VRML을 이용한 2D 게임캐릭터의 3D 아바타 표현
(3D Avatar Representation of 2D Game Character Using VRML)

김선호* · 하수철**

1. 서 론

컴퓨터 게임의 캐릭터(character)란 게임의 주인공을 말한다. 시나리오에 의해 탄생되는 주인공은 각각의 성격을 가지게 되는데 이것은 게임의 분류와 밀접한 관계를 가고 있다. 롤 플레잉 게임의 캐릭터는 내적 심리와 아이템습득에 의한 변화를 가지고 있어야 하고 육성시뮬레이션 게임의 캐릭터는 역사적 배경과 성장에 따른 다른 모습으로의 변환데이터를 가지고 있어야한다. 이렇듯 시나리오에 의해 생성된 캐릭터는 플레이어와 게임 상황을 연결 시켜주는 중간자적인 역할을 하고 있다. 이러한 캐릭터의 행동은 표현이 필요하고 아케이드류와 어드벤쳐류의 캐릭터들의 행동은 간결한 행동의 표현으로 실시간 상황을 정확히 표현하여야 한다.

현재 사용 및 시판 중인 롤 플레잉 게임에서 주인공이 되는 캐릭터 상태 인식 수준은 단순한 표시기능으로 상태표시를 위한 인터페이스는 사용자에게 불편하다. 상태표현의 제약이 있는 스프라이트(Sprite)캐릭터는 다양한 형태 변형을 보여줄 수 없다. 이와 같이 캐릭터 상태파악이 용이하지 못함은 게임의 진행과 사용자들의 흥미를 감소시킴은 물론 게임의 상업적 성공에 장애 요소가 된다.

한편 현재 가상현실게임의 부각으로 기존 게임의 캐릭터에도 변화가 요구하고 있다. 그 중 가장 대중적이며 차세대의 Web 기반의 3D를 보여줄 수 있는 VRML을 이용한 캐릭터의 3D가시화 작업이 필요하게 되었다.

가상세계 구축 기술은 수입(import)된 가상 객체를 저장하고 검색하는 기술과 가상세계를 다양한 사용자의 관점에서 관찰할 수 있는 방법이 있다. 또한 선택된 가상 객체의 속성을 사용자의 요구에 따라 변경 시켜주는 기술인 상품성 중심의 서비스 품질 평가를 위한 표준화된 기준 및 사용성 평가 기반의 상호작용 개선기술이 그 예이다..

본 논문에서는 사용성 기반 서비스 품질 및 상호작용을 기반으로 하는 아바타(avatar) 제어를 중점으로 논의한다. 아울러 현재 출시되어진 게임의 2차원 캐릭터를 3차원 VRML 캐릭터로 전환시켜 사용자의 이해도를 향상시키고 가시적인 상태파악을 통해 게임의 몰입감을 향상시킬 수 있 방법에 대하여 논의한다.

2. 게임 캐릭터를 위한 Avatar의 생성과 제어

2.1 아바타의 생성[1,2]

순수 아바타 또는 클론(pure avatar and clones)

*대전대학교 공과대학 컴퓨터공학과 대학원
**대전대학교 공과대학 컴퓨터통신전자공학부 교수

는 실제 사용자의 모습과 행동을 그대로 컴퓨터에 반영시킨 가상인간으로, 실물에 가까운 얼굴을 가지며 실제 사용자와 일치하는 움직임을 갖는다. 가상인간의 움직임을 사용자의 움직임과 일치시키는 기법을 실시간 로토스카피(real-time rotoscopy)라 한다. 이것은 모션 캡쳐를 사용하여 움직이는 데이터를 입력받고 비디오화면의 캡쳐를 이용하여 사용자 얼굴을 텍스쳐 매핑으로 완성시킨다.

실시간이라는 메타포(metaphor)를 사용하는 인도되는 가상인간(guided virtual human) 아바타는 사용자의 움직임을 가상인간에게 반영한다는 점에서 순수 아바타와 동일하지만 사용자동작에 대한 정확한 데이터를 입력시키지 못하므로 위치정보만으로 방향과 움직임의 속도 등 역 운동학(inverse kinematics)과 같은 연산작용을 수반해야한다.

자율적 가상인간(autonomous virtual human)은 컴퓨터에 의해 제어되는 것을 말한다. 그래서 시스템은 이러한 자율적 가상인간은 제어하는 모듈이 필요하다. 또한 동일한 가상공간에서 가상의 물체와 다른 가상인간을 인지해야 한다.

지각과 상호작용이 가능한 가상인간(interactive-perceptive virtual human) 역시 자율적 가상인간과 같이 컴퓨터에 의해 제어되며, 다른 가상인간들이 수행한 행동을 인식하고 있어야 한다.

다음 표 1에서는 여러 종류의 아바타의 특성과 차이점을 보인다[3].

