Table of Contents – ICS4M0 Final Culminating Activity

ICS4M0 Final Culminating Activity

Brief Summary of what you need to do

Each student will write a Java program that ***generates*** a ***coloured fractal***.

Due Date and Weighting

Your software is due on the ***last day of classes***. It is worth ***10 marks out of 30*** for your final evaluation (10% overall).

How to hand in your Software

Store all files related to your software in a ***single folder***. When you are satisfied that you have completed your work to the best of your ability, ***copy*** the folder to **I:\In\Nolfi\Ics4m0\Final Culminating Activity***YourName*, where *YourName* stands for your name.

Evaluation Criteria

The fractal program that you develop will be judged according to the following criteria:

Coding Practices (Style)

(a) The code should be logical, tidy and constructed according to the general guidelines learned throughout the course (i.e. proper indentation, comments for major blocks of code and abstruse code, meaningful identifier names, etc).
(b) The code should be as short as possible. Duplicate code should be eliminated by using methods and classes.

Difficulty of Coding

More credit will be given for fractals that are difficult to code than for those that are easy to code.

Attractiveness of Fractal

More credit will be given for fractals that are attractive and smoothly coloured than for those that are not.

Degree of Mathematical Complexity

More credit will be given for using fractal algorithms that are mathematically complex than for those that are not.

What on Earth is a Fractal?

A mathematically precise definition of fractals requires knowledge of mathematics that is far beyond the high school level. Therefore, we shall only consider an intuitive definition, which will allow us to understand the essential ideas without being encumbered by the complexities of mathematical technicalities.

In a colloquial sense, the term ***fractal*** denotes a shape that is ***recursively constructed*** or ***self-similar***, that is, a shape that appears similar at all scales of magnification and is therefore often referred to as "infinitely complex" (*definition taken from Wikipedia*).

Classification of Fractals

Adapted from a Wikipedia article

Fractals can be classified according to their self-similarity. There are three types of self-similarity found in fractals:

- ***Exact Self-Similarity***
  This is the strongest type of self-similarity; the fractal appears identical at all scales. Fractals defined by iterated function systems often display exact self-similarity.
- ***Quasi-Self-Similarity***
  This is a loose form of self-similarity; the fractal appears approximately (but not exactly) identical at all scales. Quasi-self-similar fractals contain small copies of the entire fractal in distorted and degenerate forms. Fractals defined by recurrence relations are usually quasi-self-similar but not exactly self-similar.
- ***Statistical Self-Similarity***
  This is the weakest type of self-similarity; the fractal has numerical or statistical measures that are preserved across all scales. Random fractals are examples of fractals that are statistically self-similar, but neither exactly nor quasi-self-similar.

Fractals in Nature

Adapted from a Wikipedia article

Approximate fractals are easily found in nature. These objects display self-similar structure over an extended, but finite, scale range. Examples include ***clouds***, ***snow flakes***, ***mountains***, ***river networks*** and ***systems of blood vessels***. ***Trees*** and ***ferns*** are fractal in nature and can be modelled on a computer using recursive algorithms. The recursive nature is clear in these examples — a branch from a tree or a frond from a fern is a miniature replica of the whole, not identical, but similar in nature.

A fractal is formed when pulling apart two glue-covered acrylic sheets.

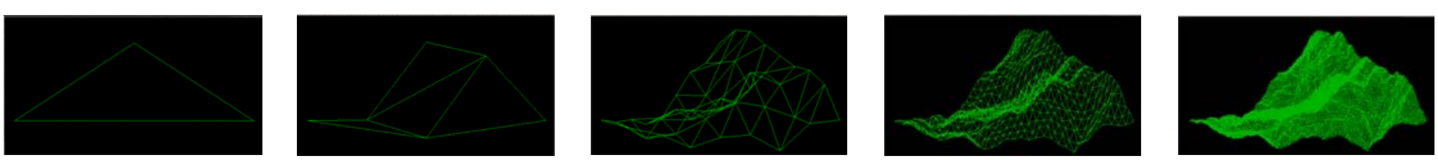
High voltage breakdown within a 4″ block of acrylic creates a fractal Lichtenberg figure.

