**I. Introduction**

本程式為以C++語言所構成之單人文字冒險遊戲，運用物件打包、物件繼承、Virtual function等特性進行實作；使用者在遊戲中扮演地下城冒險者，透過鍵盤輸入來控制遊戲進行，本程式除了**達成所有要求之基本功能**，為進一步增加遊戲趣味性再加入了額外功能，報告中將詳細介紹。

**II. Implementation**

**1. 程式執行概要**

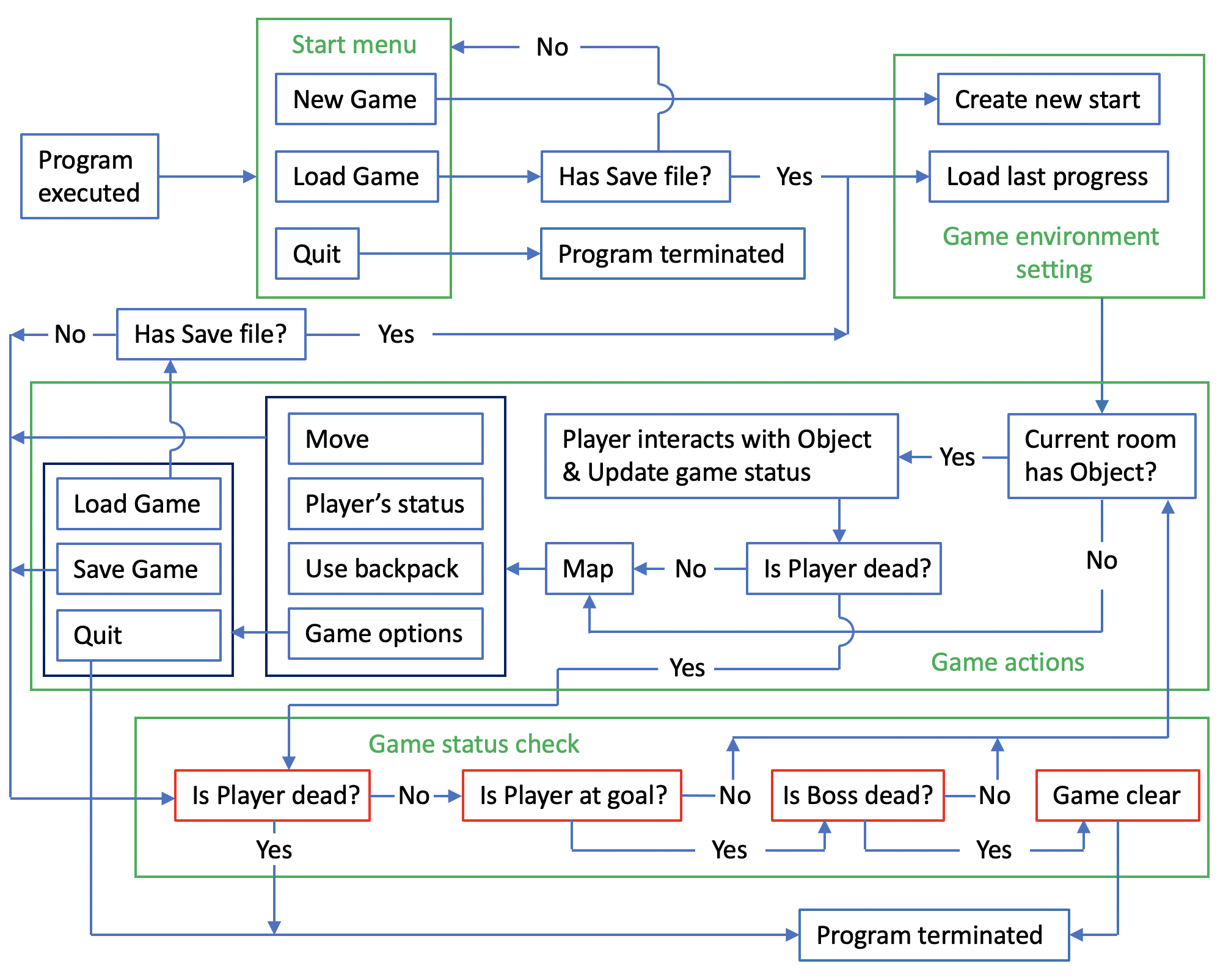


圖1: 程式運作流程圖，箭頭指向代表程式進行方向

圖1為程式運作流程圖，主要包含4項功能：**開始選單(Start menu)**、**遊戲環境參數設定(Game environment setting)**、**遊戲動作(Game actions)**、**遊戲狀態檢查(Game status check)**，下面分別對以上功能介紹：

