플레이어 행동예측을 위한 순차예측 알고리즘의 개선

Improvement of Sequential Prediction Algorithm for Player's Action Prediction

신 용 우* 정 태 충**
Yong-Woo Shin Tae-Choong Chung

요 약

게임은 여러 캐릭터와 상태공간을 갖고 있다. 그러므로 학습을 하는데 많은 시간이 걸린다.
본 논문에서는 강화학습 알고리즘을 이용하였다. 보상 값을 받아 캐릭터가 학습하게 하여 지능적으로 움직이게 하였다. 학습초기에는 학습속도가 느려진다. 순차예측 알고리즘을 개선하여 학습에 적용하였다. 기존 강화학습으로 구현된 게임과 비교 실험하였다. 실험결과 개선 구현된 게임의 성능이 학습속도 측면에서 30% 까지 향상됨을 알 수 있었다.

ABSTRACT

It takes quite amount of time to study a game because there are many game characters and different stages are exist for games. This paper used reinforcement learning algorithm for characters to learn, and so they can move intelligently. On learning early, the learning speed becomes slow. Improved sequential prediction method was used to improve the speed of learning. To compare a normal learning to an improved one, a game was created. As a result, improved character's ability was improved 30% on learning speed.

☞ KeyWords : 강화학습(reinforcement learning), 게임(game), 학습속도(learning speed), 행동예측(action prediction)

1. 서 론

2007년에 세계 온라인게임 시장규모는 70억 달러를 기록했으며, 이는 전년 대비 24.7% 의 성장을 달성하였고, 이후에도 지속적으로 성장하여 2010년에는 132억 달러 규모로 성장할 것이다 [1]. 게임 산업이 날로 발전하고 대규모로 제작되므로, 게임프로그래밍도 인공지능과 같은 분야에 투자하여 부가가치를 만들어야 할 것이다. 과거에는 엔진과 게임프로그램으로 나누어 개발하던 시점에서 이제는 좀 더 세분화하여 분야별 전문프로그래머가 필요하게 되었다.

* 정 회 원 : 동아방송예술대학 디지털영상과 교수
ywshin@dima.ac.kr
** 정 회 원 : 경희대학교 컴퓨터공학과 교수
tcchung@khu.ac.kr
[2010/01/25 투고-2010/01/26 심사(2010/03/018 2차) - 2010/04/26 심사완료]

인공지능 분야에서도 현재까지 캐릭터의 이동처리를 위해서는 패턴(Pattern), A* 알고리즘이나 유한상태기계(Finite State Machine), 퍼지(Fuzzy), 퍼지상태기계(Fuzzy State Machine) 을 이용하여 캐릭터의 자동화를 하고 있다. 패턴이란 미리 주어진 방향으로 캐릭터가 이동하게 하는 단순한 논리이다. 유한상태기계는 캐릭터의 여러 가지 행위를 상황에 따라 적용시키는 알고리즘이다. 또한 캐릭터의 움직임을 다양화 시키는 퍼지나 퍼지상태기계 그리고 길 찾기에 사용되는 A* 알고리즘 등은 캐릭터가 주어진 방향이나 목적지를 찾는 데에는 유용하게 사용할 수 있으나 캐릭터의 근본적인 지능을 높여주지는 못한다.

패턴이나 유한상태기계를 이용한 캐릭터의 이동방법은 경우의 수가 작게 제한되어 단순하고 보다 많은 경우의 수를 가지는 캐릭터의 전투상황에서 효율적인 전투를 벌이지는 못한다. 그러므

로 학습 알고리즘을 적용하여야 한다.

강화학습은 온라인 학습기법으로 캐릭터를 학습시킨다. 강화학습을 이용하여 특정목적을 달성하였을 때마다 보상을 하는 작업을 반복하다보면 캐릭터가 학습하게 된다 [2].

강화학습 분야에는 상태공간의 문제, 캐릭터의 자동화 등의 문제가 있는데, 상태공간의 효율적인 사용 등의 문제는 여러 논문에서 다루었다. 그러므로 본 논문에서는 캐릭터의 자동화부분을 다루고자 한다. 기존의 캐릭터의 자동화를 다룬 논문들은 서양보드게임인 오델로, 틱택토 등을 다루었으나[6][7] 본 논문에서는 슈팅게임을 다루고자 한다.

