한국게임학회 논문지
제5권 제1호 2005년 3월

게임환경에서 이용 가능한 프랙탈 오브젝트 변형에 관한 연구

*송행숙, **한영덕
*한일장신대학교 컴퓨터공학, **우석대학교 게임멀티미디어학과
*songhs@mm.hanil.ac.kr, **ydhan@woosuk.ac.kr

A Study of Fractal Object Deformation for Game Environment

*Hang-Sook Song, **Young-Duk Han
*Dept. of Computer Engineering, Hanil University,
**Dept. of Game& Multimedia, Woosuk University

요약

게임환경에서 자연물의 표현에 프랙탈을 이용한 경우, 주변의 힘 등에 의한 자연스러운 변형이 구현되어야 보다 사실감을 느낄 수 있다. 일반적인 연속적 변형의 특징과 프랙탈적 특징을 모두 갖는 변형을 코드변환을 도입하여 생성하는 다양한 방법을 예시하였다.

Abstract

To get the realistic deformation of fractal objects in computer games, fractal-like deformation property should be added to ordinary continuous deformations.
We, here, suggest a fractal-like vector fields producing method using the code and its modification, and some examples are given.
fractal, deformation, code, vector field

Key Words: fractal, deformation, code, vector field

1. 서론

게임이나. 온라인 채팅에서 사용된 그래픽 기술들은 물체나 배경의 사실감있는 표현에 주안점을 두고 있다. 그런데, 3차원 게임 환경 등을 보면 캐릭터의 움직임은 사실적이며 자연스럽게 표현되고 있으나 이에 비해 캐릭터의 움직임에 따른 배경 및 주변 물체의 반응은 표현되지 않거나 캐릭터와 독립적으로 표현되고 있어 사실감이 떨어지는 요인이 되고 있다. 즉 3차원 공간을 보여주는 데는 큰 문제점이 없으나, 사용자들은 3차원 공간에서 사실감을 느끼지 못한다. 이는 3차원 공간에서의 배경 및 주변 물체들의 반응이 현실에서와 다르게 나타나기 때문이다. 현실세계에서는 자신이 움직이거나 어떤 행위를 취했을 때, 주변의 물체가 그에 따라 적절히 반응을 해준다. 이 때문에 자신의 주변의 물체들이 실재하는 물체이며 살아 있는 듯이 느끼게 되는 것이다. 따라서 사실감을 증가시키기 위해서는 이동 물체의 행동에 따라 그 행동의 여파가 주변에 영향을 주고, 주변 물체들은 이 영향에 따라 반응하는 것이 표현되어야 한다.

사실감을 위하여 고려해야 할 또 하나의 요소는 삼각형이나 구형과 같은 기하학적 도형을 바탕으로 하여 물체를 나타내는 것이 해안선, 구름, 지면 등과 같은 자연의 물체를 표현하는데 한계가 있다는 것이다. 기하학적 도형을 사용하는 경우는 부자연스러울 뿐만 아니라 세밀한

표현을 위해서는 많은 양의 메모리가 필요해지는 등의 단점이 있다. 이러한 자연물들을 나타내는 수학적 모델은 프랙탈임이 잘 알려져 있다[1]. 프랙탈을 이용하면 게임에서

이용되는 자연물들인 구름, 나뭇가지, 지형, 해안선 등에 대해 사실에 근접한 묘사를 할 수 있으며. 간단한 구조로서 메모리를 절약하여 수행속도를 높일 수 있는 장점이 있다.

즉 게임환경에서 자연물의 사실감있는 표현을 위해서는 단순한 기하학적 도형대신 프랙탈을 이용하여 물체를 생성?표현해야 할 필요가 있고 또한 가해지는 변형력이나 충돌 등의 상황에 따라 그 프랙탈 도형을 적절하게 변형해 주어야 할 필요가 있다.

그런데 프랙탈 도형의 변형방법이 프랙탈이 그려진 바탕 물체를 구부리거나 잡아늘리는 것인 경우 실제 자연물의 움직임과 달라보여 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 다르게 움직이기도 하는 것이다.

