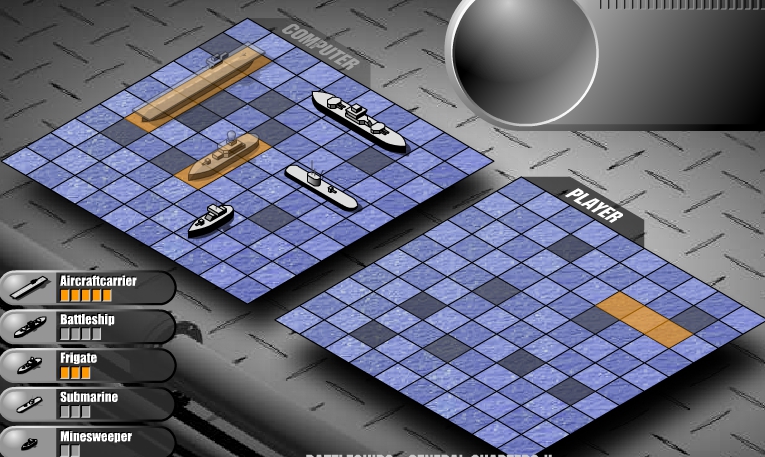
SeaBattle(海戰棋) 組員:林芷敬、張書鳴、羅孟珣

**遊戲介紹:**

遊戲中，雙方都會有一塊16\*16格的海域，以及5種長度不一的戰艦各一艘。首先，雙方須在自己的海面上佈署自己的船隻，每艘船會佔據數個格子，當所有船隻都佈署之後遊戲才正式開始。遊戲採回合制進行，雙方輪流指定對方海域中的一格進行攻擊，且攻擊後會顯示此攻擊是否命中或打偏，因此可以由此推斷接下來的進攻目標，下圖為網路上找到的遊戲畫面(不是我們的QQ)。



但不同於原版的海戰棋，我們仿造了另一個海戰棋遊戲(Cruiser)，加入了許多新的功能，我們試著讓了每一種船搭載不同的武器，並且讓同陣營的每艘船在每一回合輪流進行攻擊，而遭到擊沉的船隻將會被移除，也就是無法再使用該種船所搭載的武器，同時，在使用特殊武器進行攻擊時也將會消耗金錢，因此在遊戲後期必須注意自己所剩餘的金錢，讓遊戲的考量有更多的變化性。

船艦類型：

Cunboat：砲艦，占兩格，可使用Radar

Destroyer：驅逐艦，占三格，可使用Plasma Connon

Submarine：潛水艇，占三格，可以使用Torpedo

Cruiser：巡洋艦，占四格，可使用Missile Volley

Aircraft Carrier：航空母艦，占五格，可使用Air Strike

功能說明：

Radar(雷達)：指定一個座標，探查一個2\*2的區域是否有敵方船隻，若有則標記為o沒有則直接標記為X

Plasma Connon(電漿砲)：擊中對方可以馬上摧毀整艘船Torpedo(魚雷)：由潛艇的船頭座標發射，射擊一直線上的對方區域(擊中

目標則停止，最多攻擊五格)

Missile Volley(發射飛彈)：隨機攻擊一個4\*4區域中三個位置Air Strike(空中支援)：攻擊目標及其左上右下的格子(共三格)

**遊戲玩法:**

遊戲開始提供了4種模式，請輸入1 or 2 or 3 or 4來選擇，模式1,2,3分別有不同的金錢來對抗不同難度的電腦，而模式4則提供兩人PK。

電腦難度分為簡單和困難，兩種都只會使用普通的攻擊，但困難模式的電腦在電腦只剩一艘船之後就不會再失手，因此玩家必須速戰速決。

此外，在佈署船隻時可以選擇自動佈署或手動佈署，選自動佈署系統會幫你自動排好，若手動佈署則需要輸入”方向+座標”，如：V1a就是”垂直”的從座標”(1,a)”開始往下放，而H5i就是”水平”的從座標(5,i)往右放。

遊戲開始後，系統會排好順序讓每艘船依序進行攻擊。輸入1之後可以指定對方一格進行普通攻擊，並且不消耗金錢，而輸入2則可以使用每艘船搭載的不同武器，並消耗不等的金錢。

