Tratto da www.wikipedia.org

http://en.wikipedia.org/wiki/Mandelbrot_set

L'*insieme di Mandelbrot* è definito come l'insieme dei numeri complessi tale per cui *non* è divergente la successione definita da:

$$z_{n+1} = {z_n}^2 + c$$

con

$$z_0 = 0$$

L'insieme è un *frattale* e, nonostante la semplicità della definizione, ha una forma non banale. Solo con l'avvento del computer è stato possibile visualizzarla.

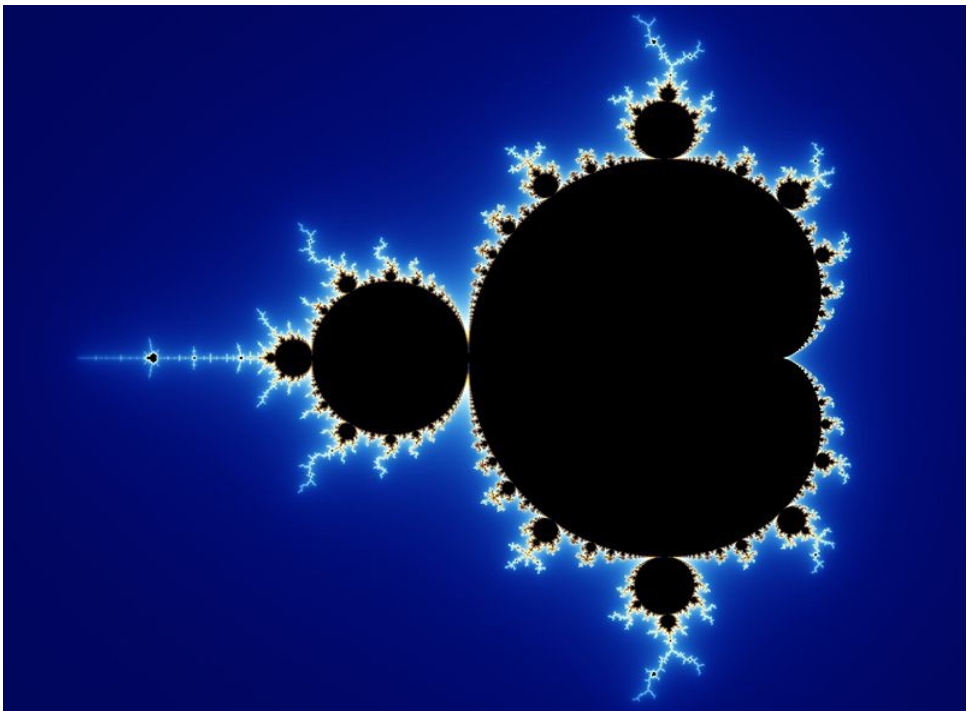

Fig. 1 - Una rappresentazione matematica dell'insieme di Mandelbrot (d'ora in poi insieme *M)*. I punti che appartengono all'insieme sono colorati di nero.

L'insieme deve il suo nome a *Benoît Mandelbrot* che nel 1975 nel suo libro *"Les Objects Fractals: Forme, Hazard et Dimension"* rese popolari i frattali. In questo libro Mandelbrot introdusse il termine *frattale* per descrivere alcuni comportamenti matematici che sembravano avere un comportamento "caotico". Questo genere di fenomeni nasce dalla definizione di curve od insiemi tramite funzioni o algoritmi ricorsivi.

Definizione formale

L'insieme di Mandelbrot M è definito a partire da una famiglia di *polinomi quadratici complessi*:

$$f_c : \mathbb{C} \to \mathbb{C}$$

nella forma:

$$f_c(z) = z^2 + c.$$

dove c è un parametro complesso. Per ogni c si considera il comportamento della successione

$$(0, f_c(0), f_c(f_c(0)), f_c(f_c(f_c(0))), \ldots)$$

ottenuta iterando $f_c(z)$ a partire dal punto z = 0; questa può o divergere all'infinito oppure essere limitata. L'insieme di Mandelbrot è definito come l'insieme dei punti c tali che la corrispondente sequenza è limitata.
Più formalmente, se $f_c^n(z)$ indica l'n-esima iterata di $f_c(z)$ (cioè $f_c(z)$ composta con sé stessa n volte), l'insieme di Mandelbrot è il sottoinsieme del piano complesso dato da:

$$M = \left\{ c \in \mathbb{C} : \sup_{n \in \mathbb{N}} |f_c^n(0)| < \infty \right\}$$

Dal punto di vista matematico, l'insieme di Mandelbrot è semplicemente un insieme di numeri complessi. Ogni numero complesso c può appartenere a M oppure no. Una rappresentazione grafica dell'insieme di Mandelbrot può essere ottenuta colorando tutti i punti c che appartengono M di nero, e gli altri di bianco. Le multicolori immagini che si vedono spesso sono generate colorando i punti esterni all'insieme in dipendenza di quanto velocemente la sequenza $|f_c^n(0)|$ diverge all'infinito.

