A* 경로계획 적용을 위한 플랫폼 게임 인공지능의 후처리 알고리즘 제안

김현태 ○ 김경중
세종대학교 컴퓨터공학과
kimhyun0416@sju.ac.kr, kimkj@sejong.ac.kr

Post Processing Algorithm Proposal of Platform Game Artificial Intelligence for A* Path Planning Application

HyunTae Kim○ KyungJoong Kim
Dept. of Computer Engineering, Sejong Univ.

요 약

게임 내에서 캐릭터가 목표하는 지점까지 장애물을 피해서 최단 경로로 움직이는 것은 매우 중요하다. 최단 경로로 길을 찾는 문제에서 A* 알고리즘은 상당히 범용적인 해결방안이다. 하지만 A* 알고리즘을 실제 게임에 적용하기에는 몇 가지 고려 해야 할 사항이 존재한다. 특히 플랫폼 게임과 같은 상하좌우로 자유자재로 움직이지 못하는 게임에서는 더욱 그 문제점이 부각된다. 본 논문에서는, 이러한 문제점을 해결하기 위해 A*로 얻어진 경로를 재처리를 하여 중간 목표 지점을 알려주는 알고리즘을 개발하였다. A* 알고리즘에 본 알고리즘을 적용하여 실제 플랫폼 게임에서 A* 알고리즘만을 사용하는 인공지능보다 빠른 움직임을 보이는 것을 실험을 통해 입증하였다.

1. 서 론

게임에서 캐릭터가 지정된 위치까지 최단경로로 이동하는 것은 매우 핵심적인 문제이다. 이를 위하여 가장 적은 비용으로 최단경로로 이동하는 다양한 알고리즘들이 존재한다. 경로탐색 방법들은 깊이 우선 탐색, 너비 우선 탐색, dijkstra 알고리즘, A* 알고리즘이 있으며 이를 변형한 Region-Based A* 와 IDA*(Interactive-Deepening A*) 알고리즘이 대표적인 예이다[1,2].

그 중 A* 알고리즘은 범용적인 길찾기 문제의 해결을 위해 가장 많이 사용하는 접근방식이다. 이 알고리즘은 장애물을 피하여 막다른 골목에 빠지지 않고 주어진 출발지와 최종 목적지까지 가는 최단 경로를 찾는다[3]. 하지만 실제 게임에 A* 알고리즘을 그대로 적용하기에는 어려움이 따른다. 특히 상하좌우를 이동이 가능한 게임이 아닌 플랫폼 게임에서는 그 특수성 때문에 A*를 적용하기 더 어렵다.

플랫폼 게임(플랫포머)이란 발판(플랫폼)이 존재하는 게임을 말한다. 이 발판은 언덕이 될 수도 있고 계단이나 장애물이 되기도 한다. 캐릭터가 발판간 이동을 하며 적을 무찌르거나 장애물을 피하는 게임이 주를 이룬다. 대표적인 예로 슈퍼마리오, 동킹 콩, 소닉등이 있다[4]. 본 논문에서는 그 플랫폼 게임 중 Geometry Friends 라는 게임을 이용한다. Geometry Friends란, IEEE 2013 Conference on Computational Intelligence in Games(CIG2013) 에 새로운 게임 인공지능 경진대회 중 하나이다. 두 캐릭터가 발판을 이동하면서 주어진 지도 안에 다이아몬드를 획득하는 게임이다. 다양한 난이도가 존재하며 싱글 플레이, 협동 플레이가 가능하다.

실제 Geometry Friends 게임에 A* 알고리즘을 적용시키면 인공지능의 움직임이 부자연스러운 경우가 많이 발생한다. [5]에서는 A* 알고리즘의 성능을 향상시키는 방법을 연구한 반면, 본 연구에서는 Geometry Friends 게임에 A* 경로계획을 이용하여 알아낸 최적의 경로를 따라 자연스럽게 움직이는 것이 가능하도록 하는 알고리즘을 개발하였다. 본 논문의 2장은 기본적인 A*알고리즘을 설명하고 실제 게임에 적용시켜본다. 3장에서는 인공지능에 자연스러운 움직임을 가능하게 하는 알고리즘을 제안하고 4장에서는 평범한 A*와 제안하는 알고리즘을 적용한 A*를 비교실험을 해본다. 5장에서는 본 연구를 통한 결론과 향후 연구에 대해 서술한다.

