



DUNGEON

Content

• Basic Function	1
• Action Menu	1
• Movement	2
• Showing Status	3
• Map	4
• Character Class & Skill	6
• Pick up Items & Wear Equipment	7
• Fightings System	8
• NPC , Trading System	10
• Record System	11
• Easter Egg	11
• Game Logic	12
• Program Detail	13
• Class UML	17
• Discussion	18
• Conclusion	18

• Action Menu

玩家在進入每個房間時可以選擇自己想要做的行動，如下圖所示：

What do you want to do?
A. Move
B. Talk to Your Friend's Steve
C. Pick up Iron Boots
D. Show Map
E. Show Status
F. Wear equipment or weapon
G. Leave the Dungeon

Your choice:

可以看到圖中有七個選項可以選擇，至於每個動作個別如何執行會再等等提到。另外如果進到一個房間後遇到怪物，玩家在擊敗該怪物之前並沒有辦法解鎖該房間連接的其他房間，意思是必須擊敗他才能繼續前進，而且打敗他之後才會有掉落物品可以撿，如上圖中 C 的選項。以下是遇到怪物會有的行動畫面：

You encounter a monster. What do you want to do?
A. Fight with Annoying Friend
B. Retreat(back to previous room)
C. Show Status
D. Wear equipment or weapon
E. Leave the Dungeon

Your choice:

實現此功能的是Dungeon.h/Dungeon.cpp中的chooseAction()函式，他會在一個 while 迴圈裡不斷執行直到 game logic 有所改變，並會判斷玩家目前可以執行的行動並輸出選擇（某些特定房間的特殊事件也是這個函數在處理），該函數在收到玩家的輸入會在一個switch迴圈裡去執行結果，主要配合的函式為：`handleMovement()`、`handleEvent()`、`handleMap()` 以及 Player Class 中的 member function `equip()`。其中最重要的是`handleEvent()`他可以根據其參數的類型同時處理多個行動，即 Object Class 中的 virtual function: `triggerEvent()`。

• Movement

在上一頁有提到玩家有許多動作可以選擇去執行，其中一項最基本的就是**移動**。玩家可以選擇要往上下左右的房間移動，但要注意的並不是每個選項都會有，而是會根據玩家當下所在的房間的情況決定。如下圖所示：

Where do you want to go?

- A. Move Up.
- B. Move Down.
- C. Move Left.

Your choice:

實現這點的是在 Dungeon.h/Dungeon.cpp 中的 member function：handleMovement()，該函式會判斷玩家目前所在房間的上下左右房間鄰居的指標是否為NULL，如果不為NULL的會就會輸出一個選項讓玩家選擇：

```
● ● ●
1 char option = 'A', input;
2 map<char, char> decision;
3 cout << "\n\033[1;33mWhere do you want to go?\033[m\n";
4 if (player.getCurrentRoom()->getUpRoom() != NULL)
5 {
6     cout << option << ". Move Up.\n";
7     decision[option] = 'u'; // stand for up
8     decision[option + 32] = 'u';
9     option++;
10 }
11 if (player.getCurrentRoom()->getDownRoom() != NULL)
12 {
13     cout << option << ". Move Down.\n";
14     decision[option] = 'd'; // stand for down
15     decision[option + 32] = 'd';
16     option++;
17 }
18 if (player.getCurrentRoom()->getLeftRoom() != NULL)
19 {
20     cout << option << ". Move Left.\n";
21     decision[option] = 'l'; // stand for left
22     decision[option + 32] = 'l';
23     option++;
24 }
25 if (player.getCurrentRoom()->getRightRoom() != NULL)
26 {
27     cout << option << ". Move Right.\n";
28     decision[option] = 'r'; // stand for right
29     decision[option + 32] = 'r';
30 }
31 }
```

