

유전자 알고리즘과 신경망을 이용한 MMORPG의 지능캐릭터 구현에 관한 연구

권장우[†], 장장훈^{††}

요 약

국내 게임시장은 MMORPG만을 생산하는 기이한 형태로 발전하고 있다. 하지만 지능형 캐릭터의 수준은 여전히 제자리걸음을 하고 있다. 본 논문에서는 유전자 알고리즘과 신경망을 사용하여 보다 뛰어난 지능을 가진 캐릭터 구현 방안을 제시하고자 한다. 또한 현재 MMORPG에서 사용되는 다른 인공지능 기술들과 비교했을 때, 그 성능이 뒤쳐지지 않음을 증명하고, 실제 MMORPG에 적용할 수 있는 구체적인 알고리즘과 구현 방법에 대해 설명한다.

A Study on Implementation of Intelligent Character for MMORPG using Genetic Algorithm and Neural Networks

Jang Woo Kwon[†], Jang Jang Hoon^{††}

ABSTRACT

The domestic game market is developmental in the form which is strange produces only the MMORPG. But the level of the intelligence elder brother character is coming to a standstill as ever. It uses a gene algorithm and the neural network from the dissertation which it sees and embodies the character which has a more superior intelligence the plan which to sleep and it presents it does. When also currently it is used complaring different artificial intelligence technologies and this algorism from the MMORPG, the efficiency proves is not turned over and explains the concrete algorithm it will be able to apply in the MMORPG and an embodiment method.

Key words: MMORPG(MMORPG), Genetic Algorithm(유전자 알고리즘), Neural Network(신경망)

1. 서 론

지금까지 국내의 게임시장은 MMORPG(Massively Multi-player Online Role Playing Game)만을 제작하는 기이한 형태로 발전했다. MMORPG는 말 그대로 수많은 사람들이 온라인에 접속하여 즐기는 롤플레잉 게임이다. 한때 전략 시뮬레이션 게임의 개발이 활발했지만, 온라인 게임의 상승세로 인하여 대부분 전략게임은 사장되었으며, 혹 개발되는 전략게임은

컴퓨터에 의해 제어되는 인공지능과의 대결이기 보다는 다른 플레이어를 상대로 대결을 하는 게임으로 제작 되었다. 스포츠 게임 역시 다른 플레이어와 경쟁하는 것을 위주로 개발하였으며 캐주얼 형태로 개발 되었다[1].

인공지능 기술의 발전에도 불구하고 국내 게임들은 그 적용사례가 적다. 특히 MMORPG에서는 다른 장르의 게임에 비해서 많은 수의 NPC(Non Player Character)를 처리해야 하기 때문에 NPC의

* 교신저자(Corresponding Author) : 권장우, 주소 : 부산시 남구 용당동 동명대학교 컴퓨터공학과(608-711), 전화 : 051)610-8394, FAX : 051)610-8349
E-mail : jwkwon@tu.ac.kr

접수일 : 2006년 11월 29일, 완료일 : 2007년 4월 18일

[†] 종신회원, 동명대학교 컴퓨터공학과

^{††} 정회원, 동명대학교 컴퓨터공학과

(E-mail : iceprince99@hanmail.net)

인공지능 처리에 그만큼 많은 시간을 투자할 수가 없다는 단점을 가지고 있다. 장비가 좋아지고 기술이 발달하고는 있지만 아직까지 대부분의 MMORPG가 FSM(Finite State Machine)[2]과 FuSM(Fuzzy State Machine)[3]등을 응용해서 사용하고 있다. 하지만 서버의 성능 향상에 따라 게임에서의 인공지능 분야는 점점 더 활발한 연구가 진행 되고 있으며, 보다 다양한 알고리즘들을 이용하여 NPC의 지능을 높이는 시도가 많아질 전망이다.[3]

본 논문에서는 국내외의 여러 게임에 적용하고 있는 유전자 알고리즘과 신경망 알고리즘을 MMORPG에 적용될 수 있는 방안을 제시한다. 본 논문의 구성은 2장에서 게임 인공지능의 배경 지식에 대해 설명하고, 3장에서 MMORPG의 유전자 알고리즘과 신경망 알고리즘을 이용하여 MMORPG의 지능형 캐릭터에 대한 설계 및 구현에 대해 기술하며, 4장에서 기존의 알고리즘과 차이점을 실험을 통하여 살펴보고, 마지막으로 5장에서 결론을 내리고 향후 연구를 제시한다.

2. 게임 인공지능 관련 연구

본 장에서는 게임에서 사용되는 인공지능 알고리즘들에 대해 알아보고, 각 알고리즘의 적용 사례를 간략하게 설명한다.

2.1 유한상태기계(Finite State Machine)

유한상태기계(Finite State Machine)는 MMORPG에서 가장 널리 사용되는 인공지능 처리 방식 가운데 하나이다. FSM이란 유한한 개수의 상태를 이용하여 NPC의 행동 양식을 표현하거나 게임 세계를 관리하는 방법이다. 여기서 상태란 NPC의 행동을 처리하기 위한 기본 단위가 되며, 각 상태는 주어지는 조건에 따라서 다른 상태로 전이될 수가 있다. 그림 1은 한 NPC의 행동 양식을 FSM으로 표현한 예이다. 그림 1의 예에서 볼 수 있듯이 몬스터의 행동 양식은 여러 개의 상태로 나누어지며 현재의 상태와 조건에 따라서 외부에 대처하는 방법이 결정된다[2].

