Journal of Korea Multimedia Society Vol. 10, No. 7, July 2007(pp. 934-943)

모바일 환경에서 불꽃의 실시간 시뮬레이션과 렌더링

우상혁[†], 조미리나[††], 박동규[†††]

요 약

컴퓨터 그래픽스 분야에서 다뤄지는 유체 시뮬레이션은 사실적인 애니메이션에 있어서 필수적인 요소이기는 하지만 그 계산량이 너무 많아서 실시간 시뮬레이션과 렌더링을 위해서는 많은 시스템 자원을 필요로 하여 주로 PC 환경에서 수행되어 왔다. 최근 휴대 전화의 성능이 빠르게 발전함에 따라 3D 게임과 같은 고급 콘텐츠를 모바일 환경에서 사용 가능하게 되었다. 본 논문에서는 PC 환경보다 비교적 성능이 제한적인 모바일 장치에서 위피 플랫폼의 NF3D API를 사용하여 실시간 유체 시뮬레이션을 구현하였다. 유체 시뮬레이션을 구현하기 위하여 나비에-스토크스 식의 풀이가 필요하며, 빠르고 안정적으로 수치해를 시뮬레이션 하기 위해서 Stam의 Stable Fluid 기법을 빌보드에서 구현하여 사용하였다. 시뮬레이션 결과는 빌보드 기법을 통해 화면에 나타냈으며, "루피 스토리"라는 모바일 3D 게임 콘텐츠에 적용하였다.

Realtime Fire Simulation and Rendering on Mobile Environment

Woo SangHyuk[†], Jo MiRiNa[††], Park DongGyu[†††]

ABSTRACT

This paper presents a real-time fire simulation on the mobile phone using stable fluid animation techniques. Stable and fast fluid simulation methods are developed in PC and console games, but fluid simulation and interactive fluid models require too much system resources for applying on mobile environment. We studied and implemented physics-based models for fluids like fire and smoke effects using billboard and stable fluids simulation method on mobile 3D system. The mobile platform of our system is WIPI, which is the standard mobile platform in Korea, also we adopted NF3D API for our 3D programming API. We implemented real-time fire simulation and added it in mobile 3D game, "Rupee Story".

Key words: Fluid Simulation(유체 시뮬레이션), Realtime Rendering(실시간 렌더링), Mobile 3D Games (모바일 3D 게임), Billboard(빌보드)

1. 서 론

물리기반 시뮬레이션 또는 물리기반 애니메이션은 물리적으로 현실감있는 애니메이션을 생성하기 위하여, 뉴턴 물리학의 법칙을 시뮬레이션에 사용하는 것을 말한다. 일반적으로 물리기반 시뮬레이션은 시뮬레이션 도중 그 형태가 변하지 않는 강체 시뮬레이션(Rigid body simulation)과 형태가 변하는 물체 시뮬레이션(Non-rigid body simulation)으로 분류하며, 형태가 변하는 물체 시뮬레이션 중에는 옷감이나 젤리, 머리카락등과 같이 스프링-댐퍼 모델로 표현할 수 있는 것과 물, 불, 연기, 구름, 강물의 흐름

※ 교신저자(Corresponding Author) : 박동규, 주소 : 경남 창원시 사림동 9번지 창원대학교 정보통신공학과(641-773), 전화 : 055)279-7634, FAX : 055)279-7639, E-mail : dgpark@sarim.changwon.ac.kr
접수일 : 2007년 3월 27일, 완료일 : 2007년 6월 5일
[†] KOG Studios 기술연구소 (E-mail : woosh@kogstudios.com)
[††] 창원대학교 정보시각화 연구실 석사과정 (E-mail : mirina@changwon.ac.kr)
[†††] 종신회원, 창원대학교 정보통신공학과 조교수
※ 본연구는 2006년도 창원대학교 교내연구비의 지원으로 이루어졌음

등 나비에-스토크스 방정식으로 표현하는 모델이 존재한다[1,2]. 이 중에서 유체 시뮬레이션은 물, 불, 연기, 구름 등을 컴퓨터에서 표현하는 분야인데, 유체 시뮬레이션 결과는 영화의 특수효과, 애니메이션, 컴퓨터 그래픽스 분야에서 다양하게 사용되며, 영화 슈렉[3]에서 볼 수 있는 주인공의 몸에서 진흙이 흘러내리는 장면, 컵에 우유를 따르는 장면이 유체 시뮬레이션 응용의 대표적인 예이다[4]. 이미 예전부터 CFD(Computational Fluid Dynamics) 분야에서는 다양한 상황에서 유체의 성질을 사실적으로 시뮬레이션 하기 위한 연구가 수행되었지만, 방대한 계산량으로 그래픽스 분야에 적용되기까지 오랜 시간이 걸렸다. 물리적 사실성에 목적을 둔 CFD와는 달리, 물리적인 정확성 보다는 시각적인 사실성을 우선하여 계산 시간을 줄이는 것이 컴퓨터 그래픽스 분야의 유체 표현 기술의 목표이다[5].

최근 모바일 하드웨어의 발전과 기술적인 진보에 힘입어 휴대 전화의 방식이 1세대의 아날로그 방식에서 디지털 방식으로 전환되고, MP3 플레이어, 디지털 카메라, DMB에 이르기까지 멀티미디어 디지털 장치들과의 컨버전스가 이루어지는 등, 성능과 기능이 빠른 속도로 발전하였다. 하드웨어의 발전과 더불어 콘텐츠 분야에서도 3D 게임과 같이 고사양의 하드웨어와 소프트웨어 기술력이 집적된 고급 콘텐츠가 등장하였다. 시장 규모의 측면에서도 모바일 게임은 PC에 기반한 온라인 게임과 더불어 빠른 성장을 이루게 되었으며, 기술적인 면에서는 PC 플랫폼의 3D 온라인 게임의 모바일 버전이 등장하기도 하였다. 이러한 발전 추세를 미루어 볼 때, 앞으로는 모바일 환경에 적합한 3D 그래픽스 표현 기술과 물리 기반 모델링이 필요할 것이다.