• Showing Status

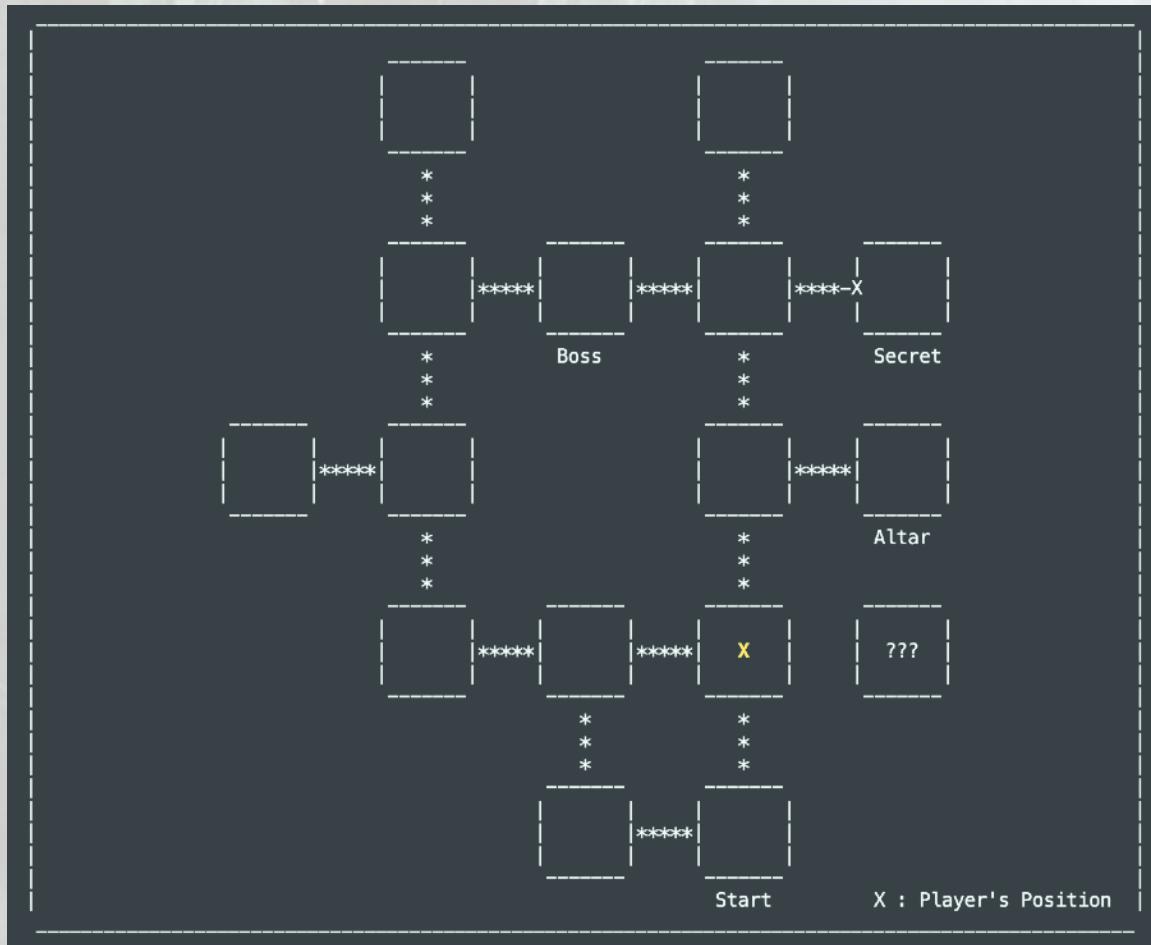
玩家可以執行的第二項基本行動是顯示狀態。會顯示出目前玩家的各種數值，如血量、魔力、攻擊、防禦、金幣等，也會顯示目前玩家身上所穿的裝備，以及裝備的各種數值，如果是空的話名稱會顯示為X。同時也可以看到玩家的背包裡的物件，如果沒有穿上的裝備會收到背包裡。另外為了方便閱讀，我還另外使用了setw()及left等函式來幫助排版，如下圖所示：

```
Player Status:  
[TA]  
> Class: Saber  
> Coin: 70  
> HP: 100/120  
> MP: 100/110  
> Attack: 255  
> Defense: 90  
Equipment:  
> Weapon (Sword): Frie Sword (HP + 20, MP + 0, ATK + 30, DEF + 30)  
> Helmet: X (HP + 0, MP + 0, ATK + 0, DEF + 0)  
> Necklace: Shadow Belle (HP + 0, MP + 0, ATK + 15, DEF + 0)  
> Armor: X (HP + 0, MP + 0, ATK + 0, DEF + 0)  
> Shield: Apocalypse (HP + 20, MP + 0, ATK + 0, DEF + 40)  
> Pants: X (HP + 0, MP + 0, ATK + 0, DEF + 0)  
> Boots: Monarch Wings (HP + 30, MP +100, ATK + 30, DEF + 0)  
Backpack:  
> Dungeon Map (Map , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)  
> Iron Shield (Shield , HP: + 0, MP: + 0, ATK: + 0, DEF: + 20)  
> Cross (Item , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)  
> Holy Grail (Item , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)  
> Iron Boots (Boots , HP: + 0, MP: + 0, ATK: + 15, DEF: + 0)  
> Iron Sword (Sword , HP: + 15, MP: + 0, ATK: + 30, DEF: + 15)  
> Elegant Key (Key , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)
```

實現此功能的是在Player.h/Player.cpp中的virtual function：triggerEvent()，handleEvent()會收到一個Object的指標並執行該Object的triggerEvent，因此可以同時對不同類型的物件去做處理（如Monster、NPC、Player）其中若是指向Player類型物件的Object指標，就會執行Showing Status，之後只需要用Player中的各個get function就可以得到每一個數值了。

