Journal of Korea Game Society 2016 Feb; 16(1): 119-128
http://dx.doi.org/10.7583/JKGS.2016.16.1.119

컴퓨터 그래픽스

코드를 활용한 프랙탈 변형의 전파 제어 방법

한영덕
우석대학교 정보보안학과
ydhan@woosuk.ac.kr

A Propagation Control Method Using Codes In The Fractal Deformation

Yeong-Deok Han
Dept. of Information Security, Woosuk University

요 약

본 논문에서는 IFS(iterated function system) 프랙탈에서 점의 코드를 활용한 변형의 개선 방법을 고려한다. 기존의 변형방법에서는 부분적 변형이 랜덤 반복에 의해 임의로 전파되므로 대개 단조로운 느낌을 주는 모양이 나타나고 있다. 이러한 점을 개선하기 위하여 코드를 활용하여 맵의 선택을 제어하는 방법을 제안한다. 제안된 방법을 적용한 결과 프랙탈의 모양 특성이 적절히 반영된 흥미로운 변형을 얻을 수 있었다. 또한 코드에 따라 변화하는 상태변수를 도입하여 좌표의 변환 외의 다른 속성의 변형을 손쉽게 구현하는 방법도 제안한다.

ABSTRACT

In this paper, we consider an improved deformation method of IFS(iterated function system) fractal using codes of fractal points. In the existing deformation methods, the intermediate results of position dependent partial deformation propagate randomly due to the randomly selected maps of iteration. Therefore, in many cases, the obtained results become somewhat monotonous feeling shapes. To improve these limitations, we propose a method in which the selection of maps are controlled by codes of fractal points. Applying this method, we can obtain interesting fractal deformation conforming with its fractal features. Also, we propose a simple method, incorporating state variables, that can be applied to deformation of some fractal features other than position coordinates.

Keywords : Fractal(프랙탈), IFS, map(맵), Deformation(변형), Code(코드), Propagation(전파)

Received: Jan. 24. 2016 Revised: Feb. 19. 2016
Accepted: Feb. 22. 2016
Corresponding Author: Yeong-Deok Han(Woosuk University)
E-mail: ydhan@woosuk.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

© The Korea Game Society. All rights reserved. This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.otg/licenses/by-nc/3.0), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. 서 론

게임이나 영화, 가상현실 등에서 배경을 이루는 자연물들은 인공적인 기하학적 모양의 물체와 달리 불규칙한 모양을 하는 경우가 많다. 해안선이나 나뭇가지, 지형과 같이 불규칙해 보이는 한편 자기반복적인 특성을 가지는 물체들에 대한 수학적 모델은 프랙탈임이 잘 알려져 있다[1].

프랙탈은 간단한 규칙이나 알고리즘을 통해 손쉽게 생성할 수 있는데, 생성하는 구체적 조건을 달리 하면 나뭇가지나 지형, 구름과 같은 다양한 형태를 만들 수 있다. 프랙탈은 이러한 특성 때문에 배경 그래픽 생성 외에 영상의 압축이나 음악 및 예술적 이미지의 디자인 등의 다양한 분야에도 응용되고 있다[2,3,4,5,6,78].

프랙탈 이미지의 생성에는 주로 축소 맵(contraction map)의 IFS(iterated function system)을 사용한다[1]. IFS를 통해 매우 다양한 프랙탈 모양을 생성할 수 있는 반면 우리가 원하는 모양이 나타나도록 IFS를 구성하거나 변형하는 것은 쉽지 않다. 따라서 이러한 문제와 관련하여 많은 연구가 되어 왔는데, 대부분 IFS의 맵을 변형하거나 맵을 적용하는 확률적 알고리즘을 변경하는데 초점을 맞추고 있다. 이러한 방법은 임의의 IFS에도 마찬가지로 적용할 수 있는 일반성이 있는 것이 장점이지만 생성된 프랙탈의 특성에는 적합하지 못할 수도 있다.

예를 들어 나뭇가지와 같은 모양에서 특정 가지만 변형하려는 경우를 생각해 보자. IFS 맵을 변형하는 방법은 맵 변형의 효과가 프랙탈 전체에 미치게 되므로 모든 부분에 유사한 변형이 나타나게 된다. 이 경우, 프랙탈의 자기유사성은 유지되지만 제한된 부분에만 변형을 적용하려는 의도와는 다른 결과라고 할 수 있다.

한편 프랙탈에 속한 점은 생성 알고리즘에서 적용된 맵의 순서의 기록 즉 코드(code)에 의해 정해진다. 이러한 코드는 점들의 대략적 위치를 가리키는 주소와 같은 의미가 있다. 따라서 코드의 정보를 변형에 활용하면 위치에 따른 제어가 가능하고 보다 다양한 변형을 기대할 수 있다. 이러한 이유로 코드 재배열을 활용하여 변형력에 대한 반응이 프랙탈 특성을 가지게 하거나[2,9,10,11] 코드의 생성에 Markov 과정을 도입하는 방법[12,13,14,15] 등이 연구되었다. 그러나 기존의 방법들도 역시 단순한 랜덤 알고리즘에 바탕을 두기 때문에 변형된 모양이 반복(iteration)에 의해 모든 곳에 제어되지 않고 전파되어 균질한 모양을 이루는 프랙탈이 형성된다. 그러나 우리가 접할 수 있는 실제 자연에서 나타나는 프랙탈 모양들은 완전한 프랙탈이 아니며 다양한 속성이 혼합된 모양이다. 이러한 면에서 완전한 프랙탈 모양은 자연물에 비해 단조롭다고 할 수 있다.

