유체 역학에 기반을 둔 애니메이션 효과†

서강대학교 강병권 · 차득현
김장희 · 임인성*

1. 서 론

컴퓨터 애니메이션 분야에서 물, 불, 폭발, 연기, 가스 등과 같은 자연현상의 사실적인 표현은 수준 높은 애니메이션의 제작에 있어 매우 중요한 사항 중의 하나이다. 이러한 유체 애니메이션 기법은 최근 영화 '슈렉'이나 '개미'등의 3차원 애니메이션에서 이미 사용된 바 있으며(그림 1), 사실적인 유체의 표현을 통해 영화의 사실성을 한 차원 높이는 데 큰 역할을 하였다. 사실적인 유체 흐름의 표현을 위해서는 다양한 수치 해석 기법을 통한 물리 기반 애니메이션 기법의 적용이 필요하며, 최근 전산 유체 역학 분야에서 오랜 기간 연구되온 유체 역학 기법들을 컴퓨터 애니메이션 제작에 적용하려는 연구들이 활발히 이루어지고 있다.

그림 1 영화 '슈렉'에서의 유체 효과 생성 예

일반적으로 이러한 물리 기반 유체 애니메이션 기법들은 주로 정확한 유체 해석에 주안점을 두고 있기 때문에 종종 계산량이 방대할 뿐만 아니라, 애니메이터가 의도하는 대로 유체의 흐름을 제어하기 어렵기 때문에 컴퓨터 그래픽스 분야에 적용하기가 용이하지 않았다. 따라서 컴퓨터의 계산 능력이 현재보다 낮았던 과거에는 물과 같은 유체의 표현을 위해 이러한 물리 기반의 애니메이션 기법을 사용하기보다는, 수작업에 의존을 하거나, 여러 사인 곡선의 합성을 통한 웨이브 곡면을 이용한 기법이나 유체 표면의 높이지도(height map)를 이용한 기법들을 통해 유체와 비슷한 모양을 만들어내는 방법을 사용하곤 하였다[1,2,3]. 그러나 이러한 기법들은 종종 유체의 표현에 유용하게 사용이 되었지만, 그 사용 범위가 제한이 된다는 단점을 가지고 있다.

전산 유체 역학 분야에는 오랜 연구를 통하여 상당히 다양한 형태의 유체의 흐름을 해석해 주는 효과적인 이론과 풀이기법들이 존재하나, 비로소 최근에서야 이러한 결과들이 컴퓨터 애니메이션 제작에 활용되기 시작하였다. 최근 N. Foster등은 유체 역학에서 사용되는 대표적인 공식중의 하나인 Navier-Stokes 공식을 애니메이션 제작에 활용하기 용이한 형태로 풀어내는 기법을 제안하여 보다 사실적인 유체의 움직임을 생성하였으며[4,5,6], 이러한 유체 역학 기반 기법들을 영화 '슈렉'등에 성공적으로 적용하였다. 또한 J. Stam은 semi-Lagrangian 기법을 이용하여 기존 방법의 불안정성을 해결하고, 계산량을 줄여서 유체의 대화식 제어가 가능하도록 하였다[8]. 이러한 유체 애니메이션 기법들은 지속적으로 발전되고 있으며, 유체 역학에 기반을 둔 유체 흐름의 효과적인 생성뿐만 아니라 컴퓨터 그래픽스 관점에서 중요한 문제들, 즉 유체의 렌더링이나 직관적인 조절 등의 문제에 대하여 활발한 연구가 진행이 되고 있다[7,9,16,17,18].

유체 역학 이론에 기반을 두고 유체 특수 효과를

---

* 종신회원

† 본 연구는 한국과학재단 목적기초연구(R01-2002-000-00512-0)지원으로 수행되었습니다.

생성하는데 있어 유의할 점 중의 하나는 유체 역학과 컴퓨터 애니메이션 두 분야의 관점의 차이이다. 즉 유체 역학 이론에 기반을 둔 자연현상의 시뮬레이션은 주로 주어진 상황(초기 조건, 주변 환경 등)에 대하여 유체가 어떻게 반응하고 흘러갈 것인가를 물리학적으로 정확하게 해석하는 것을 목표로 한다. 반면에 컴퓨터 애니메이션에서는 유체가 상황에 따라 자연스럽게 흘러가는 것을 계산해 내는 것도 유용하게 쓰일 수 있지만, 그보다 더 중요한 것은 애니메이터가 의도하는 방식으로, 즉 각본에 써 있는 대로, 유체가 흘러가도록 해줄 수 있어야 한다는 점이다. 설사 물리학적으로 약간 부정확할지라도, 애니메이터가 원하는 대로 유체를 조절할 수 있다면 이는 유체 계산에 있어 정확도보다 더 중요한 인자가 될 수가 있다. 아직 이 두 가지 요소를 모두 만족시켜 주는 유체 애니메이션 기법은 별로 존재하지 않으며[17,19], 이러한 관점에서 유체 역학 이론과 컴퓨터 애니메이션의 효과적인 접합을 위해서 앞으로 많은 문제를 해결해야 할 것으로 예상된다.

