

FRACTAL METHOD FOR NON-METAMORPHIC ANIMATION USING ITERATED FUNCTION SYSTEM ALGORITHM

¹DEWI ROSMALA, ²TEDJO DARMANTO, ³DELFIAN PUTRA CALIBRA

^{1,3}Department of Informatics Engineering, ITENAS Bandung, Bandung, Indonesia

²STMIK AMIK-Bandung, Bandung, Indonesia

E-Mail : ¹d_roskala@itenas.ac.id, ²tedjodarmanto@gmail.com, ³delfianputrac@gmail.com

ABSTRACT

In this research, the Fractal method is implemented for Non-Metamorphic animation using Iterated Function System Algorithm. This study aims to find out how implementation process of animation created by a fractal method with IFS algorithm. The method used in this design is the drawing and animating stage. The fractal method is used at the stage of drawing; therefore reading and calculation of the input of data values of the IFS codes are done. The coordinate points generated from the IFS code consisting of the dimensional coefficient relative to the frame and the values of the points so that the affine coefficient is obtained through calculating the IFS algorithm matrix and forming a fractal object. In the animating stage, the object that has been obtained from the drawing stage is processed by non-metamorphic animation process through calculating the number of locations and the duration between the points of the object location so that the object is seen moving from the point of the initial location to the point of the final location. Based on the results of the fractal method for testing IFS, it can be applied to the animation using an object of fractal which the best results requires a sufficient iteration value of 10000 times to form a full fractal object and the iteration process does not last long, and as well as the off set value search testing performed on various iteration tests, having an offset value average by 0.11735%.

Keywords: *Affine Coefficient, Drawing, Fractal, IFS Code, Non-Metamorphic Animation*

1. INTRODUCTION

Animation is a shaped picture of a set of objects (drawings) arranged regularly following a predetermined flow of movement at each additional time count [10]. The process of animation is an ordinary work done by an animator, which is where an animator first creates images or objects that are needed then combined them so that they become an animation or moving image.

The process of animation have problems encountered, such as lack of animator, obstacle in creation of animation application, and the size of the finished animated file took a huge amount of storage space, which makes the animated files lagging.

From these problems, the research conducted using fractal method in the making of non-metamorphic animation. Iterated Function System is a fractal object construct algorithm that processes elements or IFS code to be decoded and iterated according to random probability to form an object and non-metamorphic animation. Animation

created by moving objects based on x and y coordinates.

It can be formulated several problems how to implement fractal method in creating fractal object using IFS algorithm and the process of animating objects to become non-metamorphic animation.

The scope of the problems are limited to the object or drawing is made in the form of a two-dimensional figure, especially triangle creation of objects using the IFS Fractal method. The animation is displayed in the web.

2. METHODS

There are 3 main theoretical foundations which are:

a. Animation

Animation is a moving picture of a set of objects (drawings) that arranged regularly follow the flow of a movement that has been determined at each increase in the count of time that occurred.

The images can be images of living things, inanimate objects, or texts.

According to Hofstetter, Animation is consisting of four types which are:

a) Frame Animation

Frame animation is an animation created by changing objects on each frame. These objects will be visible at different locations on the screen.

b) Vector Animation

Vector animation is an animation created by changing the shape of an object.

c) Computational Animation

Computational animation or also called Non-Metamorphic is an animation created by moving objects based on x and y coordinates. Coordinate x for horizontal position and y position for vertical position.

d) Morphing

Morphing or also can be called Metamorphic is the transition of one form of object to another object by manipulating more than one frame so that it will be generated a whole movement is very gentle to change one form to another.

Animation is also referred to a technique of displaying sequential images in such a way that the audience feels the illusion of movement (motion) in the displayed image. The illusion of movement is a change that is visually detected by the eye so that changes that occur do not have to be in the form of movement but can also be a change of color [10].

b. Fractal

Fractal refers to the object that part of the body is similar to the whole in some way, and has been considered as a powerful tool to study complex shapes and structures in nature [8].

The term 'fractal' was coined in 1975 by Benoit Mandelbrot (1924-) from the Latin fractus, meaning "broken" or "irregular." This term was used to describe shapes that have the characteristic of self-similarity, i.e. that when you magnify any part it looks just like (or has the same structure) as the original [7].

Fractals are discovered as the fixed points of certain set maps [1]. Fractals are very interesting and ubiquitous objects. They are very common in nature, from blood vessels, through plants, coastlines, and lightning to the structure of the Universe. But, as it turns out, fractals are also observed in a street network or literary works [9].

c. Iterated Function System

Michael Barnsley (1988) represents fractals into mathematical models called the Iterated

Function System (IFS), through books, Fractals Everywhere. IFS is modeled as a photocopy machine called Multiple Reduction Copy Machine (MRCM). MRCM has a lot of lenses and every lens does a lot of image reduction. The image generated from the copy machine is restarted as an input to make the next copy [6].

Iterated Function System (IFS) is a collection of contracting operators that act on subsets of vector spaces. And, as such, they have a fixed point which also a subset of the underlying space. The fixed points of the IFS can be obtained by iterative processes with a random selection of operators from the IFS and can, therefore, be defined algorithmically. Very complex sets may, therefore, be defined with very few parameters (that specify the operators in the IFS.) [11].

