REMOÇÃO DE PARTES OCULTAS

Computação Gráfica e Interfaces

FCT UNIVERSIDADE NOVA DE LISBOA

Sumário

Linhas e superfícies ocultas

- Considerações preliminares

Algoritmos

- *Culling* de faces
- Z-buffer

Problema

Motivação

- Superfícies podem estar ocultadas
- Superfícies podem sobrepor-se no plano de imagem
- Superfícies podem intersectar-se

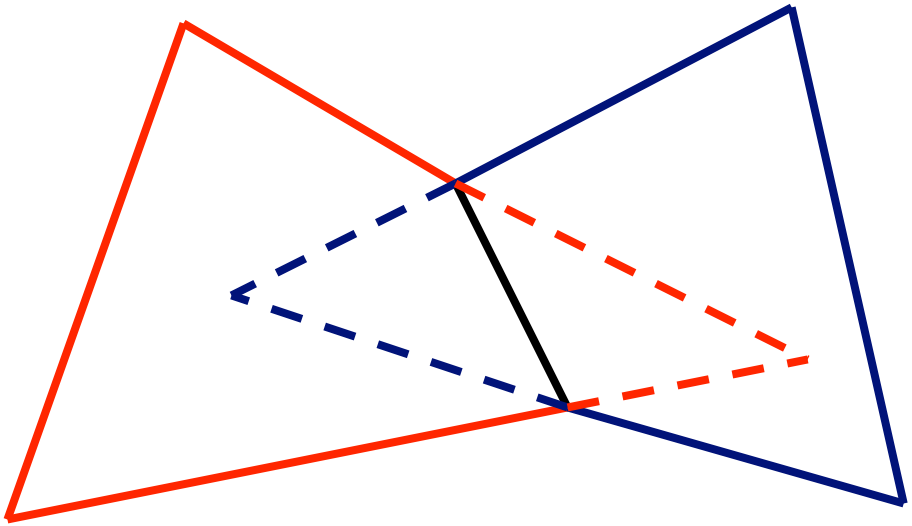
Operação de remoção

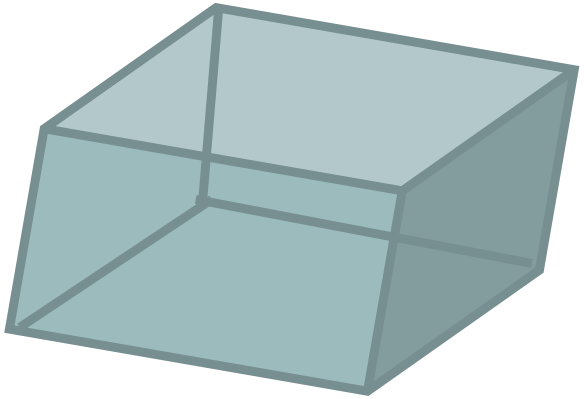
- Algures na *pipeline* gráfica, é necessário determinar quais os objectos (ou parte destes) visíveis e quais os que estão ocultos

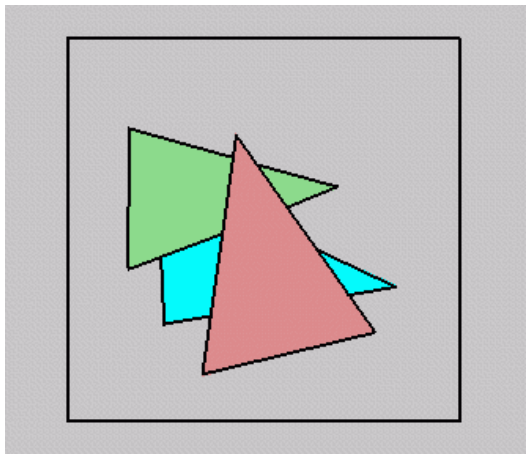
Vários algoritmos

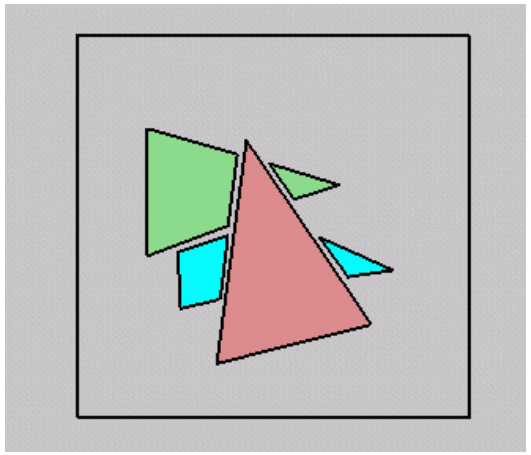
- Detecção da face posterior
- Lista ordenada em profundidade (algoritmo do pintor)
- Ray casting
- Varrimento, similar ao *fill-area*
- Z-buffer
- Subdivisão de área
- etc.

