IFS Fractals Generated by Affine Transformation with Trigonometric Coefficients and their Transformations

T.Gangopadhyay
XLRI
C.H.Area(E), Jamshedpur,
India

ABSTRACT
In IFS fractals generated by affine transformations with arbitrary coefficients often there is a lot of chaotic noise. In the present paper we study the effect of related trigonometric coefficients on affine transformations in terms of the IFS fractals generated by them. The use of related trigonometric functions as coefficients reduces the randomness of the scatter and generates meaningful shapes more frequently.

General Terms
Fractal, Algorithm, Turbo C++, Program.

Keywords
affine, IFS, swirl, horseshoe, trigonometric.

1. INTRODUCTION
In earlier papers we have studied the effect of the average of two special transformations on standard escape-time fractals (Gangopadhyay[5]) as well as their effect on popcorn fractals (Gangopadhyay[6]). In the present paper we study the effect of related trigonometric coefficients on affine transformations in terms of the IFS fractals generated by them. Usually in IFS generated fractals an arbitrary point is transformed repeatedly through multiple affine functions to produce fractal shapes. The coefficients used in these affine transformations are, however, arbitrary, and therefore unrelated. This leaves it to randomness to determine the nature of the shape of the fractal generated, often leading to chaotic configurations. Such noise is reduced when we use related trigonometric coefficients as specified in the algorithm. Trigonometric functions such as sine, cosine and tangent have been used earlier in popcorn fractals (Pickover[9]), as well as in certain attractors(Pickover[10], De Jong[3]) - but these take the form of main functions(sin(x), sin(y)) rather than coefficients. The use of trigonometric functions as coefficients does not change the linearity of the transformations as they do in the case of popcorn fractals and Pickover attractors. Although Highway Dragon curves (Bulaevsky[2], Gardner[7]) uses trigonometric functions as coefficients, they also use other divisors. Also there are only two sets of affine transformations in a dragon curve, as compared to our three. In a sense, the method used is a generalization of the process used for Dragon curves. The use of related trigonometric functions as coefficients reduces the randomness of the scatter and generates meaningful shapes more frequently. This is primarily the distinctive feature of this paper.

2. THE ALGORITHM
In iterated function systems Michael Barnsley [1] used affine transformations repeatedly on a starting point to produce fractal shapes. Draves [4] extended the scope of these transformations further. He introduced seven functional variations of the original point. Two of these, namely, swirl and horseshoe are described below:

Let (x, y) be the coordinates of the original point. Let $r = \sqrt{x^2 + y^2}$, $s = r^2$ and $t = \arctan(y/x)$. Then

a) swirl : $f(x, y) = (r\cos(2t), r\sin(2t))$.

b) horseshoe : $f(x, y) = (r\cos(t+r), r\sin(t+r))$.

We also introduce a new transformation which we term modified horseshoe. This is as follows:

c) $f(x, y) = ((r-t)\cos(t), (r-t)\sin(t))$.

In the present paper, we take trigonometric functions such as sine and cosine of angles as our coefficients in the affine transformations. Typically x is replaced by cos(angle)*x-sin(angle)*y+c and y by d*(sin(angle)*x+cos(angle)*y), where c = 0 or 1 and d = +1 or -1. Next, depending on the value of a random number we either apply one of Draves transformations, namely, the swirl, or, the modified horseshoe, or neither. In the code given below, both swirl and the modified horseshoe are applied with probability 0.3.

We colour each pixel by the number of times it is visited. The final output is contrasted againt a dark blue background generated by very weak Perlin noise[8].

In the next section we submit a programming code in Turbo C++ and generate some sample output.

