C++프로그래밍및실습

Digital Maze

진척 보고서 #1

제출일자: 2024.11.17
제출자명: 범지성
제출자학번: 214930

cf) 작성 시점에서 메인 로직에 영향을 주지 않는 코드들은 포함되지 않았을 수 있음

1. 프로젝트 목표

1) 배경

퍼즐 게임을 즐겨했기 때문에 간단한 게임에 변형을 주어 비슷한 새로운 게임을 제작하고 싶었음. C++는 내장 GUI 라이브러리가 없어 복잡한 형태의 화면이나 다양한 형태의 이벤트 처리가 어려움. 상하좌우로 움직여 상자를 밀어 모든 상자를 목표물 위에 위치시키는 게임 SOKOBAN은 비교적 형태가 단순하고 이벤트 수도 많지 않았기 때문에 GUI 프로그래밍 없이 CLI 상에서 깔끔한 구현이 용의할 것이라고 판단해 SOKOBAN의 일부 규칙을 채용한 CLI 기반 퍼즐 게임을 기획함.

2) 프로젝트 목표

이동횟수가 제한되어 있고 목표지점에 도달하는 퍼즐형 게임을 만드는 것을 목표로함.

3) 차별점

게임의 기본적인 형식은 1982년 출시된 "倉庫番(SOKOBAN)"을 참고했고 일부 장애물들은 및 규칙은 이와 비슷한 형식의 퍼즐게임인 "Helltaker"를 모방한다. (플레이어가 한 칸씩 이동, 이동 횟수 제한, 목표지점 도달). 참고한 게임과의 차별점은 다음과 같다

참고작과 달리 게임에 스토리 부여 X

기존 장애물 인부 삭제. 가시 장애물(밟으면 이동횟수 가능 횟수 2회 감소)를 삭제. 레이저 장애물(접촉 즉시 게임 실패) 삭제

새로운 엔티티 추가. 잠긴 벽과 이를 해방하는 열쇠를 추가. 플레이어의 이동에 따라 열리고 닫히기를 반복하는 공간 이동 포탈 추가

튜토리얼 스테이지가 따로 없고 시작화면에서 조작법을 확인할 수 있는 메뉴를 추가한다.

2. 기능 계획

1) 기능 1 시작 화면

- 설명: 게임의 이름과 스테이지 선택창으로 이동, 게임 설명서로 이동하는 선택지들이 사용자에게 표시되는 화면을 만든다.

(1) 세부 기능 1: 게임 설명서 창

- 설명: 시작화면에서 게임 설명서 선택 시, 아무 키나 누르면 다시 시작화면으로 돌아오는 게임 설명서 창으로 이동한다. 캐릭터 조작법과 엔티티 소개를 담는다.

(2) 세부 기능 2: 선택지 이동 및 확정

- 설명: 상/하 방향키 또는 'W', 'S'키 입력으로 선택 중인 옵션을 바꾸고 Enter를 눌러 선택을 확정하는 기능을 넣는다.

2) 기능 2 스테이지 선택창

- 설명: 전체 스테이지를 띄우고 키보드 입력으로 선택지를 이동하며 선택지를 고르고 Enter로 스테이지를 확정하게 한다. 직전 스테이지를 클리어하지 않으면 다음 스테이지는 잠긴 상태로 둔다. 시작화면으로 돌아가는 키도 구현한다.

(1) 세부 기능 1: 스테이지 로더(Loader)

- 설명: 스테이지 정보를 담은 변수로 게임을 실행하면 스테이지가 1회용이 되고 변수마다 로직을 붙여줘야 하기 때문에 게임 로직에 사용할 변수를 만들어 거기에 스테이지 정보를 불러오는 것이 좋다. 따라서 사용자의 선택에 따라 스테이지가 로드되는 로직을 만든다.

(2) 세부 기능 2: 선택지 이동 및 확정

- 설명: 상/하/좌/우 방향키 또는 'W', 'A', 'S', 'D'키 입력으로 선택 중인 옵션을 바꾸고 Enter를 눌러 선택을 확정하는 기능을 넣는다.

3) 기능 3 게임 스테이지 맵

- 설명: 실제 플레이할 게임화면을 구현한다. 현재 스테이지 레벨, 스테이지 맵, 각종 엔티티, 남은 이동 횟수 등을 표기하고, 사용자의 행동에 따라 생긴 변화를 즉시 보여준다.

(1) 세부 기능 1: 스테이지 종료 시 상호작용창

- 설명: 게임 오버 시는 다시하기, 스테이지 선택창으로 이동 선택지를 띄우고 스테이지 클리어 시 두 선택지에 더해 다음 스테이지 플레이하기 선택지를 추가한다.

(2) 세부 기능 2: 스테이지 출력

- 설명: 시작 당시 스테이지를 출력하고 플레이어가 행동을 취할 때마다 현재 출력된 화면을 모두 지운 후 변화된 스테이지와 잔여 이동 횟수를 갱신해서 출력한다.

4) 기능 4 클리어 현황 저장소

- 설명: 게임을 종료한 후 다시 실행했을 때 클리어했던 스테이지들을 바로 플레이할 수 있도록 별도의 파일을 조작해 클리어한 스테이지 레벨을 저장하도록 한다.

5) 기능 5 플레이어 엔티티

- 설명: 정해진 스테이지 맵 안에서 한 번에 한 칸씩 이동할 수 있고 플레이어의 키보드 입력에 따라 동작한다.

(1) 세부 기능 1: 이동 및 행동

- 설명: 상/하/좌/우 방향키 또는 'W', 'A', 'S', 'D'키 입력으로 플레이어가 맵 상에서 이동, 장애물 부수기, 장애물 밀기, 장애물 치기 등의 행동을 한다. 이동 또는 행동을 하면 잔여 이동 횟수가 1 줄어든다.

(2) 세부 기능 2: 행위 유효성 체크

- 설명: 플레이어가 이동하려는 방향에 장애물이 있는지 체크하고 장애물 유무에 따라 플레이어의 행위 여부를 결정한다.

6) 기능 6 게임오버 조건 검사

- 설명: 플레이어의 잔여 이동횟수가 0인 상태에서 플레이어가 이동을 시도하면 게임 오버된다.

7) 기능 7 기타 엔티티

- 설명: 맵 위에 존재하는 요소들로 플레이어가 해당 엔티티가 존재하는 좌표로 이동을 시도하면 각기 다른 상호작용이 발생한다.

(1) 세부 기능 1: 벽

- 설명: 모든 스테이지에 존재한다. 스테이지 맵이 생성될 때부터 위치가 고정되는 요소.

플레이어가 해당 위치로 이동 시도 시, 아무 일도 발생하지 않으며 이동횟수도 줄어들지 않는다.

(2) 세부 기능 2: 밀 수 있는 장애물

- 설명: 플레이어가 해당 장애물 위치로 이동 시도 시 플레이어의 위치는 변하지 않고, 장애물이 플레이어가 이동하려던 방향으로 한 칸 밀리고, 이동 횟수가 1회 줄어든다. 추가로 장애물이 벽에 가로막힌 경우에도 이동 횟수가 1회 줄어든다.

(3) 세부 기능 3: 부술 수 있는 장애물

- 설명: 해당 장애물의 위치로 플레이어가 이동을 시도하면 플레이어의 위치는 변하지 않고 해당 장애물이 사라지며, 이동 횟수가 1회 줄어든다.

(4) 세부 기능 4: 열쇠와 잠긴 벽

- 설명: 잠긴 벽은 초기에 (1)"벽"처럼 동작한다. 맵에 존재하는 모든 열쇠 엔티티와 접촉 시 열쇠 엔티티와 잠긴 벽 엔티티가 모두 사라진다.

