Fractals ;)

Autori:
Lorenzo Bossi
Stefano Massera

Università degli studi dell'Insubria

2 gennaio 2014

Indice

Capitolo 1

I frattali

I frattali, termine coniato nel 1975 dal matematico polacco Benoît Mandelbrot, sono figure geometriche caratterizzate dal ripetersi sino all'infinito di uno stesso motivo su scala sempre più ridotta. Questa è la definizione più intuitiva che si possa dare di figure che in natura si presentano con una frequenza impressionante ma che non hanno ancora una definizione matematica precisa: l'atteggiamento corrente è quello di considerare frattale un insieme F che abbia proprietà simili alle quattro elencate qui di seguito:

1. *Autosimilarità*: il frattale è unione di un numero di parti che, ingrandite di un certo fattore, riproducono tutto F; in altri termini F è unione di copie di se stesso a scale differenti.

2. *Struttura fine*: il frattale rivela dettagli ad ogni ingrandimento.

3. *Irregolarità*: il frattale non si può descrivere come luogo di punti che soddisfano semplici condizioni geometriche o analitiche. (la funzione è ricorsiva: $F = \{Z|Z = f(f(f(...)))\}$

4. *Dimensioni di autosimilarità > della dimensione topologica* La caratteristica dalla quale deriva il loro nome è che, sebbene esse possano essere rappresentate (se non si pretende di rappresentare infinite iterazioni, cioè trasformazioni per le quali si conserva il particolare motivo geometrico) in uno spazio convenzionale a due o tre dimensioni, la loro dimensione non è intera. In effetti la lunghezza di un frattale *piano* non può essere misurata definitamente, ma dipende strettamente dal numero di iterazioni al quale si sottopone la figura iniziale.

1.1 Insieme di Mandelbrot

L'insieme di Mandelbrot è un luogo geometrico che si colloca al centro di una vasta distesa bidimensionale di numeri detta piano complesso e che soddisfa la legge di Mandelbrot:

La successione dei numeri complessi c definita come $Z_{n+1} = z_n^2 + c$ non è divergente.

Quando si applica ripetutamente ai numeri la serie appena scritta, quelli all'esterno dell'insieme fuggono all'infinito, mentre quelli all'interno vanno alla deriva ondeggiando qua e là. Vicino al margine, le oscillazioni dei numeri segnano l'inizio dell'instabilità. Mandelbrot scoprì questo tipo di funzione matematica quasi per caso mentre era ricercatore al Thomas J. Watson Research Center della IBM a Yorktown Heights, New york.

Partendo dal suo lavoro sulle forme geometriche, Mandelbrot ha sviluppato un campo che ha chiamato geometria frattale, cioè lo studio matematico di forme con dimensione frazionaria. Un secondo merito da attribuire al grande Benoit è quello dell'aver definito questa geometria come *geometria della natura*. In particolare il confine dell'insieme di Mandelbrot è un frattale. In linea di principio si può effettuare uno zoom su qualsiasi parte dell'insieme e all'ingrandimento che si desidera: teoricamente l'ingrandimento che si può raggiungere utilizzando un calcolatore, è di molto superiore a quello necessario a risolvere il nucleo di un atomo.

Un modo per rappresentare graficamente l'insieme di Mandelbrot (immagine 1.1) è prendere in considerazione ogni punto del piano dei numeri complessi, ad ogni punto applicare la successione per poi colorarlo in base all'andamento della successione.

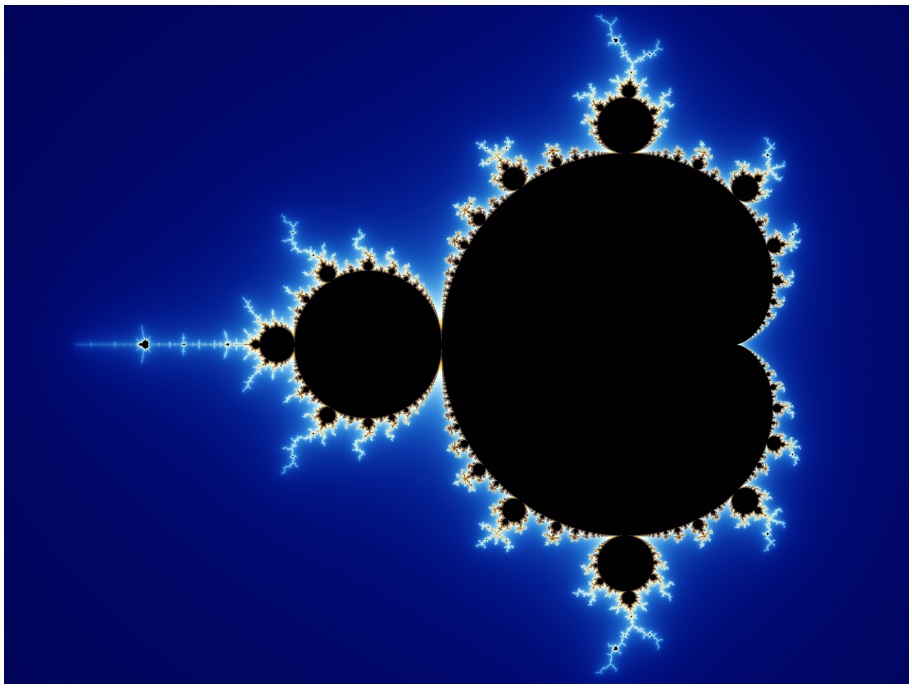


Figura 1.1: Rappresentazione grafica dell'insieme di Mandelbrot.

