Interactive Curve Design

Laura Papaleo
papaleo@disi.unige.it
Department of Computer Science
Università di Genova

Monday, April 22, 2002

Curve Drawing: Introduzione [1/2]

- In generale chi ha che fare con un sistema per il disegno di curve desidera disegnarle senza preoccuparsi di aspetti matematici e formule
- Si ha interesse affinché il lavoro sia soddisfacente e che si possa concludere senza troppa fatica!
- Occorre un sistema: **Intuitivo, Flessibile, Metodo Unificato, Invariante, Efficienza**

Curve Drawing: Introduzione [2/2]

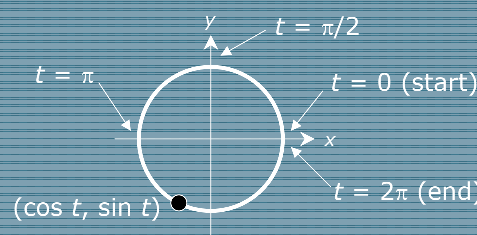
- Si vuole velocità.
- Non si vuole utilizzare troppo spazio.
- L'utente vuole specificare "voglio la curva in questo modo" e deve poterla editare in modo semplice
- Inoltre.
  - Cosa vuol dire che una curva è "smooth"?
  - Esiste una definizione di "good looking" abbastanza precisa da poterci scrivere un programmino?

Splines—Preliminari: Introduzione

- Tutti i requisiti richiesti sono soddisfatti da un tipo di curva detta **spline**.
- Le Splines sono pesantemente utilizzate in computer graphics e computer aided design & manufacturing (CAD/CAM).
- Esistono vari tipi di spline.
  - Discuteremo di Curve di Bézier, B-splines, e NURBS.
- Occorre un certo background matematico per trattare le spline. Vediamo le curve parametriche.

Splines—Preliminari: Curve Parametriche

- Una *Curva parametrica* (2-D) viene espressa come una coppi a di funzioni matematiche: $P(t) = (\ x(t),\ y(t)\ )$, ed un intervallo $[a,b]$ di possibili valori per t.
  - t è il *parametro*.
  - Ad esempio: $x(t) = \cos t$, $y(t) = \sin t$, t in $[0,2\pi]$. Cerchio centrato in (0,0).



Splines—Preliminari: Curve come movimento

- In una curva parametrica il parametro è spesso chiamato "*t*", perché si fa coincidere in molti casi con il concetto di "tempo".
- Si pensi ad esempio ad una particella che viaggia lungo una curva.
  - Il tempo inizia con $t = a$ e termina con $t = b$.
  - La **posizione** della particella al tempo t è dat dai valori di funzione : $(x(t),\ y(t))$.
  - La **velocità** della particella al tempo t è data dalle derivate prime: $(x'(t),\ y'(t))$.
  - L'**accelerazione** della particella al tempo t è invece data dalle derivate seconde : $(x''(t),\ y''(t))$.

Splines—Preliminari: Note - curve parametriche

- Le curve parametriche hanno un certo numero di vantaggi rispetto alle curve "$y=f(x)$", più "standard".
  - *Ogni curva* 2-D può essere descritta come una curva parametrica.
    - *le curve* "$y = f(x)$" sono limitate a quelle che non incontrano una linea verticale più di una singola volta
  - Con le curve parametriche non saremo mai interessati a dy/dx; ogni derivata sarà presa in relazione a *t.* Quindi, ogni linea tangente verticalmente e con slope infinto non presenta alcun problema.
- Altre cose interessanti:
  - Esistono formule molto semplici per i vettori di velocità ed accelerazione.
  - Le curve parametriche permettono anche di utilizzare definizioni semplici di funzioni di cosa vuol dire "**smooth**".

Splines—Preliminari: Smoothness Parametrica

- Vediamo come definire formalmente se una curva è *smooth* o meno.
- Per un intero non negativo k, una curva $P(t)$ è C^k (o "ha C^k continuità" o è "***k-smooth***") nell'intervallo $[a,b]$ se tutta le derivate della curva fino alla k-esima esistono e sono continue in tutti i punti di $[a,b]$.
  - Una curva è C^0 se è continua (entrambe le funzioni sono continue). Informalmente una curva è C^0 se non è "spezzata".
  - Una curva è C^1 se è differenziabile in modo continuo (ovvero se entrambe le funzioni hanno derivate continue). In altre parole una curva è C^1 se è C^0 *e*, quando una particella la percorre, non cambia all'improvviso di velocità.
  - Una curva è C^2 se, oltre ad avere i requisiti di sopra, ha anche le derivate seconde continue. In altre parole una curva è C^2 se è C^1 *e* , quando una particella la attraversa non aumenta o diminuisce in accelerazione.

Splines—Preliminari: Smoothness Geometrica

- Spesso però condizioni matematiche possono non essere "gradite"
- In pratica spesso si vuole che una curva "**sia piacevole a guardarsi**" (*look good*).
- Possiamo anche essere interessati ad avere una forma di smoothness un po' più debole di C^k.
- In questo contesto una curva è ***G^k* continua** se la *direzione* di ognuna delle sue k derivate è continua.
- G^0 come C^0.
- G^1 significa che una particella che si muove lungo la curva non avrà cambiamenti improvvisi di direzione (ma può cambiare di *velocità*).
  - La retta tangente non cambia mai bruscamente.
  - Informalmente, una curva è G^1 se *sembra* C^1.
- G^2 significa che una particella che si muove lungo una curva non cambia in modo brusco in direzione della accelerazione (sebbene possa cambiare accelerazione).

