미로 환경에서의 네트워크 가상현실 응용의 구현

한 확*, 고 욱**, 하 순 회*
서울대학교 컴퓨터공학과*, 이화여자대학교 전자공학과**

요 약

네트워크 가상현실 시스템은 먼 거리에 떨어진 사용자들 사이에 일관성 있는 가상 세계를 제공하며, 군사, 오락, 건축 등 여러 부분에 응용되고 있다. 본 논문에서는 그 동안 고성능 그래픽 워크스테이션 환경에서 중심적으로 연구되어 왔던 네트워크 가상현실 시스템을 가장 보편적인 플랫폼인 PC 환경에서 구현할 때 생기는 3차원 그래픽 처리 성능의 문제, 네트워크 대역폭과 전송 지연의 문제 등 여러 문제점들과 이에 대한 해결책들을 제시하고 건축 시뮬레이션을 응용으로 삼아 복수 사용자 가상현실을 구현하였다.

1. 소 개

컴퓨터 그래픽 시스템의 궁극적인 형태로서 가상현실 시스템은 사용자가 가상 세계의 개체들 뿐 아니라 그 시스템에 연결된 다른 사용자와도 투명하게 상호작용할 수 있는 시스템을 지향한다. 지역적 제한성을 극복한 복수 사용자의 지원은 네트워크를 기반으로 한다. 네트워크 가상현실 시스템(분산 가상현실 시스템)은 잠재적인 응용 분야가 넓기 때문에 활발한 연구 대상이 되고 있으며, 특히 고성능 그래픽 워크스테이션에 기반하여 군사 시뮬레이션 등의 응용에서 유용성을 보여 주었고, 오락, 교육, 의학, 건축 등의 분야로 그 응용 범위를 넓혀 가고 있다.

PC의 성능이 계산 집약적인 3차원 렌더링 파이프라인을 처리할 수 있을 만큼 향상됨에 따라, Doom에서 비롯된 몰입형 가상현실 게임이 등장하였고, PC용 3차원 가속기와 마이크로소프트를 비롯한 주류 회사의 3차원 API 지원은 3차원 그래픽 응용이 PC에서 활발히 개발, 이용될 것임을 암시하고 있다. 이와 더불어 인터넷에 대한 이용의 증가와 컴퓨터간의 연결성의 증가는 PC 환경에서의 복수 사용자 가상현실 응용의 가능성을 더욱 밝혀 주고 있다.

PC 환경에서의 복수 사용자 가상현실 시스템 프레임워크가 가져야 할 속성들은 다음과 같다.

- PC 환경에서 완전한 3차원 렌더링 파이프라인을 이용한 응용
- 하드웨어 플랫폼과 화면의 복잡도에 관계없이 안정된 프레임 생성 속도의 확보
- 네트워크의 대역폭, 전송 지연 등의 제한을 극복하기 위한 효율적인 메커니즘

이 논문에서는 가장 널리 분포되어 있는 PC 환경에서 복수 사용자 가상현실 시스템을 구현하기 위한 기본 개념을 제시하고, 미로 환경 내비게이션 프로그램을 그 특정한 응용으로 선택하여 구현하였다.

2. 이전의 연구들

2.1 PC용 가상현실 게임

id Software 사의 Doom[1]에서 비롯된 PC 환경의 미로 슈팅 게임들은 많은 게임 플레이어들의 주목을 받았다. Doom은 BSP 트리에 기초한 은면 제거와 레이 캐스팅 기법을 이용하여 벽, 바닥, 천장 등의 3차원 평면으로 구성된 제한된 미로 세계를 화면에 보여주며, 잘 설계된 팔레트와 텍스쳐, 스프라이트들은 이러한 제한된 3차원 세계를 효과적으로 잘 표현하여 플레이어에게 1인칭 관점에서의 몰입성을 제공한다.

3차원 그래픽 처리의 측면에서, 최근의 Quake로 이어지는 3차원 게임들은 높아지는 PC의 처리 능력을 잘 반영하고 있지만 복수 사용자를 위한 네트워크의 측면은

여전히 소규모에 머물러 있다. 그 예로 Doom은 모뎀과 IPX/SPX에 기반한 복수 사용자 게임 플레이를 지원하지만, 프레임이 생성될 때마다 각 플레이어의 위치 정보를 브로드캐스트함으로써 네트워크 대역폭에 대한 요구량이 많아지므로 10명 내외의 플레이어만을 지원할 수 있는 약점을 가지고 있다[2].

