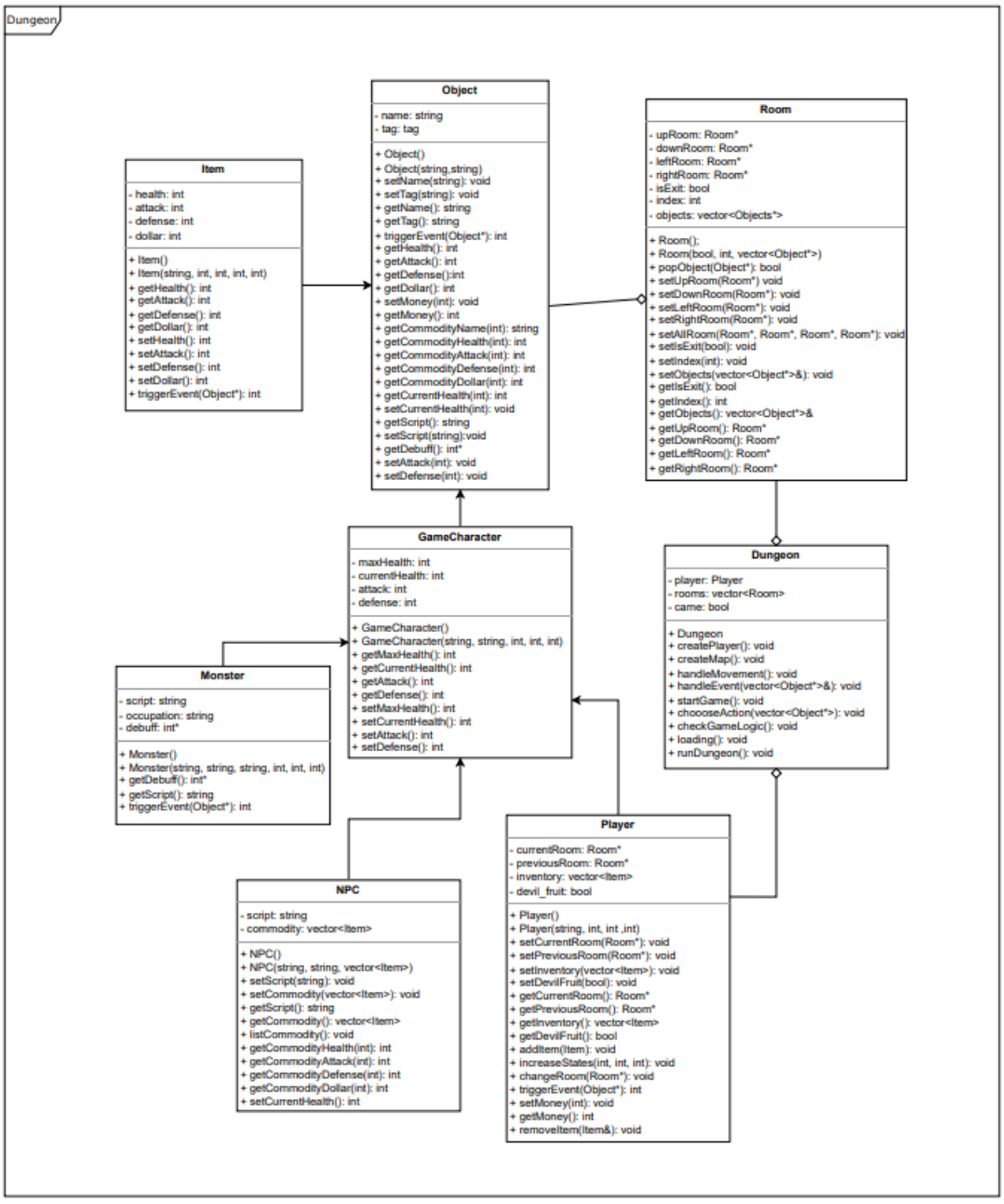