需注意的是，特殊武器除了Torpedo(魚雷)外都需要指定目標，若是Radar(雷達)和Missile Volley(發射飛彈)這樣指定n\*n範圍的，則視為指定左上角的座標，而Air Strike(空中支援)則比較特別，是指定中央的座標。

另外，Radar和Air Strike都要求輸入格必須”合法”(也就是棋盤內未被攻擊的格子)，而Missile Volley雖然沒有要求輸入格”合法”，但當重複擊中已被攻擊過的格子時就會無效，因此有可能出現Missile Volley實際上攻擊不到三格的情形。 (這個不是bug，純粹是因為我覺得這樣設計比較合理…)

**程式架構設計:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Base Class | | | | |
| warship | | | | |
| Derived Class | | | | |
| Cunboat | Destroyer | Submarine | Cruiser | AircraftCarrier |

**1.基礎類別**

warship{

public:

warship(int l,std::string s) //建構子,子類別建構時給予特定長度與名稱

virtual void attack=0 //跳出攻擊介面給玩家使用

virtual bool checksunk=0 //回合結束後用來確認是否沈船,並顯示沈船訊息

bool fire //最基本的攻擊程序,不允許玩家直接使用,主要提供給其他成員函式來呼叫,回傳bool表攻擊是否擊中

void auto\_set //讓船隨機將自己填上一個位子

void set\_position //跳出介面供玩家為該船佈署

void Normal\_fire //每艘船都有的基本攻擊!!!

pair<int,int> getposition //回傳該船第i個position(其他函式需要)

const int length; //船的長度(也就是佔幾格),每種船都有固定的長度

std :: string Category; //船的種類名

~warship();

protected:

std :: pair<int,int>\* position; //船所在的座標,數量跟長度一樣

int health; //船的血量,初始值跟長度一樣

bool sunk; //當血量歸零時,即視為沈船

}

**2.衍生類別:**

舉Destroyer為例，因為每種都差不多

class Destroyer : public warship{

public:

Destroyer(); //建構子,設定長度血量為3,種類為Destroyer. position有3格

void attack //跳出玩家操作介面

bool PlasmaConnon // Destroyer搭載的特殊武器

bool checksunk

}

**3.實作方式：　(**AI部分該組員有另外製作PPT說明，提供參考**)**

棋盤部分由兩種二維陣列來維護，但玩家有兩個所以共有4個二維陣列，一種是char board[16][16]的陣列，用字元符號來記錄”表面”的棋盤，像是船的位置、攻擊過的區域等等。

另一種陣列是warship \*gamedata[16][16]，即16\*16個warship的指標,可指向衍生類別也就是真正的船，若該位置沒有船則指向0，如此一來只要知道座標就可以透過此陣列對該位置的物件進行存取。

