코드와 변위 벡터를 이용한 프랙탈 변형

Fractal Deformation using Code and Displacement Vectors

한영덕, 김기옥
우석대학교 게임콘텐츠학과

Yeong-Deok Han(ydhan@woosuk.ac.kr), Gi-Ok Kim(gokim@woosuk.ac.kr)

요약

프랙탈의 특성에 알맞은 변형 방법을 고려하였다. IFS 프랙탈에서 점의 위치 특성은 공간적 좌표뿐만 아니라 코드에 의해서도 표현된다. 코드는 프랙탈 내에서의 점의 주소로 볼 수 있는데, 코드값을 바꾸어 생기는 점의 이동에 프랙탈적 특성이 있으므로, 코드의 정보를 이용한 세 가지 변형 방법을 제안하였다. 즉, 한점의 이동에 사용될 벡터로서 1) 코드 변환을 통해 얻어지는 다른 점에서의 벡터장 값을 이용하는 방법과 2) 코드 정보를 활용하여 변위 벡터를 정하는 방법을 구현하여 본 결과, 연속적 변형의 특징과 프랙탈적 특징을 모두 갖는 변형을 얻을 수 있었다. 또한 3) 변형이 가해질 영역을 코드를 활용하여 제한함으로써, 고사리의 경우, 보다 자연물에 적합한 변형을 얻을 수 있었다.

■ 중심어 : | 프랙탈 | 변형 | IFS | 코드 | 변위 벡터 |

Abstract

We consider a deformation method suitable for fractal. In IFS fractal, the position of a point is characterized by its code as well as by its coordinates. Code has a meaning of address for fractal. If we move a point by changing its code, the resulting movement shows fractal behavior. We propose three deformation methods based on code information. For the deformation vector of a point in fractal, 1) we use the vector of a given vector field at the point obtained by code transformation, 2) we use the vector constructed by adding predefined displacement vectors according to the code information of the point. Both methods show a fractal-like character as well as an ordinary continuous deformation character. Also, 3) we can deform fern-fractal more naturally by restricting its deforming region using code form.

■ keyword : | Fractal | Deformation | IFS | Code | Displacement Vector |

I. 서 론

게임 등에서 사용되는 그래픽 기술들은 물체나 배경을 실제의 것처럼 보이게 하는데 목표를 두고 있다고 할 수 있다. 주인공인 캐릭터의 경우는 물리적 효과와 같은 여러 기법들을 동원하여 사실에 가깝게 자연스럽게 표현하고 있다. 하지만 사용자들이 현실감을 느끼게 하기 위해서는 배경 및 주변 물체들도 사실적으로 표현되어야 한다. 즉 자연물인 나무, 해안선, 구름, 지면 등과 같은 것들을 얼마나 잘 표현하는가 하는 것이 중요하다.

접수번호 : #071029-001
접수일자 : 2007년 10월 29일
심사완료일 : 2007년 11월 23일
교신저자 : 한영덕, e-mail : ydhan@woosuk.ac.kr

이러한 자연물들을 나타내는 수학적 모델은 프랙탈임이 잘 알려져 있다[1]. 삼각형이나 구형과 같은 기하학적 도형을 바탕으로 하여 자연의 물체를 나타내는 경우는 부자연스러운 점이 있을 뿐만 아니라 세밀한 표현을 위해서는 많은 양의 메모리가 필요해지는 등의 단점이 있는 반면에 프랙탈을 이용하면 자연물들인 구름, 나뭇가지, 지형, 해안선 등에 대해 사실에 근접한 묘사를 할 수 있으며. 간단한 구조로서 메모리를 절약하여 수행속도를 높일 수 있는 장점이 있다.

한편 게임과 같은 환경에서 프랙탈을 이용하여 생성·표현된 물체들의 사실감을 보다 높이기 위해서는 변형력과 같은 영향이 가해지는 경우는 그 프랙탈 도형을 그에 따라 적절하게 변형해 주어야 한다. 그런데 기하학적 도형의 경우는 그 도형이 그려진 바탕 물체를 구부리거나 잡아 늘이는 것에 의해 변형하는 것도 만족스러운 변형 방법이라고 할 수 있으나 자연물을 묘사하고자 하는 프랙탈 도형의 경우는 실제 자연물의 움직임과 달라 보여 사실감을 떨어뜨릴 수 있다. 예를 들어 나뭇가지가 바람에 의해 움직이는 경우 나뭇가지가 그려진 고무판이 휘어지는 방식으로 나뭇가지가 변형되어 보이는 것이 아니라 전체적으로는 어느 방향으로 쏠리듯이 움직이되 세부적인 가지 하나하나는 서도 다른 방향으로 움직이기도 하는 것이다. 즉 프랙탈 도형의 각 부분마다 약간씩 다른 변형이 적용되는 방식이 프랙탈이 표현하고자하는 자연물의 움직임을 보다 사실적으로 표현한다고 할 수 있다.

본 연구에서는 위와 같은 점을 고려하여 프랙탈을 그에 가해진 변형 벡터장에 따라 변형하되 일률적으로 변형하지 않고 각 부분마다 프랙탈의 특성을 살려 다르게 변형하는 방법을 제안하고자 한다.

II. 관련 연구

물체의 모양변형 기술은 모양을 만들거나 수정 할 때 유용한 기술로 다양한 방법들이 제안되어 왔다. 예를 들면, 물체가 놓여있는 공간의 3D격자를 변화시켜 물체를 변형하는 자유 형체 변형(Free Form Deformation, FFD)[4][8]이나 모핑(Morphing) 및 구부림(Warping)[4] 등이 있다. 이들은 단일 물체의 형체를 변화시키거나 모양이 다른 두 물체 사이에 자연스러운 연속적 변화를 만들어 내는 것으로 컴퓨터 그래픽스 분야에서 많이 사용되는 기술이다. 이와는 또 다른 방법으로 프랙탈 변형 방법이 많은 시도되어 왔는데[2][3][5-7][9-11] 연속적 변형기술과는 완전히 다른 변형의 스타일의 기술이라고 할 수 있다.