2. A* 알고리즘

2.1 기본 개념

A* 알고리즘은 열린 목록(Open list)과 닫힌 목록(Closed list)를 사용한다. 열린 목록에는 아직 조사하지 않은 상태의 노드들을 담고 닫힌 목록에는 이미 조사한 상태의 노드들을 담는다. A* 알고리즘이 기존 탐색 알고리즘과 다른 점은 휴리스틱 함수를 사용하여 평가를 한다는 것이다. 그래서 출발 노드부터 목표 노드까지 인접한 노드들을 조사해가면서 가장 싼 비용의 경로를 찾는다. 가장 싼 비용을 측정하기 위하여, $f(x) = g(x) + h(x)$ 식을 사용한다. 총 비용 $f(x)$ 는

출발 노드부터 현재 노드 x까지의 비용 $g(x)$와 현재 노드 x부터 목표 노드까지 휴리스틱 비용 $h(x)$을 더한 값이다. 최종적으로 목표 노드의 인접한 노드중에서 f(x)값이 가장 낮은 노드의 부모 노드를 차례대로 되짚어 나온 경로가 최소 비용의 경로가 된다[2,3,5].

2.2 Geometry Friends

Geometry Friends 게임은 2D 환경의 플랫폼 게임으로서 초록 사각형과 노란 공의 두 캐릭터를 이용하여 제한된 시간 내에 장애물들을 피하여 다이아몬드를 얻는 게임이다. 초록 사각형은 좌우 이동과 크기 조정을 할 수 있고 노란 공은 좌우이동과 크기조정, 뛰기가 가능하다[6]. 자세한 내용은 [5]에 설명되어 있다. 실험의 복잡성을 줄이기 위해 본 실험에는 초록 사각형만 가지고 플레이를 하였다.

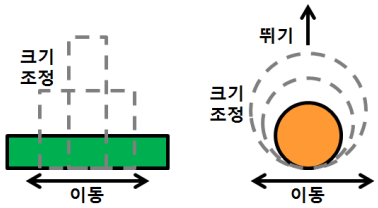

그림 1. Geometry Friends 캐릭터들의 움직임

2.2 실제 게임에 A* 알고리즘 적용

A* 를 적용하기 위한 지도의 공간 분할의 종류는 다양하지만[2] 본 논문에서는 가장 간단한 정사각형 모양으로 지도를 분할하여 A*를 적용하였다. Geometry Friends 의 지도의 전체 크기는 1024x720(pixel)이다. 한 노드의 크기는 40x40(pixel)로 전체 지도를 총 558개의 노드로 나눈다. 그리고 캐릭터(그림 2의 초록색 사각형)의 정중앙이 위치한 노드를 출발 노드로 다이아몬드의 정중앙이 위치한 노드를 목표 노드로 한다. 장애물(그림 2의 검은색 사각형들)들을 모두 인식시키면 A* 알고리즘을 통하여 그림 2와 같이 출발 노드에서 목표 노드까지의 최소 비용의 경로를 얻을 수 있다.

