Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

MiniSystem: un software di Calcolo Numerico

Relatore:
Chiar.mo Prof.
Giulio Casciola

Presentata da:
Francesca Sarti

Sessione III
Anno accademico 2013-2014

Alla mia famiglia

Indice

Elenco delle figure

Introduzione

Questa tesi si occupa del MiniSystem, un software didattico che permette la gestione di curve di Bézier e Bézier a tratti per il disegno 2D assistito dal calcolatore. Nello specifico, l'elaborato descrive il percorso che è stato fatto al fine di introdurre questo software come strumento di supporto e di esercitazione per gli studenti nell' apprendimento del "Calcolo Numerico".
Il percorso si è articolato in tre fasi.

1. La prima fase è stata caratterizzata dalla partecipazione ad alcune lezioni tenute dal professore Giulio Casciola riguardanti la spiegazione sia del MiniSystem sia di alcuni concetti del corso che sono alla base di questo software. Questo mi ha permesso di capire l'approccio con cui gli studenti si pongono davanti alle nozioni spiegate e di riprendere molti degli argomenti legati al codice.

2. La seconda fase è stata incentrata sulla creazione di un percorso di cinque esercitazioni da svolgere con l'ausilio del MiniSystem per un apprendimento graduale delle nozioni del corso. Sono stati sviluppati degli esercizi che permettono agli studenti di apprendere gradualmente i concetti visti a lezione e che, svolti con l'ausilio del software, ne permettono uno studio non solo mnemonico.

3. Nella terza ed ultima fase sono state implementate una serie di nuove funzioni per arricchire il codice del MiniSystem; sono stati implementati una serie di algoritmi di messaggistica con lo scopo sia di aiutare lo studente a muoversi all'interno del software presentato, sia di fargli capire, oltre all'aspetto grafico degli esercizi, anche i concetti matematici alla base di questi. È stata poi implementata la possibilità di compiere operazioni anche su curve di Bézier a tratti; sono stati introdotti nuovi algoritmi per la gestione di quest'ultime e alcuni degli algoritmi preesistenti sono stati modificati affinché risultassero corretti anche per gestire questo tipo di curve.

Nel primo capitolo tratteremo quindi i polinomi di Bernstein, con le importanti proprietà di cui essi godono, le curve di Bézier con i relativi metodi di rappresentazione e le proprietà che le caratterizzano, e le curve di Bézier a tratti.

Nel secondo capitolo verrà mostrato come il software descritto possa risultare d'aiuto per gli studenti, come il computer possa essere un aiuto importante nel processo di apprendimento e come all'interno di esso siano trattati molti degli argomenti del corso di "Calcolo Numerico". Verranno inoltre introdotte le esercitazioni create ed il percorso che mi ha portato alla stesura di questo elaborato.

Nel terzo capitolo verrà analizzato nel dettaglio il codice del MiniSystem sia da un punto di vista strutturale che da un punto di vista grafico; ne verranno mostrate le principali funzioni, le operazioni che è possibile compiere

sulle curve, gli algoritmi inseriti al fine di ottimizzare il codice e le modifiche apportate al programma.

Capitolo 1

POLINOMI DI BERNSTEIN E CURVE DI BÉZIER

I polinomi nella base di Bernstein, dal nome di Sergej Natanovič Bernštejn, sono una base di rappresentazione dei polinomi alternativa alla base delle potenze, meglio condizionata numericamente rispetto a quest' ultima e che gode di numerose proprietà.
Inizialmente Bernstein li introdusse per dimostrare la capacità dei polinomi di approssimare qualsiasi funzione continua su un particolare intervallo e con un arbitrario grado di accuratezza.
Dopo un primo momento di scarsa rilevanza applicativa, ebbero con l' avvento dei computer, una larga applicazione nel design interattivo, cioè con curve parametriche, portando a molti utili algoritmi che ora vengono sempre più adottati in altri campi di applicazione.

1.1 I polinomi di Bernstein

Con polinomio in forma di Bernstein si intende una espressione del tipo

$$p(x) = \sum_{i=0}^{n} b_i B_{i,n}(x) \qquad x \in [a,b] \tag{1.1}$$

dove $B_{i,n}(x)$ sono le funzioni base di Bernstein definite da

$$B_{i,n}(x) = \binom{n}{i} \frac{(b-x)^{n-i}(x-a)^i}{(b-a)^n} \qquad i = 0,..,n$$

e $b_0, b_1, b_2, ..., b_n \in R$ sono i coefficienti di $p(x)$ nella base di Bernstein.

1.1.1 Proprietà dei polinomi di Bernstein

I polinomi di Bernstein godono di proprietà importanti che rendono interessante il loro impiego sia dal punto di vista del controllo geometrico sia da quello dell' affidabilità del calcolo in aritmetica finita.
Elenchiamo nel seguito tali proprietà:

1. *Non negatività.* I polinomi di Bernstein $B_{0,n}(x), ..., B_{n,n}(x)$ assumono valori non negativi:

$$B_{i,n}(x) \geq 0 \qquad \forall x \in [a,b]$$

2. *Partizione dell' unità.* I polinomi di Bernstein $B_{0,n}(x), ..., B_{n,n}(x)$ formano una partizione dell' unità, cioè

$$\sum_{i=0}^{n} B_{i,n}(x) = 1 \qquad \forall x \in [a,b]$$

3. *Combinazione convessa.* Per le proprietà precedenti, $p(x)$ espresso nella base di Bernstein è una combinazione convessa dei b_i, da cui segue

$$\min_i(b_i) \leq p(x) \leq \max_i(b_i) \qquad \forall x \in [a, b]$$

4. *Variazione di segno dei coefficienti.* Un polinomio espresso nella base di Bernstein ha un numero di variazioni di segno minore o uguale al numero di variazione di segno dei suoi coefficienti.

5. *Traslazione e scala.* I polinomi hanno la proprietà di essere invarianti per traslazione e scala dell' intervallo di definizione, i polinomi di Bernstein, in particolare, hanno la proprietà che, una volta effettuato il mapping $t = \frac{x-a}{b-a}$, i coefficienti rimangono gli stessi. Infatti si può dimostrare che

$$B_{i,n}(x) = \frac{n!}{i!(n-i)!}(1-t)^{n-i}t^i = B_{i,n}(t) \tag{1.2}$$

Quindi un polinomio nella base di Bernstein definito in [0,1] è dato da:

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \qquad \textit{con } t \in [0, 1]. \tag{1.3}$$

6. *Unicità del massimo.* $B_{i,n}(t)$ ha un solo massimo in [0,1] e questo si ha in corrispondenza di $t = \frac{i}{n}$.

1.1.2 Formula ricorrente base di Bernstein

Il calcolo di $B_{i,n}(t)$ può essere eseguito attraverso un algoritmo ricorsivo. Infatti

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t) \qquad i = 1, 2, .., n-1 \tag{1.4}$$

dove non sono inclusi $i = 0 \quad , \quad i = n \quad e \quad B_{0,0}(t) = 1$

- Per $i = 0$, non è permesso che il secondo termine della relazione soprascritta richieda il calcolo di $B_{i-1,n-1}$.
- Per $i = n$, non è permesso che il secondo termine della relazione richieda il calcolo di $B_{i,n-1}$.

Per $i = 0 \quad e \quad i = n$

$$B_{0,n}(t) = (1-t)B_{0,n-1}(t)$$
$$B_{n,n}(t) = B_{n-1,n-1}(t)$$

dove $B_{0,n-1} \quad e \quad B_{n-1,n-1}$ sono stati calcolati al passo precedente.

Dimostriamo che la formula 1.4 vale proprio $B_{i,n}(t)$:

$$(1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t) =$$

$$= \frac{(n-1)!}{i!(n-1-i)!}t^i(1-t)^{n-1-i}(1-t) + \frac{(n-1)!}{(i-1)!(n-1-i+1)!}t^{i-1}(1-t)^{n-1-i}t$$

Da cui si ottiene:

$$t^i(1-t)^{n-i}\frac{(n-1)!}{i!(n-i-1)!} + \frac{(n-1)!}{(i-1)!(n-i)!} = t^i(1-t)^{n-i}\frac{n!}{i!(n-i)!}\frac{(n-1)}{n} + \frac{i}{n}$$

che è uguale alla $B_{i,n}(t)$ data dalla 1.2.

1.1.3 Algoritmo di de Casteljau

Per la valutazione di un polinomio nella base di Bernstein esistono principalmente due metodi, uno diretto basato sulla formula di ricorrenza vista nel

paragrafo precedente, ed uno ricorsivo, ovvero l'algoritmo di de Casteljau.
Nel primo caso, si può valutare il polinomio in questione utilizzando la formula ricorrente sulle funzioni base per valutarlo nel punto desiderato e poi effettuarne la combinazione convessa 1.3.
In alternativa, si può usare l' algoritmo di de Casteljau, che vediamo.
Nato nel 1930 a Besancon, Francia, Paul de Casteljau ha lavorato come fisico e matematico alla Citroën dove nel 1959 ha sviluppato un algoritmo per il calcolo dei valori di curve polinomiali utili per il design e successivamente riconosciute uguali a quelle che oggi chiamiamo curve di Bézier.
L' algoritmo ricorsivo di de Casteljau è tuttora il metodo più robusto e numericamente stabile per la valutazione di polinomi, anche se più lento rispetto a metodi come quello di Horner. L' algoritmo in questione si può spiegare dall' applicazione ripetuta della formula ricorrente vista.

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) = \sum_{i=0}^{n} b_i t B_{i-1,n-1}(t) + \sum_{i=0}^{n} b_i (1-t) B_{i,n-1}(t)$$

Poichè $B_{-1,n-1} = B_{n,n-1} = 0$, abbiamo che $p(t)$ può essere espresso come

$$\sum_{i=0}^{n-1} b_{i+1} t B_{i,n-1}(t) + \sum_{i=0}^{n-1} b_i (1-t) t B_{i,n-1}(t) =$$

Raccogliendo otteniamo

$$= \sum_{i=0}^{n-1} [b_{i+1} t + b_i (1-t)] B_{i,n-1}(t) =$$

$$= \sum_{i=0}^{n-1} b_i^1 B_{i,n-1}(t)$$

e riapplicando la formula ricorrente più volte si ottiene

$$\sum_{i=0}^{n} b_i^n B_{i,0}(t) = b_0^n.$$

Si ha quindi il seguente algoritmo:

$$b_i^k = t b_{i+1}^{k-1} + (1-t) b_i^{k-1}$$

con $k = 1, .., n$ e $i = 0, .., n-k$.

Riassumendo si ottiene il seguente schema triangolare:

$$\begin{array}{cccccc} b_0^0 & b_1^0 & b_2^0 & ..... & b_{n-i}^0 & b_n^0 \\ b_0^1 & b_1^1 & b_2^1 & ..... & b_{n-i}^1 & \\ ..... & & & & & \\ b_0^{n-2} & b_1^{n-2} & b_2^{n-2} & & & \\ b_0^{n-1} & b_1^{n-1} & & & & \\ b_0^n & & & & & \end{array}$$

Un' ultima considerazione sull' algoritmo presentato ci permette di dire che questo risulta stabile in quanto i coefficienti $1-t$ e t sono compresi tra 0 e 1 e quindi non c' è alcuna amplificazione dell' errore.

1.1.4 Derivate dei polinomi di Bernstein

Le derivate dei polinomi di Bernstein di grado n sono polinomi di Bernstein di grado $n-1$; in particolare la derivata di un polinomio di Bernstein è esprimibile come il grado del polinomio moltiplicato per la differenza di due polinomi di grado $n-1$.

Prendiamo un polinomio $p(t)$ nella base di Bernstein

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t),$$

la sua derivata sarà data da

$$p'(t) = \sum_{i=0}^{n} b_i B'_{i,n}(t)$$

Per quanto accennato, si puo' dimostrare che vale la seguente relazione:

$$B'_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t))$$

per $0 \leq i \leq n$. Sostituendo si ottiene

$$\begin{aligned} p'(t) &= n \sum_{i=0}^{n} b_i [B_{i-1,n-1}(t) - B_{i,n-1}(t)] = \\ &= n \sum_{i=0}^{n} b_i B_{i-1,n-1}(t) - n \sum_{i=0}^{n} b_i B_{i,n-1}(t) = \\ &= n \sum_{i=0}^{n-1} b_{i+1} B_{i,n-1}(t) - n \sum_{i=0}^{n-1} b_i B_{i,n-1}(t) = \\ &= n \sum_{i=0}^{n-1} (b_{i+1} - b_i) B_{i,n-1} = \\ &= \sum_{i=0}^{n-1} d_i B_{i,n-1}(t) \end{aligned}$$

dove $d_i = n(b_{i+1} - b_i)$ e $i = 0, ..., n-1$ che sono i coefficienti della derivata di grado $n-1$ nella base di Bernstein.

1.1.5 Primitive dei polinomi di Bernstein

Sia $p(t) = \sum_{i=0}^{n} b_i B_{i,n-1}(t)$ un polinomio di grado $n-1$ definito in [0,1] nella base di Bernstein. Una sua anti-derivata o primitiva è un polinomio di grado n che nella base di Bernstein può essere espresso come

$$p^{-1}(t) = \sum_{i=0}^{n} b_i B_{i,n}(t)$$

con

$$b_{i+1} := \frac{d_i}{n} + b_i \quad i = 0, ..., n-1$$

La scelta arbitraria del valore b_0 è data dal fatto che esistono più primitive tutte differenti a meno di una costante additiva. Volendo inoltre determinare l' integrale definito di $p(t)$ si avrà :

$$\int_0^1 p(t)dt = [p^{-1}(t)]_0^1 = b_n - b_0 = b_n$$

Questo risultato si ottiene scegliendo $b_0 = 0$.

Nel particolare avremo che:

$$b_n = \frac{d_{n-1}}{n} + b_{n-1} = \frac{d_{n-1}}{n} + \frac{d_{n-2}}{n} + b_{n-2} = .. = \frac{1}{n}\sum_{i=0}^{n-1} d_i$$

1.2 Curve di Bézier

Le curve di Bézier sono importanti curve differenziali, che prendono il nome dall' ingegnere francese Pierre Bézier. Lavorando tra il 1933 e il 1975 presso la Renault, Bézier si pose il problema di disegnare le carrozzerie delle automobili con l'ausilio del calcolatore e proprio in quegli anni realizzò il sistema UNISURF CAD CAM.

Le curve di Bézier sono oggi alla base dei sistemi CAD 2D Computer Aided Design, CAD 3D, sistemi di disegno, software e linguaggi grafici. Sono inoltre uno standard per la rappresentazione di font di caratteri, di modelli CAD in genere e sono utilizzate nello standard SVG (Scalable Vector Graphics) per la grafica vettoriale in alternativa alla grafica bitmap.

L'idea è quella di definire una curva che segua il comportamento di una poligonale data, detta "Poligono di controllo".