프랙탈 변형기술은 대부분 축소사상(contraction map) 의 IFS(Iterated Function System)에 의해 정의되는 프랙탈을 바탕으로 하고 있다. 이 경우, 프랙탈이 IFS의 끌개(attractor)로 정의되므로[1] 축소 사상들을 변화시키면 변화된 IFS 끌개 즉 변화된 프랙탈을 만들어 낼 수 있다. [2]에서는 각 축소사상의 고정점과 각 축소 사상의 끌어당기는 강도를 변화시켜 변형하였으며, [2][3][7]에서는 IFS 끌개의 개념을 활용하여 그의 전체적인 모양을 변형시키고 있다. 그러나 이러한 기술들은 IFS의 고정점들의 위치나 축소 사상의 계수 같은 매개변수 값들로 조절되기 때문에 사용자가 모양 변화에 대하여 위치별로 부분적인 제어를 하지는 못한다. 또한 일부 모양들에서는 어떻게 변형될지 직관적으로 예측하기가 어렵다는 특징이 있다.

프랙탈도 하나의 사실적 물체라고 볼 때, 바람직한 변형 방법이기 위해서는 전체적인 모양의 변화는 일반적인 물체가 힘을 받았을 때와 비슷해야 할 것이다. 즉 전반적인 변형은 프랙탈에 가해지는 힘의 벡터장에 따라 일어나야 한다. 그러나 세부적으로는 프랙탈의 특성을 나타내는 변형방법이어야 한다. 즉 단순히 프랙탈이 그려진 고무판이 늘어나는 방식으로 변형되지 않고 전체 모양이 축소되어 반복되는 프랙탈의 특성이 반영된 변형방법이어야 한다.

최근 후지모토 등에 의하여 프랙탈 위의 점에 대응되는 코드를 활용하여 이의 변화로부터 프랙탈적 특성을 갖는 변형을 얻어내는 방법이 제시되었다[12][13]. 본 논문에서는 코드가 그 점의 단순한 기하학적 위치보다 프랙탈적 특성을 더 잘 반영한다는 것에 주목하여 이를 이용한 또 다른 변형 방법들을 고려하여 보았다. 즉 코드에 따라 변형의 적용을 적절히 다르게 함으로써 프랙

탈적 특성을 가진 변형을 얻을 수 있음을 제시하고자 한다.

III. 코드를 이용한 프랙탈 변형

여기서는 IFS에 의한 프랙탈과 코드의 의미를 설명하고 일반적 변형 방법 및 이와 대비되는 코드를 이용한 변형 방법 세 가지를 제시하려 한다.

1. IFS 에 의한 프랙탈 생성과 코드

프랙탈 도형을 생성하는 방법은 여러 가지가 있다. 여기서는 그 중 IFS를 통한 프랙탈 생성과 프랙탈 위의 점에 대한 코드에 대해 알아본다.

공간 위에서 정의되고 두 점간의 거리를 줄여주는 N 개의 축소사상을 F_i $(i=0,1,...,N-1)$라 하자. 이 함수들(Iterated Function System)에 대한 끌개(attractor)는 점들의 집합으로서 다음과 같은 식을 만족하는 집합 S로 정의된다.

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

즉 S에 속한 점들은 어떠한 F_i를 가하여도 다시 S에 속하는 성질을 가진다. 이러한 S는 일반적으로 프랙탈 도형이 됨이 알려져 있다[1]. 예를 들어

$$\begin{aligned} F_0(\vec{x}) &= \begin{pmatrix}0.5 & 0\\ 0 & 0.5\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}0\\ 0\end{pmatrix}\\ F_1(\vec{x}) &= \begin{pmatrix}0.5 & 0\\ 0 & 0.5\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}0.5\\ 0\end{pmatrix}\\ F_2(\vec{x}) &= \begin{pmatrix}0.5 & 0\\ 0 & 0.5\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}0\\ 0.5\end{pmatrix} \end{aligned} \tag{2}$$

와 같은 3개의 축소사상으로 구성된 경우 이에 해당하는 프랙탈 도형은 [그림 1]과 같은 잘 알려진 시에르핀스키(Sierpinski) 개스킷(gasket)이 된다.

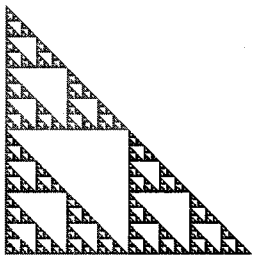


그림 1. 시에르핀스키 개스킷

또한 몇 개($N=4$)의 축소사상들을 적당히 설정한 IFS에 의하면 [그림 2]와 같이 고사리(spleenwort fern)와 같은 모양을 가진 프랙탈을 얻을 수 있음도 잘 알려져 있다[1]. 본 논문에서는 코드를 이용한 변형 방법의 구체적 적용 예로 이 두 프랙탈 도형을 사용하였다.

그림 2. 고사리

프랙탈의 정의 (1)로부터 직접 프랙탈 도형을 그리는 것은 쉽지 않다. 그러나 다음과 같은 간단한 랜덤 이터레이션 알고리즘으로 해당 프랙탈을 그릴 수 있다[1]. 즉

(i) 먼저 임의의 점 $\vec{x_0}$를 잡는다.

