Grafica vettoriale

- Come abbiamo visto in un certo dettaglio, la grafica *raster* si basa sull'idea di definire certe proprietà (colore, trasparenza, ecc.) di **ogni** pixel
  - applicazioni di tipo fotografico
- Al contrario la **grafica vettoriale** si basa sull'idea di dare una **descrizione geometrica** dell'immagine
  - applicazioni di tipo tecnico o fumettistico

Grafica vettoriale

- L'elemento base della grafica raster sono i pixel
- L'elemento base della grafica vettoriale sono gli **elementi geometrici**:
  - linee
  - curve
  - aree
- Ogni elemento è caratterizzato da certe proprietà che ne definiscono l'aspetto

Grafica vettoriale

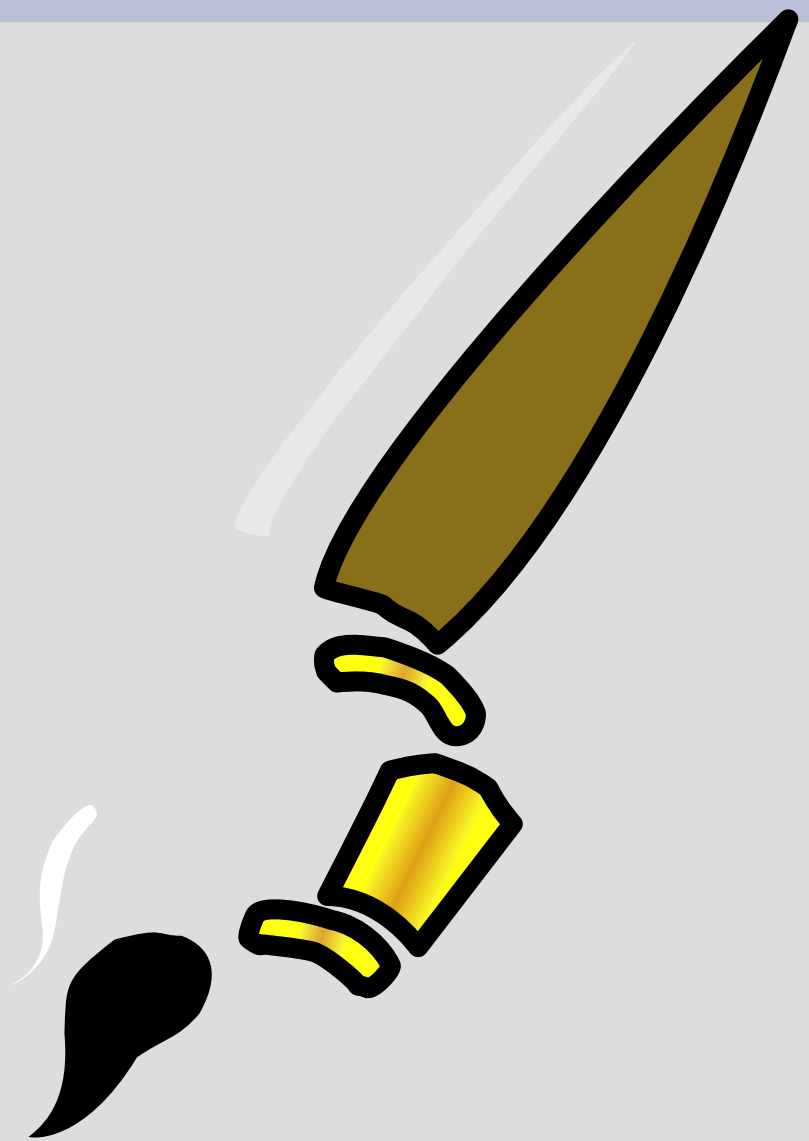
- Quali elementi siano disponibili per il disegno, e quali siano le loro proprietà specifiche, è definito da un **linguaggio**
  - ne esistono molti, noi discutiamo ora i concetti generali, più avanti vedremo i principali
- Un disegno in grafica vettoriale può essere visto come un insieme di istruzioni (*immagine*) rivolte a un disegnatore (*motore di rendering*) ed espresse in un dato linguaggio (*formato*)

Grafica vettoriale

- Un'immagine come quella qui accanto è codificata in termini di curve, aree, linee...
- Di ciascun elemento è data una **rappresentazione matematica** astratta (equazioni)

Grafica vettoriale

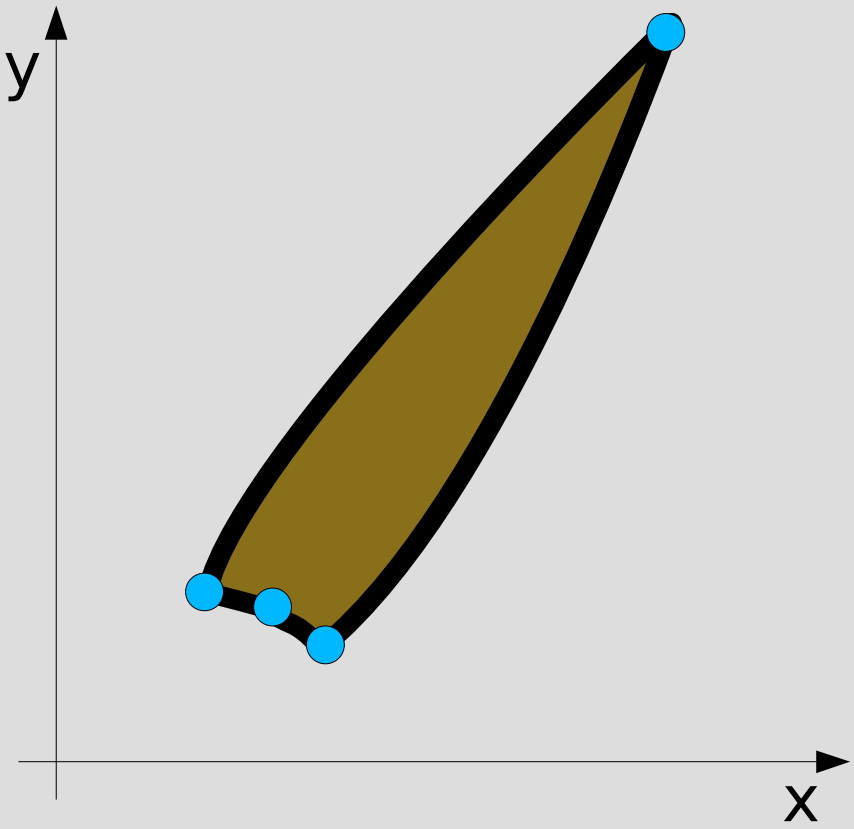
- Le varie parti che compongono l'immagine sono separabili
- Esempio:
  - varie curve chiuse
  - aree interne colorate
  - linee di confine (bordi) visibili o meno



Grafica vettoriale

- Le aree chiuse sono definite da:
  - la **forma** dell'area, data da un insieme di punti e da parametri che regolano la curvatura delle linee fra questi punti
  - il tipo di **linea** o **bordo**, dato da una forma del pennello, un colore, parametri che regolano in che modo disegnare gli angoli
  - il tipo di **riempimento**, per esempio un colore piano, un motivo, una sfumatura, e i relativi parametri

Elementi di base

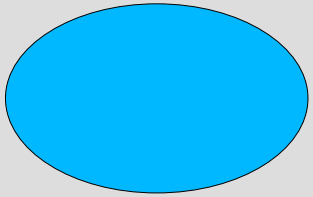


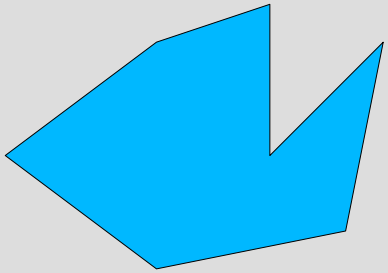
- I punti geometrici sono dati da coordinate in un piano cartesiano (x,y)
- I colori sono un punto nello spazio dei colori (r,g,b)*

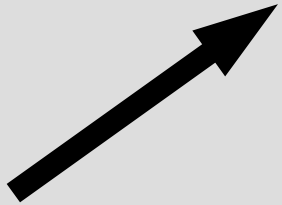
---

* sono naturalmente possibili altre codifiche dei colori, come già visto

Elementi di base



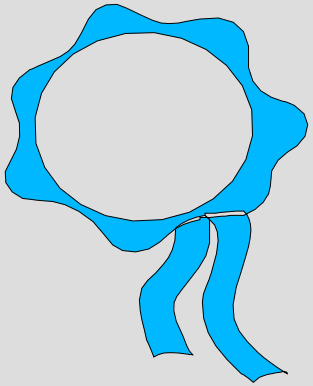


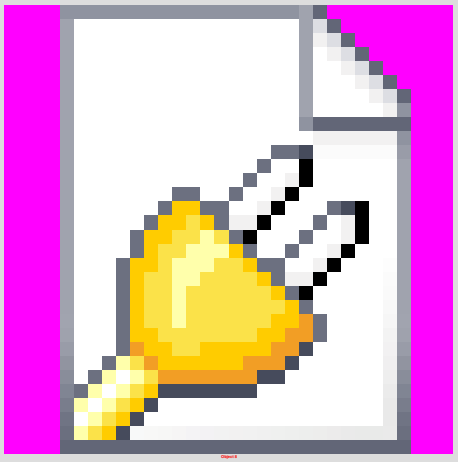


TESTO!