Uma face (de um objecto) oculta uma parte ou a totalidade de outra face (do mesmo objecto ou de outro objecto)





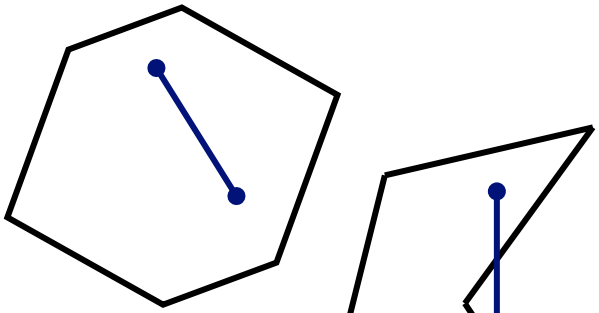




Alguns conceitos sobre polígonos (do Cap. Visualização 2D)

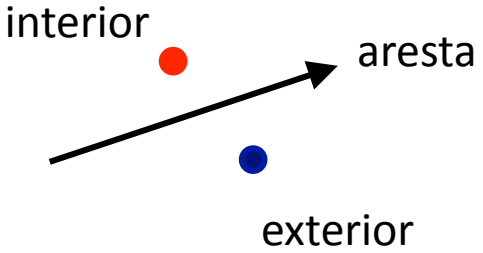
Convexidade

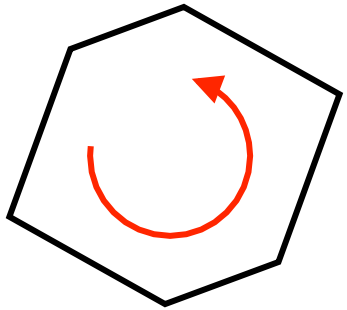
- Um polígono diz-se convexo se um segmento de recta unindo quaisquer dois pontos interiores estiver totalmente no interior



Ponto interior

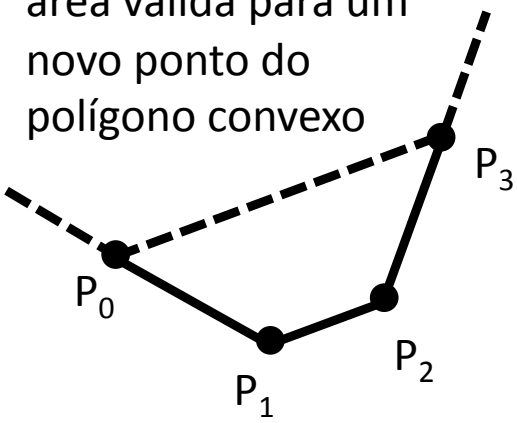
- Recorre-se ao sinal do produto externo entre vectores do plano - mesma origem, um com direcção da aresta e o outro dirigido ao ponto a testar

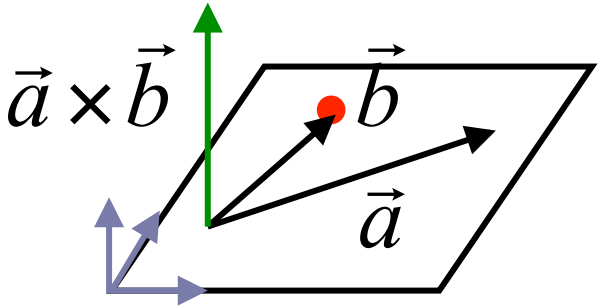




Orientação positiva das arestas

área válida para um novo ponto do polígono convexo





Metodologias clássicas

Espaço da imagem

- Para cada pixel na imagem, determina-se o objecto mais próximo do observador que é projectado no pixel e desenha-se esse pixel com a cor apropriada
- Características
  - Simples
  - Dependente da resolução
  - Possibilidade de determinação repetida de relações entre objectos
  - O esforço é proporcional ao número de objectos vezes o número de pixels

