Study on the Generation of Fractal Image and Animation Based on MATLAB and Flash Platform

Xiaoming Xie
School of Electronics and Information Technology, Sun Yat-sen University, 132 Waihuan East Road, Guangzhou 510006, China
xiexm7@mail2.sysu.edu.cn

Abstract. In practice, the content of fractal graph generated by one kind of algorithm alone is relatively toneless. Combining different algorithms can not only break the limitation of single fractal object generation but also enrich the contents of scenes. In this paper, according to the needs of scene construction, more targeted algorithms are appropriately chosen. By combing two fractal algorithms of recursion and Iterator Function System (IFS for short), two static scenarios “Old Tree, Withered Grass and Snowy Night” and “Reed Land and Snowy Night” are built on MATLAB platform, which further enriches contents of the scene. Simultaneously, a trial has been made to experiment the generation of dynamic fractal graphs. Based on the fractal algorithm with parameters, a recursive windmill animation and a lunar eclipse animation based on recursion and IFS algorithm are generated on MATLAB platform, which enhances the creativity of the scene building of fractal animation. With respect to the selection of implementation platform, this paper compares the effects of fractal graph generation between MATLAB and Flash. The results show that, thanks to its powerful animation creation ability, Flash is undoubtedly more valuable than MATLAB in the implementation of relatively simple fractal animation effect in the light of speed and convenience. Nevertheless, in order to achieve richer and more realistic animation effects, the more powerful MATLAB platform is preferred. In general, it is necessary to choose the right platform according to actual requirements of creation so as to improve the development efficiency.

1. Introduction
As Mandelbrot [1] notes, fractal is the geometry of nature. Fractal is a new-type subject that has been developed in recent 30 years, and this word is translated from the English vocabulary “fractal”, which means irregular, broken and numeral. In 1975 and 1982, a French mathematician of American nationality Benoit Mandelbrot, the creator of Fractal Theory, published two historical works: Los objetos fractales: forma, azar y dimensión and the Fractal Geometry of Nature. Since then, the fractal theory has taken shape.

The fractal theory is a strong theoretical tool for studying irregular objects in the natural world and engineering application. Overall, the fractal object has an uneven and irregular shape and such irregularity also coexists with regularity because of its self-similarity. With the development of this theory, the geometric characteristics of fractal object has no longer been limited to its self-similarity. Objects with self-affine transformation relationship between the whole and the part are also called as fractal. In general, the fractal object is featured by irregularity, self-similarity, self-affinity, self-affinity and smart structure [2].
The fractal image can simulate vividly all kinds of scenery in the nature. It furnishes a new perspective for human beings to know the nature and the world. With the rapid development of computer technology, the computing power of the computer is increasingly strong. The computer-aided fractal algorithm helps to generate complex textures and shapes, as well as producing fractal objects with physical forms and visual effects. The fractal geometry deals with the dot, line and facet of the figure from an overall perspective, so it is superior to traditional Euclidean geometry in terms of the description of nature.

At present, the fractal theory is widely applied to the fields of physics, chemistry, biology, medicine, geology and computer science [3]. In computer graphics, the fractal theory has broadened the channels and provided a new direction for computer graphics generation [4]. In aesthetics, the fractal geometry is applied to morphological creation of furniture [5], landscape design [6], architectural design [7] and textile industry [8]. In the age of digital image processing, the emergence of fractal theory has improved the compression ratio of image encoding, which is particularly important for image transmission and processing [9]. Moreover, the fractal theory also has some important applications in computer animation generative technology. It can help the computer to simulate vivid natural scenes and improve the efficiency of animation production. Therefore, it is widely applied in the field of film-making and game design.

This paper focuses on the application of fractal geometry in computer graphics. By combining recursive algorithm and IFS algorithm, corresponding fractal images are generated on MATLAB platform to build scenes and enrich the image contents. Besides the creation of static fractal images, this paper also explores the generating method of fractal animation and implements on MATLAB platform. Furthermore, in the selection of implementation platform, the effects of generating fractal images on platforms of MATLAB and Flash are compared.

