

目錄

壹、 簡介.....	1
一、 動機：.....	1
二、 分工：.....	1
貳、 遊戲介紹.....	2
一、 遊戲說明：.....	2
二、 遊戲圖形：.....	3
三、 遊戲音效：.....	7
參、 程式設計.....	8
一、 程式架構：.....	8
二、 程式類別：.....	8
三、 程式技術：.....	9
肆、 結語.....	11
一、 問題及解決方法：.....	11
二、 時間表(不包含上課時間):.....	13
三、 貢獻比例：.....	14
四、 檢核表：.....	15
五、 收穫：.....	16
六、 心得感想：.....	18
伍、 附錄.....	21

壹、簡介

一、動機

還記得在我們小的時候，不像現在擁有那麼多好玩的手機遊戲及先進線上遊戲供我們遊玩，但當時純真的我們，還是能找到專屬於我們娛樂的小天地，那就是……打「網頁遊戲」。每天放學時，最迫不及待的就是奔回家然後打開電腦收尋「史萊姆好玩遊戲區」及「遊戲天堂」……等玩小遊戲的平台，找尋是否有新的、好玩的小遊戲讓我們消遣一下。

其中，我們印象最深刻的就是玩「BOX-HEAD 樂高殭屍人」，這款遊戲能讓我們不斷地從中獲得成就感，面臨永無止境的關卡，然後打倒一隻又一隻的殭屍及魔王，逐漸地強化自己的武器，就彷彿是現實世界一樣，每天我們都會面臨不同的挑戰，我們必須克服它，並且慢慢的累積經驗，過關斬將，持續往永無盡頭的路途邁進！

現在，我們想要重溫兒時的快樂時光，於是選定要做這款小遊戲，馬上就讓我們一起回顧年少的經典回憶吧！

二、分工

(一) 邱子源

規劃遊戲製作進度及負責主遊戲程式碼的撰寫(主角、殭屍、武器……等)，遊戲試玩、除錯及撰寫報告。

(二) 洪俊銘

負責圖片的製作、美工及遊戲分支的程式碼撰寫(地圖、主頁面……等)，遊戲試玩、除錯及撰寫報告。

貳、遊戲介紹

一、遊戲說明

(一) 操作說明

主頁面：有「Options」及「START」兩個按鈕，滑鼠點擊「Options」即有主要的操作說明，滑鼠點擊「START」進入遊戲。

遊戲結束後：按下「SPACE」即可進入主畫面，然後重新開始遊戲。

1. 上下左右之方向鍵控制主角的移動方向（可以走八方位）。
2. 按下「空白鍵」為發射子彈。
3. 按下「Z」為放置定時炸彈。

(二) 遊戲規則

遊戲一開始，殭屍會從四個方向出現，並且朝主角的方向前進（主角初始位置在地圖中心點，此外，主角也能任意穿梭地圖上下左右的四個洞），主角必須發射子彈殲滅場上所有的殭屍，每殺死一隻殭屍會獲得一分的積分。







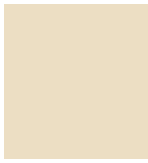










第一關會有 20 隻的殭屍，當場上所有殭屍都被殲滅時，及邁入第二關，此時會有 40 隻的殭屍，當 40 隻全被殲滅後，邁入第三關…依此類推，殭屍數以公差 20 的數量增加，直到主角被殭屍碰到後就 GAME OVER(為了方便助教玩遊戲打分數，故只做九關)。

遊戲結束後，按下空白鍵後即可重新開始遊戲。










(三) 密技

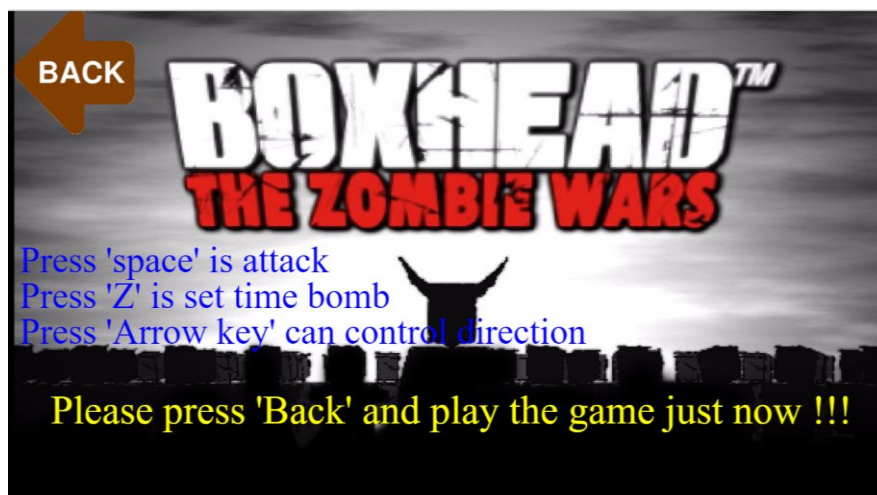
當周圍殭屍太多時，可以直接按下 Z 來設定定時炸彈，能炸死周圍之殭屍，以此來躲避殭屍的追殺。

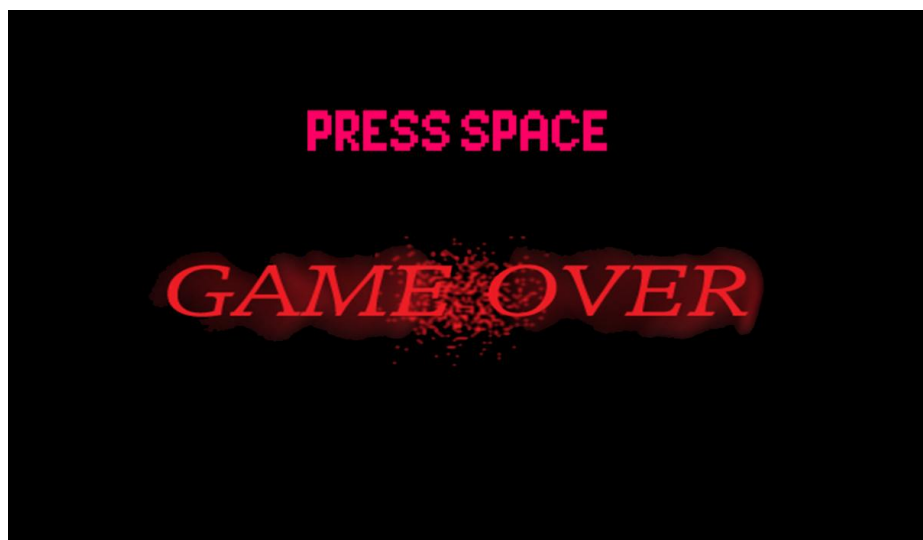
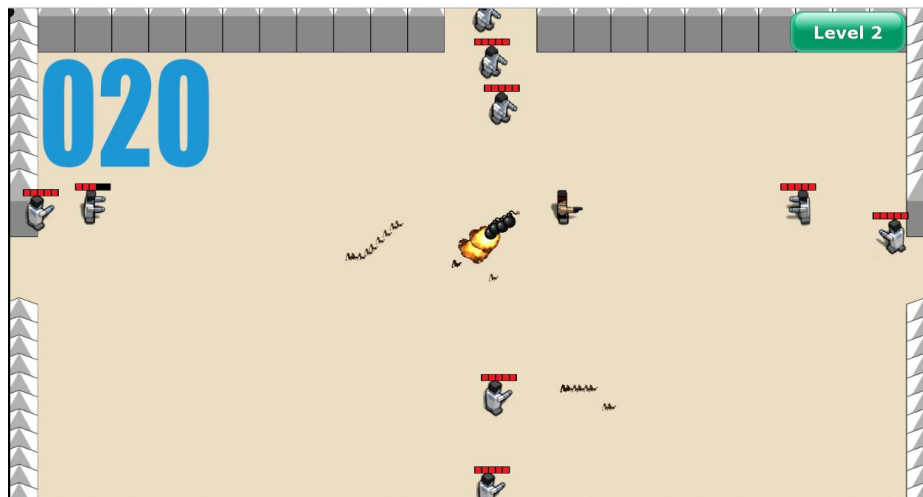
二、 遊戲圖形

