第 14 組 final_project

莊凱予 210510232
詹其佐 210510210
蘇泰宇 10502302
許孟翔 105023052

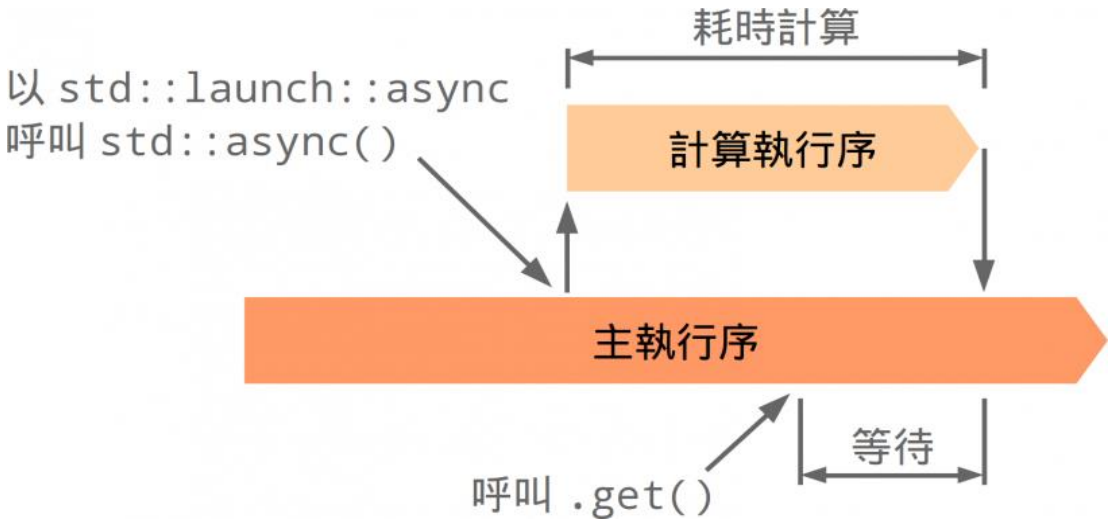
#Basic skill

● 請解釋下編譯指令，每一個參數代表的意涵（10%）

1. g++-8 代表可執行的 binary file 經由環境變數可在 BASH shell 使用 /usr/bin/g++-8 而不用在前面加路徑。
2. -std=c++17 代表決定要用 C++17 的標準去 compile。有新增一些 C++17 才能用的標準，捨去 C++17 一些以前才能用的標準或是保留某些 C++17 以前的標準。
3. -O2 代表 Optimize，效果比-O1 更好，比-O3 更差，主要加快編譯時間和 binary code 的效能。
4. -Wall 代表在編譯時啟動有關於 constructions 的 warning flag。
5. -Wextra 代表增加某些 warning flag 沒有被-Wall 涵蓋到的，例如指標在和整數 0 相比時的 warning flag，e.g. if(ptr >= 0)這種情況。
6. -fIPC 代表會產生 position independent code (PIC)，適合用於 library 做 dynamic linking，同時也避免 global offset table(GOT)的任何限制。
7. -I 代表增加 header 搜尋的路徑，根據 #include 後面是 <> 或”” ，來判斷要從編譯時當前目錄搜尋還是此 header file 的目錄搜尋，e.g.<BattleShipGame/Wrapper/Porting.h> or "AITemplate.h"。
8. -shared 代表要編譯成 shared library。
9. -o 之後的 file name 代表要生成的名字。

● 請解釋 Game.h 裡面 call 函數的功能（5%）

用 function pointer 使 function 為參數載入，之後開啟另一個執行緒(thread)跑載入的 function 並計算時間，若超過 1 秒程式則強制結束，exit(-1)代表程式不正常結束，最後如果有回傳值再做回傳。流程如下圖:



- 請解釋什麼是 Shared library，為何需要 Shared library 在 Windows 系統上有類似的東西嗎？(10%)

不同 process 之間可以共用在 memory 已經 loading 的 library，例如 main_1.cpp 和 main_2.cpp 都有 call printf() function，main_1.cpp 先執行時把 printf() call 入 RAM，之後 main_2.cpp 就可以共用已經有的 library function，可以想像，假設 main_1.cpp 和 main_2.cpp 都 load 一份 library 進來，會造成記憶體空間的浪費，也造成 binary file 佔用空間很大，和 windows 系統中 .dll file 類似。

- AI Algorithm (10%)

1.AI 進攻的時候，先打助教原始給的船的 4 個中心點，因為我們想說可能有人忘記改船的初始位置，這樣就可以直接獲勝。接著從(1,1)開始往右跳三個三個這樣打，因為船最小是 3*3 所以這樣打會比較有效率，也就是說，我們先撇除最外邊的四個邊，再來以座標 x 或 y 除 3 餘 1 的座標為優先攻擊，最後再以填滿的方式去攻擊敵方的領海，但還是不會攻擊到最外邊，因為最外邊是不可能有船的中心點的。
2.防守策略就是不移動船隻，因為我們認為在船隻被打到不能移動的情況下，沒被打到的船隻移動在 20*20 的版面上移動，覺得對防守上並沒有太顯著的效果，所以經過測試與權衡後，還是決定不移動船隻。

