입자 동역학 시뮬레이션과 선적분 볼륨 렌더링을 이용한 실시간 유체 애니메이션

(Real-time Fluid Animation using Particle Dynamics Simulation and Pre-integrated Volume Rendering)

이 정 진 † 강 문 구 †† 김 동 호 ††† 신 영 길 ††††
(Jeongjin Lee) (Moon Koo Kang) (Dongho Kim) (Yeong Gil Shin)

요 약 유체 애니메이션은 물리적 시뮬레이션과 시각적 렌더링으로 구성된다. 물리적 시뮬레이션은 입자 동역학을 이용한 해석 방법과 나비어-스토크스(Navier-Stokes) 방정식을 이용한 연속체 유동해석 방법이 많이 사용된다. 입자 동역학을 이용한 시뮬레이션은 연산 속도는 빠르나 유체의 움직임이 경우에 따라 부자연스러우며, 나비어-스토크스 방정식을 이용한 방법은 적절한 조건 하에서 사실적인 유체의 움직임을 표현할 수 있으나 방대한 연산량과 계산의 복잡성으로 인하여 실시간 응용이 어렵다. 우수한 품질의 렌더링 영상은 주로 전역적 조명 방법을 사용하여 얻을 수 있는데, 이 역시 실시간 응용에 적합한 속도를 내기에는 부적합하다. 본 논문에서는 개선된 입자 동역학 시뮬레이션과 선적분 볼륨 렌더링을 이용한 고속 유체 애니메이션 방법을 제안한다. 레나드-존스(Lennard-Jones) 모델을 이용한 입자동역학 해석기법을 이용하여 유체의 움직임을 고속으로 시뮬레이션 하였으며, 적은 수의 입자만으로도 충분한 유체의 부피를 표현할 수 있도록 연산효율을 개선하였다. 또한 실시간 렌더링을 위하여 적은 수의 슬라이스로도 우수한 품질의 영상을 빠르게 얻을 수 있는 선적분 볼륨 렌더링 방식을 사용하였다. 본 제안 방법을 사용하여 실시간 응용에 적절한 속도와 화질을 보여주는 유체 애니메이션이 가능하다.

키워드 : 유체 애니메이션, 입자 동역학 시뮬레이션, 선적분 볼륨 렌더링

Abstract The fluid animation procedure consists of physical simulation and visual rendering. In the physical simulation of fluids, the most frequently used practices are the numerical simulation of fluid particles using particle dynamics equations and the continuum analysis of flow via Navier-Stokes equation. Particle dynamics method is fast in calculation, but the resulting fluid motion is conditionally unrealistic. The method using Navier-Stokes equation, on the contrary, yields lifelike fluid motion when properly conditioned, yet the complexity of calculation restrains this method from being used in real-time applications. Global illumination is generally successful in producing premium-quality rendered images, but is also excessively slow for real-time applications. In this paper, we propose a rapid fluid animation method incorporating enhanced particle dynamics simulation method and pre-integrated volume rendering technique. The particle dynamics simulation of fluid flow was conducted in real-time using Lennard-Jones model, and the computation efficiency was enhanced such that a small number of particles can represent a significant volume. For real-time rendering, pre-integrated volume rendering method was used so that fewer slices than ever can construct seamless inter-laminar shades. The proposed method could successfully simulate and render the fluid motion in real time at an acceptable speed and visual quality.

Key words : fluid animation, particle dynamics simulation, pre-integrated volume rendering

† 학생회원 : 서울대학교 컴퓨터공학부
jjlee@cglab.snu.ac.kr
†† 비 회 원 : 서울대학교 전기컴퓨터공학부 교수
moonkang@ee.snu.ac.kr
(Corresponding author 임)
††† 비 회 원 : 숭실대학교 미디어학부 교수
dkim@ssu.ac.kr
†††† 종신회원 : 서울대학교 컴퓨터공학부 교수
yshin@cglab.snu.ac.kr
논문접수 : 2004년 4월 8일
심사완료 : 2004년 9월 14일

1. 서 론

컴퓨터 그래픽을 이용한 애니메이션에서 최근 활발한 연구가 이루어지는 분야 중의 하나는 물, 불, 연기 등의 불규칙한 운동을 실제적으로 표현하는 유체 애니메이션이다. 최근 3차원 게임 등의 응용 분야에서 실시간 유체 애니메이션의 필요성이 크게 증가하고 있으나, 유체는 움직임이 불규칙하여 사실적인 애니메이션이 어렵고, 특

히 이를 실시간으로 애니메이션하는 것은 더욱 어렵다[1-4].

기존의 유체 애니메이션 방법을 실시간 응용에 적용하기 위해서는 몇 가지 문제점들이 존재한다. 첫 번째 제약 요인은 물리적 시뮬레이션의 연산 시간이다. 사실적 유체 애니메이션을 위해서는 나비어-스토크스 방정식을 이용한 방법이 많이 사용되고 있다. 복잡한 유체의 운동을 표현하기 위해서는 많은 격자가 필요하여 오랜 연산 시간이 소요된다. 두 번째 제약 요인은 시각적 렌더링의 연산 시간이다. 포톤 사상법(Photon mapping) 등을 이용한 전역적 조명 방법은 높은 품질의 사실적 렌더링을 구현할 수 있으나 이를 위해 오랜 연산 시간이 소요된다[5]. 이와 같이 기존의 방법은 다양한 유체의 실시간 애니메이션에 적용되기에 여러 가지 제약 요인들이 있다.

본 논문에서는 이러한 제약 요인들을 효과적으로 극복하여 실시간 유체 애니메이션을 가능하게 하는 고속의 시뮬레이션 및 렌더링 기법을 제안하였다. 물리적 시뮬레이션 방법으로 기존의 입자 동역학 방식을 개선하였다. 입자 간 충돌 및 인력에 의한 상호작용에 따른 점성의 효과를 표현하기 위하여 입자의 유효 반경을 정하고, 레나드-존스 모델[6]을 이용해 입자 간 인력과 척력을 계산하였다. 또한 렌더링 방법으로 선적분 볼륨 렌더링 방식을 사용하였다[7]. 선적분 볼륨 렌더링은 적은 양의 시뮬레이션 데이타로도 슬라이스 보간 기법에 의해 고속으로 고화질의 영상을 얻을 수 있다. 또한, 시뮬레이션과 렌더링 인자를 실시간으로 조절할 수 있어 사용자가 유체의 입자적 특성과 연속체적 특성을 임의로 조정할 수 있게 되므로, 애니메이션적인 과장과 생략이 가능하며 유체의 표현 범위가 넓어진다. 본 논문에서 제안한 방법의 실험 결과는 평균적인 성능의 개인용 컴퓨터에서 실시간 응용에 적절한 속도와 화질을 보여주었다.

