재사용성 및 용이성을 위한 계층적 아바타 행위 스크립트 언어의 정의

(Definition and Application of a Layered Avatar Behavior Script Language for Reusability and Simplicity)

김 재 경 [†] 최 승 혁 [†] 손 원 성 [††] 임 순 범 [†††] 최 윤 철 [††††]
(Jaekyung Kim) (Seunghyuk Choi) (Wonsung Sohn) (Soonbum Lim) (Yoonchul Choy)

요 약 아바타 스크립트 언어는 사이버 공간에서 아바타의 동작을 제어하기 위해서 사용되는 명령어들로 이루어져있다. 사용자가 아바타 동작 스크립트 언어를 쉽게 작성하기 위해서는 스크립트 언어의 행위 표현이 복잡한 하위수준 동작 표현 요소로부터 최대한 추상화되어야 한다. 또한 작성된 시나리오 스크립트가 다양한 구현환경에 적용 될 수 있는 표준적인 구조를 가지고 있어야 한다. 이를 위해 본 논문에서는 작업수준 행위, 상위수준 동작 및 기본 동작 스크립트 언어로 구성된 계층적 아바타 행위 표현 언어를 정의하였다. 각 계층의 언어는 사용자의 스크립트 작성을 위한 행위 요소와 아바타 동작 시퀀스 및 구현 환경의 기하 정보를 분리하여 표현한다. 제안 언어를 통하여 사용자는 추상화된 스크립트 작성이 가능하며 작성된 스크립트는 번역 과정을 거쳐 다양한 구현 환경에 적용이 가능하다. 구현 결과에서는 제안 기법을 적용한 시스템을 구축하여 사이버 교육 도메인 환경에서 아바타 스크립트 시나리오의 작성 과정 및 스크립트가 다양한 응용 프로그램에 유연하게 적용되는 것을 보여준다.

키워드 : 아바타, 스크립트 언어, 행위 제어, XML

Abstract An avatar script language consists of commands set which is used to control avatar behaviors in cyberspace. The script language should be abstract from complex low-level concepts, so that a user can write down a scenario script easily without concerning about physical motion parameters. Also, the script should be defined in a standard format and structure to allow reusing in various implementation tools. In this paper, a layered script language is proposed for avatar behavior representation and control, which consists of task-level behavior, high-level motion and primitive motion script language. The script language of each layer represents behavior elements for a scenario scripting interface, an avatar motion sequence, and geometric information of implementation environment, respectively. Therefore, a user can create a scenario script by abstract behavior interface and a script can be applied to various implementations by the proposed translating process. A presentation domain is chosen for applying the proposed script language and the implementation result shows that the script is flexibly applied in several applications

Key words : Avatar, Script language, Behavior control, XML

· 본 연구는 한국과학재단 목적기초연구(R01-2004-000-10117-0 (2004))지원으로 수행되었음
† 학생회원 : 연세대학교 컴퓨터과학과
ki187cm@gmail.com
alienart@rainbow.yonsei.ac.kr
†† 정 회 원 : 경인교육대학교 컴퓨터학과 교수
sohnws@gmail.com
††† 종신회원 : 숙명여자대학교 컴퓨터과학과 교수
sblim@sookmyung.ac.kr
†††† 종신회원 : 연세대학교 컴퓨터과학과 교수
ycchoy@rainbow.yonsei.ac.kr
논문접수 : 2005년 6월 7일
심사완료 : 2006년 4월 5일

1. 서 론

사용자를 가상환경에서 표현해주는 아바타(Avatar)는 이제 평범한 사이버 공간에서 누구나 접할 수 있는 존재이다[1]. 3D 가상환경의 아바타 뿐만 아니라 MSAgent의 등장으로 웹 환경에서도 누구나 쉽게 아바타를 사용할 수 있고, 최근에는 모바일 기기의 발달로 핸드폰 및 PDA에서도 아바타를 접할 수 있다[2].

아바타의 활용에 있어서 가장 중요한 것은 역시 아바타의 애니메이션, 즉 행위다. 아바타의 행위를 통하여 우리는 아바타에 대해 친근감(Friendliness)과 사실감

(Believability)[3]을 느끼며 여러 정보를 이야기(Story-telling)[4] 형식으로 전달 받을 수 있다. 즉 어떤 정보를 전달하는데 있어서 단순히 문장이나 음성 출력만을 이용하는 것보다 아바타와 같은 사람 형태의 캐릭터가 다양한 동작과 더불어 설명을 하는 것이 훨씬 효과적이다. 이러한 효과를 'Persona Effect[5]'라고 하는데 특히 사이버 교육 분야에서 아바타의 행위를 통하여 높은 효과를 기대할 수 있다.

이와 같은 아바타를 활용한 이야기 형식의 콘텐츠 제작은 일반적으로 크게 3가지 방식으로 이루어 질 수 있다. 첫째, 3DS Max와 같은 애니메이션 소프트웨어를 이용하여 저수준의 아바타 각 관절을 조작하여 모든 동작을 생성하는 방법, 둘째, 아바타에게 인공지능(AI)을 적용하여 미리 정의된 동작 시나리오에 따라 자동으로 시나리오를 생성하는 방법[6], 그리고 셋째 저작자가 동작 제어 스크립트를 이용하여 아바타의 시나리오를 직접 작성하는 방법이다[7,13,15].

기존의 스크립트를 이용한 아바타 시나리오 작성의 경우, 아바타를 제어하기 위해 동작, 위치, 속도 및 시간 동기화 등과 같은 여러 요소들을 고려하다 보니 스크립트가 복잡해지고 길이가 길어지는 특징이 있다. 따라서 보다 용이한 시나리오 작성을 위해 스크립트의 추상화를 통하여 사용자에게 간단한 스크립트 작성 인터페이스를 제공하는 것이 필요하다.

본 논문에서는 사용자가 아바타 시나리오를 보다 적은 노력과 시간으로 작성할 수 있도록 특정 도메인 환경에서 사용될 수 있는 작업수준의 행위 스크립트 언어를 제안한다. 작업수준의 행위란 특정 목표를 달성하기 위한 아바타의 행위나 혹은 특정 대상에 대한 아바타의 행위이다. 제안 스크립트는 특정 도메인 환경에서 각각 사용되는 행위 집합을 정할 수 있으며 사용자는 해당 도메인의 행위 집합을 이용하여 시나리오 스크립트를 작성한다.

또한 제안 스크립트 언어는 재사용성을 위하여 계층적인 구조를 가지고 있는데 작업수준 행위, 상위수준 동작, 기본 동작의 총 3개 계층을 가진다. 사용자는 현재 도메인 환경에서 정의된 작업수준 행위에서 시나리오를 작성하게 되며, 이는 제안된 번역기에 의해 상위수준 동작 스크립트로 표현된다. 상위수준 동작 스크립트는 아바타의 동작을 도메인이나 구현 환경에 독립적인 요소로 표현하고 있으며 사용자가 작성한 시나리오 스크립트와 구현환경을 연결하는 표준적인 다리 역할을 한다. 마지막으로 기본 동작은 각 구현 환경의 기하정보 및 아바타 엔진이 지원하는 애니메이션 정보를 표현할 수 있으며, 상위수준 동작 스크립트의 요소들을 적절한 기본 동작으로 변환하여 아바타의 저수준 애니메이션 데이타를 호출하고 제어한다. 제안기법의 적용을 통하여 스크립트 시나리오 작성의 용이성과 재사용성을 실험하기 위하여 본 연구에서는 사이버 교육 도메인을 선정하여 이에 맞는 계층적인 아바타 행위 스크립트(Layered Avatar Behavior Script)을 정의하였으며, 시나리오의 실행을 위한 구현 환경으로는 컴퓨터 교과목을 위한 교과 과정을 설계하여 아바타를 적용하여 보았다.

본 논문은 다음과 같이 구성되어 있다. 2절에서는 기존 관련연구의 특징 및 결과에 대해 논하고, 3절에서는 제안 언어인 계층적 아바타 행위 스크립트에 대해 설명한다. 4절에서는 제안 기법을 적용한 아바타 시스템의 구현 결과에 대해 논한다. 5절에서는 제안 기법의 사용자 평가와 마지막으로 7절에서 결론 및 향후 연구에 대해서 언급하도록 한다.

2. 관련 연구

아바타 행위의 표현 및 제어에 관련된 연구는 중요한 주제이며 다양한 방식의 연구가 이루어져 왔다. 이들 연구를 아바타의 행위 표현 수준을 기준으로 하위수준 및 상위수준의 동작 표현 방식으로 나누어 볼 수 있다.

하위수준은 아바타 애니메이션을 위한 가장 기초적인 방법으로 아바타의 신체를 관절(Joint)와 신체부위(Segment)의 쌍으로 보고 각 쌍에 대해 회전 및 이동과 같은 DOF 값을 순차적인 시간흐름에 따라 적용하는 것으로, 3DS Max나 Maya와 같은 캐릭터 애니메이션 소프트웨어에서 주로 사용되고 있다. 이와 같은 방식은 사용자가 자유롭게 행위를 생성할 수 있고 행위의 종류나 개수에도 별다른 제약이 없다. 사용자가 애니메이션 기술(Skill)을 가지고 있거나 모션 캡쳐 데이타[8]를 사용하면 아주 자연스러운 행위의 사용이 가능하다. 그러나 이러한 세부적인 조작으로 인해서 간단한 아바타 애니메이션 시나리오 작성을 위해서 많은 시간과 비용이 들며, 작성된 애니메이션 시나리오는 대부분 특정 아바타의 신체구조 및 자체포맷을 사용하기 때문에 여러 환경에서 재사용되기 힘들다. 따라서 하위수준의 아바타 제어는 영화와 같이 자연스럽고 정형화되지 않은 동작을 요구하는 환경에서 주로 쓰이고 있다.

상위수준의 표현 방식은 아바타의 관절에 대한 DOF 값 등을 고려하지 않고 행위에 대한 추상적인 명령이나 요소들로 이루어진다. 상위수준 표현 방식 중 작업기반의 제어 방식은 로봇에게 추상적인 최종 결과를 할당하여 인공지능에 의해 동작을 생성하는 방식이다. 이 방식은 과거의 로보틱스 분야나 최근의 지능형 에이전트 분야에서도 많은 연구가 이루어지고 있다[19,20]. 로보틱스 공학의 전형적인 유형인 미로 찾기 시스템이나 부품 어셈블리 시스템과 같은 환경에서 사용자는 로봇에게 최

종 목표인 목적지나 조립 및 분해의 작업을 명령하면 로봇은 AI 엔진에 의해 주어진 작업에 필요한 하위수준의 동작들을 생성 및 조합하고 적절한 순서로 동작을 수행하여 최종 목표를 달성한다[19,20]. 이러한 개념은 최근 아바타에도 적용되어 사용자가 원하는 주제에 대해 자동으로 도움말 시나리오를 생성 해주거나[9], 가상환경을 구성하고 있는 정보를 인지하여 마치 살아있는 듯(Life-like) 주변 환경에 지능적으로 반응하는 경우도 있다[7].

이러한 AI 기반의 시스템은 특정 작업환경이나 간단한 시나리오를 재생하여 보여주기에는 적당하지만 작업이 복잡해질 때 아직 완전한 해결책을 제시하지 못하고 있으며 시스템 설계 단계에 미리 정의된 시나리오만을 보여주기 때문에 스크립트 저작자가 시나리오를 수정하거나 추가하는 것이 힘들다.

상위수준의 또 다른 표현 방식으로 최근에는 텍스트 형식의 스크립트를 작성하여 아바타를 제어하기 위한 상위수준의 스크립트 언어에 대한 연구도 활발히 진행되고 있다. 스크립트 언어들은 모두 아바타의 행위 표현 및 제어를 목표로 하지만 그 특성이 다양하다. TVML [17]의 경우 방송 스튜디오의 구성 및 방송에서 필요한 명령어들로 이루어진 스크립트로, 간단한 뉴스 등을 사용자가 손쉽게 작성할 수 있도록 하고 있다. 또한 CPSL[16]의 경우 사이버 교육 도우미를 이용한 시나리오 스크립트를 작성할 수 있는 언어이다. 그러나 언어의 포맷이 표준 형식이 아니라 자체 소프트웨어에서만 인식 가능하여 스크립트의 재사용에 한계가 있다.

