**최종 프로젝트 결과보고서**

전공: 컴퓨터공학과 학년: 4학년 학번: 20192135 이름: 윤영인

**1. 프로젝트 목표 및 실험 환경**

**1.1 목표**

이 프로젝트는 시간 안에 특정 지점 (츄르가 있는 지점)을 반드시 지나야 하는 미로 탈출 게임이다. 사용자가 ‘r’버튼을 누르면 BFS로 특정 지점을 거쳐 탈출 경로까지 가는 최단 경로를 찾아 3초동안 길을 그려 줌으로써 사용자가 빠르게 게임을 진행할 수 있도록 한다. 이 프로젝트를 통해 사용자는 미로 탐색 알고리즘과 그래프 이론에 대한 이해도를 향상시킬 수 있다.

**1.2 실험환경**

Visual Studio 2019

**2. 변수**

**2.1 구조체**

1) Node

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

큐를 구성하는 노드 구조체이다. r과 c 변수는 미로의 좌표를 의미하며 연결리스트로 노드를 잇기 위해 next 포인터를 정의하였다.

2) MyQueue

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

큐를 사용하여 BFS를 구현하기 위해 큐 구조체를 선언하였다. front와 rear 포인터 변수는 큐 구조체에서 front와 rear를 의미하고, len은 큐에 담긴 노드의 개수를 의미한다.

3) Pos

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

큐에서 미로를 탐색할 때 이전 위치를 저장하기 위한 구조체로, 위치의 좌표를 담기 위해 r, c 변수를 저장하여 선언하였다.

4) allPath

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

BFS로 탐색한 탈출 지점까지 잇는 경로를 찾아 연결리스트로 표현하기 위해 선언한 구조체이다. 탐색한 좌표 지점의 노드가 들어가며 head 포인터 변수는 연결리스트의 첫 시작 부분을 의미한다.

**2.2 변수**

1) int HEIGHT: 미로의 높이

2) int WIDTH: 미로의 너비

3) char \*\*input: 텍스트 파일의 미로 정보를 담는다.

4) int \*\*visited: 미로 방의 방문 여부를 저장

5) int maze\_col: 미로를 저장한 배열의 열 인덱스

6) int maze\_row: 미로를 저장한 배열의 행 인덱스

7) int isOpen: 파일이 열렸는지 판단하는 변수, 0이면 안 열렸고 1이면 열렸다.

8) int isBFS: BFS 함수를 실행했는지 판단하는 변수, 0이면 실행 안 했고, 1이면 실행했다.

9) int weight: 미로 한 칸의 간격

10) int \*\*map: input 파일에 저장된 미로를 1과 0으로 표현한 배열, 벽이면 0, 방이면 1

11) allPath \*ap, \*sp: 도착 지점까지 경로를 저장하는 연결리스트, ap는 츄르가 있는 지점->도착지점, sp는 시작 지점->츄르가 있는 지점의 경로를 저장한다.

12) int dr[4] = {1, 0, -1, 0}, dc[4] = {0, 1, 0, -1}: 탐색할 방향을 나타내는 배열로, 남, 동, 북, 서 차례로 탐색한다.

13) ofImage startImage, endImage, middleImage, churImage: 이미지를 나타내는 변수로, 츄르 지점에 가기 전까진 startImage로, 츄르 지점에 도착하면 middleImage로, 도착 지점에 도착하면 endImage로 변경되어 이미지가 이동한다. churImage는 츄르 사진을 띄우는 변수이다.

14) float imageR, imageC: 사용자가 키보드로 움직일 사진의 좌표를 저장하는 변수

15) int churR, churC: 츄르가 있는 지점의 좌표

16) bool isTimerRunning: 타이머가 움직이는지 나타내는 변수

17) float startTime, elapsedTme, hintTime, totalTime, saveTotalTime: 시간을 나타내는 변수로, startTime은 시작 시간, elapsedTime은 경과 시간, hintTime은 힌트 시작 시간, totalTime은 도착지점까지 사용자가 도착해야하는 제한 시간이다. saveTotalTime은 게임을 재시작할 경우 제한 시간을 원래대로 세팅하기 위해 초기 제한 시간을 저장한다.