표 1은 그림 1에서의 NPC가 나타낼 수 있는 모든 상태를 표현한 것이다. 이처럼 FSM은 이해하기 쉽고 구현도 어렵지 않아 특별히 뛰어난 인공지능을

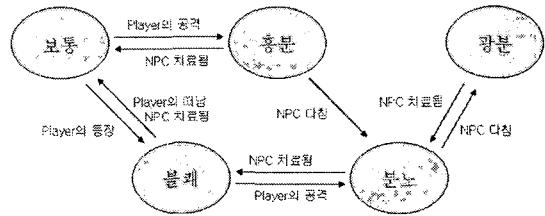


그림 1. FSM에서 NPC의 상태 변화

표 1. NPC 상태 정의

NPC의 현재 상태	외부 입력	NPC의 다음 상태
보통	Player의 등장	불쾌
보통	Player의 공격	홍분
홍분	NPC 치료됨	분노
홍분	NPC 치료됨	보통
분노	NPC 다침	광분
분노	NPC 치료됨	불쾌
광분	NPC 다침	광분
광분	NPC 치료됨	분노
불쾌	Player 떠남	보통
불쾌	Player의 공격	분노
불쾌	NPC 치료됨	보통

필요로 하지 않는 대부분의 MMORPG에서 사용된다. 그러나 복잡한 게임의 경우 상태의 수가 많아지게 되고 그에 따라 상태 디아이그램을 정리하기 어려워지며, 상태 변화를 가능하게 하는 외부 입력(조건) 루틴이 급속도로 복잡해진다. 또한 FSM을 게임의 상대편이나 몬스터로 적용한 경우 게임 진행 중 유사한 경험을 몇 번 겪고 나면 인공지능의 행동 패턴을 예측할 수 있게 되므로 게임의 재미가 반감될 수 있는 단점이 있다.

2.2 퍼지상태기계(Fuzzy State Machine)

FuSM(Fuzzy State Machine)은 FSM의 단점을 개선하기 위한 방법으로 퍼지 이론을 FSM의 상태 전이를 위한 조건 판단에 응용한 것이다. FuSM은 입력과 출력에 퍼지 함수를 적용하여 동일한 입력에도 다른 상태로의 전이를 얻을 수 있는 방법이다. 즉 FuSM은 “그렇다”와 “아니다”라는 이산적인 상태 대신 “매우 그렇다”, “그저 그렇다”, “다소 아니다”, “매우 아니다”와 같은 상태들을 표현할 수 있다[4].

퍼지 논리를 이용한 속도 제어 구현[5]에서는 퍼지 제어 이론을 적용하여 자동차의 속도를 제어하는 시뮬레이션을 제작하여 그 사용 예를 잘 보여주고 있다.

2.3 신경망

신경망은 인간의 신경 체계를 묘사한 것으로서, 수많은 간단한 커먼트들이 연결된 구조를 가지고, 입력되는 데이터의 패턴 인식에 기반을 두어 출력을 산출한다[7]. 신경망 알고리즘은 학습 능력을 가지고 있어서, 문자나 영상, 음성 등 특정한 패턴을 학습시키는데 사용되며, 게임에서도 많은 연구가 진행되고 있다.

간단한 하나의 신경망의 구조는 그림 2와 같다. n 개의 X 입력들이 각각 W라는 가중치에 의해 Y라는 뉴로드(인공 신경세포)에 연결되어 있다. 각 입력들은 각각의 가중치들과 곱해져 더해지고, 이 결과가 특정한 조건보다 크다면 출력을 통해 전달된다. 여기서 가장 중요한 점은 가중치 W들을 어떻게 설정하는가이다. 특정한 입력이 들어왔을 때 특정한 출력이 나오도록 하기 위해, 그 조건에 맞도록 적절하게 가중치들을 조정하여야 한다. 이처럼 특정 상황에 맞도록 가중치를 설정하는 것이 바로 정보의 저장, 또는 학습이라 할 수 있다. 보통 여러 개의 입력들을 반복적으로 신경망에 넣음으로써 점차적으로 가중치들을 원하는 방향으로 조절하게 된다. 문제는 어떻게 가중치들을 설정할 것인가인데, 이와 같은 학습의 방법에 따라 다양한 신경망 모델이 개발되었다. 신경망을 이용한 학습의 구현[8]에서는 위에서 설명한 신경망 모델의 여러 알고리즘 중 헵비의 규칙(Hebbian Learning Rule)을 사용하여 간단한 학습을 수행하는 게임을 구현하였다.

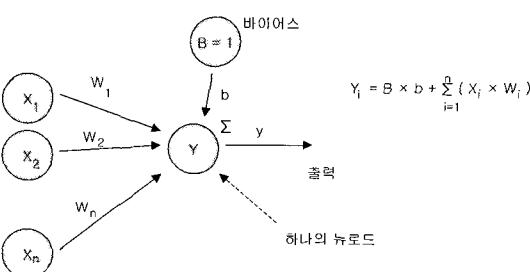


그림 2. 간단한 신경망 모델의 구조

본 논문의 3장에서 신경망 알고리즘을 응용하여 MMORPG에 적용하는 방안에 대해 논의한다.

2.4 유전자 알고리즘

유전자 알고리즘은 자연의 진화과정을 모델링 하여 구현한 탐색 알고리즘이라 할 수 있다.[7] 기본적인 개념은 주어진 문제에 대해 가능한 해들을 정해진 형태로 표현한 후 이들을 변화시키고, 그 결과들 중 적합도가 높은 것들을 선택하여 변화를 계속함으로써 보다 좋은 해에 도달한다는 것이다. 이는 마치 부모의 유전인자 중 우성의 형질을 지닌 개체가 번성해간다는 다윈의 진화론적 적자생존의 법칙과 유사하다.

주어진 문제에 가능한 해, 즉 각 개체들의 유전적 구조는 보통 고정된 길이의 이진 값으로 표현된다. 이러한 유전자는 다음과 같이 두 가지 방법으로 부모개체로부터 새로운 개체로 생성된다. 첫째는 교차변이(crossover) 또는 교배라고 하는 것으로 부모 개체들의 특정 교차점에서 그 이후의 값들을 서로 교환하는 것이다. 둘째는 돌연변이(mutation)로 0을 1로, 혹은 1을 0으로 비트 값을 반대 값으로 변환하는 것이다.

이렇게 새로운 자식 개체들을 생성한 후에 적합도 함수(fitness function)을 통해서 주어진 문제에 적합한 개체들만을 골라 선택하고, 다시 선택된 개체들을 교차변이와 돌연변이를 통해 새로운 개체들을 생성함으로써 보다 좋은 해에 도달할 때까지 반복한다. 이런 유전자 알고리즘을 이용하여 대전액션 게임의 지능형 캐릭터를 구현[9]할 수도 있지만, 본 논문의 3장에서는 유전자 알고리즘을 사용하여 MMORPG에서 다양한 지능형 NPC를 생성하는 방법을 논의한다.

