LAB 1 – Introduzione ad OpenGL e Curve di Bézier

Erika Gardini – Ingegneria Informatica A.A. 2016/2017

Per prima cosa è stato compilato ed eseguito il programma e sono stati testati i comandi già forniti. Poi, sono state osserve le OpenGL GLUT callback per catturare gli eventi click del mouse e determinare le posizioni (x,y) relative.

In particolare, la funzione:

```
void mouse (int button, int state, int x, int y);
```

restituisce il tipo di bottone del mouse (tasto destro, testo sinistro, …), lo stato (premuto, rilasciato, …) e la posizione x ed y al momento dell'evento.
Quando viene effettuato un click con il tasto sinistro del mouse, questo viene rilevato grazie alla seguente condizione:

```
if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
```

Una volta rilevata la pressione del tasto sinistro del mouse, vengono calcolate le coordinate x ed y del punto cliccato e viene disegnato un nuovo punto nella finestra.
Per attivare la ricezione degli eventi click del mouse è necessario inserire nel main il seguente comando:

```
glutMouseFunc (mouse);
```

Prima di procedere con le funzionalità da aggiungere all'applicazione, si è provato a modificare lo stile dei punti e delle linee. In particolare, il seguente comando consente di definire il colore della linea che congiunge i punti (nell'applicazione la linea è rosa):

```
glColor3f (1.0f, 0.0f, 0.8f);
```

Figura 1: glColor3f(0.0f, 0.0f, 0.0f);

Figura 2: glColor3f(1.0f, 0.0f, 0.8f);

Il comando di seguito riportato consente di definire lo stile della linea (nell'applicazione tratteggiata, `GL_LINE_STRIP` per la linea normale).

```
glEnable (GL_LINE_STIPPLE);
```

Figura 3: glEnable (GL_LINE_STIPPLE);

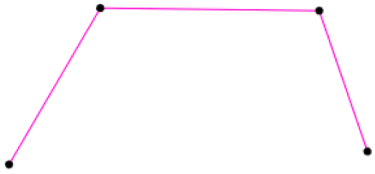


Figura 4: glEnable (GL_LINE_STRIP);

Infine, il seguente comando consente di definire lo stile della linea tratteggiata:

```
glLineStipple (1, 0x00FF);
```

Figura 5: glStipple (3, 0x00FF);

Figura 6: glStipple (1, 0x00FF);

Una volta analizzate le funzionalità fornite dall'applicazione si è potuto procedere con la realizzazione delle funzionalità richieste.
La prima di esse richiedeva di disegnare la curva di Bézier, a partire dai punti di controllo inseriti, utilizzando l'evaluator di OpenGL. Per attivare la seguente funzionalità è stato necessario aggiungere i seguenti comandi:

```
glEnable(GL_MAP1_VERTEX_3);
glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, numCV, &CV[0][0]);

for(i=0; i < 100; i++){
    glEvalCoord1f((GLfloat) i / 100.0);
}
```

In questo modo, dati i punti di controllo, viene disegnata automaticamente una curva di Bézier.

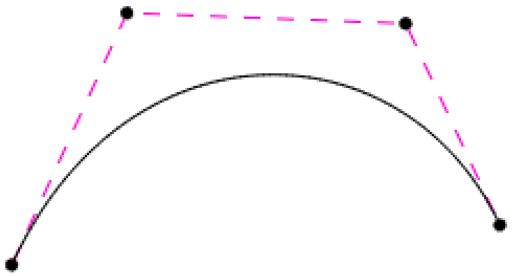


Figura 7: curva di Bézier con evaluator OpenGL

Problema: l'evaluator di OpenGL riesce a disegnare la curva solo con un numero limitato di punti di controllo. Superato il limite, la curva non viene più disegnata.

