

C++프로그래밍 및 실습

# C++로 구현한

# Rouge

진척 보고서 #1

제출일자: 2024/11/17

제출자명: 손유채

제출자 학번: 214940

# 1. 프로젝트 목표

## 1) 배경 및 필요성

다양한 게임을 접하며 자연스럽게 프로그래밍에 관심을 가지게 되었습니다. 게임을 즐길수록 저만의 게임을 만들어 세상에 선보이고 싶다는 목표가 생겼습니다. 특히 여러 장르 중에서도 매 플레이 마다 새로운 경험을 하게 하는 Roguelike 게임을 만들고 싶었습니다.

Roguelike의 시초라 할 수 있는 "Rogue"라는 게임이 있는데 이를 제 색깔로 재해석해 보고싶어 저만의 개성을 담은 Roguelike게임을 구현해 보고자 이 프로젝트의 주제로 잡았습니다.

## 2) 프로젝트 목표

로그의 특징인 무작위성 맵, 턴 기반 전투방식, 플레이어의 지속적인 죽음의 특징을 게임에 녹여내어 구현하는 것이 목표입니다

## 3) 차별점

기존의 "로그"는 여러층을 내려가는 방식이지만 이는 게임을 길게 하기에 1층만 나오게하되 매 플레이마다 맵을 바꿔게 할 예정입니다.

## **2. 기능 계획**

### **1) 무작위 맵 생성**

- 맵을 여러조각으로 만들어놓고 매 플레이 할때마다 맵 조각을 합쳐서 매 플레이마다 새로운 맵을 만드는 방식으로 무작위 맵을 만든다.

(1) 맵 생성

(2) 맵 구현

매 플레이시 랜덤하게 맵 조각을 불러와 연결하여 맵을 만듦

### **2) 플레이어 및 적들의 이동 구현**

- 방향키 입력을 받아 플레이어의 이동을 구현하고 이를 1턴으로 잡아 적들이 각 성향에 따라 이동하게 구현하게 한다

(1) 플레이어 이동 구현

(2) 적 이동 구현

### **3) 매 시도로 나오는 변화**

- 플레이어의 죽음이나 목표달성으로 아이템의 해금, 맵의 변화, 플레이 난이도를 조정.

### **4) 시작화면, 시작장소 구현**

- 게임 시작 시 바로 시작하는게 아닌 시작장소와 시작화면을 구현하여 게임에 대한 설명이나 시작 상태를 알 수 있게 해줌.

(1) 시작화면 만들기

(2) 시작장소 만들기

## 5) 다양한 옵션 구현

- 특정키 입력 시 일시정지, 플레이어 상태 등 게임에 필요한 옵션을 구현.

(1) 일시정지 구현

(2) 플레이어 상태 확인

### 3. 진척사항

#### 1) 기능 구현 - 최대한 헤더파일을 이용하려 했음

##### (1) 맵 생성

1. map.h

```
class Position{
public:
    int x;
    int y;
};

// 클래스로 장소에 대한 기본정보 구현
class Map
{
public:
    int xPosition; // 장소가 시작되는 x좌표
    int yPosition; // 장소가 시작되는 y좌표
    int height;    // 장소의 높이(세로길이)
    int width;     // 장소의 길이(가로길이)
    Position door[4]; // 장소의 문
    // 0=바닥, 1=벽, 2=문
    int **Attribute; // 장소의 속성

    Map(int x = 0, int y = 0, int h = 0, int w = 0); // 생성자 선언
    ~Map(); // 소멸자 선언
};
```

class Position으로 장소의 상하좌우에 문의 위치를 좌표를 만들고

class Map로 장소의 각 요소를 넣음

## 2. map.cpp

```
void SetAttribute(Map *map, int x, int y, int A)
{
    if (x >= 0 && x < map->height && y >= 0 && y < map->width)
    {
        map->Attribute[x][y] = A; // int A에 적은 속성값 부여
    }
}
// 호출만 하므로 const사용으로 변화를 주지않음
int GetAttribute(const Map *map, int x, int y)
{
    if (x >= 0 && x < map->height && y >= 0 && y < map->width)
    {
        return map->Attribute[x][y];
    }
    return -1; // 유효하지 않은 좌표일 경우
}
```

