ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ

Πτυχιακή Εργασία

ΥΠΟΛΟΓΙΣΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΣΕ ΜΟΡΦΟΚΛΑΣΜΑΤΙΚΑ ΣΥΝΟΛΑ

Κουτσάκης Κωνσταντίνος
Α.Ε.Μ.: 10654
Ιανουάριος 2006

Πτυχιακή Εργασία

ΥΠΟΛΟΓΙΣΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΣΕ ΜΟΡΦΟΚΛΑΣΜΑΤΙΚΑ ΣΥΝΟΛΑ (Mandelbrot και Julia)

Κουτσάκης Κωνσταντίνος
Α.Ε.Μ.: 10654
Επιβλέπων: Βουγιατζής Γεώργιος
Ιανουάριος 2006

Περίληψη

Σκοπός αυτής της εργασίας είναι να γίνει η ανάπτυξη και ανάλυση των αλγορίθμων που χρησιμοποιούνται για να σχεδιαστούν με τη βοήθεια ηλεκτρονικού υπολογιστή βασικά μορφοκλασματικά σύνολα και να γίνει υλοποίηση τους σε κάποια γλώσσα προγραμματισμού. Οι αλγόριθμοι για τη συγκεκριμένη εργασία εκμεταλλεύονται ιδιότητες των μορφοκλασματικών συνόλων όπως τη συνθήκη διαφυγής της τροχιάς ενός σημείου τους. Η υλοποίηση γίνεται σε μία αντικειμενοστρεφή γλώσσα προγραμματισμού και το πρόγραμμα που δημιουργείται περιέχει στοιχεία που μπορούν να αναδείξουν ιδιότητες των μορφοκλασματικών συνόλων, όπως η αυτο-ομοιότητα και η συσχέτιση των δύο βασικών μορφοκλασματικών συνόλων Mandelbrot και Julia.

Abstract

The purpose of this dissertation is to analyse algorithms that are used to draw basic fractal sets, using the computer and to bring a computer programme into effect by using a programming language. The algorithms in the present dissertation take advantage of properties met by fractals, such as the escape condition of orbits. An object oriented programming language is used to create the programme and the programme itself presents elements that feature certain properties of fractals, such as self-similarity and the relation between two basic fractal sets, Mandelbrot and Julia.

Πίνακας Περιεχομένων:

1.1 Fractals

Το fractal (το οποίο στα ελληνικά μετφράζεται ως μορφοκλασματικό σύνολο) είναι ένα γεωμετρικό αντικείμενο που εμφανίζει κάποια "μη κανονικότητα" σε κάθε κλίμακα μήκους στην οποία το μελετάμε. Τα fractal ως σχήματα θεωρητικά εμπεριέχουν άπειρη λεπτομέρεια και μπορεί να έχουν τη λεγόμενη αυτο-ομοιότητα (self-similarity) σε κάποιες δομές τους, η οποία εμφανίζεται σε διαφορετικά επίπεδα μεγέθυνσης. Σε πολλές περιπτώσεις ένα fractal μπορεί να προκύψει από έναν τύπο που δηλώνει επαναληπτική διαδικασία.

> Ο όρος fractal επινοήθηκε το 1975 από τον Benoit Mandelbrot, από τη λατινική λέξη fractus που σημαίνει σπασμένος. Προτού να εισάγει ο Mandelbrot τον όρο αυτόν η κοινή ονομασία για τέτοιες δομές (η χιονονιφάδα του Koch για παράδειγμα) ήταν "τερατώδης καμπύλη" (monster curve) [1].

Η πιο χαρακτηριστική ιδιότητα των fractal είναι ότι είναι γενικά περίπλοκα, δηλαδή εμφανίζουν ανωμαλίες στη μορφή τους και κατά συνέπεια δεν είναι αντικείμενα τα οποία μπορούν να οριστούν με τη βοήθεια της ευκλείδιας γεωμετρίας. Αυτό υποδεικνύεται από το ότι τα fractal έχουν λεπτομέρειες, η οποίες γίνονται ορατές κατά τη μεγέθυνση τους σε κάποια κλίμακα. Για να γίνει αντιληπτός αυτός ο διαχωρισμός των fractal σε σχέση με την ευκλείδια γεωμετρία αναφέρουμε ότι αν μεγενθύνουμε κάποιο αντικείμενο το οποίο μπορεί να οριστεί με την ευκλείδια γεωμετρία, παραδείγματος χάριν την περιφέρεια μιας έλλειψης, αυτή μετά από αλεπάλληλες μεγεθύνσεις απλά θα εμφανίζεται ως ευθύγραμμο τμήμα. Η συμβατική ιδέα της καμπυλότητας η οποία αντιπροσωπεύει το αντίστροφο της ακτίνας ενός προσεγγίζοντος κύκλου, δεν μπορεί ωφέλιμα να ισχύσει στα fractal επειδή αυτή εξαφανίζεται κατά τη μεγέθυνση. Ένα fractal αντιθέτως θα εμφανίζει κατόπιν μεγεθύνσεων λεπτομέρειες που δεν ήταν ορατές σε μικρή κλίμακα μεγέθυνσης.

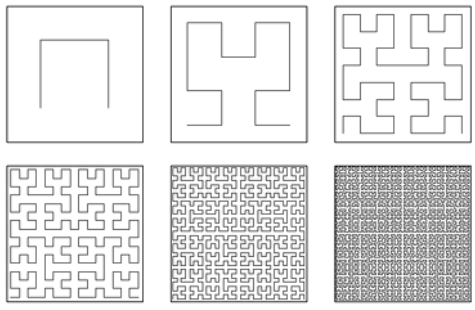
Τα χαρακτηριστικά σύμφωνα με τα οποία θα μπορούσε να οριστεί κάθε είδος fractal αποδείχτηκε πολύ δύσκολα να συμπτυχθούν σε έναν ακριβή μαθηματικό ορισμό. Ο Mandelbrot όρισε ως fractal "το σύνολο για το οποίο η διάσταση Hausdorff - Besicovitch υπερβαίνει αυστηρά την τοπολογική του διάσταση" [2]. Για ένα fractal που παρουσιάζει πλήρη αυτο-ομοιότητα η διάσταση Hausdorff που το χαρακτηρίζει είναι ίση με την αντίστοιχη διάσταση Minkowski-Bouligand.

Τα προβλήματα που σχετίζονται με τον ορισμό των fractal περιλαμβάνουν μεταξύ άλλων:

- Δεν υπάρχει ακριβής ορισμός για το "μη-κανονικό".
- Δεν υπάρχει ένας μόνο ορισμός για τη διάσταση.
- Η έννοια της αυτο-ομοιότητας δεν είναι ακριβής.
- Δεν ορίζονται όλα τα fractal με επαναληπτικές διαδικασίες.

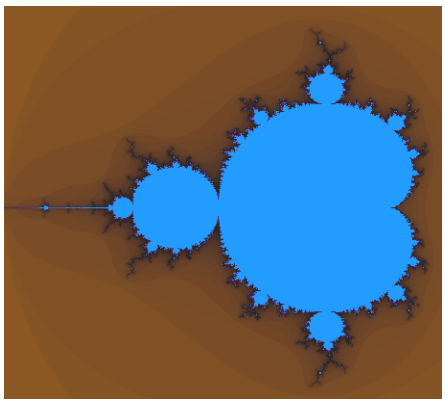
Για αυτό το λόγο γίνεται διαχωρισμός των fractal σε τρεις γενικές κατηγορίες ως προς τον τρόπο κατασκευής τους και σε τρεις κατηγορίες ως προς το τι είδους αυτο-ομοιότητα εμφανίζουν. Ως προς την κατασκευή τους τα fractal διαχωρίζονται σε:

- Αυτά που προκύπτουν από συστήματα συναρτήσεων που ορίζουν μία επαναληπτική διαδικασία (Iterated Function Systems - IFS). Κατά τη διαδικασία αυτή δημιουργούνται διάφορα αντίγραφα του ίδιου αντικειμένου σε κάθε ένα από τα οποία επιδρά μία συνάρτηση. Παραδείγματα αυτής της



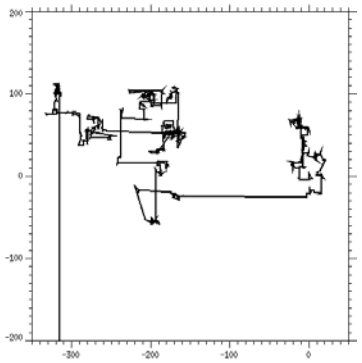
Εικόνα 1: Έξι διαδοχικές επαναλήψεις που δημιουργούν από αρχικά τρία ευθύγραμμα τμήματα την καμπύλη του Hilbert.

κατηγορίας είναι ο Τάπητας και το Τρίγωνο του Sieprinski, η καμπύλη του Hilbert, η χιονονιφάδα του Koch και άλλα.



Εικόνα 2: Το σύνολο Mandelbrot.

- Fractal που προκύπτουν από μία αναδρομική σχέση που εφαρμόζεται σε κάθε σημείο ενός διαστήματος (όπως για παράδειγμα το μιγαδικό επίπεδο). Το σύνολο του Mandelbrot και το fractal του Lyapunov αποτελούν παραδείγματα αυτού του είδους. Αυτά τα fractal λέγονται επίσης χρονικώς διαφεύγοντα (time-escape fractals).
- Τυχαία fractal που παράγονται από στοχαστικές και όχι αιτιοκρατικές διαδικασίες όπως οι πτήσεις Lévy.



Εικόνα 3: Παράδειγμα της πτήσης Lévy.

Μία άλλη κατηγοριοποίηση των fractal γίνεται σύμφωνα με την αυτο-ομοιότητα που εμφανίζουν αυτά:

- Ακριβής αυτό-ομοιότητα. Είναι ο ισχυρότερος τύπος αυτό-ομοιότητας. Όταν υπάρχει, ένα fractal εμφανίζεται το ίδιο σε διαφορετικές κλίμακες. Τα fractal που παρουσιάζουν αυτό-ομοιότητα από συναρτήσεις επαναληπτικής διαδικασίας εμφανίζουν συνήθως ακριβή αυτό-ομοιότητα.

- Ημι-αυτο-ομοιότητα. Είναι μια πιο χαλαρή μορφή αυτό-ομοιότητας. Όταν υπάρχει αυτή, ένα fractal εμφανίζεται περίπου (αλλά όχι ακριβώς) ίδιο σε διάφορες κλίμακες. Στα ημί-αυτό-όμοια fractal γίνεται φανερό μετά από μεγενθύνσεις ότι περιέχουν σε μικρά αντίγραφα την απεικόνιση που περιλαμβάνει ολόκληρο το fractal, σε διαστρεβλωμένες και εκφυλισμένες μορφές. Fractal που ορίζονται από εφαρμογή αναδρομικών σχέσεων σε διάστημα εμφανίζουν συνήθως ημί-αυτό-ομοιότητα.
- Στατιστική αυτό-ομοιότητα. Είναι ο πιό αδύνατος τύπος αυτό-ομοιότητας. Όταν την εμφανίζει ένα fractal, διατηρούνται αριθμητικά ή στατιστικά μεγέθη ανά τις κλίμακες μεγέθυνσης. Οι περισσότεροι λογικοί ορισμοί για τον όρο "fractal" υπονοούν σιωπηρά κάποια μορφή στατιστικής αυτό-ομοιότητας - η ίδια η διάσταση fractal είναι ένα αριθμητικό μέτρο που διατηρείται σε κάθε κλίμακα. Τα τυχαία fractal είναι παραδείγματα fractal που δεν είναι στατιστικά αυτό-όμοια, αλλά ούτε ακριβώς ούτε ημί-αυτό-όμοια.

Επειδή τα fractal εμφανίζουν τεράστια λεπτομέρεια σε όποια κλίμακα και αν τα μεγενθύνουμε, είναι λογικό το συμπέρασμα ότι κάνένα φυσικό αντικείμενο δεν είναι fractal. Εντούτοις, πολλά φυσικά αντικείμενα εμφανίζουν ιδιότητες παρόμοιες με τα διάφορα είδη fractal για κάποιο περιορισμένο εύρος της κλίμακας αυτο-ομοιότητας.

1.2 Επαναληπτικές Διαδικασίες και Διακλαδώσεις

1.2.1 Αριθμητικές Επαναληπτικές Διαδικασίες

Μία από τις πιο σημαντικές ανακαλύψεις στα μαθηματικά του προηγούμενου αιώνα που εξακολουθεί να απασχολεί πολλούς μαθηματικούς ακόμη και σήμερα είναι η αντίληψη ότι ακόμη και τα πιο απλά δυναμικά συστήματα μπορεί να εμφανίζουν εξαιρετική δυσκολία στην πρόβλεψη της εξέλιξης τους στο χρόνο. Ας πάρουμε για παράδειγμα τις εξισώσεις:

$$y = x^2 + c \tag{1.2.1}$$

και $$x = y \tag{1.2.2}$$

με $x, y, c \in R$. Οι εξισώσεις αυτές αποτελούν τις εκφράσεις μιας παραβολής και μίας ευθείας αντίστοιχα. Πολλά από τα δημοφιλή μαθηματικά του παρόντος μπορούν να περιγραφούν με τη βοήθεια των δύο παραπάνω, ή αντίστοιχων εξισώσεων.

Ένας τρόπος για να ερμηνευθούν οι εξισώσεις που μας απασχολούν είναι ως δύο καμπύλες στο επίπεδο. Ένας άλλος όμως με περισσότερο ενδιαφέρον είναι να τις δει κάποιος σαν ένα σύνολο οδηγιών (όπως σε έναν αλγόριθμο):

1. Δοθέντος πραγματικού αριθμού x υπολογίστε το τετράγωνο του και προσθέστε τη σταθερά c. Έστω το αποτέλεσμα y.
2. Με το y αμετάβλητο, ονομάστε το αποτέλεσμα x.
3. Επαναλάβετε το βήμα 1 με την τιμή που βρέθηκε στο βήμα 2.

Οι πρώτες δύο οδηγίες μαζί, δίνουν μία απεικόνιση του ενός αριθμού στον άλλον:

$$f : x \to x^2 + c \tag{1.2.3}$$

Αν πραγματοποιήσουμε το παραπάνω για όλους τους πραγματικούς θα έχουμε μια απεικόνιση των πραγματικών στους πραγματικούς που θα εκφράζεται ως:

$f : R \to R$. Η προσθήκη του τρίτου βήματος μας δίνει μία *απεικόνιση επαναληπτικής διαδικασίας* (*iterated mapping*). Θα χρησιμοποιήσουμε το συμβολισμό $f^n(x)$ για να εκφράσουμε τη n-οστή επανάληψη (αυτή δηλαδή που προκύπτει μετά από n εφαρμογές του αλγόριθμου της επανάληψης στην αρχική τιμή μας, τη x). Έτσι έχουμε τη σειρά αριθμών:

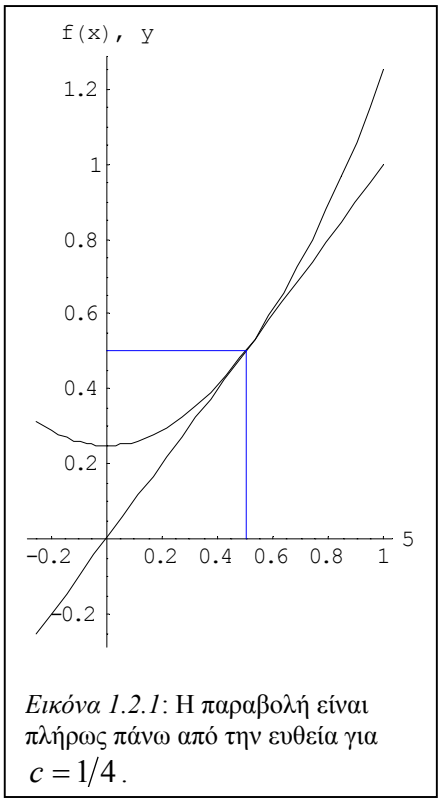
$x, f(x), f^2(x), f^3(x),..., f^n(x),...$ η οποία καλείται *τροχιά* (*orbit*). Η *αρχική συνθήκη* για το x έχει καθιερωθεί κυρίως από εφαρμογές στην επιστήμη των υπολογιστών να λέγεται *σπόρος* (*seed*) της τροχιάς. Παρατηρούμε ότι δεν υπάρχει συνθήκη στον αλγόριθμο που να υποδεικνύει πότε σταματάει η εκτέλεση του. Αν όμως ένας άνθρωπος εκτελέσει τον αλγόριθμο και όχι ένας υπολογιστής, θα εφεύρει ενδεχομένως κάποτε μία συνθήκη που θα τερματίζει τον αλγόριθμο. Η συνθήκη αυτή θα προκύψει από ένα δείγμα τιμών που θα λάβει. Για να κατανοηθεί καλύτερα αυτό ας εξετάσουμε τις διάφορες περιπτώσεις:

Για $c = 0$ από την εξίσωση (1.2.3) $f : x \to x^2 + c$, εύκολα συμπεραίνουμε το αποτέλεσμα: Για $|x| > 1$, θα είναι $f^n \to \infty$. Για $x = 1$, $f^n \to 1$ και για $|x| < 1$, θα είναι $f^n \to 0$, που συνοψίζονται παρακάτω:

$$f^n \to \begin{cases} \infty, & |x| > 1 \\ 1, & |x| = 1 \\ 0, & |x| < 1 \end{cases} \qquad \text{για } n \to \infty .$$

Όλες οι τροχιές φτάνουν στο μηδέν ή στο άπειρο εκτός από αυτές με σπόρο $x = \pm 1$. Τα σημεία μηδέν και άπειρο, λέγονται *ελκυστές* (*attractors*) ή *λεκάνες έλξης* (*sinks*, *basins of attraction*) επειδή έλκουν τις τροχιές των σημείων γύρω τους, ενώ τα σημεία $x = \pm 1$, λέγονται *απωθητές* (*repellers*) για τον αντίθετο λόγο.

Όταν το $c = 1/4$ η παραβολή που δίνεται από την εξίσωση (1.2.1) $y = x^2 + c$, είναι εφαπτόμενη και πάνω από την ευθεία (1.2.2) $x = y$, όπως φαίνεται στο διάγραμμα στην Εικόνα 1.2.1. Στην προκειμένη περίπτωση η παραβολή έχει ένα σημείο τομής με την ευθεία το οποίο είναι το $P(1/2, 1/2)$. Στην επαναληπτική διαδικασία τώρα, οι αρχικές συνθήκες με απόλυτη τιμή μεγαλύτερη από το $1/2$ θα φτάσουν στο άπειρο, ενώ αυτές στο διάστημα $0 \le |x| \le 1/2$, πλησιάζουν ασυμπτωματικά το $1/2$. Μετά από 700 επαναλήψεις οι τροχιές τους φτάνουν στο 0.499. Τα αποτελέσματα των πρώτων επαναλήψεων φαίνονται στον Πίνακα 1.2.1 παρακάτω:



Εικόνα 1.2.1: Η παραβολή είναι πλήρως πάνω από την ευθεία για $c = 1/4$.

n	-1	-0,75	-0,5	-0,25	-0,1	0
1	1,25	0,8125	1/2	1/3	1/4	1/4
2	1,8125	0,910156	0,5	0,347656	0,3176	0,3125
3	3,53515625	1,078384	0,5	0,370865	0,35087	0,347656
4	12,74732971	1,412913	0,5	0,387541	0,37311	0,370865
5	162,7444148	2,246323	0,5	0,400188	0,389211	0,387541
6	26485,99454	5,295967	0,5	0,41015	0,401485	0,400188
7	701507907,2	28,29726	0,5	0,418223	0,41119	0,41015
8	$4{,}92113 \cdot 10^{17}$	800,985	0,5	0,424911	0,419077	0,418223
9	$2{,}42176 \cdot 10^{35}$	641577,3	0,5	0,430549	0,425626	0,424911
10	$5{,}8649 \cdot 10^{70}$	$4{,}12 \cdot 10^{11}$	0,5	0,435373	0,431157	0,430549
11	$3{,}4397 \cdot 10^{141}$	$1{,}69 \cdot 10^{23}$	0,5	0,439549	0,435897	0,435373
12	$1{,}1832 \cdot 10^{283}$	$2{,}87 \cdot 10^{46}$	0,5	0,443204	0,440006	0,439549
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
∞	∞	$+1/2$	$+1/2$	$+1/2$	$+1/2$	$+1/2$

Πίνακας 1.2.1: Οι τροχιές που δεν οδηγούνται στο άπειρο έλκονται στο $+1/2$.

Το σημείο έλξης αλλάζει καθώς αλλάζει και η σταθερά. Αν τώρα η παραβολή έχει δύο σημεία τομής με την ευθεία αυτά θα βρίσκονται από τις τιμές των ριζών της εξίσωσης:

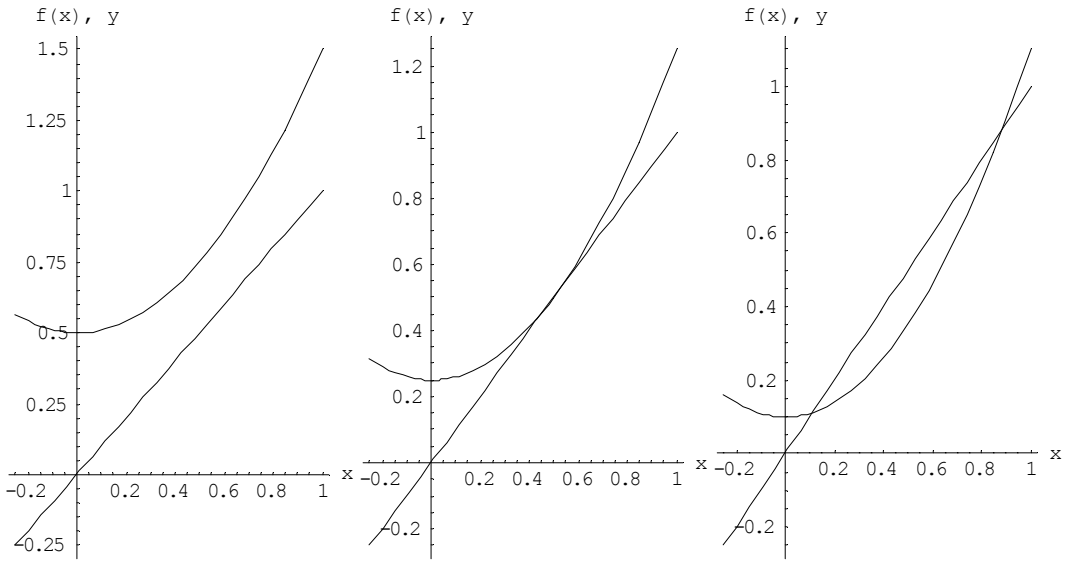
$$x^2 + c = x \tag{1.2.4}$$

Με περαιτέρω ανάλυση μπορούμε να αποδείξουμε ότι η μικρότερη από τις δύο ρίζες είναι ελκυστής και η μεγαλύτερη απωθητής. Το έχουμε ήδη δείξει για την ειδική περίπτωση $c = 0$. Ας δούμε το ίδιο για $c = -3/4$. Εδώ η εξίσωση (1.2.4) έχει ρίζες τα $x = -1/2$ και $x = +3/2$. Τα αποτελέσματα των πρώτων επναλήψεων φαίνονται στον Πίνακα 1.2.2 παρακάτω:

n	1,75	1,5	1	0,75	0,5	0,25
1	2,3125	1,5	0,25	-0,1875	-0,5	-0,6875
2	4,59765625	1,5	-0,6875	-0,71484	-0,5	-0,27734
3	20,38844299	1,5	-0,27734	-0,239	-0,5	-0,67308
4	414,9386077	1,5	-0,67308	-0,69288	-0,5	-0,29696
5	172173,2981	1,5	-0,29696	-0,26992	-0,5	-0,66181
6	29643644594	1,5	-0,66181	-0,67714	-0,5	-0,312
7	$8{,}78746\cdot10^{20}$	1,5	-0,312	-0,29148	-0,5	-0,65265
8	$7{,}72194\cdot10^{41}$	1,5	-0,65265	-0,66504	-0,5	-0,32404
9	$5{,}96283\cdot10^{83}$	1,5	-0,32404	-0,30772	-0,5	-0,645
10	$3{,}5555\cdot10^{167}$	1,5	-0,645	-0,65531	-0,5	-0,33398
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
∞	∞	$+3/2$	$-1/2$	$-1/2$	$-1/2$	$-1/2$

Πίνακας 1.2.2: Όπως ειπώθηκε, η μεγαλύτερη από τις δύο ρίζες ($x = 3/2$) είναι σημείο απώθησης και το $x = -1/2$ ελκυστής.

Παρατηρούμε επίσης πως η τροχιά πλησιάζει το $-1/2$ κάνοντας ταλάντωση μεταξύ θετικών και αρνητικών προσεγγίσεων όπου κάθε μία εξ αυτών προσεγγίζει το $-1/2$ ασυμπτωτικά. Αυτή η συμπεριφορά φαίνεται καθαρά στον Πίνακα 1.2.2 και αποτελεί ένδειξη χαοτικής συμπεριφοράς του συστήματος όπως θα φανεί παρακάτω.



Εικόνα 1.2.2: Από αριστερά προς τα δεξία, $c > 1/4$, $c = 1/4$, $-3/4 < c < 1/4$. Σε αυτές τις τιμές της σταθεράς c, έχουμε κανένα, ένα και δύο σημεία ισορροπίας αντίστοιχα.

Για τιμές της σταθεράς $c < -3/4$, οι τροχιές που προηγουμένως θα πλησίαζαν τη μικρότερη από τις δύο ρίζες, τώρα ταλαντώνονται μεταξύ δύο διακριτών τιμών. Το σημείο που ήταν προηγουμένως ελκυστής έχει τώρα *διακλαδωθεί* (*bifurcated* - *bifurcate*) και η τροχιά δεν είναι πλέον σταθερή αλλά περιοδική και εναλλάσσεται μεταξύ δύο τιμών. Καθώς το c μειώνεται ακόμη περισσότερο, άλλη μία *διακλάδωση* (*bifurcation*) γίνεται και η περίοδος διπλασιάζεται στο 4 και μετά στο 8 και ούτω καθεξής επ'άπειρον. Εντούτοις, η απόσταση μεταξύ δύο διαδοχικών διακλαδώσεων πλησιάζει το μηδέν με τέτοιον τρόπο, ώστε ο διπλασιασμός της περιόδου φτάνει στο άπειρο για πεπερασμένη τιμή της σταθεράς c, λίγο πιο κάτω από το -1.4. Πέρα από αυτήν την τιμή, οι τροχιές που προηγουμένως ήταν περιοδικές τώρα κινούνται στο διάστημα $[-2,2]$, χωρίς αυτή η κίνηση να εμφανίζει κάποια κανονικότητα και επισκέπτονται κάθε τιμή του παραπάνω διαστήματος. Αυτή η συμπεριφορά ονομάζεται *χαοτική συμπεριφορά*. Επιπλέον, αρχικές συνθήκες που ήταν στις πρώτες επαναλήψεις κοντά μεταξύ τους, μετά από μερικές επιπλέον επαναλήψεις θα

ακολουθήσουν πολύ διαφορετικές τροχιές. Αυτή η συμπεριφορά που υπερβαίνει την ευαίσθητη *εξάρτηση από τις αρχικές συνθήκες* ονομάζεται χαοτική επίσης και οι τιμές της παραμέτρου c για τις οποίες συμβαίνει αυτό αποτελούν το *χαοτικό καθεστώς* (*chaotic regime*). Η σειρά των διακλαδώσεων που οδηγούν στο χαοτικό καθεστώς είναι γνωστη ως η *διαδρομή διπλασιασμού περιόδου που οδηγεί στο χάος* (*period-doubling route to chaos*) [3].

1.2.2 Επαναληπτικές Διαδικασίες Συναρτήσεων

Πέρα από τις επαναληπτικές διαδικασίες που γίνονται με αριθμούς και παρουσιάζουν έλξη σε ένα σημείο ή άπωση από ένα σημείο, έχουμε και συναρτήσεις που εμφανίζουν τέτοια συμπεριφορά. Ένα παράδειγμα αυτών δίνεται από συναρτήσεις που παρουσιάζουν *Μονοδιάστατα Διακριτά Δυναμικά Συστήματα* (*One Dimensional Discrete Dynamical Systems*). Σε αυτά έχουμε Εξισώσεις Διαφορών πρώτης τάξης και η διαδικασία δίνεται από τα παρακάτω:

$$x_{n+1} = f(x_n) \tag{1.2.5}$$

με $x_i \in R$ και $n = 0,1,2,...$ $(n \in N)$.

Συνήθως η συνάρτηση f ορίζεται στο χώρο C^0 ή στο C^∞. Ως αρχικές συνθήκες λαμβάνουμε τη συνθήκη: $x = x_0$ και η τροχιά είναι το σύνολο των σημείων μέχρι την τιμή του *n* που υπολογίζουμε, δηλαδή: $T = \{x_0, x_1, x_2, ...\}$.

Κάθε αναλυτική έκφραση της μορφής $x_n = \varphi(n)$ που ικανοποιεί τη διαφορική εξίσωση του δυναμικού συστήματος, δηλαδή κάθε διαφορική εξίσωση που παράγει μία τροχιά *T* για το σύστημα, αποτελεί τη μερική αναλυτική λύση του συστήματος.

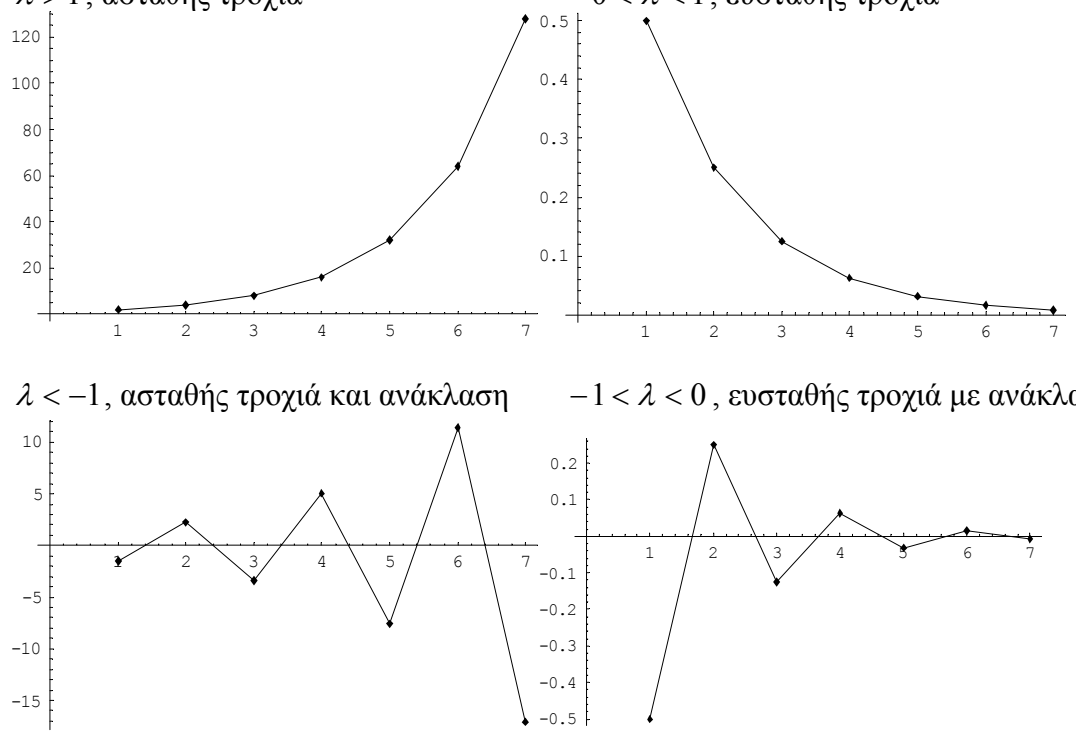
Επιπλέον, κάθε αναλυτική έκφραση της μορφής: $x_n = \varphi(n, c),$ με $c = c(x_0)$, που ικανοποιεί τη διαφορική εξίσωση και παράγει όλες (ή σχεδόν όλες) τις τροχιές του συστήματος, αποτελεί τη γενική λύση του συστήματος.

Σημειώνεται ότι άλλοι συμβολισμοί είναι: $x_n = x(n)$ και $x_{n+1} = x(n+1)$. Όταν αναφερόμαστε σε μεταβολή των δυναμικών συστημάτων με το χρόνο, η μεταβλητή *n* παίζει το ρόλο χρόνου που λαμβάνει διακριτές τιμές. Αν ισχύει: $f = f(x_n, n)$ η εξίσωση είναι μη αυτόνομη. Για της μη γραμμικές διαφορικές εξισώσεις, γενικά, δεν υπάρχει λύση. Τέλος, αν υπάρχει η αντίστροφη της f δηλαδή αν: $x_n = f^{-1}(x_{n+1})$ και η f είναι μονότιμη, το σύστημα λέγεται αντιστρέψιμο και η τροχιές μπορούν να οριστούν και για αρνητικές τιμές του *n* δηλαδή για παρελθοντικές χρονικές τιμές.

