2020년도 2학기 컴퓨터공학설계및실험I

13주차 결과 보고서

20170175 김태안

1. 실습 목적

본 실험은 1주차 미로 만들기 프로젝트의 결과물인 ~.maz 파일을 읽어서 이를 윈도우 화면상에 표시함을 목적으로 한다. 실험목적인 GUI(Graphical User Interface) 구축과 동시에 3주차에서 행할 프로젝트를 위하여 학생들은 창의적이고 효율적인 자료구조를 설계하여야 한다. GUI를 용이하게 설계하기 위하여 OpenFrameWorks를 사용하여 구현하도록 한다.

1. 실습 구현 내용

실습에서는 적당한 가로와 세로를 가진 완전 미로 파일만을 사용하나, 개인적인 호기심으로 다음의 기능을 추가하였다. 먼저, 미로의 가로 세로 제한 없이 미로의 입력에 따라 화면에 맞는 크기의 미로를 출력한다. 이때, 미로는 화면의 가운데에 출력한다. 그리고, 완전 미로와 불완전 미로를 모두 탐색할 수 있는 프로그램을 구현하기 위해 트리가 아닌 그래프를 사용하며, 그 중 인접 리스트를 사용해 미로 정보를 저장한다.

인접 리스트 구현을 위해 node와 SLList를 다음과 같이 정의하였다. 인접 리스트는 WIDTH \* HEIGHT 크기의 SLList 배열이다.

|  |  |
| --- | --- |
| class node {  public:  int cellNum;  bool visited;  int parent;  node\* link;  node() {  cellNum = -1;  visited = false;  parent = -1;  link = NULL;  }  }; | class SLList {  private:  node\* first;  node\* end;  public:  SLList();  node\* front();  void insert(int cNum, int par);  void freeSLL();  void printSLL();  }; |

구현한 SLList의 멤버 함수는 다음과 같다.

SLList();

SLList의 생성자

node\* front();

SLList의 맨 앞 노드를 반환한다.

void insert(int cNum, int par);

SLList의 맨 뒤에 cNum과 par를 cellNum과 parent로 가지는 노드를 삽입한다.

void freeSLL();

SLList의 모든 노드의 메모리를 해제하고 SLList를 해제한다.

void printSLL();

디버깅을 위한 함수로, SLList의 모든 노드를 출력한다.

구현한 오픈프레임워크 함수는 다음과 같다.

void setup();

초기 변수를 설정한다. 화면의 Frame Rate를 15로 설정하고, 배경을 흰색으로 칠한다. 전역 변수와 flag를 모두 0으로 설정한다.

void keyPressed(int key);

종료 명령인 Q를 입력 받으면 FreeMemory 함수를 호출하고 프로그램을 종료한다. 이후 DFS와 BFS 경로를 그리는 명령인 D와 B를 추가할 예정이다.

void keyReleased(int key);

l을 입력 받으면 파일을 불러온다. .maz 파일을 선택하면 파일명을 readFile 함수로 넘겨준다. readFile 함수가 파일을 성공적으로 열면 loadFlag를 1로 설정한다.

bool readFile(const char\* filename);

파일명을 인자로 받아 해당 파일을 연다. 파일 열기에 성공하면 다음 과정을 진행한 뒤 true를. 실패하면 에러 메시지와 함께 false를 반환한다. 기존에 파일을 열어 미로를 저장했다면 FreeMemory 함수를 호출한다. 파일을 열어 그 내용을 ofBuffer maz 변수에 저장한다. 그리고, 파일을 한 줄 씩 읽으며 미로의 가로와 세로를 계산해 저장한 뒤, 관련 메시지를 콘솔에 출력한다. mazeFlag를 1로 설정해 draw 함수가 미로를 그려도 됨을 알린 뒤, WallSizeCal 함수와 AdjListGen 함수를 호출하고 파일을 닫는다. 가로의 계산은 파일을 한 줄씩 읽으며 맨 마지막 줄에 도달할 때까지 HEIGHT를 1씩 추가한 뒤, 1을 빼고 2로 나눈다. (미로의 한 칸은 +, |의 두 줄로 이루어지기 때문이다.) 세로의 계산은 첫 줄의 길이를 length() 함수를 사용해 구한다. 따라서 가로와 세로의 계산은 O(Lines of .maz file)의 시간 복잡도를 가진다.

