Final report

전공: 컴퓨터공학 학년: 2학년 학번: 20221559 이름: 박준우

1. 프로젝트 목표, 실험 환경에 대한 설명

메이즈 실습을 진행함에 있어 나에게 다가온 불편함이 몇 가지가 있었고, 이를 해결하고보고자 하였다.

첫 번째로는 우선 나는 맥을 주로 사용하는데, 2, 3주차 dfs, bfs 구현에서 dfs, bfs 모드를 켜고 끄기 위해서 winlibs 라이브러리를 사용하는 스켈레톤 코드를 제공받아, 윈도우 환경에서만 실험이 가능했다는 점이었다. 이를 해결하기 위해 가능하면 맥에서 직접 구현해서 진행해도 된다고 하였는데, 이것은 실습 진행 시점상 시간이 부족해 진행하지 못하였다. 그래서 기말 프로젝트에서 해당 기능을 구현한 스켈레톤 코드를 직접 만들어 후배들이 사용할 수 있게 된다면 좋겠다고 생각했다.

두 번째는 1주차에 진행했던 메이즈를 만든 후 이것을 출력한 결과물은 그것 자체로 보기가 상당히 불편했다. 그리고 이것을 openframeworks에서 불러와서 사용하기밖에 하지 못했다는 점이다. 이에 착안하여 openframeworks 상에서 메이즈를 만들고, 이것을 바로 활용하고, 이를 또 파일로 출력할 수 있다면 편할 것 같다고 생각했다.

실험 환경으로는 macOS Sequoia 15.3.2, Xcode Version 16.4(16F6), openFrameWorks v0.12.1\_osx\_release를 이용했다.

2. 각 변수에 대한 설명

Int HEIGHT, WIDTH;

미로의 높이, 너비를 가지고 있는 변수이다. 미로를 읽거나, 생성할 때 정해진다. 미로가 없다면 쓰레기값을 갖고 있다.

Float defaultOffsetX, defaultOffsetY, blockWidth;

미로의 각 칸을 정사각형으로 그렸는데, 미로의 왼쪽 끝 시작 좌표의 x, y값과, 이 정사각형의 한 변의 길이를 갖고 있다. 윈도우 크기와 미로 사이즈에 의해 자동으로 결정되고, 창의 크기가 변함에 따라 조정된다.

Isdfs, isbfs, mazestate;

Dfs, bfs 경로가 켜져있는지의 여부, 미로가 불러와지거나 생성되어져 있는지에 대한 여부를 나타낸다.

ofxPanel gui;

ofx에서 제공하는 메뉴바? 같은 클래스이다.

ofxButton ~~~~~ 각 기능을 하는 버튼들이다.

ofTrueTypeFont 화면상에 글자를 적어주기 위해 가져온 클래스이다.

해당 폰트 파일이 필요하다.

Vector<vector<char>> maze

미로 각 칸의 상하좌우 벽이 있는지를 가지고 있는 2차원 벡터이다.

Vector<vector<int>> visited

미로 각 칸을 방문하였는지의 여부를 가지고 있는 2차원 벡터이다.

Bfs 시에는 출발점부터의 거리를 나타내는데에도 쓰인다.

Vector<pair<pii, pii>> everyRoute

탐색 과정에서 쓰였던 모든 경로(edge)를 담고 있는 컨테이너이다.

Vector<pii> Route

탐색 과정에서 찾은 최단거리의 경로(vertex)를 담고 있는 컨테이너이다.

Float windowWidth, windowHeight;

윈도우 크기를 담고 있는 변수이다. 바뀌면 같이 업데이트된다.

3. 각 함수에 대한 설명

Setup() 각 변수를 초기화해준다.

Draw() 매 프레임에 윈도우에 그려주는 함수이다.

미로가 있는지, 탐색이 켜져있는지 확인해서 켜져있으면 그려준다.

Windowresized() 윈도우 크기가 변하면 미로 칸 크기를 바꿔준다.

dfsPressed(), bfsPressed() dfs, bfs 버튼이 눌렸을 때의 동작을 정의하고 있다.

이미 해당 탐색이 켜져있으면 끄고 지우고, 다른게 켜져있으면 그 경로를 없앤 뒤 탐색한다.

