C++프로그래밍및실습

Digital Maze

진척 보고서 #3

제출일자: 2024.12.15
제출자명: 범지성
제출자학번: 214930

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 스테이지 초기화

- 설명: 게임을 플레이하던 중 'r' 키를 누르면 스테이지를 재시작한다.

10) 기능 10 스테이지 선택창으로 복귀

- 설명: 게임을 플레이하던 중 'b' 키를 누르면 스테이지를 선택창으로 돌아간다.

11) 기능 11 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) { // Print Page Title
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(Entity ***stage, int size, int stamina) {
56      for(int i = 0; i < size; i++){
57          for(int j = 0; j < size; j++){
58              if(stamina % 2 == 1 && stage[i][j]->GetSymbol() == 'W') // later change this with Warp->isActivate
59                  cout << ' ';
60              else
61                  cout << stage[i][j]->GetSymbol();
62          }
63          cout << endl;
64      }
65  }
```

(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 반환한다. 'b', 'r' 키도 입력을 그대로 반환한다. 나머지 키가 입력될 때는 앞서 나온 키 중 하나가 입력될 때까지 다시 키 입력 받기를 시도한다.

- 코드

```
int KeyListener::GetPlayerKey() { // 4 Direction, Ctrl + Z, 'b', 'r'
    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:
            case 'b':
            case 'r':
                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

```
20  int KeyListener::StageSelectionKey() { // Enable Enter, 4 Direction
21      while(1){
22          // get key input until get 4 direction or Enter
23          int move = _getch();
24          switch(move){
25              case ENTER:
26              case LEFT:
27              case RIGHT:
28              case UP:
29              case DOWN:
30                  return move;
31              case 'w':
32                  return UP;
33              case 'a':
34                  return LEFT;
35              case 'd':
36                  return RIGHT;
37              case 's':
38                  return DOWN;
39
40          }
41      }
42
43  }
```

(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

```
68  int KeyListener::EnableEnter() { // only Enter Allowed
69      while(1) {
70          int move = _getch();
71          if (move == ENTER) return move;
72      }
73  }
```

(3) Entity

- *적용된 배운 내용*

인스턴스가 함수를 호출할 때 해당 함수는 인스턴스의 맴버에 변화를 주는 일이 없도록 해야함을 명시하기 위해 함수 선언 시 함수를 const로 선언할 수 있다.

- 설명:

게임 플레이 중 생성되는 모든 엔티티의 공통 분모로 일부 클래스의 상위 클래스다.

- private 필드:

① symbol: 엔티티의 종류를 구분하는 기호

- 생성자:

(1) Entity()

- 기능:

symbol을 ' '으로 초기화하며 인스턴스를 생성한다.

- 코드

```
7       Entity(){symbol = ' ';}
```

(2) Entity(char s)

-매개변수

① s: 생성될 인스턴스가 가지는 기호

-기능:

symbol을 s로 초기화하며 인스턴스를 생성한다.

-코드

```
8       Entity(char s){symbol = s;}
```

- public 함수:

(1) void SetSymbol(char c)

-매개변수

① c: 인스턴스가 새롭게 가질 기호

- 기능:

인스턴스의 symbol을 c로 변경한다.

- 코드

```
10        void SetSymbol(char c) { symbol = c;}
```

(2) char const GetSymbol()

- 기능:

인스턴스의 symbol을 반환한다.

- 코드

```
11        char const GetSymbol() {return symbol;}
```

(4) Player

- 설명:

게임 플레이를 할 유저 인스턴스를 생성하기 위한 클래스다. Entity 클래스를 상속하고 있다.

- private 필드:

① stamina: 유저의 잔여 이동 가능 횟수

- 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)

-매개변수

① s: 인스턴스가 가질 스테미나 값

-기능:

stamina값이 s이고 상위 클래스 Entity의 멤버 symbol이 'P'인 Player 인스턴스를 생성한다.

-코드

```
9       Player(int s):Entity('P'){ stamina = s; }
```

- public 함수:

(1) bool CheckAlive()

- 기능:

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

- 코드

```
3   bool Player::CheckAlive() { // check remaining stamina and if lower than 0 return false(dead)
4       if(stamina <= 0)
5           return false;
6       else
7           return true;
8   }
```

(2) int GetStamina()

- 기능:

인스턴스의 stamina 반환

- 코드

```
9   int Player::GetStamina() { return stamina; }
```

(3) void IncreaseStamina()

- 기능:

객체의 stamina 값을 1 증가한다.

- 코드

```
10  void Player::IncreaseStamina() { stamina++; }
```

(4) void DecreaseStamina()

- 기능:

  인스턴스의 stamina가 1 감소한다.

- 코드

```
11  void Player::DecreaseStamina() { stamina--; } // might be used if any harmful obstacles are designed, at this point only for movement
```

(5) Portal

- 설명:

  Entity 클래스를 상속한 게임 스테이지 상의 구조물에 관한 클래스다. 해당 클래스의 인스턴스는 반드시 한 쌍을 이루어 생성되며, 플레이어가 두 인스턴스 중 한 인스턴스에 접촉하며 나머지 한 쪽으로 순간이동 되고 해당 Portal 인스턴스 쌍은 소멸한다.

- private 필드:

  ① x: 자신의 생성 위치 x좌표

  ② y: 자신의 생성 위치 y좌표

  ③ next_x: 연결된 Portal 인스턴스의 x좌표

  ④ next_y: 연결된 Portal 인스턴스의 y좌표

- 생성자:

(1) Portal(int x, int y)

-매개변수

  ① x: 인스턴스의 x 좌표

  ② y: 인스턴스의 y 좌표

- 기능:

맵 상에서 생성된 위치를 기준으로 x, y 값을 초기화하고 symbol을 'W'로 초기화한 Portal 인스턴스를 생성한다.

- 코드

```
3   Portal::Portal(int x, int y):Entity('W'){
4       this->x = x;
5       this->y = y;
6   }
```

- public 함수:

(1) void Connect(Portal *p)

-매개변수

① p: 함수 호출한 인스턴스와 연결할 Portal 인스턴스

- 기능:

p의 x, y 좌표를 Connect를 호출한 인스턴스의 next_x와 next_y로 저장한다.

- 코드

```
7   void Portal::Connect(Portal *p){
8       next_x = p->x;
9       next_y = p->y;
10  }
```

(2) int GetConnectedX()

- 기능:

인스턴스에 연결된 Portal의 x좌표를 반환한다.

- 코드

```
11  int Portal::GetConnectedX(){ return next_x;}
```

(3) int GetConnectedY()

- 기능:

인스턴스에 연결된 Portal의 y좌표를 반환한다.

- 코드

```
12    int Portal::GetConnectedY(){ return next_y;}
```

(6) 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 stage: 스테이지 맵 상의 엔티티가 가지고 있던 기호 정보

- public static 상수 필드:

① SIZE: 스테이지 맵 가로, 세로 폭

- 생성자

(1) StageNode(Entity ***stage, int x, int y)

-매개변수

① stage: 스테이지 맵 정보

② x: 스테이지 상 Player 인스턴스의 열 좌표

③ y: 스테이지 상 Player 인스턴스의 행 좌표

- 기능:

플레이어의 x, y 좌표 정보를 저장하고, stage의 모든 정보가 아닌 기호 정보만 추출해 2차원 배열 형태로 저장한다.

- 코드

```
3   StageNode::StageNode(Entity ***stage, int x, int y) {
4       this->x = x;
5       this->y = y;
6       // deepcopy stage
7       for(int i = 0; i < SIZE; i++) {
8           for(int j = 0; j < SIZE; j++) {
9               this->stage[i][j] = stage[i][j]->GetSymbol();
10          }
11      }
12  }
```

(7) Stage

- *적용된 배운 내용*

객체를 동적 할당 하는 new 키워드와 동적 할당된 메모리의 할당을 해제하는 delete 키워드를 통해 메모리 관리를 한다.

- 설명:

stage map 생성, game play 기능을 가진 클래스다.

- private 상수:

① SIZE: 맵의 가로 세로 최대 사이즈

- private 필드:

① stage: 현재 플레이 중인 스테이지 맵의 상태

② x: Player 인스턴스의 x좌표

③ y: Player 인스턴스의 y좌표

④ stack: stack 인스턴스, 플레이어 행동 직전 상태를 담은 StageNode 객체를 원소로 하는 스택

- 생성자

(1) Stage(int stage_flag)

- 매개변수:

① stage_flag: 스테이지 번호

- 설명

스테이지를 생성하는 함수가 호출되면서 생성되는 인스턴스의 필드들을 초기화한다.