Η γενική μορφή της γραμμικής διαφορικής εξίσωσης, είναι:

$$x_{n+1} = \lambda x_n \tag{1.2.6}$$

με $\lambda \in R$. Η γενική της λύση θα είναι: $x_n = x_0 \lambda^n = x_0 e^{n \log|\lambda|}$. Τον τύπο της τροχιάς τον διακρίνουμε ανάλογα με το διάστημα στο οποίο βρίσκεται η τιμή του λ. Έτσι λαμβάνουμε τις διαφορετικές τροχιές για διάφορες τιμές του λ, όπως φαίνεται στα γραφήματα στην επόμενη σελίδα:



Εικόνα 1.2.3: Χαρακτηρισμός τροχιών ανάλογα με τη τιμή του λ. Από πάνω αριστερά με φορά των δεικτών του ρολογιού οι ακριβείς τιμές του λ: $\lambda = 2, \lambda = 0.5, \lambda = -1.5, \lambda = -0.5$ και σε όλες $x_0 = 1$.

Για $\lambda = 0$, έχουμε ειδική περίπτωση όπου $x_n = x_0$ για κάθε *n*.

Όταν ικανοποιείται η συνθήκη: $x_{n+1} = x_n$, έχουμε σταθερό σημείο που συμβολίζεται

με x^* ή ισοδύναμα όταν:

$$f(x^*) - x^* = 0 \qquad (1.2.7)$$

Κοντά στο x^* μπορούμε να γραμμικοποιήσουμε το σύστημα ουτοπώς:

Για: $x_n = x^* + \delta x_n$ και $x_{n+1} = x^* + \delta x_{n+1}$:

$$x_{n+1} = f(x_n) \Rightarrow x^* + \delta x_{n+1} = f(x^* + \delta x_n) \Rightarrow (A.T.) \Rightarrow$$
$$\Rightarrow x^* + \delta x_{n+1} = f(x^*) + \left.\frac{df(x)}{dx_n}\right|_{x_n = x^*} \delta x_n + O(\delta x_n{}^2)$$

Έτσι προκύπτει ότι: $\delta x_{n+1} = \lambda \delta x_n$, με $\lambda = \left.\frac{df(x)}{dx}\right|_{x=x^*}$ που δίνει το δυναμικό σύστημα

στο $x = x^*$.

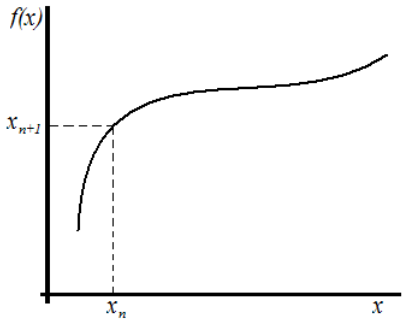
Μπορούμε επίσης να γνωρίζουμε την ευστάθεια ή αστάθεια των σταθερών σημείων. Αν έχουμε $|\lambda| < 1$, έχουμε γραμμική ευστάθεια. Για $|\lambda| > 1$, προκύπτει γραμμική αστάθεια και για $|\lambda| = 1$, έχουμε παραβολικό σταθερό σημείο. Για $\lambda < 0$, προκύπτει αστάθεια ή ευστάθεια με ανάκλαση.

Πολλές φορές μπορεί να προκύψουν περιοδικά σημεία στην τροχιά που προκύπτει από την αναδρομή που εφαρμόζουμε στη συνάρτηση. Αν συμβολίσουμε: $x_1 = f(x_0)$, $x_2 = f(x_1) = f(f(x_0)) = f^2(x_0)$ και ούτω καθεξής ώστε: $x_k = f(x_{k-1}) = f(f(f...f(x_0) \underbrace{)...)}_{\substack{k-1 \\ \text{παρενθεσεις}}} = f^k(x_0)$. Οπότε το x_k, θα προκύπτει κάθε φορά από τη σύνθεση της f με τον εαυτό της *k* φορές: $x_k = \underbrace{(f \circ f \circ ... \circ f)}_{\substack{k \\ \text{φορές}}}(x_0)$.

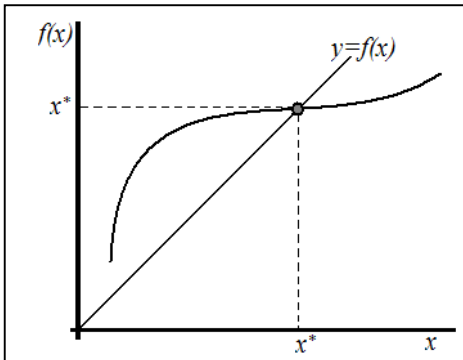
Ένα σημείο $x^{(p)}$ λέγεται περιοδικό με περίοδο *p*, αν: $x_{n+p} = x_n$, για κάθε *n* ή ισοδύναμα, όταν ένα σταθερό σημείο της απεικόνισης: $x_{n+1} = f^p(x_n)$ ή σύμφωνα με την εξίσωση 1.2.7, όταν: $f^p(x^{(p)}) - x^{(p)} = 0$.

Η περιοδική τροχιά που αντιστοιχεί σε περιοδικό σημείο περιόδου *p*, αποτελείται από *p* στοιχεία: $T_p = \{x_0, x_1, x_2, ..., x_{p-1}\}$. Εύκολα συμπεραίνουμε ότι κάθε $x \in T_p$ είναι ένα περιοδικό σημείο περιόδου *p*. Η ευστάθεια των περιοδικών σημείων που ανήκουν στην T_p είναι η ίδια με το σταθερό σημείο της απεικόνισης f^p. Όλα τα σημεία μιας περιοδικής τροχιάς έχουν την ίδια ευστάθεια.

Με τα παραπάνω μπορεί να γίνει γραφική αναπαράσταση των τροχιών. Όλα τα σημεία $P = (x_n, x_{n+1})$ ανήκουν στην καμπύλη $y = f(x)$, όπως φαίνεται και στη διπλανή εικόνα. Πέρα από αυτό, εύκολα κάποιος συμπεραίνει από τον ορισμό των σταθερών σημείων, ότι αυτά θα αναπαρίστανται από τα σημεία τομής των καμπυλών $y = f(x)$ και $y = x$, όπως απεικονίζεται στην εικόνα 1.2.5 παρακάτω.

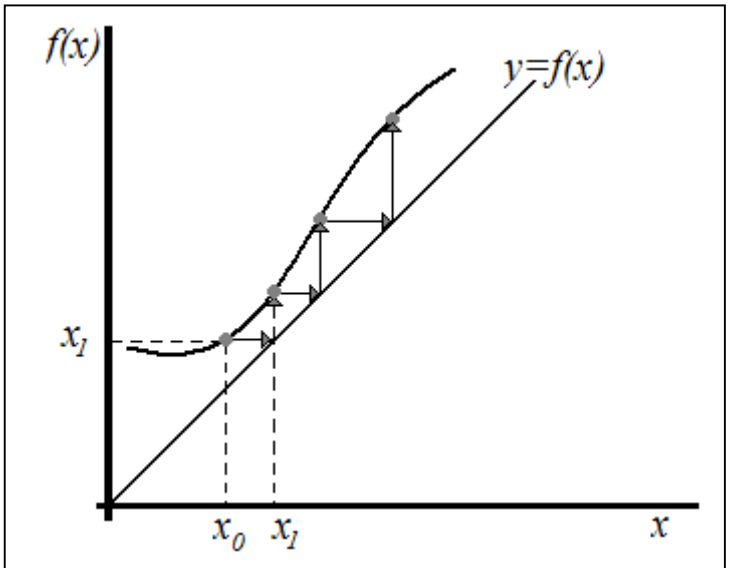


Εικόνα 1.2.4: Όλα σημεία $P = (x_n, x_{n+1})$, ανήκουν στην καμπύλη $y = f(x)$.



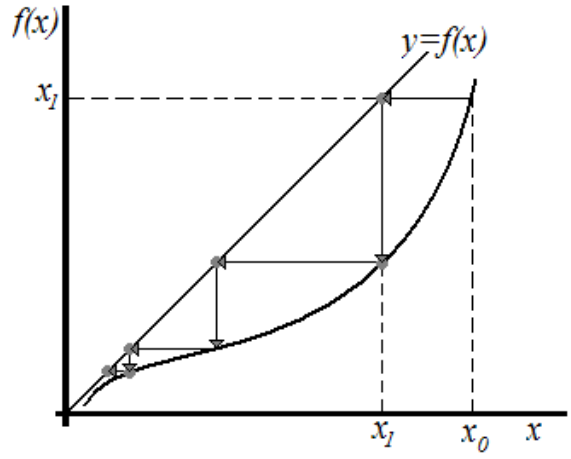
Εικόνα 1.2.5: Τα σημεία τομής της $y = f(x)$ με την $y = x$, δίνουν τα σταθερά σημεία.

Η εξέλιξη της τροχιάς μπορεί να προβλεφθεί, ανάλογα με τις σχέσεις που έχουν μεταξύ τους η συνάρτηση με το x. Έτσι, αν ισχύει: $f(x) > x > 0$, δηλαδή αν η ευθεία $y = x$, είναι κάτω από την καμπύλη των x_i, τότε $\lim x_n \to \infty$ καθώς το $n \to \infty$, όπως φαίνεται και στην εικόνα 1.2.6.



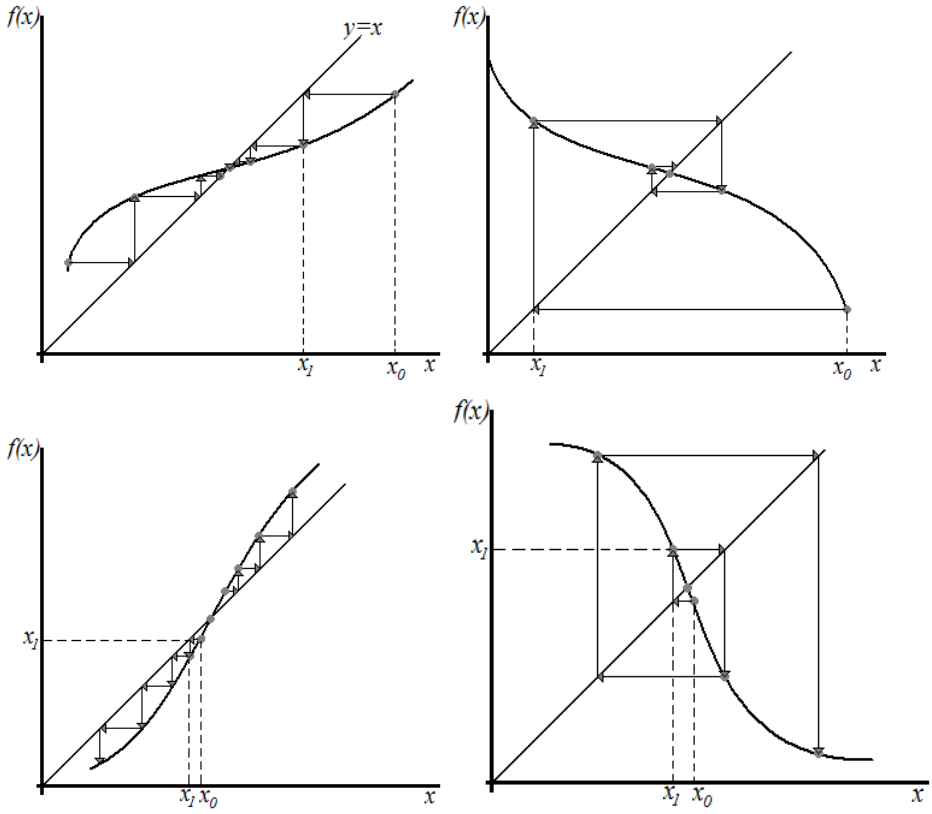
Εικόνα 1.2.6: Τροχιά για $f(x) > x > 0$. Καταλήγουμε για $n \to \infty$ να έχουμε: $\lim x_n \to \infty$.

Παρόμοια, όταν ισχύει $0 < f(x) < x$, δηλαδή όταν η ευθεία $y = x$ είναι πάνω από την καμπύλη των x_i, τότε: $\lim x_n = 0$, καθώς το $n \to \infty$.



Εικόνα 1.2.7: Για $0 < f(x) < x$, προκύπτει ότι: $\lim x_n = 0$ όταν $n \to \infty$.

Οι τροχιές γύρω από ευσταθή σταθερά σημεία μπορούν επίσης να διακριθούν, αυτή τη φορά ανάλογα με την τιμή που έχει η $f'(x)$ στο σταθερό σημείο. Για $0 < f'(x) < 1$ η τροχιά είναι ευσταθής. Για $-1 < f'(x) < 0$ έχουμε ευσταθή τροχιά με ανάκλαση. Για $1 < f'(x)$ έχουμε ασταθή τροχιά και τέλος για $f'(x) < -1$, έχουμε ασταθή τροχιά με ανάκλαση. Αυτά συνοψίζονται στην εικόνα 1.2.8.



Εικόνα 1.2.8: Από πάνω αριστερά και με τη φορά των δεικτών του ρολογιού: $0 < f'(x) < 1$, $-1 < f'(x) < 0$, $1 < f'(x)$, $f'(x) < -1$.

1.3 Σύνολα Julia

Τα *Σύνολα Julia* περιγράφτηκαν πρώτη φορά από τον Γάλλο μαθηματικό Gaston Maurice Julia (3-2-1893, 19-3-1978). Σε ηλικία 25 ετών, το 1918, εξέδωσε το 199 σελίδων άρθρο του "*Mémoire sur l'itération des fonctions rationnelles*" που εκδόθηκε στο *Journal de Mathématiques Pures et Appliquées* ένα γαλλικό περιοδικό μαθηματικών [4]. Στο άρθρο του αυτό περιέγραφε την αναδρομή μιας ρητής συνάρτησης.

Για να παράγουμε το σύνολο Julia θεωρούμε δοθέντα z_0 και c σύμφωνα με αυτά θεωρούμε την απεικόνιση της αναδρομικής συνάρτησης f_c:

$$z_{n+1} = f_c(z_n) \equiv z_n^{\,2} + c \qquad (1.3.1)$$

στο μιγαδικό επίπεδο για τις μιγαδικές παραμετρικές τιμές του c. (Η εξίσωση (1.1) αποτελεί τη λογιστική απεικόνιση $x_{n+1} = rx_n(1-x_n)$ με νέες μεταβλητές $x = 1/2 - z/r$ και $c = (2r - r^4)/4$) [5].

Από την παραπάνω αναδρομή λαμβάνουμε μια σειρά μιγαδικών αριθμών z_1, z_2, z_3... οι οποίες αποτελούν την τροχιά του z_0. Για δοθέν και σταθερό c πολλές τιμές του z_0 δίνουν τροχιές που φεύγουν στο άπειρο (δηλαδή το μέτρο $|z_n|$ αυξάνεται ανάλογα με το n, μέχρι να έχει ως όριο το άπειρο). Για μερικές όμως τιμές του c, ορισμένες τιμές των z_0 δίνουν τροχιές που καταλήγουν να είναι περιοδικές. Τέλος μερικές αρχικές τιμές δίνουν χαοτικές τροχιές. Αυτές οι (χαοτικές) αρχικές τιμές ουσιαστικά δίνουν το σύνολο Julia της απεικόνισης που συμβολίζεται ως J_c. Μερικοί συγγραφείς επίσης ορίζουν το *Πλήρες Σύνολο Julia*, το οποίο συμβολίζεται ως K_c, το οποίο είναι το σύνολο όλων των z_0 που δίνουν τροχιές που δεν τείνουν προς το άπειρο. Το "κανονικό" σύνολο Julia J_c είναι το σύνορο του πλήρους συνόλου Julia.

Θεώρημα 1:

Το σύνορο της λεκάνης έλξης του $z^* = \infty$ σχηματίζει ένα σύνολο Julia J_c της $f_c(z)$ το οποίο εξαρτάται από το *c*:

$$J_c = \textit{σύνορο του } \left\{ z \middle| \lim_{n\to\infty} f_c{}^n(z) \to \infty \right\}$$ [5].

Για το παραπάνω θεώρημα υπάρχουν ακόμη τρεις ισοδύναμοι ορισμοί.

Θεωρούμε ότι $f_c(z_{n+1}) = z_n{}^2 + c$

Δυναμικός Ορισμός: Το J_c είναι το όριο του συνόλου των σημείων που δίνουν ασταθείς περιοδικές τροχιές. Ένα σημείο $z_0 \in C$ δίνει περιοδική τροχιά αν $f_c{}^n(z_0) = z_0$ για κάποιο *n* (εδώ το $f_c{}^n$ είναι η *n*-οστή σύνθεση της f_c με τον εαυτό της). Το σημείο δίνει απωθητική περιοδική τροχιά αν: $\left| \left(f_c{}^n \right)' (z_0) \right| > 1$.

Ορισμός Μιγαδικής Ανάλυσης: Το J_c είναι το σύνολο των σημείων *z* τέτοιο ώστε η οικογένεια των συναρτήσεων $\left\{ f_c{}^n \right\}$ δεν είναι μία κανονική οικογένεια συναρτήσεων σε κάθε γειτονία του *z*.

Ορισμός Γραφικών Υπολογιστή: Το J_c είναι το όριο του συνόλου των σημείων z_0 για τα οποία $\left| \mathrm{Im}\left(f_c{}^n(z_0) \right) \right| \to \infty$, καθώς το $n \to \infty$. Αυτό σημαίνει ότι το J_c είναι το όριο του συνόλου των σημείων των οποίων οι τροχιές διαφεύγουν στο άπειρο στην κατεύθυνση του θετικού ή του αρνητικού φανταστικού άξονα. [6]

Οι παραπάνω τρεις ορισμοί είναι ισοδύναμοι με το *Θεώρημα 1*. Είναι όμως σημαντικό να σημειωθεί ότι ο *Ορισμός Γραφικών Υπολογιστή* ισχύει μόνο για πολυωνυμικές (και ρητές) συναρτήσεις. Υπάρχουν διαφορετικές κατευθύνσεις διαφυγής που δίνουν ανάλογο ορισμό για άλλου τύπου συναρτήσεις (όπως εκθετικές και τριγωνομετρικές).

Ένα άλλο σημαντικό θεώρημα από τους Julia (1918) και Fatou (1919) δηλώνει ότι το σύνολο Julia είναι τοπολογικά συνεκτικό (δεν μπορεί να γραφεί ως disjoint ένωση δύο άλλων τόπων) αν και μόνο αν $\lim_{n\to\infty} f_c{}^n(0) \not\to \infty$.

Πέρα από αυτά, υπάρχει το θεώρημα που δηλώνει πως αν το μέτρο του z_n υπερβεί το 2 για κάποιο n, τότε το σημείο αυτό δεν ανήκει στο σύνολο Julia. Αυτή η ιδιότητα (η οποία ισχύει και για το σύνολο Mandelbrot), κάνει απλό τον καθορισμό των συνθηκών ούτως ώστε να σχεδιαστούν τα σύνολα αυτά με τη βοήθεια υπολογιστή.

Όπως αναφέρθηκε προηγουμένως, για κάθε c, η πλειονότητα των τροχιών τείνει στο άπειρο. Για αυτό το λόγο το άπειρο περιγράφεται ως *ελκυστής* για το σύστημα. Το σύνολο όλων των σημείων z_0 για τα οποία οι τροχιές έλκονται στο άπειρο δημιουργεί *λεκάνη έλξης* για το άπειρο.

Ανάλογα με την επιλογή του c είναι δυνατόν να υπάρξει και άλλος ελκυστής στο σύστημα. Ενώ το άπειρο αποτελεί *σημειακό ελκυστή (point attractor),* ο δεύτερος ελκυστής μπορεί να είναι είτε σημειακός, είτε περιοδικός κύκλος (ο σημειακός ελκυστής είναι ουσιαστικά ένας περιοδικός κύκλος περιόδου ίσης με τη μονάδα). Το ακριβές σχήμα της λεκάνης έλξης στον δεύτερο ελκυστή, εξαρτάται από το c.

Αν ο δεύτερος ελκυστής δεν υπάρχει για κάποιο c, τότε το σύνολο Julia αυτού του c, είναι τοπολογικά συνεκτικό όπως έχουμε πει και είναι στην ουσία το όριο μεταξύ της λεκάνης έλξης του απείρου και της λεκάνης έλξης του πεπερασμένου ελκυστή. Αν δεν υπάρχει δεύτερος ελκυστής (δηλαδή αν το άπειρο αποτελεί τον μοναδικό ελκυστή) τότε το σύνολο Julia είναι μη συνεκτικό σύνολο *σκόνης του Cantor* (*Cantor dust*) όπως ονομάζεται [7].

Ένα από τα σημαντικά αποτελέσματα της δουλειάς του Gaston Julia είναι ότι *κάθε λεκάνη έλξης περιλαμβάνει τουλάχιστον ένα κρίσιμο σημείο της απεικόνισης* (αν η απεικόνιση είναι ρητή - και η δευτεροβάθμια απεικόνιση είναι ρητή) [8]. Εφόσον η δευτεροβάθμια απεικόνιση έχει δύο κρίσιμα σημεία (το 0 και το ∞), μπορούν να υπάρχουν το πολύ δύο λεκάνες έλξης. Επειδή ξέρουμε ότι το άπειρο είναι πάντα

ελκυστής, μπορούμε να συμπεράνουμε αν υπάρχει δεύτερος ελκυστής εξετάζοντας την τροχιά του 0.

Εφόσον το σύνολο Julia είναι το όριο λεκανών έλξης, το σύνολο αυτό καθεαυτό περιγράφεται ως *απωθητής* (*repeller*) επειδή όλες οι τροχιές του απομακρύνονται από αυτό.

Ως προς το σχεδιασμό του, το σύνολο Julia μπορεί να σχεδιαστεί, λαμβάνοντας την επαναληπτική διαδικασία που το ορίζει και να την εφαρμόζουμε για κάθε σημείο *z* του μιγαδικού επιπέδου διατηρώντας το *c* σταθερό. Με τις επαναλήψεις προκύπτει το αν η τροχιά του *z* ανήκει ή όχι στο σύνολο Julia (από το αν αυτή καταλήγει κάποτε να είναι μεγαλύτερη του 2 ή όχι). Σε έναν υπολογιστή κάθε *εικονοστοιχείο* (*pixel*) αντιστοιχεί σε ένα σημείο του μιγαδικού επιπέδου. Προφανώς διπλανά εικονοστοιχεία της οθόνης λόγω διαστάσεων δεν εκφράζουν τη συνέχεια των "διπλανών" μαθηματικών σημείων πράγμα που καθιστά εφικτό το σχεδιασμό του συνόλου Julia σε υπολογιστή, γιατί αλλιώς θα χρειαζόταν να γίνουν άπειροι υπολογισμοί ή να επινοηθεί ένας άλλος αλγόριθμος που ενδεχομένως θα ήταν πολύ πιο πολύπλοκος από τον υπάρχοντα. Το σύνολο Julia μπορεί να σχεδιαστεί και με μία μέθοδο που προέρχεται από σύστημα συναρτήσεων που ορίζει επαναληπτική διαδικασία (I.F.S παράγραφος 1.1). Η μέθοδος αυτή αποτελεί μέθοδο τυχαίας επιλογής (όπως η μέθοδος Monte Carlo). Αντί να χρησιμοποιείται η *μέθοδος διαφυγής* (*time escape method*) για να βρεθούν τα σημεία που δεν ανήκουν στο σύνολο Julia, χρησιμοποιείται η *μέθοδος τυχαίας επιλογής* μέσω του αντίστροφου τύπου για να καθορίσει τη σύγκλιση ενός σημείου προς το όριο του συνόλου.

Για παράδειγμα, για το σύνολο Mandelbrot, ο αντίστροφος τύπος είναι:

$$z_{n+1} = \sqrt{z_n - c} \tag{1.3.2}$$

Σε κάθε επανάληψη μία από τις δύο ρίζες επιλέγεται τυχαία. Μετά από έναν ικανά μεγάλο αριθμό επαναλήψεων σχεδιάζεται η τρέχουσα θέση του *z*. Η διαδικασία έπειτα επαναλαμβάνεται για μια διαφορετική, τυχαία σειρά ριζών.

Η διαδικασία "αντιστροφής" της απεικόνισης ουσιαστικά εναλλάσσει τους ελκυστές με τους απωθητές. Το σύνολο Julia τώρα είναι ένας ελκυστής και το άπειρο πιθανότατα μαζί με τον συνήθη περιοδικό κύκλο είναι απωθητές. Κάθε σύνολο Julia που έχει δομή fractal είναι ένας *παράξενος ελκυστής* για την αντίστροφη απεικόνιση.

Σημειώνεται ότι συγκεκριμένα μέρη του συνόλου Julia όπως τα σημειά του εσωτερικού του, είναι δύσκολο να γίνουν προσβάσιμα με τον αντίστροφο αλγόριθμο, επειδή τα ακραιά σημεία είναι πολύ πιο ισχυρά ως ελκυστές. Αυτό μπορεί να αποφευχθεί με τη συνεχή χρήση ίδιων ριζών τυχαίων αριθμών για διάφορες επαναλήψεις, πράγμα που καθιστά τους κεντρικούς ελκυστές πιο ισχυρούς.

1.4 Σύνολο Mandelbrot

Το Σύνολο Mandelbrot περιγράφηκε για πρώτη φορά το 1906 από τον Pierre Fatou, έναν γάλλο μαθηματικό που μελετούσε μιγαδική αναλυτική δυναμική. Ο Fatou μελέτησε αναδρομικές διαδικασίες σαν την:

$$z \to z^2 + c \tag{1.4.1}$$

Ξεκινώντας από κάποιο σημείο z_0 στο μιγαδικό επίπεδο, τα διαδοχικά σημεία παράγονται με επανειλημμένη εφαρμογή του τύπου που εκφράζεται από την (1.4.1). Η ακολουθία των σημείων που λαμβάνεται ονομάζεται τροχιά του z_0.

Ο Fatou συνειδητοποίησε ότι η τροχιά του $z_0 = 0$ υπό τον μετασχηματισμό (1.4.1) θα έδινε κάποια εικόνα για τη συμπεριφορά τέτοιων συστημάτων[9]. Όμως υπάρχει άπειρος αριθμός τέτοιων συναρτήσεων - μια για κάθε τιμή του *c*. Ο Fatou δεν είχε πρόσβαση σε υπολογιστή που μπορούσε να υπολογίσει και να σχεδιάσει τις τροχιές ικανού αριθμού τέτοιων συναρτήσεων, εντούτοις προσπάθησε να το κάνει με το χέρι. Απέδειξε ότι μόλις ένα σημείο κινείτο σε απόσταση μεγαλύτερη του 2 από την αρχική τιμή, η τροχιά θα έφευγε στο άπειρο.

Ο Fatou δεν είδε ποτέ την εικόνα αυτού που αποκαλούμε σήμερα σύνολο Mandelbrot όπως το γνωρίζουμε σήμερα, επειδή ο αριθμός των υπολογισμών που απαιτούνται για να το παράγουν είναι πολύ περισσότερο από ότι μπόρεσε να υπολογίσει και να απεικονίσει με το χέρι. Ο καθηγητής Benoît Mandelbrot ήταν ο πρώτος που χρησιμοποίησε υπολογιστή για να απεικονίσει το σύνολο.

Μαθηματικώς, το σύνολο Mandelbrot ορίζεται ως το σύνολο των σημείων *c* στο μιγαδικό επίπεδο για τα οποία η αναδρομική διαδικασία που ορίζεται από την:

$$z_{n+1} = z_n^{\,2} + c \tag{1.4.2}$$

για $z_0 = 0$, δεν τείνει στο άπειρο.

Η σειρά αυτή, εκτείνεται μαθηματικά για κάθε *c* στο μιγαδικό επίπεδο σύμφωνα με τα παρακάτω:

$$
\begin{aligned}
c &= x + iy \\
z_0 &= 0 \\
z_1 &= {z_0}^2 + c \\
&= x + iy \\
z_2 &= {z_1}^2 + c \\
&= (x + iy)^2 + x + iy \\
&= x^2 + 2ixy - y^2 + x + iy \\
&= x^2 - y^2 + x + (2xy + y)i \\
z_3 &= {z_2}^2 + c = \ldots
\end{aligned}
$$

και ούτω καθεξής.

Αν κάνουμε τις ακόλουθες δύο αντικαταστάσεις: $z_n = x_n + iy_n$ και $c = a + ib$ και υπολογίζουμε ξανά τα παραπάνω προκύπτουν οι ακόλουθες σχέσεις για τα x_{n+1} και y_{n+1}:

$$x_{n+1} = {x_n}^2 - {y_n}^2 + a \tag{1.4.3}$$

και

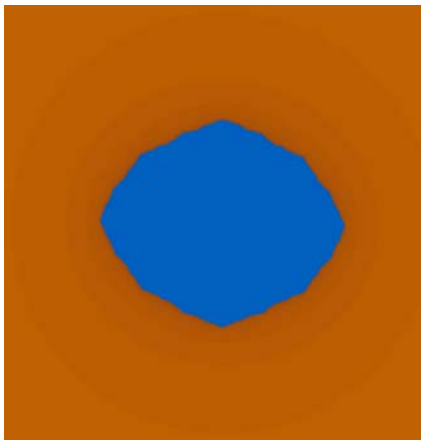
$$y_{n+1} = 2x_n y_n + b \tag{1.4.4}$$

Το σύνολο Mandelbrot μπορεί να διαιρεθεί σε άπειρα επιμέρους σύνολα, το μεγαλύτερο εκ των οποίων είναι ένα καρδιοειδές. Υπάρχει μία μετρήσιμη απειρότητα κύκλων και ελλείψεων που προσεγγίζουν πολύ καλά τον κύκλο (ο μόνος πραγματικός κύκλος είναι ο μεγαλύτερος και βρίσκεται δίπλα από το καρδιοειδές) που εφάπτονται του καρδιοειδούς αλλά έχουν διάφορες διαμέτρους και πολλές εξ αυτών τείνουν ασυμπτωτικά στο μηδέν. Πέρα από αυτό, ο κάθε κύκλος και κάθε έλλειψη έχει τη δική του μετρήσιμη απειρότητα που αφορά σε μικρότερους κύκλους που προέρχονται από την περιφέρεια του και αυτό το σύνολο των κύκλων έχει επίσης διαμέτρους που τείνουν ασυμπτωτικά στο μηδέν. Η απειρότητα των κύκλων και ελλείψεων αυτών επαναλαμβάνεται σε κάθε κλίμακα και δίνει έτσι τη μορφοκλασματική δομή του

συνόλου. Σημειώνεται ότι η απειρότητα αυτή των κύκλων και ελλείψεων δεν εξαντλεί το σύνολο Mandelbrot. Εμφανίζονται και άλλα καρδιοειδή καθώς και άλλοι σχηματισμοί που δεν είναι ενωμένοι με τους κύκλους.

Το σύνολο Mandelbrot, δημιουργήθηκε από τον Benoit Mandelbrot ως ενδεικτικό για τα σύνολα Julia: κάθε σημείο στο μιγαδικό επίπεδο του συνόλου Mandelbrot, αντιστοιχίζεται σε διαφορετικό σύνολο Julia. Τα σημεία στο εσωτερικό του συνόλου Mandelbrot αντιστοιχίζεται στα συνεκτικά σύνολα Julia και τα σημεία στο εξωτερικό του συνόλου αντιστοιχίζονται στα μη συνεκτικά σύνολα Julia.

Τα σύνολα Julia που προκύπτουν από τα σημεία στο εσωτερικό του συνόλου Mandelbrot όπως για παράδειγμα αυτά κοντά στο (0,0), είναι απλά γεωμετρικά σχήματα και αυτό που προκύπτει για $\mathrm{Re}(c) = \mathrm{Im}(c) = 0$, είναι ομόκεντροι κύκλοι. Τα σύνολα Julia που προέρχονται από σημεία έξω από το σύνολο Mandelbrot και σε αρκετή απόσταση σε σύγκριση με το μεγαλύτερο καρδιοειδές, εμφανίζονται σα σκόνη διαφόρων χρωμάτων. Επίσης το σύνολο Mandelbrot περιέχει σε κάποια σημεία του δομές που θυμίζουν έντονα σύνολα Julia.



Εικόνα 1.4.1: Το σύνολο Julia στο σημείο (-0,182,0.028).

Εικόνα 1.4.2: Το σύνολο Julia στο σημείο (-1,-1).

Ο σχεδιασμός του συνόλου Mandelbrot βασίζεται στον υπολογισμό των τροχιών που προκύπτουν από όλα τα *c* που απεικονίζονται.

Όπως και στο σύνολο Julia, μπορεί να αποδειχθεί ότι αν το μέτρο του μιγαδικού z_n γίνει μεγαλύτερο από το 2 (σε καρτεσιανή μορφή αν $\sqrt{x_n^{\,2} + y_n^{\,2}} > 2$), η τροχιά της επαναληπτικής διαδικασίας για το *c* που μελετάται, θα τείνει στο άπειρο και συνεπώς δε θα ανήκει στο σύνολο Mandelbrot. Αυτή η τιμή, που είναι γνωστή και ως *τιμή διαφυγής* ή *εγκατάλειψης* (*bail out value*), επιτρέπει τον τερματισμό των υπολογισμών σύντομα για σημεία που βρίσκονται έξω από το σύνολο Mandelbrot.

Η απόδειξη του θεωρήματος αυτού ακολουθεί παρακάτω:

Θεώρημα:

Έστω η συνάρτηση $g(z) = z^2 + c$ *χρησιμοποιείται ως κανόνας για την επαναληπτική διαδικασία και έστω το* $|c| < 2$ *(συνθήκη που ισχύει στην περιοχή όπου απεικονίζονται τα σύνολα Mandelbrot και Julia). Θεωρούμε ένα σημείο εκκίνησης των επαναλήψεων* z_0 *στο μιγαδικό επίπεδο. Αν το μέτρο οποιουδήποτε σημείου της τροχιάς που ξεκινάει από το* z_0*, ξεπεράσει το 2, τότε η τροχιά που ξεκινάει από το* z_0 *διαφεύγει στο άπειρο.*

Απόδειξη:

Υποθέτουμε ότι το σημείο z_1 *ανήκει στην τροχιά και* $|z_1| < 2$.

