노이즈 편집을 이용한 그래픽스 객체 편집
(Editing Graphical Objects using Noise Editing)

윤 종 철 † 이 인 권 †† 최 정 주 †††
(Jong-Chul Yoon) (In-Kwon Lee) (Jung-Ju Choi)

요 약 노이즈 함수는 난수의 추가를 통해 실제 자연현상에 대한 애니메이션을 더욱 더 실감나게 하는 방법으로 쓰여 왔다. 본 연구에서는 노이즈의 통계적 특징은 보존하면서, 사용자가 원하는 제약에 맞는 결과 값을 산출하는 방법을 제시한다. 이상적 형태의 통계 값과, 실제 노이즈 함수의 편향치의 통계 값의 차이를 최소화 하는 최적화 기법을 통해 노이즈의 성질을 유지하면서 손쉽게 제어를 할 수 있는 방법을 제안하고자 한다. 이 방법을 통해 노이즈가 적용된 애니메이션이나 모양에 대한 세부적인 제어가 가능하다.

키워드 : 노이즈, perlin 노이즈, 애니메이션 제어, 점진적 텍스쳐

Abstract Noise is used to create realistic animations that look like natural phenomena as well as procedural textures and shapes by adding randomness to graphical applications. In this paper, we suggest a method to edit noise values to satisfy the constraints that reflect the user's demands while maintaining the inherent statistical features of the noise function. Noise editing uses optimization to minimize the difference between the statistical characteristics of the ideal and edited versions of a noise source. Using our editing method, detailed control of animation and shape data that include noise is possible.

Key words : noise, perlin noise, animation control, procedural texture

1. 서 론

Perlin[1]이 노이즈 함수를 소개한 이래로, 노이즈는 점진적 텍스처(Procedural Texture)의 생성뿐 아니라 자연스러운 애니메이션의 제작 또는 인위적인 형태를 자연스럽게 만드는 방법으로 쓰여 왔다. 컴퓨터 그래픽스에서 정의하는 노이즈 함수란, 난수의 일반적 개념인 제약이 없는 난수, 즉 화이트 노이즈와는 다른 개념이다. 컴퓨터 그래픽스에서 말하는 이상적인 노이즈 함수의 성질은 재사용이 가능하면서도, 비 주기적인 성질을 만족해야 한다[2].

본 논문에선 노이즈 함수의 결과 값을 제어하는 문제를 고려한다. 즉, 노이즈의 다양한 적용에 있어, 사용자의 요구에 따르는 노이즈 함수를 만드는 방법을 제시한다. 예를 들어 주어진 애니메이션이 점진적 형태로 만들어졌을 때, 보다 자연스러움을 주기 위해 단일 노이즈 함수 또는 노이즈 함수의 합(Fractal Sum)이 더해진다[2-5]. 하지만 난수가 적용된 이후에는 정확한 형태를 예상할 수 없기 때문에, 사용자가 원하는 다른 제약을 정확히 만족하기는 힘들다. 만약 물리 시뮬레이션과 같이 계산이 복잡할 경우, 제약을 위해 애니메이션 패스를 처음부터 다시 계산하는 것은 시간의 낭비를 가져온다. 심지어 패스를 다시 계산한다고 해도 그것이 제약을 만족하리란 보장은 없다. 따라서 이런 경우, 바탕이 되는 패스의 제어가 아닌 추가되는 난수에 대한 제어, 즉 노이즈를 지역적으로 제어할 수 있는 방법이 필요하다.

만약 원하는 특정 결과값을 위해, 노이즈함수의 근원인 편향치에 대한 인위적인 변이가 있었을 때는, 앞에서 말한 이상적인 노이즈의 특성들이 유지되기 힘들다. 왜냐하면 편향치에 대한 인위적인 변이는 난수의 분포가 한쪽으로 치우치는 결과를 가져올 수 있기 때문이다. 예를 들어 그림 1(b)는 편향치의 인위적 변이 때문에 노이즈 값의 분포가 편향된 결과를 보여준다. 따라서 노이즈의 통계적 분포를 유지하면서 난수를 제어할 수 있는

· 본 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2004-8-0014)
† 학생회원 : 연세대학교 컴퓨터과학과
media19@cs.yonsei.ac.kr
†† 정 회 원 : 연세대학교 컴퓨터과학과 교수
iklee@yonsei.ac.kr
††† 종신회원 : 아주대학교 미디어학부 교수
jungju@ajou.ac.kr
논문접수 : 2005년 3월 10일
심사완료 : 2005년 9월 20일

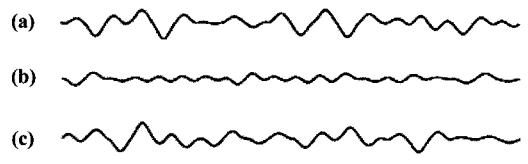


그림 1 다른 난수 분포를 가진 일 차원 노이즈: (a) 등한 난수분포를 가진 노이즈 함수, (b) 기울어진 분포를 가진 노이즈 함수(편향치가 2/3은 양수 1/3은 음수), (c) 본 논문의 제어 방법을 통해 500번의 제어가 적용된 후 결과

방법이 필요하다.