일반적인 강화학습으로 캐릭터를 학습시킬 때, 캐릭터가 학습이 완료되었을 때는 최적 값을 검색하여 원하는 곳으로 이동하도록 활용하면 된다. 그러나 캐릭터를 학습시키는 과정에서 많은 시간이 소요되게 된다. 처음에 아무런 지식이 없는 캐릭터 에게 지식을 조금씩 주입하는 과정에서 참고할 수 있는 자료가 많지 않아 적절하지 못한 값이 주어지고 그것은 학습시간의 낭비가 된다.

본 논문의 필요성은 이러한 부분에 있다. 학습 초기시간의 단축을 위하여 순차예측 알고리즘을 개선한 알고리즘을 제안한다. 본 논문에서 제안한 방법은 게임의 장르에 관계없이 학습이 필요한 게임에서 활용이 가능하다. 학습 시간의 단축이 가능하게 한다.

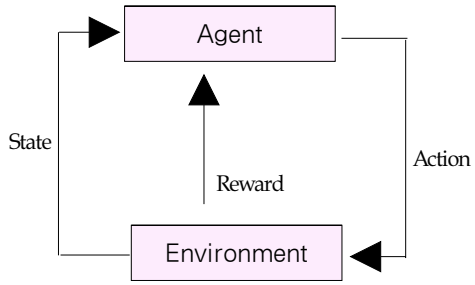
논문 [2] 에서는 강화학습을 이용하여 슈팅게임을 구현하였다. 그러나 학습과정에서, 학습이 많이 일어나지 않은 초반부의 학습속도가 다소 느려졌다. 이러한 현상은 학습이 많이 이루어지지 않아, 선택할 수 있는 값이 많지 않기 때문이다. 본 논문에서는 이런 단점을 없애기 위하여, 학습과정에서의 최선 값을 산출하는 부분에서 학습되지 않은 동일한 값이 나올 때, 제안하는 행동예측 방법으로 유리한 값을 선택하도록 하였다. 실험결과 [2] 에서의 게임보다 개선된 강화학습을 적용한 게임이 우수한 경기를 벌이는 것을 알 수 있었다.

본 논문의 구성은 1장 서론에 이어 2장에서는 관련연구에 대해 살펴보고 3장에서는 지능형 게임의 구현에 대해 알아보고 4장에서는 실험 및 결과에 대해 알아본다. 마지막으로 5장에서는 최종 결론을 맺도록 한다.

2. 관련연구

2.1 강화학습

강화학습이란 많은 상태들의 집합인 환경에서 목적달성을 위한 행위를 수행하여 보상을 받음으로 필요한 행위를 학습하는 것을 말한다.



(그림 1) 강화학습의 모델

환경에는 목적달성에 필요하거나, 필요하지 않은 많은 상태들이 존재한다. 에이전트는 각각의 상태를 경험하여 목적달성의 경우 환경으로부터 보상을 받게 된다. 목적달성을 할 수 없는 상태인 경우 보상을 받을 수 없다. 그러므로 많은 시행착오를 겪을 수 있으며, 모든 상태를 경험해 보아야 한다.

강화학습에는 시간차이(Temporal Difference), Q-learning, SARSA (State-Action- Reward-State-Action) 알고리즘 등이 있다. 시간차이 알고리즘은 상태 전이와 보상 값을 바탕으로 반복적으로 가치 함수를 학습한다. SARSA 는 결정된 전략에 따라 α_{t+1}을 선택한다. Q-learning은 사전에 모델을 설정하거나, 학습할 필요가 없으며, 상태공간을

충분히 경험한다면, 최적의 전략을 만들 수 있어 다양하게 활용되고 있다.

강화학습알고리즘으로서 일반적으로 많이 사용되는 Q-learning 에 대해 알아보자.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_{t+1} + \gamma \max_{a} Q(s_{t+1},a') - Q(s_t,a_t)) \quad \text{(식1)}$$

특정상태 s 에서의 행위 a 에 대한 보상 r 을 얻을 수 있도록 식이 주어져 있다 [2]. t 는 이전, t+1 은 이후시간단위를 말한다. max 란 최대값을 구함이고, α는 학습속도매개변수 로서 0에서 1 사이의 범위이다. γ는 할인계수로서 0 에서 1 사이의 범위이다.