본 논문에서는 지금까지 주로 PC 환경에서 주로 구현되었던 유체 시뮬레이션을 모바일 환경에서 구현하였다. 그리고 기존의 유체 시뮬레이션 중 불과 연기를 표현한 유체 시뮬레이션을 응용하여 모바일 환경에서 불과 연기의 시뮬레이션을 개발하였다. 구현된 시뮬레이션은 그림 1과 같이 기존에 제작된 모바일 3D 게임 "루피 스토리"에 이식하여 성능을 측정하고 게임 캐릭터와의 상호작용을 구현하였다. 이 게임은 캐릭터의 움직임을 키패드로 조작하고 OK 버튼을 입력하여 공격을 가하는 전형적인 일인칭슈팅게임(FPS:First Person Shooting)이다.

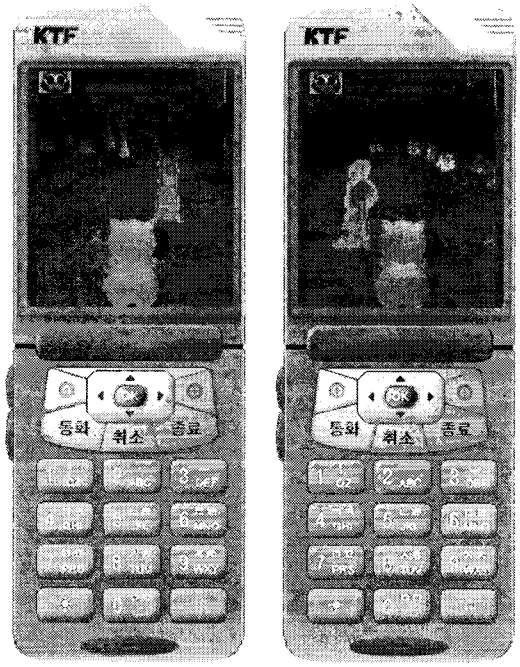


그림 1. 모바일 3D 게임에서의 구현된 불과 연기의 시뮬레이션

본 논문의 시뮬레이션과 응용프로그램은 위피(WIPI)환경에서 구현하였다. 위피환경에서 사용가능한 3D API로는 와우포엠의 NF3D, 리코시스사의 M3D, 고미드사의 G3D등이 있으며, 본 논문의 게임과 시뮬레이션은 와우포엠사의 NF3D 환경에서 구현하였다.

2. 관련 연구

2.1 컴퓨터 그래픽스에서의 유체 표현 기술

물리적으로 유체의 움직임은 연속적인 공간에서 연속된 시간의 흐름에 따라 발생하게 되지만, 컴퓨터 그래픽 시스템에서는 이산화된 공간에서 이산적인 시간간격에 따라 유체의 움직임을 시뮬레이션 하여야 한다[6-8]. 컴퓨터 그래픽스 분야의 유체 시뮬레이션이 본격적으로 시작된 시점은 1997년 Foster의 논문에서 부터이다[2]. Foster의 방법은 나비에-스토크스(Navier-Stokes) 방정식을 이용하는데 이 방정식에서는 유체의 속도장 u를 (ux, uy, uz) 성분으로 각각 분해하여 나비에-스토크스 방정식을 각각에 대하여 이산화하여 수치 적분을 수행하는 방법이다. 이 방법은 안정적인 시뮬레이션을 위하여 매

우 짧은 시간 간격을 사용해야 했기 때문에 계산효율이 낮고 불안정성 문제가 발생하는 단점이 있다. 1996년 Foster의 논문[9]은 처음으로 나비에-스토크스 식을 적용했으나 빠르게 수렴하지 않는 불안정성을 내포하는 등의 단점이 있었다. 이러한 단점 극복하고 나비에-스토크스 방정식을 이용한 유체 시뮬레이션을 실용적으로 수행할 수 있는 방법이 1999년에 발표된 Stam의 논문인 Stable Fluids 기법이다[10]. Stam은 이전의 유체 시뮬레이션 논문에서 보였던 불안정성 문제를 해결하고 좀 더 긴 시간 간격으로 시뮬레이션이 가능한 새로운 방법을 제안하게 된다[10]. Stam은 나비에-스토크스 식을 구성하는 질량 보존 식과 에너지 보존의 식을 헬름홀츠-하지(Helmholtz-Hodge) 분해 방법을 이용하여 속도에 대한 하나의 식으로 표현을 하고 외부 힘, 대류(이동), 확산에 의한 계산된 값의 발산을 막기 위한 투영으로 보정하는 방식으로 시뮬레이션을 수행하였다. Stam의 논문이 가지는 가장 큰 기여는 바로 확산에 의한 해의 발산을 막기 위하여, 외부 힘, 대류(이동), 확산에 의한 계산된 값을 비발산장으로 옮기는 방법을 기술한 것이다. 헬름홀츠-하지 분해는 임의의 벡터장을 그 벡터장의 기저벡터들의 합으로 재구성할 수 있다는 사실에 기초하고 있다. 예를 들어 그리드에 정렬된 벡터 $\vec{v}=(x,y)$는 $\vec{v}=x\hat{i}+y\hat{j}$와 같이 그리드의 축에 정렬된 단위 기저 벡터(unit basis vector)인 $\hat{i}$와 $\hat{j}$로 나타낼 수 있다는 것이며, 마찬가지로 벡터장 역시 발산이 없는 벡터장과 그 스칼라장의 그래디언트로 재구성 할 수 있다는 것이 헬름홀츠-하지 분해의 가장 큰 의미이다[10-12,7]. Stam의 방법이 이전의 방법들에 비해 안정적인 이유는 단순히 이웃한 셀의 값으로부터 현재 셀의 속도를 계산하는 방식이 아닌, 현재 셀에 위치한 파티클을 역추적 하여 이전에 있었던 셀로부터 속도를 가져오는 준 라그랑지 방식을 사용했기 때문이다.