2.2 고성능 그래픽 워크스테이션에 기반한 네트워크 가상현실 시스템

80년대 중반부터, 미국 국방성은 군사 재정 감축에 대한 대비책으로서 많은 비용을 수반하는 실제 훈련의 대안으로, 컴퓨터에 기반한 군사 훈련과 war game simulation에 노력을 기울여 왔다. 고성능 그래픽 시스템 환경의 대규모 분산 시뮬레이션에서 어떻게 효율적인 정보 교환이 이루어질 수 있는가에 대한 내용이 연구의 주를 이루었으며, 이는 SIMNET (Simu-lation Network)와 DIS (Distributed Interactive Simulation) 프로토콜로 구체화되었다[3][4]. DIS 프로토콜은 추측 항법(dead reckoning) 알고리즘을 이용하여 브로드캐스트 패킷 갱신 방식을 이용했을 때 생기는 대량의 플랫폼간 데이터 교환 패킷의 수를 줄임으로써 네트워크를 보다 효율적으로 이용하여, 각 개체의 상태 정보 계산을 여러 플랫폼에 분산하여 처리함으로써 프로세싱의 부하 균형을 이루어 내었다[2].

추측 항법 알고리즘의 이용에도 불구하고, 1000명 이상의 플레이어가 시뮬레이션에 참여하는 경우에 다시 나타나는 대역폭과 지연의 문제를 해결하기 위해 Naval Post Graduate School에서 개발 중인 NPSNET system[5]은 가상 세계의 분할, AOIM(Area Of Interest Manager)에 의한 객체 필터링과 멀티캐스트 네트워크를 결합하여 각 호스트 사이에 교환해야 하는 상태 정보의 갱신 패킷 수를 효율적으로 줄여 내었다

그러나 이 시뮬레이션 시스템들은 고성능 워크스테이션에 기반해 있기 때문에 그에 따른 전용 API를 사용하여 플랫폼의 확장성에 제한을 가지며, DIS 프로토콜은 군사 응용을 중심으로 설계되었으므로 이를 개선하기 위한 연구 작업이 계속되고 있다[2].

2.3 MUD, MOO와 Graphical MUD

텍스트에 기반한 가상 세계를 제공하는 머드(MUD, Multi-User Dungeon/Dimension)[6] [7]는 진화를 계속하여 고전적인 LPmud와 같은 킬러 게임에서 TinyMUD, LambdaMOO와 같은 사회적, 교육적인 영역으로 가상 공간을 넓혀 가고 있다. 머드에서 제공하는 가상 세계는 많은 수의 방과, 이 방에 대한 묘사, 그리고 그 방 사이의 출입구로 정의된 데이터베이스의 형태로 제공되며, 이 데이터베이스는 하나의 서버에서 유지, 관리를 수행하고 각 사용자는 서버 접속 능력을 가진 클라이언트들을 통해 데이터베이스 내에 정의된 각 방을 출입하고, 다른 개체들과 상호작용할 수 있다. 그래픽 머드는 텍스트에 기반한 머드에 그래픽 인터페이스를 추가한 것으로 그래픽 속성을 가지는 객체들이 통신 대역폭과 전송 시간을 낭비하는 것을 막기 위해, 이 객체들을 포함한 보다 큰 용량의 클라이언트를 요구한다.

머드는 유연성, 확장성, 가상 세계에 대한 기술 능력 등에 있어서 Doom 등의 3차원 게임에 비해 이점을 가지지만, 클라이언트 서버에 기반한 통신 구조는 사용자 수가 많아짐에 따라 서버에 과중한 부하를 주어, 50-100명의 사용자만을 지원할 수 있게 되는 약점이 있다[8].

2.4 VRML

VRML(Virtual Reality Modeling Language)은 네트워크를 통해 여러 사용자들이 공통의 가상세계에 표준적인 접근을 할 수 있도록 하기 위해 제안되었다[15]. 기존의 OpenInventor 포맷을 확장한 VRML 포맷은 3차원 객체를 위한 기본적인 형식들을 제공하며, 사용자는 그 가상세계가 정의된 데이타베이스를 미리 다운로드 받음으로써 내비게이션을 비롯한 여러 상호작용을 할 수 있다. 그러나 이의 기반인 인터넷은 상대적으로 느린 속도를 가지며, WWW에 기초한 오버헤드가 큰 프로토콜 스택을 통해 통신하는 점과 실시간 그래픽 처리를 위한 객체 수준의 지원이 없는 점으로 인해, VRML은 네트워크 수준의 문제가 해결될 미래의 플랫폼에서 보다 광범위하게 이용될 것이다.

2.5 PC에 기반한 네트워크 가상현실 시스템에서의 이슈들

프로세서의 계산 능력이 향상되고 3차원 가속기의 개발이 활발해 지고 있지만 고성능 그래픽 워크스테이션에 비해 아직 충분한 처리 능력을 PC는 갖지 못하고 있으므로, 이미지의 질과 이미지 생성의 속도 사이의 trade-off를 감안할 때, 여기에서는 이미지 생성의 속도(frame rate)를 증가시키기 위한 노력이 많이 필요하다.

인간에게 편안함을 제공할 수 있는 프레임 속도는 30 fps(frame per second) 정도이며, 상호작용이 많은 실

시간 응용에서 프레임 속도의 하한은 10 fps 정도로 볼 수 있다. 그러나 플랫폼의 처리 능력, 화면의 복잡도, 각 물체의 detail 정도에 따라 각 프레임의 생성, 디스플레이 속도는 많은 편차를 가지게 되며, 급격한 움직임을 가지는 일련의 프레임들을 생성하여 사용자에게 혼란을 일으킬 수 있다[9].

