

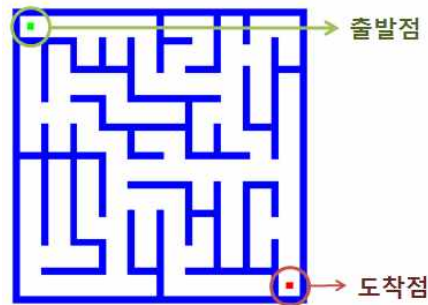
1. 목 적 : 미로 길 찾기

본 실험에서는 2주차의 숙제를 통하여 만든 MFC 응용 프로그램의 DFS 버튼을 클릭했을 때 DFS 방법으로 미로를 탐색하여 그 결과를 화면에 표시하는 프로그램을 작성한다.

2. 프로젝트 과제

DFS에 의한 미로 길 찾기

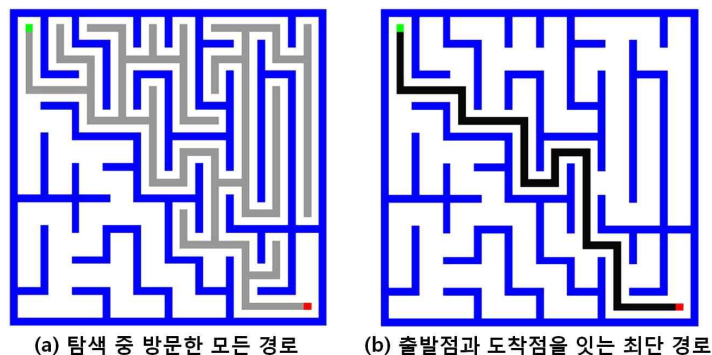
미로 길 찾기 문제는 2주차까지 설계한 미로의 출발점과 도착점을 연결하는 가장 짧은 경로를 탐색해 출력하는 문제이다. NxM 미로의 출발점과 도착점은 각각 1행1열과 M행N열의 방향으로 설정한다.



[그림 1] 미로의 출발점과 도착점.

실습에서는 DFS(Depth First Search : 깊이 우선 탐색)방법으로 미로의 출발점부터 도착점을 통과하는 경로를 탐색한다. 2주차에서 생성한 MFC 응용 프로그램의 DFS 버튼을 클릭하면 DFS방법으로 탐색한 경로를 화면에 출력하도록 프로그램을 작성한다.

[그림 2]의 (a)는 DFS를 통해 문제를 해결하는 과정에서 방문하게 되는 모든 길(이하 탐색경로로 명명)을 나타내고, [그림 2]의 (b)는 (a)와 같은 미로에 대하여 출발점과 도착점을 연결하는 경로(이하 탈출경로로 명명)를 나타낸다. 실험에서는 [도구]-[DFS] 메뉴를 선택하거나 DFS 버튼을 선택하면 (a)의 탐색경로와 (b)의 탈출경로를 모두 표시한다. 두 경로는 서로 구분이 가능하도록 색을 달리해서 표시하거나 hatch등을 이용하도록 하자.



[그림 2] DFS를 통한 미로 탐색.

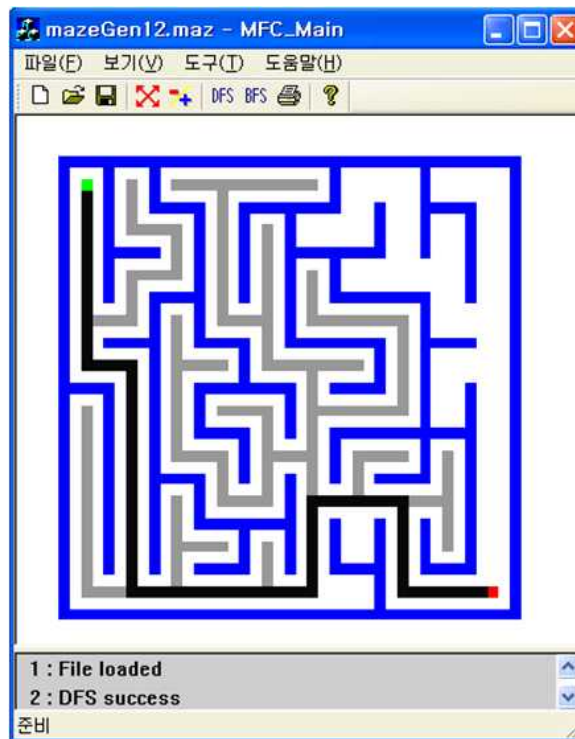
(a) 탐색경로, (b) 탈출경로

입력형식: 미로 텍스트 파일(~.maz)

제작환경: 2주차 숙제결과인 DFS, BFS 메뉴와 버튼이 추가된 MFC 응용 프로그램.


