# VRML과 EAI를 이용한 3D 상호작용 가상공간시스템

염 창근°, 박 경환

동아대학교 컴퓨터공학과

# 3D Interactive Virtual Space System based on VRML and EAI

Chang-Gun Yum °, Kyung-Hwan Park

Dept. of Computer Engineering, Dong-A University

## 요 약

본 논문에서는 월드와이드웹상에서 3차원 인터페이스를 기술하는 표준 사양인 VRML과 자바의 EAI(External Authoring Interface)를 통하여 사용자 간에 발생하는 상호작용들을 동기화 시키는 방법을 소개한다. 제시한 방법에서는 별도의 독립적인 응용프로그램을 작성하거나 동기화를 위하여 확장된 VRML을 사용할 필요없이 웹브라우저와 바로 호환하여 사용할 수 있으며 기존의 방대한 웹문서와 연계가 쉬우므로 범용적인 자료구축이 가능하다. 하지만, 가장 최근의 VRML97에서 조차도 VRML이 제시하는 가상 공간 다중 사용자 환경의 지원은 아직 미비하다. 더 이상 단순히 3차원 월드를 탐험하는 시기는 지났으며, 같은 공간상에서 혼자가 아닌 여럿이 함께 하는 다중 사용자 환경을 제공해야 할 것이다. 이에 자바의 네트웍 기능과 가상 공간의 외부에서 동적으로 월드를 제어할 수 있는 EAI를 이용하여 부족했던 다중 사용자 환경을 지원함으로써 가상 공간 시스템을 구축할 수 있다. 그러나, 가상 공간에서 일어나는 이벤트는 단지 동일 브라우저에서만 유효하기 때문에 다른 사용자에게 전달할 때는 이벤트를 원격지의 이벤트와 연동하기 위하여 다른 방법이 필요하다. 사용자 상호작용 시스템에 있어서 이러한 이벤트를 결정하는 가장 중요한 요소는 아바타의 행동양식(avatar's behavior)이라 할 수 있다. 가상 공간에서 일어나는 이벤트의 대부분이 사용자에 의해서 발생되는 것들이다. 즉, 아바타의 행위에 따라서 사용자 서로 간에 상호작용이 일어나게 되며 이들 이벤트를 서로 동기화 함으로써 실시간 3차원 상호작용 시스템을 구현한다. 이렇게 구현된 시스템은 전자 상거래, 가상 쇼핑몰, 가상 전시회, 또는 3차원 게임이나 가상교육 시스템과 같은 웹기반 응용프로그램에 사용될 수 있다.

## 1 서론

최근 인터넷은 가장 접하기 쉬운 단어중의 하나이다. 정보제공자에 의한 일방적인 정보전달 방식에서 이제는 사용자 스스로가 정보를 찾아가는 시대가 된 것이다. 인터넷 서비스 중 월드와이드웹(World Wide Web)은 너무나도 많은 변화를 가져온 것으로서 단일 정보로만 관리되던 문서나 그래픽, 또는 음악이나 동영상과 같은 데이터들을 HTML이라는 기술언어를 통하여 하나로 묶여지게 되었다. 이들 평면적인 데이터는 곧 XML(eXtensible Markup Language)을 통해서 다시 한번 새단장할 것으로 기대된다. 이러한 흐름속에서 기존의 데이터들과는 성격을 달리하는 3차원 데이터를 웹에서 표현할 수 있도록 하는 VRML[1]이 등장하였다. 이 3차원 표현 기술은 VRML1.0 사양에서는 단순히 가상 공간을 만들고 움직임 조차 없이 단지 탐색하는 수준에서 그쳤지만, VRML97에 와서는 사용자와의 상호작용을 가능케한 효과적인 노드들이 추가되었다. 이들 노드를 통해서 사용자는 물체를 옮기거나 시간의 흐름에 따라서 실세계를 그대로 반영한 가상 공간을 만드는 것도 가능하다. 하지만, 가상 공간은 단지 3차원 정보를 표현하는 것에 머물지는 않는다. 가상 공간은 3차원으로 표현됨과 동시에 다수의 사용자를 대상으로 하기 때문이다. 즉, 다중 사용자 기반의 가상 공간이 제공되어야 한다. 그러나, 이러한 시스템을 구현하기에 따르는 문제점이 있는데, 두 가지 측면에서 살펴보면 3D 월드 표현 문제와 다중 사용자 환경 동기화 문제[2]이다. 3D 월드 표현에 있어서 인터넷에서 3차원 인터페이스를 기술하기 위한 표준인 VRML을 이용하는 것이 적합하다. 그러나, 동기화 문제에 있어서는 VRML이 표명하는 가상 공간 다중 사용자 환경 제공과는 거리가 멀다. VRML의 기본 사양으로는 객체의 추가나 삭제등의 갱신 작업을 외부로부터 동적으로 수행하지 못해서 VRML이 표명하는 다중 사용자 환경을 지원하지 않는다. 그래서, 이를 개선하기 위해 제안된 것이 EAI(External Authoring Interface)이다[2]. EAI의 등장으로 브라우저 외부에서 VRML을 제어할 수 있게 됨으로써 상호작용적(interactive)이며 동시에 동적인 월드를 구축할 수 있다. 이런 동기화된 시스템을 설계하는데 있어서 가장 중요한 요소는 아바타의 행동양식(avatar's behavior)이라 할 수 있다. 가상 공간에서 일어나는 이벤트의 대부분이 사용자에 의해서 발생되는 것들이다. 즉, 아바타의 행위에 따라서 사용자 서로 간에 상호작용이 일어나게 되며 이들 이벤트를 서로 동기화 함으로써 실시간 3차원 상호작용 시스템을 구현하