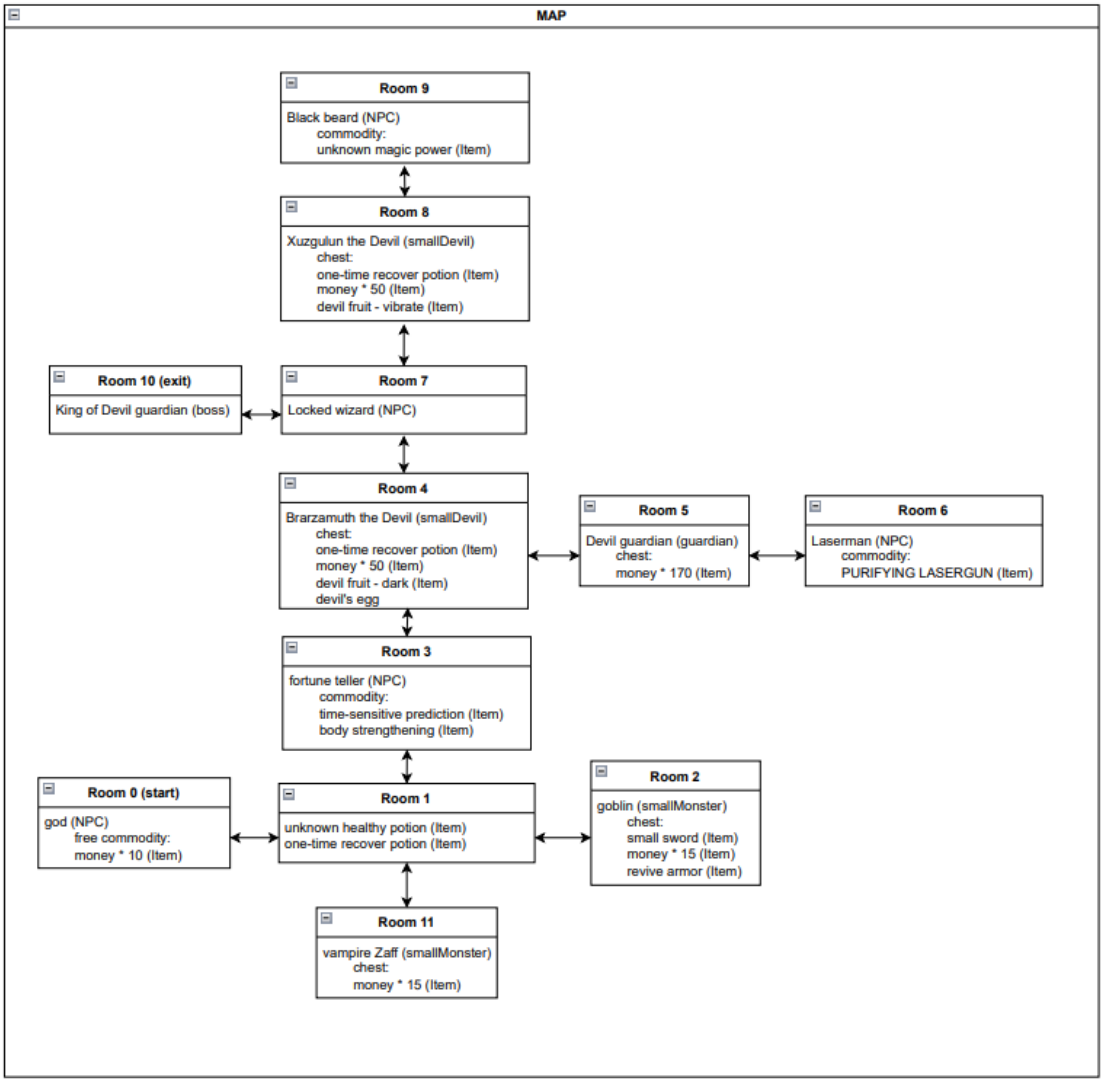
Dungeon

