Johnny Go !?

Computer Organization and Assembly Languages Final Project

Authors:

B95902018 邱珮甄

B95902014 陳奕安

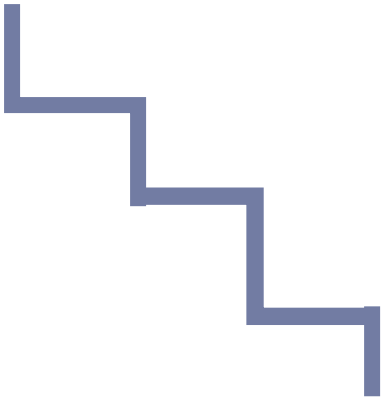
B95902075 林士涵

B95902121 黃詠筑

NTU_CSIE Fall, 2007

Johnny Go !?

小囧下樓梯



前言

期末考終於結束了，但是還有最後一科作業我們才正式告別大二上——組語 project。

但是對於原本就不是程式高手的我們這一組的成員們來說，要用組語寫出一份像樣的報告其實是頗具難度的，但是身為一群台大資工的學生，如果連這一步都克服不了，以後也沒什麼好談的了，我們如是想，也就更不願意草草了事圖個輕鬆，於是，我們選擇了學期尾聲教的 intel x86，決定全程都用組語寫出一個”小朋友下樓梯”的小遊戲，希望能藉由這次期末報告讓我們本身的程式能力獲得進步。

選用 intel 的原因在於我們本身並不是很有經驗的 programmer，而對於組語的經驗更是少之又少，大概就僅限於平常的作業範圍吧！平常寫一些 ACM 或一些小程式之類的也不會想動用到組語，所以為了能夠更加熟悉組合語言，且為了儘快上手，我們選擇了有有經驗的學長（再次感謝 B94 楊逸民學長提供寶貴的經驗）可以請教的 intel 當作我們的語言。

而會選小朋友下樓梯這款遊戲，最主要是考慮到這個遊戲的組成元素較為單純（兩個，小朋友和樓梯），有助於我們四人分工上的方便，而且具有動態性，不是益智性的靜態遊戲，遊戲性較高，因此我們選擇了這款小遊戲。

而經過了一個禮拜幾乎不眠不休的討論+coding+debug，歷經兩千多行的 code，程式功力薄弱的我們也終於寫出了一些成果，以下請看我們盡己所能做出來的”小囧下樓梯”！

目錄

小囧的出現

—— 一切都從「點」開始

實作：邱珮甄
Report：邱珮甄

原先在決定要做「小朋友下樓梯」時，大家就有在討論我們的主角「小朋友」是要以什麼樣的形態出現，還在天馬行空的期間，大家都十分憧憬做出很千變萬化的小朋友下樓梯，紛紛避重就輕的討論我們要設什麼樣的關卡、有什麼樣的死亡方式…等等，討論這些愉快的 topic 時間總是一下就過去了。直到有經驗的學長在一旁給了我們當頭棒喝：「你們現在拼命想這些夢幻的關卡，還不如先來思考要如何使一團點自動的往下移動！」當時大家才猛然發覺，最基本但最必要的其實早在眼前，只是我們都刻意忽略了！於是，這便成了我們著手 project 的起始點…。

研究方式與遇到的困難：

1. 一點自動直直向下掉

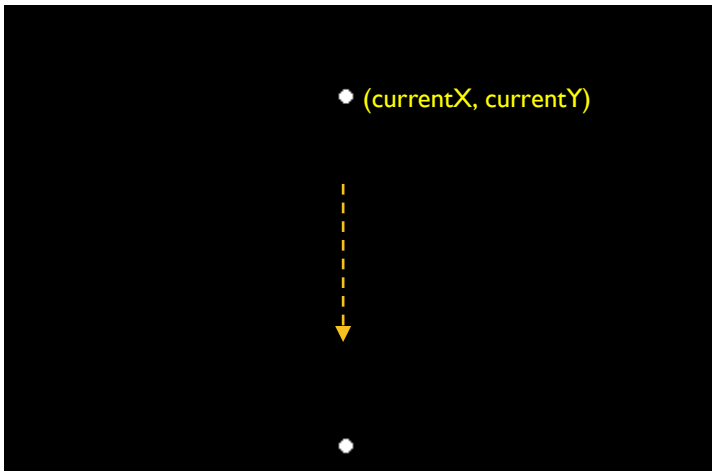
小朋友下樓梯和一般遊戲不太一樣的地方是，它並不是單純接收按鍵才會移動，在沒按鍵且不在樓梯上的時候，是會在空中直直墜落的。因此，我們打算以先完成「一個點能夠直直墜落」為目標，再進行下一步驟。

