12주차 결과보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

1-1.

결과보고서를 쓰기에 앞서 이 결과보고서에는 실습 시간에 작성한 우리가 실제 미로를 그리는데 사용한 자료구조 위주로 설명하였다.

다음은 ofApp.h 헤더파일에서 실험시간에 필요한 전역변수 및 flag, 필요한 함수를 새로 설정해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 readFile() 은 파일을 읽을 때 사용하는 함수이다. 그리고 freeMemory() 는 미로를 그리기 전에 .maz 파일에서 미로를 받아올 때 사용하는 input 2차원 배열을 동적으로 메모리를 할당하는데 이를 다시 free 시켜주는 함수이다. HEIGHT는 미로의 높이, WIDTH는 미로의 너비, char\*\* input을 통해 .maz 파일의 미로를 받아오고 isOpen은 파일이 열렸는지 열리지 않았는지 확인하는 flag이다. Bool drawing은 readFile()에서 파일을 받아와 input 배열에 정상적으로 입력되어 미로를 그릴 수 있는지 없는지 확인하는 flag이다.

여기서부터는 ofApp.c 파일에서 미로를 받아오고 또 윈도우 창에 그려주는 작업을 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Setup 함수에서는 c 파일이 실행될 때 초기값을 설정해주는 곳이다.

isOpen 은 file이 열린 상태인지 열리지 않은 상태인지 확인하는 flag이다. Drawing이 0이라는 의미는 아직 readfile() 함수가 실행되지 않아 input 배열에 저장된 정보가 없어 미로를 그리지 못한다는 뜻이다. 이를 통해 새로운 미로를 윈도우 창에 띄우고 싶을 때 만약 미로가 이미 그려졌다면(drawing이 1이라면) freememory를 통해 input 2차원 배열 메모리를 free 시키고 새로운 input 메모리를 재할당하여 새로운 미로를 그린다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음은 setup 함수 내에서 윈도우 창에 tool bar버튼을 추가해주는 코드이다. 여기서 새로운 toolbar 메뉴를 만들기 위해 AddPopupMenu() 함수를 이용하여 “Toolbar” 라는 이름을 가진 popupmenu 를 새로 추가해 주었다. 이후 AddPopupItem() 을 통해 “Toolbar” 메뉴 하위 메뉴(item) 에 13 주차 실습에서 미로를 그릴 “Show DFS”, “Show BFS” 라는 popup item 을 만들어 주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Draw() 함수는 .maz 파일에서 input 동적메모리 할당 2차원 배열에 해당 미로 정보들을 저장하고이를 통해 윈도우 창에 미로를 그리는 함수이다. 우선 ofSetColor(100), ofSetLineWidth(5) 함수들을 이용하여 사용자가 원하는 미로의 색깔, 그리고 미로의 선 두께 등을 결정해 줄 수 있다. Float x와 float y는 미로를 그릴 때 사용하는 x, y 좌표를 의미한다. Float로 미로를 그리는 좌표를 받은 이유는 미로를 그릴 때 사용하는 ofDrawLine 함수에서 float를 받을 수 있기 때문에 int에 비해 더욱 세부적인 값을 수정이 가능한 float를 사용했다. (하지만 실제 구현에 필요한 실험 목적 특성 상 그냥 윈도우 화면에 미로만 보여주면 되서 세밀한 값이 필요 없기 때문에 미로 한 칸의 길이를 30 으로 설정했다.) 이때 drawing이 1이라면 input 2차원 배열에 미로에 대한 정보들이 저장되었다는 뜻으로 이제 미로를 그릴 수 있다.

2중 for문을 통해 첫 for문에는 전체 미로의 높이만큼 반복한다. 이때 HEIGHT\*2+1 만큼 돌아가는 이유는 아래에서 설명하겠지만 HEIGHT는 실제 미로의 높이를 의미하고 HEIGHT\*2+1 은 전체 미로의 높이이다. 이 말은 .maz에서 미로 데이터를 받아올 때 가장 첫 줄은 미로의 가장 윗 테두리 벽이다. 그리고 두번째 줄이 실제 미로의 높이의 1칸, 세번째 줄은 다시 under 벽을 의미하는 줄이다. 즉 전체 미로에는 HEIGHT\*2+1 만큼의 높이가 존재한다. 마찬가지로 두번째 for문에서는 전체 미로의 가로만큼 반복한다. 이는 WIDTH\*2+1 인데 WIDTH는 실제 미로의 가로 길이를 의미한다. 전체 미로의 가로 길이는 처음엔 가장 왼쪽의 미로 테두리 벽, 다음은 실제 미로의 한 방의 가로, 다음은 side 벽을 의미하는 가로가 들어오게 된다. 최종적으로 전체 가로 길이는 WIDTH\*2+1 이 된다.