2. Creation of snowy night image - combination of recursive and IFS algorithms

Different algorithms could simulate different natural objects in reality. The fractal image generated by one algorithm is very single in its content. If different algorithms could be used in combination in scene construction, a rather rich scene would be generated. Next, this study will apply different algorithms to generate corresponding fractal objects based on our needs and combine them in the same image to build natural scenes.

2.1. Creation of the scene “Old Tree, Withered Grass and Snowy Night”

This scene graph contains a row of withered grasses and a withered tree as its main scenery, in which the withered grasses are generated by recursive algorithm that is relatively simple among so many fractal algorithms, in consideration of the efficiency of program execution. Although IFS algorithm can be used to simulate high-fidelity natural scene, its implementation complexity is far higher than that of the recursive algorithm, and the IFS code is required to be obtained in the first place. If each withered grass is generated by calling IFS algorithm, time and memory resources will be consumed highly. In view of these problems of IFS algorithm, the part of withered grass is finished by recursive algorithm. As the withered grass is small in shape, it can just be generated at a low frequency of recursion. The final results of creation shows that the effects can be satisfied (Figure 1). As only one withered tree is required to be generated, IFS algorithm is used to achieve a vivid effect. The generation process of the scene graph and part of the codes (MATLAB) used are listed as follows. (1) Determine the hierarchy relationship of the scenery (also known as the layer relationship) to further sequence creation. In figure 1, the crescent is on the bottom layer, the fractal tree is on the second layer, the withered tree is on the third layer and the snow is on the top layer. (2) The image takes shape from the bottom layer. Firstly one moon is generated - a white circle is drawn. Then cover the white circle partly with another circle of the same background color to present a crescent effect. (3) Generation of a clump of grasses. The codes used for creating a clump of grasses are listed as follows.

```matlab
function f=bulrush(ax,ay,bx,by,alfa)
c=0.05; PI=3.1415926; H=sqrt((bx-ax)*(bx-ax)+(by-ay)*(by-ay)); h=H/2; alpha=(alfa/180)*PI;
```
if bx<ax
    beta=atan((by-ay)/(bx-ax))+PI;
elseif bx==ax
    beta=PI/2;
else
    beta=atan((by-ay)/(bx-ax));
end
x=[ax,bx]; y=[ay,by]; plot(x,y,'Color',[rand() rand() rand()],'LineWidth',2); hold on;
if H>c
    gx=(bx+ax)/2; gy=(by+ay)/2;
    cx=gx+h*cos(beta+alpha); cy=gy+h*sin(beta+alpha);
    ex=bx+h*cos(beta+alpha); ey=by+h*sin(beta+alpha);
    x=[gx,cx]; y=[gy,cy]; plot(x,y,'Color',[rand() rand() rand()],'LineWidth',2); hold on;
    x=[bx,ex]; y=[by,ey]; plot(x,y,'Color',[rand() rand() rand()],'LineWidth',2); hold on;
    bulrush(gx,gy,cx,cy,alfa); bulrush(bx,by,ex,ey,alfa);
end
axis([-2 2 -2 2]); axis off;
for N=1:100
    ax=rand(); ay=0.035; bx=rand()*0.02-0.01+ax; by=rand()*0.07+0.07;
    bulrush(ax,ay,bx,by,15); hold on;
end
A total of 100 withered grasses is generated. “ax” and “ay” denote respectively as the horizontal and vertical coordinate of the withered grass position. The value of “ax” is produced by a random number, so the positions of the 100 withered grasses are all random. “bx” and “by” are related to the height of the withered grass. The height of each withered grass is also random and falls between 0.7 and 1.4.