2. 관련 연구

Foster[1]는 3차원 비압축성 나비어-스토크스 방정식을 적용하여 유체의 유동을 물리적으로 시뮬레이션하여 가시화하는 방법을 제안하였다. Stam[2]은 역학적인 정확성을 희생하여 큰 시간 증분(time step)에 대해서도 나비어-스토크스 방정식을 안정적으로 풀 수 있는 방법을 제시하였으며, 물리현상의 엄밀한 재현보다는 시각적으로 다양하고 역동적인 영상을 효과적으로 산출해야 하는 유체 애니메이션의 특성을 잘 반영하였다. Foster[3]는 나비어-스토크스 방정식을 이용하여 사실적인 유체의 움직임을 음함수면(implicit surface) 방법으로 가시화하였다. Enright[4]는 이 방법을 발전시켜 동적 음함수면(dynamic implicit surface)과 입자 레벨 셋(particle level set) 방법을 복합적으로 이용하였다. 렌더링은 물리적 몬테 카를로 광선 추적법(physically based Monte Carlo ray tracer)에 기반을 둔 포톤 사상 방법을 사용하여 사실감을 더하였다. 이상의 방법들은 유동의 시뮬레이션과 렌더링에 많은 연산 시간이 소요되므로 실시간의 적용은 용이하지 않다.

입자 동역학에 기반을 둔 시뮬레이션 기법들을 사용할 경우 비교적 빠른 유체 현상의 모델링이 가능하다. Miller[8]는 점성을 갖는 유체를 표현하기 위해 입자 간에 스프링을 연결하는 모델을 제안하였다. Terzopolous[9]는 분자 역학을 이용하여 입자 간 상호작용을 모델링하였다. 하지만, 이러한 방법들은 적은 수의 입자를 사용하였을 때, 어느 정도 실시간 애니메이션이 가능하나 영상의 화질이 부자연스럽고, 입자의 수가 증가하면, 영상의 화질은 적당하나 입자 간 연결이 지수 함수적으로 증가하게 되어 연산 시간이 오래 걸린다.

실시간 유체 애니메이션의 효용성은 물리적 역학방정식의 해석 속도뿐만 아니라 렌더링 속도에 의해 크게 좌우된다. 유체는 그 표면 형상이 복잡하고 빛을 반사시키거나 투영시키는 등의 복잡한 광학적 현상을 수반하므로 고성능의 렌더링 알고리즘을 필요로 한다. Jensen[5]이 제안한 포톤 사상 방법은 최근의 유체 애니메이션에 많이 사용되고 있다. 이 방법은 영상이 사실적이지만, 한 프레임의 영상을 만들기 위해서 수 분의 시간이 소요되기 때문에 실시간 응용에는 부적합하다. 불규칙한 형상의 변화가 존재하는 문제에 효과적 적용이 가능한 볼륨 렌더링 기법은 Levoy[10]에 의해 제안된 광선 투영법(ray casting)으로부터 시작되었다. 볼륨 렌더링은 데이타 처리량이 많으므로 공간 도약법[11-12]이나 조기 광선 중단법[13] 등을 적용하여 이를 가속하였다. 최근에는 그래픽스 하드웨어의 발달에 따라 하드웨어 텍스처를 볼륨 렌더링의 가속에 이용하는 기술이 제안되고 있다[14].

본 연구에서는 주어진 공간을 균등한 간격의 격자들로 분할하고 유체의 운동에 따라 각 격자 내에 유체가 차지하는 부피를 계산하여 유체의 불규칙한 입체적 형상을 스칼라량의 분포로 모델링 하였다. 유체 입자 간 점성 효과를 표현하기 위해서는 입자 간 인력과 척력이 고려되어야 한다. 본 연구에서는 분할된 격자 각각의 미소체적에 공통으로 포함된 입자들에 대해 제한적으로 역학적 상호작용을 고려하여, 전체 영역의 입자들의 상호작용을 계산할 경우에 비하여 연산량을 큰 폭으로 감소시켰다. 또한 같은 미소체적 내의 입자들 사이에서도 거리가 비교적 먼 입자 간에는 레나드-존스 모델이 아닌 강구충돌 모델을 사용하여 연산량을 감소

시켰다. 렌더링 부분에 있어서는 Engel[7]의 선적분 볼륨 렌더링 방식을 사용하여 적은 양의 시뮬레이션 데이타로도 고화질의 영상을 만들 수 있도록 하였으며, 하드웨어 텍스쳐 가속 기술을 적용하여 렌더링 속도를 높였다.

3. 실시간 유체 애니메이션

다양한 유체의 물리적 특성을 표현하기 위한 시뮬레이션 인자와 시각적 특성을 표현하기 위한 렌더링 인자가 설정되어야 한다. 주요한 시뮬레이션 인자로는 밀도, 입자 수, 입자 반지름 등이 있다. 주요 렌더링 인자로는 볼륨 렌더링의 불투명도 변환 함수(opacity transfer function)에 사용되는 색상과 불투명도가 있다. 다음 단계에서는 경계 조건과 초기 조건이 정의된다. 경계 조건은 벽과 같은 경계면으로 구성되고, 각 면은 삼각형 메쉬(mesh)로 구성된다. 유체가 외부로부터 유입되는 지점의 초기 속도와 내부 공간의 초기 분포상태 등에 의해 유동의 초기조건이 결정되며, 시간에 따라 역학적 균형상태의 변화로 인해 유체 입자의 움직임이 구현된다. 마지막 단계는 입자 동역학 기반 시뮬레이션과 선적분 볼륨 렌더링을 통합한 애니메이션 과정이다. 사용자는 실시간 시뮬레이션이 진행되는 도중 임의의 순간에 입력데이타와 경계조건을 수정하여 유동을 변화시킬 수 있다.

