MFC를 이용한 Search Algorithm 학습 프로그램

단국대학교 공과대학 응용컴퓨터공학과
백 현 오
sss42850@gmail.com

Search Algorithm learning program using MFC

Hyunoh Baek
Dept Applied Computer Engineering Dankook Univ

요약

본 프로젝트는 단국대학교 2018년 2학기 로봇공학 개론의 프로젝트 결과물로 MFC를 이용한 Search Algorithm 학습 프로그램을 목표로 진행되었다.
사용자는 지도 정보를 수정해 원하는 탐색 알고리즘을 선택해 학습할 수 있고, 결과로 나오는 경로에 대해 Smoothing 알고리즘도 수행해 볼 수 있다.

1. 서론

본 프로젝트는 수업 중에 배운 탐색 알고리즘들의 이해를 돕기 위한 프로그램을 만드는 것을 목표로 사용자가 직접 맵의 크기, 출발지, 도착지, 장애물을 설정하여 다양한 경우의 맵을 생성해보고, 생성한 맵을 따라 탐색 알고리즘 BFS, DFS, Greedy, A*를 사용자가 설정한 방향 순서에 맞게 테스트해보고 학습할 수 있는 프로그램을 만드는 것으로 진행되었다.

프로젝트에 사용되는 언어는 C, C++을 사용하였다. IDE로는 Visual Studio를 이용을 했다. Visual Studio 안에 내장된 MFC를 이용해 윈도우 API를 사용했다.

2. 프로젝트 구성 및 구현 방법

2.1 전체 구성

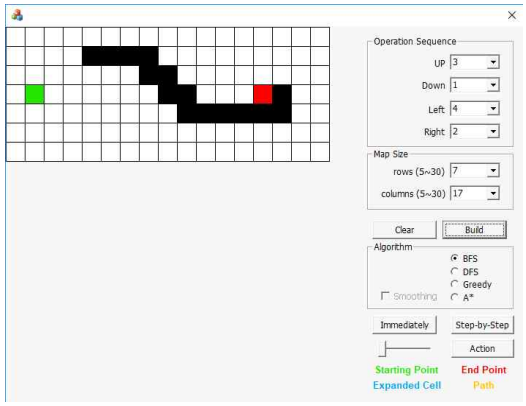
전체 실행 부분은 두 가지로 나뉜다.
첫 번째는 지도를 사용자의 설정에 맞게 변환하는 과정이다.
두 번째는 그 지도를 이용하여 탐색 알고리즘을 통해 단계별 진행을 통하여 학습하는 것이다.

2.2 소스 코드 구현 설명

2.2.1 맵 설정 단계

사용자로부터 맵의 사이즈를 combo box를 이용해 받아와 빈 맵을 만든다. 이때 출발지와 도착지는 자동으로 설정이 된다.

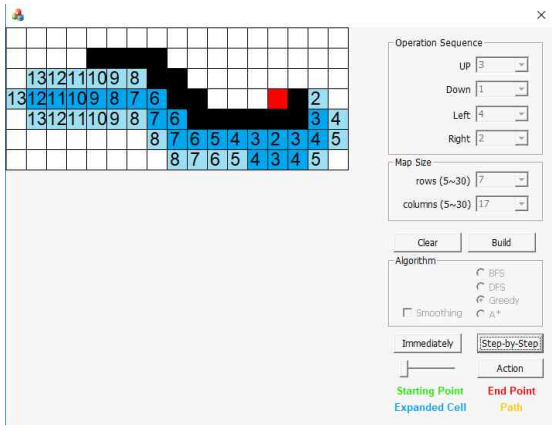
출발지의 위치와 도착지의 위치는 마우스 클릭, 드래그를 이용하여 움직일 수 있게 하였다. 이때, 출발지와 목적지는 장애물 위에 놓일 수 없고, 맵 밖으로 나가는 경우를 방지해 놓았다. 장애물의 위치도 마찬가지로 클릭, 드래그를 이용하여 생성할 수 있다. 장애물의 삭제는 클릭만 가능하게 구현을 해 놓았다. 맵은 Build 버튼을 이용해 초기화 및 선언을 할 수 있고, 맵의 설정값은 지우지 않고 아래 알고리즘의 실행과정만 지우는 Clear 버튼이 있다. [그림1] 은 사용자가 직접 맵을 수정한 그림이다.