**(a) 開始選單 & 遊戲環境參數設定：**

開始選單為程式執行後讓使用者選擇哪種遊戲環境參數設定；選擇新遊戲(New Game)的情況下，會指示程式以預設數值來設定，大多數為固定數值，一些物件參數使用了隨機函數來生成；選擇讀檔(Load Game)的情況，會先判別是否有存檔資料(Has save file?)，有則用存檔數值來設定遊戲環境，沒有則讓使用者重新在開始選單做選擇；選擇離開(Quit)則會讓程式直接終止。

**(b) 遊戲動作：**

此功能主導了大部分遊戲的進行；首先會檢查角色(Player)目前房間是否含有物件(Object)，有物件的情況下強制使角色與物件互動；為了單純處理，只讓每個房間最多含一個物件，物件包含三種：怪物(Monster)、寶箱(Chest)、商店(Shop)，在對不同物件與不同互動場景下，角色與物件的數值狀態均進行更新；互動後最重要的是檢查角色是否死亡(Is Player dead?)，是的話無需進行後續動作，直接轉移至遊戲狀態檢查。

在沒有物件或角色為死亡的情況下，會先顯示地圖(Map)，讓使用者在後續選擇移動方向較為便利，再來讓使用者選擇移動(Move)、顯示角色狀態(Player’s status)、使用背包(Use backpack)、遊戲選單(Game options)，選擇選項後會完成對應動作，再轉移至遊戲狀態檢查；遊戲選單中含有存檔(Save game)、讀檔(Load game)、離開(Quit)三個選項；存檔無論成功與否，均會轉移至遊戲狀態檢查；讀檔成功會轉至遊戲環境參數設定，失敗則轉至遊戲狀態檢查；離開是直接終止程式。

**(c) 遊戲狀態檢查：**

此功能用來判斷與控制遊戲是否結束；遊戲結束分成通關狀態與非通關狀態，前者條件為角色擊敗最終魔王(Boss)且抵達終點房間，後者為角色死亡；功能判斷遊戲結束則會終止程式，遊戲未結束則會轉移至遊戲動作功能。

**2. 程式實作細節**

圖2呈現了本程式所有物件以及對應之Class Diagram；一個地下城冒險應含有玩家、怪物、寶箱、商店、道具與迷宮等基本要素，這裡玩家以class Player實作、怪物以class Monster實作、商店以class NPC實作，寶箱與道具以class Item實作，迷宮以class Room實作，以上物件最後都會包含在class Dungeon中生成，而控制遊戲存讀檔以class Record實作，同樣也會包含在class Dungeon中生成；最後，只要main function生成class Dungeon，再呼叫執行地下城遊戲的function，即可執行遊戲。下面會分別說明：**(a) 各class實作概要**，**(b) 基本功能實作細節**，**(c) 額外功能實作細節**。

**(a) 各class實作概要：**

**a-1. Object, Item, GameCharacter：**

Object為所有遊戲角色與道具的原型，含有共通屬性參數(attribute parameter)：name和tag，後續繼承Object的class可以再另外設定name，例如玩家的名字；tag雖然在遊戲進行中不會顯示，但可在存檔上作為辨識資訊；Object的操作方法(operation method): triggerEvent設定為pure virtual function，能讓繼承Object的class各自去override並定義自己要從事的行為，程式執行中可透過late binding自動呼叫對應class的triggerEvent，在此，Object被設計為abstract class，不可被實體化；後續繼承Object的class A想與繼承Object的class B互動，則可使用指令：A.triggerEvent(pointer B)，triggerEvent的返回值為bool，可以用來代表互動成功與否或其他。

一張含有 文字 的圖片

自動產生的描述

圖2: 程式中各物件之Class Diagram

Item繼承了Object用來生成各種道具，道具設計上是讓玩家角色的能力值提升，因此操作方法

triggerEvent內定義了玩家使用道具後，如何提升玩家角色的能力值，包含血量、攻擊力、錢、爆擊率，以上四項也就作為了Item的屬性參數(health, attack, money, criticalAttackRate)，triggerEvent的傳入參數為Player pointer；本遊戲中生成的Item分成三種：寶箱道具、回復藥水、武器，triggerEvent中透過辨識道具的Object::name來分別處理與玩家的互動。

