인터랙티브한 애니메이션 캐릭터 제작을 위한 인공지능 미들웨어 설계

이승섭°, 조경은*, 엄기현**
동국대학교 영상대학원 멀티미디어학과°, 동국대학교 영상미디어대학 게임멀티미디어공학과* **
{c127°, cke*, khum**}@dongguk.edu

A Design of AI Middleware for Making Interactive Animation Characters

Seungsub Lee°, Kyungeun Cho*, Kyhyun Um**
Dept. of Multimedia, Graduate School of Digital Media & Contents, Dongguk University°
Dept. of Game & Multimedia Engineering, College of Digital Media & Contents, Dongguk University* **

요 약

대부분의 디자이너는 3DS MAX와 같은 전문 3D 애니메이션 저작도구를 사용하여 수작업으로 애니메이션을 제작한다. 이 방법은 많은 시간과 노력을 필요로 하며, 애니메이션 캐릭터들이 서로 상호작용할 수 없다. 이를 개선하기 위해, 본 논문에서는 3DS MAX 플러그인 형태의 인공지능 미들웨어를 설계하고 미들웨어에 필요한 인공지능 표현 구조와 내부 처리 방안을 제안한다. 제안 방법은 캐릭터가 보유할 인공지능 요소를 도형과 선분으로 그려 표현하는 방법으로 캐릭터의 인공지능 구조를 제작한다. 실험을 위해 기존 방법과, 제안하는 방법을 사용하여 동일한 애니메이션을 제작하고 작업량을 측정하였다. 실험 결과 소규모 작업에서는 기존의 방법과 비슷하거나 작업량이 많으나, 대규모의 작업에서는 기존 방법에 비해 최대 43%의 작업량 감소를 확인하였다. 본 논문에서 제안하는 방법을 사용하면, 애니메이션에서 캐릭터간의 상호작용이 가능하며 작업량 감소 효과를 얻을 수 있다.

ABSTRACT

Most designers use professional 3D animation tools such as 3DS MAX to manually create animation. This manual method requires a great deal of time and efforts, and does not allow animation characters to interact with one another. In this paper, we design an AI middleware of form as 3DS MAX plug-in to solve these issues. We present an AI expression structure and internal processing method for this middleware, and the method for creating AI character's structure. It creates AI character's structure by drawing figures and lines for representing AI elements. For experiment, we have produced same animations with the traditional method and our method, and measured the task volume in both methods. This result verifies that the task volume is similar or higher than the traditional method in small-scale tasks, but up to 43% of the task volume is reduced in large-scale tasks. Using the method proposed in this paper, we see that characters in an animation interact each other, and task volume in large-scale tasks are reduced.

Keyword : AI middleware, character AI structure, interactive animation characters

※ 교신저자(Corresponding Author) : 조경은,
주소 : 서울시 중구 필동 3가 26 동국대학교(100-715), 전화 : 02)2260-3834, E-mail : cke@dongguk.edu

1. 서 론

대부분의 디자이너는 MAYA, 3DS MAX(이하 MAX)와 같은 애니메이션 저작도구를 사용하여 수작업 형태로 애니메이션을 제작한다. 애니메이션에 등장하는 모든 개체의 키 프레임(key-frame)을 잡아 제작하는 수작업 방식은 개체의 동작, 환경의 변화 등을 세부적으로 제어할 수 있다는 장점이 있으나 많은 작업량과 작업시간이 들어가며, 애니메이션에 등장하는 개체간의 상호작용이 불가능하다. 이를 해결하기 위해서는, 가상의 애니메이션 공간에서 자율적으로 행동하며 인공지능을 보유하고 인터랙티브한 애니메이션 캐릭터가 필요하다.

인공지능 캐릭터를 제작하기 위해 캐릭터가 동작하는 환경에서 정보를 얻는 방법과, 이 정보를 토대로 의사결정 구조를 표현하는 방법, 이 구조에 따라 행동을 수행하는 방법이 필요하다. MAX의 경우 키 프레임 에디터를 사용하여 프레임 단위로 애니메이션을 제작하는 기능만을 지원하며, 애니메이션에 등장하는 캐릭터가 스스로 행동할 수 있는 조건 판단 기능이나, 주변 환경에서 물체와의 충돌 혹은 거리, 종류 등을 탐지하는 기능은 제공하지 않는다. 기존 애니메이션 저작도구를 확장하여 애니메이션 캐릭터의 인공지능을 제작하는 방법이 필요하다.

본 논문에서는 기존의 애니메이션 저작도구와 연동하여 애니메이션에서 주변 환경을 인지하고 자율적으로 판단하며 행동하는 인터랙티브한 캐릭터를 제작하는 방법을 제시하고 "인공지능을 표현하는 방법"과 "표현된 인공지능을 처리하는 방법"으로 나누어 설명한다. 제안하는 방법대로 인터랙티브한 캐릭터를 사용하여 애니메이션을 제작하는 경우, 작업량을 감소시키는 것이 가능하다.

본 논문구성은 다음과 같다. 2장에서는 본 논문과 관련된 연구들을 설명한다. 3장에서 애니메이션의 캐릭터가 정보를 얻는 방법 및 사용자가 그래픽 객체를 배치하여 캐릭터의 의사결정 구조를 표현하는 방법을 설명한다. 4장에서는 캐릭터의 인공지능 구조를 시스템 내부에서 해석하여 지정된 행동을 수행하는 시스템 내부 처리 기법에 대해 소개한다. 5장에서 제안하는 방식을 사용하여 제작한 애니메이션의 결과 화면과 성능 분석을 수행하고, 끝으로 6장에서 결론을 서술한다.

