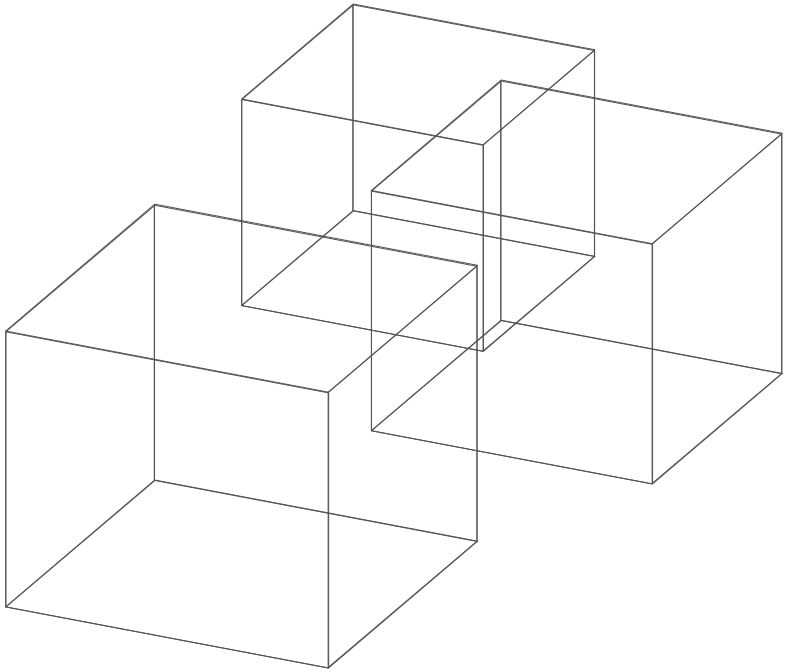
Remoção de Linhas e Superfícies Ocultas

Objectivos

- Compreender a necessidade de algoritmos de remoção de linhas e/ou superfícies ocultas
- Método do produto interno e suas limitações
- Algortimo de Z-buffer

