FASTER GENERATION OF ALGEBRAIC FRACTALS

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE AWARD OF THE DEGREE OF

MASTER OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

TO

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA - 147 001

SUPERVISOR

Shri. Himanshu Aggarwal
Ms. Kanwalinderjit Kaur

SUBMITTED BY

LAKHWINDER KAUR
REG. NO. ME-129/98(R)/1

DEPARTMENT OF COMPUTER SCIENCE
TECHNICAL TEACHERS' TRAINING INSTITUTE
SECTOR 26, CHANDIGARH-160019
2000

CERTIFICATE

Certified that this Thesis entitled "Faster generation of algebraic fractals" being submitted by Lakhwinder kaur in the partial fulfillment of the requirements, for the award of degree of Master of Engineering (Computer Science and Engineering) of Thapar Institute of Engineering and Technology, Patiala is a record of candidate's own work carried under our supervision and guidance.

This thesis has not been submitted to any other university or institute for the award of any degree.

SUPERVISORS

(Himanshu Aggarwal)
Lecturer
Department of
Computer Science,
T.T.T.I., Chandigarh.

(Kanwalinderjit Kaur)
Lecturer
RCC, Chandigarh
9/2/2K

(V.P. PURI)
Prof. IMCO &
Head, Department
of Computer Science,
T.T.T.I., Chandigarh.

ACKNOWLEDGEMENTS

The successful completion of this work is a subject of great joy. I would like to express my gratitude to all those who have been of great help during this marathon effort.

I am thankful to my supervisor Mr. Himanshu Aggarwal, *Lecturer TTTI Chandigarh* for his valuable suggestions, support and encouragement throughout my M.E. course as well as during this thesis work.

I wish to thank Miss Kanwalinderjit Kaur, *Lecturer RCC Chandigarh,* whose advice and guidance has enabled me to complete this work.

I wish to express my gratitude to Dr. Kehar Singh, *Ex-Professor & Head, Department of mathematics, GNDU Amritsar,* for continuous encouragement and motivation at each stage.

I am grateful to the Principal, *Technical Teachers' Training Institute, Chandigarh* for providing the opportunity to carry out the present thesis work.

Further, I would like to express my sincere thanks to *Prof.* V.P. Puri, *Head, Department of Computer Sc.*, and Dr. (Mrs.) Renu Vig, *Assistant Professor, Department of Computer Sc.*, for their critical suggestions during this period.

Besides, I am thankful to all the faculty and staff members of *Department of Computer Sc.* for their intellectual support throughout my stay at TTTI, Chandigarh.

Last but not least, I am also grateful to my friends, colleagues and family members who helped me to sustain my determination to accomplish this work. Without their help, support and valuable suggestions it would be extremely difficult to finish this work.

(Lakhwinder Kaur)

Abstract

Fractals, its concepts and applications have been widely studied in most of the natural sciences. In this thesis, various aspects related to fractals are discussed, with a special focus on *algebraic fractals*. The biggest constraint in creating the fractal images is high execution time, to reduce that a new technique for faster generation of connected algebraic fractals have been proposed, analyzed and compared with the conventional technique in terms of execution time required for generation of fractals. The comparison clearly shows that the new technique is more efficient than that of conventional one as it requires much less time. Also three techniques proposed by Rojas[24] for Mandelbrot set have been reviewed and suggested that these techniques can also be applied to all other algebraic fractals.

Table of Contents

CHAPTER 1

Introduction

1.1 Computer Graphics

Computer graphics is one of the most exciting and rapidly growing field in computer science. As the volume of information increases a problem arises: how can this information be efficiently and effectively transferred between the machine and human? This machine can easily generate tables of hundreds of pages long. Such a printout is useless unless reader does not have a time to understand it. Computer graphics strikes directly at this problem. A diagram or a graph can replace huge table of numbers and reader can understand it at a glance

Computer graphics has become an important discipline within Computer Science and Engineering. We find that anything that involves making shapes or pictures through computers can qualify to be included in computer graphics. Data are presented visually through shapes, colors and textures rather than by tables of numbers and words. Words and numbers are replaced wherever possible by pictures, because eye brain system is better at recognizing and interpreting visual representations .The ability to converse pictorially with a computer is revolutionizing the way computers being used in all areas.

Computer graphics is not limited to, work on animation, antialiasing, computer aided design, geometry, the design and analysis of graphics algorithms, geometric

modeling, graphics hardware, graphic languages, graphic user interfaces, graphic systems, human factors, image synthesis, interaction techniques, lightning models, shading applications and rendering . Today we find computer graphics used in such diverse areas as business, industry, hospitals, art, entertainment, research, education and training etc.

1.2 Fractals

Recently many researchers have concentrated their work on modeling complex objects using mathematical and statistical tools popularly known as 'Fractal Geometry'. In computer graphics we use fractal geometry to generate displays of natural objects and visualization of various mathematical and physical systems.

In the past two decades, Fractals and its concepts have become central tools in most of the natural sciences, such as Physics, Chemistry, Biology, Geology, Material Science etc. Word '*Fractal*' is derived from a Latin word '*Fractus*,' which means irregular. The term Fractal was coined by Benoit B. Mandelbrot in 1975. The concept of fractals exploded into the consciousness of the scientists, when Mandelbrot's book 'The Fractal Geometry of Nature' was published in 1982[1].

Fractals offer an extremely compact method for describing the objects and their formations. Many structures have an underlying hardware regularity, known as self-similarity. If one examines these objects at different sizes and scales, he repeatedly encounters the same fundamental elements. This repetitive pattern defines the fractal dimension of the structure.

Fractals are used to provide 'naturalistic' shapes for objects such as clouds, mountains, coastlines, mountains, grass, plants, trees and fire. Connection between fractals and the structure of natural objects is so close that it seems almost every thing of nature can be simulated using fractals. Fractal geometry represents objects more gracefully than does Euclidean methods, which do not realistically model natural objects, such as mountains, clouds etc., and natural phenomenon such as Brownian motion and cantor dust, which having irregular and fragmented features. Standard shapes of Euclidean geometry such as a straight line, polygons, conics, quadric surfaces and polyhedra, which, are characterized by having integer dimensionality, cannot be used as building blocks for complex objects.

Fractals are expressed not in primary shapes but in algorithms, and set of mathematical or statistical procedures. The basic principal of generating fractals involves repetitive application of such procedures over a domain of space.

Several techniques for generating fractals have been developed . Two major techniques popularized by mandelbrot's book [1] have attracted wide interest.

- Koch construction.
- Function iteration in the complex domain.

According to Mandelbrot's generalization, the Koch construction consists of recursively replacing edges of an arbitrary polygon (called the initiator) by an open polygon (the generator) reduced and displaced so as to have the same end points as those of interval being replaced.

In contrast the method of function iteration refers to notions of complex analysis. The main idea is to analyze sequence of numbers $\{x_n\}$ generated by the formula $x_{n+1}=f(x_n)$

where x is a complex function. The fractal, called Julia set is a set invariant with respect to 'f'. The sequences of points originating outside the fractal may gradually approach it in which case the Julia set is said to be attractor of the process 'f' or they may diverge from the fractal and the set is then called a repeller of `f'.

1.3 Salient features of Fractals

A fractal is constructed from a 'collage' of transformed copies of itself and thus inherently possesses the following properties:

1.3.1 Self-Similarity:

Self-similar objects have parts that are scaled-down versions of the entire object. Starting with an initial shape, we construct the object subparts by applying a scaling parameter to the overall shape. We can use same scaling factor for all subparts, or can use different scaling factors for different scaled-down subparts. The parts then have the same statistical properties.

Self-similarity of a fractal is best illustrated by the Von Koch snowflake example [34], starting with a line segment made up of four sub-segments a, b, c and d (figure 1.1).

Fig. 1.1

Figure 1,2,3 and 4 (of fig. 1.1) showing Von Koch curves of order 1,2,3,4.

In figure 2 each of these sub-segment is replaced with the original figure 1 and figure 3 each of the sub-segment is replaced with either figure 1, the original or with figure 2 the immediate preceding [34]. The result after 2^n steps in case-1 is same as the result after n steps in case-2. If this process is repeated infinitely, the result is said to be self similar, i.e. entire object is similar to a subportion of itself. We are not talking of self similarity from pure mathematical or statistical point of view, any object that is not exactly self similar may still seem fractal i.e. the objects that look like them self when scaled down can still be termed fractals.

Natural objects are simulated with procedures that theoretically repeat an infinite number of times. Statistically self-similar fractals are used to model trees, shrubs and plants etc.

Self-affine fractals have parts that are formed with different scaling parameters, S_x, S_y, S_z in different co-ordinate directions. Random variables can be included to obtain statistically self-affine fractals. Terrain, water, and clouds are typically modeled with statistically self-affine construction method. Invariant fractal sets are formed with non-linear transformations. This class of fractals include the Mandelbrot set, which is a self-squaring fractal, formed with squaring function formed in complex space.

1.3.2 Infinite Scalability:

Smooth curves are one-dimensional objects, whose length defined between two points. But fractal curve contains an infinite detail at each point along the curve, so we cannot say what its length is. But as we come closer, more details emerge. When one zooms in still further, the length of the fractal curves goes longer and longer, for example outline of a mountain shows more variations as we go closer to it. Similar types of

curves are for coastlines and edges of clouds .If we see coastline from a satellite it has certain level of ruggedness, caused by bays, inlets and peninsulas. As one flies in for a closer, a bay is now seemed to have a certain ruggedness of its own that was not visible before. As we continue to zoom in more and more, smaller rocks and pebbles seem to produce about the same level of ruggedness.

1.3.3 Fractional Dimension:

We can describe the amount of variation in the object detail with a number called the *fractal dimension*. It is a measure of the roughness or fragmentation of the object. It is difficult to calculate. The fractal dimension, also referred to, as fractional dimension is the basis for the name "fractal". A line is one dimension and a plane is a two dimension. According to traditional geometry, line is still one dimensional, even though it can be used to fill up the surface of a plane without crossing over it. Intuitively it is felt that, it should have dimension two. If this is the case then it can be concluded that there was a transformation when the line ceased to be one-dimensional. According to mathematician, Felix Housdroff[1], a curve of infinite length, which fits into a finite region of a plane, must have a dimension between one and two. Mandelbrot devised a method for computation of the fractional dimension for such curves [31] .the detailed variation can be described with a number D i.e. fractal dimension.

An expression for the fractal dimension of a self-similar fractal can be given by

$$(\text{Number of segments})^{(1/\text{fractal dimension})} = 1/\text{Scaling factor}$$

Let S is a scaling factor, n is the no of subparts or number of subdivisions of a unit straight-line segment, a square and a cube. Solving this expression for D, we have

$$D = \ln(n) / \ln(1/S)$$

A line segment has one dimension. If we divide the line segment into N equal parts, each part looks like scaled by a factor of N as shown below:

$N^{1/D} = N$ i.e. D=1

A square has two dimensions. If we divide a square into N equal parts, each part look like the original scaled down by a factor of $N^{(1/2)}$ as shown below:

$N^{(1/D)}=N^{(1/2)}$ i.e. D=2

A cube has dimension 3. If we divide a cube into N equal parts, each part look like the original when scaled down by a factor of $N^{(1/3)}$ as shown below:

$N^{(1/D)}=N^{(1/3)}$ i.e. D=3

When Von Koch snowflake is divided into four pieces each resulting piece look like the original scaled down by a factor of three or scaling factor 1/3, then we have:

[a] [b]

First iteration in the generation of Von Koch curve

$$4^{(1/D)} = 3$$

or. $D = \ln(4)/\ln(3)$

or. $D = 2.26$

So, Fractal objects cannot be represented with Euclidean geometry methods as fractal curves cannot be described as one-dimensional or two-dimensional shapes. Such curves represented in a two-dimension coordinate system, can only be described mathematically with fractional dimension, between one and two. When a fractal curve is described in three–dimensional space, it has fractional dimension between one and three. Similar types of curves are for coastlines and edges of clouds.

1.4 Types of Fractals

Basically fractals are of three types:

i) Algebraic Fractals

ii) Stochastic Random Fractals

iii) Geometric Fractals

1.4.1 Algebraic Fractals

These types of fractals are the results of repetitive application of a specified transformation function to a point in a specified region of space. Mandelbrot set and Julia sets are the most famous algebraic fractal curves. The mathematics of iterations of algebraic functions has been studied for decades. However beauty and complexity of patterns resulting from such iterative calculations has only recently been explored in detail, due to advances in computer Graphics.

Mandelbrot[1] has been largely responsible for extending the theoretical and graphical representation of iterative functions as a new class of geometry called fractal

geometry. It was who first used the transformation function $f(z) = z^2+c$ to produce very interesting patterns.

1.4.2 Stochastic fractals

These fractals are generated by random processes [32,33]. These fractals can be generated by random recursively replacing each segment in a random elbow with a smaller random elbow, until some stopping criteria is met. As shown in fig. (1.2), suppose that we have a line segment L, with end points a and b. The segment L is replaced by the two segments from a to c and c to b. For a fractal curve, c is a randomly chosen point along the perpendicular bisector s of L. Where S passes through the midpoint m of segment L and perpendicular to it. Any point c along S, for some value of t, has the following parametric form[34]:

$$C=(m_x-(b_y-a_y)*t,m_y-(b_x-a_x)*t)$$

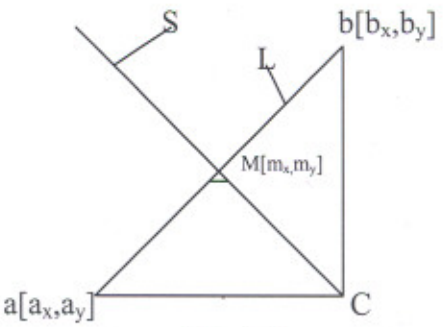


Fig. 1.2

1.4.3 Geometric Fractals

These types of fractals are generated by geometric patterns. For their construction we should have a generator and an initiator. In this process each segment of the initiator is

replaced by the generator, reduced in size and displaced so that the end points coincide with those of the segment being replaced by a scaled copy of the generator. This process repeats for the particular number of iterations. One of the simplest geometric fractal is Von Koch's curve. We use initiator and a generator as shown in fig.1.3. It is initiated with horizontal line of length of one unit (Fig. 1.3(a)). To create the curve of first order C_0, divide the line into three segments, while replacing the middle section with a bump having sides of length 1/3 (Fig. 1.3(b)). Same process is applied to each line segment of curve c1 to get the second order curve. So each segment is increased in length by a factor of 4/3, the total curve length is 4/3 larger. In general, to have c_{i+1} from c_i, place a bump an every one of its segments. So Ci has total length $(4/3)^i$, which increases with I as more iterations are done as shown in fig. 1.3(c). As 'i' approch towards infinity, the length of the curve also becomes infinite, but the curve remains confined to a finite area.