- 코드

```
130  Stage::Stage(int stage_flag){ // according to flag, build different stage map
131      BuildStage(stage_flag);
132  };
```

- private 함수:

(1) void BuildSupport(int stamina, char stage[][SIZE])

- 매개변수:

① stamina: 생성할 Player 엔티티의 stamina 변수값

② stage: 생성할 스테이지 맵의 초기 엔티티 배치 정보가 담긴 배열

- 기능:

stage의 엔티티 위치 정보에 따라 해당 위치에 알맞은 Entity 인스턴스를 동적으로 할당한다. 생성된 한 쌍의 포탈을 연결하는 작업도 이 과정에서 수행된다.

- 코드

```
13  void Stage::BuildSupport(int stamina, char stage[][SIZE]) {  // Supporter of BuildStage Function
14      Portal *p = nullptr;
15      for(int i = 0; i < SIZE; i++){
16              for(int j = 0; j < SIZE; j++)
17                  if(stage[i][j] == 'P')  // allocate Player
18                      this->stage[i][j] = new Player{stamina};
19                  else if(stage[i][j] == 'W'){  // allocate Warp Portal
20                      if(p == nullptr){  // First Portal Created
21                          p = new Portal{j, i};
22                          this->stage[i][j] = p;
23                      }
24                      else{  // Second Portal Created
25                          Portal *q = new Portal(j, i);
26                          // Connect portals
27                          p->Connect(q);
28                          q->Connect(p);
29                          this->stage[i][j] = q;
30                      }
31                  }
32                  else
33                      this->stage[i][j] = new Entity{stage[i][j]};
34      }
35  }
```

(2) void BuildStage(int stage_flag)

-매개변수

① stage_flag: 선택된 스테이지 번호

- 기능:

비어있던 stage필드에 배열을 동적으로 할당한다. stage_flag에 따라 미리 만들어진 스테이지 맵의 배치 정보를 BuildSupport 함수에 보내 할당한 배열에 Entity 인스턴스들을 할당한다.

- 코드

```
void Stage::BuildStage(int stage_flag){ // Load stage according to flag
    // Build Stage with Dynamic Allocation
    // since entities position has no pattern, better type whole stage
    stage = new Entity**[SIZE];
    for(int i = 0; i < SIZE; i++){
        stage[i] = new Entity*[SIZE];
    }
    if (stage_flag == 1){ // Obstacle - Wall only
        char stage1[SIZE][SIZE] = {{'#','#','#','#','#','#','#','#','#','#','#','#','#'},
                                   {'#','P',' ',' ',' ',' ',' ','#',' ',' ','@','#'},
                                   {'#',' ','#','#','#','#','#',' ',' ','#','#','#'},
                                   {'#',' ','#',' ',' ',' ','#',' ','#','#',' ','#'},
                                   {'#',' ',' ',' ','#',' ','#',' ',' ',' ',' ','#'},
                                   {'#',' ','#','#','#',' ','#',' ',' ','#','#','#'},
                                   {'#',' ','#',' ',' ',' ','#',' ',' ','#',' ','#'},
                                   {'#',' ','#',' ','#','#','#',' ','#','#',' ','#'},
                                   {'#',' ','#',' ',' ',' ',' ',' ',' ',' ',' ','#'},
                                   {'#',' ','#','#','#','#','#',' ','#','#','#','#'},
                                   {'#',' ',' ',' ',' ',' ',' ','#',' ',' ',' ','#'},
                                   {'#','#','#','#','#','#','#','#','#','#','#','#'}
                                   };
        BuildSupport(35, stage1);
    }
```

```
    else if(stage_flag == 2){ // Obstacle - Wall, Breakable Box, and Pushable Box
        char stage2[SIZE][SIZE] = {{'#','#','#','#','#','#','#','#','#','#','#','#','#'},
                                   {'#','P','B','#',' ',' ',' ','#',' ',' ',' ',' ','#'},
                                   {'#','#',' ',' ','#',' ',' ',' ','#',' ','#','#','#'},
                                   {'#',' ',' ',' ','#','#','#',' ',' ','#',' ',' ','#'},
                                   {'#',' ',' ',' ','B','O',' ',' ',' ',' ',' ',' ','#'},
                                   {'#','#','B','#','#','#','#',' ',' ','#','#',' ','#'},
                                   {'#','#','O','#',' ',' ',' ',' ',' ','#','#',' ','#'},
                                   {'#','#',' ',' ','#','#','#','#','#',' ',' ','#','#'},
                                   {'#','#',' ',' ','#',' ',' ',' ','#',' ',' ','O','#'},
                                   {'#',' ',' ',' ',' ','O',' ',' ',' ',' ','O','B','#'},
                                   {'#','#',' ',' ','#',' ',' ',' ','#','O',' ',' ','#'},
                                   {'#','#','#','#','#','#','#','#','#','#','#','#','#'}
                                   };
        BuildSupport(35, stage2);
    }
    else if(stage_flag == 3){ // Obstacle - Wall, Breakable Box, Pushable Box, and Warp Portal
        char stage3[SIZE][SIZE] = {{'#','#','#','#','#','#','#','#','#','#','#','#','#'},
                                   {'#','P','#','@','#',' ',' ',' ','O','#','@',' ','#'},
                                   {'#',' ',' ','#','B','#',' ',' ','O',' ',' ','O','#'},
                                   {'#',' ',' ','B','W','#',' ',' ',' ','#','#','#','#'},
                                   {'#',' ',' ','#',' ',' ','B',' ','#','#',' ',' ','#'},
                                   {'#',' ',' ','#',' ',' ','#',' ',' ',' ',' ',' ','#'},
                                   {'#',' ',' ','#',' ',' ','#',' ','#','#','#','#','#'},
                                   {'#',' ',' ','B','B','B',' ',' ',' ',' ',' ',' ','#'},
                                   {'#',' ',' ','#','#','#','#',' ',' ','#',' ','#','#'},
                                   {'#',' ',' ',' ','O',' ',' ',' ',' ',' ','B',' ','W','#'},
                                   {'#',' ',' ','#','#','#','#',' ',' ','#','#','#','#'},
                                   {'#','#','#','#','#','#','#','#','#','#','#','#','#'}
                                   };
        BuildSupport(30, stage3);
    }
```

```
    }
    else if(stage_flag == 4){
        char stage4[SIZE][SIZE] = {{'#','#','#','#','#','#','#','#','#','#','#','#','#'},
                                   {'#','P',' ',' ',' ',' ',' ',' ','L',' ','@','#'},
                                   {'#','#','O','#',' ','#',' ',' ','L',' ',' ','#'},
                                   {'#','#','#','#',' ','#',' ',' ','W','L','L','#'},
                                   {'#',' ',' ',' ',' ',' ','#',' ',' ','B',' ',' ','#'},
                                   {'#','#',' ','#','#','#',' ','#',' ','#',' ','#'},
                                   {'#',' ',' ',' ',' ',' ',' ',' ','#',' ','#',' ','#'},
                                   {'#','#','#','#','#','#','#','#','#',' ','#','B','#'},
                                   {'#','O',' ','O',' ',' ',' ',' ',' ','O','#','B','#'},
                                   {'#','K','O','W','O','#',' ',' ',' ',' ','#','B','#'},
                                   {'#',' ',' ','O',' ',' ',' ',' ','#',' ',' ',' ','#'},
                                   {'#','#','#','#','#','#','#','#','#','#','#','#','#'}
                                   };
        BuildSupport(36, stage4);
    }
    // else buildDummyStage();  // Not needed anymore
}
```

(4) void ChangeBoard(int action)

-매개변수

① x: 위치를 바꿀 엔티티 중 하나의 x좌표

② y: 위치를 바꿀 엔티티 중 하나의 y좌표

③ next_x: 나머지 타겟 엔티티의 x좌표

④ next_y: 나머지 타겟 엔티티의 y좌표

- 기능:

좌표를 받은 두 엔티티의 stage 상의 위치를 swap한다.

- 코드

```
void Stage::ChangeBoard(int x, int y, int next_x, int next_y){ // activate when player success moving
    Entity *tmp = stage[y][x];
    stage[y][x] = stage[next_y][next_x];
    stage[next_y][next_x] = tmp;
}
```

(4) void Warp(int next_x, int next_y)

-매개변수

① next_x: 플레이어가 접근한 Portal 인스턴스의 x좌표

② next_y: 플레이어가 접근한 Portal 인스턴스의 y좌표

- 기능:

인자로 주어진 좌표 상에 존재하는 Portal 인스턴스와 연결된 Portal 인스턴스의 위치로 Player 인스턴스의 위치를 옮기고 여기에 사용된 한 쌍의 Portal 인스턴스의 기호를

' '로 변경해 포탈로써의 기능을 무력화한다.

