador for menor que a distância associada ao pixel, já escrito, pertencente a polígono tratado em fase anterior.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

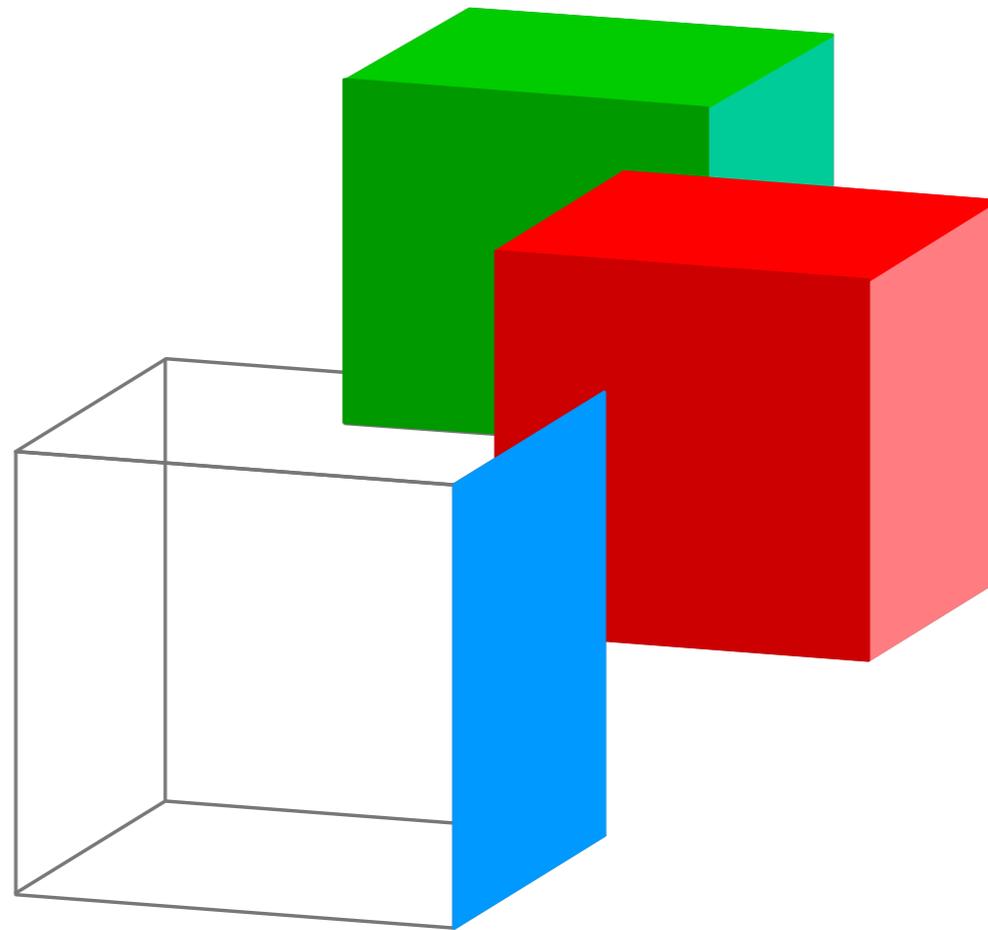
Cada pixel de um polígono só é escrito se não houver sobreposição com nenhum outro ou, caso contrário, se a sua distância ao observador for menor que a distância associada ao pixel, já escrito, pertencente a polígono tratado em fase anterior.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

