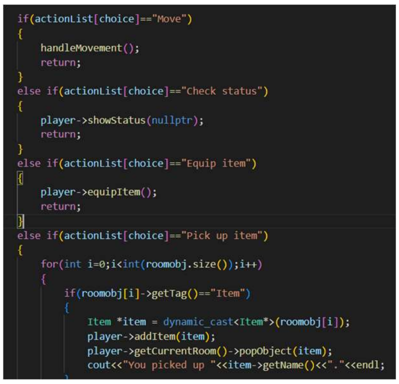
Implementing detail

Action Menu: (Dungeon.cpp 302~424lines)



```
if(actionList[choice]=="Move")
{
    handleMovement();
    return;
}
else if(actionList[choice]=="Check status")
{
    player->showStatus(nullptr);
    return;
}
else if(actionList[choice]=="Equip item")
{
    player->equipItem();
    return;
}
else if(actionList[choice]=="Pick up item")
{
    for(int i=0;i<int(roomobj.size());i++)
    {
        if(roomobj[i]->getTag()=="Item")
        {
            Item *item = dynamic_cast<Item*>(roomobj[i]);
            player->addItem(item);
            player->getCurrentRoom()->popObject(item);
            cout<<"You picked up "<<item->getName()<<"."<<endl;
```

(Dungeon.cpp 302~424lines)

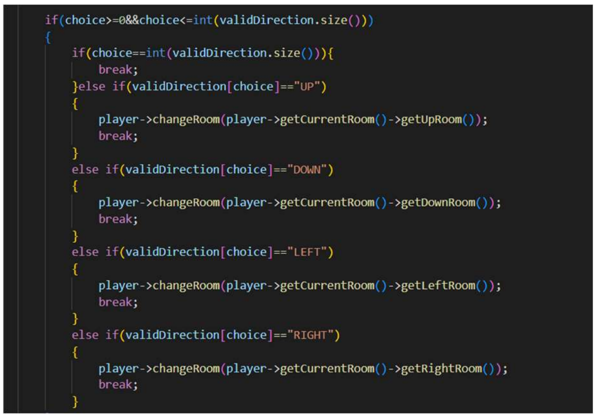
Move, Check status, Equip item, Pick Up Item,

Door to next stage, Fight with boss/monster, Talk with NPC

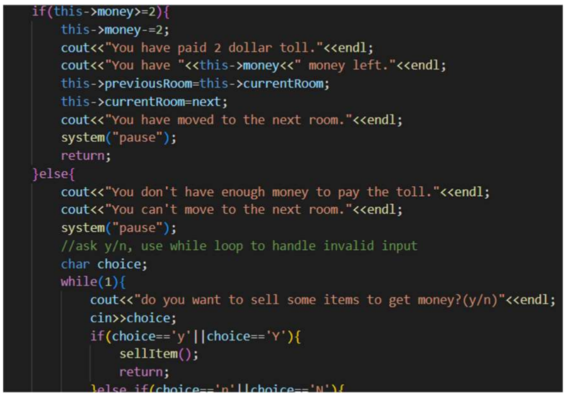
Movement(with Toll)

(Dungeon.cpp 249~300、player.cpp 32~61)

Go UP, Go DOWN, Go LEFT, Go RIGHT



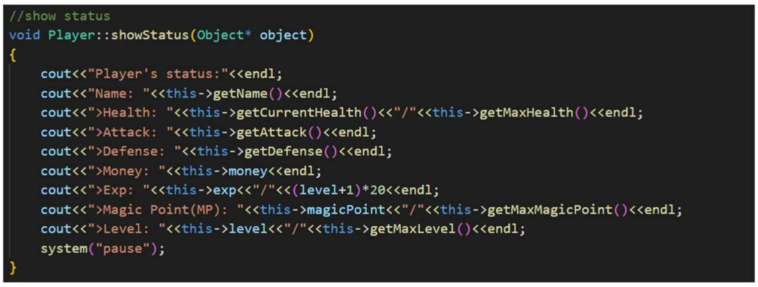
```
if(choice>=0&&choice<=int(validDirection.size()))
{
    if(choice==int(validDirection.size())){
        break;
    }else if(validDirection[choice]=="UP")
    {
        player->changeRoom(player->getCurrentRoom()->getUpRoom());
        break;
    }
    else if(validDirection[choice]=="DOWN")
    {
        player->changeRoom(player->getCurrentRoom()->getDownRoom());
        break;
    }
    else if(validDirection[choice]=="LEFT")
    {
        player->changeRoom(player->getCurrentRoom()->getLeftRoom());
        break;
    }
    else if(validDirection[choice]=="RIGHT")
    {
        player->changeRoom(player->getCurrentRoom()->getRightRoom());
        break;
    }
```



```
if(this->money>=2){
    this->money-=2;
    cout<<"You have paid 2 dollar toll."<<endl;
    cout<<"You have "<<this->money<<" money left."<<endl;
    this->previousRoom=this->currentRoom;
    this->currentRoom=next;
    cout<<"You have moved to the next room."<<endl;
    system("pause");
    return;
}else{
    cout<<"You don't have enough money to pay the toll."<<endl;
    cout<<"You can't move to the next room."<<endl;
    system("pause");
    //ask y/n, use while loop to handle invalid input
    char choice;
    while(1){
        cout<<"do you want to sell some items to get money?(y/n)"<<endl;
        cin>>choice;
        if(choice=='y'||choice=='Y'){
            sellItem();
            return;
        }else if(choice=='n'||choice=='N'){
```

Showing Status

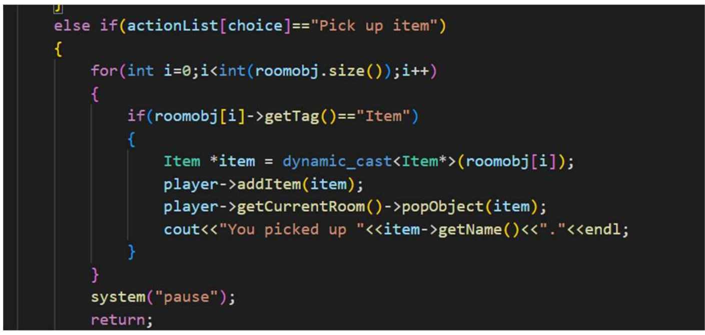
(player.cpp 85~97 lines)



