한국컴퓨터정보학회 동계학술대회 논문집 제20권 제1호 (2012. 1)

게임 맵에서 경로 찾기 해법을 위한 시뮬레이티드 어닐링 알고리즘

강명주°
°청강문화산업대학교 컴퓨터게임전공
e-mail:mjkkang@ck.ac.kr

A Simulated Annealing Algorithm for Path Finding in A Game Map

Myung-Ju Kang°
°Dept of Computer Game, ChungKang College of Cultural Industries
e-mail:mjkkang@ck.ac.kr

● 요 약 ●

게임 인공지능에서 경로 찾기는 매우 중요한 분야이다. 캐릭터나 NPC(Non Player Charater)가 목적지까지의 경로를 탐색하기 위해서는 복잡하게 구성된 게임 맵에서 오브젝트나 벽 등의 장애물을 회피하면서 가능한 최단 경로를 찾아야 한다. 기존의 일반적인 경로 찾기 알고리즘은 캐릭터의 움직임이 자유로운 간단한 게임 맵에서는 전역 최소해 탐색이 가능하다. 그러나 다양한 오브젝트 등이 배치된 큰 규모의 게임 맵에서는 캐릭터가 목적지까지의 경로를 탐색하는 과정에서 지역 최소해(Local Minima)로의 수렴이 발생함으로써 이를 탈피하는 것이 어렵고, 결국에는 목적지까지 도달하지 못하는 상황이 발생하게 된다. 본 논문에서는 이러한 기존 경로 찾기 알고리즘의 단점을 해결하기 위해 시뮬레이티드 어닐링 알고리즘을 제안하였다.

키워드: 경로 찾기(Path Finding), 시뮬레이티드어닐링(Simulated Annealing)

I. 서론

게임 인공지능에서 캐릭터의 경로 찾기는 매우 중요한 부분 중의 하나이다. 다양한 오브젝트와 벽 등의 장애물로 구성된 복잡한 게임 맵에서 캐릭터나 NPC(Non Player Character)가 목적지까지 길을 찾아 이동하기 위해서는 장애물을 회피하면서 가능한 최단 경로로 이동해야 한다.

지금까지 경로 찾기 해법을 위한 알고리즘들에 대한 많은 연구가 진행되어 왔다. 가장 기본적인 방법은 현재 위치에서 임의의 방향으로 이동시키는 것이다. 이 방법은 주인공 캐릭터가 목적지까지 갈 수 있다는 것을 보장해 주지 못하며, 목적지까지 도달하더라도 최단 경로로 이동하는 경우는 매우 드물다[1]. 또 하나의 방법은 스택을 이용한 깊이우선탐색 방법을 사용한다. 깊이우선 탐색은 맵 트리에서 깊이를 먼저 탐색하는 방법이다. 최악의 경우 맵 전체를 탐색해야하는 경우가 발행할 수 있다. 또한 캐릭터가 목적지까지 도달할 수는 있지만 최단 경로를 찾는 것이 힘들다[1].

지금까지 많이 사용되는 경로 찾기 방법은 A* 알고리즘이다. A* 알고리즘은 주인공 캐릭터가 시작 위치에서 현재 도착한 위치까지의 거리와 앞으로 이동해야 할 목적지까지 거리의 추정치를 더하여 최소값을 갖는 경로로 이동시키는 방법이다[2]. 현재 위치에서 앞으로 남은 목적지까지 남은 거리의 계산은 일반적으로 유클리디안 거리로 계산된다. 따라서, 현재 위치까지의 정보는 계속 갱신이 되며 앞으로 남은 목적지까지의 거리 추정치를 계속 계산해야 한다. 또한, 주인공 캐릭터가 벽이나 장애물을 만났을 때 이를 회피하여 다시 계산해야하는 단점이 있다.

최근에는 유전알고리즘과 같은 휴리스틱 알고리즘을 이용한 경로 찾기 방법이 연구되고 있다[3]. 본 연구에서는 유전 알고리즘과 비슷한 휴리스틱 방법으로 기존 방법의 단점을 해결하기 위해 시뮬레이티드 어닐링 알고리즘(Simulated Annealing Algorithm, SA)을 제안한다.