18) bool startFlag: 게임이 시작했는지 확인하는 변수

19) bool recommend: 사용자가 힌트를 사용 중인지 확인하는 변수

20) bool isFail: 게임에 실패했을 경우를 나타내는 변수

21) bool churFlag: 츄르가 있는 지점에 갔는지를 나타내는 변수

22) bool arriveFlag: 게임에 성공했을 경우를 나타내는 변수

**3. 함수**

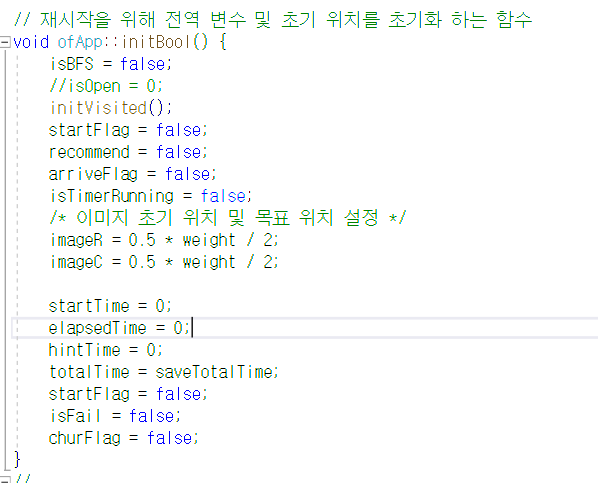
1) setup()

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

게임 화면이 띄워질 때 프로그램을 세팅하는 함수이다. 타이틀 바에 “Save the Cat!” 문구가 띄워지며, 프레임 속도를 초당 15 프레임으로 설정하였다. 헤더 파일에서 선언한 startFlag, recommend 변수를 false로 초기화하였고, 움직일 캐릭터의 초기 위치를 설정하였다. 또한 화면에 글자를 띄우기 위해 timerFont와 failFont를 verdata.ttf 글씨체와 15, 30으로 크기를 설정하였다. 또한 소리를 사용하기 위해 startSound, failSound, successSound, churSound, timerSound에 소리 파일을 로드했다.

2) initBool()



게임을 다시 시작하기 위해 전역 변수를 초기화하는 함수이다. 헤더 파일에서 선언한 전역 변수들을 모두 0 또는 false로 초기화하였고, 움직일 캐릭터의 위치도 초기 위치로 설정하였다.

3) appMenuFunction()

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

게임 창 상단에 띄울 메뉴를 구성하는 함수이다. Open을 누를 경우, 동적 할당된 메모리를 해제하고, 변수를 초기화하며 readFile로 파일을 불러온다. Exit을 누를 경우, 동적 할당된 메모리를 해제하고 창을 종료한다. Full screen을 누르면, 창이 전체화면으로 띄워진다.

4) readFile()

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Open 메뉴를 눌렀을 때, 미로 파일을 읽는 함수이다. 파일의 확장자가 .maz일 경우에만 파일을 가져오며 ofBuffer 타입의 file과 buffer 변수에 가져온 .maz 파일을 넣는다. 가져온 파일이 존재하면 “We found the target file.” 문구를 출력하고 isOpen 변수를 1로 세팅한다. 만약 존재하지 않으면 “Target file does not exists.” 문구를 출력하고 false를 리턴하고 함수를 종료한다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

읽은 파일을 저장한 buffer를 한 줄씩 읽어가며 미로 그림의 사이즈를 측정한다. 미로 파일을 저장할 배열을 생성하기 위해 행과 열 인덱스의 총 크기를 결정한다. maze\_col은 WIDTH \* 2 + 1, maze\_row는 buffer의 line 수를 잰 HEIGHT로 설정하였다.

파일 별 total time을 설정하였는데, 5일 경우 15초, 그 외 파일들은 20초로 설정하였다. 또한 게임을 재시작할 경우, 제한 시간을 원래 시간으로 세팅하기 위해 saveTotalTime에 기존 제한 시간을 저장하였다.