Guardando le immagini bisogna tenere presente che tutti i punti di colore diversi dal nero non appartengono all'insieme di Mandelbrot. La bellezza di queste immagini sta in gran parte nell'alone di colori assegnati ai punti in fuga. L'insieme è coperto da filamenti e miniature di se stesso. In realtà nessuno dei mini Mandelbrot è una copia esatta dell'insieme genitore e nessuno di essi è uguale ad un altro. Ogni quadrato della regione di confine ne racchiude infinite

di queste miniature, di cui nel migliore dei casi, solo qualcuno è visibile con un ingrandimento scelto casualmente.

L'insieme di Mandelbrot può essere così considerato l'oggetto più complesso della matematica.

1.2 Insieme di Julia

Definito dal matematico francese Gaston Maurice Julia intorno al 1978 a partire da una funzione olomorfa[1], consiste di tutti quei punti il cui comportamento dopo ripetute iterazioni della funzione è caotico, nel senso che può cambiare drasticamente in seguito ad una piccola perturbazione iniziale.

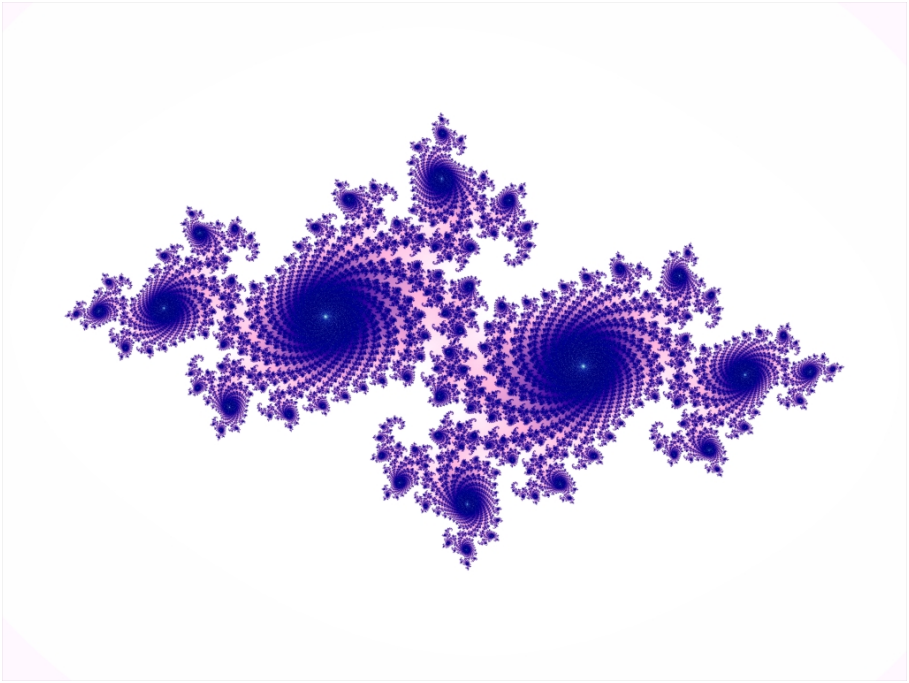


Figura 1.2: Rappresentazione grafica dell'insieme di Julia.

Gli insiemi di Julia hanno moltissimi punti in comune con l'insieme di Mandelbrot, tant'è che quest'ultimo permette di indicizzarli (immagine 1.3). Ad ogni punto del piano complesso corrisponde un diverso insieme di Julia, tale l'insieme è connesso se il punto appartiene all'insieme di Madelbrot. Gli insiemi di Julia non banali corrispondono ai punti che descrivono il bordo dell'insieme di Mandelbrot, mentre quelli interni descrivono gli insiemi più semplici e i punti esterni, lontani dal bordo, generano piccoli insiemi di Julia formati da piccoli insiemi connessi.

[1] Funzione definita su un sottoinsieme aperto del piano dei numeri complessi $\mathbb{C}$ con valori in $\mathbb{C}$ che sono differenziabili in senso complesso in ogni punto del loro dominio.