이번 연구에서 우리는 노이즈의 지역적 제어를 위해, 난수값의 근원이 되는 편향치를 제어해서 원하는 형태를 가지는 노이즈를 구성하는 방법을 제시한다. 이를 위해 카이-스퀘어(Chi-square goodness of fit test) 통계치를 사용하여 분포의 차를 최소화 하는 최적화 기법을 구성하였다. 카이-스퀘어 통계치는 이상적인 편향치 벡터의 분포와 제어된 편향치벡터의 분포의 차이를 측정하는데 쓰인다. 이 방법을 통해 원래 노이즈 함수의 통계적 난수 패턴을 최대한 유지한 체, 주어진 제약을 만족할 수 있는 방법이 가능해졌다(그림 1 참조). 비록 이번 연구에 있어 효과적인 예시를 위해 편향치 노이즈 알고리즘만을 사용했지만, 노이즈 제어 기술은 난수에 기반을 둔 다른 노이즈 알고리즘에도 적용이 가능하다[2].

본 논문의 구성은 다음과 같다. 먼저 2절에서 노이즈 함수의 연구와 활용에 관련연구를 서술한 후, 3절에서 Perlin의 편향치 노이즈 함수와 카이-스퀘어 테스트 기법에 대한 소개를 하겠다. 4장과 5장에선 제약된 노이즈를 정의하기 위한 최적화 문제에 대한 소개와 그것에 대한 확장을 논의하겠다. 6장에선 본 논문의 노이즈 제어 문제가 어떻게 적용될 수 있는지에 관한 실험적 결과를 보이고, 마지막으로 7장에서 결론을 맺는다.

2. 관련 연구

Perlin[1,6]이 노이즈 함수를 소개한 이후로, 노이즈는 점진적 텍스처의 생성[2,3,7], 물 또는 불과 같은 다양한 자연형상에 표현[4,5,8,9], 그리고 움직이는 모션을 좀 더 자연스럽게 하는 방법으로 사용되어 왔다[10]. 노이즈는 그 동안 자연현상을 시뮬레이션 하는 거의 모든 과정에서 쓰여 왔으며 때때로 복잡한 물리기반의 시뮬레이션에 대한 대안으로도 사용되었다[2,4]. 또한 최근에는 실시간 그래픽스를 위해 노이즈 알고리즘이 하드웨어로도 구현되고 있다[11]. 자연현상을 표현하는 다른 알고리즘으로 스펙트럼 합성(spectral synthesis)[12]가 제안되었지만 제어에 대한 유동성 부족으로 Perlin의 알고리즘이 컴퓨터 그래픽스에 있어 더 선호되고 있다.

그 동안 노이즈의 적용에 대한 많은 연구는 존재했었지만 그것을 제어하는 연구는 거의 존재 하지 않았다. 즉, 지금까지의 연구들은 단일 노이즈나 노이즈의 합으로 이루어지는 프랙탈 합에 대한 기본적 변수에만 초점을 맞추고 있어서, 결과의 수정에 대한 고려는 이루어지지 않았다.

최근 Perlin[13]은 기존의 노이즈 알고리즘의 두 가지 문제점을 해결하는 새로운 알고리즘을 제시하였다. 그는 모든 정의역에서 C^2 연속성을 만족하기 위한 새로운 보간법을 제시하였고, 16개의 고정된 편향치 벡터를 통한 결과값이 한쪽으로 치우치는 결과를 막음과 동시에 빠른 계산이 가능하게 하였다. 우리는 이 두 가지 개선점 중 편향치 벡터를 고정하는 방법은 채택하지 않았다. 왜냐하면 본 연구의 알고리즘 자체가 편향치 벡터의 변이를 통해 원하는 결과값을 얻는 것이기 때문이다. Ebert [2]등은 다양한 노이즈 알고리즘과, 노이즈 함수의 여러 특성을 세부적으로 설명하였다.