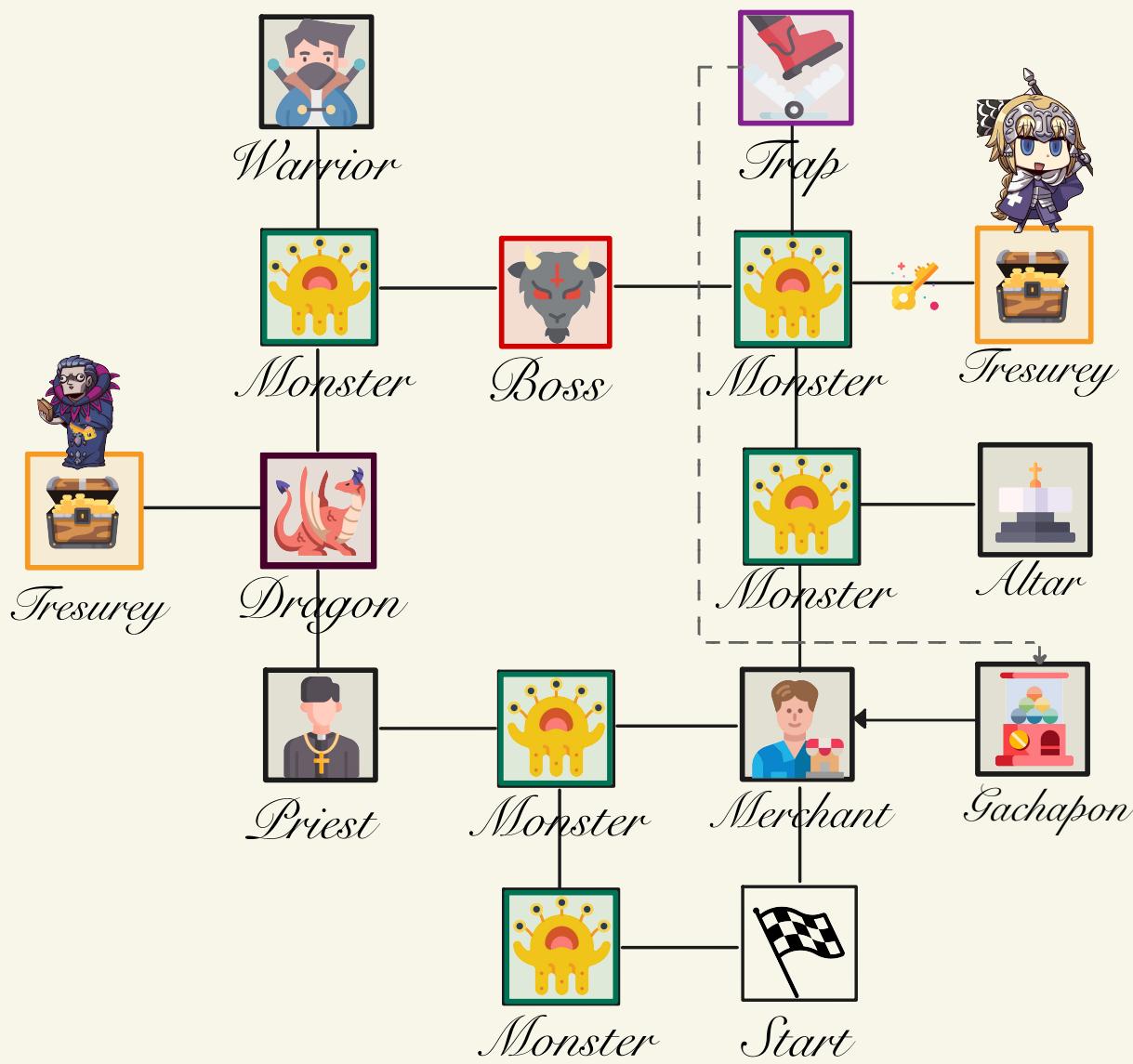
• Map

玩家可以執行的行動之一是顯示地圖，但條件是必須擁有該物品，可以透過跟商人「Your Friend Steve」購買，之後會在chooseAction()中判斷玩家是否有該物品（利用Player Class中的checkMap()函式），有的話則可以根據玩家所在位置去輸出整張地下城的地圖，由Dungeon.cpp中的handleMap()函式處理，方法是窮舉出玩家所有位置的可能性並放在Switch迴圈內做判斷。



地下城地圖，標示了各房間主要物件：

Map



• Character Class & Skill

• 職業

玩家在剛進入遊戲時可以選擇想要的職業，總共有七個職業：Saber、Archer、Lancer、Caster、Caster、Assassin、Berserker，也會有不同的能力分配、對應的武器、以及不同的招式，若撿到其他職業的武器也不能用，只能裝備自己職業的武器。

Choose your character's class:

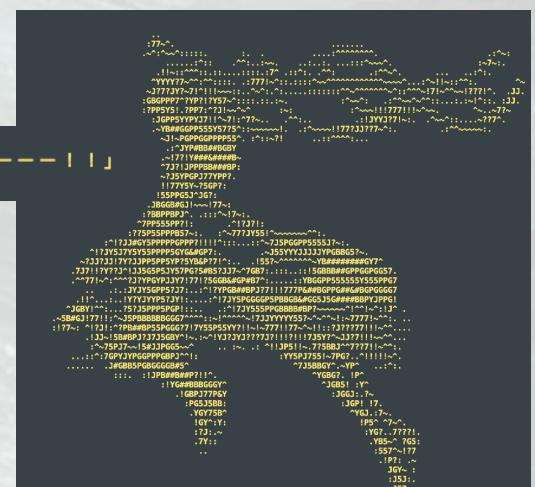
- A. Saber (HP: 70, MP: 10, ATK: 30, DEF: 20)
- B. Archer (HP: 50, MP: 20, ATK: 50, DEF: 10)
- C. Lancer (HP: 50, MP: 10, ATK: 50, DEF: 20)
- D. Caster (HP: 50, MP: 30, ATK: 40, DEF: 20)
- E. Assassin (HP: 60, MP: 10, ATK: 50, DEF: 10)
- F. Rider (HP: 60, MP: 20, ATK: 40, DEF: 10)
- G. Berserker (HP: 70, MP: 0, ATK: 50, DEF: 30)

Your Choice: A

• 技能

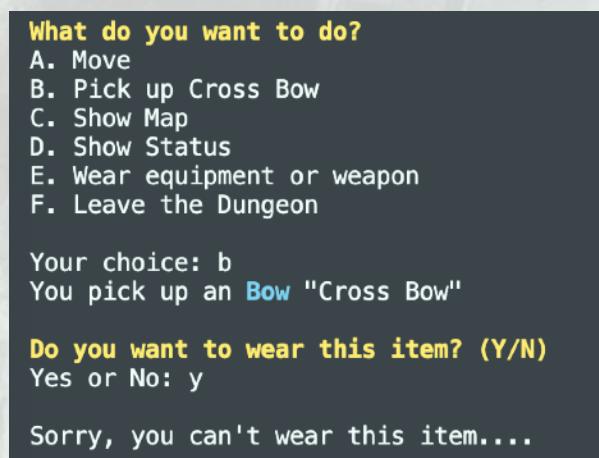
主要分為普通技能一、二，以及大招，普通技能主要是用來加buff（回血、加攻擊防禦、或是補MP），而大招則是要耗費較多的MP才能使用，但效果也相對的較強，可以打出很高的傷害。實作方式是另外開一個Class Skill，private member有技能名稱、消耗的魔力量、技能種類（回血、攻擊等），以及對應的數值，最後還有招式的台詞。