2.2 오일러 방식과 라그랑지 방식의 차이

유체 시뮬레이션의 두 가지 큰 분류로는 그리드 기반 방법인 오일러 방식과 파티클에 기반을 둔 라그랑지 방식이 있다. 오일러 방식은 고정된 계산 노드를 가지는 것이 특징이며, 고정된 공간 내에서 유체가 움직인다. 그리고 고정된 계산 노드는 해당 위치에서 유체가 가질 속도를 저장하고 있다. 라그랑지 방식은 계산 노드가 유체를 따라 이동하며, 따라서 유체를 입자의 집합으로 표현한다. Stam의[10] 기법과 Fedkiw의[11] 기법은 모두 오일러 방식에 해당된다. 그런데 Stam의 방식에 준 라그랑지 기법이 포함되어 있다고 하는 이유는 이류(移流)항의 계산에서 준 라그랑지 입자처럼 계산 노드를 다루기 때문이다. 그 외의 부분에서 Stam의 방식은 오일러 기법이며, 이류항의 계산도 고정된 그리드 각각에 대해 수행된다.

라그랑지 기법을 이용한 유체 시뮬레이션 기법 중에서 대표적인 논문으로는 Desbrun의 1996년 논문[12], Hadap의 2001년 논문 [13], Müller의 2003년 논문 [14]이 있다. 이 방법은 지나치게 많은 메모리를 사용하는 오일러 기법의 단점을 피할 수 있다는 장점이 있으나, 이웃 입자 찾기에 많은 시간을 소비해야 하며, 비압축성을 보장하기 위해 복잡한 코드를 작성해야 한다.

3. 나비에-스토크스 방정식과 Stable Fluid

3.1 나비에-스토크스 방정식

유체의 움직임을 표현하기 위해 전통적으로 사용한 방법은 나비에(Navier)가 1823년 비압축성 점성 유체에 관한 운동 방정식으로 내어 놓은 나비에-스토크스 방정식이다. 나비에-스토크스 방정식은 점성을 가지지 않은 비압축성 유체의 움직임을 기술하는 방정식으로 유체내의 속도장(velocity field)이 u이고, 밀도가 ρ, 압력이 p, 점성계수가 ν, 외부 힘이 f라고 할 때 다음과 같이 표현되는 방정식이다[10].

$$\frac{\partial u}{\partial t}=-(u\cdot\nabla)u-\frac{1}{p}\nabla p+\nu\nabla^2 u+f \tag{1}$$

$$\nabla\cdot u=0 \tag{2}$$

수식 2는 유체의 속도장이 시간에 따라 어떻게 변화하는지를 기술하면서, 동시에 속도장의 발산(divergence)이 0이라는 제한조건을 만족해야 함을 나타낸다. 속도장의 발산이 0이라는 것은 유체가 차지하는 공간내에 새로운 유체의 유입이나 유출이 없다는 것을 의미한다.

수식 1에서는 벡터(vector) 함수 u와 스칼라(scalar) 함수 p가 미지수이며 다음과 같이 정의되는 나블라(nabla) 연산자 ∇의 세 가지 형태가 존재한다.

기울기(gradient) $\nabla u = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \end{pmatrix}$

라플라시안(Laplacian)

$$\Delta u = \nabla \cdot \nabla u = \begin{pmatrix} \frac{\partial^2 u}{\partial x^2} & \frac{\partial^2 u}{\partial y^2} & \frac{\partial^2 u}{\partial z^2} \\ \frac{\partial^2 v}{\partial x^2} & \frac{\partial^2 v}{\partial y^2} & \frac{\partial^2 v}{\partial z^2} \\ \frac{\partial^2 w}{\partial x^2} & \frac{\partial^2 w}{\partial y^2} & \frac{\partial^2 w}{\partial z^2} \end{pmatrix}$$

발산(divergence) $\nabla \cdot u = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$

수식 (1)의 $-(u \cdot \nabla)u$는 유체의 이류(advection 혹은 convection)를 나타내는 수식이다. 이 부분은 유체의 관성을 표현하는 부분으로써 유체가 흐르는 속력과 방향을 지속하게 해준다. $\nu\nabla^2 u$는 유체의 점성을 나타내는 점성(viscosity) 수식이며, ν는 유체의 고유한 점성을 나타내는 점성 상수이다. 이 부분은 유체가 흐르며 서로에게 흐름을 방해하는 작용을 식으로 표현한 것이다. 점성 상수가 클수록 유체는 끈끈한 성질을 띠게 된다. $-\frac{1}{\rho}\nabla p$는 압력(pressure) 항이며 ρ는 압력 상수이다.

마지막 부분의 f는 유체에 미치는 외부 힘을 의미한다. 실제 유체에서는 특별히 영향을 미치는 외부 힘이 없더라도 중력이 f를 통해 항상 영향을 미치고 있다. 외부 힘은 시뮬레이션 할 때 상호작용을 통해 유체를 원하는 방향으로 제어하도록 사용된다.

3.2 Stam의 Stable Fluid 기법

이 기법은 안정적인 나비에-스토크스 방정식 수치 적분 (근사)방법으로 준(準) 라그랑지 (semi-Lagrangian) 기법을 사용하였는데, Stam는 나비에-스토크스 방정식을 구성하는 항들의 의미를 분석하여, 이류(移流)항을 준 라그랑지 기법으로 근사하였다[10]. Stam의 방법은 유체의 흐름을 안정적으로 시뮬레이션 하는 기법으로 이류 이후의 속도장의 최대치가 이류 이전 속도장의 최대치보다 항상 작도록 하여 이류시의 발산 문제를 해결하였다.