모든 가능한 상태공간에서의 특정상태 s 에서의 행위 중 가장 좋은 보상 r값을 저장하고 산출한다. 처음에는 모든 상태에 대하여 보상을 얻기 위해 무작위로 행위를 하여 경험을 쌓게 된다. 그러나 시간이 지나 경험한 상태가 많아질수록 보상을 토대로 불필요한 상태의 경험을 필요치 않는다. 거의 모든 상태를 경험함으로서 학습이 완료되어 지능적으로 동작하게 된다.

2.2 뎀스터 샤퍼 이론 (Dempster Shafer Theory)

뎀스터샤퍼 이론이란 하나의 증거로부터 신념을 계산하기보다, 여러 가지 증거들이 서로를 확인한다는 사실을 활용한 이론이다.

$$Combined(N) = \frac{\sum_{X \cap Y} S1(X) * S2(Y)}{1 - \sum_{X \cap Y} S1(X) * S2(Y)} \quad \text{(식2)}$$

X는 첫 번째 증거의 힌트이고, Y는 두 번째 증거의 힌트, 그리고 Si(A) 는 출처 i의 사건 A에 관련된 신념의 질량이다. 출처들이 제공한 증거를 행렬의 곱으로 결합한다 [3].

1). 두 힌트들에 관련된 사건들의 집합 X와 Y의 교집합 N을 구한다. 두 사건이 모순된다면 공집합이 나온다.

2). 두 힌트에 관련된 신념들을 곱하고, 사건 N에 대한 결합된 신념분포의 곱을 더한다.

3). 분포를 정규화하기위해, 모순된 힌트들에 관련된 신념 질량들의 곱을 1에서 뺀 값으로 분포를 나눈다.

2.3 순차 예측 (Sequential Prediction)

순차예측은 어떠한 값들의 열(sequence) 이 주어졌을 때 그 다음에 나올 값을 찾는 문제이다 [4]. 순차예측을 이용하여 패턴(pattern) 을 인식하는 것이 가능하다. 주어진 문자열에 패턴이 존재하면 예측가능성이 높아지는 경향이 존재한다. 반복적 패턴이 발생할 때 예측이 수월해 진다. 문자열 부합 예측이라는 기법을 사용할 수 있다. 다음과 같은 문자열이 존재한다고 하자.

B A A B A B B A B B B A A A A B A A A B B A B

열의 끝에서부터 한 글자씩 늘려가며 일치하는 부분 문자열들을 찾는다. 처음에는 B를 찾는다. 다음으로는 A B를 찾는다. B는 여러 번 존재한다. A B 또한 여러 번 존재한다. 그 다음으로는 B A B를 찾는다. B A B 도 여러 번 존재한다. 이런 식으로 찾다보면 A B B A B 까지 찾을 수 있다. 이렇게 열의 끝에 부합하는 가장 긴 부분 문자열을 찾아낸다. 이 부분문자열의 다음 값은 B 이므로 우리는 다음에 나올 값으로 B를 예측할 수 있다.

2.4 N그램 통계모형을 이용한 플레이어 행동 예측

적과 싸우는 과정에서 방어는 공격만큼 중요하다. 효과적인 방어를 위해서는 적의 행동을 분석하여 다음 행동을 예측, 추정할 수 있어야한다. 국소적인 맥락과 구조에 따라 사건의 확률이 증가하거나 감소하는 성격의 문제들은 N그램(N-gram)

이라는 통계모형을 통해서 풀 수 있다 [5].

1 2 1 2 1 2 1 2 1 2 2

위와 같은 문자열이 있을 때, 다음에 나올 글자를 예측한다고 하자. 먼저 기존에 나온 문자열을 분석하여야 한다. 현재까지는 2 가 1 뒤에 나오는 경우가 네 번이고, 2 가 2 뒤에 나오는 경우가 한 번 있다. 그러므로 1 뒤에 2 가 나올 확률은 80%, 2 뒤에 2 가 나올 확률은 20% 라고 볼 수 있다.

N그램은 기존에 발생했던 긴 열을 분석하여, 정규적인 부분 열들을 분석하여 다음글자를 예측한다.