首先，要能夠呈現一個點，我們使用到 int 10h 中的 0Ch (Write Graphics Pixel) 來幫我們實現。範例 code 如下：

```
mov     ah, 0Ch                 ; write graphics pixel
mov     al, COLOR_ball          ; 要畫的點顏色
mov     bh, 0                   ; video page 0
mov     cx, currentX            ; 要畫的點 X 座標
mov     dx, currentY            ; 要畫的點 Y 座標
int     10h
```

利用了 int 10h 中的 0Ch，便可在螢幕上呈現想要畫的圖案。(Figure 1)
成功的話出一個點之後，接下來就必須思考該如何使它直直下墜。我們用的方式是先將上一個點擦去(意即以背景色圖滿)，再於下一個我們希望的地方重

畫一個點，重畫的方式如同以上，以目前只是要直直下墜的情況為例，只要 currentY 遞增，便可達成。如此以 loop 的方式不斷將畫面刷新，直到點已超過邊界。判斷點是否超過邊界的方式是以 conditional jump 的方式，當 currentY 已達 200(即邊界值)，便可 exit。

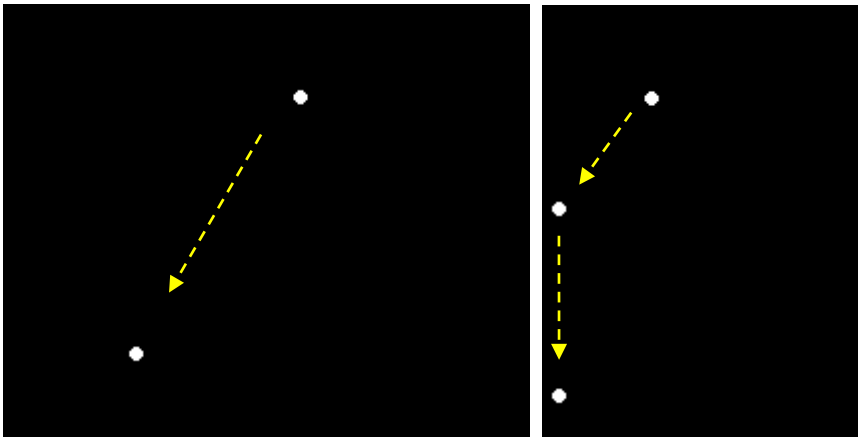


2. 一點可以在空中左右移動

小朋友下樓梯這個遊戲特別的地方就在於，因為小朋友大部分的時間是不在樓梯上的，在空中時便會自然往下墜，此時如果玩家按左鍵或是右鍵，便應該形成斜下的情形，此時便需接收 keyboard 的訊息(詳細情形請參照 Keyboard 部分)，當玩家按左鍵時，點便會往左下移動；當玩家按右鍵時，點便會往右下移動。但此時會有一些特別情況發生，也就是當點現在位置位於左右邊界時，點無法往左(右)下，只能夠往下移動。因此必須要有一個 conditional jump，去判斷此時的 X 座標是否於左右邊界。範例 code 如下：

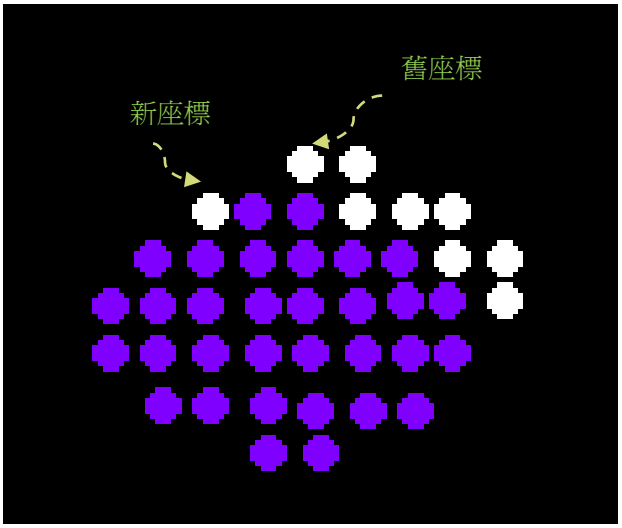
```
mov     ah, 0Ch                 ; write graphics pixel
mov     al, COLOR_ball
mov     bh, 0
mov     cx, currentX
cmp     cx, 0                   ; 判斷此時的 X 座標是否在左邊界
je      NoKeyWaiting            ; 如果已在左邊界便 jump 離開
sub     cx, 1                   ; 往左一格
mov     currentX, cx            ; 移動之後重設 X 座標
mov     dx, currentY
inc     dx                      ; 往下一格
int     10h
```