본 연구에 쓰여진 아바타는 지각과 상호작용이 가능한 가상 인간의 분류로 제작되었다. 아바타의 제작은 3차원 그래픽 객체인 3D Studio MAX R3를 이용하여 각각의 개별적인 객체를 생성하였다.

그림 1의 내용은 3D MAX를 이용하여 가상공간에 들어갈 아바타를 제작하는 과정이다, 이렇게 생성된 아바타는 VRML 파일인 *.wrl 파일로 수출(export)해서 가상현실공간으로 옮겨가면 그림 2와 같이 Web 브라우저에 생성되게 된다.

이때 모델링 된 객체들은 각각 독립적인 객체

표 1. 아바타의 분류

행 동	순수 아바타	인도되는 가상인간	자율적 가상인간	지각과 상호작용의 가상인간
평평한 땅 걷기	센서를 통해서만 가능.	걷는 엔진이나 속도, 보폭을 계산하는 함수가 필요	걷는 엔진이 필요	
굴곡이 있는 땅 걷기		현재 없음	자유롭게 걷는 엔진이 필요 일반화된 알고리즘이 필요함.	
움켜쥐기	장갑을 사용할 수 있지만 피드백을 구현하기 힘듬.	역 운동학	제공되는 함수로 가능	
장애물 피하기	가능하지 않음	그래프 이론에 기반 한 알고리즘을 사용하여 구현할 수 있음.	시각(vision)기반의 항해(Navigation)	
몸짓인식하기	제공하지 않음			제공되어야함
얼굴애니메이션	비디오카메라를 사용	적은 수의 파라미터를 사용	모델기반 애니메이션	
가상인간과 통신	컴퓨터가 생성하지 않음		제한된 통신	인식(perception)기반 통신
사용자와의 통신	실생활의 통신과 일치함		제공하지 않음	아바타를 통해 통신

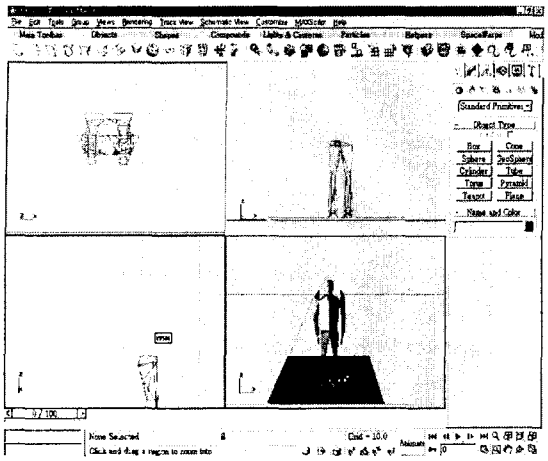

그림 1. 3D MAX를 이용한 3차원 객체 제작

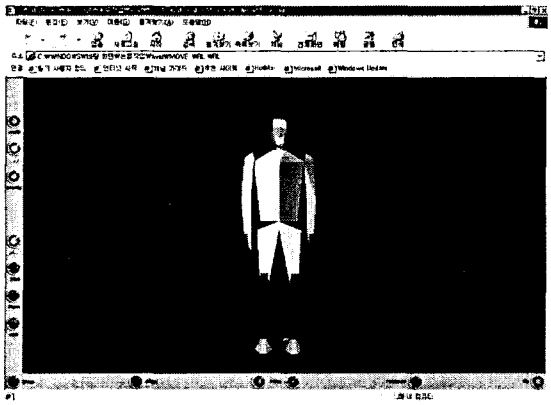

그림 2. 웹 브라우저상황의 아바타 모습

들이어야 한다. 왜냐하면 각각의 캐릭터 상태에 대한 제어를 해야 할 때 독립된 객체가 아닌 그룹화 된 단일 객체라면 게임 캐릭터의 상태를 정확하게 표현할 수 없기 때문이다.

생성된 아바타는 머리, 몸통, 팔 등에 대한 각각의 헤더부분이 생성된다.

```
# This node related to object [TITAN -- dynamic holder]
Transform {
  center 0.35 0.9288 -0.3
  children
# This node related to object [TITAN - rotating holder]
Transform {
  children
# This node related to object [TITAN - bounce]
Transform {
  children [
# This node related to object [TITAN - head sorting holder]
Transform {
  children
# This node related to object [TITAN - head]
Transform {
  children Transform {
   children Shape {
    appearance Appearance {
     material Material {
     }
{
    }
.....
# This node related to object [TITAN -- dynamic holder]
```

이텔릭체 부분은 아바타의 이벤트가 입력되게 되면 바로 AUTO 상태로 변화되어 Web 브라우저 상황에서 가동되게 된다.