Condizione di divergenza

Si può dimostrare che se il modulo di z_n è maggiore di 2 allora la successione divergerà e quindi il punto c sarà esterno all'insieme di Mandelbrot. Il minimo valore di n per cui $|z_n| > 2$ è un indice di quanto "lontano da bordo" si trova un punto e viene spesso utilizzato per la visualizzazione a colori dell'insieme.

Fine estratto wikipedia

Vogliamo realizzare un'applet che visualizzi l'insieme di Mandelbrot. Per mantenere il più possibile un approccio “object oriented”, procediamo nel seguente modo: creaimo dapprima la classe *MandelDraw* che estende *JPanel* (quindi si tratta di un pannello) e poi aggiungiamo il pannello alla classe *MandelBrotJApplet* (la nostra JApplet). Abbiamo bisogno inoltre delle classi *Complesso* (che rappresenta un numero del piano complesso) e della classe *Pixel*, una classe di supporto (che rappresenta un pixel del pannello).

Equazioni di trasformazione o di mappatura

Queste equazioni ci consentiranno di “mappare” un “punto” del piano complesso in un “pixel” del pannello, secondo lo schema seguente:

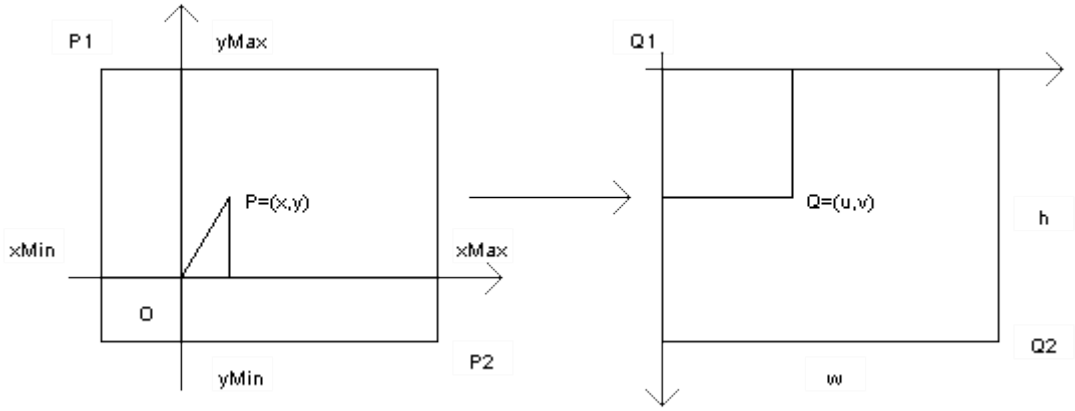

Fig. 2 – Il piano complesso e il piano del pannello

Supponiamo che le relazioni che intercorrono tra le coordinate (x,y) del punto P del piano complesso e il punto Q=(u,v) del piano del pannello siano di tipo lineare. Cioè ad esempio supponiamo che siano del tipo:

$$x = a*u + b$$
$$y = c*v + d$$

Ora imponiamo che il punto P1 sia "mappato" nel punto Q1 (*condizione1*) e che il punto P2 sia "mappato" nel punto Q2 (*condizione2*).
Dalla condizione1, otteniamo pertanto le equazioni

$$xMin = a*0 + b = b$$
$$yMax = c*0 + d = d$$

Dalla condizione2 e tenendo conto del risultato precedente, otteniamo facilmente con qualche passaggio algebrico:

$$a = (xMax - xMin)/w$$
$$c = (yMin - yMax)/h$$

Per cui le equazioni diventano:

