Curve di Bézier

Studenti

Caltabiano Daniele

Contarino Daniele

Seminario

Metodi Matematici per l'ottimizzazione

A.A. 2012/2013

Indice

1. Cenni storici
2. Curve di Bézier
3. Polinomi di Bernstein
4. Algoritmo costruttivo di de Casteljau
5. Applicazioni pratiche
6. D2Bezier
7. Analisi Computazionale e dati sperimentali

Cos’è una curva?

E’ un oggetto unidimensionale e continuo che viene rappresentato con delle funzioni continue e differenziabili

Esempi:

Retta, parabola o circonferenza

Un pò di storia…

1911: Polinomi di Bernstein

1959 : Paul de Casteljau, ingegnere della Citroen, realizza un algoritmo per costruire curve e superfici nello spazio

1962: Pierre Bèzier, ingegnere della Renault, formalizza l'espressione matematica delle curve costruite tramite l'algoritmo di de Casteljau

Curve di Bézier
Introduzione

Le curve di Bézier sono usate per disegnare curve nello spazio, dall'andamento morbido, a partire da un numero finito di punti

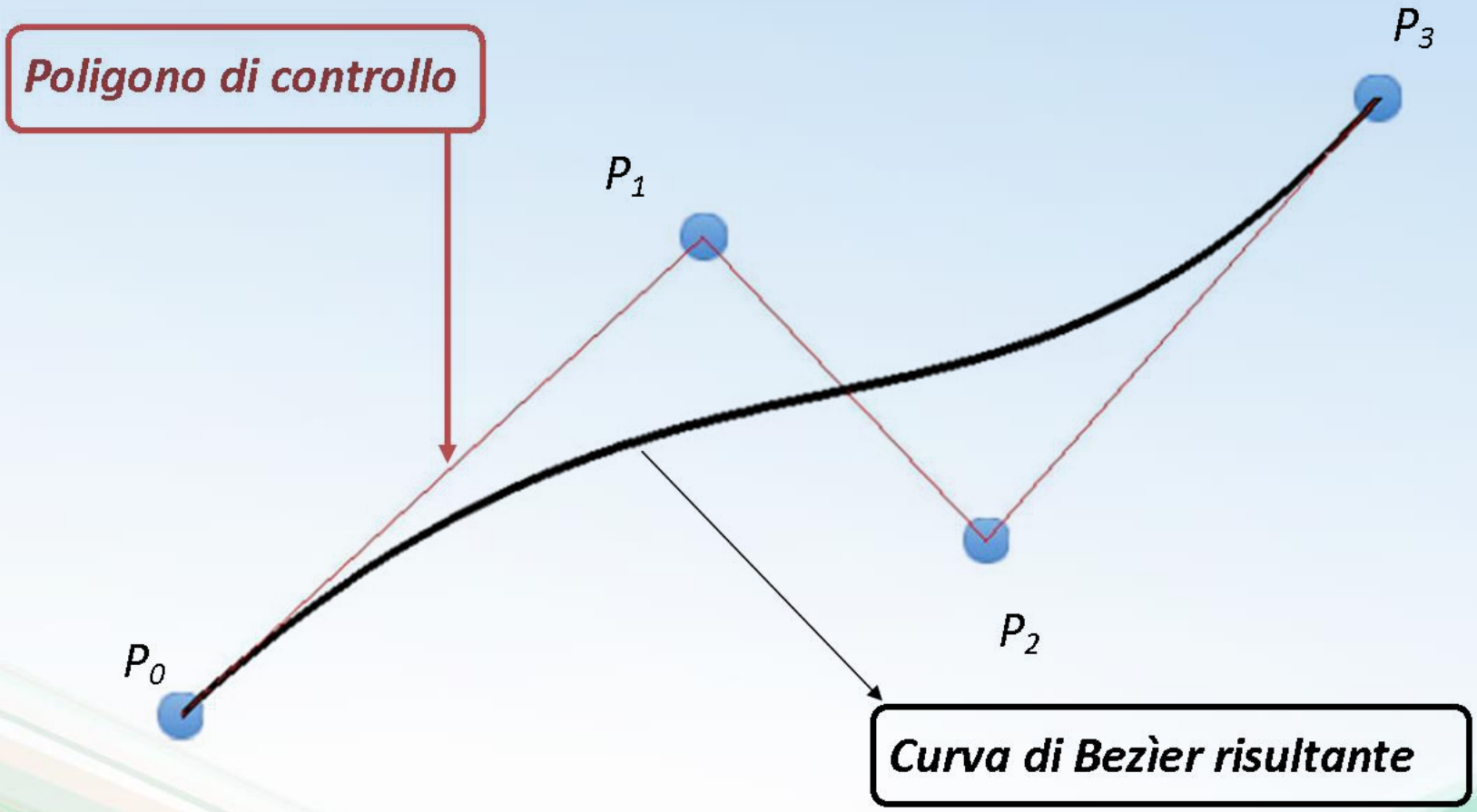
Sono ampiamente usate per le loro proprietà, che permettono ai designer di disegnare curve gradevoli e di qualità

Curve di Bézier

Principio fondamentale

Vengono definite da un *poligono* i cui vertici sono chiamati *punti di controllo* della curva risultante

Il grado della curva è dato dal (numero di vertici -1) del poligono di controllo.

