KITTEN: 다중 스레드 가상현실 시스템
(KITTEN: A Multi-thread Virtual Reality System)

김 대 원 † 이 선 우 † 원 광 연 †† 이 광 형 †††
(Daewon Kim) (Sonou Lee) (Kwangyun Wohn) (Hyung Lee-Kwang)

요 약 가상현실 시스템은 참여자에게 자연스러운 상호작용, 몰입감과 더불어 사실적인 영상을 생성해 주어야 한다. 이를 위해서는 가상 세계의 복잡도나 시뮬레이션의 복잡도에 비교적 영향을 받지 않으면서 균일하며 빠른 렌더링 속도를 제공하는 것이 중요하다. 본 논문에서는 가상현실 시스템을 구성하는 하부 기능인 상호작용, 시뮬레이션, 렌더링 모듈을 다중 쓰레드 방식으로 구성하고 이를 병렬 수행시키는 시스템을 설계하고 구현하였다. 이렇게 함으로써 렌더링 모듈의 수행이 상호작용, 시뮬레이션 모듈의 수행과 독립적으로 이루어질 수 있도록 하였다. 따라서 제안된 시스템은 가상 객체들 간의 시뮬레이션 태스크가 복잡한 가상 세계 어플리케이션에서 보다 균일하고 빠른 렌더링 속도를 제공하게 된다. 본 논문에서는 제안된 시스템을 장면 복잡도와 시뮬레이션 복잡도가 높은 어플리케이션을 선정하여 그 성능 향상을 실험을 통하여 검증하였다.

Abstract A virtual reality system must provide participants with a natural interaction, a sufficient immersion, and mostly, realistic images. To achieve this, it is crucial to provide a fast and uniform rendering speed regardless of the complexity of virtual worlds, or the complexity of simulation. In this paper, a virtual reality system which offers an improved rendering performance for complex virtual reality applications has been designed and implemented. The key idea of the proposed system is to exploit the multi-thread scheme in system module design, and execute each modules in parallel. Taking such design approach, rendering, simulation, and interaction can be executed independently. Hence, in applications where a simulation is complex or a scene is very large, this system can provide a more uniform and faster frame rates. The proposed method has been experimented under the various application environments in which scenes and simulations are very complex.

1. 서 론

가상 현실(Virtual Reality, VR)이란 컴퓨터가 만들어낸 가상의 세계를 사용자에게 다양한 감각 채널을 통해 제공함으로써, 사용자로 하여금 생성된 가상 세계에 몰입함과 동시에, 가상 세계 내에서 현실 세계에서와 같은 자연스러운 상호작용을 가능하도록 하는 제반 기술과 이러한 기술에 필요한 이론적 바탕을 지칭한다[1]. 이와 같은 가상 현실을 실현시키기 위해서는 가상 환경을 시뮬레이션 해주는 가상현실 시스템이 필요하다. 가상현실 시스템은 사용자가 원하는 가상 세계를 제작하고 이를 시뮬레이션하며, 제작된 가상 세계 내에서 자연스럽게 상호작용할 수 있는 기능을 제공해야 한다. 최근 들어 가상 현실에 대한 관심의 증가에 힘입어 복잡한 가상 세계 어플리케이션을 수행하는데 있어 우수한 성능을 제공하기 위한 많은 연구가 이루어지고 있다[2][3][4][5][6][7][8][9][10][11].

일반적으로 언급되는 가상현실 시스템의 성능의 척도 중 가장 중요한 요소로서 시스템의 렌더링 속도, 즉 화면 갱신율(frame rates)이 있다. 어플리케이션 수행시 장면 복잡도나 시뮬레이션 복잡도에 비교적 영향을 받지 않으면서 빠른 렌더링 속도를 제공하는 것은 참여자에게 현

· 본 연구의 일부는 과학기술정책 연구소 및 서울대 자동제어특화연구센터의 지원에 의하여 수행되었음. ETRI의 네트워크 가상 현실 개임을 위한 기반 기술 및 소프트웨어 개발의 연구비 지원하에 연구되었음.

† 비 회 원 : 한국과학기술원 전자전산학과
dwkim@monami.kaist.ac.kr
solee@vr.kaist.ac.kr
†† 정 회 원 : 한국과학기술원 전자전산학과 교수
wohn@cs.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
khlee@cs.kaist.ac.kr
논문접수 : 1999년 7월 16일
심사완료 : 2000년 4월 26일

실감과 몰입감을 주는 데 있어 매우 중요한 요소이다. 기존의 시스템에서는 가상 세계 운영을 위한 하부 기능인 시뮬레이션 기능과 렌더링 기능을 순차적으로 수행시키는 방식이 주로 사용되고 있다. 이러한 시스템은 시뮬레이션의 복잡도가 높은 환경에서는 시뮬레이션 모듈의 병목 현상으로 인해 전체 시스템의 렌더링 성능을 저하시키는 단점을 지닌다[3][4][5][6]. 이에 따라 시뮬레이션과 렌더링 기능을 분리시켜 수행하는 시스템들도 개발되었다. 그러나 이러한 시스템 역시 사용자와의 상호작용 기능을 시뮬레이션이나 렌더링 어느 한 쪽 모듈에 포함시켜 수행시킴으로써, 시뮬레이션이나 렌더링 기능 수행 시 발생하는 사용자의 입력 갱신을 처리하지 못하는 단점을 지닌다[7][8][9][10][11].

본 논문에서는 가상현실 어플리케이션 수행 시, 가상 세계의 복잡도나 시뮬레이션의 복잡도에 비교적 영향을 받지 않으면서 빠른 렌더링 속도를 제공하는 시스템인 KITTEN을 설계하고 구현하였다. 이를 위한 접근 방법으로 가상 세계 운영을 위한 하부 기능인 사용자 상호작용 기능, 시뮬레이션 기능, 렌더링 기능을 다중 스레드로 구성하고 각 기능을 병렬적으로 수행하도록 하였다. 이렇게 함으로써 두 가지의 장점을 갖게 된다. 첫째, 시뮬레이션 기능과 렌더링 기능이 독립적으로 분리되어 병렬 수행하게 된다. 따라서 가상 세계 어플리케이션의 복잡도, 특히 시뮬레이션의 복잡도에 비교적 영향을 받지 않으면서 향상된 화면 갱신율을 제공할 수 있다. 둘째, 상호작용 기능을 다른 기능과 분리시켜 수행함으로써 시뮬레이션이나 렌더링 수행 시, 사용자의 입력 갱신을 빠르게 처리할 수 있다. 즉, 마우스나 키보드 등의 입력 센서 값 갱신에 따른 가상 세계의 응답 지연(response lag)을 감소시키는 장점을 가진다.

2. 관련연구

2.1 기존의 가상현실 시스템

가상세계 참여자는 실제 환경과 유사하게 만들어진 인공적 세계 내에서 시각, 청각, 촉각 등과 같은 다양한 감각 기관들을 이용하여 그 속에서 정의된 객체와 상호작용하며 다양한 경험과 정보를 주고받는다. 이와 같은 가상현실을 실현시키기 위해서는 가상 세계를 시뮬레이션해주는 컴퓨터 시스템이 필요한데, 이를 가상현실 시스템이라고 한다. 가상현실 시스템은 사용자가 원하는 가상 세계를 생성하고 생성된 가상 세계 내에서 사용자들이 자유롭게 가상 세계를 둘러보고 조작할 수 있도록 효과적인 상호작용과 시뮬레이션을 제공하게 된다. 따라서 가상현실 시스템은 가상환경을 만들고 운영하기 위한 다양한 기능들을 제공하게 되는데, 이러한 기능들을 구현하는 방식과 초점이 되는 기능에 따라서 다양한 시스템이 제작될 수 있으며, 지금까지 많은 종류의 시스템들이 소개되었다[2][3][4][5][6][7][8][9][10][11].

가상현실 시스템의 설계 시 중요한 고려 사항은 전체 시스템의 수행 방식을 결정하는 것이다. 수행 방식의 결정은 시스템의 렌더링 속도를 결정하는데 매우 중요하며, 본 논문에서는 이에 초점을 둔다. 지금까지 제안된 대표적인 수행 방식은 단일 시뮬레이션 루프 방식과 다중 프로세싱을 이용한 병렬 수행 방식의 두 가지로 나눌 수 있다.