Μέσω της τριγωνικής ανισότητας λαμβάνουμε:

$|g(z_1)| = |z_1^{\,2} + c| \geq |z_1|^2 - c > |z_1|^2 - |z_1| = (|z_1| - 1)|z_1|$,
καθώς $|z_1| < 2$.

Για τον ίδιο λόγο $|z_1| - 1 = 1 + \varepsilon_1 > 1$. *Έτσι* $|z_2| = |g(z_1)| = (1 + \varepsilon_1)|z_1| > 2$

Εφαρμόζοντας την ίδια μέθοδο άλλη μια φορά για κάποιο στοιχείο $\varepsilon_2 > 0$*, έχουμε:*

$|z_3| = |g(z_2)| = (1 + \varepsilon_2)|z_2| = (1 + \varepsilon_1)(1 + \varepsilon_1)|z_1|$.

Συνεχίζοντας επαγωγικά, αποδεικνύεται ότι η τροχιά διαφεύγει στο άπειρο καθώς οι παράγοντες που πολλαπλασιάζουν το $|z_1|$ *είναι πάντοτε μεγαλύτεροι της μονάδας.*

Για σημεία στο εσωτερικό του συνόλου, οι τιμές του *c* δεν δίνουν τροχιές του z_n που τείνουν στο άπειρο και κατά συνέπεια οι υπολογισμοί δε φτάνουν ποτέ σε ένα τέλος. Οπότε μετά από έναν ικανό αριθμό επαναλήψεων μπορούν να σταματήσουν οι υπολογισμοί. Εφόσον δεν μπορούμε να κάνουμε άπειρους υπολογισμούς, συμπεραίνεται ότι το σύνολο Mandelbrot που απεικονίζουμε είναι μία προσέγγιση του πραγματικού συνόλου. Φυσικά, αν κάποιος ασχοληθεί με τη δημιουργία του συνόλου Mandelbrot σε υπολογιστή, θα είναι σε θέση να καταλάβει πότε τα σημεία του συνόλου που απεικονίζονται δίνουν μία ικανοποιητική προσέγγιση του συνόλου και αν δε συμβαίνει αυτό, τότε αυξάνεται ο αριθμός των επαναλήψεων για να ληφθεί μία καλύτερη προσέγγιση, με κόστος την καθυστέρηση στους υπολογισμούς και στην απεικόνιση του συνόλου.

Το σύνολο Mandelbrot σύμφωνα με τους μαθηματικούς ορισμούς που παρατίθενται παραπάνω, θα είναι ασπρόμαυρο, γιατί ένα σημείο είτε ανήκει στο σύνολο, είτε όχι. Οι περισσότερες απεικονίσεις που δημιουργούνται από υπολογιστή όμως σχεδιάζονται με κάποιο βάθος χρώματος. Στην πιο κοινή μέθοδος δημιουργίας, τα σημεία τα οποία διαφεύγουν στο άπειρο, επομένως αυτά που δεν ανήκουν στο σύνολο, το χρώμα τους υποδεικνύει τον αριθμό των επαναλήψεων που χρειάστηκαν για να φτάσουν στο τελευταίο σημείο της τροχιάς τους από την αρχική τιμή. Έτσι εμφανίζονται σχήματα που θυμίζουν ομόκεντρους κύκλους, η υπέρθεση των οποίων δίνει μία προς μία τις προσεγγίσεις του συνόλου.

Για να καταλάβουμε αν το σημείο *c*, πρόκειται να ανήκει ή όχι στο σύνολο Mandelbrot, πρέπει να υπολογιστεί η τροχιά του z_n σε κάθε επανάληψη σύμφωνα με τα παρακάτω:

Για $z_n = x_n + iy_n$, τότε $|z_n| = \sqrt{{x_n}^2 + {y_n}^2}$. Μία από τις πολλές βελτιστοποιήσεις που μπορούμε να κάνουμε είναι να μην υπολογίζουμε τη ρίζα για να ελέγξουμε αν $\sqrt{{x_n}^2 + {y_n}^2} > 2$, αλλά να ελέγξουμε αν: ${x_n}^2 + {y_n}^2 > 4$. Έτσι αν το $|z_n|^2 < 4$, χρωματίζεται το σημείο του συνόλου με κάποιο συγκεκριμένο χρώμα (συνήθως μαύρο), αλλιώς χρωματίζεται με κάποια φόρμουλα αντιστοίχισης χρώματος, ανάλογα με την τιμή του *n*. Χρησιμοποιώντας τον αριθμό των επαναλήψεων που απαιτείται από το σημείο για να διαφύγει, είναι η ευκολότερη και η πιο κοινή μέθοδος καταγραφής της "ταχύτητας" με την οποία διαφεύγει αυτό το σημείο. Αυτός ο αριθμός μπορεί απευθείας να αντιστοιχιστεί με κάποιο χρώμα μέσω ενός πίνακα αναφοράς που συνδέει τον αριθμό των επαναλήψεων με μία παλέττα χρώματος, ή με τη χρήση κάποιου κατάλληλου αλγόριθμου.

Κλείνοντας την παράγραφο θα γίνει αναφορά σε μία γενίκευση για το σύνολο Mandelbrot, η οποία κατηγοριοποιεί ως σύνολα Mandelbrot, όλα τα σύνολα που προκύπτουν από επαναληπτική διαδικασία συναρτήσεων (όπως αυτή της αναδρομικής διαδικασίας: $z_{n+1} = {z_n}^2 + c$) στην απεικόνιση των οποίων αυτό που μεταβάλλεται στους άξονες της απεικόνισης στο μιγαδικό επίπεδο είναι η πραγματική και η φανταστική τιμή του σταθερού όρου *c*, όπως στο σύνολο Mandelbrot που ορίστηκε. Έτσι για παράδειγμα ως σύνολο Mandelbrot θα αναφέρεται και η απεικόνιση: $f^c(z) = z^3 + c$.

Αναφορές:

[1]: Wikipedia, the Free Encyclopedia. (□□□□□□□□□ □□□π□π□□□□□: 6 □□□□□□□□□ 2005). *Fractal.* □□□□□□□□□ 25 Σεπτεμβρίου, 2005, απ□ http://en.wikipedia.org/wiki/Fractal

[2]: Bogomonly A. (1996, Copyright 1996-2005 Alexander Bogomonly). *Fractal Curves and Dimension.* □□□□□□□□□ 1, □□□□□□□□□, 2005, □π□: http://www.cut-the-knot.org/do_you_know/dimension.shtml

[3]: Elert G. (1995). The Chaos Hypertextbook ™, *Iterations.* Ανακτήθηκε: 4 Οκτωβρίου, 2005, από http://hypertextbook.com/chaos/12.shtml

[4]: Wikipedia, the Free Encyclopedia. (□□□□□□□□□ □□□π□π□□□□□: 6 □□□□□□□□□ 2005). *Gaston Julia.* □□□□□□□□□ 25 Σεπτεμβρίου, 2005, απ□ http://en.wikipedia.org/wiki/Gaston_Julia

[5]: Schuster H.G. (2005), Deterministic Chaos: An Introduction. John Wiley & Sons.

[6]: Devaney R.L. (1989). Film and Video as a Tool in Mathematical Research. *The Mathematical Intelligencer 11(2).* 33-34.

[7]: Elert G. (1995). The Chaos Hypertextbook ™, *Julia Sets.* Ανακτήθηκε: 4 Οκτωβρίου, 2005, από http://hypertextbook.com/chaos/22.shtml

[8]: Wikipedia, the Free Encyclopedia. (□□□□□□□□□ □□□π□π□□□□□: 26 Σεπτεμβρίου, 2005). *Julia Set.* □□□□□□□□□ 17 Σεπτεμβρίου, 2005, απ□ http://en.wikipedia.org/wiki/Julia_Set

[9]: Knill O., (2005), The Mandelbrot Set. □□□□□□□□□ 7 □□□□□□□□□, 2005, □π□: http://www.math.harvard.edu/archive/118r_spring_05/handouts/mandelbrot.pdf

Βιβλιογραφία:

1. Μπούντης Τ., (2004), Ο θαυμαστός κόσμος των fractal. Leader Books, Αθήνα.

2. Voytazis G. (n.d.). One Dimensional Discrete Dynamical Systems, (σημειώσεις).

2.1 Η Γλώσσα Προγραμματισμού C♯

Η C♯ που προφέρεται see sharp (si: ʃa:p, σύμφωνα με το IPA - International Phonetic Alphabet, αναθεώρηση 1993 [1]), είναι μια *αντικειμενοστρεφής* (*object-oriented*) γλώσσα προγραμματισμού, που αναπτύχθηκε από τη Microsoft σα μέρος της παρουσίασης και εξέλιξης της πλατφόρμας .NET της ίδιας εταιρίας. Η Microsoft βασίζει τη C♯ στις γλώσσες προγραμματισμού C++ και Java. Η C♯, σχεδιάστηκε έτσι ώστε να παρέχει μία ισορροπία μεταξύ της C++ με τη μεθοδολογία της *ραγδαίας ανάπτυξης εφαρμογών* (*RAD - Rapid Application Development*), της Visual Basic, της Delphi και της Java. Το όνομα της C♯ σύμφωνα με την εταιρία που την κατασκεύασε δηλώνει εξέλιξη από τη C++. Το σύμβολο ♯ εκφράζει τη δίεση στη μουσική σημειολογία και σημαίνει την εκτέλεση της νότας που το ακολουθεί ένα ημιτόνιο ψηλότερα από τον τόνο που εκφράζει η εν λόγω νότα. Έτσι δηλώνεται με διαφημιστικού τύπου συνειρμό η εξέλιξη της C♯ σε σχέση με τη C++, που η εταιρία θέλει να δηλώσει.

Η *δομή .NET* (*.NET Framework*) είναι μία *πλατφόρμα ανάπτυξης λογισμικού* (*software development platform*) επικεντρωμένη στη ραγδαία ανάπτυξη εφαρμογών, *ανεξαρτησία πλατφόρμας* (*platform independence - cross platform*) και *διαφάνεια δικτύου* (*network transparency*). Οι χαρακτηρισμοί αυτοί δηλώνουν ότι ως περιβάλλον ανάπτυξης εφαρμογών λογισμικού έχει ως στόχους τη λειτουργία τους σε περισσότερες από μία πλατφόρμες (π.χ. διάφορες εκδόσεις των Windows, κυρίως μέσω των εφαρμογών διαδικτύου που παράγονται) και την ανάγκη εξασφάλισης προστασίας μόνο των δεδομένων του χρήστη του δικτύου και όχι του ίδιου του δικτύου. Η δομή .NET απευθύνεται σε εφαρμογές *διακομιστών* (*server*) και *ατομικού περιβάλλοντος εργασίας* (*desktop*). Η δομή .NET παρέχει αντικειμενοστρεφείς

λειτουργίες που έχουν ως σκοπό να υλοποιήσουν *κατοπτρισμό* (*reflection*) ώστε να αποκρύπτονται οι χαμηλού επιπέδου λειτουργίες της εφαρμογής που δε χρειάζονται στη διαδικασία ανάπτυξης. Εντούτοις μπορούν αν ο προγραμματιστής το χρειαστεί να χρησιμοποιηθούν. Η δομή .NET ανταγωνίζεται τη γλώσσα προγραμματισμού Java που έχει αναπτυχθεί από τη Sun και την τεχνολογία J2EE και ως αποτέλεσμα αυτού αντιγράφει πολλά στοιχεία αυτών.

Η C♯ είναι η γλώσσα προγραμματισμού που αντανακλά πιο άμεσα τη δομή .NET στην οποία βασίζεται αποκλειστικά. Κατά τη *σύνθεση* (*compilation*) του προγράμματος, ο κώδικας του μεταφράζεται σε *ενδιάμεσο κώδικα* (*intermediate language code*) και συγκεκριμένα σε κώδικα της MSIL (Microsoft Intermediate Language) που παρέχει πληροφορίες απευθείας για τον επεξεργαστή και τα περιφερειακά του, οι οποίες όπως γίνεται κατανοητό τρέχουν σε ένα ενδιάμεσο *περιβάλλον εκτέλεσης* (*runtime environment*). Ως εκ τούτου δεν υπάρχει *μη-διευθυνόμενο πρόγραμμα* (*unmanaged program*) στη C♯, πράγμα που σημαίνει ότι το πρόγραμμα μέσω του περιβάλλοντος στο οποίο αυτό τρέχει μπορεί να διακόψει κάποια λειτουργία που λαμβάνει χώρα στον επεξεργαστή και να ανακαλέσει πληροφορίες σχετικές με τη συγκεκριμένη οδηγία που έχει προς τον επεξεργαστή. Πριν να γίνει σύνθεση του προγράμματος, η ενδιάμεση γλώσσα, συντίθεται σε *τοπικό κώδικα μηχανής* (*native machine code*). Εφόσον γίνεται αυτή η ενδιάμεση σύνθεση, η διευθυνόμενη εκτέλεση εγγυάται το ακριβές εύρος λειτουργιών που πρόκειται να εκτελέσει ο κώδικας. Έτσι μπορούμε να έχουμε *συλλογή απορριμάτων* (*garbage collection*), *χειρισμό εξαιρέσεων* (*exception handling*), *κλάσεις* (*classes*), *διεπαφές* (*interfaces*), *εντολείς* (*delegates*) και άλλα χαρακτηριστικά.

Συγκρινόμενη με τις γλώσσες προγραμματισμού C και C++, η C♯ έχει κάποια επιπλέον και κάποια ελλειπή χαρακτηριστικά, μερικά από τα οποία είναι:

- Δεν μπορούν να χρησιμοποιηθούν *μη επεξεργασμένοι δείκτες* (*raw pointers*), παρά μόνο σε μία συγκερκιμένη μέθοδο της σύνθεσης που χαρακτηρίζεται ως *μη ασφαλής* (*unsafe mode*). Η πρόσβαση στα *αντικείμενα* (*objects*) γίνεται ως επί το πλείστον με ασφαλείς *αναφορές* (*references*), ώστε να μην μπορούν να γίνουν *άκυρες* (*invalid*) και οι περισσότεροι αριθμητικοί τύποι μεταβλητών ελέγχονται για *υπερχείλιση* (*overflow*). Οι δείκτες μπορούν να χρησιμοποιηθούν μόνο στους λεγόμενους *τύπους αξίας* (*value types*). Τα αντικείμενα που χειρίζεται ο συλλέκτης απορριμάτων μπορούν να είναι μόνο *τύποι αναφοράς* (*reference types*). Διευκρίνιση:
  Οι τύποι αξίας περιέχουν απευθείας τα δεδομένα τους. Έτσι κάθε μεταβλητή τύπου αξίας, περιέχει και την τιμή η οποία της έχει ανατεθεί. Οι τύποι αξίας αποθηκεύουν τους ίδιους και τις τιμές τους στην περιοχή της μνήμης που λέγεται *στοίβα* (*stack*). Η περιοχή αυτή είναι μια περιοχή της μνήμης όπου αποθηκεύονται στοιχεία με τη μέθοδο: *τελευταίο-μέσα*, *πρώτο-έξω* (*last-in*, *first-out*).
  Οι τύποι αναφοράς εμπεριέχουν μία αναφορά στα δεδομένα τους. Τα αντικείμενα, για παράδειγμα, είναι τύποι αναφοράς. Περισσότεροι από ένας τύποι αναφοράς μπορούν να εμπεριέχουν αναφορά στο ίδιο αντικείμενο. Έτσι είναι δυνατόν οι πράξεις σε έναν τύπο αναφοράς να επηρεάζουν άλλες μεταβλητές που αναφέρονται στο ίδιο αντικείμενο. Οι τύποι αναφοράς εμπεριέχουν αναφορά στην περιοχή της μνήμης όπου βρίσκονται τα αντικείμενα και λέγεται *σωρός* (*heap*).

- Τα αντικείμενα δεν μπορούν να απελευθερωθούν *σαφώς* (*explicitly*) δηλαδή με μία εντολή στο πρόγραμμα όπως γίνεται σε άλλες αντικειμενοστρεφίες γλώσσες προγραμματισμού, αλλά αυτό για αυτό το σκοπό υπάρχει ο συλλέκτης απορριμάτων, ο οποίος τα αφαιρεί όταν δεν υπάρχει καμία αναφορά σε αυτά. (Εντούτοις αντικείμενα που αντιπροσωπεύουν μη διευθυνόμενους *πόρους* (*resources*), μπορούν να δεχτούν εντολή να τους απελευθερώσουν μέσω της διεπαφής `IDisposable`).
- Μόνο *απλή κληρονομικότητα* (*single inheritance*) είναι διαθέσιμη, αλλά μία κλάση μπορεί να υλοποιήσει όσες *αφηρημένες* (*abstract*) διεπαφές ζητήσει ο προγραμματιστής. Αυτό έχει ως σκοπό την απλοποίηση της υλοποίησης και λειτουργίας του περιβάλλοντος εκτέλεσης.
- Η C♯, είναι περισσότερο *ασφαλής στη γραφή* (*typesafe*) από τη C++. Οι μόνες *υπονοούμενες* (*implicit*) μετατροπές που επιτρέπονται ως προεπιλογή, είναι οι ασφαλείς μετατροπές, όπως π.χ. *ακέραιοι* (*integers*) σε *δεκαδικούς κινητής υποδιαστολής* (*floating point numbers*). Δεν υπάρχουν συνεπείς μετατροπές μεταξύ *λογικών τιμών Bool* (*Booleans*) σε ακεραίους, μεταξύ *μελών απαρίθμησης* (*enumeration members*) και ακεραίους, όπως επίσης δεν υπάρχουν κενοί δείκτες. Τέλος κάθε υπονοούμενη μετατροπή που γίνεται από τον προγραμματιστή πρέπει να δηλώνεται σαφώς ως τέτοια σε αντίθεση με τους *κατασκευαστές αντιγραφής* (*copy constructors*) της C++.
- Η σύνταξη για δήλωση *πίνακα* (*array*) είναι διαφορετική στη C♯ από αυτήν στη C++.

  Γίνεται ως εξής: `int[] a = new int[5]`

  και όχι: `int a[5]`

- Τα μέλη απαρίθμησης μπαίνουν στο δικό τους *χώρο ορισμού* (*namespace*). Ο Χώρος Ορισμού είναι ένα σύστημα οργάνωσης που χρησιμοποιείται για να αναγνωρίζει ομάδες σχετικών μεταξύ τους κλάσεων.
- Στη C♯ δεν υπάρχουν *πρότυπα* (*templates*)
- Οι *ιδιότητες* (*properties*) είναι διαθέσιμες και επιτρέπουν στις *μεθόδους* (*methods*) να καλούνται μέσω σύνταξης που θυμίζει την *πρόσβαση* σε *μέλος δεδομένων* (*data member access*).
- Είναι διαθέσιμος πλήρης κατοπτρισμός.

Παρακάτω ακολουθεί ένα παράδειγμα του κλασικού προγράμματος "Hello World" στη C♯ που τυπώνει στην κονσόλα το μήνυμα "Hello World":

```
using System;

namespace Hello_World
{

      public class HelloWorldClass
      {

            public static void Main()
            {
                  System.Console.WriteLine("Hello World");
            }
      }
 }
```

Κάθε γραμμή κώδικα παραπάνω εκτελεί και συγκεκριμένη λειτουργία.

Έχουμε αρχικά τη γραμμή κώδικα:

```
using System;
```

που δηλώνει τη χρήση των πόρων της *βιβλιοθήκης* (*library*) που κατονομάζονται (εν προκειμένω το `System`). Συνήθως το `using` αναφέρεται πολλές φορές στην αρχή ενός προγράμματος, ανάλογα με το πλήθος των αναφορών στη βιβλιοθήκη που ο προγραμματιστής χρειάζεται. Φυσικά, ανάλογα με τις λειτουργίες που θέλουμε να επιτελεί, υπάρχουν διαφορετικές αναφορές. Ενδεικτικά αναφέρονται μερικές:

`using System.Drawing; using System.Windows.Forms; using System.Data;`

Ακολουθεί ο *χώρος ορισμού* (*namespace*) του προγράμματος μας. Εφόσον έχουμε να κάνουμε με ένα τόσο μικρό πρόγραμμα αυτός μπορεί και να παραληφθεί οπότε και εννοείται να έχει το ίδιο όνομα με το όνομα του προγράμματος. Εδώ δηλώνεται και ανοίγει άγκιστρο για να συμπεριλάβει τα στοιχεία που τον αποτελούν:

```
namespace Hello_World {
```

Έπειτα ορίζεται η κλάση που πρέπει να υλοποιηθεί για να λειτουργήσει το πρόγραμμα. Αυτό γίνεται γράφοντας:

```
public class HelloWorldClass {
```

Τα περιοχέμενα μετά από το άγκιστρο που ακολουθεί τον ορισμό της, αποτελούν τα στοιχεία της κλάσης. Η λέξη `public` που χρησιμοποιείται δηλώνει ότι αντικείμενα που δεν ανήκουν σε αυτή την κλάση, μπορούν να τη χρησιμοποιήσουν ελεύθερα.

Ακολουθεί το σημείο του κώδικα που δηλώνει τη μέθοδο (ή συνάρτηση) που εκτελεί το πρόγραμμα με το όνομα `Main`. Αυτή ορίζεται ως εξής:

```
public static void Main() {
```

Και είναι το σημείο όπου το πρόγραμμα ξεκινάει την εκτέλεση του. Ο όρος `public` δηλώνει ότι η συνάρτηση μπορεί να χρησιμοποιηθεί ελεύθερα από αντικείμενα που δεν προέρχονται από την κλάση `HelloWorldClass`. Ο όρος `static` είναι ενδεικτικός για το πως θα γίνει η εκτέλεση και δηλώνει ότι:

i) η συνάρτηση ανήκει στην κλασή,
ii) η υλοποίηση της συνάρτησης `Main` θα γίνει πριν την υλοποίηση κάποιου αντικειμένου που προέρχεται από την εν λόγω κλάση και τέλος,
iii) το `static` μέλος (εδώ η συνάρτηση `Main`), είναι κοινό σε όλες τις υλοποιήσεις της κλάσης.

Ο όρος `void` δηλώνει ότι η συνάρτηση είναι τύπου `void`, δηλαδή δεν επιστρέφει κάποια τιμή. Στα άγκιστρα περιλαμβάνονται οι λειτουργίες της συνάρτησης.

Τέλος έχουμε τη δήλωση των λειτουργιών που θα επιτελέσει η συνάρτηση:

```
System.Console.WriteLine("Hello World");
```

Εδώ στην ουσία γράφεται το ζητούμενο της μεθόδου. Η λέξη `Console` αναφέρεται σε κλάση του `System` και αντιπροσωπεύει μία κονσόλα γραμμής εντολών (σαν αυτή του ψευδό-DOS που παρέχουν τα Windows XP), όπου ένα πρόγραμμα μπορεί να τυπώνει και να δέχεται κείμενο. Το πρόγραμμα καλεί τη μέθοδο `WriteLine` της κλάσης `Console` η οποία έχει ως αποτέλεσμα η *σειρά χαρακτήρων* (*string*) που δίνεται σε αυτήν ως *είσοδος* (*input*) να εμφανιστεί στην κονσόλα.

Έτσι στην έξοδο λαμβάνουμε το μήνυμα Hello World.

2.2 Η εφαρμογή του αλγορίθμου απεικόνισης του συνόλου Mandelbrot

Από αυτό το τμήμα της εργασίας και έπειτα θα αναλυθεί το πως έγινε η εφαρμογή των αλγορίθμων που παράγουν τα διάφορα μορφοκλασματικά σύνολα που μας απασχολούν.

Για το σύνολο Mandelbrot αρχικά θα ασχοληθούμε πρώτα με το ζητούμενο και τα δεδομένα που εισάγονται από αυτό. Το ζητούμενο είναι: "να σχεδιαστεί στον υπολογιστή, με τη βοήθεια κάποιας γλώσσας προγραμματισμού, το σύνολο Mandelbrot". Συμπεραίνουμε ότι πρέπει να σχεδιαστεί το σύνολο που προέρχεται από την Εξίσωση 1.4.2: $z_{n+1} = z_n^{\,2} + c$ για τα οποία το μέτρο του z_{n+1} δεν τείνει στο άπειρο . Αναλύουμε την παραπάνω επαναληπτική διαδικασία. Εφόσον $z_n = x_n + iy_n$ και $c = a + ib$ κάνοντας πράξεις υπολογίζουμε τα x_{n+1} και y_{n+1} οπότε έχουμε:

$x_{n+1} = x_n^{\,2} - y_n^{\,2} + a$ και $y_{n+1} = 2x_n y_n + b$.

Επίσης γνωρίζουμε πως αν το μέτρο του z_{n+1} υπερβεί το 2, θα είναι εξασφαλισμένο ότι το σημείο αυτό θα διαφύγει στο άπειρο, γεγονός που περιορίζει δραματικά τις επεναλήψεις που χρειάζεται να γίνουν για τα σημεία που ανήκουν στο σύνολο Mandelbrot. Από την άλλη πλευρά όμως πρέπει να βρεθεί ένας πεπερασμένος αριθμός, ο αριθμός επαναλήψεων, ο οποίος αφενός πρέπει να είναι ικανά μεγάλος ώστε να μη σταματάει η διαδικασία πριν να βρεθεί πραγματικά αν το σύνολο ανήκει όντως στο σύνολο Mandelbrot ή όχι και αφετέρου να είναι αρκούντως μικρός ώστε η συνολική διαδικασία να γίνεται μέσα σε λίγο χρόνο. Η αναζήτηση αυτού του αριθμού θα γίνει κυρίως με εφαρμογή της μεθόδου δοκιμής-και-λάθους και με σύγκριση των εικόνων που λαμβάνονται με τη γνωστή μορφή του συνόλου Mandelbrot.

Έτσι, εφόσον είναι γνωστός ο αριθμός των επαναλήψεων μπορούμε να υπολογίζουμε το μέτρο της τροχιάς για κάθε *n* και να γίνει έλεγχος σε κάθε βήμα το αν το μέτρο $z_{n+1} = \sqrt{x_{n+1}^{\ 2} + y_{n+1}^{\ 2}}$ υπερβαίνει το 2 και να διακόπτεται η διαδικασία για το συγκεκριμένο *c* στο μιγαδικό επίπεδο για το οποίο υπολογίζεται το αν αυτό ανήκει ή όχι στο σύνολο Mandelbrot.

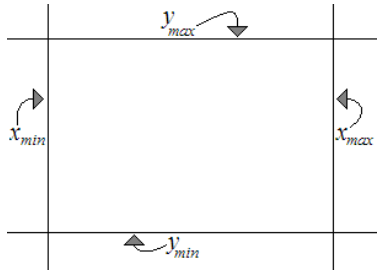
Περαιτέρω, για την απεικόνιση, πρέπει να έχουμε ένα *στοιχείο* (*component*) το οποίο να παρέχει τη δυνατότητα σχεδιασμού πάνω σε αυτό και θα πρέπει να έχει γενικά ορθογώνιο σχήμα (Bitmap σε περαιτέρω αναφορά). Το στοιχείο αυτό θα εκφράζει το μιγαδικό επίπεδο και κάθε εικονοστοιχείο του θα αντιστοιχίζεται σε ένα σημείο του μιγαδικού επιπέδου.

Ξεκινώντας από ένα εικονοστοιχείο του παραπάνω στοιχείου, λαμβάνουμε το αντίστοιχο *c* και για αυτή την τιμή, ξεκινούμε την αναδρομική διαδικασία $z_{n+1} = z_n^{\ 2} + c$, δηλαδή υπολογίζουμε τα: $x_{n+1} = x_n^{\ 2} - y_n^{\ 2} + a$ και $y_{n+1} = 2x_n y_n + b$ για την τιμή του z_0 που έχουμε ορίσει. Ελέγχουμε αν κάθε φορά το μέτρο $z_{n+1} = \sqrt{x_{n+1}^{\ 2} + y_{n+1}^{\ 2}}$ υπερβαίνει το 2 και αν όχι σταματάμε τη διαδικασία στο μέγιστο αριθμό επαναλήψεων (*n*) που έχουμε ορίσει. Αν το υπερβεί σε κάποια ενδιάμεσο βήμα, σταματάμε τότε τη διαδικασία. Σημειώνεται εδώ, ότι στην εφαρμογή εισάγεται χρώμα στα σημεία που δεν ανήκουν στο σύνολο Mandelbrot, ανάλογα με το πόσες επαναλήψεις χρειάζονται σε κάθε σημείο για να διαφύγει στο άπειρο. Ο χρωματισμός του εικονοστοιχείου γίνεται μέσω ενός τύπου αντιστοίχισης του αριθμού της τελευταίας επανάληψης που έγινε με κάποιο χρώμα.

Για τον αλγόριθμο που θα σχεδιάζει το σύνολο Mandelbrot χρειαζόμαστε έναν αριθμό μεταβλητών κάθε μία εκ των οποίων θα εκφράζει μία τιμή χρήσιμη στην απεικόνιση του συνόλου. Έτσι θα έχουμε τις παρακάτω μεταβλητές:

$\mathrm{Re}(c)$, $\mathrm{Im}(c)$, $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$, x_n, y_n, *απόσταση εικονοστοιχείων στον άξονα x*, *απόσταση εικονοστοιχείων στον άξονα y*, *tempx* (δεκαδικοί)
looper, *z*, *w* (ακέραιοι)

- Οι μεταβλητές $\mathrm{Re}(c)$ και $\mathrm{Im}(c)$ είναι το πραγματικό και το φανταστικό μέρος αντίστοιχα, του τρέχοντος *c*.
- Η $x_{\min}$ είναι η ελάχιστη τιμή του πραγματικού μέρους του *c* που εμφανίζεται. Δηλαδή η ελάχιστη τιμή του άξονα *x* του μιγαδικού επιπέδου, όπως απεικονίζεται από το πρόγραμμα μέσω του στοιχείου Bitmap. Αντίστοιχα, η $x_{\max}$ είναι η μέγιστη τιμή του πραγματικού μέρους του *c* που εμφανίζεται, η $y_{\min}$ η ελάχιστη τιμή του φανταστικού μέρους του *c* και τέλος η $y_{\max}$ η μέγιστη τιμή του φανταστικού μέρους του *c*. Τα $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$ σχηματίζουν ένα τετράγωνο "παράθυρο" στο εσωτερικό του οποίου σχεδιάζεται το σύνολο Mandelbrot (Εικόνα 2.2.1).



Εικόνα 2.2.1: Το "παράθυρο" που σχηματίζεται από τις τιμές των μεταβλητών $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$.

- Η μεταβλητή x_n είναι η τρέχουσα τιμή του πραγματικού μέρους της τροχιάς του *z* (x_n) και εκφράζει την τιμή $\mathrm{Re}(z_{n+1}) = x_{n+1}$ αλλά και την τιμή $\mathrm{Re}(z_n) = x_n$. Αυτή στο πρόγραμμα θα υπολογίζεται σύμφωνα με την εξίσωση (1.4.3):

$x_n = x_n^2 - y_n^2 + \mathrm{Re}(c)$. Για το αρχικό x_n τίθεται τιμή προγραμματιστικά (για το σύνολο Mandelbrot όπως ορίστηκε, τίθεται ίση με μηδέν). Έπειτα κατά τη διάρκεια της επαναληπτικής διαδικασίας λαμβάνει τιμές ανάλογα με την τρέχουσα τιμή της ίδιας μεταβλητής μέσω της εξίσωσης (1.4.3).

- Όμοια η μεταβλητή y_n, εκφράζει το φανταστικό μέρος της τροχιάς του *z*, παίρνει τιμές σύμφωνα με την εξίσωση (1.4.4): $y_n = 2x_n y_n + \mathrm{Im}(c)$ και η αρχική της τιμή τίθεται προγραμματιστικά.
- Η μεταβλητή *απόσταση εικονοστοιχείων στον άξονα x* εξαρτάται από τις $x_{\min}$ και $x_{\max}$ και δηλώνει την απόσταση στον πραγματικό άξονα του μιγαδικού επιπέδου που εκφράζουν δύο διπλανά μεταξύ τους εικονοστοιχεία. Έτσι θα έχουμε:

  απόσταση εικονοστοιχείων στον άξονα x = $(x_{\max} - x_{\min})$ /Bitmap.Width

  όπου το .Width δηλώνει την *ιδιότητα* (*property*) του μήκους, του στοιχείου Bitmap. Εδώ θεωρούμε ότι το Bitmap είναι αντικείμενο όπως θα ισχύει στη γλώσσα προγραμματισμού που θα εφαρμόσουμε τον αλγόριθμο και ως εκ τούτου (το Bitmap) έχει ιδιότητες. Όμοια η μεταβλητή *απόσταση εικονοστοιχείων στον άξονα y* εξαρτάται από τις $y_{\min}$ και $y_{\max}$ και δηλώνει την απόσταση στον φανταστικό άξονα του μιγαδικού επιπέδου, που εκφράζουν δύο εικονοστοιχεία το ένα πάνω από το άλλο. Οπότε θα είναι:

  απόσταση εικονοστοιχείων στον άξονα y = $(y_{\max} - y_{\min})$ /Bitmap.Height

  όπου το Height εκφράζει την ιδιότητα του ύψους του αντικειμένου Bitmap.
- Η μεταβλητή *tempx* εκφράζει μία προσωρινή τιμή αποθήκευσης των δεδομένων της x_n για να μη λαμβάνονται λάθος αποτελέσματα στον υπολογισμό της y_n που ακολουθεί τον υπολογισμό της x_n στον ίδιο βρόχο. Αυτό γίνεται γιατί η τιμή της x_n αλλάζει για το τρέχον σημείο πριν υπολογιστεί η y_n. Έτσι ο υπολογισμός της x_n μέσα στο βρόχο επανάληψης που τον εκτελεί, ανάγεται σε υπολογισμό της *tempx*, γίνεται ο υπολογισμός της y_n και μετά τίθεται x_n = *tempx*.
- Η μεταβλητή *looper* εκφράζει την τρέχουσα τιμή του *n*. Ως εκ τούτου παίρνει ακέραιες τιμές.

