

DOCUMENTATION

01/24/2008

GoGoCYY !



PRESENTED BY

B95902092 姚甯之

B95902040 蕭喬尹

B95902026 周冠汝

B94902056 馮維青

Documentation

01/24/2008

主題

THE DAY OF CYY

動機

這學期以 GBA 來學習 ARM Assembly 讓我們覺得十分有趣，我們尤其好奇那些有趣的遊戲究竟是怎麼做出來的。GBA 的 architecture 較為 simple，我們雖然無法像 TOY 那樣了解硬體實作，但以 programmer view 卻也是十分的底層了（相較於 INTEL Assembly）。在老師的大力推薦下我們也覺得這是相當有趣的期末 project。

然而，在閱讀 TONC 文件時其實遇到了不少的困難，雖然我們有盡力讀懂它所介紹的 GBA 原理了，但評估我們目前的 coding 實力，恐怕難以在短時間內以它做出一定 scale 的程式（大部分的心思恐怕都只能放在順利將 VRAM、OAM 對應到 output，而難以做出其他的遊戲元素）

當然，有了 TONC 的基礎再來看 HAM 的 implement 方式對我們也是個學習。當我們看了 HAM tutorial 教給我們的工具性操作，便更具體的了解 HAM 是如何把這個 we against memory 的 gap 包裝起來，使得開發效率大大提升。基於對 GBA 原理的了解也讓我們知道 HAM 封裝的部分可能對我們有甚麼負面影響，在有 bug 時也能適時改變作法。以上是我們轉向使用 HAM 開發的主因。

原理

有些上課已講過的 GBA Graphics 規格在此就不再贅述。我們看文件學習的主要是老師上課沒有提及的 BG 之 tile mode 和 sprite。BG 與 sprite 各有一個 256 色 16bit color 的調色盤，其實兩者的 .bmp data 都存在 VRAM 中，VRAM 可畫分為六個 charblock，前四個是儲存 tiles 和 tile map；後兩個是給 sprite 使用。Tile、sprite 與 bitmap mode 最大的不同就是他們存的是調色盤上的 entry（所以他們在螢幕上的每個 pixel 只需要用 8bit or 4bit【將一個 256 色切成 16 個小調色盤模式】表示即可），因此較省 memory，故遊戲開發大多採用 tile mode。

BG 如果使用了 tile mode，顯示就需要幾個稍嫌繁複的原理。

首先我們有 tile set【由數個 8*8 tile 集合而成的 array，每個 tile 在這個 array 中有個 index】
Tile map【由 tile index 組成的 array，剛好可對應到我們使用的 tile，map index

又對應到 GBA screen 上】

這是十分省空間的顯示方式，因為一張圖上其實有重複的元素。

至於 screen 可以捲動其實也是 GBA 硬體內建的功能，只要我們指名我們的 screen 左上角在 map 上的座標，螢幕顯示就是在該 map 位置。在 HAM 中這比較單純，可直接用 SetBGXY 去改變 SCREEN 在 MAP 中的絕對位置，而達到捲動的效果。

在 GBA 中是用 3 個 register 去 control 所有的 background，其中主要的是 REG_BGxCNT，此為一 16bit 的 register。其實我們藉由 ham_InitBG() 中設定去更改此 register 的內容，像是設定是否要模糊化、決定各 background 間的相對順序。在讀了 TONC 後，更能了解在 HAM 的函式背後真正所介入到的部分

GBA 中掌控 **SPRITE** 屬性的位置是 OAM (共可掌握 128 個 regular sprite)。若以 regular sprite 來說 (affine sprite 較難我們沒有接觸)，每個 sprite 在 OAM 中有三個 attribute，每個 attribute 兩 byte，它們在不等長的 bit field 中 carry 如下的 sprite 屬性

Attribute0 (y 座標 obj_mode 調色盤模式 shape 馬賽克功能 特殊 graphic 效果功能)

Attribute1 (x 座標 affine_index Hflip/Vflip size)

Attribute2 (data 在 VRAM 中的位址 priority 小調色盤 index)

所以說，想在螢幕上顯示 sprite 的步驟如下：

- 將 sprite 調色盤 load 進 memory
- 將 bmp data 存入 VRAM 的後兩個 charblock
- 將 OAM 中的 attribute 設好

但要注意的是，設定 OAM 中 attribute 的動作必須在 VBlank 中才能做，否則畫面會產生 tearing。但無可避免的，我們可能是在 VBlank 中呼叫自己的函式陸續改變 sprite attributes，所以好方法是我們在 VBlank 中的程式邏輯都是針對一個 OAM buffer 進行存取，而每個 VBlank 一開頭的第一件事則是把 OAM buffer 抄去真正的 OAM。這個 timing 雖然單獨看起來很怪 (這次 VBlank 抄寫的是上次 VBlank 改變的東西) 但因為 VBlank 是每六十分之一秒就發生一次，人眼是無法辨別的。

HAM 正是替我們包裝了以上繁複的 memory 存取工作，並且也同時處理了 memory allocate 的問題。

劇情大綱

一名敬業且深受學生愛戴的老師—Mr.cyy 如往常一般的到台大上班。今天是學生們

的 final project Demo 日，他十分期待能看到一些有趣的作品。

沒想到 Mr.cyy 一踏入椰林，便受到椰子猛烈的攻擊，他僥倖地逃過後走入系館，在系館迎接他的卻是一堆的香蕉人。

怎麼了!究竟是怎麼了?! 為什麼美好的一天變得如此詭異?

