On Generating Julia Set Fractal Shaped Images through Mobius Transformations

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

ABSTRACT
Iterating two pairs of Mobius transformations as attractors generates fractals that are known as Mobius fractals. In the present paper one studies pairs of specific Mobius transformations that generate shapes very similar to Julia set fractals.

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

Keywords
Mobius, IFS, Julia,.

1. INTRODUCTION
A Mobius transformation of a complex number z is of the form (az+b)/(cz+d), where ad - bc ≠ 0 (Krantz [4],,Needham [5]). In earlier papers the present author has studied the effect of multiple rotations on Mobius transformations in terms of the IFS fractals generated by them (Gangopadhyay[2]) In the present paper one studies pairs of Mobius transformations that generate shapes very similar to Julia set fractals. Julia set fractals are normally generated by first initializing a complex number z = x + yi where x and y are pixel coordinates in the range of about -2 to 2. Then, z is repeatedly updated using: $z = z^2 + c$ where c is a specific complex number. This generates a specific Julia set. Fractal(Julia[3], Sims[6]). In this paper an alternate approach that generates Julia set fractal like images is described. .The approach uses multiple rotations on Mobius transformations. This is the main distinctive feature of this paper.

2. THE ALGORITHM
The algorithm takes two generic Mobius transformations, the first with coefficients a1, b1,c1 and d1 and the second with a2, b2, c2 and d2. Each is then rotated multiple times through angles that are multiples of a given angle. This given angle for a transformation has the value 360/m in degrees, where m is the number of rotations for that transformation. The generation of Julia set fractal like images is achieved by choosing certain specific values for the Mobius transformation coefficients.

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

3. THE CODE
The code uses a function tgmob which is declared first. The function has eleven parameters – the two sets of complex coefficients a1, b1, c1, d1 and a2, b2, c2, d2, an integer n which gives the total number of rotations and an integer m which gives the number of rotations per transformation. Both the tgmob function and the code are given below:

```
void tgmob(cmplx a1,cmplx b1,cmplx c1,
cmplx d1,cmplx a2,cmplx b2,cmplx c2,
cmplx d2, int n,int m)
{int j,k; double x=1,y=1,px; int scale=250;
cmplx z=cmplx(.1,.5); float ang;
for(long i=0;i< 3700000;i++)
{  px=x;
j=random(999)%1000;
k=j%n;
if(k<m){ang=(k*360/m)*3.14/180.;
z=(a1*z+b1)/(c1*z+d1);
x=real(z),y=imag(z);px=x;
x=x*cos(ang)-y*sin(ang);
y=px*sin(ang)+y*cos(ang);
z=cmplx(x,y); }
else{ang=((k-m)*360/(n-m))*3.14/180.;
z=(a2*z+b2)/(c2*z+d2);
x=real(z),y=imag(z);px=x;
x=x*cos(ang)-y*sin(ang);
y=px*sin(ang)+y*cos(ang);
z=cmplx(x,y); }
x=real(z),y=imag(z);
int x1=scale*x+512,y1=scale*y+400;
if(x1>0&&x1<1024&&y1>0&&y1<800)
putpixel(x1,y1,getpixel(x1,y1)+1);
}}
void main()
{
cmplx ii=cmplx(0,1);
tgmob( .75,0, 3.,.5*ii+sqrt(2.)/2.,(.5-ii)*.7,.5*.7,.5,.5+ii,4,2);
getch();
closegraph();
}
```

The output of the sample code is illustrated in Figure 1.