출력형식: 입력파일을 읽어 들인 후 활성화된 DFS메뉴, 버튼을 클릭하면 DFS 탐색을 수행한다. 각각 초록색 및 빨간색으로 표시된 시작점과 도착점을 연결하는 탈출경로와 탐색경로를 서로 구분이 갈 수 있도록 출력한다.

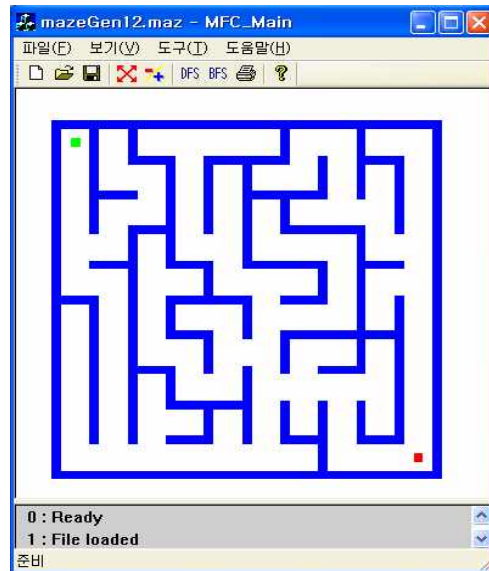
입출력 예:




[그림 3] DFS 버튼 클릭 후 출력화면.

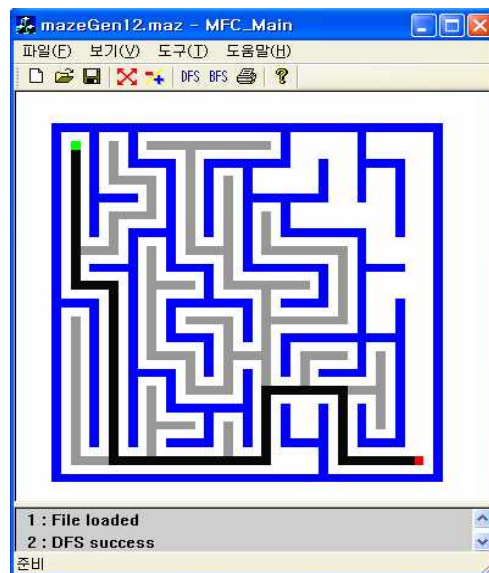
윈도우 설계 : 이번 실험을 통해 구현하려는 프로그램의 전체적인 화면은 위의 출력 예제가 된다. 이번 실험을 통해 만들 프로그램을 수행순서로 설명한다면 아래와 같다.

1. 2주차에서 만든 MFC 응용 프로그램에서 사용자가 메뉴의 “열기”나 툴바 내의 을 눌러 미로 텍스트 파일(~.maz)을 선택한 후 읽어 들이면 [그림 4]와 같이 빈 화면에 출발점과 도착점이 표시된 미로가 그려진다.



[그림 4] 미로 파일 읽어 들임.

2. 사용자가 [도구]-[DFS]메뉴, 툴바의 를 누르면 [그림 5]와 같이 화면에 DFS를 통해 미로를 탐색한 결과가 그려진다.



[그림 5] DFS 버튼 클릭.

3. 종료는 윈도우의 오른쪽 상단의 X아이콘이나 아니면 파일메뉴의 끝을 누르면 프로그램이 종료된다.