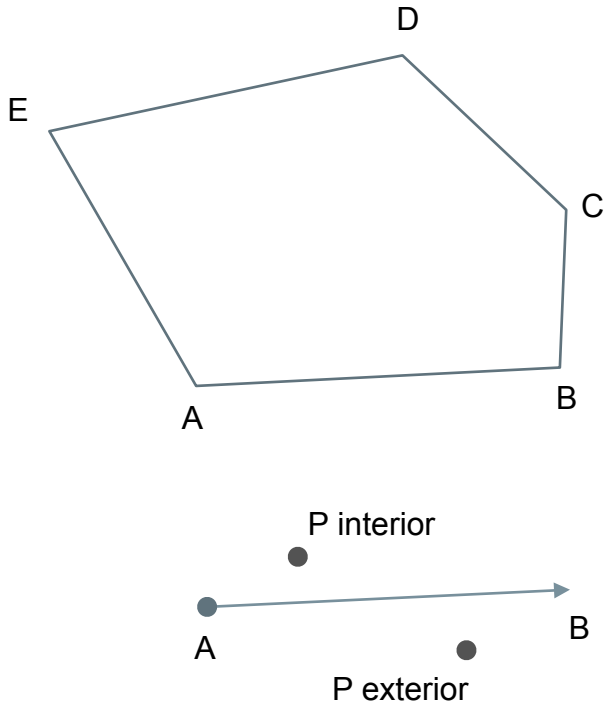
Necessidade de HLHSR*



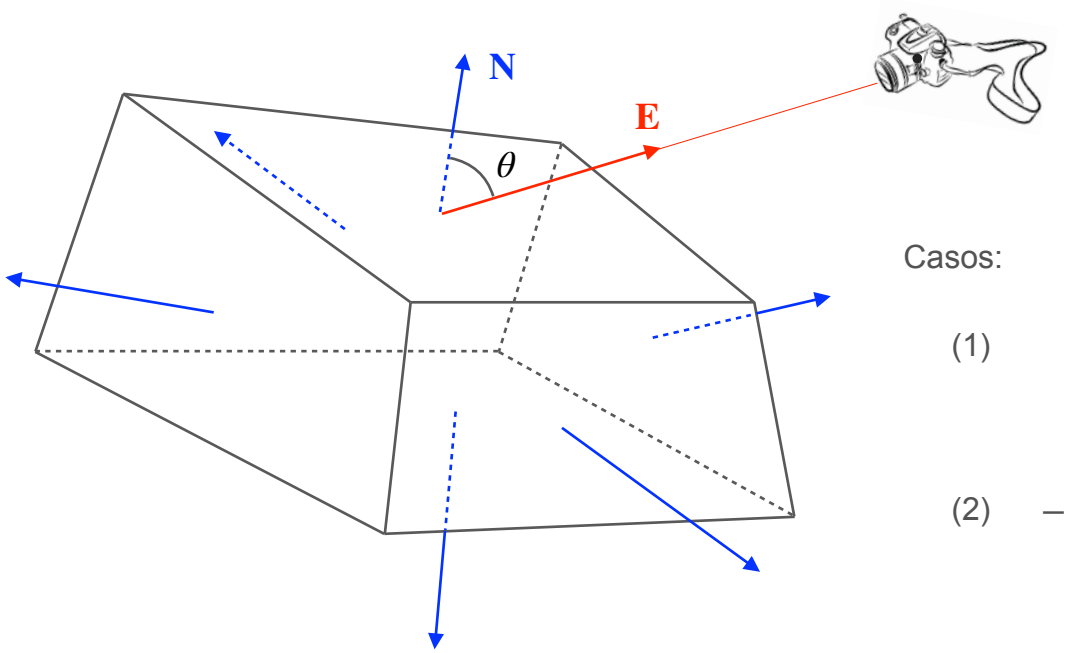
* Hidden Line Hidden Surface Removal

Considerações Preliminares

- Um polígono (convexo) tem orientação positiva se todos os seus pontos ficam sistematicamente à esquerda de todas as arestas do mesmo.
- AB x AP (produto externo) poderá ser usado para determinar se P se encontra à esquerda ou à direita de AB.
- Se considerarmos que o polígono está no plano ij, sendo k a direção perpendicular, o sinal da componente de AB x AP segundo k será positivo caso o ponto se encontre à esquerda de AB e negativo caso se encontre à sua direita



Método do Produto Interno*



$$\cos\theta = \frac{\mathbf{N}\cdot\mathbf{E}}{|\mathbf{N}\cdot\mathbf{E}|}$$

Casos:

(1) $0 < \cos\theta \leq 1 \Leftrightarrow 0° \leq \theta < 90°$

↳ A superfície é visível

(2) $-1 \leq \cos\theta \leq 0 \Leftrightarrow 90° \leq \theta \leq 180°$

↳ A superfície não é visível

Apenas se garante um resultado correto numa cena apenas com poliedros convexos e sem sobreposição

* Backface culling

Método do Produto Interno

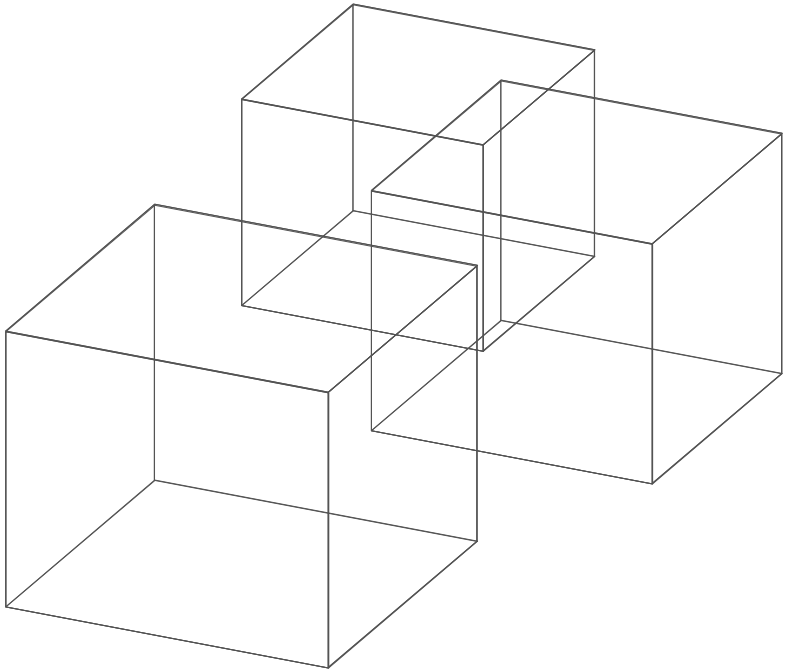
- Em WebGL aplica-se o método usando:

  `gl.enable(gl.CULL_FACE)`

- Pode-se definir o tipo de faces a ocultar:

  `gl.cullFace(face)`

  face = gl.FRONT | gl.BACK



Método do Produto Interno

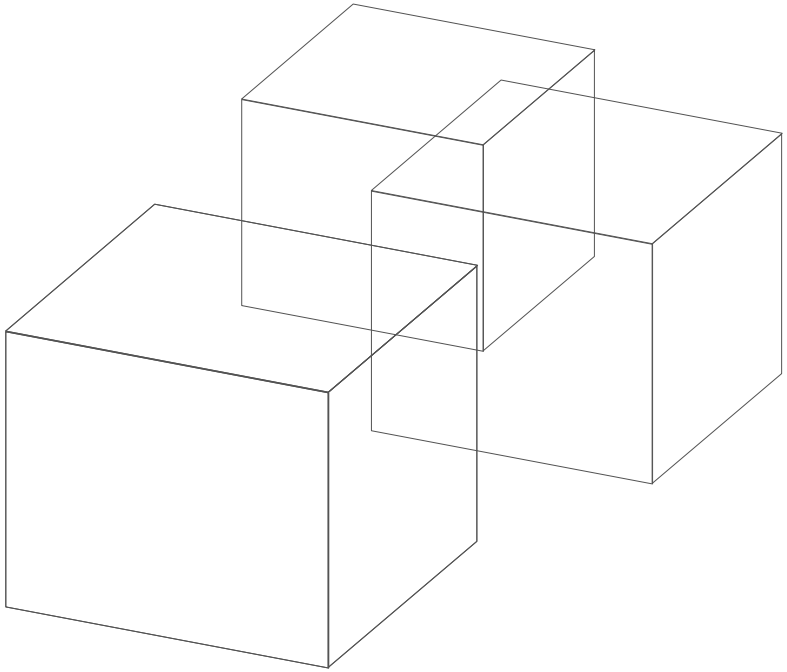
- Em WebGL aplica-se o método usando:

  `gl.enable(gl.CULL_FACE)`

- Pode-se definir o tipo de faces a ocultar:

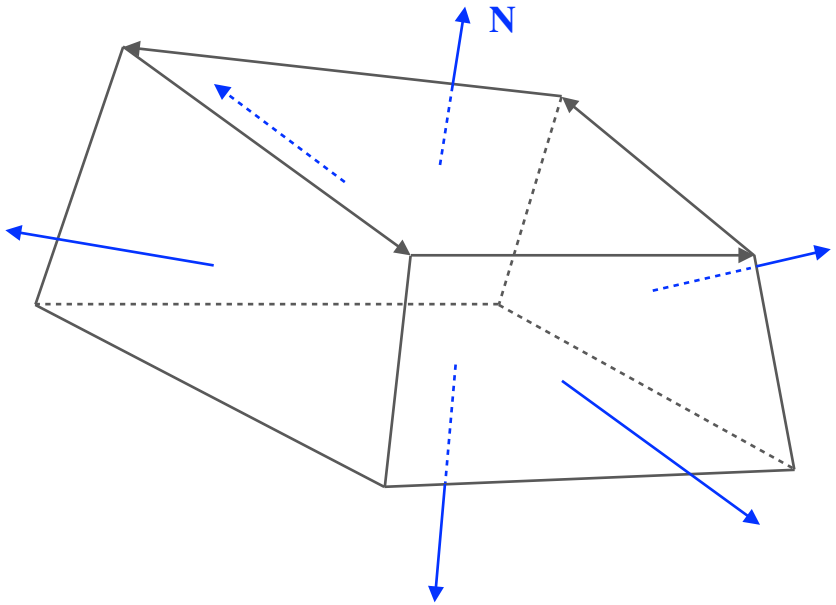
  `gl.cullFace(face)`

  face = gl.FRONT | gl.BACK



Exemplo com remoção de faces voltadas para trás: gl.cullFace(gl.BACK)