Stam 기법의 문제점으로는 유체 내의 소용돌이를 제대로 표현하지 못하고 너무 빨리 안정화되는 결과로 인하여 소용돌이치는 유체의 형태를 표현하기가 어렵다는 점이 있다. 따라서 소용돌이 유폐(Vorticity Confinement)를 추가적으로 다루어 사실성을 높이려는 연구가 이어졌다[11]. Fedkiw의 소용돌이 유폐(Vorticity Confinement) 기법은 유체 내의 소용돌이도를 찾아내고 이를 유지하는 추가의 힘을 가하는 방법이다[11]. 속도장 u에서 소용돌이도 ω는 수식 3과 같이 속도장의 커얼(curl)을 얻는 식에 의하여 구할 수 있다. 이 방법은 유체의 소용돌이를 증폭하는 효과가 있으며, 이를 사용하여 Fedkiw는 Stam의 기법에 비하여 더욱 사실적인 유체 동작을 생성하였다.

$$\omega = \nabla \times u \tag{3}$$

4. 불과 연기 시뮬레이션

4.1 부양력과 냉각 인자

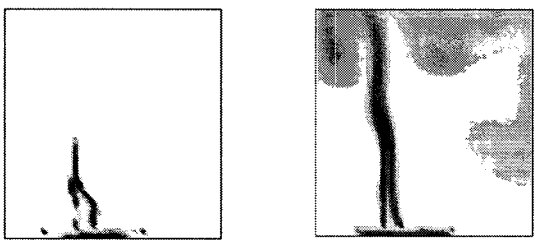
불과 연기의 시뮬레이션을 만들기 위해서는 유체의 애니메이션에 불과 연기의 속성과 색상을 결정하는 기능을 추가해야 한다. 본 논문에서는 우선 Stam이 구현한 안정적인 유체애니메이션 기법에서 사용한 밀도장(density field)을 온도장(temperature field)의 개념으로 바꾸어 불과 연기 시뮬레이션에 사용했다. 불 시뮬레이션에서는 불이 가지는 높은 열과 연기가 이류 확산과정에 의하여 주위의 온도가 낮은 이웃 셀로 확산되며, 이로 인한 외부 힘이 생성되어 유체의 흐름이 발생하게 된다. 본 논문에서는 더욱 사실적인 시뮬레이션을 위하여 위해 부양력과 냉각 인자를 추가하였다. 부양력(buoyancy)은 불의 방향이 아래에서 위로 향하도록 지속적으로 적용되는 외부 힘(external force)의 한 종류이다. 냉각인자(cooling factor)의 영향으로 불에 해당하는 높은 온도의 셀에서 상대적으로 온도가 낮은 이웃의 그리드 셀로 전달될 때, 일정 비율로 점차 낮은 온도가 전달된다[1,2,9].

4.2 불과 연기의 색상

물리적으로 불이 빛을 만드는 이유는 두 가지이다. 첫째로 높은 온도의 가스 때문이다. 이것은 금속이 가열되어 빛을 발하는 원리와 유사하다. 기체가 온도가 상승하면, 더욱 많은 에너지가 방출된다. 상대적으로 온도가 높지 않은 기체는 긴 파장의 빛을 내뿜는다. 온도가 높아짐에 따라 점점 짧은 파장의

빛이 나온다. 파장이 긴 것에서 짧은 순서로 나열하면 빨강, 초록(노랑), 흰색으로 보이는 파랑의 순서이다. 특정 파장의 빛 에너지는 온도 t를 생성하며, 흑체 복사로 알려진 플랑크 에너지 분포식으로 주어진다. 온도가 낮을 경우 매우 긴 파장의 에너지가 나오기 때문에 육안으로는 불빛을 볼 수 없지만, 온도가 높아질수록 열선(적외선)의 세기가 차츰 강해져서 525°C 이상이 되면 육안으로 볼 수 있는 붉은색이 나타나기 시작한다. 이 에너지 분포에 따르면 섭씨 4000도까지 방출하는 빛은 어둡고 대부분이 빨간색 영역이다. 섭씨 5750도에서는 빛을 매우 밝고 푸르스름한 색상을 띄게 된다. 흑체 복사 그래프를 이용하면 어떤 온도에서든지 불꽃의 색상과 밝기를 정확하게 계산할 수 있다. 불이 빛을 내는 다른 이유로는 화학 작용이 있다. 그러나 실제 시뮬레이션에서는 계산량이 너무 많기 때문에 화학적인 반응을 완벽하게 재현할 수 없으며, 내부의 불이 타는 영역을 추적하는 것도 불가능하다.

시뮬레이션에서는 단지 불이 날 때 소모되는 가스의 양과 온도에 의존하여 불이 타는 영역과 내부의 색상을 구할 수 있기 때문에 이러한 시뮬레이션을 그래디언트 기능을 사용하여 간단하게 시뮬레이션 할 수 있다. 그림 2의 (a)는 내부의 온도가 낮고 불이 타는 영역이 비교적 좁은 경우를 그래디언트 기능을 사용하여 시뮬레이션 한 것이며, (b)는 내부의 온도가 높고 불이 타는 영역이 비교적 넓고 연기가 발생한 경우를 시뮬레이션 한 모습이다. 그래디언트 기능은 특정한 범위의 색상을 미리 부여한 후 온도에 따라 색상값을 보간하여 중간 색상을 만드는 방식을 사용한다. 이와 같이 할 경우 선형보간기법 만으로도 다양한 형태와 색상의 불과 연기를 만들 수 있다.



(a) (b)

그림 2. 다양한 화학 반응에 따른 불 시뮬레이션으로 두 개의 서로 다른 그래디언트 색상을 사용하여 표현하였다. (a)와 (b)의 내부와 외부는 동일한 온도이지만 서로 다른 코어 색상을 적용하였다.

본 논문에서는 모바일 환경에서의 불 시뮬레이션을 간단히 하기 위하여 온도에 의해 불의 색상을 결정하였다. 시뮬레이션 공간의 각각의 그리드 셀은 온도 데이터를 가지고 있다. 이러한 온도 데이터는 그래디언트 기능에 의하여 RGB 색상값을 가지게 된다. RGB 색상으로 나타내기 위해 0부터 1사이의 온도 값을 255단계로 사상하였다. RGB 색상을 미리 설정해두고, 온도의 임계치에 따라 연기와 불을 나누어 그려주었다. 불의 색상을 결정하는 함수에서 미리 할당된 색상 사이의 값들을 선형 보간하여 그래디에이션 효과를 구현하였다. 본 논문에서는 모바일 환경의 시뮬레이션 되는 불의 색상은 온도가 높은 곳에서 낮은 곳으로 갈수록 흰색, 노란색, 빨강색으로 변화되도록 구현하였다.