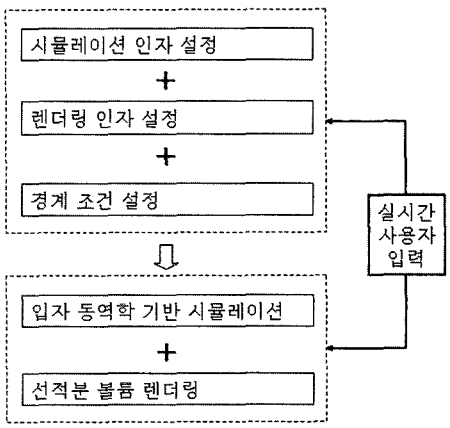


그림 1 실시간 유체 애니메이션의 개요

4. 입자 동역학 시뮬레이션

4.1 운동 방정식

본 논문에서는 입자 동역학의 기본 방정식[6]을 기반으로 하여 실시간 유체 애니메이션의 특성에 맞게 개선된 알고리즘을 사용하여 시뮬레이션을 수행하였다. 먼저 구형의 입자들을 이용하여 임의의 형태의 물체를 구성한다. N개의 구형 대칭인 입자들에 대해서 입자 내부의 포텐샬(potential) 함수는 $E(r^N)$으로 표시될 수 있다. 여기서 r^N은 입자들의 질량의 중심의 위치를 나타내는 벡터들의 집합으로 $r^N = \{r_1, r_2, r_3, \ldots r_N\}$으로 표시될 수 있다. 총 입자간 포텐샬 에너지 $E(r^N)$는 교환 법칙에 따라 각각의 입자 간 상호작용의 합으로 다음의 식 (1)과 같이 구해질 수 있다.

$$E(r^N) = \sum_{i=1}^{N} \sum_{j=1}^{N} u(r_{ij}), (i \neq j) \quad (1)$$

(r_{ij} : 입자 i와 j간의 거리)

입자 i와 j 간의 기본적인 포텐샬 에너지 $u(r_{ij})$는 레나드-존스 포텐샬[15-16]을 이용하였다. 입자 간에 작용하는 힘은 가까운 거리 내의 입자 간에는 반발력이 작용하고, 어느 거리 이상에 위치한 입자 간에는 인력이 작용하게 된다. 거리 r_{ij}만큼 떨어진 입자 i와 j에 대해서 두 입자 사이의 포텐샬 에너지 $u(r_{ij})$는 식 (2)와 같이 표현된다.

$$u(r_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \quad (2)$$

(ε : 상호작용 강도를 나타내는 계수, σ : 길이 스케일 계수)

이때, 입자 i와 j 사이에 작용하는 힘 f_{ij}는 다음의 식 (3)과 같이 계산할 수 있다.

$$f_{ij} = -\frac{du(r_{ij})}{dr_{ij}} = \frac{48\varepsilon}{\sigma^2}\left[\left(\frac{\sigma}{r_{ij}}\right)^{14} - \frac{1}{2}\left(\frac{\sigma}{r_{ij}}\right)^{8}\right] r_{ij} \quad (3)$$

입자 간의 포텐샬 에너지에 의한 힘은 주어진 포텐샬 장 내에서 보존되므로, 입자 i에 작용하는 포텐샬 에너지에 의한 힘 $F_{i,p}$는 다음의 식 (4)와 같이 계산된다.

$$F_{i,p} = -\frac{\partial E(r^N)}{\partial r_i} = m_i \ddot{r}_i \quad (4)$$

(m_i : 입자 i의 질량)

그림 2는 레나드-존스 포텐샬과 적당한 간격에서 입자 간에 작용하는 힘을 나타낸 것이다. 입자 간에 작용하는 힘의 부호가 양에서 음으로 바뀌는 위치는 입자의 반지름으로 생각될 수 있다.

그리고, 입자 i에 작용하는 마찰력 $F_{i,f}$는 식 (5)와 같이 주어진다.

$$F_{i,f} = -\zeta \dot{r}_i \quad (5)$$

(ζ : 마찰계수)

결과적으로 입자 i에 작용하는 힘들을 합한 운동 방정식은 아래의 식 (6)과 같이 서술된다.

$$-\frac{\partial E(r^N)}{\partial r_i} + \zeta \dot{r}_i = 0 \quad (6)$$

이상의 식들은 입자를 연구(soft sphere)로 가정한 것으로, 입자가 대표하는 유효반경과 비슷한 거리에 분포된 입자 간의 상호작용을 고려할 때에는 유용하다. 그러

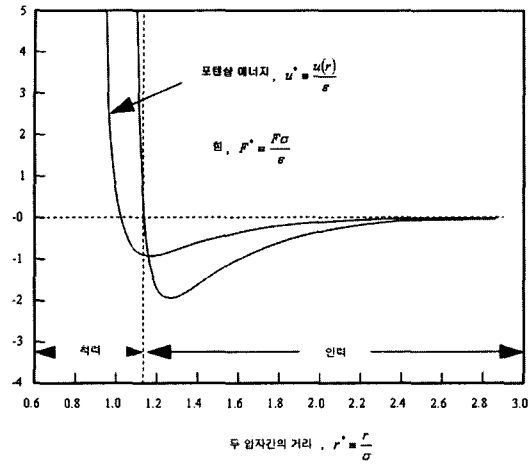


그림 2 레나드-존스 포텐셜과 힘과의 관계

나 입자의 유효반경에 비해 훨씬 큰 거리만큼 떨어진 입자들 사이에서는 충돌의 확률이나 인력의 세기가 미약하므로 각 입자들을 강구(hard sphere)로 가정하여 물리적 사실감을 크게 저하시키지 않으면서도 연산부하를 크게 줄일 수 있다. 강구로 가정된 입자 간의 충돌 후의 입자들의 속도는 식 (7)과 같이 주어진다.

$$u_i = \frac{1}{m_i+m_j}[(m_i - em_j)u_i + m_j(1+e)u_j]$$
$$u_j = \frac{1}{m_i+m_j}[(m_i - em_j)u_j + m_i(1+e)u_i]$$
$$v_i = \frac{1}{m_i+m_j}[(m_i - em_j)v_i + m_j(1+e)v_j]$$
$$v_j = \frac{1}{m_i+m_j}[(m_i - em_j)v_j + m_i(1+e)v_i]$$
$$w_i = \frac{1}{m_i+m_j}[(m_i - em_j)w_i + m_j(1+e)w_j]$$
$$w_j = \frac{1}{m_i+m_j}[(m_i - em_j)w_j + m_i(1+e)w_i] \quad (7)$$

여기서, u_i, v_i, w_i는 입자 i의 x, y, z 방향 속도 성분을 나타내며, m은 질량, e는 입자와 입자간의 복원계수를 나타낸다. 본 연구에서는 분할된 공간을 표현하는 각각의 미소체적에 공통으로 포함된 입자들 가운데 임계거리에 가까운 위치에 놓인 입자들에 대해 제한적으로 식 (6)의 레나드-존스 모델을 이용한 역학적 상호작용을 고려함으로써, 충돌의 가능성이 적은 입자들 사이의 상호작용과 관련된 연산량을 큰 폭으로 절감하였다. 또한 거리가 비교적 먼 입자 간에는 레나드-존스 모델이 아닌 식 (7)의 강구충돌 모델을 사용하여 부가적인 연산을 줄였다.