選單介紹

GOGO CYY 完整劇情模式

Stage1：椰林關卡

Stage2：系館關卡

Stage3：魔王關卡

STAGE1

BG：

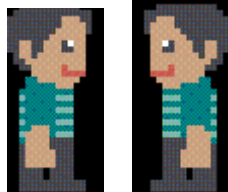
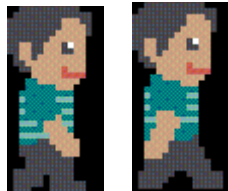
背景：系館

Size：512*512

當 cyy 進入砸椰子遊戲後，背景定格。直到遊戲結束後才能捲動


人物：



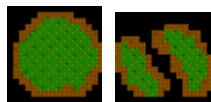
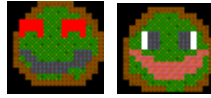
人物有以下功能並根據以下功能更換圖片

功能	啓動方式	備註&PIC
Stand 	什麼按鍵都未觸碰	以之前狀態的方位決定方向
Right walk 	按右鍵	
left walk	按左鍵	

		
Right run 	按右鍵+按下鍵	
left run 	按左鍵+按下鍵	
Right jump 	按右鍵+按上鍵	
left jump 	按左鍵+按上鍵	
Jump 	按上鍵	以之前的狀態決定方位

Sprite :

名稱	功能	出現模式	消失模式
愛心 	顯示 cyy 生命指數	一開始即設定	被攻擊

冰淇淋 	若愛心未滿格能補血	隨機出現	When cyy gets it
熱狗麵包 	若愛心未滿格能補血	隨機出現	When cyy gets it
椰子 	攻擊 cyy	進入遊戲模式後隨機出現	(1)離開遊戲模式 (2)當 cyy 踩他 (3)當 cyy 被它攻擊後 (4)掉到地上
邪惡椰子魔 	攻擊 cyy	進入遊戲模式後隨機出現	(1)離開遊戲模式 (2)當 cyy 踩他 (3)當 cyy 被它攻擊後 (4)掉到地上

STAGE2

BG :

系館

Size : 512*512

主要是將一 512*512 的圖分成三個樓層，只有到樓梯口觸發某些鍵才能上樓

Sprite :

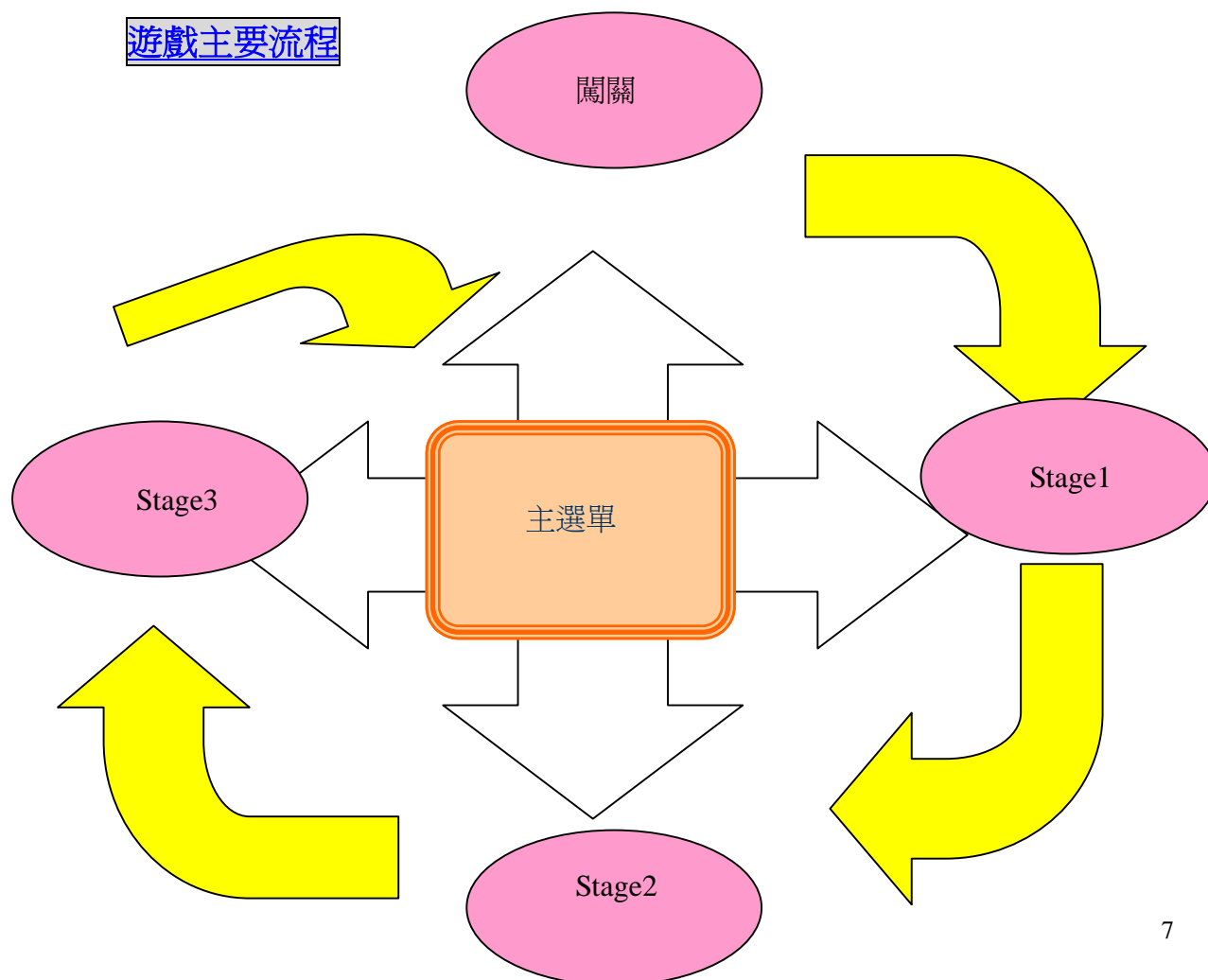
名稱	功能	出現模式	消失模式
香蕉先生 	攻擊 cyy	已固定	被 cyy 踩到
大根 	使 cyy 無所畏懼		when cyy gets it