본 논문은 최근 컴퓨터 애니메이션 분야에 활발히 적용되고 있는 물리 기반 유체 애니메이션 기법을 소개하는 것을 목적으로 한다. 이를 위하여 본 서강대학교 컴퓨터 그래픽스 연구실에서 개발 중인 유체 특수효과 생성 소프트웨어를 예를 들어 설명을 하고자 한다. 특히 물, 진흙, 우유 등과 같은 유체와 가스, 연기 등의 유체 등 두 가지 부류의 유체 생성에 적합한 기법에 대하여 간략히 소개하도록 한다.

본 연구실에서 개발하고 있는 소프트웨어는 크게 유체의 흐름을 정의하는 편미분 방정식을 풀어주는 수치 엔진과 이러한 수치 계산의 결과로부터 사실적인 영상을 생성해 주는 렌더링 엔진의 두 부분으로 구분되어 있다. 우선 2장에서는 전산 유체 역학 분야에서 사용되는 대표적인 공식인 Navier-Stokes 공식에 기반을 둔 수치 엔진의 구현에 사용된 방법에 대하여 설명하고, 3장에서 광선 추적법 및 등가면 추출법, 그리고 포톤매핑 기법 등에 기반을 둔 렌더링 엔진에 대하여 설명한다. 4장에서는 본 소프트웨어를 통하여 생성한 유체 애니메이션 결과를 보이고, 마지막으로 5장에서 결론을 맺는다.

2. Navier-Stokes 공식을 이용한 수치 엔진

Navier-Stokes 공식은 전산 유체 역학 분야에서 유체의 흐름을 계산하기 위하여 사용되는 대표적인 공식이다. Navier-Stokes 공식은 크게 아래 식 (1)과 (2)와 같이 두 부분으로 나눌 수 있다. 식 (1)은 질량 보존의 법칙에 관한 식으로, 유체가 어떤 식으로 흐르던 그 질량은 보존된다는 것에 대한 식이다. 식 (2)는 운동량 보존의 법칙을 통해 유체의 속도를 계산하는 부분이다.

$$\nabla \cdot \vec{U} = 0 \tag{1}$$

$$\vec{U}_t = \nu \nabla \cdot (\nabla \vec{U}) - (\vec{U} \cdot \nabla)\vec{U} - \frac{1}{\rho}\nabla P + g \tag{2}$$

여기서 $\vec{U}$는 속도를 나타내는 벡터이고 ∇는 그래디언트 연산자이다. $\nu \nabla \cdot (\nabla \vec{U})$는 유체의 점성을 계산하는 부분이고 $(\vec{U} \cdot \nabla)\vec{U}$는 유체의 대류 현상을, $-\frac{1}{\rho}\nabla P$는 압력 보정을 나타내는 부분이다. 마지막으로 g는 외부 힘을 나타내는 부분으로 유체에 가해지는 외부의 힘이 유체의 흐름에 미치는 영향을 고려한다. 매 프레임마다 이러한 공식을 통해 유체의 흐름의 속도를 계산하고 이렇게 구해진 속도장을 통해 유체의 모양을 렌더링할 수 있다. 이러한 Navier-Stokes 공식은 원래 연속적인 공간에서 정의되어 있지만, 컴퓨터 애니메이션에서 연속적인 공간에 대한 계산은 불가능하므로, 이러한 공간을 이산적인 격자점으로 이루어진 볼륨 데이터 형태의 공간으로 재구성하여 계산한다. 본 논문에서는 Navier-Stokes 공식을 이용한 유체 애니메이션을 크게 물, 진흙, 우유와 같은 액체의 경우에 적합한 방법과 연기, 가스와 같은 기체에 대해 적합한 방법으로 구분하였다. 두 방법은 약간의 방법상의 차이점은 있으나, 기본적으로 Navier-Stokes 공식을 따르기 때문에 전체적으로 큰 차이점은 없다고 할 수 있다.

2.1 물, 진흙, 우유와 같은 유체에 대한 Navier-Stokes 공식의 풀이

N. Foster 등은 Navier-Stokes 공식을 다음과 같이 3차원에 대해 풀어서 물, 진흙, 그리고 우유 등과 같은 유체에 대한 애니메이션을 구현하였다[4,6]. 이 기법에서 위의 Navier-Stokes 공식의 식 (2)는 위에서 언급한 바와 같이 다음의 세 가지 부분으로 구분할 수 있으며, 각 부분은 독립적으로 계산이 수행된다.

2.1.1 점성 부분(viscosity term)

이 부분은 물, 우유, 진흙과 같은 액체의 성격을 띠는 유체가 가지는 점성을 나타내기 위한 부분으로서, 중앙 차분(central difference) 방법을 사용하여 이산화할 수 있다.