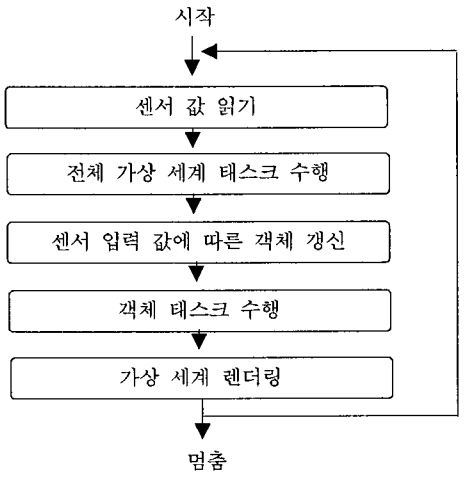

그림 1 단일 시뮬레이션 루프 방식의 수행

단일 시뮬레이션 루프 방식을 채택하는 시스템에서는 어플리케이션의 수행이 하나의 메인 루프를 순환하면서 이루어진다. 즉, 가상 세계 시뮬레이션 루프를 한바퀴 돌 때마다 상호작용 모듈에서 사용자의 입력을 검사하고 시뮬레이션 모듈에서 이를 계산해서 가상 세계에 반영한다. 그리고 렌더링 모듈에서 시뮬레이션된 결과를 사용자에게 그려주는 단일 루프 방식으로 전체 시스템이 수행되는 것을 의미한다. 단일 시뮬레이션 루프 방식으로 설계된 가상현실 시스템의 수행 과정은 그림 1과 같은 형태를 취하며, 대표적인 시스템으로는 *WorldToolKit*™, *SuperScapeVRT*™ 시스템이 있다[4][6].

다른 설계 방식으로서 다중 프로세싱을 이용한 병렬 수행 방식이 있다. 이 방식은 가상현실 시스템을 이루는 모듈들을 다중 프로세싱(multi-processing) 방식으로 설계하고 이를 병렬적으로 수행함으로써 시스템의 전체적 성

능을 향상시키는 방법이다. 모듈들을 다중 프로세싱으로 설계하고 병렬 수행하는 접근 방식은 크게 두 가지로 분류될 수 있다. 첫째, 렌더링 파이프라인을 논리적으로 여러 개의 독립된 모듈로 나누고 각 모듈들을 병렬 수행시킴으로써 렌더링 속도를 중점적으로 향상시키는 방식이 있다. 이 방식의 대표적인 시스템으로는 *Performer*™와 *Argus*™가 있다[9][10]. 둘째, *Virtual Windtunnel, Alice* 시스템과 같이 시스템의 상위 레벨에서 시뮬레이션(계산) 부분과 렌더링(그래픽스) 부분을 다중 프로세싱으로 설계하여 시뮬레이션과 렌더링을 독립적으로 수행시키는 방식이 있다[7][11][13]. 이들 시스템에서는 다중 프로세싱 방법으로 light-weight 프로세스(스레드)를 사용하였으며 이는 본 논문의 시스템 디자인 접근 방법과 유사하다.

2.2 기존 시스템의 문제점

본 절에서는 기존의 시스템에서 채택하고 있는 수행 방식의 문제점에 대해서 기술한다. 먼저 단일 시뮬레이션 루프 방식의 경우 직관적인 이해가 쉬우며 구현이 용이하다는 장점이 있어, 현재 *WorldToolKit*™ 을 비롯한 많은 시스템에서 채택하고 있다[3][4][5][6][12]. 그러나 루프에 의한 순차적 수행 방법은 각 모듈에 걸리는 부하의 양에 따라 전체적인 시스템 성능이 저하되는 단점이 존재한다. 어플리케이션 수행 중 시뮬레이션이나 렌더링 어느 한쪽 모듈에 계산 부하가 많이 작용될 경우, 시스템의 전체적인 성능에 영향을 미치게 되어 렌더링 속도를 현저히 떨어뜨리게 된다. 따라서 실시간 렌더링과 같이 빠른 수행 속도를 제공하는 데 어려움이 있다. 특히 유체역학, 충돌검사와 같이 객체의 동적 시뮬레이션이 복잡해질 경우, 시뮬레이션 모듈에 시스템 성능의 큰 병목 현상이 발생하게 되어 많은 렌더링 성능 저하를 초래한다.

다중 프로세싱을 채택한 시스템들은 단일 시뮬레이션 루프 방식의 단점을 보완하기 위해서 제안되었다고 볼 수 있다. 따라서 단일 시뮬레이션 루프 방식에 비해 향상된 렌더링 성능을 제공하지만, 이 역시 단점을 지닌다. 대표적 시스템인 *Virtual Windtunnel*과 *Alice*를 예로 들어보자[7][11]. 이 시스템들은 계산을 담당하는 시뮬레이션 기능과 그래픽을 담당하는 렌더링 기능을 스레드(light-weight process)를 이용하여 병렬로 수행하는 구조로 설계되었으며, 상호작용 기능은 시뮬레이션에 포함되어 처리된다. 따라서 시뮬레이션과 렌더링간의 병목현상은 두드러지지 않지만, 가상현실 시스템을 이루는 다른 핵심기능인 상호작용 기능이 시뮬레이션 모듈에 제약을 받게 된다. 다시 말해서 시뮬레이션이 수행 중일 때 참여자의 갱신된 입력을 처리하지 못하는 단점을 지니는 것이다. 이러한 점은 시뮬레이션과 상호작용이 강조되는 어플리케이션에서는 참여자에게 충분한 몰입감을 줄 수 없으며, 이에 본 논문에서는 이에 대한 보완점을 제안하는 시스템 디자인에 반영하였다.

위의 단점 외에도 일반적으로 다중 프로세싱 방식을 이용할 경우 여러 가지 고려해야 할 사항들이 있다. 우선 시스템의 구현이 어렵고, 운영체제가 다중 프로세스 개발 및 수행 환경을 제공해 주어야 한다. 또한 프로세스간의 데이타 이동에 따른 공유 및 일관성 문제와, 다른 플랫폼으로의 이식도 어려운 단점을 지닌다. 그리고 시스템 설계 시 프로세스간의 통신 메커니즘과 가상 세계에서의 시간 개념 및 부하 균형 등에 대한 디자인 이슈가 발생한다. 이들 각 요소는 서로 연관이 없는 상호 독립적인 문제가 아니라, 서로가 긴밀한 관계에 놓여 있어서 신중한 선택이 요구된다. 이에 대한 본 논문의 접근 방법은 먼저 다중 프로세스 개발 환경을 제공하기 위하여 시스템의 플랫폼으로 다중 CPU를 갖는 Silicon Graphics Octane™을 사용하였다. 또한 이렇게 개발된 시스템이 다른 플랫폼으로 쉽게 이식될 수 있도록, 객체 지향 프로그래밍 기법과 렌더링 기능을 업계 표준인 OpenGL을 이용하여 작성하였다. 이와 같이 함으로써 다른 시스템 하드웨어로의 호환과 확장을 쉽게 할 수 있어 전체 시스템의 플랫폼 독립성을 추구하였다. 그리고 중요한 이슈인 프로세스간의 데이타 공유 및 일관성 문제는 다중 프로세스간의 메시지 전달 규약을 이용하여 해결하였다. 각 프로세스간에 필요한 공유 정보는 기 정의된 메시지의 형태로 관리되며 필요한 경우 메시지 큐를 이용하여 메시지를 주고 받음으로써, 병렬적으로 수행되는 다중 프로세스간에 정보 교환을 처리한다.

3. 모듈 병렬화 시스템

본 절에서는 논문에서 제안하는 시스템의 설계와 구조에 대해서 기술한다. 먼저 3.1절에서 제안하는 모듈 병렬화 방법과 그에 따른 장점에 대해서 살펴본다. 3.2절에서는 제안된 방법에 따른 전체적 시스템 구조에 대해서 기술한다. 그리고 3.3절에서는 제안된 시스템과 기존 시스템을 간략히 비교하여 본다.

3.1 모듈 병렬화 제안

일반적으로 가상현실 시스템의 설계에 있어서 가장 우선적으로 고려되어야 할 사항으로 시스템의 성능을 들고 있다[14]. 가상현실 분야에서 말하는 시스템 성능이란 대부분 렌더링 속도, 즉 화면 갱신율(frame rates)을 의미한다. 그리고 확장된 의미로 사용될 경우 사용자의 요구가 반영되어 결과로 나타나는 응답 속도의 의미를 포함하기도 한다. 다시 말하자면, 가상현실 시스템의 성능을 극대

화한다는 것은 렌더링 속도를 빠르게 하는 것과 동시에 균일한 속도로 가상 세계를 사용자에게 보여주는 것을 의미한다. 궁극적으로 가상 세계를 구성하는 장면의 복잡도나 시뮬레이션의 복잡도에 비교적 영향을 받지 않고 빠른 렌더링 속도가 균일하게 제공되는 것이 매우 중요하다.

본 논문에서 제안하는 가상현실 시스템의 목적은 가상 세계 어플리케이션의 복잡도에 비교적 영향을 받지 않는 향상된 렌더링 속도를 제공하는 것이다. 이를 위해 시스템을 구성하는 핵심 모듈을 다중 스레드로 구성하여 병렬 수행하도록 설계하였다. 이와 같이 함으로써 하나의 스레드 모듈에 순간적으로 많은 부하가 걸리더라도 나머지 스레드의 수행에는 제한을 가하지 않게 되어 시스템의 전체적인 성능 저하를 방지하였다. 각 스레드 모듈 - 상호작용, 시뮬레이션, 렌더링 - 은 비동기적으로 독립된 작업을 병렬 수행하며 최대의 성능을 제공하게 된다. 여기서 비동기적으로 병렬 수행되는 각 모듈간의 정보 교환은 중앙의 중계자를 통한 메시지 전달 방식을 채택하여 해결하였다.