1.2.1 Curve in forma parametrica

Dati $n+1$ punti $p_0, p_1, p_2, ..., p_n$ del piano, $p_i = (x_i, y_i)$ con $i = 0, ..., n$, detti "Punti di Controllo", la curva di Bézier è definita da:

$$C(t) = \sum_{i=0}^{n} p_i B_{i,n}(t) \quad con \quad t \in [0,1]$$

dove ogni elemento del vettore è una funzione polinomiale nella base di Bernstein

$$B_{i,n}(t) \quad i = 0, ..., n$$

Una tale curva è detta essere in forma parametrica in quanto definita in termini del parametro t ; al variare di t nell' intervallo [0,1] si avranno tutti i punti della curva.
L' unione dei punti di controllo costituisce il poligono di controllo; i punti di tale poligonale sono scelti in funzione della curva da rappresentare e sono in numero pari alle funzioni di Bernstein $(n + 1)$.
Queste funzioni possiedono alcune importanti proprietà che elenchiamo nel seguito.

1.2.2 Proprietà curve di Bézier

1. Il grado di una curva di B'ezier definita da $n+1$ punti di controllo è n.

2. Riprendendo la curva definita precedentemente, essa passa per p_0 e p_n ovvero per i punti di controllo estremi.

3. Le poligonali di controllo $p_0, p_1, ..., p_n$ e $p_n, p_{n-1}, ..., p_0$ descrivono la stessa curva.

4. *Proprietà del guscio convesso.* Questa proprietà assicura che la curva di Bézier definita da $n+1$ punti di controllo assegnati giace completamente nel convesso formato dai punti di controllo.

5. *Proprietà di Variazione Diminishing.* La proprietà afferma che il numero di intersezioni di una qualsiasi retta con la curva è sempre minore o uguale al numero di intersezioni della stessa retta con il poligono di controllo.

6. *Controllo Locale.* Variando la posizione di un punto di controllo la forma della curva di Bézier cambia in ogni punto.

7. *Invarianza affine.* Una curva di Bézier su cui viene eseguita una trasformazione affine corrisponde alla curva di Bézier disegnata a partire dai punti di controllo trasformati.

1.2.3 Disegno di curve di Bézier

Un metodo di rappresentazione delle curve di Bézier utilizza l'algoritmo di de Casteljau che si basa sull' applicazione ricorsiva della proprietà per cui, un polinomio di Bernstein di grado n può essere rappresentato in funzione di una combinazione lineare di polinomi di Bernstein di grado $n-1$.
Si può dimostrare che tale processo ricorsivo applicato a u_0 porta alla valutazione della curva di Bézier nel punto.

Consideriamo $u_0 \in [0,1]$ risulta:

$$C(u_0) = p_0^n$$

dove

$$p_i^j = (1-u_0)p_i^{j-1} + u_0 p_{i+1}^{j-1}$$

Numericamente, si valuta una curva di Bézier $C(u)$ in un punto $u_0 \in [0,1]$, costruendo il cosiddetto "Schema di de Casteljau":

$$\begin{array}{lllllll} p_0^0 & p_1^0 & p_2^0 & ..... & p_{n-i}^0 & P_n^0 \\ p_0^1 & p_1^1 & p_2^1 & ..... & p_{n-i}^1 & \\ ..... & & & & & \\ p_0^{n-2} & p_1^{n-2} & p_2^{n-2} & & & \\ p_0^{n-1} & p_1^{n-1} & & & & \\ p_0^n & & & & & \end{array}$$

$$p_0^n = C(u_0)$$

i cui elementi si ottengono applicando le precedenti formule ricorsive.

L' algoritmo di de Casteljau ha un particolare significato geometrico: se si dividono i lati del poligono di controllo secondo i rapporti u_0 e $1-u_0$, si ottengono dei punti che, collegati fra loro, individuano un nuovo poligono; ripetendo tale procedura per n volte, si perviene alla fine a un ultimo punto coincidente con $C(u_0)$.
Inoltre, l' ultimo lato dell' ultimo poligono considerato è tangente alla curva in $C(u_0)$.

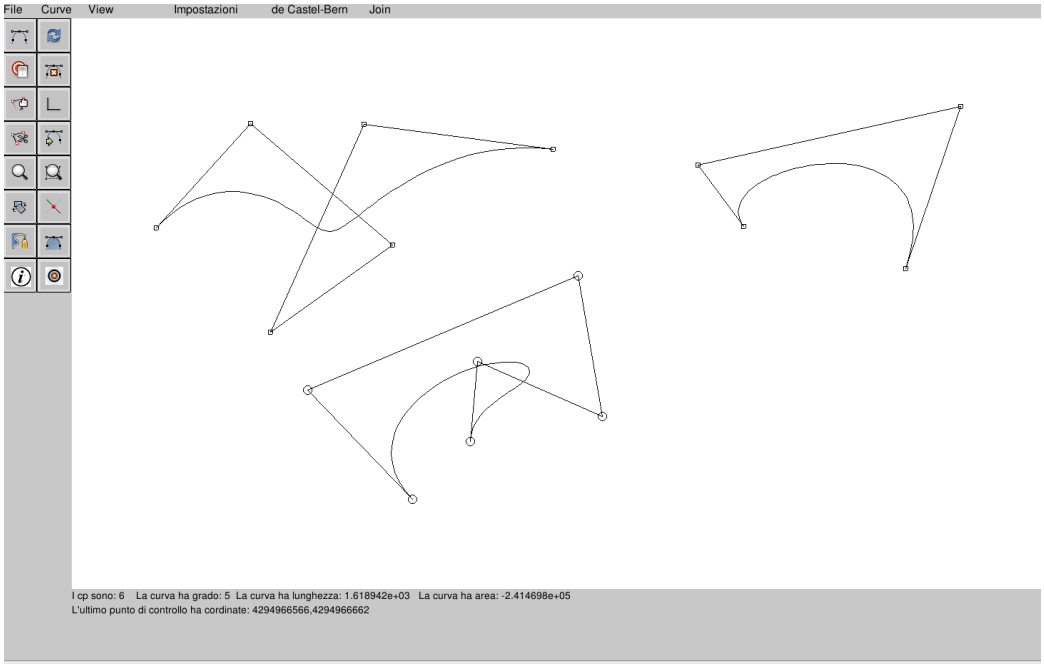

Figura 1.1: Esempi di curve di Bézier

1.3 Curve di Bézier a tratti

Le curve costituite da un unico polinomio sono spesso poco adeguate per descrivere forme geometriche complesse. I principali inconvenienti sono i seguenti:

- Per soddisfare un gran numero di vincoli è necessario avere un grado molto elevato, dato che per interpolare n punti dobbiamo utilizzare una curva polinomiale di Bézier di grado $n-1$. Curve di grado elevato sono però difficili da manipolare e numericamente instabili a causa dell' aumento degli errori di troncamento.

- Descrivere una curva con un unico segmento di Bézier non è semplice, poiché, anche se è possibile modificare la forma della curva spostando

i punti di controllo, lo spostamento interessa comunque tutta la curva.

La soluzione per ovviare a questi inconvenienti è quella di utilizzare curve polinomiali a tratti, cioè della forma:

$$pp(x) = \begin{cases} p_0(x) & x \in [x_0, x_1] \\ p_1(x) & x \in [x_1, x_2] \\ ..... & ..... \\ ..... & ..... \\ p_{m-1}(x) & x \in [x_{m-1}, x_m] \end{cases}$$

dove i polinomi p_i *con* $i = 0, ..., m-1$ sono tutti di grado n e soddisfano le seguenti condizioni:

- $p_{i-1}(x_i) = p_i(x_i) = y_i$
- Fissato $k \leq n-1$, non negativo, deve valere $p_{i-1}^{\ell}(x_i) = p_i^{\ell}(x_i)$ con $\ell = 1, ..., k$ *e* $i = 1, ..., m-1$.

1.3.1 Tipi di continuità

Il modo in cui due segmenti di curve si connettono alle estremità prende il nome di continuità.
Si classificano due tipi di continuità: continuità parametrica C e continuità geometrica G.

- Curve G1 possono diventare C1 in seguito a una ri-parametrizzazione.
- C1 implica G1 ma non il contrario.
- C2 e G2 hanno la stessa curvatura nel punto di contatto.

Continuità parametrica

In generale due curve continue parametricamente saranno continue anche geometricamente, ma il viceversa non è verificato. Prese due curve p e q e sia u il loro punto di contatto, diciamo che le due curve sono unite con continuità parametrica k se nel punto di giunzione u hanno la stessa derivata parametrica fino all'ordine k.

Continuità C0/G0

Se gli estremi di ciascuna curva si trovano nello stesso punto dello spazio, le curve presentano continuità di posizione (G0/C0) presso gli estremi. In altre parole, le due curve in questione si toccano in corrispondenza dei loro estremi.

Continuità C1

Ovviamente per avere un incollamento di tipo C1 deve esserci prima di tutto una continuità di tipo C0, che come abbiamo appena visto è equivalente alla richiesta che il punto finale e il punto iniziale dei due poligoni di controllo coincidano.
Oltre a ciò è necessario che nel punto di congiunzione la derivata prima delle due curve, sia uguale sia in modulo che in direzione.

Continuità geometrica

Questo tipo di continuità è espressa con G ed è un tipo di continuità debole. Richiede che le derivate nei punti di giuntura abbiano la stessa

direzione ma non necessariamente lo stesso modulo.

Continuità G1

Misura la posizione e la direzione della curva agli estremi. In altre parole, le due curve non solo si toccano, ma vanno nella stessa direzione in corrispondenza del punto in cui si toccano.
La direzione è determinata dal primo e dal secondo punto di controllo su ciascuna curva. Se questi due punti ricadono su una linea, le due curve sono tangenti alle estremità. Nell'immagine seguente, è rappresentata una curva a tratti con continuità C0/G0 (in alto a sinistra), una curva a tratti con continuità C1 (in basso a sinistra) e una curva a tratti con continuità G1 (a destra).

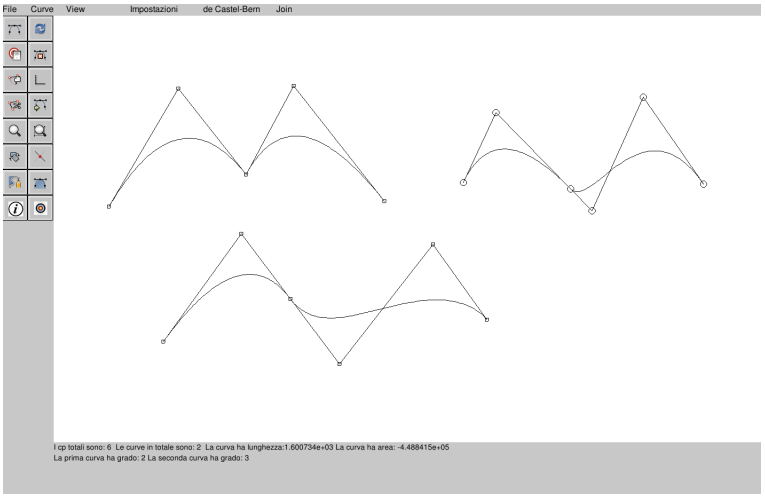

Figura 1.2: Tipi di continuità

Capitolo 2

MINI SYSTEM NELLA DIDATTICA

Gli studi sull'apprendimento hanno portato nel passato a definire i requisiti necessari perché uno studente assimili efficacemente un concetto, e quale metodo debba essere seguito per trasmetterlo. In particolare Atkin e Karplus (1962) hanno definito una strategia in seguito chiamata ciclo di Karplus che si articola in tre fasi:

1. Verifica ed esplorazione dei concetti imparati.

2. Discussione dei risultati ottenuti.

3. Sviluppo dei nuovi concetti.

La prima fase si basa sul fatto che l'apprendimento è tanto più efficace quanto maggiore è la possibilità per lo studente di sperimentare. La seconda indica la necessità di verificare i risultati ottenuti o per mezzo di una discussione in classe o per mezzo di soluzioni preesistenti. E la terza e ultima fase presuppone la messa in pratica delle conoscenze acquisite nei punti precedenti. Il

MiniSystem permette di sviluppare pienamente questa tecnica; esso infatti è nato con l' idea di aiutare gli studenti nel loro processo di apprendimento delle nozioni di Calcolo Numerico permettendo loro di verificare i concetti acquisiti durante il corso in maniera pratica e dandogli la possibilità, dopo un attento studio degli argomenti, di sviluppare personalmente dei nuovi algoritmi per aumentare le funzionalità di questo software.

È proprio in questo ambito che il supporto dei calcolatori assume un ruolo centrale. I vantaggi dell' uso del calcolatore nella didattica sono infatti numerosi. La prima cosa da valutare è il grande interesse che il computer suscita negli studenti. Questo fattore infatti potrebbe far avvicinare alla materia anche gli studenti meno attratti da quest' ultima. Le esercitazioni al computer inoltre, permettono agli studenti di applicarsi su problemi che altrimenti risulterebbero complicati e farebbero perdere troppo tempo. Un ulteriore vantaggio è dato dalla possibilità di avere un insegnamento orientato più verso il singolo. La possibilità infatti di poter testare in modo autonomo i risultati ottenuti, dà la possibilità agli studenti di auto-valutarsi.

Questo lavoro di tesi vuole integrare al materiale didattico tradizionale, esercizi ed esempi eseguibili nell' ambiente MiniSystem. Il percorso svolto per ottenere questo risultato si compone di tre fasi: una prima fase di analisi del codice del MiniSystem e osservazione in classe, una seconda fase in cui è stato elaborato un percorso di esercitazioni per gli studenti e un' ultima fase basata sull'implementazione di una serie di algoritmi utili sia per gli studenti che per l'arricchimento del codice.

2.1 Primo approccio con il MiniSystem

Il percorso che ha portato all'introduzione del MiniSystem come mezzo di supporto e verifica per gli studenti ha avuto inizio nel momento in cui il software in questione è stato presentato agli studenti a lezione. La partecipazione ad alcune lezioni mi ha infatti permesso di avere una panoramica generale del software, di riprendere alcuni concetti del corso che sono implementati in quest'ultimo, ma soprattutto di analizzare l' atteggiamento con cui gli studenti si pongo davanti a dei concetti per loro nuovi e di osservare quali aspetti e quali modalità attraggano maggiormente la loro attenzione. In particolar modo questa prima fase mi ha portato a riflettere su due punti:

1. Innanzitutto ho capito quanto il computer possa avere un ruolo fondamentale nel processo di apprendimento; mentre infatti nelle normali lezioni didattiche, alcuni studenti effettuavano unicamente un lavoro di ricopiatura delle nozioni spiegate dal professore, durante le lezioni di spiegazione del MiniSystem, che quindi richiedevano il supporto di un computer, anche gli studenti più orientati verso altre materie apparivano interessati, partecipi ed interagivano nelle lezioni.

2. In secondo luogo, ho riscontrato come gli studenti abbiano la necessità di non imparare unicamente i concetti teorici della materia ma di vedere soprattuto come questi possano essere messi in pratica. Questa opportunità gli permette infatti di avere un' idea più chiara di quello che stanno studiando e quindi un miglior approccio con la materia.