게 된다. 가상 공간이 VRML을 이용함에 따라 직관적이며, 통일된 인터페이스를 제공할 수 있어서 사용자는 실세계를 방문한 듯한 느낌을 받을 수 있다. 또한, 기존에 쉽게 접할 수 있는 브라우징 시스템을 그대로 이용하므로 사용자가 가상 공간을 이용하기 위한 추가부담 없이 가상 공간 전반에 접근 할 수 있다. 가상 공간의 이점은 자칫 평면적이고 단조로워 질 수 있는 웹기반 시스템을 3차원 가상공간을 제공함으로써 흥미유발 및 몰입(immersion)하도록 유도하는데 있다.

## 2 3D 상호작용 가상공간시스템

기존의 채팅 프로그램과 같이 상호작용 가상공간시스템은 그렇게 큰 차이는 없다. 채팅처럼 사용자 간에 대화를 나누며, 대화방을 옮겨 다니듯이 가상 공간 상에서 장소를 이동시킨다. 하지만, 두 환경이 각각 제공하는 정보의 양은 매우 차이가 있다. 이러한 이점 때문에 최근의 3차원 주류의 프로그램들이 쏟아져 나오는 것은 당연한 일인지도 모른다. 가장 쉽게 접할 수 있는 프로그램을 든다면 3D게임이나 그래픽 프로그램을 예로 들 수 있고, 이렇게 대중성 있는 프로그램들을 통해서 우리는 차츰 3차원 환경에 익숙해져 왔다. 이런 환경의 변화는 머지 않아 운영체제의 인터페이스까지도 바꿔게 할 것이며 이를 예측할만한 기술을 살펴본다면 당연 VRML의 등장이다.

### 2.1 VRML

VRML은 다양한 가상 공간을 저작 가능하게 하며 단일 사용자와의 효과적인 상호작용을 지원한다. 차츰 웹사이트를 VRML로 구축한 사례들이 늘어나고 있으며 VRML을 이용한 게임도 볼 수 있다. 그러나, 상호작용 가상공간시스템을 구축하기 위해서는 현재까지의 VRML97 사양으로도 많이 미비하다. 즉, 기본적인 VRML 사양으로는 아직까지는 구현하기가 어렵다. 이런 이유로 상호작용 가상공간시스템을 구현한 제품들은 VRML을 고려하지 않고 독립적인 응용프로그램의 형태로 제품을 개발하거나 고려하더라도 독자적으로 확장된 VRML 사양을 이용하여 개발하고 있다.

### 2.2 다중사용자환경을 지원하는 실행환경

일반적인 VRML 브라우저는 VRML97 사양만을 충족하므로 가상 공간 다중 사용자 환경 시스템의 브라우저가 아닌 이상 동기화된 시스템은 기대하기 힘들다. 그래서, 다중 사용자 환경을 지원하는 시스템들은 특정 응용프로그램을 제공하고 있으며 각각 독자적인 동기화 기술들을 가지고 있어서 시스템간의 호환성이 부족하다. 반면에 VRML의 사양을 만족하면서 확장된 형태의 브라우저 제품도 있어서 매우 다양하다. 확장형 브라우저는 표준 VRML에서 지정하는 노드 외에 자체적인 확장 노드를 추가한 것으로 동기화 시스템 목적에 맞게 추가하였다. 이런 제품들을 구분해서 프로그램과 브라우저 별로 살펴보면 먼저 Circle of Fire Studios사의 Active Worlds2.0[3]과 Chaco사의 Pueblo나 VRScout[4]등의 응용프로그램 형태가 있으며, Blaxxun Interactive사의 Community Server3.0[5]과 Sony사의 Community Place와 같은 확장형 브라우저가 있다. 이 중에서 Blaxxun Interactive사의 Community Server3.0과 Community Clients는 VRML97사양을 충분히 지원하는 가장 보편화 된 시스템이다. Community Server와 Clients는 사용자마다 독자적인 월드를 구축할 수 있게끔 월드 파일을 공급해주며 이 파일을 자신의 시스템에 저장한 뒤 아바타를 추가하고 서버를 수행시키면 자신만의 가상 공간을 저작하게 해준다. 그러나, 이 시스템의 가장 큰 특징은 다양한 아바타의 제공을 들 수 있다. 이것은 다양한 사용자의 욕구를 수용한 것으로 사용자는 자신이 원하는 아바타를 취사선택하여 월드를 탐험할 수 있게 된다. 이 들 아바타들은 확장성을 고려하여 설계되었으며 써드 파티등을 통해서 꾸준히 새로운 아바타들이 생겨나고 있다.

