http://dx.doi.org/10.5392/JKCA.2013.13.06.048

효율적인 유체 시뮬레이션을 위한 투영 단계에서의 멀티 코어 프로세서와 그래픽 프로세서의 병렬처리

Parallel Processing of Multi-Core Processor and GPUs in Projection Step for Efficient Fluid Simulation

김선태, 정휘룡, 홍정모
동국대학교 컴퓨터공학과

Sun-Tae Kim(stkim@atelierj.pro), Hwi-ryong Jung(hrjung@atelierj.pro),
Jeong-Mo Hong(jmhong@dongguk.edu)

요약

최근 영화나 CF등에 사용되는 컴퓨터 그래픽스(Computer Graphics, 이하 CG)분야의 유체 시뮬레이션에서는 CPU와 GPU를 혼합하여 사용하는 기술들이 소개되고 있다. 본 논문에서는 유체 시뮬레이션 수행을 위한 투영 단계에서 멀티쓰레드를 이용하여 기존의 CPU와 GPU 간의 작업을 순차적으로 수행하던 방식을 개선하여 CPU와 GPU 간의 작업을 병렬처리 방법을 제시하였다. 제시된 방법을 통해 많은 계산량을 필요로 하는 유체시뮬레이션의 효율성을 높일 수 있었다.

■ 중심어 : | 유체 시뮬레이션 | 병렬 처리 | 멀티쓰레딩 |

Abstract

In these days, the state-of-art technologies employ the heterogeneous parallelization of CPU and GPU for fluid simulations in the field of computer graphics. In this paper, we present a novel CPU-GPU parallel algorithm that solves projection step of fluid simulation more efficiently than existing sequential CPU-GPU processing. Fluid simulation that requires high computational resources can be carried out efficiently by the proposed method.

■ keyword : | Fluid Simulation | Parallel Processing | Multithreading |

Ⅰ. 서 론

컴퓨터 그래픽스(Computer Graphics, 이하 CG)분야에서 유체 시뮬레이션 기술은 미리 정의된 조건에 수학과 물리 방정식에 의해 수치적 해를 계산하여 특정한 자연 현상을 현실성 있게 묘사하는 기술이다.
특히 자연법칙에 합당하며 시각적으로도 만족할 만한 고품질의 시뮬레이션을 수행하기 위해서는 수치상으로 일정 수준 이상을 만족하는 해를 구하는 과정이 필요하다. 이 과정에서 컴퓨터 자원을 이용한 많은 연산과정은 필수적이며, 정확한 해에 근접한 값을 요구할수록 필요한 연산의 양은 비례하여 증가하게 된다. 하지만 컴퓨터 계산 능력과 자원은 한계가 있기 때문에 사용할 수 있는 시뮬레이션 품질에는 제약이 존재한다. 이 제

* 본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2012년도 콘텐츠 산업기술지원사업과 한국과학재단 2012년도 일반 연구자지원사업의 연구결과로 수행되었음(No. NRF-2011-0023134).

접수번호 : #130222-004
접수일자 : 2013년 02월 22일
심사완료일 : 2013년 06월 11일
교신저자 : 홍정모, e-mail : jmhong@dongguk.edu

약을 극복하여 주어진 자원 안에서 목적에 맞는 시뮬레이션 결과를 얻기 위한 다양한 연구들이 진행되었다.

최근에는 하드웨어 분야에서 중앙 처리 장치(Central Processing Unit, 이하 CPU)는 단일 코어 시스템에서 2개 이상으로 구성된 멀티 코어 시스템으로 발전하였고, 그래픽스 처리 장치(Graphic Processing Unit, 이하 GPU)는 병렬화를 목적으로 최적화하여 설계된 코어들로 이루어진 다중 코어(Many-Core)시스템으로 발전하고 있다. 이 같은 하드웨어의 발전방향은 단일 시스템 환경에서는 제한되었던 고품질의 시뮬레이션가능하게 하였고 관련 연구들이 활발히 진행되는 계기가 되었다.