본 논문에서는 이러한 점을 개선하는 방안으로 기존의 코드를 활용한 프랙탈의 부분적 변형의 방법의[11] 바탕 위에 역시 코드를 이용하여 변형의 전파 여부를 제어하는 새로운 방법을 제안한다. 또한 코드를 이용하여 부분적 생략, 색의 변화 등을 통해 변화된 모양을 손쉽게 얻어내는 방법도 제안한다.

2절에서는 관련 연구를 살펴보고 3절에서는 IFS 프랙탈 및 코드의 의미를 설명하며 4절에서는 코드를 이용한 변형과 변형의 전파 제어 방법의 제안을, 5절에서는 코드와 상태변수를 활용한 간단한 변형 방법을 제안하고 6절에서 결론을 맺는다.

2.관련 연구

프랙탈 변형기술은 대부분 축소 맵의 IFS에 의해 정의되는 프랙탈을 바탕으로 하고 있다. 최초의 프랙탈 변형은 Barnsley 등에[16] 의한 두 IFS 간의 확률 보간(interpolation) 방법이었는데, 중간의 과정에서 프랙탈의 형체가 흐트러지기 때문에 좋은 방법이라고는 할 수 없다. Hart와 Das[17] 및 이를 발전시킨 Burch와 Hart의[18] 방법에서는 IFS의 맵들을 고정점, 회전, 축소의 매개변수들로 분

해하고 각 맵의 매개변수 사이를 보간함으로써 좀 더 나은 프랙탈 보간 결과를 얻었다. 그러나 맵을 보간하는 방법은 일반적으로 프랙탈 도형이 가지고 있는 고유한 연결관계(topology)를 변형 도중에 유지하는 것이 어려워 직접 적절한 보간 경로를 지정해야 한다. Martyn은[19] 이러한 점을 개선하기 위해 사전에 IFS 맵을 규격화하여 변형 과정에서 모양이 안정되도록 하는 방법을 연구하기도 하였다.

한편 Dekking은[12] IFS 에 Markov 과정 이론을 도입하였으며, Barnsley 등은[13,14] 전이 확률(transition probability) 변화를 프랙탈 변형에 활용하는 연구를 하였으며, Zhang은[15] Markov 방식 하의 프랙탈 모델링의 방법에 대해 연구하였다. 또한 최근에 Yang 등은[20] Markov 방식과 코드 공간에서의 ext-IST(extended iterated shuffle transformation)을 결합한 방법을 연구하였다.

이상의 방법들은 IFS를 구성하는 맵을 변형하거나 코드의 랜덤 생성에 변화를 꾀하는 것으로서 임의의 프랙탈 생성에 대해서도 적용할 수 있는 일반성이 있는 방법이다. 그러나, 그 변형이 반복 과정에서 전체에 고루 퍼지게 되며 변형된 결과를 직관적으로 예측하기가 어렵다는 특징이 있다. 따라서 구체적인 프랙탈 도형에 대해 그 도형의 특성을 고려한 특정한 변형을 구현하고자 할 때는 어려운 점이 있다.

이와는 달리 코드가 가지고 있는 점의 위치정보를 직접 변형에 활용하는 방법이 개발되었다. Fujimoto 등은[9,10] 주어진 연속적 변형과 프랙탈 위의 점에 대응되는 코드의 재배열을 결합하여 프랙탈적 특성을 갖는 변형을 얻어내었다. 이와 유사한 방법으로 송행숙 등은[2, 11] 코드 변경의 방법의 다양화, 코드에 변환을 대응시키는 방법들을 연구하였다.

그러나 이상의 방법들은 프랙탈적인 특성을 고려한 변형이라고는 할 수 있지만 IFS의 랜덤 반복 과정에서 변형의 전파가 적절히 제어되지 못하고 있다는 점에서 미흡하다고 할 수 있다. 즉 코드를 이용하여 프랙탈의 점들에 대해 변형을 적용한 경우, 맵의 랜덤 반복에 의해 이 변형이 프랙탈의 각 부분으로 전파되도록 완전히 허용되고 있거나 [2,9,10,11], 혹은 [11]의 예와 같이 전혀 전파되지 않고 일부분의 변형에 그치게 제한되고 있어 단조로운 느낌을 주고 있다.