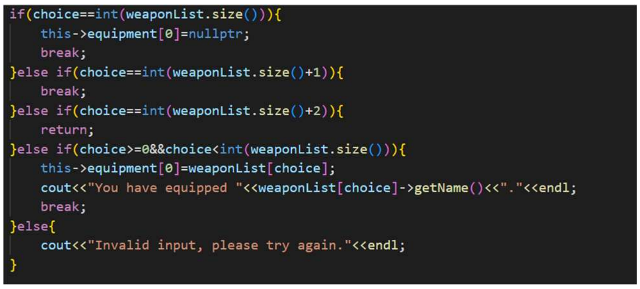
```
//show status
void Player::showStatus(Object* object)
{
    cout<<"Player's status:"<<endl;
    cout<<"Name: "<<this->getName()<<endl;
    cout<<">Health: "<<this->getCurrentHealth()<<"/"<<this->getMaxHealth()<<endl;
    cout<<">Attack: "<<this->getAttack()<<endl;
    cout<<">Defense: "<<this->getDefense()<<endl;
    cout<<">Money: "<<this->money<<endl;
    cout<<">Exp: "<<this->exp<<"/"<<(level+1)*20<<endl;
    cout<<">Magic Point(MP): "<<this->magicPoint<<"/"<<this->getMaxMagicPoint()<<endl;
    cout<<">Level: "<<this->level<<"/"<<this->getMaxLevel()<<endl;
    system("pause");
}
```

Pick up Item(dungeon.cpp 375~389 lines)

And equipment them (player.cpp 200~356lines)



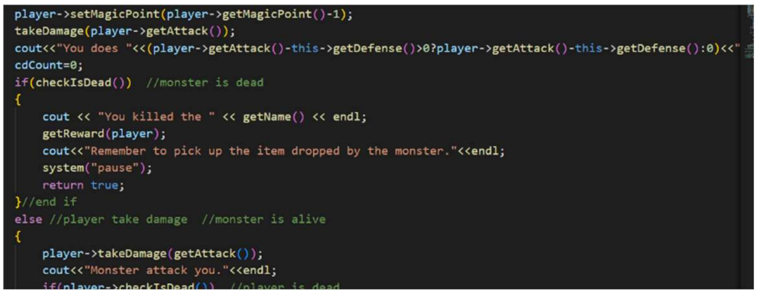
```
}
else if(actionList[choice]=="Pick up item")
{
    for(int i=0;i<int(roomobj.size());i++)
    {
        if(roomobj[i]->getTag()=="Item")
        {
            Item *item = dynamic_cast<Item*>(roomobj[i]);
            player->addItem(item);
            player->getCurrentRoom()->popObject(item);
            cout<<"You picked up "<<item->getName()<<"."<<endl;
        }
    }
    system("pause");
    return;
```



```
if(choice==int(weaponList.size())){
    this->equipment[0]=nullptr;
    break;
}else if(choice==int(weaponList.size()+1)){
    break;
}else if(choice==int(weaponList.size()+2)){
    return;
}else if(choice>=0&&choice<int(weaponList.size())){
    this->equipment[0]=weaponList[choice];
    cout<<"You have equipped "<<weaponList[choice]->getName()<<"."<<endl;
    break;
}else{
    cout<<"Invalid input, please try again."<<endl;
}
```

Fighting System

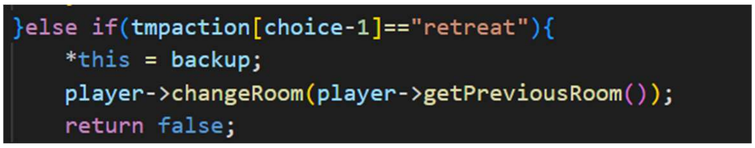
Attack(Monster.cpp 23~178 lines)



```
player->setMagicPoint(player->getMagicPoint()-1);
takeDamage(player->getAttack());
cout<<"You does "<<(player->getAttack()-this->getDefense()>0?player->getAttack()-this->getDefense():0)<<"
cdCount=0;
if(checkIsDead())  //monster is dead
{
    cout << "You killed the " << getName() << endl;
    getReward(player);
    cout<<"Remember to pick up the item dropped by the monster."<<endl;
    system("pause");
    return true;
}//end if
else //player take damage  //monster is alive
{
    player->takeDamage(getAttack());
    cout<<"Monster attack you."<<endl;
    if(player->checkIsDead())  //player is dead
```

Retreat(cost 2 dollars(since toll))

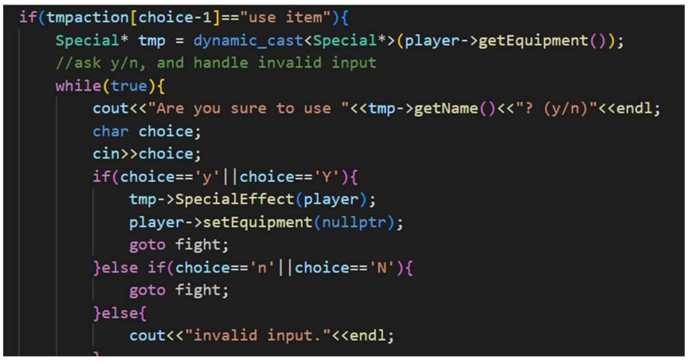
(Monster.cpp 166~169lines)



```
}else if(tmpaction[choice-1]=="retreat"){
    *this = backup;
    player->changeRoom(player->getPreviousRoom());
    return false;
```

