기말 프로젝트 보고서

전공 : 컴퓨터공학과 학년 : 2학년 학번 : 20211558 이름 : 윤준서

**1. 프로젝트 목표와 실험 환경**

본 프로젝트는 실습 시간에 다뤘던 미로 테마를 확장하여 간단한 게임 형식으로 만들어 이를 직접 플레이해볼 수 있도록 제작했다. 사용자가 직접 이동하여 미로의 출구를 찾을 수 있으며, 빠르게 출구를 찾을 수록 높은 점수를 받을 수 있는 경쟁 장치를 통해 여럿이서 함께 게임을 즐길 수 있다. 또한 점수의 크기에 따라 상위 랭크를 화면에 출력할 수 있어 승부욕을 자극한다.

본 프로젝트는 맥북 IOS의 Xcode 환경에서 OpenFrameworks로 제작했다. Window OS와 관련된 파일 또는 코드가 따로 추가로 존재하지 않는다. 하지만 실행 자체는 Window OS에서도 오류 없이 진행된다.

**2. 변수 설명**

ofPoint **pos** : 플레이어 위치를 표시하는 점.

ofPolyline **hint\_line** : 힌트를 표시하는 선.

string **message** : 윈도우 창에 메세지를 출력하는 문구.

vector<vector<bool>> **visited** : 플레이어 방문 체크를 위한 2차원 벡터.

vector<Line> **visited\_lines** : 플레이어 방문한 경로를 저장하는 벡터.

bool **move** : 플레이어의 방문 경로를 확인하기 위한 변수. visitied 벡터를 설정할 때 쓰인다.

struct **score\_rank**{ : 랭크 리스트를 구성하는 랭크 구조체. 점수와 게임 회 차로 이루어져있다

int score

int count

}

int **start\_flag** : S 버튼을 눌렀을 때 draw()에서 플레이어를 생성하도록 하는 표식.

int **hint\_flag** : H 버튼을 눌렀을 때 draw()에서 힌트 선을 출력하도록 하는 표식.

int **end\_flag** : 목표 지점에 도착했을 때 draw()에서 문구와 랭크를 출력하도록 하는 표식.

int **game\_count** : 게임 횟수를 저장하는 변수

score\_rank **list[15]** : 랭크 리스트다. ofWindow의 크기를 고려해 상위 15개만 저장한다.

int **count\_move** : 플레이어가 미로를 탈출하기 위해 움직인 총 횟수다.

이외의 변수들은 실습 시간에 다뤘던 변수들이다.

**3. 함수 설명**

void **path\_line(Line line)** : 큐에 저장한 힌트 경로를 선의 정보로 바꾸어 저장한다. 이는 draw()에서 힌트 경로를 출력할 때 용이하다.

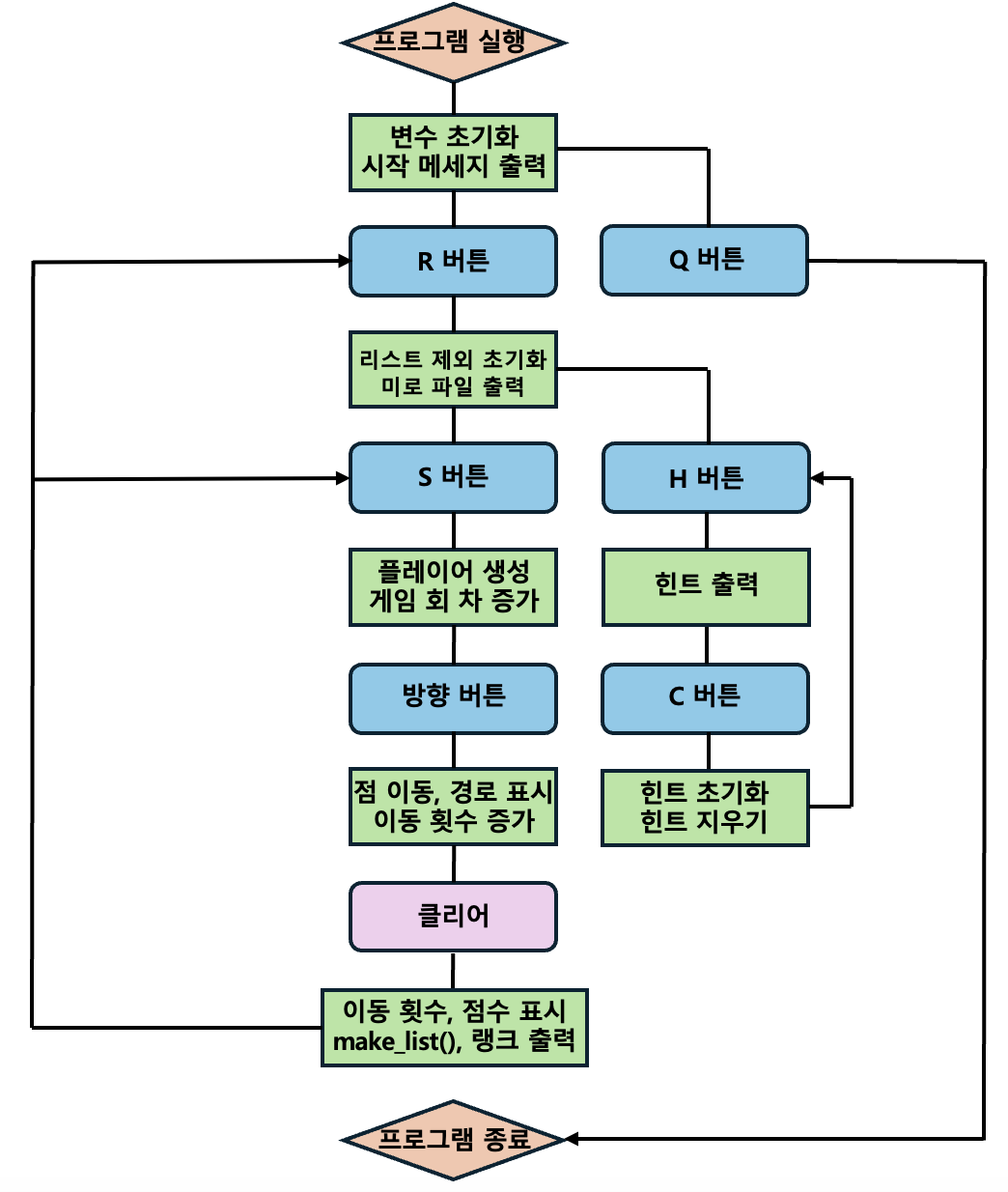
bool **hint()** : BFS 탐색을 통해 미로의 최적 경로를 찾아 이를 힌트로 플레이어에게 알려준다.