void AdjListGen();

버퍼 채로 저장한 .maz 파일을 읽으며 미로의 모습을 파악하고, WIDTH\* HEIGHT의 인접 리스트를 생성한다. 미로의 각 칸을 0, 1, 2, …, WIDTH\*HEIGHT라고 하고, 파일을 한 줄 씩 읽으며 오른쪽 경계선이 존재하지 않거나 아래쪽 경계선이 존재하지 않으면 인접한 두 칸을 인접 리스트에 삽입한다. 예를 들어, 1과 2가 인접하면 1에 2를, 2에 1을 삽입한다. 파일의 모든 내용을 한번 씩 탐색하며 저장하기 때문에 O(Number of characters in .maz file)의 시간 복잡도를 가진다.

void WallSizeCal();

미로의 각 벽은 사각형으로 이루어져 있다. +는 정사각형, -와 |는 직사각형으로 표현한다. 정사각형의 한 변의 길이이자 직사각형의 짧은 변의 길이를 sWall, 긴 변의 길이를 lWall이라고 하자. lWall이 sWall보다 3배 길다고 할 때, 화면에 꽉 찬 미로를 그릴 수 있는 벽의 길이는

이다. 이를 계산해 sWall과 lWall을 구한다. 미로를 화면의 가운데에 그린다고 할 때, draw함수에서 쓰일 미로의 시작 좌표는 다음과 같이 계산한다.

void draw();

저장한 버퍼 maz를 한 줄 씩 읽으며 미로를 그린다. 미로의 시작점은 WallSizeCal에서 계산한 (X, Y)이다. maz를 한 줄 씩 읽으며 +면 정사각형을, -나 |면 직사각형을 그린 뒤, x 좌표를 사각형의 끝으로 갱신해 다음 사각형을 그려 나간다. 한 줄을 다 읽으면 y 좌표를 갱신해 다음 줄을 그린다. 파일을 한 자 씩 읽으며 미로를 그리기 때문에 O(Number of characters in .maz file)의 시간 복잡도를 가진다.

void freeMemory();

프로그램을 종료할 때나, 새 파일을 읽을 때 실행된다. 생성한 인접 리스트 Adj의 모든 연결 리스트를 해제하고, 배열을 삭제한다. 전역 변수를 초기화 하고, 저장한 버퍼도 clear 함수를 통해 지운다.