영어 문장의 문법을 가진 스크립트를 통하여 3D 가상환경에서 아바타 애니메이션을 제어하거나[10], 다이나믹 로직 프로그래밍(DLP) 기법[18]을 이용하여 복잡한 행위를 조합하여 표현하기도 한다. 전자의 경우 역시 표준 포맷을 사용하지 않았으며 후자의 경우 스크립트를 통해 아바타의 관절의 이동 및 회전을 제어하기 때문에 동작의 표현 수준이 낮아 스크립트가 복잡하고 많은 양의 시나리오를 작성해야 한다.

AML[13]은 다양한 요소를 이용하여 행위를 표현하고 있으며 CML[14]은 아바타 동작을 의미적인 마크업(Markup)으로 정의하여 표현하고 있다. AML은 시간 동기화나 강도(Intensity)와 같은 복잡한 요소로 인하여 스크립트 작성자가 직접 작성하기에는 어려움이 있으며 CML은 주로 아바타의 감정 및 성격에 따른 제스쳐(Gesture) 변화에 중점을 두고 있다.

이 외에도 얼굴표정, 몸동작, 음성, 감정, 제스쳐 등의 아바타 행위를 복합적으로 표현하고 있는 VHML[15], 아바타의 감정에 따른 음성 표현에 중점을 둔 MPML 등이 있다.

기존 스크립트 언어들은 대부분 사용자가 용이하게 작성할 수 있는 행위 표현 보다는 감정표현, 시간 동기화, 얼굴 동작에 중점을 두고 있으며, 상위 수준의 스크립트가 실제 구현 환경에서 구체적으로 어떤 동작으로 구성될 것인지에 대한 명확한 구조는 다루고 있지 않다.

또한 앞서 언급한 계층적인 행위 표현에 대한 기본 개념은 기존 애니메이션 시스템에서 자주 사용하고 있는 구조이다. 스크립트 관련 연구의 기반이 되는 Improv [11]는 행위, 애니메이션 및 기하 엔진의 3가지 엔진을 제공하고 있는데 행위 엔진에서는 아바타의 성향이나 무드에 따른 행위 결정, 애니메이션 엔진에서는 자연스러운 동작 생성, 기하 엔진에서는 관절의 회전 값을 표현하고 있다. 그러나 Improv는 자체 스크립트 포맷과 하위수준의 애니메이션 스크립트를 사용함으로써 재사용성에 한계가 있다. 이 밖에도 HAC[12]과 같은 연구는 다수의 아바타간의 상호작용을 위해 계층적인 행위 구조를 택하고 있다. 이와 같이 목적에 따라 계층적 행위 표현 개념이 효과적으로 이용될 수 있으나, 아직 표준 포맷을 이용하여 계층 구조를 스크립트 형식으로 명확하게 분리하여 표현한 연구가 부족한 실정이다.

3. 계층적인 아바타 행위 스크립트

본 논문에서는 스크립트 작성의 용이성과 작성된 스크립트의 재사용을 위해 계층적인 스크립트 행위 표현 및 제어 구조를 정의하였다. 이를 위해 우리는 사람과 기계가 모두 쉽게 이해할 수 있고 특정 시스템에 종속적이지 않은 XML 기반의 아바타 행위 스크립트 언어를 정의하였다. 아바타 행위 스크립트 언어는 총 3계층으로 이루어져 있으며 이들은 작업수준 행위 스크립트, 상위수준 동작 스크립트, 그리고 기본 동작 스크립트로 구성되어 있다. 제안 기법의 개념적인 전체 구조는 그림 1에 나타나 있다.

이러한 구조의 가장 큰 이점은 첫째, 최상위 수준에서는 사용자가 하위수준의 데이타를 신경 쓰지 않고 추상적인 인터페이스를 이용하여 쉽게 스크립트 작성을 할 수 있다는 것이고, 둘째로 사용자가 작성한 시나리오 스크립트와 구현 환경에서 사용되는 하위수준의 스크립트를 완전히 분리하여 시나리오가 여러 응용 프로그램에서 재사용 될 수 있다는 것이다. 이제 각 계층의 스크립트에 대해 자세히 살펴보도록 한다.

3.1 작업수준 행위 스크립트 언어

가장 상위 계층인 작업수준 행위 스크립트는 활용 도메인 별로 요구되는 행위들을 표현한다. 아바타의 일반적인 행위는 매우 다양하고 그 수도 많지만, 특정 도메인에서 사용되는 행위는 제한적이다. 따라서 사용자는 아바타의 모든 행위를 고려할 필요 없이 도메인 별로 제공

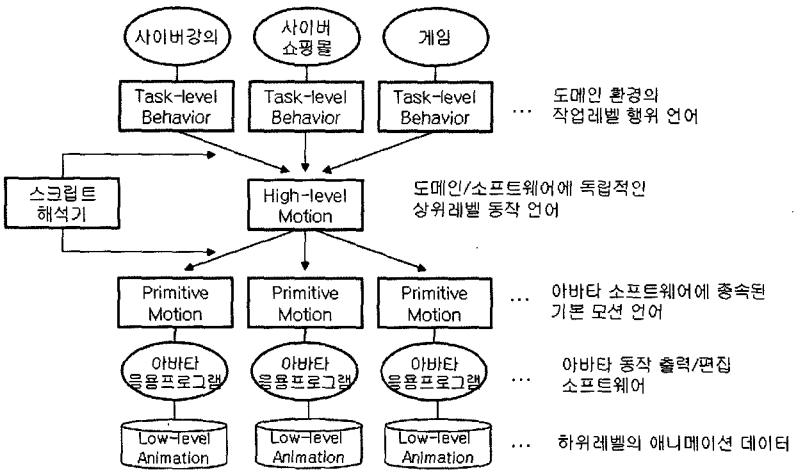


그림 1 제안 기법의 개념적인 구조

되는 작업수준 행위 집합을 이용하여 아바타 행위 제어 시나리오를 작성한다. 또한 작업수준 행위는 하위수준 애니메이션 데이타로부터 완전히 추상화된 표현으로, 사용자는 아바타의 구체적인 동작이나 기하정보를 입력하는 것이 아니라 '설명하라', '제목에 주석을 달아라'와 같은 식으로 작업위주의 명령으로 시나리오를 작성한다.

제안하는 작업수준 행위 스크립트(TBS:Task-level Behavior Script)는 특정 도메인 환경에서 이루어지는 작업들을 수행하기 위한 행위로 구성되어 있다. 즉, 구체적인 아바타의 동작을 표현 하는 것이 아니라 추상적인 작업 명령을 아바타에게 할당함으로 보다 상위 수준에서 사용자가 아바타를 제어하도록 한다. 시나리오를 작성하는 사용자에게 아바타가 '어떻게(How)' 동작을 할 것인가는 보다는 '무엇을(What)' 할 것인가를 표현해 주는 것이다.

아바타가 무엇을 할 것인가는 아바타가 속해있는 도메인 객체들의 구성에 따라 달라진다. 여기서 도메인 객체란 어떤 도메인에서 아바타가 도메인을 구성하고 있는 가상 객체들을 의미하며, 작업은 도메인 객체들과 상호작용을 통하여 이루어지는 작업들을 의미한다. 따라서 작업을 수행하기 위한 다수의 작업수준 행위와 특정 도메인의 객체들이 하나의 도메인 행위 집합으로 구성되어 사용자에게 제공된다. 사용자는 이 행위 집합을 이용하여 작업수준의 시나리오를 작성하게 된다.

3.1.1 작업수준 행위 스크립트 언어의 구조

작업수준 행위 스크립트는 사용자가 직접 작성하는 부분으로 아바타가 도메인 환경을 구성하고 있는 구성요소들과 상호작용하며 수행하여야 할 일련의 작업수준 행위(Behavior)들로 이루어진다. 따라서 행위는 가상 객체 혹은 가상 공간에서의 위치와 같은 도메인의 구성요소들을 행위의 대상으로 한다. 그리고 행위의 병렬 및 순차적인 수행을 위한 동기화 요소(Synchronization)를 포함되며, 또한 강도 및 속도와 같은 행위 제어 요소(ControlElement)들도 지원하고 있다.

ScenarioScript = Synchronization(Behaviors (Objects), ControlElement)

제안 언어는 스크립트 작성의 용이성을 위해 아바타의 행위제어를 위한 최소한의 요소로 표현 하고 있는데, 이는 아바타의 모든 행위를 표현하고자 함이 아니라 특정 도메인 내에서 사용되는 특정 행위를 표현하기 위한 것이다. 만약 모든 행위를 표현하고자 한다면 수많은 경우의 행위에 대한 다양하고 가변적인 요소를 기술해야 한다. 그러나 특정 도메인 환경 내에서 사용되는 유한한 행위에 대해서는 미리 행위 집합을 정해주고 이에 필요한 형식적인 요소들을 이용하는 것이 가능하다.

제안된 행위 표현은 모든 도메인에서 사용되는 아바타의 행위 제어를 위한 것이 아니라 특정 환경에서의 추상화된 행위 표현을 위한 것이다. 예를 들어 애니메이션 영화등과 같이 동작의 사실성을 중요하게 다루는 환경보다는, 아바타의 행위의 형태가 비교적 고정적이고 행위를 시나리오에 따라 순차적으로 수행하는 사이버 교육 혹은 가상 투어와 같은 환경에서 제안 기법의 적용을 목적으로 하고 있다.

이를 위해 본 논문에서는 사용되는 행위의 종류가 비교적 고정적이고 반복되며, 작성된 시나리오에 따라 아바타의 행위가 순차적으로 수행되는 교육 환경을 선정하고, 아바타를 이용한 발표 혹은 강의 진행을 위한 작업수준 행위 표현 스크립트를 정의하였다.

제안하는 작업수준 행위 스크립트는 그림 2와 같은 구조의 XML DTD로 정의하여 사용자가 XML을 이용하여 구조적이고 명확한 스크립트를 작성할 수 있도록 하였다.

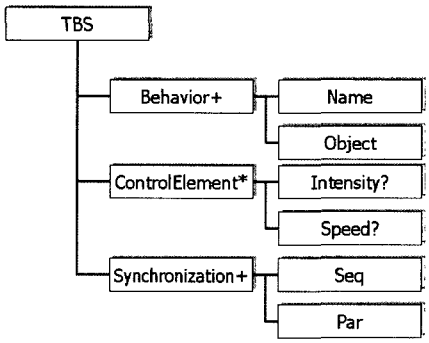


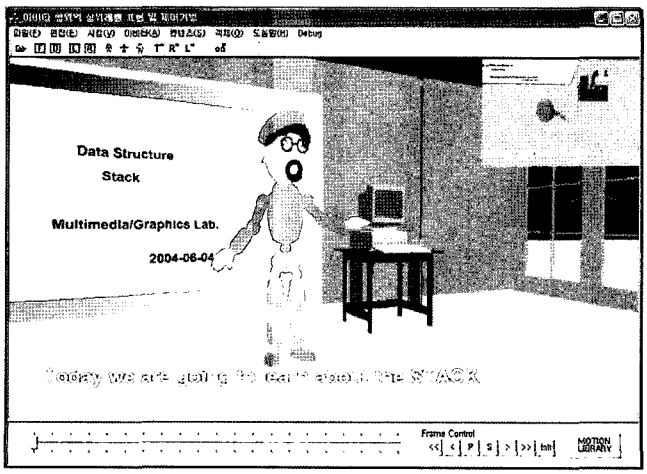
그림 2 스크립트 시나리오의 DTD 구조

작업수준 행위는 가상 환경에 존재하는 어떤 대상에 대해 이루어진다. 왜냐하면 대상이나 목적 없이 단순히 움직임만을 표현하는 것은 의미 없는 동작이지만, 목적을 가지고 어떤 대상 혹은 위치와 상호작용 하는 것은 작업의 의미를 가지기 때문이다. 'point'이라는 동작은 손을 뻗고 손가락을 어딘가로 가리키는 '동작' 그 자체를 뜻하지만 'point at the box'는 상자라는 객체를 향해 손을 가리키는 목적을 가지고 있는 '행위'이다. 따라서 작업수준 행위는 행위와 행위의 목적이 대는 대상 객체의 조합으로 이루어진다.

