
DUNGEON

目錄

Implementation detiled

1. Object: 為 GameCharacter 和 Item 兩個 class 的 base class，包含的 attribute 有兩個，分別為 Name(負責記錄名字) 和 tag(負責記錄類型)，而包含的 function 除了基本的 constructor 跟 attribute 各別的 setter 和 getter 外，還有一個名為 triggerEvent(Object* object) 的 pure virtual function，在後續不同的 derived class 中，會有不同的執行功能。

2. Room: 包含的 attribute 有七個，其中 upRoom、downRoom、leftRoom 和 rightRoom 負責記錄某間房間上下左右的房間為何，index 則是負責記錄這間房間的編號，而 isExit 則是負責記錄這間房間是否為最後一間，最後則是 objects，這個 vector 主要負責記錄這間房間有沒有其他事物，像是 NPC、Monster 和 Item 之類的。至於 funtion 的部分，除了基本的 constructor 外，主要都是各個 attribute 的 getter 和 setter，其中 setObjects 這個 function 的運作原理為 push_back 一個 Item 到 objects 中，而非直接將 objects 這個 vector 等於另外一個 vector。

3. Item: 為 Object class 的 derived class，包含了 6 個 attribute，其中 attack、defense、health(目前生命值)和 maxhealth(生命值上限)主要負責記錄 item 能夠提升玩家能力值的數值，price 負責記錄 item 的價錢，而 isTake 則是負責記錄 item 是否已被玩家拿取，避免發生重複拿取的狀況。至於 funtion 的部分，除了基本的 constructor 跟各個 attribute 的 getter 和 setter，還有一個 triggerEvent(Object* object)的 function，繼承了 Object class 內的 pure virtual function，在此處的作用為將玩家的能力值能依照 attribute 所記錄的各個數值來提升，方法為將 Player 型態的變數作為 argument 引入，並使用 dynamic_cast 將其從 Object 的型態轉成 Player 的型態，接著再利用 Players 能使用的 getter 和 setter 改變 Player 的能力值。

4. GameCharacter: 為 Object class 的 derived class，包含的 attribute 有七個，利用 maxHealth(生命值上限)、currentHealth(現有生命值)、attack(攻擊力數值)、defense(防禦力數值)、currentCD(技能冷卻回合數)、occupation(職業名稱)和 skill(技能名稱)來記錄角色的基本資訊。而 funtion 的部分，除了基本的 constructor 跟各個 attribute 的 getter 和 setter，還有一個 checkIsDead()的 funtion，執行方式為當血量<=0 時，就 return true，反之則回傳 false。

5. NPC: 為 GameCharacter class 的 derived class，包含的 attribute 有兩

個,分別為script(紀錄NPC的台詞)和commodity(紀錄NPC所擁有的Item)。而 funtion 的部分，除了基本的 constructor 跟各個 attribute 的 getter 和 setter，還有一個 listCommodity()的 function，透過 for 迴圈，將 commodity 這個 vector 內存的 Item 一個個輸出。另外還有一個 triggerEvent(Object* object)的 function,繼承了 Object class 內的 pure virtual function,在此處的作用為設計一個交易系統,方法為首先將 Player 型態的變數作為 argument 引入，並使用 dynamic_cast 將其從 Object 的型態轉成 Player 的型態,接著使用 listCommidity 這個 function 將 NPC 這個角色所擁有的 item 列出,利用 cin 讓玩家輸入想購買的商品,當玩家的錢足夠時，利用 Player 的 addItem(Item* item) funtion 來將購買的商品放入 backpack 中，然後再利用 Player 能使用的 function，來設定 Player 剩餘的錢，和提升過後的能力值。

6. Monster: 為 GameCharacter class 的 derived class,包含的 attribute 有兩個,分別為 room(紀錄 Monster 所在的房間)和 money(打倒怪獸所能獲得的金錢)。而 funtion 的部分，除了基本的 constructor 跟各個 attribute 的 getter 和 setter，還有一個 combatSystem(Player* player)的 function，作法為首先判斷 Player 的 CD，如果為 0，則詢問是否使用技能，如果選擇使用技能，則根據 occupation 的不同，透過 Player 的 getter 和 setter 使能力值得到不同方式的加成，進行攻擊;如果 CD 不為 0 或選擇不使用技能，則使用普通攻擊進行攻擊，傷害的計算方法為看玩家的攻擊力減去怪獸的防禦力，如果大於 0，則直接將怪獸血量扣除相減的結果，如果小於等於 0，怪獸的血量則扣除 1，接著判斷怪獸是否死亡，如果死亡，則對戰結束，並顯示怪獸死亡的訊息；如果沒死，則輪到怪獸攻擊玩家，傷害的計算方式則和上述相同，看怪獸的攻擊力減去玩家的防禦力的結果，如果玩家死亡，則顯示玩家死亡的訊息，假使沒死亡，則是顯示玩家和怪獸各扣了多少血量，並剩餘多少血;最後還有一個 triggerEvent(Object* object)的 function，繼承了 Object class 內的 pure virtual function，在此處的作用為判斷玩家的血量是否足以戰鬥，如果大於 0，則 return true，如果小於等於 0，或是引入的 argument 指向 NULL，則 return false。