본 연구에서는 게임 배경 등에 사용되는 프랙탈을 그에 가해진 변형 벡터장에 따라 변형하되 프랙탈의 특성을 살림으로써 보다 사실감있게 보이도록 하는 변형 방법을 고려하였다.

논문의 구성은 2절에서는 관련연구를 간단히 살펴보고, 3절에서는 프랙탈 변형을 도입하고 적용 예를 보며, 4절은 결론 및 향후 연구 과제로 하였다.

2. 관련연구

물체의 모양변형은 물체의 모양을 만들거나 수정 할 때 유용한 기술로 다양한 방법들이 제안되어 왔다. 예를 들면 ,Free Form Deformation(FFD)[4,8]는 3D물체의 모양을 물체가 놓여있는 공간의 3D격자를 변화시켜 변형하는 것으로 다른 모양의 두 물체 사이에 자연스러운 연속적 변화를 만들어 내는 방법으로 많이 이용되고 있다. 2D, 3D 모핑(Morping) 및 구부림(Warping)[4]은 단일 물체 내에서 변화하는 것으로 컴퓨터 그래픽스 분야에서 많이 사용되는 기술이다. 여기에 또 다른 방법으로 프랙탈 변형 방법이 많은 시도되어 왔는데[2,3,5,~7,9,~11] 연속적 변형기술인 FFD와 완전히 다른 변형의 스타일을 만들어 내는 기술이라고 할 수 있다. 프랙탈 변형은 전체모양을 연속적으로 변형시키는 연속적 변형과는 달리 모양의 일부 부분이 전체 규모 내에서 반복적으로 나타나는 프랙탈의 특성에 알맞는 변형기술이어야 한다.

프랙탈 변형기술은 많은 연구자들이 지속적으로 연구해오고 있는 분야인데 대부분 축소사상(contraction map) 의 IFS (Iterated Function System)에 의해 정의되는 프랙탈을 바탕으로 하고 있다. 이 경우, 프랙탈이 IFS 의 attractor로 정의되므로[1] 축소 사상들을 변화시키면 변화된 IFS attractor 즉 변화된 프랙탈을 쉽게 만들어 낼 수 있다. [2]에서 제안된 방법은 각 축소사상의 고정점과 각 축소 사상의 끌어당기는 강도를 변화시켜 변형하는 것이다. 또한 [2,3,7]에서는 IFS attractor의 개념을 활용하고 그의 전체적인 모양을 변형시키고 있다. 그러나 이러한 기술들은 IFS의 축소 사상의 변화가 프랙탈 전체 모양에 영향을 주고, 고정점들의 끌어당기는 강도나 축소 사상의 계수 같은 매개변수 값들로 조절되기 때문에 사용자는 모양 변화에 위치에 따른 부분적인 제어를 할 수 없게 된다. 또한 일부 모양들에서는 변형을 직관적으로 예측하는 것이 어렵게 되기도 한다.

프랙탈도 하나의 사실적 물체라고 볼 때, 바람직한 변형방법이 갖추어야할 요소 중 하나는 전체적인 모양의 변화는 일반적인 물체의 변형과 같이 물체에 가해진 힘의 방향에 따라 일어나야 한다는 것이다. 즉 물체가 받는 힘을 나타내는 벡터장에 따라 그에 비례하는 변형이 일어나야 한다. 두 번째는 프랙탈의 특성을 나타내는 변형방법이어야 한다는 것이다. 즉 단순히 프랙탈이 그려진 고무판이 늘어나는 방식으로 변형되지 않고 자기 반복적인 프랙탈의 특성이 변형방법에도 반영되어야 한다.