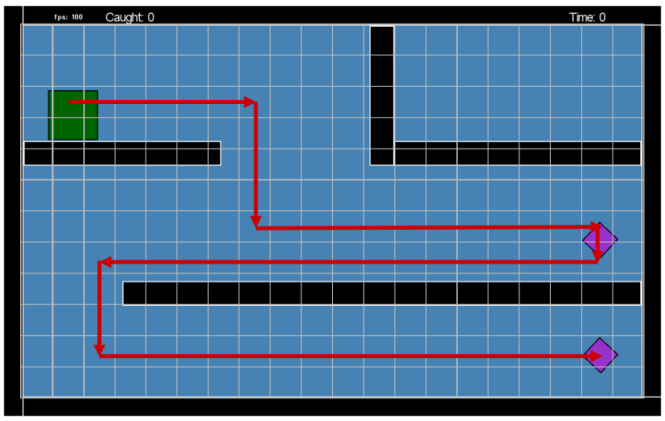

그림 2. A*로 얻은 최단 거리의 경로

3. A* 경로 계획 처리 알고리즘

3.1 기존 A* 알고리즘 사용의 단점

그림 2와 같이, A*로 목표 노드(이 게임에서는 다이아몬드)까지의 최단 경로는 구할 수 있지만, 실제 게임 내에서 에이전트가 움직일 때 그 경로대로 움직이면 몇 가지 단점이 존재한다. 에이전트가 A*로 얻은 경로를 따라 순차적으로 움직이면, 다음 노드까지 움직이고 정지 후 그 다음 노드까지 움직인다. 결론적으로, 에이전트가 멈칫거리며 움직이기 때문에 움직임의 속도도 매우 느려진다.

또한 에이전트의 크기가 40x40인 한 노드의 크기보다 크기 때문에 바로 다음 노드에 에이전트 일부분이 걸쳐있는 경우가 존재하게 된다. 이러한 경우가 발생하면, 에이전트의 움직임은 매우 부자연스러워지고 장애물 사이에 끼이는 문제가 발생한다.

3.2 제안하는 방법

A*를 통해 얻어진 경로는 순차적으로 리스트 안에 저장된다. 리스트의 맨 처음부터 차례대로 노드를 따라가면 목표 노드까지 도달한다. 제안하는 알고리즘은 A*로 얻어진 경로를 재처리를 하여 게임 상에 에이전트 A 가 단번에 움직일 수 있는 중간 목표 노드인 m을 출력해준다. 경로 P 에 현재 노드를 n_0, 그 다음 노드를 n_1 라 할 때, 제안된 알고리즘은 알고리즘 1과 같다.

```
# Case_A
If ( A is n0 )
  While (P.count != 0 ) do
    If ( A.y == n1.y)
      m ← n1;
    Remove P(−1);
  End While
# Case_B
If ( A's around is n0 )
  While ( P.count != 0 ) do
    If ( A's around is n0 && n0.x == n1.x)
      m ← n1;
    Remove P(−1);
  End While
If ( !Case_A && !Case_B ) m ← n0;
Return m
```

알고리즘 1. A* 경로 처리를 위한 알고리즘

4. 실험

4.1 실험 방법

제안된 방법의 성능을 보이기 위하여, 본 논문에서는 동일한 에이전트 안에 A* 알고리즘을 이용하고 제안된 알고리즘을 함께 사용하였을 때와 사용하지 않을 때를 나누어 성능분석을 해본다. 실험은 두 개로 나누어 진행하였다. 먼저 동일한 지도 안에 움직임을 파악하기

위하여, 그림 2에서 일정한 시간마다 에이전트의 위치를 로그로 기록하여 움직인 궤적을 알아본다.

두 번째로, 실제 CIG 2013 경진대회에서 사용한 레벨 1부터 5까지의 지도를 사용하여 두 경우에 대해 성능을 분석하였다. 공정성을 위하여 각 레벨별로 10번씩 수행 후 평균을 내서 최종 점수를 구한다.

4.2 실험 결과

그림 2의 지도에서 한 실험의 결과, 그림 3과 그림 4에서 볼 수 있듯이 그림 4가 그림 3보다 로그로 찍힌 점 사이의 간격이 멀다. A*만 사용하였을 경우, 총 39번 로그가 남겨졌으며, A*와 제안된 알고리즘을 함께 사용하였을 경우, 총 23번 로그가 남겨지는 것을 확인하였다. 결론적으로 A*만 사용하였을 경우가 제안된 알고리즘과 A*를 함께 사용하였을 때보다 느린 움직임을 보이는 것을 확인할 수 있었다.

또한 레벨 1~5까지의 성능 평가에서도, 표 1과 같이 레벨1,2,5에서는 각각 제안된 알고리즘의 Case_B 부분이 빠짐으로 성능의 차이가 많이 났다. 반면, 레벨 4의 경우는 직선코스가 적고 장애물이 많아서 거의 비슷한 점수가 나왔다. 각 레벨별 t-test 검증 결과, 모든 레벨에서 p-value가 0.05보다 낮게 나왔다. 결론적으로, 제안된 알고리즘을 같이 사용하였을 때가 A*만 사용할 때보다 성능향상을 보임을 확인할 수 있다.