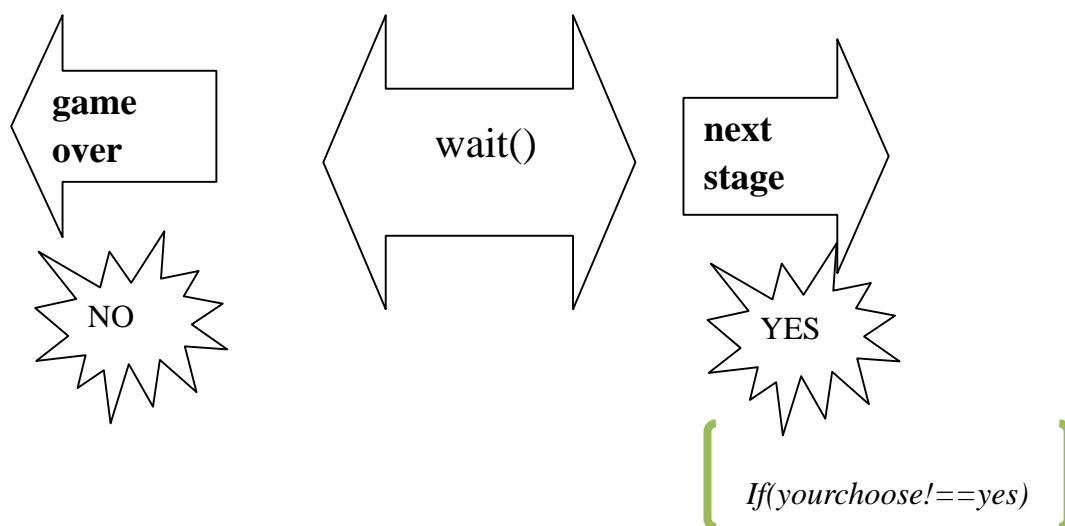
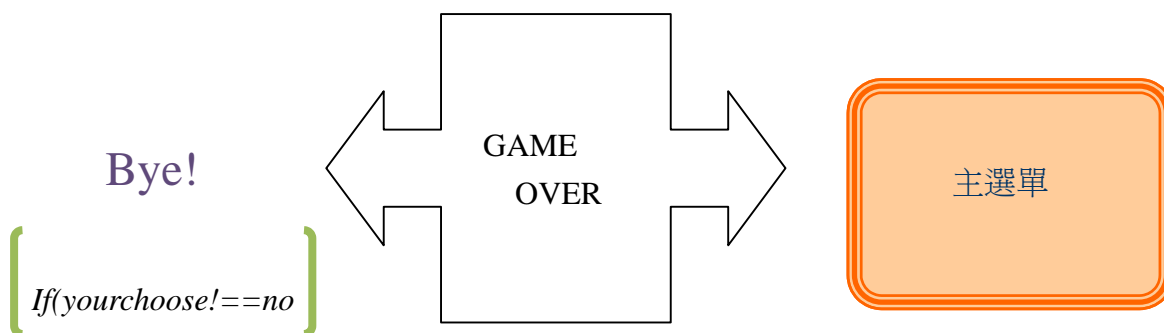
STAGE3

Sprite :