3. 시스템 설계 및 구현

본 장에서는 신경망 알고리즘을 사용하여 MMORPG에 적합한 NPC의 학습 시스템을 구현하고, 유전자 알고리즘을 사용하여 MMORPG에서의 다양한 NPC 개체를 생성시키는 방안에 대해 기술한다.

3.1 신경망을 이용한 NPC의 학습

일반적으로 신경망 알고리즘은 특별한 패턴의 학

습에 사용한다. 신경망을 이용한 지능형 게임 캐릭터의 구현[6]과 신경망 지능 캐릭터의 게임 환경 변화에 대한 적용방법[9]에서는 신경망 알고리즘을 대전 액션 게임에서 상대 캐릭터의 행동에 따라 적절하게 반응하도록 학습시키는데 사용했다. 하지만 대전 액션이나 다른 장르에서 사용한 것처럼 신경망 알고리즘의 학습 단계는 MMORPG에는 서버에 부하만 줄 뿐이다. 대전 액션에서는 매번 결투마다 지능형 캐릭터가 단 하나만 존재하고, 스포츠 게임에서도 그리 많지 않은 지능형 캐릭터 수를 가지고 있다. MMORPG는 다양한 종류의 NPC가 동시에 다발적으로 연산을 수행해야 하기 때문에 복잡한 연산은 시스템 구현에 방해만 될 뿐이다. MMORPG에서도 신경망 알고리즘의 학습단계를 충분히 거친 후 게임에 적용시킨다면 서버의 성능에 큰 영향을 미치지 않겠지만 결국 단순히 학습된 결과로만 움직이는 일반적인 FSM 구조와 다를 바가 없게 된다.

(가) NPC의 신경망 구성

본 논문에서 구현한 NPC의 신경망 시스템은 그림 3과 같다. NPC는 Player와의 전투 경험을 저장하는 두뇌와 전투 경험을 학습하기 위한 신경망으로 구성된다. NPC는 Player와의 전투에서 NPC's Brain에 저장된 행동을 하거나 신경망으로 학습을 하면서 Player에게 적절한 행동을 한다.

그림 3과 같은 구조를 가진 NPC가 Player와의 전투가 시작되면 그림 4와 같은 순서로 행동을 결정짓게 된다.

먼저 전투가 시작되면 주위에 학습중인 동료가 있는지 확인한다. 같은 종류의 NPC 중에 Player와 전투를 하면서 학습중인 동료가 있다면 자신은 NPC's Brain에 저장되어 있는 이전 전투 경험으로 행동을

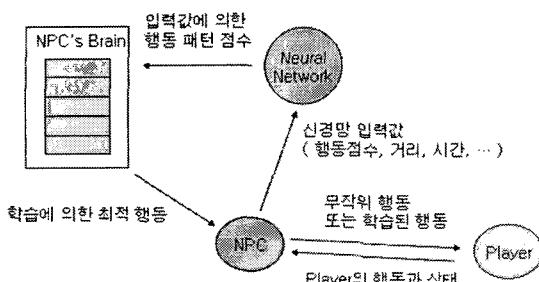


그림 3. 신경망의 시스템의 구조

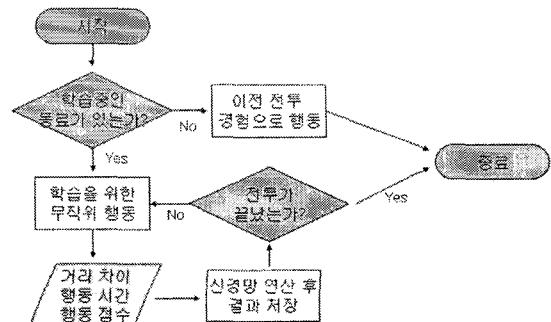


그림 4. NPC의 전투 학습 알고리즘

결정한다. 이는 동시 다발적으로 발생하는 학습량에 의해 서버가 과부하 되는 것을 막기 위함이다. 만약 학습중인 동료가 없다면 신경망에 의한 학습을 시작하게 되고 전투가 끝날 때까지 반복하게 된다. NPC의 신경망 구조는 그림 5에서처럼 입력계층, 은닉계층, 출력계층으로 구성되며 Player의 상태와 어떤 행동을 하는지, Player와 NPC 사이의 거리, 그리고 NPC의 행동을 입력 값으로 받아 각 계층 사이의 연결 강도를 조절하게 된다. 신경망 학습 방법은 (나)절에서 다시 언급하도록 한다.

NPC는 Player와의 전투에서 필요한 정보를 습득하고, 전투 경험이 많아질수록 Player의 행동과 상태에 따라 그에 알맞은 행동을 하게 된다. 먼저 Player와 NPC가 전투에서 사용할 수 있는 행동을 표 2와 같이 정의한다.

NPC는 그림 6과 같이 Player의 종류만큼 학습 결과를 저장할 수 있는 공간을 가지고 있으며, 같은 종류의 NPC는 이를 모두 공유하는 구조를 가지고 있다. 그룹 A에서 하나의 NPC가 학습을 시작하면, 다

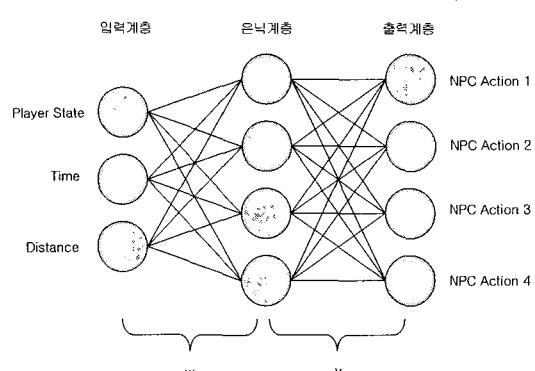


그림 5. 신경망의 구조

표 2. NPC와 Player의 능력 및 행동

Type	Health	Magic	Action
Player A	90	30	Attack, Skill A, B, C
Player B	50	80	Attack, Skill A, D, E
Player C	30	120	Attack, Skill B, E, F
NPC A	20	20	Attack, Skill D, G