2. 관련 연구

게임 분야에서 NPC의 이동, 무리행동 등의 인공지능을 제작하기 위한 인공지능 미들웨어에 관련된 연구들이 활발하게 진행 중이다[1,2,3]. 이 장에서는 본 논문과 유사한 목적으로 설계된 인공지능 저작도구들을 소개하고, 기존의 방법들과 본 논문에서 제안하는 방법과의 연관성을 설명한다.

Mind Editor는 인공지능 구조를 입력하는 편집기와 시뮬레이션 뷰어로 이루어진 군중 시뮬레이션을 위한 저작도구이다[4]. 이로는 각 캐릭터가 감정을 보유하고, 감정에 따라 행동하는 군중 장면을 표현한다. 인공지능 구조를 제작하려면 텍스트 프로그래밍 언어인 LUA 스크립트 언어를 사용한다.

AI. Implant는 게임, 애니메이션에 등장하는 다수의 객체를 조작할 수 있는 인공지능 저작도구이다[5,6]. AI. Implant로는 군중 장면을 편리하게 표현할 수 있으나, 인공지능 관련 기능 외에 자체적인 애니메이션을 조작하는 기능은 포함하지 않는다. 애니메이션 조작 시스템을 사용자가 추가로 구현해야 한다. Simbionic은 디자이너와 프로그래머의 협업을 통해 게임 NPC의 인공지능 구조 제작 기간을 단축할 수 있는 저작도구이다[7,8,9]. 인공지능 구조를 시각적으로 표현하지만, 사용자는 프로그램 코드를 작성해야한다. 또한 애니메이션 재생 시스템을 제공하지 않으므로 애니메이션 재생 시스템을 구현하거나, MAX와 같은 애니메이션 저작도구와 데이터를 연동하기 위한 데이터 파싱 시스템을 구현해야 한다.

인공지능 저작도구는 크게 텍스트 기반 프로그래밍 방식의 저작도구와, 시각 기반 프로그래밍 방식의 저작도구로 분류한다[4]. 전자는 일반적으로 사용하는 C언어 혹은 LUA 등의 스크립트 언어를

사용하여 프로그램을 제작하는 방식이다. 시각 기반 프로그래밍 방식은 그래픽 에디터를 사용하여 아이콘, 선, 도표 등의 그래픽 객체들을 만들고 이들 간의 관계와 이벤트의 흐름을 작성하여 프로그램을 제작하는 방식이다.

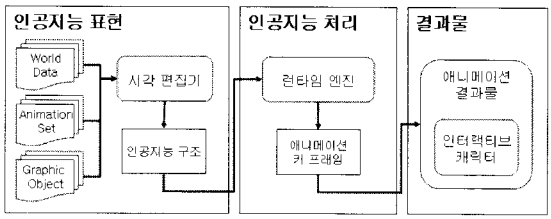
애니메이션 제작 시 인공지능을 도입하는 이유는 빠른 시간 안에 더 만족할 만한 결과물을 얻기 위함이다. 대부분의 인공지능 저작도구는 이러기에 충분하지 않다. 텍스트 기반 프로그래밍 방식에 비해 상대적으로 기술 습득에 필요한 시간이 적은 시각 프로그래밍 기법을 사용하여야 한다[10,11].

애니메이션을 위해 기존 인공지능 저작도구를 사용하는 것은 두 가지 문제점이 있다. 첫 번째로 기존의 인공지능 저작도구는 텍스트 프로그래밍 방식만을 사용하거나, 시각 프로그래밍을 사용하지만 여전히 주가 되는 기능은 텍스트 프로그래밍 방식을 사용하여 기술 습득에 많은 시간이 걸린다. 두 번째로 기존 인공지능 저작도구들은 애니메이션 제작이 아닌 게임 혹은 시뮬레이션을 목표로 제작되어 애니메이션 분야에서 필요한 키 프레임 조작 기능 등을 포함하지 않는다.

본 논문에서는 기존의 인공지능 저작도구의 단점을 보완하여 인터랙티브한 캐릭터를 구현하는데 시각 프로그래밍 기법을 사용하고, 애니메이션 키 프레임을 조작하는 기능을 포함하는 인공지능 미들웨어를 설계한다.

3. 인공지능의 표현 방법

인공지능 미들웨어는 캐릭터의 인공지능을 표현하고 처리하는 방법을 제공한다. 사용자는 미들웨어의 시각 편집기에서 제공하는 그래픽 객체를 사용하여 캐릭터의 인공지능 구조를 시각적인 그래프 형태로 표현한다. 시각 편집기는 캐릭터의 자율적인 판단을 위해 환경 정보와 캐릭터가 보유한 동작 애니메이션 정보를 제공한다. 그래프 형태의 인공지능 구조는 런타임(runtime) 엔진을 통해 애니메이션 키 프레임 형태로 변환된다. 이 과정을 간략하게 나타낸 것이 [그림 1]이다.

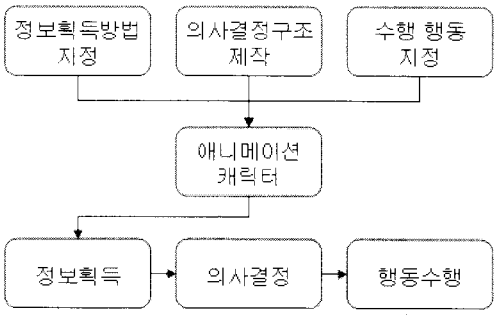


[그림 1] 미들웨어의 인공지능 처리 흐름

이 장에서는 캐릭터가 애니메이션에서 환경의 정보를 얻거나, 상황을 판단하고 행동하도록 만드는 방법을 다루고, 그래픽 객체를 사용하여 캐릭터 인공지능 구조를 표현하는 시각 프로그래밍 기법을 설명한다.