名稱	功能	出現模式	消失模式
紫愛心 	顯示 KTW 生命指數	一開始即有設定	被攻擊
香蕉先生 	攻擊 cyy	隨機出現	被 cyy 踩到
KTW 	被 cyy 攻擊	分身術隨機出現	被 cyy 踩到 4 次

遊戲主要流程





要結構可以分以下六大部分：

主選單：遊戲一開始會進入主選單，有四種模式可以選擇

分別是闖關、第一關、第二關、第三關

闖關：(GoGoCyy) 從第一關開始挑戰

第一關：(椰林大道)

1. 完成關卡會讓玩家選擇是否繼續，是 — 切換到第二關模式，否 — 則切換到 over
2. 如果死掉的話 切換到 over

第二關：(系館)

1. 完成關卡後會讓玩家選擇是否繼續，是 — 切換到第三關模式，否 — 則切換到 over
2. 如果死掉的話 切換到 over

第三關：魔王關

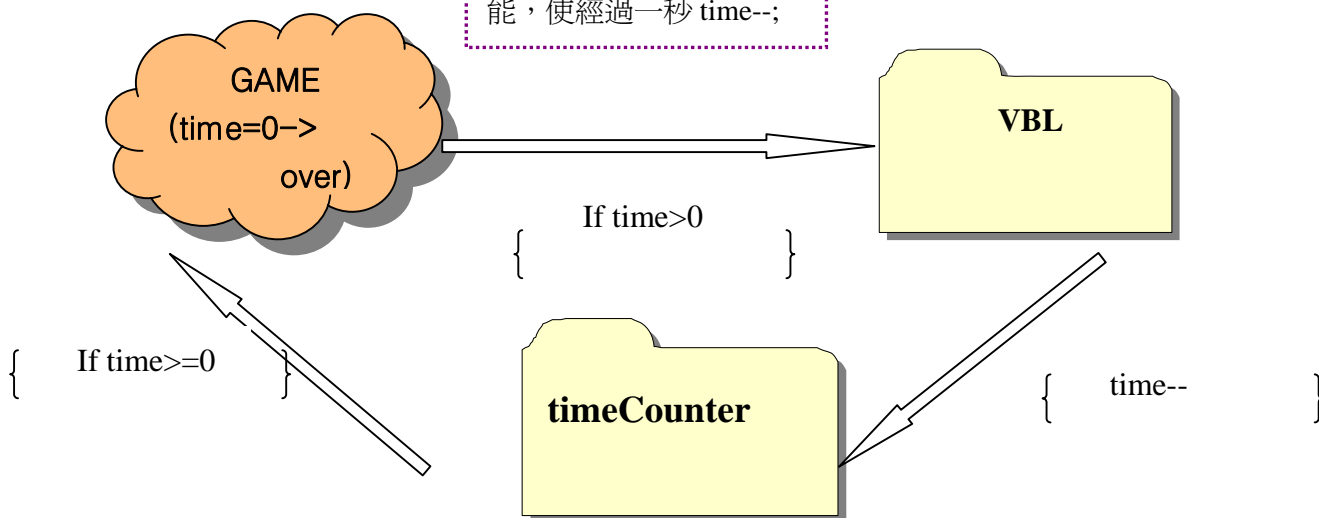
完成關卡或死掉會切換到 over

OVER： 會讓玩家選擇是否再玩一次，是 — 切換到主選單，否 — 則再會

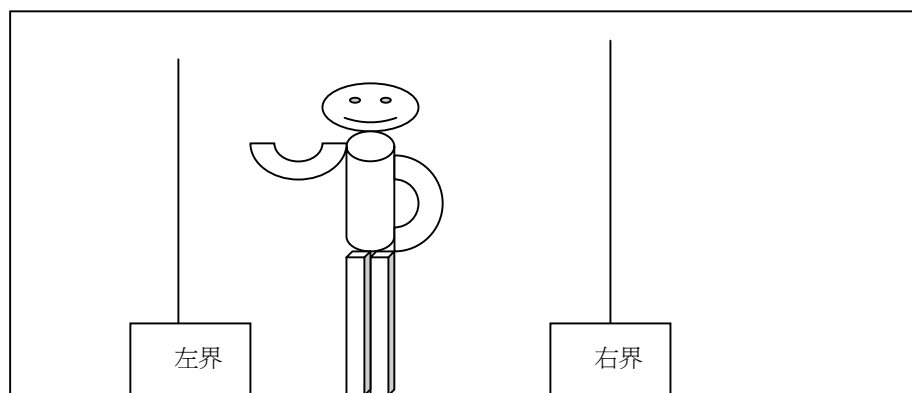
如何去實作一些 function

倒數計時：

Game 啓動 應用 vbl 每 1/60 秒被呼叫一次的空能，使經過一秒 time--;