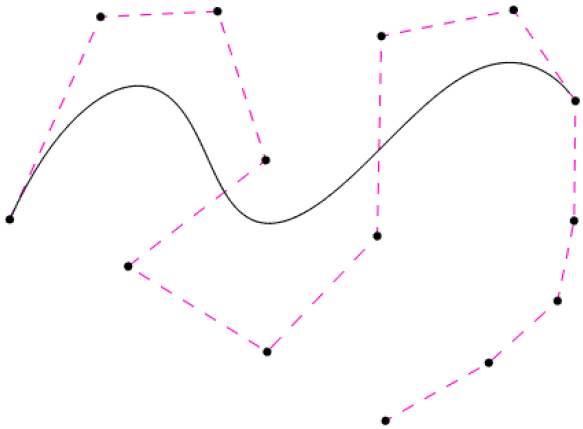


Figura 8: problema di OpenGL evaluator

Una possibile soluzione al problema è quella di sostituire la routine di OpenGL con l'algoritmo di De Casteljau. L'algoritmo di De Casteljau si basa sul ripetere iterativamente l'operazione di interpolazione lineare fra due punti in base ad un parametro *t* passato come argomento.
In particolare, l'algoritmo calcola per ciascun segmento (unione dei punti di controllo) un punto, ottenuto come segue:

$$Lerp(t, a, b) = (1 - t) \cdot a + t \cdot b$$

Il seguente calcolo viene riapplicato ai nuovi punti trovati fino ad ottenere un unico punto finale, che sarà un punto della curva di Bézier.
Ripetendo l'algoritmo per valori di *t* differenti è possibile ottenere l'intera curva di Bézier.
Il numero dei passi svolti dall'algoritmo di De Casteljau è pari al grado della curva e quindi pari a *n-1* (dove *n* è il numero di punti di controllo).
La curva di Bézier ottenuta interpola solo il primo e l'ultimo punto di controllo.

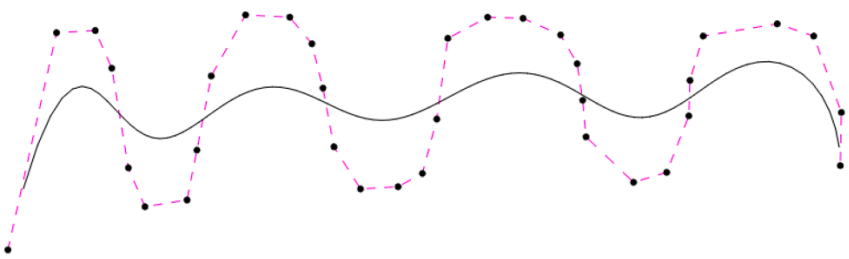


Figura 9: curva di Bézier con algoritmo di De Cateljau

L'algoritmo di De Casteljau consente di realizzare una curva di Bézier tanto precisa quanto maggiore è il numero di ripetizioni dell'algoritmo stesso (alto numero di valori *t*).

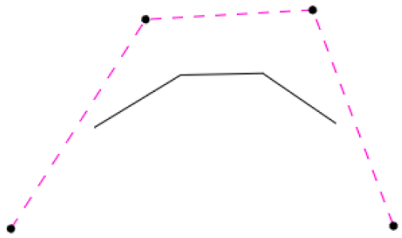


Figura 10: De Casteljau con 5 differenti valori di t

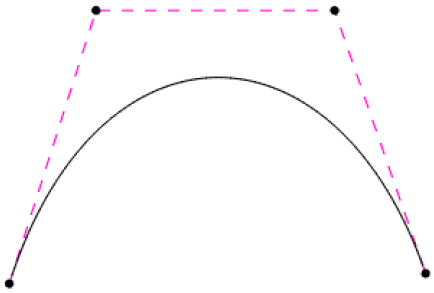
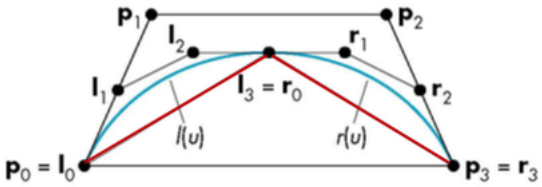