Fileopenpressed() fileopen 버튼이 눌렸을 때의 동작을 정의하고 있다.

무슨 파일을 입력받을 것인지를 입력받고, 해당 파일이 미로 파일인지의 여부를 검사한 후,

미로 변수에 파일을 읽어 값을 넣어준다.

Filesavepressed() filesave 버튼이 눌렸을 때의 동작을 정의하고 있다.

Maze가 없으면 저장할 게 없으므로 무시한다.

있다면 현재 시간을 파일명에 포함해 미로 파일로 형식에 맞춰 저장한다.

Mazegenpressed() mazegen 버튼이 눌렸을 때의 동작을 정의하고 있다.

미로가 이미 있다면 해당 미로를 지우고,

누르기 전 설정되어 있는 생성하고자 하는 미로의 크기를 가져와서, 미로 생성 함수를 호출한다.

Customexit() exit 버튼이 눌리면 로그에 종료한다 적고 프로그램을 종료한다.

processOpenFileSelection()

파일이 열렸을 때 실행되는 함수이다. 파일을 열어 내용에 따라 미로 자료구조를 만들어준다.

getRandBool() 말 그대로이다. Eller’s algorithm에 사용하기 위해 만들었다.

createMaze() eller’s algorithm을 이용해 미로를 만든다.

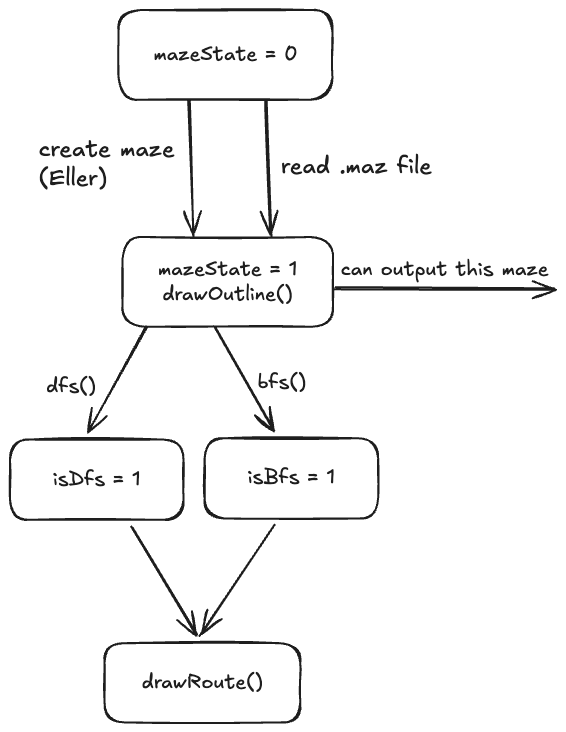
Drawmazeoutline() 미로가 있는 경우에 미로의 외벽을 그려준다.

Clear() 미로의 모든 요소 (벽 존재 여부, 방문점들, 등등)을 삭제(초기화)한다.

drawRoute() dfs나 bfs로 찾은 경로를 그려준다. 최단거리와 이용했던 경로의 색깔이 다르다.

Dfs(), bfs() 각각의 알고리즘으로 길을 찾아준다.

4. 프로젝트 전체 플로우차트 + 자료구조 및 알고리즘 + 시간/공간 복잡도 상세히 설명



자료구조는 미로 자체가 격자 그래프 형태이기 때문에 2차원 배열의 형태를 띄고 있으면 된다고 생각했다. 따라서 미로의 상하좌우 벽이 있는가를 bitmask로 가지고 있는 2차원 벡터를 만들어 주었다. 그리고 dfs/bfs의 결과를 저장하는 것도 c++의 vector 컨테이너를 사용해도 모든 원소가 이어져 있거나, 순서가 상관 없기 때문에 괜찮았다. Dfs에서는 재귀를 사용하지 않고, 스택을 이용하여 구현하였고, bfs는 queue를 이용하여 구현하였다.