[a] [b]

[c]

Initiator(a) and generator (b) for the Von Koch curve

Fig. 1.3

1.5 Fractal generation procedure

A fractal object can be generated by repeated applying a specified transformation function to points with in a region of space. The transformation function 'F' generates successive levels of detail with calculation:

$$A_1=F\{Z(A_0)\},\ A_2=F\{Z(A_1)\},\ A_3=F\{Z(A_2)\},\ldots\ldots$$

Where $A_0=(X_0,Y_0,Z_0)$ is a selected initial point. Either deterministic or random generation procedure can be used at each iteration. The transformation function f(Z) can be defined in terms of geometric transformations (translation, rotation, scaling), or it can be set up with non-linear co-ordinate transformations and decision parameters. Transformation function can be applied to an initial set of primitives, like lines, areas, volumes, curves, surfaces, and solid objects, or it can be applied to a specified set of points. As we know, fractal objects have infinite detail at each point, but the transformation function is used only for finite number of times. The amount of the detail included in the final display of the image depends on the number of iterations. performed and the resolution of the display system. The process of generating fractal object is computationally intensive. Because, firstly, the transformation function is used repeatedly, secondly, complex numbers are used and thirdly, calculations are carried out for large number of pixels.

CHAPTER 2

Applications of Fractals

Among the methods available for characterization of complicated artistic mathematical and physical phenomena, fractal geometry is emerging as most important tool in computer graphics. Fractals can be seen as mysterious expressions of beauty representing exquisite preordained shapes that mimic the universe. They show the practical connection between mathematics and mysteries of art. In the recent years only, the beauty of fractals has attracted the wide interest of scientists, engineers, artists, and advertisers etc. We have described briefly the various applications of fractals in the following sections.

2.1 Fractals in modeling natural objects:

As fractals fill up space more efficiently than ordinary smooth curves and fractal surfaces are economic, therefore, they seem to suit nature's purposes. So, fractal geometry representations for natural objects can be commonly applied in many fields to describe and explain the features of natural phenomena because many natural objects and man made processes exhibit irregular surfaces with intricate details and scale invariance. An effective means of generating an irregular surface with irregular motion in a flexible way to allow the solution of such problems as realistically modeling waterfall, rapids or ocean waves, all of which present serious challenges to computer graphic researchers, is only fractals.

A very wide variety of natural phenomena have been modeled with the help of fractal methods. In graphics applications, fractal representations have been used to model terrain's, clouds, water, trees, plants, feathers, fur and various surfaces textures and just to make pretty patterns. In other disciplines, this technique has been found quite successful in rendering the realistic images of stars, rivers, islands, moon craters, in rain fields, variations in stock market, in music in traffic flow and in boundaries of convergence regions for numerical analysis technique[33], because these objects are characteristic in general by no regular feature or simple microscopic structure. So only fractals but no other method can be effective in modeling them.

Rough mountains, terrains are topographical scaled up versions of fracture metallic surfaces. The mountains outlined against the sky continue to have the same jagged shapes as we view it from a closer and closer position. As we near the mountain, the smaller details in the individual ledges and boulders become apparent. Moving even closer we see the outlines of rocks, then the stones and thereafter grains of sand. At each step, the outline reveals more twists and turns. If we take the grains of sand and put them under a microscope, we would again see the same detail repeated down to the molecular level or it can be said that mountains exhibit self-similarity. Similarly clouds have been found to be self-similar across several orders of magnitude. The boundaries of areas of rain definitely have been revealed as fractal and that rainfall occurs in rather irregular bursts. Moreover, rainfall variations over short and long scale are similar pointing to the fact that the temporal structure of rain is also fractal. Study of acid rains [33] through fractals to gain a better insight into phenomenon can definitely make it possible to make important forecasts so that ill effects of acid rain

on Eco-system can be minimized. Similar, shapes describe coastlines, which has bays, inlets, estuaries, rivulets, drainage's and ditches which, all look alike. A computer generated animated sequence has coastlines which are recognizable as real coastlines, can certainly be modeled through a fractal plane. Fractal curves are used to model natural object's boundaries such as shorelines, terrains, clouds, water etc. are typically modeled with fractal surfaces while fractal solids are used to model cloud properties such as water vapor density or temperature within a region of space.

It is only the self-invariance of fractals that exhibit many surprises, which defy general intuition. One can get an instant dead planet landscape with craters, dry rivers, valleys, canals, fatty clouds and with other land features and weather patterns. Hence modeling self-similarity at the same scale seems to be a way to generate models of natural phenomena, which have quite appealing look. The research in development of such techniques which allow the empirical determination of parameters of flexible canonical stochastic models, to fit specific natural object, hold the promise of rich rewards for fractals.

2.2 Fractal in the study of various physical and chemical phenomena:

In physics, for example, concept of fractals has been used to study fluid dynamics, Brownian motion, air flow over a space shuttle, numerical modeling of thunderstorms etc. Scientists have developed methods to study polymers and porous materials such as aerogels[29]. The aerogels find potential applications in industries because they are bad conductors of heat and can be used as good thermal insulators.

2.3 Fractals in study of cracks in metals:

Fractured surfaces can never be joined together perfectly, no matter, how hard is tried out because once the part of any material, e.g., metal bar is broken, the surface do not remain smooth because of bumpiness. Fractal geometry finds wide applications in studying the properties of surfaces in contact, because one of the simple but powerful consequences of the fractal geometry of surface is that, surfaces in contact do not touch every where because of roughness.

2.4 Fractals in creations of maps in geography:

As fractal Brownian motion is self similar, an interesting application allows the imaginary coastlines on a map to be created from fractal polylines which look so real that those of us who are ignorant about in geography would have difficulty in arguing that the coastlines of map have not been traced but are computer generated.

2.5 Fractals in study of space science:

The "cantor dust"[32] is a close abstraction of the structure of the universe, which has a lot of void. Recent work indicates that galaxies are not uniformly distributed but they form a sponge like network with a fractal dimension of 1.2. The planetary structures surrounded by satellite structures are images generated through self-squared function of fractals.

2.6 Protein modeling and stereospecific viewing of molecular structure through fractals:

Recent studies on protein molecules show that they possess fractal structure. Hemoglobin molecules have also been found to possess quite rough structure. The surface of poliovirus likewise also has been found to be a fractal.

2.7 Fractals in studying physiological processes of microscopic life forms:

"Biomorphs"[15] are mathematically created creatures which inhabit the complex plane though they resemble microscopic organisms that can be imagined as flourishing in a drop of pond water. The creation shows mitosis of biomorphs or simple life forms like Amoebae, Euglena etc. are mathematically extensions of fractals. Hence use of fractals can further be widened to study incredible variety of physiological processes not only in microscopic life forms but also in plants and other aquatic life forms like snails etc.

2.8 Fractal in study of complex irregular and discontinuous motions:

Although various effective techniques have been developed for creating a series of images of a scene in which smooth and continuous motions of objects in the scene are depicted but to create complex irregular motions such as path of lightening bolt or motion of leaf of some skinny bush of brambles or corn in wind or unfolding of crumpled piece of paper etc, Brownian motion of fractals or stochastic technique can also be the best solution.

2.9 Medical applications of fractals:

Medical applications also make extensive use of fractals in studying physical functions, to design artificial limbs and to plan and practice surgery. Branching of blood vessels was found to be fractal by Mandelbrot[1]. It is nature's arrangement only that huge surface area constituted by the blood circulatory system has been squeezed into a limited volume just as weird Koch curve squeezes a line of infinite

length into smaller area. After publication of these physiological findings, scientists began to find fractal organization in almost all parts of the body. Urinary system, biliary duct of the liver, the labyrinth of special fibers carrying pulses of electric current to heart muscles, all seemed to be designed through a fractal plan. Anatomical fractals seem to be nature's invention out of necessity because they are areawise "economic". As a matter of fact, nature has no other alternative but to go fractals.

2.10 Fractals in scientific visualization and art:

Fractals are now days widely used both in fine arts and commercial art. Artists use them for designing object shapes and specifying "object motion". "Paintbrush" program is used to produce electronic paintings of various types. In commercial art, in motion pictures, in advertising and television, commercial music videos etc., various scenes or objects can be constructed with the help of fractals. They have also contributed in field of scientific visualization and filmmaking. Scientific visualization has become an important part of research in medicine sciences, and in other domains. Fractals are a great help, as they can be generated using very little information. For drawing fractals of physical shapes or objects various techniques such as fractal Brownian motion, midpoint subdivision, repetitive string rewriting, affine transformation etc. have been evolved. Using ultra-high resolution graphics, it has been possible to generate phenomenally realistic scenes. In fact high dimensional fractals were used by Hollywood cinema world for creating special effects in movies, e.g., the imaginary landscape of outer space in films STARTREK was generated using fractals. In the same way, sea sequences were generated in film ABYSS, and a non-existing jungle was created in the film JURASSIC PARK.

To mimic the desired effects in movies and TV serials, self-similarity or scale invariance of fractals is of great use. For creating a particular scene, say that of landscape, the exact detail may require vast computer memory. However this may be dispensed with using the ideas of fractals. The reconstructed scene may not be exact replica of original landscape in all details but may look convincing enough to be acceptable. This kind of trick used for mimicking the desired effect has been dubbed "Fractal forgery".

Fractal applications are possible in another area, i.e. High Definition (HD) television. A special attachment to a conventional TV set can create details using fractal logic, thus improving the quality of the picture without actually loading more information on the signal. This possibility, originally suggested a few years ago, is yet to materialize.

2.11 Fractal compression:

"Fractal compression" is latest development in the field of compression technology, which can change the assumptions behind lossy and lossless compressions. Fractal compression was invented by mathematicians Bransley and sloan [37]. Through this technology it would be possible to compress an image upto 200:1 or more without any loss of resolution. Fractal images are stored as mathematical formula e instead of bitmaps, so they can be decompressed to resolutions higher and lower than those of the original. The fractal compression can further improve if we have more processing power. Fractal compression is asymmetric system. It takes ages to compress but decompression is quick and its application is ideal for vedio. Another important field which is emerging is called "Fractal image Enhancement" a process that actually adds details missing from an uncompressed image, and related technology is still limited

and a lot of new frontiers are still to be opened to make it suitable for further applications.

CHAPTER 3

Literature Survey

According to picover [2], for characterizing complicated physical and mathematical structures and phenomena, computer graphics can be used to produce visual representations with a spectrum of perspectives. The paper describes algorithms for computer graphics rendering a particular class of chaotic structures created from complex iteration. Mathematical feedback loops similar in spirit to those of Julia set theory [1] have been used to produce the pattern. The spiral forms resulting from the iteration of complex equations $f(\psi) = F(\psi)+\mu$ indeed have shapes of startling intricacy. The graphic experiments done with a variety of accompanying parameters, show the complexity of "transition region" between the convergence and divergence. The term transition region denotes the fact that points inside the bounding shapes have different fate upon iteration those on outside. Author has specially focussed on one small region of complex plane and the resulting maps reveal a new class of shapes.

Klein et al [3] have discussed the boundaries of coexisting attractor basins, as a common source of fractal structures in discrete maps. Chaotic attractors in continuous systems of of ordinary differential equations also have a fractal microstructure. A generation mechanism for self-similar fractal boundaries is proposed, which gives a closer connection between "chaos" and "fractals". The study discusses the connection between continuous dynamical systems and discrete mappings to find a generation

mechanism of nontrivial fractal boundaries. Fractal boundaries arise in non-linear dynamical systems as borderlines separating coexisting attractor basins. Attractor basins are defined as the domain within which all-continuous trajectories or discrete orbits tend to the same limit set as time proceeds. Self-similar non-differential structures have for a long time, resisted reproduction in generic continuous systems, which form the main class of dynamical sitemaps that are of applicational interest.

Bradley[4] has given a broad introduction to the causes and effects of chaos, an interesting and rapidly growing field. The discussion has been done at a level that is between non-linear dynamics and chaos. In a deterministic physical system, chaos show complicated, unpredictable and seemingly random behavior. A system need not be complex, to exhibit chaotic behavior, however, it must be non linear, and if its state evolves in continuous time, it must have more than one degree of freedom.

Phillip [5] has measured warped midgets in the Mandelbrot set using an algorithm that allows the positions of the head, and cardioid atoms (north and south) of any midget to be found, once the cursor on the computer terminal has been placed somewhere inside any midget. Two distortions of midgets exist, one is linear and another is angular distortion. The angle and distance measures of warped midgets from sea horse valley of the Mandelbrot set and from other sea horse valleys of midgets, whether on the spike or on tendrils above atoms, all fall closely together in one part of the north/south plane. Measures of warped midgets from tendrils above the major atoms on the surface of the cardioid fall closely together in another part of the north/south plane. This different way of looking at the Mandelbrot set offers an interesting way of studying the distortions of midgets.

Gujar and Bhavsar[6], have generalized the transformation function $Z \leftarrow Z^2+C$ to $Z \leftarrow Z^{\alpha}+C$ for generating fractals images, where α is any real parameter . A multitude of interesting, intriguing, and rich families of fractals is generated by changing a single parameter α. Direct relationships are observed between α and the visual characteristics of the fractal images in the C-plane. The exponent α can be represented as $\alpha = \pm(\eta + \varepsilon)$, where η and ε are the integer and fractional parts, respectively. It is found that when α is a positive integer number, the resulting image contains lobular structures. The number of major lobes equals (η -1). When α is a negative integer number, the generated fractal image is a planetary structure consisting of overlapping central planets surrounded by satellite structures. The number of satellite structures equals (η +1). A continuous variation of α between two consecutive integers result into a continuous proportional change between two limiting fractal images. Several conjunctures, about the visual characteristics of the images and the value of α, have also been explained by the Authors. Gujar, Bhavasar and Vangala[7] have used the transformation function f(Z): $Z \leftarrow Z^{\alpha} + C$, for generating fractal images in the complex Z-plane. When α is a positive integer, the fractal images has a lobular structure with α major lobes. When α is a negative integer, the image has a planetary configuration consisting of a central planet with $|\alpha|$ major satellite structures. For noninteger values of α, additional embryonic lobular structures proportional in size to the fractional part of α, are observed. Bhavsar, Gujar and Vangala[8] have generated fractals from the self squared transformation functions $Z \leftarrow Z^2 + C$, where Z and C are complex quantities. The process of generating these fractal images being iterative in nature is computationally intensive. The process of

generating these fractal images, being iterative in nature, is computationally intensive. Authors propose three vectorization techniques for generating algebraic fractals from $Z \leftarrow Z^2 + C$, namely use of long vectors, short vectors and short vectors with replenishment is the best.

