MUD game 1차 개발 보고서

203342

소프트웨어공학과

조영진

1. 서론

A. 프로젝트 목적 및 배경

i. 중간고사 전까지 배운 CPP 언어를 배경으로 터미널에서 진행하는 간단한 MUD(Multi-User-Dungeon) 게임을 개발하며 디자인 패턴을 학습하기 위함.

B. 프로젝트 목표

i. 간단한 MUD(Multi-User-Dungeon) 게임 개발

2. 요구사항

A. 사용자 요구사항

i. 2차원 맵에서 상하좌우로 이동하며 목적지에 도착하는 것을 목표로 하는 게임

B. 기능 계획

i. 사용자에서 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받기
   ① 명령어 입력 받을 때마다 HP를 함께 출력

ii. 유저는 HP 20을 가지고 게임 시작
   ① HP가 0이 되면 "체력이 다 떨어졌습니다. 실패...."를 출력하고 종료

iii. 입력 받은 명령어에 따라 각 명령을 수행하기
   ① 지도: 현재 지도를 출력
   ② 종료: 게임을 종료
   ③ 이동: 좌표 유효 검증 후 이동
      - 좌표가 유효하지 않을 시 메시지 출력 후 재입력 대기
      - 이동 시마다 HP가 1씩 감소

iv. 좌표 상 아이템, 포션, 적, 목적지를 발견하면 상호작용
   ① 아이템
      - "아이템을 발견했다! 의외의 행운이 따르는군?" 출력
   ② 포션
      - "포션! 의외로 맛있는걸??" 출력
      - 유저의 HP가 2 증가
   ③ 적
      - "적을 맞닥뜨렸다...! 무찌르자!" 출력
      - 유저의 HP가 2 감소
   ④ 목적지
      - 도달 시 "목적지에 도착했습니다! 축하합니다!"을 출력하고 종료

C. 함수 계획

i. 메인 함수: 사용자에게 값을 계속 입력 받고, 그에 대한 함수 호출
ii. displayMap 함수: 현재 유저의 위치를 포함해 맵 전체를 출력하는 함수
iii. checkState 함수: 유저가 위치한 타일이 특수한(아이템, 포션, 적, 목적지) 타일인지 확인하고 상호작용하는 함수
iv. checkXY 함수: 유저가 이동하고자 하는 좌표(2차원)가 유효한지 확인하는 함수
v. move 함수: 유저가 입력한 방향에 따라 움직이는 함수
vi. action 함수: 유저가 명령어를 입력했을 때 "종료"를 제외하고 나머지 행동을 실행하는 함수

3. 설계 및 구현

A. 기능 별 구현 사항

i. 유저의 HP를 출력해주고 상/하/좌/우/지도/종료 중 하나를 입력 받기

① 스크린샷

```
// 사용자의 입력을 저장할 변수
string user_input = "";

cout << "현재 HP: " << userHP <<"  명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
cin >> user_input;
```

② 입력
- 상/하/좌/우/지도/종료 명령어

③ 결과
- user_input(string)에 입력 받은 명령어 할당
- 유저의 현재 HP를 출력

④ 설명
- user_input을 string으로 선언하여 입력 받은 명령어를 대입

ii. 유저는 HP 20을 가지고 시작, HP가 0이 되면 게임 종료

① 스크린샷

```
// 유저의 체력 선언
int userHP = 20;
```

```
// 체력의 하락으로 종료
if (userHP <= 0) {
    cout << "체력이 다 떨어졌습니다. 실패...." << endl;
    break;
}
```

② 설명
- 유저의 HP 20으로 선언

- 유저의 HP 0과 같거나 작으면 while loop 탈출

iii. 지도 명령 수행 함수

① 스크린샷

```
// 지도와 사용자 위치 출력하는 함수
void displayMap(int map[][mapX], int *user_x, int *user_y) {
    for (int i = 0; i < mapY; i++) {
        for (int j = 0; j < mapX; j++) {
            if (i == *user_y && j == *user_x) {
                cout << " USER ¦"; // 양 옆 1칸 공백
            }
            else {
                int posState = map[i][j];
                switch (posState) {
                case 0:
                    cout << "      ¦"; // 6칸 공백
                    break;
                case 1:
                    cout << "아이템¦";
                    break;
                case 2:
                    cout << "  적  ¦"; // 양 옆 2칸 공백
                    break;
                case 3:
                    cout << " 포션 ¦"; // 양 옆 1칸 공백
                    break;
                case 4:
                    cout << "목적지¦";
                    break;
                }
            }
        }
        cout << endl;
        cout << " ------------------------------ " << endl;
    }
}
```