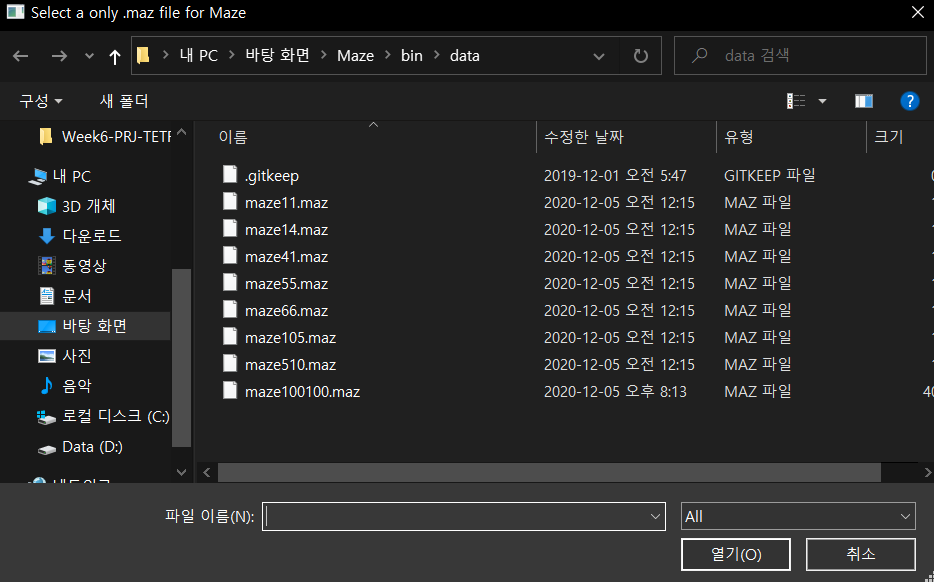
1. 실습 환경

Microsoft Visual Studio Community 2019 버전 16.7.7

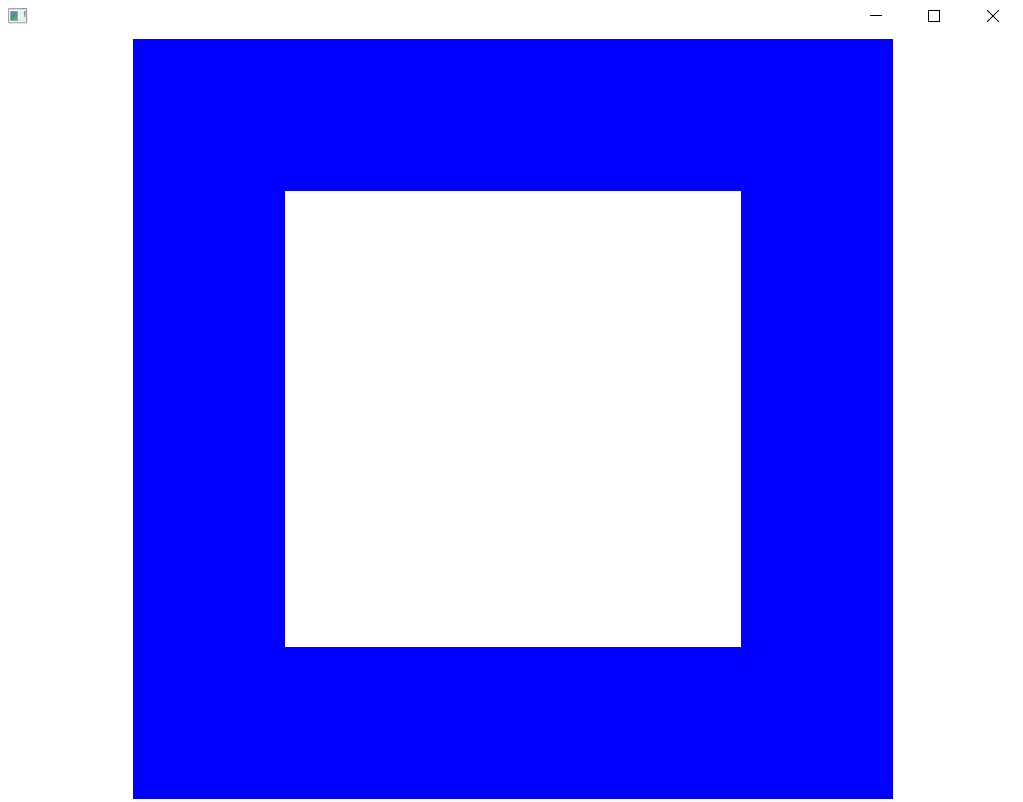