- 分工與進度規劃 (5%)

1.進度規劃: Gmaerunner 在還沒做出完成品前，先提供 AI 贋品，在 AI 完成得差不多後，再將 Gamerunner 與 AI 做整合以及架構上的統整後，再進行修改 AI

的內容。

2. 分工:

1. AI part : 莊凱予, 詹其佒
2. Gamerunner part: 蘇泰宇, 許孟翔
3. 解釋參數, 函數意涵, Shared library: 許孟翔
4. AI algorithm: 莊凱予 詹其佒

- 心得 (10%)

1. 莊凱予 :

在實作 AI part 的時候，剛開始再對架構還不清楚的時候，原本想說直接判斷哪邊是敵人，直接更改 gamerunner 那邊的資訊，但後來才發現如果以物件導向的概念來寫的話，就可以將 AI 視為一個物件，也就是說，每次都更改 AI 裡面的資料就好了，至於攻擊的部分，原本想採用每回合透過 gamerunner 傳進來的 Board 來做下次攻擊策略的依據，但是由於在實作時，雙方的共識有一點小落差，所以最後就採用直接像是助教的亂數打法，只是我們自己將最有可能為船核心的座標視為優先攻擊的座標，至於整體而言，最大的麻煩是，還不知道 gamerunner 的全貌，就開始進行 AI part 的部分，就會有部分是衝突的，但因為 AI 是被 call 的，所以應該是 AI 需要配合 gamerunner 的寫法才對，所以在最後的磨合後，也刪除了許多不必要的 code，才得到這個結果，也透過這個分組作業學習到很多該如何分工合作。

2. 詹其佒 :

一開始寫 AI 的時候以為只能用助教宣告的東西，所以不知道那些函數要怎麼實行，後來詢問助教才確定可以自己再多寫一些變數。在宣告 TA::Board map{20}的時候，原本不知道要用大括號，試了很久，後來問孟翔才解決。在理解每個函數時一直碰壁，通常都是在反覆去看有用到他的地方、回傳值、還有傳進去的值，來了解這些函數。原本不知道 way 是用來做甚麼的，想說它裡面就只有放地圖的點，後來上網查 shuffle 跟 seed，還有比對函數 queryWhereToHit，才知道他是選擇攻擊的點，之後我跟同學討論了一下要怎麼決定攻擊得方法，才得出最後的結論。在 callbackReportEnemy 也是用很笨的方法來判斷是不是有打到船。感覺這次理解比做的還要多，大部分都是在問問題，跟理解別人寫的 code，理解了之後才可以 debug，很感謝組員的幫忙。

3. 蘇泰宇 :

實作 gamerunner 時，孰悉 BattleShipGame 和 AIInterface 中各函數的功能及使用方式相當重要。根據遊戲的規則，並搭配這些函數的使用，即可逐步

建構出 gamerunner。過程中發現呼叫 AIInterface 中的函數時，都需要透過 call function 傳入函數名稱和參數，並對此感到有些疑惑，後來才了解 call function 其實是利用 async 函數對 AI 中的函數做平行運算，猜測是為了保證 gamerunner 的執行流暢。此外，處理 ship 的座標也有些麻煩，因為 board 用的座標和平常習慣的 x,y 座標軸不同，需要稍微注意。

4. 許孟翔 ：

其實一直以來都很想參加一些開源專案，想和他人一同開發一個專案，來練習自己 coding 的功力，因為以後在開發一個大型專案時是不可能一個人全部寫的，練習閱讀他人的 code 和自己寫出高可讀性·高維護性的代碼是非常重要的，這也是我一直所努力的目標以及喜愛寫程式的原因，這次的 final project gamerunner 的骨幹算是我建構的，然後畫張圖盡可能簡單表達這次 final project 的流程，在寫這些 code 的同時，我一直考慮是不是會有更好的寫法，組員能不能藉由我的 code 輕鬆讀懂且寫出別的部分？ 但後來聽到組員有稱讚其實滿好懂，內心其實滿開心的，也算是獲得一些肯定，再來就是 Basic skill 的部分，第一題的參數有些真的很冷門哈哈，為此還特地去查了 linux manual，還有寫另外的 code 測試這些參數到底有沒有用，e.g. -Wextra 用指標另外測試有加參數會不會過，還真的 error 不給過= =，第二題和第三題屬於作業系統會 cover 到的部分，寫起來算是滿輕鬆的，也順便複習上學期學到的相關知識，總之，感謝助教出這份有團隊合作的 project，對我學習如何 teamwork 有很大幫助。