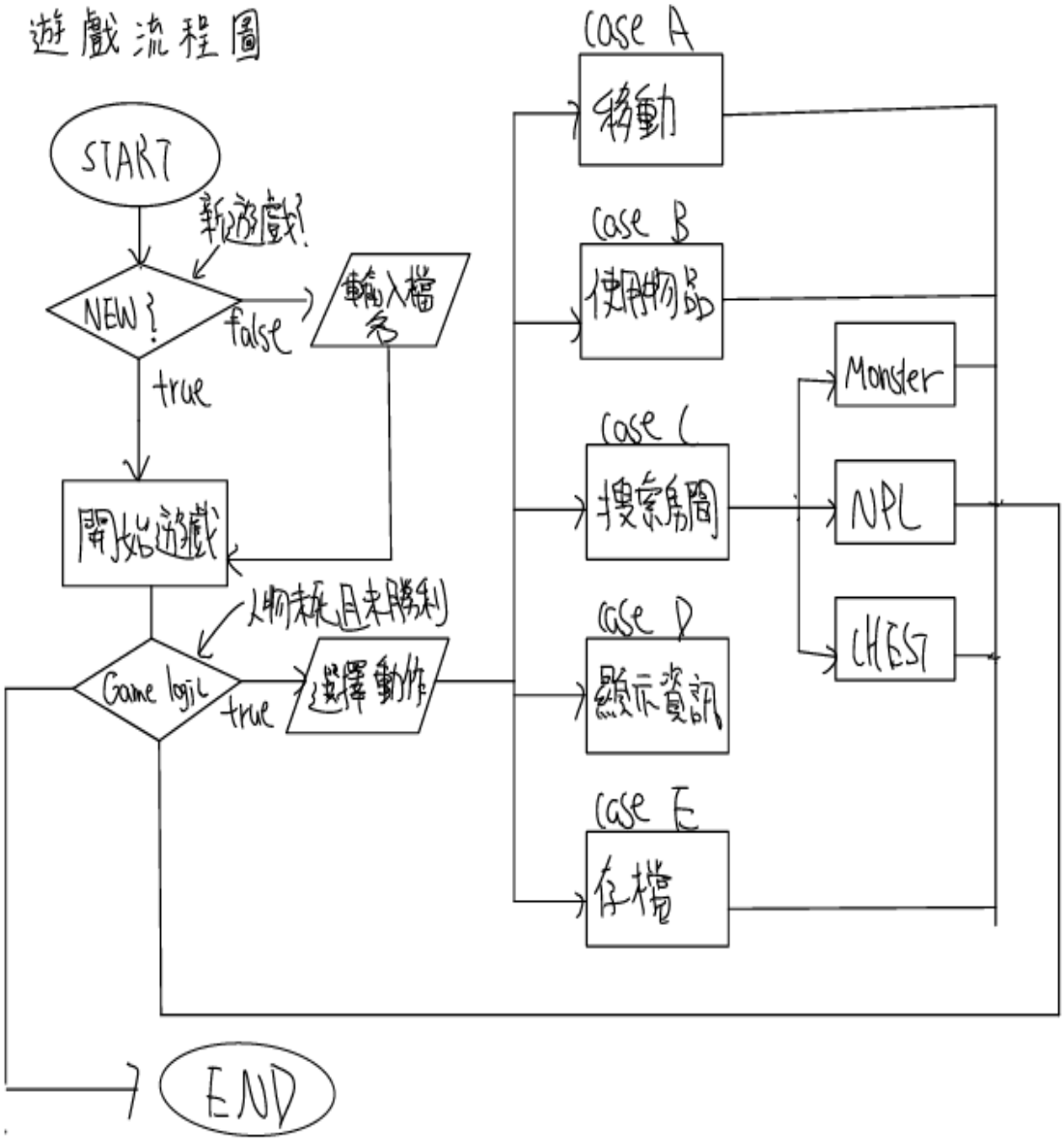
OOP 期中作業

DUNGEON GAME

0711506

王偉誠

遊戲流程圖


START
新遊戲?
NEW?
false
輸入檔名
true
開始遊戲
人物未死且未勝利
Game logic
true
選擇動作
case A
移動
case B
使用物品
case C
搜索房間
case D
顯示資訊
case E
存檔
Monster
NPC
CHEST
END

Introduction

- 這個期中作業中，整個 project 包含 7 個 class 和一個自己建立的工具函數庫。(其繼承關係可見下文之 UML 圖)
- 遊戲出現之物件基本介紹

  1. 物件介紹

     - ✓ Object：遊戲內的基本物件，其餘幾個物件都是繼承這個 class
     - ✓ GameCharacter：在遊戲中出現的角色
     - ✓ Player：玩家
     - ✓ NPC：在遊戲內出現的非怪物角色
     - ✓ Chest：寶箱，玩家找到後可以打開獲得＂驚喜＂
     - ✓ Monster：怪物，玩家遭遇後可以選擇進行戰鬥或撤退
     - ✓ Item：物品，可以在遊戲裡透過與 NPC 或擊倒怪物獲得
       - ◆ Recovery：可能由怪物掉落或者與 NPC 交易獲得，用來回復 health
       - ◆ Weapon：可能由怪物掉落或者與 NPC 交易獲得，用來增加 attack
       - ◆ Treasure：可能由怪物掉落或者與 NPC 交易獲得，

用來施放課金一擊(ultimate skill)與破關條件之一

2. Task 系統

- 遊戲中需要滿足的破關條件有下列三項
  - ✓ Task 1：kill monster *1
  - ✓ Task 2：collect item *1
  - ✓ Task3：find the exit