Dfs 알고리즘은 방문여부를 체크하는 2차원 벡터를 이용하여 한번 방문한 정점은 다시 방문하지 않도록 했기 때문에, 메인 반복문의 반복 횟수는 미로의 크기에 상한이 있음을 알 수 있고, 최대 스택에 쌓일 수 있는 원소 개수도 마찬가지로 미로의 크기에 상한이 있다. 따라서 시, 공간 복잡도는 모두 O(WIDTH\*HEIGHT = maze size)임을 설명하였다.

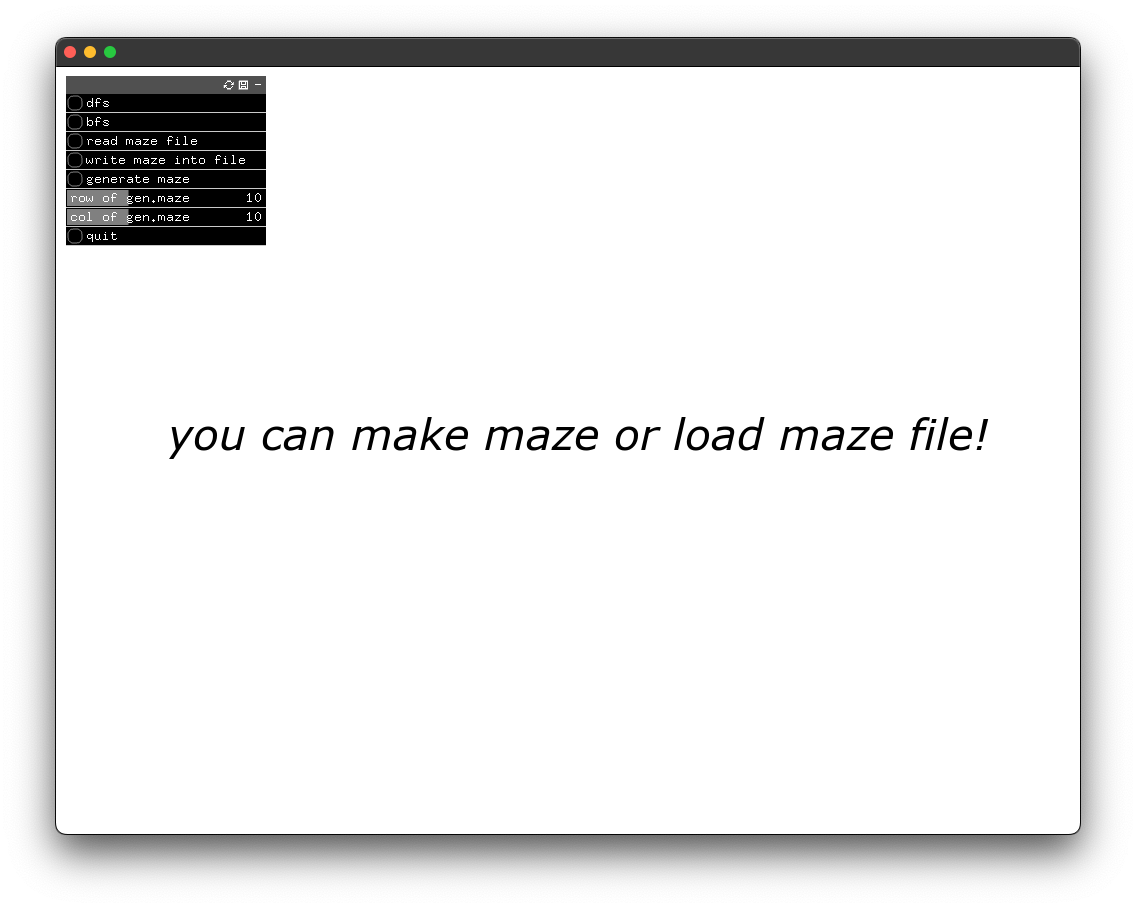
Bfs 알고리즘 또한 동일하나, 다시 추적하는 과정 또한 미로의 크기에 상한이 있으므로, 상수 배로 더 걸릴 수 있으나, 이또한 O(maze size)임을 설명할 수 있다.

또한 eller’s algorithm 에서도 각 줄에 대해서 처리해 줄 때 각 줄에 속한 원소의 개수에 반복문의 횟수가 제한됨을 설명할 수 있다. 이에 따라 이것의 시간, 공간 복잡도 또한 O(maze size)가 된다.