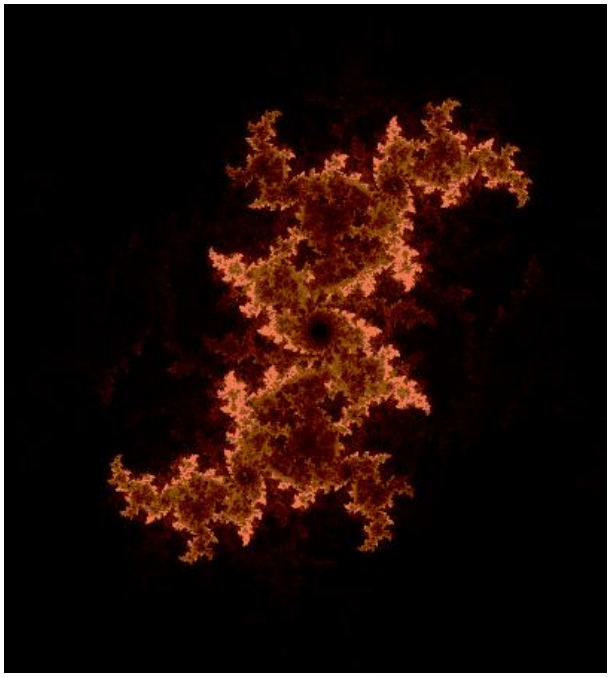

Fig 1 : Output of the sample code

In Figure 2, we produce a standard Julia set fractal for c=.289-.5i and note its resemblance to the output in Figure 1.

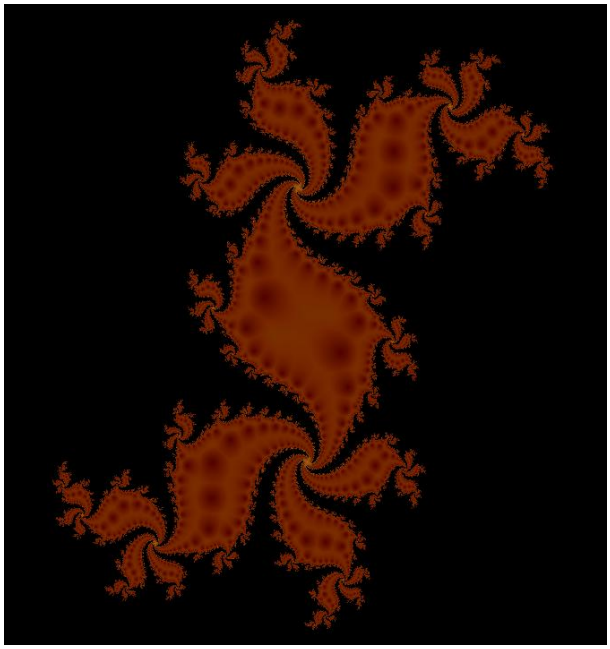

Fig 2 : Julia set fractal with c=.289-.5i

In Figures 3, 5 7 and 9 we produce further examples of Mobius fractals which resemble standard Julia set fractals. Here ii stand for the imaginary number √t(-1).

To obtain the image in Figure 3, we suitably change the parameters of the function tgmob in the sample code. The new parameters are given by the expression: tgmob(1.175,0,-6,.5*ii+ √(3.)/2.,.35-.7*ii,.35,.5,(.5+ii),4,2);

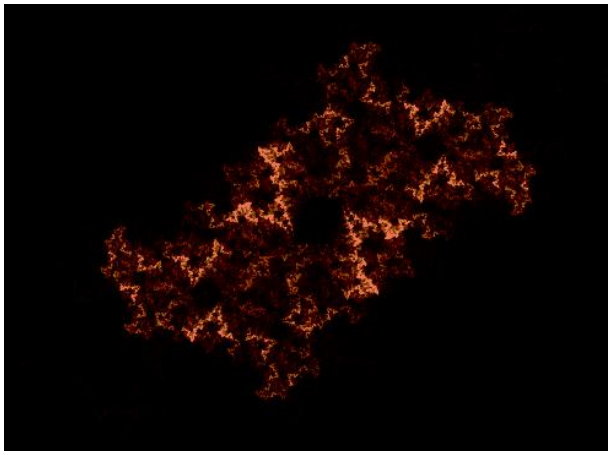

Fig. 3 : Output of sample code for tgmob(1.175,0, -6,.5*ii+ √ (3.)/2.,.35-.7*ii,.35,.5,(.5+ii),4,2);

In Figure 4, we produce a standard Julia s et fractal for c=-.257-.642i and note its resemblance to the output in Figure 3.

