Esercitazione 1 - curve di Bézier

OpenGL è una libreria che permette di scrivere applicazioni che trattano grafica bidimensionale e tridimensionale. Il concetto di base è quello delle callback, ovvero funzioni definite dallo sviluppatore che vengono invocate al momento opportuno (ad esempio in risposta al movimento del mouse sulla finestra, o alla pressione di un tasto del mouse).

Punto 1
Osservazioni:

- il *main* inizializza OpenGL e collega le callback necessarie al corretto funzionamento dell'applicazione
- i punti di controllo della curva di Bézier sono rappresentati da un array bidimensionale (un numero intero indica il numero di punti di controllo attualmente inseriti)
- si possono effettuare tre operazioni:
    - click sinistro sulla finestra: permette di aggiungere un punto di controllo sulla finestra (oltre i 64 punti, i primi punti sono rimossi)
    - pressione del tasto 'f' della tastiera: permette di rimuovere il primo punto di controllo inserito
    - pressione del tasto 'l' della tastiera: permette di rimuovere l'ultimo punto di controllo inserito
- la funzione *keyboard()* permette di rilevare la pressione di un tasto della tastiera

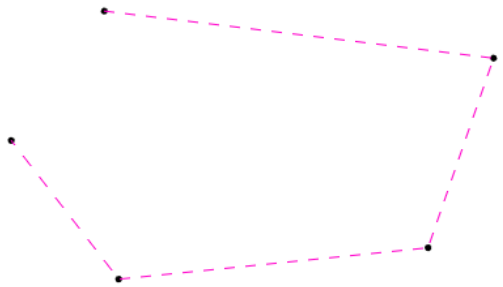

Figura 1 - Esempio di finestra dopo l'inserimento di cinque punti

Punto 2
La callback utilizzata da OpenGL per catturare gli eventi click del mouse sulla finestra è la seguente:

```
void mouse(int button, int state, int x, int y) { ... }
```

dove:

- *button* rappresenta il tipo di click
    - GLUT_LEFT_BUTTON: click con il tasto sinistro del mouse
    - GLUT_MIDDLE_BUTTON: click con il tasto centrale del mouse
    - GLUT_RIGHT_BUTTON: click con il tasto destro del mouse
- *state* rappresenta lo stato del pulsante del mouse
    - GLUT_DOWN: pressione di un tasto del mouse
    - GLUT_UP: rilascio di un tasto del mouse
- *x* e *y* rappresentano la posizione in cui è avvenuto il click. Tale posizione è espressa in pixel a partire dall'angolo in alto a sinistra della finestra ([0, 0]x[WindowWidth, WindowHeight]).
  OSSERVAZIONE: OpenGL calcola la posizione dei punti a partire dall'angolo in basso a sinistra della finestra; inoltre, *x* e *y* devono essere compresi nell'intervallo [0, 1]. Pertanto, si rende necessaria una conversione di coordinate prima di inserire il punto di controllo nel relativo array.

```
void evaluatePositions(int x, int y, float * xPos, float * yPos, float * zPos) {
  (* xPos) = ((float) x) / ((float) (WindowWidth - 1));
  (* yPos) = ((float) y) / ((float) (WindowHeight - 1));
  (* yPos) = 1.0f - (* yPos); // Flip value since y position is from top row.
  (* zPos) = 0;
}
```

Come si può notare, prima si effettua la divisione di *x* e *y* rispettivamente per la larghezza e l'altezza della finestra per portarli nell'intervallo [0, 1]. Successivamente, si esprime la coordinata *y* rispetto al lato inferiore della finestra (si osservi come *x* non abbia bisogna di alcuna conversione).

Punto 3

Tramite OpenGL è possibile modificare lo stile di punti e linee:

- punti:
  - *glPointSize(size)* consente di modificare la dimensione del punto
  - *glColor3f(float R, float G, float B)* consente di modificare il colore del punto
- linee:
  - *glLineWidth(size)* consente di modificare lo spessore della linea
  - *glColor3f(float R, float G, float B)* consente di modificare il colore della linea
  - *glEnable(GL_LINE_STIPPLE)* e *glLineStipple(factor, pattern)* consentono di disegnare una linea tratteggiata
    N.B.: *pattern* è un intero a 16 bit che indica quali pixel della linea devono essere disegnati (bit a 1) e quali no (bit a 0), *factor* rappresenta il numero di volte che viene esaminato ciascun bit del *pattern* prima di passare al successivo.