Scopo della prima fase è stato quindi quello di studiare e analizzare il codice del MiniSystem e iniziare ad individuare un possibile approccio studente-MiniSystem in base alle reazioni osservate a lezione.
Vediamo ora nello specifico come molti degli argomenti del corso di "Calcolo Numerico" sono alla base del MiniSystem e come gli studenti possano interagire con questo software.

2.1.1 Il "Calcolo Numerico" nel MiniSystem

Come già detto, nel MiniSystem sono presenti molti degli argomenti trattati nel corso di Calcolo Numerico; nello specifico, esso permette la progettazione e il disegno di curve 2D per mezzo di curve polinomiali e curve polinomiali a tratti nella forma di Bernestein/Bézier. Le curve in questione possono essere progettate dal nulla principalmente con tre modalità: modellazzione di forma, interpolazione e approssimazione. In tutti e tre i casi è possibile compiere diverse operazioni sia per la gestione delle curve, che per la modifica e l'eliminazione.
Vediamo ora alcune operazioni possibili nel MiniSystem evidenziando soprattutto come sono collegate ad aspetti di "Calcolo Numerico".

- *Definizione di curve e immediata rappresentazione grafica*
  Il MiniSystem offre la possibilità di disegnare una curva dandone in input i relativi punti di controllo e ottenere immediatamente la rappresentazione grafica di quest'ultima.
  Per fare ciò si fa uso della valutazione polinomiale che è uno dei primi concetti di utilizzo di polinomi in "Calcolo Numerico".

Nello specifico, all'interno del codice sono implementati due algoritmi che si occupano di valutare un polinomio e che vengono richiamati dalla funzione "bezier_draw" che permette sia la valutazione che il disegno di una curva. Particolarmente importante risulta anche l'implementazione dell'algoritmo "vis-norm" che permette di disegnare una curva di Bézier mediante un algoritmo adattivo basato su suddivisioni ricorsive. La funzione che viene richiamata è "vis-adaptive" che permette di visualizzare la curva con un approccio adattivo al pixel.

- *Interpolazione di punti.*

  Oltre a permettere l'interpolazione di un set di punti con un'interpolazione a tratti, il MiniSystem permette anche l'interpolazione polinomiale ovvero l'interpolazione di un set di punti con un polinomio intero. Le funzioni che si occupano di ciò implementano i metodi di interpolazione polinomiale visti a lezione.

- *Interpolazione locale e globale.*

  All'interno del codice è possibile interpolare una serie di punti sia in maniera locale che in maniera globale; nel primo caso, vengono determinate le curve di Bézier di grado 3 che li interpolano 2 a 2 insieme ad una stima delle derivate prime, mentre nel secondo caso, si determina la curva per mezzo della risoluzione di un sistema lineare. Nel file "interp.c" , infatti, è implementato un algoritmo che permette di fattorizzare una matrice in LU con L triangolare inferiore e U triangolare superiore usando il metodo di eliminazione di Gauss con scambio delle righe (pivoting parziale) e perno massimo.

È inoltre presente un ulteriore algoritmo che risolve un sistema lineare a partire dalla fattorizzazione LU della matrice dei coefficienti.

- *Disegno delle tangenti e delle normali.*

  Un'ulteriore operazione offerta dal Mini-System è quella di disegnare le tangenti e le normali di una curva di Bézier. La funzione che si occupa di ciò è implementata nel file "draw.c" e calcola la relativa curva derivata per mezzo di una combinazione lineare.

- *Calcolo dell'area e della lunghezza.*

  All'interno del codice sono implementati due algoritmi rispettivamente per il calcolo dell'area e della lunghezza di una curva. Entrambi si basano su un algoritmo che permette di valutare l'integrale definito della funzione passata in input; è una routine automatica che si basa sulle formule di Newton-Cotes relative ad 8 intervalli. Nello specifico, sappiamo che data una curva $C(t)$ con $t \in [0, 1]$, la lunghezza è espressa come

  $$L = \int_0^1 ||C'(t)||dt = \int_0^1 \sqrt{x'(t)^2 + y'(t)^2 dt}$$

  con $x(t)$ ed $y(t)$ le componenti della $C(t)$ e $x'(t)$ e $y'(t)$ le componenti della $C'(t)$.

  Infatti l'infinitesimo di una curva sarà dato da $dC(t)$ che approssimeremo con $||C'(t)||dt$ e la cui lunghezza sarà $\int_0^1 ||C'(t)||dt$.

  Una volta progettata la curva e calcolato la sua lunghezza L_C, vogliamo determinare una curva $G(t)$ di lunghezza L_G avente la stessa forma di $C(t)$. A tal fine si noti che se la curva C(t) viene scalata di un fattore

s la sua lunghezza viene scalata dello stesso fattore; infatti sia

$$G(t) = \begin{pmatrix} sx(t) \\ sy(t) \end{pmatrix}$$

la curva C(t) scalata del fattore s allora sarà

$$L_G = \int_0^1 |\sqrt{[sx'(t)]^2 + [sy'(t)]^2 dt} = s \int_0^1 dt \sqrt{x'(t)^2 + y'(t)^2 dt} = sL_C$$

Segue che assegnata la lunghezza L_G sarà sufficiente calcolare $s = \frac{L_G}{L_C}$ e scalare la curva $C(t)$ per tale fattore ottenendo una curva $G(t)$ della forma della $C(t)$ e di lunghezza desiderata. Per quel che riguarda l'area, sappiamo che può essere espressa come:

$$A = +/- \frac{1}{2} \int_0^1 C(t) x C'(t) dt = \frac{1}{2} \int_0^1 [x(t)y'(t) - x'(t)y(t)] dt$$

con $x(t)$ ed $y(t)$ le componenti della $C(t)$ e $x'(t)$ e $y'(t)$ le componenti della $C'(t)$.

Per calcolare tale area sarà necessario utilizzare una formula di integrazione numerica, e se si desidera l'approssimazione con una certa tolleranza si può utilizzare il metodo di Simpson adattivo, ma si può anche osservare che la funzione integranda, a differenza del caso della lunghezza della curva, è una funzione polinomiale.

Il segno di quest'area è positivo se la curva è parametrizzata in senso antiorario nel piano xy.

- *Intersezione tra due curve.*

  Date due curve $C(t)$ con $t \in [0, 1]$e $D(s)$ con $s \in [0, 1]$, si vogliono determinare i punti di intersezione che consistono, a seconda delle applicazioni, nelle coppie $(\dot{t}, \dot{s})$, nei punti $\dot{Q} = (q\dot{x}, q\dot{y})$ o in entrambi. Una

soluzione consiste nel suddividere le due curve date in corrispondenza dei loro punti estremi ottenendo così due insiemi di tratti di curve che rappresentano rispettivamente la $C(t)$ e la $D(s)$; l'algoritmo continua nel confrontare ogni tratto della $C(t)$ con quelli della $D(s)$. Si osservi che ognuno di tali segmenti di curva è sicuramente contenuto nel rettangolo di vertici opposti gli estremi del segmento di curva. Questa osservazione è tanto semplice quanto potente in quanto permette di fare un test molto semplice e veloce per controllare se due tratti di curva si intersecano o meno. Se non si intersecano i rettangoli, non si intersecheranno nemmeno i tratti di curva; se si intersecano i rettangoli si procede suddividendo i due tratti a metà e ricontrollando i quattro accoppiamenti.

- *Determinazione punto interno o esterno ad una curva.*
  Data una curva chiusa piana $C(t)$ con $t \in [0, 1]$ nella forma di Bézier (o Bézier a tratti) e un punto Q del piano, si vuole determinare se il punto è interno o esterno alla regione di piano delimitata dalla curva. Per risolvere il problema si possono determinare le intersezioni tra la curva $C(t)$ e la semiretta uscente da Q; a questo punto si valutano le intersezioni, se sono dispari allora il punto è interno, altrimenti è esterno. Consideriamo ora l' espressione della curva in forma parametrica e l''equazione della retta in forma cartesiana

$$G(t) = \begin{pmatrix} c_1(t)) \\ c_2(t) \end{pmatrix} e \quad ax + by + c = 0$$

  Se sostituiamo le componenti x e y della curva nell'equazione della retta

otteniamo il seguente risultato:

$$aC_1(t) + bC_2(t) + c = 0$$

e se $C(t) = \sum_{i=0}^{n} p_i B_{i,n}(t)$ con $p_i(x_i, y_i)^T$ allora sostituendo si ottiene

$$a\sum_{i=0}^{n} x_i B_{i,n}(t) + b\sum_{i=0}^{n} y_i B_{i,n}(t) + c\sum_{i=0}^{n} B_{i,n}(t) = 0$$

e quindi

$$\sum_{i=0}^{n}(ax_i + by_i + c)B_{i,n}(t) = 0$$

Si tratta di determinare gli zeri di un polinomio di grado n nell'intervallo [0,1]. Le soluzioni saranno i parametri della curva in corrispondenza dei quali la curva e la retta si intersecano. Valutando la curva in corrispondenza di tali parametri si ottengono le coordinate cartesiane dei punti di intersezione. Quindi per determinare se un punto $Q = (q_x, q_y)^T$ è interno o esterno, si considera la retta orizzontale $y = qy$ e si determinano le radici di

$$\sum_{i=0}^{n}(y_i - q_i)B_{i,n}(t) = 0$$

Si valuti poi la componente x della curva in corrispondenza delle soluzioni trovate e si contino i valori trovati maggiori di q_x.

- *Distanza di un punto da una retta.*

  Data una curva piana $C(t)$ nella forma di Bézier (o Bézier a tratti) e un punto Q del piano, si vuole determinare il punto della curva più prossimo a Q. Si tratta di trovare quel punto della curva per cui risulta

ortogonale alla tangente alla curva, cioè

$$(C(t) - Q)C'(t) = 0$$

Si devono determinare gli zeri di un polinomio di grado $n(n-1)$.

2.1.2 I casi d' uso del MiniSystem

MiniSystem ha l'obiettivo di supportare tutti gli studenti nel loro percorso di studio, da quelli che hanno già appreso completamente i concetti del corso, a quelli che devono ancora acquisire pienamente la padronanza della materia. Esistono quindi due possibili approcci che gli studenti possono avere con il software presentato.

Verifica dei concetti acquisiti

Una possibilità è quella di utilizzare le numerose potenzialità del MiniSystem per capire in modo pratico le nozioni mostrate durante il corso o per verificare di aver appreso in modo corretto queste ultime.
Questa modalità di utilizzo è possibile a tutti gli studenti e permette di accompagnare i ragazzi in tutto il percorso di studio; il MiniSystem, infatti, gestisce molti degli aspetti del corso di Calcolo Numerico e quindi risulta utile sia per esercitarsi sia per vedere come sono stati implementati gli algoritmi spiegati a lezione.

Implementazione di nuove funzioni

Una diversa possibilità è quella di interfacciarsi col MiniSystem dopo aver acquisito una buona padronanza dei concetti imparati. In questo caso, è possibile modificare o introdurre nuove funzioni all'interno del codice, che possano arricchire le potenzialità del software in questione. Questa modalità, oltre ad aiutare lo studente nella sua preparazione sugli argomenti del corso, lo spinge a sviluppare nuove idee e quindi a mettersi in prova in esercizi non propriamente standard.
Alcuni aspetti che potrebbero essere implementati sono ad esempio:

1. Caricamento e salvataggio di file in formato SVG (Scalable Vector Graphics).

2. Studiare l' interpolazione polinomiale mediante un nuovo approccio noto in letteratura come PIA (Progressive Iteration Approximation).

3. La funzione intersect chiama la funzione Cbezier clipping per determinare i punti di intersezione. Nel file zerip.c sono presenti più algoritmi per determinare gli zeri di funzioni polinomiali; analizzarne ed usarne alcuni.

4. I punti di intersezione determinati, vengono memorizzati per usi successivi fin a che non interviene una nuova intersezione. Prevedere di conservarli per ogni intersezione effettuata.

5. Curve o punti da file potrebbero non essere visualizzati se, quando caricati, non sono contenuti nella window in quel momento definita.

Prevedere una funzione che al caricamento determini la più piccola area rettangolare contenente le informazioni caricate e aggiorni la window.

2.2 Le esercitazioni

La seconda fase è stata focalizzata sulla creazione di un possibile percorso di esercitazioni da integrare al corso di "Calcolo Numerico" ed eseguibile con l' ausilio del MiniSystem. L'obiettivo è quello di fare in modo che gli studenti possano accompagnare lo studio delle nozioni teoriche con esercizi che gli permettano sia di verificare di aver appreso correttamente e pienamente un concetto, sia di vedere come la teoria può essere messa in pratica.

1. Prima esercitazione

   Introduce gli studenti nell'ambiente del MiniSystem permettendo loro di approfondire la conoscenza di quest' ultimo attraverso la presentazione delle principali operazioni possibili e di esercizi guidati. Sono assegnati una serie di quesiti, non con lo scopo di introdurre un qualche argomento del corso, ma con l'unica finalità di aiutare gli studenti nel loro primo approccio con questo software, guidandoli nell'esecuzione delle operazioni base da compiere.

2. Seconda esercitazione

   Verifica la conoscenza dei ragazzi sulle curve di Bézier e sui concetti fondamentali di queste ultime. L' esercitazione è composta da 9 quesiti, che partendo dalla semplice richiesta di disegnare una curva arrivano, in

modo graduale, a fare in modo che lo studente possa sviluppare algoritmi propri. Scopo di questa esercitazione è mostrare come, ad esempio, gli algoritmi di de Casteljau o l'utilizzo dei polinomi di Bernstein per il disegno di curve, agiscono in pratica o come venga implementato il calcolo dell'area e della lunghezza di una curva.

3. Terza esercitazione

   È focalizzata sullo studio delle curve di Bézier definite per interpolazione. È composta anch' essa da 9 quesiti che vanno dal determinare la curva di Bézier in base a un set di punti dati, alla richiesta di implementare la gestione dell' interpolazione di curve chiuse e periodiche. Una delle finalità di questa esercitazione è mostrare come i punti vengano interpolati mediante un' interpolazione locale o globale. All' interno del codice sono sviluppati due algoritmi per l' individuazione di un curva di Bézier per interpolazione; il primo la individua determinando le curve di Bézier di grado 3 che interpolano i punti due a due , il secondo, permette di individuarla per mezzo della risoluzione di un sistema lineare. In quest' ultimo caso, l'algoritmo richiama, per la risoluzione del sistema lineare, un algoritmo per la fattorizzazione LU della matrice per mezzo dell'algoritmo di Gauss con scambio delle righe e perno massimo. Un ulteriore esercizio per gli studenti sarebbe vedere, studiare e capire l'implementazione di questi algoritmi.