1. UML



.pdf at [stanleyshen2003/oop_midterm_project (github.com)]

2. Map



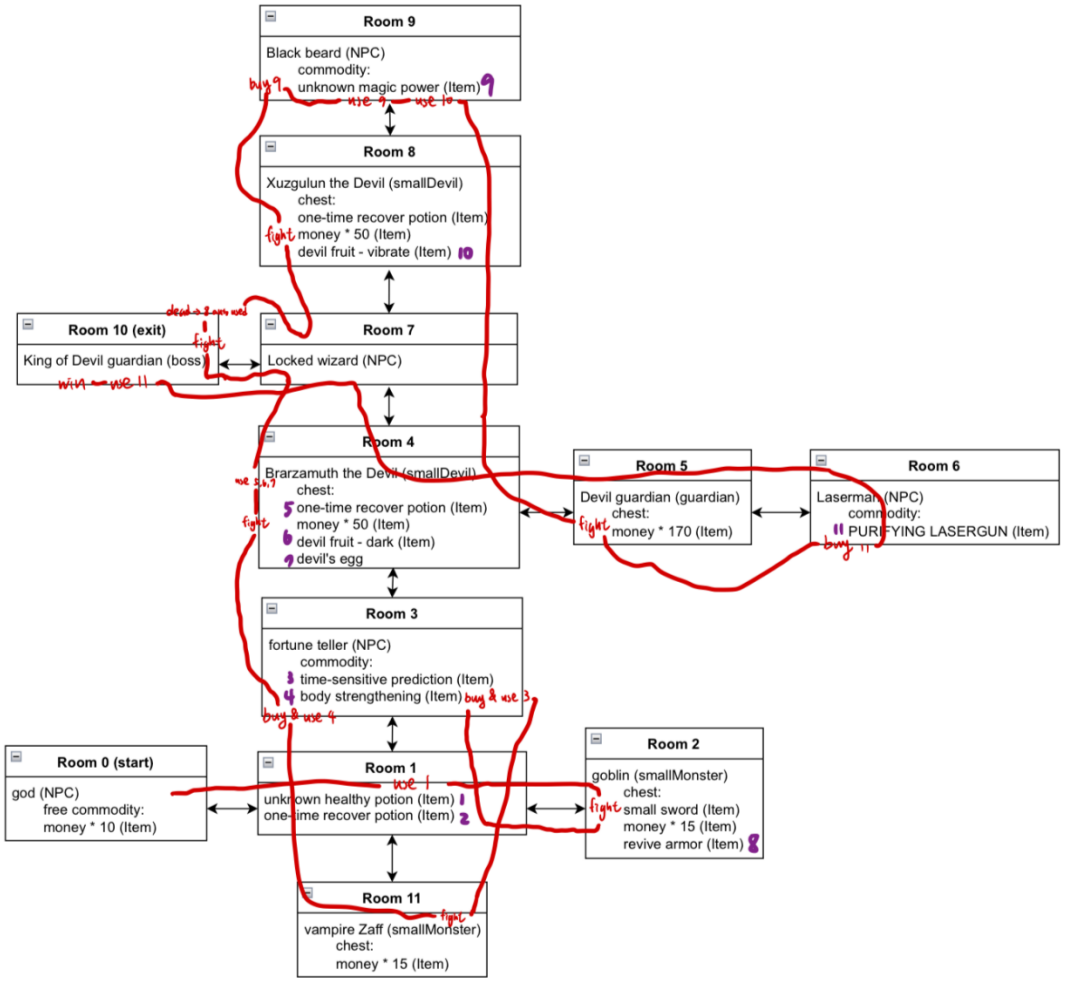
.pdf at [stanleyshen2003/oop_midterm_project (github.com)]

3. intro. to the objects

- unkown healthy potion: add random status
- one-time recover potion: recover all the loss health
- money: used when buying (can not be used in backpack)
- small sword: add ATK 15 when armed
- revive armor: revive when dead, go back to the previous room, stems from mobile game 傳說對決
- time-sensitive prediction: to know there's a strong monster in room 4
- body strengthening: add HEALTH 100; ATK 30; DEF 30
- All monsters: name from greek mythology

- devil fruit - dark/vibrate: add random status, can have only one
  Black beard: the pirate that ate two devil's fruit in the comic 海賊王, and the magic power is sells is to help the player to have 2 fruits
- devil's egg: add HEALTH 200; ATK 30; DEF 30
- PURIFYING LASERGUN: eliminate the demon, used in battle section, idea from Constantine
- King of Devil guardians: boss, can be defeated only by PURIFYING LASERGUN

4. way of solving



My design (prerequisite of fighting):

- have to use "unknown healthy potion" before fighting 2 monsters
- before fighting monster in Room 4, you have to use "body strengthening"
- before fighting monster in Room 5, you have to had two devil fruits

5. Implementation

(1) action menu (I hide some code in open backback part)

```
void Dungeon::chooseAction(vector<Object*> objects) {
    cout<<endl;
    cout<<"============================="<<endl;
    cout<<endl;
    cout<<"1 : Move"<<endl;
    cout<<"2 : Check Status"<<endl;
    cout<<"3 : Open BackPack"<<endl;
    bool isNPC = false;
    if(objects.size() != 0) {
        if(objects[0]->getTag() == "npc"){
            cout<<"4 : Buy something from the NPC"<<endl;
            isNPC = true;
        }
    }
    int trigger = 0;
    string input;
    cout<<endl;
    while(cin>>input){
        if(input=="1"){                                         // move
            handleMovement();
            break;
        }
        else if(input=="2"){                                    // show status
            player.triggerEvent(&player);
            break;
        }
        else if(input=="3"){                                    // open backpack
            vector<Item> inside = player.getInventory();
            vector<Item> temp ={};
            cout<<endl<<"(1) money * "<<inside[0].getDollar()<<endl;
            temp.push_back(inside[0]);
            inside.erase(inside.begin());
            while (inside.size()>0){
                int countAmount = 1;
                for(int i = 1;i<(int)inside.size();i++){
                    if(inside[0].getName() == inside[i].getName()){
                        countAmount++;
                        inside.erase(inside.begin() + i);
                        i--;
                    }
                }
                cout<<"("<<temp.size()+1<<") "<<inside[0].getName()<<" * "<<countAmount<<endl;
                temp.push_back(inside[0]);
                inside.erase(inside.begin());
            }
            string useOrNot="";                                  // use item in packpack
            cout<<endl<<"you want to use any of them? (y/n)"<<endl;
            while(cin>>useOrNot){
            break;
        }
        else if(input=="4" && isNPC){                           // interact with NPC
            trigger = objects[0]->triggerEvent(&player);
            if (trigger < 10) {
                Item newItem(objects[0]->getCommodityName(trigger), objects[0]->getCommodityHealth(trigger), objects[0]->getCommodityAttack(trigger), objects[0]->getCommodityDefense(trigger), 0);
                player.addItem(newItem);
                cout<<endl<<objects[0]->getName()<<" : sure, here it is."<<endl;
                //Item money = Item("money",0,0,0,objects[0]->getCommodityDollar(trigger));
                //player.removeItem(money);
            }
            break;
        }
        else{
            cout<<"invalid input!!"<<endl;
        }
    }
}
```