3. 배경(Background)

3.1 Perlin의 편향치 노이즈 알고리즘

우리가 제시할 방법은 Perlin의 편향치 노이즈 알고리즘을 기반으로 하고 있다. 편향치 노이즈 알고리즘이란, 의사난수(PRN: Pseudorandom numbers)로 만들어진 단위 편향치 벡터로 구성된 편향치 테이블 G와 PRN 정수로 채워진 테이블 P를 을 사용하여 상호 인덱싱(indexing)을 통해, 얻어진 편향치 값의 보간을 통해 결과값을 얻는다(그림 2 참조). 인덱싱을 위한 정수 격자는(integer lattice) 노이즈 함수가 정의된 차원에 의해 결정된다. 따라서 N차원 노이즈는 결과값 계산을 위해 2^N 만큼의 편향치 벡터를 필요로 한다. 아래의 개요는 3차원 좌표(x_1,x_2,x_3)가 주어졌을 때, 어떤 과정을 통해 노이즈 값이 계산되는지를 설명한 것이다.

Step 1: M 개의 3차원 PRN 단위 벡터를 구성하여 M 만큼의 크기를 가진 G[0,...,M-1] 테이블에 저장한다. 또한 같은 M 만큼의 크기를 가지는 다른 테이블 P에 0에서 M-1까지의 PRN 순열을 저장해 둔다.

Step 2: 주어진 좌표의 각각의 차원에 대한 3개의 정수 간격, 즉 $[q_0^i, q_1^i]$ (i=1,2,3)을 얻는다. 여기서 mod M이고 $q_0^i = \lfloor x_i \rfloor$ $q_0^i = q_1^i + 1$을 나타낸다. 각각의 차원에 대한 3개의 정수 간격은 주어진 좌표(x_1,x_2,x_3)를 포함하는 하나의 정사면체를 생성한다.

Step 3: Step 2에서 계산된 정사면체의 각각의 점에

대한 8개의 편향치 벡터를 다음과 같이 얻어온다: $g_{jkl} = G[P[P[P[q_1^j]+q_2^k]+q_3^l]]$. 여기서 jkl 0에서 1까지의 가능한 모든 수열을 뜻한다.

Step 4: 노이즈의 결과 값은 여덟 개의 내적 계산, 즉 $g_{jkl}\cdot[v_{jkl}-(x_1,x_2,x_3)]$의 가중치 합을 통해 계산된다. 여기서 가중치는 $s(t)=6t^5-15t^4+10t^3$로 표현되는 이즈 곡선(ease curve)공식을 통해 계산된다. 우리는 펄린(Perlin)에 의해 개선된 이즈 곡선을 사용하였다[13].

위의 과정중 step 4의 가중치 합을 구할 때 정수격자와의 차이가 내적항으로 들어가기 때문에 노이즈 함수는 항상 정수 격자에서 0을 지나는 성질을 가지게 된다.

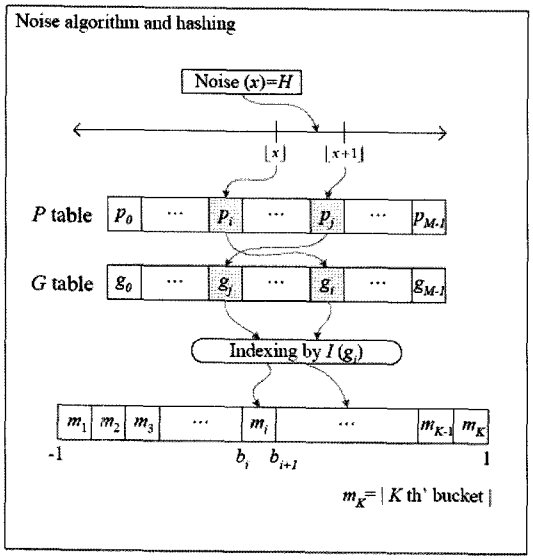


그림 2 노이즈 알고리즘과 해쉉 매커니즘

3.2 카이-스퀘어(chi-square) 통계치 테스트

카이-스퀘어 테스트[14,15]는 관찰한 데이터의 분포와 원하는 분포를 비교하는데 사용되어 왔다. $Y = \{y_1,..., y_m\}$를 관찰된 데이터의 집합이라 가정했을 때, Y 의 정의역인 $[a,b]$를 일정한 크기를 가진 K 개의 버켓(bucket)으로 나눌 수 있다. 이때 각각의 분리된 버켓을 $Y_k(k=1,...,K)$로 나타낼 수 있다. Z가 원하는 분포를 나타내고, Y 와 Z가 비슷한 분포를 가지길 원한다고 가정하면 카이-스퀘어 통계치는 아래와 같이 계산된다.