Fractal branching occurs on a microwave-irradiated DVD

Romanesco broccoli showing very fine natural fractals

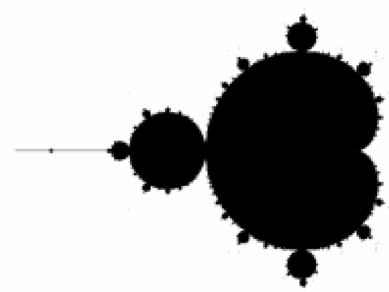
A fractal fern computed using an Iterated function system

The surface of a mountain can be modelled on a computer using a fractal. Start with a triangle in 3D space and connect the central points of each side by line segments, resulting in 4 triangles. The central points are then randomly moved up or down within a defined range. The procedure is repeated, cutting the range in half after each iteration. The recursive nature of the algorithm guarantees that the whole is statistically similar to each detail.

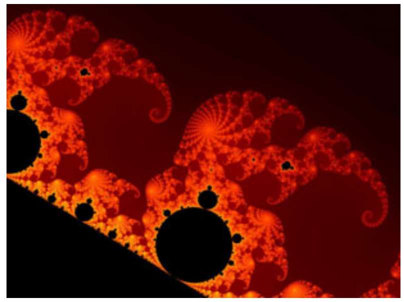


A Famous Fractal - The Boundary of the Mandelbrot Set

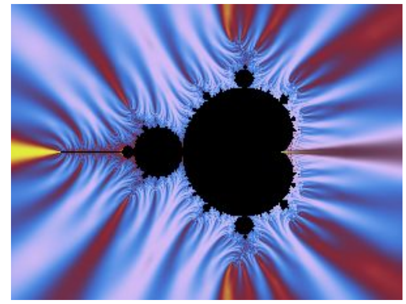
The Geometry of the Mandelbrot Set



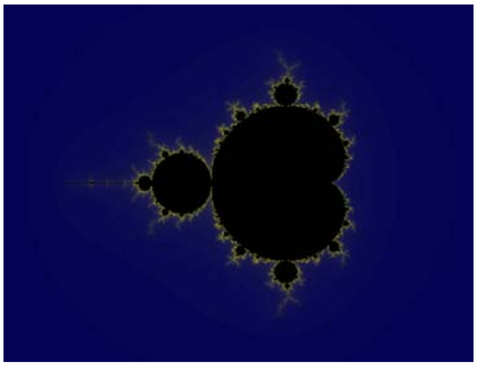
The Mandelbrot Set
Notice the self-similarity at several scales.



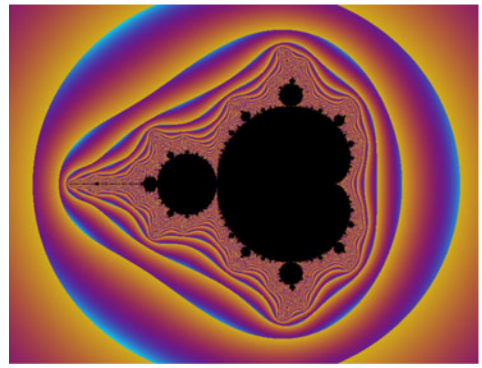
A Close-up View of the Boundary
Self-similarity is evident here at a tiny scale.



The Exterior of the Mandelbrot Set Coloured using the "Triangle Inequality" Method



