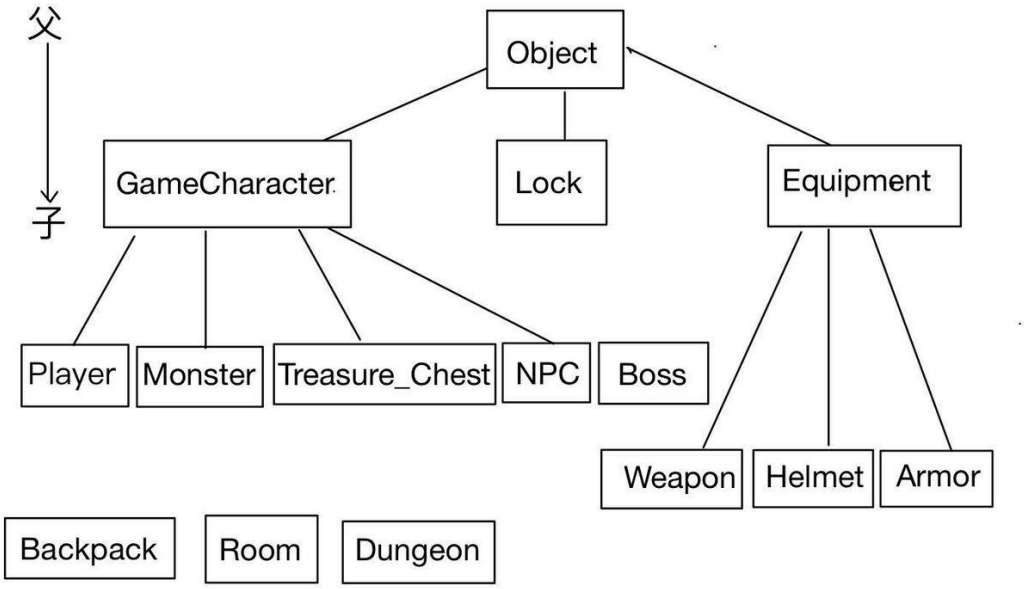
110550089 周冠辰

# 一、程式基本架構及運行原理




以上為我程式中所使用的各種類別及其繼承關係。

Dungeon 這個類別基本上是支撐著整個程式的架構，將各種類別及函式結合在一起運行。

以下為 Dungeon 內的幾個 function

| | |
|---|---|
| void createPlayer(); | 創建角色 |
| void createMap(); | 創建地圖 |
| void handleMovement(); | 處理房間間的移動 |
| void handleEvent(Object*); | 處理進入房間後所 |
| void startGame(); | 開始運行 |
| void chooseAction(); | 選擇動作 |
| bool checkGameLogic(); | 檢查是否結束遊戲 |
| void runDungeon(); | 運行地下城 |
| void showBackpack(); | 打開背包 |
| void printMap(); | 印出地圖 |

接著開始概略講解運作原理(省略大部分的 code)

首先是 main 裡面唯一被用到的 startGame()

```
startGame(){                開始遊戲
    void createMap();       藉由讀取 txt 檔裡面的資訊，建造出各種各樣的地圖
    void createPlayer();    將玩家的角色資訊給初始化
    void runDungeon();      開始運行
}
```

在開始後，會先建立地圖及初始化玩家，接著就開始在房間中做出各種選擇。

```
void Dungeon::runDungeon() {
    while (1) {
        this->chooseAction();                   //不斷的做出選擇
        if (this->checkGameLogic()) {           //每次選擇完就會檢查遊戲邏輯
            exit(0);                            //判斷是否達成特定條件而結束
        }
    }
}
chooseAction(){
    Switch(case){
    case ‘a’:   handleMovement(); handleEvent(Object*);
                //先進入下一個房間，然後開啟那個房間裡的事件
    case ‘b’:   player.triggerevent();
                //藉由 player 的 function 來印出角色資訊
    case ‘c’:   showBackpack();
                //這個 function 會去 call Backpack 裡面的 function
                //來印出背包內的物品及更換裝備
    case ‘d’:   printMap();
                //房間彼此間除了用 linklist 串起來外，還存在一個二維陣列裡面
                //就由讀取那個座標是否存在房間，以及讀取玩家的所在座標來
                //印出地圖
    }
}
```

While 迴圈會讓這個函式會不斷的重複運行已做出各種選擇直到達到特定條件而結束遊戲。

以上為大致上遊戲運行的原理，接下來講解我如何達到規定的要求。

# 二.各項要求如何實作

1. Movement

角色的移動是藉由 Dungeon 裡面的 handleMovement()和 handleEvent(Object*)所進行的。

handleMovement()