3. Interaction 系統

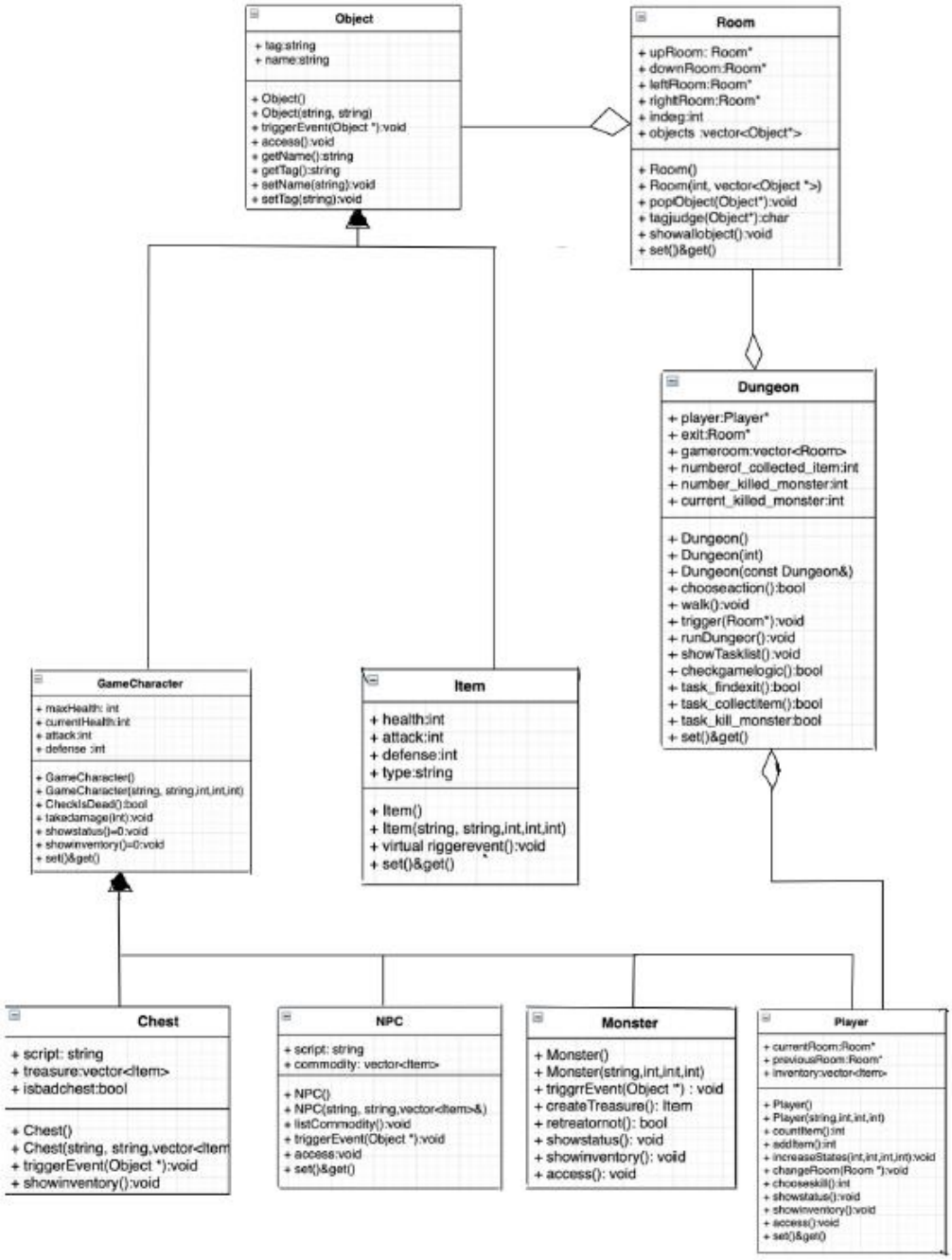
- 在遊戲進行過程中，玩家會與 NPC 及 monster 等物件互動
  - ✓ NPC：玩家跟和他們進行交易，獲得 item
  - ✓ Monster：玩家可與其進行對戰，戰勝可獲得掉落物，戰敗死亡則遊戲結束
  - ✓ Chest：在搜索房間時，有可能會發現寶箱，打開會得到驚喜，可能是怪物或是真的寶物

4. Record 系統

- 在 Dungeon 中設計有存檔系統，玩家可以延續上次的遊戲進度
  - ✓ 讀檔：可以自行輸入設定檔名稱，自行設定遊戲或讀取舊檔延續進度

✓ 存檔：預設是儲存於 backupfile.txt 中，玩家也可自行修改檔名，讀檔時只要輸入指定的檔名即可讀取進度。

UML



Implementation details

Header File

1. Dungeon.h

```
#ifndef DUNGEN_H_INCLUDED
#define DUNGEN_H_INCLUDED
#include "Player.h"
#include "Room.h"
#include "Object.h"

using namespace std ;

class Dungeon {
private:
    Player *player;
    Room *exitroom;
    vector <Room> gameroom;
    int numberof_collected_item;
    int numberof_killed_monster;
    int current_killed_monster;
public:
    Dungeon();
    Dungeon(int);
    Dungeon(const Dungeon&);
    bool chooseaction();
    void walk();
    void trigger(Room *);
    void runDungeor();
    void showTasklist();
    void showinformation();
    bool checkgamelogic();
    bool task_findexit();
    bool task_collectitem();
    bool task_kill_monster();

    /*  Getter   */
    vector<Room> getGameroom()const;
    Player *getPlayer()const;
    Room *getExitroom()const;
    int getNumberofcollectItem()const;
    int getNumberofkilledMonster()const;
    int getCurrentkilledMonster()const;

};

#endif // DUNGEN_H_INCLUDED
```

2. Object.h

```cpp
#ifndef OBJECT_H_INCLUDED
#define OBJECT_H_INCLUDED

#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Object
{
private:
    string name;
    string tag;
public:
    Object();
    Object(string,string);

    /* pure virtual function */
    virtual void triggerEvent(Object *) ;

    /* Setter & Getter*/
    void setName(string);
    void setTag(string);
    string getName();
    string getTag();

    virtual void access(){} ;
};

#endif // OBJECT_H_INCLUDED
```

3. GameCharacter.h

```cpp
#ifndef GAMECHARACTER_H_INCLUDED
#define GAMECHARACTER_H_INCLUDED

#include <iostream>
#include <string>
#include "Object.h"
using namespace std;

class GameCharacter: public Object
{
private:
    int maxHealth;
    int currentHealth;
    int attack;
    int defense;
public:
    GameCharacter();
    GameCharacter(string,string,int,int,int);
    bool checkIsDead();
    void takeDamage(int);

    /* Set & Get function*/
    void setMaxHealth(int);
    void setCurrentHealth(int);
    void setAttack(int);
    void setDefense(int);
    int getMaxHealth();
    int getCurrentHealth();
    int getAttack();
    int getDefense();

    /* show function*/
    virtual void showstatus()=0;
    virtual void showinventory()=0;
};
#endif // GAMECHARACTER_H_INCLUDED
```

4. NPC.h

```cpp
#ifndef NPC_H_INCLUDED
#define NPC_H_INCLUDED

#include <iostream>
#include <string>
#include <vector>
#include "GameCharacter.h"
#include "Player.h"
#include "Item.h"
#include "Object.h"

using namespace std;

class NPC: public GameCharacter
{
private:
    string script;
    vector<Item> commodity;
public:
    NPC();
    NPC(string, string, vector<Item>&);
    void listCommodity(); /*print all the Item in this NPC*/

    /* Virtual function that you need to complete   */
    /* In NPC, this function should deal with the   */
    /* transaction in easy implementation           */
    void triggerEvent(Object*);

    /* Setter & Getter*/
    void setScript(string);
    void setCommodity(vector<Item>);
    void showstatus();
    void showinventory();
    string getScript();
    vector<Item> getCommodity();
    void access();
};


#endif // NPC_H_INCLUDED

```

5. Monster.h

```cpp
#ifndef ENEMY_H_INCLUDED
#define ENEMY_H_INCLUDED

#include <iostream>
#include <string>
#include <vector>
#include "GameCharacter.h"
#include "Player.h"
#include "Object.h"

using namespace std;
class Monster: public GameCharacter
{
private:
public:
    Monster();
    Monster(string,int,int,int);
    //Item treasure ;
    /* Virtual function that you need to complete   */
    /* In Monster, this function should deal with   */
    /* the combat system.                           */
    void triggerEvent(Object*);
    Item createTreasure();
    bool retreatornot();
    //virtual
    void showstatus();
    void showinventory();
    void access();
};


#endif // ENEMY_H_INCLUDED

```

6. Chest.h

```cpp
#ifndef CHEST_H_INCLUDED
#define CHEST_H_INCLUDED
#include "GameCharacter.h"
#include "Item.h"
#include "Object.h"
#include "Player.h"
#include <vector>

using namespace std;

class Chest :public GameCharacter
{
private:
    string script;
    vector<Item> treasure;
    bool isbadchest;
public:
    Chest();
    Chest(string, string,vector<Item>&,bool);

    /*Setter & Getter*/
    string getScript();
    vector<Item> getTreasure();
    bool getisbadchest();
    void setScript(string);
    void setTreasure(vector<Item>);
    void setIsbadchest(bool);
    /*virtual function*/
    void triggerEvent(Object *);
    void showinventory();
    void showstatus();
    void access();
};

#endif // CHEST_H_INCLUDED
```