3.1 애니메이션의 인공지능 캐릭터

애니메이션에서 캐릭터가 주변 환경을 인지하고 스스로 판단하고 적절한 행동을 하기 위해서는 세 가지가 필요하다. 환경 정보를 얻는 방법, 그 정보를 사용하여 판단할 자신의 의사결정 구조, 판단에 따라 실행할 행동이 필요하다. 이를 위해 디자이너는 [그림 2]와 같이 캐릭터가 판단할 조건과 이에 따라 수행할 행동을 지정한다.

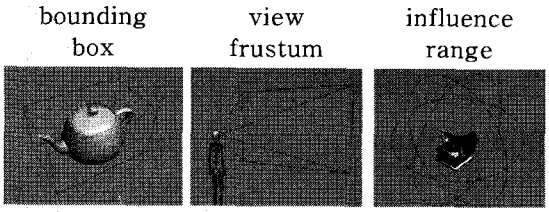


[그림 2] 인공지능 캐릭터가 동작하는 구조

3.1.1 정보 획득

미들웨어는 캐릭터가 정보를 수집하는 방법으로 영역 정보를 사용하여 충돌을 검출하는 방법과 미들웨어 시스템 내부 변수의 주소에 직접 접근하는 방법을 사용한다. 첫 번째는 경계 상자(bounding

box)이다. 경계 상자는 캐릭터나 물체의 경계 영역을 표현하는 직육면체 모양의 영역으로, 모서리 8곳의 정점 위치 정보를 포함한다. 물체나 캐릭터 간의 충돌 검사에 사용한다. 두 번째는 시야 절두체(view frustum)이다. 시야 절두체는 캐릭터의 시야 범위를 표현하기 위한 사각뿔 모양의 영역이다. 시야 절두체는 눈의 위치, 최대 시야 거리, 시야각 정보를 포함한다. 시야 절두체와 다른 물체나 캐릭터의 경계 상자의 충돌 여부로써 캐릭터가 대상이 시야 범위에 존재하는지 판단한다. 세 번째는 영향 범위(influence range)이다. 영향 범위는 구(Sphere) 형의 영역을 나타내기 위한 것으로, 중심점과 반지름 정보를 포함한다. 이 세 가지의 영역 정보 모형의 예가 [그림 3]이다.



[그림 3] 데이터 획득을 위한 영역 정보 모형

시스템 변수의 주소에 접근하는 방법은 충돌 검출로 정보를 수집할 수 없는 상황에 사용한다. 사용자가 생성한 정수 혹은 실수형 변수의 값이나 캐릭터의 속도, 캐릭터의 상태 등을 알아야 하는 경우에는 직접 변수의 주소에 접근한다.

3.1.2 의사 결정

캐릭터는 주어진 정보를 바탕으로 자신의 의사결정 구조에 따라 상황 판단을 한다. 이를 위해서 캐릭터가 자율적으로 의사결정 과정을 수행하기 위한 구조와 방법이 필요하다. MAX는 키 프레임 애니메이션 기법을 사용한다. 키 프레임 애니메이션은 [정의 1]과 같이 표현할 수 있다. [정의 1]의 구조에서는 캐릭터 등의 객체가 특정 프레임에 특정 행동 애니메이션을 수행하는 행위만을 포함한다. 그래서 미들웨어는 [정의 2]와 같이 객체가 조건을 판단하여 상태를 전이시키고, 상태에 따라 행동하는 FSM으로 캐릭터 의사결정 구조를 표현한다. 결과 애니메이션은 MAX 상에서 동작하므로 결국 [정의 1]을 따른다. 미들웨어는 [정의 2]와 같은 FSM 구조로써 캐릭터 의사결정을 진행하고, 최종적으로 애니메이션 키 프레임을 생성할 때 [정의 1]의 형식으로 의사결정 구조를 표현한다.

객체 O 가 프레임 F 에 애니메이션 B 를 수행

[정의 1] 키 프레임 애니메이션 구조

객체 O는 조건 C 에 의해 상태 S 로 변화
객체 O는 상태 S 에서 행동 A 를 수행

[정의 2] 캐릭터의 의사결정 구조

캐릭터의 의사결정 과정은 애니메이션의 시작부터 종료까지 반복적으로 수행된다. 애니메이션의 시작 프레임부터 루프를 돌며 경계 상자, 시야 절두체, 영향 범위를 통해 얻은 데이터를 사용하여 조건을 체크하고 상태를 전이시킨다. 조건 체크가 끝나면 상태를 검사하여 해당 상태의 행동을 실행하는 형태로 캐릭터의 의사결정을 수행한다. 이 구조가 [알고리즘 1]이다.

```
while( 시작 <= 현재 프레임 <= 종료 ) {
    if( 조건 1 )   상태 1로 전이
      else         상태 2로 전이
  ...
    if( 상태 1 )   행동 1 수행
    if( 상태 2 )   행동 2 수행
  ...
    프레임 증가 }
```

[알고리즘 1] 캐릭터의 의사결정 진행 루프

3.1.3 행동 수행

행동은 데이터를 수정하는 행위, 애니메이션을 변경하는 행위, 사용자가 지정한 행동을 수행하는 행위 등 세 종류로 분류한다. 첫 번째로 사용자는 애니메이션 제작 과정에서 수치를 변경해야 할 때