Shirriff [9] explores the methods of generating fractals from mapping $Z \leftarrow 1/Z^{\alpha} + C$ and discusses the structure of resulting images.Lyapunov exponents and cyclic periodicity show details of these fractals that are obscured by prior escape time technique. The resulting fractals are shown to a hypocycloid shape.

Glynn[10] has presented two versions of the evolution of the Gingerbread Man for the iteration to $Z \leftarrow Z^{\alpha} + C$.One version is asymmetric evolution, as positive real α increases and another version is symmetric evolution. Fractal "growth" has been described that occurs along certain shock lines, a floating object appears while approaching certain positive even integer values of α.

Lakhtkia [11] has described about Julia sets of switced processes. Such processes have significant importance in the study of Dynamical systems in which several free standing processes may be involved,. Let f(Z) and g(Z) both be elemental polynomial processes i.e.,

$$Z_{n+1}=f(z_n)=(Z_n)^p+C_f \qquad (1)$$

And

$$Z_{n+1}=g(Z_n)=(Z_n)^q+C_g \qquad (2)$$

Here Z, C_f, C_g are complex, while $p>1$ and $q>1$ are integers. The processes such as

$$Z_{n+1}=f(g(Z_n)),$$

and

$$Z_{n+1}=\alpha.f(Z_n)+(1-\alpha)g(Z_n)\ ; \quad 0 \le \alpha \le 1 \qquad (4)$$

are seemingly more complicated than either (1) or (2). All four processes from (1) to (4), are polynomial processes. For example, the julia set of process (1) has p-fold symmetry,that of process (2) has q-fold symmetry, that of process(3) has (p+q)-fold symmetry, while that of process (4) has r-fold symmetry where r=max{p,q}. Switched processes are inherently different from the free-standing processes (1) to (4), and may be easily encountered, for example, in control systems where control laws may be specified in piecewise fashion continuing with the use of elemental polynomial processes, a switched process may be of the form:

$$Z_{n+1}=f(z_n)=(Z_n)^p+C_r \qquad |Z_n| \le r_o$$

And

$$Z_{n+1}=g(Z_n)=(Z_n)^q+C_r \qquad |Z_n| > r_o$$

Where r_o act as a buffer between the two free standing processes involved. Rossler and Michelitsch[12] have manipulated the imaginary part of the complex-analytic quadratic equation, by iteration, to produce a set, called "burning ship". The name was assigned to it because of its appearance. It is an analogue to the Mandelbrot set. Its nonanalytic "quasi"-Julia set yields surprising graphical shapes. The real and imaginary parts of function Z^2+C are:

$$\text{Real} \quad : x_{n+1} = x_n^2-y_n^2-c_x$$

and

$$\text{Imaginary} \quad : y_{n+1} =2x_ny_n-c_y$$

Rossler and Michelitsch have used equation:

$$x_{n+1} = x_n^2-y_n^2-c_x$$

and

$$y_{n+1} =2|x_ny_n| - c_y$$

to generate quasi-Julia sets. The word "quasi" is used because the present maps are not analytic.

Kim, Kim and Shin[13] have visualised and exploited the convergence patterns of sequences of partial products of infinite Blaschke products,to generate color graphics images,where a Blaschke product 'f' is an interesting function in itself and plays a very important role in the analytic function theory. For example ,any bounded analytic function 'f' on the unit disc of complex plane ,can be factored as a product of the Blaschke product 'B' carrying all the zeros of 'f' an d another bounded analytic function 'f_o' without zeros having the same superimum norm as 'f'.

Dewdney[14] describes the generation of biomorphs and sea shells etc. Stuedell[15] has presented an informal computer graphics investigation of biomorphs. The graphical entities resemble biological structures, which undergo mitosis (cell division) as the parameters in a mathematical feedback loop are varied. It has been assumed that as one varies the constant C associated to the complex function $Z \leftarrow Z^{\alpha} + C$, the interior region goes through stages which resembles biological mitosis (cell division). In the process, biomorphsin each cell go through various stages of mitosis.

Michael[16] has demonstrated various patterns and produced computer art by using discontinuous functions and alternate q-systems to common fractal methods. Usually fractals are produced by iterating a function employing a small set of mathematical operations (arithmetic and trignometric function), which are defined within the complex number system.Discontinuous functions, such as, Boolean functions,the conditional functions and root-finding methods, such as, Aitken's and Muller's methods have been used to produce fractals in q-systems other than the complex number system.

Bowman[17] has provided a brief tutorial on the simple Iterative Function Systems (IFS) technique and has shown some resulting fractals arising from minor alterations of the input parameters . A simple fractal arising from an IFS could be generated by hand. For example,place three points at the corners of an equilateral triangle on a piece of paper. Then, choose any point with in the triangle and randomly select one of the corners. Draw an imaginary line between the original point and the chosen corner. The next point will be located at the centre of this line. Again, randomly select a corner and previouly found point.Repeat this process several hundred' times. Do not make marks on the paper for first 10 repetitions,but after that place a dot at each measured midpoint. The process of placing points can be described in more mathematical terms. For a given set of 'n' fixed points,(xf[i],yf[i]),the co-ordinate (x,y) of the next point to be plotted are given by:

$$X=sc * (xp - xf[i]) + xf[i]$$

$$Y=sc * (yp - yf[i]) + yf[i]$$

Where 'i' randomly indexes th 'n' fixed points, and (xp,yp)are the co-ordinates of the point just previously plotted and sc=1/b, where b is a measure of the strength of attraction of each fixed point and thus , b>1.

Nettleton and garigliano[18] have described the Iterated Function Systems (IFSs) consisting of a set of contraction mapping which can be used as a means producing shapes and fractals. The finding of an IFS which represent an arbitrary shape is often referred to as inverse problem. It has been cleared that a shape can be represented as an IFS by choosing contraction mappings such that they form a collage that exactly covers the original shape.

Monro and Dudbridge[19] have described new algorithm for coding and displaying fractals by Iterated Function Systems (IFSs) in a lesser time to run to completion more accurately and efficiently, where heavy computation inhibits the application of fractal technology. A fractal contains infinitely many points, and in principle any of the proposed algorithms could render them in infinite time by consuming infinite memory. But only a representative set of pixels is required on a graphics screen of finite resolution, which is called the rendering of the fractal.

Evolutionary Programming is different from genetic algorithm. Evolutionary Programming is a stochastic search technique inspired by the processes of natural evolution. An initial population of solutions is randomly generated and successive generations produced via a series of operators, which act on the previous generation. This process is continued for desired number of iterations. The study shows that the evolutionary programming approach outperforms genetic algorithm.

Goertzel[20] has explained a variation of the genetic algorithm called the "eugenic genetic algorithm" to provide an effective tool for locating those elements of R^{12}, that belong to the Julia set of a plane quadratic mapping. This provides an unprecedently rapid method for the automatic generation of strange attractors. Here Julia set is the base for generating the attractors corresponding to the parameter vectors within the generalized Julia set of an iteration. While these attractors do not display the same fractal intricacy as Julia set, they do present a striking variety of visual forms. The method for accelerating the attractor generation process is to replace Sprott's Monte Carlo method with a more sophisticated "genetic algorithm" optimization scheme.

Sprott[21] has produced a two dimensional map using, a pair of coupled quadratic difference equations with randomly chosen coefficients, which is repeatedly iterated by computer. The map is tested for stability and sensitivity to initial conditions. The process is repeated until a chaotic solution is found. In this way a computer can generate large collection of strange attractors that are all different, and most of which have considerable aesthetic appeal. A fractal known as strange attractor, is a geometrical object of non-integer dimension and structure on all size scales, although the structure is not self-similar. A single equation with different coefficients can produce almost endless variety of strange attractors.

Sprott and Pickover[22] have described some simple approaches where by a computer can automatically select parameters and generate a large collection of diverse, aesthetically appealing fractal patterns based on general quadratic map basins. It is learnt that non-linear equations can have complicated solutions. The solutions are most interesting if they involve at least two variables, x and y, which can be used to represent horizontal and vertical positions, and the variables are advanced step-by-step in an iterative process. The most general two-dimensional quadratic iterated map is:

$$X=a+bx+cx^2+dxy+ey+fy^2$$

$$Y=g+hx+i\,x^2+jxy+ky+l\,y^2$$

Where 'a' through 'l' are constants chosen at random but are held same throughout the calculation.

Pickover[23] has described a class of integer sequences with unusual graphical behaviors and histories.

Rojas[24] presents a new technique to calculate popular Mandelbrot set more efficiently. The technique used is "Divide and conquer", which consists of recursively dividing the calculating region into subregions. "Interior" patterns of the Mandelbrot set can be generated by two methods, "the epsilon cross" and "star trails" as reported by Hopper[25]. These two methods are based on examining the results of each iteration of inner loop of Mandelbrot set algorithm. The inner loop produces the variables 'x' and 'y', which correspond to the X and Y co-ordinates of a point on a complex plane. The behavior of this point as it moves about the plane on successive iterations of the inner loop has been analyzed in various ways to create new images. The "epsilon cross" method adds a test at the end of the inner loop to check the position of the point (x,y) after each pass through the inner loop. The "star trails" method is based on the patterns formed by plotting points calculated from several iterations of the inner loop on X-Y plane. Some locations inside (and outside) the Mandelbrot set yield clusters of points occurring alone or groups.

Prusinkiewicz and sandness[26] have described two methods for generating Koch curves analogous to the commonly used iterative methods for producing images of Julia sets. First method is based on characterization of Koch curves as smallest nonempty set closed with respect to union of similarities on the plane. This method is known as "Attracting method". The "Attracting method" is similar to the method used for generating images of Julia sets termed the Inversion Iterated Method (IIM). This method is relatively fast and particularly useful when studying the impact of parameter changes on the curve shapes. The Julia set has been generated by using the inversion iteration method, by iterating two mappings:

$$f_1(Z) = +\sqrt{(Z+1)}$$
$$f_2(Z) = -\sqrt{(Z+1)}$$

Another method called 'Repelling method' involves a nontrivial problem of selecting the appropriate transformation to be applied at each iteration step. Bhavsar, et al[27] have introduced the concept of traversal strategies to generate numerous fractal images in contrast to a single fractal image obtainable otherwise. Two basic components required for generating a geometric fractal are an initiator and a generator. This part is then reduced and displaced so as to have its endpoints coincide with those of the segment being replace. A similar process is carried out for the subsequent iterations. Another method is, to use the size of the generated segments to terminate the process. If there are M segments in the generator and Q segments in the initiator, the resulting image after N iterations consists of QM^N segments. Every level of iteration contains smaller copies of the generator exhibiting the property of exact self-similarity.

Stephen,Nicholas[28] have presented an algorithm for some other methods with which the pattern-generating schemes are themselves patterns with some built-in information on orientation in later levels of iteration.

David[29] has described some fascinating artistic and mathematical aspects of fractals based on polyhedral models. The comparison has been made of tetrahedrons and dodecahedrons, generated by author, with the Koch curve and fine aspects of difference have been discussed.

CHAPTER 4

GENERATION OF ALGEBRAIC FRACTALS

The Algebraic fractals are generated by employing the algebraic transformation function iteratively:

$$Z_{n+1} \leftarrow f(Z_n)$$

Where Z is a complex variable and Z_0 represents the initial value of Z and Z_i the value of the complex quantity at the i^{th} iteration. The most famous algebraic fractal objects are the Mandelbrot set and the Julia set. These objects are generated from the same function:

$$F(Z)=Z^2+C$$

Where C is a complex constant.

4.1 Mandelbrot set

The Mandelbrot set is defined as the set of all complex C values for which the iterative process, $Z_{n+1}= Z_n^2 +C$, does not escape to infinity with the start point $Z_0= (0,0)$. Each point in the plane has a pair of coordinates (x,y) that can also be interpreted as the complex number x+iy.

The *basic approach* of checking whether the point escapes to infinity or not can be best illustrated with example as follows. The non-fractal squaring operation $f(Z) = Z^2$ in the complex plane transforms points according to their relation to the unit circle

(Fig. 4.1). If a complex number has modulus <1, then squaring it repeatedly makes it go towards zero. If it has modulus >1, repeated squaring it makes it tend towards infinity. If it has modulus = 1, then points remain on circle. Numbers with modulus '1' has still modulus '1' after repeated squaring. Thus, some complex numbers "fall toward zero", some "fall towards infinity" and some do neither. The last group forms the boundary.

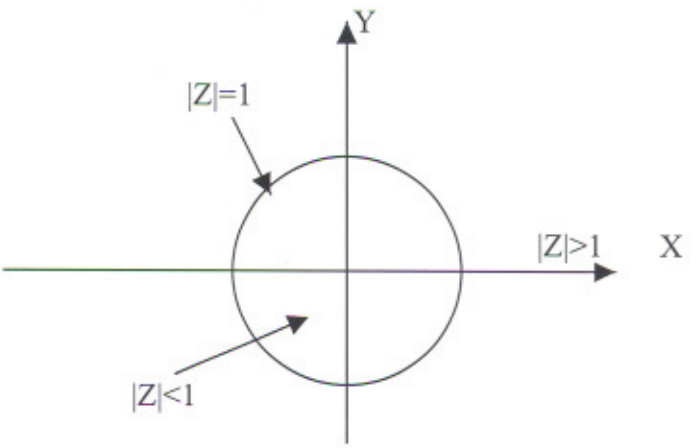


Fig. 4.1

A *fractal image* consisting of the Mandelbrot set is constructed as a two-dimensional array of pixels wherein each pixel is represented as a pair of (x,y) coordinates. The values of the pixel coordinates x and y are based on real and imaginary parts of c(x+iy). The correspondence between a point c of complex plane and the pixel position(p,k) in the fractal image (display) is shown in fig. 4.2.