7. tool.h

```cpp
#ifndef TOOL_H_INCLUDED
#define TOOL_H_INCLUDED


#include"Object.h"
#include "Room.h"
#include "Dungen.h"
#include<vector>
#include<string>
#include<iostream>
#include<fstream>

using namespace std;

int getnumberfrom_ifstream();
int findpos(vector<Object *> &inventory,string s);
int findpos2(vector<Item> &ll,string type);

vector<Object *> setroomobject(ifstream &,int);

void traverseroom(vector<Room> roomlist);
void backup(Dungeon &dun);
void saveplayer(Player *,vector<Room>,ofstream &);
Player* reading(Player *playptr,vector<Room> &room,int & exit,int &numberofcollect,int &currentkilled,int &numberofkilled,char *,ifstream &);

#endif // TOOL_H_INCLUDED
```

Virtual Function

1. 設計

在這個遊戲中，設計了 4 個 virtual function 來幫助實作

I. tiggerEvent()

- ✓ tiggerEvent 宣告在 Object 中，使用 virtual 的目的是為了讓 room 中用來存取 upcast 物件的 Object* 可以正確存到理想的 tiggerEvent
- ✓ 在 monster 實作為戰鬥系統
- ✓ 在 Chest 實作為開寶箱
- ✓ 在 NPC 實作為交易系統

II. access()

- ✓ access 設計的用意是作為媒介，讓我可以在沒有 downcast 的狀況下，藉由 dynamic link 對物件進行存取
- ✓ 如下圖的實作 showinventory 是定義於 GameCharacter 中，Object*不能直接呼叫，但藉由 virtual function 就可以成功呼叫

```
void Player::access(){
    showInventory();
};
```

III. showinventory()、showstatus()

- ✓ 為了確保用 Object*或 GameCharacter*指標存取資訊時，能夠 link 到正確的 function，所以將顯示資訊的 function 定義為 virtual function
- ✓ 這兩個 function 用途類似，都是用來顯示資訊，只是因為不同的 case 需要顯示資訊不同，所以才區分成兩個不同的 function

2. 討論

- 這個 project 中，我主要用的 virtual function 是 tiggerEvent，因為在實作時幾乎都直接將 Object* downcast 到我需要的型態，結果對其他 3 個 virtual function 使用似乎不夠多。本來在思考要不要把這三個改回一般 function，讓效率提升一點，但覺得可以保留未來擴充性，就沒有改了。

GameStart

1. Code

```
Dungeon::Dungeon(int i)
{
    srand(unsigned(time(NULL)));
    int exitindex;
    cout<<"Do you want to load previous data ?\nA.Yes\nB.No,Start a new game"<<endl;
    char *filename=new char[100];
    char c;
    cin>>c;
    cin.get();
    if(c=='a'||c=='A'){
        cout<<"please input the file name"<<endl;
        cin.getline(filename,100);
        cout<<filename<<endl;
        ifstream infile(filename);
        player=reading(player,gameroom,exitindex,numberof_collected_item,current_killed_monster,numberof_killed_monster,filename,infile);

    }
    else if(c=='b'||c=='B'){
        filename="backup.txt";
        ifstream infile(filename);
        player=reading(player,gameroom,exitindex,numberof_collected_item,current_killed_monster,numberof_killed_monster,filename,infile);
        cout<<"please input the player name"<<endl;
        string str;
        cin>>str;
        player->setName(str);
    }

    exitroom=&gameroom[exitindex];
}
```

```
void Dungeon::runDungeon()
{
    Dungeon backdun=(*this);
    cout<<"Hello "<<player->getName()<<" Welcome to Dungeon !"<<endl;

    char c;
    if(chooseaction())
        cout<<"victory !"<<endl;
    else
        cout<<"game over >_< "<<endl;
    cout<<"try again ? (y/n) "<<endl;
    cin>>c;
    if(c=='y'){
        backdun.runDungeon();
    }

    cout<<"thank you!"<<endl;
}
```

2. 說明

I. 呼叫 constructor，先詢問玩家要讀取舊檔或是開始新遊戲，再用 reding()讀取指定的 txt 檔中的資料。

II. 取得資料後開始執行 rundungeon()

III. 先用 copy constructor 備份初始化資料，然後開始遊戲，由 chooseaction 判斷輸出 victory 或 game over

IV. 最後詢問玩家是否要重新開始遊戲，如果要重新開始，則由備份的初始資料 backdun 執行 rundungeon()。

Choose Action

1. Code

```
bool Dungeon::chooseaction(){
    char tmp ;
    bool conti  = checkgamelogic();
    while(conti){
        cout << "What do you want to do ?" << endl ;
        cout << "A.Move\nB.Using item\nC.Search the room\nD.Show information\nE.Save file" << endl ;
        cin >> tmp ;
        switch(tmp){
            case 'a' : case 'A' : walk(); break ;
            case 'b' : case 'B' : player->usingItem();break;
            case 'c' : case 'C' : trigger(player->getCurrentRoom()); break;
            case 'd' : case 'D' : showinformation();break;
            case 'e' : case 'E' : backup(*this);break;
            default : cout<<" Invalid input " << endl; conti = false ;break;
        }
        conti=checkgamelogic();
    }
    if(task_findexit()&&task_kill_monster()&&task_collectitem()){
        showTasklist();
        cout<<"Congradulation you finish all the task !!"<<endl;
    }


    if(player->getCurrentHealth()<=0)
        return false;
    else
        return true;

};
```

2. 說明

- 主要功能是依玩家輸入指令不同，呼叫不同的功能
- Walk
- UsingItem
- Trigger
- ShowInformation
- Backup

3. 結果

```
What do you want to do ?
A.Move
B.Using item
C.Search the room
D.Show information
E.Save file
```

Movment(walk)

1. Code

```
void Dungeon::walk(){
    if(player->getCurrentRoom()->getUpRoom() != NULL ){
        cout << "U. Go up into Room" << player->getCurrentRoom()->getUpRoom()->getIndex() << " \n" ;
    }
    if(player->getCurrentRoom()->getDownRoom() != NULL ){
        cout << "D. Go down into Room" << player->getCurrentRoom()->getDownRoom()->getIndex() << " \n" ;
    }
    if(player->getCurrentRoom()->getLeftRoom() != NULL ){
        cout << "L. Go left into Room" << player->getCurrentRoom()->getLeftRoom()->getIndex() << " \n" ;
    }
    if(player->getCurrentRoom()->getRightRoom() != NULL ){
        cout << "R. Go right into Room" << player->getCurrentRoom()->getRightRoom()->getIndex() << " \n" ;
    }

    char tmp ;
    cin>>tmp;
    switch(tmp){
        case 'u' : case 'U' :   player->setPreviousRoom(player->getCurrentRoom());
                                player->setCurrentRoom(player->getCurrentRoom()->getUpRoom());
                                break;
        case 'd' : case 'D' :   player->setPreviousRoom(player->getCurrentRoom());
                                player->setCurrentRoom(player->getCurrentRoom()->getDownRoom());
                                break;
        case 'l' : case 'L' :   player->setPreviousRoom(player->getCurrentRoom());
                                player->setCurrentRoom(player->getCurrentRoom()->getLeftRoom());
                                break;
        case 'r' : case 'R' :   player->setPreviousRoom(player->getCurrentRoom());
                                player->setCurrentRoom(player->getCurrentRoom()->getRightRoom());
                                break;
        default  : cout<<"Invalid input"<<endl;
    }
    if(!task_findexit())
        cout << "Enter the Room" << player->getCurrentRoom()->getIndex() << endl ;
    else{
        cout<< "You may found the exit! Check you task state!"<<endl;
    }

    if(player->getCurrentRoom()->getObjects().empty()){
        return;
    }
    else if(player->getCurrentRoom()->objects[0]->getTag()=="Monster"){
        trigger(player->getCurrentRoom());
    }
};
```

