한국컴퓨터그래픽스학회 Vol. 4, No. 1, P. 47-54

분산 가상공간내 분신간 상호작용을 위한 네트웍 구조 및 프로토콜 설계

The Network Architecture and Protocol for Avatar Interaction in Distributed Virtual Environment

성만규, 박찬종, 김동현

ManKyu SUNG, ChanJong PARK, DongHyun KIM

시스템공학연구소 감성공학연구부 가상현실연구실

요 약

다중 참여자를 지원하는 가상현실 시스템은 기존의 가상현실 시스템에 비해 현실감을 증대시키는 효과가 있으나 네트웍 구조 및 프로토콜에 대한 새로운 문제를 야기하고 있다. 이는 가상현실 시스템의 실시간이라는 제약조건과 네트웍상에 보내어 지는 방대한 정보의 양에 비해 기존의 TCP/IP를 사용하는 인터넷의 성능이 이에 못미치기 때문이다. 본 논문에서는 분신을 통한 다중참여자의 가상공간 탐색 및 서로간의 상호작용을 위한 네트웍 구조 및 프로토콜의 설계 및 구현에 관한 것이다. 대상 가상공간은 여러개의 방으로 이루어진 가상 빌딩을 대상으로 하였으며 전체 가상공간은 다수의 서버에 의해 관리되도록 하였다. 사용자를 나타내는 분신은 9개의 동작을 나타낼수 있으며, 서로간에 제안된 프로토콜에 의하여 메시지 교환을 통해 서로의 상태를 인지 하도록 하였다

Abstract

The multi-user VR system can more enhance the sense of reality than existing single user VR system, but it also cause a significant problem about network structure and protocol, since the current Internet using TCP/IP can not afford to deliver such a massive information required to the real-time constraint of multi-user VR system. In this paper, we introduce new network structure and protocol for multiple user's virtual world navigation and interaction through their avatars. The virtual worlds are based on virtual building and managed by distributed servers. The avatars have 9 behaviors and recognize the state of each other by pre-defined network protocol.

1. 서론

가상현실(Virtual Reality)시스템은 사용자로 하여금 실세계에서 느끼는 현실감을 그대로 느끼게 해주는 시스템은 일컫는다. 하지만 인간이 느끼는 현실감은 단순히 환경적 요인, 즉 사실적으로 표현된 건물이나 사물에서 비롯될 뿐 아니라 실제적으로는 사회에서 느끼는 다른 사람들과의 상호작용을 통해 보다 직접적인 현실감을 느끼게 된다[1]. 이런 의미에서 진정한 가상현실시스템은 일종의 Social Ware라고도 표현 할 수 있다.

다른사람과의 상호작용을 위해서는 사용자

들의 통신 미디어인 네트웍 지원이 필수적이며, 더불어 네트웍상에 사용자를 표현할 수 있는 방법이 필요하게 되는데 흔히 이를 '분신'(Avatar)라고 표현한다[2]. 만약 네트웍상에 사용자를 대신하는 분신이 완전히 사용자의 의지와 감정을 표현 하며 가상 오브젝트와의 사실적 상호작용을 할 때 사용자는 화면상에 나타나는 분신과의 일체감을 느끼며 이를 통하여 분신이 들어있는 가상공간에 대한 간접적인 현실감을 느끼는 것이다. 그러므로 사용자의 행동 및 감정을 표현하는 분신은 일반적으로 인간형 모델(Humanoid Model)을 사용하게된다.

가상현실시스템의 특성상 리얼타임이라는 제한조건은 네트웍 관리에서 많은 요구조건이 필요하게 된다. 즉 기존의 인터넷등에서 발생하는 대역폭이나 지연 문제를 해결할 수 있는 방법을 제공하여야 하며 또한 시스템이 나타내야 할 가상공간의 조건, 즉 야외나 넓은 운동장, 혹은 유원지 같은 넓은 공간이냐 아니면 가상건물 내에서와 같은 좁은 영역이냐에 따라 전체 시스템의 네트웍 구조가 달라져야 한다.

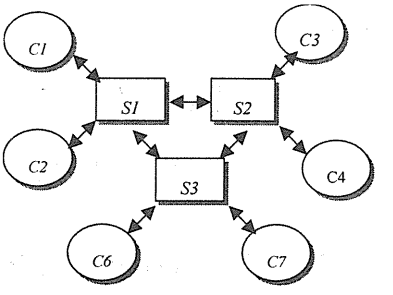
본 연구는 상대적으로 좁은 영역을 차지하는 가상건물 내에서 분신끼리의 다양한 상호작용을 위한 네트웍 모델 및 프로토콜의 설계에 관한 연구이다.