「——東ねるは星の息吹、輝ける命の奔流。受けるが良い！『約束された勝利の剣』——！」



• Pick up Items & Wear Equipment

玩家若目前所在的房間有物品掉落，會顯示在Action Menu中並詢問玩家是否要撿起該物品。若撿起的物品屬於裝備或武器類的，系統會接著詢問是否要裝備該物品，如果該武器類型符合玩家的職業的話就會成功裝備，如果否的話就無法裝備，如下圖：



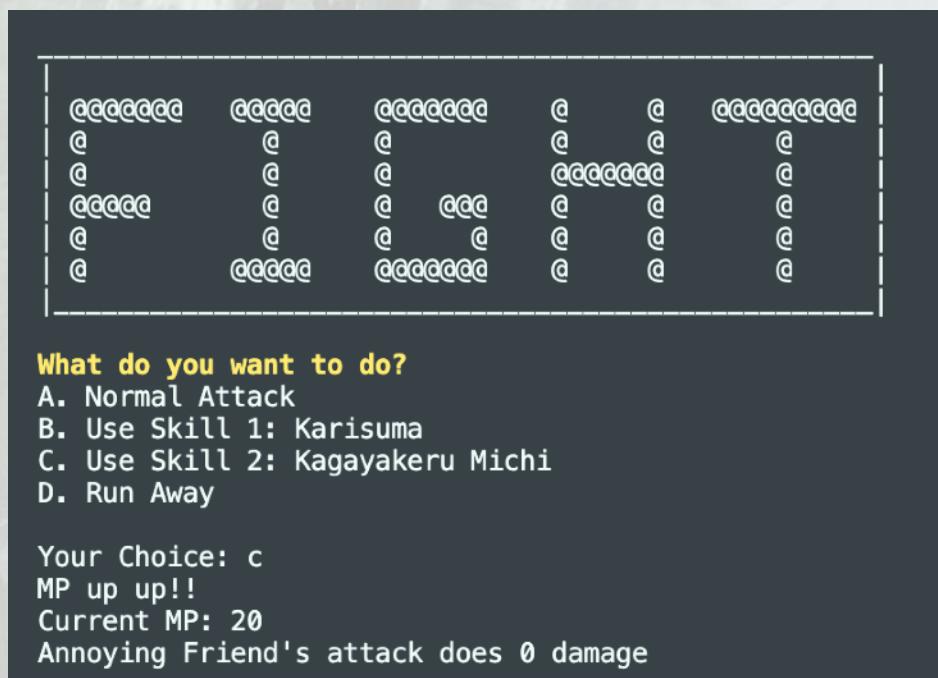
另外，若玩家想要更換身上的裝備，或是裝備背包中的物品，他也可以在Action Menu中選擇這個選項，系統會接著呼叫Player Class 中的equip()，頁面如下：



接著會根據玩家擁有該類型的物品數量去一一列出可以裝備的選項讓玩家做選擇，如果已經裝備同類型的物品也會將其更換並收到背包中。

• Fighting System

戰鬥系統是由Monster中的triggerEvent()函式負責處理，該戰鬥是回合制的，每回合玩家可以選擇想要做的行動，如普通攻擊、或是使用技能，但並不是所有技能都可以用，需要滿足一定數量的魔力量（MP）才可以使用，且用完後亦會消耗MP，因此沒辦法一直開，除非該技能是增加MP類型的技能，這樣就可以每回合都使用，不過玩家仍會被怪物攻擊，因此要評估一下怪物的傷害再做決定。而傷害的計算方式是：自己的攻擊力扣掉敵方的防禦力=造成的傷害，兩邊的計算都是如此，因此如果防禦力夠高的話，說不定怪物的攻擊不會造成任何傷害。如下圖：



另外如果玩家如果評估戰況後覺得打不過，也可以選擇撤退，等準備好後再次挑戰，已造成的傷害仍舊會在，並不會被清除。如下圖：

What do you want to do?

- A. Normal Attack
- B. Use Skill 1: Karisuma
- C. Use Skill 2: Kagayakeru Michi
- D. Use Ultimate Skill: Excalibur
- E. Run Away

Your Choice: e

Monster Goblin is still alive.....