Espaço do objecto

- Para cada objecto, determinam-se as partes cuja vista não é obstruída por outras, desse ou de outro objecto, e desenham-se essas partes com a cor apropriada
- Características
  - Metodologia complicada e de cálculo intensivo
  - Independente da resolução
  - O mesmo pixel pode ser escrito várias vezes
  - É apropriado para cenas com poucos polígonos – o esforço é proporcional ao quadrado do número de objectos

Sumário

Linhas e superfícies ocultas

- Considerações preliminares

Algoritmos

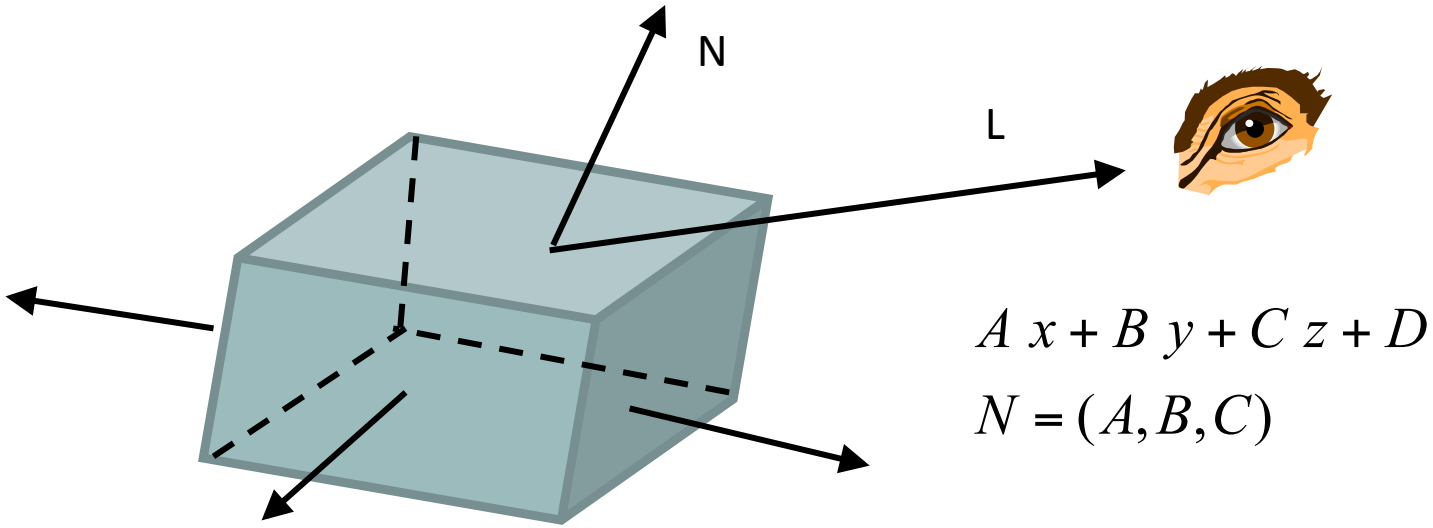
- *Culling* de faces
- Z-buffer

Método de *Culling* de faces para poliedros convexos

Pretende-se não desenhar as faces que não são visíveis segundo a direcção do observador

Metodologia aplicável logo nos estágios iniciais da *pipeline*, i.e. após a modelação dos objectos, o que lhe confere relevância em termos de eficiência global