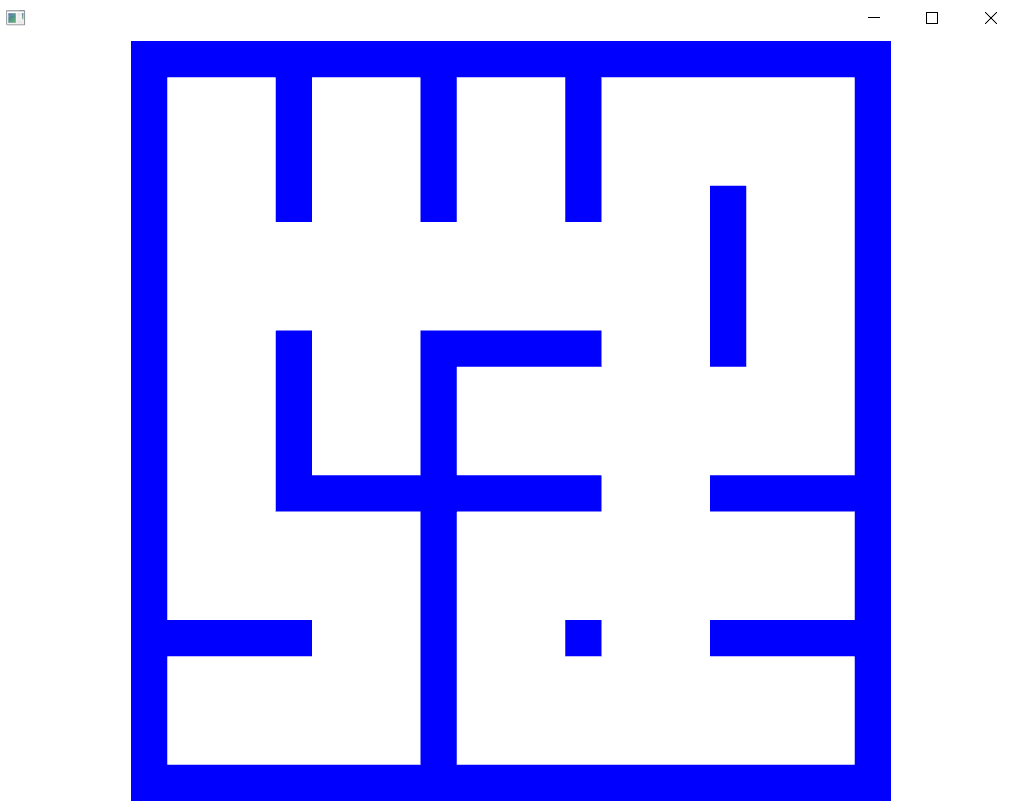
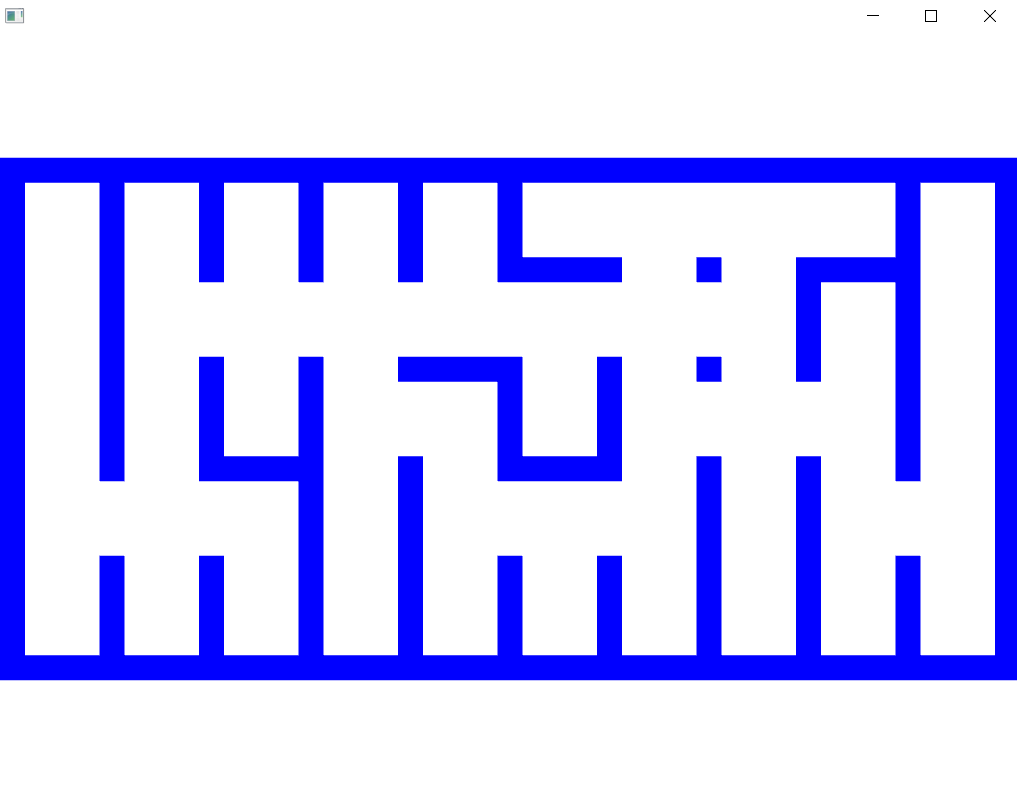
OpenFrameworks v0.11.0 vs2017 Release