如此，便可讓點在空中左右移動(Figure 2)。



3. 「囧」可以在空中左右移動

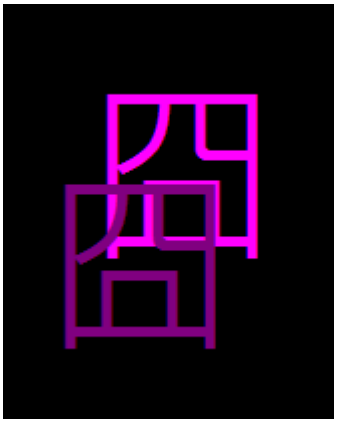
成功的讓一個點在空中左右移動之後，接下來的任務便是要讓一「團」點能夠在空中左右移動。這會是一個相當麻煩的作業，因為要想辦法讓這團點中的每一個點都能夠成功移到正確的位置。一開始我們想過許多種方法，一是將整個畫面刷新，也就是重畫每一個 pixel，但這樣可能會遇到的困難是速度會過慢，而我們要移動的只是一團相同顏色且知道大小的點而已，因此後來決定不刷新整個畫面，而是只重畫整團點。當我們還沒決定要用什麼圖案時，是以「球」的造型登場！利用一個 procedure 專門製造球的功能，先將現在位置的球擦去，再利用 invoke 傳入新座標給製造球的 procedure，最後於新座標畫出一個移動過後的球(Figure 3)。



但由於由點構成的球實在不是很好看，因此我們決定不以球做為我們的「小朋友」，在幾番討論之下，我們決定以 Johnny—囧尼，做為我們的主角！因此，移動的就會是一個「囧」字，我們使用的方法與球類似，用一個 procedure 專門製造「囧」，之後在我們遊戲的過程中不斷的刷新畫面，擦去原先的「囧」，新增新位置的「囧」。若是判斷到使用者按左(右)鍵便更新一個新的座標，並以新座標 invoke「囧」的 procedure，若沒有接收到按鍵便直直下墜。範例 code 如下：

```
;-----------------------------------
Jone PROC,
        X: WORD,
        Y: WORD
; Jone drawing
;-----------------------------------
…
```

```
mov     cx, currentX
cmp     cx, 0
je      NoKeyWaiting
dec     currentX
inc     currentY
invoke  Jone, currentX, currentY
; 設了新座標之後 invoke Jone 的
procedure 去製作新座標的「囧」
```



但在完成之後我們發現，左右移動時或許是因為要判斷按鍵，因此視覺上讓人覺得移動較慢，為避免這個情形，我們採取的策略是，一次向左(右)移動兩格，向下仍然維持移動一格，如此便可達成左右與向下移動不會有太大的速差。

完成「囧」的移動只是我們 Final Project 的開胃菜，接下來與樓梯的合併才會是真正困難的「主菜」，但是完成此項任務對我們來說，已經是一針強大的強心劑了！

樓梯的產生

實作：陳奕安
Report：陳奕安

研究動機：

這一時期我們還沒有任何樓梯的雛形，所以第一階段要做的是先畫出一個不會動的樓梯在螢幕上，第二階段就是讓它動起來，而第三階段就是同時讓很多個樓梯隨機出現並且動起來。

研究過程：

第一階段：

這一部分問題比較少，我們的方法是畫三條水平線疊在一起，這個部份用到了 int 10h 中的 0Ch，也就是”對一個 pixel”著色，重複這個動作並在恰當的時機換行，即可達成目的，而這裡最重要的就是 ASM 迴圈的寫法。

這裡值得注意的是要隨時注意暫存器裡存的究竟是什麼東西，因為暫存器 cx 隨時都要當 counter 而同時也是 call 0Ch 時儲存 x 軸的位置，所以一定要很注意 stack 中的情況，我們整個製作的過程常常發生 stack 中的東西忘記 pop 而使資料錯亂的情形。

第二階段：

我們讓樓梯動起來的方式是不斷的把樓梯擦掉再重畫在上面一格的位置，但是如果這樣做的話會讓整體速度下降，所以我們改成只擦掉樓梯最下面一層，然後在樓梯上面一層畫一層新的樓梯，這樣就得到”往上一格”的效果，而且要移動的部份也變少了，簡單來說，就是比對移動前後差異的部份，然後去改變那些差異的部分即可，速度有因此加快了不少。

第三階段：

這一部份相當困難，因為組語中暫存器的數目有限，但是如果一直用 memory 宣告變數在執行上又會非常慢，這樣就無法讓每個樓梯的位置都記錄下來，如果不紀錄而是去地毯式的讀取螢幕上的每一個點的話，則會發生運算過久的情形，所以我們一開始不知道的時候，曾經做出一個幾乎是一格一格動的樓梯群，所以我們發現爆搜絕對不是一個可行的途徑，只好另尋他法……