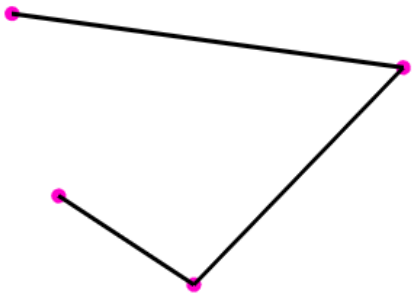

Figura 2 - Esempio di finestra dopo la modifica di stile a punti e linee

Punto 4

Il seguente blocco di codice consente di disegnare la curva di Bézier tramite l'implementazione di OpenGL.

```
glEnable(GL_MAP1_VERTEX_3);
glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, numCV, &CV[0][0]);
glBegin(GL_LINE_STRIP);
glColor3f(0.0f, 0.0f, 0.0f);
for (int i = 0; i <= ITERATIONS; i++) {
  float t = (float) i / 100;
  glEvalCoord1f(t);
}
glEnd();
```

Le prime due istruzioni abilitano il disegno di curve di Bézier tramite l'algoritmo implementato da OpenGL. Si noti come sia necessaria specificare quanti e quali siano i punti di controllo e l'intervallo su cui è definito il parametro *t* (in questo caso nell'intervallo [0, 1]).
Il calcolo dei punti appartenenti alla curva di Bézier viene effettuato tramite la primitiva *glEvalCoord1f()* che necessita del valore attuale del parametro *t*.

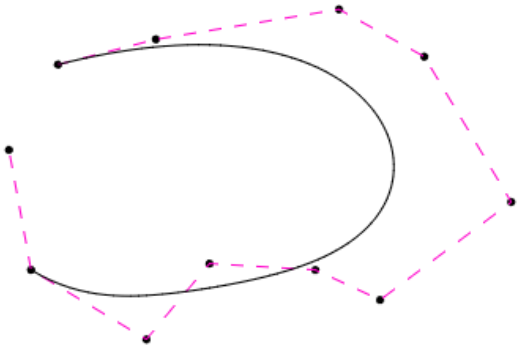

Figura 3 - Curva di Bézier disegnata tramite l'implementazione di default di OpenGL

OSSERVAZIONE: l'implementazione di OpenGL non consente di disegnare curve con più di 10 punti di controllo.

Punto 5

Per superare il problema del numero massimo di punti di controllo imposto da OpenGL il disegno della curva è stato implementato con l'algoritmo di de Casteljau (tasto '1' della tastiera).

L'algoritmo di de Casteljau si basa sul concetto di interpolazione lineare il quale risulta così definito:

$$z = (1 - t)x + ty \quad dove\ t \in [0, 1]$$

Al variare di *t* tra 0 e 1 si ottengono tutti i punti del segmento che congiunge i punti *x* e *y*.

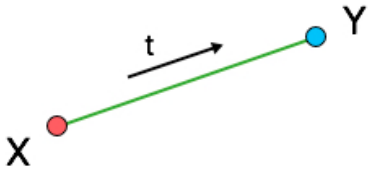

Figura 4 - Interpolazione lineare

<u>Un esempio dell'algoritmo di de Casteljau con 4 punti di controllo</u>.

Per ogni valore di t, l'algoritmo:

- al primo passo calcola l'interpolazione lineare tra i punti di controllo b_i^0. In particolare l'interpolazione tra b_0^0 e b_1^0, tra b_1^0 e b_2^0 e tra b_2^0 e b_3^0
- al secondo passo calcola l'interpolazione lineare tra i punti di controllo b_i^1 calcolati precedentemente. In particolare l'interpolazione tra b_0^1 e b_1^1 e tra b_1^1 e b_2^1
- al terzo passo calcola l'interpolazione lineare tra i punti di controllo b_i^2 e ottiene il valore finale b_0^3

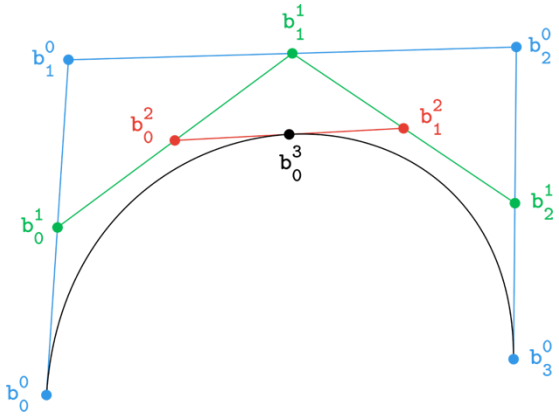

Figura 5 - Esempio di algoritmo di de Casteljau

OSSERVAZIONI: si noti come ad ogni passo il numero di punti su cui si calcola l'interpolazione lineare diminuisca. L'algoritmo termina quando rimane solamente un punto; quest'ultimo rappresenta il punto della curva di Bézier per un determinato valore del parametro *t*.