- Figure geometriche regolari
  - rettangoli, ellissi, ...
- Polilinee
  - aperte o chiuse
- Linee semplici
- Immagini raster
- Testo

Elementi di base

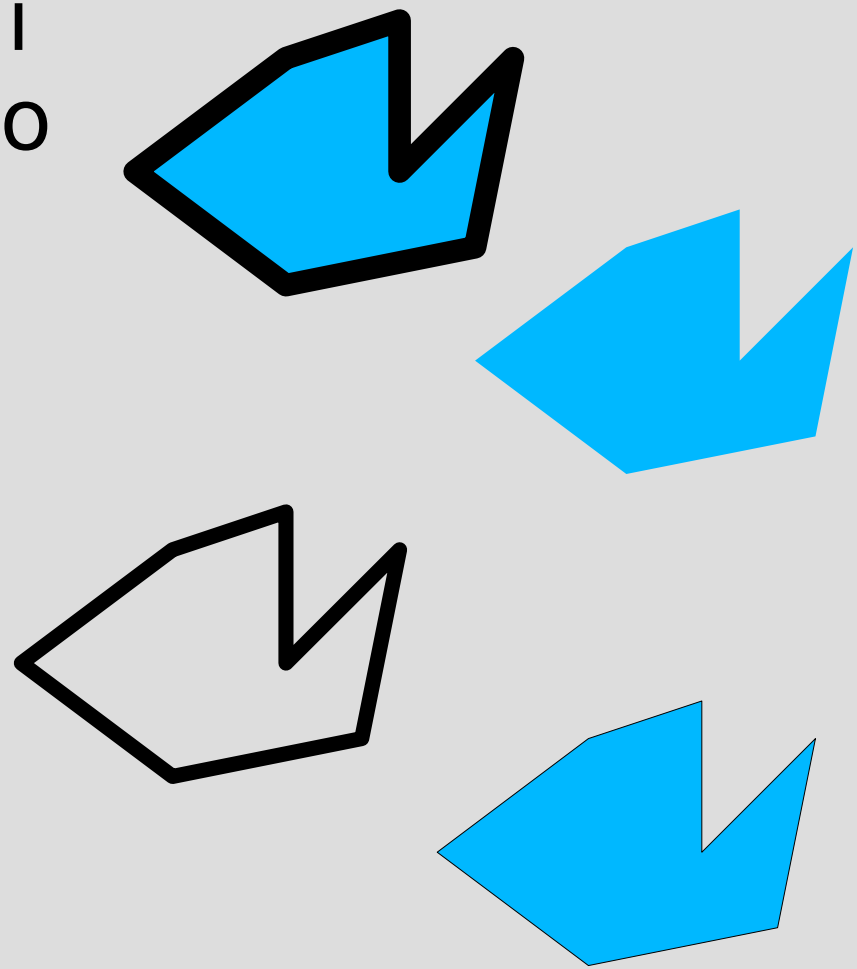




- In effetti, i formati vettoriali sono quasi **object-oriented**
- Alcuni formati possono anche avere elementi di base “strani”
  - applet Java
  - filmati, animazioni, audio
  - oggetti COM (Word, Excel,...)
  - oggetti grafici speciali (stelle, fumetti, ...)

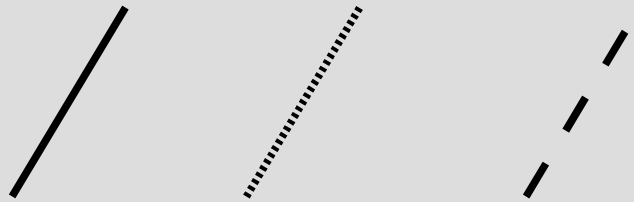
Elementi di base

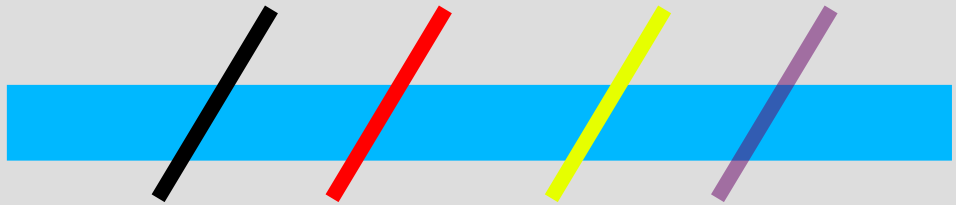
- Solitamente, di ogni elemento si possono definire separatamente
- attributi del bordo (linea)
- attributi del riempimento (area)

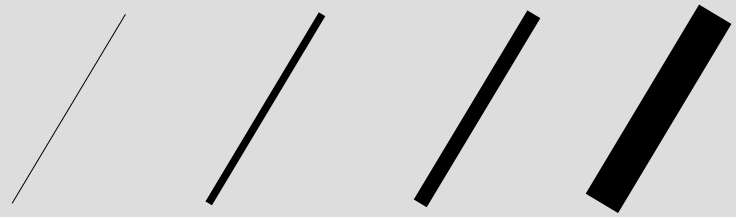


Attributi delle linee

- stile
  - invisibile, continuo, tratteggiato, ...
- colore
  - colore pieno, sfumatura, trasparenza
- larghezza
  - spessore della linea
- pennello (raro)

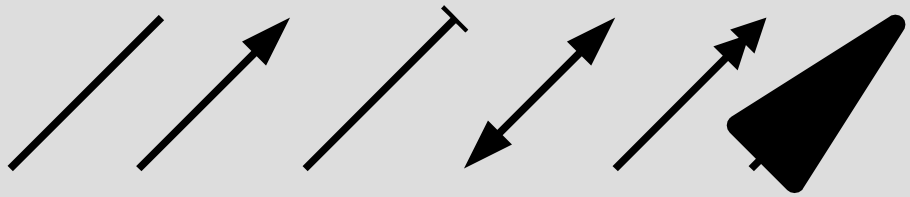




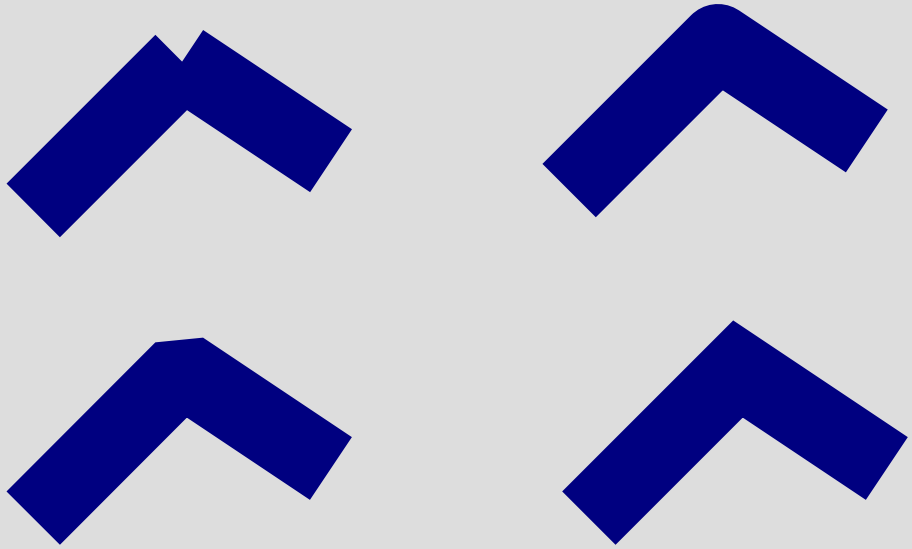


Attributi delle linee

- Forma alle estremità
  - semplice, con frecce, con simboli vari



- Tipo di disegno ai vertici
  - semplice, arrotondato, smussato, mitrato

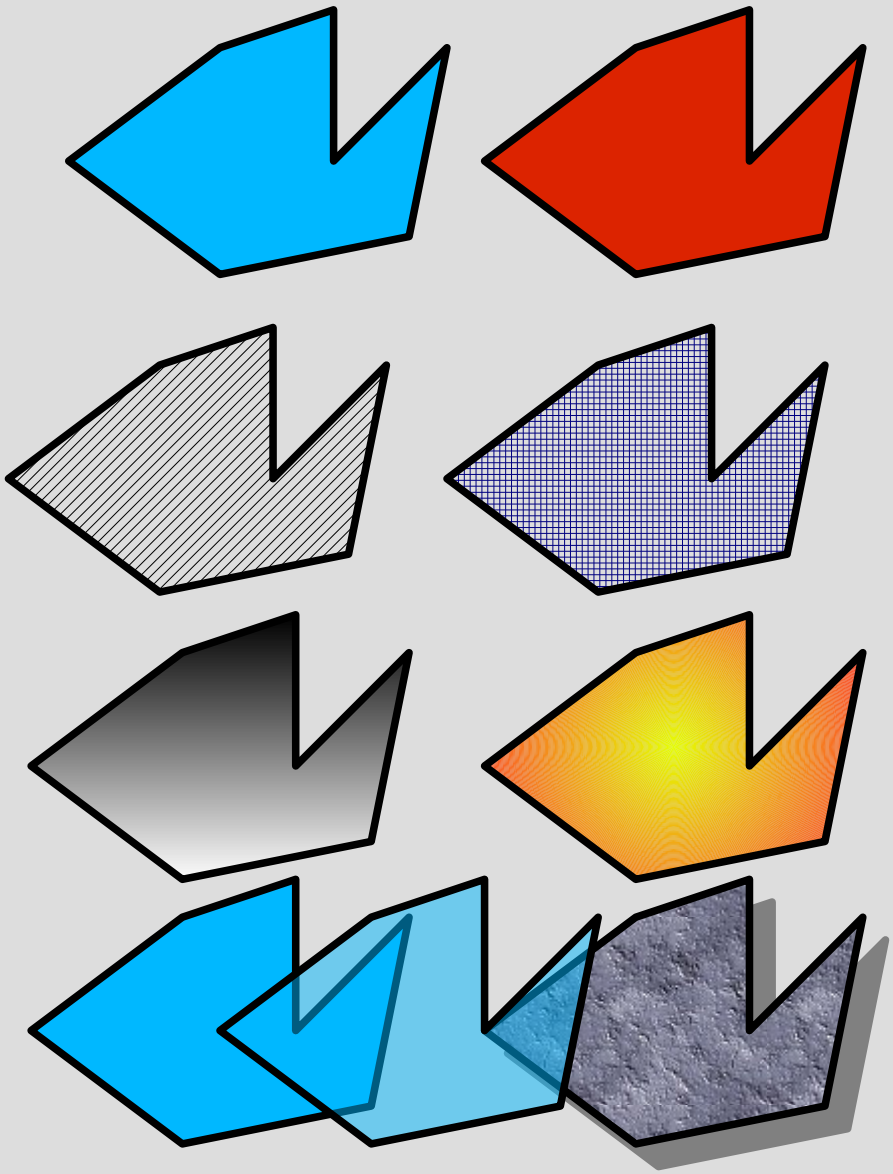


Attributi delle linee

- Quando l'attributo **pennello** è supportato, è possibile realizzare effetti “calligrafici”
  - forma
  - dimensione
  - movimento
  - pressione

