

계층적 행위 스크립트 표현을 통한 아바타-객체 행위 제어를 위한 인터페이스 기법

(An Interface Technique for Avatar-Object Behavior Control using Layered Behavior Script Representation)

최승혁[†] 김재경[†] 임순범[‡] 최윤철^{***}

(Seung-Hyuk Choi) (Jae-Kyung Kim) (Soon-Bum Lim) (Yoon-Chul Choy)

요약 본 논문에서는 아바타의 행위(Avatar Behavior)를 상위 레벨 명령어(High-Level Behavior)들로 이루어진 스크립트로 제어하는 기법을 제안한다. 아바타 행위를 추상화 정도에 따라 구분하고 이를 표준화된 계층적 스크립트(Layered Script)로 정의 함으로써 사용자는 스크립트를 재사용할 수 있고 행위의 추상화 조절이 가능하다. 또한 가상 환경이 복잡해짐에 따라 아바타 행위도 다양하게 늘어날 뿐 아니라 아바타-객체 상호작용 처리 역시 복잡해지고 있다. 이러한 문제점을 해결하기 위해 아바타와 객체간의 상호작용을 위한 객체 모델을 제안하여 아바타와 객체간 벌어지는 행위들을 객체 안에 분산시켜 표현함으로써 객체지향 방식으로 아바타 행위를 유연하게 제어할 수 있도록 설계하였다. 이를 위해 제안 객체 모델에서는 객체의 상태에 따라 사용 가능한 행위가 결정되는 컨텍스트 메뉴(context menu) 인터페이스와 동작 생성 모델을 정의하였다. 또한 사용자는 기존의 2D 혹은 텍스트기반 스크립트 작성기법을 벗어나 제안된 3D 인터페이스 기법을 통하여 실시간으로 아바타의 행위 스크립트를 작성 및 재생 할 수 있다. 본 연구에서는 제안 기법의 활용을 위해 프레젠테이션 도메인 환경의 시스템을 구축하고 아바타-객체 행위제어 및 스크립트 생성 기법을 적용하였다. 본 논문에서는 아바타의 행위(Avatar Behavior)를 일종의 상위 레벨 명령어(High-Level TaskBehavior)들로 이루어진 스크립트로 제어하는 기법을 제시한다. 아바타 행위를 추상화 정도에 따라 구분하고 이를 표준화된 계층적 스크립트(Multi-LevelLayered Script)로 정의 함으로써 사용자는 쉽게 스크립트를 재사용할 수 있고 행위의 추상화 정도도 쉽게 조절이 가능하다. 또한 또한 아바타와 객체간의 상호작용을 위한 객체 모델을 제시한다. 가상 환경이 복잡해짐에 따라 아바타 행위도 다양하게 늘어날 뿐 아니라 아바타-객체 상호작용 처리 역시 복잡해지고 있다. 이러한 문제점을 해결하기 위해 아바타와 객체간의 상호작용을 위한 객체 모델을 제안하여 아바타와 객체간 벌어지는 모든 행위들을 객체 안에 분산시켜 저장표현함으로써 객체지향 방식으로 아바타 행위를 유연하게 제어 객체의 수와 무관하게 상호작용을 처리할 수 있도록 설계하였다. 이를 위해 또한 복잡해진 제안 객체 모델에서는 객체의 상태에 따라 사용 가능한 행위가 결정되는 가상 환경을 위해 새로운 인터페이스로 컨텍스트 메뉴(context menu) 인터페이스와 동작 생성 모델을 제시한다. 정의하였다. 객체 모델에서 객체의 상태 정보와 행위 정보를 분석해 아바타가 할 수 있는 행위를 컨텍스트 메뉴로 제공하기 때문에 사용자는 가상 환경의 상태에 상관 없이 직관적으로 명령을 줄 수 있다. 또한 사용자는 기존의 2D 혹은 텍스트기반 스크립트 작성기법을 벗어나 사용자는 제안된 3D 인터페이스 기법을 통하여 실시간으로 아바타의 행위 스크립트를 작성 및 재생 할 수 있다. 본 논문에서 제시한 시스템은 기존의 아바타 중심적인 제어를 객체에 분산함으로써 효율적이고 직관적인 명령을 내릴 수 있고 또한 손쉬운 시나리오 생성을 가능하게 하였다. 본 연구에서는 제안 기법의 활용을 위해 프리젠테이션 도메인 환경의 시스템을 구축하고 아바타-객체 행위제어 및 스크립트 생성 기법을 적용하였다.

키워드 : 아바타-객체 상호작용, 행위제어, 스크립트언어, 아바타제어 인터페이스

Abstract In this paper, we suggested an avatar control technique using the high-level behavior. We separated behaviors into three levels according to level of abstraction and defined layered scripts.

· 본 연구는 한국과학재단 목적기초연구(R01-2004-000-10117-0(2004))지원
으로 수행되었음

[†] 학생회원 : 연세대학교 컴퓨터과학과

alienart@rainbow.yonsei.ac.kr
ki187cm@rainbow.yonsei.ac.kr

^{‡‡} 종신회원 : 숙명여자대학교 멀티미디어과학과 교수

sblim@sookmyung.ac.kr

^{***} 종신회원 : 연세대학교 컴퓨터과학과 교수

ycchoi@rainbow.yonsei.ac.kr

논문접수 : 2005년 10월 11일

심사완료 : 2006년 7월 28일

Layered scripts provide the user with the control over the avatar behaviors at the abstract level and the reusability of scripts. As the 3D environment gets complicated, the number of required avatar behaviors increases accordingly and thus controlling the avatar-object behaviors gets even more challenging. To solve this problem, we embed avatar behaviors into each environment object, which informs how the avatar can interact with the object. Even with a large number of environment objects, our system can manage avatar-object interactions in an object-oriented manner. Finally, we suggest an easy-to-use user interface technique that allows the user to control avatars based on context menus. Using the avatar behavior information that is embedded into the object, the system can analyze the object state and filter the behaviors. As a result, context menu shows the behaviors that the avatar can do. In this paper, we made the virtual presentation environment and applied our model to the system.

In this paper, we suggested the technique that we controling an the avatar control technique using the high-level behavior. We separated behaviors into three levels byaccording to level of abstract levelion and defined multi-leveIlayered script. Multi-levelLayered script offers that the user can control avatar behavior at the abstract level and reuses script easily. We suggested object models for avatar-object interaction. Because, TtThe 3D environment is getting more complicated very quickly, so that the numberss of avatar behaviors are getting more variableincreased. Therefore, controlling avatar-object behavior is getting complex and difficultWe need tough processing for handling avatar-object interaction. To solve this problem, we suggested object models that embedded avatar behaviors into object for avatar-object interaction. insert embedded all avatar behaviors into object. Even though the numbers of objects areis large bigger, it can manage avatar-object interactions by very efficientlyobject-oriented manner. Finally Wewe suggested context menu for ease ordering. User can control avatar throughusing not avatar but the object-oriented interfaces. To do this, Oobject model is suggested by analyzing object state and filtering the behavior,behavior and context menu shows the behaviors that avatar can do. The user doesn't care about the object or avatar state through the related object.

Key words : Requirements Change Management, Requirements Change Management Process, Software Product Lines

1. 서 론

최근 3차원 그래픽 기술의 발달로 가상 환경에 대한 관심이 증가되고 있다. 그 중 아바타는 사람이나 동물과 같이 살아있는 듯한 느낌을 전해주기 때문에 웹, 게임, 영화 등 다양한 방면으로 활용되고 있다. 하지만 아바타 연구의 대부분은 자연스러운 동작을 생성하는 것에 집중되어 있을 뿐 아바타 제어에 관련된 연구는 상대적으로 적다 할 수 있다. 이 때문에 현재 3D 아바타의 동작 제어 방법은 다소 복잡하여 전문가만이 가능하거나 일반인의 경우 극히 한정된 행위만을 지원하기 때문에 활용 범위 역시 한정되어 있는 실정이다.

아바타 행위 제어의 가장 초기 형태는 저 수준 애니메이션이라 할 수 있다. 3DS Max, Maya와 같은 편집 툴에서 아바타의 신체 각 관절을 조작하여 동작을 생성하거나 모션 캡쳐(Motion Capture) 장비를 이용하여 동작을 생성하는 것이다. 이는 세밀한 동작의 생성이 가능하지만 전문 애니메이터(Animator)나 값비싼 장비가 요구되기 때문에 일반인이 사용하기 힘들다는 단점이 있다.

이러한 저 수준 애니메이션보다 한 단계 발전한 기법은 스크립트를 이용한 아바타 제어 기법이다. 최근 3차

원 가상 환경의 발달로 인해 다양한 가상 시뮬레이션 기법들이 연구 되고 있고, 이러한 시뮬레이션 프로그램이 활성화되면서 아바타의 행위를 일종의 상위 레벨 명령어들로 이루어진 스크립트로 제어하는 연구가 활발하게 이루어지고 있다. 이러한 상위 레벨 스크립트를 아바타 제어에 사용하게 되면 사용자는 일일이 관절을 조작 할 필요 없이 '걸기'와 같은 좀 더 추상적인 명령으로 편리하게 아바타를 제어할 수 있을 뿐 아니라 이를 기록, 재생할 수 있다. 하지만 추상화된 행위는 단순히 아바타를 추상화 정도에 따라 원하지 않는 행위들이 생겨 날 수 있기 때문에 사용자가 추상화 정도를 쉽게 조절 할 수 있어야 한다. 또한 추상화된 행위의 처리에 있어서 아바타 자체 행위의 경우 쉽게 처리가 가능하지만 아바타-객체간 상호 작용을 처리하기 위해서는 공간적, 논리적 모호성을 제거해주어야 한다. 또한 가상 환경이 복잡 해짐에 따라 객체의 수와 더불어 행위수도 늘어나게 되는데 기존의 기법들은 아바타 중심적으로 되어 있기 때문에 객체를 늘리는 데 어려움이 따르며 사용자가 일일이 스크립트로 작성해야 하기 때문에 환경이 복잡해지면 명령을 내리는 것도 더욱 복잡해 진다. 이를 위해 분

산형 시스템 도입이 필요하며 손쉽게 명령을 내릴 수 있는 새로운 인터페이스 개발 역시 필요하다.

본 논문에서는 행위 추상화 문제를 해결하기 위해 추상화 정도에 따라 아바타 행위를 구분하였고 계층별로 사용자 접근이 가능하게 하여 아바타 행위의 추상화 정도를 쉽게 조절할 수 있게 하였다. 이 때문에 좀 더 유연하게 아바타를 제어할 수 있다. 뿐만 아니라 표준형식인 XML 기반의 계층적 스크립트를 사용함으로써 하드웨어, 운영체제 등 플랫폼에 구애 받지 않고 스크립트를 재사용 할 수 있어 스크립트 생산성을 높였다. 또한 아바타-객체간 상호 작용을 처리하기 위해 아바타와 객체간 벌어지는 모든 행위들을 객체 안에 분산시켜 저장할 수 있는 객체 모델(Object Model)을 정의하였다. 이는 객체 상태에 따른 아바타 행위 모호성과 공간적 논리적 모호성을 제거해줄 수 있다. 또한 직관적인 사용자 인터페이스 제공이 가능한데, 상태에 따라 아바타-객체간 벌어질 수 있는 행위들을 판단하여 할 수 있는 행위들만 컨텍스트 메뉴로 보여주기 때문에 사용자는 가상 환경 내의 상태 변화를 생각할 필요 없이 제공되는 컨텍스트 메뉴 인터페이스를 통하여 손 쉽게 아바타 및 객체를 제어할 수 있다. 뿐만 아니라 사용자는 아바타 하나만을 통해 명령을 내리는 것이 아니라 분산된 객체를 통해서 명령을 내릴 수 있기 때문에 객체가 많은 복잡한 가상 환경 내에서는 좀 더 효율적으로 아바타를 제어할 수 있다. 마지막으로 컨텍스트 메뉴를 통해 명령을 내리면 실시간으로 그 행위가 애니메이션으로 보여주고 스크립트로 기록됨으로써 직관적으로 시나리오를 생성할 수 있다.

제안 기법의 활용을 위해 본 논문에서는 프리젠테이션 도메인 환경에서의 강의 시스템 환경을 구축하고 아바타-객체 모델과 컨텍스트메뉴 및 계층적 행위 제어 스크립트 기법을 적용하였다.

최근 아바타 활용과 관련된 많은 연구가 진행되고 있으며 있지만, 실제로 웹, 사용되고 있는 분야는 게임, 영화 등 극히 한정된 곳에서만 여러 분야에서 활발히 활용되고 있다. 이러한 아바타의 활용을 위해 중요한 이슈 중의 하나는 3D 아바타의 동작 제어 기법이다. 그러나 현재 3D 아바타의 동작 제어 방법은 다소 복잡하여 전문 애니메이터가 아닌 이처럼 아바타 활용이 부진한 이유 중 하나가 바로 아바타의 제어가 일반 사용자에게는 아바타의 제어가 어려운 설정이다. 너무 어렵다는 것이다.

아바타 행위 제어의 가장 초기 형태는 저 수준 애니메이션이라 할 수 있다. 3DS Max, Maya나 Poser와 같은 편집 툴에서 아바타의 신체 각 관절을 조작하여 동작을 생성하거나 모션 캡처(Motion Capture) 장비를

이용하여 동작을 생성하는 것이다. 이는 세밀한 동작의 생성이 가능하지만 전문 애니메이터(Animator)나 값비싼 장비가 요구되기 때문에 일반인이 사용하기 힘들다는 단점이 있다.

최근 3차원 가상 환경의 발달로 인해 다양한 가상 시뮬레이션 기법들이 연구 되고 있다. 이러한 시뮬레이션 프로그램이 활성화되면서 아바타의 행위를 일종의 상위 레벨 명령어들로 이루어진 스크립트로 제어하는 연구가 활발하게 이루어지고 있다. 이러한 상위 레벨 스크립트를 아바타 제어에 사용하게 되면 사용자는 일일이 관절을 조작 할 필요 없이 'walk'와 같은 좀 더 추상적인 명령을 줄 수 있기 때문에 좀 더 편리하게 아바타를 제어 할 수 있다. 하지만 추상화된 행위는 추상화 정도에 따라 원하지 않는 행위들이 생겨날 수 있기 때문에 사용자가 추상화 정도를 쉽게 조절 할 수 있어야 할 뿐 아니라 또한 효율적인 시나리오 생성을 위해서는 스크립트 재사용성을 높여야 한다. 또한 추상화된 행위의 처리에 있어 아바타 자체 행위의 경우 쉽게 처리가 가능하지만 아바타-객체간 상호 작용을 처리하기 위해서는 다양한 모호성을 제거해주어야 한다. 또한 가상 환경이 복잡해짐에 따라 행위수도 급격히 늘어나게 되는데 기존의 아바타 중심적인 명령이나 텍스트 기반의 명령들을 그대로 사용하게 된다면 사용자는 보다 복잡한 방식으로 아바타 행위를 제어해야 한다. 예전 혼란을 야기할 것이다. 이를 위해 손쉽게 명령을 내릴 수 있는 새로운 인터페이스 개발이 필요하다.

본 논문에서는 이러한 문제점을 해결하기 위해 추상화 정도에 따라 아바타 행위를 구분하였고 이를 기반으로 객체 기반의 계층적 스크립트를 정의 하였다. 표준형식인 XML 기반의 계층적 스크립트를 사용함으로써 하드웨어, 운영체제 등 플랫폼에 구애 받지 않고 스크립트를 재사용 할 수 있다. 또한 아바타 행위의 추상화 정도를 쉽게 조절할 수 있기 때문에 좀 더 유연하게 아바타를 제어할 수 있다.

또한 아바타-객체간 상호 작용을 처리하기 위해 아바타와 객체간 벌어지는 모든 행위들을 객체 안에 분산시켜 저장할 수 있는 객체 모델을 정의하였다. 이는 객체 상태에 따른 아바타 행위 모호성과 공간적 논리적 모호성을 제거해줄 수 있다. 또한 직관적인 사용자 인터페이스 제공이 가능한데, 상태에 따라 아바타-객체간 벌어질 수 있는 행위들을 판단하고 할 수 있는 행위들만 컨텍스트 메뉴로 보여주기 때문에 사용자는 가상 환경 내의 상태 변화에 상관 없이 따라 재구성되어 제공되는 행위 인터페이스를 통하여 직관적으로 아바타, 객체를 제어할 수 있다. 또한 뿐만 아니라 사용자는 아바타 하나가 아닌 다양한 객체를 통해서 명령을 내릴 수 있기 때문에

객체가 많은 복잡한 가상 환경 내에서는 좀 더 편리하게 아바타를 제어할 수 있다. 또한 그리고 컨텍스트 메뉴를 통해 명령을 내리면 실시간으로 그 행위가 애니메이션으로 보여주고 스크립트로 기록됨으로써 손쉽게 실시간으로 시나리오를 생성할 수 있다.

제안 기법의 활용을 위해 본 논문에서는 이러한 기술의 기반으로 외부 콘텐츠와 결합하여 교육프리젠테이션 도메인 환경에서의 가상강의 시스템 환경을 구축하고 아바타-객체 모델과 컨텍스트메뉴 및 계층적 행위 제어 스크립트 기법을 적용하였다. 시뮬레이션을 생성해보았다. 전자 책 표준 XML(EBKS 1.1)을 사용하여 강의 콘텐츠를 실제 생성하였다.

