최종 프로젝트 – 미로

20180032 남기동

자료구조(그래프)

- typedef struct _node* node_pointer;
- typedef struct _node {
- 	int row;
- 	int col;
- 	int index;
- 	int middle_x;
- 	int middle_y;
- 	node_pointer link[4]; //0은 up, 1은 right, 2는 down, 3은 left를 의미한다
- }node;

자료구조(스택, 큐)

- 스택: DFS를 구현하기 위해 사용하는 자료구조
  - Push, pop을 포함

- 큐: BFS를 구현하기 위해 사용하는 자료구조
  - Addq, Deleteq를 포함

멤버 변수와 멤버 함수 개괄(1주차)

- void make_maze(): 아래 함수들을 포함하여 완전 미로를 만들고 maze.maz 파일 안에 저장
- void Init_maze(int temp_height, int temp_width, int** maze): 2차원 배열의 미로를 초기화
- void print_maze(int temp_height, int temp_width, int** maze): 디버깅 목적으로 미로의 결과물을 콘솔에 출력
- void fwrite_maze(int temp_height, int temp_width, int** maze): 미로의 결과물을 maze.maz 이름의 파일을 생성하고 저장
- void Eller_algorithm(int temp_height, int temp_width, int** maze): 엘러의 알고리즘으로 완전 미로를 생성하는 알고리즘

멤버 변수와 멤버 함수 개괄(2주차)

- bool readFile(): maz확장자 파일을 읽어서 미로의 정보를 저장해오는 역할을 하는 함수
- void freeMemory(): 동적 할당했던 메모리들을 해제하는 역할을 하는 함수
- int** maze: 파일의 미로 정보를 담는 이차원 배열
- int HEIGHT: 미로의 높이
- int WIDTH: 미로의 너비
- node_pointer* graph: graph[MAX_VERTICES]로 동적 할당된다
- int height_graph: 미로 방 기준, 미로의 높이
- int width_graph: 미로 방 기준, 미로의 너비
- int num_of_terms: 미로 방의 개수

멤버 변수와 멤버 함수 개괄(3주차 - DFS)

- bool DFS(): DFS 경로 탐색을 활용하여 도착점까지의 너비 우선 탐색의 결과를 도출하는 함수
- void dfsdraw(): DFS가 성공적으로 완료되어 경로가 나왔을 때 이를 화면에 그리는 함수
- int* visited: 각 노드(미로의 방)에 방문했는지를 표시하기 위한 일차원 배열
- int* stack: DFS구현을 위한 자료구조로 num_of_terms만큼 동적 할당해서 경로 저장 용도로 활용
- int top: stack의 구현을 위해 필요한 변수
- int* all: DFS함수에서 방문했던 모든 경로를 저장하기 위한 용도의 일차원 배열
- int top_all: all을 stack으로 구현할 때 이용하는 변수
- void push(int current_pos): stack의 기본 기능으로 현재 위치를 입력 받아 스택에 넣는다
- int pop(): stack의 기본 기능으로 스택의 top에 해당하는 값을 반환한다

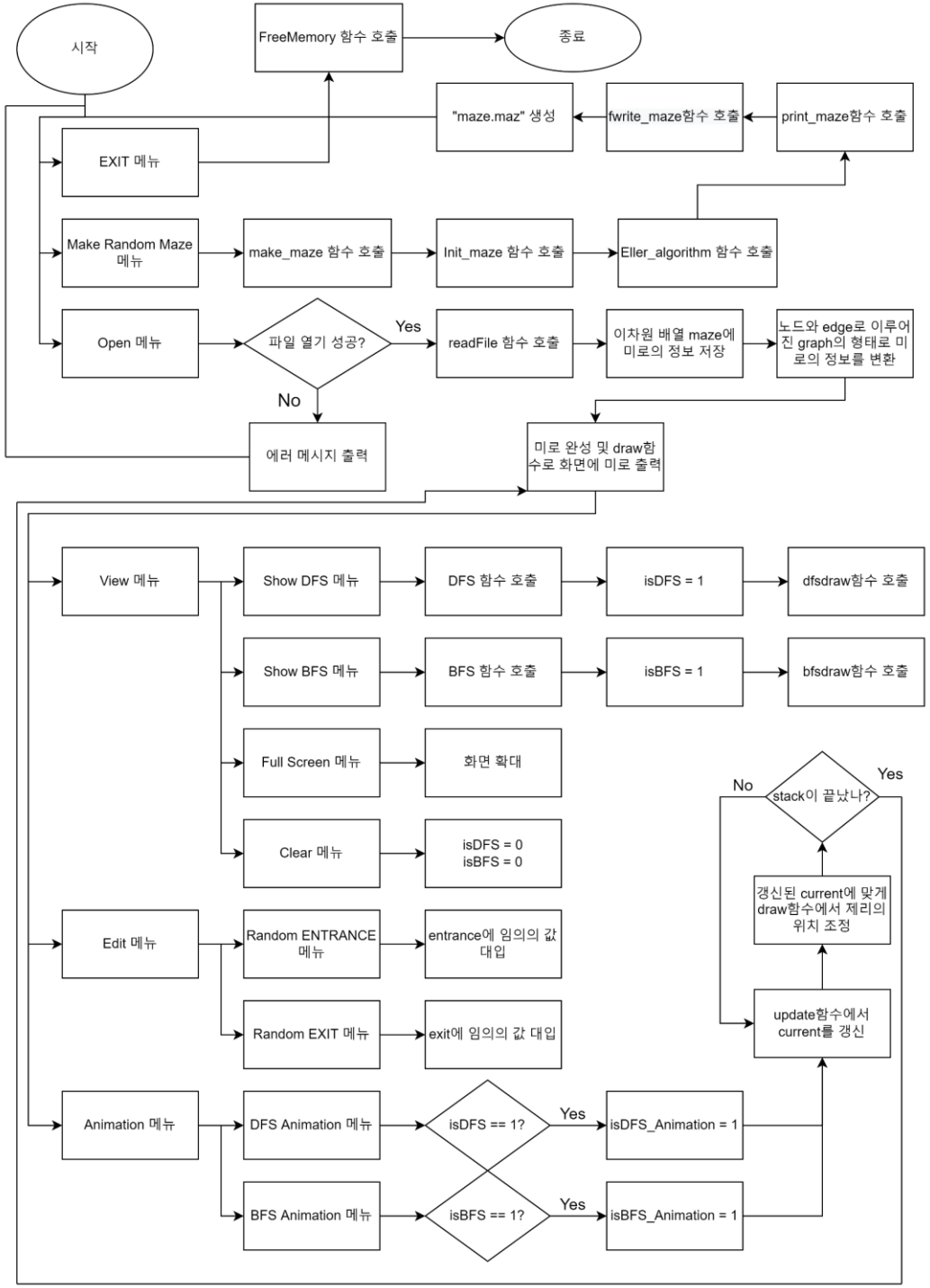
멤버 변수와 멤버 함수 개괄(3주차 - BFS)

