A* 경로계획과 규칙기반 시스템을 이용한 Geometry Friends 인공지능 개발

김현태 ○ 김경중

세종대학교 컴퓨터공학과

kimgus0416@hanmail.net, kimkj@sejong.ac.kr

Development of Artificial Intelligence for Geometry Friends Using A* Path Planning and Rule-based Systems

Hyun-tae Kim○ Kyungjoong Kim

Dept. of Computer Engineering, Sejong University

요 약

경로탐색은 게임 프로그래밍 안에서 상당히 중요한 주제이다. 본 논문에서는, 플랫폼 게임의 한 종류인 Geometry Friends에서 변형된 A*알고리즘과 규칙기반 시스템을 이용한 방법을 제안하였다. 플랫폼 게임에서 발생할 수 있는 A*알고리즘의 문제점을 지적하고 기존 규칙기반만을 이용한 인공지능의 한계를 알아본다. 본 연구에서 제안하는 방법으로 인공지능을 개발하고 실험을 통하여 사람과 인공지능간의 움직임을 직접 비교해 보았다.

1. 서 론

플랫폼 게임(플래포머)은 플랫폼(발판)이 등장한다. 캐릭터는 점프를 이용하여 플랫폼을 옮겨 다니거나 장애물을 피한다. 대표적인 플랫폼게임으로 슈퍼마리오, 동키 콩, 메이플스토리가 있다. 본 논문에서 다루는 Geometry Friends 게임 또한 플랫폼 게임으로 분류할 수 있다.

Geometry Friends는 IEEE 2013 Conference on Computational Intelligence in Games(CIG2013) 에 새로운 게임 대회 중 하나로서, 두 캐릭터를 이용하는 컴퓨터 게임이다. 이 퍼즐 게임 플랫폼은 2D 환경에서 플레이를 할 수 있는데 중력과 마찰이 존재한다. 그리고 두 캐릭터는 각각 좌우로 움직일 수 있으며 초록 사각형은 상하방향키를 이용하여 수직과 수평으로 모양을 변형 할 수 있다 (단, 질량은 변하지 않는다.). 노란 공은 상하방향키를 이용하여 부피가 커지거나 점프를 할 수 있다. 그 게임의 목표는 이 두 캐릭터를 이용하여 최대한 짧은 시간 내에 지도에 존재하는 다이아몬드를 다 얻는 것이다. 그러기 위해서는 가장 최적의 움직임을 하면서 빠른 시간 안에 장애물들을 피하여 모든 다이아몬드를 얻어야 한다.

최적의 움직임을 보이기 위해서는 목표 지점까지의 최단 거리를 알 수 있는 경로탐색 방법이 요구된다. 이런 경로탐색 방법은 게임 프로그래밍 안에서 상당히 중요한 주제이다. 경로탐색 방법은 Dijkstra 알고리즘, Best-First-Search, A* 알고리즘 등 다양한 경로탐색 알고리즘이 존재한다[1]. 그 중 A* 알고리즘은 최소 비용 경로를 찾는 가장 빠른 알고리즘으로 알려져 있다[2].

다양한 게임분야에서 A*를 사용하지만, 그대로 사용하기보다는 각자의 환경에 맞게 변형을 시켜 사용하는 경우를 많이 찾아볼 수 있다. 왜냐하면, A*를 그대로 사용할 경우 지도의 크기가 커질수록 계산해야 하는 노드의 수가 증가하기 때문이다. 그에 따라 열린 목록과 닫힌 목록에 많은 양의 노드가 추가·삭제되어, 속도가 늦어지는 문제가 발생한다[3]. 본 논문에서는, Geometry Friends라는 게임을 위하여 변형된 A*를 제안하고 규칙기반 시스템을 함께 사용하여 인공지능을 개발하였다.