(5) 세부 기능 5: 포탈

- 설명: 스테이지 위에 두 개의 포탈이 존재하고 플레이어가 1회 이동함에 따라 포탈이 활성화되고 비활성화되기를 번갈아 반복한다. 포탈이 활성화되는 동시에 플레이어가 포탈 위치로 이동하면 나머지 한 쪽 포탈이 있던 위치로 이동되며 포탈은 소멸한다. (*주의 * 비활성화된 포탈 위에 플레이어가 지나갈 때 포탈이 소멸되지 않도록 로직을 만든다.)

(6) 세부 기능 6: 목적지 엔티티

- 설명: 모든 스테이지에 존재한다. 플레이어의 잔여 이동횟수가 0 이상인 상태에서 해당 엔티티 위치로 이동 시 스테이지 클리어. 다음 스테이지가 잠겼다면 다음 스테이지 해금

8) 기능 8 실행취소

- 설명: 게임을 진행하던 중 의도하지 않았던 버튼을 눌러서 처음부터 다시 플레이 해야 되는 경우가 생길 수 있기 때문에 게임 플레이 중 "CTRL + z"를 누르면 이전 상태로 되돌리 수 있는 기능을 추가한다.

9) 기능 9 세부 스테이지

(1) Stage 1

- 설명: 벽 엔티티들과 하나의 목적지 엔티티로 이루어진 미로형 스테이지

(2) Stage 2

- 설명: 밀 수 있는 장애물과 부술 수 있는 장애물, 벽 엔티티로 목적지 엔티티로 향하는 길을 방해하는 퍼즐형 스테이지

(3) Stage 3

- 설명: 밀 수 있는 장애물, 부술 수 있는 장애물, 열쇠와 잠긴 벽, 벽 엔티티로 목적지 엔티티로 향하는 길을 방해하는 퍼즐형 스테이지

(4) Stage 4

- 설명: 밀 수 있는 장애물, 부술 수 있는 장애물, 열쇠와 잠긴 벽, 포탈, 벽 엔티티로 목적지 엔티티로 향하는 길을 방해하는 퍼즐형 스테이지

3. 진척사항

1) 클래스 및 함수 구현

(1) Frame

- 설명:

화면 디자인 및 출력에 관여하는 함수들의 집합이다.

- public 함수:

(1) void printLine(int num)

-매개변수

① num: 출력할 문자의 개수

- 기능:

num의 수 만큼 '='를 출력한 후 줄을 바꾼다.

- 코드

```
3   void Frame::printLine(int num){ // print '=' 'num' times
4       for(int i = 0; i < num; i++)
5           cout << "=";
6       cout << endl;
7   }
```

(2) void printTitle(int num, string name)

-매개변수

① num: 제목에 할당될 가로 폭 길이

② name: 출력할 제목 텍스트

- 기능:

printLine()을 호출한 후, 할당된 범위의 중앙에 name을 출력한 후, 다시 printLine()을 호출한다.

- 코드

```
 8   void Frame::printTitle(int num, string name){
 9       printLine(num);
10       cout.width((num + name.length()) / 2);
11       cout << name << endl;
12       printLine(num);
13   }
```

(3) void printConfirmAlert()

- 기능:

Enter키로 선택을 확정하라는 안내 문구를 출력한다.

- 코드

```
14   void Frame::printConfirmAlert(){ // Notify user to press Enter
15       cout << endl << endl << "Enter : Confirm";
16   }
```

(4) void printOption(string name, bool selected, int num)

-매개변수

① name: 출력할 옵션 텍스트

② selected: 해당 옵션이 선택됐는지 여부

③ num: 출력물 앞 공백 크기

- 기능:

선택된 옵션이면 앞에 '*'을 붙여서 아니면 그냥 앞을 공백으로 두고(여백은 num만큼) name을 출력한다.

- 코드

```
17   void Frame::printOption(string s, bool selected, int num){ // Print Option line
18       cout << endl;
19       cout.width(num);
20       if (selected == true){
21           cout << "* " << s << endl;
22       }
23       else if(selected == false){
24           cout << "  " << s << endl;
25       }
26   }
```

(5) void printButtonLine(int start, int selected)

-매개변수

① start: 출력할 두 버튼 중 첫 버튼의 번호

② selected: 선택된 버튼의 번호

- 기능:

start번 버튼과 그 다음 순서 버튼을 같은 줄에 출력하고, selected에 해당하는 번호의 버튼이면 버튼 앞에 '*'이 출력된다.

- 코드

```
28  void Frame::printButtonLine(int start, int selected){ // Print two button on stage selection page
29      cout << endl;
30      for(int i = 0; i < 2; i++){
31          cout.width(8);
32          if(start + i == selected) // print * for selected option
33           cout << "* ";
34          else
35              cout << "";
36      }
37      cout << endl;
38      for(int i = 0; i < 2; i++){
39          cout.width(5);
40          cout << "" << "---";
41      }
42      cout << endl;
43      for(int i = 0; i < 2; i++){
44          cout.width(5);
45          cout << "" << "|" << start + i << "|";
46      }
47      cout << endl;
48      for(int i = 0; i < 2; i++){
49          cout.width(5);
50          cout << "" << "---" ;
51      }
52      cout << endl;
53  }
```

(6) void printStage(char board[][12], int size)

-매개변수

① board: 출력할 12 x 12 크기의 게임판

② size: 출력할 게임판의 가로, 세로 길이

- 기능:

게임판을 입력 받아 이를 그대로 출력해준다.

- 코드

```
55  void Frame::printStage(char board[][12], int size){
56      for(int i = 0; i < size; i++){
57          for(int j = 0; j < size; j++){
58              cout << board[i][j];
59          }
60          cout << endl;
61      }
62  }
```

(2) KeyListener

- *적용된 배운 내용*

정적 멤버를 선언해 객체 생성 없이 클래스만으로 바로 함수와 멤버에 접근한다.

- 설명:

일반적으로 생각하는 listener와 달리 키 이벤트에 따라 특정 작업을 handling하지는 않는다. 보유한 함수를 통해 특정 상황에서 사용자에게 특정 키 입력만 허용한다.

- static 상수(해당 키에 대응하는 아스키 코드 정수값):

① UP: 위쪽 방향키

② DOWN: 아래쪽 방향키

③ LEFT: 왼쪽 방향키

④ RIGHT: 오른쪽 방향키

⑤ ENTER: 엔터키

⑥ CTRL_Z: Ctrl + z 키

- public static 함수:

(1) int getPlayerKey()

- 기능:

버퍼 없이 바로 키 하나를 입력 받고, 해당 값이 상/좌/하/우 방향키 또는 w/a/s/d키 일 때 UP/LEFT/DOWN/RIGHT를 반환하고, Ctrl + Z키 일 때는 CTRL_Z를 반환하며 나머지 키가 입력될 때는 앞서 나온 키 중 하나가 입력될 때까지 다시 키 입력 받기를 시도한다.

- 코드

```
int KeyListener::getPlayerKey() { // 4 Direction, Ctrl + Z
    while(1){
        // get key input until meet up, down or ENTER
        int move = _getch();
        switch(move){
            case LEFT:
            case RIGHT:
            case UP:
            case DOWN:
            case CTRL_Z:
                return move;
            case 'w':
                return UP;
            case 'a':
                return LEFT;
            case 'd':
                return RIGHT;
            case 's':
                return DOWN;

        }
    }
```

(2) int titleKey()

- 기능:

버퍼 없이 바로 키 하나를 입력 받고, 해당 값이 상/하 방향키 또는 w/s키 일 때 UP/DOWN를 반환하고, Enter키 일 때는 ENTER를 반환하며 나머지 키가 입력될 때는 앞서 나온 키 중 하나가 입력될 때까지 다시 키 입력 받기를 시도한다.