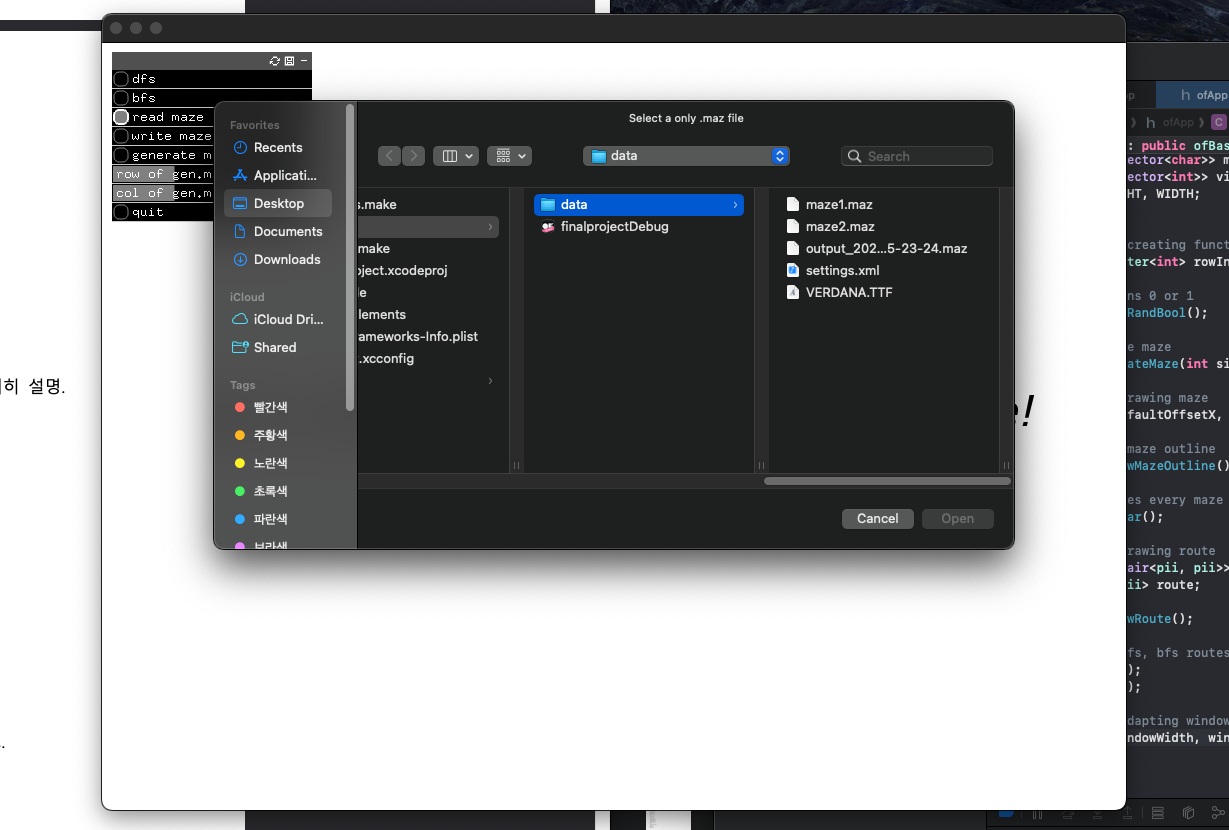
5. 창의적 구현에 대한 설명

하단 6번의 첫번째 시작 화면 왼쪽 상단의 메뉴바를 보면, 생성하고자 하는 미로의 가로, 세로 크기를 드래그로 조정 가능하게 하여 마우스로만 조작 가능한 프로그램을 만든 것이 창의적이었던 것 같다.