解決方法：

最後我決定從改良演算法方面著手，最後想出來的方法是把所有樓梯彼此建立關係，也就是固定住樓梯與樓梯之間”Y 座標”的巨哩，如此一來我只要記錄第一個樓梯的位置，即可加上樓梯的間距進而找到下一個樓梯，但是 X 座標的話由於是 random 產生位置的，所以無法記錄其間距（且若連 X 座標都規定固定間距的話，遊戲就不具遊戲性了），所以就只好從那一行的頭（X＝0）的地方開始搜，但是至少已經比原本的每個點爆搜快上許多了。

演算法比較：

舊演算法(爆搜)：每次要搜尋 320*200 個 pixel

新演算法(連動)：只需搜尋 320*7 個 pixel，因為同時最多 6～7 個樓梯。

結論：這一部份演算法幫了大忙！

指尖的動與靜—Keyboard

實作：邱珮甄、黃詠筑
Report：邱珮甄、黃詠筑

KEYBOARD 的部分，起初大家準備的方向與重點都放在課本第 15 章作為參考基準，其中會用到最多的大概是 10h 和 11h。在我們的遊戲中，KEYBOARD 的需求並不複雜，大概要用到的就是左右鍵、SPACE(暫停)、ESC(離開)以及任意鍵繼續，判斷的部分應該都差不多。比較插曲的部分大概是因為，先前有被提議及考量到不知道 15 章部分的 FUNCTION 支援，會不會使遊戲呈現一種 LAG 的畫面，就是有那種一格一格移動不順暢的感覺，所以便考量要用 16 章提供的方法去著手，做出類似 KEYBOARD PORT 的東西，可能對遊戲操作流暢度能提高。於是我稍看了一下 16 章;；然而很意外的是，我們設計的遊戲對 15 章提供的 FUNCTION 有很良好的適應力，意思是說，先前擔憂的問題似乎並沒有發生，左右鍵可以流暢的配合，檢查 BUFFER、有沒有 WAITING 的輸入、沒有則繼續檢查、有則取出檢察看否符合所需、是則執行。而因為這樣的意外，減低了我們去理解 16 章 FUNCTION 的工作量並能有效運用 15 章，不過看過 16 章還是可以得到不一樣的收穫！

在 int 16h 中的 11h(Check Keyboard Buffer)，return 的值為：If a key is waiting, ZF = 0, ah = scan code, al = ASCII; otherwise, ZF = 1. 我們將想判斷的鍵的 scan code 查出之後，與 ah 比對，便可得知玩家是否按下此鍵。

至於實作上，由於我們需要判斷的不多，因此並無特將此建立一個 procedure，只在遊戲的主要 procedure 中進行判斷，以下是範例 code：

```
continueN:                          ; 判斷功能鍵
        mov     ah, 11h
        int     16h
        cmp     ah, 1B              ; 判斷 scan code 是否為 esc
        je      Stop                ; 若相等則停止遊戲
        in      al, 60h
        cmp     al, 39h             ; 判斷 ASCII 是否為 SPACE
        jne     Go                  ; 若非則遊戲繼續進行
        mov     ah, 10h             ; 否則暫停遊戲直到有任意鍵產生
        int     16h
        jmp     continueN
```

```
KP_left:                                ; 判斷左鍵
        mov     ah, 11h
        int     16h
        jz      NoKeyWaiting
        cmp     ah, 4Bh                 ; 判斷 scan code 是否為 4Bh(左鍵)
        jne     KP_right                ; 若非則去判斷右鍵
```

小囧下樓梯—初階版

實作：邱珮甄、陳奕安、林士涵
Report：林士涵

研究動機

對於一個小朋友下樓梯而言，個別做出了樓梯和小朋友當然不夠，重點是要讓他們能夠連結起來，因此如何將小朋友和樓梯結合起來就成了我們必須研究的部分。

研究過程

首先我們就將囧(小朋友)的移動相對於樓梯限定了範圍，使他不至於走到樓梯內，這個部份就是給定許多條件讓囧判斷，首先我們先判斷囧是否在樓梯上，方法就是讀囧的最下面那排 PIXLE 的下面那排 PIXLE 有沒有樓梯的值，如此就可以把情況分成樓梯上或不是樓梯兩種，首先是囧不在樓梯上時，由於囧在移動時，畫面刷新一次左右方向的移動都是動兩格，而上下移動則是一格，所以我們必須判斷左右鍵是否有效，方法就是判斷現階段按左或按右後囧要移到的地方有沒有樓梯(也就是是否為黑色)