제안 언어는 교육 환경에서 사용되는 행위를 사용자에게 제공하고 있는데, 행위 시나리오에서 사용 가능한 행위는 도메인 행위 집합에서 정의된 행위들로 구성되며 행위의 대상이 되는 객체들은 도메인 객체에서 정의된 객체들을 사용한다. 예를 들어 사이버 강의실에서 아바타가 화면을 가리키며 설명하는 행위는 다음 그림 3

```
<?xml version="1.0" encoding="UTF-8"?>
<TBS>
   <Sync type="seq">
      <Behavior name="greet" object="student"/>
      <Behavior name="point" object="screen"/>
      <ControlElement speed="fast" intensity="strong">
      <Behavior name="talk" object="student">Today we are going to learn...</Behavior>
      </ControlElement>
      ...omit...
      <Behavior name="write" object="screen"/>
      <Behavior name="talk" object="student">The answer is...</Behavior>
   </Sync>
 </TBS>
```

(a) Presentation 도메인에서의 Task-level Behavior Scenario Script의 예



(b) Presentation 도메인 환경에서 스크립트에 의해 동작하는 아바타

그림 3 아바타 시나리오 스크립트의 예 및 실행 화면

과 같이 작성된다.

다음으로 행위 시나리오에서 아바타의 행위는 하위수준의 동작을 표현하는 것이 아니라 추상적인 작업 수준의 행위이기 때문에 보통 순차적으로 이루어진다. 그러나 때로는 동시에 작업을 수행하는 경우도 있는데, 예를 들어 강사 역할을 하는 아바타가 칠판에 필기를 하며 동시에 어떤 대상에 대해 학생들에게 설명을 하는 작업 등이다. 이를 위해 행위 시나리오는 행위의 <Sync> 태그를 통하여 병렬 및 순차 수행을 표현한다.

또한 부가적으로 사용자가 아바타의 특정 행위들을 강조하거나 속도를 제어할 수 있도록 몇 가지 기본적인 태그들을 지원하고 있다. <Speed> 태그는 행위의 속도를 'slowest, slower, slow, normal, fast, faster, fastest'의 7단계로 제어할 수 있으며 <Intensity> 태그도 'weakest, weaker, weak, normal, strong, stronger, strongest'의 7단계로 행위를 강조하거나 약하게 변화시킨다. 작업수준 행위에서는 사용자가 대략적인(Coarse) 행위 제어 요소를 지정하고 실제 어느 정도의 기하학적인 값(Value)으로 행위를 제어할 것인지는 구현환경의 특성에 따라 하위 수준에서 결정하게 된다.

이와 같은 개념을 표현하기 위해서 제안하는 작업수준 행위 스크립트는 XML DTD 형식으로 구성되어 있으며 전체 DTD는 [부록A]에 첨부하였다

3.1.2 도메인 객체 모델

제안 언어는 도메인에 따라서 다른 종류의 행위 집합을 제공하는데, 앞서 언급하였듯이 작업수준 행위는 도메인을 구성하고 있는 객체와의 상호작용을 통하여 이루어 지므로 객체와 아바타간의 행위들의 집합이 도메인에서 사용되는 작업수준 행위의 집합으로 정의된다. 즉, 도메인을 구성하는 임의의 객체가 k개의 아바타-객체 행위를 제공한다면 n개의 객체로 이루어진 도메인 환경은 n*k개의 행위로 이루어진 작업수준 행위 집합을 가지게 되는 것이다.

또한 작업수준 행위는 아주 추상화된 요소만으로 행위를 표현하고 있으므로 다음 단계인 상위수준 동작 스크립트로 행위를 변환하기 위해서는 이를 동작 순서로 분화시키고 필요한 요소들을 생성시켜주는 정보가 필요하다.

이와 같은 정보를 표현하기 위해 도메인 객체 모델을 정의하였는데 이는 사용자에게 도메인의 행위집합을 제공하고 작업수준 행위 스크립트 번역기에서 상위수준 동작 스크립트를 생성하기 위한 정보로서 이용된다.

먼저 도메인 객체 모델은 XML 형식으로 정의된 행위 스크립트와의 호환성과 특정 환경에 종속되지 않도록 하기 위해 XML 구조(그림 4)로 정의되었으며 XML DTD는 부록D에 첨부하였다. 최상의 요소인 'Object-List'는 도메인을 구성하는 Object 요소들로 구성되어 있다. 즉 객체 리스트는 현재 도메인 환경에서 사용되는 객체들의 명단을 모아놓은 것이다.

'Object' 요소는 크게 'ControlPoint' 및 'Behaviors' 요소로 이루어지는데 각 요소의 특징은 다음과 같다.

1) ControlPoint 요소

'ControlPoint'는 'position', 'direction' 그리고 'part'로 구성되어 있다. 'Position'은 아바타가 객체와 행위 상호작용을 취할 때 객체의 주변 어느 곳에 위치할 것인지를 상대적으로 표현하며, 'Direction'은 아바타의 방향을 지정하며 'part'는 조작이 행해지는 객체의 특정 부분을 표현한다.

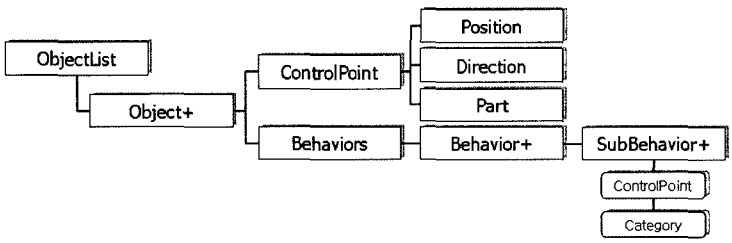
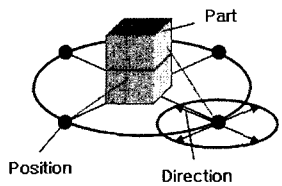


그림 4 도메인 객체 모델의 구성요소



Element	Values
Position	front, behind, left, right, center, $user\_defined_n$
Direction	object, backward, forward, left, right, $user\_defined_n$
Part	$part_1$ ... $part_n$

그림 5 'ControlPoint'의 구성요소

2) Behaviors 요소

'Behaviors'는 작업수준 행위를 수행하기 위한 'Sub-Behavior'들로 구성되어 있다. 하나의 작업수준 행위는 하나 이상의 'Sub-Behavior'들로 수행이 된다. 각각의 'Sub-Behavior'들은 'ControlPoint'를 인자로 받아 대응되는 객체의 위치에서 각 동작을 수행할 것을 지정한다. 또한 'Category' 인자를 통하여 해당 행위가 'Navigation', 'Manipulation' 혹은 'Gesture' 분류인지 지정하며 이는 상위수준 동작 스크립트로 변환될 때 해당 분류에 다른 요소 집합을 생성하는 기준이 된다.

다음 그림 6은 제안 기법의 적용환경인 교육 도메인에서 쓰이는 기본 객체를 객체 정보 모델을 이용하여 정의한 예이다.

교육 도메인을 구성하는 객체와 행위 및 요소들은 각 'Object' 요소 하위의 'ControlPoint' 및 'Behaviors' 요소에 정의되어 해당 도메인에서 필요한 행위들을 제공하게 된다. 따라서 각 도메인마다 고유한 객체 및 행위 집합을 가지고 있으며 이는 도메인 설계자 혹은 디자이너가 정의하여 사용자에게 제공한다. 본 논문에서는 제안 기법의 프로토타입 구현을 위해 같이 교육 도메인에서 사용될 수 있는 다음 그림 7과 같은 객체들과 연관된 작업수준 행위로 이루어진 집합을 정의하여 사용하였다.

정의된 객체 모델에 따라 다음 그림 8과 같이 도메인 환경을 구성하고 아바타는 객체들과 지정된 작업수준 행위로 상호작용하는 것을 볼 수 있다.

Domain Object	Related Task-level Behavior
Avatar	Greet, Introduce, Bye, Talk
Computer	Next_page, Prev_page
Text	Annotate, Highlight, Point
Screen	Write, Erase, Point
Box	Get, Put

그림 7 교육 도메인의 객체 모델 예

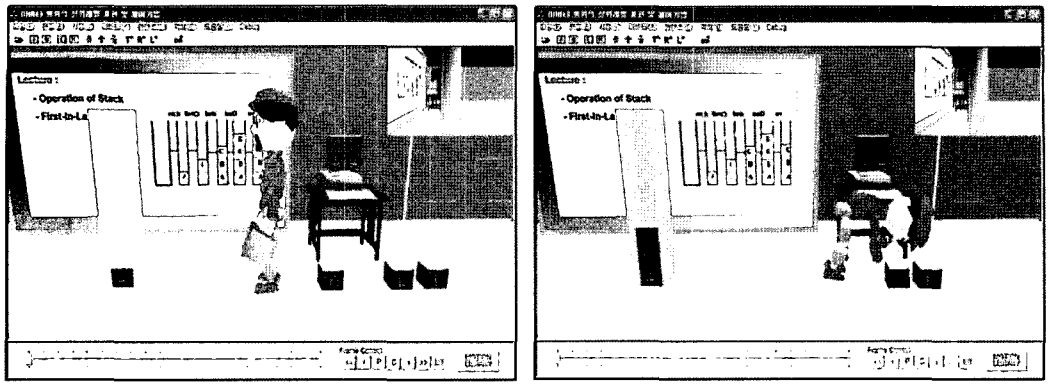
또한 객체 정보는 필요에 따라 'Object' 요소를 추가하고 하위 요소들을 정의함으로써 사용자가 확장 및 수정 할 수 있다.

3.2 상위수준 동작 스크립트 언어

앞서 언급한 작업수준 행위 스크립트는 도메인 환경에 종속적인 행위와 객체정보로 이루어져 있으며 다음에 설명할 하위수준인 기본 동작 스크립트는 구현 환경에 종속적인 물리적인 정보들을 표현한다. 따라서 이들 두 스크립트의 중간에서 교량역할을 하는 중간 단계의 스크립트가 필요한데 이를 상위수준 동작 스크립트라 정의하였다. 즉, 사용자가 작성하는 작업수준 행위 스크립트는 아바타가 무엇을 해야 할 것인가, 즉 수행 해야 할 작업을 표현하고 있으며 실제 작업 수행을 위해서 아바타가 어떤 동작들을 취해야 하는지는 표현하고 있지 않다. 앞서 언급하였듯이 하나의 작업 수행을 위해 필요한 많은 동작들을 사용자가 일일이 고려하지 않도