미로 파일을 저장하기 위해 input과 map을 maze\_row \* maze\_col 만큼 동적할당하였고 미로의 칸을 방문했는지 확인하기 위한 배열인 visited 배열도 maze\_row \* maze\_col만큼 동적할당하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

동적할당한 visited 배열을 0으로 초기화하고, buffer를 line 별로 돌면서, input 파일에 미로 파일을 저장하였다. 그리고 읽은 input 파일이 ‘ ‘이면 방이므로 1, ‘ ‘ 이 아닐 경우엔 벽이므로 0으로 map에 미로 파일을 0과 1로 변환하여 저장하였다. 그리고 랜덤으로 츄르의 위치를 설정하기 위해 map이 벽이 아닐 경우, 벡터형 변수인 randomMap에 위치를 저장한다. 그리고 rand() 함수를 이용하여 randIdx를 설정하여 랜덤으로 츄르의 위치를 설정하였다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

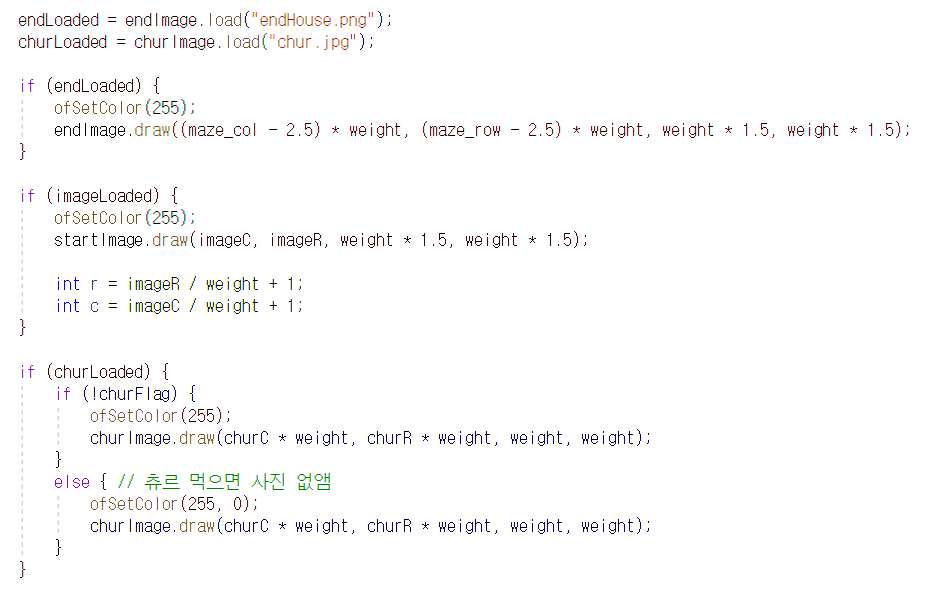
만약 읽은 파일의 확장자가 .maz 파일이 아닐 경우, “Needs a ‘.maz’ extension” 문구를 띄우고 false를 리턴하여 함수를 종료한다.

5) draw()

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

읽은 미로 파일을 게임 창에 그리는 함수이다. 간격이 weight인 미로의 테두리인 직사각형을 그리고, input 파일을 돌면서 간격이 weight인 가로 및 세로 벽들을 그린다. 미로를 다 그렸으면, 출발하거나 게임에 실패했을 때 쓰일 이미지, 츄르를 먹고 아직 도착하지 않았을 경우의 츄르를 먹고 나서 쓰일 이미지, 도착했을 때 쓰일 이미지를 가져온다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

또한 도착 지점에 띄울 endImage와 츄르 이미지를 가져온다. 이미지를 가져왔으면 도착 지점에 이미지를 띄우고, 사용자가 상, 하, 좌, 우로 키보드를 눌러 이미지를 이동시킬 때마다 이미지를 이동시켜 띄우도록 한다. 츄르 이미지를 가져왔으면, 츄르를 아직 먹지 않았으면 츄르 이미지를 churR, churC 위치에 띄우고, 먹었다면 이미지를 지운다. 그리고 화면에 키보드 조작 방법 및 타이머를 그린다. 타이머는 제한 시간에서 남은 시간을 측정하기 위해 (제한 시간 – 소요 시간)을 계산하여 화면에 띄웠다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

만약 시간 안에 도착 지점에 도착하지 못해서 isFail 변수가 1이 되면, Fail 문구를 띄우고, 시간 안에 츄르를 먹고 도착 지점에 도착하여 arriveFlag가 1이 되면 success 문구를 띄운다. 사용자가 힌트를 얻기 위해 r 키보드 버튼을 눌러 BFS() 함수를 실행하여 isBFS가 1이 되면 탐색한 최단 경로를 bfsdraw로 그리도록 하고, 파일을 아직 가져오지 않은 경우엔 “You must open file first” 문구를 띄운다. 그리고 게임 창 하단에 “20192135 YounginYoon’s comsil final project”를 띄운다.