先用 if else 判別上下左右是否可以走並將結果存入一個 vector，並列印出選項讓玩家選擇。再用 if else 判別玩家前往的方向，將 Player 內存取的房間位址轉換到新的位址並轉換 Player 內存取的玩家所在座標讓 printMap 可以順利運行。

```
void Dungeon::handleMovement() {
    cout << "------------------------------------------------------------" << endl;
    cout << "go which way?" << endl;
    vector<char> order(6);
    int index = 0;
    char a = 'a';
    if (this->player.getCurrentRoom()->getUpRoom() != NULL) {
        cout << a << ":go up\n";
        a++;
        order[index] = 'u';
        index++;
    }
    if (this->player.getCurrentRoom()->getDownRoom() != NULL) {
        cout << a << ":go down\n";
        a++;
        order[index] = 'd';
        index++;
    }
    if (this->player.getCurrentRoom()->getLeftRoom() != NULL) {
        cout << a << ":go left\n";
        a++;
        order[index] = 'l';
        index++;
    }
    if (this->player.getCurrentRoom()->getRightRoom() != NULL) {
        cout << a << ":go right\n";
        a++;
        order[index] = 'r';
        index++;
    }
    char inptr;
    cin >> inptr;
    int num = 0;
    if (inptr >= 'a') {
        num = inptr - 'a';
    }
    else num = inptr - 'A';
    if (order[num] == 'u') {
        this->player.changeRoom(player.getCurrentRoom()->getUpRoom());
        this->player.setprex(player.getx());
        this->player.setprey(player.gety());
        this->player.setx(player.getx() - 1);
    }
    else if (order[num] == 'd') {
        this->player.changeRoom(player.getCurrentRoom()->getDownRoom());
        this->player.setprex(player.getx());
        this->player.setprey(player.gety());
        this->player.setx(player.getx() + 1);
    }
    else if (order[num] == 'l') {
        this->player.changeRoom(player.getCurrentRoom()->getLeftRoom());
        this->player.setprex(player.getx());
        this->player.setprey(player.gety());
        this->player.sety(player.gety() - 1);
    }
    else if (order[num] == 'r') {
        this->player.changeRoom(player.getCurrentRoom()->getRightRoom());
        this->player.setprex(player.getx());
        this->player.setprey(player.gety());
        this->player.sety(player.gety() + 1);
    }
    else cout << "wrong input\n";
}
```

handleEvent(Object*)

先用 if else 判別房間內是否有物品（房間內的物體是用 std::queue 去儲存），有的就開始將物品一個一個觸發事件，完成事件物品就會消失，特殊條件下，ex:玩家撤退、死掉、或擊敗最終 boss 的情況下，triggerEvent()會回傳 false 來結束這個 function。玩家死亡或擊敗最終 boss 的情況在結束這個 function 後會運行 checkGameLogic()而結束遊戲。玩家撤退的情況下，物體仍會留在房間裡面等待玩家下次進入。遇上 NPC 時，觸發完事件離開後，再次回來時 NPC 還會存在。

```
void Dungeon::handleEvent(Object* event) {
    if (!this->player.getCurrentRoom()->emptyobject()) {
        while (!this->player.getCurrentRoom()->emptyobject()) {
            if (this->player.getCurrentRoom()->getqueue().front()->triggerEvent(event)) {
                this->player.getCurrentRoom()->popobject();
            }
            else {
                //checkGameLogic();
                break;
            }
        }
    }
    else {
        cout << "------------------------------------------------------------" << endl;
        cout << "nothing is in the room\n";
    }
}
```

```
--------------------------------------
What you want to do:
a:move
b:show status
c:show backpack
d:show the map
a
--------------------------------------
go which way?
a:go down
b:go left
c:go right
a
--------------------------------------
You met a monster:Guma
--------------------------------------
Guma have: 20 hp
you have: 100 hp        5 potions
choose action:
a:hit the monster
b:use the potion for heal
c:use the potion on monster
d:do nothing
e:retreat to the previous room
```

(1)選擇移動

(2)印出可移動的方向並讓玩家選擇

(3)移動進入下一個房間後
會開始一個一個按照順序觸發每個 Object 對應的事件。

2. Showing Status

這個功能我直接利用 Player 繼承下來的 triggerEvent();

藉由逐一存 Player 存的資料及 Player 內 Backpack 存的資料來印出玩家狀態。

```
bool Player::triggerEvent(Object* player) {
    cout << "-----------------------------------------------------" << endl;
    cout << "Name: " << this->getName() << endl;
    cout << "Hp: " << this->getCurrentHealth() << "/" << this->getMaxHealth() << endl;
    cout << "ATK: " << this->getAttack() << endl;
    cout << "DEF: " << this->getDefense() << endl;
    cout << "Money: " << this->getmoney() << "$" << endl;
    cout << "Helmet:" << this->getBackpack()->getwhelmet()->getName()<<   endl;
    cout << "Armor:" << this->getBackpack()->getwarmor()->getName() << endl;
    cout << "LeftWeapon:" << this->getBackpack()->getwleftweapon()->getName() << endl;
    cout << "RightWeapon:" << this->getBackpack()->getwrightweapon()->getName() << endl;
    return true;
}
```

```
--------------------------
What you want to do:
a:move
b:show status
c:show backpack
d:show the map
b
--------------------------
Name: chouchou
Hp: 100/100
ATK: 10
DEF: 0
Money: 100$
Helmet:round_hat
Armor:old_clothes
LeftWeapon:old_shield
RightWeapon:old_sword
--------------------------
What you want to do:
a:move
b:show status
c:show backpack
d:show the map
```