(2) movement

```
void Dungeon::handleMovement() {
    string input = "";
    Room* allRoom[4] = {};
    string ways[] = { "a","w","d","s" };
    allRoom[0] = player.getCurrentRoom()->getLeftRoom();
    allRoom[1] = player.getCurrentRoom()->getUpRoom();
    allRoom[2] = player.getCurrentRoom()->getRightRoom();
    allRoom[3] = player.getCurrentRoom()->getDownRoom();
    bool finish = false;
    cout << "\nchoose your direction :" << endl;
    if (allRoom[0] != NULL) {
        cout << "a) go left;  ";
    }
    if (allRoom[1] != NULL){
        cout << "w) go up;  ";
    }
    if (allRoom[2] != NULL) {
        cout << "d) go right;  ";
    }
    if (allRoom[3] != NULL) {
        cout << "s) go down;  ";
    }
    cout<<"n) not going";
    cout << endl;
    while (cin >> input) {
        if(input == "n"){
            break;
        }
        for (int i = 0;i < 4;i++) {
```

```
            if (input == ways[i]) {
                if (allRoom[i] != NULL) {
                    player.setPreviousRoom(player.getCurrentRoom());
                    player.setCurrentRoom(allRoom[i]);
                    finish = true;
                    break;
                }
            }
        }
        if (finish) {
            break;
        }
    }
    if(input!="n"){
        this->came = false;
        cout<<endl<<endl;
        cout<<"you are entering a room..."<<endl;
        usleep(500000);
    }

}
```

(3) show status

```
int Player::triggerEvent(Object* obj) {
    cout << endl<<"This is your status" << endl;
    cout << "Health : " << getCurrentHealth() << "/" << getMaxHealth() << endl;
    cout << "ATK    : " << getAttack() << endl;
    cout << "DEF    : " << getDefense() << endl;
    return 1;
}
```

(4) pickup item

- print message

```
int Item::triggerEvent(Object* obj){
    if(getName()=="chest")
        cout<<"You found and opened a chest!"<<endl;
    else if (getName() == "money") {
        cout << "\nyou received " << getDollar() << " dollars." << endl;
    }
    else{
        cout<<"\nYou picked up "<<getName()<<endl;
    }
    return 1;
}
```

- add to player (npc part is hidden)

```
for (int i = 0;i < (int)objects.size();i++) {
    ////////////////////////////////////////
    //cout<<objects[i]->getAttack()<<endl;
    if(exitRoom){
        break;
    }
    if(player.getCurrentHealth()<=0) break;
    if(objects[i]->getTag()=="npc" && this->came==false){
    if(objects[i]->getTag()=="npc"){
    usleep(500000);
    choice = player.getCurrentRoom()->getObjects()[i]->triggerEvent(objects[i]);
    if (objects[i]->getTag() == "item") {
        Item newItem(objects[i]->getName(), objects[i]->getHealth(), objects[i]->getAttack(), objects[i]->getDefense(), objects[i]->getDollar());
        if(objects[i]->getName()!="chest"){
            player.addItem(newItem);
        }
        player.getCurrentRoom()->popObject(objects[i]);
        i--;

    }
```