之所以分成這兩種是為了方便分工，比如說當我需要call AI來幫我下棋時，我只要給他board的盤面資料，他就可以判斷出下一步較合適的位子。

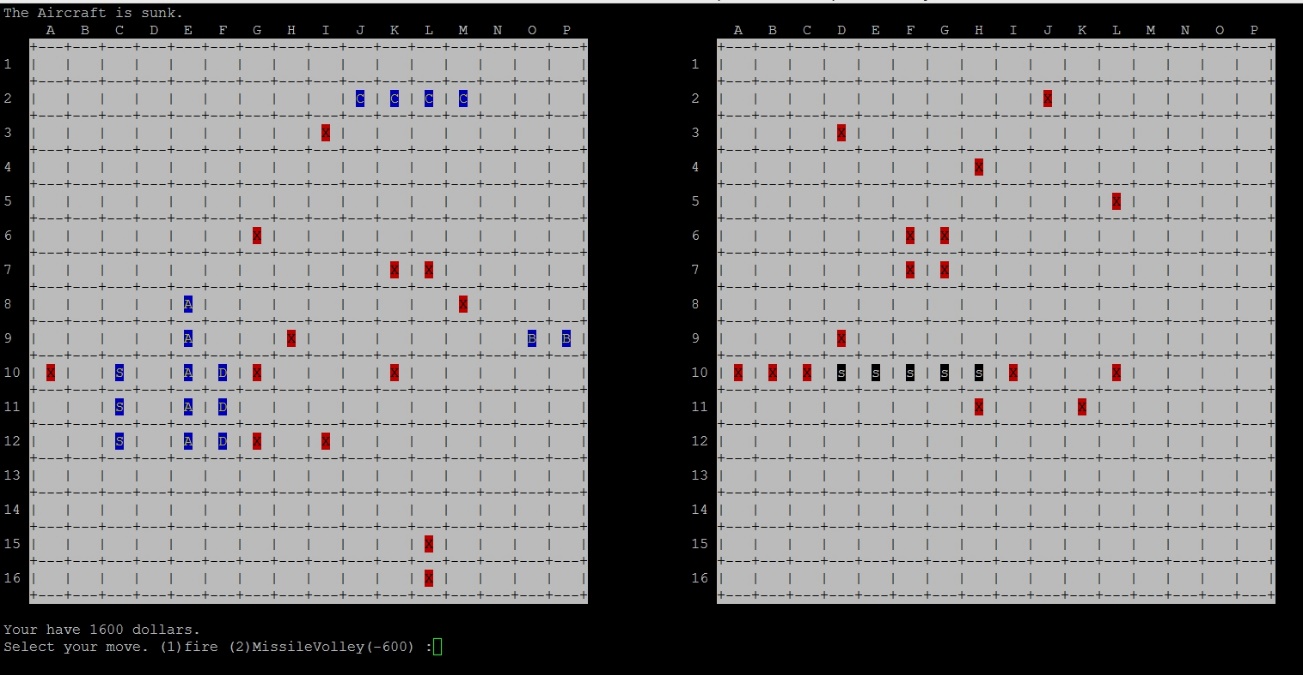
接著是main funtion的架構，多虧了virtual funtion強大的功能讓我可以使用兩個list<\*warship>來協助遊戲流程進行，大概的流程如下:

建立一個list將每艘船的位址丟進去->用iterator現在指向的位置來決定由哪艘船進行攻擊->攻擊完後iterator改指向下一艘船(若iterator到底則從頭開始) ->攻擊後檢查對方是否有沈船,若船沉了便從list中移除

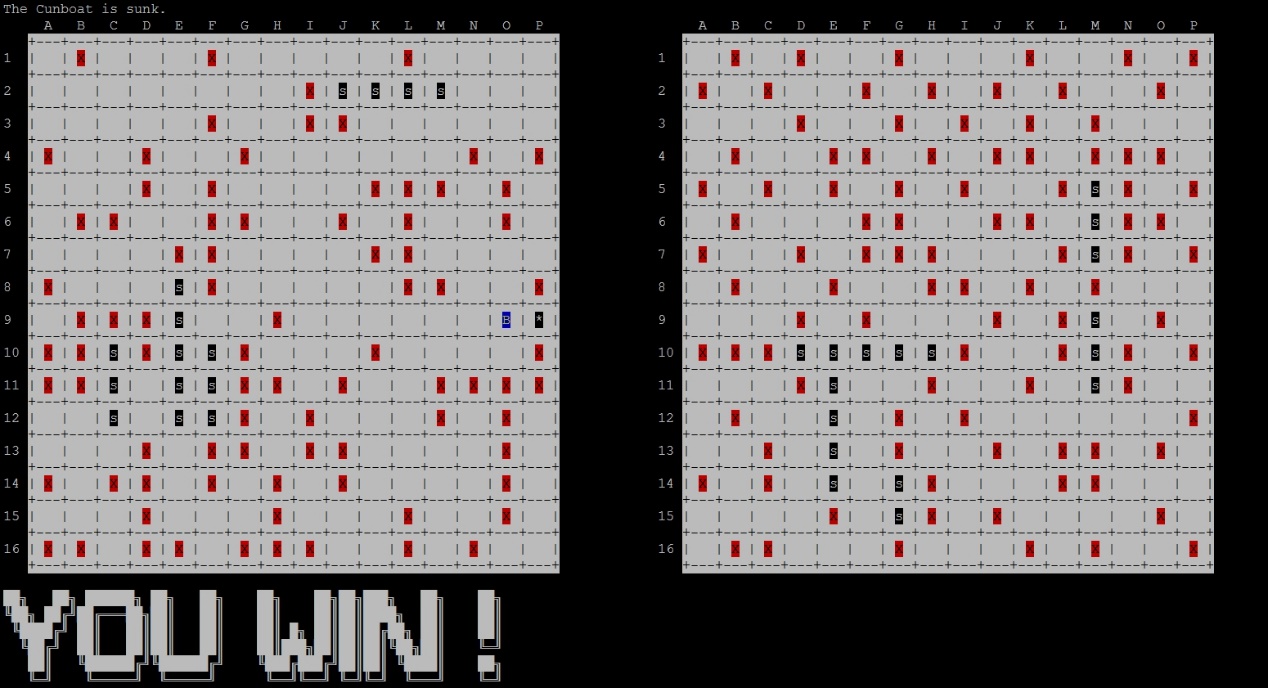
這樣的設計有兩大優點：第一是這樣可以讓各艘船能輪流攻擊到，即使其中有船沉了也能繼續運作。第二點，也是最重要的一點，那就是這樣的設計允許我在list放置不同順序、不同數目、不同種類的船，而且不需要去更改這邊的code，比如說如果我今天想將5v5改成6v6，我只要在雙方的list裡多push\_back一艘船就好，或者如果我想將Cunboat改成一台Submarine，那我只要在push\_back時改成放入一台Submarine的位址就行了。雖然我們在這次的作品中沒有使用到這項功能(因為不知道怎麼呈現)，但是這部分的設計提供了這遊戲一些擴充的空間。