2. 說明

先判斷四周是否有房間，如果房間存在則輸出該房間的選項接著由使用者輸入指定的路徑，根據得到的資料做判斷，並藉由 setCurrentRoom 及 setPreviousRoom 來改變玩家當前位置進入房間後檢查該房間是否為出口及該房間是否存在 monster，若存在 monster 則開始與 monster 互動(註：trigger()會在後面說明)

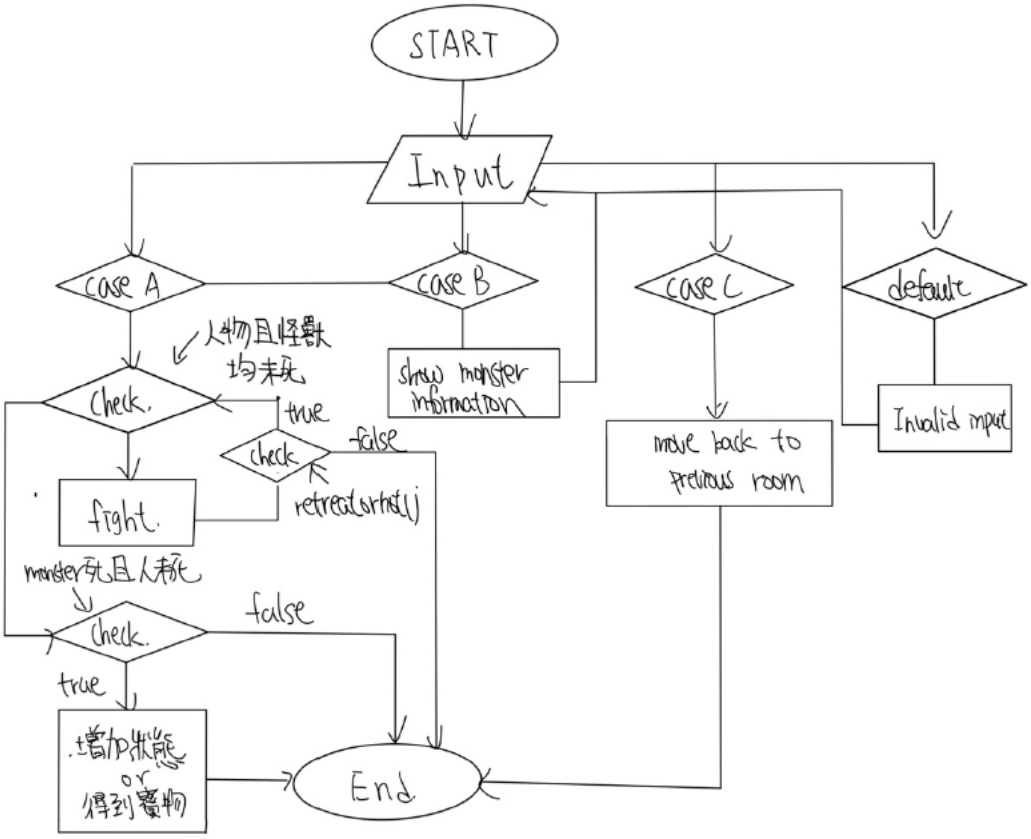
3. 結果

```
What do you want to do ?
A.Move
B.Using item
C.Search the room
D.Show information
E.Save file
a
D. Go down into Room8
R. Go right into Room2
d
Enter the Room8
```

```
What do you want to do ?
A.Move
B.Using item
C.Search the room
D.Show information
E.Save file
a
U. Go up into Room2
D. Go down into Room0
R. Go right into Room4
r
Enter the Room4
There is a <Monster> is in this room.
What do you want to do?
a.Eenter the monster territory ?
b.Retreat
```

Fighting

1. 流程圖



2. Code 與細節

```
void Monster::triggerEvent(Object* play){
    Player *playptr=static_cast<Player*>(play);
    char temp;
    cout<<endl;
    cout<<"What do you want to do"<<endl;
    cout<<"a.Fight with "<<this->getName()<<endl;
    cout<<"b.check the information with the monster"<<endl;
    cout<<"c.Retreat"<<endl;
    cin>>temp;

    if(temp=='a'||temp=='A')
    {
        cout << "System: Start Fighting!!" <<endl ;
        while(!playptr->checkIsDead()&&!this->checkIsDead()){
            int dam=playptr->chooseskill();
            cout<<endl ;
            this -> takeDamage(dam) ;
            cout << "You -> Monster  Damage:" << dam <<endl ;
            cout << "Your health : "<< ((playptr->getCurrentHealth()>=0)?playptr->getCurrentHealth():0) << " / " << "Monster health : " << ((this->getCurrentHealth()>=0)?this->getCurrentHealth():0) <<endl ;
            if(this->checkIsDead()){
                cout << "You beat " << this->getName() << ", Congraduation !" <<endl ;
                cout <<"Now you can get a part of the monster ability!"<<endl;
                char c;
                cout<<"a.health: +"<<this->getMaxHealth()/2<<"\nb.attack : +"<<this->getAttack()/2<<"\nc.defense : +"<<this->getDefense()/2<<endl;
                cin>>c;
                switch (c)
                {
                    case 'a':
                        playptr->increaseStates(this->getMaxHealth()/2,this->getMaxHealth()/2,0,0);
                        break;
                    case 'b':
                        playptr->increaseStates(0,0,this->getAttack()/2,0);
                        break;
                    case 'c':
                        playptr->increaseStates(0,0,0,this->getDefense()/2);
                        break;
                }
                Item treasure=createTreasure();
                cout<<"You get "<<treasure.getName()<<"<"<<treasure.getType()<<">"<<endl;
                playptr->addItem(treasure);
                break;
            }
            playptr->takeDamage(this->getAttack());
            cout << "Monster -> You  Damage:" << this->getAttack() <<endl ;
            cout << "Your health : "<< ((playptr->getCurrentHealth()>=0)?playptr->getCurrentHealth():0) << "/" << "Monster health : " << ((this->getCurrentHealth()>=0)?this->getCurrentHealth():0) <<endl ;
            if(playptr->checkIsDead()){
                cout << "You are killed by the monster ! " <<endl ;
                break;
            }
            if(!retreatornot()){
                playptr->setCurrentRoom(playptr->getPreviousRoom());
                cout<<"Back to Room"<<playptr->getCurrentRoom()->getIndex()<<endl;
                break;
            }
        }
    }
```

```
    else if(temp=='b'||temp=='B'){
        this->access();
        triggerEvent(playptr);
    }
    else if(temp=='c'||temp=='C'){
        playptr->setCurrentRoom(playptr->getPreviousRoom());
        cout<<"Back to Room "<<playptr->getCurrentRoom()->getIndex()<<endl;
        return;
    }
    else{
        cout<<"Invalid input"<<endl;
        triggerEvent(playptr);
    }

};
```

- 戰鬥系統的實在是在 monster 的 tiggerEvent 中
- 戰鬥系統可分為

  I.Fighting,II.CheckMonsterStatus,III.retreat 三部分

  I. Fighting

i. Skill system

  - 戰鬥可以選擇不同攻擊方式 a.normal attack,b.usingitem,c.ultimateskill(只在有 treasure 時才會顯示)
  - ultimateskill 會隨機消耗一個背包中的 treasure 來打出 normal 的 3 倍傷害(實作可看下方 Code)
  - 輸入錯誤會再次執行 chooseskill()
  - Usingitem 則是定義在 player.h 中

```
int Player::chooseskill()
{
    cout<<"a.normal atack("<<this->getAttack()<<")"<<endl;
    cout<<"b.using item"<<endl;
    if(countitem()>0){
        cout<<"c.Ultimate abilitiy("<<3*this->getAttack()<<"/one treasure)"<<endl;
    }

    char c;
    cin>>c;
    if(c=='a'||c=='A'){
        return this->getAttack();
    }
    else if(c=='b'||c=='B'){
        usingItem();
        return 0;
    }
    else if(c=='c'||c=='C'){
        int pos=findpos2(inventory,"treasure");
        popitem(pos);
        return 3*this->getAttack();
    }
    else{
        cout<<"System : invalid input"<<endl;
        return chooseskill();
    }
}
```