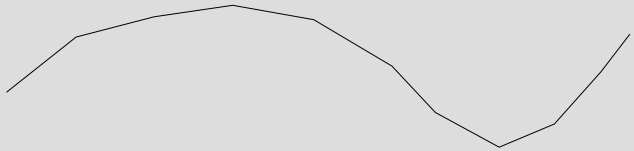
Attributi delle aree

- Riempimento
  - colore, tratteggio, sfumatura, bitmap
  - trasparenza
  - ombreggiatura (raro)
- A seconda dei linguaggi, varie combinazioni

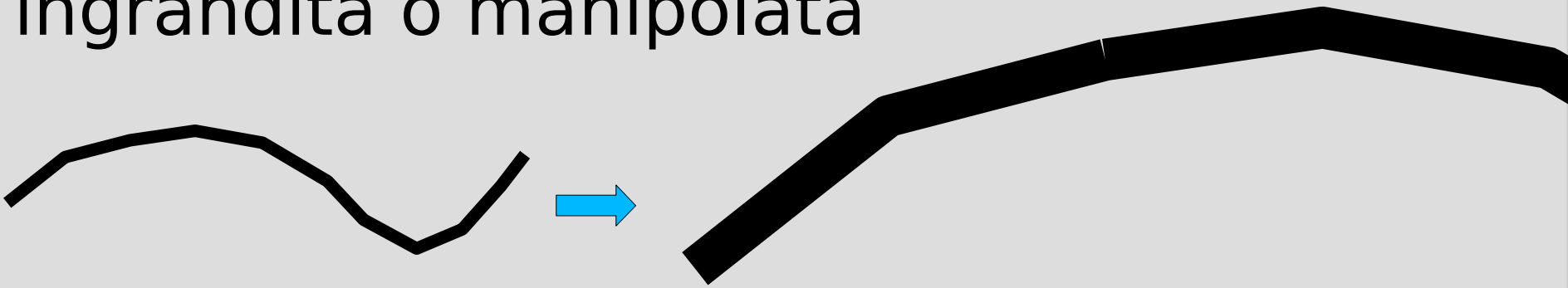


Definizione di curve

- Il caso più semplice è quello in cui la “curva” è in realtà formata da una sequenza di segmenti di retta:

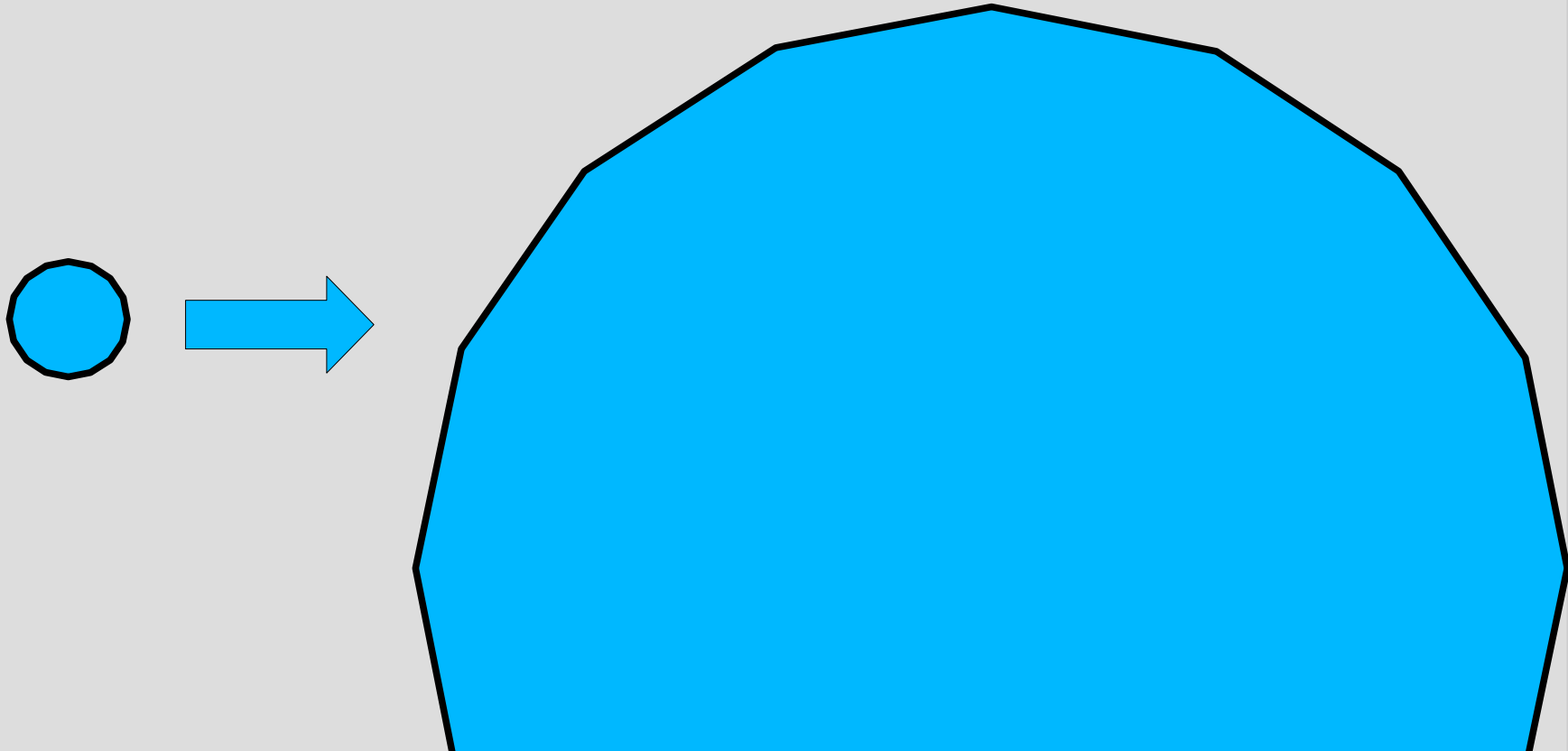


- Tuttavia, questa approssimazione è troppo grossolana se la curva viene ingrandita o manipolata



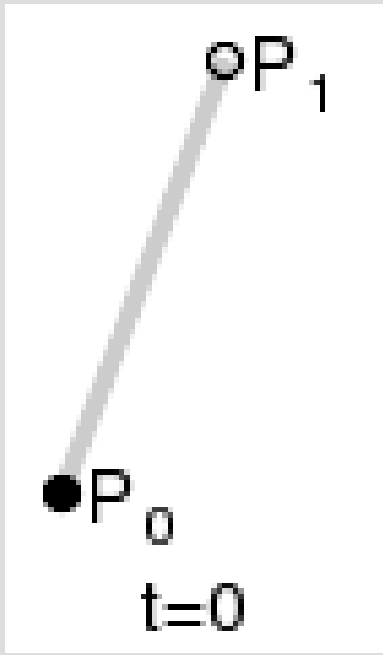
Definizione di curve

- Analogamente avviene per l'approssimazione di un cerchio con un poligono con un numero elevato di lati:



Definizione di curve

- Si usano quindi delle reali curve (in senso **matematico**):
  - **funzioni parametriche** di tipo analitico
  - le funzioni parametriche hanno un valore che varia al variare di un **parametro** *t*
  - al momento del disegno, il motore di rendering calcola le equazioni e disegna la curva corrispondente

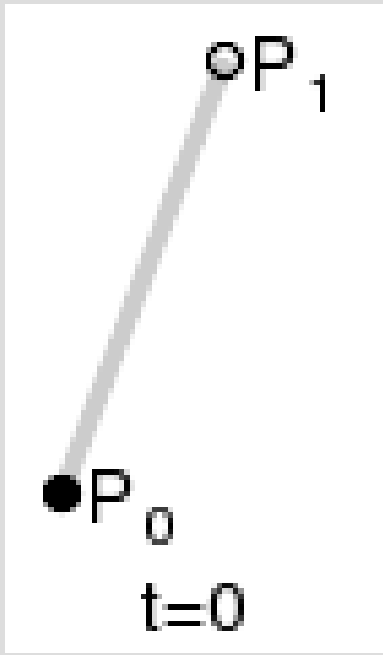


Esempio: equazione di una retta

- Il caso più semplice è quello della retta
- Il segmento di retta fra due punti P_0 e P_1 è definito dall'equazione:

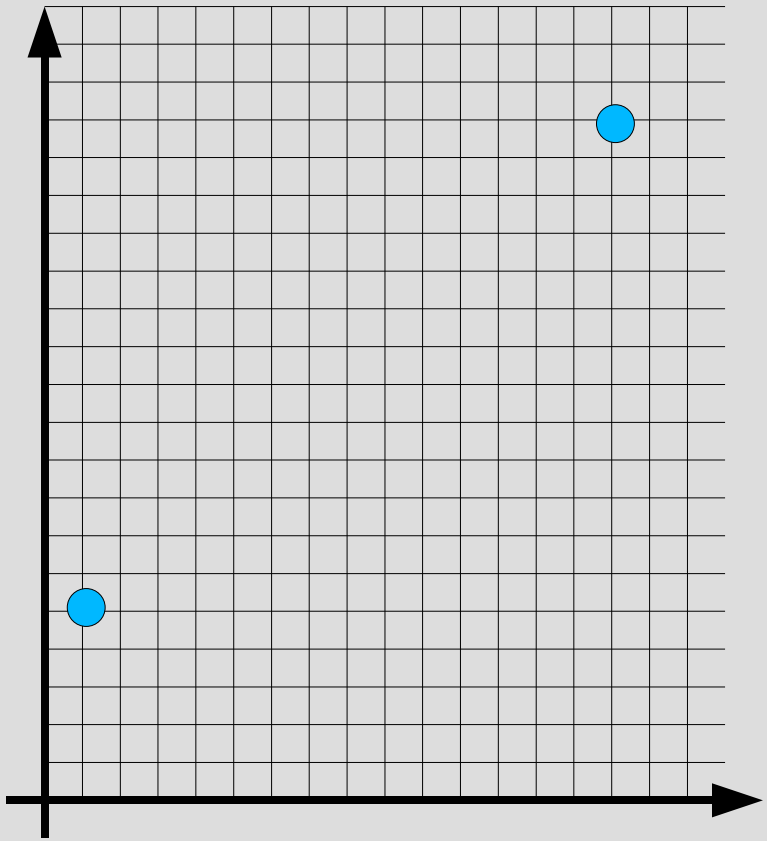
$$B(t)=(1-t)P_0+tP_1, \text{con } t\in[0,1]$$

  - va notato che P_0 e P_1 sono due vettori, ovvero coppie $[x\ y]$
  - t è invece un numero reale; la moltiplicazione si effettua con entrambe le componenti



Esempio: equazione di una retta

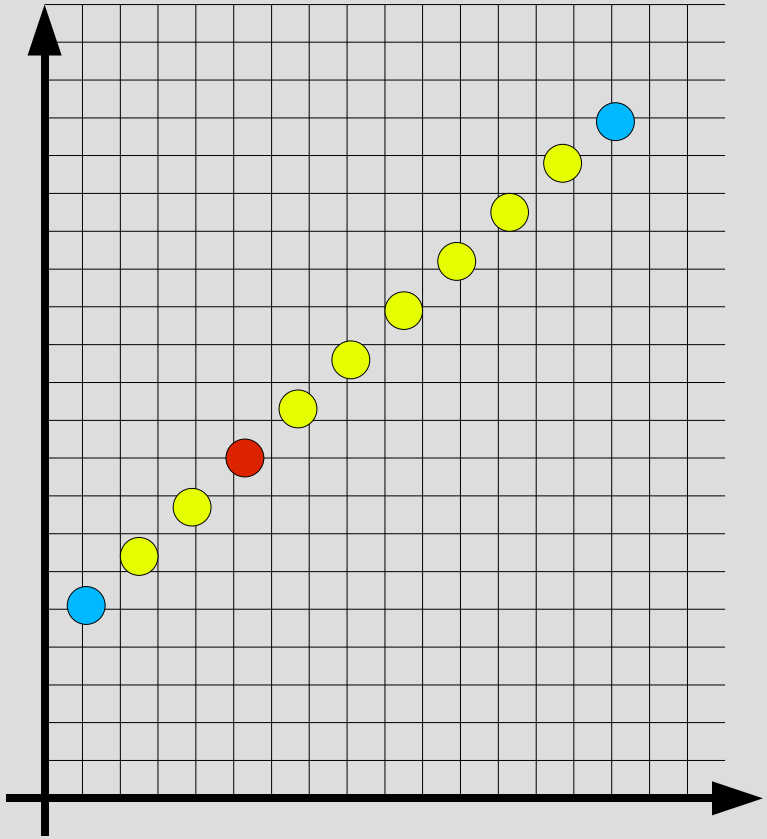
- Proviamo a calcolare manualmente i punti risultanti dall'equazione $B(t)=(1-t)P_0+tP_1$
- Avremo: $P_0=(1,5)$ e $P_1=(15,18)$
- t varierà fra 0 e 1



Esempio: equazione di una retta

- $B(t)=(1-t)P_0+tP_1$
- $P_0=(1,5)$ $P_1=(15,18)$
- t varierà fra 0 e 1

	B(t)	
t	x	y
0,0	1,0	5,0
0,1	2,4	6,3
0,2	3,8	7,6
0,3	5,2	8,9
0,4	6,6	10,2
0,5	8,0	11,5
0,6	9,4	12,8
0,7	10,8	14,1
0,8	12,2	15,4
0,9	13,6	16,7
1,0	15,0	18,0



Es.: per t=0,3, abbiamo

x = (1-0,3)*1 + 0,3*15 = 5,2

y = (1-0,3)*5 + 0,3*18 = 8,9

Definizione di curve

- Esistono molte equazioni che definiscono curve
  - fra le principali: **spline**, **curve di Bezier**
- Tutte si basano sul concetto di **punti di controllo**
- Manipolando opportunamente (ovvero, spostando) i punti di controllo, si può regolare interattivamente la forma della curva

Spline

- Le **spline** sono curve passanti esattamente per *n* punti di controllo dati
- La curva è composta da *n*-1 segmenti, ciascuno dei quali è una approssimazione **polinomiale quadratica** o **cubica** passante per due o tre punti di controllo consecutivi
- In più, si chiede che le **derivate prime** ai punti di contatto dei segmenti siano uguali, ed eventualmente che le **derivate seconde** siano 0, cosicché la spline risultante è continua e “morbida”

Spline (demo)

Spline

- Le spline hanno un funzionamento intuitivo...
  - la curva passa sempre per i punti di controllo
- ... ma causano difficoltà di manipolazione
  - in alcuni casi, la curva “impazzisce”, piccole variazioni nei punti di controllo causano grandi variazioni della curva

Curve di Bezier

- Le **curve di Bezier** prendono un approccio diverso:
  - la curva è suddivisa in frammenti
  - ogni frammento inizia e finisce in un punto di controllo
  - la curvatura è però stabilita da ulteriori punti di controllo secondari, tramite i quali si può stabilire la **tangente** alla curva nei punti principali
  - la curva **non** passa per i punti di controllo secondari

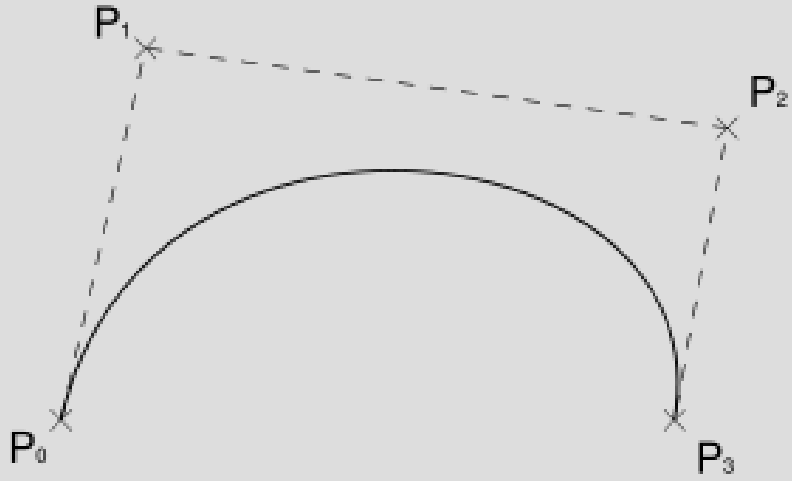
Curve di Bezier

- Le curve di Bezier sono curve parametriche, analoghe al caso della retta che abbiamo già visto
- Anzi... l'equazione parametrica della retta è proprio una **curva di Bezier lineare**
- In grafica, si usano maggiormente le **curve di Bezier quadratiche** e **cubiche**