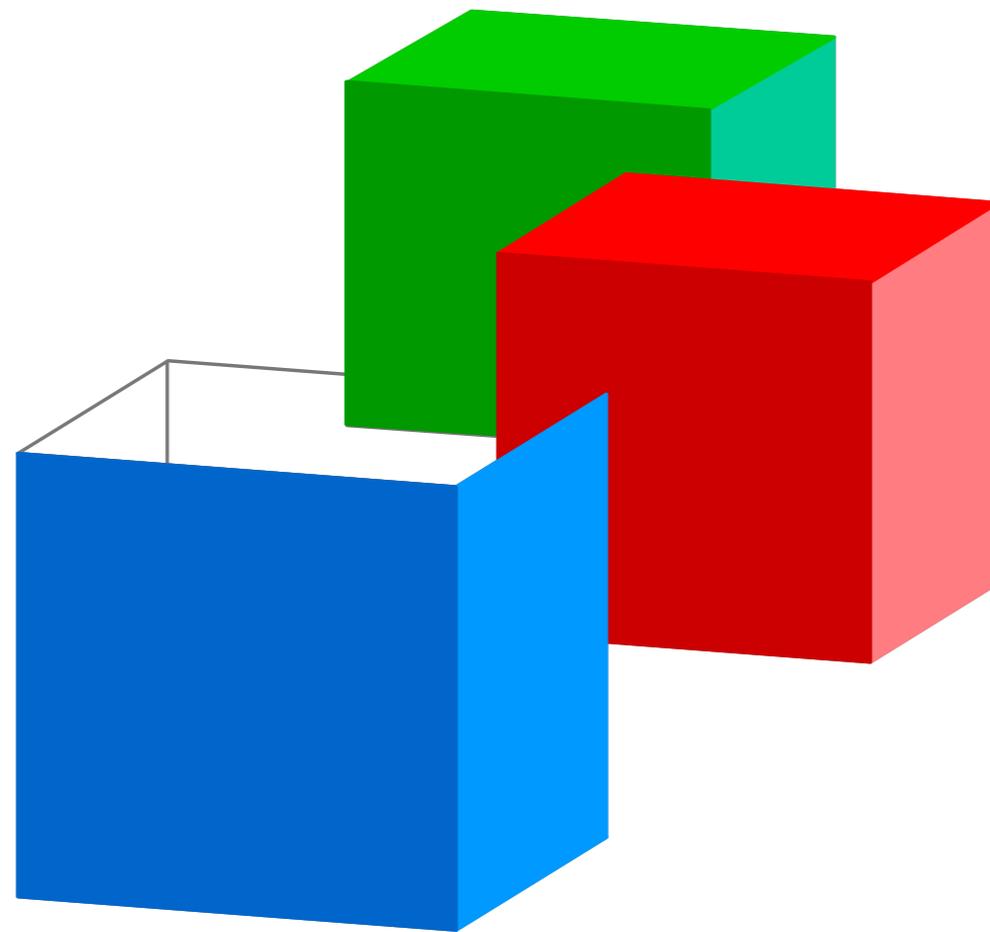
Cada pixel de um polígono só é escrito se não houver sobreposição com nenhum outro ou, caso contrário, se a sua distância ao observador for menor que a distância associada ao pixel, já escrito, pertencente a polígono tratado em fase anterior.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