項目	圖片	圖片	圖片
起始畫面			
起始畫面 & 結束畫面			
			
遊戲地圖			
			
遊戲地圖	 	 	
			
			
主角			

主角			
主角			
殭屍			
殭屍			
殭屍			 (殭屍血條)
主角武器			
主角武器			
主角武器			
主角武器			

主角武器			
主角武器			
主角武器			
分數牌	0123	456	789



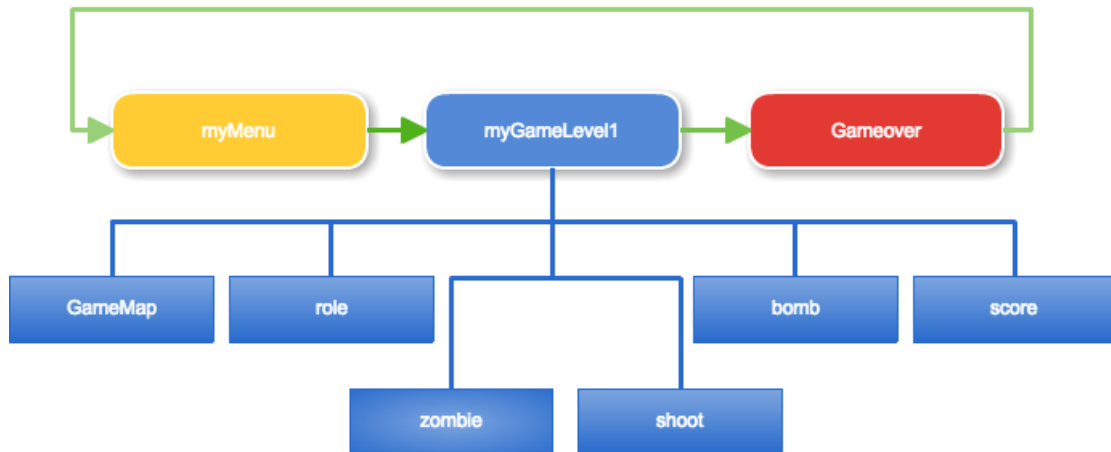


三、遊戲音效

時機	用途說明	音效檔名
遊戲前	起始畫面的音效	Pucker_Up.mp3
	操作說明頁面的音效	Evil_March.mp3
遊戲中	遊戲進行中的音效	bgm.mp3
	主角射子彈的音效	Gunshot.mp3
	主角放置定時炸彈的音效	Explosion_sound.mp3
	主角走路的音效	Running.mp3
	殭屍被擊中的音效	hit.mp3
	殭屍死掉的音效	zombie.mp3
遊戲後	GAME OVER的音效	Gameover1.mp3
		Gameover2.mp3

參、程式設計

一、 程式架構



二、 程式類別

類別名稱	類別功能	類別行數
GameMap	繪製整個遊戲的地圖	116 行
Gameover	進入遊戲結束畫面	91 行
loadGame	載入所有有創立的 js 檔	55 行
myGameLevel1	控制主角的功能以及處理殭屍、子彈、主角之間的事件。	519 行
myMenu	主頁面，控制進入操作說明或開始遊戲	139 行
option	遊戲之操作說明	56 行
score	計算及顯示分數	49 行
Shoot	控制主角武器使用狀況	50 行
zombie	控制殭屍之一切行動	167 行
總程式行數		1242 行

三、程式技術

(一) 主畫面的製作方法

利用滑鼠移動判斷游標是否有在「Options」或是「START」上，倘若，點擊它，即可進入下一個頁面，同時在這裡也有載入主頁面之音效。

(二) 主角的撰寫方法

主角是一開始在 load 的時候即 new 一個 role 的物件，生成在遊戲畫面的中央。主角有八個方位的走路方向，而換方位時圖片也會跟著換。於是我們是用先宣告八個方位的圖片，預設是往下走。當換方位時，存下主角現在的座標，便將現在方位的圖片 detach，再將新方位的圖片在剛剛存下的座標上 attach。主角可從上入口進入再從下入口出來或相反，也能從左入口進入再從右出口出來亦可相反。

(三) 武器的撰寫方法

每按一下空白鍵就會 new 一個新的子彈物件 (shoot) 出來，子彈會依照主角現在面對的方向飛行。在 myGameLevel1 裡偵測是否射到殭屍，若射到，將子彈 detach 掉，殭屍扣一個血條；若飛行 600 格沒射到，則會自動 detach。

每按一下「Z」鍵就會 new 一個新的炸彈物件 (bomb) 出來，會有一個炸彈的圖片先 attach 出來，過了引燃時間後(update100 次)，炸彈就會爆炸 (會有爆炸的動畫)。當爆炸時只要在範圍內的殭屍都會直接死亡。

(四) 繪製地圖的方法

先將我們要建立的地圖以陣列儲存起來 (用 0、1 表示，0 為牆壁；1 為地板)，由於地板是蓋在牆壁下面，所以先跑一次迴圈將所有地板都鋪好後，在跑一次迴圈建立牆壁，如此一來，地圖就大功告成了。

(五) 分數的產生方式

每當主角將一隻殭屍射到血量歸零，即加一分。再將分數轉換成百位十位個位三個數字，三個數字再轉成數字的圖片並 attach 到遊戲畫面。

(六) 殭屍的製造方法

每新增一隻殭屍就會 new 一個新的 zombie 物件出來，每一關都會有不同數量的殭屍分別從四個入口進來，因此每個殭屍也會有代表自己的數字，將那個數字除四取餘數，餘 1 從上出口出來，餘 2 從右出口出來，餘 3 從下出口出來，整除從左出口出來。

每隻殭屍都有八方位地走路方向，會自動偵測主角位置。八方位走路的圖片是用 AnimationSprite 寫的，用 xy 軸比大小的方式判斷出殭屍該往哪個方向走。找出方向後，在找出該方向的走路圖片，播放出對應方向的 Animation (this.zombie.start({from:0, to: 0});)。

每隻殭屍都有 5 格血條，血條的製作方法，一條黑色的背景條，再加上五個血色正方形。一個正方形代表一血量。血量歸零即死亡，死亡的方法是用 detach，但 detach 後並不會讓殭屍物件消失，只是畫面上看不到而已，因此殭屍仍會往主角座標走，只是看不到，所以只要將死掉的殭屍標注起來，碰到主角，將會視為無效。

(七) 遊戲結束的條件

殭屍碰到主角即結束（當殭屍與主角座標一樣時）。

肆、結語

一、問題及解決方法

(一) 程式碼方面

Q1：剛開始我們將全部的程式碼都打在 myGameLevel1 的 js 檔裡，導致程式碼雜亂不章。

A1：打這種大量的程式一定要先想好架構，分好不同類型的專案再著手進行撰寫程式碼的動作。我們就是急於在第一次 demo 時想秀出多一點東西而把主角及殭屍寫在一起，然而，demo 完後才創立新的 framework，再將殭屍的程式碼挪過去，在 function 裡宣告需要連結的 js 檔名，打對路徑，即可得到正確的資料。

(二) 地圖方面

Q2：地板與地板間有不接續的狀況出現，及地板的圖片會蓋過牆壁。

A2：一開始我們沒考慮到地板的像素與牆壁的像素不同大小，才會導致不接續的狀況或是有圖片印不出來的情況，最後我們用先將地板鋪完再建立牆壁的方式即可解決以上之問題。

(三) 角色方面

Q3：主角始終只能走四個方位，無法讀取斜著走的圖片。

A3：一開始我們同時按下左上、左下、右上或右下之鍵盤時，還是只會讀取其中一顆按鈕，但我們善用「if」、「else if」及「else」等功能，排列優先順序後，即可讓主角走八方位。

(四) 角色方面

Q4：主角在某一角度射擊時，會導致程式停止運作而當機。

A4：這其實不是甚麼大問題，只是在考驗我們的細心度，做一個龐大的專案，若在一堆相似的程式碼中只需要微改一些變數，必須要注意每個細節是否正確，否則一個小地方錯誤就會造成遊戲出現 bug。

(五) 殭屍方面

Q4：殭屍死掉還是會碰到主角導致 GameOver 的 bug。

A4：殭屍死亡的方式是將他 detach，但 detach 後並不會讓殭屍物件消失，只是畫面上看不到而已，因此殭屍仍會朝主角的方向移動，只是看不到，所以只要將死掉的殭屍標注起來，碰到主角，將會視為無效。

(六) 武器方面

Q5：子彈不能 detach。

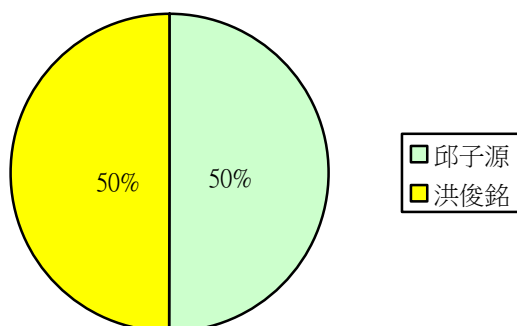
A5：版本不一樣，須更新 Framework 即可解決此問題。舊版的 detach 只會 detach 物件周邊的圖片，而新版的則會把整個畫面都 detach。

二、 時間表（不包含上課時間）

週數	日期	個人花費時間	工作規劃	遊戲圖片
1	2/25~3/3	邱子源：3 小時 洪俊銘：4 小時	練習 git 及體驗範例程式碼	
2	3/3~3/10	邱子源：3 小時 洪俊銘：3 小時	完成地圖，並且讓角色能在地圖上行走	
3	3/10~3/17	邱子源：3 小時 洪俊銘：3 小時	完成地圖 並且讓角色能在地圖上行走	
4	3/17~3/25	邱子源：5 小時 洪俊銘：4 小時	主角改圖片並且有範圍限制	
5	3/25~4/1	邱子源：6 小時 洪俊銘：6 小時	新增殭屍角色（殭屍會往主角方向移動）	
6	4/1~4/8	邱子源：7 小時 洪俊銘：6 小時	多新增幾隻殭屍，設計 menu	
7	4/8~4/15	邱子源：5 小時 洪俊銘：6 小時	主角可穿梭地圖的上下左右四個洞。新增射擊功能	
8	4/15~4/22	邱子源：3 小時 洪俊銘：4 小時	新增射擊功能可射擊殭屍	