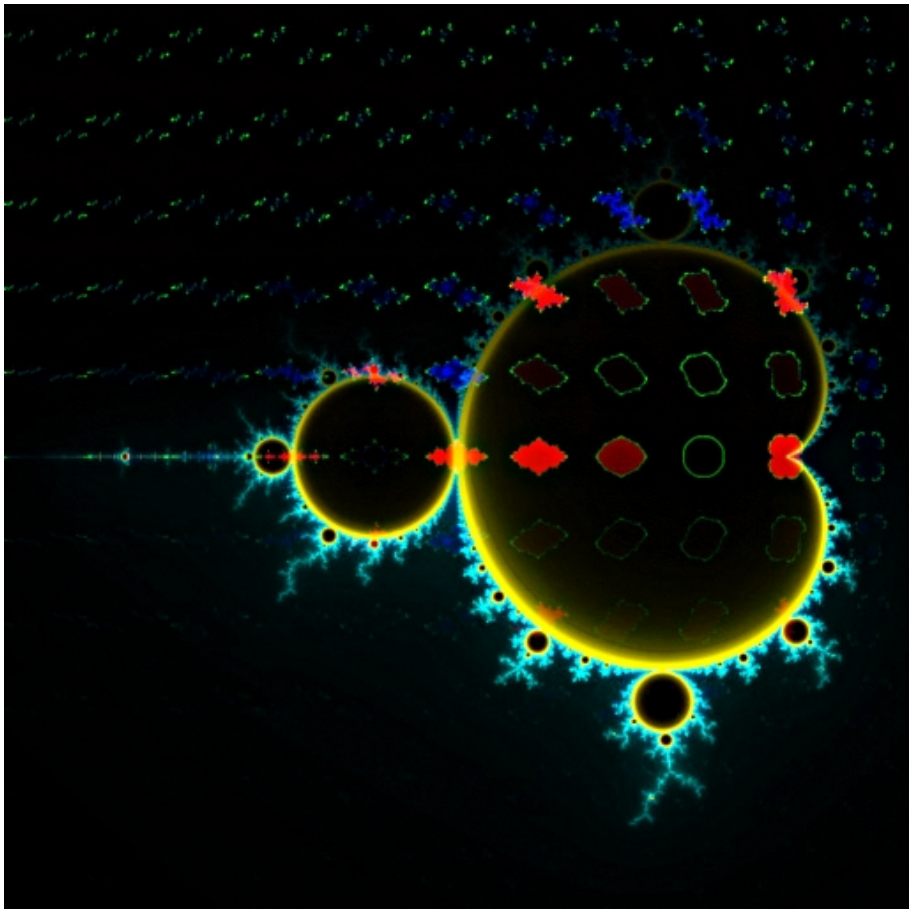


Figura 1.3: Alcuni insiemi di Julia al variare di c nell'insieme di Mandelbrot.

1.3 Utilizzo

I frattali sono molto utilizzati in matematica e in Computer Graphics per descrivere figure decisamente complesse e difficili da campionare. Tali figure appaiono frequentemente in natura, come la forma delle coste in foto satellitari, la struttura di un cavolfiore o della superficie polmonare.

Una costa è un esempio molto facile da capire riguardo alla complessità di un frattale poiché, vista da satellite, potrebbe essere zoomata quasi all'infinito senza avere mai il dettaglio minimo (vedi immagine 1.4).

Un altro esempio immediato di frattale presente in natura è il cavolfiore, dove ogni foglia è simile all'intera figura e viceversa, inquanto ogni ramo porta 13 rami 3 volte più piccoli (vedi immagine 1.5).

Figura 1.4: Particolare della costa della Svezia.

Figura 1.5: Struttura autosimile del cavolfiore

Capitolo 2

Il programma

Il progetto ha avuto come obiettivo lo sviluppo di un programma che permettesse la generazione di frattali in scala di grigi, l'applicazione di una scala di colori ed infine la generazione di range images a partire dai disegni dei frattali.

Durante lo sviluppo ci si è però accorti che alcuni frattali (in particolar modo molti insiemi di Julia) presentavano poche sfumature di grigi, privilegiando unicamente colori molto chiari e tendenti al bianco e più o meno grandi masse nere (a causa della tendenza non prevedibile dei numeri di *andare all'infinito*). Si è quindi introdotto un ulteriore modulo che permettesse un bilanciamento dell'istogramma dell'immagine tramite l'utlilizzo di una curva esponenziale.

2.1 Implementazione

Uno degli obiettivi era corredare il programma di una interfaccia grafica semplice ma accattivante che ne rendesse immediato l'utilizzo e facile la visualizzazione in anteprima dei frattali creati, senza dovere necessariamente salvarle su file e aprirle con un visualizzatore esterno.
Inizialmente si era pensato quindi al java, dato il supporto nativo per la creazione di GUI e la presenza di numerose classi per gestire il salvataggio e la rappresentazione in memoria delle immagini. Successivamente però, visto l'onerosità dei calcoli da effettuare, si è preferito cercare un linguaggio più efficiente e veloce. La scelta quindi è ricaduta sul C++. Per non dovere rinunciare ai vantaggi di una interfaccia grafica si sono usati per lo sviluppo le librerie QT4.2 di Trolltech® che contengono al loro interno anche classi per il salvataggio e la gestione delle immagini. Un altro punto a favore della scelta delle librerie QT4.2 è stato il cross plattaform, ovvero il supporto di tutti i maggiori sistemi operativi presenti in commercio (Linux, Unix, Windows, MacOS), rendendo quindi il codice compilabile senza bisogno di alcuna modifica in ambienti diversi.