- bool BFS(): BFS 경로 탐색을 활용하여 도착점까지의 너비 우선 탐색의 결과를 도출하는 함수
- void bfsdraw(): BFS가 성공적으로 완료되어 경로가 나왔을 때 이를 화면에 그리는 함수
- int* queue: BFS를 구현하기 위해 사용하는 queue 자료구조
- void addq(int current_pos): 현재 위치를 입력으로 받아서 rear을 증가시켜 queue에 하나 더한다
- int deleteq(): queue의 기본적인 기능 중 하나로 front를 증가시켜 queue를 하나 제거한다
- int front, rear: queue의 구현을 위해 필요한 front와 rear
- int rear_all: all에 queue로 저장하기 위해 사용하는 rear 역할의 변수

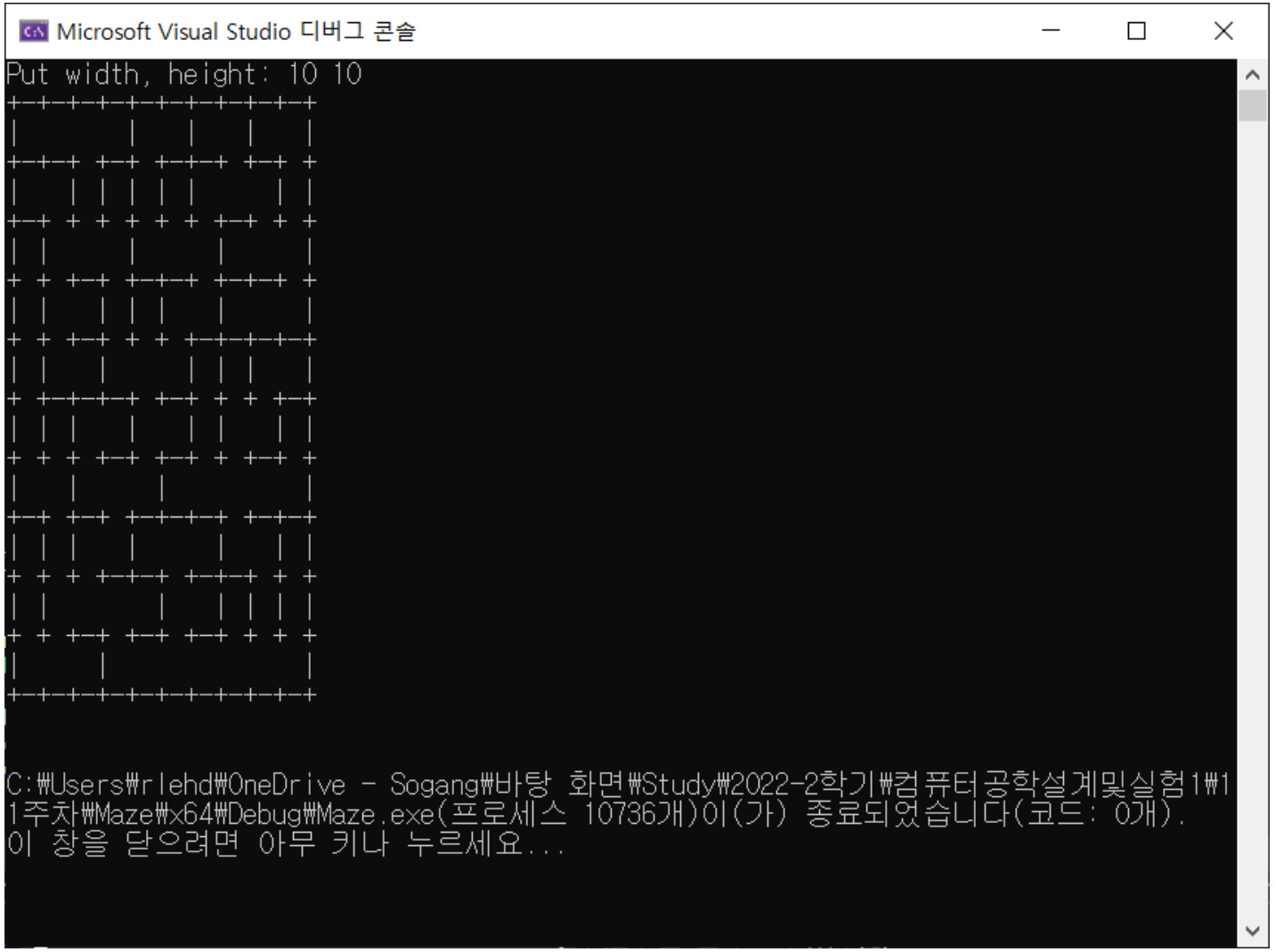
멤버 변수와 멤버 함수 개괄(그 외)

- int isOpen: 파일이 열렸는지를 판단하는 변수. 0이면 안 열렸고 1이면 열렸다.
- int isDFS: DFS함수를 실행시켰는지 판단하는 변수. 0이면 실행 안 했고 1이면 실행했다.
- int isBFS: BFS함수를 실행시켰는지 판단하는 변수. 0이면 실행 안 했고 1이면 실행했다.
- ofImage Jerry: 제리의 이미지를 저장하는 변수
- ofImage Cheese: 치즈의 이미지를 저장하는 변수
- int entrance: 미로의 시작점을 의미
- int exit: 미로의 도착점을 의미
- int current_top: stack의 top과 같은 역할을 하는 변수
- int current: stack[current_top]을 대입해서 현재 위치해야 하는 미로의 방을 가리키는 index
- int isDFS_Animation: 1이면 DFS_Animation을 실행시키도록 하는 flag
- int isBFS_Animation: 1이면 BFS_Animation을 실행시키도록 하는 flag

플로우 차트


시작
FreeMemory 함수 호출
종료
"maze.maz" 생성
fwrite_maze함수 호출
print_maze함수 호출
EXIT 메뉴
Make Random Maze 메뉴
make_maze 함수 호출
Init_maze 함수 호출
Eller_algorithm 함수 호출
Open 메뉴
파일 열기 성공?
Yes
No
readFile 함수 호출
이차원 배열 maze에 미로의 정보 저장
노드와 edge로 이루어진 graph의 형태로 미로의 정보를 변환
에러 메시지 출력
미로 완성 및 draw함수로 화면에 미로 출력
View 메뉴
Show DFS 메뉴
DFS 함수 호출
isDFS = 1
dfsdraw함수 호출
Show BFS 메뉴
BFS 함수 호출
isBFS = 1
bfsdraw함수 호출
Full Screen 메뉴
화면 확대
Clear 메뉴
isDFS = 0
isBFS = 0
Edit 메뉴
Random ENTRANCE 메뉴
entrance에 임의의 값 대입
Random EXIT 메뉴
exit에 임의의 값 대입
Animation 메뉴
DFS Animation 메뉴
isDFS == 1?
Yes
isDFS_Animation = 1
BFS Animation 메뉴
isBFS == 1?
Yes
isBFS_Animation = 1
No
stack이 끝났나?
Yes
갱신된 current에 맞게 draw함수에서 제리의 위치 조정
update함수에서 current를 갱신

1주차 실습



엘러의 알고리즘을 이용하여
완전 미로를 생성

2주차 실습(1. readFile)

- 1. int** maze에 파일에서 읽은 미로의 정보를 저장
  - 모서리(+)는 1, 가로벽(-)은 2, 세로벽(|)은 3, 미로의 방( )은 0으로 간주