② 입력

- map[][](2D array): 현재 지도 배열
- *user_x: 현재 유저의 X좌표
- *user_y: 현재 유저의 Y좌표

③ 결과

- 전체 지도를 출력
- 유저의 위치를 출력

④ 설명

- 2차원 배열을 순회하며 지도의 상태를 출력
- 유저의 좌표와 같을 경우 유저를 출력

iv. 종료 명령 수행

① 스크린샷

```
if (user_input == "종료")
    // 프로그램 종료
    cout << "종료합니다.";
    break;
```

② 설명

- 종료가 입력되면 "종료합니다."를 출력하고 while loop를 탈출하여 프로그램을 종료

v. 이동 명령 수행 함수

① 스크린샷

```
bool move(int dx, int dy, int *user_x, int *user_y, int *userHP, int map[][mapX]) {
    // 이동만을 담당하는 함수 dx, dy만큼 검증하여 이동하는 함수
    *user_x += dx;
    *user_y += dy;
    if (!checkXY(user_x, user_y)) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        *user_x -= dx;
        *user_y -= dy;
        return false;
    }
    *userHP -= 1;
    return true;
}
```

② 입력

- dx: 유저가 이동하고자 하는 X좌표의 크기
- dy: 유저가 이동하고자 하는 Y좌표의 크기
- *user_x: 유저의 현재 X좌표
- *user_y: 유저의 현재 Y좌표
- *userHP: 유저의 현재 HP
- map[][]: 현재 맵의 상태

③ 반환

- bool: 맵을 벗어나면 false, 그렇지 않다면 true

④ 결과

- 유저가 이동하거나, 이동하지 않고 다시 실행하기 위해 false를 반환

⑤ 설명

- checkXY함수를 사용하여 유효한 좌표인지 확인하고 유효하지 않을 경우 에러 메시지를 출력하고 false를 반환
- 이동 좌표 검증이 끝나면 유저의 HP – 1을 진행

vi. 입력 받은 명령어에 따라 명령을 수행하는 함수 & 코드블럭

① 스크린샷

```
if (user_input == "종료") {
    // 프로그램 종료
    cout << "종료합니다.";
    break;
} else {
    bool isAction = action(user_input, map, &user_x, &user_y, &userHP);
    if (!isAction)
        continue;
}
```

```
bool action(string action, int map[][mapX], int *user_x, int *user_y, int *userHP) {
    // 상, 하, 좌, 우 이동과 지도 출력, 입력값 검증을 하는 함수
    // 포인터들로 실질적 값의 변경이 가능
    if (action == "상") {
        // 위로 한 칸 올라가기
        bool isMoved = move(0, -1, user_x, user_y, userHP, map);
        if (isMoved) {
            cout << "위로 한 칸 올라갑니다." << endl;
            displayMap(map, user_x, user_y);
            cout << "너무 어두워서 이동하기가 어렵다. HP: " << *userHP << " -> " << *userHP-1 << endl;
            return true;
        } else {
            return false;
        }
    }
    else if (action == "하") {
        // TODO: 아래로 한 칸 내려가기
        bool isMoved = move(0, 1, user_x, user_y, userHP, map);
        if (isMoved) {
            cout << "아래로 한 칸 내려갑니다." << endl;
            displayMap(map, user_x, user_y);
            cout << "너무 어두워서 이동하기가 어렵다. HP: " << *userHP << " -> " << *userHP-1 << endl;
            return true;
        } else {
            return false;
        }
    }
    else if (action == "좌") {
        // TODO: 왼쪽으로 이동하기
        bool isMoved = move(-1, 0, user_x, user_y, userHP, map);
        if (isMoved) {
            cout << "왼쪽으로 한 칸 이동합니다." << endl;
            displayMap(map, user_x, user_y);
            cout << "너무 어두워서 이동하기가 어렵다. HP: " << *userHP << " -> " << *userHP-1 << endl;
            return true;
        } else {
            return false;
        }
    }
```

```
    else if (action == "우") {
        // TODO: 오른쪽으로 이동하기
        bool isMoved = move(1, 0, user_x, user_y, userHP, map);
        if (isMoved) {
            cout << "오른쪽으로 한 칸 이동합니다." << endl;
            displayMap(map, user_x, user_y);
            cout << "너무 어두워서 이동하기가 어렵다. HP: " << *userHP << " -> " << *userHP-1 << endl;
            return true;
        } else {
            return false;
        }
    }
    else if (action == "지도") {
        // TODO: 지도 보여주기 함수 호출
        displayMap(map, user_x, user_y);
    }
    else {
        cout << "잘못된 입력입니다." << endl;
        return false;
    }
    return true;
}
```