(ii) $\vec{x_k}$ $(k=0,1,2,...)$ 에 대하여, $0 \leq i_k \leq N-1$ 의 범위에서 임의의 i_k를 선택하고 이로부터 정해지는 축소 사상 F_{i_k}를 통해 $\vec{x}_{k+1}$를 얻는다. 즉

$$\vec{x}_{k+1}=F_{i_k}(\vec{x_k}) \tag{3}$$

이 때 충분히 큰 정수 K에 대해서 $k > K$ 일 때부터 유효한 점으로 보고 평면에 표시한다.

각 사상은 축소사상이므로 고정점(fixed point)을 가지고 있다. 시에르핀스키 개스킷의 경우는 세 꼭지점이 고정점에 해당된다. 한번 사상을 가할 때마다 점은 그 사상의 고정점 쪽으로 끌리게 되는데 가하는 사상의 번호가 임의이므로 점들은 고정점 사이의 공간에 랜덤하게 찍히게 된다. 이 점들이 프랙탈 도형을 이루게 된다.

위와 같은 방법으로 얻어진 점의 열들은 최초 점 $\overrightarrow{x_0}$ 및 적용한 사상의 번호 $n_1, n_2, n_3, \ldots$에 의하여 결정된다. 그런데 사상의 축소시키는 성질 때문에 최근에 가한 사상은 점의 위치에 큰 영향을 주는 반면 가한지 오래된 사상은 점의 위치에 작은 영향을 주게 된다. 따라서 한 점의 위치는 그 점에 도달케 한 최근의 사상의 번호들로 근사적으로 표현되며 초기점 $\overrightarrow{x_0}$와 가한지 오래된 사상들은 무시할 수 있다. 예를 들어 100번의 사상을 가했다면 후반부의 사상 번호인 $\ldots, n_{98}, n_{99}, n_{100}$ 가 점의 위치를 거의 결정한다. 따라서 이를 고려하여 번호들을 역순으로 나열하여 $[m_1, m_2, m_3, \ldots]$ ($\equiv [n_{100}, n_{99}, n_{98}, \ldots]$) 과 같이 만든 것을 이 점의 코드(code)라고 한다.

시에르핀스키(Sierpinski) 개스킷에 대한 코드의 예를 [그림 2]에 나타내었다. 코드의 첫째자리 m_1이 각각 0, 1, 2 인 영역과 $m_1 = 0$ 이고 m_2가 각각 0, 1, 2 인 영역을 표시하고 있다.

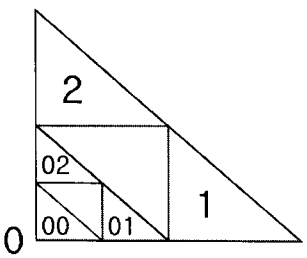


그림 3. 점의 위치와 코드의 관계

단 위와 같은 코드화에 있어서는 프랙탈 위의 점 $\overrightarrow{x}$에 대응되는 코드가 2개 이상일 수 있다. 왜냐하면 다른 순서로 사상을 가한 결과가 같은 점이 될 수 있기 때문이다. 그러나 주어진 코드로부터 정해지는 점 $\overrightarrow{x}$는 유일하다.

$$[m_1, m_2, \ldots] \Rightarrow \overrightarrow{x} \tag{4}$$

점의 코드가 가지는 기하학적 의미는 모든 IFS에 대해 동일하지는 않다. 그러나 대체로 [그림 3]에 나타난 바와 같이 m_1이 가장 큰 주소를 나타내고 m_2는 그 안에서의 세부주소를 나타내는 방식으로 점차 세밀한 위치 정보를 나타내는 것으로 볼 수 있다.

2. 변형 벡터장에 따른 연속적 프랙탈 변형

게임 배경 등에서 사용하는 프랙탈 물체는 게임 내의 상황에 따라 받는 힘이나 충돌 등에 반응하여 변형될 필요가 있는데 일반적으로 물체의 변형은 그 물체가 속한 공간을 벡터장을 통해 변형함으로써 구현할 수 있다. 즉 $\overrightarrow{x}$ 의 이동점 $\overrightarrow{x'}$ 를 연속적 변형 벡터장 $\overrightarrow{V}(\overrightarrow{x})$ 를 통해

$$\overrightarrow{x'} = \overrightarrow{x} + \overrightarrow{V}(\overrightarrow{x}) \tag{5}$$

와 같이 정하는 방법이다. [그림 1, 2]의 프랙탈 도형에 대해 벡터장 $\overrightarrow{V}(\overrightarrow{x})$를

$$\begin{cases} V_x(x, y) = 5A\,(y - 0.3)^3 \\ V_y(x, y) = A\,x\,(x^2 - 1.5x + y + 0.5) \end{cases} \tag{6}$$

와 같은 형으로 잡았을 때, (5)를 적용하여 변형한 것이 [그림 4]에 나타나 있다. 작은 스케일에서의 프랙탈 모양은 벡터장의 변화에 따라 위치마다 다르게 변형되어 있지만 점들의 연결 상태는 동일함을 볼 수 있다.

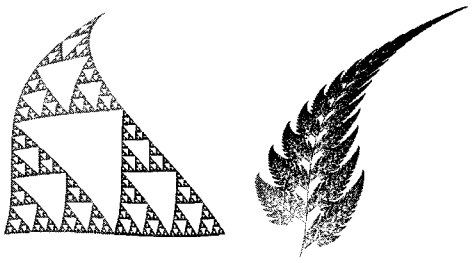


그림 4. 벡터장에 의한 연속 변형의 예

이러한 변형은 일반적 물체의 변형으로는 적당하나 자연물을 묘사하는 프랙탈의 변형으로서는 부자연스러운 점이 있다. 앞서 언급한 바와 같이 바람에 의해 움직여진 나뭇가지의 예라면 점이 어느 가지에 속했는가에 따라 서로 다른 이동량을 가지게 해야 자연스럽게 보일 것이다. 그런데 위에서 설명한 바와 같이 점이 속한 가지를 나타내는 것은 점의 코드라 할 수 있다. 따라서 프랙탈적 특성을 고려한 변형이 되기 위해서는 공간 좌표뿐만이 아니라 점의 코드에 따라서도 점의 이동이 달라지는 방식이 되어야 할 것이다.