$$D^2=\sum_{k=1}^{K}\frac{(|Y_k|-|Z_k|)^2}{|Z_k|} \quad (1)$$

$|.|$이 버켓에 포함된 데이터의 양을 표현한다고 가정하면, $|Z_k|$는 Z의 분포에서 K번째 버켓에 들어갈 기대 값을 말한다. 예를 들어, 만약 Z가 균등한 난수 분포를 가졌다고 가정하면, 모든 K에 있어 $|Z_k|=M/K$이다. 식 (1)에서 D^2 값이 작아질수록 Y의 분포는 Z의 분포와 유사해 진다고 볼 수 있다. $(|Y_k|-|Z_k|)^2$를 $|Z_k|$로 나누는 것은, K번째 버켓의 기대 값이 크면 클수록 제곱차인 가 D^2 값에 주는 영향이 줄어듦을 의미한다. 버켓의 크기는 엄지손가락의 법칙(rule-of-thumb)[16]을 따라서 각각의 버켓에 최소한 5개의 데이터가 들어갈 수 있는 크기로 결정하였다. 본 연구에선 256개의 크기를 가지는 편향치 테이블을 사용하였으므로, 카이-스퀘어 테스트를 위해 50개의 버켓으로 테이블을 나누었다.

4. 최적화를 통한 노이즈의 제어

앞장의 노이즈 알고리즘에서, 편향치를 제외한 나머지 항들의 정리를 통해 노이즈 함수 "Noise(x)"는 N차원의 좌표 값 x를 통해 유도된 편향치들의 공식으로 아래와 같이 나타낼 수 있다.

$$\text{Noise}(x)=c_1g_1+c_2g_2+...+c_Jg_J, \quad (2)$$

여기서 $J = 2^N$ 이고 $g_j,(j=1,2,...,J)$는 x좌표를 통해 해쉉한 편향치 벡터 테이블 G로부터 얻어지는 값이다. 계수인 c_j는 노이즈 계산식을 통해 얻어진다. 노이즈에 제어를 가한다는 뜻은 특정 x좌표에서 유도될 노이즈의 값을 원하는 값 H로 제약함을 뜻한다(즉, Noise(x)=H). 이 경우에 주어진 제약을 위해 편향치 벡터 g_j가 바뀌어야 한다. 간단히, 위의 일차 식을 풀어 얻어진 g_j를 통해 제약을 가할 수 있다. 하지만 연속적인 제약이 노이즈에 적용될 경우, 의사난수의 분포가 편중되어 테이블이 노이즈 함수가 가져야 할 이상적인 통계적 특징을 잃어버릴 위험성이 있다(그림 1 참조). 따라서 편향치 테이블의 분포를 보존하기 위해, 편향치 테이블이 최대한 균등한 난수 분포를 가지는 방향으로 제약이 이루어져야 한다.

4.1 최적화 문제

간단한 이해를 위해 노이지 함수를 1차원으로 가정하겠다. 1차원 노이즈의 식을 편향치를 기준으로 나열하면 Noise(x)=$c_ug_u+c_vg_v$로 나타낼 수 있다. 카이-스퀘어 테스트를 위한 K개의 버켓이 존재하고, 데이터의 전체 분포가 균일한 난수 분포를 가진다고 할 때, 각 버켓의 기대값 $Z=M/K$로 정의 할 수 있다. 제약을 만족하기 위해 편향치 g_u와 g_v는 새로운 값 h_u와 h_v으로 변하게 된다. 또한, 편향치의 변화에 의해 K번째 버켓에 들어가는 데이터의 양 m_k는 모든 k에 대해 r_k로 변하게 된다. 변화된 분포에 대한 카이-스퀘어 통계치는 아래와 같은 공식으로 나타낼 수 있다.

$$D^2 = \sum_{k=1}^{K} \frac{(r_k - Z)^2}{Z} \quad (3)$$

식 (3)에서 분모의 Z는 상수 항이므로 무시할 수 있다. 위의 식을 이용하여 우리는 카이-스퀘어 통계치를 최소화 하는 아래와 같은 최적화 문제를 구성하였다.

$$\text{Minimize} \quad D^2 = \sum_{k=1}^{K} (r_k - Z)^2, \quad (4)$$

subject to

$$c_u h_u + c_v h_v = H, \quad \text{and} \quad (5)$$

$$-1 \le h_u, h_v \le 1 \quad (6)$$

식 (5)에서 주어진 제약은 노이즈 제약식을 식 (2)의 형태로 표현한 것이다. 식 (6)은 편향치의 고유한 성격인 크기에 대한 제약을 나타낸다. 하지만 우리는 위의 최적화 문제를 수정해야 할 필요성이 있다. 제약을 나타내는 식 (5)와 식 (6)은 편향치의 값에 관한 식인 반면에, 목적함수 D^2는 버켓에 들어가 있는 편향치의 숫자에 관한 식이기 때문이다.