2.5 최근접 문자열 부합 알고리즘

임의의 두 문자열이 얼마나 잘 부합하는지를 파악하는데 사용할 수 있는 알고리즘이다[10]. 주어진 문자열과 정확히 일치하는 문자열 ID가 없는 경우 주어진 문자열에 가장 가까운 기존 문자열을 추측해서 알려준다. 비교해야할 두 문자열의 길이가 길고 문자열의 일부 문자가 다를 경우 비교는 어려워진다. 관련이론으로는 레벤슈타인 거리공식(Levenshtein Distance Formula) 가 있다. 이 공식은 한 문자열을 다른 문자열로 변환하는데 필요한 편집횟수를 알려준다. 편집횟수가 더 적을수록 두 문자열이 더 잘 부합하는 것이다. 또 다른 알고리즘으로 비탭(Bitap Algorithm) 이 있다. 한 문자열 안에 주어진 패턴과 거의 비슷한 부분 문자열이 있는지를 알려주는데 이를 위해 패턴의 각 문자마다 하나의 비트로 된 비트마스크(Bit Mask) 를 사용한다.

최근 연구동향으로는 지지벡터기계(Support Vector Machine) 이나 모형기반(Model Base) 접근방식이 있다.

지지벡터기계는 신경망 같은 학습분류자의 일종이나, 신경망등과는 조금 다르다 [9]. 선형분류자(Linear Classifier) 들을 일반화한 것으로, 구축과 훈련 알고리즘에서 신경망들보다 더 뛰어난 특성을 보인다. 게임 AI(Artificial Intelligence) 개발에는 규칙기반 접근방식이 흔히 쓰인다 [8]. 규칙기반 기법은 종종 지능적이지 못한 행동이 나오기도 한다. 게임세계 모형들(행동, 행동의 결과, 관찰 등) 에 기초한 모형 기반 접근방법은 좀 더 안정적일 뿐만 아니라 개발비용과 시간이 적고 수정이 쉽다.

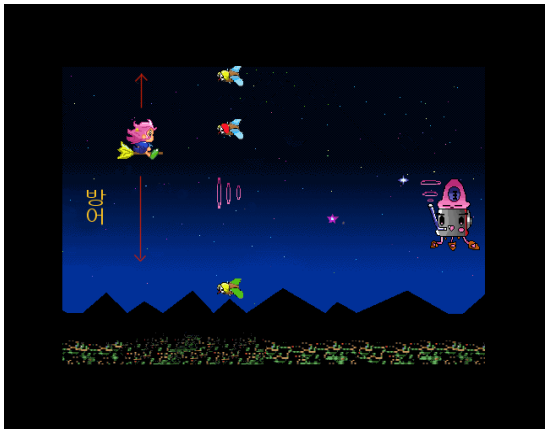
3. 지능형 게임의 구현

본 논문에서는 적군캐릭터(enemy force) 와의 대결을 통해 스스로 학습하는 아군캐릭터(our force) 를 구현한다.

3.1 캐릭터의 설계

캐릭터들은 크게 아군캐릭터와 적군캐릭터가 있다.

아군캐릭터가 하나 존재한다. 적군캐릭터의 경우 네 가지 종류의 적군캐릭터가 존재한다. 아군캐릭터는 적군캐릭터의 공격을 방어하게 된다. 방어의 의미는 실질적으로는 적군캐릭터 또는 적군캐릭터의 총알을 아군캐릭터가 위아래로 움직여 피하는 것이다.



(그림 2) 아군캐릭터와 적군캐릭터의 움직임

적군캐릭터의 공격은 일직선으로 한다. 적군캐

릭터는 스스로의 몸을 움직이거나 총알을 발사하여 아군을 공격한다. 적군캐릭터나 적군캐릭터의 총알이 아군캐릭터와 충돌하지 않도록 아군캐릭터는 적절히 방어하여야 한다.

각각의 아군캐릭터와 적군캐릭터의 위치는 메모리에 각각 상황별로 저장되며 각각의 상황에서의 학습점수가 저장된다. 아군캐릭터가 움직일 때에는 각 상황에서의 선택할 수 있는 점수 중 유리한 것을 선택하게 된다. 그러나 학습의 초창기에는 저장된 데이터가 많지 않아 좋은 선택을 못하게 된다.

적군캐릭터는 공격을 하게 되고, 아군캐릭터는 방어를 하게 된다. 적군캐릭터는 아군캐릭터 방향으로 움직여 아군캐릭터를 공략하게 되고, 아군캐릭터는 학습된 보상점수에 의해 가장 좋은 방향을 선택하여 이동한다.