(문제 1) 그래픽 처리 부하의 동적 균형으로 프레임 생성 속도의 편차를 줄여야 한다.

프레임 속도를 일정한 범위 내에 한정되게 하는 문제는 비행 시뮬레이터 시스템에서 활발하게 연구되었는데, 여기에서는 물체의 여러 Level of Detail(LOD)을 적절히 선택함으로써 렌더링할 다각형의 수를 일정한 범위 내로 한정시켰다. Funkhouser[17]는 한 프레임을 렌더링하는데 필요한 시간은 다각형 총수와 사용된 렌더링 알고리즘에 의해 좌우됨을 보였다. 각 물체에 대해 렌더링 비용(cost)과 기여도(benefit)에 대한 두 함수를 계산함으로써 그는 적당한 LOD와 렌더링 알고리즘을 선택하여 한정된 범위 내의 프레임 속도를 얻어낼 수 있었다. IRIS Performer[18]의 경우 역시 LOD를 선택하기 위한 피드백 메커니즘을 제공하지만, 어떤 시뮬레이션 과정에서 과도한 피드백 impul- se로 인해 진동 효과가 나타날 수 있는 단점이 있다.

복수 사용자 응용의 경우 각 사용자가 이용하는 플랫폼의 그래픽 처리 성능 차이가 존재하게 된다. 이 때, 모든 플랫폼에 처리 능력에 관계없이 같은 양의 계산 부하를 할당한다면, 느린 플랫폼의 사용자는 낮은 프레임 속도로 인해 충분한 정보를 제공받지 못하게 된다. 예를 들어, 사용자 A가 사용자 B보다 더 고성능의 플랫폼을 사용하고 있다고 가정할 때, A가 방 안의 한 물체를 집어들었고, 이에 해당되는 갱신 메시지를 B에게 보냈다고 가정하자. 이 때 B의 플랫폼은 낮은 처리 성능으로 인해 이전 프레임의 렌더링을 다 끝마치지 못한 상태인데 갱신 메시지를 받는다면 이 메시지를 먼저 처리해야 할 것이다. 이로 인해 B는 A에 비해 지역 사용자 입력을 받아 처리할 기회가 줄게 되며, 이는 A와 B 사이의 처리성능 차이로 나타나게 된다. A가 계속하여 이러한 갱신 메시지를 보내면, 이 성능 차이는 점점 축적되어 두 사용자 사이의 비동기적 상황을 낳게 된다. 여기에서 두 번째 문제를 도출할 수 있다.

문제 2) 플랫폼에 독립적으로 일정한 프레임 생성 속도를 유지해야 한다.

일정한 프레임 속도가 보장된다 하더라도 복수 사용자 시스템인 경우에는 각 사용자에 대한 정보, 각 사용자가 가상 세계에 가한 수정 정보 등이 네트워크 대역폭, 지연, 신뢰성 문제 등으로 동기화되지 않을 수 있다. 이러한 문제는 가상 세계에 연결된 사용자의 수가 많아짐에 따라 더욱 심각해질 것이다[10].

(문제 3) 네트워크 대역폭, 지연, 신뢰성에 의해 야기되는 가상 환경의 동기화와 scalability의 문제를 해결해야 한다.

3. 시스템의 개요

본 시스템의 개략적인 구성도는 그림 1과 같이 4 개의 부분으로 구성된다[16].

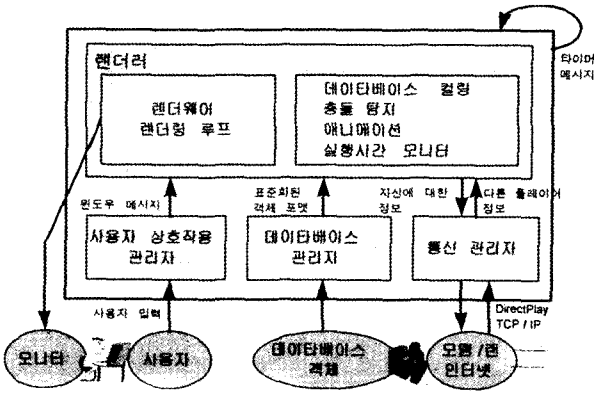


그림 1. 시스템 다이어그램

사용자 상호작용 관리자는 사용자로부터 입력을 받아 새로운 viewing parameter들을 계산하여 이를 렌더링 루프에 전달한다. 렌더링 루프 부분은 3차원 그래픽 처리를 위한 핵심적인 부분으로서, 이 프로그램에서는 RenderWare API를 이용하여 구현하였다. 가상 환경에 대한 정보를 담고 있는 데이터베이스와 시나리오 파일은 데이터베이스 관리자의 처리를 거쳐 정격화된 포맷으로 렌더러 부분에 주입된다. 원격에 위치한 사용자와 데이터베이스의 갱신 정보를 처리하기 위한 통신 관리자는 시스템의 네트워크 인터페이스와의 패킷 전송, 수신

을 담당한다. 렌더러 내에는 동적인 데이터베이스 컬링을 통한 실시간 화면 갱신과 충돌 처리, 객체의 애니메이션을 위한 여러 루틴들, 그리고 실행 시간 모니터 루틴이 각 렌더링 국면마다 수행된다.