Curve di Bezier

- **Lineare**

$$B(t)=(1-t)P_0+tP_1$$



- **Quadratica**

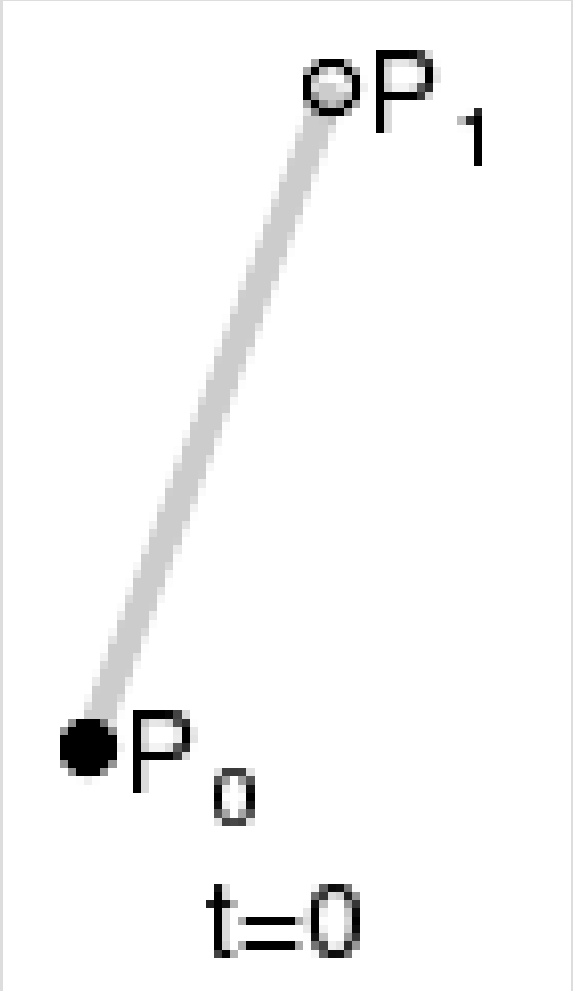
$$B(t)=(1-t)^2P_0+2t(1-t)P_1+t^2P_2$$

- **Cubica**

$$B(t)=(1-t)^3P_0+3t(1-t)^2P_1+3t^2(1-t)P_2+t^3P_3$$

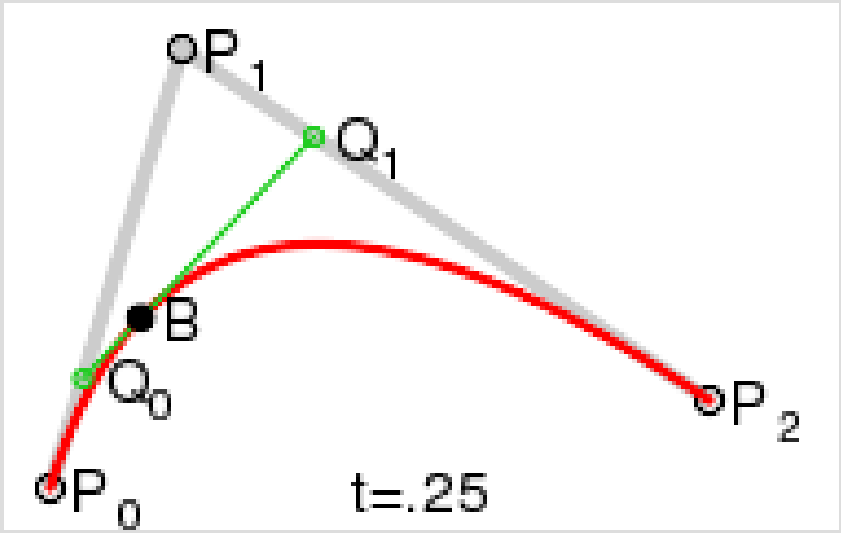
Curve di Bezier

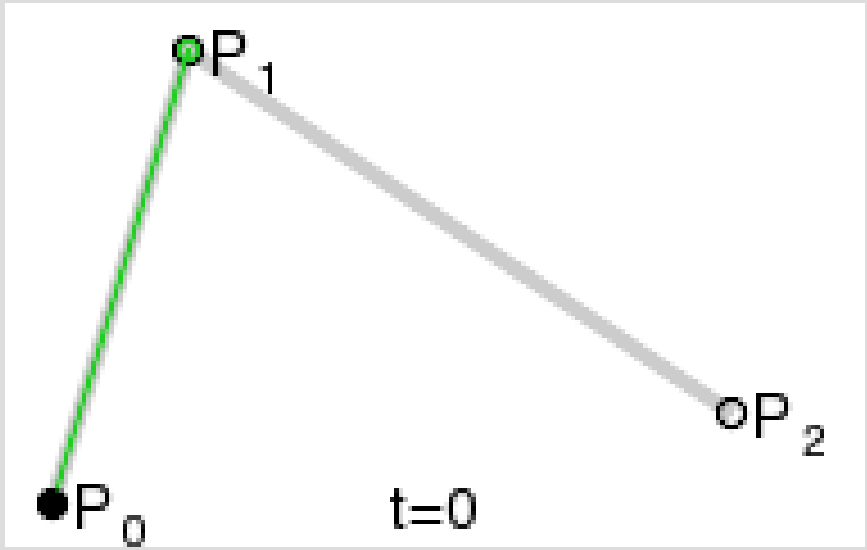
- Le curve di **Bezier lineari** sono quelle che già conosciamo
- Hanno **due** punti di controllo (P_0 e P_1)
- Il punto B(t) si sposta al variare di *t* tracciando la retta che collega P_0 e P_1



Curve di Bezier

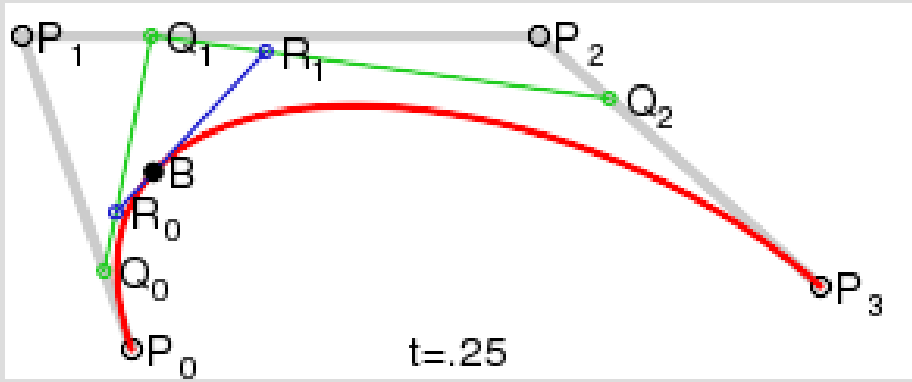
- Le **Bezier quadratiche** hanno **tre** punti di controllo
- Al variare di *t*, due punti Q_0 e Q_1 si spostano fra P_0 e P_1 e fra P_1 e P_2
- B(t) si sposta anche lui fra Q_0 e Q_1

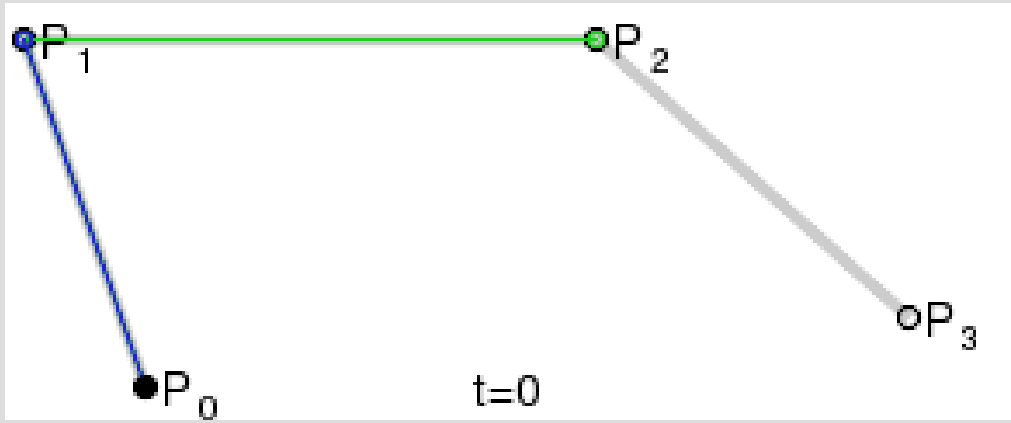




Curve di Bezier

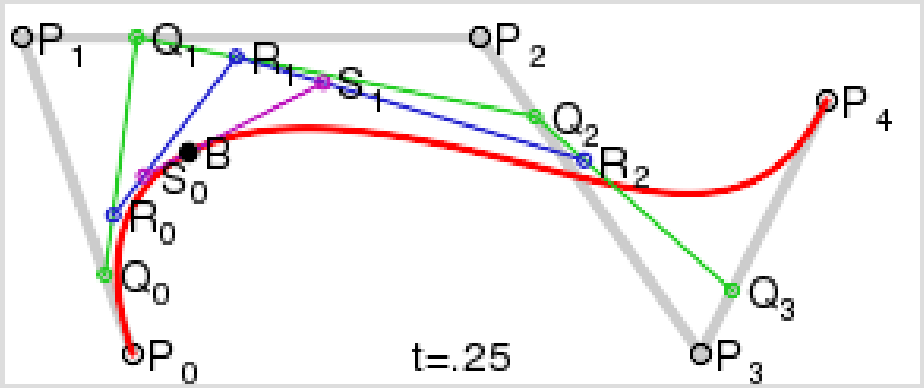
- Le **Bezier cubiche** hanno **quattro** punti di controllo
- Q_0, Q_1 e Q_2 descrivono tre Bezier lineari
- R_0 e R_1 si spostano fra Q_0 e Q_1 e fra Q_1 e Q_2, rispettivamente, descrivendo così due Bezier quadratiche
- B(t) si sposta fra R_0 e R_1 e descrive una Bezier cubica

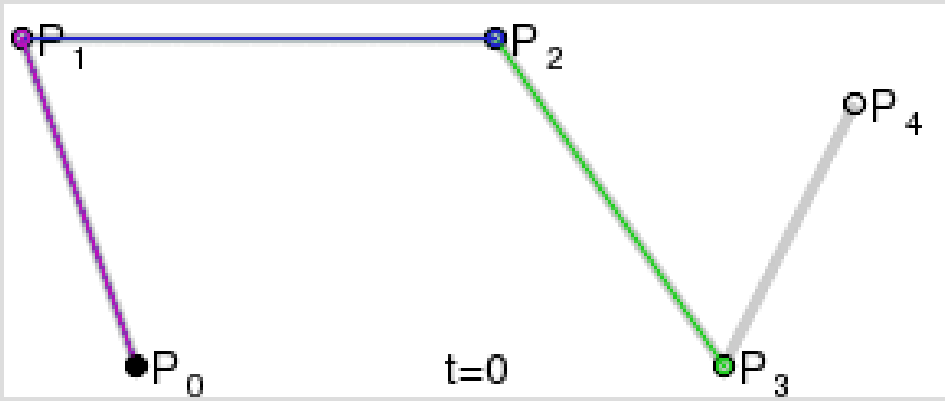




Curve di Bezier

- Il gioco può continuare a piacimento, ma raramente si usano curve di dimensione maggiore
  - troppo costose da calcolare
  - difficili da controllare (troppi punti di controllo)
  - si preferisce unire più segmenti cubici