4.2 입자와 벽 사이의 상호작용 모델링

입자 i가 벽 k에 충돌할 때 입자의 속도는 벽에 평행한 성분과 수직한 성분으로 나뉠 수 있다. 그 중 수직한 성분의 x, y, z 성분인 $Vn = u_{n,i}\,i + v_{n,i}\,j + w_{n,i}\,k$은 다음과 같은 식 (8)로부터 구해진다.

$$u_{n,i} = \frac{a_k u_i + b_k v_i + c_k w_i}{a^2+b^2+c^2} \times a_k$$
$$v_{n,i} = \frac{a_k u_i + b_k v_i + c_k w_i}{a^2+b^2+c^2} \times b_k$$
$$w_{n,i} = \frac{a_k u_i + b_k v_i + c_k w_i}{a^2+b^2+c^2} \times c_k \quad (8)$$

여기서 a_k, b_k, c_k는 벽 k를 나타내는 아래와 같은 평면의 방정식의 계수이다.

$$a_k x + b_k y + c_k z + d_k = 0 \quad (9)$$

또한, 벽과 평행한 접선 방향의 속도 성분 $Vt = u_{t,i}\,i + v_{t,i}\,j + w_{t,i}\,k$는 식 (10)과 같이 입자의 속도에서 수직 방향 성분을 빼주어서 구할 수 있다.

$$u_{t,i} = u_i - u_{n,i}$$
$$v_{t,i} = v_i - v_{n,i}$$
$$w_{t,i} = w_i - w_{n,i} \quad (10)$$

그리고, 입자 i와 벽 k의 충돌 후에는 수직 속도의 방향은 반대가 되고, 그 크기는 복원계수 e만큼 감쇄되며, 접선 속도는 마찰 계수 ζ에 영향을 받게 된다. 따라서 다음의 식 (11)과 같이 표현된다.

$$u_i^{new} = -eu_{n,i} + (1-\zeta)u_{t,i}$$
$$v_i^{new} = -ev_{n,i} + (1-\zeta)v_{t,i}$$
$$w_i^{new} = -ew_{n,i} + (1-\zeta)w_{t,i} \quad (11)$$

이러한 충돌 후의 속도 성분을 사용해서 벽에 부딪히는 입자들의 운동을 모델링 할 수 있다. 이 속도 성분을 시간에 대해 적분하면 입자의 위치 데이타를 얻을 수 있게 되어, 다음의 렌더링 단계에서 임의의 시간과 위치에서의 유체 입자들의 분포를 예측할 수 있다.

5. 시뮬레이션 데이타의 선적분 볼륨 렌더링

5.1 시뮬레이션 데이타로부터 렌더링 데이타 생성

시뮬레이션 데이타를 구성하는 입자는 공간을 자유롭게 움직일 수 있기 때문에 기본적으로 점 자료 구조(point data structure)의 형태를 가진다. 이러한 시뮬레이션 데이타를 볼륨 렌더링에 적용하기 위해서는 볼륨 렌더링에 적합한 자료 구조로 변환이 필요하다. 먼저, 공간을 한 변의 길이가 d인 정육면체의 일정한 단위 격자로 세분한다. 각 격자에 해당하는 공간에 포함된 입자들의 개수가 n_{grid}이고, 입자의 반지름이 r이라고 하면, 각 격자 내부에서 입자들이 차지하는 체적분율(volume fraction)을 다음과 같이 구할 수 있다. 이 체적분율 값이 볼륨 렌더링을 위한 볼륨 데이타가 된다.

$$\text{체적분율} = \frac{n_{grid} \times \frac{4}{3}\pi r^3}{d^3} \quad (12)$$

5.2 하드웨어 가속 선적분 볼륨 렌더링

볼륨 렌더링은 3차원의 공간에 분포된 물질의 단위 부피당 체적분율에 해당하는 밀도값을 균등한 간격의 단면으로 샘플링하여 입체적으로 가시화하는 기술이다[8]. 이 방법은 반투명한 볼륨의 2차원 투영을 계산하여 이미지를 얻는다. 이때 전체 볼륨 내의 각 복셀은 실제 데이타로부터 얻어진 색상과 불투명도를 갖는다. 볼륨 렌더링의 한 방법인 광선 투영법은 시점에서 출발하는 광선을 따라서 볼륨 데이타를 샘플링하고, 샘플링한 점들의 색상과 불투명도를 적분하여 픽셀에 투영한다.

선적분 볼륨 렌더링이란 볼륨 데이타의 인접한 두 슬라이스 사이의 색상과 불투명도를 미리 적분하여 2차원 텍스쳐에 저장해 두고 렌더링 시에 이를 이용하여 음영화(shading)하는 볼륨 렌더링의 한 방법이다[7]. 슬라이스의 값만으로 합성(compositing)하는 기존의 방법[10]보다 슬라이스 사이의 색상과 불투명도를 적분하여 합성하므로 적은 양의 슬라이스 수로도 좋은 화질을 얻을 수 있다. 또한, 전처리 단계로 색상과 불투명도를 미리 적분해 놓기 때문에 빠른 시간에 렌더링이 가능하다.