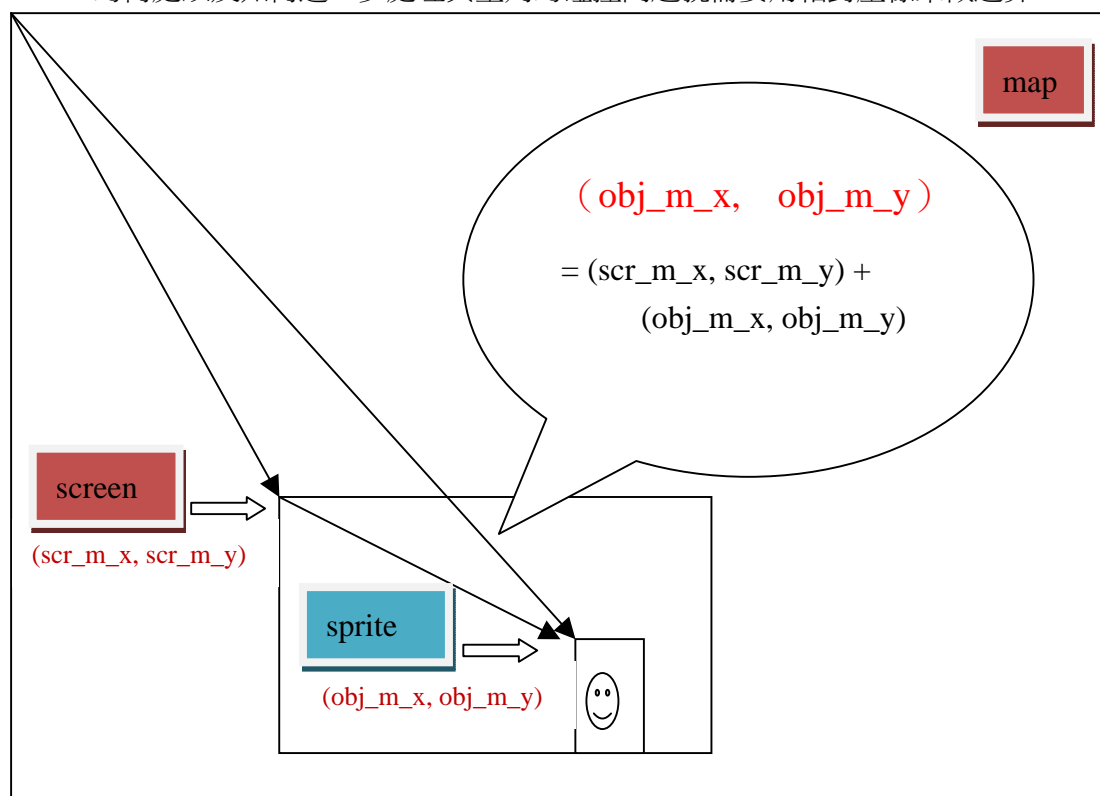
如何讓人看起來像一直在螢幕中間行走：



向右移動:
 If(人的 x 座標 < 右界) 由人去移動
 Else 由 screen 去移動
 向左移動:
 If(人的 x 座標 > 左界) 由人去移動
 Else 由 screen 去移動

動作遊戲中畫面捲動與碰撞等

首先必須說明 GBA 的硬體中能讓我們掌控的主要是 screen 左上角相對 map 的絕對座標 (scr_m_x, scr_m_y) 以及每個 sprite 左上角相對 screen 的相對座標 (obj_m_x, obj_m_y) 但其實 sprite 相對於 map 的絕對座標 (obj_m_x, obj_m_y) 也是需要掌控的。我們必須由絕對座標來知道 sprite 何時 visible 才合理；至於他出現在 screen 的何處以及如何進一步處理與主角的碰撞問題就需要用相對座標來做運算。



簡而言之，整個座標系統的計算流程是：

- (1) 由 o_m_x, y 知道 sprite 該不該出現在 screen 上 【calScrVis()】
- (2) 若出現，就會再算出他在 screen 的相對位置 o_s_x, y 【calScrPos()】
- (3) 之後所有 visible 的 sprite 的碰撞計算都依賴(o_s_x, o_s_y)算出來 【allHit()】
- (4) 有了碰撞情形，再依 sprite 的屬性 (treasure, enemy or land) 得知碰撞結果 【detectXXX()】
- (5) 不只 sprite 的 o_s_x, y 要 update，若他是會移動的話 o_m_x, y 也要更新 【stage2PlayUpdatePos()】