- Οι μεταβλητές *w* και *z* εκφράζουν τη θέση του τρέχοντος εικονοστοιχείου. Μέσω αυτών μπορούμε να κάνουμε την αντιστοίχιση του στο τρέχον σημείο του μιγαδικού επιπέδου (x_n, y_n).

Σύμφωνα με τα παραπάνω, ο αλγόριθμος σε *ψευδοκώδικα* (*pseudocode*) παίρνει την παρακάτω μορφή (εξαιρούνται οι ορισμοί των μεταβλητών):

```
Re(c) = x_min ;
For (z=0; z<Bitmap.Width; z++)    {
Im(c) = y_min ;
      For (w=0; w<Bitmap.Height; w++)      {
            x_n = y_n = 0;
            looper = 0;
            While (looper < 100 Και x_n^2 + y_n^2 <4)      {
                  tempx = x_n^2 - y_n^2 + Re(c) ;
                  y_n = 2x_n y_n + Im(c) ;
                  x_n = tempx;
                  looper++;
                                                      }
            Χρωμάτισε το (z,w) από την τιμή της looper;
            y_n += απόσταση εικονοστοιχείων στον άξονα y;
                                                }
      x_n += απόσταση εικονοστοιχείων στον άξονα x;
                                    }
```

Γίνεται χρήση ενός διπλού βρόχου `For` ο οποίος σαρώνει από πάνω αριστερά το Bitmap με κατεύθυνση προς τα δεξιά και κάτω, διότι η αρίθμηση των εικονοστοιχείων σε τέτοια γραφικά αντικείμενα στις γλώσσες προγραμματισμού, γίνεται θεωρόντας (*0*,*0*) το πρώτο στοιχείο πάνω αριστερά. Τα εικονοστοιχεία στον οριζόντιο άξονα αυξάνονται με κατεύθυνση προς τα δεξιά και αυτά στον κατακόρυφο άξονα προς τα κάτω.

Για αυτό τίθεται αρχικά $\mathrm{Re}(c) = x_{\min}$ και $\mathrm{Im}(c) = y_{\min}$ και στο τέλος του βρόχου `While` το x_n αυξάνεται σύμφωνα με την *απόσταση εικονοστοιχείων στον άξονα x* και το y_n σύμφωνα με την *απόσταση εικονοστοιχείων στον άξονα y*. Το αποτέλεσμα της σάρωσης είναι να βρισκόμαστε σε κάθε επανάληψη, σε ένα εικονοστοιχείο στο οποίο έχουν αντιστοιχιστεί οι τιμές $\mathrm{Re}(c)$ και $\mathrm{Im}(c)$.

Από τις τιμές $\mathrm{Re}(c)$ και $\mathrm{Im}(c)$ που αντιστοιχίστηκαν σε κάθε εικονοστοιχείο, υπολογίζουμε το χρώμα του, ανάλογα με το αν αυτό ανήκει ή όχι στο σύνολο Mandelbrot και ανάλογα με το πόσες επαναλήψεις αυτό χρειάζεται για να διαφύγει η τροχιά του στο άπειρο. Έτσι έχουμε έναν βρόχο `While` ο οποίος επαναλαμβάνεται για όσο η μεταβλητή *looper* βρίσκεται μεταξύ ενός ορίου που θα θέσουμε και ταυτόχρονα (λογικό Και) για όσο το τετράγωνο του μέτρου της τροχιάς του z_{n+1} είναι μικρότερο του 4. Σημειώνεται ότι η μέγιστη τιμή που μπορεί να λάβει η μεταβλητή *looper* αρχικά τίθεται εδώ αυθαίρετα ίση με 100 και ότι για λόγους ταχύτητας στην εκτέλεση της εφαρμογής, δεν θέτουμε ως έλεγχο για το μέτρο της τροχιάς να είναι ο: $\sqrt{{x_{n+1}}^2 + {y_{n+1}}^2} < 2$, αλλά ο ${x_{n+1}}^2 + {y_{n+1}}^2 < 4$, για να μην επιβαρύνουμε τον υπολογιστή με τον υπολογισμό της τετραγωνικής ρίζας. Μέσα στο βρόχο `While` γίνονται οι πράξεις υπολογισμού του x_n και y_n όπως αναφέρθηκε προηγουμένως:

$tempx = {x_n}^2 - {y_n}^2 + \mathrm{Re}(c)$; $y_n = 2x_n y_n + \mathrm{Im}(c)$; $x_n = tempx$;

Ο χρωματισμός του εικονοστοιχείου από την τιμή της μεταβλητής *looper*, θα γίνει με στόχους αφενός να έχουμε μία παλέττα χρωμάτων που θα δίνει ένα ευκρινές σύνολο Mandelbrot, αφετέρου να γίνεται με μέσα που μπορεί να παρέχει η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί τα οποία θα προσφέρουν ικανοποιητική ταχύτητα στην εκτέλεση. Η υλοποίηση του χρωματισμού θα αναφερθεί εκτενέστερα παρακάτω, όπου σχολιάζεται ο πηγαίος κώδικας της εφαρμογής που γράφτηκε.

Από το πρόγραμμα που συζητείται, απαιτούνται κάποιες λειτουργίες, πέρα από το σχεδιασμό των διαφόρων συνόλων. Έτσι θα μπορεί ο χρήστης να θέτει τις τιμές $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$ για να θέτει έτσι το "παράθυρο" το οποίο θα δηλώνει την περιοχή του συνόλου που θα εμφανιστεί. Για αυτό το παράθυρο παρέχεται και η απεικόνιση της τιμής της κλίμακας της έκτασης του άξονα *x* σε σχέση με την έκταση του άξονα *y*, η οποία ανανεώνεται με κάθε αλλαγή μίας εκ των τιμών $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$. Η κλίμακα αναγράφεται ως "*scale x/y*" και προκύπτει από τη σχέση:

$scale\ x/y = \frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}}$. Λογικά αναμένουμε αυτή η τιμή να είναι ίση με τη μονάδα. Επειδή ο χρήστης του προγράμματος όμως, μπορεί να αλλάζει την τιμή των μεταβλητών $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$ ανεξάρτητα τη μία από την άλλη, η απεικόνιση της κλίμακας παρέχεται σα βοήθημα για να επιδιώκει ο χρήστης τη μονάδα στην τιμή αυτή. Η μονάδα εκφράζει την ίση έκταση του άξονα *x* με αυτήν του άξονα *y*, πράγμα που σημαίνει ότι δε θα έχουμε τανυσμό στην απεικόνιση του εκάστοτε συνόλου. Εντούτοις, θα φανεί ότι δε θα είναι δυνατόν η τιμή αυτή να είναι πάντα μονάδα ακόμη και αν ο χρήστης δεν αλλάζει άμεσα μία από τις μεταβλητές $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$. Αυτό θα συμβαίνει αφενός επειδή η κάθε τιμή από αυτές λαμβάνεται από αντιστοίχιση της τιμής σε κάποιο εικονοστοιχείο τα οποία δεν εκφράζουν γραμμική σχέση, αλλά διακριτή, και αφετέρου επειδή χρησιμοποιούνται μεταβλητές τύπου *double* (*δεκαδικός διπλής ακρίβειας*) που παρέχουν κάποιους περιορισμούς ως προς τη μέγιστη και ελάχιστη τιμή που μπορούν να εκφράσουν. Έτσι καθώς μειώνεται η κλίμακα, γίνεται εμφανές ότι θα έχουμε αποκλίσεις από τη μονάδα. Για παράδειγμα, αν η μεταβλητή για τον *x* ξεκινάει από την τιμή 1,147999999999999 και καταλήγει στην τιμή 0,049000000000001, θα είναι:

Έκταση άξονα x = 1,147999999999999 - 0,049000000000001 = 1,098999999999998.

Αν ταυτόχρονα, η μεταβλητή για τον *y* ξεκινάει από το 1,323000000000000 και καταλήγει στο 0,223999999999999, τότε θα είναι:

Έκταση άξονα y = 1,323000000000000 - 0,223999999999999 = 1,099000000000001.

Επομένως θα λαμβάνουμε: *Scale x/y* = $\frac{1,098999999999998}{1,099000000000001} = 0,999999999999996.$

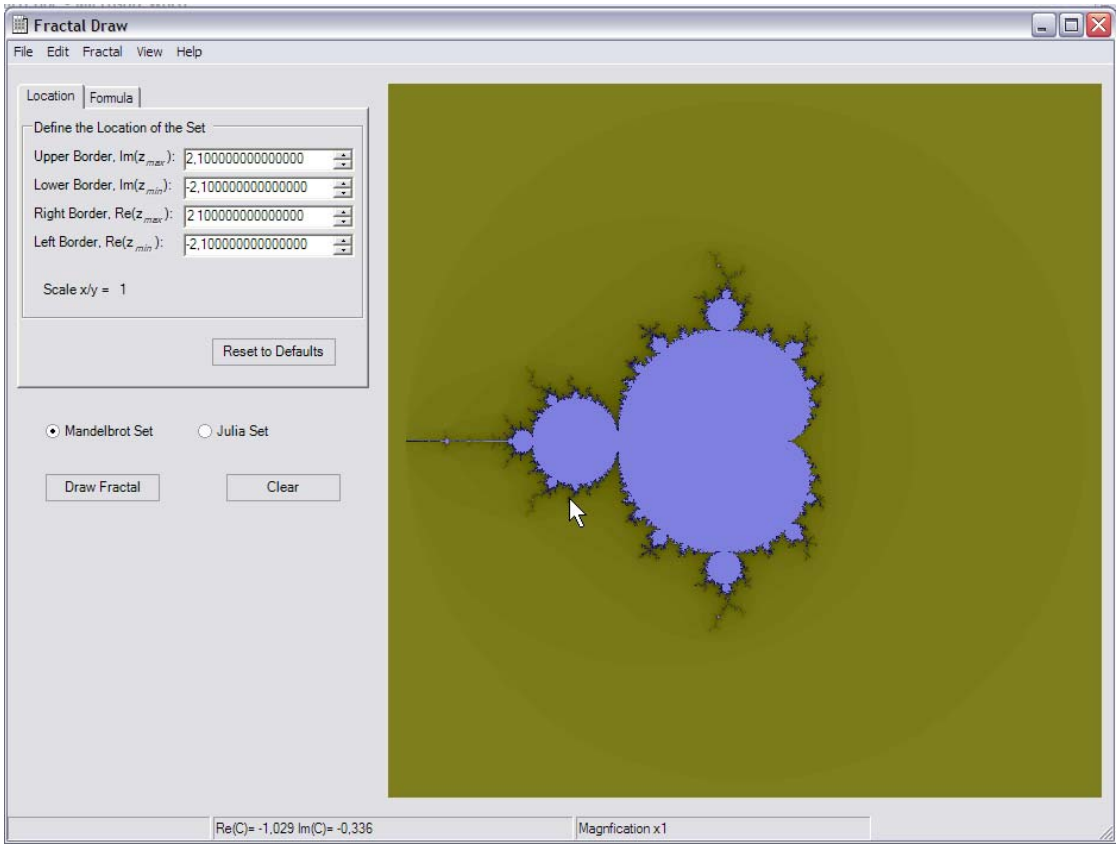
Όπως γίνεται κατανοητό, η απόκλιση αυτή από τη μονάδα και γενικά όσες ενδεχομένως εμφανιστούν δε θα είναι τόσο μεγάλες ώστε να θεωρείται ότι έχουμε σημαντική παραμόρφωση του συνόλου ή τμήματος του συνόλου που εμφανίζεται.

Πέρα από αυτό, ζητούμε το πρόγραμμα να δίνει στο χρήστη τη δυνατότητα να επιλέξει τον αριθμό των επαναλήψεων που θα εκτελεί αυτό για να σχεδιάσει το σύνολο. Πέρα από αυτό, παρέχεται και δυνατότητα στο χρήστη να ορίσει τα αρχικά σημεία $\mathrm{Re}(z_0)$ και $\mathrm{Im}(z_0)$. Αλλάζοντας τις τιμές αυτές, φαίνεται πως αλλοιώνεται η συμμετρία του μορφοκλασματικού συνόλου.

Το πρόγραμμα επίσης, θα παρέχει λειτουργία μεγέθυνσης (*zoom*) για να μπορούν εύκολα να γίνουν ορατές αυτο-όμοιες δομές που εμφανίζονται υπό μεγέθυνση στα μορφοκλασματικά σύνολα. Στη λειτουργία αυτήν, μπορεί ο χρήστης να σχεδιάσει ένα τετράγωνο με το *ποντίκι* (*mouse*) κρατώντας πατημένο το αριστερό του πλήκτρο. Σχεδιάζεται υποχρεωτικά τετράγωνο για να μην έχουμε αλλοιώσεις στην κλίμακα αλλά και για να υπάρχει πραγματική αντιστοιχία του μέρους του συνόλου που επιλέγεται με αυτό που πραγματικά θα απεικονιστεί μετά τη μεγέθυνση. Μετά την επιλογή ο χρήστης μπορεί μέσω του δεξιού πλήκτρου από το ποντίκι να επιλέξει τη λειτουργία μεγέθυνσης Κάτω δεξιά, απεικονίζεται το μέγεθος της μεγέθυνσης που προκύπτει από τη σχέση: $\frac{4,2}{x_{\max} - x_{\min}}$ γιατί 4,2 είναι η αρχική έκταση των αξόνων *x* και *y*. Υποθέτουμε ότι ο λόγος *scale x/y* είναι ακριβώς ίσος με τη μονάδα.

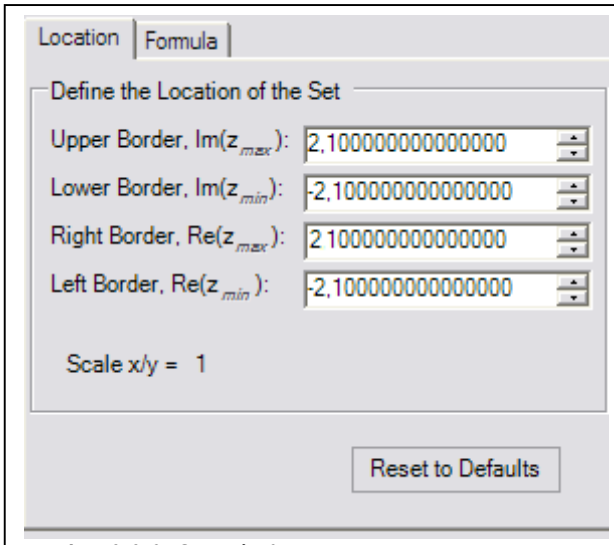
Τέλος στο πρόγραμμα απεικονίζεται το σημείο του μιγαδικού επιπέδου το οποίο διατρέχεται από τον *κέρσορα* (*cursor*) του ποντικιού και μπορεί αν έχει απεικονιστεί το σύνολο Mandelbrot, με διπλό ή δεξί κλικ να εμφανίσει το σύνολο Julia που αντιστοιχεί στο σημείο που επιθυμείται.

Στην παρακάτω εικόνα φαίνεται το παράθυρο του προγράμματος και εξηγούνται εποπτικά οι λειτουργίες που αναφέρθηκαν παραπάνω:



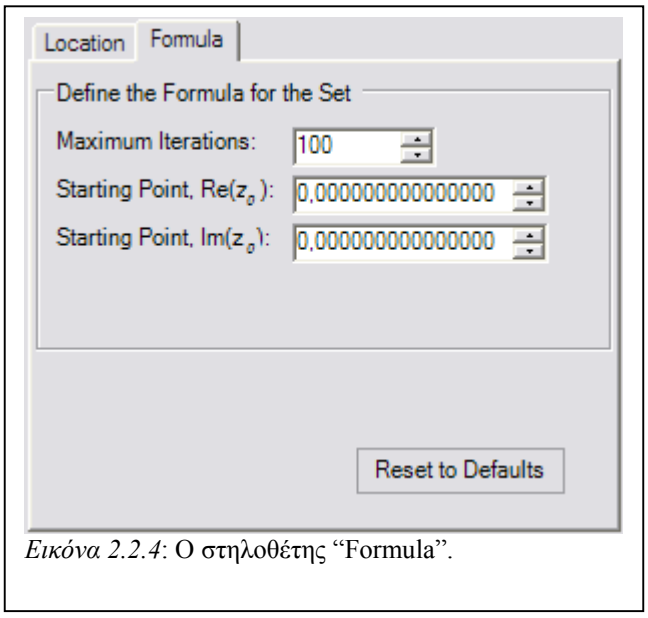
Εικόνα 2.2.2: Το κύριο παράθυρο του προγράμματος όπου έχει σχεδιαστεί το σύνολο Mandelbrot.

Στην Εικόνα 2.2.2 φαίνεται το κύριο παράθυρο του προγράμματος. Με το πλήκτρο με την αναγραφή "*Draw Fractal*" σχεδιάζεται το σύνολο που επιθυμούμε ανάλογα με το ποιο είναι επιλεγμένο στο *Radio Button* (Mandelbrot Set 'Η Julia Set). Το σύνολο θα σχεδιαστεί ανάλογα με το ποια περιοχή καθορίζεται από τις τιμές που διακρίνονται πάνω αριστερά που καθορίζουν μία περιοχή του μιγαδικού επιπέδου όπως φαίνεται στην Εικόνα 2.2.3.



Εικόνα 2.2.3: Ο στηλοθέτης με την ονομασία Location.

Αριστερά φαίνεται ο *στηλοθέτης* (*tab*) με την ονομασία Location. Από πάνω προς τα κάτω οι αριθμητικές τιμές καθορίζουν τις τιμές: $y_{\max}$, $y_{\min}$, $x_{\max}$ και $x_{\min}$ αντίστοιχα. Το scale x/y δηλώνει την έκταση του άξονα *x* ως προς αυτήν του άξονα *y* και το πλήκτρο "Reset to Defaults" επαναφέρει τις αριθμητικές τιμές στις προεπιλεγμένες που είναι αυτές που αναγράφονται εν προκειμένω.



Εικόνα 2.2.4: Ο στηλοθέτης "Formula".

Επίσης υπάρχει ο στηλοθέτης "Formula" στον οποίο μπορεί να τεθεί ο μέγιστος αριθμός επαναλήψεων (Maximum Iterations) όπως και την αρχική τιμή του *z*. Το πλήκτρο "*Reset to Defaults*" επαναφέρει τις αριθμητικές τιμές στις προεπιλεγμένες που είναι αυτές που φαίνονται στην εικόνα.

Εφόσον έχουν εξηγηθεί οι βασικές λειτουργίες του προγράμματος, ακολουθεί ο κώδικας που σχεδιάζει το σύνολο Mandelbrot όπως αυτός έχει γραφεί στο πρόγραμμα και αναλύεται.Ο κώδικας αποτελεί την εφαρμογή στη C♯, του ψευδοκώδικα που εξηγήθηκε παραπάνω.

```
public void DrawMandel()
      {
            double xmin = (double)xminUpDown.Value;
            double ymin = (double)yminUpDown.Value;
            double xmax = (double)xmaxUpDown.Value;
            double ymax = (double)ymaxUpDown.Value;
            pictureBox1.Image = original_bitmap;
            double x, y, x1, y1, tempx = 0.0;
            int looper = 0;
            int z, w = 0;
            double linearPtoX = 0.0;
            double linearPtoY = 0.0;
            linearPtoX = (xmax - xmin) / (pictureBox1.Width);
            linearPtoY = (ymax - ymin) / (pictureBox1.Height);
            x = xmin;
            for(z = 0; z < original_bitmap.Width; z++)
            {
                  y = ymin;
                  for(w = 0; w < original_bitmap.Height; w++)
                  {
                        x1 = (double)ReZzeroUpDown.Value;
                        y1 = (double)ImZzeroUpDown.Value;
                        looper = 0;
                        while(looper < (int)iterationsUpDown.Value &&
(x1 * x1) + (y1 * y1) < 4)
                        {
                              tempx =(x1 * x1) - (y1 * y1) + x;
                              y1 = 2 * x1 * y1 + y;
                              x1 = tempx;
                              looper++;
                        }
      double percent = (looper / (double)(iterationsUpDown.Value));
                        int val = ((int)(percent * 255));
original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(
128-val)),(Math.Abs(128-val)),Math.Abs((32-val));
                        y += linearPtoY;

                  }
                  x += linearPtoX;
            }

            pictureBox1.BackgroundImage = original_bitmap;

      }
```

Στην πρώτη γραμμή του κώδικα έχουμε τη δήλωση `public void DrawMandel()`. Είναι μία δήλωση συνάρτησης τύπου void που σημαίνει ότι δεν επιστρέφει κάποια τιμή. Επίσης έχει προσβασιμότητα τύπου public γεγονός που δηλώνει ότι μπορεί να χρησιμοποιηθεί από αντικείμενα κάποιας άλλης κλάσης και όχι αποκλειστικά αυτής που παράγει τη συνάρτηση `DrawMandel`.

Στις γραμμές 3 ως και 6 ορίζονται οι μεταβλητές `xmin`, `xmax`, `ymin` και `ymax` που αντιστοιχούν προφανώς στις ψευδομεταβλητές: $x_{\min}$, $y_{\min}$, $x_{\max}$, και $y_{\max}$ αντίστοιχα. Με τον ορισμό τους, τους ανατίθενται οι τιμές των αντίστοιχων αριθμητικών που βλέπει ο χρήστης, αυτών της εικόνας 2.2.3.

Ακολουθεί ο ορισμός των υπόλοιπων μεταβλητών: `x, y, x1, y1, tempx, linearPtoX, linearPtoY` τύπου double και `looper, z, w` τύπου *int* (*ακέραιοι*). Η `linearPtoX` τίθεται ίση με την έκταση του άξονα *x* που απεικονίζεται προς τον αριθμό των εικονοστοιχείων που έχει το στοιχείο Bitmap. Εδώ έχουμε τον αριθμό των εικονοστοιχείων να ορίζεται από το μήκος και ύψος του στοιχείου `pictureBox1` του προγράμματος. Επειδή στο πρόγραμμα χρησιμοποιούνται πάνω από ένα στοιχείο τύπου Bitmap, έχουμε αυτό το στοιχείο για να μπορούν να εμφανίζονται σε υπέρθεση τα Bitmap αυτά. Όμοια με το `linearPtoX` ορίζεται το `linearPtoY` ίσο με την έκταση του άξονα *y* προς το ύψος του στοιχείου `pictureBox1`.

Στη γραμμή 15 τίθεται η μεταβλητή `x` ίση με τη `xmin` και στη γραμμή 16 ξεκινάει η σάρωση των οριζοντίων γραμμών εικονοστοιχείων. Αυτό γίνεται με τον πρώτο βρόχο `for` που για την τιμή της ακέραιας μεταβλητής `z` και για όσο είναι αυτή μέσα στα πλαίσια του `pictureBox1` εκτελεί τον κώδικα που περικλείει. Τίθεται η μεταβλητή `y` να είναι ίση με την `ymin` και ξεκινάει η εκτέλεση του δεύτερου βρόχου `for` που σαρώνει τα κατακόρυφα εικονοστοιχεία σύμφωνα με τη μεταβλητή `w`.

Ακόλουθα, τίθενται στις μεταβλητές `x1` και `y1` οι τιμές τους σύμφωνα με αυτές που αναγράφονται στο στηλοθέτη "Formula". Οι μεταβλητές αυτές εκφράζουν το πραγματικό και φανταστικό μέρος της τιμής του z_0 αντίστοιχα. Επίσης τίθεται η αρχική τιμή της μεταβλητής `looper` που εκφράζει την τιμή του αριθμού των επαναλήψεων (n).

Στη γραμμή 24 ξεκινάει η εκτέλεση ενός βρόχου `while`. Ο βρόχος αυτός θα εκτελείται για όσο η μεταβλητή `looper` είναι μικρότερη από τον μέγιστο αριθμό επαναλήψεων που έχει καθορίσει ο χρήστης στο στηλοθέτη "Formula" (`looper < (int) iterationsUpDown.Value`) και ταυτόχρονα για όσο το τετράγωνο του μέτρου του z_n είναι μικρότερο από το 4 (`(x1 * x1) + (y1 * y1) < 4`), γεγονός που εξασφαλίζει το ότι η τροχιά δεν έχει διαφύγει στο άπειρο. Οπότε σε κάθε κύκλο εκτέλεσης του βρόχου `while`, έχουμε την ανάθεση τιμών από το σώμα του βρόχου: Τίθεται η τιμή της `tempx` ίση με το $x_n^2 - y_n^2 + \mathrm{Re}(c)$ το οποίο σε κώδικα γράφεται `tempx =(x1 * x1) - (y1 * y1) + x;`. Έπειτα τίθεται η τιμή της `y1` (`y1 = 2 * x1 * y1 + y`). Εφόσον έχει ολοκληρωθεί ο υπολογισμός του προκείμενου z_n αναθέτουμε την τιμή `tempx` στη μεταβλητή `x1` για να υπολογιστεί σωστά αν το τετράγωνο του μέτρου είναι μικρότερο από το 4 που αποτελεί συνθήκη επανεκτέλεσης του βρόχου `while` στον οποίο αναφερόμαστε και τέλος αυξάνεται η τιμή της `looper`.

Εφόσον έχουν γίνει όλοι οι κατάλληλοι υπολογισμοί, απομένει να γραφεί ο κώδικας για τον χρωματισμό κάθε εικονοστοιχείου. Στη 32η γραμμή κώδικα ορίζεται η μεταβλητή `percent` η οποία εκφράζει το ποσοστό του αριθμού των επαναλήψεων που έγιναν για το τρέχον σημείο του μιγαδικού επιπέδου, ως προς το μέγιστο αριθμό επαναλήψεων.

Από την τιμή percent θα γίνει χρωματισμός των εικονοστοιχείων. Στο στοιχείο Bitmap μπορεί να χρησιμοποιθεί η κωδικοποίηση RGB (Red Green Blue) για την ανάθεση χρώματων. Έτσι ορίζεται για κάθε εικονοστοιχείο το τι ποσοστό κόκκινου, πράσινου και γαλάζιου χρώματος θα έχει, το οποίο συνθετικά θα παράγει το πραγματικό χρώμα του. Το ποσοστό εκφράζεται από έναν ακέραιο αριθμό, με το ελάχιστο για κάθε χρώμα να είναι το 0 (0% του αντίστοιχου χρώματος) και το μέγιστο να είναι το 255 (100% του αντίστοιχου χρώματος). Η μεταβλητή `percent` είναι τύπου double και από τη σχέση που την ορίζει θα βρίσκεται αποκλειστικά στο διάστημα [0,1]. Κανονικοποιούμε την τιμή αυτή για να βρίσκεται στην κλίμακα [0,255] και μετατρέπουμε τον αριθμό σε ακέραιο, τιμή που εκφράζεται από την ακέραια μεταβλητή `val` (`int val = ((int)(percent * 255));`). Τώρα μπορεί να γίνει ανάθεση χρώματος σε κάθε εικονοστοιχείο από την τιμή της `val`. Το τρέχον εικονοστοιχείο χρωματίζεται με τον τύπο των γραμμών κώδικα 34 και 35 που έχει προκύψει για καλύτερη ευκρίνεια.

Κατόπιν η *y* αυξάνεται κατά τη `linearPtoY` και κλείνει ο πρώτος βρόχος `for`. Αμέσως μετά η *x* αυξάνεται κατά τη `linearPtoX` και κλείνει και ο δεύτερος βρόχος `for`. Με αυτές τις αυξήσεις οι τιμές των *x* και *y* έχουν πάρει τις κατάλληλες τιμές για να γίνει η επαναληπτική διαδικασία στο επόμενο εικονοστοιχείο.

Τα εικονοστοιχεία που λαμβάνουν χρώμα στη συνάρτηση `DrawMandel` είναι αυτά του αντικειμένου τύπου Bitmap με όνομα `original_bitmap`. Στο τέλος της συνάρτησης (γραμμή 42), τίθεται ως "φόντο" του στοιχείου `pictureBox1` το Bitmap αυτό και έτσι λαμβάνεται η απεικόνιση του συνόλου.

2.3 Η εφαρμογή του αλγορίθμου απεικόνισης του συνόλου Julia

Σε αυτή την παράγραφο θα αναλυθεί το ππως δημιουργείται ο πηγαίος κώδικας που εμφανίζει τα διάφορα σύνολα Julia. Στο σύνολο Mandelbrot αυτό που άλλαζε σύμφωνα με τις συντεταγμένες στο μιγαδικό επίπεδο, ήταν η πραγματική και η φανταστική τιμή της σταθεράς *c*. Στο σύνολο Julia αυτό που αλλάζει είναι το πραγματικό και φανταστικό μέρος της z_0. Έτσι έχουμε να υπολογίσουμε το σύνολο αυτό που δίνεται από την Εξίσωση 1.4.2: $z_{n+1} = z_n^{\,2} + c$ για τα οποία το μέτρο του z_{n+1} δεν τείνει στο άπειρο, με το z_0 να μεταβάλλεται ανάλογα με το σε ποιο σημείο του μιγαδικού επιπέδου γίνεται η επαναληπτική διαδικασία. Η ανάλυση της επαναληπτικής διαδικασίας θα δώσει ίδια αποτελέσματα με αυτά που έχουμε για το σύνολο Mandelbrot. Εφόσον $z_n = x_n + iy_n$ και $c = a + ib$ υπολογίζουμε τα x_{n+1} και y_{n+1} οπότε έχουμε: $x_{n+1} = x_n^{\,2} - y_n^{\,2} + a$ και $y_{n+1} = 2x_n y_n + b$.

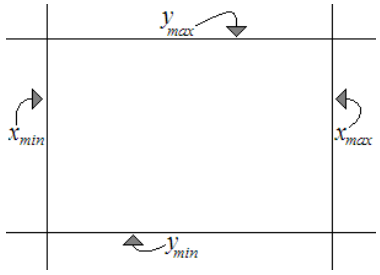
Η απεικόνιση θα γίνει μέσω του προγράμματος των οποίων οι βασικές λειτουργίες αναφέρθηκαν στην προηγούμενη παράγραφο, οπότε θα έχουμε πάλι τα ίδια στοιχεία που θα θέσουν τον προγραμματισμό μέσα σε κάποια πλαίσια που θα επιτρέψουν τη λειτουργία του προγράμματος και θα κάνουν εύκολη την ανάλυση των λειτουργιών που θα επιτελούν. Έτσι υπάρχει πάλι το στοιχείο Bitmap, στο οποίο ουσιαστικά θα γίνει η επεξεργασία, το στοιχείο pictureBox1 που θα εμφανίζει αυτό το Bitmap και οι μεταβλητές που θα χρησιμεύσουν στην ανάθεση τιμών χρήσιμων για την απεικόνιση του συνόλου.

Οι ψευδομεταβλητές που θα χρησιμοποιηθούν για την απεικόνιση του συνόλου Julia θα είναι οι παρακάτω:

$\mathrm{Re}(c)$, $\mathrm{Im}(c)$, $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$, x_n, y_n, *απόσταση εικονοστοιχείων στον άξονα x*, *απόσταση εικονοστοιχείων στον άξονα y*, *tempx* (δεκαδικοί διπλής ακρίβειας)
looper, *z*, *w* (ακέραιοι)

- Οι μεταβλητές $\mathrm{Re}(c)$ και $\mathrm{Im}(c)$ είναι το πραγματικό και το φανταστικό μέρος αντίστοιχα, του τρέχοντος *c*. Αυτές οι μεταβλητές αποτελούν τον *σπόρο* (*seed*) για το κάθε σύνολο Julia που θέλουμε να απεικονιστεί μιας και παραμένουν σταθερές σε κάθε επαναληπτική διαδικασία.
- Η μεταβλητή x_n είναι η τρέχουσα τιμή του πραγματικού μέρους της τροχιάς του *z* (x_n) και εκφράζει την τιμή $\mathrm{Re}(z_{n+1}) = x_{n+1}$ αλλά και την τιμή $\mathrm{Re}(z_n) = x_n$. Αυτή στο πρόγραμμα θα υπολογίζεται σύμφωνα με την εξίσωση (1.4.3):
  $x_n = x_n^{\,2} - y_n^{\,2} + \mathrm{Re}(c)$. Για το αρχικό x_n τίθεται η τιμή σύμφωνα με τη θέση του εικονοστοιχείου στην οποία αντιστοιχεί στο x_n στο μιγαδικό επίπεδο. Έπειτα κατά τη διάρκεια της επαναληπτικής διαδικασίας λαμβάνει τιμές ανάλογα με την τρέχουσα τιμή της ίδιας μεταβλητής μέσω της εξίσωσης (1.4.3).
- Όμοια η μεταβλητή y_n, εκφράζει το φανταστικό μέρος της τροχιάς του *z*, παίρνει τιμές σύμφωνα με την εξίσωση (1.4.4): $y_n = 2x_n y_n + \mathrm{Im}(c)$ και η αρχική της τιμή τίθεται ανάλογα με τη θέση του εικονοστοιχείου στο οποίο γίνεται η επαναληπτική διαδικασία.