(1)選擇 show Status

(2)開始印出各項資訊

(3)繼續做出下個選擇

3. Pick up item

利用各個裝備所屬類別的 virtual triggerEvent()來將裝備撿起，以 Weapon 為例:

```
bool Weapon::triggerEvent(Object* player){
    cout << "-------------------------------------------------" << endl;
    Player* maincharacter = dynamic_cast<Player*>(player);
    cout << "A weapon is on the floor" << endl;
    cout << "[ " << this->getName() << " +ATK " << this->getAttack() << " +DEF " << this->getDefense() << " +HP " << this->getHealth() << "
    cout << "(a).pick up\n";
    cout << "(b).throw it away\n";
    char deci;
    while (1) {
        cin >> deci;
        if (deci == 'a' || deci == 'A') {
            maincharacter->getBackpack()->addweapon(this);
            cout << "It is in your backpack now\n";
            break;
        }
        else if (deci == 'b' || deci == 'B') {
            break;
        }
        else {
            cout << "wrong input -> input your decision again\n";
        }
    }
    return true;
}
```

這個 function 傳入的值為 Player 所在的位址，才能將物品傳入玩家的背包裏面。先詢問玩家撿取意願，願意的話則放入背包，不願意的話則把它扔掉。(return true 會讓物品消失)

```
-------------------------------------------------
Calcute have: 5 hp
you have: 95 hp 5 potions
choose action:
a:hit the monster
b:use the potion for heal
c:use the potion on monster
d:do nothing
e:retreat to the previous room
a
-------------------------------------------------
you caused 5 on the monster.
the monster dead
-------------------------------------------------
A weapon is on the floor
[ longspear +ATK 15 +DEF 1 +HP 0 ]
(a).pick up
(b).throw it away
a
It is in your backpack now
-------------------------------------------------
What you want to do:
a:move
b:show status
c:show backpack
d:show the map
```

(1)把怪物打倒後房間內可能會留下裝備

(2)triggerEvent()會讓你選擇是否要撿起武器

(3)撿完後繼續做出下個選擇

## 4. Fighting System

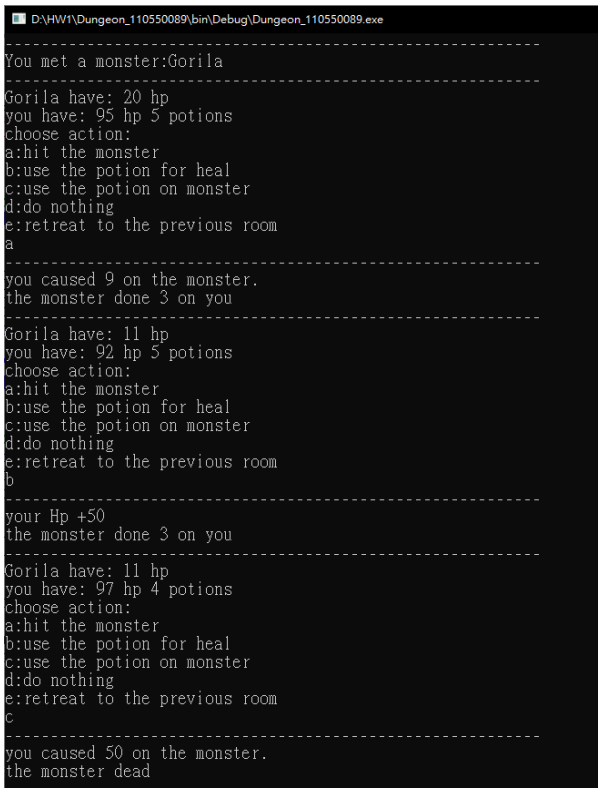
藉由 Monster 內的 virtual triggerEvent()將 fighting system 實作

```
bool Monster::triggerEvent(Object* player) {
    Player* maincharacter = dynamic_cast<Player*>(player);
    cout << "------------------------------------------------------------" << endl;
    cout << "You met a monster:"<<this->getName()<<endl;
    while (1) {
        cout << "------------------------------------------------------------" << endl;
        cout << this->getName() <<" have: " << this->getCurrentHealth() << " hp\n";
        cout << "you have: " << maincharacter->getCurrentHealth() << " hp\t" << maincharacter->getBackpack()->getpotion() << " potions\n";
        cout << "choose action:" << endl;
        cout << "a:hit the monster" << endl;
        cout << "b:use the potion for heal" << endl;
        cout << "c:use the potion on monster" << endl;
        cout << "d:do nothing" << endl;
        cout << "e:retreat to the previous room" << endl;
        char inptr;
        int damage = 0;
        cin >> inptr;
        cout << "------------------------------------------------------------" << endl;
        switch (inptr) {
        case 'a':
        case 'A':
            damage = maincharacter->getAttack() - this->getDefense();
            if (damage > 0) {
                this->takeDamage(damage);
                cout << "you caused " << damage << " on the monster.\n";
            }
            else {
                cout << "you caused 0 damage\n";
            }
            break;
        case 'b':
        case 'B':
```

```
        case 'b':
        case 'B':
            if (maincharacter->getBackpack()->getpotion() > 0) {
                cout << "your Hp +50\n";
                if (maincharacter->getCurrentHealth() + 50 >= maincharacter->getMaxHealth()) {
                    maincharacter->setCurrentHealth(maincharacter->getMaxHealth());
                    maincharacter->getBackpack()->setpotion(maincharacter->getBackpack()->getpotion() - 1);
                    break;
                }
                else {
                    maincharacter->setCurrentHealth(maincharacter->getCurrentHealth() + 50);
                    maincharacter->getBackpack()->setpotion(maincharacter->getBackpack()->getpotion() - 1);
                    break;
                }
            }
            else {
                cout << "you don't have any potion\n";
                break;
            }
        case 'c':
        case 'C':
            if (maincharacter->getBackpack()->getpotion() > 0) {
                this->takeDamage(50);
                cout << "you caused 50 on the monster.\n";
                maincharacter->getBackpack()->setpotion(maincharacter->getBackpack()->getpotion()-1);
                break;
            }
            else {
                cout << "you don't have any potion\n";
                break;
            }

        case 'd':
        case 'D':
```

```
        case 'd':
        case 'D':
            cout << "you didn't do any action\n";
            break;
        case 'e':
        case 'E':
            maincharacter->changeRoom(maincharacter->getPreviousRoom());
            maincharacter->setx(maincharacter->getprex());
            maincharacter->sety(maincharacter->getprey());
            return false;
        default:
            "wrong input\n";
            break;
        }
        if (this->checkIsDead()) {
            cout << "the monster dead\n";
            int getgold = this->getMaxHealth() / 2;
            maincharacter->addmoney(getgold);
            return true;
        }
        int _damage = this->getAttack() - maincharacter->getDefense();
        if (_damage < 0) {
            _damage = 0;
        }
        maincharacter->takeDamage(_damage);
        cout << "the monster done " << _damage << " on you\n";
        if (maincharacter->checkIsDead()) {
            return false;
        }
        continue;
    }
}
```