3. THE CODE
The code uses a function tgatt which is declared first.

```
void tgatt(float a, float b,float cc,
float dd,float aa,float bb)

{
randomize();
float at1=random(360),
at2=random(360),at3=random(360);

float x=10,y=10,z=10,px;
int x1,y1;float u=160*3.14/180;

for(long i=1;i<1500000;i++)
{int j=random(1000);
int k=j/333;
px=x;
```

```
float de1=cos((at1)*3.14/180)/1.7414,
df1=sin((at1)*3.14/180)/1.7414;
float de2=cos((at2)*3.14/180)/1.7414,
df2=sin((at2)*3.14/180)/1.7414;
float de3=cos((at3)*3.14/180)/1.7414,
df3=sin((at3)*3.14/180)/1.7414;

if(k==0){x=(de1)*(px)-df1*(y);
         y=df1*(px)+de1*(y);}
if(k==1){x=(de2)*(px)-df2*(y)+1;
         y=-df2*(px)-de2*(y);}
if(k==2){x=(de3)*(px)-(df3)*(y)+1;
         y=(df3)*(px)+(de3)*(y);}


float r=sqrt(x*x+y*y);float t;
t=atan(y/x);
if(j<a){x=(r)*cos(2*t),
          y=(r)*(sin(2*t));}
else if(j<b){x=(r-t)*cos(t),
          y=(r-t)*(sin(t));}
x1=cc+aa*x-50,y1=dd+bb*y;
putpixel(x1,y1,getpixel(x1,y1)+1);


}
}

void main()
{

tgatt(300,600,200,150,180,230);
getch();
closegraph();
}
```

The output of the sample code is illustrated in Figure 1. Here at1=360, at2=330 and at3=45. The resulting output is illustrated in Figure 1.

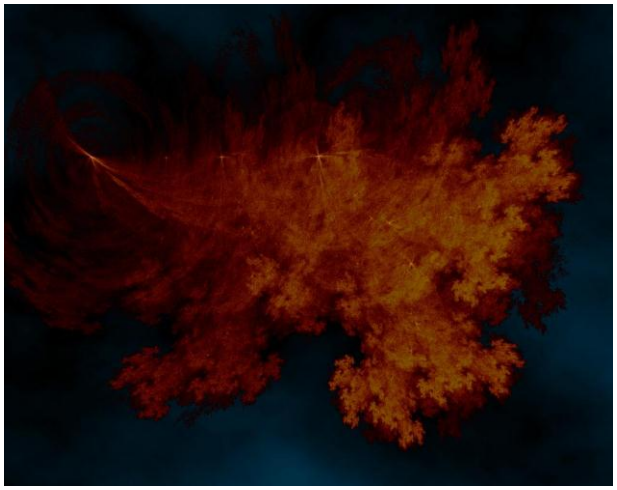

Fig 1 : Output of the sample code

By changing the angles at1, at2 and at3 in the function tgatt we can generate altogether different images. Let at1=320, at2=240 and at3=70..

The resulting output is illustrated in Figure 2.

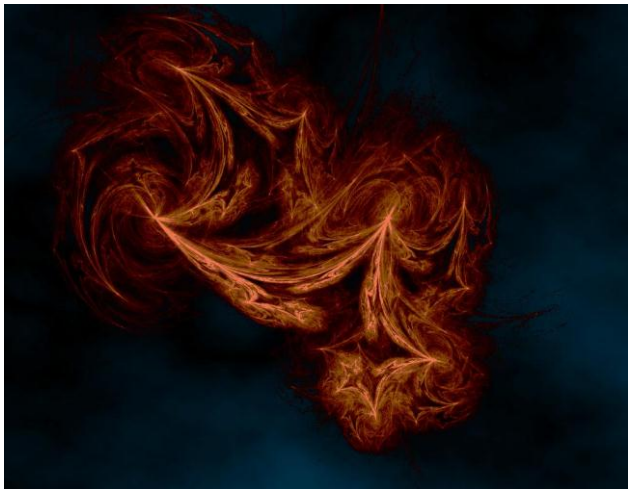

Fig. 2 : Output of sample code with new values of angles

4. VARIATIONS ON THE SAME THEME

In the sample code apart from changing the angles at1, at2, at3 we may also change the frequency of the two transformations by changing the parameters a and b in the function tgatt.

VARIATION 1.

Let at1=131, at2=306 and at3=39. Also, let a=200 and b=600.

The resulting output is illustrated in Figure 3.