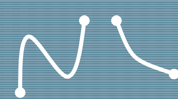
Splines—Preliminari: Curve in Pezzi [1/2]

- In pratica definiamo le curve in pezzi (definite in certi intervalli del parametro t)
- Saremo interessati alla continuità nelle "giunzioni":
  - Utilizzeremo una funzione per, ad esempio, $0 \le t < 1$.
  - Un'altra per $1 \le t < 2$, e così via…
- Perché l'intera curva sia smooth, tali funzioni devono quindi collegarsi in modo accettabile.

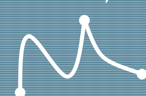
- Per esempio perché una curva sia interamente C^0, le funzioni devono avere lo stesso valore nel punto di giunzione ...
- Perché la curva sia C^1, le funzioni devono avere lo stesso valore e la stessa derivata nel punto di giunzione….

Splines—Preliminari: Curve in Pezzi [2/2]

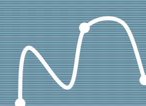
- Ad esempio:
  - Se due funzioni non hanno lo stesso valore nella giunzione la curva risultante non è C^0.



  - Se due funziono hanno lo stesso valore ma non la stessa derivata la curva risultante può essere C^0, ma non è C^1.



  - Se due funziono hanno lo stesso valore e la stessa derivata al punto di giunzione allora la curva risultante è C^1.



Splines—Preliminari: Polinomi

- Si cerca sempre di studiare funzioni polinomiali che descrivano le curve.
  - Esempio: $x(t) = 6.5t^5 - 7.2t^3 + 3t^2 - 2.1t - 5$.
- Le funzioni polinomiali hanno un certo numero di caratteristiche interessanti.
  - Sono facili da descrivere.
    - Servono i coefficienti (sopra: 6.5, 0, -7.2, 3, -2.1, -5).
    - Si può immagazzinare una funzione polinomiale utilizzando pochi double. Mentre altre funzioni possono richiedere strutture più complesse.
  - Il calcolo delle derivate è semplice
    - Es: nx^{n-1}.
  - "facile" = "veloce".
  - Calcolare il valore in un punto è veloce : uniche operazioni che occorre fare sono addizioni sottrazioni e moltiplicazioni.

Splines—Preliminari: Dove vogliamo arrivare?

- Cerchiamo quindi di disegnare curve che:
  - Siano descritte da funzioni polinomiali.
    - O talvolta utilizziamo un polinomio diviso per un altro polinomio (*funzioni razionali*).
  - Siano G^1.
    - *Almeno* G^1; spesso si può ottenere anche C^1 o di più.
  - Possano essere descritte facilmente, dando una serie di punti in cui devono passare (o passarvi vicino).
    - Se siamo in grado di descrivere una curva in questo modo allora possiamo permettere che l'utente la disegni e la editi in modo semplice.

Esempi Smoothness : Determinare C^0 e C^1 [1/2]

- Esercizi:
  - Le seguenti funzioni sono C^0? E C^1?
    - *NOTA: stiamo guardando ad una sola funzione. Quando si trattano curve in 2D, avremo funzioni sia per la x che per la y.*

$$1.\ x(t) = \begin{cases} 1 - t^2, & \text{if } 0 \le t < 1; \\ 2 - t^2, & \text{if } 1 \le t \le 2. \end{cases}$$

$$2.\ x(t) = \begin{cases} 1 - t^2, & \text{if } 0 \le t < 1; \\ t^2 - 4t + 3, & \text{if } 1 \le t \le 2. \end{cases}$$

$$3.\ x(t) = \begin{cases} 1 - t^2, & \text{if } 0 \le t < 1; \\ t^2 - 2t + 1, & \text{if } 1 \le t \le 2. \end{cases}$$

Esempi Smoothness : Determinare C^0 e C^1[2/2]

- Risposte:
  1. Non è C^0 (e quindi non C^1).
  2. È (C^0 e) C^1.
  3. È C^0, ma non è C^1.

Esempi Smoothness: Alcune NOTE

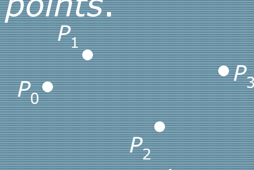
- In esempi reali tratteremo almeno due funzioni.
- Quindi, nel caso 2D, una curva è C^0 se entrambe le funzioni x e y sono C^0;
- Similmente, è C^1 se entrambe x e y funzioni sono C^1.
- Altra caratteristica comune alle spline è che la computazione delle componenti x e y può essere fatta separatamente;
  - Quindi, le curve 3-D (e 4-D e 5-D …) possono essere utilizzate essenzialmente utilizzando le stesse idee del 2-D ma con più funzioni

Interactive Curve Design: Introduzione

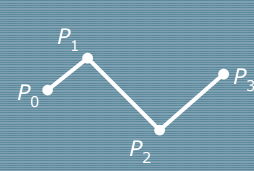
- Una cosa importante riguardo la descrizione di curve "interfaccia utente".
- L'utente non vuole utilizzare formule
- Vuol esprimere concetti "Ho bisogno di una curva più o meno così"
- L'utente vuole poter editare una curva senza incominciare dall'inizio.

Interactive Curve Design: Punti di controllo [1/3]

- Una tecnica di successo per l'interactive curve design è quella di permettere di specificare una curva mediante *control points*.

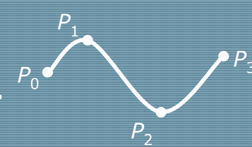


  - Unendo i punti di controllo si ottiene il *control polygon*.

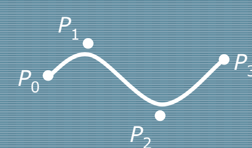


Interactive Curve Design: Punti di controllo [2/3]

- Data una sequenza di punti di controllo possiamo disegnare una curva smooth in due modi:
  - *Interpolando* (curva che passa sui punti di controllo).

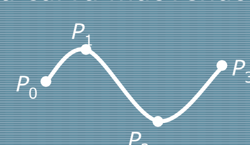


  - *Approssimando* (curva che passa vicino ai punti di controllo).