- 2. 그래프 자료구조를 활용하여 maze의 정보를 node들의 그래프로 변환
  - Link[0]은 위쪽, link[1]은 오른쪽, link[2]은 아래쪽, link[3]은 왼쪽
  - 열려 있는 방향에 해당하는 노드와 edge를 만들어준다

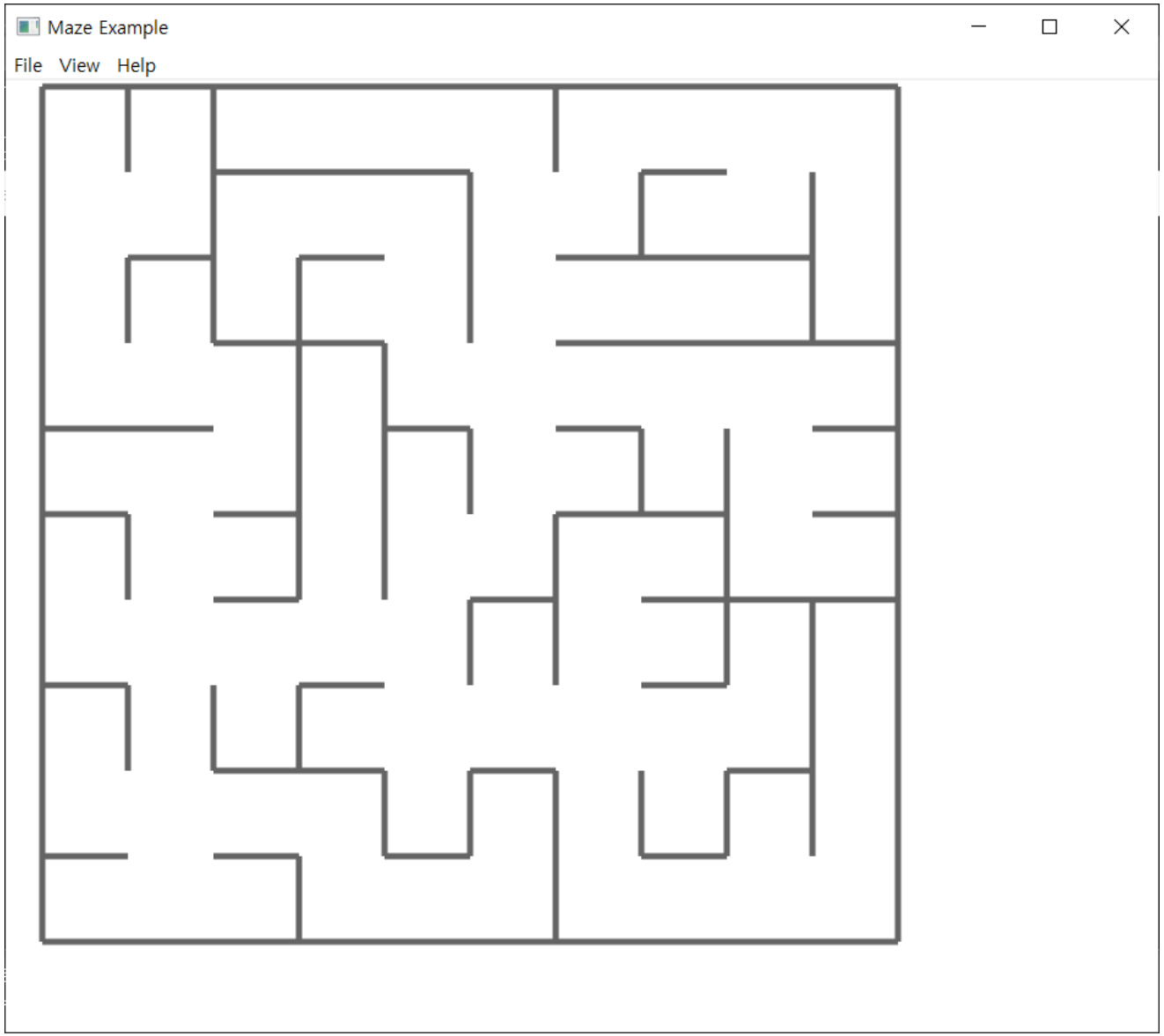
- 시간 복잡도, 공간 복잡도: O(height * width)

2주차 실습(2. freeMemory)

- 1. maze와 graph 메모리 해제
- 2. isDFS가 1이라면
  stack, visited, all을 메모리 해제
- 3. isBFS가 1이라면
  queue, stack, visited, all을 메모리 해제

2주차 실습(3. draw)

- X, y의 값을 기준이 되는 점의 꼭짓점으로 설정
- If(isOpen)인 경우에만 미로를 그리도록 설정
- 연결된 노드가 없는 경우, ofDrawLine을 사용해서 벽을 그린다
- Node의 인덱스를 1개씩 더해가면서 행, 열을 이동하며 그린다