### 2.3 EAI를 이용한 VRML 월드 동기화

일반적인 브라우저를 사용하면서 VRML 월드를 동기화 하려면 먼저 스크립트(Script) 노드와 EAI(External Authoring Interface)[5]에 대한 이해가 선행되어야 한다. 스크립트 노드는 VRML에서 동적으로 애니메이션이 가능케하는 노드로써 스크립트 노드가 없다면 사용자의 요구에 즉각적으로 응답하는 가상 공간의 구현은 어렵다. 그러나, 스크립트 노드는 VRML 월드 파일에 포함되서 코딩되거나 따로 URL을 통해서 내장되는 특성 때문에 비록 동적인 가상 공간의 구현 가능하나 독립적인 월드 밖에는 구성할 수 없다. 이를 보안하기 위해서 나온 것이 EAI으로써 스크립트 노드와 구별하자면 특정 VRML 월드 파일과 관련없이 외부에서 VRML 월드 파일을 저작하게 하는 인터페이스이다. EAI는 자바 애플릿에 포함되어서 같은 웹페이지에 포함된 VRML 브라우저와의 동적인 인터페이스를 구축하고 여기에 브라우저에 적재된 특정 VRML 월드 파일에 접근할 수 있는 기능을 제공함으로써 스크립트가 수행하는 모든 기능을 똑같이 지원한다[7][8]. EAI를 통해서 VRML 월드의 동기화가 가능한 이유는 단지 애플릿으로 구현된다는 것인데, 이것이 바로 동기화를 가능하게 하는 직접적인 원인이 된다. 애플릿으로 구현된 EAI는 VRML 월드 파일을 동적으로 접근함과 동시에 애플릿이 가지는 네트웍 기능으로 원격지의 가상 공간 동기화 서버 프로그램과 통신한다. 애플릿은 새로운 사용자가 접속할 때 마다 그들을 대신하는 아바타를 생성하는 기능을 가지고 있으며 위의 기능들은 이벤트를 전달함으로써 수행한다. 단일 사용자가 발생시킨 이벤트를 변환 알고리즘을 거쳐 특정 메시지를 생성하여 서버로 보내고 이를 다시 서버에 접속한 나머지 사용자에게 전송하여 원격지의 사용자에게도 동일한 이벤트를 부여할 수 있게 되는 것이다.

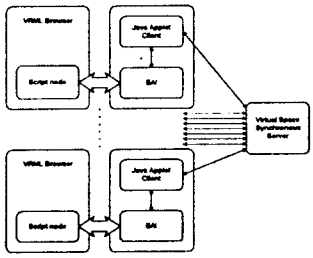
## 3 가상공간시스템의 개발

상호작용 가상공간시스템은 다수의 사용자[9]를 고려해야 하므로 단일 시스템에서 발생하는 예외상황보다 복잡하며 빈도도 높다. 더욱이 VRML 사양에서부터 인한 기능 지원의 부족에서오는 에러에 대한 대처 정책이 서버에서는 꼭 필요하다. 이러한 구현 기법과 더불어 설계 시점에서 고려한 사항으로는 아바타의 행동양식이다. 가상공간시스템의 중심은 바로 아바타이며 아바타가 발생하는 이벤트의 흐름이 곧 사용자간의 상호작용으로 연결되기 때문이다. 이들 아바타의 생성에서부터 소멸시점까지를 하나의 시스템 주기로 해서 각각의 시점에서 발생되는 이벤트를 묶어 관리토록 하였다.

### 3.1 설계 모델

상호작용시스템은 기본적으로 클라이언트-서버 모델을 사

용하였다. 서버에는 접속 초기에 전송하는 VRML 외형 정보와 클라이언트에게 적재될 애플릿 이진 코드가 있으며 클라이언트와 지속적인 연결을 유지하면서 동기화 작업을 수행하는 가상 공간 동기화 서버 프로그램이 위치하게되고 클라이언트에는 전송된 애플릿이 로컬 시스템의 VRML 브라우저와 인터페이스를 생성하고 메시지 전송을 담당하게 된다.



[그림 1] 클라이언트-서버 동기화 모델

[그림1]은 다중 사용자 기반의 가상 교육 공간 동기화를 위한 모델의 간략화 된 구성이다. EAI는 VRML 브라우저로의 스크립트 처리를 위한 인터페이스를 제공하며, 자바 애플릿 클라이언트는 가상 교육 공간 동기화 서버와 이벤트의 실시간 처리를 담당한다.