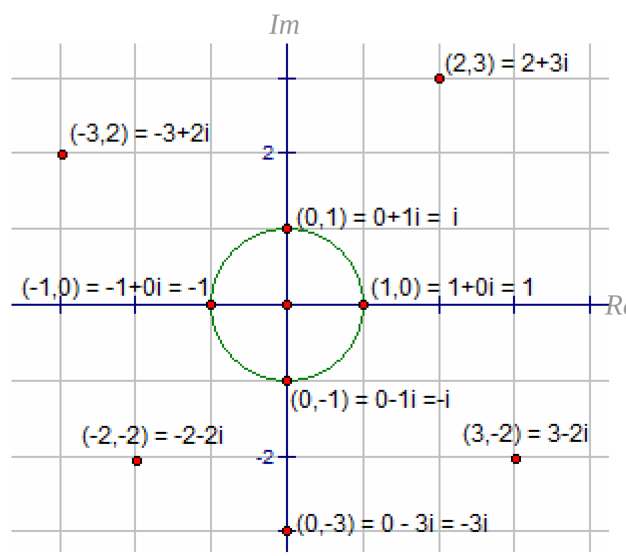
The Exterior of the Mandelbrot Set Coloured using the "Iterations" Method



The Exterior of the Mandelbrot Set Coloured using the "Modulus" Method

A Primer on Complex Numbers

To understand how the Mandelbrot set is generated, it is necessary to have a basic understanding of ***complex numbers***. Complex numbers are of the form $a+bi$, where $a\in\mathbb{R}, b\in\mathbb{R}$ and $i=\sqrt{-1}$. The real number *a* is called the ***real part*** of $a+bi$ and the real number *b* is called the ***imaginary part*** of $a+bi$.



Since $i=\sqrt{-1}$, it follows that $i^2=-1$. Due to our intimate familiarity with the real numbers, which have the property that $x^2\geq 0$ for all $x\in\mathbb{R}$, this at first appears to be a peculiar or even absurd notion. However, once we become well acquainted with the ***geometry of complex numbers***, it becomes easier to accept the "reality" that the square of the imaginary number *i* actually equals -1.

Complex numbers are plotted by making use of the Cartesian plane. We only need to become accustomed to a few minor modifications.

- The Cartesian plane is renamed the ***complex plane***.
- The *x*-axis is renamed the ***real axis***.
- The *y*-axis is renamed the ***imaginary axis***.

Using this framework, it becomes possible to give a geometric meaning to multiplication by the imaginary number *i*:

Multiplication by *i* is equivalent to a counter-clockwise rotation by 90° about the origin.

Let's examine how this works by starting at the complex number 1 on the real axis and following the unit circle.

$$1i=i \text{ (i.e. } (1,0)\to(0,1)\text{)}$$

$$i(i)=i^2=-1 \text{ (i.e. } (0,1)\to(-1,0)\text{)}$$

$$-1i=-i \text{ (i.e. } (-1,0)\to(0,-1)\text{)}$$

$$-i(i)=-i^2=-(-1)=1 \text{ (i.e. } (0,-1)\to(1,0)\text{)}$$

Operations on Complex Numbers

Addition

$$(a+bi)+(c+di)$$
$$=(a+c)+(b+d)i$$

e.g.

$$(2-3i)+(-5-i)$$
$$=(2+(-5))+(-3+(-1))i$$
$$=-3-4i$$

Subtraction

$$(a+bi)-(c+di)$$
$$=(a-c)+(b-d)i$$

e.g.

$$(2-3i)-(-5-i)$$
$$=(2-(-5))+(-3-(-1))i$$
$$=7-2i$$

Multiplication

$$(a+bi)(c+di)$$
$$=ac+adi+bci+bdi^2$$
$$=(ac-bd)+(ad+bc)i$$

e.g.

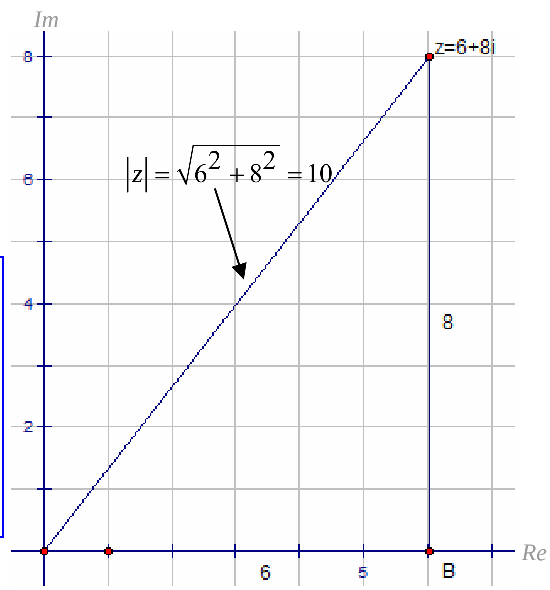
$$(2-3i)(-5-i)$$
$$=2(-5)+2(-i)-3i(-5)-3i(-i)$$
$$=-13+13i$$

