Reversi

資工一 張閔堯 B11902084
資工一 李冠誼 B11902090

- 工作分配

  張閔堯、李冠誼 ： 演算法設計、演算法實作、物件與介面設計

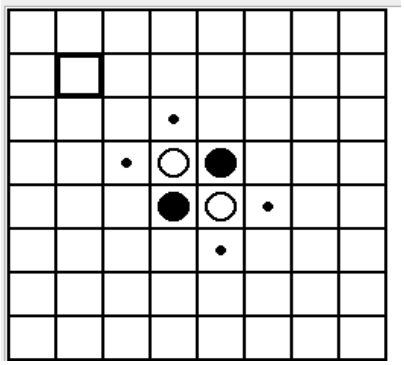
  註 :我們都在同時同地工作，時常互相討論與支援，因此工作內容相仿。

一、簡述

我們在這次的 final project 中選擇製作的是一種名為黑白棋(Reversi)的遊戲，這個遊戲是在 19 世紀由英國人發明，並在後來由長谷川五郎將其發展。遊戲規則如下(參考維基百科)

1.初始狀態

會有兩個黑棋、兩個白棋對稱的放在 8*8 的棋盤中，如下圖:



2.行棋規則

通常黑子先行。雙方輪流落子。只要落子和棋盤上任一枚己方的棋子在一條線上(橫、直、斜線皆可)夾著對方棋子，就能將對方的這些棋子轉變為我己方(翻面即可)