② 입력

- user_input(string): 유저가 입력한 명령어를 저장하는 변수
- action(string): "종료" 명령을 제외하고 함수에 유저의 명령어를 전달하는 인자
- *user_x: 유저의 현재 X좌표
- *user_y: 유저의 현재 Y좌표
- *userHP: 유저의 현재 HP
- map[][]: 현재 맵의 상태

③ 반환

- bool: 명령이 잘 실행되었으면 true, 그 반대라면 false 반환

④ 결과

- 유저의 명령어를 입력 받고 명령을 수행

⑤ 설명

- "종료" 명령은 실행 시 action함수를 호출하지 않고 바로 종료
- 나머지 명령은 action함수로 들어가 명령을 실행
- 이동 명령 시 move함수를 호출하고 유효하지 않은 이동(반환값 false)은 false를 반환하여 명령어를 다시 입력 받게 작동
- 지도 명령 시 displayMap함수를 호출하여 맵을 출력하고 true를 반환

vii. 좌표 상 아이템, 포션, 적, 목적지를 발견하면 상호작용하는 함수

① 스크린샷

```
// 유저 위치의 타일이 특수한 타일인지 확인하고 상호작용하는 함수
bool checkState(int map[][mapX], int *user_x, int *user_y, int *userHP) {
    switch (map[*user_y][*user_x])
    {
    case 1:
        // 무기/갑옷을 발견
        cout << "아이템을 발견했다! 의외의 행운이 따르는군?" << endl;
        return false;
    case 2:
        // 적을 조우
        cout << "적을 맞닥뜨렸다..! 무찌르자!" << endl;
        cout << "적을 쓰러뜨렸지만 HP를 2 잃었다.: " << *userHP << " -> " << *userHP-2 << endl;
        *userHP -= 2;
        return false;
    case 3:
        // 포션을 발견
        cout << "포션! 의외로 맛있는걸??" << endl;
        cout << "HP가 2 회복되었다.: " << *userHP << " -> " << *userHP+2 << endl;
        *userHP += 2;
        return false;
    case 4:
        // 타일의 값이 4인 경우 도착한 것이므로 true를 return해서 게임을 종료
        return true;
    default:
        return false;
    }
}
```

② 입력
- *user_x: 유저의 현재 X좌표
- *user_y: 유저의 현재 Y좌표
- *userHP: 유저의 현재 HP
- map[][]: 현재 맵의 상태

③ 반환
- bool: 목적지에 도착한 경우 true, 나머지의 경우 false.

④ 결과
- 유저가 현재 위치한 X, Y좌표에 특수한 이벤트가 있는지 확인하고, 있다면 상호작용
- 포션은 HP + 2
- 적은 HP - 2

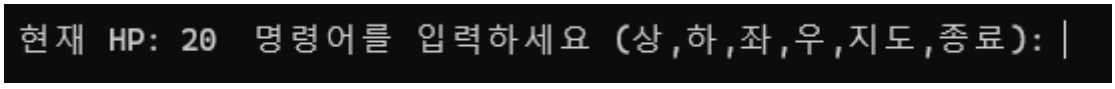
⑤ 설명
- map의 값이 1인 경우 아이템, 2인 경우 적, 3인 경우 포션, 4인 경우 목적지
- switch 구문으로 map의 값 판별