A two-dimensional fractal object can be encoded into a two-dimensional iteration function code (2D IFS) that represents a set of transformation coefficients. The IFS code is the set of affine transformations of any contractive function that can be translated into fractal imagery in accordance with the collage theorem and property of the likeness [2]. Mathematically, the IFS 2D code can be represented by the affine transformation equations with 'a', 'b', 'c', 'd', 'e' and 'f' as the coefficients of a transformation function :

$$w \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \quad (1)$$

The collection of IFS codes as representations of fractal objects can be converted into fractal imagery by using deterministic algorithms. The solid portion of the image depends on the probability of the contractive function as the representation of the image portion. To construct fractal imagery, IFS codes with probability coefficients can be translated by random iteration algorithms [3].

The sum of the affine functions is calculated based on the number of lines of the IFS code. The IFS code obtained from the initialization is being calculated with the value of x and y that have been initialized to get the coefficient of affine value.

The number of lines in the text file input is the sum of the affine object functions through the IFS code reading procedure. The total of the opportunity is being obtained from the chances of the affine function when doing the iteration process. The part is done through the variables that become the last chance of previous affine transformation coefficient which is added at the beginning of the chance range of affine transformation coefficient

value so that the point coordinate value of the affine coefficient becomes random and different.

In the IFS decode procedure, the pixel or point is distributed according to the odds at the coordinate location for each value of the affine coefficient function transformed through the IFS code to perform the pixel image process and iteratively decodes using the matrix search function of the affine transformation coefficient value.

The value of w or affine transformation coefficient value is the coordinate of the pixel image location that has been processed through the IFS decode, and is derived from the calculation of a line of affine functions initialized at the beginning (a , b , c , d , e , and f) and the values of x and y , to get the coordinates of the location to be drawn on a canvas, frames, and so on.

2.1. Designing

The design used in this research is divided into two stages, which are drawing stage and the animation stage. The drawing stage focuses on the implementation of the fractal method, which reads and calculates the data of the elements needed to find the value of the affine transformation coefficient obtained by computing the IFS algorithm matrix. The value of affine transformation coefficient that has been obtained has the iteration process to get the location of pixel coordinates so as to form a fractal object.

At the animating stage, fractal objects that have been obtained from the stage of the image is animating by non-metamorphic animation process by calculating the location of the first coordinate object and the location of the final coordinates so that the object is seen moving from the initial location to another location.

2.1.1. Drawing Stage

In the drawing stage, the fractal object builder algorithm that processes the IFS code and other elements are used to generate the coefficient value of affine transformation to form the fractal object.

IFS codes consisting of the values of variables a , b , c , d , e , and f , are calculated with the values of x and y by using the search coefficient matrix function of affine coefficient contained in *Formula 1*.

The value of affine transformation coefficient that has been obtained, followed by the iteration process by calculating the value of affine coefficient with a scale that has been initialized at the beginning, so that obtained the coordinates of the pixel location on the frame or canvas.

The matrix function *formula 1* is the key in implementing fractal method this research, so the code in PHP language will be as follows:

```
$x1 = ($affine1[0] * $x1) + ($affine1[1] * $y1) + $affine1 [4];
$y1 = ($affine1[2] * $x1) + ($affine1[3] * $y1) + $affine1 [5];
```

Affine function used alone, is input that initialized before to in process of decode IFS. Here is an array of IFS codes variables:

```
$affine1 = array (0.5, 0, 0, 0.5, 1, 1);
$affine2 = array (0.5, 0, 0, 0.5, 1, 50);
$affine3 = array (0.5, 0, 0, 0.5, 50, 50);
```

And the next most vital code in implementing this fractal method is the code to represent an object point.

```
imagepixel ($ image, $ xnew, $ ynew, $ color);
```

Where $\$draw$ is the size of the image frame, $\$xnew$ and $\$ynew$ are the results of the affine function calculation and $\$color$.

```
$image = @imagecreate ($width, $height) or die ("Description");
```

$\$color$ is the predefined RGB color using the code:

```
$color = imagecolorallocate ($ bg, 0, 0, 0)
```

0 in the code that determines the color is the decimal value of a color, and the value of 0, 0, 0 is black.

2.1.2. Animating Stage

in the animating stage, the fractal object resulting from the drawing stage is processed to become non-metamorphic animation. Non-metamorphic is an animation created by moving objects based on x and y coordinates. Coordinate x for horizontal position and y position for vertical position, so in the animating process, a fractal object will be made to move from start location to another location.

The first process is to determine where the object is located on the canvas or frames along with the duration of time the object animation moves from the location to another location. In addition, the initialized animation time is divided into several chunks of time corresponding to the number of coordinate points of location that objects will be set.

From the amount of point duration that obtained, then determine how long the duration of time at each point by determining the percentage duration. By combining the overall animation settings containing the fractal object with the location of the initial location coordinates and other location coordinates where the location of the fractal object moves according to the division of the total duration of time into the duration of the portion as much as the coordinate point that being set. From the results that obtained when animation executed, a fractal object will be seen to moving.

