12주차 예비보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

1. <ReadFile 함수>

ReadfIle 함수는 1주차 실습에서 생성한 .maz를 파일로 불러와서, 내부적인 변수로 저장하는 과정을 수행하는 함수이다. 각 줄 마다 어떠한 역할을 수행하는 지 설명해보고자 한다. 설명은 주석의 형태로 달아주었다.

// 파일을 불러오는 부분으로 ofileDialogResult 클래스를 openFileResult라는 이름으로 선// 언하여, openSystemLoadDialog을 통해 파일 불러오기 창을 “Select .maz file” 라는 문// 구가 title 되게하여 띄우고, 파일을 불러온다.

ofFileDialogResult openFileResult = ofSystemLoadDialog(“Select .maz file”);

string filePath; // 파일 경로를 담을 문자열 변수를 선언한다.

Size\_t pos; // 확장자를 거르기 위한 과정으로, 확장자 직전의 “.”의 위치를 저장한다.

// 위의 줄에서 파일을 선택하여서 파일 불러오기를 성공했을 때만 수행하기 위해서 조건// 문으로 작성한 부분이다.

If (openFileResult.bSuccess)

// 파일 선택이 되었다면 로그를 남겨주기 위해 생성한 클래스이다.

ofLogVerbose(“User selected a file”);

// 불러온 파일의 이름을 filename이라는 문자열 변수에 저장한다.

string fileName = openFileResult.getName();

// 파일 이름을 콘솔에 출력한다.

printf(“file name is “%s\n”, fileName);

// 파일 경로를 filePath 변수에 저장한다.

filePath = openFileResult.getPath();

// open을 콘솔에 출력해준다.

printf(“open\n”);

// 파일 경로 문자열의 마지막 “.”의 인덱스를 저장한다.

pos = filePath.find\_last\_of(“.”);

// pos가 존재하고, pos+1부터 끝까지 문자열을 확인하였을 때 maz 이면,

// 즉, 확장자가 maz이면 조건문안의 내용을 실행한다.

if (pos != string::npos && pos != 0 && filePath.substr(pos+1) == “maz”)

// ofFile 클래스를 이용해 파일객체를 정의한다.

ofFile file(fileName);

// 파일이 존재하지 않는다면

If (!file.exists())

// 존재하지 않는다는 에러메세지를 출력해주고 파일 읽기에 //실패했으므로 false를 리턴한다.

cout << “Target file does not exists.” << endl;

return false;

// 파일이 존재한다면

else

// 존재한다는 메세지를 출력해주고, isOpen을 1로 변경해준다.

cout << “We found the target file.” << endl;

isOpen = 1;

// 버퍼로 파일을 로드한다.

OfBuffer buffer(file);

// ToDo 부분

// .maz 파일을 input으로 받아서 적절히 자료구조에 넣는다.

// 미로의 세로와 가로를 저장한다.

HEIGHT = height of the maze;

WIDTH = width of the maze;

// 미로의 모든 정보를 저장할 input배열을 동적할당한 후 정보를 저장// 한다.

Input = new int\*[HEIGHT];

for each HEIGHT i:

Input[i] = new int[WIDTH];

// 그래프 탐색과정에서 쓰이는 visited배열을 동적할당 한다.

visited = new int\*[HEIGHT];

for each HEIGHT i:

visited[i] = new int[WIDTH];

// 파일 읽기가 성공했으므로 true를 리턴한다.

return true;

// 파일 확장자가 maz가 아닌 경우

else

// 에러메세지를 콘솔에 출력해주고, 파일 읽기를 실패했으므로 false를 // 리턴해준다.

printf(“ Needs a ‘.maz’ extension\n”);

return false;

<freeMemory 함수>

freeMemory 함수는 프로그램을 작동시키면서 할당해주었던 메모리를 모두 해제하는 함수이다. 메모리를 해제해주어야 여러 파일을 불러오며 진행해야 하는 과정에서 메모리 관련 에러들이 생기지 않고 원활히 작동할 수 있다.

// ToDo 부분

// malloc한 memory를 free해주는 함수

// visited 배열을 new로 동적할당해주었기 때문에 delete로 메모리를 해제해준다.

// 이차원 배열이므로 일차원 배열들을 모두 해제해주고, 더 상위 배열을 해제해준다.

For each HEIGHT x:

delete[] visited[x];

delete[] visited;

// input 배열을 new로 동적할당해주었기 때문에 delete로 메모리를 해제해준다.

// 이차원 배열이므로 일차원 배열들을 모두 해제해주고, 더 상위 배열을 해제해준다.

