최종 프로젝트

보고서

20211511 김서아

최종 프로젝트 보고서

전공: 컴퓨터공학과 학년: 2학년(3학기) 학번: 20211511 이름: 김서아

**1. 프로젝트 목표, 실험 환경**

실험 환경은 C++언어와 visual studio를 기반으로 OpenFramework를 사용하여 구현하였다. 기존 maze프로젝트를 기반으로 하여 최종 프로젝트를 진행하였다. 프로젝트의 목표는 한 학기 동안 진행한 3가지 프로젝트(tetris, waterfall, maze)에서 다룬 내용 및 아이디어를 통합하여 미로찾기 게임을 만드는 것이다. 통합한 아이디어는 maze 프로젝트에서 구현한 미로 graph 자료구조를 기반으로 하여, tetris 프로젝트의 game기능을 추가하였고, waterfall의 사용자 key값을 받는 interactive한 기능을 추가하였다. 또한 실습시간에 다루지 않았던 OpenFramework의 함수 기능도 사용하였다.

**2. 각 변수에 대한 설명**

vector<string> Input

: input으로 받은 미로를 string으로 저장하는 변수이다.

int input\_h

: Input 변수로 받은 미로의 text기준 높이를 임시로 저장하는 변수이다. 후에 미로의 정확한 높이를 계산할 때 사용된다.

int input\_w

: Input 변수로 받은 미로의 text기준 너비를 임시로 저장하는 변수이다. 후에 미로의 정확한 너비를 계산할 때 사용된다.

int HEIGHT

: input\_h를 2로 나누어 정확한 미로의 높이를 저장하는 변수이다.

int WIDTH

: input\_w를 2로 나누어 정확한 미로의 너비를 저장하는 변수이다.

int\* Visited

: bfs 탐색을 하면서 이미 방문한 노드인지 아닌지 확인하기 위해 쓰이는 변수이다. 이미 방문을 했다면 1, 아직 방문을 하지 않았다면 0을 값으로 가진다.

int first\_open

: input 파일을 처음으로 open했는지 여부를 체크하는 변수이다.

int isOpen

: input 파일이 성공적으로 열렸는지 확인하는 변수이다.

bool isbfs

: bfs 함수를 실행했는지 여부를 확인하는 변수이다. 실행했다면 true, 실행하지 않았다면 false값을 갖는다.

int player\_x

: player의 x좌표 기준 현재 위치를 저장하는 변수이다.

int player\_y

: player의 y좌표 기준 현재 위치를 저장하는 변수이다.

int canmove

: player가 미로 안에서 더 이상 상, 하, 좌, 우로 이동할 수 있는지 여부를 확인하는 변수이다. 이동할 수 있으면 1, 이동할 수 없는 위치이면 0을 값으로 갖는다.

int target

: gameover되는 위치 즉, player가 도착해야 하는 지점을 저장하는 변수이다. 미로의 제일 우측하단에 있는 노드의 위치를 의미한다.

int t\_color

: target의 위치를 사용자에게 잘 보이게 하기위해 ofSetColor의 파라미터로 넣어 ofDrawTriangle로 생성된 target 위치 도형이 반짝일 수 있도록 색을 지정하는데 쓰이는 변수이다.

int treasure\_num

: 사용자가 원하는 난이도에 따라 maz 파일을 input으로 정한 것을 바탕으로 treasure의 개수를 저장하는 변수이다. easy는 1, medium은 3, difficult는 6을 값으로 갖는다.

int find\_treasure

: player가 수집한 treasure의 개수를 저장하는 변수이다.

int treasure[10]

: 랜덤하게 생성된 treasure위치를 저장하는 변수이다.

int find\_error

: player가 모든 treasure를 다 찾은 후에, target지점에 도착했는지 여부를 판단하는 함수이다.

int find\_item

: player가 item을 획득했는지 여부를 확인하는 변수이다. 획득했다면 1, 아니면 0을 값으로 갖는다.

int item\_loc

: 랜덤하게 생성된 item의 위치를 저장하는 변수이다.

int finish\_Flag

: 게임이 종료되었는지 확인하는 변수이다. 종료되기 전까지는 0, player가 모든 treasure를 다 찾은 후에 target지점에 도착하면 1을 값으로 갖는다.

int gameStart

: player가 게임을 시작했는지 여부를 확인하는 변수이다. 해당 변수가 1이되면 playtime 측정을 시작한다.

