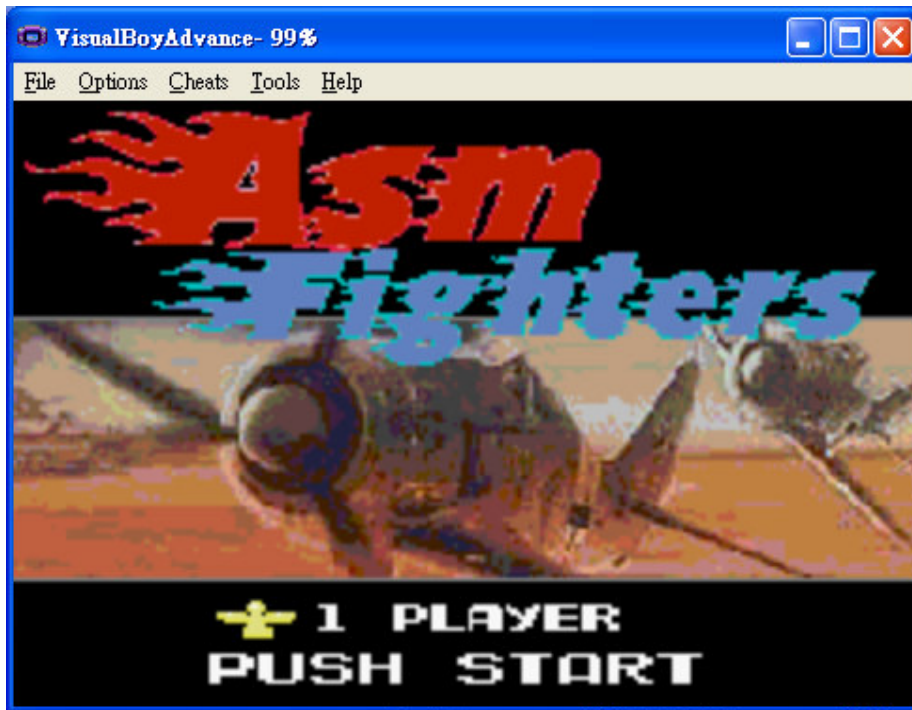


Computer Organization and Assembly Languages

Final Project

Asm Fighters



資工二 B95902004 洪三權

資工二 B95902022 莊智涵

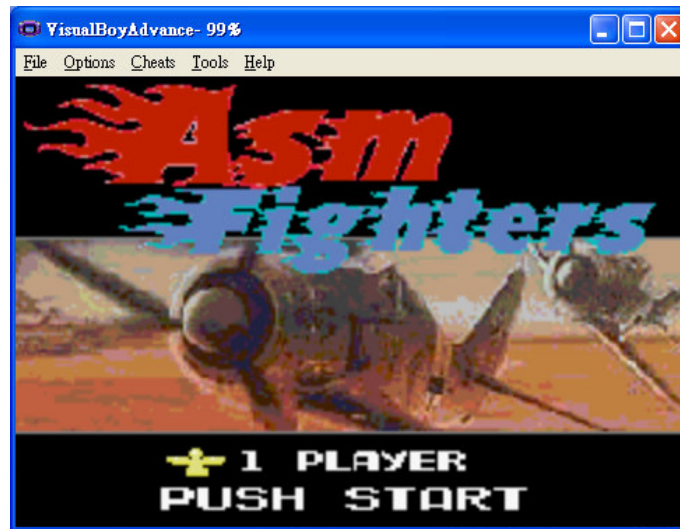
資工二 B95902064 蔡青樺

一、設計環境

Visual HAM 2.4.2



二、遊戲內容與特色



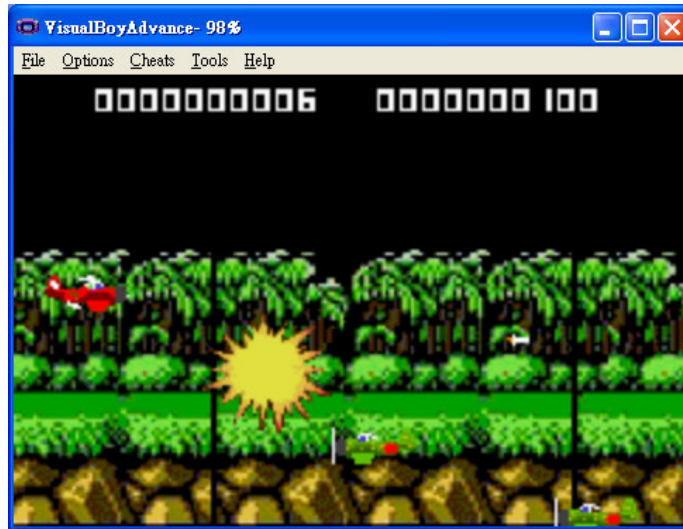
起始畫面

總共有一關。剛開始，邪惡特攻隊會隨機產生，並且瞄準 ASM 號俯衝，英勇又聰明的玩家必須巧妙的閃躲以及反擊，每射擊一架敵機，右上角的分數會增加 10 分。當超過一定時間，或者是分數超過一百，則魔王轟炸機將會由右往左緩緩進場。每隔一定時間，會多方向射擊飛彈，請聰明的玩家必須小心。當魔王轟炸機遭受到一定次數的轟擊之後將會墜落，而本遊戲也將結束。

遊戲途中，若是不小心遭敵軍撞擊，或者是遭碰觸到敵方飛彈，ASM 號機將會爆炸而提前遊戲結束，玩家請務必注意。

衝鋒吧！英勇的 ASM 號！





遊戲畫面(ASM 號、子彈與邪惡特攻隊遭受攻擊後爆炸)



遊戲畫面(ASM 號、魔王轟炸機與雙方子彈以及邪惡特攻隊)

特色一：記分欄與時間欄

在螢幕的右上角是玩家目前的分數欄，而左上角是目前已經過的時間。由於HAM內建的 ham_DrawText 有 bug，所以決定重寫。

特色二：會追蹤玩家的邪惡特攻隊



每一次螢幕重刷(Vbl)，每一架特攻隊都會計算與玩家之間垂直距離，並做調整衝往玩家方向。

特色三：裝有散彈與導彈的魔王轟炸機



爲了增加遊戲的豐富度，每經過特定的間隔時間以後，魔王轟炸機會向玩家發射三發散彈與一發導彈。導彈發射之前計算發射點與玩家之間的 cot 值，作爲子彈行進的依據，而散彈則是固定角度分別爲 135 度、180 度、225 度。

特色四：捲動的樹林背景

相信眼尖的玩家一定會注意到，本背景修改於魂斗羅的首關背景。由於 ASM 號是飛機，我們希望藉由背景向左捲動讓玩家有飛機向右飛的感覺。



結束畫面(ASM 號墜落)



破關畫面

三、使用說明

按鍵	效果
↑	ASM 號往上
↓	ASM 號往下
→	ASM 號往右
←	ASM 號往左
A 鈕	發射飛彈

四、組語實作部份

1. 使用暫存器來讀取遊戲中的時間

內建的時間暫存器有四個：REG_TMxD，x=1,2,3,4，而控制其表現的暫存器也有對應四個：REG_TMxCNT，x=1,2,3,4。