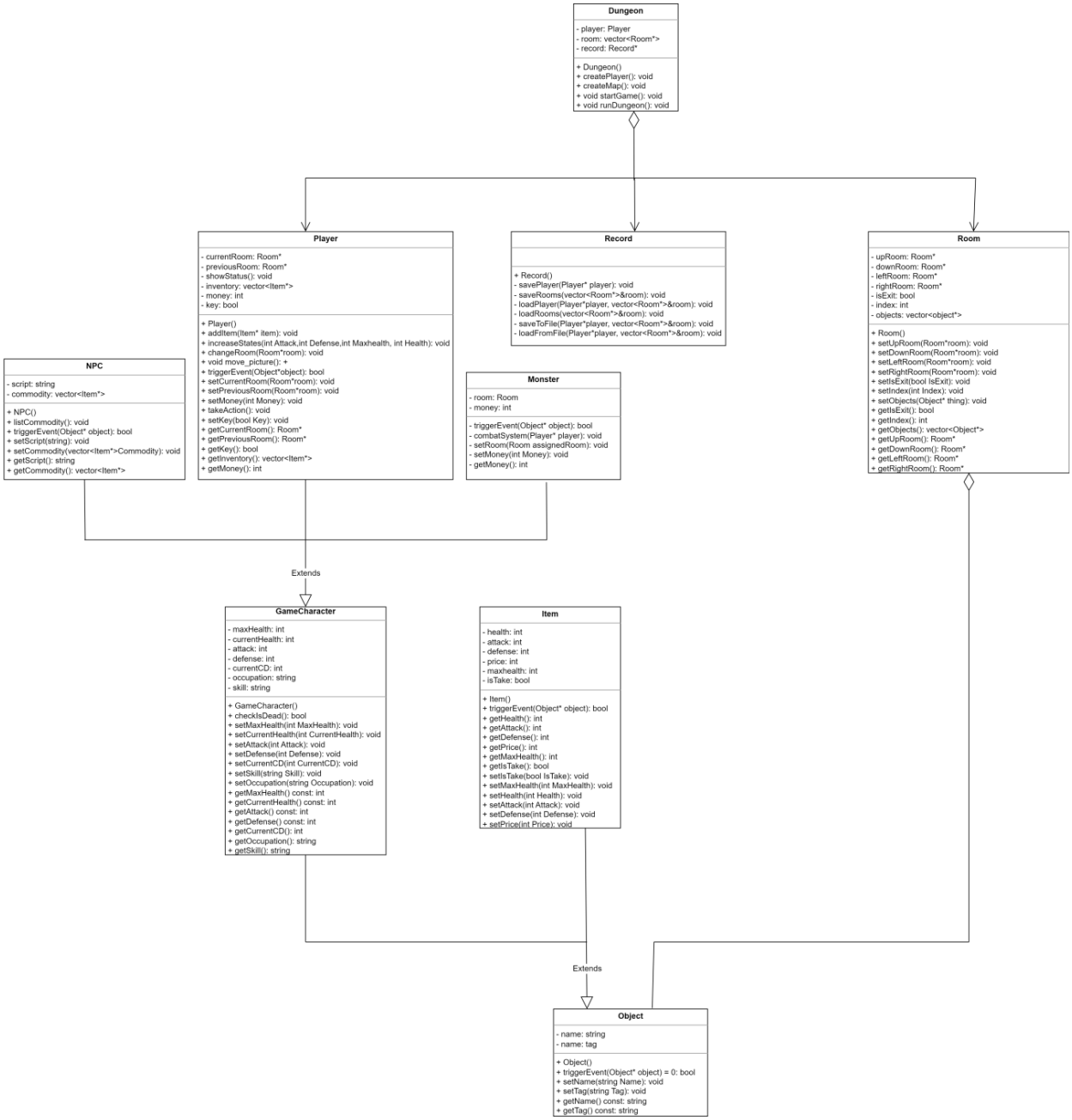
7. Player: 為 GameCharacter class 的 derived class，包含的 attribute 有 5 個，分別為 currentRoom(紀錄 Player 現在所在的房間)、previousRoom(紀錄 Player 上間所在的房間)、inventory(一個紀錄 Player 擁有什麼物品的 vector)、money(紀錄 Player 有多少錢)、key(紀錄 Player 是否有拿到鑰匙)。而 funtion 的部分，除了基本的 constructor 跟各個 attribute 的 getter 和 setter，還有幾個額外的 function，首先為 void increaseStates(int Attack,int Defense,int Maxhealth, int Health)，這個 function 主要是