2. Related Work관련 연구

아바타의 행위를 상위 레벨 명령어들로 제어하는 연구[1]가 활발하게 이루어지고 있다. 이러한 상위 레벨 명령들을 아바타 제어에 사용하게 되면 좀 더 추상적인 명령을 줄 수 있기 때문에 사용자는 편리하게 아바타를 제어할 수 있다. 이를 연구는 크게 아바타가 행위의 중심이 되는 아바타 스크립트, 그리고 아바타의 행위를 객체에 분산시킨 객체 기반 아바타 행위 스크립트로 구분할 있다. 아바타 스크립트 언어를 이용한 아바타 제어는 미리 정의된 동작 명령어와 속성 값(Parameter)을 아바타에게 입력시키고, 아바타는 그 명령어에 따른 저 수준 동작 데이터를 호출, 합성하여 가상 환경 및 속성 값에 따라 동작 데이터를 재구성하여 사용자에게 재생하는 방법이다. 저 수준 동작 데이터와 달리 아바타 관절의 회전 값 등을 고려하지 않고 ‘걷기’, ‘열기’ 등과 같은 상위레벨의 행위 명령을 통하여 아바타를 제어하기 때문에 사용자 이해 및 작성성이 보다 용이하다. 또한 표준 포맷을 사용한 스크립트는 특정 구현 환경에 종속되지 않고 여러 응용프로그램에서 재사용될 수 있다. 이와 같은 스크립트 언어에는 CML[2], AML[3], STEP[4], TVML[5]등이 있다. 그러나 대부분의 스크립트 언어가 아바타 자체의 동작만을 표현하고 있기 때문에 다양한 객체들과의 상호작용을 제어하기에는 한계가 있다. 최근 3차원 그래픽스의 발전으로 인해 3차원 가상 환경 내 객체들이 계속 복잡해지고 있는 추세이다. 이러한 가상 환경 내 다양한 객체들과 아바타를 효과적으로 이용하기 위해서는 새로운 기법이 필요하다.

객체기반 아바타 제어기법은 객체 및 아바타의 행위를 각각의 객체에 분산하여 저장하고 이를 이용하여 동적인 애니메이션을 생성하는 기법이다. 아바타의 행위는 대상이 되는 객체 종류에 따라 다양하게 표현될 수 있다. 따라서 아바타가 행할 수 있는 모든 동작을 하나의 아바타에 표현한다는 것은 매우 힘든 일이라 할 수 있

다. 이 때문에 아바타가 해당 객체와 상호작용 시 필요 한 행위들을 각각의 객체 내부에 표현해 두는 것이다. 따라서 아바타는 걸기나 밟기 등과 같은 기본적인 행위 모델만 가지고 있고 그 외에 객체와의 상호작용을 할 때에는 해당 객체의 행위 모델을 이용하여 그에 적합한 행위를 수행 할 수 있다는 장점이 있다. 대표적인 연구로 Alice[6], Smart Object[7,8], OSR[9]등이 있다.

그러나 기존 연구들은 행위 모델 표현에 있어 표준 형식에 기반하지 않고 자체 포맷을 사용 함으로서 다양한 환경에 적용이 어렵다. 또한 가상 환경이 복잡해짐에 따라 아바타와 객체간의 행위도 복잡해지고 다양해지는 데 기존의 연구[2,5,10]는 한정된 도메인 환경에서만 이루어질 수 있도록 설계되어 있기 때문에 복잡하고 다양한 환경에서 객체의 행위를 유연하고 효율적으로 관리하기 힘들다. 뿐만 아니라 객체는 상황에 따라 아바타에게 여러 가지 다양한 행위를 제공해 줄 수 있는데 예를 들어 문 객체의 경우 닫혀있을 때와 열려있을 때 사용할 수 있는 행위가 다르다. 그러나 기존 연구[5,8,11]는 항상 고정된 행위 인터페이스만을 제공하기 때문에 사용자에게 환경 정보에 따른 행위 인터페이스 모델을 제공해주는 것이 바람직하다. 이에 따른 유저 인터페이스에 대한 연구가 부족하다 할 수 있다.

이러한 문제점을 해결하기 위해 본 논문에서는 주어진 환경과 상황에 따라 다르게 대처할 수 있는 행위 정보와 로직(logic)을 컴포넌트화 하여 각 객체에 분산시켜 넣어주었다. 이렇게 컴포넌트화 하게 되면 가상 환경에 객체를 확장할 때 객체 관련 컴포넌트만 추가 등록 해주면 되기 때문에 쉽게 확장할 수 있다. 또한 객체를 효율적으로 관리 할 수 있는데, 특정 객체 행위를 처리할 때 가상 환경 내 모든 객체를 다 처리하는 것이 아니라 그 때 그 때 관련된 객체만 처리하면 되기 때문에 가상환경 내 객체의 수가 늘어난다거나 상태가 변화한다 해도 처리에 필요한 양은 증가하지 않는다. 또한 객체에 들어있는 컴포넌트에서 환경 정보에 따라 다른 행위 인터페이스 모델을 제공해주기 때문에 사용자는 객체의 상태를 생각할 필요 없이 직관적으로 객체를 제어 할 수 있다. 스크립트 기반의 제어 기법 중 또 하나의 중요한 요소는 바로 행위의 추상화에 있다. 본 연구에서는 아바타 객체간 행위를 분석하여 3 단계 계층으로 나누어 구조화 하였다. 이러한 구조화는 특히 행위의 추상화 레벨과 관련이 깊은데, 사용자는 이를 사용하여 편리하게 추상화 레벨을 조절하여 표현할 수 있다. 또한 스크립트 재사용 측면에서도 유리하다. 상위 레벨의 스크립트는 정확한 기하(geometry) 정보를 담은 것이 아니라 단순히 논리적 정보들로 이루어져 있기 때문에 어플리케이션이나 하드웨어에 독립적으로 재사용할 수 있다.

이러한 재사용성은 스크립트의 생산성을 증대시킨다. 마지막으로 행위의 추상화로 인한 다양한 모호성을 계층적 구조를 통해 해결하였다.

아바타의 행위를 일종의 상위 레벨 명령어들로 제어하는 연구[1]가 활발하게 이루어지고 있다. 이러한 상위 레벨 명령들을 아바타 제어에 사용하게 되면 좀 더 추상적인 명령을 줄 수 있기 때문에 사용자는 편리하게 아바타를 제어할 수 있다. 이들 연구는 크게 아바타가 행위의 중심이 되는 아바타 스크립트, 그리고 아바타의 행위를 객체에 분산시킨 객체 기반 아바타 행위 스크립트로 구분할 있다. 아바타 스크립트 언어를 이용한 아바타 제어는 미리 정의된 동작 명령어와 속성 값(Parameter)을 아바타에게 입력 시키고, 아바타는 그 명령에 따른 저 수준 동작 데이터를 호출, 합성하여 가상 환경 및 속성 값에 따라 동작 데이터를 재구성하여 사용자에게 재생하는 방법입니다. 저 수준 동작 데이터와 달리 아바타 관절의 DOF 값 등을 고려하지 않고 ‘walk’, ‘open’ 등과 같은 상위레벨의 행위 명령을 통하여 아바타를 제어하기 때문에 사용자 이해 및 작성하기가 이보다 용이하다. 또한 표준 포맷을 사용한 스크립트는 특정 구현 환경에 종속되지 않고 여러 응용프로그램에서 재사용될 수 있다. 이와 같은 스크립트 언어에는 CML[2], AML[3], STEP[4], TVML[5] 등이 있다. 그러나 대부분의 스크립트 언어가 아바타 자체의 동작만을 표현하고 있기 때문에 다양한 객체들과의 상호작용을 제어하기에는 한계가 있다. 최근 3차원 그래픽스의 발전으로 인해 3차원 가상 환경 내 객체들이 계속 복잡해지고 있는 추세이다. 이러한 가상 환경 내 다양한 객체들과 아바타를 효과적으로 이용하기 위해서는 새로운 기법이 필요하다.

객체기반 아바타 제어기법은 객체 및 아바타의 행위를 각각의 객체에 분산하여 저장하고 이를 이용하여 동적인 애니메이션을 생성하는 기법이다. 아바타의 행위는 대상이 되는 객체 종류에 따라 다양하게 표현될 수 있다. 따라서 아바타가 행할 수 있는 모든 동작을 하나의 아바타에 표현한다는 것은 매우 힘든 일이라 할 수 있다. 이 때문에 아바타가 해당 객체와 상호작용 시 필요한 행위들을 각각의 객체 내부에 표현해 두는 것이다. 따라서 아바타는 걷기나 말하기 등과 같은 기본적인 행위 모델만을 가지고 있고 그 외에 객체와의 상호작용을 할 때에는 해당 객체의 행위 모델을 이용하여 그에 적합한 행위를 수행 할 수 있다는 장점이 있다. 대표적인 연구로 Alice[6], Smart Object[7,8], OSR[9] 등이 있다.

그러나 기존 연구들은 행위 모델 표현에 있어 표준 형식에 기반하지 않고 자체 포맷을 사용 함으로서 다양한 환경에 적용이 어렵다. 또한 가상 환경이 복잡해짐에

따라 아바타와 객체간의 행위도 복잡해지고 다양해지는 데 기존의 연구[10,11]는 한정된 도메인 환경에서만 이루어질 수 있도록 설계되어 있기 때문에 복잡하고 다양한 환경에서 객체의 행위를 유연하고 효율적으로 관리하기 힘들다. 뿐만 아니라 객체는 상황에 따라 아바타에게 여러 가지 다양한 행위를 제공해 줄 수 있는데 예를 들어 문 객체의 경우 닫혀있을 때와 열려있을 때 사용할 수 있는 행위가 다르다. 그러나 기존 연구[12-14]는 항상 고정된 행위 인터페이스만을 제공하기 때문에 사용자에게 환경 정보에 따른 행위 인터페이스 모델을 제공해주는 것이 바람직하다.

또한 가상 시뮬레이션에서 직관적으로 기록, 재생이 가능해야 하는데 이에 따른 유저 인터페이스에 대한 연구가 부족하다 할 수 있다. 이러한 문제점을 해결하기 위해 본 논문에서는 주어진 환경과 상황에 따라 다르게 대처할 수 있는 행위 셋을 컴포넌트화 하여 각 객체에 분산시켜 넣어주었다. 이러한 분산 시스템은 객체를 세분화하여 관리 할 수 있기 때문에 쉽게 컴포넌트화 할 수 있고 그때 그때 필요한 객체만 처리하면 되기 때문에 가상환경 내 객체의 수가 늘어난다거나 상태가 변화 한다 해도 쉽게 처리 할 수 있다. 또한 객체에 들어 있는 컴포넌트에서 환경 정보에 따른 행위 인터페이스 모델을 제공해주기 때문에 사용자는 직관적으로 객체를 제어할 수 있다. 사용자는 필요한 객체를 선택하고 여기서 할 수 있는 행위를 인터페이스를 통해 쉽게 확인하고 명령을 내릴 수 있기 때문이다. 스크립트 기반의 제어 기법 중 또 하나의 중요한 요소는 바로 행위의 추상화에 있다. 본 연구에서는 아바타 객체간 행위를 분석하여 3 단계로 구조화 하였다. 이러한 구조화는 특히 행위의 추상화 레벨과 관련이 깊다. 사용자가 추상화 정도를 쉽게 조절 하기 위해 각 계층별로 행위의 추상화 정도를 다르게 적용 할 수 있다. 또한 추상화에 따른 다양한 모호성이 생겨날 수 있는데 이러한 모호성을 계층적 구조를 통해 해결하였다.

3. 객체 기반 아바타 제어 기법

본 논문의 목적은 3차원 가상 환경 내 객체를 스크립트로 제어하고 이를 기록하여 가상 시나리오를 생성하고 재생하는 것이다. 여기서 중요한 사항은 어떻게 아바타의 행위를 스크립트로 표현할 것인가와 어떠한 방식으로 스크립트를 생성할 것인가 하는 것이다. 본 논문에서는 객체 기반의 계층적 스크립트를 이용하여 두 가지 문제점을 해결하였다. 즉 ‘의자에 앉아’와 같은 식으로 추상화된 행위를 이용하여 아바타를 제어하는 방식이다.

아바타의 행위는 단독적으로 이루어지기 보다는 대부

분 객체와 관련 지어 이루어진다. 이러한 점을 이용하여 아바타가 할 수 있는 행위를 객체에 분산 시켜 관리하였다. 이러한 분산 시스템은 효율적인 아바타-객체 상호작용 처리를 가능하게 할 뿐만 아니라 사용자 인터페이스에도 큰 영향을 준다. 기존에는 아바타 제어를 위해 아바타를 선택하여 명령해야 했다. 하지만 할 수 있는 행위들이 객체에 저장되어 있기 때문에 사용자는 단지 행위와 관련된 객체를 통하여 명령을 내릴 수 있는 것이다. “의자에 앉아라”의 경우는 의자를 선택한 후 “앉기”를 명령하면 되는 것이다. 이러한 인터페이스는 기존의 수많은 명령들을 검색해야 했던 번거로움을 없애고 효율적이고 직관적으로 명령을 내릴 수 있다는 장점이 있다.

본 논문의 목적은 3차원 가상 환경 내에서 가상 시나리오를 생성하고 재생하는 것이다. 시나리오를 생성, 재생하기 위해 적합한 방법은 바로 구조화된 스크립트를 이용하여 기록하는 것이라 할 수 있다. 여기서 중요한 사항은 어떻게 아바타의 행위를 스크립트로 표현할 것인가와 어떠한 방식으로 스크립트를 생성할 것인가 하는 것이다. 본 논문에서는 객체 기반의 계층적 스크립트를 이용하여 두 가지 문제점을 해결하였다. 우선 ‘의자에 앉아’와 같은 식으로 추상화된 행위를 이용하여 아바타를 제어하는 방식이다.

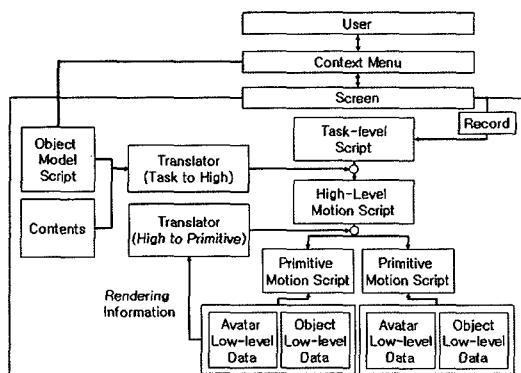


그림 1 System 전체 시스템 구성도

그림 1은 본 논문에서 제시한 아바타 시스템의 전반적인 시스템 구성도이다. 우선 가상 환경 내에 존재하는 모든 객체 모델들을 정의한다. 사용자는 객체가 제시하는 인터페이스(컨텍스트 메뉴)를 통해 명령을 내리고 이러한 명령은 로 저장이 된다. 이는 변환기를 통해 추상화된 명령은 논리적인 행위 시퀀스인 상위레벨 행위 스크립트(High-Level Behavior Script)로 변환된다. 이는 다시 3차원 가상 환경의 정보들을 얻어 정확히 어떠한 위치에서 어떠한 동작을 해야 하는지를 표현한 기본

동작 스크립트(Primitive Motion Script)로 변환되어 되고, 최종적으로 여기에 있는 각종 파라미터 값들을 반영하여 애니메이션을 합성한다. 아바타의 행위는 단독적으로 이루어지기 보다는 대부분 객체와 관련 지어 이루어진다. 이러한 점을 이용하여 아바타가 할 수 있는 행위를 객체에 분산 시켜 관리하였다. 이러한 분산 시스템은 효율적인 아바타-객체 상호작용 처리를 가능하게 할 뿐만 아니라 사용자 인터페이스에도 큰 영향을 준다. 기존에는 아바타 제어를 위해 아바타를 선택하여 명령해야 했다. 하지만 할 수 있는 행위들이 객체에 저장되어 있기 때문에 사용자는 단지 행위와 관련된 객체를 통하여 행위를 명령을 선택하여 실행할 수 있는 것이다. “의자에 앉아라”의 경우는 의자를 선택한 후 “앉기”를 명령하면 되는 것이다. 이러한 인터페이스는 기존의 수많은 명령들을 검색해야 했던 번거로움을 없애고 효율적이고 직관적으로 명령을 내릴 수 있다는 장점이 있다.

그림 1은 본 논문에서 제시한 아바타 시스템의 전반적인 시스템 구성도이다. 우선 가상 환경 내에 존재하는 모든 객체 모델들을 정의한다. 사용자는 객체가 제시하는 인터페이스(컨텍스트 메뉴)를 통해 명령을 내리고 이러한 명령은 작업레벨 행위 스크립트(Task L-level Behavior Script)로 저장이 된다. 이는 변환기를 통해 추상화된 명령은 논리적인 행위 시퀀스인 상위레벨 동작 스크립트(High-Level Motion Script)로 변환된다. 이는 다시 3차원 가상 환경의 정보들을 얻어 정확히 어떠한 위치에서 어떠한 동작을 해야 하는지를 표현한 기본 동작 스크립트(Primitive Motion Script)로 변환되어 되고, 최종적으로 여기에 있는 각종 파라미터 값(parameter value)들을 반영하여 애니메이션을 합성한다.

3.1 아바타 행위 연구

아바타의 행위 연구는 그 목적에 따라 다양하게 구분될 수 있다. 아바타의 감정이나 성격 등을 표현하기 위한 행위 연구, 아바타간 대화를 위한 행위 연구, 군중 아바타 제어를 위한 행위 연구 등 그 목적에 따라 다양한 연구들이 진행되고 있다. 본 논문에서는 아바타의 행위 제어에 있어 사람이 명령할 때 어떻게 해석하여 행동하는가에 대해 연구하였다. 즉 사람이 명령할 때 그 속에 내재된 함축된 의미들이 어떻게 구성되어 있고 이를 어떻게 풀어 최종적으로 사람이 행동하는가에 대한 연구이다.