2. 분산 네트웍 모델

네트웍상의 트래픽을 줄이는 기본적인 방법은 전체가상공간을 여러개의 Zone 혹은 Cell로 나누어서 여러개의 서버가 각각의 Zone을 관리하는 것이다[3]. 이 모델은 클라이언트가 전체 가상공간의 일부분에 대한 메시지만 송/수신하므로 전체 네트웍 트래픽을 줄일 수 있지만 어떤 기준으로 가상공간을 분리할 것인지에 대한 해결방법이 제공되어야 한다.

기존의 가상공간을 Cell로 나누는 접근방법에 대한 연구는 관심영역관리기(AOIM:*Area of Interest Manager*)를 시스템에 두어서 전체 가상공간을 6각형의 Cell로 나누어 각 Cell마다 개별적인 멀티케스트 주소를 이용하는 방법과[4], 현 사용자의 뷰포인트를 기준으로 사용자가 보이는 계층적 Cell 구조를 보내는 방법이 제안되었다[3].

[그림 1]은 분산 네트웍 모델의 예를 나타낸 것이다.



S1~S3 : 서버
C1~C7 : 클라이언트

[그림 1] 분산 네트웍 모델

그림 1에서와 전체 가상공간을 3개의 서버, 즉 *S1~S3*로 분산시켜 관리하며 각 서버들 끼리는 직접적으로 데이터를 주고 받을 수 있다.

3. 시스템 설계

3.1 가상공간

본 시스템은 상대적으로 적은 가상공간으로 이루어진 가상건물내 환경을 목표로 하였다. 기존의 전쟁 시뮬레이션용으로 만들어진 가상공간은 그 목적상 상당히 넓은 가상공간이 요구되었으며 이로 인하여 정규적인 형태의 Zone으로 전체 가상공간을 분리 할 수밖에 없었다. 하지만 가상건물 내에서는 여러 개의 서버에 각 가상공간을 구분해 관리할 때, 가상공간끼리는 서로 시각적, 청각적 사방으로 가로 막인 벽으로 인한 단절이 존재한다. 즉, 건축물내의 가상공간을 예를 들 때 각 건물내의 방은 벽면으로 인한 시각적, 청각적 단절 효과가 있으며 이러한 사실은 다중 참여 가상공간내의 메시지 트래픽을 절감시킬 수 있는 단서를 제공하게 된다[5]. 이로 인하여 특정 분신은 특정시간에 하나의 가상공간에 반드시 포함되어야 하며 각 가상공간끼리는 서로 배타적이기 때문에 같은 가상공간 내에 존재하는 다른 분신과의 상호작용만 고려될 뿐 다른 공간내의 분신과는 어떤 정보 교환도 이루어지지 않는다.

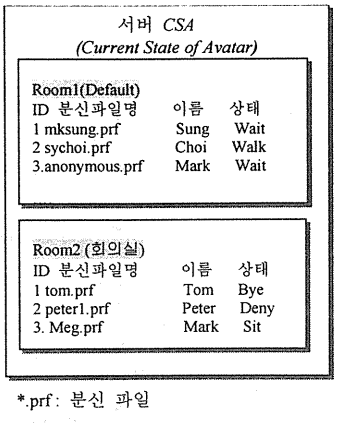
본 시스템에서의 하나의 방은 하나의 특정 서버에 의해 관리되며 특정 서버는 프로세싱

파워, 서버의 네트웍 기능에 따라 하나 혹은 두 개 이상의 방을 관리하게된다. 또한 각 방은 자신의 공간내의 분신에 따라 동적으로 멀티캐스트 그룹을 생성한다.

모든 방과 분신들은 텍스트 형태로 되어 있는 파일형태로 저장되거나 전송된다.

3.2 서버

분신서버는 현 클라이언트에 대한 등록 및 관리를 하며 클라이언트의 이벤트에 대한 라우팅역할을 한다[6][7]. 또한 분신 서버는 서버에 등록되어 있는 각각의 클라이언트의 현 가상공간상에서의 상태에 대한 정보를 갖게 되며, 특정 가상공간이 어느 서버에 존재 하는지에 대한 정보도 관리 하게 된다. [그림 2]는 서버내에 관리되고 있는 분신의 현재 상황에 대한 리스트 구조(CSA)의 예를 나타낸 것이다.