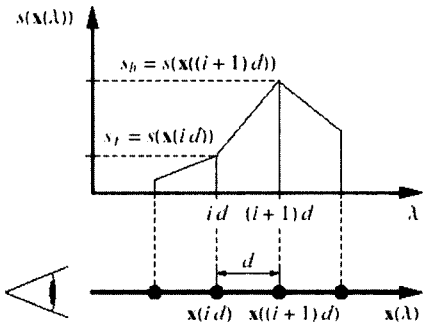


그림 3 광선 경로상의 위치와 밀도 값의 관계

그림 3은 광선의 경로상의 위치와 밀도 값 s와의 관계를 도시하였다. 그림 3에서 i번째 광선 부분에서 방출되는 색상과 불투명도를 곱한 값을 $\tilde{C}_i$라고 하고, i번째 광선 부분의 불투명도를 α_i라고 하면, 볼륨 렌더링에 의한 한 픽셀의 색상은 다음과 같이 계산될 수 있다.

$$I \approx \sum_{i=0}^{n} \tilde{C}_i \prod_{j=0}^{i-1}(1-\alpha_j) \tag{13}$$

i번째 광선 부분에 대해서 앞쪽의 밀도 값은 $s_f = s(x(id))$가 되고, 뒤쪽의 밀도 값은 $s_b = s(x((i+1)d))$가 된다. $\tilde{c}(s(x(\lambda)))$는 $x(\lambda)$에서의 색상, $\tau(s(x(\lambda)))$는 시점으로부터의 거리가 멀어짐에 따라 불투명해지도록 하는 소멸 계수, λ는 시점에서 현재 위치까지의 거리라고 하면, i번째 광선 부분의 불투명도는 다음과 같이 근사 될 수 있다.

$$\begin{aligned}\alpha_i &= 1-\exp\left(-\int_{id}^{(i+1)d}\tau(s(x(\lambda)))d\lambda\right)\\ &\approx 1-\exp\left(-\int_0^1 \tau((1-w)s_f+ws_b)ddw\right)\end{aligned} \tag{14}$$

결국, α_i는 s_f, s_b와 광선 부분의 거리인 d의 함수가 된다. i번째 광선 부분의 $\tilde{C}_i$는 다음과 같이 계산될 수 있다.

$$\begin{aligned}\tilde{C}_i \approx &\int_0^1 \tilde{c}((1-w)s_f+ws_b)\times\\ &\exp\left(-\int_0^w \tau((1-w')s_f+w's_b)ddw'\right)ddw\end{aligned} \tag{15}$$

그러나 이런 적분식을 각 광선 부분마다 적용하려면 시간이 많이 걸리므로 가속화를 위해 다음과 같은 방법을 쓸 수 있다. $T(s)=\int_0^s \tau(s)ds$을 정의하면, $\alpha_i = \alpha(s_f, s_b, d)$을 식 (16)과 같이 T를 사용해서 나타낼 수 있다.

$$\alpha(s_f, s_d, d) \approx 1-\exp\left(-\frac{d}{s_b-s_f}(T(s_b)-T(s_f))\right) \tag{16}$$

비슷한 방법으로 $K(s)=\int_0^s \tilde{c}(s)ds$을 정의하면, $\tilde{C}_i = \tilde{C}(s_f, s_b, d)$을 식 (17)과 같이 K를 사용해서 나타낼 수 있다.

$$\tilde{C}(s_f, s_d, d) \approx \frac{d}{s_b-s_f}(K(s_b)-K(s_f))) \tag{17}$$

s_f, s_b, d의 각 조합에 대해 식 (14), 식 (15)를 계산하는 것 대신, 적분 함수 T(s), K(s)를 한번씩 계산해서 이 함수를 식 (16), 식 (17)에 적용하였다. T(s), K(s)는 근사 값을 계산하여 s에 대한 1차원 배열로 저장한다. d가 일정하다고 가정하면, T와 S로부터 미리 계산된 색상과 불투명도는 s_f, s_b의 함수가 되어 sf와 sb를 텍스쳐 좌표로 하는 2차원 텍스쳐 메모리에 저장할 수 있다[14]. 이 방법을 사용하면 그래픽스 하드웨어를 이용하여 색상과 불투명도 계산을 가속할 수 있다.

5.3 렌더링 인자

다양한 유체의 색상을 표현하기 위해서 그림 4의 불투명도 변환 함수를 사용한다. 볼륨 데이타는 입자의 체적분율 값을 사용하여 0과 1사이의 실수 값을 갖는다. 해당 볼륨데이타를 나타내는 V1, V2, V3, V4의 점에 유체의 다양한 시각적 효과를 위해서 색상을 지정한다. 각 점의 사이 값들은 선형 보간을 통해서 계산된다.

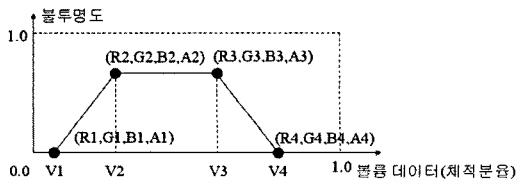


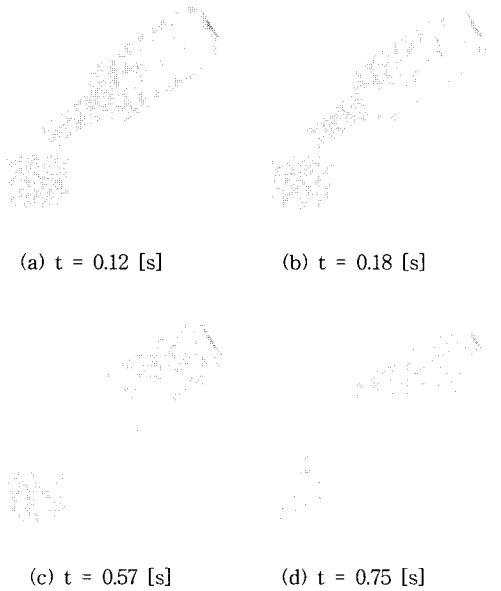
그림 4 불투명도 변환 함수

6. 실험 결과

본 연구에서 제안된 방법의 효용성을 검증하기 위해 컵에 물을 따르는 과정에 대하여 실시간 애니메이션을 수행하였다. 실험은 Pentium IV 2.4GHz CPU와 1.0GB의 메모리를 장착한 PC에서 이루어졌고 그래픽스 하드웨어는 GeForce FX 5800를 사용하였다.