데이터 수정 행위를 수행한다. 많은 수의 경우에 애니메이션을 제작하면서 모든 상황에 일치하는 변수 혹은 데이터를 제공하는 것은 불가능하다. 미들웨어는 사용자가 직접 데이터 타입을 지정하여 생성하는 방법으로 문제를 해결한다. 두 번째로 애니메이션을 조작하는 행위가 존재한다. 애니메이션에서 캐릭터는 여러 가지의 동작을 보유하고 상황에 따라 동작을 변경한다. 미들웨어는 캐릭터의 각 동작 애니메이션을 로드하여 상황에 맞는 애니메이션 키 프레임을 생성한다. 세 번째로 캐릭터의 기하 정보를 조작하는 행위가 존재한다. 이는 캐릭터가 이동, 혹은 회전하거나 바라보는 방향을 바꾸는 등의 캐릭터가 실제로 움직이는 행위를 말한다. 캐릭터 이동, 캐릭터 시야 변경, 지정된 경로를 따라 이동하는 등의 행위들이 존재한다.

3.2 인공지능 표현을 위한 시각 프로그래밍 기법

미들웨어는 인터랙티브하게 행동하는 캐릭터의 인공지능 구조를 표현하기 위해 시각 프로그래밍 기법을 사용한다. 본 논문에서 제안하는 시각 프로그래밍 기법은 원, 마름모, 평행사변형, 직사각형 네 종류의 도형을 사용하여 캐릭터의 의사결정 구조를 표현한다. 이 네 종류의 도형을 그래픽 객체라 한다. 사용자는 각 그래픽 객체를 생성하고 그래픽 객체가 보유할 정보를 입력한다. 그 후 그래픽 객체간의 연결을 지정하여 캐릭터의 의사결정 구조를 제작한다. 이 절에서는 표현하는 대상을 설명하고, 그래픽 객체의 정의와 그래픽 객체가 포함하는 정보를 설명한다.

3.2.1 그래픽 객체의 정의

미들웨어는 [정의 2]의 요소를 나타내기 위해 네 종류의 그래픽 객체를 사용한다. 자신의 의사결정 구조를 가지며 스스로 행동이 가능하거나, 애니메이션의 다른 개체에 영향을 끼치는 캐릭터나 사물 등을 Object라 한다. Condition은 Object가 보유하며, 사용자가 제작하는 조건 판단 구문이다. State는 Condition의 진리 값을 가지며, Condition과 Action을 연결하는 스위치 역할을 한다. Action은 수행할 행위를 지정하기 위해 사용한다. 행위를 실행하는 명령은 런타임 엔진에 존재한다.

[표 1] 그래픽 객체의 종류

객체 이미지	객체명	역할	특징
Object	Object (주체)	사용자가 제작하는 최상위 객체	인공지능 캐릭터, 사물 등에 적용 가능
Condition	Condition (조건)	사용자가 제작하는 조건 판단 구문	if then else 구문을 표현
State	State (상태)	조건 판단에 따른 리턴 값	Condition과 Action을 연결
Set Ani Set Data Behavior	Action (행동)	상태의 값에 따라 지정된 행동을 수행	behavior, animation, set data 행동이 가능

3.2.2 그래픽 객체간의 관계

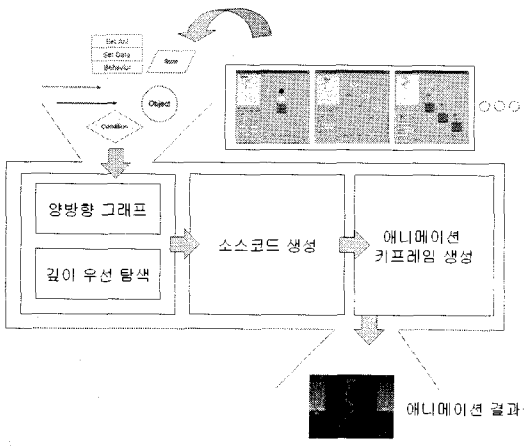
캐릭터의 의사결정 구조를 표현하기 위해, 여러 조건이 복합적으로 연결된 복잡한 의사결정 구조가 필요하다. 그래픽 객체들의 관계를 지정하는 연결 방법은 3가지이다. Condition과 Condition, Condition과 State, State와 Action을 연결하는 경우이다. 이를 위해 세 가지의 연결선을 사용한다. Condition이 참인 경우 다음 Condition이나 State와 연결되는 파란 색의 True Line, 거짓인 경우 연결되는 빨간 색의 False Line, 조건 판단과 관계없이 무조건 다음 Condition이나 State로 연결하는 검은 색의 And Line이 있다. 연결선으로 객체를 연결한 경우, 각 객체는 자신과 연결된 객체의 주소 포인터를 저장한다. [표 2]는 연결선을 정리한 것이다.

[표 2] 그래픽 객체를 연결하는 연결선

객체 이미지	명칭	설명
→	True Line	Condition에서 조건이 true일 경우 State나 Condition으로 넘어가는 연결선.
→	False Line	Condition에서 조건이 false일 경우 State나 Condition으로 넘어가는 연결선.
→	And Line	병렬 처리를 위해 연결하는 선, 조건이 참이나 거짓이거나에 관계없이 연결된다.

4. 표현된 인공지능의 내부 처리 기법

미들웨어 시스템은 캐릭터가 인터랙티브하게 동작하도록 하기 위해, 그래픽 객체들로 구성된 그래프 형태의 인공지능 구조를 시스템 내부적으로 처리 가능한 형태로 변환하며, 이를 분석하여 애니메이션 키 프레임을 생성한다. 이후, 사용자는 최종 결과물로 키 프레임 애니메이션 형태로 구성된 인터랙티브한 캐릭터가 포함된 애니메이션을 얻는다. [그림 4]는 이러한 과정을 나타낸 것이다.