4. 테스트

A. 기능 별 테스트

i. 사용자에서 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받기

① 명령어 입력 받을 때마다 HP를 함께 출력

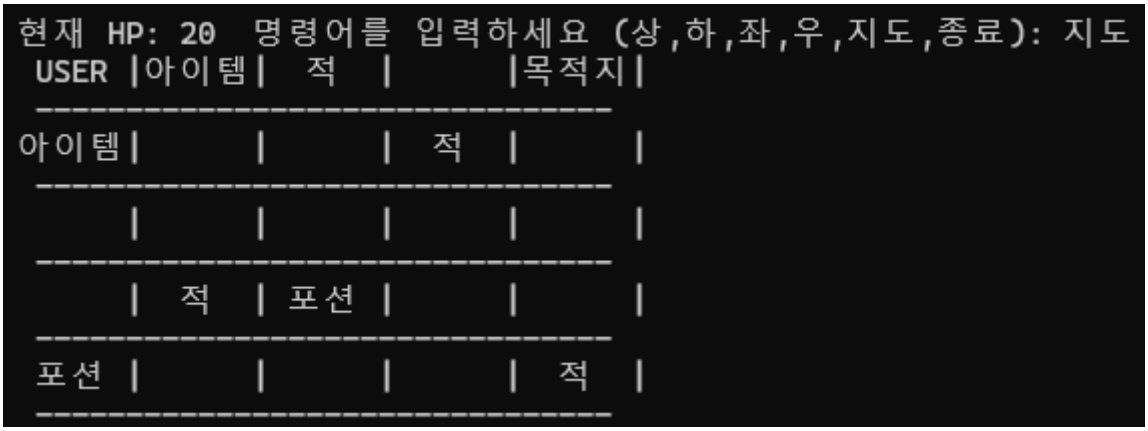


ii. 유저는 HP 20을 가지고 게임 시작

① HP가 0이 되면 "체력이 다 떨어졌습니다. 실패...."를 출력하고 종료

iii. 입력 받은 명령어에 따라 각 명령을 수행하기

① 지도: 현재 지도를 출력



② 종료: 게임을 종료

```
현재 HP: 20  명령어를 입력하세요 (상,하,좌,우,지도,종료): 종료
종료합니다.Press any key to continue . . .
```

③ 이동: 좌표 유효 검증 후 이동

- 좌표가 유효하지 않을 시 메시지 출력 후 재입력 대기
- 이동 시마다 HP가 1씩 감소

```
현재 HP: 20  명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
너무 어두워서 이동하기가 어렵다.: 20 -> 19
아래로 한 칸 내려갑니다.
      |아이템|  적  |      |목적지|
 ------------------------------
 USER |      |      |  적  |      |
 ------------------------------
      |      |      |      |      |
 ------------------------------
      |  적  | 포션 |      |      |
 ------------------------------
 포션 |      |      |      |  적  |
 ------------------------------
아이템을 발견했다! 의외의 행운이 따르는군?
```

iv. 좌표 상 아이템, 포션, 적, 목적지를 발견하면 상호작용

① 아이템

- "아이템을 발견했다! 의외의 행운이 따르는군?" 출력

```
아이템을 발견했다! 의외의 행운이 따르는군?
```

② 포션

- "포션! 의외로 맛있는걸??" 출력
- 유저의 HP가 2 증가

```
포션! 의외로 맛있는걸??
HP가 2 회복되었다.: 16 -> 18
```

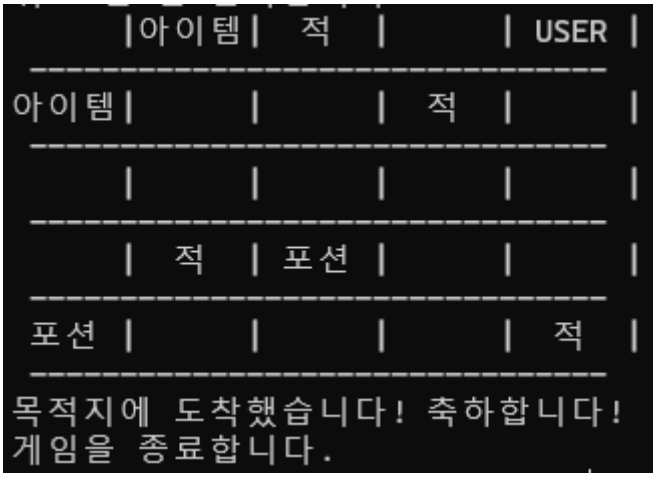
③ 적

- "적을 맞닥뜨렸다...! 무찌르자!" 출력
- 유저의 HP가 2 감소

```
적을 맞닥뜨렸다..! 무찌르자!
적을 쓰러뜨렸지만 HP를 2 잃었다.: 16 -> 14
```

④ 목적지

- 도달 시 "목적지에 도착했습니다! 축하합니다!"을 출력하고 종료



5. 결과 및 결론

A. 프로젝트 결과

i. MUD-game을 제작하였음.

ii. base code에서 함수들의 기능을 더욱 명확하고 단순하게 변경하였음

iii. 따라서 main 함수가 더욱 간단해짐

B. 느낀 점

i. move, action 함수를 분리하는 것에 시간이 조금 걸렸습니다.

ii. 함수들의 의존성에 대해서 더욱 고민해볼 수 있었습니다. 현재의 코드는 의존성이 높지 않지만 나중에 더 추가된다면 객체를 추가하는 방법을 고려해봐야 할 것 같습니다.