EX

＼＼＼●→囧的左下角

＼？＼＼判斷問號的位子是否為樓梯來決定可否向左下角移動

＼＼＼＼

而向下移動的情形由於只會向下一格所以若囧不在樓梯上即沒有限制。

第二部分則是如果囧判斷的結果他在樓梯上，由於這種情形囧必須隨著樓梯一同上升(每次上升一格）所以反過來判斷左上，右上等。

最後就是一個我們當初思考很久的 BUG，在說明之前我們必須先解釋囧要著陸在樓梯時的情況，根據我們的 CODE，我們會先動完囧在去將所有的樓梯向上一格，這個情形看似沒有問題，但是當囧走完後剛剛好在樓梯上時，由於樓梯不知道所以他只好乖乖

的上升一格，而把囧給吃掉了，因此我們後來思考後，將整個流程改成了先去動囧，囧動完後不急著動樓梯，而是在去檢查現在囧是否剛好在樓梯上方(這個檢查的方法是反過來去檢查囧的下方是否為樓梯，這樣一來可以省去很多的麻煩和時間)，若囧剛好在樓梯的上方，我們就跳過這次樓梯向上的動作，來避免囧被吃掉，又如果囧不在樓梯上就去移動整個樓梯系統，移動完後在去進行下一次的囧的移動，就這樣周而復始。

當初 De 這個 Bug 時我們是這樣分析的，因為執行的結果錯誤都發生在著陸時，所以我們將著陸的動作好好的透析了一番，情況有下面兩種：

1. 囧動完以後離樓梯一格，換樓梯動，樓梯上升兩者梯在一起
2. 囧動之前離樓梯一格，被判定不在樓梯上，需要一個含有向下的移動，顧著陸於樓梯上

而上述的情況就是發生在情況二的時候。

其他的部分大概就是一些小 bug，但是這些小 bug 在 de 起來卻是相當的困難，由於沒有完善的 DeBug 系統，所以我們採用”DELAY 大法”，也就是把 call DELAY 加進某段 CODE 中，來判斷實際執行時，是否有進到這段 CODE 中，以達到 DeBug 的效果。

小囧即將面臨的挑戰—關卡的設計

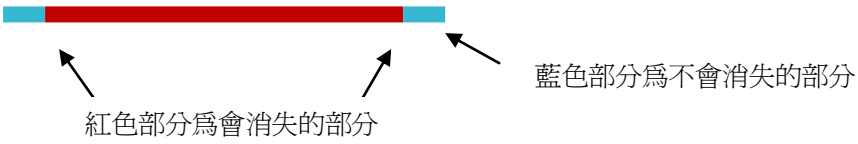
消失的樓梯

實作：邱珮甄
Report：邱珮甄

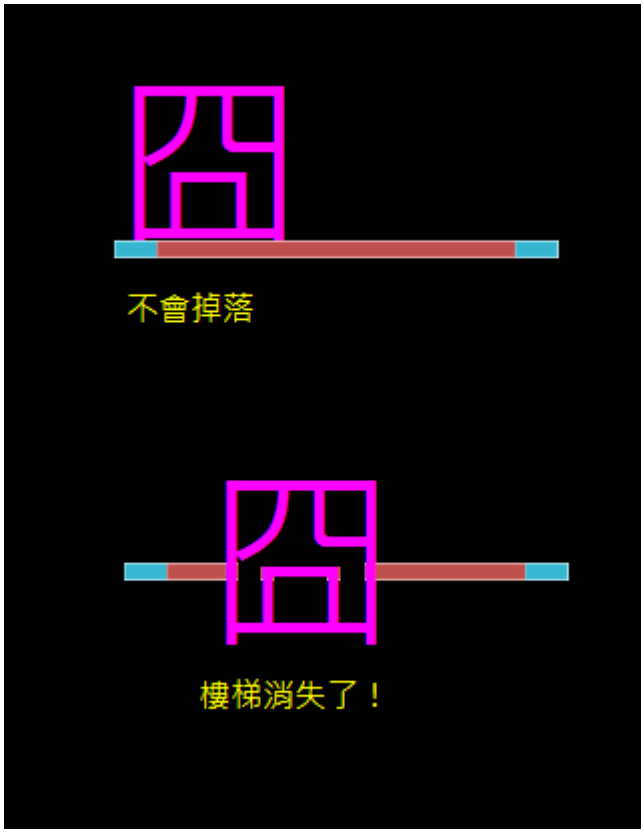
消失的樓梯顧名思義就是會消失，也就是當玩家很開心的把「囧」安心的放到這個樓梯上面時，會發現「囧」居然會穿越過去！這樣子的設計也是十分刺激的，但在製作上仍然是遇到了不少困難…。