4. 3차원 그래픽 처리

4.1 객체 데이타베이스의 구조

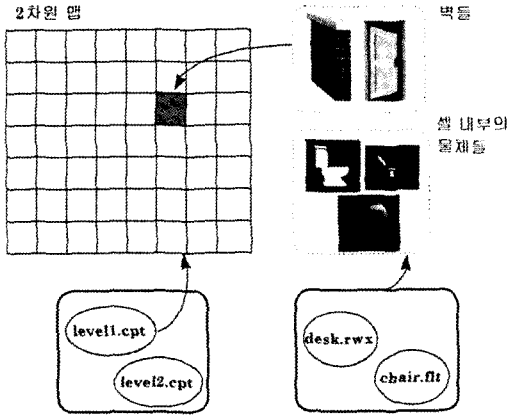


그림 2. 객체 데이타베이스의 구조

가상 세계는 여러 개의 셀로 나뉘어진 가상 세계의 평면도로서의 맵 파일과 맵에서의 각 셀에 위치한 객체들의 기하학적 정의를 가지는 객체 파일들로 나뉘어진다. (그림 2)

맵 파일은 각 셀이 가지는 속성들 (벽의 유무, 바닥의 레벨 등)과 각 셀이 가지는 객체들의 파일 이름들의 배열로서 정의된다. 각 객체 파일들은 RenderWare 고유의 포맷 외에 OpenFlight 포맷, 3D Studio 포맷 등의 다양한 형태의 포맷으로 제공되며, 이들은 렌더러에서 내부적으로 변환 과정을 거친 후 렌더링 루프에 보내어진다. 렌더러는 이러한 맵 파일을 읽어 들여 가상 세계를 로드하며, 실행 시간에 데이터베이스 컬링과 LOD 처리를 거쳐 보다 적은 수의 프리미티브들만을 렌더링 루프에 보내게 된다.

4.2 안정된 프레임 속도를 위한 그래픽 기능들의 구현

4.2.1 데이터베이스 컬링

현재 사용자가 어느 위치에 있고 어느 방향을 향하고 있느냐에 따라서, 그림 3과 같이 프리미티브들의 가능한 가시성 집합 (PVS, Potentially Visible Set)이 달라지게 된다[11]. 렌더러는 프레임을 생성할 때마다 가능한 가시성 집합을 재계산하는데, 이 계산은 셀 단위로 이루어진다. 현재의 시점에서 바라본 시야 내에 들어가는 모든 셀은 가능한 가시성 집합에 포함되며, 그림 3에서 회색으로 칠해진 부분이 그에 해당된다. 이 집합 내의 모든 셀과 그에 속한 모든 객체는 렌더링 루프로 보내진다.

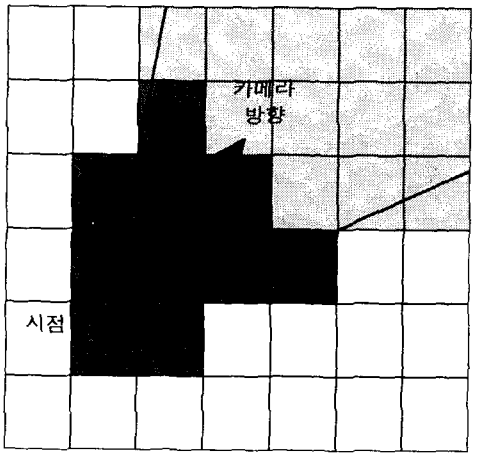


그림 3. 데이타베이스 컬링과 LOD

4.2.2 LOD (Level Of Detail)

가능한 가시성 집합에 속하는 각 셀에 대해 사용자와 그 셀의 중심 사이의 거리에 근거하여 LOD 레벨을 선택함으로써 먼 거리에 있는 객체들의 처리 시간을 줄일 수 있다. 그림 3에서 회색의 그레이 레벨은 LOD 레벨에 상응한 명도를 가지고 채색되었다. 대상이 되는 객체의 LOD 정의는 수행 시간에 생성되지 않고, 그 객체의 모델링 단계에서 지정된다.

4.2.3 수행시간 모니터

실시간 그래픽 응용의 가장 중요한 척도인 프레임 속도에 영향을 미치는 변수들은 화면의 복잡도(다각형의 개수, 다각형의 상대적 크기, 조명 모델, 텍스쳐 매핑 여부 등), 메모리 대역폭(윈도우의 크기, 한 픽셀 당 칼라 비트 수 등이 이에 영향을 미친다), 프로세서의 부동소숫점연산 처리 능력 등으로 다양하며 실행 시간에 시시각각으로 변할 수 있다[12].

