그리드 기반 게임에서 Voronoi diagram을 이용한 곡선 경로찾기

전현주[0] 유견아
덕성여자대학교 컴퓨터과학과
{hzoo[0], kyeonah@duksung.ac.kr}

Finding a smooth path by using a Voronoi diagram in grid-based games

Hyunjoo Jeon[0] Kyeonah Yu
Dept. of computer science, Duksung Women's University

요 약

컴퓨터 게임에서 움직이는 캐릭터를 위한 자연스러운 경로의 계획은 게임의 현실감을 측정하는 중요한 척도이다. Voronoi 다이어그램은 컴퓨터 기하학 분야에서 잘 알려진 로봇 경로계획 알고리즘 중 하나이다. Voronoi 다이어그램은 두 장애물로부터 같은 거리에 있는 선분과 점들로 구성되어 생성된 경로가 장애물로부터 멀리 떨어진 안전한 길이고 사람이 실제로 택하는 경로와 유사하다는 것이 장점이다. 본 논문에서는 셀-기반 게임 환경에서 움직이는 캐릭터의 이동경로를 찾기 위해 Voronoi 다이어그램을 적용하는 방법을 제안하고 구현한다. 제안된 방식과 기존에 많이 사용되던 그리드 기반 A* 알고리즘의 적용 결과를 비교 분석한다.

1. 서 론

최근 컴퓨터 게임은 점점 등장하는 캐릭터가 다양해지고 현실감을 증대하는 방향으로 발전하고 있다. 그러나 아직까지 주인공 이외의 캐릭터(이하 NPC, nonplayer character)들은 각자의 목표 지향적(goal-directed) 운동을 한다기보다 미리 계산되어 주어진 단순 운동을 반복하는 경우가 많다. 게임 사용자가 현실감을 느끼게 하기 위해서는 NPC들이 서로 충돌을 회피하며 목적 있는 움직임을 보여 주어야 한다. NPC가 움직이는 경로를 계획하는 일은 이를 위한 첫걸음이라고 할 수 있다. 컴퓨터 게임에서 경로 찾기에 많이 사용되는 인공지능 알고리즘은 A* 알고리즘이다. 게임의 배경을 사각형의 그리드, 혹은 볼록 다각형으로 분할하고 각각의 셀을 노드로 하여 목표로의 도달을 최대로 보장하는 이웃 노드로 이동하는 경로를 계획한다. 이와 같은 기존의 방식은 노드수가 많아 탐색공간이 커져 알고리즘 수행의 효율성이 문제가 될 뿐 아니라 찾아진 경로도 이웃하는 노드로 이동할 때마다 꺾임이 있는 형태로 나와 이를 해결하기 위한 여러 연구가 진행되고 있다[1][2].

대표적인 로봇 경로 찾기 알고리즘중 하나인 Voronoi 다이어그램은 두 장애물에 이르는 거리가 같은 점들로 이루어진 다이어그램이며 이는 장애물에서 멀리 떨어진 자유공간에 경로를 생성하도록 해준다. Voronoi 다이어그램은 직선과 포물선으로 구성되어 있으며 로봇의 시작점으로부터 가까운 다이어그램의 한 점으로 들어가면 다이어그램을 따라 경로가 형성되고 목표점과 가까운 곳에서 빠져 나가도록 경로를 계획하는 방식이다. 로보틱스에서 이 방식은 로봇에게 장애물과 멀리 떨어진 안전한 경로를 따라 가도록 계획해 주는 것이 장점으로 알려져 있다 [3]. 실제 사람들이 시야가 트인 채, 장애물 사이를 걸어가야 할 때 택하는 경로와 가장 유사한 경로를 제공해 준다.

본 연구에서는 셀 기반으로 배경화면이 생성되는 컴퓨터 게임에서 여러 모양의 장애물의 Voronoi 다이어그램을 계산하고 이를 이용해 움직이는 캐릭터를 위한 이동 경로를 찾는 알고리즘을 제안한다. 셀 기반으로 생성되는 장애물의 경계가 매끄럽지 않기 때문에 우선 경계선을 추정하여 정점의 좌표를 구한 후 Voronoi 다이어그램을 구한다. Voronoi 다이어그램에 의한 결과와 기존의 방식에 의해 생성된 경로와 비교하여 장단점을 분석해본다.