Division

$$\frac{a+bi}{c+di}$$
$$=\left(\frac{a+bi}{c+di}\right)\left(\frac{c-di}{c-di}\right)$$
$$=\frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$$

The Modulus (Absolute Value) of a Complex Number

The ***modulus*** or ***absolute value*** of a complex number is an extremely important operation that is used to measure the "size" of a complex number. As shown in the diagram to the right, the modulus of a complex number *z*, denoted $|z|$, is equal to the ***distance from the origin to z***.



The following are some formal definitions, including the definition of $|z|$.

> ***Definitions***
>
> **1.** The symbol $\mathbb{C}$ is used to denote the set of complex numbers.
>
> **2.** Suppose that $z\in\mathbb{C}$, where $z=x+iy, x\in\mathbb{R}, y\in\mathbb{R}$. Then $\mathrm{Re}(z)=x$ denotes the ***real part of z*** and $\mathrm{Im}(z)=y$ denotes the ***imaginary part of z***.
>
> **3.** The ***modulus*** or ***absolute value*** of $z=x+iy$ is denoted $|z|$ and is equal to $\sqrt{x^2+y^2}=\sqrt{\mathrm{Re}(z)^2+\mathrm{Im}(z)^2}$

The Mandelbrot Sequence

For any ***fixed value*** $c\in\mathbb{C}$, consider the ***Mandelbrot sequence***, which is defined ***recursively*** as follows for all $n\in\mathbb{N}$:

$$\begin{cases} z_n=0, \text{ if } n=1 \\ z_{n+1}=z_n^2+c, \text{ if } n\geq 2 \end{cases}.$$

For a particular value of *c*, there are ***two*** possibilities.

1. The value of $|z_n|$ grows larger and larger indefinitely as *n* gets larger. That is, $|z_n|$ "blows up" to infinity.
2. There is a constant *D* such that the value of $|z_n|\leq D$ no matter how large *n* is made. In other words, in this case the value of $|z_n|$ remains bounded. It does not "blow up" to infinity.

The ***Mandelbrot set*** consists of all the values of $c\in\mathbb{C}$ for which the Mandelbrot sequence ***does not "blow up" to infinity***.

To generate a picture of the Mandelbrot set, the following is done:

1. If the chosen value of *c* causes $|z_n|$ to "blow up" to infinity, then ***c is not plotted*** on the complex plane.
2. If the chosen value of *c* does not cause $|z_n|$ to "blow up" to infinity, then ***c is plotted*** on the complex plane.

Formal Definition of the Mandelbrot Set

If we define the set $S_c=\{x\in\mathbb{R}: x=|z_n| \text{for all } n\in\mathbb{N}\}$ according to the value chosen for *c*, then the ***Mandelbrot set*** can be defined as the set *M*, where $M=\{c\in\mathbb{C}:\sup S_c\neq\infty\}$. (Note that $\sup A$, read "the supremum of *A*," is simply the ***least upper bound of A***. In other words, $\sup A$ is the smallest value that is larger than or equal to all the elements of *A*.)
In other words, the Mandelbrot set consists of all $c\in\mathbb{C}$ such that the Mandelbrot sequence is ***bounded***. It can be shown that for a chosen value of $c\in\mathbb{C}$, if $|z_n|>2$ for any $n\in\mathbb{N}$, then $\sup S_c=\infty$. ***Therefore, all values*** $c\in M$ ***must satisfy*** $|z_n|\leq 2$ ***for all*** $n\in\mathbb{N}$.