3. 실험방법

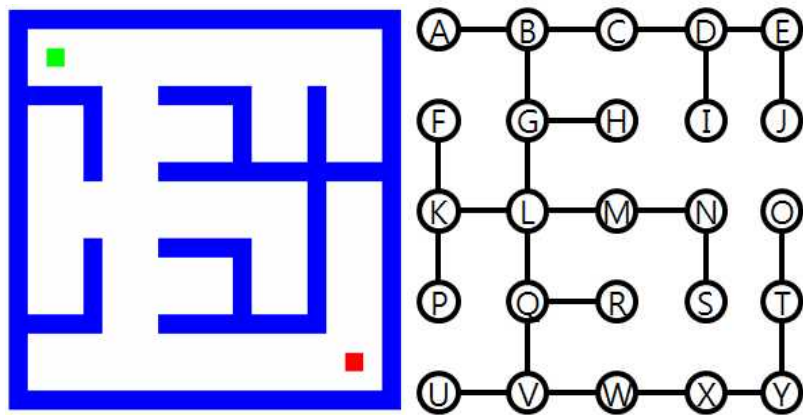
3-1 문제해결

주어진 문제를 해결하기 위해선 미로가 그래프로 표현가능하다는 사실과 그래프를 탐색하는 가장 기본적인 이론인 DFS 알고리즘을 이해하여야 한다. 2주차 실험 예비보고서를 통해 학생들 모두 DFS 알고리즘을 이해하고 있을 것으로 생각되므로 두 알고리즘에 대한 자세한 설명은 생략한다. 두 알고리즘을 구현하기 위해서는 미로를 저장하고 있는 자료구조 외에 어떤 자료구조가 필요한지 생각해보자.

3-2. 반복 함수 형태의 DFS(Depth First Search : 깊이 우선 탐색)

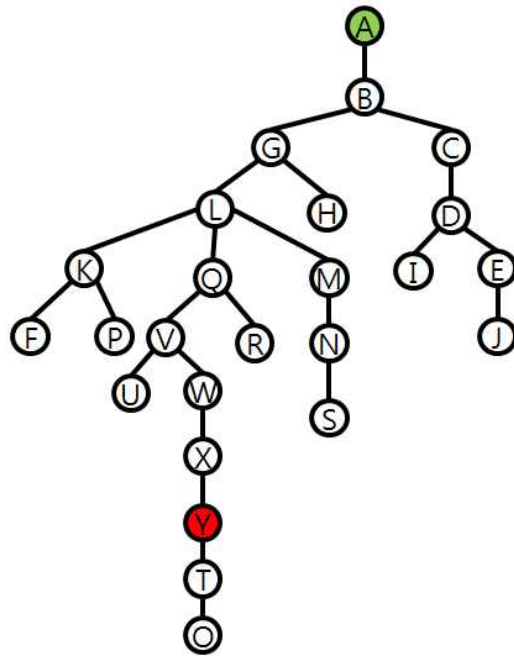
일반적으로 알려진 DFS 알고리즘은 재귀적 형태이지만, 재귀함수의 경우 함수가 자기 자신을 호출할 때 parameter를 넘겨주기 위해 컴퓨터 내부의 stack memory를 사용한다. 만약 매우 큰 사이즈의 미로가 입력될 경우 DFS과정에서 stack overflow가 발생할 가능성이 있으므로 반복(iterative)함수의 형태로 구현할 필요가 있다.

DFS 알고리즘을 반복함수 형태로 구현하기 위해선 더 이상 방문할 수 있는 자식 node가 존재하지 않는 node에서 이전 node로 되돌아가기 위하여 현재까지 거쳐 왔던 node들을 저장할 stack이 필요하다. [그림 6]처럼 미로를 일종의 그래프로 생각하자.



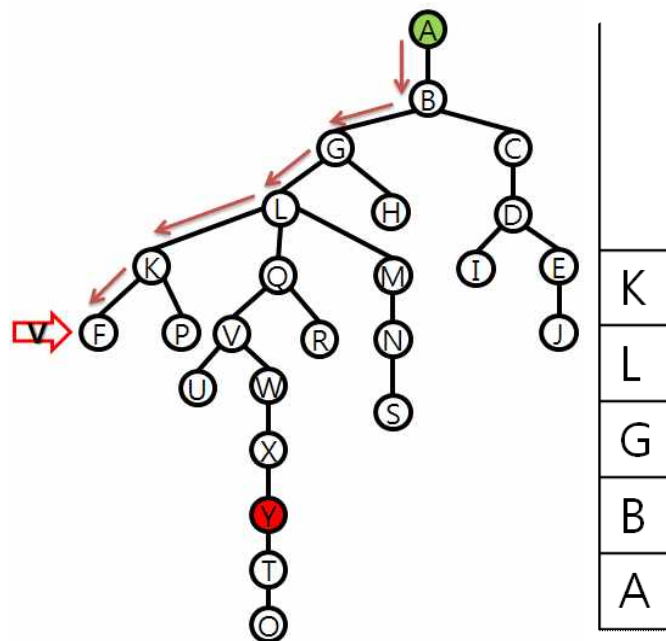
[그림 6] 미로(좌)에 대응하는 그래프(우)

[그림 6]의 오른쪽 그래프를 node A를 root로 갖는 spanning tree로 보기 쉽게 바꾸어 보면 [그림 7]과 동일한 형태를 지닌다.