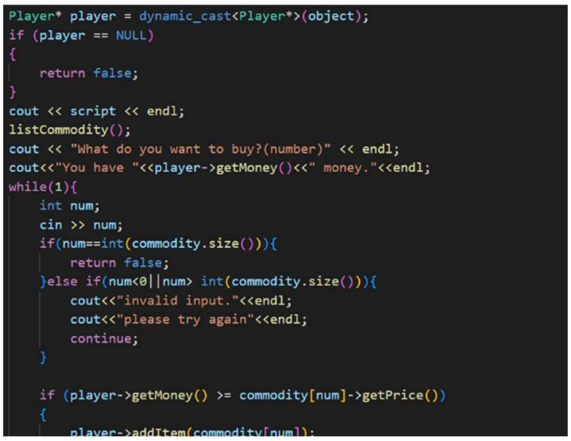
Use special item(if is equip on)

(Monster.cpp 149~165lines)



```
if(tmpaction[choice-1]=="use item"){
    Special* tmp = dynamic_cast<Special*>(player->getEquipment());
    //ask y/n, and handle invalid input
    while(true){
        cout<<"Are you sure to use "<<tmp->getName()<<"? (y/n)"<<endl;
        char choice;
        cin>>choice;
        if(choice=='y'||choice=='Y'){
            tmp->SpecialEffect(player);
            player->setEquipment(nullptr);
            goto fight;
        }else if(choice=='n'||choice=='N'){
            goto fight;
        }else{
            cout<<"invalid input."<<endl;
```

NPC &Trading system(NPC.cpp 45~99lines)



```
Player* player = dynamic_cast<Player*>(object);
if (player == NULL)
{
    return false;
}
cout << script << endl;
listCommodity();
cout << "What do you want to buy?(number)" << endl;
cout<<"You have "<<player->getMoney()<<" money."<<endl;
while(1){
    int num;
    cin >> num;
    if(num==int(commodity.size())){
        return false;
    }else if(num<0||num> int(commodity.size())){
        cout<<"invalid input."<<endl;
        cout<<"please try again"<<endl;
        continue;
    }

    if (player->getMoney() >= commodity[num]->getPrice())
    {
        player->addItem(commodity[num]);
```

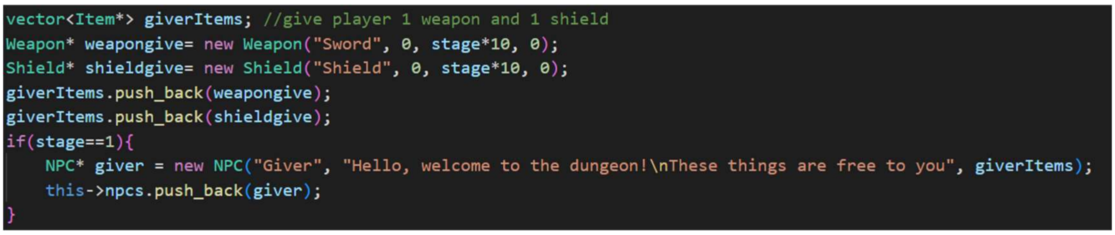
Different script between 1 and other stage

Trading System(seller_NPC)

Sell something, only exist in Room 5 of every stage

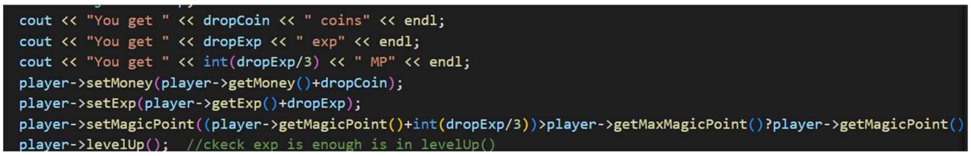
Chest(Giver_NPC)(Dungeo.cpp 91~122lines)

Free store, all items are free, only exist in the first room of every stage



```
vector<Item*> giverItems; //give player 1 weapon and 1 shield
Weapon* weapongive= new Weapon("Sword", 0, stage*10, 0);
Shield* shieldgive= new Shield("Shield", 0, stage*10, 0);
giverItems.push_back(weapongive);
giverItems.push_back(shieldgive);
if(stage==1){
    NPC* giver = new NPC("Giver", "Hello, welcome to the dungeon!\nThese things are free to you", giverItems);
    this->npcs.push_back(giver);
}
```

Money

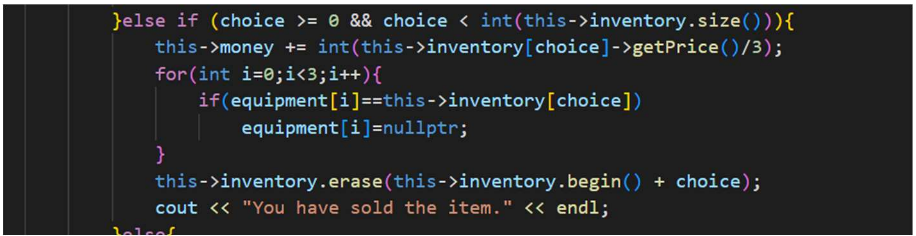


```
cout << "You get " << dropCoin << " coins" << endl;
cout << "You get " << dropExp << " exp" << endl;
cout << "You get " << int(dropExp/3) << " MP" << endl;
player->setMoney(player->getMoney()+dropCoin);
player->setExp(player->getExp()+dropExp);
player->setMagicPoint((player->getMagicPoint()+int(dropExp/3))>player->getMaxMagicPoint()?player->getMagicPoint()
player->levelUp();  //ckeck exp is enough is in levelUp()
```

Earn by:

Kill monster(Monster.cpp 180~207)/boss(Monster.cpp 234~275lines)

Sell item (Player.cpp 100~130lines)



```
}else if (choice >= 0 && choice < int(this->inventory.size())){
    this->money += int(this->inventory[choice]->getPrice()/3);
    for(int i=0;i<3;i++){
        if(equipment[i]==this->inventory[choice])
            equipment[i]=nullptr;
    }
    this->inventory.erase(this->inventory.begin() + choice);
    cout << "You have sold the item." << endl;
}else{
```