[그림 4] 인공지능 처리 기법의 개요

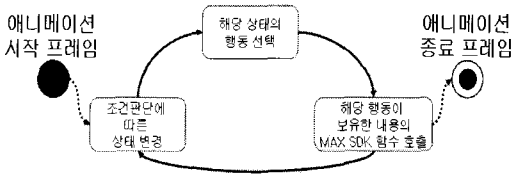
이 장에서는 인공지능 구조를 처리하기 위해 사용하는 자료구조, 인공지능 구조를 해석하여 조건을 판단하고 행동을 수행하는 내부 처리구조를 생성하는 방법, 애니메이션 키 프레임을 생성하는 방법을 설명한다.

4.1 내부 처리를 위한 자료구조

임의의 인공지능 구조를 처리하기 위해 다음과 같은 세 가지의 규칙을 따른다. 첫째, 사용자가 생성할 객체 개수를 알 수 없으므로, 동적 메모리 할당이 가능한 연결 리스트를 사용한다. 둘째, 의사결정 구조로 서로 다른 부모 노드 A, B가 동일한 자식노드 C를 가지는 경우가 발생하므로 트리 구조가 아닌 그래프 구조를 사용한다. 셋째, 의사결정 구조에서 조건에 따라 참, 거짓을 판별하고 다시 하위 노드에서 조건 판별이나 행동을 수행하는 구조 상 깊이우선 탐색을 사용하여 그래프를 운행한다.

4.2 내부 처리구조 생성 방법

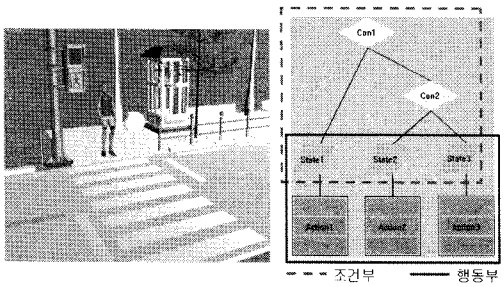
캐릭터의 의사결정은 애니메이션의 시작부터 종료까지 반복적으로 실행된다. 내부 처리구조 생성 방법은 특정 상황에 캐릭터가 수행할 행동을 선택하고, 해당 행동이 보유한 내용대로 키 프레임을 생성하는 MAX SDK함수를 호출하는 것이다. [그림 5]는 이러한 내부 처리구조가 수행되는 구조를 나타낸다.



[그림 5] 내부 처리구조 생성 방법의 개요

[그림 6]의 (a)는 내부 처리구조를 생성하는 방법을 보이기 위한 예시 애니메이션이다. 캐릭터는 신호등이 빨간색인 경우 제자리에서 대기하고, 신호등이 파란색인 경우 걸어서 길을 건넌다. 만약 신호등이 파란색으로 깜박거리면 뛰어서 길을 건너는 인공지능 구조를 보유한다. [그림 6]의 (b)는

[그림 6]의 (a)에 등장하는 캐릭터가 갖는 그래프 형태의 의사결정 구조이다. 의사결정 구조를 처리하기 위해 조건 판단에 따른 상태 변경만을 수행하는 조건부와 상태 판단에 따른 행동만을 수행하는 행동부로 분류한 형태의 구조가 필요하다. [그림 6]의 (b)에서 점선 부분은 조건 판단만을 수행하는 조건부이며, 실선 부분은 행동을 수행하는 행동부이다.



(a) 애니메이션 화면 (b) 의사결정 구조

[그림 6] 건널목을 건너는 애니메이션 예

4.2.1 조건부의 코드 생성

인공지능 구조대로 조건을 판단하고 수행될 행동을 선택하려면 객체의 현재 상태를 알아야 한다. 조건부에서 코드를 생성하는 목적은 특정 시간에 객체가 하나의 상태만을 갖기 위함이다. Condition은 조건 판단을 위해 논리 값 두 개와 논리 값을 비교하는 연산자를 보유한다. 그리고 다음 객체를 가리키는 포인터를 보유한다. 객체의 상태를 변화시키기 위해 Condition은 사용자가 지정한 논리 값과 연산자를 사용하여 자신의 조건 구문을 판별한다. 판별 결과에 따라 자신의 다음 객체가 Condition이라면 다시 조건 구문을 판별하고, 다음 객체가 State라면 해당 State에 참 혹은 거짓 값을 부여하는 코드 구조가 필요하다. 이를 정리한 것이 [알고리즘 2]이다.

[그림 7]은 [그림 6]의 (b) 그래프 구조에서 [알고리즘 2]에 따라 조건부를 C언어 형태로 코드화 한 것이다. 캐릭터는 신호등 Object의 상태를 기준으로 조건을 판별한다. 신호등 Object는 빨간색인 경우(상태 1), 파랑색이 깜빡이는 경우(상태 2), 파랑색인 경우(상태 3)의 세 가지 상태를 보유한다.

```
struct con{ int tarA, tarB, Oper; //논리값A, B, 연산자
            con* tr_con, fa_con; //다음 Condition
            sta* tr_sta, fa_sta; }; //다음 State
main() {  graphDFS(con);  }
void graphDFS(con* in){
switch(in->Oper){ // 연산자 판별
  case AGreaterThenB: // 연산자가 > 라면
    if(in->tarA > in->tarB) // 논리값A>논리값B라면
      if(in->tr_sta != null) in->tr_sta = true;
      else graphDFS(in->tr_con);
    else // 논리값A<=논리값B라면
      if(in->fa_sta != null) in->fa_sta = true;
      else graphDFS(in->fa_con);
  case... // >외의 연산자인 경우..
}}
```