이와 같이 모듈 병렬화 구조로 설계함으로써 단일 시뮬레이션 루프 시스템에서 발생하는 시뮬레이션 모듈의 병목현상으로 인한 렌더링 모듈의 지연을 방지할 수 있다. 시뮬레이션 모듈과 렌더링 모듈은 독립적으로 병렬 수행된다. 따라서 시뮬레이션의 복잡도에 비교적 영향을 받지 않는 렌더링 속도를 제공할 수 있게 된다. 특히 메시지 전달 방법을 이용한 비동기 병렬 수행 구조는 시뮬레이션과 렌더링간에 엄격한 동기화가 요구되지 않는 어플리케이션일 경우 최대의 렌더링 속도를 제공하게 된다. 또한 시뮬레이션 기능과 렌더링 기능의 병렬 수행에 따른 이점이외에도, 상호작용 기능도 시뮬레이션 기능으로부터 독립시켜 병렬 수행시킴으로써 기존의 다중 프로세싱의 단점을 보완할 수 있다. 이 역시, 시뮬레이션과 렌더링간의 관계와 유사하게 이해될 수 있으며, 시뮬레이션 수행 중 갱신되는 참여자 입력센서 값을 가상 세계에 빠르게 반영하게 되는 장점을 지닌다. 이러한 점은 가상현실 어플리케이션이 실시간성을 강조한다는 측면에서 매우 중요한 점이다.

3.2 제안된 시스템 구조

본 논문에서 제시하는 시스템 구조는 기 제시된 가상현실 시스템의 기능적 모델을 기반으로 재구성한 것이다[1]. 기능적 모델은 가상현실 저작과 운영에 필요한 핵심 기능들의 조합으로 구성된 하부 구조를 형성한다. 따라서 이러한 기능들을 구성하는 방식에 따라 실제 시스템 제작 시 그 구조가 결정된다. 본 논문에서 제안하는 가상현실 시스템인 KITTEN의 전체적인 시스템 구조는 그림 2와 같다.

KITTEN은 가상 세계의 하부구조를 기능별로 아바타(avatar), 시뮬레이터(simulator), 렌더러(renderer), 코디네이터(coordinator) 모듈로 나누었다. 그리고 모듈들을 다중 스레드 방식으로 구성함으로써, 각 모듈이 다른 모듈에 독립적으로 수행되는 병렬화 프로그래밍의 기법으로 설계하였다. 다중 스레드 방식의 시스템을 구성함으로써 시스템의 전체적인 성능이 어느 한 모듈에 종속되지 않도록 하였다. 이 점은 가상현실 시스템이 실시간성을 강조하는 시스템이라는 점에서 매우 중요한 요소이다. 아바타 모듈은 사용자의 의도를 가상 세계에 전달하는 사용자 인터페이스에 관련된 모듈로서 마우스, 조이스틱, 마그네틱 센서 등의 장치를 지원하며 입력신호의 의미를 해석한다. 시뮬레이터 모듈은 가상 세계에 적용되는 물리 법칙을 운영하는 모듈로서 실세계와 얼마나 비슷한 작용을 하는지가 결정되는 모듈이다. 렌더러 모듈은 변화된 가상 환경을 시각화하여 사용자에게 제공하는 모듈이다. 코디네이터 모듈은 가상 세계의 전체적인 운영 및 수행에 따른 관리를 담당하며, 다른 모듈간의 정보 교환 통신을 담당한다.

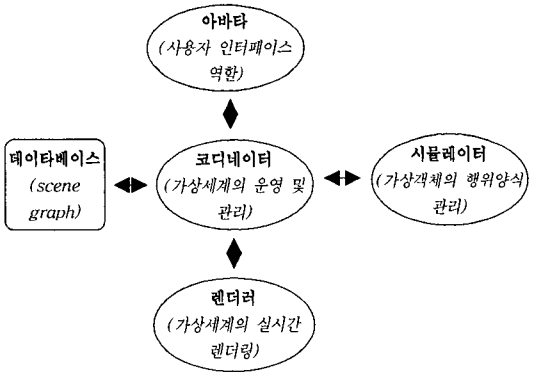

그림 2 KITTEN의 시스템 구조

3.3 기존 시스템에 대한 특성

본 논문의 앞 절에서 다중 스레드를 이용한 모듈 병렬화 제안과 시스템 구조에 대해서 살펴보았다. 그렇다면 이와 같은 구조가 기존의 다른 시스템에 비해서 가지는 특성 및 장점은 무엇인가? 기존의 가상현실 시스템은 2.1절에서 언급한 바와 같이 단일 시뮬레이션 루프 방식의 시스템과 다중 프로세싱 방식의 시스템으로 나눌 수 있다. 이에 본 절에서는 각 방식과 본 논문에서 제안하는

시스템과의 정성적 비교부터 시작한다.

먼저 단일 시뮬레이션 루프 방식의 시스템에 대한 장점은 명확하다. 가상세계 시뮬레이션 태스크의 부하가 큰 어플리케이션인 경우, 이 방식은 시뮬레이션 모듈에의 병목현상으로 그 다음 렌더링이 수행되지 못한다. 이에 비해 제안하는 시스템은 두 기능이 병렬적으로 수행됨으로써 시스템의 전체적 성능 저하를 방지할 수 있다. 이와 같은 맥락에서 *Virtual Windtunnel*이나 *Alice*와 같은 다중 프로세싱 시스템에 대한 장점도 이해할 수 있다. 즉, 상호작용과 시뮬레이션이 중요한 어플리케이션에서 기존의 시스템은 복잡한 시뮬레이션 수행 중 변경되는 사용자 입력을 처리하지 못한다. 그러나 제안하는 시스템에서는 상호작용 기능과 시뮬레이션 기능을 병렬적으로 수행시킴으로써, 새로 발생 및 변경되는 사용자 입력을 최소의 지연시간으로 가상 세계에 반영할 수 있다.

다중 프로세싱 방식의 다른 구조인 Performer™ 시스템이 채택하고 있는 파이프 라인 병렬화 방식을 생각해 보자. 파이프 라인 방식의 시스템의 경우, 시뮬레이션과 렌더링간의 파이프 구성으로 명확한 순서를 지키며 최대의 성능을 이끌어 내도록 설계되었다. 그림 3 (S: 시뮬레이션, R: 렌더링)은 정상적인 파이프라인의 수행흐름을 도시한 것이다. 따라서 각 프레임에 대해 파이프 라인이 이상적으로 운영될 경우 우수한 성능을 제공하며 모듈간의 엄격한 동기화가 필요한 시뮬레이션에 적합한 방식이다.

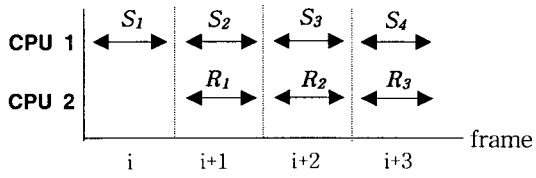

그림 3 정상적인 파이프라인 수행 과정

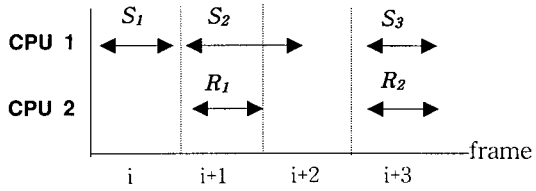

그림 4 시뮬레이션이 길어지는 경우의 파이프라인 수행과정

하지만 어느 한 프레임에서 시뮬레이션 수행 시간이 길어질 경우 - 예를 들어 *Perfomer*™의 경우 한 프레임 간격은 1/60초인데 - 문제가 발생한다. 엄격한 동기화를 유지하기 위해 그림 4와 같이, 다음 프레임의 시뮬레이션과 렌더링이 한 프레임 건너 뛰어 그 다음 프레임에서 시작되는 현상이 발생한다. 따라서 이와 같은 구조에서는 시스템의 엄격한 동기화는 유지되나, 프레임에 대한 CPU와 렌더링 시간의 낭비를 초래할 수 있다.

이에 비해 다중 스레드 병렬화로 설계한 본 시스템의 경우, 다중 CPU에서 각 모듈이 서로 독립적으로 수행된다. 본 시스템의 수행과정은 그림 5에서 보는 바와 같이 각 모듈이 비동기적으로 수행하면서 최대의 성능을 제공하게 된다. 이와 같은 경우 엄격한 동기화는 보장되지 않지만, 스레드 스케줄링에 따른 CPU 낭비를 막고, 시뮬레이션이 길어지더라도 렌더링 모듈에서 비교적 균일하고 빠른 렌더링을 수행하여 결과를 사용자에게 제공할 수 있는 특성을 지닌다. 따라서 제안된 시스템은 엄격한 동기화가 요구되지 않는 어플리케이션에서 이점을 지닌다.