- 코드

```
int KeyListener::titleKey() { // Enable Enter vertical direction key, and 'w', 's'
    while(1){
        // get key input until meet up, down or ENTER
        int move = _getch();
        switch(move){
            case ENTER:
            case UP:
            case DOWN:
                return move;
            case 'w': // Handle alphabet as direction key
                return UP;
            case 's':
                return DOWN;
        }
    }

}
```

(3) int stageSelectionKey()

- 기능:

버퍼 없이 바로 키 하나를 입력 받고, 해당 값이 상/좌/하/우 방향키 또는 w/a/s/d키 일 때 UP/LEFT/DOWN/RIGHT를 반환하고, Enter키 일 때는 ENTER를 반환하며 나머지 키가 입력될 때는 앞서 나온 키 중 하나가 입력될 때까지 다시 키 입력 받기를 시도한다.

- 코드

```
int KeyListener::stageSelectionKey() { // Enable Enter, 4 Direction
    while(1){
        // get key input until get 4 direction or Enter
        int move = _getch();
        switch(move){
            case ENTER:
            case LEFT:
            case RIGHT:
            case UP:
            case DOWN:
                return move;
            case 'w':
                return UP;
            case 'a':
                return LEFT;
            case 'd':
                return RIGHT;
            case 's':
                return DOWN;

        }
    }

}
```

(4) int enableEnter()

- 기능:

버퍼 없이 바로 키 하나를 입력 받고, 해당 값이 Enter키 일 때 ENTER를 반환하며 나머지 키가 입력될 때는 앞서 나온 키 중 하나가 입력될 때까지 다시 키 입력 받기를 시도한다.

- 코드

```
int KeyListener::enableEnter(){ // only Enter Allowed
    while(1){
        int move = _getch();
        if (move == ENTER) return move;
    }
}
```

(3) Moveable

- *적용된 배운 내용*

클래스 함수 작성 시, this 키워드가 해당 클래스의 인스턴스 자기 자신의 포인터 역할을 한다.

- 설명:

게임 플레이 중에 위치가 바뀔 수 있는 엔티티의 공통 분모를 가진 상위 클래스

- private 필드:

① x: 엔티티 x좌표(열 좌표)

② y: 엔티티 y좌표(행 좌표)

- 생성자:

(1) Moveable()

- 기능:

기본 생성자, x, y를 1로 세팅한다

- 코드

```
8           Moveable(){ x = 1; y = 1;};
```

(2) Moveable(int x, int y)

-기능:

x, y 값을 주어진 인자로 초기화하면서 인스턴스를 생성한다.

-코드

```
22  void Moveable::setLocation(int x, int y){
23      this->x = x;
24      this->y = y;
25  }
```

- public 함수:

(1) void move(int direction)

-매개변수

① direction: KeyListener의 static 상수 중 방향에 관련된 상수 값

- 기능:

direction에 대응하는 방향으로 인스턴스의 x, y 값을 변환한다.

- 코드

```
6  void Moveable::move(int direction){ // entity change position according to key stroke
7      if(direction == KeyListener::UP){ // move upper
8          y--;
9      }
10     else if(direction == KeyListener::DOWN){ // move lower
11         y++;
12     }
13     else if (direction == KeyListener::LEFT){ // move left
14         x--;
15     }
16     else if (direction == KeyListener::RIGHT){ // move left
17         x++;
18     }
19 }
```

(2) int getX()

- 기능:

  인스턴스의 x 반환

- 코드

```
20    int Moveable::getX(){ return x; }
```

(3) int getY()

- 기능:

  인스턴스의 y 반환

- 코드

```
21    int Moveable::getY(){ return y; }
```

(4) void setLocation(int x, int y)

-매개변수

① x: 설정할 x 좌표

② y: 설정할 y 좌표

- 기능:

  인스턴스의 x, y값을 매개변수 x, y값으로 바꾼다.

- 코드

```
22  void Moveable::setLocation(int x, int y){
23      this->x = x;
24      this->y = y;
25  }
```

(4) Player

- 설명:

게임 플레이를 할 유저 인스턴스를 생성하기 위한 클래스다. Moveable을 상속하고 이가 가진 함수 중 변형된 것은 없다.

- private 필드:

① stamina: 유저의 잔여 이동 가능 횟수

- public static 상수:

① PLAYER: 유저의 스테이지 맵 상 기호, 'P'

- 생성자:

(1) Player()

- 기능:

Player 인스턴스 기본 생성자, stamina가 0인 인스턴스를 생성한다.

- 코드

```
12      Player(){ stamina = 0; } // Default for make at Least Trash Instance
```

(2) Player(int s)

-기능:

stamina값이 s인 Player 인스턴스를 생성한다.

-코드

```
13      Player(int s){ stamina = s; }
```

- public 함수:

(1) bool checkAlive()

- 기능:

  인스턴스의 stamina가 0 이하인지 체크해서 0이하면 false 그 외 true를 반환한다.

- 코드

```
23  bool Player::checkAlive(){ // check remaining stamina and if lower than 0 return false(dead)
24      if(stamina <= 0){
25          return false;
26      }
27      else
28          return true;
29  }
```

(2) int getStamina()

- 기능:

  인스턴스의 stamina 반환

- 코드

```
30  int Player::getStamina() { return stamina;}
```

(3) void setStamina(int n)

-매개변수

① n: 설정할 stamina 값

- 기능:

  객체의 stamina 값을 n으로 바꾼다.

- 코드

```
31  void Player::setStamina(int n){ stamina = n;}
```

(4) void decreaseStamina()

- 기능:

  인스턴스의 stamina가 1 감소한다.

- 코드

```
38  void Player::decreaseStamina(){ stamina--; } //
```

(5) StageNode

- 설명:

스테이지 정보, 플레이어 좌표, 플레이어 체력 정보를 담은 인스턴스를 제작하기 위한 클래스다. 게임 플레이 중 실행취소 기능을 구현하기 위한 노드 생성용 클래스

- public 필드:

① x: 플레이어 x좌표

② y: 플레이어 y좌표

③ hp: 플레이어 스테미나

④ stage: 스테이지 맵

- public static 필드:

① size: 스테이지 맵 가로, 세로 폭

- 생성자

(1) StageNode(char s[size][size], Player u)

-매개변수

① s: 스테이지 맵 배열

② u: Player 인스턴스

- 기능:

s를 stage필드에 복제하고, u의 x, y, stamina 값을 각각 생성할 인스턴스의 x, y, hp로 초기화 한다.

- 코드

```
66  // For Ctrl + Z
67  StageNode::StageNode(char s[size][size], Player u){
68      copy(&s[0][0], &s[0][0] + (size*size), &stage[0][0]);
69      x = u.getX();
70      y = u.getY();
71      hp = u.getStamina();
72  }
```

(6) Stage

- 설명:

Player 인스턴스를 가지고 stage map 생성, game play 기능을 가진 클래스다.

- private 필드:

① user: Player 인스턴스, 해당 스테이지를 플레이하는 유저

② stage: 현재 플레이 중인 스테이지 맵의 상태

③ stack: stack 인스턴스, 플레이어 행동 직전 상태를 담은 StageNode 객체를 원소로 하는 스택

- 생성자

(1) Stage(int stageFlag)

- 매개변수:

① stageFlag: 스테이지 번호

- 설명

선택된 스테이지 번호에 따라, 생성할 Player의 stamina를 정하고, stage를 달리한다.