若玩家打敗該怪獸，玩家會得到十塊錢，若打敗的是小王（青眼白龍）則會得到三十塊。

Player Status:
[TA]
> Class: Saber
> Coin: 20
> HP: 50/50
> MP: 0/10
> Attack: 140
> Defense: 20

Monster Status:
[Annoying Friend]
> HP: 0/150
> MP: 0/0
> Attack: 5
> Defense: 0

Congratulaitons!! This monster is dead!!
You get 10 coins

Your attack does 310 damage.
Blue-Eyes White Dragon's attack does 10 damage

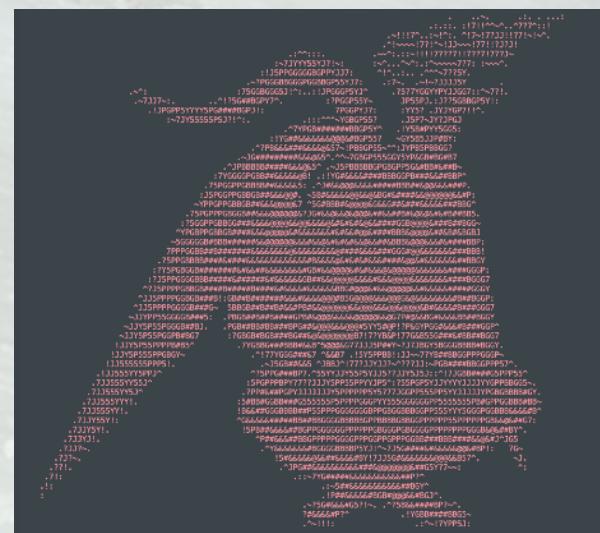
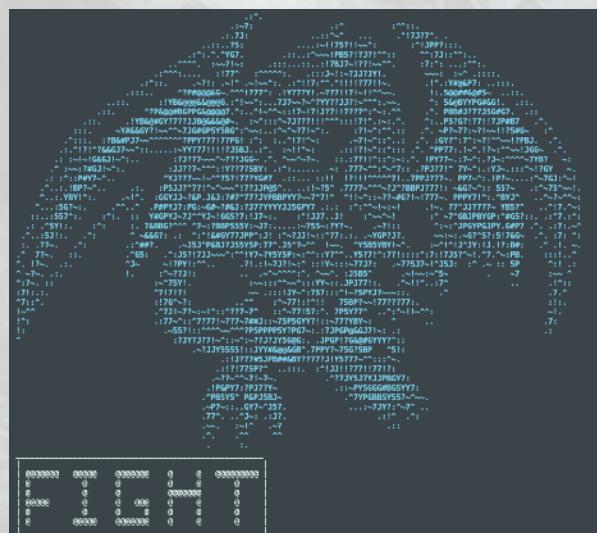
Player Status:
[TA]
> Class: Saber
> Coin: 50
> HP: 30/60
> MP: 0/10
> Attack: 330
> Defense: 50

Monster Status:
[Blue-Eyes White Dragon]
> HP: 0/200
> MP: 0/0
> Attack: 60
> Defense: 20

(Something special things was dropped from the dragon)

Congratulaitons!! Dragon is dead!!
You get 30 coins.

打敗怪物後有可能會掉落一些裝備或武器，可以在Action Menu中執行動作，有趣的是如果遇到的是小王或是最後的Boss，會有很帥的戰鬥畫面輸出，如下：



• NPC, Trading System

NPC的Trading System是由是由NPC中的triggerEvent()函式負責處理，該功能會輸出NPC的台詞並用藍色顯示，接著列出所有該NPC擁有的物品以物品名稱、能力數值、以及價錢，如下。選擇想要購買的商品後，函式會判斷你是否擁有足夠的錢，如果不夠的話會顯示訊息，如下：

```
" Dododo~ do~ dodo~ do~ Steveeeee! "

> A. XenoBlade      (Sword      , HP: + 20, MP: + 0, ATK: + 30, DEF: + 20)    40 coins
> B. Truestrike     (Bow        , HP: + 15, MP: + 5, ATK: + 50, DEF: + 0)    40 coins
> C. Gae Bolg        (Lance      , HP: + 10, MP: + 0, ATK: + 50, DEF: + 10)   40 coins
> D. Avada Kedavra  (Staff      , HP: + 0, MP: + 20, ATK: + 10, DEF: + 0)    40 coins
> E. Sacrifice       (Dagger     , HP: + 0, MP: + 0, ATK: + 70, DEF: + 0)    40 coins
> F. Eternal Rest    (Whip       , HP: + 30, MP: + 0, ATK: + 40, DEF: + 0)    40 coins
> G. Apocalypse        (Shield     , HP: + 20, MP: + 0, ATK: + 0, DEF: + 40)   20 coins
> H. Crystal Heart    (Necklace   , HP: + 0, MP: + 40, ATK: + 0, DEF: + 0)    25 coins
> I. King's Brand     (Helmet     , HP: + 20, MP: + 20, ATK: + 10, DEF: + 0)   30 coins
> J. Exit.

f
Sorry, You don't have enough money.
```

另外有些NPC會提供特殊的物品，這些物品不用錢，並會在特殊的房間起作用，如下：

```
" Yorokobe syounen. Kimi no moti nozomi wa youyaku kanou..... "

> A. Cross          (Item      , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)    0 coins
> B. Holy Grail      (Item      , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)    0 coins
> C. Exit.
```

```
" I am Gilles de Rais, and I have come to you at your invitation. From here on, I shall be at your command. "

> A. Elegant Key    (Key       , HP: + 0, MP: + 0, ATK: + 0, DEF: + 0)    0 coins
> B. Exit.
```

• Record System

該遊戲也有建立存檔系統，會在玩家離開遊戲時詢問是否要存檔，如果玩家有存檔紀錄，也會在進入遊戲時詢問是否要讀取上次的檔案。

• Easter egg

1. 陷阱&扭蛋機

玩家在進入某一間房間時（編號14），會觸法陷阱並強制與一個能力較高的怪物戰鬥，如果成功的話會被傳送至一個扭蛋機，每用一次10金幣，且有5%的機會抽到最大獎的裝備。

```
You've fallen into a trap, and you encounter a monster.  
Defeat it or you'll die....  
  
You've entered a mysterious place. And there is a machine in front of you.  
Do you want to use it? (Y/N)  
Yes/No :y  
  
It's a gachapon machine. Little prize: 10%, Big prize: 90%.  
Put in 10 coins to get a gachapon. Yes or No?  
  
Your Choice: y  
Sorry...You got nothing. Keep trying.  
Put in 10 coins to get a gachapon. Yes or No?  
Your Choice: y  
  
Not Bad!! You got 10 coins back!! Keep trying.  
Put in 10 coins to get a gachapon. Yes or No?
```

