SPACE WAR

* 專案主題

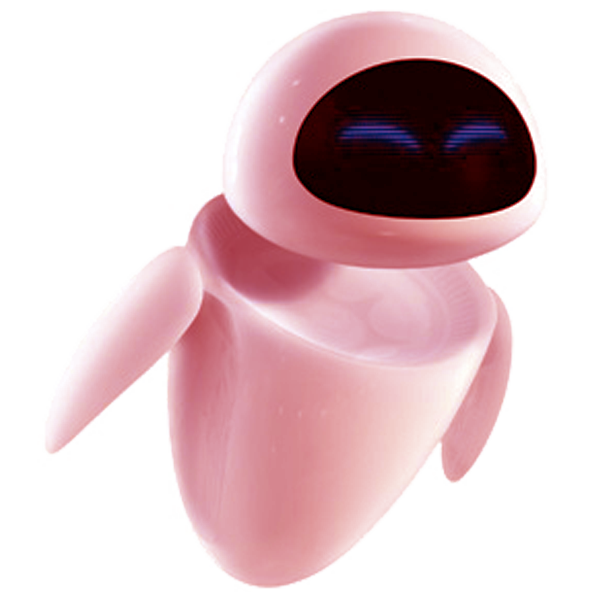
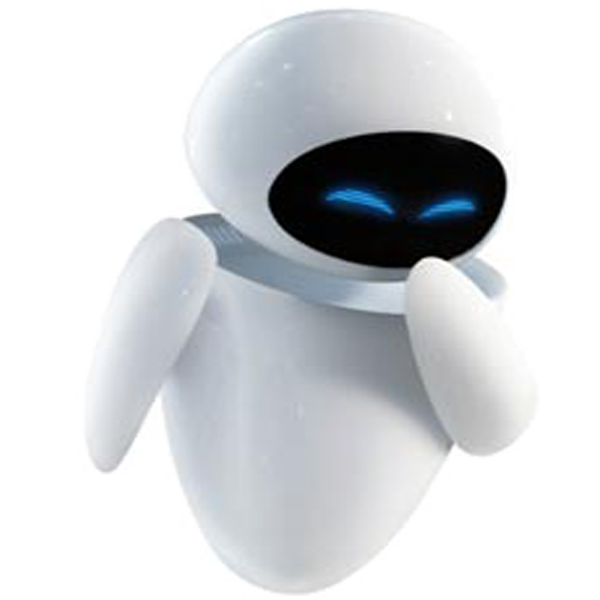
以守塔為主題的炸彈超人遊戲，在此模式下雙方並非以擊殺對方為主要目標，而是以摧毀對方主堡為獲勝目標。

* 遊戲背景

星際戰爭，母星守衛戰。因為資源的爭奪，兩個國度分別以兩個星球作為據點，而玩家們為特派戰鬥員，以摧毀對方據點，阻止對方的前進為任務。

* 遊戲方式

玩家可以選擇直接進攻，或是獲取足夠道具增進自身能力再來行動，甚至利用建築等技能進行防守阻撓對方進攻。玩家死亡後可以在重生點復活，但是每次死亡除了增加復活時間，對手主堡的生命值也會同時增加，因此戰略上的選擇十分重要。



* 遊戲元素介紹
* 主角(包含技能)