4. Quarta esercitazione

   È incentrata sullo studio delle curve di Bézier a tratti. Tra i quesiti proposti troviamo la richiesta di riprodurre le funzioni seno e coseno per

mezzo di curve di Bézier e la costruzione di curve di Bézier a tratti. Viene inoltre chiesto di analizzare più nello specifico il fenomeno di Runge e il calcolo dei nodi di Chebyshev. In questa esercitazione inoltre, vengono richiamati altri argomenti del corso come la determinazione degli zeri di funzioni polinomiali e lo studio delle curve spline.

5. Quinta esercitazione

   Presupponendo che gli studenti abbiano già acquisito familiarità con il software, ha lo scopo di far sviluppare direttamente a loro, particolari aspetti di quest' ultimo. Nello specifico è richiesta l'implementazione sia di algoritmi non ancora presenti nel codice, ma che potrebbero arricchirlo ulteriormente sia di algoritmi che modifichino funzioni già presenti, migliorandole o aggiungendo nuove funzionalità.

2.3 Implementazione delle nuove funzioni

L'ultima parte di questo percorso è stata focalizzata sull'implementazione di una serie di funzioni con tre precisi scopi:

1. Introdurre nel codice del MiniSystem una serie di algoritmi che aiutino gli studenti nel primo approccio con questo software.

2. Implementare degli algoritmi che permettano all' utente di percepire non solo l' aspetto grafico delle operazioni ma anche l' aspetto matematico alla base delle funzioni.

3. Introdurre nel MiniSystem la possibilità di gestire non solo curve di Bézier semplici ma anche curve di Bézier a tratti.

Vediamo ora più nello specifico come sono stati raggiunti questi obiettivi.

2.3.1 Funzioni di supporto all' utente

Queste funzioni sono state implementate con l'obiettivo di rendere più facile e più intuitivo il primo approccio con il software presentato in questa tesi. Nella versione precedente, infatti, molte operazioni davano per scontato che l' utente avesse già una buona padronanza delle principali operazioni possibili, ma in questo modo utenti non esperti, rischiavano di non usufruire pienamente di tutte le potenzialità del MiniSystem. Per ovviare a questo problema, è stata implementata una finestra di dialogo permanente nella parte bassa della finestra nella quale vengono fatti comparire dei messaggi utili per il corretto utilizzo delle operazioni disponibili. Sono inoltre stati implementati una serie di algoritmi che permettono all'utente di aver sempre una panoramica completa sulle impostazioni della window e della viewport.

2.3.2 Funzioni per l' aspetto matematico

Un altro possibile rischio a cui si poteva andare incontro utilizzando precedentemente il MiniSystem era quello di perdere il concetto matematico alla base del disegno. Per porre rimedio a questo problema è stata introdotta una nuova finestra di dialogo che permette, ogni qual volta viene inserita una nuova curva o viene compiuta un' operazione su di essa, di far apparire

a video tutte le informazione su quest' ultima, come le coordinate dell' ultimo control point inserito, l'area, la lunghezza, il grado e il numero totale di control point. Nel caso in cui l' utente sia interessato a conoscere tutte le informazioni sulla curva e quindi anche le coordinate di tutti i control point inseriti, basterà selezionare l' apposito bottone "info" disponibile nella barra laterale. È stato inoltre introdotto un nuovo menu "impostazioni" che permette all' utente di modificare i valori della Window direttamente da schermo; nella versione precedente del codice questo non era possibile, i valori infatti potevano essere modificati solo attraverso il codice.

2.3.3 Curve di Bézier a tratti

L'ultimo step di questo percorso è consistito nell' inserire alcuni algoritmi per la gestione di curve di Bézier a tratti. Nella versione iniziale del MiniSystem infatti, era possibile unicamente creare due curve ed effettuare su di esse un Join per mezzo della relativa voce nel menù "curve" che permetteva di unirle con la continuità desiderata.
Al fine di migliorare il codice, è quindi stata modificata la funzione di creazione di una curva affinché potesse gestire anche curve a tratti con diverse continuità e sono state modificate le funzioni che effettuavano il join tra due curve in modo tale da trasformarle successivamente al join in un unica curva a tratti. È stato inoltre implementato un nuovo algoritmo che permette di caricare a video un' immagine ed effettuare su di esse il disegno di una curva semplice, a tratti o per interpolazione.

Tutte le funzioni presentate in questo paragrafo verrano poi riprese nel capitolo successivo per una descrizione più accurata degli algoritmi implementati.

2.4 Obiettivi attesi

Come già detto, queste esercitazioni sono state create come supporto per gli studenti. L'obiettivo è quello di permettere un più facile apprendimento dei concetti mostrandone non solo l'aspetto teorico ma anche l'aspetto pratico e come questi concetti possano essere sviluppati a livello algoritmico. Per mezzo di esse infatti, gli studenti possono, sperimentando loro stessi o analizzando il codice offerto, avere una panoramica di tutti gli argomenti del corso a partire dalla stima dell' errore fino ad arrivare all' integrazione numerica. Ruolo centrale in questo percorso è acquisito dal computer. Le possibilità grafiche offrono infatti allo studente una maniera espressiva e immediata per comprendere gli argomenti.
Ci si attende quindi un maggior interesse da parte degli studenti, una miglior interazione alunno-professore e uno studio più costante della materia.

Capitolo 3

MiniSystem

Mini-System è un software didattico sviluppato per il disegno di curve di Bézier, che implementa alcuni metodi di calcolo numerico efficienti per la gestione di polinomi e polinomi a tratti nella base di Bernstein.
Il codice è scritto in linguaggio C e per la parte grafica fa uso delle librerie SDL 1.2, sia per la GUI che per le funzioni di disegno.

3.1 Motore di calcolo e motore grafico

Il codice sorgente del MiniSystem è composto da:

1. La "gui" che risulta nettamente separata dalle funzioni di calcolo e dalle funzioni di disegno e che contiene le funzioni di inizializzazione e visualizzazione dei bottoni, dei menu e dell'interfaccia grafica .

2. Il "controller" che contiene le funzioni di callback associate ai bottoni e ai menu e altre funzioni per mezzo delle quali traduce i comandi di input in chiamate a funzioni di calcolo.

3. Il "motore di calcolo" che contiene una serie di funzioni efficienti per gestire polinomi e polinomi a tratti nella base di Bernstein come la valutazione numerica, la valutazione di derivate e altre operazioni che vedremo nello specifico nei prossimi paragrafi.

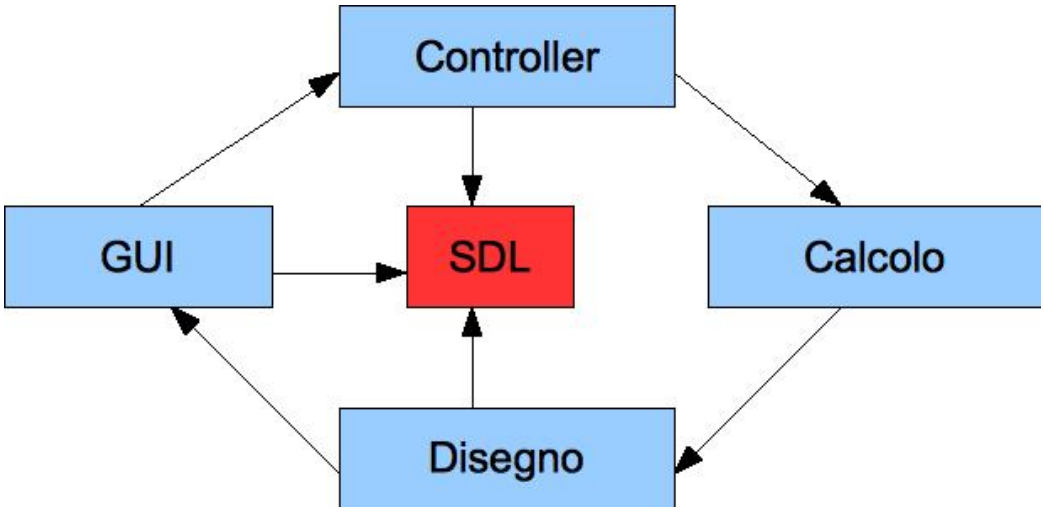

Figura 3.1: Suddivisione del codice sorgente

In questo capitolo spiegheremo come ottenere e compilare il codice sorgente del MiniSystem, inizieremo a navigare al suo interno da un punto di vista strutturale (file e directory che lo compongono) e da un punto di vista grafico.

3.1.1 Il codice del MiniSystem: come ottenerlo e come compilarlo

La prima cosa da fare è scaricare dalla pagina web del corso l'archivio "MiniSystem.tgz" e scompattarlo nella propria home directory.
Verrà creata una cartella di nome "MiniSystem" contenente il codice e al-

cuni file di esempio. Prima di eseguirlo, è necessario ottenere le librerie SDL1.2 scaricandole dal sito www.libsdl.org oppure cercandole come pacchetti disponibili nella propria distribuzione linux digitando le seguenti linee da terminale

```
apt-cache search SDL
```

In debian, ad esempio, sono disponibili i pacchetti:

1. libsdl1.2debian
2. libsdlimage1.2
3. libsdlttf2.

A questo punto, verifichiamo che il codice sia funzionante compilandolo ed eseguendolo per mezzo delle seguenti istruzioni.

```
cd .../MiniSystem/src
make
/minsys
```

3.1.2 Motore di calcolo

Esploriamo ora il codice sorgente che abbiamo scaricato con le istruzioni del paragrafo precedente. Si tratta di una collezione di funzioni efficienti per gestire polinomi, polinomi a tratti nella base di Bernstein e altri concetti

studiati durante il corso “Calcolo Numerico”.
Le principali funzioni sono:

- Valutazione numerica.
- Valutazione di derivate.
- Suddivisione.
- Interpolazione locale e globale di punti.
- Integrazione numerica di linea e di area.
- Intersezione o ricerca di radici.
- Trasformazioni geometriche.

Una volta scompattato, l’archivio minisystem.tgz risulterà suddiviso nel seguente modo.

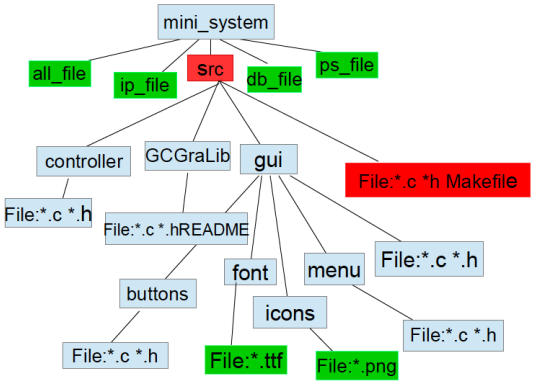

Figura 3.2: La struttura del codice di MiniSystem

Vediamo ora nello specifico le strutture dati utilizzate all'interno del codice e come le principali funzioni vengono implementate.

Le strutture dati

Le principali strutture dati utilizzate all'interno del codice sono descritte nel file "types.h". Elenchiamo di seguito le più importanti:

$$typedef\ double\ real\_t;$$

che permette di utilizzare all'interno del codice la precisione doppia, ma con una semplice modifica anche la precisione singola.
Vengono poi impostate alcune costanti come:

- MAX_DEGREE Che rappresenta il grado massimo di una curva.
- MAX_CURVE Che rappresenta il numero massimo di curve.

E infine vengono definite le seguenti strutture dati:

- Bezier_t
  è una struttura dati che permette di memorizzare sia una curva di Bézier semplice che una curva di Bézier a tratti. È composta da 5 campi.
  - int order
    Rappresenta l'ordine della curva
  - real_t xcp[MAX_DEGREE+1];
    È un vettore di dimensione massima MAX_DEGREE contenente le coordinate x dei control point della curva.

  - real_t ycp[MAX_DEGREE+1];

    È un vettore di dimensione massima MAX_DEGREE contenente le coordinate y dei control point della curva.

  - real_t zero,one;

    Rappresenta il dominio parametrico della curva.

  - struct Bezier_t *next;

    Punta a una struttura dello stesso tipo; se si tratta di una curva singola, allora punterà a NULL, altrimenti punterà alla successiva curva.

- point

  Point è adibita a memorizzare una lista di MAX_DEGREE punti a partire dai quali si definisce una curva di Bézier per interpolazione. È composta da 4 campi.

  - int order

    Rappresenta lòrdine della curva

  - real_t xcp[MAX_DEGREE+1];

    È un vettore di dimensione massima MAX_DEGREE contenente le coordinate x dei control point della curva.

  - real_t ycp[MAX_DEGREE+1];

    È un vettore di dimensione massima MAX_DEGREE contenente le coordinate y dei control point della curva.

  - real_t zero,one;

    Rappresenta il dominio parametrico della curva.

– struct Bezier_t *next;

Punta a una struttura dello stesso tipo; se si tratta di una curva singola, allora punterà a NULL, altrimenti punterà alla successiva curva.

Nel file const.h poi sono definite :

- SCREEN_W e SCREEN_H che definiscono l'ampiezza della finestra schermo.

La finestra schermo viene utilizzata per contenere una barra dei menù e dei bottoni, oltre a un'area di disegno che chiameremo tecnicamente viewport e che viene settata chiamando la funzione draw_env presente nel file "screen.c" nella cartella controller.
Sempre in questo file sono dichiarate le costanti relative alla barra dei menù e alla barra dei bottoni.
Nel main vengono dichiarati:

1. Una lista di MAX_CURVE puntatori a strutture di tipo Bezier_t.

$$Bezier\_t * p[MAX\_CURVE + 1];$$

2. Una lista di interi che permetterà di capire in modo più agevole se la curva i-esima è stata allocata ed è in uso nel sistema o meno. Nel primo caso ip[i] sarà =1, altrimenti = 0.

$$int\ ip[MAX\_CURVE + 1];$$

3. Una lista di MAX_CURVE puntatori a Point.

$$Point * q[MAX\_CURVE + 1];$$

4. Una lista di interi iq che ha la stessa funzione del vettore ip con l' unica differenza che tiene traccia delle curve definite per interpolazione.

Nel main inoltre, viene definita la window iniziale, ossia la finestra sul mondo e per la precisione viene definita a essere un'area rettangolare [-10, 10] x [-10, 10].

Le funzioni

Vediamo ora nello specifico come il MiniSystem gestisce la creazione e la modifica delle curve, le operazioni su di esse, il loro disegno e il modo in cui vengono salvate.

- *Funzioni di gestione curve*

  Il MiniSystem offre la possibilità di compiere diverse operazioni sulle curve di Bézier, in particolare le operazioni di creazione ed eliminazione sono accessibili dal menù Curve e dalla barra dei bottoni. Per quel che riguarda la creazione, nel momento in cui si introduce nel sistema una nuova curva di Bézier o una curva di Bézier a tratti, per mezzo della funzione "new_id_curva" viene cercato il primo puntatore libero della lista p a Bezier_t o della lista q a Point se la curva è definita per interpolazione; dopo aver allocato lo spazio per la curva, viene richiamata la funzione "new_curva" che permette di gestire sia curve semplici che curve a tratti e che, in base all'input dato dall'utente, salva i dati nella relativa struttura "bezier_t". Nel caso in cui invece si vogliano assegnare una serie di punti da interpolare, basterà scegliere la relativa opzione dal menù "Curve" per decidere se interpolarli con

un metodo globale o locale; nel primo caso, i punti verranno interpolati per mezzo della risoluzione di un sistema lineare mentre nel secondo caso, verrà individuata la curva di Bézier di grado 3 che li interpola 2 a 2.