그림 5, 6의 결과에서 불투명도 전이 함수는 V1(0, 0, 255), V2(36, 65, 91), V3(154, 208, 228), V4(0, 0, 160)의 색상을 주었고, V1, V2, V3, V4에 해당하는 볼륨 데이타의 값은 각각 0.21, 0.23, 0.28, 0.30이다. 이 값들은 물의 효과적인 렌더링을 위해 많은 실험에 의해서 결정되었다. 병과 컵의 형상을 모델링하기 위해 사용된 삼각형의 개수는 243개이다. 그림 5에서는 입자의 수(n)를 2,000개로 하였으며, 그림 6은 입자의 수를 10,000개로 하였을 때의 결과이다. 그림 5의 예제에서는 730×520 해상도의 영상을 애니메이션 할 때 7~8 fps의 속도가 얻어졌으며, 시뮬레이션에 소요된 시간이 전체의 46%를 차지하였다. 그림 6의 예제에서는 730×520 해상도의 영상을 애니메이션 할 때 3~4 fps의 속도가 얻어졌다. 또한, 입자수가 늘어나더라도 렌더링을 위한 볼륨 데이타의 양에는 변화가 없으므로 렌더링 시간에는 큰 변화가 없었다. 입자의 개수를 증가시킨 경우 그림 5와 그림 6의 비교에서 볼 수 있는 바와 같이 매끄러운 물의 표면과 세밀한 물방울의 표현이 가능해지며, 정지 영상을 통해서는 확인이 어려우나 동영상의 경우 그림 6의 경우가 그림 5에 비해 입자 개수 증가에 따른 연산 속도의 저하로 인해 유체의 움직임이 끊어지는 경향이 나타났다. 이와 같이 입자의 개수를 늘릴수록 더욱 세밀하고 자연스러운 유체 표면을 표현할 수 있으나 연산 시간도 증가하므로, 실시간 적용 시에는 유체운동의 사실감과 소요시간 사이에 적절한 최적화가 필요하다.

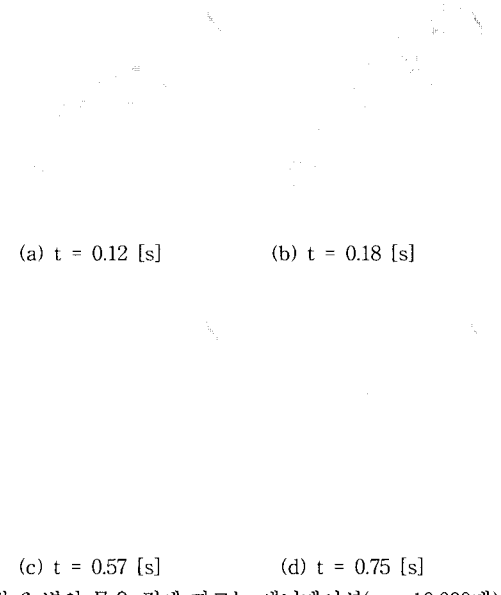
유체와 상호작용하는 벽면의 기하학적 형상을 더욱 정밀하게 표현하기 위해서는 삼각형으로 모델링된 벽면의 삼각형 개수를 증가시킬 필요가 있다. 그러나 삼각형 개수가 많아질 경우 각 입자의 충돌 여부를 판별하는데 많은 연산 시간이 소요된다. 삼각형의 개수와 연산 시간의 관계를 고찰하기 위하여 삼각형의 개수를 달리하여 각각 82 프레임 동안의 평균 FPS를 측정하였다. 그 결과 삼각형 개수가 증가함에 따라 그림 7에서 볼 수 있듯이 연산 속도가 감소했다. 일반적으로 유체 표면의 형상이 불규칙하므로, 위의 예제와 같은 병과 컵의 형상의 경우 삼각형의 개수를 243개에서 972개로 약 4배 증가시킨 경우에도 연산 시간의 증가에 비해 물리적인 사실감과 영상의 화질 개선은 거의 없었다. 따라서 벽면의 형상을 적절히 표현할 수 있는 범위 내에서 삼각형 개수를 최소한으로 줄여 연산속도를 향상시키는 것이 효율적이다.



(a) t = 0.12 [s] (b) t = 0.18 [s]

(c) t = 0.57 [s] (d) t = 0.75 [s]

그림 5 병의 물을 컵에 따르는 애니메이션(n = 2,000개)



(a) t = 0.12 [s] (b) t = 0.18 [s]

(c) t = 0.57 [s] (d) t = 0.75 [s]

그림 6 병의 물을 컵에 따르는 애니메이션(n = 10,000개)

그림 8에서는 병의 물을 컵에 따르는 실험을 통해 실제로 촬영된 유체 운동의 사진과 본 연구에서 제안한 실시간 애니메이션 방법으로 구현된 영상을 비교하였다. 빠른 속도로 흘러내리는 유체를 촬영하기가 어렵기 때문에 실제 촬영된 영상과 본 제안 방법으로 구현된 영

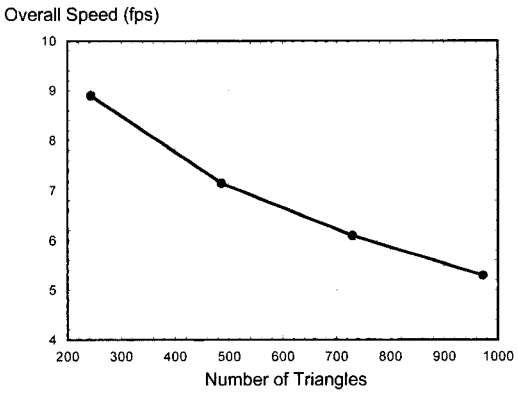
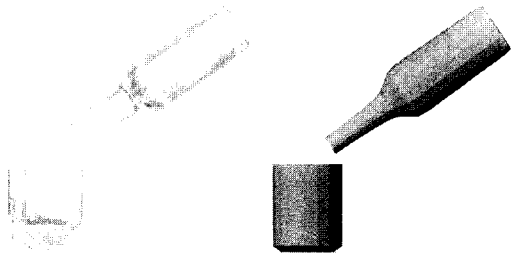
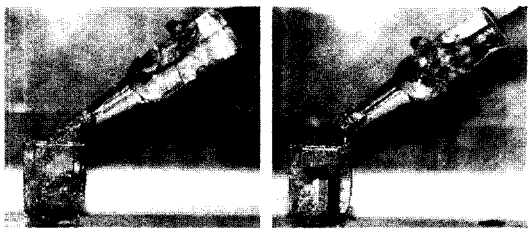