clock\_t start

: 게임을 시작하는 시점의 시간을 저장하는 변수이다.

clock\_t end

: 게임을 종료하는 시점의 시간을 저장하는 변수이다.

float playtime

: end에서 start 값을 뺀 값인 사용자의 playtime을 저장하는 변수이다. play하는 동안 screen의 우측에 실시간으로 출력된다.

int first\_bfs

: 사용자가 bfs 탐색을 여러 번 실행하지 않도록 제어하는 변수이다. 아직 실행하지 않았을 때에는 1, 한 번이라도 실행한 후에는 0을 값으로 갖는다.

int bfs\_loc

: bfs 탐색을 실행했을 때의 player의 현재위치 즉, bfs탐색을 시작할 노드 값을 갖는 변수이다.

ofSoundPlayer mySound

: game을 진행하는 동안 배경음으로 쓰일 mp3 파일을 받는 변수이다.

ofImage player

: screen에 출력할 player 유닛의 이미지를 받는 변수이다.

ofImage item

: screen에 출력할 item 유닛의 이미지를 받는 변수이다.

ofImage treasure\_img

: screen에 출력할 treasure 유닛의 이미지를 받는 변수이다.

ofTrueTypeFont font

: screen에 출력할 글자의 폰트를 받는 변수이다.

**3. 각 함수에 대한 설명**

void keyPresseed

: input으로 key값을 눌렀을 때, 실행되는 함수로 playtime시간 측정을 시작하고, 방향키를 입력받았을 때에는 player의 유닛이 이동한다. 또한 ‘h’ key를 입력 받으면 힌트를 제공한다.

void keyReleased

: input으로 받은 key가 더 이상 눌려 있지 않을 때, 실행되는 함수로 ‘h’key에서 한 번 손을 떼면 더 이상 최단 거리 힌트를 사용자에게 제공하지 않는다. (hint는 한 번만 사용가능)

bool readFile

: input으로 maz파일을 읽어서 미로의 기본 정보들을 변수에, makeGrpah를 호출하여 미로를 graph에 저장하고, treasure\_item\_check를 호출하여 유닛을 생성하는 함수이다.

void makeGraph

: input으로 받은 미로를 adjacency linked list graph를 이용하여 저장하는 함수이다.

void freeMemory

: 미로 찾기 게임을 진행하는 동안 동적할당 했던 변수들을 메모리해제 하는 함수이다.

void addq

: Queue 자료구조에 노드를 push하는 함수이다. Bfs 탐색할 때 사용된다.

void delete

: Queue 자료구조에서 노드를 pop하는 함수이다. Bfs 탐색할 때 사용된다.

bool BFS

: player의 현재 위치에서 target 지점까지 최단 거리를 계산하는 함수이다.

void bfsdraw

: BFS함수를 바탕으로 계산된 최단거리를 screen에 출력하는 함수이다. 사용자가 힌트를 사용할 때 보여준다.

void treasure\_item\_make

: treasure와 item유닛의 위치를 랜덤하게 생성하는 함수이다.

void treasure\_item\_check

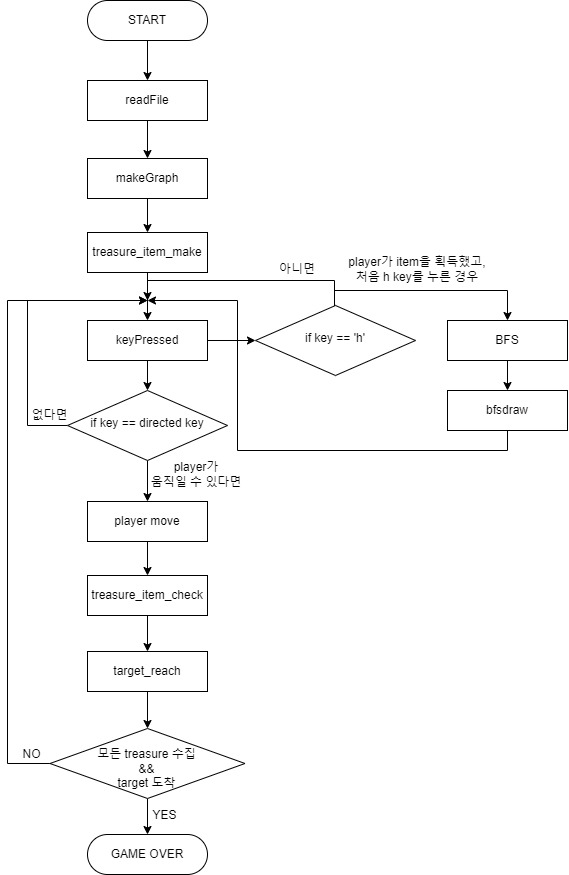
: player가 treasure나 item을 획득했는지 확인하고, screen에서 안 보이게 해주는 함수이다.

void target\_reach

: player가 target위치에 도착했는지 확인하고, 모든 treasure를 수집한 경우에 playtime을 계산하고 gameover하는 함수이다.