2. Voronoi 다이어그램

원래 Voronoi 다이어그램은 점들의 집합에 대해 정의되어 있다. 같은 직선상에 있지 않은 점들의 집합 $P=\{p_1,p_2, \cdots, p_n\}$가 있을 때, 점 p_i의 Voronoi 영역(Voronoi region) $R(p_i)$는 다음과 같이 정의된다[4].

$$R(p_i) = \{p: |p_i-p| \leq |p_j-p|, \ \forall j \neq i\}$$

즉, Voronoi 사이트(site) p_i에 대해 $R(p_i)$는 p_i에 가장 가까운 점들의 집합이다.

Voronoi 다이어그램은 그 이론의 완벽함 뿐 아니라 다양한 응용 분야로 인해 많은 연구가 되었는데 경로찾기(pathfinding)는 이들 응용 분야 중 하나이다. 보통 Voronoi 다이어그램에 의해 찾은 경로는 장애물을 최대한으로 멀리하여 움직임으로서 충돌 위험을 최소화하는 해주는 장점이 있다. 장애물의 모양이 다양하기 때문에 점 Voronoi 다이어그램을 확장한 버전의 Voronoi 다이어그램을 이용해야 한다. 확장된 Voronoi 다이어그램의 경우, 이등분선은 장애물의 기하학적 요소에 따라 다음 3가지 경우로 분류된다.

1. 정점과 정점 - 직선 (점과 점 사이의 이등분선)
2. 모서리와 모서리 - 직선 (선분과 선분의 이등분선)
3. 모서리와 정점 - 포물선(선분과 점에 이르는 거리가 같은 궤적)

그림 2.1은 이 경우에 의해 생성된 Voronoi 다이어그램을 보여준다. 좌측 장애물의 정점v를 기준으로 살펴보면, v와 우측 장애물 모서리 e_1과의 이등분선은 그 사이의 이등분점을 정점으로 하는 포물선이 되며 v에서 e_2의 법선 방향으로 포물선과 만

나는 점부터 e_1과 e_2 사이의 이등분선이 두 장애물 사이의 보로노이 모서리가 된다. e_1과 e_3 사이에서도 마찬가지로 Voronoi 영역을 구할 수 있다.

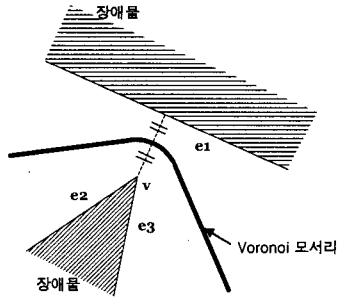


그림 2.1 다각형 사이의 Voronoi 다이어그램

3. 경로찾기 구현 및 실험

최근 스타크래프트와 같은 게임에서 사용하는 맵 에디터는 셀 단위를 기반으로 한다. 셀 기반 에디터로 제작된 맵에 Voronoi 다이어그램을 적용하기 위해서는 근접한 경계선을 추정한 후에 Voronoi 다이어그램을 계산하고 경로를 찾아낸다.

3.1 맵 에디터

본 연구에서는 배경을 128*128의 정사각형 타일로 나누고 각각의 타일은 16*16의 도트(dot)로 이루어진 셀-기반 맵 에디터를 사용한다. 그림 3.1에 나오는 6가지 지형을 사용하였으며 윈도우 상의 결과를 보기 위해 27*22의 맵-데이터를 이용한다. 3등분된 윈도우의 오른쪽은 실제 사용자가 마우스를 이용해 제작한 맵의 모양을 보여주고, 왼쪽 상단은 현재 표현 가능한 지형들을 표시한다(그림 3.2). 왼쪽 하단은 전체적인 맵의 모양을 표시하는 미니-맵(mini map)을 위치시키며 현재 표현되고 있는 부분은 사각형 틀 안에서 보여주고 있다. 사용자는 왼쪽 상단의 지형을 선택한 후, 오른쪽 윈도우에 원하는 맵을 만든다.