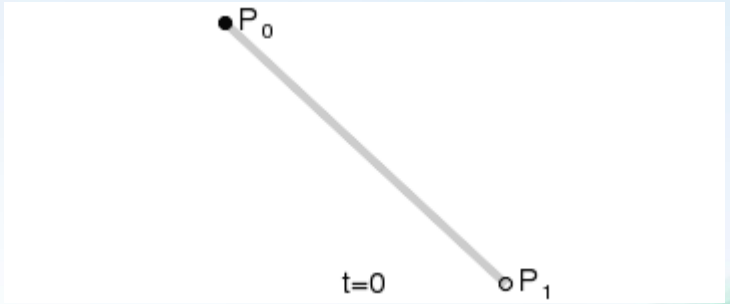

Poligono di controllo
P_3
P_1
P_2
P_0
Curva di Bezìer risultante

Curve di Bézier

lineare

Dati due punti p_0 e p_1 dello spazio affine, la curva di Bézier determinata da questi due punti e l'interpolazione lineare del segmento che li congiunge.

Essa si muove di moto rettilineo uniforme e la legge oraria che descrive questo moto è:



$$\gamma P_0 P_1 = (1 - t)P_0 + tP_1$$

$$t \in [0, 1]$$

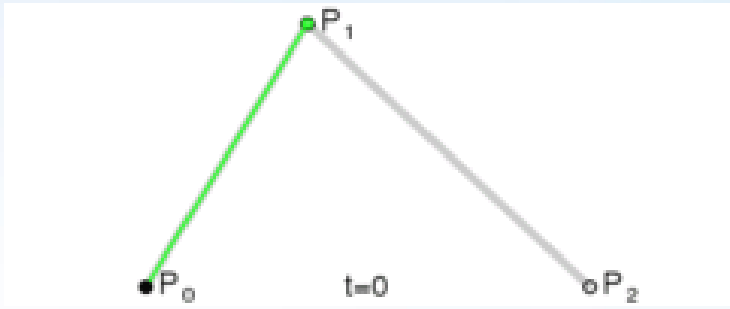
Il segmento si riduce ad un unico punto se P0 = P1

Curve di Bézier

quadratica

Dati tre punti p_0, p_1 e p_2 dello spazio affine, possiamo iterare l'interpolazione lineare ottenendo così una curva di Bézier quadratica definita dalla funzione:

$$\gamma P_0 P_1 P_2 = (1-t)^2 p_0 + 2t(1-t)p_1 + t^2 p_2 \qquad t \epsilon [0,1]$$



Di seguito la costruzione :

$$B(t) = \gamma p_0 p_1 = (1-t)p_0 + tp_1$$

$$C(t) = \gamma p_1 p_2 = (1-t)p_1 + tp_2$$

$$D(t) = \gamma P_0 P_1 P_2 = (1-t)B + tC$$

Se i tre punti sono allineati, si ottiene ancora un segmento, anche se non più percorso a velocita costante

Curve di Bézier

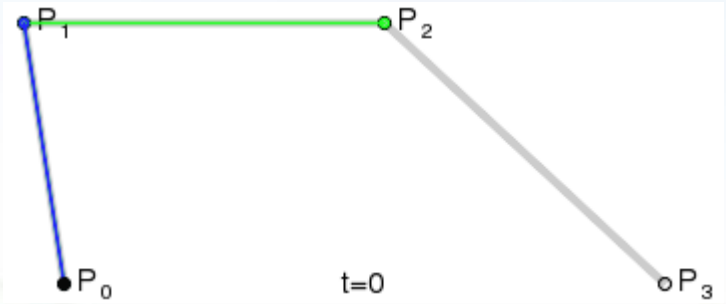
cubica

In modo analogo si definiscono le curve di Bézier di terzo grado, iterando una volta in più il procedimento di interpolazione lineare.

Dati 4 punti p_0, p_1, p_2 *e* p_3 la legge oraria che descrive questo moto è data dalla funzione parametrica:

$$\boldsymbol{\gamma P_0 P_1 P_2 P_3 = (1-t^3)p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t)p_2 + t^3 p_3}$$

$$\boldsymbol{t \in [0,1]}$$



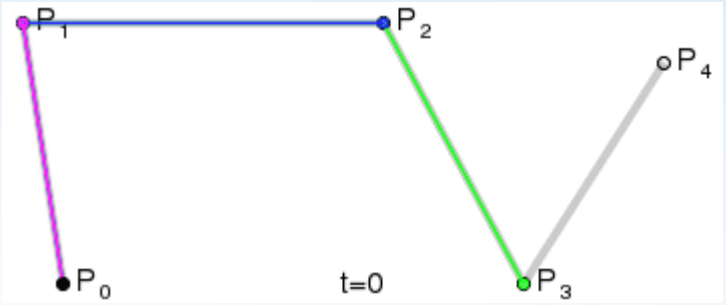
Curve di Bézier

caso generale

Possiamo quindi dare una definizione generale delle curve di Bézier.