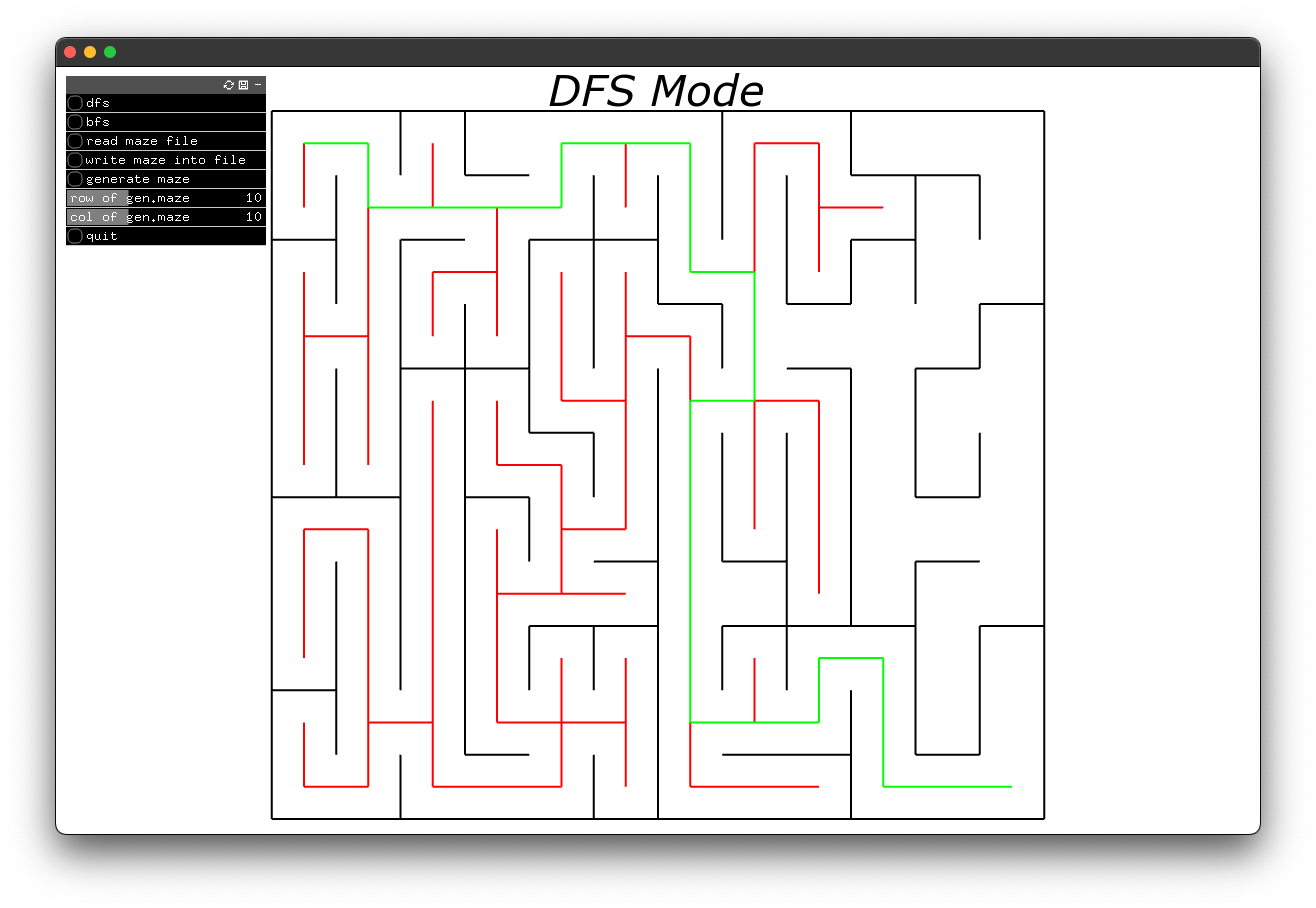
6. 시작 화면



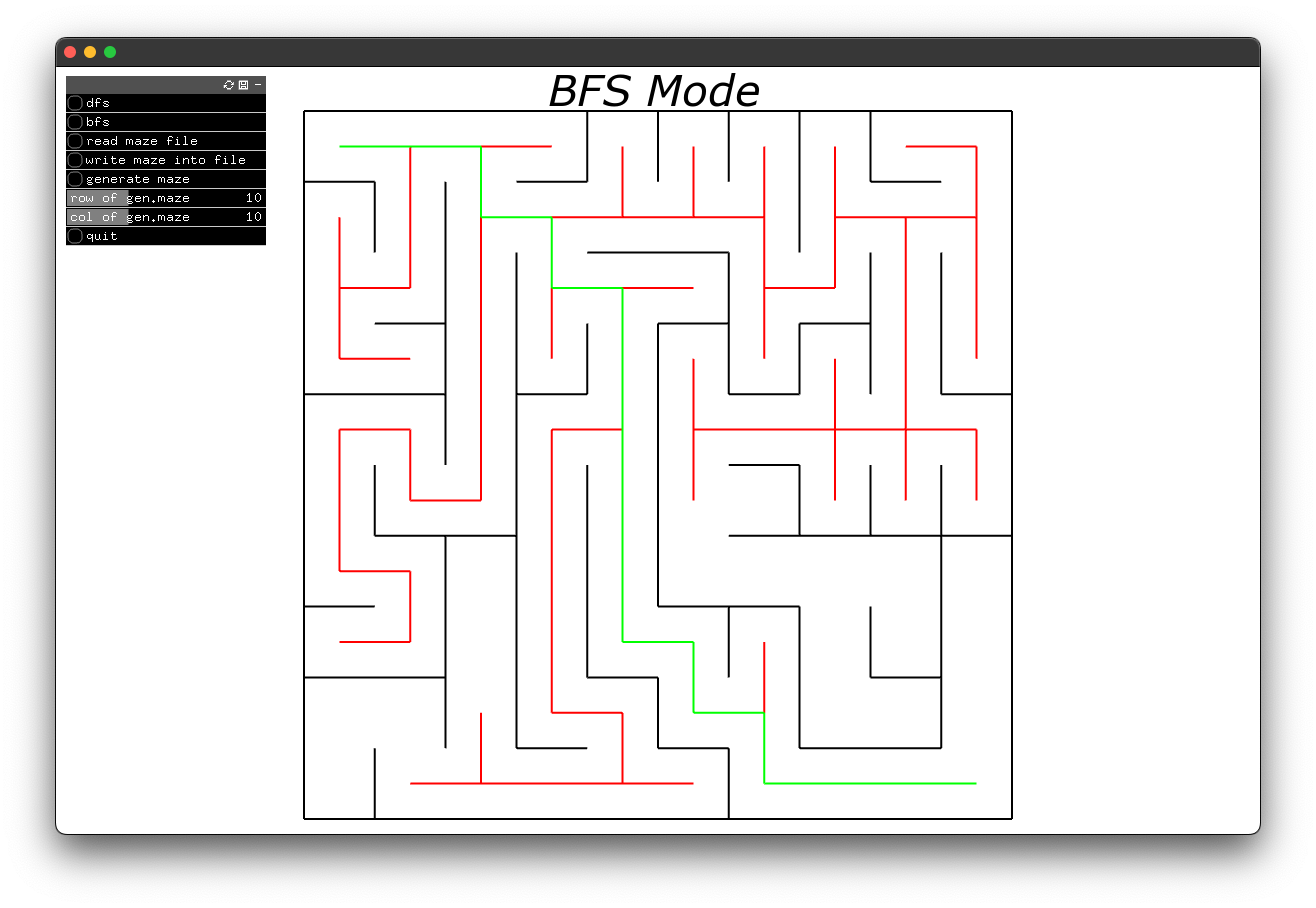
입력 파일 선택을 골랐을 때 뜨는 하는 창 화면



입력 파일을 읽어와서, dfs 버튼을 1회 누른 화면



크기를 10x10으로 설정 후 bfs 버튼을 1회 누른 화면



사진이 너무 많아져 저장 등의 실행결과는 발표 동영상에 포함된 실행 영상에 포함하겠음.

7. 느낀 점 및 개선사항

오픈프레임웍스를 사용해 보면서 잘 추상화 되어있는 클래스는 사용하기 정말 편리하다는 것을 느낄 수 있었다. 이를 통해 오픈소스 개발에 흥미를 느끼게 된 것 같고, c++으로 생각한 것을 구현해 냈다는 성취감을 느낄 수 있었고, 나의 실력에 자신감을 가지게 되었다.

개선사항은 없다.