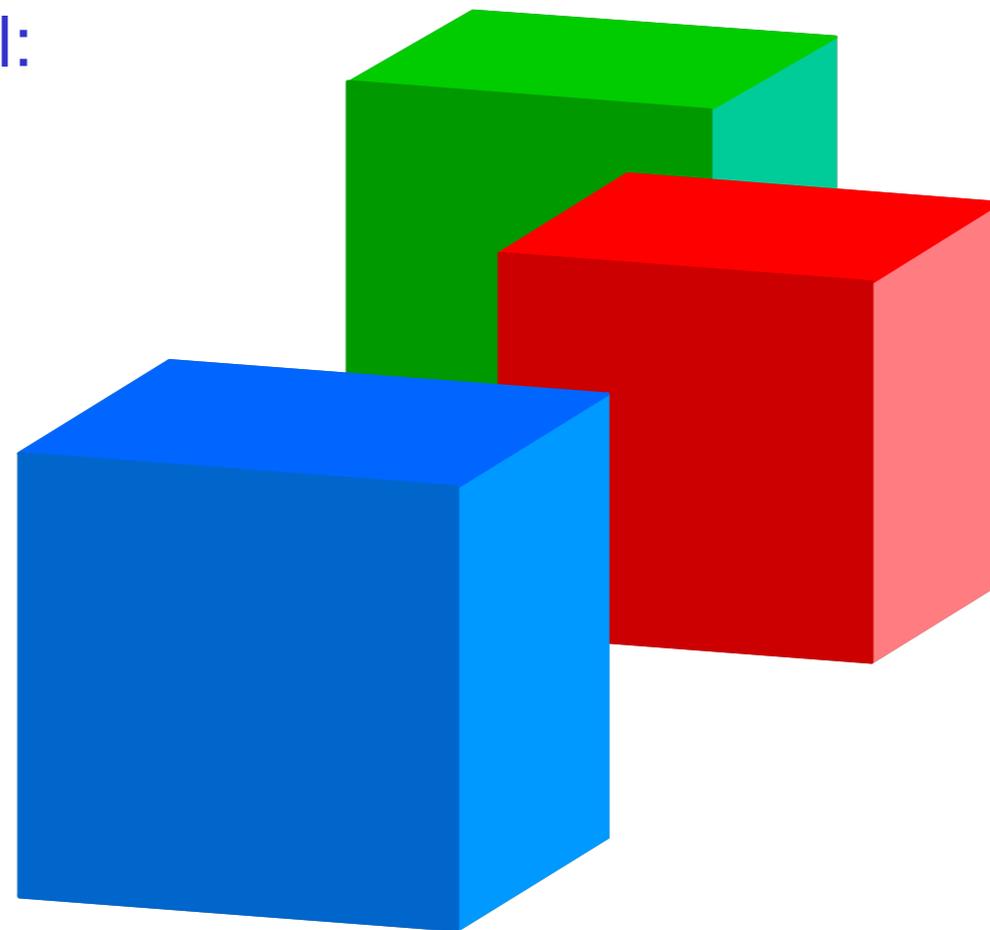
Cada pixel de um polígono só é escrito se não houver sobreposição com nenhum outro ou, caso contrário, se a sua distância ao observador for menor que a distância associada ao pixel, já escrito, pertencente a polígono tratado em fase anterior.



# Algoritmo de Z-Buffer

## Aplicação do algoritmo de Z-Buffer

Resultado final:



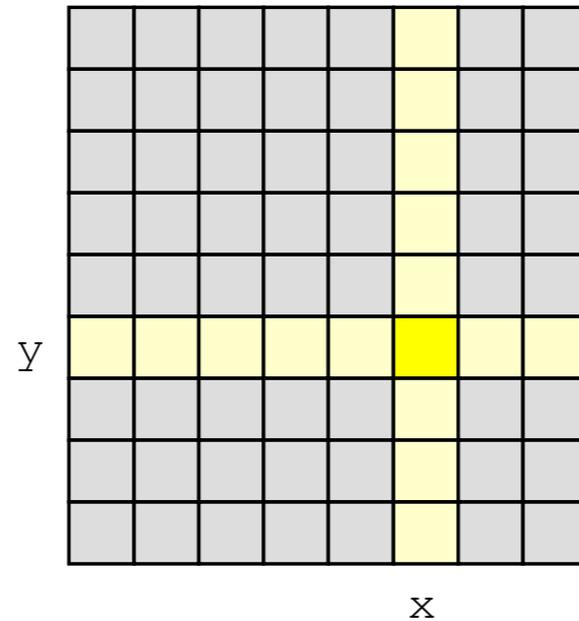
*M. Próspero*

# Algoritmo de Z-Buffer

## Estruturas de Dados

Refresh Buffer  
(RB)