Dati $n+1$ punti, P_0,…, P_n , dello spazio affine, la curva di Bézier di ordine n determinata dai punti dati è la curva:

$$\gamma P_0 \dots P_n\,(t) = \sum_{k=0}^{n} B_k^n(t) P_k \quad per \quad t \in [0,1]$$



Dove i coefficenti B_k^n sono i polinomi di Bernstein di grado n, quindi:

$$B_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k} \quad per\ k = 0, \dots, n$$

Curve di Bézier

proprietà principali

- La curva di Bézier è sempre interna alla poligonale avente come vertici i punti di controllo P_k.

Questa proprietà è conseguenza del fatto che i polinomi di Bernstein sono positivi nell'intervallo [0, 1] ed inoltre

$$\sum_{k=0}^{n} B_{n.k}(t) = 1$$

Questa proprietà garantisce inoltre che la curva non oscilla rispetto ai punti di controllo.

Curve di Bézier

proprietà principali

- Calcolando le derivate della curva agli estremi, cioè per $t = 0$ e $t = 1$ si trova:

$$\boldsymbol{P'(0) = n(P_1 - P_0)}$$

$$\boldsymbol{P'(1) = n(P_n - P_{n-1})}$$

particolare si vede che la tangente agli estremi coincide con la retta congiungente rispettivamente i primi due e gli ultimi due punti di controllo.

Curve di Bézier

proprietà principali

- La Curva inizia in P_o e termina in P_n

Essendo che la curva di Bézier è descritta dalla funzione $B_n(t) = \sum_{k=0}^{n} \binom{n}{k} t^k (1-t)^{n-k} P_k$

$$B_n(t) = \begin{cases} P_0 \; se \; t = 0 \\ P_n \; se \; t = 1 \end{cases}$$

Ponendo $0^0 \equiv 1$

Curve di Bézier
proprietà principali

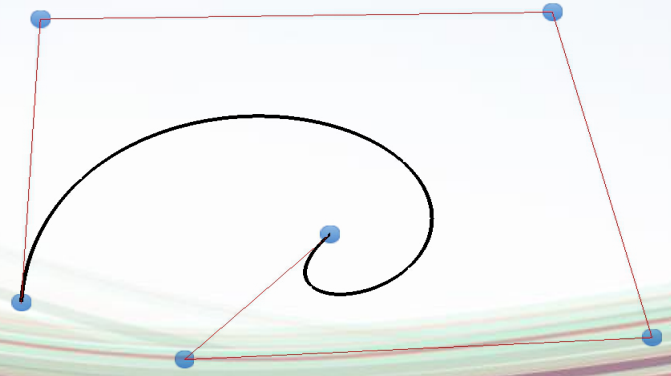
- Una curva può essere spezzata in qualsiasi punto in un arbitrario numero di sotto curve, ognuna delle quali è essa stessa una curva di Bézier

- La curva è una linea retta se e solo se tutti i punti di controllo giacciono sulla curva

Curve di Bézier
proprietà principali

Convex Hull

La curva giace completamente nella convex hull dei punti di controllo, in quanto ogni punto della curva è una combinazione baricentrica di questi



Curve di Bézier
svantaggi

Sebbene una curva di Bézier possa essere modificata tramite i suoi vertici di controllo, il controllo che ne risulta non è sufficientemente locale:

- se aggiungiamo all'insieme dei vertici di controllo un solo vertice è necessario ricalcolare completamente l'equazione parametrica della curva;
- la sostituzione di un solo punto di controllo causa un cambiamento di forma dell'intera curva.
- Per rappresentare accuratamente alcune forme complesse `e necessario un grado elevato.

Curve di Bezìer
note

- Le curve più usate sono le quadratiche e le cubiche
- Curve di grado più alto sono computazionalmente più onerose da calcolare
- Per realizzare forme più complesse si possono unire più curve di secondo e terzo ordine tra di loro

Polinomi di Bernstein

E’ lo strumento matematico che utilizza Bézier per la formulazione delle sue curve.

Definizione

Dati interi n e k, si definisce k-esimo polinomio di Bernstein di grado n

$$B_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k}, t \in [0,1], k = 0, \dots, n$$

Polinomi di Bernstein
Proprietà

Fissato un intero $n \geq 0$, siano $B_k^n(t)$ i polinomi di Bernstein di grado n, valgono le seguenti proprietà:

- ***Positività***

  Assumono valori non negativi

  $$B_{n,k\geq 0,\forall t\, \epsilon\, [0,1],k=0,1,\ldots,n}$$

- ***Simmetria***

$$B_{n-k}^n(t) = B_k^n(1-t)$$

$$per\ ogni\ k = 0, \ldots, n, per\ ogni\ t\epsilon[0,1]$$

Polinomi di Bernstein
Proprietà

- Partizionano l'unità

$$\sum_{k=0}^{n} B_k^n(t) = 1$$

- La derivata del polinomio di Bernstein di grado n è un polinomio di grado n-1

Algoritmo di De Casteljau

E’ un metodo che permette di costruire in modo semplice e algoritmico le curve nello spazio aventi rappresentazione parametrica di tipo polinomiale.