(5) fighting system

- monsters are store in the objects[0] of each room, and are encountered in handleEvent()

```
void Dungeon::handleEvent(vector<Object*> &objects) {
    int choice;
    bool exitRoom = false;
    for (int i = 0;i < (int)objects.size();i++) {
        ////////////////////////////////////////
        //cout<<objects[i]->getAttack()<<endl;
        if(exitRoom){
            break;
        }
        if(player.getCurrentHealth()<=0) break;
        if(objects[i]->getTag()=="npc" && this->came==false){
        if(objects[i]->getTag()=="npc"){
        usleep(500000);
        choice = player.getCurrentRoom()->getObjects()[i]->triggerEvent(objects[i]);
        if (objects[i]->getTag() == "item") {

        // battle part //
        else if (objects[i]->getTag() == "monster" ){
            if (choice == 0) {
                player.setCurrentRoom(player.getPreviousRoom());
                cout<<"you successfully retreated..."<<endl;
                this->came = true; //
                return;
            }
            else {
                vector <Item> tempInven = player.getInventory();
                for(int j=0;j<(int)tempInven.size();j++){
                    if(tempInven[j].getName()=="PURIFYING LASERGUN"){
                        cout<<"do you want to activate the PURIFYING LASERGUN? (y/n)"<<endl;
                        string x="";
                        while(cin>>x){
                            if(x=="y" || x == "n")
                                break;
                        }
                        if(x=="y"){
                            cout<<"\nLASERRRRRRRRRRRRRR!"<<endl;
                            cout<<"Devil eliminated!!!\n"<<endl;
                            objects[i]->setCurrentHealth(0);
                        }
                    }
                }
                if(player.getDevilFruit()){
                    cout << objects[i]->getName()<<": you are... Devilll!!!!\n";
                    cout << "the devil is debuffed" << endl;
                    int* debuff = objects[i]->getDebuff();
                    objects[i]->setCurrentHealth(objects[i]->getCurrentHealth()-debuff[1]);
                    objects[i]->setAttack(objects[i]->getAttack()-debuff[1]);
                    objects[i]->setDefense(objects[i]->getDefense()-debuff[2]);
                }
                while (player.getCurrentHealth() > 0 && objects[i]->getCurrentHealth() > 0) {
                    int temp;
                    temp = objects[i]->getDefense()-player.getAttack();
                    if(temp>=0) temp=-1;
                    objects[i]->setCurrentHealth(objects[i]->getCurrentHealth() + temp);
                    if (objects[i]->getCurrentHealth() <= 0) {
                        cout << "\nyou won!\n" << endl;
                        player.getCurrentRoom()->popObject(objects[i]);
                        i--;
                        break;
                    }
                    temp = player.getDefense()-objects[i]->getAttack();
                    if(temp>=0) temp=-1;
                    player.setCurrentHealth(player.getCurrentHealth() + temp);
                }
                if (player.getCurrentHealth() <= 0) {
                    vector <Item> tmp = player.getInventory();
                    for (int i = 0;i < (int)tmp.size();i++) {
                        if (tmp[i].getName() == "revive armor") {
                            cout<<"\nyou are dead during the fight..."<<endl;
                            cout<<"revive armor activated..."<<endl;
                            cout<<"you are sent to the previous room"<<endl;
                            player.setCurrentRoom(player.getPreviousRoom());
                            player.setCurrentHealth(player.getMaxHealth());
                            exitRoom = true;
                            player.removeItem(player.getInventory()[i]);
                            break;
                        }
                    }
                }
            }
        }
    }
}
```

The code deal with the battle part, debuff and 2 of the item usage.

Damage calculation: damage = max(0, attacker's ATK – defender's DEF)

Debuff: tiggered when the player has eaten the devil fruit. Monster's status decrease by the value stored in the monster

Item use: "PURIFYING LASER" is used before battle, "revive armor" is used automatically when dead.

- choose to fight or not

```
int Monster::triggerEvent(Object* obj) {
    cout << "you bumped into " << getName() << "." << endl;
    cout << script;
    cout << "do you want to have a fight? (y/n)" << endl;
    string fight;
    cin >> fight;
    while (fight != "y" && fight != "n") {
        cout << "invalid input!" << endl;
        cout << "do you want to have a fight? (y/n)" << endl;
        cin >> fight;
    }
    if (fight == "y")
        return 1;
    else {
        return 0;
    }

}
```

(6) NPC

- NPCs are store in the objects[0] of each room, and are encountered in handleEvent, when entering the room, the NPC will say its script

```
void Dungeon::handleEvent(vector<Object*> &objects) {
    int choice;
    bool exitRoom = false;
    for (int i = 0;i < (int)objects.size();i++) {
        ////////////////////////////////////////
        //cout<<objects[i]->getAttack()<<endl;
        if(exitRoom){
            break;
        }
        if(player.getCurrentHealth()<=0) break;
        if(objects[i]->getTag()=="npc" && this->came==false){
            cout<<"\n\nA weird guy comes and talks to you"<<endl;
            loading();
            cout<<objects[i]->getScript()<<endl;
            if(objects[i]->getName()=="god"){
                objects[i]->setScript("god : It seems that we've met before...\ngod : You should get going...");
            }
            this->came = true;
            continue;
        }
        if(objects[i]->getTag()=="npc"){
            continue;
        }
```

