시스템 성능에 독립적인 실시간 물 시뮬레이션 방법

*오동수[0] **김동신 ***이종원 *유관우[†]

*경북대학교 컴퓨터 공학과, **영진전문대학교, ***KOG Soft Inc
tentmakerdongsu@hanmail.net eastgod@yjc.ac.kr wonlee@kogsoft.com kwryu@knu.ac.kr

A real-time water simulation method independent of the system performance

*Dong-su Oh[0] **Dong-shin Kim ***Won Lee *Kwan-woo Ryu[†]
*Dept. of Computer Engineering, Kyungpook National University
**Yeungjin Junior College
***KOG Soft Inc

요 약

그래픽스 분야에서 물의 동작을 사실적으로 표현하는 것은 컴퓨터 애니메이션이나, 게임, 영화의 특수 효과로써 중요한 분야이다. 그러나 사실적인 물의 움직임을 모델링 하기 위해서 물리 기반 모델을 사용할 경우 상당히 많은 계산 시간이 소요된다. 같은 크기로 모델링 된 물 시뮬레이터를 성능이 다른 컴퓨터에서 동작시키면 프레임 수가 달라진다. 성능이 높은 시스템에서는 계산속도가 빠르므로 프레임수가 많이 나오며, 낮은 시스템에서는 계산 속도가 느리므로 프레임 수가 작게 나온다. 본 논문에서는 애니메이션이나 게임과 같은 분야에서 컴퓨터의 성능에 따라 물 모델 크기를 사용자의 요구에 맞도록 적절히 조절함으로써 실시간에 적용이 가능한 물 시뮬레이션 시스템을 제안한다.

1. 서 론

영화나, 컴퓨터 게임 분야에서 물의 동작을 사실적으로 표현하는 것이 중요시되고 있다. 바다 물의 동작을 표현하기 위한 연구가 컴퓨터 애니메이션 초기에 많이 진행되었다[1][2]. 이러한 모델에서는 실제적인 물의 파장(wave)이 가지는 물리적인 성질보다는 매개함수(parametric functions)를 사용하여 나타낸다. 이 방법들은 실제와 비슷한 장면들을 만들어 내지만, 물리 수식에 기반을 둔 모델이 아니다.

최근에 computational fluid dynamics(CFD)에 기초한 물리 모델을 통해서 물의 동작을 표현하는데 많은 연구들이 진행되고 있으며, 보다 자연에 가까운 물의 동작을 만들어 낸다[3]. 그러나 이런 모델은 물리 기반 모델이므로 계산량이 많아, 실시간 애니메이션이나 게임에 적용하기 어려운 한계점을 가지고 있다. 게임과 같은 분야에서 물의 동작을 표현하기 위해서는 적절한 프레임 수(fps: frame per second)가 보장되어야 한다.

본 논문에서는 첫 번째 O'Brien이 제안한 물 모델을 실시간 애니메이션이나 게임 등과 같은 곳에서 적용이 가능하도록 계산 과정을 개선하여 시뮬레이션 하였으며, 안정적인 결과를 얻을 수 있었다[4]. 즉, 볼륨을 보존하기 위해서 여러 번의 반복 계산을 하지 않고 한번에 계산을 수행할 수 있도록 함으로써 계산량을 줄였다. 두 번째 실시간 게임이나 애니메이션 분야에서 적절한 프레임 수를 보장하기 위해서, 시스템의 성능에 따라 물 모델의 크기(grid size)를 조절함으로써 사용자가 원하는 속도를 보장해주는 실시간 물 시뮬레이션 방법을 제안한다. 즉, 속도가 빠른 시스템에서는 물 모델의 크기를 크게 사용함으로써, 더욱 사실적으로 보이도록 하는 반면 속도가 느린 시스템에서는 물 모델의 크기를 줄임으로써 적절한 프레임 수를 얻는다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련연구를 살펴본다. 3장에서는 전체 시스템 구성도 와 물 모델에 대해서 알아본다. 4장에서는 초당 프레임 수에 맞는 물 모델 크기(grid size)에 대한 실험결과를 제시한다. 5장에서는 결론 및 향후 연구를 기술한다.