Di seguito si fornisce l'implementazione dell'algoritmo di de Casteljau (non vincolato ad un determinato numero di punti di controllo).

```
void deCasteljau(float controlPoints[MAX_CV][3], int n, float t, float pointToDraw[3]) {
  float coordX[MAX_CV];
  float coordY[MAX_CV];
  float coordZ[MAX_CV];

  for (int i = 0; i < n; i++) {
    coordX[i] = controlPoints[i][0];
    coordY[i] = controlPoints[i][1];
    coordZ[i] = controlPoints[i][2];
  }

  for (int i = 0; i < (n - 1); i++) {
    for (int j = 0; j < (n - i - 1); j++) {
      coordX[j] = (1 - t) * coordX[j] + t * coordX[j + 1];
      coordY[j] = (1 - t) * coordY[j] + t * coordY[j + 1];
      coordZ[j] = (1 - t) * coordZ[j] + t * coordZ[j + 1];
    }
  }

  pointToDraw[0] = coordX[0];
  pointToDraw[1] = coordY[0];
  pointToDraw[2] = coordZ[0];
}
```

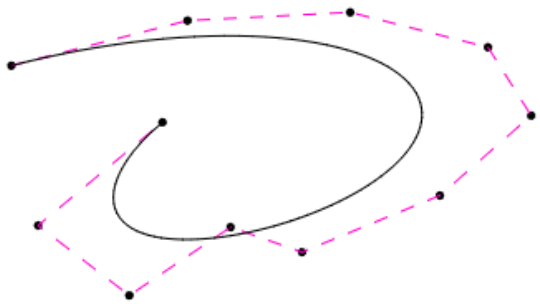

Figura 6 - Curva di Bézier disegnata con algoritmo di de Casteljau

OSSERVAZIONE: modificando il numero di valori t per i quali viene valutata la curva di Bèzier si ottiene un disegno più/meno *smooth*.

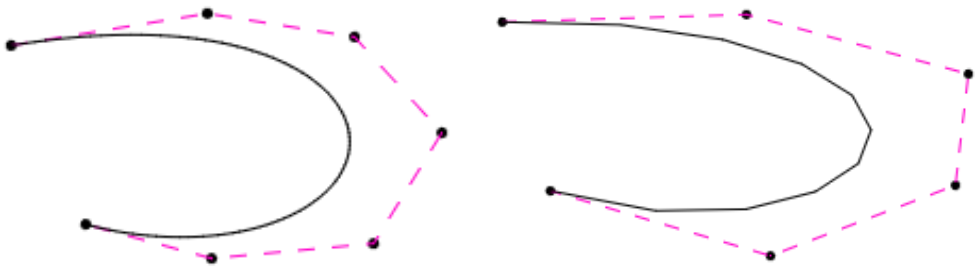

Figura 7 - A sinistra la curva di Bézier è stata valutata per 100 valori di t, a destra per 10 valori di t

Punto 6a

Per ottimizzare il rendering della curva di Bézier (invece che valutare la curva su intervalli fissi come fatto in precedenza), si divide la curva in sotto-curve sempre più piccole fino a quando ogni sotto-curva è sufficientemente simile ad uno segmento; la curva di Bézier è quindi costituita dall'insieme dei segmenti

ottenuti. Questo procedimento viene chiamato *suddivisione adattiva della curva di Bézier* (tasto '2' della tastiera).

Idea fondamentale:

- calcolare per ogni punto di controllo interno (tutti tranne il primo e l'ultimo) la distanza tra il punto di controllo stesso e la corda che congiunge il primo e l'ultimo punto di controllo
  - se la distanza calcolata per ogni punto di controllo è inferiore ad una certa soglia si disegna un segmento che congiunge il primo e l'ultimo punto di controllo
  - altrimenti, si divide la curva in due sotto-curve e si ripete il procedimento per entrambe le sotto-curve

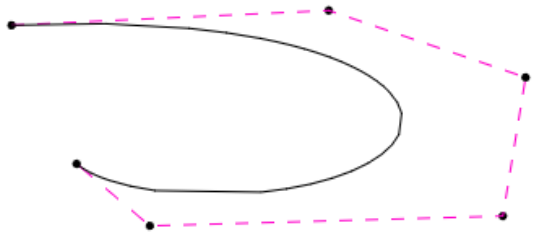

Figura 8 - Curva di Bézier disegnata con algoritmo di suddivisione adattiva

OSSERVAZIONE: modificando il parametro di soglia (*MAX_DISTANCE*) la curva di Bèzier risultante è più/meno *smooth*.