Per la valutazione della curva sono disponibili due metodi: l'algoritmo di de Casteljau e la valutazione per mezzo dei polinomi di Bernstein; di default la curva viene valutata per mezzo dell'algoritmo di de Casteljau ma è possibile modificare questa opzione per mezzo del relativo menu "de CastBern" presente nella barra dei menu.

Nel momento in cui invece si vuole eliminare una curva o un insieme di punti, vengono richiamate rispettivamente le funzioni "delete_curva" e "delete_points" che permettono di deallocare lo spazio precedentemente occupato e di impostare come curva attiva una delle rimanenti sull'area di disegno. Un' ulteriore possibilità per cancellare una curva è quella di utilizzare la funzione "reset_curve" che cancella tutte le curve disegnate. Quando nell'area di disegno sono presenti più di una curva e si vuole eseguire un'operazione su una curva diversa da quella attiva è necessario attivarla; la funzione che si occupa di ciò è la "active_curve" che in base al punto passato in input rende attiva la curva selezionata ponendo la variabile globale "active" uguale al valore della curva.

- *Funzioni di calcolo delle curve*

  All'interno del codice sono implementate numerose operazioni per il calcolo e la modifica delle curve tutte accessibili per mezzo delle funzioni associate ai bottoni e alle voci del menu "curve". Due delle operazioni più importanti sono quelle per la valutazione dell'area e della lunghez-

za di una curva di Bézier; queste funzioni contenute nel file "Bezier.c", richiamano la funzione "quant8" che è una routine automatica di tipo adattivo basata sulle formule di NewtonCotes relative ad 8 tratti e che restituisce, oltre al valore dell'area o alla lunghezza, una stima della grandezza dell'errore commesso, il numero di volte in cui la funzione è stata calcolata e un indicatore di attendibilità che ci permette di capire l'attendibilità della soluzione.

Altrettanto importanti sono le funzioni associate ai JOIN. Per mezzo delle voci JOINC0, JOINC1, JOING1 del menu "curve" è possibile raccordare due curve in modo tale da trasformarle in un' unica curva a tratti. Nel caso in cui si scelga la funzione JOINC0, la seconda curva verrà modificata affinché il suo primo punto di controllo coincida con l'ultimo della prima curva. Nel caso in cui si voglia una continuità di tipo G1, la funzione modificherà la seconda curva in modo tale che il penultimo punto della prima curva sia su una retta con il secondo punto della successiva curva, mentre nell'ultimo caso, l'algoritmo richiamerà le due funzioni spiegate precedentemente e poi imporrà che la derivata abbia anche ugual modulo. In tutti e tre i casi, al termine l'algoritmo richiama la funzione "unisci" che permette di trasformare le due curve in un unica curva a tratti.

Le funzioni appena spiegate a loro volta richiamano altri algoritmi che possono essere richiamati anche come operazioni singole come: "rotate" che permette di ruotare la curva attiva, "translate" che permette di traslarla e "scale_unif" che permette di scalarla. Un'ultima importante operazione che è possibile compiere su due curve è quella di determi-

narne i punti di intersezione per mezzo della funzione “intersect” che richiama il metodo di Bezier_clipping per determinare gli zeri di funzioni polinomiali e restituisce le informazioni riguardo al punto trovato.
È possibile inoltre richiamare la funzione “split” che permette sia di dividere una curva in base ad un punto assegnato col mouse, sia di dividerla in seguito ad una intersezione nel punto in questione.
Oltre a quelle appena citate esistono molte altre operazioni di modifica di una curva che ora elenchiamo:

  - “modify_cp” che, dopo aver selezionato con il mouse un punto di controllo, permette di spostarlo modificando di conseguenza la curva.
  - “vis_adaptive” che permette di visualizzare la curva con un approccio adattivo al pixel.

- *Funzioni di disegno delle curve*

  Gran parte delle funzioni che gestiscono la visualizzazione delle curve nel sistema sono implementate nel file ”draw.c”.
  Tra le più importanti ritroviamo “Bezier_draw_poly” e “ bezier_draw_points” che permettono di visualizzare la poligonale i controllo. Nello specifico “bezier_draw_points” permette di disegnare i punti di controllo in base al parametro ricevuto in input: se stiamo disegnando i punti della poligonale di controllo, verranno rappresentati con dei cerchi, altrimenti verranno disegnati con delle stelle o delle x in base alla loro funzionalità. Nel caso in cui invece si vogliano visualizzare le tangenti e le normali per mezzo dell’opzione nel menù curve, verrà richiamata la fun-

zione “bezier_draw_tangent” che dopo aver valutato la curva in base al metodo scelto, ne valuterà la derivata e disegnerà a video la tangente. Lo stesso procedimento è alla base della funzione “bezier_draw_normal” che permette di disegnarne le normali.

- *Funzioni di caricamento e salvataggio delle curve*
  Il MiniSystem offre la possibilità di salvare e caricare le curve in diversi formati che saranno esposti nei prossimi paragrafi. Per il momento ci limitiamo a descrivere le funzioni che li gestiscono. Per il salvataggio, si può decidere se salvare la curva in formato .db , in formato .all o in formato .ip. Nel primo caso, scegliendo la relativa opzione “saveCurve” del menù curve, il file verrà strutturato nel seguente modo:
  FILENAME
  DEGREE
  N.C.P

  Seguito da tutti i punti di controllo. Scegliendo questa modalità però è possibile salvare solo una curva polinomiale e non una curva a tratti. Salvando invece il file in formato .all, esso verrà strutturato nel seguente modo: la prima riga indicherà il numero n di punti della curva, le successive n righe rappresenteranno i control point e dalla riga n si descriverà la successiva curva nello stesso modo. Nel caso in cui invece si vogliano salvare una serie di punti, è necessario scegliere la relativa opzione e verrà creato un file con questa struttura:
  FILENAME
  N.P

Seguito dall'elenco di tutti i punti. Per quel che riguarda il caricamento, esistono cinque opzioni:

- Open Curve

  Carica una curva salvata in un file di tipo .db.

- Open All

  Permette di caricare una curva salvata in un file di tipo .all. Valuta i due valori della prima riga; se il primo è maggiore di 0 e il secondo è maggiore/uguale a 0, allora se il secondo è diverso da 1, si tratta di una curva a tratti, altrimenti si tratta di una curva semplice.

- Load gl.points

  Carica da file dei punti da interpolare con metodo di interpolazione a tratti locale.

- Open ps

  Carica un file in formato Postscript. Requisito fondamentale per la lettura del file è che inizi con PS.

- *Funzioni di visualizzazione*

  Per quel che riguarda la visualizzazione dell'area di disegno, tutte le operazioni possibili sono contenute nel menu "view" e in alcuni bottoni nella barra laterale.

  In particolare, è gestita la possibilità di fare diversi tipi di zoom modificando le impostazione della window. L'operazione "zoom" permette di decidere se effettuare uno "zoomIn" o uno "zoomOut" in base al tasto del mouse selezionato e, in entrambi i casi, la finestra viene ripropor-

zionata in base ai parametri passati in input.
Diversamente la funzione “zoom_area” permette di effettuare lo zoom di un’area selezionata col mouse; la funzione prende infatti in input il primo punto selezionato col mouse che rappresenta l’angolo sinistro in alto della nuova finestra e il secondo parametro che rappresenta l’angolo in basso a destra, e in base a questi due valori ricalcola la window. Ci sono poi le funzioni “standard_zoom” e “scale” che permettono rispettivamente di riportare i valori della window a quelli di default e di scalare una curva.

- *Funzioni di gestione della window e della viewport*
  Nel paragrafo precedente abbiamo visto come la window possa essere facilmente modificata, esistono però molte altre operazioni che si occupano della gestione di quest’ultima e della viewport.
  Come già detto nel file ”const.h” sono inizializzate le costanti SCREEN_W e SCREEN_ H che definiscono l’ampiezza della finestra schermo. Questa viene utilizzata per contenere una barra dei menù e dei button, oltre ad un’area di disegno che chiameremo tecnicamente viewport e che viene settata chiamando la funzione “draw_env” presente nel file ”screen.c” nella cartella controller;sempre in questo file sono presenti le funzioni “win_view” e “view_win” che permettono di effettuare rispettivamente il mapping window-viewport e viewport-window.
  Inoltre, nel momento in cui effettuiamo un “redraw” vengono richiamate le funzioni “clear_view” e “clear_screen” che permettono di cancellare la schermata e ricaricare l’ultima salvata.

Funzioni per lo scambio di messaggi

Uno degli obiettivi di questa tesi è quello di inserire il MiniSystem all'interno di un percorso didattico come supporto per gli studenti. A questo fine sono state aggiunte funzioni che permettono la visualizzazione di messaggi con lo scopo sia di far percepire agli studenti, oltre che l'aspetto grafico, l'aspetto matematico dei loro esercizi, sia di aiutarli a muoversi all'interno di una piattaforma a loro sconosciuta.
Vediamo nello specifico come questi nuovi algoritmi sono stati implementati e in che modo risultano utili al fine didattico. Per prima cosa è stato creato un nuovo file "mess.c" contenente tutte le dichiarazioni necessarie e l'implementazione di queste funzioni.
Sono stati dichiarati un array di caratteri e un array di stringhe che permettono di utilizzare la funzione Dialog già presente nel sistema e che permettono di tenere traccia dei messaggi utili da fornire all'utente.
Le funzioni implementate sono le seguenti:

- void info()

  Permette di ottenere tutte le informazioni sia su una curva semplice che su una curva a tratti selezionando il relativo bottone "InfoPoint" nella barra dei bottoni laterale. La seguente funzione non prende valori in input ma considera la variabile globale active che in ogni momento contiene la curva attiva nell'area di disegno e ne stampa le informazioni in una finestra di dialogo.

  Se non ci sono curve disegnate, la funzione restituisce un messaggio negativo.

- void visualizza()

  Permette di visualizzare come sono settati i parametri della viewport e della window ed eventualmente modificarli per mezzo della relativa funzione.

  Di default i parametri vengono impostati a [-10,10]x[10,-10] per la window e a 800 e 600 per la viewport.

  Al fine di poter gestire un'ulteriore opzione che permetta di modificare questi parametri, è stata cambiata la funzione Dialog sia per permettere l'acquisizione di valori negativi, sia per stampare a video più righe contemporaneamente.

- void stampa()

  Scopo di questa funzione è quello di far apparire a video dei messaggi per l'utente d'aiuto sull'utilizzo del MiniSystem.

  Per prima cosa viene creata una nuova SDL_Surface per mezzo della funzione "SDL_CreateRGBSurface" messa a disposizione da SDL, vengono impostati tutti i parametri e viene copiata con la funzione "SDL_BlitSurface" nella SDL_Surface già esistente screen. Infine la Surface screen viene aggiornata e viene eliminata la surface precedentemente creata. Ogni volta in cui si vuol fare apparire un messaggio si richiama la funzione stampa passandogli in input la stringa da stampare a video.

- void ricarica(int k, int l)

  Permette di far apparire tutte le informazioni sulla curva attiva nella parte bassa della finestra. Ogni volta in cui si inserisce una nuova curva

o se ne modifica una già disegnata, le informazioni vengono ricaricate permettendo all'utente di avere sempre un'idea precisa di quello che sta disegnando.
Nello specifico, l'algoritmo crea una nuova SDL_Surface come nella funzione stampa() ed in base al parametro passato in input fa apparire le relative informazioni sulla curva.
Prende in input due variabili :

- int k Che rappresenta il numero di control points definiti (parzialmente se sto disegnando la curva o totali se ho terminato).
- int l Che rappresenta il tipo di curva che sto disegnando.
    * Se l è uguale a 1 significa che non ho ancora terminato di disegnare la curva e quindi i punti di controllo e tutte le relative informazioni sono ancora parziali.
    * Se l è uguale a 2 significa che ho terminato di disegnare la curva e quindi oltre alle informazioni precedenti, stampo anche l'area e la lunghezza di quest'ultima
    * Altrimenti, significa che ho disegnato una curva a tratti e quindi stampo il tipo di continuità salvato nella variabile globale join ed il grado della prima e della seconda curva.

Questa funzione risulta particolarmente importante in quanto mette in luce non solo l'aspetto grafico ma anche all'aspetto matematico che sta alla base delle operazioni decise dall'utente.

I file per il salvataggio delle curve

Le curve, come abbiamo visto precedentemente, possono essere salvate o caricate per mezzo delle funzioni contenute nel file io.c. Possono essere salvate in 4 differenti formati:

- ip_file sono file che contengono le coordinate di punti 2D utili per definire una curva per interpolazione.
- db_file è il formato proprietario per curve nurbs 2D di XCModel; in ogni file è descritta in un formato standard una curva polinomiale. Il formato è stato semplificato togliendo l'informazione coordinata peso tipica delle nurbs.
- all_file è un formato per liste di curve polinomiali e curve polinomiali a tratti ma esclusivamente cubiche.
- ps_file si tratta del formato postscript vettoriale di Adobe System; questo formato può solo essere letto e non salvato, come invece tutti gli altri.

3.1.3 Motore grafico

Il motore grafico è il componente che fornisce contenuti da visualizzare all'applicazione. Inoltre una volta inizializzato, non necessita di interventi esterni da parte dell'utente. Nel caso specifico del Mini-System, la GUI è nettamente separata dalle funzioni di calcolo e dalle funzioni di disegno; c'è un control-

ler che traduce i comandi di input (gui) in chiamate a funzioni di calcolo. Analizziamo ora in modo più approfondito la GUI del MiniSystem.

La GUI

Acronimo di "Graphical User Interface" indica una qualsiasi interfaccia utente che utilizzi elementi grafici per permettere l'accesso ai servizi forniti da un'applicazione o da un sistema operativo e la comunicazione di dati. La GUI utilizza un ambiente grafico evitando così all'utente di dover imparare una serie di comandi.

Nello specifico, la gui del MiniSystem è composta da un'area di disegno, una barra di menù a tendina contenente sei menù e posizionata in alto sulla finestra e una barra di due colonne di bottoni, situata sulla parte sinistra della finestra.

Figura 3.3: La GUI del MiniSystem

Il codice sorgente della GUI di Mini System comprende una directory radice contenente il file main.c e il Makefile, e tre sottodirectory: GCGraLib, controller e gui. Il main.c contiene l'inizializzazione e l'esecuzione dell'interfaccia grafica mentre le tre sottodirectory comprendono le funzioni di seguito indicate:

- GCGraLib contiene una libreria grafica di interfaccia con le SDL.
- Controller contiene una serie di funzioni :
  - Per gestire gli eventi come la pressione o il rilascio del mouse;
  - Per gestire gli eventi come la pressione o il rilascio del mouse;
  - Di callback per le operazioni dei bottoni e dei menù;
- Gui contiene le funzioni di inizializzazione e di visualizzazione dell'interfaccia grafica, dei bottoni, della barra dei bottoni, dei menù e della barra dei menù.

