程式設計期末專案第 32 組

一、組員名單

生工二 戴琮愷 b11602054、經濟一 黃昱祺 b11303006
農藝二 許懷莊 b11601032、公衛二 卓庭榛 b11801004

二、主題設計與發想

我們的主題是棋盤回合制遊戲，靈感來源是組員玩過的，曾經佔有遊戲市場一席之地的 SLG。常見同類遊戲有聖火降魔祿、最終幻想策略版等。本組的遊戲主題是從棋盤類遊戲為起點，加上相關 RPG 的角色、道具、故事情節、角色對話等元素，完成以 RPG 角色為主的棋盤遊戲。我們也希望透過選擇較小眾的 SLG 為主題來推廣多樣化的遊戲類型，推翻小眾遊戲就是難玩的刻板印象，以實現更多元的遊戲體驗。

三、系統設計

我們的遊戲類型為 SLG (Simulation Games)，此遊戲類型就如同象棋遊戲，需要透過與對手輪流行動以及戰略的策劃，消滅對方隊伍中的所有角色成員。回合制的設計是由一方操作完自身剩餘的角色後，再輪到另一方進行相同的操作，直到一方將對手隊伍以所有角色全數擊敗，該玩家獲勝，我們遊戲目的是將對手隊伍所操縱的角色成員全數擊敗。不同於許多遊戲透過人機之間的博弈來推動遊戲，我們的遊戲設計是讓兩名玩家對打，引導玩家對棋盤遊戲有進一步的思考。在這份專案中我們用到了 Inheritance、Polymorphism、Operator Overloading、Exception Handling 這些概念進行實作。

- 角色（character）

  我們將許多 RPG 元素加進遊戲中，玩家操縱的每位角色都有自己的特點(攻擊力、攻擊範圍、移動範圍等數值)。遊戲當中，有一個 pure abstract class 主類別(character)。代表各種角色共同基本資料及函數。它擁有特定的屬性，包括血量、攻擊力、防禦力等。

  - ✓ 成員變數:

    symbol: 角色的符號、name: 角色的名稱。
    origin_healthPoints: 角色的原始生命值。
    current_healthPoints: 角色的當前生命值。
    attackPoints: 角色的攻擊力。
    defensePoints: 角色的防禦力、speedPoints: 角色的速度(礙於時間限制並無對防禦力、速度有複雜遊戲設計)。
    x 和 y: 角色在地圖上的位置。
    canAttackIndex: 可以攻擊的索引列表。

  - ✓ 建構函數:

  提供兩個建構函數，一個是預設建構函數，另一個是接受各種屬性的建構函數。

✓ 成員函數:

char getSymbol() const、string getName() const、int get_current_HP、int get_origin_HP、int getAttackPoints、int getDefensePoints、int getSpeedPoints、int getX、int getY: 這些函數用於獲取角色的基本屬性。

void displayInfo: 顯示角色的信息於介面上供玩家檢視自身角色情況。

void takeDamage: 角色受到傷害的函數、void attack: 角色進行攻擊的函數。

virtual void move(Map& chessBoard, Team& otherTeam) = 0: 純虛擬函數，表示角色的行動，因各角色的攻擊、移動範圍不同，需要在派生類中實作。在移動角色時, 會先從 Map 上移除該角色, 然後計算新位置後再放回 Map 上。

virtual bool checkCharacter(character& character) const = 0;: 這是一個純虛擬函數，用於檢查敵方角色是否在攻擊範圍內，因各角色的攻擊範圍不同，需在派生類中實作。

void checkObstacle, void obstacleHeal, void obstaclePlus: 檢查 Map 上道具位置、道具回復血量、道具加攻擊力的函數。

void chooseToAttack: 搭配 checkChararcter 檢查 Map 上其他角色位置，有可攻擊的敵方角色時提示可攻擊。

- Team1 角色:

對應到專案資訊中 Player 這個 class，我們設置四個代表四種角色的 child class，這四個 child class 分別繼承了 character 這個 parent class 的角色基本資料以及函數，並透過與 character 實現多型，在角色初始化時，定義出各角色的基本能力數值差異，以及 move 和 checkCharacter 這兩個純虛擬函數。

四種角色基本數值:

Knight(劍士)	Wizard(法師)	Archer(射手)	Tank(坦克)
HP:180/180 Attack: 60	HP:80/80 Attack: 100	HP:80/80 Attack: 100	HP:220/220 Attack: 45
刺探、破壞對方陣型，搭配加成道具會變成最強力的角色	主力攻擊來源，但血量低，移動的容錯率低	主力攻擊來源，但血量低，移動的容錯率低	血量高，保護主力攻擊來源，或搶佔重要地圖格子

四種角色攻擊、移動範圍:

Knight	攻擊範圍為以自身為中心之十字，十字大小分別為上下左右各 2 格內之未死亡敵方隊伍角色 橫向移動之格數加縱向移動之格數5格內(包含)
Wizard	4 X 4 的方格內之未死亡敵方隊伍角色 橫向移動之格數加縱向移動之格數 6 格內(包含)
Archer	3 X 3 的方格內之未死亡敵方隊伍角色 橫向移動之格數加縱向移動之格數 4 格內(包含)
Tank	4 X 4 的方格內之未死亡敵方隊伍角色 橫向移動之格數加縱向移動之格數 6 格內(包含)

- Team2 角色

  我們的棋盤遊戲中並不會有共同敵人，所以專案資訊中的敵人(Enemy)這個 class 在我們的遊戲中對應到的就是 Team2 的角色類別。

  兩個 Team 的角色基本資料與攻擊、移動範圍都是相同的，不一樣的是 Symbol 這個 char，Team2 的角色在初始化時 Symbol 改為小寫英文字母，且都是以小寫字母顯示在地圖上。

- 道具 (Obstacle)

  對應到專案資訊中的 Item 這個 class ，我們的道具有 2 種，分別是 + 以及 * 這兩種道具。

  - ✓ 成員變數:

    char symbol：代表道具的字元符號。

    string name：道具的名稱。

    int x 和 int y：代表道具在地圖上的位置。

  - ✓ 建構函數：

    用來初始化道具的屬性，接受初始的 symbol、name、x 和 y。

  - ✓ 成員函數：

    char getSymbol() const：回傳道具的字元符號。

    int getX() 和 int getY()：回傳道具在地圖上的位置。

    void applyEffect(character& someone, Map& chessBoard)：應用道具的效果到指定的角色上。這可以包括改變角色的屬性，造成傷害，或者其他特殊效果。

+補給品道具	當有任何角色走到這一格，即搶得此道具，道具消失，角色回補 30 點血量
*增強藥劑道具	當有任何角色走到這一格內，即搶得此道具，道具消失，角色獲得永久加成 20 點攻擊力

- 棋盤地圖 (Map)

  對應到專案資訊中的 Game 這個 class，我們的地圖-Map 是一 15*15 的方陣地圖也是玩家的活動範圍。Map 負責除儲存及展示個角色與道具於地圖上的位子及狀態，如代表各角色與道具的符號，並在角色與道具被擊敗或吃掉時移除相對應的符號資料。本組利用 vector 紀錄地圖資訊，並提供相關函數進行角色與道具的展示、儲存及清除等操作。

  成員變數:

  int rows 和 int cols：地圖的行數和列數。

  vector map：用來表示地圖的二維字符向量，可能包含角色和障礙物。

  - ✓ 建構函數: 用來初始化地圖的屬性，接受初始的行數和列數。
  - ✓ 成員函數:

void draw() const：將地圖內容顯示出來，可以在控制台或圖形界面上呈現。

void cleanCharacter(character& character) 和 void cleanObstacle(Obstacle& obstacle)：用來清除地圖上的角色和道具。

void placeCharacter(const character* character) 和 void placeObstacle(const Obstacle& obstacle)：將角色和道具放置到地圖上。

bool isCellOccupied(int x, int y) const：檢查地圖上特定座標是否已經被角色佔據。

- 隊伍(Team)

為了區分出兩隊的成員，我們效仿了作業十的 Zoo，可以讓我們方便的管理一個隊伍的運作，也能幫助我們在分清楚敵我雙方角色的執行動作。一隊的角色數是 3 人，而同一種角色種類在同一隊中最多只能出現 2 人。

- ✓ 成員變數:

  vector members：包含所有隊伍成員的向量。

  bool teamMark：一個標誌，用於區分出是第一隊或第二隊。

- ✓ 建構函數: 描述 Team 類別的建構和解構過程。
- ✓ 成員函數:

  size_t getSize() const：取得隊伍成員的數量。

  character& operator[]：實作 operator overloading，通過索引訪問隊伍成員。

  vector getMembers() const：取得所有隊伍成員的參考。

  void setTeamMark()：設置隊伍的標誌。

  以及一系列的 void addClassXXX(int x, int y)函數：用於將特定類型的角色添加到隊伍中。

  bool getteamMark()：取得隊伍標誌的狀態。

  bool allMembersDead()：檢查是否所有隊伍成員都已經死亡。

- 輸入或移動錯誤處理機制:

在 exception.cpp 與.h 中我們實作了 try-catch (Exception Handling)來防止使用者輸入不符合規定的角色名稱、角色數量或角色佔位等。

InvalidInputException: 在執行角色動作時，若玩家輸入的不是 move 或 attack ，跳出錯誤訊息讓玩家重新輸入

OccupiedCellException:在角色移動時檢查欲移動的格子是否已有角色佔領，有的話跳出錯誤訊息讓玩家重新輸入

OutOfBoundsException:在角色移動時檢查欲移動的格子是否超出地圖邊界範圍，超出的話跳出錯誤訊息讓玩家重新輸入

InvalidCharacterException: 初始佈陣時檢查玩家輸入所選角色名時是否有錯誤，如果不是規定的角色名稱，跳出錯誤提醒玩家重新輸入。

InvalidQuantityException:檢查玩家加隊伍角色時輸入角色數量超過規定數量 2 或小於等於 0，跳出錯誤提醒玩家重新輸入。並用於提醒玩家還可以加入多少角色

InvalidLocationException:我們規定玩家只能在初始布陣時將角色放在角落，若玩家輸入 x、y 座標不符規定位置或超出地圖範圍，跳出錯誤訊息提醒玩家重新輸入。

我們的設計讓玩家能夠自由決定各自隊伍中各角色的起始站位，先手的玩家有先選格子的優勢，而後手的玩家則可以觀察先手擺放陣容再決定自己角色的佈陣，我們認為這樣設計可以做出自由度更高，遊戲戰局更多樣化的系統。

而地圖中的道具可說是整個遊戲局勢的關鍵，可能恢復足夠讓你擊倒剩下的敵人的血量，可能會提升攻擊力使 Knight 可以秒殺 Archer、Wizard。有了道具，玩家所要思考的東西不再只是如何攻擊對手角色，同時還要阻止對手獲取地圖上的物件，需要思考的戰略變得更多，場上的一舉一動都成為影響成敗的要素，整個棋盤上就是智力拚搏的戰場。

四、演算法

1. 遊戲回合制以及遊戲結束方式

初始化:

- 將地圖初始化，在地圖上設置隨機座標的特殊道具
- 顯示地圖於介面上讓玩家可以思考佔位座標
- 創建兩支隊伍 team1 以及 team2，再根據使用者的輸入選擇不同的角色類型，數量，位置座標，加入隊伍以及地圖當中，並顯示於介面上
- 生成地圖後，印出關於遊戲背景故事的文字敘述，讓玩家可以代入劇情

遊戲循環:

- 遊戲按照回合進行，team1 角色全部執行完動作後才輪到 team2，team2 中角色全數移動完才換下一回合。
- 在玩家選擇輸入角色要執行的動作時，對於當前行動的角色進行下列動作:
  - ✓ 顯示角色資訊(current_HP，current_position，attackPoints)
  - ✓ 顯示可供攻擊的目標。(void character ::chooseToAttack)
  - ✓ 允許玩家選擇移動角色或發動攻擊。(void character:: move)
    此時會有提示語與詢問要攻擊還是移動，使用者若輸入 move，則需再輸入 x y(移動步數)，若輸入 attack，則需再輸入想攻擊之敵方角色的編號
    若選擇移動，確認準備移動的座標是在規定範圍中(符合角色移動步數限制且不超出地圖)，並確認該座標上沒有其他角色(bool IsCellOccupied)，最後移動該角色至指定座標。
  - ✓ 檢查是否碰到特殊道具，並確認道具效果，隨即將特殊物品於地圖上清除。
  - ✓ 對於兩支隊伍的每個角色依次重複這些步驟

角色對話訊息:

- 在特定的回合時，顯示特定角色之間的對話，營造遊戲緊張氛圍。

遊戲結束:

- 遊戲持續到 team1 或 team2 其中一隊的所有角色都被擊敗為止(當血量歸 0 時即視為擊敗)。
- 當遊戲結束時，就跳出遊戲結束的訊息以及結尾的故事情節。