$$x = ((xMax - xMin)/w)*u + xMin$$
$$y = ((yMin - yMax)/h)*v + yMax$$

Le classi

```
/**
 *
 *  Complesso.java
 */
package frattalijapplet;

/**
 *
 * @author maurizio
 */
public class Complesso {
    private double x;
    private double y;

    // costruttore "nullo"
    public Complesso() {
        x = 0;
        y = 0;
    }

    // costruttore con due argomenti
    public Complesso(double x, double y) {
        this.x = x;
        this.y = y;
    }

    // somma tra due numeri complessi
    public Complesso somma(Complesso p) {
        return new Complesso(this.x+p.getX(),this.y+p.getY());
    }

    // prodotto
    public Complesso prodotto(Complesso p) {
        return new Complesso(this.x*p.getX()-this.y*p.getY(),this.x*p.getY()
+this.y*p.getX());
    }

    // prodotto di uno scalare per un numero complesso
    public Complesso prodScal(double k) {
        return new Complesso(k*this.getX(),k*this.getY());
    }

    // getter e setter
    public double getX() {
        return x;
    }
```

```
    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }

    // restituisce la distanza del punto dall'origine
    public double modulo() {
        return Math.sqrt(x*x+y*y);
    }

} // fine classe Complesso

package frattalijapplet;

/**
 *
 *  Pixel.java
 */

/**
 *
 * @author maurizio
 */
public class Pixel {
    public int u;
    public int v;

    public Pixel() {
        u=0;
        v=0;
    }

    public Pixel(int u, int v) {
        this.u = u;
        this.v = v;
    }

} // fine classe Pixel
```

```
/*
 * MandelbrotDraw.java
 *
 * Created on 13 aprile 2008, 13.49
 *
 */

package frattalijapplet;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class MandelbrotDraw extends JPanel {
  double xMin=-0.61944667;
  double xMax=-0.42533338;
  double yMin=0.43555254;
  double yMax=0.63399994;

  //double xMin=-2;
  //double xMax=0.5;
  //double yMin=-1.55;
  //double yMax=1.55;

  static int maxIter=100;
  static int maxCol=255;

  @Override
  public void paintComponent(Graphics g) {
       super.paintComponent(g);
       Dimension d=getSize();
       int u;
       int v;
       int i;

       int red=0;
       int green=0;
       int blue=0;

       int k=0;
       int w = d.width;
       int h = d.height;

       // Faccio un loop sul piano complesso
       for(u=0; u<w; u++) {
          for(v=0; v<h; v++) {
             Pixel q = new Pixel(u,v);
             Complesso c = trasforma(q,w,h);
```

```
            // Poniamo z=0
             Complesso z = new Complesso();

             for(i=0; i<maxIter ; i++) {
                 // z = z*z+c
                 z=z.prodotto(z).somma(c);
                 // Se la distanza di z supera 2, si esce
                 if(z.modulo()>2) break;
             } // fine for

             if (z.modulo()>2) {
                // il punto non appartiene a M
                // per il calcolo del colore si veda
                //http://www.apropos-logic.com/nc/MandelbrotFractal.html
                 k=maxCol-(maxCol*i)/maxIter;
                 k=Math.min(k, 240);
                 red=k;
                 green=k;
                 blue=k;
             } else {
                 // al posto del solito colore nero scriviamo:
                 k=((int) (100*z.modulo()))/2+1;
                 red = (101*k)&maxCol;
                 green = (149*k)&maxCol;
                 blue = (199*k)&maxCol;
             }

             // Settiamo il colore
             g.setColor(new Color(red,green,blue));

             // disegnamo un punto
             g.drawLine(u,v,u,v);
         } // fine for v
      } // fine for u
   } // fine metodo PaintComponent

    public Complesso trasforma(Pixel q, int w, int h) {
        double x = ((xMax-xMin)/w)*q.u+xMin;
        double y = ((yMin-yMax)/h)*q.v+yMax;
        return new Complesso(x,y);
    }

} // fine classe
```

```
/**
 * MandelbrotJApplet.java
 */

package frattalijapplet;

import java.awt.Container;

/**
 *
 * @author maurizio
 */
public class MandelbrotJApplet extends javax.swing.JApplet {

    /** Creates a new instance of MandelbrotJApplet */
    public MandelbrotJApplet() {
    }

    public void init() {
      Container contentPane=getContentPane();
      contentPane.add(new MandelbrotDraw());
    }
} // fine classe MandelBrotJApplet
```

Segue codice della pagina web MandelbrotJApplet.html

```
<HTML>
<HEAD>
<TITLE>JApplet che genera un insieme di Mandelbrot</TITLE>
<style type="text/css">
<!--
body {
      font-family: Trebuchet MS, Geneva, Arial, helvetica, sans-serif;
      font-size:13px;
}
-->
</style>
</HEAD>
<BODY>
<H2>Insieme di Mandelbrot per l'intervallo [-0.62,-0.42] e [0.43,0.63]</h2>
<P>
<APPLET codebase="classes" code="frattalijapplet/MandelbrotJApplet.class"
width="600"
height="600"></APPLET>
</P>
<P>Generato con NetBeans e modificato con BlueFish Editor</P>
</BODY>
</HTML>
```