3. DISCUSSION

In the drawing and animating process, the elements to be studied are as follows:

a. Frame Size

Frame size is the size of the frame or canvas that is where the pixel or fractal object is depicted to be displayed on the monitor screen.

b. IFS Code

The IFS code is a unique code to obtain the value of the affine transformation coefficient. IFS code consists of variables a, b, c, d, e, and f, which are important elements in the construction of fractal objects and function as a regulator of objects to be built

c. Values of x and y

x and y are the initialization values used to construct fractal objects that are computed by the IFS code to obtain the coefficient value of affine transformation.

d. Scale

Scale is the value for the scaling of the final coordinates obtained from the product times with the final result of IFS decoding.

e. Number of iterations

The number of iterations is the value used to determine the number of pixel images to construct fractal objects.

f. Time Duration

Time duration used in non-metamorphic fractal object animation to determine how long the animation process lasts from start to end.

g. Object location

Object location or coordinate locations is the location of the object during the animation process takes place. Next is discussion scenario. The first discussion scenario is the drawing stage by experimenting to insert the affine transformation coefficient calculation function to build the fractal object in order to display on the web. There are the discussion scenario at the drawing stage:

- Testing of 4 IFS code builders of different fractal objects, but the x and y values, scales, and iterations used are the same.
- Testing of 3 different x and y values, but same IFS, scale, and iteration codes.
- Testing of 3 different scale values, but the IFS code, the x and y values, and the iterations used the same.
- Testing of 7 different iteration values, but the IFS code, the values of x and y, and the scale used is the same.

- Testing the calculation of off set values on each fractal object constructed by doing 30 experiments on the same object according to the theory according to Roscoe (1975), the rule of thumbs.

While the second discussion scenario is the animating stage conducted experiments directly by creating animated images of fractal objects. Non-metamorphic animation results from fractal objects are displayed over the web. Here is the discussion scenario at the stage of animate is testing the fractal object in a total duration by 10 seconds and 4 points of object displacement locations.

Prior to the discussion of what is done during the study, from the two stages described in the design stage, first will be discussed about the flow of the research process described in *Figure 1*.

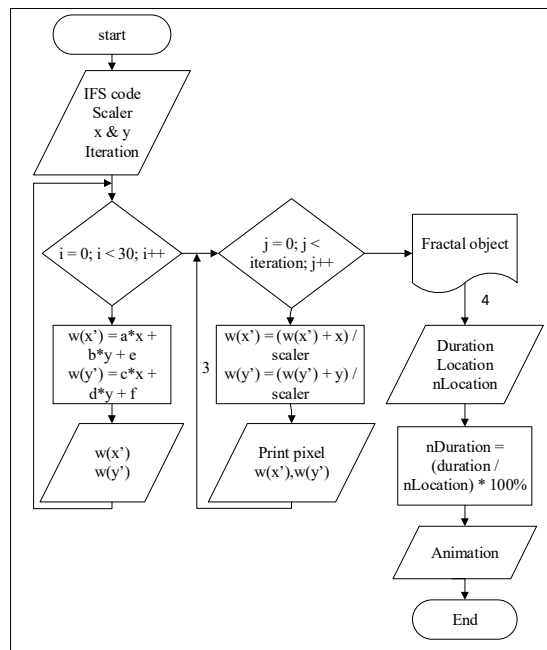


Figure 1. Flowchart Program

Explanation of the steps at each stage are described as follows:

- The first program begins by including the IFS code of the constructed fractal object (a, b, c, d, e, and f), along with the values of x and y, the scale, and the number of iterations.
- Second by entering into looping to find the value of affine transform coefficient (w(x') and w(y')) using IFS algorithm function, and the result data is stored temporarily to find the last w value, in this case, the last value is the thirtieth.
- The last w(x') and w(y') values called to be calculated by adding the first x and y values

randomly and divided by the scale entered, and the result of the new w value obtained is the location of the drawn pixel coordinates. This process is repeated as many inputs to the number of iterations illustrated an object.

- d. The finished object image is printed as input data in the animation process along with the input of the object location during non-metamorphic animation takes place, the number of location of the object during the animation, and the duration of the object for the animation process.
- e. The total duration used by the animation from the beginning to the end of the calculation of the division by the number of locations to find the duration of animation from each location.
- f. The image object that has been processed is displayed on the monitor display as the finished animation.

3.1. Data Set

Data collection is a step in an important process, because only by getting the right data then the research process will last until the researchers get answers from the formulation of the problem that has been set.

- a. Frame Size
The size data of the frame or canvas. The size data of the frame or canvas used is 300 x 300 pixels.
- b. IFS Code

Table 1. IFS code for a Sierpinski triangle.

Row of Affine	a	b	c	d	e	f
1	0.5	0	0	0.5	1	1
2	0.5	0	0	0.5	1	50
3	0.5	0	0	0.5	50	50

The IFS code in *Table 1* is a unique IFS code forming triangle and has three lines of IFS code or three coefficients of affine transformation.

Table 2. IFS code for a square.

Row of Affine	a	b	c	d	e	f
1	0.5	0	0	0.5	1	1
2	0.5	0	0	0.5	50	1
3	0.5	0	0	0.5	1	50
4	0.5	0	0	0.5	50	50

The IFS code in *Table 2* is a unique IFS code forming a square and has four lines of IFS code or four coefficients of affine transformation.

Table 3. Random IFS code for testing 2 lines of IFS code.

Row of Affine	a	b	c	d	e	f
1	0.5	0	0	0.5	1	1
2	0.5	0	0	0.5	50	50

The IFS code in *Table 3* is the IFS Code of any arbitrary object and has two lines of IFS code or

affine transformation coefficient. The goal is to test what objects are formed by using two lines of IFS code.

Table 4. Random IFS code for testing 3 lines of IFS code.