[그림 2] CSA 리스트 예

분신서버는 가상공간에 대해서는 전체 가상공간에서 자신의 관심영역(Area-of interest)을 가지며 자신의 영역에 대한 모든 기하학적 정보를 포함하고 있다. 클라이언트가 분신서버에 로긴을 하면 분신서버내에 존재하는 default가상공간에 대한정보를 클라이언트에게 보내주며 이에 대한 탐색에 따라 분신서버내에 가상공간이 존재하면 이를 보내주고 만약 존재치 않으면 특정 가상공간과 서버간의 연결 테이블을 참조하여 다른 분신 서버로부터 해당 가상공간을 전달 받는다. [표 1]은 서버-가상공간 연결 테이블인 AOI 테이블의 한 예를 든 것이다.

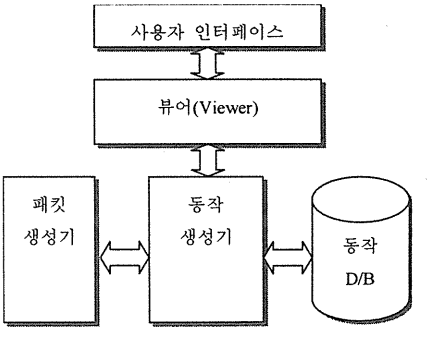
서버끼리는 서버가 담당하고 있는 가상공간에 대한 정보를 계속 업데이트하며 이를 유지, 관리한다.

[표 1] AOI 테이블 예

Server ID	*IP Address*	*AOI*
S1	*150.183.101.63*	*Room1, Room2*
S2	*150.183.101.65*	*Room3, Room4*
S3	*150.183.101.68*	*Room5*
S4	*150.183.101.70*	*Room6, Room7, Room8*

3.3 클라이언트

클라이언트는 분신의 브라우져 역활을 한다. 클라이언트는 자신의 분신에 대한 동작을 생성하며 다운 받은 가상공간에 대하여 분신을 통해 탐색(navigation)을 한다. [그림 3]은 클라이언트 모습을 나타낸 것이다.



[그림 3] 분신 클라이언트

사용자가 사용자 인터페이스를 통해 자신의 분신에게 행동 명령을 내리면 해당 명령에 대한 스크립트를 동작 데이터베이스에서 가져와서 동작 생성기을 통해 애니메이션 된다. 본 시스템에서는 분신의 동작을 수행시키기 위한 사용자 인터페이스로 키보드 매핑을 사용하였다. 즉, 하나의 동작에 하나의 키보드를 매핑하여 해당 키를 누르면 정해진 동작이 수행될 수 있도록 하였다.

사용자 인터페이스를 통해 자신의 분신에 내린 명령은 패킷생성기를 이용하여 패킷형태

로 바뀐 후 네트웍을 통해 서버로 전달되며 서버는 이 패킷을 해당 방에 있는 다른 클라이언트들에게 멀티캐스트 그룹을 참조하여 전송하게된다.

빠른 렌더링을 위해서 클라이언트는 서버측에 있는 가상공간 및 분신파일을 자신의 로컬 디스크에 복사 한 후 각 클라이언트 간에는 메시지 교환을 통해 가상공간내 분신의 일관성(Consistency)을 유지하게 된다.

3.4 네트웍 프로토콜

서버와 클라이언트간에는 일정한 형태의 패킷을 통해 메시지를 전송하거나 파일등을 전송하게 된다.

[그림 4]는 패킷의 구조를 나타낸 것이다

MSG type	String	UINT	Action	Object

[그림 4] 패킷 구조

- MSG type : 보내는 메시지의 Type
- String : 파일 전송일 경우 파일의 이름 혹은 채팅시 문자열
- UINT : Reserved Field
- Action : 분신의 상태
- Object : 파일(Option) , 분신의 위치 (Position, Orientation)

3.4.1 초기화

초기화 단계는 다음 3단계로 나뉘어져 있다.