3.特殊情形

如果在任一位置落子都不能夾住對手的任一顆棋子，就要讓對手下子

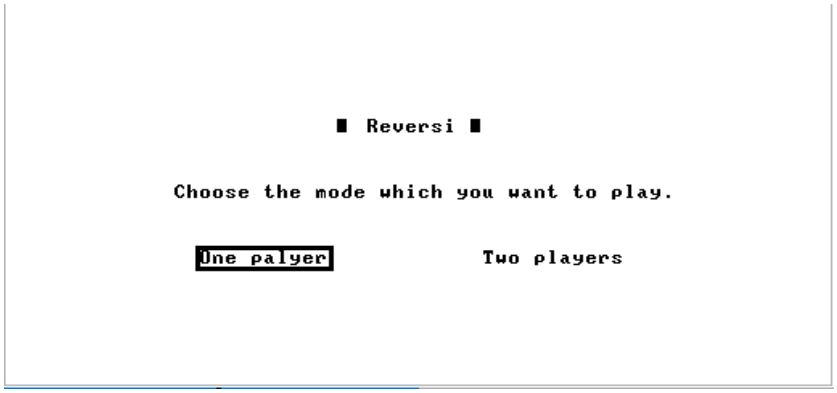
4.勝負判定

當雙方皆不能下子時，遊戲就結束，子多的一方勝。

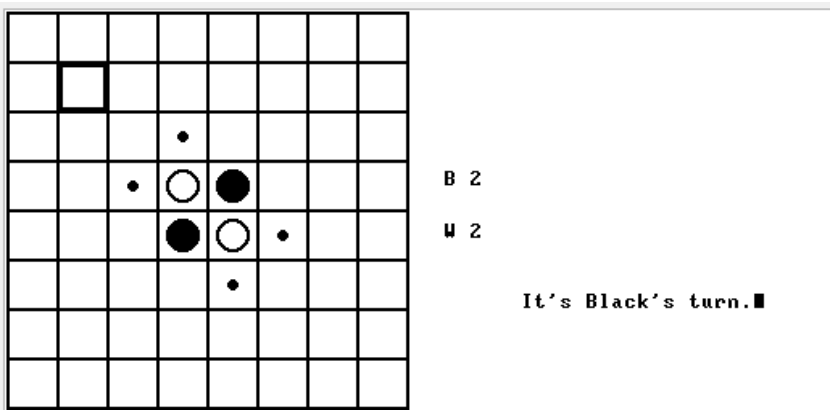
- 遊戲展示影片

https://youtu.be/-BovabkGDrY

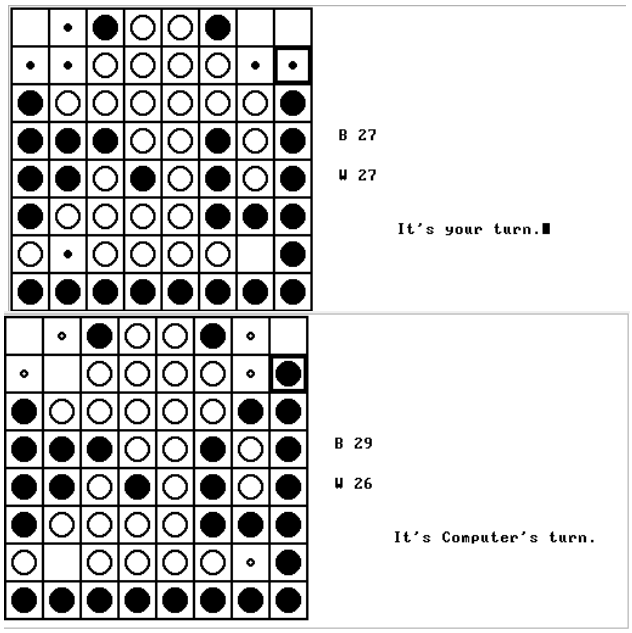
二、演示畫面



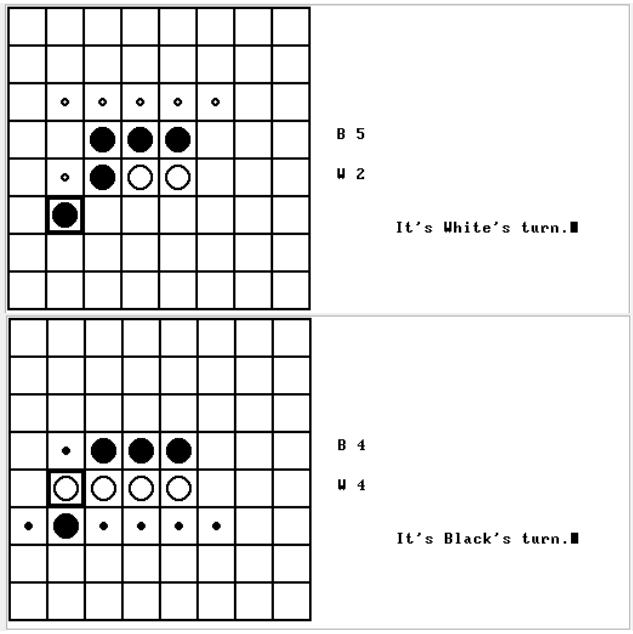
模式選擇(以左右鍵選擇，並以 enter 確定)



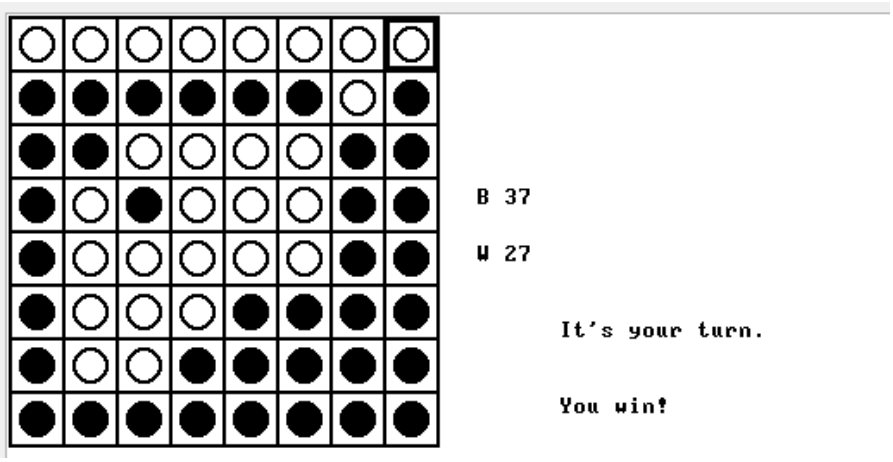
起始畫面(小圓點代表可下的方格，以方向鍵選擇並以 enter 落子)



