Journal of Digital Contents Society Vol. 7. No. 1. March. 2006(pp. 35-40)

게임 인공지능 기술의 제어 시스템 유형 및 문제점 연구

유선준*

요 약

게임 인공지능 기술은 등장 캐릭터의 지능적인 행동을 구현하고 플레이어에 의해 조종되지 않고 자동적으로 움직이는 NCP(Non Player Characters)의 행동을 제어하는데 요구된다. 본 논문에서는 이러한 게임 인공지능 기술의 제어 시스템으로 사용되는 Movement Scripts, FSM(Finite State Machines), Hierarchical State Machines, Fuzzy State Machines 방식과 캐릭터가 현재 위치에서 목적지까지 갈 수 있도록 이동 경로를 제어하는 Pathfinding techniques 방식을 기술하고 각 유형의 한계점을 비교 제시하였다.

A Study on Types and Limitations of Control Systems in Computer Game Artificial Intelligence

Sun-Joon Yu*

Abstract

Game AI(Artificial Intelligence) technologies implement the movement of autonomous characters and control the movement of Non-Player Characters(NPC). In this paper, we present several types of game AI control systems such as Movement Scripts, FSM(Finite State Machines), Hierarchical State Machines, Fuzzy State Machines, and Pathfinding techniques and their limitations.

Key Words : Game AI, Movement Scripts, FSM (Finite State Machines), Hierarchical State Machines, Fuzzy State Machines, Pathfinding techniques

1. 서론

시뮬레이션, 애니메이션, 의사결정 등 다양한 분야에서 활용되고 있는 인공지능 기술은 1990년대 후반부터 게임 분야에서 두드러지게 활용되면서 등장 캐릭터들의 행동을 보다 지능적으로 구현하여 게임의 재미를 더하고 있다.

기존에 컴퓨터에 의해 제어되는 캐릭터나 에이전트로 정의되던 '게임 인공지능'은 최근에는 좀 더 구체적으로 스스로 생각할 수 있고 주변 환경이나 과거의 경험 등에 따라서 지능적으로 행동할 수 있는 자율성을 지닌 캐릭터나 에이전트로 정의된다. 게임에서 인공지능은 등장 캐릭터의 지능적인 행동을 구현하고 플레이어에 의해 조종되지 않고 자동적으로 움직이는 NCP (Non Player Characters)들의 움직임이나 애니메이션 동작 등을 제어하거나 캐릭터가 현재 위치에서 목적지까지 갈 수 있도록 이동 경로를 제어하는 역할을 담당한다.[2]

캐릭터들의 자율적인 행동은 규칙 기반 추론을 기반으로 한 추론 엔진을 사용하여 이루어지고 있는데, 규칙 기반 추론 엔진은 비교적 간단하고 구현하기 쉽지만 규칙의 종류와 변화가 다양하지 않기 때문에 게임 플레이어가 쉽게 규칙을 파악하여 게임의 재미를 잃게 된다. 따라서 점점 더 캐릭터들의 행동 패턴을 복잡하고 다양하게 구현하여 예측하기 어렵게 만들어 현실적인 게임을 즐길 수 있도록 인공지능 기술이 요구되고 있다.

본 논문에서는 이러한 게임 인공지능 기술 중에서 플레이어에 의해 조종되지 않고 자동적으로 움직이는 NCP (Non Player Characters)의 움직임을 제어하는 시스템 유형을 살펴보고 각 유형의 문제점 및 한계점을 비교 제시하였다.

2. 제어 시스템의 유형

NPC의 움직임을 제어하는 시스템의 유형으로

※ 제일저자(First Author) : 유선준
접수일 2005년 8월 24일, 완료일 2005년 12월 5일
* 남서울대학교 멀티미디어학과 겸임교수
sunjyu@korea.ac.kr

는 간단하면서도 반복적인 움직임을 수행하도록 기술하는 Movement Scripts, 유한 개수의 상태로 전이되면서 NPC의 행동 양식이나 게임 세계를 관리하는 FSM (Finite State Machines), FSM에서 상태의 증가로 인한 문제점을 해결하기 위한 계층적 FSM 기술인 Hierarchical State Machines, FSM에 퍼지(fuzzy) 이론을 접목하여 행동 패턴을 다양하게 출력해 주는 Fuzzy State Machines, A* 알고리즘을 사용하여 장애물을 피해 목적지까지의 경로를 찾는 길 찾기 방법인 Pathfinding techniques 등이 있다. [1],[2]

2.1 Movement Scripts

Movement Scripts란 간단한 게임에서 캐릭터의 움직임을 제어하는 일반적인 용어로, 반복적인 움직임이나 몇 가지의 행동을 바꾸어 가면서 움직임을 나타낼 때 사용되는 제어 방법이다. 예를 들면, 단순한 움직임을 표현하기 때문에 적 캐릭터들이 특정 지역을 반복된 패턴으로 움직이거나 플레이어 접근 시 점프하여 이동하는 유형으로 표현된다.