Il programma nonostante possa visualizzare solo un'immagine per volta, è strutturato in maniera da tenerne in memoria un massimo di 4. In particolare le 4 immagini salvate sono: il frattale, il frattale con applicato il filtro esponenziale per la correzione dei grigi, il frattale colorato, il frattale in 3D. Ognuna di queste è calcolata da un modulo indipendente che necessita come input una serie di parametri e le immagini precedenti.

2.2 Generazione

Il modulo che genera i frattali ha bisogno dei seguenti input:

- la dimensione in pixel dell'immagine da creare
- gli estremi del piano di Argand-Gauss da rappresentare nell'immagine
- eventualmente il parametro complesso da usare come primo numero della successione per la creazione degli insiemi di Julia

Per comodità è stata inoltre inserita una combo box che permettesse la compilazione in automatico dei campi relativi alle caratteristiche del frattale (ovvero tutti i precedenti esclusa la dimensione delle immagini) per potere richiamare subito i frattali più caratteristici.
Per dare maggiore flessibilità al programma si è scelto di non inserire questi valori direttamente nel codice del programma, ma piuttosto di leggerli da semplici file di testo (con estensione *frt*) posizionati nella sottodirectory *frattali*. Una possibile aggiunta futura al programma potrebbe essere la possibilità di salvare questi settaggi di default direttamente dal programma.

Come si può vedere nell'immagine 2.1, tutti i campi descritti precedentemente sono accessibili tramite i comandi presenti sul lato destro della finestra.

Una volta allocato spazio per la rappresentazione interna del frattale, si procede al calcolo del frattale vero e proprio scorrendo ogni pixel dell'immagine, ricavando le coordinate nel piano di Argand-Gauss, e applicando la successione di Mandelbrot per 100 volte o finché supera il valore 4. Finita questa fase, si controlla quante volte si sono iterati i passi precedenti, se si è raggiunto il massimo si colora il pixel di nero, se si è al minimo di bianco, in tutti gli altri casi del grigio ottenuto per interpolazione lineare dei due estremi.

2.3 Bilanciamento dei grigi

Il bilanciamento di grigi risulta essere un modulo abbastanza semplice. Si procede come segue: si alloca nuovo spazio in memoria per tenere traccia della nuova immagine, si controlla pixel per pixel il frattale precedentemente creato e applica al pixel preso in considerazione un semplice filtro esponenziale.

Nel dettaglio, la funzione esponenziale lavora nel seguente modo: se x è il valore di grigio dell'immagine di partenza, il valore dell'immagine di arrivo è $y = f(x) = be^{\frac{x}{a}} - 1$ con a e b due parametri legati tra loro per imporre il passaggio nei punti $(0, 0)$ e $(255, 255)$ (in modo da conservare rispettivamente il nero e il bianco). a è calcolato a partire da un valore preso in input (come si può vedere dall'immagine 2.2) tra 1 e 100 (1 correzione minima, 100 massima). Più precisamente $a = 100 - (parametro * 90/100)$ mentre $b = \frac{255}{e^{255/a}-1}$.

La scelta di questi range è stata arbitraria e dettata unicamente da numerose prove empiriche volte a cercare i valori che dessero risultati migliori. Mentre la scelta di usare proprio un filtro esponenziale è scaturita dall'analisi degli istogrammi di svariati frattali, che mostravano come spesso è predominante la presenza di molte sfumature di grigio molto chiare e nero, senza passare per grigi scuri.

```
aa=100-(i->correzioneGrigiVal()*90/100);
```

```
bb=255.0/(exp(255.0/aa)-1);

for(int a=0;a<img->get_x();a++){
  for(int b=0;b<img->get_y();b++){
    double x=img->get_px(a,b)[pixel::r];
    double y=bb*(exp(x/aa)-1);
    grigi->get_px(a,b).set((int)y);
  }
  o->progress ((int)(100*a/img->get_x()));
}
```

2.4 Colore

Questo modulo genera una nuova immagine a colori partendo dall'immagine in scala di grigio del frattale, applicando per ogni pixel una funzione $f : (R) \to (R)^3$ che associa ad ogni valore di grigio una tripletta (R,G,B).

Per mantenere il programma il più flessibile possibile, tale funzione non è definita nel codice ma si ottiene prelevando il colore dell'ennesimo pixel di un'immagine a colori.