- 코드

```
133  void Stage::Warp(int next_x, int next_y){ // When player enter Activated portal player warp and portal vanish
134      Portal *p = dynamic_cast<Portal*>(stage[next_y][next_x]);  // Entered Portal
135      // Player warp
136      ChangeBoard(x, y, p->GetConnectedX(), p->GetConnectedY());
137      // Portal Disappear
138      stage[y][x]->SetSymbol(' ');
139      p->SetSymbol(' ');
140      // Update player's location
141      x = p->GetConnectedX();
142      y = p->GetConnectedY();
143  }
```

(5) void Undo(Player *user)

-매개변수

① user: 현재 게임을 플레이 중인 Player 인스턴스의 참조변수

- 기능:

이전 행동을 취소하는 동작이다. user의 스테미나를 1 증가하고, stack의 최상단에 있는 StageNode 인스턴스를 가져와 user가 참조하는 Player 인스턴스를 StageNode에 저장된 직전 유저의 위치로 이동시키고 엔티티 파괴나 엔티티 이동에 의한 변화를 이전 상태로 복구하기 위해 StageNode 속 이전 맵의 정보를 토대로 Entity들의 symbol을 재설정한다. 이전 상태 복구가 끝난 후 사용된 StageNode는 stack에서 버려진다.

- 코드

```
149  void Stage::Undo(Player *user){ // Restore previous state
150      if(stack.empty() == false){
151          user->IncreaseStamina();
152          StageNode tmp = stack.top();
153          ChangeBoard(x, y, tmp.x, tmp.y);
154          for(int i = 0; i < SIZE; i++){ // Change STAGE with data in StageNode
155              for(int j = 0; j < SIZE; j++){
156                  stage[i][j]->SetSymbol(tmp.stage[i][j]);
157              }
158          }
159          x = tmp.x;
160          y = tmp.y;
161          stack.pop();
162      }
163  }
```

(6) void Unlock()

- 기능:

해당 함수가 호출되면 Lock 엔티티 역할을 하고 있던 Entity 인스턴스들의 기호를 모두 ' '로 변경해 Lock 엔티티를 무력화한다.

- 코드

```
164  void Stage::Unlock(){ // Remove every lock
165      for(int i = 1; i < SIZE - 1; i++){
166          for(int j = 1; j < SIZE - 1; j++){
167              if(stage[i][j]->GetSymbol() == 'L')
168                  stage[i][j]->SetSymbol(' ');
169          }
170      }
171      Sound::Unlock();
172  }
```

- public 함수:

(1) int Play(Frame f, int stage_flag)

-매개변수

① f: Frame 인스턴스

② stage_flag: 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

```cpp
int Stage::Play(Frame f, int stage_flag){ // Default Logic of game play, might be changed according to obstacles
    int clear_flag = 0; // if flag is 1 clear
    char encounter; // Entity that is which is on
    Player * user;
    string title[5] = {"Stage0", "Stage1", "Stage2", "Stage3", "Stage4"};
    while(true){
        user = dynamic_cast<Player*>(stage[y][x]);
        // refresh displayed screen
        system("cls");
        f.PrintTitle(12, title[stage_flag]);
        cout << endl;
        f.PrintStage(stage, SIZE, user->GetStamina());
        cout << "Stamina : " << user->GetStamina() << endl;
        cout << endl << "Ctrl + z : Undo";
        cout << endl << "r : Reset Stage";
        cout << endl << "b : Back to Selection";
        if(encounter == '@'){ // if reaching goal clear!
            clear_flag = 1;
            break;
        }

        int action = KeyListener::GetPlayerKey(); // get user action
        if(action == KeyListener::CTRL_Z){ // return to previous state if Ctrl + Z pressed
            Undo(user);
            continue;
        }
        else if(action == 'r') { // restart this stage
            clear_flag = 2;
            break;
        }
        else if(action == 'b') { // back to stage selection
            clear_flag = 3;
            break;
        }
        if(!user->CheckAlive()) break;  // if user try to move when stamina is 0 or less game over
        // Change stage according to player's movement
        int next_x = x;
        int next_y = y;
        // Identity the object which player's gonna encounter
        if(action == KeyListener::UP) --next_y;
        else if(action == KeyListener::DOWN) ++next_y;
        else if(action == KeyListener::LEFT) --next_x;
        else if(action == KeyListener::RIGHT) ++next_x;
        encounter = stage[next_y][next_x]->GetSymbol();
```

```cpp
        // Change stage according to player's movement
        int next_x = x;
        int next_y = y;
        // Identity the object which player's gonna encounter
        if(action == KeyListener::UP) --next_y;
        else if(action == KeyListener::DOWN) ++next_y;
        else if(action == KeyListener::LEFT) --next_x;
        else if(action == KeyListener::RIGHT) ++next_x;
        encounter = stage[next_y][next_x]->GetSymbol();
        // Identify Action
        if(encounter == 'L' || encounter == '#' || (encounter == 'W' && user->GetStamina() % 2 == 0)){ // If player meets wall or try to enter disabled portal, no action perform
            // Action Failed
            Sound::MoveFail();
            continue;
        }
        else{ // Action performed
            stack.push(StageNode{stage, x, y});  // Before action save present state in stack
            user->DecreaseStamina();  // Every action cost stamina
            if(encounter == ' ' || encounter == '@' || encounter == 'K'){ // Player change position
                if(encounter == 'K'){ // If player get key, unlock all the locks
                    Unlock();
                }
                stage[next_y][next_x]->SetSymbol(' ');
                ChangeBoard(x, y, next_x, next_y);
                x = next_x;
                y = next_y;
                Sound::Move();
            }
            else if(encounter == 'B'){ // Break Obstacle
                stage[next_y][next_x]->SetSymbol(' ');
                Sound::Break();
            }
```

```cpp
            else if(encounter == 'O'){ // Kick Obstacle
                // position beyond 'O'
                int next_x2 = 2 * next_x - x;
                int next_y2 = 2 * next_y - y;
                if(stage[next_y2][next_x2]->GetSymbol() == ' '){ // Push Obstacle if space exist
                    ChangeBoard(next_x, next_y, next_x2, next_y2);
                }
                Sound::Kick();
            }
            else if(encounter == 'W'){ // Player Enter Portal and teleport
                Warp(next_x, next_y);
                Sound::Warp();
            }
        }
    }
    return clear_flag; // 0 = fail, 1 = clear, 2 = restart stage, 3 = back to stage selection
}
```

(8) Progress

- *적용된 배운 내용*

파일 입출력 스트림을 다루는 ifstream, ofstream 객체를 적용해 외부 파일을 조작한다.

- 설명:

플레이어가 스테이지를 클리어한 현황을 외부 파일에 저장해 게임을 종료했다가 재시작 했을 때 해당 진행 상황을 유지할 수 있도록 외부 파일을 읽고 쓰는 작업을 함수화한 클래스다.

Windows Api를 적용한 클래스로 system("cls") 함수와 더불어, 이 때문에 해당 프로그램은 Windows 운영체제에서만 동작한다.

- public 함수:

(1) int Get()

- 기능:

현 프로젝트에 있는 "save/savedat.dat"(상대주소) 파일에 저장된 스테이지 해금 현황을 불러와 해독한 후 정수값으로 반환한다. 해당 파일이나 디렉토리가 존재하지 않으면 스테이지 1이 해금됐다는 정보를 담고 1을 반환한다.

- 코드

```
int Progress::Get() {  // Get Previous Progress
    // Relative Path of File and Directory
    LPCTSTR save = _T("save");
    LPCTSTR save_dat = _T("save\\savedat.dat");
    ifstream f{"save/savedat.dat"};  // open progress information
    if(!PathFileExists(save)) {  // if file doesn't exist
        system("mkdir save");  // mkdir
    }
    if(!PathFileExists(save_dat)){  // if file doesn't exist create new save file
        ofstream f2;
        if(!f2){  // critical error
            system("cls");
            cerr << "ERROR!! Fail to Save Data. Program Terminate in 3 seconds" << endl;
            Sleep(3000);
            exit(1);
        }
        f2.open("save/savedat.dat");
        f2 << Encode(1);  // make new savedat.dat file
        f2.close();
        return 1;  // unlock first stage
    }
    else{  // Load stage progress
        if(!f){  // critical error
            system("cls");
            cerr << "ERROR!! Fail to Save Data. Program Terminate in 3 seconds" << endl;
            Sleep(3000);
            exit(1);
        }
        int n;
        f >> n;
        f.close();
        return Decode(n);  // decode and return stage progress
    }
}
```

(2) void Save(int opened)

-매개변수

① opened: 현재 해방된 스테이지

- 기능:

opened를 암호화한 후 "save//save.dat" 파일에 덮어쓰기 한다.

- 코드

```
46  void Progress::Save(int opened) {  // Save clear progress
47      int n = Encode(opened);  // Encode opened
48      ofstream f{"save/savedat.dat"};
49      if(!f){  // critical error
50          system("cls");
51          cerr << "ERROR!! Fail to Save Data. Program Terminate in 3 seconds" << endl;
52          Sleep(3000);
53          exit(1);
54      }
55      f << n;  // Save encoded data
56      f.close();
57  }
```

(3) int Encode(int val)

-매개변수

① val: 타켓 정수값

- 기능:

val을 내부 규칙에 따라 암호화한다. 플레이어가 스테이지 클리어가 아닌 외부 파일 직접 조작으로 스테이지를 해금하는 경우를 방지하기 위한 기능이다.