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ObjectList SYSTEM "domaininfo.dtd">
<ObjectList domain="presentation">
    <Object name="text">
        <ControlPoint name="p1">
            <Position value="right"/>
            <Direction value="forward"/>
            <Part value="default"/>
        </ControlPoint>
        <ControlPoint name="p2">
            <Position value="right"/>
            <Direction value="backward"/>
            <Part value="default"/>
        ...
        <Behaviors>
            <Behavior name="point">
                <SubBehavior name="go" controlpoint="p1" category="navigation"/>
                <SubBehavior name="point" controlpoint="p1" category="manipulation"/>
                <SubBehavior name="stand" controlpoint="p2" category="gesture"/>
            </Behavior>
            <Behavior name="annotate">
                <SubBehavior name="go" controlpoint="p1"/>
                <SubBehavior name="draw_line" controlpoint="p1"/>
            ...
    </Object>
</ObjectList>
```

그림 6 교육 도메인의 객체모델의 예



(a) 교육 도메인 객체의 정의의 예 (b) 교육 도메인에서 객체들과 상호작용하는 아바타

그림 8 교육 도메인 객체의 정의 예 및 아바타와의 상호작용

록 정의되었기 때문이다. 때문에 작업수준 행위를 아바타의 동작 시퀀스들로 변환하여 표현해주는 계층이 필요하다.

상위수준 동작 스크립트는 도메인 및 구현 환경과 독립적으로 존재 하기 위해 아바타의 동작을 요소를 통하여 표현한다. 이를 위해 제안 언어에서는 작업수준 행위를 상위수준 동작으로 구성 및 표현하기 위해서 필요한 요소들을 분석하고 정의하였다.

본 논문에서는 상위수준 동작을 크게 3가지 분류로 나누어 표현하고 있다. 가상 공간에서 아바타의 이동 동작을 표현하는 항해(Navigation) 동작, 가상 객체들과 상호작용하는 조작(Manipulation) 동작, 그리고 인사 혹은 유휴상태와 같은 제스처(Gesture) 동작이 그것이다(그림 9). 작업수준 행위 스크립트는 이와 같은 상위수준 동작들의 조합으로 구성된 스크립트로 변환이 된다.

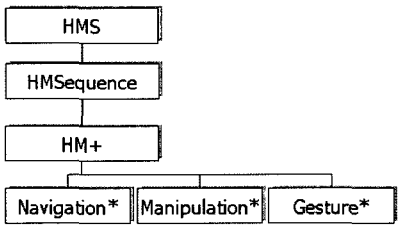


그림 9 상위수준 동작 스크립트의 구성

이때 상위수준 동작 스크립트는 아바타의 일반적인 동작을 물리적인 기하정보나 특정 애니메이션 엔진의 형식이 아닌 추상적인 요소들로 표현한다. 따라서 특정 도메인이나 구현 환경에 종속되지 않고 동작을 표현한다. 일반적이고 포괄적인 요소로 모든 종류의 동작을 공통적으로 표현하던 기존연구[14]와 달리, 제안 스크립트에서는 3가지 분류의 동작이 가지는 특성에 맞게 요소를 정의하여 동작을 표현한다. 상위수준 동작 스크립트의 전체 DTD는 [부록B]에 첨부하였다.

3.2.1 공통(Common) 요소

동작의 종류에 상관없이 적용될 수 있는 공통 요소들은 항해, 조작, 말하기 및 제스처 동작에 모두 사용된다. 동작의 명칭, 동작들간의 동기화 제어, 우선 순위나 전체적인 속도 조절이 공통 요소에 속한다.

- Motion : 작업수준 행위를 구성하는 상위수준 동작의 명칭
- Sync : 상위수준 동작들의 논리적인 동기화를 제어
- Priority : 복수의 동작이 동시 수행될 때 동작의 우선권을 지정
- Speed : 동작의 재생 속도 지정

'Motion'에서는 동작의 명칭(name)과 식별자(id)로 이루어져 있다. 'Sync' 요소에서는 동작간의 시간적인 재생 관계를 표현한다. 여기서 'Sync' 요소는 객체간 가능한 7가지 동기화를 정의한 'Allen's relations'[21]을 사용하고 있다(그림 10). 동기화 표준으로 널리 쓰이는 SMIL의 경우, 초 단위의 물리적인 시간 요소로 미디어

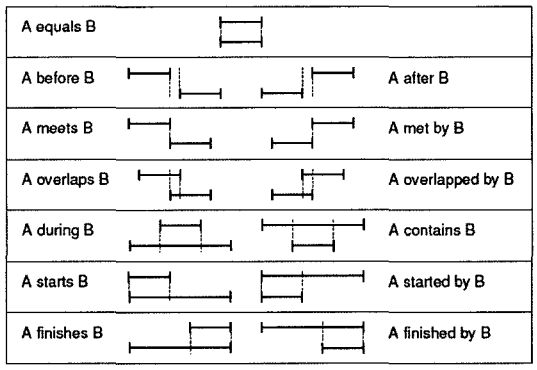


그림 10 Allen's Relation

간의 동기화를 표현하고 있는데 제안된 상위수준 동작 스크립트는 구현환경에 독립적인 추상적인 표현 방식을 사용하고 있기 때문에 적합하지 않았으며, 'Allen's relation'의 경우에는 구체적인 시간 순서보다는 논리적인 객체간의 동기화 순서를 표현하고 있기에 본 동작 스크립트의 표현에 적합하다.

'Priority'는 두 개 이상의 동작이 병렬 수행될 때 우선 적으로 애니메이션 할 동작에 보다 높은 속성을 지정하며 'Speed'는 동작의 속도를 'slowest'부터 'fastest'까지 7단계로 표현한다

3.2.2 항해(Navigation) 요소

아바타의 작업 행위는 일반적으로 조작 동작을 위해 이동 동작이 선행된다. 이동 동작은 공간 내에서의 움직임에 관련된 것이므로 이를 요소화 할 때 고려해야 할 요소는 다음과 같은 요소들로 정의하였다.

- Destination : 아바타 이동의 종착지
- Direction : 이동중의 아바타 방향
- Acceleration : 이동시 가속 및 감속 방법을 지정
- Attachment : 이동시 객체의 소지 여부

먼저 'Destination'은 아바타가 작업을 수행하기 위해 도달해야 하는 최종 목적지로 객체명을 지정한다. 'Direction'은 목적지로 이동시 아바타의 머리와 몸체의 방향을 각각 지정한다. 'Acceleration' 요소는 이동시 아바타의 가속도를 선형 혹은 비선형으로 제어할 것인가를 지정한다. 'Attachment'는 이전 작업으로 아바타가 객체를 들고 이동할 것인지를 표현한다.

3.2.3 조작(Manipulation) 요소

'Manipulation' 동작 제어 요소는 아바타와 객체로 나누어져 있다. 아바타의 조작과 관련된 동작은 아바타 자신 뿐 아니라 객체와의 상호작용을 통해서 이루어 진다 따라서 객체를 어떻게 조작할 것인가를 고려해야 한다. 작업 수준에서는 객체에 대해 행위만을 명령하기 때문에 어떤 위치 및 방향에서 어떻게 객체에 동작을 수행할 것인지 모호하다. 때문에 객체 정보를 처리하여 보다 자세하게 어떻게 동작할 것인가를 지정할 수 있도록 다음과 같은 요소를 정의하였다.

- Object : 조작할 객체명과 리소스에 대한 참조
- MPosition : 객체 조작시 아바타의 상대적인 위치
- MDirection : 객체 조작시 아바타의 방향
- MPart : 조작이 행해지는 객체 파트

'Object'는 대상 객체에 대한 정보로 조작할 객체명과 구현 환경에서 쓰일 리소스 모델에 대한 참조를 표현한다. 'MPosition' 및 'MDirection'은 객체를 기준으로 아바타가 상대적으로 어느 위치 및 방향에서 동작을 수행할 것인가를 결정한다. 'MPart'는 아바타가 조작하는 객체의 부분으로 기본값(Default)일 경우는 구현 환경에서 정해진 부분으로 처리되거나 객체의 특정 부분의 명칭의 지정으로 객체의 조작 위치를 정한다. 그림 11은 이와 같은 요소에 의해 표현되는 'point' 행위의 예이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<HMS>
    ...omit...
    <HMSequence TB="point">
        <HM id="h1" name="walk" category="navigation">
            <Destination object="screen" position="front"/>
            <Direction>
                <Face direction="object"/>
                <Body direction="object"/>
            </Direction>
            <Acceleration type="linear"/>
            <Attachment attached="false"/>
            <Common>
                <Sync hmA="h1" relation="meets" hmB="h2"/>
                <Priority value="normal"/>
                <Speed value="normal"/>
            </Common>
        </HM>
        <HM id="h2" name="point" category="manipulation">
            <Object name="screen"/>
            <MPosition position="front"/>
            <MDirection direction="object"/>
            <MPart part="default"/>
            <Common>
                <Sync hmA="h1" relation="met_by" hmB="h2"/>
                <Priority value="normal"/>
                <Speed value="normal"/>
            </Common>
        </HM>
        ...omit...
    </HMSequence>
</HMS>
```

그림 11 'point' 행위를 상위수준 동작 시퀀스로의 표현한 예

3.2.4 제스처(Gesture) 요소

대부분의 작업수준 행위는 객체와의 상호작용으로 이루어 지지만 일부의 행위는 아바타 자신만의 동작으로 수행된다. 예를 들면 인사를 하거나 말하는 동작, 유휴 상태의 동작 등이 그러하다. 이와 같은 동작들은 강도에 따라 동작이 다양하게 변화하는데 예를 들어 강도를 높이면 보다 큰 제스처와 함께 말하는 동작을 재생하거나 반대의 경우도 가능하다. 또한 이동이나 조작과 동작과 같이 목적지나 도착이나 작업의 종료로 동작이 종료되는 것이 아니기 때문에 적절한 재생 시간을 정해 줄 수 있도록 지속 기간을 정해주어야 한다.

- Intensity : 동작의 세부 조정
- Duration : 동작의 지속 기간 지정

'Intensity'는 'weakest'부터 'strongest'까지 총 7단계로 지정된다. 어떤 알고리즘으로 얼마의 가중치로 동작을 강조할 것인가는 구현 환경의 기능에 달려있다. 'Duration' 역시 'shortest'부터 'longest'까지 7단계로 지정된다. 단, 공통 요소인 Sync 요소에 의해 다른 동작과 동기화가 이루어졌을 경우에는 'Sync' 요소의 지정값을 따른다.

3.3 기본 동작 스크립트 언어

상위수준에서 생성된 동작 시퀀스는 구현 환경에 독립적인 추상적인 표현 방식을 가지고 있으므로 이것이

실제 구현환경에 적용되려면 구현 환경의 물리적인 정보를 표현해주는 스크립트가 필요한데 이것이 기본 동작 스크립트이다.

기본 동작은 특정 아바타 엔진 혹은 동작 라이브러리가 지원하는 동작의 종류, 가상 환경에 배치된 객체들의 기하 정보등과 같은 구현 환경의 물리적인 정보를 표현한다. 기본 동작 역시 상위수준 동작과 같이 요소로 동작을 표현하는데 큰 차이점은 이들 요소가 구현 환경에 종속된 것이라는 것이다. 즉, 상위수준 동작 스크립트는 구현물의 물리적인 동작 및 기하정보에 따라 적절한 기본 동작 스크립트의 요소 형식으로 변환된다. 생성된 기본 동작 스크립트는 최종적으로 구현환경에서 사용하는 하위수준 애니메이션을 호출하고, 스크립트의 요소에 의해 애니메이션 데이타를 적절히 재구성하여 아바타 동작을 제어하게 된다.

기본 동작은 구현 환경의 아바타 엔진이나 동작 라이브러리가 지원하는 동작을 제어하기 위한 스크립트이다. 따라서 소프트웨어마다 각기 다른 종류의 기본 동작 집합을 표현한다. 기본 동작도 상위수준 동작 스크립트와 마찬가지로 요소들로 동작을 제어하고 있는데, 이들 요소들은 현재 구현환경의 기하 정보를 반영하므로 2D 혹은 3D 좌표값, 초 단위의 시간과 같이 물리적인 값으로 동작을 제어한다.

기본 동작 역시 상위수준 동작과 마찬가지로 이동, 조작 및 제스처와 연관된 요소들을 가지고 있다. 그러나 구현 환경의 정보들을 표현 할 수 있도록 보다 낮은 표현 수준을 가진다. 따라서 상위수준 동작 언어에서 같이 객체명이나 의미적인 단어로 표현되던 요소의 값들이 그림 12와 같이 3D 좌표계나 구현 환경에서 사용되는 수치값으로 변환되어 표현된다. 각 요소 표현 형식 및 방법은 다음과 같다.

• 동작

상위수준 동작의 'Navigation', 'Manipulation' 및 'Gesture'에 해당하는 기본 동작을 구현물의 동작 라이브러리에 등록되어 있는 동작으로 변환한다. 번역기는 상위수준 동작들과 동작 라이브러리의 동작들에 대한 정보를 구성하고 이를 참조한다.