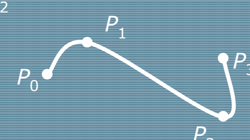


Interactive Curve Design: Punti di controllo [3/3]

- È possibile editare la curva muovendo i punti di controllo (click and drag):
  - Curva originale.



  - Curva dopo il movimento di P_2.



  - L'idea è quindi di
    - Disegnare i punti di controllo.
    - Osservare la curva.
    - Aggiustare i punti di controllo fino al risultato desiderato.

Curve di Bézier: Introduzione

- Bézier Curves furono inventate contemporaneamente da Paul de Casteljau e Pierre Bézier negli anni 60.
- Venivano usate per il design di automobili
- Le curve di Bézier usano un singolo polinomio per **approssimare** un qualsiasi numero di punti di controllo

- Rivediamo un po' di *interpolazione linerare*.

Interpolazione lineare [1/3]

- *L'Interpolazione lineare* (*lirping* [*lerping*] *tweening*) è un semplice modo per recuperare valori tra due valori specificati
  - Esempio, dati $f(0)=4.1$, and $f(1)=4.5$. Un valore ragionevole per $f(0.5)$?
    - 4.1 + 4.5 / 2 =4.3.
    - In assenza di più informazioni 4.3 sembra essere un valore ragionevole per f(0.5).
    - Se in base ai due valori dati *definiamo f per interpolazione lineare* allora $f(0.5)=4.3$.
  - L'interpolazione lineare è una generalizzazione della tecnica qui sopra mostrata.

Interpolazione lineare [2/3]

- Supponiamo *f*(0)=*a*, e *f*(1)=*b*.
- Per definire valori di *f* tra 0 ed 1 per interpolazione lineare, Usiamo la seguente formula:

$$f(t) = (1-t)a + tb, \quad 0 \le t \le 1.$$

  - Usata solo tra 0 ed 1.
  - Ok per valori di *t*=0 or *t*=1.

- Esercizio:
  - Sia *f*(0)=3.2, ed *f*(1)=-1.7. Trovare un valore ragionevole per *f*(0.2) Usando l'interpolazione lineare.

Interpolazione lineare [3/3]

- Risposta:
  - Usando la formula

$$f(0.2) = (1-0.2)\times 3.2 + 0.2 \times (-1.7) = 2.22.$$

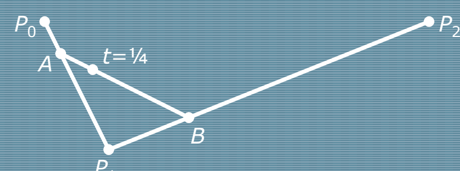
  - Quindi un valore ragionevole è *f*(0.2) = 2.22.

Curve di Bézier: L'Idea [1/3]

- Data una sequenza di punti di controllo (P_0, P_1, P_2), definiamo una funzione polinomiale *f* tale che:
  - $f(0)=P_0$ (il primo punti di controllo).
  - $f(1)=P_2$ (l'ultimo punto di controllo).
  - Per trovare valori intermedi ripetiamo l'operazione di interpolazione lineare
  - Nel caso dei tre punti, dato *t* tra 0 ed 1:
    - Interpoliamo linearmente prima tra P_0 e P_1.
    - Poi tra P_1 e P_2.
    - Alla fine interpoliamo linearmente tra i due valori ottenuti.

Curve di Bézier: L'Idea [2/3]

- Quindi, dati tre punti (P_0, P_1, P_2), se volessimo trovare il punto t=¼ opereremmo come segue.
  - Trovare il punto t=¼ per linear interpolation tra P_0 e P_1 (*A*).
  - Trovare il punto t=¼ per linear interpolation tra P_1 e P_2 (*B*).
  - Trovare il punto t=¼ per linear interpolation tra *A* e *B*.
  - L'ultimo punto è il punto t=¼ sulla curva di Bézier.



Curve di Bézier: L'Idea [3/3]

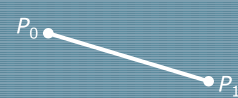
- Questa idea di lirping ripetuto funziona considerando un qualsiasi numero di punti di controllo.
- Ovviamente per più punti di controllo abbiamo bisogno di più livelli di lirping.
  - La funzione risultante è sempre polinomiale e di grado inferiore di uno rispetto al numero di punti di controllo.
    - Esempio: 3 control points (P_0, P_1, P_2), polinomiale di grado 2.

Disegnare Curve di Bézier: Overview

- Vedremo due modi per trattare curve di Bézier:
  - L'algoritmo di **De Casteljau**.
    - Che computa le coordinate dei punti sulla cura usando ripetute interpolazioni lineari
    - Semplice da usare e visualizzare lento per curve complesse.
  - Usando i **polinomi**.
    - Questo rende semplice la computazione di punti sulla curva

Curve di Bézier: De Casteljau Alg. [1/5]

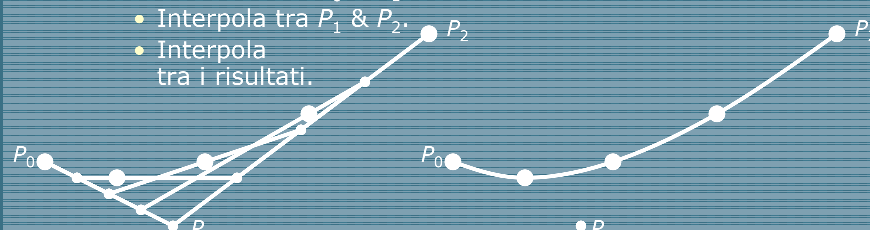
- Paul de Casteljau sviluppò un metodo per disegnare le curve di Bézier che usa **ripetute interpolazioni lineari**.
- Primo caso: 2 punti di controllo (P_0, P_1).
  - $f(0) = P_0$; $f(1) = P_1$.
  - Per trovare il punto assegnato ai valori, interpoli e basta.