- 코드

```
82  Stage::Stage(int stageFlag){ // later according to flag, will build different stage map
83      switch(stageFlag){
84          case 1:
85          case 2:
86              user = Player{35}; // Least movement for stage1 is 29, for stage2 32
87              break;
88          default: // for dummy stage
89              user = Player{100};
90              break;
91      }
92      buildStage(stageFlag);
93  };
```

- private 함수:

(1) void buildWall() // Dummy Stage 생성용 추후 제거 예정

- 기능:

stage의 모든 원소를 ' '(공백 문자)로 바꾼 후, stage의 행 또는 열 인덱스가 0 또는

size – 1인 곳을 벽 엔티티 기호로 바꾼다.

- 코드

```
12  void Stage::buildWall(){ // Every Stage has same border
13      for(int i = 0; i < size; i++){ // Fill with empty space
14          for(int j = 0; j < size; j++){
15              stage[i][j] = ' ';
16          }
17      }
18      for (int i = 0; i < size; i++){  // make border
19          stage[i][0] = '#';
20          stage[0][i] = '#';
21          stage[i][size - 1] = '#';
22          stage[size - 1][i] = '#';
23      }
24  }
```

(2) void buildDummyStage() // 모든 스테이지 생성 완료 후 제거 예정

- 기능:

buildWall()을 호출한 후 (행, 열)일 때 (1, 1)을 플레이어 기호로, (size – 2, size – 2를) 목적지 기호로 전환한다.

- 코드

```
45  void Stage::buildDummyStage(){
46      buildWall();
47      stage[size - 1][size - 1] = '#';
48      stage[1][1] = 'P'; // Player
49      stage[10][10] = '@'; // Goal
50  }
```

(3) void buildStage(int stageFlag)

-매개변수

① stageFlag: 선택된 스테이지 번호

- 기능:

stageFlag에 따라 미리 만들어진 스테이지 맵을 stage 필드에 복사한다. 스테이지에 반복적인 패턴은 없어 각 스테이지를 미리 만들어 지역변수에 저장했다. (현재 Stage1과 Dummy Stage만 구현)

- 코드

```
void Stage::buildStage(int stageFlag){ // Load stage according to flag
    // since entities position has no pattern better type whole stage
    if (stageFlag == 1){ // Obstacle - Wall only
        char stage1[size][size] = {{'#','#','#','#','#','#','#','#','#','#','#','#'},
                                   {'#','P',' ',' ',' ',' ',' ','#',' ',' ','@','#'},
                                   {'#',' ','#','#','#','#','#',' ',' ','#','#','#'},
                                   {'#',' ','#',' ',' ',' ',' ','#',' ','#','#',' ','#'},
                                   {'#',' ',' ',' ','#',' ','#',' ',' ',' ',' ','#'},
                                   {'#',' ','#','#','#',' ','#',' ',' ','#','#','#'},
                                   {'#',' ','#',' ',' ',' ','#',' ',' ','#',' ','#'},
                                   {'#',' ','#',' ','#','#','#',' ','#','#',' ','#'},
                                   {'#',' ','#',' ',' ',' ',' ',' ',' ',' ',' ','#'},
                                   {'#',' ','#','#','#','#','#',' ','#','#','#','#'},
                                   {'#',' ',' ',' ',' ',' ',' ','#',' ','#',' ','#'},
                                   {'#','#','#','#','#','#','#','#','#','#','#','#'}
                                   };
        copy(&stage1[0][0], &stage1[0][0] + (size*size),&stage[0][0]);
    }
    else if(stageFlag == 2){
        char stage2[size][size] = {{'#','#','#','#','#','#','#','#','#','#','#','#'},
                                   {'#','P','B','#',' ',' ','#',' ',' ',' ',' ','#'},
                                   {'#','#',' ','#',' ',' ','#',' ','#','#','#','#'},
                                   {'#',' ',' ','#','#','#','#',' ','#',' ',' ','#'},
                                   {'#',' ',' ','B','O',' ',' ',' ',' ',' ',' ','#'},
                                   {'#','#','B','#','#','#','#',' ','#','#',' ','#'},
                                   {'#','#','O','#',' ',' ',' ',' ','#','#',' ','#'},
                                   {'#','#',' ','#','#','#','#','#',' ','#','B','#'},
                                   {'#','#',' ','#',' ',' ','#',' ',' ','O','O','#'},
                                   {'#',' ',' ',' ','O',' ',' ',' ','O','B','#','#'},
                                   {'#','#',' ','#',' ',' ','#','O',' ',' ','@','#'},
                                   {'#','#','#','#','#','#','#','#','#','#','#','#'}
                                   };
        copy(&stage2[0][0], &stage2[0][0] + (size*size),&stage[0][0]);
    }
    else buildDummyStage();
}
```

(4) void changeBoard(int action)

-매개변수

① action: KeyListener의 방향키 관련 상수값 중 하나

- 기능:

유저의 움직임에 따라 stage 상의 유저 위치를 바꾼다. (추후 스테이지 업데이트로 장애물 추가에 따라 로직이 변경될 수 있음)

- 코드

```
void Stage::changeBoard(int action){ // activate when player success moving
    // change position in board, user's location, and decrease statmina
    stage[user.getY()][user.getX()] = ' ';
    user.move(action);
    stage[user.getY()][user.getX()] = 'P';
    user.decreaseStamina();
}
```

- public 함수:

(1) int play(Frame f, int stageFlag)

-매개변수

① f: Frame 인스턴스

② stageFlag: 선택된 스테이지 레벨

- 기능:

스테이지를 플레이한다. 스테이지 레벨, 스테이지 맵 현황, 남은 스테미나가 출력된다. 플레이어는 이동 또는 실행 취소를 할 수 있다. 플레이어가 이동을 할 때마다 행동 직전의 상태를 임시 저장한다. 실행 취소를 하면 저장했던 직전 상태를 불러온다. 행동이 끝나면 해당 행동을 반영해서 화면을 재출력한다. 남은 이동횟수가 0이 됐을 때 이동을 시도하면 Game over (0 반환), 이동횟수 안에 목적지에 도달하면 Game Clear (1 반환), 플레이어 행동에 대한 자세한 서술은 기능 구현에서 한다.

- 코드

```
int Stage::play(Frame f, int stageFlag){ // Default Logic of game play, might be changed according to obstacles
    int clearFlag = 0; // if flag is 1 clear
    char encounter; // Entity that is which is on
    string title[5] = {"Stage0", "Stage1", "Stage2", "Stage3", "Stage4"};
    while(true){
        // refresh displayed screen
        system("cls");
        f.printTitle(12, title[stageFlag]);
        cout << endl;
        f.printStage(stage, size);
        cout << "Stamina : " << user.getStamina() << endl;

        if(encounter == '@'){ // if reaching goal clear!
            clearFlag = 1;
            break;
        }

        int action = KeyListener::getPlayerKey(); // get user action
        if(action == KeyListener::CTRL_Z){ // return to previous state if Ctrl + Z pressed
            if(stack.empty() == false){
                StageNode tmp = stack.top();
                user.setLocation(tmp.x, tmp.y);
                user.setStamina(tmp.hp);
                copy(&tmp.stage[0][0], &tmp.stage[0][0] + (size*size), &stage[0][0]);
                stack.pop();
                continue;
            }
        }
        if(!user.checkAlive()){ // if user try to move when stamina is 0 or less game over
            break;
        }
        // Change stage according to player's movement
        if(action == KeyListener::UP){
            encounter = stage[user.getY() - 1][user.getX()];
            if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
            if(encounter == ' ' || encounter == '@'){
                stack.push(StageNode{stage, user}); // save state before change
                changeBoard(action, user);
            }
            else if(encounter == 'B'){
                user.decreaseStamina();
                stage[user.getY() - 1][user.getX()] = ' ';
            }
        }
```

```
138         else if(action == KeyListener::DOWN){
139             encounter = stage[user.getY() + 1][user.getX()];
140             if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
141             if(encounter == ' ' || encounter == '@'){
142                 changeBoard(action, user);
143             }
144             else if(encounter == 'B'){
145                 user.decreaseStamina();
146                 stage[user.getY() + 1][user.getX()] = ' ';
147             }
148         }
149         else if(action == KeyListener::LEFT){
150             encounter = stage[user.getY()][user.getX() - 1];
151             if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
152             if(encounter == ' ' || encounter == '@'){
153                 changeBoard(action, user);
154             }
155             else if(encounter == 'B'){
156                 user.decreaseStamina();
157                 stage[user.getY()][user.getX() - 1] = ' ';
158             }
159         }
160         else if(action == KeyListener::RIGHT){
161             encounter = stage[user.getY()][user.getX() + 1];
162             if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
163             if(encounter == ' ' || encounter == '@'){
164                 changeBoard(action, user);
165             }
166             else if(encounter == 'B'){
167                 user.decreaseStamina();
168                 stage[user.getY()][user.getX() + 1] = ' ';
169             }
170         }
171     }
172     while(!stack.empty()){stack.pop();}
173     return clearFlag;
174 }
```