9	4/22~4/29	邱子源：3 小時 洪俊銘：3 小時	新增多隻殭屍	
10	4/29~5/6	邱子源：4 小時 洪俊銘：5 小時	新增射擊功能 可射擊殭屍	
11	5/6~5/13	邱子源：8 小時 洪俊銘：8 小時	射擊殭屍，殭屍會受到傷害，新增血條	
12	5/13~5/20	邱子源：10 小時 洪俊銘：8 小時	殭屍會打主角，主角會死。會有分數	
13	5/20~5/27	邱子源：6 小時 洪俊銘：6 小時	做最後修改	
14	5/27~	邱子源：35 小時 洪俊銘：35 小時	完成所有遊戲細節，製作期末報告	
總共花費時間		邱子源：101 小時 洪俊銘：101 小時		

三、貢獻比例：



四、檢核表：

	項目	完成否	無法完成的原因
1	解決 Memory leak	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
2	自定遊戲 Icon	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
3	全螢幕啟動	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
4	修改 Help->About	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
5	初始畫面說明按鍵及滑鼠之用法與密技	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
6	上傳setup檔	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
7	報告字型、點數、對齊、行距、頁碼等格式正確	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
8	報告封面、側邊格式正確	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	

五、收穫

(一) 邱子源

1. 最大的收穫就是活用 class 的概念，讓每個物件都能有專屬於自己的功能及變數，同一種物件也只要用陣列宣告就能有一樣的功能、副程式跟變數等。一開始在打這遊戲的時候，我們把所有的程式都打在 myGamelevel1，造成整個程式架構非常凌亂，也無法任意呼叫物件的變數，或 new 一個一樣的物件出來。
2. js 最方便得地方就是不用特別去宣告他是哪種資料型態，即可直接使用，甚至有的變數可以拿來存放圖片。
3. 物件與 this 的熟練運用，呼叫另外物件的變數，例如：我要在 myGamelevel1 中使用 zombie 中的 hurttime 時，只要打對他的 class，像 this.zombie.hurttime，就能準確的呼叫到你要的值。
4. 終於能把之前所學的所有程式架構或方法活用在一個大程式上，而不是像以前一樣，一個程式只用到一個程式技巧。當把所有的方法一起使用的時候，有時架構會變得很亂，因此必須非常清楚自己程式的架構。
5. 善用動畫的來存放圖片，AnimationSprite 可存放多張圖片，播放變成動畫。我利用這點來存放同個物件的多張圖片，再透過一次只播放一張，要換圖片時只需換數字即可。

(二) 洪俊銘

1. 當初我們會選定要做 html5 除了是看它的範例程式碼感覺比較簡單明瞭外，另一方面也是想學有別於 C 及 C++ 外的程式語言，經過了這學期的磨練，我覺得我對 java script 有一定程度的瞭解了，這也是自己第一次自學程式碼，滿有成就感的。
2. 瞭解到許多專屬 function 的用法，例如：key up、key down、mousemove、mouseup、mousedown...等，以往在打 C 語言總是只有單調的輸入輸出，但這次能夠搭配著鍵盤及滑鼠讓螢幕上的圖片物件移動，使我感覺我的程式能力又往前跨了一大步。
3. 雖然老師說本次 project 的重點不在於圖片的精美度，但我總還是希望自己做的遊戲的圖片不能太醜陋，於是下功夫去專研 photoshop 的一些隱藏功能，所以一整學期下來，除了程式能力進步不少外，修圖、設計排版的能力也跟著進步了。
4. 能適時的使用「console.log」來 Debug，遇到問題不再只是仰賴他人的幫助，能夠靠自己的能力找出 bug 並且排除。
5. 這次在撰寫遊戲的過程中，能讓我發揮之前所學的基礎，並且打理好自己的想法，在轉換成程式語言呈現給大家。這次的程式量不像之前一樣百行左右就能搞定的，做這種上千行程式碼的 project，一定要事先想好架構，然後在打在不同的專案裡，最後在慢慢合併在一起，絕不能想到甚麼就打甚麼，否則絕對 bug 一堆，而讓我們無所適從。

六、心得感想

(一) 邱子源

上完了整個學期的 OOP 實習以後，我學到了很多很多，也終於有自己完成一個大 project 的感覺。我跟我的隊友在其實在上學期的 OOP 課程中，沒有學得很精，所以在學期初要選題目的時候，我們非常的頭痛。一心想要做出一個好玩又完整的遊戲，一方面又覺得自己的能力不足，在經過多方考兩量以及與老師的討論，我們決定做 boxhead 殭屍射擊遊戲，雖然遊戲畫面看似簡單，但其實是個很精緻遊戲。決定完題目的時候，我鬆了一口氣，因為這遊戲似乎滿簡單的，對於我們這種程式沒有很強的來說，似乎不用太煩惱，於是便沒有很認真於這門課上面。

隨著時間過去，發現已經過了兩個禮拜了，自己卻只做完老師一開始規定的功課後，我開始緊張了，因為我們的進度跟別組的比起來慢了很多。於是我便開始認真研究 HTML5，利用老師給的範例程式下去改，終於在一個禮拜內趕出了一個可以用鍵盤上下左右移動的物件。再配合上隊友做出的地圖，終於有個遊戲的雛形出來了，當下真的很開心，因為以往的程式設計都是一些死板板的題目，大部分都是只有文字的輸入輸出在做變化，很少有這種以圖片當物件做設計的題目。看到他們能利用鍵盤操作時，真的覺得很有成就感，是又激起了我對打程式的熱情，一個禮拜內又做出殭屍的角色。

一連做出了主角跟殭屍後，展現了第一次 demo，雖然跟別組起來，我們的遊戲看起來稍顯薄弱，功能較沒他組那麼多，因此我們沒有獲得很高的成績。過了第一次 demo 以後，信心大受打擊，又遇到期中考，我們連續好幾個禮拜沒有認真擴充我們的遊戲。直到考完期中以後，才發現自己又落後別人一大截，於是又開始努力擴充我們的遊戲，但是一直遇到瓶頸，像是子彈的圖片射出去的時候沒辦法消除，所以會一直停留在畫面。

問了助教以後也一直沒有得到解答，一直糾結於我的程式碼是哪裡有大錯，後來經過助教不斷的測試，證實我的程式碼是對的，於是我們開始找是不是 framework 哪裡有 bug，後來助教更新了 framework 以後，子彈變成正常消除，而不會一直卡在螢幕上。因為這個 bug，我們卡住了一兩個禮拜，不過好險順利解決了。

之後第二次 demo 展示出的已經是遊戲的完整架構，有主角、有殭屍、可以打殭屍、被殭屍碰到會死亡... 等等，我們的遊戲終於幾乎已經是個完整的遊戲了。過了第二次 demo，我們便開始著手於新增遊戲的趣味度，像是：新增分數、新增音效、讓殭屍出來的頻率變快、給主角新的武器——炸彈... 等等。

終於，我們完成了我們的遊戲。雖然跟原作差很多，但是就遊戲的角度而言，已經是個完整的遊戲了。能做出一個遊戲，對我來說，真的不容易，已經突破了自己很多。做完真的很有成就，也覺得自己上學期沒有學得很好的 OOP 也獲得了很多，不論是 class 的概念，還是其他在資料結構學到的一些程式設計方法，終於有應用到的感覺了。雖然 Debug 的過程很辛苦，但每修復一個 Bug 就覺得自己得程式語言又成長了一點，一個學期累積下來，我真的覺得我的程式能力在這堂課成長了很多，謝謝教授還有助教們，每次發問都很耐心的回答我們，即便有時候問的問題是很基本的概念，但是教授跟助教都會耐心的教我們，或是幫我們 Debug，真的很感謝，有你們我們才在這堂課得到了那麼多。

(二) 洪俊銘

在學期初時，由於我跟我的隊友 C++ 的底子都不是很好，所以我們一度懷疑自己到底是否能獨立完成一個遊戲的製作。當時，徬徨的我們也是經過了百般的抉擇後才挑選了「HTML5」來撰寫我們的遊戲，想說這樣能夠跳脫 C 及 C++ 來磨練自己學習新的程式語言，希望能藉由本學期扎實的學習重新燃起對程式的學習慾望。

然而，一開始光是練習 git 及體驗範例程式碼就被衝突及各種 bug 搞得滿是挫折，但是我們並沒有因此而放棄，從原本是用雲端來互相連通程式碼到慢慢地能夠接受使用 git 來做連結，面對 bug 時也能自己上網找到 java script 特殊的使用語法及技巧來解決問題，此外，我覺得收穫最大的就是我們能善用「console.log」來 Debug，不再是遇到問題就一昧的仰賴助教及同學的幫忙，如果我們能靠自己的力量解決 bug，我們的印象一定會相當深刻，往後遇到類似的問題就不會在出錯了，Debug 成功後也會很有成就感！