1) 서버의 역할

① 각 클라이언트의 아바타 정보를 유지(사용자 ID와 성별 및 아바타 위치와 방향)
② 클라이언트 간의 이벤트 전파에 따른 동기화
③ 이벤트 중재

2) 클라이언트의 역할

① 서버로부터 전송된 메시지 처리
② 사용자로부터 발생된 이벤트의 처리 및 메시지화

### 3.2 시스템 개발 환경

상호작용 가상공간시스템을 수행하려면 EAI를 지원하는 브라우저 이어야만 한다. 다행히도 대부분 접할 수 있는 브라우저들은 거의 EAI를 지원하고 있다. 수행 가능한 브라우저로는 Netscape Communicator 4.06 or later for Win32, Netscape Communicator 4.0 for IRIX, MS Internet Explorer 4.0 for Win32 등이며 VRML97 Plugin들로는 Cosmo Software Cosmo Player 2.1 or later for Win32, Cosmo Software Cosmo Player 2.1 or later for Macintosh, Intervista WorldView 2.0 or later for Win32, Blaxxun interactive CC3D 등이다. 여기에서 VRML97 Plugin들은 제약 사항이 없으나 브라우저에는 제약 사항이 따른다. 3D 상호작용 시스템에서 쓰인 애플릿의 이벤트 모델이 자바 1.1 이벤트 모델을 쓰고 있기 때문에 넷스케이프는 실험결과 Communicator 4.06 이상 버전에서부터 1.1 이벤트 모델을 지원하였기에 그 이후의 버전에서 제대로 동작하였고, Internet Explorer는 4.0 버전 이후부터 1.1 이벤트 모델을 지원하므로 역시 수행 가능하였다. 그러나 이들 브라우저와 Plugin들은 가장 손쉽게 접할 수 있는 프로그램들이므로 시스템을 수행하는데 문제점은 없다.

### 3.3 구현 방법

시스템에서 가장 핵심사항은 아바타의 행동양식이다. 아바타는 사용자의 분신으로서 가상 공간을 사용자를 대신해서 탐험하게 된다. 다중 사용자간에 일어나는 상호작용은 모두가 이 아바타의 영향으로 생성되고 아바타를 중심으로 이벤트들을 분석하면 대부분의 이벤트를 처리할 수 있게 된다. 이들 아바타가 생성시키는 이벤트를 조사하면 두가지로 나눠지는데 먼저 아바타 자신에서 발생하는 이벤트와 아바타가 외부의 물체에게 가하는 이벤트로써 양분된다. 아바타에서만 발생하는 이벤트들은 새로운 아바타가 가상 공간에 생성되거나 소멸되는 이벤트와 이동하는 방법인 걷는것과 뛰는 것, 그리고 날아다니는 등 여러가지가 있으나 현실세계를 모델링하였으므로 걷는 것과 뛰는 것의 두가지만을 아바타의 이동 방법으로 정하였다. 또한 아바타는 가상 공간에서 특정 방향으로 회전이 가능하다. 현재는 가상 공간에서 아바타의 머리방향으로만을 축으로 해서 회전이 가능하다. 즉, 왼쪽이나 오른쪽으로만 회전이 이루어진다. 끝으로 아바타는 자신의 감정을 표현할 방법이 필요하다. 이렇듯 아바타의 생성에서부터 소멸에 이르기까지 발생하는 이벤트는 매우 다양하고 이 이벤트의 실시간 처리가 이루어져야만 다중 사용자를 지원하는 시스템이 구현된다.

### 3.4 아바타의 행동양식 분석

#### 3.4.1 아바타 본위의 행동양식

가상교육 아바타에서 본위에서 발생하는 이벤트는 이동(translation)과 활동(action)가 있는데 이동은 걷기(walk)나 회전(rotation)과 같은 아바타의 자표 이동에 관한 이벤트를 나타내고 행위는 외부의 객체에 영향을 미치지 않는 범위 내에서 일어나는 인사(bow)나 춤추기(dance) 같은 이벤트를 의미한다

[그림 2] 아바타 본위의 행동양식

#### 3.4.2 아바타로부터 파생되는 행동양식

아바타로부터 파생되는 이벤트는 공간을 구성하는 객체 중에서 애니메이션 정보를 가지고 있는 동시에 센서에 의해서 동작하는 모든 객체로 이벤트를 발생시키는 행위(do something)를 의미한다. 사용자는 가상 공간에 있는 문을 열고 들어갈 수 있으며 이 때에 문은 사용자의 행위(behavior)에 의해서 작동하기 시작한다. 이런 행동들은 여러 사용자에 의해서 단일 객체를 동시에 이벤트를 발생시킬 경우가 있으므로 서버에서는 이벤트를 중재할 정책이 필요하다.