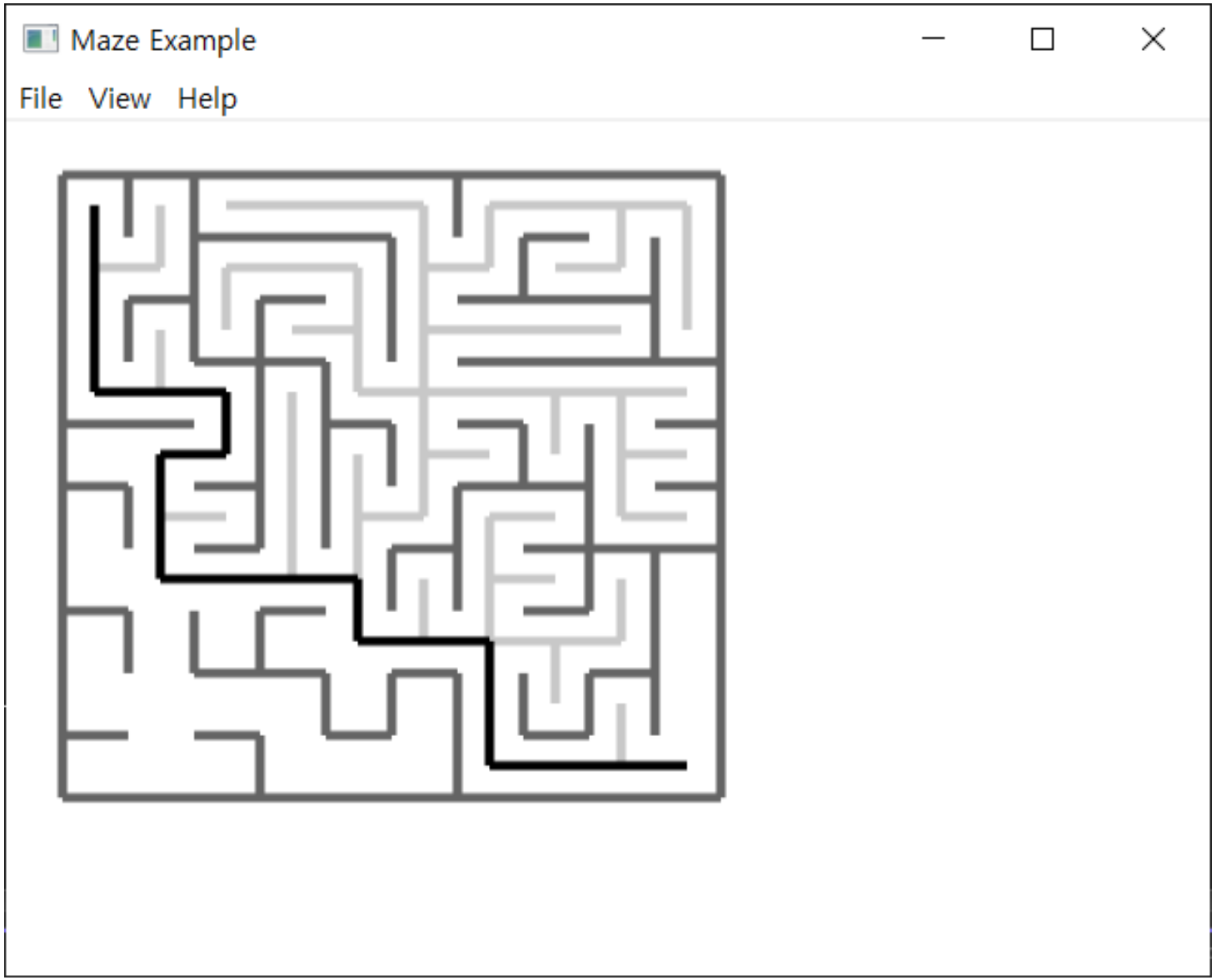

Maze Example
File View Help

3주차 실습(DFS)

- 시작점에서 출발하여 가능한 방향으로 계속 나아가다가 더 이상 전진할 수 없는 경우 이전 위치로 돌아와서 다시 갈 수 있는 방향을 찾아 이동(깊이 우선 탐색)

- Stack을 이용해서 구현(이동하면 push, 돌아올 때는 pop)

- 시간, 공간 복잡도 : O(v+e)

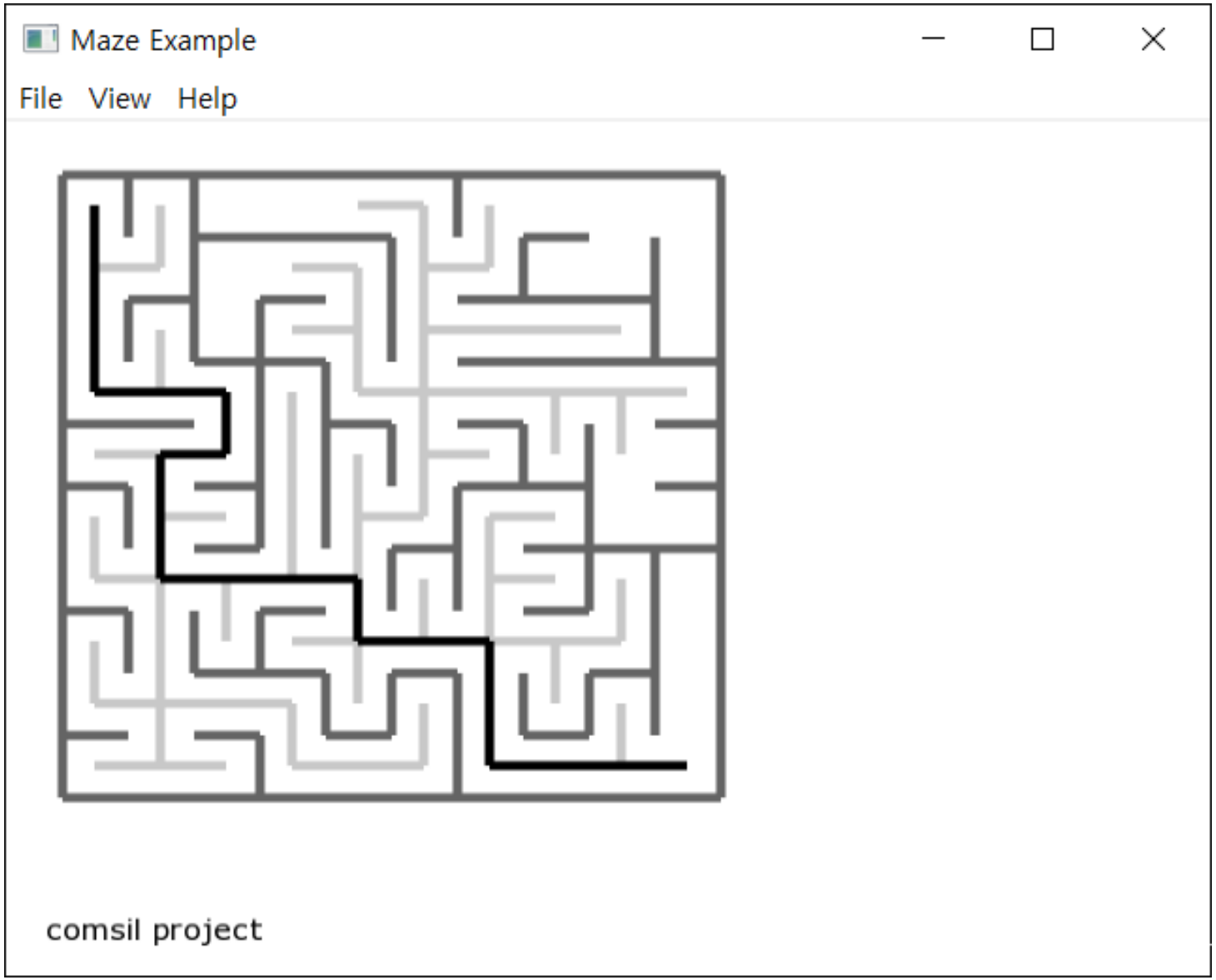

Maze Example
File
View
Help

3주차 실습(BFS)

- 시작점에서 출발하여 동일한 거리의 지점들을 방문하는 방식으로 도착점을 찾는 방식이다(너비 우선 탐색)

- Queue를 이용해서 구현(이동하면 addq, 다음 위치는 deleteq)