這個學期也進入尾聲了，看著我們的遊戲終於慢慢地成形內心也是無限的感動，還記得學期初不太熟悉 java script 時，每個禮拜總是設定一樣的進度然後都還是完成不了，謝謝助教及老師當時並沒有嚴厲的苛責我們，反而是適時地提點我們，讓我們不在一個問題上不斷地徘徊而找不到出口，也謝謝隊友的幫忙，整學期下來我們的合作是相當愉快的，分工也相當明確，常常能夠互相 cover 來完成進度。

這個學期的遊戲製作真的讓我受益良多，多學習了一種程式語言，也有符合學期初的期待，使我重新燃起學習程式的興趣了！

伍、附錄

GameMap

[illegible]

```

//先印出地板
for (i = 0; i < this.simplemap.length; i++) {
    this.map[i] = new Array;
    for (j = 0; j < this.simplemap[i].length; j++) {
        switch (this.simplemap[i][j]) {
            case 0:
                continue;
            case 1:
                this.map[i][j] = new Framework.Sprite(define.imagePath + 'ground.png');
                this.detail[i][j].canwalk = true;
                break;
        }
        this.map[i][j].position.x = this.position.x + (this.MW * j) + this.MW / 2;
        this.map[i][j].position.y = this.position.y + (this.MH * i) + this.MH / 2;
        this.detail[i][j].minX = this.map[i][j].position.x;
        this.detail[i][j].maxX = this.map[i][j].position.x + 32;
        this.detail[i][j].minY = this.map[i][j].position.y;
        this.detail[i][j].maxY = this.map[i][j].position.y + 32
        rootScene.attach(this.map[i][j]);
    }
}
//再印牆壁
for (i = 0; i < this.simplemap.length; i++) {
    this.map[i] = new Array;
    for (j = 0; j < this.simplemap[i].length; j++) {
        switch (this.simplemap[i][j]) {
            case 0:
                this.map[i][j] = new Framework.Sprite(define.imagePath + 'wall.png');
                this.detail[i][j].canwalk = false;
                break;
            case 1:
                continue;
        }
        this.map[i][j].position.x = this.position.x + (this.MW * j) + this.MW / 2;
        this.map[i][j].position.y = this.position.y + (this.MH * i) + this.MH / 2;
        this.detail[i][j].minX = this.map[i][j].position.x;
        this.detail[i][j].maxX = this.map[i][j].position.x + 32;
        this.detail[i][j].minY = this.map[i][j].position.y;
        this.detail[i][j].maxY = this.map[i][j].position.y + 32
        rootScene.attach(this.map[i][j]);
    }
}

};
this.initialize = function(){
};
this.update = function(){
};
this.draw = function(rootScene){
};
}

```

Gameover

```
var Gameover = Framework.Class(Framework.Level, {

  load: function () {
    this.gameover = new Framework.Sprite(define.imagePath + 'Game over.png');
    this.go_back = new Framework.Sprite(define.imagePath + 'press space.png');
    this.gameover.position={
      x:800,
      y:450
    };
    this.go_back.position={
      x:800,
      y:200
    };
    this.rootScene.attach(this.gameover);
    this.timer = 0;
    this.play = 1;
    console.log("score=%d",this.score_sum)
    for(i=0;i<50;i++)
    {
      this.timer++;
    }
    if (this.timer==49)
    {
      this.play=-(this.play);
      this.timer=0;
    }
    if(this.play==1)
    {
      this.rootScene.attach(this.go_back);
    }
    else
    {
      this.rootScene.detach(this.go_back);
    }
    this.digit = new Framework.AnimationSprite({url:[define.imagePath + 'zero.png', define.imagePath +
    'one.png',define.imagePath + 'two.png',define.imagePath + 'three.png',
    define.imagePath + 'four.png', define.imagePath + 'five.png', define.imagePath + 'six.png', define.imagePath +
    'seven.png', define.imagePath + 'eight.png', define.imagePath + 'nine.png']});
    this.tens = new Framework.AnimationSprite({url:[define.imagePath + 'zero.png', define.imagePath +
    'one.png',define.imagePath + 'two.png',define.imagePath + 'three.png',
    define.imagePath + 'four.png', define.imagePath + 'five.png', define.imagePath + 'six.png', define.imagePath +
    'seven.png', define.imagePath + 'eight.png', define.imagePath + 'nine.png']});
    this.hundred = new Framework.AnimationSprite({url:[define.imagePath + 'zero.png', define.imagePath +
    'one.png',define.imagePath + 'two.png',define.imagePath + 'three.png',
    define.imagePath + 'four.png', define.imagePath + 'five.png', define.imagePath + 'six.png', define.imagePath +
    'seven.png', define.imagePath + 'eight.png', define.imagePath + 'nine.png']});
    this.digit.position = {x:800,y:700};
    this.tens.position = {x:1000,y:700};
    this.hundred.position = {x:1200,y:700};
    this.digit.num = this.score_sum%10;
    if(sum>=10)
    {
      this.tens.num = ((this.score_sum-this.digit.num)/10)%10;
      if(sum>=100)
      {
        this.hundred.num = ((this.score_sum-this.digit.num-this.tens.num*10)/100)%10;
      }
      else
      {
        this.hundred.num = 0;
      }
    }
  }
  else
```



```

    {
      this.tens.num = 0;
    }
    this.digit.start({ from: this.digit.num, to: this.digit.num });
    this.tens.start({ from: this.tens.num, to: this.tens.num });
    this.hundred.start({ from: this.hundred.num, to: this.hundred.num });
  },

  update: function () {
    for(i=0;i<50;i++)
    {
      this.timer++;
    }
    if (this.timer==49)
    {
      this.play=-(this.play);
      this.timer=0;
    }
    if(this.play==1)
    {
      this.rootScene.attach(this.go_back);
    }
    else if(this.play==-1)
    {
      this.rootScene.detach(this.go_back);
    }
  },
  //按下空白鍵後回主頁
  keydown: function (e) {
    if(e.key === 'Space'){
      Framework.Game.goToLevel('menu');
    }
  },
});

```

mainGame

```
//當有要加關卡時，可以使用 addNewLevel  
//第一個被加進來的 Level 就是啟動點，所以一開始遊戲就進入 MyMenu  
Framework.Game.addNewLevel({ menu: new MyMenu() });  
Framework.Game.addNewLevel({ level1: new MyGame() });  
Framework.Game.addNewLevel({ gameover: new Gameover() })  
Framework.Game.addNewLevel({ option: new Option() });  
  
//讓 Game 開始運行  
Framework.Game.start();
```