  - La risultante curva è un segmento

Curve di Bézier: De Casteljau Alg. [2/5]

- Caso successivo: 3 control points (P_0, P_1, P_2).
  - $f(0) = P_0$ (primo CP); $f(1) = P_2$ (Ultimo CP).
  - Per trovare altri valori di f (per un certo valore t):
    - Interpola tra P_0 & P_1.
    - Interpola tra P_1 & P_2.
    - Interpola tra i risultati.



  - La curva risultante è un segmento di parabola.

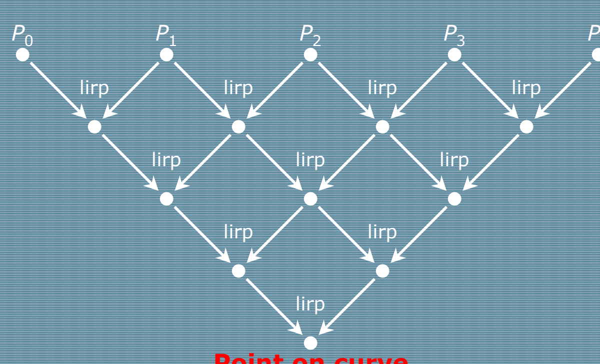
Curve di Bézier: De Casteljau Alg. [3/5]

- Caso dopo : 4 control points (P_0, P_1, P_2, P_3).
  - $f(0) = P_0$ (primo CP.); $f(1) = P_3$ (ultimo CP.).
  - Altri valori (per un certo valore di t):
    - Interpola tra P_0 & P_1, P_1 & P_2, and P_2 & P_3.
    - Interpola tra 1st & 2nd punti trovare e tra 2nd & 3rd.
    - Interpola tra i risultati finali.

  - La risultante curva è un segmento di una curva cubica (grado 3 del polinomio)

Curve di Bézier: De Casteljau Alg. [4/5]

- Per più punti di controllo lo schema è lo stesso



Curve di Bézier: De Casteljau Alg. [5/5]

- Input: p [0:n] di n +1 punti e numero reale u
- Output: punto sulla curva, p(u)

```
for i:=0 to n do
   q[i]:=p[i];
          // Salvataggio dell'input
for k:=1 to n do
   for i:=0 to n-k do
        q[i]:=(1-u)*q[i]+u*q[i+1];
          // applicazione della formula
return q[0];
```


Curve di Bézier: Uso del polinomio [1/3]

- Una curva di Bézier è descritta da un polinomio
  - Dati n control points (P_0, P_1, …, P_{n-1}), La curva di Bézier avrà il polinomio di grado $n-1$.
- Come trovare il polinomio?

Curve di Bézier: Uso del polinomio [2/3]

- Per due punti di controllo (P_0, P_1), il polinomio che descrive la curva di Bézier è $P_0 \cdot (1-t) + P_1 \cdot t$.
  - La formula di interpolazione lineare.
  - In realtà due formule una per la *x*, ed una per la *y (in 2-D)*
- Per tre control points (P_0, P_1 , P_2), il polinomio è $P_0 \cdot (1-t)^2 + P_1 \cdot 2(1-t)t + P_2 \cdot t^2$.
- Per quattro il polinomio è $P_0 \cdot (1-t)^3 + P_1 \cdot 3(1-t)^2 t + P_2 \cdot 3(1-t)t^2 + P_3 \cdot t^3$.

Curve di Bézier: Uso del polinomio [3/3]

- Quindi per 4 CP $P_0 \cdot (1-t)^3 + P_1 \cdot 3(1-t)^2 t + P_2 \cdot 3(1-t)t^2 + P_3 \cdot t^3$.
- Fatta da quattro pezzi detti ***Bernstein polynomials.***
  - $B^3{}_0(t) = (1-t)^3$.
  - $B^3{}_1(t) = 3(1-t)^2 t$.
  - $B^3{}_2(t) = 3(1-t)t^2$.
  - $B^3{}_3(t) = t^3$.

$$B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

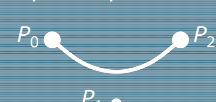
- Pattern?
  - Scende la potenza di (1-*t*).
  - Aumenta quella di *t*.
  - I coefficienti sono detti *binomial coefficients*.

Curve di Bézier: Proprietà [1/6]

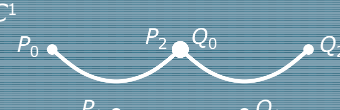
- Good ☺
  - Endpoint interpolation.
  - Unire curve di Bézier smoothly è semplice.
  - Affine invariance.
  - Convex-hull property.
  - Variation diminishing property
- Bad ☹
  - No interpolazione tra control-point (tranne primo e ultimo).
  - Non controllo locale.

Curve di Bézier: Proprietà [2/6]

- Endpoint Interpolation
  - Se curve di Bézier non passano per i punti di controllo fatta eccezione per il punto iniziale e quello finale

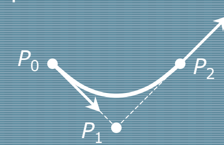


  - Per fare in modo che due curve di Bézier si uniscano basta fare che il primo punto della seconda sia l'ultimo punto della prima.
    - Questo ci dice C^0 ma non C^1

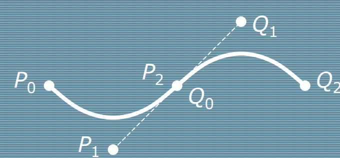


Curve di Bézier: Proprietà [3/6]

- Smooth Joining
  - Curva tangente al segmento tra il primo punto ed il secondo e tra il penultimo punto e l'ultimo



  - Per unire le curve di Bézier con continuità G^1 basta posizionare gli lutimi due punti di controllo della prima ed i primi due della seconda sulla stessa linea.