† 「이 논문은 2002년도 경북대학교특성화사업팀(KNURT) 연구비에 의하여 연구되었음」

2. 관련 연구

Darwyn Peachey는 사인함수(sinusoidal functions)를 사용해서 파장을 모델링 했다[1]. 해변으로 접근하는 파도와 해변에서 부서지는 파도의 애니메이션과 랜더링에 적당한 모델이다. 이 모델에서는 파도가 부서지거나 다른 물체와 충돌할 때 일어나는 물보라(spray) 등을 모델링 하기 위해서 입자 시스템(Particle system)을 사용한다.

Kass는 Navier-Stokes equations를 간략화 해서 2차원의 얕은 물(2D shallow water)에 사용되는 단순화된 수식을 만들었으며, 물 표면의 높이필드(Height filed)를 계산하는 방법이다[5]. 이 방법은 실시간에 물 동작을 표현할 수 있지만, 부서지거나 부딪히는 물의 동작을 표현할 수 없으며, 외부물체와 상호작용도 표현할 수 없다.

O'Brien은 튀기는 물(splashing fluid)의 동적인 시뮬레이션 방법을 도입하여 외부의 물체와 상호 작용하는 물의 동작을 모델링 했다[4].

Foster는 더욱 현실적인 물의 동작을 표현하기 위해 처음으로 3차원 Navier-Stokes equations의 이산화를 통해서 3차원(three-dimension) 시뮬레이터를 개발했다[3]. 이 시뮬레이터는 많은 물의 동작을 표현할 수 있다. 특히, 물의 굴절, 반사, 회절, 부서지거나, 부딪히는 물, 외부물체와의 상호작용 등과 같은 물의 많은 동작을 표현할 수 있다.

Fedkiw는 물의 애니메이션에 대한 일반적인 방법을 제공한다[6]. 이 논문에서는 물과 오브젝트와의 상호작용이 많이 개선되었으며, 물의 동작을 컨트롤 커브(control curve)를 통해서 제어 할 수 있다. 또한 Navier-Stoke equations에 기초한 기술로서, 아주 높은 현실적인 물의 동작을 생성할 수 있다.

3. Water Simulation

3. 1 시스템 구성

물의 물리 기반 시뮬레이션에서는 계산량이 많으므로 컴퓨터의 성능에 많이 의존적이다. 계산량에 비해 성능이 떨어지면,

실시간 게임과 같은 곳에서 원하는 프레임 수가 나올 수 없다. 본 논문에서는 이와 같은 문제점을 해결하기 위해 컴퓨터 성능에 따라 물 모델의 크기(grid size)를 조절함으로써 사용자가 요구하는 초당 프레임 수(fps)를 보장하는 시뮬레이션 시스템을 제안한다. 초당 프레임 수는 소요된 시간(elapsed time)의 역수로 아래와 같이 구할 수 있다.

$$fps = \frac{1}{elapsed\ time} \quad (1)$$

전체 시스템의 알고리즘은 아래 표 1과 같다.

표 1. 전체 시스템의 알고리즘 흐름도

Ⅰ. 목표 프레임 수(target_frame)를 입력받는다.
Ⅱ. 현재 프레임 수(current_frame)를 얻는다.
Ⅲ. 현재 프레임 수과 목표 프레임 수간의 차를 계산한다. different = target_frame - current_frame
Ⅳ. 차이의 절대값을 threshold 값과 비교한다. | different | < threshold
Ⅴ. 차이의 절대값이 grid_size보다 큰 값이면 grid_size를 조절해서, 다시 Ⅱ로 돌아간다.
Ⅵ. 차이의 절대값이 grid_size보다 작으면, 그때의 current_frame을 가지고 시스템에서 사용하게 된다.

본 시스템에서, 물 모델의 크기(size), 물의 가로 세로 길이는 초기화 할 때 각각에 대해서, 64, 64, 64 값으로 설정하였다. 입력함수에서 target_fame을 입력받으며, target_frame은 사용자가 원하는 초당 프레임 수이다. current_frame은 현재의 시스템의 프레임 수를 나타낸다. threshold 값은 현재의 시스템의 프레임 수와 사용자가 원하는 목표 프레임 수간의 허용 오차를 말한다. 프레임 차이가 허용오차 보다 큰 경우는 양수 값이면, 물 모델 크기(grid size)를 줄임으로써 프레임 수를 높일 수 있고, 음수 값이면 물 모델 크기(grid size)를 높임으로써 프레임 수를 조절할 수 있다. 위의 과정을 반복함으로써 사용자가 요구하는 적절한 프레임 수를 얻을 수 있다.