myMenu

```
var MyMenu = Framework.exClass(Framework.GameMainMenu, {
    //初始化 loadingProgress 需要用到的圖片
    initializeProgressResource: function() {
        this.loading = new Framework.Sprite(define.imagePath + 'loading.jpg');
        this.loading.position = { x: Framework.Game.getCanvasWidth() / 2, y: Framework.Game.getCanvasHeight() / 2 };
    },
    //在 initialize 時會觸發的事件
    loadingProgress: function(ctx, requestInfo) {
        this.loading.draw(ctx);
        ctx.font = '90px Arial';
        ctx.textAlign = 'center';
        ctx.fillStyle = 'white';
        ctx.fillText(Math.round(requestInfo.percent) + '%', ctx.canvas.width / 2, ctx.canvas.height / 2 + 300);
    },
    load: function(){
        var photoLink =
        [
            define.imagePath + 'Boxhead-1.jpg'
        ];
        this.oo = new Framework.AnimationSprite({
            url: [define.imagePath + 'option.png', define.imagePath + 'option11.png']
        });
        this.photo = new Framework.AnimationSprite({ url: photoLink, loop: true, speed: 0.05 });
        this.bott = new Framework.AnimationSprite({
            url: [define.imagePath + 'start.png', define.imagePath + 'start11.png']
        });
        this.bott.start({ from: 0, to: 0 });
        this.change = 0;
        this.oo.start({ from: 0, to: 0 });
        this.optionTouch = { x: 0, y: 0 };
        this.isTouch_option = false;
        this.previousTouch = { x: 0, y: 0 };
        this.currentTouch = { x: 0, y: 0 };
        this.isTouchArrow = false;
        this.center = new Framework.Scene();
        this.center.position = {
            x: Framework.Game.getCanvasWidth() / 2,
            y: Framework.Game.getCanvasHeight() / 2
        };
        this.oo.position = {
            x: Framework.Game.getCanvasWidth() / 4,
            y: Framework.Game.getCanvasHeight() / 17 * 10
        };
        this.bott.position = {
            x: Framework.Game.getCanvasWidth() / 2,
            y: Framework.Game.getCanvasHeight() / 7 * 6
        };
        this.photo.position = {
            x: 0,
            y: 0
        };
        this.center.attach(this.photo);

        //rootScene 為系統預設的容器，由於其他東西都被 attach 到 center 上
        //將物件 attach 到 center 上，順序是會影響繪製出來的效果的
        this.rootScene.attach(this.center);
        this.rootScene.attach(this.bott);
        this.rootScene.attach(this.oo);
        this.photo.start();
        this.audio = new Framework.Audio({
            kick: {
                mp3: define.musicPath + 'kick2.mp3',
```

```

        //ogg: define.musicPath + 'kick2.ogg',
        //wav: define.musicPath + 'kick2.wav'
    }, song1: {
        mp3: define.musicPath + 'Pucker_Up.mp3',
        //ogg: define.musicPath + 'Hot_Heat.ogg',
        //wav: define.musicPath + 'Hot_Heat.wav'
    }
    },
    //播放時，需要給 name，其餘參數可參考 W3C
    this.audio.play({ name: 'song1', loop: true });

    },
    initialize: function() {
    },
    update: function() {
        this.rootScene.update();
        this.oo.update();
        this.bott.update();
    },
    draw: function(parentCtx) {
        this.rootScene.draw(parentCtx);
    },

    mouseup: function(e) {
        this.isTouch = false;
        this.isTouch1 = false;
    },

    mousemove: function(e) {
        if (e) {
            console.log(e.x, e.y);
        }
        this.optionTouch = { x: e.x, y: e.y };
        if (this.optionTouch.x > this.oo.upperLeft.x && this.optionTouch.x < this.oo.upperRight.x &&
this.optionTouch.y > this.oo.upperLeft.y && this.optionTouch.y < this.oo.lowerLeft.y) {
            this.oo.start({ from: 1, to: 1, loop: true, speed: 0.5 });
            this.isTouch1 = true;
        }
        else {
            this.oo.start({ from: 0, to: 0, loop: true, speed: 0.5 });
            this.isTouch1 = false;
        }
        this.previousTouch = { x: e.x, y: e.y };
        if (this.previousTouch.x > this.bott.upperLeft.x && this.previousTouch.x < this.bott.upperRight.x &&
this.previousTouch.y > this.bott.upperLeft.y && this.previousTouch.y < this.bott.lowerLeft.y) {
            console.log(e.x, e.y);
            this.bott.start({ from: 1, to: 1, loop: true, speed: 0.5 });
            this.isTouch = true;
        }
        else {
            this.bott.start({ from: 0, to: 0, loop: true, speed: 0.5 });
            this.isTouch = false;
        }
    },

    mousedown: function(e) {
        if ((this.isTouch1 == 1) && (this.optionTouch.x > this.oo.upperLeft.x && this.optionTouch.x <
this.oo.upperRight.x && this.optionTouch.y > this.oo.upperLeft.y && this.optionTouch.y < this.oo.lowerLeft.y)) {
            this.audio.stopAll();
            Framework.Game.goToLevel('option');
        }
        if ((this.isTouch == 1) && (this.previousTouch.x > this.bott.upperLeft.x && this.previousTouch.x <
this.bott.upperRight.x && this.previousTouch.y > this.bott.upperLeft.y && this.previousTouch.y < this.bott.lowerLeft.y)) {
            {
                this.audio.stopAll();
                Framework.Game.goToNextLevel();
            }
        }
        this.previousTouch = this.currentTouch;
    }

```

```
        this.optionTouch = this.currentTouch;
    },
    mouseup: function(e) {
        this.isTouchArrow = false;
        this.isTouch_option = false;
    },
    touchstart: function (e) {
        this.mousedown({ x: e.touches[0].clientX, y: e.touches[0].clientY });
    },
    touchend: function (e) {
        this.mouseup();
    },
    });
```

myGameLevel1

```
var MyGame = Framework.Class(Framework.Level , {
    load: function(){
        var characterPosition;
        this.gameMap = new GameMap();
        this.gameMap.load(this.rootScene);
        this.rootScene.attach(this.gameMap);
        this.score = new score();
        this.score.load(this.rootScene);
        this.bomb = [];
        // this.rootScene.attach(this.score);
        this.zombie = [];
        this.zombie[0]=new Zombies();
        this.zombie[0].load(this.rootScene,1);
        // this.zombie[2]=new Zombies();
        // this.zombie[2].load(this.rootScene);
        this.ZombieFreq = 0;
        this.ZombieNum = 1;
        this.entrance=1;
        this.shoot = [];
        this.MH=1000;
        this.MW=1000;
        this.level =20;
        this.die=0;
        this.move=3;
        this.bomb_timer=[];
        this.Zombie_speed=2;
        this.Zombie_out=40;
        MU=0;
        MD=0;
        ML=0;
        MR=0;
        this.bullet=1;
        shoot=0;
        shootdirection = [];
        judge=[];
        dis = [];
        Roledirection = 0;
        freq=0;
        this.pic = new Framework.Sprite(define.imagePath + 'character.png');
        this.roledown = new Framework.Sprite(define.imagePath + 'character.png');
        this.roleup = new Framework.Sprite(define.imagePath + 'roleup.png');
        this.roleleft = new Framework.Sprite(define.imagePath + 'roleleft.png');
        this.roleright = new Framework.Sprite(define.imagePath + 'roleright.png');
        this.roleru = new Framework.Sprite(define.imagePath + 'MRU.png');
        this.rolelu = new Framework.Sprite(define.imagePath + 'MLU.png');
        this.roleld = new Framework.Sprite(define.imagePath + 'MLD.png');
        this.rolerd = new Framework.Sprite(define.imagePath + 'MRD.png');
        this.role = this.pic;
        this.role.position = {x:800,y:500};
        this.rootScene.attach(this.role);
        // this.score = new score();
        // this.score.load(this.rootScene);
        this.shoot[0]=new Shoot();
        this.shoot[0].load(this.rootScene,this.role);
        this.shoot[0].update;
        this.shoot[0].detach(this.rootScene,0);
        this.speed =10;
        score_sum = 0;
        // this.score.update(score_sum);
        //宣告音樂檔
        this.audio = new Framework.Audio({
            kick: {
                mp3: define.musicPath + 'kick2.mp3',
```

```

    }, song1: {
        mp3: define.musicPath + 'Pucker_Up.mp3',
    }, shot: {
        mp3: define.musicPath + 'Gunshot.mp3',
    }, hit: {
        mp3: define.musicPath + 'hit.mp3',
    }, bgm: {
        mp3: define.musicPath + 'bgm.mp3',
    }, gameover1: {
        mp3: define.musicPath + 'Gameover1.mp3',
    }, gameover2: {
        mp3: define.musicPath + 'Gameover2.mp3',
    }, running: {
        mp3: define.musicPath + 'Running.mp3',
    }, zombie: {
        mp3: define.musicPath + 'zombie.mp3',
    },
    });
this.level_num = 1;
this.level_img = new Framework.AnimationSprite({url:[
    define.imagePath + 'Bomb.png',
    define.imagePath + 'Level1.png',
    define.imagePath + 'Level2.png',
    define.imagePath + 'Level3.png',
    define.imagePath + 'Level4.png',
    define.imagePath + 'Level5.png',
    define.imagePath + 'Level6.png',
    define.imagePath + 'Level7.png',
    define.imagePath + 'Level8.png',
    define.imagePath + 'Level9.png'],});
this.level_img.position = {x:1450,y:40};
this.level_img.start({from:1, to: 1});
this.rootScene.attach(this.level_img);
this.audio.play({ name: 'bgm', loop: true });
for(i=1;i<=100;i++)
{ //爆炸動畫
    this.bomb[i] = new Framework.AnimationSprite({url:[
        define.imagePath + 'Bomb.png',
        define.imagePath + 'explosion1.png',
        define.imagePath + 'explosion2.png',
        define.imagePath + 'explosion3.png',
        define.imagePath + 'explosion4.png',
        define.imagePath + 'explosion5.png',
        define.imagePath + 'explosion6.png',
        define.imagePath + 'explosion7.png',
        define.imagePath + 'explosion8.png',
        define.imagePath + 'explosion9.png',
        define.imagePath + 'explosion10.png',
        define.imagePath + 'explosion11.png',
        define.imagePath + 'explosion12.png',
        define.imagePath + 'explosion13.png',
        define.imagePath + 'explosion14.png',
        define.imagePath + 'explosion15.png',
        define.imagePath + 'explosion16.png',
        define.imagePath + 'explosion17.png',
        define.imagePath + 'explosion18.png',
        define.imagePath + 'explosion19.png',
        define.imagePath + 'explosion20.png'],});
    // this.bomb[i].position = {x:this.role.position.x,y:this.role.position.y};
    this.bomb_timer[i]=0;
}
this.bomb_put = 0;
this.bomb_num = 100;
// this.shot_sound = new Framework.Audio({

// });
// this.hit_sound = new Framework.Audio({

