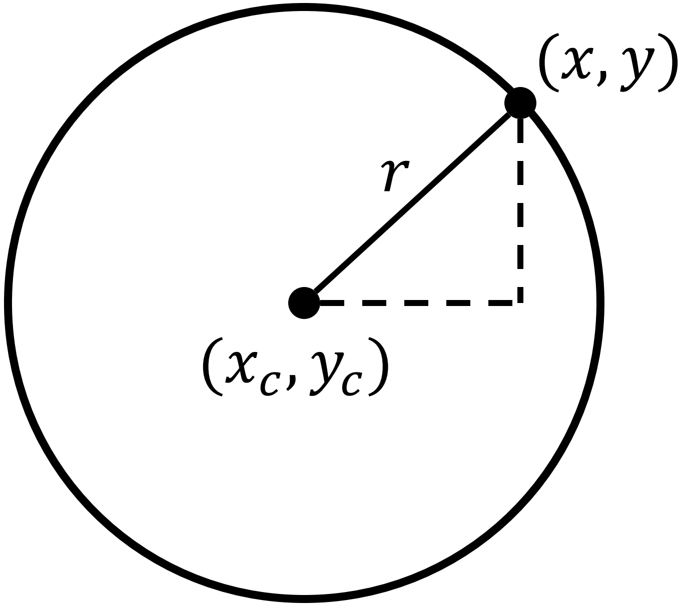
CÍRCULOS, ELIPSES E CURVAS

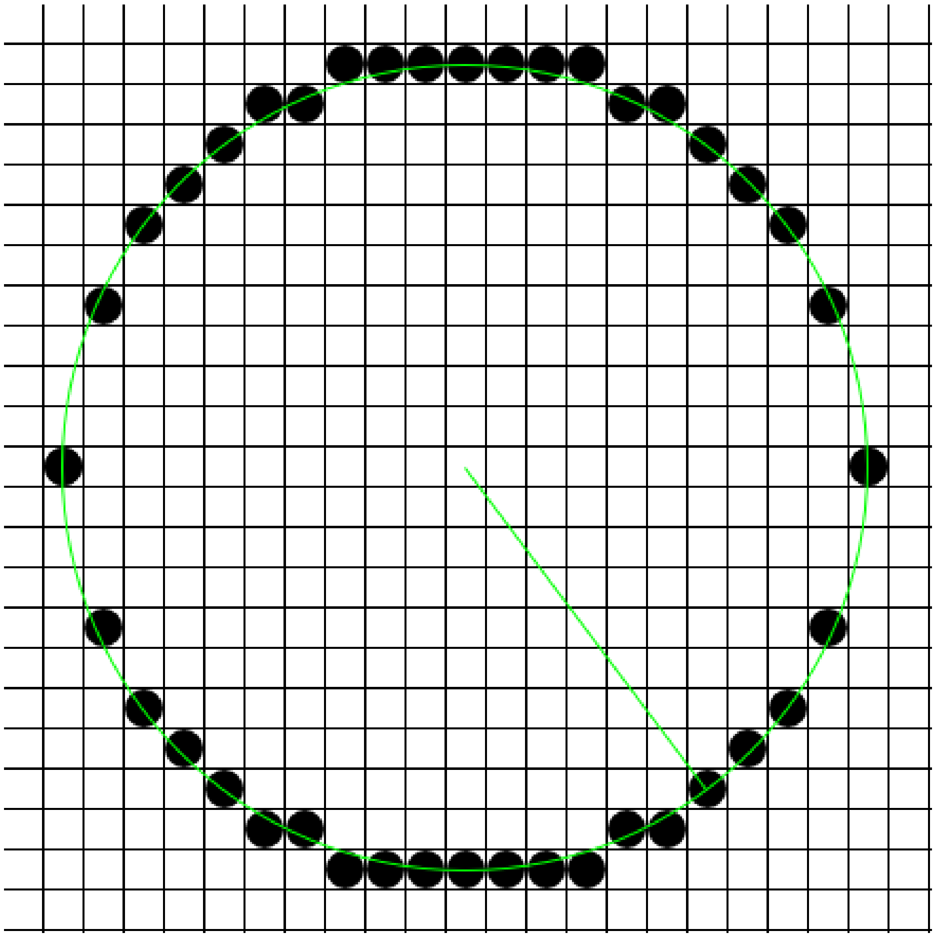
Prof. Dr. Bianchi Serique Meiguins
Prof. Dr. Carlos Gustavo Resque dos Santos