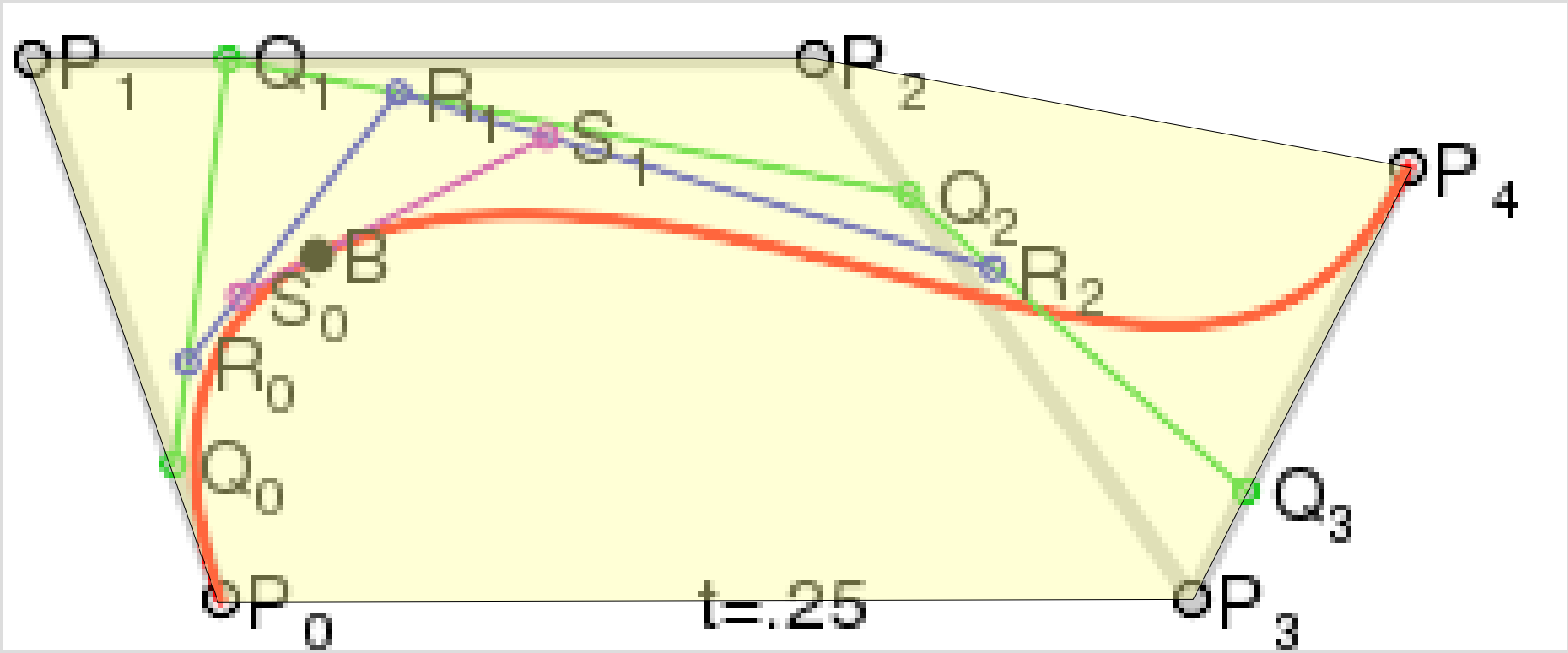
Curve di Bezier

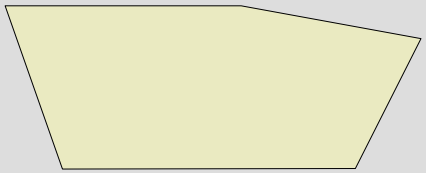
- Nel caso di Bezier cubiche, i punti di controllo hanno un'intepretazione semplice e graficamente intuitiva:
  - il primo e il quarto punto stabiliscono dove la curva comincia e finisce
  - il secondo e il terzo punto stabiliscono:
    - la tangente alla curva agli estremi (angolo rispetto al primo e quarto)
    - la “forza” con cui la curva è tirata verso di loro (distanza rispetto al primo e quarto)

Curve di Bezier

- Oltre ad avere proprietà rilevanti di *continuità* e *regolarità* che rendono le curve di Bezier esteticamente piacevoli, esse godono di alcune altre proprietà matematiche utili
  - per esempio, una curva è sempre contenuta nell'**inviluppo convesso** dei suoi punti di controllo
  - non “scappa” come fanno le spline!
  - vale per qualunque ordine

Curve di Bezier



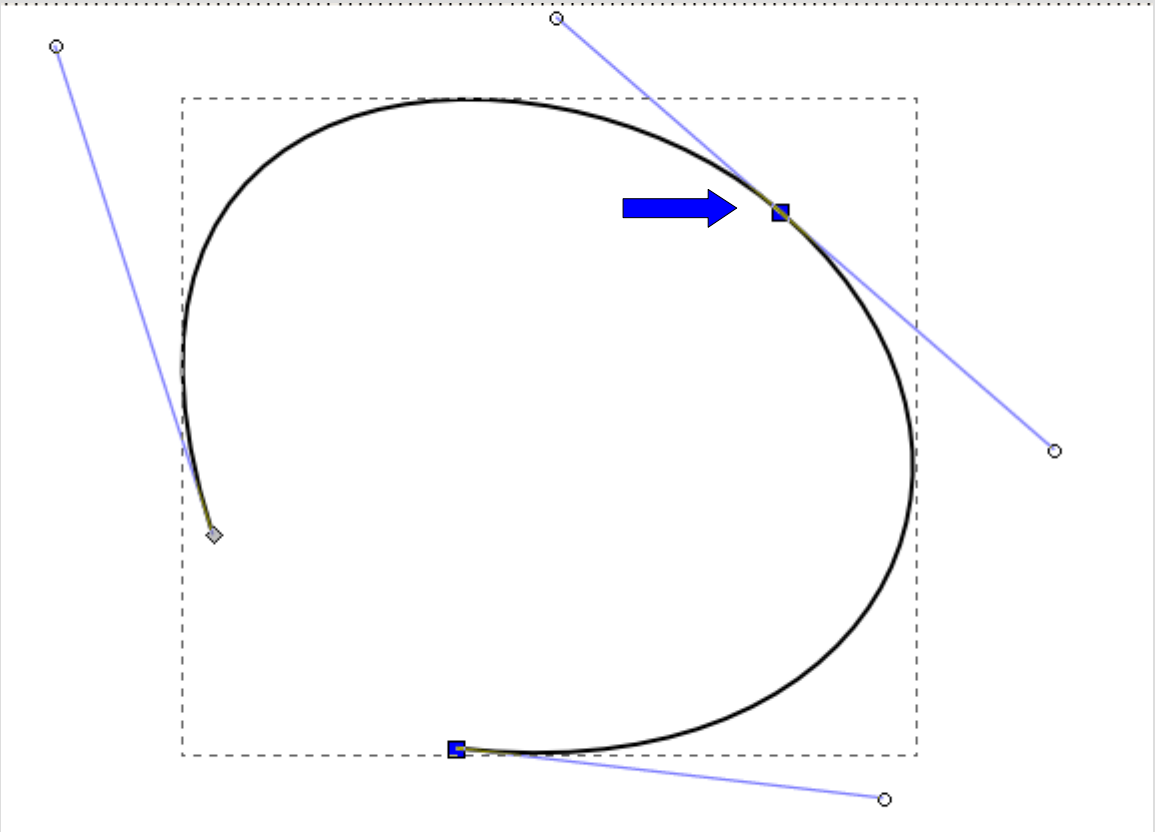


Inviluppo convesso di $\{P_0, P_1, P_2, P_3, P_4\}$

Bezier (demo)

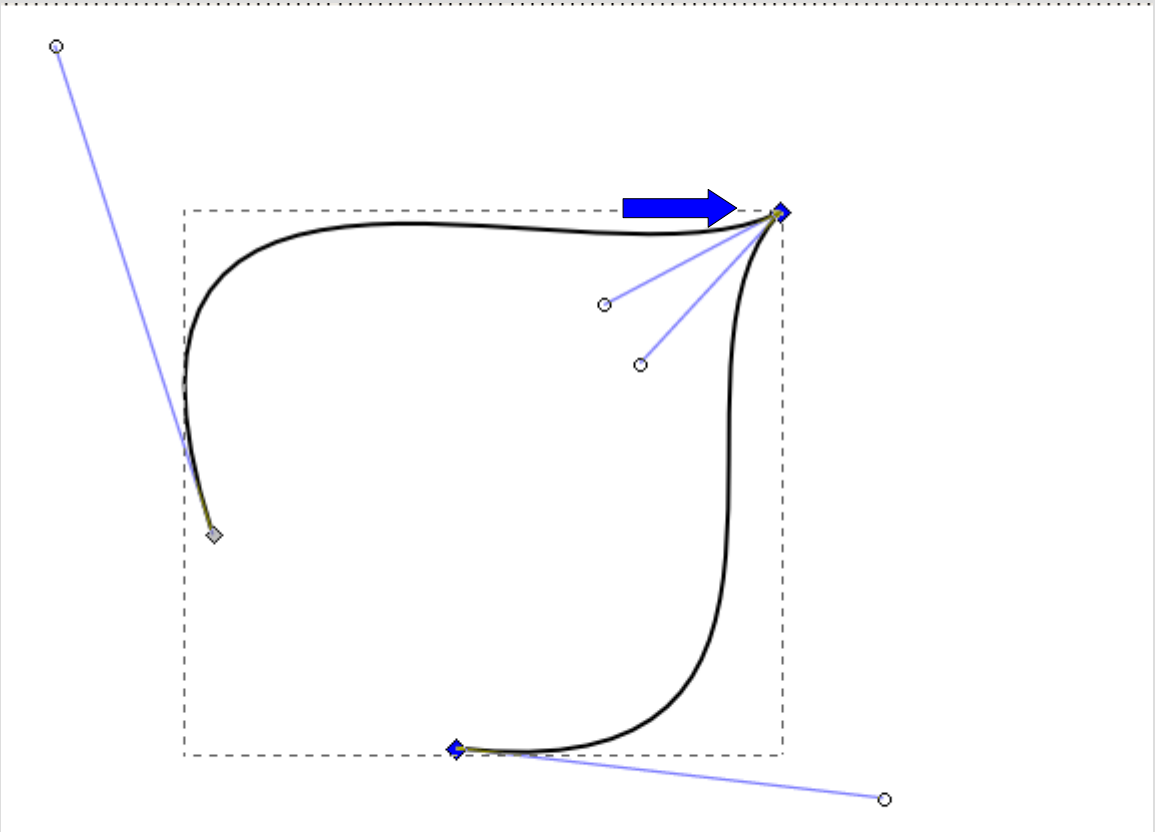
Le curve di Bezier cubiche

- Gli strumenti di editing di immagini vettoriali spesso supportano curve composte da segmenti, ciascuno dei quali è una Bezier cubica
- Nei punti intermedi, i punti di controllo dei due segmenti adiacenti sono normalmente **colineari**
- È possibile però modificarli in modo da creare delle **cuspidi**