- 增強系統
- Monster 被擊敗後，玩家可以獲得其 50%的屬性
- 要獲得 1.health,2.attack,3.defense 由玩家選擇
- 然後 random 出寶物，並將其添加至玩家得背包中

```
Item Monster:: createTreasure()
{
    int tt=rand()%3;
    if(tt==0){
        int health;
        health=80+rand()%100;
        return Item("pill<super>","recovery",health,0,0);
    }
    else if(tt==1){
        int att,def,health;
        att=rand()%50;
        def=rand()%50;
        health=rand()%100;
        return Item("dragon_sword<supre>","weapon",health,att,def);
    }
    else if(tt==2){
        int att,def,health;
        att=rand()%20;
        def=rand()%20;
        health=rand()%10;
        return Item("dragon_blood","treasure",health,att,def);
    }
}
```

ii. UsingItem

```
void Player::usingItem()
{
    if(inventory.size()==0){
            cout<<"System : you do not have any Item"<<endl;
            return;
        }
        this->access();
        cout<<"Input the index of the Item that you want to use(1 ~ "
            <<inventory.size()<<")\nBack to previous(-1)"<<endl;
        int index;
        cin>>index;
        if(index==-1)
            return;
        index--;
        if(inventory[index].getType()=="weapon")
            wear_gear(index);
        else if(inventory[index].getType()=="recovery")
            using_recovery(index);
        else if(inventory[index].getType()=="treasure")
            cout<<"you can't use Item directly"<<endl;
}
```

- access()是顯示背包所有的 Item
- 玩家要輸入想使用的 Item
- 不同 Item 的 type 會對應不同動作

A. Weapon：穿上去(increaseState)

```
void Player::wear_gear(int index)
{
    this->increaseStates(inventory[index].getHealth(),inventory[index].getHealth(),inventory[index].getAttack(),inventory[index].getDefense());
    popitem(index);
}
```

B. Recovery：回復血量，不能超過最大值

```
void Player::using_recovery(int index)
{
    int diff=(getMaxHealth()-getCurrentHealth());
    this->increaseStates(0,((diff>inventory[index].getHealth())?inventory[index].getHealth():diff),0,0);
    popitem(index);
}
```

C. Treasure：不能直接使用

II. CheckMonsterStatus

✓ 由 access()輸出怪物當前資訊

III. Retreat

✓ 將玩家移動回上一個 Room

3. 結果

```
What do you want to do
a.Fight with Liwei
b.check the information with the monster
c.Retreat
a
System: Start Fightting!!
a.normal atack(30)
b.using item
```

```
System: Start Fightting!!
a.normal atack(30)
b.using item
a

You -> Monster   Damage:30
Your health : 100 / Monster health : 150
Monster -> You   Damage:30
Your health : 70/Monster health : 150
```

```
Now you can get a part of the monster ability!
a.health: +25
b.attack : +30
c.defense : +50
a
       MaxHealth:100--->125
   CurrentHealth:30--->55
You get dragon_sword<supre><weapon>
```

```
System: Start Fightting!!
a.normal atack(30)
b.using item
c.Ultimate abilitiy(90/one treasure)
b
---------<Backpack information>----------
1.
-------<treasure>------
name : Dragon_heart
health :0
attack :30
defense : 0
2.
-------<weapon>------
name : dragon_sword<supre>
health :45
attack :42
defense : 7
Input the index of the Item that you want to use(1 ~ 2)
Back to previous(-1)
```

```
What do you want to do
a.Fight with Liwei
b.check the information with the monster
c.Retreat
b
--------<Monster>--------
          name :      Liwei
     maxHealth :        180
 currentHealth :        180
        attack :         30
       defense :        100
```

```
What do you want to do
a.Fight with Liwei
b.check the information with the monster
c.Retreat
c
Back to Room 2
What do you want to do ?
```

```
2.
-------<weapon>------
name : dragon_sword<supre>
health :45
attack :42
defense : 7
Input the index of the Item that you want to use(1 ~ 2)
Back to previous(-1)
2
       MaxHealth:125--->170
   CurrentHealth:125--->170
          Attack:30--->72
        Denfense:30--->37
```

Show Information

1. Code

```
void Dungeon::showinformation(){
    cout<<"A.show the backpack"<<endl
        <<"B.the state of the task"<<endl
        <<"C.show the current room"<<endl
        <<"D.show current status"<<endl;
    char tmp;
    cin>>tmp;
    switch(tmp)
    {
        case 'a' : case 'A' : player->access(); break ;
        case 'b' : case 'B' : showTasklist();break;
        case 'c' : case 'C' : cout<<"------<Position>-------"
                                <<endl<< "You are in the Room"
                                << player->getCurrentRoom()->getIndex() << endl ;break;
        case 'd' : case 'D' :player->showstatus();
    }

}
```

2. 說明

- Show Information 可以分為四個部分

I. Show backpack

```
void Player::showinventory()
{
    cout<<"---------<Backpack information>----------"<<endl;
    if(inventory.empty())cout<<"The back pack is empty ..."<<endl;
    for(int i=0;i<(int)inventory.size();i++)
    {
        cout<<i+1<<"."<<endl;
        inventory[i].showitem();
    }
}
```

II. Show the task

```
void Dungeon::showTasklist(){
    cout<<"------<Task1>-------"<<endl;
    cout<<"Get to the exit : "<<((player->getCurrentRoom()==exitroom)?"Yes":"No")<<endl;
    cout<<"------<Task2>-------"<<endl;
    cout<<"Collect Item :"<<player->countitem()<<"/"<<numberof_collected_item<<endl;
    cout<<"------<Task3>-------"<<endl;
    cout<<"Killed Monster :"<<current_killed_monster<<"/"<<numberof_killed_monster<<endl;

}
```

III. Show current room(在 1.即可看到實作)

IV. Show status

```
void Player::showstatus()
{
    cout<<"--------<player>--------"<<endl;
    cout<<setw(17)<<"name : "<<setw(10)<<getName()<<endl;
    cout<<setw(17)<<"maxHealth : "<<setw(10)<<getMaxHealth()<<endl;
    cout<<setw(17)<<"currentHealth : "<<setw(10)<<getCurrentHealth()<<endl;
    cout<<setw(17)<<"attack : "<<setw(10)<<getAttack()<<endl;
    cout<<setw(17)<<"defense : "<<setw(10)<<getDefense()<<endl;
}
```

3. 結果

```
What do you want to do ?
A.Move
B.Using item
C.Search the room
D.Show information
E.Save file
d
A.show the backpack
B.the state of the task
C.show the current room
D.show current status
a
---------<Backpack information>----------
The back pack is empty ...
What do you want to do ?
```

```
A.show the backpack
B.the state of the task
C.show the current room
D.show current status
b
------<Task1>-------
Get to the exit : No
------<Task2>-------
Collect Item :0/1
------<Task3>-------
Killed Monster :0/1
```

```
A.show the backpack
B.the state of the task
C.show the current room
D.show current status
c
------<Position>-------
You are in the Room0
```