2) 기능 구현

(1) 시작화면

- 변수

① selected: 시작화면에 제공된 선택지 중 선택된 것의 인덱스를 true로 하는 변수

② move: 사용자로부터 받은 입력을 저장하는 변수

③ option: 현재 선택된 옵션을 가리키는 포인터

④ pageFlag: main함수에서 실행할 페이지를 가리키는 변수

⑤ printer: 화면 구성을 위한 Frame 인스턴스

- 설명:

해당 코드는 사용자 입력 한 번이 한 iteration인 반복문 속에 있다. system("cls")로 현재 화면에 출력된 문자를 모두 지우고 사용자 동작에 따라 변화된 상태를 새로 출력한다.

게임의 제목, 스테이지 선택하기, 게임 설명서, 게임 종료하기 옵션이 있음을 알리기 위해 출력되고, 선택 확정을 Enter로 수행함을 알리는 문구가 출력된다. selected[k]가 true인 경우 해당 옵션은 앞에 '*' 문자를 붙여 출력된다.

- 적용된 배운 내용

출력기 printer 인스턴스는 새롭게 정의한 Frame 클래스의 인스턴스다. Frame 클래스 내에 정의한 public 함수 printTitle(), printOption(), printConfirmAlert(), printLine()을 이용했다. 자세한 로직은 Class 구현란 참조

- 코드 스크린샷

```
25      while(pageFlag != CENTINEL){
26          // Display Main Page
27          if(pageFlag == 0){
28              const int OPTION_SIZE = 3;
29              int option = 0; // option pointer, this will get 0~2
30              bool selected[OPTION_SIZE] = {true, false, false}; // selected option's value is true
31              while(true){
```

```
32                  // Display Changed State
33                  system("cls");
34                  printer.printTitle(26, "Digital Maze");
35                  printer.printOption("Select Stage", selected[0], 7);
36                  printer.printOption("Instructions", selected[1], 7);
37                  printer.printOption("    Exit    ", selected[2], 7);
38                  printer.printConfirmAlert();
39                  cout << endl << endl;
40                  printer.printLine(26);
```

```
42                  int move = KeyListener::titleKey(); // determine action
43                  if(move == KeyListener::ENTER) break; // Selection Determined
44                  // Change Selection According to Input
45                  // if not Confirming the move is option changing
46                  selected[option] = false;
47                  if(move == KeyListener::UP){
48                      if(option == 0)
49                          selected[option = 2] = true;
50                      else
51                          selected[--option] = true;
52                  }
53                  else if(move == KeyListener::DOWN){
54                      if(option == 2)
55                          selected[option = 0] = true;
56                      else
57                          selected[++option] = true;
58                  }
59              }
60              if(option == 2)
61                  pageFlag = CENTINEL; // Terminate Game
62              else
63                  pageFlag = option + 1; // Move onto Selected Page
64          }
```

-세부 기능

1) 설명서 페이지

- 설명:

사용자에게 해당 게임의 조작법, 요소들에 대해 소개하는 페이지를 출력하고 유저의 Enter 입력을 대기한다. Enter 입력이 발생하면 Title로 돌아간다.

-코드

```
106         else if(pageFlag == 2){
107             system("cls");
108             // most code in this line will goto Frame Class
109             // Print title
110             printer.printTitle(58, "Instructions");
111             // How to Play
112             cout << "* Movement" << endl;
113             cout << "   W          ^" << endl;
114             cout << " A   D  or  <   > : Move to the Direction Once" << endl;
115             cout << "   S          v" << endl;
116             cout << endl << "'Crtl + Z' : Undo" << endl;
117             cout << endl << "* Entities" << endl;
118             cout << "'P' : Player" << endl;
119             cout << "'#' : Wall - Can't break, push or kick. It's fixed" << endl;
120             cout << "'L' : Lock - Can't break, push or kick. Might be disposable" << endl;
121             cout << "'K' : Key - Unlock the 'Lock' Entity" << endl;
122             cout << "'#' : Wall - Can't break, push or kick. It's fixed" << endl;
123             cout << "'W' : Warp Portal - Two object exist." << endl;
124             cout << "                   Touching one, warp to the other" << endl;
125             cout << "'B' : Breakable Box - You can break it by kicking it once" << endl;
126             cout << "'O' : Pushable Rock - You can push it." << endl;
127             cout << "                     You can kick it though it's blocked" << endl << endl;
128 
129             cout.width(58);
130             cout << "Enter : Return to Title...";
131 
132             KeyListener::enableEnter(); // Loof until Pressing Enter
133             pageFlag = 0; // Return to Main Page
134         }
```

(2) 스테이지 선택창

- 변수

① selected: 사용자의 입력을 저장한 변수

② pageFlag: main함수에서 실행할 페이지를 가리키는 변수

③ printer: 화면 구성을 위한 Frame 인스턴스

④ stageFlag: 현재 선택된 스테이지 번호를 저장하는 변수

- 설명:

페이지의 제목이 출력된 후, 네 개의 스테이지를 선택할 수 있는 선택지와 시작화면(Title)로 돌아갈 수 있는 선택지가 주어지며, 해당 선택지는 상/좌/하/우 방향키 또는

w/a/s/d 키를 눌러 바꿀 수 있고 Enter로 선택을 확정할 수 있다.

- 적용된 배운 내용

플레이어가 적절한 행동을 취할 때까지 해당 페이지를 계속 실행할 수 있도록 while(true)로 무한 루프를 만들고 if문으로 조건부로 while을 탈출한다.

- 코드 스크린샷

```
65          // Display Selection Page
66          else if(pageFlag == 1){
67              while(true){
68                  system("cls");
69                  // Print title
70                  printer.printTitle(24, "Select Stage");
71                  // button
72                  printer.printButtonLine(1, stageFlag);
73                  printer.printButtonLine(3, stageFlag);
74                  // change code under
75                  printer.printOption("Back to Title", !stageFlag, 2);
76                  printer.printConfirmAlert();
77                  // cout.width(25);
78                  // cout << "Enter : Confirm";
79
```

```
80                  int selected = KeyListener::stageSelectionKey();
81                  if(selected == 13) break; // confirm selection
82                  else if(stageFlag == 0) stageFlag = 3; // When pointer is on back to title, next selection is always stage 3
83                  else if(selected == KeyListener::UP){
84                      if(stageFlag / 3 == 0) stageFlag += 2;
85                      else stageFlag -= 2;
86                  }
87                  else if(selected == KeyListener::DOWN){
88                      if(stageFlag == 3) stageFlag = 0;
89                      else if(stageFlag == 4) stageFlag -= 2;
90                      else stageFlag += 2;
91                  }
92                  else if(selected == KeyListener::LEFT){
93                      if(stageFlag % 2 == 1) stageFlag++;
94                      else stageFlag--;
95                  }
96                  else if(selected == KeyListener::RIGHT){
97                      if(stageFlag % 2 == 0) stageFlag--;
98                      else stageFlag++;
99                  }
100             }
101             // If stage is selected go onto page 3, else go onto page 0
102             if(stageFlag == 0)
103                 pageFlag = 0;
104             else
105                 pageFlag = 3;
106         }
```