2.2 게임캐릭터의 행위결정을 위한 아바타의 행위모델링

아바타의 가상공간을 구축하려면 우선 공간에 놓일 아바타를 모델링 하는 일이 필요하다. 아바타의 모델링은 형상 모델링과 행위 모델링으로 나눌 수 있다. 행위 모델링은 아바타가 외부로부터 어떤 자극(힘, 위협, 명령 등)을 받았을 때, 어떻게 반응할 것인가를 기술하는 것이다. 즉 행위결정을 위해서 행위 모델링이 필요하다. 일반적으로 행위는 두 레벨로 나누어 모델링 되는데, 상위레벨의 행위는 스크립트 언어를 이용하여 모델링하고, 하위레벨의 행위는 운동학적/동역학적으로 기술한다. 스크립트 레벨의 행위기술은 물체가 주어진 상황에서 어떤 행위를 할 것인가를 정하는 것이다. 만약 우리가 가상공간의 물리적 상태와

움직이는 물체나 사용자의 아바타에 대한 내면상태를 정확하게 알고 있다면 주어진 상황에서 어떤 행위가 일어날지 매우 정확하게 예측할 수 있을 것이다. 그러나 이것은 매우 어려운 작업이므로 주어진 상황에 대응되는 행위를 결정하는 문제는 어느 정도 작위적이다.

상위레벨의 행위 모델링의 최종결과는 운동학적 또는 동역학적 행위기술이다. 운동학적 행위기술은 해당물체의 속도나 가속도를 기술하고 동역학적 법칙레벨의 행위기술은 질량, 탄성계수 등 해당물체의 물리적 성질, 이 물체가 받는 힘, 만족해야 할 운동법칙, 그리고 다른 물체와의 관계에서 만족해야 할 제약조건(constraints)을 기술한다. 운동학적 또는 동역학적 행위기술을 만족하는 행위를 실시간에 생성하는 작업은 매우 어려운 문제로서 연구가 진행되고 있는 분야이기 때문에 본 논문에서는 모션 캡쳐 등을 이용하여 미리 동작을 만들어 놓고 이를 주어진 행위기술을 만족하도록 변형하는 방법을 이용했다[4].

대부분의 상용 게임들을 보면 로고 화면 다음에는 메뉴 화면이 나오게 마련이다. 이 메뉴를 구성하는 방법도 각 회사마다 독특한 방법을 취하고 있다. 예전에는 메뉴 같은 부수적인 것보다는 게임 자체에만 신경 써야 한다는 고정관념 때문에 메뉴 화면을 전혀 디자인하지 않은 채 대중에게 공개했다. 하지만 게임은 일종의 시대적, 사회적 문화를 대변한다고 본다면 메뉴 화면 자체와 그 인터페이스의 구성을 정교하게 해야 한다.

일반적인 게임의 인터페이스구성을 살펴보면 Staff, Story, Game, Option등의 메뉴를 두고 각 메뉴에는 자신의 고유 함수를 실행시킬 수 있는 프로그램 준비한다. 이러한 구성들을 GUI(Graphic User Interface)방식을 이용하여 구축하고 있다[5].

또한 그림 4와 그림 5는 일반적인 게임의 메뉴

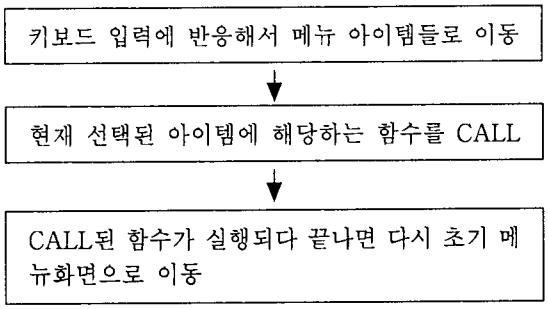

그림 3. GUI 진행단계

의 흐름도 이다. 일반적으로 기존의 롤플레잉 게임의 구성은 게임이 진행되는 과정에서 사용자의 호출이 있을 시에 메뉴가 디스플레이 되고 메뉴의 구성은 일반적으로 옵션과 캐릭터정보, 아이템정보, 저장, 게임종료 등으로 나뉘어 나타난다.

그러나 아케이드류의 메뉴방식은 초기화면에 메뉴가 나오고 그 초기화면에서 게임의 진행과 옵션, 스토리 등을 선택할 수 있다.