Row of Affine	a	b	c	d	e	f
1	0	0	0	0.5	0	1
2	0.42	0.32	0.22	0.5	1.6	25
3	0.5	0	0.1	0.5	10	60

The IFS code in *Table 4* is the IFS Code of any arbitrary object because the values of variables a, b, c, d, e, and f use random or analytic values and have three lines of IFS code or three coefficients of affine transformation. The goal is to test what objects are formed by using three lines of IFS code that have random IFS code variables.

- c. Values of x and y
The x and y values used in the test are x = 300 and y = 300, x = 100 and y = 100, and x = 500 and y = 500.
- d. Scale
Scale values used in the tests are 1, 2, and 3.
- e. Number of iterations
The number of iterations used in the test is 1, 10, 100, 1000, 10000, 100000, and 1 million.
- f. Time Duration
The time duration used in the animation testing from start to finish is for 10 seconds.
- g. Object location
The four coordinate point locations for the objects used in the test are coordinates (0,0), (0,450), (600, 450), and (0,0).

3.2. Drawing Stage Discussion

In the test of the drawing stage, testing is done by implementing the fractal method with Iterated Function System algorithm to be displayed on the web. Testing is done by using PHP code to build an image by using point according to the definition of fractal itself.

3.2.1. IFS Code Testing

The first test is to test four different IFS codes, with x = 300 and y = 300, scale = 2, and iteration = 10000.

- a. Testing the IFS Code of Convergence of Triangle.

The first test of IFS code testing, tested by constructing a triangular-shaped fractal object. By using IFS code in *Table 1* and *Formula Function 1*, then performed calculations to find the affine transformation coefficient as follows:

$$w_{1.1}(x') = (0.5 * 300) + (0 * 300) + 1 = 151$$

$$w_{1.1}(y') = (0 * 300) + (0.5 * 300) + 1 = 151$$

↓

$$w_{1.2}(x') = (0.5 * 151) + (0 * 151) + 1 = 76.5$$

$$w_{1.2}(y') = (0 * 151) + (0.5 * 151) + 1 = 76.5$$

$$\begin{aligned} &\downarrow \\ &w_{1.30}(x') = 2 \\ &w_{1.30}(y') = 2 \\ \\ &w_{2.1}(x') = (0.5 * 300) + (0 * 300) + 1 = 151 \\ &w_{2.1}(y') = (0 * 300) + (0.5 * 300) + 50 = 200 \\ &\downarrow \\ &w_{2.2}(x') = (0.5 * 151) + (0 * 200) + 1 = 76.5 \\ &w_{2.2}(y') = (0 * 151) + (0.5 * 200) + 50 = 150 \\ &\downarrow \\ &w_{2.30}(x') = 2 \\ &w_{2.30}(y') = 100 \\ \\ &w_{3.1}(x') = (0.5 * 300) + (0 * 300) + 50 = 200 \\ &w_{3.1}(y') = (0 * 300) + (0.5 * 300) + 50 = 200 \\ &\downarrow \\ &w_{3.2}(x') = (0.5 * 200) + (0 * 200) + 50 = 150 \\ &w_{3.2}(y') = (0 * 200) + (0.5 * 200) + 50 = 150 \\ &\downarrow \\ &w_{3.30}(x') = 100 \\ &w_{3.30}(y') = 100 \end{aligned}$$

From the calculation result, the value of affine transformation coefficient for IFS code line one is $w(x') = 2$ and $w(y') = 2$, line two is $w(x') = 2$ and $w(y') = 100$, and line three is $w(x') = 100$ and $w(y') = 100$. After that, the result of the w value obtained is entered into the calculation for the pixel depiction process as much as the iteration value and produces the object as in *Figure 2*.

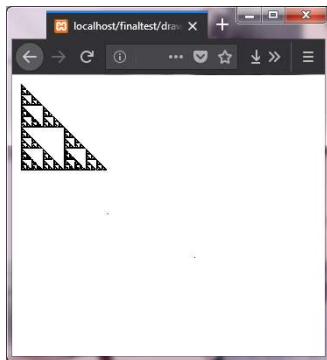


Figure 2. IFS Triangle Code Testing.

b. Testing the IFS Code of Convergence of Square.

The second test of IFS code testing, tested by constructing a square-shaped fractal object. By using almost the same calculation to find the coefficient of affine transformation by using the IFS code in *Table 2*, the resulted values of $w(x')$ and $w(y')$ line one are 2 and 2, line two are 100 and 2, line three are 2 and 100, and line four are 100 and 100. With the value obtained previously, produces the object as in *Figure 3*.

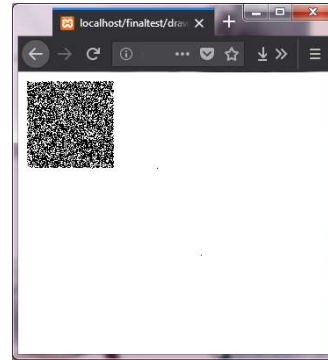


Figure 3. IFS Square Code Testing.

c. Testing the IFS Code of Convergence of 2 IFS Codes.

The third test of IFS code testing, constructing a fractal object using 2 lines IFS codes. By using almost the same calculation to find the coefficient of affine transformation by using the IFS code in *Table 3*, the resulted values of $w(x')$ and $w(y')$ line one are 2 and 2, and line two are 100 and 100. With the value obtained previously, produces the object as in *Figure 4*.