(3) 스테이지 맵

- 변수

① pageFlag: main함수에서 실행할 페이지를 가리키는 변수

② printer: 화면 구성을 위한 Frame 인스턴스

③ stageFlag: 현재 선택된 스테이지 번호를 저장하는 변수

④ gameresult: 게임 결과값을 저장하는 변수

⑤ choice: 게임이 끝난 후 나온 선택지 중 선택된 것을 표시하는 변수

⑥ pointer: choice에서 선택된 옵션의 인덱스 포인터 변수

⑦ move: 유저의 입력을 저장하는 변수

- 설명:

stageFlag에 따라 Stage 인스턴스를 생성하고 인스턴스의 play() 함수로 게임을 플레이한다. 플레이 결과에 따라 실패 또는 성공 여부를 알리고, 실패했을 때는 다시하기, 스테이지 선택으로 돌아가기 선택지를 유저에게 제공하고, 성공했을 때는 다음 스테이지 플레이하기 선택지를 추가로 보여준다. 단, 마지막 스테이지는 다음 스테이지가 없기 때문에 다음 스테이지 플레이 버튼을 출력하지 않는다.

- 적용된 배운 내용

클래스 정의 및 인스턴스 생성(자세한 내용은 KeyListener 클래스, Stage 클래스 참조)

- 코드 스크린샷

```
127         // Actual Game Playing Page
128         else if(pageFlag == 3){
129             Stage s{stageFlag};
130             // this logic will be done after at least stage 1 is built
131             int gameresult = s.play(printer, stageFlag);
132             // Stage End, Give Multiple Choice
133             bool choice[3] = {true, false, false};
134             int pointer = 0;
135             int move = 0;
136             while(move != KeyListener::ENTER){
137                 system("cls");
138                 if (gameresult == 0 || stageFlag == 4){ // Last stage can't go onto next stage
139                     // try again option and back to stage selection option
140                     if(gameresult == 0)
141                         printer.printTitle(14 ,"Fail..");
142                     else
143                         printer.printTitle(14, "Clear!");
144                     printer.printOption("Try Again?", choice[0], 1);
145                     printer.printOption("Map Select", choice[1], 1);
146                     printer.printConfirmAlert();
147                     move = KeyListener::titleKey(); // determine action
148                     if(move != KeyListener::ENTER){ // Since only two options it will toggle
149                         choice[0] = !choice[0];
150                         choice[1] = !choice[1];
151                     }
152                 }
```

```
153                 else{ // if game clear
154                     printer.printTitle(14, "Clear!");
155                     printer.printOption("Try Again?", choice[0], 1);
156                     printer.printOption("Map Select", choice[1], 1);
157                     printer.printOption("Next Stage", choice[2], 1);
158                     printer.printConfirmAlert();
159                     move = KeyListener::titleKey(); // determine action
160                     if(move == KeyListener::UP){
161                         choice[pointer] = false;
162                         if(pointer == 0){
163                             choice[pointer = 2] = true;
164                         }
165                         else{
166                             choice[--pointer] = true;
167                         }
168                     }
169                     else if(move == KeyListener::DOWN){
170                         choice[pointer] = false;
171                         if(pointer == 2){
172                             choice[pointer = 0] = true;
173                         }
174                         else{
175                             choice[++pointer] = true;
176                         }
177                     }
178                 }
179             }
180             // At this point pageFlag is 3, therefore only choice[1] need to change the flag
181             if(choice[1])
182                 pageFlag = 1; // go onto selection page
183             else
184                 ++stageFlag; // go onto next stage
```

(5) 플레이어 엔티티

- 설명:

스테이지 맵 안에서 유저 입력에 따라 한 칸 이동하거나 특정 행동을 취할 수 있는 엔티티 (기호 : 'P'), 자세한 설명은 Player 클래스 참조

- 세부 기능:

[1] 이동 및 행동

- 설명: (기타 엔티티 추가에 따라 변화예정)

유저의 행동에 따라 stage 상의 'P'의 위치와 Player 인스턴스 user의 x, y값을 변환하고 유저의 stamina를 1 감소한다. 자세한 사항은 Player 클래스 move(), decreaseStamina() 참조

- 코드 스크린샷

```
61  void Stage::changeBoard(int action){ // activate when player success moving
62      // change position in board, user's location, and decrease statmina
63      stage[user.getY()][user.getX()] = ' ';
64      user.move(action);
65      stage[user.getY()][user.getX()] = 'P';
66      user.decreaseStamina();
67  }
```

[2] 행위 유효성 체크

- 변수:

① encounter: 유저가 향하려는 방향에 자리 잡은 엔티티 기호 저장 변수

② stage: 스테이지 맵 배열

③ user: Player 인스턴스

④ action: 입력 받은 키 값 저장 변수

- 설명:

사용자가 방향키 입력을 받아 이동을 시도할 때, 유저가 향하려는 방향에 존재하는 엔티티가 무엇인지 판별해 행위 여부를 결정한다. 이동이나 행동에 성공하면 stack에 행위 직전 상태를 stageNode 인스턴스로 만들에 저장한다.

- 코드 스크린샷

```
126     if(action == KeyListener::UP){
127         encounter = stage[user.getY() - 1][user.getX()];
128         if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
129         if(encounter == ' ' || encounter == '@'){
130             stack.push(StageNode{stage, user}); // save state before change
131             changeBoard(action, user);
132         }
133         else if(encounter == 'B'){
134             user.decreaseStamina();
135             stage[user.getY() - 1][user.getX()] = ' ';
136         }
137     }
138     else if(action == KeyListener::DOWN){
139         encounter = stage[user.getY() + 1][user.getX()];
140         if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
141         if(encounter == ' ' || encounter == '@'){
142             changeBoard(action, user);
143         }
144         else if(encounter == 'B'){
145             user.decreaseStamina();
146             stage[user.getY() + 1][user.getX()] = ' ';
147         }
148     }
149     else if(action == KeyListener::LEFT){
150         encounter = stage[user.getY()][user.getX() - 1];
151         if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
152         if(encounter == ' ' || encounter == '@'){
153             changeBoard(action, user);
154         }
155         else if(encounter == 'B'){
156             user.decreaseStamina();
157             stage[user.getY()][user.getX() - 1] = ' ';
158         }
159     }
160     else if(action == KeyListener::RIGHT){
161         encounter = stage[user.getY()][user.getX() + 1];
162         if(encounter != '#') stack.push(StageNode{stage, user}); // only encountering wall doesn't change the state
163         if(encounter == ' ' || encounter == '@'){
164             changeBoard(action, user);
165         }
166         else if(encounter == 'B'){
167             user.decreaseStamina();
168             stage[user.getY()][user.getX() + 1] = ' ';
169         }
170     }
```

(6) 게임 오버 조건 검사

- 변수

① user: Player 인스턴스

- 설명:

Stage의 play 내부 로직 중 하나. 유저의 잔여 체력을 검사해 체력이 0 이하이면 play를 루프하는 while문을 탈출한다. checkAlive()는 Player 클래스 참조

- 코드 스크린샷

```
107         if(!user.checkAlive()){ // if user try to move when stamina is 0 or less game over
108             break;
109         }
```

(7) 기타 엔티티

[1] 벽(기호 : '#')

- 설명:

스테이지에 고정된 엔티티, 플레이어가 해당 엔티티의 위치로 이동을 시도하면 플레이어가 아무 행동도 하지 않은 것으로 취급된다. (아무 행동을 취하지 않도록 만들기 위해 별도의 로직을 생성하지 않았음)

여기부터는 불완전 또는 미구현 엔티티

[2] 부술 수 있는 장애물(기호 : 'B')

[3] 밀 수 있는 장애물(기호 : 'O')

(8) 실행 취소

- 변수

① user: Player 인스턴스

② action: 유저의 키 입력

③ stage 스테이지 맵

④ stack: 스테이지 이전 상태들이 저장된 스택

⑤ tmp: 스택의 최상위 stageNode 인스턴스를 불러올 임시 변수

- 설명:

사용자가 Ctrl + z를 입력할 때, stack이 비어 있지 않은 경우 스택 최상위에 저장된 직전 스테이지 상태를 불러와 상태를 되돌린다.