Void SetAttribute

입력 – Map \*map : 맵 클래스, int x : 장소의 x좌표, int y : 장소의 y좌표, int A : 속성 number

반환 값 – 없음

결과 – 특정좌표가 map을 안에 있는 좌표면 그 좌표에 속성 부여

int GetAttribute

입력 – const Map \*map : 맵 클래스, int x : 장소의 x좌표, int y : 장소의 y좌표

반환 값 – 속성값

결과 – 특정좌표가 map안에 있는 좌표면 그 좌표의 속성값 반환

```
// 특정 좌표로 커서 옮기기
void Gotxy(int x, int y)
{
    if (x < 0)
        x = 0;
    if (y < 0)
        y = 0;
    COORD Pos = {static_cast<SHORT>(y), static_cast<SHORT>(x)};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
```

Void Gotxy

입력 – int x : 콘솔의 x좌표, int y : 콘솔의 y좌표

반환 값 – 없음

결과 – 콘솔에도 좌표가 있는데 그 좌표로 커서를 이동하여 출력에 도움을 줌

```
Map **mapSetup()
{
    int rows = 3; // 장소 수 설정
    Map **maps = new Map *[rows];
    maps[0] = createMap(13, 13, 6, 8);
    maps[1] = createMap(40, 2, 6, 8);
    maps[2] = createMap(40, 10, 6, 8);
    connectDoor(maps[0]->door[3], maps[2]->door[1]);
    return maps;
}
```

Map \*\*mapSetup

입력 – 없음

반환 값 – 새로운 Map객체 생성

결과 – 장소를 만들기 위한 프리셋을 만듦, 추후에 랜덤 장소 만들기 방식 채용 예정

```

Map *createMap(int x, int y, int h, int w)
{
    Map *newmap = new Map(x, y, h, w);

    newmap->xPosition = x;
    newmap->yPosition = y;
    newmap->height = h;
    newmap->width = w;
    // 문을 랜덤한 위치에 만들
    srand(time(NULL));
    // 위쪽문
    newmap->door[0].x = rand() % (w - 2) + (newmap->xPosition) + 1;
    newmap->door[0].y = newmap->yPosition;

    // 좌측문
    newmap->door[1].y = rand() % (h - 2) + (newmap->yPosition) + 1;
    newmap->door[1].x = newmap->xPosition;

    // 밑 문
    newmap->door[2].x = rand() % (w - 2) + (newmap->xPosition);
    newmap->door[2].y = (newmap->yPosition) + (newmap->height);
    // 우측문
    newmap->door[3].y = rand() % (h - 2) + (newmap->yPosition) + 1;
    newmap->door[3].x = (newmap->xPosition) + (newmap->width) - 1;

    return newmap;
}

```

Map \*creatMap

입력 – int x : 만들 장소의 콘솔 상 x좌표 시작점, int y : 만들 장소의 콘솔 상 y좌표 시작점, int h : 만들 장소의 세로 길이, int w : 만들 장소의 가로 길이

반환 값 – 새로운 장소 객체

결과 – 새로운 장소 객체에 입력 값을 대입해 만들고 srand, rand를 이용해 난수로 장소의 출입문을 할 때 마다 무작위로 생성



```

void drawMap(Map *R)
{
    int x, y;

    // 장소 생성시 위 아래 -----형태로 출력
    for (x = R->xPosition; x < (R->xPosition) + (R->width); x++)
    {
        Gotxy(R->yPosition, x);
        cout << "-";
        SetAttribute(R, R->yPosition - R->yPosition, x - R->xPosition, 1);
        Gotxy((R->yPosition) + (R->height), x);
        cout << "-";
        SetAttribute(R, (R->yPosition) + (R->height) + 1 - R->yPosition, x - R->xPosition, 1);
    }
    // 장소 생성시 좌우 |....| 형태로 출력
    for (y = R->yPosition + 1; y < (R->yPosition) + (R->height); y++)
    {
        Gotxy(y, R->xPosition);
        cout << "|";
        SetAttribute(R, y - R->yPosition, 0, 1);
        Gotxy(y, (R->xPosition) + (R->width) - 1);
        cout << "|";
        SetAttribute(R, y - R->yPosition, (R->width) - 1, 1);
        for (x = R->xPosition + 1; x < (R->xPosition) + (R->width - 1); x++)
        {
            Gotxy(y, x);
            cout << ".";
        }
    }
    for (int i = 0; i < 4; i++)
    {
        Gotxy(R->door[i].y, R->door[i].x);
        cout << "D";
        SetAttribute(R, R->door[i].y - R->yPosition, R->door[i].x - R->xPosition, 2);
    }
}

```

void drawMap

입력 - Map \*R : 구현할 장소객체

반환 값 - 없음

결과 - 장소 객체 있는 콘솔상의 x, y좌표 위치와 높이,길이, 문의 위치를 받아와 장소를 출력, 이때 벽 부분, 문 부분, 필드 부분에 맞는 속성을 부여