[그림 7] 그래프의 tree 형태.

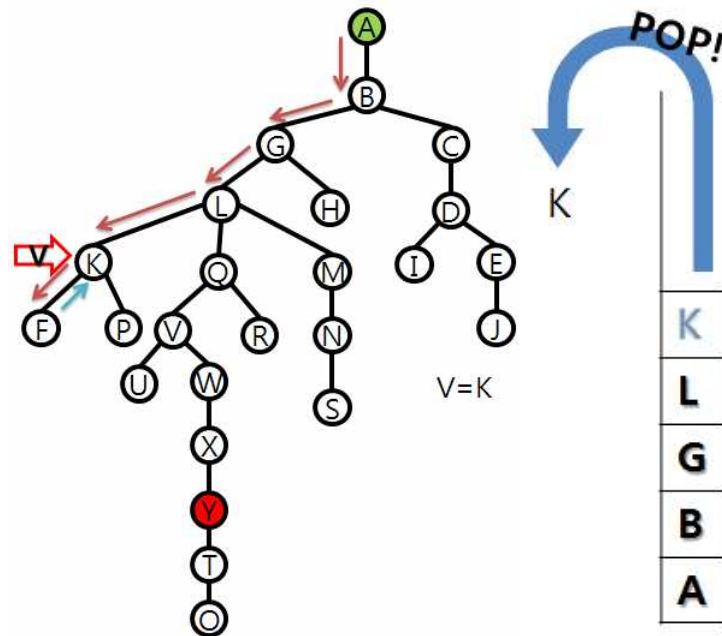
node A부터 시작하여 그래프로 변환된 미로를 DFS 탐색방법으로 계속 탐색하면서 출구인 node Y를 찾을 때, [그림 8]의 F처럼 방문한 적이 없었던 자식 node가 더 이상 존재하지 않는 vertex에 도달하게 되면



[그림 8] F : 더 이상 자식 node를 찾을 수 없는 node.

[그림 9]처럼 stack에 저장해둔 node를 pop시켜서 이전 node로 되돌아 갈 수 있을 것이

다. 자신이 구현한 미로의 자료구조를 어떻게 stack에 저장할 것인지 생각해보자.



[그림 9] Stack의 pop을 이용한 부모 node로의 backtrack.

3-4. GUI 구현

주어진 그래픽 툴킷으로 화면에 그림을 그리는 방법은 2주차 실험 교재에서 설명했으니 생략한다. 이번 프로젝트에서 그림으로 그려야 할 내용은 두 가지로 하나는 실제로 미로의 출발점과 도착점을 연결하는 경로이고, 다른 하나는 이 경로를 찾기 위해 DFS 알고리즘에서 탐색한 모든 경로들이다. 화면에 그림을 그리는 기능의 drawBuffered() 함수에서 미로를 탈출하는 경로와 이를 위해 탐색했던 경로들의 정보를 얻을 수 있도록, DFS를 수행하는 동안에 이 정보들을 적절한 자료구조에 저장하여야 한다. 이를 고려하면서 구현하도록 하자.

4. 숙제 및 보고서 작성

4-1 예비보고서

1. DFS와 BFS의 시간 복잡도를 계산하고 그 과정을 설명한다.
2. 자신이 구현한 자료구조 상에서 DFS와 BFS 방법으로 실제 경로를 어떻게 찾는지 설명한다. 특히 DFS 알고리즘을 iterative한 방법으로 구현하기 위한 방법을 생각해보고 제시한다.

