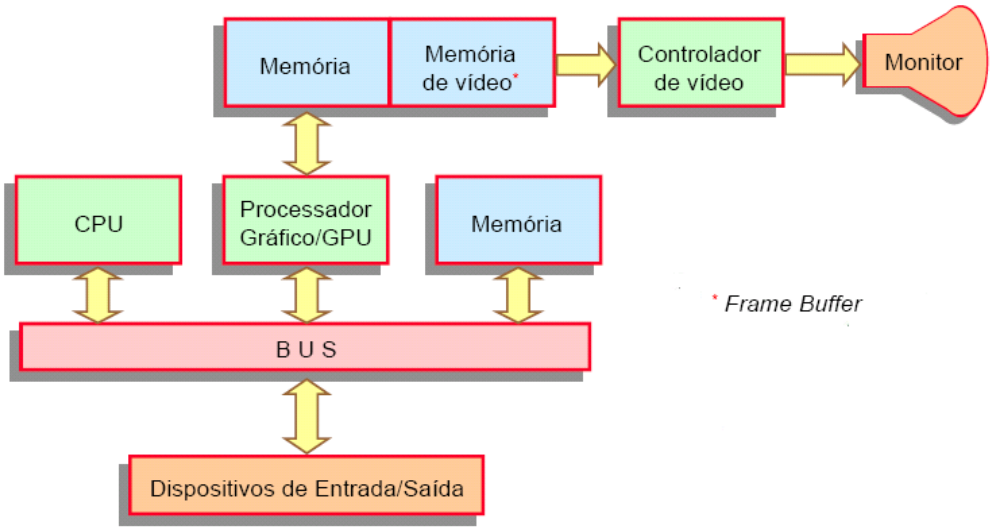
Computação Gráfica

- Criação, armazenagem e manipulação de modelos de objectos e subsequentes imagens por meio de computador.

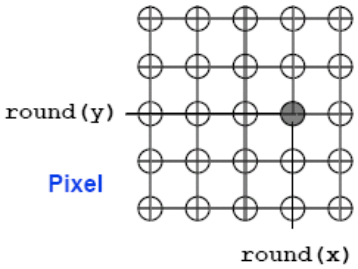
=> Arquitectura dum sistema gráfico *raster*:



Rasterização de primitivas gráficas

=> **Pontos:**

WRITE_PIXEL(round(x), round(y), value**)**



=> **Polígonos:**

- Preenchimento do polígono definido pela sequência dos vértices, independentemente dos valores pré-existentes na memória (2D).

1. Algoritmo Par-Impar (*Even-Odd*)

- É um algoritmo de varrimento por linhas, em que os pixels são testados não individualmente, mas tirando partido da…

a) coerência por linha de varrimento:

- Se um dado pixel duma linha pertence ao polígono, é provável que os pixels vizinhos também pertençam.

b) coerência por aresta do polígono:

- Se uma aresta intersecta uma dada linha de varrimento, é provável que intersecte também as linhas vizinhas.

- Etapas do algoritmo *FILL AREA*: (Para cada linha de varrimento)

1. Calcular todas as intersecções com as arestas do polígono

   Nota: as arestas horizontais são excluídas do algoritmo
2. Ordenar as intersecções segundo os valores crescentes da abcissa
3. Preencher todos os pixels entre pares de intersecções

- Alguns vértices dão um número ímpar de intersecções, o que leva a preenchimento incorrecto. Para resolver este problema faz-se o preenchimento entre cada par de intersecções, mas exclui-se as arestas para as quais a ordenada da linha de varrimento seja as sua ordenada máxima.

=> **Linhas:**

- Segmentos de recta definem-se pelas coordenadas dos extremos P_1 e P_2 por hipótese de valores inteiros.

- A cor, grossura e tipo de traço são atributos possíveis para as linhas. Há implementações em que o pixel corresponde a P_2 não é activado, permitindo que não seja escrito mais do que uma vez se se tratar duma linha poligonal.

1. Algoritmo Incremental (*DDA*)

```
procedure LINE (x1,y1,x2,y2,value: integer);
                                        { when  -1 ≤ m ≤ +1 only }
var   dx,dy,x,y,m: real;

begin
      if x1<>x2 then
      begin
            dy := y2-y1; dx := x2-x1; m := dy/dx; y := y1;
            for x:=x1 to x2 do
            begin
                  WRITE_PIXEL(x,round(y),value); y := y + m
            end
      end
      else if y1=y2 then WRITE_PIXEL(x1,y1,value)
end;
```

2. Algoritmo de *Bresenhan*

```
procedure LINE_BRESENHAM (x1,y1,x2,y2,value: integer);

   { Condição de aplicabilidade:  0 ≤ m ≤ 1
     Para resolver nos outros octantes faz-se uma conversão para este! }

var   dx,dy,ks,kt,d,x,y,x_end: integer;

begin
      dx := abs(x2-x1);   dy := abs(y2-y1);   d := 2*dy - dx;
      ks := 2*dy;  kt := 2*(dy - dx);
      if x1 > x2 then  begin  x := x2;  y := y2;  x_end := x1  end
                 else  begin  x := x1;  y := y1;  x_end := x2  end;
      WRITE_PIXEL(x,y,value);
      while x < x_end do
      begin
            x := x + 1;
            if d < 0 then  d := d + ks
                     else  begin  y := y + 1;  d := d + kt  end;
            WRITE_PIXEL(x,y,value)
      end
end;
```

2. Algoritmo do Ponto Médio

- O algoritmo de Bresenham não é de generalização fácil para as cónicas em geral

- Aplicação ao caso duma recta, cuja equação é

$$F(x,y) = ax + by + c = 0$$

- Comparando com a forma explícita

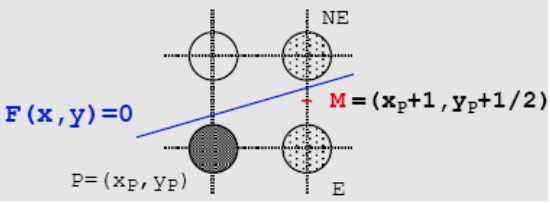
$$y = mx + B = (dy/dx)\,x + B$$

- determinam-se os coeficientes da forma implícita:

$$F(x,y) = dy\,x - dx\,y + B\,dx = 0 \quad \Rightarrow \quad \begin{aligned} a &= dy \\ b &= -dx \\ c &= B\,dx \end{aligned}$$

- Hipótese: $dy \geq 0$, por escolha da ordem entre os pontos que definem os segmento de recta

- Exemplo:



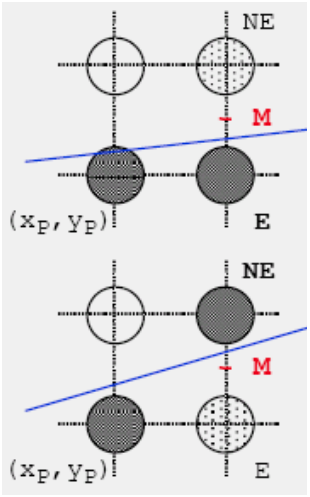
$$y = mx + b, com\ 0 \leq m \leq 1$$

- Comparação no ponto médio M para se conhecer o novo ponto P com abcissa x_p+1:

```
P  =  if  F(M) < 0  then  E
                    else  NE
```

$$F(M) = a(x_P + 1) + b(y_P + 1/2) + c = d'$$

- Como d` = F(M), no passo seguinte e quando..



..a escolha for E (novo M, para x_p+2, manterá a ordenada)

$$d' = F(x_P + 2,\ y_P + 1/2) = a(x_P + 2) + b(y_P + 1/2) + c$$

ou seja, em linguagem de programação: $d' := d' + a$

..a escolha for NE (novo M, para x_p+2, manterá a ordenada)

$$d' = F(x_P + 2,\ y_P + 3/2) = a(x_P + 2) + b(y_P + 3/2) + c$$

ou seja, em linguagem de programação: $d' := d' + a + b$

- Como, por hipótese, as coordenadas das extremidades do segmento de recta são os valores inteiros, a e b também o são:

Inicialização da variável de decisão no ponto $P(x_1, y_1)$:

$$\begin{aligned} F(x_1+1, y_1+1/2) &= a(x_1+1) + b(y_1+1/2) + c \\ &= a\,x_1 + b\,y_1 + c + a + b/2 \\ &= F(x_1, y_1) + a + b/2 \\ &= a + b/2 \end{aligned}$$

- O valor inicial da variável de decisão seria:

$$d' = F(x_1 + 1,\ y_1 + 1/2) = a + b/2$$

- Mas, para se trabalhar exclusivamente com valores inteiros, multiplique-se por 2 e substitua-se 2d` por d:

$$d = 2a + b = 2dy - dx$$

Conclusão: $P :=$ if $d<0$ then $\{E\}$

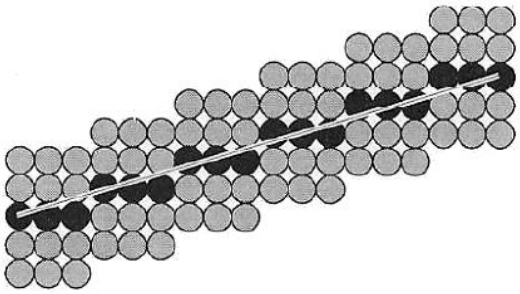
$$x := x + 1 \ \wedge\ d := d + 2 \times dy$$

else $\{NE\}$

$$x := x + 1\ ;\ y := y + 1 \ \wedge\ d := d + 2 \times (dy - dx)$$

Grossura das Linhas

=> **Replicação de Pixels**



+ Eficiente e facilmente adaptável aos algoritmos de conversão raster

− As linhas terminam sempre vertical ou horizontalmente

− Pode implicar existência de vazios nos pontos de ligação de troços

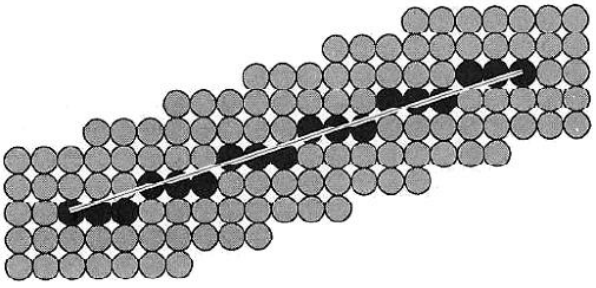
− É necessário decidir quando a grossura tiver um valor par