當初的想法是在消失的樓梯移動時，判斷「囧」是否在樓梯上方，如果是的話便將整條樓梯塗黑，讓「囧」直直墜落。但由於我們的樓梯生成是需要跟上層的間距的，若將整條樓梯全部刷黑，則下面的樓梯便會無從得知此樓梯的位置而在生成上便有困難。因此我們決定只將「囧」的下方 15 個 pixel 的樓梯消失，之後樓梯上升時便會再畫成新的且完好的一個，如此便可製造「囧」突然穿越但樓梯還在的窘境。

看似解決的問題其實仍未解決，因為我們的樓梯上升是依據最左邊 pixel 的位置生成新的，但若是「囧」位於消失的樓梯上方，且接觸到左角時，消失的部分便會是最左邊的幾個 pixel，在樓梯上升時，查驗最左邊的 pixel 位置便會發生錯誤而導致樓梯向右移動。為避免這樣的情況，我們決定重新設計消失的樓梯，將其變成由兩個固定部分鑲嵌住的樓梯，只有中間部分會消失。



於是在判斷「囧」是否會掉落時，必須確定「整個」「囧」都在中間紅色部分，若有一處接觸到藍色部分，則不會掉落，我們的做法是先判斷「囧」下方最左邊的 pixel 是否為紅色，若否，則可繼續停留；若是，則判斷最右邊的點是否也在紅色部分上，若否，則也可繼續停留。



至於讓樓梯消失的方法，我們利用將紅色部分視為和背景色一樣的地位，讓「囧」可直直墜落，並且一次墜落三格(三個為樓梯寬度)，造成快速驚悚的效果！

帶刺的樓梯

實作：陳奕安
Report：陳奕安

研究動機：

希望做出一款帶有”扣分”特性的樓梯，靈感來自正版的小朋友下樓梯。

研究過程：

總共可分為生產、移動和扣分判定三個部份。

生產：

基本上與普通樓梯的生產相同，但是特別的是它是雙色間隔的樓梯，所以生產的時候要比一般樓梯多一道”重覆上色”手續，也就是我先將樓梯塗成同一種顏色，接下來就在上面每五個 pixel 的地方塗上一點另一種顏色，如此便完成了帶刺樓梯的生產。

移動：

也就是和普通樓梯的移動相同，而畫法則跟上述的生產方式相同。

扣分判定：

這是這個樓梯最重要也最困難的一部分，我們的設計是當囧一碰到這個樓梯就會被扣四分，一但分數被扣完就進入 gameover 畫面，但是當囧停在上面不動的時候卻不能重複扣分（不然一下就死掉了）。

3．解決方法：

為了達到這個目的，我們設計了一個 ScoreFlag 的 data（與記分板用的相同），每次進入迴圈都會判斷囧的下方有沒有樓梯，若無，則 ScoreFlag 設為 0；若有，則判斷是什麼樣的樓梯（與記分板機制類似），而當判定為帶刺樓梯時，則檢查此時 ScoreFlag 是否是 0，若為 0 則分數扣四然後 ScoreFlag 改為 1，這樣如果待在帶刺樓梯上的話，就會因為 ScoreFlag 是 1 而無法進行扣分，而一離開樓梯則 ScoreFlag 會被歸為 0，下次就可進行加減分的動作。在經過測試之後，這個方法的速度跟效果都符合我們的預期。