- Η $x_{\min}$ είναι η ελάχιστη τιμή του πραγματικού μέρους του *z* που εμφανίζεται. Δηλαδή η ελάχιστη τιμή του οριζόντιου άξονα του μιγαδικού επιπέδου, όπως απεικονίζεται από το πρόγραμμα μέσω του στοιχείου Bitmap. Αντίστοιχα, η $x_{\max}$ είναι η μέγιστη τιμή του πραγματικού μέρους του *z* που εμφανίζεται, η $y_{\min}$ η ελάχιστη τιμή του φανταστικού μέρους του *z* και τέλος η $y_{\max}$ η μέγιστη τιμή του φανταστικού μέρους του *z*. Τα $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$ σχηματίζουν ένα τετράγωνο "παράθυρο" στο εσωτερικό του οποίου σχεδιάζεται το σύνολο Julia (Εικόνες 2.2.1 και 2.2.5).



Εικόνα 2.2.5: Το "παράθυρο" που σχηματίζεται από τις τιμές των μεταβλητών $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$.

- Η μεταβλητή *απόσταση εικονοστοιχείων στον άξονα x* εξαρτάται από τις $x_{\min}$ και $x_{\max}$ και δηλώνει την απόσταση στον πραγματικό άξονα του μιγαδικού επιπέδου που εκφράζουν δύο διπλανά μεταξύ τους εικονοστοιχεία. Έτσι θα έχουμε:

  απόσταση εικονοστοιχείων στον άξονα x = $(x_{\max} - x_{\min})$ /Bitmap.Width

  όπου το .Width δηλώνει την ιδιότητα του μήκους, του αντικειμένου Bitmap. Όμοια η μεταβλητή *απόσταση εικονοστοιχείων στον άξονα y* εξαρτάται από τις $y_{\min}$ και $y_{\max}$ και δηλώνει την απόσταση στον φανταστικό άξονα του μιγαδικού επιπέδου, που εκφράζουν δύο εικονοστοιχεία το ένα πάνω από το άλλο. Οπότε θα είναι:

  απόσταση εικονοστοιχείων στον άξονα y = $(y_{\max} - y_{\min})$ /Bitmap.Height

  όπου το Height εκφράζει την ιδιότητα του ύψους του αντικειμένου Bitmap.

- Η μεταβλητή *tempx* όπως και η αντίστοιχη στον κώδικα για το σύνολο Mandelbrot, εκφράζει μία προσωρινή τιμή αποθήκευσης των δεδομένων της x_n για να μη λαμβάνονται λάθος αποτελέσματα στον υπολογισμό της y_n που ακολουθεί τον υπολογισμό της x_n στον ίδιο βρόχο. Αυτό γίνεται γιατί η τιμή της x_n αλλάζει για το τρέχον σημείο πριν υπολογιστεί η y_n. Έτσι ο υπολογισμός της x_n μέσα στο βρόχο επανάληψης που τον εκτελεί, ανάγεται σε υπολογισμό της *tempx*, γίνεται ο υπολογισμός της y_n και μετά τίθεται $x_n = tempx$.
- Η μεταβλητή *looper* εκφράζει την τρέχουσα τιμή του *n*. Ως εκ τούτου παίρνει ακέραιες τιμές.
- Οι μεταβλητές *w* και *z* εκφράζουν τη θέση του τρέχοντος εικονοστοιχείου. Μέσω αυτών μπορούμε να κάνουμε την αντιστοίχιση του στο τρέχον σημείο του μιγαδικού επιπέδου (x_n, y_n).

Έτσι λαμβάνουμε τον κάτωθι ψευδοκώδικα για τον υπολογισμό και την απεικόνιση του συνόλου Julia:

```
Re(c) = από seed;
Im(c) = από seed;
For (z=0; z<Bitmap.Width; z++)     {
      For (w=0; w<Bitmap.Height; w++)      {
            x_n = x_min + z* απόσταση εικονοστοιχείων στον άξονα x;
            y_n = y_min + w* απόσταση εικονοστοιχείων στον άξονα y;
            looper = 0;
            While (looper < 100 Και x_n^2 + y_n^2 <4)      {
                  tempx = x_n^2 - y_n^2 + Re(c);
                  y_n = 2x_n y_n + Im(c);
                  x_n = tempx;
                  looper++;
                                                      }
            Χρωμάτισε το (z,w) από την τιμή της looper;
                                          }
                                    }
```

Όμοια με τον αλγόριθμο για το σύνολο Mandelbrot, γίνεται χρήση ενός διπλού βρόχου `For` ο οποίος σαρώνει από πάνω αριστερά το Bitmap με κατεύθυνση προς τα δεξιά και κάτω. Το ποια λαμβάνεται ως η τιμή του seed θα εξηγηθεί στην υλοποίηση του αλγόριθμου στη C♯.

Στο σύνολο Julia, η τιμή των μεταβλητών στο μιγαδικό επίπεδο που αντιστοιχεί σε κάθε εικονοστοιχείο αντιστοιχίζεται με τις σχέσεις:

$$x_n = x_{\min} + z * \textit{απόσταση εικονοστοιχείων στον άξονα } x;$$
$$y_n = y_{\min} + w * \textit{απόσταση εικονοστοιχείων στον άξονα } y;$$

Όπου γίνεται η αντιστοίχιση σύμφωνα με τη θέση του τρέχοντος εικονοστοιχείου με τη βοήθεια των μεταβλητών *z* και *w* (που δίνουν τη θέση του τρέχοντος εικονοστοιχείου), πολλαπλασιαζόμενες με τις μεταβλητές *απόσταση εικονοστοιχείων στον άξονα x* και *απόσταση εικονοστοιχείων στον άξονα y* αντίστοιχα. Προσθέτουμε την ελάχιστη τιμή, $x_{\min}$, στην x_n και $y_{\min}$, στην y_n γιατι η αρίθμηση δεν πρέπει να ξεκινάει από το μηδέν, εφόσον αυτή εξαρτάται από το "παράθυρο" του μιγαδικού επιπέδου στο οποίο γίνεται η απεικόνιση του συνόλου. Έτσι έχει γίνει η αντιστοίχιση της θέσης του εικονοστοιχείου στο μιγαδικό επίπεδο με το διπλό βρόχο `For`.

Έπειτα με βρόχο `While` υπολογίζονται το πραγματικό και φανταστικό μέρος για κάθε τιμή του z_{n+1} σε κάθε επανάληψη. Ο βρόχος `While` εκτελείται για όσο η μεταβλητή *looper* βρίσκεται μεταξύ ενός ορίου που τίθεται και ταυτόχρονα για όσο το τετράγωνο του μέτρου της τροχιάς του z_{n+1} είναι μικρότερο του 4. Σημειώνεται πάλι, ότι υπολογίζεται το τετράγωνο του μέτρου για να αποφευχθεί ο υπολογισμός της τετραγωνικής ρίζας που ενδεχομένως καθυστερήσει τους υπολογισμούς. Μέσα στο βρόχο `While` γίνονται οι πράξεις υπολογισμού του x_n και y_n όπως αναφέρθηκε προηγουμένως:

$$tempx = x_n^2 - y_n^2 + \mathrm{Re}(c)\ ; \quad y_n = 2x_n y_n + \mathrm{Im}(c)\ ; \quad x_n = tempx;$$

Οι στηλοθέτες “Location” και “Formula” στο σύνολο Julia εξυπηρετούν ακριβώς τους ίδιους σκοπούς που εξυπηρετούσαν στο σύνολο Mandelbrot. Στον μεν “Location” καθορίζεται το “παράθυρο” σχεδιασμού του συνόλου, στον δε “Formula” ο αριθμός των επαναλήψεων. Οι αρχικές συνθήκες του δεύτερου στηλοθέτη, δεν παίζουν κάποιον ρόλο στο σύνολο Julia, αφού το z_0 στο σύνολο αυτό, έχει τονιστεί ότι εξαρτάται από τη θέση του σημείου στο μιγαδικό επίπεδο.

Ακολουθεί ο πηγαίος κώδικας του προγράμματος στη C♯ και η ανάλυση του.

```
public void DrawJulia()
{
      Bitmap original_bitmap = new
Bitmap(pictureBox1.Width,pictureBox1.Height);
      pictureBox1.Image = original_bitmap;
      double tempx, xmin, xmax, ymin, ymax = 0.0;
      double x1 = (double)ReZzeroUpDown.Value;
      double y1 = (double)ImZzeroUpDown.Value;
      int looper, z, w = 0;
      double linearPtoX, linearPtoY = 0.0;
      xmin = (double)xminUpDown.Value;
      ymin = (double)yminUpDown.Value;
      xmax = (double)xmaxUpDown.Value;
      ymax = (double)ymaxUpDown.Value;
      linearPtoX = (xmax - xmin) / original_bitmap.Width;
      linearPtoY = (ymax - ymin) / original_bitmap.Height;
      for(z = 0; z < original_bitmap.Width; z++)
      {
            for(w = 0; w < original_bitmap.Height; w++)
            {
            x1 = (double)xminUpDown.Value + linearPtoX*z;
            y1 = (double)ymaxUpDown.Value - linearPtoY*w;
            looper = 0;
                  while(looper < iterationsUpDown.Value && (x1 * x1)
+ (y1 * y1) < 4)
                  {
                  tempx = (x1 * x1) - (y1 * y1) + thisCRe;
                  y1 = 2 * x1 * y1 + thisCIm;
                  x1 = tempx;
                  looper++;
                  }
            double percent = (looper /
(double)(iterationsUpDown.Value));
            int val = ((int)(percent * 255));
            original_bitmap.SetPixel(z,w,
System.Drawing.Color.FromArgb((Math.Abs(128-val)),(Math.Abs(128-
val)),Math.Abs((32-val))));
            }
      }
pictureBox1.BackgroundImage = original_bitmap;

}
```

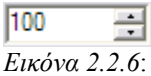
Στην πρώτη γραμμή του πηγαίου κώδικα, ορίζεται η συνάρτηση `DrawJulia` τύπου void (δεν επιστρέφει κάποια τιμή) και προσβασιμότητας public (προσβάσιμη σε κάθε μέλος του προγράμματος που ζητά την κλήση της). Στην τρίτη γραμμή ορίζεται το στοιχείο `original_bitmap` τύπου Bitmap που θα χρησιμεύσει στην απεικόνιση του συνόλου. Σαν εικόνα που θα εμφανίζει το στοιχείο `pictureBox1` τίθεται αμέσως αυτό το Bitmap που ορίστηκε. Αυτό γίνεται για να καταλάβει τη θέση του συνόλου Mandelbrot που ενδεχομένως έχει προηγουμένως εμφανιστεί και για να μην υπάρχει υπέρθεση των δύο διαφορετικών Bitmap. Κατόπιν θα τεθεί και ως "φόντο" του `pictureBox1` το ίδιο Bitmap για τον ίδιο λόγο αλλά και για να επιτραπεί αργότερα η λειτουργία μεγέθυνσης που από πλευράς γραφικών σε συνάρτηση με τη διαδραστικότητα με τον χρήστη, βασίζεται στην υπέρθεση.

Στην 6η σειρά ορίζονται οι μεταβλητές `tempx, xmin, xmax, ymin` και `ymax` τύπου double (δεκαδικού διπλής ακρίβειας).

Στις σειρές 7 και 8 λαμβάνεται το seed για την πραγματική και φανταστική τιμή της σταθεράς *c* σύμφωνα με την οποία θα σχεδιαστεί το σύνολο Julia. Αυτό γίνεται με τις εκφράσεις:

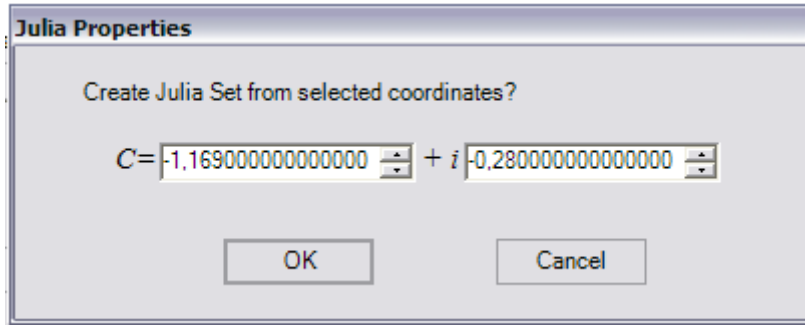
```
double x1 = (double)ReZzeroUpDown.Value;
double y1 = (double)ImZzeroUpDown.Value;
```

Ορίζεται αρχικά η μεταβλητή `x1` τύπου double και λαμβάνει την τιμή του `ReZzeroUpDown`. Αυτό το στοιχείο αποτελεί έκφανση αντικειμένου της κλάσης `numericUpDown` της C♯. Ένα παράδειγμα αντικειμένου της κλάσης φαίνεται στην Εικόνα 2.2.6. Ένα αντικείμενο αυτή της κλάσης περιέχει μία αριθμητική τιμή που μπορεί να αυξηθεί ή να μειωθεί με κλικ του ποντικιού πάνω σε ένα από τα αντίστοιχα κουμπιά που αυτή έχει. Ο χρήστης επίσης μπορεί να θέσει μία τιμή αν η ιδιότητα της ReadOnly είναι ίση με true.



Εικόνα 2.2.6: Αντικείμενο της κλάσης `numericUpDown`.

Η εμφάνιση του αριθμού που εκφράζει το αντικείμενο γίνεται όπως επιθυμεί ο προγραμματιστής θέτωντας όσες θέσεις για δεκαδικά ψηφία του επιτρέπει η μεταβλητή τύπου decimal [2]. Εν προκειμένω, χρησιμοποιούνται 15 θέσεις για δεκαδικά ψηφία και η μεταβλητή τύπου decimal μετατρέπεται για χρήση από το πρόγραμμα σε τύπου double. Η μετατροπή αυτή δηλώνεται με τη λέξη (double) σε παρένθεση πριν την τιμή που περιγράφεται από μεταβλητή τύπου decimal. Στον πηγαίνο κώδικα η τιμή αυτή είναι η ιδιότητα `Value` του αντικειμένου `ReZzeroUpDown`. Οπότε η μεταβλητή `x1` παίρνει την τιμή της από την τιμή του `ReZzeroUpDown` και η μεταβλητή `y1` από την τιμή του `ImZzeroUpDown.Value`. Τα αντικείμενα αυτά βρίσκονται σε άλλη *φόρμα παραθύρου* (*windows form*) από το κεντρικό του προγράμματος. Η φόρμα αυτή εμφανίζεται με δύο τρόπους: αν έχει απεικονιστεί σύνολο Mandelbrot με διπλό κλικ στο `pictureBox1` ή με δεξί κλικ σε αυτό και επιλογή "*Draw Julia* Set *from this Point*". Αυτή φαίνεται στην Εικόνα 2.2.7. Το πραγματικό μέρος είναι το αριστερό `numericUpDown` στην Εικόνα 2.2.7 και το φανταστικό το δεξί. Οι μεταβλητές που είναι γραμμένες είναι αυτές του σημείου στο οποίο έγινε διπλό ή δεξί κλικ. Σημειώνεται ότι αν δε γίνει δεξί ή διπλό κλικ και ο χρήστης επιλέξει από την κύρια φόρμα του προγράμματος να σχεδιάσει το σύνολο Julia, χωρίς να το έχει κάνει προηγουμένως, τότε θα σχεδιαστεί αυτό με τις *προκαθορισμένες* (*default*) τιμές μιας αριθμητικής μεταβλητής. Επομένως θα σχεδιαστεί το σύνολο Julia για $c = (0,0)$.



Εικόνα 2.2.7: Η φόρμα από την οποία λαμβάνεται το seed για τη δημιουργία του συνόλου Julia.

Στις σειρές 9 και 10 του κώδικα ορίζονται οι ακέραιες μεταβλητές `looper`, `z` και `w` και οι μεταβλητές δεκαδικών διπλής ακρίβειας `linearPtoX` και `linearPtoY`. Έπειτα οι μεταβλητές `xmin`, `xmax`, `ymin` και `ymax` λαμβάνουν τις τιμές τους από τις αντίστοιχες τιμές του στηλοθέτη "Location" και ακολουθούν οι σχέσεις:

```
linearPtoX = (xmax - xmin) / original_bitmap.Width;
linearPtoY = (ymax - ymin) / original_bitmap.Height;
```

Από αυτές δίνεται η τιμή που εκφράζει τις αποστάσεις που δηλώνουν δύο συνεχόμενα εικονοστοιχεία στους άξονες *x* και *y* αντίστοιχα.

Στη 17η σειρά κώδικα ξεκινάει ο πρώτος βρόχος `for` που σαρώνει τα οριζόντια εικονοστοιχεία και στη 19η ο δεύτερος που σαρώνει τα κατακόρυφα. Το ζεύγος μεταβλητών (`z`,`w`) δίνει τη συντεταγμένη του τρέχοντος εικονοστοιχείου. Η αντιστοίχιση εικονοστοιχείων στο μιγαδικό επίπεδο γίνεται με τις σχέσεις:

```
x1 = (double)xminUpDown.Value + linearPtoX*z;
y1 = (double)ymaxUpDown.Value - linearPtoY*w;
```

Στη γραμμή 24 ξεκινάει ο βρόχος `while` που εκτελείται για όσο η μεταβλητή `looper` είναι μικρότερη της προκαθορισμής μέγιστης τιμής επαναλήψεων στο στηλοθέτη "Formula" και για όσο το τετράγωνο του μέτρου του z_{n+1} είναι μικρότερο του 4. Υπάρχει και εδώ η μεταβλητή προσωρινής αποθήκευσης της τιμής της `x1`, με το όνομα `tempx`.

Τέλος στις σειρές 32 έως και 37 γίνεται ο χρωματισμός του κάθε εικονοστοιχείου με τον ίδιο ακριβώς τρόπο που γίνεται για το σύνολο Mandelbrot.

2.4 Άλλες λειτουργίες του προγράμματος

2.4.1 Η εφαρμογή της επιλογής των συντεταγμένων του κέρσορα για την απεικόνιση του συνόλου Julia

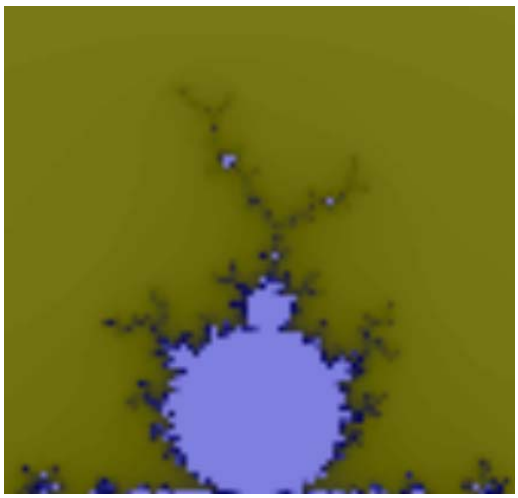
Όπως αναφέρθηκε σε προηγούμενη παράγραφο, το σύνολο Mandelbrot αρχικά σχεδιάστηκε ως "χάρτης" για όλα τα σύνολα Julia, εφόσον σε κάθε σημείο του αντιστοιχίζεται και ένα διαφορετικό σύνολο Julia. Αυτή η αντιστοίχιση μπορεί να εκφραστεί μέσω του προγράμματος που δημιουργήθηκε. Ο χρήστης του προγράμματος έχει τη δυνατότητα να επιλέξει με το ποντίκι του υπολογιστή ένα σημείο του συνόλου Mandelbrot και από εκεί να σχεδιάσει το αντίστοιχο σύνολο Julia.

Για αυτόν το σκοπό γίνεται χρήση μερικών μεταβλητών, του *γεγονότος* (*event*) που συμβαίνει με την κίνηση του ποντικιού με το όνομα MouseMove και μία μετατροπή των συντεταγμένων από συντεταγμένες εικονοστοιχεία σε συντεταγμένες του μιγαδικού επιπέδου. Σε όλες τις γλώσσες αντικειμενοστρεφούς προγραμματισμού υποστηρίζονται *γεγονότα* που εκφράζουν αλληλεπίδραση του χρήστη με το πρόγραμμα. Ένα τέτοιο είναι της κίνησης του ποντικιού (MouseMove). Το γεγονός αυτό έχει ιδιότητες δύο εκ των οποίων είναι οι συντεταγμένες της θέσης του κέρσορα σε εικονοστοιχεία ως προς τη φόρμα στην οποία το γεγονός απευθύνεται. Με χρήση λοιπόν του MouseMove μπορούμε να έχουμε την τιμή των συντεταγμένων. Αυτές πολλαπλασιαζόμενες με την παράσταση $j_{\min} + \frac{\left(\left|j_{\min}\right| + \left|j_{\max}\right|\right)}{bitmap.width \mid bitmap.height}$, όπου j μπορεί να είναι x ή y. Αν $j=x$, επιλέγεται το *bitmap.width* για τη διαίρεση, αλλιώς το *bitmap.height*. Τέλος, οι συντεταγμένες του μιγαδικού επιπέδου που προκύπτουν, τίθενται ως seed για το σύνολο Julia και έτσι σχεδιάζεται αυτό στην επιθυμητή

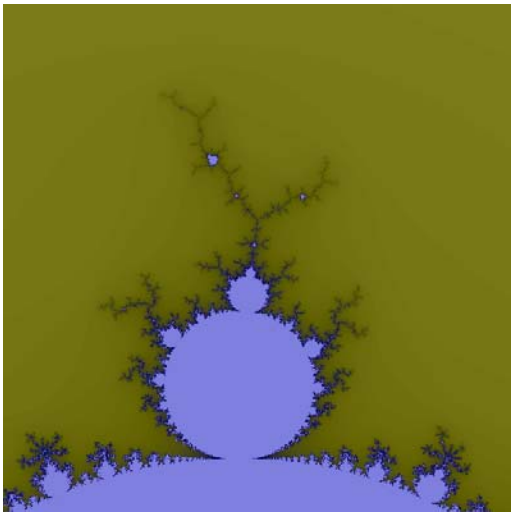
περιοχή.

2.4.2 Η εφαρμογή του αλγορίθμου μεγέθυνσης στο σύνολο Mandelbrot

Σε αυτή την παράγραφο θα αναλυθεί το πως δημιουργείται ο πηγαίος κώδικας που πραγματοποιεί τη λειτουργία της μεγέθυνσης στο σύνολο Mandelbrot. Αν χρησιμοποιηθεί κάποιος από τους υπάρχοντες αλγόριθμους μεγέθυνσης της εικόνας για το στοιχείο bitmap που απεικονίζει το σύνολο, θα χαθεί η λεπτομέρεια, καθώς το σύνολο δεν υπολογίζεται ξανά, αλλά τα εικονοστοιχεία της περιοχής στην οποία θα γίνει μεγέθυνση πολλαπλασιάζονται με έναν *παράγοντα μεγέθυνσης* (*zoom factor*) μεγαλύτερο της μονάδας, όπου θα έχει εφαρμοστεί ενδεχομένως κάποια μέθοδος εξομάλυνσης των γραφικών, που προϋπάρχει στη γλώσσα προγραμματισμού. Σαν αποτέλεσμα θα βλέπαμε μια εικόνα σαν τη *2.4.1*. Για να λυθεί αυτό το πρόβλημα, πρέπει να επιλεγεί η περιοχή στην οποία ζητείται να γίνει μεγέθυνση από τον χρήστη του προγράμματος και να υπολογίζεται ξανά το σύνολο στην περιοχή αυτή. Έτσι εφόσον ο υπολογισμός του συνόλου γίνεται "εικονοστοιχείο προς εικονοστοιχείο",



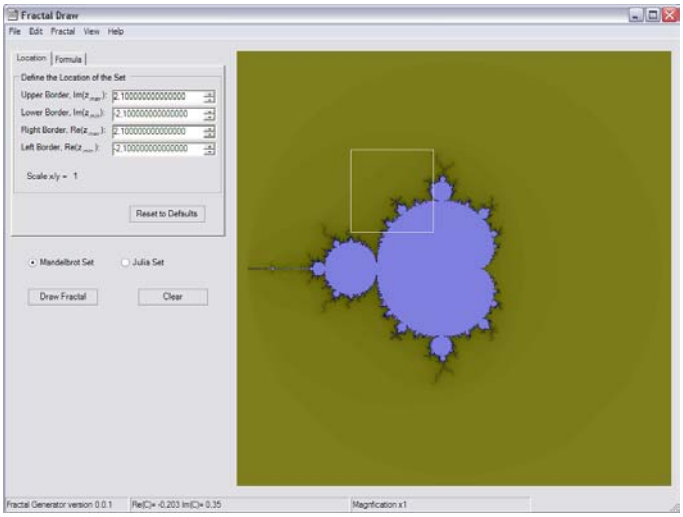
Εικόνα 2.4.1: Τμήμα του συνόλου Mandelbrot με μεγέθυνση από τον υπάρχοντα αλγόριθμο μεγέθυνσης του λειτουργικού συστήματος.



Εικόνα 2.4.2: Το ίδιο τμήμα του συνόλου Mandelbrot με την *Εικόνα 2.4.1*, με μεγέθυνση από τον αλγόριθμο που σχεδιάστηκε.

εξασφαλίζεται η απεικόνιση της λεπτομέρειας που έχουν τα μορφοκλασματικά σύνολα εις βάρος ενδεχομένως του χρόνου υπολογισμού, ο οποίος εντούτοις, στους σύγχρονους υπολογιστές για μεγεθύνσεις της της τάξης του δισεκατομμυρίου, δεν ξεπερνάει τα 7 sec για 500 επαναλήψεις. Στην *Εικόνα 2.4.2* δίνεται προς σύγκριση η ίδια περιοχή με αυτήν της *Εικόνας 2.4.1*, με μεγέθυνση από τον αλγόριθμο που υλοποιήθηκε στην παρούσα εργασία.

Ήδη στην ανάλυση της δημιουργίας του προγράμματος, έχουν αναφερθεί κάποια αντικείμενα προγραμματισμού, τα οποία μπορούν μέσω των ιδιοτήτων τους, να παρέχουν τα βασικά στοιχεία για την υλοποίηση του αλγορίθμου μεγέθυνσης. Το στοιχείο bitmap, θα χρησιμοποιηθεί πάλι για την απεικόνιση του μεγενθυμένου συνόλου. Το γεγονός MouseMove που αναφέρεται στην προηγούμενη παράγραφο, θα χρησιμοποιηθεί για τις ιδιότητες του: MouseMove.X και MouseMove.Y που δίνουν τη θέση του κέρσορα του ποντικιού του υπολογιστή στο στοιχείο bitmap, αλλά σε συντεταγμένες εικονοστοιχείων. Επίσης θα χρησιμοποιηθούν κάποιες προσωρινές συντεταγμένες αποθήκευσης που ορίζονται ως αντικείμενα στις οποίες θα τοποθετούνται οι τιμές των συντεταγμένων του κέρσορα ανά πάσα κίνηση του και τέλος η συνάρτηση DrawRect που απεικονίζει σε κάποιο γραφικό στοιχείο (εν προκειμένω στο bitmap) ένα παραλληλόγραμμο.



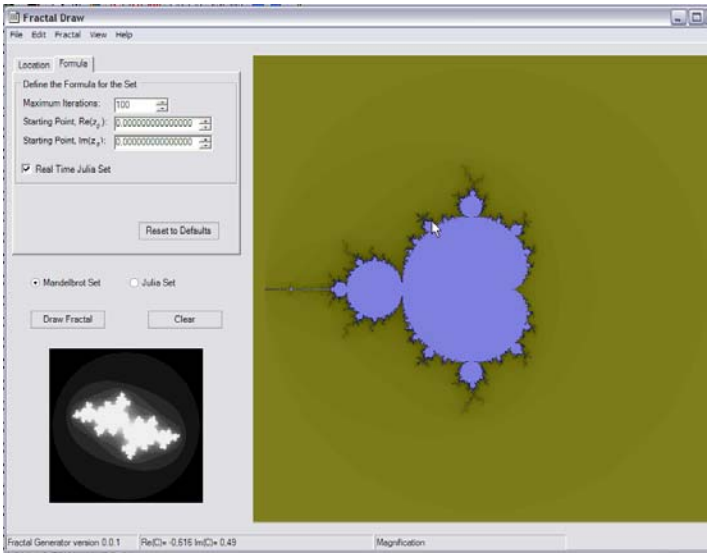
Εικόνα 2.4.3: Το παράθυρο του προγράματος όπου σχεδιάζεται από τον χρήστη τετράγωνο με σκοπό να γίνει μεγέθυνση σε αυτή την περιοχή.

Αρχικά με την κίνηση του ποντικιού του υπολογιστή λαμβάνονται οι τιμές των συντεταγμένων του. Αυτές αποθηκευόνται σε μια προσωρινή

μεταβλητή και μετατρέπονται μέσω ενός τύπου στις αντίστοιχες συντεταγμένες του μιγαδικού επιπέδου όπως στην **Παράγραφο 2.4.1**. Με τη συνθήκη ταυτόχρονης κίνησης του ποντικιού και ταυτόχρονα πατημένου του αριστερού πλήκτρου του, ο χρήστης του προγράμματος μπορεί να σχεδιάσει ένα τετράγωνο. Η συνάρτηση DrawRect, σχεδιάζει παραλληλόγραμμο, εντούτοις τίθεται περιορισμός ώστε οι κάθετες πλευρές να είναι ίσες μεταξύ τους και έτσι ανάγεται σε τετράγωνο. Η περιοχή του τετραγώνου δηλώνει την περιοχή όπου θα εκτελεστεί ο αλγόριθμος μεγέθυνσης. Οι συντεταγμένες της περιοχής λαμβάνονται από τις συντεταγμένες των πλευρών του τετραγώνου. Έτσι η κάθε πλευρά του τετραγώνου εκφράζει το που θα μεταφερθεί κάθε μεταβλητή από τις $x_{\min}$, $x_{\max}$, $y_{\min}$ και $y_{\max}$. Αφού γίνει η αντίστοιχη επιλογή για τη μεγέθυνση, μετατρέπονται οι συντεταγμένες του, από συντεταγμένες εικονοστοιχείων, σε συντεταγμένες του μιγαδικού επιπέδου από τον ακόλουθο τύπο (για το x): $x_{\min} = \frac{x_{\min-\tau\varepsilon\tau\rho\alpha\gamma\acute{\omega}\nu\sigma\upsilon}}{bitmap.width} \cdot X$, όπου $x_{\min-\tau\varepsilon\tau\rho\alpha\gamma\acute{\omega}\nu\sigma\upsilon}$ η τετμημένη της αριστερής πλευράς του τετραγώνου, bitmap.width το μήκος σε εικονοστοιχεία του στοιχείου bitmap και X το συνολικό μήκος της περιοχής που απεικονίζεται. Όμοια υπολογίζονται και οι άλλες πλευρές του παραθύρου όπου θα γίνει η απεικόνιση της μεγέθυνσης. Αφού υπολογιστούν από το πρόγραμμα όλες αυτές οι τιμές γίνεται ξανά ο υπολογισμός του συνόλου Mandelbrot για τη νέα περιοχή που ορίζεται. Έτσι δεν υπάρχει απώλεια ποιότητας και μπορεί να παρατηρηθεί η αυτο-ομοιότητα που εμφανίζει το σύνολο. Με αυτήν την υλοποίηση του αλγορίθμου, η μεγέθυνση μπορεί να φτάσει στην τάξη του 10^{12}, γεγονός που απαιτεί χρήση ακεραίου τύπου Int64 για να την εκφράσει. Ο περιορισμός της μεγέθυνσης σε αυτή την κλίμακα οφείλεται στη χρήση μεταβλητών τύπου double στον υπολογισμό των μεταβλητών του παραθύρου που εκφράζει την περιοχή απεικόνισης του συνόλου. Με χρήση μεταβλητών decimal η μεγέθυνση θα μπορούσε να φτάσει και μεγαλύτερες τιμές.

2.4.3 Η εφαρμογή της απεικόνισης του συνόλου Julia σε πραγματικό χρόνο

Σύμφωνα με όσα αναφέρθηκαν στην **παράγραφο 2.4.1** με μία μικρή βελτιστοποίηση του αλγορίθμου του συνόλου Julia, μπορεί να σχεδιάζεται το αντίστοιχο σύνολο σε πραγματικό χρόνο καθώς ο κέρσορας βρίσκεται στο αντίστοιχο σημείο του συνόλου Mandelbrot. Η βελτιστοιποίηση έγκειται ***α)*** στο να μην υπολογίζονται περαιτέρω τιμές εφόσον το σύνολο θα απεικονίζεται σε μία συγκεκριμένη περιοχή με συγκεκριμένο μέγεθος, οπότε είναι σταθερές και οι αποστάσεις που εκφράζουν μεταξύ τους διαδοχικά εικονοστοιχεία, ***β)*** στο μικρό αριθμό επαναλήψεων που θα χρησιμοποιείται για την απεικόνιση του συνόλου Julia πραγματικού χρόνου και ***γ)*** στη συμμετρία κάθε συνόλου Julia ως προς τον άγονα *x*. Έτσι λαμβάνονται οι τιμές των συνταταγμένων του κέρσορα όπως στην **παράγραφο 2.4.1**, και γίνεται η ίδια μετατροπή για να προκύψουν οι συντεταγμένες στο μιγαδικό επίπεδο. Με αυτές ως seed, σχεδιάζεται αυτή τη φορά το μισό σύνολο Julia, για θετικές τιμές του *y* και το άλλο μισό αναστρέφεται και αντιγράφεται κάτω από το υπολογισμένο. Έτσι μπορεί να γίνει εμφανής στο χρήστη του προγράμματος η αντιστοίχιση κάθε σημείου του συνόλου Mandelbrot με ένα σύνολο Julia.