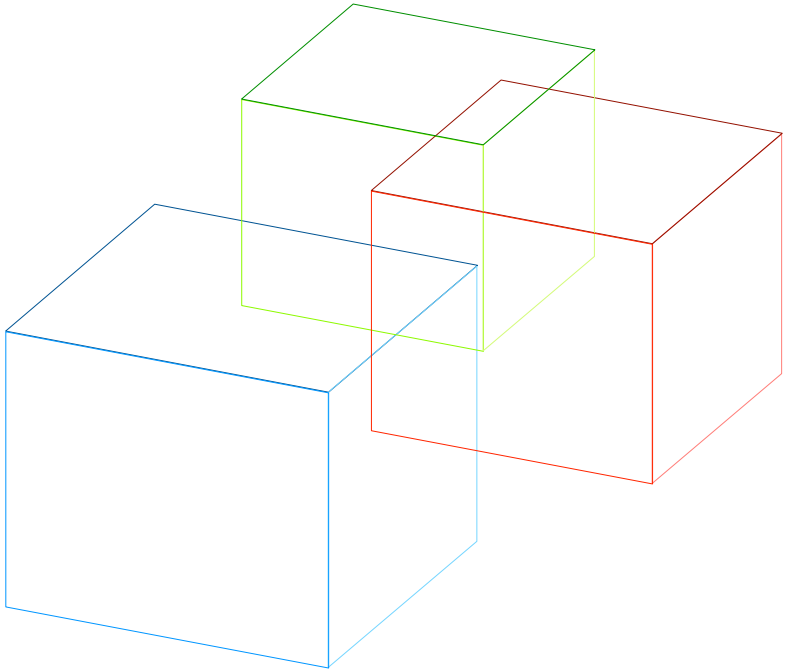
Método do Produto Interno



- Se todos os polígonos tiverem uma orientação positiva, quando observados do exterior do poliedro, os vetores normais podem ser facilmente calculados
- Tomando 3 vértices consecutivos (2 arestas), efetua-se o produto externo
- No caso da projeção ortogonal, bastará investigar a componente da normal $\mathbf{N}$ segundo z
- Se $\mathbf{N}_z > 0$, o polígono será visível

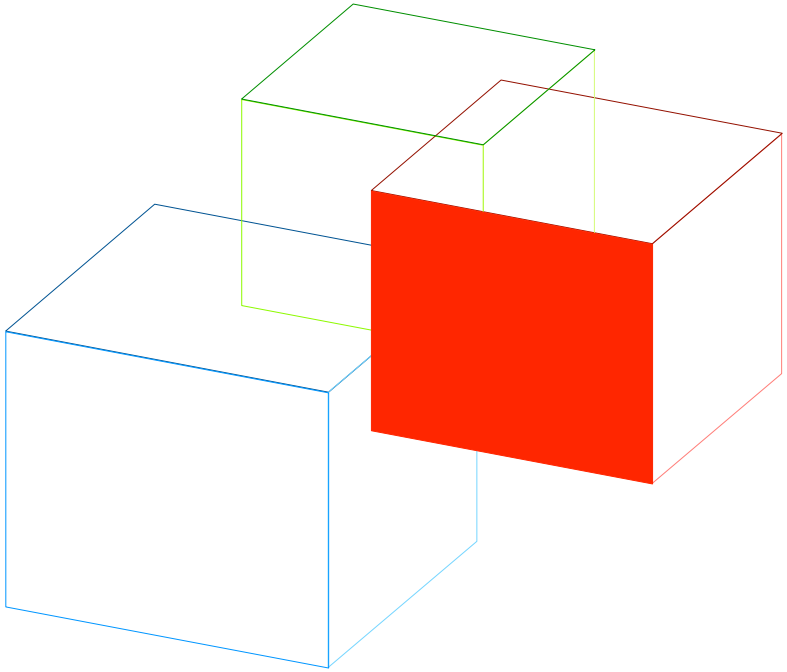
Algoritmo de Z-Buffer

- As linhas apenas estão desenhadas para se perceber a geometria a desenhar
- O algoritmo vai, neste caso, ser aplicado a superfícies, imaginando que já se aplicou o método do produto interno (backface culling)
- A situação inicial corresponde a um canvas sem qualquer desenho…