2. 聖女貞德&吉爾德雷

如果玩家選的是Saber職階，遇到吉爾德雷（Gilles de Rais）時他會把玩家誤認成是貞德（Fate梗），因此會把鑰匙交給他。接著玩家就可以到神秘房間（編號13）解鎖，裡面會遇到貞德，他會給玩家一件很強力的裝甲，該房間也會掉落 Saber 的專屬寶具。

```
You've entered a secret chamber.  
You have the key. Welcome!  
  
What do you want to do?  
A. Move  
B. Talk to Jeanne d'Arc  
C. Pick up Excalibur  
D. Show Map  
E. Show Status  
F. Wear equipment or weapon  
G. Leave the Dungeon  
  
Your choice: b  
  
"Hello! Hahaha, someone must mistake you for me."  
> A. Luminoise Eternelle (Armor      , HP: +500, MP: +500, ATK: +500, DEF: +500)      0 coins  
> B. Exit.
```

3. 祭壇

如果玩家在神父那裡拿到聖杯以及十字架，可以在祭壇的地方回滿血量及魔力，並會提升玩家攻擊力以及防禦力各20點，但用過一次後就不能再次使用了。如果玩家沒有拿到聖杯或十字架，系統會提示你說你不夠虔誠，代表你會需要去找神父拿這些物品。

```
May the God bless you.  
You've accepted God's blessing, you are feeling very good now.  
  
You are not religious enough. Go back.
```

• Game Logic

整個遊戲的主要流程為以下：

1. startGame()

- 1) 判斷玩家是否有存檔：`record.loadFromFile()`
 - 1) True：詢問玩家是否要讀檔
 - 1) 是：從上次遊玩的地放續
 - 2) 否：將舊檔案紀錄刪除並開始新遊戲
 - 2) False：開始新遊戲（建立房間物件、建立角色）

2. checkGameLogic()

- 1) 判斷玩家是否死亡：`checkIsDead()`
 - 1) True：回傳True
 - 2) False：回傳所在房間的`isExit`(只有通關後才為1)

3. chooseAction()

如果`checkGameLogic()`為False就會不斷執行`chooseAction()`

- 1) 判斷是否處於特殊房間，並執行對應功能
- 2) 判斷是否有地圖，並展示地圖
- 3) 判斷是否遇到怪物，並戰鬥（`triggerEvent()`）
- 4) 判斷是否有NPC，並交易（`triggerEvent()`）
- 5) 判斷是否有掉落物品，並拾起（`triggerEvent()`）
- 6) 展示玩家狀態
- 7) 移動
- 8) 更換、穿上裝備
- 9) 離開遊戲

要注意的是若 3)判斷怪物存在後，4,5,6的功能並無法使用。

若玩家擊敗 Boss，玩家所在的房間`isExit`值會設為1，當`checkGameLogic() = true` 時，會判斷 `isExit` 值，如果是1的話則玩家勝利，如果是0的話則遊戲失敗。

• Dungeon.cpp / Dungeon.h

- **runDungeon()**
 - 執行遊戲的函數，負責整個遊戲的運行。包含一開始的startGame以及chooseAction等函式，最後會輸出遊戲結束時的特殊畫面。
- **startGame()**
 - 開始遊戲，判斷玩家是否要舊紀錄。方法是直接讀取"Save_Player.txt"，如果讀出來的結果玩家名為空字串則視為沒有舊檔案（該文件為空白）。如果要開始新遊戲，則會建立玩家以及房間資訊，使用到createPlayer()及createMap()。另外會輸出一些遊戲須知，讓玩家可以更清楚整個遊戲規則。
- **createPlayer()**
 - 建立角色資訊。玩家可以輸入名字，且可以有空格符號，使用的是getline()來讀取。第二個可以選擇職業，不同職業有不同能力值，會顯示給玩家看。輸入完後會呼叫Player的Constructor來建立一個物件。
- **createMap()**
 - 建立地圖。先建立好每個房間的物品（一個Object的陣列），如果有NPC的話，要另外建立該NPC的商品（一個Item的陣列），最後建立好每一個房間的物品後。再去一次建立每一個房間，接著再去設定有相鄰的房間的指標。
- **chooseAction()**
 - 處理玩家要執行的行動，方法是使用各種if迴圈判斷目前玩家的房間狀態，包含是否有特殊事件或是遇到怪物，亦或是使否有地圖等等，並初始化一個字元為'A'，每多一個選項就將其加一，並用一個map記錄該選項所對應的動作，動作的紀錄是使用另一個代表該行動的字元（例如移動就是' m '、打怪就是' f'...），最後根據玩家輸入所對應的字元去做個別的行動（使用switch迴圈）。
- **checkGameLogic()**
 - 讓遊戲終止的條件。如果玩家死亡的話就會回傳true，代表遊戲終止，否則要玩家所處的房間isExit值設為1才會停止。
- **handleMovement()**
 - 處理玩家的移動行為，會判斷玩家所在的房間的鄰居為哪幾個方位，如果是NULL的話不會輸出該選項。
- **handleEvent()**
 - 處理各種triggerEvent的事件，該函數會有一個Object類型的指標，接著我們就會呼叫這個指標的triggerEvent()，即可根據所指向的物件類型來做動作。
- **handleMap()**
 - 展示目前所在的位置地圖。會利用getCurrentRoom()判斷玩家的位置，並根據其結果去印出地圖。

