하드웨어를 이용한 효율적인 인공풍 시뮬레이션 방법

이 남 경[†] · 백 낙 훈[††] · 유 관 우[†††]

요 약

본 논문에서는 자연풍에 비해 상대적으로 작은 영역에 영향을 주는 인공풍을 시뮬레이션하는 방법을 제안한다. 이를 위해 인공풍의 진행 형태를 모델링하는 방법을 제안하고, 제안하는 바람 모델이 시뮬레이션 환경에 미치는 영향을 계산하는 효율적인 방법도 제안한다. 제안하는 방법에서는 인공풍의 영향을 계산하는 수식이 기존의 조명 모델(Illumination Model)에서의 조도 계산식(Intensity Equation)과 유사함을 보이고, 이를 이용하여 바람에 의한 영향을 직접 수식으로 계산하지 않고 집중광선(Spot Light)에 대한 조도 계산식을 사용하여 효과적으로 인공풍의 힘을 계산한다. 제안하는 방법은 실시간 처리가 가능하며, 컴퓨터 게임이나 가상 현실과 같은 다양한 분야에 적용할 수 있다.

키워드 : 컴퓨터 그래픽스, 인공풍, 조명 모델, 실시간 시뮬레이션

An Efficient Hardware-Based Simulation Method for Artificial Winds

Namkyung Lee[†] · Nakhoon Baek[††] · Kwan-Woo Ryu[†††]

ABSTRACT

In this paper, we present a simulation model for artificially generated winds which affect relatively restricted regions in comparison with natural winds. We first propose an artificial wind propagation model, and then propose an efficient way of calculating the effect of this wind model in the simulation environment. Through showing that our wind force calculation equation is similar to the typical intensity equation for illumination models, we can calculate the wind force indirectly by using the intensity equations for spotlights, and hence we can reduce the simulation time. Our method shows real-time capabilities, and thus can be used various real-time applications including computer games, virtual environments, etc.

Key Words : Computer Graphics, Artificial Wind, Illumination Model, Real-time Simulation

1. 서 론

바람이 불어 덜컹거리는 창문, 바람에 펄럭이는 깃발, 이리저리 날리는 나뭇잎들, 선풍기 바람에 날리는 종이들, 그리고 바람을 따라 휘어지는 풀들과 같이 바람은 우리 생활 주변에서 가장 흔하게 볼 수 있는 자연 현상 중 하나이다. 하지만 이런 바람에 의한 영향을 수학적으로 모델링 하려면 유체의 흐름을 다루어야 하는 어려움이 있다.

컴퓨터 그래픽스 분야에서는 기존의 연구 결과들이 자연 현상으로서의 바람을 표현하는 데에 초점을 맞추어 왔다 [1-9]. 반면에, 바람에 관계된 현상들 중에는 자연적으로 발생한 바람 못지 않게 인공적으로 생성되는 경우도 상당한 비중을 차지한다. 즉, 인간의 입으로 생성할 수 있는 바람이나, 선풍기, 에어컨, 환풍기 등에서 만들어지는 인공풍(人工風)은 자연풍(自然風)과는 다른 방법으로 모델링 될 수 있을 것이다.

본 논문에서는 인공풍의 진행 형태를 모델링 하는 방법을 제안하고, 제안하는 바람 모델이 시뮬레이션 환경에 미치는 영향을 계산하는 효율적인 방법도 제안한다. 제안하는 방법에서는 인공풍의 영향을 계산하는 수식이 기존의 조명 모델(Illumination Model)에서의 조도 계산식(Intensity Equation)과 유사함을 보이고, 이를 이용하여 바람에 의한 영향을 직접 수식으로 계산하지 않고 집중광선(Spot Light)에 대한 조도 계산식을 사용하여 인공적인 바람의 영향을 계산한다. 조명 모델의 조도 계산식은 이미 하드웨어로 구현되어 있기 때문에, 이를 이용하여 계산량을 줄일 수 있다. 마지막으로 컴퓨터 애니메이션에서의 효율적인 처리를 위해 바람에 영향을 받는 물체를 찾아내는 효과적인 방법도 제안한다.