또한 기존에는 하드웨어의 제약 및 유체 시뮬레이션 기법의 한계로 실시간 유체 시뮬레이션을 수행하기 어렵거나 수행을 하더라도 시각적인 품질이 많이 낮아지는 단점이 있었다. 하지만 최근에는 멀티 코어 CPU와 GPU를 활용하여 시각적 품질을 유지하면서 실시간 유체 시뮬레이션을 효율적으로 수행하는 연구들이 소개되었다[1-3]. Ihmsen 등은 단일 CPU에서의 한계를 극복할 수 있는 다중 CPU를 사용한 유체 시뮬레이션 기법[1]을 소개하였고 Zhang 등은 다중 GPU를 활용하여 유체 시뮬레이션을 수행한 연구를 소개하였다[2].

Nakasato 등은 CPU와 GPU를 활용하여 파티클 시뮬레이션을 효과적으로 수행하는 방법[4]을 제안하였고 Domínguez 등은 CPU-GPU 혼합 클러스터에서 파티클 시뮬레이션을 수행하였다[5].

이와 같은 연구 흐름에서 주어진 자원을 최대한 활용하기 위하여 다중 격자(Multi-Grid) 방식에서 각 단계(level)에서 수행되는 다중 CPU와 GPU 사이의 순차적 작업 진행 방식을 개선하여 병렬처리로 작업을 진행하는 기법을 제안하고, 실제 실험을 통해서 성능향상 정도를 확인한다.

II. 유체 시뮬레이션

유체 시뮬레이션 중에서 많이 활용 되고 있는 액체 시뮬레이션은 Foster 등이 Navier-Stokes 3차원 방정식을 액체 시뮬레이션에 적용하는 방법을 제안하여 활용분야를 확장하였으며[6], Stam은 효율적으로 시뮬레이션의 수치적 안정성을 증가시키기 위해서 외재적 적분기법(Explicit integration methods) 대신 Semi-Lagrangian 기법 등의 암시적 기법을 적용하여 빠른 시간에 시뮬레이션이 가능하도록 하였다[7]. 2001년 Foster와 Fedkiw는 시뮬레이션 할 때 Semi-Lagrangian 방법과 효과적인 선형풀이 방법인 공액구배법(Conjugate gradient)을 사용하고, 입자와 경계면으로부터 거리에 따른 부호를 갖는 거리 함수로 정의된 레벨셋(Levelset)을 같이 이용하여 물의 표면을 추적 및 표현함으로써 시뮬레이션의 품질을 높였다[8].

유체 시뮬레이션은 크게 4단계로 구성되며 초기화 단계 이후 3단계가 반복되며 진행된다.

첫째 이류(Advection)단계에서는 시뮬레이션 공간상에 정의된 현재 속도장을 이용하여 주어진 밀도장, 온도장, 레벨셋, 속도장 등을 이동 시키는 과정이다.

둘째 소싱(Sourcing)단계에서는 유체의 밀도나 속도 등이 발생하는 근원(Source)에 해당하는 위치에 밀도, 속도 등을 새로 소싱하는 과정이다.

마지막으로 투영(Projection)단계에서는 현재 진행중인 시뮬레이션 시간상에서 유체 공간상의 매질들의 밀도, 속도 데이터들로 인해 발생하는 압력을 먼저 구하고, 이 압력에 의해 발생하는 속도를 이용하여 기존 속도에 갱신하는 과정이다[9]. [그림 1]은 유체 시뮬레이션 진행이 시뮬레이션 초기화, 이류, 소싱, 투영 단계 등으로 진행되는 흐름을 보여 주고 있다. 본 논문에서는 4번째 단계인 투영단계에서 수행되는 부분을 CPU와 GPU의 병렬처리 기법을 적용 하고자 한다.

유체 시뮬레이션에서 투영부분은 유체의 비압축성을 유지하기 위해 필요한 연산이다. 이 부분은 주어진 조건들을 이용하여 행렬(Matrix)을 구성하고 반복하여 해를 구하는 방식으로 진행되기 때문에 많은 연산을 필요로 한다. 또한 시뮬레이션의 해상도가 증가할수록 조건을 만족하는 해를 구하기 위한 행렬의 크기도 커지기 때문에 고품질의 시뮬레이션 결과를 위한 고해상도 시뮬레이션 수행 시 많은 시간이 소요된다.

