Esercitazione 1 - Disegno di curve di Bézier

Lorenzo Gasparini
Corso di fondamenti di Computer Graphics

2 settembre 2019

1 Descrizione

L'applicazione consente all'utente di disegnare una curva di Bézier inserendo punti in una finestra tramite il click del mouse e rimuovendoli tramite il taso f ed l. Inoltre l'utente può selezionare il tipo di algoritmo da utilizzare per il render scegliendolo tra i 3 disponibili.

2 Obiettivi

1. Compilare e far girare il programma. Provare i controlli da keyboard. Il left mouse button aggiunge un punto. I comandi 'f' e 'l' rimuovono il primo e l'ultimo punto dalla lista di punti, rispettivamente. Oltre ai 64 punti, i primi punti sono rimossi.

2. Osservare come il programma usa le OpenGL GLUT callback per catturare gli eventi click del mouse e determinare le posizioni (x, y) relative.

3. Provare a cambiare lo stile di punti e linee.

4. Disegnare la curva di Bezier a partire dai punti di controllo inseriti, utilizzando l'evaluator di OpenGL (glMap1f(), glMapGrid1f(), glEvalMesh1()). Ricordarsi di abilitare il disegno di curve con glEnable(GL MAP1 VERTEX3).

5. Sostituire alle routine di OpenGL il disegno della curva mediante algoritmo di de Casteljau.

6. Integrare nel programma in alternativa uno dei seguenti punti:

   (a) disegno di una curva di Bezier mediante algoritmo ottimizzato basato sulla suddivisione adattiva.

   (b) disegno interattivo di una curva di Bezier composta da tratti cubici, dove ogni tratto viene raccordato con il successivo con continuità C0, C1 o G1 a seconda della scelta utente da keyboard.

7. Permettere la modifica della posizione dei punti di controllo tramite trascinamento con il mouse.

3 Svolgimento Obiettivi

3. Per variare la grandezza della linea basta cambiare il valore dentro glLineWidth presente nella funzione display. Per variare invece il colore della linea bisogna cambiare i valori di glColor3f presente nella funzione evaluateBezier.

   Per cambiare la gradezza dei punti bisogna variare glPointSize presente all'interno della funzione initRendering mentre il colore dei punti nel glColor3f posizionato subito prima del glBegin(GL POINTS) dentro la funzione display.

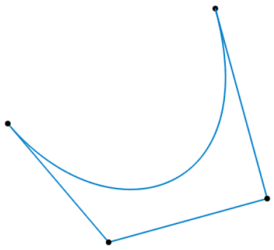

Figura 1: Curva più fine.

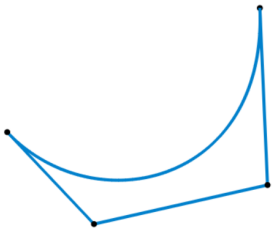

Figura 2: Curva più spessa.

4. Il codice è presente all'interno della funzione evaluateBezier. Tramite la funzione glMap1f definisco la funzione di Bézier e gli passo tutti i punti di controllo ed il grado della funzione.

   All'interno dello switch seleziono il metodo di rendering da utilizzare.

   L'evaluator di OpenGL corrisponde al valore 0. Tramite la funzione glEvalCoord1f si valuta il punto della funzione al parametro i/t; i è l'index attuale del for mentre t è il numero totale di punti di valutazione.

   Il valore di t è modificabile tramite i tasti '+' e '-', tale valore cambia la precisione della curva.

   Tale algoritmo riesce ad interpolare un numero massimo di 10 punti.

5. L'algoritmo di deCasteljau è stato implementato nella maniera classica. Si effettuano una serie di lerp atte ad interpolare i vari punti in maniera iterativa fino ad arrivare al punto finale.

   Dati ad esempio 4 punti il primo ciclo genera un set di 3 punti interpolati dai primi 4, al secondo si passa ad un set di 2 punti ed al terzo si ottiene un singolo punto. Le interpolazioni sono effettuate alla posizione indicata dal parametro t passato come parametro.

   L'interpolazione (LERP) è svolta come segue: LERP(t,a,b) = (1-t)a + tb

   Una volta ottenuto il vertice finale questo viene stampato tramite glVertex3f.