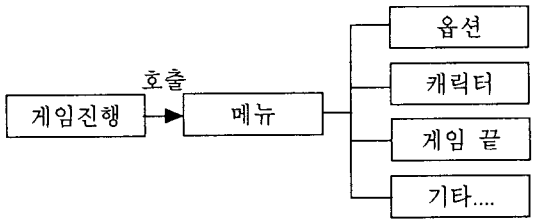

그림 4. 롤플레잉 메뉴 흐름도

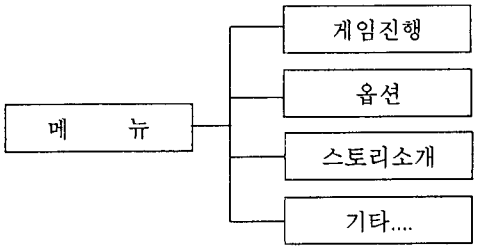

그림 5. 아케이드 메뉴

위에서 언급한 일반게임의 인터페이스를 보완하기 위하여 아바타를 통한 캐릭터의 상태를 3차

원으로 가시화 시켜 이것을 행위 모델링 하는 방법과 그로 인한 게임캐릭터와 사용자의 상호작용에 대한 개념은 그림 6과 같으며 이는 캐릭터의 상태요구를 쉽게 인식할 수 있을 것이다. 일반 인터페이스 방식의 캐릭터 상태표시는 2차 상태요구가 필요시, 다시 캐릭터의 이벤트로 돌아가는 반면 아바타를 통한 인터페이스방식은 2차 상태 요구를 아바타 상태표시에서 바로 알 수 있게 된다. 이것은 캐릭터의 상태 입력이 단편적인 데이터가 아닌 다양한 상태를 포함하며 캐릭터의 이벤트에서 상태 입력이 들어오고 있기 때문이다. 즉 일반 인터페이스 방식은 캐릭터의 상태표시에서 현재 사용자가 요구하는 하나의 상태만 출력하고 삭제된다. 따라서 사용자의 2차 상태요구가 캐릭터의 다른 상태를 요구할 시에는 다시 호출하여 다른 인터페이스로 출력하게 되는 단점이 있다. 그러나 아바타를 통한 인터페이스를 구축하게 되면 그러한 2차 상태요구가 있을 때 다양한 정보를 아바타가 저장하고 있다가 사용자에게 직관적이며, 신속하게 캐릭터의 상태를 표시할 수 있다[6,7].

3. 2D 캐릭터와 3D 상태의 상호작용

3.1 2차원 게임 캐릭터의 상태표현

게임상의 2차원 캐릭터가 사용자에게 현재의 상태를 인식시키기 위해서는 표 2와 같은 요소의 제어가 필요하다.

표 2. 캐릭터 제어 요소

필 요 요 소	제어 내용
충 돌	검색, 제어, 이동
아 이 템	습득, 저장, 인출
현 재 상 태	메뉴, 상태 창

표 2와 같이 캐릭터 제어의 필요요소 중 하나인 충돌은 2D 게임캐릭터가 3D 아바타에 캐릭터의 상태를 보내 아바타가 현재 진행되는 게임 캐릭터의 진행상황에 맞는 상태표현을 위한 가장 중요한 부분이다.

2D캐릭터의 간단한 충돌 메커니즘은 C++로 작성되었을 때 다음과 같다.

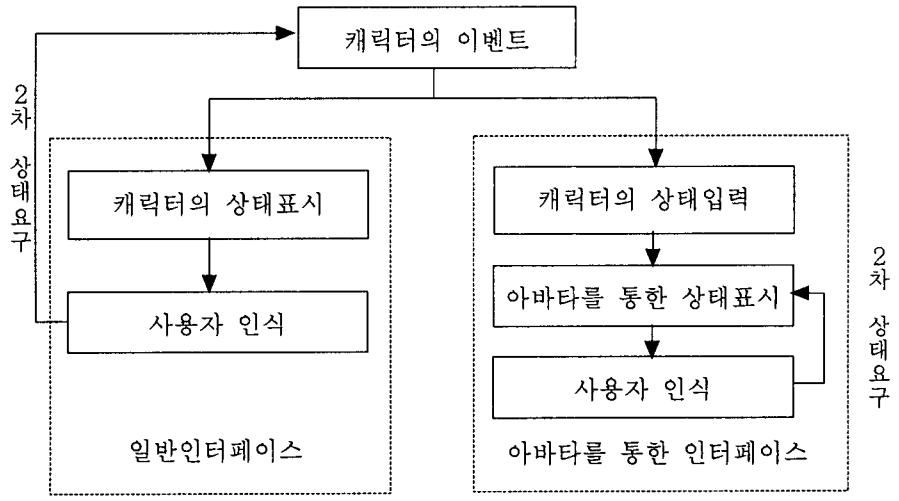

그림 6. 일반 인터페이스 방식과 아바타를 통한 인터페이스방식의 흐름도

```
stuct character {
 int X, Y; // 캐릭터의 x, y 좌표
 int Halfx, Halfy; //캐릭터 x, y 크기의 반을 나타내는 길이
};
struct character A;
struct character B;
  if(abs(A.X-B.X) < (A.Halfx+B.Halfx) &&
     abs(A,Y-B.Y) < (A.Halfy+B.Halfy))
// 충돌 처리루틴( ) ;
// *abs() 함수는 절대값을 의미하는 산술 함수이다.
// 아이템을 저장하기 위한 레코드는 다음과 같은 변수가 필요하다.
class HaveRec {
  public:
  int ID;
  char *title;
  HaveRec(int id, char *title) {ID=id; Title=title; }
};
```

캐릭터를 클래스화 하려면 표 3과 같은 공통점을 추출해야 한다.