아바타 행위들은 추상화 정도에 따라 표 1과 같이 크게 3단계로 구분이 가능하다. 첫 번째는 추상화 행위 레벨(Abstract Behavior Level)이다. 이 레벨의 명령들은 한 명령에 다양한 의미들이 압축되어 있는 레벨이다. 추상화 행위는 사람의 지식을 필요로 하는가 혹은 논리적

인 의미로 구성되어 있는가에 따라 지식 기반 행위(Knowledge based Behavior)와 논리적 행위(Logical Behavior)로 구분할 수 있다. 지식 기반 행위는 사람의 지식이 반영되어야 해결 할 수 있는 행위이다. 즉 기계를 분해하거나 조립한다거나, 강의를 할 때 단순히 논리적인 행위로만은 해결 할 수 없고 특정 영역에 대한 배경 지식을 가지고 있어야만 그 행위를 할 수 있는 것이다. 이러한 지식 기반 행위는 다양한 논리적 행위들이 합쳐져 또 다른 행위가 생성되는 것이다. 논리적 행위는 추상화 되어 있지만 논리적으로 해결이 가능한 행위이다. 예를 들어 ‘방으로 들어가라’라고 명령하면 방의 입구를 찾아 걸어가고 문을 열고 방으로 들어가는 일련의 행위들이 조합되어 있지만 논리적으로 쉽게 풀어 해결 할 수 있는 행위들이다. 이러한 행위들은 환경이 변화 하여도 똑 같은 결과를 가져와야 하는데 사람의 위치나 문의 상태 등에 따라 행위들이 변화하기 때문에 이러한 다양한 상태들을 파악해야 문제를 해결할 수 있다.

아바타의 행위 연구는 그 목적에 따라 다양하게 구분지울 수 있다. 아바타의 감정이나 성격 등을 표현하기 위한 행위 연구, 아바타간 대화를 위한 행위 연구, 군중 아바타 제어를 위한 행위 연구 등 그 목적에 따라 다양한 연구들이 진행되고 있다. 본 논문에서는 아바타의 행위 제어에 있어 사람이 명령할 때 어떻게 해석하여 행동하는 가에 대해 연구하였다. 즉 사람이 명령할 때 그 속에 내재된 함축된 의미들이 어떻게 구성되어 있고 이를 어떻게 풀어 최종적으로 사람이 행동 하는 가에 대한 연구이다.

두 번째 단계는 의미적으로 조괄 수 있는 행위들의 최소 단위인 행위 레벨(Behavior Level)이다. 즉 ‘문을 열어라’, ‘컴퓨터로 이동해라’, ‘칠판을 가리켜라’와 같이 하나의 행위가 하나의 의미를 담고 있는 행위들이다. 이는 객체와 상호작용하는가 안 하는가에 따라 아바타-객체 행위(Object-Avatar Behavior)와 아바타 단독 행위(Avatar-Self Behavior)로 구분 지을 수 있다. 아바타-객체 행위는 객체와 연관이 있는 행위들 즉, 문을 연다

거나 칠판을 가리킨다거나 컴퓨터 키보드를 치는 등의 행위들이다. 이러한 행위들 중 고려해야 할 사항은 우리는 단순히 ‘문을 열어라’ 하면 어떠한 문이든지 열어야 한다는 것이다. 그 문이 미닫이 문이든지, 회전 문이든지, 아니면 평범한 문이든지 상관 없이 문을 열 수 있어야 한다. 즉 같은 것으로 분류되는 객체들은 그것의 모양이 어떻게 되던지 간에 똑 같은 결과를 내야 한다는 것이다. 아바타 단독 행위는 객체와 상관없이 스스로 하는 행위이다. 예를 들면 놀라거나 잠시 생각을 한다거나 그 자리를 맴 돈다거나 하는 행위이다. 이러한 행위는 대부분 사람의 생각이나 느낌들을 표현할 때 필요한 행위들이라 할 수 있다.

마지막 단계의 행위는 의미가 없는 단순히 움직임을 표현하는 동작 레벨(Motion Level)이다. 이는 위에서 행위들과 달리 단순히 움직임을 나타내는 움직임의 조각들이라고 할 수 있다. 이것은 아바타의 움직임을 나타내는 아바타 동작(Avatar Motion)과 객체의 움직임을 나타내는 객체 동작(Object Motion)으로 구분 지을 수 있다. 위에서 언급했던 행위와 구분되는 것은 아바타의 동작을 위해 구체적인 위치 정보나 아바타의 움직임 정보가 필요하다는 것이다. 아바타가 한 걸음 한걸음 움직인다거나 문고리를 잡아 돌린다거나 등의 동작을 하기 위해서는 아바타의 위치, 문의 위치, 문고리의 위치 등을 정확히 알 수 있어야 할 뿐 아니라 그에 따른 아바타의 각 관절의 각도들도 알아야 한다. 지금까지 아바타의 행위를 추상화 정도에 따라 구분 지어 보았다. 이러한 행위 분석을 다음과 같은 정의를 가지고 아바타 시스템에 적용하였다.

아바타 행위들은 추상화 정도에 따라 표 1과 같이 크게 3단계로 구분이 가능하다. 첫 번째는 추상화 행위 레벨(Abstract Behavior Level)이다. 이 레벨의 명령들은 한 명령에 다양한 의미들이 암축되어 있는 레벨이다. 추상화 행위는 사람의 지식을 필요로 하는가 혹은 논리적인 의미로 구성되어 있는가에 따라 지식 기반 행위(knowledge based Behavior)와 논리적 행위(logical

표 1 가상 환경에서 아바타, 객체 행위

구분	스크립트	행위	설명	예
Abstract Behavior Level	Task L-level Behavior Script	Knowledge based Behavior(KB)	지식 기반 행위	teach XML
		Task Level Behavior(TB)	논리적 행위	Enter the room
Behavior Motion Level	High L-level Behavior Motion Script	Avatar Self Behavior(SB)	아바타 자체 행위	Walk
		Avatar-Object Behavior(AOB)	객체 상호 작용 행위	Open the door
Action Level	Primitive Script	Avatar Action (AA)	아바타 행동의 최소 단위	Walk (one step)
		Object Action (OA)	객체 행동의 최소 단위	문고리 돌리기

behavior)로 구분할 수 있다. 지식 기반 행위는 사람의 지식이 반영되어야 해결 할 수 있는 행위이다. 즉 기계를 분해하거나 조립한다거나, 강의를 할 때 단순히 논리적인 행위로만은 해결 할 수 없고 조립이나 특정 강의에 대한 배경 지식을 가지고 있어야만 그 행위를 할 수 있는 것이다. 지식 기반 행위는 다양한 논리적 행위들이 합쳐져 또 다른 행위가 생성되는 것이다. 논리적 행위는 추상화 되어 있지만 논리적으로 해결이 가능한 행위이다. 예를 ‘방으로 들어가라’ 라고 명령하면 방의 입구를 찾아 걸어가고 문을 열고 방으로 들어가는 일련의 행위들이 조합되어 있지만 논리적으로 쉽게 풀어 해결 할 수 있는 행위들이다. 이러한 행위들은 환경이 변화 하여도 똑 같은 결과를 가져와야 하는데 사람의 위치나 문의 상태 등에 따라 행위들이 변화하기 때문에 이러한 다양한 상태들을 파악해야 문제를 해결할 수 있다.

두 번째 단계는 의미적으로 쪼갤 수 있는 행위들의 최소 단위인 행위 레벨(Behavior Level)이다. 즉 ‘문을 열어라’, ‘컴퓨터로 이동해라’, ‘칠판을 가리켜라’와 같이 하나의 행위가 하나의 의미를 담고 있는 행위들이다. 이는 객체와 상호작용하는가 안 하는가에 따라 아바타-객체 행위(Object-Avatar Behavior)와 아바타 단독 행위(Avatar-Self Behavior)로 구분 지을 수 있다. 아바타-객체 행위는 객체와 연관이 있는 행위들이다. 예를 들면 즉, 문을 연다거나 칠판을 가리킨다거나 컴퓨터 키보드를 치는 등의 행위들이다. 이러한 행위들 중 고려해야 할 사항은 우리는 단순히 ‘문을 열어라’ 하면 어떠한 문이든지 열어야 한다는 것이다. 그 문이 미닫이 문이든지, 회전 문이든지, 아니면 평범한 문이든지 상관 없이 문을 열 수 있어야 한다. 즉 같은 것으로 분류되는 객체들은 그것의 모양이 어떻게 되던지 간에 똑 같은 결과를 내야 한다는 것이다. 아바타 단독 행위는 객체와 상관없이 스스로 하는 행위이다. 예를 들면 놀라거나 잠시 생각을 한다거나 그 자리를 맴 돈다거나 하는 행위이다. 이러한 행위는 대부분 사람의 생각이나 느낌들을 표현할 때 필요한 행위들이라 할 수 있다.

마지막 단계의 행위는 의미가 없는 단순히 움직임을 표현하는 행동 레벨(Action Level)이다. 이는 위에서 행위들과 달리 단순히 움직임을 나타내는 움직임의 조각들이라고 할 수 있다. 이것 역시 객체와 상호작용 하는가에 따라 아바타 객체 행동(Object-Avatar Action)과 아바타 단독 행동(Object-Avatar Action)으로 구분 지을 수 있다. 위에서 언급했던 행위와 구분되는 것은 아바타의 행위를 위해 구체적인 위치 정보나 아바타의 움직임 정보가 필요하다는 것이다. 아바타가 한 걸음 한걸음 움직인다거나 문고리를 잡아 돌린다거나 등의 행동을 하기 위해서는 아바타의 위치, 문의 위치, 문고리의

위치 등을 정확히 알 수 있어야 할 뿐 아니라 그에 따른 아바타의 각 관절의 각도들도 알아야 한다. 지금까지 아바타의 행위를 추상화 정도에 따라 구분 지어 보았다. 이러한 행위 분석을 다음과 같은 정의를 가지고 아바타 시스템에 적용하였다.

정의 1. 추상화된 행위의 해석을 위해 다음과 같은 형태의 함수 형태로 구체화 할 수 있다.

$$B = F(I_a, I_o, G_o)$$

모든 행위는 객체와 연관 지어 이루어진다. 즉 모든 행위는 아바타에게 명령을 내기긴 하지만 실질적으로 그 명령들은 아바타 자신을 포함한 특정 객체와 연관되어 있다는 것이다. 즉, 기 때문에 사용자는 명령을 내릴 때 하나의 아바타만을 통해서 명령을 내리는 것이 아니라 최종 목표goal과 관련이 있는 객체를 통해서 명령을 내릴 수 있다는 것이다. 즉 이를 수식으로 표현하면 위와 같이 표현이 가능하다. 모든 행위는 특정 객체 o와 관련되어 연관 되어 명령을 할 수 있는 최종 목표(Goal), 목표와 관련 있는 객체 정보(Io), 그리고 아바타 상태 정보(Ia)를 받아 논리적인 알고리즘 함수 F()를 통해 해결한다.

정의 2. 추상화된 행위는 아래와 같이 행위 분할을 통해 구체화 될 수 있다.

```
<ABS>=<LB>|<KB>|<LB><ABS>|<KB><ABS>
<KB>=<LB>|<LB><KB>
<LB>=<SB><LB>|<AOB><LB>|<SB>|<AOB>
<SB>=<AM><SB>|<AM>
<AOB>=<SB>| (<AM><OM>)|<SB><AOB>|
(<AM><OM>)<AOB>
```

<ABS>: Avatar Behavior Script, <LB>: Logical Behavior, <KB>: Knowledge based Behavior, <SB>: Avatar Self Behavior, <AOB>: Avatar-Object Behavior, <AM>: Avatar Motion, <OM>: Object Motion

본 논문에서는 행위의 추상화 정도에 따라 등급을 나누고 이를 통해 행위의 분할 및 호출이 가능하다. 필요에 따라선 같은 등급에서 행위를 분할 호출 하기도하며 또 상황에 따라서는 상위 등급에서 하위 등급으로 행위 분할하게 된다. 이러한 과정을 통해 추상 행위 내 압축 정보들을 풀어 구체화 시킬 수 있다. 위의 식은 이러한 구체화 과정을 BNF 형식으로 표현한 것이다. 이러한 변환 과정은 정해진 규칙에 의해서 세분화 되고 최종적으로 최하위 레벨인 기본 동작 레벨까지 내려와 최종적으로 애니메이션화 할 수 있다.

```

<ABS>=<TB>|<KB>
<KB>=<TB>|<OB>
<TB>=<SB><TB>|<OB><TB>|<TB>
<SB>=<SA><SB>|<SB>
<OB>=<SB>|(<SA><OA>)|<SB><OB>|(<SA><OA>)<OB>|<OB>
  
```

본 논문에서는 행위의 추상화 정도에 따라 등급을 나누고 이를 통해 행위의 분할 및 호출이 가능하다. 필요에 따라선 같은 등급에서 행위를 분할 호출 하기도하며 할 수 있고 그 외에는 대부분은 상위 등급에서 하위 등급으로 행위 분할하게 된다. 이러한 해석으로써 정보를 구체화한다. 이 과정을 통해 추상 행위 내 압축 정보들을 풀어 구체화 시킬 수 있다. 나갈 수 있다. 위의 식은 이러한 구체화 과정을 BNF 형식으로 표현한 것이다. 각각의 레벨로 이러한 변환 과정은 정해진 규칙에 의해서 세분화 되고 최종적으로 최하위 레벨인 행동 레벨까지 내려와 최종적으로 애니메이션화 할 수 있다.

정의 3. 아바타와 객체들의 상호 작용은 관련된 객체들만 부분적으로 이루어진다. 아바타와 객체 혹은 객체와 객체간의 상호작용을 위해서는 객체-아바타간 통신이 이루어져야 한다. 즉 특정 이벤트가 벌어졌을 때 통신을 통해 상호작용이 가능한 것이다. n개의 객체가 존재한다고 할 때 아바타의 통신은 객체 1부터 객체 n 까지 모두 아바타와 통신이 가능하다. 하지만 객체와 객체간 통신의 경우 객체 모두와 통신하는 것이 아니라 필요로 되는 특정 객체끼리만 통신이 이루어진다. 이로써 하나의 행위가 완성되기 위해 모든 객체에 대한 정보를 알 필요가 없고 필요로 되는 객체 몇 개의 정보만 가져오면 된다. 이는 객체의 수가 늘어나도 효율적으로 상호작용을 처리할 수 있다는 장점이 있다. 바타와 객체들의 상호 작용은 다음과 같은 형태로 이루어진다.

$$\begin{aligned}
 O &= \{O_1, O_2, O_3, \dots, O_n\} \\
 C(A) &= \{C(aA, oO_1), C(A, O_o_2) \dots C(A, O_o_n)\} \\
 C(O_i) &= \{C(A, O_{oi}), C(O_{oi}, O_{o1}), C(O_{oi}, O_{o4}) \dots \\
 &\quad C(O_{oi}, O_{ok})\}
 \end{aligned}$$

아바타와 객체 혹은 객체와 객체간의 상호작용을 위해서는 객체-아바타간 통신이 이루어져야 한다. 즉 특정 이벤트가 벌어졌을 때 통신을 통해 상호작용이 가능한 것이다. n개의 Object가 존재한다고 할 때 아바타의 통신($C(A)CA$)은 객체 1과 아바타간의 통신($C(A, O_1)Cao_1$)부터 객체 n과 아바타간의 통신($C(A, O_n)Cao_n$)이 가능하다. 하지만 객체 i의 객체간 통신($C(O_i)Coi$)의 경우 객체 모두와 통신하는 것이 아니라 필요로 되는 특

정 객체끼리만 통신($C(O_i, O_k)CoiO_k$)이 이루어진다. 이로써 하나의 행위가 완성되기 위해 모든 객체에 대한 정보를 알 필요가 없고 필요로 되는 객체 몇 개의 정보만 가져오면 된다. 이는 객체의 수가 늘어나도 효율적으로 상호작용을 처리할 수 있다는 장점이 있다.

정의 4. 객체명은 같지만 종류가 달라지는 경우 그 타입에 따라 다른 형태의 애니메이션이 생성된다. 즉 ‘문을 열어라’ 할 경우 문의 종류 즉, 미닫이문, 회전문, 보통 문, 등에 따라 그 행위가 변형된다. 체명은 같지만 종류가 달라지는 경우 그 타입에 따라 다른 형태의 애니메이션이 생성된다. 즉 ‘문’을 열어라’ 할 경우 문의 종류 즉, 미닫이문, 회전문, 보통 문, 등에 따라 그 행위가 변형된다.

지금까지 3차원 가상 환경 내에 아바타 행위 연구에 대해 살펴보았다. 우선 추상화 레벨에 따른 계층적 모델을 제안하였고 이를 해석하기 위한 원칙들을 제시하였다. 다음 장에서는 모델링을 적용한 실제 시스템 구현을 통하여 얻어진 활용 사례들을 알아보도록 하겠다.

지금까지 3차원 가상 환경 내에 아바타 행위 연구에 대해 연구를 위한 정의를 살펴보았다. 이와 같은 우선 추상화 레벨에 따른 계층적 모델을 제안하였고 이를 해석하기 위한 원칙들을 제시하였다. 다음 장에서는 모델링을 적용한 실제 시스템 구현을 통하여 얻어진 활용 사례들을 알아보도록 하겠다.