Spend by:

Buy item from seller.(NPC.cpp 49~99lines)

Change room(Tolls) (Player.cpp 32~61lines)

Exchange to MP(Player.cpp 132~166line) #when in fighting but don't have MP

Game Logic(Dungeon.cpp 426~474lines)

Win:

Player level >5 (Dungeon.cpp 445~453lines)

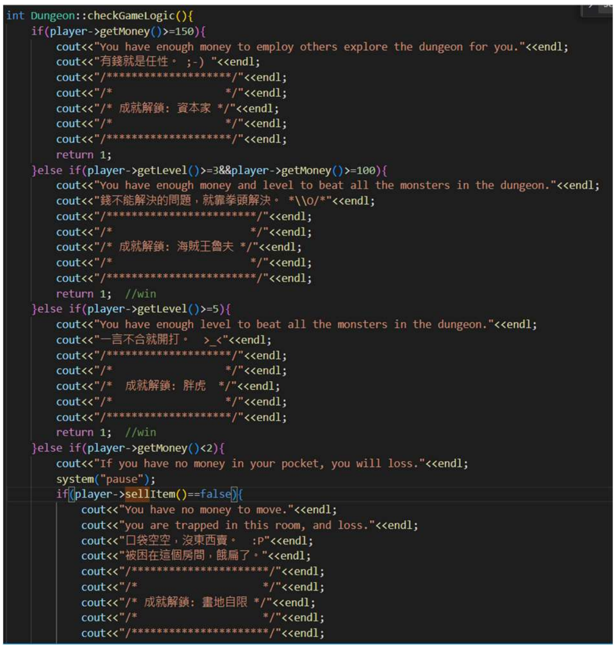
Player have more than 150 dollars(Dungeon.cpp 427~435 lines)

Player level>3 && monry>=100(Dungeon.cpp 436~444lines)

Loss:

Die(handle in fighting system) (Monster.cpp 68~74lines)

No money(and don't have anything valuable to sell) (Dungeon.cpp 454~470lines)



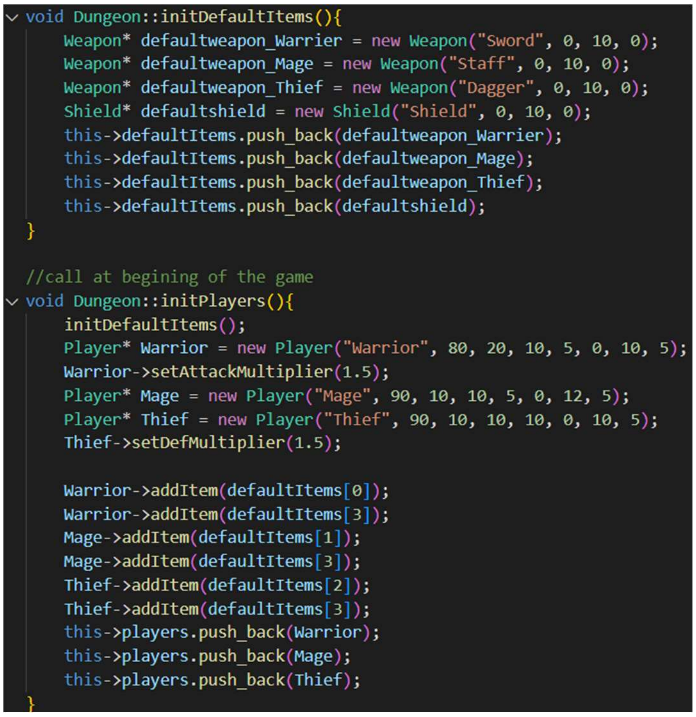
```
int Dungeon::checkGameLogic(){
    if(player->getMoney()>=150){
        cout<<"You have enough money to employ others explore the dungeon for you."<<endl;
        cout<<"有錢就是任性。 ;-) "<<endl;
        cout<<"/*******************/"<<endl;
        cout<<"/*                 */"<<endl;
        cout<<"/* 成就解鎖: 資本家 */"<<endl;
        cout<<"/*                 */"<<endl;
        cout<<"/*******************/"<<endl;
        return 1;
    }else if(player->getLevel()>=3&&player->getMoney()>=100){
        cout<<"You have enough money and level to beat all the monsters in the dungeon."<<endl;
        cout<<"錢不能解決的問題，就靠拳頭解決。 *\\o/*"<<endl;
        cout<<"/***********************/"<<endl;
        cout<<"/*                     */"<<endl;
        cout<<"/* 成就解鎖: 海賊王魯夫 */"<<endl;
        cout<<"/*                     */"<<endl;
        cout<<"/***********************/"<<endl;
        return 1;  //win
    }else if(player->getLevel()>=5){
        cout<<"You have enough level to beat all the monsters in the dungeon."<<endl;
        cout<<"一言不合就開打。  >_<"<<endl;
        cout<<"/*******************/"<<endl;
        cout<<"/*                 */"<<endl;
        cout<<"/*  成就解鎖: 胖虎  */"<<endl;
        cout<<"/*                 */"<<endl;
        cout<<"/*******************/"<<endl;
        return 1;  //win
    }else if(player->getMoney()<2){
        cout<<"If you have no money in your pocket, you will loss."<<endl;
        system("pause");
        if(player->sellItem()==false){
            cout<<"You have no money to move."<<endl;
            cout<<"you are trapped in this room, and loss."<<endl;
            cout<<"口袋空空，沒東西賣。  :P"<<endl;
            cout<<"被困在這個房間，餓扁了。"<<endl;
            cout<<"/*********************/"<<endl;
            cout<<"/*                   */"<<endl;
            cout<<"/* 成就解鎖: 畫地自限 */"<<endl;
            cout<<"/*                   */"<<endl;
            cout<<"/*********************/"<<endl;
```