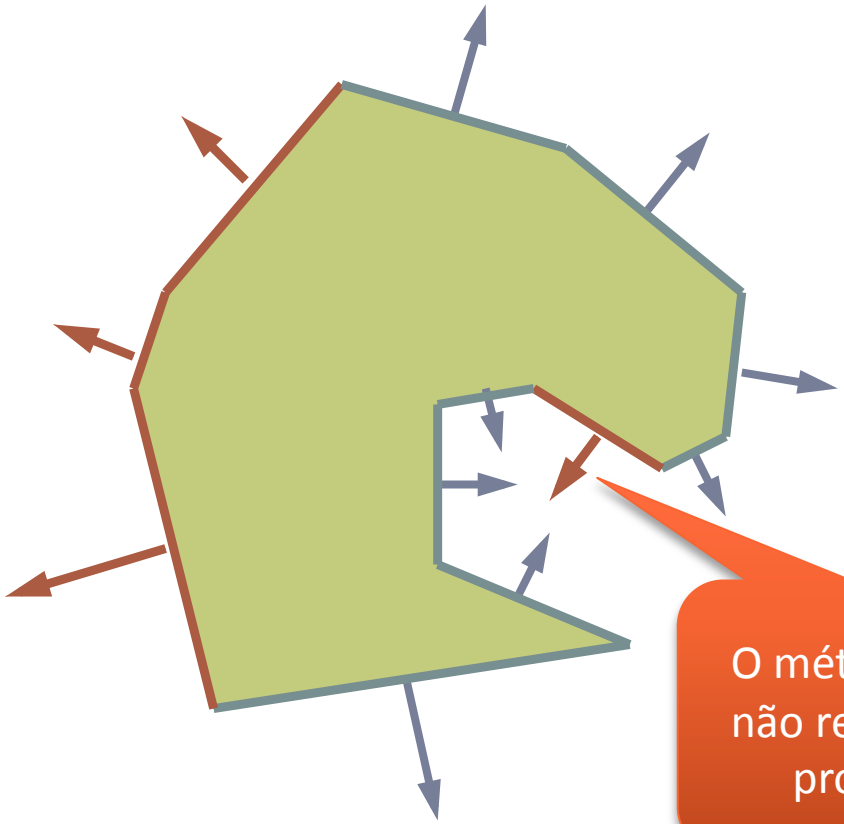
Uma superfície é visível quando o ângulo entre as direcções L e N situa-se no intervalo [0, 90 graus [ou seja, L . N > 0