그림 7 삼각형 개수와 시뮬레이션 및 렌더링 속도와의 관계



(a) 병과 컵의 실사 및 3차원 모델링 형상

(b) 실시간으로 애니메이션 되고 실시간으로 병과 컵의 사진에 합성된 유체



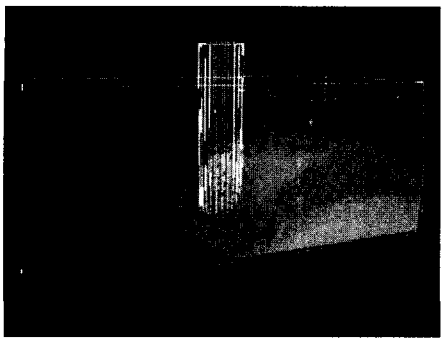
(c) 실제 촬영된 사진

그림 8 실시간 애니메이션 결과와 실사의 비교

상이 정확히 시간대 별로 비교되지는 못하였으나, 실사와 제안 방법의 전체적인 유체의 운동이 유사한 경향을 보였다. 실사와 애니메이션 결과가 차이를 보이는 이유는 병과 컵의 내벽 형상 모델링의 부정확성, 실험 조건의 정확한 수치적 재현의 어려움 등에서 우선적으로 원인을 찾을 수 있다. 또한, 다수의 가정이 도입된 입자동역학 해석 방식을 이용하여 유체의 움직임을 정확히 재현하는데 한계가 존재한다. 그러나 공학적인 데이타의 분석 목적이 아닌 실시간 애니메이션의 목적에 사용하기에는 그림 8의 결과물로도 충분한 사실감을 제공하는 것으로 판단된다.

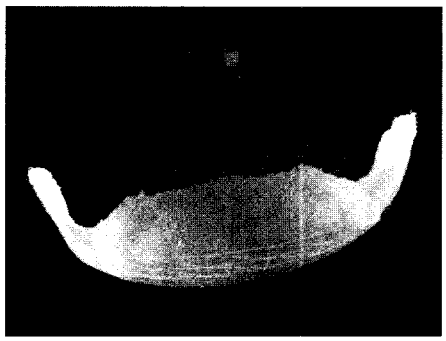
그림 9에서는 비실시간 유체효과 구현을 전문으로 하는 상용 프로그램인 RealFlow의 결과와 본 제안 방법에 의한 결과를 비교하였다. 그림 9(a)는 RealFlow로 160,000개의 입자를 사용한 결과이고, 그림 9(b)는 본 제안 방법으로 8,000개의 입자를 사용한 결과이다. 그림 9(c)는 RealFlow로 5,000,000개의 입자를 사용한 결과이며, 그림 9(d)는 본 제안 방법으로 6,000개의 입자를 사용한 결과이다. 그림 9(a), 9(c)는 시뮬레이션에 사용된 모든 입자를 전역적 렌더링 전에 프리뷰(preview)의 형태로 나타낸 그림으로 모든 점이 화면상에 표시되어 세밀하게 보이나 실제로 유체 표면을 추정하여 렌더링한 후에는 표면상의 작은 점들은 나타나지 않으므로 최종 화면에는 세밀한 입자의 형상이 표현되지 않는다. 그림 9(a), 9(c)의 렌더링 결과는 제작사에서 제공하지 않아 시각적인 품질을 직접 비교하지 못하였으나, 그림 9(b), 9(d)에서 볼 수 있는 바와 같이 본 제안 방법을 사용하여 매우 적은 개수의 입자만으로 충분히 사실적인 유동의 표현이 가능하였다.

그림 9(e)는 RealFlow를 개발한 Nextlimit사의 인터넷 웹사이트에 게시된 예제로서, 프리뷰 뿐만 아니라 렌더링된 영상과 연산 시간이 제시되어 있다. 이 예제에는 450,000개의 입자가 사용되었으며, 약 10초 분량의 유체 동영상을 만들어 내기 위해 약 4시간 30분의 시간이 소요되었다. 그림 9(f)는 9(e)와 유사한 유동을 본 제안 방법으로 구현한 결과이다. 그림 9(e)의 시뮬레이션에 사용된 경계 조건이 제시되어 있지 않아 정확한 모델링은 불가능하였으며, 시각적으로 관찰할 때 가급적 유사한 유동이 표현되도록 하였다. 그림 9(f)는 9(e)의 예보다 훨씬 적은 개수인 10,000개의 입자를 사용하였으며, 실시간 애니메이션 방식으로 10초 분량의 영상이 초당 1.6 fps의 속도로 산출되었다. 영화 품질인 초당 24fps를 기준으로 하면, 본 제안 방법을 사용하여 실시간이 아닌 오프라인(off-line) 방식으로 10초 분량의 자연스러운 움직임을 구현하는 데는 약 150초의 시뮬레이션 및 렌더링 시간이 소요될 것으로 예상된다. 영상의 품질은 반사광이나 투과광의 사실적인 표현 등의 측면에서 Real-Flow가 다소 우수하나 실행 속도 측면에서는 본 제안

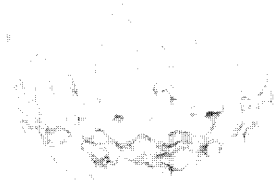


(a) RealFlow(입자수 = 160,000개)

(b) 본 제안 방법(입자수 = 8,000개)

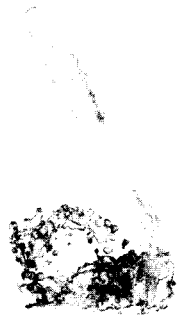


(c) RealFlow(입자수 = 5,000,000개)



(d) 본 제안 방법(입자수 = 6,000개)

(e) RealFlow(입자수 = 450,000개)



(f) 본 제안 방법(입자수 = 10,000개)

그림 9 상용 프로그램인 RealFlow와의 비교

방법이 월등히 빠르다. 본 제안 방법의 구현에 사용된 하드웨어는 Intel Pentium IV 2.4GHz CPU와 512MB RAM, ATI Radeon 9800 그래픽 카드이며, RealFlow를 이용한 시뮬레이션에 사용된 컴퓨터 사양은 공개되어 있지 않다. 하드웨어의 성능 차이를 무시하고 연산 시간만을 단순히 비교하면 본 제안 방법이 RealFlow에 비해 약 108배가량 빠른 것으로 나타난다. 그러나 RealFlow는 실시간용 소프트웨어가 아니며 전역적 렌더링을 통해 얻어지는 영상의 화질이 본 제안 방법에 비해 전반적으로 우수한 것을 감안하면, 속도를 기준으로 하여 두 방법을 단순히 비교하기 보다는 각각 적용분야의 특성에 맞게 활용될 수 있을 것으로 판단된다.