透透 Player 內的 getter 和 setter 來增加 Player 的能力值。第二個則是 changeRoom(Room*room)的 function，這個 function 主要是透過 Player 內的 getter 和 setter 去更新玩家移動後 currentRoom 和 previousRoom 的值。而第三個為 move_picture()的 function，這個 function 是為了讓玩家移動時，能顯示出類似行人走路的簡易動畫，透過 10*16 的 int 陣列，把內部的值從 10 進位，轉成 2 進位，並用 for 迴圈使得每張圖皆能跑一遍，即可產生行人走路的簡易動畫。第四個為 takeAction()，主要負責輸出各種選項，讓玩家能夠選擇。第五個為 showStatus()，是一個 private 的 function，主要負責輸出 Player 目前的各個基本數值(像是 attack、skill、money……)。最後則是 triggerEvent(Object* object)的 function，繼承了 Object class 內的 pure virtual function，在此處的作用為呼叫 showStatus()這個 function，來輸出 Player 目前的各個基本數值。

8. Dungeon: 包含的 attribute 有三個，分別為 player(即玩家所代表的角色)、room(用來記錄整個地圖每間房間的 vector)、record(用來處理遊戲的儲存與載入)。Dungeon 除了基本的 constructor，還用到許多的 function，第一個是 createPlayer()，這個 function 主要是用來創建玩家的角色，首先會請玩家輸入名字，接著選 occupation，根據選擇的 occupation 來設定 player 的基本能力數值(像是 attack、skill、money……)，接著透過 player 的 triggerEvent(Object* object)來展現出創建後的角色的各個基本數值。第二個則是 createMap()，這個 function 主要是要將一間間的房間串聯起來，並設定好每間房間應有的 object，執行方法首先為創造 11 間房間，編號 0 到 10，其中編號 0 的房間為不存在的房間，表示這個方向並無房間，接著便可以開始透過使用 setDownRoom(Room*)、setUpRoom(Room*)、setLeftRoom(Room*)、setRightRoom(Room*)開始連接房間，接著再去設定每一間房間可能會出現的 NPC、Item 或是 Monster，最後再透過 setObjects(Object*)的 function 將它放入此間房間的 objects vector 裡面。第三個則是 handleMovement()，這個 function 主要是負責處理玩家選擇的各種動作(有 4 個: 1.check your status 2.move to another room 3.check your backpack 4.save file)，當玩家選擇 1 時，透過 player 的 triggerEvent(Object* object)來展現出角色目前的各個能力數值；如果選擇 2，則讓玩家選擇要往上下左右哪個方向移動，並在成功移動之後，顯示”now you are in the room (編號)”，並透過 player 的 setter 跟 getter 來更新 currentRoom 和 previousRoom 的數值，假使移動的方向上沒有房間，則顯示”There is no room in this direction. Please choose again.”；如果選擇了 3，則是利用 getInventory()來取得 player 的 inventory，接著去判斷這個 vector 是否為空，如果為空，則輸出” You don't have anything in your backpack.”，如果不為空，則利用 for 迴圈將背包內的每一個 item