본 논문은 인공풍을 컴퓨터 게임, 가상 현실(Virtual Reality)이나, 컴퓨터 애니메이션 등에서 효과적으로 다룰 수 있도록 하는 데에 목적을 두고, 실시간 처리(Real-time

※ 이 논문은 정보통신부의 재원으로 한국전자통신연구원 HD게임연구팀의 일부 지원을 받았음. 또한 교육인적자원부의 재원으로 한국학술진흥재단의 일부 지원을 받았음(R05-2004-000-12641-0).
† 준 회 원: 경북대학교 대학원 컴퓨터공학과 박사과정
†† 정 회 원: 경북대학교 전자전기컴퓨터학부 조교수(교신저자)
††† 정 회 원: 경북대학교 컴퓨터공학과 교수
논문접수: 2006년 9월 8일, 심사완료 : 2006년 11월 16일

Processing)에 초점을 맞추었다. 본 논문에서 제시하는 모델은 자연풍에 대한 기존의 연구 결과들과 함께 사용됨으로써 통합된 바람 모델을 만들어 낼 수 있을 것이다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 바람과 관련된 이전 연구들을 살펴보고 3장에서는 본 논문에서 제안하는 인공풍 모델에 대해 설명한다. 4장에서는 제안하는 바람 모델의 구현 결과를 보이고, 마지막으로 5장에서 결론 및 향후 과제들을 기술한다.

2. 이전 연구들

바람에 대한 몇몇 연구들이 있어왔지만, 컴퓨터 그래픽스 분야의 이전 연구들은 저자들이 아는 한 모두 자연풍에 대해서 연구해 왔다. 이에 따라 시뮬레이션 환경 전체에 영향을 주는 속도장(Velocity Field)의 형태로 바람을 표현하는 것이 일반적이었다. 이전 연구들은 크게 프리미티브(Primitive)를 이용하는 방식과 계산 유체역학(Computational Fluid Dynamics)을 이용하는 방식으로 나눌 수 있다.

Wejchart와 Haumann[1]은 바람에 날리는 낙엽들의 움직임을 나타내기 위해서 공기역학(Aerodynamics) 모델을 사용하였다. 그들은 복잡한 바람의 장을 라플라스 방정식을 만족하는 흐름 프리미티브(Flow Primitive)들을 중첩 사용하여 단순하게 표현하였다. Shinya와 Fournier[2]는 건축공학의 연구를 바탕으로 간단한 확률적인 바람모델을 제안하였는데, 퓨리어 변환과 난수를 사용하여 만든 바람으로 들풀, 대나무 등의 움직임을 시뮬레이션하였다. Stam과 Fiume[3]은 담배연기와 같은 연기의 움직임을 나타내기 위해서 바람의 장을 두 단계로 나누어 만들었다. 전반적인 바람의 형태는 Wejchart방법과 같이 흐름 프리미티브를 사용하여 생성했고, 사용자가 원하는 위치에 작은 크기의 난수 벡터들을 설정하여 전체 바람의 장을 만들었다. Perbet과 Cani[4]는 바람의 영향을 받는 풀들의 움직임을 표현하기 위해서 프리미티브와 시간에 따라 변하는 확률적인 변수를 사용해서 바람의 장을 만들었다. 이처럼 프리미티브들을 중첩 사용하여 바람의 장을 만드는 방법들은 쉽고 간단하게 바람을 만들 수 있는 장점이 있지만, 바람의 장이 고정되어 사용자와의 상호작용이 어렵다는 단점이 있다.

바람과 같은 유체의 움직임을 표현한 나비어-스톡 방정식(Navior-stoke's Equation)을 계산하기 위해, 계산 유체역학 기법을 사용한 연구들도 있어 왔다. Stam[5]은 나비어 스톡 방정식 중에서 비선형항인 이류항(Advection Term)때문에 생기는 많은 계산량을 줄이기 위해 세미-라그랑지안(Semi-lagrangian)방법과 수치연산 시 긴 시간 간격을 보장하는 암시적 적분방법(Implicit Integration Method)을 사용하여 대부분의 경우에 안정적인 유체 계산법(Stable Fluid Slover)를 제안했다. Stam의 방법은 연기나 물과 같은 유체를 표현하는데 좋은 접근법이 되고 있다. 이런 Stam의 방법을 개선한 여러가지 연구들[6-9]이 최근까지 많이 진행되고 있다. 반면에 계산 유체역학 기법들은 많은 계산량을 요구하므로 실시간 적용이 불가능하다는 단점을 가지고 있다.

이처럼, 기존의 연구 결과들은 모두 시뮬레이션 환경 전체에 영향을 주는 자연풍에 대한 연구에 집중했고, 주로 유체 역학적 방법에 기초하여, 계산량이 많거나 사용자와의 상호작용이 어렵다는 단점을 가지고 있다.

3. 인공풍 모델

본 논문에서는 자연풍에 비해 상대적으로 작은 영역에 영향을 주는 인공풍을 시뮬레이션하는 방법을 제안한다. 제안하는 방법은 실시간 처리가 가능하며, 컴퓨터 게임이나 가상 현실과 같은 다양한 분야에 적용할 수 있다.