이하에서는 코드 변환을 통해 변형을 얻는 방법과 코드에 변위벡터를 대응시켜 변형을 얻는 방법 그리고 코드를 통해 점이 속한 부분을 구분하고 이에 따라 달리 변환을 적용하는 방법 세 가지를 제시하고자 한다.

3. 코드 변환과 이를 이용한 프랙탈 변형

프랙탈적 특성을 보이는 변형이면서도 전체적으로는 앞의 연속 변형과 비슷한 변형을 만들어내기 위해 다음과 같은 방법을 생각할 수 있다. 즉 기본적인 변형 벡터장은 연속 변형의 것과 같은 것을 사용한다. 그러나 주어진 점을 이동하는데 적용할 벡터를 그 점에서의 벡터값을 사용하지 않고 약간 다른 값을 사용한다. 이러한 방법이 연속변형과 대략 비슷한 변형이 될 것은 당연하다. 이제 다른 벡터값을 사용하는 방법에 있어서 근처의 다른 점의 벡터를 사용하는 것을 고려하자. 이때 다른 점을 정하는 방법에 코드의 변환을 이용할 수 있다.

어떤 점이 가진 코드의 일부를 변환하여 새 코드를 만들면 이에 대응되는 점은 근처의 다른 점이 된다. 예를 들어 시에르핀스키 개스킷에서 코드의 셋째 자리와 5번째 자리를 교환하는 코드 변환을 하면 [1111201...]과 [1111001...]은 각각 새로운 점 [1121101...]과 [1101101...]로 이동한다. 이 때 코드의 둘째 자리까지는 같으므로 큰 스케일에서의 위치 변화는 없으며 또한 여섯째 이후는 바뀌지 않으므로 아주 작은 스케일에서도 점의 프랙탈 내의 상대적 위치에는 변화가 없다. 그러나 셋째에서 다섯째 자리 사이의 코드가 바뀌었으므로 이 스케일에서는 다른 영역으로 이동하였다고 할 수 있다. 따라서 코드 변환에 의하여 점을 이동하는 방법은 프랙탈의 특성을 고려한 변환이라 할 수 있으며 코드 변환을 통해 대응된 다른 점 위에서의 벡터장을 원래 점의 이동에 사용하는 변형 방법은 일반적 연속 변환과는 달리 프랙탈적 특성을 보이게 된다.

위의 방법을 정리하면 다음과 같다. N개의 축소 사상으로 이루어진 IFS 프랙탈 및 변형 벡터장 $\vec{V}(\vec{x})$가 주어졌을 때,

(i) 프랙탈 위의 각 점 $\vec{x}$의 코드 $[m_1, m_2, m_3, ...]$를 구한다. (랜덤 알고리즘으로 프랙탈을 생성한다면 점과 코드는 동시에 구해진다.)

(ii) (i)의 코드를 특정한 변환 방법에 따라 변환하여 새 코드 $[m'_1, m'_2, m'_3, ...]$를 얻고 이에 대응되는 점 $\vec{x'}$를 구한다.

(iii) 새로운 점 $\vec{x'}$에서의 벡터값 $\vec{V}(\vec{x'})$을 이용해

$$\vec{x''} = \vec{x} + \vec{V}(\vec{x'}) \quad (7)$$

와 같이 이동점 $\vec{x''}$을 얻는다.

이러한 방법에서 (ii)의 코드의 변환에는 특별한 제한이 없으므로 그 가짓수가 무한하다. 그리고 각각은 다른 변형 결과를 생성할 것이다. 그런데 코드 변환에서 i가 충분히 큰 m_i에 대한 변환은 작은 스케일에서의 변환으로 $\vec{x}$와 $\vec{x'}$의 차이가 매우 작아 무시할 수 있으므로 유한한 구간의 코드만 변환하는 방법들만 고려하여도 실용적으로는 충분하다고 볼 수 있다. 따라서 본 연구에서는 전체 코드 중 일정 구간의 m_i 즉, 적당한 두 자연수 k_b, k_e에 대하여 $k_b \le i \le k_e$ 인 구간에 속한 m_i들만 변환하는 것을 고려하였다.

후지모토 등의 결과는 위의 방법에서 코드의 변환 방법으로 k_b, k_e 사이의 코드를 역순으로 재배열하는 방법을 사용한 것에 해당한다[12][13]. 즉 작은 스케일과 큰 스케일의 위치를 서로 뒤바꾸는 변환을 사용하였다고 할 수 있다.

본 논문에서는 일반적으로 코드 변환이 평행 이동이나 회전과 같은 어떤 단순한 기하학적인 변환과 달리 프랙탈적 특성을 보인다는 점에 주목하여 보다 여러 가지 방식의 코드변환을 변형에 적용하여 보았다. 특히

코드 값의 배치를 변화시키는 대신 코드 값을 다른 값으로 바꾸는 변환을 적용하였다. 다음은 이러한 몇 가지 코드 변환의 결과로 얻어진 변형의 예들이다.

3.1 코드값의 치환(permutation)에 의한 변환

여기서는 코드의 일부구간 즉 $k_b \le i \le k_e$ 의 m_i에 대하여 값을 $0 \rightarrow 1$, $1 \rightarrow 2$, ... 와 같이 치환시키는 변환을 사용하여 변형하였다. 즉 다음의

$$P_N(m) = \begin{cases} m+1 & (0 \le m < N-1) \\ 0 & (m = N-1) \end{cases} \tag{8}$$

과 같이 정의된 함수를 이용하여

$$m'_i = P_N(m_i) \quad (k_b \le i \le k_e) \tag{9}$$

와 같이 코드를 변환한 뒤 (7)식을 적용하였다.