La libreria grafica SDL

è una libreria scritta in C "Cross-Platform Multimedia" ovvero che permette di creare un'applicazione solo volta e farla girare su più sistemi operativi. Fornisce accesso a basso livello alla tastiera, al mouse, al joystick e al 2D hardware per mezzo della libreria openGL.

Creata nel 1997 da Sam Lantinga, è stata utilizzata per effettuare il porting di alcuni famosi videogiochi dal sistema windows a Linux.

La libreria è open-source; in particolare è rilasciata sotto licenza GNU LGPL, dunque può essere utilizzata per lo sviluppo di applicazioni commerciali, anche closed-source, a patto che le modifiche apportate al codice sorgente del software stesso vengano rese pubbliche e che sia inserito un linking dinamico alla libreria stessa.

La libreria è suddivisa in otto sottosistemi: Video, Audio, CD-ROM, Joystick, gestione eventi, I/O file, processi e Timer. Oltre a questi sottosistemi, vi sono delle librerie ufficiali che aggiungono delle altre funzionalità:

- SDL_image
- SDL_mixer
- SDL_net
- SDL_rtf
- SDL_ttf

In particolare SDL_image e SDL_ttf sono state utilizzate all'interno di MiniSystem, per gestire formati di immagine diversi da BMP e per fare il rendering dei font TrueType. Una struttura molto importante utilizzata all'interno del MiniSystem, è la SDL_Surface. Essa rappresenta un'area grafica di disegno come ad esempio una finestra dello schermo. Una struttura di questo tipo ha una width, una height, un pixel format e altri parametri che non stiamo a enunciare.
La SDL_Surface è inoltre alla base di molte funzioni SDL che operano su

di essa; un esempio è la SDL_FillRect che riempe la SDL_Surface, precedentemente creata, con un colore o la SDL_BlitSurface che copia una intera Surface o solo una sua parte, in un'altra surface. Al termine dell'utilizzo, la SDL_Surface deve essere eliminata per mezzo della chiamata SDL_FreeSurface. Con la libreria SDL_image si può gestire il caricamento e il disegno di immagini in molti formati grafici come BMP, GIF, JPG, PPM, ecc. La funzione

$$SDL\_Surface * IMG_Load(constchar * file)$$

Carica un file per essere usato come un'immagine in una nuova surface. Questa inoltre richiama la IMG_LoadTyped_RW, con l'estensione utilizzata per il file e ciò permette di includere tutti i tipi supportati.
Al termine dell'operazione, ritorna un puntatore all'immagine come una nuova SDL_Surface.
Particolarmente importante è anche la struttura SDL_Event che permette di memorizzare gli eventi generati dalla tastiera e dal mouse. Gli eventi vengono messi in una coda con politica FIFO e per controllare se ci sono eventi attivi, si utilizza la funzione SDL_PollEvent. Riportiamo la struttura SDL_Event.

```
typedef union {
Uint8 type;
SDL_ActiveEvent active;
SDL_KeyboardEvent key;
SDL_MouseMotionEvent motion;
SDL_MouseButtonEvent button;
```

SDL_JoyAxisEvent jaxis;

SDL_JoyBallEvent jball;

SDL_JoyHatEvent jhat;

SDL_JoyButtonEvent jbutton;

SDL_ResizeEvent resize;

SDL_ExposeEvent expose;

SDL_QuitEvent quit;

SDL_UserEvent user;

SDL_SysWMEvent syswm;

} SDL_Event;

GCGralib

GCGraLib è una libreria scritta in C, che permette di fare grafica 2D in modo semplice, sfruttando le funzionalità. messe a disposizione dalla libreria SDL. Le principali funzioni sono:

- void GC_PutPixel(SDL_Surface *surface, int x, int y, Uint32 pixel);
- Uint32 GC_GetPixel(SDL_Surface *surface, int x, int y);
- void GC_DrawLine(SDL_Surface *s, int x0, int y0, int x1, int y1, Uint32 color);
- void GC_HorizLine(SDL_Surface *s, int x0, int x1, int y, Unit32color);
- void GC_VerticLine(SDL_Surface *s, int x, int y0, int y1, Uint32 color);

- void GC_DrawRect(SDL_Surface *s,int ax,int ay,int width,int height, Uint32 color)

Definizione di Window e Viewport

Prima di terminare con la descrizione del motore grafico ritengo importante definire i concetti di window e di viewport che poi verranno ripresi nel capitolo successivo.

Definiamo window un 'area rettangolare del piano di disegno che tipicamente contiene ci. che viene disegnato da noi.

Definiamo viewport un'area rettangola dello schermo su cui vogliamo rappresentare il disegno.

3.2 Guida al MiniSystem

Come abbiamo già illustrato, MiniSystem offre varie possibilità di utilizzo, dalla semplice creazione di una curve di Bézier al disegno di curve per interpolazione. Vediamo ora come questi problemi sono trattati nello specifico.

3.2.1 Creazione e gestione curve di Bézier semplici

Il software presentato offre molteplici opzioni di creazione e modifica di una curva. Analizzando nel particolare il caso delle curve di Bézier semplici è possibile modificarle o ottenere dati su di esse in modo immediato. Innanzitutto, è stata inserita una finestra di dialogo nella parte bassa della finestra

che permette, ogni qual volta si inserisce una nuova curva, di visualizzarne le informazioni principali come il numero di control point, il grado, le coordinate dell'ultimo punto di controllo inserito e altre informazioni. è possibile inoltre, per mezzo dei relativi bottoni nella barra a sinistra, modificarne i punti di controllo, effettuarne la suddivisione in un punto selezionato con il mouse, ruotarla o effettuare il filling e calcolarne l'area e la lunghezza attraverso le voci "area" e "length" del menù Curve. È stato poi introdotto un nuovo menù "de Castel-Bern" che permette di decidere se valutare la curva o con l'algoritmo de Casteljau o per mezzo dei polinomi di Bernstein visti nel Capitolo1.

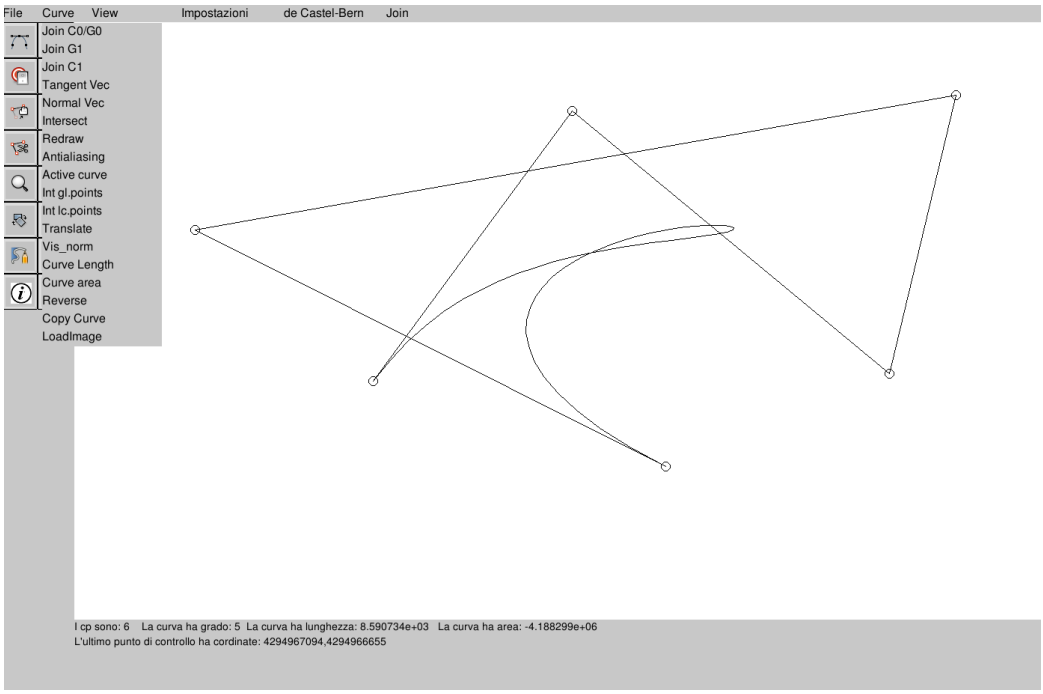

Figura 3.4: Curva di Bézier semplice

3.2.2 Creazione e gestione curve di Bézier a tratti

All' avvio, il menù è settato di default per una valutazione con l'algoritmo di de Casteljau.
Un' ulteriore aggiunta che è stata fatta al fine di aumentare le potenzialità del software, . stata l'implementazione della funzione "copia" che permette di creare una curva identica a quella attiva. È stata inoltre introdotta la possibilità di attivare o disattivare l'opzione vis-norm senza dover modificare il codice.

Le operazioni che il MiniSystem permette sulle curve di Bézier a tratti sono numerose. Oltre a quelle già elencate per le curve semplici, infatti, sono stati aggiunti nuovi algoritmi al fine di gestire al meglio i diversi tipi di continuità.. Vediamo le modifiche fatte e in che modo queste permettano di lavorare con le curve a tratti.

- È stato aggiunto un menù "Join" nella barra dei menù contenente tre sottomenù : JOINC0, JOINC1, JOING1. Questo nuovo menù permette all'utente di costruire non solo curve semplici, ma anche curve a tratti, possibilità prima non prevista dal software. Per fare ciò, l'utente dovrà prima di tutto selezionare il tipo di continuità desiderata dal menù JOIN e successivamente selezionare l'opzione " New curve" nella barra dei bottoni.
  A questo punto si sviluppano tre possibilità:

  - Per immettere un nuovo punto di controllo per quella curva, basterà premere il tasto sinistro sul punto desiderato.

- Per indicare che quello . l'ultimo punto della curva e che non vogliamo proseguire basterà premere il tasto destro del mouse.
- Per indicare che quello è l'ultimo punto della curva e desideriamo iniziare una nuova curva con la continuità selezionata dal menù JOIN, si premerà il tasto centrale del mouse o, nel caso in cui non sia presente, il tasto LSHIFT, e si comincerà così a disegnare la nuova curva di Bézier collegata alla precedente per mezzo del campo "struct Bézier_t *next ".

Al fine di implementare questa nuova funzionalità, è stata inoltre modificata la funzione "new_curva" presente nel file curve.c che precedentemente permetteva unicamente il disegno di una curva di Bézier semplice.

Attualmente nel momento in cui viene premuto il tasto centrale del mouse o, in alternativa il tasto SHIFT , viene richiamata una nuova funzione "UnioneTratti" che come prima cosa, poiché la continuità C0 è necessaria in tutti i casi, assegna come primo punto della nuova curva l'ultimo control point della curva precedente; dopo di ché valuta il tipo di continuità selezionata.

1. Se la continuità è C0, il controllo ritorna alla funzione "new_curva" che procede nel disegno degli altri punti;
2. Se la continuità è G1, calcola un ipotetico secondo punto della nuova curva, traccia un segmento che congiunge questo punto con il precedente, invita l'user a selezionare come secondo punto un punto sulla retta in questione e ritorna il controllo a new_curve; La

funzione “new_curve” successivamente verificherà che il punto sia effettivamente sul segmento e in caso contrario sposterà automaticamente il punto inserito sulle retta in modo tale che la direzione nel punto di congiunzione sia la stessa.

3. Se la continuità C1, calcola il secondo punto affinché la derivata prima nel punto di congiunzione sia uguale sia in direzione che in modulo e assegna questo punto come secondo control point della curva in questione.

- È stato implementato un nuovo algoritmo che permette, dopo aver costruito due nuove curve, di raccordarle con la continuità desiderata e ottenere così una sola curva a tratti. L’algoritmo in questione prende in input le due curve di Bézier già raccordate per mezzo dell’opzione JOIN e le trasforma in un’unica curva a tratti restituendo quest’ultima e liberando lo spazio non più usato.

- Gli algoritmi di calcolo dell’area e della lunghezza della curva sono stati modificati affinché fossero corretti anche per curve a tratti. È stato sufficiente inserire un nuovo controllo per capire se la curva attiva è singola o a tratti e in base a ciò proseguire nell’algoritmo.

- Infine tutti gli algoritmi per la gestione delle curve semplici, sono stati adattati affinché premettessero la gestione anche di curve a tratti.

3.2.3 Creazione e gestione curve di Bézier ottenute per interpolazione

Prima di soffermarci sulla descrizione delle funzioni che permettono di lavorare con questo tipo di curve, riprendiamo brevemente il concetto di interpolazione.

Interpolazione Lineare e interpolazione polinomiale

Sia data una sequenza di n numeri reali distinti e per ciascuno di questi sia dato un secondo numero. Ci proponiamo di individuare una funzione f tale che sia $f(x_k) = y_k$ per $k = 1..n$. Una coppia viene chiamata punto dato e f viene detta funzione interpolante, o semplicemente interpolante, per i punti dati. L'interpolazione lineare è rapida e facile, ma può risultare ben poco precisa. Il principale svantaggio dell'interpolante lineare sta per. nel fatto che essa non . derivabile nei punti .

Interpolazione Polinomiale

Per ovviare ai problemi dell'interpolazione lineare, possiamo provare a utilizzare un polinomio che costituisca un'interpolante per tutti i punti dati. Per loro stessa natura, le funzioni polinomiali sono continue e indefinitamente differenziabili in un qualsiasi intervallo, tuttavia si perde il vantaggio di poter calcolare in modo indipendente l'interpolante per ciascuna coppia di punti. L'interpolazione polinomiale presenta alcuni svantaggi come ad esempio il fatto che il calcolo del polinomio d'interpolazione sia molto costoso. Inoltre l'interpolazione polinomiale non risulta molto esatta nell'intero dominio della

funzione; in particolare nei punti estremi dell'intervallo si manifesta il cosiddetto fenomeno di Runge. Questi svantaggi possono essere evitati usando altre interpolazioni come l'interpolazione polinomiale a tratti. Una possibilità per valutare il polinomio interpolante è la seguente:

Forma di Bernstein:

Determina il polinomio interpolante mediante soluzione di un sistema lineare, ma a partire dalla base di Bernstein. Si può osservare che la matrice die polinomi base di Bernstein nei punti risulta meglio condizionata che non la matrice di Vandermonde.

Interpolazione polinomiale a tratti

Con interpolazione polinomiale a tratti si intende l'interpolazione di un set di dati su un intervallo con più polinomi di grado basso ciascuno dei quali definito in un sotto intervallo dell'intervallo dato. L'interpolante polinomiale a tratti è di classe C_k se è una funzione continua fino alla derivata di ordine k.
Esistono due metodi di interpolazione polinomiale a tratti molto usati nella pratica: interpolazione globale e interpolazione locale.

Interpolazione locale

Per interpolazione locale si intende un metodo che determina il polinomio a tratti interpolante $pp(x)$ ricavando ogni singolo polinomio indipendente-

mente dagli altri.