2. Geometry Friends 의 특수성

Geometry Friends는 플랫폼 게임의 하나이기 때문에 플랫폼이 존재한다. 이 플랫폼은 발판이 되기도 하고 장애물이 되기도 한다. 타 플랫폼 게임과 같이, Geometry Friends도 플랫폼을 옮겨 다니거나 장애물을 피하는 것에서 점프가 중요한 요소이다. 하지만, Geometry Friends는 점프를 하는 노란 공과 모양을 변형 하는 초록 사각형을 동시에 플레이 한다. 점프도 중요하지만 두 캐릭터간의 “협동”이 더 중요한 요소이다. 두 캐릭터를

잘 활용하여야 제한된 시간 안에 빠르게 지도에 있는 모든 다이아몬드를 획득할 수 있다.

좋은 "협동"을 하기 위해서는 개별적인 캐릭터의 효율적인 행동이 우선시 되어야 한다. 본 논문에서는, 각 캐릭터의 행동을 결정하는 인공지능을 개발하는 데 초점을 맞추었다. 두 캐릭터가 다른 움직임을 보이기 때문에, 규칙기반만을 이용한 인공지능으로 캐릭터의 행동을 결정하게 된다면 비효율적인 움직임을 보인다. 본 실험에서는, A*를 이용하여 각 캐릭터와 최단거리에 존재하는 다이아몬드를 먼저 얻어서 최적의 움직임을 결정한다.

3. A* 알고리즘

A* 알고리즘이란 인접한 노드들을 조사해가면서 출발 노드부터 목표 노드까지 가장 싼 비용의 경로를 찾는 알고리즘을 뜻한다. 기본적으로 A*는 열린 목록(Open list)과 닫힌 목록(Closed list)을 가진다. 열린 목록은 아직 조사하지 않은 상태들을 담은 목록이고 닫힌 목록은 이미 조사한 상태를 담은 목록이다. A* 에서는 출발 노드에서 현재 노드 x까지의 비용 $g(x)$와 현재 노드 x부터 목표 노드까지의 휴리스틱 비용 $h(x)$을 더하여 총 비용 $f(x) = g(x) + h(x)$ 을 구한다. 현재 노드에서 인접한 노드 중 $f(n)$ 이 가장 작은 노드를 선택하여 이동하게 된다. 목표 노드에 도달할 때까지 이 과정을 계속 반복을 한다. 각 노드는 가장 적은 비용으로 자신에 이르도록 한 노드(부모 노드)를 가리키는 포인터를 가진다. 검색이 끝나고 나서 목표 노드로부터 그 포인터를 따라 거꾸로 되짚어가면서 가장 적은 비용의 경로를 찾아낸다[2,3].

A*는 주어진 출발지와 최종 목적지까지 가는 최단 경로를 찾아내며 장애물을 피하여 막다른 골목에 빠지지 않고 길을 찾는다. 하지만, 검색을 해야 할 공간이 커지면 알고리즘의 성능은 급격히 떨어지고 CPU와 메모리 사용량도 증가한다. 또한, A* 알고리즘이 빈번하게 호출되면 CPU 자원의 요구량은 매우 커진다[2].

이러한 문제가 Geometry Friends 게임에서 상당히 크게 나타난다. Geometry Friends는 지도의 형태가 다양하고 각 지도마다 목적지가 되는 다이아몬드의 수가 다양하다. 캐릭터는 여러 번 A*를 반복하여 다이아몬드 중 캐릭터와 최단거리에 놓여진 다이아몬드를 찾기 때문에 CPU 자원의 요구량이 커진다. 기본적으로 지도의 크기가 크기 때문에 CPU와 메모리 사용량도 증가한다.

본 논문에서는, 큰 검색 공간에서 실용성을 높이기 위해 플랫폼 게임에서 고려할 수 있는 변형된 A*알고리즘을 제안한다. 이 변형된 알고리즘을 LBA*(Limitation of Behaviors A*) 알고리즘이라 부른다.