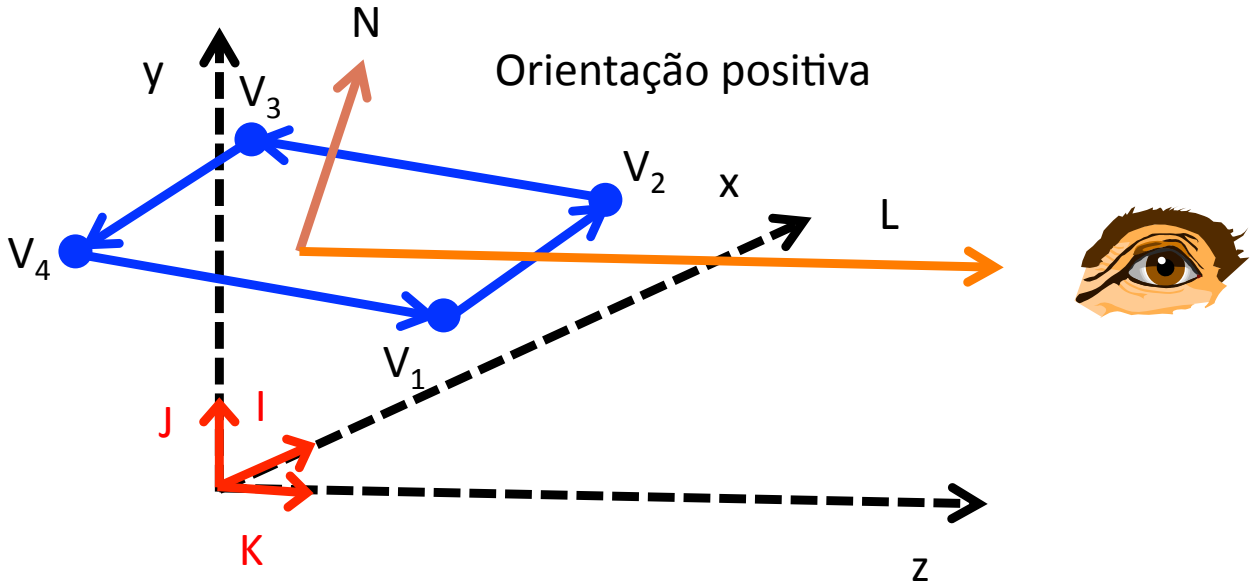


$$A\,x + B\,y + C\,z + D$$

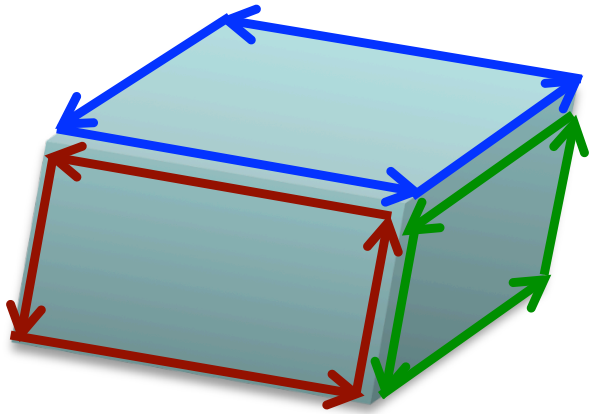
$$N = (A, B, C)$$

Repare-se que o método é aplicável apenas a poliedros convexos





$$N = (V_2 - V_1) \times (V_3 - V_2) = \begin{vmatrix} I & J & K \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_2 & y_3 - y_2 & z_3 - z_2 \end{vmatrix}$$



Teste da condição

- Considerando a projecção ortogonal, basta analisar o sinal de uma das componentes de N
- Com o observador colocado segundo o eixo OZ, direccionado para –Z e com o plano de projecção XY, o que é o caso do OpenGL, então L = (0,0,1) e assim o teste L . N > 0 resume-se apenas ao teste da componente z do vector normal N: se positivo, a face é visível

Em OpenGL

- gl.glEnable(GL.GL_CULL_FACE)
- gl.glCullFace(GL.GL_BACK ou GL.GL_FRONT), indicando qual o tipo de faces a ocultar. Por exemplo, com GL.GL_BACK, apenas ficam os polígonos frontais relativamente ao observador

Algoritmo de Z-Buffer (Ed Catmull, 1973)

Adequado para remoção de superfícies ocultas

Algoritmo no espaço da imagem, aplicável após a operação de projecção

A resolução do problema da visibilidade é feita ao nível de cada pixel de forma independente e com ordem de processamento polígono a polígono

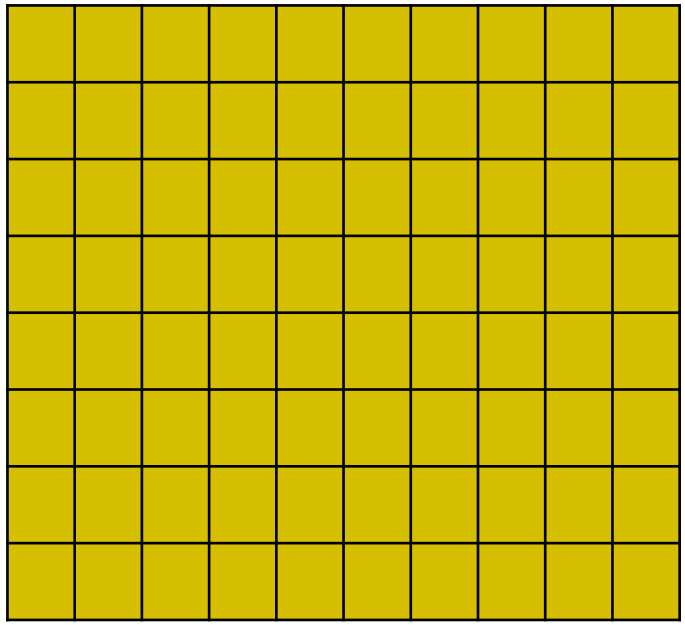
Embora os resultados intermédios possam ser diferentes, a ordem de processamento dos polígonos é irrelevante

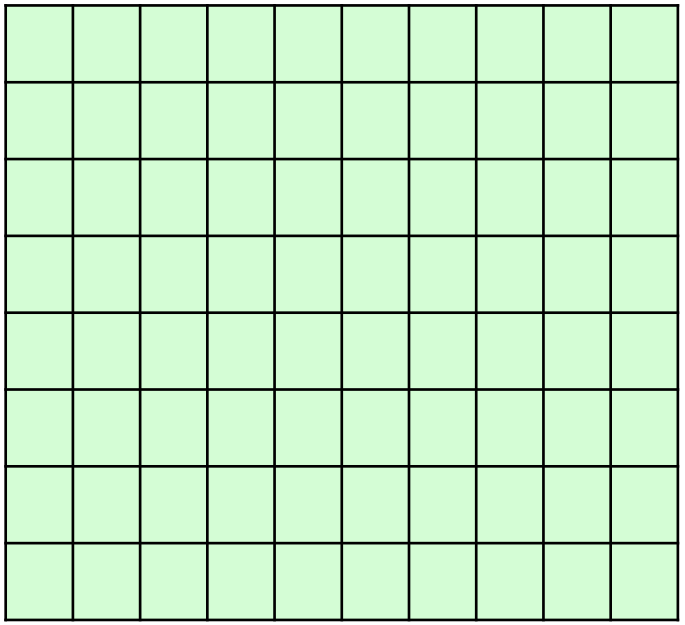
Cada pixel de um polígono só é escrito se não houver sobreposição com nenhum outro ou, caso contrário, se a sua distância ao observador for menor que a do pixel, já escrito, pertencente a polígono tratado em fase anterior