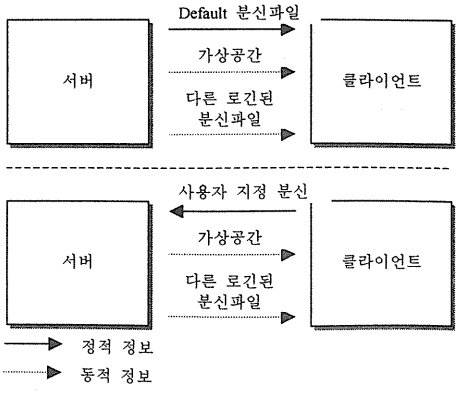
1)서버 측에서의 서버 프로그램을 구동 및 필요한 가상공간 파일들을 초기화

2)클라이언트의 해당 서버에 로긴

3)클라이언트의 가상공간과 분신 파일등 필요한 정보를 서버로부터 얻는 단계로 이루어져 있다.

처음에 서버는 자신의 서버 프로그램을 구동 함으로서 자신이 관리하는 가상공간파일들에 대한 초기화를 수행한다. 그 후에 클라이언트는 서버에 접속을 시도한다. 클라이언트는 서버측에 내장되어 있는 Default 분신을 사용 할 수도 있고 자신이 갖고 있는 사용자 지정 분신을 사용할 수도 있다. Default 분신을 사용할 경우에는 서버로부터 Default 분신과, 가상공간 그리고 해당 방에 이미 있었던 다른 분신 파일들을 다운로드 받게되며, 반면에 사용자 지정 분신을 사용할 경우에는 다른 클라이언트에게 그 분신 파일을 전송하기 위해 서버측으로 자신의 사용자 지정 분신을 전송 하게된다. 후에 서버는 새로운 분신이 방에 도착하면 그 방의 멀티캐스트 그룹을 참조하여 그 방의 다른 클라이언트들에게 새로 도착한 분신 파일을 전송한다.

새로 가상공간에 도착한 클라이언트는 서버로부터 해당공간의 CSA를 참조 받아서 기존의 클라이언트로부터 나온 분신들의 상태를 전송받는다. 이는 기존의 분신들의 상태를 그대로 자신의 클라이언트에 표현하기 위한 절차이다. [그림 5]는 분신 클라이언트와 서버간에 이루어진 초기화 단계를 나타낸 것이다. 분신은 여러 방들을 옮겨다니기 때문에 가상공간과 다른 사용자 분신 파일은 동적 정보이다.



[그림 5] 초기화 단계

가상공간, 즉 방의 전송이 끝나면 클라이언트는 방에 대한 분석을 통해서 다른 가상공간으로의 이동 통로가 되는 문의 위치, 가상 오브젝트들의 위치들을 파악한다. 이 정보는 처음 가상공간에 분신이 들어 왔을때의 초기 위치을 정하거나 가상 오브젝트와의 충돌감지시 사용된다.

3.4.2 분신간 상호작용

일단 초기화가 끝나면 분신간에는 메시지 교환을 통해 상호작용을 하게된다. 분신의 상태는 분신의 동작에 따라 달라지며 변화된 자신의 상태는 패킷 형태로 변환되어 서버측에 전달된다. 서버는 CAS리스트를 참조하여 다시 그 방의 다른 클라이언트에게 이 패킷을 멀티캐스팅해서 다른 클라이언트내에 있는 자신의 분신의 상태를 변화시킨다.

본 시스템에서 사용되는 분신의 동작은 총 9가지 이며 모두 Direct Kinematics기법에 의해 만들어졌다. 9가지의 동작은 크게 분신의 위치(Position, Orientation)가 변경되는 동작과 위치를 변경하지 않는 동작으로 구분되며 동작 수행 방식에 따라 한번명령을 수행한 후 곧바로 Wait상태가 되는 one-shot 동작과 한번 수행시킨 후 다시 토글하지 않는 한 명령을 계속 수행하는 Toggle동작으로 구분할 수 있다.