**遊戲擷取畫面:**

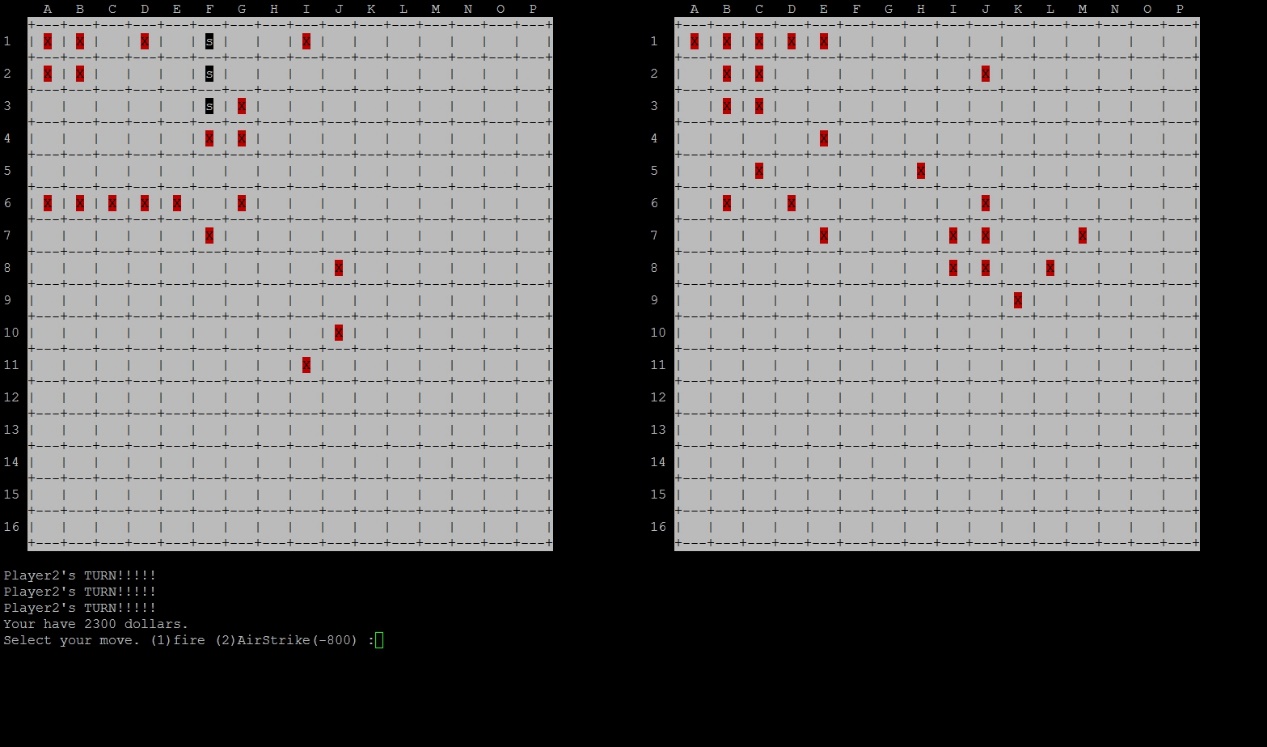
↑遊戲開始畫面(選擇模式)