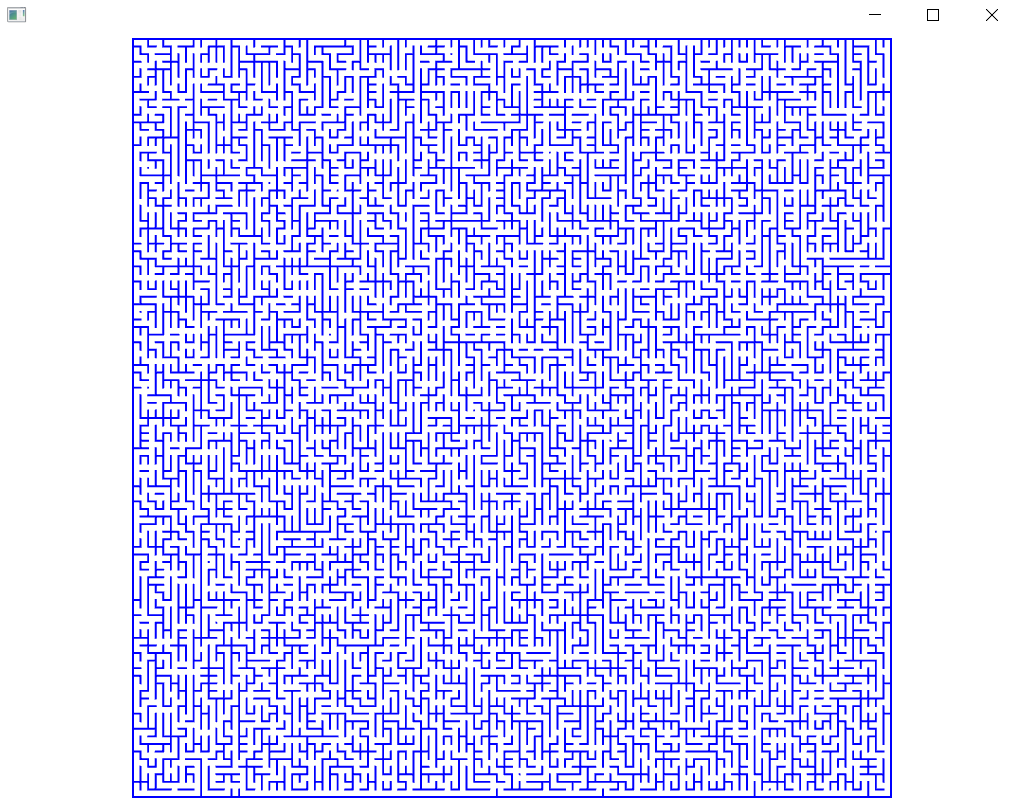
1. 실습 결과 및 분석

l 버튼을 눌러 미로를 불러올 수 있다.