輸出；如果選擇了 4，則是會使用 Record 中 saveToFile(&player, room)的這個 function，將目前 player 和 room 的情況存檔下來(輸出 txt 檔)。第四個 function 為 startGame()，主要功用為呼叫 createMap()和 createPlayer()這兩個 function，將遊戲剛開始的基本背景設定好:最後一個 function 則是 runDungeon()，負責遊戲大部分的運行，首先，先決定是否讀取之前的存檔，如果要，就使用 record 中 loadFromFile(Player*, room*)的 function，來讀取之前存的檔案(txt 檔)，如果不要就使用 startGame()開始新的遊戲，然後利用 while(true)的迴圈讓程式不斷重複，迴圈內首先利用 handleMovement()來讓玩家選擇動作，如果玩家有移動房間，則根據移動到的房間內的 object 來做出不同的情境，比如說如果有 Monster，則詢問玩家要戰鬥或是撤退，如果玩家選擇戰鬥，則透過 Monster 的 triggerEvent(Object* object)來判斷是否可以進行戰鬥，如果可以，則繼續呼叫 combatSystem(Player* player)來進行戰鬥，直到玩家或怪物死亡，或是玩家選擇撤退，如果玩家死亡，則詢問玩家是否要儲存遊戲，如果需要，則使用 record 中的 saveToFile(Player*player, vector<Room*>&room) function 來儲存；又比如說如果房間內有 trader(NPC)，則先詢問玩家是否有要購買商品，如果有，則透過 NPC 的 triggerEvent(Object* object)來進行交易；又或者是房間內有寶箱，如果此 item 的 isTake 為 0 時，則詢問玩家是否要開啟寶箱，如果要開啟寶箱，則同樣透過 Item 的 triggerEvent(Object* object)來提升玩家的能力值，並利用 player 的 addItem(Item* item) 將寶箱內的物品放入 inventory 內。而遊戲最後要通關，則是必須打敗第 10 號房的 Boss，此外要進入第 10 號房還有一個條件，那就是要先打敗第 6 號房的怪獸，並在之後打開寶箱拿到鑰匙，才有辦法進入第 10 號房。

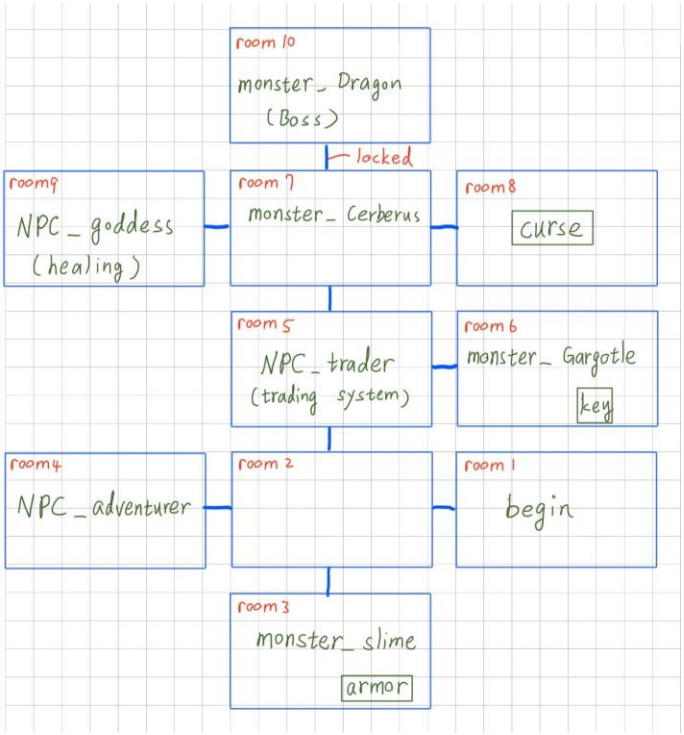
9. Record: 有 4 個 private function，分別為 savePlayer(Player* player)(將 player 的相關資訊輸出成 txt 檔)、saveRooms(vector<Room*>&room)(將整個地圖的相關資訊輸出成 txt 檔)、loadPlayer(Player*player, vector<Room*>&room)(從之前存的 txt 檔，讀入相關資訊，創建角色)、loadRooms(vector<Room*>&room)(從之前存的 txt 檔，讀入相關資訊，創建地圖)。另外除了基本的 constructor，還有兩個 public function，分別為 saveToFile(Player*player, vector<Room*>&room)(透過 savePlayer(Player* player)和 saveRooms(vector<Room*>&room))兩個 function 來存取所有資料。) 跟 loadFromFile(Player*player, vector<Room*>&room)(透過 loadPlayer(Player*player, vector<Room*>&room)和 loadRooms(vector<Room*>&room)讀取所有的相關資訊，創建當時存檔時的情形)。