REG_TMxCNT @ 0400:0102 + 4x																
F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
-							En	I	-				CM	Fr		

bits	name	define	description
0-1	Fr	TM_FREQ_y	Timer frequency. 0-3 for 1, 64, 256, or 1024 cycles, respectively. y in the define is the number of cycles.
2	CM	TM_CASCADE	Cascade mode. When the counter of the <i>preceding</i> (x-1) timer overflows (REG_TM(x-1)D= 0xffff), this one will be incremented too. A timer that has this bit set does <i>not</i> count on its own, though you still have to enable it. Obviously, this won't work for timer 0. If you plan on using it make sure you understand exactly what I just said; this place is a death-trap for the unwary.
6	I	TM_IRQ	Raise an interrupt on overflow.
7	En	TM_ON	Enable the timer.

註 1：此資料出自 <http://www.coranac.com/tonc/text/timers.htm>

由於 GBA 之處理器 1024 cycle 之頻率是 224Hz，所以 REG_TM0D 每跳 0x4000 = 224 次會需要 1 秒鐘。若把 REG_TMxD 設為 -0x4000，則每一秒鐘會 overflow 一次。把 REG_TM1CNT 的 cascade 設為 1(即 REG_TM1CNT = 0x0004)，則 REG_TM0D 每 overflow 一次會使 REG_TM1D 增加一。如此，我們可以得到一個每秒跳一次的時鐘。

再定義一個變數 pre，每更新一次畫面時儲存 REG_TM1D 的值，當 pre 和現在的 REG_TM1D 不一樣時，則表示時間又增加一秒，所以我們可以用來做以秒為單位的計時器。