g_u와 g_v가 원래 들어가 있던 버켓 u,v의 크기는 $[b_u,b_{u+1}]$ 과 $[b_v,b_{v+1}]$로 나타낼 수 있다. u와 v가 꼭 다른 버켓이란 가정은 없다. g_u와 g_v가 h_u와 h_v로 변이할 때 버켓 u에 들어간 편향치의 양은 m_u에서 r_u로 변할 것이다. 즉 아래와 같이 정의 할 수 있다.

$$r_u = m_u - 1 + B_u(h_u) + B_u(h_v), \quad (7)$$

여기서 $B_u(h)$는 아래와 같이 정의되는 복스카르 (Boxcar) 함수이다.

$$b_u(h) = \begin{cases} 1, & \text{if} \quad b_u \le h \le b_{u+1} \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

같은 식으로, r_v또한 편향치가 변이된 후 버켓 v에 들어갈 양이다. 지금까지의 내용을 바탕으로 목적함수인 식 (4)는 아래와 같이 다시 정의 할 수 있다.

$$\begin{aligned} D^2 = \; & \left(m_u - 1 + B_u(h_u) + B_u(h_v) - Z\right)^2 \\ & + \left(m_v - 1 + B_v(h_v) + B_v(h_v) - Z\right)^2 \\ & + \sum_{k=1,k\in\{u,v\}}^{K} \left(m_k + B_k(h_u) + B_k(h_v) - Z\right)^2 \end{aligned} \quad (9)$$

4.2 다차원 노이즈로의 확장

다차원 노이즈 함수의 계산을 위해선, 편향치 테이블에서 $J=2^N$개의 편향치 벡터인 $g_j,(j=1,...,J)$를 사용해야 한다. 즉 식 (2)의 형태로 Noise(x)=$c_1g_1+...+c_Jg_J$로 나타낼 수 있다. 따라서 원하는 노이즈값의 제어를 위해 계산되어야 할 새로운 편향치 벡터 h_j는 Noise(x)=$c_1h_1+...+c_Jh_J=H$로 나타낼 수 있다. N차원의 확장된 최적화 문제를 위한 목적함수는 아래와 같이 정의 된다.

$$\begin{aligned} D^2 = & \sum_{j=1}^{J} \left\{ m_{I(g_j)} - 1 + \left(\sum_{q=1}^{J} B_{I(g_j)}(h_q) \right) - Z \right\}^2 + \\ & \sum_{k=1,k\in\{I(g_j)\mid 1\le j\le J\}}^{K} \left\{ m_k + \left(\sum_{j=1}^{J} B_{k(h_j)} \right) - Z \right\}^2 \end{aligned} \quad (10)$$

여기서 $I(g_j)$는 편향치 벡터 g_j가 변이 전 속해있던 버켓의 인덱스를 뜻한다. 위 목적함수의 제약 식은 아래와 같다.

$$\sum_{j=1}^{J} c_j h_j = H \quad \text{and} \quad \|h_j\| = 1, \quad \text{for } j = 1,2,...,J, \quad (11)$$

여기서 $\|\cdot\|$는 벡터의 크기를 뜻한다.

4.3 근사치 최적화 함수

우리는 목적함수의 최소값을 구하기 위해 미분할 필요성이 있으므로, 그림 3과 같이 복스카르 함수와 근사한 구간 함수를 정의했다. 근사화된 곡선의 어떤 점이라도 원래의 복스카르 함수의 내부에 존재한다. 본 연구에선 간단한 계산을 위해 근사화를 간단한 이차 곡선을 통해 구현하였다.

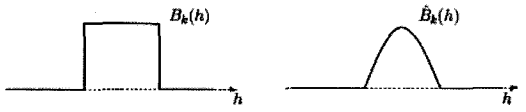


그림 3 복스카르 함수와 근사치 곡선

그림 4는 원래의 의사 난수 분포를 가지는 노이즈에 여러 번의 많은 제어를 가했을 때 카이-스퀘어 통계 값의 변화를 보여준다. 그림에서 보듯, 이상적인 Z값을 기준으로 하였기 때문에 카이-스퀘어 통계 값이 거의 일정하게 감소함을 알 수 있다. 우리는 최적화 문제를 풀기 위해 Spellucci가 제안한 등호 제약이 들어간 순차적 이차 프로그래밍방법(sequential equality constrained quadratic programming method)[17]과 액티브 셋 테크닉(active set technique)[18]을 사용하였다.