4-2 숙제

BFS에 의한 미로 길 찾기 문제

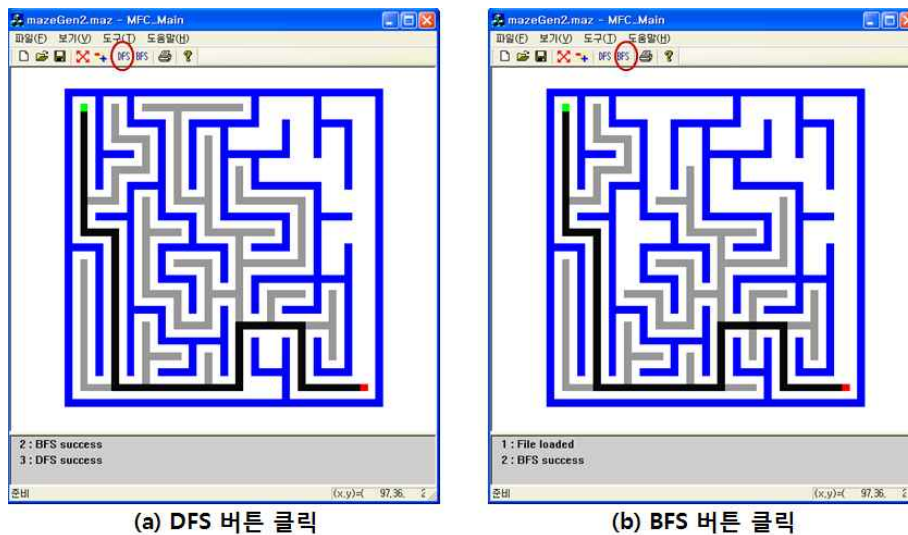
그래프 이론에서 그래프를 탐색하는 잘 알려진 방법에는 3주차 실험에서 진행했던 DFS 탐색 이외에도 BFS(Breadth First Search : 너비 우선 탐색) 방법이 존재한다. 2주차 및 3주차 예비보고서에서 조사한 내용 및 3주차 실습에서 완성한 프로그램을 바탕으로 BFS버튼을 누르면 BFS 방법을 통한 탈출 경로 및 탈출과정에서 방문했던 모든 경로를 표시하는 프로그램을 작성하자.

입력형식: 미로 텍스트 파일(~.maz)

제작환경: 3주차 실습결과인 DFS탐색이 가능한 MFC 응용 프로그램.


출력형식: 입력파일을 읽어 들인 후 활성화된 DFS, BFS 메뉴 및 버튼을 클릭하면 해당하는 방법으로의 탐색을 수행한다. 각각의 수행방법으로 찾은 탈출경로와 경로 탐색 수행 중 방문한 모든 길을 서로 구분이 갈 수 있도록 출력한다(그림 10 참조).

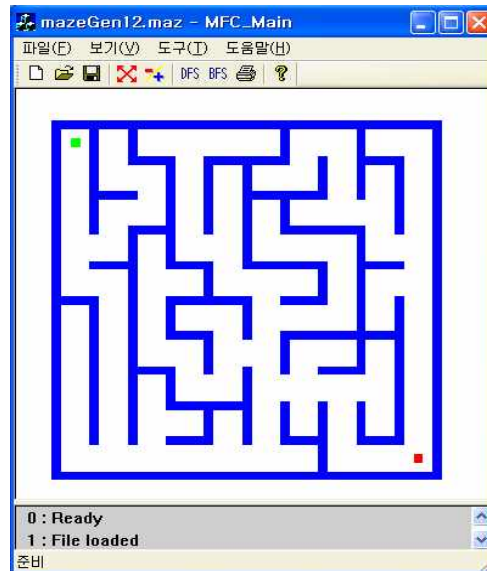
입출력 예:




[그림 10] DFS, BFS 버튼 클릭 시 출력.

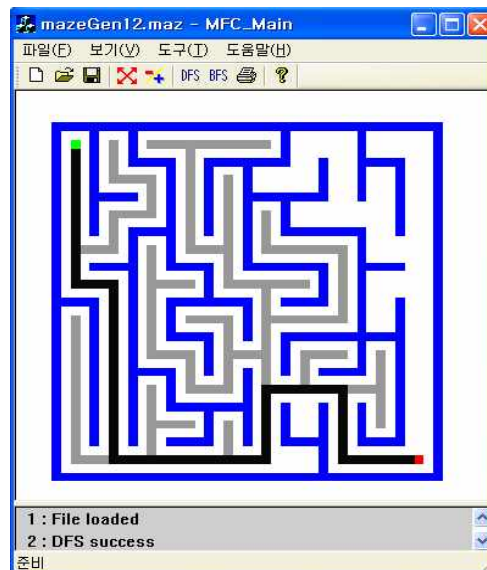
윈도우 설계 : 이번 실험을 통해 구현하려는 프로그램의 전체적인 화면은 위의 출력 예제가 된다. 이번 실험을 통해 만들 프로그램을 수행순서로 설명한다면 아래와 같다.

1. 2주차에서 만든 MFC 응용 프로그램에서 사용자가 메뉴의 “열기”나 툴바 내의 을 눌러 미로 텍스트 파일(~.maz)을 선택한 후 읽어 들이면 [그림 11]와 같이 빈 화면에 출발점과 도착점이 표시된 미로가 그려진다.