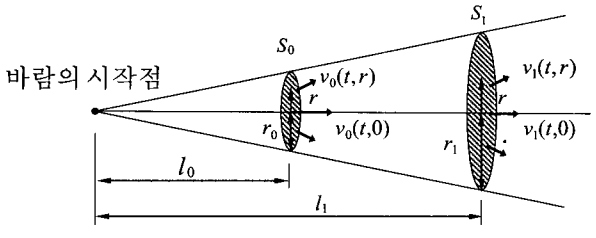
본 절에서는 인공풍을 생성하기 위해 필요한 세 가지 요소들을 기술한다. 우선, 3.1절에서 인공풍이 진행해 나가는 형태를 전체적으로 모델링 한다. 그리고 3.2절에서는 바람이 물체에 부딪쳤을 때, 해당 물체에 가해지는 힘을 계산한다. 이 때, 본 논문에서 제안하는, 바람의 힘을 계산하는 식과 조명 모델의 조도 계산식의 유사성을 이용하여, 시뮬레이션 속도를 높이는 방법에 대해 기술한다. 마지막으로, 3.3절에서 바람의 영향을 받는 물체를 찾아내는 방법에 대해 기술한다.

3.1 바람의 형태

이전 연구 결과들에서 바람은 시뮬레이션 환경 전체에 영향을 주는 장(Field)의 형태로 모델링 되었다. 하지만, 선풍기, 에어컨, 사람의 입에서 만들어지는 바람은 제한된 영역에 있는 물체들에 영향을 미친다. 따라서 인공풍은 자연풍과는 다른 방식으로 모델링 해야 한다.

본 논문에서는 (그림 1)에서와 같이 인공풍은 바람의 시작점에서 가상의 원뿔 형태(Cone shape)로 퍼져 나간다고 가정한다. 바람은 진행축을 기준으로 상하가 완전 대칭되게 퍼져나간다. 예를 들어, 선풍기에서 만들어지는 바람은 전체적으로 원뿔 형태로 퍼져나가고, 원뿔축을 중심으로 상하 대칭으로 진행한다. 원뿔 안에 있는 물체들이 선풍기에서 나오는 바람의 영향을 받으며, 선풍기 바람의 중심축에 가까이 있는 물체일수록 바람의 힘을 더 크게 받는다.

(그림 1)에서 바람의 시작점은 단면 S_0의 중심으로부터 거리 l_0만큼 떨어져 있다. 사용자는 인공 바람의 형태를 변경하기 위해서 l_0와 S_0의 반지름인 r_0를 조절 할 수 있으며, 또한 S_0의 중심에서의 초기 바람속도 $v_0(t,0)$를 조절할



(그림 1) 바람의 진행형태

수 있다. 계산의 편의를 위해 S_0의 중심에서 r만큼 떨어진 위치에서의 바람속도는 $v_0(t,r)$로 정의한다. 앞서 설명한 것처럼 원뿔의 중심에 가까이 있는 물체일수록 원뿔 가장자리에 위치한 물체에 비해 더 큰 바람의 영향을 받는다. 이를 수식적으로 표현할 수 있는 다양한 함수들이 있지만, 본 논문에서는 계산의 편의를 위해 코사인 함수를 사용하여 이를 근사한다. 즉 $v_0(t,r)$는 $v_0(t,0)\cos r(0 \le r < r_0)$로 설정하였다.

3.2 바람의 힘 계산

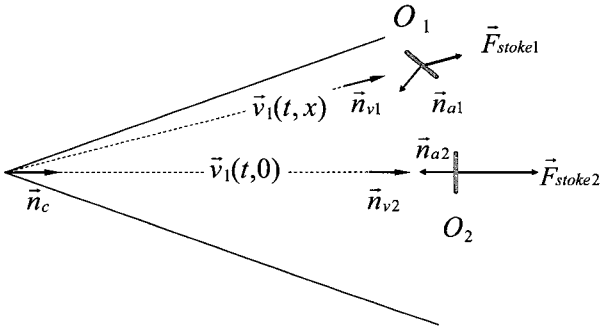
인공풍이 진행해 나가는 형태를 결정했다면, 다음은 바람의 영향을 받을 물체들에 가해지는 힘을 구하여야 한다. 이를 위해 먼저 바람의 시작점으로부터 거리 $l_1(>l_0)$만큼 떨어져 있는 위치에서의 바람 속도를 구해야 한다. 몇몇 이전 연구들[1, 2]에서는 공기역학의 수식들을 사용하였다. 본 논문에서는 인공풍에 적용할 수 있게 이들 수식을 변형한다. 이를 위해 공기는 압축되지 않고, 원뿔 형태 밖으로 공기가 빠져나가지 않는다고 가정한다. 이는 정상 풍속의 바람에 적용될 수 있다[1]. 이런 가정하에 유체의 흐름에서는 단위 시간에 단면 S_0와 S_1을 각각 통과하는 유체의 질량이 같아야 한다. 따라서, 이 두 단면 사이에서의 유체의 흐름은 연속 방정식(Continuity Equation)으로 다음과 같이 표현할 수 있다.