− Menor grossura ($\times 1/\sqrt{2}$) à aproximação de 45º

− Menos pixels na passagem de replicação vertical a horizontal

=> Boa solução para linhas pouco grossas

=> **Forma do Aparo**



+ Comparando com **Replicação de Pixels**, um aparo quadrado, por exemplo, melhora a qualidade dos extremos duma linha

+ Um aparo circular (se possível) evitará o problema

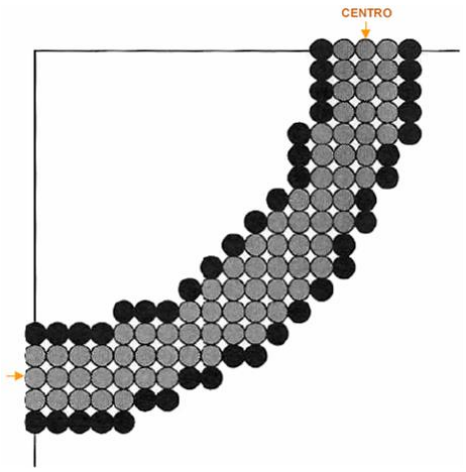
− Execução mais lenta

− Situações que não deve permitir sobreposição na escrita de pixels

− Altera o comprimento duma linha

− Grossura média varia com o declive, sendo maior ($\times\sqrt{2}$) a 45º

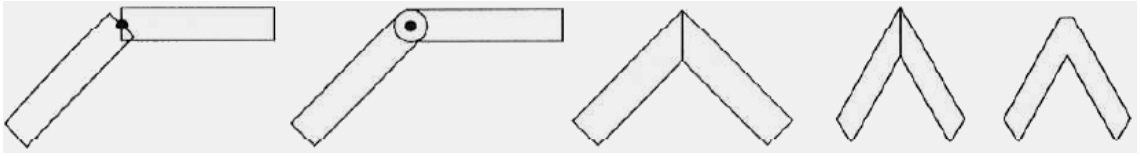
=> **Preenchimento de Áreas**



- Criação de duas fronteiras, igualmente afastadas para cada lado: a um *segmento de recta* corresponde um *rectângulo*.

+ Não altera o comprimento duma linha
+ Pode aplicar-se o algoritmo *FILL AREA* *
+ Um aparo circular (se possível) evitará o problema
− Dificuldades de realização em alguns casos (elipses)
− * mas se poderá notar desfasamento em relação à *linha central*

=> **Aproximação por poligonais (** Grossas **)**



- Uma curva é aproximada por troços rectilíneos, usando-se depois os algoritmos de conversão de linhas e polígonos.

+ Eficiência na geração (e recorte)
− Dificuldades com a aparência dos pontos de junção

Enquadramento

=> **Formato de um rectângulo**

$$a = \frac{X_{\max} - X_{\min}}{Y_{\max} - Y_{\min}}$$

Exemplos → 4:3 e 16:9

- Condição para manter as proporções da imagem no enquadramento:

$$a_{\text{janela}} = a_{\text{visor}} \longrightarrow \frac{X_2 - X_1}{Y_2 - Y_1} = \frac{X'_2 - X'_2}{Y'_2 - Y'_2} \longrightarrow E = B$$

=> **Coordenadas no Ecrã**

- Para cada ponto P(x,y) far-se-ia corresponder um vector $P = \begin{bmatrix} X \\ Y \end{bmatrix}$
- Para o caso do enquadramento janela – visor ter-se-ia:

$$\begin{aligned} x' &= (x - A).B + C \\ y' &= (y - D).E + F \end{aligned} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} B & 0 \\ 0 & E \end{bmatrix} \cdot \left(\begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} A \\ D \end{bmatrix} \right) + \begin{bmatrix} C \\ F \end{bmatrix}$$

- Mas haverá maneira de se obter o mesmo resultado apenas com operadores multiplicativos e do mesmo tipo?...

$$P' = M_3 . M_2 . M_1 . P$$

- As transformações de coordenadas resolvem-se uniformemente através do cálculo matricial usando coordenadas homogéneas. A cada ponto P(x,y,z) faz-se corresponder o vector $P=\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$. Portanto os operadores (M) sobre os pontos (P) são matrizes 3x3 em **2D** e 4x4 em **3D**. Usam-se na forma $P^{'}=M.P$

- Tratamento Matemático:

- Translação:

$$T(T_x,T_y)=\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} x^{'} &= x+T_x \\ y^{'} &= y+T_y \end{aligned}$$

- Mudança de escala:

$$S(S_x,S_y)=\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} x^{'} &= S_x \times x \\ y^{'} &= S_y \times y \end{aligned}$$

- Rotação:

$$R(\alpha)=\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Para resolver $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} B & 0 \\ 0 & E \end{bmatrix} \cdot \left(\begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} A \\ D \end{bmatrix} \right) + \begin{bmatrix} C \\ F \end{bmatrix}$ com operadores multiplicativos $P' = M_3 . M_2 . M_1 . P$ a solução é:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & C \\ 0 & 1 & F \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} B & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -A \\ 0 & 1 & -D \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

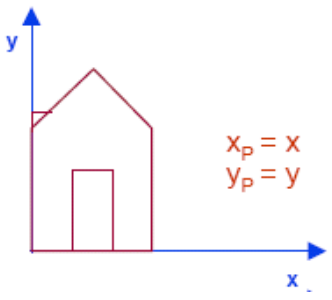
ou

$$P' = T(C,F).S(B,E).T(-A,-D).P$$

Projecções Geométricas planas

- A superfície de projecção é um plano.
- As projectantes são linhas rectas.

=> **Projecção Ortogonal**



$$P' = M_{ORT} . P$$

$$M_{ORT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

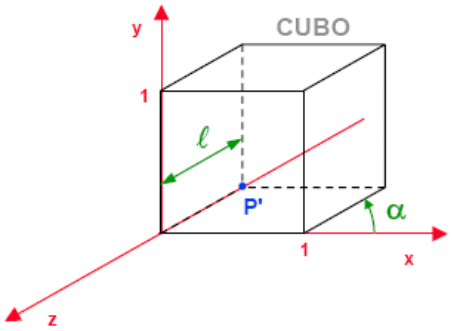
+ Mostra as dimensões exactas das faces paralelas ao plano de projecção

− Pode ser difícil avaliar a forma tridimensional do objecto

=> **Projecção Oblíqua**

ℓ – factor de redução ou de encurtamento

α – ângulo de fuga



$$M_{ORT} = \begin{bmatrix} 1 & 0 & -\ell\cos\alpha & 0 \\ 0 & 1 & -\ell\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- As projecções oblíquas são determinadas / caracterizadas por:

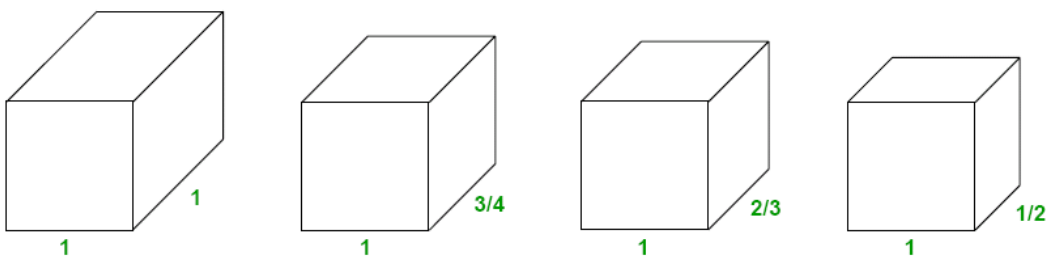
a) Pelo ângulo(β) que as projectantes fazem com o plano de projecção (z=0)

b) Pela orientação das projectantes, independentemente do ângulo com o plano de projecção (valores de 45º ou 30º para α)

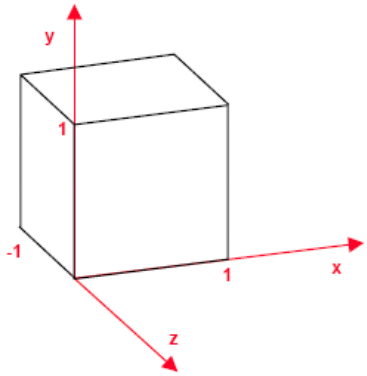
Cavaleira: $\ell = 1 \quad \leftrightarrow \quad \beta = 45º$