따라서 변형된 모양이 반복에 의해 전파되는 것을 적절히 제어하는 방식으로 위와 같은 점을 개선한다면 프랙탈적 특성과 비프랙탈적 특성이 혼합된 모양이라는 점에서 단조롭지 않고 보다 흥미로운 모양을 얻을 수 있을 것이다. 본 논문에서는 이러한 방안의 하나로 코드를 활용한 변형 전파 제어 방법을 제안한다.

3. IFS 프랙탈과 코드

프랙탈 도형은 여러가지 방법으로 생성될 수 있는데 여기서는 IFS를 통한 프랙탈 생성을 고려한다. $x-y$ 평면 위에서 정의되고 두 점간의 거리를 줄여주는 N개의 축소 맵을 $\omega_i\ (i=0,1,...,N-1)$라 하자. 이 IFS에 대한 끌개(attractor)는 점들의 집합으로서 다음과 같은 식을 만족하는 집합 S로 정의된다.

$$S=\bigcup_{i=0}^{N-1}\omega_i(S) \tag{1}$$

위 식은 S에 속한 점들에 어떠한 ω_i를 적용하여도 다시 S에 속한다는 것을 의미한다. 이러한 조건을 만족하는 S는 일반적으로 프랙탈 도형이 됨이 알려져 있다[1].

축소 맵으로는 일반적으로 다음과 같은 아핀 맵(affine map)을 사용한다.

$$\omega\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}a & b\\c & d\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}e\\f\end{pmatrix} \tag{2}$$

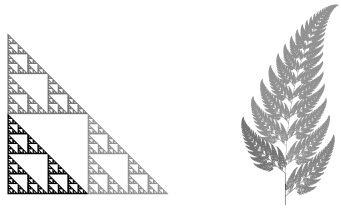
예를 들어 [Table 1][Talbe 2]의 맵에 의한 IFS에 의해서는 [Fig. 1]과 같은 시에르핀스키 개스킷(Sierpinski's gasket)과 고사리(Barnsley's fern)가 생성된다. 도형의 색은 마지막으로 적용한 맵의 번호에 따라 점을 검정(0), 빨강(1), 파랑(2), 초록(3)으로 표시한 것이다.

[Table 1] Maps for Sierpinski's gasket

map	a	b	c	d	e	f	prob.
ω_0	0.5	0	0	0.5	0	0	1/3
ω_1	0.5	0	0	0.5	0.5	0	1/3
ω_2	0.5	0	0	0.5	0	0.5	1/3

[Table 2] Maps for Barnsley's fern

map	a	b	c	d	e	f	prob.
ω_0	0	0	0	0.16	0	0	0.01
ω_1	0.85	0.04	-0.04	0.85	0	1.6	0.85
ω_2	0.2	-0.26	0.23	0.22	0	1.6	0.07
ω_3	-0.15	0.28	0.26	0.24	0	0.44	0.07



[Fig. 1] Sierpinski's gasket and Barnsley's fern

IFS에 의한 프랙탈 도형은 카오스 게임(chaos game)이라고 하는 다음과 같은 간단한 랜덤 반복 알고리즘을 통해 그릴 수 있다[1]. 즉

(i) 임의의 초기 점 (x_0, y_0)를 잡는다.

(ii) N개의 맵 중 임의의 ω_i $(0 \leq i \leq N-1)$를 확률 p_i로 선택하고 이를 (x_k, y_k)에 적용해 다음 점 (x_{k+1}, y_{k+1})을 얻는다. 즉,

$$(x_{k+1}, y_{k+1}) = \omega_i(x_k, y_k), \ (k = 0,1,2,...) \quad (3)$$

(iii) (ii)를 반복하며, 충분히 큰 정수 K에 대해서 $k > K$ 일 때부터 유효한 점으로 보고 평면에 표시한다.

각 맵은 축소맵이므로 고정점(fixed point)을 가지고 있다. 개스킷의 경우는 세 꼭지점이 고정점에 해당된다. 한번 맵을 적용할 때마다 점은 그 맵의 고정점 쪽으로 끌리게 되는데 적용하는 맵의 번호가 임의이므로 점들은 고정점들 사이의 공간에 랜덤하게 찍히며 프랙탈 도형을 이루게 된다.

위와 같은 방법에서 k번째 점 (x_k, y_k)는 최초 점 (x_0, y_0) 과 적용한 k개의 맵의 번호들 $i_1, i_2, ..., i_{k-1}, i_k$에 의하여 결정된다.

$$(x_k, y_k) = \omega_{i_k}\omega_{i_{k-1}} \cdots \omega_{i_2}\omega_{i_1}(x_0, y_0) \quad (4)$$