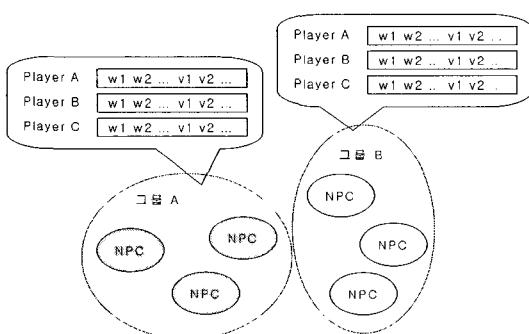


그림 6. NPC의 두뇌 구조

른 NPC들은 전투 중에도 학습을 하지 않는다. 즉, 하나의 NPC가 학습을 하는 것이 아니라 하나의 NPC 그룹이 학습을 하는 형태가 되는 것이다.

이런 구조를 가지게 된 배경은 MMORPG의 특성상 같은 종류의 NPC가 많기 때문이다. 또한 대전 액션 게임이나 스포츠 게임에서처럼 Player와 동등한 능력을 가진 개체가 아니라, 보다 약하게 설정하기 때문에 짧은 시간의 학습만으로도 원하는 결과를 얻을 수가 있다.

(나) 신경망을 이용한 전투 학습

Player와의 전투에서 학습을 시작한 NPC는 그림 3의 신경망 구조에 따라 매 턴마다 Player의 상태, 거리, 다음 행동까지의 시간 등의 입력을 받아 NPC가 할 수 있는 행동 중 하나를 수행하게 된다.

신경망은 일반적인 전방향 신경망(feedforward neural networks)을 사용한다. 다음은 입력 값을 이용하여 은닉 계층과 출력 계층의 노드들의 출력을 계산하는 식이다.

$$f_j = f\left(\sum_{i=0}^{N_i} (x_i w_{ij})\right) \quad (1) [7]$$

$$z_j = f\left(\sum_{i=0}^{N_h} (h_i v_{ij})\right) \quad (2) [7]$$

식(1)과 (2)에서, $f(x)$ 는 sigmid 함수, h_j 는 j번째 은닉 노드의 출력, N_i 는 입력 노드의 개수, x_i 는 i번째 입력 노드, w_{ij} 는 i번째 입력 노드와 j번째 은닉 노드 간의 링크 가중치, z_j 는 j번째 출력노드, N_h 는 은닉 노드의 개수, v_{ij} 는 i번째 은닉 노드와 j번째 출력 노드 간의 링크 가중치를 나타낸다.

NPC의 초기 학습 단계에서는 Player의 행동에 대한 출력을 예상하고 학습을 하는 것이 아니기 때문에 무작위로 자신의 행동을 결정한 뒤 그 결과를 기억해둔다. 여기서 결과는 Player의 HP와 NPC의 HP 변화량을 합한 것을 말한다. 초기 학습 단계를 거친 NPC는 강화 학습을 하게 되는데 “신경망을 이용한 지능형 게임 캐릭터의 구현”[6]에서 사용한 오류 역전파 알고리즘을 사용하여 이전 전투에서의 출력 값과 다음 번 전투에서의 출력 값을 비교하여 보다 나은 결과의 가중치를 높이도록 했다.

보다 나은 행동임을 결정짓는 점수 계산은 식(3)과 식(4)에 의해 계산된 결과로 판단한다. 식(3)에서 score는 Player의 HP와 NPC의 HP 변화량을 의미하며, z_j 는 j번째 출력 노드, 식(4)의 v_{ij} 는 i번째 은닉 노드와 j번째 출력 노드 간의 링크 가중치, $a(t)$ 는 학습률 함수를 나타낸다.

$$d_j = z_j \cdot (1 - z_j) \cdot score \quad (3) [6]$$

$$v_{ij} += a(t) \cdot d_j \cdot h_j \quad (4) [6]$$

(다) 신경망을 이용한 학습과 퇴화

MMORPG는 NPC의 신경망 학습에서 완벽한 해를 찾는 문제는 아니다. 다른 장르의 게임보다 많은 수의 NPC 개체가 소멸과 생성을 반복하는 게임이다. 초기 학습이 되어있지 않은 상태에서 상대한 Player와 많은 시간 반복된 학습을 통해 환경에 적응한 NPC를 상대하는 Player와의 형평성 문제도 생긴다. 이러한 이유 때문에 본 연구에서는 NPC의 퇴화라는 개념을 도입하여 NPC의 학습 절차를 역으로 돌리는 시도를 하였다. 원리는 그림 7과 같이 Player와의 전투 학습에서 전투 결과가 이전 경험보다 좋아서 연결강도를 변화 시킬 때 이전 경험의 연결강도를 전투 경험 스택에 넣어둔다.

만약 일정 시간동안 전투가 없거나 특별한 이벤트에 의해 NPC의 지능을 낮춰야하게 되면 전투 경험 스택에 있는 이전 연결강도를 현재의 연결강도로 바

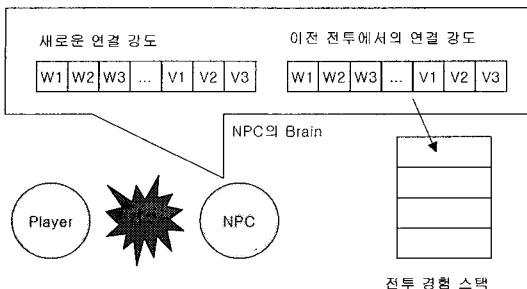


그림 7. NPC의 전투 경험 스택

꾸어준다. 이때 또 다시 학습을 통해 연결강도를 계산하지 않도록 현재의 연결강도를 저장해 두는 방법도 있다. 이런 구조는 시스템의 메모리 공간만 차지하게 될 뿐 신경망 계산에 걸리는 시간을 단축할 수 있는 장점이 있다.