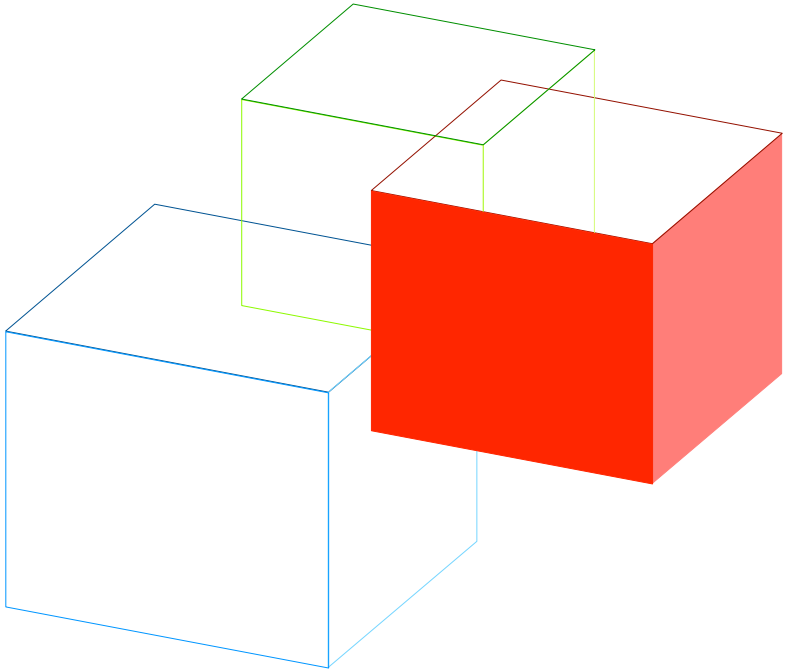
Algoritmo de Z-Buffer

- A primeira face a ser pintada não coloca qualquer dúvida sobre o resultado



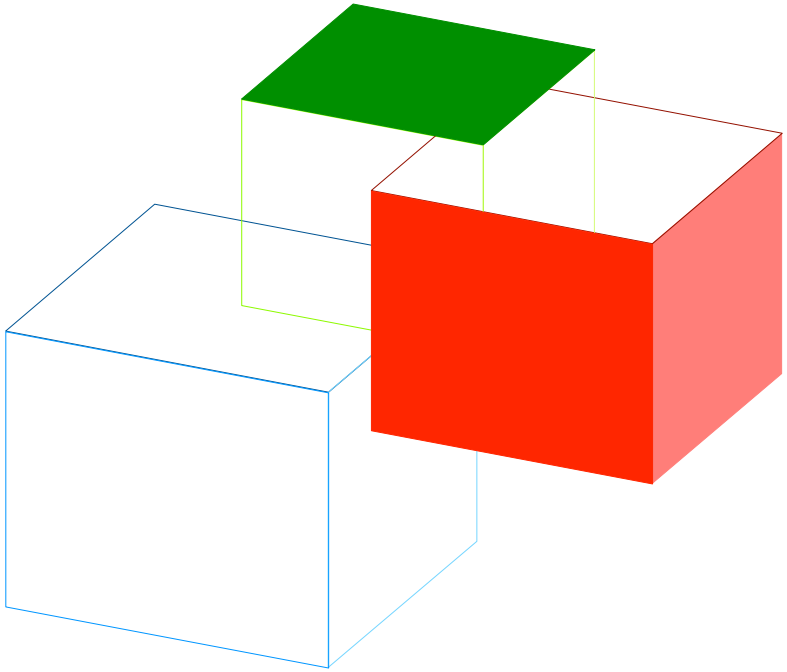
Algoritmo de Z-Buffer

- A segunda face também não coloca qualquer dúvida visto não se sobrepor a qualquer outra que tivesse já sido desenhada