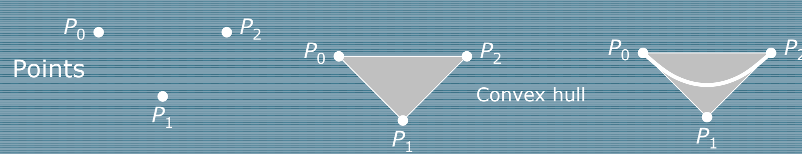


Curve di Bézier: Proprietà [4/6]

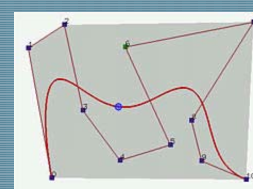
- Affine Invariance
  - Translation.
  - Rotation.
  - Scaling.
  - Shearing.
  - Reflection.
  - Le curve di Bézier hanno la proprietà che applicando una trasformazione ai punti di controllo tale trasformazione viene automaticamente applicata a tutti i punti della curva.
    - Per ruotare la curva di rotate a Bézier, ad esempio, basta applicare una rotazione ai punti di controllo
- Le trasformazioni funzionano come vogliamo.

Curve di Bézier: Proprietà [5/6]

- Convex-Hull Property
  - Il *convex hull* di un insieme di punti è definito come la più piccola regione convezza possibile che li contiene.



  - Una curva di Bézier giace interamente nel guscio convesso dei suoi punti di controllo



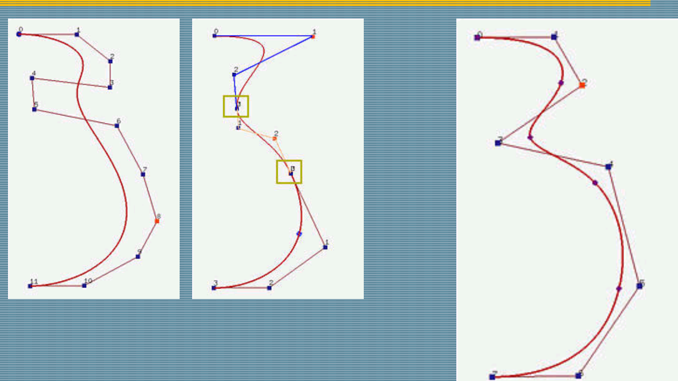
Curve di Bézier: Proprietà [6/6]

- Variation Diminishing Property
  - Se la curva è contenuta in un piano allora non esistono segmenti che intersecano la curva più volte di quanto intersechino la control polyline.

Migliorare le prestazioni [1/5]

- Le curve di Bézier hanno alcune proprietà che non sono desiderabili.
- Per esempio, non permettono interpolazione dei punti di controllo
  - (fatta eccezione per il primo e l'ultimo)
- Un altro problema è che le curve di Bézier mancano di *controllo locale*.
  - Una curva ha *controllo locale* se muovendo un singolo punto di controllo solo un porzione della curva viene influenzata
  - Una curva di Bézier dipende invece da tutti i punti di controllo

Migliorare le prestazioni [2/5]



- B-spline di grado 3 con 8 punti di controllo e 5 segmenti di curva di grado 3 che si uniscono per formarla

Migliorare le prestazioni [3/5]

- In generale, descriviamo una curva in termini di punti di controllo usando le *funzioni di blending*:
  - Una curva parametrizzata

$$P_0 f_0(t) + P_1 f_1(t) + P_2 f_2(t) + P_3 f_3(t)$$

  ha come funzioni di blending f_0, f_1, f_2, f_3.
  - Le funzioni di blending dicono come i control points interagiscono nella creazione della curva.
  - Nel caso delle curve di Bézier le blending function sono i polinomi di Bernstein.

Migliorare le prestazioni [4/5]

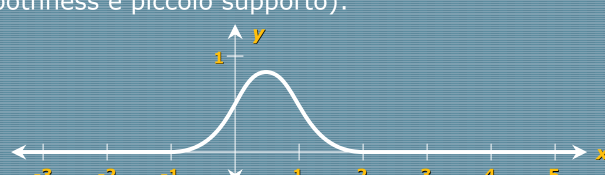
- Se la funzione di blending è zero per un particolare valore di *t*, allora i corrispondenti punti di controllo non influenzano la curva a quel particolare valore di *t*.
- Quindi, per risolvere il problema del controllo locale, vogliamo definire funzioni di controllo che sono zero fatta eccezione per certi valori di *t*.
  - Il range in cui una funzione è non zero è detto **supporto**.
  - È desiderabile avere funzioni con **supporto piccolo** per avere più controllo.

Migliorare le prestazioni [5/5]

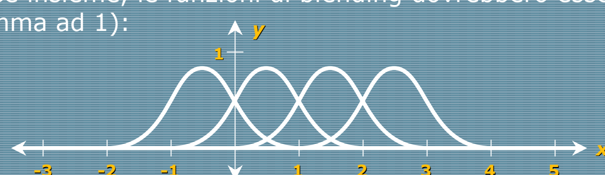
- Altre proprietà per le funzioni di blending:
  - Devono poter essere computate velocemente ed in modo accurato
  - Devono avere "un supporto piccolo" (devono essere zero per la maggior parte dei valori di *t*).
  - **Smooth**.
  - Permettono l'interpolazione dei punti di controllo se voluta.
  - Per ogni valore di *t*, la somma delle blending functions deve essere 1.