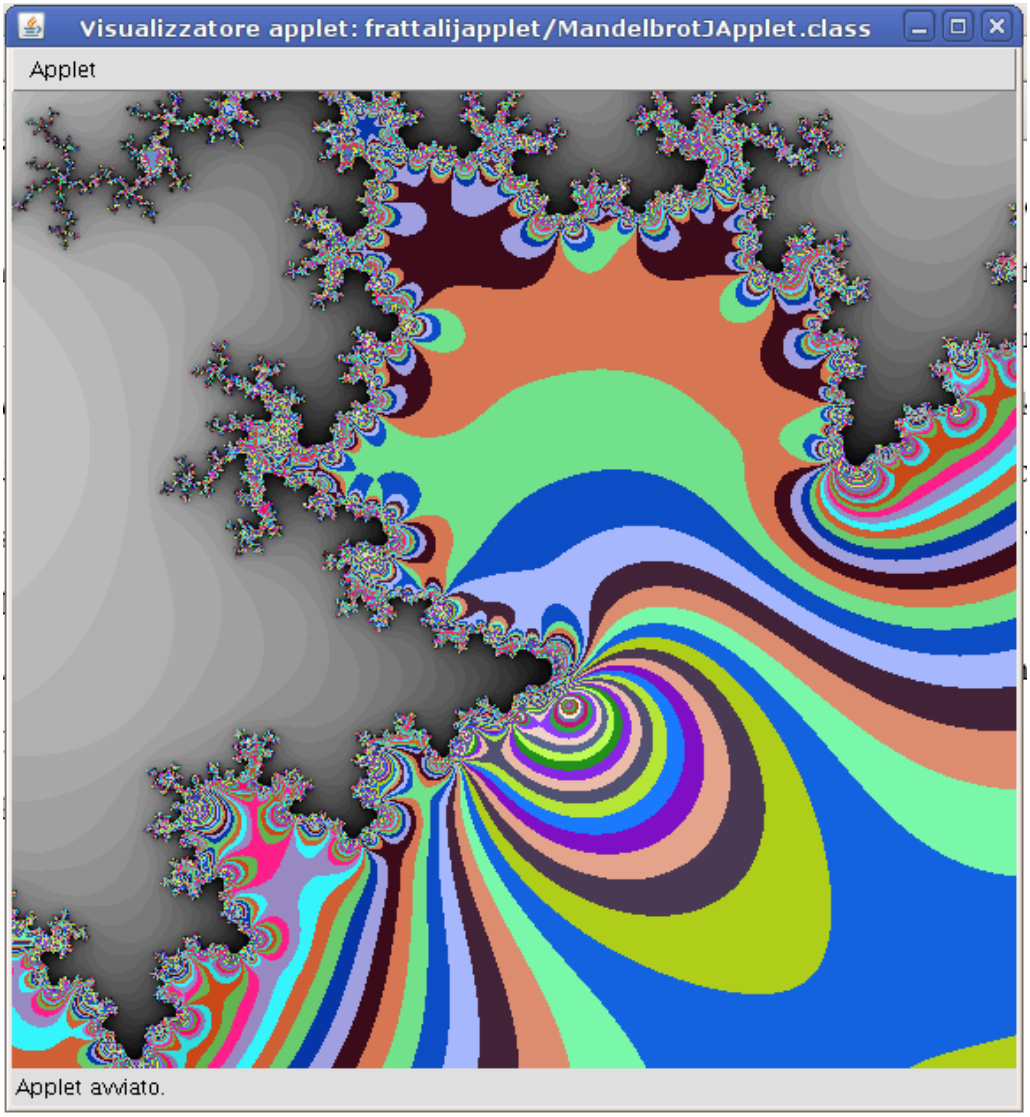

Fig. 4 – Insieme di Mandelbrot ottenuto per xMin=-0.6194667, xMax=-0.42533338, yMin=0.43555254, yMax=0.63399994

E' possibile ottenere degli "zoom" dell'insieme M semplicemente variando i parametri xMin, xMax, yMin e yMax (ad esempio richiedendoli in input all'utente – si lascia allo studente come esercizio). E' possibile, volendo, introdurre lo zoom facendo più semplicemente il drag and drop di alcune porzioni dell'insieme M. Ovviamente ciò complicherebbe notevolmente il codice.