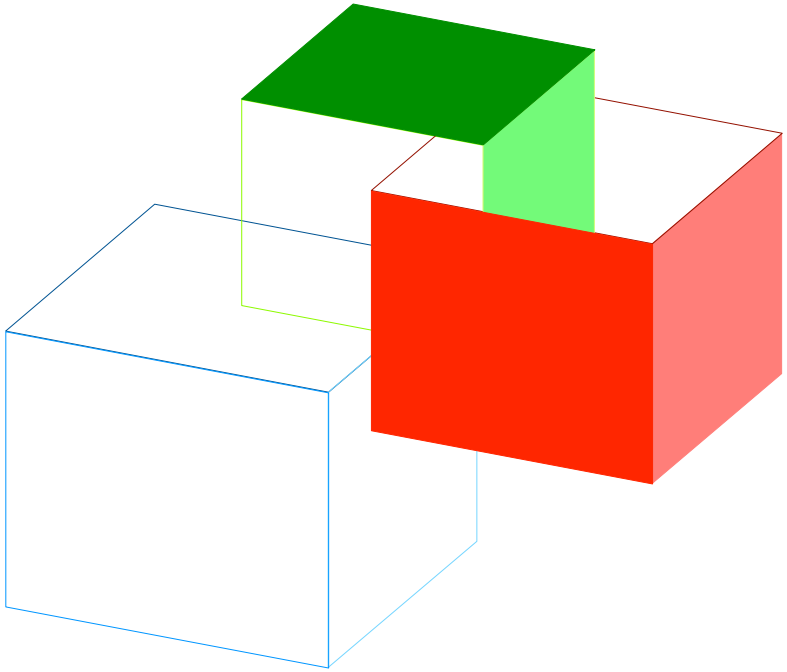
Algoritmo de Z-Buffer

- O mesmo se pode dizer da terceira face a ser desenhada



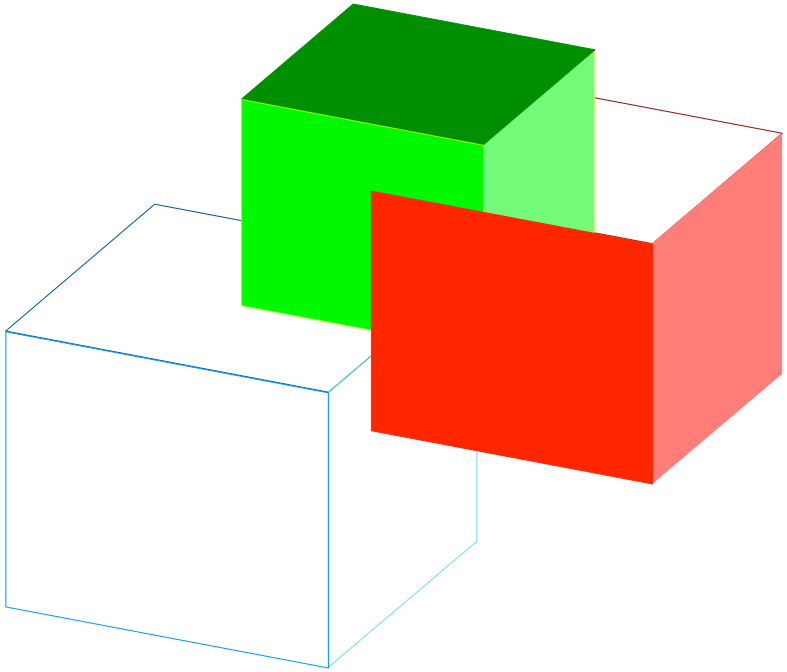
Algoritmo de Z-Buffer

- Ao desenhar-se a quarta face é necessário ter algum algoritmo que determine, para cada pixel, se este deverá ou não ser escrito
- A aplicação do algoritmo consiste em testar a profundidade associada a cada pixel durante o varrimento da primitiva
- Pixels que iriam ser desenhados mas que se encontram a maior distância da câmara não chegam a ser pintados



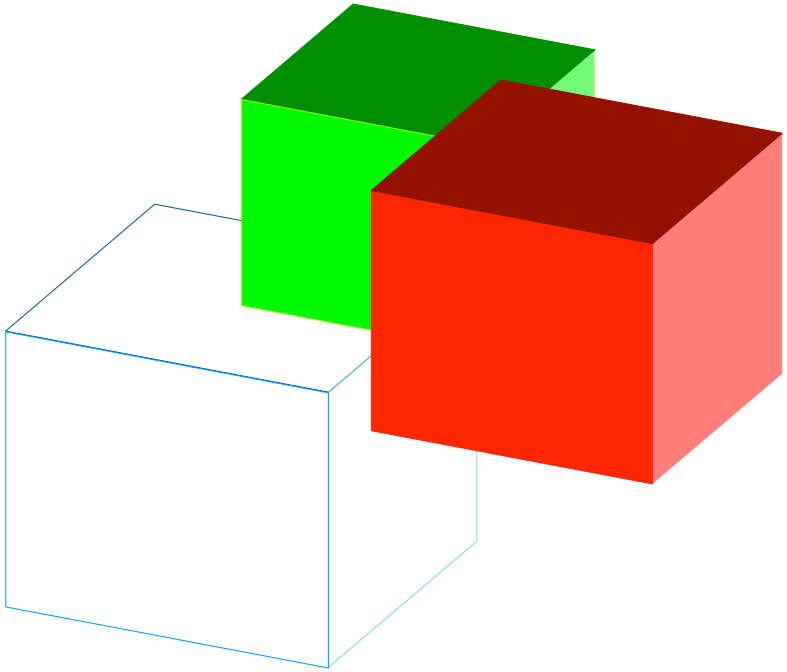
Algoritmo de Z-Buffer

- Ao desenhar-se a quinta face, procede-se da mesma maneira que no passo anterior