• 대상 객체 및 아바타의 방향, 위치

기본 동작에서는 3D 및 2D 좌표계를 사용하고 있으며 상위수준에서 표현하는 목표 객체명을 구현 환경의 위치좌표로 변환하기 위해서 가상 공간을 구성하고 있는 장면그래프(Scene Graph)를 순회하며 좌표를 계산한다. 제안 시스템에서는 가상 공간을 표현하기 위해 VRML 포맷을 사용하고 있으며 장면그래프에서 객체를 식별하기 위한 정보로 객체명 등의 정보를 디자인 단계에서 지정하였다.

• 속도 및 강도

속도와 강도는 애니메이션 엔진에서 지원하는 수치에 따라 계산되어 변환된다. 예를 들어 아바타의 이동시 속도의 가중치를 표현하는 '<Acceleration type="incremental"/>'와 같은 상위수준 요소는 본 논문에서 구현된 시스템에서 지원하는 삼각함수 곡선을 이용한 가속 보간 기능을 표현하기 위해 '<Acceleration value="sin(t)"/>'와 같은 형태로 변경되며 강도는 정수로 표

```
<?xml version="1.0" encoding="UTF-8"?>
<PMS>
    <Seq>
        ...omit...
        <PM id="p1" name="walk.xml" category="navigation" dur="3s">
            <Destination>
                <Coord3D x="-120" y="-20" z="-32"/>
            </Destination>
            <Direction>
                <Face><Coord3D x="0" y="78" z="0"/></Face>
                <Body><Coord3D x="0" y="78" z="0"/></Body>
            </Direction>
            <Acceleration type="sin(90)"/>
            <Attachment attached="0"/>
            <Common>
                <Repeat value="3"/>
                <Priority value="1"/>
                <Speed value="5"/>
            </Common>
        </PM>
        <PM id="p2" name="point.xml" category="manipulation" begin="3s" dur="2s">
            <MPosition>
                <Coord3D x="-120" y="-20" z="-32"/>
            </MPosition>
            <MDirection>
                <Coord3D x="0" y="89" z="0"/>
            </MDirection>
            <MPart>
                <Coord3D x="-125" y="12" z="14"/>
            </MPart>
            <Common>
...omit...
```

그림 12 'point' 동작 순서를 기본 동작 스크립트로 표현한 예

현된다.

• 반복 동작

걷기와 같이 반복되는 동작은 가상 공간에서 아바타와 대상 객체 사이의 거리를 계산하여 적절한 반복 회수를 지정한다. 예를 들어 아바타의 위치가 (x_a, y_a, z_a), 객체의 위치가 (x_t, y_t, z_t), 그리고 걷는 기본 동작의 한 주기로 갈 수 있는 거리를 D_w라 하면 목적지까지의 기본 동작의 반복회수는 다음과 같다.

$$repeat = \text{Square root of } | (x_a-x_t) + (y_a-y_t)+ (z_a-z_t) | / D_w$$

• 동기화

기본 동작의 동기화는 SMIL의 PAR, SEQ 요소를 채용하여 표현한다. 이는 상위수준의 논리적인 동기화 값에 따라 하위수준 동작 데이타를 물리적인 초단위로 제어할 수 있도록 하기 위함이다. 즉, 실제로 재생될 애니메이션 데이타의 지속시간에 따라 상위수준의 논리적인 동기화가 구현환경에 적용될 수 있는 물리적인 동기화로 계산되어 변환된다.

예를 들어 칠판 객체에 걸어가서 칠판을 가리키며 말하는 동작을 하는 상위수준 동작의 경우는 그림 13에서 보는 것과 같이 명확한 시간 개념이 아닌 동작의 전후 연결 관계만을 지정하고 있다. 기본 동작에서는 이와 같은 관계들을 현재 구현 환경에서 실제 애니메이션을 재생할 필요한 시간을 계산하여 지정한다.

4. 구현 결과

본 절에서는 제안된 기법을 이용한 아바타 활용 시스템에 대해서 설명한다. 제안 시스템은 사이버 교육 도메인에서 아바타를 활용하고 있으며 저자, 즉 강사가 작성한 작업수준 행위 시나리오에 따라 아바타는 학생들에게 강의를 진행한다. 구현 환경은 일반적인 강의실로 강사 아바타, 강의 칠판, 텍스트, 컴퓨터, 탁자, 상자 객체 등으로 구성되어 있다. 저자는 작업수준 행위 스크립트 언어를 이용하여 아바타와 이들 객체들간의 행위를 기술한다. 작성된 스크립트는 제안 번역기에 의해 상위수준 스크립트로 변환되고, 이는 다시 기본 동작 스크립트로 변환되어 구현 시스템에 적용된다(그림 14).

스크립트 번역기의 역할은 상위 수준의 스크립트를 읽어 들이고 이를 파싱(Parsing)하여 하위 수준의 스크립트 형식으로 변환 및 생성하는 것이다. 제안 기법에서는 하향식 파서(Top-down parser)를 이용하여 최상위 수준인 작업수준 스크립트를 상위수준 동작 및 기본 동작 스크립트로 번역한다. 따라서 제안 기법에서는 작업수준 행위 스크립트 번역기와 상위수준 동작 스크립트 번역기의 두 가지 종류의 번역기를 사용한다.

1) 작업수준 행위 번역기

작업수준 행위 번역기는 행위 분할기(Behavior Divider)와 요소 생성기(Parameter Generator)로 구성되어 있다. 이들은 각각 상위수준 동작 시퀀스를 생성하고 각각의 동작들에 대해 앞 절에서 정의한 요소 값을 생성한다. 행위 분할기에서는 사용자가 작성한 하나의 작업수준 행위를 번역기에 미리 정의된 작업계획 템플릿(Task-planning template)에 의해 한 개 이상의 하위 행위(Sub-Behavior)로 분할한다. 하위 행위는 다시 복수의 하위 행위로 분할 될 수 있으며 행위가 더 이상 하위 행위로 변환되지 않을 때까지 작업을 수행한다.

작업이 완료되면 작업수준 행위가 부모 노드인 트리 형태의 행위 구조가 완성된다. 트리의 단말 노드들은 더 이상 하위 행위로 분할 되지 않는 행위이며 요소 생성기는 이들 행위에 동작 제어에 필요한 요소들을 생성한다. 마지막으로 트리의 단말 노드들을 순서대로 구성하면 해당 작업수준 행위에 대한 상위수준 동작 스크립트가 생성된다.

2) 상위수준 동작 스크립트 번역기

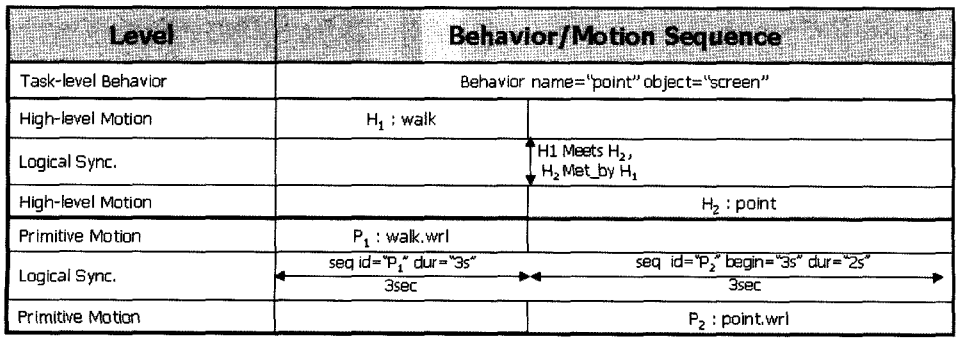


그림 13 상위수준 동작 및 기본동작 스크립트의 동기화 표현

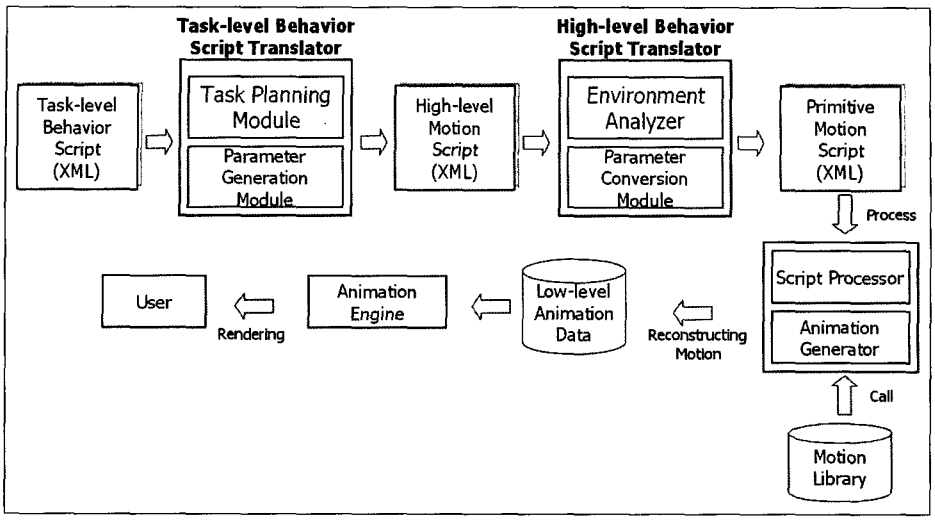


그림 14 시스템 구성도

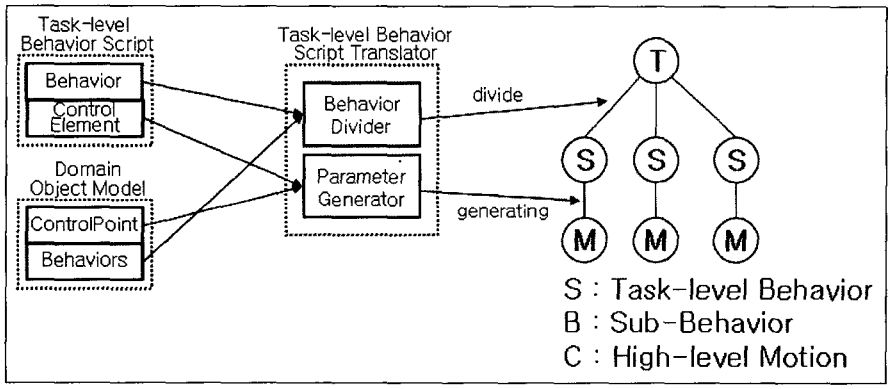


그림 15 작업수준 행위 스크립트 번역기의 구성 및 변환과정

상위수준 동작 스크립트 번역기는 생성된 상위수준 동작 스크립트를 읽어 들여 구현환경의 정보를 추출 및 분석하여 요소를 기본 동작 스크립트에 맞도록 변환한다. 상위레벨 동작의 자연어적인 정보를 이와 같은 구현환경의 수치 정보로 변환하기 위해서는 변환 대응 정보가 필요하다. 제안 시스템의 구현 환경에서는 이러한 정보들을 관리하기 위해서 환경 정보 파일을 가진다. 환경정보 파일은 상위수준 동작명과 애니메이션 데이타 파일의 대응 정보, 가상 공간에서 각 객체의 중심위치 좌표와 이를 기준으로 상대적인 전후 및 좌우 지점의 좌표 등을 담고 있다. 구현환경에 속한 번역기는 이들 정보를 이용하여 기본 동작 스크립트의 정보를 생성한다. 번역기는 환경변수 파일의 정보를 이용하여 추상적인 상위수준 동작 요소를 물리적인 수치로 바꾸어 기본 동작 스크립트를 생성한다.

기본 동작 스크립트는 응용 프로그램에 따라 표현 정보나 구조가 다를 수 있다. 제안 언어에서는 스크립트를 재생하기 위한 기본적인 요소들을 정의하고 있으며 번역기에서는 이들 요소들을 응용 프로그램의 기본 동작 스크립트의 표현 양식에 맞는 형식으로 변환한다.