單人模式中的玩家回合與電腦回合



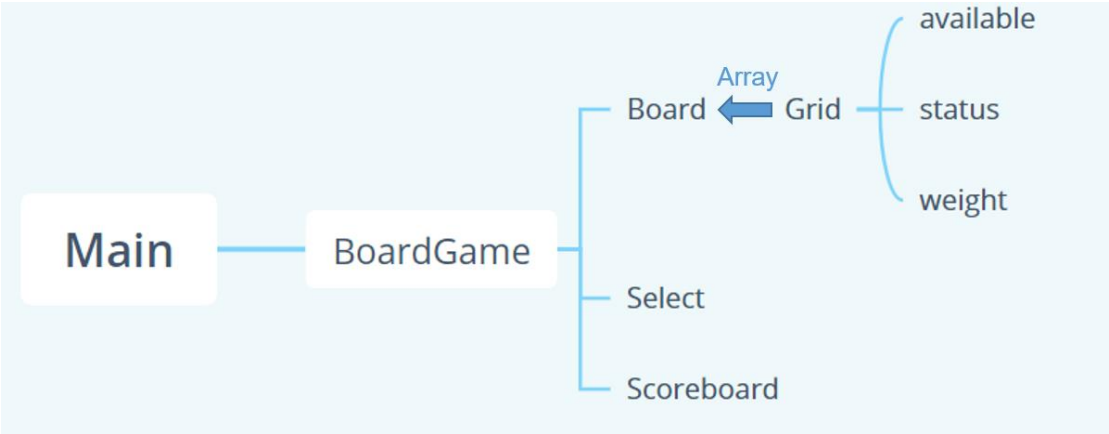
雙人模式中的黑白回合



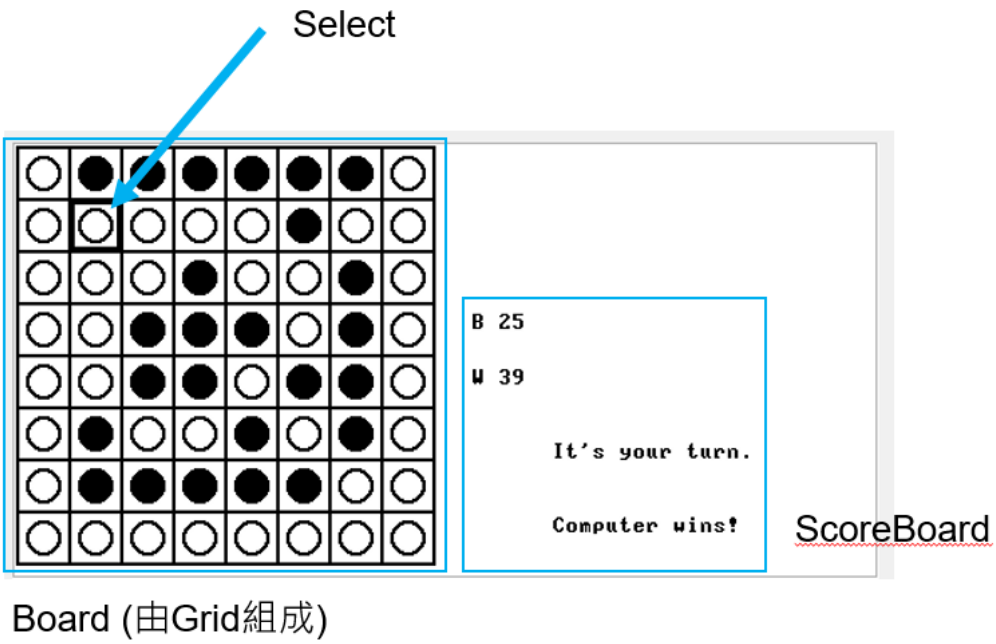
無棋可下時將顯示誰是勝利者

三、程式架構

我們的程式架構如下圖，此外，還有一個幫手函式 Draw 負責畫出各種圖形



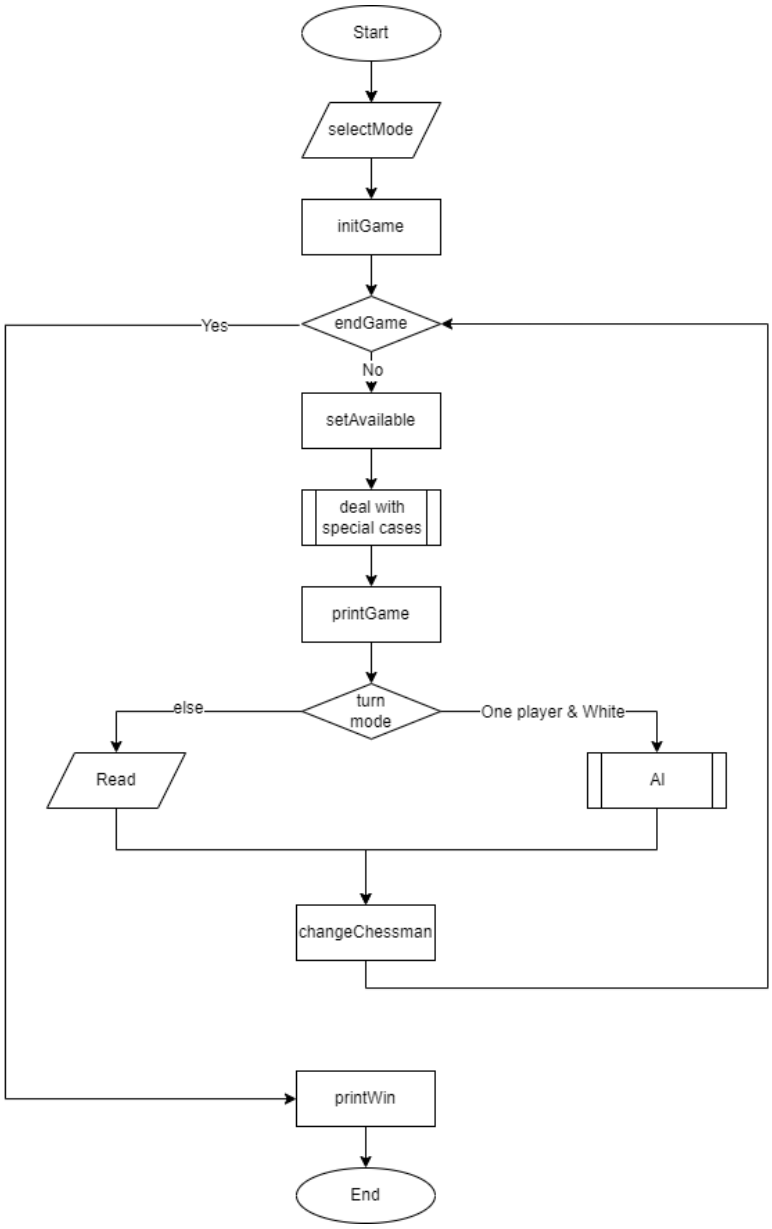
這是我們介面與物件的關係



(一) Main

Main 中有各種遊戲流程控制、使用者輸入的程式碼，並引入 BoardGame 物件，下頁圖為 Main 的流程控制。首先讓使用者選擇單人或雙人模式，接下來進入迴圈，以下為迴圈流程

1. endGame : 判斷是否已有贏家產生
2. setAvailable (BoardGame 的 method) : 計算可落子的點
3. Special Cases : 當一方無法落子，此時應直接由另一方落子
4. printGame (BoardGame 的 method) : 讀取各個物件並輸出
5. read & AI : 判斷玩家或 AI 落子，並輸出合法座標
6. changeChessman (BoardGame 的 method) : 翻轉棋子



(二) BoardGame

此物件中有許多遊戲進行時所需的重要演算法

1.AI

我們製作的 AI 策略是先依照正常情況下黑白棋盤每個位置的珍貴性與戰略性來給予權重，每次輪到 AI 下棋時，AI 會將每個點的權重乘上每個 grid 的 available * (-1) (因 available 為布林值，-1 代表可下，0 代表不可下)，並選擇運算值最高的位置落子。

57	6	50	41	42	51	8	58
5	1	33	25	26	34	2	7
49	40	21	13	14	22	35	52
48	32	20	0	0	15	27	43
47	31	19	0	0	16	28	44
56	39	24	18	17	23	36	53
9	3	38	30	29	37	4	11
59	10	55	46	45	54	12	60

AI 權值表(中間四格無法落子，權值為 0)