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{h^2}(T(x+h) - 2T(x) + T(x-h)) + O(h^2) \tag{3}$$

$$\nu\nabla \cdot (\nabla \vec{U}) = \nu\nabla \cdot \begin{pmatrix} \frac{\partial \vec{U}}{\partial x} \\ \frac{\partial \vec{U}}{\partial y} \\ \frac{\partial \vec{U}}{\partial z} \end{pmatrix} = \nu \begin{pmatrix} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \\ \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \\ \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \end{pmatrix} \tag{4}$$

식 (3)은 중앙 차분 방법을 반복적으로 적용하여 이차 미분을 근사한 것이고, 식 (4)는 식 (3)을 통해 점성 부분을 정리한 것이다. 식 (3)을 이용해 식 (4)를 근사화 하면 $O(h^2)$의 오차로 계산이 가능하다.

2.1.2 대류 부분(convection term)

이 부분은 method of characteristic 방법을 사용하여 계산할 수 있다. Method of characteristic 방법은 유체의 흐름을 나타내는 궤적 위에서의 속도는 모두 동일하다는 것을 이용한 방법으로서, 유체는 이전의 흐름을 계속 유지하려는 성질을 이용하는 방법이다. 따라서 이것을 풀기 위해서는 현재의 지점에서 과거의 속도를 가져와야 하는데, 그렇게 하기 위해서는 현재의 지점에서 과거의 시점으로 역으로 추적하며 따라가 추적된 지점에서의 속도를 가져오면 된다. 본 수치 엔진에서는 가급적 정확한 계산을 위하여 4차 룬지-쿠타(Runge-Kutta) 방법을 사용하였다.

2.1.3 압력 보정

이 부분은 8개의 격자점으로 정의되는 셀(cell)의 여섯 면을 통해 흘러 들어오고 나가는 유체의 흐름의 차이로부터 생기는 압력차를 라플라스 연산자(Laplacian operator)를 적용하여 생성되는 선형 방정식을 풀어서 보정하는 부분이다.

$$\nabla^2 p = \frac{\rho\nabla \cdot \vec{U}}{\Delta t} \tag{5}$$

$$\sum_{n=ijk} (p_{n+1} + p_{n-1}) - 6p = \rho\frac{\Delta\tau}{\Delta t} \sum_{n=ijk} (u_{n+1} - u_n) \tag{6}$$

식 (5)를 각 셀들의 중앙에서 적용하여 이산화한 식이 (6)이다. 여기서의 $\nabla^2 p$는 압력의 지역적인(spatial) 변화를 뜻하고 $\vec{U}$는 식 (2)를 압력 부분 전까지 계산한 속도장을 뜻한다. 그리고 $n\pm1$의 의미는 현재의 셀 옆에 있는 셀(i, j, k 방향으로)을 의미한다. $\Delta\tau$는 각 셀의 크기를 뜻하는데, 여기서는 공간을 이산화할 때 사용된 셀이 i, j, k 각 방향에 대해서 같은 크기를 갖는 정육면체라고 가정한다. 그리고 u_{n+1}과 u_n은 셀을 둘러싸고 있는 여섯 면의 속도 중 하나는 원점에 가까운 것, 하나는 또 다른 것(원점에서 먼 것)의 속도를 뜻한다. 이것을 적용하면 미지수인 압력 벡터에 대하여 AP = B와 같은 형태의 선형 방정식을 얻을 수 있다. 이 선형식을 통해 얻어진 각 셀의 압력을 통해 다음 식 (7)과 같이 속도를 보정한다. 이렇게 보정된 속도장은 위의 식 (1)에서와 같이 질량 보존의 법칙을 만족하게 된다.

$$u_{ijk}^{t+\Delta t} = u_{ijk} - \frac{\Delta t}{\rho\Delta\tau}(p_n - p_{n-1}) \tag{7}$$

2.2 연기, 가스와 같은 기체에 대한 Navier-Stokes 공식의 풀이

유체의 한 형태라고 할 수 있는 연기나 가스 역시 Navier-Stokes 공식을 풀어냄으로서 우리가 원하는 움직임을 계산해 낼 수 있다. 점성 부분과 대류 부분은 물을 시뮬레이션 하는 부분과 동일한 방법으로 풀어내고 있으나, 압력 부분의 계산에 있어서 다른 방법을 사용하고 있다[8]. 또한 물과는 달리 연기의 시뮬레이션은 몇 가지 추가적인 요소를 포함한다. 먼저 연기는 밀도의 움직임이므로 Navier-Stokes 공식에 의해 계산된 속도장 외에 밀도장의 흐름을 시뮬레이션 하는 것이 필요하다. 연기에 영향을 미치는 추가적인 힘들도 계산이 되어야 하는데, 그 첫 번째는 기체의 온도에 따른 부양력에 의한 힘이고 두 번째는 연기 내부의 소용돌이 운동 상태를 나타내는 힘이다.