4. 제안하는 방법

[1]과 [2]에서 A* 알고리즘을 최적화하기 위한 다양한 방법들이 소개되었다. 해당 절에서는 조금 더 제한적인 행동을 하는 Geometry Friends 게임 내에서 LBA* 알고리즘을 제안한다. 예를 들면, 초록 사각형은 점프하는 것이 불가능하다. 이럴 경우, 자신이 행동할 수 있는 범위 이외에 검색은 불필요하고 결과적으로 CPU와 메모리 사용량을 증가시킨다. 이와 같은 문제에서는 좀 더 근본적인 알고리즘에 의한 각 노드로의 접근횟수를 줄이는 것이 필요하다.

4.1 제한적 행동범위에 따른 영역 축소

기존 A* 알고리즘의 경우 타일 형태의 지도 안에서 전체 범위에 대해 알고리즘을 수행하는 것이 일반적이다. 제안하는 방법은 출발 노드가 갈수 있는 방향이 제한 된 경우, 접근이 불가능한 영역을 임시 장애물로 인식하여 탐색 영역을 축소한다. 이와 같이 영역을 축소시키는 방법을 제안하는 LBA* 알고리즘이라고 부른다. 이 경우 접근이 불가능한 영역은 탐색을 하지 않기 때문에 열린 목록에 불필요한 부분이 삽입되지 않는다.

4.2 제안하는 알고리즘

제안한 방법은 알고리즘을 수행하기 전에 미리 자신이 움직일 수 있는 영역에 대하여 판단을 한다. $n \times m$의 노드로 이루어진 지도에 각 노드 N_{ij} $(i = 1, \dots, n$, $j = 1, \dots, m)$ 는 여러 상태의 값들을 가진다. 그 중 장애물을 $\mathcal{O}$, 캐릭터가 이동할 수 없는 영역을 $\mathcal{T}$ 라 할 때, $\{\mathcal{O}_{ij}, \mathcal{T}_{ij}\} \in N_{ij}$ 가 성립한다.

$$\mathcal{O} = \begin{cases} \text{참} & \text{노드 } N_{ij} \text{가 장애물일 경우} \\ \text{거짓} & \text{그렇지않으면} \end{cases}$$

$$\mathcal{T} = \begin{cases} \text{참} & \text{캐릭터가 이동할 수 없는 영역일 경우} \\ \text{거짓} & \text{그렇지않으면} \end{cases}$$

A*를 실행하기 전에 각 $\mathcal{O}$, $\mathcal{T}$ 의 값을 갱신하여 준다. 그 후 A*알고리즘을 수행할 때, 현재 노드가 n인 경우 n주위의 노드가 갈 수 있는 영역인지 검사를 한다. 그

때 O와 T의 값 중 하나라도 참이라면 갈 수 없다고 판단을 한다.

5. 제안하는 방법을 이용한 실험

그림 1과 같은 지도에서 규칙기반만으로 인공지능을 개발할 경우, 사이에 장애물이 존재를 하여 캐릭터와 다이아몬드와의 거리는 상당히 길다. 하지만, 캐릭터는 다이아몬드 사이에 장애물이 있다는 것을 인식할 수 없어서 장애물과 계속 충돌한다. 이와 같은 상황이 발생하는 이유는 모든 경우에 대한 규칙기반 시스템을 만들기 힘들기 때문이다. 그래서 4절에서 제안한 LBA*를 이용하여 실제 최단거리에 위치한 다이아몬드를 파악하고 그 경로를 따라 규칙기반 시스템을 사용하여 이동한다면 최단시간에 모든 다이아몬드를 획득할 수 있다.

5.1 실험 조건

본 게임은 1240x720 픽셀로 영역이 나누어져 있어서 40x40 픽셀이 한 노드인 31x18 크기의 격자형 지도를 만들었다. 지도는 그림 1와 같다. 다른 변수를 제거하기 위하여 두 캐릭터 중 초록 사각형을 움직일 수 없도록 가두고 노란 공만 움직이게 설정하였다. 본 실험에서는, 규칙기반만을 이용한 인공지능의 단점을 보여주기 위하여 실제로 가까운 캐릭터와 다이아몬드 사이에 긴 장애물을 만들었다.