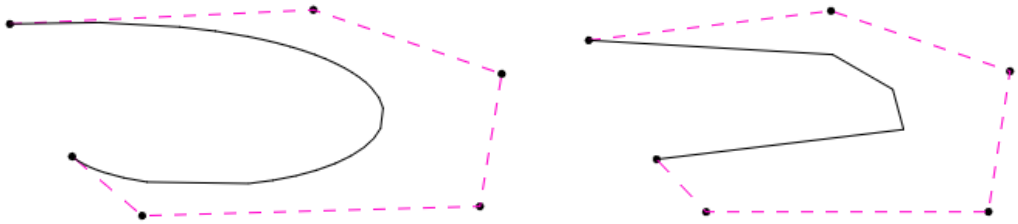

Figura 9 - A sinistra il valore di soglia è 0.01 mentre a destra è 0.05

Punto 7

Il seguente blocco di codice permette di spostare uno dei punti di controllo trascinandolo con il mouse (tenendo premuto il tasto destro del mouse).

```
bool toMove = false;
int pointToMoveIndex = -1;

void mouse(int button, int state, int x, int y) {
  float xPos = -1; float yPos = -1; float zPos = -1;
  evaluatePositions(x, y, &xPos, &yPos, &zPos);

  if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
    for (int i = 0; i < numCV; i++) {
      if ( (CV[i][0] <= xPos + 0.05 && CV[i][0] >= xPos - 0.05) &&
                              (CV[i][1] <= yPos + 0.05 && CV[i][1] >= yPos - 0.05) ) {
        toMove = true;
        pointToMoveIndex = i;
      }
```

```
    }
  }

  if (button == GLUT_RIGHT_BUTTON && state == GLUT_UP) {
    toMove = false;
    pointToMoveIndex = -1;
  }
}

void motion(int x, int y) {
  float xPos = -1; float yPos = -1; float zPos = -1;
  evaluatePositions(x, y, &xPos, &yPos, &zPos);

  if (toMove == true) {
    CV[pointToMoveIndex][0] = xPos;
    CV[pointToMoveIndex][1] = yPos;
    CV[pointToMoveIndex][2] = zPos;

    glutPostRedisplay();
  }
}
```

N.B.: *glutPostRedisplay()* forza il ridisegnamento della finestra.

Alla pressione del tasto destro del mouse, si verifica se il mouse è su uno dei punti di controllo; in tal caso, si memorizza l'indice di quest'ultimo nell'array dei punti di controllo. Inoltre, si abilita un flag che indica alla funzione *motion()* di spostare il punto di controllo nel caso in cui l'utente sposti il mouse tenendo premuto il tasto destro dello stesso.
Al rilascio del tasto destro del mouse, il flag viene disabilitato per indicare la terminazione dello spostamento del punto di controllo precedentemente selezionato.

N.B.: lo spostamento del punto è real-time: la modifica alla curva di Bézier viene percepita istantaneamente dall'utente.

OSSERVAZIONE: il punto viene selezionato se si preme con il tasto destro del mouse in suo intorno (in questo caso grande 0.05).

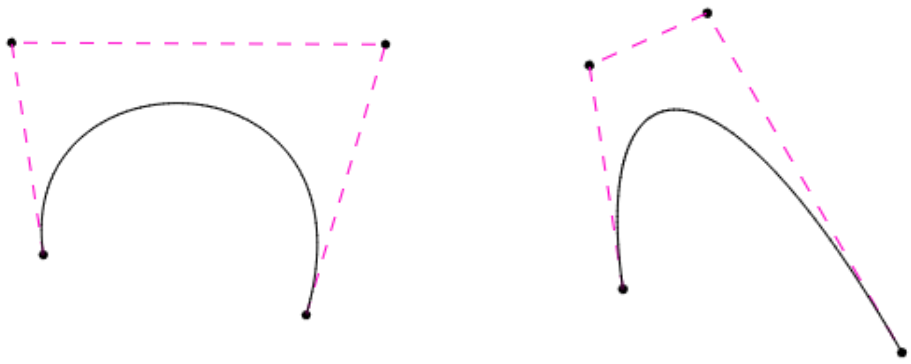

Figura 10 - Spostamento di un punto di controllo