Dungeon
- player: Player
- room: vector<Room*>
- record: Record*
+ Dungeon()
+ createPlayer(): void
+ createMap(): void
+ void startGame(): void
+ void runDungeon(): void
Player
- currentRoom: Room*
- previousRoom: Room*
- showStatus(): void
- inventory: vector<Item*>
- money: int
- key: bool
+ Player()
+ addItem(Item* item): void
+ increaseStates(int Attack,int Defense,int Maxhealth, int Health): void
+ changeRoom(Room*room): void
+ void move_picture(): +
+ triggerEvent(Object*object): bool
+ setCurrentRoom(Room*room): void
+ setPreviousRoom(Room*room): void
+ setMoney(int Money): void
+ takeAction(): void
+ setKey(bool Key): void
+ getCurrentRoom(): Room*
+ getPreviousRoom(): Room*
+ getKey(): bool
+ getInventory(): vector<Item*>
+ getMoney(): int
Record
+ Record()
- savePlayer(Player* player): void
- saveRooms(vector<Room*>&room): void
- loadPlayer(Player*player, vector<Room*>&room): void
- loadRooms(vector<Room*>&room): void
- saveToFile(Player*player, vector<Room*>&room): void
- loadFromFile(Player*player, vector<Room*>&room): void
Room
- upRoom: Room*
- downRoom: Room*
- leftRoom: Room*
- rightRoom: Room*
- isExit: bool
- index: int
- objects: vector<object*>
+ Room()
+ setUpRoom(Room*room): void
+ setDownRoom(Room*room): void
+ setLeftRoom(Room*room): void
+ setRightRoom(Room*room): void
+ setIsExit(bool IsExit): void
+ setIndex(int Index): void
+ setObjects(Object* thing): void
+ getIsExit(): bool
+ getIndex(): int
+ getObjects(): vector<Object*>
+ getUpRoom(): Room*
+ getDownRoom(): Room*
+ getLeftRoom(): Room*
+ getRightRoom(): Room*
NPC
- script: string
- commodity: vector<Item*>
+ NPC()
+ listCommodity(): void
+ triggerEvent(Object* object): bool
+ setScript(string): void
+ setCommodity(vector<Item*>Commodity): void
+ getScript(): string
+ getCommodity(): vector<Item*>
Monster
- room: Room
- money: int
- triggerEvent(Object* object): bool
- combatSystem(Player* player): void
- setRoom(Room assignedRoom): void
- setMoney(int Money): void
- getMoney(): int
Extends
GameCharacter
- maxHealth: int
- currentHealth: int
- attack: int
- defense: int
- currentCD: int
- occupation: string
- skill: string
+ GameCharacter()
+ checkIsDead(): bool
+ setMaxHealth(int MaxHealth): void
+ setCurrentHealth(int CurrentHealth): void
+ setAttack(int Attack): void
+ setDefense(int Defense): void
+ setCurrentCD(int CurrentCD): void
+ setSkill(string Skill): void
+ setOccupation(string Occupation): void
+ getMaxHealth() const: int
+ getCurrentHealth() const: int
+ getAttack() const: int
+ getDefense() const: int
+ getCurrentCD(): int
+ getOccupation(): string
+ getSkill(): string
Item
- health: int
- attack: int
- defense: int
- price: int
- maxhealth: int
- isTake: bool
+ Item()
+ triggerEvent(Object* object): bool
+ getHealth(): int
+ getAttack(): int
+ getDefense(): int
+ getPrice(): int
+ getMaxHealth(): int
+ getIsTake(): bool
+ setIsTake(bool IsTake): void
+ setMaxHealth(int MaxHealth): void
+ setHealth(int Health): void
+ setAttack(int Attack): void
+ setDefense(int Defense): void
+ setPrice(int Price): void
Extends
Object
- name: string
- name: tag
+ Object()
+ triggerEvent(Object* object) = 0: bool
+ setName(string Name): void
+ setTag(string Tag): void
+ getName() const: string
+ getTag() const: string

UML ↑

Results:

遊戲地圖:



遊戲最開始會問玩家是否要讀取舊檔，如果選擇要，就會呼叫 record 的 loadFromFile(Player*player, vector<Room*>&room)，從上次存檔的地方開始。(如下圖)

```
Do you want to load the previous record?
1.Yes   2.No
1

Now you are in room 9
What do you want to do?
1.check your status    2.move to another room  3.check your backpack   4.save file
1

Name: Jason
Attack: 800       Defense: 350
HP: 950 / 950
Occupation: Tank
Skill: Absolute-Shield     CD: 4
Money: 2500
```