6) keyPressed()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

키보드 조작 함수이다. ‘esc’ 버튼을 누를 경우, 전체 화면이면 작은 창으로 조절하고, 작은 창일 땐 창을 종료하도록 한다. ‘f’ 버튼을 누르면 전체화면 모드가 되거나 전체화면 모드를 끌 수 있다. ‘s’ 또는 ‘S’ 버튼을 누를 경우 게임을 시작하는데, startFlag, isTimerRunning을 true로 설정하고, 타이머를 측정하기 위해 startTime을 ofGetElapsedTimeMillis로 측정한다. 그리고 시작 사운드를 play한다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

게임을 시작하고 나서는 사용자는 상, 하, 좌, 우로 키보드를 눌러 이미지를 이동시키는데, 이동 시키기 전에 checkToMove로 이동할 수 있는지 판단하여 이동 가능하면 이동시킨다. ‘r’ 또는 ‘R’ 버튼을 누르면 사용자는 3초동안 힌트를 얻을 수 있다. 따라서 recommend 변수를 true로 설정하고, 3초를 측정하기 위해 hintTime을 ofGetElapsedTimeMillis로 측정한다. 그리고 경로를 저장할 ap와 sp를 null로 초기화한 후, 현재 이미지의 위치를 간격을 제거한 위치로 설정하여 해당 위치에서 BFS로 최단 경로를 측정한다. BFS로 경로를 탐색했으면 isBFS를 true로 설정하고 타이머 사운드를 play한다. ‘q’를 누를 경우 동적 할당한 메모리들을 할당 해제하고 창을 꺼 게임을 종료한다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

‘e’ 버튼을 누를 경우, 게임을 재시작하기 위해 initBool로 모든 전역 변수와 이미지 위치를 초기화한다.

7) checkToMove()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

사용자가 키보드로 위치를 조작할 때 이동가능한 위치인지 판단하는 함수이다. 인자로 받아온 위치가 미로를 벗어났거나 간격 weight를 제거한 위치에서 map이 벽일 경우 false를 리턴하고 이동 가능한 위치라면 true를 리턴한다. 그리고 만약 그 위치가 츄르가 있는 위치인데 아직 츄르를 먹지 않아서 churFlag가 false이면 제한 시간을 5초 늘리고, chutFlag를 true로 설정한 후 츄르 사운드를 play한다. 또한 도착 지점에 도착했을 때, 츄르를 먹은 상태라면 타이머를 멈추고 게임을 종료하기 위해 isTimerRunning, startFlag를 false로 설정하고 arriveFlag를 true로 설정한 후 successSound를 play한다.

8) makeNode()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

큐를 구성하는 노드를 만드는 함수이다. 각 지점의 좌표 값을 인자로 받으며, Node 타입의 메모리를 동적으로 할당한다. 할당한 노드 공간에 인자로 받아온 좌표 값을 넣어주고 next 포인터에 NULL을 넣고 리턴한다.

9) makeQueue()

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

큐를 생성하는 함수로, MyQueue 타입의 메모리를 동적으로 할당하여 front와 rear에 nullptr과 len에 0을 넣어 초기화한 후 리턴한다.

10) isQueueEmpty()

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

큐가 비었는지 확인하는 함수로, len이 0이면 true, 아니면 false를 반환하도록 하였다.

11) pushQueue

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

큐에 새로운 노드를 push하는 함수로, 선입선출인 큐의 특성에 맞도록 구현하였다. 큐가 비어있으면 front와 rear에 새로운 노드를 넣었다. 비어있지 않으면 선입선출을 하도록 rear 위로 새로운 노드를 넣고 새로운 노드로 rear를 업데이트 하였다. 큐에 노드를 추가하였으면 len을 1 증가시켰다.