이 2중 for문을 돌면서 이제 input의 [0][0] 부터 [HEIGHT\*2][WIDTH\*2] 까지 Index의 값이 무엇인지 확인하여 ofDrawLine으로 미로의 선분들을 그려준다.

만약 input의 값이 ‘+’ 라는 것은 미로의 모서리들을 의미한다. 즉 이는 아무것도 그려주지 않는다.

만약 input의 값이 ‘-‘ 라는 것은 bottom, under 벽이나 가장 윗 테두리, 아래 테두리를 의미하기 때문에 ofDrawLine(x,y,x+30,y) 로 (x,y) 에서 (x+30,y) 의 선분을 그린다. 즉 미로의 한방의 가로 길이는 30이라는 뜻이다. 이후 x좌표에도 실제로 30을 더하여 다음 선분을 그릴 준비를 한다.

만약 input의 값이 ‘|’ 라면 side 벽이나 미로의 양 옆 테두리를 의미하기 때문에 ofDrawLine(x,y,x,y+30)으로 (x,y) 에서 (x,y+30)의 선분을 그린다. 즉 미로의 한방의 세로 길이는 30이라는 뜻이다. 여기서는 수직선만 그리기 때문에 추가적인 x,y 좌표의 수정이 필요 없다.

만약 input 값이 ‘ ‘ 이라는 것은 해당 공간이 방이라는 의미이거나 under 벽 이나 side 벽인데 뚫긴 곳이기 때문에 if(j%2==1)을 통해 빈 방일 때만 찾아준다. 이 때는 x좌표를 30만큼 더해주어 해당 방을 건너 뛰고 그려야 한다.

이후 2번째 for문 즉 한 행이 다 그려졌다면 이제 다음 행을 그려야 한다. 그래서 x 값을 0으로 초기화하여 다시 왼쪽부터 순차적으로 그리도록 한다. 만약 i%2==1 이라면 실제 미로를 의미하는 미로들의 한 행의 방들을 그린 것이기 때문에 y 역시 30을 더해주어 다음 행을 그릴 준비를 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