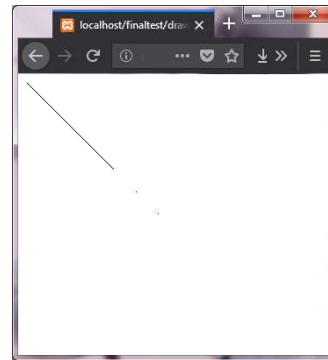


Figure 4.2 Lines IFS Code Testing.

d. Testing the IFS Code of Convergence of 3 random IFS Codes.

The fourth or last test of IFS code testing, constructing a fractal object using 3 lines IFS codes. By using almost the same calculation to find the coefficient of affine transformation by using the IFS code in *Table 4*, the resulted values of $w(x')$ and $w(y')$ line one are 0 and 3, line two are 40 and 68, and line three is 20 and 124. With the value obtained previously, produces the object as in *Figure 5*.

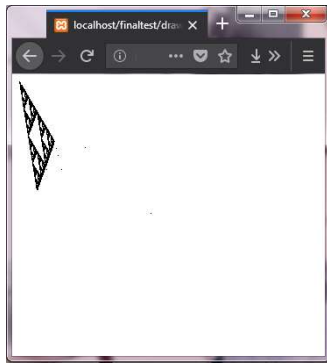


Figure 5. 3 Lines Random Variable IFS Code Testing.

3.2.2. Testing values of x and y

The second test carried out is testing three different things x and y in forming a fractal object. the values of x and y to be tested are as follows :

- x = 100 and y = 100
- x = 300 and y = 300
- x = 500 and y = 500

By using the IFS code in Table 1 and the same scale and iteration values to find the value of the affine transform coefficient, here is the calculation on the first row with different x and y values.

$$w_{100.1}(x') = (0.5 * 100) + (0 * 100) + 1 = 51$$

$$w_{100.1}(y') = (0 * 100) + (0.5 * 100) + 1 = 51$$

↓

$$w_{100.2}(x') = (0.5 * 51) + (0 * 51) + 1 = 26.5$$

$$w_{100.2}(y') = (0 * 51) + (0.5 * 51) + 1 = 26.5$$

↓

$$w_{100.30}(x') = 2$$

$$w_{100.30}(y') = 2$$

$$w_{300.1}(x') = (0.5 * 300) + (0 * 300) + 1 = 151$$

$$w_{300.1}(y') = (0 * 300) + (0.5 * 300) + 1 = 151$$

↓

$$w_{300.2}(x') = (0.5 * 151) + (0 * 151) + 1 = 6.5$$

$$w_{300.2}(y') = (0 * 151) + (0.5 * 151) + 1 = 76.5$$

↓

$$w_{300.30}(x') = 2$$

$$w_{300.30}(y') = 2$$

$$w_{500.1}(x') = (0.5 * 500) + (0 * 500) + 1 = 251$$

$$w_{500.1}(y') = (0 * 500) + (0.5 * 500) + 1 = 251$$

↓

$$w_{500.2}(x') = (0.5 * 251) + (0 * 251) + 1 = 126.5$$

$$w_{500.2}(y') = (0 * 251) + (0.5 * 251) + 1 = 126.5$$

↓

$$w_{500.30}(x') = 2$$

$$w_{500.30}(y') = 2$$

From the calculation results using different x and y values of 100, 300, and 500 on the test with the first line of the IFS code, the same affine transform coefficient is obtained, so that in the development of the fractal object will be the same. The result is made no difference as in Figure 6.

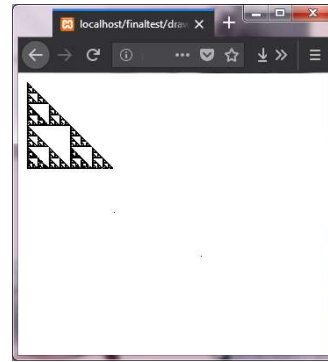


Figure 6. The result of testing values of x and y.

3.2.3. Scale Testing

The third test is 3 different scale tests to form a fractal object, with IFS code in Table 1, x = 300 and y = 300, and iteration = 10000.

- Scale Testing = 1

The first test of scale testing, the test input scale = 1. The result of the constructed fractal object is shown in Figure 7.

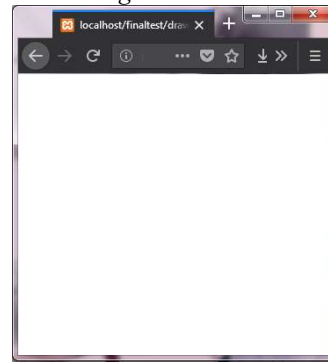


Figure 7. Result of scale = 1

In Figure 7, there is no visible fractal object. Because the built-in fractal object exceeds the size of the frame (300 x 300 pixels). Here is the coordinate location of the calculation of affine transformation coefficient in Table 5.

Table 5. Coordinates of pixel of scale = 1

Number of Iteration	Coordinate
1	302 , 400
2	304 , 500
3	306 , 600
4	308 , 602
5	310 , 702
..	..
..	..
9996	339968 , 674442
9997	339970 , 674444
9998	339972 , 674544
9999	339974 , 674644
10000	339976 , 674744

b. Scale Testing = 2

The second test of scale testing, the test input scale = 2. The result is the same as in *figure 2* and *Figure 6*. Because the result of the calculation of the affinity transformation coefficient is the same.

c. Scale Testing = 3

The third test of scale testing, the input test scale = 3. The result of the constructed fractal object is shown in *Figure 8*.