[알고리즘 2] 조건 – 상태 알고리즘

```
if( Con1 == true ) // 신호등 색 == 빨강
  State1 = true; // 캐릭터 상태 1로 변경
else
// 신호등 색 == 파랑 깜빡임
 if( Con2 == true )
   State2 = true; // 캐릭터 상태 2로 변경
// 신호등 색 != (빨강 && 파랑 깜빡임)
 else
   State3 = true; // 캐릭터 상태 3으로 변경
```

[그림 7] [그림 6]의 (b) 조건부의 코드화 결과

4.2.2 행동부의 코드 생성

행동부의 코드 생성 단계에서는 먼저 현재 객체의 상태에 따라 수행될 행동을 선택한다. 다음으로 선택된 행동의 MAX SDK 함수를 호출하여 애니메이션 키 프레임을 생성한다. 현재 객체의 상태를 알아내기 위해 모든 State를 검색하여 현재 true값을 갖는 State를 찾고, 해당 State와 연결된 Action을 찾아 행동을 수행하는 코드 구조가 필요하다. [알고리즘 3]은 이러한 코드 구조를 나타내며, [알고리즘 3]에서 StateCount는 전체 State의 개수를 나타낸다.

```
for(int i = 0 ; i < StateCount ; i++)
if ( State[i]->Value == true )
  State[i]->GetAct(Excute);
```

[알고리즘 3] 상태 – 행동 알고리즘

애니메이션 키 프레임을 조작하기 위해 객체는 자신의 행동에 런타임 엔진의 함수를 호출하는 코드를 포함한다. [그림 8]은 [그림 6]의 (b) 그래프 구조에서 [알고리즘 3]에 따라 행동부를 C언어 형태로 코드화한 것이다. State1이 참이면 신호등이 빨간색이므로 캐릭터 Object가 제자리에서 서있는 애니메이션의 ID를 매개변수로 애니메이션을 변경하는 함수를 호출한다. State2가 참이라면 신호등이 파란색으로 깜빡거리고 있는 상태이므로, 애니메이션 변경 함수를 호출하여 캐릭터 Object의 달리기 애니메이션으로 변경하고 데이터 변경 함수를 사용하여 캐릭터의 속도를 20으로 변경한다. 다음으로 이동하려는 곳의 X, Y, Z 좌표를 매개변수로 이동 함수를 호출한다. State3이 참이라면 캐릭터 Object의 걷기 애니메이션을 구동하고 속도를 10으로 설정하여 이동시킨다.

```
if ( State1 == true )
    Character->PlayAnimation(Stop);
if ( State2 == true )
    Character->PlayAnimation(Run);
    Character->SetData(Speed, 20);
    Character->Move(X, Y, Z);
if ( State3 == true )
    Character->PlayAnimation(Walk);
    Character->SetData(Speed, 10);
    Character->Move(X, Y, Z);
```

[그림 8] [그림 6]의 (b) 행동부의 코드화 결과

5. 실험 및 분석

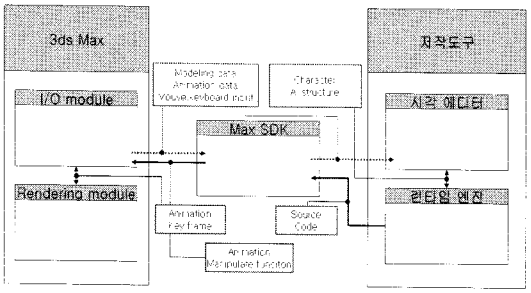
이 장에서는 수작업 형태로 애니메이션을 제작하는 기존의 방법과 본 논문에서 제시하는 시각 프로그래밍 기법을 사용하여 인공지능 캐릭터를 제작하고 애니메이션에서 사용하는 방법을 비교하는 실험을 진행한다. 실험에서는 기존의 수작업 형태로 실현하지 못하는 인터랙티브한 캐릭터를 제작하는 것이 가능함을 보인다.

5.1 실험 목적 및 방법

본 실험 목적은 동일한 규모의 애니메이션 제작 작업에서 수작업 방식에 비해 인공지능 캐릭터를 사용하여 애니메이션을 제작하는 것이 효율적임을 입증하는 것이다. 효율성을 판단하기 위한 기준으로 작업에 소요되는 시간과 작업량이 존재한다. 작업 시간을 측정하는 경우는 초보자와 숙련자 간의 편차가 크고 작업에 소요되는 최소, 최대 시간을 명확하게 제시하기 어렵다. 또한 기존의 인공지능 저작도구들이 사용하는 스크립트를 사용하는 방법은 각각 사용하는 언어가 다르고 초보자와 숙련자 간의 작업량 및 작업 시간 편차를 예측하기 어렵다. 따라서 본 실험은 동일한 작업 규모에서 수작업과 미들웨어를 사용하여 애니메이션 제작을 진행하고, 완료될 때까지의 작업량을 비교하는 방법으로 진행한다. 실제 사용자가 진행한 작업 양을 측정하기 위해 사용자의 키보드의 입력 건수, 마우스의 클릭 수와 드래그 수를 각 1회로 하여 모두 더한 값을 사용한다.

