ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Corso di Laurea Magistrale in
Ingegneria Informatica

Disegno di curve di Bézier

LAB 1

Fondamenti di Computer Graphics M

Cristian Davide Conte 0001034932

Anno Accademico: 2022-2023

Contents

Chapter 1

Algoritmo di de Casteljau

L'applicazione sviluppata permette di inserire dei punti di controllo cliccando all'interno della viewport, di rimuovere il primo di questi tramite il tasto "f" e l'ultimo alla pressione del tasto "l". Una volta inseriti almeno 2 control point è possibile disegnare la corrispondente curva di Bézier tramite l'algoritmo di de Casteljau. Quest'ultimo è un algoritmo iterativo che tramite interpolazione lineare, basata su un parametro "t", tra punti di controllo successivi permette di trovare il punto sulla curva (coordinate (x,y)) corrispondente al valore "t" passato. "t" viene fatto variare tra 0 e 1 ottenendo "MaxNumPtsCurve" (=100) punti della curva che vengono salvati in un array chiamato "CurveArray" che a sua volta viene utilizzato per poi disegnare la curva tramite OpenGL.
È la funzione "drawScene" che si occupa di effettuare i controlli sul numero di control point, sulle modalità di rendering e che, all'aggiunta/rimozione/modifica di nuovi control point, fa il render della scena.

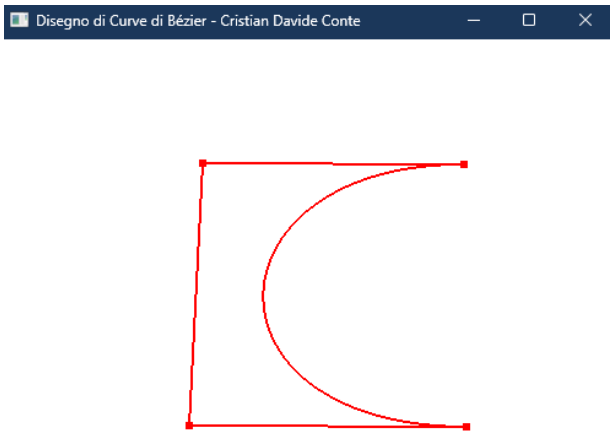

Figure 1.1: Curva di Bézier - Algoritmo di de Casteljau

Chapter 2

Modifica posizione punti di controllo

Per permettere la modifica della posizione dei punti di controllo della curva ci si appoggia alle API OpenGL; in particolare si utilizzano le due funzioni:

```
1    glutMouseFunc(myMouseFunc);
2    glutMotionFunc(myMotionFunc);
```

"glutMouseFunc" permette di rispondere alla pressione di un tasto del mouse con la funzione di callback "myMouseFunc" fornendo dettagli su quale tasto sia stato effettivamente premuto. "glutMotionFunc" permette di rispondere ad un drag-event tramite il callback "myMotionFunc" fornendo informazioni sulla nuova posizione del mouse.

A questo punto la modifica della posizione dei punti di controllo viene implementata tramite ricerca del control point che è il 2d-nearest-neighbour del punto in cui era il mouse alla pressione del tasto sinistro ("searchPointInPointArray"). Se nessun NN viene trovato, allora un nuovo punto di controllo viene aggiunto. Altrimenti un flag booleano viene attivato e non appena si verifica un drag-event le coordinate del punto trovato al passo precedente vengono aggiornate con quelle correnti del mouse.