Interpolazione globale

Per interpolazione globale si intende un metodo che determina il polinomiale a tratti $pp(x)$ interpolante ricavando i polinomi in modo dipendente dagli altri.
Nel caso in cui si tratti di polinomi a tratti cubici con massima regolarità si dice interpolazione spline cubica.
Detto ciò, passiamo a esporre le principali funzioni per la gestione delle curve ottenute per interpolazione.
Il codice permette di gestire sia l'interpolazione globale che quella locale per mezzo delle voci "int gl.points" e "int lc.points" nel menù curve. Le funzioni in questione sono nel file "interp.c". L'algoritmo che permette di interpolare un set di dati in maniera locale prende in input una serie di punti e determina le curve di Bézier di grado 3 che li interpolano a 2 a 2; al contrario la funzione che interpola i dati in maniera globale, determina la curva di Bézier che li interpola mediante soluzione di un sistema lineare. Una nuova funzionalità aggiunta è la possibilità di caricare un' immagine in un qualsiasi formato e di individuare la curva che ne disegna i contorni dando in input i punti che definiscono l'immagine. Per fare ciò, è necessario salvare l'immagine nella cartella "img" del Mini-System e selezionare l'opzione "LoadImage" del menù curve. A questo punto basterà indicare il percorso dell'immagine e quest'ultima sarà caricata sulla viewport. La funzione in questione fa uso di una nuova SDL_Surface che viene copiata nella viewport e che nel momento

in cui premiamo il tasto destro del mouse e quindi terminiamo la curva, viene eliminata.

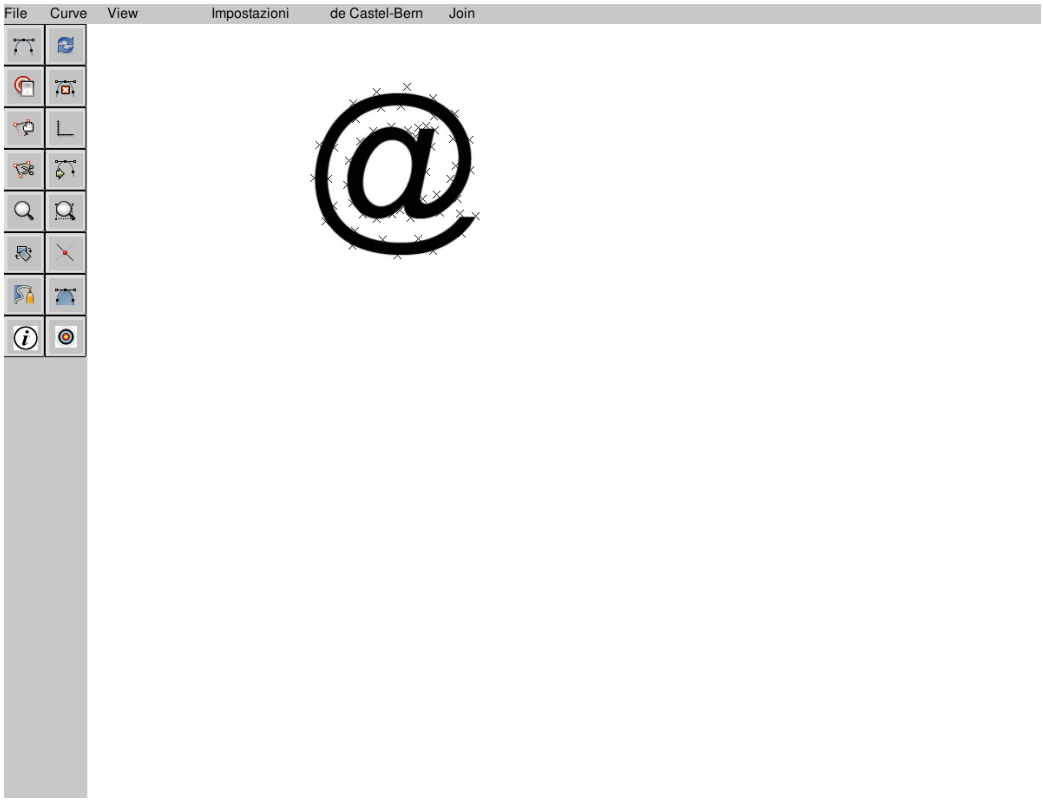

Figura 3.5: Esempio di inserimento di un'immagine

3.2.4 Salvataggio e caricamento delle curve

Per quel che riguarda il salvataggio, dopo aver disegnato a video la curva, si può decidere di salvarla selezionando la relativa opzione nel menu curve in base alla forma di quest'ultima. Le possibilità sono le seguenti:

- save_curve Permette di salvare una curva in formato .db.
- save_all Permette di salvare una curva in formato .all.

- load_ps Permette di salvare una serie di punti .

Dopo aver selezionato una delle voci appena elencate, comparirà una finestra di dialogo in cui basterà inserire il nome da dare al file che conterrà la curva e per confermare basterà premere il tasto Invio oppure fare clic sul bottone Ok. Ne caso in cui invece si voglia caricare un file preesistente, sarà necessario scegliere la relativa opzione, indicare il nome del file con il relativo percorso e una delle funzioni sottostanti si occuperà del caricamento.

- load_curve Carica un file in formato db_file.

- load_point Carica un file in formato ip_file

- load_ps Carica un file di tipo ps_file.

3.2.5 Modifica impostazioni di visualizzazione

Nel capitolo 2 abbiamo osservato che nel main.c viene definita la window iniziale, ossia la finestra sul mondo e per la precisione, che viene definita come un'area rettangolare [-10, 10] x [-10, 10].
Sono stati implementati degli algoritmi che permettono di modificare queste impostazioni attraverso una finestra di dialogo. È stato infatti introdotto un nuovo menù "Impostazioni" per mezzo del quale è possibile compiere alcune operazioni sulla viewport e sulla window. Il men ù Impostazioni è composto da 5 voci, noi analizzeremo unicamente le prime tre.

- VisualizzaParametri

  Permette di visualizzare come sono impostati i valori della window e

della viewport. Al fine di poter creare questa funzione, è stata modificata la funzione preesistente Dialog affinché potesse stampare a video non solo una riga di testo ma più righe contemporaneamente.

- Imposta Parametri

  Permette di reimpostare a proprio piacimento i parametri della window. Nel caso in cui si decida di non inserire alcun valore o si prema il bottone CANC, allora la variabile in questione verrà inizializzata a 0. Affinché l'utilizzo di questa opzione fosse ottimale, è stata modificata ulteriormente la funzione Dialog in modo tale da poter inserire anche valori negativi. Non è stata gestita la possibilità di modificare i valori della viewport in quanto per modificarli è sufficiente espandere o ridurre la finestra.

- ImpostaParametriDefault

  Permette di reimpostare i valori della finestra ai valori di Default.

Un ulteriore miglioramento è stato apportato alle funzioni che si occupano dello zoom aggiungendo un bottone "MostraPrimoPiano" che, dopo aver disegnato una curva ed aver fatto uno zoom, permette di tornare ai valori di visualizzazione della curva iniziali. La funzione in questione tiene infatti traccia dei valori di partenza in modo tale da reimpostarli nel caso in cui sia richiesto.

Figura 3.6: Menù "Impostazioni"

3.2.6 Possibili nuove funzioni del MiniSystem

Come mostrato nei paragrafi precedenti, il MiniSystem offre varie possibilità di progettazione e di disegno di curve 2D per mezzo di curve polinomiali e curve polinomiali a tratti nella forma di Bernestein/Bézier. In questa relazione sono state presentate unicamente le operazioni più comuni e maggiormente interessanti dal punto di vista teorico, ma all'interno del codice ne sono sviluppate molte altre e altrettante possono essere ancora implementate.

Una possibilità che non è stata presentata nel dettaglio è quella di caricare e modificare file .ps che, come abbiamo spiegato nel relativo paragrafo, corrisponde ad un formato postscript vettoriale di Adobe System; questo formato può essere letto, modificato ma non salvato. Interessante sarebbe quindi l'implementazione di un nuovo algoritmo che permetta di caricare a video un qualsiasi file .ps, modificarlo e risalvarlo permettendo così il ritocco di immagini vettoriali.
Lo stesso procedimento potrebbe essere applicato a file di tipo .pdf dopo aver gestito anche il caricamento ed il salvataggio di questi file.
Un esempio di ciò che è stato appena descritto è mostrato nelle immagini sottostanti; il file "snoopy.ps" è stato caricato, modificato e se fosse possibile salvarlo, si otterrebbe un'immagine del tutto simile alla precedente con le modifiche apportate.

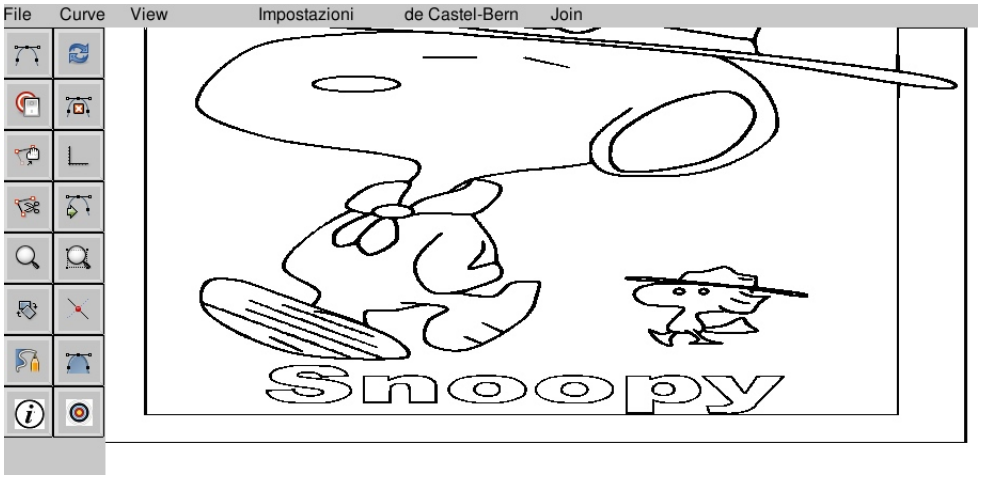

Figura 3.7: Immagine vettoriale in bianco e nero prima della modifica

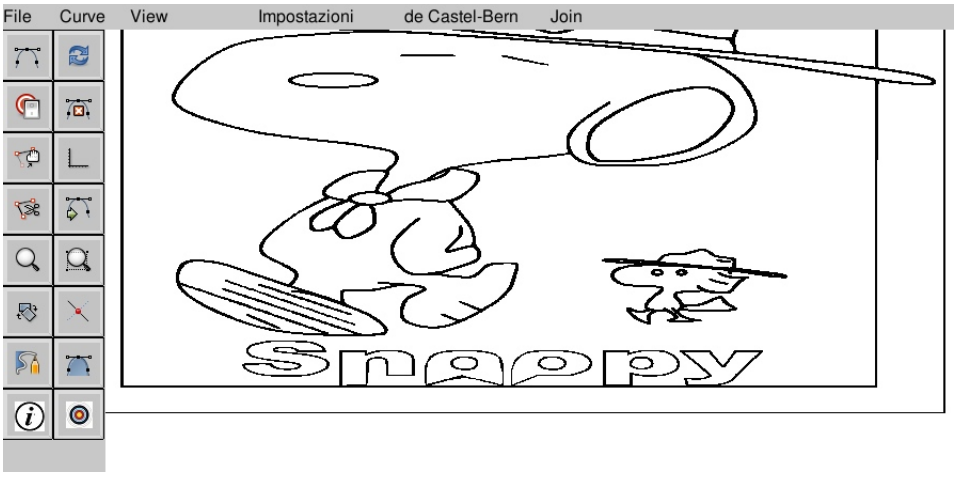

Figura 3.8: Immagine vettoriale in bianco e nero dopo la modifica

Conclusione

Questo elaborato descrive il lavoro di tesi che ha portato all' inserimento del MiniSystem come strumento di supporto e di esercitazione per gli studenti nell' apprendimento del "Calcolo Numerico".
L'obiettivo era quello di aumentare le potenzialità del MiniSystem rendendolo ancora più utile al fine didattico e aggiungendo la possibilità di gestire anche curve di Bézier a tratti; a questo fine è stato sia ipotizzato un possibile approccio studente–MiniSystem sia modificato il codice preesistente.
Il codice è stato modificato sia per rendere il software in questione più semplice ed intuitivo per gli studenti, sia per poter gestire anche curve di Bézier a tratti, che sono trattate nel programma di Calcolo Numerico ma che precedentemente non era possibile disegnare.
Per quel che riguarda l'approccio studente–MiniSystem, dopo un'attenta analisi dei metodi di studio degli studenti, sono stati implementati una serie di esercizi, da svolgere con l'ausilio del software in questione, con lo scopo di aiutarli ad apprendere in modo più semplice le nozioni spiegate e di fargli capire come le nozioni teoriche imparate possano essere messe in pratica.
Il lavoro svolto consentirà agli studenti di esercitarsi direttamente con il MiniSystem dandogli inoltre l'opportunità di applicarsi anche su argomenti del

corso che prima non erano trattati.

In futuro il MiniSystem potrà essere ulteriormente modificato al fine di prevedere deformazione di curve per animazioni, tracing (passaggio da immagini raster a vector), rendering (passaggio da immagini vector a raster) e altre numerose funzioni.

Appendice A

Esercitazioni

Esercitazione 1

Introduzione al Mini_system

Scaricare dalla pagina web del corso l'archivio "mini_system.tgz" e scompattarlo nella propria home directory. Verrà creata una cartella di nome "mini_system" contenente il codice ed alcuni file di esempio.
Prima di eseguirlo, è necessario scaricare la Libreria SDL 1.2 (vedi pagina web www.libsdl.org).
Per ulteriori specifiche, consultare il pdf "mini_system_1415" scaricabile dalla pagina del corso.

Si svolga la seguente esercitazione che ha come obiettivo quello di approfondire la propria conoscenza dell'ambiente mini_system.

Breve introduzione:

Operazioni possibili con il mini_system.

Operazioni associate ai bottoni:

New_curve	Permette di disegnare una nuova curva.
Active_curve	Permette di selezionare la curva da rendere attiva.
Cp yes/no	Permette di visualizzare o cancellare i cp.
Axes y/n	Permette di visualizzare o cancellare gli assi.
Delete Curve	Permette di cancellare la curva attiva.
Reset curve	Permete di cancellare tutte le curve presenti in quel momento.
Modify cp	Permette di modificare la posizione di un punto di controllo.
Split	Permette di suddividere la curva in un punto specifico.
Redraw	Ridisegna tutto ciò che è presente nell'area di disegno.
Filling	Riempe le regioni dello schermo racchiuse da curve di Bèzier.
Intersect	Permette di ottenere informazioni su due curve che si intersecano.
Rotate	Permette di ruotare la curva di Bèzier attiva.
InfoPoint	Permette di ottenere tutte le informazioni sulla curva.
MostraPrimoPiano	Permette di riportare la curva alla grandezza originale.