2.2 FSM (Finite State Machines)

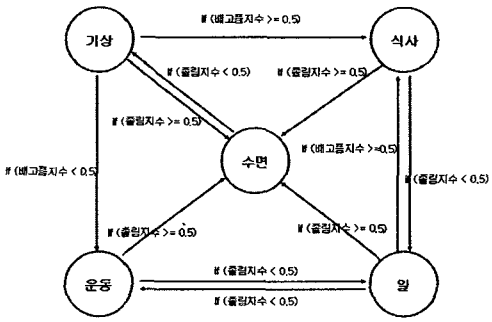
FSM (Finite State Machines)은 대부분의 게임 인공지능 엔진에서 사용되는 제어 시스템으로 유한한 개수의 상태들로 전이되면서 NCP (Non Player Characters)의 행동 패턴을 표현한다.

단순히 캐릭터에게 언제, 어떻게 움직이라고 표현하는 Movement Scripts 보다 한 단계 진보한 단계로, Movement Scripts로 이루어 져 있는 각 상태(state)가 결합되어 다양한 행동 패턴을 표현할 수 있어서 실제적으로 캐릭터들의 움직임을 더 지능적으로 구현할 수 있다. QuakeII에서는 FSM 방법을 사용하여 적들의 행동 패턴이 더 활발하고 플레이어가 대항하기가 어렵게 구성되었다.

FSM이 변형된 방법으로는 비결정적(Non-deterministic) 상태 전이, 상태 계층화 등의 변형 또는 현재의 상태와 환경적인 조건들 조합하여 유동적이고 연속적으로 새로운 상태로 전이하는 행동기반 시스템(behavior-based system) 기법 등이 있다.

다음은 캐릭터의 행동 양식을 표현하는 스크립트로 각 상태는 Movement Script가 될 수 있다. 이 스크립트에 따라 상태가 전이되는 단계는 그림1과 같다.

```
switch(state) {
   case 기상 :
      일어나기();
      if (배고픔지수 >= 0.5f)
         state = 식사;
      else
         stat = 운동;
      break;
   case 식사 :
      식사하기();
      if (졸림지수 >= 0.5f)
         state = 수면;
      else
         state = 일;
      break;
   case 운동 :
      운동하기();
      if (졸림지수 >= 0.5f)
         state = 수면;
      else
         state = 일;
      break;
   case 일 :
      일하기();
      if (졸림지수 >= 0.5f)
         state = 수면;
      else
         state = 운동;
      break;
   case 수면 :
      잠자기();
      if (졸림지수 < 0.5f)
         state = 기상;
      break;
}
```

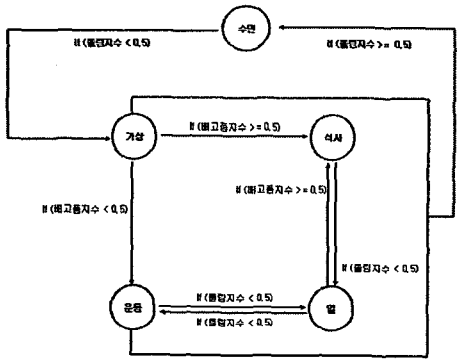


(그림1) 캐릭터의 FSM

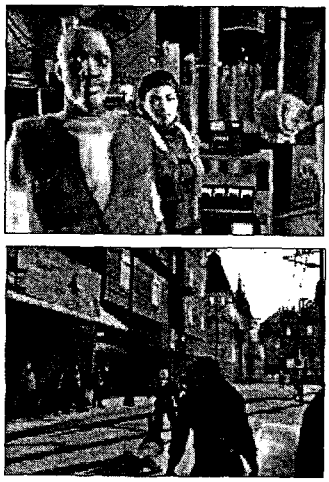
2.3 Hierarchical State Machines

Hierarchical State Machines 방식은 한 상태에서 다른 상태를 포함하고 있는 것으로 하위 상태들은 부모 상태가 연결된 다른 모든 상태에 효과적으로 연결될 수 있다. Hierarchical State Machines 방식은 연결 형태는 보다 단순하면서 움직임은 복잡하게 표현할 수 있다. 따라서 전이되는 상태 수는 줄이고 적절한 시점에서 적절한 행동을 구현하는 것이 가능하다.

