FSM의 직관적인 모델링을 위한 에디터 설계

송병근°, 이현진, 김상균, 서재현
인제대학교 전산학과 게임개발연구실
amidala°, cyberpd@gdl.inje.ac.kr
skkim, jaiseu@cs.inje,ac.kr

Design for Intuitive Modeling of FSM Editor

ByungKeun Song°, HyunJin Lee, SangKyoon Kim, JaiHyun Seu
Inje University Computer Science

요 약

이 논문에서는 게임에서 인공지능 구현에 많이 사용되는 FSM(Finite State Machine)을 이용하여 다양한 인격의 NPC(Non-Player Character)를 생성함에 있어서, 게임 디자이너의 NPC 패턴 설계에 도움을 주기위해, 직관적인 FSM 상태 모델링 에디터를 설계하고자 한다. 이 툴을 이용하여, FSM의 각 상태와 상태의 변이에 따른 여러 가지 반응을 게임 디자이너가 직관적으로 알 수 있도록 함으로써 보다 다양한 인격을 가진 NPC를 생성할 수 있을 것으로 기대된다.

1. 서 론

게임 속에서 인공지능은 슈팅 게임에서 사용되는 등장 패턴과 사용자의 움직임에 따른 회피, 공격과 같은 간단한 인공지능부터 시뮬레이션이나 액션 게임에서 사용되는 복잡한 주변 상황을 인식하고, 수학적인 계산을 하여 일련의 행동을 하게 하는 복잡한 인공지능까지 아주 다양하게 사용되고 있다.

이러한 인공지능을 구현하기 위한 기법 중, FSM을 이용한 NPC의 인공지능 구현은 최근 급성장하고 있는 온라인 RPG 게임에서 많이 사용하고 있다. FSM은 구현이 다른 인공지능 기법들보다는 상대적으로 간단하면서도 다양한 형태의 NPC 행동 패턴을 생성할 수 있는 장점을 가지고 있다.

FSM은 말 그대로 게임 객체가 가질 수 있는 일련의 상태와 각 상태의 전이를 위해 필요한 이벤트를 정의하여 객체의 상태 변이를 구현하는 것이다. 이러한 상태 변이를 이용해서 각 NPC마다 독특한 인성을 부여하게 되는 것이다.

게임에서의 일반적인 NPC 행동 패턴 구현 방법은 게임 디자이너가 행동 패턴을 설계하고, 게임 프로그래머가 하드 코딩으로 프로그램 내에 이러한 행동 패턴을 구현을 하게 된다. 이 경우, 행동 패턴이 단순하게 만들어지고, 확장성도 떨어지는 문제점을 가지게 된다. 이러한 문제 때문에 행동 패턴을 제어하는 여러 가지 요인을 게임 디자이너가 외부에서 제어 할 수 있도록 프로그램을 만들게 된다. 이렇게 함으로서 단순한 패턴을 여러가지 환경에 따라 복잡하게 만들수 있게 할 수 있다. 그러나, 이러한 방법에도 문제점이 나타나게 되는데, 프로그램 내에서 제어 되는 모든 인자에 대한 이해가 게임 디자이너에게 요구 되어 지게 된다.

본 논문에서는 이러한 게임 디자이너와 프로그래머의 능률적인 역할 분담과 디자이너가 인공지능 소스 코드를 이해하기 위해 사용하는 시간의 낭비를 막고, NPC의 패턴을 생성함에 있어서 좀더 직관적이고 비주얼한 환경을 제공하기 위한 에디터를 제안하고자 한다. 이렇게 함으로서 보다 많은 NPC 인성을 디자인하고, 게임에 적용시킴으로서 좀 더 흥미 있는 게임을 구성할 수 있을 것이다.