[그림 1]

2.2.2 Search 알고리즘

탐색 알고리즘으로는 BFS, DFS, Greedy, A*를 이용하였다. BFS는 queue를 이용해 구현하였고, DFS는 stack, Greedy와 A*는 priority queue를 이용하였다. 여기서 Greedy와 A*에 우선순위 큐를 사용한 이유는 Greedy와 A*가 heuristic value인 h를 사용하기 때문이다. 두 알고리즘은 진행해 나가며 주변 cell들의 cost를 비교하여 진행해 나가는 상황이 발생하는데, 우선순위 큐를 이용해 cost를 비교할 필요 없이 push 동작과 함께 정렬하여 바로 사용하기 위해 사용하였다. 각각의 값에 vector 변수를 선언하여 이전 위치의 포인터들을 계속해서 쌓아나가 목적지에 도달할 때 차례로 꺼내어 각각의 알고리즘으로 간 최적의 경로를 나타내 준다. 사용자가 직접 radio button을 이용해 학습하고자 하는 알고리즘을 선택할 수 있게 해두었다. 또한, 알고리즘을 실행할 때 상, 하, 좌, 우와 같이 어느 방향에 대해서 우선으로 실행할 것인지 설정할 수 있는 Operation Sequence 부를 구현해 놓았다. 사용자는 이 값을 설정해 값의 순위에 따라 알고리즘이 동작하게 했다. 알고리즘의 진행 과정을 보여주기 위한 Step-by-Step 버튼은 알고리즘의 진행을 한 단계씩 보여주는 기능을 하고, Slider의 값을 조정하여 알고리즘의 과정을 동적으로 보여주는 Action 버튼을 구현했고, 마지막으로 동작 과정은 생략하고 결과만 보여주는 Immediately 버튼을 구현했다. [그림2] 는 Greedy 알고리즘의 단계별 진행 중의 그림이다.



[그림 2]

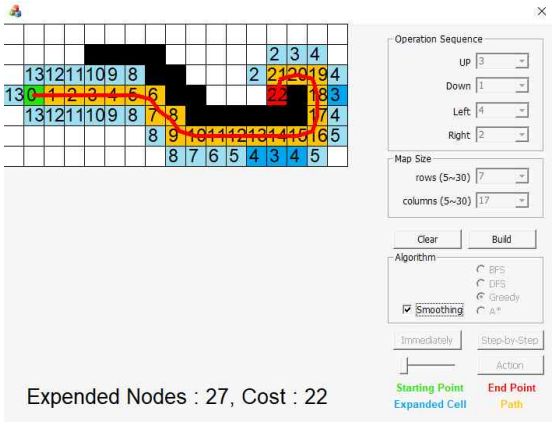
2.2.3 Smoothing 알고리즘

탐색 알고리즘을 이용해 찾은 경로를 단순히 보여주는 것뿐만 아니라, 수업 시간에 배운 Smoothing 알고리즘 중 Gradient Descent를 이용하여 경로를 부드러운 길을 만들어 표시하는

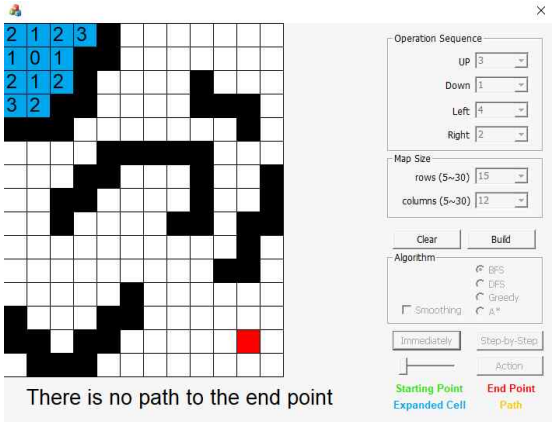
기능도 넣어 보았다. 알고리즘에 들어가는 가중치 값은 미리 설정해두었다. Smoothing은 check box로 구현해 두었다. 평상시에는 박스가 비활성 상태가 되어있어 체크할 수 없으나, 도착지를 찾아 알고리즘이 종료되면 박스가 활성 상태가 되어 체크를 하게 되면 부드러운 이동 경로를 확인 할 수 있다.

2.3 실행 결과

알고리즘에 따라 진행해 나가다가 도착지에 도달하게 되면 경로에 해당하는 Expanded Cell들이 Path로 바뀌게 되고 Path를 따라 진행해 나감에 따른 cost가 새로 적히게 된다. 프로그램의 하단에는 Expanded 된 노드들의 수와 실제 이동 시에 필요한 cost가 적어진다. 만약 출발지로부터 도착지까지의 경로가 존재하지 않는다면 클리어와 빌드버튼을 제외하고 모든 버튼이 비활성 상태가 되며, 도착지까지의 경로를 못 찾았다는 메시지를 띄운다. [그림 3.a] 그림은 Greedy 알고리즘을 통해 도착지에 도달한 결과 모습과 Smoothing을 체크해 부드러운 경로도 같이 나타내고 있다. [그림 3.b]는 경로를 찾지 못한 그림을 나타내고 있다.



[그림 3.a]



[그림 3.b]

3. 결론

구현하면서 문제가 되었던 부분들은 많았으나 대부분 해결되어 현재 프로젝트에는 큰 문제가 없어 보인다. 이 프로젝트를 만들며 다시 한번 느끼게 된 것이 있다면, 수업을 들을 때는 이해가 되었다고 생각하고 집에 돌아가 다시 한번 볼 때는 아무리 필기를 다시 보고 생각해보아도 떠오르지 않는 학생들을 위해 이와 같은 교육목적의 프로그램이 많이 개발되어 배포되었으면 한다.