如同炸彈超人主角，但除人放炸彈以外，我們的特派戰鬥員還擁有建設牆壁能力。

* 防禦塔

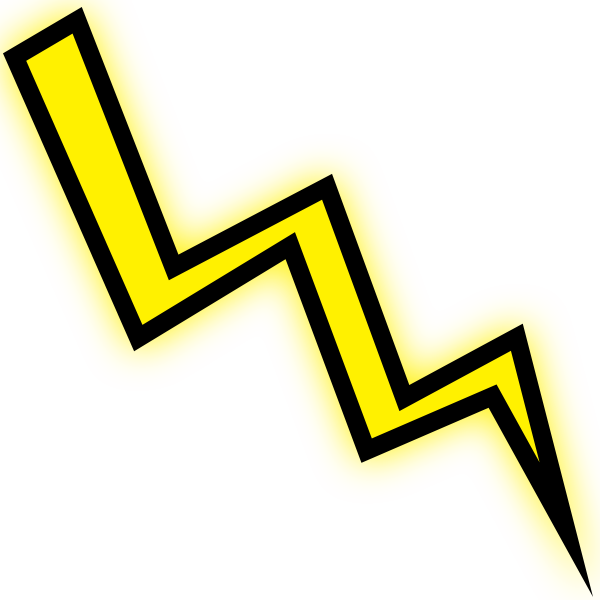
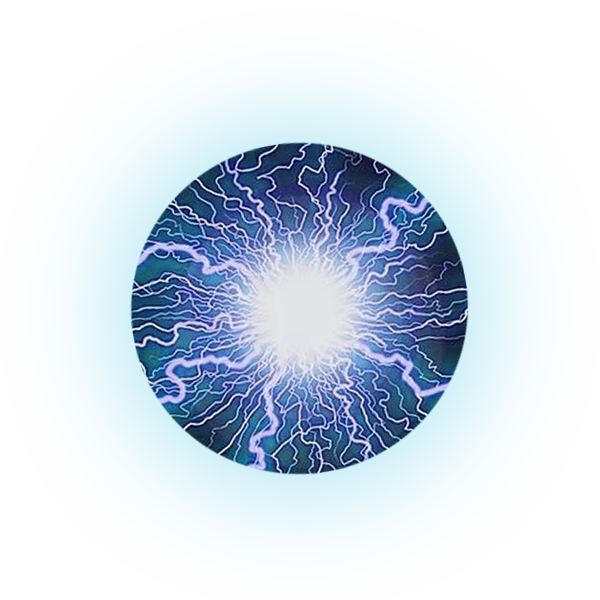
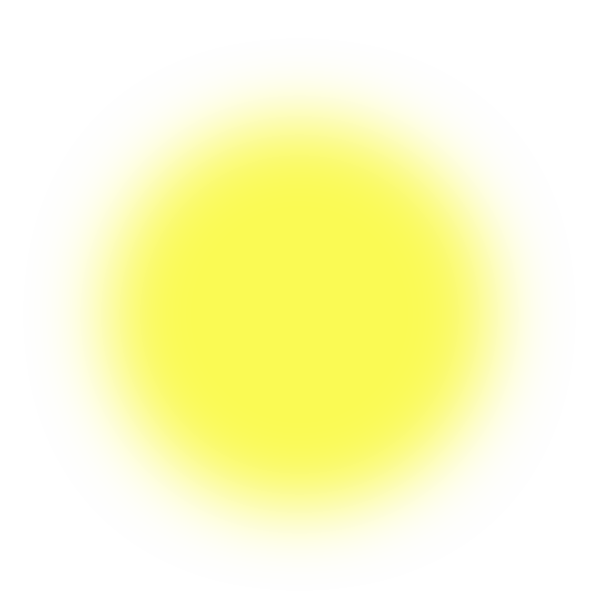
定時進行轟炸阻撓對手進攻的地圖元素。

* 主堡

遊戲的核心，決定勝負的關鍵，有生命值。

* 遊戲道具

可以藉由吃到遊戲道具增加角色能力

加速 加炸彈數 加炸彈威力

* 小組成員及貢獻度(貢獻度為平分，每人皆為1/6)
* 苗新弈 100306074：程式總架構與炸彈衝擊波
* 魏冠宇 100306011：地圖建構，運算工具
* 李育賜 100306085：碰撞偵測，音效
* 高挺桂 100306007：角色操作及多鍵盤input偵測
* 邱冠中 100306019：炸彈設置，防禦塔，Update
* 林意婷 100306017：美術設計
* 系統特色
* **原創性**