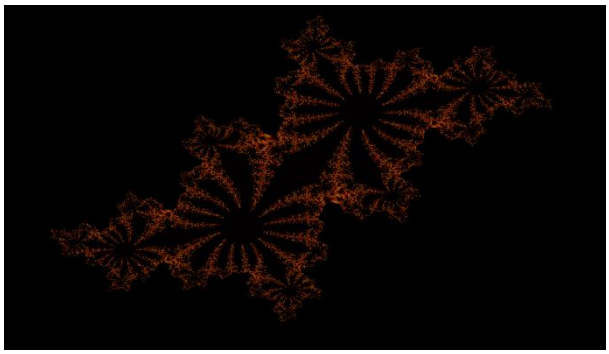

Fig 4 : Julia set fractal with c=-.257-.642i

To obtain the image in Figure 5, we suitably change the parameters of the function tgmob in the sample code. The new parameters are given by the expression: tgmob(ii,0,2,.5*(ii)+.4, √ (1.5), √ (.35), √ (.35), √ (1.5),4,2);

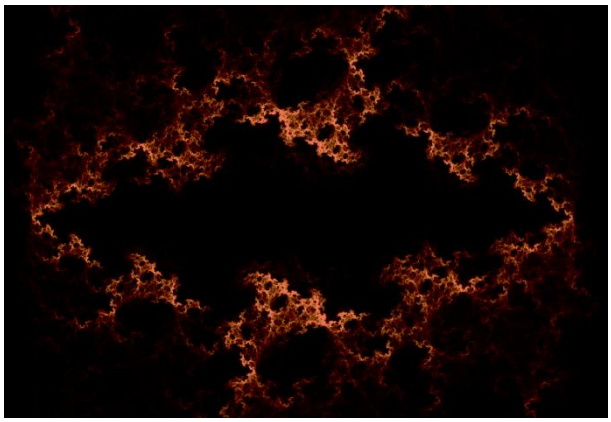

Fig. 5 : Output of sample code for tgmob(ii,0,2,.5*(ii)+.4,(√ (1.5)), √ (.35), √(.35), √(1.5),4,2)

In Figure 6, we produce a standard Julia set fractal for c=-.7043-.121i ;and note its resemblance to the output in Figure 5.

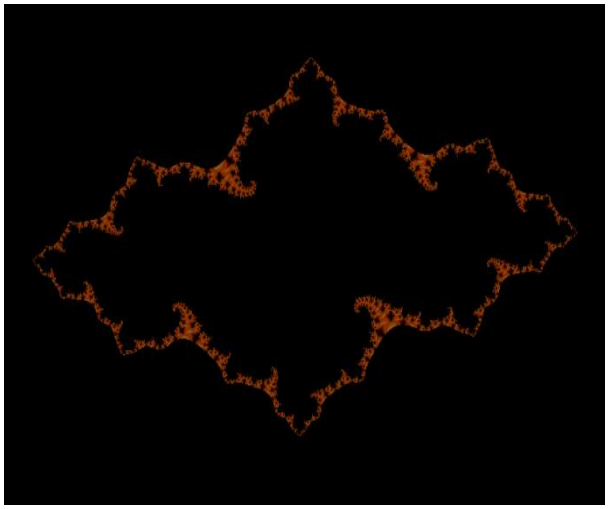

Fig 6 : Julia set fractal with c=-.7043-.121i

To obtain the image in Figure 7, we suitably change the parameters of the function tgmob in the sample code. The new parameters are given by the expression:tgmob(.75,0,2.,1.75*ii+ √ (2.)/1.5,(.5-ii)*.7,.5*.7,.5,.5+ii,4,2);