그림1에서 표현한 캐릭터의 FSM을 Hierarchical State Machines 방식으로 표현하면 그림2와 같다. 그림1에서는 기상, 식사, 운동, 일 각 상태에서 졸림지수가 0.5이상인 경우 수면 상태로 전이되고, 졸림지수가 0.5미만인 경우 수면 상태에서 기상 상태로 전이된다. 이 경우 각 상태마다 졸림지수가 0.5이상이라는 조건을 따져서 수면 상태로 전이된다. 그렇지만 그림2의 Hierarchical State Machines 제어 시스템에서는 모든 상태에서 공통으로 졸림지수가 0.5이상이라는 조건을 따져서 수면 상태로 전이됨으로써 보다 효과적이라고 할 수 있다.



(그림2) 캐릭터의 Hierarchical State Machines



(그림3) 게임 Half-Life 2

Hierarchical State Machines 제어 시스템 방식이 적용된 게임의 예로 '하프라이프(Half Life)'를 들 수 있는데, '하프라이프'에서는 일련의 행동들을 스케줄로 묶어 계층적으로 처리하였다. 그림3은 밸프 소프트웨어사에서 2004년 11월에 발표한 '하프라이프2'의 장면이다.

2.4Fuzzy State Machines

Fuzzy State Machines는 전통적 논리에서 사용하는 참 또는 거짓 값 대신 실수(real-valued numbers)를 사용하여 degrees of membership을 표현하는 퍼지 논리(fuzzy Logic)를 기반으로 하는 state machine으로 Fuzzy State Machine을 사용하면 여러 상태에 적용된 동일한 조건을 전체적인 움직임에 적용할 수 있다.[6]

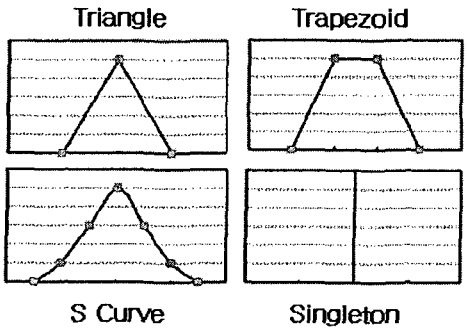
퍼지논리는 사실(facts), 객체(objects), 그리고 이들이 참이나 거짓으로 알고 있거나 알지 못하는 관계(relation)들을 다루는데, 지식을 표현하기 위해 이 세계에 존재하는 것에 대한 'degree of truth'와 사실에 대한 'degree of belief' '2가지 기법을 사용한다.[7]

현재 퍼지논리와 Fuzzy State Machines를 이용한 여러 상용 엔진이 존재한다. 예로, 퍼지논리 시스템을 쉽게 생성하고 응용프로그램으로 통합하는 기능을 제공하는 Louder Than A Bomb! Software사의 'Spark!'는 퍼지논리 시스템을 쉽게 생성할 수 있는 기능을 제공하는 퍼지논리 에디터이다.[8]

Spark!에서는 membership function으로 반환되는 반응에 따라 Fuzzy State Machine이 잠재 상태인지 활성(firing) 상태인지를 결정한다. 또한, 특정 시점에서 캐릭터를 제어하기 위한 상태를 결정하거나 특정 상황에서 여러 상태를 결합하기 위한 방법을 결정하기도 한다.[1]

다음은 Spark!에서 제공하는 membership function의 예이다. 각 membership function의 노드 포함 유형은 그림4에 나타나 있다.

Triangle : 퍼지논리에서 사용되는 가장 일반적인 membership function으로 3개의 노드를 포함하고 있다.
Trapezoid : 4개의 노드 포함
S Curve : 노드의 y값을 변경할 수 있게 해주는 유일한 membership function으로 여러 유형의 곡선을 만들 수는 있지만 교차하는 것은 허락하지 않는다.
Singleton : 퍼지논리의 특별한 경우로 non-fuzzy membership function이며 비연속적인 값을 표현한다.



(그림4) A Bomb! Software사의 Spark! - membership function 노드 연결 유형

2.5Pathfinding techniques

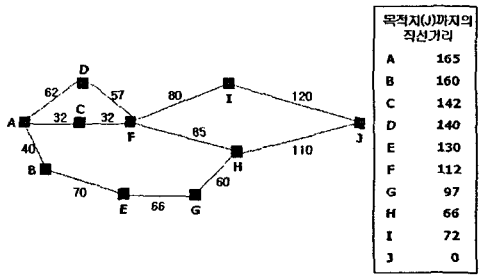
길 찾기는 게임에서 많이 사용되는 인공지능 기술 중 하나로 현재의 위치에서 목적지나 목표물까지의 경로를 찾는 제어 시스템 방식이다. 길 찾기 알고리즘 중 가장 많이 사용되는 방법은 A*(에이 스타) 알고리즘으로 탐색을 수행하는데 매우 효과적이다.