2. 利用暫存器與組語偵測玩家之動作：

GBA 上面有四個按鍵：上(↑)、下(↓)、左(←)、右(→)，我們可以利用組合語言來偵測按鍵的動作。

首先先把指標指向要改變的目標，也就是玩家所控制的戰機。在此我們需要使戰機能夠上下左右移動位置，所以我們必須改變戰機物件的坐標。當玩家按下任何一個按鍵時，會更改在 0x40000130 這個位置的值，此位址個位元所代表之意義如下：

130h - KEYINPUT (formerly P1) - Key Status (R)

Bit	Expl.	
0	Button A	(0=Pressed, 1=Released)
1	Button B	(etc.)
2	Select	(etc.)
3	Start	(etc.)
4	Right	(etc.)
5	Left	(etc.)
6	Up	(etc.)
7	Down	(etc.)
8	Button R	(etc.)
9	Button L	(etc.)
10-15	Not used	

註 2：資料來源 http://www.ngine.de/hamdoc/korth_gbatek.html#keypadinput

當我們按下按鍵時，會使相對應的位置改成 0；當我們放開按鍵時，會使相對應的位置改成 1。所以我們在寫組語時，也必須將這個 0x00000130 位址引入。於是，我們可以把偵測按鍵並產生相對應的動作的部分交給組合語言。引入這些資料之後，利用 **CMP** 與 **Branch**，可以將原本的 **if...else** 結構轉換。

當按下「↑」鍵，使戰機向上移動(y--)並檢查邊界(y>=0)；當按下「↓」鍵，使戰機向下移動(y++)並檢查邊界(y<=240 - 32，32 為戰機寬度)，當按下「←」鍵，使戰機向左移動(x--)並檢查邊界(x>=0)，當按下「→」鍵，使戰機向右移動(y++)並檢查邊界(y>=140)。

另外，由於暫存器中關於負數是以最高的位元的值來表示，1 代表負數，0 代表正數，所以在組合語言中要判斷 $x < 0$ 或 $y < 0$ 之邊界時，需要先檢查最高位之值。

五、困難與解決方式

1. 加入圖片

必須先將圖片轉成 256 色的 bmp 檔，再用 **gfx2gba.exe** 轉檔，最後 **include** 進遊戲。剛開始因為不確定執行 **gfx2gba.exe** 時的參數應該要怎麼下，一直轉出壞掉的圖檔，卡了很久；將圖檔 **include** 進遊戲的時候也因為之前轉出的圖檔是壞掉的，一直以爲是 **include** 方式有問題。此外，因為不同的 **mode** 有不同的功能，還要研究在哪一個 **mode** 下圖檔才能正常顯示、背景才能正

常捲動。

2. 圖檔過大

因為背景圖檔過大導致記憶體用量過多，解決的方式是 reset memory，也就是呼叫 ham_ResetBg() 這個函式。

3. 物件過多

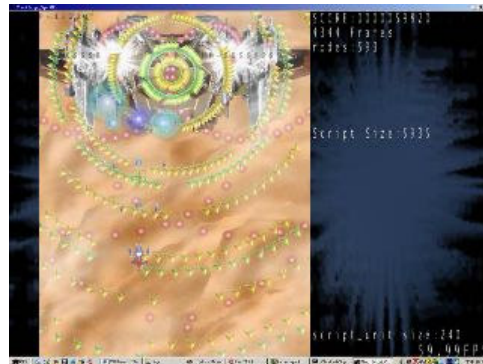
Object memory 的空間有限，若是一次需要顯示太多 object，則會出現記憶體不足的情況。

另外，因為我們使用「爆炸」此物件的時機是當我方飛機與敵方子彈相遇，或是當敵方飛機與我方子彈相遇，若是在判定「爆炸」之後未立即將子彈與飛機之坐標移出畫面，則下一枚子彈與該飛機會馬上相遇，如此便會在瞬間造成連續爆炸的情況。而這種情況會瞬間需要大量的記憶體來記憶新增的「爆炸」物件的位置，以致於記憶體不足。解決方式是在判定碰撞之後，顯示爆炸，並立即把兩物件移出邊界，以避免他們與其他的物件繼續發生反應

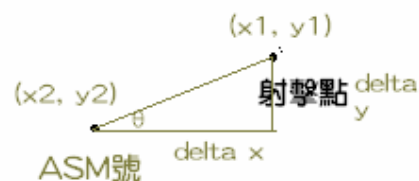
4. 彈幕

為了增加遊戲的趣味性，想要做出能射出多角度子彈的遊戲功能(如右圖)。一開始，參考網路上的資料，想要在每個子彈的 struct 變數中設定飛行角度($0^\circ \sim 360^\circ$)，每隔一定時間將子彈的 x, y 各加上 $\cos \theta, \sin \theta$ 即可。可是後來發現，gba 的螢幕座標是以整數為主，而 $\cos \theta, \sin \theta$ 卻是浮點數，在加減運算的時候會出問題(例如：並不存在(3.5, 4.5) 的座標)，為了解決這個問題，後來想到了另外一個方法。