개스킷과 고사리에 적용된 결과를 [그림 5]에 나타내었다. k_b, k_e 값에 따라 변형이 나타나는 스케일이 달라지며, k_b, k_e 값이 작으면 커다란 삼각형이나 가지의 부분에서 변형이 나타나고 값이 크면 작은 삼각형이나 가지의 부분에서 변형이 나타나게 되는데 $k_b = 5$, $k_e = 7$ 의 경우를 예로 나타내었다. 전체적으로는 [그림 4]의 연속 변형과 비슷하나 개스킷의 경우는 작은 삼각형으로 분할되어 변형된 것을 볼 수 있으며 고사리의 경우는 6~7번 째 가지 부근에서 약간 어긋나는 모습을 하고 있는 것을 볼 수 있다. 일반적으로 k_b, k_e 값을 조절하면 개스킷의 삼각형 조각의 크기나 고사리의 어긋난 가지의 위치를 변화시킬 수 있다.

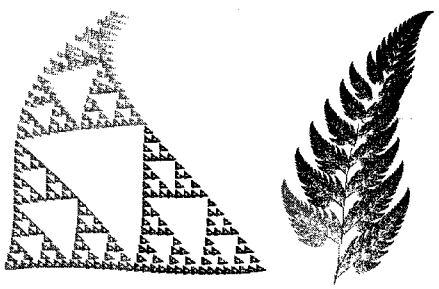


그림 5. $k_b = 5$, $k_e = 7$

3.2 코드값 끼리의 교환에 의한 변환

여기서는 코드의 일부구간 즉 $k_b \le i \le k_e$ 의 m_i에 대하여 특정 값끼리 서로 바꾸는 변환을 사용하였다. 예를 들어 $1 \rightarrow 2$, $2 \rightarrow 1$ 로 바꾸는 변환 즉,

$$P_{12}(m) = \begin{cases} 2 & (m = 1) \\ 1 & (m = 2) \\ m & (m \ne 1, 2) \end{cases} \tag{10}$$

과 같이 정의된 함수를 이용하여

$$m'_i = P_{12}(m_i) \quad (k_b \le i \le k_e) \tag{11}$$

와 같이 코드를 변환하였다.

그 변형된 결과를 [그림 6][그림 7]에 나타내었는데 변환의 효과를 파악하기 쉽게 변환 구간의 크기가 최소인 $k_b = k_e$ 경우를 나타내었다. 앞의 치환에 의한 방법과 마찬가지로 k_b, k_e 값이 작으면 큰 스케일에서의 변화가 크면 작은 스케일에서의 변화가 나타남을 볼 수 있다. 개스킷에서는 나뉘는 삼각형의 크기가 고사리에서는 방향이 틀어지는 가지의 위치가 달라짐을 볼 수 있다.

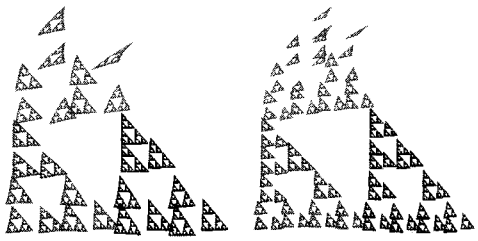


그림 6. $k_b = 2$, $k_e = 2$와 $k_b = 3$, $k_e = 3$

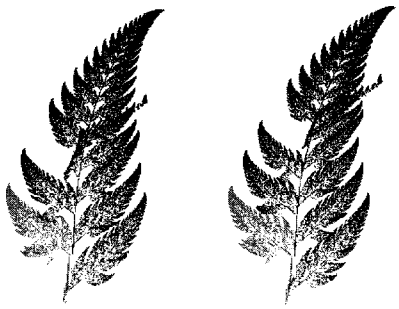


그림 7. $k_b = 2$, $k_e = 2$와 $k_b = 3$, $k_e = 3$

3.3 코드값 사이에 삽입

여기서는 코드 내의 일부 구간 즉 $k_b \le i \le k_e$ 에서

그 사이에 특정 값(예를 들어 0) 을 삽입하는 변환을 사용하였다. 즉

$$[m_1,\cdots,m_{k_b},m_{k_b+1},\cdots,m_{k_e},\cdots] \rightarrow [m_1,\cdots,m_{k_b},0,m_{k_b+1},0,\cdots,m_{k_e},\cdots] \quad (12)$$

와 같은 변환으로써, 앞의 두 변환은 코드 변환이 적용되는 범위 밖의 코드는 불변인데 반하여 여기서는 뒷부분의 모든 코드가 영향을 받아 밀리게 되어 k_b+1 번째 부터 모든 코드 값이 바뀌는 결과가 된다. 따라서 작은 스케일의 모양까지 영향을 받게 되는 특성이 있다.

변환에 의한 결과를 [그림 8][그림 9]에 나타내었다. 앞의 두 방법에 비해 불규칙성이 증가한 변형 모습을 보여주고 있으며, 여기서도 k_b, k_e 값에 따라 삼각형 조각의 크기나 변형되는 고사리 가지의 위치가 달라짐을 볼 수 있다.