Figura 11: De Casteljau con 100 differenti valori di t

Problema: il numero di ripetizioni dell'algoritmo di De Casteljau non tiene conto dell'aspetto dei punti di controllo e potrebbe essere "costoso inutilmente". Infatti, se i punti di controllo fossero tutti allineati, non sarebbe necessario ripete l'algoritmo per molti valori del parametro *t*, la curva sarebbe una linea e per disegnarla basterebbe unire fra loro i punti di controllo.

Figura 12: curva di Bézier con punti di controllo allineati

Il problema può essere risolto utilizzando un algoritmo ottimizzato basato sulla suddivisione adattativa. L'algoritmo sfrutta le funzionalità di De Casteljau, ma si basa su un Flat Test. In particolare, si ripete l'algoritmo solo finché non si è raggiunta la tolleranza desiderata.
Per tolleranza si intende la distanza del punto calcolato dal segmento che unisce i punti di controllo.



Terminata la realizzazione delle funzionalità per disegnare le curve di Bézier è stato possibile procedere con la realizzazione dell'ultima richiesta: consentire la modifica della posizione dei punti di controllo tramite trascinamento con il mouse.
Per aggiungere la funzionalità si sono estese le funzionalità del mouse già presenti; in particolare, alla pressione del click destro del mouse, se la posizione x ed y rilevata dalla callback di OpenGL si trova nell'intorno di uno dei punti di controllo, questo deve essere selezionato e spostato nel punto di rilascio del mouse. A tal scopo è stato necessario attivare un'altra callback:

```
glutMotionFunc(mouseMotion);
```

Questa funzionalità consente di rilevare gli spostamenti del mouse e restituisce ad ogni spostamento le coordinate x ed y relative alla posizione del mouse. Quindi, se il tasto destro del mouse è premuto "vicino" ad un punto di controllo, il movimento del mouse viene rilevato, il punto segue lo spostamento effettuato ed al momento del rilascio le sue coordinate verrano aggiornate con quelle al momento del rilascio.
La curva di Bèzier ottenuta mediante l'algoritmo adattivo si modifica automaticamente in base alla nuova posizione del punto di controllo.

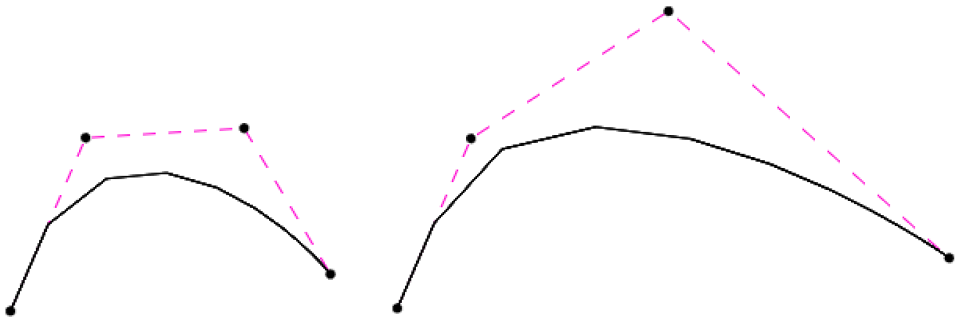


Figura 13: spostamento degli ultimi due punti

Problema: poiché la curva di Bézier interpola solo il primo e l'ultimo punto di controllo e poiché il grado del polinomio è elevato, la curva di Bézier ottenuta non è fedele ai punti di controllo; inoltre, allo spostamento di uno di essi la curva di Beziér si modifica totalmente e non solo nella zona vicina al punto modificato.
Per risolvere questo problema è necessario realizzare una curva con un polinomio di grado molto basso.