appMenuFunction 함수에서는 윈도우 창에서의 버튼을 클릴 할 때 일어나는 일로 Open 이라는 버튼을 누른다면 readFile() 함수를 실행하여 .maz 파일을 읽어 들어야 한다. 그리고 Exit 버튼을 눌렀을 때 drawing이 1, 즉 미로가 이미 그려져 있다면 freeMemory 함수를 실행하여 input 2차원 동적 할당 배열의 메모리를 해제 시켜준다. 그리고 윈도우 창을 종료 시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이번 실습 시간에서 미로를 윈도우 창에 그릴 수 있도록 .maz 파일을 읽어 input 2차원 배열에 저장하는 핵심 함수인 readfFile()에 대해 알아보겠다. 이는 위의 사진과 아래 사진으로 구분 지어 캡쳐 하였다. 비록 미리 상단의 코드들은 주어져 있었지만 간단히 설명을 해보았다. openFileDialogResult 클래스를 통해 openfileresult를 선언하고 ofSystemLoadDialog 로 파일 선택시 .maz 파일만을 선택할 수 있도록 유도하기 위해 이에 해당하는 문구를 창 왼쪽 상단에 띄웠다. 이후 파일의 경로를 가지고 있는 filePath가 있고 pos 는 해당 파일이 pos 뒤로 ‘maz’ 와 일치하는지 학인하여 .maz파일이 맞는지 확인할 때 사용하는 데 여기에는 “.” 의 위치가 들어가야 한다. 만약 openFileResult.bSuccess가 true 이면 다음 과정을 시행한다. 이후 ofLogVerbose 로 파일을 선택하였다는 로그도 남긴다. 그리고 fileName 에는 openFileResult.getName() 을 통해 파일의 이름을 받아와 저장한다. 그리고 filePath에 openFileResult.getPath() 를 통해 파일의 경로를 받아와 저장한다. Pos에는 filePath.find\_last\_of(“.”)로 ‘.’ 의 위치를 입력하고 이것을 통해 pos!=0&&filePAth.substr(pos+1)==”maz” , pos가 0이 아니고 pos 이후의 문자열이 maz임을 판단하고 이에 맞다면 ofFile 을 이용하여 file을 받는다. 만약 파일이 존재하지 않는다면 결국 파일을 못읽었기 때문에 더이상 readFile 함수를 진행할 필요가 없어 false를 return 하고 그렇지 않다면 isOpen을 1로 하여 정상적으로 파일을 가져와 읽기를 시행할 수 있음을 알린다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이후 여기서는 실제 실험자가 작성한 코드로 .maz 파일의 미로 정보를 input 배열에 저장하는 과정이다. 우선 string num\_line을 설정하여 여기에 .maz 파일의 정보를 한 줄 씩 받아오도록 한다. 만약 이미 미로가 그려져 있다면(drawing이 1이라면) 즉 미로가 그려져 있는 상태에서 다시 새로운 미로를 그리기 위해서는 input 배열이 메모리 해제가 되어야 하고 다시 새롭게 메모리를 재할당 받아야 해서 freeMemory 함수를 실행한다. 이후 실제 미로의 가로, 세로길이를 의미하는 WIDTH, HEIGHT를 0으로 초기화 한다. 실제 미로의 가로, 세로 길이를 먼저 받아 들여야 이후 for문에서 input 배열의 2차원 메모리 할당 및 정보 입력을 할 수 있다. 그래서 while(getline(file,num\_line)) 을 통해 getline을 이용하여 file의 정보를 한 줄 씩 num\_line에 받아와 이게 0이 아닐 때 까지 HEIGHT를 1씩 늘린다. 전체 가로의 길이는 자연스럽게 한 줄의 길이를 의미하는 num\_line.length()임을 알 수 있다. 그리고 이제 미로의 정보를 저장할 input 배열을 2차원으로 동적할당 받아야 하는데 우선 new char\*[HEIGHT]로 높이 만큼 new를 한 다음 for문을 통해 input[i] 마다 new char[num\_line.length()] 메모리를 준다. 메모리를 정상적으로 할당 했다면 이제 HEIGHT 에는 실제 미로의 높이를 부여하기 위해 HEIGHT=HEIGHT/2를 한다. 위에서도 설명했지만 실제 미로의 높이\*2+1=전체 미로의 높이 이다. WIDTH 역시 실제 미로의 가로 길이를 저장하기 위해 num\_line.length를 2로 나눈 수를 저장한다. 실제 미로의 가로\*2+1=전체 미로의 가로 이다. 이후 과정은 .maz 파일에 들어있는 미로 정보를 input 배열에 저장하는데 이미 file 은 위에서 HEIGHT, WIDTH를 받아오는데 while 문을 통해 끝까지 .maz 파일의 끝까지 이동했기 떄문에 다시 file2라는 곳을 통해 .maz 파일을 새로 불러온다. String word 를 선언하여 여기에 실제 파일 내용을 한 줄씩 읽어 들인다. 이후 2중 for문에서 첫 for문은 전체 미로의 높이인 HEIGHT\*2+1 만큼 돌면서 getline(file2,word)을 하여 file2의 내용을 한 줄 씩 word에 저장한다. 그리고 다음 안쪽 for문에서 전체 미로의 가로 길이인 WIDTH\*2+1 만큼 for문을 돌면서 input[i][j] 에 word[j]를 넣어준다. 즉 한 문자 씩 input 배열에 저장한다. 이후 2중 for문이 모두 끝났다면 drawing을 1로 하여 draw() 함수에서 미로를 그릴 수 있도록 해준다.

만약 파일이 “.maz” 확장자 파일이 아니라면 오류 메세지를 출력하고 false를 return 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마지막 사진은 freeMemory 함수인데 여기서는 input 2차원 동적할당 배열의 메모리를 해제 시켜주는 곳이다. 첫 1차원 배열들의 메모리를 for문을 돌면서 delete[] 를 통해 모두 해제를 시켜주고 그 상위 메모리 역시 delete를 통해 지워준다. 이 함수는 exit 버튼을 누를 때, 이미 미로가 그려져 있는 상태에서 다시 새로운 미로를 그리고 싶을 때 호출된다.

1-2.

해당 미로를 읽어 들이고 다시 미로를 윈도우 창에 그리는 그래픽 전환 작업에서의 시간 및 공간 복잡도에 대해 알아보자. draw() 함수에서 미로를 그릴 때 input 2차원 배열에서 순차적으로 iput [0][0] 부터 input [HEIGHT\*2+1][WIDTH\*2+1] 까지 탐색하면서 그리기 때문에 2중 for 문을 돌린다. 즉 WIDTH, HEIGHT는 변경될 수 있는 값이므로 시간 복잡도는 O(WIDTH\*HEIGHT) 이다. 또한 readfile() 함수에서 input 2차원 동적 할당을 해줄 때, .maz 파일로부터 읽어드린 정보를 다시 input 배열에 순차적으로 단어 1개씩 저장해줄 때 2중 for문을 사용하였다. 여기서도 시간 복잡도는 O(WIDTH\*HEIGHT)이다. freeMemory에서 input 메모리를 해제시켜 줄 때는 HEIGHT만큼의 for문을 돌기 때문에 시간 복잡도는 O(HEIGHT) 이다. 즉 최종적인 시간 복잡도는 O(WIDTH\*HEIGHT) 이다.