4.2 유전자 알고리즘을 이용한 개체의 다양화

본 단락에서는 NPC의 능력과 성격을 결정짓는 수치 유전자와 성향 유전자의 동작에 대해 설명한다.

(가) 일반적인 방법의 NPC 생성

MMORPG는 게임의 특성상 NPC들의 생성이 극히 단순하다. 미리 정의해둔 수치로 NPC를 생성해내고, 그 NPC가 소멸되면 처음 생성할 때와 같은 데이터로 NPC를 다시 생성한다. 일반적인 MMORPG에서의 NPC 생성은 이 과정의 반복일 뿐이다.

표 3에서처럼 데이터베이스나 메모리에 NPC에 대한 기본적인 데이터를 준비해 둔다. NPC 객체를 생성하고, NPC A에 대한 데이터를 넣어주면 NPC A가 생성되고, 이 과정을 50번 반복하면 NPC A 개체가 50개 생성된다. 이 규칙을 응용하여 서로 다른 능력을 가진 NPC A 집단을 구성하려면 표 3의 정해진 수치를 기준으로 \pm 범위를 주면된다. 예를 들면 NPC A를 생성할 때 Health 수치를 정의되어진 90으로 사용하는 것이 아니라 난수를 발생시켜 기본적인 수치인 90에 난수 값을 더하거나 빼서 NPC A의 능

표 3. NPC Table

Type	Health	Magic	Power	Speed
NPC A	90	30	10	5
NPC B	50	80	8	4
NPC C	30	120	15	7

력을 각각 다르게 생성하는 것이다. 이 규칙의 단점은 특히 약한 집단을 생성하거나, 특별히 강한 집단을 생성하려면 NPC 생성을 담당하는 부분에 새로운 생성 규칙을 넣어야 한다.

(나) 수치 유전자를 이용한 NPC 생성

수치 유전자는 표 3에서처럼 미리 정의된 데이터에 영향을 주는 유전자이다. (1) 단락의 마지막에 제시된 문제를 해결하려면 NPC의 초기 유전자를 어떻게 설정하는가에 따라 강한 집단과 약한 집단, 그리고 두 집단의 섞인 집단을 각각 생성해 볼 수 있다.

먼저 표 4와 같이 Health에 영향을 주는 유전자를 4 bit의 수로 표현하고, 각 수치에 따라 알맞은 능력을 정의한다.

4 bit의 수는 16가지 상태를 표현할 수 있으므로 이를 백분율로 계산하면 수치 1은 6.25%의 영향력을 갖게 된다. 유전자에 이 공식을 적용하여 NPC Health 공식을 정의할 수 있다.

$$\text{Health} = \text{BaseHP} + \text{BaseHP} * ((\text{Genetic} - 8) * 6.25) / 100 \quad (5)$$

여기서는 4 bit의 중간 값으로 8을 선택했다. BaseHP는 표 3의 미리 정의 되어진 NPC의 Health이고, Genetic은 표 4에 정의된 유전자 중 하나이다. 만약 어떤 NPC가 Health에 영향을 주는 0110 유전자를 가지고 있다면 식(1)에 의해 NPC의 Health는 87.50가 된다. A 지역에는 특별히 Health가 높은 NPC만 배치하고, B 지역에는 Health가 낮은 NPC만 배치하고 싶다면 Health에 영향을 미치는 유전자를 조절하면 간단하게 해결 할 수 있다.

수치 유전자를 4 bit로 표현할 경우 모두 16가지 서로 다른 능력을 표현할 수 있으며, 이런 수치 유전자를 표 3의 Health 뿐만 아니라 Magic, Speed, Power 등에 함께 적용하면 유전자 배열은 16 bit가

표 4. NPC의 Health 유전자 정의

유전자	능력 설명	Health 영향력
0000 (0)	매우 허약함	$-8 * 6.25 = -50.00\%$
0001 (1)	조금 많이 허약함	$-7 * 6.25 = -43.75\%$
(생략)	(생략)	(생략)
1110 (14)	조금 많이 건강함	$+6 * 6.25 = +37.50\%$
1111 (15)	매우 많이 건강함	$+7 * 6.25 = +43.75\%$

되고 65536 만큼의 서로 다른 능력을 가진 NPC를 생성할 수 있다.

(다) 성향 유전자를 이용한 NPC 생성

성향 유전자는 NPC의 성격을 결정짓는 유전자이다. 본 단락에서는 NPC에게 성향을 나타내는 유전자를 제공함으로써 수치적인 능력의 차이를 보이는 NPC가 아니라 행동의 차이를 보이는 NPC를 생성하고자 한다.

어떤 시스템에서 생명이 위험한 상태가 되면 도망을 가는 NPC를 구현하고자 한다. 여기서 위험한 상태에 대한 정의를 하는 부분이 인공지능 기법에 따르게 정의 되는데 FSM에서는 “위험한 상태는 NPC의 Health가 10% 남았을 때”처럼 명확하게 정의해야 하며, NPC의 Health가 10%가 되면 어김없이 도망가는 행동을 보인다. FuSM에서는 “위험한 상태는 NPC의 Health가 5%(많이 위험한 상태), 10%(위험한 상태), 15%(조금 위험한 상태) 일 때”로 정의할 것이다. FuSM에서는 조금 차이가 있지만 가끔 15% 일때 도망가기도 하고, 5%가 되어서야 도망갈 때도 있다.

FSM과 FuSM에서 NPC의 Health 수치에 따라 도망가는 행동을 취하게 된다. Player는 이런 NPC와 몇 번 전투를 해보면 NPC의 도망가는 시기를 예측하게 되고 그 시점에 해야 할 행동을 준비하고 있게 된다. 하지만 본 논문에서는 이용하여 위험한 상태를 판단하는 기준을 위 두 경우처럼 수치를 정해 주는 것이 아니라 성향 유전자에 따라 결정되도록 구현하였다.