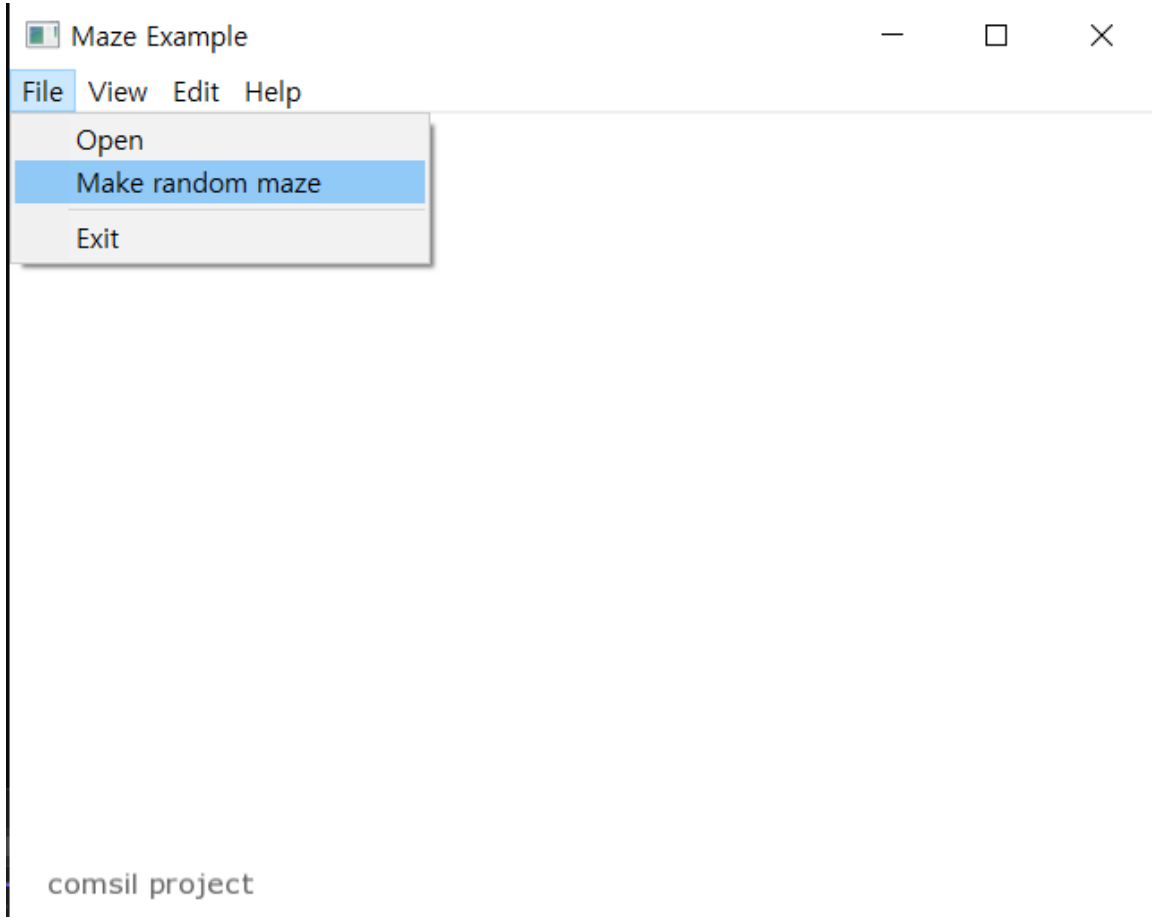
- 시간, 공간 복잡도 : O(v+e)


Maze Example
File View Help
comsil project

새롭게 추가하거나 변경한 사항

- (1) make random maze 세부 메뉴 만들기
- (2) make_maze( )함수를 통해 HEIGHT와 WIDTH를 입력 받아서 새로운 미로 파일을 생성
- (3) 종착점에는 치즈를, 시작점에는 제리의 모습을 그려서 표시
- (4) 미로의 시작점과 끝나는 점을 랜덤하게 배치
- (5) DFS, BFS 이후에 Clear 기능 추가(미로만 남기기)
- (6) DFS, BFS 애니메이션 생성

make random maze 세부 메뉴 만들기

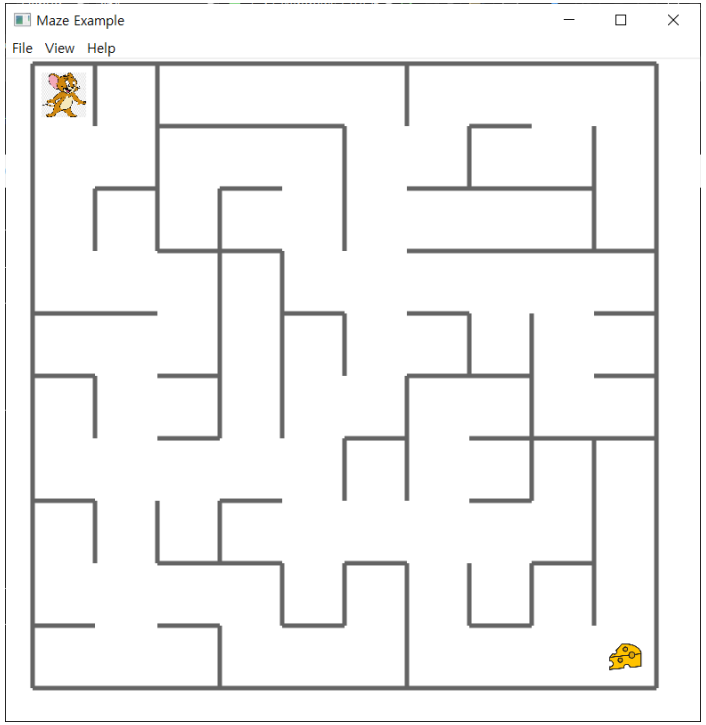


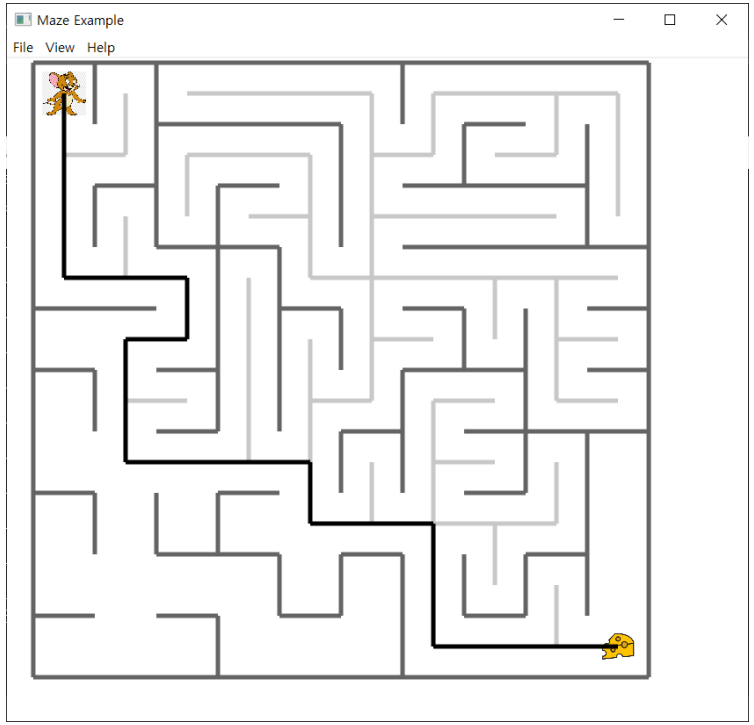
make_maze( )함수로 새로운 미로 파일을 생성

- void make_maze()
- void Init_maze(int temp_height, int temp_width, int** maze)
- void print_maze(int temp_height, int temp_width, int** maze)
- void fwrite_maze(int temp_height, int temp_width, int** maze)
- void Eller_algorithm(int temp_height, int temp_width, int** maze)