void **make\_list(int new\_score, int count)** : 게임이 종료된 후 점수와 해당 게임의 회 차를 랭크 리스트에 저장하여 갱신한다.

이외의 함수들은 실습 시간에 다뤘던 함수들이다.

**4. 기타 설명**

**4 - 1. 플로우 차트**



**4 - 2. 자료구조와 알고리즘**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 새로운 점수와 해당 게임 회 차를 랭크 리스트에 삽입한다.

2. 점수의 크기에 따라 내림차순으로 정렬한다.

3. 새 점수가 저장되었을 경우 종료한다.

텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

탈출에 성공한 경우 end\_flag = 1로 설정한다.

이 때 탈출하는 시점에도 이동 횟수를 증가시킨다.

매 이동마다 moved = true로 만들어 플레이어의 이동 경로를 저장한다.

moved를 이용해 플레이어의 위치를 저장한다.

2차원 벡터 visited를 통해 각 칸 마다 방문 여부를 저장한다. 즉 이들이 유효한 값을 가지는 총 칸 개수가 플레이어의 총 이동 횟수가 된다.

방문한 지점을 선의 형태로 시작 좌표와 끝 좌표를 line 벡터에 저장한다. 이 벡터를 그대로 draw()에서 이용하면 경로 출력을 쉽게 할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

ofWindow에 문구를 출력하는 방식이다. string message 변수에 원하는 문구를 부여하고 ofDrawBitmapString() 함수를 통해 출력할 수 있다.

이를 이용하여 게임이 종료되었을 때 총 이동 횟수와 최종 점수를 출력한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이외의 자료구조와 알고리즘은 실습시간에 다뤘던 내용들이다.

**4 - 3. 시간 복잡도와 공간 복잡도**

**hint()** : BFS를 이용한 최단 경로 탐색 함수다. 시작점 (1, 1)에서 시작하는데, 최악의 경우 미로의 모든 방을 연산해야 하므로 시간 복잡도는 O(WIDTH \* HEIGHT)이다. 공간 복잡도 또한 최악의 경우 미로의 모든 방을 방문해야 하므로 O(WIDTH \* HEIGHT)이다.

**make\_list()** : 새로운 점수를 리스트에 저장한다. 이 때 최악의 경우 리스트 내의 모든 원소와 크기를 비교해야 하므로 시간 복잡도는 O(n)이다. 이 때 n은 원소의 개수이다. 공간 복잡도 또한 최악의 경우 모든 원소를 참조해야 하므로 O(n)이다.

플레이어의 경로를 출력하는 과정 또한 hint와 그 방식이 비슷하므로 최악의 경우 시간 복잡도와 공간 복잡도 모두 O(WIDTH \* HEIGHT)이다.

**5. 창의적 구현 항목**

첫 번째로, S 버튼을 눌러 플레이어를 생성하여 방향 키를 입력하면 점이 미로 속을 이동하며 정해진 출구로 탈출하도록 설계했다. 이 때 플레이어가 지나간 길을 표시하여 점수에 영향을 주지 않도록 하여 난이도를 조정했다.