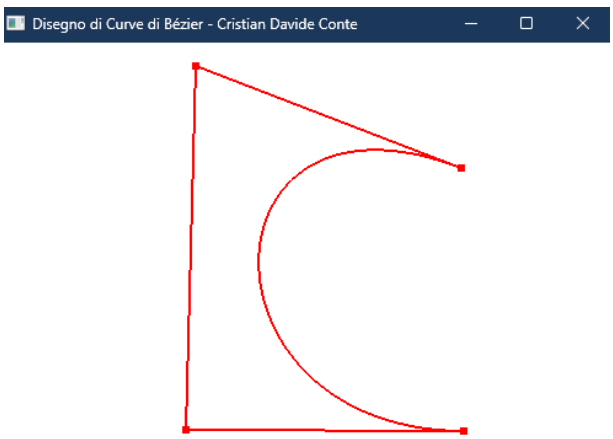

Figure 2.1: Curva di Bézier - Punto di controllo spostato

Chapter 3

Catmull-Rom Spline

Le spline Catmull-Rom sono curve di Bézier interpolanti, ciò vuol dire che passano per i punti di controllo al posto di approssimarne la posizione. Le curve di Bézier hanno la proprietà di interpolare il primo e l'ultimo punto di controllo approssimando gli altri: le curve di Bézier interpolanti a tratti di Catmull-Rom sfruttando questa proprietà creando dei control point artificiali tra due punti di controllo inseriti dall'utente. L'effetto finale è quello di una curva che passa per tutti i punti specificati dall'utente.
I punti di controllo aggiuntivi vengono creati a partire da quelli dell'utente approssimando la loro posizione tramite la definizione di derivata qualora possibile (il primo e l'ultimo punto non hanno control point precedenti/successivi per cui non è possibile applicare esattamente la definizione, ma si ripiega su differenze forward/backward al posto di quelle centrali per il parametro "m").

I punti di controllo aggiuntivi sono date da queste formule:

P_i = ...
P_i^+ = P_i + $\frac{1}{3}$ * m
P_{i+1}^- = P_{i+1} + $\frac{1}{3}$ * m
P_{i+1} = ...
m = $\frac{P_{i+1}-P_{i-1}}{2}$ //Se possibile

Dal punto di vista implementativo, i punti di controllo aggiuntivi vengono inseriti in un array ("PaintPointArray") separato da quello dei punti di controllo inseriti dall'utente ("PointArray"). I punti di controllo possono essere mostrati/nascosti alla pressione del tasto destro del mouse.