12) popQueue()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

큐에서 노드를 팝하는 함수로, front를 팝하도록 구현하였다. 큐가 비어 있으면 NULL을 반환하고, 그렇지 않으면 front를 리턴하기 위해 front의 좌표를 담은 새로운 노드를 생성한다. 만약 큐의 길이가 1이면 front와 rear에 nullptr을 넣고, 그렇지 않으면 front를 그 다음 노드로 변경한다. front를 free하고 큐의 길이를 1 줄인 후 front의 좌표를 담아 생성한 노드를 리턴한다.

13) initVisited()

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

방문했는지 확인하는 배열인 visited을 0으로 초기화하는 함수이다.

14) makePath()

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

BFS로 탐색한 모든 경로와 탈출 지점까지 잇는 경로를 찾아 넣기 위한 연결리스트를 생성하는 함수이다. 연결리스트를 allPath 타입의 메모리 공간으로 동적할당하고 head 부분에 NULL을 넣어준 후 리턴한다.

15) BFS()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

큐를 탐색하면서 이전 좌표를 저장하기 위해 Pos 타입의 2차원 배열을 선언하고 -1로 초기화하였다. ap에 경로를 담기 위해 makePath로 초기화하였다. 만약 츄르를 아직 먹지 않은 상태에서 사용자가 r을 눌러 힌트를 얻고자 할 땐 findChur 함수로 현재 위치에서 츄르 지점으로 가는 최단 경로를 탐색하고 시작 위치를 츄르가 있는 지점으로 설정하여 큐에 넣어준다. 만약 츄르를 먹은 상태에서 힌트를 얻고자 할땐 시작 위치를 인자로 받아온 위치로 설정하여 큐에 넣어준다. initVisited 함수로 visited 배열을 0으로 초기화하고 시작 위치의 방문 체크를 한다. 큐가 비지 않을 동안 탐색을 하는데 큐를 pop하여 탐색할 위치인 cur을 정하였다. cur의 위치가 탈출 지점일 경우 while문을 탈출한다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

만약 cur이 탈출 지점이 아닐 경우, 현재 위치에서 남, 동, 북, 서 차례로 한 칸씩 이동하여 움직일 수 있는 방인지 확인한다. 만약 아직 방문하지 않은 이동가능한 방일 경우 방문 체크를 해주고 큐에 담는다. 그리고 prev 배열의 이동한 위치에 현재 좌표 값을 넣어 이전 경로를 저장하도록 한다. bfs는 탐색하면서 탈출 지점에 도착했을 경우가 가장 최단 경로이므로, 이전 경로가 담긴 배열을 확인하면서 최단 경로를 저장하는 ap에 경로를 담도록 하였다. pR과 pC에 탈출 지점에 담긴 이전 좌표를 담아서 시작 지점이 될 때까지 경로를 역추적하여 업데이트하도록 하였다. 이 과정은 최단 경로를 따라 이전 노드를 거슬러가면서 역추적하는 함수이다. 최단 경로를 ap에 담았으면 큐와 prev 배열을 할당 해제한다.

16) addPath()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

BFS로 탐색한 최단 경로들을 ap에 노드로 추가하는 함수이다. 만약 ap가 비어 있으면 head에 새로운 노드를 넣어주고, 그렇지 않으면 가장 끝 노드를 찾아 새로운 노드를 넣어준다.

17) findChur()

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

자동 생성된 설명

아직 츄르를 먹지 않은 경우에 사용자가 힌트를 얻기 위해 ‘r’ 버튼을 눌렀을 때, 사용자 위치에서 츄르 지점까지의 최단 경로를 탐색하는 함수이다. 시작 위치를 사용자의 현재 위치로 설정하여 큐에 담았고, 큐가 비지 않을 동안 탐색을 하는데 큐를 pop하여 탐색할 위치인 cur을 정하였다. cur의 위치가 츄르가 있는 지점일 경우 while문을 탈출한다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

나머지는 BFS 함수와 구현이 동일하므로 설명을 생략하겠다.