최근 후지모토 등에 의하여 프랙탈 위의 점에 코드를 부여한 뒤 이 코드를 변화시켜 얻은 새로운 점을 이용해 프랙탈적 특성을 갖는 변형을 얻어내는 방법이 제시되었다[12,13]. 본 논문에서는 코드변화에 따른 점의 이동이 프랙탈적 특성을 갖는 위치 변동이며 이를 변형 벡터장과 결합하여 활용하면 프랙탈적인 변형방법을 얻을 수 있다는 점에 주목하여 간단하면서도 다양한 형태변화를 표현할 수 있는 몇가지 코드변화 방법을 제시하였고 또한 그 적용 결과를 나타내었다.

3. 코드 변환을 이용한 프랙탈 변형

프랙탈 도형을 생성하는 방법은 여러 가지가 있을 수 있으나, 여기서는 그 중 IFS를 통한 프랙탈과 그 변형 방법을 고려해 본다.

공간 위에서 정의되고 두 점간의 거리를 줄여주는 n개의 축소사상을 $F_i, (i=0,1,\ldots,n-1)$라 하자. 이 함수들(Iterated Function System)에 대한 끌개(attractor)는 공간의 점들의 집합 S로 다음과 같은 식으로 정의된다.

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

S는 일반적으로 프랙탈 도형이 됨이 알려져 있다[1]. 예를 들어

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

와 같이 3개의 축소사상의 경우 이에 해당하는 프랙탈 도형은 〈그림 1〉과 같은 시에르핀스키(Sierpinski) 개스킷이 된다. 함수의 contraction 성질을 활용하여 다음과 같이 간단한 알고리즘으로 해당 프랙탈을 그릴 수 있다.

즉 1. 먼저 임의의 점 (x_0, y_0)를 잡는다.

2. F_i 중 임의로 하나(F_{n_i})를 골라 (x_0, y_0)에 가하여 (x_i, y_i)을 얻는다.

3. 마찬가지로 (x_{i-1}, y_{i-1})에 F_{n_i}를 가하여(x_i, y_i)를 얻는 과정을 무한히 반복한다. 이 때 충분히 큰 N에 대해서 $i > N$ 일 때부터 유효한 점으로 보고 평면에 표시하게 되면 프랙탈 도형을 얻게 된다.

위와 같은 방법으로 얻어진 프랙탈 위의 점의 위치는 최초 점 (x_0, y_0) 및 적용한 사상의 수열 $k_1, k_2, \ldots$ 에 의하여 결정된다. 그런데 사상의 축소되는 성질 때문에 최근에 작용한 사상은 점의 위치에 큰 영향을 주는 반면 작용한지 오래된 사상은 점의 위치에 작은 영향을 줌을 볼 수 있다. 따라서 점의 위치를 나타내는데 (x_0, y_0)와 작용한지 오래된 사상은 무시할 수 있다. 이러한 성질에 따라 프랙탈 위의 한 점은 그 점에 도달케 한 최근의 사상의 번호부터 역으로 쓴 $(m_1, m_2, \ldots)$로 나타낼 수 있음을 알 수 있다. 단 위와 같은 코드화에 있어서는 프랙탈 위의 임의의 점 $\vec{x}$에 대응되는 코드가 2개 이상일 수 있다. 왜냐하면 다른 순서로 map을 가한 결과가 같은 점이 될 수 있기 때문이다. 그러나 주어진 코드로부터 정해지는 점 $\vec{x}$는 유일하다.