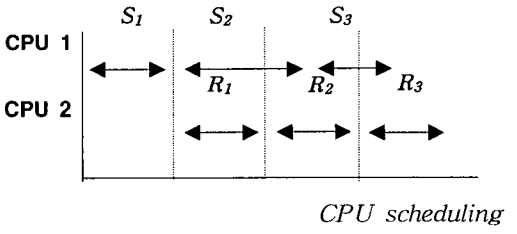

그림 5 모듈 병렬화 방식의 비동기 수행과정

본 논문에서는 위와 같이 엄격한 동기화가 요구되지 않는 경우뿐만 아니라, 시뮬레이션과 렌더링간의 엄격한 동기화가 요구되는 어플리케이션을 수행하기 위하여 다음과 같이 설계하였다. 병렬적으로 수행되는 각 모듈간의 정보 교환을 해결하기 위하여 기 정의된 메시지와 메시지 큐를 이용한 메시지 전달 방법을 사용한다. 모듈간에 다양한 메시지가 정의되는데, 시뮬레이션과 렌더링간의 동기를 처리하기 위하여 본 시스템에서는 두 모듈간에 동기 메시지를 정의한다. 즉, 시뮬레이터와 렌더러 모듈간에 엄격한 동기화가 필요한 어플리케이션의 경우에는 시뮬레이터 모듈은 현재 프레임에 대한 전체 태스크 수행이 끝난 후 그 결과를 동기 메시지와 함께 렌더러 모듈로 전달한다. 렌더러 모듈은 비동기적으로 계속 새롭게 장면을 그리는 것이 아니라, 시뮬레이터 모듈로부터 동기메시지를 받은 경우 - 한 프레임에 대한 시뮬레

이션이 끝난 경우 - 에만 새로운 렌더링을 수행하는 구조를 가진다. 결론적으로 본 시스템은 각 모듈간에 비동기 수행방식으로 최대의 성능을 제공하며, 엄격한 동기화가 요구되는 어플리케이션일 경우 동기 메시지 처리에 대한 선택사항을 활성화함으로써 각 모듈간의 동기화를 이루게 된다.

4. 시스템 구현

본 절에서는 제안된 다중 스레드 가상현실 시스템을 구성하는 핵심 모듈인 아바타 모듈, 시뮬레이터 모듈, 렌더러 모듈, 코디네이터 모듈의 설계와 구현에 대해서 기술한다. 각 세부 모듈의 설계에 대한 기술은 본 시스템의 운영과 수행에 대한 전반적인 내용을 포함한다.

4.1 아바타 모듈

아바타 모듈은 가상현실 시스템의 사용자 인터페이스 역할을 담당하는 모듈로서, 사용자 윈도우의 관리, 센서 및 아바타 제어와 관련된 부분을 처리한다. 센서 입력 및 아바타 제어 기능은 가상현실 시스템에서 사용자와의 상호 작용을 다룬다는 점에서 매우 중요한 기능이다. 본 논문에서 설계된 아바타 모듈은 기 제시된 가상현실 시스템의 개념적 모델을 바탕으로 재구성되었다[1]. 그림 6에서와 같이 아바타는 가상세계 내에서 사용자의 대리인으로, 사용자의 키보드, 마우스 등 p-sensor(physical sensor)의 입력에 반응하여 자신에게 정의된 v-effector(VE, virtual effector)를 통하여 가상세계와 상호작용한다.

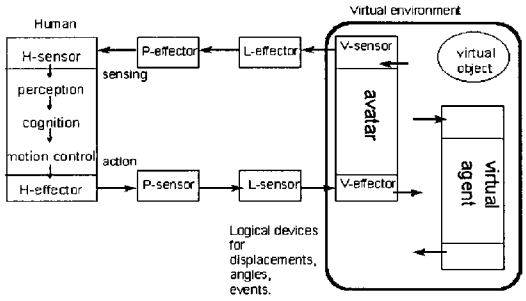

그림 6 기 제시된 VR의 개념적 모델[1]

효율적인 아바타의 관리를 위해 아바타는 계층구조를 가진다. 먼저 하나의 아바타(A)를 그림 7과 같이 가상 세계 내에서의 외형에 해당하는 가상객체(O) 노드와 아바타가 가지는 행위 양식의 대상이 되는 v-effector(VE) 노드의 조합으로 구성하였다.

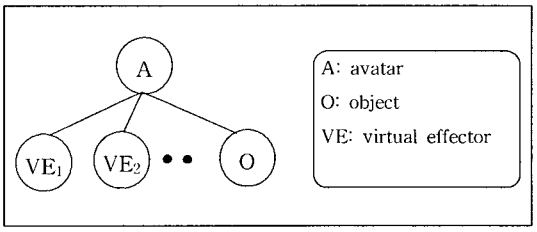

그림 7 아바타의 구성

첫째, 아바타의 외형은 가상 세계 내에서 참여자의 모습을 대신하는 것으로써 그림 8의 계층을 가지는 인체 모델과 같이 실제 인간과 같은 형태를 취할 수도 있지만, 가상세계 내에서 운항만을 목적으로 할 경우 자전거, 비행기, 자동차 등과 같은 외형을 나타낼 수도 있다.

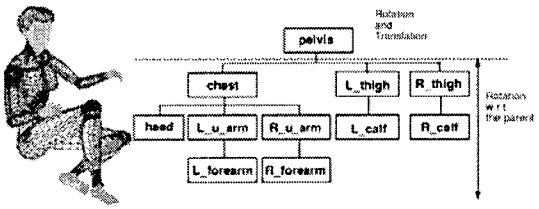

그림 8 인체 모델 계층 구조의 예[19]

둘째로, 가상 세계 내에서 아바타의 행위양식의 창구가 되는 VE는 가상 세계와의 상호작용을 위한 수단이 되며, 걷기 행위양식을 위한 다리, 잡기 행위양식을 위한 손 등 여러 VE를 하나의 아바타가 가질 수도 있을 것이다. 각각의 VE노드는 그림 9와 같이 3차원 가상 공간에서의 상호작용 방법을 정의한 IT(Interaction Task) 노드와 논리적 센서(LS) 노드로 구성하였다.

제시된 그림을 구성하는 각 노드를 간략히 설명하면 다음과 같다. IT노드는 가상 공간과의 상호작용 방법에 대한 기본적인 정의 혹은 분류를 나타내는 것으로서, 운항, 조작, 선택 등이 이에 속한다. 논리적 센서 노드는 행위양식을 위한 값을 표현하는 것으로 위치, 방향, 각, 속도, 각속도 등이 될 수 있다. 즉 아바타가 이동이라는 가상 세계 행위양식을 포함할 경우 아바타는 운항에 해당하는 IT 노드와 방향 및 속도 등의 값을 갖는 논리적 센서 노드로 구성된 다리(아바타의 외형이 자동차일 경우 바퀴)라는 VE를 가질 것이다.

가상 세계에 참여하는 사용자가 사용하는 도구는 키보드, 마우스, 데이타글러브와 같은 물리적 장치이다. 이러한 장치를 통해 입력되는 값의 종류 및 의미는 장치별로

다르며, 이를 아바타의 행위양식에 필요한 값의 형태인 논리적인 센서의 값으로 사용하기 위해서는 그림 6에서와 같이 일련의 처리과정을 필요로 한다. 이를 위해 논리적 센서 노드를 그림 9에서와 같이 실제 입력 장치에 해당하는 물리적 장치(PS; 마우스, 키보드 등)와 마스킹(M) 노드, 가상지각(VP) 노드로 구성하였다. 마스킹 노드는 물리적 장치를 통해 들어오는 입력 성분에 대해 필요로 하는 성분만을 필터링하거나 필요없는 부분을 마스킹하는 역할을 담당한다. 예로 기차라는 아바타가 있다고 가정하자. 운항이라는 상호작용의 경우 기차는 철로 위를 달리므로 전/후진만을 고려하면 된다. 만일 마우스와 같은 물리적 장치의 입력 성분 중 Y 성분을 논리적 센서 노드의 전/후진 방향의 속도값으로 사용할 경우 Y방향 성분을 제외한 나머지 성분은 제거시켜야 할 것이다. 가상지각 노드는 물리적 장치로 들어오는 값의 형태가 논리적 센서의 값의 형태로 직접 사용하기에 어려운 경우에 사용된다. 예로 데이타글러브와 같은 입력장치의 경우 현재 장갑의 위치 및 방향, 각 손각락의 굽힘정도 등이 입력값으로 사용된다. 이러한 값은 쥐기/놓기 혹은 가위/바위/보 등과 같은 손의 포스처를 나타내는 논리적인 값으로 바로 사용하기에는 곤란하다. 가상지각 노드에서는 이와 같이 물리적 입력 장치의 복잡한 입력값에 대해 이들 값을 분석하고 분류하여 해당되는 논리적인 값으로 변환하는 전처리 역할을 담당한다.

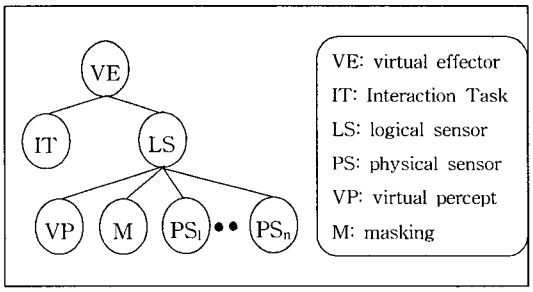

그림 9 VE의 구성

이와 같이 아바타 모듈은 사용자의 의도가 물리적 장치를 통하여 가상세계에 전달되는 일련의 과정을 담당한다. 이 과정은 기존의 그래픽스 어플리케이션과 같이 단순한 입력신호처리에 그치는 것이 아니라 경우에 따라서는 사용자의 의도를 분석하기 위해 상당히 복잡한 계산을 필요로 한다.