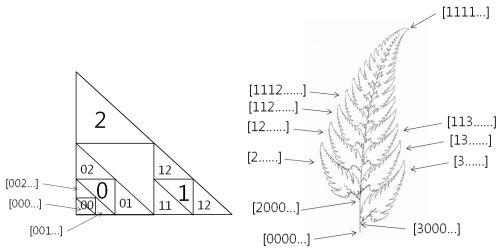
그런데 맵의 축소시키는 성질 때문에 최근에 적용한 맵은 점의 위치에 큰 영향을 주지만 적용한지 오래된 맵은 거의 영향을 주지 않게 된다. 따라서 한 점의 위치는 그 점에 도달케 한 가장 최근의 맵의 번호부터 역순으로 나열한 $i_k, i_{k-1}, ...$로 대신 표현될 수 있으며 초기 점 (x_0, y_0)와 적용한지 오래된 맵들은 무시할 수 있다. 이렇게 역순으로 나열한 맵의 번호

$$[\alpha_0, \alpha_1, \alpha_2, ...] \equiv \overline{\alpha}, \ (\alpha_0 = i_k, \ \alpha_1 = i_{k-1}, ...) \quad (5)$$

를 이 점의 코드(code)라고 정의한다. 일반적으로 프랙탈 위의 한 점 (x, y)에는 2개 이상의 코드가 대응될 수 있다. 왜냐하면 다른 맵들을 적용한 결과가 같은 점이 될 수 있기 때문이다. 그러나 주어진 코드로부터 정해지는 점은 유일하다. 따라서 코드 $\overline{\alpha} = [\alpha_0, \alpha_1, \alpha_2, ...]$로 정해지는 점 (x, y)를 다음과 같이 표기할 수 있다.

$$[\alpha_0, \alpha_1, \alpha_2, ...] \Rightarrow (x, y) = \omega_{\alpha_0}\omega_{\alpha_1}\omega_{\alpha_2}...(x_0, y_0) = P_{[\alpha_0, \alpha_1, \alpha_2, ...]} \quad (6)$$

구체적인 코드와 점의 대응 관계의 예를 [Fig. 2]에 나타내었다. 가장 최근에 적용한 맵의 번호인 α_0에 의해 점이 속한 큰 구역이 정해지고 그 전에 적용한 맵의 번호인 α_1에 의해 작은 구역이 정해지는 것을 볼 수 있으며 고사리의 경우는 이들이 겹치기도 하는 것을 볼 수 있다.



[Fig. 2] Correspondence between codes and points

점의 코드가 가지는 기하학적 의미가 모든 IFS에 대해 동일하지는 않다. 그러나 대체로 [Fig. 2]에서 볼 수 있는 바와 같이 α_0가 가장 큰 주소를 나타내고 α_1는 그 안에서의 세부주소를 나타내며 점차 세밀한 위치 정보를 나타내는 것으로 볼 수 있다.

예를 들어 고사리의 왼쪽 아래에서 2번째 가지는 코드 패턴이 $[1,2,...]$와 같은 점들이다. 이 가지가 줄기와 접한 점은 코드 [1,2,0,0,0...]의 점으로 $P_{[1,2,0,0,0...]}$로 나타낼 수 있다.

4. 코드를 활용한 변형 전파 제어

코드를 활용한 부분적 변형을 적용하고 나아가 변형된 점의 좌표가 다음 반복에서 전파되는 여부를 적절히 제어할 수 있다면 프랙탈 특성의 정도를 조절하는 결과가 되므로 보다 흥미로운 모양을 생성할 수 있다. 여기서는 다음과 같이 반복시 코드를 활용하여 전파를 제어하는 방법을 제안한다.

4.1 코드를 활용한 변형 전파 제어 방법

먼저 다음과 같은 기호들을 정의하자.

-현재 점의 코드 $\overline{\alpha}=[\alpha_0,\alpha_1,\alpha_2,...]$:
($단,\ \alpha_i=0,1,2,...,N-1$)

-현재 점 $P=\omega_{\alpha_0}\omega_{\alpha_1}\omega_{\alpha_2}...P_0$

-다음 점에 적용할 맵 번호 α_{-1} : 다음 전파방향을 판단하기 위해 사용한다.

-변형할 부분의 코드 패턴 $\overline{\alpha^{(D)}}\equiv[\alpha_0^{(D)},\alpha_1^{(D)},*,*...]$: *은 임의의 값이 허용되는 것을 의미한다.

-가할 변형 맵 T

-변형 전파가 허용될 맵 번호의 집합 $\tilde{d}\equiv\{d_1,d_2,...\}$, ($단,\ d_i=0,1,2,...,N-1$)

제안하는 방법은 다음과 같이 IFS의 카오스 게임 알고리즘을 변형한 것이다. 즉 기본적인 IFS 반복 알고리즘을 수행한다. 이 때 다음에 적용할 맵 번호 α_{-1}도 생성해 둔다. 현재 점 P의 코드 $\overline{\alpha}$가 어떤 특정 패턴 $\overline{\alpha^{(D)}}$에 해당 되는 경우는 변형 맵 T를 적용하여 변환된 점 $P'=TP$를 구하고 그렇지 않은 경우는 $P'=P$이다. 점 P'을 출력한다. T를 적용한 경우, α_{-1}이 전파 허용 번호인 $\tilde{d}$에 속하면 변환된 점으로 반복하고 그렇지 않으면 변환되기 전의 점으로 환원하여 반복한다. 이를 도식적으로 정리하면 다음과 같다.