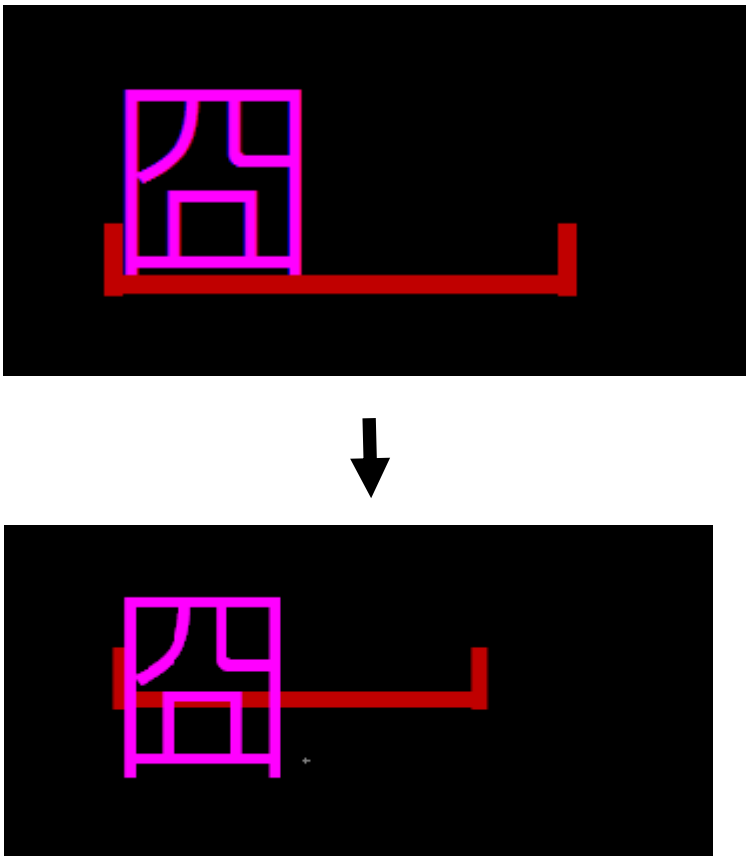
你想出去嗎？ 嘿嘿！不讓你出去！

—— |____| 形樓梯

實作：邱珮甄
Report：邱珮甄

當我們在討論遊戲性時，不免總是想到原版小朋友下樓梯的樓梯樣式，但想像力豐富的我們，當然希望有些創新！於是便創造了屬於我們自己的|____|形樓梯。

|____|形樓梯的意思是，當「囧」掉入|____|形的凹槽時，會被困住而無法逃脫，因此只好眼睜睜的被樓梯帶上去而死亡。看似容易的關卡製作，卻花了我們相當多的時間。由於「囧」對初版樓梯的判斷會因為左(右)有樓梯而直直下墜，因此一開始加入|____|形樓梯時，「囧」遇到凹槽的內側邊界便也會直直下墜，而造成如下圖的情況：



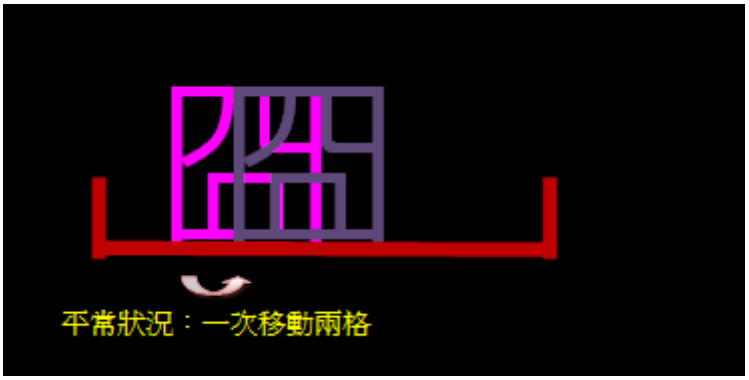
因此我們勢必要多增加條件以避免上述情形發生。我們採取的方式是，先判斷「囧」是否位於樓梯上，如果是的話便跳過判斷左右是否為樓梯的情況直接進行左右的條件。判斷「囧」是否位於樓梯上的方式為：將「囧」下面一行掃過，如果有任何一點並非

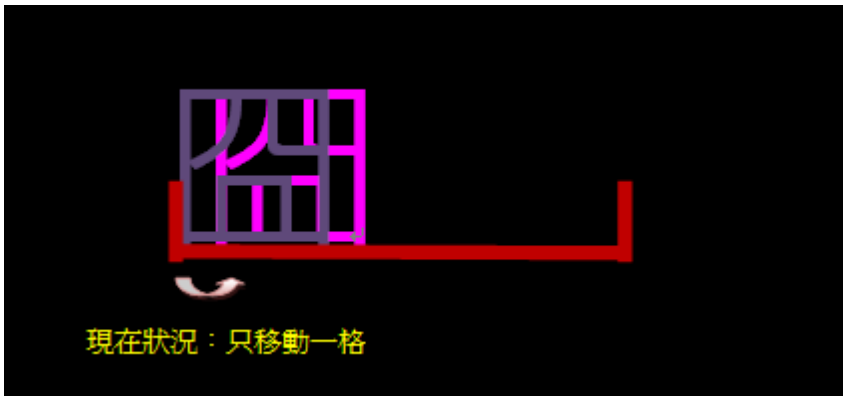
黑色，便屬於在樓梯上的 condition。在此使用了 int 10h 的 0Dh(Read Graphics Pixel)，在螢幕上特定位置讀取 graphics pixel，並回傳 pixel value 於暫存器 al 中。範例 code 如下：

```
        mov     cx, 15                  ; 判斷是否在 stair 上面
    L_checkStair0:
        push    cx
        mov     ah, 0Dh                 ; Read Graphics Pixel
        mov     bh, 0                   ; video page 0
        mov     cx, currentX
        mov     dx, currentY
        int     10h
        cmp     al, 0                   ; 看是否為背景顏色(黑)
        jne     KP_left2                ; 若不是的話則為在樓梯上
        inc     currentX
        pop     cx
        Loop    L_checkStair0           ; 檢查整個「囧」的下方
```