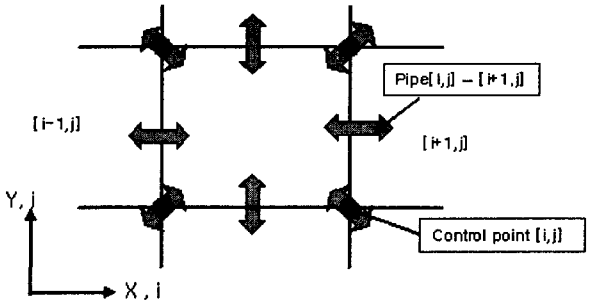

그림 1. 각각의 수직칼럼과 8개의 방향으로 연결된 가상 파이프 및 사각 격자로 나누어진 물 모델[4].

3.2 Water Simulation Model

본 논문에서 구현한 물의 기본 모델은 O'Brien이 제안한 모델을 사용했으며, 이 모델은 3가지 요소로 구성된 모델이다.[3]: Volume Model, Surface Model, Spray Model. 본 논문에서는 Volume Model 과 Surface Model을 사용해서, 물의 동작을 구현하였다. O'Brien이 제안한 모델에서 실시간 애니메이션이나 게임 등에 적용이 가능하도록 하기 위해서 물 동작을 구현할 때 O'Brien이 제안한 모델의 계산 과정을 개선해서 시뮬레이션 하였으며, 안정적인 결과를 얻을 수 있었다.

3.2.1 Volume Model

물의 몸체(main body)는 부피(volume)를 모델링 하기 위해서 제안되었으며, 몸체는 연결된 칼럼(column)의 사각 격자(rectilinear gird)로 나누어진다. 이들 칼럼사이의 흐름(flow)은 인접한 칼럼을 연결하는 가상의 파이프를 통해서 이루어진다(그림1). 파이프 안의 흐름을 결정하는 방정식은 유체역학의 물리 법칙으로부터 유도된다. 격자 안 칼럼의 정적압력 계산에서 중력에 의한 압력뿐만 아니라, 외부의 물체에 의한 충격을 시뮬레이션 하기 위해 외부압력(E_{ij})에 관한 요소가 추가되어서, 칼럼[i,j]에서의 전체 총 압력(P_{ij})은 아래의 수식(2)으로 쓸 수 있다[4.]

$$P_{ij} = h_{ij}\rho g + p_0 + E_{ij} \quad (2)$$

위의 수식에서 ρ는 유체의 밀도이고, g는 중력가속도이고, h_{ij}는 칼럼[i,j]에서의 높이며, P_0는 대기압력이다.

유체의 흐르는 속도는 시스템 안에 있는 각 파이프를 통해서 이루어진다. 각 시간 간격에서 힘(force)이 파이프 안의 유체의 흐름을 가속시킨다. 칼럼[i,j]와 8개의 이웃 중($[k,l] \in \eta_{ij}$) 하나의 파이프와의 압력차이($\triangle P_{ij\to kl}$)를 통해서 힘이 결정된다. 따라서 파이프 안에서 유체의 흐르는 가속도($a_{ij\to kl}$)는 수식(3)으로 계산할 수 있다[4].

$$a_{ij->kl} = \frac{\rho g(h_{ij} - h_{kl}) + E_{ij} - E_{kl}}{\rho l} \quad (3)$$

위의 수식에서 l은 파이프의 길이를 의미한다. 일정 시간 간격($\triangle t$) 동안 파이프 면적이(c)인 파이프 안의 유체의 가속도가 상수라고 가정하면 파이프 안의 흐름($Q_{ij\to kl}$)은 수식(4)와 같이 쓸 수 있다[4].

$$Q^{t+\Delta t}_{ij->kl} = Q^{t}_{ij->kl} + \Delta t(ca_{ij->kl}) \quad (4)$$

그리고 일정 시간간격 동안 전체 칼럼의 볼륨변화($\triangle V_{ij}$)를 수식(5)으로 쓸 수 있다[4].

$$\Delta V_{ij} = \Delta t \sum_{kl\in\eta_{ij}} \left[\frac{Q^{t+\Delta t}_{ij->kl} + Q^{t}_{ij->kl}}{2} \right] \quad (5)$$