Círculo

- Matemáticamente:
  - $(x - x_c)^2 + (y - y_c)^2 = r^2$



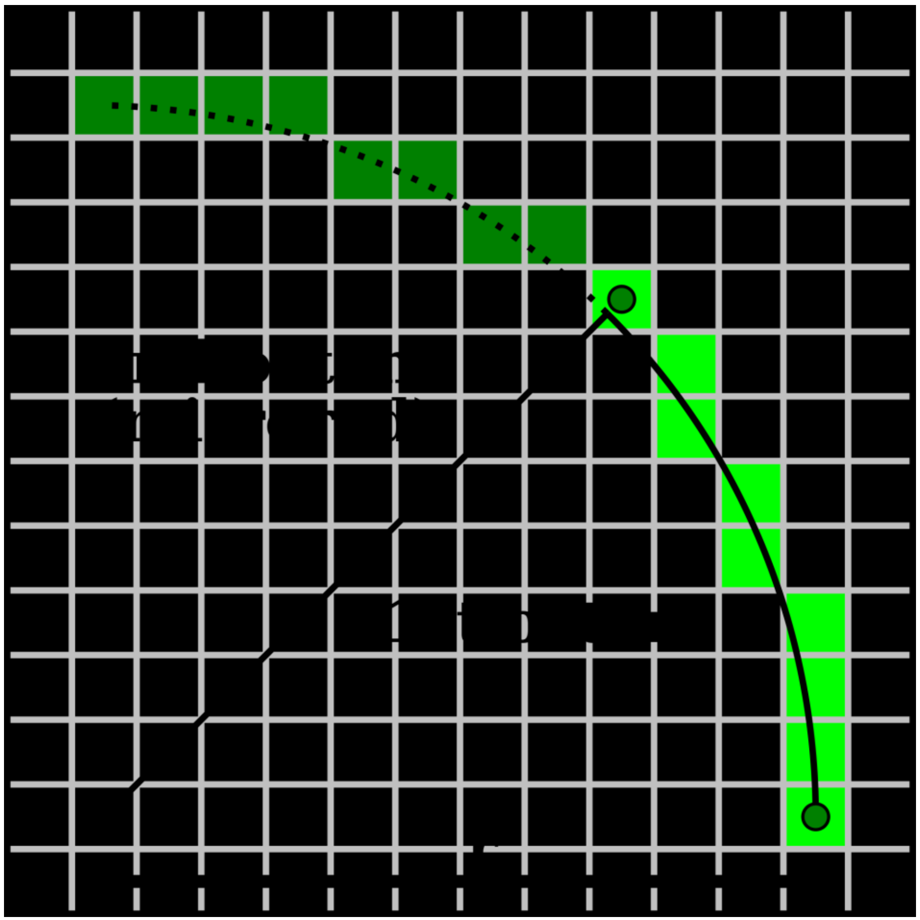
Problema do Passo

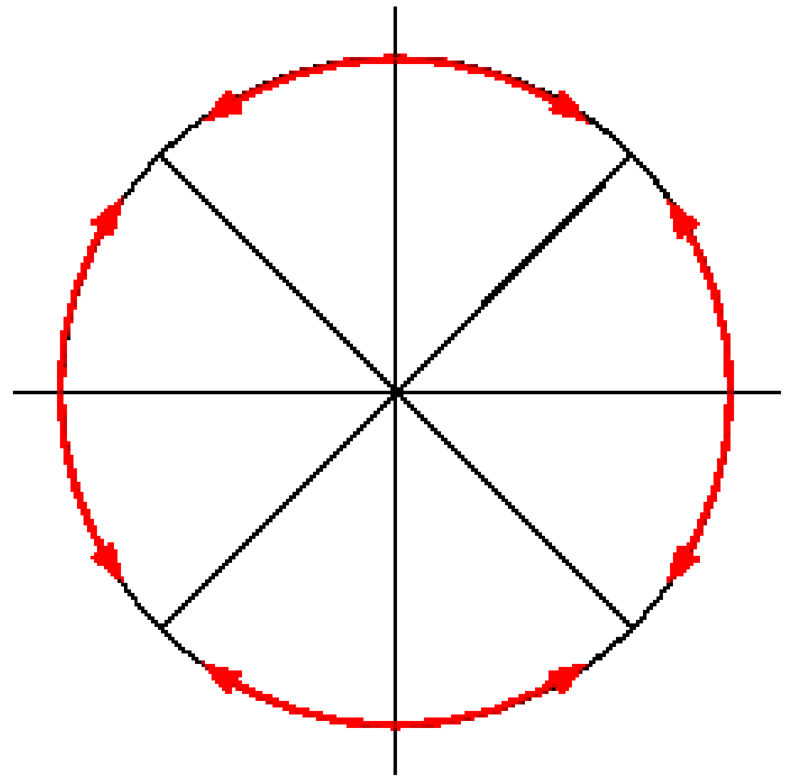


- Utilizando
  - $x_{n+1} = x_n + 1$
- E a fórmula
  - $y = y_c + \sqrt{r^2 - (x - x_c)^2}$
- Utilizando como ponto inicial
  - $(0, r)$