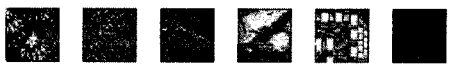


그림 3.1 16*16 도트의 타일들

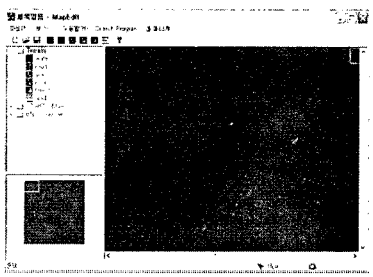


그림 3.2 맵-에디터의 구성

3.2 경계선 추정

위와 같이 제작된 맵은 그림 3.3과 같이 셀 단위로 장애물이 생성되는데 우선 영상처리 분야에서 이미지의 경계선을 추출하는데 사용되는 Sobel mask 연산을 픽셀단위 대신 지형의 정보를 가지고 있는 셀 단위로 응용 적용하여 장애물의 경계선을 찾아낸다[5].

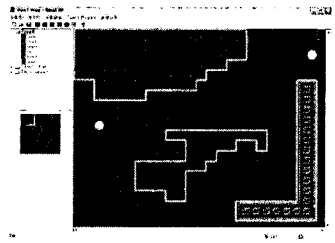


그림 3.3 셀기반 에디터에 의해 생성된 예제 맵

그러나 추출된 경계선은 그림3.3과 같이 셀 단위로 이루어진 맵 에디터의 특성대로 수직, 수평 방향으로만 이루어져 있어 계단모양으로 나타난다. 이와 같이 불필요한 정점이 생성되면 Voronoi 다이어그램의 계산에도 좋지 않은 영향을 줄 뿐 아니라 Voronoi 다이어그램 자체도 꺾임이 많은 곡선 모양이 되어 경로를 계획하는 데에 바람직하지 않다. 그러므로 수평, 수직 모서리가 일정한 기준 이상 반복되는 구간을 대각선화하여 경계선을 추정한 후, 모서리의 식을 추출한다. 수평, 수직 연속구간의 기준을 5 셀로 하였을 때, 계단모양의 경계를 확장하여 대각선으로 나타낸 결과를 그림 3.4에 보여준다.

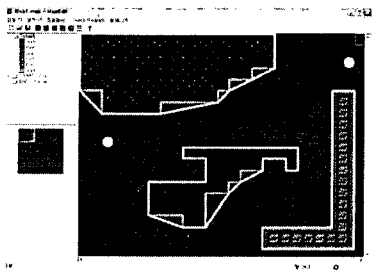


그림 3.4 경계선 추정 결과

3.3 Voronoi 다이어그램 생성 및 경로 찾기

wall과 water, forest, rock을 장애물로 간주한 환경 위에 시작점과 목표점을 설정하고 Voronoi 다이어그램을 구하는 과정은 장애물의 기하학적 요소에 따라 다음의 세 가지 경우로 진행된다.

① 장애물의 정점과 정점의 경우 대각선 방향과 위, 아래 모든 방향의 셀에 대한 정보를 탐색하여 중점을 구한다.

② 각 장애물의 경계선 부분에 위치한 선과 선의 경우 각 선의 법선들이 만나는 점을 이용하여 중선을 그린다.

③ 한 장애물의 정점과 다른 장애물의 선이나 윈도우 벽의 경우 포물선의 방정식을 이용하여 중선을 그려준다. 이때, 점과 선의 중선이 ②의 경우에 의해 그려진 선과 만나게 되면 현재 계산하고 있는 과정을 중지한다.

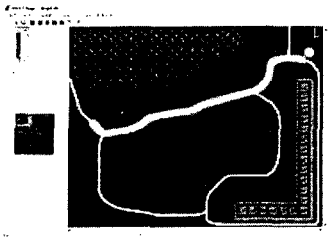


그림 3.5 적용한 Voronoi 다이어그램 결과

이와 같이 생성된 Voronoi 다이어그램(그림 3.5)을 이용하여 경로를 찾는 과정은 다음과 같다.