倘若更簡略的說，座標間的依賴關係如下（箭頭右方的值須依賴左方計算出來）

o_m_x → o_s_x → new o_s_x
 → new o_m_x

Stage2 就是此功能完全展現的地方，可以以 stage2PlayVbl() 來說明會最為清楚

```
void stage2PlayVbl()
{
    ham_CopyObjToOAM(); //將 OAM 中的 sprite 更新
    stage2PlayQueryButton(); //偵測玩家們的 key input
    calScrVis(); //以(scr_m_x, scr_m_y) 輻射 GBA 的 screen 大小，
                //以看佈關於 map 上的 sprite 是否該出現
    calScrPos(); // if 主角的目前位置看得到該 sprite->?計算其於螢幕的座標
    calBana(); // 計算 enemy sprite(bana)該如何移動來攻擊主角
    jump(); // 計算主角跳躍的垂直位移與水平位移
    allHit(); //以(obj_s_x,obj_s_y) 計算畫面上所有出現的 sprite 和主角的碰撞關係
    detectTrea(); //由 allHit() 計算的碰撞結果 來更新 treasure 與主角的狀態
    detectEnemy(); //由 allHit() 計算的碰撞結果 來更新 enemy 與主角的狀態
    detectLand(); //由 allHit() 計算的碰撞結果 來更新 land 與主角的狀態
    stage2PlayUpdatePos(); //由以上的計算結果 來更新所有 sprite 在 screen 上的位置以及它
                            //們在 map 上的位置
    stage2PlayUpdateAnim(); //由以上的計算結果 來更新所有 sprite 在 screen 上的動畫
    return ;
}
```

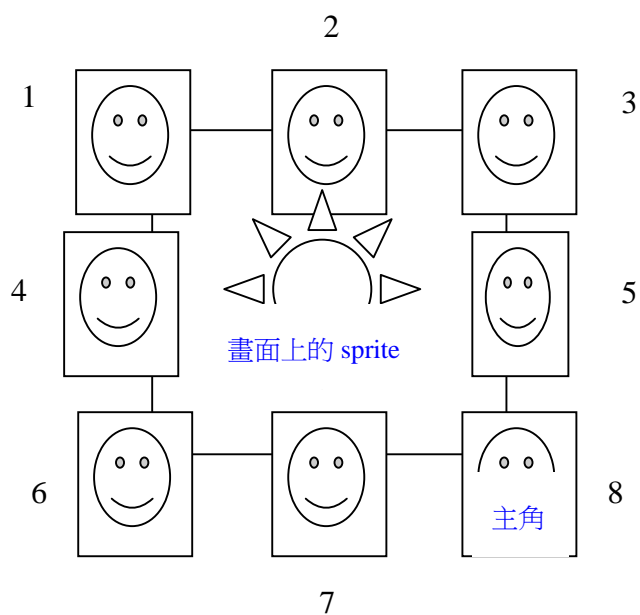
jump()

Jump 不似其他功能可以輕易的一次在 vbl 中就完成，必須給他一段時間逐次讓主角跳起並且落下，因此我們在主角的 struct 中定義了 jump_flag（是否在跳躍狀態）jump_phase（在上升還是下降）jump_count（垂直位移量）hori_dir（水平位移量）

```
jump(){
    if( 主角在跳躍){
        if(主角在上升狀態){
            垂直位移量每次減少（模擬重力加速度）
            水平位移量依跳躍方式往不同方向增加
        }
        If(撞到頂端) 垂直位移量歸零進入下降狀態
        If(垂直位移量被歸零) 進入下降狀態

    }else if(主角在下降狀態){
        垂直位移量每次增加（模擬重力加速度）
        水平位移量依跳躍方式往不同方向增加
        If(撞到可落腳處) 跳躍 flag 歸零
    }
}
```

Hit()



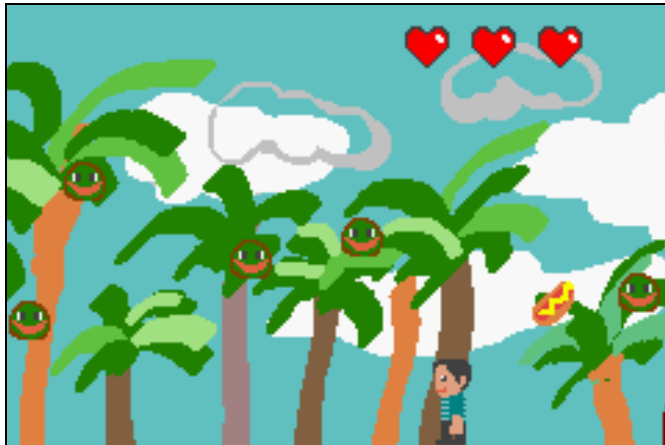
Hit() 會計算畫面中主角與其他 sprite 的碰撞情形，如上圖編號 1-8 種狀況，並將碰撞結果記錄在 sprite 的 struct 中。detectXXX() 會去檢查畫面上每個 sprite 的 hit_sta，再依 sprite 屬性做出合理的判斷。(ex. treasure 從八個方向都是吃掉、敵人只有從上方才能踩死、land 要更詳細的利用八種 hit_sta 來計算是否順利站上去或是滑下來)