The value of $c(c_x , c_y)$ for (p,k)th pixel is defined as c_x+ic_y

where

$i=(-1)^{1/2}$

$c_x = xmin_{complex\ plane} + p * xstep$
$c_y = ymin_{complex\ plane} + k * ystep$

xstep = xsize of complex plane/xsize of display plane

i.e. xstep= (xmax - xmin)/no_of_columns

and
ystep = ysize of complex plane/ysize of display plane
i.e. ystep= (ymax - ymin) / no_of_rows

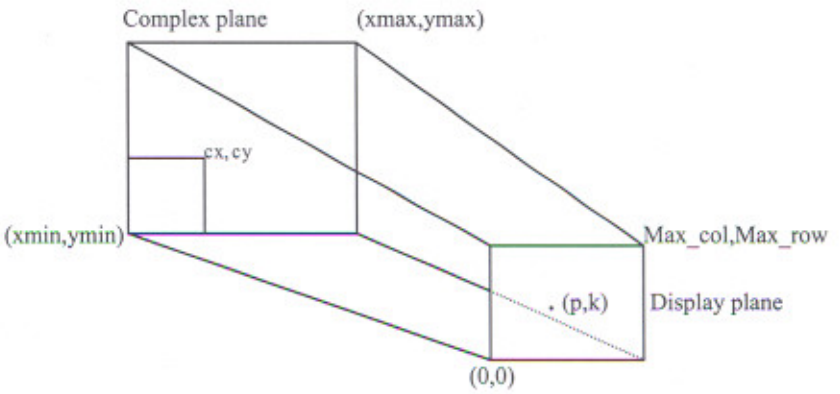


Fig. 4.2

For each pixel in the image, the value of z is computed iteratively. This iterative process terminates when the absolute value of z escapes to infinity means diverges beyond a certain preset limit (R) or when the number of iterations exceeds the maximum allowable number of iterations (MITER). The number of iterations(iter) determines the color of that pixel in the image. Generally black color is assigned to finite points (that don't blow up) and other pixels are colored according to value of 'iter'(usually a color look-up table is used for this purpose).Suppose that we want to know if the point C with coordinate (2,0) in the plane belongs to the Mandelbrot set or not. First we have to take the point Z_0=(0,0) as starting point of the iteration . The next point is defined by the relation,

$$Z_1=Z_0^2+C \qquad (1)$$

These are operations with complex numbers (one complex multiplication and one complex addition) and they performed according to the formulae

$$(a+ib)(c+id) = (ac-bd)+i(bc+ad)$$
$$(a+ib)+(c+id) = (a+c)+i(b+d)$$

If we apply these formulae to relation (11) we get,

$$Z_1 = (0,0)(0,0)+(2,0)$$
$$= (2,0)$$

In the next step we get,

$$Z_2 = (2,0)(2,0)$$
$$= (6,0)$$

We repeat this procedure several times. Each time it is notified that the next point moves further away from the origin. It can be said that succession of points escapes to infinity. The same procedure applied to point C=(0,0) yields a stationary point(stationary at the origin). Because this succession does not escape to infinity, it is enough to see if it is already out of a circle big enough around the origin. For example, the circle of radius R=25 can be taken. We declare a point succession as one going to infinity if the distance of one of its points from the origin is greater than the radius of the circle.

If the point has coordinate (X,Y) we have to test only if the square root of X^2+Y^2 is bigger than R. In practice, we cannot calculate an infinite succession of points. We have to set a limit, say, of 100 iterations. If in 100 iterations the succession has not surpassed the boundaries of circle of radius R, we accept C as a member of the Mandelbrot set and we paint this point on the screen with the appropriate color (Fig. 5.2).

4.2 Julia set

Julia-Fatou set is also the result of same self-squared function $f(Z)=Z^2+C$ depending on the intial position of Z selected for iteration, If we draw the set of points that go

neither. The resulting sequence $Z_0, Z_1, Z_2, \ldots Z_n$ of a set of iterations behave in one of the two ways[38]. Either it roams freely, moving towards infinity, or it is trapped with in a certain region of complex plane. First set of points are known as escape set, and those remain confined to region belong to prisoner set. For a point inside the prisoner set, the sequence Z_k remain with in numerical prison, whereas for a point outside the prison set, the sequence moves away from center of the plane and escapes towards infinity. The prison set and escape set are separated by a narrow boundary known as Julia set. For every central parameter 'C' the resulting fractal image lies in one of the two categories, a connected Julia set (fig 4.3a) or a Julia set consisting of infinite number of disconnected points like dust (fig. 4.3b).The Julia set of fractals are generated by varying the value of Z and keeping C as constant. The fractals created in this manner are called Z-plane fractals.

4.3 Relationship between Mandelbrot set & Julia set

The Mandelbrot set is a one page dictionary of Julia sets or Mandelbrot set is a map of Julia sets. This is so called because if we enlarge the Mandelbrot set sufficiently at any given point C we obtain something that looks very much like Julia sets at that point. So Mandelbrot set is nothing but made up of Julia sets each point of display, or Julia set at a particular point is the manifestation of Mandelbrot set at that point. Suppose we have Julia set for which we set the value of C to be W. So the initial value ,Z_0 of Julia set is also W. Then we have:

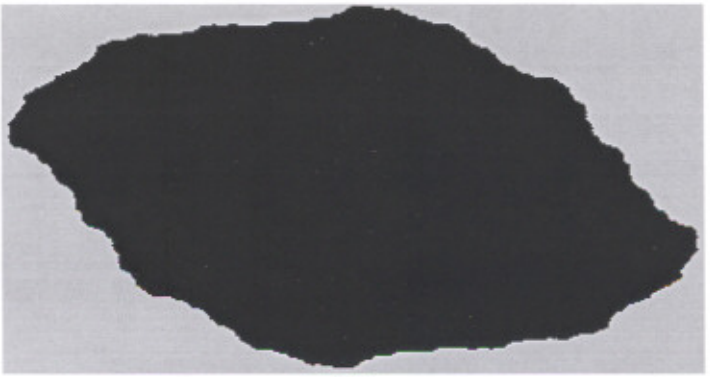
$$Z_0 = W$$

$$Z_1 = W^2 + W$$

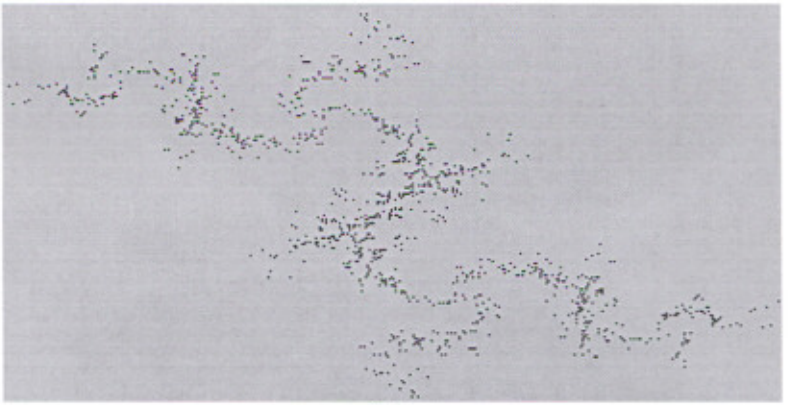
Now we look at the Mandelbrot set at the point where C is W. We have

$$Z_0 = W$$

$$Z_1 = W^2 + W$$



[a]



[b]

Fig 4.3

We say that the result of iteration is same for Julia and Mandelbrot set at that point where at these points. Suppose if we move from this point by distance C_1, where C_1 is complex number. Then mathematical representation of Julia set is :

$$Z_0 = W + C_1$$

$$Z_1 = (W+C_1)^2 + W$$

And the mathematical representation of Mandelbrot set is :

$$Z_0 = W + C_1$$

$$Z_1=(W+C_1)^2 + W+C_1$$

the result of iteration will again be the same for the two sets but as C_1 increases the result diverge more and more. Hence for very large magnifications and where C_1 is very small, Mandelbrot set and Julia set should be same.

4.4 Algorithm for generaton of algebraic fractal by whole scan method:

1. Select Transformation function f
2. Select the values of MITER and R // MITER is number of iteration
   // R Size of Escape circle
3. Select a pixel, i.e.(x,y) coordinates of a pixel
4. Select intial values for c and z_0 based // i.e. c-plane or z-plane
   on type of fractal desired
5. Set the iteration count iter, to zero
6. Compute the new value of z as $z \leftarrow f(z)$
7. Increment the iteration count 'iter' by one.
8. Repeat steps (6) & (7) until the // based on MITER and R
   termination condition is satisfied
9. Assign a color to the pixel based on the
   Iteration count iter, using a preselected color table.
10. Repeat steps (3) through (8) for all the pixels in the desired image.

CHAPTER 5

PRESENT WORK

Many aesthetically appealing pictures have been derived by previous workers, but problem with fractals is high execution time. After going through literature [1-38], it was felt that there are some sources of inefficiency, which can be avoided, to improve the performance of fractal generation algorithms. The main purpose of this project work is to show some ways in which the necessary calculations to draw the algebraic fractals are accelerated. As the generation of set $z_{n+1} \leftarrow z_n^2+1$ has already been discussed, It can be noticed that there are three main sources of inefficiency:

* Criteria for Escaping to infinity(Large Size of Escape circle, Large number of iterations)
* Order of calculations
* Floating point operations

which must be considered in order to improve the efficiency of algebraic fractal generation algorithms. With these aspects in mind, some of the experimental analyses are being reported here in the following sections. In addition, a technique has been given which speeds up the generation of connected Algebraic fractals.

5.1 Escape circle:

It is now clear that in order to decide whether or not a point belongs to set, many calculations must be done iteratively. In each iteration several floating-point additions and multiplications have to be performed and the distance from the origin has to be

compared to the radius R of the escape circle. Here we find first source of inefficiency when dealing with the calculations. It is clear that greater the radius of the escape circle, the larger the number of iterations that are needed to verify that a point succession has trespassed the circle borders. Some programs use very big values for R, in the hope that this will yield a better result. It has been observed that it is enough to use a radius between 2 and 10 for most of the fractals. For example for mandelbrot set it is enough to select a circle of radius 2, If only the region of interest of plane being investigated (the rectangle with x-boundaries -2.25 and .75 and y-boundaries -1.5 and 1.5) then, any point in succession that skips out of this circle will never return again and will move further away in successive iteration. On the other hand, it is not possible to use a radius smaller than 2 for Mandelbrot, because points to the right of the point (-2,0) and arbitrarily near to it belong to the Mandelbrot set.

5.2 Number of iterations:

The second source of inefficiency, is a limit on the number of iterations., used for deciding whether a point belongs to the specified set or not. There are programs which use 1000 iterations to draw main island of the set using a low resolution [24]. It may require approx 2 to 3 hours to run the program. There is a kind of superstitious feeling that the graphics will look better with more iterations.

Actually, it is not necessary for the programs to perform so many iterations to obtain attractive pictures. For example it is possible to draw main island of the Mandelbrot set with good resolution using no more than 25 iterations. More iterations bring only negligible variations to the graphic as indicated in fig. 5.1b to 5.1c and my experience is that rougher calculation looks even better than the finer. Also, time is increasing

drastically as number of iterations increases as shown in the figures 5.1a to 5.1c for transformation function $(c+5z^6)/(6z^5)$, with no change in picture appearance in fig. 5.1b to 5.1c. It has been observed that 16 iterations are sufficient for this set.

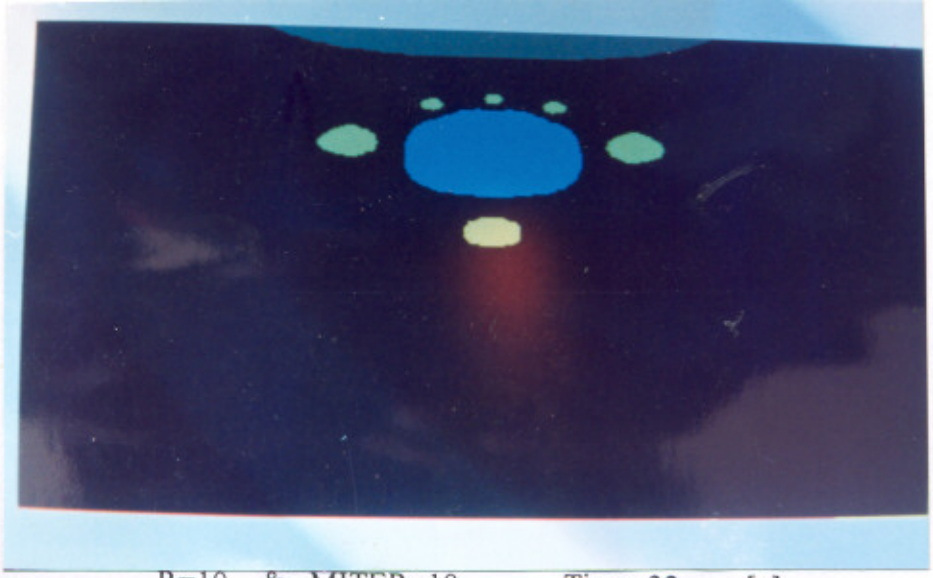
Basic idea is points which belongs to any set (say Mandelbrot set) have infinite escape time, that is they do not escape to infinity at all. Points outside the specified set have different escape times, but the nearer the point is to a region of the specified set, the greater is the escape time. So we should not consider excessive number of iterations.

Thus, one should not think that more iteration always imply a better picture. It depends on the kind of picture in which one is interested. For example, If one is only interested in the Mandelbrot set proper (fig. 5.2a), then some details can be lost in very chaotic regions of the plane.

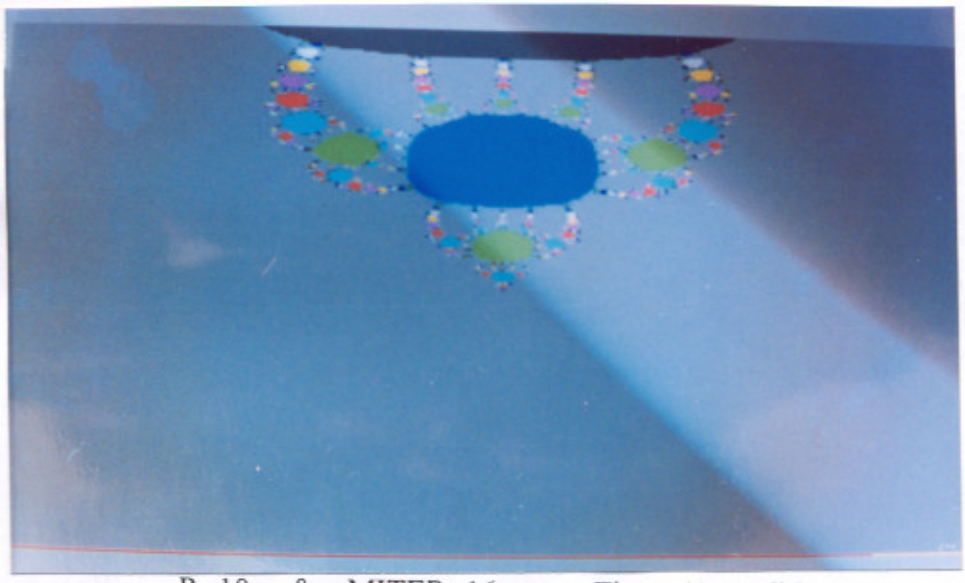
But, if the number of iterations increase excessively, the fine strokes that constitute the body of the set become thinner and thinner. It is very difficult for the center of each pixel to be exactly in one of these thin lines. It is more probable that the center falls to one or the other sides of the lines and in this case pixel will not be colored (Because for each pixel its center is taken as representative and that the calculations have a meaning only for this point). As indicated by fig. 5.3b some details will be lost.

