

First about the whole production of the game, instead of doing a text-based, I visualize the whole game process, so it's gonna be a little different from the requirements. And basically put all my effort onto the gameplay so the content's gonna be a little weak.

GAME LOGIC

The whole game's concept is about escaping the dungeon. The goal is to find the exit room and get out. There're 6 room types.

- Start Room

This's just a Room the player's initially put.

- Exit Room

Our goal is to get to this room.

- Empty Room

self-explanatory name.

- Monster Room

This room will consist of random amount of monsters.

- Npc Room

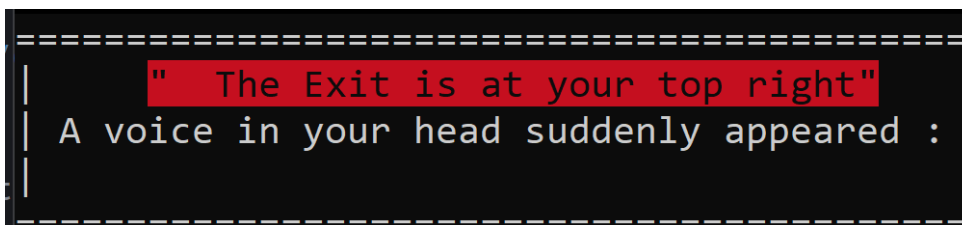
This room will have a NPC standing in the middle of the room.

- Chest Room

This room will have a Chest in the middle of the room.

By interacting would result in either a monster or a Item.

The winning condition is to get to the exit somewhere on the map. By defeating the monsters, there'd be a hint on where the exit room is.



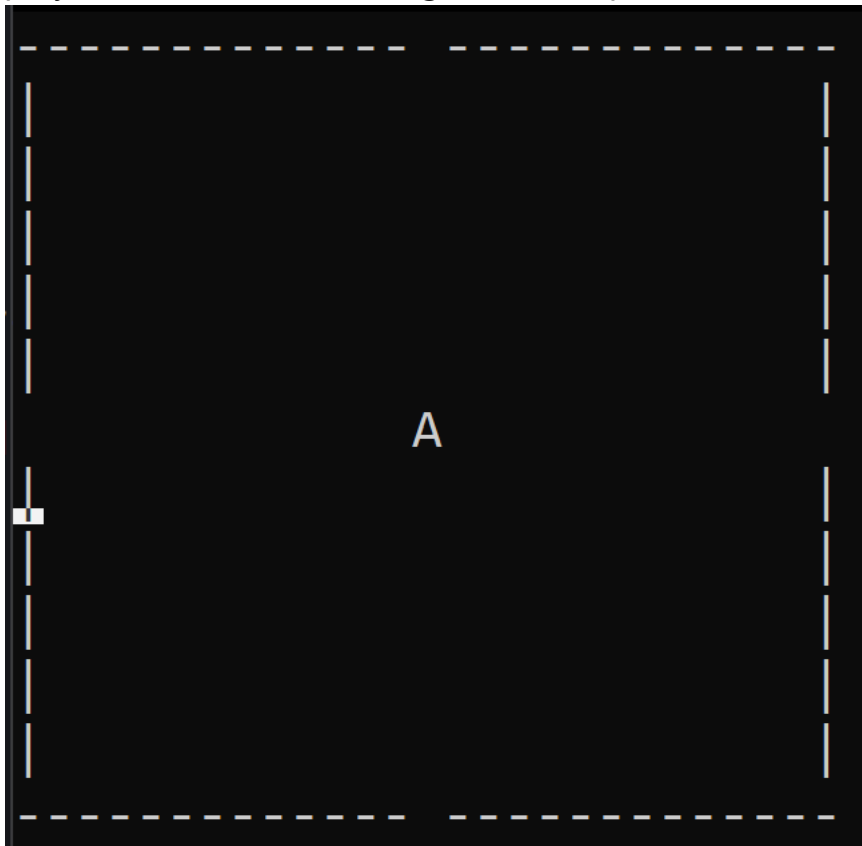
Also the exit room is locked, the player's gonna need a key to be able to enter. The key would be randomly

dropped from monsters.