- 코드

```
3   int Progress::Encode(int val) {
4       return (val + 3) * 9;
5   }
```

(4) int Decode(int code)

-매개변수

① code: Encode에의해 암호화된 정보

- 기능:

code값을 복호화하고 반환한다.

- 코드

```
7   int Progress::Decode(int code) {
8       return code / 9 - 3;
9   }
```

(9) Sound

- 설명:

게임의 다양한 효과음을 실행하는 작업을 함수화해 모아놓은 클래스

Windows Api 및 .wav 파일 적용

음원 파일 출처: https://taira-komori.jpn.org/freesoundkr.html

- static public 함수:

- 기능:

각 함수별로 지정된 음원을 재생하고 음원이 재생될 충분한 시간과 플레이어의 과도한 연속동작을 제한할 딜레이를 형성한다.

(1) void Select()

- 코드

```
3   void Sound::Select(){  // When user change option of something
4       PlaySound(TEXT("sound/select.wav"), 0, SND_FILENAME | SND_ASYNC);
5   }
```

(2) void Confirm()

- 코드

```
6   void Sound::Confirm(){  // When user confirm something with ENTER
7       PlaySound(TEXT("sound/confirm.wav"), 0, SND_FILENAME | SND_ASYNC);
8       Sleep(500);
9   }
```

(3) void Move()

- 코드

```
10  void Sound::Move(){  // When Player move on stage
11      PlaySound(TEXT("sound/move.wav"), 0, SND_FILENAME | SND_ASYNC);
12      Sleep(100);
13  }
```

(4) void MoveFail()

- 코드

```
14  void Sound::MoveFail(){  // When Player encounter wall
15      PlaySound(TEXT("sound/moveFail.wav"), 0, SND_FILENAME | SND_ASYNC);
16      Sleep(100);
17  }
```

(5) void Kick()

- 코드

```
18  void Sound::Kick(){  // When Player kick obstacle
19      PlaySound(TEXT("sound/kick.wav"), 0, SND_FILENAME | SND_ASYNC);
20      Sleep(100);
21  }
```

(6) void Break()

- 코드

```
22  void Sound::Break(){  // When Player break obstacle
23      PlaySound(TEXT("sound/select.wav"), 0, SND_FILENAME | SND_ASYNC);
24      Sleep(100);
25  }
```

(7) void Warp()

- 코드

```
26  void Sound::Warp(){  // When Player enter portal
27      PlaySound(TEXT("sound/warp.wav"), 0, SND_FILENAME | SND_ASYNC);
28      Sleep(100);
29  }
```

(8) void Unlock()

- 코드

```
30  void Sound::Unlock(){  // When Player get key
31      PlaySound(TEXT("sound/unlock.wav"), 0, SND_FILENAME | SND_ASYNC);
32  }
```

(9) void Clear()

- 코드

```
33  void Sound::Clear(){  // When Player clear the stage
34      PlaySound(TEXT("sound/clear.wav"), 0, SND_FILENAME | SND_ASYNC);
35  }
```

(10) void Fail()

- 코드

```
36  void Sound::Fail(){  // When Player fail the stage
37      PlaySound(TEXT("sound/fail.wav"), 0, SND_FILENAME | SND_ASYNC);
38  }
```

(11) void Back()

- 코드

```
39  void Sound::Back(){  // When Player drop the stage
40      PlaySound(TEXT("sound/back.wav"), 0, SND_FILENAME | SND_ASYNC);
41  }
```

(12) void Reset()

-코드

```
42  void Sound::Reset(){  // When Player restart the stage
43      PlaySound(TEXT("sound/reset.wav"), 0, SND_FILENAME | SND_ASYNC);
44  }
```

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

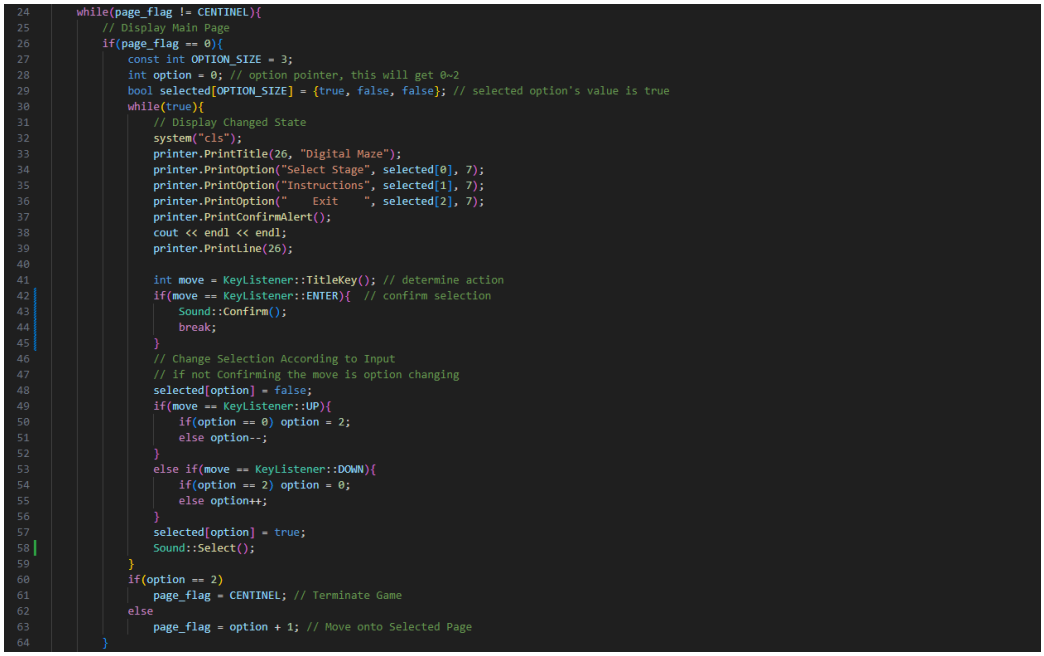
④ page_flag: main함수에서 실행할 페이지를 가리키는 변수

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.

- 코드 스크린샷



```
while(page_flag != CENTINEL){
    // Display Main Page
    if(page_flag == 0){
        const int OPTION_SIZE = 3;
        int option = 0; // option pointer, this will get 0~2
        bool selected[OPTION_SIZE] = {true, false, false}; // selected option's value is true
        while(true){
            // Display Changed State
            system("cls");
            printer.PrintTitle(26, "Digital Maze");
            printer.PrintOption("Select Stage", selected[0], 7);
            printer.PrintOption("Instructions", selected[1], 7);
            printer.PrintOption("    Exit    ", selected[2], 7);
            printer.PrintConfirmAlert();
            cout << endl << endl;
            printer.PrintLine(26);

            int move = KeyListener::TitleKey(); // determine action
            if(move == KeyListener::ENTER){  // confirm selection
                Sound::Confirm();
                break;
            }
            // Change Selection According to Input
            // if not Confirming the move is option changing
            selected[option] = false;
            if(move == KeyListener::UP){
                if(option == 0) option = 2;
                else option--;
            }
            else if(move == KeyListener::DOWN){
                if(option == 2) option = 0;
                else option++;
            }
            selected[option] = true;
            Sound::Select();
        }
        if(option == 2)
            page_flag = CENTINEL; // Terminate Game
        else
            page_flag = option + 1; // Move onto Selected Page
    }
```

-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

```
109         // Display Instructions
110         else if(page_flag == 2){
111             system("cls");
112             // most code in this line will goto Frame Class
113             // Print title
114             printer.PrintTitle(58, "Instructions");
115             // How to Play
116             cout << "* Movement" << endl;
117             cout << "   W          ^" << endl;
118             cout << " A   D  or  <   > : Move to the Direction Once" << endl;
119             cout << "   S          v" << endl;
120             cout << endl << "'Crtl + Z' : Undo" << endl;
121             cout << endl << "* Entities" << endl;
122             cout << "'P' : Player" << endl;
123             cout << "'#' : Wall - Can't break, push or kick. It's fixed" << endl;
124             cout << "'L' : Lock - Can't break, push or kick. But disposable" << endl;
125             cout << "'K' : Key - Unlock the 'Lock' Entity" << endl;
126             cout << "'W' : Warp Portal - Two object exist." << endl;
127             cout << "      Touching one, warp to the other" << endl;
128             cout << "      Portal must be activated at the moment you're touching" << endl;
129             cout << "'B' : Breakable Box - You can break it by kicking it once" << endl;
130             cout << "'O' : Pushable Rock - You can push it." << endl;
131             cout << "                     You can kick even when you can't push" << endl << endl;
132 
133             cout.width(58);
134             cout << "Enter : Return to Title...";
135 
136             KeyListener::EnableEnter(); // Loof until Pressing Enter
137             Sound::Confirm();
138             page_flag = 0; // Return to Main Page
139         }
```

(2) 스테이지 선택창

- 변수

① selected: 사용자의 입력을 저장한 변수

② page_flag: main함수에서 실행할 페이지를 가리키는 변수

③ printer: 화면 구성을 위한 Frame 인스턴스

④ stage_flag: 현재 선택된 스테이지 번호를 저장하는 변수

⑤ opened: 현재 해금된 최대 스테이지

- 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. 아직 해금되지 않은 스테이지는 선택할 수 없다.