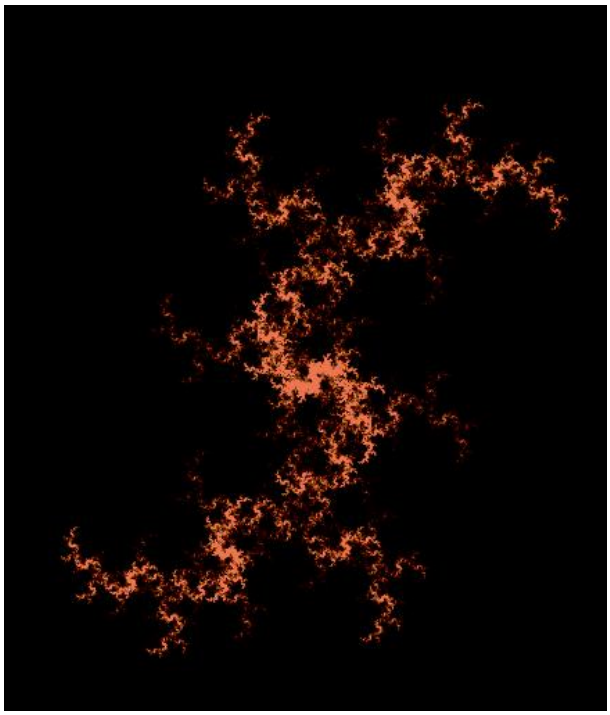

Fig. 7 : Output of sample code for tgmob(.75,0,2.,1.75*ii+ √(2.)/1.5,(.5-ii)*.7,.5*.7,.5,.5+ii,,4,2)

In Figure 8, we produce a standard Julia s et fractal for c=.41-.3157i ;and note its resemblance to the output in Figure 7.

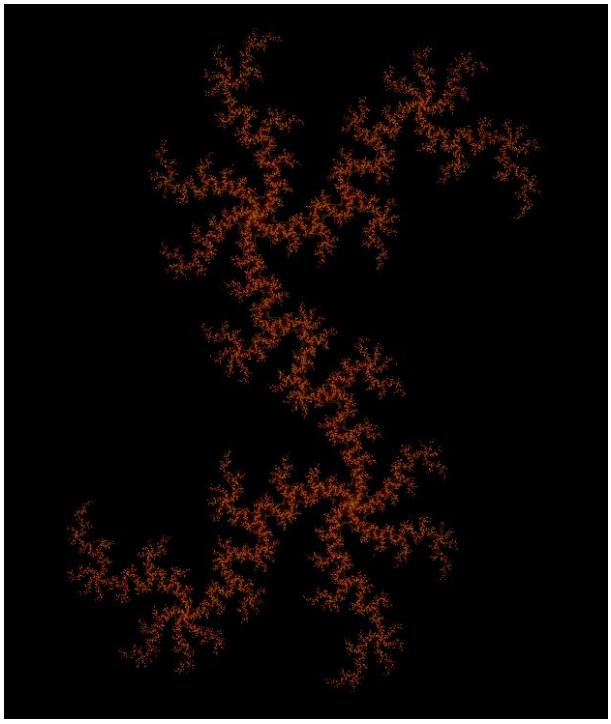

Fig 8 : Julia set fractal with c=.41-.3157i

To obtain the image in Figure 9, we suitably change the parameters of the function tgmob in the sample code. The new parameters are given by the expression:tgmob(1.,0.,3,ii-.2, √ (1.5), √(.5), √ (.5), √ (1.5),4,2);