如果不選擇讀取舊檔，則從最基本的創建角色和地圖開始(使用 startGame())。

```
Do you want to load the previous record?
1.Yes   2.No
2
Welcome to this magical dungeon.
Your task is trying to find the exit. Good luck!

What's your name?
Jason

Which occupation do you want to choose?(TYPE NUMBER)
1.WARRIOR    2.TANK    3.WIZARD
2

Name: Jason
Attack: 250       Defense: 300
HP: 400 / 400
Occupation: Tank
Skill: Absolute-Shield     CD: 5
Money: 500
```

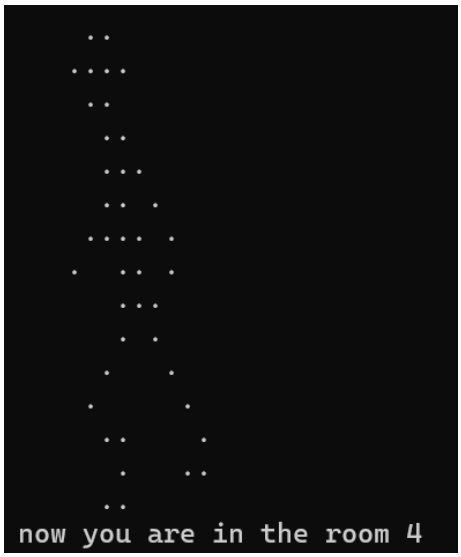
此遊戲最基本的 4 個選擇便是 handleMovement()這個 function 提供的 4 個選項(如下圖)。

```
now you are in the room 2
What do you want to do?
1.check your status    2.move to another room  3.check your backpack   4.save file
```

當選擇 1 時，會呼叫 player 的 triggerEvent(Object* object)，顯示玩家基本能力數值。(如下圖)

```
What do you want to do?
1.check your status    2.move to another room  3.check your backpack   4.save file
1

Name: Jason
Attack: 400       Defense: 250
HP: 300 / 300
Occupation: Warrior
Skill: Whirlwind     CD: 4
Money: 1000
```

當選擇 2 時，則會先使用 player 中 move_picture()的 function 顯示行人走路的動圖，接著再用 player 中的 setter 更改移動後的房間位置，並將其顯示出來。(如下圖)



當選擇 3 時，則利用 player 中 getInventory()的 function 將背包中有的 item 輸出。(如下圖)

```
What do you want to do?
1.check your status    2.move to another room  3.check your backpack   4.save file
3

backpack: armor | sword | attack_rune
```

當選擇 4 時，則利用 record 中的 saveToFile(Player*player, vector<Room*>&room)的 function 來進行存檔。

接著則是各個房間會遇到的突發事件。

1. room4:在此房間會遇到一個 adventurer，並觸發對話，之後會有一個 adventurer 遺留的武器，如果玩家選擇接受，就能獲得相對應玩家職業的武器。接著使用 Item 的 triggerEvent(Object* object)來提升玩家的基本能力數值，然後使用 addItem(Item* item)將其放入背包中。(如下圖)

```
adventurer : I...I encountered a horrible monster and got attacked. I am afraid I'll die soon.
Pl..Please defeat the monster and save the world on my behalf. (adventurer died)

The adventurer left a weapon to you.
Do you want to take?
1.Yes      2.No
1
Congratulations you got a shield!!!
Now your defense will increase 100.
```

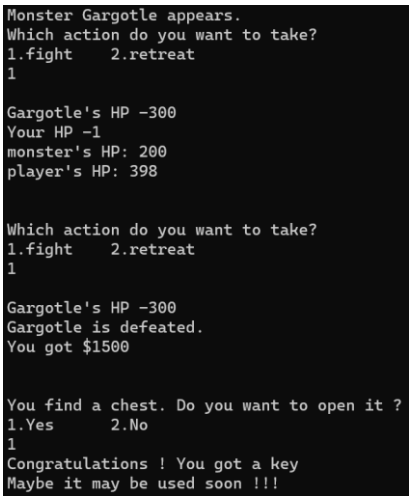
2. room3: 在此房間會遇到一個 monster，然後詢問玩家要選擇 fight 或是 retreat,如果選擇 retreat 則退回上個房間,如果選擇 fight,則使用 Monster 的 triggerEvent(Object* object)來進行戰鬥，如果玩家死亡，則詢問是否需要存檔。當玩家戰勝 monster 後，會跑出一個寶箱，如果玩家選擇開啟，就能獲得 armor，接著使用 Item 的 triggerEvent(Object* object)來提升玩家的基本能力數值，然後使用 addItem(Item* item)將其放入背包中。(如下圖)

```
now you are in the room 3
Monster Slime appears.
Which action do you want to take?
1.fight    2.retreat
1

Slime's HP -100
Your HP -1
monster's HP: 100
player's HP: 399


Which action do you want to take?
1.fight    2.retreat
1

Slime's HP -100
Slime is defeated.
You got $500


You find a chest. Do you want to open it ?
1.Yes      2.No
1
Congratulations ! You got a new weapon: armor
Now your defense will increase 50
```