(i) IFS 반복(코드 및 점 업데이트):

$\overline{\alpha}=[\alpha_0,\alpha_1,\alpha_2,...]\ \leftarrow\ \alpha_{-1}$ *을 왼쪽에 삽입*

$P=\omega_{\alpha_0}P''$

(ii) 다음에 적용할 맵 번호인 새로운 α_{-1}*을 생성*

(iii) 코드 패턴 검사, 변환 및 출력, 다음 맵 번호에 따라 전파여부 설정:

if $\overline{\alpha}\approx\overline{\alpha^{(D)}}$ *then* $P'=TP$, *draw* P'

$$if\ \alpha_{-1} \in \tilde{d}\ then\ P'' = P'$$
$$else\ P'' = T^{-1}P'$$
$$else\ P' = P,\ draw\ P',\ P'' = P'$$

(iv) (i)로 돌아가 반복한다.

위의 방법이 기존의 랜덤 반복에 의한 IFS 프랙탈 생성과 다른 점은 (a)현재 점의 코드가 특정 코드패턴일 때만 변형을 적용하는 점과 (b)미래에 적용할 맵의 번호에 따라 변형의 전파 허용 여부를 정하는 점이다.

(b)에서 양 극단적 경우를 살펴보면, 전파허용이 전혀 없는 경우는 [Fig. 3]에서 볼 수 있듯이 전체 프랙탈에서 다른 부분에는 전혀 변화가 없이 해당 부분만을 오려내어 변형을 적용한 것 같은 결과가 된다.[11] 또한 모두 전파 허용을 하는 경우는 IFS에서 단순히 맵을 변형한 방법으로 환원된 것으로 볼 수 있다. 왜냐하면 다음과 같이 점에 적용한 맵들을 2개씩 묶어 하나의 맵으로 간주할 수 있으므로

$$P = (\omega_{\alpha_0}\omega_{\alpha_1})(\omega_{\alpha_2}\omega_{\alpha_3})...P_0 \qquad (7)$$

시에르핀스키 개스킷의 경우는 총 $3\times3=9$개의 맵으로 구성된 IFS로도 생각할 수 있다. 이 때 예를 들어 $\overline{\alpha^{(D)}}=[1,2,*,*...]$의 패턴이 나타날 때 T를 추가하는 변형을 하고 전파 허용을 한다면, 9개 맵의 IFS로서는 전체 맵 중 $(\omega_1\omega_2)$만을

$$(\omega_1\omega_2) \Rightarrow T(\omega_1\omega_2) \qquad (8)$$

와 같이 변화시킨 것으로 간주할 수 있는 것이다.

따라서 위에서 제안한 방법은 기존의 프랙탈 변형 방법을 두 극단으로 포함하고 있으며 그 사이에서는 적절히 전파를 제어하는 융통성 있는 방법이라고 볼 수 있다.

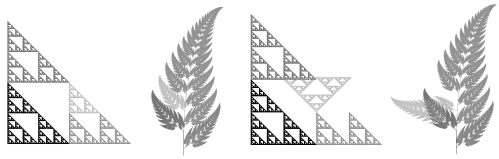
위의 방법을 적용했을 때 나타나는 모양은 다음과 같이 짐작할 수 있다. 먼저 코드 패턴이 $\overline{\alpha^{(D)}}$인 부분은 맵 T가 적용되어 직접 변환된 모양으로 나타나게 된다. 이 모양은 다시 전파가 허용된 맵들의 적용에 따라 각 맵의 고정점 방향으로 끌리며 축소된 모양으로 나타나게 된다. 이 때 전파가 허용되지 않은 맵은 변형된 부분에 직접 적용되지 못하지만 다른 맵이 적용된 이 후에는 가능하므로 전파가 허용되지 않은 맵 방향으로도 간접적으로 전파된 작은 크기의 변형 모양이 나타나게 된다.

4.2 코드를 활용한 변형 전파 제어 예

여기서는 앞에서 설명한 방법을 적용하여 코드를 활용한 부분적 변형을 적용하고 이 변형이 전파가 전혀 안되도록 한 경우, 완전히 전파되도록 한 경우, 부분적으로 전파되도록 한 경우의 예를 구체적으로 비교해 본다.

코드 패턴이 $\overline{\alpha^{(D)}}=[1,2,*,*...]$와 같은 점들만 변형하는 것을 고려하자. 이 점들은 [Fig. 3]에서 하늘색으로 표시된 부분에 해당되는데, 고사리의 경우 왼쪽 아래에서 2번째 가지를 나타낸다. 이에 대해 변형 맵은 [Table 3]의 변환 T_1을 사용했으며, T_1은 2번째 가지의 접합부에 해당하는 코드 [1,2,0,0,0...]의 점 $P_{[1,2,000...]}$을 고정점으로 하고, 회전각 $\pi/4$, 확대비율 1.5인 아핀 변환이다.