[표 2]는 본 시스템에서 사용된 분신의 동작을 나타낸 것이다

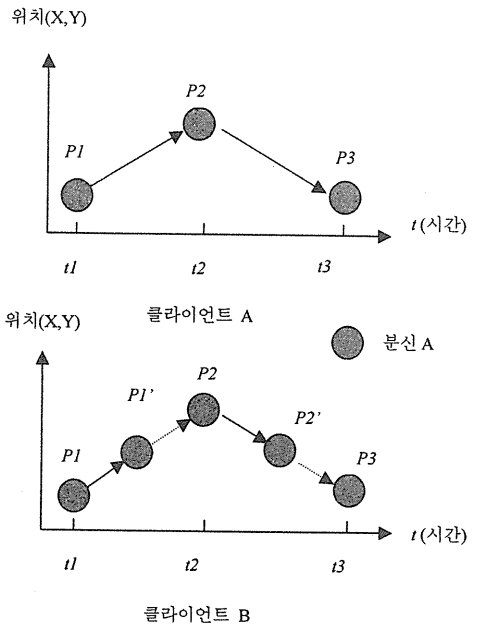
[표 2] 분신의 동작

Type	의 미	분신 위치 변경 여부	동작 수행방식
WALK	앞으로 전진	o	Toggle
AGREE	고개를 상하로 흔듬	x	Toggle
DENY	고개를 좌우로 흔듬	x	Toggle
BOW	허리를 굽힘	x	one-shot
BYE	손을 좌우로 흔듬	x	Toggle
SIT	자리에 앉음	x	Toggle
JUMP	제자리 점프	x	one-shot
TURN LEFT	좌측으로 방향을 바꿈	o	one-shot
TURN RIGHT	우측으로 방향을 바꿈	o	one-shot

모든 분신의 동작은 클라이언트의 모션 데이터 베이스에 스크립트 형태로 저장되며 사용자가 동작을 명령하면 모션생성기를 이용해 애니메이션 된다.

앞으로 전진하는 명령의 경우 Toggle 명령이므로 다음명령이 도착할 때까지 정속도로 계속 앞으로 전진한다. 이때 네트웍 지연(Latency) 때문에 각 클라이언트들은 같은 분신에 대하여 앞으로 전진하는 명령을 서로 다른 시간에 수행 될 수 있으므로 이는분신의 위치에 대한 불일치를 유발시킨다. 이를 보정하기 위하여 어느 특정 분신의 상태가 변하면, 즉 분신에 새로운 동작 명령이 수행되면 그에 해당되는 패킷에 그 시점에서의 분신의 위치값을 패킷에 실어보내게 되며 서버를 통해 받는 다른 클라이언트들은 그 패킷내의 위치정보를 기반으로 해당 분신의 위치(Position, Orientation)를 보정한다.

기존의 NPSNET등과 같은 분산 공유 가상공간에서 각 클라이언트와 서버간의 움직이는 물체에 대한 정보 교환 시 사용되는 DIS(Distributed Interactive Simulation)를 기반으로 하는 Dead Reckoning 알고리즘은 각 클라이언트에서 움직이는 물체의 초기값과 속도를 기반으로 다음 위치를 추측하면서 실제 위치값과 추측한 위치값이 어느 임계값보다 크면 다시 패킷을 전송하는 방법을 사용함으로서 네트웍 트래픽을 줄이고 있다[8]. 본 시스템에서는 앞으로 전진하는 명령의 경우 정속도 운동을 가정하였으므로 동작을 명령하는 데 소요되는 네트웍 지연만 존재치 않는 다고 가정하면 같은 움직이는 물체에 대해서는 각 클라이언트에서 모두 같은 위치를 유지하게 된다. 그림6은 네트웍 지연으로 유발되는 클라이언트끼리의 분신 위치 보정 과정을 나타낸 것이다. 예를 들어 클라이언트A로부터 나온 분신A에게 클라이언트 A가 t1시점에 앞으로 전진하는 명령을 내리면 이 명령은 패킷으로 변환되며 서버에 전달되며 서버는 그 방의 다른 클라이언트B에 패킷을 다시 전달한다. 이때 해당 패킷의 네트웍 지연으로 패킷이 늦게 도착하여 명령을 늦게 수행한다면, 실제 클라이언트A상에 분신A는 P2의 위치에 있는 상태에서 클라이언트B의 분신A는 P1'에 오게 된다 이때 다시 t2 시점에서 분신A의 동작을 멈추게 한다면 이 패킷은 다시 클라이언트B에 전달되며 이 패킷내의 분신의 위치정보를 이용하여 P1'~P2까지의 거리를 보정하게 된다.