Blending Functions: Supporto, Smooth, Somma

- Le funzioni di blending devono avere la seguente forma… (smoothness e piccolo supporto):



- Messe insieme, le funzioni di blending dovrebbero essere così (Somma ad 1):

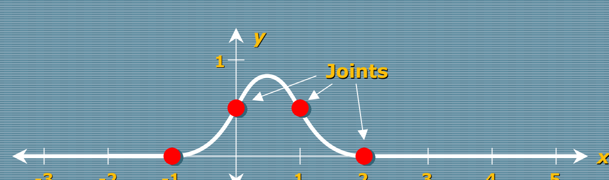


Blending Functions: Polinomi & Ordine

- Definiamo le funzioni di blending come **funzioni polinomiali a pezzi**.
  - Ovvero le funzioni saranno definite in pezzi ed ogni pezzo sarà un polinomio.
  - Tali funzioni hanno un *ordine (m)*: un intero positivo.
    - Il grado di un polinomio sarà *m-1*.
    - La smoothness (C^k) sarà *m-2*.
    - Esempio: Blending functions di ordine 4 saranno piecewise polynomial di grado 3 con smoothness C^2.
    - **NOTA**: Saranno spesso polinomi di basso grado, ☺

Un po' di terminologia

- Definiremo le funzioni unendo i polinomi.



- I punti in cui i polinomi si incontrano vengono chiamati ***joints***.
- I valori del parametro (*t*) in corrispondenza dei joints sono detti ***knots***.
  - Esempio knots *t* = -1, 0, 1, 2.
- Se i knots sono equidistanti la curva è detta ***uniforme***.

B-Splines: Introduzione

- La seconda tipologia di curve che tratteremo prende il nome di *B-splines*.
  - "B" sta per "basis".
  - B-splines sviluppate negli anni '70.
    - Ahuja, Catmull & Rom, Clark, Coons, Cox, deBoor, Forest et al...
  - B-splines sono computate utilizzando le *basis functions*.
    - Le B-spline basis functions sono funzioni di blending che seguono il ragionamento che abbiamo fatto.

B-Splines: Il caso semplice

- Le B-splines possono essere definite con qualsiasi valore di knots ci piaccia
- Consideriamo il caso di particolari valori di knots:

$$t = 0, 1, 2, 3, \ldots$$

  - B-Spline Uniforme… la formula è facilmente generalizzabile.

B-spline: Le Funzioni Base

- Le funzioni base di Bézier sono usate come **pesi** nelle definizioni delle curve.
  - Le funzioni base B-spline saranno usate nello stesso modo.
- Sia U un insieme di m+1 numeri reali non decrescenti, $u_0 \leq u_1 \leq u_2 \leq \ldots \leq u_m$,
- ogni u_i è detto **knot**,
- l'nsieme U è detto **Knot vector** e
- l'intervallo semi-aperto $[u_i, u_i+1)$ è chiamato the **i-th knot span**
  - Si noti che alcune u_i possono essere uguali e per questo alcuni intervalli essere nulli.

B-Spline: Molteplicità di un nodo

- Se un nodo u_i compare **k** volte, con $k>1$, è detto **nodo multiplo di molteplicità k**, u_i (k);
- Altrimenti, è detto **nodo semplice**.
- Se i nodi sono equidistanti, il vettore dei nodi *U* è detto **uniforme**.

B-Splines:
Basis Functions

- Supponiamo di avere $L+1$ punti di controllo (relativi agli L knots): P_0, P_1, P_2, …, P_L.
- Vogliamo definire le basis function (blending function) una per ogni punto di controllo.
- Le basis functions per una B-spline di ordine m saranno

$$N_{0,m}(t),\ N_{1,m}(t),\ N_{2,m}(t),\ \ldots, N_{L,m}(t).$$

- E la curva sarà parametrizzata nel seguente modo

$$P_0 N_{0,m}(t) + P_1 N_{1,m}(t) + P_2 N_{2,m}(t) + \cdots + P_L N_{L,m}(t).$$

B-Splines:
Definire le Basis Functions

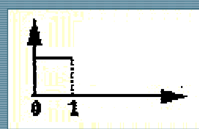
- Definiamo le B-spline basis functions, secondo la formula di formula di **Cox-de Boor**.
  - Iniziamo da quelle di ordine 1 (polinomi di grado 0, senza smoothness):

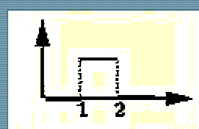
$$N_{k,1}(t) = \begin{cases} 1, & \text{if } k < t \le k+1; \\ 0, & \text{otherwise.} \end{cases}$$

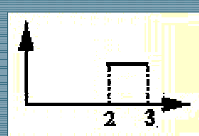
  - Definiamo le basis function di ordine superiore in modo ricorsivo;
    Per $m > 1$ (grado m-1, smoothness C^{m-2}):

$$N_{k,m}(t) = \frac{t-k}{m-1} N_{k,m-1}(t) + \frac{k+m-t}{m-1} N_{k+1,m-1}(t).$$

B-Spline: Definizione Basis Function



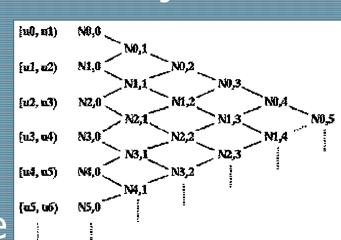




- Se il grado è zero (*m*=0), sono tutte funzioni step.
- La unzione base $N_{i,0}(u)$ è 1 se u è in [ui,ui+1).
- Dati 4 knots $u_0=0$, $u_1=1$, $u_2=2$ ed $u_3=3$:
  - i knot span sono [0,1), [1,2), [2,3)
  - le funzioni di base di grado 0 sono:
    - $N_{0.0}$ (u) = 1 in [0.1) e 0 altrove,
    - $N_{1.0}$ (u) = 1 in [1.2) e 0 altrove e
    - $N_{2.0}$(u) = 1 in [2.3) e 0 altrove.

B-Spline: De Cast. e Fun. Base [1/3]

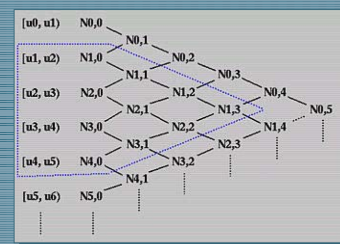
- Schema triangolare di De Casteljau:
  - Gli span sulla 1° col.
  - Tutte le fun. base di grado zero sulla seconda.