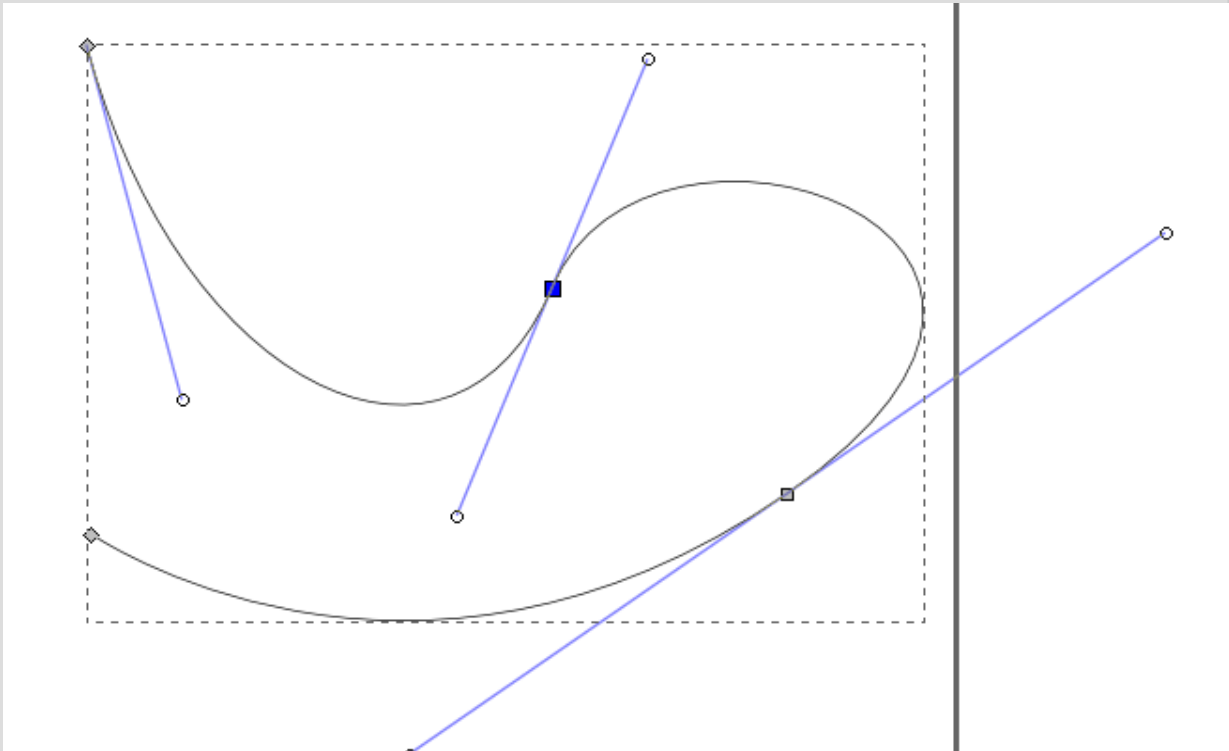
Le curve di Bezier cubiche

- Gli strumenti di editing di immagini vettoriali spesso supportano curve composte da segmenti, ciascuno dei quali è una Bezier cubica
- Nei punti intermedi, i punti di controllo dei due segmenti adiacenti sono normalmente **colineari**
- È possibile però modificarli in modo da creare delle **cuspidi**

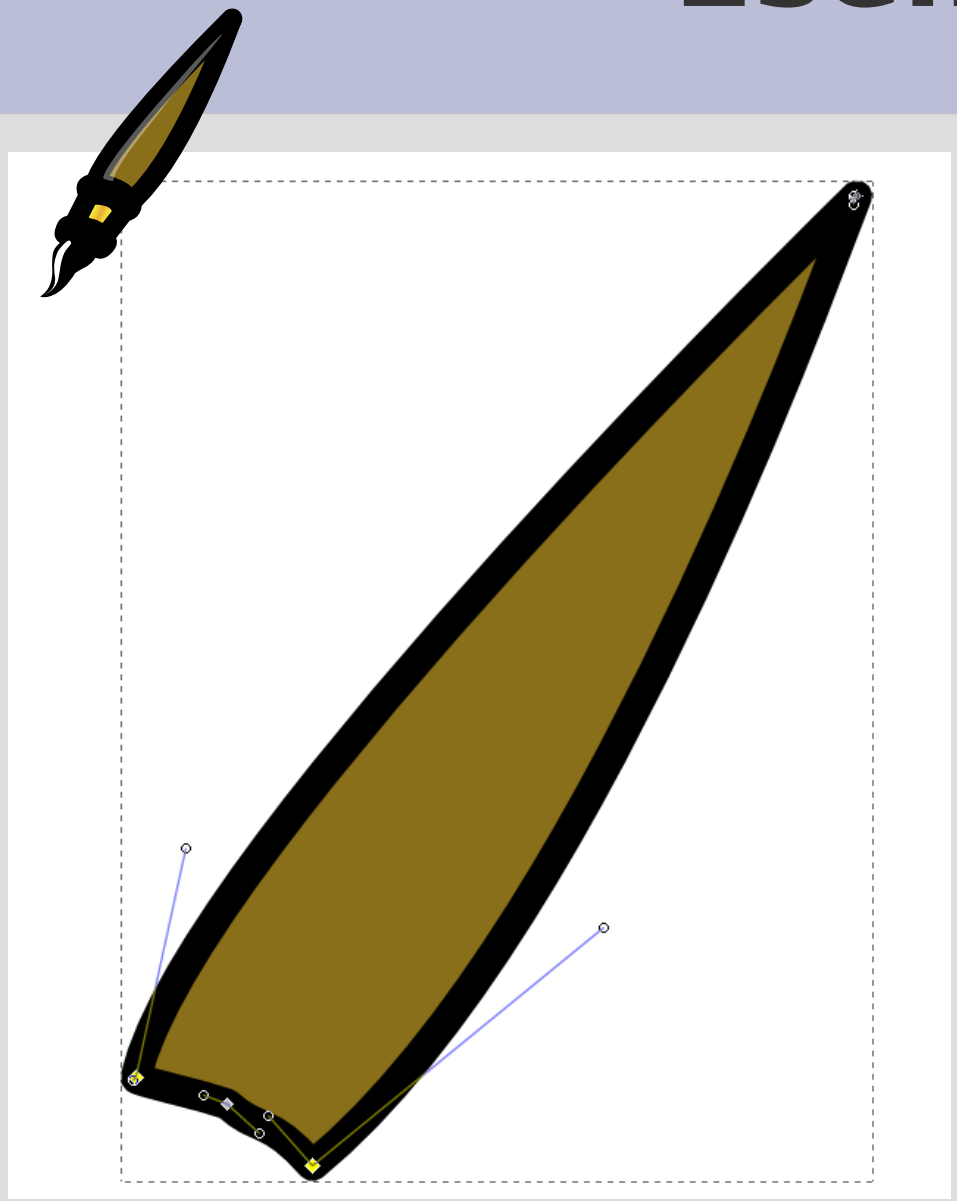


Path

- Con un po' di destrezza, è possibile creare curve di grande complessità
- Sequenze di curve di Bezier come queste sono dette a volte **path** (percorsi, cammini, tracciati)
- Le modalità con cui si editano le curve variano leggermente a seconda dei programmi, ma il principio è lo stesso



Esempio



- Per esempio, l'impugnatura del pennello è un path composto da quattro segmenti
- Essendo un path chiuso, ha quattro punti di controllo principali, per i quali passa il bordo
- Ognuno di questi ha due punti di controllo secondari
- Questi ultimi non sono colineari, e infatti ci sono delle cuspidi (“angoli”)
- Per finire, il path è reso con una linea nera spessa e un'area in colore marrone

Altre operazioni sui path

- I linguaggi vettoriali più avanzati e i programmi di editing migliori consentono di applicare altre operazioni ai path
- Per esempio, è possibile unire più percorsi in uno solo, o definire un nuovo percorso che contiene solo le aree comuni a due altri percorsi, e così via...
- Grande varietà, le possibilità sono moltissime!