Finally, what will be the upper limit of iterations, is a very subjective question. It depends in which region of a figure one is interested in and the resolution under consideration. More specifically with different regions with a particular resolution , the number of iterations will be different. The following conclusions inferred:

a. Experience has shown, that the number of iterations required to generate the image grows approximately proportional to the logarithm of the magnification



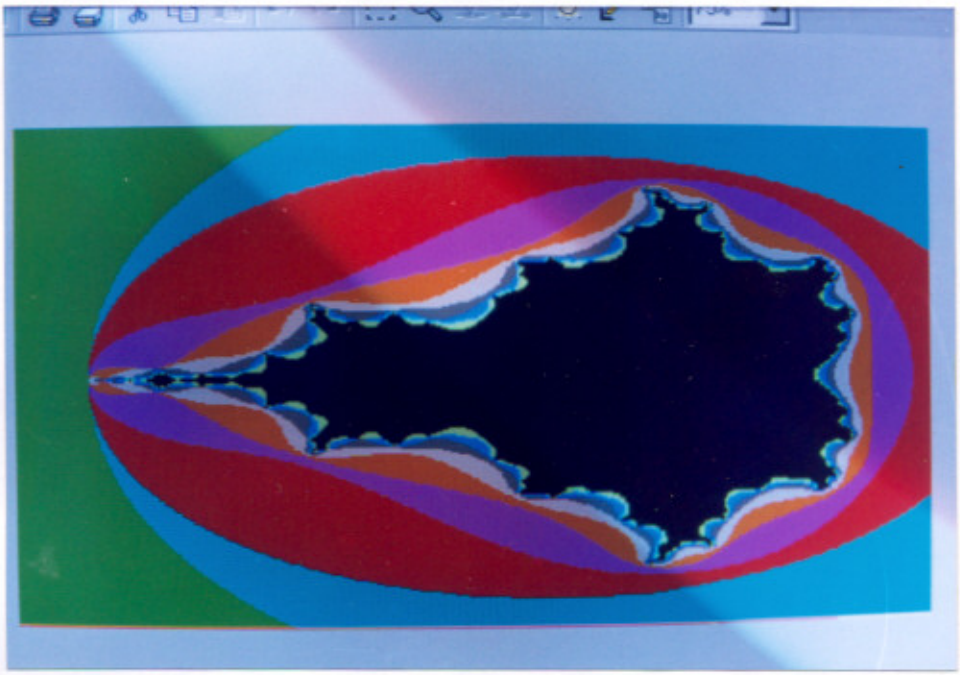
R=10 & MITER=10 Time=32 [a]



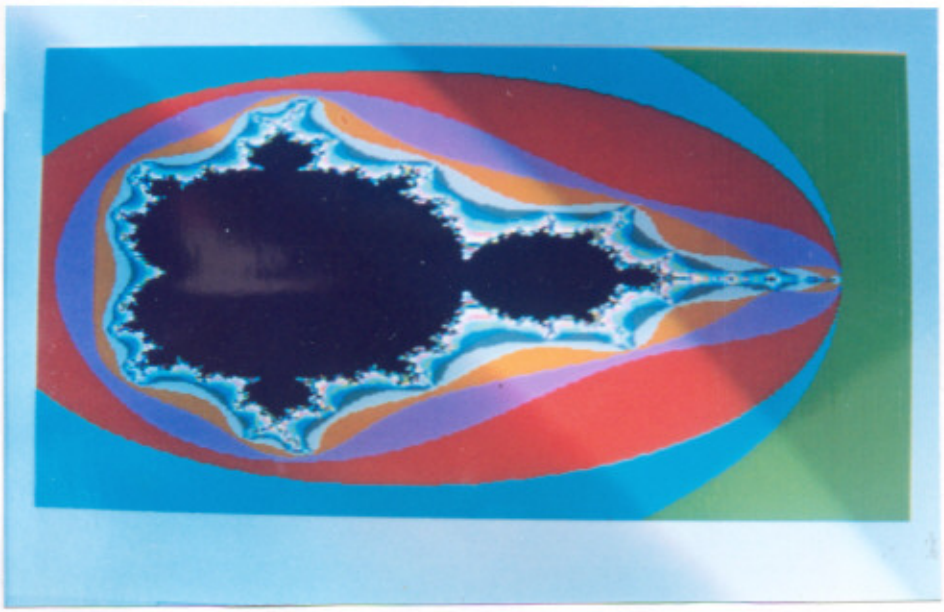
R=10 & MITER=16 Time=48 [b]

R=10 & MITER=25 Time=117 [c]

Shows comparison how time increased with increase in iterations ([a] to [c])
For transformation function $z=(z+5z^6)/(6z^5)$

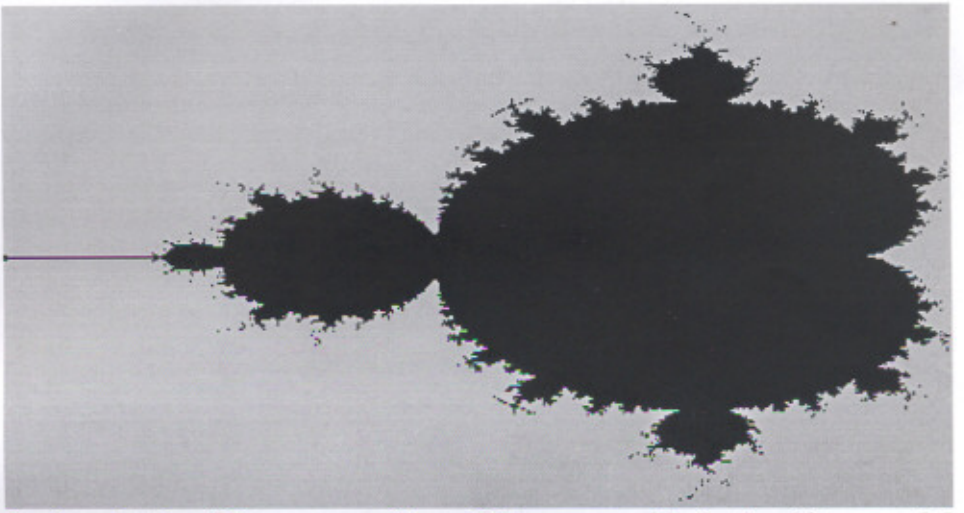


R=4 & MITER=10 Time =8 No details [a]

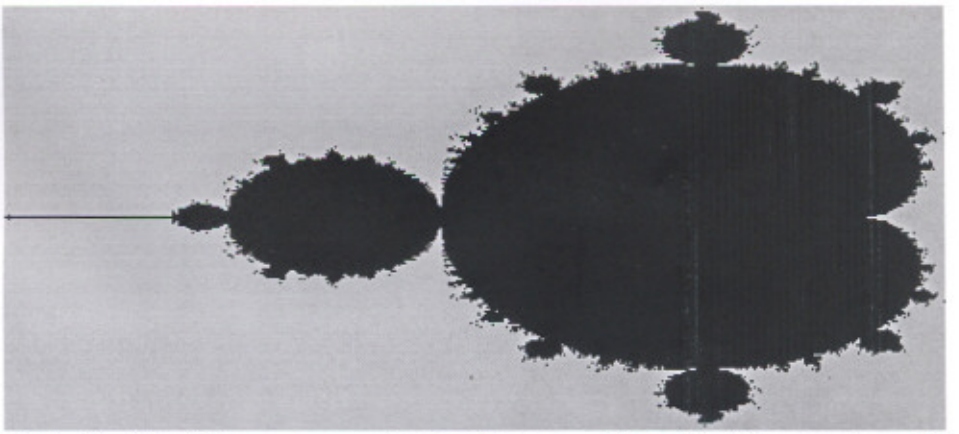


R=4 & MITER=25 Time=9 More detailed [b]

Mandelbrot set ($f(z) \leftarrow z^2+c$
Shows comparison magnification is proportional to number of iterations
Fig. 5.2



R=4 and iter=30 **time =9.5sec** black & white [a]



R=4 & iter=64 **time=13sec** Some Details missing [b]

Shows comparison as the iteration increases excessively, execution time increases and some details lost for Mandelbrot set

Fig. 5.3

factor.

b. If the set is being drawn in color, one should always iterate deeper.

c. If one is plotting the escape regions, especially near to the boundary of the specified set, with a monochrome screen, using black for the even regions and white for the odd ones, one should iterate deeper.

5.3 The Order of Calculations:

The third source of inefficiency found in many programs is the ordering of calculations. For example, there are two routines for Mandelbrot set, which uses as arguments the coordinates of the point and decides if it should be plotted or not:

Routine 1:

```
Plot( float cx, float cy)
{  Float x1,y1,x2,y2,r;
   int  iter=1;
   const MITER=100;
   R=2.0;      // Radius
   x1=0;  y1=0;
   r= sqrt(x1*x1+y1*y1);
   while( r <R and  iter <MITER)
      {   x2=x1*x1-y1*y1+cx;
          y1=2*x1*y1+cy;
          x1=x2;
          r=sqrt(x1*x1+y1*y1+cx;
          iter++;
} if ( k=limit) plot the point;
}
```

Routine 2:

```
Plot( float cx, float cy)
{  Float x1,y1,x2,y2,b1,b2,r;
   int  iter=1;
   const MITER=100;  // Number of iterations
   R=4.;              //Radius square
   x1=x;   y1=y;     //Assign initial value
   b1=sqr(x1);        //store square in b1 & b2
   b2=sqr(y1);
   r=b1+b2;
   while((r<R)  and iter<limit)
   {
       y1=2*x1*y1+cy;
       x1=b1-b2+cx;
       b1=sqr(x1); b2=sqr(y1);
       r=b1+b3;   iter++;
   }
```

The Routine 1, shown above does the job, but too many calculations are wasted. Some simple improvements are the following:

(a) The square root is not necessary. The comparison can be made against the square of R. One call to the SQRT function is saved in each iteration.

(b) It is possible to start with the point (x,y) i.e. non-zero instead of of the point (0,0),

Because of the first iteration takes us always to this point.

(c) The squares of x1 and y1 can be calculated one time, but they can be used twice, first for the calculation of x2 and then for the calculation of r.

(d) The squares of x1 and y1 could be calculated with a special routine, such as SQR in PASCAL. The square of a floating-point number can be calculated easier than a floating point multiplication[24].

With this many unnecessary operation are eliminated as shown in routine 2. Points (a), (b), and (d) must be considered in almost all the *Algebraic Fractal generation algorithms*.(c) may or may not used depending upon the transformation function used. Same way we have to consider all other simple improvements, if possible in other transformation functions. Like cube of a complex number($z=x+iy$) can be replaced by separately calculating its real part equal to $x(x^2-3y^2)$ and imaginary part equal to $y(3x^2-y^2)$. For example, by this simple replacement of z^3 in z^3+c set, with R=25 and MITER=25, execution time reduced from 6sec. to 3sec. Same way try to avoid use of power function(pow), for smaller power of complex numbers, by this one can save approx. 10 to 15 percent of time.

5.4 Avoiding Floating Point Operations:

Several million floating-point operations are required in order to have an aesthetic graphic. Even with a fast workstation using a floating-point co-processor, the entire process take several hours if one is investigating a very small region of the set.

Raul Rojas[24] described the ways by which some of the floating point operations can be avoided and implemented. He claimed that one can save 20% of the CPU time by replacing two simple operations in routine 2 (multiplication of 2 to floating point

number and comparison r<R by integer operations) on which floating point co-processor is not present.

For this, one should know how the floating-point result is stored in the computer. As the storage of floating-point number is system dependent, we have to write special routines for avoiding floating point operations for specific system.

5.5 Design and Implementation of New technique for faster generation of connected algebraic fractals:

All the refinements mentioned previously do not essentially change the calculation procedure. We still have to go row by row, pixel by pixel(called whole-scan), calculating the escape time. However it is possible to get an additional speed up by using a rather different strategy to calculate the points in the graphic.

This is useful for the connected sets. Connected set means each point of it is not isolated. There is always a path of points in the connected set communicating between any two arbitrary points in the set. We can exploit this property to reduce the size of rectangular window and then subdivide the resulting window (heterogeneous quadrant) into subwindows. First of all draw the sides of the window. If black color is used for the points in the Mandelbrot set and white for the points outside, then color of the sides only has to be checked. If the sides contain black points only then it is sure that whole window is black, because it contains the points in the stated set only. This happens because if a white point were to be located somewhere in the window, there should be a path of white points going out of it and at least one of the sides should contain white points(fig.5.4). The reverse is also true. If the sides of the window are found to be white, then the whole window can be colored white. No black points could

be inside, because that would imply that a path of black points should traverse the window. The only exception is when a window totally contains the stated set. In this case, it is checked if the origin is located in the window.

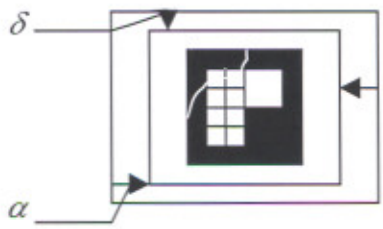


Fig. 5.4

5.5.1 Implementation Procedure:

(a) Define the coordinates of a window in the real plane and its size in pixels.

(b) Look at the sides of the whole window:

(b.1) If the sides of the window have the same color, reduce the size of the window by subtracting α from x-coordinate and δ from y-coordinate. Repeat this step starting with (b) until:

(b.1.1) Window with either side of less than two pixels left, then stop.

(b.1.2) or If there are different colors in the sides of the window.