5. 이미지 기반 렌더링

5.1 빌보드

렌즈 섬광 효과를 비롯한 많은 특수 효과들을 만들기 위하여 자주 사용하는 방법은 관측자와 마주하는 다각형을 만들고 이 다각형 위에 하나의 이미지를 렌더링 하는 방법이다. 이와 같이 시야 방향을 기반으로 하여 다각형의 방향을 결정하는 것을 빌보드 기법(billboarding)이라고 하며, 이 때 그 다각형을 빌보드(billboard)라고 한다. 이때 다각형의 방향은 고정되어 있지 않으며, 관측 조건이 변함에 따라 다각형의 방향도 변해야 한다. 알파 텍스쳐 처리와 애니메이션이 결합된 빌보드 기법은 형태가 고정되지 않는 물체나 여러 가지 자연 현상들을 표현하는데 유용하게 사용할 수 있다. 연기, 불, 안개, 폭발, 에너지 장벽, 증기가 뿜어져 나오는 모습 등은 이러한 기술로 표현할 수 있는 몇 가지 객체들의 예이다[15].

화면 정렬 빌보드(screen-aligned billboard)는 빌보드 기법 중에서도 매우 단순한 형태이다. 이것은 2차원 스프라이트와 유사하며 고정된 상향 벡터를 가진다. 월드 중심 빌보드(world-oriented billboard)에는 관측 평면에 정렬되는 빌보드와 시점 지향 빌보드(viewpoint oriented billboard)가 있다. 시점 지향 빌보드는 임포스터에 적합하다. 임포스터 기법은 현재 시점에서 보아 복잡한 물체를 하나의 이미지로 만들어서 이미지 텍스처로 렌더링하는 기법으로 즉석에서 만들어지는 빌보드이다[7].

본 논문에서 사용한 빌보드 기법은 축 빌보드(axial billboard) 기법이다. 주요 원리는 텍스처를 입힌 객체가 관측자와 정면으로 마주하지 않는다. 대신에 보편 공간의 축을 중심으로 회전시킬 수 있고, 허용하는 범위 내에서 최대한 관측자의 방향을 향하도록 조정할 수 있다. 이 기법은 주로 나무를 표현하기 위해 주로 사용한다. 표면 기하 구조를 이용하여 나무를 표시하는 대신에 한 개의 빌보드를 사용한다. 월드 상향 벡터는 나무줄기 방향의 축을 따라 설정된다. 그림 3에서 보는 것과 같이, 나무는 해바라기처럼 관측자의 움직임에 따라 관측자를 지향한다. 이런 유형의 빌보드 기법에서 보편 공간 상향 벡터는 고정되어 있으며, 시점 방향 벡터는 조정 가능하다. NF3D API는 프로그래머가 쉽게 이용할 수 있는 빌보드 API를 제공하고 있으며 본 논문에서는 이를 사용하였다.

5.2 알파블렌딩 기법

시뮬레이션 결과를 렌더링 할 때 빌보드가 3D 공간에서의 배경과 후면의 물체를 가리지 않기 위해서 투명한 속성을 가져야 한다. 이를 위해 알파 블렌딩(alpha blending) 이라는 개념을 사용한다. 유연성 있게 투명 효과를 내기 위해 투명한 물체의 색상을 그 뒤에 있는 물체의 색상과 혼합하는 것이다. NF3D에서는 스프라이트의 메시(mesh) 정보를 얻은 후 이미지의 속성을 설정하는 함수를 통해서 특정 색상을 투명으로 만들 수 있다. 빌보드 전체적인 투명도는 0부터 100까지 파라미터의 조정을 통하여 투명도 조절이 가능하다. 그림 4는 빌보드에 다른 투명도를 설정했을 때의 모습을 보여주고 있다. 그림 4의 (a)는 투명도를 0%로 주었을 때의 시뮬레이션 모습이며, (b)는 불 시뮬레이션을 빌보드에서 수행한 후 이 빌보드의 투명도를 50%로 주어 시뮬레이션 한 모습이다. 단순히 알파블렌딩을 수행할 경우 배경화면과의

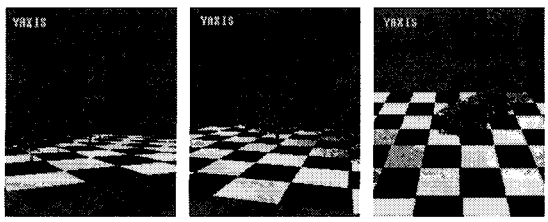
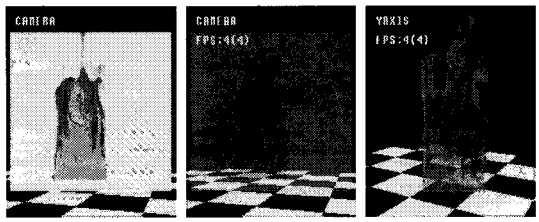


그림 3. NF3D 환경에서 빌보드를 사용한 나무 표현



(a) (b) (c)

그림 4. 빌보드 (a)투명도 0% 설정, (b)투명도 50%, (c)흰색을 제거한 상태의 알파 블렌딩

부조화가 남아 있으나 빌보드의 흰색 부분을 얻어 이를 투명한 색상으로 변경하게 되면 (c)와 같이 배경 이미지와 자연스러운 불 시뮬레이션 결과를 얻을 수 있다.