4.2 시뮬레이터 모듈

시뮬레이터 모듈은 가상 세계 내에서 발생하는 변화를 처리하는 모듈로서, 가상 객체의 행위 양식과 가상 세계 전체에 부여된 법칙을 결정하고 운영한다. 기본적으로 문의 열림과 닫힘, 선풍기 날개의 회전 등과 같은 객체 고유의 행위 양식을 처리한다. 또한 중력 작용, 충돌 검사와 같은 가상 세계의 전체적인 물리 법칙을 정의할 수 있으며, 이들은 시간 임계적 처리를 통해 효율적으로 운영된다. 이와 같은 가상 객체 행위 양식의 결정은 가상 세계를 구성하는데 있어서 매우 중요한 역할을 차지하며 가상현실 시스템의 성격을 결정지을 수도 있다. 이렇듯 객체의 행위 양식이 가상 세계의 매우 중요한 구성 요소임에도 불구하고 기존 시스템의 경우 기하학 객체가 가지는 속성의 일부로 간주하고 관리하였다[2]. 하지만 기하학 객체의 행위 양식 자체를 독립적인 객체로 볼 수 있으며, 따라서 본 시스템에서는 효율적인 행위 양식의 관리를 위해서 기하학 객체와 분리시켜 운영한다.

KITTEN에서 가상 객체의 행위 시뮬레이션 과정은 다음과 같다. 시뮬레이터 모듈은 코디네이터 모듈로부터 객체에 대한 행위 양식의 등록 지시를 받으면, 이를 시뮬레이터 모듈내의 태스크 리스트에 등록하고 관리한다. 이 후 시뮬레이터 모듈 내부의 메인 함수에서 프레임 갱신율을 유지하면서 각 객체에 대해서 등록된 태스크를 수행하게 된다. 그리고 태스크의 수행 결과는 시뮬레이터 모듈 내부의 메시지 큐에 쌓여진 후, 현재 프레임에 대해 모든 태스크의 수행이 끝난 다음 코디네이터 모듈로 전달되게 된다. 메시지를 받은 코디네이터 모듈은 다시 렌더러 모듈로 메시지를 전달함으로써 시뮬레이션된 결과를 렌더링하게 된다. 이렇게 함으로써 시뮬레이션의 결과가 렌더러에서 바로 반영되어 나타날 수 있도록 하였다. 만일 시뮬레이션 된 결과가 렌더러와의 동기화가 필요한 경우에는, 시뮬레이터 모듈에서 코디네이터 모듈로 전달하는 메시지에 동기신호를 첨가하여 전달할 수 있도록 하였다. 이렇게 함으로써 시뮬레이터 모듈이 렌더러 모듈과의 wall clock에 의한 동기화를 이룰 수 있으며 시스템의 일관성이 유지될 수 있도록 하였다. 메시지 전달과 코디네이터 모듈에 대한 자세한 설명은 4.4절을 참고한다.

4.3 렌더러 모듈

렌더러 모듈은 가상 세계의 장면(scene) 정보와 다른 모듈들로부터 전달되어 오는 갱신 정보를 바탕으로 현재 장면(scene) 상태에 대한 영상을 생성한다. 렌더러 모듈은 가상현실 시스템의 하부 모듈 중 시스템 하드웨어와 가장 큰 종속성을 갖는 모듈이다. 따라서 렌더링 모듈의 설계는 다른 시스템 하드웨어로의 호환과 확장에 매우 중요한 역할을 하게 된다. 이에 KITTEN에서는 렌더링을

담당하는 렌더러 모듈을 업계 표준 그래픽 라이브러리인 OpenGL을 기반 라이브러리로 선정하여 구현함으로써 전체 시스템의 플랫폼 독립성을 추구하였다.

가상현실 시스템에서의 렌더링 기능은 사용자로 하여금 사실감과 몰입감을 느끼게 할 수 있는 결과를 보여주어야 한다. 사실감을 제공하기 위해서는 많은 수의 다각형으로 세밀하게 표현된 객체를 렌더링하여야 하며, 충분한 몰입감을 느끼게 하기 위해서는 빠르고 일정한 프레임 갱신 속도로 그래픽 객체들을 그려주어야 한다. 이를 위해 본 시스템에서는 렌더링 속도를 향상시키고 렌더링 파이프라인으로 유입되는 다각형의 수를 줄이기 위하여 culling, LOD(level of detail) 그리고 모드 변화 최적화 기법을 사용하였다.

가상 세계의 현재 시점에서 보여지는 다각형만을 그리기 위한 culling과 복잡한 객체를 상세도에 따라 다단계 레벨로 표현하는 LOD 기능은 렌더링 파이프라인으로 유입되는 다각형의 수를 줄이는 대표적인 방법이다[15][16]. 렌더러 모듈에서 제공하는 culling 기법은 가시성을 이용한 view-frustum culling으로, 현재의 view-frustum 내부에 위치하는 다각형들만 그리게 된다. View-frustum내에 존재하는 다각형들을 효율적으로 검출하기 위해 객체 정보에 의한 사진트리(quad-tree)와 다각형 정보에 의한 사진트리를 구성하여 사용하였다. Kitten에서 사용하는 LOD 방법은 시점을 고려한 LOD 기법과 물체 표면의 수직벡터 성분을 보존하는 LOD 기법을 이용하였다[17].

또한 렌더러 모듈은 빠른 실시간 렌더링을 위해서 모드 변화 최적화 기법을 사용하였다. 모드 변화 최적화 기법이란 렌더링 파이프라인을 효율적으로 유지하기 위해 그래픽 파이프라인 내부에서 일어나는 모드의 변화를 최소화하는 방법으로, 하부 그래픽 라이브러리인 OpenGL의 매커니즘에 직접, 간접적으로 관련된다. OpenGL에서는 그래픽 객체를 그리는 순서를 고려함으로써 렌더링 성능을 향상시킬 수 있다. 예를 들어, 같은 형태의 다각형들을 모아서 렌더링하는 것이 그렇지 않은 경우에 비해 효과적이다. 삼각형은 삼각형끼리, 사각형은 사각형끼리 모아서 렌더링할 경우 더 나은 성능을 제공한다. 이러한 모드 변환에 따른 특성은 다각형의 종류 외에 텍스처, 표면 재질에도 적용된다. 이를 위해 KITTEN에서는 다양한 실험을 통해 최적의 렌더링 모드를 선택하였다. 실험을 통해 얻어진 모드 최적화 방법에 대해 살펴보면 텍스처를 최우선으로 하여 표면 재질, 다각형의 순으로 같은 종류끼리 정렬하여 렌더링 파이프라인에 전달하는 경우 최고의 성능을 얻을 수 있었다. 그림 10은 렌더러 모듈의 구성과 수행 흐름을 보여준다.

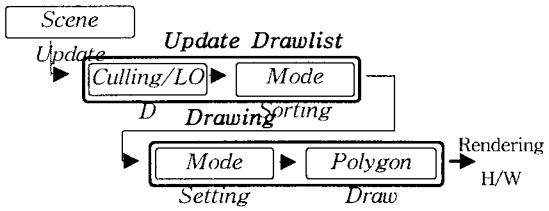

그림 10 랜더러 모듈의 수행과정

렌더러 모듈은 장면의 갱신이 요청될 때 새로운 렌더링을 수행하게 된다. 먼저 렌더링의 대상이 되는 객체에 대하여 속성을 갱신한다. 현재 시점에 대하여 변화된 위치 및 파라메터의 설정, 객체에 대한 변형 행렬의 재설정이 수행된다. 그 결과는 위에서 언급한 culling과 LOD과정, 모드 최적화를 위한 모드 순서화 과정을 거쳐 렌더링을 위한 새로운 drawlist(새로 그려져야 할 객체들의 리스트)를 구성한다. 새로운 drawlist를 구성하는 다각형들은 이 후 하드웨어 렌더링을 통하여 화면에 그려지게 된다.

4.4 코디네이터 모듈

코디네이터 모듈은 가상 세계의 전체적인 운영 및 수행에 따른 관리를 담당하는 메인 모듈로서 가장 중심적인 역할을 수행한다. 코디네이터 모듈은 기능적으로 다음과 같은 두 가지의 역할을 담당한다. 첫째, 병렬적으로 독립되어 수행되는 각 모듈간 정보교환의 중계자의 역할을 담당한다. 이를 위해 어플리케이션 수행을 시작하면서 아바타, 시뮬레이터, 렌더러 모듈을 위한 스레드를 생성하고 각 모듈들을 스레드에 할당한다. 그리고 비동기적으로 동작하는 스레드간의 통신을 위해 메시지 전달을 처리하도록 설계하였다. 둘째, 가상 세계 표현을 위한 scene graph를 구성하고 이를 관리하는 역할을 한다.