2. NPC 상태와 상태 변이 구성

간단한 FSM의 상태와 상태 전이를 발생시키는 이벤트에 대해서 살펴보자. 먼저 초기 단계로서 객체의 상태를 구체적으로 구분을 해야 한다. 게임에서 몬스터 NPC의 상태를 몬스터의 행동으로 구분을 했을 때, 기본적으로 공격, 후퇴, 배회, 정지라는 네 가지의 상태를 가진다고 하자. 다음으로는 상태의 전이를 발생시키는 조건으로서, 플레이이의 등장, 플레이어의 공격, 플레이어의 죽음, 플레이이의 안보임, 플레이의 후퇴 등의 입력을 생각할 수

있을 것이다. 이것을 그림으로 나타내면 [그림 1]과 같다.

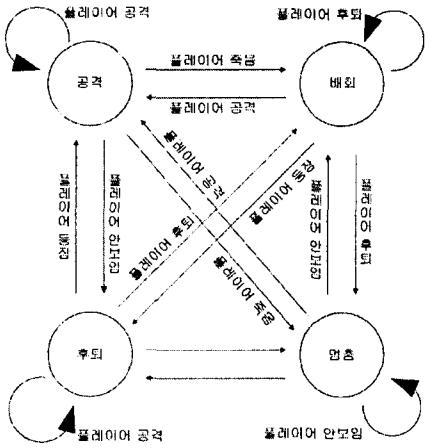

그림 1

[그림 1]에서는 상태 전이가 발생하는 이벤트에 아무런 제약 사항을 두지 않고, 해당 이벤트가 발생하면 바로 다음 단계로 상태가 전이되는 가장 기본적인 FSM을 만든 것이다.

여기에 NPC의 성향에 대한 내용을 추가하여 행동 패턴을 디자인한다면 보다 더 나은 행동 패턴을 만들 수 있다. 예를들면, NPC의 성향이 공격적이고 그룹을 형성하고 있다면, 플레이어 공격을 하기 위해 접근을 할 경우, 일단 같은 무리들이 있는 지역으로 유인을 하여 일정거리 이상 접근할 때, 주위의 무리들과 같이 공격을 하게 행동 패턴을 디자인 할 수 있다. 이러한 행동은 [그림 1]의 기본 행동에서 파생된 뛰어난 행동 패턴을 가지게 되는데, 그에 대한 상태 전이도가 [그림 2]에 나타나 있다.

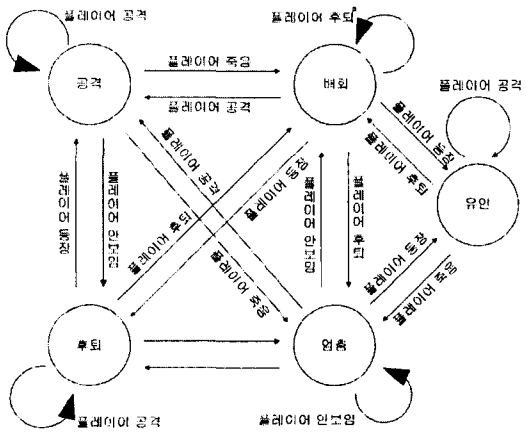

그림 2

3. 상태 및 상태 전이 이벤트 일반화

각각 생성된 FSM 상태 전이도를 기반으로 게임 디자이너가 툴을 이용하여 NPS의 행동 패턴을 디자인할 때 사용할 수 있게 일정한 규칙을 가지고 다시 정의를 해야 한다. 이 규칙을 메인 프로그래머는 인공지능 루틴 부분에 각 상태와 이벤트를 처리하기 위한 구현을 하게 된다.

본 논문에서는 각 상태와 상태 전이 이벤트를 다음과 같은 형태로 전부 일반화 시켰다.

1. 시작 상태
2. 전이 조건
3. 완료 상태

여기서 1.시작 상태와 3.완료 상태는 게임 디자이너가 정의한 NPC의 각 상태가 들어가게 되고, 2. 전이 조건은 각 상태가 다른 상태로 전이 되도록 하는 제약 사항이 들어가게 된다. 이 제약 사항에 여러 가지 인자를 적용하여 다양한 형태의 행동 패턴의 디자인이 가능하게 되는 것이다[그림 2].