GameCharacter繼承了Object，作為所有遊戲角色的原型，本遊戲設計了三種遊戲角色：玩家、怪物、商店；商店主要為販賣道具給玩家，不會攻擊也不會死亡消失；玩家和怪物可以互相攻擊對方，怪物死亡會消失，玩家死亡則會導向遊戲結束，因此GameCharacter的屬性參數和操作方法主要為玩家和怪物的共同屬性參數和操作方法；屬性參數包含：最大血量(maxHealth)、目前血量(currentHealth)、攻擊力(attack)、錢(money)、爆擊率(criticalAttackRate)；操作方法包含兩項：1. 決定角色是否死亡(checkIsDead)，此項檢查角色目前血量是否為0來決定角色是否死亡，2. 承受攻擊(takeDamage)，將對手的攻擊值作為傳入參數，減少角色目前血量；同樣GameCharacter被設計為abstract class，不用override triggerEvent，也不可被實體化。

**a-2. Room：**

Room沒有繼承其他class；地下城的空間是由一個個Room實體相互串接所構成，因此屬性參數中用4個Room pointer來指向被串接的Room，分別代表在上、下、左、右、四個方位的房間，對應屬性參數分別為upRoom, downRoom, leftRoom, rightRoom；一個房間設計最多含有一個道具或遊戲角色，因此以一個Object pointer(屬性參數object)指向道具或遊戲角色實體；玩家與道具或遊戲角色實體互動完後，若要消滅該實體，可執行操作方法popObject；由於每此執行程式所生成之Room實體記憶體位置並非固定，無法作為存讀檔的資訊，此問題能透過讓每個Room擁有固定編號(屬性參數index)解決；遊戲通關條件之一為玩家抵達終點Room，因此透過屬性參數isExit決定此Room是否為終點，而屬性參數isVisited決定Room的資訊是否呈現在地圖上。

**a-3. Player：**

Player繼承了GameCharacter，此為使用者主要操控之遊戲角色；第一個功能是能在地下城空間移動，因此使用2個Room pointer參數currentRoom和previousRoom，分別記錄角色目前與上一次移動的位置資訊，這兩個參數的互相搭配是為了讓戰鬥撤退功能順利執行，避免發生死迴圈；每次執行移動便會執行操作方法changeRoom

第二個功能是讓Player能透過撿寶箱道具、使用回復藥水或裝備武器來提升Player能力數值， updateStatus這個操作方法用來改變Player能力數值，傳入參數則放入使用道具的屬性參數，return值為bool，true代表Player能力數值改變成功，false則為失敗

第三個功能是查看Player能力數值，定義在操作方法triggerEvent中，基本上不用與其他物件互動，因此傳入參數為null pointer，操作方法內基本上就是呈現Object, GameCharacter, Player的屬性參數。

第四個功能為使用背包道具，由Player屬性參數backpack, backpackMaxSize和操作方法useBackpack組合達成，backpack以vector乘載Item pointer，目的是讓放入、取出、移動背包中的道具較有效率，而backpackMaxSize可以控制背包內存有道具上限；在useBackpack中，會先檢查背包內是否有道具，無則離開，有則列出所有道具讓使用者選擇，使用者可以選擇使用道具或丟棄道具。

第五個功能是控制Player等級提升，為額外功能，屬性參數level記錄Player目前等級，操作方法levelUP則控制等級提升時Player能力數值如何提升。

**a-4. Monster：**

Monster繼承了GameCharacter，作為地下城中冒險者的敵人；Monster能力數值基本上都定義在GameCharacter的屬性參數，triggerEvent定義了如何與Player對戰，傳入參數為Player pointer；對戰中可能發生事件：1. 角色攻擊(Monster對Player或Player對Monster)、2. Player查看自身狀態、3. Player使用回血道具、4. Player進行撤退；第一項事件定義在操作方法attackFunction中，傳入參數第一個GameCharacter pointer代表攻擊方，第二個GameCharacter pointer代表被攻擊方，攻擊方給予的攻擊值加入了額外功能隨機函數，有機會給對手造成2 倍傷害；第二項事件和第三項事件分別使用Player::triggerEvent和Player::useBackpack即可；第四項事件定義在操作方法retreatFunction中，同樣加入了額外功能隨機函數決定撤退成功與否，失敗的情況下Player會受到一次Monster的攻擊。