$$(m_1, m_2, \ldots) \rightarrow \vec{x} \tag{2}$$

코드에 점이 어떠한 방식으로 대응되는가를 모든 IFS 에 대해 일률적으로 규정하기는 어렵지만, 대체적으로 m_1이 가장 큰 주소를 나타내고 m_2는 그 안에서의 세부주소를 나타내는 것으로 볼 수 있다. 따라서 주어진 코드의 일부를 특정 방식에 따라 바꾸면 프랙탈 내에서 비슷한 다른 점을 나타내게 된다. 이 때 점의 위치 변화는 불연속적이다. 이러한 점의 이동 방식의 중요한 특징은 일반적인 연속성 있는 이동과 달리 프랙탈적 특성을 띤 이동방법이라는 점이다. 예를 들어 시에르핀스키 개스킷에서 코드의 첫째 자리와 5번째 자리를 교환하는 코드 변환 하에서, (1111201...)과 (1111001...)은 각각 새로운 점 (2111101...)과 (0111101...)로 이동하는데 이동 후에는 두 점의 거리가 이동 전보다 매우 커짐을 볼 수 있다. 또한 6번째 이후는 바뀌지 않으므로 작은 스케일에서는 점의 프랙탈 내의 위치에 변화가 없다. 이러한 특성을 활용하면 단순한 연속적 변환과 달리 프랙탈적 특성을 띤 변형을 만들 수 있고 보다 자연스럽게 프랙탈 물체의 변형을 만들어낼 수 있다.

게임 배경 등에서 사용하는 프랙탈 물체가 좀 더 자연스럽게 보이기 위해서는 게임 내의 상황에 따라 받는 힘이나 충돌 등에 반응하여 변형될 필요가 있다. 일반적으로 물체의 변형은 그 물체가 속한 공간을 벡터장을 통해 변형함으로써 구현할 수 있다. 그러나 단순히 공간의 변형에 의하여 그 위에 놓인 프랙탈이 변형되는 방식은 프랙탈적인 특성을 충분히 고려한 변형이라 할 수 없고 따라서 그 변형이 부자연스럽게 보일 수 있다. 이는 프랙탈 위의 점을 단순히 공간에 놓인 점들로 간주하고 변형시켰기 때문이다. 앞서 언급한 바와 같이 바람에 의해 움직여진 나뭇가지의 예라면 점이 어느 가지에 속했는가에 따라 서로 다른 이동량을 가지게 해야 자연스럽게 보일 것이다. 즉 공간좌표값에 따른 점의 위치보다 어느 가지에 속했는가가 프랙탈적 특성의

위치정보라 할 수 있다. 따라서 점을 나타내는 좌표 뿐만 아니라 점의 코드에 따라서도 점의 이동량이 달라지는 방식이어야 프랙탈적 특성을 고려한 변형이 될 것이다.

점의 코드를 이용하여 프랙탈적 특성의 변형을 얻는 방법은 여러 가지가 가능하다. 일반적인 물체 변형 벡터장이 프랙탈 도형에 주어졌을 때, 이를 바탕으로한 프랙탈적 특성을 갖는 변형의 몇가지 예들을 아래에 제시하였다.

3.1 연속 변환에 따른 변형

이는 공간의 변형을 나타내는 벡터장에 따라 그 위에 놓인 프랙탈도 같이 변형되는 방법으로 벡터장을 $\overrightarrow{V(x)}$ 라 하면 $\vec{x}$ 의 이동점 $\vec{x'}$ 는 다음과 같다.

$$\vec{x'} = \vec{x} + \overrightarrow{V(x)} \quad (3)$$

〈그림1〉을 적당한 벡터장을 잡아 이 방법으로 변형한 것이 〈그림2〉에 나타나 있다. 작은 스케일에서의 프랙탈 모양은 축소 · 확대되거나 비틀려 있지만 연결관계는 동일하다.

3.2 코드변환을 이용한 프랙탈 변형

작은 스케일에서의 연결관계에까지 영향을 미치면서 큰 스케일에서는 〈그림2〉와 비슷한 변형을 얻기위해 코드변환을 이용해 다른 점의 벡터장을 차용해 점을 이동시키는 방법을 생각하자. 즉 점 $\vec{x}$ 의 코드가 $(m_1, m_2, \ldots)$일 때, 이를 변환한 코드 $(m'_1, m'_2, \ldots)$이 나타내는 점 $\vec{x'}$ 를 이용해 $\overrightarrow{V(x')}$ 를 얻고, 이로부터

$$\vec{x''} = \vec{x} + \overrightarrow{V(x')} \quad (4)$$

와 같이 이동점 $\vec{x''}$ 을 얻는 방법이다. 이 경우, 변환 방법을 적당히 택하면 큰 스케일에서는 1)과 비슷하나 작은 스케일에서는 프랙탈적 특성을 갖는 변형을 구현할 수 있다. 이 때, 코드의 변환에는 여러 가지 방법이 가능하며 각각은 다른 변형을 생성할 것이다. 다음은 여러 가지 방식의 코드변환과 함께 (4)를 사용하여 변형을 한 예들이다.