II. 시뮬레이티드 어닐링 알고리즘

2.1 시뮬레이티드 어닐링 알고리즘

SA 알고리즘은 금속의 담금질 특성을 이용하여 조합 최적화 문제를 해결하는 방법이다. 즉, 높은 온도에서 시작하여 온도를 서서히 낮추며 안정된 상태로 이동하면서 최적해를 구하는 방법이다.

SA 알고리즘은 현재 상태의 해와 이웃하는 해를 이용한다는 면에서 국부탐색 알고리즘과 비슷하다. 그러나 국부탐색 알고리즘은 이웃하는 해가 현재 상태의 해보다 좋을 경우에만 해 공간의 탐색이 계속 이루어지는 반면, SA 알고리즘에서는 비록 현재 상태의 해가 이웃하는 해보다 좋지 않더라도 Metropolis 알고리즘에 의한 확률적인 방법에 의해 이웃 해를 현재 상태의 해로 설정함으로써 국부탐색 알고리즘에서는 배제될 수 있는 해 공간을 탐색 할

수 있도록 한 것이다. 따라서, SA 알고리즘은 모든 해 공간의 탐색이 가능하며, 이로 인하여 전역 최소해를 구할 수 있다.

```
(1) begin
(2)   select an initial solution i ∈ S;
(3)   select an initial temperature T;
(4)   while stopping criterion not true
(5)     count = 0;
(6)     equilibrate( );
(7)     choose a neighbor j ∈ N(i);
(8)       δ = f(j) - f(i);
(9)     if δ < 0 then
(10)        i = j;
(11)    else if random[0, 1) < exp(- δ/T) then
(12)        i = j;
(13)    end if
(14)    count = count + 1;
(15)    reduce temperature T;
(16)  end while
(17) end
```

그림 1. 시뮬레이티드 어닐링 알고리즘
Fig 1. Simulated Annealing Algorithm

[그림 1]은 SA 알고리즘을 나타내고 있다. SA 알고리즘에서 단계 (11)과 단계 (12)는 새로운 상태의 해가 현재 상태의 해보다 나쁠 경우에 Metropolis 조건에 의해 새로운 상태를 결정하는 단계이다. 이는 최종적인 전역 최소해로 수렴해 가는 과정에서 현재 상태의 해가 지역 최소해(Local Minima)에 빠질 수 있는 경우를 탈피하기 위한 것이다. 단계 (15)는 온도를 서서히 낮추는 단계로서, 냉각 스케줄에 의해 온도 변화가 이루어진다.

SA 알고리즘에서는 냉각 스케줄을 어떻게 설정하느냐에 따라 SA 알고리즘의 성능에 큰 영향을 준다.

2.2 냉각 스케줄

냉각 스케줄은 SA 알고리즘에서의 해 공간 탐색에 있어서 유한회의 탐색을 가능하게 하기 위한 것이다. 일반적으로 초기 온도를 높게 설정하여 서서히 냉각 스케줄에 따라 감소시킴으로써, SA의 초기 상태에서는 이웃해로의 전이가 가능한 많이 이루어지도록 하여 지역 최소해를 탈피하도록 하며, 상태가 진행됨에 따라 낮은 온도를 유지함으로써 안정된 상태로 진행된다. 그러나 적용되는 문제에 따라 어떤 냉각 스케줄을 사용하느냐가 SA 알고리즘의 성능을 결정하게 되며, 적합한 냉각 스케줄의 선택은 실험적으로 얻을 수 있다[4].

일반적으로 사용되는 냉각스케줄은 (식 1)과 (식 2)를 많이 사용한다[4]. 여기서 T_0는 초기 온도를 그리고 $T(k)$는 k번째 상태의 온도를 나타낸다.

$$T(k) = 0.9 \times T(k-1) \quad \cdots\cdots\cdots\cdots \text{(식 1)}$$

$$T(k) = \frac{T_0}{\log(1+k)} \quad \cdots\cdots\cdots\cdots \text{(식 2)}$$

2.3 벡터 구성

게임에서 경로 찾기를 구현하기 위해서는 지능 캐릭터가 현재의 위치에서 목표 위치로 가기 위한 경로를 탐색해야 한다. SA 알고리즘을 이용하여 경로를 탐색하기 위해서는 먼저 알고리즘에 적용할 벡터를 구성해야 한다.