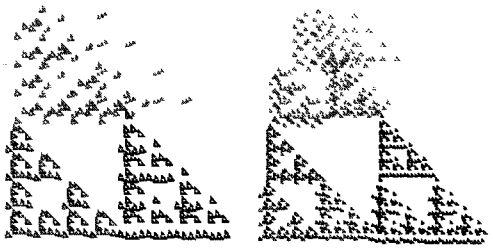


그림 8. $k_b=1$, $k_e=4$와 $k_b=2$, $k_e=5$

그림 9. $k_b=1$, $k_e=4$와 $k_b=2$, $k_e=5$

3.4 코드의 특정 값을 다른 값으로 대치

여기서는 코드 내의 일부 구간 즉 $k_b \le i \le k_e$ 의 m_i에 대하여 코드 값이 특정 값 p인 경우 다른 특정 값 q로 대치하는 방법을 사용하였다. 즉

$$R_{pq}(m)=\begin{cases} q & (m=p) \\ m & (m\neq p) \end{cases} \quad (13)$$

과 같이 정의된 함수를 이용하여

$$m'_i = R_{pq}(m_i) \quad (k_b \le i \le k_e) \quad (14)$$

와 같이 코드를 변환하였다. 이 변환은 $R_{pq}(p)=R_{pq}(q)=q$의 특징이 있으므로 서로 다른 두 코드가 같아 질 수도 있으며 따라서 서로 다른 점들에 동일한 이동 벡터를 적용하는 경우가 생긴다. $p=2$, $q=0$인 경우의 결과를 [그림 10]에 나타내었다. 동일한 이동 벡터의 적용 효과로 삼각형의 모양이 상당히 유지되고 있는 것을 볼 수 있다.

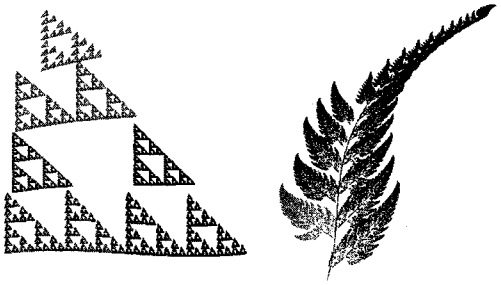


그림 10. $k_b=2$, $k_e=4$ **공통**

이상에서 살펴본 방법들은 후지모토 등의 방법과 마찬가지로 프랙탈적 특성을 띠는 변형 결과를 주고 있다. 또한 연속 변형의 모습과 대략 닮은 변형이 됨을 볼 수 있다. 이는 코드의 변환에 의한 점과 점의 대응이 갖는 일반적 특성과 벡터장에 의한 변형을 적절히 조합한 결과인데, 후지모토 등이 사용한 코드를 역순으로 재배열 하는 변환보다 단순한 코드 변환으로도 얼마든지 유용한 변형을 만들 수 있음을 보여 준다.

4. 코드와 변위벡터를 이용한 프랙탈 변형

앞의 예들에서 본 바와 같이 변형이 프랙탈적 특성을 띠는 것은 점의 코드를 활용하여 변형한 데에 원인이 있다. 이러한 점에 착안하여 여기서는 보다 직접적으로 점의 코드에 적당한 이동 벡터를 대응시켜 변환하는 방법을 고려하였다.

4.1 코드에 변위벡터를 대응시킨 변형

여기서는 외부의 영향을 나타내는 변형 벡터장을 적용하기에 앞서 프랙탈의 자체적인 모양 변화를 얻어내기 위하여 점의 코드에 점을 이동시킬 어떤 벡터를 대응시키는 방법을 고려하였는데, 각 코드 값마다 특정 변위 벡터를 대응시켜 움직임을 주는 변형을 고려하였다. 즉 IFS 의 사상의 개수 N만큼 미리 적당한 벡터들 $\{\vec{d}_0, \vec{d}_1, ..., \vec{d}_{N-1}\}$을 준비한 뒤 각 자리의 코드값 $m_i = 0, ..., N-1$에 따라 벡터를 대응시켜 점을 이동시킨다. 이 때, 앞의 방법과 같이 코드의 일부구간 즉 $k_b \le i \le k_e$ 에 대해 변환을 적용하였다. 또한 i가 클수록 작은 스케일의 위치를 나타내므로 i가 증가하면 적당한 비율 δ ($0 < \delta < 1$)만큼 이동량을 감소하게 하여 작은 스케일에서 프랙탈의 모양이 과도하게 변화되는 것을 억제 할 수 있도록 하였다. 즉 다음과 같은 식으로 위치를 이동시킨다.

$$\vec{x'} = \vec{x} + \sum_{k_b \le i \le k_e} \delta^{(i-k_b)} \vec{d}_{m_i} \qquad (15)$$

식 (15)에서 δ의 값은 공비의 역할을 하는데 이를 조절함으로써 i가 커질 때의 기여도를 조절할 수 있다. 여기에서는 $\delta = 0.65$의 값을 사용하였다. 또한 변위벡터들은 서로 평행하지 않은 임의의 방향으로 적당히 잡았으며 개스킷의 경우($N=3$)는 $\vec{d}_0$부터 $(0.1, 0.1)$, $(0.1, 0.2)$, $(0.2, 0)$를, 고사리의 경우($N=4$)는 추가로 $(-0.3, -0.1)$를 잡아 사용하였다. 변형의 결과를 [그림 11][그림 12]에 나타내었다.

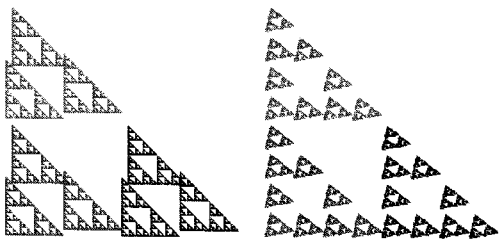


그림 11. $k_b = 1$, $k_e = 1$와 $k_b = 3$, $k_e = 5$

이 방법은 코드 값마다 주어진 고정된 벡터를 대응시켜 변형하므로 [그림 11]에 잘 나타나 있는 것과 같이 보다 기하학적인 특성을 띠는 변형이 되고 있다. 따라서 변형 모습에 규칙성이 있고 보다 인위적인 컨트롤을 하려는 경우에 적절한 방법이 될 것으로 생각된다. 여기서도 k_b, k_e의 값에 따라 삼각형과 가지에서 변형이 일어나는 스케일이 달라지고 있음을 볼 수 있다.