1단계: 시작점이 가장 근접한 다이어그램 상의 점을 찾아 이동

2단계: 현재 위치로부터 목표점까지 최적경로를 계산하면서 다이어그램을 따라 이동

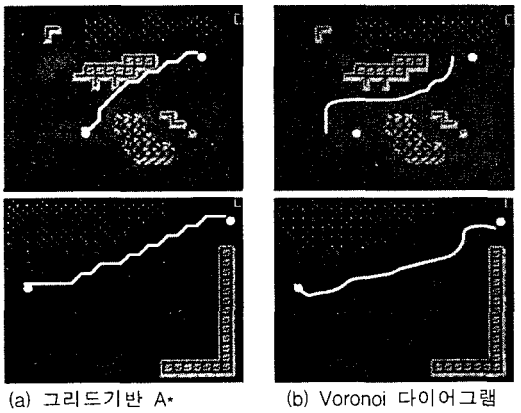
3단계: 계산되어 나가는 위치가 목표점의 셀과 일치하면 계산을 멈추고 이동을 중지

목표점까지의 최적경로를 판단하는 기준은 게임의 목적에 따라 여러 가지로 적용될 수 있으나 본 연구에서는 가장 일반적이면서 간단한 기준인 유클리디언 거리를 이용하였으며 그림 3.5에서 Voronoi 다이어그램위의 굵은 선은 위의 순서에 따라 진행된 경로 찾기의 결과를 보여준다.

3.4 Simulation 결과

셀 기반 지형을 생성하기 위한 맵-에디터는 VC++를 이용해 제작하였다. 제작된 맵은 위치정보와 각 위치에 따른 셀의 정보를 2차원 배열의 맵 데이터(map data)로 저장하게 된다. 이렇게 저장된 맵 데이터의 값에 따라 셀 테이블(cell table)의 데이터들을 호출하여 게임을 이룬다[6][7]. 여기서 셀-테이블은 각 셀의 비트맵에 따른 고유 플래그를 저장하게 되고 플래그의 종류에 따라 이동가능 여부를 결정하게 된다.

Voronoi 다이어그램을 이용하여 생성된 경로와 기존에 사용되는 그리드 기반 A* 알고리즘의 결과와 비교하였다. 그리드 기반의 경우 시작점을 중심으로 주변의 이웃하는 8개(수평, 수직, 대각선 방향)의 셀을 노드로 A*알고리즘을 적용하여 최소거리를 가지는 이웃 셀로 이동하면서 같은 과정을 반복한다.



(a) 그리드기반 A* (b) Voronoi 다이어그램

그림 3.6 결과 비교

그림 3.6은 두가지 서로 다른 배경에 대한 그리드 기반 A* 알고리즘에 의한 결과와 Voronoi 다이어그램에 의한 결과를 보여주는데 전자는 서론에서 지적한 바와 같이 이웃한 셀로 이동할 때마다 경로가 꺾여서 캐릭터가 이 경로를 따라 움직이는 것이 자연스럽지 않은 반면 Voronoi 다이어그램에 의한 경로는 장애물의 사이를 적당한 곡선을 그리며 진행하여 사람이 움직이는 것과 크게 다르지 않은 자연스러운 경로를 찾는 것을 확인할 수 있다.

4. 결론

본 연구에서는 컴퓨터 게임에서 자연스러운 경로를 찾기 위해 Voronoi 다이어그램을 이용하였다. 셀 기반 에디터로 제작된 게임에서 Voronoi 다이어그램을 이용하기 위해 장애물의 경계선의 식을 추정하여 구하였으며 이렇게 구해진 경로는 기존에 많이 사용되고 있는 경로찾기 방법인 그리드 기반 A* 탐색 알고리즘 결과와 비교하였을 때, 더욱 자연스러운 경로를 찾음을 알 수 있었다. 향후에는 유틀리디언 거리 이외에 그리드 기반 탐색에서 그리드 단위로 저장하고 탐색에 이용하던 여러 정보(지형의 질, 고도 등)들을 어떻게 Voronoi 다이어그램에 반영할 것인가에 대한 연구와 다각형 장애물 이외의 다양한 형태의 장애물로의 확장에 대한 연구도 필요하다.

[참고문헌]

[1] M. Pinter, "Towards more realistic pathfinding", Game Developer Magazine, March, 2001

[2] A. Botea, M. Muller, J. Schaeffer, "Near optimal hierarchical path-finding", Journal of game development, vol 1(1), pp 1-22, 2004

[3] Robot Motion planning, J.C. Latombe, Kluwer Academic Publishers, 1991

[4] Computational Geometry in C, J. O'Rourke, Cambridge, 1998

[5] Digital Image processing Algorithms and Application, I. Pitas, 2000

[6] Ai Game Programming Wisdom, Rabin, Steve (EDT), Charles River Media, 2002

[7] Game Programming Gems , Deloura, Mark (EDT), Charles River Media, 2000