3) 기본 동작 스크립트에 의한 애니메이션 재생

기본 동작 스크립트가 생성되면 스크립트 재생기가 기본 동작 스크립트의 요소에 맞게 동작 라이브러리의 애니메이션들을 호출 및 재구성하여 아바타를 애니메이션 시킨다. 본 논문에서는 기본 동작 라이브러리의 구축을 위해 아바타 동작 편집기를 구현하여 이를 이용하였다.

개발한 동작 편집기는 H-Anim 표준 명명법을 따르는 16 관절쌍으로 이루어진 아바타의 각 관절을 키프레임 방식으로 조작하여 XML 기반의 저수준 애니메이션 데이타(Low-level animation data)를 생성하고 이를 기본 동작 라이브러리에 등록하는 기능을 가진다.

```
<EnvironmentVariables>
    <MotionMapping>
        <Motion hm="go" pm="walk.xml"/>
        <Motion hm="greet"
pm="wave_hands.xml"/>
        <Motion hm="point"
pm="raise_right_hands.xml"/>
        ...
    </MotionMapping>
    <ObjectCoord>
        <object name="screen">
            <center x="116" y="-101" z="50"/>
            <boundingbox x="94" y="57" z="12"/>
        </object>
        ...
    </ObjectCoord>
    <speed>
        <degree max="fastest" value="10"/>
        ...
    </speed>
    ...
</EnvironmentVariables>
```

그림 16 환경변수 파일의 예

생성된 저수준 애니메이션 데이타는 H-Anim 표준 명명법을 따르므로 이와 호환되는 제안 스크립트 시스템의 아바타 모델에 적용이 가능하다. 따라서 아바타 동작 편집기에서 생성한 애니메이션 데이타들은 하나의 동작 라이브러리로 구축되고, 기본 동작 스크립트가 스크립트 처리기에서 해석되어 해당 동작 시퀀스를 순차적으로 호출하게 된다. 예를 들어 'walk.xml', 'raise_ righ_hand.xml' 과 같은 애니메이션 데이타들을 호출하면 이들은 아바타 렌더링 엔진(Avatar Rendering Engine)에 제공되며 최종적으로 H-Anim 표준을 따르는 관절을 회전 및 이동시켜 아바타 애니메이션을 사용자에게 제공하게 된다. 이와 같은 과정은 다음 그림에 나타나 있다.

또한 기본 동작 스크립트는 구현 환경의 물리적인 구조에 대한 정보에 맞게 요소를 표현하고 있으므로 같은

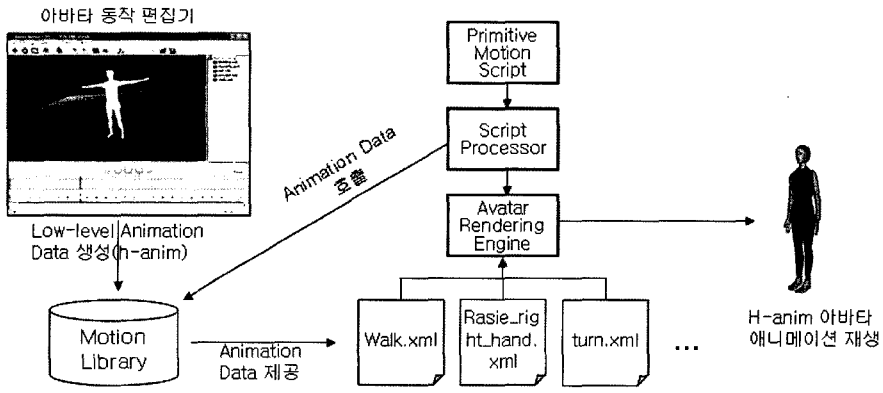
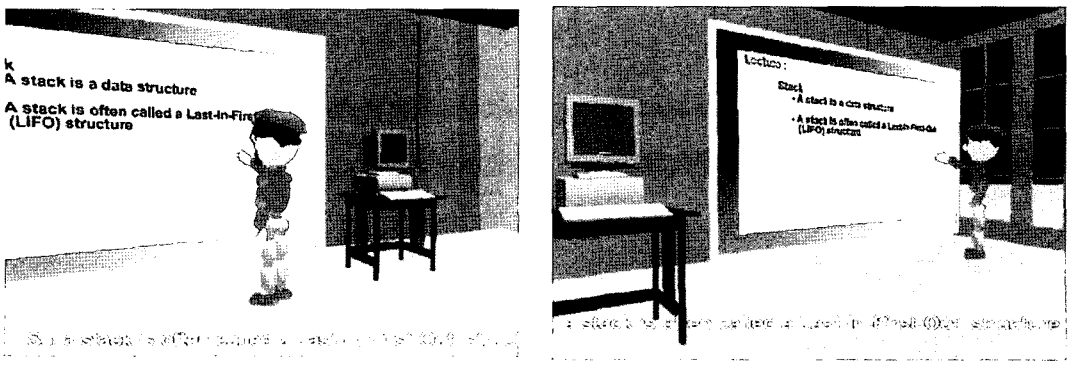


그림 17 기본 동작 스크립트에 의한 저수준 애니메이션 구성 및 재생



(a) 객체들과 상호작용하는 아바타 (b) 객체의 위치가 재구성된 환경

그림 18 사이버 교육 환경에서 강의를 진행하는 아바타

작업수준 스크립트가 물리적인 구조가 다른 구현환경에 적용되어도 기본 동작만 영향을 받는다. 따라서 그림 15와 같이 아바타는 객체의 위치가 재구성된 경우에도 적절한 위치에서 애니메이션을 재생하는 것을 볼 수 있다.

또한, 작업수준 스크립트 언어는 완전히 다른 구현 환경이나 다른 환경에도 적용될 수 있다. 이는 제안된 계층적 행위 표현 구조에 의해 응용 프로그램과 스크립트 언어가 3계층으로 분리되어 있기 때문에 스크립트 언어가 구현 환경의 종류에 크게 구애 받지 않고 다양하게 적용이 가능하도록 한다.

이러한 스크립트의 재사용성은 상위수준 동작 스크립트가 존재하기에 가능해진다. 작업수준과 기본 동작 스크립트로만 이루어진 시스템이라면 n개 도메인의 작업수준 스크립트를 n개의 응용 프로그램에 종속된 기본 동작 스크립트로 변환하려면 n^2개의 스크립트 번역기가 요구된다. 제안 시스템의 경우 다수의 도메인 환경의 스크립트를 하나의 상위수준 동작으로 변환한다. 상위수준 동작은 작업수준 행위 수행에 필요한 아바타의 행위를 동작으로 분화시켜 시퀀스로 연결하였으며 동작 제어에 필요한 다양한 요소들을 객체모델을 이용하여 생성된다.

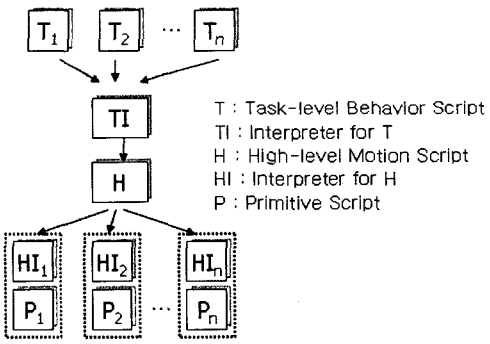


그림 19 제안 시스템의 스크립트와 번역기의 변환 구조

따라서 n개의 응용 프로그램에서는 총 n개의 번역기를 가지고 동일한 형식의 상위수준 동작 스크립트를 해당 응용 프로그램에서 사용되는 기본 동작 스크립트로 변환하여 결과적으로 여러 도메인에서 사용되는 작업수준 행위 스크립트를 실행할 수 있다.

임의의 응용 프로그램에서 사용하는 스크립트 형식으로 상위수준 동작 스크립트를 변환할 때, 3.2절에서 설명한 요소 정보들을 이용하게 된다. 만약 이들 요소가 아바타의 모든 동작을 표현해야 한다면 변환과정이 매우 복잡하고 제대로 이루어 지지 않을 수 있지만, 제안 시스템은 유한한 도메인 환경에서의 작업을 수행하기 위한 동작들에 대한 요소 정보를 이용하는 것이므로 이러한 변환 과정이 가능하여 진다.

구현 결과에서는 교육 도메인 환경의 작업수준 스크립트를 앞서 언급한 3D 응용 프로그램 외에 다음과 같은 웹 응용 프로그램과 및 모바일 응용 프로그램 시스템을 구현하여 적용하였다.

5. 실험 결과 및 토의

본 장에서는 제안 스크립트에 대한 사용성을 평가하기 위하여 사용자 평가를 수행하였다. 본 실험에서는 사용자들에게 제안 스크립트의 내용을 소개하고, 스크립트의 작성을 쉽게 하도록 제작된 별도의 편집기를 제공하여 10가지의 동작을 정의하도록 하였다. 또한 관련 연구에서 언급한 AML과 CPSL과 제안 스크립트 언어를 비교하여 보도록 한다. 모든 실험은 사용자들로 하여금 만족도를 평가하도록 하였으며 그 이유는 스크립트에 대한 사용성 측정은 컴퓨터상에서 정량적인 측정이 어렵기 때문이다.

5.1 실험 1

본 실험에서는 먼저 사용자들에게 각각 10가지의 동작을 제시하고 이것을 본 연구에서 제안한 스크립트를

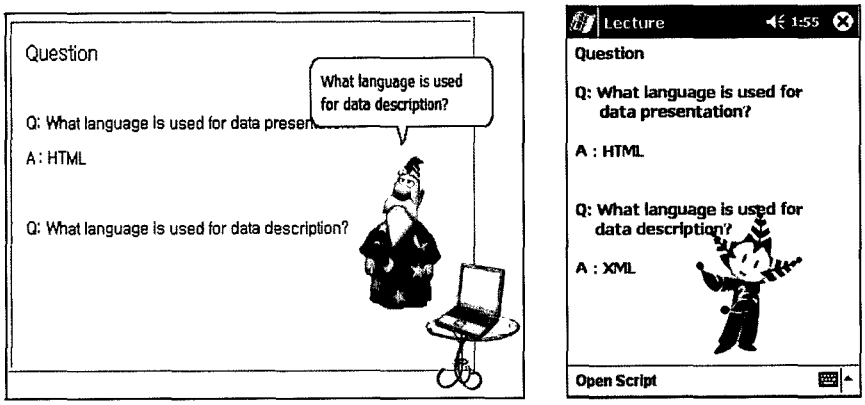


그림 20 동일 작업수준 행위 스크립트를 웹 및 모바일 응용 프로그램에 적용한 예

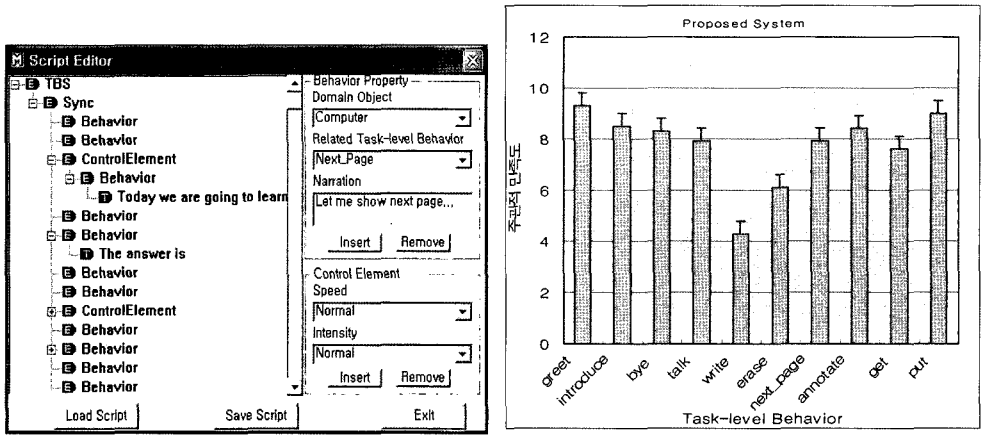


그림 21 실험에 사용된 스크립트 편집기와 평가 결과

이용하여 작성하도록 하였다. 스크립트 작성은 본 연구에서 개발한 다음 그림 21과 같은 별도의 편집기를 이용하며, 정의된 동작은 구현 시스템을 통하여 사용자에게 출력된다. 스크립트 작성 후에는 자신이 작성한 스크립트 내용과 실제 시스템에서 구현된 동작에 자신의 의도가 얼마만큼 정확히 반영되는가를 알아보기 위하여 1(최하의 정확도)부터 10(최상의 정확도)까지의 항목을 포함한 설문 조사를 실시하였다. 정확도 5이하의 경우는 그 이유를 별도의 설문지에 작성하도록 하였다. 본 실험에는 20명의 실험자(남자 17, 여자 4, 대학원생)가 참여하였으며, 실험을 위해서는 교육 도메인 환경의 프로토타입 시스템을 사용하였다.