基本上就是用 while 迴圈不斷進行雙方的攻擊，在特定條件下在會停止這個 function(玩家死亡、怪物死亡或玩家撤退)，除了普通的攻擊外，可以使用藥水回血，使用藥水攻擊，撤退至前一格房間，甚至是什麼都不做。擊敗怪物後玩家還會獲得金錢。

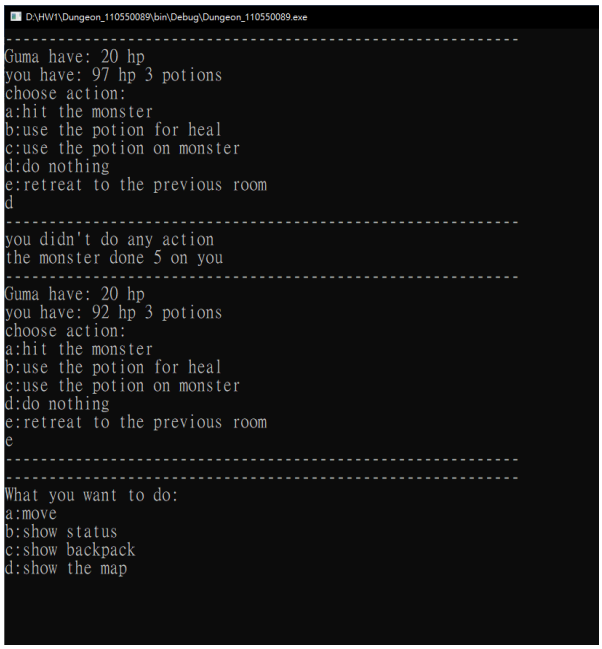



可以選擇普通攻擊

用藥水恢復

用藥水攻擊

怪獸死亡觸發房間的下個物件或是繼續做選擇。




可以選擇什麼都不做

也可以撤退到前一個房間

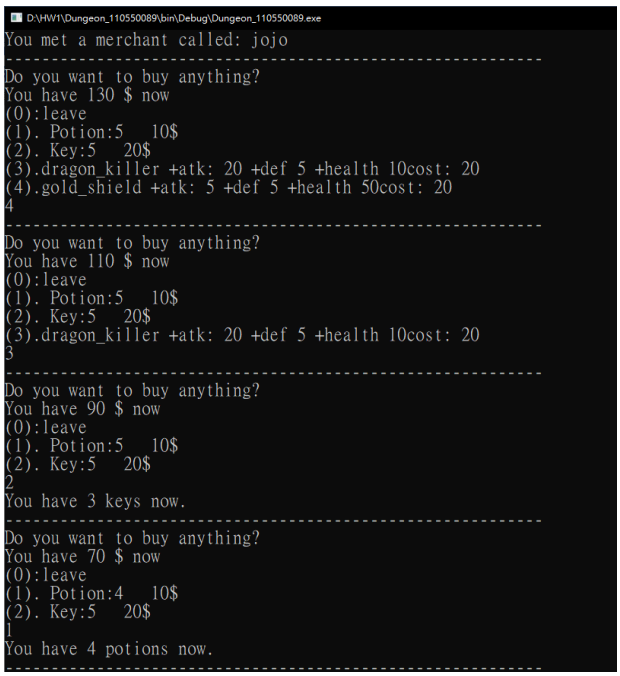
5. NPC

NPC 在我的遊戲內扮演的是商人的角色，會販賣鑰匙、藥水及裝備。販賣系統藉由 NPC 的 virtual triggerEvent()來實作

```
bool NPC::triggerEvent(Object* player) {
    Player* maincharacter = dynamic_cast<Player*>(player);
    cout << "You met a merchant called: " <<this->getName()<< endl;
    while (1) {
        cout << "------------------------------------------------------------" << endl;
        cout << "Do you want to buy anything?\n";
        cout << "You have " << maincharacter->getmoney() << " $ now\n";
        cout << "(0):leave\n";
        this->listCommodity();
        int act;
        cin >> act;
        if (act < 0 || act >= commodity.size() + 3) {
            cout << "wrong input\n";
            continue;
        }
        if (act == 0) {
            return false;
        }
        else if (act == 1) {
            if (this->numofpotion > 0) {
                if (maincharacter->getmoney() >= 10) {
                    maincharacter->getBackpack()->setpotion(maincharacter->getBackpack()->getpotion() + 1);
                    cout << "You have " << maincharacter->getBackpack()->getpotion() << " potions now." << endl;
                    this->numofpotion--;
                    maincharacter->addmoney(-10);
                    continue;
                }
                else {
                    cout << "you don't have enough money!!\n";
                    continue;
                }

            }
            else {
                cout << "no more potion!!!\n";
                continue;
            }

        }
        else if (act == 2) {

            if (this->numofkey > 0) {
                if (maincharacter->getmoney() >= 20) {
                    maincharacter->getBackpack()->setkey(maincharacter->getBackpack()->getkey() + 1);
                    cout << "You have " << maincharacter->getBackpack()->getkey() << " keys now." << endl;
                    this->numofpotion--;
                    maincharacter->addmoney(-20);
                    continue;
                }
                else {
                    cout << "you don't have enough money!!\n";
                    continue;
                }
            }
            else {
                cout << "no more key!!!\n";
                continue;
            }
        }
        else {
            if (maincharacter->getmoney() >= this->getCommodity()[act - 3]->getcost()) {
                this->getCommodity()[act - 3]->triggerBackpack(maincharacter);
                int a = act - 3;                                    Player* NPC::triggerEvent::maincharacter
                this->commodity.erase(commodity.begin() + a);
                continue;

            }
            else {
                cout << "you don't have enough money!!\n";
                continue;
            }
        }
    }
```

用 while 迴圈不斷販售商品，直到玩家想離開(input 0)。

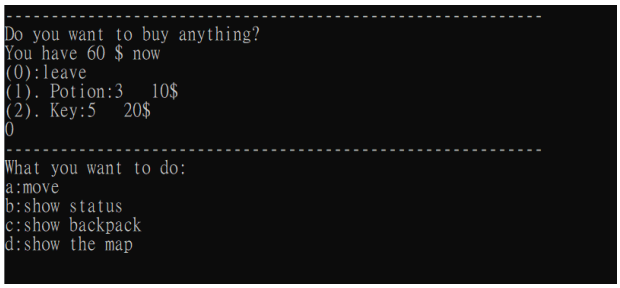
會先列印出所有商品，並讓玩家輸入想要的商品的號碼，判別玩家的錢是足夠的後，將物品放入玩家的背包裡。