For each HEIGHT x:

delete[] input[x];

delete[] input;

// DFS 함수에서 필요한 배열들의 메모리를 해제해준다.

free(stack); // iterative dfs를 구현하기 위해 필요했던 stack의 메모리를 해제해준다.

free(checkeDfs); // dfs로 탐색했던 모든 위치를 화면에 표시하기 위해 필요했던 배열의 메모리를 해제해준다.

<Draw 함수>

Draw 함수는 화면에 그림을 그려주는 함수로, .maz 파일을 로드하여 미로의 정보를 가져온 이후에는 미로를 그려주고, dfs과정을 진행하여 탈출경로를 찾아낸 경우에는 탈출경로와 탐색경로를 모두 그려주고, bfs과정을 진행하여 탈출경로를 찾아낸 경우에는 탈출경로와 탐색경로를 모두 그려주는 함수이다.

// 화면의 바닥에 comsil project를 출력해주기 위해 만든 문자배열이다.

Char str[256];

// 미로를 그릴 색깔은 파란색이다.

ofSetColor(0,0,255);

// 미로를 그릴 선의 굵기는 5이다.

ofSetLineWidth(5);

// ToDo 부분

// 저장된 자료구조를 이용하여 미로를 그린다.

// 미로를 그려준다.

For each HEIGHT i:

For each WIDTH j:

// 가로벽이면 가로로 길게 그림을 그려준다.

If (input[i][j] == “-“)

Draw horizontal line

// 세로벽이면 세로로 길게 그림을 그려준다.

else If (input[i][j] == “|“)

Draw vertical line

// DFS 가 실행되었다면, 탈출경로와 DFS 탐색을 한 경로를 둘다 화면에 그려준다.

If DFS has done:

// 경로는 굵기가 5인 선으로 그려준다.

ofSetLineWidth(5);

// 파일이 열려있다면 탈출경로와 DFS 탐색을 한 경로를 둘다 화면에 그린다.

// 그려야 할 두 가지 과정은 dfsDraw에 구현되어 있다.

If (isOpen):

dfsDraw();

// 파일이 열려있지 않다면 에러메세지를 출력한다.

else:

cout <<”You must open file first” << endl;

// show DFS나 show BFS 메뉴에 체크가 되어있다면

If (bShowInfo):

// comsil project라는 문구를 str에 저장해준다.

sprint(str, “ comsil project”);

// str에 저장된 문자열을 선택된 지정된 위치에 그려준다.

myFont.drawString(str, 15, ofGetHeight()-20);

1. ofxWinMenu는 깃허브에 있는 오픈 소스 프로젝트이다. 따라서, 깃허브에서 프로젝트가 올라온 디렉토리의 readme.md를 읽어보며 메뉴와 버튼 추가방법에 대해 조사하였다. 이 프로젝트는 windows.h 헤더파일의 멤버함수 및 클래스를 이용한다. 가장 먼저, menu = new ofxWinMenu(this, hWnd); 를 이용해 현재 클래스에 대한 메뉴 객체 포인터를 생성한다. 그리고 CreateMenuFunction을 통해 메뉴를 만드는데 인자로는 menu가 어떻게 작동해야 하는지를 담고있는 함수포인터가 들어온다. 그리고 윈도우 메뉴를 만들기 위해서는 windows.h 헤더파일에 정의되어있는 클래스인 HMENU를 이용하여서 HMENU hMenu = menu->CreateWindowMenu();를 작성한다. 이를 통해 메뉴를 추가할 수 있는 준비를 해주고 “FILE” 메뉴를 만들기 위해서 HMENU hPopup = menu->AddPopupMenu(“FILE”)이라고 해주면 된다. 이렇게 한 후 FILE 메뉴에 마우스를 가져다대면 하위 버튼들이 등장하게 된다. 만약, “Open”이라는 하위 버튼을 생성하고 싶다면 menu->AddPopupItem(hPopup, “Open”, false, false); 라고 작성해주면 된다. 인자 중에 첫번째 false는 체크되어 있지 않다는 것을 의미하고, 두번째 false는 자동체크가 되어있지 않아야 한다는 것을 의미한다. 이 과정을 통해 미로 프로그램 윈도우 창에서 메뉴와 버튼 추가를 할 수 있다. 또한, 하위버튼끼리 서로 구분해주려면 menu->AddPopSeparator(hPopup)을 해주면된다. 희미한 선이 생겨서 버튼끼리 구분이 가능해진다. 또한, 메뉴와 버튼을 윈도우 창에 띄우기 위해서는 menu->SetWindowMenu();를 작성해주어야 한다.