Gabinete: $\ell = 0{,}5 \quad \leftrightarrow \quad \beta = 63{,}4º$

Ortogonal: $\ell = 0 \quad \leftrightarrow \quad \beta = 90º$

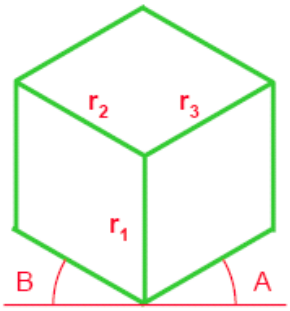


=> **Projecção Axonométrica**



$$M_{AX} = M_{ORT} \,.\, R_X(\gamma).R_Y(\theta) \quad (z = 0)$$

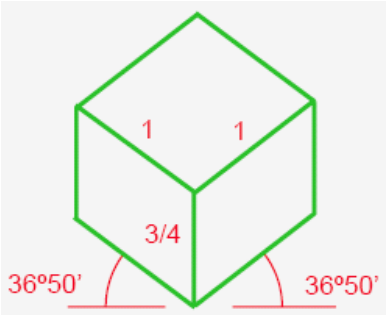
- <u>Desenho Axonométrico</u>



- *Isometria*:

→ A = B = 30º

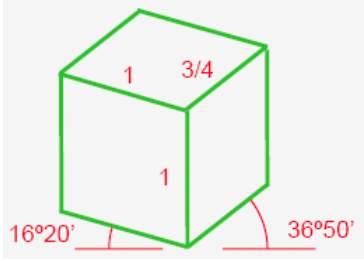
→ $r_1 = r_2 = r_3 = 1$



- *Dimetria*:

→ A = B = 36º 50´

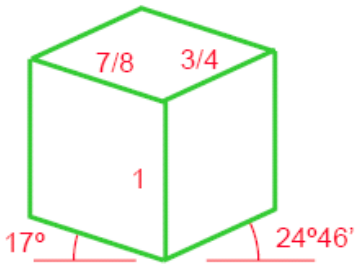
→ $r_1 = \frac{3}{4}$, $r_2 = r_3 = 1$



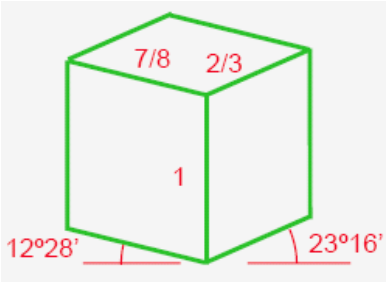
→ A = 16º 20´ | B = 36º 50´

→ $r_1 = r_2 = 1$, $r_3 = \frac{3}{4}$

- *Trimetria*:



→ A = 17º | B = 24º 46´
→ $r_1 = 1$, $r_2 = 7/8$ $r_3 = 3/4$



→ A = 12º 28´ | B = 23º 16´
→ $r_1 = 1$, $r_2 = 7/8$ $r_3 = 2/3$

- <u>Projecção Axonométrica</u>

→ $\theta = arctg\left(\sqrt{tg(A)/tg(B)}\right) - \pi/2$

→ $\gamma = arcsen\left(\sqrt{tg(A).tg(B)}\right)$

→ $r_1 = cos(\gamma)$ $r_2 = cos(\theta)/cos(B)$ $r_3 = -sen(\theta)/cos(A)$

- Uma projecção axonométrica é determinada / caracterizada:

a) Pelos ângulos que os eixos coordenados locais ao objecto fazem com o plano de projecção (ou…)

b) Pelos três factores de escala (ou…)

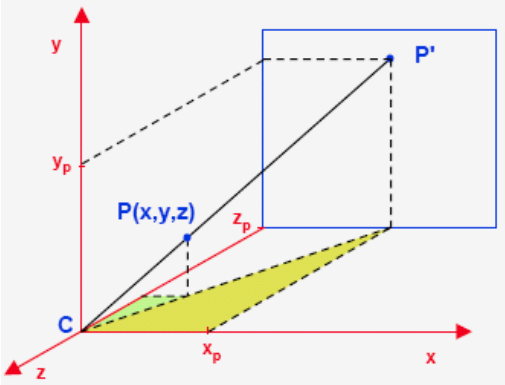
c) Pelos ângulos entre os eixos coordenados depois de projectados

<u>Conclusões</u>:

- O paralelismo de linhas é preservado mas os ângulos não o são
- Os comprimentos são medidos usando-se factores de escala correspondentes às três direcções axiais.

=> **Projecção Perspectiva**

- Plano de projecção em z = d ≠ 0 e centro de projecção C na origem:



$$\frac{x_P}{d} = \frac{x}{z} \longrightarrow x_P = \frac{x}{z/d}$$

$$\frac{y_P}{d} = \frac{y}{z} \longrightarrow y_P = \frac{y}{z/d}$$

$$z_P = d \longrightarrow z_P = \frac{z}{z/d}$$

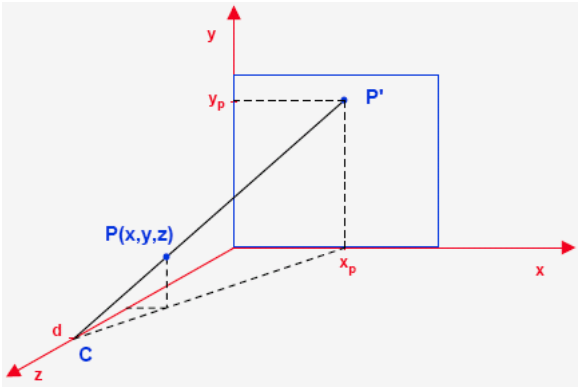
- Coordenadas da imagem de **P**: $P' = \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$

- Coordenadas homogéneas de **P'**: $P' = \begin{bmatrix} X = x \\ Y = y \\ Z = z \\ W = z/d \end{bmatrix}$

- Podem obter-se de **P** pela aplicação da M_{PER}: $\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = M_{PER} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

$$M_{PER} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

- *Plano de projecção em z = 0 e centro de projecção C na (0,0,d) com d ≠ 0::*



$$\frac{x_P}{d} = \frac{x}{d-z} \longrightarrow x_P = \frac{x}{1-z/d}$$

$$\frac{y_P}{d} = \frac{y}{d-z} \longrightarrow y_P = \frac{y}{1-z/d}$$

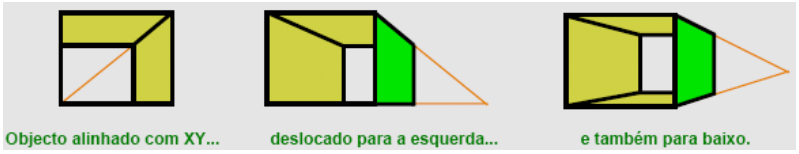
$$z_P = 0 \longrightarrow z_P = \frac{0}{1-z/d}$$

- Podem obter-se de **P** pela aplicação da M'$_{PER}$: $\begin{bmatrix} x \\ y \\ z \\ 1-z/d \end{bmatrix} = M'_{PER} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

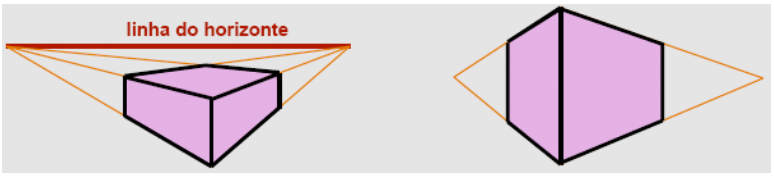
$$M'_{PER} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \qquad d \to \infty \Rightarrow M'_{PER} \to M_{ORT}$$

- *Implicações do paralelismo das direcções principais do objecto com os eixos*

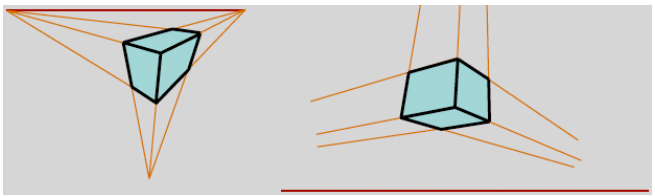
1 ponto de fuga:



2 ponto de fuga:



3 ponto de fuga:



Curvas e Superfícies

=> **Curva**

- Especificada por uma ou mais equações com **uma** só variável independente

- Descrição não-paramétrica

  - Forma explícita:

  $$Q = (x, y = f(x), z = g(x))$$

  - Forma implícita:

  $$Q = (F(x, y, z) = 0, G(x, y, z) = 0)$$

- Descrição paramétrica

$$Q = (x = f(t), y = g(t), z = h(t)) \qquad a \le t \le b$$

=> **Superfície**

- Especificada por uma ou mais equações com **duas** variáveis independentes

- Descrição não-paramétrica

  - Forma explícita:

  $$Q = (x, y, z = f(x, y))$$

  - Forma implícita:

  $$Q = (F(x, y, z) = 0)$$

- Descrição paramétrica

$$Q = (x = f(s,t), y = g(s,t), z = h(s,t)) \qquad a \le t \le b\,,\ c \le s \le d$$

=> **Curvas e Superfícies**

- Descrição não-paramétrica

  - Forma explícita:

  1. Facilidade de cálculo
  2. Não pode representar uma correspondência que não seja função
  3. Não se podem aplicar directamente transformações por operadores matriciais

  Exemplo: $y = \sqrt{4 - x^2}$ e $y = -\sqrt{4 - x^2}$, $-2 \le x \le 2$

  - Forma implícita:

  1. Representação de correspondências que não sejam funções
  2. Pode ser difícil a determinação das raízes
  3. Não se podem aplicar directamente transformações por operadores matriciais

  Exemplo: $x^2 + y^2 - 4 = 0$, $-2 \le x \le 2$

- Descrição não-paramétrica

  1. Representação de correspondências que não sejam funções
  2. Podem aplicar-se directamente transformações por operadores matriciais