遊戲截圖

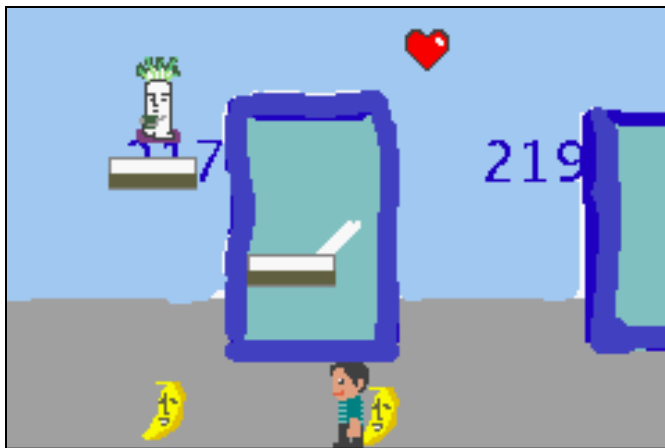
主選單



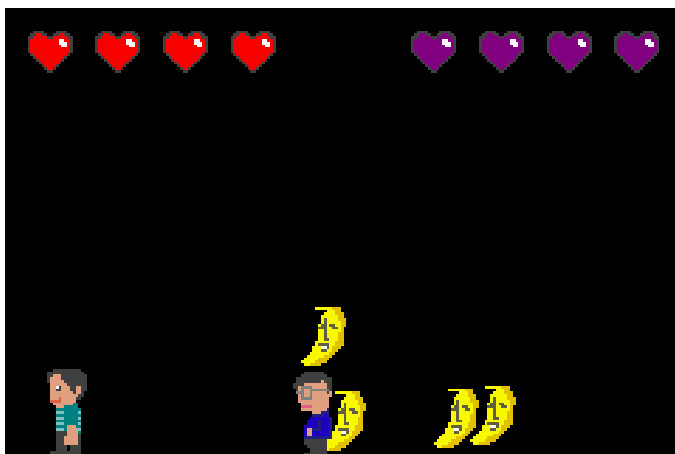
Stage1



Stage2



Stage3



維青的話

整個 project 是集中在期末考後的五天半進行 coding，但我們早在十二月中開始就花不少時間一起討論、看 document、嘗試 tutorial 等等（火星人助教可以作證 XD）project 穿插在作業考試間的這段日子真的是很操，但進了資訊系一年多總算有一次團體 coding project 的經驗，其實也挺有趣的。雖說幾千行實在也算不上大程式（而且因為沒有 OO 可以用以致有不少重複的 code XD）卻也是我們的初體驗呀：第一次深切體會討論 spec 的重要、為什麼要好好封裝 code 以提高修改與再利用的彈性...這些上學期 OO 課言猶在耳的種種突然非常清晰.= 我想我們的 code 中應該看得出希望 apply 所學的企圖，但難免有些經驗不足的錯誤（還有些爲了趕工或快速解決 bug 而失去一般性的邏輯/\），無論如何，有這樣的實作經驗都讓我對自己身爲 programmer 的這個角色有多一點覺悟，這是非常需要經驗的，否則空有理論雛形卻不能精、確、速地 implement 自己的想法實在是十分窘迫@@

除了個人的學習體認，小組合作也是很棒的經驗。我們四人平常都算不上是那種驚動萬教的強者，有的就是一起一步步的討論、協調與努力。與十二月還在爲 project 徬徨的四人相比，現在總算能帶點成就感的說：我們已非昔日的吳下阿蒙囉（至少進步了一點點啦：P）

喬尹的話

是進大學以來的第二次小組性 project（第一次是大一下的 UML），與 UML 不一樣的體會是，以前寫程式都是自己在電腦前埋頭苦幹，寫寫作業、寫寫 ACM，一份 code 不過兩三百行，雖然知道一些 coding 的不成文規定或是小技巧（如 #define 常數、MACRO、註解或是 head 檔之類），但卻鮮少想去使用，往往都是認爲「我很了解這變數或 function 的作用，沒有必要去多寫浪費時間。」然而，這樣的寫 code 態度在這次的 project 裡大大地被改變了。

在團體 project 中，一個人所寫的幾百行 code 不過是其中的一小部份，組合起來才是一個完整的程式。於是某個常數不 define、某個 function 名稱過度簡略、變數名稱重複或沒有規律，這樣的「壞習慣」，最後造成整組得多作一些無謂的微調。小組成員們在一開始雖然對於要作的成品有個共同概念，但不表示每個人的思考取名方式都相同，所以除了一開始協調完整之外，單人的 code 好習慣應該也是應該從一個資訊系學生開始培養起的。

另外對於 OO 的優點也有所體會。因爲我的工作也是負責 SPRITE，在設計各關角色時，相同的角色大部份的 attribute 都會相同，只有少部份針對當關所加的 attribute，這在 OO 中利用繼承相當簡單就能處理。但在 PROCEDURE ORIENTED 中，只要有一點不同，就得重新規劃一次，這對於 maintain code 來說是一個大麻煩，於此可了解到爲何現

在汎用的應用程式，越來越多是由 JAVA 組成。

上了大學，各科的小組作業都會為我帶來各式不同的體會。這次特別的是它是個屬於我專門領域的一次小組實作作業，對於這樣第一次的經驗，腦中對於未來工作的形象越清楚了那麼一點