4. 데이터 저장 구조

메인 인공지능 코드에서 정의되어 있는 여러 가지 상태와 상태 전이에 대한 리소스 트리를 이용하는 FSM 에디터 툴에서 만든 NPC 행동 패턴 오브젝트의 데이터 저장 구조는 다음과 같다.

```
unsigned nTransistions;
int      InputID;
int      OutputID;
int      Value;
int      StateID;
```

1. 'nTransistions'는 각 상태별 유한 상태 개수이며, 개수는 한 NPC가 가지는 모든 상태보다 하나 작은 수이다.
2. 'InputID'는 상태 전이를 위한 이벤트 ID로서 전체 이벤트들 사이에서 유일한 값을 가지고 있다. 이 값은 메인 프로그램의 인공지능 구현 부분에서 정의된 ID와 같은 값을 가진다.

3. 'OutputID' 상태 출력 ID로서 InputID와 마찬가지로 유일한 값을 가지고, 역시메인 프로그램의 인공지능 구현 부분에서 정의된 ID와 일치한다.
4. 'Value'는 각 상태를 전이 시키는 변수들의 설정값으로서 이 값을 이용하여 다양한 행동 패턴을 정의하게 된다.
5. 'StateID'는 상태의 고유 ID로서 유니크한 값을 가지고, 각각의 NPC가 동일한 상태에 있다면 ID 값은 같다.

이러한 데이터 값을 오브젝트를 정의한 데이터 파일의 한 부분에 추가를 하면 되기 때문에 NPC 행동 패턴 디자인을 위해서 또 다른 데이터 파일을 생성할 필요는 없게 된다.

5. 결론 및 향후 연구

게임에는 많은 종류의 NPC들이 존재하고 이러한 NPC들은 게임 내에서 정의된 인공지능 루틴에 따라 동작을 하게 된다. 이러한 인공지능 루틴을 NPC 종류에 따라서 전부 하드 코딩으로서 구현을 하기에는 무리가 있고, 구현을 한다고 하더라도 재사용이나 확장성이 떨어지게 된다. 그렇게 때문에 NPC 행동 패턴의 정의는 하드 코딩으로 구현을 하고, 하드 코딩된 패턴을 ID를 이용하여 외부의 오브젝트 파일에서 NPC의 행동을 정의하게 된다면, 재사용성이나 확장성은 하드 코딩으로 구현을 할 때보다 훨씬 높아지게 된다. 또, NPC의 행동 패턴을 디자인할 때, 직관적으로 행동 패턴을 알 수 있게 고안된 툴을 이용한다면 하드 코딩을 이용하여 구현했을 때처럼 디자이너가 게임 인공지능을 구현한 소스 코드에 대한 이해를 해야 하는 일은 발생하지 않을 것이다.

이것은 게임 디자이너에게 있어서 불필요한 시간 낭비를 방지하고, 게임 프로그래머에게 있어서는 재사용성과 확장성이 떨어지는 하드 코딩에서 벗어 날 수 있게 될 것이고, 플레이어에 있어서는 보다 다양한 NPC의 행동 패턴으로 인해 게임을 한층 더 재미있게 즐길 수 있게 한다.

향후 과제로서 FSM의 상태에 대한 일반화를 보다 더 효율적으로 할 수 있는 방법 모색과 FSM 이외의 인공지능 디자인에 본 논문에서 이용한 방법을 적용할 수 있는지에 대한 연구가 더 필요하겠다.

4. 참고 자료

[1] Richard Rouse 저, 최현호 역, 게임 디자인 이론과 실제, 정보문화사, 2001
[2] Andrew Rollings, Dave Morris 공저, 한쿨임 팀 역, Game Architeture and Design : 게임 아키텍쳐 & 디자인 1, 제우미디어, 2001
[3] S. Rabin 공저, 류광, 장원석 공역, AI Game Programming Wisdom, 정보문화사, 2003
[4] 2003년 동계 한국 게임 학회 학술