```

```

// });

//按下按鍵發生的事件
this.keydown = function(e,list){
    if(e.key === 'Right'){
        MR=1;
        this.audio.play({ name: 'running', loop: true });
    }
    if(e.key === 'Left'){
        ML=1;
        this.audio.play({ name: 'running', loop: true });
    }
    if(e.key === 'Up'){
        MU=1;
        this.audio.play({ name: 'running', loop: true });
    }
    if(e.key === 'Down'){
        MD=1;
        this.audio.play({ name: 'running', loop: true });
    }
    if(e.key === 'Space'){
        shoot=1;
        freq =15;
    }
    if(e.key === 'Z'){
        this.bomb_put = 1;
    }
};

//按鍵鬆開發生的事件
this.keyup = function(e,list)
{
    if(e.key === 'Right'){
        MR=0;
        this.audio.stop('running');
    }
    if(e.key === 'Left'){
        ML=0;
        this.audio.stop('running');
    }
    if(e.key === 'Up'){
        MU=0;
        this.audio.stop('running');
    }
    if(e.key === 'Down'){
        MD=0;
        this.audio.stop('running');
    }
    if(e.key === 'Space'){
        shoot=0;
    }
    if(e.key === 'Z'){
        this.bomb_put = 0;
    }
};

this.update = function(){

    this.score.update(score_sum);
    this.ZombieFreq ++;
    //殭屍生成
    if ((this.ZombieFreq>=this.Zombie_out)&&(this.ZombieNum < this.level))
    {
        this.entrance=this.ZombieNum % 4 ;
        this.ZombieFreq = 0;
        this.zombie[this.ZombieNum]= new Zombies();
        this.zombie[this.ZombieNum].load(this.rootScene,this.entrance);
        this.ZombieNum++;
    }
}

```



```

if ((this.bomb_put==1)&&(this.bomb_num>0))
{ //放置炸彈
    this.rootScene.attach(this.bomb[this.bomb_num]);
    this.bomb[this.bomb_num].position = { x:this.role.position.x,y:this.role.position.y};
    this.bomb[this.bomb_num].start({ from:0, to: 0, loop: true, speed: 20});
    this.bomb_put = 0;
    this.bomb_timer[this.bomb_num]=1;
    this.bomb_num--;
}
for(i=100;i>0;i--)
{
    if((this.bomb_timer[i] != 0 ) && (this.bomb_timer[i]<=100))
    {
        console.log(this.bomb_timer[i]);
        this.bomb_timer[i]++;
    }
    if(this.bomb_timer[i]==100)
    { //炸彈爆炸
        console.log(this.bomb_timer[i]);
        this.rootScene.attach(this.bomb[this.bomb_num]);
        this.bomb[i].start({ from:0, to: 20, loop: false, speed: 20});
    }
}
if(this.die == this.level){
    this.level+=10;
    this.level_num++;
    this.level_img.start({ from:this.level_num, to: this.level_num});
    this.die=0;
    this.ZombieNum=0;
    this.Zombie_speed++;
}
if(this.level>=40)
{ //改變殭屍生成頻率
    this.Zombie_out=30;
}
else if(this.level>=60)
{
    this.Zombie_out=15;
}
for(i=1;i<this.bullet;i++)
{
    if(judge[i]==1)
    {
        for (j = 0 ; j < this.ZombieNum ; j++) //殭屍被射中
        {
            if(((this.shoot[i].shoot.position.x > this.zombie[j].zombie.position.x -20)
            &&(this.shoot[i].shoot.position.x < this.zombie[j].zombie.position.x +20))
            &&(this.shoot[i].shoot.position.y < this.zombie[j].zombie.position.y+20 )
            &&(this.shoot[i].shoot.position.y > this.zombie[j].zombie.position.y-20 )
            &&(this.zombie[j].hurttime != 0)){
                console.log(shootdirection[i]);
                if(shootdirection[i] == 1)
                {
                    this.zombie[j].zombie.position={ x:this.zombie[j].zombie.position.x,y:this.zombie[j].zombie.position.y -this.speed };
                }
                if(shootdirection[i] == 2)
                {
                    this.zombie[j].zombie.position={ x:this.zombie[j].zombie.position.x,y:this.zombie[j].zombie.position.y +this.speed };
                }
                if(shootdirection[i] == 3)
                {
                    this.zombie[j].zombie.position={ x:this.zombie[j].zombie.position.x
                    -this.speed ,y:this.zombie[j].zombie.position.y };
                }
                if(shootdirection[i] == 4)
                {

```

```

        this.zombie[j].position = {x:this.zombie[j].zombie.position.x
+this.speed ,y:this.zombie[j].zombie.position.y };
    }
    if(shootdirection[i] == 31)
    {
        this.zombie[j].position = {x:this.zombie[j].zombie.position.x
-this.speed*0.707,y:this.zombie[j].zombie.position.y -this.speed*0.707};
    }
    if(shootdirection[i] == 32)
    {
        this.zombie[j].position = {x:this.zombie[j].zombie.position.x
-this.speed*0.707,y:this.zombie[j].zombie.position.y +this.speed*0.707};
    }
    if(shootdirection[i] == 41)
    {
        this.zombie[j].position = {x:this.zombie[j].zombie.position.x +
this.speed*0.707,y:this.zombie[j].zombie.position.y - this.speed*0.707};
    }
    if(shootdirection[i] == 42)
    {
        this.zombie[j].position = {x:this.zombie[j].zombie.position.x +
this.speed*0.707,y:this.zombie[j].zombie.position.y + this.speed*0.707};
    }
    this.zombie[j].hurt(this.rootScene);
    this.audio.play({ name: 'hit', loop: false });
    if(this.zombie[j].hurttime<=0){
        this.audio.play({ name: 'zombie', loop: false });
        this.die++;
        score_sum ++;
    }
    judge[i]=0;
    shoot[i]=0;
    this.shoot[i].detach(this.rootScene,i);

    // this.score.update(this.score_num);
    break;
}
else
{
    this.shoot[i].update(shootdirection[i]);
    dis[i]++;
}
}
if(dis[i]>600)
{
    // console.log(dis[i]);
    shoot[i]=0;
    judge[i]=0;
}
}
else if(judge[i]== 0){ //沒射中殭屍時子彈的處理
    // console.log(dis[i]);
    this.shoot[i].detach(this.rootScene,i);
    shoot[i]=0;
    judge[i]=2;
}
}

// this.zombie[2].update(this.role,this.rootScene);
// this.zombie.update(this.role,this.rootScene);
for (i = 0 ; i < this.ZombieNum ; i++)
{
    this.zombie[i].update(this.role,this.rootScene,this.Zombie_speed);
    if((this.zombie[i].zombie.position.x == this.role.position.x)&&(this.zombie[i].zombie.position.y ==
this.role.position.y)&&(this.zombie[i].hurttime != 0))
    {
        Framework.Game.goToNextLevel(score_sum);
        this.audio.stopAll();
    }
}

```

```

        this.audio.play({ name: 'gameover1', loop: false });
        this.audio.play({ name: 'gameover2', loop: false });
    }
}
roleX=this.role.position.x ;
roleY=this.role.position.y ;
this.rootScene.detach(this.role);
if ((MU==1)&&(ML==1)) { //role go to left_up
    this.role.position = {
        x: this.role.position.x -2,
        y: this.role.position.y - 2
    };
    this.rolelu.position = this.role.position;
    this.role = this.rolelu;
    Roledirection = 31;
}

else if ((MU == 1) && (MR == 1)) { //role go to right_up
    this.role.position = {
        x: this.role.position.x + 2,
        y: this.role.position.y - 2
    };
    this.roleru.position = this.role.position;
    this.role = this.roleru;
    Roledirection = 41;
}

else if ((MD == 1) && (ML == 1)) { //role go to left_down
    this.role.position = {
        x: this.role.position.x - 2,
        y: this.role.position.y + 2
    };
    this.roleld.position = this.role.position;
    this.role = this.roleld;
    Roledirection = 32;
}

else if ((MD == 1) && (MR == 1)) { //role go to right_down
    this.role.position = {
        x: this.role.position.x + 2,
        y: this.role.position.y + 2
    };
    this.rolerd.position = this.role.position;
    this.role = this.rolerd;
    Roledirection = 42;
}
else if(MR==1){
    this.role.position = {
        x: this.role.position.x +2,
        y: this.role.position.y
    };
    this.roleright.position = this.role.position;
    this.role = this.roleright;
    Roledirection = 4;
}
else if(ML==1){
    this.role.position = {
        x: this.role.position.x -2,
        y: this.role.position.y
    };
    this.roleleft.position = this.role.position;
    this.role = this.roleleft;
    Roledirection = 3;
}
else if(MU==1){
    this.role.position = {
        x: this.role.position.x ,
        y: this.role.position.y -2
    };
}