- Per computare $N_{i,1}$(u), devono essere utilizzate $N_{i,0}$(u) e $N_{i+1,0}$(u).
  - Gli $N_{i,1}(u)$ vengono scritti nella terza colonna.
- Questo processo continua fino a che tutti gli $N_{i,p}$ (u) sono stati computati.

B-Spline: De Cast. e Fun. Base [2/3]

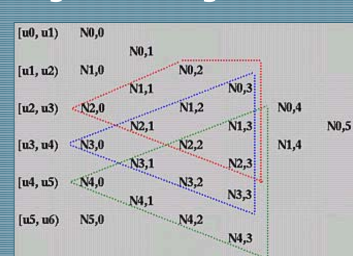
- $N_{i,1}(u)$ viene computato con $N_{i,0}(u)$ ed $N_{i+1,0}(u)$
- È non zero in $[u_i, u_{i+2})$



- **La funzione di base $N_{i,p}(u)$ è non-zero in $[u_i,u_{i+p+1})$**

B-Spline: De Cast. e Fun. Base [3/3]

- Dato un intervallo semi-aperto $[u_i,u_{i+1})$
  - per sapere quali funzioni di base sono non nulle in esso applicare la regola triangolare in senso inverso
  - Nell'intervallo $[u_i,u_{i+1})$, al più p+1 funzioni di grado p sono non nulle:
    - $N_{i-p,\,p}(u)$, $N_{i-p+1,p}(u)$, $N_{i-p+2,p}(u)$, …, $N_{i-1,p}(u)$ $N_{i,p}(u)$.



B-Spline: Definizione della curva

- Dati n+1 punti di controllo $\mathbf{p}_0, \mathbf{p}_1 \ldots, \mathbf{p}_n$
- Dato un **knot vector** $U=\{u_0,u_1\ldots,u_m\}$,
- la curva **B-Spline** di **grado p** è:

$$\mathbf{p}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{p}_i$$

- dove $N_{i,p}(u)$ è funzioni base B-Spline di grado **p**.
  - Equazione simile a quella di Bézier.
  - Una curva di B-Spline necessita di più informazioni

B-Spline: Osservazioni

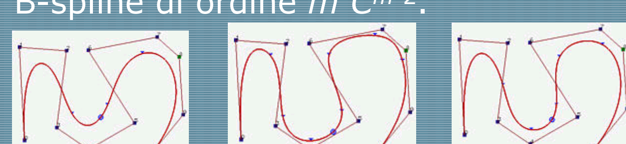
- $N_{i,p}(u)$ assomiglia a $B_{n,i}(u)$, ma
- Il grado delle funzioni base B-Spline è un input,
- mentre il grado delle funzioni base Bézier dipende dal numero di punti di controllo.
- Per modificare una B-Spline, posso
  - modificare le posizione dei punti di controllo
  - le posizioni dei nodi
  - il grado della curva.

B-Spline: Proprietà [1/6]

- Good
  - Piecewise polynomial (degree, smoothness) ☺
  - Affine Invariance ☺
  - STRONG Convex-Hull Property ☺
  - Local Control ☺
  - Interpolation talvolta ☺/☹
- Bad
  - Non posso rappresentare forma rationali ☹

B-Spline: Proprietà [2/6]

- B-splines sono piecewise polynomial. ☺ Più precisamente:
  - Tra due joint consecutivi, viene descritta una B-spline di ordine m da un polinomio di grado m-1.
    - Facili da disegnare più semplici delle curve di Bézier.
  - Una B-spline di ordine m C^{m-2}.

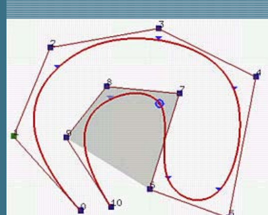


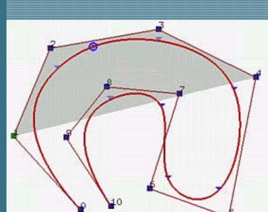
stessa control polyline stessi knots clamped e spaziati uniformamente. grado 7, grado 5 e grado 3

B-Spline: Proprietà [3/6]

- Proprietà di invarianza affine. ☺
  - Applicando una trasformazione affine ai punti di controllo tale trasformazione viene applicata a tutti i punti della curva
  - Ruotando i punti di controllo ruota la curva etc…

B-Spline: Proprietà [4/6]

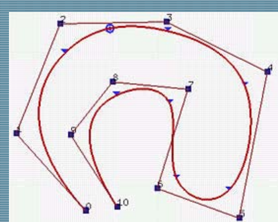


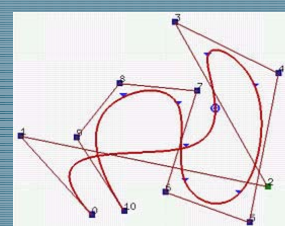


- Le B-splines posseggono la proprietà del convex-hull. ☺
- Anzi hanno una proprietà più forte delle curve di Bézier. ☺☺☺
  - Detto un punto ***active*** se, per quel particolare valore del parametro la funzione di base del control point è non-zero.
  - Ogni punto sulla B-spline giace all'interno del convex hull dei suoi ***active* control points**.

B-Spline:
Proprietà [5/6]

- Le B-splines hanno *controllo locale* ☺☺☺
  - Il movimento di un punto di controllo influenza una piccola porzione della curva.