(4) IFS algorithm is applied to generate fractal tree.

function f=tree_ifs_2(ax,ay)
figure('color',[1,1,1]);
x=[ax ay];
a1=[0.195 0.344;-0.488 0.443]; b1=[0.4431 0.2452];
a2=[0.462 -0.252;0.414 0.361]; b2=[0.2511 0.5692];
a3=[-0.058 0.453;-0.07 -0.111]; b3=[0.5976 0.0969];
a4=[-0.035 -0.469;0.07 -0.022]; b4=[0.4884 0.5069];
a5=[-0.637 0;0 0.501]; b5=[0.8562 0.2513]; r=rand(1,100000);
for N=1:100000
    if (r(1,N)>=0)&(r(1,N)<=0.25)
        x=x*a1+b1; plot(x(1),x(2),'Color',[0.80 0.75 0.44]); hold on;
    elseif (r(1,N)>0.25)&(r(1,N)<0.5)
        x=x*a2+b2; plot(x(1),x(2),'Color',[0.80 0.75 0.44]); hold on;
    elseif (r(1,N)>0.5)&(r(1,N)<0.75)
        x=x*a3+b3; plot(x(1),x(2),'Color',[0.80 0.75 0.44]); hold on;
    elseif (r(1,N)>0.75)&(r(1,N)<0.95)
        x=x*a4+b4; plot(x(1),x(2),'Color',[0.80 0.75 0.44]); hold on;
    else
        x=x*a5+b5; plot(x(1),x(2),'Color',[0.80 0.75 0.44]); hold on;
    end
end
axis off
(5) Adorn the fractal image with snowflakes.
The snowflakes are produced by drawing “*” at the coordinate points which are generated by random numbers. The final effect is shown in figure 1.

![Figure 1. The image of “Old Tree, Withered Grass and Snowy Night” generated by recursive and IFS algorithm.](image1)

![Figure 2. The image of “Reed Land and Snowy Night” generated by recursion and IFS algorithm.](image2)

2.2. Creation of the scene “Reed Land and Snowy Night”
An attempt of the 3D visual richness is conducted in the above situations and the reeds are further turned to a reed land. Such 3D visual presentation from close-by examples to those far off just requires, from the perspective of visual principle, a deceasing tendency of the reeds row by row, thereby producing effects of a piece of reed land. The codes for generating this image are just required to be modified from the code of reed part in figure 1 and the rest would be maintained. The final effect is shown in figure 2. The generating codes for reed land are listed as follows.

```matlab
t=[500 400 300 250 200 120 120 120 120];
a_x=[1 0.96 0.92 0.88 0.84 0.76 0.76 0.76 0.76]; b_y_1=[0.06 0.03 0.03 0.02 0.02 -0.06 -0.06 -0.06 -0.06]; b_y_2=[0.61 0.43 0.26 0.17 0.07 -0.06 -0.02 -0.02 -0.02];
n=0;
for N=1:9
    n=n+1;
    for N1=1:t(n)
        ax=((rand()*2)-1)*a_x(n); ay=-0.8; bx=rand()*0.02-0.01+ax;
        by=-rand()*b_y_1(n)-b_y_2(n);
        bulrush(ax,ay,bx,by,15); hold on;
    end
end
```

The vectors “t”, “a_x”, “b_y_1” and “b_y_2” are parameters used to control the number and height ranges of reeds in each row, in which “t” denotes the number of reeds in each row, “a_x” is used to control the range of reeds, “b_y_1” and “b_y_2” is used to control the height range of the reeds.

3. Generation of fractal animation based on MATLAB platform

3.1. Basic principle of fractal animation: fractal algorithm with parameters
Most fractal algorithms are required to contain parameters to control the appearance of the fractal objects. In other words, if one wants to change the appearance of the object generated by fractal algorithm, only the parameters of fractal algorithm are required to be modified. Taking IFS algorithm as an example, the appearance of the fractal object generated by IFS algorithm rests on IFS code. If the IFS codes are changed, so does the appearance of the fractal object. Such changes may be trivial or great, which is dependent on the role of the IFS codes play in the whole code block. Based on this
principle, control of the continuous change of one key parameter in the algorithm can produce a fractal image with a series of continuous changes. If we consider these continuously changing fractal images as the animation frame and play them continuously, an effect of fractal animation could be produced. The following study is based on MATLAB platform. It generates several pieces of fractal animation respectively by recursive algorithm and IFS algorithm with parameters.