$$C_0\, v_0(t,0) = C_1\, v_1(t,0) \tag{1}$$

식 (1)에서 C는 단면의 넓이를, $v(t,0)$는 유체 즉 공기의 속도를 의미한다. 또한 첨자 0과 1은 각각 원뿔의 꼭지점으로부터 l_o와 l_1만큼 떨어진 곳에서의 값들을 말한다. 식 (1)과 (그림 1)의 기하학적인 관계를 통해서 다음 식을 유도할 수 있다.

$$\frac{v_1(t,0)}{v_0(t,0)} = \frac{C_0}{C_1} = \frac{\pi r_0^2}{\pi r_1^2} = \frac{\pi l_0^2}{\pi l_1^2} = \frac{l_0^2}{l_1^2} \tag{2}$$

단면적이 A인 단면 S_1에 가해지는 힘은 스톡스 항력 방정식(*Stoke's drag equation*)[8]을 사용하여 다음과 같이 구할 수 있다.



(그림 2) 인공풍의 힘 계산

$$\vec{F}_{\text{stoke}} = \rho A v_1^2(t,0)(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v) \tag{3}$$

식 (3)에서 $\vec{n}_v$과 $\vec{n}_a$은 각각 바람의 방향을 가리키는 단위 벡터와 단면의 단위 법선 벡터를 나타내며 상수 ρ는 유체의 밀도이다. 식 (2)와 식 (3)을 이용해서 바람의 근원으로부터 거리 l_1만큼 떨어진 곳에 있는 물체에 가해지는 바람의 힘을 다음과 같이 구할 수 있다.

$$\vec{F}_{\text{stoke}} = \rho A \frac{v_0^2(t,0)}{l_1^4}(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v) \tag{4}$$

(그림 2)의 물체 O_1과 같이 원뿔 중심축 위에 있는 물체들은 식 (4)를 사용하여 물체에 가해지는 바람의 힘을 구할 수 있다. 물체 O_2와 같이 중심축에서 벗어난 물체에 가해지는 바람의 힘을 구하기 위해서는 식 (4)를 다음과 같이 수정할 수 있다.

$$\vec{F}_{\text{stoke}} = \rho A \frac{v_0^2(t,r)}{l_1^4}(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v) = \rho A \frac{v_0^2(t,0)\cos r}{l_1^4}(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v) \tag{5}$$

위 식에서 $\cos r$은 그림 (2)의 원뿔 중심축을 나타내는 단위 벡터 $\vec{n}_c$와 $\vec{n}_v$의 내적으로 구할 수 있다. 따라서 이를 사용하여 식 (5)를 다시 정리하면 다음과 같다.

$$\begin{aligned}\vec{F}_{\text{stoke}} &= \rho A \frac{v_0^2(t,0)\cos r}{l_1^4}(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v) \\ &= \alpha A v_0^2(t,0)(\vec{n}_c \cdot \vec{n}_v)(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v)\end{aligned} \tag{6}$$

식 (6)에서 $v_0(t,r)$는 사용자의 설정에 따라 바람의 형태를 결정하는 데에 사용된다. 또, 상수 $\alpha = \rho / l_1^4$는 원하는 원뿔의 형태에 따라 다르게 설정할 수 있다. 실제 세계에서 바람은 물체와 부딪치면서 굴절하거나 회절하는 현상이 일어날 수 있다. 특히, 상대적으로 작은 물체의 뒷부분에서는 회절된 바람이 영향을 미칠 수도 있다. 또, 바람 자체가 가지는 무질서성을 반영하여야 할 필요도 있다. 본 논문에서는 이러한 점들을 고려하기 위해, Stam과 Fiume[3]가 택한 것과 같은 방식으로, 식 (6)에서 계산한 힘에 무질서한 항을 더하여 최종적으로 다음과 같은 식을 사용한다.

$$\vec{F}_{\text{wind}} = \vec{F}_{\text{stoke}} + \vec{F}_{\text{random}} \tag{7}$$

무질서하게 더해지는 $\vec{F}_{random}$는 사용자가 정의하는 상수 β를 이용하여 $|\vec{F}_{random}| < \beta |\vec{F}_{stoke}|$로 계산되며, $\vec{F}_{random}$의 방향은 무작위로 설정하였다. 각 물체에 가해질 힘이 결정되면 물리기반 모델링 기법[11]에 따른 시뮬레이션이 가능하다.