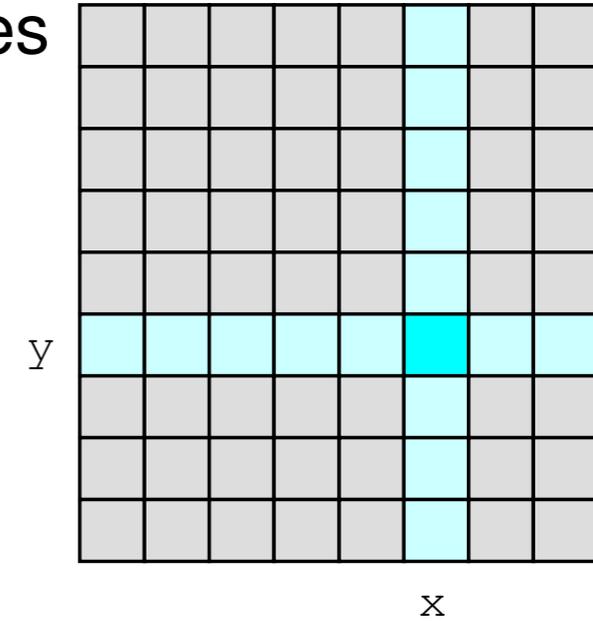
$n \times m$  valores  
de cor RGB



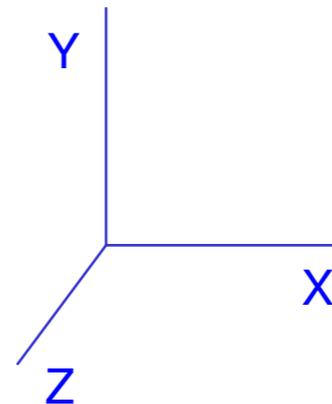
Inicialização com a cor de fundo.

Z-Buffer  
(ZB)

$n \times m$  valores  
em z



Inicialização com o menor valor possível.



M. Próspero

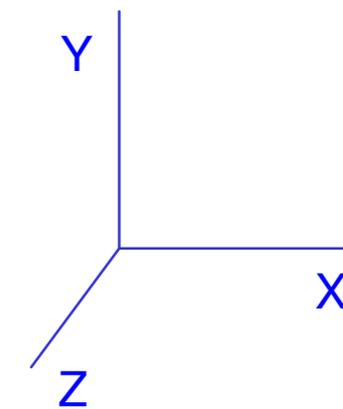
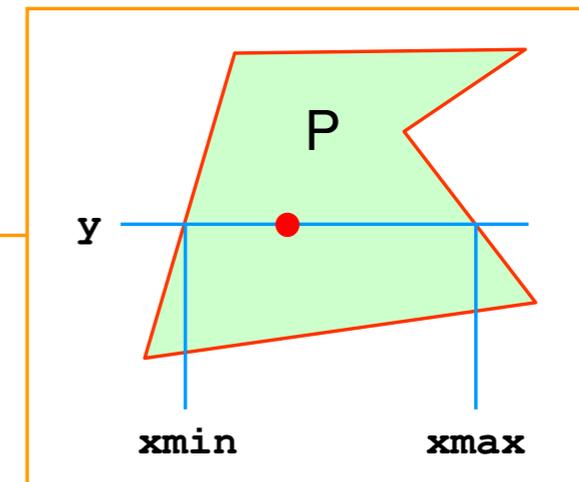
# Algoritmo de Z-Buffer

DADOS:

Polígonos planos, em projeção ortogonal, e conhecidos também os valores em z (profundidade) dos vértices.

Algoritmo de varrimento, para cada polígono P:

```
for ( y = ymin; y <= ymax; y++ ) {  
    Calcular xmin e xmax ; // v. algoritmo de FILL AREA  
    for ( x = xmin; x <= xmax; x++ ) {  
        z = f(P,x,y); // cálculo via equação do plano de P  
        if ( z > ZB[x,y] ) { // o pixel (x,y) passa o teste  
            ZB[x,y] = z ;  
            RB[x,y] = RGB(P,x,y) ;  
        }  
    }  
}
```