可以買裝備

買鑰匙

買藥水



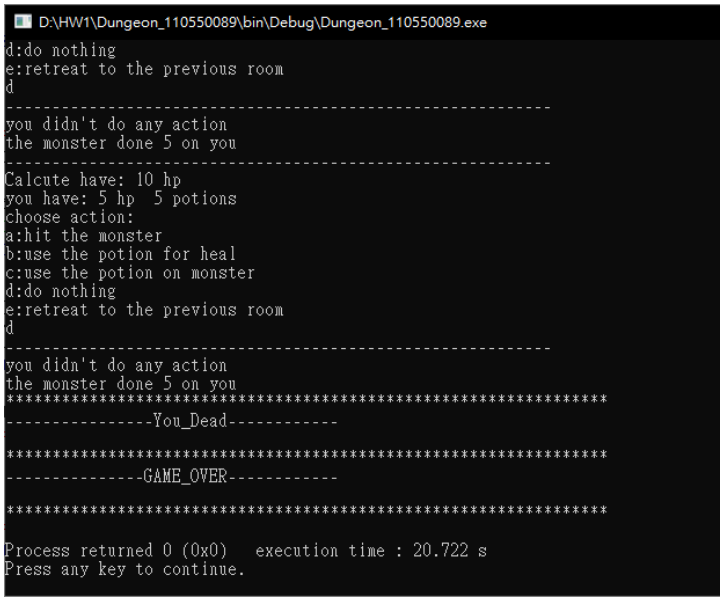

輸入為 0 時離開商人系統
繼續選擇行動

6. GameLogic

在 Dungeon 內的 checkGameLogic()會在做完選擇後，觸發完事件後，去判斷是否結束遊戲，有分勝利跟死亡兩種情況。

```
bool Dungeon::checkGameLogic() {
    if (this->player.getCurrentRoom()->getbossdead()) {
        cout << "*********************************************************************" << endl;
        cout << "Congratualation !! You defeated the final boss and escaped from the dungeon.\n\n";
        cout << "*********************************************************************" << endl;
        cout << "-----------------VICTORY------------\n\n";
        cout << "*********************************************************************" << endl;
        return true;
    }
    else if(player.checkIsDead()){
        cout << "*********************************************************************" << endl;
        cout << "----------------You Dead------------\n\n";
        cout << "*********************************************************************" << endl;
        cout << "---------------GAME OVER------------\n\n";
        cout << "*********************************************************************" << endl;
        return true;
    }
    else {
        return false;
    }
}
```


擊敗最終 BOSS 後，會 return false 結束 triggerEvent()，並開始運行 checkGameLogic()來結束遊戲，判定玩家的勝利。




當玩家死亡時，也會 return false 結束 triggerEvent()，並開始運行 checkGameLogic()來結束遊戲，判定玩家的輸了。

7. Backpack System

```
class Backpack
{
private:
    int numofpotion = 5;
    int numofkey = 3;
    Helmet* wearinghelmet = NULL;
    Armor* wearingarmor = NULL;
    Weapon* wearing_leftweapon = NULL;
    Weapon* wearing_rightweapon = NULL;
    Helmet helmet[100];
    Armor armor[100];
    Weapon weapon[100];
    int helindex = 0;
    int armindex = 0;
    int weaindex = 0;
public:
    Backpack();
    void showbackpack();
    void changeequip(Player*);
```

Backpack 裡面可以存放各種裝備，已有變數是存取穿著中的裝備。

我將藥水跟鑰匙這種沒有各自特別性質的物品以 int 的型態去存取它們的個數。

我將秀裝備及換裝備的系統整合在一起。

showbackpack()就是單純的將資料印出。

changeequip(Player*) 則是負責整個背包系統運行的函式。

```
void Backpack::changeequip(Player* player){
    Player* maincharacter = dynamic_cast<Player*>(player);
    while (1) {
        this->showbackpack();
        cout << "-------------------------------------------------------------" << endl;
        cout << "(a)change helmet\n" << "(b)change armor\n" << "(c)change lefthand weapon\n" << "(d)change righthand weapon\n" << "(e)closed backpack\n";
        char in;
        cin >> in;
        if (in == 'a' || in == 'A') {
            cout << "Wearing Helmet: "<< maincharacter->getBackpack()->getwhelmet()->getName()<<endl;
            for (int i = 0; i < helindex; i++) {
                if (this->helmet[i].getwear()) {