1) 여기서는 코드의 일부구간을 특정 값으로 대치하는 변환을 $(m_i = \ldots = m_j = 0)$ 사용하였다. 즉 특정 스케일 단계에서 세부적 위치를 무시하고 벡터를 얻어 점의 위치 이동에 사용한 경우이다. 그 변형된 결과를 〈그림3-1,2〉에 나타내었다.

2) 여기서는 코드 내의 두 정수를 서로 교환하는 방법 $(\ldots m_i \ldots m_j \ldots) \leftrightarrow (\ldots m_j \ldots m_i \ldots)$ 을 사용하였다. 그 변형된 결과를 〈그림4-1,2〉에 나타내었다.

3) 여기서는 코드 내의 일부 구간을 제거하고 뒷부분을 앞으로 당기는 변환을 사용하였다.
$T_{ij} : (\ldots m_{i-1} m_i \ldots m_j m_{j+1} \ldots) \rightarrow (\ldots m_{i-1} m_{j+1} \ldots)$ 이는 벡터장의 모양을 축소하여 프랙탈의 각 부분에 적용하는 효과를 준다. 그 변형된 결과를 〈그림5-1,2〉에 나타내었다.

4) 이것은 3)에 의한 변환들을 중첩시킨 것인데 단 작은 스케일이 되면 벡터장도 그에 따라 크기를 줄이도록 하여 지나치게 점이 이동하는 것을 막았다. 즉

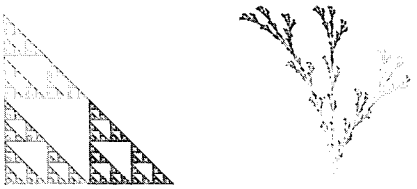
$$\vec{x''} = \overrightarrow{V(x)} + \sum_{k=0}^{j} \delta^{(k+1)} \overrightarrow{V(T_{0k}(x))}, \quad 0 \le \delta \le 1$$

와 같이 이동점을 얻는 방법이다. 변형된 결과를 〈그림6-1,2〉에 나타내었다.

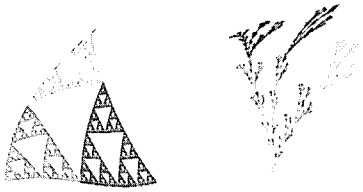
4. 결론 및 향후 연구과제

이상과 같이 보통의 연속적 변환과 비슷하면서 프랙탈적 특성을 갖는 변형 방법을 코드 변환과 변환된 점의 벡터값의 차용을 통해 구현해 보았다. 그 결과 코드변환의 방법에 따라 다양한 결과가 나타남을 볼 수 있었고 또 중첩을 시킴으로써 여러 가지 특징을 혼합할 수도 있음을 볼 수 있었다. 앞으로 보다 다양한 프랙탈에 대하여 이와 같은 변형 방법을 적용해봄으로써 더욱 자연스러운 변형 그리고 상황에 알맞은 변형 방법을 개발할 필요가 있다고 생각된다.

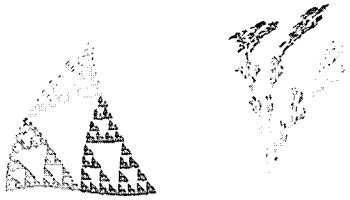
나아가 3차원 자연물의 모델링을 프랙탈을 이용하는 것과 주변 물체에 의한 변형력 및 그에 대한 반응을 구현하는 것도 게임의 사실감 고조를 위해 연구해볼 필요가 있다고 생각된다. 또한 계산 시간을 단축하기 위한 알고리즘과 게임엔진에 활용될 수 있는 API 의 개발 등도 향 후 필요하리라 생각된다.