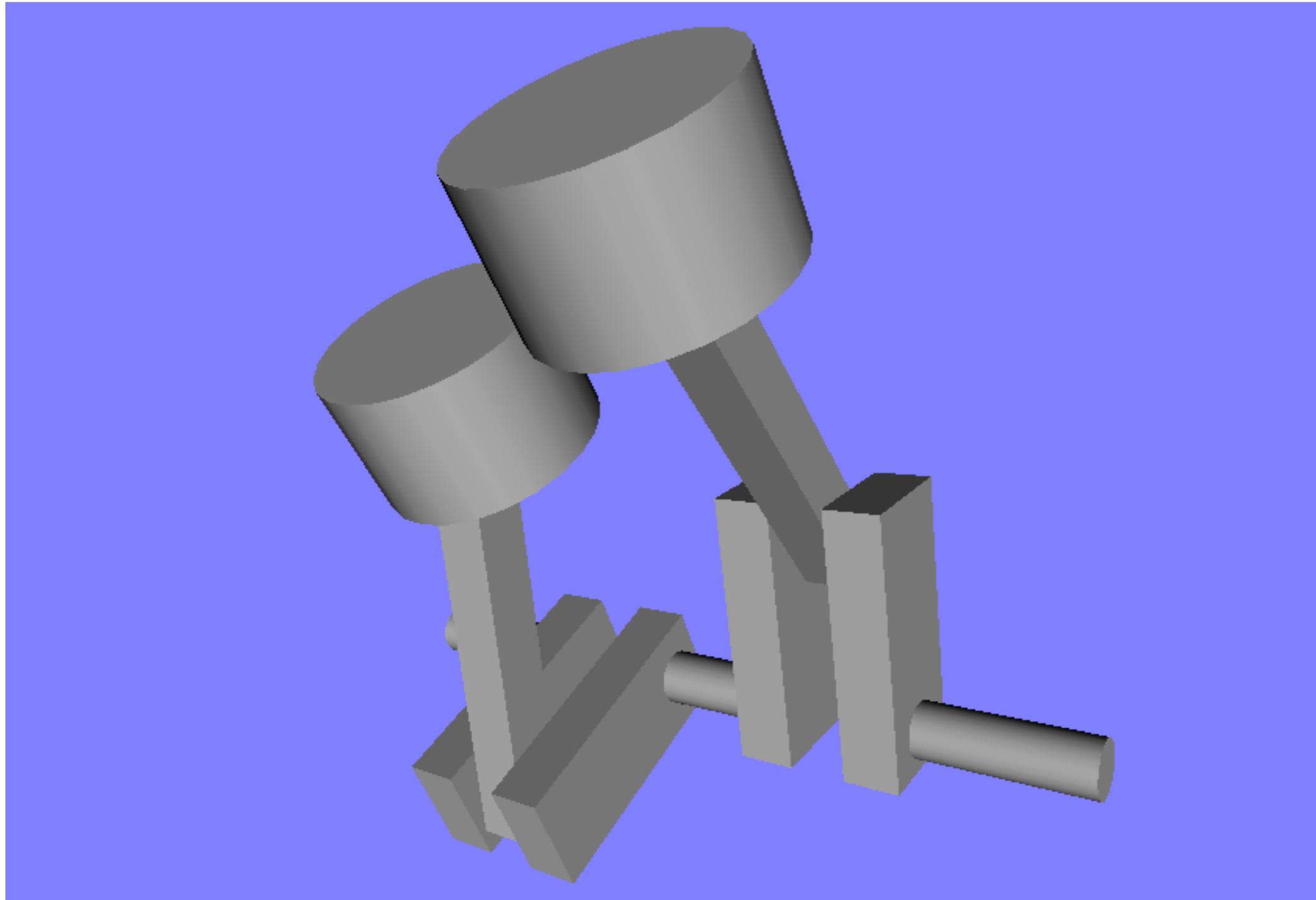
MUITO IMPORTANTE:

- Como Z-buffer se aplica depois da projeção, então esta terá de preservar a coordenada **z** (substituindo-se 0 por 1 na correspondente matriz da projeção que foi anteriormente deduzida)!
- Na API de OpenGL, em vez de **z** usa-se a distância **dz** ao ponto de vista, pelo que o teste será **dz < ZB[x,y]**.
- Prova-se que qualquer projeção se pode transformar em ortogonal.

*default: glDepthFunc(GL\_LESS)*

M. Próspero

# Exemplo



Aplicação de Z-buffer, associado a um modelo de iluminação

*M. Próspero*