Εικόνα 2.4.4: Το παράθυρο του προγράμματος με ενεργοποιημένη την επιλογή Julia set πραγματικού χρόνου.

Αναφορές:

[1]: Roach P., (2000), Phonetics and Phonology: 3rd Edition, Cambridge University Press, Cambridge, 91.

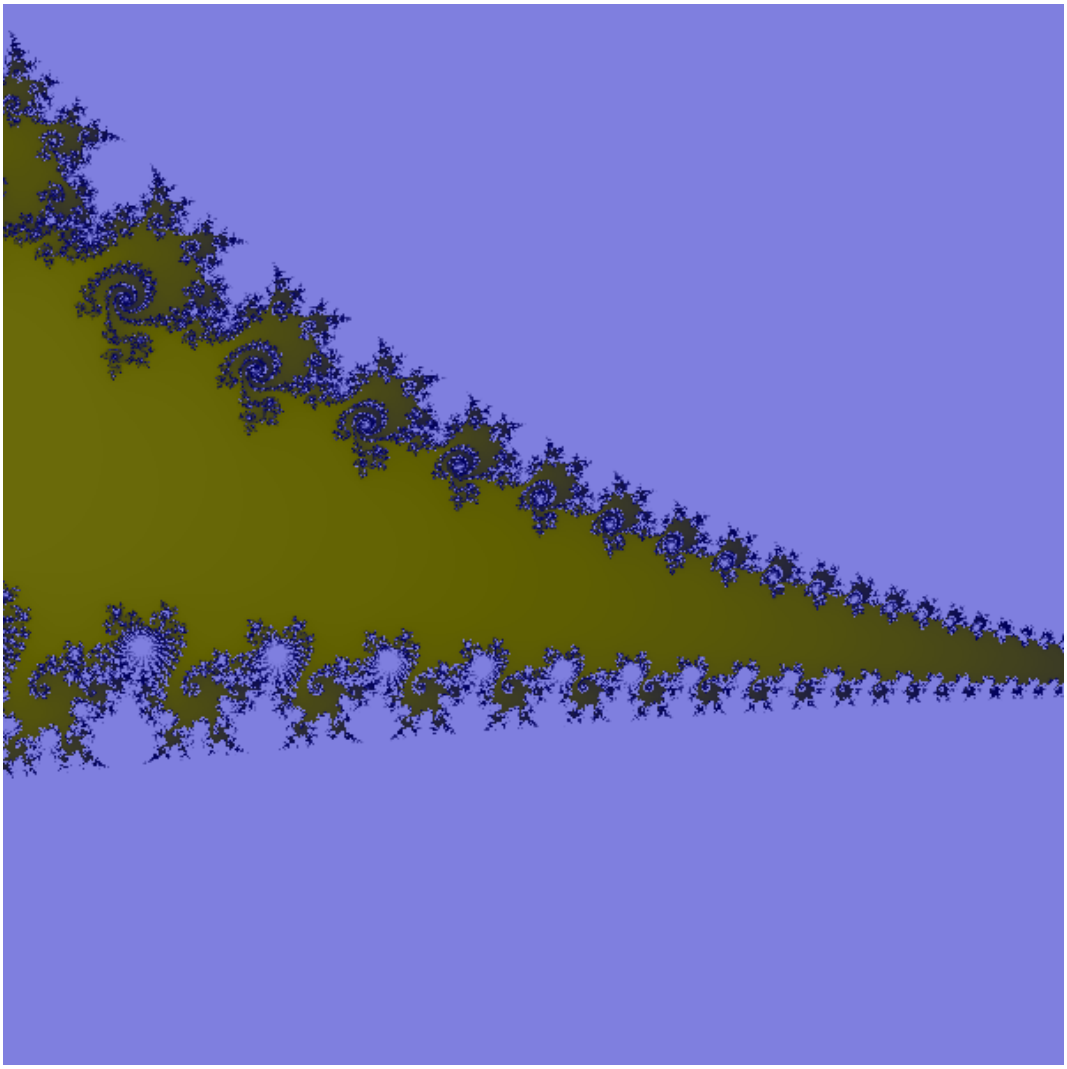
[2]: Liberty J., (2003), Programming C#, 3rd Edition, O'Reilly & Associates, Inc., Sebastopol CA, ISBN: 0-596-00489-3, 312.

Βιβλιογραφία:

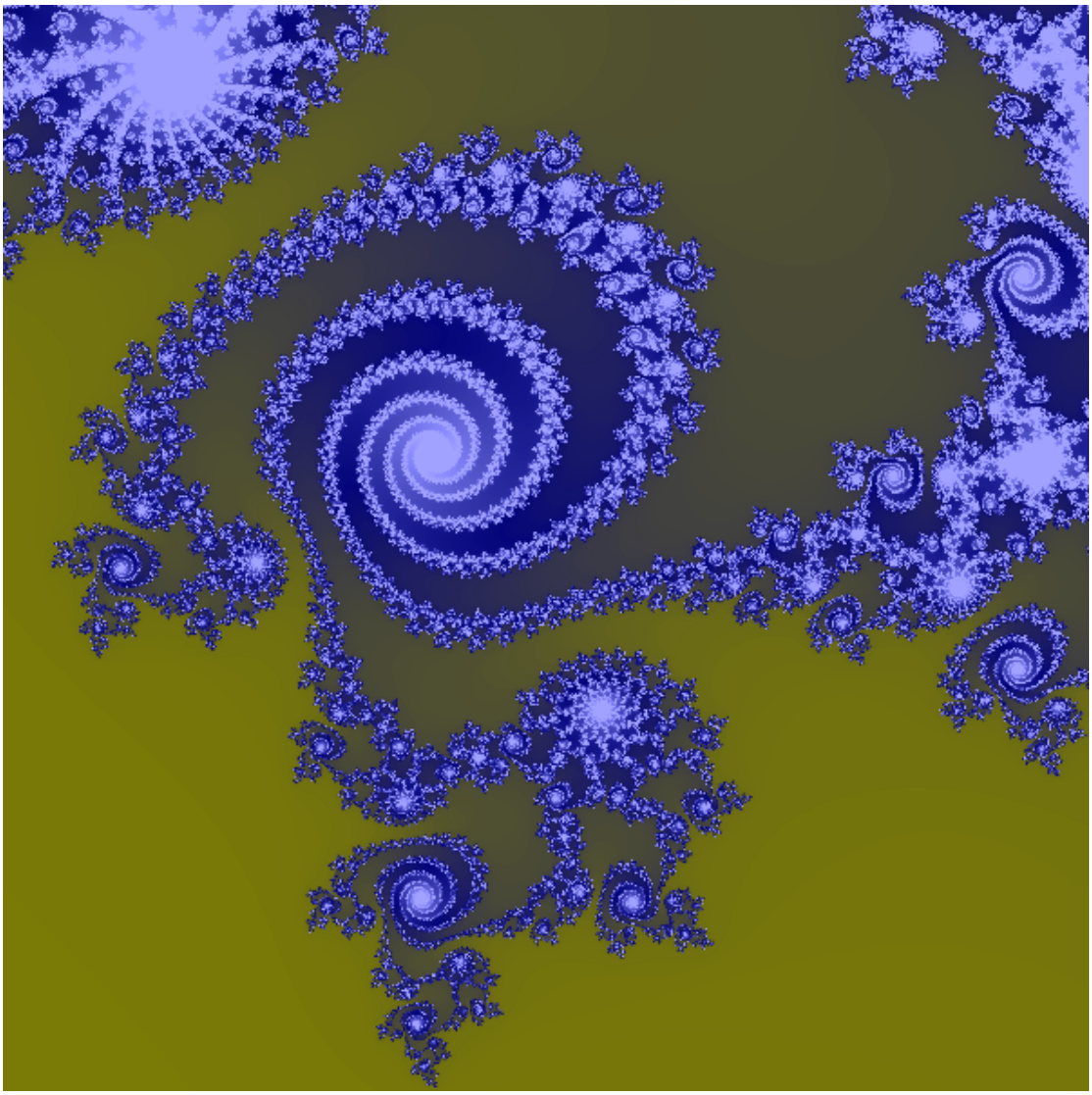
1. n.a. (2002). msdn® training. 2609A: Introduction to C♯ Programming with Microsoft® .NET. Microsoft Corporation.

3.1 Εικόνες μορφοκλασματικών συνόλων

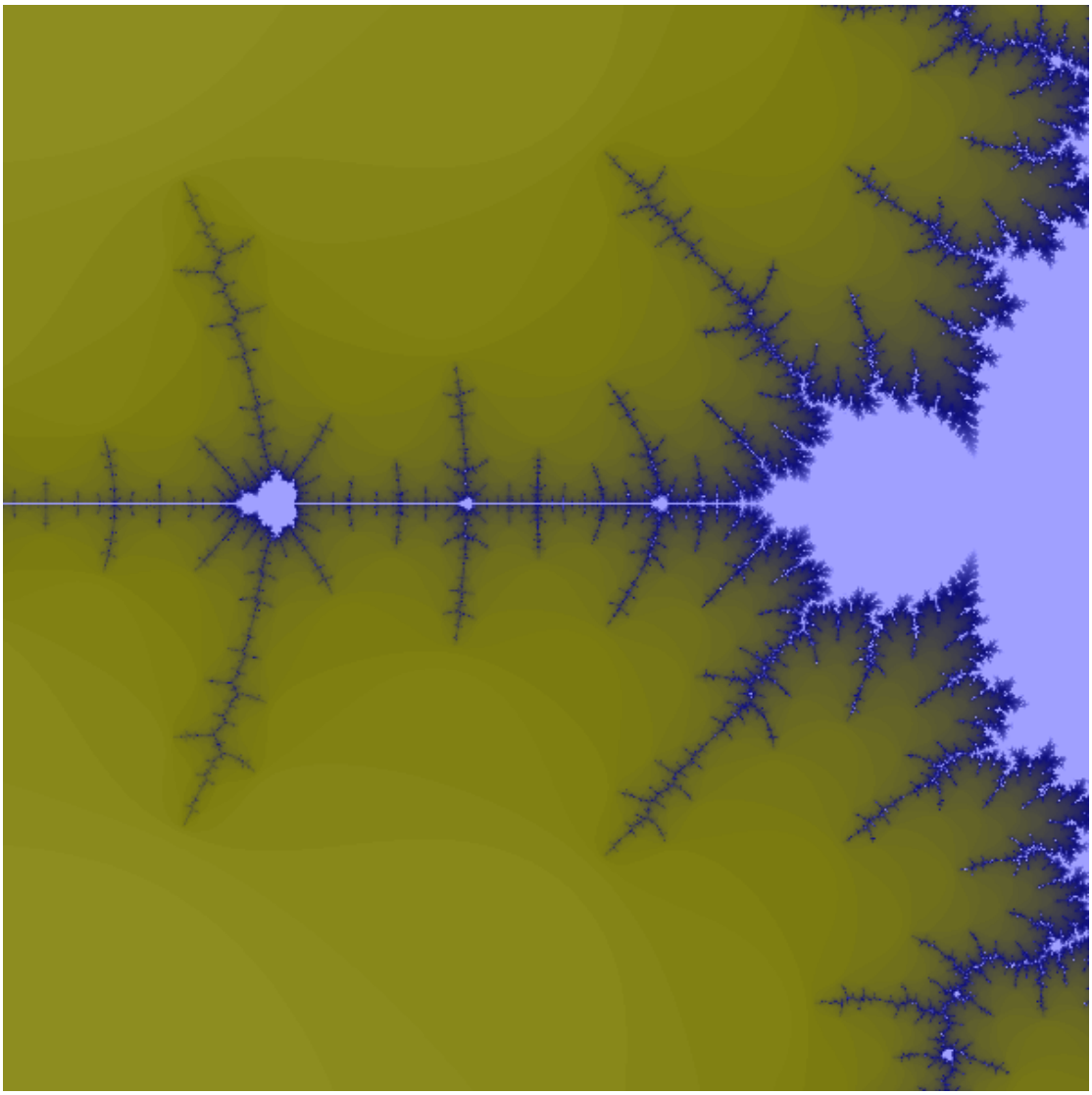
Ακολουθούν εικόνες που προήλθαν από το πρόγραμμα που δημιουργήθηκε.



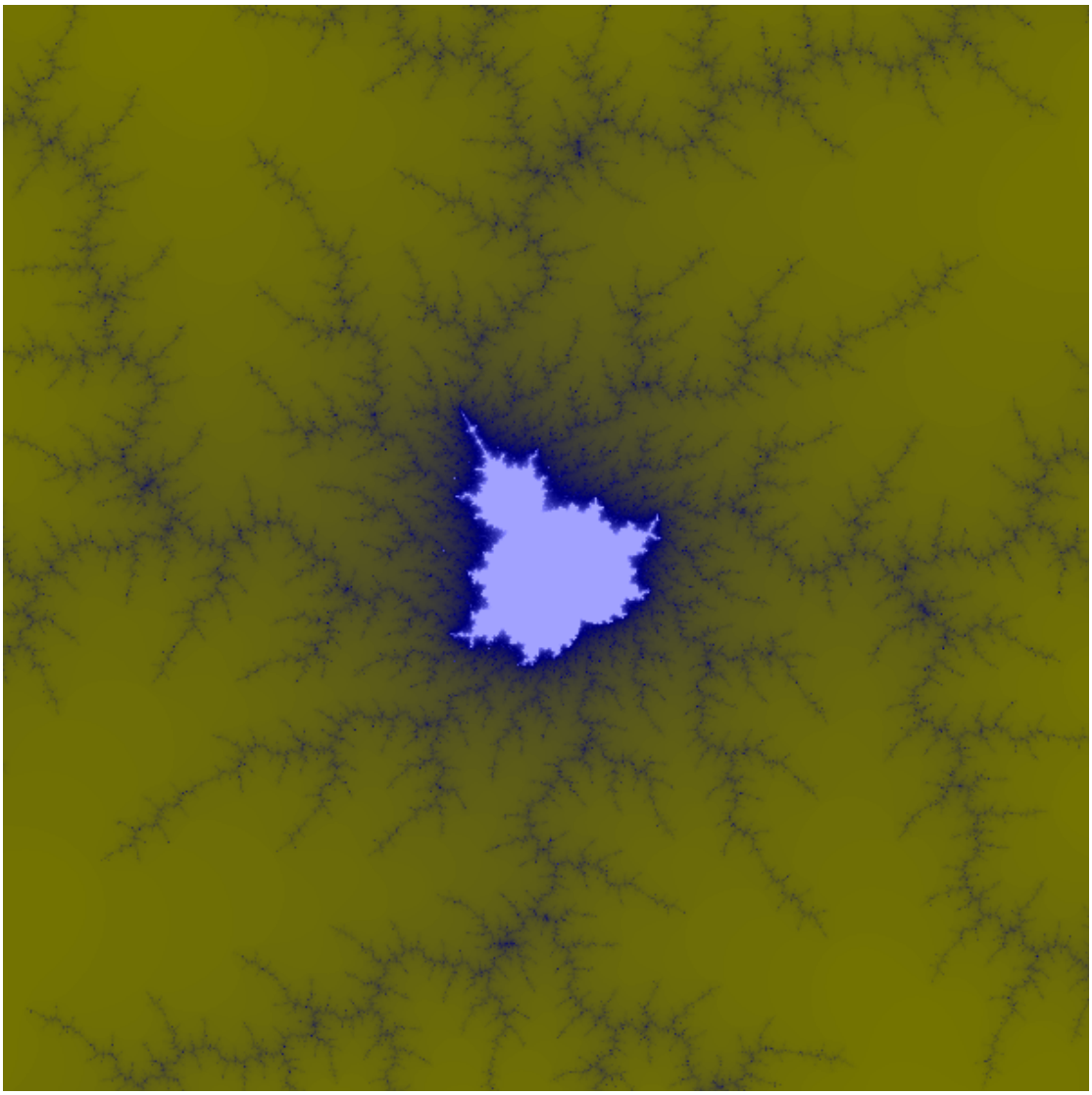
Εικόνα 3.1.1: Σύνολο Mandelbrot. 300 επαναλήψεις στην περιοχή:
$y_{\max}$ = -0,633080000000000 $y_{\min}$ = -0,679490000000000,
$x_{\max}$ = -0,147070000000000, $x_{\min}$ = -0,193480000000000.



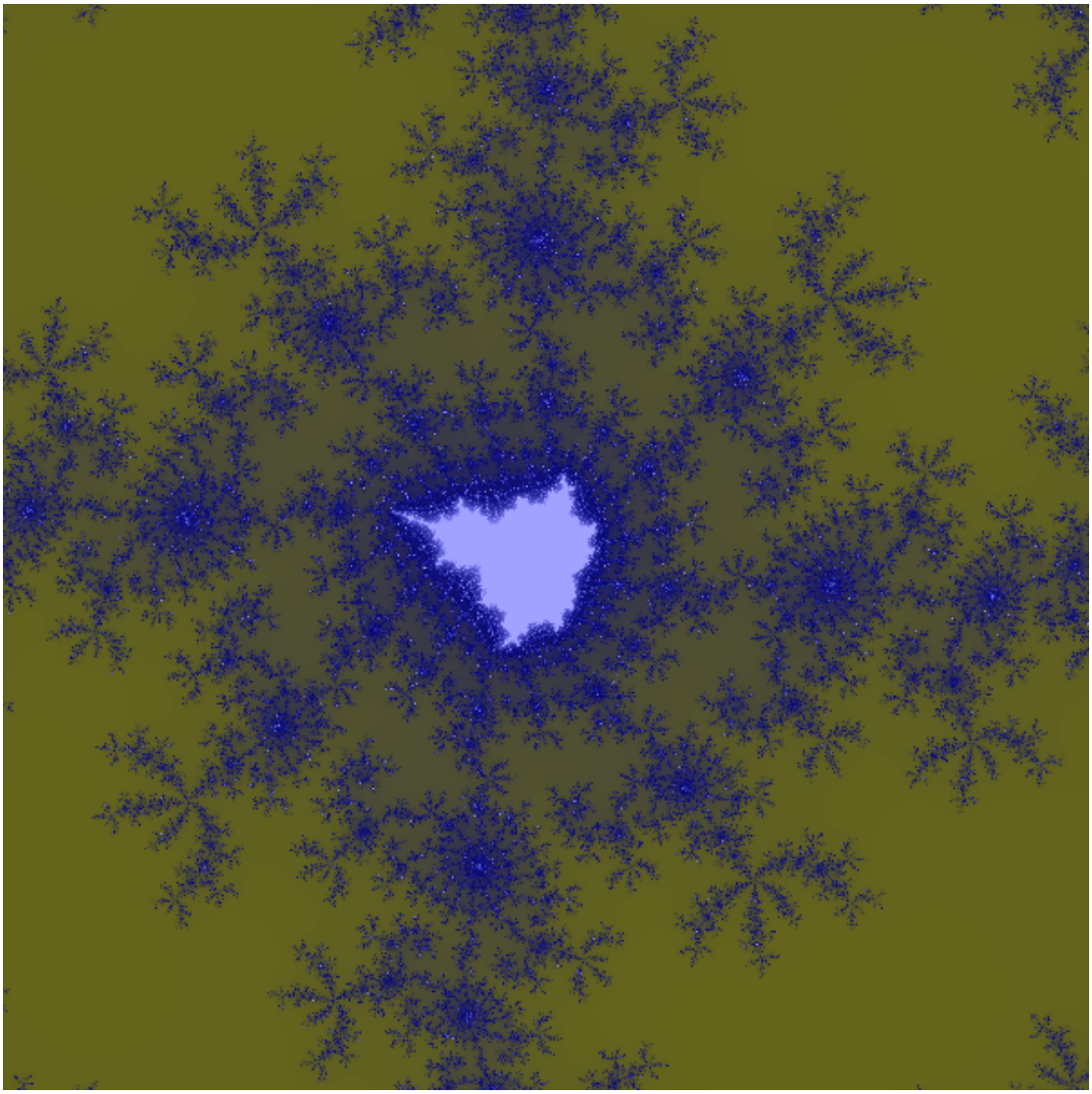
Εικόνα 3.1.2: Σύνολο Mandelbrot. 300 επαναλήψεις στην περιοχή:
$y_{\max}$ = -0,656130300000000 $y_{\min}$ = -0,659069600000000,
$x_{\max}$ = -0,168263900000000, $x_{\min}$ = -0,171203200000000.



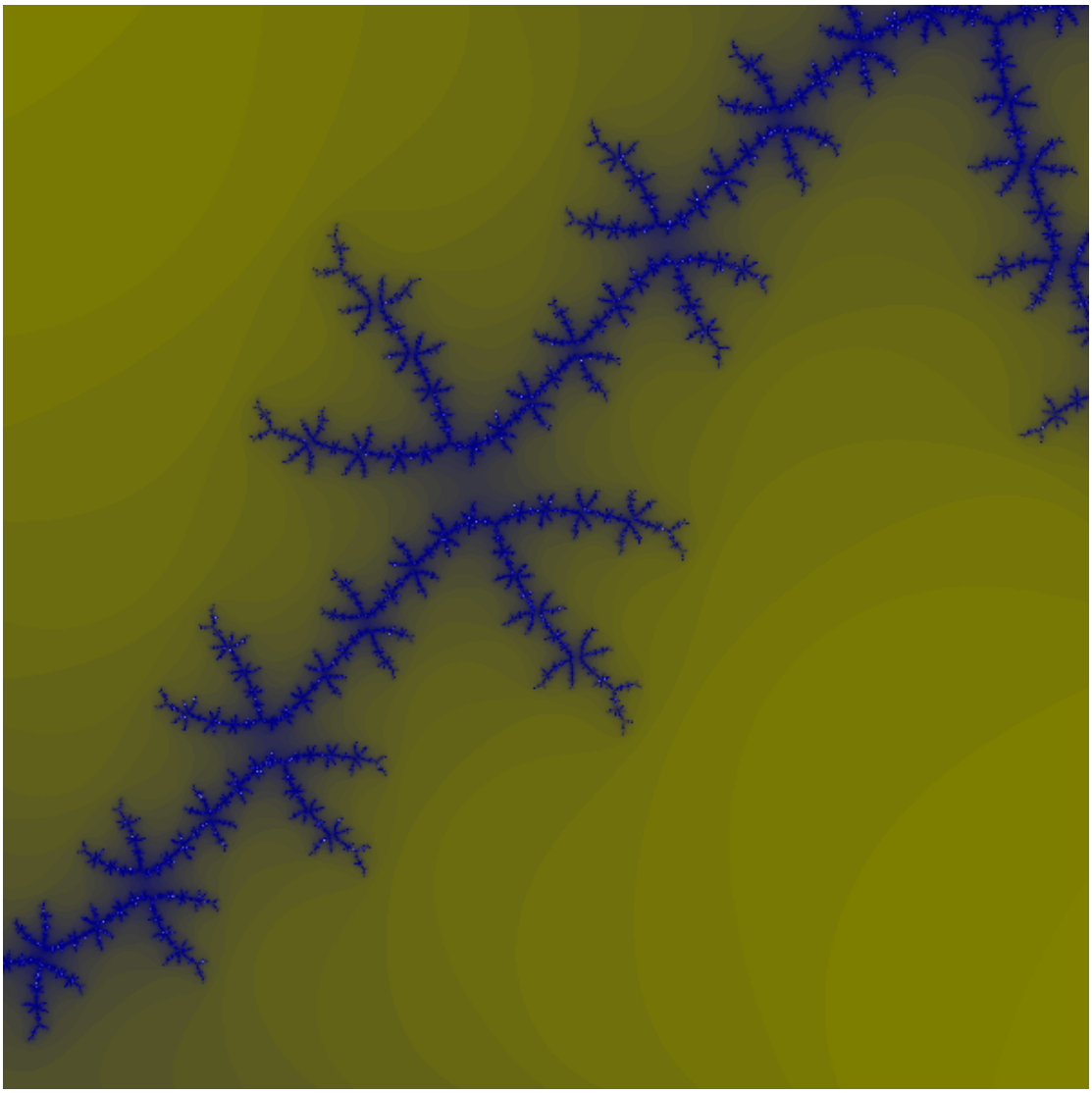
Εικόνα 3.1.3: Σύνολο Mandelbrot. 100 επαναλήψεις στην περιοχή:
y_{max} = 0,091000000000001, y_{min} = -0,076999999999999,
x_{max} = -1,351000000000001 , x_{min} = -1,519000000000001.



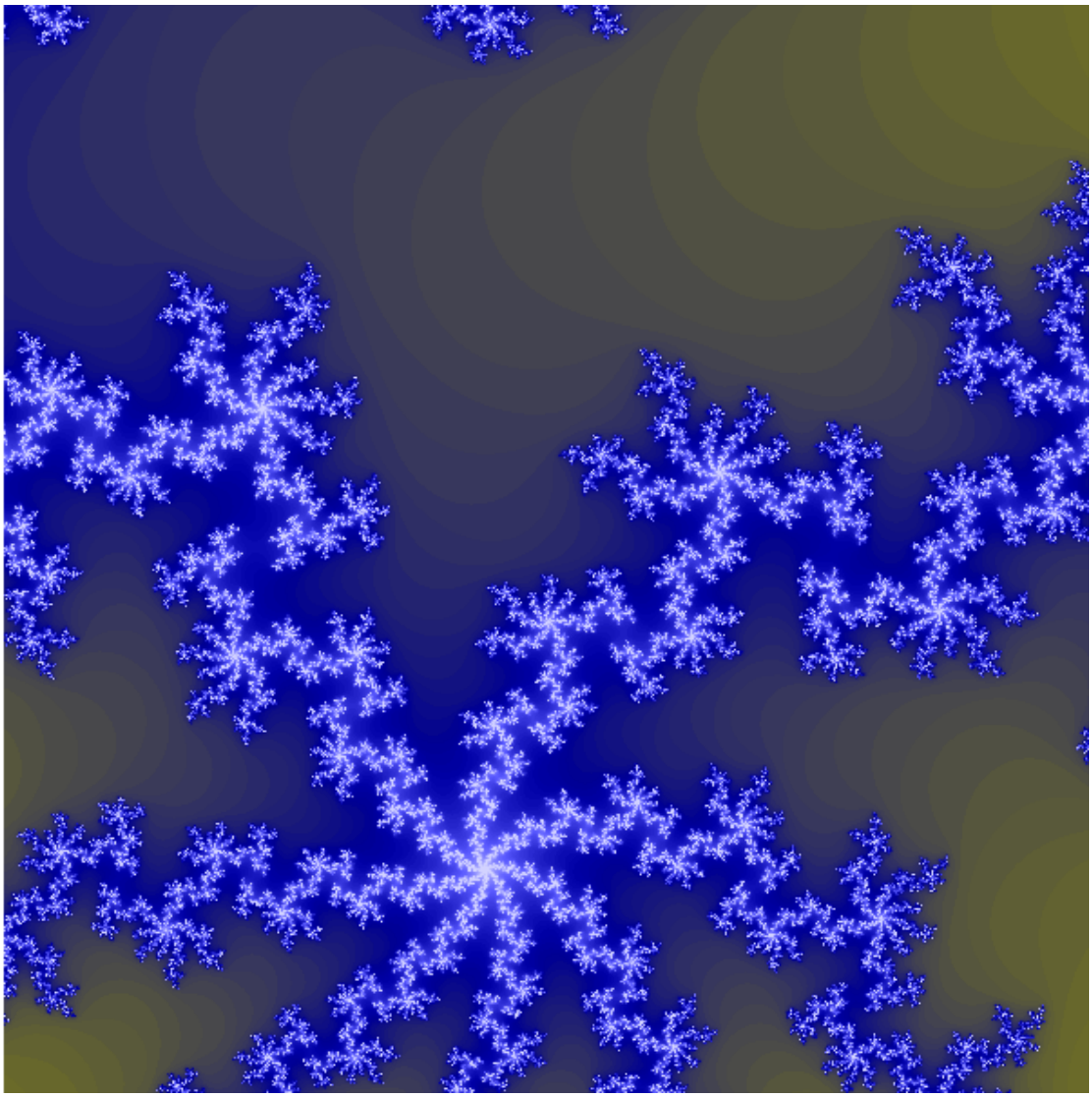
Εικόνα 3.1.4: Σύνολο Mandelbrot. 450 επαναλήψεις στην περιοχή:
$y_{\max} = 0{,}076433000000001$, $y_{\min} = 0{,}076335000000001$,
$x_{\max} = -1{,}369816000000000$, $x_{\min} = -1{,}369914000000000$.



Εικόνα 3.1.5: Σύνολο Mandelbrot. 1000 επαναλήψεις στην περιοχή:
$y_{\max} = 0{,}589844894197221, y_{\min} = 0{,}589842708174999,$
$x_{\max} = -0{,}535538975124999, x_{\min} = -0{,}535541161147221.$



Εικόνα 3.1.5: Σύνολο Julia. 100 επαναλήψεις στην περιοχή:
y_{max} = -0,081666666666666, y_{min} = -0,124599999999999,
x_{max} = -0,021140000000000, x_{min} = -0,064073333333334.



Εικόνα 3.1.5: Σύνολο Julia. 100 επαναλήψεις στην περιοχή:
y_{max} = -0,013999999999999, y_{min} = -0,055999999999999,
x_{max} = -0,105000000000000, x_{min} = -0,063000000000000.