Also there's a timer of 3 mins in which the player needs to escape from the dungeon, the player will die when time's up. The timer is also implemented by multithreading.

MOVEMENT

First the movement, I visualize the whole map and the player as an arrow walking in the map.



You can walk up, down, left, right, and spacebar to interact with any object on the map.



The implementation is basically based on the library `conio.h`, it provides a function called `getch()` which enables me to process the input from keyboard without user hitting `'\n'` first.

```
char c;
while(c = getch()) {
    if(c == 'w') {
        // move up
    }
    // things like that.
}
```

The whole game Map is 11 * 11 and the roomsize is 11 * 25. Player can enter other room by standing at the door hitting spacebar. When there's any object at the place the player's going or the room player's going is outofbound, there will be no responses.

SHOWING STATUS

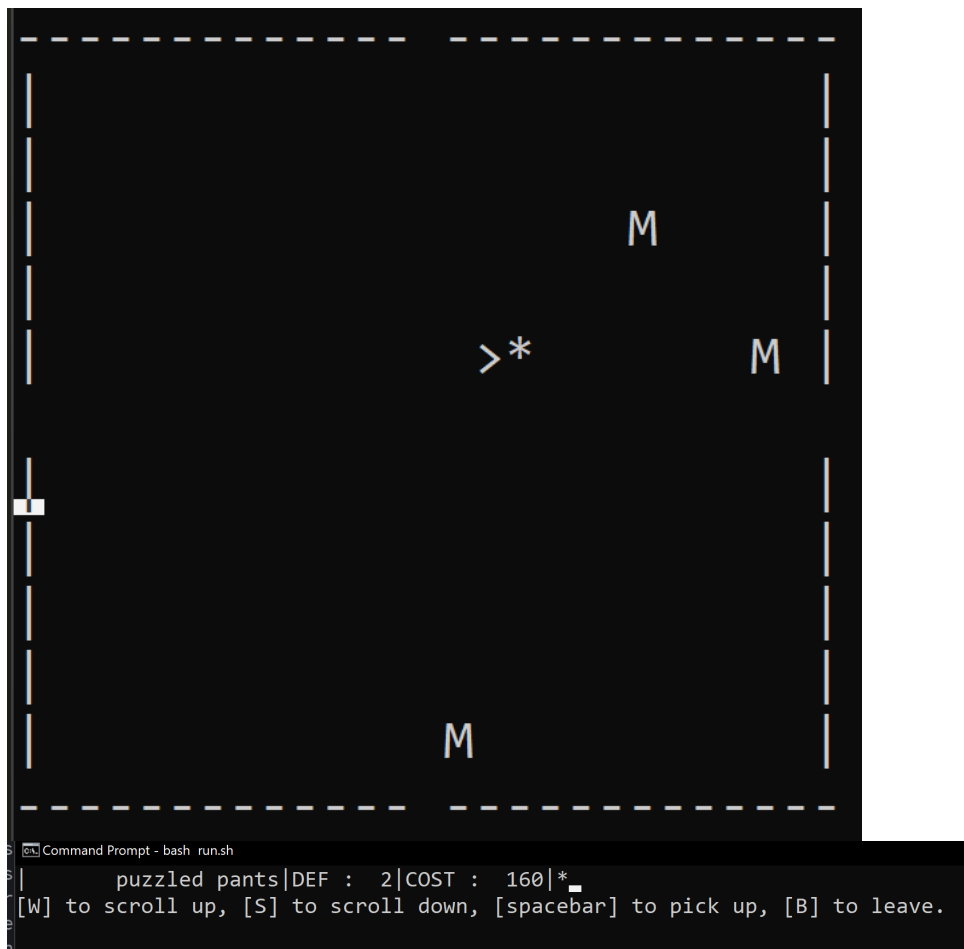
Player can check their status whenever they want when wandering on the map. As I stated on the movement, I have a function listening for inputs continuously. And when the player press E, the game will invoke the player's function `showstatus`, showing all current status of player.

```
=====
Name : test
HP : 
EXP :  ( lvl : 1 )
ATK : 1000
$1000
=====
Press b to return
```

Also it listens for input, when player press B, it'll go back to the game.