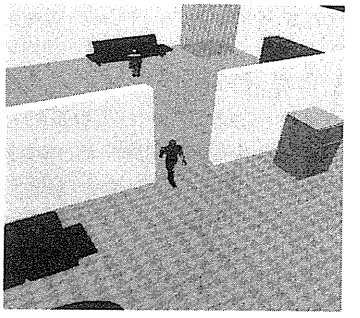


[그림 6] 분신 위치 보정

3.4.3 분신간 방이동

분신의 방이동은 위에서 언급한 바대로 모두 방문을 통해서 이루어진다. 분신이 방문에 접근하여 충돌이 되면 초기화 단계 중 3)번 단계를 다시 수행 함으로서, 이동하고자 하는 가상공간 및 그방에 있는 다른 분신에 대한 정보를 전송 받는다. 또한 예전 방의 다른 클라이언트 및 새로운 방안의 다른 클라이언트에게 메시지를 보내 분신이 탈퇴 및 새로운 분신이 도착하였음을 알린다.

분신이 가상공간을 이동하여 방문 사이를 지나갈때는 두 방을 동시에 랜더링 함으로서 두 방이 연결되어 있음을 인식하도록 하였다. [그림 7]은 한 분신이 가상공간을 이동하는 시점을 나타낸 것으로 두 방이 동시에 보임을 알 수 있다. 해당 분신이 새로운 가상공간으로 완전히 진입한 후에는 예전이 있던방을 랜더링 하지 않음으로서 속도 향상에 도모 하였다.



[그림 7] 분신의 가상공간 이동

3.4.4 분신의 서버이동

만약 분신이 지금 로긴하고 있는 서버가 관리하고 있는 방들을 벗어난 다른 서버가 관리하는 방으로 이동하길 원하면 서버는 AOI 테이블을 참조하여 해당 서버의 어드레스와 포트번호를 메시지에 실어 보내 주면 클라이언트는 보내준 정보를 기반으로 초기화의 2)단계부터 다시 시작한다

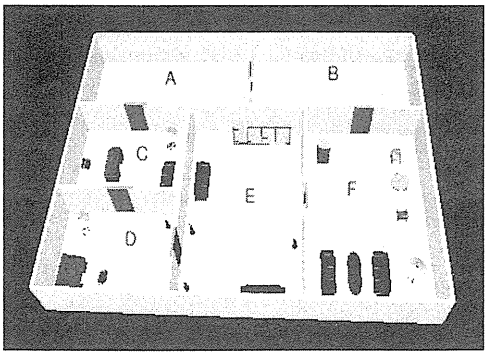
4. 프로토 타입

위의 설명을 기초로 프로토 타입 시스템을 개발 하였다. 개발 환경은 다음과 같다.

- 플랫폼 : Pentium 166 Mhz(3차원 가속기 카드 장착)
- 메인메모리 : 64M
- 라이브러리 : Open Inventor
- 사용언어 : Visual C++
- 네트웍환경 : LAN

위의 개발환경은 실제로는 직접적으로 랜더링을 수행하는 클라이언트 프로그램을 위한 것으로 서버 프로그램은 단순히 가상공간 관리 및 패킷 라우팅 역할만을 수행하므로 고성능의 그래픽 기능은 필요하지 않다.

본 시스템에서 사용된 가상공간은 가상 오피스를 대상으로 하였다. 가상 오피스 내에는 총 6개의 방으로 이루어져 있으며 공간을 2개의 분산 서버에 의해 관리하도록 하였다. [그림 8]은 전체 가상오피스의 모습을 나타낸 것이다.



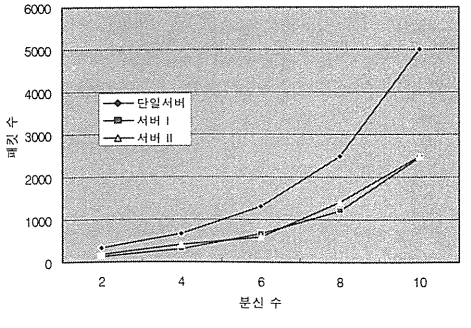
[그림 8] 가상 오피스

[표 3]은 각 서버가 관리하는 방의 파일 크기를 비교한 것이다.