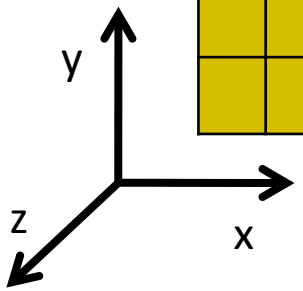
Dados: os polígonos planos, em projecção ortogonal e conhecidos os valores de profundidade (em z)

Buffer de refrescamento, de dimensão do ecrã e inicializado com a cor do fundo

Buffer de profundidade, de dimensão do ecrã e inicializado com o menor valor possível

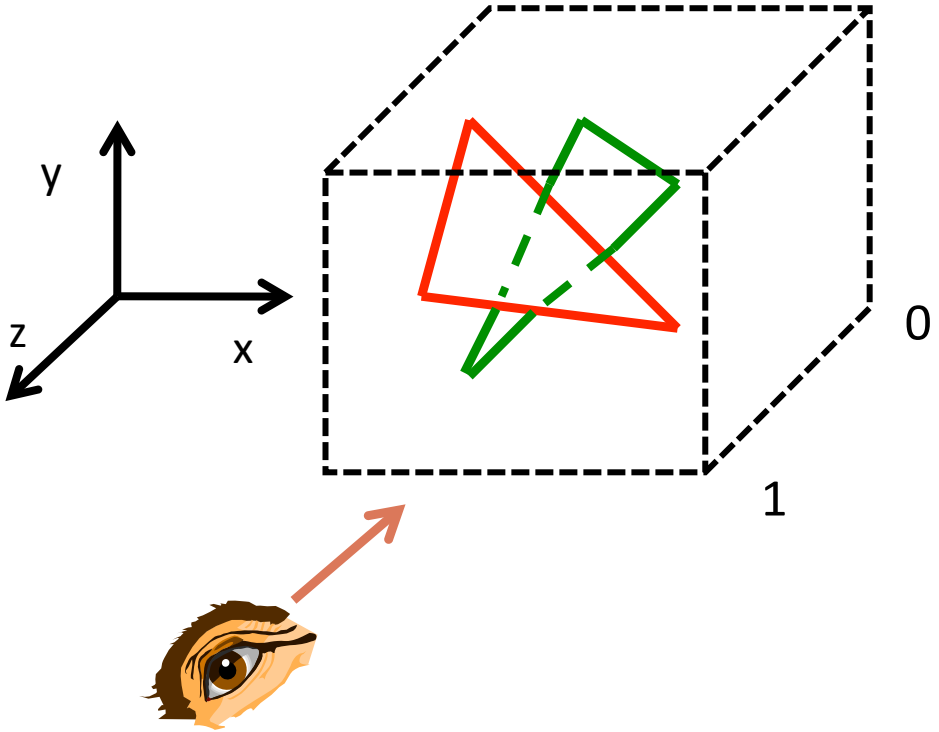


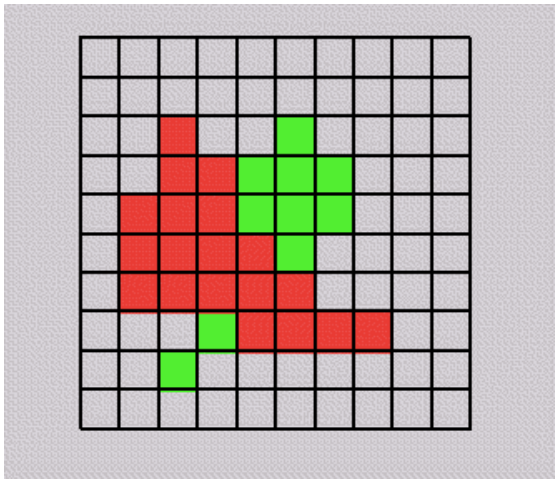




No final, o buffer de profundidade contém os valores de z visíveis e o buffer de refrescamento as cores associadas a esses polígonos visíveis (ou partes)