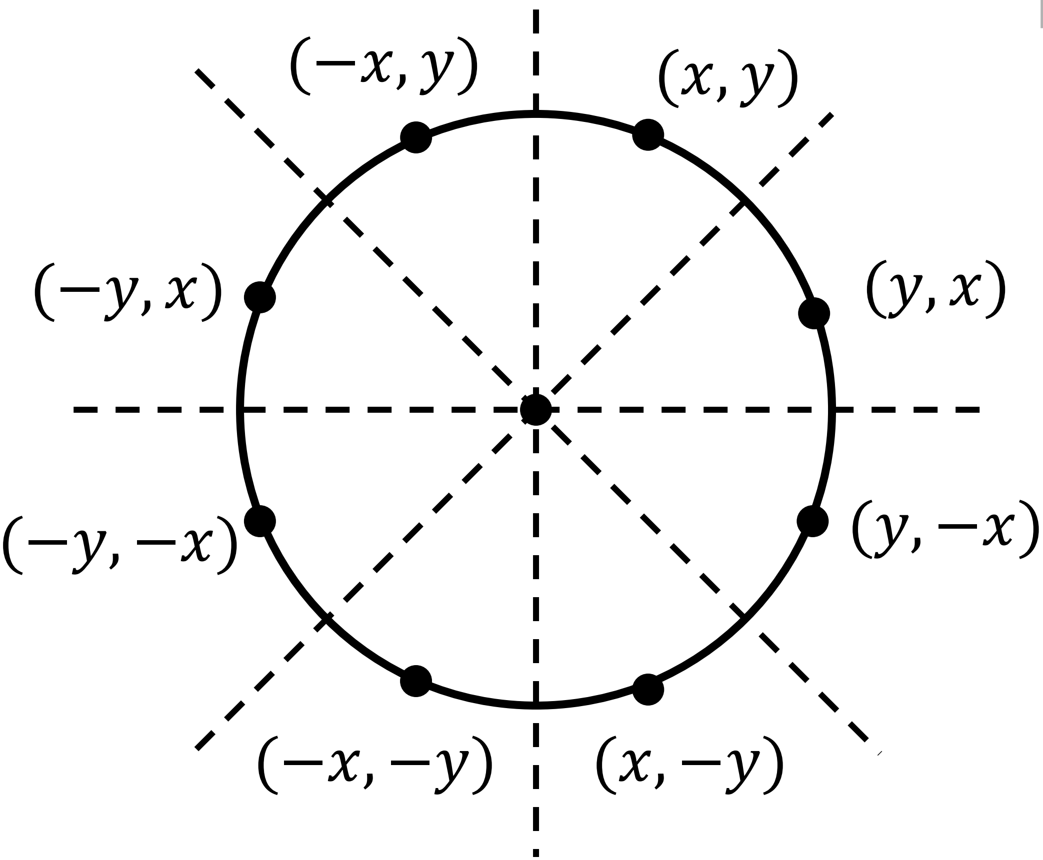
Problema do Passo

- Usando octantes





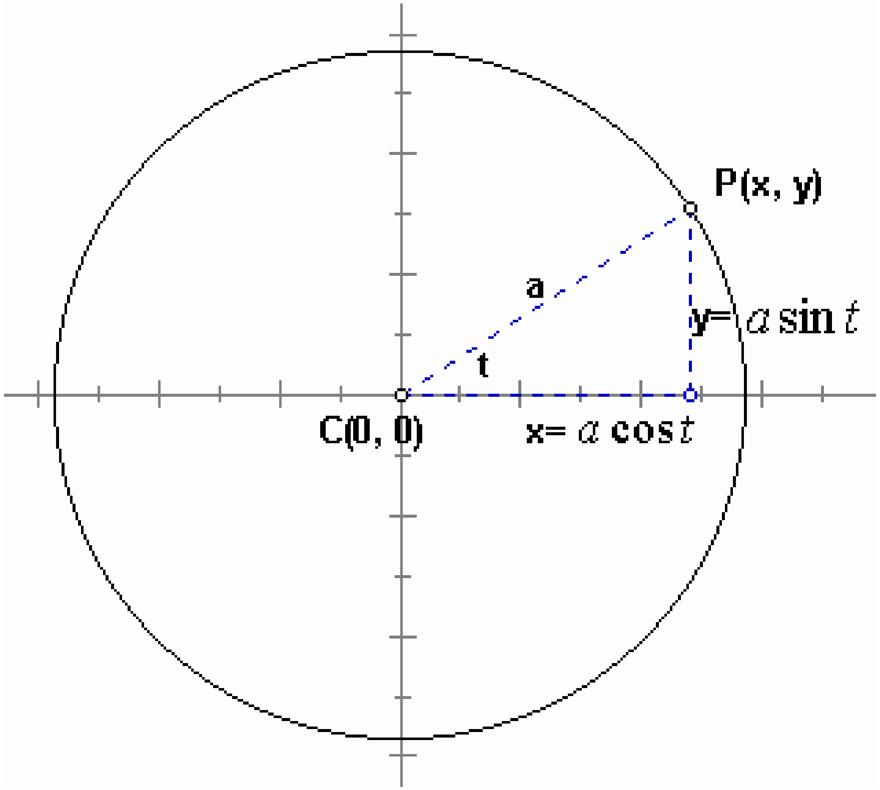
Simetria do Círculo



- Reflexão de pontos em octantes:

[redacted]	[redacted]
$-x$	y
[redacted]	[redacted]
$-x$	$-y$
[redacted]	[redacted]
$-y$	x
[redacted]	[redacted]
$-y$	$-x$

Usando Coordenadas Polares



- $x = r \times \sin(\sigma)$
- $y = r \times \cos(\sigma)$

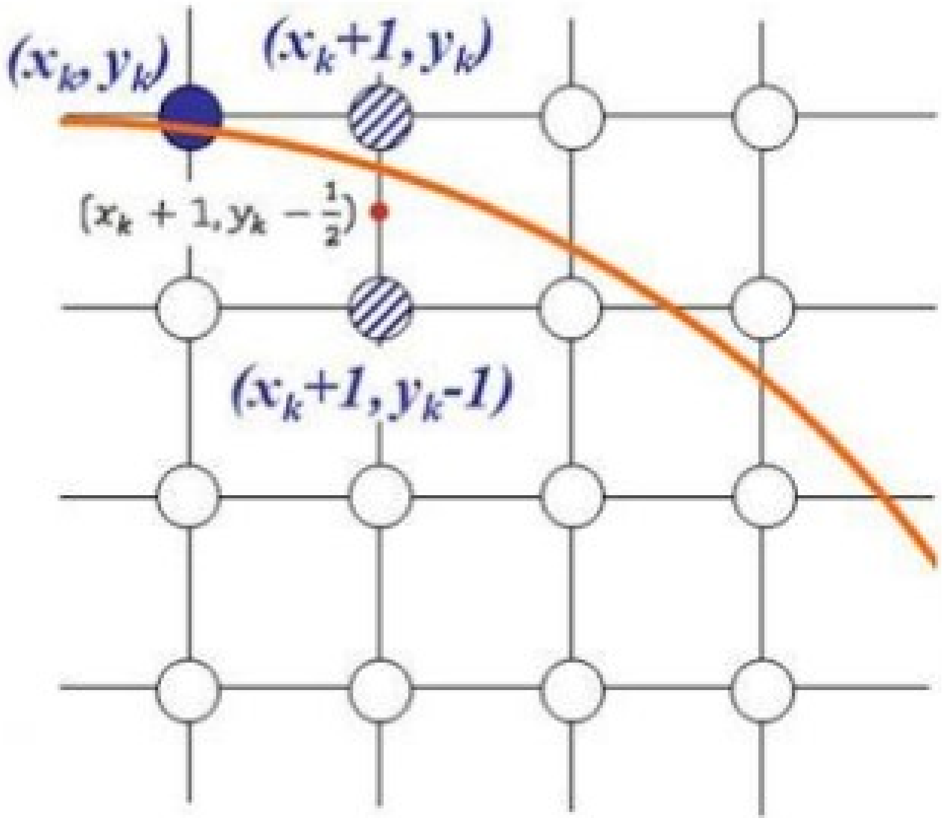
- $\sigma \mathrel{+}= step$

Problemas

- Usando a fórmula do círculo
  - Potenciação
  - Raiz Quadrada

- Usando Coordenadas Polares
  - Encontrar passo para o ângulo σ
  - Utilização de seno e cosseno na fórmula

Algoritmo do Ponto Médio



- Reflexão de octantes
- Ponto inicial $(0, r)$
- Calcula o erro recursivamente
- Decide se:
  - $y_{i+1} = y_i - 1$
- Ou
  - $y_{i+1} = y_i$

Algoritmo do Ponto Médio

Recebe: raio e centro

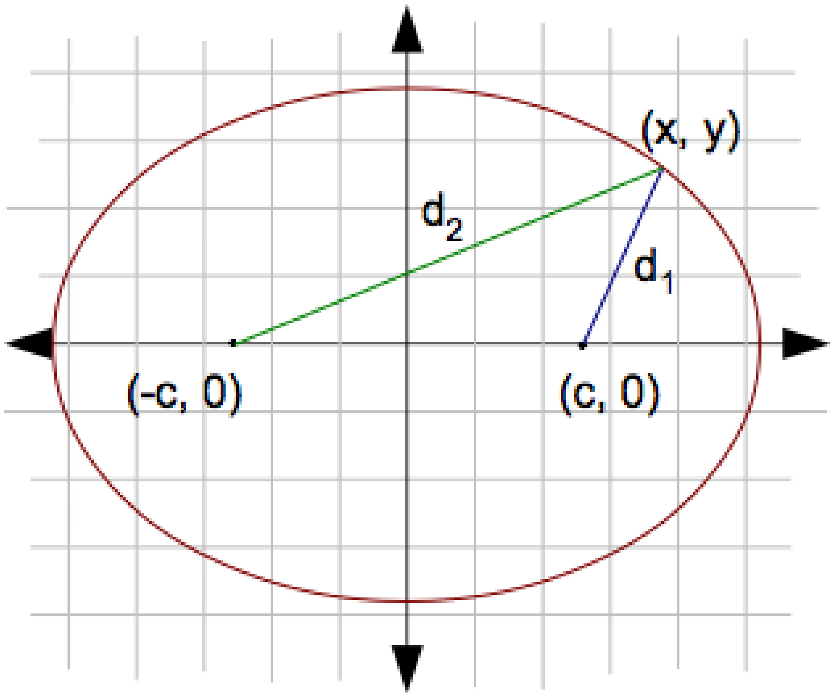
Desloca o centro para (0,0)