Fissato un valore $t \epsilon [0,1]$, esso permette di calcolare il punto corrispondente sulla curva $c(t)$ mediante interpolazioni lineari ripetute a partire dai punti di controllo.

Algoritmo di De Casteljau
Costruzione generale

Ogni lato del poligono di controllo, viene diviso in 2 parti formando un nuovo punto. I nuovi punti calcolati, collegati tra loro, determinano un
poligono con n-1 lati. Ognuno di questi n-1 lati viene suddiviso in 2 parti dando luogo ad una batteria di nuovi punti.
Questi nuovi punti, collegati tra loro, determinano un
poligono con n - 2 lati.

Così procedendo si ottiene al termine un solo lato che, suddiviso in 2 parti permette di determinare il punto sulla curva.

Algoritmo di De Casteljau
interpretazione geometrica

Caso base

Dati Due punti Po e P1 e sia t un parametro reale compreso tra 0 e 1, costruiamo la funzione:

$$\boldsymbol{P_0^1(t) = (1-t)P_0^0 + tP_1^0}$$

Che rappresenta una curva di Bézier lineare

Algoritmo di De Casteljau interpretazione geometrica

Costruzione di una Curva quadratica

Dati tre punti P_0, P_1 e P_2 , $t \epsilon [0,1]$

- **Primo Passo**

$$P_0^1(t) = (1-t)P_0^0 + tP_1^0$$
$$P_1^1(t) = (1-t)P_1^0 + tP_2^0$$

- **Secondo Passo**

$$P_0^2(t) = (1-t)P_0^1 + tP_1^1$$

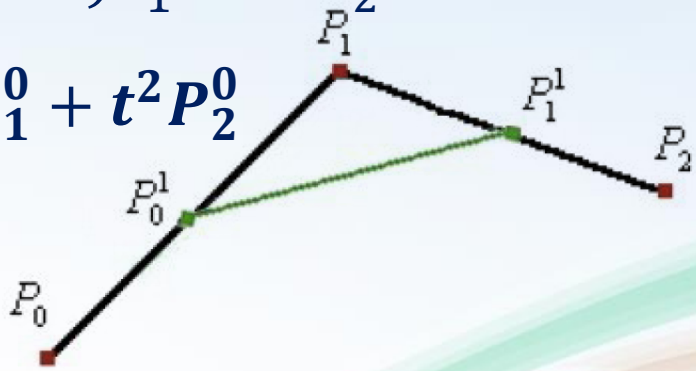
Algoritmo di De Casteljau
interpretazione geometrica

- Sostituendo dai passi precedenti si ha:

$$P_0^2(t) = (1-t)[(1-t)P_0^0 + tP_1^0] + t[(1-t)P_1^0 + \mathrm{t}P_2^0]$$

$$= (1-t^2)P_0^o + (1-t)tP_1^0 + t(1-t)P_1^0 + t^2P_2^0$$

$$= \boldsymbol{(1-t^2)P_0^o + 2t(1-t)P_1^0 + t^2P_2^0}$$



Che è la curva di Bézier quadratica

Algoritmo di De Casteljau
interpretazione geometrica

Costruzione di una curva cubica

Analogamente, per 4 Punti P_0, P_1, P_2, P_3 si ha la seguente formula:

$$\boldsymbol{P_0^3(t) = \left(1 - t^3\right)P_0^o + 3t(1-t)^2 P_1^0 + 3t^2(1-t)P_2^o + t^3 P_3^o}$$