표 3. 게임캐릭터의 공통점

2D 게임 캐릭터	3D 게임 캐릭터
캐릭터위치인 X,Y값을 가짐	캐릭터위치인 X, Y, Z 값을 가짐
한번에 움직일 dot 정보를 가짐	한번에 움직일 dot 정보를 가짐
이미지폰트 크기를 가짐	스케일링 값을 가짐
생성과 소멸의 정보를 가짐	생성과 소멸의 정보를 가짐
폰트를 담을 버퍼를 가짐	폰트를 담을 버퍼를 가짐

본 연구가 2D 게임 캐릭터의 3차원 표현이므로 2D 게임캐릭터의 공통점 즉 위치값과 이동, 크기의 변화에 대한 게임에서의 진행 중 행위의 공통점을 바탕으로 그들의 공통된 속성을 자바나 Java나 C++을 이용하여 다음과 같이 클래스화 한다.

```
class LOCATE { // 위치 정보 (x, y)클래스
  public;
  int X;
  int Y;
  public;
  void locate(int x, int y);
};
Class CHARACTER ; public LOCATE {
  public:
  void *characterfont; // 캐릭터 폰트 버퍼
  int Cfontx, Cfonty, Halfx, Halfy; // 캐릭터 폰트 크기
  int Type; // 캐릭터의 종류를 구별하는 ID
  char Isview; // 캐릭터의 출력 여부를 가리킴
  char Ishit; // 캐릭터가 다른 캐릭터부터 충돌했는지를 가리킴
  char Tab, Tabx, Taby; // 한 번에 이동할 수 있는 거리를
  Dot 수로 나타냄
};
```

캐릭터의 상태와 VRML 상의 아바타가 서로 상호작용하기 위해서는 그림7과 같이 캐릭터의 상태를 파일화 한다.

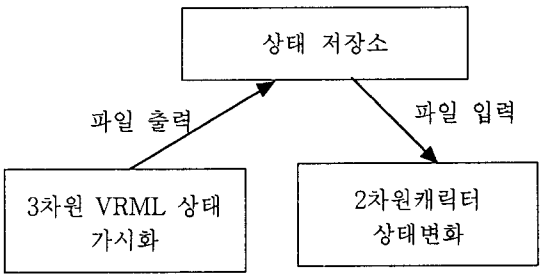

그림 7. 상태 입출력 경로

아래는 아바타의 상태 외부입력을 위해 게임 캐릭터의 변경된 상태를 파일 형식으로 저장하여 캐릭터의 상태를 아바타에 입력시키는 C++ 프로그램의 일부분이다.

```
void CPassData::PassCharInfo()
{
  CFile f;
  char tmp[80];
```

```
  char* pFileName = "charinfo.dat";
  // 파일을 연다.
  f.Open(pFileName, CFile::modeCreate
        CFile::modeWrite);
  // 변경된 캐릭터의 속성을 저장한다.
  sprintf(tmp, "체력 : %d\r\n스피드 : %d\r\n공격력 : %d\r\n
        마법 : %d\r\n", ARMY.nHealth, ARMY.nSpeed,
        ARMY.nAttack, ARMY.nmagic);
  // 캐릭터의 속성을 파일에 기록한다.
  f.Write(tmp,48);
  f.Close();
}
```

3.2 아바타의 상태 입력 모듈

그림 7에서와 같이 캐릭터의 상태가 파일화 되어 저장되어진 저장소에서 일단의 상태 파일을 입력받아야만 3차원 아바타는 상태를 디스플레이 한다. 일단의 디스플레이가 끝나게 되면 VRML의 뷰포인트를 조정하여 아바타의 상태를 입체적이고 직관적으로 살펴볼 수 있다. 그러나 브라우저 상황의 아바타를 직접적으로 제어 할 수는 없고 상태 저장소에서 들어오는 파일의 신호로만 상태를 제어할 수 있다 브라우저 상황에서는 아바타의 전후좌우와 아래, 위를 관찰 할 수 있게 된다.

VRML 파일의 외부 입력 부분은 다음과 같다.

```
if(expt? (me))
      grav (me)=0;
   if (zdrive (me)==0)
   {
    sturn=random(40);
    if (sturn==1)
    {..... }
   .....
   ......
   }
```

여기서 expt가 외부입력을 받아들이는 부분이다 expt의 외부 입력 값은 바로 상태저장소에 있는 게임 캐릭터의 상태파일이다.

VRML(Virtual Reality Markup Language)은 3차원 가상공간을 표시하는데 비해, 비교적 데이터 전송량이 적게 소요된다. 보통 동화상 파일은 작은 재생시간의 파일을 읽어 들이는데도 상당한 시간이 필요하지만 VRML은 3차원 공간의 형상을 기술한 텍스트 에디터를 전송하기 위해 시간적으로 유리하다. 또한 다른 그래픽 표현의 표현방법과 비교해서 인터넷과 결합이 잘 되는 장점[8]이 있어 본 연구에서는 VRML을 이용하였다.