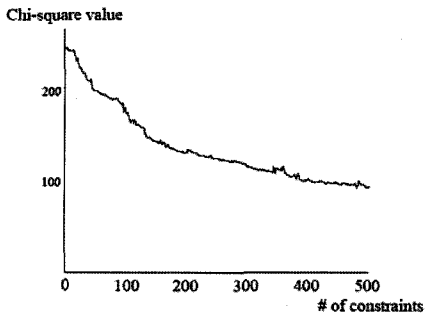


그림 4 노이즈 함수에 제어가 가해짐에 따른 카이-스퀘어 통계치의 변화

5. 해가 없을 경우의 보완점

이번 절에선, 우리가 지금까지 제안한 노이즈 제어의 알고리즘을 통한 최적화 문제가 풀리지 않을 경우에 대한 대안을 제시하려 한다. 최적화 문제가 풀리지 않는 경우는 최적화 과정의 두 가지 제약이 서로 겹치지 않을 경우이다(그림 5). 이번 장에서, 우리는 이런 문제점을 극복하기 위해 정의역 이동(Noise shifting)과 노이즈 진폭 한계치 조절(Noise scaling)라는 추가적인 요소를 제안한다.

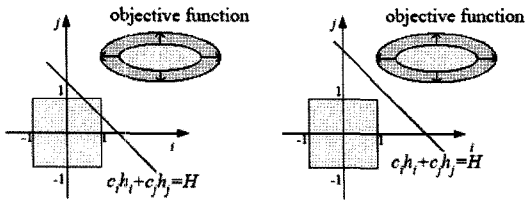


그림 5 1차원 노이즈에 경우 목적함수와 제약 식의 관계: (a) 해가 존재할 경우: 팽창하는 에너지 함수(목적함수)가 처음 만나는 붉은 선위의 점이 해이다. (b) 해가 없는 경우: 두 개의 제약식이 서로 만나지 않을 경우 해가 존재하지 않는다.

5.1 노이즈 정의역 이동

3절에서 언급했던 알고리즘에 따르면, 편향치 노이즈 함수는 언제나 정수 격자에서 0을 지나는 특징을 갖고 있다(zero-crossing). 만약 사용자가 노이즈 제어 중 정수 격자에 특정 값을 할당하게 되면 최적화 문제에선 해를 찾을 수 없다. 하지만, 우리는 노이즈 함수를 통해 연속되는 특정한 값들을 원하는 게 아니라 난수의 패턴을 원한다. 따라서 노이즈 함수의 기본적 성격인 정적임(stationary)을 사용할 수 있다: "비록 정의역이 움직인다 하여도, 노이즈 함수는 언제나 비슷한 난수 패턴을 가진다." 이 성질을 이용해서 우리는 0을 지나는 문제를 해결 할 수 있다.

그림 6의 (a)와 (b)에서 볼 수 있듯이, 정수 격자 근처에 노이즈 값을 제어하는 유동성은 정수격자의 중간에서 제어하는 것보다 떨어진다. 왜냐하면 격자 근처에서 값의 변이가 일어날 때 노이즈의 한계 값인 $\|\text{Noise}(x)\| \le 1$ 을 지킬 수 없을 확률이 커지기 때문이다. 비슷한 이유로, 작은 값을 가지는 노이즈 구간에 제어를 가할 경우가 더 큰 유동성을 가지게 된다(그림 6의 (c)와 (d) 참조).

우리는 해가 없는 경우를 최소화 하는, 즉 제어의 유동성을 최대화 시킬 수 있는 도메인의 이동 Δx 를 구하는 최소화 문제를 고안해 냈다.

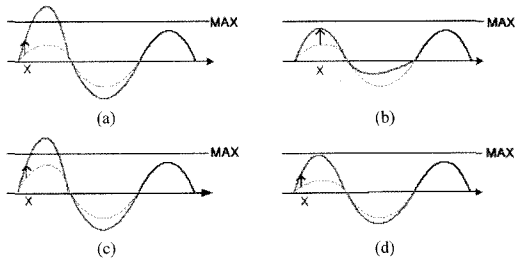


그림 6 노이즈 정의역 이동 조건: (a) 제약이 0을 지나는 정수격자 근처일 때는 해가 없을 경우가 생긴다, (b) (a)와 같은 크기의 제어가 격자 중심에서 일어날 경우는 해가 존재, (c)는 기존의 노이즈 값이 (d) 보다 크므로 해가 없을 가능성이 크다.

$$\text{minimize} \quad w_m\{x_m-(x+\Delta x)\}^2+\{\text{Noise}(x+\Delta x)\}^2, \quad (12)$$

여기서 $x_m=\lfloor x\rfloor+0.5$ 를 뜻한다. 식 (12)에서 첫 번째 항은 정수 격자와 들어온 좌표와의 차이를 나타내고, 두 번째 항은 노이즈 함수의 결과 값의 크기를 나타낸다. 가중치 w_m은 두 항의 공헌도를 제어하는데 쓰인다. $\text{Noise}(x+\Delta x)$ 를 미분하는 것은 의미가 없으므로, 위의 목적함수의 최소화 문제를 풀기 위해, 우리는 주어진 노이즈 함수를 3차 보간한 결과를 사용하였다.