Operazioni associate alle funzioni nei menu:

Curve

Join	Permette di unire due curve in base alla continuità desiderata.
Normal vec	Permette di visualizzare i vettori normali alla curva attiva.
Tangent vec	Permette di visualizzare i vettori tangenti alla curva attiva.
Antialiasing	Permette di attivare o disattivare la visualizzazione con aliasing.
Translare	Permette di translare una curva nello spazio desiderato.
Int. lc.points	Permette di interpolare una serie di punti con una interpolazione locale a tratti
Int. gl.point	Permette di interpolare una serie di punti con una interpolazione globale a tratti.
Vis norm	Permette di visuliazzare una linea che congiunge gli estremi della curva
Reverse	Permette di invertire l'ordine dei punti di controllo della curva selezionata.
LoadImages	Permette di caricare un immagine sull'area di disegno.

View

Zoonin/out	Permette di fare lo zoom di un punto dell'area di disegno.
Zoom_Area	Permette di fare lo zoom di un'area selezionata.
Zoom default	Riporta la curva alla dimensione originale.

Impostazioni

Visualizza paramentri	Permette di visualizzare come sono settate alcune informazioni in quel momento.
Imposta paramenti	Permette di settare alcuni parametri(vxmin,vxmax,wxmax..).
Esegui test zeri	Procedura che esegue i 4 metodi (per trovare gli zeri) n volte e ne calcola il tempo.
Parametridefault	Permette di reimpostare le informazioni di default.

Casteljau-Bernstein

Permette di selezionare la modalità di studio della curva di Bèzier.

Join

Permette di selezionare il tipo di continuità desiderata nel caso in cui si voglia disegnare una curva di Bèzier a tratti.

File

Open curve:	Carica una curva da file.
Save curve:	Salva una curva su file .
Open .all:	Carica una curva da file .
Save .all:	Salva una curva su file.
Load gl.points:	Carica da file dei punti da interpolare con metodo di interpolazione a tratti globale.
Load lc.points:	Carica da file dei punti da interpolare con metodo di interpolazione a tratti locale.
Save points:	Salva dei punti su file.
Open ps:	Carica un file in formato Postscript.

Le cartelle **ip file**, **db file**, **all file**, **ps file**, contengo esempi di curve in diversi formati. La gestione, il caricamento e il salvataggio di queste ultime avviene per mezzo del codice presente nel file **io.c**. Vediamo brevemente i diversi tipi di file:

• **ip file** sono file che contengono le coordinate di punti 2D utili per definire una curva per interpolazione;

• **db file** èil formato proprietario per curve nurbs 2D di XCModel; in ogni file è descritta in un formato standard una curva polinomiale. Il formato è stato semplificato togliendo l'informazione coordinata peso tipica delle nurbs;

• **all file** è un formato per liste di curve polinomiali e curve polinomiali a tratti esclusivamente cubiche;

• **ps file** si tratta del formato postscript vettoriale di Adobe System; questo formato pu`o solo essere letto e non salvato, come invece tutti gli altri.

Per ulteriori specifiche, consultare il pdf "minsys" presente nella pagina del corso.

Quesiti:

1)

A(1.0 , 2.0) B(5.0 , 7.0) C(7.0 , 5.0)

Creare un nuovo file "prova.all" contenente i punti elencati e salvarlo nella cartella "all_file" della cartella mini_system.

Aprendo il file per mezzo dell'opzione " Open All(.all) ", verrà disegnata la curva di Bèzier associata e la relativa poligonale di controllo.
Valutarla prima con l'algoritmo di DeCasteljau e poi con l'algoritmo di Bernstein.

2)
Data la curva di Bèzier ottenuta precedentemente, calcolarne la lunghezza e l'area per mezzo delle funzioni "Curve area" e "Curve length" presenti nel menu "Curve".

3)
Mantenendo la curva costruita al punto 1, assegnare per mezzo del bottone "new curve" i control points di una nuova curva in modo tale che le due curve si intersechino.
Individuare poi, per mezzo della funzione "intersect", le informazioni sui punti di intersezione e farsi stampare a video le informazioni sulle due curve (area, lunghezza, cp, ….).

4)
Per mezzo della funzione "split" suddividere le due curve nei punti di intersezione e dopo aver selezionato una delle 4 curve ottenute, visualizzarne i vettori tangenti e i vettori normali.

5)
Date due curve di ugual grado, è possibile effettuare un join con continuità C0, C1 o G1, scegliendo la voce corrispodente dall'apposito menu "Curve".
Disegnare quindi due nuove curve, fare due copie di ciascuna curva ed effettuare su una coppia un join di tipo C0, su un altra un join di tipo C1 e sull'ultima un join di tipo G1.
Analizzare le differenze tra i tre join.

6)
Cancellare tutte le curve costruite fino questo momento.
Selezionare l'opzione "int_lc_points" dal menu Curve e assegnare i 5 punti da interpolare;
a questo punto compariranno le curve di Bèzier di grado 3 che interpolano i punti dati a 2 a 2.
Effettuare nuovamente la stessa operazione, dando in input 5 punti diversi, affinchè le due curve si intersechino in 2 punti.

7)
Disegnare una curva a tratti selezionando precedentemente il tipo di continuità desiderata dal menù "Join".

8)
Caricare un'immagine nel formato .jpeg per mezzo della funzione "load image" presente nel menu "Curve"e riprodurre quest'ultima dando in input i punti che ne formano i contorni ed ottenendo la curva di Bèzier che li interpola.

9)
Al termine visualizzare se le impostazioni di base sono state modificate e in quest'ultimo caso riportarle allo stato iniziale.

Esercitazione 2

Valutazione e disegno curve di Bèzier e poligonali di controllo

1)
Scrivere l'equazione della curva di Bèzier avente i seguenti punti di controllo:

P0 (1, 0), P1 (1.5, 3), P2 (3, 4), P3 (7, 5) e P4 (6, 9).

Disegnare poi sia la curva di Bèzier che la relativa poligonale di controllo.

2)
Dato il punto P5 (5, 2) valutare se è interno o esterno alla curva costruita al punto 1.
Implementare poi un algoritmo che esegua questa istruzione e testarlo sul mini_system.

3)
Data la curva costruita al punto 1, modificarla affinchè la sua area aumenti del doppio.
Implementare poi un algoritmo che esegua questa istruzione e testarlo sul mini_system.

4)
Data la curva costruita al punto 1, modificarla affinche la sua lunghezza aumenti del doppio.
Implementare poi un algoritmo che esegua questa istruzione e testarlo sul mini_system.

5)
Cancellare la curva precedente.
Costruire due curve con due punti di intersezione e calcolare l' area compresa tra queste ultime.

6)
Una proprietà delle curve di Bèzier è quella per cui, se la curva è contenuta in un piano, allora non esistono segmenti che intersecano la curva più volte di quanto intersechino la poligonale di controllo.
Verificare questa proprietà.

7)
Una proprietà delle curve di Bèzier assicura che la curva di Bézier, definita da n punti di controllo assegnati, giace completamente nel dominio convesso formato dai punti di controllo. Il dominio convesso di un insieme di punti è il più piccolo insieme convesso che contiene tutti i punti.
Verificare questa proprietà.

8)
Nel codice non è presente questo algoritmo, provare ad implementarlo.

Supponiamo di aver disegnato una curva di Bézier di grado n e di voler modificare solo una parte della curva.
Dobbiamo cercare quindi di aggiungere un ulteriore punto di controllo lasciando però inalterata la forma della curva.
Per fare ciò, sarà necessario lasciare il primo e l'ultimo punto di controllo immutati e aggiungere nuovi punti intermedi.

9)
Implementare la valutazione della derivata seconda di un polinomio nella base di Bernstein e utilizzarla per disegnare la curvatura a pettine sulla curva. Per una curva piana c(t) = (x(t), y(t)) la funzione curvatura è data da:

$$k(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'^2 + y'^2)^{3/2}}.$$

La curvatura a pettine consiste nel riportare, per ogni punto della curva, sul vettore normale alla curva per quel punto, il valore con segno della curvatura, e nel congiungere questi punti per ottenere una curva a pettine.

Esercitazione 3

Valutazione e disegno curve di Bèzier definite per interpolazione

1)
Determinare la curva di Bèzier che interpola i seguenti punti:

P0 (2 , 1), P1(3 , 4) e P2 (2 , 1)

Disegnare poi sia la curva di Bèzier che la relativa poligonale di controllo.

2)
Implementare un algoritmo che calcoli l'area compresa tra la curva e la sua poligonale di controllo.

3)
L'area di una curva sarà positiva o negativa a seconda del verso di percorrenza della curva. Appurato questo prevedere una nuova opzione che scali la curva per avere un'area predefinita.

4)
All'interno del codice è presente una funzione "Vis_norm" che permette di disegnare una curva di Bèzier per mezzo di un algoritmo adattivo basato su suddivisioni ricorsive.
Analizzarla e provare ad implementarla.

5)
All'interno del codice è implementata la funzione "baricentro" che permette di trovare il baricentro del più piccolo rettangolo che contiene la poligonale di controllo.
Provare ad implementarla.

6)
Disegnare una nuova curva di Bèzier per interpolazione e implementare un algoritmo che ne determini i punti estremi.

7)
Riprendendo la curva costruita al punto precedente, ruotarla per mezzo del bottone "rotate".
Stabilire se la curva mantiene le proporzioni. In caso di risposta negativa, implementare un algoritmo che risolva questo problema.

8)
Presa la curva costruita precedentemente, e un punto Q(6 , 5) del piano, si implementi un algoritmo per determinare la distanza del punto dalla curva.

9)
Per ogni metodo di interpolazione presente, gestire l'interpolazione di curve chiuse e periodiche.

Esercitazione 4

Valutazione e disegno curve di Bèzier a tratti

1)
Dati i seguenti punti, determinare le curve di Bèzier di grado 3 che li interpolano a 2 a 2.

P0 (1, 0), P1 (2, 3), P2 (3, 4), P3 (4, 5), P4 (6, 9) e P5(10, 11)

2)
Le curve di Bèzier non sono in grado di riprodurre esattamente una circonferenza.
Un' approssimazione molto diffusa simula una circonferenza con 4 curve di Bèzier.
Disegnarla.
(Suggerimento - I punti di controllo devono essere posti perpendicolarmente al cerchio, ad una distanza tale da far passare il punto medio della curva per il punto che si trova ad un angolo di 45 sulla circonferenza.)

3)
Il problema dell'approssimazione della funzione trigonometrica seno (o coseno) viene, invece, affrontato imponendo il passaggio per 9 punti, di 8 curve di Bézier connesse tra loro.
Disegnarla.

4)
Il fenomeno di Runge mostra che, se la funzione sottostante, nell'intervallo [5,-5] ,viene interpolata con una funzione p(x) di grado <=n, all' aumentare del numero di nodi (equidistanti) p(x) non converge a f(x) ma l'errore aumenta.

$$f(x) = \frac{1}{1+x^2} \quad x \in [-5, 5]$$

Si può dimostrare che al crescere di n la successione dei polinomi non converge puntualmente ad f(x) e che i corrispondenti resti diventano in modulo arbitrariamente grandi.
Verificare il fenomeno di Runge.

5)
Molte funzioni sono indirizzate alla singola curva polinomiale e non ad una curva a tratti, come Tangent, Normal, ecc.. Geralizzarle affinchè operino correttamente anche nel caso di curve a tratti.

6)
L'andamento dell'errore all'aumentare dei nodi, che si manifesta per errori equispaziati, non si riscontra invece se vengono utilizzati i nodi di Chebyshev.

Tali nodi si ottengono suddividendo uniformemente una circonferenza e proiettando i punti in cui la circonferenza viene divisa sull'asse delle ascisse (ovvero sul diametro).
Le coordinate x di tali nodi quindi corrisponderanno ai punti di Chebyshev e non saranno più equispaziati sul diametro, pur essendo stati costruiti a partire da punti equispaziati sulla circonferenza.
Si implementi una funzione per il calcolo dei nodi di Chebyshev.

7)
La funzione intersect richiama la funzione CBèzier clipping per determinare i punti di intersezione.
Nel file zerip.c sono presenti pù algoritmi per determinare gli zeri di funzioni polinomiali.
Analizzarli ed usarne alcuni.

8)
Riprendi la curva costruita al punto 1 ed effettua il test degli zeri presente nel menu "Impostazioni" analizzandone il risultato.

9)
Una curva spline è costruita congiungendo curve polinomiali mantenendone continuità e regolarità.
In genere il tipo di continuità adottato è di tipo C2, questo significa che l'ultimo punto della curva antecedente ed il primo punto della curva successiva coincidono e la tangente e la curvatura delle due curve in quel punto è identica.
Cercare di implementare questo tipo di continuità.

Esercitazione 5

Interagire con il MiniSystem

1)
Non è gestita la possibilità di definire curve chiuse, né polinomiali, nè polinomiali a tratti; una volta definita una curva chiusa (primo ed ultimo punto di controllo coincidenti) si potrebbe prevedere che la chiusura sia C0, G1, C1, ecc.

2)
Interpolazione polinomiale cubica a tratti C1 “int lc points” ; intervenire sulla parametrizzazione e scelta della stima delle derivate.

3)
Le varie tecniche di join di curve (C0, G1, C1, ecc.) permettono di raccordare due curve, ma queste continuano ad essere separate; gestire il join in modo che una volta che due curve siano state raccordate venga vista dal sistema come una sola curva a tratti.

4)
Analizzare l’opzione Curve Length per calcolare la lunghezza della curva attiva. Calcola la lunghezza di una curva polinomiale o anche di una curva a tratti? Eventualmente generalizzare.

5)
Ad Intersezione effettuata, se si richiede Split, le curve intersecate vengono splittate in corrispondenza dei punti di intersezione; in alternativa Split permette di splittare una curva in corrispondenza di un punto della curva individuato per picking del mouse. Sdoppiare le due funzionalità di splitting.

6)
Analizzare l’opzione Curve Area per calcolare l’area della curva attiva definita fra la curva e l’origine (se la curva è chiusa calcola l’area della regione interna). Calcola l’area di una curva polinomiale o anche di una curva a tratti? Eventualmente generalizzare.

7)
La matrice dei polinomi base di Bernstein nei punti è totalmente positiva, questo implica che la sua fattorizzazione LU non necessiti di scambi di riga. Verificarlo sperimentalmente.

Bibliografia

[1] G. Casciola. Documentazione del MiniSystem, A.A.2014/2015.

[2] G. Casciola. Dispensa di Calcolo Numerico, A.A.2014/2015.
http://www.dm.unibo.it/casciola/CORSI/disp_1415/disp_cn1415.pdf

[3] G. Cairo. Guida all'utilizzo della GUI di MiniSystem, grafica 2013/2014.

[4] G. Casciola. mini_system_1415, A.A.2014/2015.

[5] SDL. Simple DirectMedia Layer.
http://www.libsdl.org