3.2 Multi-level Script 계층적 행위 스크립트

아바타의 행위를 사용자가 편리하게 사용할 수 있는 방식은 행위를 추상화하는 방식이다. 즉, ‘강의를 하세요’, ‘문을 여세요’ 등의 추상화된 명령만으로도 아바타를 동작 할 수 있게 하는 것이다. 추상화된 명령으로 아바타를 제어하기 위해서는 추상화된 명령의 압축된 정보들을 풀어 표현할 수 있는 구조적 스크립트가 필요한데 이를 위해 계층적 스크립트를 정의 하였다. 위에서 추상화 정도에 따라 행위를 3가지 레벨로 나누어 설명하였는데, 이를 스크립트에 그대로 반영하여 3개의 계층을 두어 각각의 스크립트를 정의하였다. 계층적 스크립트의 장점은 우선 사용자가 편리하게 추상화 레벨을 조절하여 표현할 수 있다는 것이다. 강의의 예를 들면 행위를 단순히 ‘여기를 가리키세요’라는 저 수준 행위부터 ‘XML 강의를 하세요’와 같은 고수준의 행위까지 편리하게 만들 수 있는 것이다. 또한 스크립트 재사용 측면에서도 유리하다. 상위 레벨의 스크립트는 정확한 기하(geometry) 정보를 담은 것이 아니라 단순히 논리적 정보들로 이루어져 있기 때문에 어플리케이션이나 하드웨어에 독립적으로 재사용할 수 있다[12]. 이 때문에 한 번 만들어진 시나리오는 필요할 때 언제든지 재사용이 가능하다. 계층적 스크립트는 추상화된 행위로

표현된 최상위 레벨부터 기하정보를 담고 있는 최하위 레벨까지 변환기를 통해 변환이 되어 구체화 되고 최종적으로 애니메이션에 필요한 다양한 정보들을 얻어 애니메이션이 재생된다. 아바타의 행위를 스크립트 중 사용자가 편리하게 사용할 수 있는 방식은 추상화가 이루어진 형태이다. 즉 ‘강의를 하세요’, ‘문은 여세요’ 등의 추상화된 명령만으로도 아바타가 동작 할 수 있는 것이다. 이러한 추상화된 명령을 구조적으로 나타내기 위해 계층적 스크립트 구조를 제안한다. 계층적 스크립트 구조를 따름으로써 얻어지는 장점들은 우선 사용자가 편리하게 추상화 레벨을 조절 할 수 있다는 것이다. 강의의 예를 들면 단순히 ‘여기를 가리키세요’라는 저수준 행위부터 ‘XML 강의를 하세요’와 같은 고수준의 행위 까지 편리하게 만들 수 있는 것이다. 또한 스크립트 재사용 측면에서도 유리하다. 상위 레벨의 스크립트는 정확한 기하(geometry) 정보를 담는 것이 아니라 단순히 논리적 정보들로 이루어져 있기 때문에 어플리케이션이나 하드웨어에 독립적으로 재사용할 수 있다[15]. 이 때문에 한 번 만들어진 시나리오는 필요할 때 언제든지 재사용이 가능하다. 계층적 스크립트는 최상위 레벨에서는 추상화된 행위로 표현을 하고 해석기를 통해 하위 계층으로 내려 가면서 애니메이션에 필요한 다양한 정보들을 구체화 한다.

(1) 3.2.1 Task Level Script 작업레벨 행위 스크립트

작업레벨 행위 스크립트는 최상위 레벨로써 각각의 명령들은 객체 중심적으로 설계되어 있기 때문에 어느 객체에 어느 명령을 할 것인가에 대한 정보가 들어있다. 추상화 행위는 위에서 언급했듯이 논리적 행위와 지식 기반 행위로 구분 지을 수 있다. 논리적 행위는 논리적인 판단으로 쉽게 해석이 가능한 행위 이로써 이는 객체모델에서 정의되어지기 때문에 객체 모델에서 설명하겠다. 지식 기반 행위는 사용자의 지식이 들어가 하나의 행위가 된 것으로써 시스템에서 지원하는 일반 행위들을 조합하여 완성된 하나의 행위를 생성한다. 이렇게 생성된 행위들은 그림 2에서와 같이 패키지(Package)화하여 저장되고 이는 다시 행위로 호출되어 사용할 수 있다. 이러한 패키지의 사용으로 ‘방으로 들어가기’와 같은 추상화 레벨이 낮은 행위부터 ‘강의 하기’와 같은 추상화 레벨이 높은 행위까지 쉽게 조절하여 사용할 수 있다.

작업레벨 행위 스크립트Task Level Script는 최상위 레벨로써 추상화된 명령들로 이루어져 있다. 각각의 명령들은 객체 중심적으로 설계되어 있기 때문에 어느 객체에 어느 명령을 할 것인가에 대한 정보가 들어있다. 추상화 행위는 위에서 언급했듯이 일반논리적 행위와

지식 기반 행위로 구분 지을 수 있다. 일반논리적 행위는 논리적인 판단으로 쉽게 해석이 가능한 행위 이로써 하다. 이는 이는 객체모델에서 정의되어 지기 때문에 객체 모델에서 설명하겠다. 지식 기반 행위는 사용자의 지식이 들어가 하나의 행위가 된 것으로써 시스템에서 지원하는 일반 행위들을 조합하여 완성된 하나의 행위를 생성한다. 이렇게 생성된 행위들은 그림 2에서와 같이 패키지(Package)화하여 저장되고 이는 다시 행위로 호출되어 사용할 수 있다. 이러한 패키지의 사용으로 ‘방으로 들어가기’와 같은 추상화 레벨이 낮은 행위부터 ‘강의 하기’와 같은 추상화 레벨이 높은 행위까지 쉽게 조절하여 사용할 수 있다.

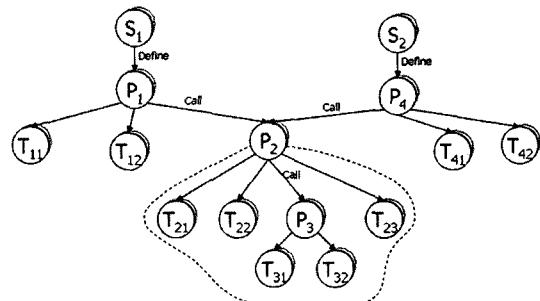


그림 2 패키지Package ((지식 기반 행위)

(2) 3.2.2 High Level Script 상위레벨 동작 스크립트

상위레벨 행위 스크립트는 위에서 설명한 행위 레벨과 같은 레벨로써 의미적으로 가장 잘게 나눌 수 있는 단위 행위이다. 단위 행위는 구체적인 위치 정보나 아바타의 관절 정보들은 필요로 하지 않고 아바타가 행동할 수 있는 논리적인 정보들만 필요로 하기 때문에 어플리케이션이나 플랫폼에 독립적으로 사용할 수 있어 스크립트의 재사용이 가능하다. 여기서 말하는 논리적인 정보들이란 공간적인 관계, 아바타 객체의 구조적 관계, 속도 등이 될 수 있다. 이러한 논리적 정보들을 그림 3과 같이 구조적으로 정의하였다. 상위레벨 동작 스크립트High Level Script는 위에서 설명한 행위 레벨과 같은 레벨로써 의미적으로 가장 잘게 나눌 수 있는 단위 행위이다. 단위 행위는 구체적인 위치 정보나 아바타의 관절 정보들은 필요로 하지 않고 아바타가 행동할 수 있는 논리적인 정보들만 필요로 하기 때문에 어플리케이션이나 플랫폼에 독립적으로 사용할 수 있어 스크립트의 재사용이 가능하다 할 수 있다. 여기서 말하는 논리적인 정보들이란 공간적인 관계, 아바타 객체의 구조적 관계, 속도 등이 될 수 있다. 이러한 논리적 정보들을 그림 3과 같이 구조적으로 정의하였다.

아바타 행위는 특정 위치로 이동하기 위한 이동 행위

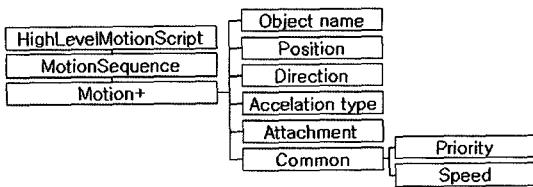


그림 3 High Level Motion Script Structure

와 그 외 이동이 없는 행위로 구분 지을 수 있다. 우선 이동에 필요한 요소들을 살펴보면 어디로 이동해야 하는지에 대한 공간적인 위치 관계가 필요한데 이를 객체 중심적으로 설계하였다. 이를 위해 <Object Name>과 <Destination Position> 그리고 <Direction>을 정의하였다. 그림 4는 공간적 요소를 논리적으로 표현한 그림이다. Object를 기준으로 ‘왼쪽 앞’부터 ‘오른쪽 뒤’까지 위치를 9가지 상대 위치를 정의하였다. 그리고 방향은 객체와 아바타의 관계를 따서 아바타가 객체를 바라보면 ‘forward’ 아바타가 객체 반대 방향이면 ‘backward’로 정의하였다.

이동이 없는 행위에서는 우선 어느 객체와 상호 관련이 있는지를 표현하기 위해 <Object Name>을 정의하였다. 여기서의 객체는 책상과 같은 객체 뿐 아니라

아바타 자신도 하나의 <object name>이 될 수 있다. 이 때문에 말을 한다거나, 감정을 표현할 때는 <object name>이 ‘avatar’ 자신이 되는 것이다. 두 번째로 고려해야 할 사항은 공간적인 위치이다. 객체와의 상호 작용이 어디서 이루어질 것인가를 표현하기 위해 <Destination Position>과 <Direction>을 정의 하였다. 이는 위에서 설명한 그림 4와 같은 구조이다. 마지막으로 상호 작용이 객체의 어느 부분에서 이루어지는가를 나타내는 <Part>를 정의 하였다. 마지막으로 공통적으로 사용된 요소들은 애니메이션의 강약을 나타내는 <Speed> <Acceleration Type>이 있고 아바타가 객체를 들거나 업을 때 필요한 <Attachment>가 있다.

(3) 3.2.3 Primitive Script 기본 동작 스크립트

Primitive Script 기본 동작 스크립트는 가장 하위 레벨의 스크립트이다. 이 스크립트에서는 3차원 가상환경에 종속적으로 스크립트가 생성된다. 기본 동작Primitive 모델은 그림 5와 같이 크게 4가지로 구분 지을 수 있다. 아바타 모션을 위해 전운동학(Forward Kinematics)과 역운동학(Inverse Kinematics)에 관련된 정보를 담는 부분과 객체의 상호작용 중 구조적 변화를 요구하는 Attach Information 정보 부분 그리고 그 외에 속도나 반복 등을 설정 할 수 있는 페리미터parameter 부분이 있다. 이러한 기본 동작 스크립트primitive script를 통해 모션 DB데이터베이스에서 모션을 가져와 합성하여 최종적인 아바타 애니메이션을 생성한다.

3.3 객체 모델Object Model

본 논문의 목표는 가상 환경 내에서 다양한 객체를 효율적으로 조작하기 위해 그림 6과 같은 객체 모델(Object Model)을 제안한다. 객체 모델을 통해 사용자는 아바타 행위 인터페이스를 제공 받을 수 있으며 이 속에 담긴 행위 논리적 규칙을 통해 작업레벨 행위 스크립트에서 상위레벨 동작 스크립트 전환이 가능하다.

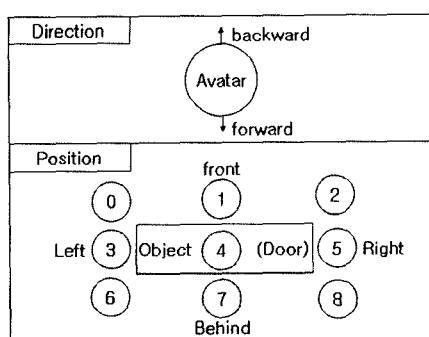


그림 4 공간 요소

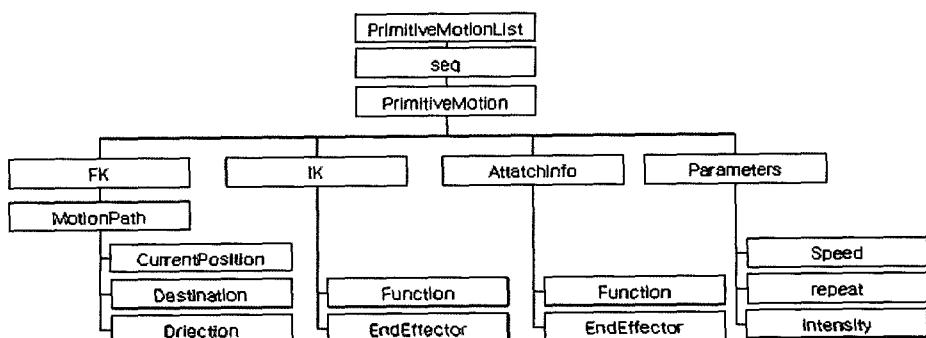


그림 5 Primitive Script Structure

이러한 처리가 가능하기 위해서는 객체 모델에서 다음과 같은 3가지 사항을 고려해야 한다.

본 논문의 목표는 가상 환경 내에서 다양한 객체를 효율적으로 조작하기 위해 그림 6과 같은 객체 모델(Object Model)을 제안한다. 객체 모델을 통해 사용자는 아바타 행위 인터페이스를 제공 받을 수 있으며 이 속에 담긴 행위 논리적 규칙rule을 통해 작업레벨 행위 스크립트Task Level Script에서 상위레벨 동작 스크립트High Level Script 전환이 가능하다. 이러한 처리가 가능하기 위해서는 객체 모델에서 다음과 같은 3가지 사항을 고려해야 한다.

- 객체의 상태 제공
- 아바타 제어를 위한 행위 인터페이스 제공
- 객체 상호 작용에 필요한 논리적 rule규칙 제시적용

제안된 객체 모델 스크립트에서는 위의 3가지 요소를 반영하여 컨텍스트, 행위 인터페이스, 행위 시퀀스로 나

누어 XML DTD를 정의하였다. 컨텍스트는 객체의 상태, 현재 객체가 속해있는 도메인 정보 및 공간 정보들을 담고 있다. 행위 인터페이스는 사용자에게 제공되는 행위 인터페이스를 구성하는 요소이며 행위 시퀀스는 객체에 행위 명령이 전달되었을 때 실제 아바타가 객체에 행하는 단위 행위들의 리스트이다. 제안된 객체 모델 스크립트에서는 위의 3가지 요소를 반영하여 컨텍스트, 행위 인터페이스, 동작 시퀀스로 나누어 XML DTD를 정의하였다. 컨텍스트는 객체의 상태, 현재 객체가 속해 있는 도메인 정보 및 공간 정보들을 담고 있다. 행위 인터페이스는 사용자에게 제공되는 행위 인터페이스를 구성하는 요소이며 동작 시퀀스는 객체에 행위 명령이 전달되었을 때 실제 아바타가 객체에 행하는 동작들의 리스트이다.

3.3.1(1) 컨텍스트(Context)

추상화된 아바타 행위들은 다양한 모호성을 만든다.

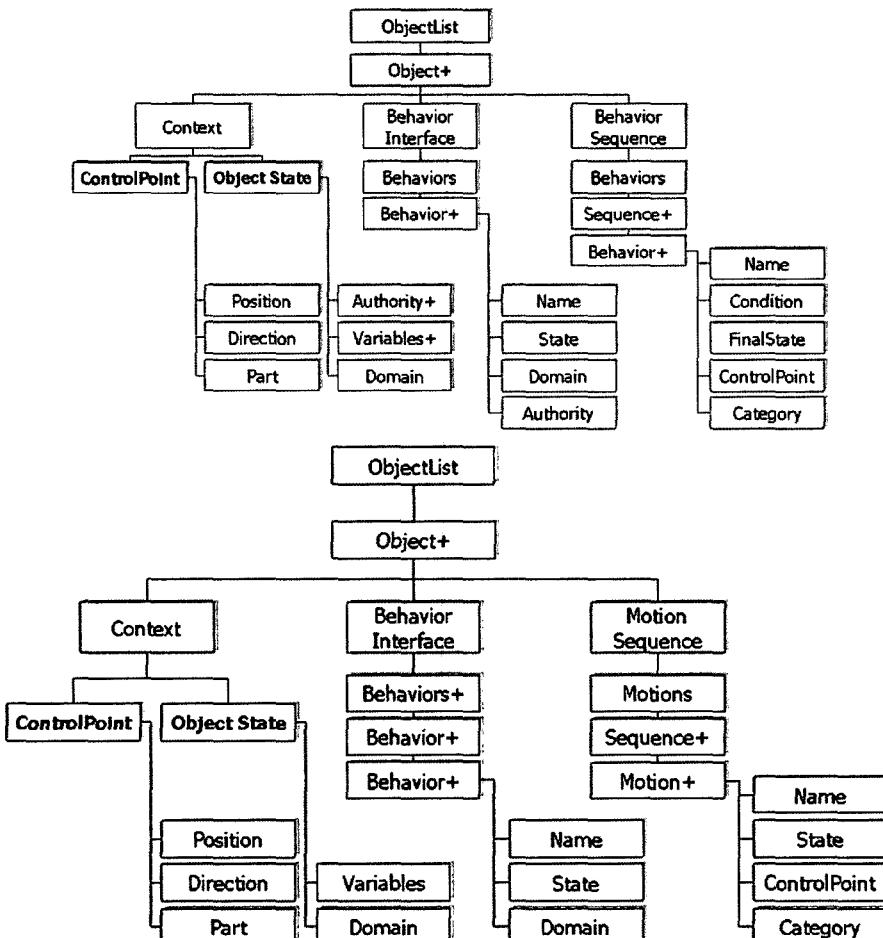


그림 6 Object Model

이러한 모호성은 크게 아바타 행위에서 오는 모호성 그리고 공간적 모호성으로 나누어 생각할 수 있다. 우선 아바타 행위에서 오는 모호성을 살펴보면, 대부분 객체의 상태와 연관되어 있다는 것을 알 수 있다. 즉, 같은 객체라도 아바타는 해당 객체의 상태에 따라서 다른 행위를 수행할 수 있기 때문이다. 예를 들어 같은 문이라도 문이 닫혀있을 경우에는 여는 행위를, 열려있으면 닫는 행위를 할 수 있다. 또한 같은 행위라도 객체의 상태에 따라서 실제 동작이 틀려질 수 있다. 예를 들어 “문으로 들어가기”라는 아바타-객체간의 행위는 문의 개폐 상태에 따라 전자의 경우 문을 열고 들어가는 동작이 생성되어야 하고 후자의 경우 문 여는 동작 없이 그냥 걸어 들어가는 동작이 생성되어야 한다. 마지막으로 객체의 물리적인 형태에 따라 밀거나 당기는 등 세부적인 동작이 달라져야 한다는 것이다.

아바타-객체 행위 표현에서는 객체 상태뿐만 아니라 공간적인 요소도 고려 해야 한다. 예를 들어 ‘탁자로 가기’라는 아바타-객체 행위 표현은 구체적으로 아바타가 어떤 방향으로 걸어가고 테이블의 어느 위치에 정확히 갈 것인가 등을 표현하고 있지 않기 때문에 이러한 것을 제어할 수 있는 공간적인 표현 요소가 필요하다.