18) addShortestPath

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

findChur 함수로 탐색한 사용자 위치->츄르 지점의 최단 경로들을 ap에 노드로 추가하는 함수이다. addPath와 같은 방식으로 구현하였다.

19) bfsdraw()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

ap와 sp에 담긴 최단 경로들을 선으로 화면에 그리는 함수이다. 츄르를 아직 먹지 않아서 sp에 경로가 담긴 경우엔 prev와 cur 노드의 위치를 설정하여 간격을 weight로 설정하여 선을 그려준다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

또한 ap도 마찬가지로 위와 같은 방식으로 ap에 담긴 경로들을 선으로 그려준다.

20) update()

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

어플리케이션의 상태를 업데이트하는 함수이다. 타이머에 시간을 표시하기 위해, ofGetElapsedTime 함수로 현재 시간을 측정하고 startTime을 빼서 소요 시간을 측정한다. 만약 소요 시간이 제한 시간을 넘겼을 경우, isTimerRunning, startFlag를 false로 설정하여 게임을 멈추고 타이머에 0을 표시하기 위해 elapsedTime을 totalTime \* 1000으로 설정하였다.

21) freeMemory()

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

동적 할당한 메모리들을 할당 해제하는 함수이다. 헤더에서 선언하여 배열로 사용한 input, map, visited를 할당해제하고, 경로가 담긴 ap와 sp를 할당해제하였다.

**4. 플로우차트 및 자료구조 (알고리즘), 시간/공간 복잡도**

**4.1 플로우차트**

텍스트, 도표, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

**4.2 자료구조 및 알고리즘**

최종 프로젝트를 진행하면서 사용한 자료구조는 연결리스트와 큐이며, 알고리즘은 BFS를 사용하였다. 큐를 직접 구현하기 위해 노드 구조체를 생성하였고, 각 노드 별로 next 포인터를 사용하여 단일 연결리스트로 큐를 구현하였다.

큐는 선형 자료구조로, FIFO(First-In-First-Out) 원칙으로 동작하여 가장 맨 앞 요소는 front, 가장 맨 뒤 요소는 rear로 표현한다. 즉, 가장 먼저 들어온 요소가 가장 먼저 나오는 방식으로 동작하여 줄을 서서 기다리는 상황과 동일하다. Enqueue로 새로운 요소를 넣을 때는 큐의 rear에 추가되며, Dequeue로 요소를 삭제할 때는 front에 있는 요소를 삭제한다. 주로 연결리스트나 배열로 구현되며 해당 프로젝트에선 연결리스트를 이용하여 구현하였다.

연결리스트는 데이터 요소들을 노드로 구성하여 연결된 구조를 가지며, 각 노드는 다음 노드를 가리키기 위해 포인터를 사용한다. 연결리스트를 사용하면 동적으로 데이터를 추가, 삭제할 수 있어 메모리 공간 유지에 효율적이다.

BFS (Breath-First Search) 알고리즘은 그래프 탐색 알고리즘으로, 가까운 너비에 있는 노드를 우선으로 탐색한다. 해당 프로젝트에선 사용자의 위치를 시작 정점으로 하여 해당 정점에서 가까운 정점들을 먼저 방문하고, 멀리 떨어져 있는 정점들은 나중에 방문한다. 이 알고리즘은 두 노드 사이의 최단 경로 또는 임의의 경로를 찾을 때 사용한다.

**4.3 시간 및 공간 복잡도**

BFS 함수로 경로를 탐색할 때 필요한 시간은 큐의 크기에 비례한다. 따라서 시간 복잡도는 O(V+E)이며, V는 노드의 수, E는 간선의 수를 의미한다. 또한 공간 복잡도는 prev 배열과 노드 개수에 비례하므로 O(N^2)이다.

또한 연결리스트를 사용하여 큐를 구성하거나, 경로를 저장할 때 연결리스트의 모든 노드들을 순회하면서 삽입, 삭제한다. 따라서 시간 복잡도는 연결리스트의 노드의 수에 비례하므로 O(n)이며, 공간 복잡도 역시 단일 연결리스트에 저장되는 노드의 개수에 비례하므로 O(n)이다.