2. 角色移動函數的計算方式

目的及整體邏輯:

角色移動的主要目的在遊戲地圖上調整角色的位置，以達成戰術目標或規避敵人攻擊。而移動的邏輯涉及玩家輸入、合法性檢查、目標位置計算和實際移動執行。

使用者輸入:

當玩家輸入 move 時觸發角色移動，玩家還需要輸入欲移動的座標偏移量 (moveX, moveY)。

移動範圍檢查及目標位置檢查:

移動範圍受到不同角色最大移動距離的限制，各角色 class 中的 move 函數都有不同的移動最大量為 const int MAX_MOVE_SUM ，則在檢查步驟時就會確保總移動距離不超過移動最大量，且同時保證角色至少移動了一格。

計算目標位置(targetX, targetY)為當前位置(x,y)與移動偏移量(moveX,moveY)的和。

檢查目標位置是否在地圖範圍內，且是否已經被其他角色佔據。

移動執行:

更新角色的座標位置(x, y)為目標位置 (targetX, targetY)。確保地圖上反映出角色的新位置，並確保該位置不再處於空白狀態。

錯誤處理:

處理可能的錯誤情況，保證遊戲的穩定執行。使用 try-catch 機制處理可能的 OutOfBoundsException 和 OccupiedCellException，提供清晰的錯誤訊息，以協助使用者理解錯誤的來源。

3. 角色攻擊範圍的計算方式

目的與整體邏輯:

角色攻擊範圍計算的主要目的是確定角色能夠攻擊到的敵對目標，該演算法通過檢查其他角色的位置和條件，確定哪些角色是攻擊範圍內的目標。

範圍計算:

角色攻擊範圍是以格子數量為單位進行計算的，這裡使用的是格子距離而非實際距離。通常是以角色當前位置為中心，在特定範圍內的格子被視為攻擊範圍，透過座標差的絕對值，確定其他角色的相對位置是否在攻擊範圍內。

目標選擇與結果返回:

使用 checkCharacter 函數確保目標是符合攻擊條件的對象。排除已死亡的目標，以確保攻擊僅針對活著的敵對角色。

將符合攻擊條件的目標的索引儲存在 canAttackIndex 中，以供後續使用，如果存在可攻擊的目標，將其顯示出來，提供給使用者參考。

4. 道具加成的方式

這裡的加成演算法針對兩種道具類型：補給品（supplement）和增強藥劑（plus）。

演算法流程:

檢查道具類型。

根據道具類型執行相應的效果。

更新角色的生命力或屬性值。

補給品(supplyment):

當角色與補給品交互時，觸發 applyEffect 函數。使用 obstacleHeal 函數實現生命力的回復。

增強藥劑(plus):

當角色與增益藥劑交互時，同樣觸發 applyEffect 函數。使用 obstaclePlus 函數實現攻擊力數值的提升。

總結:

我們的遊戲演算法圍繞在管理回合、管理該怎麼處理角色的行動(移動以及攻擊)、特殊道具的效果以及玩家輸入的內容展開。遊戲會不斷重複以上的算法，直到有其中一方達到獲勝的條件。

五、分工方式

許懷莊:程式碼撰寫、程式結構討論、程式碼後期修訂、專案影片拍攝上傳。

戴琮愷:主題發想、角色 class 雛形構想、程式結構討論、切分標頭檔。

黃昱祺:程式碼撰寫、程式結構討論、程式碼後期修訂、專案報告、遊戲展示影片拍攝、專案報告撰寫。

卓庭榛:程式碼撰寫、程式結構討論、程式碼後期修訂、專案影片拍攝。

六、組員心得

1. 戴琮愷:其實在做這次專案之前，我一直都有一個自己做遊戲的夢想，但沒想到第一次做遊戲就是在時間那麼急迫的情況之下去製作的，如果說在這次的專案當中遇到最大的困難的話，應該可以說是給我們的時間真的太急迫，而且課堂需要顧的東西真的太多了，其實真的讓我們很多的想法都被壓縮很難以實現，沒辦法百分百的去熟悉我們手邊正在使用的工具，除了這點之外我覺得跟不同的人一起製作遊戲分工的部分也是很困難的一個點，我們很難去了解一個人究竟該去做多少工作才是適當的，而且我們在離專案死線越來越近的時候就開始對自己做出來的東西有一點的焦慮，因為其實跟我們剛開始設計的東西是有一段不小的差距的，但我們還是希望可以把我們剛開始的想法給實現，所以中途也是不斷地修改，因為這同時也會是一份攸關我們成績的作業，所