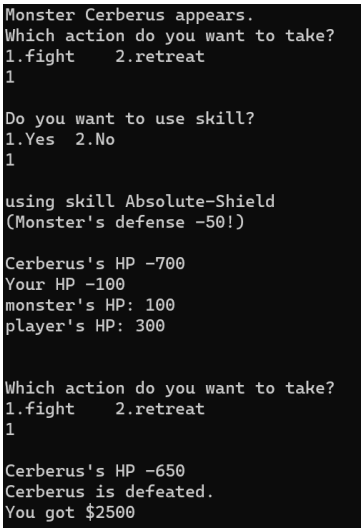
3. room5: 在此房間會遇到一個 trader，並觸發對話，當玩家選擇和 trader 進行交易，就使用 NPC 的 triggerEvent(Object* object)開啟交易系統，讓玩家選擇要購買的商品。(如下圖)

```
trader : Welcome my shop, youngster. Do you want to buy something?
1.of course.     2. No.You look suspicious!
1
Which commodity do you want to trade? (If you want to leave, press 0)
(Now you have $1000)
(1)attack_rune($300)     (2)defense_rune($300)     (3)health_rune($300)     (4)health_potion($100)
1

Congratulations !!! Now you are more powerful !
Which commodity do you want to trade? (If you want to leave, press 0)
(Now you have $700)
(1)attack_rune($300)     (2)defense_rune($300)     (3)health_rune($300)     (4)health_potion($100)
2

Congratulations !!! Now you are more powerful !
Which commodity do you want to trade? (If you want to leave, press 0)
(Now you have $400)
(1)attack_rune($300)     (2)defense_rune($300)     (3)health_rune($300)     (4)health_potion($100)
0
What do you want to do?
1.check your status    2.move to another room  3.check your backpack   4.save file
3

backpack: shield | armor | attack_rune | defense_rune
```

4. room6: 在此房間會遇到一個 monster，然後詢問玩家要選擇 fight 或是 retreat,如果選擇 retreat 則退回上個房間,如果選擇 fight,則使用 Monster 的 triggerEvent(Object* object)來進行戰鬥，如果玩家死亡，則詢問是否需要存檔。當玩家戰勝 monster 後，會跑出一個寶箱，如果玩家選擇開啟，就能獲得一把 key，使用 addItem(Item* item)將其放入背包中，這把 key 為最後進入 Boss 房所需要的必要品。(如下圖)



5. room7: 在此房間會遇到一個 monster，然後詢問玩家要選擇 fight 或是 retreat,如果選擇 retreat 則退回上個房間,如果選擇 fight,則使用 Monster 的 triggerEvent(Object* object)來進行戰鬥，如果玩家死亡，則詢問是否需要存檔。(如下圖)



6. room8: 當玩家進入房間後，會跑出一個寶箱，如果玩家選擇開啟，就會遭受詛咒(curse)，造成玩家基本能力數值降低，這邊同樣使用 Item 的 triggerEvent(Object* object)來更動玩家的基本能力數值。(如下圖)

```
You find a chest. Do you want to open it ?
1.Yes      2.No
1
Oops ! You got a curse from a demon
Your attack, defense, and health decrease 50 !!!
```

7. room9: 在此房間會遇到一個 goddess，並觸發對話，讓玩家選擇是否要接受治癒(Healing)，如果玩家選擇接受，就扣除 100 元，並將其血量回滿(使用 player 的 getter 和 setter 來設定治癒後的血量)，最後使用 player 的 triggerEvent(Object* object)，顯示玩家基本能力數值。。 (如下圖)

```
goddess : It has been a long time no visitors coming! Brave man, how can I help you?
1.Healing(need cost $100)      2.No,thanks
1

Now you can have more adventure!
Name: Jason
Attack: 800       Defense: 350
HP: 950 / 950
Occupation: Tank
Skill: Absolute-Shield     CD: 4
Money: 2500
```