각 시스템에 따른 사용자들의 정확도를 분석하기 위하여 본 실험에서는 One-way ANOVA(analysis of variance)를 사용하였으며 그 결과는 그림 21과 같다. 그림 21은 사용자들이 정의한 각 동작들의 평가에 대한 평균을 나타내고 있다. 또한 실험결과 전체적으로 유의함($F(1,114) = 15.8$, $P < 0.005$)이 없었으며, 이는 각 동작들의 타입이 전체만족도에 미치는 영향이 없다는 사실을 보여주고 있다. 실험 중 5이하의 만족도를 나타낸 'write' 동작은 설문결과 스크린에 필기를 하는 것이 직관적이지 못하고 또한 구체적으로 어느 영역에 필기를 하는 것인지 사용자가 이해하기 어려운 부분이 있었으며, 향후 이것은 수정되어야 할 것이다. 또한 설문결과 편집기를 사용한 동작정의는 한계가 있음을 알 수 있었으며, 향후 보다 직관적인 사용자 인터페이스가 필요할 것으로 판단된다.

5.2 실험 2

본 실험에서는 관련 연구에서 언급한 AML과 CPSL과 제안 스크립트 언어간의 사용성을 비교한다. AML은 동작과 스피치의 동기화와 다양한 요소를 통한 동작의 제어가 가능하며 CPSL은 아바타를 이용하여 사이버 강의를 진행시키기 위한 언어이다.

시나리오를 작성의 용이성을 측정하기 위해서 피실험자에게 각 스크립트의 문법 및 작성법을 숙지하도록 한 후 스크립트를 학습하는데 소요된 시간을 측정하였다. 다음으로 각 스크립트를 이용하여 10가지의 간단한 작업을 수행하는 시나리오를 작성하도록 하였다. 예를 들어 1. 문을 연다, 2. 책상으로 걸어간다, 3. 손을 흔든다 등의 10가지 작업들을 각 3가지 스크립트를 이용하여 작성하였다. 실험을 위하여 사용자들의 스크립트 작성시간을 측정하였으며, 각 스크립트 언어에 대한 1부터 10까지의 학습 용이도를 사용자가 평가하도록 하였다.

피실험자는 총 20명으로 14명의 남자와 6명의 여자로 구성되었으며 모두 대졸 이상의 학력을 가졌다. 결과를 분석하기 위해서 ANOVA를 이용하여 반복측정을 하였으며 그림 22는 실험 결과를 보여주고 있다.

그림에서 보듯이 사용자들은 제안된 언어를 이용하여 보다 적은 시간과 노력으로 스크립트를 작성하였음을 알 수 있다. 또한 각 스크립트 언어간의 학습 용이도 또

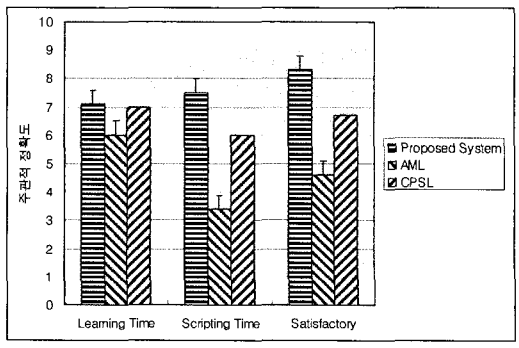


그림 22 스크립트 작성에 대한 사용자 만족도

한 제안 스크립트를 가장 긍정적으로 평가 하였으며, 실험을 통하여 유의효과(Significant effects)를 찾을 수 있었다. ($F(2,57) = 8.98$, $P < 0.05$). 그 의미는 제안 스크립트를 통하여 가장 적은 시간으로 동작을 정의할 수 있었으며, 그것이 학습 용이도에 영향을 미친다는 것이다. 그 이유는 대부분의 피실험자들이 기존 언어들이 동기화, 좌표계 형식의 위치지정, 기타 요소를 지정하는 것이 학습 시간 및 작성에 어려움을 느낀 반면 제안 언어에서는 이러한 요인이 상대적으로 적기 때문이었다. 따라서 스크립트 작성시에는 위와 같은 요소들을 최소화하는 동시에 위 요소들을 사용하여야 하는 경우 별도의 사용자 인터페이스를 제공하여 사용성을 높이는 방법이 반드시 필요하다고 판단된다.

6. 결론 및 향후 연구

제안 스크립트 언어는 계층적 구조를 가지며 작업수준 행위, 상위수준 동작 및 기본 동작 스크립트 언어의 총 3가지로 구성되어 있다. 작업수준 행위는 특정 도메인 환경 내에서 아바타가 수행하는 작업들로 구성되어 있어 사용자가 작업수준의 행위들을 아바타에게 명령함으로써 보다 쉽게 아바타 행위 시나리오 스크립트를 작성할 수 있도록 한다. 이는 하위의 상위수준 동작으로 변환되어 표현되는데 이것은 추상화된 요소들로 작업을 수행하기 위한 동작들을 표현 및 제어하며 특정 도메인이나 구현 환경에 종속되지 않는다. 상위수준 동작은 다시 기본 동작 스크립트로 변환되는데 이는 특정 구현물의 아바타 엔진이 및 기하 정보를 반영하여 동작을 제어하게 된다. 이와 같은 계층적 구조를 통하여 최상위 계층인 작업수준 행위 스크립트는 구현환경과 완전히 분리되어 다양한 응용 프로그램에서 재사용이 가능하다.

즉, 제안 기법의 목적은 사용자가 구현환경의 복잡한 요소를 신경 쓰지 않고 추상화된 스크립트 언어를 통하여 아바타 행위를 쉽게 표현 및 제어하고, 또한 작성된 스크립트의 재사용성을 높여 여러 환경에 적용이 가능하게 하는데 있다.

향후 연구로는 GUI기반의 스크립트 작성 인터페이스 기법을 개발하여 사용자가 직관적으로 아바타 행위를 제어하고 이를 시나리오 스크립트로 생성하는 것이 필요하다.

참 고 문 헌

[1] Prendinger, H.: Life-like Characters. Life-like characters book, Springer-Verlag, pp. 3-17, 2003.

[2] M. Gutierrez, F. Vexo, D. Thalmann, "The Mobile Animator: Interactive Character Animation in Collaborative Virtual Environments," IEEE Virtual Reality 2004 conference, pp. 125-132, Chicago, Illinois, March 27-31, 2004.

[3] Patrick Doyle, "Believability through context using "knowledge in the world" to create intelligent characters," International Conference on Autonomous Agents, Pages: 342-349, 2002.

[4] Marc Cavazza, Fred Charles, Steven J., "Interacting with virtual characters in interactive storytelling," International Conference on Autonomous Agents, pp. 318-325, 2002.

[5] James C. Lester, Sharolyn A., Susan E. Kahler, S. Todd Barlow, Brian A. Stone, Ravinder S. Bhogal, "The persona effect: affective impact of animated pedagogical agents," Proceedings of the SIGCHI, pp. 359-366, 1997.

[6] Thalmann, D.: Autonomy and Task-Level Control for Virtual Actors. Programming and Computer Software, No. 4, 1995.

[7] Badler, N., Bindiganavale, R., Allbeck, J., Schuler, W., Zhao, L., Palmer, M., "A parameterized action representation for virtual human agents," Embodied Conversational Agents, ed Cassell, J., MIT Press, Cambridge, pp. 256-284, 2000.

[8] Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica Hodgins, and Nancy Pollard, Interactive Control of Avatars Animated with Human Motion Data, ACM Transactions on Graphics (SIGGRAPH 2002), volume 21, number 3, 491-500, July 2002.

[9] E. Andre, J. Muller, and T. Rist. WebPersona: A LifeLike Presentation Agent for the World-Wide Web. In Proc. of the IJCAI Workshop on Animated Interface Agents: Making them Intelligent, Nagoya, 1998.

[10] Xiaoli Yang; Petriu, D.C.; Whalen, T.E.; Petriu, E.M., Script language for avatar animation in 3D virtual environments, Virtual Environments, Human-Computer Interfaces and Measurement Systems, VECIMS '03. 2003 IEEE International Symposium on, pp. 101-106, 2003.

[11] Perlin, A. Goldberg, "Improv: A System for Scripting Interactive Actors in Virtual Worlds," Proc. Siggraph 96, H. Rushmeier, ed., ACM Press, NewYork, pp. 205-216, 1996.

[12] Marc S. Atkin, Gary W. King, David L. Westbrook, Brent Heeringa, Paul R. Cohen, "Hierarchical agent control: a framework for defining agent behavior," Proceedings of the fifth international conference on Autonomous agents, pp. 425-432, 2001.

[13] Kshirsagar, S., Thalmann, D., Kamyab, K.: Avatar Markup Language. Proceeding of the workshop on Virtual environments, 169-177, 2002.

[14] Arafa, Y., Mamdani, E.: Scripting embodied agents behaviour with CML. Proceeding of Intelligent User Interfaces, pp. 313-315, 2003.

[15] Marriott, A., Stallo, J.: VHML- Uncertainties and

Problems A. discussion. Proceeding of Embodied conversational agents for AAMAS2002, Bologna, Italy, 2002.
[16] Yoshiaki, S., Matsuda, H.: Design and Implementation of Scenario Language for Cyber Teaching Assistant. International conference on Computers in Education, 2001.
[17] Hayashi, M.: TVML. ACM SIGGRAPH 98 Conference on applications, 292-297, 2003.
[18] Huang, Z., Eliens, A., Visser, C.: Implementation of a scripting language for VRML/X3D-based embodied agents. Proceeding of web technology, pp. 91-100, 2003.
[19] Lester, C., Zettlemoyer, S., Gregoire, P., Bares, H.: Explanatory Lifelike Avatars. Autonomous Agents, pp. 30-45, 1999.
[20] Ricket, J., Johnson, W.: Task-Oriented Collaboration with Embodied Agents in Virtual Worlds. Embodied Conversational Agents, MIT Press, pp. 95-122, 2000.
[21] Frédéric Devillers IRISA, Campus de Beaulieu, F-35042 Rennes, "A scenario language to orchestrate virtual world evolution," 2003 ACM SIGGRAPH/ Eurographics Symposium, San Diego, California, pp. 265-275, 2003.
[22] Bowman, D. and Hodges, L., "Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments," The Journal of Visual Languages and Computing, vol. 10, no. 1, pp. 37-53, 1999.