4. 2차원 캐릭터와 3차원 가시화 구현

4.1 게임 캐릭터의 상태입력

본 연구에서는 롤플레잉이나 아케이드 게임상의 캐릭터 상태을 입력받기 위하여 아바타에 대한 클래스를 지정하였다. 첫 번째 클래스에서는 아바타의 가장 일반적인 속성을 정의하였다. 아바타의 객체는 각각의 독립 된 객체로 서로 다른 명시가 되어있다.

```
class 아바타_Element {
  private int 오른팔
  private int 왼팔
  private int 오른다리
  private int 왼다리
  private int 머리
  private int 가슴

  public int 오른팔_act(int x, int y, int z) {
        좌표축을 중심으로 오른팔에 대한 면에 해당
        ......
      }
}
```

다음은 클래스 아바타_Element의 자식에 해당하는 행동에 대한 클래스 아바타_Action extends 아바타_Element 의 정의를 내린 부분이다. 상태입력이 되었을 때 이동에 대한 정의이다. 그룹화

된 아바타를 형태의 변화 없이 이동 또는 행동할 수 있도록 하기 위한 것이다.

```
class 아바타_Action extends 아바타_Element {
  private int 오른쪽
  private int 왼쪽
  private int 앞쪽
  private int 뒷쪽
  public int //오른쪽_act(int x, int y, int z) {
        //좌표축을 중심으로 오른팔에 대한 면에 해당
          .....
    }
  public int 왼쪽_act(int x, int y, int z) {
        // 좌표축을 중심으로 오른팔에 대한 면에 해당
          .....
    }
  public int 앞쪽_act(int x, int y, int z) {
        //좌표축을 중심으로 오른팔에 대한 면에 해당
          .....
    }
 public int 뒷쪽_act(int x, int y, int z) {
        //좌표축을 중심으로 오른팔에 대한 면에 해당
          .....
   }
}
```

이렇게 정의된 클래스를 바탕으로 프로그램에 구현된 인터페이스는 그림 8의 Avatar State Controller이다.

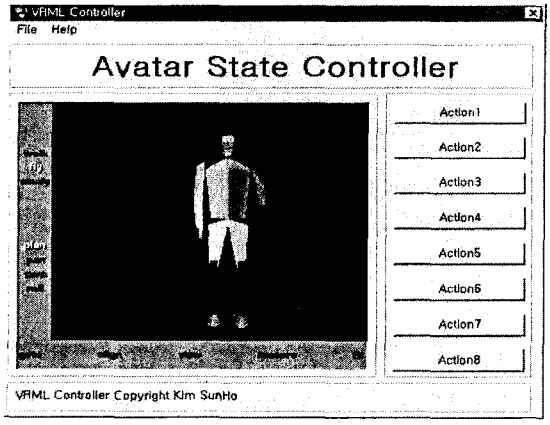

그림 8. Avatar State Controller 인터페이스

정의된 클래스들을 기초로 상태입력 즉 이벤트가 발생하였을 때를 Action()이라 정하고 VRML 파일에 Action()에 해당하는 값을 입력시키기 위한 함수부분은 다음과 같다.

```
class 아바타 {
  public static void(String args[]) {
    아바타_Element object_ele = new 아바타_Element();
    아바타_Action object_act = new 아바타_Action();
  switch(Event Action) {
   case Action1: //앞으로에 대한 이벤트일 경우
    아바타_Action.앞쪽_act(움직일 3차원 좌표값)
      System.out.println(결과값을 출력)
  break;
  case Action1:
   System.out.println( )
  break;
  case Action1:
   System.out.println( )
  break;
  case Action1:
   System.out.println( )
  break;
  case Action1:
   System.out.println( )
  break;
  case Action1:
   System.out.println( )
  break;
  default:
   System.out.println( )
  }
}
```

4.2 게임 캐릭터의 상태출력

캐릭터의 상태 입력으로 변경된 객체의 위치값은 각 객체의 위치정보에 입력되게 되고 그 결과물로 객체의 변형을 가져오게 된다. 이벤트인 Action 1에는 상태입력을 위하여 최종 출력된 파일의 내용에는 변형된 아바타의 팔에 대한 위치정보 (예: center 0.055 0.7 -0.13)가 입력되게 된다.

0.055는 X, 0.7은 Y, -0.13은 Z축의 변형된 좌표이다. 이 좌표를 VRML에 입력하기 위해서는 스크립트 노드를 사용할 수 있다. 앞에서 설명한 아바타 Action 클래스 코드를 호출하기 위해서 VRML 내에 다음과 같은 스크립 노드를 가지게 했다.

```
Script {
        url "클래스가 위치하고 있는 url 주소"
        eventin SFBool start
    }
```

위치한 클래스를 호출하여 파일의 속성을 정의하고 변형된 위치 값을 쓰게 되면 다음과 같은 형식으로 변화하게 된다.