如右圖所所示， $\cot \theta = (x1 - x2) / (y1 - y2)$ ，代表每隔多少的 delta x, y 的值就變動 1。讓每一顆



取自於 <http://kyovbdx.myweb.hinet.net/tutorials/ProgramShoot/new/tutorial01.htm>



子彈的 `struct` 變數記住其 $\cot \theta$ 值，每隔一個循環就變動 `y` 一次，形成子彈的軌道。

原本想要設計魔王轟炸機會發射 360 度彈幕，但實驗結果會造成遊戲太過於困難，使得玩家難以破關，而且 `Obj Memory` 也有可能過載而爆炸。最後只用來發射三發飛彈與一枚導彈，以避免玩家過不了關的窘境。

5. 邊界判定困難

因為原先將物件的座標用 `u8` 表示，即 0~255。這種表示方式在用組合語言編寫的那段會出現問題。因為 `u8` 無法表示負數，所以 `x--` 到零之後會從 255 再繼續減，導致物件會從一端消失，再突然出現在另一端的情況。因為出現這種狀況，我們把它改成 `s16`，並且利用檢查最高位來決定它的正負號。再利用檢查邊界的方式，便可以把它改成限制在螢幕範圍裡，不至於跑出螢幕外之方式。

6. 不熟悉 HAM 提供的函式 (`ham_DrawText()`有 bug!!!)

當我們要在螢幕上顯示字的時候，我們去查了一下 HAM 提供的函式。發現了 `ham_DrawText()` 這個函式。但是在遊戲撰寫途中，發現了利用此 `ham_DrawText()` 在某些情況下並不能正確地顯示我們所需要的字。另一方面，在此函式的說明文件中也有提到，這個函式非常的慢，通常是用於 `debug` 中。所以最後我們決定自己寫一個函式來顯示我們需要的字。包括遊戲進行中的分數與時鐘等，都是利用數字的圖檔組合而成的。

7. 無法加入音樂

製作遊戲的過程中，我們試過許許多多的方法希望可以加入背景音樂，卻受限於記憶體的大小而無法成功（我們有詢問過成功加入音效的組別，一致認為是因為我們的遊戲所包含的元件較多，所以無法額外加入音效）或者是放出來的音樂因為節奏不對而淪為噪音。請教別組之後，知道要更改 `makefile` 的設定，卻發現我們的 `makefile` 跟別組差異頗多；設定 `krawall` 之後，雖然 `project` 可以 `build`，不過模擬的時候卻連 HAM 的 `logo` 動畫都沒有顯示出來，陷入無法克服困難的狀況。

六、未來展望

1. 增加關卡

礙於時間緊迫，目前製作出來的遊戲只有一個關卡，未來可以製作更多的關卡，已增加遊戲的趣味性。

2. 加入音效

雖然我們試過許多加入音效的方法，卻受限於記憶體大小或放出的音樂節奏不對。不過我們仍然相信可以成功加入背景音樂的方法是存在的，如果未來有機會，我們希望可以研究出如何加入音效。

七、心得

1. 註解的存在十分重要！清楚的註解和有意義的變數名稱可以讓組員們在最短的時間內了解某段程式所代表的意義，也增加 DeBug 與更新程式時的效率。
2. 經過這段時間，深深感受到高階語言的簡潔和方便。即便是 C 語言，相對於組合語言也親切頗多。

八、感謝名單



九、參考資料

1. HAM online documentation

<http://www.ngine.de/hamdoc/index.html>

2. 掌上游戏机设计指南——GBA 程序设计初步掌握

<http://www.mcxb.com/sysProgram/Java/JavaJ2ME/140481.html>

3. HAM 教程翻译本

<http://dev.gameres.com/Program/Other/GBA/HamTutorial1.htm>

<http://bbs.ogdev.net/TopicContent.aspx?BoardID=2&TopicID=6177>

<http://www.gamego.cn/kafajishu/detail.asp?id=551>

<http://www.gamego.cn/kafajishu/detail.asp?id=552>

<http://www.linuxpk.com/19725.html>

4. 淺談射擊遊戲

<http://kyovbdx.myweb.hinet.net/tutorials/ProgramShoot/new/tutorial01.htm>

<http://kyovbdx.myweb.hinet.net/tutorials/ProgramShoot/new/tutorial02.htm>