이러한 다양한 모호성을 해결하기 위해서는 모호성을 제거할 수 있는 정보들을 객체가 가지고 있어야 한다. 이러한 객체 정보들을 컨텍스트를 통해 정의 하였다. 즉 컨텍스트는 객체의 다양한 상태들을 표현하여 그에 따라 아바타 행위 시퀀스를 변화 할 수 있도록 한 아바타-객체간의 행위를 제어하기 위해 필요한 환경 정보라고 할 수 있다.

본 논문에서는 컨텍스트를 그림 6과 같이 크게 2가지로 구분하였는데, 첫 번째는 객체 상태 컨텍스트로서 객체의 상태 변수와 도메인 타입을 설정하기 위한 요소로 구성되어 있다. 두 번째는 제어점 컨텍스트로서 아바타-객체 행위 제어 시 필요한 공간 정보들, 즉 아바타의 위치, 방향 및 조작 부위 점 등을 표현한다.

추상화된 아바타 행위들은 다양한 모호성을 만든다. 이러한 모호성은 크게 아바타 행위에서 오는 모호성 그리고 공간적 모호성으로 나누어 생각할 수 있다. 우선 아바타 행위에서 오는 모호성을 살펴보면, 대부분 객체의 상태와 연관되어 있다는 것을 알 수 있다. 즉, 같은 객체라도 아바타는 해당 객체의 상태에 따라서 다른 행위를 수행할 수 있기 때문이다. 예를 들어 같은 문이라도 문이 닫혀있을 경우에는 여는 행위를, 열려있으면 닫는 행위를 할 수 있다. 또한 같은 행위라도 객체의 상태에 따라서 실제 동작이 틀려질 수 있다. 예를 들어 “Enter the door”라는 아바타-객체간의 행위는 문의 개폐 상태에 따라 전자의 경우 문을 열고 들어가는 동작

이 생성되어야 하고 후자의 경우 문 여는 동작 없이 그냥 걸어 들어가는 동작이 생성되어야 한다. 마지막으로 객체의 물리적인 형태에 따라 밀거나 당기는 등 세부적인 동작이 달라져야 한다는 것이다.

아바타-객체 행위 표현에서는 객체 상태뿐만 아니라 공간적인 요소도 고려 해야 한다. 예를 들어 ‘go to the table’이란 아바타-객체 행위 표현은 구체적으로 아바타가 어떤 방향으로 걸어가고 테이블의 어느 위치에 정확히 갈 것인가 등을 표현하고 있지 않기 때문에 이러한 것을 제어할 수 있는 공간적인 표현 요소가 필요하다.

이러한 다양한 모호성을 해결하기 위해서는 모호성을 제거할 수 있는 정보들을 객체가 가지고 있어야 한다. 이러한 객체 정보들을 컨텍스트를 통해 정의 하였다. 즉 컨텍스트는 객체의 다양한 상태들을 표시하여 그에 따라 아바타 행위 시퀀스를 변화 할 수 있도록 한 아바타-객체간의 행위를 제어하기 위해 필요한 환경 정보라고 할 수 있다.

본 논문에서는 컨텍스트를 그림 6과 같이 크게 2가지로 구분하였는데, 첫 번째는 객체 상태 컨텍스트로서 객체의 상태 변수와 도메인 타입을 설정하기 위한 요소로 구성되어 있다. 두 번째는 제어점 컨텍스트로서 아바타-객체 행위 제어 시 필요한 공간 정보들, 즉 아바타의 위치, 방향 및 조작 부위 점 등을 표현한다.

(1) 객체 상태 컨텍스트

객체 상태 컨텍스트는 객체가 가질 수 있는 다양한 상태들을 저장하는 곳이다. 객체의 종류에 따라 다양한 상태들이 있을 수 있으므로 상태 변수는 사용자 정의에 의해 변수명과 값으로 정의되고 각각의 객체는 이를 변수리스트를 가지고 상태에 따라 그 값이 반영된다. 이러한 상태 컨텍스트들을 통해 객체의 상태를 판단하여 사용자에게 현재 상태에 적합한 아바타-객체 행위를 제공하게 된다. 예를 들어 문 객체에 ‘Door_Open’이란 변수를 하나 생성한다. 문 객체에 여러 아바타-객체 행위가 있겠지만 그 중 대표적으로 문을 열고 닫는 것을 생각해보겠다. ‘Door_Open’이란 변수가 참일 때 문은 사용자에게 ‘닫기’란 행위를 제공하여 문을 닫게 할 수 있게 하고 반대일 경우에는 ‘열기’란 행위를 제공하여 문을 열 수 있게 하는 것이다. 또한 상태 변수는 행위를 구성하는 행위 시퀀스에도 영향을 미칠 수 있다. 위의 예와 같이 ‘열기’라는 명령을 내려보겠다. 하지만 아바타의 위치에 따라 행위가 바뀔 수 있기 때문에 아바타 위치 상태를 나타내는 ‘AvatarPos’란 변수를 생성한다. 이는 아바타가 문과 상호작용하기 위해 필요한 정 위치에 서 있는가를 체크하는 변수로서, 아바타가 상태변수에 정의된 위치에 있지 않을 때에는 문까지 걸어와서 열게 되

고 만약 변수에 정의된 위치에 있을 때는 걷는 동작 없이 문을 여는 동작을 행하게 된다.

객체 상태 컨텍스트는 객체가 가질 수 있는 다양한 상태들을 저장하는 곳이다. 객체의 종류에 따라 다양한 상태들이 있을 수 있으므로 상태 변수는 사용자 정의의 의해 변수명과 값으로 정의되고 각각의 객체는 이를 변수 리스트를 가지고 상태에 따라 그 값이 반영된다. 이러한 상태 컨텍스트들을 통해 객체의 상태를 판단하여 사용자에게 현재 상태에 적합한 아바타-객체 행위를 제공하게 된다. 예를 들어 Door 객체에 'Door_Open'이란 변수를 하나 생성한다. Door문 객체에 여러 아바타-객체 행위가 있겠지만 그 중 대표적으로 문을 열고 닫는 것을 생각해보겠다. 'Door_Open'이란 변수가 참일 때 Door문은 사용자에게 'Close'란 행위를 제공하여 문을 닫게 할 수 있게 하고 반대일 경우에는 'Open'이란 행위를 제공하여 문을 열 수 있게 하는 것이다. 또한 상태 변수는 행위를 구성하는 동작 시퀀스에도 영향을 미칠 수 있다. 위의 예와 같이 'Open'이라는 명령을 내려보겠다. 하지만 아바타의 위치에 따라 행위가 바뀔 수 있기 때문에 아바타 위치 상태를 나타내는 'AvatarPos' 란 변수를 생성한다. 이는 아바타가 문과 상호작용하기 위해 필요한 정 위치에 서 있는가를 체크하는 변수로서, 아바타가 상태변수에 정의된 위치에 있지 않을 때에는 문까지 걸어와서 열게 되고 만약 변수에 정의된 위치에 있을 때는 걷는 동작 없이 문을 여는 동작을 행하게 된다.

객체 상태는 상태변수 외에 도메인 타입을 표현할 수 있다. 이는 같은 객체라도 도메인이 달라지면 제공하는 행위 인터페이스가 변할 수 있기 때문에 필요한 요소이다. 예를 들어 자동차라는 객체가 있을 때 시뮬레이션 도메인에서는 운전과 같은 행위를 제공할 수 있지만, 정비 도메인에서는 조립/분해와 같은 행위를 제공해 주어야 하기 때문이다. 즉 도메인에 따라서 같은 객체라도 다른 행위를 제공해 줄 수 있는 것이다.

(2) 제어점 컨텍스트

공간 컨텍스트는 객체 주변의 임의의 제어점들로 정의되며 아바타가 위치하여 객체를 조작하게 하는 지점이다. 객체가 가지는 행위와 구조에 따라서 제어점은 다수가 될 수 있으며 각 제어점은 3가지 요소(Element)를 가진다. 이들 요소는 아바타의 위치(Position) 및 방향(Direction), 그리고 객체에서 아바타의 조작이 행해지는 부분(Part)으로 구성되어 있다. 이는 상위레벨 행위 스크립트 부분에서 설명하였다.

공간 컨텍스트는 객체 주변의 임의의 제어점들로 정의되며 아바타가 위치하여 객체를 조작하게 하는 지점이다. 객체가 가지는 행위와 구조에 따라서 제어점은 다

수가 될 수 있으며 각 제어점은 3가지 요소(Element)를 가진다. 이들 요소는 아바타의 위치(Position) 및 방향(Direction), 그리고 객체에서 아바타의 조작이 행해지는 부분(Part)으로 구성되어 있다. 이는 High Level Script 부분에서 설명하였다.

3.3.2(2) Behavior Interface 행위 인터페이스

제안 행위 모델에서는 사용자에게 행위 인터페이스를 제공하기 위해 <BehaviorInterface>를 정의하였다. 객체는 사용 가능한 행위 리스트를 가지고 있으며, 또한 각 행위는 아바타와 객체의 동작 시퀀스로 구성되어 하나의 행위를 수행하기 위한 동작들을 생성한다. 이러한 행위 인터페이스는 사용자가 아바타 행위 제어를 위해 객체에 접근할 때 보여지는 행위들로 구성된다. 행위 인터페이스는 객체 상태 컨텍스트를 속성 값(Parameter)으로 가지는 행위 명으로 구성되며 이중 현재 컨텍스트와 일치하는 속성 값을 가진 행위를 사용자에게 선택적으로 제공한다. 예를 들어 그림 7과 같이 객체에 'Door_Closed'이란 상태 변수의 값에 따라 제공되는 인터페이스가 달라지게 되는데, 변수 상태가 'v1(false)'이면 '열기'이라는 행위 인터페이스는 사라지는 대신 '닫기'라는 인터페이스가 생겨나게 되고 'v2(true)'면 반대의 인터페이스를 제공하는 것이다. 강의 도메인 환경에서도 이러한 것은 적용이 가능한데 강의 화면 페이지가 마지막이 되면 'Next Page'라는 행위 인터페이스가 사라지게 된다. 이렇게 함으로서 사용자는 가상 환경의 상태를 고려할 것 없이 그 때 그 때 가능한 행위 리스트들만 제공받기 때문에 시나리오 생성이 좀 더 쉽다. 제안 행위 모델에서는 사용자에게 행위 인터페이스를 제공하기 위해 객체는 사용 가능한 행위 리스트를 가지고 있으며, 또한 각 행위는 아바타와 객체의 동작 시퀀스로 구성되어 하나의 행위를 수행하기 위한 동작들을 생성한다. 이러한 행위 인터페이스는 사용자가 아바타 행위 제어를 위해 객체에 접근할 때 보여지는 행위들로 구성된다. 행위 인터페이스는 객체 상태 컨텍스트를 패러미터로 가지는 행위명으로 구성되며 이중 현재 컨텍스트와 일치하는 패러미터를 가진 행위를 사용자에게 선택적으로 제공한다.

그림에서 보듯이 행위 엘리먼트들은 앞서 설명한 컨텍스트 엘리먼트에서 정의된 상태 변수 및 도메인 정보를 패러미터로 받고 이는 행위 선택 모듈에서 처리되어 사용 가능한 행위 리스트들이 사용자에게 제공되는 것이다.

3.3.3(3) Motion Sequence 동작 시퀀스

마지막으로 행위 시퀀스(Behavior Sequence) 요소에 대해 알아보겠다. 추상화된 행위는 다수의 단위 행위로 쪼개어질 수 있는데. 예를 들어 닫혀있는 문에 '문으로 들어가기'의 행위를 지시하면 아바타는 '걷기(walk)', 문 열기(open), 걷기(turn), 문 닫기(close)'의 단위 행

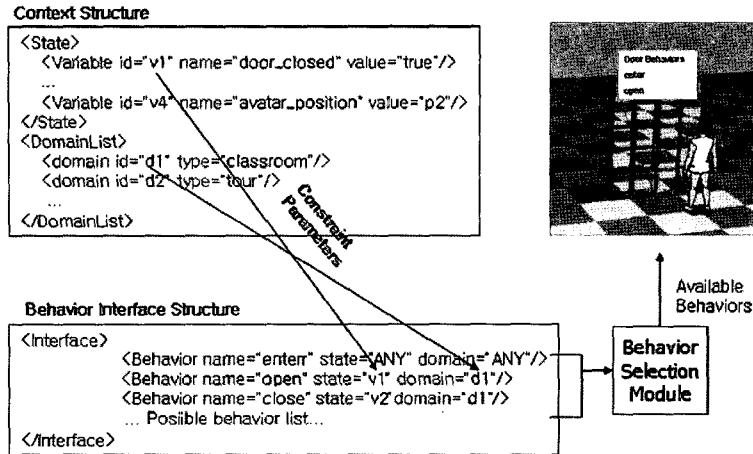


그림 7 the Example of Behavior Interface

위 시퀀스로 움직이게 된다. 시퀀스 엘리먼트에는 정의된 행위를 수행하기 위한 이와 같은 단위 행위들의 순차적인 리스트로 구성되어 있다. 그러나 문이 열려있는 경우 아바타는 문을 여는 동작을 생략하고 걸어 들어가게 되는데, 이는 앞서 정의한 객체 상태 컨텍스트에 따라 이를 판단함으로써 해결할 수 있다. 행위 시퀀스의 경우 행위 인터페이스와 유사하게 컨텍스트에서 정의된 제어점과 상태 변수들을 패러미터로 받아 단위 행위 선택 모듈에서 이를 처리하여 현재 상황에 적합한 단위 행위 시퀀스를 생성하여 아바타를 제어하게 된다. 마지막으로 동작 시퀀스(Motion Sequence) 요소에 대해 알아보겠다. 행위는 다수의 동작으로 구성되는데, 예를 들어 닫혀 있는 문에 'enter the door'의 행위를 지시하면 아바타는 'walk, grab knob, pull knobopen, walk, turn, grab

knob, push knobclose'의 동작 시퀀스로 움직이게 된다. 시퀀스 엘리먼트에는 정의된 행위를 수행하기 위한 이와 같은 동작들의 순차적인 리스트로 구성되어 있다. 그러나 문이 열려있는 경우 아바타는 문을 여는 동작을 생략하고 걸어 들어가게 되는데, 이는 앞서 정의한 객체 상태 컨텍스트에 따라 이를 판단할 수 있도록 각 동작 요소들은 상태 속성을 가지고 있기 때문입니다. 또한 객체를 조작하기 위한 제어점에서 아바타가 올바로 동작하도록 제어점 속성을 가지고 있다. 동작 시퀀스의 경우도 행위 인터페이스와 유사하게 컨텍스트에서 정의된 제어점과 상태 변수들을 패러미터로 받아 동작 선택 모듈에서 이를 처리하여 현재 상황에 적합한 동작 시퀀스를 생성하여 아바타를 제어하게 된다.

그림 8은 아바타와 객체의 상태에 따라 아바타 행위

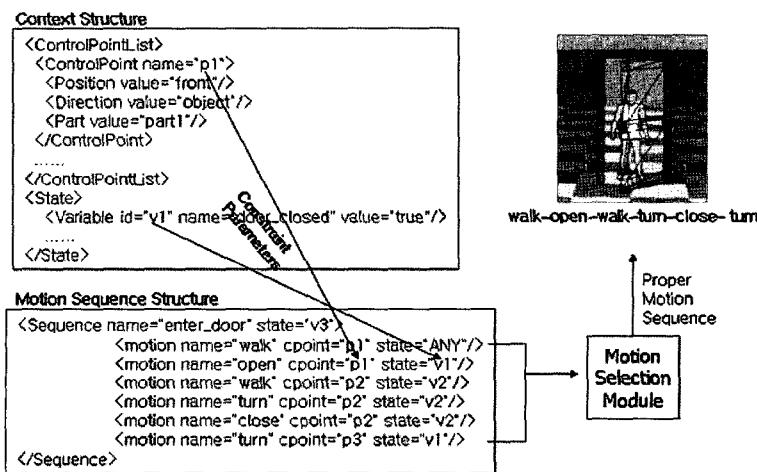


그림 8 상태에 따른 아바타 행위 시퀀스

가 어떻게 변화하는지를 보여주는 그림이다. 우선 기본적으로 아바타가 방 안으로 들어가기 위해 필요한 행위 시퀀스는 ‘걷기-문열기-걷기-돌기-문닫기’라 할 수 있다. 하지만 아바타나 객체의 상태에 따라 이러한 행위 시퀀스에 변화가 생긴다. 즉 문의 상태가 열려있으면 ‘문열기’라는 행위가 필요 없게 되고 ‘걷기’-‘걷기’는 하나의 ‘걷기’로 통합이 되어 최종적으로는 ‘걷기-돌기-문닫기’라는 행위 시퀀스를 생성하는 것이다.

행위 시퀀스를 생성함에 있어서 중요한 점은 행위 시퀀스의 재사용성과 아바타 객체간의 상호작용을 고려해야 한다는 것이다. 아바타 행위의 재사용성을 높이기 위해 행위 시퀀스 안에 미리 정의된 행위를 재귀적으로 재사용 할 수 있도록 설계하였다. 이는 행위 분할 방식을 통해 해결하였는데 아바타 행위는 더 작은 행위로 분할이 가능하고 이는 최종적으로 기본 단위 행위 시퀀스로 분할이 된다는 것이다. 이러한 행위의 재사용성은 객체 모델을 정의할 때 좀 더 빠르게 생성 할 수 있는 장점이 있다. 예를 들어 “컴퓨터 켜기”라는 명령을 내렸을 때 아바타는 방 밖에 있고 컴퓨터는 방 안에 있었다고 가정하자. 이런 행위를 위한 동작 시퀀스를 생성하기 위해서는 방 안을 들어가는 부분이 들어가야 하는데 이럴 때 미리 정의된 ‘(방으로)들어가기’이라는 행위를 불러와 재사용 하는 것이다. 즉, ‘(방으로)들어가기-(컴퓨터로)가기-(컴퓨터)켜기’의 형태로 간략하게 명령을 내릴 수 있는 것이다.