本遊戲的程式碼皆為我們自行開發，並沒有參考任何原始碼或是開源碼。

* **程式整體架構**

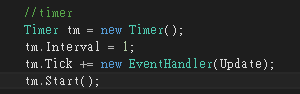
由於WPF並不是個為了設計遊戲而生的環境，因此我們參考知名遊戲引擎unity的架構，作為我們的主要架構。我們在各個script中放入在最一開始時執行的Start()和不斷刷新用的Update()，以此來運作我們的遊戲。

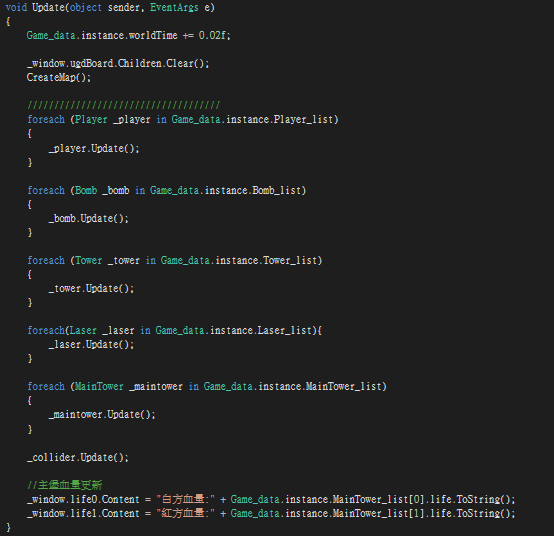
而由於本組在程式編寫上，有多達五人參與，為了能做好分工，同時令日後易於擴充程式，我們在開始開發遊戲時，大量的運用物件導向的概念，盡可能的去設計出了個有結構的程式框架。

其中，我們有幾個共用的script來串起整個框架：

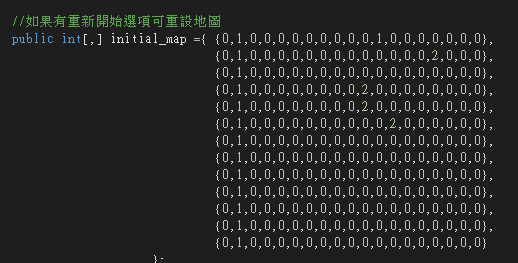
* ”Game”負責控制遊戲的進行，對遊戲中的物件們做Start()和Update()的呼叫。
* ”Game\_data”作為遊戲的資料庫，以List<>來管理遊戲中的各個物件。
* “Tools”程式設計師的工具箱，存放一些共用a的function，在需要時調用。
* **UPDATE**

我們的遊戲模仿時下流行的Unity3D遊戲引擎，使用一種Update的技術概念，在Unity中它的作用是放在Update中的所有程式碼會以每秒約60次的速度重複執行，所以許多事件的發生與否、偵測等等我們都放在裡頭，可說是讓我們炸彈超人遊戲執行的核心概念。

我們以timer配合多個不同List (bomb.player.tower等等)的方式，再搭配foreach跑陣列的手法，在每秒執行1000次Update這個function。各個放在Update裡的不同class( bomb.player….) 會透過foreach分別呼叫屬於自己class裡面的update，所以下面截圖的update等於是個總指揮，指揮所有的update事件，各個class裡面譬如bomb之類的update，才會寫著屬於他們自己需要不斷重複更新及判斷的事件，譬如這顆炸彈是否被碰撞；這個腳色是否要移動等等等。



* **棋盤式地圖建構**

建構一個易於編寫擴充的地圖，以一個二維陣列來存放數據，由於陣列的長寬格數視覺上與實際遊戲地圖完全相同，因此可以令設計者很直覺地去設計遊戲地圖，並且在未來可以發展客製化設計。