```

```

    };
    this.roleup.position = this.role.position;
    this.role = this.roleup;
    Roledirection = 1;
}
else if(MD==1){
    this.role.position = {
        x: this.role.position.x ,
        y: this.role.position.y +2
    };
    this.roledown.position = this.role.position;
    this.role = this.roledown;
    Roledirection = 2;
}

if ((shoot == 1) && (freq == 15 ))
{
    this.shoot[this.bullet]=new Shoot();
    this.audio.play({ name: 'shot', loop: false });
    this.audio.setVolume('shot', 0.5);
    shootdirection[this.bullet]=Roledirection;
    this.shoot[this.bullet].load(this.rootScene,this.role);
    dis[this.bullet]=0;
    judge[this.bullet]=1
    this.bullet++;
    freq = 0;
}
else if ((shoot == 1) && (freq < 15 ))
{
    freq++;
}

//for (var item in shootdirection) this.rootScene.detach(item);
//設定入口通道
if ((this.role.position.x <= 70) && (this.role.position.y >= 385) && (this.role.position.y <= 480)) {
    if(this.role.position.x<=10){
        this.role.position = {
            x: 1550,
            y: this.role.position.y
        };
    }
}
if ((this.role.position.x <= 70) && ((this.role.position.y <= 385) || (this.role.position.y >= 480))) {
    this.role.position = {
        x: 70,
        y: this.role.position.y
    };
}
if ((this.role.position.x >= 1540) && (this.role.position.y >= 385) && (this.role.position.y <= 480))
{
    if (this.role.position.x >= 1570) {
        this.role.position = {
            x: 10,
            y: this.role.position.y
        };
    }
}
if ((this.role.position.x >= 1540) && ((this.role.position.y <= 385) || (this.role.position.y >= 480))) //?k???
{
    this.role.position = {
        x: 1540,
        y: this.role.position.y
    };
}
if ((this.role.position.y <= 80) && (this.role.position.x >= 760) && (this.role.position.x <= 895))
{
    if (this.role.position.y <= 10) {
        this.role.position = {

```

```

        x: this.role.position.x,
        y: 835
    };
    }
}
if ((this.role.position.y <= 80) && ((this.role.position.x <= 760) || (this.role.position.x >= 895)))
{
    this.role.position = {
        x: this.role.position.x,
        y: 80
    };
}
if ((this.role.position.y >= 825) && (this.role.position.x >= 760) && (this.role.position.x <= 895))
{
    if (this.role.position.y >= 845) {
        this.role.position = {
            x: this.role.position.x,
            y: 10
        };
    }
}
if ((this.role.position.y >= 825) && ((this.role.position.x <= 760) || (this.role.position.x >= 895)))
{
    this.role.position = {
        x: this.role.position.x,
        y: 825
    };
}
this.rootScene.attach(this.role);
};
},
});

```

option

```
var Option = Framework.Class(Framework.Level, {

  load: function () {
    this.option = new Framework.Sprite(define.imagePath + 'Boxhead-1.jpg');
    this.option.position = {
      x: 800,
      y: 450
    };
    this.back = new Framework.Sprite(define.imagePath + 'back.png');
    this.back.position = {
      x: 110,
      y: 110
    };
    this.rootScene.attach(this.option);
    this.rootScene.attach(this.back);
    this.previousTouch = { x: 0, y: 0 };
    this.isTouch = false;
    this.audio = new Framework.Audio({
      song2: {
        mp3: define.musicPath + 'Evil_March.mp3',
      },
    });
    this.audio.play({ name: 'song2', loop: true });
  },
  draw: function (parentCtx) {
    parentCtx.font = '50pt bold';
    parentCtx.fillStyle = 'blue';
    parentCtx.textAlign = 'left';
    parentCtx.fillText("Press 'space' is attack", 10, 475);
    parentCtx.fillText("Press 'Z' is set time bomb", 10, 540);
    parentCtx.fillText("Press 'Arrow key' can control direction", 10, 605);
    parentCtx.font = '55pt bold';
    parentCtx.textAlign = 'center';
    parentCtx.fillStyle = 'yellow';
    parentCtx.fillText("Please press 'Back' and play the game just now !!!", 800, 750);
  },
  mousemove: function (e) {
    this.previousTouch = { x: e.x, y: e.y };
    if (this.previousTouch.x > this.back.upperLeft.x && this.previousTouch.x < this.back.upperRight.x &&
    this.previousTouch.y > this.back.upperLeft.y && this.previousTouch.y < this.back.lowerLeft.y) {
      this.isTouch = true;
    }
    else {
      this.isTouch = false;
    }
  },
  mousedown: function (e) {
    if ((this.isTouch == 1) && (this.previousTouch.x > this.back.upperLeft.x && this.previousTouch.x <
    this.back.upperRight.x && this.previousTouch.y > this.back.upperLeft.y && this.previousTouch.y <
    this.back.lowerLeft.y)) {
      this.audio.stopAll();
      Framework.Game.goToLevel('menu');
    }
  },
  mouseup: function (e) {
    this.isTouch = false;
  },
});
```

SCORE

```
var score = function(){

    this.load=function(rootScene){
        this.digit = new Framework.AnimationSprite({ url:[define.imagePath + 'zero.png', define.imagePath +
'one.png',define.imagePath + 'two.png',define.imagePath + 'three.png',
        define.imagePath + 'four.png', define.imagePath + 'five.png', define.imagePath + 'six.png', define.imagePath +
'seven.png', define.imagePath + 'eight.png', define.imagePath + 'nine.png']});
        this.tens = new Framework.AnimationSprite({ url:[define.imagePath + 'zero.png', define.imagePath +
'one.png',define.imagePath + 'two.png',define.imagePath + 'three.png',
        define.imagePath + 'four.png', define.imagePath + 'five.png', define.imagePath + 'six.png', define.imagePath +
'seven.png', define.imagePath + 'eight.png', define.imagePath + 'nine.png']});
        this.hundred = new Framework.AnimationSprite({ url:[define.imagePath + 'zero.png', define.imagePath +
'one.png',define.imagePath + 'two.png',define.imagePath + 'three.png',
        define.imagePath + 'four.png', define.imagePath + 'five.png', define.imagePath + 'six.png', define.imagePath +
'seven.png', define.imagePath + 'eight.png', define.imagePath + 'nine.png']});
        this.digit.start({ from:0, to: 0});
        this.tens.start({ from:0, to: 0});
        this.hundred.start({ from:0, to: 0});
        this.digit.position = { x:300,y:30};
        this.tens.position = { x:200,y:30};
        this.hundred.position = { x:100,y:30};
        rootScene.attach(this.digit);
        rootScene.attach(this.tens);
        rootScene.attach(this.hundred);
        this.hundred.num = 0;
        this.tens.num = 0;
        this.digit.num = 0;
    };

    this.update=function(sum){
        this.digit.position = { x:300,y:180};
        this.tens.position = { x:200,y:180};
        this.hundred.position = { x:100,y:180};
        this.digit.num = sum% 10;
        if(sum>=10)
        {
            this.tens.num = ((sum-this.digit.num)/10)% 10;
            if(sum>=100)
            {
                this.hundred.num = ((sum-this.digit.num-this.tens.num*10)/100)% 10;
            }
            else
            {
                this.hundred.num = 0;
            }
        }
        else
        {
            this.tens.num = 0;
        }
        this.digit.start({ from:this.digit.num, to: this.digit.num});
        this.tens.start({ from:this.tens.num, to: this.tens.num});
        this.hundred.start({ from:this.hundred.num, to: this.hundred.num});
    };
};
```

Shoot

```
var Shoot = function(){

    this.load = function (rootScene, role) {
        this.shoot = new Framework.Sprite(define.imagePath + 'bullet.png');
        this.shoot.position = {x:role.position.x, y:role.position.y};
        rootScene.attach(this.shoot);
    };

    this.update=function(Roledirection){
        this.speed = 2;

        if(Roledirection == 1)
        {
            this.shoot.position = {x:this.shoot.position.x ,y:this.shoot.position.y - this.speed};
        }
        if(Roledirection == 2)
        {
            this.shoot.position = {x:this.shoot.position.x ,y:this.shoot.position.y + this.speed};
        }
        if(Roledirection == 3)
        {
            this.shoot.position = {x:this.shoot.position.x - this.speed,y:this.shoot.position.y };
        }
        if(Roledirection == 4)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed,y:this.shoot.position.y };
        }
        if(Roledirection == 31)
        {
            this.shoot.position = {x:this.shoot.position.x -this.speed*0.707,y:this.shoot.position.y -this.speed*0.707};
        }
        if(Roledirection == 32)
        {
            this.shoot.position = {x:this.shoot.position.x -this.speed*0.707,y:this.shoot.position.y +this.speed*0.707};
        }
        if(Roledirection == 41)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed*0.707,y:this.shoot.position.y -
this.speed*0.707};
        }
        if(Roledirection == 42)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed*0.707,y:this.shoot.position.y +
this.speed*0.707};
        }
    };

    this.detach=function(rootScene,i){
        rootScene.detach(this.shoot);
        delete this.shoot;
    };
};
```