2.setAvailable

setAvailable 是我們用來判斷每個還未有棋子的方格是否可被落子，在此函示開始執行時，先將所有 grid 的 available 設為 false，接著對每個未落子的 grid 向外八個方位進行搜尋，若是有任一方位出現此空格與此回合玩家之棋子共同夾住對手棋子的狀況，便將此個 grid 的 available 設為 true。

3.changeChessman

changeChessman 負責的是棋子的翻面工作，其執行使用的參數是當回合玩家落子的座標和玩家顏色。運算的方法是對每個 grid 進行 setAvailable 相似的八方位運算，並將所有與自己同顏色之棋子夾住的對手棋子變為自己的顏色。

4.printGame

printGame 是我們將棋盤呈現給玩家的重要關鍵，首先它會將 8*8 的棋盤印出，接著依據每個 grid 的 available 和 status 繪製棋盤，若是此 grid 的 status 為-1 我們就印出黑棋，1 則印出白棋，若是 status 為 0 且 available 為 true，我們印出當回合玩家棋子的縮小版代表這裡是合法可落子的位置。

(三)Grid

Grid 雖然並無包含許多複雜的內容，卻是組成我們黑白棋遊戲的最基本結構，以下介紹 Grid 中所包含的變數。

1. status

status 代表當這個 Grid 是否有棋子，若是還未被下子 status 為 0，若是其上為黑子 status 為-1，為白子則為 1。

2. available

available 的值表示這個 Grid 是否是合法可落子的方格，若為 false 則不可下，true 則是可下。

3. weight

weight 是我們在實作 AI 時才加進來的變數，代表這個 Grid 在戰略上的是否該立即搶下，weight 越高代表 AI 會越重視這點。

(四)其他

1. Board

Board 這個物件由 64 個 grid 所組成，是一個代表當前棋局的二維陣列，當我們需要存取 grid 時常以 Board 中的函式來著手。

2. Draw

Draw 這個物件並不含任何 field 變數，它的主要功能是配合 Boardgame 中的 printGame 將當前的棋局畫出。

3. Scoreboard

Scoreboard 這個物件負責記錄當前的黑白棋數量，以供 Main 中的 printWin 與 Boradgame 的 printGame 輸出棋局狀況與結果。

四、遇到的困難

我們第一個遇到的困難便是我們對於 Jack 的不熟悉，使我們在實做許多我們於 C 語言中可以輕易實行的事物不斷遇到困難，像是二維陣列等，我們也花了不少時間在鑽研其使用方式。

除此之外，我們對於物件不熟悉也造成了嚴重的障礙，例如前期我們並沒有很熟 function 和 method 的差異，導致在使用時出了許多奇怪的問題。此外，在 Jack 中物件的 constructor 牽涉記憶體分配，雖然在 C 語言中我們已稍微涉略記憶體分配。但 jack 中甚至陣列也是以物件實作，因此我們在實作時花了不少時間試誤與參考範例程式碼，在其中我們常常遇到各種記憶體操作的失誤，最終透過適當 new 與 dispose 解決部分 heap overflow 的問題。我們也因為 Jack 的加減乘除不是依據正常的運算邏輯而出了許多差錯，除錯了許久才發現。

然而讓我們頭痛最久的問題是當我們在做最後的除錯時，發現我們棋下到最後幾手時會出現 heap overflow 的情況，經過了近四個小時的奮鬥，最後我們透過減少在字串輸出時對 heap 的使用解決這個問題，程式終於能正常運作。

五、學習心得

在這次的專題中，我們首次認知到了與他人一同合作製作出一個大型程式作品的感覺，以前寫程式都是自己一個人把題目想出來並想出來，但這次的專題需要與別人討論，並一同研究我們都不擅長的物件導向觀念，鑽研一個我們不熟悉的語言，完全是一種全新的體驗。