- 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

```
66          // Display Selection Page
67          else if(page_flag == 1){
68              while(true){
69                  system("cls");
70                  // Print title
71                  printer.PrintTitle(24, "Select Stage");
72                  // button
73                  printer.PrintButtonLine(1, stage_flag);
74                  printer.PrintButtonLine(3, stage_flag);
75                  // change code under
76                  printer.PrintOption("Back to Title", !stage_flag, 2);
77                  printer.PrintConfirmAlert();
78
79                  int selected = KeyListener::StageSelectionKey();
80                  if(selected == KeyListener::ENTER){  // confirm selection
81                      Sound::Confirm();
82                      break;
83                  }
84                  if(opened == 1){  // only stage 1 is allowed
85                      stage_flag = !stage_flag;
86                  }
87                  else if(opened == 2){  // stage 1 & 2 allowed
88                      if(stage_flag == 0 || selected == KeyListener::UP || selected == KeyListener::DOWN)
89                          stage_flag = !stage_flag;
90                      else if(selected == KeyListener::LEFT || selected == KeyListener::RIGHT)
91                          stage_flag += 2 * (stage_flag % 2) - 1;
92                  }
```

```
93                  else if(opened == 3){  // stage 1, 2 & 3 allowed
94                      if((selected == KeyListener::LEFT || selected == KeyListener::RIGHT) && stage_flag != 3 && stage_flag != 0)
95                          stage_flag += 2 * (stage_flag % 2) - 1;
96                      else if(selected == KeyListener::DOWN){
97                          if(stage_flag == 3 || stage_flag == 0) stage_flag = !stage_flag;
98                          else stage_flag = 3;
99                      }
100                     else if(selected == KeyListener::UP){
101                         if(stage_flag == 0) stage_flag = 3;
102                         else stage_flag = stage_flag / 3;
103                     }
104                 }
105                 else if(opened == 4 || opened == 5){  // Every stage is allowed
106                     if(selected == KeyListener::UP){
107                         if(stage_flag == 0) stage_flag = 3;
108                         else if(stage_flag / 3 == 0) stage_flag = !stage_flag;
109                         else stage_flag -= 2;
110                     }
111                     else if(selected == KeyListener::DOWN){
112                         if(stage_flag == 3 || stage_flag == 4 || stage_flag == 0) stage_flag = !stage_flag;
113                         else stage_flag += 2;
114                     }
115                     else if(selected == KeyListener::LEFT || selected == KeyListener::RIGHT){
116                         if (stage_flag == 0);  // if back to stage is selected, do nothing
117                         else stage_flag += 2 * (stage_flag % 2) - 1;
118                     }
119                 }
120                 Sound::Select();
121             }
122             // If stage is selected go onto page 3, else go onto page 0
123             if(stage_flag == 0)
124                 page_flag = 0;
125             else
126                 page_flag = 3;
127         }
```

(3) 스테이지 맵

- 변수

① page_flag: main함수에서 실행할 페이지를 가리키는 변수

② printer: 화면 구성을 위한 Frame 인스턴스

③ stage_flag: 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:

stage_flag에 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. 사용자가 게임을 중도 포기했으면 스테이지 선택창으로 돌아가고 사용자가 스테이지 재시작을 희망했다면 해당 루프를 다른 조건 변화 없이 다시 실행하도록 한다.

- 코드 스크린샷

```
159         // Actual Game Playing Page
160         else if(page_flag == 3){
161             Stage s{stage_flag};
162             // this logic will be done after at least stage 1 is built
163             int gameresult = s.Play(printer, stage_flag);
164             if(gameresult == 2) {  // Reset this Stage
165                 Sound::Reset();
166                 continue;
167             }
168             else if(gameresult == 3) {  // Leave stage while it's not ended
169                 Sound::Back();
170                 page_flag = 1;
171                 continue;
172             }
173             else if(gameresult == 0) Sound::Fail();
174             else if(gameresult == 1) {
175                 Sound::Clear();
176                 if(opened == stage_flag){  // if the stage is first cleared
177                     p.Save(++opened);  // update savedat.dat
178                 }
179             }
180             // Stage End, Give Multiple Choice
181             bool choice[3] = {true, false, false};
182             int pointer = 0;
183             int move;
```

```
            while(1){
                system("cls");
                if (gameresult == 0 || stage_flag == 4){ // Last stage can't go onto next stage
                    // try again option and back to stage selection option
                    if(gameresult == 0)
                        printer.PrintTitle(14 ,"Fail..");
                    else
                        printer.PrintTitle(14, "Clear!");
                    printer.PrintOption("Try Again?", choice[0], 1);
                    printer.PrintOption("Map Select", choice[1], 1);
                    printer.PrintConfirmAlert();
                    move = KeyListener::TitleKey(); // determine action
                    if(move != KeyListener::ENTER){ // Since only two options it will toggle
                        choice[0] = !choice[0];
                        choice[1] = !choice[1];
                    }
                }
```

```
                else{ // if game clear
                    printer.PrintTitle(14, "Clear!");
                    printer.PrintOption("Try Again?", choice[0], 1);
                    printer.PrintOption("Map Select", choice[1], 1);
                    printer.PrintOption("Next Stage", choice[2], 1);
                    printer.PrintConfirmAlert();
                    move = KeyListener::TitleKey(); // determine action
                    choice[pointer] = false;  // there's only selection change action left
                    if(move == KeyListener::UP){
                        if(pointer == 0) pointer = 2;
                        else --pointer;
                    }
                    else if(move == KeyListener::DOWN){
                        if(pointer == 2) pointer = 0;
                        else ++pointer;
                    }
                    choice[pointer] = true;
                }
                if(move == KeyListener::ENTER) {
                    Sound::Confirm();
                    break;  // confirm selection
                }
                Sound::Select();
            }
            // At this point pageFlag is 3, therefore only choice[1] need to change the flag
            if(choice[1])
                page_flag = 1; // go onto selection page
            else if(choice[2])
                ++stage_flag; // go onto next stage

        }
    }
    return 0;
}
```

(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:

유저의는 사용자의 입력에 따라 이동, 장애물 파괴, 장애물 밀기, 장애물 차기 등의 행동을 수행할 수 있다. 행동을 수행하기 전 현재 상태를 담은 StageNode를 생성해 stack에 push 한다. 각각의 행동은 스테미나를 1 소모한다. 상세한 엔티티별 상호작용은 기타 엔티티에서 후술. 자세한 logic은 Stage 클래스의 함수들 참조

- 코드 스크린샷

```
225         else{ // Action performed
226             stack.push(StageNode{stage, x, y});  // Before action save present state in stack
227             user->DecreaseStamina();  // Every action cost stamina
228             if(encounter == ' ' || encounter == '@' || encounter == 'K'){ // Player change position
229                 if(encounter == 'K'){ // If player get key, unlock all the locks
230                     Unlock();
231                 }
232                 stage[next_y][next_x]->SetSymbol(' ');
233                 ChangeBoard(x, y, next_x, next_y);
234                 x = next_x;
235                 y = next_y;
236                 Sound::Move();
237             }
238             else if(encounter == 'B'){ // Break Obstacle
239                 stage[next_y][next_x]->SetSymbol(' ');
240                 Sound::Break();
241             }
242             else if(encounter == 'O'){ // Kick Obstacle
243                 // position beyond 'O'
244                 int next_x2 = 2 * next_x - x;
245                 int next_y2 = 2 * next_y - y;
246                 if(stage[next_y2][next_x2]->GetSymbol() == ' '){ // Push Obstacle if space exist
247                     ChangeBoard(next_x, next_y, next_x2, next_y2);
248                 }
249                 Sound::Kick();
250             }
251             else if(encounter == 'W'){ // Player Enter Portal and teleport
252                 Warp(next_x, next_y);
253                 Sound::Warp();
254             }
255         }
```

[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

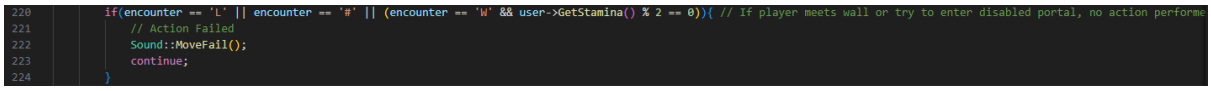
④ action: 입력 받은 키 값 저장 변수

- 설명:

사용자가 방향키 입력을 받아 이동을 시도할 때, 유저가 향하려는 방향에 존재하는 엔티티가 무엇인지 판별해 행위 여부를 결정한다. 한 장소에 고정된 파괴 불가능한 엔티티와 접촉하면 그 외 모든 다른 작업 없이 iteration이 종료돼 행동에 실패하게 된다. 행위

에 성공하면 바로 위에서 서술한 코드가 실행된다.