zombie

```
this.load=function(rootScene,entrance){
    this.zombie = new Framework.AnimationSprite({url:[define.imagePath + 'zombieDown.png', define.imagePath +
'zombieUp.png',define.imagePath + 'zombieLeft.png',define.imagePath + 'zombieRight.png',
    define.imagePath + 'ZLD.png', define.imagePath + 'ZRU.png', define.imagePath + 'ZRD.png', define.imagePath +
'ZLU.png']});
    this.zombie.start({from:0, to: 0});
    // this.blood = [5];
    this.bloodback = new Framework.Sprite(define.imagePath + 'BLOOD_BACK.png');

    this.blood0 = new Framework.Sprite(define.imagePath + 'blood.png');
    this.blood1 = new Framework.Sprite(define.imagePath + 'blood.png');
    this.blood2 = new Framework.Sprite(define.imagePath + 'blood.png');
    this.blood3 = new Framework.Sprite(define.imagePath + 'blood.png');
    this.blood4 = new Framework.Sprite(define.imagePath + 'blood.png');

    this.blood0.position = {x:this.zombie.position.x-24,y:this.zombie.position.y-35};
    this.blood1.position = {x:this.zombie.position.x-12,y:this.zombie.position.y-35};
    this.blood2.position = {x:this.zombie.position.x,y:this.zombie.position.y-35};
    this.blood3.position = {x:this.zombie.position.x+12,y:this.zombie.position.y-35};
    this.blood4.position = {x:this.zombie.position.x+24,y:this.zombie.position.y-35};

    this.bloodback.position = {x:this.zombie.position.x,y:this.zombie.position.y-35};
    if (entrance == 0)
    {
        this.zombie.position = {x:800,y:-10};
    }
    else if(entrance == 1)
    {
        this.zombie.position = {x:800,y:1000};
    }
    else if(entrance == 2)
    {
        this.zombie.position = {x:-10,y:480};
    }
    else if(entrance == 3)
    {
        this.zombie.position = {x:1610,y:480};
    }
    rootScene.attach(this.zombie);
    rootScene.attach(this.bloodback);
    rootScene.attach(this.blood0);
    rootScene.attach(this.blood1);
    rootScene.attach(this.blood2);
    rootScene.attach(this.blood3);
    rootScene.attach(this.blood4);
    this.time=0;
    this.speed =15;
    this.hurttime=5;
};

this.update=function(role,rootScene){
    var roleX=role.position.x ;
    var roleY=role.position.y ;
    this.time++;
    this.speed = 2;
    this.blood0.position = {x:this.zombie.position.x-24,y:this.zombie.position.y-35};
    this.blood1.position = {x:this.zombie.position.x-12,y:this.zombie.position.y-35};
    this.blood2.position = {x:this.zombie.position.x,y:this.zombie.position.y-35};
    this.blood3.position = {x:this.zombie.position.x+12,y:this.zombie.position.y-35};
    this.blood4.position = {x:this.zombie.position.x+24,y:this.zombie.position.y-35};
    this.bloodback.position = {x:this.zombie.position.x,y:this.zombie.position.y-35};
    if (((this.zombie.position.x > roleX ) && (this.zombie.position.y == roleY )) && (this.time==2))
    {
```

```

        this.zombie.position = {
            x : this.zombie.position.x - this.speed,
            y : this.zombie.position.y
        };
        this.zombie.start({ from:2, to: 2});
        this.time=0;
    }
    if (((this.zombie.position.x < roleX ) && (this.zombie.position.y == roleY ))&& (this.time==2))
    {
        this.zombie.position = {
            x : this.zombie.position.x + this.speed,
            y : this.zombie.position.y
        };
        this.zombie.start({ from:3, to: 3});
        this.time=0;
    }
    if (((this.zombie.position.y > roleY ) && (this.zombie.position.x == roleX ))&& (this.time==2))
    {
        this.zombie.position = {
            x : this.zombie.position.x ,
            y : this.zombie.position.y -this.speed
        };
        this.zombie.start({ from:1, to: 1 });
        this.time=0;
    }
    if (((this.zombie.position.y < roleY ) && (this.zombie.position.x == roleX ))&& (this.time==2))
    {
        this.zombie.position = {
            x : this.zombie.position.x ,
            y : this.zombie.position.y +this.speed
        };
        this.zombie.start({ from:0, to: 0});
        this.time=0;
    }
    if (((this.zombie.position.y < roleY ) && (this.zombie.position.x > roleX ))&& (this.time==2)) //LD
    {
        this.zombie.position = {
            x : this.zombie.position.x -this.speed * 0.707,
            y : this.zombie.position.y +this.speed * 0.707
        };
        this.zombie.start({ from:4, to: 4});
        this.time=0;
    }
    if (((this.zombie.position.y > roleY ) && (this.zombie.position.x > roleX ))&& (this.time==2)) //LU
    {
        this.zombie.position = {
            x : this.zombie.position.x -this.speed * 0.707,
            y : this.zombie.position.y -this.speed * 0.707
        };
        this.zombie.start({ from:7, to: 7});
        this.time=0;
    }
    if (((this.zombie.position.y < roleY ) && (this.zombie.position.x < roleX ))&& (this.time==2)) //RD
    {
        this.zombie.position = {
            x : this.zombie.position.x +this.speed * 0.707,
            y : this.zombie.position.y +this.speed * 0.707
        };
        this.zombie.start({ from:6, to: 6});
        this.time=0;
    }
    if (((this.zombie.position.y > roleY ) && (this.zombie.position.x < roleX ))&& (this.time==2)) //RU
    {
        this.zombie.position = {
            x : this.zombie.position.x +this.speed * 0.707,
            y : this.zombie.position.y -this.speed * 0.707
        };
        this.zombie.start({ from:5, to: 5});
    }

```

```

        this.time=0;
    }

    if (this.time>2)
    {
        this.time=0;
    }
};

this.hurt=function(rootScene){
    this.hurttime--;
    if(this.hurttime==4)
    {
        rootScene.detach(this.blood4);
    }
    else if(this.hurttime==3)
    {
        rootScene.detach(this.blood3);
    }
    else if(this.hurttime==2)
    {
        rootScene.detach(this.blood2);
    }
    else if(this.hurttime==1)
    {
        rootScene.detach(this.blood1);
    }
    else if(this.hurttime<=0)
    {
        rootScene.detach(this.blood0);
        rootScene.detach(this.zombie);
        rootScene.detach(this.bloodback);
    }
};

};

        this.hundred.num = ((sum-this.digit.num-this.tens.num*10)/100)% 10;
    }
    else
    {
        this.hundred.num = 0;
    }
}
else
{
    this.tens.num = 0;
}
this.digit.start({ from:this.digit.num, to: this.digit.num});
this.tens.start({ from:this.tens.num, to: this.tens.num});
this.hundred.start({ from:this.hundred.num, to: this.hundred.num});
};

};
var Shoot = function(){

    this.load = function (rootScene, role) {
        this.shoot = new Framework.Sprite(define.imagePath + 'bullet.png');
        this.shoot.position = {x:role.position.x, y:role.position.y};
        rootScene.attach(this.shoot);
    };

    this.update=function(Roledirection){
        this.speed = 2;

        if(Roledirection == 1)
        {
            this.shoot.position = {x:this.shoot.position.x ,y:this.shoot.position.y - this.speed};
        }
    }
};

```

```

        if(Roledirection == 2)
        {
            this.shoot.position = {x:this.shoot.position.x ,y:this.shoot.position.y + this.speed};
        }
        if(Roledirection == 3)
        {
            this.shoot.position = {x:this.shoot.position.x - this.speed,y:this.shoot.position.y };
        }
        if(Roledirection == 4)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed,y:this.shoot.position.y };
        }
        if(Roledirection == 31)
        {
            this.shoot.position = {x:this.shoot.position.x -this.speed*0.707,y:this.shoot.position.y -this.speed*0.707};
        }
        if(Roledirection == 32)
        {
            this.shoot.position = {x:this.shoot.position.x -this.speed*0.707,y:this.shoot.position.y +this.speed*0.707};
        }
        if(Roledirection == 41)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed*0.707,y:this.shoot.position.y -
this.speed*0.707};
        }
        if(Roledirection == 42)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed*0.707,y:this.shoot.position.y +
this.speed*0.707};
        }
    };

    this.detach=function(rootScene,i){
        rootScene.detach(this.shoot);
        delete this.shoot;
    };
};
var Shoot = function(){

    this.load = function (rootScene, role) {
        this.shoot = new Framework.Sprite(define.imagePath + 'bullet.png');
        this.shoot.position = {x:role.position.x, y:role.position.y};
        rootScene.attach(this.shoot);
    };

    this.update=function(Roledirection){
        this.speed = 2;

        if(Roledirection == 1)
        {
            this.shoot.position = {x:this.shoot.position.x ,y:this.shoot.position.y - this.speed};
        }
        if(Roledirection == 2)
        {
            this.shoot.position = {x:this.shoot.position.x ,y:this.shoot.position.y + this.speed};
        }
        if(Roledirection == 3)
        {
            this.shoot.position = {x:this.shoot.position.x - this.speed,y:this.shoot.position.y };
        }
        if(Roledirection == 4)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed,y:this.shoot.position.y };
        }
        if(Roledirection == 31)
        {
            this.shoot.position = {x:this.shoot.position.x -this.speed*0.707,y:this.shoot.position.y -this.speed*0.707};
        }
    }
};

```

```
        if(Roledirection == 32)
        {
            this.shoot.position = {x:this.shoot.position.x -this.speed*0.707,y:this.shoot.position.y +this.speed*0.707};
        }
        if(Roledirection == 41)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed*0.707,y:this.shoot.position.y -
this.speed*0.707};
        }
        if(Roledirection == 42)
        {
            this.shoot.position = {x:this.shoot.position.x + this.speed*0.707,y:this.shoot.position.y +
this.speed*0.707};
        }
    };

    this.detach=function(rootScene,i){
        rootScene.detach(this.shoot);
        delete this.shoot;
    };
};
```