서버 I		서버 II	
A	4KB	D	93KB
B	4KB	E	162KB
C	135KB	F	168KB

[표 3] 가상공간 크기 비교

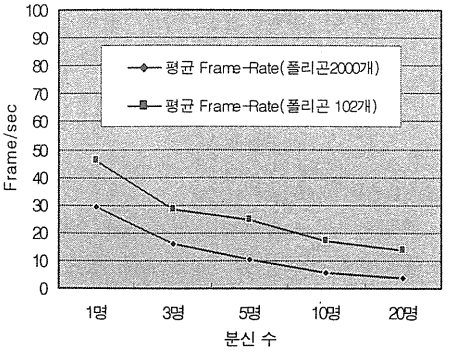
[그림 9]는 하나의 단일 서버가 모든 방을 관리할 때의 패킷량과 [표 3]에서와 같이 서버I과 서버II로 분산시키고, 모든 분신들이 고루게 분산 되었다고 가정했을 때 평균 패킷량을 비교한 것이다.



[그림 9] 단일 서버 / 분산 서버

[그림 9]에서 보듯이 단일 서버로 전체 가상공간을 관리 하였을 때 보다 2개의 서버로 분산관리 했을 때가 네트웍 트래픽면에서 효율적임을 알 수 있다.

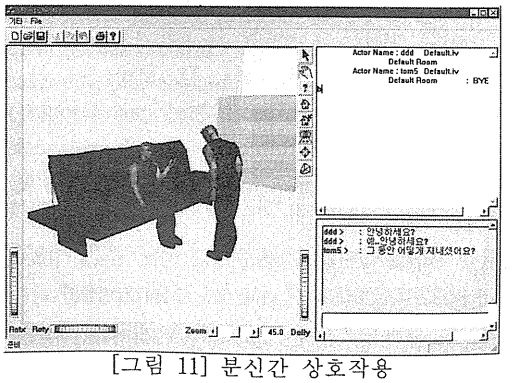
[그림 10]는 분신의 폴리곤 갯수와 분신의 수에 대한 평균 프레임 레이트를 분석한 것이다.



[그림 10] 분신수에 대한 프레임 레이트

[그림 10]에서 나타나 듯이 분신의 폴리곤 개수가 2000개이고, 분신의 수가 최대 20명때는 전체 그래픽 성능이 10 Frame/sec 미만으로 떨어지지만 폴리곤 개수가 100여개 정도일 때는 10 Frame/sec이상임을 알 수 있다. 즉, 분신의 폴리곤 개수가 전체 그래픽 성능에 많은 영향을 미치게 된다.

[그림 11]은 2명의 클라이언트가 자신의 분신을 이용해 어느 특정 방에서 서로 인사하는 모습을 나타낸 것이다.



[그림 11] 분신간 상호작용

5. 결론 및 향후 계획

본 논문에서는 여러개 방으로 이루어진 분산 가상공간내 다중 참여자를 지원하는 가상현

실 시스템을 위한 네트웍 구조 및 필요 프로토콜을 개발 하였고 이를 검증하기 위한 프로토타입 시스템을 LAN상에서 개발 하였다.

향후 연구방향으로 LAN환경 뿐 아니라 전화선을 이용한 모뎀 네트웍을 지원하여야 할 것으로 여겨지며 보다 많은 다중 참여자를 지원하기 위한 효율적 네트웍 관리 미들 웨어의 개발이 필요할 것으로 여겨진다.

6. Acknowledgement

본 연구는 시스템공학연구소에서 수행하는 "분신 행동양식 처리 및 상호작용 소프트웨어 개발(과제번호: 9P00200)" 결과물의 일부입니다.

참고문헌

[1] Social Virtual Reality Research, http://www.merl.com/threads/social/index.html

[2] Tog K. Capin, Igor Sunday Pandzic,et al : A Dead Reckoning Algorithm for Virtual Human Figures, Proc. IEEE VRAIS'97, pp161~169, 1997니다.

[3] W.Broll : Distributed Virtual Reality for Everyone-a Framework for Networked VR on the Internet, Proc. IEEE VRAIS'97, pp121, 1997

[4] M.Macedonia, M.zyda, D. Pratt et al : Exploiting Reality with Multicast Groups : A Network Architecture for Large-scale Virtual Environments, IEEE VRAIS'95, pp2~10, 1995

[5] B.Roehle : Channeling the data flood, pp32~38, IEEE Spectrum, March 1997

[6] T. Funkhouser : RING:A Client-Server System for Multi-user Virtual Environment, Symposium on Interactive 3D Graphics, 1995

[7] D.Amselem : A Window on Shared Virtual Environments, Presence Vol4, No 2, Spring 1995

[8] Deering, Stephen, "Host Extensions for IP Multicasting", RFC 1112, 1989