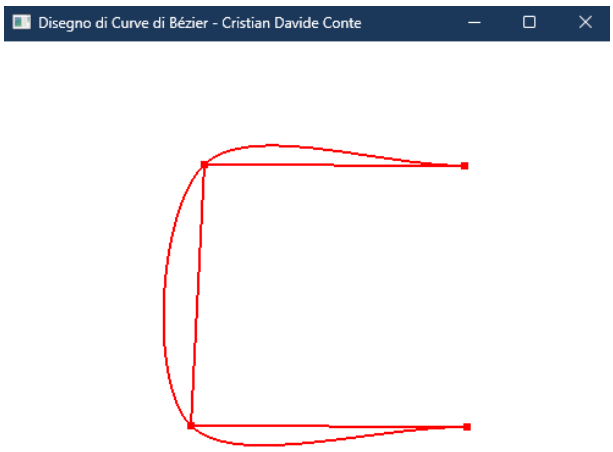

Figure 3.1: Curva di Bézier - Catmull-Rom

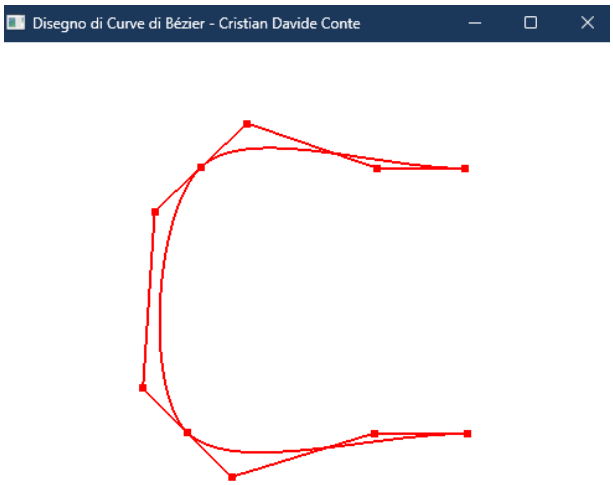

Figure 3.2: Curva di Bézier - Catmull-Rom + CP

Chapter 4

Adaptive Subdivision

L'algoritmo di suddivisione adattiva permette di utilizzare iterativamente l'algoritmo di de Casteljau per suddividere di volta in volta la curva in sottocurve più piccole e più flat. L'obiettivo è quello di arrivare ad una suddivisione tale per cui la sottocurva i-esima è approssimabile ad un segmento. Come condizione di stop viene utilizzata una variabile "threshold" e la distanza tra i control point interni e la retta passante per quelli esterni: se la distanza punto-retta è sempre minore della threshold allora l'algoritmo può disegnare il segmento corrispondente. Se la condizione di planarità non venisse soddisfatta, l'algoritmo procederebbe ricorsivamente con una nuova suddivisione. La threshold può essere dinamicamente modificata tramite rotellina del mouse.
La parte ricorsiva dell'algoritmo può essere vista come l'esplorazione di un binary-tree (depth-first-search) in cui ogni nodo contiene un'approssimazione più o meno giusta del tratto di curva corrente: più si scende nell'albero più l'approssimazione diventa affidabile/corretta. L'ordine delle chiamate ricorsive dell'algoritmo di Adaptive Subdivision può invece essere visto come la strategia di esplorazione dell'albero (left-root-right se si procede prima con la suddivisione sinistra della curva, altrimenti right-root-left). Questo sarebbe utile nel caso si volesse implementare una cache per velocizzare i calcoli.
Al fine di semplificare l'inserimento dei punti della curva (estremi dei segmenti trovati), si è scelto di procedere chiamando ricorsivamente l'algoritmo di Adaptive Subdivision sulla sottocurva sinistra, poi su quella destra.