### 3. player.h

```
class Player
{
public:
    int xPosition, yPosition, health;
    Player();
};
```

class Player으로 유저의 정보를 만들

### 4. player.cpp

```
Player::Player() : xPosition(0), yPosition(0), health(20) {}

Player *playerSetup()
{
    Player *newPlayer;
    newPlayer = new Player;

    newPlayer->xPosition = 14;
    newPlayer->yPosition = 14;
    newPlayer->health = 20;

    Gotxy(newPlayer->xPosition, newPlayer->yPosition);
    cout << "@"; // 유저를 나타내는 문자

    return newPlayer;
}
```

입력 – Player \*playerSetup : 구현할 장소객체

반환 값 – 없음

결과 – 장소 객체 있는 콘솔상의 x, y좌표 위치와 높이,길이, 문의 위치를 받아와 장소를 출력, 이때 벽 부분, 문 부분, 필드 부분에 맞는 속성을 부여

## 5. command.cpp

```
// 유저 움직임 함수
void playerMove(int x, int y, Player *user)
{
    Gotxy(user->yPosition, user->xPosition);
    cout << ".";
    user->xPosition = x;
    user->yPosition = y;
    Gotxy(user->yPosition, user->xPosition);
    cout << "@";
}
```

void playerMove

입력 – int y, int x : 유저가 이동할 x,y위치, Player \*user : 유저의 데이터

반환 값 – 없음

결과 – 장소 객체 있는 콘솔상의 x, y좌표 위치와 높이,길이, 문의 위치를 받아와 장소를 출력, 이때 벽 부분, 문 부분, 필드 부분에 맞는 속성을 부여

```
// 장소 유효성 검사
void checkPosition(int newy, int newx, Player *user, Map *map)
{
    // 특정 좌표의 속성 확인
    int Attribute = GetAttribute(map, newy - map->yPosition, newx - map->xPosition);

    // 속성이 '0' (통로)일 경우에만 이동
    if (Attribute == 0) {
        playerMove(newy, newx, user);
    }
}
```

Void checkPosition

입력 – int newy, int newx : 유저가 이동할 x,y위치, Player \*user : 유저의 데이터, Map \*map : 장소 데이터

반환 값 – 없음

결과 – 이동할 좌표의 속성을 확인 후 이동가능한 속성이면 playerMove로 이동

```

void Handleinput(int input, Player *user, Map *map)
{
    int newy;
    int newx;
    switch (input)
    {
        case 'w':
        case 'W':
            newy = user->yPosition - 1;
            newx = user->xPosition;
            break;
        case 's':
        case 'S':
            newy = user->yPosition + 1;
            newx = user->xPosition;

            break;
        case 'a':
        case 'A':
            newy = user->yPosition;
            newx = user->xPosition - 1;

            break;
        case 'd':
        case 'D':
            newy = user->yPosition;
            newx = user->xPosition + 1;

            break;
        default:
            break;
    }

    checkPosition(newx, newy, user, map);
}