2절에서 지적한 (문제 1)과 (문제 2)의 해결을 위해

프레임 생성 속도를 감시하고, 이에 따라 렌더링에 영향을 미치는 인자들을 변경할 필요가 있으며 이는 그림 4와 같은 형태의 실행 시간 모니터에 의해 이루어진다.

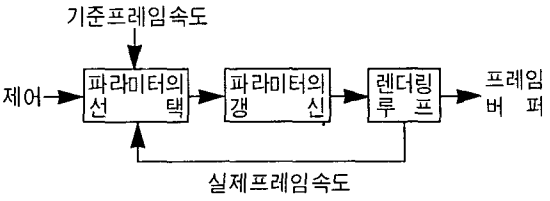


그림 4. 일정한 프레임 속도를 위한 실행시간 모니터

여기에서는 렌더링 레벨의 변화에 따라 렌더링에 영향을 미치는 인자들의 값을 변경하는 방식을 이용하였다. 렌더링 레벨은 다음과 같이 화면의 질에 영향을 미치는 정도가 작은 순으로 선택되도록 순서를 정하였다.

- 레벨 0 : 모든 렌더링 인자들은 가장 좋은 질의 화면을 생성하도록 설정됨
- 레벨 1 : 텍스쳐 매핑 여부
- 레벨 2, 3 : FOV (Field of View)와 DOV (Depth of View)의 조정
- 레벨 4: 윈도우 크기의 조정

5. 네트워크 통신 처리

네트워크 가상현실 시스템을 위한 분산 아키텍처는 클라이언트 서버 시스템과 동료 대 동료 시스템으로 나누어질 수 있다[8]. 클라이언트 서버 시스템의 경우에 가상 세계에 대한 데이터베이스는 서버에 저장되어 있으며 모든 데이터베이스에 관한 처리는 서버에서 이루어진다. 따라서 사용자들은 자신의 정보가 갱신될 때마다 서버에 정보 갱신 패킷을 보내야 하며, 많은 사용자가 빈번한 패킷을 보내면 서버에서 처리해야할 패킷 수가 너무 많아져 scalability에 문제를 보이게 된다. 반면, 동료 대 동료 시스템은 데이터베이스 자체가 클라이언트에 분산되어 있고 패킷 처리는 많은 시스템에 분산되기 때문에 보다 균형적인 부하 균형이 이루어지나, 패킷의 전송, 수신을 위한 절차가 복잡해지는 단점이 있다. 여기에서는 동료 대 동료 시스템을 가정하여 그림 5와 같은 통신 아키텍처를 상정하였다.

각 호스트 사이에서 교환되는 메시지들은 가상 세계 내의 객체들에 대한 상태 정보 값을 가지고 있으며[4], 그림 6과 같은 포맷과 교환 프로토콜을 가진다.

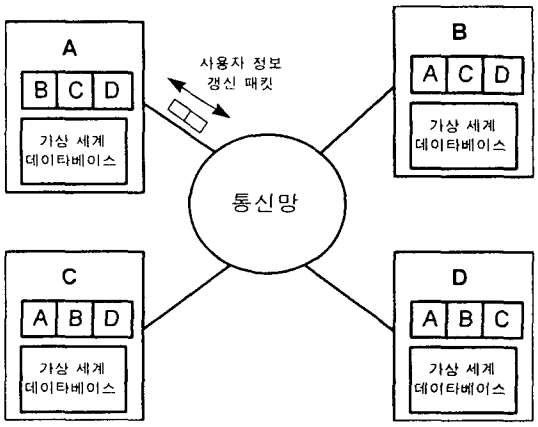
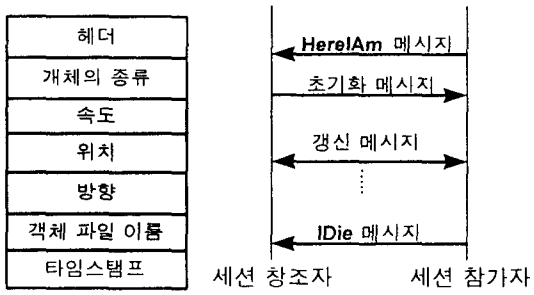


그림 5. 통신 아키텍처



사용자 위치 갱신 패킷 메시지 교환 프로토콜

그림 6. 메시지 포맷과 메시지 교환 프로토콜

2절에서 지적한 (문제 3)은 각 사용자에 대한 정보가 변할 때마다 패킷을 보내는 것이 사용자 수가 증가함에 따라 많은 수의 대역폭을 소모하게 됨을 의미한다. 이를 해결하기 위해 추측 항법 알고리즘을 이용하여 패킷의 수를 줄인다. 추측 항법에는 여러 종류의 기법이 사용될 수 있는데 여기에서는 사용자의 가장 최근의 위치 이동점을 3개까지 저장해 두었다가 위치 변화의 각도에 따라, 2차 또는 3차의 곡선 맞추기 알고리즘을 적용하는 기법을 채택하였다[13].