이벤트 입력 전

```
normalPerVertex FALSE
            }
      }
      scale 0.11 0.9025 0.26
   }
}
translation 0 0.7258 0
   }
 Transform {
        children Transform {
      children Shape {
            appearance Appearance {
            material Material {
          }
      }
.....
normalPerVertex FALSE
                  }
          }
          scale 0.11 0.9025 0.26
   }
  translation 0.4902 0.7258 0
  }
 Transform {
     children Transform {
          children Shape {
               appearance Appearance {
               material Material {
               }
          }
```

화면에 나타나는 아바타의 상태는 그림 9 와 같다.

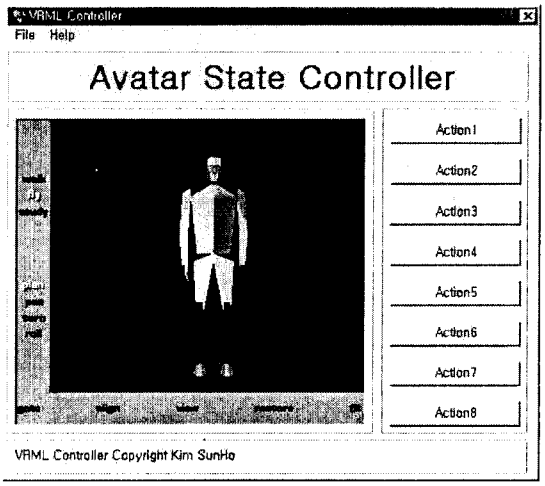

그림 9. 이벤트 입력전의 아바타 상태

이벤트 입력 후

```
normalPerVertex FALSE
                  }
            }
            scale 0.11 0.9025 0.26
      }
     }
     translation 0 0.7258 0
}
Transform {
     center 0.055 0.7 -0.13 [파일로 상태입력 된 부분]
    children Transform {
      children Shape {
            appearance Appearance {
            material Material {
            }
      }
.....
normalPerVertex FALSE
                  }
```

```
                    }
                    scale 0.11 0.9025 0.26
            }
            rotation 1 0 0 0.785398 [입력된 데이터에 의한
            위치 값이 변형된 된 부분]
            translation 0.7258 0
}
Transform {
      children Transform {
            children Shape {
                  appearance Appearance {
                        material Material {
                        }
                  }
```

결과로 화면에 나타나는 아바타의 상태는 그림 10과 같다.

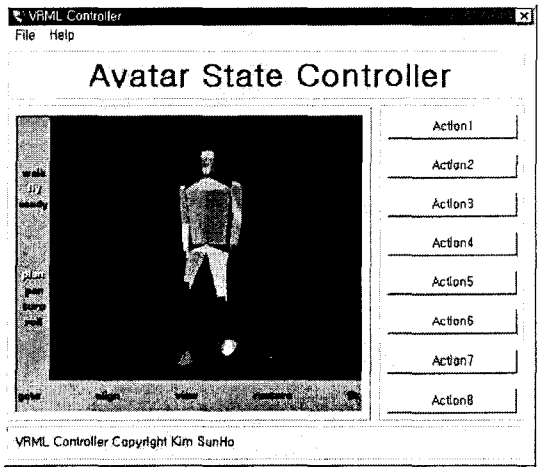

그림 10. 이벤트 입력 후 변화 상태

4.4 기존 게임 인터페이스와의 비교

일반 롤플레잉 게임이나 아케이드 게임의 캐릭터 상태의 디스플레이는 게임의 중간에 캐릭터의 상태를 파악하는 상태 창이 디스플레이 되고 그 수치를 가지고 사용자는 캐릭터의 상태를 알 수 있는 방식이다. 그러나 본 논문의 아바타를 이용한 캐릭터 상태의 3차원 표현 방법을 이용하여 캐릭터 상태를 디스플레이 하면 알고 있는 사실을 바탕으로 논리와 연상이 작용하여 의미를 파악할 수 있는 인지 개념의 디스플레이가 아닌 시각적인 유사점과 차이점을 통하여 상호작용을 직접적으로 표현하게 된다. 따라서 시간적인 연결과 운율적인 연결에 의한 사용자 연상작용인 직관적 디스플레이가 사용자의 인식률을 향상시켰다. 즉 각각의 텍스트 상태의 데이터를 시각화함으로써 안정적이고 정확한 상태인식이 가능하다. 따라서 상태를 보기 위해서 게임을 중단할 필요 없이 실시간으로 캐릭터의 상태를 알 수 있고 그 전의 상태를 알려고 할 때도 VRML의 특징인 가상공간의 항해를 통하여 캐릭터의 상태를 전후좌우로 돌아다니며 알 수 있다. 본 연구 결과로 향상된 점은 표 4와 같다

표 4. 게임진행 중 직관적인 상태표현 가능여부 비교

캐릭터의 상태	기존게임의 상태표현	아바타를 통한 상태표현
공격력	△	◉
방어력	◉	◉
아이템획득	△	◉
경험치	×	×
민첩성(스피드)	×	×
건강치	×	◉
레벨상승	△	△
지력	◉	◉
무기	△	◉

이러한 3D 가시화를 통하여 사용자에게 정보를 전달하게 되면 사용자의 게임에 대한 운영능력 습득이나 캐릭터 상태에 대한 기억이 필요 없이 게임에만 몰입하게 된다. 또한, 이러한 아바타들은 인터넷에서 바로 다운로드해서 캐릭터의 형태에 변화를 줄 수 있다.