시스템 안의 부피를 보존하기 위해서는 2가지 조건이 시스템 안에서 만족되어야 한다. 첫 번째는 비 압축성 유체이기 때문에 파이프의 한쪽 끝의 흐름의 크기와 다른 쪽 끝의 흐름의 크기는 같고, 부호는 반대이다. 수식(6)으로 쓸 수 있다[4].

$$\forall ij : \forall kl \in \eta_{ij} : Q_{ij->kl} = -Q_{kl->ij} \quad (6)$$

두 번째는 부피가 음수가 되는 것은 물리적 의미에서 맞지 않으므로, 모든 칼럼 안의 부피는 양수로 유지되어야한다. 수식(7)으로 쓸 수 있다[4].

$$V^{t+\Delta t}_{ij} \geq 0 \Leftrightarrow V^{t}_{ij} \geq -\Delta V^{t}_{ij} \quad (7)$$

계산 중에 칼럼의 부피가 음수가 될 수 있으며, 이러한 경우 위의 두 가지 조건을 통해서 격자 안의 모든 칼럼이 양의 부피를 가질 때까지 계산을 반복한다.

O'Brien이 제안한 방법은 수식(6)과 수식(7)이 만족되지 않으면 만족 할 때까지 반복한다. 이러한 반복은 실시간 물 시뮬레이션에 적합하지 않다. 따라서 본 시스템에서는 실시간에 물의 동작을 표현하기 위해서 수식(6)에서는 파이프의 양쪽 끝의 절대치가 다를 경우 계속해서 반복하지 않고 threshold($\varepsilon > 0$)를 정해서 그 값보다 크면, 그때의 $Q_{ij\to kl}$, $Q_{kl\to ij}$, 값을 '0'으

로 설정했으며, 아래의 수식(8)을 통해서 계산 할 수 있다.

$$| Q_{ij \to kl} + Q_{kl \to ij} | > \varepsilon \quad (8)$$

여기에서 ε값을 계속해서 줄여 감으로써 발산하지 않고 안정적인 결과를 실시간에 얻을 수 있다.

또한 실시간 처리를 위해서 수식(7)을 모든 칼럼에서 양의 부피를 가질 때까지 반복하지 않았으며, 아래의 수식(9)과 같이 하여서 계산할 수 있다.

$$\text{if}(V_{ij}^{t} + \Delta V_{ij}^{t} < 0) \{ \Delta V_{ij} = Q_{ij \to kl} = Q_{kl \to ij} = 0; \} \quad (9)$$

이때의 ΔV_{ij}의 값과 8개 방향의 파이프의 흐름 $Q_{kl \to ij}$, $Q_{ij \to kl}([k,l] \in \eta_{ij})$의 값을 '0'으로 계산함으로써, 여러 번의 반복계산을 피할 수 있으며, 한번만에 안정적인 시뮬레이션 결과들을 얻을 수 있다.

3.2.2 Surface Model

표면모델(surface model)은 외부 물체가 시스템과 상호작용하게 한다. 즉, 표면에 충돌하거나 떠 있는 물체는 표면모델에 힘을 가하게 된다. 이와 같은 외부 압력이 부피모델에 전달되며, 표면의 위치는 각 칼럼의 부피 모델에 의해서 결정된다. 표면모델의 격자 점(grid point)은 부피모델의 제어 점(control point)과 일대일 대응이 이루어진다.

격자 점(grid point)의 수직위치(Z_{ij})는 격자 점을 둘러싼 4개의 칼럼의 높이를 평균함으로써 구할 수 있으며, 수식(10)으로 쓸 수 있다[4].

$$Z_{ij} = \frac{h_{ij} + h_{i,j+1} + h_{i+1,j} + h_{i+1,j+1}}{4} \quad (10)$$

각 제어 점(control point)에 작용하는 힘은 외부 압력으로 변환된다. 따라서 외부압력은 그 제어 점과 연결된 칼럼에 적용되어 수식(11)으로 쓸 수 있다[4].

$$E_{ij} = -\frac{f_e}{4dxdy} \quad (11)$$

위의 수식에서 dx, dy는 칼럼의 가로 세로 길이를 의미한다.

4. 실험결과

본 논문은 펜티엄Ⅲ 850 와 펜티엄Ⅲ 450 에서 Directx SDK를 사용해서 C++로 구현하였다. 두 개의 시스템에서 VGA CARD (GeForce-Ⅱ)와 RAM(320M)을 같게 설정했다. 아래의 표2는 두 개의 시스템에서 fps와 격자 크기를 비교한 것이다.