- triggerEvent() deal with the buying action

```
int NPC::triggerEvent(Object* player) {
    if(commodity.size()==0){
        cout<<endl<<getName()<<" : Um... I have nothing to sell XD"<<endl;
        cout<<getName()<<" : the only thing I could say is..."<<endl;
        cout<<script;
        cout<<endl;
        return 200;
    }
    string buy = "";
    cout<<endl<<getName() << " : do you want to buy anything?" << endl<<endl;
    listCommodity();
    cout <<endl<<getName()<< " : which one do you want?" << endl;
    buy = "";
    cin >> buy;
    while (buy.length() > 1) {
        cin >> buy;
    }
    int choice;
    choice = buy[0] - '1';
    if (choice < commodity.size()) {
        if (commodity[choice].getDollar() > player->getMoney()) {
            cout << "not enough money!!" << endl;
        }
        else {
            player->setMoney(player->getMoney() - commodity[choice].getDollar());
            return choice;
        }
    }
    return 100;
}
```

- In the action menu, you can buy something from NPC, the following code is in chooseAction()

```
else if(input=="4" && isNPC){                          // interact with NPC
    trigger = objects[0]->triggerEvent(&player);
    if (trigger < 10) {
        Item newItem(objects[0]->getCommodityName(trigger), objects[0]->getCommodityHealth(trigger), objects[0]->getCommodityAttack(trigger), objects[0]->getCommodityDefense(trigger), 0);
        player.addItem(newItem);
        cout<<endl<<objects[0]->getName()<<" : sure, here it is."<<endl;
        //Item money = Item("money",0,0,0,objects[0]->getCommodityDollar(trigger));
        //player.removeItem(money);
    }
    break;
}
else{
    cout<<"invalid input!!"<<endl;
}
```

(7) Game logic

- The condition to break

```
bool Dungeon::checkGameLogic() {
    if (player.getCurrentRoom()->getIsExit() && player.getCurrentRoom()->getObjects()[0]->getCurrentHealth()<=0)
        return false;
    if (player.getCurrentHealth() <= 0)
        return false;
    return true;
}
```

- Then implement the code in runDungeon()

```
void Dungeon::runDungeon() {
    startGame();
    cout<<"game started!\nyou entered a room..."<<endl;
    while (checkGameLogic()) {
        handleEvent(player.getCurrentRoom()->getObjects());
        if(player.getCurrentHealth()<=0 || rooms[10].getObjects()[0]->getCurrentHealth()<=0)
            break;
        chooseAction(player.getCurrentRoom()->getObjects());
    }
    if(player.getCurrentHealth()>0){
        cout<<"Congratulations!\nYou are such a strong fighter\nYOU WON!"<<endl;
    }
    else{
        cout<<"oops, you are dead\ntry again next time!"<<endl;
    }
}
```

I used the handleEvent first because I think it is better to stay in one room in every iteration, so I add the break condition.

(8) Character class design

I designed it in Class Monster, there are 4 kind of occupations: smallMonster, smallDevil, guardian, boss. They have status (HEALTH, ATK, DEF) : (50, 12, 12), (200, 40, 40), (700, 140, 140), (999999, 999999, 999999) respectively.

```
Monster::Monster(string name,string occupation,string script, int dehealth, int deattack, int dedefense):GameCharacter(name,"monster",50,12,12){
    this->occupation = occupation;
    this->debuff = new int[3];
    this->debuff[0] = dehealth;
    this->debuff[1] = deattack;
    this->debuff[2] = dedefense;
    this->script = script;

    if(occupation == "smallDevil"){
        this->setMaxHealth(200);
        this->setCurrentHealth(200);
        this->setAttack(40);
        this->setDefense(40);
    }
    else if(occupation== "guardian"){
        this->setMaxHealth(700);
        this->setCurrentHealth(700);
        this->setAttack(140);
        this->setDefense(140);
    }
    else if(occupation=="boss"){
        this->setMaxHealth(999999);
        this->setCurrentHealth(999999);
        this->setAttack(999999);
        this->setDefense(999999);
    }
}
```

The debuff is triggered when you ate the devil fruit (because there will be a strong devil in your body), it is implemented in the battle part

```
if(player.getDevilFruit()){
    cout << objects[i]->getName()<<": you are... Devilll!!!!\n";
    cout << "the devil is debuffed" << endl;
    int* debuff = objects[i]->getDebuff();
    objects[i]->setCurrentHealth(objects[i]->getCurrentHealth()-debuff[1]);
    objects[i]->setAttack(objects[i]->getAttack()-debuff[1]);
    objects[i]->setDefense(objects[i]->getDefense()-debuff[2]);
}
```