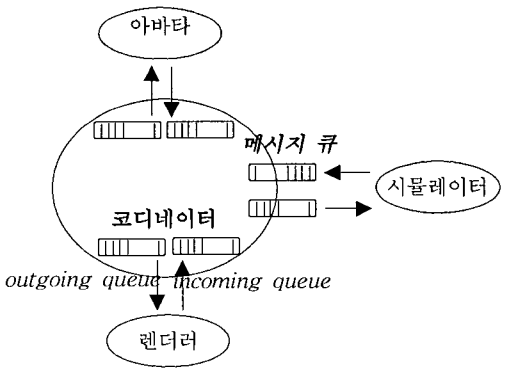

그림 11 메시지 큐를 이용한 모듈간의 통신

4.4.1 모듈간 정보 교환의 중계자 역할

KITTEN 시스템의 가장 큰 특징은 균일하고 향상된 렌더링 속도를 제공하기 위해서 각 모듈들을 비동기 병렬 수행시키는 것이다. 따라서 각 모듈간의 동기화 문제와 정보 공유 방식을 해결해야 하는데, 코디네이터 모듈이 이를 관장한다. 먼저 코디네이터 모듈은 스레드를 생성하여 각 모듈의 수행을 시작시킨다. 그리고 독립된 스레드로 수행되는 각 모듈들간의 일관성 있는 정보 유지를 위해 메시지와 모듈간에 연결된 큐에 의한 메시지 전달 방식을 채택하였다. 모듈간에 상호 필요한 정보(동기 정보 포함)는 모두 메시지의 형태로 변환되어 다른 모듈에 전달되게 되는데, 여기서 모듈간의 메시지 전달을 코디네이터 모듈이 담당한다. 그림 11은 메시지 큐를 이용한 모듈간의 통신을 도시한 것이다.

이와 같은 메시지 전달 방식의 장점은 모듈간의 동기화 문제를 줄이고 객체 지향적 시스템 설계에 있어 시스템 구성을 명료히 한다. 또한 이후 본 시스템을 네트워크를 통한 다중 참여자 가상현실 시스템으로 확장할 경우, 네트워크 모듈을 기존의 시스템에 추가하기 쉽다. 이것은 네트워크 모듈의 수행을 위한 메시지와 메시지 큐의 정의만 첨가되면 기존 시스템의 다른 부분에 거의 영향을 주지 않고 시스템을 확장할 수 있기 때문이다.

메시지 교환은 코디네이터 모듈과 다른 세 모듈간에 발생한다. 코디네이터 모듈은 다른
세 모듈에 대해서 각각의 입, 출력 큐를 생성하여 할당하고, 이 후 발생되는 모든 메시지에
대해 메시지 분배자로서의 역할을 담당한다. 아바타, 시뮬레이터, 렌더러 모듈은 독립된 스레드로 수행하면서 처리한 각종 정보를 기 정의된 메시지의 형태로 변환하여 코디네이터 모듈에 전달한다. 코디네이터 모듈은 반복된 메시지 처리 루프를 수행하면서 각 모듈에서 보낸 메시지를 목적 모듈에 전달하게 된다. 그리고 목적 모듈에서는 코디네이터로부터 받은 메시지를 처리하게 된다.

4.4.2 Scene graph 관리

코디네이터 모듈은 모듈간 정보 교환의 중계자 역할 이외에, 가상 세계 데이타베이스 관리자로서 가상 세계의 scene graph를 생성하고, 어플리케이션의 요청에 따라 이를 갱신한다. 또한 변경된 정보를 타 모듈들에게 전파함으로써 가상 세계 데이타베이스의 일관성을 유지한다. 그리고 scene graph를 구성하는 각 객체에 대한 식별자(ID)에 의한 접근 권한과 각종 클래스 객체에 대한 정보를 제공함으로써 가상 세계 정보의 질의 응답 부분을 수행하기도 한다.

KITTEN에서 scene graph의 설계는 가상 세계를 표현하는 데 있어서, 가상 세계를 구성하는 모든 요소들을 포함할 수 있는 강력한 표현력과 실제 어플리케이션 수행 시 성능을 극대화 할 수 있는 효율성이라는 어떻게 보면 상호 배타적인 요소를 함께 고려하여 설계하였다. 먼저 표현력을 높이기 위해서 scene graph의 구성 노드로 장면, 기하학 객체, 행위양식, 광원, 시점, 카메라, 센서 등의 가상 세계를 이루는 모든 구성 요소를 가질 수 있게 하였다. 그리고 이와 같이 표현된 구성 요소간의 관계, 예를 들어 계층적으로 구성된 기하학 객체간의 부모-자식관계나 형제 관계의 표현을 명확히 하였다. 또한 객체와 연결된 행위 양식, 센서 등과의 연결 관계도 충분히 표현할 수 있도록 설계하였다.

그림 12는 가상 세계에 표현된 로봇 팔을 scene graph 측면에서 살펴본 한 예이다. 좌측의 그림과 같은 다관절 로봇이 있을 경우, 물체 자체는 body(O1)와 arm(O2), finger(O3, O4)의 계층적인 구조로 이루어져 있으며, 광원(L1), 카메라(C1, C2) 등이 함께 어울려 전체적인 장면을 구성하게 된다. 위 그림에서 전체 scene graph의 루트 노드로 현재의 장면을 가지고, 자식 노드로 그 장면을 구성하는 기하학 객체와 광원, 시점을 가진다. 시점이 두 개 이상인 경우처럼 여러 개의 구성 요소가 한 장면에 존재할 경우에는 그래프의 같은 레벨에서 형제 노드로 추가된다. 그리고 로봇 팔과 같은 계층적 객체는 그래프 상에서 부모, 자식, 형제 노드로 표현할 수 있는 노드 특성으로 표현된다. 또한 렌더링 시 효율적 갱신을 위해 객체의 재질 요소와 변형 행렬을 객체 내부에 포함하도록 설계하였다.

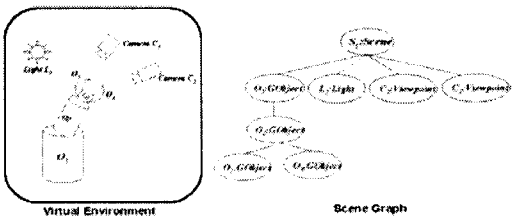

그림 12 로봇 팔의 가상 세계 표현

5. 성능평가 실험

본 절에서는 다중 스레드를 이용하여 설계한 본 시스템을 기존의 단일 시뮬레이션 루프 방식의 시스템과 성능 평가 실험을 수행한 내용을 기술한다.

5.1 측정 기준의 선택

다중 스레드를 이용하여 모듈의 병렬 수행 방식으로

설계된 본 시스템의 성능 평가 실험에서는 단일 시뮬레이션 루프 방식의 시스템과의 비교 실험을 통해 그 성능 향상을 알아보았다. 단일 시뮬레이션 루프 방식의 시스템으로는 SARAH 시스템을 이용하였다. SARAH는 한국과학기술원 인공지능 연구실에서 개발한 가상 세계 저작 시스템이다. SARAH는 Silicon Graphics 워크스테이션의 GL 라이브러리를 기반으로 제작된 시스템으로, 렌더링 모듈의 기본 구조는 KITTEN의 렌더러 모듈과 동일하다.

실험은 일반적인 가상현실 어플리케이션의 대표적인 두 가지 도메인인 장면 복잡도와 시뮬레이션 복잡도가 높은 가상 환경에서 두 시스템의 렌더링 성능을 비교하였다. 렌더링 성능의 척도로는 어플리케이션 수행 시 변화하는 화면 갱신율을 사용하였다. 즉, 전체적인 화면 갱신율의 속도 향상 여부와 수행하는 어플리케이션의 복잡도에 비교적 영향을 받지 않으면서 얼마나 균일한 갱신율을 제공하는가의 여부가 실험의 평가 기준이다. 가상 세계의 장면 복잡도 실험으로는 Virtual Campus 네비게이션 실험을, 시뮬레이션 복잡도를 위한 어플리케이션으로는 구의 충돌 검사 실험을 수행하였다.

5.2 Virtual Campus Navigation

실험에 사용된 Virtual Campus는 한국과학기술원(KAIST)을 대상으로 제작한 대규모 가상 세계로서, 일반적인 장면 복잡도에 대한 렌더링 성능 실험의 test-bed로 사용하였다[18]. 가상 캠퍼스를 내비게이션함으로써 사용자는 캠퍼스 내를 실제로 투어(tour)하는 것과 유사한 효과를 얻을 수 있을 뿐 아니라, 캠퍼스에 대한 제반 정보를 잘 정리된 형태로 제공받을 수 있다. 그림 13은 Virtual Campus 네비게이션의 스크린 샷이다.

