Esercitazione 1

Stefano Romanazzi

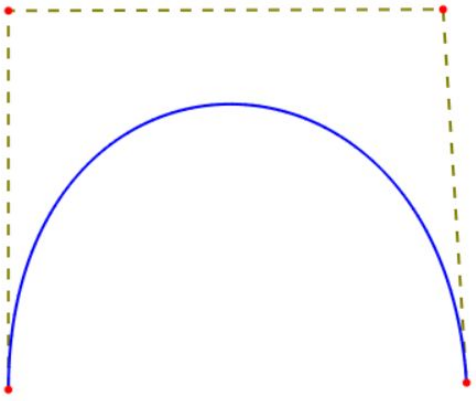
Student ID: 819445

E-mail: stefan.romanazzi@gmail.com

Panoramica

L'esecuzione del programma è avviata tramite l'inizializzazione di varie caratteristiche visive della scena e dei punti e delle linee coinvolti nell'esercitazione. Ciò avviene tramite la funzione *initRendering*, che permette di decidere lo spessore delle linee e dei punti tramite le direttive *glLineWidth* e *glPointSize*. Successivamente viene ridotto l'effetto di aliasing e i punti sono resi circolari.

Altri dettagli sull'aspetto delle linee e dei punti generati sono decisi nulla funzione *Display* dove vengono impostati tramite le *glBegin*. I punti sono disegnati in rosso, le linee tratteggiate che li congiungono in verde e le curve di Bézier nelle tre differenti modalità sono disegnate in blu.



Le linee tratteggiate sono rappresentate tramite l'abilitazione del valore di OpenGL *GL_LINE_STIPPLE*, che consente successivamente di controllarne la frequenza dei tratteggi tramite la funzione *glLineStipple,* il cui valore è stato impostato a 0.5. Aumentando tale valore è possibile ridurre la frequenza dei tratteggi, che saranno più lunghi e in numero inferiore. L'abilitazione del parametro di OpenGL *GL_LINE_STRIP* invece, consente di tracciare in automatico linee a due a due tra i vertici processati, pertanto ciò avviene attraverso la funzione glVertex2f, alla quale sono passati tutti i punti inseriti tramite mouse.

Tali vertici sono memorizzati nel vettore *CV*, avente come dimensioni 64 (il numero massimo di punti che il programma consente di visualizzare) e 3 (le tre componenti dello spazio x, y, z), per un totale di 64□3.

Nella fase successiva viene applicata una delle tre modalità previste dal programma, che saranno spiegate successivamente. Infine, in corrispondenza dei vertici memorizzati nel vettore *CV*, vengono disegnati i punti con l'aspetto deciso nella funzione di inizializzazione.

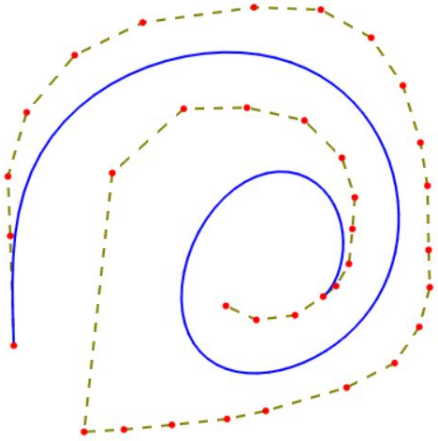
Funzioni di servizio

Le funzioni di servizio aggiunte al programma sono poche, ma essenziali per spiegare le tre modalità previste dal programma per tracciare le curve di Bézier.

Funzione	Tipo di ritorno	Parametri	Descrizione
lerp	float	float x0 float x1 float t	Esegue l'interpolazione lineare tra le coordinate x0 e y0, in funzione del parametro t.
flatTest	bool	float dist	Test utilizzato nella versione adattiva dell'algoritmo di de Casteljau, verifica se la distanza passatagli è inferiore alla soglia massima 0.001.
getDistancePL	float	float outX float outY float x2 float y2 float x3 float y3	Calcola la distanza tra la retta definita dai punti 2 e 3, e il punto esterno *out*. I parametri in input sono le coordinate di tutti i punti in gioco.

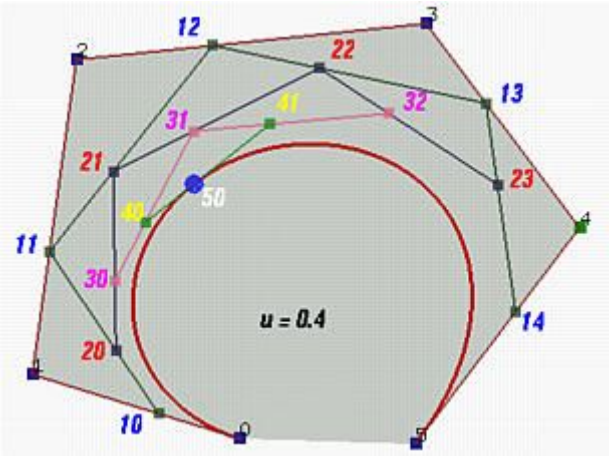
Modalità 1: Bézier tramite direttive OpenGL