표 2. fps에 따른 두 시스템간 격자 크기 비교

grid size / fps	펜티엄 Ⅲ 450	펜티엄 Ⅲ 850
100	30	38
90	32	40
80	34	42
70	37	44
60	40	48
50	42	52
40	48	58
30	55	65
20	69	81
10	96	113

실험의 초기설정에서 격자 크기를 64로 해서 비교를 수행했다. 보는 바와 같이 두 개의 시스템에서 프레임 수에 따른 격자 크기의 차이가 평균 8-10정도의 차이고, 프레임수의 차이는 10프레임이상 이다. 실험결과를 통해서 실시간 컴퓨터 애니메이션이나 게임과 같은 곳에서 10프레임 이상 차이는 상당히 큰 것이다. 따라서 컴퓨터 성능에 따라 격자 크기를 조절함으로써 사용자가 요구하는 프레임 수를 얻을 수 있다. 그림2, 그림3은 펜티엄 Ⅲ 850으로 격자 크기 64일 때의 물 모델의 시뮬레이션 결과이다.

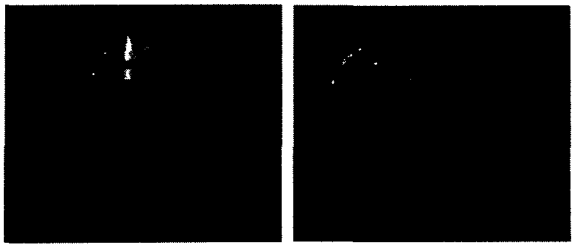

그림2. 텍스쳐 맵핑된 물 모델의 실험한 결과 예.

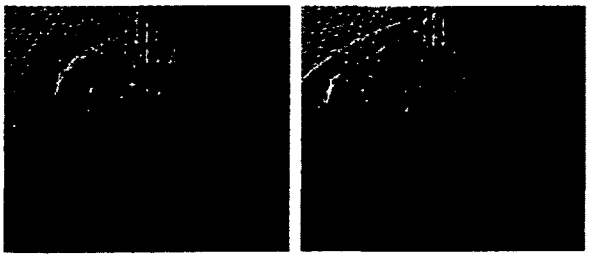

그림3. 와이어 프레임으로 본 물 모델의 실험결과 예

5. 결론 및 향후 연구

본 논문에서는 격자 크기(grid size)를 조절함으로써 사용자가 원하는 프레임 수를 보장할 수 있는 시스템을 제안했다. 본 논문에서 제안한 모델은 실시간 애니메이션이나 게임과 같은 곳에 적용할 수 있는 모델이다. 본 논문에서는 O'Brien이 제안한 모델에서 실시간 시뮬레이션에 적용이 가능하도록 개선하였다. 시뮬레이션 할 때 볼륨을 보존하기 위해서 여러 번의 반복 계산을 하지 않고 한번에 계산을 수행할 수 있도록 함으로써 계산량을 줄였으며, 안정적인 결과들을 얻을 수 있었다.

향후 연구로서는 사실적인 물의 동작을 표현하기 위해서 입자 시스템의 도입이 필요하다. 또한 부딪히거나 부서지는 물의 동작이나, 3차원 물 시뮬레이터 모델을 실시간 애니메이션이나 게임에 적용 할 수 있도록 계산량을 줄이는 연구가 필요하다.

6. 참고문헌

[1] Darwyn Peachey. "Modeling Waves and surf". Proceedings of SIGGRAPH'86, pp 65-74, 1986.

[2] Alain Fournier and William Reeves. " A simple Model of ocean Waves".Proceedings of SIGGRAPH '86, PP. 75-84, 1986.

[3] Nick Foster and Dimitri Metaxas. "Realistic Animation of Liquids". Proceedings of GI'96, pp.204-212, 1996.

[4] James F. O'Brien and Jessica K. Hodgins. "Dynamic Simulation of Splashing Fluids". Computer Animation '95, pp.198-205, 1995.

[5] Michael Kass and Gavin Miller. "Rapid, Stable Fluid Dynamics for Computer Graphics". Proceedings of SIGGRAPH'90, in Computer Graphics, Vol. 24, No.4, pp 49-57, 1990.

[6] Nick Foster and Ronald Fedkiw. "Practical Animation of Liquids". Proceedings of SIGGRAPH 2001, pp 23-30, 2001.