                    cout <<"(" <<  i << ")" << ". " << this->helmet[i].getName() << " +ATK" << this->helmet[i].getAttack() << " +DEF" << this->helmet[i].get
                }
                else {
                    cout <<"(" <<  i << ")" << ". " << this->helmet[i].getName() << " +ATK" << this->helmet[i].getAttack() << " +DEF" << this->helmet[i].get
                }
            }
            cout << "input the index of helmet:\n";
            int index;
            cin >> index;
            if(index<0||index>=helindex){
                cout<<"wrong input!!!\n";
                continue;
            }
            maincharacter->decreaseStates(wearinghelmet);
            setwhelmet(index);
            maincharacter->increaseStates(wearinghelmet);
            continue;
        }

        else if (in == 'b' || in == 'B') {
            cout << "Wearing Armor: "<< maincharacter->getBackpack()->getwarmor()->getName()<<endl;
            for (int i = 0; i < armindex; i++) {
                if (this->armor[i].getwear()) {
                    cout <<"(" <<  i << ")" << ". " << this->armor[i].getName() << " +ATK" << this->armor[i].getAttack() << " +DEF" << this->armor[i].getDef
                }
                else {
                    cout <<"(" <<  i << ")" << ". " << this->armor[i].getName() << " +ATK" << this->armor[i].getAttack() << " +DEF" << this->armor[i].getDef
                }
            }
            cout << "input the index of armor:\n";
            int index;
            cin >> index;
            if(index < 0 ||index >= armindex){
                cout<<"wrong input!!!\n";
                continue;
            }
            maincharacter->decreaseStates(wearingarmor);
            setwarmor(index);
            maincharacter->increaseStates(wearingarmor);
            continue;
        }
```

```
else if (in == 'c' || in == 'C') {
    cout << "Wearing LeftWeapon: "<< maincharacter->getBackpack()->getwleftweapon()->getName()<<endl;
    for (int i = 0; i < weaindex; i++) {
        if (this->weapon[i].getwear()) {
            cout <<"(" <<  i << ")" << ". " << this->weapon[i].getName() << " +ATK" << this->weapon[i].getAttack() << " +DEF" << this->weapon[i].get
        }
        else {
            cout <<"(" <<  i << ")" << ". " << this->weapon[i].getName() << " +ATK" << this->weapon[i].getAttack() << " +DEF" << this->weapon[i].get
        }
    }
    cout << "input the index of weapon:\n";
    int index;
    cin >> index;
    if(index < 0 ||index >= weaindex){
        cout<<"wrong input!!!\n";
        continue;
    }
    maincharacter->decreaseStates(wearing_leftweapon);
    setwleftweapon(index);
    maincharacter->increaseStates(wearing_leftweapon);
    continue;
}
```