Παράρτημα Ι: Πηγαίος Κώδικας

```
using System;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Drawing.Drawing2D;
using System.Threading;

namespace FractalDraw
{
	/// <summary>
	/// Summary description for DrawFractalForm.
	/// </summary>
	public class FractalDrawForm : System.Windows.Forms.Form
	{
		private System.Windows.Forms.Button btnDrawFractal;
		private System.Windows.Forms.Button btnClear;
		public System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.StatusBar statusBar1;
		private System.Windows.Forms.MainMenu mainMenu1;
		private System.Windows.Forms.MenuItem menuItem1;
		private System.Windows.Forms.MenuItem menuItem3;
		private System.Windows.Forms.MenuItem menuItem4;
		private System.Windows.Forms.MenuItem menuItem5;
		private System.Windows.Forms.MenuItem menuItem6;
		private System.Windows.Forms.MenuItem menuItem7;
		private System.Windows.Forms.MenuItem menuItem8;
		private System.Windows.Forms.StatusBarPanel statusBarPanel1;
		private System.Windows.Forms.ToolTip toolTip1;
		private System.Windows.Forms.ToolTip toolTip2;
		private System.Windows.Forms.MenuItem menuItem9;
		private System.Windows.Forms.MenuItem menuItem10;
		private System.Windows.Forms.StatusBarPanel statusBarPanel2;
		private System.Windows.Forms.MenuItem menuItem11;
		private System.Windows.Forms.TabControl tabControl1;
		private System.Windows.Forms.TabPage tabPageLocation;
		private System.Windows.Forms.TabPage tabPageFormula;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.Label label5;
		private System.Windows.Forms.Label label6;
		private System.Windows.Forms.Label label7;
		private System.Windows.Forms.Label label8;
		public System.Windows.Forms.NumericUpDown ymaxUpDown;
		public System.Windows.Forms.NumericUpDown yminUpDown;
		public System.Windows.Forms.NumericUpDown xminUpDown;
		public System.Windows.Forms.NumericUpDown xmaxUpDown;
		private System.Windows.Forms.Label label9;
		public System.Windows.Forms.NumericUpDown iterationsUpDown;
		private System.Windows.Forms.Button bttnReset;
		private System.Windows.Forms.Label label10;
		public System.Windows.Forms.Label XoverYlabel;
		private System.Windows.Forms.Label label11;
		private System.Windows.Forms.Label label12;
		private System.Windows.Forms.Label label13;
		private System.Windows.Forms.Label label14;
		private System.Windows.Forms.NumericUpDown ReZzeroUpDown;
		private System.Windows.Forms.NumericUpDown ImZzeroUpDown;
		private System.Windows.Forms.GroupBox groupBox1;
		private System.Windows.Forms.GroupBox groupBox2;
		private System.Windows.Forms.Label label15;
		private System.Windows.Forms.Button btnFormulaReset;
		private System.Windows.Forms.RadioButton rdioMandel;
		private System.Windows.Forms.RadioButton rdioJulia;
		private System.Windows.Forms.MenuItem menuItem13;
		private System.Windows.Forms.MenuItem menuItem12;
		private System.Windows.Forms.MenuItem menuItem2;
		private System.Windows.Forms.SaveFileDialog saveFileDialog;
		public System.Windows.Forms.ContextMenu contextMenu1;
```

```
		private System.Windows.Forms.MenuItem menuItem14;
		private System.Windows.Forms.MenuItem menuItem15;
		private System.Windows.Forms.MenuItem menuItem16;
		private System.Windows.Forms.StatusBarPanel statusBarPanel3;
		private System.Windows.Forms.PictureBox pictureBox2;
		private System.Windows.Forms.MenuItem menuRealTimeJulia;
		public System.Windows.Forms.CheckBox RealTimeJliaChkBx;
		private System.Windows.Forms.PictureBox pictureBox3;
		private System.ComponentModel.IContainer components;


		public FractalDrawForm()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if (components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.components = new System.ComponentModel.Container();
			System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(FractalDrawForm));
			this.btnDrawFractal = new System.Windows.Forms.Button();
			this.btnClear = new System.Windows.Forms.Button();
			this.pictureBox1 = new System.Windows.Forms.PictureBox();
			this.contextMenu1 = new System.Windows.Forms.ContextMenu();
			this.menuItem14 = new System.Windows.Forms.MenuItem();
			this.menuItem15 = new System.Windows.Forms.MenuItem();
			this.menuItem16 = new System.Windows.Forms.MenuItem();
			this.menuRealTimeJulia = new System.Windows.Forms.MenuItem();
			this.statusBar1 = new System.Windows.Forms.StatusBar();
			this.statusBarPanel1 = new
System.Windows.Forms.StatusBarPanel();
			this.statusBarPanel2 = new
System.Windows.Forms.StatusBarPanel();
			this.statusBarPanel3 = new
System.Windows.Forms.StatusBarPanel();
			this.mainMenu1 = new System.Windows.Forms.MainMenu();
			this.menuItem1 = new System.Windows.Forms.MenuItem();
			this.menuItem13 = new System.Windows.Forms.MenuItem();
			this.menuItem12 = new System.Windows.Forms.MenuItem();
			this.menuItem2 = new System.Windows.Forms.MenuItem();
			this.menuItem3 = new System.Windows.Forms.MenuItem();
			this.menuItem11 = new System.Windows.Forms.MenuItem();
			this.menuItem9 = new System.Windows.Forms.MenuItem();
			this.menuItem10 = new System.Windows.Forms.MenuItem();
			this.menuItem4 = new System.Windows.Forms.MenuItem();
			this.menuItem5 = new System.Windows.Forms.MenuItem();
			this.menuItem6 = new System.Windows.Forms.MenuItem();
			this.menuItem7 = new System.Windows.Forms.MenuItem();
			this.menuItem8 = new System.Windows.Forms.MenuItem();
			this.toolTip1 = new
System.Windows.Forms.ToolTip(this.components);
			this.XoverYlabel = new System.Windows.Forms.Label();
```

```
                        this.label2 = new System.Windows.Forms.Label();
                        this.label4 = new System.Windows.Forms.Label();
                        this.label1 = new System.Windows.Forms.Label();
                        this.label3 = new System.Windows.Forms.Label();
                        this.bttnReset = new System.Windows.Forms.Button();
                        this.btnFormulaReset = new System.Windows.Forms.Button();
                        this.RealTimeJliaChkBx = new System.Windows.Forms.CheckBox();
                        this.toolTip2 = new
System.Windows.Forms.ToolTip(this.components);
                        this.tabControl1 = new System.Windows.Forms.TabControl();
                        this.tabPageLocation = new System.Windows.Forms.TabPage();
                        this.groupBox2 = new System.Windows.Forms.GroupBox();
                        this.label8 = new System.Windows.Forms.Label();
                        this.label5 = new System.Windows.Forms.Label();
                        this.label10 = new System.Windows.Forms.Label();
                        this.xminUpDown = new System.Windows.Forms.NumericUpDown();
                        this.ymaxUpDown = new System.Windows.Forms.NumericUpDown();
                        this.label7 = new System.Windows.Forms.Label();
                        this.label6 = new System.Windows.Forms.Label();
                        this.yminUpDown = new System.Windows.Forms.NumericUpDown();
                        this.xmaxUpDown = new System.Windows.Forms.NumericUpDown();
                        this.tabPageFormula = new System.Windows.Forms.TabPage();
                        this.groupBox1 = new System.Windows.Forms.GroupBox();
                        this.label15 = new System.Windows.Forms.Label();
                        this.label13 = new System.Windows.Forms.Label();
                        this.ReZzeroUpDown = new System.Windows.Forms.NumericUpDown();
                        this.label12 = new System.Windows.Forms.Label();
                        this.iterationsUpDown = new
System.Windows.Forms.NumericUpDown();
                        this.label11 = new System.Windows.Forms.Label();
                        this.label9 = new System.Windows.Forms.Label();
                        this.ImZzeroUpDown = new System.Windows.Forms.NumericUpDown();
                        this.label14 = new System.Windows.Forms.Label();
                        this.rdioMandel = new System.Windows.Forms.RadioButton();
                        this.rdioJulia = new System.Windows.Forms.RadioButton();
                        this.saveFileDialog = new System.Windows.Forms.SaveFileDialog();
                        this.pictureBox2 = new System.Windows.Forms.PictureBox();
                        this.pictureBox3 = new System.Windows.Forms.PictureBox();

        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanel1)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanel2)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanel3)).BeginInit();
                        this.tabControl1.SuspendLayout();
                        this.tabPageLocation.SuspendLayout();
                        this.groupBox2.SuspendLayout();

        ((System.ComponentModel.ISupportInitialize)(this.xminUpDown)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.ymaxUpDown)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.yminUpDown)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.xmaxUpDown)).BeginInit();
                        this.tabPageFormula.SuspendLayout();
                        this.groupBox1.SuspendLayout();

        ((System.ComponentModel.ISupportInitialize)(this.ReZzeroUpDown)).BeginInit();

        ((System.ComponentModel.ISupportInitialize)(this.iterationsUpDown)).BeginInit()
;

        ((System.ComponentModel.ISupportInitialize)(this.ImZzeroUpDown)).BeginInit();
                        this.SuspendLayout();
                        //
                        // btnDrawFractal
                        //
                        this.btnDrawFractal.AccessibleDescription = "";
                        this.btnDrawFractal.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
                        this.btnDrawFractal.Location = new System.Drawing.Point(32,
344);
                        this.btnDrawFractal.Name = "btnDrawFractal";
                        this.btnDrawFractal.Size = new System.Drawing.Size(96, 23);
                        this.btnDrawFractal.TabIndex = 0;
                        this.btnDrawFractal.Text = "Draw Fractal";
```

```
                        this.toolTip1.SetToolTip(this.btnDrawFractal, "Draws specified
fractal");
                        this.btnDrawFractal.Click += new
System.EventHandler(this.btnDrawFractal_Click);
                        //
                        // btnClear
                        //
                        this.btnClear.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                        this.btnClear.Location = new System.Drawing.Point(184, 344);
                        this.btnClear.Name = "btnClear";
                        this.btnClear.Size = new System.Drawing.Size(96, 23);
                        this.btnClear.TabIndex = 1;
                        this.btnClear.Text = "Clear";
                        this.toolTip2.SetToolTip(this.btnClear, "Clears fractal image");
                        this.btnClear.Click += new
System.EventHandler(this.btnClear_Click);
                        //
                        // pictureBox1
                        //
                        this.pictureBox1.BackColor = System.Drawing.SystemColors.Window;
                        this.pictureBox1.ContextMenu = this.contextMenu1;
                        this.pictureBox1.Cursor = System.Windows.Forms.Cursors.Default;
                        this.pictureBox1.Location = new System.Drawing.Point(320, 16);
                        this.pictureBox1.Name = "pictureBox1";
                        this.pictureBox1.Size = new System.Drawing.Size(600, 600);
                        this.pictureBox1.TabIndex = 2;
                        this.pictureBox1.TabStop = false;
                        this.pictureBox1.Click += new
System.EventHandler(this.pictureBox1_Click);
                        this.pictureBox1.MouseEnter += new
System.EventHandler(this.pictureBox1_MouseEnter);
                        this.pictureBox1.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseUp);
                        this.pictureBox1.DoubleClick += new
System.EventHandler(this.pictureBox1_DoubleClick);
                        this.pictureBox1.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseMove);
                        this.pictureBox1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.PictureBox1_MouseDown);
                        //
                        // contextMenu1
                        //
                        this.contextMenu1.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {


this.menuItem14,
this.menuItem15,
this.menuItem16,
this.menuRealTimeJulia});
                        //
                        // menuItem14
                        //
                        this.menuItem14.Index = 0;
                        this.menuItem14.Text = "Zoom into selected rectangle";
                        this.menuItem14.Click += new
System.EventHandler(this.menuItem14_Click);
                        //
                        // menuItem15
                        //
                        this.menuItem15.Index = 1;
                        this.menuItem15.Text = "-";
                        //
                        // menuItem16
                        //
                        this.menuItem16.Index = 2;
                        this.menuItem16.Text = "Draw Julia Set from this point";
                        this.menuItem16.Click += new
System.EventHandler(this.menuItem16_Click);
                        //
                        // menuRealTimeJulia
                        //
                        this.menuRealTimeJulia.Index = 3;
                        this.menuRealTimeJulia.Text = "Real time Julia Set";
                        //
                        // statusBar1
                        //
```

```
                        this.statusBar1.Location = new System.Drawing.Point(0, 650);
                        this.statusBar1.Name = "statusBar1";
                        this.statusBar1.Panels.AddRange(new
System.Windows.Forms.StatusBarPanel[] {


this.statusBarPanel1,
this.statusBarPanel2,
this.statusBarPanel3});
                        this.statusBar1.ShowPanels = true;
                        this.statusBar1.Size = new System.Drawing.Size(930, 22);
                        this.statusBar1.TabIndex = 3;
                        //
                        // statusBarPanel1
                        //
                        this.statusBarPanel1.AutoSize =
System.Windows.Forms.StatusBarPanelAutoSize.Contents;
                        this.statusBarPanel1.Text = "Fractal Generator version 0.0.1";
                        this.statusBarPanel1.Width = 171;
                        //
                        // statusBarPanel2
                        //
                        this.statusBarPanel2.Text = "Cursor Coordinates";
                        this.statusBarPanel2.ToolTipText = "Displays the Coordinates of
the cursor position in the current complex plane. Dou" +
                                "ble click to create Julia Set.";
                        this.statusBarPanel2.Width = 305;
                        //
                        // statusBarPanel3
                        //
                        this.statusBarPanel3.Text = "Magnification";
                        this.statusBarPanel3.ToolTipText = "The amount of magnification
from the original size";
                        this.statusBarPanel3.Width = 250;
                        //
                        // mainMenu1
                        //
                        this.mainMenu1.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {


this.menuItem1,
this.menuItem3,
this.menuItem9,
this.menuItem4,
this.menuItem5});
                        //
                        // menuItem1
                        //
                        this.menuItem1.Index = 0;
                        this.menuItem1.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {


this.menuItem13,
this.menuItem12,
this.menuItem2});
                        this.menuItem1.Text = "File";
                        //
                        // menuItem13
                        //
                        this.menuItem13.Index = 0;
                        this.menuItem13.Shortcut = System.Windows.Forms.Shortcut.CtrlS;
                        this.menuItem13.Text = "Save Image As...";
                        this.menuItem13.Click += new
System.EventHandler(this.menuItem2_Click);
                        //
                        // menuItem12
                        //
                        this.menuItem12.Index = 1;
                        this.menuItem12.Text = "-";
                        //
                        // menuItem2
                        //
                        this.menuItem2.Index = 2;
                        this.menuItem2.Shortcut = System.Windows.Forms.Shortcut.AltF4;
                        this.menuItem2.Text = "Exit";
```

```
                        this.menuItem2.Click += new
System.EventHandler(this.menuItem2_Click_1);
                        //
                        // menuItem3
                        //
                        this.menuItem3.Index = 1;
                        this.menuItem3.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {


this.menuItem11});
                        this.menuItem3.Text = "Edit";
                        //
                        // menuItem11
                        //
                        this.menuItem11.Index = 0;
                        this.menuItem11.Shortcut = System.Windows.Forms.Shortcut.CtrlC;
                        this.menuItem11.Text = "Copy ImageTo Clipboard";
                        this.menuItem11.Click += new
System.EventHandler(this.menuItem11_Click);
                        //
                        // menuItem9
                        //
                        this.menuItem9.Index = 2;
                        this.menuItem9.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {


this.menuItem10});
                        this.menuItem9.Text = "Fractal";
                        //
                        // menuItem10
                        //
                        this.menuItem10.Index = 0;
                        this.menuItem10.Text = "Select Type";
                        this.menuItem10.Click += new
System.EventHandler(this.menuItem10_Click);
                        //
                        // menuItem4
                        //
                        this.menuItem4.Index = 3;
                        this.menuItem4.Text = "View";
                        //
                        // menuItem5
                        //
                        this.menuItem5.Index = 4;
                        this.menuItem5.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {


this.menuItem6,


this.menuItem7,


this.menuItem8});
                        this.menuItem5.Text = "Help";
                        //
                        // menuItem6
                        //
                        this.menuItem6.Index = 0;
                        this.menuItem6.Text = "Version History";
                        //
                        // menuItem7
                        //
                        this.menuItem7.Index = 1;
                        this.menuItem7.Text = "-";
                        //
                        // menuItem8
                        //
                        this.menuItem8.Index = 2;
                        this.menuItem8.Text = "About";
                        this.menuItem8.Click += new
System.EventHandler(this.menuItem8_Click);
                        //
                        // XoverYlabel
```

```
                        //
                        this.XoverYlabel.Location = new System.Drawing.Point(80, 136);
                        this.XoverYlabel.Name = "XoverYlabel";
                        this.XoverYlabel.Size = new System.Drawing.Size(128, 23);
                        this.XoverYlabel.TabIndex = 14;
                        this.XoverYlabel.Text = "1";
                        this.toolTip1.SetToolTip(this.XoverYlabel, "The amount of
magnification of the X axis scale over the Y axis scale");
                        //
                        // label2
                        //
                        this.label2.Location = new System.Drawing.Point(8, 24);
                        this.label2.Name = "label2";
                        this.label2.Size = new System.Drawing.Size(128, 23);
                        this.label2.TabIndex = 5;
                        this.label2.Text = "Upper Border, Im(z      ):";
                        this.toolTip1.SetToolTip(this.label2, "Maximum Value = 15,
Minimum Value = -15");
                        //
                        // label4
                        //
                        this.label4.Location = new System.Drawing.Point(8, 72);
                        this.label4.Name = "label4";
                        this.label4.Size = new System.Drawing.Size(128, 23);
                        this.label4.TabIndex = 7;
                        this.label4.Text = "Right Border, Re(z      ):";
                        this.toolTip1.SetToolTip(this.label4, "Maximum Value = 15,
Minimum Value = -15");
                        //
                        // label1
                        //
                        this.label1.Location = new System.Drawing.Point(8, 48);
                        this.label1.Name = "label1";
                        this.label1.Size = new System.Drawing.Size(128, 23);
                        this.label1.TabIndex = 4;
                        this.label1.Text = "Lower Border, Im(z     ):";
                        this.toolTip1.SetToolTip(this.label1, "Maximum Value = 15,
Minimum Value = -15");
                        //
                        // label3
                        //
                        this.label3.Location = new System.Drawing.Point(8, 96);
                        this.label3.Name = "label3";
                        this.label3.Size = new System.Drawing.Size(120, 23);
                        this.label3.TabIndex = 6;
                        this.label3.Text = "Left Border, Re(z      ):";
                        this.toolTip1.SetToolTip(this.label3, "Maximum Value = 15,
Minimum Value = -15");
                        //
                        // bttnReset
                        //
                        this.bttnReset.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                        this.bttnReset.Location = new System.Drawing.Point(160, 192);
                        this.bttnReset.Name = "bttnReset";
                        this.bttnReset.Size = new System.Drawing.Size(104, 23);
                        this.bttnReset.TabIndex = 17;
                        this.bttnReset.Text = "Reset to Defaults";
                        this.toolTip1.SetToolTip(this.bttnReset, "Resets the numericals
of the \"Location\" tab page");
                        this.bttnReset.Click += new
System.EventHandler(this.bttnReset_Click);
                        //
                        // btnFormulaReset
                        //
                        this.btnFormulaReset.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
                        this.btnFormulaReset.Location = new System.Drawing.Point(160,
192);
                        this.btnFormulaReset.Name = "btnFormulaReset";
                        this.btnFormulaReset.Size = new System.Drawing.Size(104, 23);
                        this.btnFormulaReset.TabIndex = 9;
                        this.btnFormulaReset.Text = "Reset to Defaults";
                        this.toolTip1.SetToolTip(this.btnFormulaReset, "Resets the
numericals of the \"Formula\" tab page");
                        this.btnFormulaReset.Click += new
System.EventHandler(this.btnFormulaReset_Click);
                        //
```

```
                        // RealTimeJliaChkBx
                        //
                        this.RealTimeJliaChkBx.Location = new System.Drawing.Point(8,
104);
                        this.RealTimeJliaChkBx.Name = "RealTimeJliaChkBx";
                        this.RealTimeJliaChkBx.Size = new System.Drawing.Size(136, 24);
                        this.RealTimeJliaChkBx.TabIndex = 9;
                        this.RealTimeJliaChkBx.Text = "Real Time Julia Set";
                        this.toolTip1.SetToolTip(this.RealTimeJliaChkBx, "Check to
enable Real Time Julia Set. It disables Zoom functionality");
                        this.RealTimeJliaChkBx.CheckStateChanged += new
System.EventHandler(this.JliaChkBx_CheckChanged);
                        //
                        // tabControl1
                        //
                        this.tabControl1.Controls.AddRange(new
System.Windows.Forms.Control[] {


this.tabPageLocation,


this.tabPageFormula});
                        this.tabControl1.Location = new System.Drawing.Point(8, 16);
                        this.tabControl1.Name = "tabControl1";
                        this.tabControl1.SelectedIndex = 0;
                        this.tabControl1.Size = new System.Drawing.Size(296, 256);
                        this.tabControl1.TabIndex = 2;
                        //
                        // tabPageLocation
                        //
                        this.tabPageLocation.Controls.AddRange(new
System.Windows.Forms.Control[] {


this.groupBox2,


this.bttnReset});
                        this.tabPageLocation.Location = new System.Drawing.Point(4, 22);
                        this.tabPageLocation.Name = "tabPageLocation";
                        this.tabPageLocation.Size = new System.Drawing.Size(288, 230);
                        this.tabPageLocation.TabIndex = 0;
                        this.tabPageLocation.Text = "Location";
                        this.tabPageLocation.ToolTipText = "Specifies the borders of the
fractal to be generated.";
                        //
                        // groupBox2
                        //
                        this.groupBox2.Controls.AddRange(new
System.Windows.Forms.Control[] {


        this.XoverYlabel,
        this.label8,
        this.label5,
        this.label2,
        this.label10,
        this.xminUpDown,
        this.ymaxUpDown,
        this.label7,
        this.label4,
        this.label6,
        this.label1,
        this.yminUpDown,
        this.label3,
        this.xmaxUpDown});
                        this.groupBox2.Location = new System.Drawing.Point(0, 8);
                        this.groupBox2.Name = "groupBox2";
                        this.groupBox2.Size = new System.Drawing.Size(288, 168);
                        this.groupBox2.TabIndex = 15;
                        this.groupBox2.TabStop = false;
                        this.groupBox2.Text = "Define the Location of the Set";
                        //
                        // label8
                        //
                        this.label8.AutoSize = true;
```

```
                        this.label8.Enabled = false;
                        this.label8.Font = new System.Drawing.Font("Microsoft Sans
Serif", 6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                        this.label8.Location = new System.Drawing.Point(94, 104);
                        this.label8.Name = "label8";
                        this.label8.Size = new System.Drawing.Size(16, 10);
                        this.label8.TabIndex = 11;
                        this.label8.Text = "min";
                        //
                        // label5
                        //
                        this.label5.AutoSize = true;
                        this.label5.Enabled = false;
                        this.label5.Font = new System.Drawing.Font("Microsoft Sans
Serif", 6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                        this.label5.Location = new System.Drawing.Point(103, 32);
                        this.label5.Name = "label5";
                        this.label5.Size = new System.Drawing.Size(19, 10);
                        this.label5.TabIndex = 8;
                        this.label5.Text = "max";
                        //
                        // label10
                        //
                        this.label10.Location = new System.Drawing.Point(16, 136);
                        this.label10.Name = "label10";
                        this.label10.Size = new System.Drawing.Size(64, 23);
                        this.label10.TabIndex = 13;
                        this.label10.Text = "Scale x/y =";
                        //
                        // xminUpDown
                        //
                        this.xminUpDown.DecimalPlaces = 15;
                        this.xminUpDown.Increment = new System.Decimal(new int[] {

                                                    1,

                                                    0,

                                                    0,

                                                    983040});
                        this.xminUpDown.Location = new System.Drawing.Point(136, 96);
                        this.xminUpDown.Maximum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                0});
                        this.xminUpDown.Minimum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                -2147483648});
                        this.xminUpDown.Name = "xminUpDown";
                        this.xminUpDown.Size = new System.Drawing.Size(144, 20);
                        this.xminUpDown.TabIndex = 3;
                        this.xminUpDown.Value = new System.Decimal(new int[] {

                                               21,

                                               0,

                                               0,

                                               -2147418112});
                        this.xminUpDown.GotFocus += new
System.EventHandler(this.xminUpDown_GotFocus);
```

```
                        this.xminUpDown.ValueChanged += new
System.EventHandler(this.xminUpDown_ValueChanged);
                        //
                        // ymaxUpDown
                        //
                        this.ymaxUpDown.DecimalPlaces = 15;
                        this.ymaxUpDown.Increment = new System.Decimal(new int[] {

                                                    1,

                                                    0,

                                                    0,

                                                    983040});
                        this.ymaxUpDown.Location = new System.Drawing.Point(136, 24);
                        this.ymaxUpDown.Maximum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                0});
                        this.ymaxUpDown.Minimum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                -2147483648});
                        this.ymaxUpDown.Name = "ymaxUpDown";
                        this.ymaxUpDown.Size = new System.Drawing.Size(144, 20);
                        this.ymaxUpDown.TabIndex = 0;
                        this.ymaxUpDown.Value = new System.Decimal(new int[] {

                                                21,

                                                0,

                                                0,

                                                65536});
                        this.ymaxUpDown.GotFocus += new
System.EventHandler(this.ymaxUpDown_GotFocus);
                        this.ymaxUpDown.ValueChanged += new
System.EventHandler(this.ymaxUpDown_ValueChanged);
                        //
                        // label7
                        //
                        this.label7.AutoSize = true;
                        this.label7.Enabled = false;
                        this.label7.Font = new System.Drawing.Font("Microsoft Sans
Serif", 6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                        this.label7.Location = new System.Drawing.Point(101, 80);
                        this.label7.Name = "label7";
                        this.label7.Size = new System.Drawing.Size(19, 10);
                        this.label7.TabIndex = 10;
                        this.label7.Text = "max";
                        //
                        // label6
                        //
                        this.label6.AutoSize = true;
                        this.label6.Enabled = false;
                        this.label6.Font = new System.Drawing.Font("Microsoft Sans
Serif", 6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                        this.label6.Location = new System.Drawing.Point(103, 56);
                        this.label6.Name = "label6";
                        this.label6.Size = new System.Drawing.Size(16, 10);
                        this.label6.TabIndex = 9;
                        this.label6.Text = "min";
                        //
```

```
                        // yminUpDown
                        //
                        this.yminUpDown.DecimalPlaces = 15;
                        this.yminUpDown.Increment = new System.Decimal(new int[] {

                                                    1,

                                                    0,

                                                    0,

                                                    983040});
                        this.yminUpDown.Location = new System.Drawing.Point(136, 48);
                        this.yminUpDown.Maximum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                0});
                        this.yminUpDown.Minimum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                -2147483648});
                        this.yminUpDown.Name = "yminUpDown";
                        this.yminUpDown.Size = new System.Drawing.Size(144, 20);
                        this.yminUpDown.TabIndex = 1;
                        this.yminUpDown.Value = new System.Decimal(new int[] {

                                               21,

                                               0,

                                               0,

                                               -2147418112});
                        this.yminUpDown.GotFocus += new
System.EventHandler(this.yminUpDown_GotFocus);
                        this.yminUpDown.ValueChanged += new
System.EventHandler(this.yminUpDown_ValueChanged);
                        //
                        // xmaxUpDown
                        //
                        this.xmaxUpDown.DecimalPlaces = 15;
                        this.xmaxUpDown.Increment = new System.Decimal(new int[] {

                                                    1,

                                                    0,

                                                    0,

                                                    983040});
                        this.xmaxUpDown.Location = new System.Drawing.Point(136, 72);
                        this.xmaxUpDown.Maximum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,

                                                0});
                        this.xmaxUpDown.Minimum = new System.Decimal(new int[] {

                                                15,

                                                0,

                                                0,
```

```
                                        -2147483648});
                    this.xmaxUpDown.Name = "xmaxUpDown";
                    this.xmaxUpDown.Size = new System.Drawing.Size(144, 20);
                    this.xmaxUpDown.TabIndex = 2;
                    this.xmaxUpDown.Value = new System.Decimal(new int[] {

                                        21,

                                        0,

                                        0,

                                        65536});
                    this.xmaxUpDown.GotFocus += new
System.EventHandler(this.xmaxUpDown_GotFocus);
                    this.xmaxUpDown.ValueChanged += new
System.EventHandler(this.xmaxUpDown_ValueChanged);
                    //
                    // tabPageFormula
                    //
                    this.tabPageFormula.Controls.AddRange(new
System.Windows.Forms.Control[] {


this.btnFormulaReset,
this.groupBox1});
                    this.tabPageFormula.Location = new System.Drawing.Point(4, 22);
                    this.tabPageFormula.Name = "tabPageFormula";
                    this.tabPageFormula.Size = new System.Drawing.Size(288, 230);
                    this.tabPageFormula.TabIndex = 1;
                    this.tabPageFormula.Text = "Formula";
                    //
                    // groupBox1
                    //
                    this.groupBox1.Controls.AddRange(new
System.Windows.Forms.Control[] {


        this.label15,
        this.label13,
        this.ReZzeroUpDown,
        this.label12,
        this.iterationsUpDown,
        this.label11,
        this.label9,
        this.ImZzeroUpDown,
        this.label14,
        this.RealTimeJliaChkBx});
                    this.groupBox1.Location = new System.Drawing.Point(0, 8);
                    this.groupBox1.Name = "groupBox1";
                    this.groupBox1.Size = new System.Drawing.Size(288, 136);
                    this.groupBox1.TabIndex = 8;
                    this.groupBox1.TabStop = false;
                    this.groupBox1.Text = "Define the Formula for the Set";
                    //
                    // label15
                    //
                    this.label15.Font = new System.Drawing.Font("Times New Roman",
6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                    this.label15.Location = new System.Drawing.Point(104, 80);
                    this.label15.Name = "label15";
                    this.label15.Size = new System.Drawing.Size(8, 8);
                    this.label15.TabIndex = 8;
                    this.label15.Text = "0";
                    //
                    // label13
                    //
                    this.label13.Font = new System.Drawing.Font("Times New Roman",
6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                    this.label13.Location = new System.Drawing.Point(104, 56);
                    this.label13.Name = "label13";
                    this.label13.Size = new System.Drawing.Size(8, 8);
                    this.label13.TabIndex = 4;
                    this.label13.Text = "0";
```

```
                        //
                        // ReZzeroUpDown
                        //
                        this.ReZzeroUpDown.DecimalPlaces = 15;
                        this.ReZzeroUpDown.Increment = new System.Decimal(new int[] {
                                                                    1,
                                                                    0,
                                                                    0,
                                                                    983040});
                        this.ReZzeroUpDown.Location = new System.Drawing.Point(128, 48);
                        this.ReZzeroUpDown.Minimum = new System.Decimal(new int[] {
                                                                100,
                                                                0,
                                                                0,
                                                                -2147483648});
                        this.ReZzeroUpDown.Name = "ReZzeroUpDown";
                        this.ReZzeroUpDown.Size = new System.Drawing.Size(128, 20);
                        this.ReZzeroUpDown.TabIndex = 6;
                        this.ReZzeroUpDown.GotFocus += new
System.EventHandler(this.ReZzeroUpDown_GotFocus);
                        //
                        // label12
                        //
                        this.label12.Location = new System.Drawing.Point(8, 72);
                        this.label12.Name = "label12";
                        this.label12.Size = new System.Drawing.Size(120, 23);
                        this.label12.TabIndex = 3;
                        this.label12.Text = "Starting Point, Im(z  ):";
                        //
                        // iterationsUpDown
                        //
                        this.iterationsUpDown.Location = new System.Drawing.Point(128,
24);
                        this.iterationsUpDown.Maximum = new System.Decimal(new int[] {
                                                                    10000,
                                                                    0,
                                                                    0,
                                                                    0});
                        this.iterationsUpDown.Name = "iterationsUpDown";
                        this.iterationsUpDown.Size = new System.Drawing.Size(72, 20);
                        this.iterationsUpDown.TabIndex = 1;
                        this.iterationsUpDown.Value = new System.Decimal(new int[] {
                                                                100,
                                                                0,
                                                                0,
                                                                0});
                        this.iterationsUpDown.GotFocus += new
System.EventHandler(this.iterationsUpDown_GotFocus);
                        //
                        // label11
                        //
                        this.label11.Location = new System.Drawing.Point(8, 48);
                        this.label11.Name = "label11";
                        this.label11.Size = new System.Drawing.Size(120, 23);
                        this.label11.TabIndex = 2;
                        this.label11.Text = "Starting Point, Re(z  ):";
                        //
                        // label9
                        //
                        this.label9.Location = new System.Drawing.Point(8, 24);
                        this.label9.Name = "label9";
```

```
                        this.label9.Size = new System.Drawing.Size(112, 23);
                        this.label9.TabIndex = 0;
                        this.label9.Text = "Maximum Iterations:";
                        //
                        // ImZzeroUpDown
                        //
                        this.ImZzeroUpDown.DecimalPlaces = 15;
                        this.ImZzeroUpDown.Increment = new System.Decimal(new int[] {

                                                    1,

                                                    0,

                                                    0,

                                                    983040});
                        this.ImZzeroUpDown.Location = new System.Drawing.Point(128, 72);
                        this.ImZzeroUpDown.Minimum = new System.Decimal(new int[] {

                                                100,

                                                0,

                                                0,

                                                -2147483648});
                        this.ImZzeroUpDown.Name = "ImZzeroUpDown";
                        this.ImZzeroUpDown.Size = new System.Drawing.Size(128, 20);
                        this.ImZzeroUpDown.TabIndex = 7;
                        this.ImZzeroUpDown.GotFocus += new
System.EventHandler(this.ImZzeroUpDown_GotFocus);
                        //
                        // label14
                        //
                        this.label14.Font = new System.Drawing.Font("Times New Roman",
6F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((System.Byte)(161)));
                        this.label14.Location = new System.Drawing.Point(104, 80);
                        this.label14.Name = "label14";
                        this.label14.Size = new System.Drawing.Size(8, 8);
                        this.label14.TabIndex = 5;
                        this.label14.Text = "0";
                        //
                        // rdioMandel
                        //
                        this.rdioMandel.Checked = true;
                        this.rdioMandel.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
                        this.rdioMandel.Location = new System.Drawing.Point(32, 296);
                        this.rdioMandel.Name = "rdioMandel";
                        this.rdioMandel.TabIndex = 4;
                        this.rdioMandel.TabStop = true;
                        this.rdioMandel.Text = "Mandelbrot Set";
                        //
                        // rdioJulia
                        //
                        this.rdioJulia.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                        this.rdioJulia.Location = new System.Drawing.Point(160, 296);
                        this.rdioJulia.Name = "rdioJulia";
                        this.rdioJulia.TabIndex = 5;
                        this.rdioJulia.TabStop = true;
                        this.rdioJulia.Text = "Julia Set";
                        //
                        // saveFileDialog
                        //