- 코드 스크린샷 (주어진 코드는 바로 위 코드와 연결된 if문)



```
220         if(encounter == 'L' || encounter == '#' || (encounter == 'W' && user->GetStamina() % 2 == 0)){ // If player meets wall or try to enter disabled portal, no action performe
221             // Action Failed
222             Sound::MoveFail();
223             continue;
224         }
```

(6) 게임 오버 조건 검사

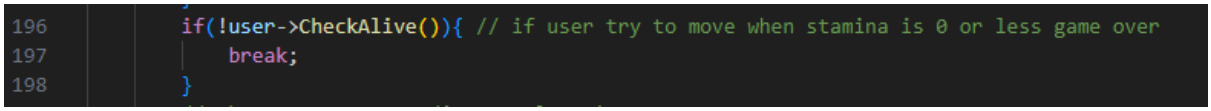
- 변수

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



```
196         if(!user->CheckAlive()){ // if user try to move when stamina is 0 or less game over
197             break;
198         }
```

(7) 기타 엔티티

[1] 벽(기호 : '#')

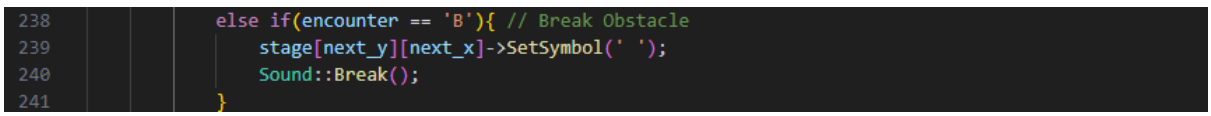
- 설명:

스테이지에 고정된 엔티티, 플레이어가 해당 엔티티의 위치로 이동을 시도하면 플레이어가 아무 행동도 하지 않은 것으로 취급된다.

[2] 부술 수 있는 장애물(기호 : 'B')

플레이어의 이동을 방해하는 엔티티, 해당 위치로 플레이어가 이동을 시도하면 플레이어의 위치는 변하지 않고 엔티티는 파괴되어 사라진다.

- 코드



```
238             else if(encounter == 'B'){ // Break Obstacle
239                 stage[next_y][next_x]->SetSymbol(' ');
240                 Sound::Break();
241             }
```

[3] 밀 수 있는 장애물(기호 : 'O')

플레이어의 이동을 방해하는 엔티티, 해당 위치로 플레이어가 이동을 시도하면 플레이어는 해당 장애물을 찬다(스테미나 1 소모). 플레이어가 해당 장애물을 찼을 때, 장애물 뒤가 빈 공간이면 장애물이 밀려나고 아니라면 플레이어의 스테미나만 소모된 채로 아무 일도 일어나지 않는다.

- 코드

```
242         else if(encounter == 'O'){ // Kick Obstacle
243             // position beyond 'O'
244             int next_x2 = 2 * next_x - x;
245             int next_y2 = 2 * next_y - y;
246             if(stage[next_y2][next_x2]->GetSymbol() == ' '){ // Push Obstacle if space exist
247                 ChangeBoard(next_x, next_y, next_x2, next_y2);
248             }
249             Sound::Kick();
250         }
```

[4] 잠긴 벽(기호 : 'L')

기본적으로 '#'와 동일하게 동작하고 특수한 조건을 만족하면 소멸한다.

[5] 열쇠(기호 : 'K')

해당 엔티티가 존재하는 위치로 플레이어가 이동하면 해당 엔티티는 소멸하고 동시에 스테이지에 존재하는 모든 'L' 엔티티도 소멸한다. 상세한 로직은 Stage 클래스의 Unlock() 함수 참조

- 코드

```
228         if(encounter == ' ' || encounter == '@' || encounter == 'K'){ // Player change position
229             if(encounter == 'K'){ // If player get key, unlock all the locks
230                 Unlock();
231             }
232             stage[next_y][next_x]->SetSymbol(' ');
233             ChangeBoard(x, y, next_x, next_y);
234             x = next_x;
235             y = next_y;
236             Sound::Move();
237         }
```

[6] 목적지 엔티티(기호 : '@')

플레이어가 해당 엔티티가 있는 위치로 이동하면 스테이지가 클리어를 판별하는 변수가 클리어 했을 때의 값으로 바뀌며 해당 스테이지가 종료된다.

```
186         if(encounter == '@'){ // if reaching goal clear!
187             clear_flag = 1;
188             break;
189         }
```

[7] 포탈 엔티티(기호 : 'W')

플레이어의 잔여 스테미나가 짝수일 때만 활성화되고 활성화된 상태에서는 벽처럼 작

동한다. 포탈 엔티티는 두 개가 하나의 쌍을 이루고 있으며 플레이어가 포탈 엔티티가 있는 곳의 좌표로 이동하는데 성공하면 연결된 다른 한 쪽 포탈 엔티티의 위치로 이동하고 포탈 엔티티 쌍은 소멸하게 된다. 자세한 로직은 Stage 클래스 Warp() 함수 참조

- 코드

```
251         else if(encounter == 'W'){ // Player Enter Portal and teleport
252             Warp(next_x, next_y);
253             Sound::Warp();
254         }
```

(8) 실행 취소

- 변수

① user: Player 인스턴스

② action: 유저의 키 입력

- 설명:

사용자가 Ctrl + z를 입력할 때, stack이 비어 있지 않은 경우 스택 최상위에 저장된 직전 스테이지 상태를 불러와 상태를 되돌린다. 자세한 로직은 Stage 클래스 Undo() 참조

- 코드 스크린샷

```
192         if(action == KeyListener::CTRL_Z){ // return to previous state if Ctrl + Z pressed
193             Undo(user);
194             continue;
195         }
```

(9) 스테이지 초기화

- 변수

① action: 유저의 키 입력

- 설명:

사용자가 'r'키를 입력하면 현재 플레이 중인 스테이지가 초기 상태로 재구성된다.

- 코드 스크린샷

[1] Stage 클래스 play 함수 내부 로직

```
198         else if(action == 'r') { // restart this stage
199             clear_flag = 2;
200             break;
201         }
```

[2] play로부터 2를 반환 받은 main()함수 로직 (stage_flag, page_flag 변동 없이 다음 iteration 진입)

```
135             int gameresult = s.Play(printer, stage_flag);
136             if(gameresult == 2) continue; // Reset this Stage
```

(10) 스테이지 선택창으로 복귀

- 변수

① action: 유저의 키 입력

- 설명:

사용자가 'b'키를 입력하면 현재 스테이지가 즉시 종료되고 스테이지 선택창으로 되돌아간다.

- 코드 스크린샷

[1] play() 함수 일부

```
202         else if(action == 'b') { // back to stage selection
203             clear_flag = 3;
204             break;
205         }
```

[2] main() 함수 일부

```
137             else if(gameresult == 3) {
138                 page_flag = 1;
139                 continue;
140             }
```

(11) 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

- 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                                     };
```

[3] 스테이지 3

- 설명:

벽 엔티티, 목적지 엔티티, 부술 수 있는 엔티티, 밀 수 있는 엔티티, 포탈 엔티티로 구

성된 스테이지, 유저 초기 stamina는 30이다.

-코드

```
76          char stage3[SIZE][SIZE] = {{'#','#','#','#','#','#','#','#','#','#','#'},
77                                     {'#','P',' ','@','#',' ',' ','O','#','@',' ','#'},
78                                     {'#',' ','#','B','#',' ',' ','O',' ',' ','O','#'},
79                                     {'#',' ','B','W','#',' ',' ',' ','#','#','#','#'},
80                                     {'#',' ','#',' ',' ','B',' ','#','#',' ',' ',' ','#'},
81                                     {'#',' ','#',' ','#',' ',' ',' ','#',' ',' ',' ','#'},
82                                     {'#',' ','#',' ','#',' ','#','#','#','#','#','#','#'},
83                                     {'#',' ','B','B','B',' ',' ',' ',' ',' ',' ',' ','#'},
84                                     {'#',' ','#','#','#','#',' ','#',' ','#','#','#'},
85                                     {'#',' ',' ',' ',' ','O',' ',' ',' ','B',' ',' ','W','#'},
86                                     {'#',' ','#','#','#','#',' ','#','#','#','#','#'},
87                                     {'#','#','#','#','#','#','#','#','#','#','#','#'}
88                                     };
```

[4] 스테이지 4

- 설명:

벽 엔티티, 목적지 엔티티, 부술 수 있는 엔티티, 밀 수 있는 엔티티, 포탈 엔티티, 열쇠와 잠긴 벽 엔티티로 구성된 스테이지, 유저 초기 stamina는 36이다.

-코드

