# Grafica

*Anno Accademico 2018-2019*

# Esercitazione 1

Matteo Berti, Matricola 889889

25 giugno 2019

### Comandi da tastiera

| **Tasto** | **Azione** |
|---|---|
| Esc | Terminazione dell'esecuzione |
| f | Rimozione del primo control point |
| l | Rimozione dell'ultimo control point |
| o | Modalità di disegno: OpenGL |
| d | Modalità di disegno: DeCasteljau |
| i | Abilita/disabilita la scelta interattiva della continuità |
| 0 | Continuità $C^0$ [in modalità interattiva] |
| 1 | Continuità $G^1$ [in modalità interattiva] |
| 2 | Continuità $C^1$ [in modalità interattiva] |

### 3. Stile di punti e linee

Il colore dei punti di controllo, dei segmenti che li uniscono e della curva è stato modificato utilizzando la funzione `glColor3f()`, cambiando colore prima che l'elemento venga disegnato tramite `glVertex2f()`.

### 4. Disegnare la curva di Bézier

Prima di tutto è stato abilitato il disegno della curva con la funzione `glEnable(GL_MAP1_VERTEX_3)` e si è settato `glMap1f()`. Infine per disegnare effettivamente la curva si son creati 100 *line strips* che seguono i punti sulla curva e tramite la funzione `glEvalCoord1d()` sono stati valutati e mostrati a schermo.

### 5. Sostituire la routine OpenGL con deCasteljau

L'algoritmo di deCasteljeau viene applicato a 100 punti sulla curva. All'algoritmo viene passato in input un $t \in [0, 1]$ e un *indice* di partenza che indica da quale posizione dell'array contenente tutti i punti di controllo si parte per costruire la curva di (al massimo) 4 punti. Per prima cosa viene copiata la porzione dei punti di controllo per evitare di sovrascriverli, poi si applica l'interpolazione lineare per valutare la curva di Bézier nella sua $t - esima$ porzione. Infine si rappresenta il punto ottenuto aggiungengo un vertice alla *line strip*.

**6. Disegnare interattivamente la curva di Bézier impostando la continuità tra due tratti**
I tipi di continuità offerti sono $C^0$, $G^1$ e $C^1$, di seguito illustrati:

1. $\mathbf{C^0}$: per la continuità $C^0$ non è richiesto nessun tipo di elaborazione, un punto può essere inserito ovunque a prescidere dalla posizione del precedente.

2. $\mathbf{G^1}$: con la continuità $G^1$ quando si vuole inserire un punto $P_k$ è necessario che sia sulla stessa retta che passa per i punti $P_{k-1}$ e $P_{k-2}$, ad una distanza arbitraria. Per prima cosa quando la continuità impostata è $G^1$ si calcola la distanza euclidea tra l'ultimo punto di controllo della curva e il punto cliccato sullo schermo con la funzione `euclideanDistance(A, B)`, in questo modo si ottiene la distanza a cui il nuovo punto sarà posto sulla retta passante per gli ultimi due punti di controllo già presenti. Successivamente con la funzione `getPointInLine(P0, P1, dist, dir)` si calcolano le coordinate del nuovo punto ad una distanza `dist` da `P1` seguento la retta che passa per `P0` e `P1`. Si può anche specificare la direzione se si vuole che il punto sia dopo `P1` o prima.

   Data l'equazione nota della retta passante per due punti:

   $$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

   Si pone nella forma $y = mx + q$ isolando la y:

   $$y = \frac{y_2 - y_1}{x_2 - x_1} \cdot x + \frac{y_1 - y_2}{x_2 - x_1} \cdot x_1 + y_1$$

   In questo modo si sono identificati $m$ e $q$. Per trovare il punto sulla retta alla distanza $d$ si calcola l'intersezione della retta con un cerchio di raggio $d$ e con centro in $(x_2, y_2)$.

   Data la retta $y = m(x - x_2) + y_2$ il cerchio con centro in $(x_2, y_2)$ è dato da $(x - x_2)^2 + (y - y_2)^2 = d^2$, sostituendo la y nell'equazione del cerchio con quella della retta si ottiene: $(x - x_2)^2 + m^2(x - x_2)^2 = d^2$ che può essere fattorizzato come $(x - x_2)^2(1 + m^2) = d^2$, dopo pochi passaggi isolando la x si ha:

   $$x = x_2 \pm \frac{d}{\sqrt{1 + m^2}}$$

   In conclusione, calcolando la distanza euclidea dal punto cliccato su schermo all'ultimo punto di controllo e cercando l'intersezione della retta passante per gli ultimi due punti di controllo con il cerchio di raggio d e centro nell'ultimo punto, si ottiene il nuovo punto desiderato, con continuità $G^1$.