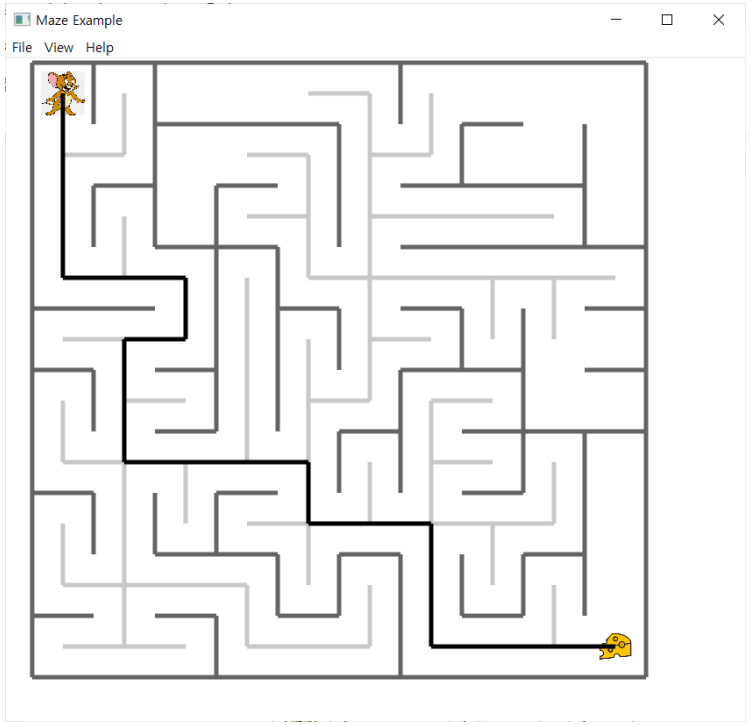
- 시간 복잡도: O(n^2)
- 공간 복잡도: O(n)