(표 1) (캐릭터 간 충돌의 경우의 보상점수)

공격주체	대 상	보상점수
적군캐릭터1	아군캐릭터	-100
적군캐릭터2	아군캐릭터	-100
적군캐릭터3	아군캐릭터	-100
적군총알	아군캐릭터	-100

아군캐릭터가 적군캐릭터의 총알이나 적군캐릭터와 충돌한 경우에는 -100 의 보상이 주어진다. 처음에는 학습이 이루어지지 않았기 때문에 아군캐릭터와 적군캐릭터의 충돌은 빈번하게 이루어지게 된다.

그러나 학습이 이루어진 후에는 충돌이 일어나지 않게 된다.

```
procedure Q_Learning
{
        Find Max from QTable
        Select player character action from QTable
        generate random
        Move player character
        Move enemy character from e_action()

        if (catch)
                Generate plus reward
        if (be captured)
                Generate minus reward
        Update QTable
}
```

(그림 3) Q-learning 알고리즘

3.2 제안하는 알고리즘

강화학습에서의 상태정보는 계속되는 게임의 결과에 따라 보상 값들이 상태정보에 누적되게 되어있다. 게임이 어느 정도 진행되면 누적된 값에 의해 아군캐릭터에 가장 유리한 값을 선택할 수 있도록 되어있다.

게임의 초창기에는 상태정보를 0으로 초기화하여 모두 같은 값을 갖도록 한다. 그리고 게임이 어느 정도 진행되지 않은 초창기에는 아직 보상값이 많이 누적되지 않아 동일한 값들을 많이 가지고 있다. 이 때 보통 무작위로 그중 하나의 값을 선택하여 게임을 진행하게 된다. 이 때 적절한 위치로 이동하지 못하고 엉뚱한 방향으로 움직여 학습이 잘 이루어지지 않는다.

본 논문에서는 이러한 단점을 없애고자, 동일한 값들이 추출될 때, 플레이어 행동예측 방법을 제안하여 유리한 방향으로 움직이도록 Q-learning 알고리즘을 개선하였다.

단계 1 : 아군캐릭터의 움직일 수 있는 방향을 가상의 구역으로 나누고, 각각에 고유한 번호를 매긴다. 적의 공격이 일어나면 그 구역의 번호를 문자열에 추가하는 것이다. 공격이 계속 이어지면, 열에는 반복적인 패턴이 나타나게 될 것이다.

단계 2 : 세 번째 부터 발생하는 문자는 기존 발생된 문자들과의 비교를 통하여, 부합크기를 계산할 수 있다. 1 2 1 로 나타날 경우에 1 은 기존 발생하였으므로 부합크기는 1 이 된다.

단계 3 : 기존 발생된 문자들이 1 2 1 2 로 발생되었다면 2 는 이미 두 번이나 같은 값이 반복되므로 부합크기는 2가 된다. 그 다음에 나올 값을 예측한다면 1 이 될 것이다. 부합크기는 클수록 신뢰성이 크게 된다.

단계 4 : 마지막 문자를 기존 발생된 문자열의 처음부터 검색하고, 일치하면 마지막 문자에서 하나를 더 취하고 또 검색한다. 부합되는 문자열 중에서 가장 긴 문자열을 취하고, 그 부분 문자열의 다음 원소를 전체열의 다음 원소로 결정한다. 1 2 1 2 2 1 인 경우에 맨 처음의 1 은 부합크기가 1 인 반면에, 두 번째와 세 번째인 2 1 은 부합크기가 2 이다. 이런 경우 부합크기가 2 인 2 1 다음의 2 가 예측 원소로 사용되어야 한다.

열 : 1 2 1 2 2 1
부합크기 : 1 2

단계 5 : 순차예측 알고리즘에서 긴 부분문자열만 채택한 점을 개선하였다. 긴 부분문자열과 짧지만 자주 반복되는 문자열의 우선순위를 결정한다.

(식3) 은 예측 값을 산출하기 위한 수식이다. l 이란 문자열을 의미한다. d는 문자열 끝까지의 거리이다. r이란 반복횟수이다. 예측 값은 부분문자열의 길이를 문자열 끝까지의 거리로 나눈 값이다. 긴 부분문자열과 짧지만 자주 반복되는 부분문자열이 동시에 발생하는 경우에는, 짧은 부분문자열의 평균을 구해 긴 부분문자열과 비교한다. 예측 값이 큰 원소를 선택한다.

$$Estimate = \frac{\sum_{i=1}^{N}(l/d)}{r} \quad \text{(식3)}$$

단계 6 : 결정된 최종 예측 값의 다음 값을 선택한다. 동일한 부분문자열의 경우에는 통상적으로 볼 때, 가장 최근의 부분문자열의 다음 값을 예측 값으로 사용한다.