3.2. Basic ideas of generating animation on MATLAB platform
Generating continuously changing fractal image by continuously changing parameters also mean that a new piece of fractal image will be generated after a change of parameter each time. As the implementation of fractal algorithm may consume some time, the time interval between two adjacent images may be long. If the animation is generated by drawing firstly and displaying next, it will appear so discontinuous and affect the visual effect. Therefore, this study uses a method of generation frame by frame firstly and centralized playback next. After each piece of image is generated, the command “getframe” in MATLAB will be used to get each fractal image generated, namely, each frame of the animation. Then the frames will be saved via the command of “movie2avi” as an avi file. Finally, the effects of fractal animation will be seen continuously by playing this avi file. In addition, the play speed and loop playback may be set through relevant commands. Next this study will analyze in detail the design process and key codes of the fractal animation generated on MATLAB platform.

3.3. Windmill animation based on recursion
The animation effects of the windmill animation in this design are presented by controlling the continuous changes of the reed shape via the continuously changing parameters. In order to control the reed shape more intuitively and conveniently, the reed generated function is required to be modified. Specific details are shown as follows.

```matlab
function f=bulrush_2(ax,ay,b,alfa_1,alfa_2)
    PI=3.1415926; alfa_1=alfa_1*(PI/180); bx=b*cos(alfa_1); by=b*sin(alfa_1);
    bulrush(ax,ay,bx,by,alfa_2);
end

Unlike the generated function of the above-mentioned reed bulrush (ax,ay,bx,by,alfa_1), bulrush_2 controls the shape of reed directly by passing its height (b) and angle (alfa_1) and both contribute to calculations of “bx” and “by” value, then the bulrush function will be called again to generate the reed. Such modification enables more intuitive and convenient control of the reed shape and simplifies the animation process of windmill animation. The generation process of the windmill animation is listed as follows.

(1) Generation of the windmill.

```matlab
bulrush_2(0,0,1,0,-18); M(1)=getframe; r=0; dr=5;
for N=1:72
    clf; r=r+dr; bulrush_2(0,0,1,r,-18);
    if(r>=50)
        bulrush_2(0,0,1,45,-18);
    end
end
```

Similar to if(r>=50), the other 6 reeds will be generated respectively at 90°, 135°, 180°, 225°, 270° and 315°, and form a complete windmill when the precondition of “if” statement changes. So specific codes are not given here. The processes and effects of windmill generation are shown in figure 3.

![Figure 3. Windmill generation processes and effects.](image)

(2) Rotation of the windmill.
Two parameters should be defined in this step. One controls the pendulum angle of small subbranches of the reed and the amplitude of swing ranges from -18 to 18. The other denotes the swing pace of the small subbranch and the subbranch swings 1° in each frame. The effects of windmill rotation are shown in figure 4.

(3) The windmill changes in size while rotating with smear.

Four parameters should be defined in this step. The first one controls the pendulum angle of small subbranches of the reeds and the amplitude of swing ranges from -18 to 18. The second one denotes the swing pace of the small subbranch and the subbranch swings 1° in each frame. The third one controls the length of the reeds, ranging from 0.028 to 1. The last one denotes the pace of length changing. The animation effects are shown in figure 5.

3.4. Lunar eclipse based on recursive and IFS algorithms

In the image of “Old Tree, Withered Grass and Snowy Night” (Figure 1), the crescent is formed by partial overlap of two circles and its shape changes with the movement of the circle-cover. If the process of continuous crescent changes is gotten in the form of frame, continuous playback helps to get the whole process of lunar eclipse. In addition, the continuous changes of parameters allow the reeds to render an effect of swinging with the wind. The generation process in shown as follows.

(1) Generate the 1st frame of animation.

In this step, we can define two vectors “ax” and “1”, both are used to save the length information of each reed generated in the first frame. Only in this way can the reed of same length be generated at the same location in each of the following frames and the discontinuity between frames could be avoided.

(2) The process of lunar eclipse and swinging reeds.

There are four parameters defined in this step. The parameter “s” denotes the horizontal coordinate of the circle-cover, and “ds” denotes the pace of its change in each frame. The parameter “angel” is used to control the angle of reed, “da” is the pace of its change in each frame. In this animation, firstly the reeds swing to an angle of 105° and then swing back and forth between 103°and 105°. This effect is controlled by mark bit “flag”. Once these parameters are defined, we can simply start to generate frame by frame through using a “for” loop.

(3) Generation of .avi files.

The final effects are shown in figure 6.
4. The combination of Flash and fractal animation

As a powerful animation creation software, Flash is a rich animation creation tool that provides many advantages for animation creators. Although Flash is widely used in animation, the current research on the combination of Flash and fractal animation is still relatively rare. After Flash develops the function of supporting the copy of a movie clip, it has become very convenient to develop fractal animations on the Flash platform. The similarity between the part and the whole of fractals enables us to use the same small parts as the fractal overall shape to generate the final whole by the iteration of one layer after another. This design implements LS algorithm by writing script codes and simulates a plant’s growth process from sparse to dense by creating components and adding simple shape to the components. The following is the script codes and analysis of the two LS algorithms with single-rule and multi-rule.

4.1. Implementation of single-rule LS algorithm