As usual, a ***specific example*** should help to clarify matters. Suppose that we choose $c = 0.5 + 0.5i$. The following table, constructed using Microsoft Excel, gives the values of z_n and $|z_n|$ for $n = 1, \ldots, 14$.

n	z_n	$\|z_n\|$
1	0	0
2	0.5 + 0.5i	0.70711
3	0.5 + i	1.11803
4	-0.25 + 1.5i	1.52069
5	-1.6875 – 0.25i	1.70592
6	3.28515625 + 1.34375i	3.54935
7	9.48658752441406 + 9.328857421875i	13.305
8	3.46776206069622 + 177.498044870794i	177.53192
9	-31493.0305592448 + 1231.54197170139i	31517.10122
10	990294278.677464 – 77569977.3995685i	993327669.9
11	9.74665656987549E+017 – 1.53634209631866E+017i	9.867E+17
12	9.26369672541763E+035 – 2.99483975733211E+035i	9.73577E+35
13	7.68470118484163E+071 – 5.5486574506296E+071i	9.47851E+71
14	2.8267032795879E+143 – 8.52795489702672E+143i	8.9842E+143

For $c = 0.5 + 0.5i$, we see that the Mandelbrot sequence is not bounded. After 6 iterations, $|z_n|$ is already greater than 2 and by 14 iterations, $|z_n|$ explodes to a value greater than 10^{43} googols! Therefore, $c = 0.5 + 0.5i$ is ***not*** in the Mandelbrot set and so, it is ***not plotted*** on the complex plane.

Now let's see if $c = 0.1 + 0.2i$ fares any better than $c = 0.5 + 0.5i$.

n	z_n	$\|z_n\|$
1	0	0
2	0.2 + 0.1i	0.22361
3	0.23 + 0.14i	0.26926
4	0.2333 + 0.1644i	0.28541
5	0.22740153 + 0.17670904i	0.28799
6	0.220485371028619 + 0.180367812121662i	0.28486
7	0.216081251188073 + 0.17953692795453i	0.28094
8	0.214457598615653 + 0.177589128053755i	0.27844
9	0.2144541632011 + 0.176170675885312i	0.27754
10	0.214954481072396 + 0.175561069755114i	0.27754
11	0.215383739719543 + 0.175475277291451i	0.27782
12	0.215598582395064 + 0.175589042902713i	0.27805
13	0.21565123674327 + 0.175713497467862i	0.27817
14	0.215630222716514 + 0.17578566608286i	0.27820

In this case, after 14 iterations $|z_n|$ remains very small, which makes it very likely that $c = 0.1 + 0.2i$ ***is*** a member of the Mandelbrot set. Since the Mandelbrot sequence appears to be bounded for $c = 0.1 + 0.2i$, then this point ***is plotted*** on the complex plane.

Colouring the Exterior of the Mandelbrot Set

When it comes to colouring, the exterior of the Mandelbrot set is where all the action is! This is particularly true near the boundary of the set. The points lying outside the boundary of the Mandelbrot set all have something in common; $|z_n|$ eventually "blows up" to infinity. More importantly for these points, however, is that $|z_n|$ does not always "blow up" to infinity at the same rate. For some points, $|z_n|$ goes to infinity rather slowly. For others, $|z_n|$ approaches infinity very rapidly. We can use this as the basis for colouring (**e.g.** iterations method).

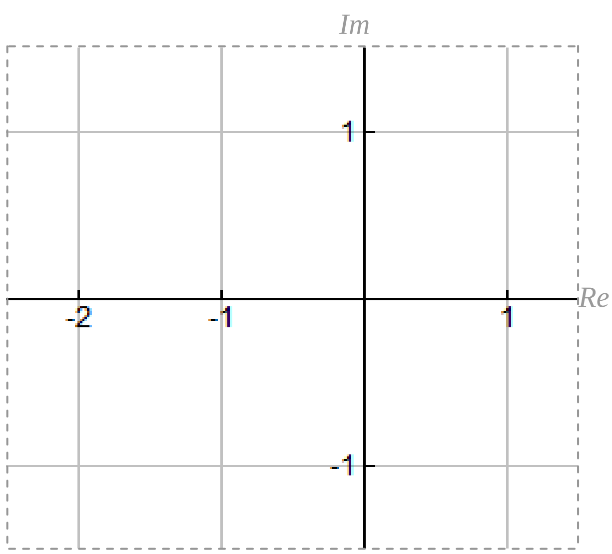
The following is a description of just a few colouring methods.