강화학습의 초창기에는 보상 값이 많이 누적되지 않아 동일한 0의 값들을 갖고 있다. 그러므로 아군캐릭터가 가야할 방향을 제안된 플레이어 행동예측 방법으로 결정하여 진행한다.

이 방법을 사용함으로서 생기는 알고리즘 사용시간은 15000 개의 학습 횟수를 기준으로 하였을 때, 기존 Q-learning 과의 알고리즘 실행시의 시간차는 $\frac{1}{10}$초 정도로 미미함 으로, 큰 오버헤드는 없다고 본다.

4. 실험 및 결과

하나의 상태공간은 8 × 5 이므로 40개의 격자셀로 구성된다. 아군캐릭터의 관점에서 볼 때 적군캐릭터가 넷일 경우에 고려하여야 할 요소들은 다음과 같다.

㉮ 아군캐릭터의 위치
㉯ 적군 보스캐릭터 총알의 위치
㉰ 적군캐릭터 셋의 위치
㉱ 아군캐릭터가 방어를 위해 이동 가능한 공간

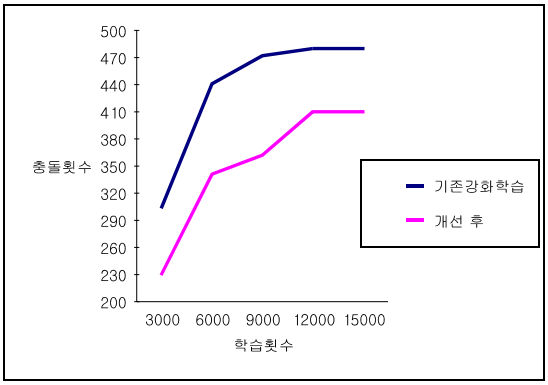
보스캐릭터의 위치는 고정되어 있으므로 상태공간을 고려할 필요는 없다. 아군캐릭터의 상태공간은 40 이므로 나머지 고려하여야 할 요소들의 경우 네 번째 경우를 제외하고는 모두 40 이 된다. 네 번째 경우는 상 또는 하로 이동하므로 2 가 된다. 그러므로 상태공간의 크기는 40 × 40 × 40

× 40 × 40 × 2 로 204,800,000 이 된다.

강화학습이 반영된 캐릭터와 개선된 강화학습으로 움직이는 캐릭터로 실험하였다. 적군캐릭터인 보스캐릭터의 경우 일정한 위치에서 총알을 발사하게 된다.

(표 2) 기존 강화학습과 개선된 강화학습에서의 비교결과

충돌 / 학습 횟수	강화학습 후					제안한 강화학습 후				
	적1	적2	적3	적4	계	적1	적2	적3	적4	계
3,000	76	67	85	77	305	55	54	65	57	231
6,000	113	94	119	115	441	93	74	89	85	341
9,000	124	104	124	120	472	94	84	94	90	362
12,000	128	106	125	121	480	108	96	105	101	410
15,000	128	106	125	121	480	108	96	105	101	410



(그림 4) 기존 강화학습과 개선된 강화학습에서의 비교결과

(표 2) 와 (그림 4) 에서 보면 알 수 있듯이 강화학습을 하는 캐릭터와 개선된 캐릭터 실험결과로 차이를 보임을 알 수 있다.

충돌횟수를 비교해 보면 약 20% 에서 30% 까지 개선된 강화학습에서 충돌횟수를 줄이는 것을 알 수 있다.

(그림 4)에서 기존의 강화학습과 개선된 후의 강화학습의 실험결과를 비교해 보았다. 기존방법의 실험결과가 높은 충돌횟수를 나타내는 반면에, 개선후의 실험결과는 좀 더 적은 충돌횟수를 나타내는 것을 볼 수 있다.