  Exemplo:
  $$\begin{cases} x = 2\cos(t) \\ y = 2\,sen(t) \end{cases}, \; 0 \le t \le 2\pi$$
  ou
  $$\begin{cases} x = 2\cos(2\pi t) \\ y = 2\,sen(2\pi t) \end{cases}, \; 0 \le t \le 1$$

- Alguns requisitos no design de **Curvas**:

- Interpolar ou aproximar um certo número de pontos conhecidos, obtendo-se a equação da curva

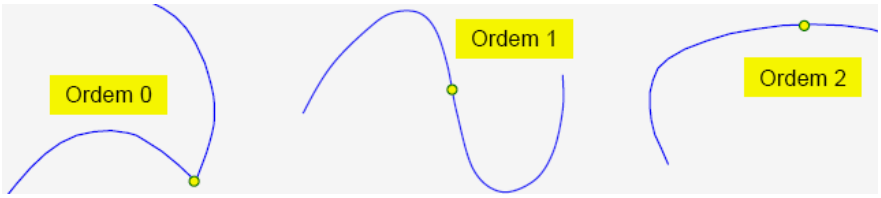
- Controlar através de pontos conhecidos e de forma previsível (local ou globalmente)

- Haver independência da forma da curva em relação aos eixos

- Permitir correspondências que não sejam funções

- Existir tendência para suavizar pequenas irregularidades

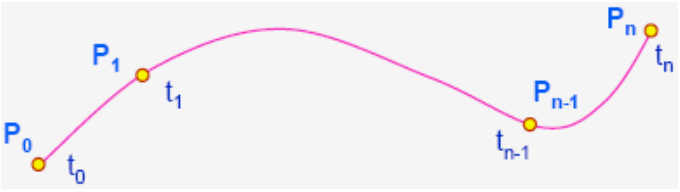
- Permitir a continuidade entre os troços que constituam uma curva complexa



1ª Conclusão: Usar descrições paramétricas

=> **Curvas de Interpolação**

- Problema:



- Encontrar uma curva Q(t) que passe por n+1 pontos P_i

- Condições do problema:

1. $Q(t_i) = P_i \ , \ i = 0...n$
2. Continuidade das funções e suas derivadas

- Resolução:

1. Usar polinómios interpoladores

1. Interpolação de Lagrange

- Existe um e um só polinómio de grau n que resolve o problema:

$$Q(t) = \sum_{i=0}^{n} P_i L_{in}(t)$$

em que, $$L_{in}(t) = \frac{(t-t_0)...(t-t_{i-1})(t-t_{i+1})...(t-t_n)}{(t_i-t_0)...(t_i-t_{i-1})(t_i-t_{i+1})...(t_i-t_n)}$$

- Mas um polinómio de grau n tem até n-1 extremos relativos e n-2 pontos de inflexão

- Em medições precisas, quando se usam muitos pontos, o resultado poderá ser uma oscilação indesejada da curva

2. Interpolação por Splines (naturais)

- A teoria dos splines trata da interpolação polinomial, no geral e por troços

- Definição de Spline:

  - Uma função S(t), escalar ou vectorial, definida no intervalo $[t_0, t_n]$, é um SPLINE de ordem k (ou grau k-1) se:

    1. S(t) é um polinómio de grau k-1 em cada intervalo $[t_i, t_{i+1}]$, com $t_o < \ldots < t_i < t_{i+1} < \ldots < t_n$
    2. S(t) e as suas derivadas de ordem 1..k-2 são contínuas em todo o intervalo onde é definida

  - *Dados*: n+1 pontos $P_i \ (i = 0...n)$ e nós $t_i \in [t_0, t_n]$

  - *Objectivos*: Encontrar os polinómios cúbicos interpoladores por troços e que definem a função Q(t) em $[t_0, t_n]$ tal que pertence a C^2 nos nós

- Dedução dos coeficientes:

  - Sendo polinómio cúbico, para o intervalo $[t_k, t_{k+1}]$ pode escrever-se:

$$Q_k(t) = a_k + b_k\,(t - t_k) + c_k\,(t - t_k)^2 + d_k\,(t - t_k)^3$$

  - Condições fronteira:

$$Q(t_k) = P_k \qquad Q(t_{k+1}) = P_{k+1}$$

$$\left(\frac{d\,Q_k}{dt}\right)_{t=t_k} = R_k \qquad \left(\frac{d\,Q_k}{dt}\right)_{t=t_{k+1}} = R_{k+1}$$

  - Substituindo e resolvendo dá:

$$a_k = P_k \qquad b_k = R_k$$

$$c_k = \frac{3\,(P_{k+1} - P_k)}{h_k^2} - \frac{2\,R_k}{h_k} - \frac{R_{k+1}}{h_k} \qquad d_k = \frac{2\,(P_k - P_{k+1})}{h_k^3} + \frac{R_k}{h_k^2} + \frac{R_{k+1}}{h_k^2}$$

$$\text{com} \quad h_k = t_{k+1} - t_k$$

- Condições adicionais:

  - As 4 equações deduzidas para os coeficientes referem-se a cada um dos troços, pelo que haverá ao todo 4n equações. As derivadas R_i ainda não são conhecidas, pelo que precisamos de n+1 equações.

  - Condição para a segunda derivada:

$$\left(\frac{d^2 Q_k}{dt^2}\right)_{t=t_{k+1}} = \left(\frac{d^2 Q_{k+1}}{dt^2}\right)_{t=t_{k+1}}$$

  - Ao todo serão n-1 equações, da forma:

$$2\,c_k + 6\,d_k\,(t_{k+1} - t_k) = 2\,c_{k+1}$$

  - Para não deixarmos o sistema indeterminado, há que introduzir mais 2 equações:

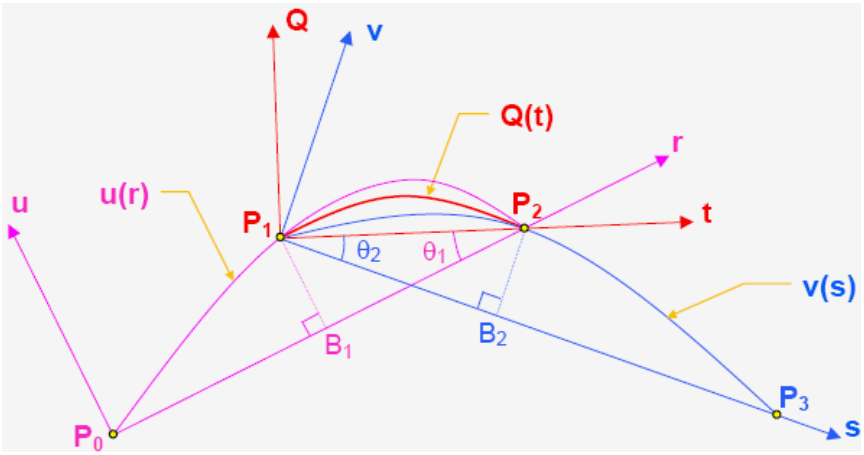
- Desvantagens dos Splines naturais:

  - Qualquer que seja a alternativa tomada para a resolução da indeterminação do sistema, o ajuste da curva resultante depende inteiramente da qualidade dos pontos dados, não se garantindo, portanto, a ausência de oscilações indesejáveis.

  - Outro inconveniente é o do controlo global (e não local) da curva

Conclusão: Não é boa solução para design interactivo

3. Interpolação por Parábolas



- Para se obter Q(t) entre P_1 e P_2 calculam-se duas parábolas u(r) e v(s), definidas por três pontos cada. Q(t) obtém-se por média ponderada de u(r) e v(s), usando-se uma função linear em t:

$$Q(t) = u(r)\,(t_0 - t)/t_0 + v(s)\,t/t_0 \qquad \text{com} \quad 0 \le t \le t_0 \quad \text{e} \quad t_0 = |P_2 - P_1|$$

⟶ Características:

1. Q(t) é cúbica em t
2. Q(t) é de classe C^1
3. O controlo é local, apropriado para design interactivo
4. O cálculo é pesado…

- Cálculos:

  - Parábola definida por P_0 , P_1 e P_2 $(0 \le r \le d_r)$:

$$u(r) = P_0 + r\,(P_2 - P_0)\,/\,d_r + \alpha\, r\,(d_r - r)\,(P_1 - B_1)$$

  - com:

$$d_r = |P_2 - P_0| \qquad B_1 = P_0 + x_r\,(P_2 - P_0)$$

$$x_r = (P_1 - P_0)\,(P_2 - P_0)\,/\,(P_2 - P_0)^2 \qquad \alpha^{-1} = d_r^{\,2}\, x_r\,(1 - x_r)$$

$$r = x_r\, d_r + t \cos\theta_1 \qquad \cos\theta_1 = (P_2 - P_1)\,(P_2 - P_0)\,/\,(t_0\, d_r)$$

- Parábola definida por P_1 , P_2 e P_3 $(0 \le s \le d_s)$:

$$v(s) = P_1 + s\,(P_3 - P_1) / d_s + \beta\, s\,(d_s - s)\,(P_2 - B_2)$$

- com:

$$d_s = |\,P_3 - P_1\,| \qquad B_2 = P_1 + x_s\,(P_3 - P_1)$$

$$x_s = (P_2 - P_1)\,(P_3 - P_1) / (P_3 - P_1)^2 \qquad \beta^{-1} = d_s^{\,2}\, x_s\,(1 - x_s)$$

$$s = x_s\, d_s + t \cos\theta_2 \qquad \cos\theta_2 = (P_2 - P_1)\,(P_3 - P_1) / (t_0\, d_s)$$