6. 실 험

그림 7은 미로 환경의 가상 세계를 만들기 위한 가상 세계 편집기의 모습을 보여 주고 있다. 각 셀의 변 주위

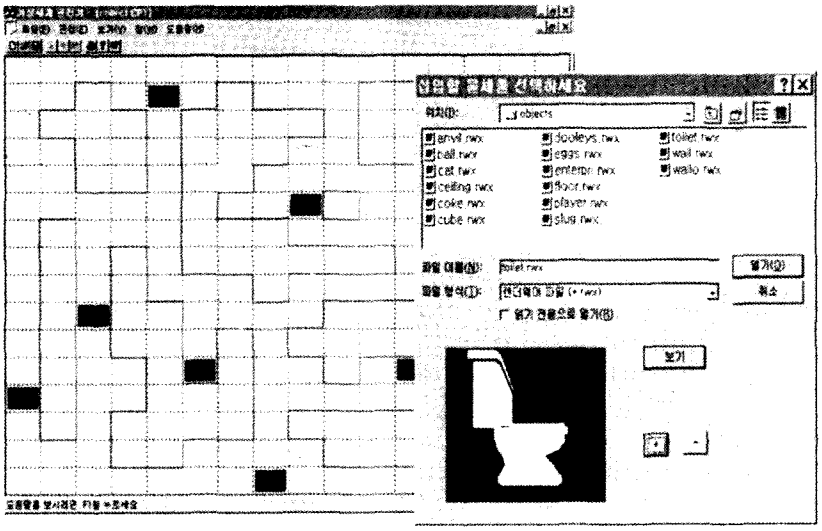


그림 7. 가상 세계 편집기

의 굵은 선은 그 셀에서 바라본 벽에 해당되며, 검게 칠해진 부분은 그 셀에 다른 내부 물체를 놓았음을 뜻한다.

그림 8은 위와 같은 과정을 통해 만들어진 가상 세계에 대한 내비게이션의 한 장면을 각각 보여 주고 있다. 편집기에서 지정한 벽들과 내부 물체들, 그리고 이 가상 세계에 진입한 다른 사용자들의 모습들을 볼 수 있다. 3차원 그래픽 처리와 통신에 대한 성능 정보는 윈도우 아래쪽의 상태 라인에 나타나 있다.

그림 9는 전형적인 미로 환경을 사용자가 내비게이트 할 때 얻어진 프레임 속도의 변화를 보여주고 있다

여기에서는 150MHz Pentium PC에 있는 한 명의 사용자가 약 1800 개의 벽이 있는 미로를 내비게이트한다고 가정하였다. 수행시간 모니터를 사용하지 않은 경우 오른쪽 그래프와 같이 프레임 속도는 급변하는 것에 반해, 이를 사용한 경우 왼쪽 그래프와 같이 기준 프레임 속도인 10 fps 주변에 집중됨을 볼 수 있다. 초기를 비롯하여 내비게이션 과정에서 프레임 속도가 갑자기 떨어지는 순간 또한 볼 수 있으나, 이 순간은 짧게 분산되어 있음을 볼 수 있다. 왼쪽 그림 아래에 표시된 막대 그래프는 수행 시간 모니터가 각 시점에서 요구되는 처리의 부하량에 따라 조정하는 렌더링 레벨의 변화를 나타내며 오른쪽 그림 아래의 점선 그래프는 .사용자의 시야에 들어오는 다각형의 갯수 변화를 나타낸다.

그림 10은 추측 항법 알고리즘에 의해 필요한 갱신 메시지 수가 감소함을 보여주고 있다. 윗부분의 그래프

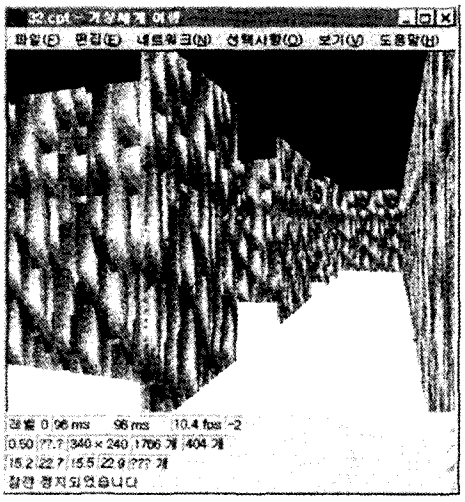


그림 8. 가상 세계 내비게이터

는 추측 항법을 사용한 경우 시간이 지남에 보내는 갱신 메시지를 나타내고 있으며, 아래쪽 그래프는 사용자의 움직임이 있을 때마다 갱신 메시지를 보내는 경우를 나타내고 있다. 한 프레임에서 메시지를 보낼 확률은 추측 항법을 사용하지 않았을 경우 0.324248인데 비해, 사용하는 경우에는 0.051675로 현격히 감소하였다.