아바타가 하나님의 행위를 완성하기 위해서는 아바타 스스로의 행동과 더불어 다양한 객체들간의 상호작용을 고려해야 한다. 예를 들어 프리젠테이션 스크린과 컴퓨터가 있을 때 아바타가 화면의 페이지를 넘기는 명령을 내린다고 가정하자. 사용자가 칠판에 ‘다음 페이지’라는 명령을 줄 것이고 이에 따라 다음과 같은 행위 시퀀스가 이루어져야 한다. 우선 아바타는 컴퓨터까지 가서 키보드 버튼을 눌러야 하고 그 다음에 프리젠테이션 페이지가 넘어가야 하고 최종적으로 다시 칠판까지 걸어오는 행위 시퀀스가 생성되어야 하는 것이다. 하지만 여기서 문제점은 아바타와 컴퓨터간의 상호작용뿐만 아니라 컴퓨터와 프리젠테이션 스크린과의 상호작용도 존재한다는 것이다. 이를 위해 객체상 상호 통신이 가능하도록 객체 행위 인터페이스를 정의하였다. 그림 9는 이러한 상호 작용의 예를 표현한 그림이다. 그림 8은 아바타와 객체의 상태에 따라 아바타 행위가 어떻게 변화하는지를 보여주는 표이다. 우선 기본적으로 아바타가 방 안으로 들어가기 위해 필요한 동작 시퀀스는 walk-open-walk-turn-close라 할 수 있다. 하지만 아바타나 객체의 상태에 따라 이러한 동작 시퀀스에 변화가 생긴다. 즉 문의 상태가 열려있으면 open이라는 행위가 필요 없

게 되고 walk-walk는 하나의 walk로 통합이 되어 최종적으로는 walk-turn-close라는 동작 시퀀스 motion sequence를 생성하는 것이다.

동작 시퀀스를 생성함에 있어서 중요한 점은 동작 시퀀스의 재사용성과 아바타 객체간의 상호작용을 고려해야 한다는 것이다. 아바타 행위의 재사용성을 높이기 위해 동작 시퀀스 안에 미리 정의된 행위를 재귀적으로 재사용 할 수 있도록 설계하였다. 이는 행위 분할 방식을 통해 해결하였는데 아바타 행위는 작은 단위의 행위로 분할이 가능하고 이는 최종적으로 행위의 기본 단위인 동작 시퀀스로 분할이 된다는 것이다. 이러한 행위의 재사용성은 객체 모델을 정의할 때 좀 더 빠르게 생성 할 수 있는 장점이 있다. 예를 들어 “turn on the computer”라는 명령을 내렸을 때 아바타는 방 밖에 있고 컴퓨터가는 방 안에 있었다고 가정하자. 이런 행위를 위한 동작 시퀀스를 생성하기 위해서는 방 안을 들어가는 부분이 들어가야 하는데 이럴 때 미리 정의된 ‘enter the room’이라는 행위behavior를 불러와 재사용 하는 것이다. 즉, enter the room - walk - turn on의 형태로 간략하게 동작 시퀀스를 생성할 수 있는 것이다.

아바타가 하나님의 행위를 완성하기 위해서는 아바타 스스로의 행동과 더불어 다양한 객체들간의 상호작용을 고려해야 한다. 예를 들어 프레젠테이션프리젠테이션 스크린과 컴퓨터가 있을 때 아바타가 화면의 페이지를 넘기는 명령을 내린다고 가정하자. 사용자가 칠판에 ‘next page’라는 명령을 줄 것이고 이에 따라 다음과 같은 동작 시퀀스가 이루어져야 한다. 우선 아바타는 컴퓨터까지 가서 키보드 버튼을 눌러야 하고 그 다음에 프레젠테이션프리젠테이션 페이지가 넘어가야 하고 최종적으로 다시 칠판까지 걸어오는 동작 시퀀스가 생성되어야 하는 것이다. 하지만 여기서 문제점은 아바타와 컴퓨터간의 상호작용뿐만 아니라 컴퓨터와 프레젠테이션프리젠테이션 스크린과의 상호작용도 존재한다는 것이다. 이를 위해 객체상 상호 통신이 가능하도록 객체 행위 인터페이스를 정의하였다. 그림 9는 이러한 상호 작용의 예를 표현한 그림이다.

우선 페이지를 넘기기 위해 아바타가 컴퓨터를 조작하는 행위를 호출하게 되고 컴퓨터는 다시 스크린에 페이지가 넘어가는 행위를 호출하게 된다. 이런 방식으로 객체 행위 인터페이스를 통해 행위가 분할되고 또 호출되는 방식으로 아바타 객체간 상호작용을 해결 할 수 있다.

우선 페이지를 넘기기 위해 아바타가 컴퓨터를 조작하는 행위를 호출하게 되고 컴퓨터는 다시 스크린에 페이지가 넘어가는 행위를 호출하게 된다. 이런 방식으로 객체 행위 인터페이스를 통해 행위가 분할되고 또 호출되는 방식으로 아바타 객체간 상호작용을 해결 할 수 있다.

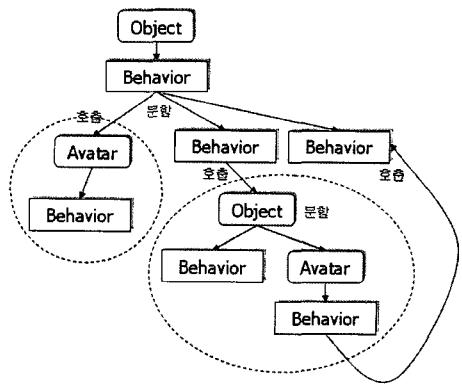


그림 9 객체 행위 인터페이스를 통한 행위 분할 및 호출

3.4 외부 Contents컨텐츠콘텐츠

객체 모델의 문제점은 객체 모델만으로는 사용자에게 다양한 정보 전달이 어렵다는 것이다. 즉, 강의를 할 때 행위는 한정되어 있지만 근본적으로 강의 내용은 항상 바뀐다. 뿐만 아니라 기상 캐스팅의 경우도 시간의 변화에 따라 기상 정보 내용도 끊임 없이 변화된다. 이러한 다양한 정보 전달을 위해서는 각각의 정보들을 담고 있는 구조화된 문서가 필요하다. 이러한 구조화된 외부 정보 문서를 본 논문에서는 콘텐츠(Contents)라고 정의하였다. 객체 모델의 문제점은 객체 모델만으로는 사용자에게 다양한 정보 전달이 어렵다는 것이다. 즉, 강의를 할 때 행위는 한정되어 있지만 근본적으로 강의 내용은 항상 바뀐다. 뿐만 아니라 기상 캐스팅의 경우도 시간의 변화에 따라 기상 정보 내용도 끊임 없이 변화된다. 이러한 다양한 정보 전달을 위해서는 각각의 정보들을 담고 있는 구조화된 문서가 필요하다.

이러한 구조화된 외부 정보 문서를 본 논문에서는 콘텐츠(Contents)라고 정의하였다.

구조화된 정보를 바로 객체에 적용함으로써 사용자에게 직관적인 인터페이스를 제공해 줄 뿐만 아니라 콘텐츠 속의 개체(Entity) 구조들을 객체 구조와 동일하게

적용함으로써 객체를 좀 더 효율적으로 관리할 수 있다. 본 논문에서는 교재 콘텐츠 EBKS 1.1 DTD subset을 콘텐츠로 사용하였는데 여기서는 <그림 10>과 같이 Book, Title, Paragraph 등을 객체화 하였고 객체간의 구조 관계를 그대로 가져왔기 때문에 객체를 효율적으로 관리 할 수 있다. 즉, 사용자는 책, 페이지 단위로 객체를 그룹 지어 조작할 수 있는 것이다. 또한 전자 책의 내용이 바뀌어도 시스템이 돌아가기 때문에 다양한 내용으로 강의를 생성할 수 있다.

구조화된 정보를 바로 객체에 적용함으로써 사용자에게 직관적인 인터페이스를 제공해 줄 뿐만 아니라 콘텐츠컨텐츠콘텐츠 속의 개체(entity) 구조들을 객체 구조와 동일하게 적용함으로써 Object객체를 좀 더 효율적으로 관리할 수 있다. 본 논문에서는 교재 콘텐츠컨텐츠콘텐츠 EBKS 1.1 DTD subset을 콘텐츠컨텐츠콘텐츠로 사용하였는데 여기서는 그림 10과 같이 Book, Title, Paragraph 등을 객체화 하였고 객체간의 구조 관계를 그대로 가져왔기 때문에 객체를 효율적으로 관리 할 수 있다. 즉, 사용자는 책, 페이지 단위로 객체를 그룹 지어 조작할 수 있는 것이다. 또한 전자 책의 내용이 바뀌어도 시스템이 돌아가기 때문에 다양한 내용으로 강의를 생성할 수 있다.

3.5 컨텍스트 메뉴

Context Menu

기존의 연구에서는 스크립트 생성은 직접 손으로 작성해야만 하는 번거로움이 있었다. 사용자가 직접 작성 을 할 경우 시간이 많이 소요될 뿐 아니라 잘못된 스크립트로 오류가 날 가능성이 있기 때문에 문제점이 있었다. 이러한 문제점을 해결하기 위해 본 논문에서는 컨텍스트 메뉴를 제공한다. 즉 사용자가 스크립트를 직접 작성하는 것이 아니라 가상 환경 내에서 객체 모델에 정의한 인터페이스를 통해 스크립트를 작성하는 것이다. 객체 모델에는 객체 상태들을 기록하는 컨텍스트와 인터페이스를 제공한다. 행위 인터페이스는 객체 상태 변

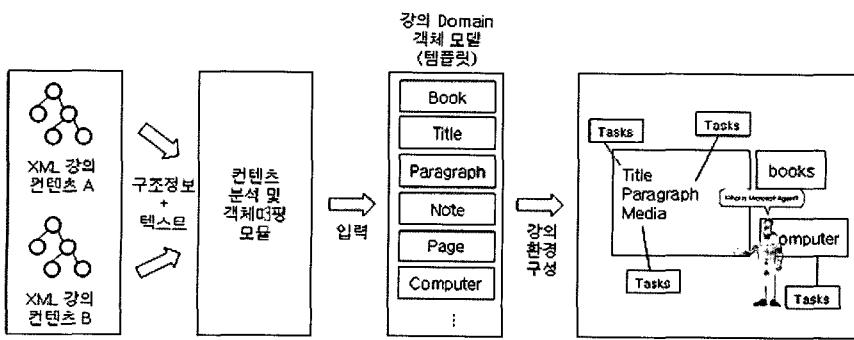


그림 10 컨텐츠의 시스템 적용

수의 값에 따라 할 수 있는 행위와 할 수 없는 행위로 구분이 되고 사용자에게 할 수 있는 행위만 컨텍스트 메뉴를 통해 보여주는 것이다. 사용자는 객체의 상태를 고려하지 않고 보여지는 데로 행위를 선택할 수 있기 때문에 직관적이고 편리하게 사용이 되며 스크립트 오류 역시 막을 수 있다. 선택된 행위는 정해진 규칙에 따라 단위 행위 시퀀스로 변환되어 되고 이는 바로 화면에 애니메이션으로 보여주기 때문에 스크립트가 쓰여지는 동시에 애니메이션을 확인 할 수 있어 직관적으로 시나리오를 생성 할 수 있다. 그럼 11은 교육 환경에서 페이지의 상태에 따른 컨텍스트 메뉴의 변화를 보여주고 있다. 강의 콘텐츠는 정해진 페이지와 줄을 가지게 되는데 평상시에는 'Prev'가 가능하지만 마지막 페이지가 되었을 때는 더 이상 페이지를 넘길 수 없기 때문에 'Prev'라는 명령이 사라진 것을 볼 수 있다. 이렇게 상태에 따라 할 수 있는 행위들만 보여주기 때문에 사용자는 객체의 상태를 고려할 필요가 없고 스크립트 작성 역시 객체 상태에 따른 모호성을 미리 막아 줄 수 있다. 기존의 연구에서는 스크립트 생성은 직접 손으로 작성해야만 하는 번거로움이 있었다. 사용자가 직접 작성할 경우 시간이 많이 소요될 뿐 아니라 잘못된 스크립트로 오류가 날 가능성이 있기 때문에 문제점이 있었다. 이러한 문제점을 해결하기 위해 본 논문에서는 컨텍스트 메뉴를 제공한다. 즉 사용자가 스크립트를 직접 작성하는 것이 아니라 객체 모델에 정의한 인터페이스를 통해 스크립트를 작성하는 것이다. 객체 모델에는 객체 상태들을

을 기록하는 컨텍스트와 인터페이스를 제공한다. 행위 인터페이스는 객체 상태 변수의 값에 따라 할 수 있는 행위와 할 수 없는 행위로 구분이 되고 사용자에게 할 수 있는 행위만 컨텍스트 메뉴를 통해 보여주는 것이다. 사용자는 객체의 상태를 고려하지 않고 보여지는 데로 행위를 선택할 수 있기 때문에 직관적이고 편리하게 사용이 되면며 스크립트 오류 역시 막을 수 있다. 선택된 행위는 정해진 규칙에 따라 동작 시퀀스로 변환되어 되고 이는 바로 화면에 애니메이션으로 보여주기 때문에 스크립트가 쓰여지는 동시에 애니메이션을 확인 할 수 있어 직관적으로 시나리오를 생성 할 수 있다. 그럼 11은 교육 환경에서 페이지의 상태에 따른 컨텍스트 메뉴의 변화를 보여주고 있다. 강의 콘텐츠콘텐츠콘텐츠는 정해진 페이지와 줄을 가지게 되는데 평상시에는 'nextPrev page'가 가능하지만 마지막 페이지가 되었을 때는 더 이상 페이지를 넘길 수 없기 때문에 'Prevnext page'라는 명령이 사라진 것을 볼 수 있다. 이렇게 상태에 따라 할 수 있는 행위들만 보여주기 때문에 사용자는 객체의 상태를 고려할 필요가 없고 스크립트 작성 역시 객체 상태에 따른 모호성을 미리 막아 줄 수 있다.

3.5.6 스크립트 해석기

1) 3.6.1 작업레벨 행위 스크립트 Task Level Script to High Level Script 해석기

작업 레벨의 스크립트는 해석기를 통해 상위 레벨 스크립트로 변환을 하게 된다. 작업레벨 행위 해석기는 추상화된 행위를 단위 행위 시퀀스로 변환을 해주는 역할을

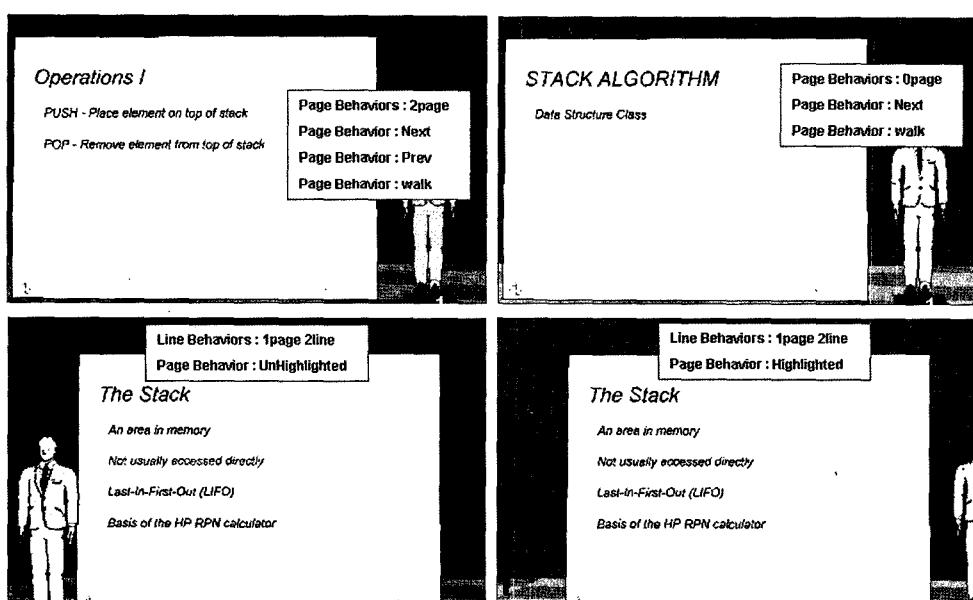


그림 11 door 상태에 따른 context menu

한다. 이는 객체 모델을 이용하여 변환하게 되는데 객체 모델에 객체 상태와 행위 인터페이스에 따른 동작 시퀀스가 미리 정의되어 있기 때문에 쉽게 변환이 가능하다.

상위작업 레벨의 스크립트는 해석기를 통해 하위상위 레벨 스크립트로 변환을 하게 된다. 그림 12와 같이 Task to High 작업레벨 행위 해석기는 추상화된 행위를 단위인 동작행위 시퀀스로 변환을 해주는 역할을 한다. 이는 객체 모델을 이용하여 변환하게 되는데 객체 모델에 객체 상태와 행위 인터페이스에 따른 동작 시퀀스가 미리 정의되어 있기 때문에 쉽게 변환이 가능하다.

2) 3.6.2 High Level Script 해석기 to Primitive Level Script 상위레벨 동작 스크립트 해석기

플랫폼 독립적인 상위레벨 동작 스크립트를 플랫폼 종속적인 기본동작 스크립트로 변환하기 위해서는 3차원 가상 환경의 기하학적인 정보와 애니메이션 렌더링 정보가 필요하다. 이러한 정보들을 관리하기 위해서 그림 12와 같이 환경 정보 스크립트를 정의하였다. 환경 정보는 객체 자체에 대한 정보를 나타내는 <Object>와 가상환경 내 공간 정보를 나타내는 <Position>, <Boundary>, <PositionSet> 부분으로 나누어 정의하였다. 이러한 정보들은 아바타와 객체간 상호 작용이 정확이 어느 부분에서 일어나는지를 제공해준다.

환경 정보를 이용하여 상위레벨 동작 스크립트를 기본 동작 스크립트로 변환 할 수 있다. 이러한 변환 과정을 그림 13과 같다.