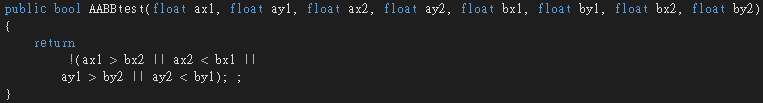
數字代表偵測時地圖的狀態，判定數字為何後進入偵測的方法，並且決定那個座標的格子狀態，例如牆壁、炸彈等。

並且因為Update的運行，判定是一直重複進行，所以可以藉由地圖進行一些瞬間變化的呈現，例如牆壁被炸掉顯示物品等。

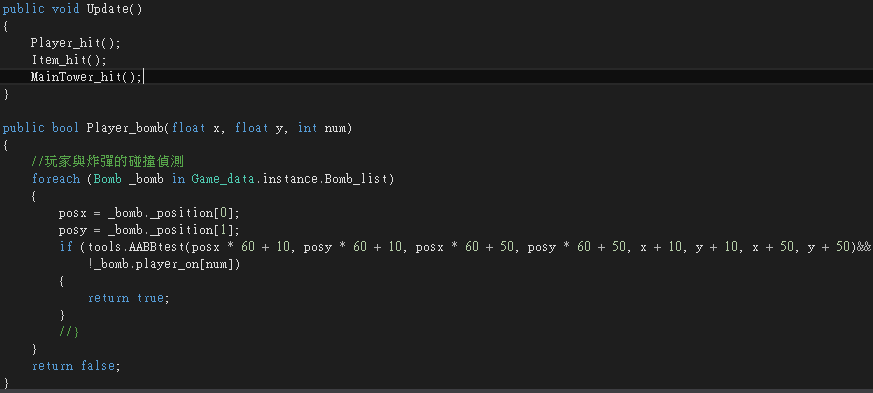
* **碰撞偵測**

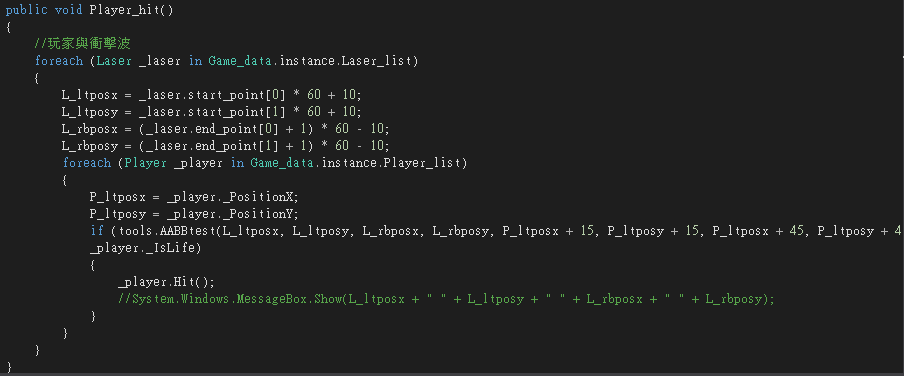
由於我們遊戲是以炸彈超人為基礎，而這種遊戲牽涉到許多不同的碰撞，像是人物跟炸彈的碰撞、人物跟障礙物的碰撞、人物吃道具以及被水球炸到也都是需要用碰撞來判定。因此我們對不同的事件都給予他個別的碰撞偵測。

在許多炸彈超人的遊戲中，會單純以”所屬格子”作為判定依據。但本組認為，那種判定方式不夠真實，像是玩家可能在畫面上看到對手被衝擊波擊中，但由於對手被系統認定為處於另一個格子，因次判定為沒有擊中，諸如此類的情況。是以本組選擇了程式設計上較為複雜的”碰撞偵測”。