본 연구에서는 게임의 캐릭터나 NPC가 현재 위치에서 다음 위치로 이동할 수 있는 방향 정보를 벡터로 구성하는 방법을 사용하였다. 현재 위치에서 갈 수 있는 방향은 8개의 방향이므로 2진 코드 3자리를 하나의 방향으로 설정할 수 있다. 다음은 캐릭터의 움직임 방향을 3자리 비트로 나타낸 것이다.

비트코드	디코딩	방향
000	0	동
001	1	서
010	2	남
011	3	북
100	4	남동
101	5	남서
110	6	북동
111	7	북서

벡터는 다음과 같이 비트코드의 스트링으로 표현할 수 있다.

111101101101011100

이 벡터스트링은 3비트씩 나누어 분리하고 분리된 각 벡터는 디코딩을 통해 방향을 해설할 수 있다. 다음은 위의 벡터를 디코딩한 예이다.

벡터스트링	111101101101011100					
유전자	111	101	101	101	011	100
디코딩	7	5	5	5	3	4
방향	북서	남서	남서	남서	북	남동

[그림 2]는 위의 디코딩 정보를 게임 맵에 적용하였을 때 경로 이동 순서를 나타내고 있다. 0의 위치가 시작 위치이고 디코딩 정보의 순서에 따라 이동하였을 때 마지막 위치는 8의 위치가 된다. 이 예에서는 벽이나 오브젝트 등의 장애물을 표현하지 않은 상태이다. 벽이나 오브젝트 등의 장애물을 만나는 경우에는 충돌처리를 하여 이동이 되지 않도록 하여야 한다.

				1	
			2		0
	5	3			
	4	6			

그림 2. 디코딩 예
Fig 2. The Example of Decoding

2.4 이웃해 생성 방법

본 논문에서는 이웃해 생성 방법으로 Flip 방법을 사용한다. [그림 3]은 Flip 방법을 이용한 이웃해 생성 방법을 나타내고 있다. 즉, 현재 상태의 벡터 요소 중 여섯 번째 요소를 선택하여 반전시킴으로서 벡터를 생성하는 방법이다.

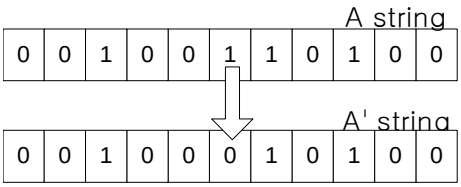

그림 3. Flip 방법을 이용한 새로운 상태의 벡터 생성
Fig 3. Flip method

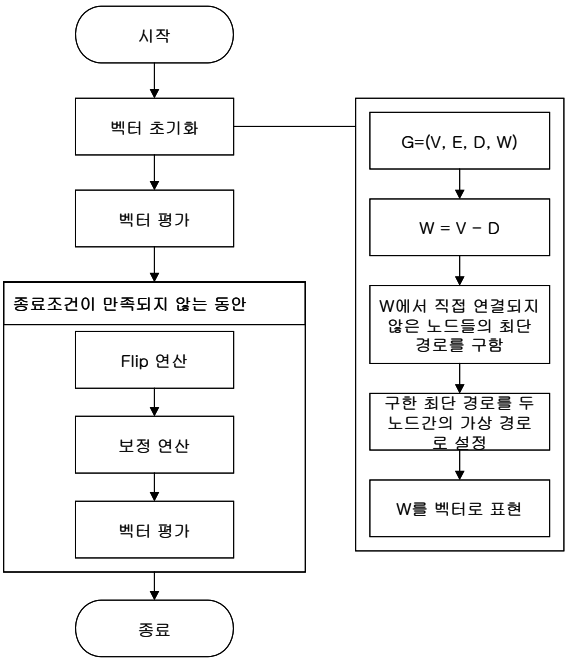

그림 4. 제안된 SA 알고리즘
Fig 4. Proposed SA Algorithm

Flip 방법을 이용하여 새로운 벡터를 생성하게 되면, 새로운 벡터는 각 문제의 특징에 따라 가능하지 않은 경우가 생길 수 있다. 이 때 이를 보정할 수 있는 연산이 필요하다. 게임 맵에서는 오브젝트가 포함되어 있기 때문에 경유할 수 없는 노드가 존재하며, 이를 해결하기 위한 보정연산이 필요하다.