La maniera più semplice ed immediata per disegnare una curva di Bézier è usando le direttive fornite da OpenGL. Questa modalità, sebbene soggetta a limitazioni, consiste solo in una fase di inizializzazione in cui bisogna abilitare la modalità tramite il codice GL_MAP1_VERTEX_3 (apposito per i vertici in 3D) e nella successiva chiamata della funzione glMap1f con la quale si mappano i vertici da associare come punti di controllo della curva e altri parametri secondari relativi alla curva. Dopo l'inizializzazione è possibile passare alla fase di disegno della curva, ciclando su un insieme di valori da valutare. Tali valori, se connessi tramite linee, daranno origine al disegno vero e proprio della curva di Bézier. La curva di Bézier tuttavia si fermerà ad un certo grado, quello massimo supportato da OpenGL, richiedendo un'implementazione specifica tramite algoritmo di de Casteljau.



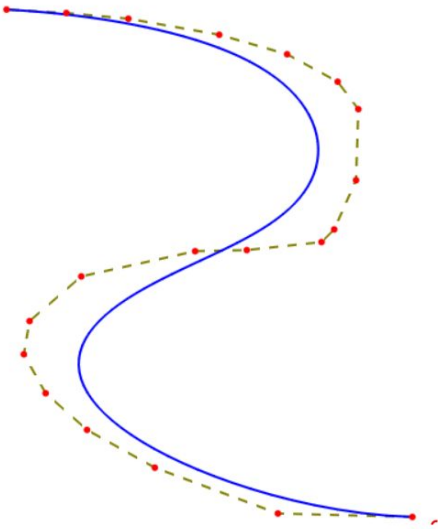
Modalità 2: implementazione dell'algoritmo di de Casteljau

L'algoritmo di de Casteljau sfrutta l'interpolazione lineare tra due punti per valutare la curva. La funzione *lerp* calcola un punto compreso tra due punti in input al variare di un parametro t. Tale parametro viene dunque impiegato per suddividere i segmenti analizzati in proporzione al suo valore, nel dominio [0, 1]. Ad ogni iterazione si considera un differente poligono di controllo formato dai segmenti su cui sarà applicata la lerp. Il primo poligono di controllo è definito dai punti di controllo veri e propri. Nelle iterazioni successive si ottiene il nuovo poligono di controllo ottenuto congiungendo con dei segmenti i punti ottenuto dall'applicazione della *lerp* sui segmenti dell'iterazione precedente.



L'algoritmo può essere dunque scritto ricorsivamente.

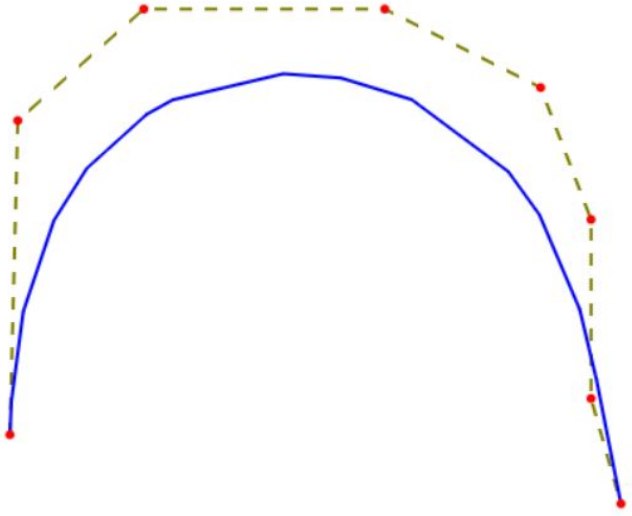
Come risultato della conversione della formula in algoritmo si ottiene dunque la curva disegnata in maniera piuttosto chiara e poco sgranata, richiamando la funzione *de_casteljau* 100 volte con parametro *t* differente e variabile nell'intervallo [0, 1].



Modalità 3: de Casteljau adattivo

La versione adattiva dell'algoritmo di de Casteljau consiste nel suddividere ricorsivamente il poligono di controllo con la relativa curva in due parti, tramite la tecnica divide-et-impera, consentendo di applicare de Casteljau separatamente fino al soddisfacimento del flat test. Tale test verifica se la distanza di tutti i punti di controllo del poligono considerato hanno una distanza dalla corda tracciata tra il primo e l'ultimo punto di controllo del poligono, che sia inferiore ad una soglia prefissata. Il flat test è eseguito dalla funzione *flatTest* e la distanza tra ogni punto e la corda è calcolata servendosi della funzione *getDistancePL* La soglia è stata fissata a *0.001* e l'algoritmo procede con la suddivisione ricorsiva della curva fino al soddisfacimento del flat test lungo tutto il poligono di controllo. Si occupa di questa procedura la funzione *adaptiveDecasteljau.* Qui di seguito sono rappresentati due screenshot rispettivamente con una soglia bassa ed una alta, per evidenziarne gli effetti sull'aspetto della curva.

Soglia del flat test → 0.01



Soglia del flat test → 0.001