Iterations Method	*Modulus Method*	*Exponential Smoothing Method*
The Mandelbrot sequence is generated until $\lvert z_n \rvert$ exceeds a certain fixed value. The number of iterations required to exceed this value is then used to determine the colour of the pixel located at *c* on the complex plane.	The Mandelbrot sequence is generated until $\lvert z_n \rvert$ exceeds a certain fixed value. The value of $\lvert z_n \rvert$ is then used to determine the colour of the pixel located at *c* on the complex plane.	On a small scale (i.e. high degree of magnification), the "iterations" method can lead to colour "banding" due to the rather abrupt transition from one colour to another. To prevent this problem, a second sequence is computed at the same time as the Mandelbrot sequence is generated: $$s_{n+1} = s_n + e^{-\lvert z_{n+1} \rvert}$$ The value of s_n is used to determine the colour of the pixel located at *c* on the complex plane. This allows for smoother colour transitions on a minute scale.

Other colouring methods include *decomposition*, *binary decomposition*, *orbit traps*, *direct orbit traps*, *distance estimator*, *Gaussian integer*, *gradient*, *triangle inequality average* and *lighting*.

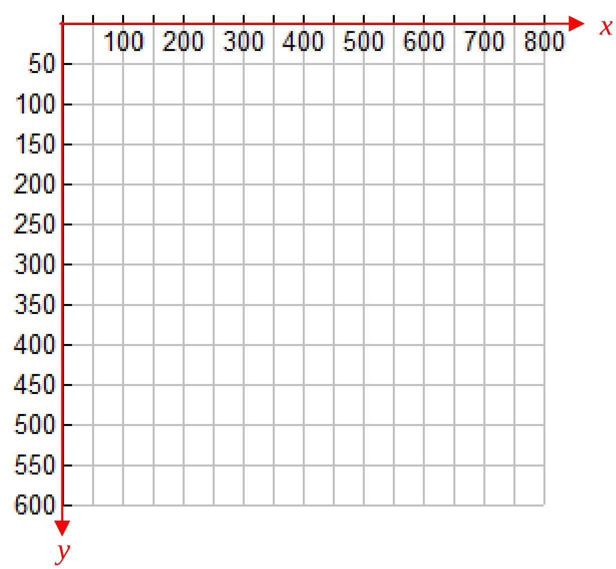
Writing a Java Program to Generate the Mandelbrot Set

The Mandelbrot set lies in a region of the complex plane that is very close to the origin. Generally, the points in the Mandelbrot set and its immediate exterior are plotted for real values ranging from -2.5 to 1.5 and for imaginary values ranging from -1.5 to 1.5. This poses a slight problem when writing computer programs because screen co-ordinates do not correspond to the ranges given above. Therefore, it is necessary to find equations that can translate between screen co-ordinates and actual complex plane co-ordinates.



The Mandelbrot set lies in this region of the complex plane.

To render the Mandelbrot set on a computer screen, co-ordinates in the range shown at the left must be translated to screen co-ordinates.