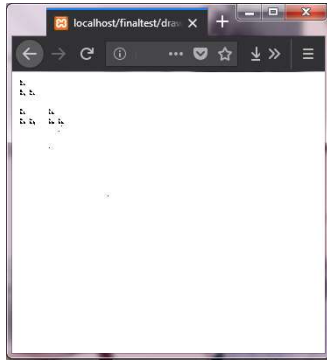


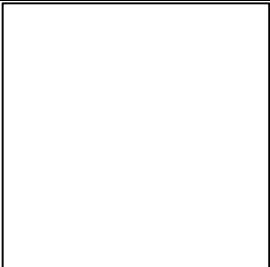
Figure 8. Result of scale= 3

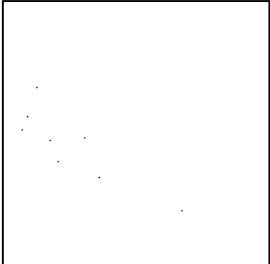
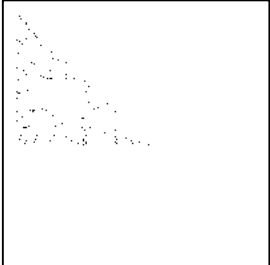
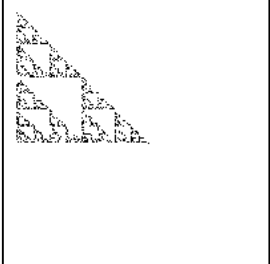
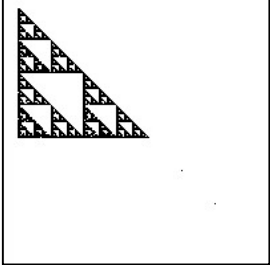
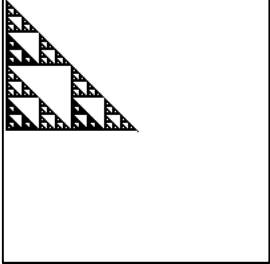
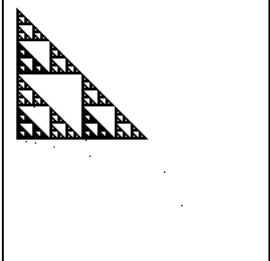
the result shown in *Figure 8*, looks fractal object its size becomes small. Since the value of the affine transformation coefficient is divided by scale = 3, it makes the location of the coordinates small.

3.2.4. Iteration Testing

The fourth test carried out was the testing of different iterative values, which were tested on 7 different iteration values in the forming of a fractal object, and the result is shown in *Table 6* which shows the result of the construction of the fractal object. The size of the frame or canvas used is 200 x 200 pixels and the x, y coordinates tested are 300,300 and IFS code in *Table 1*.

Table 6. Summary of test results on the number of iterations.

No.	Fractal Images	Number of Iterations
1		Iteration = 1

2		Iteration = 10
3		Iteration = 100
4		Iteration = 1000
5		Iteration = 10000
6		Iteration = 100000
7		Iteration = 1000000

The first test of iteration testing, the test input iteration = 1. The result is no fractal object is built. But actually there is a pixel is imaged, but because the iteration value used is 1, making the object invisible because that is built with only 1 pixel.

The second test of iteration testing, the test input iteration = 10 which displays a number of pixel dots are illustrated.

The third test of iteration testing, the test input iteration = 100, the pixel image has already begun to form a triangle.

The fourth test of iteration testing, the test input iteration = 1000, pixels already form a triangle, but the result is still blurry.

The fifth test of iteration testing, the test input iteration = 10000, the triangle is perfectly formed.

The sixth test of iteration testing, the test input iteration = 100000, the triangular fractal object is still the same as in the fifth test, but the triangle object looks thicker.

The seventh test of iteration testing, the test input iteration = 1000000, just like in the sixth test, the triangle object looks thicker than the previous test result.

From six tests of drawing stages by testing the number of iterations from 1 to 1 million times, it produces images of fractal objects that are constructed. This shows that in the drawing process, the more the number of iterations the fractal object will appear more dense and thick or unlike a set of dots or pixels

3.2.5. Off set value testing

The off set value is the result of the iteration where the pixel is located away from the fractal object. In the test to obtain the test result on the off-set value, the test was carried out for 30 samples and the research was also done with 4 different iterations, but the IFS code, the x and y values, and the same scale.

a. Off set value of iteration = 1000

Testing the value of the first set off is 1000 times iteration, the results of the percentage in Table 7 as follows:

Table 7. Off set value of iteration = 1000

Number Off set	Occurrences	Percentage
1	3 times	3/30 * 100% = 10%
2	6 times	6/30 * 100% = 20%
3	12 times	12/30 * 100% = 40%
4	1 time	1/30 * 100% = 3.3 %
5	4 times	4/30 * 100% = 13.3%
6	1 time	1/30 * 100% = 3.3%
7	2 times	2/30 * 100% = 6.6%
8	1 time	1/30 * 100% = 3.3%

From 30 times the experiment, the percentage of off set values is probable. From Table 7, the result of testing the off set value of 1000-times iteration that often occurs is 3 coordinates with possibly 40%, then two coordinates with possibly 20%, 5 coordinates with 13.3% possibility, 1 coordinate with 10% possibility, 7 coordinates with a probability of 6.6%, and for the offset values of 4, 6, and 8 the coordinates only appear with a probability of 3.3% of 30 trials.