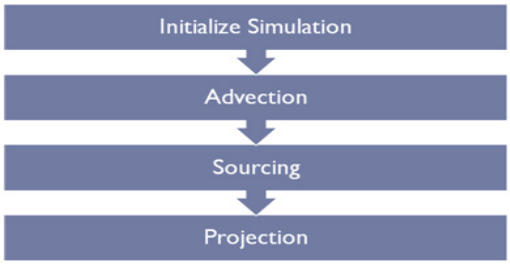


그림 1. 유체 시뮬레이션 주요 과정도

최근에는 이 투영단계를 효율적으로 수행하기 위해서 다중격자(Multi-grid)방식이 많이 사용되고 있다. 다중격자 방식은 압력장 등의 시뮬레이션에 관련된 정보가 저장된 기본 해상도의 격자를 미리 정의된 단계(Level)별로 가우스 자이델 완화(Gauss-Seidel smoothing) 방식 등을 이용하여 해당하는 격자의 해에 근사한 값을 구하고 다운 샘플링(Down Sampling)하는 제한(Restriction) 단계를 진행한다. 가장 낮은 해상도에 해당하는 단계(Coarse Level)에서는 비교적 정확하게 직접 해를 구한 후 제한과정과 반대되는 확장(Prolongation)을 수행하며 저해상도에서 고해상도 방향으로 진행하게 된다. 이때 제한과 확장 과정이 V자 모양으로 진행하는 V-Cycle 방법이나 제한과 확장을 두 차례 이상 반복하며 W형태로 진행하는 W-Cycle 방식으로 진행하여 특정한 조건을 만족하는 해를 빠르게 구할 수 있게 된다[10].

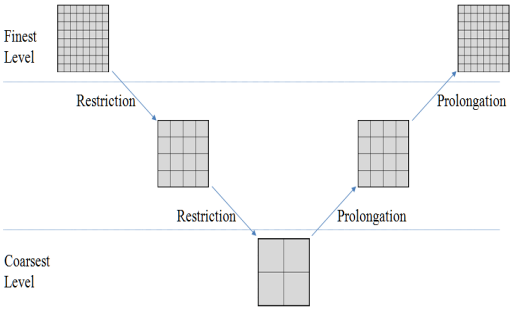


그림 2. 일반적인 Multigrid 기법 V Cycle 과정도

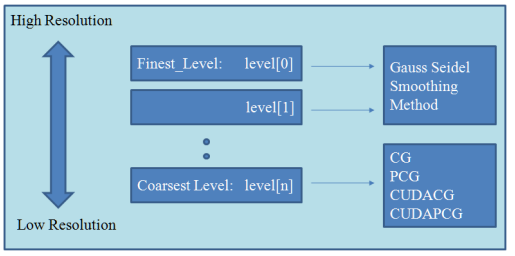


그림 3. 각 단계(Level)별 수행 개념 도식화.

[그림 2][그림 3]에서는 시뮬레이션의 기본 해상도를 미리 정의된 여러 개의 해상도로 다운샘플링 한 후 각 단계별로 가우스-자이델 완화 기법 이나 공액구배법 방식 등으로 해를 구하는 것을 도식화하여 나타내고 있다. 정밀 단계(Finest Level)에서는 가장 고해상도이며 시뮬레이션의 기본 해상도를 나타내고 있으며, 거친 단계(Coarse Level)는 미리 정의된 다중 격자 해상도 중에서 가장 낮은 해상도를 갖는 단계를 나타내고 있다. 정밀 단계는 가장 높은 해상도를 갖고 있기 때문에 공액구배법 방식 등을 이용하여 직접적으로 정확한 해를 구하기 위해서는 많은 시간이 소요된다. 따라서 분리된 단계 중에서 비교적 높은 해상도를 갖는 단계에서는 가우스-자이델 완화 기법을 사용하여 근사한 해를 찾는다. 이에 반해 낮은 해상도를 갖는 단계에서는 연산 수행시간이 비교적 크지 않기 때문에 CPU에서 공액구배법을 수행하거나 GPU에서 고속으로 공액구배법(CUDACG)을 수행하여 정확한 해를 계산할 수 있다.

Ⅲ. CPU/GPU 병렬 처리 기법

실험에 사용된 시뮬레이션 엔진에서는 멀티 코어의 멀티 쓰레딩을 이용하여 시스템을 구성하고 CPU의 사용률을 높였다. 그리고 낮은 해상도를 갖는 단계에서는 다중코어로 구성된 GPU를 사용하여 시뮬레이션의 비압축성을 유지하는데 필요한 푸아송 방정식의 해를 구함으로써 CPU에서 해를 구하는 것 보다 빠르게 시뮬레이션을 수행한다.