[그림 3] 아바타에 의해 파생되는 행위

이들 두가지의 이벤트 발생 경로를 중심으로 여러 개의 파생되는 이벤트들을 규정하면 이벤트의 확장시 기존의 이벤트 처리 루틴에 새로운 처리 코드를 추가만 함으로써 갱신이 가능하다. 본 논문에서는 기본적으로 간단히 네가지의 기본 행동으로 아바타의 기본 행위로 규정하였다.

### 3.4.3 전체 가상공간의 데이터 적재

처음 사용자가 가상 공간에 접속하게 되면 서버에서는 VRML 월드 파일을 클라이언트에게 전송한다. 이때는 단지 HTTP전송 프로토콜에 의해서 비동기로 전송되며 동적으로 갱신이 필요없는 가상 공간 외형 정보와 행위를 기술한 정보들이 아바타 정보를 제외하고서는 이시점에 적재된다. 동시에 VRML 브라우저와 통신할 애플릿이 서버로부터 적재되며 이 애플릿은 가상 공간 동기화 서버와 연결되어 이벤트의 실시간 처리를 담당한다. 애플릿은 곧바로 VRML 브라우저와 상호작용 하기 위한 초기화 작업을 수행한다. 아래의 코드는 서버와의 연결을 설정하는 작업이다.

```
try {
    s = new Socket(host, port);
    in = new DataInputStream(s.getInputStream());
    out = new DataOutputStream(s.getOutputStream());
//연결설정의 여부를 결정.
    connected = true;
//T h r e a d !!!
     (reader = new Thread(this)).start();
       if(doc.length() > 0)
         put("doc" + sep + doc);
         } catch(Exception e) {
         ex = e;
         System.err.println("연결할 수가 없습니다.: "+ex);
         System.out.println("connection error"); }
```

초기화 작업이 끝나면 애플릿은 가상 공간 동기화 서버(Virtual Space Synchronous Server)와의 지속적인 연결을 유지하게 된다. 초기화 작업에서 수행되는 코드는 VRML 브라우저에 의해 메모리에 적재된 가상 공간 정보에서 이후부터 수행될 모든 이벤트의 입출력이 일어나는 최상위 노드의 이름값을 가져오는 것으로 이어진다. 각종 노드들에 접근하기 위한 작업이 아래의 코드에서 행해진다.

```
browser = Browser.getBrowser(this);
    if(browser == null)
    {
      JSObject win = JSObject.getWindow(this);
      JSObject doc = (JSObject) win.getMember("document");
      JSObject embeds = (JSObject) doc.getMember("embeds");
      browser = (Browser) embeds.getSlot(0);
    }
     try {
      avatar = (Node) browser.getNode("Avatar");
      root = (Node) browser.getNode("ROOT");
      viewpoint = (Node) browser.getNode("VP");
      SyncStart = (Node) browser.getNode("SyncStart");}
catch (Exception e) { System.out.println("Node Exception"); }
```

getBrowser 명령을 이용하여 애플릿은 VRML 브라우저의 인터페이스를 가져오는데 이 값이 null이면 Netscape의 Communicator이며 따로 인터페이스를 가져오는 코드가 추가로 필요한 반면 null이 아니라면 브라우저가 IE4.0 으로 인식된다. 인터페이스를 얻었으면 아바타의 이동 및 회전 이벤트를 발생시키는 영역인 Proximity Sensor 노드의 DEF 이름인 Avatar의 레퍼런스를 저장하는 동시에 아바타의 외형 정보를 추가하게 될 ROOT 노드를 저장한다. 이 외에도 추가로 동기화에 필요한 노드의 레퍼런스를 저장해야 한다.

### 3.4.4 아바타의 생성

아바타는 사용자를 가장 잘 대변하는 분신으로서 가상세계를 탐험하게 된다. 사용자는 본 시스템에 접속할 때 자신의 ID와 성별을 입력하게된다. ID는 사용자를 구분하는 유일한 항목이며 서버에서는 클라이언트를 구분할 때 이 ID와 호스트의 이름을 정보로써 저장한다. 현재 지원 가능한 아바타는 단 2종류인 남성과 여성이지만, 아바타의 모델링과 시스템과는 독립적으로 작업이 가능하고 사용자마다 독특한 아바타를 창조하여 서버에서 관리할 수 있다. 이렇게 입력된 사용자 정보는 서버에서 분석된 후에 특정 위치에 특정 방향으로 바라보는 아바타를 생성시킬 수 있는 정보를 저장하고 이를 다시 나머지 사용자들에게 전송하게된다. 이 때 전송되는 정보로는 아바타의 ID, 성별, 좌표값, 방향값, 대화장소, 그리고 이름이다. 아바타의 실제 외형을 생성함에 있어서 두가지 다른 생성 규칙이 적용된다. 첫째는 처음으로 접속하는 사용자에 대해서 기존에 존재했던 다른 사용자들의 아바타 정보를 서버로부터 적재한 후 각각을 특정 위치와 방향으로 생성해야만 한다. 둘째로 기존에 존재했던 사용자들에게 처음 접속한 아바타의 정보를 서버로부터 적재해서 생성하는 것이다. 이들 아바타 정보로써 각각의 클라이언트에서는 자신을 제외한 나머지 아바타의 외형을 동적으로 생성하여 앞에서 보였던 ROOT 노드에 추가한다. 발생 경로를 살펴보면,

```
position.setValue(X, Y, Z);
direction.setValue(X, Y, Z, R);
newAvatar = new Avatar(gchat, name, s, position, direction);
AvatarArray.addElement(newAvatar);
addChildren.setValue(newAvatar.transArray);
```