5. 결 론

본 논문에서는 기존의 2차원 캐릭터의 게임 진행상의 상태표시의 각종 문제점을 분석하고, 개선된 정보전달 방법으로 VRML을 이용한 3차원 아바타를 게임캐릭터의 상태 아바타로 적용시키는 것을 제안하였다..

구현된 3차원 게임 캐릭터의 아바타는 기존의 텍스트 위주의 사용자와의 상태 정보 전달방법보다 정확하며 직관적으로 인식 될 수 있었다. 또한 VRML의 기술을 이용하여 캐릭터의 상태를 실시간 3D 렌더링을 할 수 있었다. 따라서 기존 게임의 정해진 움직임이 아닌 사용자의 임의 제어를 통하여 보여지므로 사용자들의 흥미를 향상시킬 수 있다. 제시된 VRML의 기술이 게임의 인터페이스에 도입됨으로써 사용자 인식도를 향상시킴은 물론 가상의 아바타 인터페이스를 재사용할 수 있도록 하여 아바타 캐릭터를 교체하는 것만으로도 새로운 환경을 만들 수 있게 하였다.

참 고 문 헌

[1] 박세근, "가상현실 갤러리에서 아바타의 이동에 따른 상호작용에 관한 연구", 한남대학교 석사학위논문, 1999

[2] H. Noser, "Navigation for Digital Actors Based on Synthetic Vision Memory and Learning", Computers and Graphics Vol.19,No.1, Pergamon Press, UK, 1995.

[3] M. Cavazza. R. Earnshaw, N. Magnment Thalmann, and D.Thalmann, "Motion Cotrol of Virtual Human", IEEE CG&A, Vol.18, No.5, Sep./Out. 1998, pp.24~31.

[4] 정문렬, "가상현실 시스템에서의 이벤트 핸들링 및 행위결정", 한국정보처리 학회지, 제5권 제2호, 1998.

[5] 홍석기 외 3인 공저, 게임 하듯이 게임 만들기, 가남사, 1997

[6] 김선호, 하수철, "VRML을 이용한 2차원 게임캐릭터의 3차원 가시화" 제14회 산학연 멀티미디어학술대회, 1998

[7] 김선호, "VRML을 이용한 2D 게임 캐릭터의 3D 아바타 표현에 관한 연구", 대전대학교 대학원 석사학위 논문, 2000. 8.

[8] Ed. Tittel 외 3명, 최영란 역, "Inside Secrets VRML", 도서출판 삼각형,1998

[9] Jon De Goes, 고경희, "C++ 3D게임 프로그래밍", 도서출판 혜지원, 1997년 6월

[10] VRML, The virtual Reality Modeling Language 97 Specification, http://www.vrml.org/specifications, VRML 97, 1997

* 본 연구는 교수 연구 년 기간에 수행되었음.

김 선 호

- 1996년 한남대학교 응용미술과(학사)
- 2000년 대전대학교 컴퓨터공학과(석사)
- 1996년~1998년 맥스포기획 CG팀
- 1998~현재 한남대학교 멀티미디어학부 조교
- 1999~현재 멀티미디어 컨텐츠·기술센터 연구원
- 관심분야 : VRML, 게임컨텐트, 컴퓨터그래픽스, 영상처리

하 수 철

- 1981년 홍익대학교 컴퓨터공학과 졸업(학사)
- 1986년 홍익대학교 대학원 컴퓨터공학과(석사)
- 1990년 홍익대학교 대학원 컴퓨터공학과(박사)
- 2000년~현재 한국디지털컨텐츠학회 이사
- 2000년~현재 한국컴퓨터산업교육학회 이사
- 1999년~현재 멀티미디어 컨텐츠·기술센터 소장
- 1999년 한국정보처리학회 논문지 편집위원
- 1998년~현재 한국정보처리학회 멀티미디어시스템연구회 부위원장
- 1997년~현재 소프트웨어연구센터(SOREC) 운영위원
- 1996년 한국전자통신연구원 초빙연구원
- 1991년~1992년 플로리다 주립대학교/텍사스 주립대학교 객원교수
- 1981년~1984년 *Army Logistics Command.* System Analyst
- 1987년~현재 대전대학교 컴퓨터공학과 정교수
- 관심분야 : 소프트웨어공학(객체지향화), 멀티미디어응용, 게임공학, 시각언어, 프로토콜기술언어