실험에 사용된 다중격자 방식을 수행할 때에는 낮은 해상도 단계를 제외한 나머지 단계에 대해서는 CPU를

사용한 가우스 자이델 완화 연산을 하고, 낮은 해상도를 갖는 단계에서는 공액구배법을 이용하여 비교적 정확하게 해를 찾게 된다. 시뮬레이션의 해상도를 n단계로 분리 했을 때 가장 낮은 단계를 GPU에서 수행할 수도 있다. [그림 3]에서 단일 GPU로 구성된 시스템의 경우 가장 높은 해상도를 갖는 level[0]부터 level[n-1]단계까지 CPU의 작업이 진행된 이후 level[n] 단계는 GPU에서의 작업이 진행되는 순차적인 작업이 이루어진다. 기존 방식에서는 GPU에서 작업을 수행하는 동안 CPU의 쓰레드들은 대기상태에 머물러 있어야 했다. 이는 곧 자원의 활용도를 떨어뜨리고, 전체적인 시뮬레이션 수행시간을 증가시킨다.

본 논문에서는 CPU에서 생성된 쓰레드들이 GPU에서 공익구배법으로 정확하게 해를 찾는 동안 CPU자원이 대기상태에 머무는 것을 개선하기 위해서 GPU 작업을 별도의 쓰레드로 생성하여, CPU와 GPU에서 시뮬레이션에 필요한 작업이 병렬적으로 이루어지는 방법을 제시하였다.

제안한 방법에서는 우선 CPU에서 생성된 쓰레드들은 시뮬레이션 필요한 계산량이 많은 작업에 들어가기 전에 GPU에서 작업을 병렬적으로 수행 할 수 있도록, 별도의 쓰레드를 생성하여 시작하게 만든다.

CPU에서 생성된 쓰레드들은 GPU작업을 수행할 별도의 쓰레드와 별개로 자신에게 주어진 일인 가우스 자이델 완화 기법 등을 진행하게 된다. [그림 3]에서 단일 GPU 시스템에서는 level[n]단계를 수행하는 쓰레드를 생성 및 수행하고, 다중 GPU 시스템에서는 level[n]과 level[n-1]단계를 GPU에서 수행하기 위한 쓰레드들을 생성 및 수행을 시작하고 나머지 단계들은 CPU에서 작업을 병렬적으로 수행하게 된다.

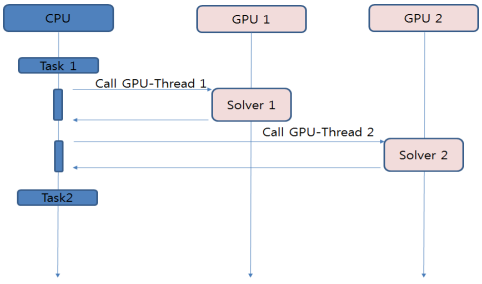


그림 4. 기존의 CPU와 다중 GPU 순차 처리 흐름도

[그림 4]은 다중 GPU 환경에서 일반적으로 정의된 알고리즘을 순차적으로 진행하는 것을 보여준다. [그림 4]에서 제시된 시스템에서는 CPU에서 진행되는 작업(Task1)을 시작하고 종료된 것을 확인 한 후에 GPU에서 수행할 작업(Solver1)을 호출하고, 작업(Solver1)이 완료될 때까지 CPU의 쓰레드는 대기 상태에 머문다. 이후 GPU1에서 작업(Solver1)이 종료되면서 CPU 작업 쓰레드는 다음단계에 필요한 작업(Solver2)을 GPU2에 호출하게 된다. 그 전 단계와 마찬가지로 GPU2에서 수행되는 작업(Solver2)이 끝날 때까지 CPU의 작업 쓰레드는 대기하게 되는데, 이때 쓰레드에 할당되어 있던 CPU와 메모리들의 자원들은 다른 연산을 하지 않기 때문에 시스템의 전체 사용률은 줄어들게 된다.