Una volta attivato il modulo si deve infatti selezionare da un apposito menù a tendina (vedi immagine 2.2) quale scala applicare. Ogni nome presentato corrisponde ad un'immagine PNG nella sottodirectory *colori*. Supponiamo di avere un pixel con valore di grigio g, andremo a vedere di che colore è il pixel di coordinate $(0, g)$ e useremo tale colore nell'immagine risultante. In questo modo aggiungere nuove colorazioni nel programma significa semplicemente aggiungere immagini PNG di dimensione $256x1$ pixel. Se l'immagine è più grande i pixel aggiunti saranno ignorati, se è più piccola quelli mancanti saranno considerati neri.

```
colore=new image_2d (img->get_x(),img->get_y());
int red[256],green[256],blue[256];
i->getColori(red,green,blue);
for(int a=0;a<img->get_x();a++){
  for(int b=0;b<img->get_y();b++){
    int gg=img->get_px(a,b)[pixel::r];
    colore->get_px(a,b).set(red[gg],green[gg],blue[gg]);
  }
  o->progress ((int)(100*a/img->get_x()));
}
```

2.5 Range image

Questo modulo serve a creare la proiezione di un oggetto tridimensionale creato partendo da un'immagine bidimensionale dove si immagina di alzare ogni pixel in funzione della sua luminosità.

È indubbiamente il modulo che ha richiesto una maggiore attenzione nello sviluppo perché nonstante sia concettualmente semplice, ci si è scontrati con molti problemi in fase di implementazione che in alcuni casi hanno comportato

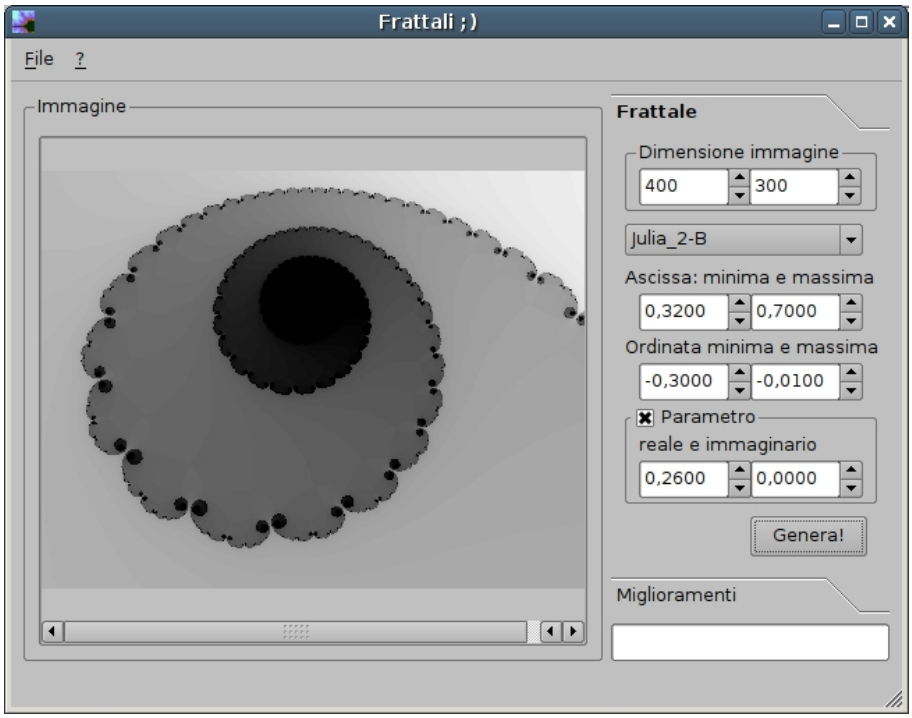


Figura 2.1: Screenshot della schermata di generazione di un frattale.

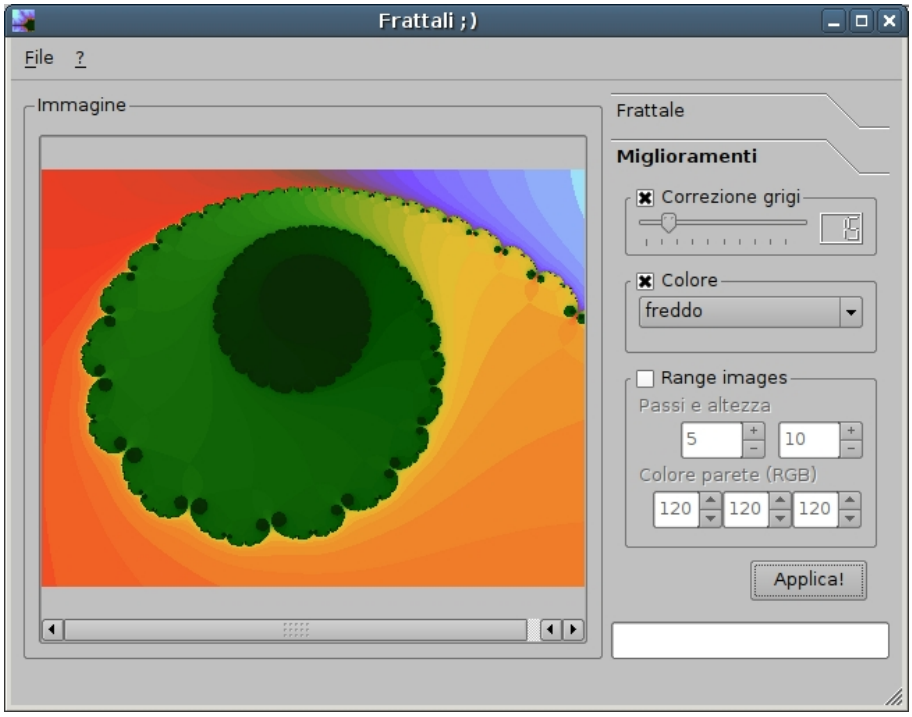


Figura 2.2: Screenshot del programma con il finto colore applicato.

la completa riscrittura del codice (che a differenza dei casi precedenti non verrà trascritto per intero per motivi di spazio).