↑遊戲中，左邊是玩家的區域，而右邊的黑色格子代表電腦的船被擊中了



↑對電腦獲勝～



↑２Ｐ模式

**分工情形:**

林芷敬：計畫發想、程式架構設計、分配工作，撰寫檔案：warship.cpp、warship2.cpp、warship3.cpp、funtions.cpp、main.cpp、warship.h、funtions.h

張書鳴：計畫討論、負責設計AI下棋，撰寫檔案：valueMap.cpp、valueMap.h

羅孟珣：計畫討論、負責遊戲圖像的設計與顯示，撰寫檔案：Display.cpp、Display.h

**心得報告:**

林芷敬：

很高興這次的專題能親手弄出一個簡單的遊戲，其實我從期中就開始構思這次的期末專題了，考量到沒有動畫的實作經驗，而且感覺在Unix系統上浪費時間用shell搞動態畫面好像沒什麼意義(還不如直接用Unity寫算了)，所以我還是希望能夠用可以簡單呈現的遊戲來作為這次專題的主題。因此我一開始想到了早期很流行的棋盤回合制戰略遊戲，這樣的遊戲可以在平面上很好表達也能擴增很多內容的豐富性，畢竟國內外可以參考的例子多到不能再多，而架構也很容易發想，只要讓盤面上所有單位都用同一個base class去繼承並用一個型態為base class的pointer的二維陣列就可以輕易地維護一個戰場的資料，不過在沒有"好看好懂"的介面的情況下，做出一個原創遊戲實在是很難讓別人看懂到底怎麼玩，所以最後還是覺得選了這個雖然有點冷門，但是真實存在的棋盤遊戲。(沒聽過的話去google海戰棋是真的有這遊戲喔)

由於我比較熱衷於這部分，所以這次專題的計畫都是由我來主導，而我的隊友則是由我提出要求並分配工作，這次的專題我很努力地想當成一個正式的專案來做，務求精確的code和分工，我試著讓自己設計的部分更完備好讓其他隊員能夠直接接軌，因此我們幾個隊員可以不需要了解對方code是如何實作的就完成專題。不過，由於事前不夠周慮，導致在打code時改了幾次需求，最後重看一次自己的部分還是覺得很多funtion的功能分的很不清楚，但是也就懶得改了~~

總之，最後計劃順利完成了，我自己覺得這個作品的完成度還算夠，即使要變更難度或一些設定也只要修改一些參數就好了，不過還是希望這個檔案能夠結合一些３Ｄ動畫的以及一些方便操作的介面啊，不然用鍵盤輸入座標有夠麻煩的。

張書鳴：

這個 OOP 專題並沒有我原本所想像的要花大量的時間去撰寫，我原本是想說要與夥伴們一同討論、思考，然後熬夜、爆肝、睡眼惺忪的去demo，但事實上，我的部分大概花了一個晚上構思、一個晚上寫出來，然後我就將我的部分交出去了，不過 麻煩的是，後來發現我的程式部分和遊戲規則與隊友所希望的呈現方式不相同，因為我很久以前就有玩過海戰棋，而那時的記憶已漸漸消逝，而我想說反正就是如我所想像的方式玩，應該不會錯吧，但是，記憶是會被竄改的(請參考:伊莉莎白．羅芙托斯 (Elizabeth Loftus)在 TED 上的演講 )，於是反倒誤解了遊戲規則，此時我的心已涼 了，一想到還要將之前的程式去做繁瑣的 debug，就倍感辛勞與絕望，畢竟 debug 是一件相當煩人的事情，不但傷害我的視力而且還相當消耗精力、腦力、時間，抓蟲 抓整天都不曉得自己到底是電機系還是昆蟲系了，儘管如此，此時我又再次認知到溝 通的重要性，有時即便是你覺得再尋常不過的事情也要與隊友確認過想法一致，否則到時出包、有問題就麻煩了，這裡應該提一下方孝孺的指喻:[ 此疾之奇者，雖病在 指，其實一身病也，不速治，且能傷生。然始發之時，終日可愈；三日，越旬可愈； 今疾且成，已非三月不能瘳(音：抽)。終日而愈，可治也；越旬而愈，藥可治也；至於 既成，甚將延乎肝膈，否亦將為一臂之憂。非有以禦其內，其勢不止；非有以治其外，疾未易為也。]