Character Class Design

Occupation(職業): Warrior, Mage, Thief(Dungeon.cpp 71~89lines)

Skills(攻擊倍率、防禦倍率)(Dungeon.cpp 74、76lines)

Corresponding weapon (Dungeon.cpp 59~68lines)



```cpp
void Dungeon::initDefaultItems(){
    Weapon* defaultweapon_Warrier = new Weapon("Sword", 0, 10, 0);
    Weapon* defaultweapon_Mage = new Weapon("Staff", 0, 10, 0);
    Weapon* defaultweapon_Thief = new Weapon("Dagger", 0, 10, 0);
    Shield* defaultshield = new Shield("Shield", 0, 10, 0);
    this->defaultItems.push_back(defaultweapon_Warrier);
    this->defaultItems.push_back(defaultweapon_Mage);
    this->defaultItems.push_back(defaultweapon_Thief);
    this->defaultItems.push_back(defaultshield);
}

//call at begining of the game
void Dungeon::initPlayers(){
    initDefaultItems();
    Player* Warrior = new Player("Warrior", 80, 20, 10, 5, 0, 10, 5);
    Warrior->setAttackMultiplier(1.5);
    Player* Mage = new Player("Mage", 90, 10, 10, 5, 0, 12, 5);
    Player* Thief = new Player("Thief", 90, 10, 10, 10, 0, 10, 5);
    Thief->setDefMultiplier(1.5);

    Warrior->addItem(defaultItems[0]);
    Warrior->addItem(defaultItems[3]);
    Mage->addItem(defaultItems[1]);
    Mage->addItem(defaultItems[3]);
    Thief->addItem(defaultItems[2]);
    Thief->addItem(defaultItems[3]);
    this->players.push_back(Warrior);
    this->players.push_back(Mage);
    this->players.push_back(Thief);
}
```

Optional Enhancement

Player MP(fighting need mp) (Monster.cpp 40~53lines)

Monster CD (Monster.cpp 33、87 lines)

Move needs Toll (Player.cpp 33~41lines)

Sell Item to get money (Player.cpp 100~130lines)

Special Item (Monster.cpp 225~231lines)

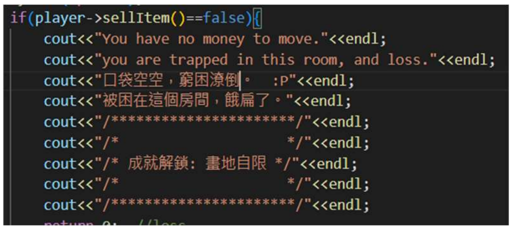
Use Special Item(TmpDefence) (Item.cpp 43~93lines)

Have 5 stage(Monster.cpp 501~519lines)

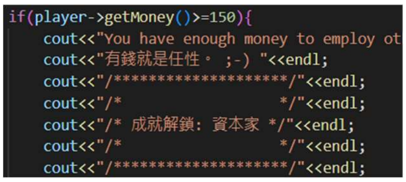
Player level (Player.cpp 64~89lines)

Exchange MP(from money) (Player.cpp 132~166lines)

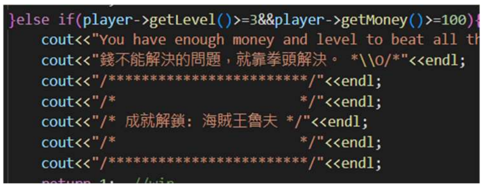
Different end point(achievement system) (Dungeon.cpp 426~479lines)



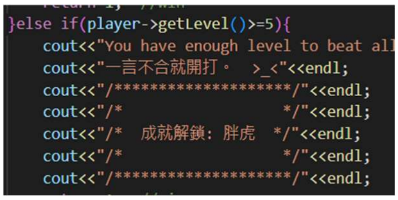
```cpp
if(player->sellItem()==false){
    cout<<"You have no money to move."<<endl;
    cout<<"you are trapped in this room, and loss."<<endl;
    cout<<"口袋空空，窮困潦倒。   :P"<<endl;
    cout<<"被困在這個房間，餓扁了。"<<endl;
    cout<<"/**********************/"<<endl;
    cout<<"/*                    */"<<endl;
    cout<<"/* 成就解鎖: 畫地自限 */"<<endl;
    cout<<"/*                    */"<<endl;
    cout<<"/**********************/"<<endl;
    return 0;   //loss
```



```cpp
if(player->getMoney()>=150){
    cout<<"You have enough money to employ ot
    cout<<"有錢就是任性。 ;-) "<<endl;
    cout<<"/********************/"<<endl;
    cout<<"/*                  */"<<endl;
    cout<<"/* 成就解鎖: 資本家 */"<<endl;
    cout<<"/*                  */"<<endl;
    cout<<"/********************/"<<endl;
```