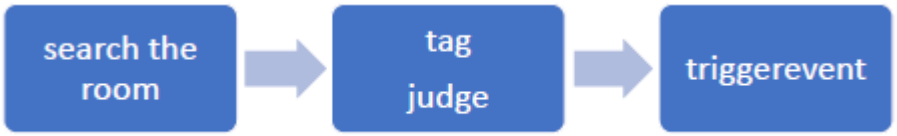
```
A.show the backpack
B.the state of the task
C.show the current room
D.show current status
d
--------<player>--------
           name :        aaa
      maxHealth :        100
  currentHealth :        100
         attack :         30
        defense :         30
```

Search the Room

1. 說明

- Search the room 是 choose action 的其中一個功能，利用 trigger()來尋找 Room 內部的 Object，再由 tagjudge 來輸出不同 type 物件的對應字串，再開始進行互動

- 流程圖



2. 實作

Trigger function

```
void Dungeon::trigger(Room *tmpptr){
    if(tmpptr->getObjects().empty()){
        cout<<"There is nothing in the Room"<<endl;
        return;
    }
    cout << "There is a <"<<tmpptr->objects[0]->getTag() << "> is in this room." << endl;
    char tmp ;
    tmp = tmpptr->tagjudge(tmpptr->objects[0]);
    if(tmp=='a'){
        GameCharacter *gamecharptr=static_cast<GameCharacter*>(tmpptr->objects[0]);
        gamecharptr->triggerEvent(player);
        if(gamecharptr->checkIsDead()){
            if(gamecharptr->getTag()=="Monster")
                current_killed_monster++;
            player->getCurrentRoom()->popObject(gamecharptr);
        }
    };
    if(tmp=='b'){
        this->player->setCurrentRoom(player->getPreviousRoom());
        cout<<"back to Room"<<player->getCurrentRoom()->getIndex()<<endl;
    };
    if(tmp=='B'){
            return;
    }
}
```

Detail：

I. 我們無法確定 room 中的 ojbect*所指向的物件是何種

型態，但可以確定房間內可能存在的三種物件 NPC, Monster, Chest 都是 gamecharacter(繼承)，所以在操作時我將 room 中的 object* downcast 成 GameCharacter，令其可以使用定義在 class GameCharacter 中的功能

II. 函數一開始，先檢查是否有 object，如果沒有，則終止函數

III. 經過 tagjudge(),若選擇與 object 互動則將 player 傳入該 object 的 triggerevent 中。互動結束後，若擊殺怪獸則更新任務狀態

IV. 因為在遇到 monster 時只能回到 previousroom，所以使用 if 來區分不與 object 互動的 case

Tagjudge function

```
char Room::tagjudge(Object* objptr){
    char chr;
    cout << "What do you want to do?\n" ;
    if(objptr->getTag()=="Monster"){
        cout << "a.Enter the monster territory ?" <<endl ;
        cout<<"b.Retreat" <<endl ;
    }
    else if(objptr->getTag()=="NPC"){
        cout << "Interact with the NPC ? " <<endl ;
        cout<< "a.Yes !"<<endl;
        cout<<"b.No,negelect the npc"<<endl;
    }
    else if(objptr->getTag()=="CHEST"){
        cout << "a.Open the chest" <<endl ;
        cout << "b.Abandom the chest"<<endl;
    }
    cin>>chr;
    while(chr!='a'&&chr!='b'){
        cout<<"Invalid input"<<endl;
        cout<<"please type in again(a/b)"<<endl;
        cin>>chr;
    }
    if(chr=='a')
        return chr;
    else if(chr=='b'&&objptr->getTag()=="Monster")
        return 'b';
    else if(chr=='b'&&(objptr->getTag()=="NPC"||objptr->getTag()=="CHEST"))
        return 'B';
}
```

Detail：

I. 根據不同的 tag 輸出不同的提示字串，接著選擇是否要互動(a.要, b.不要)

II. 因為要滿足前述，對於不與 monster 互動的 special case，故以回傳不同的大小寫，來達成目的

TriggerEvent

- TriggerEvent 是以 virtual function 方式實作，利用 dynamic link 的性質，使被 upcast 的物件可以存取到正確的函數。
- 以下分別對 NPC,Monster, Item, Chest 各個 class 中 overwrite 的 TriggerEvent 做說明。
- 因為與物件互動的均為 player 故在呼叫時，大多傳入 player 的指標

I. NPC

1. Code

```
void NPC::triggerEvent(Object* play){
    showstatus();
    Player *playptr=static_cast<Player*>(play);
    if(this->getCommodity().empty() )
    {
        cout<<"Sorry,all the commdity have been sold out"<<endl;
        return;
    }

    cout << "Do you want trade your health with my commodity?(y/n)" <<endl
         << "cost 10 current to get an obj"<<endl;
    char q ;
    cin >> q ;
    if(q=='y'){
        if(playptr->getCurrentHealth()<=10){
            cout<<"Sorry do not have enough health to trade with me"<<endl;
            return ;
        }
        showinventory();
        int index ;
        cout << "which do you want to trade?" << endl ;
        cin >> index ;
        cout<<"NICE CHOOSE!"<<endl;
        playptr->increaseStates(0,-10,0,0);
        commodity[index-1].triggerEvent(playptr);
        commodity.erase(commodity.begin()+(index-1));
    }
}
```

2. 說明

   i. 先檢查 NPC 內是否還有 Item 可以交易，true 則 continue，false 則 return

   ii. 接著檢查，是否有足夠的生命值可以交易，true 則 continue，false 則 return

   iii. 由 showinventory()印出所有的 Item,讓玩家選擇

   iv. 接著呼叫 Item 的 triggerEvent，並將該 Item 從 vector 中移除

3. 結果

```
What do you want to do ?
A.Move
B.Using item
C.Search the room
D.Show information
E.Save file
c
There is a <NPC> is in this room.
What do you want to do?
Interact with the NPC ?
a.Yes !
b.No,negelect the npc
```

```
Welcome
Do you want trade your health with my commodity?(y/n)
cost 10 current to get an obj
y
1.
-------<treasure>------
name : Dragon_heart
health :0
attack :30
defense : 0
2.
-------<weapon>------
name : knife<normal>
health :5
attack :20
defense : 5
```

```
which do you want to trade?
1
NICE CHOOSE!
   CurrentHealth:100--->90
adding Dragon_heart into the backpack
```

```
Welcome
Do you want trade your health with my commodity?(y/n)
cost 10 current to get an obj
y
1.
-------<weapon>------
name : knife<normal>
health :5
attack :20
defense : 5
2.
-------<weapon>------
name : suit<normal>
health :0
attack :0
defense : 20
3.
-------<treasure>------
name : Dragon_blood
health :100
attack :0
defense : 0
which do you want to trade?
```

II. Chest

1. Code

```
void Chest::triggerEvent(Object * play){
    showstatus();
    Player *playerptr=static_cast<Player*>(play);
    cout<<"Do you want to open the chest (y/n)?"<<endl;
    char temp;
    cin>>temp;
    if(temp=='y'||temp=='Y'){
        if(getisbadchest()){
            cout<<"Shit ! There is no treasure inside"<<endl;
            cout<<"Warning ! You got poison ! "<<endl;
            playerptr->increaseStates(0,-10,0,0);
        }
        else{
            cout<<"Wow ! Found "<<treasure[0].getName()<<"<"<<treasure[0].getType()<<">"<<endl;
            treasure[0].triggerEvent(playerptr);
        }
        this->takeDamage(100);
    }
}
```