L'idea iniziale era quella di rappresentare ogni punto dell'immagine alla base in un array tridimensionale. Scorrendo poi la base di tale array, analizzare ogni punto e alzarlo in base alla luminosità del punto. Questo è stato il metodo utilizzato per le prime versioni del programma, e ha permesso di notare alcuni difetti dell'idea di base.
Un primo problema incontrato era che calcolare l'altezza partendo dalla luminosità dell'immagine a colori, introduceva errori dovuti al fatto che la luminosità della scala di colori applicata non era necessariamente lineare con quella dei grigi. Si è subito optato quindi per una soluzione diversa: calcolare le altezze partendo dall'immagine in scala di grigi (eventualmente corretti con il filtro esponenziale), facendo si che a maggiore intensità di grigio corrispondesse maggiore altezza, per poi ad ogni pixel associare il colore corrispondente nell'immagine a colori in una seconda passata. Con questo metodo la *trasformazione in 3D* del frattale dava risultati migliori, ma non aumentava di molto la chiarezza sull'effettiva forma del frattale proiettato.

Un secondo problema è stato quindi aumentare la chiarezza dell'immagine, si è pensato allora di aggiungere delle ombre. In un primo metodo per disegnare le ombre è stato implementato scorrendo più volte l'array alla ricerca delle pareti illuminate e in ombra per applicare le onde proprie e in un successivo momento una nuova scansione per cercare le ombre portate.
Appena iniziato a usare il programma per generare immagini sempre più grandi, ci si è però accorti che tale metodo era da abbandonare, primo per l'eccesivo numero di accessi ad ogni singolo punto, secondo perché mantenere in memoria tutta la struttura tridimensionale portava il programma ad un eccessivo e ingiustificato consumo di RAM (talvolta superiore a qualche centinaio di mega byte).

Tale metodo ha comunque permesso di notare che risultati buoni si potevano ottenere proiettando l'immagine con un'assonometria ortogonale e immaginando una sorgente di luce distante all'infinito con direzione diagonale dall'alto verso destra e entrante nel monitor.

La soluzione finale, trovata dopo varie riscritture del codice, consiste nel creare un array bidimensionale delle dimensioni dell'immagine dove salvare l'altezza di ogni pixel, calcolare la dimensione della nuova immagine ottenuta proiettando quelle precedenti. Per ogni pixel dell'immagine finale si scorre seguendo il percorso inverso che farebbe un raggio di luce nell'immagine tridimensionale, interrogando per ogni profondità le altezze memorizzate nell'array alla ricerca del primo punto *solido* che possa illuminare il pixel. Una volta trovato, se corrisponde ad un punto di *testa* si colora del colore corrispondente nell'immagine di partenza. Se è un punto di una parete verticale invece colora di un colore arbitrario che identifica le pareti. Nel caso poi in cui sia una parete si controlla se tale parete non è illuminata, in tal caso si scurisce. Nel caso la parete sia in luce, si ripercorre quindi la strada del raggio di luce fino alla sua fonte alla ricerca di eventuali ostacoli, che se trovati renderanno il pixel più scuro.

Come si può vedere nella figura 2.3 questo modulo prende come parametri il colore delle parati (in forma di tripletta RGB), il numero di scalini e l'altezza di ciascuno. Questo perché fin dall'inizio si era notato che facendo alzare ogni punto da 0 a 255 in base al loro grigio associato, l'immagine risultante era

decisamente poco leggibile, molto meglio limitare il numero di salti e renderli ben visibili aumentando il secondo parametro.