1. x = 0
2. y = r
3. p = 1 - r

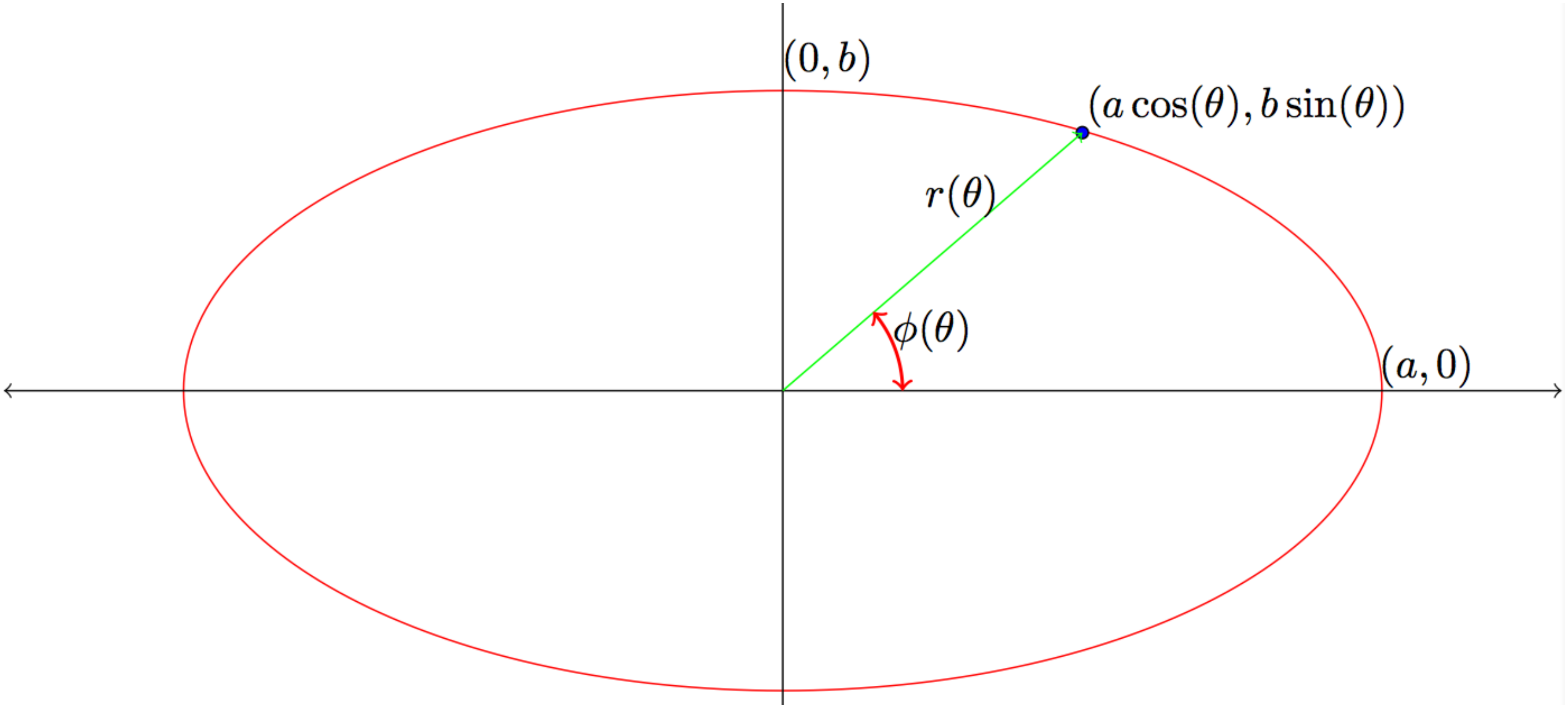
1. desenhaPonto(x,y)
2. Enquanto(x<y)
   1. x++;
   2. Se(p<0)
      1. p+=2*x + 3
   3. Senão
      1. y--;
      2. p+=2*x – 2*y + 5
   4. desenhaPonto(x,y)

Elipse

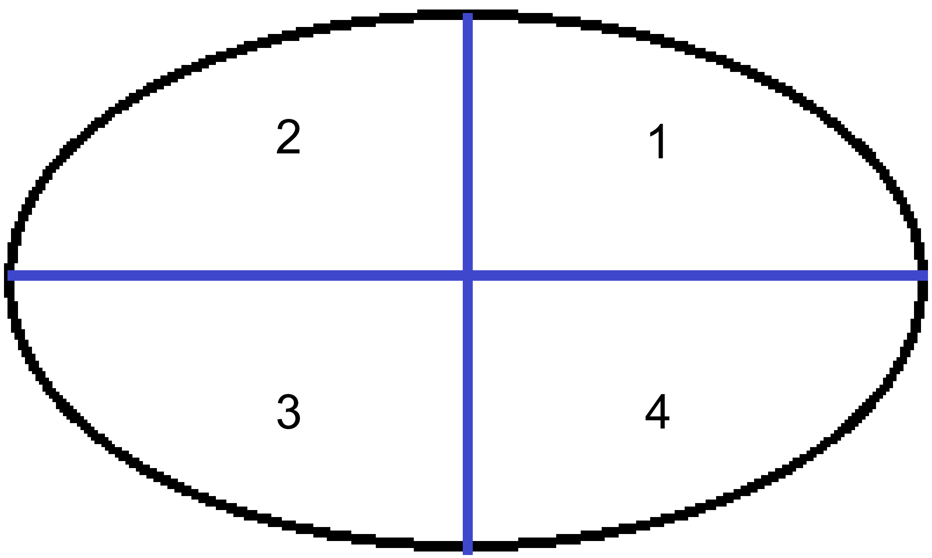
- $d_1 + d_2 = \sqrt{(x - x_c)^2 + (y - y_c)^2} + \sqrt{(x - x_{-c})^2 + (y - y_{-c})^2}$