먼저 겁이 많은 유전자를 정의 하려면 그 반대의 의미를 가진 겁이 없는 성향을 함께 가지도록 한다. 즉, 표 5에서와 같이 겁이 많고 적음을 하나의 유전자로 표현한다. 여기서 도망갈 확률은 NPC의 Health 와 상관없이 도망갈 확률이고, 이 유전자에 의해

표 5. 두려움을 나타내는 유전자의 정의

유전자	성격	도망갈 확률
0000	매우 겁이 많음	$(16-0)*6.25 = 100\%$
0001	조금 많이 겁이 많음	$(16-1)*6.25 = 93.75\%$
(생략)	(생략)	(생략)
1110	조금 많이 겁이 없음	$(16-15)*6.25 = 6.25\%$
1111	매우 겁이 없음	$(16-16)*6.25 = 0\%$

Health가 떨어져도 불구하고 겁이 많은 NPC를 표현할 수 있다.

(라) 다양한 개체 생성을 위한 세대교체 방법

보다 다양한 NPC 개체를 생성하기 위해서 일반적으로 유전자 알고리즘에서 사용되는 교배 연산자와 돌연변이 연산자를 사용한다.

1) 교배 연산

교배 연산은 개체의 진화와 다양한 개체의 생성에 사용된다. 교배 연산에는 몇 가지 방법이 있는데, 본 시스템에서는 Single Point Crossover와 Two Point Crossover 방법을 사용한다.

가. Single Point Crossover

Single Point Crossover는 특정 위치의 한 자리를 서로 바꾸는 방법으로 그림 8과 같이 두 유전자의 어느 한 부분을 선택하여 유전자를 교체함으로써 이루어진다.

나. Two Point Crossover

Two Point Crossover는 그림 9와 같이 특정 위치의 두 유전자를 선택하여 교체하는 방법이다. 이 방법은 Single Point Crossover 보다 교배되는 유전자 수가 많아 보다 짧은 세대에 개체의 유전자가 폭넓게 변할 확률이 높다.

2) 돌연변이 연산자

개체의 유전자 스키마[10,11]를 극복하기 위해 사용한다. 그림 10에서와 같이 교배에 참여하는 유전자

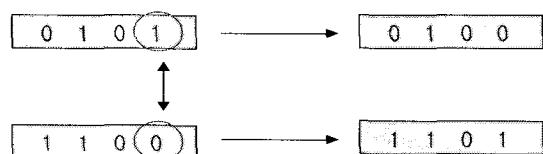


그림 8. Single Point Crossover의 예

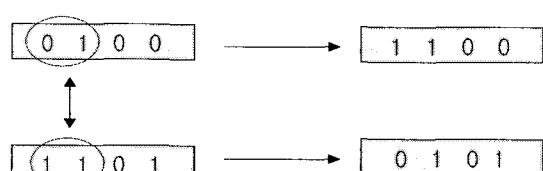


그림 9. Two Point Crossover의 예



그림 10. 유전자 스키마의 예

가 단 두 종류일 때, 유전자의 마지막 자리는 몇 세대를 걸쳐 교배를 하여도 1이 될 수가 없다.

이 문제점을 보완하기 위해 돌연변이(Mutation) 연산자를 사용한다. 돌연변이는 유전자 배열에서 특정 위치의 유전자를 바꾸는 연산자이다. 그림 10에서 두 유전자 배열을 교배한 다음 돌연변이 연산자에 의해 마지막 위치의 유전자를 0에서 1로 바꾸어 준다.

이 과정을 통하여 유전자 스키마를 극복할 수 있다. 여기서 돌연변이 연산이 발생할 확률은 테스트를 통하여 적당한 수치를 찾는 것이 좋다. 너무 많은 돌연변이가 생기면 초기에 설정해둔 건강한 유전자 그룹에서 세대교체를 몇 번 거치지 않고 허약한 유전자가 생기는 문제가 생기게 된다.

NPC의 생성에 유전자 알고리즘을 사용하여 다양한 종류의 객체를 생성시키고, 신경망 학습과 성향 유전자를 이용한 NPC 행동을 결정짓게 함으로써 Player로 하여금 NPC가 진화 또는 퇴화 하는 것처럼 보이게 하는 실험을 4장 실험 및 결과에서 확인해 본다.

4. 실험 및 결과

FSM과 FuSM 알고리즘을 이용하여 구현한 NPC와 신경망과 유전자 알고리즘을 이용하여 구현한 NPC의 상태 변화와 행동을 처리하는데 서버에 미치는 부하와 성능에 대해 비교해보고 MMORPG의 NPC를 설계할 때 고려해야 할 사항에 대해 알아본다.

4.1 실험 환경

실험은 신경망을 이용하여 구현한 NPC와 FSM과 FuSM에서 구현한 NPC와 비교했을 때 학습률을 확인하고, 학습 빈도수에 따른 성능 실험을 하여 MMORPG에서 실시간 학습이 전체 성능에 미치는 영향을 알아본다. 또한 유전자 알고리즘을 사용하여 개체 수의 다양화와 각 무리군의 변화를 확인한다.

4.2 신경망을 이용한 학습의 빈도수에 따른 성능 실험

3종류의 서로 다른 능력을 가진 Player를 5 유닛 생성하고 정해진 패턴으로 NPC를 공략하도록 했다. 그림 11은 Player 유닛을 하나씩 추가하면서 NPC의 학습 빈도를 확인한 결과 Player가 아무리 늘어나도 NPC 개체의 그룹 수 이상은 학습을 하지 않음을 나타내는 그래프이다. 그림 12는 학습 빈도에 따른 신경망 학습 시간을 도표로 나타낸 것이다. 이 결과로 비추어볼 때 실제 서비스하는 MMORPG의 NPC 개체 그룹의 종류가 1000종이 동시에 학습을 한다고 해도 시스템에 큰 무리를 주지 않음을 예측할 수 있다.