A* 알고리즘은 출발지점에서 목표지점까지 가장 비용이 적게 드는 경로 (최단경로)를 찾는데, 시작노드에서 현재 노드까지의 비용(goal)과 현재 노트에서 목표 노드까지의 예상 비용(heuristic cost)을 계산하여 다음에 이동할 경로를 결정짓는다. 예상 비용을 계산하는데 여러 방법이 있겠지만 일반적으로 현재 위치에서 목표까지의 일직선 거리 값을 이용한다. 이것을 식으로 표현하면 다음과 같다. f의 값이 작을수록 최단 경로일 가능성이 높다. [4]

$$f = g + h \quad (1)$$

f : fitmess
g : goal
h : heuristic cost

현재 위치에서 다음 노드로 이동할 때 현재 위치에 연결된 노드가 여러 개인 경우에는 시작 노드에서 f값이 가장 작은 노드를 선택하여 목표 노드에 도착할 때까지 탐색을 수행한다. 그림5는 시작노드 A에서 목적지 노드 J까지의 경로와 각 노드에서 목적지 노드까지의 직선거리를 나타낸다. 각 노드에서 목적지 노드까지의 직선거리가 예상 비용(heuristic cost)이다.



(그림5) 각 노드간의 거리와 각 노드에서 목적지까지의 직선거리

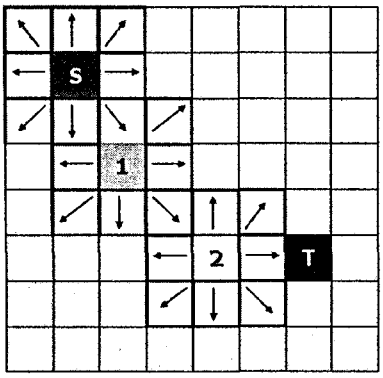
A* 알고리즘을 사용하여 시작노드에서 목표 노드까지의 최단 경로를 구하는 예를 살펴보자.

A노드에서 이동할 수 있는 다음 노드는 B, C, D 3개의 노드가 있는데 각 노드까지의 f값을 식 (1)을 사용하여 구하면 다음과 같다.

A->B : 40 + 160 = 200
A->C : 32 + 142 = 174
A->D : 62 + 140 = 202

위의 계산에 따라 f값이 가장 작은 경로를 선택하면 A에서 선택할 다음 노드는 C가 된다. 이렇게 하여 f값이 가장 작은 경로를 찾아 계속 탐색을 수행하면 목표 노드까지 최단 경로를 찾게 된다.

시작 노드에서 주변 노드를 검색하여 최단 경로를 찾아서 목표 노드까지 도달하는 예를 그림6에서 제시 하였다. 그림에서 굵은 테두리 영역은 이미 지나온 노드로 닫힌 상태를 의미한다. 목표 노드에 도달한 다음에는 저장되어 있는 상태 경로를 따라 시작노드에서 목표노드까지의 경로를 구한다. S는 시작노드, T는 목표노드, 1과 2는 S에서 T까지의 최단경로 상에 있는 노드를 나타낸다.



(그림6) 검색 노드를 확장하면서 목표 노드 도달

3. 제어 시스템의 한계

Movement Scripts는 간단한 움직임이나 두세 가지의 행동을 반복하는 데는 적합하지만, 복잡한 게임에는 적용할 수 없고, 플레이어가 게임 진행 중 유사한 상황을 몇 번 겪고 나면 캐릭터의 행동 패턴을 미리 예상할 수 있다.

FSM(Finite State Machines)은 단순하고 이해하기 쉬우며 작성이 용이하고 처리속도가 빠르다는 장점도 있지만 [3] 상태가 많아질수록 연결이 복잡해지고, 게임 플레이어가 게임을 여러 번 진행할 경우 유사한 패턴을 몇 번 반복하고 나면 캐릭터들의 움직임을 미리 예상할 수 있어서 게임의 재미가 반감될 수 있다. [2] 따라서, 상태들의 복잡한 연결 관계는 Hierarchical State Machines를 사용하여 상태들을 그룹화, 계층화하여 좀 더 단순하고 명확하게 표현할 수 있다. 그렇지만, FSM(Finite State Machines)과 Hierarchical State Machines 자체만으로는 어떤 일도 할 수 없으며 각 상태에서 사용되고 있는 다른 기술들에 전적으로 의존해야 한다.