6. 모바일 환경의 불 애니메이션

6.1 모바일 플랫폼

불 시뮬레이션 결과를 모바일 환경에서 렌더링하기 위하여 WIPI 에뮬레이터와 NF3D SDK를 사용하였다. NF3D는 3D 어플리케이션 개발 환경이며, 모바일 3D 표준 API인 OpenGL ES 스펙을 기반으로 한다. SDK에 포함된 API들은 개발자들이 이용 가능한 3D 라이브러리를 지원한다. 모바일 3D SDK에서 개발된 프로그램은 WIPI 에뮬레이터를 통해 실행이 가능하며, LCD 크기, 키패드, 사운드 등과 같이 게임이나 어플리케이션 제작에 필요한 사항들을 실제 모바일 환경과 동일하게 수행할 수 있는 동작 환경을 제공한다. 본 논문에서 NF3D의 SDK 를 선택한 이유는 기존에 개발된 게임 콘텐츠에 유체 시뮬레이션을 적용하기 위해서이며, 유체 시뮬레이션 자체는 개발 SDK나 특정 플랫폼에 종속되지 않고 구현이 가능하다. 속도 장과 온도 장의 데이터를 저장하기 위해서는 배열이 필요하다. 시뮬레이션을 위해 2차원 배열을 사용하였다. 그리드의 데이터를 저장하는 배열을 만들기 위해서 핸들기반으로 메모리를 사용하였다. 시뮬레이션 공간이 되는 화면 버퍼의 크기는 가로, 세로 각각 128개의 셀로 이루어진 그리드 기반이다.

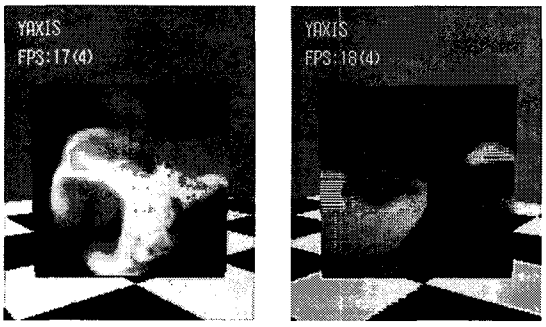
6.2 빌보드상의 시뮬레이션

이미지 기반의 애니메이션을 구현하기 위해서 앞장에서 언급한 빌보드 기법을 사용하였다. 그림 5의

(a)는 빌보드에 속도장과 대류, 확산하는 유체를 표시한 결과이며, (b)는 가로, 세로 각각 64개의 그리드로 구성되어 있는 유체가 존재하는 공간을 나타낸 것이다. 그리드를 구성하는 셀의 개수는 조정 가능하며 셀이 많을수록 렌더링 품질은 향상되며, 연산 비용은 증가한다. NF3D 에서는 빌보드의 회전 속성을 세 가지 형태로 지원한다. 이 경우에는 빌보드가 월드 좌표계의 Y축을 중심으로 카메라의 이동이나 회전에 따라 카메라에 정면이 되도록 회전한다. 단, Y축에 고정되어 있으므로 빌보드 바로 위에 카메라가 위치할 경우에는 사각형 하나로 이루어진 빌보드의 특성으로 인해 시각적인 사실감이 떨어지게 된다.

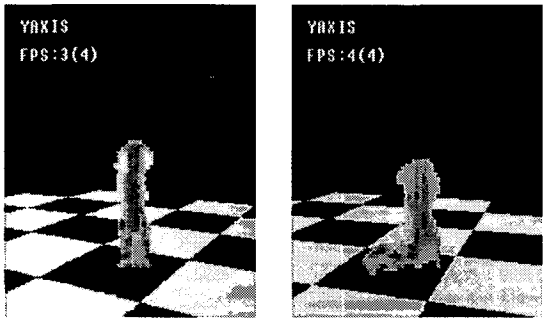
6.3 외부 힘을 사용자 입력으로 설정하기

본 논문에서는 나비에-스토크스 식의 마지막 항인 외부 힘 f 을 사용자가 제어할 수 있도록 하였다. 외부 힘 f 에 중력, 부양력 등을 지속적으로 적용할 수 있고, 부가적으로 사용자가 직접 제어하거나 게임에서의 객체와의 상호작용에 따른 움직임을 조절하는 것도 가능하다[16]. 속도장은 사용자가 모바일 장치의 키패드를 통해 입력한 외부 힘에 따라 변화시킬 수 있다. 변화되는 속도의 크기와 방향을 그림 5과 같이 빨간색 선의 패턴으로 알 수 있다. 빨간 선은 속도장에 있는 각 셀의 중심에서의 속도 벡터의 크기와 방향을 나타낸다. 사용자와의 상호작용을 구현한 후 이를 바탕으로 게임속 캐릭터와의 상호작용을 구현하였다. 불과 연기가 이글거리는 곳에 게임 캐릭터가 접근하게 되면 외부힘이 발생하게 되며 이 외부힘이 빌보드상의 유체에 영향을 주게 된다. 그림 6의 (a)는 유체에 외부힘이 작용하지 않은 경우이며 (b)는 유체에 외부힘이 적용된 결과이다.



(a) (b)

그림 5. 속도장에 따른 유체의 이동 (a) 유체 시뮬레이션 화면 (b) 그리드의 배경셀을 화면에 표시하여 시뮬레이션 한 유체



(a) (b)