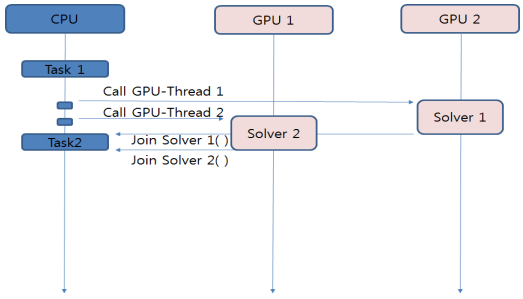


그림 5. 본 논문에서 제안한 CPU와 다중 GPU 순차 처리 흐름도

본 논문에서 제안한 방식인 [그림 5]의 경우 각 GPU 작업을 관리할 쓰레드를 2개 생성한 후 각 각 Solver1과 Solver2을 GPU1과 GPU2에서 수행 하도록 호출하고 곧 바로 CPU의 쓰레드들은 바로 다음 작업(Task2)를 수행하게 된다.

이 같은 처리 방식은 CPU와 GPU의 작업이 병렬적으로 진행되기 때문에 기존방식에서는 Task2, Solver1, Solver2의 각 수행시간을 합한 시간이 필요했지만 제안한 방법에서는 각 수행시간 중 가장 오래 걸리는 작업의 속도에 근접한 수행 시간을 갖는다. GPU의 작업이 종료된 이후에 다음 단계의 연산이 가능할 때 Join( )함수를 사용함으로써 병렬화 할 때 만든 쓰레드의 작업 종료 시점까지 기다릴 수 있다. 이 같은 과정을 통해서 CPU와 GPU에서 작업이 병렬로 처리되기 때문에 기존

방식에서 CPU가 GPU의 작업이 끝날 때까지 대기하던 시간을 방지할 수 있으며, 이론적으로 그 시간만큼의 시뮬레이션의 성능 향상을 기대할 수 있다.

Ⅳ. 실 험

제안한 병렬처리 기법을 다중격자(Multi-grid)기법을 사용하는 격자기반 시뮬레이션에 적용하여 실제 수행 성능의 향상 여부를 실험을 통하여 확인 하도록 한다. 실험을 수행하기 위해 다중 코어를 지원하는 시스템에서 진행되었다.

CPU는 물리적인 코어 12개를 제공하는 Intel xeon x5690(3.47GHz)을 사용하였고, GPU는 512개의 코어를 제공하는 Nvidia GTX580를 사용하였고 고해상도의 시뮬레이션을 위해서 Ram 96GB을 사용하였다. 또한 CPU의 다중 쓰레드 및 GPU 제어 쓰레드 관리를 위해서 Boost 라이브러리1.48 등을 사용하였다.

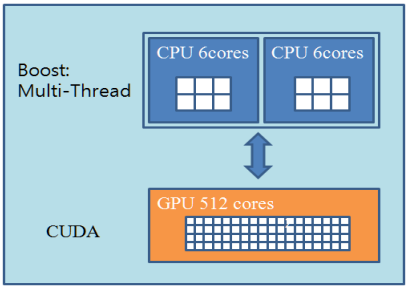


그림 6. 단일 시스템에서의 CPU-단일GPU 구성도

Ⅴ. 결 과

실험은 유체 시뮬레이션의 격자 해상도를 변경하며 기존의 수행 방법과 병렬처리 기법을 적용했을 때의 수행속도차이를 비교하는 방식으로 진행하였다.

[그림 7]에서는 기존의 순차적 수행 방식과 대비하여 병렬처리 방식을 사용했을 때 수행속도의 향상 정도를 나타내고 있다. 파란색 막대 그래프는 시뮬레이션의 해상도 256^3과 512^3에서의 1프레임당 걸린 수행시간을 나타내고 있으며, 빨간색과 초록색 막대 그래프는 병렬처리 방식을 사용했을 때 수행시간을 나타내고 있다. 빨간색은 시뮬레이션 해상도를 n으로 분해 했을 때 level[n]과 level[n-1] 단계를 단일 GPU에서 수행하며, 초록색 그래프는 다중 GPU에서 동시에 처리한 것을 보여주고 있다. 즉 [그림 7]에서 확인할 수 있듯이 병렬처리 방식을 사용하였을 때 각 256^3 해상도에서 단일 GPU 사용시 1.54배, 다중 GPU 사용시 1.71배 512^3 해상도에서 단일 GPU 사용시 1.62배 다중 GPU 사용시 1.86배의 수행 속도 향상이 있는 것을 확인할 수 있었다.