2. 說明

   i. 寶箱設計有 good、bad 兩種

   ii. 進入 function 後由玩家決定是否開啟，如果開啟則繼續執行，不開則 return

   iii. 如果是 good chest，打開會獲得 treasure，反之則會受到傷害

   iv. 結束後將寶箱生命歸零，讓其在回到 trigger()函數後，能夠把該物件從 room 中移除

3. 結果

```
What do you want to do ?
A.Move
B.Using item
C.Search the room
D.Show information
E.Save file
c
There is a <CHEST> is in this room.
What do you want to do?
a.Open the chest
b.Abandom the chest
```

```
Open to find some suprise
Do you want to open the chest (y/n)?
y
Shit ! There is no treasure inside
Warning ! You got poison !
   CurrentHealth:85--->75
```

```
There is a <CHEST> is in this room.
What do you want to do?
a.Open the chest
b.Abandom the chest
a

Opne to find some suprise
Do you want to open the chest (y/n)?
y
Wow ! Found Dragon_eye<treasure>
adding Dragon_eye into the backpack
```

III. Item

1. Code

```
void Item::triggerEvent(Object* player){
    Player *playerptr=static_cast<Player*>(player);
    char c;

    if(type=="weapon"){
        cout<<"Wear "<<this->getName()<<"(a) ,or put it into the backpack(b)"<<endl;
        cin>>c;
        if(c=='a'||c=='A'){
            playerptr->increaseStates(getHealth(),getHealth(),getAttack(),getDefense());
        }
        else{
            playerptr->addItem(*this);
        }
    }
    else if(type=="recovery"){
        cout<<"Use "<<this->getName()<<"(a) ,or put it into the backpack(b)"<<endl;
        cin>>c;
        if(c=='a'||c=='A'){
            int diff=(playerptr->getMaxHealth()-playerptr->getCurrentHealth());
            playerptr->increaseStates(0,((diff>this->getHealth())?this->getHealth():diff),0,0);
        }
        else{
            playerptr->addItem(*this);
        }
    }else if(type=="treasure"){
        cout<<"adding "<<this->getName()<<" into the backpack"<<endl;
        playerptr->addItem(*this);
    }
}
```

2. 說明

i. 在前面提到的 NPC 和 Chest 中都有呼叫到 Item 的 triggerEvent，此函數主要的目的是要對不同的 type的 Item 做不同的處理

ii. Weapon:直接根據裝備數據增加狀態或放入背包

iii. Recovery:回復生命值(不會超過 MaxHealth)或放入背包

iv. Treasure:直接添加入背包

3. 結果

```
Use pill<super>(a) ,or put it into the backpack(b)
a
  CurrentHealth:209--->249
```

IV. Monster (見 Ffighting)

Task System

1. 說明

I. 在遊戲中，需分別完成 1.擊殺指定數目怪物(task_kill_monster)，2.蒐集到指定數目的 treasure (task_collectitem)，3.找到出口(task_findexit)，三種任務才能破關

II. 在遊戲間可以藉由 showinformation()函數來檢視當前任務狀態(見 ShowInformaiton 說明)

III. task_findexit:檢查 player 的 currentroom 是否為 exitroom

IV. task_kill_monster：檢查擊殺 monster 數目是否超過指定數目

V. task_collectitem：檢查 treasure 搜集的數目是否超過

VI. 當完成全部三個任務即為勝利

2. Code

```
bool Dungeon::task_findexit(){
    static int idx=0;
    if(player->getCurrentRoom()==exitroom){
        return true;
    }
    else{
         return false;
    }

}

bool Dungeon::task_kill_monster(){
    if(this->getCurrentkilledMonster()>=this->getNumberofkilledMonster()){
         return true;
    }
    else{
        return false;
    }
}

bool Dungeon::task_collectitem(){
    if(player->countitem()>=numberof_collected_item){
        return true;
    }
    else{
        return false;
    }

}
```

Backup System

1. Code

```
void backup(Dungeon &dun)
{
    ofstream outfile("backupfile.txt");
    if(outfile.bad()){
         cout<<"Something Wrong! Can't save the file"<<endl;
         return;
    }
    /*save player*/
    saveplayer(dun.getPlayer(),dun.getGameroom(),outfile);
    outfile<<dun.getExitroom()->getIndex()<<" "<<dun.getNumberofcollectItem()<<" "<<dun.getCurrentkilledMonster()<<" "<<dun.getNumberofkilledMonster()<<endl;
    cout<<"Finish back up"<<endl;
}
```

```
void saveplayer(Player *player,vector<Room> gameroom,ofstream &outfile)
{
        outfile<<gameroom.size()<<endl;
    for(int i=0;i<gameroom.size();i++)
    {
        outfile<<  i  <<" "<<gameroom[i].objects.size()<<endl;
        outfile<<((gameroom[i].getUpRoom()!=NULL)?gameroom[i].getUpRoom()->getIndex():-1)<<" ";
        outfile<<((gameroom[i].getDownRoom()!=NULL)?gameroom[i].getDownRoom()->getIndex():-1)<<" ";
        outfile<<((gameroom[i].getLeftRoom()!=NULL)?gameroom[i].getLeftRoom()->getIndex():-1)<<" ";
        outfile<<((gameroom[i].getRightRoom()!=NULL)?gameroom[i].getRightRoom()->getIndex():-1)<<" ";
        outfile<<endl;
        for(int j=0;j<gameroom[i].objects.size();j++)
        {
            if(gameroom[i].objects.empty())
            {
                cout<<"break"<<endl;
                break;
            }

            string tagofobj=gameroom[i].objects[j]->getTag();

            if(tagofobj=="Monster")
            {
                Monster *monptr=static_cast<Monster *>(gameroom[i].objects[j]);
                outfile<<monptr->getTag()<<" "<<monptr->getName()<<" "<<monptr->getMaxHealth()<<" "<<monptr->getCurrentHealth()<<" "<<monptr->getAttack()<<" "<<monptr->getDefense()<<" "<<endl;
            }
            else if(tagofobj=="NPC")
            {
                NPC *npcptr=static_cast<NPC*>(gameroom[i].objects[j]);
                outfile<<npcptr->getTag()<<" "<<npcptr->getName()<<" "<<npcptr->getScript()<<" "<<npcptr->getCommodity().size()<<endl;

                vector<Item> outvec=npcptr->getCommodity();

                for(int i=0;i<outvec.size();i++){
                    outfile<<outvec[i].getName()<<" "<<outvec[i].getType()<<" "<<outvec[i].getHealth()<<" "<<outvec[i].getAttack()<<" "<<outvec[i].getDefense()<<endl;
                }

            }
            else if(tagofobj=="CHEST")
            {
                Chest *chestptr=static_cast<Chest*>(gameroom[i].objects[j]);
                outfile<<chestptr->getTag()<<" "<<chestptr->getName()<<" "
                       <<chestptr->getScript()<<'||'<<endl
                       <<((chestptr->getisbadchest())?0:1)<<" "<<chestptr->getTreasure().size()<<endl;


                vector<Item> outvec=chestptr->getTreasure();

                for(int i=0;i<outvec.size();i++){
                    outfile<<outvec[i].getName()<<" "<<outvec[i].getType()<<" "<<outvec[i].getHealth()<<" "<<outvec[i].getAttack()<<" "<<outvec[i].getDefense()<<endl;
                }

            }
        }
    }
    outfile<<player->getName()<<" "<<player->getMaxHealth()<<" "<<player->getCurrentHealth()<<" "
           <<player->getAttack()<<" "<<player->getDefense()<<" "
           <<((player->getCurrentRoom()!=NULL)?player->getCurrentRoom()->getIndex():-1)<<" "
           <<((player->getPreviousRoom()!=NULL)?player->getPreviousRoom()->getIndex():-1)<<" "<<player->getInventory().size()<<endl;

    vector<Item> outvec=player->getInventory();

    for(int i=0;i<outvec.size();i++){
        outfile<<outvec[i].getName()<<" "<<outvec[i].getType()<<" "<<outvec[i].getHealth()<<" "<<outvec[i].getAttack()<<" "<<outvec[i].getDefense()<<endl;
    }
}
```