본 논문에서는 바람의 힘을 세 벡터들, 바람의 방향을 가리키는 단위 벡터 $\vec{n}_v$, 단면의 정규단위 벡터 $\vec{n}_a$ 그리고 원뿔 중심축을 나타내는 단위 벡터 $\vec{n}_c$을 이용해 구했다. 이는

조명 모델 중 집중광선(Spot light)의 조도 계산식과 유사하다. 집중광선은 비교적 작은 각도 범위 내에서 특정한 한 방향으로 빛을 방출하는 점 광원을 말하며 점 광원으로부터 발산되는 빛의 각도를 제한한 경우로 볼 수 있다. 일반적으로 주변반사광과 전반사광의 영향을 고려하지 않은 집중광선의 조도(Illumination Intensity)를 구하는 식은 다음과 같다[12, 13].

$$I = A_{att}(I_d k_d)(-L \cdot X)^e (L \cdot N) \quad (8)$$

식 (8)에서 I_d 와 k_d 는 각각 난반사광(Diffuse Light)의 조도와 표면의 반사계수이고, e 는 광택계수이다. 또, L 와 X 와 N 는 각각 집중광원의 방향을 나타내는 벡터, 집중광선의 중심축을 나타내는 벡터, 시점방향을 나타내는 벡터이다. 표면의 밝기는 광원으로부터의 거리에 반비례하므로 이를 반영하기 위해 감쇠항(Attenuation Term)으로 A_{att} 를 사용한다.

식 (6)과 식 (8)에서, 만약 광원의 위치가 인공바람의 시작점과 동일한 위치에 있고, 집중광선의 중심축 X 의 방향과 $\vec{n}_c$ 의 방향이 같다면, 식 (8)의 $(-L \cdot X)$ 항과 $(L \cdot N)$ 항은 각각 식 (6)의 $(\vec{n}_c \cdot \vec{n}_v)$ 항과 $(\vec{n}_v \cdot -\vec{n}_a)$ 항에 대응된다. 또한 식 (8)의 상수항들은 식 (6)의 상수항들과 유사하다. 따라서 식 (8)을 이용하여 식 (6)을 다시 정리할 수 있다.

$$\vec{F}_{stoke} = \alpha A v_0^2(t,0)(\vec{n}_c \cdot \vec{n}_v)(\vec{n}_v \cdot (-\vec{n}_a))(\vec{n}_v) = \kappa I \vec{n}_v \quad (9)$$

식 (9)에서 κ 는 사용자 정의 상수이다. 본 논문에서는 식 (9)를 이용하여 바람의 힘을 직접 계산하지 않고, 빛의 조도를 사용하여 바람의 힘을 쉽게 구할 수 있다. 빛의 강도를 구하는 것은 현재 출시되고 있는 대부분의 그래픽카드에 하드웨어적으로 구현되어 있으므로, 이를 이용하여 시뮬레이션 속도를 향상시킬 수 있다.

3.3 바람의 영향을 받는 물체 찾기

앞서 제시한 바람 모델에서, 물체가 원뿔의 내부에 포함되어 있는 경우에 바람의 영향을 받는다. 반면에, 특정한 면이 다른 물체에 의해 가려지는 경우에는 바람의 직접적인 영향을 받지 않는다. 따라서, 원뿔의 꼭지점으로부터 가시(Visible)적이고, 원뿔의 내부에 포함된다는 조건을 만족하는 물체들만이 바람의 영향을 받는다. 바람의 영향을 받는 물체들을 찾기에 앞서, 부분적으로 가려진 물체들을 고려하기 위해서 물체들은 메쉬(Mesh)로 구현되어 있다고 가정한다.

본 논문에서는 바람의 영향을 받는 면들을 찾기 위해서 가시면 추출 방법(Visible Surface Detection)을 응용한다[12, 13]. 이를 위해 먼저 가시면 추출 방법에서의 카메라의 위치를 바람 모델의 원뿔의 꼭지점에 대응시키고, 바람의 진행 방향을 카메라의 시점방향과 일치하게 설정한다. 본 논문에서 제안하는 바람 모델은 원뿔 형태이므로, 가상의 원뿔은 렌더링 파이프라인을 거치면서 절두체(Frustum)의 뒷면과 교차하여 원의 형태를 지닌 단면을 생성한다. 물체가 원뿔 내에 위치하는가는, 렌더링 파이프라인을 거쳐 생성된 최종 화면상에서 원뿔에 대응되는 원의 내부에 물체가 포함되는지를 묻는 2차원 문제가 된다. 컴퓨터 그래픽스에서 사용되는 가시면 추출 방법들에는 여러 가지가 있다. 본 논문에서는 일반적으로 적용가능하고, 처리 시간을 향상시키기 위해 깊이 버퍼 기법(Depth Buffer Method)을 사용한다[12, 13]. 이 방법은 그래픽스 하드웨어의 지원을 받을 수 있다는 장점이 있다. 원래의 깊이 버퍼 기법에서는 가시면들이 표시된 화면이 최종 결과가 된다. 하지만 바람 모델의 경우에는 어느 면이 보이는 지를 찾아내야 한다. 또, 바람의 모델링에 있어서는 그 특성상 면 상에서 영향을 받는 부분을 정확히 찾아낼 필요는 없다.