변형을 적용하되 전파 허용을 전혀 하지 않을 때 나타난 결과는 [Fig. 3]의 오른쪽 두 그림과 같다. 해당 코드 영역만 변형이 되었음을 볼 수 있다. 변환 T_1에 의해 얻은 좌표를 그리는 데에만 사용하고 반복에는 반영하지 않았기 때문이다. [11]의 예에 해당되며 단조로운 느낌을 준다고 할 수 있다.

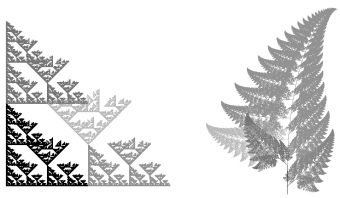


[Fig. 3] Before(left) and after(right) deformation of the region of code≈[1, 2, *, *...]; no propagation of deformation

[Table 3] Affine transformations used in examples

map	rotation	scale	(x_f, y_f)
T_1	$\pi/4$	1.5	$P_{[1,2,0,0,0...]}$
T_2	$\pi/6$	1	$P_{[2,0,0,0,0...]}$
T_3	$\pi/4$	1.5	$P_{[1,1,2,0,0,0...]}$

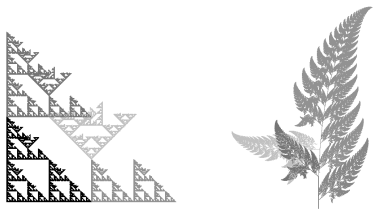
변형의 전파가 완전히 허용되었을 때 나타난 결과는 [Fig. 4]와 같다. 앞에서 설명한 것처럼 동일한 프랙탈 모양을 갖는 각각 9개와 16개의 맵으로 확장된 IFS에서 하나의 맵씩 다른 맵으로 변경한 IFS의 프랙탈로 볼 수 있다. 따라서 변형은 되었지만 전체적으로 균질한 느낌을 주며 단조롭다고 할 수 있다.



[Fig. 4] Deformation of the region of code pattern [1, 2, *, *...]; free propagation of deformation

다음의 [Fig. 5]는 $\alpha_{-1}=2$ 인 경우, 즉 다음 맵이 ω_2일 때만 변형의 전파가 허용되었을 때 나타난 결과이다. 고사리의 경우 이에 해당하는 맵은 전체를 왼쪽 아래 가지로 축소시키는 변환인데, 따라서 왼쪽 아래 첫 번째 가지에는 두 번째 가지의 변형이 직접 반영되어 있는 것을 볼 수 있다. 왼쪽 아래 세 번째 가지 내의 왼쪽 부분에 반영된 것은 이후 간접적으로 ω_1에 의해 위쪽으로 전파되어 나타난 모양이다. ω_1에 의해 직접 전파된다면 왼쪽 세 번째 가지도 두 번째 가지와 같은 모양을 하게 될 것인데 전파가 차단되어 그런 모양이 나타나지 않고 있다. 또 오른쪽 아래 가지를 만드는 ω_3 방향으로는 전파가 되지 않으므로 오른쪽 가지들의 모양은 거의 변화가 없다. 물론 간접적 변화는 생기지만 크기가 축소되어 크게 눈에 띄지 않고 있다.

[Fig. 5]를 [Fig. 3] 및 [Fig. 4]와 비교해 보면 전파의 부분적 허용을 통해 흥미로운 모양이 만들어져 프랙탈로서의 특징을 유지하면서도 단조로운 느낌이 개선되었음을 볼 수 있다.



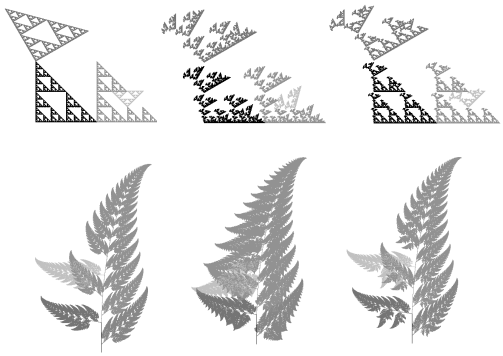
[Fig. 5] Deformation of the region of code pattern [1, 2, *, *...]; propagation direction: ω_2

다음의 [Fig. 6]은 역시 전파의 부분적 허용으로 생성된 결과로서 왼쪽의 두 그림은 $\alpha_{-1}=1\,(\omega_1)$로 오른쪽은 $\alpha_{-1}=3\,(\omega_3)$의 방향으로만 전파가 허용된 결과를 나타낸 것이다. 역시 단조로운 느낌이 개선되었음을 볼 수 있다.

[Fig. 6] Deformation of the region of code pattern [1, 2, *, *...]; propagation direction: ω_1, ω_1, ω_3.

[Fig. 7]은 여러 개의 코드 패턴 부분에 각기 다른 부분적 변형을 적용하는 경우의 결과이다. 변형을 적용한 부분은 코드 패턴이 각각 $[2,*,*...]$ 및 $[1,1,2,*,*...]$인 점들인데 고사리의 경우 왼쪽 아래에서 첫 번째와 세 번째 가지를 나타낸다. 적용한 변형 맵은 각각 [Table 3]의 변환 T_2와 T_3이다. 전파가 허용 안 된 경우(왼쪽), 완전히 허용된 경우(중앙), ω_2 방향으로만 허용된 경우(오른쪽)를 비교하여 나타내었다. 변형이 적용되는 코드 패턴

의 수를 늘리고 전파 방향을 적절히 조절함으로써 다양한 모양을 얻을 수 있음을 볼 수 있다.