5. 결 론

게임시나리오가 게임의 흥미를 이끌어내지만, 캐릭터의 자동화문제는 여전히 남은 숙제라고 할 수 있다. 캐릭터의 자동화로 인해 게임의 재미가 더할 수 있고, 그것은 강화학습을 비롯한 여러 가지 학습알고리즘을 효과적으로 이용할 때, 가치를 더할 수 있다.

본 논문에서는 강화학습 알고리즘을 이용하여 아군캐릭터와 적군캐릭터가 전투하는 상황에서 적군캐릭터가 아군캐릭터를 공격할 때에 음의 보상 값을 부여하여 게임캐릭터가 학습하게 하여 지능적으로 움직이게 하였다.

일반적인 강화학습에서는 일정기간 보상 값이 쌓이기 전에는 정확한 값을 산출하기가 어렵다. 그러므로 일정기간 동안은 동일한 값이 산출되었을 때 무작위 난수 값을 발생시켜 평균적으로 균등하게 순서를 부여한다. 그러나 본 논문에서는 이러한 단점을 개선하기 위하여 동일 값이 산출될 때, 플레이어 행동예측방법을 제안하여 학습의 속도향상을 시도하였다. 그 결과 학습속도가 향상됨을 알 수 있었다.

구현된 아군캐릭터가 지능적으로 잘 움직이는지 확인하기위해, 게임을 제작하여 기존의 강화학습으로 움직이는 캐릭터와 개선된 강화학습을 적용한 캐릭터를 비교하였다.

실험결과 개선된 강화학습이 더 많은 학습속도 향상을 보임을 알 수 있었다.

향후 연구방향으로는 최근접 문자열 알고리즘을 개선하거나 지지벡터기계를 개선한 플레이어 행동 예측 알고리즘을 제작 하고자한다

참 고 문 헌

[1] 한국게임산업진흥원, “2008 대한민국 게임백서”, 2009

[2] 신용우, “강화학습을 이용한 지능형 게임캐릭터의 제어”, 인터넷정보학회논문지, 제8권 5

호, pp 91-97, 2007

[3] Laramee, Francois Dominic, "A Rule-based Architecture Using Dempster-Shafer Theory", AI Game Progrmming Wisdom, Charles River Media, 2002.

[4] Mommersteeg, Fri, "Pattern Recognition with Sequential Prediction", AI Game Progrmming Wisdom, Charles River Media, 2002.

[5] Laramee, Francois Dominic, "Using N-Gram Statistical Models to Predict Player Behavior", AI Game Progrmming Wisdom, Charles River Media, 2002.

[6] Imran Ghory, "Reinforcement learning in board games.", available at http://www.cs.bris.ac.uk/Publications/Papers/2000100.pdf, 2004.

[7] Nee Jan van Eck, Michiel van Wezel., "Reinforcement Learning and its Application to Othello", available at http://www.few.eur.nl/few/people/mvanwezel/rl.othello.ejor.pdf, 2004

[8] Armand Prieditis, "Applying Model-Based Decision-Making Methods to Games: Applying the Locust AI Engine to Quake III", Game Programming Gems 6, Charles River Media, 2006.

[9] Julien Hamaide, "Short-Term Memory Modeling Using a Support Vector Machine", Game Programming Gems 6, Charles River Media, 2006.

[10] James Boer (ArenaNet), "Closest-String Matching Algorithm", Game Programming Gems 6, Charles River Media, 2006.

[11] Steve Woodcock, "Game AI : The State of the Industry", Game Developer Magazine, 2000.

[12] Steve Rabin, AI Game Programming Wisdom 2, Charles River Media, 2003

[13] Steve Rabin, AI Game Programming Wisdom, Charles River Media, 2002

◐ 저 자 소 개 ◑

신 용 우(Yong-Woo Shin)

2004년 경희대학교 컴퓨터공학과 지능시스템전공 (박사수료)
1990년~2000년 프리랜서게임프로그래머, LG 데이콤
2000년~현재 동아방송예술대학 디지털영상과 교수
2010년~현재 DIMA웹스 대표
관심분야 : 게임제작, 인공지능, 디지털영상
E-mail : ywshin@dima.ac.kr

정 태 충 (Tae-Choong Chung)

1987년 한국과학기술원 전자공학전공 (공학박사)
1987년~1988년 KIST 시스템공학센터 선임연구원
1988년~현재 경희대학교 컴퓨터공학과 교수
관심분야 : 기계학습, 보안, 최적화, 에이전트
E-mail : tcchung@khu.ac.kr