그림 6. 외부 힘에 의해 움직이는 불과 연기

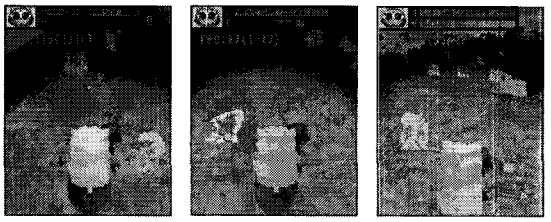


그림 7. 게임 콘텐츠에 적용한 불 시뮬레이션

6.4 시뮬레이션 프레임의 갱신율 조정

본 논문에서 구현한 불 시뮬레이션은 NF3D 기반으로 구현된 일인칭 모바일 3D 슈팅 게임인 "루피스토리"라는 게임 콘텐츠에 적용하였다[17]. 5장 에서 언급한 빌보드 기법을 사용하여 불 애니메이션을 그림 7과 같이 게임 배경의 일부로 배치하였다. 빌보드 갱신율은 게임 상황에 따라서 변경되도록 구현하였다. 빌보드 애니메이션의 갱신율을 조정가능 하도록 만든 이유는 게임 상황에 따라 불 시뮬레이션이 보이지 않거나 멀리서 보이는 경우를 위한 것이다. 만일 불이 게임 캐릭터 주위에 있지 않고 보이지도 않을 경우에는 굳이 시뮬레이션을 수행하여 자원을 낭비할 필요가 없으며, 불이 게임 캐릭터에 대하여 멀리 있을 경우에는 갱신율을 떨어뜨리는 것이 바람직하다. 빌보드를 갱신시키기 위하여 본 논문에서는 NF3D에서 지원하는 API를 사용하여 빌보드를 다시 그리는 주기를 설정하여 갱신율을 조절하였다. 갱신율을 조절하는 기준은 게임 캐릭터와 불의 거리이며 단계별로 나누어서 조절하였다. 게임 캐릭터와 불의

거리가 가까울수록 빌보드 화면은 더 자주 갱신된다. 빌보드의 장면이 새로 그려지는 주기가 짧아질수록 빌보드 자체의 품질은 향상되지만, 게임 전체적인 성능은 약화되므로 빌보드 렌더링의 품질과 시뮬레이션의 실시간 특성 사이의 상충되는 점을 고려하여 갱신율을 결정하였다. NF3D API는 초당 프레임 갱신율을 화면에 표시하는 API를 제공하는데 이를 바탕으로 손쉽게 시뮬레이션 상황을 파악할 수 있다.

6.5 게임과의 상호작용

본 논문에서 구현한 불 시뮬레이션은 게임에 등장하는 주인공 객체와의 상호작용을 고려하여 게임에 적용되었다. 빌보드와 근접한 곳에서의 게임 캐릭터의 움직임이 불 시뮬레이션에 영향을 주게 되어, 불의 움직임에 변화를 만들어낸다. 즉 게임의 객체와 빌보드 사이의 거리가 설정해 놓은 기준보다 가까워지면 그리드의 정해진 셀에 특정 방향으로의 힘을 적용한다. 적용된 힘은 속도장에 영향을 미치고 결과적으로 불의 움직임이 발생한다. 객체와 빌보드의 거리는 3차원 벡터로 저장된 (X, Y, Z) 좌표 정보를 이용하여 벡터의 차를 구하는 함수를 사용했다. 게임 캐릭터와 빌보드의 자연스러운 상호작용을 위해서는 빌보드의 초당 프레임 수가 높아야 하지만 빌보드 프레임 수의 증가는 게임 성능에 부하를 주기 때문에 실험을 통해 적절한 값을 찾아야 한다.

6.6 초당 프레임 갱신율

불 애니메이션을 위한 빌보드 갱신율에 따른 게임 FPS(frame per second)의 변화는 6~24 fps 로 나타났다. 빌보드 애니메이션의 갱신 속도를 빠르게 할수록 게임의 속도가 느려지고, 화면의 갱신 속도는 낮아진다. 그리고 시뮬레이션 공간이 되는 그리드의 수, 즉 배열의 크기를 조정해도 FPS 에 영향을 미친다. 기존에 개발된 모바일 3D게임에 불 시뮬레이션을 위하여 서로 다른 크기의 배열로 설정하여 실험한 결과, 빌보드를 새로 그려주는 시간 간격에 따른 초당 프레임 수를 그림 8에 나타내었다. 실험 결과에 의하면 96x96 크기의 그리드를 사용할 경우 초당 10 프레임 이상의 시뮬레이션이 가능하므로 실시간 게임에 사용하기에 어려움이 없는 것으로 나타났다. 모바일 게임의 특성상 디스플레이의 크기가 320×240으로 제한되어 있으므로 96x96 크기의 그리드를 사용할 경우 게임의 품질에도 영향을 주지 않는 것으로 나타났다.

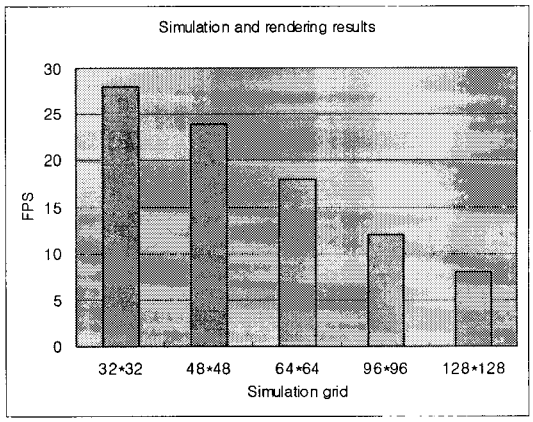


그림 8. 시뮬레이션 그리드의 크기를 변경시켰을 때의 프레임 갱신비율(1024Kb의 힙 메모리를 사용함)

7. 결론 및 향후 연구 과제

7.1 결 론

본 논문은 불 시뮬레이션을 모바일 환경에서 실시간으로 구현하였으며, 이러한 시뮬레이션이 게임에 등장하는 캐릭터와 상호작용하도록 하였다. 불 시뮬레이션을 구현하기 위해서는 Stam이 제안한 안정적인 유체 시뮬레이션 기법을 응용하였으며, 시뮬레이션 결과는 빌보드 기법을 사용하여 애니메이션으로 나타내었다. 독립적인 불 시뮬레이션의 실행 결과는 실시간성을 만족하였고, 게임 콘텐츠에 적용했을 때에도 만족하는지 시험하였다. 이를 위해서 불 시뮬레이션을 모바일 3D 게임에 적용하여 게임과의 상호작용을 구현하였다. 게임에서 불 시뮬레이션의 비중을 높인 결과 게임의 속도와 FPS에 영향을 미치게 되었으나, 불 시뮬레이션의 렌더링 품질과 속도의 비중을 조절하여 실시간성과 상호작용을 모두 만족할 수 있었다.