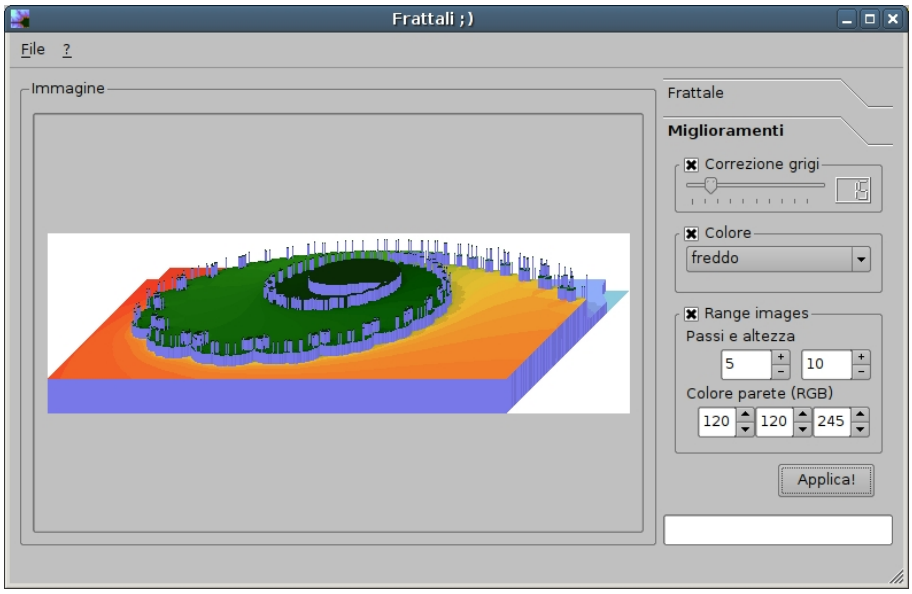


Figura 2.3: Screenshot del programma con generato una immagine in finto rilievo.

Capitolo 3

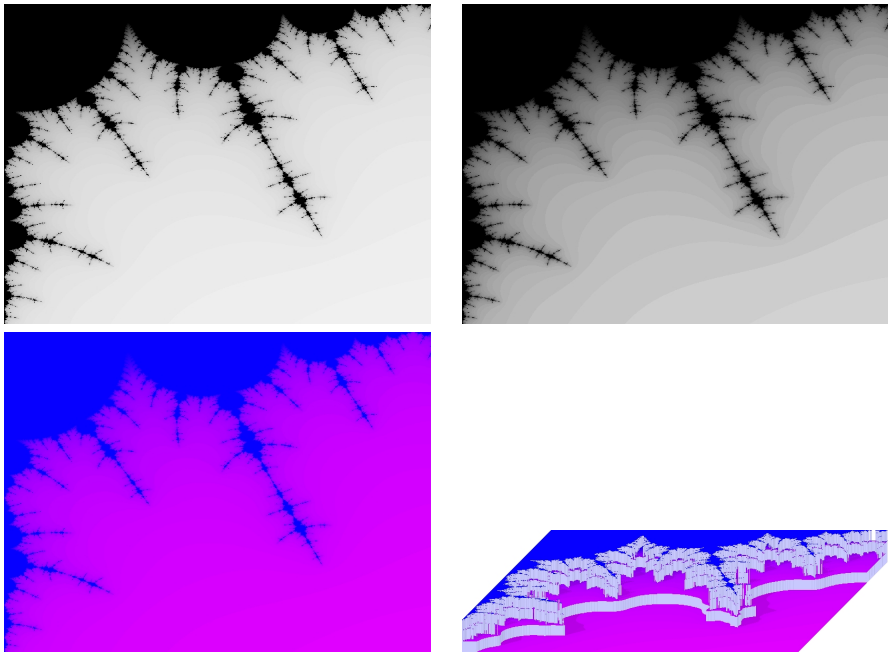
Alcuni esempi

Questo capitolo si propone di visualizzare alcune immagini create utilizzando questo programma, per mostrarne le potenzialità nonché alcuni aspetti teorici analizzati.

3.1 Fino al 3D

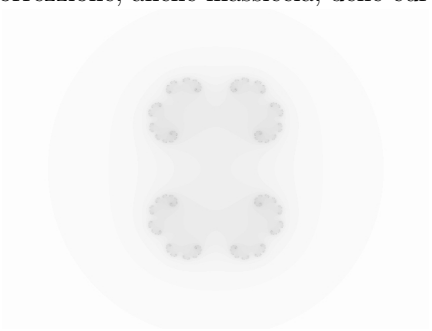
Si mostra nelle seguenti immagini la *vita* che può percorrere un'immagine nel programma (nello specifico un particolare dell'insieme di Mandelbrot):

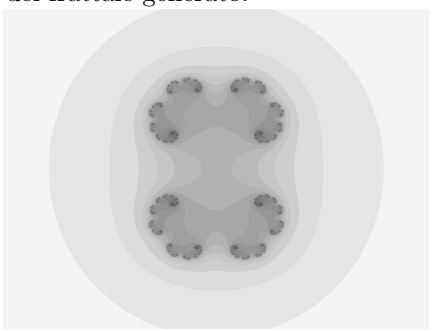
- in originale,
- con un leggero fattore di correzione di grigi,
- colorato,
- infine messo in rilievo.



3.2 La scala di grigi

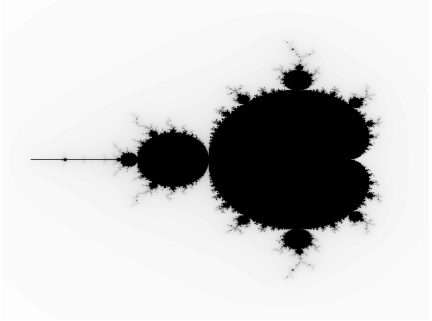
Qui di seguito viene riportato un esempio di come talvolta sia necessaria una correzione, anche massiccia, delle curve del frattale generato.