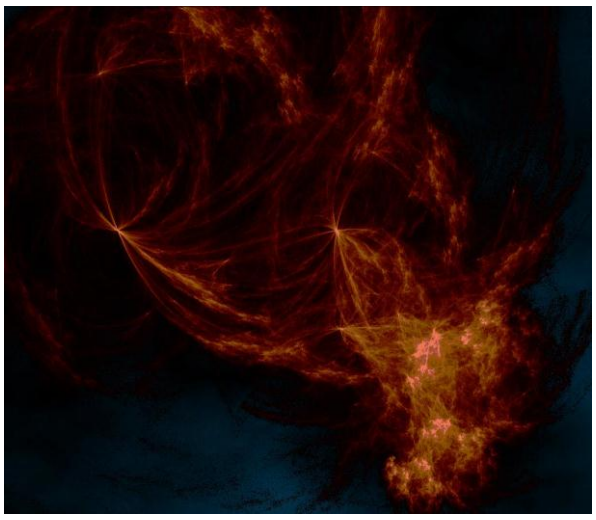

Fig. 3 : Output of Variation 1

VARIATION 2.

Let at1=216, at2=221 and at3=246. Also, let a=100 and b=200. The resulting output is illustrated in Figure 4.

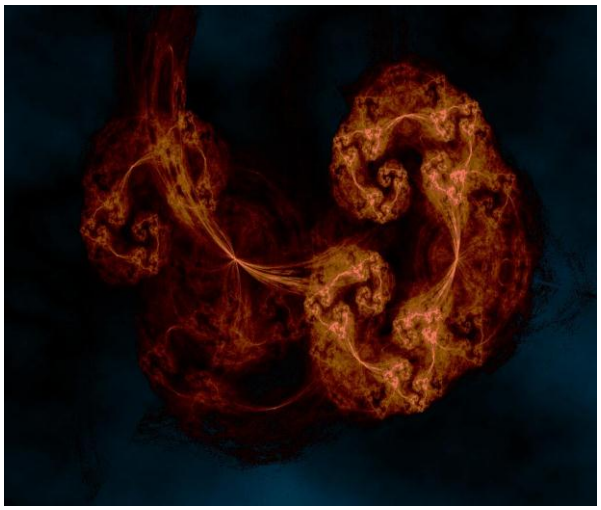

Fig. 4 : Output of Variation 2

VARIATION 3.

Let at1=6, at2=329 and at3=303. Also, let a=100 and b=400. The resulting output is illustrated in Figure 5.

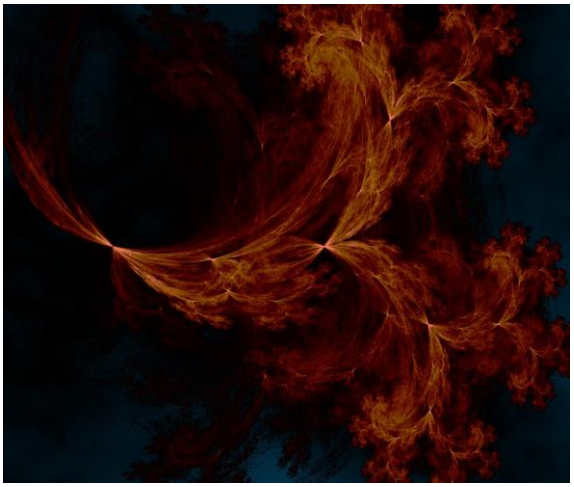

Fig. 5 : Output of Variation 3

VARIATION 4.

Let at1=251, at2=69 and at3=42. Also, let a=200 and b=400.

The resulting output is illustrated in Figure 6.