표 6. 실험 환경

대상	설명
실험 시스템 사양	Pentium 4, 1.5 GHz / 1 GB
사용 언어 및 툴	C++, 자체 개발한 NPC 전투 테스트 툴
실험 NPC	5종류의 NPC 각 1000개씩 총 5000 유닛
실험 Player	3종류의 서로 다른 능력을 가진 캐릭터

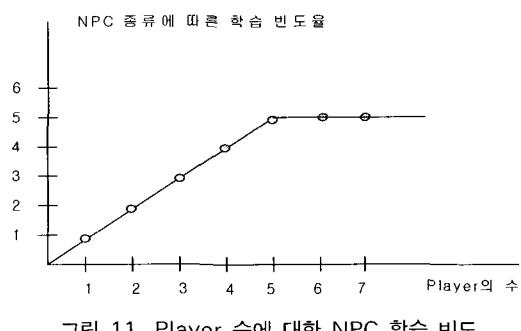


그림 11. Player 수에 대한 NPC 학습 빈도

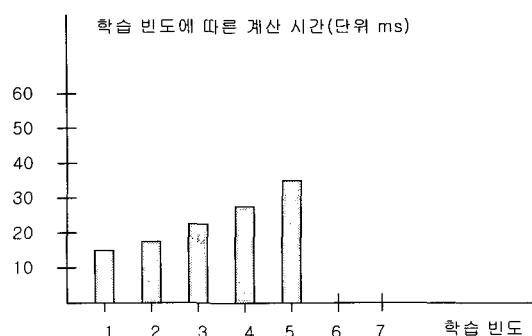


그림 12. 학습 빈도에 따른 신경망 학습 시간

그림 13은 FSM에서의 행동 결정 시간과 신경망 학습을 통한 행동 결정 시간의 차이를 나타낸 것이다. 학습을 하는 동안 행동을 결정하는데 다소 많은 시간이 걸릴 수 있다. 하지만 사용자들이 게임을 즐기기에 무난한 시간은 300ms이므로 신경망 학습의 행동 결정 시간은 별 문제가 되지 않는다. 또한 그림 14에서 보는 바와 같이 이전에 학습한 결과를 가져와서 행동을 결정할 때에는 FSM에서 행동을 결정하는 것과 별반 차이가 없음을 확인 할 수 있다. 이러한 결과로 볼 때 학습하는 빈도를 줄임으로써 신경망 학습을 실시간으로 수행하면서 지능이 변하는 NPC를 구현할 수 있음을 확인 할 수 있다.

4.3 유전자를 이용한 NPC 개체의 다양화 실험

3장에서 제시한 특별히 강한 집단과 특별히 약한 집단을 각각 다른 지역에 배치시키고 각 세대가 변화할 때마다 개체의 유전자가 어떻게 진화하고 그 능력의 차이가 의도한 대로 다양한 개체가 생성되는지 실험해 보았다. 그림 15에서는 특별히 강한 유전자들을 가진 NPC, 그림 16에서는 특별히 약한 유전자들을 가진 NPC의 세대교체에 따른 유전자 변화를 나타낸 표이다. 매번 실험 때마다 비슷하긴 하지만 다

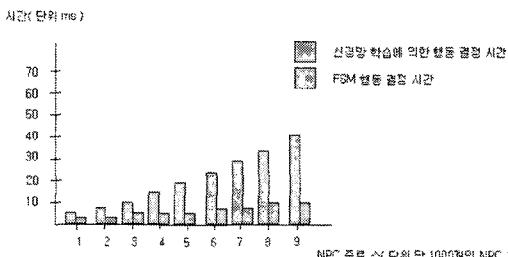


그림 13. 신경망과 FSM에서의 NPC 행동 결정 시간 비교



그림 14. 이전 전투 경험과 FSM에서의 NPC 행동 결정 시간 비교

른 결과 값이 산출되어 결과를 예측하기가 어려웠다. 그럼 16에서는 돌연변이에 의해 특별히 약한 집단에서 강한 유전자를 가진 개체가 생성되었고, 그로 인하여 약한 집단이 강한 집단으로 변해가고 있음을 확인할 수 있다.

그림 17은 겁이 많고 적음을 나타내는 성향 유전자의 개체 번식을 도식화 한 것이다. 이러한 유전자를 가진 NPC는 세대가 변함에 따라 Player에게 매번 다른 성향을 보여주면서 NPC의 지능이 변하는 것처럼 보이게 만드는 효과가 있다. 이러한 성향 유전자의 세대 교체율 또한 테스트를 통하여 적당한 수치로 결정해야 한다.

그림 18은 겁이 많음을 나타내는 성향 유전자를 가진 NPC가 실제 해당 성향을 발생시키는 빈도를 나타낸 것이다. 게임의 콘텐츠에 따라 유전자에 미치는 수치를 조절하면서 발생 빈도를 바꿀 수 있으며, 난수 발생 방법에 따라 그 결과에 조금씩 차이가 있을 수 있다.