그림 12. $k_b = 1$, $k_e = 1$와 $k_b = 3$, $k_e = 5$

4.2 변위벡터와 변형벡터장을 중첩한 변형

앞의 방법에 외부의 변형 벡터장까지 추가하여 프랙탈 특성과 외력에 의한 변형의 특성을 같이 갖는 변형으로 다음과 같은 변환을 생각할 수 있다.

$$\vec{x'} = \vec{x} + \vec{V}(\vec{x}) + \sum_{k_b \le i \le k_e} \delta^{(i-k_b)} \vec{d}_{m_i} \qquad (16)$$

변형 벡터장 $\vec{V}(\vec{x})$만을 추가한 것이므로 (15)에 의한 변형과 작은 스케일에서의 특성은 같음을 볼 수 있다. [그림 13][그림 14]에 $\delta = 0.5$인 경우와 $\delta = 0.65$인 경우를 각각 함께 나타내었는데 δ가 작을수록 변화의 폭이 작음을 볼 수 있다.

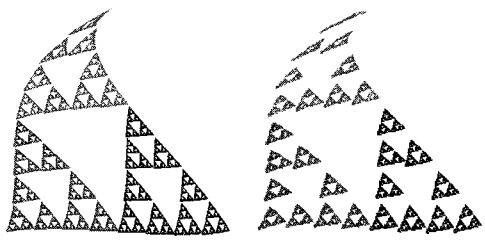


그림 13. $k_b = 3$, $k_e = 5$ **공통**

그림 14. $k_b=3$, $k_e=5$ 공통

5. 코드를 이용한 부분별 프랙탈 변형

앞에서 코드 변환 방법과 코드에 변위 벡터를 대응시키는 방법으로 프랙탈적 특성의 변형을 만들 수 있음을 보았다. 그러나 프랙탈이 고사리와 같은 자연물을 나타내는 경우는 가지가 줄기에서 떨어지는 등의 부자연스러움이 나타나는데 이는 앞의 변형 방법에서는 가지와 줄기의 접합 부분을 특별히 취급할 수 없었기 때문이다. 그러나 고사리의 각 가지부분을 코드에 의해 구분할 수 있음을 활용하면 좀 더 자연스러운 변형을 만들어 낼 수 있다.

먼저 고사리의 각 가지 부분이 코드와 어떻게 대응지를 보면 [그림 15]와 같다.

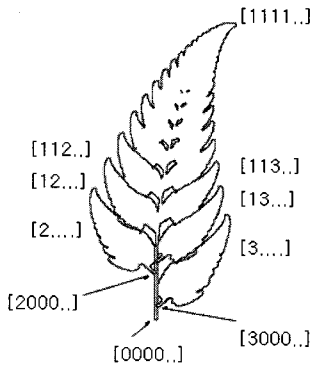


그림 15. 고사리의 점과 코드

예를 들어 왼쪽 아래의 가지에 속하는 점들의 코드는 $m_1=2$ 로 고정된 $[2, m_2, m_3, ...]$와 같은 형태를 띠며 오른쪽 아래의 가지에 속하는 점들의 코드는$m_1=3$ 으로 고정된 $[3, m_2, m_3, ...]$와 같은 형태를 띤다.(고사리의 IFS사상의 번호를 정하는 것에 따라 코드 값은 달라질 수 있다.) 또한 그 중 가지와 줄기가 접하는 부분은 코드가 각각 $[2, 0, 0, 0, ...]$와 $[3, 0, 0, 0, ...]$인 점에 해당 된다. 따라서 만약 왼쪽 아래의 가지만 변형하되 줄기에 접합된 부분은 움직이지 않도록 하려면 $[2, m_2, m_3, ...]$의 형태를 가진 코드의 점들만 이동시키되, 그 중 $[2, 0, 0, 0, ...]$의 코드를 갖는 점은 움직이지 않도록 하면 된다. 즉

$$[2, m_2, m_3, ...] \Rightarrow \vec{x} \quad (\forall m_2, m_3, ...) \tag{17}$$

로 정의되는 점 $\vec{x}$ 및 그 중

$$[2,0,0,...] \Rightarrow \vec{x_f} \tag{18}$$

로 정의 되는 $\vec{x_f}$에 대하여

$$\vec{x'} = \vec{X}(\vec{x}) \quad 단, \ \vec{x_f} = \vec{X}(\vec{x_f}) \tag{19}$$

로 변형을 정의하면 된다. 여기서 함수 $\vec{X}(\vec{x})$는 원하는 변형의 모양에 따라 적절히 설정하면 되는데 아래의 예에서는 간단히

$$\vec{x'} = \vec{x_f} + A \cdot (\vec{x} - \vec{x_f}) \tag{20}$$

와 같은 선형의 함수를 사용하였다. 여기서 A는 2×2 행렬을 나타낸다.

[그림 16]은 (20)을 사용하여 고사리에 부분적 변형을 가한 예이다. 왼쪽의 그림은 코드가 $[2, ...]$인 부분과 $[3, ...]$인 부분 즉 고사리에서 왼쪽의 첫 번째 가지와 오른쪽의 두 번째 가지에 각각 회전 행렬을 적용한 결과이고, 오른쪽의 그림은 코드가 $[1,1,3, ...]$인 부분 즉 고사리의 오른쪽의 세 번째 가지에 확대시키는 행렬을 적용한 결과이다. 가지와 줄기의 접합부분이 떨어지지 않아 보다 자연스러운 변형이 되고 있음을 볼 수 있다.