以在製作遊戲的過程，不免會在心裡有點壓力造成同儕之間的不愉快，但總而言之，做出東西的成就感確實是無可比擬的啦。

我也想回頭說說我在這份專案當中到底學到了什麼，其實在製作專案的剛開始，應該說我有點好高騖遠嗎，我們剛開始是希望實現圖形化介面的，那時候我甚至去學了一點 SDL，但到最後發現靠期末的這一點時間，我很難去掌握這個工具，只能說是比較可惜的，再來學習到的東西應該可以說是怎麼跟他人合作的技巧吧，就如同我在上一段所說的，剛開始大家到底該做多少工作是比較難知道的，畢竟我們這整組幾乎都是從程式小白開始學習的，而我學習到最多的應該可以說是要怎麼構想一份程式，然後要怎麼去切割整份完整的程式變成好幾個方便於維修的小程式碼，我的心得大致上就到這邊了。

2. 許懷莊:我主要負責遊戲中的道具部分，這是一個相對較晚才開始著手的工作。這個經驗讓我深刻體會到，要與團隊成員的已有程式碼進行整合是一項挑戰。我們的遊戲流程相比其他遊戲更為複雜，因此需要編寫更多的錯誤處理機制，比如 try-catch，以防止使用者操作時出現錯誤。在處理 class Team 的程式碼時，涉及了大量的指標操作，這讓錯誤的可能性大增。這次經歷使我真正意識到了程式碼模組化的重要性。另一方面，在時間分配上也是一大挑戰。我們不僅需要處理道具、角色攻擊以及觸發效果的邏輯，還要考慮螢幕顯示等問題。這些工作佔用了我們大量的時間，進而壓縮了我們調整遊戲平衡的時間。這次的經驗非常有趣，也讓我對程式設計有了更深入的理解，並體會到了團隊合作和時間管理的重要性。

3. 黃昱祺:在製作遊戲專案時，遇到了許多沒想過的問題，例如在分工後原先以為只要搞懂自己負責的部分就好，實際上卻仍要花費時間讀懂組員們寫的 code ，也需要思考如何將自己的 code 寫得更易閱讀。在 debug 時，若有人寫出了不易理解的 code ，常常還要等其他人慢慢讀懂後才能幫忙。除此之外，我們在遊戲設計上也有很多溝通討論的過程，尤其是在道具放幾個於地圖中之類的細節上的設計。組員們都有自己合理的設計理念，我們也在整合意見中學習到很多溝通、表達及統合能力。我認為這次的作品對於我們而言是仍有許多進步空間的作品，許多原先更複雜的設計想法都礙於本組時間及分工的問題，沒有很好的實作出來。不過整體而言這份遊戲對我來說還是非常有成就感的作品，看著原先的想法一步步的實現於電腦介面上，內心的滿足感油然而生，也更確立自己對程式設計的熱情與喜愛。我會帶著這次專案中學習到的溝通、團隊合作以及程式能力，以及在專案中遇到的問題經驗，更好的在日後的程式團隊合作機會中表現，並產出更符合期待的作品。

4. 卓庭榛:這是我第一次實際上與別人合作寫一份專案，在這之前，基本上都是個人作業而已，原先以為只是簡單的一人負責一點，最後再將全部拼起就好，但實際與他人合作時，發現不只需要把各自的部分做好，還需要額外花許多時間讀懂其他人的 code，因為每個人打 code 的習慣皆有所不同，理解完後在打 code 的過程中，也必須確保自己的 code 能被其他人輕易讀懂，甚至過程中還發現程式碼會因為作業系統的不同，而出現一方有 bug 但另一方沒有 bug 的問題，因為得確保所有人都能共同使用同一份程式碼，這時候就必須以整個團隊為主體去修復那個 bug。此外，意見上的撮合也是在我意料之外且花時間的點，不管是開始實作前還是在實作中都會發生各自對於該不該加某函數或物件都有自己的看法，但我們在處理這部分問題還是以最簡單的多數決處理。這次的成品我認為仍有許多地方可以變更好，但時間分配上可能因為是第一次所以沒有分的很好，下次還有機會做專案的話就可以借鑑此次的經驗，並再做出更高品質的作品了。