(b.2) If there are different colored pixels in the sides of the window (Heterogeneous window), set its corresponding flag to 1, divide it into four quadrants and set its flag to zero. Check the sides of all the quadrants one by one:

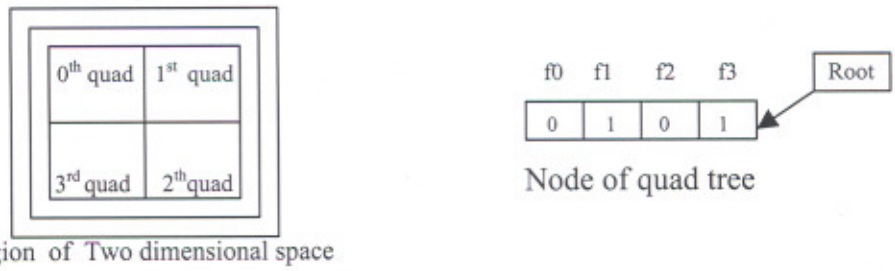
(b.2.1) If four sides of the quadrant are of same color, paint that quadrant with this color set its flag zero.

(b.2.2) If sides have pixels of different colors or size of the

either one of the quadrant side is less than five pixels, then paint the pixels of that quadrant by their corresponding colors and set its corresponding flag zero.

(b.2.3) Repeat steps (b.2) to (b.2.2) until all the quadrants will be painted.

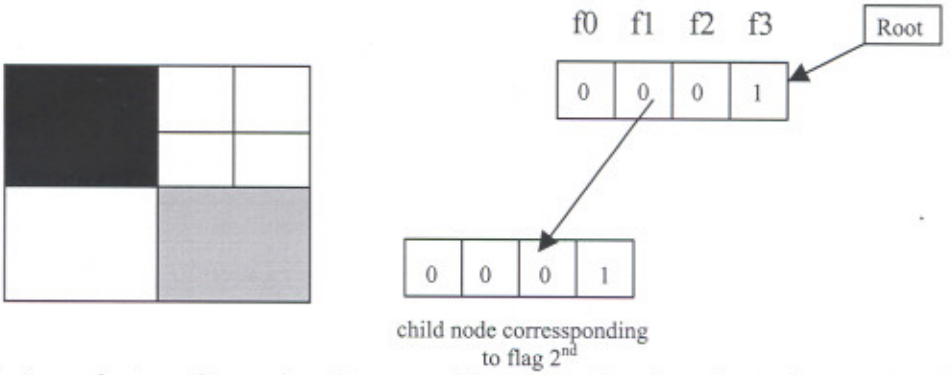
For implementation, after reducing the size of display area, I use quadtree approach. At the highest level, root represented by single node, corresponds to first heterogeneous window (fig. 5.5). Next level contains the root's four children, which correspond to the four quadrants of the selected window and so on. At the lowest level nodes corresponds to smallest squares having either sides less than four pixels.



Region of a two-dimensional space divided into numbered quadrants and the associated Quadtree node with associated flags

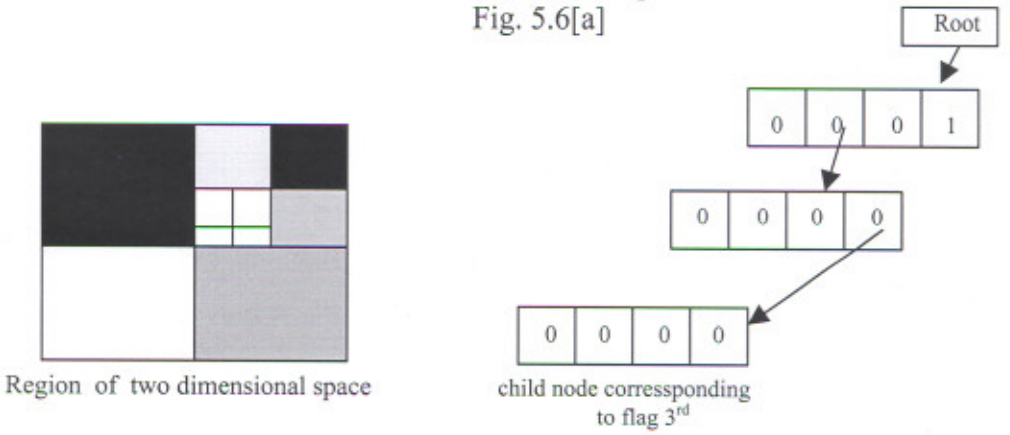
Fig. 5.5

In (Fig. 5.5), flag f0 and f2 are zero implies 0^{th} and 2^{nd} are homogeneous, so color 0^{th} and 2^{nd} quadrants with their respective colors(corresponding to the color of their sides). Flags f1 and f3 are 1 implies 1^{st} and 3^{rd} are heterogeneous quadrants, so subdivide respective quadrants (1^{st} & 3^{rd}) into further quadrants as shown in figures 5.6[a],5.6[b] one by one.



Region of a two-dimensional space with two levels of quadrant division and the And the associated quadtree representation

Fig. 5.6[a]



Region of a two-dimensional space with three levels of quadrant division and the And the associated quadtree representation

Fig. 5.6 [b]

Now repeat the same procedure for 3rd quadrant until all flags become zero. For example, How this technique is applied to mandelbrot set is shown in the (fig: 5.7).

A substantial improvement has been found through the use of this method which is much faster and running time can be cut short to approx. one-half or one-sixth of the time required in executing $Z_{n+1} \leftarrow Z_n^{\alpha} + 1$ as compared to time taken by whole scanning method (Fig 5.7 to Fig. 6.2). It is clear from the table 5.2 that the time taken by suggested method to execute the same transformation function is drastically lower than the normal method. The experimentation has been done on pentium 200 MHz processor. It is very important to mention that table 5.2 was observed using only 25

iterations, and window with x-boundaries -2 and 2 and y-boundaries -2 and 2, and α=25, δ=15. If we use many more iterations, the difference between one method and other could be much more drastic.

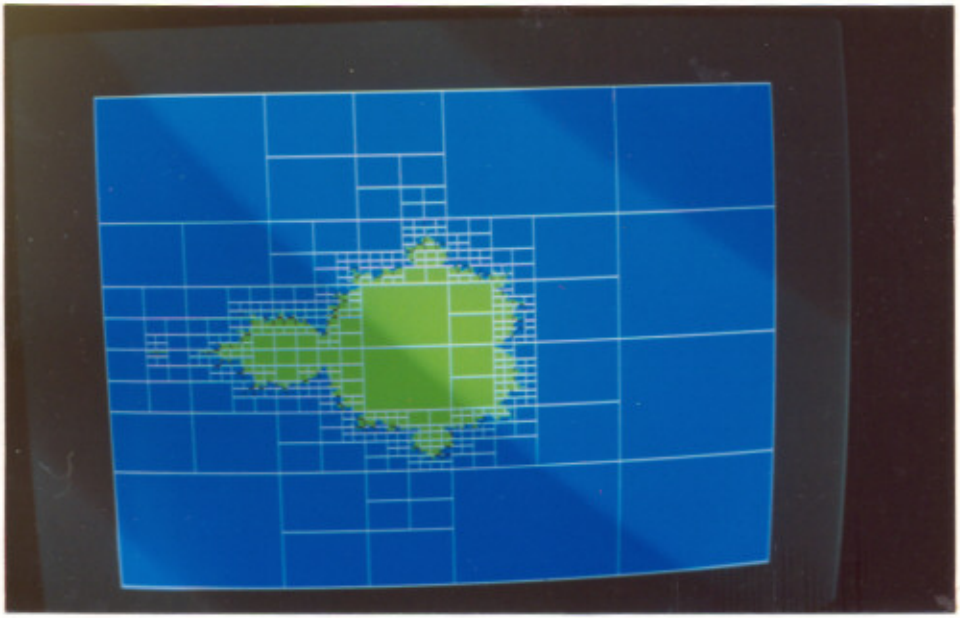
Table 5.1 : Parameters used for the functions in table 1:

S.No.	Function Used	Intial condition(z_0) $z_0=x_0+iy_0$	Maximum Value	Minimum Value
1.	$z_{n+1} \leftarrow z_n^2 + c$	$x_0 = .1$ $y_0 = 0$	x=2.25 y=2.25	x=-2.25 y=-2.25
2.	$z_{n+1} \leftarrow z_n^{\alpha} + c$ +ve α (except α=2)	$x_0 = .5$ $y_0 = .5$	x=2.00 y=2.00	x=-2.00 y=-2.00
3.	$z_{n+1} \leftarrow z_n^{\alpha} + c$ -ve α	$x_0 = 1$ $y_0 = 1$	x=2.00 y=2.00	x=-2.00 y=-2.00

Table 5.2 : Difference in execution time of Whole-scan & New suggested Methods

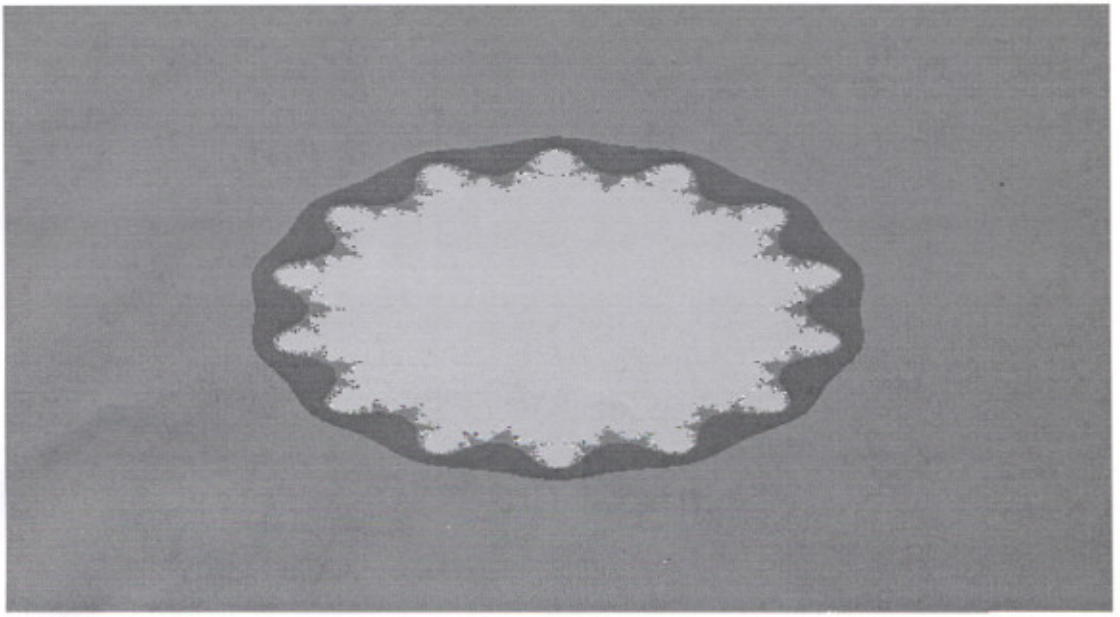
S.No.	Function Used	Time Taken By Whole-Scan Method (in Sec.)	Time Taken By Suggested Method (in Sec.)
1.	$z_{n+1} \leftarrow z_n^2 + c$	9.00	2.58
2.	$z_{n+1} \leftarrow z_n^6 + c$	7.00	3.79
3.	$z_{n+1} \leftarrow z_n^{15} + c$	11.00	5.76
4.	$z_{n+1} \leftarrow z_n^{4.5} + c$	15.00	7.69
5.	$z_{n+1} \leftarrow z_n^{8.5} + c$	17.00	9.34
6.	$z_{n+1} \leftarrow z_n^{12.5} + c$	18.00	9.61
7.	$z_{n+1} \leftarrow z_n^{-2} + c$	22.00	8.40
8.	$z_{n+1} \leftarrow z_n^{-6} + c$	30.00	11.64
9.	$z_{n+1} \leftarrow z_n^{-15} + c$	41.00	14.50
10.	$z_{n+1} \leftarrow z_n^{-4.5} + c$	61.00	23.51
11.	$z_{n+1} \leftarrow z_n^{-8.5} + c$	67.00	23.29
12.	$z_{n+1} \leftarrow z_n^{-12.5} + c$	74.00	23.00

The suggested method also grows relatively faster than whole-scan method as resolution increases. For example, if one starts with a window size of say 256 by 256 and one increases this to 512 by 512, then there are four times as many points as before. With the traditional method, four times more calculations are made, so that the CPU time grows linearly with the number of pixels used. With the new method one does not calculate all points on the screen.