Considerando os polígonos após a transformação de projecção, com a coordenada z normalizada no intervalo [0,1]

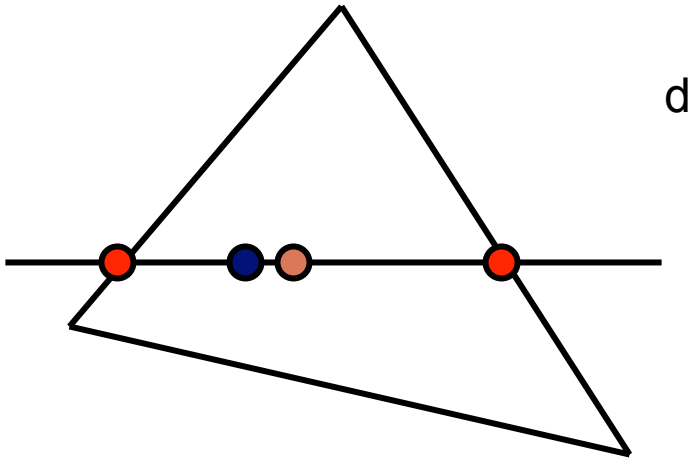




Exemplo de melhoria de eficiência no cálculo (incremental) da profundidade z

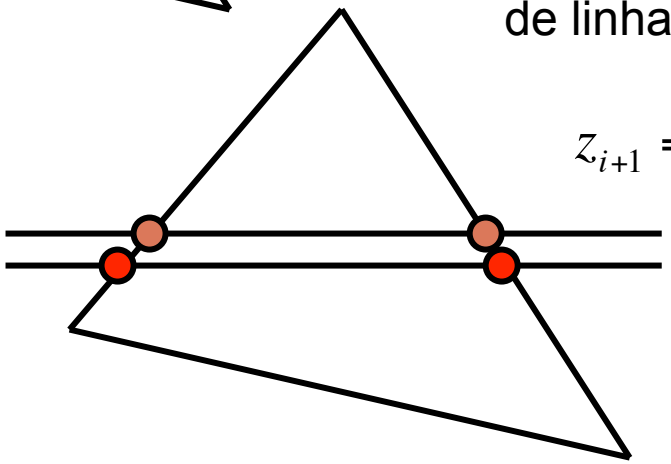
equação do polígono:

$$A\,x + B\,y + C\,z + D = 0$$



dentro de uma linha de varrimento

$$z_{i+1} = z_i - k_1\ , \quad com\ k_1 = \frac{A}{C}$$



de linha para linha

$$z_{i+1} = z_i - k_2\ , \quad com\ k_2 = \frac{B}{C}$$

Algoritmo

```
Inicializa o buffer de refrescamento(cBuffer) com a cor de fundo
Inicializa o buffer de profundidade (zBuffer) com o z mínimo
Para a projecção de cada polígono ou objecto (P/O)
   Para cada linha de varrimento dentro dos limites de projecção do P/O
         Para cada pixel dentro desses limites
                  calcula a distância ao observador (profundidade) do pixel z(x,y)
                  Se z(x,y) > zBuffer(x,y) então
                      calcula a cor do pixel para z(x,y) do P/O
                      escreve a cor do pixel em cBuffer(x,y)
                      escreve o valor z correspondente em zBuffer(x,y)
                  Fimse
         Fimpara
   Fimpara
Fimpara
```

Obs.: Sendo o algoritmo aplicado após a projecção, é necessário preservar a coordenada z

Vantagens

- Simplicidade
- Implementação fácil em hardware
- Os polígonos podem ser processados numa ordem arbitrária
- Resposta simples ao problema de polígonos que se “entrelaçam”

Desvantagens

- Alguns cálculos seriam dispensáveis
- Problemas de precisão numérica em cenas complexas, nomeadamente com a projecção perspectiva e com polígonos muito próximos (e.g. dois polígonos paralelos e muito próximos um do outro)
- Gestão complicada dos efeitos de transparência e anti-aliasing tendo em conta a ordem arbitrária da escrita dos pixels no buffer