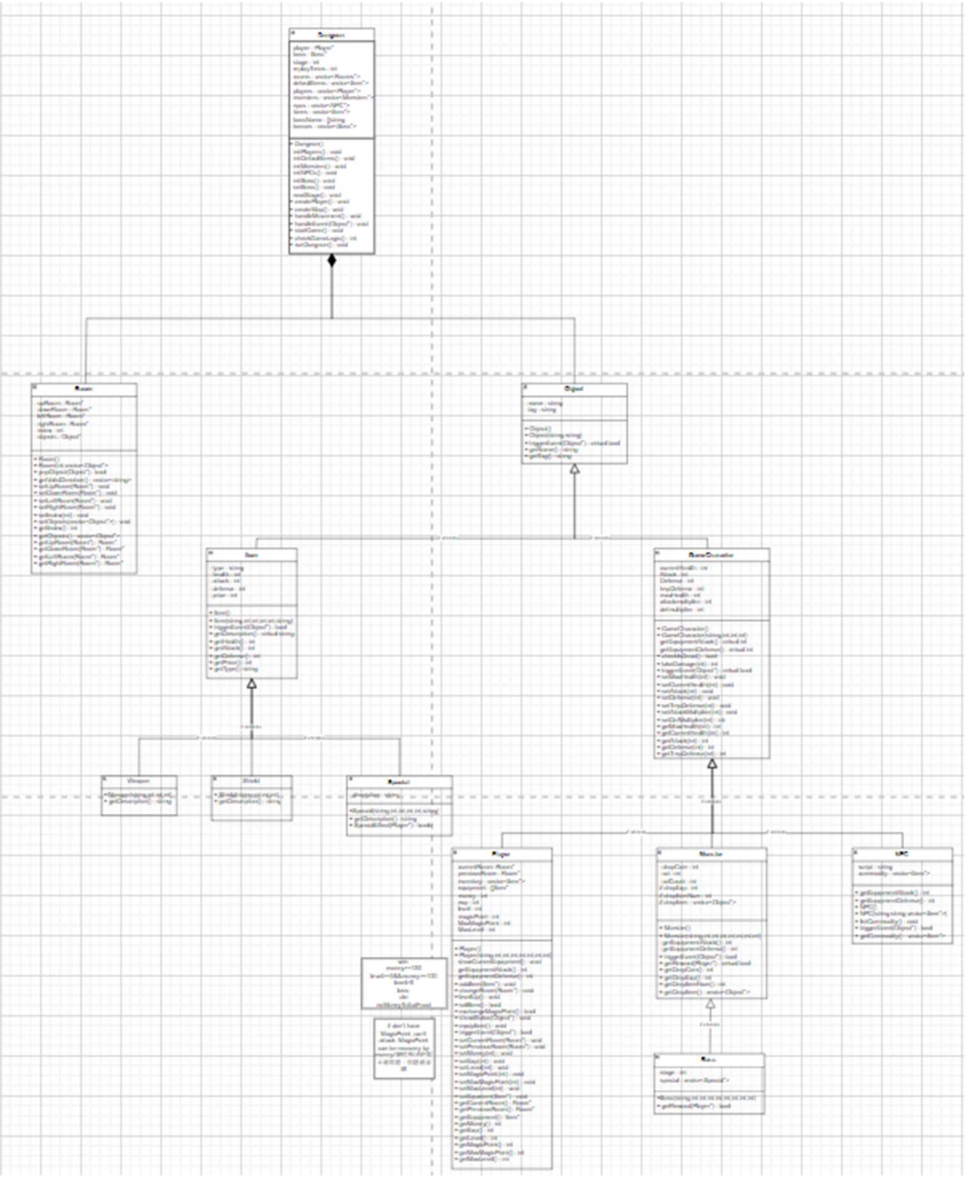


```cpp
}else if(player->getLevel()>=3&&player->getMoney()>=100){
    cout<<"You have enough money and level to beat all th
    cout<<"錢不能解決的問題，就靠拳頭解決。 *\\O/*"<<endl;
    cout<<"/************************/"<<endl;
    cout<<"/*                      */"<<endl;
    cout<<"/* 成就解鎖: 海賊王魯夫 */"<<endl;
    cout<<"/*                      */"<<endl;
    cout<<"/************************/"<<endl;
    return 1;   //win
```

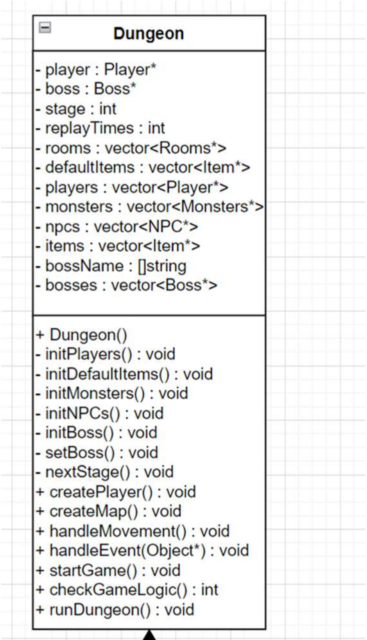


```cpp
    return 1;   //win
}else if(player->getLevel()>=5){
    cout<<"You have enough level to beat all
    cout<<"一言不合就開打。  >_<"<<endl;
    cout<<"/********************/"<<endl;
    cout<<"/*                  */"<<endl;
    cout<<"/*  成就解鎖: 胖虎  */"<<endl;
    cout<<"/*                  */"<<endl;
    cout<<"/********************/"<<endl;
```

UML design (link: DungeonUML_111550046.drawio)



Dungeon:

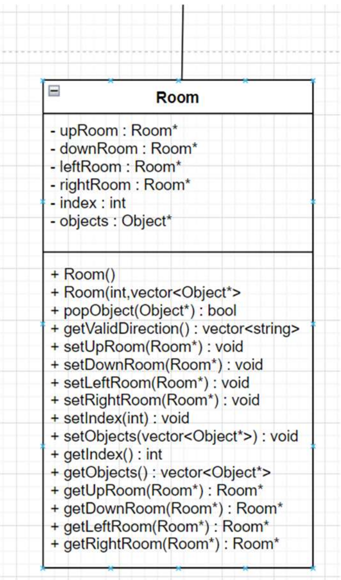


Dungeon

- player : Player*
- boss : Boss*
- stage : int
- replayTimes : int
- rooms : vector<Rooms*>
- defaultItems : vector<Item*>
- players : vector<Player*>
- monsters : vector<Monsters*>
- npcs : vector<NPC*>
- items : vector<Item*>
- bossName : []string
- bosses : vector<Boss*>

```
+ Dungeon()
- initPlayers() : void
- initDefaultItems() : void
- initMonsters() : void
- initNPCs() : void
- initBoss() : void
- setBoss() : void
- nextStage() : void
+ createPlayer() : void
+ createMap() : void
+ handleMovement() : void
+ handleEvent(Object*) : void
+ startGame() : void
+ checkGameLogic() : int
+ runDungeon() : void
```

Room:

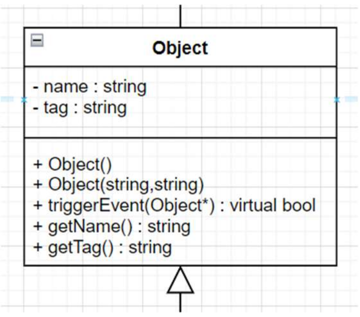


Room

```
- upRoom : Room*
- downRoom : Room*
- leftRoom : Room*
- rightRoom : Room*
- index : int
- objects : Object*
```

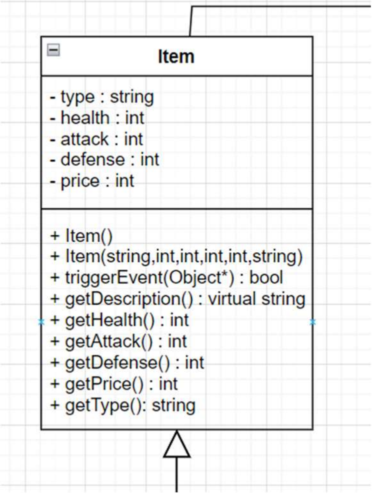
```
+ Room()
+ Room(int,vector<Object*>
+ popObject(Object*) : bool
+ getValidDirection() : vector<string>
+ setUpRoom(Room*) : void
+ setDownRoom(Room*) : void
+ setLeftRoom(Room*) : void
+ setRightRoom(Room*) : void
+ setIndex(int) : void
+ setObjects(vector<Object*>) : void
+ getIndex() : int
+ getObjects() : vector<Object*>
+ getUpRoom(Room*) : Room*
+ getDownRoom(Room*) : Room*
+ getLeftRoom(Room*) : Room*
+ getRightRoom(Room*) : Room*
```

Object:



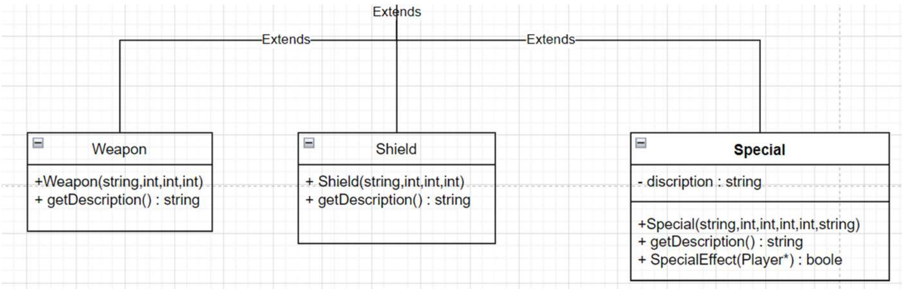
```
Object
- name : string
- tag : string
+ Object()
+ Object(string,string)
+ triggerEvent(Object*) : virtual bool
+ getName() : string
+ getTag() : string
```

Item:



```
Item
- type : string
- health : int
- attack : int
- defense : int
- price : int
+ Item()
+ Item(string,int,int,int,int,string)
+ triggerEvent(Object*) : bool
+ getDescription() : virtual string
+ getHealth() : int
+ getAttack() : int
+ getDefense() : int
+ getPrice() : int
+ getType(): string
```

Weapon & Shield & Special:

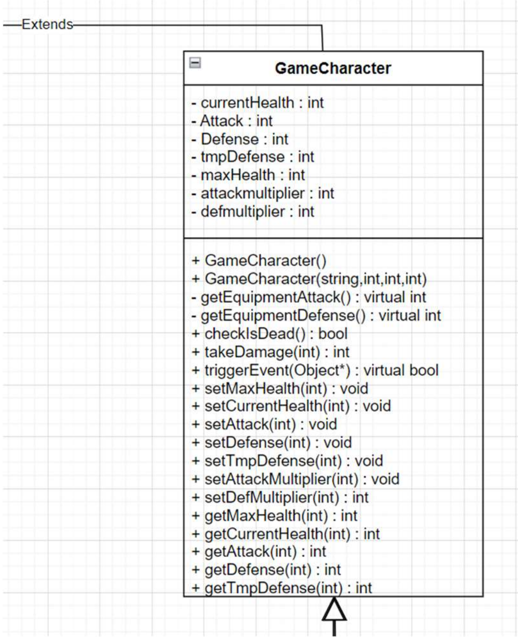


```
Weapon
+Weapon(string,int,int,int)
+ getDescription() : string
```

```
Shield
+ Shield(string,int,int,int)
+ getDescription() : string
```

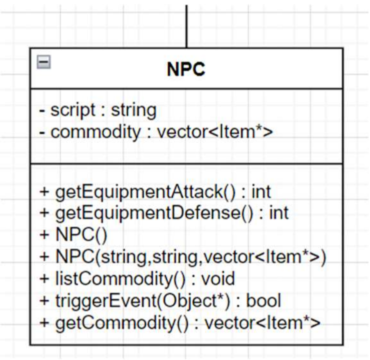
```
Special
- discription : string
+Special(string,int,int,int,int,string)
+ getDescription() : string
+ SpecialEffect(Player*) : boole
```

GameCharacter:



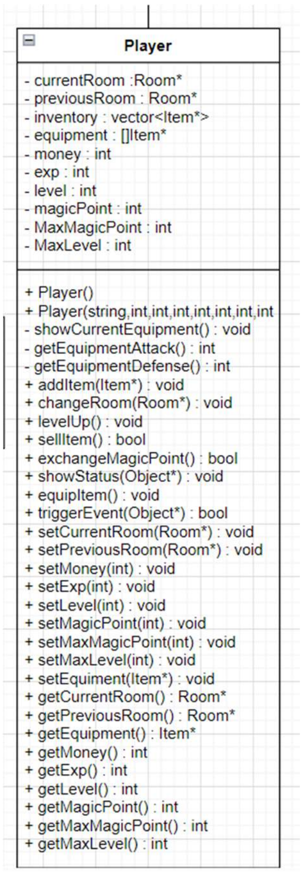
```
GameCharacter
- currentHealth : int
- Attack : int
- Defense : int
- tmpDefense : int
- maxHealth : int
- attackmultiplier : int
- defmultiplier : int
+ GameCharacter()
+ GameCharacter(string,int,int,int)
- getEquipmentAttack() : virtual int
- getEquipmentDefense() : virtual int
+ checkIsDead() : bool
+ takeDamage(int) : int
+ triggerEvent(Object*) : virtual bool
+ setMaxHealth(int) : void
+ setCurrentHealth(int) : void
+ setAttack(int) : void
+ setDefense(int) : void
+ setTmpDefense(int) : void
+ setAttackMultiplier(int) : void
+ setDefMultiplier(int) : int
+ getMaxHealth(int) : int
+ getCurrentHealth(int) : int
+ getAttack(int) : int
+ getDefense(int) : int
+ getTmpDefense(int) : int
```

NPC:



```
NPC
- script : string
- commodity : vector<Item*>
+ getEquipmentAttack() : int
+ getEquipmentDefense() : int
+ NPC()
+ NPC(string,string,vector<Item*>)
+ listCommodity() : void
+ triggerEvent(Object*) : bool
+ getCommodity() : vector<Item*>
```

Player:

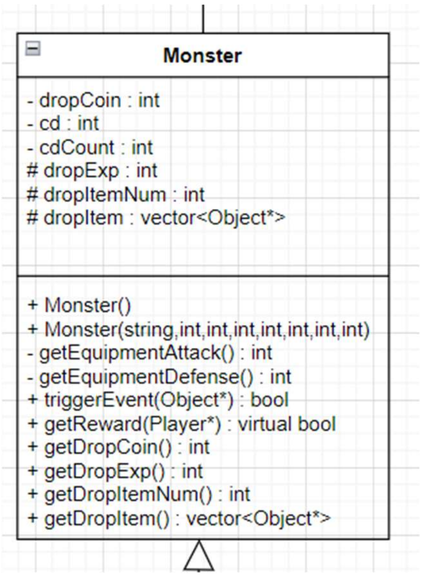


```
Player

- currentRoom :Room*
- previousRoom : Room*
- inventory : vector<Item*>
- equipment : []Item*
- money : int
- exp : int
- level : int
- magicPoint : int
- MaxMagicPoint : int
- MaxLevel : int

+ Player()
+ Player(string,int,int,int,int,int,int,int
- showCurrentEquipment() : void
- getEquipmentAttack() : int
- getEquipmentDefense() : int
+ addItem(Item*) : void
+ changeRoom(Room*) : void
+ levelUp() : void
+ sellItem() : bool
+ exchangeMagicPoint() : bool
+ showStatus(Object*) : void
+ equipItem() : void
+ triggerEvent(Object*) : bool
+ setCurrentRoom(Room*) : void
+ setPreviousRoom(Room*) : void
+ setMoney(int) : void
+ setExp(int) : void
+ setLevel(int) : void
+ setMagicPoint(int) : void
+ setMaxMagicPoint(int) : void
+ setMaxLevel(int) : void
+ setEquiment(Item*) : void
+ getCurrentRoom() : Room*
+ getPreviousRoom() : Room*
+ getEquipment() : Item*
+ getMoney() : int
+ getExp() : int
+ getLevel() : int
+ getMagicPoint() : int
+ getMaxMagicPoint() : int
+ getMaxLevel() : int
```

Monster:

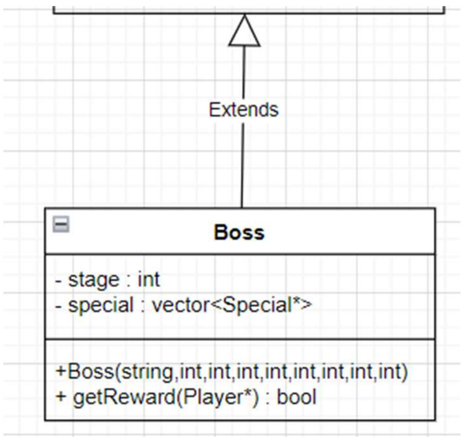


```
Monster

- dropCoin : int
- cd : int
- cdCount : int
# dropExp : int
# dropItemNum : int
# dropItem : vector<Object*>

+ Monster()
+ Monster(string,int,int,int,int,int,int,int)
- getEquipmentAttack() : int
- getEquipmentDefense() : int
+ triggerEvent(Object*) : bool
+ getReward(Player*) : virtual bool
+ getDropCoin() : int
+ getDropExp() : int
+ getDropItemNum() : int
+ getDropItem() : vector<Object*>
```

Boss:



```
Boss

- stage : int
- special : vector<Special*>

+Boss(string,int,int,int,int,int,int,int,int)
+ getReward(Player*) : bool
```

Result:

My dungeon game run result is as in the demo video, but there's many different ending ways, I suggest you can try it.

Discussion:

Writing this dungeon program cost me a lot of time(about 65 hours total), and when I finally coding to Dungeon.cpp, I found out maybe I shout first complete this file, so that I will not just depend on my predesigned UML to do it, O will have an top-down view to this min-project, and maybe doing this can significantly speed up my develop on this project.

Conclusion:

I don’t clearly know what this part(“Conclusion”) should write, but have no time to ask(or another 10% will be take away), so I decide to write down what I think I had learn in completing this mid-project.

First, I think doing these type of project should finish the top part of it, so that the lower part, which is a object be called, will be assigned a clearly function to finish, and the progress can be faster.

Second, this is the first time I design a OOP program, and let me have much experience on OOP, also during debugging progress, I notice many memory error, which be caused by wrong way pointer use, or variable. By solving these bugs, I have more clearly vision on pointer usage, which I didn’t get in the Program design class.

Finally, I have a problem want to ask TAs, doing this project cost me about 70 hours, is this a normal phenomenon? Thank you.