먼저 서버로부터 전송받은 아바타의 위치와 방향 정보를 설정하고 아바타의 이름과 장소명, 그리고 성별을 이용하여 Avatar 클래스로부터 새로운 객체를 생성하고 이들 아바타의 정보를 저장하는 배열에 추가하는 동시에 실제 VRML 브라우저에 아바타의 외형을 추가함으로써 새로운 아바타의 생성이 이루어지고 이때 아바타의 위치와 좌표값을 새로 설정하여 준다.

### 3.4.5 아바타의 소멸

아바타의 소멸은 가상 공간에 접속하고 있는 어느 클라이언트가 접속 종료 메시지를 생성하거나 브라우저 자체를 종료시키면 발생한다. 아바타를 소멸하기 위해서는 DEF로 정의된 노드라고 모두 제거되는 것은 아니다. DEF로 정의된 노드일지라도 초기에 적재될 때 이미 존재했던 노드이어야만 하기 때문이다. 즉, 본 시스템에서 제안한 동적인 아바타의 추가, 삭제 기법은 DEF 노드로써는 해결되지 않는다. 이는 아바

타의 생성과도 관계가 있는데 아바타를 생성할 시점부터 아바타를 포함하게 될 상위 Transform 노드를 생성하고 이 노드의 하부에 아바타를 추가하여야 한다. 이렇게 추가된 노드는 소멸 이벤트가 처리될 때 상위 Transform 노드를 삭제함으로써 이루어지게 된다.

3.4.6 아바타의 이동 및 회전

1) 이동 및 회전 이벤트 검출

아바타 이동 검출을 위해서 ProxinmitySensor 노드를 사용함은 앞에서 밝혔다. EAI가 제공하는 기능 중 장면 안의 eventOut 필드를 갖는 노드로부터 발생한 이벤트 발생 감지를 이용하여 아바타로부터 발생하는 이들 두 이벤트를 감시하는 것이 필요하다. 아바타는 이 영역 안에서 이동하면 Viewpoint 노드의 필드인 position에서 position_changed 이벤트가 발생하고 회전을 하면 orientation 필드에서 orientation_changed 이벤트가 발생하게 된다. 감지된 이벤트는 각각 다른 처리 과정을 거쳐 서버로 보내어진다. 발생된 이벤트에 의해서 새로운 좌표값과 방향값은 각각 avatarposition과 avatarrotation에 실수값으로 저장하고 callback 함수를 호출한다. callback 함수는 아바타의 현재 위치 정보를 가져온다.

2) 실시간 이벤트 처리

상호작용 시스템에서 실시간 처리는 동기화에 있어서 매우 중요하다. 클라이언트의 가상 공간에서 아바타가 움직일때마다 발생하는 이벤트를 모두 서버로 전송하기에는 무리이며 서버에서 처리하기에도 부하가 크다. 이를 해결하기 위해서 본 시스템에선 타이머를 이용하여 규칙적으로 서버로 메시지를 전송하는 방법을 사용하였다. 센서에서 발생되는 이벤트들은 곧바로 변수에 저장되고 타이머가 발생하는 시간 간격보다도 빈번히 발생한다면 그 이전의 메시지는 무시되고 현재 발생한 메시지가 전송된다. 이렇게 함으로써 네트웍의 부하를 줄일수 있는 반면에 아바타의 이동이나 회전 행위가 부드럽지 않게 끊김을 유발할 수도 있다. 그러나, 세세한 메시지의 전송으로 인한 부하가 VRML 브라우저의 랜더링에 영향을 미치므로 둘 사이의 적절한 절충점이 필요하다. 이 절충점은 타이머의 시간 간격을 적적히 조절하여 최적의 효과를 나타낼 수 있도록 설정함으로서 이루어진다.