〈그림 1〉

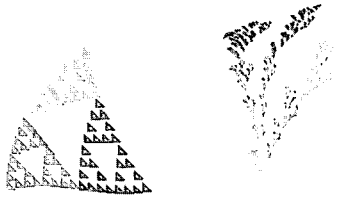


〈그림 2〉



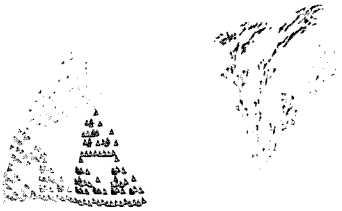
i=4, j=4 i=4, j=4

〈그림 3-1〉



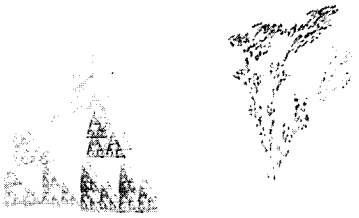
i=4, j=7 i=4, j=7

〈그림 3-2〉



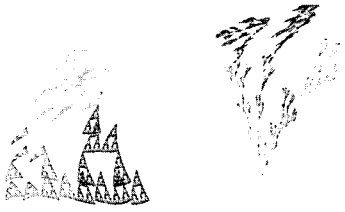
i=2, j=4 i=2, j=3

〈그림 4-1〉



i=2, j=7 i=2, j=4

〈그림 4-2〉

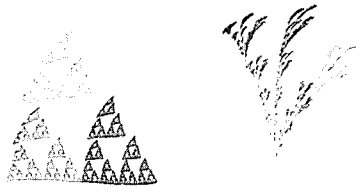


i=2, j=2 i=3, j=3

〈그림 5-1〉

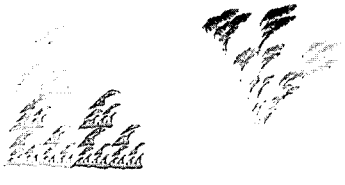
i=3, j=5 i=3, j=5

〈그림 5-2〉



j=3, δ=0.5 j=3, δ=0.5

〈그림 6-1〉



j=3, δ=0.8 j=3, δ=0.8

〈그림 6-2〉

참고 문헌

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

[2] Bowman, R.,L., Fractals Metamorphosis: a Brief Student Tutorial, Computers&graphics, Vol 9, No1, pp.157-164, 1995.

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

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

[5] Gonzalez, J., A., A Tutorial and recipe for moving fractal Trees, Computers & Graphics, Vol.22, No.2-3, pp.301-305,1998

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

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

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

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

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

[11] Zair, C. E. and, Tosan, E.,Computer Aided Geometric Design with IFS Techniques, Fractal Frontiers (Pro Fractals '97)

[12] Fujimoto, T. and Ohno, Y.,Muraoka, K., and Chiba, Fractal Deformation Based on Extended Iterated Shffle Tranformation, NICIGTAPH Inernational 2002, pp.79-84. 2002.

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

한 영 덕

1993. 9 ~ 1994. 2. 한국과학기술원 자연과학연구소 연구원
1994. 3 ~ 현재 우석대학교 게임멀티미디어학과 교수
관심분야 : 컴퓨터그래픽스, 카오스, 양자정보이론, 양자컴퓨터

송 행 숙

1997년 ~ 현재 한일장신대학교 컴퓨터정보통신학부 교수
2000년 ~ 현재 한일장신대학교 컴퓨터정보통신학부 학부장
2003. 3 ~ 2004. 9 한일장신대학교 정보문헌관장
1994 ~ 1996 전북대학교,아주대학교 컴퓨터공학과 강사
1994 ~ 1995 (주)시멘틱스 연구소 소장
2004년 ~ 현재 한국게임과학고등학교 이사
관심분야 : 컴퓨터그래픽스, HCI, 멀티미디어정보처리, 유비쿼터스응용