Em coordenadas Polares



Simetria na Elipse

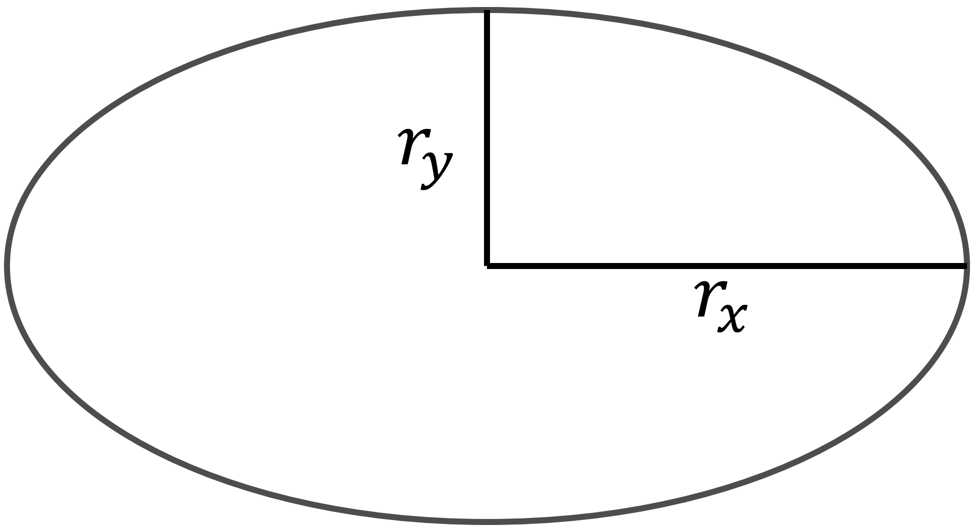


[redacted]		
1	x	y
[redacted]		
3	x	$-y$
[redacted]		

Algoritmo do Ponto Médio (Elipse)

- Armazena ${r_x}^2$ e ${r_y}^2$ antes no loop

- Utiliza apenas soma de inteiros e multiplicação por 2 dentro do loop



Curvas Bézier

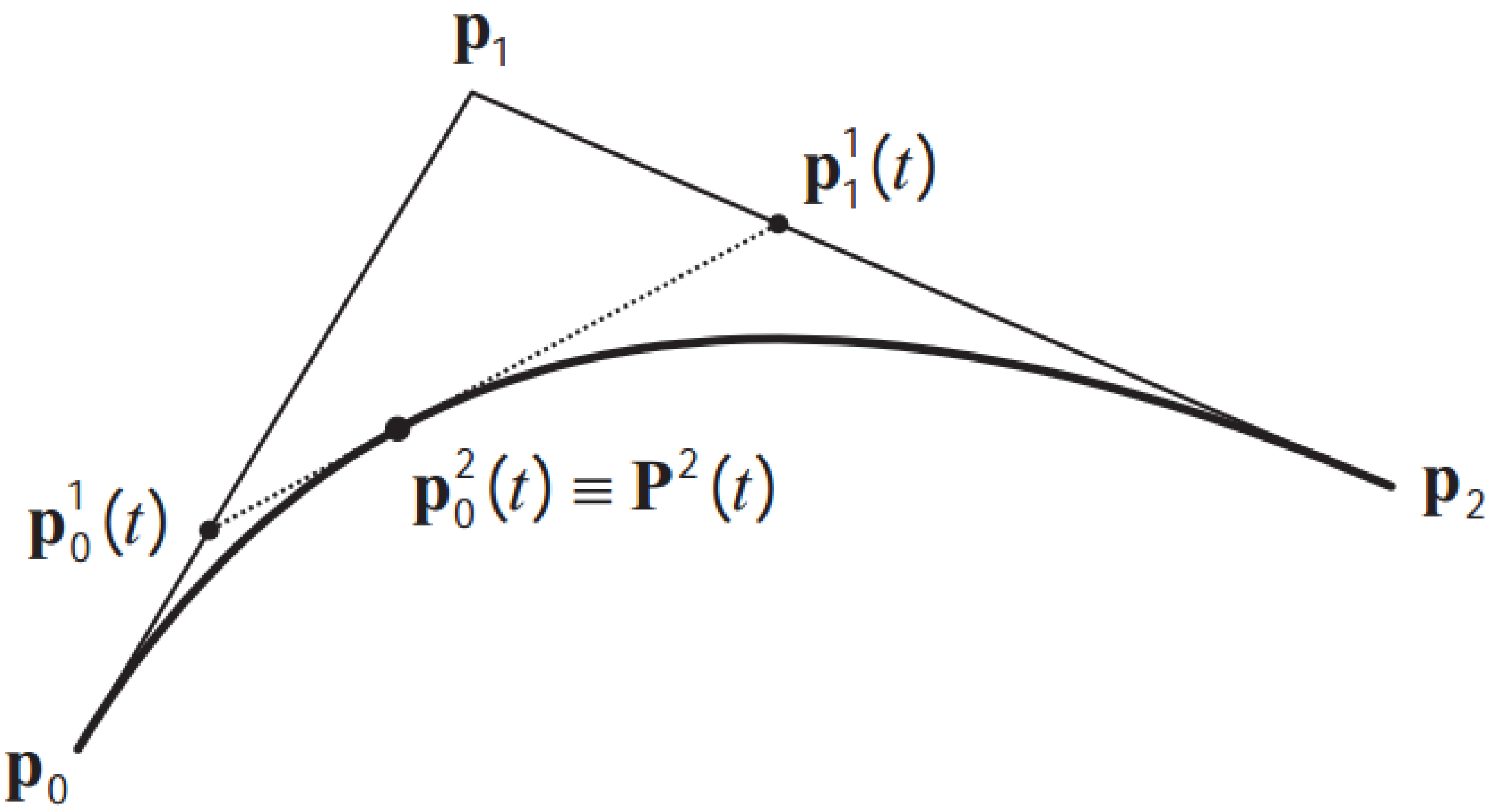
- Baseia-se na interpolação de pontos de controle.

- As interpolações são parametrizadas por um único parâmetro t.

- As interpolações são do tipo:

$$\mathbf{P}(t) = (1-t)\,\mathbf{p}_0 + t\,\mathbf{p}_1, \quad t \in [0,1],$$

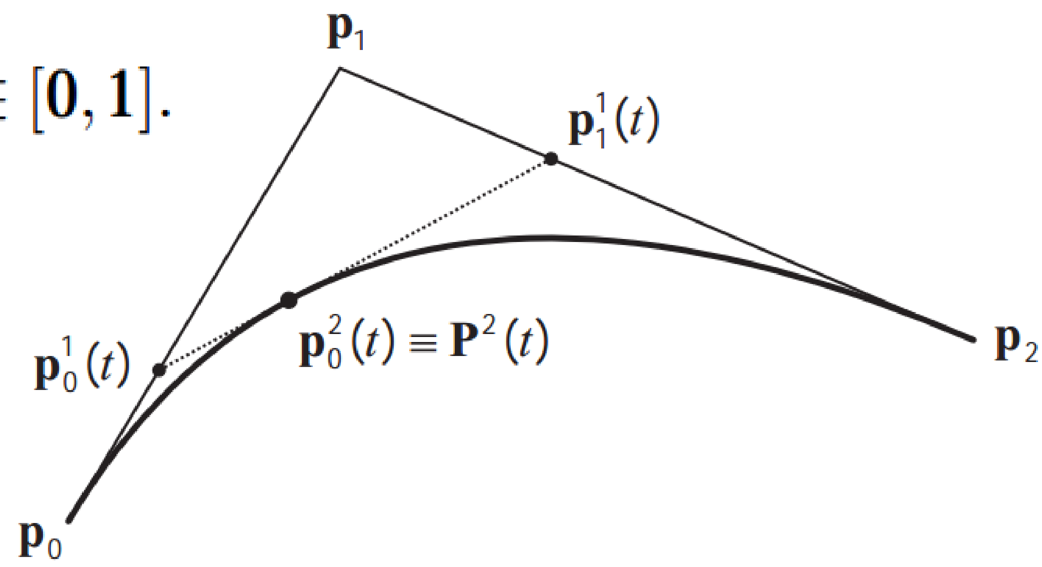
Curvas Bézier



Curvas Bézier Quadráticas

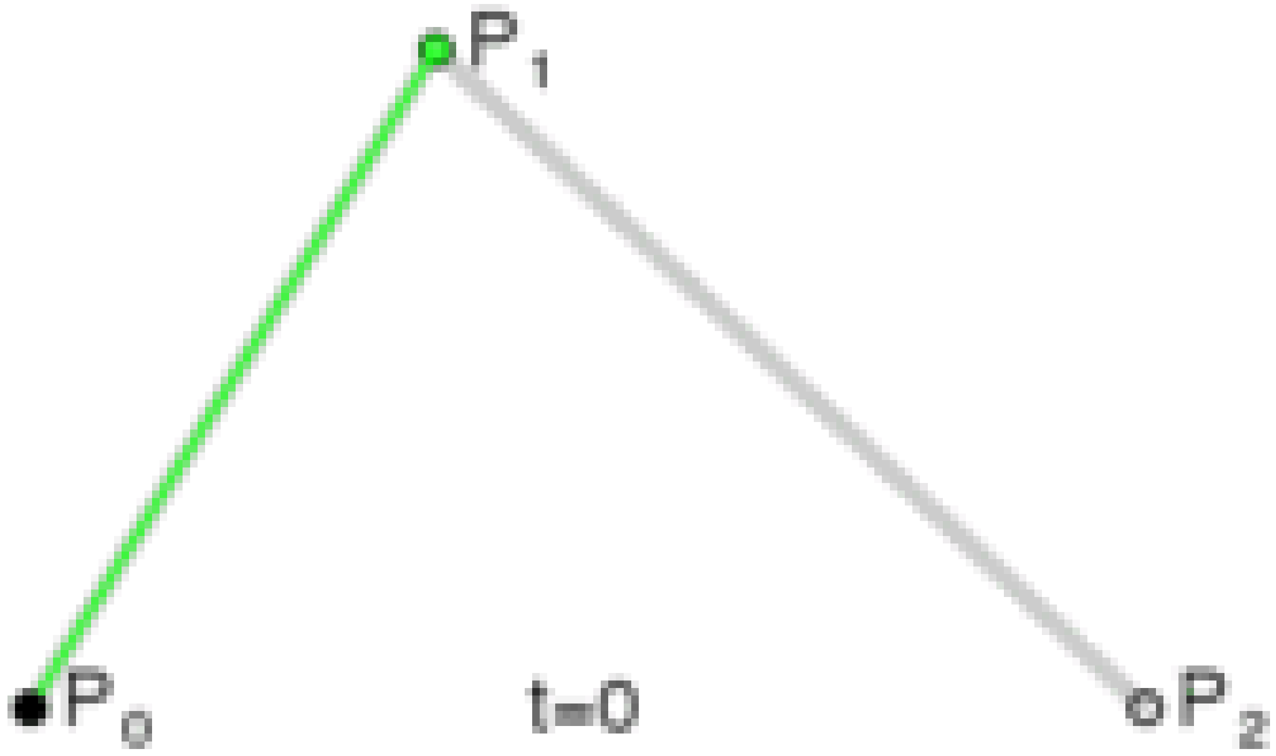
$$\mathbf{p}_0^1(t) = (1-t)\,\mathbf{p}_0 + t\,\mathbf{p}_1,$$
$$\mathbf{p}_1^1(t) = (1-t)\,\mathbf{p}_1 + t\,\mathbf{p}_2,$$