Fuzzy State Machines는 퍼지이론을 접목하여 상태의 입력과 출력에 퍼지 함수를 적용하여 동일한 조건에서 여러 다른 출력을 얻게 함으로써 캐릭터들의 행동을 예측하기 어렵게 할 수는 있지만 membership function이 객관적인 개연성 없이 주관적으로 결정되고, 추론 과정에서 추론 규칙들을 비합리적으로 조합함으로써 최종 생성된 추론 결과에 오류가 많이 발생한다.

Pathfinding technique에서 사용하는 A* 알고리즘은 가장 널리 사용되는 방법으로 길 찾기에서 다양하게 응용되고 있으나 도중에 장애물을 만났거나 지형이 변경된 경우에는 A* 알고리즘만으로 해결할 수 없는 문제점이 발생한다. 또한 방대한 지형에서의 길 찾기인 경우에는 탐색해야 할 공간이 많아짐으로써 메모리의 낭비 등으로 탐색 효율이 떨어지게 된다. 이런 경우 시작 지점과 목표지점 사이의 중간 경유지를 생성하여 길 찾기를 단계적으로 수행하는 계층적 길 찾기 방법을 이용하기도 한다. [2]

4. 결론

본 논문에서는 게임 인공지능 기술 중에서 플레이어에 의해 조종되지 않고 자동적으로 움직이는 NCP (Non Player Characters)의 움직임을 제어하는 시스템 유형으로 Movement Scripts, FSM (Finite State Machines), Hierarchical State Machines, Fuzzy State Machines, Pathfinding techniques 방법과 각 게임 인공지능 기술이 갖는 한계점을 기술하였다.

Movement Scripts에서 Fuzzy State Machines까지는 점차 발전된 기법을 적용하여 NCP의 행동 패턴을 보다 다양하고 예측하기 어렵게 하여 게임을 좀 더 현실적으로 재미있게 즐길 수 있게 하는 것을 목표로 하고 있다. 현재 게임에서의 인공지능 기술은 지속적으로 발전하고 있고 하드웨어 및 그래픽 기술 등의 발전과 함께 인공지능 기술의 비중이 점차 높아지고 있다. 미래에는 인공지능이 캐릭터와 결합하여 더욱 복잡하고 정교한 형태로 발전할 것이다. 또한 보다 높은 수준의 인공지능 개발과 적용을 통해 실제의 생명체와 비슷하게 행동하는 사이버 생명체 구현이 가능하게 될 것이므로 인공지능 기술의 발전과 함께 그에 따라 대두되는 여러 문제점을 해결할 수 있는 방안도 제시되어야 할 것이다

.

참고 문헌

[1] Oliver Edward Wood, "Autonomous Characters in Virtual Environments: The technologies involved in artificial life and their affects of perceived intelligence and playability of computer games", MSc Thesis, University of Durham, 2004.

[2] 이헌주, "게임 인공지능 기술", 전자통신동향분석, 제20권, 4호, pp. 103-109, 2005년 8월

[3] 대전대학교(컴퓨터공학과), "초고속 정보통신망에서의 멀티미디어 컴퓨터 게임 소프트웨어 개발에 관한 연구", 1997년 7월

[4] 한글Softimage, http://www.softimage.com

[5] 정보통신연구진흥원, "온라인 슈팅게임을 위한 Full 3D 핵심기술 개발" 정보통신부 최종연구개발결과보고서, 2004년 5월

[6] Damian Isla and Bruce Blumberg, "Ai game

programming wisdom: Blackboard architectures", Charles River Media, 2002
[7] Russel and Norvig, "Ai: A Modern approach", Prientice Hall, 1995
[8] Louder than a bomb! Software, http://www.louderthanbomb.com
[9] M. van Lent and J. E. Laird, eds., "Artificial Intelligence and Interactive Entertainment," Tech. Report SS-01-02, AAAI Press, 2001
[10] K.W. Li, et al,"Fuzzy Approaches to the Game of Chicken", IEEE Transaction on Fuzzy Systems,2001
[11] E. Rasmusen, Games and Information, "An Introduction to Game Theory", Fourth Edition, 2005

유 선 준

1996년 : 미국 뉴욕시립대 (Baruch College) Computer Information System 전공 (석사)
1997~1997년 : 두산정보통신 기술연구소
1998~현재 : S & L 컴퓨터 기술 번역

1999~현재:남서울대학교 멀티미디어학과 겸임교수
2001년 : 고려대학교 컴퓨터학과 박사과정 수료