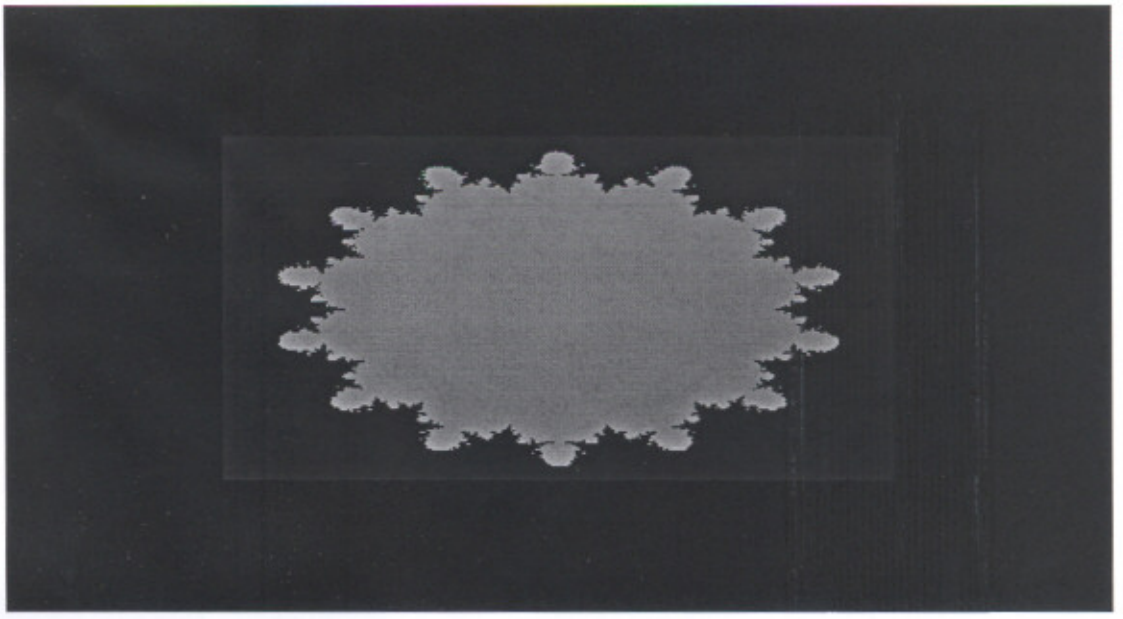


Speeded up Mandelbrot Set using New Technique : $Z \leftarrow Z^2 + C$

Fig. 5.7

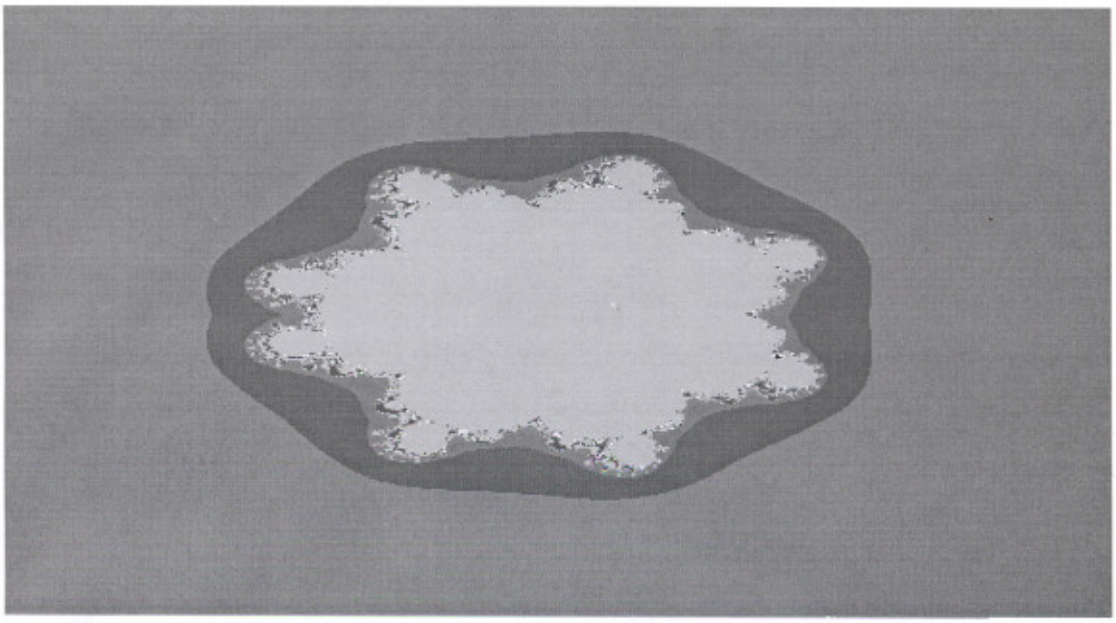


Set using All-Scan method : $Z^{15} + C$ [a]

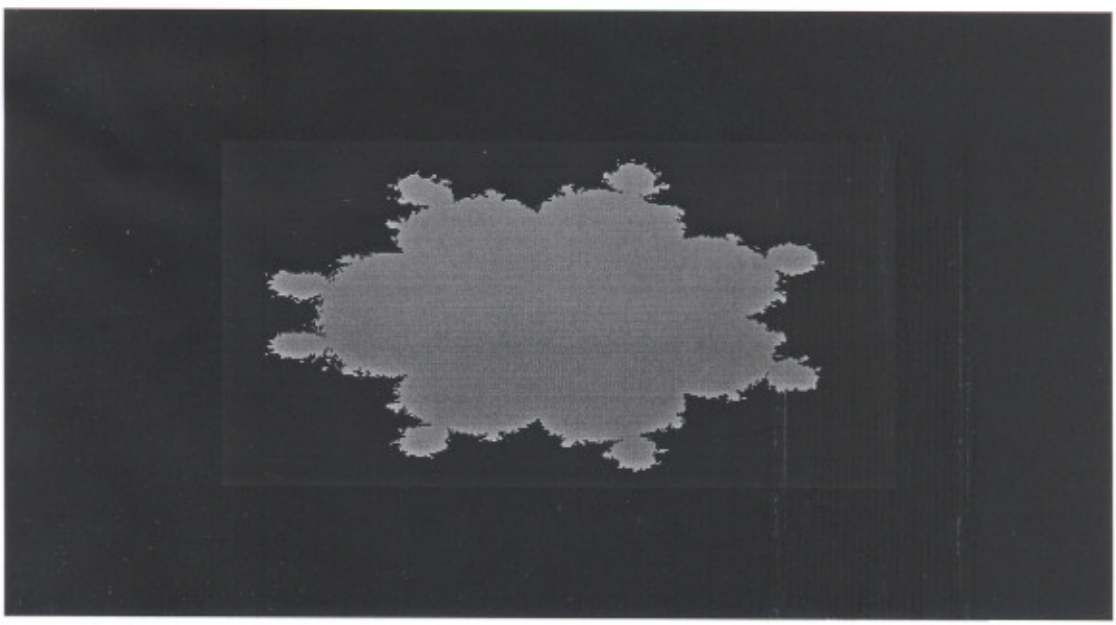


Speeded up Set using New Technique : $Z^{15} + C$ [b]

Fig. 5.8

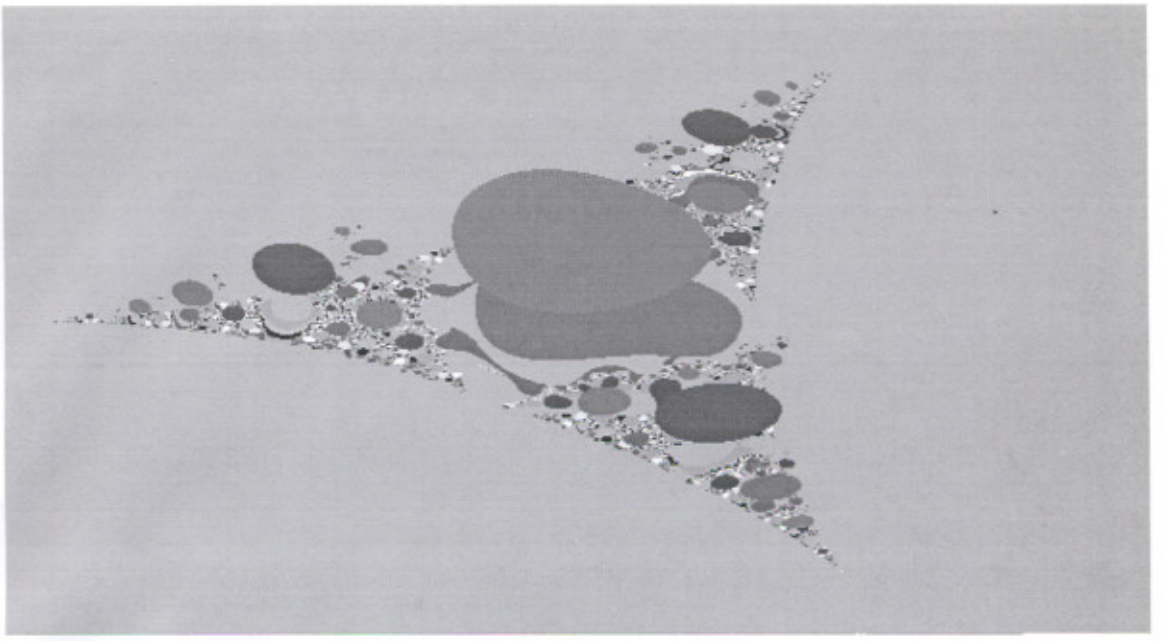


Set using All-scan method : $Z^{8.5} + C$

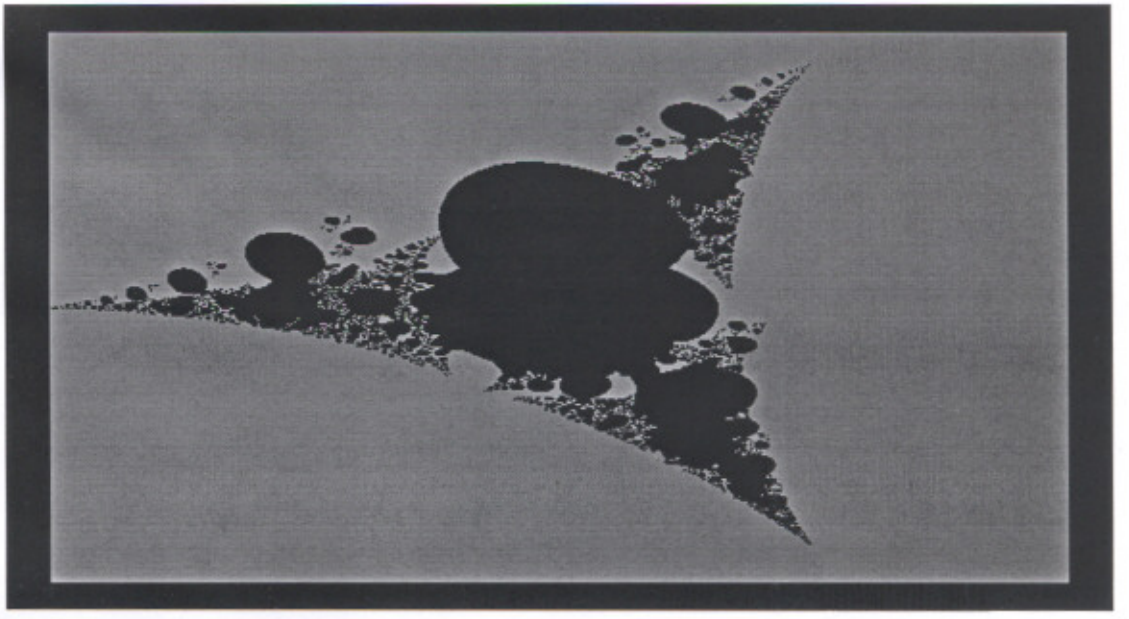


Speeded up Set using New Technique : $Z^{8.5} + c$ [b]

Fig. 5.9

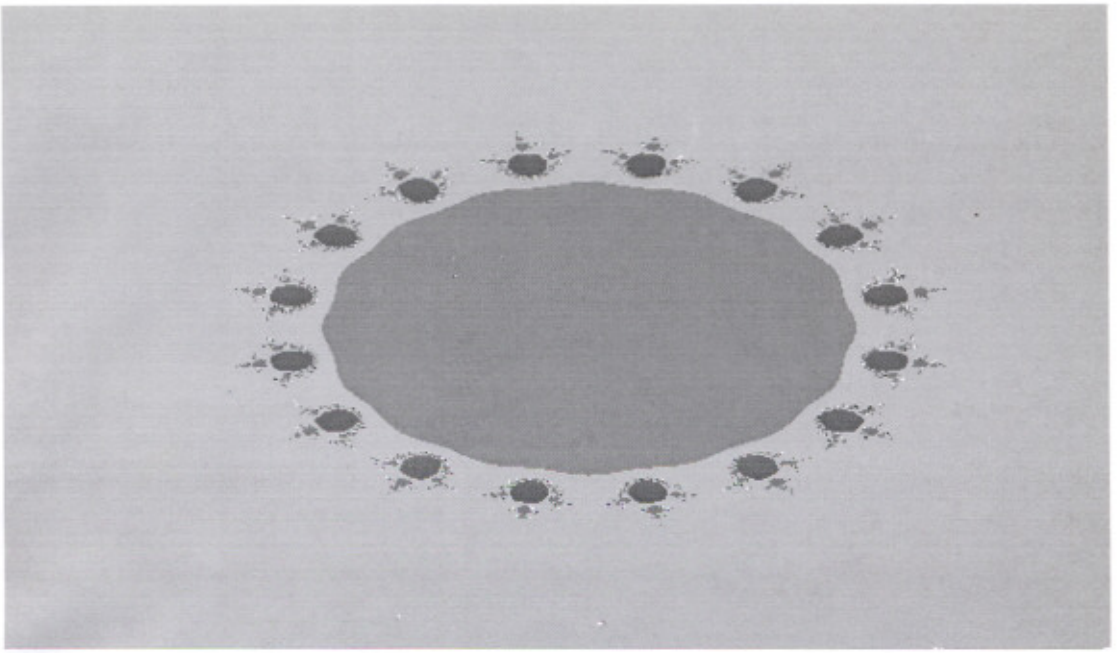


Set by All-Scan method : $Z^{-2} + C$ [a]

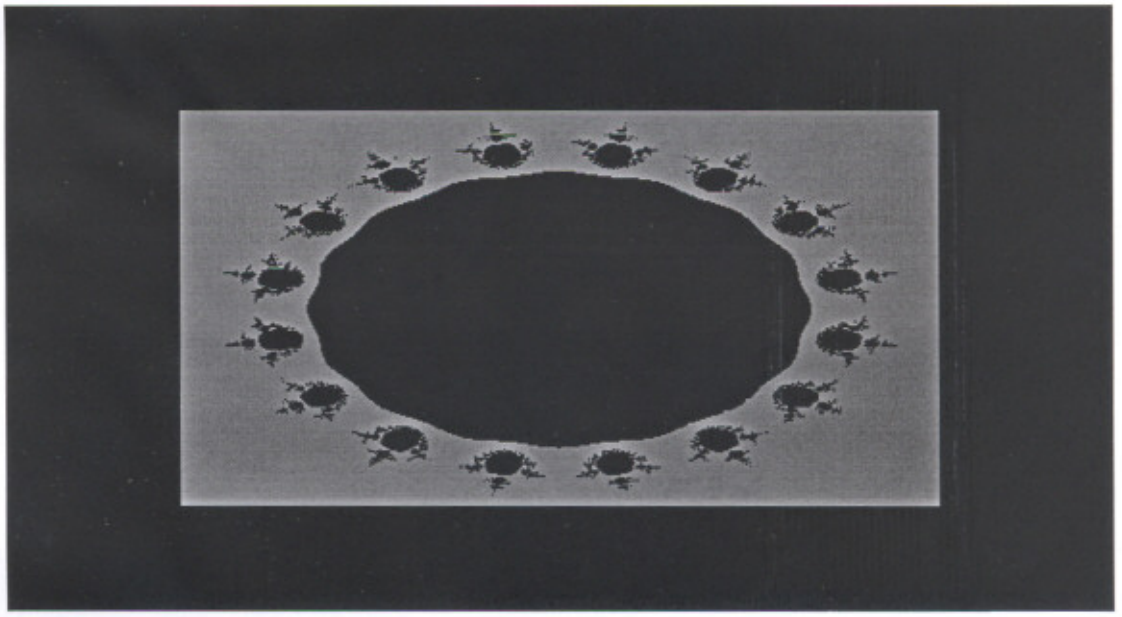


Speeded up Set using New Technique : $Z^{-2} + C$ [b]