**4. 플로우 차트, 자료구조, 알고리즘, 시간/공간 복잡도**

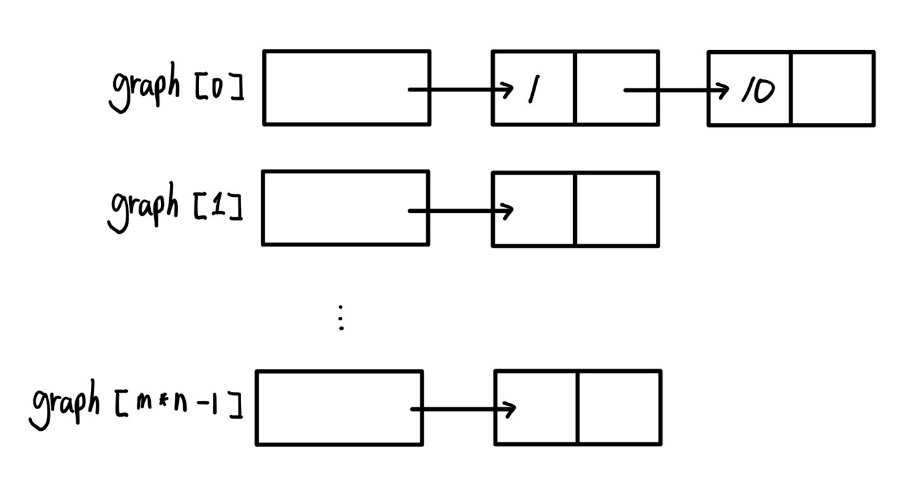
-flow chart



프로그램이 실행되면 input 파일을 읽어오고 읽어온 미로에 맞게 graph와 treasure, item 을 생성하고 화면에 출력한다. 사용자가 방향키를 입력했을 경우, 해당 방향으로 player가 이동할 수 있다면 player를 이동시킨 후, treasure나 item을 획득했는지 확인한다. 또한 target지점에 도착했는지 여부도 확인한다. 만약 모든 treasure를 수집하고 target지점에 도착했으면 게임을 종료하고, 그렇지 않으면 게임을 지속한다. 사용자가 ‘h’ key를 입력했을 경우, item을 획득한 후에 처음 ‘h’key를 입력한 것이라면 현재 위치로부터 도착 지점까지 bfs를 이용하여 최단 거리를 탐색하고, 화면에 잠시 보여준 후, 게임을 지속한다. 그렇지 않으면 아무 변화 없이 게임을 지속한다.

-자료구조, 알고리즘, 시간/공간 복잡도

입력 받은 .maz 파일에 따라 미로를 adjacency linked list를 이용하여 graph형태로 저장한다. 해당 자료구조를 시각화 하면 다음과 같다.



높이가 m 너비가 n인 미로의 경우 위와 같이 생성된다. graph[0] ~ graph[n\*m-1] 까지의 노드가 생성되고, 각 노드에 연결되어 있는 노드 즉, 각 방마다 연결되어 있는 방을 linked list 형태로 자식 노드를 연결한다. 이때 공간 복잡도는 O(n\*m) 이다.

방향키를 입력 받았을 때, player가 움직일 수 있는지 확인하는 과정을 살펴보자. 반복문을 통해 Player의 현재 위치의 graph[player 좌표]에 연결된 노드들의 vertex값과 비교하면서 값이 같은 노드가 있을 경우 해당 방향으로 움직일 수 있고, 그렇지 않은 경우 움직일 수 없다. vertex값을 비교하는 과정에서 비교해야 하는 노드는 최대 4개(상,하,좌,우)이므로 시간 복잡도는 O(1)이다.