   Di seguito il codice usato per le operazioni sopradescritte.

3. $\mathbf{C^1}$: con la continuità $C^1$ ogni nuovo punto di controllo viene aggiunto dopo all'ultimo punto presente, seguendo sempre la retta passante per l'ultimo e il penultimo punto. In questo caso la distanza a cui è posto è uguale alla distanza tra gli ultimi due punti presenti. Per trovare le coordinate del nuovo punto la procedura è del tutto simile a quella vista per $G^1$, con la differenza che la distanza euclidea qui è calcolata tra l'ultimo punto di controllo presente e il penultimo, invece che tra l'ultimo punto e le coordinate del clic del mouse.

**7. Modificare la posizione dei punti di controllo tramite trascinamento**
Per la modifica dei punti di controllo si parte con la funzione `mouseMotionFunc(x, y)` che rileva le coordinate del puntatore quando si preme il tasto sinistro del mouse sullo schermo e lo si trascina. Qui viene chiamata la funzione `movePoint(clicked, x, y)`, all'interno della quale si distingue in due casi: il punto cliccato è un punto di giunzione tra due curve cubiche distinte, oppure è un

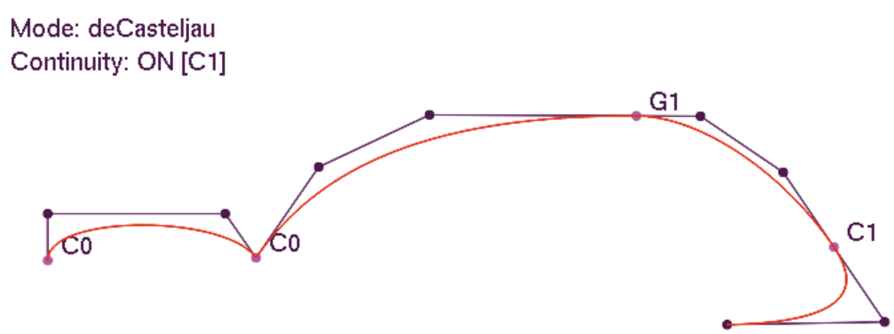
punto "interno" ad una sola curva.

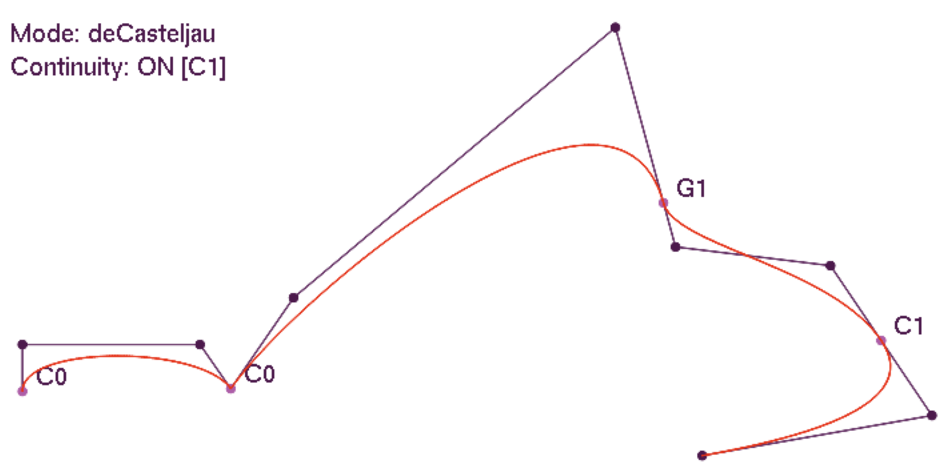
Nel primo caso si adatta la posizione in modo tale che venga mantenuta la continuità impostata:

- $C^0$: il punto viene spostato semplicemente nella posizione cliccata.
- $G^1$: il punto può essere spostato ma è vincolato alla retta passante per il punto precedente e il successivo.
- $C^1$: il punto può essere spostato ma con esso sono spostati in egual misura anche il punto di controllo precedente e quello successivo.

Nel secondo caso, ovvero se si clicca un punto *non* di giunzione, si controlla se il suo successivo o il suo precedente sono punti di giunzione, in caso affermativo:

- $C^0$: se il punto successivo/precedente ha continuità $C^0$ viene lasciato invariato.
- $G^1$: se il punto successivo/precedente ha continuità $G^1$ viene ricalcolata la posizione in modo che mantenga approssimativamente la stessa distanza dal punto di controllo successivo.
- $C^1$: se il punto successivo/precedente ha continuità $C^1$ viene ricalcolata la posizione in modo che rimanga esattamente equidistante dal punto di controllo precedente e dal successivo, a prescindere da quale dei due venga modificato.




Si noti come anche modificando i punti di giunzione la continuità venga preservata.