=> **Curvas cúbicas**

- Curva no espaço 3D:

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \\ a_z t^3 + b_z t^2 + c_z t + d_z \end{bmatrix} = \begin{bmatrix} [\,t^3\ t^2\ t\ 1\,] \,.\, [\,a_x\ b_x\ c_x\ d_x\,]^T \\ [\,t^3\ t^2\ t\ 1\,] \,.\, [\,a_y\ b_y\ c_y\ d_y\,]^T \\ [\,t^3\ t^2\ t\ 1\,] \,.\, [\,a_z\ b_z\ c_z\ d_z\,]^T \end{bmatrix}$$

- Sempre que não estiver em causa apenas uma coordenada em particular, por comodidade usar-se-á a expressão geral (vectorial)

$$Q(t) = a\,t^3 + b\,t^2 + c\,t + d = [\,t^3\ t^2\ t\ 1\,] \,.\, \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = T.A$$

Conclusão: 4 coeficientes arbitrários ⟶ pode-se impor 4 condições

=> **Curvas de Hermite**

- **Condições**: 2 pontos a interpolar e vectores tangentes nesses pontos

$$Q(0) = P_0 \qquad Q'(0) = R_0 \qquad Q(1) = P_3 \qquad Q'(1) = R_3$$

- Na forma matricial:

$$\begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix} = \begin{bmatrix} P_0 \\ P_3 \\ R_0 \\ R_3 \end{bmatrix}$$

- Substituindo cada elemento Q(t) por T.A escrever-se-á:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} . A = G_H$$

- Resolvendo em ordem a A dará:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} . G_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} . G_H = M_H . G_H$$

pelo que uma curva de Hermite é da forma: $Q(t) = T.M_H.G_H$

=> **Curvas de Bézier**

- **Condições**: As mesmas das curvas de Hermite, introduzindo-se 2 pontos intermédios que determinam os vectores tangentes.

$$Q'(0) = R_0 = 3\,(P_1 - P_0) \qquad Q'(1) = R_3 = 3\,(P_3 - P_2)$$

,donde:

$$\begin{bmatrix} P_0 \\ P_3 \\ R_0 \\ R_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} . \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

- Pelo que: $Q(t) = T.M_H.G_H \quad \Rightarrow \quad Q(t) = T.M_B.G_B$

com a Matriz de Bézier:

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- Exemplo geral da utilização de Blending Functions $B_{in}(t)$ ordem **n**:

$$Q(t) = P_0\, B_{0n}(t) + P_1\, B_{1n}(t) + P_2\, B_{2n}(t) + \ldots + P_m\, B_{mn}(t)$$

com $0 \le t \le 1$ e $n = m + 1$

- Blending Functions de ordem **n=4** para as Curvas de Bézier:

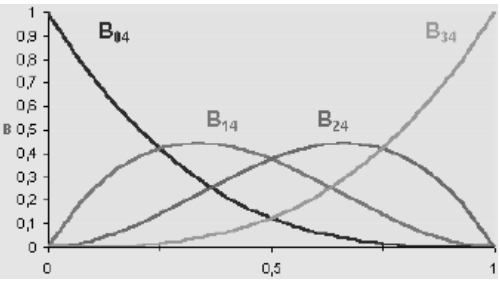
→ Se $Q(t) = T.M_B.G_B$ então $T.M_B = [B_{04} \quad B_{14} \quad B_{24} \quad B_{34}]$

$B_{04}(t) = -t^3 + 3t^2 - 3t + 1$

$B_{14}(t) = 3t^3 - 6t^2 + 3t$

$B_{24}(t) = -3t^3 + 3t^2$

$B_{34}(t) = t^3$



=> **Aproximação Bézier-Bernstein**

$$Q(t) = \sum_{i=0}^{m} P_i\, B_{in}(t)$$

com $0 \le t \le 1$ e $n = m + 1$

- Curva de ordem **n**, aplicada a **m+1** pontos e tendo como funções de peso (*Blending Functions*) os Polinómios de Bernstein:

$$B_{k,n}(t) = \frac{(n-1)!}{k!\,(n-1-k)!}\, t^k\, (1-t)^{n-1-k}$$

→ As curvas cúbicas de Bézier são o caso particular em que **n=4** pontos

=> **Algoritmo de *DE CASTELJAU***

- A partir dos pontos dados P_{0i} definem-se pontos auxiliares:

$$P_{k,n}(t) = (1 - t)\, P_{k-1,n-1}(t) + t\, P_{k-1,n}(t)$$

- Exemplo: Verificar que $P_{33}(t)$ corresponde à curva cúbica de Bézier

- Resolução:

$$\begin{aligned} P_{33}(t) &= (1 - t)\, P_{22}(t) + t\, P_{23}(t) \\ &= (1 - t)\,((1 - t)\, P_{11}(t) + t\, P_{12}(t)) + \\ &\quad t\,((1 - t)\, P_{12}(t) + t\, P_{13}(t)) \\ &= (1 - t)^2\,((1 - t)\, P_{00} + t\, P_{01}) + \\ &\quad 2\,(t - t^2)\,((1 - t)\, P_{01} + t\, P_{02}) + t^2\,((1 - t)\, P_{02} + t\, P_{03}) \\ &= (1 - t)^3\, P_{00} + (3t - 6t^2 + 3t^3)\, P_{01} + (3t^2 - 3t^3)\, P_{02} + t^3\, P_{03} \\ &= Q_{Bézier}(t) \end{aligned}$$

=> **Curvas B-Spline**

- O vector de geometria G_{Bs} é igual ao de Bézier (G_B), sendo diferente a matriz B-Spline:

$$M_{Bs} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

- Blending Functions:

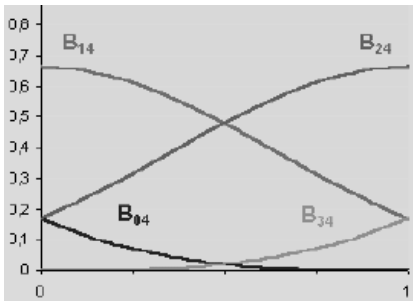
$$T \cdot M_{Bs} = [\, B_{04} \;\; B_{14} \;\; B_{24} \;\; B_{34} \,]$$

$$B_{04}(t) = (- t^3 + 3\, t^2 - 3\, t + 1) / 6$$

$$B_{14}(t) = (3\, t^3 - 6\, t^2 + 4) / 6$$

$$B_{24}(t) = (- 3\, t^3 + 3\, t^2 + 3\, t + 1) / 6$$

$$B_{34}(t) = t^3 / 6$$



Conclusão:

Início de curva ⟶ vizinhança de P_1

Final de curva ⟶ vizinhança de P_2

- Propriedades: $\mathbf{C^0}$

  - Sejam dados os seguintes vectores geometria:

$$G_{Bs_3} = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \qquad G_{Bs_4} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

  - Sendo $T \cdot M_{Bs} = \frac{1}{6} [\, -t^3 + 3t^2 - 3t + 1 \quad 3t^3 - 6t^2 + 4 \quad -3t^3 + 3t^2 + 3t + 1 \quad t^3 \,]$

  ,e considerando o intervalo $0 \le t \le 1$ para cada troço, por substituição obter-se-á:

$$Q_3(1) = \frac{1}{6} [\, 0 \quad 1 \quad 4 \quad 1 \,] G_{Bs_3}$$

$$Q_4(0) = \frac{1}{6} [\, 1 \quad 4 \quad 1 \quad 0 \,] G_{Bs_4}$$

$$Q_3(1) = Q_4(0)$$

Conclusão: Junção dos troços do mesmo ponto, implicando continuidade $\mathbf{C^0}$

- Propriedades $\mathbf{C^1}$

  - Primeira derivada:

$$\frac{d}{dt} T \cdot M_{Bs} = \frac{1}{2} [\, -t^2 + 2t - 1 \quad 3t^2 - 4t \quad -3t^2 + 2t + 1 \quad t^2 \,]$$

  - Considerando o intervalo $0 \le t \le 1$ para cada troço, por substituição obter-se-á:

$$\frac{d}{dt} Q_3(1) = \frac{1}{2} [\, 0 \quad -1 \quad 0 \quad 1 \,] G_{Bs_3}$$

$$\frac{d}{dt} Q_4(0) = \frac{1}{2} [\, -1 \quad 0 \quad 1 \quad 0 \,] G_{Bs_4}$$

$$\frac{d}{dt} Q_3(1) = \frac{d}{dt} Q_4(0)$$

Conclusão: Continuidade $\mathbf{C^1}$ no ponto de junção dos troços

- Propriedades $\mathbf{C^2}$

  - Primeira derivada:

$$\frac{d^2}{dt^2} T \cdot M_{Bs} = [\, -t+1 \quad 3t-2 \quad -3t+1 \quad t \,]$$

  - Considerando o intervalo $0 \le t \le 1$ para cada troço, por substituição obter-se-á:

$$\frac{d^2}{dt^2} Q_3(1) = [\, 0 \quad 1 \quad -2 \quad 1 \,]\, G_{Bs_3}$$

$$\frac{d^2}{dt^2} Q_4(0) = [\, 1 \quad -2 \quad 1 \quad 0 \,]\, G_{Bs_4}$$

$$\frac{d^2}{dt^2} Q_3(1) = \frac{d^2}{dt^2} Q_4(0)$$

Conclusão: Continuidade $\mathbf{C^2}$ no ponto de junção dos troços

- Continuidade e Interpolação

  - Resultado dos graus de multiplicidade dos pontos de controlo:

    > Multiplicidade 1 $\Rightarrow$ continuidade $\mathbf{C^2G^2}$

    > Multiplicidade 2 $\Rightarrow$ continuidade $\mathbf{C^2G^1}$

    > Multiplicidade 3 $\Rightarrow$ continuidade $\mathbf{C^2G^0}$

    > Multiplicidade 3 $\Rightarrow$ pode interpolar pontos de controlo

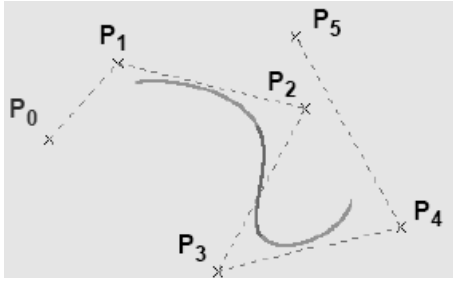
    > Multiplicidade 4 $\Rightarrow$ curva sem continuidade garantida

    > Multiplicidade 4 $\Rightarrow$ interpola até dois pontos de controlo

  - Há alternativa à interpolação de novo(s) ponto(s) de controlo dito(s) *fantasma(s)*

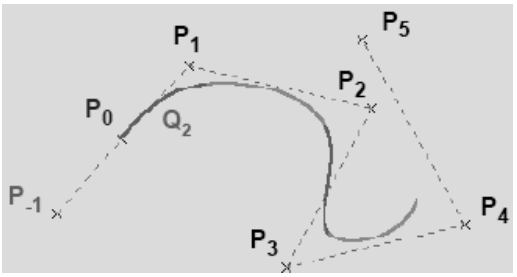
- Pontos de controlo *Fantasmas*

- Admitindo que se quer interpolar P_0

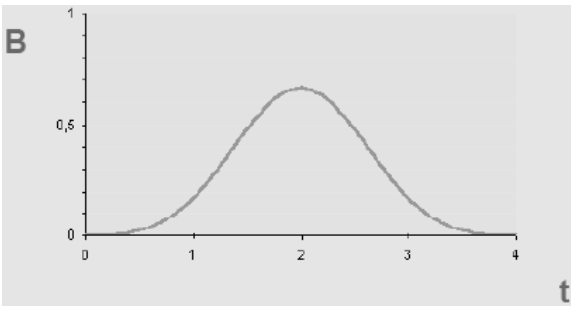


- Resolve-se matematicamente encontrando o ponto *fantasma* P_{-1} que verifique a condição:

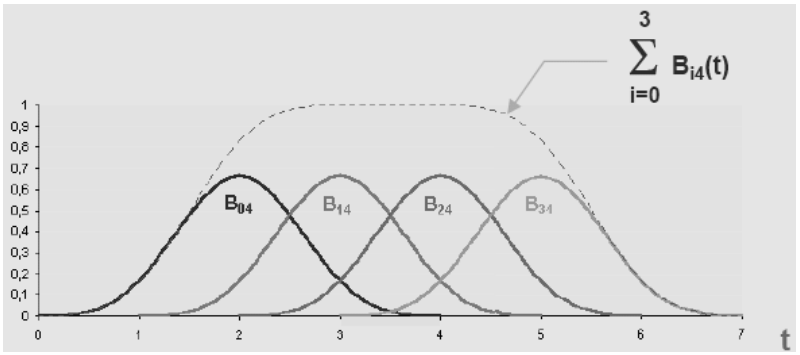
$$Q_2(t=0) = T \cdot M_{Bs} \cdot \begin{bmatrix} P_{-1} \\ P_0 \\ P_1 \\ P_2 \end{bmatrix} = P_0$$



- Para uma curva cúbica formada por diversos troços, um dado ponto de controlo contribui, no máximo, em 4 intervalos contíguos relativos ao parâmetro t. (Características evidenciadas)



- Graficamente, a função $B_{i+1,4}$ obtém-se de $B_{i,4}$ por translação de uma unidade em t. Mas a curva B-spline estará apenas definida nos <u>intervalos</u> em que os somem 1. (Uniformização resultante)



- Assim, para cada $B_{i,4}$ o parâmetro t variará de i até i+4.

- Supondo a existência de k pontos de controlo, define-se:

  → igual número de *Blending Functions*, desde $B_{0,4}$ até $B_{k-1,4}$

  → k+3 intervalos em t, sendo os limites dos intervalos designados por knot values

  → k+4 valores de nós

  → k-3 troços, de Q_3 até Q_{k-1}, respeitantes aos intervalos em que os pesos têm soma unitária

- Em vez de se ter que reduzir t ao intervalo [0,1] para a efectuação dos cálculos pelas fórmulas inicialmente dadas, os pesos deverão ter uma expressão analítica diferente.

- A melhor alternativa é a utilização do algoritmo recursivo de *Cox-deBoor*

- Esse algoritmo necessita conhecer os valores dos nós, usualmente expressos numa sequência não crescente designada por knot vector

- Algoritmo *Cox-deBoor*

$$B_{i,4}(t)=\frac{t-t_i}{t_{i+3}-t_i}B_{i,3}(t)+\frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}B_{i+1,3}(t)$$

$$B_{i,3}(t)=\frac{t-t_i}{t_{i+2}-t_i}B_{i,2}(t)+\frac{t_{i+3}-t}{t_{i+3}-t_{i+1}}B_{i+1,2}(t)$$

$$B_{i,2}(t)=\frac{t-t_i}{t_{i+1}-t_i}B_{i,1}(t)+\frac{t_{i+2}-t}{t_{i+2}-t_{i+1}}B_{i+1,1}(t)$$

$$B_{i,1}(t)=\begin{cases}1 & se\ \ t_i\le t<t_{i+1}\\ 0 & caso\ contrário\end{cases}$$

- Curvas B-spline cúbicas Não-Uniformes

- Alterar o comprimento de um ou mais intervalos é possível e permite maior versatilidade de formas sem modificação dos pontos de controlo.

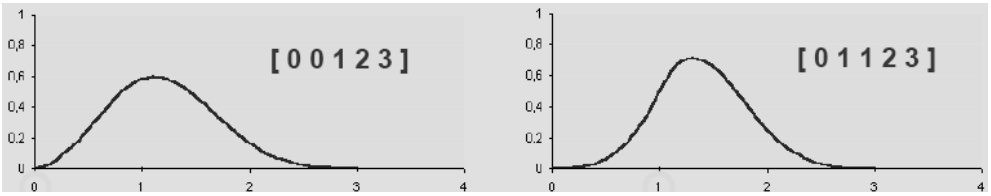
- É por vezes útil anular o comprimento de um ou mais intervalos

- Sempre que os intervalos entre nós não sejam todos iguais, a curva diz-se não-uniforme

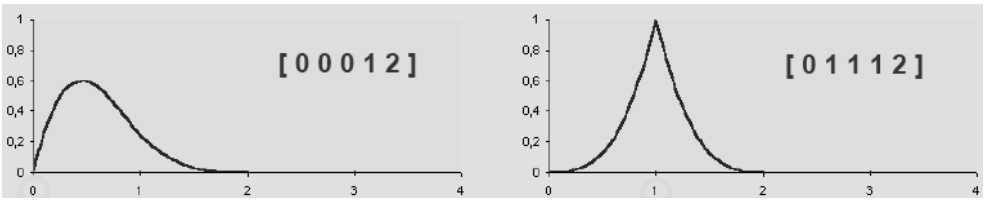
- O algoritmo de *Cox-deBoor* tanto se aplica às curvas B-spline cúbicas uniformes como às não-uniformes

- Usando o algoritmo de *Cox-deBoor*, o processo de cálculo para as curvas uniformes e não-uniformes é perfeitamente o mesmo

- Exemplos:



(multiplicidade dupla)



(multiplicidade tripla)

- NURBS (NonUniform Rational B-Spline)

  - Trata-se da divisão entre dois polinómios que são curvas B-Spline

  - Aplicação:

    - Calcula-se uma curva B-spline em coordenadas homogéneas

$$Q_{hom}(t) = [\, X=f(t) \;\; Y=g(t) \;\; Z=h(t) \;\; W=k(t) \,]^T$$

    , e converte-se em coordenadas reais

$$Q(t) = \left[\, x = \frac{X(t)}{W(t)} \;\; y = \frac{Y(t)}{W(t)} \;\; z = \frac{Z(t)}{W(t)} \,\right]^T$$

=> **Curvas Beta-Spline**

$$Q_i(t) = T \cdot M_{\beta s} \cdot G_{\beta s\, i}$$

- O vector da geometria $G_{\beta si}$ é igual ao do B-Spline G_{Bsi}, sendo a matriz $\beta - Spline$ dada por:

$$M_{\beta s} = \frac{1}{k} \begin{bmatrix} -2\beta_1^3 & 2(\beta_2 + \beta_1^3 + \beta_1^2 + \beta_1) & -2(\beta_2 + \beta_1^2 + \beta_1 + 1) & 2 \\ 6\beta_1^3 & -3(\beta_2 + 2\beta_1^3 + 2\beta_1^2) & 3(\beta_2 + 2\beta_1^2) & 0 \\ -6\beta_1^3 & 6(\beta_1^3 - \beta_1) & 6\beta_1 & 0 \\ 2\beta_1^3 & \beta_2 + 4(\beta_1^2 + \beta_1) & 2 & 0 \end{bmatrix}$$

, e em que: $k = \beta_2 + 2\,\beta_1^3 + 4\,\beta_1^2 + 4\,\beta_1 + 2$

- Parâmetros globais: $\beta_1 > 0$ (*Bias*) e $\beta_2 \geq 0$ (*Tension*)