연기의 속도장을 계산하기 위해서 Navier-Stokes 공식과 질량보존의 법칙에 대한 식을 결합한 다음과 같은 식을 사용한다.

$$\frac{\partial u}{\partial t} = P(-(u \cdot \nabla)u + \nu\nabla^2 u + f) \tag{8}$$

위의 식에 대한 전체적인 계산 방법은 식의 각

부분을 계산하는데 있어서 먼저 풀어낸 부분이 다음에 계산될 부분의 입력으로 사용됨으로서 다음 프레임을 위한 속도장을 계산해 낸다.

$$W_0(x) \xrightarrow{add\ force} W_1(x) \xrightarrow{convect} W_2(x) \xrightarrow{diffuse} W_3(x) \xrightarrow{project} W_4(x) \tag{9}$$

위 식 (9)에서의 *add force, convect, diffuse* 부분은 물에 대한 수치 엔진에서와 동일한 수치적 방법에 의해 계산된다. 마지막 단계인 *project* 부분은 현재까지 계산된 속도가 가지고 있는 오차를 보정하여 질량보존의 법칙을 만족하는 최종 속도장(divergence free field)을 계산해 내며 물에 대한 엔진에서의 압력 보정부분과 동일한 의미를 갖는다. 실제로 *project* 부분은 식 (8)에서의 투영(projection) 연산자에 P에 의해 계산된다.

또한 다음의 변형된 Navier-Stokes 공식을 이용하여 속도장의 계산과 유사한 방법으로 연기 밀도의 흐름을 계산한다. 이때의 수치적 계산은 속도장의 계산을 위한 방법과 유사하며 그 식은 다음과 같다.

$$\frac{\partial a}{\partial t} = -u \cdot \nabla a + k_a \nabla^2 a - \alpha_a a + S_a \tag{10}$$

diffuse 부분에 의해 구해진 속도장은 Helmholtz-Hodge 분해에 의해 무발산장(divergence-free field)인 u와 스칼라장 q로 다음과 같이 분해가 된다.

$$w = u + \nabla q \quad (11) \quad \longrightarrow \quad \nabla \cdot w = \nabla^2 q \quad (12)$$

식 (12)를 계산함으로서 q를 계산할 수 있고 식 (11)에 대입하여 u를 구할 수 있다. Poisson 공식인 식 (12)는 중앙 차분을 통하여 공간적으로 이산화 할 수 있으며, 결국 희소 선형(sparse linear) 계의 형태로 표현된다. 이를 풀어냄으로서 q를 구한다. 다음 식에 의해 투영 연산자 P의 의미를 이해할 수 있다.

$$u = Pw = w - \nabla q \tag{13}$$

연기의 흐름에 영향을 미치는 다른 요소들은 외부 힘으로서 수치 엔진에 적용된다. 각 프레임마다 적용되는 이 힘을 계산함으로서 이 요소들의 영향을 반영하고 제어할 수 있다. 본 논문에서 구현한 수치 엔진에서는 연기가 가지고 있는 온도에 의해 발생하는 부양력과 연기 내부의 소용돌이 상태를 표현해 줄 수 있는 힘을 계산한다. 부양력(식 (14))과 소용돌이 힘(식 (15))을 나타내는 식은 다음과 같다.

$$f_{bouy} = -\alpha p z + \beta (T - T_{amb}) z \tag{14}$$

$$f_{conf} = \epsilon h \left(\frac{\nabla \nabla \times u}{\nabla \nabla \times u} \times (\nabla \times u) \right) \tag{15}$$

부양력은 단순한 계산으로 구할 수 있으며 소용돌이 힘은 공간적인 이산화 기법을 이용하여 이산화 함으로서 풀어 낼 수 있다. 이때 부양력을 계산하기 위해 밀도가 가지고 있는 온도 역시 그 흐름을 추적해 주어야 한다. 이것은 Navier-Stokes 공식의 *convect* 부분과 같이 method of characteristic 방법으로 구할 수 있다.

3. 유체 렌더링 엔진

본 논문에서 기술하고 있는 유체 애니메이션 기법은 물리 기반의 수치 엔진을 통한 사실적인 장면을 생성하는 것을 목표로 하고 있다. 따라서 수치 엔진에서 계산된 유체의 데이터를 사실적으로 가시화해 줄 수 있는 렌더링 엔진이 요구된다. 본 연구실에서 개발하고 있는 유체 렌더링 엔진은 파티클 시스템 및 포톤 매핑 기법에 기반을 둔 광선 추적법을 기본으로 하고 있다.

3.1 물, 진흙, 우유 등과 같은 유체 렌더링

3.1.1 파티클 시스템에 의한 유체 렌더링

본 논문에서는 물, 진흙, 그리고 우유와 같은 형태의 유체의 흐름을 표현하기 위해서 파티클 시스템을 이용한 등가면 추출법으로 렌더링을 수행하였다. 유체의 흐름을 나타내기 위한 파티클은 앞서 수치 엔진에서 계산된 속도장에 의해 매 프레임마다 그 위치를 새로 계산하게 되며 이러한 파티클들이 유체의 형태를 나타내는 기준이 된다. 모든 파티클이 일정한 크기를 가지며, 각 파티클의 속도는 수치 엔진에서 계산된 속도장의 값을 삼선형 보간하여 이 속도를 통해 그 위치를 새로 계산하게 된다. 이렇게 다시 계산된 파티클들의 위치에서 유체의 표면의 모양을 등가면 추출법에 의하여 만들어낼 수 있다. 한 지점 x_P에 위치한 파티클에 대한 유체의

표면은 다음과 같이 얻을 수 있다.

$$\Phi_p(x) = \sqrt{(x_i - x_{pi})^2 + (x_j - x_{pj})^2 + (x_k - x_{pk})^2} - r \tag{16}$$

위 식(16)에서 r은 파티클의 반지름을 나타내며, 이 파티클 x_P에서 유체의 표면은 $\Phi_P(x) = 0$에 해당하는 지점이 된다. 따라서 전체적인 유체의 표면은 모든 격자점에서의 값을 그 지점에서 가장 가까운 파티클 x_P에 대한 $\Phi_P(x)$의 함수 값으로 갖는 볼륨 데이터를 구성하고 이 볼륨 데이터에 대하여 Marching Cubes[10]와 같은 등가면 추출법을 수행하여 다면체 모델로 구해낼 수 있다.