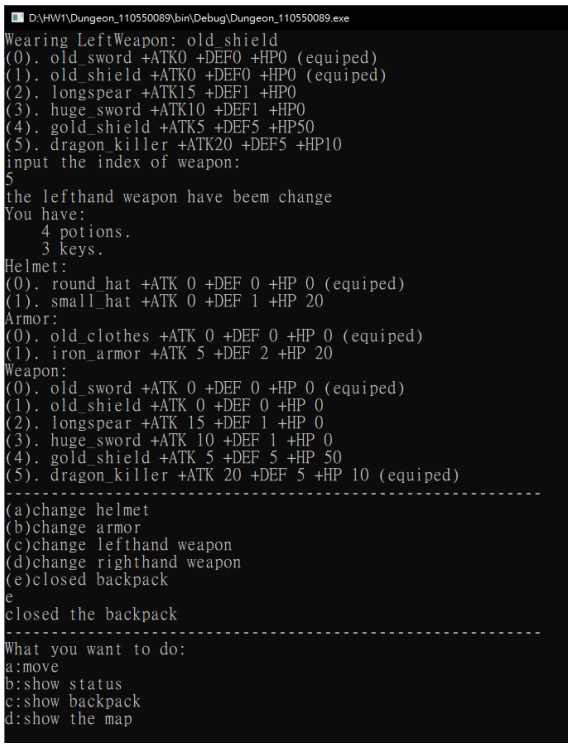
```
    else if (in == 'd' || in == 'D') {
        cout << "Wearing RightWeapon: "<< maincharacter->getBackpack()->getwrightweapon()->getName()<<endl;
        for (int i = 0; i < weaindex; i++) {
            if (this->weapon[i].getwear()) {
                cout << i << ". " << this->weapon[i].getName() << " +ATK" << this->weapon[i].getAttack() << " +DEF" << this->weapon[i].getDefense() << '
            }
            else {
                cout << i << ". " << this->weapon[i].getName() << " +ATK" << this->weapon[i].getAttack() << " +DEF" << this->weapon[i].getDefense() << '
            }
        }
        cout << "input the index of weapon:\n";
        int index;
        cin >> index;
        if(index < 0 ||index >= weaindex){
            cout<<"wrong input!!!\n";
            continue;
        }
        maincharacter->decreaseStates(wearing_rightweapon);
        setwrightweapon(index);
        maincharacter->increaseStates(wearing_rightweapon);
        continue;
    }
    else if (in == 'e' || in == 'E') {
        cout << "closed the backpack\n";
        break;
    }
    else {
        cout << "wrong input\n";
        continue;
    }
}
}
```

先將背包內所有物品依序印出來，然後讓玩家選擇要更換的裝備，再讓玩家選擇要換的裝備的編號，進行交換。

```
------------------------------------------------------------
What you want to do:
a:move
b:show status
c:show backpack
d:show the map
c
------------------------------------------------------------
You have:
    4 potions.
    3 keys.
Helmet:
(0). round_hat +ATK 0 +DEF 0 +HP 0 (equiped)
(1). small_hat +ATK 0 +DEF 1 +HP 20
Armor:
(0). old_clothes +ATK 0 +DEF 0 +HP 0 (equiped)
(1). iron_armor +ATK 5 +DEF 2 +HP 20
Weapon:
(0). old_sword +ATK 0 +DEF 0 +HP 0 (equiped)
(1). old_shield +ATK 0 +DEF 0 +HP 0 (equiped)
(2). longspear +ATK 15 +DEF 1 +HP 0
(3). huge_sword +ATK 10 +DEF 1 +HP 0
(4). gold_shield +ATK 5 +DEF 5 +HP 50
(5). dragon_killer +ATK 20 +DEF 5 +HP 10
------------------------------------------------------------
(a)change helmet
(b)change armor
(c)change lefthand weapon
(d)change righthand weapon
(e)closed backpack
c
```