- 코드 스크린샷

```
112         if(action == KeyListener::CTRL_Z){ // return to previous state if Ctrl + Z pressed
113             if(stack.empty() == false){
114                 StageNode tmp = stack.top();
115                 user.setLocation(tmp.x, tmp.y);
116                 user.setStamina(tmp.hp);
117                 copy(&tmp.stage[0][0], &tmp.stage[0][0] + (size*size), &stage[0][0]);
118                 stack.pop();
119                 continue;
120             }
121         }
```

(9) 세부 스테이지

- 설명:

스테이지가 올라감에 따라 스테이지의 엔티티 배치, 엔티티 종류가 달라지기 때문에 스테이지를 생성하는 로직을 만들기보다는 스테이지 모양을 그대로 2차원 배열에 저장한다.

[1] 스테이지 1

- 설명:

벽 엔티티와 목적지 엔티티만 존재하는 미로탈출형 스테이지 해당 스테이지의 유저 stamina는 35다.

-코드

```
char stage1[size][size] = {{'#','#','#','#','#','#','#','#','#','#','#','#'},
                           {'#','P',' ',' ',' ',' ',' ','#',' ',' ',' ','@','#'},
                           {'#',' ','#','#','#','#','#',' ',' ','#','#','#'},
                           {'#',' ','#',' ',' ',' ',' ','#',' ','#','#',' ','#'},
                           {'#',' ',' ',' ','#',' ','#',' ',' ',' ',' ',' ','#'},
                           {'#',' ','#','#','#',' ','#',' ',' ','#','#','#'},
                           {'#',' ','#',' ',' ',' ',' ','#',' ',' ','#',' ','#'},
                           {'#',' ','#',' ','#','#','#',' ','#','#',' ',' ','#'},
                           {'#',' ','#',' ',' ',' ',' ',' ',' ',' ',' ',' ','#'},
                           {'#',' ','#','#','#','#','#',' ','#','#','#','#','#'},
                           {'#',' ',' ',' ',' ',' ',' ','#',' ','#',' ',' ','#'},
                           {'#','#','#','#','#','#','#','#','#','#','#','#','#'}
                           };
```

[2] 스테이지 2 (형태만 구현, 플레이 불가)

- 설명:

벽 엔티티, 목적지 엔티티, 부술 수 있는 엔티티, 밀 수 있는 엔티티로 구성된 스테이지, 유저 초기 stamina는 35다.

-코드

```
42          char stage2[size][size] = {{'#','#','#','#','#','#','#','#','#','#','#','#','#'},
43                                     {'#','P','B','#',' ',' ','#',' ',' ',' ',' ',' ','#'},
44                                     {'#','#',' ','#',' ',' ','#',' ','#','#','#','#','#'},
45                                     {'#',' ',' ','#','#','#','#',' ','#',' ',' ',' ','#'},
46                                     {'#',' ',' ','B','O',' ',' ',' ',' ',' ',' ',' ','#'},
47                                     {'#','#','B','#','#','#','#',' ','#','#',' ',' ','#'},
48                                     {'#','#','O','#',' ',' ',' ',' ',' ','#','#',' ','#'},
49                                     {'#','#',' ',' ','#','#','#','#','#',' ','#','B','#'},
50                                     {'#','#',' ',' ','#',' ',' ','#',' ',' ',' ','O','O','#'},
51                                     {'#',' ',' ',' ',' ','O',' ',' ',' ',' ','O','B','#','#'},
52                                     {'#','#',' ',' ','#',' ',' ',' ','#','O',' ',' ',' ','@','#'},
53                                     {'#','#','#','#','#','#','#','#','#','#','#','#','#'}
54                                     };
```

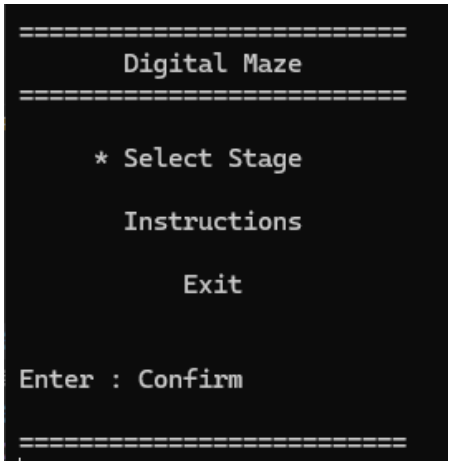
2) 테스트 결과

(1) 시작화면

- 설명

게임 제목과 옵션, 옵션 포인터 역할인 '*'가 잘 출력됨을 확인, 상/하 방향키 입력 시 선택된 옵션 줄로 '*' 위치가 이동함을 확인함. 각 옵션 별로 Enter 입력 시 의도한 동작 수행함.

- 테스트 결과 스크린샷



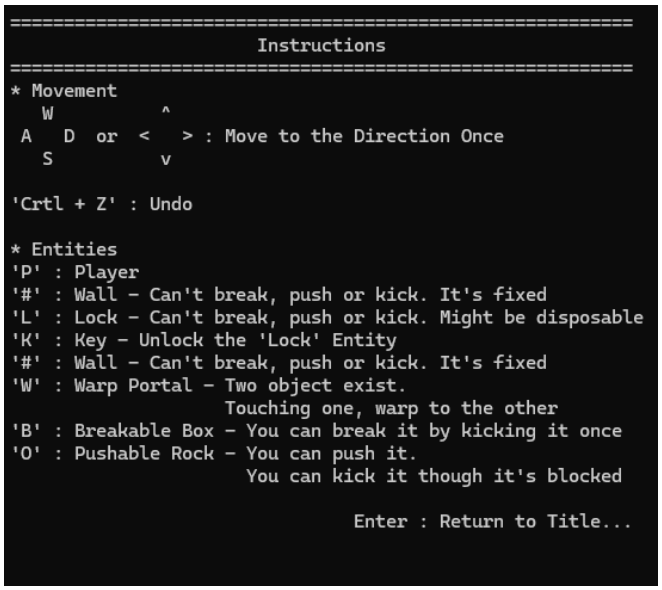
- 세부기능:

[1] 페이지 설명서

- 설명:

시작화면의 Instructions 옵션을 선택하면 출력되는 게임 플레이 방법과 게임 엔티티 소개가 담긴 페이지, Enter를 입력하면 시작 화면으로 복귀한다.