본 논문에서는 각각의 면 위에 표본점들을 분포시킨 후, 각 표본점들이 대응되는 면 상의 일정 부분의 가시성을 결정하도록 하여 처리 속도를 향상시켰다. 이를 위해서 전처리 과정(Preprocessing)으로 표본점들의 위치 정보와 인덱스를 자료구조에 저장한다. 렌더링 파이프라인을 거친 최종화면에서 이들 표본점들을 읽어와서, 표본점들의 가시성 여부를 결정한다. OpenGL의 경우[14], 객체 공간 좌표를 취하여 변환된 윈도우 좌표를 반환하는 *gluProject* 함수를 이용하여 표본점이 투영될 2차원의 좌표를 계산할 수 있다. 또 특정좌표의 픽셀(pixel)정보를 *glReadPixel* 함수를 읽어 올 수 있다. 본 논문에서는 *gluProject* 함수와 *glReadPixel* 함수을 사용하여 표본점들의 픽셀 정보를 읽어온다. 읽어온 픽셀이 원뿔의 내부에 있고, 인덱스가 자료구조의 인덱스 값과 일치한다면, 그 표본점은 바람의 영향권 안에 있는 점이다. 만약 표본점이 바람의 영향권 안에 있다면, 픽셀의 색상값과 식 (10)을 이용하여 효율적으로 바람의 힘을 계산할 수 있다. 이처럼 본 논문에서 제안하는 방법을 사용하면, 바람의 영향을 받는 면을 찾는 것과 바람의 힘을 계산하는 것을 프레임버퍼를 단지 한 번 읽어 오는 것으로 해결할 수 있다.

4. 실험 결과

앞 절에서 설명한 가상 바람 모델을 이용하여 바람의 영향을 받는 물체들의 움직임을 물리 기반 모델링 기법[11]으로 모델링하는 프로토타입 시스템을 구현하였다. 그래픽스 출력을 위한 OpenGL 라이브러리와 Visual C++ 6.0으로 프로그램하였다. 실험은 인텔 펜티엄 4 2.8 GHz 프로세서와 GeForce FX5700 그래픽 카드를 장착한 개인용 컴퓨터에서 하였다. 향후에 전체 계산 과정에 최적화 기법들을 적용하면, 더 빠른 실행 속도를 보일 것으로 기대된다.

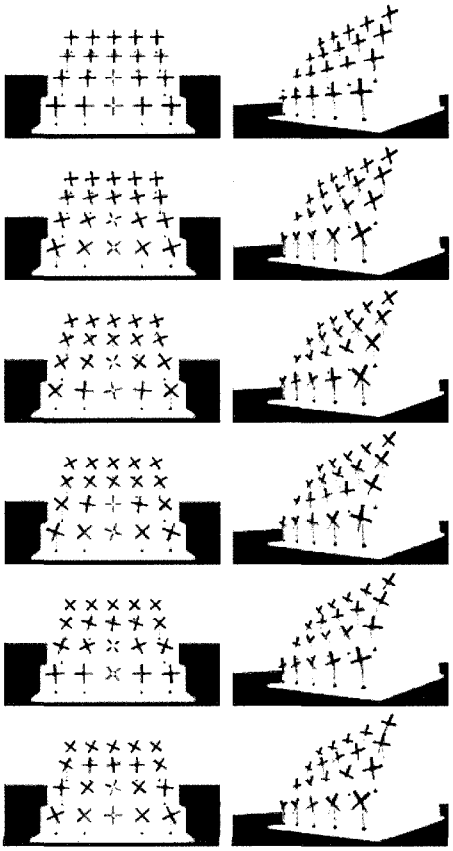
(그림 3)은 바람개비들이 정면에서 부는 바람의 영향을 받았을 때의 움직임을 보여 준다. 이 때 왼쪽 열은 정면에서 본 이미지들이고, 오른쪽 열은 동일한 프레임을 측면에서 본 이미지들이다. 실험을 위해 정면에서 화면 방향으로 집중광선을 설정하였다. 바람 모델의 시작점을 집중광선의 광원과 같은 위치에 두고, 바람의 진행 방향도 집중광선의

중심축과 같게 한다면, 더 밝은 위치에 있는 바람개비일수록 더 강한 바람의 영향을 받게 된다. (그림 3)의 경우는 가장 아래 줄에 있는 가운데 바람개비가 가장 밝으므로, 날개가 가장 빨리 회전한다.