$$t \in [\mathbf{0}, \mathbf{1}].$$



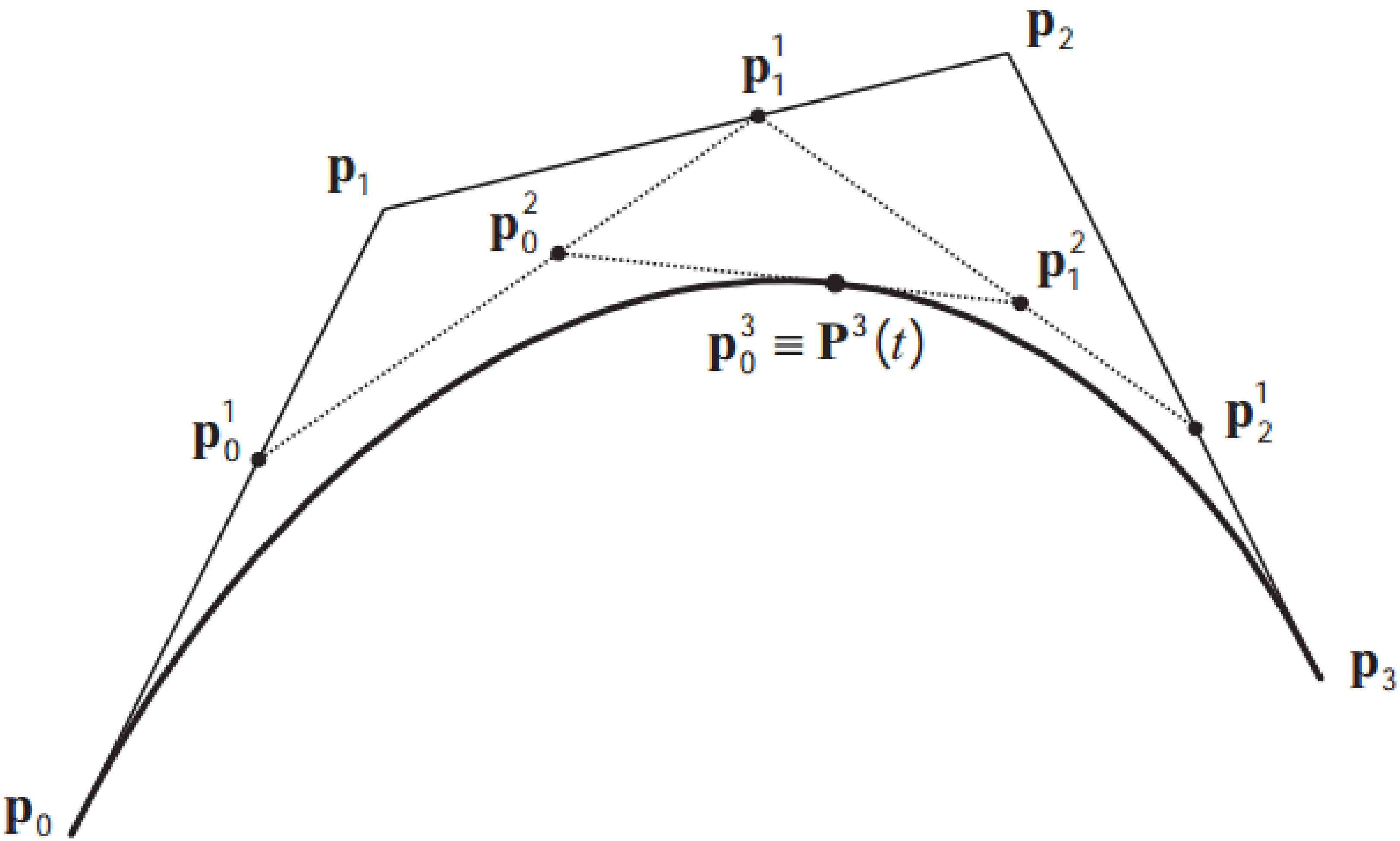
$$\mathbf{p}_0^2(t) = (1-t)\,\mathbf{p}_0^1(t) + t\,\mathbf{p}_1^1(t)$$
$$= (1-t)^2\,\mathbf{p}_0 + 2t(1-t)\,\mathbf{p}_1 + t^2\,\mathbf{p}_2.$$

Variando t temos

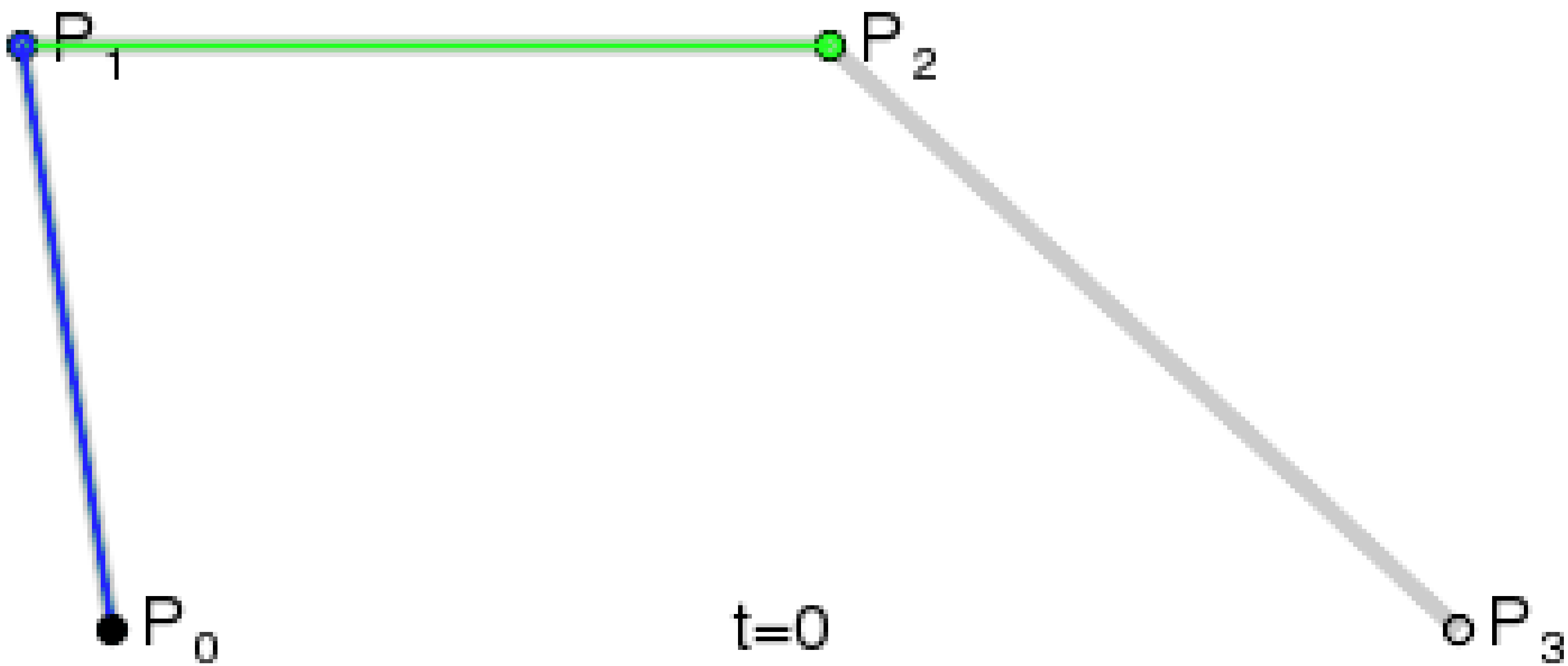


https://pt.wikipedia.org/wiki/Curva_de_B%C3%A9zier#/media/File:B%C3%A9zier_2_big.gif

Bézier Cúbico



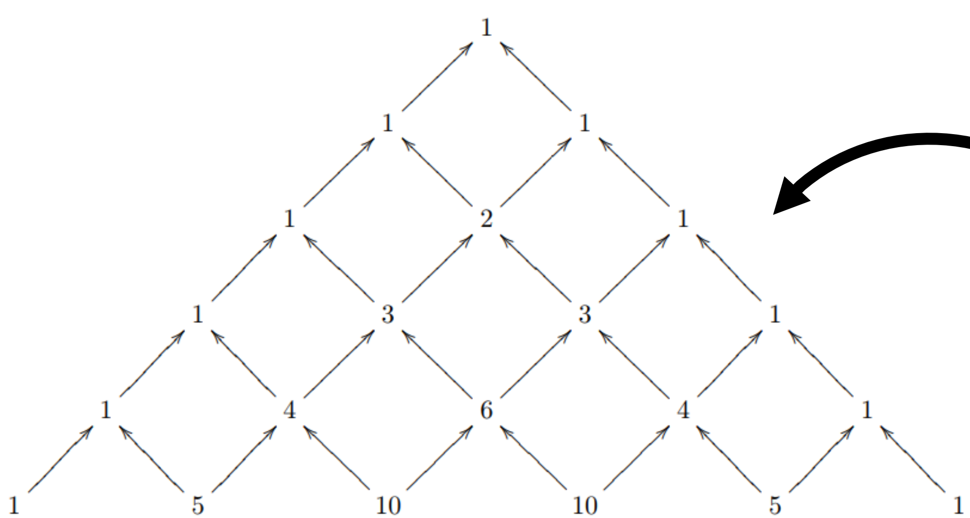
Bézier Cúbico



https://pt.wikipedia.org/wiki/Curva_de_B%C3%A9zier#/media/File:B%C3%A9zier_3_big.gif