시작점에는 제리를, 도착점에는 치즈의 이미지 그리기



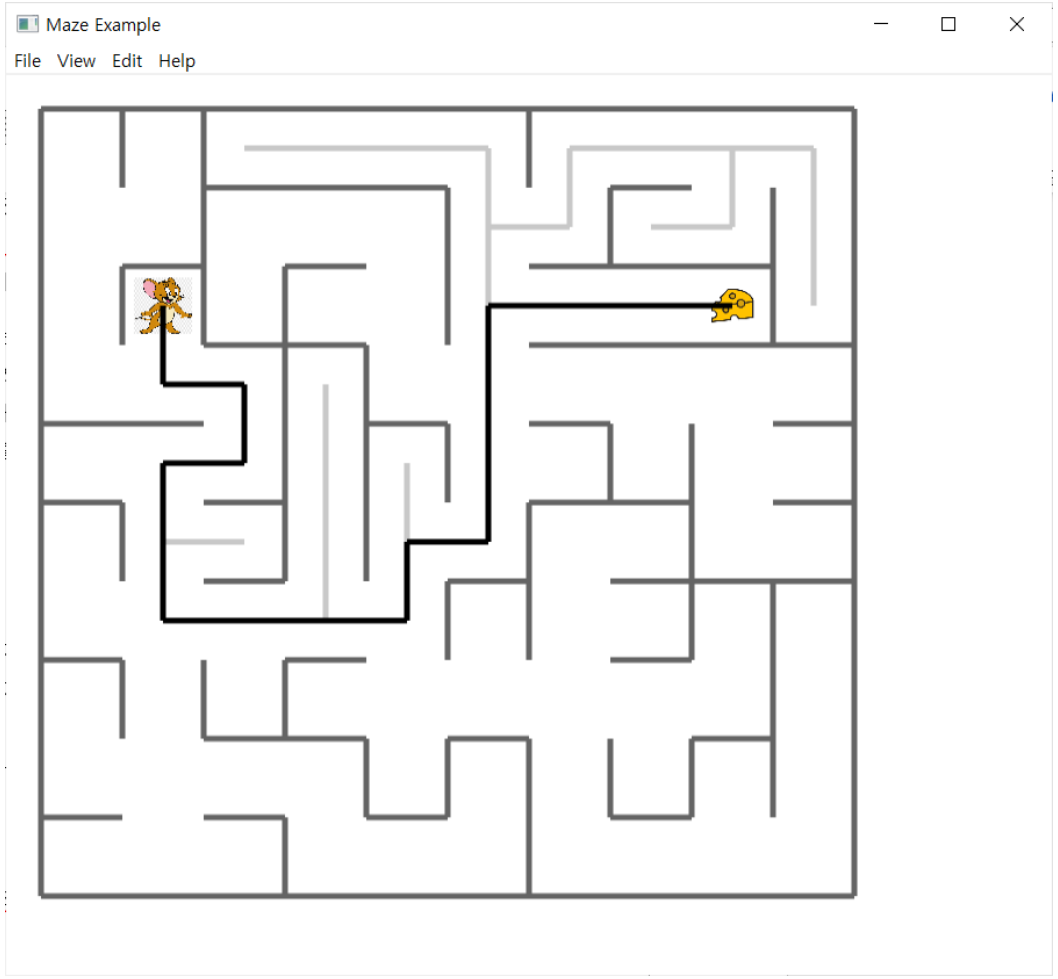


DFS

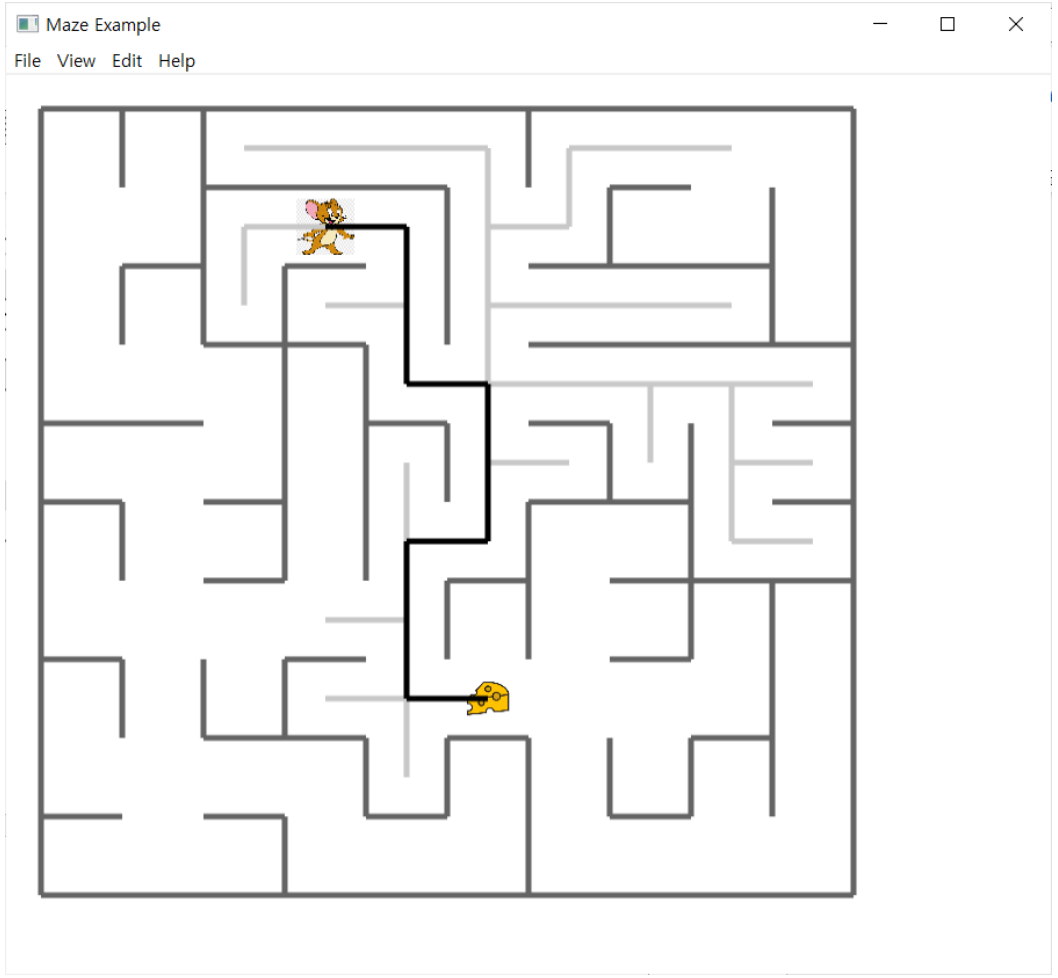


BFS

미로의 시작점과 도착점을 랜덤하게 배치

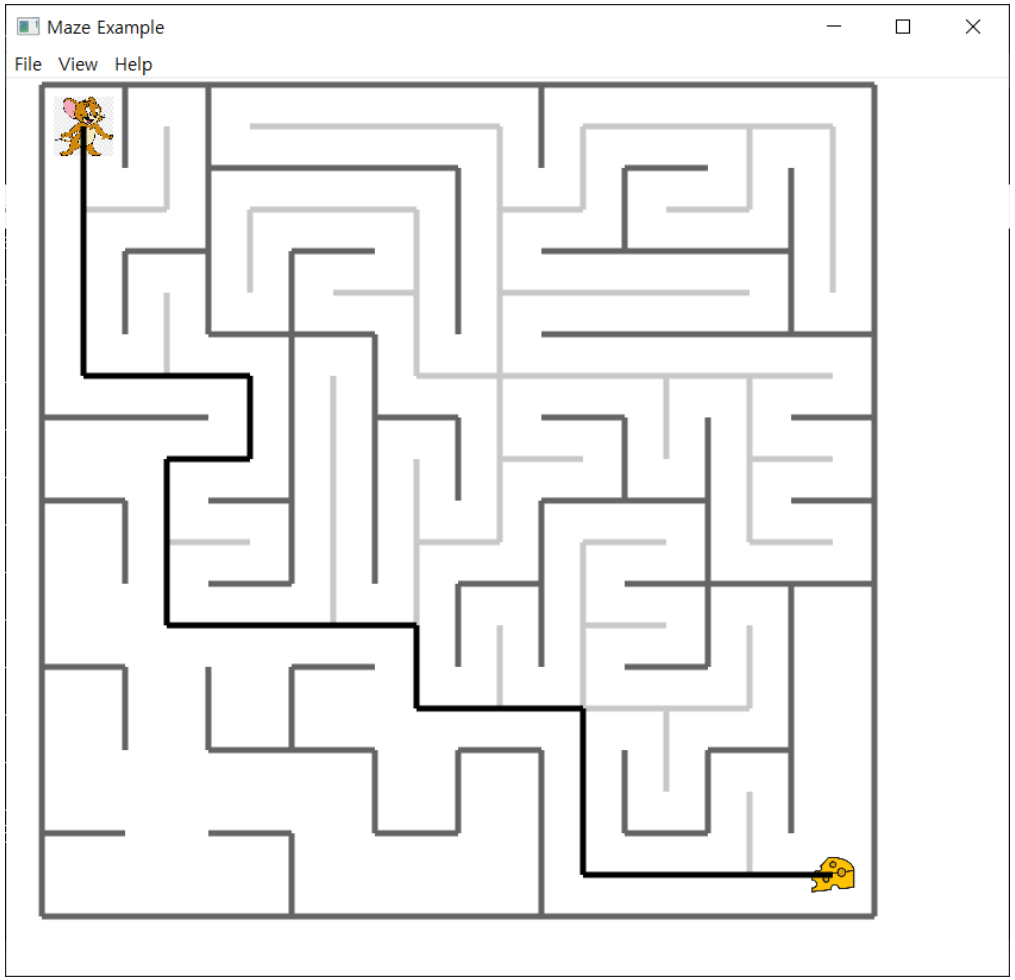