$$\boldsymbol{t \, \epsilon \, [0, 1]}$$

Algoritmo di De Casteljau

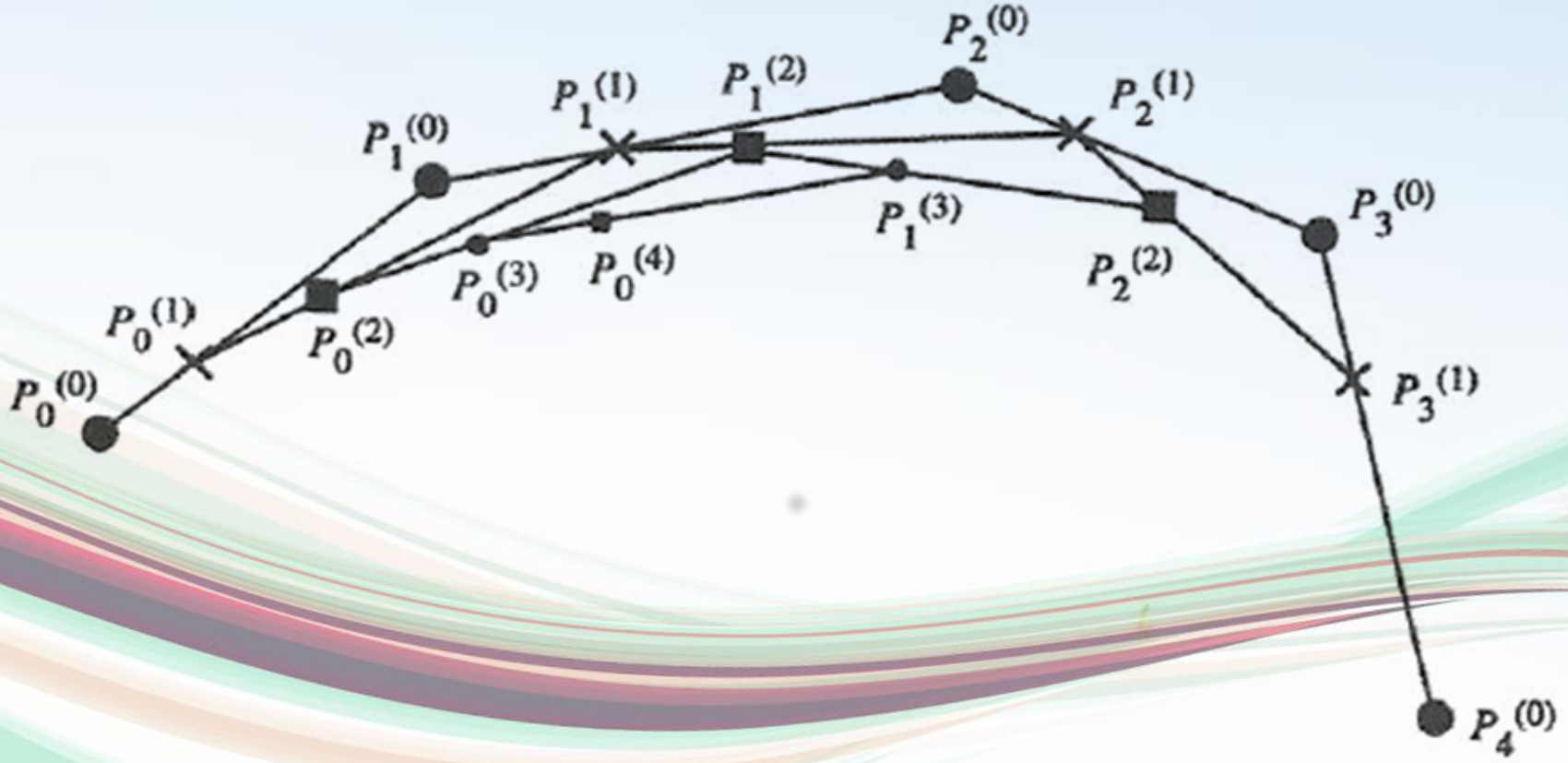
In generale, dati n+1 punti , la curva di grado n è data dalla seguente equazione di ricorrenza

$$\boldsymbol{P_i^k(t) = (1-t)P_i^{k-1}(t) + tP_{i+1}^{k-1}(t)} \qquad \boldsymbol{t \in [0,1]}$$

$$con\ k = 1,2,\ldots.,n \quad , \qquad i = 0,1,\ldots,n-k$$

Algoritmo di De Casteljau
esempio

Costruzione grafica di de Casteljau partendo da 5 vertici di controlloP_0^o, P_1^o, P_2^o, P_3^o, P_4^o per ottenere il punto sulla curva P_0^4



Algoritmo di De Casteljau

Considerazioni

La costruzione geometrica alla base dell'algoritmo di de Casteljau gode delle seguenti proprietà:

STABILITA' in quanto i coefficienti $1 - t$ e t sono compresi tra 0 e 1 e quindi non c'è amplificazione dell'errore.

RICORSIVITA' in quanto viene applicata la stessa formula per ogni passaggio.

Algoritmo di De Casteljau
analisi computazionale

PASSO 1: A partire da n+1 punti si determinano n punti.

Per trovare ogni punto occorrono 2 moltiplicazioni e 1 addizione

2n moltiplicazioni n addizioni

PASSO 2: A partire da n punti si determinano n-1 punti.

Per trovare ogni punto occorrono 2 moltiplicazioni e 1 addizione,

quindi 2(n-1) moltiplicazioni n-1 addizioni

….