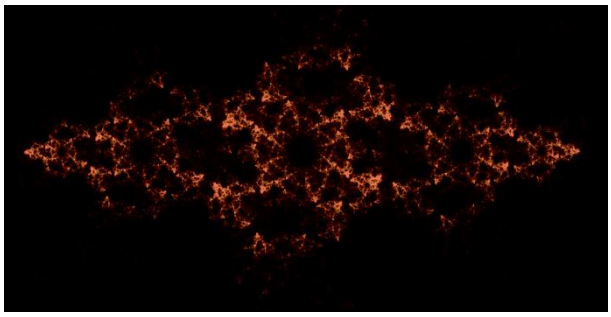

Fig. 9 : Output of sample code for tgmob(1.,0.,3,ii-.2, √ (1.5), √(.5), √ (.5), √(1.5),4,2)

In Figure 10, we produce a standard Julia s et fractal for c=.-.737-.1376i ;and note its resemblance to the output in Figure 9.

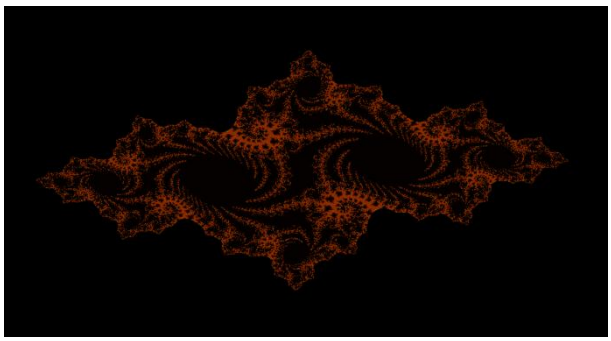

Fig 10 : Julia set fractal with c=.-.737-.1376i

In Figures 11 – 15, we produce further examples of Jluia set fractal like images.

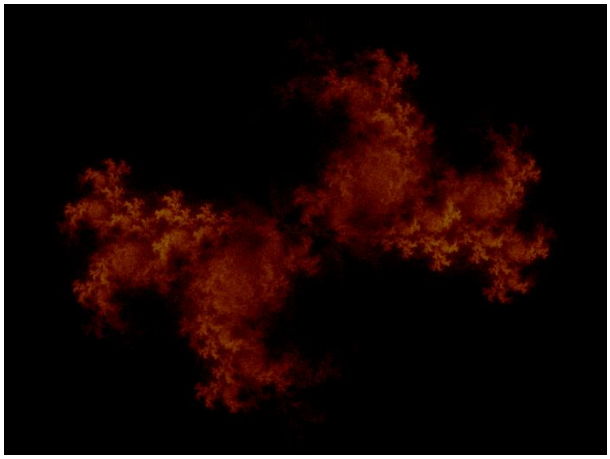

Fig.11 : Output of sample code fortgmob(1.,0.0,.6*ii+ √ (3.)/2.,1,(.5-ii)*.7,.5*.7,.5*.7,(.5+ii),4,2)

Fig.12 : Output of sample code fortgmob(.75,0,2.,.75*ii+ √ (2.)/1.5,(.5-ii)*.7,.5*.7,.5,.5+ii,4,2)

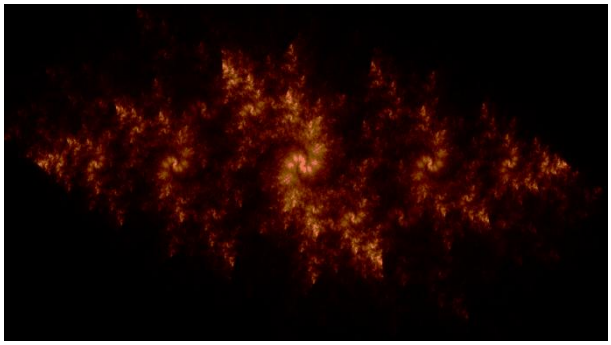

Fig.13 : Output of sample code fortgmob(ii+.5,0.0,1,.85*(ii)+1.5,(√(1.5)),√(.35),√(.35),√(1.5),4,2).