5.2 실험 및 분석

그림 1에서, 사람과 LBA*와 규칙기반을 이용한 인공지능 그리고 오직 규칙기반을 이용한 인공지능을 비교하는 실험을 수행하였다. 규칙기반만을 이용한 실험의 경우, 바로 위에 있는 다이아몬드를 획득하기 위해 장애물과 계속 충돌하였다. 하지만 LBA*와 규칙기반으로 이용한 실험의 경우, 사람과 매우 유사한 궤적으로 움직였다. 비록 모든 결과가 좋지는 못하였으나, 최고기록 같은 경우 사람이 16초, 제안한 방법이 15초로 모든 다이아몬드를 얻는 시간은 더 빠른 것을 볼 수 있었다.

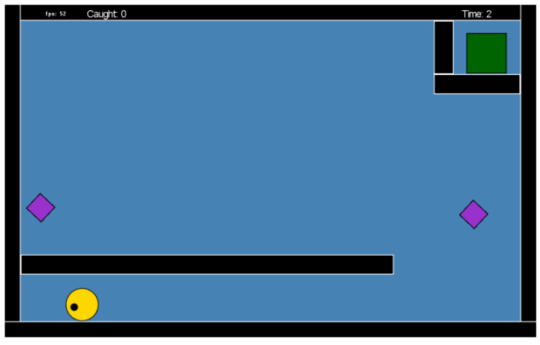

그림 1 실험을 위해 만든 지도

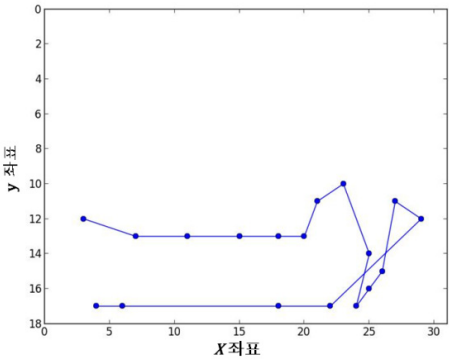

그림 2 사람이 움직인 궤적

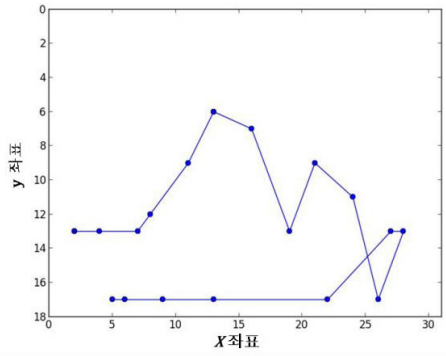

그림 3 LBA*와 규칙기반으로 움직인 궤적

6. 결론 및 향후 연구

본 논문에서는, A*알고리즘을 변형한 방식과 규칙기반 시스템을 이용한 인공지능 개발을 제안하였다. 제안된 LBA*와 규칙기반으로 사람이 생각하는 경로와 매우 유사한 경로로 움직일 수 있음을 보여주었다. 하지만, 사람과 같은 자연스러운 움직임을 위해서는 개선이 필요하다. 앞으로는 제안된 LBA*를 기반으로 진화 알고리즘으로 향상된 움직임을 수행하도록 연구를 진행할 예정이다.

7. 감사의 글

이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 뇌과학 원천기술개발사업임 (2010-0018950)

REFERENCES

[1] 조성현, “경로 정보를 이용한 길찾기 알고리즘,” 한국게임학회지, vol. 13, no. 1, pp. 31-40, 2013.

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

[3] B. Stout, “The Basics of A* for Path Planning,” *Game Programming Gems*, Charles River Media, pp. 254-263, 2000.