- Propriedades de β-Splines:

  - Se $\beta_1 = 0$ e $\beta_2 = 0$ então $M_{\beta s} = M_{Bs}$

  - Convex Hull

  - Controlo global por β_1 e β_2 se aplicarem a toda a curva

  - C^0 e G^2 nas junções de troços

  - C^1 nas junções de troços quando $\beta_1 = 1$

=> **Curvas de Catmull-Rom (ou Splines de Overhauser)**

$$Q_i(t) = T \cdot M_{CR} \cdot G_{Bs_i}$$

$$M_{CR} = \frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

- Blending Functions:

$$T \cdot M_{CR} = [\ B_{04} \quad B_{14} \quad B_{24} \quad B_{34}\ ]$$

$$B_{04}(t) = (-t^3 + 2t^2 - t)/2$$
$$B_{14}(t) = (3t^3 - 5t^2 + 2)/2$$
$$B_{24}(t) = (-3t^3 + 4t^2 + t)/2$$
$$B_{34}(t) = (t^3 - t^2)/2$$

- <u>Propriedades</u>

  - Considerando o intervalo $0 \le t \le 1$ para cada troço:

$$Q_i(0) = \frac{1}{2}[\ 0 \quad 2 \quad 0 \quad 0\ ]\ G_{Bs_i} = P_{i-2}$$
$$Q_i(1) = \frac{1}{2}[\ 0 \quad 0 \quad 2 \quad 0\ ]\ G_{Bs_i} = P_{i-1}$$
$$Q_{i+1}(0) = \frac{1}{2}[\ 0 \quad 2 \quad 0 \quad 0\ ]\ G_{Bs_{i+1}} = P_{i-1}$$

$$Q_i(1) = Q_{i+1}(0)$$

(C^0 e Interpolação de pontos de controlo)

$$\frac{d}{dt} T \cdot M_{CR} = \frac{1}{2}[-3t^2 + 4t - 1 \quad 9t^2 - 10t \quad -9t^2 + 8t + 1 \quad 3t^2 - 2t\ ]$$

$$\frac{d}{dt} Q_i(1) = \frac{1}{2}[\ 0 \quad -1 \quad 0 \quad 1\ ]\ G_{Bs_i} = \frac{P_i - P_{i-2}}{2}$$
$$\frac{d}{dt} Q_{i+1}(0) = \frac{1}{2}[\ -1 \quad 0 \quad 1 \quad 0\ ]\ G_{Bs_{i+1}} = \frac{P_i - P_{i-2}}{2}$$

$$\frac{d}{dt} Q_i(1) = \frac{d}{dt} Q_{i+1}(0)$$

(Continuidade de C^1)

=> **Superfícies Bicúbicas na forma Paramétrica**

$$Q(s,t) = a_{11} * s^3 * t^3 + a_{12} * s^3 * t^2 + a_{13} * s^3 * t + a_{14} * s^3 + a_{21} * s^2 * t^3 + a_{22} * s^2 * t^2 + a_{23} * s^2 * t + a_{24} * s^2 + a_{31} * s * t^3 + a_{32} * s * t^2 + a_{33} * s * t + a_{34} * s + a_{41} * t^3 + a_{42} * t^2 + a_{43} * t + a_{44}$$

- Está na forma de:

$$Q(s,t) = S \cdot A \cdot T^T \quad , \quad 0 \le s \le 1 \text{ e } 0 \le t \le 1$$

$$S = [\, s^3 \;\; s^2 \;\; s \;\; 1 \,] \qquad T = [\, t^3 \;\; t^2 \;\; t \;\; 1 \,]$$

- <u>Forma de Hermite</u>

  - Curva de Hermite:

$$Q(s) = S \cdot M_H \cdot G_H$$

  - Superfície:

$$Q(s,t) = S \cdot M_H \cdot G_H(t) = S \cdot M_H \cdot \begin{bmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{bmatrix}$$

  - Sejam elementos de $G_H(t)$ curvas de Hermite:

$$P_1(t) = T \cdot M_H \cdot \begin{bmatrix} g_{11} \\ g_{12} \\ g_{13} \\ g_{14} \end{bmatrix} \qquad P_4(t) = T \cdot M_H \cdot \begin{bmatrix} g_{21} \\ g_{22} \\ g_{23} \\ g_{24} \end{bmatrix}$$

$$R_1(t) = T \cdot M_H \cdot \begin{bmatrix} g_{31} \\ g_{32} \\ g_{33} \\ g_{34} \end{bmatrix} \qquad R_4(t) = T \cdot M_H \cdot \begin{bmatrix} g_{41} \\ g_{42} \\ g_{43} \\ g_{44} \end{bmatrix}$$

$$\begin{bmatrix} P_1(t) & P_4(t) & R_1(t) & R_4(t) \end{bmatrix} = T \cdot M_H \cdot \begin{bmatrix} g_{11} & g_{21} & g_{31} & g_{41} \\ g_{12} & g_{22} & g_{32} & g_{42} \\ g_{13} & g_{23} & g_{33} & g_{43} \\ g_{14} & g_{24} & g_{34} & g_{44} \end{bmatrix}$$

- Por transposição:

$$G_H(t) = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \cdot M_H^T \cdot T^T = \underline{G}_H \cdot M_H^T \cdot T^T$$

- Por substituição em $Q(s,t) = S.M_H.G_H(t)$: (superfície de Hermite)

$$Q(s,t) = S \cdot M_H \cdot \underline{G}_H \cdot M_H^T \cdot T^T$$

$$\underline{G}_H = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix}$$

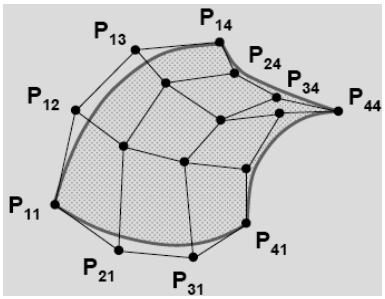
- Interpretação da matriz de geometria de Hermite G_H:

$$\underline{G}_H = \begin{bmatrix} Q(0,0) & Q(0,1) & \partial Q(s,t)/\partial t\,|_{s=0,t=0} & \partial Q(s,t)/\partial t\,|_{s=0,t=1} \\ Q(1,0) & Q(1,1) & \partial Q(s,t)/\partial t\,|_{s=1,t=0} & \partial Q(s,t)/\partial t\,|_{s=1,t=1} \\ \partial Q(s,t)/\partial s\,|_{s=0,t=0} & \partial Q(s,t)/\partial s\,|_{s=0,t=1} & \partial^2 Q(s,t)/\partial s\partial t\,|_{s=0,t=0} & \partial^2 Q(s,t)/\partial s\partial t\,|_{s=0,t=1} \\ \partial Q(s,t)/\partial s\,|_{s=1,t=0} & \partial Q(s,t)/\partial s\,|_{s=1,t=1} & \partial^2 Q(s,t)/\partial s\partial t\,|_{s=1,t=0} & \partial^2 Q(s,t)/\partial s\partial t\,|_{s=1,t=1} \end{bmatrix}$$

(As 1[as] derivadas são os vectores tangente e as segundas denominam-se *twist vectors*)

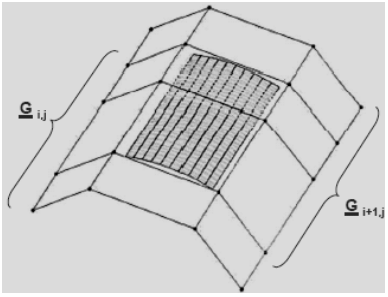
- <u>Forma de Bézier</u>

$$Q(s,t) = S \cdot M_B \cdot \underline{G} \cdot M_B^T \cdot T^T$$



- Forma de B-spline

$$Q(s,t) = S \cdot M_{Bs} \cdot \underline{G} \cdot M_{Bs}^{T} \cdot T^{T}$$



- Garante-se C^2 pela utilização dos pontos de controlo via matriz de geometria $G_{i,j}$ à semelhança das curvas B-spline com o vector de geometria G_i

- Transformações geométricas de curvas e superfícies

- Aplicam-se aos coeficientes dos vectores ou das matrizes de geometria

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \qquad V = \begin{bmatrix} V_x \\ V_y \\ V_z \\ 0 \end{bmatrix}$$

(S,R,T) (S,R)

- Plano tangente a uma superfície Bicúbica

- Vector tangente segundo s:

$$\frac{\partial}{\partial s} Q(s,t) = \frac{\partial}{\partial s} (S \cdot M \cdot \underline{G} \cdot M^{T} \cdot T^{T})$$
$$= [\ 3s^2 \ \ 2s \ \ 1 \ \ 0\ ] \cdot M \cdot \underline{G} \cdot M^{T} \cdot T^{T}$$

- Vector tangente segundo t:

$$\frac{\partial}{\partial t} Q(s,t) = \frac{\partial}{\partial t} (S \cdot M \cdot \underline{G} \cdot M^{T} \cdot T^{T})$$

$$= S \cdot M \cdot \underline{G} \cdot M^{T} \cdot \begin{bmatrix} 3t^2 \\ 2t \\ 1 \\ 0 \end{bmatrix}$$

- Vector normal a uma superfície Bicúbica

  - Vector normal:

$$\frac{\partial}{\partial s} Q(s,t) \times \frac{\partial}{\partial t} Q(s,t) = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ x_s & y_s & z_s \\ x_t & y_t & z_t \end{vmatrix}$$

Recorte

=> **Recorte (Clipping) por janelas rectangulares**

- Pontos:

  - P(x,y) é visível se não for exterior à janela,

$$x \leq x_{max} \wedge x \geq x_{min} \wedge y \leq y_{max} \wedge y \geq y_{min}$$

- Linhas (segmentos de recta)

  - PQ é visível de P for visível e Q for visível

1. Método da força bruta

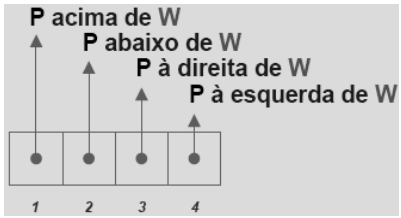
- Teste de paralelismo
- Resolução de um sistema de duas equações
- O ponto de intersecção das rectas pertence aos segmentos?