但後來其實還好，沒有很麻煩，改一下小地方、code 複製下就可以運作了，然而，古人說的好:一波未平，一波又起，過了約莫一兩天，正當我認為萬事大吉、可以安然度過此學期、可以將諸事拋諸腦後之際，隊友傳來了一個不詳的訊息，[你的 BUG 有點嚴重……]，我就知道事情才沒那麼順利，就像我家的蟑螂永遠打不完一樣、宿舍 裡的蚊子總是層出不窮、BUG 總是會在你最意料不到的時間跑出來，阿不是，是會在任何時刻跑出來，但問題不在於說有 BUG，而是因為我那時還要做別的事情，煩憂的事情不斷，這個 BUG 簡直就是讓我崩潰的最後一根羽毛，言歸於此，該解的 BUG 還 是要解完，於是我坐下來、弓著背、雙手放在螢幕上、推了推我的眼鏡，活像個肥宅 一般目不轉睛、毫不停歇地找蟲，明明才過一兩天，這幾段程式碼就好像不是我打的一樣，看都看不懂，但我還是得硬著頭皮開始 debug，此時，我腦海中忽地想到 Rasmus Lerdof, PHP 之父:[我不是一個真的工程師。我把東西弄一弄，弄到能跑之後 就不管了。真的工程師會說「這段程式能跑，但記憶體沒管理好，我們來修好它」。 我只會說，一直重新開機不就好了。]，是吧~~，看看人家大師都這麼說了，反正能跑就好啦，我又何必如此執著於小小的幾行 code 呢~但是顯然這段話不大好，如果大家都真的這樣想的話，客服電話應該會暴增，然後客服的工作應該會多很多吧:一個人寫的爛軟體將會給數十人帶來全職工作，想著想著，才原來 BUG 是出在讀取超出矩陣的範圍呀，找了許久，於是我抱著數十人將要失業的壓力，將 BUG 給修正，再傳給可 靠的隊友，確認沒什麼大問題之後，此次的 OOP 專題才終於告了一段落了，總而言之，從這次的專題，我學到要做好專題必須要有:1.良好的溝通 2.縝密的思考 3.無盡的 耐心 4.一顆還過得去的肝 5.初音(還有欲音、鏡音雙子、GUMI 醬、IA)，才得以成為一 位優良的軟體工程師。

羅孟珣：

一開始接觸到這個題目時，我的直覺就是因為遇到棋盤、方格的問題，我們必須要面對各種困難的二維陣列、指標、以及我較不熟悉的繼承…。但是開始分工之後，真的要先感謝一起努力的好隊友兼室友，一路上的包容以及支援我。我是分配到較簡單的部分－遊戲畫面輸出。我並沒有遇到太大的問題，其中比較需要思考的地方，是要在同一個畫面中，輸出兩個棋盤（雖然跟隊友的AI邏輯思考比起來，這個還是相對簡單的…）。

接下來，為了使畫面變得更美觀，自己上網研究了如何用ANSI escape code改變輸出文字的背景與顏色，發現其實沒有想像中的麻煩，並定義了一些方便使用的字串；另外，因為自己有在用PTT，常常看到上面許多精美的ANSI圖案或動畫，便想說可以利用網路上的資源，將一些酷炫的字體樣式，套用在這次的遊戲介面中，便能在遊戲中顯示不同的字型。

在這次的專題中，除了OOP本身的知識應用外，我覺得我更學習到了團隊合作以及自行上網解決問題的能力，這種大家致力於把專題的成果弄得更加完善的感覺與感動，是我在其他作業或是課程中學習不到的。