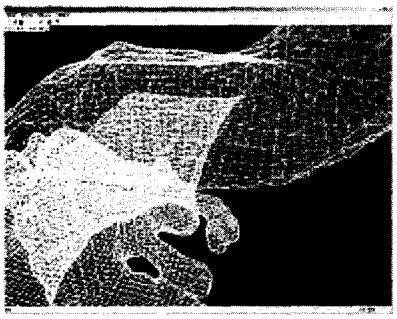


그림 2 등가면 추출법을 통해 얻어진 유체 표면 다면체 데이타

3.1.2 Level Set 기법에 의한 유체 렌더링

위에서 설명한 파티클을 이용한 등가면 추출법은 반지름이 일정한 구의 모양을 이용하여 등가면을 생성하기 때문에, 최종적으로 유체를 생성했을 때, 표면이 부드럽지 못하고 울퉁불퉁한 모양을 만들어 낸다는 문제점이 있다. 이를 해결하기 위해 level set 기법을 사용한다. level set 기법은 유체의 표면으로부터 떨어져 있는 거리를 signed distance 함수를 이용해 공간에 정의를 하여 유체를 표현한다[11,12]. 이렇게 표현된 signed distance 정보를 이용하여 위의 경우와 마찬가지로 등가면 추출법을 이용하여 유체 표면을 생성할 수 있다. Level set 기법으로 정의되는 signed distance field ϕ의 시간에 따른 변화는 다음의 식으로 표현할 수 있다.

$$\phi_t + \vec{u} \cdot \nabla\phi = 0 \tag{17}$$

여기서 $\vec{U}$는 해당 지점에서의 속도를 의미한다. 시간에 따라 식 (17)을 계속 진행해 가다 보면 유지되어야 하는 signed distance의 성질이 깨지게 된다. 이 성질이 깨진 상태에서 (17)을 사용하면 각 지점에서의 법선 벡터를 정상적으로 얻을 수 없기 때문에 오차가 계속 누적되어 우리가 원하는 결과를 얻을 수 없게 된다. 따라서 이를 보정해 주어 수치적으로 안정한 상태를 만들어 주어야 하는데 이를 재초기화(re-initialization)이라 한다. 재초기화를 위해서 다음과 같은 형태의 식을 사용한다.

$$\phi_\tau + sgn(\phi_0)(\nabla\phi - 1) = 0 \tag{18}$$

여기서 ϕ_τ는 임의의 time step에 대한 ϕ의 증분을 뜻하고 $sgn(\phi_0)$는 다음과 같다.

$$sgn(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + (\Delta x)^2}} \tag{19}$$

여기서 ϕ_0는 재초기화를 하기 전의 ϕ값이다. 하지만 이 방법은 유체의 세세한 특징까지 표현하지 못하고, 종종 유체 형태를 과도하게 부드럽게 만들어버리는 경향이 있다. 또 시간에 따른 유체의 움직임을 이 방법으로 추적하면 유체의 부피를 조금씩 잃어버려 계산상의 정확성이 떨어진다는 문제도 발생한다. 따라서 이 두 방법의 문제점을 보완하기 위해 기본적으로 level set 기법을 사용하고, 여기에 파티클 시스템을 결합하는 방법을 사용하였다[7]. 보완하는 방법은 일단 시간에 따른 유체 움직임의 추적을 두 방법을 이용해 독립적으로 수행한다. 이후에 파티클이 표현하는 유체를 이용해 level set 기법이 잃어버린 유체의 세세한 특징이나 잃어버린 부피를 찾아준다. 이는 공간상에 정의된 signed distance를 직접 수정하면서 수행한다. 이렇게 하면 level set 기법의 단점을 극복할 수 있고 보다 정확히 유체의 흐름을 추적할 수 있게 된다.

3.2 연기, 가스 등과 같은 유체 렌더링

물이나 진흙, 우유와 같은 표면이 강조되는 유체와는 달리 연기, 가스 등의 밀도장으로 표현되는 유체는 좀 더 복잡한 렌더링 기법을 필요로 한다. 연기, 가스와 같은 기체의 렌더링은 유체의 투명도와 빛의 분산까지 고려하여 렌더링이 되어야 하며, 표

면이 강조되지 않는다는 점에서 물이나 진흙과 같은 유체와는 구분된다. 따라서 본 논문에서 구현한 렌더링 엔진에서는 이러한 기체의 특성을 잘 나타낼 수 있는 렌더링 기법으로 forward ray-marching 기법과 포톤 매핑 기법을 이용한 방법을 사용하였다[9,13].