그림 16. 두 부분을 회전, 한 부분을 확대

또한, 코드를 지정하는 것에 따라 더 작은 부분에만 변형을 가하는 것과 여러 부분에 변형을 가하는 것도 얼마든지 가능하며, 고사리 외의 다른 프랙탈에 적용하는 것도 가능하다. 다만 자연물을 나타내는 프랙탈의 경우, 요구되는 자연스러운 변형을 구현하기 위해서는 실제 자연물에서 특징적인 어떤 부분이 코드의 형태를 통해 손쉽게 지정될 수 있어야 할 것이며, 여러 부분을 각기 다르게 변형하기 위해서는 각각의 부분마다 변형 함수 $\vec{X}(\vec{x})$ 를 설정해야 한다.

한편, [그림 16]은 고사리의 가지에 회전 변형과 같은 연속 변형을 가하였기 때문에 그림에서 가지를 오려내어 회전시키는 것과 비슷하게 보인다. 그러나 프랙탈의 부분끼리 공간적으로 서로 겹치는 경우도 코드 상으로 구별하여 서로 다른 변형을 가할 수 있다는 점에서 차이가 있다. 또한 변형 함수 $\vec{X}(\vec{x})$ 에 앞에서 본 코드를 활용한 변형 방법을 적용하여 부분별 변형이 프랙탈적 특성의 변형이 되게 할 수도 있을 것이다.

IV. 결론 및 과제

이상과 같이 보통의 연속적 변형과 대략 비슷한 모습을 가지며 또한 프랙탈적 특성을 갖는 변형 방법을 점의 코드를 활용하여 구현하였다.

주어진 변형 벡터장에 대하여 점의 이동이 코드 변환된 점에서의 벡터장 값에 따라 이루어지는 방법을 구현해 본 결과 코드변환의 방법에 따라 다양한 변형 모습이 나타났는데, 후지모토 등의 코드 위치 변경 방법이 아닌 코드 값의 변경에 의한 변환도 프랙탈적 특성을 나타내어 프랙탈의 변형 방법으로 적합함을 볼 수 있다.

또한 점의 코드를 활용한 다른 변형으로 코드에 직접적으로 이동 벡터를 대응시켜 변형하는 방법을 구현하였는데, 보다 기하학적 규칙성을 띤 변형에 적합한 방법임을 알 수 있었다.

한편 코드가 프랙탈 위에서의 주소와 같다는 것을 활용하면 프랙탈의 일부분에만 특정 변형을 가하는 방법을 구현할 수 있는데, 이를 고사리와 같은 자연물에 적용한 경우 보다 자연스러운 변형 결과를 얻을 수 있었다.

앞으로 프랙탈에서 코드가 가지는 주소의 의미를 활용한 다른 변형 방법들과 이들을 적절히 중첩시키는 방법들 등을 더 연구해 볼 필요가 있다고 생각된다.

참 고 문 헌

[1] M. F. Barnsley, *Fractals Everywhere, 2nd ed.*, Academic Press, Boston, 1993.

[2] R. L. Bowman, "Fractals Metamorphosis: a Brief Student Tutorial," Computers & graphics, Vol.19, No.1, pp.157-164, 1995.

[3] B. Burch and J. C. Hart, "Linear Fractal Shape Interpolation," Graphics Interface '97, pp.155-162, 1997.

[4] J. Gomes, L. Darsa, B. Costa, and L. Velho, *Warping and Morphing of Graphical Objects*, Morgan Kaufmann, 1999.

[5] J. A. Gonzalez, "A Tutorial and Recipe for Moving Fractal Trees," Computers & Graphics, Vol.22, No.2-3, pp.301-305, 1998.

[6] M. E. Montiel, A. S. Aguado, and E. J. Zaluska, "Topology in Fractals," Chaos, Solitions and Fractals, Vol.7, No.8, pp.1187-1207, 1996.

[7] M. Peruggia, *Discrete Iterated Function Systems*, A K peters, 1993.

[8] T. Sederberg and S. Parry, "Free-form Deformation of Solid Geometric Models," Computer graphics (SIG-GRAPH '86), Vol.20, No.4, pp.151-160, 1986.

[9] C. E. Zair and E. Tosan, "Fractal Modeling Using Free Form Techniques," EUROGRAPHICS '96, Vol.15, No3, pp.269-278, 1996.

[10] C. E. Zair and E. Tosan, "Unified IFS-based Model to Generate Smooth or Fractal Forms, Surface Fitting and Multiresolution Methods," Vanderbilt University Press, pp.345-354, 1997.

[11] C. E. Zair and E. Tosan, "Computer Aided

Geometric Design with IFS Techniques," Fractal Frontiers (Pro Fractals '97), pp.443-452, 1997.

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

[13] T. Fujimoto, Y. Ohno, K. Muraoka, and N. Chiba, "Fractal Deformation Using Displacement Vectors Based on Extended Iterated Shuffle Tranformation," 藝術科學會論文誌(일본) Vol.1, No.3 pp.134-146, 2002.

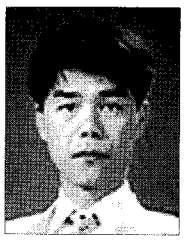
저 자 소 개

한 영 덕(Yeong-Deok Han) 정회원

- 1989년 2월 : 한국과학기술원 물리학과 (이학석사)
- 1993년 2월 : 한국과학기술원 물리학과 (이학박사)
- 1994년 3월 ~ 현재 : 우석대학교 게임콘텐츠학과 교수

<관심분야> : 양자 정보, 게임물리

김 기 옥(Gi-Ok Kim) 정회원



- 1983년 2월 : 한국과학기술원 물리학과 (이학석사)
- 1987년 8월 : 한국과학기술원 물리학과 (이학박사)
- 1987년 9월 ~ 현재 : 우석대학교 게임콘텐츠학과 교수

<관심분야> : 컴퓨터 시뮬레이션