이제 저장한 grpah를 BFS함수를 통해 탐색하는 경우를 살펴보자. 탐색하면서 노드를 add, delete할 queue 자료구조, Visited 배열, 부모 노드를 저장할 queue\_arr 배열을 이용하였다. Player가 현재 위치한 노드부터 방문하면서 해당 노드의 자식 노드를 방문하고 queue에 add한다. 또한, 어느 노드로부터 방문한 노드로 올 수 있었는지 기억하기 위해 방문한 노드의 부모 노드를 queue\_arr 배열에 저장한다. 더 이상 방문할 자식 노드가 없다면 queue에서 delete하여 나온 노드로 이동하여 다시 자식 노드를 방문하는 과정을 반복한다.

어떤 노드를 방문했을 때에 해당 노드의 Visited값은 1이 되어, 더 이상 방문하지 않는다. (초기 Visited 값은 0) 결국 n\*m 개의 모든 노드들은 최대 1번씩만 탐색될 수 있으므로 이때 시간 복잡도는 O(n\*m) 이다.

BFS 탐색을 한 결과를 바탕으로 bfsdraw하는 함수를 살펴보자. player의 현재 위로부터 도착 지점까지의 최단 거리를 보여주기 위해, 앞서 부모 노드를 저장했던 queue\_arr를 이용하여 도착 지점으로부터 어느 노드로부터 왔는지 역추적한다. 이때 공간 복잡도는 부모 노드를 저장하고 있는 노드는 최대 n\*m-1개 이므로 공간 복잡도와 시간 복잡도 모두 O(n\*m)이다.

**5. 창의적 구현**

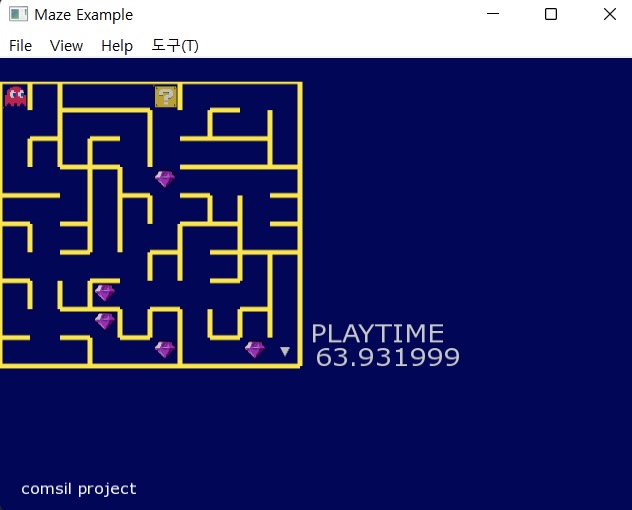
기존 maze 프로젝트에서 tetris와 waterfall 프로젝트의 아이디어, Openframework 함수들을 이용하였다. 먼저 사용자는 input으로 받는 미로 텍스트 파일을 통해서 쉬움, 보통, 어려움 난이도를 선택할 수 있다. 각각의 난이도의 차이점은 미로 크기, 찾아야 하는 treasure의 개수이다. input파일을 선택한 후에는 해당 난이도에 맞는 미로가 그려지고, player, treasure, item도 screen에 나타난다. 이때 실험시간에 다루지 않았던 Openframework의 함수를 사용해서 game에 맞는 mp3파일을 배경음악으로 load하였고, 유닛이 screen에 그려질 때도 해당 유닛에 맞는 이미지 파일을 load 하였다.

모든 것이 screen에 성공적으로 출력된 후에는 player는 미로 찾기 게임을 시작한다. Player는 방향키 →, ←, ↑, ↓를 사용하여 유닛을 움직일 수 있고, treasure를 다 모으고 target 지점에 도착해야만 게임이 종료된다. item을 획득한 경우에는 단 1번 ‘h’ key를 누르는 동안 현재 player의 위치에서 도착 지점까지 최단거리로 이동하는 방법을 hint로 제공한다. 게임을 진행하는 동안 우측에 playtime을 실시간으로 보여주며, player가 성공적으로 게임을 마치면 gameover 메시지와 함께 playtime을 screen에 출력하고 게임을 종료한다.