```javascript
function ls(w,ax,ay,t)
{
    var t_0=0; _root.stack=[]; var stack_e=0; var x=ax; var y=ay; var w_temp=new String("");
    var alpha=-90; var alfa=25; var i=1;
    if(t>0)
    {
        for(i=0; i<w.length; i++)
        {
            ……//Generating the path under the rule: F -> F[-F]F[+F]F
        }
        this. ls(w_temp,x,y,t-1);
    }
    if(t==0)
    {
        for(i=0;i<w.length;i++)
        {
            switch(w.charAt(i))
            {
                case('F'): ……//step forward and draw a line
                case('['): ……//push the current direction and position into stack
                case(']'): ……//pop from stack
                case('-'): alpha=alpha-alfa; break;
                case('+'): alpha=alpha+alfa; break;
            }
        }
    }
}
```

In the “function ls (w, ax, ay, t)”, an array is used to implement the stack frame of the LS algorithm and creates objects of “String” for the strings so that character(s) is convenient for operation such as traversal, replace, connections etc. The animation effects are achieved mainly through the shape tweening of the component. By adding a component instance to the main scene with “attachMovie()”, and adding the key frame to the new layer in the main scene, the function is added to the frame codes. The final effects are shown in figure 7.
4.2. Implementation of multi-rule LS algorithm

“Multi-rule” in the multi-rule LS algorithm refers to the rule of multiple replacement. In the single-rule LS algorithm, replacement is required only when character ‘F’ appears in a string, while multi-rule requires to replace multi characters. Hence, the script codes only need to be partially modified. Firstly, add judgement statement in the string to be replaced according to the newly added rule. For example, in this multi-rule design, there are two replacement rules:

\[ L \rightarrow [LL+[L+[L-[R-[R++R]]]R]; \quad R \rightarrow [R-[R++R]] \]

Simply add the following statement to the first “for” loop in the single-rule function.