실시간으로 유체 애니메이션을 구현하는 기술은 매우 드물어 직접적인 비교의 대상을 찾기 어려우며, 연기나 안개 등 경계면이 흐리고 모호한 경우에 한해 Stam[2]의 연구 등에서 제한적으로 나타나고 있다. 그러나 연기나 안개와 같은 영상에 대해서는 시뮬레이션에 사용된 격자의 해상도와 연산에 사용된 여러 조건의 결과물로

얻어지는 영상의 화질을 정량적으로 비교하기가 어렵다. 실시간 유체 애니메이션이 가능한 상용 소프트웨어는 아직까지 출시되지 않아 직접적인 비교는 어렵다. RealFlow는 실시간 응용을 위해 개발된 프로그램이 아니므로 적절한 비교 대상이 아닐 수 있다. 그러나 RealFlow는 기존의 유체 시뮬레이션 소프트웨어 가운데 속도가 빠르고 기능이 우수하고, 이와 비교하여 본 제안 방법은 적은 수의 입자로도 적절한 화질을 보여주었다.

7. 결 론

본 논문에서는 입자 동역학을 이용한 시뮬레이션 방법과 선적분 볼륨 렌더링 방법을 통합한 실시간 유체 애니메이션 방법을 제시하였다. 효율적 연산을 위해 개선된 입자 동역학 시뮬레이션과 선적분 볼륨 렌더링의 텍스쳐 하드웨어 가속을 통해 실시간 응용에 적합한 속도를 얻을 수 있었다. 연산 속도를 높이기 위해 적은 개수의 입자로도 충분한 부피감을 표현하고 입자 간 상호작용력을 고려할 수 있는 레나드-존스 모델을 채택하였다. 입자 수의 부족에 따른 영상의 부자연스러움 극복하기 위하여 개선된 해석 알고리즘을 사용하였으며, 적은 개수의 슬라이스로도 좋은 화질을 보여주는 선적분 볼륨 렌더링을 적용하여 실험한 결과 우수한 화질의 실시간 유체 애니메이션이 가능하였다.

참 고 문 헌

[1] Nick Foster, "Realistic Animation of Liquids," Proceedings of Graphical Models and Image Processing, pp. 204-212, 1996.

[2] Jos Stam, "Stable Fluids," Proceedings of ACM SIGGRAPH '99, pp. 121-128, 1999.

[3] Nick Foster, Ronald Fedkiw, "Practical animation of liquids," Proceedings of ACM SIGGRAPH '01, pp. 23-30, 2001.

[4] D. Enright, "Animation and Rendering of Complex Water Surfaces," Proceedings of ACM SIGGRAPH '02, pp. 734-744, 2002.

[5] Henrik Wann Jensen and Per H. Christensen, "Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps," Proceedings of SIGGRAPH '98, pp. 311-320, July 1998.

[6] D. C. Rapaport, The art of molecular dynamics simulation, Cambridge University Press, 1995.

[7] Klaus Engel, Martin Kraus and Thomas Ertl, "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading," ACM Siggraph/Eurographics Workshop on Graphics Hardware 2001, pp. 9-16, 2001.

[8] Miller, Gavin S. P. and Pearce,A., "Globular Dynamics: A Connected Particle System for Animating Viscous Fluids," Computers and Graphics, Vol. 13, No. 3, pp. 305-309, 1989.

[9] Terzopoulos, Platt and Fleischer, "From Goop to Glop: Melting Deformable Models," Graphics Interface, 1989.

[10] Marc Levoy, "Efficient Ray Tracing of Volume Data," ACM Transactions on Graphics, Vol. 9, No. 3, pp. 245-261, July, 1990.

[11] Yagel, R. and Shi, Z., "Accelerating volume animation by space-leaping," Proceedings of IEEE Visualization '93, pp.62-69, 1993.

[12] Sramek, M. and Kaufman, A., "Fast ray-tracing of rectilinear volume data using distance transforms," IEEE Transactions on visualization and computer graphics, Vol.6, No.3, pp. 236-252, 2000.

[13] Danskin, J. and Hanrahan, P., "Fast algorithms for volume ray tracing," Workshop on Volume Visualization, pp. 91-98, 1992.

[14] S. Rottger., M. Kraus and T. Ertl, "Hardware-accelerated volume and isosurface rendering," In Proc. Of Visualizastion '00, pp. 109-116, 2000.

[15] McQuarrie, D.A., Statistical Mechanics, Harper and Row, New York, pp. 21-32, 1976.

[16] Maitland, G.C., Rigby, M., Smith, E. B. and Wakeham, W. A., Intermolecular Forces, Clarendon Press, Oxford, pp. 125-130, 1981.

이 정 진

2000년 2월 서울대학교 기계공학부 학사 2002년 2월 서울대학교 컴퓨터공학부 석사. 2002년 3월~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 컴퓨터를 이용한 자동 진단, 자연 현상 애니메이션, 가상 대장 내시경, 볼륨 렌더링

강 문 구

1991년 2월 서울대학교 기계공학과 학사 1993년 2월 서울대학교 기계공학과 석사 1997년 8월 서울대학교 기계공학과 박사 1997년~1998년 Univ. of California, Los Angeles, Post-doctor. 1998년~2000년 서울대학교 정밀기계설계공동연구소 특별연구원. 2001년 3월~현재 서울대학교 전기컴퓨터공학부 BK21 조교수. 관심분야는 열유체공학, microfluidics, MEMS 공정, 실시간 유체효과 시뮬레이션 및 렌더링

김 동 호

1990년 2월 서울대학교 전자공학과 학사 1992년 2월 한국과학기술원(KAIST) 전기 및 전자공학과 석사. 2002년 12월 미국 The George Washington University 전산학과 박사. 2003년 2월~2003년 8월 서울대학교 컴퓨터공학부 PostDoc 연구원. 2003년 9월~현재 숭실대학교 미디어학부 전임강사. 관심분야는 실시간 렌더링, 게임 그래픽스, 애니메이션, 볼륨 렌더링

신 영 길

1982년 2월 서울대학교 계산통계학과 학사. 1984년 2월 서울대학교 계산통계학과 석사. 1990년 2월 미국 University of Southern California 전산학과 박사. 1990년 2월~1992년 2월 경북대학교 전자계산학과 전임강사. 1992년 3월~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 볼륨 렌더링, 하드웨어 기반 렌더링, 의료 영상 처리