                        this.saveFileDialog.DefaultExt = "bmp";
                        this.saveFileDialog.FileName = "Fractal1";
                        this.saveFileDialog.Filter = "bitmap files (*.bmp)|*.bmp|All
Files (*.*)|*.*";
                        this.saveFileDialog.Title = "Save the dislpayed fractal";
                        //
                        // pictureBox2
                        //
                        this.pictureBox2.Location = new System.Drawing.Point(55, 392);
                        this.pictureBox2.Name = "pictureBox2";
                        this.pictureBox2.Size = new System.Drawing.Size(200, 100);
                        this.pictureBox2.TabIndex = 6;
```

```
                        this.pictureBox2.TabStop = false;
                        this.pictureBox2.Click += new
System.EventHandler(this.pictureBox2_Click);
                        //
                        // pictureBox3
                        //
                        this.pictureBox3.Location = new System.Drawing.Point(56, 492);
                        this.pictureBox3.Name = "pictureBox3";
                        this.pictureBox3.Size = new System.Drawing.Size(200, 100);
                        this.pictureBox3.TabIndex = 7;
                        this.pictureBox3.TabStop = false;
                        //
                        // FractalDrawForm
                        //
                        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                        this.ClientSize = new System.Drawing.Size(930, 672);
                        this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                                        this.pictureBox3,

                                                        this.pictureBox2,

                                                        this.rdioJulia,

                                                        this.rdioMandel,

                                                        this.tabControl1,

                                                        this.statusBar1,

                                                        this.pictureBox1,

                                                        this.btnClear,

                                                        this.btnDrawFractal});
                        this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
                        this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
                        this.MaximumSize = new System.Drawing.Size(936, 704);
                        this.Menu = this.mainMenu1;
                        this.MinimumSize = new System.Drawing.Size(936, 704);
                        this.Name = "FractalDrawForm";
                        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                        this.Text = "Fractal Draw";

        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanel1)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanel2)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanel3)).EndInit();
                        this.tabControl1.ResumeLayout(false);
                        this.tabPageLocation.ResumeLayout(false);
                        this.groupBox2.ResumeLayout(false);

        ((System.ComponentModel.ISupportInitialize)(this.xminUpDown)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.ymaxUpDown)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.yminUpDown)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.xmaxUpDown)).EndInit();
                        this.tabPageFormula.ResumeLayout(false);
                        this.groupBox1.ResumeLayout(false);

        ((System.ComponentModel.ISupportInitialize)(this.ReZzeroUpDown)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.iterationsUpDown)).EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.ImZzeroUpDown)).EndInit();
                        this.ResumeLayout(false);

                }
                #endregion

                /// <summary>
```

```
        /// The main entry point for the application.
        /// </summary>
        //[STAThread]



        [STAThread]
        static void Main()
        {
                Application.Run(new FractalDrawForm());

        }


        #region Instantiations
    double thisCRe = new double();
        double thisCIm = new double();
        double ReZzero = new double();
        double ImZzero = new double();
        bool mandelBool = new bool();
        bool juliaBool = new bool();
        int clickTimes = new int();
        public Bitmap original_bitmap = new Bitmap(600,600);
        public Bitmap bmap = new Bitmap(600,600);
        public Bitmap realSmallBmap = new Bitmap(200,100);
        FractalSelector mySelection = new FractalSelector();
        Point pStart = Point.Empty;
        Point pFinish = Point.Empty;
        #endregion

        #region clearScreen
        /// <summary>
        /// Function ClearScreen paints original_bitmap white hence
        /// the fractal drawing area is cleared.
        /// It is the function that button btnClear (labelled 'Clear') uses.
        /// </summary>
        public void ClearScreen()
        {
                Bitmap clear_bitmap = new Bitmap(600,600);
                pictureBox1.Image = clear_bitmap;

                for (int k=0; k<clear_bitmap.Height; k++) //first 'for' scans
pixels vertically and gives color
                {
                        for (int l=0; l<clear_bitmap.Width; l++)//second 'for'
scans pixels horizontally
                        {
                                clear_bitmap.SetPixel(k,l,Color.GhostWhite);
                        }
                }

        }
#endregion

        #region Julia
        private void DrawJulia()
        {
                double Sx = -2.1;
                double Sy = -2.1;
                double Fx = 2.1;
                double Fy = 2.1;
                Color[] cs = new Color[256];
                Bitmap original_bitmap = new Bitmap(600,600);
                pictureBox1.Image = original_bitmap;
                double xmin = 0.0;
                double xmax = 0.0;
                double ymin = 0.0;
                double ymax = 0.0;
                double x1 = 0.0;
                double y1 = 0.0;
                double x = 0;
                double y = 0;
                int looper= 0;
                double linearPtoX, linearPtoY = 0.0;
                double CRe = -1.11;
                double CIm = 0.313;
```

```
                        //double tempy = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / original_bitmap.Width;
                        linearPtoY = (ymax - ymin) / original_bitmap.Height;
                        x1 = xmin;
                        y1 = ymax;
                        for(int z = 0; z < original_bitmap.Width; z++)
                        {

                                for(int w = 0; w < original_bitmap.Height; w++)
                                {
                                        x=0.0;
                                        y=0.0;
                                        looper = 0;
                                        while(looper < 100 && (x1 * x1) + (y1 * y1) < 4)
                                        {
                                                y1 = 2 * x1 * y1 + CIm;
                                                x1 = (x1 * x1) - (y1 * y1) + CRe;
                                                //y1=y;
                                                //x1=x;
                                                looper++;
                                        }


                                        double percent = looper / (100.0);
                                        int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb(Math.Abs(127-
val),Math.Abs(127-val),Math.Abs(80-val)));

                                }
                                x1+=linearPtoX;
                                y1-=linearPtoY;

                        }
                        bmap = original_bitmap;
                        pictureBox1.BackgroundImage = (Image)original_bitmap;
                        statusBarPanel1.Text = x1.ToString() +" , " + y1.ToString() +" ,
" + linearPtoY.ToString();

                }
                #endregion

                #region RemainderMandel
                private void DrawRemainderMandel()
                {
                        double Sx = -2.1;
                        double Sy = -2.1;
                        double Fx = 2.1;
                        double Fy = 2.1;
                        Color[] cs = new Color[256];
                        Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / this.Width;
                        linearPtoY = (ymax - ymin) / this.Height;
                        x = xmin;
                        for(z = 0; z < this.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < this.Height; w++)
                                {
                                        x1 = 0;
                                        y1 = 0;
                                        looper = 0;
                                        while(looper < 5000 && (x1 * x1) + (y1 * y1) < 4)
                                        {
                                                tempx = (x1 * x1) - (y1 * y1) + x;
```

```
                                        y1 = 2 * x1 * y1 + y;
                                        x1 = tempx;
                                        looper++;
                                }
                                double percent = Math.IEEERemainder(x/y,looper /
(5000.0));
                                int val = Math.Abs(((int)(percent * 255)));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                y += linearPtoY;
                        }
                        x += linearPtoX;
                }
                bmap = original_bitmap;
                pictureBox1.BackgroundImage = (Image)bmap;

        }
        #endregion

        #region CosSinMandel

        private void DrawCosSinMandel()
        {
                double Sx = -2.1;
                double Sy = -2.1;
                double Fx = 2.1;
                double Fy = 2.1;
                Color[] cs = new Color[256];
                Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                pictureBox1.Image = original_bitmap;
                double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                int looper, z, w = 0;
                double linearPtoX, linearPtoY = 0.0;
                xmin = Sx;
                ymin = Sy;
                xmax = Fx;
                ymax = Fy;
                linearPtoX = (xmax - xmin) / this.Width;
                linearPtoY = (ymax - ymin) / this.Height;
                x = xmin;
                for(z = 0; z < this.Width; z++)
                {
                        y = ymin;
                        for(w = 0; w < this.Height; w++)
                        {
                                x1 = 0;
                                y1 = 0;
                                looper = 0;
                                while(looper < 5000 && (x1 * x1) + (y1 * y1) < 4)
                                {
                                        tempx = (x1 * x1) - (y1 * y1) + x;
                                        y1 = 2 * x1 * y1 + y;
                                        x1 = tempx;
                                        looper++;
                                }
                                double percent = Math.Sin(x/y)*Math.Cos(looper /
(5000.0));
                                int val = Math.Abs(((int)(percent * 127)));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                y += linearPtoY;
                        }
                        x += linearPtoX;
                }
                bmap = original_bitmap;
                pictureBox1.BackgroundImage = (Image)bmap;

        }
        #endregion

        #region CosMandel

        private void DrawCosMandel()
        {
                double Sx = -2.1;
```

```
                        double Sy = -2.1;
                        double Fx = 2.1;
                        double Fy = 2.1;
                        Color[] cs = new Color[256];
                        Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / this.Width;
                        linearPtoY = (ymax - ymin) / this.Height;
                        x = xmin;
                        for(z = 0; z < this.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < this.Height; w++)
                                {
                                        x1 = 0;
                                        y1 = 0;
                                        looper = 0;
                                        while(looper < 5000 && (x1 * x1) + (y1 * y1) < 4)
                                        {
                                                tempx = (x1 * x1) - (y1 * y1) + x;
                                                y1 = 2 * x1 * y1 + y;
                                                x1 = tempx;
                                                looper++;
                                        }
                                        double percent = Math.Cos(looper / (1.0));
                                        int val = Math.Abs(((int)(percent * 255)));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(80-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                        y += linearPtoY;
                                }
                                x += linearPtoX;
                        }
                        bmap = original_bitmap;
                        pictureBox1.BackgroundImage = (Image)bmap;

                }
                #endregion

                #region Mandelbrot

                public void DrawMandel()
                {
                        double xmin = (double)xminUpDown.Value;
                        double ymin = (double)yminUpDown.Value;
                        double xmax = (double)xmaxUpDown.Value;
                        double ymax = (double)ymaxUpDown.Value;
                        Color[] cs = new Color[256];
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx = 0.0;
                        int looper = 0;
                        int z, w = 0;
                        double linearPtoX = 0.0;
                        double linearPtoY = 0.0;
                        linearPtoX = (xmax - xmin) / (pictureBox1.Width);
                        linearPtoY = (ymax - ymin) / (pictureBox1.Height);
                        x = xmin;
                        for(z = 0; z < original_bitmap.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < original_bitmap.Height; w++)
                                {
                                        x1 = (double)ReZzeroUpDown.Value;
                                        y1 = (double)ImZzeroUpDown.Value;
                                        looper = 0;
                                        while(looper < (int)iterationsUpDown.Value && (x1
* x1) + (y1 * y1) < 4)
                                        {
                                                tempx =(x1 * x1) - (y1 * y1) + x;
                                                y1 = 2 * x1 * y1 + y;
```

```
                                        x1 = tempx;
                                        looper++;
                                }
                                double percent = (looper /
(double)(iterationsUpDown.Value));
                                int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(128-
val)),(Math.Abs(128-val)),Math.Abs((32-val))/*,Math.Abs((val))*/));
                                y += linearPtoY;

                        }
                        x += linearPtoX;
                }

                        pictureBox1.BackgroundImage = original_bitmap;

        }

        #endregion

        #region JuliaTest

        private void DrawJuliaTest()
        {

                Bitmap original_bitmap = new
Bitmap(pictureBox1.Width,pictureBox1.Height);
                pictureBox1.Image = original_bitmap;
                double tempx, xmin, xmax, ymin, ymax = 0.0;
                double x1 = (double)ReZzeroUpDown.Value;
                double y1 = (double)ImZzeroUpDown.Value;
                int looper, z, w = 0;
                double linearPtoX, linearPtoY = 0.0;
                xmin = (double)xminUpDown.Value;
                ymin = (double)yminUpDown.Value;
                xmax = (double)xmaxUpDown.Value;
                ymax = (double)ymaxUpDown.Value;
                linearPtoX = (xmax - xmin) / original_bitmap.Width;
                linearPtoY = (ymax - ymin) / original_bitmap.Height;

                for(z = 0; z < original_bitmap.Width; z++)
                {

                        for(w = 0; w < original_bitmap.Height; w++)
                        {
                                x1 = (double)xminUpDown.Value + linearPtoX*z;
                                y1 = (double)ymaxUpDown.Value - linearPtoY*w;
                                looper = 0;
                                while(looper < iterationsUpDown.Value && (x1 *
x1) + (y1 * y1) < 4)
                                {
                                        tempx = (x1 * x1) - (y1 * y1) + thisCRe;
                                        y1 = 2 * x1 * y1 + thisCIm;
                                        x1 = tempx;
                                        looper++;
                                }
                                double percent = (looper /
(double)(iterationsUpDown.Value));
                                int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(128-
val)),(Math.Abs(128-val)),Math.Abs((32-val))/*,Math.Abs((val))*/));
                        }
                }
                pictureBox1.BackgroundImage = original_bitmap;

        }

        #endregion

        #region CosSet

        public void DrawCosSet()
        {
                double Sx = -3.0;
                double Sy = -3.0;
```

```
                        double Fx = 3.0;
                        double Fy = 3.0;
                        Color[] cs = new Color[256];
                        Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / this.Width;
                        linearPtoY = (ymax - ymin) / this.Height;
                        x = xmin;
                        for(z = 0; z < this.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < this.Height; w++)
                                {
                                        x1 = 0;
                                        y1 = 0;
                                        looper = 0;
                                        while(looper < 100 && Math.Sqrt((x1 * x1) + (y1 *
y1)) < 2)
                                        {
                                                tempx = Math.Cos(x1)*Math.Cosh(y1) + x;
                                                y1 = -(Math.Sin(x1)*Math.Sinh(y1) + y);
                                                x1 = tempx;
                                                looper++;
                                        }
                                        double percent = looper / (100.0);
                                        int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                        y += linearPtoY;
                                }
                                x += linearPtoX;
                        }
                        bmap = original_bitmap;
                        pictureBox1.BackgroundImage = (Image)bmap;
                }
                #endregion

                #region SinSet

                public void DrawSinSet()
                {
                        double Sx = -3.5;
                        double Sy = -3.5;
                        double Fx = 3.5;
                        double Fy = 3.5;
                        Color[] cs = new Color[256];
                        Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / this.Width;
                        linearPtoY = (ymax - ymin) / this.Height;
                        x = xmin;
                        for(z = 0; z < this.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < this.Height; w++)
                                {
                                        x1 = 0;
                                        y1 = 0;
                                        looper = 0;
                                        while(looper < 100 && Math.Sqrt((x1 * x1) + (y1 *
y1)) < 2)
                                        {
```

```
                                        tempx = Math.Sin(x1)*Math.Cosh(y1) + x;
                                        y1 = Math.Cos(x1)*Math.Sinh(y1) + y;
                                        x1 = tempx;
                                        looper++;
                                }
                                double percent = looper / (100.0);
                                int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                y += linearPtoY;
                        }
                        x += linearPtoX;
                }
                bmap = original_bitmap;
                pictureBox1.BackgroundImage = (Image)bmap;
        }
        #endregion

        #region ExpSet

        public void DrawExpSet()
        {
                double Sx = -3.1;
                double Sy = -3.1;
                double Fx = 3.1;
                double Fy = 3.1;
                Color[] cs = new Color[256];
                Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                pictureBox1.Image = original_bitmap;
                double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                int looper, z, w = 0;
                double linearPtoX, linearPtoY = 0.0;
                xmin = Sx;
                ymin = Sy;
                xmax = Fx;
                ymax = Fy;
                linearPtoX = (xmax - xmin) / this.Width;
                linearPtoY = (ymax - ymin) / this.Height;
                x = xmin;
                for(z = 0; z < this.Width; z++)
                {
                        y = ymin;
                        for(w = 0; w < this.Height; w++)
                        {
                                x1 = 0;
                                y1 = 0;
                                looper = 0;
                                while(looper < 100 && Math.Sqrt((x1 * x1) + (y1 *
y1)) < 2)
                                {
                                        tempx = Math.Cosh(x1)*Math.Cosh(y1) +
Math.Sinh(x1)*Math.Cosh(y1) + x;
                                        y1 = Math.Cosh(x1)*Math.Cosh(y1)
+Math.Sinh(x1)*Math.Sin(y1) + y;
                                        x1 = tempx;
                                        looper++;
                                }
                                double percent = looper / (100.0);
                                int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                y += linearPtoY;
                        }
                        x += linearPtoX;
                }
                bmap = original_bitmap;
                pictureBox1.BackgroundImage = (Image)bmap;
        }

        #endregion

        #region CoshSet

        public void DrawCoshSet()
        {
```

```
                        double Sx = -3.3;
                        double Sy = -3.3;
                        double Fx = 3.3;
                        double Fy = 3.3;
                        Color[] cs = new Color[256];
                        Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / this.Width;
                        linearPtoY = (ymax - ymin) / this.Height;
                        x = xmin;
                        for(z = 0; z < this.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < this.Height; w++)
                                {
                                        x1 = 0;
                                        y1 = 0;
                                        looper = 0;
                                        while(looper < 100 && Math.Sqrt((x1 * x1) + (y1 *
y1)) < 2)
                                        {
                                                tempx = Math.Cosh(x1)*Math.Cos(y1) + x;
                                                y1 = Math.Sinh(x1)*Math.Sin(y1) + y;
                                                x1 = tempx;
                                                looper++;
                                        }
                                        double percent = looper / (100.0);
                                        int val = ((int)(percent * 255));

               original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                        y += linearPtoY;
                                }
                                x += linearPtoX;
                        }
                        bmap = original_bitmap;
                        pictureBox1.BackgroundImage = (Image)bmap;
                }
                #endregion

                #region SinhSet

                public void DrawSinhSet()
                {
                        double Sx = -6.0;
                        double Sy = -6.0;
                        double Fx = 6.0;
                        double Fy = 6.0;
                        Color[] cs = new Color[256];  //Struct System.Drawing.Color -
Represents an ARGB color
                        Bitmap original_bitmap = new Bitmap(this.Width,this.Height);
                        pictureBox1.Image = original_bitmap;
                        double x, y, x1, y1, tempx, xmin, xmax, ymin, ymax = 0.0;
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = Sx;
                        ymin = Sy;
                        xmax = Fx;
                        ymax = Fy;
                        linearPtoX = (xmax - xmin) / this.Width;
                        linearPtoY = (ymax - ymin) / this.Height;
                        x = xmin;
                        for(z = 0; z < this.Width; z++)
                        {
                                y = ymin;
                                for(w = 0; w < this.Height; w++)
                                {
                                        x1 = 0;
                                        y1 = 0;
                                        looper = 0;
```

```
                                        while(looper < 100 && Math.Sqrt((x1 * x1) + (y1 *
y1)) < 2)
                                        {
                                                tempx = Math.Sinh(x1)*Math.Cos(y1) + x;
                                                y1 = Math.Cosh(x1)*Math.Sin(y1) + y;
                                                x1 = tempx;
                                                looper++;
                                        }
                                        double percent = looper / (100.0);
                                        int val = ((int)(percent * 255));

        original_bitmap.SetPixel(z,w,System.Drawing.Color.FromArgb((Math.Abs(60-
val)),(Math.Abs(255-val)),Math.Abs(128-(val)),Math.Abs((val))));
                                        y += linearPtoY;
                                }
                                x += linearPtoX;
                        }
                        bmap = original_bitmap;
                        pictureBox1.BackgroundImage = (Image)bmap;
                }
                #endregion

                #region Real Time Julia

                private void DrawRealTimeJulia()
                {
                        Bitmap original_bitmap2 = new Bitmap(200,100);
                        Bitmap original_bitmap3 = new Bitmap(200,100);
                        pictureBox2.Image = original_bitmap2;
                        pictureBox3.Image = original_bitmap3;
                        double tempx, xmin, xmax, ymin, ymax = 0.0;
                        double x1 = 0;
                        double y1 = 0
                        int looper, z, w = 0;
                        double linearPtoX, linearPtoY = 0.0;
                        xmin = -2.1;
                        ymin = -2.1;
                        xmax = 2.1;
                        ymax = 2.1;
                        linearPtoX = 0.021;
                        linearPtoY = 0.021;

                        for(z = 1; z < 200; z++)
                        {

                                for(w = 1; w < 100; w++)
                                {
                                        x1 = (double)xminUpDown.Value + linearPtoX*z;
                                        y1 = (double)ymaxUpDown.Value - linearPtoY*w;
                                        looper = 0;
                                        while(looper < 12 && (x1 * x1) + (y1 * y1) < 4)
                                        {
                                                tempx = (x1 * x1) - (y1 * y1) + thisCRe;
                                                y1 = 2 * x1 * y1 + thisCIm;
                                                x1 = tempx;
                                                looper++;
                                        }


        original_bitmap2.SetPixel(z,w,System.Drawing.Color.FromArgb(20*looper,20*looper
,20*looper));
                                        original_bitmap3.SetPixel(199-z,99-
w,System.Drawing.Color.FromArgb(20*looper,20*looper,20*looper));

                                }

                        }

                }


                #endregion

                public class TempNumericals

                {
                        public static double NumericalReC;
```

```
                        public static double NumericalImC;
                        public static double scaleXoverY;
                        public static double RightBorder;
                        public static double LeftBorder;
                        public static double UpperBorder;
                        public static double LowerBorder;
                        public static double ReCstatic;
                        public static double ImCstatic;

                }



                TempNumericals myNumericals  = new TempNumericals();

                private void pictureBox1_MouseEnter(object sender, System.EventArgs e)
                {
                        this.pictureBox1.Focus();
                }

                private void JliaChkBx_CheckChanged(object sender, System.EventArgs e)
                {
                        if(RealTimeJliaChkBx.Checked==false)
                        {
                                pictureBox3.Visible=false;
                                pictureBox2.Visible=false;
                        }
                        else
                        {
                                pictureBox3.Visible=true;
                                pictureBox2.Visible=true;
                        }
                }

                private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
                {
                        double ReCtoPrint = (((double)xminUpDown.Value +
(((double)Math.Abs(xminUpDown.Value)+(double)Math.Abs(xmaxUpDown.Value))/
pictureBox1.Width)*e.X));
                        double ImCtoPrint = (((double)ymaxUpDown.Value -
(((double)Math.Abs(yminUpDown.Value)+(double)Math.Abs(ymaxUpDown.Value))/
pictureBox1.Height)*e.Y));
                        int thisX = e.X;
                        int thisY = e.Y;

                        statusBarPanel2.Text = "Re(C)= " + ReCtoPrint + " Im(C)= " +
ImCtoPrint;
                        TempNumericals.NumericalReC = ReCtoPrint;
                        TempNumericals.NumericalImC = ImCtoPrint;
                        thisCRe = (double)TempNumericals.NumericalReC;
                        thisCIm = (double)TempNumericals.NumericalImC;



        if(this.RealTimeJliaChkBx.Checked==true&&mandelBool==true&&e.X==e.X|e.Y==e.Y)
                        {DrawRealTimeJulia();}
                        else
if(RealTimeJliaChkBx.Checked==false&&rdioMandel.Checked==true&&mandelBool==true&&e.But
ton.Equals(MouseButtons.Left))
                        {
                                pFinish.X = e.X;
                                pFinish.Y = e.Y;
                                Pen myPen = new Pen(Color.Gainsboro, 1);

                                Bitmap bq = original_bitmap;
                                Point temp = new Point(e.X, pStart.Y + (e.X -
pStart.X));
                                Rectangle rect = Rectangle.FromLTRB(pStart.X, pStart.Y,
temp.X, temp.Y);
                                Bitmap bp = new Bitmap(temp.X+1,temp.Y+1);
                                Graphics g = Graphics.FromImage((Image)bp);
                                g.DrawImage((Image)bp, 0, 0);
                                g.DrawRectangle(myPen, rect);
                                pictureBox1.Image = (Image)bp;
                                pictureBox1.Update();
                                TempNumericals.LeftBorder  = rect.Left;
```

```
                        TempNumericals.RightBorder = rect.Right;
                        TempNumericals.LowerBorder = rect.Top;
                        TempNumericals.UpperBorder = rect.Bottom;

                }

                else
if(rdioJulia.Checked==true&&juliaBool==true&&e.Button.Equals(MouseButtons.Left))
                {
                        pFinish.X = e.X;
                        pFinish.Y = e.Y;
                        Pen myPen = new Pen(Color.Gainsboro, 1);

                        Bitmap bq = bmap;
                        Point temp = new Point(e.X, pStart.Y + (e.X -
pStart.X));
                        Rectangle rect = Rectangle.FromLTRB(pStart.X, pStart.Y,
temp.X, temp.Y);
                        Bitmap bp = new Bitmap(temp.X+1,temp.Y+1);
                        Graphics g = Graphics.FromImage((Image)bp);
                        g.DrawImage((Image)bq, 0, 0);
                        g.DrawRectangle(myPen, rect);

                        pictureBox1.Image = (Image)bp;
                        pictureBox1.Update();
                        TempNumericals.LeftBorder  = rect.Right;
                        TempNumericals.RightBorder = rect.Left;
                        TempNumericals.LowerBorder = rect.Top;
                        TempNumericals.UpperBorder = rect.Bottom;
                }


        }

        #region notInterestingInZoom


        private void PictureBox1_MouseDown(object sender, MouseEventArgs e)
        {
                pStart.X = e.X;
                pStart.Y = e.Y;
        }

        private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
        {
                pFinish.X = e.X;
                pFinish.Y = e.Y;
        }

        private void  xmaxUpDown_GotFocus (object sender, System.EventArgs e)
        {
                xmaxUpDown.Select(0,18);
        }

        private void  ymaxUpDown_GotFocus (object sender, System.EventArgs e)
        {
                ymaxUpDown.Select(0,18);
        }

        private void  yminUpDown_GotFocus (object sender, System.EventArgs e)
        {
                yminUpDown.Select(0,18);
        }

        private void  xminUpDown_GotFocus (object sender, System.EventArgs e)
        {
                xminUpDown.Select(0,18);
        }


        private void xminUpDown_ValueChanged(object sender, System.EventArgs e)
        {
                double scaleXoverY = Math.Abs(((double)(xmaxUpDown.Value)-
(double)(xminUpDown.Value))/((double)(ymaxUpDown.Value)-(double)(yminUpDown.Value)));
                XoverYlabel.Text = scaleXoverY.ToString();
                XoverYlabel.Update();
        }
```

```
		private void yminUpDown_ValueChanged(object sender, System.EventArgs e)
		{
			double scaleXoverY = Math.Abs(((double)(xmaxUpDown.Value)-
(double)(xminUpDown.Value))/((double)(ymaxUpDown.Value)-(double)(yminUpDown.Value)));
			XoverYlabel.Text = scaleXoverY.ToString();
			XoverYlabel.Update();
		}

		private void xmaxUpDown_ValueChanged(object sender, System.EventArgs e)
		{
			double scaleXoverY = Math.Abs(((double)(xmaxUpDown.Value)-
(double)(xminUpDown.Value))/((double)(ymaxUpDown.Value)-(double)(yminUpDown.Value)));
			XoverYlabel.Text = scaleXoverY.ToString();
			XoverYlabel.Update();
		}

		private void ymaxUpDown_ValueChanged(object sender, System.EventArgs e)
		{
			double scaleXoverY = Math.Abs(((double)(xmaxUpDown.Value)-
(double)(xminUpDown.Value))/((double)(ymaxUpDown.Value)-(double)(yminUpDown.Value)));
			XoverYlabel.Text = scaleXoverY.ToString();
			XoverYlabel.Update();
		}

		private void  ReZzeroUpDown_GotFocus (object sender, System.EventArgs
e)
		{
			ReZzeroUpDown.Select(0,18);
		}

		private void  ImZzeroUpDown_GotFocus (object sender, System.EventArgs
e)
		{
			ImZzeroUpDown.Select(0,18);
		}

		private void  iterationsUpDown_GotFocus (object sender,
System.EventArgs e)
		{
			iterationsUpDown.Select(0,18);
		}

		private void menuItem8_Click(object sender, System.EventArgs e)
		{
			About myAbout = new About();
			myAbout.ShowDialog();
		}

		private void btnClear_Click(object sender, System.EventArgs e)
		{
			ClearScreen();
		}

		private void btnDrawFractal_Click(object sender, System.EventArgs e)
		{
			if(rdioMandel.Checked==true)
			{DrawMandel(); mandelBool=true; clickTimes=1;}
			else if(rdioJulia.Checked==true)
			{DrawJuliaTest(); juliaBool=true; mandelBool=false;}

		}


		private void pictureBox1_DoubleClick(object sender, System.EventArgs e)
		{
			JuliaProperties myJulia = new JuliaProperties();

			myJulia.JliaNmrcUDCRe.Value =
(decimal)TempNumericals.NumericalReC;
			myJulia.JliaNmrcUDCIm.Value =
(decimal)TempNumericals.NumericalImC;


			if(mandelBool==true&&myJulia.ShowDialog()==DialogResult.OK)
			{
				xminUpDown.Value = Decimal.Round(-2.1M,1);
```

```
                yminUpDown.Value = Decimal.Round(-2.1M,1);
                xmaxUpDown.Value = Decimal.Round(2.1M,1);
                ymaxUpDown.Value = Decimal.Round(2.1M,1);
                xminUpDown.Update();
                yminUpDown.Update();
                xmaxUpDown.Update();
                ymaxUpDown.Update();
                iterationsUpDown.Value = 100;
                ReZzeroUpDown.Value = 0;
                ImZzeroUpDown.Value = 0;

                thisCRe = (double)myJulia.JliaNmrcUDCRe.Value;
                thisCIm = (double)myJulia.JliaNmrcUDCIm.Value;
                TempNumericals.ReCstatic = thisCRe;
                TempNumericals.ImCstatic = thisCIm;
                DrawJuliaTest();
                rdioJulia.Checked=true;
                mandelBool=false;
                juliaBool=true;
                clickTimes=1;
        }

}


private void menuItem10_Click(object sender, System.EventArgs e)
{

        if(mySelection.ShowDialog()==DialogResult.OK)
        {
                if(mySelection.radioBttnMandel.Checked==true)
                {DrawMandel();}
                else if(mySelection.radioBttnJulia.Checked==true)
                {DrawJuliaTest();}
                else if(mySelection.radioBttnSinM.Checked==true)
                {DrawSinSet();}
                else if(mySelection.radioBttnCosM.Checked==true)
                {DrawCosSet();}
                else if(mySelection.radioBttnExpM.Checked==true)
                {DrawExpSet();}
                else if(mySelection.radioBttnSinhM.Checked==true)
                {DrawSinhSet();}
                else
                {DrawCoshSet();}
        }
}

private void menuItem11_Click(object sender, System.EventArgs e)
{
        Clipboard.SetDataObject(original_bitmap,true);
}

private void bttnReset_Click(object sender, System.EventArgs e)
{
        xminUpDown.Value = Decimal.Round(-2.1M,1);
        yminUpDown.Value = Decimal.Round(-2.1M,1);
        xmaxUpDown.Value = Decimal.Round(2.1M,1);
        ymaxUpDown.Value = Decimal.Round(2.1M,1);
        xminUpDown.Update();
        yminUpDown.Update();
        xmaxUpDown.Update();
        ymaxUpDown.Update();

        this.statusBarPanel3.Text = "Magnification x1";
}

private void btnFormulaReset_Click(object sender, System.EventArgs e)
{
        iterationsUpDown.Value = 100;
        ReZzeroUpDown.Value = 0;
        ImZzeroUpDown.Value = 0;
}

private void menuItem2_Click(object sender, System.EventArgs e)
{

        if(saveFileDialog.ShowDialog()==DialogResult.OK)
```

```
                {

                }
        }

        private void pictureBox1_Click(object sender, System.EventArgs e)
        {


        }

        private void menuItem2_Click_1(object sender, System.EventArgs e)
        {
                Application.Exit();
        }

        #endregion

        private void menuItem14_Click(object sender, System.EventArgs e)
        {
                double distX  = (double)xmaxUpDown.Value -
(double)xminUpDown.Value;
                double distY  = (double)ymaxUpDown.Value -
(double)yminUpDown.Value;
                double oDistX = (double)xminUpDown.Value;
                double oDistY = (double)yminUpDown.Value;

                Int64 magnification = new Int64();

                magnification = (Int64)Math.Abs((decimal)4.2/(xmaxUpDown.Value-
xminUpDown.Value));


                if(mandelBool==true)
                {
                        xminUpDown.Value   =
(decimal)(TempNumericals.LeftBorder/pictureBox1.Width)*(decimal)distX;
                        xminUpDown.Value  += (decimal)oDistX;

                        xmaxUpDown.Value   =
(decimal)(TempNumericals.RightBorder/pictureBox1.Width)*(decimal)distX;
                        xmaxUpDown.Value  += (decimal)oDistX;

                        yminUpDown.Value   =
(decimal)(TempNumericals.LowerBorder/pictureBox1.Height)*(decimal)distY;
                        yminUpDown.Value  += (decimal)oDistY;

                        ymaxUpDown.Value   =
(decimal)(TempNumericals.UpperBorder/pictureBox1.Height)*(decimal)distY;
                        ymaxUpDown.Value  += (decimal)oDistY;
                        DrawMandel();
                        mandelBool=true;
                        this.statusBarPanel3.Text = "Magnfication x" +
magnification.ToString();
                }

                else if(juliaBool==true)
                {
                        thisCRe = (double)TempNumericals.ReCstatic;
                        thisCIm = (double)TempNumericals.ImCstatic;

                        xminUpDown.Value   =
(decimal)(TempNumericals.LeftBorder/pictureBox1.Width)*(decimal)distX;
                        xminUpDown.Value  += (decimal)oDistX;

                        xmaxUpDown.Value   =
(decimal)(TempNumericals.RightBorder/pictureBox1.Width)*(decimal)distX;
                        xmaxUpDown.Value  += (decimal)oDistX;

                        yminUpDown.Value   =
(decimal)(TempNumericals.LowerBorder/pictureBox1.Height)*(decimal)distY;
                        yminUpDown.Value  += (decimal)oDistY;

                        ymaxUpDown.Value   =
(decimal)(TempNumericals.UpperBorder/pictureBox1.Height)*(decimal)distY;
                        ymaxUpDown.Value  += (decimal)oDistY;
```

```
                                DrawJuliaTest();
                                juliaBool=true;
                                this.statusBarPanel3.Text = "Magnfication x" +
magnification.ToString();
                        }
                        //xmaxUpDown.Value = -xmaxUpDown.Value;

                }

                private void menuItem16_Click(object sender, System.EventArgs e)
                {
                        JuliaProperties myJulia2 = new JuliaProperties();

                        myJulia2.JliaNmrcUDCRe.Value =
(decimal)TempNumericals.NumericalReC;
                        myJulia2.JliaNmrcUDCIm.Value =
(decimal)TempNumericals.NumericalImC;
                        //thisCRe = TempNumericals.NumericalReC;
                        //thisCIm = TempNumericals.NumericalImC;

                        if(mandelBool==true&&myJulia2.ShowDialog()==DialogResult.OK)
                        {
                                xminUpDown.Value = Decimal.Round(-2.1M,1);
                                yminUpDown.Value = Decimal.Round(-2.1M,1);
                                xmaxUpDown.Value = Decimal.Round(2.1M,1);
                                ymaxUpDown.Value = Decimal.Round(2.1M,1);
                                xminUpDown.Update();
                                yminUpDown.Update();
                                xmaxUpDown.Update();
                                ymaxUpDown.Update();
                                iterationsUpDown.Value = 100;
                                ReZzeroUpDown.Value = 0;
                                ImZzeroUpDown.Value = 0;

                                thisCRe = (double)myJulia2.JliaNmrcUDCRe.Value;
                                thisCIm = (double)myJulia2.JliaNmrcUDCIm.Value;
                                DrawJuliaTest();
                                rdioJulia.Checked=true;
                                mandelBool=false;
                                juliaBool=true;
                        }
                }

                private void pictureBox2_Click(object sender, System.EventArgs e)
                {

                }

        }

}
```