5.2 진폭 한계치 조절

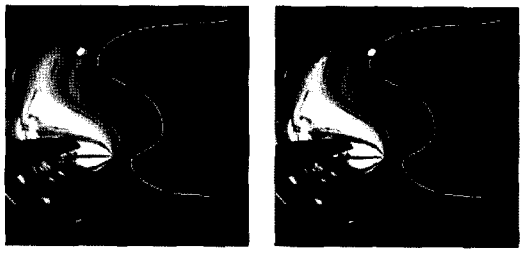
사용자는 노이즈 함수의 진폭 한계치 이상의 변이를 요구 할 가능성이 있다. 이럴 경우 우리는 사용자의 요구를 만족하면서 노이즈의 특성을 보존하기 위해 전체 노이즈의 진폭 한계치를 조절하는 방법을 택하였다. 만약 너무 큰 변이를 요구해서 Noise(x)=H가 풀리지 않을 때, 우리는 제어를 만족하기 위해 진폭에 한계치를 확장 하였다.

$$\text{Noise}(x)=\frac{H}{S}, \quad (13)$$

S는 주어진 제어의 크기에 맞는 적당한 양수 값이다. 결과적으로 전체 노이즈의 적용에 있어서도 진폭을 조절한 노이즈 $S\cdot\text{Noise}(x)$ 가 사용된다.

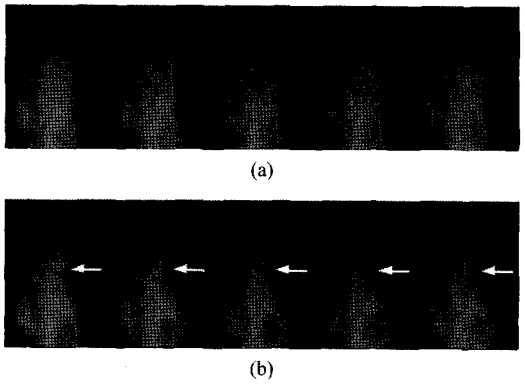
6. 실험 결과

그림 7은 애니메이션 패스에 제어를 가한 경우를 보여준다. 원래의 패스 $P(t)$에 어떤 일차원 노이즈 Noise(t)가 더해졌을 경우, 움직이는 물체가 다른 물체



(a) (b)

그림 7 애니메이션 패스 제어 : (a) 원래의 흰색 커브에 노이즈 함수가 추가된 파란색 커브가 다른 물체를 뚫고 지나간다. (b) 노이즈 제어를 사용해서 충돌을 방지



(a)

(b)

그림 8 불꽃 애니메이션 제어: (a) 기존의 불 애니메이션, (b) 화살표 방향으로 제어가 이루어진 후의 결과

를 뚫고 들어갈 위험이 생긴다(그림 7(a)). 충돌문제를 해결하기 위해, 충돌 전 노이즈에 제약을 가하게 되면 문제 해결이 가능하다. 우리는 물체들이 움직이면서 자동으로 충돌을 계산해 미리 노이즈를 제약하는 시스템을 만들었다.

그림 8에서 볼 수 있는 불 애니메이션 또한 노이즈 제어 기술이 들어간 것이다. 애니메이션은 이미 널리 알려진 방법인 3차원 노이즈가 더해진 자유 곡선을 통해 3차원 텍스처를 만드는 방법을 사용했다. 우리는 노이즈의 특정부분에 계속해서 특정 방향으로의 제어를 줌으로써, 바람이 불어와 불이 한쪽으로 치우치는 결과를 간단히 시뮬레이션 할 수 있었다. 이 기술을 이용하면 불의 모양을 바꾸기 위해 자유 곡선을 바꿀 필요성이 없어진다.

그림 9에선 주어진 점진적 텍스처(그림 9(a))를 흰 화살표 방향으로 제어를 가하는 결과인 그림 9(b)를 볼 수 있다. 이를 통해 점진적 텍스처 제약 툴의 개발이 가능하다. 그림 9(c)는 제어 전 텍스처와 제어 후 텍스처의 차이를 보여준다. 마지막으로 그림 9(d)는 노이즈 진

(a) (b) (c) (d)

그림 9 점진적 텍스처의 제어 : (a) 노이즈 함수를 사용한 기존의 나무무늬, (b) 화살표 방향으로 제어한 결과 (c) (a)와 (b)의 차이, (d) 진폭을 두 배로 확장했을 때의 결과

(a) (b)

그림 10 지형 제어: (a)제어 전 지형, (b)제어 후 지형

폭한계치를 조절해서 더욱 큰 변이를 일으킨 결과이다.