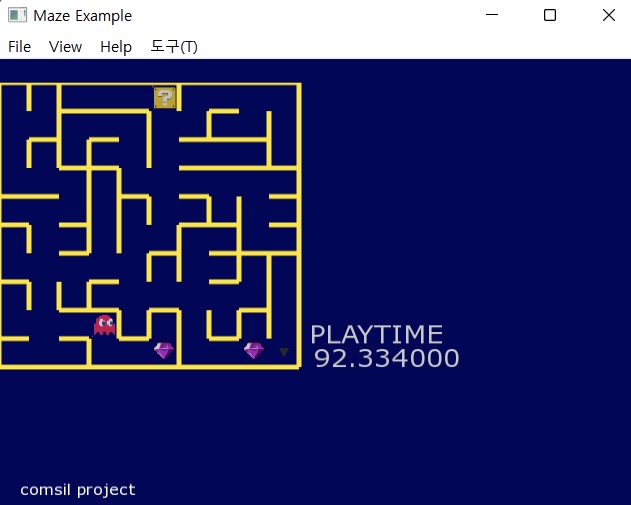
**6. 프로젝트 실행 결과 캡쳐**

텍스트이(가) 표시된 사진

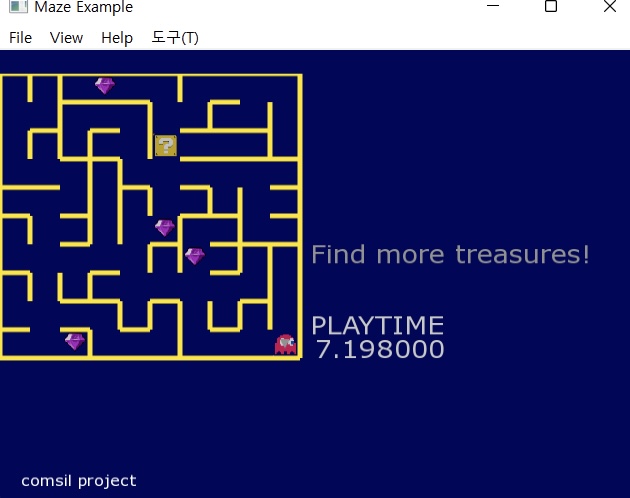
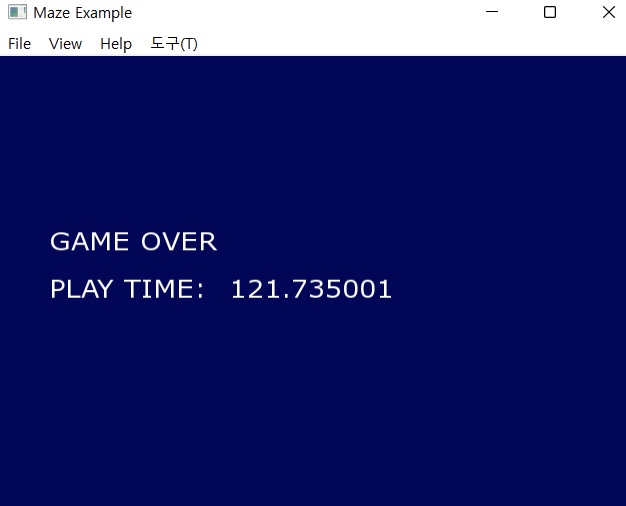
자동 생성된 설명 ← easy 모드

 ← medium 모드

 ← difficult 모드

보물을 먹을 때 item을 사용할 때

보물을 다 찾지 않고 도착했을 때 게임오버가 되었을 때

**7. 느낀 점 및 개선 사항**

프로젝트를 진행하면서 인상깊었던 아이디어들을 통합하여 하나의 최종 프로젝트로 구현해보았더니 결과물이 매우 뿌듯했다. 한 학기 동안 배운 내용들이 정리되는 느낌이 들었고, 처음 프로젝트를 진행할 때에 어렵게 느껴졌던 것들이, 최종 프로젝트를 만드는 지금 시점에서는 보다 조금 쉽게 느껴지는 부분도 있었다. 정말 알차게 배웠고, 조금 더 성장했다고 생각한다. 특히 Openframework를 이용하여 코드를 기하학적인 도형, 그림으로 출력하는 것은 매우 재밌었다.

개선하고 싶은 부분은 실험 시간에 진행했던 프로젝트의 도움 없이 완전히 새로운 프로젝트를 만들 수 있었으면 더 좋았을 것 같다. 아직은 나의 능력 밖의 일이라 생각되어 기존 프로젝트에서 시작했지만, 기회가 된다면 처음부터 끝까지 직접 내가 구현한 프로젝트를 만들어보고자 한다.