**a-5. NPC：**

NPC繼承了GameCharacter，作為地下城中的道具商店並販賣兩種商品：武器與回復藥水，因此擁有屬性參數commodity來承裝道具，與Player::backpack相同，以vector乘載Item pointer；triggerEvent定義了如何販賣道具給Player，傳入參數為Player pointer；商店與Player的互動包含了: 1. 陳列商品、2. 玩家選擇商品、3. 玩家購買商品，前兩項定義在操作方法chooseCommodity，最後一項定義在操作方法buyCommodity。

**a-6. Dungeon & Record：**

Record沒有繼承其他class，主要功能為存讀取遊戲資訊，因此由多個操作方法組成；需要存讀檔時，會分別執行saveToFile和loadToFile；前者function內部會再執行saveDungeon, savePlayer, saveRooms, saveMonster, saveItem, saveNPC，將必要遊戲資訊輸出至txt文件；在有存檔之txt文件情況下，後者function內部會執行loadDungeon, loadPlayer, loadRooms, loadMonster, loadItem, loadNPC讀取txt文件，更新既有物件之屬性參數或生成新物件。

Dungeon也沒有繼承其他class，前面介紹了Dungeon含有玩家與迷宮，迷宮中有寶箱、商店、怪物，因此用屬性參數player和rooms分別代表玩家和迷宮，前者為class Player，後者以vector乘載Room pointer；本遊戲設計的迷宮為數個房間排列成矩形，因此用屬性參數map\_row和map\_col記錄矩形邊長；前面提到，遊戲通關條件為抵達終點房間與擊敗最終魔王，這裡在生成地下城時，會使終點與魔王在不同房間，為使確認魔王生存較有效率，用了屬性參數boss\_room來存放魔王所在房間的Room pointer；魔王在地下城產生時便只會創造一個，消滅後便不會在生成，而地圖上的寶箱與魔王之外的怪物在消失後有機會再生成，透過屬性參數目前寶箱數量(currentChestNumber)、最大寶箱數量(maxChestNumber)、目前魔王之外怪物數量(currentMonsterNumber)、最大魔王之外怪物數量(maxMonsterNumber)互相搭配決定。

圖３呈現了如何透過Dungeon的操作方法構建整個遊戲，首先在main function生成Dungeon實體後，整個遊戲在runDungeon這個操作方法中進行；runDungeon內會先執行startGame設