The script is printed in triggerEvent()

```
int Monster::triggerEvent(Object* obj) {
    cout << "you bumped into " << getName() << "." << endl;
    cout << script;
    cout << "do you want to have a fight? (y/n)" << endl;
    string fight;
    cin >> fight;
    while (fight != "y" && fight != "n") {
        cout << "invalid input!" << endl;
        cout << "do you want to have a fight? (y/n)" << endl;
        cin >> fight;
    }
    if (fight == "y")
        return 1;
    else {
        return 0;
    }

}
```

(9) Other details

- Use string to deal with invalid inputs. For example:

```
string input;
cout<<endl;
while(cin>>input){
    if(input=="1"){
        handleMovement();
        break;
    }
    else if(input=="2"){
        player.triggerEvent(&player);
        break;
    }
```

- Money stored only in player.inventory[0]

```
void Player::setMoney(int money) {
    addItem(Item("money", 0, 0, 0, money - inventory[0].getDollar()));
}

void Player::addItem(Item additem) {
    if (additem.getName() == "money" && inventory.size()!=0) {
        inventory[0].setDollar(inventory[0].getDollar() + additem.getDollar());
    }
    else {
        inventory.push_back(additem);
    }
}

void Player::removeItem(Item &item){
    if(item.getName()=="money"){
        inventory[0].setDollar(inventory[0].getDollar()-item.getDollar());
        return;
    }
    for(int i=1;i<inventory.size();i++){
        if(inventory[i].getName()==item.getName()){
            inventory.erase(inventory.begin()+i);
            break;
        }
    }
}
```

- Constraint of having 1 devil fruit and how to have 2

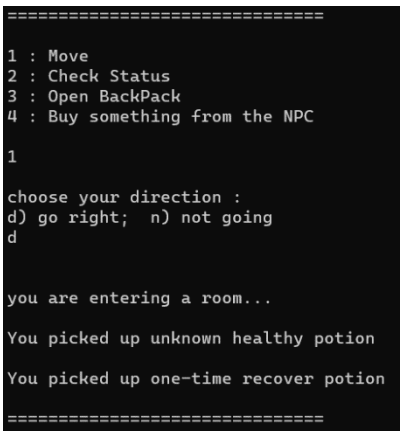
  player.devil_fruit is true if you have a devil fruit, and the magic power can set player.devil_fruit to be false.

```
else if(temp[use].getName()=="unknown healthy potion" || temp[use].getName()=="body strengthening" ||tem
else if(temp[use].getName()=="one-time recover potion"){
else if(temp[use].getName()=="time-sensitive prediction"){
else if(temp[use].getName()=="devil fruit - dark" || temp[use].getName()=="devil fruit - vibrate"){
    if(player.getDevilFruit()==false){
        player.increaseStates(temp[use].getHealth(),temp[use].getAttack(),temp[use].getDefense());
        player.setCurrentHealth(player.getCurrentHealth()+temp[use].getHealth());
        cout<<"\noops, you ate the devil fruit! QuO"<<endl;
        player.removeItem(temp[use]);
        player.setDevilFruit(true);
    }
    else{
        cout<<"\nYou can ONNNNNNNLLLLLLY eat 1 Devil fruit at a time!!"<<endl;
    }
}
else if(temp[use].getName()=="unknown magic paper"){
    if(player.getDevilFruit()==false){
        cout<<"you should eat one first!"<<endl;
    }
    else {
        cout<<"\na weird voice appeared..."<<endl;
        cout<<"Black beard : let me give you some help with the devil fruits!"<<endl;
        cout<<"you now know how to have 2 devil fruits!"<<endl;
        player.setDevilFruit(false);
        player.removeItem(temp[use]);
    }
}
player.triggerEvent(&player);
break;
```

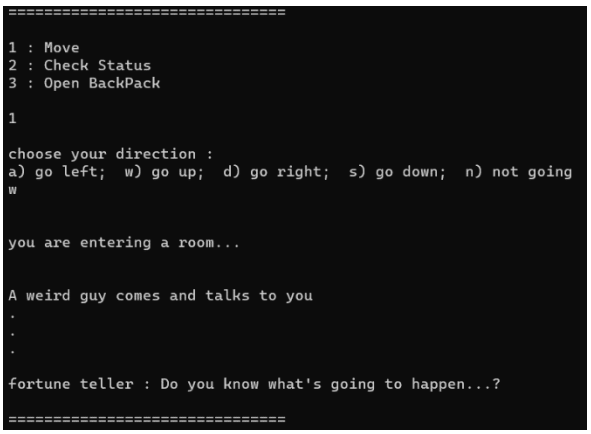
- Animation for NPC (use library to pause)

```
void Dungeon::loading(){
    for(int i=0;i<3;i++){
        cout<<"."<<endl;
        usleep(500000);
    }
    cout<<endl;
}
```