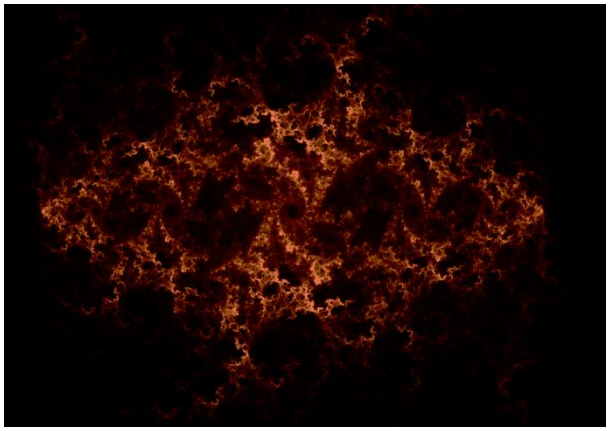

Fig.14 : Output of sample code fortgmob(ii,0,2.5,(ii)+.6,(√(1.5)), √ (.35), √(.35),√(1.5),4,2)

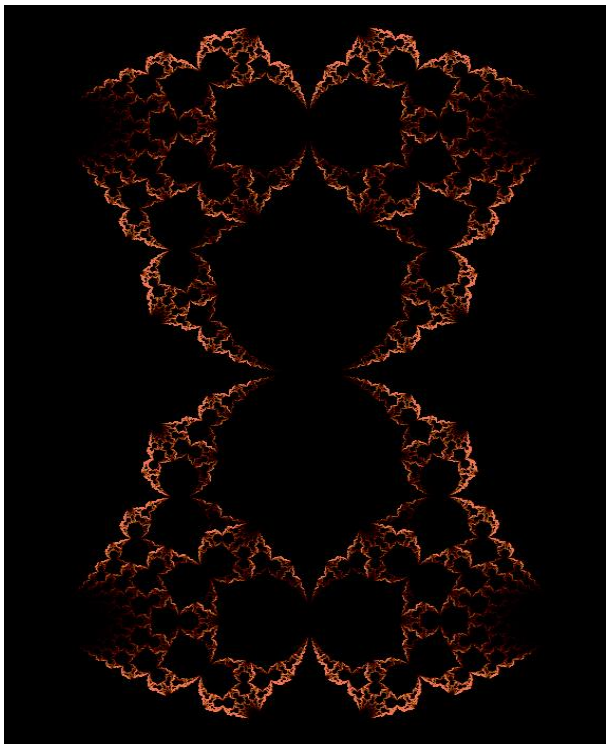

Fig.15 : Output of sample code fortgmob(-ii,0,1.+ii,ii,1.-ii,1.,1.,1.+ii,4,2);

4. CONCLUSION

This paper presents pairs of Mobius transformations which generate shapes very similar to Julia set fractals In subsequent studies one will further explore the generation of images that resemble cubic, quartic and sin julia set fractals. One will also explore the generation of Newton fractal (Bourke[1]) like images through Mobius transformations.

These are the aspects that would be explored in future work.

5. ACKNOWLEDGMENTS

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

6. REFERENCES

[1] Bourke, P.: An Introduction to fractals ,http://paulbourke.net/fractals/fracintro/

[2] Gangopadhyay, T. The effect of multiple rotations on Mobius transformations in generating IFS Fractalss, International journal of Computer Applications 142(6):18-22, May 2016

[3] Julia, G. Mémoire sur l'itération des fonctions rationnelles, Journal de Mathématiques Pures et Appliquées, 1918

[4] Krantz, S. G. "Möbius Transformations." §6.2.2 in Handbook of Complex Variables. Boston, MA: Birkhäuser, p. 81, 1999..

[5] Needham, T. "Möbius Transformations and Inversion." Ch. 3 in Visual Complex Analysis. New York: Clarendon Press, pp. 122-188, 2000.html.

[6] Sims , K Understanding Julia and Mandelbrot sets,. http://www.karlsims.com/julia.html