2. 說明

I. Backup 及 saveplayer 是實作在 tool.cpp 中的功能函式

II. 因為 Dungeon 內許多的物件都是以 vector 形式儲存，所以在存檔時，主要是以迴圈(次數由 vector 的 size 決定)來實作

III. 因為考量資料讀取,和在建構環境時的先後順序，儲存的順序為:房間房間內物件 player 資料。因為在建立 Dungeon 環境時是先建立 room，藉由這樣的儲存順序便可以方便讀取，省去將 txt 檔案內資料暫存再使用的麻煩。

IV. 根據不同的物件型態，資料也有所不同，故在回圈內部會先判斷該 object 的 tag 再做出相對應的動作

V. 而在指標的處理，因為方便起見，若指標指向 NULL 則存為-1

```
10
0 1
1 -1 -1 -1
NPC Uncle_Rogers Welcome 4
Dragon_heart treasure 0 30 0
knife<normal> weapon 5 20 5
suit<normal>  weapon 0 0 20
Dragon_blood treasure 100 0 0
1 1
2 0 -1 4
NPC Doctor_Chopper Good_morning! 3
pill<normal> recovery 30 0 0
pill<normal> recovery 30 0 0
pill<super> recovery 100 0 0
2 1
3 1 7 -1
CHEST Old_chest
Open to find some suprise!
0 0
3 1
6 2 -1 -1
Monster Liwei 180 180 30 100
4 1
-1 -1 1 5
Monster BenTen 50 50 60 100
5 0
-1 -1 4 -1
6 1
-1 3 -1 -1
CHEST Ancient_chest
Opne to find some suprise!
1 1
Dragon_gem treasure 100 100 100
7 1
-1 8 -1 2
NPC Doctor_Giorno Good_morning! 3
pill<medium> recovery 40 0 0
pill<medium> recovery 40 0 0
pill<super> recovery 100 0 0
```

Reading System

1. Code

```
Player* reading(Player *playptr,vector<Room> &room,int &exit,int &numberofcollect,int &currentkilled,int &numberofkilled,char *filename,ifstream &infile)
{
    int numberofroom;
    infile>>numberofroom;
    vector<Room> room2(numberofroom);
    for(int i=0;i<numberofroom;i++)
    {
        Room r;
        room.push_back(r);
    }
    for(int i=0;i<room.size();i++)
    {
        int numberofobj;
        infile>>room[i].index;
        infile>>numberofobj;;
        int u,d,l,r;
        infile >> u >> d >> l >> r;
        if(u>=0)
            room[i].upRoom = &room[u];
        else
            room[i].upRoom = NULL;
        if(d>=0)
            room[i].downRoom = &room[d];
        else
            room[i].downRoom = NULL;
        if(l>=0)
            room[i].leftRoom = &room[l];
        else
            room[i].leftRoom = NULL;
        if(r>=0)
            room[i].rightRoom = &room[r];
        else
            room[i].rightRoom = NULL;

        if(numberofobj==0){
            vector<Object *>obj(0);
            room[i].setObjects(obj);
        }
        else{
            room[i].setObjects(setroomobject(infile,numberofobj));
        }
    }

    /*player*/
    string name;
    int maxh,curh,att,def,inventory_num,current,previous;
    infile>>name>>maxh>>curh>>att>>def>>current>>previous>>inventory_num;
    playptr=new Player(name,maxh,att,def);
    playptr->setCurrentHealth(curh);
    if(current!=-1){
        playptr->setCurrentRoom(&room[current]);
    }
    else{
        playptr->setCurrentRoom(NULL);
    }
    if(previous!=-1){
        playptr->setPreviousRoom(&room[previous]);
    }
    else{
        playptr->setPreviousRoom(NULL);
    }

    vector <Item> inventory;
    for(int i=0;i<inventory_num;i++)
    {
        string itemname,type;
        int att,def,health;
        infile>>itemname>>type>>health>>att>>def;
        Item it(itemname,type,health,att,def);
        inventory.push_back(it);
    }
    playptr->setInventory(inventory);
    infile>>exit>>numberofcollect>>currentkilled>>numberofkilled;
    return playptr;
}
```

```
vector<Object *> setroomobject(ifstream &infile,int numberofobj)
{
    vector<Object *>objectlist;
    string objtag;
    string name;
    for(int i=0;i<numberofobj;i++)
    {
        infile>>objtag>>name;
        if(objtag=="NPC")
        {
            string script;
            int numberofinventory;
            vector <Item> commoditylist;
            infile>>script;
            infile>>numberofinventory;
            for(int i=0;i<numberofinventory;i++)
            {
                string itemname,type;
                int att,def,health;
                infile>>itemname>>type>>health>>att>>def;
                Item it(itemname,type,health,att,def);
                commoditylist.push_back(it);
            }
            NPC *npcptr=new NPC(name,script,commoditylist);
            objectlist.push_back(npcptr);
        }
        if(objtag=="CHEST")
        {
            string script;
            int numberofinventory,goodornot;
            bool tag;
            getline(infile,script,'|');
            infile>>goodornot;
            if(goodornot==0)tag=true;
            else tag=false;
            infile>>numberofinventory;
            vector <Item> commoditylist;
            for(int i=0;i<numberofinventory;i++)
            {
                string itemname,type;
                int att,def,health;
                infile>>itemname>>type>>health>>att>>def;
                Item it(itemname,type,health,att,def);
                commoditylist.push_back(it);
            }
            Chest *chestptr=new Chest(name,script,commoditylist,tag);
            objectlist.push_back(chestptr);
        }
        if(objtag=="Monster")
        {
            int maxh,curh,att,def;
            infile>>maxh>>curh>>att>>def;
            Monster *monsptr=new Monster(name,maxh,att,def);
            monsptr->setCurrentHealth(curh);
            objectlist.push_back(monsptr);
        }
    }
    return objectlist;
}
```

2. 說明

   I. readingsystem 主要由 reading()及 setobject()兩個函數，來設定 Dungoen 內部的 data

   II. 讀取時則是根據設計好的 txt 檔的格式，藉由 ifstream 會自動抓取空白或換行的直接讀取性質，便可達到目的

III. 因為傳入的 room vector 及其他 Dungeon 內的 data 是 call by reference，因此可以直接做設定

IV. 至於 player 的部分則是重新宣告一個 Player 指標 new 出一個新的記憶體空間，資料設定好再回傳記憶體位址

V. 讀取順序是先將房間的資訊讀進來，建好房間之後，再將 room 內用來存取屋內物件的 vector 以 reference 傳給 setObject()函數來設定

Disussion

✓ 遇到的問題

i. Virtual function 的運用：寫完才發現，showinventory()和 showstatus()如果定義成 member function，或許會比 virtual function 更有效率

ii. 用 txt 檔生成整個 Dungeon 比較方便，但可能會被玩家自行修改地圖，或許用自動生成會更好

iii. 本來打算導入升級系統，但是後來發現不如直接吸收怪物的數值，除了有同樣功能，也更容易實作

Conclusion

在寫這份 project 的過程中，我學到最多的就是如何 downcast 與 upcast，再來就是各種繼承關係的運用，雖然寫的過程中充滿各

種奇怪的 bug，但是寫完整份 Dungeon 真的很有成就感，也感謝助教與老師的教導，讓我對 OOP 更加認識。