6. Results



Move & pickup



Move & ecounter

```
================================

1 : Move
2 : Check Status
3 : Open BackPack
4 : Buy something from the NPC

4

fortune teller : do you want to buy anything?

here are all the items I have :
1) time-sensitive prediction
2) body strengthening
3) not buying...

fortune teller : which one do you want?
1

fortune teller : sure, here it is.

================================
```

Buy inventory

```
==============================

1 : Move
2 : Check Status
3 : Open BackPack

3

(1) money * 7
(2) unknown healthy potion * 1
(3) one-time recover potion * 1
(4) time-sensitive prediction * 1

you want to use any of them? (y/n)
y

which one would you like to use?
2

you drank the potion...

This is your status
Health : 152/152
ATK    : 14
DEF    : 14

==============================
```

Use item

```
==============================

1 : Move
2 : Check Status
3 : Open BackPack

1

choose your direction :
a) go left;  w) go up;  d) go right;  s) go down;  n) not going
d


you are entering a room...
you bumped into goblin.
goblin: MONEY!!!
do you want to have a fight? (y/n)
y

you won!

You found and opened a chest!

You picked up small sword

you received 15 dollars.

You picked up revive armor

==============================
```

Fighting

```
==============================

1 : Move
2 : Check Status
3 : Open BackPack
4 : Buy something from the NPC

1

choose your direction :
w) go up;  s) go down;  n) not going
w


you are entering a room...
you bumped into Brarzamuth the Devil.
Brarzamuth the Devil: ARE YOU AFRAID??
do you want to have a fight? (y/n)
y

you are dead during the fight...
revive armor activated...
you are sent to the previous room

==============================
```

Dead during fight, revive armor activated

```
==============================

1 : Move
2 : Check Status
3 : Open BackPack

3

(1) money * 107
(2) one-time recover potion * 3
(3) time-sensitive prediction * 1
(4) devil fruit - vibrate * 1

you want to use any of them? (y/n)
y

which one would you like to use?
4

You can ONNNNNNNLLLLLLY eat 1 Devil fruit at a time!!

This is your status
Health : 726/737
ATK    : 128
DEF    : 111

==============================
```

Taking 2 devil fruit

```
you are entering a room...
you bumped into King of Devil guardian.
King of Devil guardian: I'm the chosen one
do you want to have a fight? (y/n)
y
do you want to activate the PURIFYING LASERGUN? (y/n)
y

LASERRRRRRRRRRRRRRR!
Devil eliminated!!!

King of Devil guardian: you are... Devilll!!!!
the devil is debuffed
Congratulations!
You are such a strong fighter
YOU WON!

Process returned 0 (0x0)   execution time : 1274.595 s
Press any key to continue.
```

Using laser & win

7. Discussion

(1) function based programming vs. OOP in game making

If you use functional programming on making a complex game, the code will be messy and hard to read. Before I take the course, I made a game using function based programming in another programming language (space game - OpenProcessing). You can see that there are approximately 500 lines of code for all kinds of functions, and it is really hard to understand the structure by reading the code. In my work, there are several objects like bullet, rocks, and player, and I used lots of arrays to store their variable which makes the code hard to understand, but if I had done it with OOP, I only need an array of object to store all of them, which will make the main function neater and easier to maintain.

(2) Why virtual function?

Virtual function is placed in the base class and overrided in the derived class and the function of derived class version can be called if you have a derived class object stored in a based class pointer. If we do not use the virtual function, the objects of each room will have to own a vector for each type of objects, i.e. vector<NPC>, vector<Monster>, and vector<Item>. This structure will make it harder than vector<Object*> when we want to design more kind of object in the room. For example, if you want to add a object of a new case, you will have to modify the class Room with a vector of new object, but if you use virtual function, you can save the new class object in vector<Object*> and call the object by using virtual functions.

8. Conclusion

In the future, I might add some additional system for my dungeon game, for example, a random event system or a pet system that can turn the monster you were fighting with to your companion pet with a pokemon ball or something else.

Overall, I think the project helped me a lot on knowing what the general structure of a game written in OOP paradigm should be like. In this project, I had a more concrete idea on several OOP techniques like inheritence and virtual functions, especially on the using of virtual function and why it should be used.