Altre operazioni sui path

- Ecco alcuni esempi:
  - unione
  - differenza
  - intersezione
  - esclusione
  - divisione
  - taglio

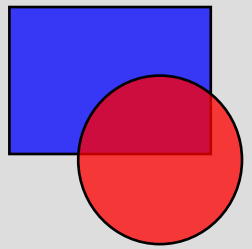
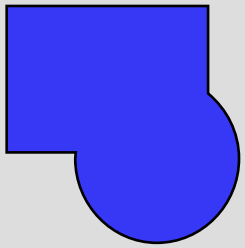
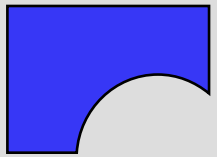


figure originali

unione

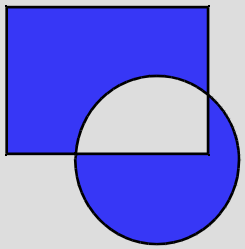


differenza

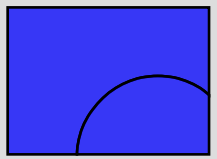


intersezione

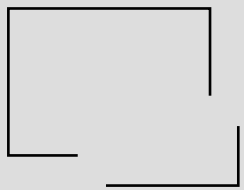
esclusione



divisione



taglio



Altre operazioni sui path

- Alcune operazioni svolte dai programmi svolgono funzioni di utilità:
  - **semplificazione** di un path: si rimuovono i punti di controllo troppo vicini gli uni agli altri, o che non alterano la forma di una curva
  - **tracciatura** di un path: si crea un nuovo path, che segue i bordi (considerando la dimensione della penna) di quello vecchio
  - **estrusione** di un path: si crea un nuovo path “allargando” quello vecchio
  - **...**

Trasformazioni 2D

- Il grande vantaggio della grafica vettoriale è che le immagini vettoriali descrivono **entità matematiche**
- È immediato manipolare **matematicamente** tali entità
- In quasi tutte le manipolazioni **non si perde informazione**
  - **Grande differenza rispetto alla grafica raster!**

Trasformazioni 2D

- Un esempio: **ingrandimento**
  - In grafica raster, i pixel diventano visibili, l'immagine è sgranata: cattiva qualità
  - In grafica vettoriale, il risultato rimane perfetto

Trasformazioni 2D

- Un esempio: **rotazione**
  - La stessa cosa accade con una rotazione di pochi gradi: le immagini raster diventano “scalettate”
  - Le immagini vettoriali rimangono intatte




Trasformazioni 2D

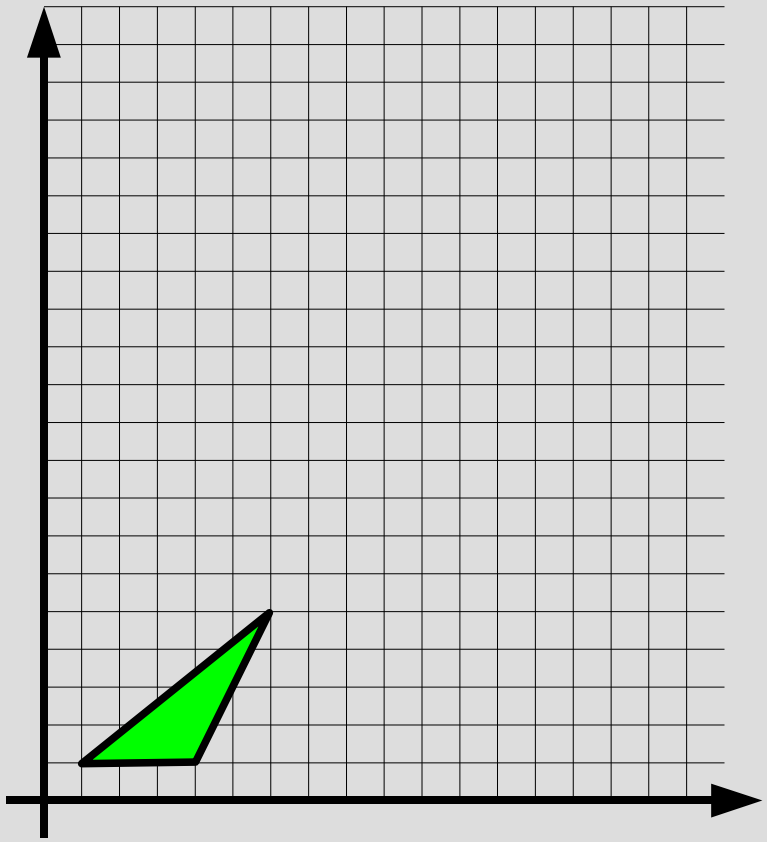
- La ragione di queste differenze risiede nel diverso modo in cui vengono svolte le operazioni su raster e su vettori:
  - su raster, si manipolano i singoli pixel (ingrandendoli, spostandoli), ma poi il risultato deve essere nuovamente approssimato al pixel
  - su vettori, si calcolano le nuove posizioni dei punti di controllo, quindi si ridisegnano curve e altri elementi geometrici ex-novo

Trasformazioni 2D

- **Spostamenti**
  - si somma lo spostamento $\delta=[x\ y]$ a tutti i punti
- **Ingrandimenti**
  - si moltiplicano le coordinate dei punti per il fattore di scala α
- **Rotazioni**
  - si moltiplicano le coordinate dei punti per il seno e il coseno dell'angolo di rotazione θ
- **...**

Trasformazioni 2D

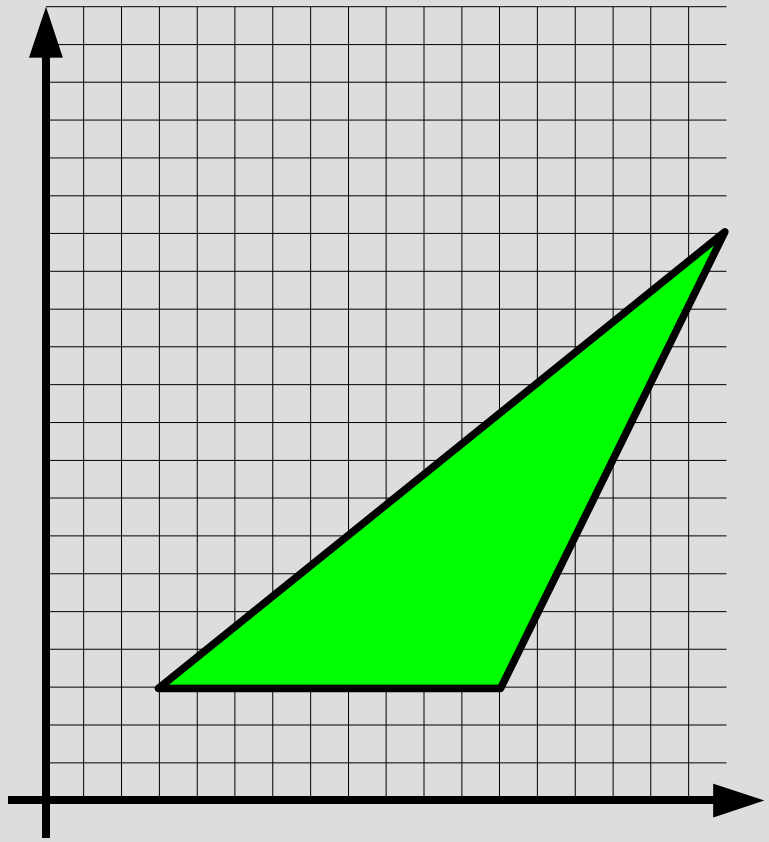
- Esempio:
  - triangolo [1,1],[4,1],[6,5]



Trasformazioni 2D

- Esempio:
  - triangolo T_0=[1,1],[4,1],[6,5]
- **Scaliamo** con α=3.0
  - triangolo $T_1=\alpha T_0$

  T_1=[3,3],[12,3],[18,15]

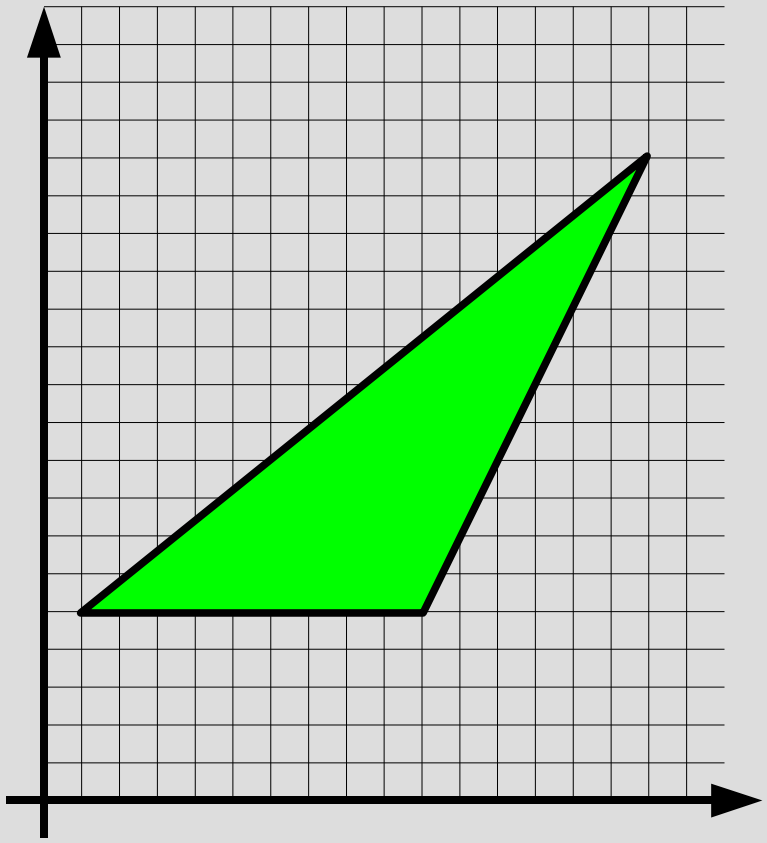


Trasformazioni 2D

- Esempio:
  - triangolo T_0=[1,1],[4,1],[6,5]
- Scaliamo con α=3.0
  - triangolo $T_1=\alpha T_0$

    T_1=[3,3],[12,3],[18,15]
- **Trasliamo** di δ=[-2,2]
  - triangolo $T_2=T_1+\delta$

    T_2=[1,5],[10,5][16,17]



Esercizio

- Disegnare il path composto da due segmenti (curve di Bezier cubiche), identificati dai seguenti punti di controllo:
  - Primo segmento:
    [0,0], [0,5], [5,5], [5,0]
  - Secondo segmento:
    [5,0], [5,-5], [10,-10], [10,0]