그림 10은 지형 조절에 사용된 노이즈 제어를 보여준다. 일반적인 평지에 노이즈로 만든 난수를 높이로 적용해 계곡과 같은 굴곡을 만들어 내었다. 만약 디자이너가 기본 형태의 변이 없이, 특정 지역의 높이를 바꾸고 싶다면, 노이즈 제어 기술은 효과적으로 쓰일 수 있을 것이다.

7. 결론 및 향후 계획

본 논문에선 지금까지 노이즈 자체의 고유한 성질을 유지하면서 사용자가 원하는 형태로 노이즈를 제어할 수 있는 방법을 제안하였다. 우리는 우리의 연구가 여러 다른 분야에 적용되어 더욱 발전할 수 있을 것이라 예상한다. 비록 우리가 많은 실험적 예제를 보였지만, 이것보다는 훨씬 많은 영역에 노이즈 제어 기술이 적용될 수 있을 것이다.

노이즈의 제어를 위해, 본 논문에선 편향치 테이블의 이상적 분포를 측정하는 근거로 카이-스퀘어 통계치 테스트만을 사용하였다. 하지만 이미 널리 알려진 시리얼 테스트(serial test), 런즈 테스트(runs test), 상호성 테스트(correlation test)와 같은 다른 통계치 테스트 방법도 적용이 가능 할 것이다. 따라서 목적함수의 디자인에 있어서 더욱 최적화된 많은 형태로 확장이 가능할 것이다.

참 고 문 헌

[1] K. Perlin. "An image synthesizer," In SIGGRAPH '85 Proceedings, pages 287-296, 1985.

[2] D. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, S. Worley, B. Mark, and J. Hart. Texture & Modeling: A Procedural Approach, 3rd Ed. Morgan Kaufmann, 2002.
[3] A. A. Apodaca and L. Gritz. Advanced Renderman: Creating CGI for Motion Pictures. Morgan Kaufmann, 2000.
[4] A. Lamorlette. "Structural modeling of frames for a production," SIGGRAPH '02 Proc, pages 729-735, 2002.
[5] F. K. Musgrave. "Great balls of fire," SIGGRAPH '97 Animation Sketches, Visual Proc., pages 259-268, 1997.
[6] K. Perlin. "A unified texture/reflectance model," In SIGGRAPH '84 Advanced Image Synthesis course notes, 1984.
[8] J. P. Lewis. "Algorithms for solid noise synthesis," Computer Graphics, 23(3):263-270, 1989.
[7] K. Perlin and F. Neyret. "Flow noise," In Proceedings of SIGGRAPH '01 Technical Sketches and Applications, page 187, 2001.
[9] Joshua Schpok, Jpseph Simons, David S. Ebert, and Charles Hansen. "A real-time cloud modeling, rendering, and animation system," In Proceedings of Eurographics /SIGGRAPH Symposium on Computer Animation, pages 160-166, 2003.
[10] K. Perlin. "Realtime responsive animation with personality," IEEE Transactions on Visualization and Computer Graphics, 1(1):5-15, 1995.
[11] J. C. Hart. "Perlin noise pixel shaders," In Proceedings of Graphics Hardware 2001: Eurographics/ SIGGRAPH Workshop, pages 87-94, 2001.
[12] G. Gardner. "Simulation of natural scenes using textured quadric surfaces," Computer Graphics, 18(3):11-20, July 1984.
[13] K. Perlin. "Improving noise," In SIGGRAPH '02 Proceedings, pages 681-682, 2002.
[14] H. T. Reynolds. Analysis of Nominal Data. Sage Publications, 1984.
[15] D. E. Knuth. The Art of Computer Programming. Addison-Wesley, 1998.
[16] H. T. Reynolds. Analysis of Nominal Data. Sage Publications, 1984.
[17] P. Spellucci. "An sqp method for general nonlinear programs using only equality constrained subproblems," Mathematical Programming, 82:413-448, 1998.
[18] R. Fletcher. Practical Methods of Optimization. JohnWiley and Sons, 1987.

윤 종 철
2003년 아주대학교 미디어학사. 2005년 아주대학교 미디어 석사. 2005년~현재 연세대학교 컴퓨터과학과 박사과정. 관심분야는 컴퓨터 그래픽스, 컴퓨터 애니메이션, 영상처리

이 인 권
정보과학회논문지 : 시스템 및 이론
제 32 권 제 8 호 참조

최 정 주
정보과학회논문지 : 시스템 및 이론
제 32 권 제 1 호 참조