8. room10: 唯有玩家背包內有鑰匙才能進入此房間(利用 getKey()來判斷)，進入後，在此房間會遇到一個 boss，然後詢問玩家要選擇 fight 或是 retreat，如果選擇 retreat 則退回上個房間，如果選擇 fight，則使用 Monster 的 triggerEvent(Object* object)來進行戰鬥，如果玩家死亡，則詢問是否需要存檔。當玩家戰勝 boss 後，玩家即勝利，遊戲結束。

```
You use the key opening the boss room !!!
Monster dragon appears.
Which action do you want to take?
1.fight    2.retreat
1

dragon's HP -500
Your HP -450
monster's HP: 500
player's HP: 500


Which action do you want to take?
1.fight    2.retreat
1

dragon's HP -500
monster's HP: 0
player's HP: 500


You defeat the boss and save the world !!!
Game Over!!!
```

對戰系通中落敗的情形(如下圖)

```
Monster Cerberus appears.
Which action do you want to take?
1.fight    2.retreat
1

Cerberus's HP -50
Your HP -150
monster's HP: 750
player's HP: 248


Which action do you want to take?
1.fight    2.retreat
1

Do you want to use skill?
1.Yes  2.No
1

using skill Absolute-Shield
(Monster's defense -50!)

Cerberus's HP -100
Your HP -150
monster's HP: 650
player's HP: 98
```

```
Which action do you want to take?
1.fight    2.retreat
1

Cerberus's HP -50
Your HP -150
You are dead.


do you want to save file?
1.Yes   2.No
1
```

Discussion:

1. 在使用 virtual function 時，因為傳入的 argument 為 Object*的型態，而缺少了許多需要的資料(像是 player 的 attack、monster 的 HP 等)，而不知如何下手。後來查閱課本才曉得可以使用 dynamic_cast 進行 downcasting，如此一來便能取得想要的資料。
2. 如果兩個 class 都有互相使用到以對方為型態的變數時，要先將一個定義好後，接著 compile，再去定義另外一個。而不是同時將兩個都一次寫好，然後才去 compile，這樣會造成兩個 class 都互相不知道對方的定義。
3. 在撰寫程式中，有一個 function 始終無法達到我變更數值的指令，後來在 debug 時才發現，原來是因為我使用了 call by value。Call by reference 或 Call by address 在這種較大規模的程式中，我覺得好用許多，除了不需要使用額外的記憶體，還能讓你確實更改你想變的數據。
4. 在前期寫 dungeon 時，覺得從 main function 開始慢慢刻，應該也能完成此次作業，因此我原本認為使用到 OOP 的概念有點多此一舉。但當我寫到後期，才發現了 OOP 的好處，將一個個的人物、物品或動作模組化，不只能避免相同動作的程式碼反覆出現，還能讓整個程式變得更簡潔，讓撰寫者與觀看者都能更快了解這個程式的運作原理。

Conclusion:

此次的 dungeon 遊戲，相較於以往練習寫的程式，規模大上了許多，因此對於練習使用 OOP 的技巧與概念十分有幫助。在設計這次的遊戲時，我認為最一開始在紙上畫的 UML 圖是很重要的，除了能夠讓我們在一開始慢慢去設想自己遊戲所需的規模以及物件，更重要的是，在之後開始寫程式時，能讓你快速了解各個物件的關聯以及負責功用，減少需要一邊寫程式一邊思考邏輯的時間。再來便是各個 CLASS 的刻寫，我覺得這個步驟是最為枯燥的，因為沒辦法即時看到寫出的程式的功效，但這個步驟卻又極其重要，相當於我們在製造一台大機器所有需要的零件，如果不嚴謹地將每個零件的功用定義好，將來在組裝時，便會造成很大的差錯。最後就是利用 Dungeon class，把遊戲從頭到尾開始刻畫出來，由於大部分的功能前面的class都已經定義好了，所以這一步做得十分的迅速，也是從這時我才知道，OOP 是一個非常有用的概念。對於此程式我認為大體上還算完整，但比其平常玩的遊戲，就顯得單調許多，未來可以試著多增加一些元素在遊戲中，像是房間的隨機生成、魔寵的飼養、戰鬥的簡易小動畫等，這些都可以藉由 OOP 的概念使得遊戲更加生動，並將 dungeon 的規模更加擴大。