힌트의 경우 언제든 켰다 끌 수 있는 방식으로 설계했다. 단, 힌트를 한 번이라도 사용했을 시 미로를 탈출했을 때 힌트를 사용했다는 문구가 출력되도록 했다. 반대로 힌트를 한 번도 사용하지 않았을 경우 미로를 탈출했을 때 힌트를 사용하지 않았다는 문구가 출력되도록 했다.

게임을 클리어하면 플레이어의 총 이동 횟수와 점수가 출력된다. 이 때 점수는 (미로의 크기) - (플레이어가 움직인 총 횟수) 로 설정하였다. 별도의 가중치를 설계하지 않았기 때문에 최단 경로를 기준으로 미로의 크기가 클 수록 높은 점수를 얻기 유리하다. 힌트를 사용하더라도 불이익이 따로 부여되지는 않는다. 해당 점수는 당시 게임의 회 차와 함께 별도의 리스트에 저장되며, 상위 15위 안에 드는 경우 ofWindow에 항시 출력된다.

**6. 프로젝트 실행 결과**

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

프로젝트를 처음 실행했을 때의 ofWindow 화면이다.

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

게임을 진행 중인 화면. 파란색 점은 플레이어의 현위치다.

노란색 선은 플레이어의 이동 경로다. 빨간색 선은 힌트 경로다.

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

미로의 출구를 찾았을 때의 화면. 이동 횟수와 점수, 게임 회 차가 출력되어 있다.

첫 게임이라 랭크 리스트에 처음 등록된 모습을 볼 수 있다.

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

게임을 여러 번 진행한 후의 랭크 리스트 모습이다.

힌트를 사용하지 않았다는 것이 파란색 문구로 표시되어 있다.

**7. 느낀 점과 개선 사항**

완전히 새로운 프로젝트를 창조한 것이 아닌 기존 수업시간에 다뤘던 내용을 확장하여 제작한 프로젝트이기에 독창성이 낮아지는 것은 사실이다. 하지만 퀄리티는 낮더라도 높은 완성도의 프로젝트를 제작하고자 하였고, 수업 시간에 다뤘던 내용들을 응용하며 한 학기 동안 배운 개념들을 다시 복습하며 완벽히 익히고자 하였다.

랭크 리스트의 경우 처음엔 Linked List를 통해 구현하고자 했지만 ofWindow의 화면 크기 상 그렇게 많은 점수들을 출력할 수 있진 않을 것 같아 일반적인 구조체 배열을 이용해 메모리 누수의 가능성 없이 안전하게 구현하였다.

새로운 미로 파일을 불러들일 때 초기화가 제대로 되지 않아 수정하는 데 많은 시간을 썼다. 하지만 수정을 거듭하면서 초기화 선언의 위치를 정확히 확인할 수 있었다. 특히 랭크 리스트를 생성할 때 그 위치가 잘못되어 계속 똑같은 점수가 출력되기도 했다. flag 변수의 위치에 따라 출력의 결과가 완전히 달라지기에 코드 순서를 자세히 확인하며 수정을 반복했다.

수업 시간에 미로를 다루면서 이 프로젝트에 대한 아이디어가 바로 떠올랐다. 단기간 내에 필자의 역량을 감안하여 완성도 있는 프로젝트를 완성했다는 사실에 성취감을 느낀다. 하지만 객관적으로 낮은 퀄리티임은 사실이다. 하지만 이번 프로젝트 활동을 기점으로 필자는 더 많은 프로젝트 경험을 하고자 하는 의욕과 자신감이 생겼다. 이는 복학 이후 밑바닥부터 다시 시작하는 컴퓨터공학과 생활에 중요한 분기점이 될 것이다.

개선 사항으로는 첫 번째로 랭크 리스트에 플레이어가 임의로 이름을 저장할 수 없는 점이다. 처음엔 테트리스 수업 때처럼 게임이 종료된 경우 플레이어의 이름을 점수와 함께 저장하려 했지만, 게임 종료 후 cin을 통해 이름을 입력하려 하자 ofWindow에서 계속 에러가 생기는 바람에 이를 포기하고 대안으로 게임 회 차를 저장하여 구분 짓는 방법을 택했다.

두 번째로 힌트를 사용할 때 플레이어의 위치를 기준으로 출력하고자 하였다. 하지만 플레이어가 지나간 길이 이미 출력되어 있기에 시작점부터 도착점을 연결한 최단 경로를 출력해도 지나간 길에 이어져 출력되므로 가독성이 용이하다 판단해 기각하였다.

세 번째로 프로그램을 처음 실행할 때 미로의 크기를 입력해 랜덤으로 생성된 미로를 출력하는 방법을 구상하였지만, R 키를 누르고 미로의 크기 입력을 시도할 때 Xcode 환경 오류가 계속 발생하여 이를 해결하지 못했다. 또한 게임의 목표가 '여럿이서 경쟁을 통해 함께 즐기기' 이므로 서로 같은 미로를 사용하는 것이 낫다고 판단해 미로 파일 자체를 입력 받는 기존 방식을 택했다.