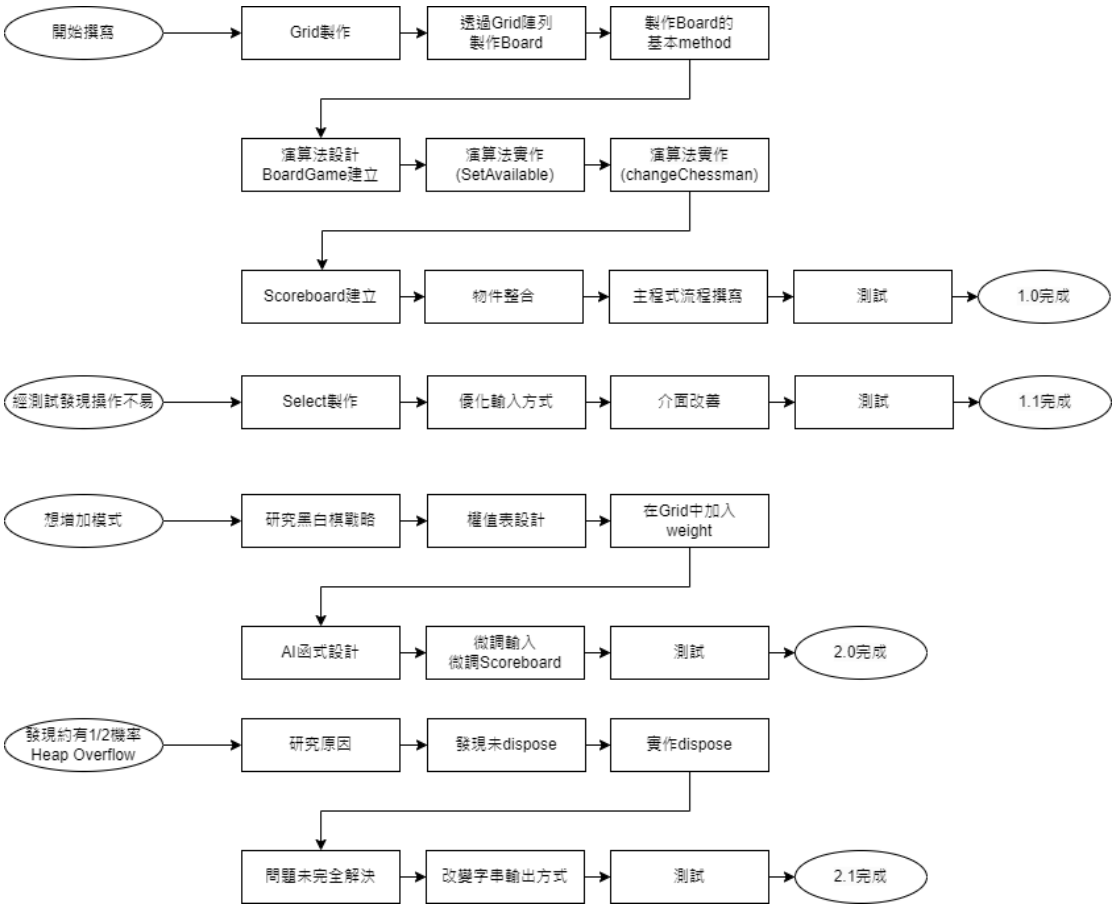
在這次專題中最令我們印象深刻的可說是物件的方便性與特性，我在製作物件時學習到物件的概念，這讓我們在合作時的分工能力得到了大大的強化，我們相互支援各物件功能實做與物件合併運用。在這個過程中，不僅能與隊友一同 debug，更能透過互相討論增加演算法的效率。

同時，我們也體認到友善 UI 對於使用者的方便性。例如在座標部分，我們原先本想採用使用者輸入座標的方式，但實際請他人一同試驗後發現這種輸入模式容易輸入錯誤的同時還費神，最後我們新增 Select

這個物件讓使用者只要用上下左右鍵即可輸入座標。一個小小的 UI 改動即可讓遊戲體驗變得更加完美，這也讓我體認到了測試的重要性，不僅可以發現 bug 更可以得到各種回饋。

六、版本歷程與製作過程

- ver 1.0 兩人模式完成
- ver 1.1 改善座標輸入
- ver 2.0 新增 AI
- ver 2.1 修復 Overflow 問題



註：

1.若程式執行到一半就強制暫停，有時 VMEmulator 會當掉

2.在單人模式中電腦下棋會有一定的延遲，這是為了能讓使用者看到電腦下棋狀況而刻意加入 Sys.wait()所致。