```
92          char stage4[SIZE][SIZE] = {{'#','#','#','#','#','#','#','#','#','#','#','#','#'},
93                                     {'#','P',' ',' ',' ',' ',' ',' ',' ','L',' ',' ','@','#'},
94                                     {'#','#','O','#',' ',' ','#',' ',' ','L',' ',' ',' ','#'},
95                                     {'#','#','#','#',' ',' ','#',' ',' ','W','L','L','#'},
96                                     {'#',' ',' ',' ',' ',' ','#',' ',' ','B',' ',' ',' ','#'},
97                                     {'#','#',' ','#','#','#',' ',' ','#',' ',' ','#',' ','#'},
98                                     {'#',' ',' ',' ',' ',' ',' ',' ',' ','#',' ',' ','#',' ','#'},
99                                     {'#','#','#','#','#','#','#','#',' ',' ','#','B','#'},
100                                    {'#','O',' ',' ','O',' ',' ',' ',' ',' ','O','#','B','#'},
101                                    {'#','K','O','W','O','#',' ',' ',' ',' ','#','B','#'},
102                                    {'#',' ',' ',' ','O',' ',' ',' ',' ','#',' ',' ',' ','#'},
103                                    {'#','#','#','#','#','#','#','#','#','#','#','#','#'}
104                                    };
```

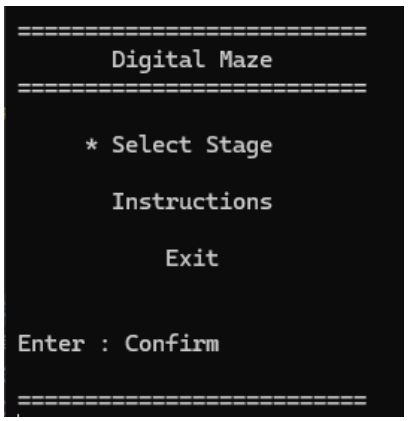
4. 테스트 결과

(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



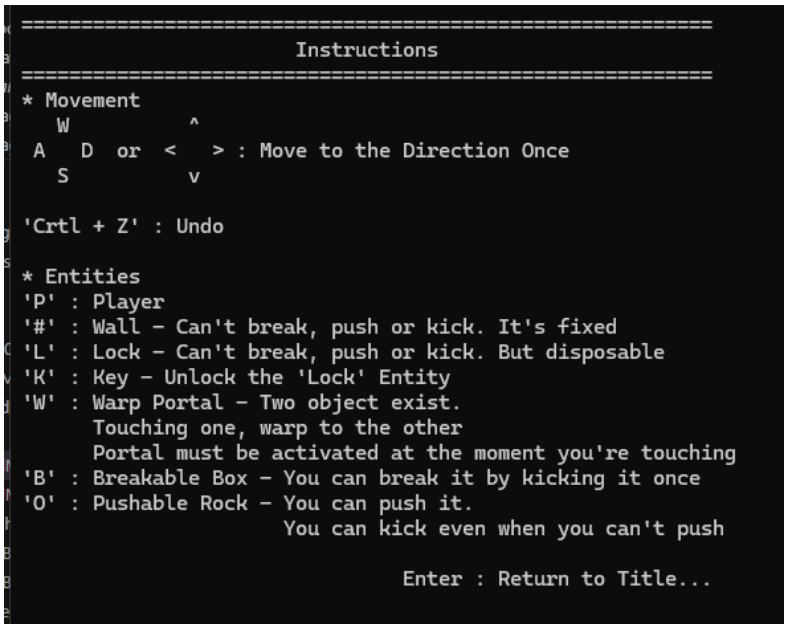
- 세부기능:

[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

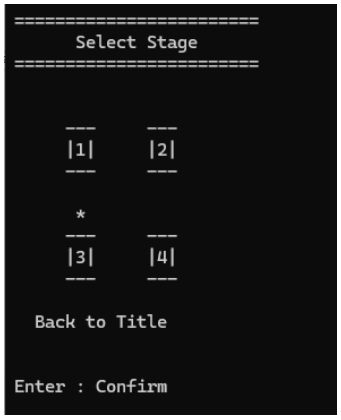


(2) 스테이지 선택창

- 설명

페이지의 제목, 4개의 스테이지 버튼, 시작화면 복귀 옵션, 그리고 선택된 옵션에 대한 '*' 포인터가 잘 작동했고, 스테이지 해금 상태에 따라 선택되면 안되는 스테이지는 선택이 되지 않는 것도 확인했다. 키 입력에 의한 포인터 이동, Enter로 선택 확정이 잘 작동했다.

- 테스트 결과 스크린샷



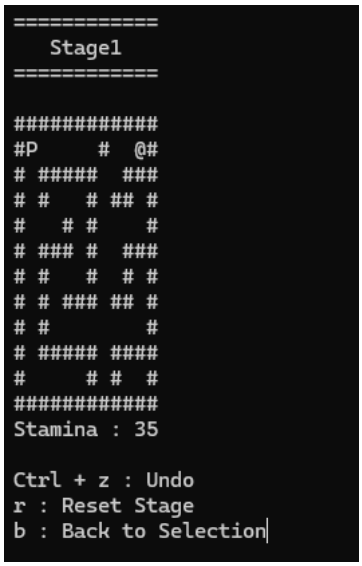
(3) 스테이지 맵

- 설명

스테이지 번호가 담긴 제목, 현재 스테이지 맵 상태, 유저의 잔여 체력이 출력됐다. 게임 클리어 메시지는 잔여 stamina가 0 이상일 때 목적지 엔티티에 도착하면 출력된다.

- 테스트 결과 스크린샷

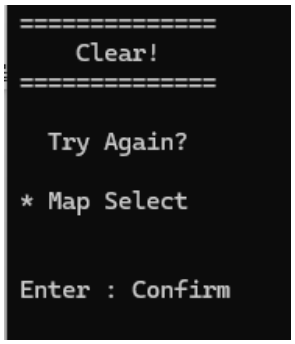
[1] 게임 플레이



[2] 게임 오버

[3] 게임 클리어 (Stage 1~3)

[4] 게임 클리어 (최종 Stage)



(5) 플레이어 엔티티 및 기타 엔티티 간의 상호 작용

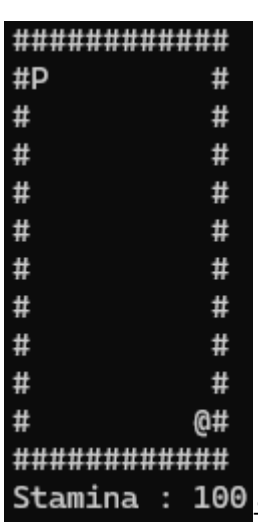
- 이동 및 행동

- 설명