변환을 위해 우선 행위 타입을 참조한다. 행위 타입은 크게 4가지로 구분되는데 이동 행위, 아바타 단독 행위, 객체가 붙는 행위, 아바타-객체 행위가 그것이다. 첫 번째, 이 행위가 이동 행위이면 환경 정보를 이용하여 이동 경로를 탐색하게 된다. 환경 정보에 있는 객체 위치 정보 그리고 아바타와 객체의 상대적 위치인 <PositionSet>을 이용하여 정확히 어느 위치로 이동해야 하는가가 계산되며 그 중 <Boundary> 정보를 이용하여

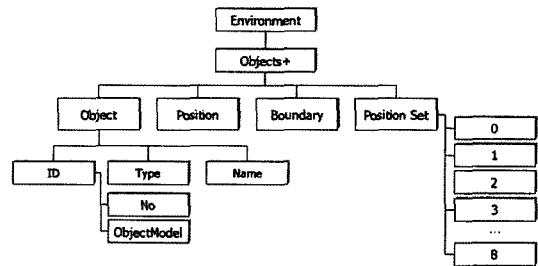


그림 13 환경 정보 XML 구조

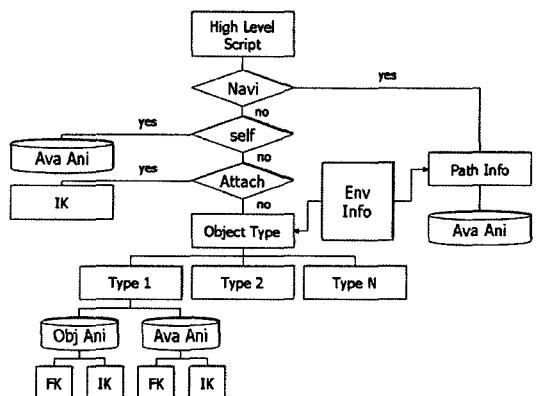


그림 14 상위레벨 동작 스크립트 해석기 알고리즘

객체를 어떻게 피해서 이동할 것인지도 함께 계산 된다. 두 번째, 이 행위가 아바타 단독 행위이면 이것은 객체와 무관한 행위이기 때문에 단순히 아바타 모션만 불러와 행위를 생성한다. 세 번째, 이 행위가 아바타에 객체를 붙이는 행위이면 이는 아바타의 어느 부위에 붙을 것인지를 계산하고 객체를 원하는 지점으로 이동시킨 후 3차원 객체간 계층적 구조를 변화시킨다.

플랫폼 독립적인 High Level Script 상위레벨 동작 스크립트를 플랫폼 종속적인 기본동작 스크립트 Primitive Level Script로 변환 하기 위해서는 3차원 가상 환

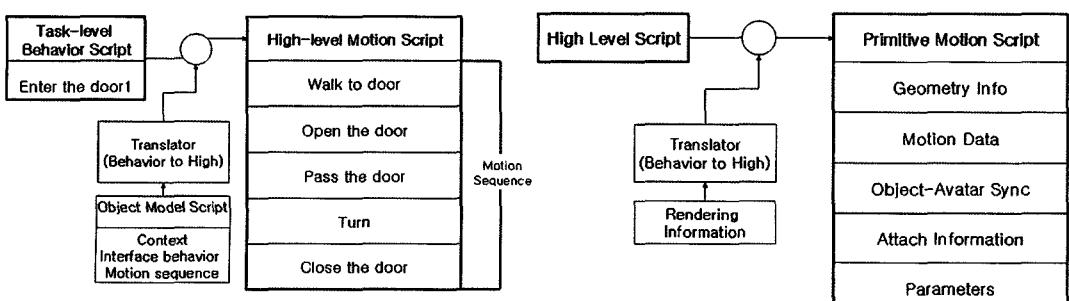


그림 12 제안 해석기의 동작구조

경의 기하학적인 정보와 애니메이션 렌더링 정보가 필요하다. 각각의 모션들과 아바타 애니메이션 렌더링 정보를 적용함으로써 애니메이션을 생성 할 수 있다. 여기서 모션의 종류는 객체 동작(object motion)과 아바타 동작(avatar motion)으로 구분이 된다. 아바타와 객체간 상호 작용은 이벤트 방식으로 처리가 된다. 즉 아바타가 상호작용할 객체의 특정 위치에 도달하게 되면 이벤트가 발생하여 객체와 아바타가 상호작용하게 된다. 이때 아바타 모션의 처리 방식은 전운동학 Forward Kinematics 방식과 Inverse Kinematics 역운동학 방식으로 처리할 수 있도록 설계하였고 동기화(sync 부분은 Object-Avatar Sync 부분을 이용하여 처리한다.

아바타 모션을 적용함에 있어 고려해야 할 사항이 있는데 그것은 바로 객체의 종류이다. 객체의 종류에 따라 위치, 동작, 시간 등 물리적 특성이 변하기 때문에 이에 따른 Primitive Motion Script 기본 동작 스크립트의 재구성이 이루어져야 한다. 즉, 객체의 물리적 구조에 따라 각각 다른 기본 동작 스크립트가 적용되어야 한다. 그럼 13은 사용자가 동일한 'enter' 행위를 선택했을 때 아바타 행위가 변화하는 것을 보여주는 그림이다. 문의 종류에 따라 아바타가 다른 동작을 수행하는 것을 볼 수 있다. 미닫이 문에서는 아바타가 문을 옆으로 미는 동작을 보여주며, 일반 문은 잡아당기면서 'enter' 명령을 상황에 맞게 실행하고 있다.

마지막으로 행위가 아바타-객체 행위일 경우 아바타, 객체 모두 모션을 적용해야 하는데, 이러한 모션은 객체의 종류에 따라 달라지기 때문에 이를 고려해주어야 한다. 즉, 위에서 설명하였듯이 문의 경우 문의 종류에 따라 위치, 동작, 시간 등 물리적 특성이 변하기 때문에 이에 따른 기본 동작 스크립트의 재구성이 이루어진다. 즉, 객체의 물리적 구조에 따라 각각 다른 기본 동작 스크립트가 적용되어야 하는데, 그림 14에서 보듯이 사용자가 동일한 'enter' 행위를 선택 하였지만 아바타 행위는 일반 문이거나 아니면 미닫이 문이거나에 따라 변화되는 것을 볼 수 있다. 미닫이 문에서는 아바타가 문을 옆으로 미는 동작을 보여주며, 일반 문은 잡아당기면서 'enter' 명령을 상황에 맞게 실행하고 있다.

3.7 저 수준 레벨 모션 동기화

3차원 환경에서 아바타, 객체 애니메이션은 그림 15와 같이 이루어진다. 위에서 언급하였듯이 '문 열기'라는 작업레벨 행위를 주었을 경우 상위레벨 행위 스크립트에서는 '(문으로)걷기-(문)열기-걷기-돌기-(문)닫기'라는 단위 행위로 나누어 지고 이는 다시 기본 동작 스크립트에 내려와 아바타, 객체의 실질적으로 애니메이션을 불러오게 된다. 모든 상위레벨 동작은 논리적 최소단위이기 때문에 기본 동작에 일대일 대응한다. 여기서 고려

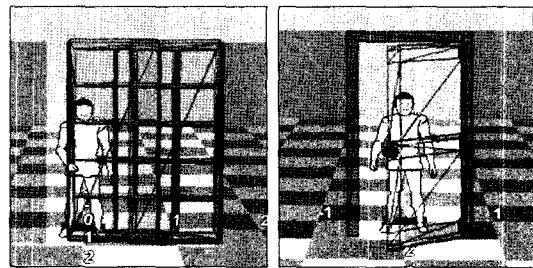


그림 15 객체의 type종류에 따른 모션 변화

해야 할 사항은 각각의 객체의 동기화 부분이라 할 수 있다. 객체 각각의 동작들은 그 동작이 끝나면 이벤트를 발생시킨다. 예를 들어 '걷기'의 경우 원하는 위치까지 가게 되면 이벤트가 발생되면서 객체 상태가 변화된다. '(문)열기' 부분을 보면 각 객체 별로 끝나는 시점은 다르지만 객체 모델에서 정의한 조건을 모두 만족시키는 시점은 한 곳이다. 이렇게 모든 조건을 만족시키는 점에 도달하게 되면 다음의 동작을 불러오는 방식으로 동기화가 이루어 진다. TTS 부분은 아바타의 음성을 표현하는 부분으로써 명령 단위가 작업레벨 행위이다. 이 때문에 하나의 작업레벨 행위가 끝나는 시점과 TTS가 끝나는 시점을 서로 비교해야 한다.

4. 시스템 구현 결과

2003 시스템 구현은 JAVA 3D API를 이용하여 구현하였으며 교육 도메인에서 강의 시뮬레이션을 생성하였다. 우선 아바타 모션 라이브러리를 구축하기 위하여 아바타 모션 편집기를 개발하였다. 아바타나 객체의 각 관절 부분을 조절하여 객체의 포즈를 생성하고 이를 키프레임 방식으로 보간하여 단위 동작을 생성하였다. 이렇게 만들어진 단위 동작들은 모션 라이브러리에 등록이 되며 이는 동작 데이터베이스로서 관리된다. 본 아바타 시스템은 그림 16과 같이 나타날 수 있다. 사용자는 3차원 GUI 환경에서 컨텍스트 메뉴를 통하여 직관적으로 명령을 내리고 이는 작업레벨 행위 스크립트로 기록이 된다. 스크립트는 해석기를 통해 기본 동작 스크립트까지 변환이 되고 이 정보들이 애니메이션 모듈로 보내져 실시간으로 애니메이션이 생성되어 보여진다.

시스템 구현은 JAVA 3D API를 이용하여 구현하였으며 교육 도메인에서 강의 시뮬레이션을 생성하였다. 우선 행위의 최소 단위인 아바타 모션 라이브러리를 구축하기 위하여 아바타 모션 편집기를 개발하였다. 아바타나 객체의 각 관절 부분을 조절하여 객체의 포즈를 생성하고 이를 키프레임 방식으로 보간하여 단위 동작을 생성하였다. 이렇게 만들어진 단위 동작들은 모션 라이브러리에 등록이 되며 이는 Motion DB동작 데이터베

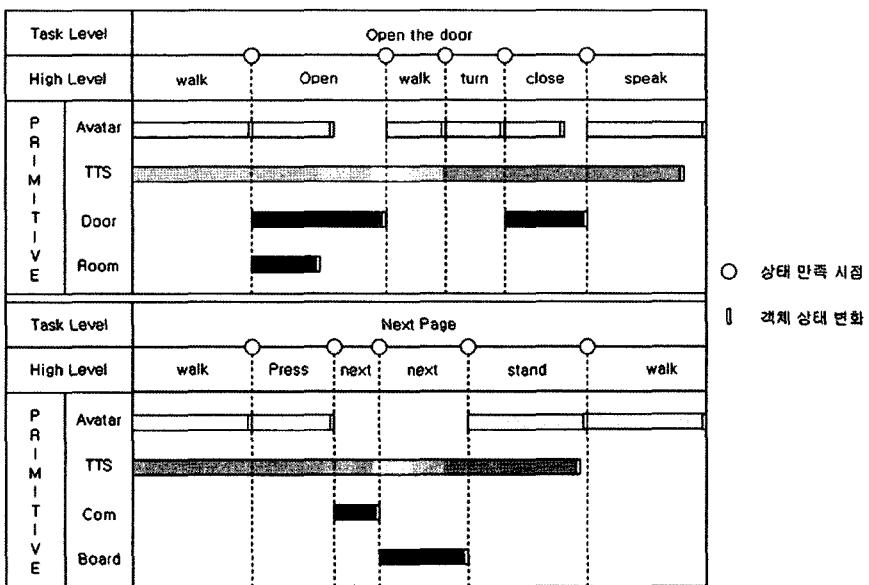


그림 16 객체-아바타 동기화

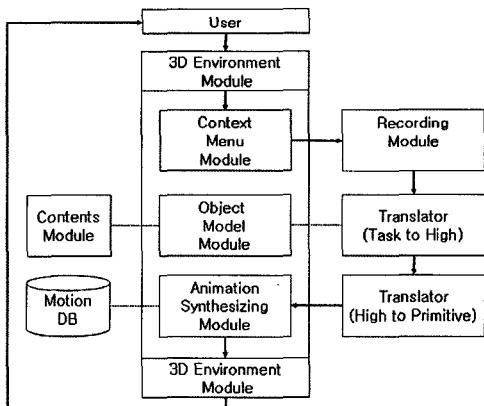


그림 17 시스템 구성도

이스로서써 관리된다. 본 아바타 시스템은 그림 14와 같이 나타날 수 있다. 사용자는 3차원 환경에서 직접 명령을 내리고 이는 스크립트로 기록이 된다. 스크립트는 해석기를 통해 Primitive기본 동작 스크립트 까지 변환이 되면 이 정보는 애니메이션 모듈로 보내져 애니메이션이 생성된다.

교육도메인에서 활용된 외부 콘텐츠는 한국 전자 책 표준인 EBKS 1.1 DTD의 서브 셋을 사용하였다. 기본적으로 스택 강의에 필요한 프레젠테이션프리젠테이션 자료를 XML로 구성한 후 시나리오를 생성해 보았다. 우선 표 2와 같이 EBKS 1.1 DTD의 엘리먼트들을 객체모델로 생성하고 강의에 필요한 다양한 행위들을 정의 하였다.

표 2 교육환경에서 아바타 행위	
Domain Object	Related Tasks
Book	Lecture
Screen	Next, Prev, Goto
Paragraph	Explain, Point, Read, Highlight, Unhighlight, Mark, Unmark, Ask, Demonstration
Note	Write, Erase, Demonstration
Door	Enter, Open, Close
Boxes	Move into, Move out, Push, Pop
Avatar	Greet, Talk, Walk, Attention, Bye

자체 표준인 EBKS 1.1 DTD의 서브 셋을 사용하였다. 기본적으로 스택 강의에 필요한 프레젠테이션프리젠테이션 자료를 XML로 구성한 후 시나리오를 생성해 보았다. 우선 표 2와 같이 EBKS 1.1 DTD의 엘리먼트들을 객체모델로 생성하고 강의에 필요한 다양한 행위들을 정의 하였다.

강의 시나리오를 생성하기 위해 기본적으로 객체에서 제공하는 행위인터페이스를 통해 스크립트 작성을 한다. 그러나 기본적으로 제공되는 행위 이외에 사용자 지식 기반 행위도 필요한데 그 예가 바로 강의에 필요한 테 모였다. 강의 중간에 사용할 수 있는 “push-pop operation” 네모의 경우 기존의 행위 인터페이스들을 이용하여 하나의 패키지로 저장하였다. 패키지는 하나의 지식 기반 행위로써 한 번 정의하여 저장하면 하나의 행위처럼 재사용이 가능하기 때문에 필요할 때 쉽게 불러다 사용할 수 있어 스크립트 생성 시간을 줄일 수 있었다. 강의 시나리오를 생성하기 위해 기본적으로 객체

에서 제공하는 행위인터페이스를 통해 스크립트 작성은 한다. 하나의 행위를 명령하면 직관적으로 그 행위가 실행이 된다. 우선 미리 정의된 객체 모델을 사용하여 강의 중간에 push-pop 데모로 활용하기 위한 Push-Pop 패키지를 생성하였다. 이는 하나의 지식 기반 behavior 행위로써 재사용이 가능하기 때문에 쉽게 행위처럼 불러다 사용할 수 있었다.

콘텐츠에서 제공되는 다양한 객체들에 객체모델을 정의하여 다양한 아바타 행위를 생성하였는데 페이지를 넘기거나 특정 라인에 하이라이트를 하거나 원하는 곳을 가리키는 행위들을 생성하여 강의에 사용되었다. 또한 아바타의 위치나 상태에 상관없이 'point line3'라는 명령을 하면 그림 17과 같이 지정된 곳까지 걸어와 그 곳을 가리키는 것을 확인 할 수 있었다. 또한 객체의 범위를 조절할 수 있었는데 책상과 박스 3개를 하나의 객체로 정의함으로써 그림 17과 같이 'push'나 'pop' 같은 복잡한 행위를 간단하게 조작할 수 있었다.

아바타는 강의에서 꼭 필요한 페이지를 넘기거나 각 줄 별로 하이라이트를 하거나 원하는 곳을 가리키는 행위들이 가능하다. 또한 아바타의 위치나 상태에 상관 없이 'point line3'라는 명령을 하며 그림 15와 같이 지정된 곳까지 걸어와 그 곳을 가리키는 것을 확인 할 수 있었다. 또한 객체의 범위를 조절할 수 있었는데 그림 15에서와 같이 책상과 박스 3개를 하나의 객체로 정의함으로써 push나 pop 같은 복잡한 행위를 간단하게 조작할 수 있었다.

5. 실험 및 토의

스크립트를 이용한 아바타 제어기법은 여러 종류가 있지만 현재 GUI 환경에서 실시간으로 스크립트를 생성할 수 있는 시스템은 매우 한정적이라 할 수 있다. 그 중 최근까지 활발하게 진행이 된 시스템은 바로 Alice v2.0[6]이라 할 수 있다. 비록 Alice의 활용 목적은 본 시스템과 다소 거리가 있을 수 있지만 인터페이스 측면

에서 본 논문과 비슷하게 가상환경 내에서 직접 스크립트를 생성하기 때문에 Alice v2.0과 본 논문에서 제시한 시스템과 비교 평가하였다. 실험을 하기 위해 우선 대학 학부생 9명 및 대학원생 11명의 남녀 20명을 대상으로 30분간 시스템 사용법을 설명하였다. 그 후 10분 길이의 강의 시나리오를 생성하게 하였다. 동일한 가상 환경을 만들기 위해 두 시스템 모두 객체를 아바타, 칠판, 그리고 상자로 한정 지었다. 시나리오에는 아바타가 강의장으로 들어와 슬라이드를 넘기며 설명을 하는 행동을 취하고, 스택 구조의 예를 설명하기 위해 상자를 pop, push 하는 과정 등이 들어 있다.