공간 복잡도는 결국 전체적인 코드의 자료구조 상 미로의 정보들을 저장하고 있는 input의 2차원 동적 할당에서의 O(WIDTH\*HEIGHT) 임을 알 수 있다.

1-3.

사실 12주차 예비보고서의 pseudo 코드를 작성할 때 거의 모든 코드 역시 구현 한 상태에서 짜서 큰 차이가 보이지 않는다. 이 때 가장 많이 고민한 것은 어떻게 maz 파일에서 HEIGHT, WIDTH를 받아오고 또 여기의 미로에 대한 정보 역시 받아올까 였다. 결국 두 곳에서 파일을 받아와서 하나는 HEIGHT, WIDTH 길이를 저장할 때 쓰이고 다른 하나는 미로의 정보를 받아 올 때 쓰기로 하였다. 그리고 두번째로 많이 고민한 것은 freeMemory를 어느 순간에 해줘야 할 까 였다. 이는 exit 버튼을 누를 때, 미로가 그려진 상태에서 다시 미로를 그릴 때 호출하면 되었다.

2.

2번에 대한 내용은 어떻게 보면 1-3 에서 작성한 내용의 연장선이라고 생각하면 된다. 사실 파일을 읽어 들이는 과정에 대해 이전까지는 대부분 코드가 주어져 있다 보니 큰 고민을 하지 않았었다. 하지만 이번 실습에서는 파일 읽는 과정을 코드로 구현해야 했기 때문에 getline에 대해 공부를 할 수 있었다. Getline을 통해서는 한 줄 씩 파일의 정보들을 받아 올 수 있는데 만약 “\n” 이 있다면 자연스럽게 한 줄을 다 읽었음을 알려주는 것을 관찰할 수 있었다. 이때 파일을 한 번만 읽어 HEIGHT, WIDTH 를 저장하고 다시 이를 통해 input 배열에 미로의 정보를 넣을 수 있을까 에 대해 고민했다. 하지만 getline 특성 상 한 줄 씩만 받아오고 다시 윗 줄을 읽지는 못하는 것 같았다. 그래서 HEIGHT, WIDTH를 저장하는 동시에 그 미로에 대한 정보를 읽어 들이는 것은 getline만으로는 힘들다는 것을 알았다.

그리고 waterfall, tetris 등 이전 실습에서도 가장 힘들었던 메모리에 대한 오류 역시 이 실습에서도 발생하였다. 주로 미로가 이미 그려진 상태에서 다시 미로를 그릴 때였는데 그냥 exit파일 누를 때만 처음엔 메모리를 해제 하면 된다고 생각했는데 실제로는 새로운 미로를 그릴 때마다 input 배열을 새로 할당 받아야 함을 알 수 있었다.

마지막은 ofFrameWork에 대한 것인데 우선 비록 우리가 윈도우 창의 버튼, pop, item 등을 직접 구현을 하지는 않았지만 추가적인 “toolbar” popup 메뉴와 하위 메뉴 “show dfs”, “show bfs” popup item 들을 넣어주어 어떻게 윈도우 창에 메뉴 및 버튼을 넣고 어떻게 동작하는지에 대해서는 이해할 수 있었다. 그리고 ppt에서 파일을 열 때 기본확장자를 ~.maz 에 대해서만 설정하도록 3페이지에 적혀 있었다. 이를 해결하기 위해 현재 5개의 소스 파일을 순서대로 다시 읽어보고 많은 검색을 해보았지만 쉽게 찾을 수 없었다. 결국 openframework 파일의 libs에서 이에 대한 비슷한 해답을 찾을 수 있었다. Of\_v0.11.2\_vs2017\_release->Libs->openframeworks->utils 의 ofSystemUtils.cpp 파일이 있다. 여기에 ofn.IpstrFilter 가 기본값인 All 로 되어있는데 이를 “Example File(\*.maz)\0\*.maz\0All\0\*.\*\0” 로 바꾸어 .maz 파일을 불러올 때의 창의 기본 확장자를 .maz로 바꿨다. 이렇게 모르는 것들을 직접 하나씩 찾아가면서 openframework를 조금씩 이해할 수 있었다. (비록 제출 시에는 해당 파일만 제출하고 ofSystemUtils.cpp 는 제출하지 않았으니 아래와 같이 기본 확장자를 Example File(\*.maz) 로 나오게 하지는 않을 것이다.)

텍스트이(가) 표시된 사진

자동 생성된 설명