會先印出所有物品

讓玩家選擇要換裝備還是關上背包




選完要換的物品後要選擇物品的編號

輸入(e)可以離開背包系統

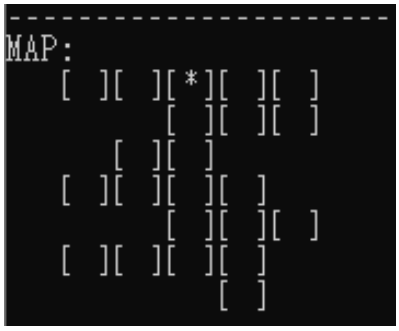
8. createMap

這是一個可以更換地圖並讀取的系統，也就是可以擴充更多的地圖。

利用 fstream 的功能，以及將地圖座標化後，依照讀取到的資料將房間一間間的創造出來。

讀取資料時，首先是讀取房間的個數，接著一間一間房間輸入。

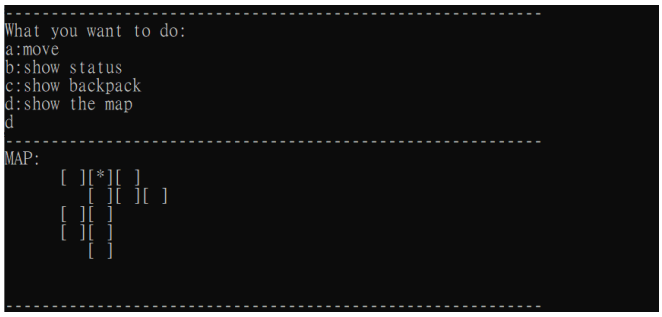
輸入房間時，輸入房間的座標，房間內物品的數量，接著各個物品的型態跟資料。最後在輸入的最後一間房間下面加上 Boss 的房間，一張地圖就完成了。目前我總共設計了三張，但設計的步驟還不太直觀，需再改進。


9. printMap

```
void Dungeon::printMap()
{
    cout << "-------------------------------------------------------" << endl;
    cout << "MAP:\n";
    for (int i = 0; i < 7; i++) {
        for (int j = 0; j < 7; j++) {
            if (mapp[i][j] == NULL) {
                cout << "   ";
            }
            else {
                if (i == player.getx() && j == player.gety()) {
                    cout << "[*]";
                }
                else cout << "[ ]";
            }
        }
        cout << endl;
    }
}
```

藉由判斷每個座標上的房間是否存在來印出地圖，並在讀到玩家所在房間時特別標記。須注意的是 Player 裡面除了存玩家當前的座標外，還需要存前一間房間的座標，不然在撤退時會出錯。



## 三、討論

這次的 Dungeon 的實作我覺得我有不少該改進的地方。

1.缺乏物件導向程式設計的精神

我認為我並沒有將物件導向的想法徹底套用在這個遊戲裡，而只是用以前單純的想法，很普通的把功能給做出來。我在後面詢問朋友後才發現，virtual triggerEvent 在這個程式的用處比我想像中還更廣泛，像是我可以將 Backpack 寫成 Object 的繼承，將 showBackpack 直接用 Backpack 的 triggerEvent 實作。這一次我原本想藉此好好熟悉 vector 的相關用法，但是卻一直出錯，資料會一直不見，最好只還改用傳統 C style 寫法的陣列。

2.可以在擴充的功能與需改進的細節

如果接下來有機會，我應該會嘗試加入職業的系統，不同的職業可以有不同的技能，而這個應該也是可以用 triggerEvent()去完成。再來就是我後來再重新看一次 code 的時候，發現有不少寫的不是很聰明的地方，例如:在一個 class 裡面的 function 本來就可以直接存取到 private 裡面的變數，我卻是用 getter 去取得資料。在一些變數的命名上也應該更謹慎且易於辨認，不然寫到最後連自己都搞不清楚。

3.如何完成一個有一定規模的 project

第一次做 project，我第一步就是給的 template 給都放上去，結果反而搞不清楚該如何著手。應該先將整個整個程式的大綱想清楚，從最基礎的東西一步一步將需要的東西加上去。我認為這是需要經驗去累積的，以後應該多去嘗試寫這樣上百上千行的 project，能增進程式的技巧並對於程式語言有更深一步的理解。

## 四、結論

在剛開學時，教授以組合玩具車來講解物件導向程式設計的概念，當時的我還沒有什麼特別的感覺，在平常寫一些幾十行就可以完成的題目時，也不覺得有什麼特別的地方。但在完成這個 Dungeon 的作業後，我才對於將所有的東西視為物件時，所擁有的特殊性質，有了不同的理解。不用繁瑣的打類似的功能，且能在以經建立的基礎上更容易的新增加功能，且整個程式也相較容易去維護。

期許自己能以這次經驗為基石，未來在嘗試開發自己的軟體或是學習更進階的技巧時，能將物件導向程式設計的能力派上用場。