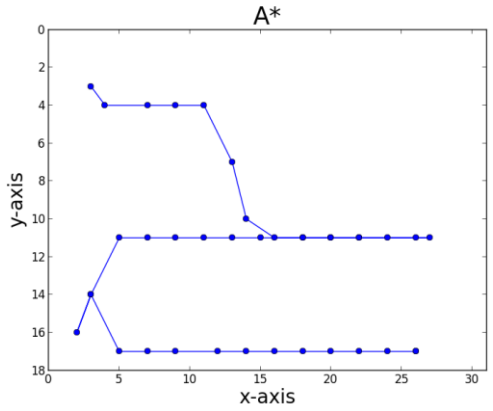

그림 3. A* 알고리즘으로 움직인 궤적

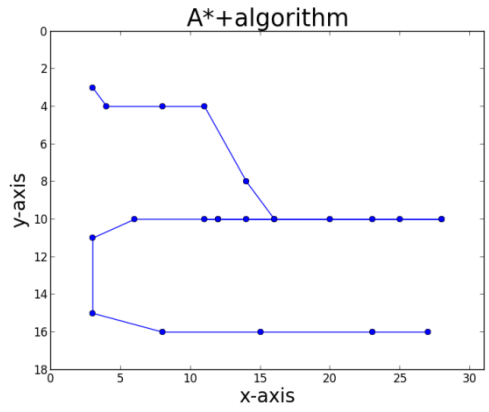

그림 4. A*와 제안하는 알고리즘으로 움직인 궤적

표 1. A* 와 제안하는 알고리즘 + A* 성능비교

레벨	A* (평균±표준편차)	A* + 알고리즘 (평균±표준편차)	t-test (p-value<0.05)
1	0±0	662±42.85	5.07E-12
2	148.5±70.5	407±22.83	2.19483E-06
3	1735.5±41.38	2002.5±32.35	1.04507E-08
4	141±3	155±2.27	2.05083E-07
5	750±273.86	1673.611±605.1	0.001422505
총합	2775±321.31	4900.111±595.33	

5. 결론 및 향후 연구

본 논문에서는, A* 알고리즘을 실제 게임에 적용시켜보고 그 문제점을 파악하고 해결하기 위한 알고리즘을 개발하였다. 특히 직선 구간에서 A*만 사용하였을 때보다 현저하게 성능이 향상되는 것을 확인할 수 있었다. 그리고 게임의 특성상 A*가 찾은 경로만 따라갈 경우, 실제로 장애물에 걸리는 경우나 바로 다음 노드로 이동하지 못하는 문제도 해결할 수 있었다. 추후에는 지도의 공간 분할 문제를 타일형 뿐만 아니라 다른 형태로 지도를 분할하고 그에 맞게 A* 알고리즘을 수행하고자 한다. 또한 [5]에서 제안한 방법을 본 연구에서 수행한 부분과 연계시켜 더욱 성능이 뛰어난 인공지능을 만들 예정이다.

6. 감사의 글

이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국 연구재단의 지원을 받아 수행된 중견연구자지원사업(2013-016589) 및 뇌과학 원천기술개발사업임(2010-0018950)

REFERENCES

[1] Daniel Sanchez-Crespo Dalmau, *Core Techniques and Algorithms in Game Programming*, NEW RIDERS, pp. 228-236, 2003.

[2] 이세일, “타일맵에서 A* 알고리즘을 이용한 유닛들의 길찾기 방법 제안”, 한국컴퓨터정보학회지, vol. 9, no. 3, pp. 71-77, 2004.

[3] K. Pallister, *Game Programming Gems 5*, 정보문화사, 류광 역, pp. 452-476, 2006.

[4] http://en.wikipedia.org/wiki/Platform_game

[5] 김현태, 김경중, “A* 경로계획과 규칙기반 시스템을 이용한 Geometry Friends 인공지능 개발”, 한국컴퓨터종합학술대회 논문집, vol.6, pp. 1532-1534, 2013.

[6] http://gaips.inesc-id.pt/geometryfriends/