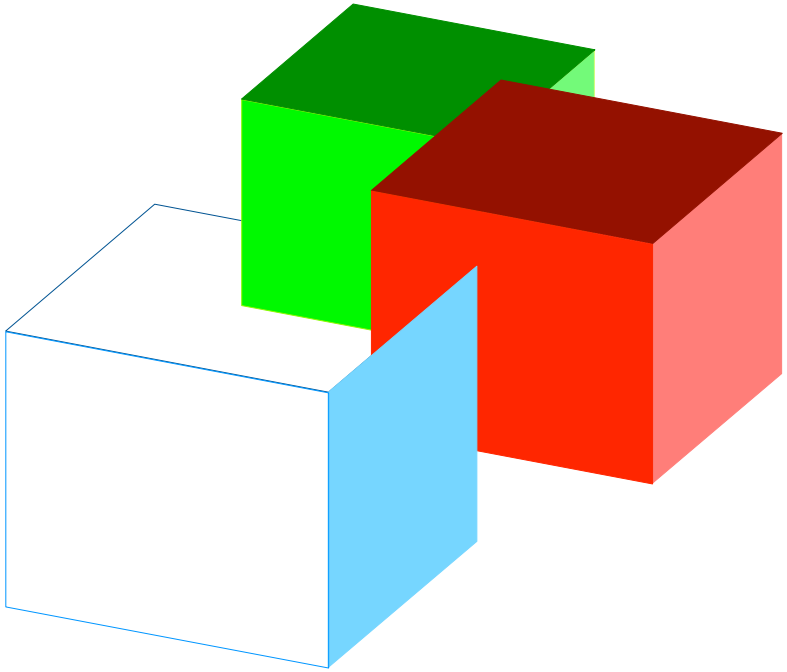
Algoritmo de Z-Buffer

- Neste caso, todos os pixels da face a desenhar estão mais perto da câmara do que os entretanto já desenhados, pelo que se deverão sobrepor a qualquer conteúdo desenhado até ao momento