3.2.1 Forward Ray-Marching 알고리즘

기존의 광선 추적법이 물체의 표면에서 쉐이딩 계산이 이루어졌던 것과 달리, 연기나 가스의 경우 표면뿐만이 아니라 물질 내부에서도 쉐이딩 계산이 이루어져야 한다. 이와 같은 물질을 렌더링하기 위하여 기존의 광선추적법과는 다른 렌더링 기법이 필요하며, 이러한 렌더링 기법을 ray-marching 기법이라 한다(그림 3).

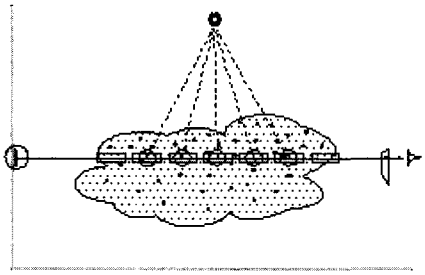


그림 3 Ray-Marching 기법의 구조

본 논문에서 구현한 렌더링 엔진에서는 연기나 가스와 같은 물질의 특성상 누적된 불투명도 값에 따라 효과적인 조기 광선 종료가 가능한 forward ray-marching 알고리즘을 이용하였다(식 (20))[9].

$$L_n(x_n, \vec{w}) = L_{n-1}(x_{n-1}, \vec{w}) + e^{-\tau(x_{n-1})\Delta x_n(\vec{w}\cdot\nabla)} L_s(x'_n, \vec{w}) \tag{20}$$

3.2.2 포톤 매핑 기법

포톤 매핑(photon mapping)이란 효과적인 전역 조명(global illumination)을 계산하기 위하여 개발된 기법으로서 기존의 광선 추적법이나 래디오시티 기법 등에서 구현하기 어려웠던 완전한 전역 조명 모델을 효과적으로 나타낼 수 있는 방법이다[13]. 우선 각 광원에서 퍼져 나오는 빛의 입자를 포톤(photon)이라 하고, 전체 장면에서 이 포톤이 부딪히는 부분이 광원으로부터 빛이 도달하는 부분으로 보고, 그 지점의 밝기를 높여주는 기법이다. 각 포톤들이 진행하는 경로에 대한 계산은 광선 추적법에서와 같은 방법으로 계산할 수 있으며, 물체와의 교차점에 대한 계산도 같은 방법으로 수행된다. 한 포톤이 물체에 부딪혔을 경우, 그 포톤이 반사되거나 그 자리에 흡수되는 두 가지의 경우를 Russian-Roulette 방법으로 결정하게 된다. 이와 같이 각 광원에 방출된 포톤들의 경로를 추적하고 포톤들이 흡수된 위치 정보를 저장한 것을 포톤 맵(photon map)이라 한다.

이러한 과정을 통해 포톤 맵을 구성한 후에 광선 추적법을 이용하여 전경을 렌더링하게 된다. 이 때, 각 포톤들에 의한 간접 조명을 계산하기 위해 쉐이딩 계산이 일어나는 지점에서의 포톤의 밀도를 구하게 되는데, 해당 지점에서부터 가까운 순서대로 n개의 포톤을 찾아낸 다음, 각 포톤의 광도를 계산하여 그 포톤들이 위치한 지역의 넓이로 나누어 해당 지역에서의 광도를 계산하게 된다((그림 4), 식 (21)). 따라서 한 지점에서 가까운 n개의 포톤을 찾는 작업을 효율적으로 처리하기 위하여, 포톤을 저장할 때 K-D 트리 혹은 보로노이 다이어그램(Voronoi diagram)과 같은 공간 분할 자료구조를 이용하여 저장하게 된다. 최적화된 balanced K-D 트리의 경우 $O(\log N)$의 탐색시간 안에 가장 가까운 지점을 찾아낼 수 있으며, k개의 가까운 포톤을 찾아내는 데에 평균적으로 $O(k+\log N)$의 탐색 시간을 갖는다.

$$L_r(x, \vec{w}) \approx \frac{1}{\pi r^2}\sum_{p=1}^{N} f_r(x, \vec{w_p}, \vec{w})\Delta\Phi_p(x, \vec{w_p}) \tag{21}$$

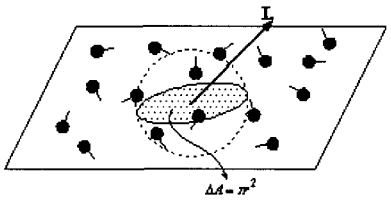


그림 4 한 지점에서의 포톤의 밀도를 통한 조명 계산

3.2.3 밀도장 데이터에 대한 포톤 매핑

일반적인 물체에 대한 포톤 매핑기법은 광원에서 방출된 포톤이 물체의 표면에 부딪힐 때마다 그 물질의 특성에 따른 작용이 발생한다. 그러나 연기나

가스와 같은 밀도장으로 이루어져 있는 유체의 경우에는 물체의 표면에서 반응이 일어나는 것이 아니라, 밀도장 내부에서 밀도 값에 따라 반응이 일어나게 된다. 따라서 이러한 밀도장 데이터에 대해서는 포톤의 경로도 다음과 같은 다른 방법으로 추적하게 된다.