• Object.h / Object.cpp

- 為一個抽象類別，擁有pure virtual function
- 提供 triggerEvent 這個 virtual function 來給 GameCharacter 以及 Item 繼承
- 提供基本資料：name, tag，讓其子類別都可以使用，也可以互相引用

• Item.h / Item.cpp

- 有HP, MP, ATK, DEF這四項數值，可以讓玩家的基本數值提升
- 使用Tag來標示該物品是何種類型（武器、裝備、鑰匙等等）
- 有 virtual function : triggerEvent，當玩家撿起房間的物品時觸發，會將此物品放入玩家的背包中。

• GameCharacter.h / GameCharacter.cpp

- 有最大血量、目前血量、最大魔力、目前魔力、攻擊、防禦六項數值給 Player, Monster, NPC 繼承
- **checkIsDead()**
 - 依照角色目前血量確認角色是否死亡。
- **takeDamage()**
 - 造成角色傷害
- **healing()**
 - 回復角色血量
- **checkEnoughMagic()**
 - 確認角色是否有足夠技能去施展技能
- **useMagic()**
 - 使用魔力施展技能，扣除魔力

• NPC.h / NPC.cpp

- 有各自的說明台詞，有些則是會根據你的職業做出不同回應。
- 有各自的商品 (`vector<pair<int,Item>>`)，是一個pair類型的vector，第一個數據是儲存該商品的價錢，第二個才是該商品
- **listCommodity :**
 - 列出所有商品的名稱、數值，以及金額
- **triggerEvent() :**
 - 進行整個交易的過程，包含輸出台詞、列出商品、確認金錢是否足夠等等

• Player.cpp / Player.h

- 新增 `private data : int coin` 作為金錢，可以用來交易或是轉蛋
- 新增 `private data : Skill skill[3]` 作為技能，分別對應一技能、二技能以及大招
- 新增 `private data : Item weapon` 作為武器，因為每個職業武器不同故與裝備隔開
- 新增 `private data : pair<string, string> occupation` 作為職業，第一個資料是職業名稱，第二個武器名稱，在設定時是一個pair一起設定的，比較方便。
- 將 `inventory` 拆為 `map<string, Item> equipment` 以及`vector<Item> backpack`，`equipment`為身上穿的裝，且有固定的類型，因此使用`map`來記錄比較方便，其他的物品及沒有穿的裝則收在`backpack`中。
- `addItem(Item)`
 - 將撿到的物品或是購買的物品收到背包
- `popItem(string)`
 - 將某指定名稱的物品從背包移除
- `equip()`
 - 會詢問玩家想換哪個部分的裝備，並去判斷使否有符合的裝備，有的話就輸出選項，接著再根據玩家的選擇去更換 / 穿上裝備。
- `changeWeapon() / changeEquipment()`
 - 會根據輸入判斷該部位使否已經穿著了其他裝備，如果有的話就將其收到背包中，沒有的話就直接把新物品穿上去。
 - 把玩家的基礎能力數值增加上裝備的數值。
- `changeRoom()`
 - 將玩家目前的位置更改到輸入的房間。
- `increaseState()`
 - 將玩家的各數值提升
- `checkMap()`
 - 檢查玩家是否有「Map」這個物品
- `showStatus()`
 - 顯示玩家的各種能力值
- `showEquipment()`
 - 顯示玩家目前穿著的裝備
- `showBackpack()`
 - 顯示玩家背包所有的物品
- `initializeSkill()`
 - 根據玩家的職業去初始化其技能。
- `triggerEvent()`
 - 顯示玩家的各項數值及裝備、物品

• Monster.h / Monster.cpp

- **triggerEvent()**:

- 進行整個戰鬥的過程。詳細過程在先前已經提過在此不多贅述。

• Skill.h / Skill.cpp

- 有**skillName**, **script**, **MP**, **type**等各項資料

- **Script**為使用技能會輸出的台詞

- **MP**為使用該技能所需的魔力量

- **Type**為一個**pair**類型的參數，第一個參數為**string**，記錄這個技能屬於哪種類型（**ATKup**, **DEFup**, **Heal**, **Damage**）共四種，第二個數值是其對應參數量，例如說**type**若是（“**Damage**”, 50），意思是該技能可以額外造成怪物50點傷害的意思（另需加上基本攻擊數值）

• Room.h / Room.cpp

- 有指向上下左右鄰居房間的指標 (**Room***)

- 每個房間有**index**以及**isExit**

- **Index**是用來標記特定房間用的，玩家不會知道目前所處於的房間編號，該資料用途只是用來設定特殊事件用。

- **isExit**則是判斷遊戲勝利機制的條件，如果玩家打贏boss的話會設為1。

- 總共有十六個房間，存在一個**Boss**以及一個**小Boss**，另外還有陷阱房、轉蛋機、以及祭壇，還有一間所著的秘密房間（貞德在裡面，需要Saber職階的角色）。