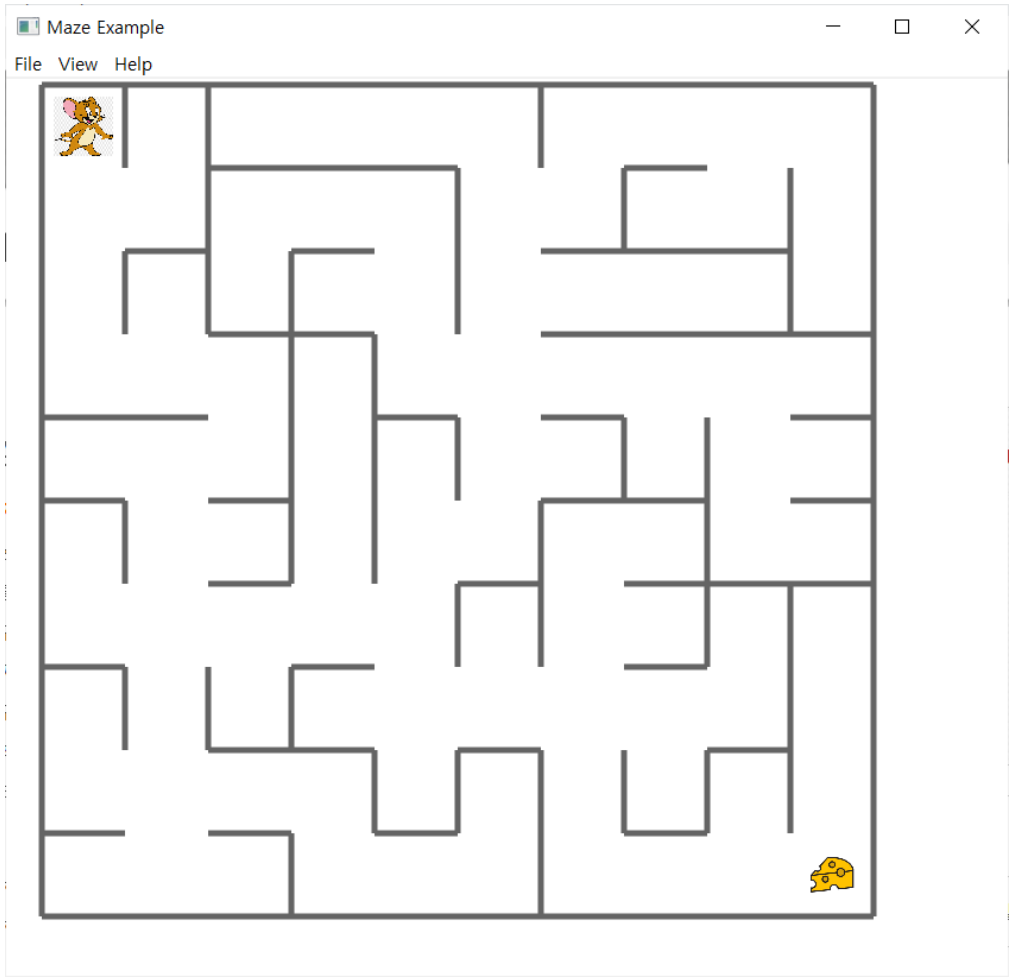
DFS



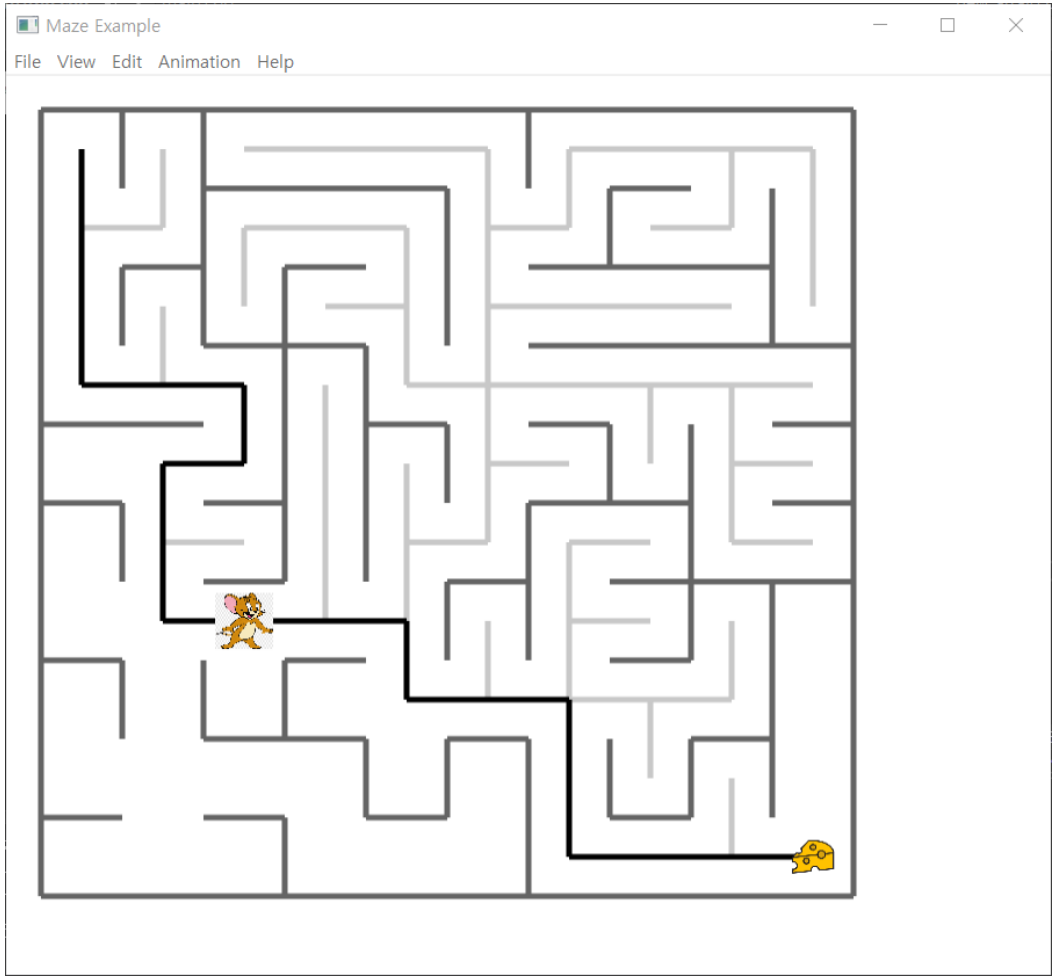
BFS

Clear 기능 추가(미로만 남기기)

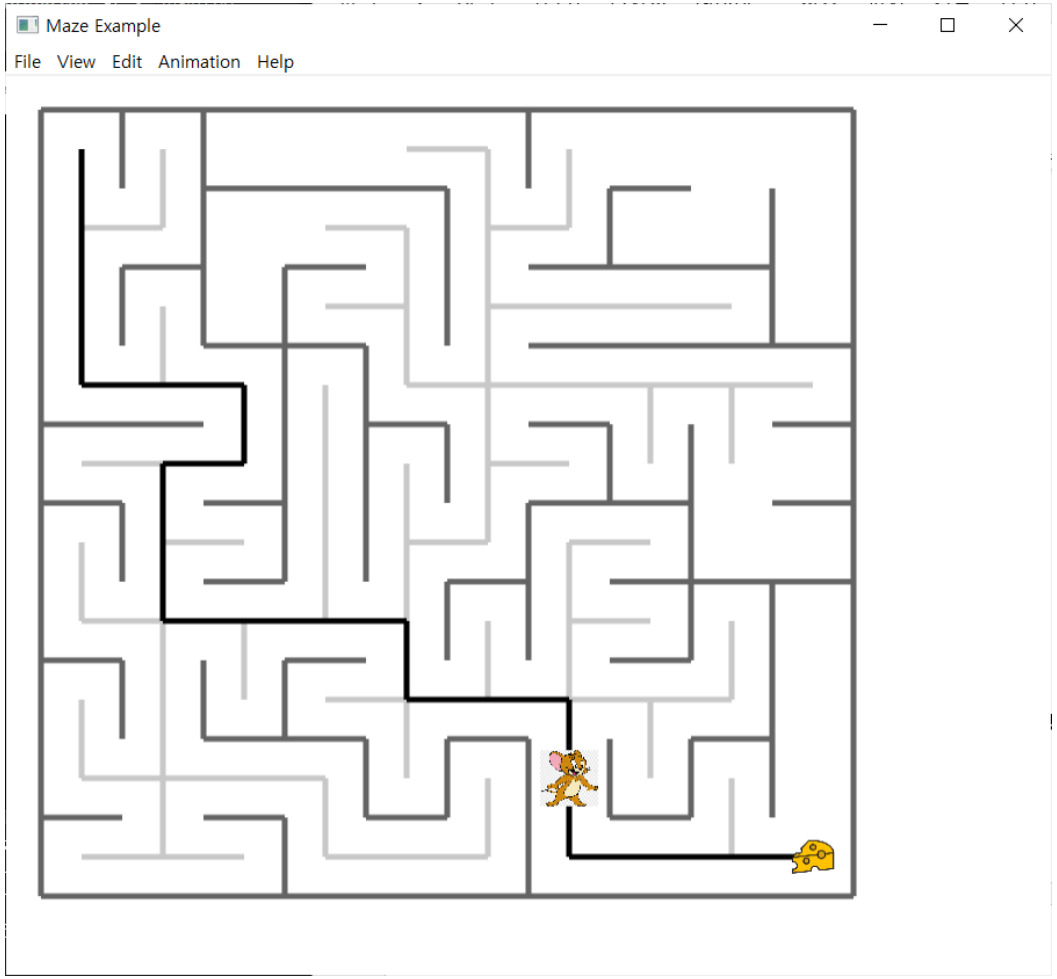




DFS, BFS 애니메이션 생성



DFS



BFS

실행 동영상

실행 동영상