Then the percentage of off set values obtained will yield the following off set possibility equation :

$$\text{Possible Off set} = \frac{(\text{Total Occurrences}) / \text{Experiment}}{\text{Iterate}} * 100\%$$

Thus, the calculation is obtained:

$$\text{Possible Off set} = \frac{((1 * 3) + (2 * 6) + (3 * 12) + (4 * 1) + (5 * 4) + (6 * 1) + (7 * 2) + (8 * 1)) / 30}{1000} * 100\%$$

Possible Off set = 0.343% or average 3.43 value

The calculation result from the equation to find the percentage of the possibility of the number of off set value for iteration as much as 1000 times that is 0.343% of the total iteration.

b. Off set value of iteration = 5000

The test of the first set off value is the iteration 5000 times, the results of the percentage in Table 8 as follows:

Table 8. Off set value of 5000 iterations

Number Off set	Occurrences	Percentage
1	4 times	4/30 * 100% = 13.3%
2	7 times	7/30 * 100% = 23.3%
3	7 times	7/30 * 100% = 23.3 %
4	6 times	6/30 * 100% = 20 %
5	2 times	2/30 * 100% = 6.6 %
6	3 times	3/30 * 100% = 10 %
7	1 time	1/30 * 100% = 6.6%

From 30 times the experiment, then got the percentage of possible off set value obtained. From Table 8, the result of the value of off set of iteration 5000 times that often appears are 2 and 3 coordinates with the possibility of 23.3%, then 4 coordinates with possibly 20 %, 1 coordinate with 13.3% possibility, 6 coordinates with 10%, 5 coordinates with a probability of 6.6%, and for a set off value of 7 coordinates only appear with possibly 3.3% of 30 attempts.

Then the percentage of off set values obtained will yield the following off set possibility equation :

$$\text{Possible Off set} = \frac{(1 * 4) + (2 * 7) + (3 * 7) + (4 * 6) + (5 * 2) + (6 * 3) + (7 * 1)}{30} * 100\%$$

Possible Off set = 0.0653% or an average of 3.265 values

The calculation results from the equation for looking percentage of the possibility of the number of off set values for iteration as much as 5000 times that is 0.0653% of the total iteration.

c. Off set value of Iteration = 10000

The first set off value test is 10000 times iteration, the results of the percentage in Table 9 as follows:

Table 9. Off set value of 10000 iteration

Number Off set	Occurrences	Percentage
1	1 time	1/30 * 100% = 3.3%
2	4 times	4/30 * 100% = 13.3%
3	13 times	13/30 * 100% = 43.3%
4	4 times	4/30 * 100% = 13.3%
5	3 times	3/30 * 100% = 10%
6	3 times	3/30 * 100% = 10%
8	2 times	2/30 * 100% = 6.6%

From 30 times the experiment, then got the percentage of possible off set value obtained. From Table 9, the result of testing the off-set iteration 10000 times that often arises is as much as 3 coordinates with possibly 43.3%, next is 2 and 4 coordinates with 13.3% probability, 5 and 6 coordinates with 10% probability, 8 coordinates with 6.6%, and for off set value 1 coordinate only appears with possibly 3.3% of 30 experiments.

Then the percentage of off set values obtained will yield the following off set possibility equation :

$$\text{Possible Off set} = \frac{((1 * 1) + (2 * 4) + (3 * 13) + (4 * 4) + (5 * 3) + (6 * 3) + (8 * 2))}{30} * 100\%$$

Off Possible set = 0.0376 % or an average 3.76 values.

The calculation results from the equation look for the percentage of the possibility of the number of off set for the iteration of 10000 times that is equal to 0.0376% of the total iteration.

d. Off set value of iteration = 15000

Testing the value of the first set off is the iteration 15000 times, the results of the percentage in Table 10 as follows:

Table 10. Off set value of 15000 iteration

Number Off set	Occurrences	Percentage
1	3 times	3/30 * 100% = 100%
2	8 times	8/30 * 100% = 26.6%
3	9 times	13/30 * 100% = 30 %
4	1 time	4/30 * 100% = 3.3%
5	3 times	3/30 * 100% = 10%
6	1 time	3/30 * 100% = 3 %
7	5 times	5/30 * 100% = 16.6%

From 30 times the experiment, then got the percentage of possible off set value obtained. From Table 10, the result of the test of the off-set iteration value of 15000 times that often arises is 3 coordinates with possibly 30%, then 2 with 26.6% possibility, 7 coordinates with 16.6% probability, 1 dam 5 coordinates with possibility 10%, 4 and 6 coordinates with 3% probability of 30 experiments.

Then the percentage of off set values obtained will yield the following off set possibility equation :

$$\text{Possible Off set} = \frac{(((1 * 3) + (2 * 8) + (3 * 9) + (4 * 1) + (5 * 3) + (6 * 1) + (7 * 5)) / 30)}{15000} * 100\%$$

Possible Off set = 0.0235% or 3.525 average value

The calculation results from the equation to find the percentage of the possibility of the number of off set values for iteration as much as 15000 times that is equal to 0.0235% of the total iteration.

From the off set value is found, it can also produce an average percentage and average of the number of values off set that often arise:

$$\text{Average percentage} = \frac{0.343 + 0.0653 + 0.0376 + 0.0235}{4} = \frac{0.4694}{4} = 0.11735\%$$

$$\text{Average off set} = \frac{3.43 + 3.265 + 3.76 + 3.525}{4} = \frac{13.98}{4} = 3.495$$

The calculation result from the average percentage obtained, ie 0.11735% and off set value obtained, ie 3.495 off set value.