5.2 실험 시스템 설계 및 구현 환경

설계한 인공지능 미들웨어는 시각 에디터와 런타임 엔진으로 구성된다. 런타임 엔진은 MAX SDK의 정점 이동, 회전, 크기 변환 등의 함수들을 포함하며 이를 호출하여 캐릭터의 기하 정보를 변형, 키 프레임을 생성한다. [그림 9]와 같이 인공지능 미들웨어는 MAX SDK의 함수를 사용하여 모델링 데이터와 애니메이션 데이터, 사용자의 키보드 및 마우스의 입력 정보를 MAX로부터 얻어온다. 사용자는 시각 에디터를 실행하고, 캐릭터 인공지능 구조를 그래프 형식으로 그린다. 이후 런타임 엔진으로 애니메이션을 구동하기 위한 MAX SDK함수를 포함한 내부 처리구조를 C언어 코드 형태로 생성한다. 이 코드를 사용하여 애니메이션 생성 명령을 MAX에게 전달하여 애니메이션 키 프레임을 생성한다. 이러한 과정을 거쳐 사용자는 최종적으로 키 프레임으로 구성된 애니메이션 파일을 얻는다.

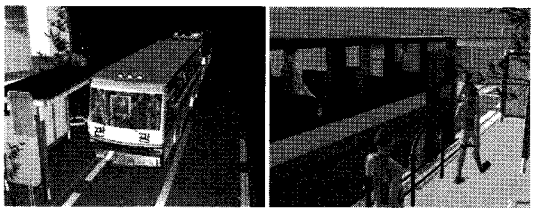


[그림 9] 인공지능 미들웨어의 동작 형태

MAX SDK는 MAX의 대부분의 기능들을 C언어 기반의 함수 형태로 지원한다. MAX SDK를 사용하여 기존에 MAX에서 지원하는 기능을 수정하거나 플러그인 형태로 자신이 원하는 기능의 추가가 가능하다. 미들웨어는 Microsoft Visual C++를 기반으로 MAX SDK의 함수를 포함하여 동적 연결 라이브러리(DLL)로 제작한다. MAX는 플러그인을 위한 폴더를 보유하며, MAX 구동 시에 해당 폴더 내에 존재하는 모든 DLL파일을 로드하는 형태로 플러그인들과 연동한다.

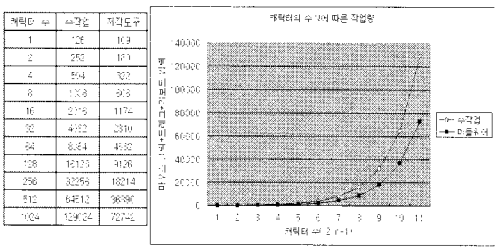
5.3 실험 내용 및 결과 분석

실험에서는 영화 혹은 애니메이션에서 흔히 제작되는 다수의 캐릭터가 등장하는 군중 장면의 경우를 예로 들어 제안하는 방법의 효율성을 보인다. 각 캐릭터가 보유한 동작 애니메이션을 변경하며 주어진 경로를 따라 이동하는 애니메이션을 수작업 형태와 미들웨어를 사용한 형태로 제작하였다. 실험에서 등장하는 캐릭터의 수를 증가시키며 실험을 진행하고 그 외의 변수는 모두 동일하게 고정하였다. 각 캐릭터는 걷기, 계단을 올라가기, 버스 좌석에 앉기, 3가지의 기본 애니메이션을 보유하며 각 기본 애니메이션은 모두 1회씩 사용한다. 또한 각 캐릭터에게 주어진 이동 지점의 수는 10개로 동일하다. 각 캐릭터는 도심을 배경으로 한 버스 정류장에서 일렬로 대기하고 있으며, 정류장에 버스가 도착할 경우 차례대로 버스에 탑승하는 장면이다. [그림 10]은 진행한 실험의 스크린 샷이다.



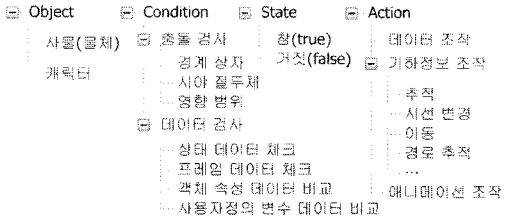
[그림 10] 실험 애니메이션 스크린 샷

[그림 11]은 캐릭터의 수가 2^{n-1}개일 때, 기본 동작 애니메이션 3개, 애니메이션 변경 횟수 2회, 이동 경로 10개, 24프레임의 기본 동작을 사용하여 240프레임의 애니메이션을 제작할 때의 수작업과 미들웨어의 작업량을 측정한 결과이다.



[그림 11] 캐릭터 수 증가에 따른 작업량 변화

인터랙티브한 애니메이션 캐릭터는 다양한 상황의 애니메이션 장면에서 사용 가능해야 한다. 제안하는 시각 프로그래밍 기법에서 네 종류의 그래픽 객체는 [그림 12]와 같은 정보를 포함한다.



[그림 12] 미들웨어의 기능일람표

미들웨어는 예를 들어 사물이 특정 프레임을 검사하여 값을 변경하는 행위, 캐릭터가 충돌을 검사하여 애니메이션을 변경하는 경우와 같이 [그림 12]의 각 기능을 조합하여 사용한다.

5.4 실험 평가

실험에서 제작한 캐릭터를 애니메이션에서 사용할 때와 수작업으로 작업할 때의 작업량을 측정한 결과, 제안 방법이 수작업보다 작업량이 감소하였다. [그림 11]의 실험 결과 데이터를 살펴보면, 캐릭터 수가 적을 때, 즉 애니메이션의 규모가 작을 때는 수작업과 미들웨어의 차이가 적으나, 캐릭터 수가 증가할수록 미들웨어의 작업량이 더 줄어든다. 이 실험결과는 소규모의 애니메이션보다는 대규모의 애니메이션일수록 미들웨어를 사용할 경우 작업 효율이 높아짐을 의미한다. 제안 방법은 대규모의 캐릭터가 출현하는 군중 시뮬레이션 분야에서 사용할 경우 탁월한 효과를 보인다고 판단된다. FSM으로 나타낼 수 없는 복잡한 인공지능의 경우, 인공지능 구조 제작에 있어 난해함이 있다.