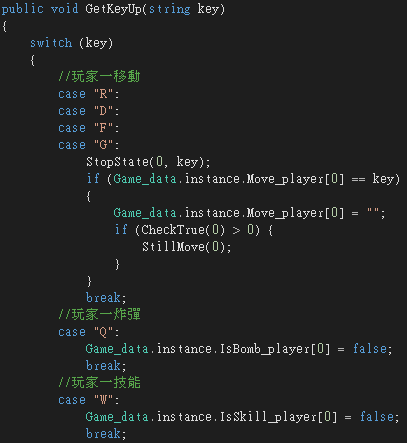
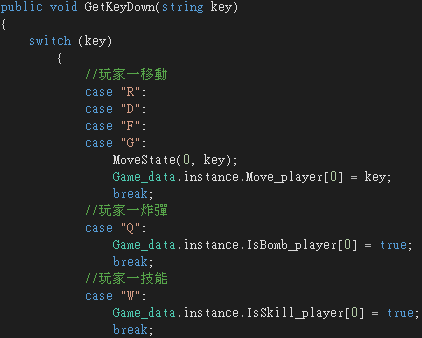
* 這是我們碰撞偵測的核心

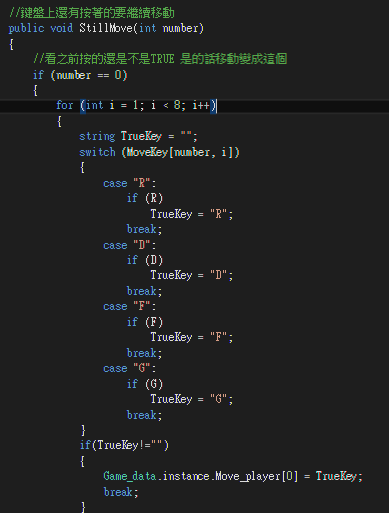
這是透過傳入兩個物體的兩個對角的X跟Y座標來做判斷的碰撞測試，設計者可以藉由調整傳入的範圍，來降低或增加碰撞的敏感度，以此為基礎我們分別把每種碰撞測試寫出來



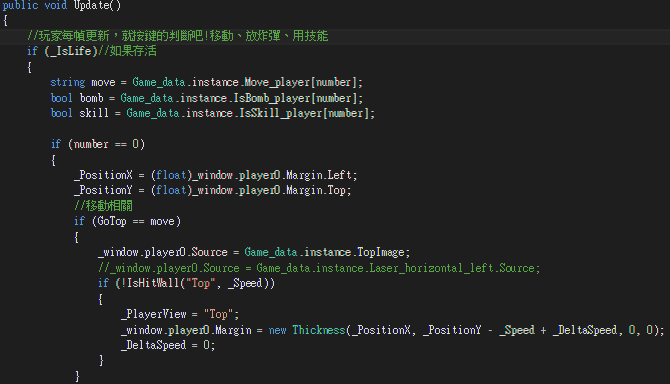


* 這是其中的兩種碰撞偵測
* 同時接受多個Input
* **同時接受多個鍵盤KeyDown、KeyUp**

我們遊戲在接收鍵盤的按下、放開，是使用WPF內建的PreviewKeyDown跟PreviewKeyUp，把取到特定的按鍵丟到自己寫的GetKeyBoard裡面做處理。主要是把有按下的鍵紀錄到遊戲的Game\_data，之後Player就以Game\_data中的值來判斷是否移動、放炸彈、放技能，就能輕鬆把不同人物的操作分開，實現多按鍵同時操作。

另外，為了讓人物移動順暢，在接受鍵盤按下時，會順便紀錄按鍵按下的步驟(MoveState中記錄近八次按鍵)。如果後來按的按鍵放開時，之前按下的按鍵還沒放開，就會取到那個按鍵存到Game\_data中，讓人物能夠往該方向繼續移動而不中斷。

* **人物的操作**

人物的移動、放炸彈、放技能等，都是再update時，判斷Game\_data中相對應的值和人物的狀態(移動速度、是否碰撞、炸彈數量、技能數量)來操作人物。