[Fig. 7] Deformation of the region of code pattern [2, *, *...], [1,1,2,...]; propagation direction: none(left), free, ω_2(right).

5. 코드를 활용한 기타 변형 제어

앞 절에서 살펴본 방법은 코드에 따라 어떤 변환을 선택하여 적용하는 것이었는데 이를 상태 변수를 도입하여 확장할 수 있다. 즉 어떤 상태를 기록하는 변수를 도입하고, 매 반복시 나타나는 코드가 입력이 되어 상태 변수의 변화가 야기되며, 변화된 상태변수 값에 따라 정해진 처리를 하는 방법이다. 앞 절의 방법도 이의 특수한 예로 간주할 수 있다.

5.1 코드와 상태변수를 이용한 변형 방법

제안하는 방법은 점의 출력여부나 색을 상태 변수를 이용하여 제어하여 변형을 얻는 방법으로 다음과 같다.

점의 출력과 생략을 제어하기 위한 bool형 상태변수인 *dodraw*를 도입한다. *dodraw*는 값이 *true*이면 점이 정상적으로 출력되게 하고 *false*이면 출력이 되지 않게 하는 역할을 한다. *dodraw*의 값은 IFS 반복 과정에서 코드에 의해 변화한다. 즉 미리 정한 특정 코드 패턴이 나타나면 $dodraw = true$ or $false$로 상태가 변화한다. 따라서 반복 과정에서 모든 점이 출력되는 것이 아니며 그 시점의 *dodraw*의 값에 따라 점의 출력 여부가 정해진다.

[Fig. 8]은 이러한 방법을 적용한 결과이다. 왼쪽은 코드 패턴이 $[2,*,*...]$, $[1,1,2,*,*...]$일 때 *true*로, $[1,2,*,*...]$일 때 *false*로 상태변수가 변화하도록 했을 때 나타난 결과이며, 가운데는 $[1^{(4)},2,*,*...]$, $[1^{(6)},2,*,*...]$일 때 *true*로, $[1^{(3)},2,*,*...]$, $[1^{(5)},2,*,*...]$, $[1^{(7)},2,*,*...]$일 때 *false*로 변화하는 조건을 추가했을 때의 결과이다.

[Fig. 8] Deformation using code dependent state variables

같은 방법을 적용하되 출력 색을 나타내는 상태변수 *drawcolor*를 도입하여 얻은 결과를 [Fig. 8]의 오른쪽에 나타내었다. *drawcolor*의 값을 코드 패턴 $[2,*,*...]$, $[1,2,*,*...]$, $[1,1,2,*,*...]$, $[1^{(3)},2,*,*...]$이 나타날 때 각각 파랑, 초록, 하늘색, 빨강으로 변화되게 했을 때의 결과이다.

5.2 코드 값 조건을 활용한 변형 방법

상태 변수를 도입하는 방법 외에 코드 값의 통계적 분포모양이나 나타난 코드 값의 빈도에 대한 부등식 같은 조건 등 다양한 방법으로 점의 출력에 변화를 주어 모양을 변형할 수 있다. [Fig. 9]의 왼쪽과 가운데의 그림은 전체 코드(길이 15) 내에서 $[...,1,1,2,...]$의 패턴을 찾았을 때 발견된 횟수가 1회(왼쪽) 또는 2회(가운데) 이상 나타나면 $dodraw = false$가 되도록 했을 때의 결과이다. 오른쪽의 그림은 전체 코드 내에서 맵 번호 2와 3이

나타난 횟수가 3이상이면 $dodraw = false$가 되도록 했을 때의 결과이다. 양쪽 가지를 만드는 맵의 횟수를 제한했기 때문에 줄기 부분만 강조된 결과가 나온 것으로 볼 수 있다.

이러한 방법을 다른 IFS 프랙탈 도형에 그대로 적용할 수 있는 것은 아니다. 그러나 다루어야 할 프랙탈에 맞추어 코드 패턴과 맵 번호 등을 적절히 구체화 한다면 간단히 구현할 수 있어 쓸모 있는 방법이 될 수 있다.

[Fig. 9] Deformation using code statistics

6. 결 론

이상과 같이 본 논문에서는 IFS 프랙탈 생성시 코드의 정보를 활용한 기존의 부분적 변형 방법을 개선하는 방안으로 변형의 전파를 또한 코드를 이용하여 제어하는 방법을 제안하였다. 제안한 방법을 적용한 결과, 변형이 전혀 전파되지 않는 경우나 완전히 자유롭게 전파되는 경우에 생성되었던 단조로운 느낌의 프랙탈 모양에 비해 개선된 프랙탈 모양을 얻을 수 있었다. 또한 코드에 기반한 상태변수를 통해 좌표 변화가 아닌 다른 속성의 변화를 얻는 간단하고 구체적 방법을 제안하였고 그 적용 결과로 손쉽게 흥미로운 변형을 얻을 수 있었다. 제안된 방법들은 계산양이 적고 구현 방법도 간단하여 프랙탈 변형에 편리하게 이용될 수 있을 것으로 생각된다.