```java
if(w.charAt(i)=='L'||w.charAt(i)=='R')
{
    if(w.charAt(i)=='L')
    {
        w_temp=w_temp.concat("LL+[L+[L-[R-[R++R]]]R]");
    }
    else
    {
        w_temp=w_temp.concat("[R-[R++R]]");
    }
}
```

The others are similar to the previous single-rule. The final effect is shown in figure 8.

![Figure 8](image)

Figure 8. A plant’s growth process by applying Multi-rule LS algorithm.

What can be concluded from the above figure 7 and figure 8 is that although it’s easier to implement LS algorithm on Flash, deeper sense of recursion can’t be realized because of the limitation of Flash computing power, thus the effect of plant simulation is not as expected. The main idea of LS algorithm is moving forward and drawing lines, which limits the animation effect of Flash and makes the effect monotonous and not obvious. From this perspective, the MATLAB platform’s computing power is obviously much stronger than Flash because it is capable of deep recursion, and the simulation of plants on MATLAB is better than Flash in terms of fidelity.

5. Conclusion and discussion

In practice, the content of fractal graph generated by one kind of algorithm alone is relatively toneless. Combining different algorithms can not only break the limitation of single fractal object generation but also enrich the contents of scenes. In this paper, according to the needs of scene construction, more targeted algorithms are appropriately chosen. By combing two fractal algorithms of recursion and IFS, two static scenarios “Old Tree, Withered Grass and Snowy Night” and “Reed Land and Snowy Night”
are built on MATLAB platform, which further enriches contents of the scene. Simultaneously, a trial has been made to experiment the generation of dynamic fractal graphs. Based on the fractal algorithm with parameters, a recursive windmill animation and a lunar eclipse animation based on recursion and IFS algorithm are generated on MATLAB platform, which enhances the creativity of the scene building of fractal animation.

With respect to the selection of implementation platform, this paper compares the effects of fractal graph generation between MATLAB and Flash. The results show that, the computing power of Flash is far less than that of MATLAB, so it might not be possible to realize the complex fractal algorithm on Flash. However, thanks to its powerful animation creation ability, Flash is undoubtedly more valuable than MATLAB in the implementation of relatively simple fractal animation effect in the light of speed and convenience. What’s more, Flash has great flexibility, which can create vivid animation effect freely and also can make the realization of man-machine interactive operation more convenient. Nevertheless, in order to achieve richer and more realistic animation effects, the more powerful MATLAB platform is preferred. In general, MATLAB and Flash both have their advantages and disadvantages. Therefore, it is necessary to choose the right platform according to actual requirements of creation so as to improve the development efficiency.

Fractal geometry, with its unique charm, enormously enriches human's cognition of the world. It has extremely high theoretical value and has attracted many scholars’ attentions in the academic circle. Nowadays, fractal theory has been applied in many fields such as signal processing, image coding, as well as new material research and development. In the future, fractal theory will achieve more fruitful results with further explorations being done. For example, in the field of architecture and home decoration, architect and building designer can use a complete fractal graph to decorate the whole surface of a wall or produce wallpaper and home ornamentation composed of fractal pattern. In the field of information security, fractal theory can contribute to information encryption technology as well. In the aspect of natural disaster warning, prediction accuracy of natural disasters such as earthquake and tsunami can be improved by using fractal theory. Besides, in the area of computer graphics, with the development of computer technology, the process of fractal image exploitation in computer will become more efficient and convenient, and the contents will be more colorful.

References
[1] Mandelbrot B (Translated by Wang J) 1991 Fractal - the geometry of nature World Science 11 1–4
[2] Lou L 2010 Communication Theory and Technology of Image (Beijing: Tsinghua University Press) (in Chinese)
[3] Zhu H and Ji C 2011 Fractal Theory and its Application (Beijing: Science Press) (in Chinese)
[4] Chang T 2013 Application of Fractal Geometry in Computer Graphics Computer CD Software and Applications 17 314–5 (in Chinese)
[5] Li S and Geng X 2018 Research on Furniture Shape Creation Based on Fractal Theory Furniture and Interior Design 1 16–7 (in Chinese)
[6] Feng Y, Tian F and Tian C 2016 Fractal Geometry Aesthetics of Traditional Chinese Gardens - Rebellion against Western Euclidean Geometry Aesthetics Huazhong Architecture 1 11–5 (in Chinese)
[7] Yan L and Yang C 2008 Application of Fractal Aesthetics in Architectural Design Journal of Harbin Institute of Technology (Social Sciences Edition) 10 28–32 (in Chinese)
[8] Tang Y, Fang K and Shen L 2009 The Comparison between Fractal Patterns and Traditional Textile Printing Patterns Journal of Textile Research 30 90–4 (in Chinese)
[9] Zhang A, He Y and Zhang J 2017 Image Compression Coding Algorithm Based on Wavelet and Fractal Theory Computer Technology and Development 27 46–50 (in Chinese)