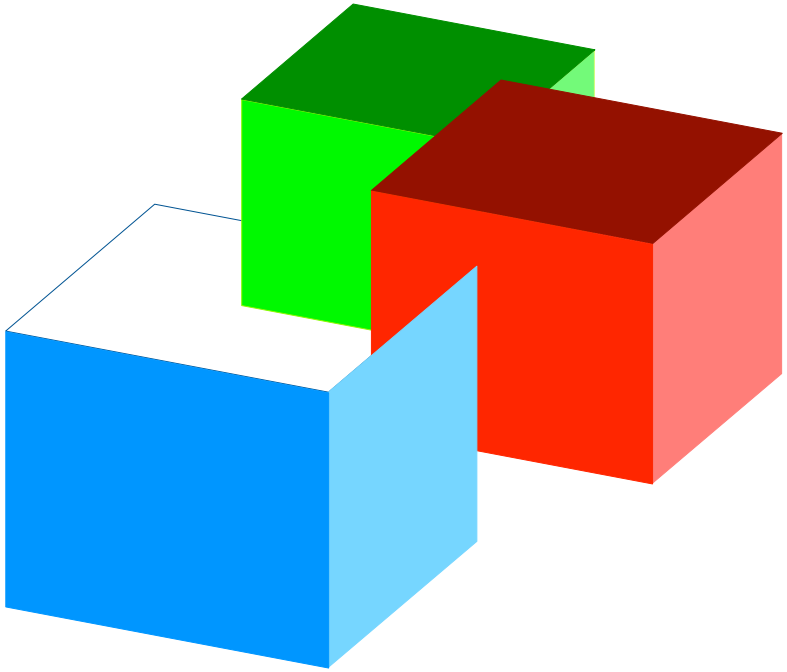
Algoritmo de Z-Buffer

- Neste passo sucedeu exatamente o mesmo que no passo anterior



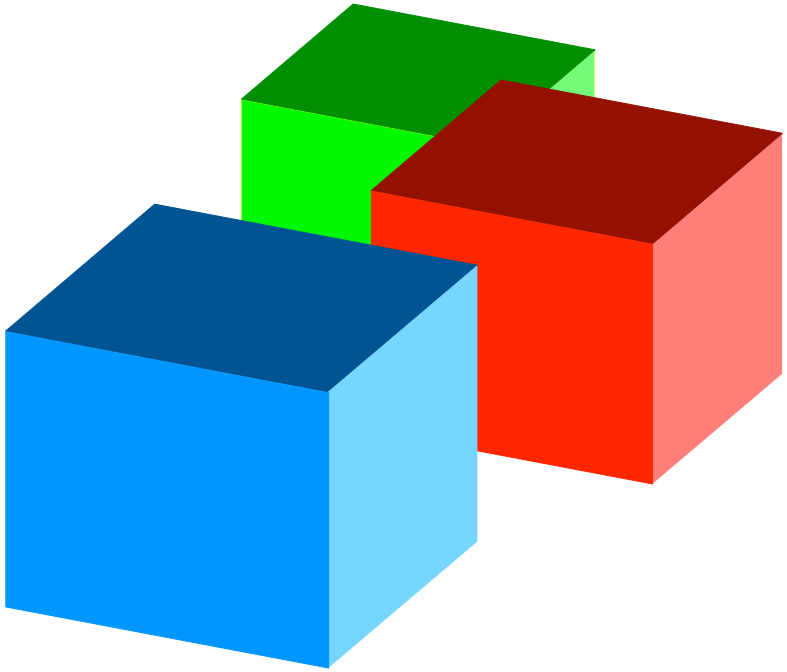
Algoritmo de Z-Buffer

- Idem…

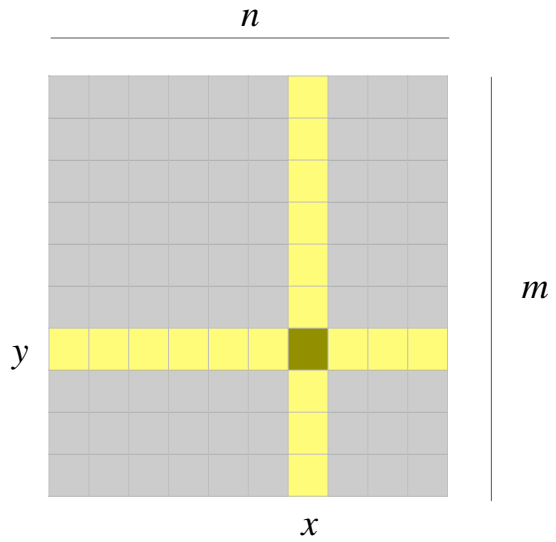


Algoritmo de Z-Buffer

- Com o desenho da última face termina o processo.
- Embora os resultados intermédios possam variar com a ordem das operações de desenho (ordem das primitivas), o resultado final será sempre o mesmo.

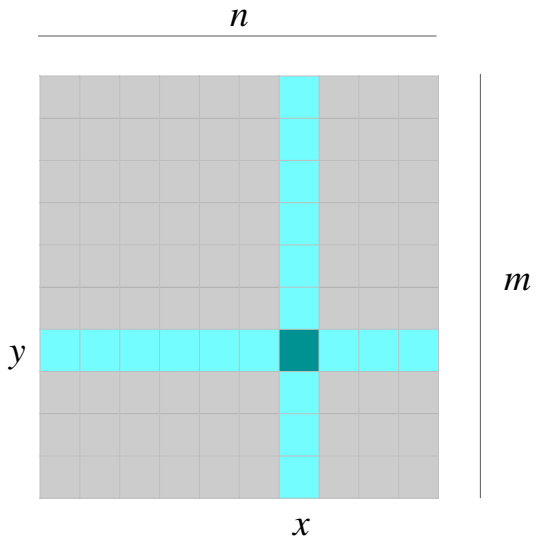


Algoritmo de Z-Buffer



Framebuffer (FB)

inicializado com a cor do fundo



Z-Buffer (ZB)

Inicializado com o maior valor possível

Algoritmo de Z-Buffer

- Seja z o valor da profundidade associado a um pixel de coordenadas (x, y) resultante da conversão por varrimento duma determinada primitiva plana
- O algoritmo de Z-buffer consiste na seguinte sequência de operações durante a fase de (potencial) escrita do pixel no ecrã:

```
if( z < ZB[x,y] ) {
    ZB[x,y] = z;
    RB[x,y] = c;
}
```

- c representa a cor do pixel (atribuída no fragment shader)