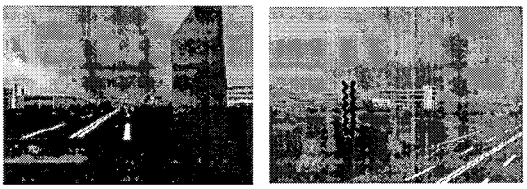

그림 13 Virtual Campus 네비게이션시 수행시의 snapshot

본 실험에서는 장면 복잡도에 따른 렌더링 성능을 알아보기 위해, Virtual Campus 네비게이션 시, 장면 내의 다각형 수가 달라짐에 따른 화면 갱신율의 변화를 측정하였다. 그림 14는 실험에 사용된 네비게이션 경로에 대한 다각형 수의 변화를 나타낸 것이다. 실험에 사용된 다각형의 수는 시점에 대한 view-frustum 내의 다각형 수를 나타낸 것이다. 그리고 네비게이션 수행 시 SARAH와 KITTEN의 프레임 진행과 그에 따른 렌더링 장면은 동일하다.

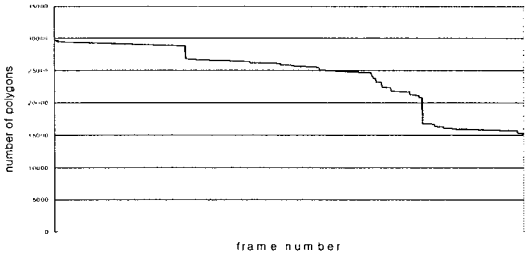

그림 14 네비게이션 수행에 따른 다각형 수의 변화

그림 15는 SARAH와 KITTEN 두 시스템에 대해서 주어진 경로에 대해, Virtual Campus 네비게이션 수행의 실험 결과를 나타낸 것이다. 차트의 가로 축은 진행되는 프레임 순서를, 세로 축은 그에 따른 화면 갱신율을 보여주고 있으며, 차트에 나타난 굵은 선과 가는 선은 각각 KITTEN과 SARAH의 화면갱신율을 나타낸 것이다. 실험 결과 그림에서 보는 바와 같이, 단일 시뮬레이션 루프 방식과 다중 스레드로 설계한 본 시스템이 비슷한 렌더링 성능을 보였다.

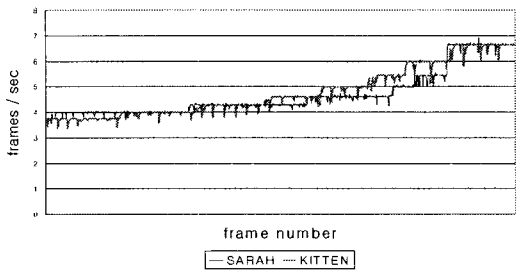

그림 15 네비게이션 수행에 따른 화면 갱신율의 변화

이와 같은 결과는 단일 시뮬레이션 루프 방식과 다중 스레드 방식의 시스템이 시뮬레이션 태스크가 없이 가상 세계 walk-through나 fly-through를 주로 하는 어플리케이션일 경우, 두 시스템의 렌더링 성능이 크게 차이가 나지 않음을 보여주는 것이다.

5.3 구의 충돌검사

Virtual Campus를 이용한 장면 복잡도에 따른 렌더링 성능 실험 결과, 기존의 단일 시뮬레이션 루프 방식의 시스템과 본 논문에서 제안한 시스템은 비슷한 성능을 보

였다. 이것은 앞 절에서 언급한 바와 같이 시뮬레이션 태스크가 없는 어플리케이션임에 따라 발생한 결과이다. 따라서 본 절에서는 복잡한 시뮬레이션 태스크를 수행하는 어플리케이션일 경우, 어떠한 렌더링 성능을 보이는지를 살펴본다.

시뮬레이션 복잡도를 위한 실험 환경으로는 자유 운동하는 구(sphere)들간의 충돌 검사를 수행하는 어플리케이션을 선정하였다. 충돌 검사는 이동과 회전에 따라 자유 운동하는 객체들간에 충돌 여부를 검사하는 것이다. 이것은 N-body problem으로서 일반적으로 정확한 충돌 여부를 알아내는 것은 힘들다. 본 실험에서는 일반적인 물체보다 간단한 모양을 갖는 구의 성질을 이용해서 충돌 검사를 수행하였다. 본 실험에 사용된 구의 수는 125개, 200개, 500개, 1000개이며, 하나의 구를 구성하는 다각형의 수는 80개이다. 실험은 총 1,500 프레임에 대해서 구의 수가 늘어남에 따른 두 시스템에서의 화면 갱신율 변화를 측정하였다. 첫 500프레임은 구가 움직이지 않는 정적인 상태로, 그 다음 500프레임은 구의 자유 운동만을 시뮬레이션하였다. 그리고 마지막 500프레임은 자유 운동하는 구들간의 충돌검사를 수행하는 과정으로 전체 실험을 구성하였다.

SARAH와 KITTEN 두 시스템에 대해 각각 구의 수를 늘려가면서 충돌검사 실험을 수행한 결과를 그림 16에 나타내었다. 그림에서 점선은 SARAH의 화면 갱신율 변화를, 실선은 KITTEN의 변화를 표시한다. 각 차트에 나타난 화면 갱신율은 경과된 100개의 프레임에 대한 평균을 계산하여 결과를 보여준 것이다.

실험 결과 단일 시뮬레이션 루프 방식의 시스템인 SARAH의 경우 충돌 검사 시뮬레이션 수행이 이루어지

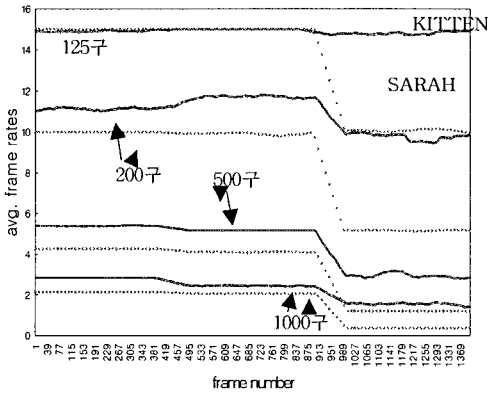

그림 16 구의 충돌검사 수행시 SARAH(점선)와 KITTEN(실선)의 화면 갱신율의 변화

면, 시뮬레이션 전과 후의 화면 갱신율이 크게 차이가 나는 것을 알 수 있었다. 이것은 앞에서 언급한 바와 같이 단일 시뮬레이션 루프 방식의 단점으로 지적된, 시뮬레이션 모듈에 많은 부하가 걸림에 따른 병목현상으로 렌더링 성능의 저하를 가져온 결과이다. 이에 비해 다중 스레드로 설계한 KITTEN의 경우 시뮬레이션 모듈과 렌더링 모듈이 분리되어 병렬적으로 동작한다. 이에 따라 시뮬레이션 태스크 수행에 따른 화면 갱신율의 변화 폭이 비교적 적은 안정적 성능을 보였다. 특히 구의 수가 125개인 경우(약 10,000개의 다각형), 평균 화면 갱신율이 15 frame/sec로 매우 균일한 성능을 보였다. 이와 같은 결과로 볼 때, 시뮬레이션 복잡도가 높은 가상 세계 어플리케이션인 경우 다중 스레드로 설계한 본 시스템이 비교적 안정적 성능을 제공한다고 할 수 있다.

6. 결 론

본 논문에서는 가상 세계 어플리케이션의 렌더링 성능 향상을 위해서 각 모듈을 병렬 수행시키는 가상현실 시스템인 KITTEN을 설계하고 구현하였다. 가상 세계의 복잡도나 시뮬레이션의 복잡도에 비교적 영향을 받지 않으면서 빠른 렌더링 속도를 제공하기 위해서, 가상 세계 운영을 위한 핵심 기능인 상호작용, 시뮬레이션, 렌더링 기능을 다중 스레드로 구성하고 병렬 수행시키도록 설계하였다. 이에 따라 복잡한 시뮬레이션 태스크의 수행 중에도 병렬적으로 렌더링이 가능하게 되어, 기존의 단일 시뮬레이션 루프 방식의 시스템의 문제점이 보완되었다. 또한 사용자의 센서 입력을 담당하는 상호작용 기능을 타 모듈과 분리시켜 병렬 수행하게 함으로써 시뮬레이션이나 렌더링 수행 중에도 갱신된 센서의 입력을 가상 세계에 반영하여 사용자 입력의 지연 시간을 최소화하였다. 본 시스템의 성능 평가 실험의 결과 시뮬레이션의 복잡도가 높은 어플리케이션에서 기대한 바와 같이 비교적 안정적인 성능을 제공하였다. 따라서 앞으로 제작하고자 하는 가상 환경이 시뮬레이션을 많이 필요로 하는 경우, 또는 시뮬레이션과 렌더링간의 엄격한 동기화가 요구되지 않는 어플리케이션일 경우 본 시스템은 우수한 성능을 발휘할 것으로 생각된다.