```

Void Hendleinput

입력 – int input : 입력 받은 키보드 키, Player \*user : 유저의 데이터, Map \*map : 장소 데이터

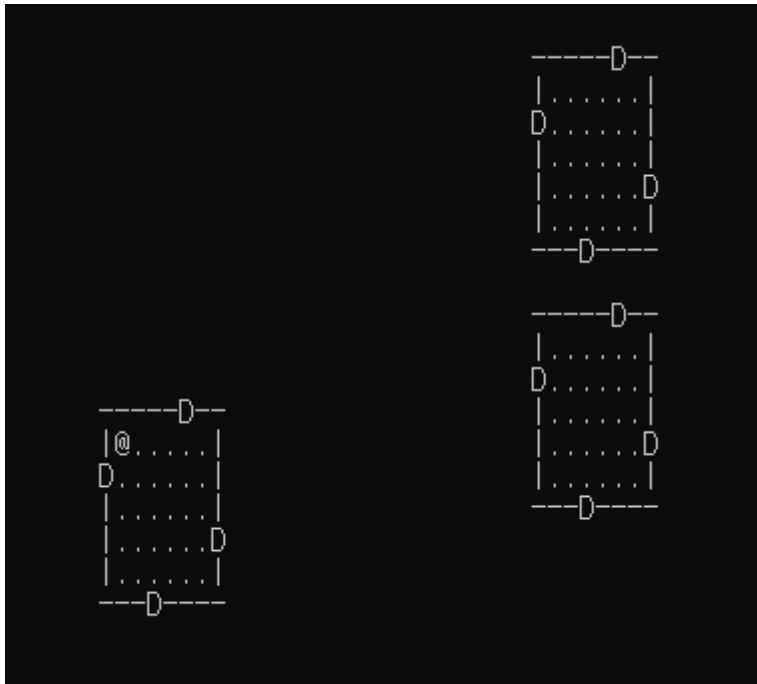
반환 값 – 없음

결과 – 입력받은 키를 확인후 새로운 이동좌표 newx, newy로 이동할 좌표를 저장 후 이동 좌표 확인

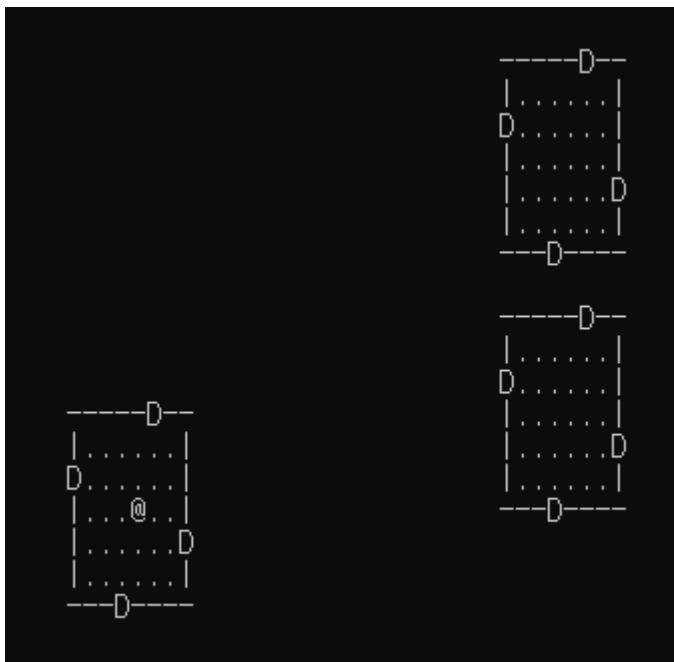
## 2) 테스트 결과

### (1) 장소 생성과 문 생성

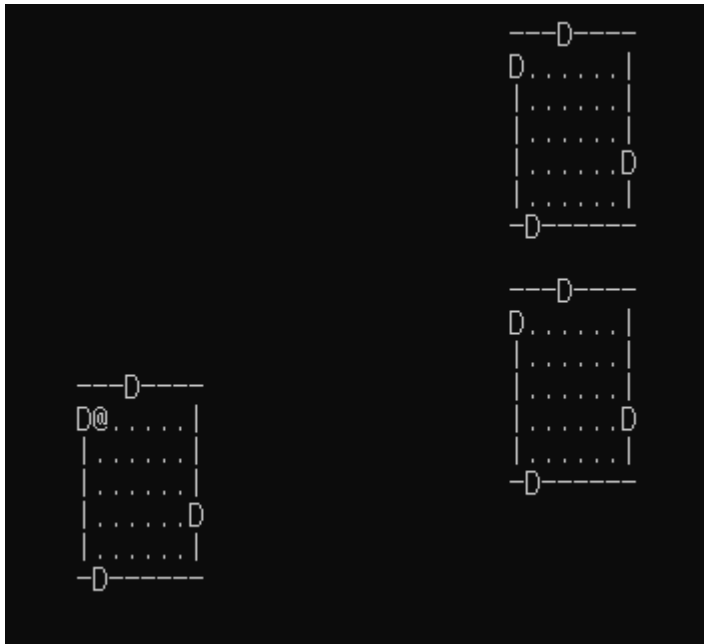
- 실행파일 실행 시 장소를 생성(. , | , \_ 로 장소를 구성, D로 문 구성, @로 유저 구성)



- wasd로 @인 유저 이동



- 실행파일 실행 시 마다 장소에 문이 랜덤 생성



## 4. 계획 대비 변경 사항

### 1) 변경 내역 제목

이전 - 유저의 이동 방식 구현을 map을 만든 다음 하려고 시도

이후 - map생성과 유저의 이동방식 구현을 동시에 진행하고 무작위 map생성 일정을 뒤로 미룸

사유 - map생성을 해보니 원하는 방향으로 설계된 것인지 확인을 위해 테스트용 유저를 만들 필요가 있었고 무작위로 map생성 구현이 생각보다 시간이 걸려 순서를 바꾸기로 함

## 5. 프로젝트 일정

[illegible]