Fig. 6.0

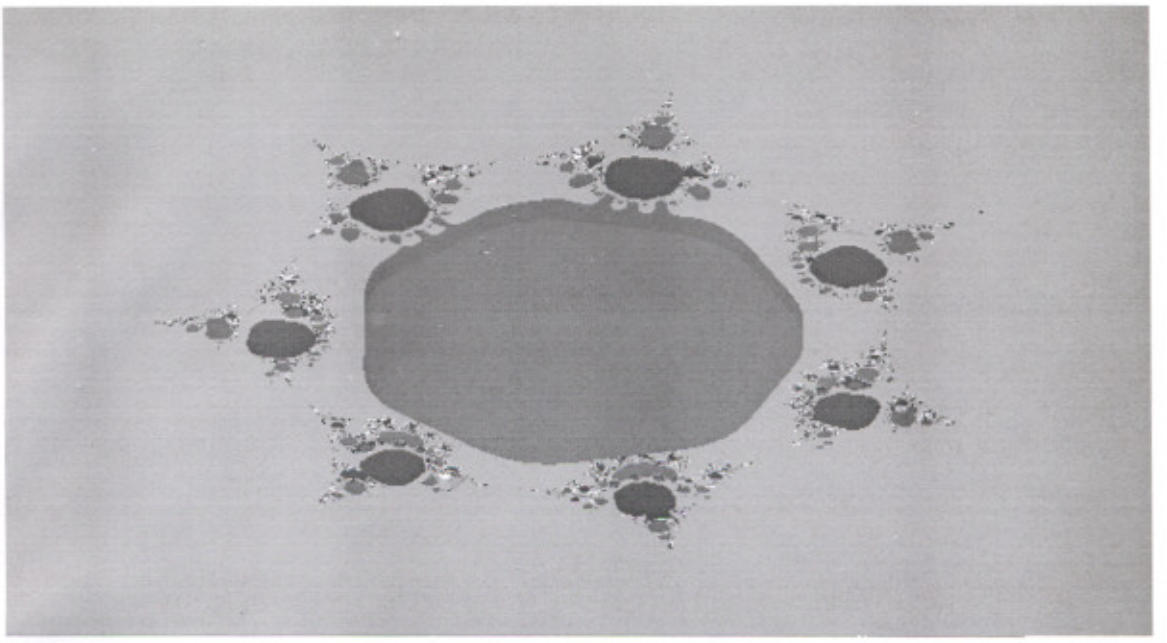


Set using All-Scan method : $Z^{-15} + C$ [a]

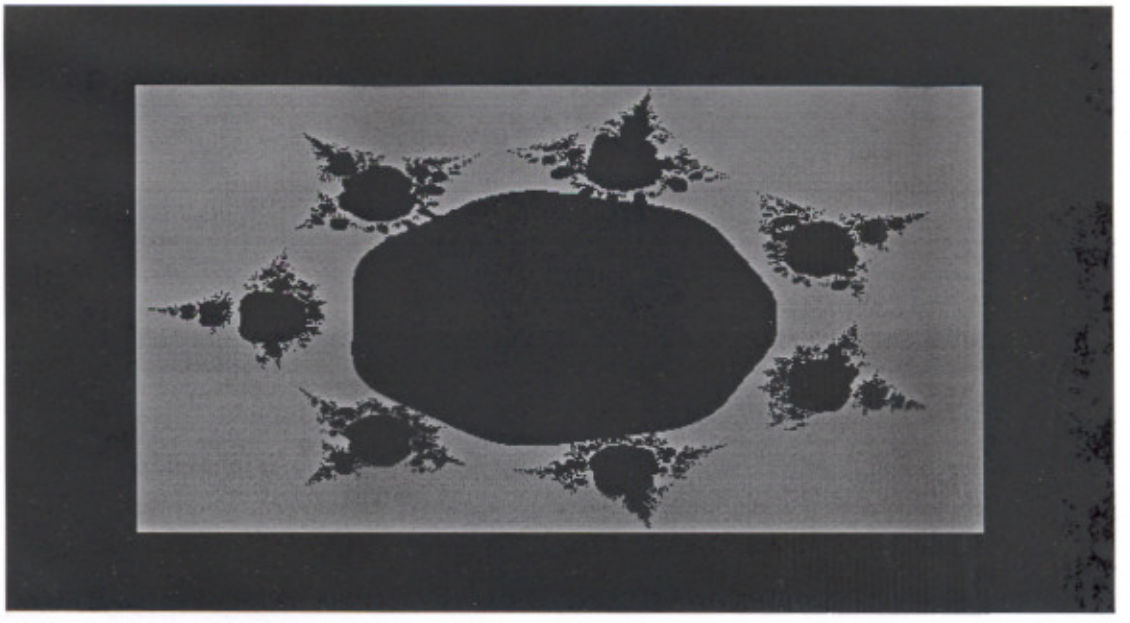


Speeded up Set using New Technique : $Z^{-15} + C$ [b]

Fig. 6.1



Set using All-Scan method : $Z^{-6} + C$ [a]



Speeded up Set using New Technique : $Z^{-6} + C$ [b]

Fig. 6.2

CHAPTER 6

Conclusion and Future Scope

6.1 Conclusion

Fractals are compact method of representation of natural images. Many of the complicated images in nature can be simulated through fractals, which otherwise are very difficult to generate by using conventional computer graphic techniques. The biggest constraint in creating the fractal images is drastically high execution time. The work presented in this thesis is an attempt towards reducing that high execution time.

In this thesis, to generate fractals in less execution time, three techniques proposed by Rojas[24] for Mandelbrot set have been reviewed. Further, it has been suggested that these techniques can also be applied to all other algebraic fractals and a new technique for faster generation of connected algebraic fractals has also been proposed. This technique (implemented using the concept of reducing the specified set window and dividing the reduced window (heterogeneous) into subwindows, then filling the window having boundary of same color)has been found to be very successful in reducing the execution time, while retaining the quality of the image.

The execution time required for generation of algebraic fractals have been computed and compared with that of commonly used conventional technique and results are presented in tabular form. The results show that the new technique is more efficient

than that of commonly used technique. The following conclusions have been inferred from the results:

- The new technique have reduced the execution time approx. by $1/3^{rd}$ to $1/7^{th}$ of the conventional method.
- A maximum of 25 iterations are sufficient, for knowing the main characteristics of the specified set.

6.2 Future Scope

As a course for future work it is suggested that the following possibilities can also be explored to increase the efficiency:

- A promising field of research would be to investigate if previous results for neighboring points could be used for other points. For example, Mandelbrot set can be investigated at fewer points and the interpolation in 2-d can be done to find the intermediate points.
- The real and imaginary parts of Z-plane can be converted into large integers and then, integer calculation can be used.
- In a distributed computing environment, parts of the whole calculation can be sent to other computers and the results may be collected through files in the server. Naturally the ultimate in speed would be obtained using a hardware device for the calculations at the fastest possible pace.

REFERENCES

1. Mandelbrot B.B., " The Fractal Geometry of Nature", W.H. Freeman, San Fransisco, CA, 1982.
2. Picover Clifford A., "Mathematics and beauty V: Turbulent complex curls", *Computer & Graphics*, Vol. 11, No. 4, 1987, pp. 499-508.
3. Klein M.,Rossler E.O., Parisi J., Baier G., Peinke J.,Kahlert C.,Hudson J.L., "Toward a better understanding of fractality in nature", *Computer & Graphics*, Vol. 15, No. 4, 1991, pp. 583-596.
4. Bradley E., "Causes and effects of chaos", *Computer & Graphics*, Vol. 19, No. 5, 1995, pp. 755-778.
5. Philip A.G.D.,"Warped midgets in the mandelbrot set", *Computer & Graphics*, Vol. 18, No. 2, 1994, pp. 755-778.
6. Gujar G. Uday,Bhavsar C. Viredra, "Fractals from $Z \leftarrow Z^{\alpha}+C$ in the complex C-plane", *Computer & Graphics*, Vol. 16, No. 1, 1992, pp. 45-49.
7. Bhavsar V.C.,Gujar U.G.,Vengala N., "Fractal images from $Z \leftarrow Z^{\alpha}+C$ in the complex Z-plane" *Computer & Graphics,* Vol. 16, No. 1, 1992, pp. 45-49.
8. Vengala N.,Bhavsar V.C., Gujar U.G., "Vectorization of generation of fractals from $Z \leftarrow Z^2+C$ on IBM 3090/180VF", *Computer & Graphics*, Vol. 17, No. 2, 1993, pp.169-174.
9. Shirriff Ken W., "An investigation of fractals generated by $Z \leftarrow 1/Z^{\alpha}+C$", *Computer & Graphics* Vol. 17 No. 5,1993, pp. 603-607.
10. Glynn E.F., "The evolution of the gingerbread man ", *Computer & Graphics*, Vol. 15, No. 4, 1991, pp. 579-582.

11. Lakhtakia A., "Julia sets of switched processes", *Computer & Graphics*, Vol. 15, No. 4,1991, pp. 597-599.

12. Michelitsch M., Rossler E.O., "The 'burning ship' and its quasi-julia sets", *Computer & Graphics*, Vol. 15, No. 4, 1992, pp 435-438.

13. Kim S.H., Kim H.O., and Shin S.Y., "Image generation by blaschke products in the unit disk", *Computer & Graphics*, Vol. 17, No. 4, 1993, pp 489-494.

14. Dewdney A.K., "Computer Recreations", Scientific American, july 1989,pp. 92-95.

15. Stuedell D., "Biomorphic mitosis ", *Computer & Graphics*, Vol. 15, No. 3, 1991, pp. 455.

16. Michael Levin, "Discontinuous and alternate Q-system Fractals", *Computer & Graphics,* Vol. 18, No. 6, 1994, pp. 873-884.

17. Bowman R.L.,"Fractal metamorphosis : a brief student tutorial", *Computer & Graphics,* Vol. 19, No. 1, 1995, pp. 157-164.

18. Nettleton D.j. and Garigliano R., "Evolving fractals", *Computer & Graphics*, Vol. 19, No.5, 1995, pp. 779-782.

19. Monro M.D.,Dudbridge F.,"Rendering algorithms for deterministic fractals", *IEEE Computer & Graphics,* 1995, pp. 32-41.

20. Goertzel Ben, "Rapid generation of strange attractors", *Computer & Graphics,* Vol. 19, No.1, 1995, pp. 151-156.

21. Sprott J.C., "Automatic generation of strange attractors", *Computer & Graphics,* Vol. 17, No. 3, 1993, pp. 325-332.

22. Sprott J.C.,"Automatic generation of general quadratic map basins", *Computer & Graphics,* Vol. 19, No.2, 1995, pp. 309-313.

23. Pickover A.C., "The crying of fractal batrachion 1,489", *Computer & Graphics*, Vol. 19, No.4, 1995, pp. 611-615.

24. Raul Rojas, "A tutorial of efficient Computer Graphics representations of the Mandelbrot Set", *Computer & Graphics*, Vol. 15, No. 1, 1991, pp. 91-100.

25. Hooper Kenneth J., "A note on some internal structures of the Mandelbrot Set", *Computer & Graphics*, Vol. 15, No. 2, 1991, pp. 295-297.

26. Prusinkiewicz P. and Sandness G., "Koch curves as attractors and repellers", *IEEE Computer Graphica and Applications*, 1988, pp. 26-40.

27. Bhavsar V.C.,Gujar U.G.,Choi S.Y.M.,Kalra P.K., "Traversed geometric fractals", *IEEE Computer & Graphics and Applications*, 1993, pp. 61-67.

28. Casey Stephen D., Reingold Nicholas, "Self Similar Fractals Sets : Theory and Procedure", *IEEE Computer & Graphics and Applications*, May 1994, pp. 73-81.

29. David H., "Two fractals based on Keplerian solids", *Computer & Graphics*, Vol. 19, No. 6,1995, pp. 885-888.

30. Steven R.T., "Graphics programming in C", *BPB Publications*, 1993.

31. Foley J.D.,Dam A.V.,Hughes J.K., "Computer Graphics, Principals and Practices", *Addison Wesley*, Reading MA, 1992.

32. Dhurandhar S.V.,Bhavsar V.C.,and Gujar U.G.,"Analysis of Z-plane Fractal Images from $Z \leftarrow Z^{\alpha}+C$ for $\alpha<0$", *Computer & Graphics*, Vol. 17, No. 1, 1993, pp. 89-94.

33. Fournier A., Fussel D.,and Carpenter L.C., "Computer Rendering of Stochastic Models", *Communication ACM*, Vol. 25,No. 6,1982,pp. 371-384.

34. Hill F.S., "Computer Graphics", *Macmillan Publising Co.*, New York, 1990.

35. Mukherjee B.K., "Fractal- the engmatic geometry of nature", *Invention Intelligence*, 1996, pp. 495-507.

36. Hearn Donald, Baker M. Pauline, "Computer Graphics", *Prentice Hall Publication*, 1996.

37. Y.Fisher, "Fractal Image Compression", In *SIGGRAPH'* 92,1992, Fractal Course Notes.

38. Mehta Dewang, "Miss India, Fractals and cable TV", *Computer and communications,* March 1994.

BIBLIOGRAPHY

1. Peitgen, H.O., Ritcher P.H., "The Languages of fractals", *Scientific American*, August 1992, pp. 40-47.
2. Clifford, A. Pickover, "*Computers, Patterns, Beauty and Chaos*" St. Martin's Press, New York(1990).
3. K. Falconer, "*Fractal Geometry*", Wiley, Chichester, UK(1990).
4. Clifford, A. Pickover, "Biomorphs: Computer Displays Of Biological Forms Generated From Mathematical Feedback Loops", *Computer Graphics Forum*, Vol. 5, pages 313-316: 1986.
5. Nortan, V.A., "Generation and display of geometric fractals in 3-D", *Computer Graphics* 16,61-67(1983).
6. Willson, S.J., "Cellular automata can generate fractals", *Preprint(*1982).
7. Mandelbrot, B.B., "Les Objects Fractal :Forme ", W.H. Freeman, San Fransisco, CA, 1982.
8. Mandelbrot, B.B., "Fractals: Form, chance and Dimension", W. H. Freeman, San Fransisco, CA, 1977.
9. Rammal, R. & Toulouse, G.," Random walks on fractal Structures and percolation clusters, *Preprint*.
10. J.E. Hutchinson, "Fractals and self-similarity", *Indiana University Mathematics Journal,*3(5);pp 713-747,1981.
11. K. Culik and J. Kari, "Computational Fractal Geometry", 1993.