Algoritmo di De Casteljau

Costo computazionale

$$2\sum_{i=1}^{n} i = 2\frac{n(n+1)}{2} = n(n+1)$$ ***moltiplicazioni***

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$ ***addizioni***

Complessità dell'algoritmo $O(n^2)$

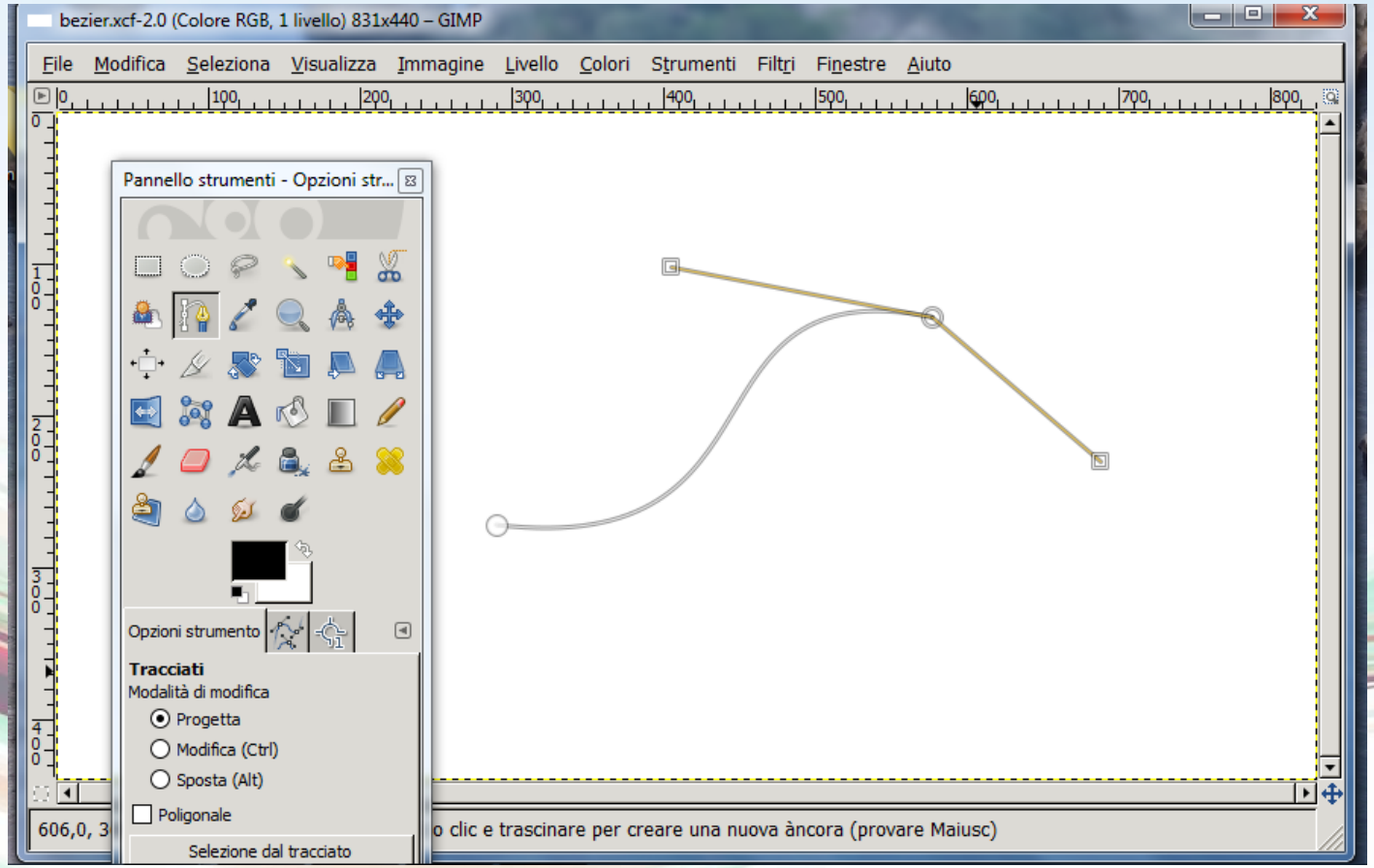
Applicazioni delle Curve di Bèzier

Esempi di applicazioni che usano le Curve di Bézier:

- Programmi di grafica (Photoshop, GIMP, inkScape, etc.);
- Programmi di CAD;
- Programmi di produttività personale (Office, etc.);

Curve di Bèzier su GIMP

GIMP è un noto editor grafico open source disponibile su più sistemi operativi. Al suo interno, tra i tanti strumenti, GIMP ha lo strumento ‘Tracciati’ che non è altro che l’applicazione delle curve cubiche di Bézier.



Curve di Bézier su GIMP

Come abbiamo detto, GIMP è un programma open source, e noi siamo andati a sbirciare all'interno del codice sorgente per vedere l'implementazione di queste curve.

Le curve di Bézier sono implementate dentro GIMP all'interno del file 'gimpcurve.c'

Curve di Bézier su GIMP

```
static void gimp_curve_plot (GimpCurve *curve, gint p1, gint p2, gint p3, gint p4){
        gint i;
        gdouble x0, x3;
        gdouble y0, y1, y2, y3;
        gdouble dx, dy;
        gdouble slope; /* the outer control points for the bezier curve. */
        x0 = curve->points[p2].x;
        y0 = curve->points[p2].y;
        x3 = curve->points[p3].x;
        y3 = curve->points[p3].y;
        /* the x values of the inner control points are fixed at * x1 = 2/3*x0 + 1/3*x3 and
        x2 = 1/3*x0 + 2/3*x3 * this ensures that the x values increase linearily with the
        parameter t and enables us to skip the calculation of the x values al togehter - just
        calculate y(t) evenly spaced. */
        dx = x3 -x0;
        dy = y3 - y0;
        g_return_if_fail (dx > 0);
         if (p1 == p2 && p3 == p4) {
                /* No information about the neighbors, * calculate y1 and y2 to get a
                straight line */
                y1 = y0 + dy / 3.0;
                y2 = y0 + dy * 2.0 / 3.0;
        }
```