2. Algoritmo de Cohen-Sutherland

- Baseia-se na definição de regiões de teste com relação à janela W e atribuição de um código binário a cada extremidade de uma linha:

1001	1000	1010
0001	0000	0010
0101	0100	0110

- Convenção para cada ponto P: $bit_i = 1$ se



- PQ é trivialmente aceite se: (código(P) or código(Q)) = 0000

- PQ é trivialmente rejeitado se: (código(P) and código(Q)) ≠ 0000

- Estratégia iterativa para os restantes casos, em que se procurará, então, pelo menos uma intersecção

- Para implementar essa estratégia iterativa, escolhe-se-á a seguinte ordem para efectuação dos testes

Bit 1 → Bit 2 → Bit 3 → Bit 4

, aplicando-se então as regras que decorrem da convenção usada:

Bits 1 diferentes → rejeitam-se as linhas acima de W e recomeça-se
Bits 2 diferentes → rejeitam-se as linhas abaixo de W e recomeça-se
Bits 3 diferentes → rejeitam-se as linhas à direita de W e recomeça-se
Bits 4 diferentes → rejeitam-se as linhas à esquerda de W e recomeça-se

- Método de resolver a intersecção:

a) Resolução de um sistema juntando as equações:

$$x = x_{max} \quad x = x_{min} \quad y = y_{max} \quad y = y_{min}$$

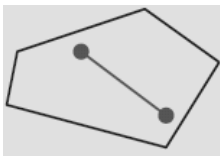
b) Substituição pelo ponto médio

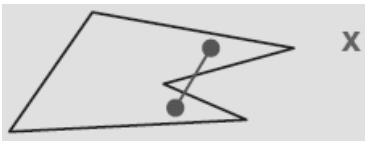
$$x_{med} = (x_P + x_Q)/2 \qquad y_{med} = (y_P + y_Q)/2$$

- Aplicado iterativamente, este algoritmo de pesquisa dicotómica necessita, no máximo, de $\log_2 M_x$ subdivisões com M_x = número máximo de pixels de uma linha

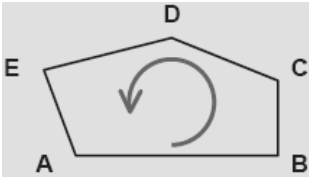
=> **Recorte (Clipping) por janelas poligonais convexas**

- Polígono convexo: Aquele em que uma linha unindo dois quaisquer pontos interiores ao polígono esteja nele totalmente contida

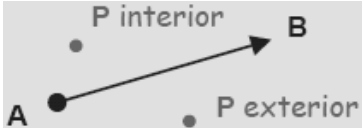




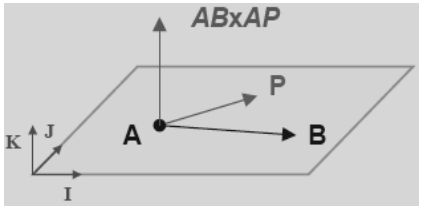
- Polígono (convexo) com orientação positiva:



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



- Matematicamente:

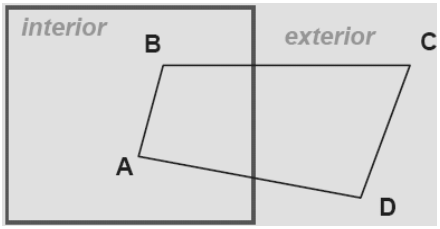


$$ABxAP = \begin{vmatrix} I & J & K \\ (x_B-x_A) & (y_B-y_A) & 0 \\ (x_P-x_A) & (y_P-y_A) & 0 \end{vmatrix} = ((x_B-x_A)(y_P-y_A)-(x_P-x_A)(y_B-y_A))\,K = k\,K$$

K > 0 ⇒ P à esquerda de AB
K < 0 ⇒ P à direita de AB

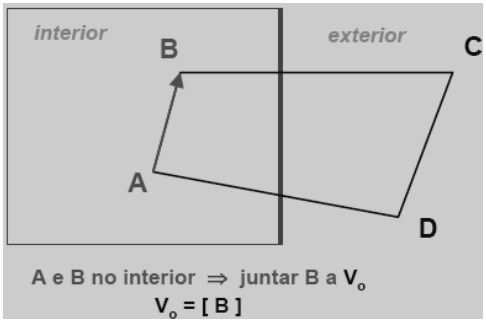
=> **Recorte (Clipping) de Polígonos por janelas rectangulares**

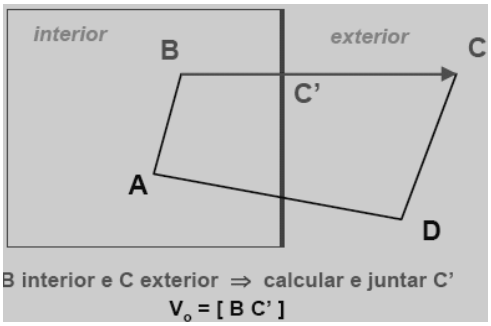
1. Algoritmo de Sutherlan-Hodgman

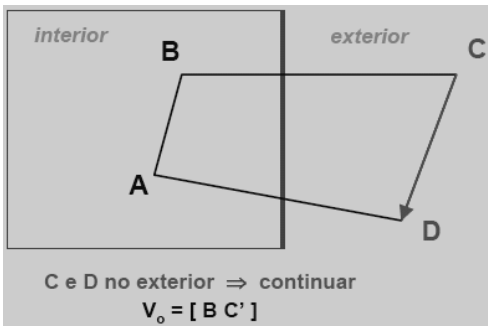


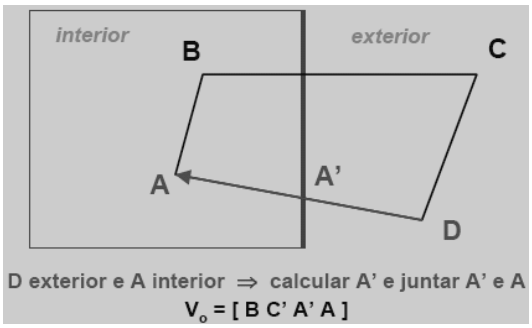
V_i = Lista de vértices de entrada = [A B C D]

V_o = Lista de vértices de saída = nil

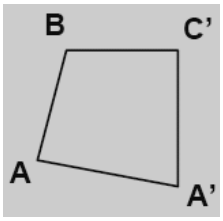




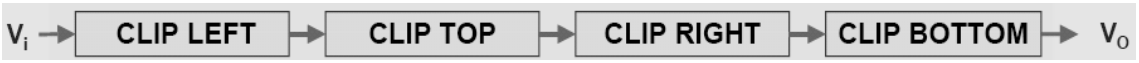




- Polígono após o recorte:



- *Clipping stages*:



(nota: esta ordem é arbitrária)

=> **Recorte (Clipping) de Polígonos**

1. Algoritmo de Weiler-Atherton

- Não gera arestas estranhas em polígonos côncavos, quer teriam de ser eliminadas em pós-processamento no caso do algoritmo de Sutherland-Hodgman.

- Principais etapas do algoritmo:
  1. Começa-se a percorrer o polígono em sentido retrógrado
  2. Se for encontrada uma intersecção com a janela e:
     2.1- a aresta em percurso estiver a entrar na janela:
     - Memoriza-se a intersecção (inclui aresta) e continua-se normalmente o percurso

2.2- a aresta em percurso estiver a sair na janela:

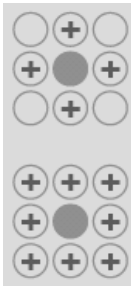
- Memoriza-se a intersecção (inclui aresta) e o percurso é desviado para a direita, continuando pelas arestas da janela até à intersecção seguinte e retomando o polígono a recortar.

Recorte

- Grupo de pixels adjacentes e com valor igual

- Tipos:

  - com ligação a 4
  - com ligação a 8



- Classificação pela definição:
  > definida pelo interior → algoritmos "flood-fill"
  > definida pela fronteira → algoritmos "boundary-fill"

1. Algoritmo iterativo

   1. new(pilha)
   2. Percorrer a linha corrente, partindo do pixel inicial até encontrar o pixel mais à direita, empilhando-o
   3. pixel_corrente := pop(pilha)

   Preencher a carreira(span = pixels adjacentes horizontalmente) do pixel corrente, caso ainda não tenho sido tratada

4. Examinar a linha imediatamente acima da carreira corrente (da direita para a esquerda), caso inda não tenha sido analisada, para encontrar todas as carreiras acessíveis que a constituem

5. Para cada carreira, empilhar o endereço do pixel mais à direita

6. Repetir 4. e 5. para a linha imediatamente abaixo da carreira corrente, se ainda estiver por tratar

7. Se a pilha não estiver vazia, recomeçar em 3), senão FIM