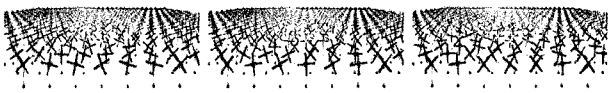
(그림 4)는 1000개의 바람개비를 시뮬레이션한 경우이다. (그림 3)의 경우와 같이 정면에서 화면 방향으로 집중광선을 설정했다. 바람개비의 위치에 따라 바람의 영향을 다르게 받고 있음을 알 수 있다. 바람의 영향권 밖에 있는 바람개비들 즉 그림의 가장자리에 위치한 바람개비들은 날개가 회전 하지 않음을 알 수 있다.

(그림 5)는 선풍기 바람에 흩어지는 종이들의 움직임을 시뮬레이션 한 것이다. 바람은 그림의 좌측에서 우측 방향으로 불고 있다. 종이와 선풍기와의 위치관계 때문에 각 종이에 가해지는 힘이 다르다는 것을 알 수 있다. 즉 바람 모델의 중심축에 가까이 있는 종이들이 더 멀리 날라 간다.

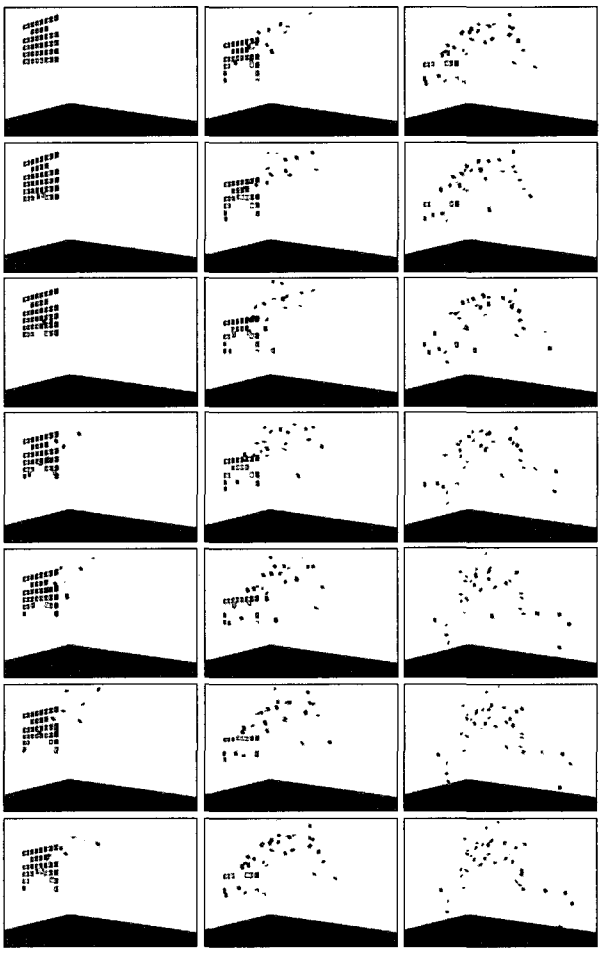
(그림 3)과 (그림 5)의 시뮬레이션은 초당 60 프레임 이상의 렌더링 속도를 보였다. 또 (그림 4)의 경우는 초당 30 프레임 이상 렌더링할 수 있다. 또한 시점으로부터 일정 거리 이상인 물체들에 대해서 LOD 기법을 사용한다면 더 좋은 결과를 얻을 수 있을 것이다. 결과에서 볼 수 있듯이, 본 논문의 바람 모델은 적용시키는 데에는 상당히 짧은 계산 시간으로도 충분하므로 본 논문이 목적했던, 가상 환경에서의 실시간 처리에 사용 가능하다.



(그림 3) 계단에 놓여진 바람개비들



(그림 4) 1000개의 바람개비들



(그림 5) 바람에 날리는 종이들

5. 결 론

실제 세계에서의 바람은 크게 자연 발생적인 것과 인위적으로 생성되는 것으로 나눌 수 있다. 지금까지의 연구 결과들은 주로 자연 현상으로서의 바람을 모델링하였다. 본 논문에서는 사람의 입이나 선풍기, 환풍기, 에어컨 등에 의해 생성되는 바람을 모델링하기 위한 바람 모델을 제시하였다.