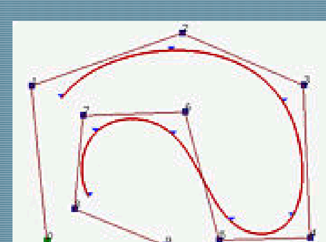




B-Spline:
Proprietà [6/6]

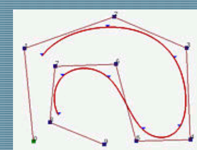
- Le B-splines tipicamente non interpolano alcun punto di controllo. ☹
- Ma utilizzando knot di molteplicità maggiore di uno è possibile ovviare a questo problema. ☺

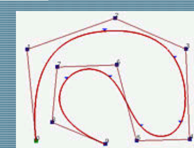
Curva OPEN

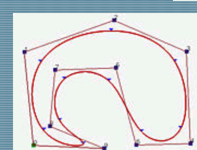


B-Spline:
Tipi di curve

- Una B-spline si dice OPEN se ha vector knot senza struttura
  - la curva non tocca la prima e l'ultima legs
- Se il primo e l'ultimo nodo hanno molteplicità p+1, la curva passa per i punti relativi e sarà di tipo CLAMPED.
- Se il primo e l'ultimo nodo coincidono la curva si dice di tipo CLOSED.







NURBS:
Motivazione

- le curve di B-spline sono curve polinomiali.
- Sono flessibili e piacevoli da manipolare ma non possono rappresentare la curva più semplice: il cerchio.
- Si ha bisogno di una estensione delle B-Spline. **Nascono le NURBS**

NURBS: Introduzione

- "**NURBS**" stands for **N**on-**U**niform **R**ational **B**-**S**pline.
- Sono state sviluppate negli anni '70 come variazione sulla B-splines.

NURBS: Definizione [1/2]

- La differenza essenziale tra le B-spline e le NURBS è il tipo di funzioni utilizzate per descriverle:
  - Le B-splines sono definite usando polinomi.
  - Le NURBS sono definite usando **funzioni razionali**.
- Le NURBS richiedono inoltre informazioni addizionali per la loro definizione
  - Servono i Control points e l'ordine *m* come per le B-splines.
  - Ma inoltre ogni control point nelle NURBS ha un **peso associato**: un numero che ne valuti l' "**importanza**".

NURBS: Definizione [2/2]

- Possiamo usare una notazione analoga alle B-splines:
  - Siano P_0, …, P_L i punti di controllo.
  - Sia t il parametro.
  - Sia m l'ordine delle basis function per P_k ($N_{k,m}$).
  - I knots possono essere numeri arbitrari
- Denotiamo con w_k il peso del punto di controllo k.
- La blending function per una NURBS di ordine m è

$$R_k(t) = \frac{w_k N_{k,m}(t)}{\sum_{k=0}^{L} w_k N_{k,m}(t)}.$$

NURBS: Due Risultati Immediati

- Se tutti i pesi sono uguali a 1, una NURBS riduce ad una B-spline .
- Le NURBS sono razionali.
  - Il valore $N_{i,m}(u)w_i$, è un polinomio di grado m.
  - Il denominatore è la somma di tutti i coefficienti e quindi è un polinomio di grado m
  - Il coefficiente del punto di controllo p_i è il quoziente di due polinomiali di grado p e la funzione p(u) è razionale.
- Le NURBS sono razionali, e quindi possono descrivere circonferenze, ellissi etc…

NURBS: Proprietà [1/5]

- Dati n+1 $p_0, p_1 \ldots, p_n$ punti di controllo, con pesi associati $w_0, \ldots w_n$
- Dato un vettore $U = \{u_0, u_1 \ldots, u_m\}$ di m+1 knots,
- la NURBS di grado *p* è: $$\mathbf{p}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{p}_i$$
  - Con $R_{i,p}(u)$ è definita come segue: $$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^{n} N_{j,p}(u)w_j}$$
  - $R_{i,p}$ (u) è una funzione razionale di grado p in u.

NURBS: Proprietà [2/5]

- **Nonnegativity** - per ogni i, p , $R_{i,p}(u)$ è non negativa.
- **Supporto locale** - $R_{i,p}(u)$ è un non-zero in $[u_i, u_{i+p+1})$
  - Poiché la $N_{i,p}(u)$ è non-zero in $[u_i, u_{i+p+1})$, anche $R_{i,p}(u)$.
- In $[u_i, u_{i+1})$, esistono al più p+1 funzioni base di grado p che siano non-zero, $R_{i-p,p}(u)$, $R_{i+1-p,p}(u)$, $R_{i+2-p,p}(u) \ldots R_{i,p}(u)$.

NURBS: Proprietà [3/5]

- La somma di tutte le funzioni non-zero di grado p in $[u_i, u_{i+1})$ è 1.
  - Se i knots sono m+1, ed il grado delle funzioni di base è p ed il numero di funzioni n +1, allora m =n+p+1.
- In un nodo di molteplicità k, la funzione $R_{i,p}(u)$ è C^{p-k} continua.

NURBS: Proprietà [1/5]

- Una NURBS può essere open, clumped e closed.
- La NURBS p(u) è composizione di curve razionali ognuno di grado p.
  - m = n+ p + 1 deve essere soddisfatta.
- Una curva clumped NURBS passa per p_0 e p_n.

NURBS: Proprietà [1/5]

- **Strong convex hull property**…
- **Local modification scheme**:
  - cambiare la posizione di un punto di controllo p_i interessa soltanto la curva $p(u)$ in $[u_i,u_{i+p+1})$.
- **Variation Diminishing Property**…

NURBS: Proprietà [1/5]

- le curve B-spline e di Bézier sono casi speciali delle NURBS
  - Se tutti i pesi sono uguali, una NURBS si trasforma in una B-Spline.
  - Se n=p la NURBS si riduce ad una curva di Bézier.
- **Projective Invariance**
  Se una trasformazione proiettiva viene applicata ad una NURBS, il risultato può essere costruito dalle immagini proiettate dei relativi punti di controllo.

Fine.

Laura Papaleo
papaleo@disi.unige.it