이동을 시도한 방향으로 맵 상의 위치가 바뀌고 stamina가 1 감소함을 확인, 이동이 불가능할 때는 이동 실패 효과음이 출력되고 stamina에 변화가 없으며 Ctrl + Z로 되감기를 해도 제자리에 있는 것이 아닌 그 전 움직임이 취소됨을 확인함.

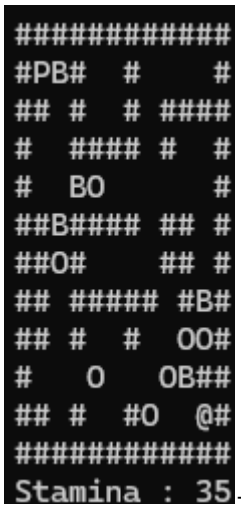
- 테스트 결과 스크린샷

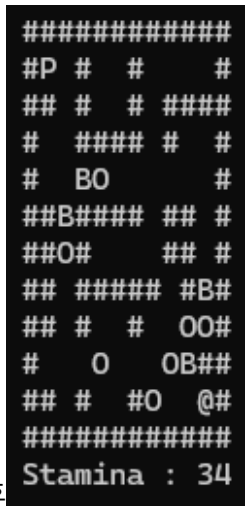
[1] 단순 이동



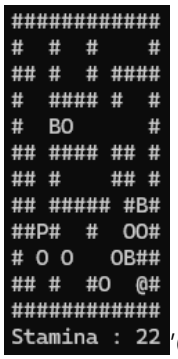
우측 이동 후

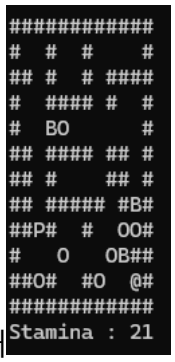

[2] 'B' 파괴



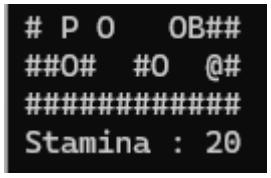
우측 이동 시도 후


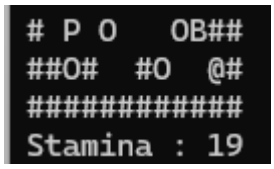
[3] 'O' 차기



'O' 뒤 공간이 있을 때


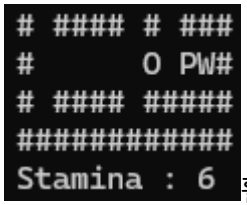
(좌하단 주목)



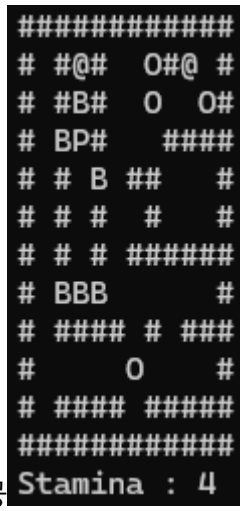
공간이 없을 때


(Stamina만 감소)

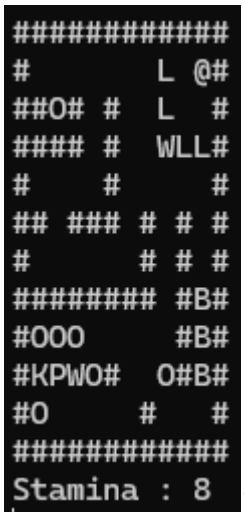
[4] 포탈 입성

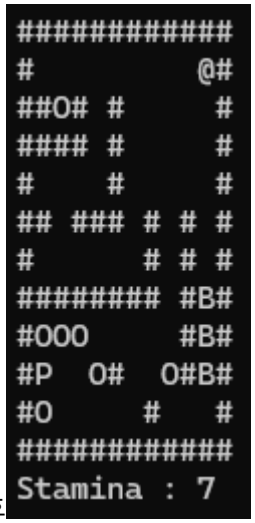


활성화된 포탈로는 접근 불가

비활성화 상태일 때 포탈의 좌표로 이동 시 워프 성공


[5] 열쇠로 잠긴 벽 엔티티 제거



열쇠 획득 후


(6) 게임 오버 조건 검사

- 설명

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

- 테스트 결과 스크린샷

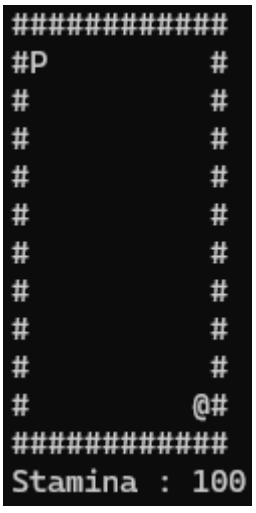
방향키 입력 후


(7) 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

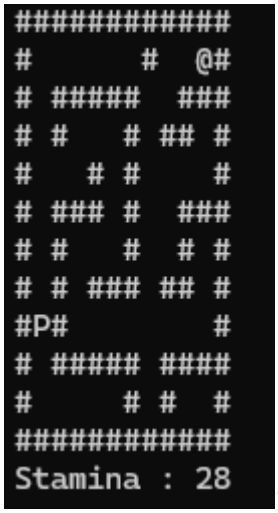
Ctrl + Z 입력 이후


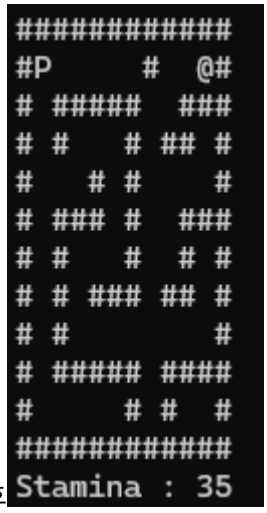
(8) 스테이지 재시작

- 설명

  플레이어가 이동을 수행한 상태에서 'r'를 누르면 최초 상태로 되돌아감

- 테스트 결과 스크린샷



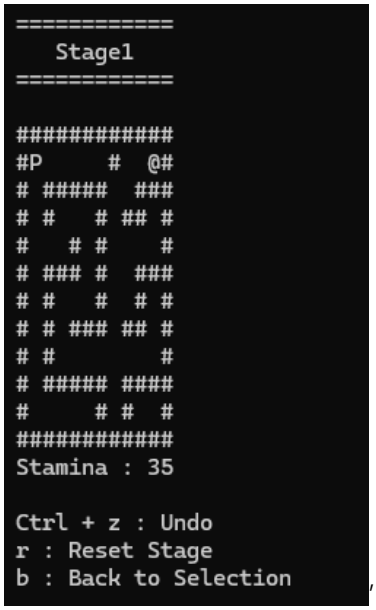
'r' 누른 후


(9) 스테이지 선택창으로 복귀

- 설명

  플레이어가 스테이지 플레이 도중 'b'를 누르면 스테이지 선택창으로 복귀함

- 테스트 결과 스크린샷

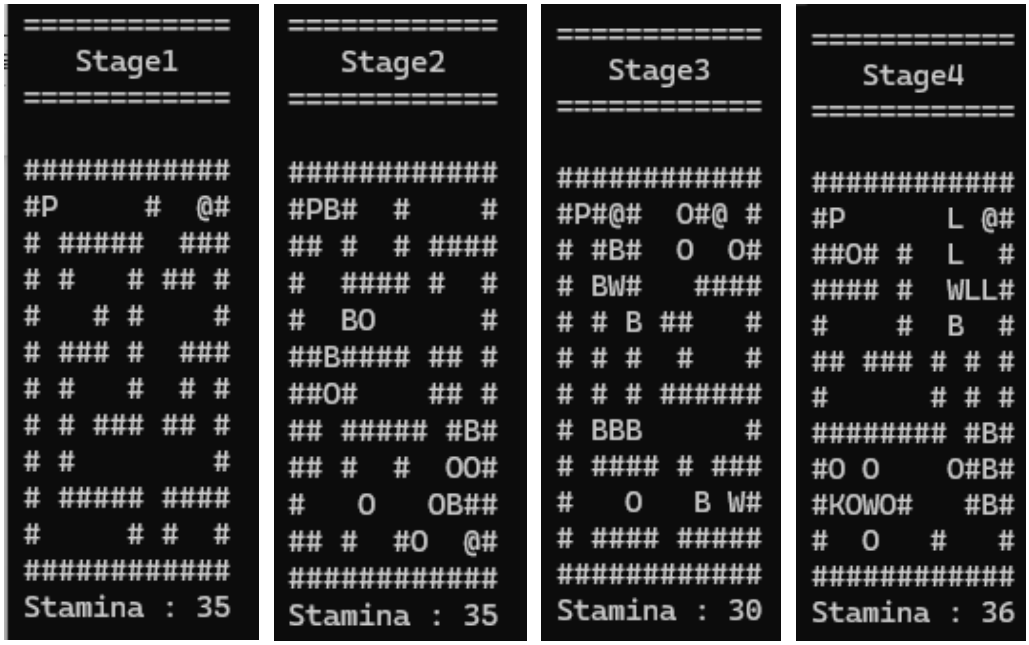


'b' 누른 후


(10) 세부 스테이지

- 설명:

  스테이지 번호 순서대로 나열됨 (각 스테이지의 클리어 가능성은 확인 완료)



5. 2차 보고 대비 변경 사항

1) 게임 스테이지 선택창에서 로직 변경

- 이전: 조건문으로 분기를 나눠 사용자 입력에 따라 어떤 작업이 수행되는지 알기 쉽게 작성된 코드

- 이후: 여러 조건 한 줄로 압축, 조건문을 대체 가능한 산술연산으로 교체

- 사유: 프로그램의 작동 방식을 파악하기에는 가독성이 좀 떨어지는 방식이지만 파이프라인 구조를 채택하고 곱셈, 나눗셈 전용 연산 유닛을 가진 현대 CPU는 일반적으로 branch instruction의 분기 예측 실패에 의한 파이프라인 비움이 다수의 곱셈, 나눗셈 연산보다 성능저하로 이어질 확률이 크다. 따라서 일부 조건문을 줄이기 위해 가독성을 희생하고 로직을 전환함.

2) 사운드 트랙 추가

- 게임의 동작을 좀 더 명확히 구분하고 풍성한 효과를 제공하기 위해 wav 파일을 좀 더 추가함

6. 프로젝트 일정

업무		11/3	11/10	11/17	11/21	12/15
제안서 작성		완료				
기능1, 기능2, 기능3			완료			
기능6, 기능8			완료			
기능 5				완료		
기능4						완료
기능7 - 1			완료			
기능 7				완료		
스테이지	Stage1		완료			
	Stage 2			완료		
	Stage3, 4				완료	
기능 9, 10				완료		
보수 및 최적화			---------->			