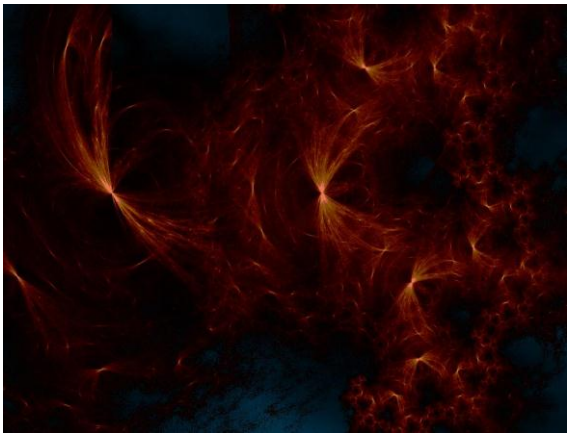

Fig. 6 : Output of Variation 4

4. Changing the equation

Sometimes dramatic effect can be achieved by changing one of the equations slightly. For instance, if in the programe, whenever k=1, x is replaced by cos(angle)*x+sin(angle)*y+1, the effect changes drastically. The output for at1= 5, at2 = 326 , at3=396 and a=100, b=400 is illustrated in figure 7.

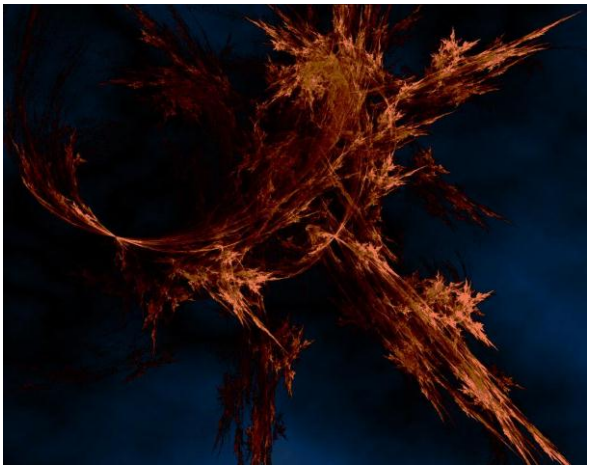

Fig. 7 : Output of modified code

Another example is displayed in figure 8. Here at1= 43, at2=279, at3=291, a=100, b=400.

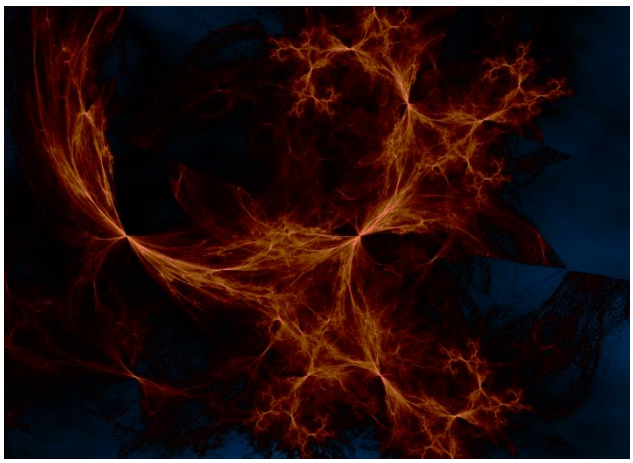

Fig. 8: Output of modified code

5. RELATED ANGLES

It is also interesting to study the effect of relating the three angles. Thus, if at2 = 120-at1 and at3=240-at1, then for at1=10, a=0, b= 70 we obtain the output depicted in figure 9. For at1=120, a=0, b=120 the output is displayed in figure 10.