[그림 4]는 위에서 설명한 게임 맵에서 경초 찾기 해법을 위한 SA 알고리즘을 나타낸다.

III. 실험결과

3.1 실험 환경

실험은 인텔 Core i5 2.53Ghz, 2G 메모리 환경에서 실시하였다. 시뮬레이션 개발은 Visual Studio 2008 C++을 이용하였다.

알고리즘에서 고려해야 할 파라미터로는 벡터의 크기, 초기 온도, 종료조건 등이 있다.

본 논문에서는 벡터의 크기는 12로 하였으며, Temperature의 초기 값은 60에서 시작하여 0.05씩 감소시켜 1이 될 때까지 서서히 낮춘다. 종료조건은 목적지까지의 경로 탐색이 이루어지면 종료한다.

3.2 실험 결과

게임 맵은 가로 20 세로 20의 타일맵으로 구성되어 있고, 출발지점에서 시작하여 목적지까지 이동하는 경로를 탐색한다. [그림 5]는 본 논문에서 사용된 게임 맵의 실제 시뮬레이션 화면을 나타내고 있다.

[그림 6]은 본 논문에서 제안한 SA 알고리즘을 이용하여 경로 찾기를 성공한 결과를 보여주고 있다.

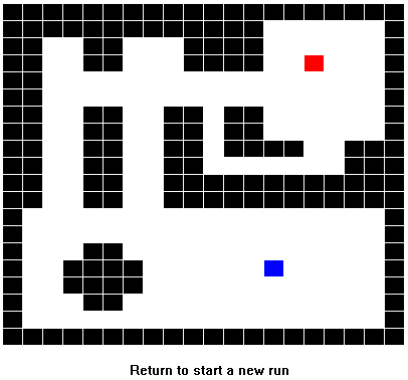

그림 5. 게임 맵
Fig 5. Game Map

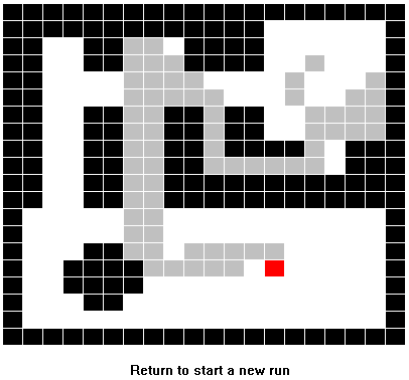

그림 6. 실험 결과
Fig 6. The result of Simulation

Ⅳ. 결론

본 논문에서는 게임에서 캐릭터나 NPC가 목적지까지의 경로를 탐색하는 SA 알고리즘을 제안하였다. 기존 알고리즘은 캐릭터의 움직임이 자유로운 간단한 게임 맵에서는 전역 최소해를 찾는 것이 가능하지만, 벽이나 오브젝트가 다양하게 배치된 복잡한 게임 맵에서는 목적지까지의 경로를 탐색하는 전역 최소해를 찾기가 힘들다는 단점이 있다.

따라서, 본 논문에서는 이러한 기존 알고리즘의 단점을 해결하기 위해 시뮬레이티드 어닐링 방법을 방법을 제안하였다.

실험을 통해 제안한 SA알고리즘이은 최단 경로 탐색에 성공함으로써 전역 최소해를 구할 수 있었다.

참고문헌

[1] Ron Penton, Data Structures for Game Programmers, Thomson, 2004

[2] Mat Buckland, AI Techniques for Game Programming, Thomson, 2002

[3] M. J. Kang, A Path Finding Using A Genetic Algorithm, Proceeding of KSCI, Vol. 14, No. 2 2009

[4] Lin, Chin-Teng and Lee, C. S. George, Neural Fuzzy Systems, Prentice Hall, 1996