상이한 여러 플랫폼 상에서 프레임 속도의 변화를 알아보기 위해 3대의 PC에서 이 응용을 수행하였다. 그

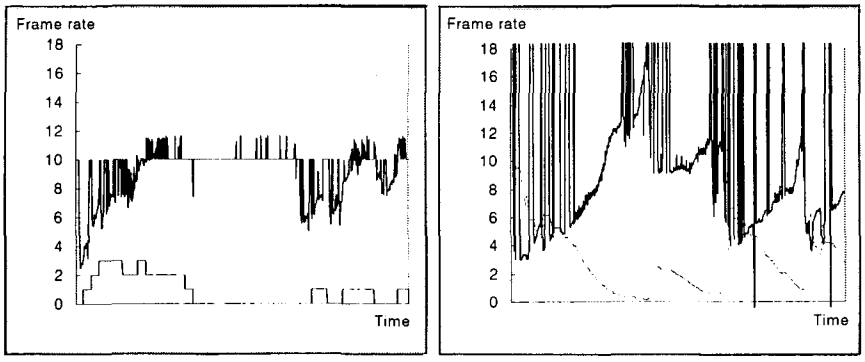


그림 9. 수행시간 모니터의 사용 여부에 따른 프레임 속도의 변화

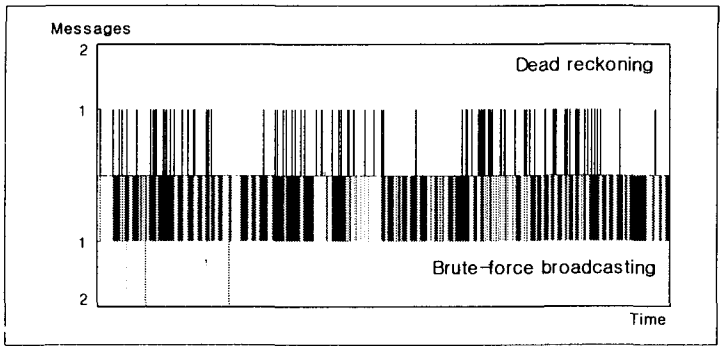


그림 10. 추측 항법 알고리즘에 의한 메시지 수의 감소

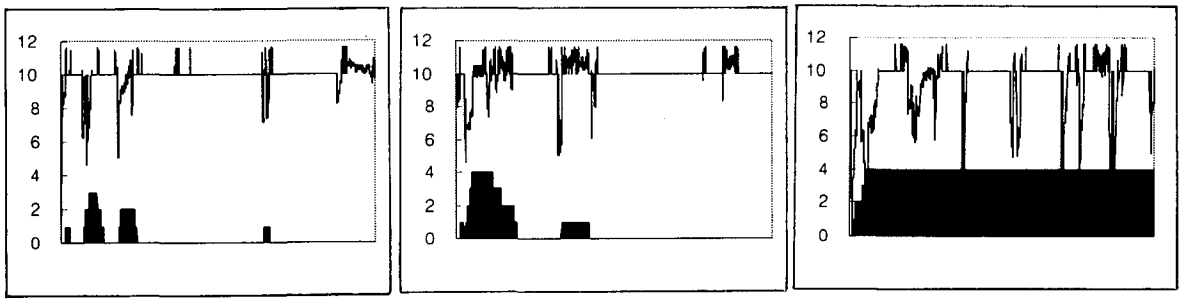


그림 10. 세 플랫폼에서의 프레임 속도의 변화
(왼쪽 위부터 각각 Pentium Pro 180MHz, Pentium 150MHz, 486DX2 66MHz)

표 1. 프레임 속도 변화에 대한 통계

	Pentium Pro	Pentium	80486
평 균	9.9685fps	9.9139fps	9.1921fps
표준편차	0.8275	0.7467	1.9045
낮은 프레임 속도의 비율	9.03%	10.63%	28.29%

결과 얻어진 프레임 속도의 변화는 그림 11에 나타나 있으며, 결과에 대한 통계는 표 1에 나타나 있다. 플랫폼의 성능 차이에 따라 차이가 있지만, 평균 프레임 속도는 거의 같게 유지되었으며, 기준 프레임 속도보다 낮은 시간의 비율은 전체 시뮬레이션 시간 중 작은 부분만을 차지하게 됨을 볼 수 있다.

7. 결론 및 향후 연구 과제

본 연구에서는 PC 환경에서 복수 사용자 가상현실 시스템을 구현하기 위한 개념들을 제시하고 그의 한 응용으로 미로 환경 내비게이션에 기반하여 효율적인 실시간 3차원 그래픽 처리와 통신을 위한 시스템을 보였다.