   La funzione viene chiamata allo stesso modo del valutatore di OpenGL, come vantaggio ha il fatto che non è limitato ad un massimo di 10 punti.

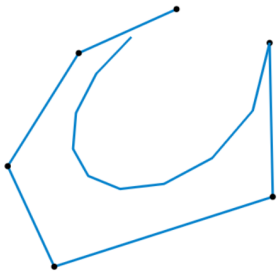

Figura 3: Curva con meno punti di controllo.

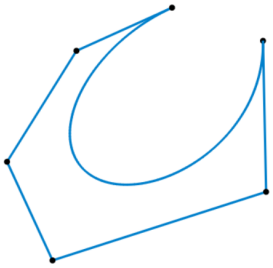

Figura 4: Curva con più punti di controllo.

6.a. L'algoritmo di adaptiveSubdivision è implementato all'interno della funzione adaptiveSubdivision. Tale algoritmo prende in input la serie di punti ed una tolleranza. Questo algoritmo è ricorsivo e si divide in due fasi:

- Nella prima fase si scorrono tutti i punti di controllo e si valuta se la distanza dei punti interni (cioè tutti tranne il primo e l'ultimo) hanno una distanza dalla linea formata dal primo e l'ultimo punto maggiore della tolleranza passata in input. Nel caso abbiano una distanza maggiore si pone una variabile di controllo drawLine a false.
- Nella seconda fase si controlla la variabile drawLine. Se tale variabile è true significa che la distanza di tutti i punti e minore della tolleranza e quindi possiamo approssimare la curva con la linea tra il primo e l'ultimo punto.

  In caso contrario dobbiamo suddividere il set di punti in due nuovi set con stessa lunghezza.

  Il meccanismo di suddivisione è molto semplice, praticamente è un passaggio dell'intepolazione di deCasteljau con parametro 0.5. Ad ogni passata però salviamo il primo e l'ultimo punto del set generato all'interno di due liste.

  Alla fine avremo due set in cui su uno saranno presenti tutti i primi valori e sull'altro avremo tutti gli ultimi valori.

  Generati questi due set si procede ricorsivamente su entrambi.

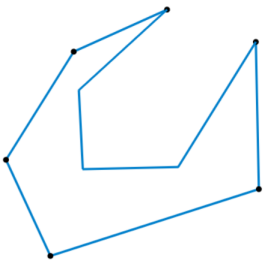

Figura 5: Curva con tolleranza maggiore.

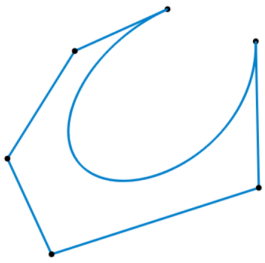

Figura 6: Curva con tolleranza minore.

7. Per questo punto sono state definite le due funzioni motionMouse e passiveMouse

MotionMouse praticamente è una funzione collegata all'evento di openGL di spostamento del mouse durante un click prolungato e viene richiamata ogni volta si verifica una situazione di quel genere.

Per il resto la funzione in se è molto semplice poichè, nel caso la variabile booleana isMovingPoint sia true, sposta il punto indicato dall'indice movingPoint alla posizione del mouse.

PassiveMouse, in maniera speculare a motionMouse, è collegata all'evento di openGL di spostamento del mouse ma questa volta quando non sta avvenendo alcun click.

La funzione, anch'essa molto semplice, scorre tutti i punti di controllo e calcola la distanza che c'è tra il punto ed il cursore. Se tale distanza è minore di un certo valore imposta il parametro mouseOverIndex al valore dell'indice del punto di controllo (In pratica salva il valore del punto di controllo sopra il quale è posizionato il cursore su una variabile)

Nel caso in cui non è sopra alcun punto salva -1 come valore.

Il resto del meccanismo è gestito dalla myMouseFunc che è la funzione di gestione del mouse. Semplicemente quando rileviamo un click con il tasto sinistro andiamo a controllare se il valore di mouseOverIndex è diverso da -1 (quindi siamo sopra un punto). In quel caso si imposta isMovingPoint a true e si salva l'indice del punto (presente dentro mouseOverIndex) dentro movingPoint così da passare le informazioni alla funzione motionMouse.