부 록

[부록 A] 작업수준 행위 스크립트 DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--================================-->
<!-- Scenario  Script Grammar DTD -->
<!--================================-->
<!ELEMENT   TBS                 (Sync)>
<!ELEMENT   Sync                (Behaviors | ControlElement)+>
<!ELEMENT   Behaviors           (#PCDATA)>
<!ELEMENT   ControlElement      (Behaviors | Sync)+>

<!ATTLIST   Sync
    type    %Synchronization;   #REQUIRED
>
<!ATTLIST   Behaviors
    name    %BehaviorSet;       #REQUIRED
    object  %ObjectSet;         #REQUIRED
>
<!ATTLIST   ControlElement
    speed   %SpeedValue;        #REQUIRED
    intensity %IntensityValue;  #REQUIRED
>

<!--================================-->
<!—        Control Element Attribute Set          -->
<!--================================-->
<!ENTITY %  SpeedValue      "(slowest | slower | slow | normal | fast | faster |fastest)">
<!ENTITY %      IntensityValue      "(weakest | weaker | weak | normal | strong | stronger |
strongest)">
```

[부록 B] 상위수준 동작 스크립트 DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--================================-->
<!-- High-level Motion Script Structure          -->
<!--================================-->
<!ELEMENT   HMS             (HMSequence)>
<!ELEMENT   HMSequence      (HM+)>
<!ELEMENT   HM              (%navigation; | %manipulation; | %gesture; | Common)+>
<!ATTLIST   HMSequence
    TBName  CDATA           #REQUIRED
    cid     CDATA           #REQUIRED
>
```

```
<!ATTLIST    HM
    name     CDATA                  #REQUIRED
    id       CDATA                  #REQUIRED
    category %category;             #REQUIRED
>
<!ATTLIST    HMSequence
    TBName   CDATA                  #REQUIRED
    cid      CDATA                  #REQUIRED
>
<!ENTITY %   category                "(gesture | manipulation | navigation)">

<!--================================-->
<!-- Common Parameters                -->
<!--================================-->
<!ELEMENT    Common                 %common;>
<!ENTITY %   common                 "(Sync, Priority, Speed)">
<!ELEMENT    Sync                   EMPTY>
<!ATTLIST    Sync
    hmA      CDATA                  #REQUIRED
    relation %relation;                 #REQUIRED
    hmB      CDATA                  #REQUIRED
>
<!ELEMENT    Priority               (#PCDATA)>
<!ATTLIST    Priority
    value    %priority;             #REQUIRED
>
<!ENTITY %   priority               "(lowest | lower | low | normal | high | higer | highest)">
<!ELEMENT    Speed                  (#PCDATA)>
<!ATTLIST    Speed
    value    %speed;                #REQUIRED
>
<!ENTITY %   speed                  "(slowest | slower | slow | normal | fast | faster | fastest)">
<!ENTITY %   relation               "(equals | before | after | meets | met by | overlaps |
overlapped by | during | contains | starts | started by | finishes | finished by)">

<!--================================-->
<!-- Navigational Parameters        -->
<!--================================-->
<!ELEMENT    Destination            EMPTY>
<!ELEMENT    Acceleration           EMPTY>
<!ELEMENT    Direction              (Face, Body)>
<!ELEMENT    Face                   EMPTY>
<!ELEMENT    Body                   EMPTY>
<!ATTLIST    Acceleration
    type     %accel;                #REQUIRED
>
<!ATTLIST    Destination
    position %position;             #REQUIRED
    object   CDATA                  #REQUIRED
>
<!ATTLIST    Face
    direction %direction;           #REQUIRED
>
<!ENTITY %   navigation             "Destination? | Direction? | Acceleration? | Attachment?">
<!ENTITY %   accel                  "(linear | nonlinear)">
<!ENTITY %   position               "(front | behind | left | right | near | default)">
<!ENTITY %   direction              "(object | user | forward | backward | left | right | default)">

<!--================================-->
<!-- Manipulation Parameters          -->
<!--================================-->
<!ELEMENT    Object                 EMPTY>
<!ELEMENT    MDirection             EMPTY>
<!ELEMENT    MPosition              EMPTY>
<!ELEMENT    MPart                  EMPTY>
<!ELEMENT    Attachment             (#PCDATA)>
<!ATTLIST    Object
    name     CDATA                  #REQUIRED
>
<!ATTLIST    MDirection
    direction %direction;           #REQUIRED
```

```
>
<!ATTLIST    MPosition
   position  %position;          #REQUIRED
>
<!ATTLIST    MPart
   part      CDATA               #REQUIRED
>
<!ATTLIST    Attachment
   attached  %attach;            #REQUIRED
>
<!ENTITY %   attach              "(true | false)">
<!ENTITY %   manipulation        "Object? | MPosition? | MDirection? | MPart?">

<!--==============================-->
<!-- Gesture Parameters           -->
<!--==============================-->
<!ELEMENT    Intensity           EMPTY>
<!ELEMENT    Speech              %speech;>
<!ELEMENT    TTS                 (#PCDATA)>
<!ELEMENT    Filename            (#PCDATA)>

<!ATTLIST    Intensity
   value     %Intensity;         #REQUIRED
>
<!ATTLIST    Speech
   type      %speech;            #REQUIRED
>
<!ENTITY %   gesture             "Intensity? | Speech?">
<!ENTITY %   Intensity           "(weakest | weaker | weak | normal | strong | stronger |
strongest)">
<!ENTITY %       speech          "(TTS | Voicefile)">
```

〔부록 C〕 기본 동작 스크립트 DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--==============================-->
<!-- Primitive Motion Script Structure                    -->
<!--==============================-->
<!ELEMENT PMS                    (Seq|Par)+>
<!ELEMENT PM                     (%navigation;, %manipulation;, Common)>
<!ATTLIST PM
   id        CDATA               #REQUIRED
   name      CDATA               #REQUIRED
   category  %category;          #REQUIRED
             %SMIL;
>
<!ELEMENT    Acceleration        EMPTY>
<!ATTLIST    Acceleration
   type      CDATA               #REQUIRED
>
<!ELEMENT    Attachment          EMPTY>
<!ATTLIST    Attachment
   attached  CDATA               #REQUIRED
>
<!ELEMENT    Body                %Coord;>
<!ELEMENT    Common              %common;>
<!ELEMENT    Coord3D             EMPTY>
<!ATTLIST    Coord3D             %Coord3D;>
<!ATTLIST    Coord2D             %Coord2D;>

<!ELEMENT    Destination         %Coord;>
<!ELEMENT    Direction           (Face, Body)>
<!ELEMENT    Face                %Coord;>
<!ELEMENT    MDirection          %Coord;>
<!ELEMENT    MPart               %Coord;>
<!ELEMENT    MPosition           %Coord;>
```

```
<!ELEMENT   Priority                    EMPTY>
<!ATTLIST   Priority
   value    CDATA                       #REQUIRED
>
<!ELEMENT   Repeat                      EMPTY>
<!ATTLIST   Repeat
   value    CDATA                       #REQUIRED
>
<!ELEMENT   Seq (PM+)>
<!ELEMENT    Par (PM+)>

<!ELEMENT   Speed EMPTY>
<!ATTLIST   Speed
   value    CDATA                       #REQUIRED
>

<!--================================-->
<!-- Motion Category                  -->
<!--================================-->
<!ENTITY %  category                    "(manipulation | navigation | gesture)">
<!ENTITY %  navigation                  "(Destination?, Direction?, Acceleration?,Attachment?)">
<!ENTITY %  manipulation                "(MPosition?, MDirection?, MPart?)">

<!ENTITY %  common                     "(Repeat?, Priority, Speed)">

<!--================================-->
<!-- Coordination Data Representation -->
<!--================================-->
<!ENTITY %  Coord                       "(Coord3D|Coord2D)">
<!ENTITY %  Coord2D
"   x       CDATA                       #REQUIRED
    y       CDATA                       #REQUIRED
    z       CDATA                       #REQUIRED
">
<!ENTITY % Coord3D
"   x       CDATA                       #REQUIRED
    y       CDATA                       #REQUIRED
    z       CDATA                       #REQUIRED
">

<!--================================-->
<!-- Synchronization Parameters       -->
<!--================================-->
<!ENTITY  % SMIL
"   dur     CDATA                      #REQUIRED
    begin   CDATA                      #IMPLIED
">
```

〔부록 D〕 도메인 객체 모델 DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--================================-->
<!—Domain Object Model                -->
<!--================================-->
<!ELEMENT       Behavior                (SubBehavior+)>
<!ATTLIST       Behavior
                name CDATA      #REQUIRED
>
<!ELEMENT       Behaviors (Behavior+)>
<!ELEMENT       ControlPoint (Position, Direction, Part)>
<!ATTLIST       ControlPoint
                name CDATA #REQUIRED
>
<!ELEMENT       Direction EMPTY>
<!ATTLIST       Direction
                value CDATA #REQUIRED
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!--================================-->
<!—Domain Object Model                -->
<!--================================-->
<!ELEMENT       Behavior                (SubBehavior+)>
<!ATTLIST       Behavior
                name CDATA      #REQUIRED
>
<!ELEMENT       Behaviors (Behavior+)>
<!ELEMENT       ControlPoint (Position, Direction, Part)>
<!ATTLIST       ControlPoint
                name CDATA #REQUIRED
>
<!ELEMENT       Direction EMPTY>
<!ATTLIST       Direction
                value CDATA #REQUIRED
>
<!ELEMENT       Motions (Sequence+)>
<!ELEMENT       Object (ControlPoint+, Behaviors, Motions)>
<!ATTLIST       Object
                name CDATA #REQUIRED
>
<!ELEMENT       ObjectList (Object)>
<!ATTLIST       ObjectList
                domain CDATA #REQUIRED
>
<!ELEMENT       Part EMPTY>
<!ATTLIST       Part
                value CDATA #REQUIRED
>
<!ELEMENT       Position EMPTY>
<!ATTLIST       Position
                value CDATA #REQUIRED
>

<!ELEMENT       Sequence (motion+)>
<!ATTLIST       Sequence
                name CDATA #REQUIRED
>
<!ELEMENT       SubBehavior EMPTY>
<!ATTLIST       SubBehavior
                name CDATA #REQUIRED
                controlpoint CDATA #REQUIRED
>
<!ELEMENT       motion EMPTY>
<!ATTLIST motion
                name CDATA #REQUIRED
                controlpoint CDATA #REQUIRED
                category (gesture | manipulation | navigation) #REQUIRED
                                                >
```

김 재 경

2000년 단국대학교 화학/전산통계 학사
2002년 연세대학교 컴퓨터과학 석사
2006년 현재 연세대학교 컴퓨터과학과 박사과정. 관심분야는 아바타 행위 제어, Annotation 생성 및 응용

최 승 혁

2004년 홍익대학교 컴퓨터과학과 학사
2006년 현재 연세대학교 컴퓨터과학과 석사과정. 관심분야는 컴퓨터 그래픽, 아바타 행위 제어, 비실사적 렌더링

손 원 성

1998년 동국대학교 컴퓨터공학 학사 2000년 동국대학교 컴퓨터공학 석사 2004년 연세대학교 컴퓨터과학과 박사 2004년~2006년 Carnegie Mellon University, Associate Researcher. 2006년 현재 경인교육대학교 컴퓨터교육학과 조교수. 관심분야는 웹 Annotation 생성 및 응용

임 순 범

1982년 서울대학교 계산통계학 학사 1983년 한국과학기술원 전산학 석사 1992년 한국과학기술원 전산학 박사 1989년~1992년 (주)휴먼컴퓨터 이사/연구소장. 1992년~1997년 (주)삼보컴퓨터 부장. 1997년~2001년 건국대학교 컴퓨터과학과 조교수. 2003년~현재 숙명여자대학교 멀티미디어학과 조교수. 관심분야는 컴퓨터 그래픽스, 멀티미디어 응용, 전자출판(폰트, 전자책, 사이버교재)

최 윤 철

1973년 서울대학교 학사. 1975년 Univ. of Pittsburgh 석사. 1976년 Univ. of California, Berkeley 석사. 1979년 Univ. of California, Berkeley 박사. 1979년~1982년 Lockheed 사 및 Rockwell International 사 연구원. 1990년~1991년 University of Massachusetts 교환교수. 2002년~2003년 일본 게이오대학 교환 교수. 1984년~현재 연세대학교 컴퓨터과학과 교수. 관심분야는 멀티미디어와 웹, 멀티미디어 문서처리, 3D 사용자 인터페이스, 아바타 인터페이스, 컴퓨터그래픽스, eLearning 및 CyberClass