Ⅵ. 결 론

본 논문에서는 제한된 하드웨어의 자원을 이용하여 효율적인 시뮬레이션 수행을 하기 위한 연구를 진행하였다. 특히 시뮬레이션 수행 중 많은 시간이 소비되는

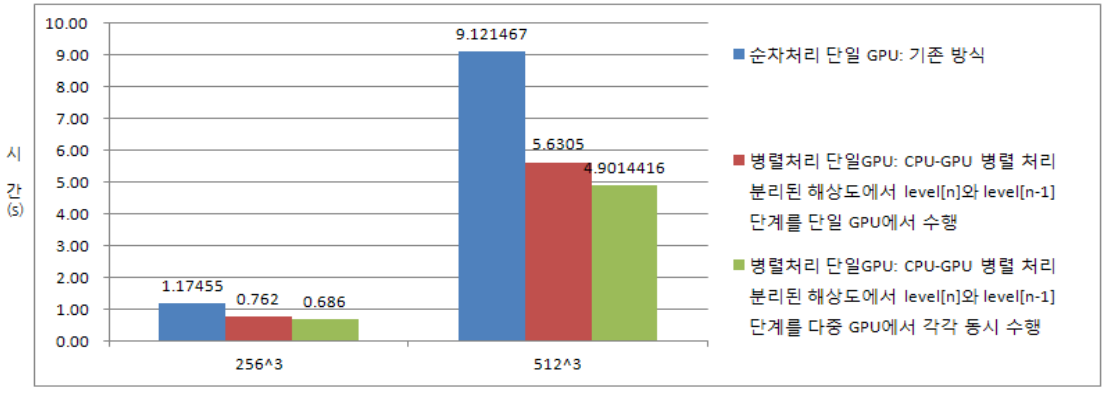


그림 7. 순차적 수행 방식 대비 병렬처리 방식 성능 비교표

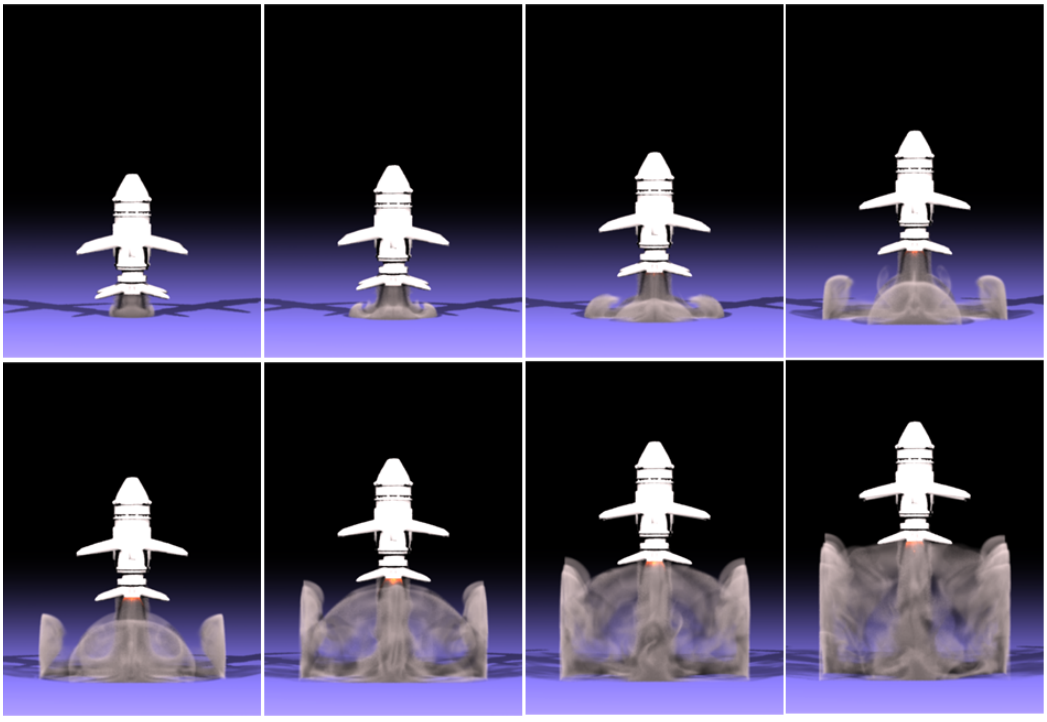


그림 8. 256^3 해상도에서 멀티 쓰레드를 이용한 CPU-GPU 병렬처리가 적용된 로켓 발사 시뮬레이션

투영단계를 빠르게 수행 하는 것을 목적으로 하였다.