**5. 창의적 구현**

해당 프로젝트는 실습 13주차의 과제를 응용하여 구현하였다. 기존 과제에선 시작에서 도착 경로까지의 최단 경로 및 방문한 모든 경로를 그리도록 하였다. 이 프로젝트에선 사용자가 직접 키보드로 방향을 움직이며 특정 지점인 츄르 지점을 들린 후 도착 지점에 도착해야 게임이 종료된다. 이때 사용자가 힌트가 필요한 경우 ‘r’ 버튼을 눌러 3초동안 BFS로 최단 경로를 찾아 그리도록 하였고, 최단 경로를 그릴 때도 츄르를 먹은 경우와 먹지 않은 경우로 나눠 그렸다. 만약 츄르를 먹지 않았을 때 사용자가 힌트를 원할 경우, 사용자의 위치에서 츄르 지점을 거쳐 도착 지점으로 가는 최단 경로를 그리도록 하였고, 츄르를 먹은 경우엔 사용자의 지점에서 도착 지점으로 가는 최단 경로를 그렸다. 츄르를 먹으면 제한 시간이 5초 늘어나며, 파일 별로 제한 시간을 다르게 설정하여 사용자가 제한 시간 안에 도착 지점에 도달하지 못했을 경우 화면에 “Fail!” 문구가 띄워지고, 시간 안에 도달했을 경우 “Success!” 문구가 띄워진다. 시작, 츄르를 먹었을 때, 도착 지점에 도달했을 때의 이미지가 바뀌며 효과음도 추가하였다.

**6. 실행 결과**

**6.1 게임 시작 (파일 별로 제한 시간이 다르다.)**

1) 5\_5.maz

텍스트, 스크린샷, 소프트웨어, 도표이(가) 표시된 사진

자동 생성된 설명

2) 10\_10.maz

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

자동 생성된 설명

3) 12\_8.maz

스크린샷, 도표, 텍스트, 직사각형이(가) 표시된 사진

자동 생성된 설명

4) maze0.maz

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

자동 생성된 설명

**6.2 츄르를 먹었을 때**

도표, 스크린샷, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명

**6.3 ‘r’ 버튼을 눌렀을 때**

1) 츄르를 아직 먹지 않았을 경우

스크린샷, 도표, 텍스트, 직사각형이(가) 표시된 사진

자동 생성된 설명

2) 츄르를 먹었을 경우

직사각형, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

**6.4 게임 종료**

1) 제한 시간 안에 도착하지 못했을 때

스크린샷, 도표, 텍스트, 직사각형이(가) 표시된 사진

자동 생성된 설명

2) 제한 시간 안에 츄르를 먹고 도착했을 때

스크린샷, 도표, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명

**7. 느낀 점 및 개선 사항**

최종 프로젝트로 미로 찾기 게임을 구현하면서, BFS와 큐 및 오픈프레임워크에 대해 자세히 알아보는 시간을 가졌다. BFS를 구현하기 위해 직접 연결리스트를 사용하여 큐를 구현하였다. 큐에 탐색할 지점을 넣을 때마다 동적으로 노드를 할당하여 큐에 push하고 현재 탐색할 지점을 큐에서 pop한 지점으로 정함으로써 큐의 동작 원리를 알 수 있었다. 또한 BFS로 최단 경로를 탐색하면서 BFS 알고리즘은 가장 먼저 도착한 경로가 최단 경로임을 알 수 있었다. 오픈프레임워크를 사용하여 효과음 및 이미지를 추가하고, 화면에 그림을 그리는 것을 구현해 보면서 다양한 애니메이션 효과에 대해 알 수 있었다.

게임 종료 시 “Success!”와 “Fail!” 문구를 띄우는 위치가 미로의 가운데에 정확하게 그리지 못했다는 점에서 각 파일 별로 미로 그림의 가운데 좌표를 찾아서 문구를 띄우는 개선사항이 필요하다. 뿐만 아니라 ‘s’ 버튼을 누를 경우 게임이 종료했을 때에도 소리가 난다는 오류가 있어, 게임이 종료했을 때 ‘s’ 버튼을 누를 경우 소리가 나지 않도록 하는 개선사항이 필요하다.