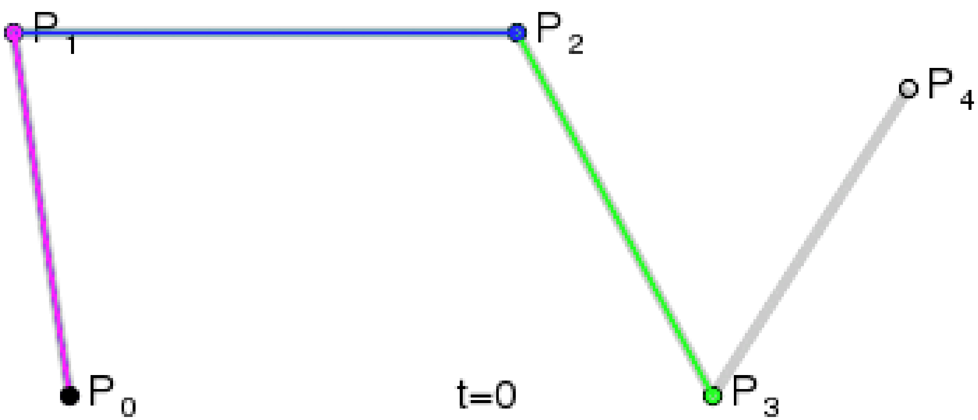
Bézier de Grau n

$$\mathbf{P}^n(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \mathbf{p}_i, \qquad t \in [0,1].$$

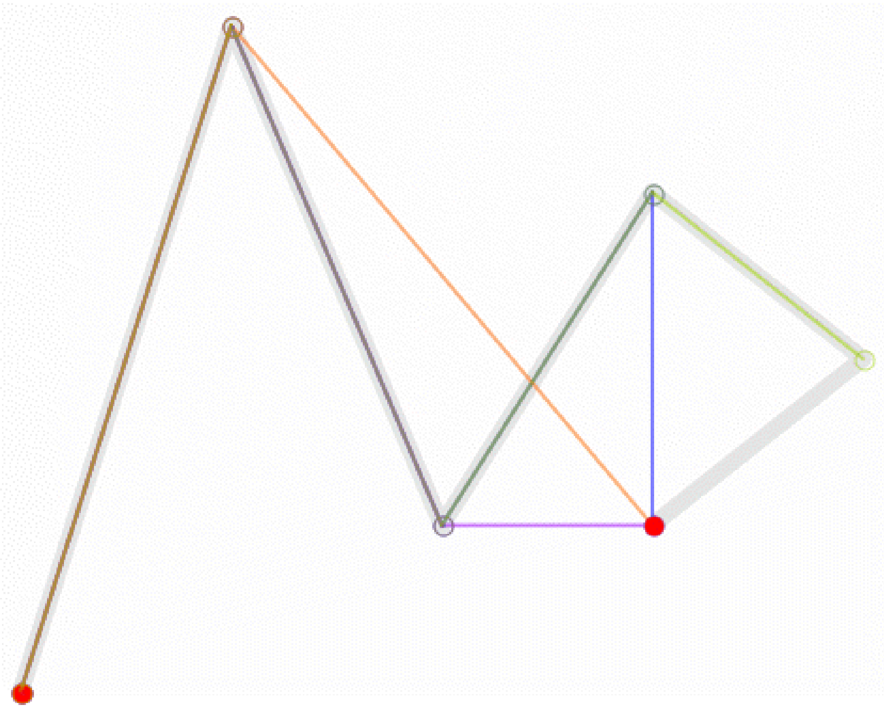


$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!}$$

Bézier de Grau n



https://upload.wikimedia.org/wikipedia/commons/a/a4/B%C3%A9zier_4_big.gif



https://upload.wikimedia.org/wikipedia/commons/0/0b/BezierCurve.gif

Algoritmo de De Casteljau

1. For the required value of t, set

$$\mathbf{p}_i^0(t) = \mathbf{p}_i, \quad i = 0, 1, \ldots, n.$$

2. Perform the linear interpolation steps

$$\mathbf{p}_i^r(t) = (1-t)\,\mathbf{p}_i^{r-1}(t) + t\,\mathbf{p}_{i+1}^{r-1}(t), \quad \begin{array}{l} r = 1, 2, \ldots, n, \\ i = 0, 1, \ldots, n-r. \end{array}$$

3. Then the point on the curve corresponding to parametric value t is

$$\mathbf{P}^n(t) = \mathbf{p}_0^n(t).$$

Algoritmo de De Casteljau

$$
\begin{tikzcd}
\mathbf{p}_0 = \mathbf{p}_0^0 \arrow[r, "1-t"] & \mathbf{p}_0^1 \arrow[r] & \mathbf{p}_0^2 \arrow[r] & \mathbf{p}_0^3 = \mathbf{P}^3(t) \\
\mathbf{p}_1 = \mathbf{p}_1^0 \arrow[ur, "t"'] \arrow[r] & \mathbf{p}_1^1 \arrow[ur] \arrow[r] & \mathbf{p}_1^2 \arrow[ur] & \\
\mathbf{p}_2 = \mathbf{p}_2^0 \arrow[ur] \arrow[r] & \mathbf{p}_2^1 \arrow[ur] & & \\
\mathbf{p}_3 = \mathbf{p}_3^0 \arrow[ur] & & &
\end{tikzcd}
$$

Algoritmo de De Casteljau

```
point bezierPoint(int n, point[] controlPt, float t){
      point pts[n+1];
      for (i=0; i <= n; i++)
            pts[i] = controlPt[i];

      for (r=1; r <= n; r++) {
            for (i=0; i <= n-r; i++) {
                  pts [i] = (1-t)*pts[i] + t*pts[i+1];
            }
      }
      return pts[0];
}
```