冠汝的話

從開始準備到結束的這段時間，大概也快要有一二個月的時間了。從開始決定主題、規劃讀書計畫、討論、實作...我們轉換了跑道不少次，也改變了自己唸書的 tempo 和討論的方式，從各唸各的到每個人負責不同段落去講解給別人聽。每一次的轉變，或多或少都得到了些東西。像是我們原本要用 tonc 去發展一些遊戲，不過因為一些考量，還是決定用 HAM 寫。雖然看似走了很多冤枉路，但因此我們對 HAM 的底層運作原理多了一些認識。

那時對於寫出一個遊戲還是有一定程度的懼怕，心裡想的就是“這麼短的時間之內要完成真的很困難!”眼前又有每個禮拜一定份量的作業和即將來到的期末考，雖然想到頭腦就發昏也只能硬著頭皮去作。

不過由於我們組算是較早開始討論的，也有固定的討論時間逼著我們前進，所以一路上雖稱不上從容不迫但也不至於慌了手腳，大方向的進度都能如期的掌握。最後的六天是最忙碌且收穫最多的，一天幾乎都坐在同一個位子沒離開，有時候為了一個 bug 快氣死，不過看到成果真的很開心，這種從無到有的感覺真的很棒，大家一起為了一件事而努力、而奮鬥的成果真的很刻骨銘心!總而言之真是記憶深刻阿~~~

也很感謝組員和老師以及助教，你們的每一句話真的都是醍醐灌頂!

甯之的話

這次做作業其實從很早就開始了，才覺得以前學到的東西都很有用，資工系的訓練果然還是有差的。這次從無到有的過程：

了解原理 -> 環境的摸熟 -> 會用函式 -> 做出小功能
-> 背景、OBJ、會跑會跳、相對位置、絕對位置、相互搭配計算
-> 有畫面 -> 加入聲音 -> 一個大關卡 -> Link -> 完整的遊戲

印象最深的兩個:

第一個: 加聲音聲音其實很複雜，但經由了解一些原理，在搭配上 HAM 的函式，感覺才用比較順手。這之間非常感謝助教的協助，我先是想用 Krawall，但是它(Krawall)現在說明不清楚(據說是因為商業化不能完全公開)，聲音檔要用什麼、拿什麼，用什麼轉上面都有點模糊，周邊搭配很難掌握，真的頭昏腦花。我的各種轉檔程式也沒

有附說明，甚至我對音樂的各種格式檔本來就不認識，也不知道這是幹什麼。經由助教的解說才了解聲波由類比轉數據的方式，和其用到的想法及原理，助教還教導我怎麼轉檔跟利用檔案，這一些若沒有助教的協助，在聲音檔上面大概真的搞不定了><

第二個: 各關卡的 Link

也是我這次最頭痛的 link，因為之前 C 這部份搞的不是很懂，原理整個亂掉，發生了起碼有上百 ERROR，看的我慌目驚心，在拼命的 Debug 和砍掉重練後，我就直接衝去找老師了。

老師講的超好，才發現我以前根本就只依據感覺和測試建立起來，我回去再測，又一堆 ERROR，但已經有改善了，我也比較懂他的錯誤訊息是在告訴我什麼。但接下來的 ERROR 就真的一測再測，測到想要吐了，天外來個一筆:系上各種訓練還真的有用，訓練我們抗壓能力--錯了還是要繼續做….

接下來就是大感動了，在又用了很久之後，我找到了助教(那時候我已經上不了五樓 正在著急)他輕易幫我找出錯誤和告訴我為什麼，於是我回宿舍後弄到很晚，總算 link 主選單和第一關，但連結第二關又出現變數衝突問題，解決了衝突又冒出了抓不到的問題，真的是一個頭兩個大，寫到這裡，其實還未做完，後續就看看 Demo 到底有什麼吧~

喔耶 DEMO 成功。但是，我很努力弄懂觀念，但發現 DEV 上行的通的東西，在 HAM 上卻行不通。戒急用忍下(時間來不及><)，先用另外的方式解決，爲了這件事，事後問了老師和助教，再查了資料，讓我又對 LINK 和電腦的底層工作又有了進一步的認識，應該是因為不同組語的定義上不一樣產生的問題，一層一層息息相關，大概了解了一些他們的關係真的學到很多，雖然過程中受挫很痛苦，但是學到東西真的很充實，我真的以後各種東西都要先弄清楚原理才行。

感謝老師助教對我問很基本的問題都認真回答，也謝謝同學們的協助和幫忙，這次 project 真的收穫很多(如果不計較過程有點讓眼睛和身體快壞掉@@)，組語的課也畫下圓滿句點，謝謝大家!

分工狀況

BG+SOUND+LINK：甯之、冠汝

SPRITE+STAGE：喬尹、維青

感謝的人

B95902091 曾怡茹同學 感謝他幫我們畫人物與背景 讓 PROJECT 增色不少！

助教

老師

還有很多在我們遇到瓶頸時解救我們的人

期末考後六天在系館一起打拼的同學們