- 테스트 결과 스크린샷



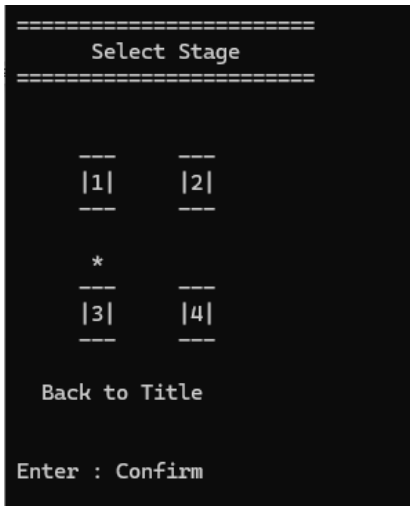
(2) 스테이지 선택창

- 설명

페이지의 제목, 4개의 스테이지 버튼, 시작화면 복귀 옵션, 그리고 선택된 옵션에 대한

'*' 포인터가 잘 작동했고, 키 입력에 의한 포인터 이동, Enter로 선택 확정이 잘 작동했다.

- 테스트 결과 스크린샷



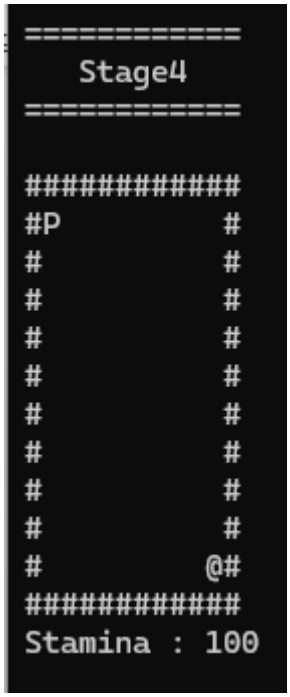
(3) 스테이지 맵

- 설명

스테이지 번호가 담긴 제목, 현재 스테이지 맵 상태, 유저의 잔여 체력이 출력됐다.

- 테스트 결과 스크린샷

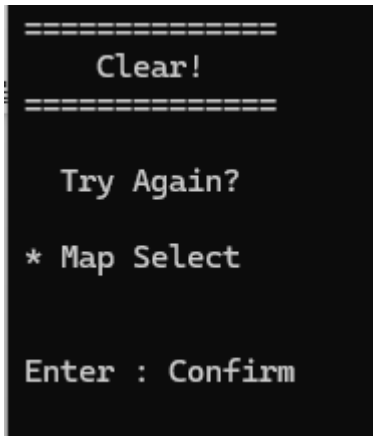
[1] 게임 플레이



[2] 게임 오버

[3] 게임 클리어 (Stage 1~3)

[4] 게임 클리어 (최종 Stage)



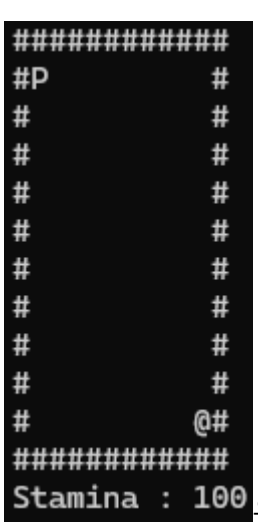
(5) 플레이어 엔티티

[1] 이동

- 설명

이동을 시도한 방향으로 맵 상의 위치가 바뀌고 stamina가 1 감소함을 확인, 이동 유효성 체크는 별도의 메시지 등이 없어 문서상 증빙이 불가능 하나 실행 취소 기능을 통해 정상 작동했음을 확인함. (이전 상태 스택에는 이동을 성공했을 때만 노드가 추가되도록 설계했고, 한 번 이동 후, 이동 불가능한 동작을 5번 시도한 후 Ctrl + Z를 한 번 누른 결과, 즉시 최초 위치로 복귀했음을 확인함)

- 테스트 결과 스크린샷



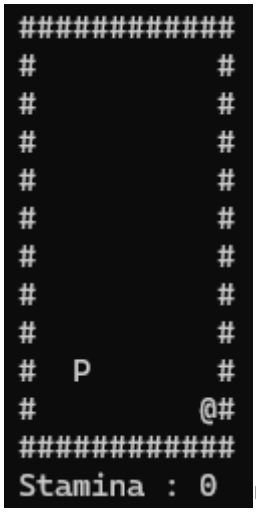
우측 이동 후


(6) 게임 오버 조건 검사

- 설명

Stamina가 0일 때 아무 방향으로 플레이어가 이동을 시도했을 때 Game over 상태로 넘어감을 확인함

- 테스트 결과 스크린샷



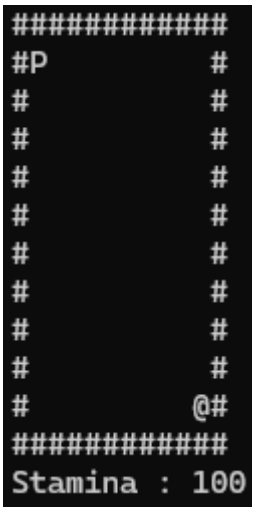
방향키 입력 후


(8) 실행 취소

- 설명

플레이어가 이동을 수행한 상태에서 Ctrl + Z를 누르면 직전 상태로 되돌아간다. 이동 불가능한 위치로 이동을 시도한 경우는 저장되지 않음도 테스트를 통해 확인함(기능 테스트 5-1 플레이어 엔티티 이동 참조)

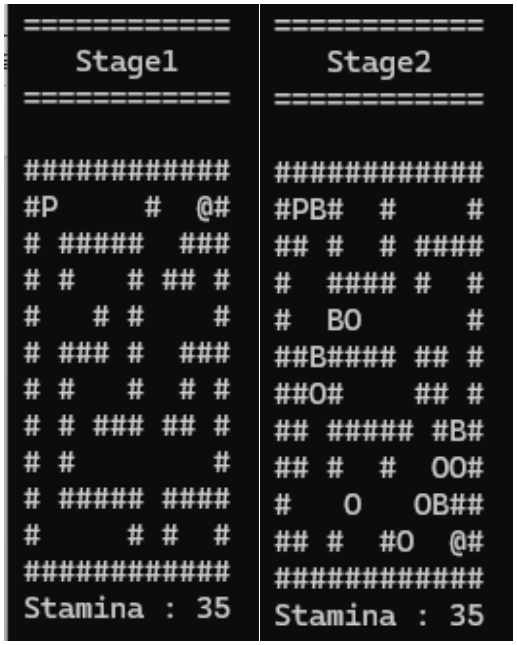
- 테스트 결과 스크린샷

Ctrl + Z 입력 이후


(9) 세부 스테이지

- 설명:

스테이지 번호 순서대로 나열됨 (현재 스테이지 2는 형태만 있고 엔티티 정상 동작 X)


Stage1
Stage2
Stamina : 35
Stamina : 35

4. 계획 대비 변경 사항

1) Dummy Stage 생성

- 이전: 4개의 Stage만 생성할 예정

- 이후: 벽과 목적지 플레이어만 존재하는 Dummy Stage 생성

- 사유: Stage를 다 만들기 전 main로직을 만들기 위해 빈 스테이지를 채울 임시 스테이지가 필요해 생성함. (추후 제거 예정)

2) 플레이어 엔티티 세부 기능 용어 변경

- 이전: 이동 유효성 체크

- 이후: 행위 유효성 체크

- 사유: 코드 작성 중, 이동 유효성을 체크하는 로직과 행동(엔티티 파괴, 엔티티 밀기) 등의 유효성을 체크하는 로직이 한 블록 안에 작성되는 것이 적절해 보인다 판단해 좀 더 포괄적인 용어 사용

5. 프로젝트 일정

업무		11/3	11/10	11/17	11/21	...
제안서 작성		완료				
기능1, 기능2, 기능3			완료			
기능6, 기능8			완료			
기능 5			진행 중(기능 7과 함께 완성 예정)			
기능4, 기능7				---------->		
기능7 - 1			완료			
기능9	Stage1		완료			
	Stage 2			---------->		
	Stage3, 4			---------->		
보수 및 최적화			---------->			