PICK UP ITEMS

When opening chest or defeating monster, there'd be random drop item, and it'll be showed as a '*' on the map. Hitting spacebar to intereact with it, a drop item list would appear.



Use w, a, s, d to choose between items and spacebar to pickup items. B to go back to the game. Also when all the items are picked up, the '*' sign will disappear.

- Start Room

This's just a Room the player's initially put.

- Exit Room

Our goal is to get to this room.

- Empty Room

self-explanatory name.

- Monster Room

This room will consist of random amount of monsters.

- Npc Room

This room will have a NPC standing in the middle of the room.

- Chest Room

This room will have a Chest in the middle of the room.
By interacting would result in either a monster or a Item.

The winning condition is to get to the exit somewhere on the map. By defeating the monsters, there'd be a hint on where the exit room is.





Also the exit room is locked, the player's gonna need a key to be able to enter. The key would be randomly dropped from monsters.

Also there's a timer of 3 mins in which the player needs to escape from the dungeon, the player will die when time's up. The timer is also implemented by multithreading.

I hate turn based-battle. I think those are very boring and unnecessary and should all be deleted. But there're several games that manages to make it fun, for example :

undertale . So I tried to imitate the way it's way of battling : dodging. When encountering a monster and entering the combat, it'll first popped out a option for

player to escape or fight.

```
=====
|
| Do you wanna Fight or Escape? |
| [f] for fight, [e] for escape |
|
=====
You :
=====
Name : test
HP : 
EXP :   ( lvl : 1 )
ATK : 1000
$1057
=====
Monster :
=====
Name : relieved beast
LVL : 4
HP :  (4000 / 4000)
=====
```

And then there are two phases:

- defense phase

Like I said, I imitate the way undertale's handles combat. Basically just dodging the attack in a amount of time.

be $hitCnt * ATK$

```
Select Command Prompt - bash run.sh

      3 secs remaining
      4 hits!!

You :
=====
Name : test
HP : 
EXP :  ( lvl : 1 )
ATK : 1000
$1057
=====
The Monster :
=====
Name : relieved beast
LVL : 4
HP :  (4000 / 4000)
=====
Press spacebar to hit the monster
```

The above phases both has a timer, and a input listener, all implemented by multithreading in the library `thread` .

```
1 | thread {timer, this}.detach();
```

This is a quite hard concept and it's also my first attempt to implement it so it took me quite a lot of time.

NPC

My NPC has two options : shopping and recovering Hp. Recovering Hp is just a function to recover the Hp to the maximum. And about the shopping, everytime the player

enters a NPC room, it will randomly generate 10 items to put on the shelf for sell.

```
* Shop
Recover Hp
[W] and [S] to scroll up and down, [spacebar] to choose, [b] to leave npc
-----
| delightful weapon|ATK : 12|COST : 1250|*|
|      bad boots  |DEF :  2|COST :  100| |
|    alert pants  |DEF :  1|COST :   50| |
| courageous chest|DEF :  3|COST :  200| |
| blushing helmet |DEF :  2|COST :  150| |
| careful weapon  |ATK :  2|COST :  200| |
| delightful boots |DEF :  3|COST :  250| |
|      bad pants  |DEF :  2|COST :  100| |
|    alert chest  |DEF :  1|COST :   50| |
| delightful potion|REC :  3|COST :   60| |
[W] and [S] to scroll up and down, [spacebar] to purchase, [B] to leave
```

BACKPACK SYSTEM

The backpack system implement the inventory for player. It shows the possession and equipment of the player.

```
EQUIPED :
| HEAD|      lucky helmet|
| CHEST|      tired chest|
| PANTS|      EMPTY|
| BOOTS|    gifted boots|
| WEAPON| combative weapon|
=====
Owned :
|      gifted boots|DEF :  1|COST :  30|ON|
|    lucky helmet |DEF :  1|COST :  75|ON|
|      tired chest|DEF :  1|COST :  60|ON|
| uglyiest boots  |DEF :  1|COST :   5| |
| combative weapon|ATK :  6|COST : 683|ON|
| graceful potion |REC :  3|COST :  60|*|
[W] and [S] to scroll up and down, [spacebar] to purchase, [B] to leave, [X] to dispose item
```