하나의 포톤이 밀도장 안으로 들어오게 되면, 매번 그 물질과 반응을 하여 포톤 맵에 저장되거나 혹은 반응하지 않고 그 방향으로 계속 진행하는 두 가지의 경우를 선택하게 된다. 두 가지 경우는 다음과 같은 누적 밀도확률함수 $F(x)$에 의해 결정된다.

$$F(x) = 1 - \tau(x_s, x) = 1 - e^{-\int_{x_s}^{x} \kappa(\xi) d\xi} \quad (22)$$

$\tau(x_s, x)$는 포톤이 물질에 들어온 시작 지점부터 현재 지점까지 누적된 불투명도가 되며, 각 포톤은 위의 확률함수를 통해 Russian-Roulette 방법으로 물질과 반응하는지 여부를 결정하게 된다. 포톤이 물질과 반응하는 경우에는 그 물질에 부딪혀 산란되는 경우와 그 위치에서 흡수되는 두 가지 경우가 발생하게 되는데, 두 가지 경우는 그 물질의 알베도(albedo) 값에 대한 확률함수로 결정되게 된다.

포톤이 그 물질과 반응하여 산란될 때 일어나는 난반사의 방향은 빛이 분산되는 방향에 대한 위상함수(phase function)에 의해 결정된다. 물체의 표면에서 난반사가 일어나는 현상을 나타내는 BRDF (Bidirectional Reflectance Distribution Function)에서와 달리 이러한 위상 함수는 그 지점을 둘러싼 구에 해당되는 범위 안에서 반사되는 방향을 결정하게 된다. 본 유체 렌더링 엔진에서는 ray-marching 알고리즘에 적용이 쉽고 구현이 간단한 Henyey-Greenstein 위상 함수를 사용하였다(식 (23)).

$$p(\theta) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g\cos\theta)^{1.5}} \quad (23)$$

이렇게 구성된 포톤 맵과 앞서 기술한 ray-marching 기법을 통해 밀도장의 렌더링을 수행할 수 있다.

4. 구현 결과

지금까지 본 논문에서 기술한 수치 엔진과 렌더링 엔진을 통해 사실적인 유체의 흐름을 시뮬레이션하고 사실적인 이미지를 렌더링할 수 있다. 그림 5-8은 본 연구실에서 개발한 유체 애니메이션 소프트웨어를 통하여 생성한 결과를 보여주고 있다.

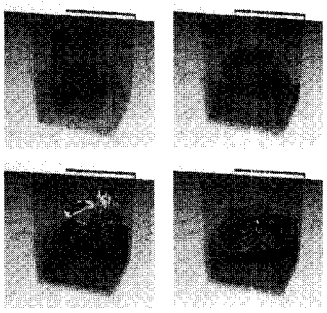


그림 5 쏟아지는 물의 애니메이션 예

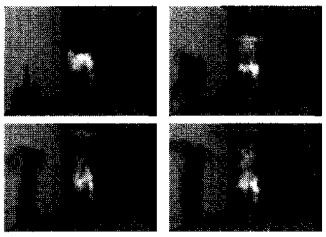


그림 6 상자를 타고 올라가는 연기 애니메이션 예

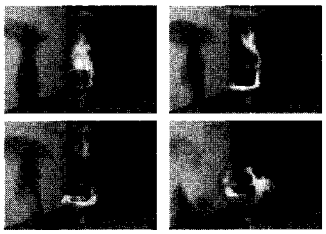


그림 7 상자를 타고 올라가는 도중에 바람에 날리는 연기 애니메이션 예

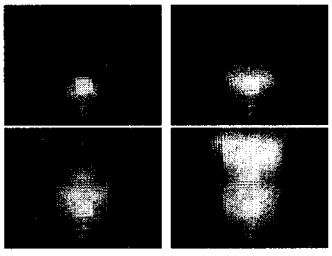


그림 8 연기 애니메이션에 대한 속도장 변화 예

5. 결 론

지금까지 사실적인 유체 애니메이션을 위한 Navier-Stokes 공식에 기반을 둔 물리 기반 유체 애니메이션 기법에 대하여 소개하였다. 이러한 기법은 3차원 애니메이션이나 3차원 게임 등에 현실감 있는 장면을 구성하기 위해서 필수적이나 실제 적용하기에 많은 어려움이 있는 것이 사실이다. 실제로 이러한 유체 애니메이션이 효과적으로 사용된 영화 '슈렉'에서는 영화 제작 과정 중에 진흙과 맥주와 같은 유체를 표현하는 작업이 가장 어려웠던 작업 중의 하나였다고 할 만큼 도전적인 작업이었다. 이러한 애니메이션 과정을 용이하게 하기 위해서는 전기한 바와 같이 애니메이터가 원하는 방식으로 유체를 조절할 수 있도록 컴퓨터 애니메이션 기법과 유체 역학의 기법을 효과적으로 결합해 주는 방법들이 개발되어야 할 것이다.