6. 결 론

본 논문에서는 인터랙티브한 애니메이션 캐릭터를 제작하기 위해 MAX 플러그인 형식의 인공지능 애니메이션 미들웨어를 설계하고, 미들웨어에서 사용가능한 인공지능 표현 구조와 내부 처리 방안을 제시하였다. 미들웨어는 인공지능을 표현하는 시각 에디터와 인공지능을 처리하는 런타임 엔진으로 분류하여 구성하였다. 이 구조에 따라 런타임 엔진을 수정하는 것만으로 새로운 인공지능 기능을 쉽게 추가할 수 있다.

사용자는 시각 편집기를 사용하여 복잡한 스크립트를 작성하지 않고 마우스 클릭과 키보드를 통한 값 입력만으로 인터랙티브한 캐릭터를 구현할 수 있다는 점에서 기존의 시각화 구조를 만들고 구조에 따라 코드를 작성하는 방식과는 차별화된다.

본 논문에서 제안하는 미들웨어를 사용하여 기존의 애니메이션 저작도구와 연동하여 인터랙티브한 캐릭터를 제작 가능함을 보였다. 실험을 통해 기존의 수작업 형태와 비교하여 작업효율성이 더 높음을 증명하였다. 또한 애니메이션의 규모가 커질수록 제안하는 방법이 효과적임을 알 수 있었다. 제안 방법을 활용하여 애니메이션 콘텐츠 외에도 AI 엔진 제작이나 게임용 AI 제작 툴 등의 다양한 콘텐츠 제작에도 응용이 가능할 것이다.

〔참고문헌〕

[1] http://www.gameai.com/toolkits.html (2007)

[2] "국내외 게임기술 동향 : 미들웨어 기술", 2007 대한민국 게임백서(하), 한국게임산업진흥원, pp.835-844, 2007.

[3] 이헌주, "게임 인공지능 기술", 전자통신동향분석 제20권 제4호, 한국전자통신연구원, pp.104-106, 2005.

[4] Björn Axelsson, "MindEditor - An End-User Tool for Implicit Design of Simulated Human Behaviour", UME UNIVERSITY Department of Computing Science Master's Thesis, 2003.

[5] Paul Kruszewski, "AI-implant: A game-AI derived general scalable model for life-form simulation in MOUT-based applications", AI-Implant White Paper, 2005.

[6] http://www.ai-implant.com/ (2007)

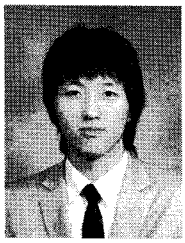
[7] Daniel Fu, Ryan Houlette, "An Authoring Toolkit For Simulation Entities", The Proceedings of the I/ITSEC, 2001.

[8] Daniel Fu, Ryan Houlette, "Putting AI in Entertainment: An AI Authoring Tool for Simulation and Games", IEEE Intelligent Systems, pp.81-84, 2002.

[9] http://www.simbionic.com/ (2007)

[10] N. C. Shu, "Visual Programming Languages : A Perspective and A Dimensional Analysis", Visual Languages, pp.11-34, 1986.

[11] Shi-Kuo Chang, "Visual Language : A Tutorial and Survey", IEEE Software, pp.29-39, 1987.



이 승 섭 (Seungsub Lee)

2006.2 한국IT전문학교(이학사)
2008.2 동국대학교, 영상대학원 멀티미디어학과 (예술공학석사)

관심분야 : 컴퓨터 게임 알고리즘, 게임 인공지능, 미들웨어 기술

조 경 은 (Kyungeun Cho)

1993.2 동국대학교 전자계산학과(공학사)
1995.2 동국대학교 컴퓨터공학과 대학원(공학석사)
2001.8 동국대학교 컴퓨터공학과 대학원(공학박사)
2002.3~2003.2 안양대 디지털미디어학부 강의전담교수
2003.3~2003.8 영산대학교 게임공학과 전임강사
2003.9~2005.8 동국대 정보산업대학 컴퓨터멀티미디어공학과 전임강사
2005.9~2006.8 동국대 정보산업대학 컴퓨터멀티미디어공학과 조교수
2006.9~현재 동국대 영상미디어대학 게임멀티미디어공학과 조교수

관심분야 : 컴퓨터 게임 알고리즘, 게임 인공지능, 멀티미디어 정보처리

엄 기 현 (Kyhyun Um)

1975.2 서울대학교 공과대학 응용수학과 공학사
1997.2 한국과학기술원 전산학과 이학석사
1994.2 서울대학교 대학원 컴퓨터공학과 공학박사
1978.3~2006.8 동국대 컴퓨터멀티미디어공학과 정교수
2006.9~현재 동국대학교 영상미디어대학 게임멀티미디어공학과 교수
1995.3~1999.2 동국대학교 정보관리처장
2001.3~2003.2 동국대학교 정보산업대학 학장
1998.9~2000.8 한국정보과학회 데이터베이스연구회 위원장
1999.4~2005.3 Int. Conf. on Database Systems for Advanced Applications Steering Committee 위원
2001.1~2002.12 한국정보과학회 논문지편집부위원장(데이타베이스)
1998.12~현재 한국멀티미디어학회 부회장, 수석부회장, 회장 역임, 자문위원(현재)
2004.1~현재 한국 게임학회 부회장 역임, 자문위원(현재)

관심분야 : 게임시스템 설계, 멀티미디어 데이터베이스, 멀티미디어 정보시스템