현재 GUI 환경에서 스크립트 생성이 가능한 것中最 가장 활발하게 진행이 된 시스템은 Alice v2.0[리퍼런스 6]이라 할 수 있다. Alice는 본 논문에서 제시한 시스템과 비슷하게 3차원 환경에서 객체를 선택한 후 행위 메뉴를 선택하는 방식으로 스크립트를 생성하는 인터페이스를 제공한다. 실험을 하기 위해 우선 일반대학 학부생 9명 및 대학원생 11명의 남녀 20명을 대상으로 30분간 시스템 사용법을 설명하였다. 그 후 10분 길이의 강의 시나리오를 생성하게 하였다. 객체는 본 시스템과 Alice와 동일하게 아바타, 칠판, 그리고 상자로 한정 지었으며, 시나리오는 아바타가 강의장으로 들어와 슬라이드를 넘기며 설명을 하는 모션을 취하고, 스택 구조의 예를 설명하기 위해 상자를 pop, push 하는 과정 등이 들어 있다.

그림 18에서 볼 수 있듯이 본 논문에서 제시한 시스템은 대부분 20분 안에 시나리오를 모두 완성한 반면 Alice의 경우 30분 이상 걸리는 경우도 발생하였다. 이러한 실험 후 사용자에게 두 시스템을 비교하는 설문 조사를 하였다. 우선 각 항목별 사용자 만족도를 조사하였는데 그 결과는 그림 19와 같다.

그림 19에서 볼 수 있듯이 본 논문에서 제시한 시스템은 대부분 20분 안에 시나리오를 모두 완성한 반면 Alice의 경우 30분 이상 걸리는 경우도 발생하였다. 이

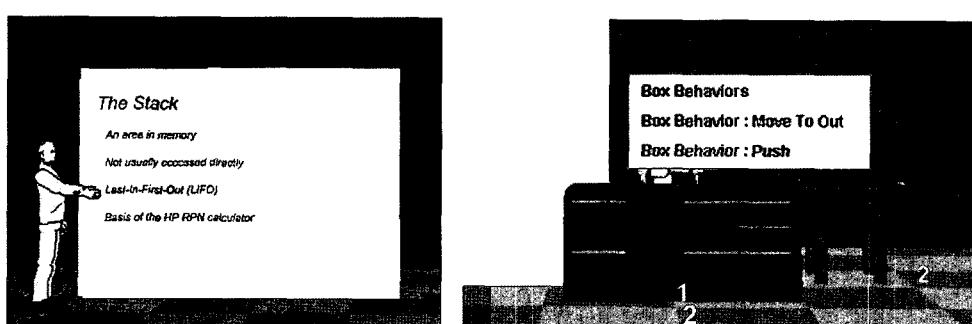


그림 18 콘텐츠를 적용한 강의 시나리오 및 컨텍스트 인터페이스

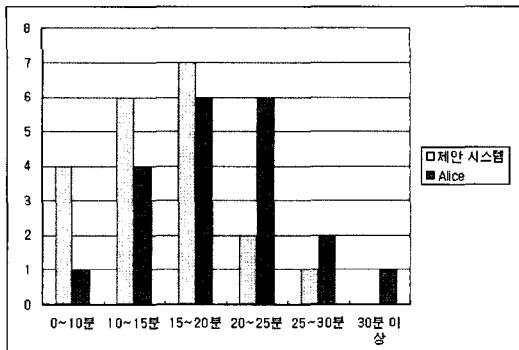


그림 19 스크립트 생성에 소요되는 시간

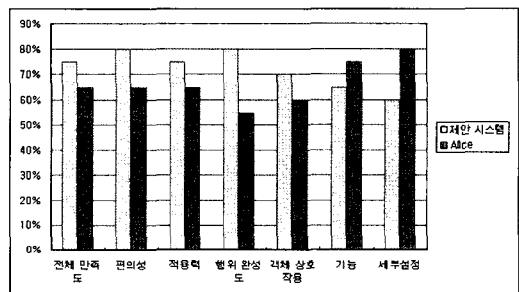


그림 20 사용자 만족도

리한 실험 후 사용자에게 두 시스템을 비교하는 설문 조사를 하였다. 우선 각 항목별 사용자 만족도를 조사하였는데 그 결과는 그림 17과 같다.

전체적이 만족도는 본 시스템이 좀 더 우수한 것으로 나왔다. 하지만 위의 결과에서 볼 수 있듯이 사용자 인터페이스와 관련한 편의성이나 적응 능력 부분에서 Alice에 비해 우수하게 평가 되었을 뿐 아니라 자연스러운 동작이나 추상화된 행위를 세부적으로 묘사하는 행위 완성도 부분에서 높은 평가를 가졌다. 또한 객체 상호작용 부문에서도 높은 평가를 받을 수 있었다. 하지만 본 시스템은 추상화된 행위들을 사용하기 때문에 세부적으로 동작을 조정하거나 사용자 시점을 변환하는 기능적인 부분에서는 Alice가 좀 더 높은 평가를 받았다.

마지막으로 본 시스템이 Alice에 비해 우수하다고 생각하는 이유를 들어보았는데, 대부분의 사용자들은 Alice의 인터페이스는 행위가 너무 많고 3차원 화면이 작아 사용하기 힘든데 반해 본 시스템은 3차원 환경에서 사용자가 가상 환경을 움직이면서 객체를 선택하고 상황에 따라 필요한 행위들만 보여주기 때문에 좀 더 편리하게 원하는 행위를 실행시킬 수 있었다는 의견이 많았다. 그 다음으로 많았던 의견은 추상화된 명령으로 구성되어 있기 때문에 적은 수의 명령으로 시나리오 완성이 가능하기 때문에 편리하다는 의견이 있었다. 그 외에 package패키지 사용이나 명령 후 바로 애니메이션 실행되어 편리했다는 의견도 있었다. 하지만 몇몇 사용자는 Alice에 비해 세부적인 설정을 할 수 없어 불편하여 원하는 대로 동작하지 않았다는 의견도 제시 되었다. 이러한 점은 제안 모델에서 향후 행위의 세분화 및 사용자가 추가적으로 세부 동작을 제어할 수 있는 확장 기능을 추가해야 할 것으로 보인다.

키지 사용이나 명령 후 바로 애니메이션 실행되어 편리했다는 의견도 있었다. 하지만 몇몇 사용자는 Alice에 비해 세부적인 설정을 할 수 없어 원하는 대로 동작하지 않았다는 의견도 제시 되었다. 이러한 점은 제안 모델에서 향후 행위의 세분화 및 사용자가 추가적으로 세부 동작을 제어할 수 있는 확장 기능을 추가해야 할 것으로 보인다.

전체적이 만족도는 본 시스템이 좀 더 우수한 것으로 나왔다. 하지만 위의 결과에서 볼 수 있듯이 사용자 인터페이스와 관련한 편의성이나 적응 능력 부분에서 Alice에 비해 우수하게 평가 되었을 뿐 아니라 자연스러운 동작이나 추상화된 행위를 세부적으로 묘사하는 행위 완성도 부분에서 높은 평가를 가졌다. 또한 객체 상호작용 부문에서도 높은 평가를 받을 수 있었다. 하지만 본 시스템은 추상화된 행위들을 사용하기 때문에 세부적으로 동작을 조정하거나 사용자 시점을 변환하는 기능적인 부분에서는 Alice가 좀 더 높은 평가를 받았다.

마지막으로 본 시스템이 Alice에 비해 우수하다고 생각하는 이유를 들어보았는데, 대부분의 사용자들은 Alice의 인터페이스는 행위가 너무 많고 3차원 화면이 작아 힘든데 반해 본 시스템은 3차원 환경에서 사용자가 가상 환경을 움직이면서 객체를 선택하고 상황에 따라 필요한 행위들만 보여주기 때문에 좀 더 편리하게 원하는 행위를 실행시킬 수 있었다는 의견이 많았다. 그 다음으로 많았던 의견은 추상화된 명령으로 구성되어 있기 때문에 적은 수의 명령으로 시나리오 완성이 가능하기 때문에 편리하다는 의견이 있었다. 그 외에 package패키지 사용이나 명령 후 바로 애니메이션 실행되어 편리했다는 의견도 있었다. 하지만 몇몇 사용자는 Alice에 비해 세부적인 설정을 할 수 없어 불편하여 원하는 대로 동작하지 않았다는 의견도 제시 되었다. 이러한 점은 제안 모델에서 향후 행위의 세분화 및 사용자가 추가적으로 세부 동작을 제어할 수 있는 확장 기능을 추가해야 할 것으로 보인다.

6. 결론 및 향후 연구

본 논문에서는 아바타 행위를 추상화 정도에 따라 3 단계로 구분하고 이를 스크립트에 적용하여 행위의 추상화 정도를 쉽고 유연하게 조절 할 수 있었다. 또한 아바타 행위 관련 정보를 객체에 분산하여 저장함으로써 사용자는 3차원 환경을 움직이면서 그 때 그 때 보이는 객체를 선택하여 직관적으로 행위 인터페이스를 제공받을 수 있었다.

본 논문에서는 가상 프레젠테이션 환경을 구성하여 아바타-객체 상호작용과 스크립트 생성의 편의성을 평가하였다. 실험은 GUI 환경에서 스크립트 생성이 가능

한 Alice v2.0와 비교하여 이루어졌으며 같은 시나리오를 완성하는데 얼마나 시간이 걸리는가를 체크하고 사용자 만족도를 조사하였다. 평균적으로 30%정도 빠른 시간 안에 스크립트 생성이 가능하였으며 사용자는 3차원 환경에서 직접 객체를 조작하기 때문에 좀 더 직관적인 인터페이스라는 평가가 나왔다. 3차원 환경이 복잡해짐에 따라 객체의 수도 늘어나고 할 수 있는 행위도 늘어나고 있다. 이러한 복잡한 환경에서 Alice와 같이 행위의 단순 분류로 아바타를 제어하게 될 경우 카테고리를 잘 나눈다고 할지라도 원하는 행위까지 가기 위해서는 여러 단계를 거쳐야 하고 그에 따라 직관성도 떨어진다. 또한 객체 내 행위들이 상태에 상관 없이 모두 보여지기 때문에 사용자에게 혼란을 주었다. 반면 본 논문에서 제시한 시스템의 경우 3차원 환경을 움직이면서 그 때 그 때 보이는 객체를 선택하면 그 상황에 맞는 행위들만 보여지므로 쉽고 빠르게 원하는 행위를 찾을 수 있다. 또한 기존의 시스템의 경우 스크립트를 모두 완성한 후에 애니메이션이 실행 되는 반면 본 논문에서 제시한 시스템의 경우는 명령 즉시즉시 애니메이션을 보여주기 때문에 사용자가 좀 더 직관적으로 시나리오 생성이 가능하다. 또한 행위의 추상화 정도를 조절할 수 있기 때문에 시나리오를 생성하기 위한 행위의 개수가 줄어든다는 장점이 있다. 즉 적은 횟수의 명령으로 원하는 시나리오가 생성되는 것이다.

지금까지는 자연스러운 동작을 만드는 저수준 동작 연구와 아바타 행위를 연구하는 고수준 행위연구가 따로 이루어지고 있다. 이 때문에 행위가 변경할 때나 객체-아바타간 상호작용 할 때 동작이 부자연스러운 부분이 나타날 뿐 아니라 동기화를 맞추는 부분에서도 문제점이 나타나고 있다. 또한 상위 레벨의 스크립트의 경우 다양한 기법들이 연구되고 있지만 표준화가 이루어지고 있지 않기 때문에 서로 통용될 수 없다. 이러한 문제점을 해결하기 위해서 아바타 행위에 대해 표준화 작업이 이루어져야 할 것이다.

본 논문에서는 아바타 행위를 추상화 정도에 따라 3 단계로 구분하고 이를 스크립트에 적용하였다. 또한 아바타 행위를 객체에 분산하여 저장함으로써 직관적인 인터페이스가 가능할 뿐 아니라 효율적인 제안 시스템 환경에서 일반적인 프리젠테이션을 위한 아바타-객체 상호 작용이 가능하였다.

실험은 GUI 환경에서 스크립트 생성이 가능한 Alice v2.0와 비교하여 이루어졌으며 같은 시나리오를 완성하는데 얼마나 시간이 걸리는가를 체크하고 사용자 만족도를 조사하였다. 평균적으로 30%정도 빠른 시간 안에 스크립트 생성이 가능하였으며 사용자는 3차원 환경에서 직접 객체를 조작하기 때문에 좀 더 직관적인 인터

페이스라는 평가가 나왔다. 3차원 환경이 복잡해짐에 따라 객체의 수도 늘어나고 할 수 있는 행위도 늘어나고 있다. 이러한 복잡한 환경에서 Alice와 같이 2차원 위주로 아바타를 제어하게 될 경우 카테고리를 잘 나눈다고 할지라도 원하는 행위까지 가기 위해서는 여러 단계를 거쳐야 하고 그에 따라 직관성도 떨어진다. 또한 객체 내 행위들이 상태에 상관 없이 모두 보여지기 때문에 사용자에게 혼란을 주었다. 반면 본 논문에서 제시한 시스템의 경우 3차원 환경을 움직이면서 그 때 그 때 보이는 객체를 선택하면 그 상황에 맞는 행위들만 보여지므로 쉽게 원하는 행위를 찾을 수 있다. 이 때문에 사용자는 좀 더 쉽게 명령을 내릴 수 있다. 또한 기존의 시스템의 경우 스크립트를 모두 완성한 후에 애니메이션이 실행 되는 반면 본 논문에서 제시한 시스템의 경우는 명령 즉시즉시 애니메이션을 보여주기 때문에 사용자가 좀 더 직관적으로 시나리오 생성이 가능하다. 또한 추상화된 스크립트를 사용함으로써 시나리오를 생성하기 위한 행위의 개수가 줄어든다는 장점이 있다. 이는 적은 횟수의 명령으로 원하는 시나리오가 생성되기 때문이다.

지금까지는 자연스러운 동작을 만드는 저수준 동작 연구와 아바타 행위를 연구하는 고수준 행위연구가 따로 이루어지고 있다. 이 때문에 행위가 변경할 때나 객체-아바타간 상호작용 할 때 동작이 부자연스러운 부분이 나타날 뿐 아니라 싱크동기화를 맞추는 부분에서도 문제점이 나타나고 있다. 또한 상위 레벨의 스크립트의 경우 다양한 기법들이 연구되고 있지만 표준화가 이루어지고 있지 않기 때문에 서로 통용될 수 없다. 이러한 문제점을 해결하기 위해서 아바타 행위에 대해 표준화 작업이 이루어져야 할 것이다.

참 고 문 헌

- [1] Rickel, J. & W. L. Johnson, "Task-Oriented Collaboration with Embodied Agents in Virtual Worlds," in Embodied Conversational Agents, Cassell, J., J. Sullivan, S. Prevost, et al., Eds. Boston: MIT Press, 2000.
- [2] Yasmine Arafa, Abe Mamdani, Scripting embodied agents behaviour with CML, Proc. IUI, (2003), pp.313-316, 2003.
- [3] S. Kshirsagar, et al, E. Mamdani, Avatar Markup Language, Proc. Eurographics, EGVE (2002), pp. 169-177, 2002.
- [4] Zhisheng Huang, et al, Implementation of a scripting language for VRML/X3D-based embodied agents, Proc web technology, (2003), pp.91-100, 2003.
- [5] Masaki Hayashi, TVML (TV program making language), ACM SIGGRAPH 98, (1998), pp.292-297. 1998.

- [6] Wanda Dann, Toby Dragon, Stephen Cooper, Kevin Dietzler, Kathleen Ryan, Randy Pausch, Objects: visualization of behavior and state, ACM SIGCSE Bulletin, v.35 n.3, September 2003.
- [7] M. Kallmann, D. Thalmann, Direct 3D Interaction with Smart Object, Proc. ACM VRST 99, London, 1999.
- [8] M. Kallmann and D. Thalmann, Modeling Behaviors of Interactive Objects for VR Applications, JVLC, Vol. 13, pp.177-195, 2002.
- [9] L. Levinson, "Connecting Planning and Acting: Towards an Architecture for Object-Specific Reasoning," PhD thesis, Univ. of Pennsylvania, 1996.
- [10] Yoshiaki Shindo Hiroshi, "Design and Implementation of Scenario Language for Cyber Teaching Assistant," Information & Communication Technology, International Conference on Computers in Education, 2001.
- [11] E. Andre, J. Muller, and T. Rist. "WebPersona: A Life-Like Presentation Agent for the World-Wide Web," In Proc. of the IJCAI Workshop on Animated Interface Agents: Making them Intelligent, 1998.
- [12] Xiaoli Yang; Petriu, D.C.; Whalen, T.E.; Petriu, E.M., "Script Language for Avatar Animation in 3D Virtual Environments," Human-Computer Interfaces and Measurement Systems, VECIMS '03. 2003 IEEE International Symposium on, pp.101 -106, 2003.
- [13] James C. Lester, Luke S. Zettlemoyer, Joel P. Gregoire, William H. Bares, "Explanatory lifelike avatars: performing user-centered tasks in 3D learning environments," Proceedings of the third annual conference on Autonomous Agents, Seattle, Washington, United States, Pages: pp.24-31, 1999.
- [14] M. Kallmann and D. Thalmann, "Modeling Behaviors of Interactive Objects for Virtual Reality Applications," Journal of Visual Languages and Computing, Vol. 13, pp.177-195, 2002.
- [15] Jae-Kyung Kim, Won-Sung Sohn, Soon-Bum Lim, Yoon-Chul Choy, "Avatar Behavior Representation and Control Technique: A Hierarchical Scripts Approach," International Symposium on Computational and Information Sciences, Shanghai, China, pp.873-878, December 16-18, 2004.

김재경

정보과학회논문지 : 소프트웨어 및 응용
제 33 권 제 2 호 참조

임순범

정보과학회논문지 : 소프트웨어 및 응용
제 33 권 제 2 호 참조

최윤철

정보과학회논문지 : 소프트웨어 및 응용
제 33 권 제 2 호 참조



최승혁

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