3.3. Animating Stage Discussion

Animation stage test result is done by giving duration process and coordinate location of a fractal object to move during animation.

The coordinates tested were 4 coordinates consisting of the initial coordinates, the second coordinate, the third coordinate, and the final coordinates.

To provide an animation from one coordinate to another, given the duration of the coordinates obtained from the total duration during

the animation takes place divided by four which is the number of coordinates and then the duration of the coordinate in the conversion into the percentage of duration.

Table 11. Location Coordinate of Fractal Objects

No.	Coordinate	Duration at any point
1	0,0	2,5 s
2	0,450	2,5 s
3	600,450	2,5 s
4	0,0	2,5 s

With the data and calculations obtained, the implemented fractal object is processed into a non-metamorphic animation displayed on the monitor screen. The duration of the travel time from one location point to another location has a duration of 2.5 s or 25% of the total duration of the animation lasts from start to finish.

Previous data sets are calculated manually to see the processes that occur in the stage of animate. The total time duration of 10 seconds or 10 s and 4 object locations performs animation ie (0,0), (0,450), (600,450), and (0,0).

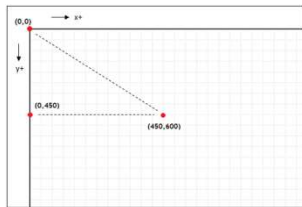


Figure 9. Location of point coordinates on the monitor screen

$$nDuration = 10/4 = 2.5 \text{ s}$$

$$nPercentageDuration = (2.5/10) * 100\% = 25\%$$

The duration of each animation from one location to another is 25%.

4. CONCLUSIONS

Based on research on the implementation of fractal method using Iterated Function System algorithm in making animation that has been done, obtained the following conclusion:

1. The iterative value affects the image detail of the constructed fractal object.
2. The IFS code variable is a unique code for building fractal objects and determining object shapes.
3. The x and y values do not affect the size of the fractal object image in construction.
4. The iteration process done more than 1000000 times will experience slow pixel depiction process, so that the fractal object construction process will be longer.

5. The scale value determines the size of the object of the fractal itself, the larger the value of the scale, the smaller the fractal object is formed, since the value of the affine transformation coefficient is divided by the value of the scale.
6. The percentage of pixels off the set of experiments using different iterations yields different percentages as follows:
 - a. Iteration 1000 with 30 times has the possibility of number of pixel off set by 0.343%.
 - b. Iteration 5000 with 30 times has the possibility of number of pixel off set by 0.0653%.
 - c. Iteration 10000 with 30 times has the possibility of number of pixel off set by 0.0376%.
 - d. Iteration 15000 with 30 times has the possibility of number of pixel off set by 0.0235%.

Of the four pixel offset test results, it is concluded in every process of constructing fractal objects having a percentage of pixel probabilities that set off less than 1% and the average constructed fractal object has a possible pixel value offset of 3.495 per fractal object created and average percentage off set by 0.11735%.

7. During testing, pixel off sets often occur in the first iteration until the eighth. In order to avoid pixel off sets reflected on the frame, a first to eighth iteration exception should be done in the looping syntax for pixel depiction.

REFERENCES

- [1] Barnsley, M. F., 2014, Fractals everywhere. Academic press.
- [2] Barnsley, M. F., and Demko, S., Iterated Function Systems and the Global Construction of Fractals, Proceedings of the Royal Society of London, Series A Mathematical and Physical Sciences, Vol. 399, No. 1817, June 8, 1985, pp. 243-275
- [3] Darmanto, T., Suwardi, I. S., & Munir, R., Cyclical metamorphic animation of fractal images based on a family of multi-transitional IFS code approach, IEEE In Control, Systems & Industrial Informatics (ICCSII), September 23-26, 2012, pp. 231-234
- [4] Darmanto, T., Suwardi, I. S., & Munir, R., Hybrid animation model of multi-object in fractal form based on metamorphic interpolation and partitioned-random iteration algorithms,

- International Journal on Electrical Engineering and Informatics, Vol. 5, No. 3, 2013, pp.285-296.
- [5] Furmanek, P., Examples of fractal objects generated as the union of terms of a sequence of sets using the ifs method. Journal of Polish Society for Geometry and Engineering Graphics, Vol. 27, December, 2015, pp.53-61.
- [6] Munir, Rinaldi., 2004, Pengolahan Citra Digital. Bandung: Informatika.
- [7] Snyder, S. S., 2006, Fractals and the Collage Theorem.
- [8] Yang, Fu., Zheng, Zeyu., Xiao, Rui., & Shi, Haibo., Comparison of two fractal interpolation methods, Physica A: Statistical Mechanics and its Applications, Vol. 469, 2017, pp.563–571.
- [9] Warchalowski, Wiktor & Krawczyk, J. Malgorzata., Line graphs for fractals, Communications in Nonlinear Science and Numerical Simulation, Vol. 44, 2017, pp.506–512.
- [10] <http://idseducation.com/articles/apa-itu-animasi/> accessed in 7.34 pm 15 January 2017.
- [11] <https://www.cut-the-knot.org/Curriculum/Geometry/ifs.shtml> accessed in 10.42 am 17 January 2017.