Curve di Bézier su GIMP

```
} else if (p1 == p2 && p3 != p4) {
        /* only the right neighbor is available. Make the tangent at the right
        endpoint parallel to the line between the left endpoint and the right
        neighbor. Then point the tangent at the left towards the control handle
        of the right tangent, to ensure that the curve does not have an inflection
        point. */
        slope = (curve->points[p4].y - y0) / (curve->points[p4].x - x0);
        y2 = y3 - slope * dx / 3.0;
        y1 = y0 + (y2 - y0) / 2.0;
} else if (p1 != p2 && p3 == p4) { /* see previous case */
        slope = (y3 - curve->points[p1].y) / (x3 - curve->points[p1].x);
        y1 = y0 + slope * dx / 3.0;
        y2 = y3 + (y1 - y3) / 2.0;
} else /* (p1 != p2 && p3 != p4) */ {
        /* Both neighbors are available. Make the tangents at the endpoints
        parallel to the line between the opposite endpoint and the adjacent
        neighbor. */
        slope = (y3 - curve->points[p1].y) / (x3 - curve->points[p1].x);
        y1 = y0 + slope * dx / 3.0;
        slope = (curve->points[p4].y - y0) / (curve->points[p4].x - x0);
        y2 = y3 - slope * dx / 3.0;
}
```

Curve di Bézier su GIMP

```
        /* finally calculate the y(t) values for the given bezier values. We can use
        homogenously distributed values for t, since x(t) increases linearily. */
        for (i = 0; i <= ROUND (dx * (gdouble) (curve->n_samples - 1)); i++) {
                gdouble y, t;
                gint index;
                t = i / dx / (gdouble) (curve->n_samples - 1);
                y = y0 * (1-t) * (1-t) * (1-t) + 3 * y1 * (1-t) * (1-t) * t + 3 * y2 * (1-t) *
                      t * t + y3 * t * t * t;
                index = i + ROUND (x0 * (gdouble) (curve->n_samples - 1));
                if (index < curve->n_samples)
                        curve->samples[index] = CLAMP (y, 0.0, 1.0);
        }
}
```

Curve di Bézier su GIMP

Considerazioni

- È un ottimo strumento per applicare le curve di Bézier in maniera semplice ed efficace

- Utilizza il polinomio di Bernstein per il calcolo della curva di Bézier

- Genera curve solo di 3 ordine, che è un buon compromesso tra flessibilità e prestazioni computazionali