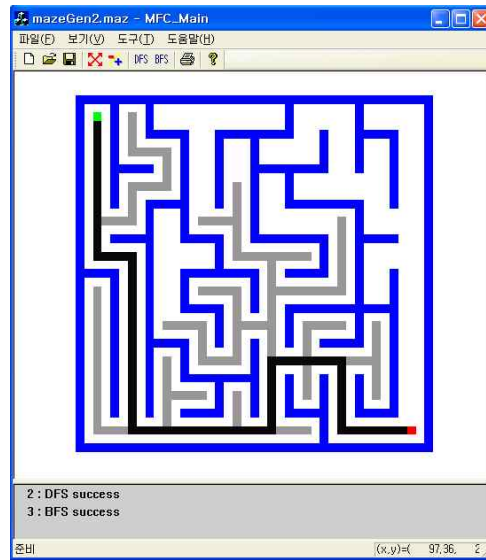
[그림 11] 미로 파일 읽어 들임.

2. 사용자가 [도구]-[DFS]메뉴, 툴바의 를 누르면 [그림 12]와 같이 화면에 DFS 방법으로 미로를 탐색한 결과가 그려진다(실습 시간에 미리 구현한 제작 환경).



[그림 12] DFS 버튼 클릭.

3. 사용자가 [도구]-[BFS]메뉴, 툴바의 를 누르면 [그림 13]과 같이 화면에 BFS를 방법으로 미로를 탐색한 결과가 그려진다(숙제로 구현해야할 내용이다).



[그림 13] BFS 버튼 클릭.

4. 종료는 윈도우의 오른쪽 상단의 X아이콘이나 아니면 파일메뉴의 끝을 누르면 프로그램이 종료된다.

문제해결 :

BFS 경로 탐색 및 탐색한 경로 저장

DFS 경로 탐색의 경우 도달하고자 하는 node(이번 실험에서는 도착점)에 도달하면 자동적으로 stack에 탈출경로가 저장된다. 반면 일반적인 BFS 경로 탐색의 경우 도착점에 도달하더라도 queue에 탈출경로가 저장되어있지는 않다. 적절한 자료구조(기존에 만들었던 자료구조를 재사용하거나 변형해도 좋고 새로운 자료구조를 만들어도 좋다)를 만들어서 탈출경로 및 방문했던 모든 경로들을 저장하도록 한다.

동일한 미로에서 DFS 수행 후 BFS, 혹은 BFS 수행 후 DFS

두 미로 탐색 방법 중 한 가지를 수행한 후 다른 방법으로 탈출경로를 재탐색 하고자 할 경우 이전 화면을 지우고 후에 선택한 방법으로 탐색한 결과를 다시 화면에 표시하여야 한다. 또한, 이전에 탐색했던 방법을 다시보기 원할 경우 탈출경로 및 탐색경로를 재탐색 하지 않고 저장된 탈출경로와 탐색경로를 표시만 하도록 해서 시간복잡도를 감소시킬 수 있다.

4-3 결과 보고서

1. 실습 및 숙제로 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술한다. 완성한 알고리즘의 시간 및 공간 복잡도를 보이고 실험 전에 생각한 방법과 어떻게 다른지 아울러 기술한다.
2. 자신이 설계한 프로그램을 실행하여 보고 DFS, BFS 알고리즘을 서로 비교한다. 각각의 알고리즘은 어떤 장단점을 가지고 있는지, 자신의 자료구조에는 어떤 알고리즘이 더 적합한지 등에 대해 관찰하고 설명한다.

PRJ-2(3주차/3주) 예비보고서

전공:

학년:

학번:

이름

PRJ-2(3주차/3주) 결과보고서

전공:

학년:

학번:

이름

참고문헌

- [1] *Maze Generation: Algorithm Recap*, Jamis Buck 지음,
<http://weblog.jamisbuck.org/2011/2/7/maze-generation-algorithm-recap>,
2012/1.
☞ 여러 가지 미로 생성 알고리즘이 자세하고 쉽게 설명되어 있다.
- [2] *Maze Generation and Solution*, Helen Dodd 지음,
<http://homepage.ntlworld.com/christopher.dodd/helend/stuff/Helen-Dodd-322805-Initial-Document.pdf>, 2012/1.
☞ 미로의 정의부터 미로 생성, 미로 길 찾기 등 미로에 관한 다양한 내용이 정리되어 있다.
여러 가지 미로 생성 알고리즘의 pseudo code와 각각의 장단점이 자세히 설명되어 있다.