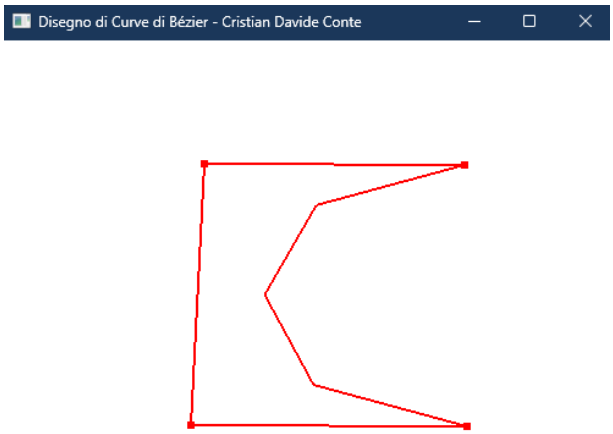

Figure 4.1: Curva di Bézier - Adaptive Subdivision soglia alta

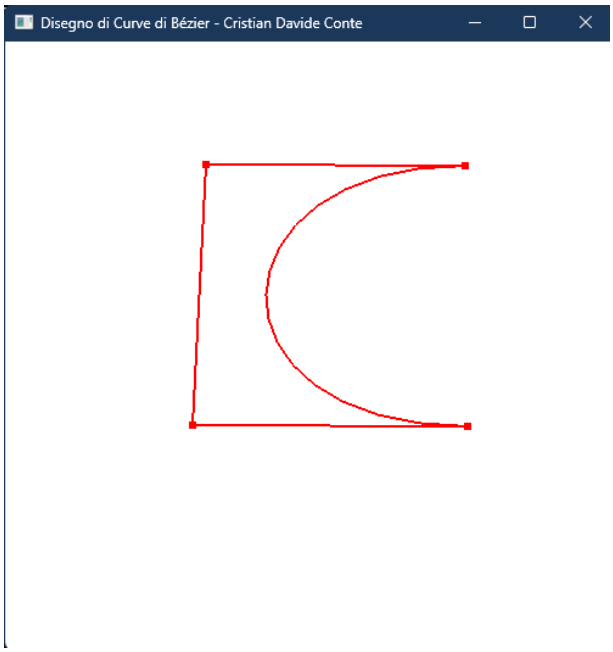

Figure 4.2: Curva di Bézier - Adaptive Subdivision soglia bassa

Chapter 5

Curve a tratti con scelta di continuità dei raccordi

Al fine di implementare la scelta del tipo di continuità dei raccordi tra i tratti della curva di Bézier, si è riutilizzato l'algoritmo di Catmull-Rom modificando il modo in cui il parametro "m" viene calcolato.
Nella nuova implementazione infatti si fa riferimento all'implementazione delle spline TCB basate su tre parametri: tensione (varia la lunghezza delle tangenti nei punti di controllo), continuity (varia la forma della curva), bias (sposta la posizione del punto di controllo lungo la curva, modificando i "pesi" che hanno le due tangenti). L'idea è quella di "spezzare" i vettori tangenti nei punti di raccordo in due vettori: left-tangent ("L") e right-tangent ("R").
I punti di controllo aggiuntivi sono quindi dati da:

$$P_i = \dots$$
$$P_i^+ = P_i + \frac{1}{3} * L_i$$
$$P_{i+1}^- = P_{i+1} - \frac{1}{3} * R_i$$
$$P_{i+1} = \dots$$
$$L_i = \frac{(1-t-a)(1-c)(1+b)}{2} * (P_i - P_{i-1}) + \frac{(1-t-a)(1+c)(1-b)}{2} * (P_{i+1} - P_i)$$
$$R_i = \frac{(1-t+a)(1+c)(1+b)}{2} * (P_i - P_{i-1}) + \frac{(1-t+a)(1-c)(1-b)}{2} * (P_{i+1} - P_i)$$

Il parametro "a" ("alpha" nel codice) permette di modificare la lunghezza dei vettori tangenti L e R in modo disomogeneo (utile per continuità G1).

È possibile cambiare tipo di continuità dei raccordi tramite i tasti: "t" (C0), "c" (C1) e "g" (G1).

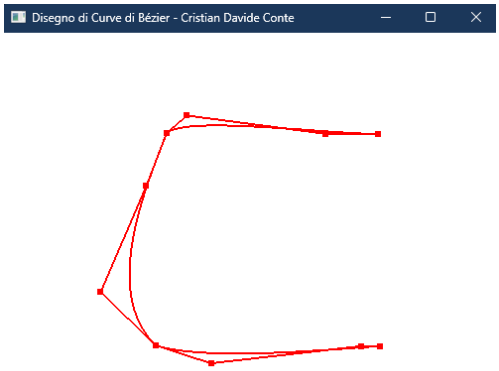

Figure 5.1: Curva di Bézier - Continuità C0

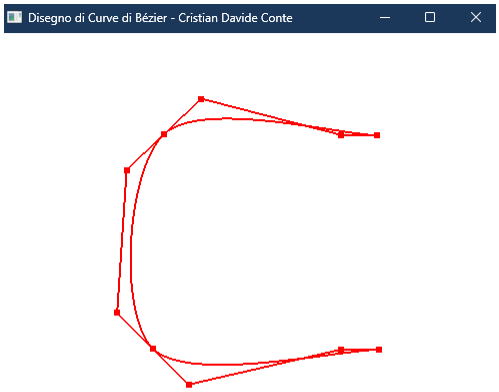

Figure 5.2: Curva di Bézier - Continuità C1

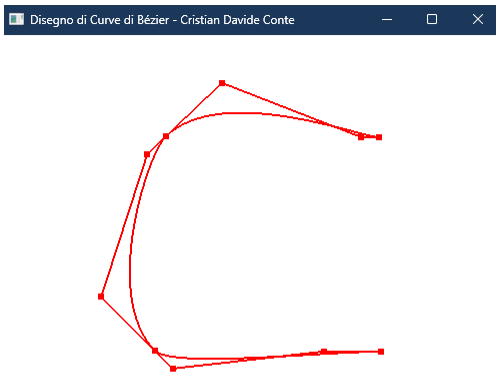

Figure 5.3: Curva di Bézier - Continuità G1