When equipping the armours or weapons, it'll invoke a function to modify the player's status. Spacebar on a unequipped armour or weapon can equip it, equipped vice versa. Also can press X to dispose item. And the key to the exit and potions will also be listed here. Spacebar on potion can drink it to recover health.

OTHER DETAILS

- About the virtual functions and inheritance, I apply them on the Room parts. As stated in game logic, there are 6 types of rooms. They are all inherited from the class Room

```

class Room
{
private:
    char facingTable[4] = {'A', 'V', '<', '>'};
    int X, Y;
public:
    virtual void enterRoom(Player&, char(*)[sizeY + 2], int(*obj)[sizeY + 2]) = 0;
    void initializeMap(Player&, char(*)[sizeY + 2], int(*obj)[sizeY + 2]);
    virtual string getRoomType() = 0;
};

```

And I have a pointer declared in dungeon look like this

```

1 | Room *gm[105][105];

```

before the game starts, I'll first decide what a room's type is and create a instance to cast into the (*gm) so when entering a room, I can basically just call

```

1 | gm[x][y] -> enterRoom();
2 | // or getting name;
3 | gm[x][y] -> getRoomType();

```

and the pure virtual function will be implemented by it's child.

- about handling visualization

keep calling `system("cls")` would cause lagging so I replace it with the following function

```

1 | const HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
2 |
3 | // Function supporting printing Room
4 | static void setCursorPosition(int y, int x) {
5 |     std::cout.flush();
6 |     COORD coord = { (SHORT)x, (SHORT)y };
7 |     SetConsoleCursorPosition(hOut, coord);
8 | }

```

It changes the cursor of the current output, thus I can just modify the place where I want to modify. For example, when player moves, I can just change (x, y) and (dx, dy); and about the coloring, I call the functions :

```

1 // set output red
2 void setOutputRed() {
3     SetConsoleTextAttribute(hOut, BACKGROUND_RED);
4 }
5
6 // set output white
7 void setOutputWhite() {
8     SetConsoleTextAttribute(hOut, BACKGROUND_RED | BACKGROUND_BLUE | BACKGROUND_GREEN);
9 }
10
11 // set output green
12 void setOutputGreen() {
13     SetConsoleTextAttribute(hOut, BACKGROUND_GREEN);
14 }
15
16 // set output original
17 void setOutputOriginal() {
18     SetConsoleTextAttribute(hOut, FOREGROUND_RED | FOREGROUND_BLUE | FOREGROUND_GREEN);
19 }

```

- about randomness

since I add a lot of random elements into the game, and the seed is frequently the same so I got a little lack of randomness, here's two ways I resolve this problem.

- randomly seeding

basically just randomly choose variables to use as seed.

- better random function

I use the `mt19937` to generate random number.

```

1 int randGen(int left, int right) {
2
3     using clk = chrono::high_resolution_clock;
4     using rand_int = uniform_int_distribution<int>;
5     using rand_double = uniform_real_distribution<double>;
6
7     mt19937 rng(clk::now().time_since_epoch().count());
8     return rand_int(left, right)(rng);
9 }

```

- about the multithreading

There's too much I wanna complain about this, if I thoroughly write out the obstacles I've encountered I guess this report's gonna be two times the size so I'm gonna pass.

- Some other minor functions I implemented

- Pausing and exiting

by pressing P can player pause the game and X for quitting the game.

- replacing equipped item
when I equip a armour, other armour would be disequipped.
- leveling
defeating a monster can get exps and player can level up to gain more Hp and ATK
- mini map
there's a minimap to the right of the game showing the room player's been to and the room player's currently at.
- Also I wrote a bash to execute the game