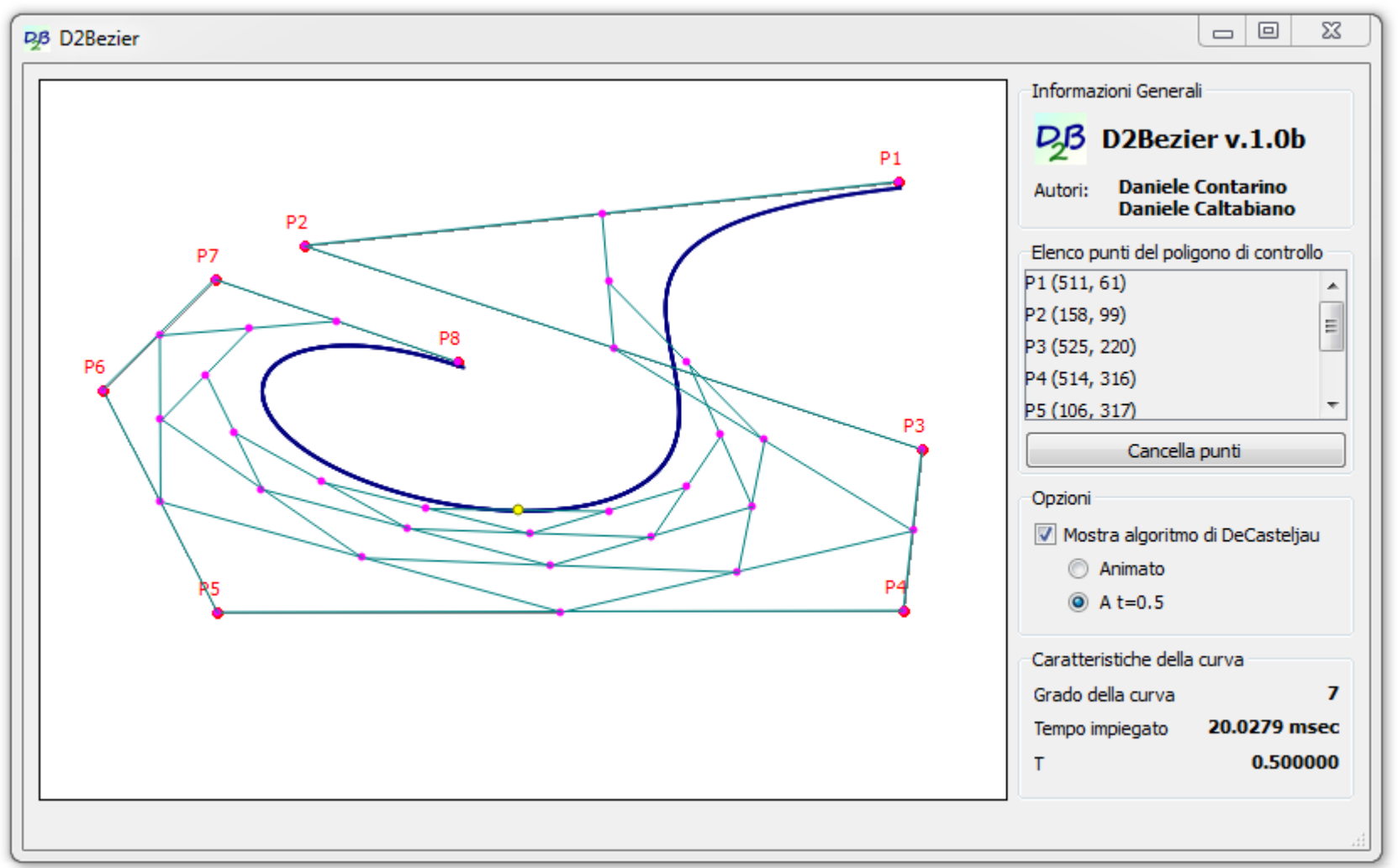
Curve di Bèzier

D2Bezier

E’ un progetto software realizzato per analizzare in dettaglio i tempi di CPU con un grado della curva alto

Il software riceve in input dei punti, all’interno di un pannello e costruisce dinamicamente la Curva

D2Bezier


D2Bezier
P1
P2
P7
P8
P6
P3
P5
P4
Informazioni Generali
D2Bezier v.1.0b
Autori: Daniele Contarino
Daniele Caltabiano
Elenco punti del poligono di controllo
P1 (511, 61)
P2 (158, 99)
P3 (525, 220)
P4 (514, 316)
P5 (106, 317)
Cancella punti
Opzioni
Mostra algoritmo di DeCasteljau
Animato
A t=0.5
Caratteristiche della curva
Grado della curva 7
Tempo impiegato 20.0279 msec
T 0.500000

D2Bèzier
codice


```
void Bezier::drawBezier (QPainter * painter){
    //Misure the time for calculate the curve
    QElapsedTimer timer;
    timer.start();

    //If  the degree's curve is least 3, do nothing
    int numOfPoints = this->pointOfControlPolygon.length();
    if (numOfPoints < 2) return;

    //Draw the control's polygon
    for(int i=0; i< this->pointOfControlPolygon.length() ; i++){
       //Show info about the point
       painter->setPen(this->verticesPolygonColor);
       painter->setBrush(this->verticesPolygonColor);
       painter->drawEllipse(this->pointOfControlPolygon[i], 3, 3);
       painter->drawText(this->pointOfControlPolygon[i].x()-10,
                         this->pointOfControlPolygon[i].y()-10, "P"+ QString::number(i+1));

       if(i > 0){
           painter->setPen(this->sidePolygonColor);
           painter->drawLine(this->pointOfControlPolygon[i-1], this->pointOfControlPolygon[i]);
       }
    }
```

D2Bèzier
codice

```
  //Evalute Bezier polinomial and draw the curve
  QList<QPointF> auxList;

  QPointF p0,p1;
  painter->setRenderHint(QPainter::Antialiasing);

  painter->setPen(this->bezierCurve);
  p0 = (QPointF) this->pointOfControlPolygon[0];
  for (int i = 1 ; i <= 1000 ; i++) {
    double t = (double)i/1000.0;
    auxList =  QList<QPointF>(this->pointOfControlPolygon);
    p1 = evalBezier(auxList, t);
    painter->drawLine(p0, p1);
    p0 = p1;
  }

  //Read the time elapsed for calculate the curve
  this->time = ((float)timer.nsecsElapsed())/1000000;
}
```

D2Bèzier
Peculiarità

L'applicazione software mostra informazioni su:

- tempi di calcolo per la costruzione della curva
- Grado della curva disegnata
- Parametro t
- Idea di base dell'algoritmo di De Casteljau.

D2Bèzier
Considerazioni

La realizzazione di questo software, ad uso esclusivamente didattico, permette di migliorare la comprensione e l'importanza di questo argomento ad pubblico con competenze eterogenee

Si è scelto di usare come linguaggio di programmazione il QT/C++, al fine di poter riutilizzare il codice per applicazioni future


CURVE DI BEZIER
CURVE DI BEZIER OVUNQUE

Curve di Bézier

Grazie per l'attenzione!

Daniele Caltabiano e Daniele Contarino

Seminario

Metodi Matematici per l'ottimizzazione

A.A. 2012/2013