현재 개발된 KITTEN에서 지원하는 모델 양식은 한국과학기술원에서 정의한 SEG2.0 (단일 세그먼트의 정의), KINETIX사의 3D Studio 포맷인 3DS 파일(자체 개발한 변환기를 통해 SEG로 변환 후 지원), 그리고 VRML 등이다. VRML(Virtual Reality Modeling Language)은 3차원 가상 세계 데이타의 인터넷 상에서의 교환을 목적으로 제정된 표준으로서 하나의 가상 세계 전체에 대한 표준

이다. KITTEN에서는 VRML내의 외형의 표현과 광원, 시점, 그리고 주요한 성질인 PROTO node에 의한 새로운 노드의 정의 및 활용과 DEF/USE에 의한 데이타 공유를 지원한다. 현재로는 스크립트에 의한 동적인 행동 양식이나 이벤트의 생성과 라우팅에 의한 상호 작용의 정의는 지원되지 않는다. VRML과 같은 표준화에 KITTEN은 유용성을 나타낼 것으로 보인다. 인터넷으로 연결된 상태에서의 복잡한 3차원 시뮬레이션과 그 결과의 렌더링이 필요한 어플리케이션에서는 지연없이 비교적 균일한 렌더링 결과를 전송해 주어야 한다. KITTEN의 경우 추후 네트워크 모듈을 담당하는 새로운 스레드를 추가하여 시뮬레이션과 렌더링, 그리고 네트워크 전송이 병렬적으로 이루어질 수 있다. 따라서 네트워크 전송과 복잡한 시뮬레이션을 요구가 증대될 경우 본 시스템의 구조가 도움이 될 것으로 생각된다.

비록 본 논문에서 제안한 형태의 구조가 여러 목적을 갖는 범용 가상현실 시스템의 다양한 요구를 만족시키기에는 부족함이 없으나, 군사적 응용과 같은 특수 목적을 위한 적용에는 몇 가지 문제점을 지닌다. 예를 들어, 빠른 시뮬레이션을 요구하는 경우 사용자의 입력으로 인한 시점의 변화가 실시간에 나타나야 하지만, 다중 스레드를 갖는 시스템의 특성상 모듈간 메시지 전달에 따른 지연이 발생한다. 또한 모듈간 정보 교환과 데이타 공유 방식으로 채택한 메시지 전달 방법은 각 스레드로 분리된 모듈이 궁극적으로 분산환경에서 서로 다른 프로세서에서 수행할 수 있는 것을 목표로 설계되었다. 따라서 단일(stand-alone) 시스템 플랫폼에서 메시지 전달 방식은 오버헤드를 지닌다고 볼 수도 있다. 이와 같은 문제점을 보완하기 위해서 다음과 같은 개선점을 생각해 볼 수 있다. 모듈간 정보 전달의 방식으로 공유 메모리 방식을 도입한다. 즉 가상세계 데이타베이스의 구조에 대한 변화만을 메시지를 사용하여 각 모듈에 변화 사실을 통보하고, 그렇지 않은 속성의 변화는 공유 메모리를 통하여 전달되도록 하는 것이다. 이렇게 함으로써 가상세계 데이타베이스에서 좀 더 많은 부분이 공유될 수 있도록 하여 메모리의 이용 효율을 높이고 좀 더 빠른 처리를 가능케 한다. 그 외에도 사용자와 상호작용의 복잡성보다는 실시간 동작성에 대한 요구가 큰 어플리케이션의 경우, 시점제어, 물건제어와 같은 가상세계에 직접적인 조작을 행하는 사용자의 입력 부분을 아바타 모듈로부터 분리하여 렌더러 모듈과 통합하여 처리함으로써 즉시 시각적 반응을 나타내게 하는 것도 고려해 볼 수 있다.

그리고 본 논문에서 제안하고 구현한 시스템을 보다 범용적인 시스템으로 발전시키기 위해서는, 군사, 교육, 건설 등의 다양한 어플리케이션을 쉽게 제작할 수 있는 어플리케이션 템플릿(template)의 개발이 필요하다. 다양한 어플리케이션에 활용할 수 있도록 하기 위해서는 본 시스템을 하부 구조로 삼고, 그 상위에 각각의 어플리케이션에 알맞은 템플릿을 개발하는 것이 요구된다. 또한 가상현실 시스템의 하부구조와 어플리케이션 템플릿을 연결해 주는 메커니즘을 제공함으로써 사용자로 하여금 원하는 가상 환경을 쉽고 빠르게 개발할 수 있도록 해야 할 것이다.

Acknowledgement

본 연구의 일부는 과학기술정책 연구소 및 서울대 자동제어특화연구센터의 지원에 의하여 수행되었음. ETRI의 네트워크 가상 현실 게임을 위한 기반 기술 및 소프트웨어 개발의 연구비 지원하에 연구되었음.

참 고 문 헌

[1] 원광연, "전산학으로서의 가상현실", 정보과학회지, 15권, 11호, pp. 5-13, 1997년 11월.

[2] J. Wernecke, "The Inventor mentor: Programming Object-Oriented 3D Graphics with Open Inventor-TM, Release 2," Addison Wesley, 1994.

[3] 성운재, 원광연. "가상세계 저작도구의 설계 및 구현", 한국시뮬레이션학회 논문지, 4(1):37-44, 1995.

[4] Sense 8 corporation, "World Tool Kit: Virtual World Development Software," 1992.

[5] VREAM Inc. "VREAM; Virtual Reality Development System Users Guide 1.0", 1993.

[6] Superscape Co., "SuperscapeVRT," http://www.superscape.com/, 1997.

[7] S. Bryson, "The Virtual Windtunnel: An Environment for the Exploration of Three Dimensional Unsteady Flows," *Proceedings of Visualization 91*, pp. 17-24, 1991.

[8] C. Show, J. Liang, M. Green, "The Decoupled Simulation Model for Virtual Reality Systems," *Proceedings of CHI 92*, pp. 321-328, 1992.

[9] J. Rohlf and J. Helman, "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *Proceedings of SIGGRAPH 94*, pp. 381-394, July 1994.

[10] H. Igehy and G. Stoll and P. Hanrahan, "The Design of a Parallel Graphics Interface, *Proceedings of SIGGRAPH 98,*" pp. 141-150, June 1998

[11] R. Pausch, T. Burnett, M. Conway, R. DeLine and R. Gossweiler, "Alice: A Rapid Prototyping System for Virtual Reality," *SIGGRAPH 94 Course Notes #2*, Chapter 15, July 1994.

[12] Sense 8 corporation, "World Up Release 2," May 1996
[13] S. Bryson, "An Extensible Interactive Visualization Framework for the Virtual Windtunnel," *Proceedings of VRAIS 97*, pp. 106-113, March 1997.
[14] S. Bryson, "Approaches to the Successful Design and Implementation of VR Applications," *SIGGRAPH 94 Course Notes #2*, chapter 9, July 1994.
[15] J. Foley, A. van Dam, S. Feiner and J. Hughes, "Computer Graphics: Principles and Practice," pp. 301-310, Addison-Wesley, 1990.
[16] P. Astheimer, P. Poche, "Level-of-detail Generation and Its Application in Virtual Reality," *Proceedings of VRST 94*, 1994.
[17] HyungSeok Kim, SoonKi Jung, and Kwangyun Wohn. "A Multiresolution Control Method Using View Directional Feature," *Proceedings of VRST 98*, pp. 163-170, Taiwan, November 1998.
[18] 서혜원, 좌수진, 원광연. "버츄얼 캠퍼스: 대규모 가상세계의 제작", 한국 그래픽스 학회 논문지, 3(2):15-25, 1997.
[19] 정순기. "Motion Analysis of Articulated Objects For Optical Motion Capture," Ph.D. thesis, 한국과학기술원, 1997.

김 대 원
1997년 경북대학교 컴퓨터공학과 졸업(학사). 1999년 한국과학기술원 전산학과 졸업(석사). 1999년 ~ 현재 한국과학기술원 전자전산학과 박사과정. 관심분야는 인공지능, 가상현실, 퍼지이론 및 응용.

이 선 우
1993년 경북대학교 컴퓨터공학과 졸업(학사). 1995년 한국과학시술원 전산학과 졸업(석사). 1995년 ~ 현재 한국과학기술원 전자전산학과 박사과정. 관심분야는 가상현실, 컴퓨터그래픽스.

원 광 연
1974년 서울대학교 응용물리학과 학사. 1974년 1979년 국방과학연구소 연구원. 1981년 the Univ. of Wisconsin 전산학과 석사. 1984년 Univ. of Maryland 전산학과 박사. 1984년 ~ 1986년 Harvard Univ. 응용과학부 강사. 1986년 ~ 1990년 Univ. of Pennsylvania 전산정보 과학과 조교수. 1990년 ~ 현재 한국과학기술원 전자전산학과 부교수. 관심분야는 virtual reality, HCI, culture computing

이 광 형
1978년 서울공대 산업공학 학사. 1980년 한국과학원 산업공학 석사. 1982년 프랑스 INSA 전산학과 석사(DEA). 1985년 프랑스 INSA 전산학과 공학박사. 1988년 1월 프랑스 국가박사(전산학 INSA-LYONI대). 1985년 ~ 1995년 한국과학기술원 전산학과 조교수 및 부교수. 1995년 ~ 현재 한국과학기술원 전자전산학과 교수. 1985년 프랑스 INSA. 1995년 미국 Stranford Research Institute. 관심분야는 퍼지 이론 및 응용, 인공지능, 전문가 시스템 등