본 논문이 제안한 바람 모델은 게임이나 가상 현실과 같은, 실시간 처리가 필요한 분야에서도 사용하기 위해서 계산량을 줄일 수 있도록 설계하였다. 또, 주어진 환경에서 가상의 바람을 받게 되는 물체와, 그 물체에 가해지는 힘을 계산하는 방법을 제시하였다. 특히 집중조명의 식을 사용하여 수식 계산을 최소화하였다. 실제 구현에 있어서는 바람의 힘을 구하는 것과 가상의 바람을 받게 되는 물체를 찾는 것이 프레임버퍼를 한 번 읽음으로써 모두 해결되게 하였다.

이러한 방법들은 인위적으로 생성된 바람이 주어진 환경에 미치는 영향을 효과적으로 반영할 수 있다.

본 논문에서 제시한 방법은 기존 연구들에서 제안한, 자연 발생적인 바람의 모델링과는 서로 보완적인 역할을 담당한다. 따라서 앞으로는 두 모델들간의 통합이 필요하다. 이를 통해 종합적인 바람 생성 모델을 설계할 수 있을 것이다. 동역학에 기초한 애니메이션 시스템과의 통합 역시 앞으로의 연구 과제이다. 또한 바람을 이용하는 사용자 인터페이스는 가상 현실이나 사용자 인터페이스 분야에서 향후에 다루어질 연구 과제이다.

참 고 문 헌

[1] J. Wejchert and D. Haumann, "Animation Aerodynamics," In the Proceedings of ACM SIGGRAPH, pp.19-22, 1991.

[2] M. Shinya and A. Fournier, "Stochastic Motion - Motion Under the Influence of Wind," In Eurographics'92, pp.119-128, 1992.

[3] J. Stam and E. Fiume, "Turbulent Wind Fields for Gaseous Phenomena," In the Proceedings of ACM SIGGRAPH, pp.369-376, 1993.

[4] F. Perbet and M.P. Cani, "Animating prairies in real-time," ACM Symposium on Interactive 3D Graphics, pp.103-110, 2001.

[5] J. Stam, "Stable fluids," In the Proceedings of ACM SIGGRAPH, pp.121-128, 1999.

[6] D. Nguyen, R. Fedkiw, and H. Jensen, "Physically Based Modeling and Animation of Fire," In the Proceedings of ACM SIGGRAPH, pp.721-728, 2002.

[7] X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakumstover and A. Kaufman, "Blowing in the wind," ACM SIGGRAPH/ EUROGRAPHICS Symposium on Computer Animation, pp.75-85, 2003.

[8] A. Selle, N. Rasmussen, and R. Fedkiw, "A Vortex Particle Method for Smoke, Water and Explosions," In the Proceedings of ACM SIGGRAPH, pp.910-914, 2005.

[9] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," Proceedings of SIGGRAPH , pp.457-462, 2004.

[10] V.L Streeter and E. Benjamin, 'Fluid Mechanics', McGraw-Hill, 1998.

[11] A. Witkin and D. Baraff, 'SIGGRAPH 2001 Course Notes on Physically-Based Modeling', 2001.

[12] D. Hearn and M.P. Baker, 'Computer Graphics', Prentice-Hall, 1997.

[13] J.D. Foley, A.S. van Dam, K. Feiner and J.F. Hughes, 'Computer Graphics - principles and practice', 2nd ED., Addison Wesley, 1990.

[14] J. Neider, T. Davis and M. Woo, 'OpenGL Programming Guide : The official guide to Learning OpenGL', Addison-Wesley, 1993.

이 남 경

e-mail : leenk@korea.com

1998년 경북대학교 컴퓨터공학과(학사)

2000년 경북대학교 컴퓨터공학과(공학석사)

2000년~현재 경북대학교 컴퓨터공학과 박사과정

관심분야 : 컴퓨터 그래픽스, 알고리즘

백 낙 훈

e-mail : oceancru@gmail.com

1990년 한국과학기술원 전산학과(학사)

1992년 한국과학기술원 전산학과(공학석사)

1997년 한국과학기술원 전산학과(공학박사)

2004년~현재 경북대학교 전자전기컴퓨터학부 조교수

관심분야 : 컴퓨터 그래픽스, 알고리즘

유 관 우

e-mail : kwryu@knu.ac.kr

1980년 경북대학교 전자공학과(학사)

1982년 한국과학기술원 전산학과(공학석사)

1990년 메릴랜드대학교 전산공학(공학박사)

1982년~현재 경북대학교 컴퓨터공학과 교수

관심분야 : 컴퓨터 그래픽스, 계산기하학, 계산 이론