7.2 향후 연구 과제

불 시뮬레이션의 공간이 2차원이기 때문에 3차원 게임에 적용했을 시 입체감이 부족한 문제가 있다. 이를 해결하기 위에서는 2차원 빌보드를 여러 각도

에서 본 모습으로 다양한 장면을 만들어 시뮬레이션 하는 방법이 필요하다. 하지만 이 경우 실시간 시뮬레이션을 수행하기에는 시스템 자원을 과도하게 소모하는 단점이 있다. 그리고 그리드 기반의 시뮬레이션의 특성상 제한된 공간에서만 유체 시뮬레이션이 가능하다. 이러한 특징으로 인해 빌보드를 게임에 적용했을 때, 빌보드와 3D 공간의 뚜렷한 경계로 인해 불, 연기의 흐름이 끊기는 문제가 발생할 수 있다. 빌보드는 정해진 개수의 그리드로 이루어진 한정된 크기의 평면이기 때문에 연기가 그리드의 범위를 벗어나서 확산되는 모습을 표현하는데 있어서 공간적인 한계가 있다. 화면상에 연속적으로 표현해야 할 불, 연기가 3D 공간과 단절되어 나타날 수 있기 때문이다. 이 문제를 해결하기 위해서 빌보드의 그리드 크기를 확장하거나 불, 연기가 확산되는 정도 등을 제어하여 그리드 범위 내에서 불 시뮬레이션 결과를 그려야 한다. 또한 모바일 장치의 성능향상이 점차 향상됨에 따라 불과 연기뿐만 아니라 물, 구름등과 같은 다양한 3차원 유체 시뮬레이션 기법과 렌더링에 관한 연구가 필요하며, 그리드 위에서 시뮬레이션 하는 방식과 함께 파티클 기반의 시뮬레이션에 대한 연구 역시 계속되어야 할 것이다.

참 고 문 헌

[1] J. Stam and E. Fiume, "Turbulent Wind Fields for Gaseous Phenomena," *In Proc. of SIGGRAPH '93*, pp. 369-376, Aug. 1993.

[2] N. Foster and D. Metaxas, "Modeling the Motion of Hot, Turbulent Gas," *In Proc. of SIGGRAPH '97*, pp. 181-188, 1997.

[3] Shrek, http://www.shrek.com/.

[4] D. Enright, S. Marschner, and R. Fedkiw, "Animation and Rendering of Complex Water Surfaces," *In Proc. Of SIGGRAPH2002*, pp. 736-744, July 2002.

[5] 표순형, 구본기, "CG 유체 표현 기술 동향," 전자통신동향분석 제 20권 제4호, pp. 29-41, Aug. 2005.

[6] J. Stam. "Real-Time Fluid Dynamics for Games," *In Proc. of the Game Developer Conference*, Mar. 2003.

[7] M. J. Harris, *Real-Time Cloud Simulation and Rendering*, Ph.D Dissertation, Chapel Hill, 2003.

[8] Rick Parent, *Computer Animation, Algorithms and Techniques*, Morgan Kaufmann, 2002.

[9] N. Foster and D. Metaxas, "Realistic Animation of Liquids," *Graphical Models and Image Proc.*, Vol. 58, No. 5, pp. 471-483, 1996.

[10] J. Stam. "Stable Fluids," *In Proc. of SIGGRAPH '99*, pp. 121-128, 1999.

[11] R. Fedkiw, J. Stam, and H. Jensen. "Visual Simulation of Smoke," *In Proc. of ACM SIGGRAPH 2001*, pp. 15-22, 2001.

[12] M. Desbrun and C. Marie-Paule. "Smoothed particles: A new paradigm for animating highly deformable bodies," *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, pp. 61-76. 1996.

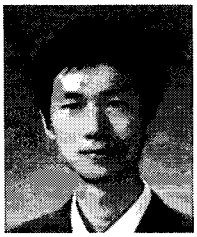
[13] S. Hadap and N. Magnenat-Thalmann, "Modelling Dynamic Hair as a Continuum," *Computer Graphics Forum*, Vol. 20, No. 3, pp. 329-338, 2001.

[14] M. Müller, D. Charypar, and M. Gross, "Particle-BasedFluid Simulation for Interactive Applications," *In Proceedings of ACM SIGGRAPH Symposium on Computer Animation (SCA) 2003,* pp. 154-159, 2003.

[15] T. Akenine-Mőller and E. Haines, *Real-Time Rendering, 2nd Edition*, AK Peters, 2002.

[16] J. Stam. "Interacting with Smoke and Fire in Real-Time," *Communications of the ACM*, Volume 43, Issue 7, pp. 76-83, 2000.

[17] 우상혁, 박보영, 조미리나, 박동규, "모바일 환경의 3D 아케이드 게임 구현," 한국멀티미디어학회 2006년도 춘계학술 발표대회논문집, 제9권 제1호, pp. 190-193, 2006.



우 상 혁

2005년 창원대학교 정보통신공학과(공학사)
2007년 창원대학교 정보통신공학과(공학석사)
2007년~현재 KOG Studios 기술연구소
관심분야 : 모바일 콘텐츠 개발, 컴퓨터 게임, 컴퓨터그래픽스

조미리나

2007년 창원대학교 정보통신공학과(공학사)
2007년~현재 창원대학교 정보시각화 연구실 석사과정
관심분야 : 모바일 콘텐츠 개발, 물리기반모델링, 컴퓨터그래픽스

박 동 규

1993년 부산대학교 전자계산학과(이학사)
1996년 부산대학교 전자계산학과(이학석사)
1999년 부산대학교 전자계산학과(이학박사)
2000년~2002년 영산대학교 멀티미디어 공학과 전임강사
2002년~현재 창원대학교 정보통신공학과 조교수
관심분야 : 모바일 스캔툴, 물리기반모델링, 컴퓨터 그래픽스