투영 단계에서 기존의 CPU-GPU 간의 계산 작업의 순차적 수행 방식을 개선하여 CPU-GPU 간의 계산 작업의 병렬화를 구현하였다. 이를 위해서 GPU의 작업을 제어할 별도의 쓰레드를 사용하였으며, 이론적으로 기대했던 부분에서 성능향상이 이루어진 것을 몇 가지의 실험을 통하여 [그림 7]에서 확인할 수 있었다. 또한 병렬로 처리되어도 이상 없이 시뮬레이션이 진행되는 것을 [그림 8]을 통해 확인할 수 있었다.

마지막으로 시스템을 구성하는 다중 GPU의 수가 증가할수록 이 기법의 사용으로 성능 향상 폭이 증가할 것으로 기대된다.

참 고 문 헌

[1] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner, "A Parallel SPH Implementation on Multi Core CPUs," Computer Graphics Forum, Blackwell Publishing Ltd, Vol.30, No.1, 2011.

[2] F. Zhang, L. Hu, J. Wu, and X. Shen, "A SPH-based method for interactive fluids simulation on the multi-GPU," Proceedings of the 10th international conference on virtual reality continuum and its applications in industry, ACM, 2011.

[3] K. Hegeman, N. Carr, and G. Miller, "Particle-based fluid simulation on the GPU," Computational Science - ICCS 2006. Springer Berlin Heidelberg, pp.228-235, 2006..

[4] N. Nakasato, G. Ogiya, Y. Miki, M. Mori, and K. Nemoto, "Astrophysical Particle Simulations on Heterogeneous CPU-GPU Systems," arXiv preprint arXiv:1206.1199, 2012.

[5] J. M. Domínguez, A. J. C. Crespo, M. Gómez-Gesteira, D. Valdez-Balderas, and B. D. Rogers,

"New OpenMP-MPI-CUDA implementation for parallel SPH simulations on heterogeneous CPU-GPU clusters," 7th SPHERIC, pp.226-275, 2012.

[6] N. Foster and D. Metaxas, "Realistic animation of liquids," Graphical models and image processing, Vol.58, No.5, pp.471-483, 1996.

[7] J. Stam, "Stable fluids," Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1999.

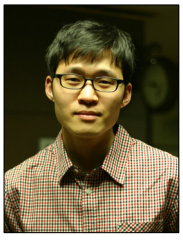
[8] N. Foster and R. Fedkiw, "Practical animation of liquids." Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, 2001.

[9] 김대영, *무발산 와동 입자와 슈퍼 샘플링 기법을 이용 한 레벨셋 기반 화염 시뮬레이션의 개선*, 동국대학교 석사학위 논문, 2011

[10] J. Bolz, L. Farmer, E. Grinspun, and P. Schröoder, "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," ACM Transactions on Graphics (TOG), Vol.22, No.3, ACM, 2003.

저 자 소 개

김 선 태(Sun-Tae Kim) 준회원



- 2010년 8월 : 동국대학교 컴퓨터공학과(공학사)
- 2012년 8월 : 동국대학교 컴퓨터공학과 석사 수료
- 2012년 8월 ~ 현재 : 동국대학교 엔터테인먼트 컴퓨팅 연구센터 연구원 재직 중

<관심분야> : 유체 시뮬레이션, GPU, VFX

정 휘 룡(Hwi-ryong Jung) 정회원

- 2005년 8월 : 아주대학교 미디어학부(공학사)
- 2007년 2월 : 한국과학기술원 전산과 문화기술학제전공(공학석사)
- 2007년 2월 ~ 현재 : 한국과학기술원 문화기술대학원 박사과정

<관심분야> : Fluid simulation, cloth simulation, parallel computing

홍 정 모(Jeong-Mo Hong) 정회원

- 2000년 2월 : 한국과학기술원 기계공학(공학사)
- 2002년 2월 : 한국과학기술원 기계공학(공학석사)
- 2005년 8월 : 고려대학교 전산학(전산학박사)
- 2005년 8월 : Standford Unversity Post-doctoral
- 2008년 8월 ~ 현재 : 동국대학교 컴퓨터공학과 조교수

<관심분야> : Visual simulation, visual special effects, ray-tracing, geometric shock waves, parallel computing