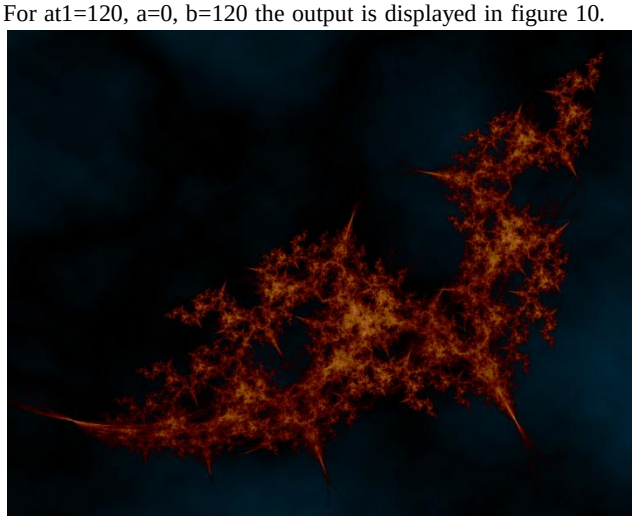

Fig. 9: Output of related angles -1

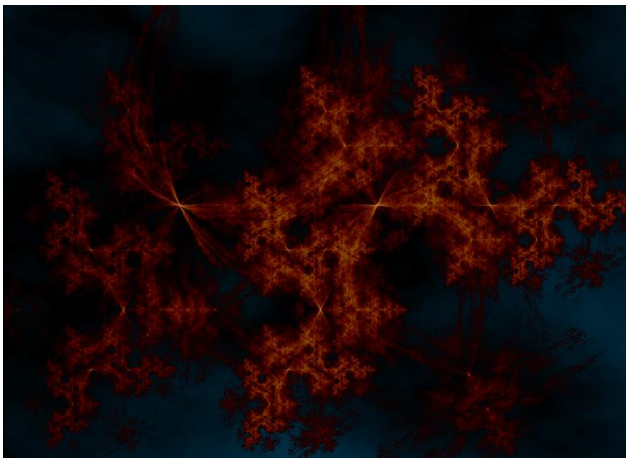

Fig. 10: Output of related angles -2

Finally for at1=10, at2=120+at1, at3=240+at1, a=0, b= 80, we display the output in figure 11.

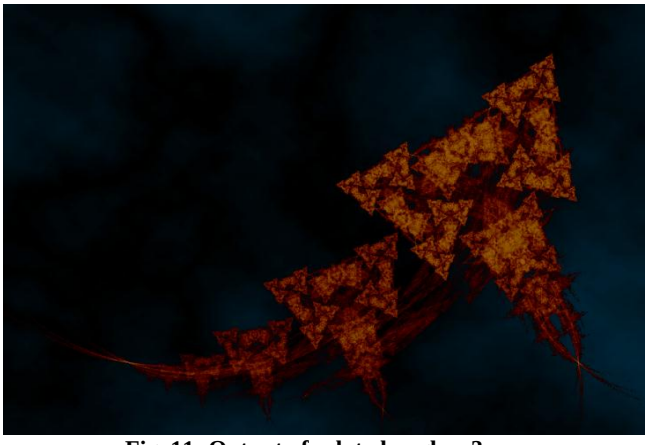

Fig. 11: Output of related angles -3

6. CONCLUSION

This paper presents the effect of applying a particular sequence of transformations on popcorn fractals. This sequence could be altered, or new transformations (such as hyperbolic) could be introduced to the sequence. Similarly transformations could also be used successfully on Chebychev fractals. These and other modifications would be explored in future work.

7. ACKNOWLEDGMENTS

The author wishes to acknowledge his debt to the referee(s) for their constructive suggestions and encouragement

8. REFERENCES

[1] Barnsley, M. 1983 Fractals Everywhere, Academic Press.

[2] Bulaevsky, J. "The Dragon Curve or Jurassic Park fractal."http://ejad.best.vwh.net/java/fractals/jurasic.shtml.

[3] De Jong, P. 1995 de Jong attractors, described in "The pattern book, Fractals, Art and Nature" by Clifford Pickover.

[4] Draves, S. 1992 The Fractal Flame Algorithm, flame3.com/flame-draves.pdf.

[5] Gangopadhyay, T. 2012 On generating skyscapes through escape-time fractals, International journal of Computer Applications 43(2012)17-19.

[6] Gangopadhyay, T. 2012 On Transforming Popcorn Fractals With Spherical And Other Functions, International journal of Computer Applications 50(2012)28-32.

[7] Gardner, M. 1978 Mathematical Magic Show: More Puzzles, Games, Diversions, Illusions and Other Mathematical Sleight-of-Mind from Scientific American. New York: Vintage, pp. 207-209 and 215-220.

[8] Perlin, K. 'Coherent noise function', original source code, http://mrl.nyu.edu/~perlin/doc/oscar.html#noise.

[9] Pickover C. quoted in Fractint formula documentation, www.nahee.com/spanky/www/fractint/popcorn_type.html.

[10] Pickover, C. Clifford attractors, http://paulbourke.net/fractals/clifford/index.html

.