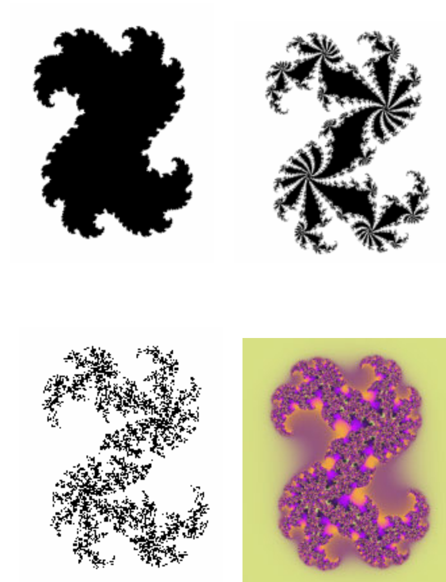
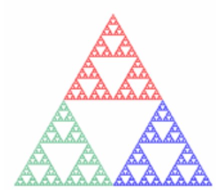
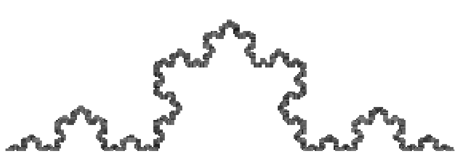
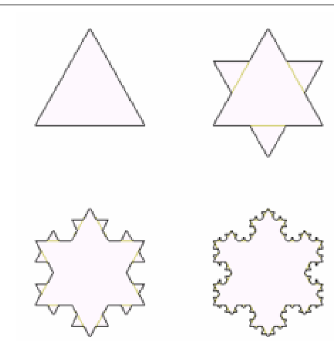


Co-ordinates in the Complex Plane		Screen Co-ordinates	
$\mathrm{Re}(z) = \dfrac{x-500}{200}$	$\mathrm{Im}(z) = \dfrac{y-300}{-200}$	$x = 200\,\mathrm{Re}(z) + 500$	$y = -200\,\mathrm{Im}(z) + 300$
-2.5	1.5	0	0
-1.5	0.75	200	150
-0.5	0	400	300
0.5	-0.75	600	450
1.5	-1.5	800	600

Using the data in the above table we can easily see how the co-ordinates in the complex plane and the screen co-ordinates are related. The relationships are nothing more than the familiar linear variety (i.e. $y = mx + b$).

Creating your own Fractal

Listed below are some examples of fractals that are appropriate for this final culminating activity. If you don't like any of these suggestions, you are free to choose any other fractal provided that your program can be completed in the time that is remaining. It would be a good idea to ask me about the appropriateness of your choice before forging ahead.

Fractal Name	***Description***	***Sample Picture(s)***
Julia Sets	Julia sets are closely related to the Mandelbrot set. As with the Mandelbrot set, the border of a Julia set is a fractal and its exterior can be coloured in a variety of interesting ways. Unlike the Mandelbrot set, there are an infinite number of different Julia sets. The black and white pictures at the right are three examples of different Julia sets. The fourth picture is a coloured version of the third Julia set. Only points that are just outside the Julia set are coloured. To generate a Julia set use the following algorithm: 1. Choose a point in the Mandelbrot set or just outside the Mandelbrot set. Call this value c. (The value chosen for c is known as the ***index*** of the Julia set.) 2. Choose z_1 in the complex plane in such a way that $-2 \leq \operatorname{Re}(z_1) \leq 2$ and $-1.5 \leq \operatorname{Im}(z_1) \leq 1.5$. 3. Using the value of z_1 chosen in step 2 and the value of c chosen in step 1, generate the resulting Mandelbrot sequence until $\lvert z_n \rvert$ exceeds 2 or a maximum number of iterations is exceeded. 4. If $\lvert z_n \rvert \leq 2$, then colour the pixel corresponding to z_1 black. Otherwise, set the colour according to the colouring scheme that you have chosen. 5. Repeat steps 2 to 4 until all pixels in the range have been coloured. (It is very important to understand that the value of c remains the same throughout the entire process.)	
Fractal Mountains	1. Begin with a triangle in 3-D space. 2. Find the co-ordinates of the midpoint of each side of the triangle. 3. Use line segments to connect the midpoints to each other. This produces 4 triangles. 4. Move each midpoint up or down by a randomly selected amount. 5. Repeat the same process on each of the four resulting triangles. 6. Stop when the triangles become "smaller" than some fixed value.	See page 2
Sierpinski Triangle	If you are interested in this one, do a search on "Sierpinski Triangle" to find out more.	
Koch Snowflake	Begin with a single line segment and then recursively alter each line segment as follows: 1. Divide the line segment into three segments of equal length. 2. Draw an equilateral triangle that has the middle segment from step 1 as its base. 3. Remove the line segment that is the base of the triangle from step 2.  The Koch Curve	 The first four iterations of the Koch Snowflake