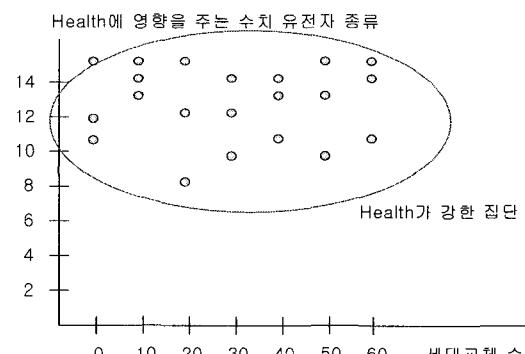


그림 15. 특별히 강한 유전자를 배치한 NPC 무리

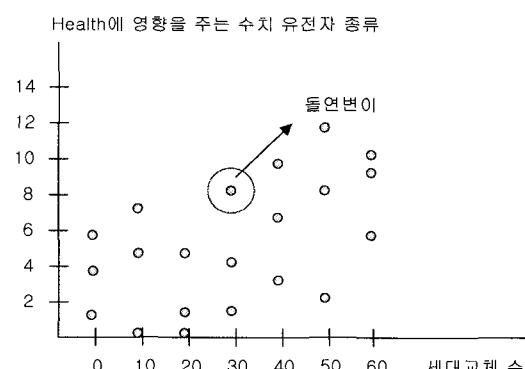


그림 16. 특별히 약한 유전자를 배치한 NPC 무리

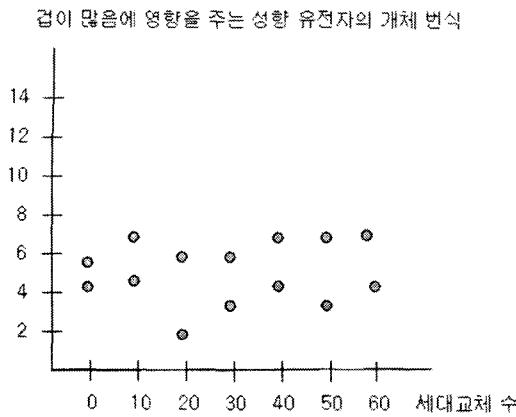


그림 17. 겁이 많음에 영향을 주는 유전자의 개체 번식

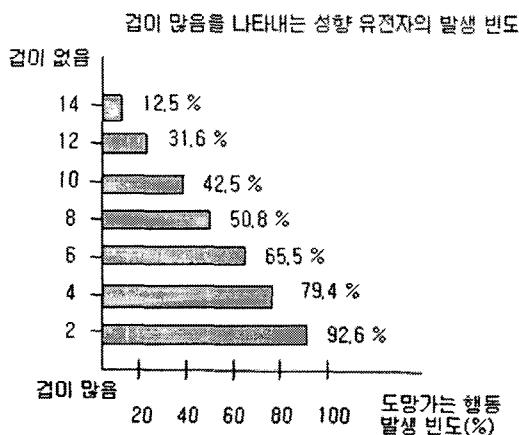


그림 18. 겁이 많음을 나타내는 유전자의 성향 발생 빈도

5. 결론 및 향후 연구

본 논문에서는 신경망과 유전자 알고리즘을 이용하여 대부분의 MMORPG에서 사용되는 FSM과 FuSM 알고리즘을 대체할 수 있는 방안을 제시하였다. 단순한 패턴에 의한 움직임이 아니라 NPC들이 Player들에게 적응하고 보다 다양하게 대응하도록 하는 것이 주 목적이었다. 너무 뛰어난 반응을 하는 것도 게임의 재미를 반감시키는 요소가 되기 때문에 학습과 트리밍을 반복 하도록 하는 것이 중요하다. 설계된 시스템에서 MMORPG의 NPC가 신경망을 이용하여 학습하는 것이 더 이상 서버에 부하만 주는 알고리즘이 아니라는 것을 알 수 있었으며, 유전자 알고리즘을 적절하게 조합하여 사용했을 때 더

욱 다양한 NPC를 표현할 수 있다는 사실도 확인하였다. 이처럼 MMORPG의 NPC는 보다 다양한 인공지능 기법들의 조합으로 구현될 가능성이 있고, 인공지능의 발전으로 게임의 다양성에도 크게 기여할 것으로 보인다.

본 연구를 토대로 현재 국내 MMORPG에 사용할 수 있는 학습, 진화형 NPC를 구현하고 있으며, 무리를 지어 행동하는 NPC들에게 협동에 대한 인공지능을 부여하는 방법도 연구 중에 있다. 또한 본 연구에서 사용한 알고리즘을 토대로 범용적인 Game AI 엔진을 개발 중에 있으며, 향후 개발되는 MMORPG에 적용할 계획이다.

참 고 문 헌

- [1] 이현주, “게임 인공지능 기술,” 전자통신동향분석 제20권 제4호 Aug. 2005.
- [2] Eric Dybsand, *Finite State Machine Class, Game Programming Gems* pp. 319-332.
- [3] 이만재, “게임에서의 인공지능 기술,” 한국정보처리학회지, Vol. 9, No. 3, pp. 69-76, May. 2002.
- [4] 김성완, “퍼지를 이용한 인공지능,” <http://www.g-matrix.pe.kr/feature/ai/fuzzy.htm>.
- [5] 우종하, 퍼지 논리를 이용한 속도 제어 구현, <http://gameai.net/Article/FuzzyRace/FuzzyRace.htm>.
- [6] 조병현, 정성훈, 성영락, 오하령, “신경망을 이용한 지능형 게임 캐릭터 구현,” 한국정보처리학회 논문지, 제11-B권 제7호, pp. 831-840, Dec. 2004.
- [7] Francois Laramee, *AI Game Programming, Wisdom*, pp. 841-853.
- [8] Andre LaMothe, *Game Programming Gems*, pp. 429-455.
- [9] 조병현, 정성훈, 성영락, 오하령, “신경망 지능캐릭터의 게임 환경 변화에 대한 적응 방법,” 전자공학회 논문지 제42권 C1 제3호, May 2005.
- [10] Kwee-Bo Sim, “유전자 알고리즘의 스키마” <http://alife.cau.ac.kr/info/EAs/GAs2.html>.
- [11] 이면섭, 조병현, 성영락, 정성훈, 오하령, “유전

자 알고리즘을 이용한 대전형 액션게임의 지능형 캐릭터,” 한국정보처리학회 논문지 B 제 12-B권 제3호, Jun. 2005.

- [12] Craig Reynolds, Boids, <http://www.red3d.com/cwr/boids/>.

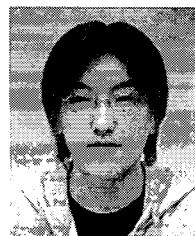


권 장 우

1990년 2월 인하대학교 전자공학과 졸업(공학사)
1992년 2월 인하대학원 전자공학과 (정보공학 전공) 공학 석사
1996년 8월 인하대학원 전자공학과 (정보공학 전공) 공

학 박사

1996년 10월 ~ 1998년 2월 특허청 심사관
1998년 3월 ~ 현재 동명정보대학교 컴퓨터공학과 부교수
2004년 4월 ~ 2006년 12월 정보통신연구진흥원 전문위원
관심분야 : 인공지능, 신경망, 신호처리, USN 등



장 장 훈

2002년 2월 동명대학교 컴퓨터 공학과 졸업(공학사)
2006년 8월 동명대학원 컴퓨터 공학과 (컴퓨터공학 전공) 공학 석사
2004년 3월 ~ 2006년 2월 (주)태울엔터테인먼트
2006년 4월 ~ 현재 (주)엠파스 정보통신연구소
관심분야 : 게임 인공지능, 신경망, 유전자 알고리즘 등