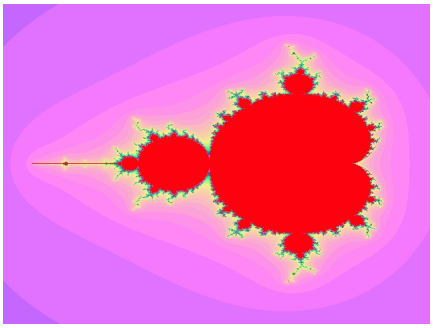


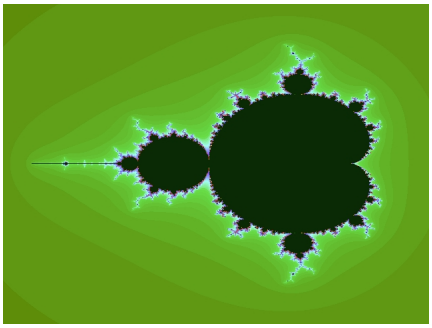


3.3 I colori

Di seguito mostrato l'insieme di Mandelbrot in originale e in due colorazioni distinte.







3.4 L'auto inclusione

Vengono mostrati ingrandimenti di due diversi frattali. Da queste immagini si può notare innanzitutto come l'insieme si ripropone simile lungo tutto il suo bordo, inoltre la stretta relazione tra gli insiemi di Julia e di Mandelbrot si può notare anche nel fatto che ingrandimenti di alcuni insiemi di Julia ripropongano forme simili all'insieme di Mandelbrot.

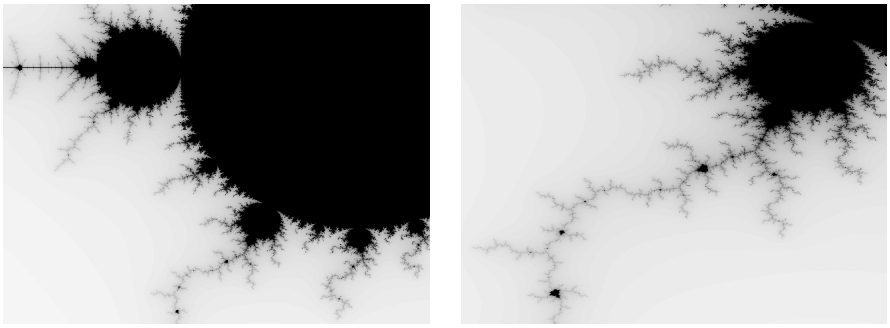


Figura 3.1: Insieme di Mandelbrot

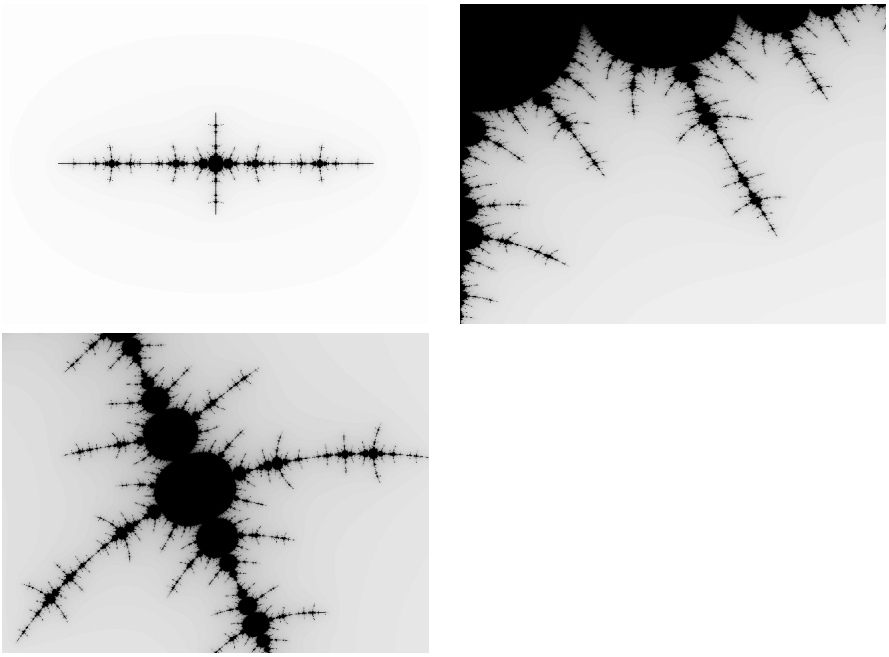


Figura 3.2: Insieme di Julia

Bibliografia

[1] http://it.wikipedia.org/wiki/Frattale

[2] http://it.wikipedia.org/wiki/Arte_frattale

[3] http://it.wikipedia.org/wiki/Lista_di_frattali_per_dimensione_di_Hausdorff

[4] http://it.wikipedia.org/wiki/Insieme_di_Julia

[5] http://it.wikipedia.org/wiki/Insieme_di_Mandelbrot

[6] http://www.miorelli.net/frattali/matematica.html