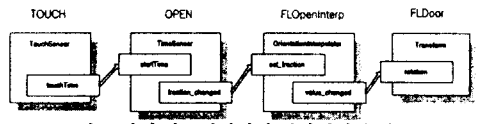
```
public void tick(Timer t)
{
    String movetemp = "doc", turntemp = "doc";
        if(movetemp.equals(movemessage)) ; else
          put(movemessage);
        if(turntemp.equals(turnmessage)) ; else
          put(turnmessage);
          movetemp = movemessage;
          turntemp = turnmessage;
}
```

3.4.7 아바타의 애니메이션

아바타 스스로가 발생시키는 이벤트 중에서 끝으로 아바타의 행위가 있다. 아바타는 사용자가 이동하거나 회전할 시에 걷는 동작을 실행한다. 걷는 동작을 위한 애니메이션 정보는 사용자가 처음에 서버에 접속할 때 적재하는 아바타의 외형 정보에 포함되어 있으며 이들 정보는 CoordinateInterpolator와 TimeSensor의 라우팅 경로에 의해 생성되어진다. CoordinateInterpolator는 TimeSensor로 부터 입력되는 시간값을 실행시간에 계산하며 아바타의 동작을 보간해서 애니메이션이 이루어진다. 클라이언트의 VRML 브라우저상에 나타나는 아바타들의 애니메이션을 위해서 특정 메시지는 필요하지 않다. 서버로부터 전송되는 아바타의 이동 정보만 있으면 클라이언트에서는 이동 정보에 맞추어서 애니메이션을 동기화하면 되기 때문이다.

3.4.8 애니메이션의 동기화 기법

아바타의 행위에 의해서 모든 이벤트들이 생성되지만 아바타 객체 자신이 발생시키는 이벤트를 제외한 나머지 이벤트들은 좀 더 처리하기가 까다롭다. 아바타가 특정 문을 열고 들어가려고 하는 경우를 예를 들어보면 문이 열리는 행위의 모든 애니메이션(animation) 정보는 접속 초기에 갖고 있지만 애니메이션의 시작 시점은 아바타가 문을 열려고 하는 순간에 발생한다. 이 이벤트들의 흐름을 그림으로 살펴보자.




[그림 4] 단일 시스템에서의 애니메이션 경로

아바타가 문을 열기 위해서 다가간 후 마우스의 클릭에 의해서 TouchSensor가 이벤트를 발생시키면 이를 TimeSensor에서 받아서 애니메이션을 시작하기 위한 시간값을 계산하여 이를 다시 PositionInterpolator로 넘겨준다. PositionInterpolator는 자신이 생성한 좌표값을 문에 해당하는 객체의 rotation 필드에게 전달하여서 자연스러운 애니메이션이 수행되게 된다. 그러나, 이런 일련의 작업 흐름은 단일 시스템에서만 일어나는 이벤트이기 때문에 원격지의 시스템에서는 전혀 이벤트의 발생 사실 조차도 알 수 없다. 이벤트의 시작은 사용자의 마우스 클릭에서부터 발생하여서 원격지 시스템에 똑같은 월드 정보가 있다고 하더라도 사용자에 의해서 물리적으로 발생한 이벤트를 원격지 시스템에서는 인식할 수 없기 때문이다. 그래서, 접속 초기에 적재하는 월드 외형 정보에는 애니메이션 정보를 가지는 모든 객체에 대하여 별도의 이벤트 전송을 위한 라우팅 경로를 재설정하는 작업이 요구된다. 이 때 사용한 노드로써 스크립트 노드가 사용되었다. 스크립트 노드는 VRML에서 지원하는 충분히 잘 설계된 노드를 사용자의 요구에 맞게 동적으로 구현하는 기능을 가지고 있다.

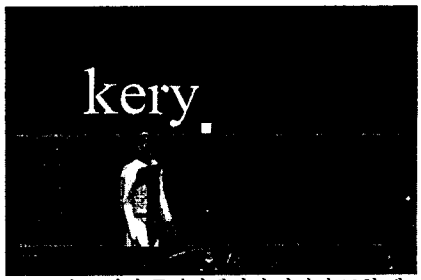
```
#VRML V2.0 utf8
DEF SyncTouchSensor Script {
    eventIn SFBool start
    eventOut SFTime stime
    field SFBool state FALSE
        url "vrmlscript:
         function start(value) {
             state = value
              stime = new SFTime();
         } " }
```

위 VRML 코드는 문의 애니메이션의 시작점을 결정하는

TouchSensor 를 동기화가 가능하도록 스크립트 노드를 정의하였다. 클라이언트에서 보내온 이벤트 정보를 서버에서 해석한 뒤 원격지 클라이언트 시스템에 전송하면 이 노드를 통해서 이벤트를 호출하는 것을 시작으로 해서 애니메이션의 시작을 결정할 수 있다. 즉, 애니메이션 정보를 가지는 모든 객체들은 그들의 행위를 시작하는 방법으로 두가지의 라우팅 경로를 가지게 되는 것이다. 하나는 로컬 시스템에서 발생하는 이벤트를 위한 경로이며 다른 하나는 나머지 원격지 클라이언트들에게 동기화가 가능하도록 하는 경로인 것이다. 이러한 메커니즘을 통해서 서버는 클라이언트 각각의 정보를 서버의 로컬 시스템에 유지하면서 모든 동기화 작업을 수행하게 된다.