지금까지 구현된 프레임워크는 초보적인 형태에 머물러 있으며, 기능적인 면의 확장을 통해 가상현실 응용을 위한 프레임워크의 제작으로 발전시키고자 한다. 이와 결부된 향후 연구 과제는 대규모 시뮬레이션에서의 멀티캐스트 통신에 대한 시뮬레이션, 보다 풍부한 가상 세계를 위해 각 객체들에 행동의 양식의 지정과, 이 객체의 행동이 통신을 통해 분산 처리되는 과정을 분석하는 것이다. 이를 위해 메시지 포맷의 보다 정교한 설계와 효과적인 애니메이션 기법들이 필요하다[14].

참 고 문 헌

[1] id Software, Doom Frequently Asked Questions
[2] B. Roehl, "Distributed Virtual Reality An Overview," http://sunee. uwaterloo. ca/~broehl/distrib. html
[3] J. Locke, "An Introduction to the Internet Networking Environment and SIMNET/DIS," unpublished work, http:// www-npsnet. cs. nps. navy. mil/npsnet/ publication. html
[4] D. R. Pratt, "A Software Architecture for the Construction and Management of Real-Time Virtual Worlds," Ph. D. Dissertation, Naval Post Graduate School, 1993.
[5] M. R. Macedonia, "A Network Software Architecture for Large Scale Virtual Environments," Ph. D. Dissertation, Naval Post Graduate School, 1995.
[6] rec. games. mud, MUD Frequently Asked Questions
[7] P. Curtis and D. A. Nichols, "Muds Grow Up: Social Virtual Reality in the Real World," Xerox PARC, Jan. 1993.
[8] M. R. Macedonia and M. J. Zyda, "A Taxonomy for Networked Virtual Environments," http://www-npsnet. cs. nps. navy. mil/npsnet/publication.html
[9] J. Helman, "Architecture and Performance of Entertainment Systems," *In SIGGRAPH '95 Course Note "Designing Real-Time 3D Graphics for Entertainment","* 1995.
[10] D. P. Brutzman, M. R. Macedonia, and M. J. Zyda, "Internetwork Infrastructure Requirements for Virtual Environments," In Proceedings of the Virtual Reality Modeling Language Symposium, Dec. 1995.
[11] S. J. Teller and C. H. S quin, "Visibility Preprocessing for Interactive Walkthroughs," *In Proceedings of SIGGRAPH '91*, 1991.
[12] M. Jones, "Lessons Learned from Visual Simulation", In SIGGRAPH '95 Course Note *"Designing Real-Time 3D Graphics for Entertainment,"* 1995.
[13] S. K. Singhal and D. R. Cheriton, "Using a Position History-Based Protocol for Distributed Object Visualization," In SIGGRAPH '94 Course Note *"Designing Real-Time 3D Graphics for Entertainment,"* 1994.
[14] D. R. Pratt, P. T. Barham, et al., "Insertion of an Articulated Human into a Networked Virtual Environment," *In Proceedings of the 1994 AI, Simulation and Planning in High Autonomy Systems Conference*, Dec. 1994.
[15] A. L. Ames, et. al., "The VRML Sourcebook," John Wiley & Sons, Inc., 1995.
[16] 고욱, "다자참여 다목적 3차원 시뮬레이션 엔진", *한국컴퓨터 그래픽스 학회지*, vol. 1, no. 2, 1995.
[17] T. A. Funkhouser and C. H. S quin, *"Adaptive Display Algorithm for Interactive Frame Rates during Visualization of Complex Virtual Environments," In Proceedings of SIGGRAPH '93*, 1993.
[18] J. Rohlf and J. Helman. "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *In Proceedings of SIGGRAPH '94*, 1994.

필 자 소 개

한 확

- 1996년 서울대학교 컴퓨터공학과 학사
- 1996 ~ 현재 서울대학교 컴퓨터공학과 석사과정 재학중
- 주관심분야 : 네트워크 가상현실, 컴퓨터 그래픽스, 비디오-이미지 혼성 렌더링

하 순 회

- 1985년 서울대학교 전자공학과 학사
- 1987년 서울대학교 전자공학과 석사
- 1992년 University of California, Berkeley 전자공학과 박사
- 1992 ~ 1993년 University of California, Berkeley, Post Doc.
- 1993 ~ 1994년 현대전자산업주식회사 선임연구원
- 1994 ~ 1996년 서울대학교 컴퓨터공학과 전임강사
- 1996 ~ 현재 서울대학교 컴퓨터공학과 조교수
- 주관심분야 : 병렬처리, 하드웨어-소프트웨어 통합설계, 디지털 시스템 설계 방법론

고 욱

- 1980년 서울대학교 전자공학과 학사
- 1984년 University of California, Berkeley 전자공학과 석사
- 1986년 University of California, Berkeley 전자공학과 박사
- 1990년 삼성전자 반도체 수석원구원
- 1995년 삼성종합기술원 컴퓨터그래픽스 가상현실 팀장
- 1995년 ~ 현재 이화여자대학교 전자공학과 조교수
- 주관심분야 : 컴퓨터 그래픽스, 가상현실, 컴퓨터 게임