본 논문에서 구현한 수치 엔진과 렌더링 엔진을 이용하여 물, 진흙, 우유 등과 같은 액체의 특성을 갖는 유체와 연기, 가스와 같은 기체의 특성을 갖는 유체에 대하여 사실감 있는 애니메이션을 구현할 수 있다. 오프라인 애니메이션에서는 계산 시간이 큰 문제는 아니나, 아직도 계산량이 실시간 애니메이션을 하기에는 많다고 할 수 있다. 이러한 유체 애니메이션 계산 과정을 최근 발표되고 있는 PC용 그래픽스 가속기의 쉐이더 기능을 통하여 최적화하는 것은 향후 수년간 중요한 연구 주제로 대두가 될 것이다.

참고문헌

[1] Fournier, A. and Reeves, W.T., "A Simple Model of Ocean Waves," ACM SIGGRAPH '86, pp. 75-84, 1986.

[2] Peachy, D., "Modeling Waves and Surf," ACM SIGGRAPH '86, pp. 65-74, 1986.

[3] Kass, M. and Miller, G., "Rapid, Stable Fluid Dynamics for Computer Graphics," ACM SIGGRAPH '90, pp. 49-57, 1990.

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

[5] Foster, N. and Metaxas, D., "Modeling the Motion of a Hot Turbulent Gas," ACM SIGGRAPH '97, pp. 181-188, 1997.

[6] Foster, N. and Fedkiw, R., "Practical Animation of Liquids," ACM SIGGRAPH 2001, pp. 23-30, 2001.

[7] Enright, D., Marschner, S. and Fedkiw, R., "Animation and Rendering of Complex Water Surfaces," ACM SIGGRAPH 2002, pp. 736-744, 2002.

[8] Stam, J., "Stable Fluids," ACM SIGGRAPH '99, pp. 121-128, 1999.

[9] Fedkiw, R., Stam, J. and Jensen, H.W.,"Visual Simulation of Smoke," ACM SIGGRAPH 2001, pp. 23-30, 2001.

[10] Lorensen, W.E. and Cline, H.E., "Marching Cubes: a High Resolution 3D Surface Reconstruction Algorithm," ACM SIGGRAPH '87, pp. 163-169, 1987.

[11] Sethian, J., Level Set Methods and Fast Marching Methods, Cambridge University Press, 1999.

[12] Osher, S., and Fedkiw, R., The Level Set Method and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2002.

[13] Jensen, H.W., Realistic Image Synthesis Using Photon Mapping, AK Peters, 2001.

[14] Jensen, H.W., Suykens, F., and Christensen, P., A Practical Guide to Global Illumination Using Photon Mapping, ACM SIGGRAPH 2001 Course Note #38, 2001.

[15] Sande, L.R., Shrek: The Story Behind the Screen, ACM SIGGRAPH 2001 Course Note #19, 2001.

[16] Rasmussen, N., Nguyen, D., Geiger, W., and Fedkiw, R.F., "Smoke Simulation for Large Scale Phenomena," ACM SIGGRAPH 2003, 2003(To appear).

[17] Treuille, A., McNamara, A., Popovic, Z., and Stam, J., "Keyframe Control of Smoke Simulations," ACM SIGGRAPH 2003, 2003(To appear).

[18] Stam, J., "Flows on Surfaces of Arbitrary Topology," ACM SIGGRAPH 2003, 2003(To appear).

[19] Foster, N., and Metaxas, D., "Controlling Fluid Animation," Computer Graphics International '97, pp. 178-188, 1997.

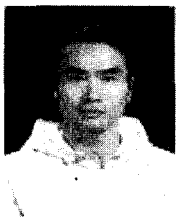
강 병 권

1999. 2 서강대학교 컴퓨터학과 공학사
2001. 2 서강대학교 컴퓨터학과 공학석사
2001. 3~현재 서강대학교 컴퓨터학과 박사과정
관심분야 : Real-Time Rendering, Volume Rendering, Liquid Animation
E-mail : wormkbk@grmanet.sogang.ac.kr

차 득 현

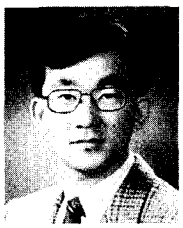
2002. 8 서강대학교 컴퓨터학과 공학사
2002. 8~현재 서강대학교 컴퓨터학과 석사과정
관심분야 : Liquid Animation, Level-Set Method
E-mail : seaboy7@grmanet.sogang.ac.kr

김 장 희



2003. 2 서강대학교 컴퓨터학과 공학사
2003. 2~현재 서강대학교 컴퓨터학과 석사과정
관심분야 : Liquid Animation, Level-Set Method
E-mail : rolldice@grmanet.sogang.ac.kr

임 인 성



1985. 2 서울대학교 계산통계학과 이학사
1987. 5 Rutgers-The State University of New Jersey 컴퓨터학과 이학석사
1991. 7 Purdue University 컴퓨터학과 이학박사
1993. 3~현재 서강대학교 컴퓨터학과 교수
관심분야 : Computer Graphics, Scientific Visualization, High Performance Computing
E-mail : ihm@mail.sogang.ac.kr