점의 코드는 점이 속한 프랙탈 내에서의 구조적 위치를 나타낸다고 볼 수 있으며 이를 활용하면 앞에서 본 바와 같이 좌표에만 의존한 방법과 다른 재미있는 결과를 얻을 수 있다. 이 외에도 다양한 활용이 있을 것으로 예상되며 앞으로 이에 대한 연구가 필요하다고 생각된다.

REFERENCES

[1] M. F. Barnsley, “Fractals everywhere”, 2nd ed. New York: Academic Press Professional, 1993.

[2] H. S. Song and Y. D. Han, “A Study of Fractal Object Deformation for Game Environment”, Journal of Korea Game Society, Vol. 5, No. 1, pp. 19-24, 2005.

[3] J. J. Lee and M. K. Kang, “3D Cloud Animation using Cloud Modeling Method of 2D Meteorological Satellite Images”, Journal of Korea Game Society, Vol. 10, No. 1, pp. 147-156, 2010.

[4] Y. C. Wee, “A Very Fast 2*2 Fractal Coding by Spatial Prediction”, Journal of KIISE: Computer Systems and Theory, Vol. 31, No. 11, 2004.

[5] J. H. Park, C. W. Park and W. S. Yang, “Fractal Image Coding for Improve the Quality of Medical Images”, J. Korean. Soc. Radiol., Vol. 8, No. 1, January 2014.

[6] J. M. Kim and H. J. Cho, “Real time Rendering of Realistic Grasses Using Fractal and Shader-Instancing”, J. Korea Multimedia Soc. Vol. 13, No. 2, pp. 298-307, Feb. 2010.

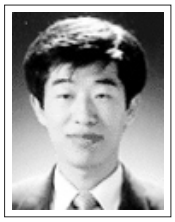
[7] A. Das and P. Das, “Fractal analysis of songs: Performer′s preference”, Nonlinear Analysis: Real World Applications, Vol. 11, Iss. 3, pp. 1790~1794, 2010.

[8] S. Draves, “The Fractal Flame Algorithm”, 1992.

[9] T. Fujimoto, Y. Ohno, K. Muraoka and N. Chiba, “Fractal Deformation Based on Extended Iterated Shuffle Tranformation”, NICOGRAPH International 2002, pp. 79-84, 2002.

[10] T. Fujimoto, Y. Ohno, K. Muraoka and N. Chiba, “Fractal Deformation Using Displacement Vectors Based on Extended Iterated Shuffle Tranformation”, The Journal

of the Society for Art and Science, Vol. 1, No. 3 pp. 134-146, 2002.
[11] Y. D. Han and G. O. Kim, “Fractal Deformation using Code and Displacement Vectors”, International Journal of Contents, Vol. 7, No. 12, pp. 322-332, 2007.
[12] F. M. Dekking, “Recurrent sets, Advances in Mathematic”, Vol. 44, pp. 78-104, 1982.
[13] M. F. Barnsley, M. A. Berger and H. M. Soner, “Mixing Markov chains and their images, Probability in the Engineering and Informational Sciences”, 2(04), pp. 387-414, 1988.
[14] X. Liu and W. Zhu et al., “Research on the effect of the parameters of Markov iterated function systems”, Computer Science, 27, pp. 68-71, 2000.
[15] L. Zhang, “A fractal modeling method based on Markov matrix”, Computer Applications and Software, 27(12), pp. 115-117, 2010.
[16] M. F. Barnsley, A. Jaquin, L. Reuter and A. D. Sloan., “A Cloud Study”(Animation), Georgia: The Computergraphical Mathematics Laboratory at Georgia Institute of Technology, 1987.
[17] J. C. Hart and S. Das, “Sierpinski blows his gasket”(Animation), SIGGRAPH Video Review 61, 1990.
[18] B. Burch and J. Hart, “Linear fractal shape interpolation”, Proceedings of the Graphics Interface, Vol. 97, pp. 155 - 162, 1997.
[19] T. Martyn, “A new approach to morphing 2D affine IFS fractals”, Computer & Graphics, Vol. 28, No. 2, pp. 249 - 272, 2004.
[20] H. Yang, M. Zhou and H. Zheng, “Improved Fractal Deformation Based on Markov Iterated Function Systems”, Journal of Information & Computational Science 10:2, pp. 365 - 373, 2013.



한 영 덕(Han, Yeong Deok)

약력 : 1993 한국과학기술원 물리학 박사
1993 국과학기술원 연수 연구원
1994- 우석대학교 정보보안학과 교수

관심분야 : 게임물리, 게임엔진, 게임그래픽