3.4.9 아바타의 월드상에서의 구별방법 및 적용 예

서버에서 제어 가능한 아바타의 갯수에는 제한이 있는데 동일한 아바타가 동시에 같은 공간에서 존재할 수 있다. 또한 아바타의 외형만을 가지고서는 누구의 아바타인지를 분간하기는 어렵기 때문에 아바타를 구분 가능하도록 ID와 연계할 필요가 있다.




[그림 5] 가상 공간상에서의 아바타 표현 예

이를 해결하기 위해서 본 시스템에서는 [그림 5]와 같이 아바타를 항시 따라다니는 ID 문자열을 아바타의 머리 위쪽에 나타나게 하였다. 이렇게 함으로써 자신이 누구와 대화를 나누고 있는지를 알 수 있다. 그러나, VRML이 아직 한글 폰트를 지원하지 않아서 자신의 ID를 한글로 입력하면 월드상에서는 제대로 표시되지 않는 문제점이 있다.

## 4 결론 및 향후 연구 과제

3차원 정보를 표현하는 방법이 다양하기는 하나, 월드와이드웹기반의 3차원 인터페이스를 구현하기 위한 표준인 VRML은 그 목적의 명확함에도 불구하고, 아직까지 3차원 인터페이스의 구현에 머물고 있다. EAI의 등장과 함께 EAI의 인터페이스를 통해서 동적이기는 하지만 다중 사용자 환경이 빈약한 VRML을 자바 애플릿의 네트웍 기능을 통해서 VRML 월드 동기화를 해결할 수 있었다. 하지만, EAI를 지원하는 VRML 브라우저가 아직 많지 않은 점은 EAI를 이용한 기술의 호환성을 어렵게 하고 있다. 이런 문제는 EAI 뿐만 아니라, 인터넷을 기반으로 하는 대부분의 기술에 해당된다. 그럼에도 EAI는 VRML에게 외부 인터페이스를 제공해서 VRML의 측면에서는 큰 변화를 가져다 주었다. 즉, 자바가 응용되는 모든 곳에 VRML이 이용될 수 있는 것이다. 둘 다 인터넷을 기본 환경으로 독자적으로 발생한 기술들이지만 필요에 의해 결합되어 인터넷에 많은 변화를 주고 있으며 또한 더욱 다양한 변화를 제공할 것이라 예상된다. VRML 월드 동기화는 3D 월드를 구현하는 모든 곳에서 EAI의 기능을 확장할 수 있다. 하지만, 네트웍의 대역폭이 아직 이런 방대한 3D 정보를 수용할 수 없어서 월드를 구성하는 공간 정보의 양이나 아바타의 정보를 제한하는 눈에 보이는 한계도 있다. 게다가 EAI의 특성상 모든 월드를 접근할 수 있는 것이 아니기에 완벽한 동기화를 구현하는 데에 문제점이 없는 것은 아니다. 그리고, 다중 사용자를 지원하려면 사용자간에 발생하는 이벤트의 충돌을 해결해야만 한다. 동일 객체에 대해서 둘 이상의 아바타가 이벤트를 동시에 발생시킨다면 이벤트의 처리에 따라서 결과가 달라지기 때문이다. 이런 문제점은 향후 개선해야 할 부분이다. 또한 안정성 있는 시스템의 구현과 3D 공간 정보 저장, 추가부담이 없는 브라우징 시스템, 그리고 HTML 브라우저와 VRML 브라우저간의 유기적인 결합을 위한 인터페이스의 연구도 이루어져야 하겠다.

## 참고문헌

[1] Andrea L.Ames, David R.Nadeau, John L. Moreland, "VRML 2.0 Sourcebook", 1997

[2] VRML 2.0 External Authoring Interface (EAI) FAQ. http://www.tomco.net/~raf/faqs/eaifaq.html

[3] Active Worlds. http://www.activeworlds.com

[4] Pueblo, VRScout. http://www.chaco.com/pueblo

[5] Community Server 3.0. http://www.blaxxun.com

[6] Cosmo Player. http://www.cosmosoftware.com

[7] Finlayson Consulting VRML Group, "VRML 2.0 EAI FAQ" 1997. http://www.tomco.net/~raf/faqs/eaifaq.html

[8] Jed Hartman, Josie Wernecke,, "The VRML 2.0 Handbook", pp.133-136, 361-374, 1996

[9] Multi-User Environments with Moving Worlds, January 30, 1996, http://www.Cosmosoftware.com/developer

[10] Michael Natkin, An Introduction to VRML 2.0 Animation Techniques for the Production Animator, Silicon Graphics. http://vrml.sgi.com/VRML97/animPaper.html

[11] 염창근, 문석원, 박경환, "가상 대학 시스템에서 EAI를 이용한 VRML world 동기 화," 1998년도 한국정보과학회 봄 학술발표논문집 Vol.25, No. 1, pp745-747.