• Record.h / Record.cpp

- 用來儲存遊戲進度的Class

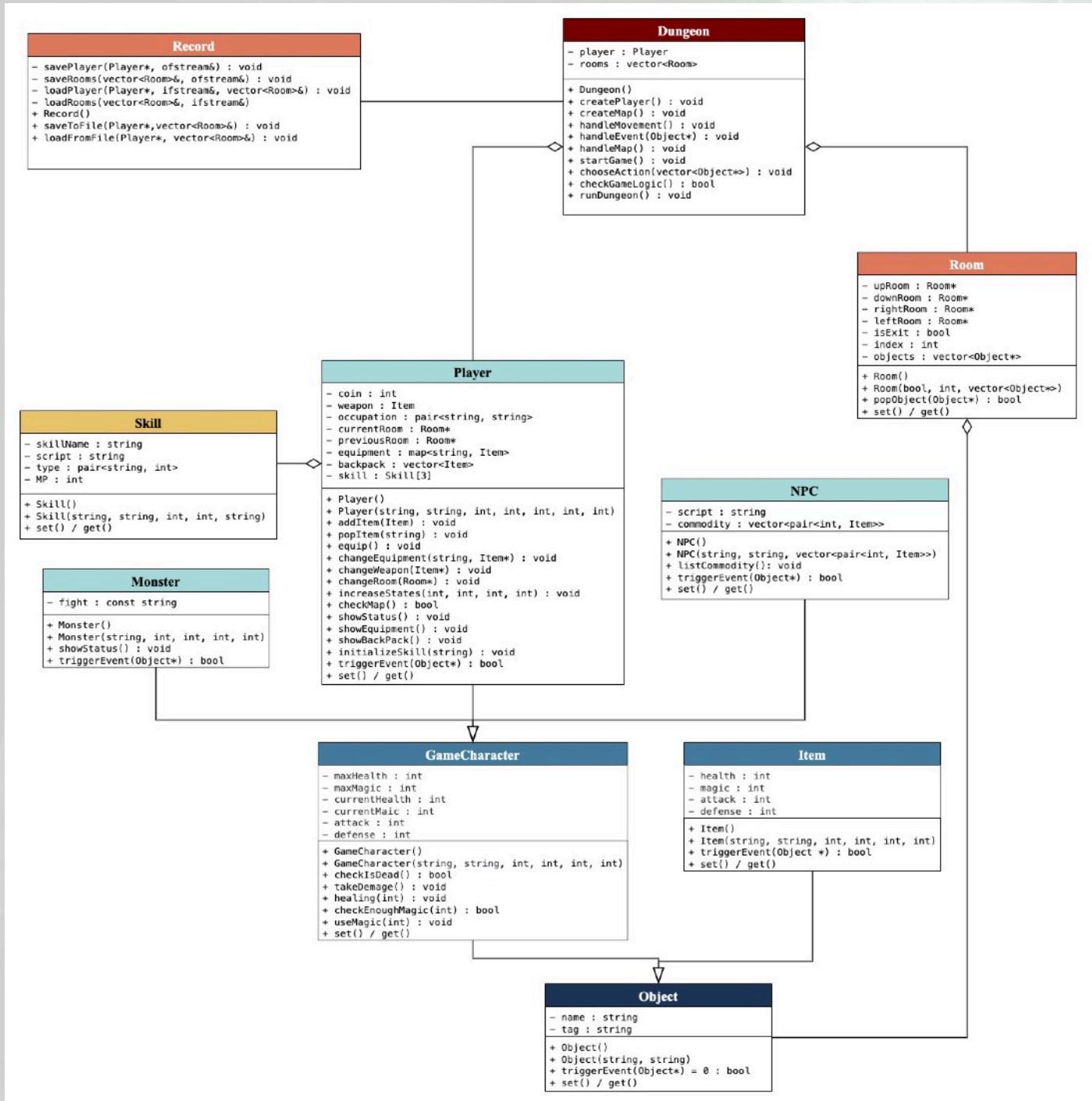
- **savePlayer()** / **loadPlayer()**

- 將玩家的各種能力數值、裝備、物品全部輸出到檔案”Save_Player.txt”中，並在玩家要讀取的時候將資訊提取出來

- **saveRooms()** / **loadRooms()**

- 將地圖各房間的物件（包括**Monster**, **NPC**, **Item**），的各項數值、商品、血量等輸出到檔案”Save_Rooms.txt”中，並在玩家要讀取的時候將資訊提取出來

• Class UML



• Discussion

1. 遊戲感覺太單調，因為我是採用固定的地圖，因此每個地方會有的物件以及掉落的物品都很單一，多玩幾次就不怎麼新奇了。
2. 顯示地圖的方式太暴力，窮舉所有可能性這個方法蠻笨的，很好奇市面上的大型遊戲怎麼做到顯示玩家目前位置的地圖的。
3. 職業的特色感覺相似度太高，除了技能名稱和畫面以外，好像沒有其他特別的差異了，感覺可以再做一些更有特色的事，比如說像貞德的那個彩蛋可以再多一點。
4. 如果之後有時間再增加新功能的話，會想要讓地圖更隨機一點，以及依據遊戲難度去生成不同的地圖。商人的

• Conclusion

這份作業總共花的時間還挺長的，以一份作業的loading來說絕對算是多的，但雖然製作過程挺辛苦的，但我卻覺得十分有趣（在做完之後）。在一開始寫架構的時候還覺得很躁，等成品真的出來真是滿滿的成就感，而且以後應該會對遊戲工程師充滿敬佩吧。另外，寫完之後也讓我對物件導向的整個概念更清楚了，實作過多型和繼承的功能後，可以很深刻的體會這兩個功能在寫大型專案時的重要以及方便性，真的可以有效率很多。總而言之，設計遊戲是個很有趣的經驗，以後有機會的還真想學習更厲害的技術，像是在Unity平台上開發等等，還有很多值得學習的事！