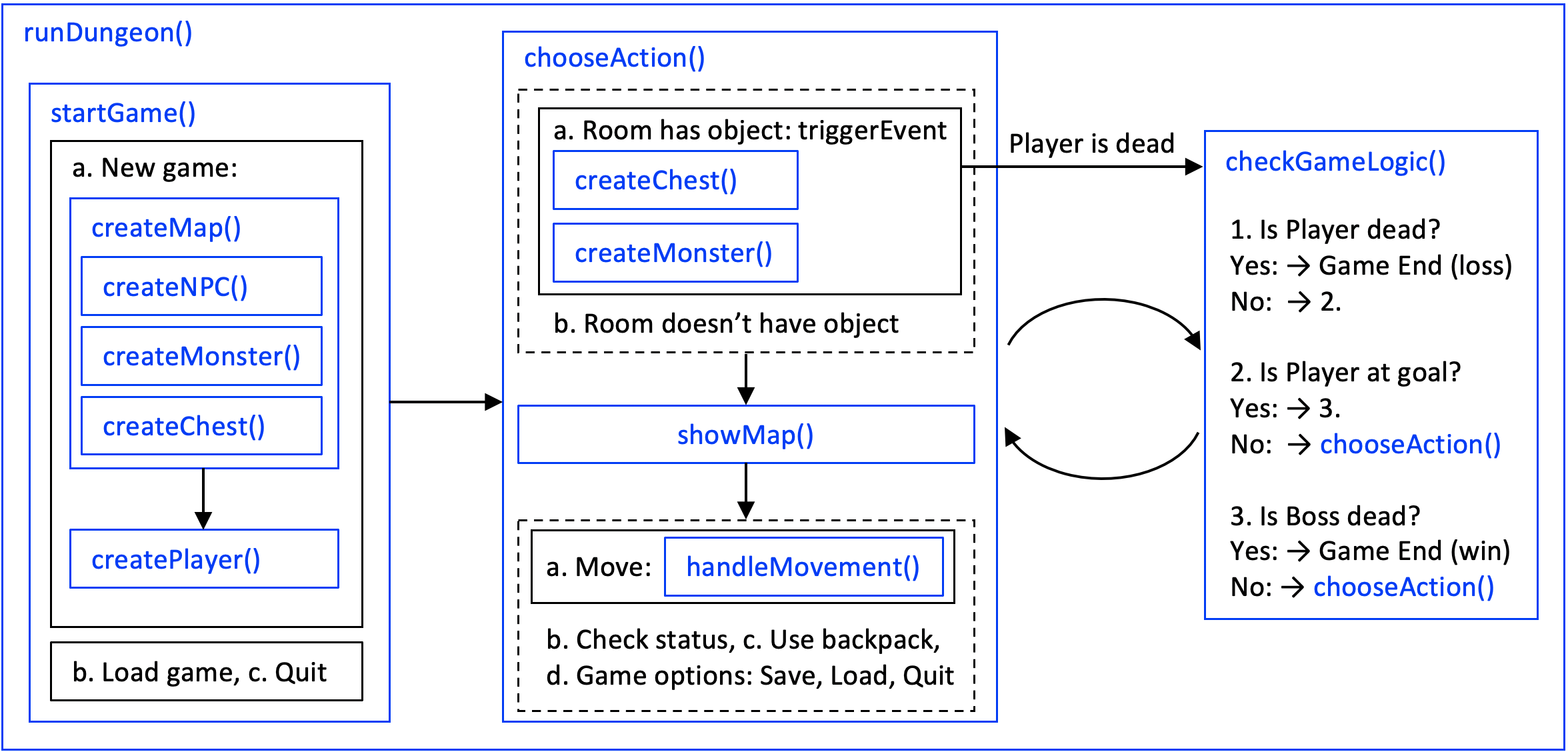


圖3: 透過class Dungeon之操作方法所構成之遊戲運行示意圖

定遊戲環境，再來進入由chooseAction和checkGameLogic共同構成的無限迴圈；使用者若在startGame中選擇新遊戲(a. New game)，程式會先執行createMap生成初始迷宮，再執行createPlayer讓使用者設定玩家角色姓名，再將角色放置於迷宮；createMap中所發生的事件依序為：1. 生成房間矩形陣列並彼此串接形成迷宮，2. 生成一個Boss放置於迷宮 3. createNPC生成含有數個道具的一個商店放置於迷宮，4. createMonster生成數個敵人放置於迷宮，5. createChest生成數個道具放置於迷宮；chooseAction主要控制遊戲的進行，對使用者的選擇做出反應；checkGameLogic會在chooseAction執行完畢後接著執行，用來判斷目前狀態是否要回到chooseAction繼續進行遊戲；在chooseAction中會先看玩家角色目前房間是否有物件，若有，即執行triggerEvent與玩家角色互動，互動後首先檢查玩家角色是否死亡，若是，則立即轉至checkGameLogic，若否，在特定條件達成下會執行createChest和createMonster再生成寶箱和敵人放置於迷宮中；

，將移動目的地房間的pointer傳入，將currentRoom覆蓋previousRoom，然後把傳入pointer覆蓋currentRoom。

；Player額外獲得的攻擊力記錄在屬性參數weaponAttack和addedAttack，細節如下：

* weaponAttack: 裝備武器獲得的攻擊力
* addedAttack: 撿寶箱道具獲得的攻擊力 + weaponAttack
* GameCharacter::attack: 玩家基礎攻擊力 + addedAttack + weaponAttack

而屬性參數weaponName用來防止Player武器重複裝備的問題，更進一步說明，武器觸發Item::triggerEvent時會檢查weaponName是否與武器的Object::name相同，不相同情況下才能裝備；若道具有回覆血量的功能(即Item::health不為0)，會先檢查GameCharacter::currentHealth是否與GameCharacter::maxHealth相同，若相同道具則無法使用，再來也設定了回血不能超出最大血量；傳入參數money設定了大於0為獲得money，小於0為花費money。

- Basic Functions

- Optional Enhancement

Results

Discussion

Conclusion