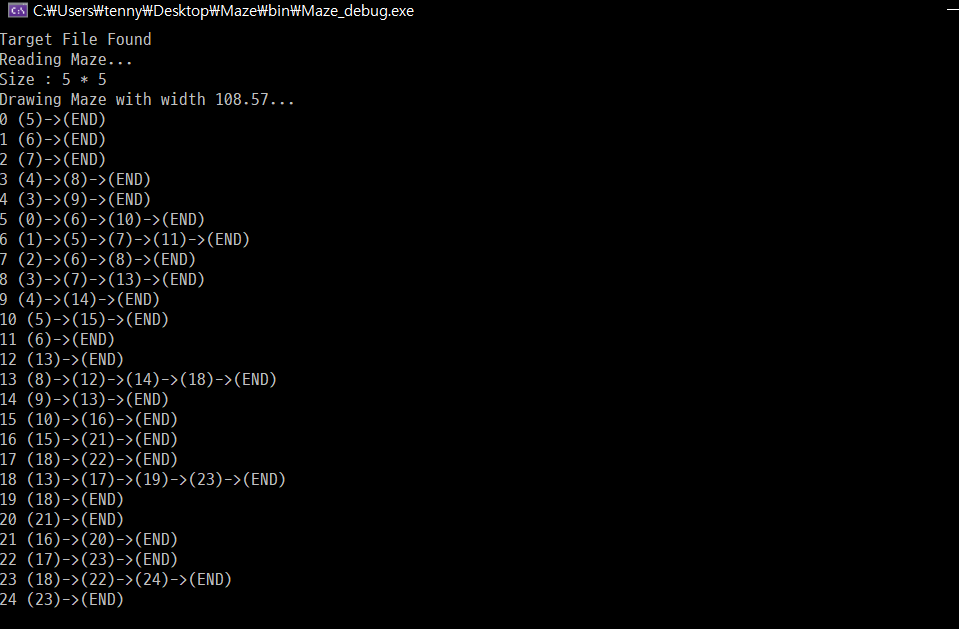
다양한 크기의 미로 파일을 불러오면 다음과 같이 창에 맞게 출력한다.



1\*1, 5\*5, 5\*10, 100\*100 미로를 출력한 화면

그 중, 5\*5 미로를 불러와 생성된 인접 리스트의 결과를 콘솔에 출력하면 다음과 같이 정상적으로 인접 리스트가 생성된 모습을 확인할 수 있다.



1. 결론

이번 실습에서는 지난 실습에서 구현한 미로 생성기를 통해 생성한 .maz 파일을 읽어 화면에 그린 뒤, 인접 리스트에 저장하는 프로그램을 작성하였다. 미로의 가로와 세로를 모른 상태로 파일만 주어지기 때문에 미로의 가로 세로 계산 -> 미로 데이터 저장 -> 미로 데이터를 읽으며 미로 그리기 -> 미로 데이터를 읽으며 인접 리스트 생성의 과정을 거치는 O(2\*WIDTH +2\*HEIGHT+3 \* HEIGHT \* WIDTH)의 시간 복잡도를 가지는 알고리즘을 사용하려 했으나, 버퍼를 통째로 저장해 데이터를 저장하는 과정을 생략하여 시간/공간 복잡도를 더 단축시키는 방법을 사용해보았다. 그 결과, 버퍼 이외의 추가로 미로를 위해 사용한 자료구조는 없었다. 그러나 미로 파일의 모든 정보를 읽게 되어 결국 비슷한 시간 복잡도를 가지게 되었다. 가로와 세로의 계산에서 버퍼의 크기 정보를 활용해 가로와 세로를 계산하는 방법을 구상해보았으나 미로 파일이 \n로 끝나는가의 여부에 따라 값이 달라져 결국 가로는 length() 함수를 사용하고, 세로는 파일을 한 줄 씩 읽는 방식으로 구현하였다.