在這樣的判斷方式之下，因為此時「囧」位於|_____|形樓梯上方，因此便不會跳去判斷左右是否為樓梯的 condition，也就不會直直往下墜而產生上圖情況了。

在此處遇到的第二問題是，由於速度的關係我們將左右移動都設為一次移動兩格，但若「囧」在|_____|形樓梯上方，且離凹槽左(右)一格之遠時，如果玩家此時按了左(右)鍵，便會一次向左(右)移動兩格，那麼就會把樓梯吃掉了！此時便必須要動用判斷左右是否為樓梯的狀況，並且一次要看離左(右)相隔一格之外的 pixel，若為樓梯顏色則只移動一格，否則仍然維持一次移動兩格。





沒想到多新增一個樓梯可以製造這麼多的麻煩！不過，在完成這種|_____|形樓梯之後，我們的遊戲性又更增加了！當玩家笨笨的跳到凹槽中，便會發生非常囧的事情—它將無法跳出…。

戳到必死無疑的 針

實作：林士涵
Report：林士涵

研究動機

最初的想法是想要做一些在畫面上高速移動的破壞性物品，以增加遊戲性。

研究過程

起初的目標是由下方冒出來的火箭，然而把 15X15 的火箭畫出來後，發現他的寬度十分寬，向上飛時(X 值皆不變，也就是直直向上)，樓梯最左邊的點會被吃掉的機會很大(因為此時樓梯的基準點會變，樓梯下一次移動時會向右方變形)，所以就開始想其他的替代方案，最後想到的方法是樓梯產生時最左點的 X 座標限定成三的倍數(0,3,6,9…)，而高速移動的破壞性物品改成從天而降的針，寬度只有二，如此一來只要將針產生的最左點的 X 座標限制為三的倍數加一即可避免上述的情形。

EX

零一二三四五六七八九十…X 座標

● ● ● ● …樓梯最左點(基準點）可能在的位子

○○ ○○ ○○ ○…針可能在的位子

然而最後寫出來執行的結果反而導致了整個針和樓梯系統的隨機產生變得十分規律，在幾經研究還無法參透之時，決定放棄這樣的念頭，也就是取消了這樣的生產位置限制，讓他在正常的情形下有著 2(針的寬度二)X6(一個畫面最多六個樓梯)/319(共這麼多種針的生產情形)，也就是 12/319 的機會會發生樓梯最左點被吃掉的情形進而使樓梯右移一到兩個 PIXEL，這算是生產高速破壞性系統最困難的地方，此外，為了讓遊戲不會太難，且在撰寫時較好掌控，我們限制了一個畫面中最多只會有一根針，讓他的移動以及判斷是否有囧在行進的路上較為單純，而被針砸到就死掉的判斷方式也就是在移動針之前判斷目前真的下方 3(因為每次向下移動三個 PIXEL)X2(寬度是二)個 PIXEL 是否有囧的值(顏色)，若有即直接判斷死亡。

EX

Y
座
標
↓
零○○
一○○←針的最底端
二？？
三？？
四？？←須判斷的有無囧的範圍
五

最後我們就在樓梯會有小部分機率被吃到變形的情形下完成了這項致命武器，將它放在充滿挑戰危機四伏的第三階段，期待帶給玩家與眾不同的感受。

破關的籌碼—記分板的設計

實作：陳奕安、林士涵
Report：陳奕安

研究動機

一個遊戲總是要有一些紀錄玩家目前進度的機制，基於這個簡單想法，我們設計了”記分板”。

研究過程：

共有兩個部份，一個是計分板的著色，一部分是加減分的判定。

著色：

在遊戲的一開始，就在螢幕的最上方畫一條 320*9 的白色當作計分條, 接著每加一分就塗上一塊 8*9 的方格，所以總共是 40 分，至於塗什麼顏色，則是依照目前分數所在的範圍而定（與遊戲的難度變化有關），當分數小於 13 則塗黃色，代表第一階段（最 Easy），大於 27 則塗紅色，代表第三階段（Hard），中間的則是咖啡色，代表第二階段（Normal），每次要生產出樓梯的時候都會看看分數是幾分，而去決定亂數範圍來調整不同樓梯出現的頻率，而扣分則是反之，不過塗的顏色就變成都塗成白色，而且塗的方塊是 32*9 代表一次扣四分。

加減分判定：

我們的構想是碰到”普通樓梯”一次加一分，然後碰到”帶刺樓梯”一次扣四分，但是不能重複計算，就是當你停留在同一個樓梯上，系統並不會重複判定而造成無止盡的加分或扣分的情形。我們加上了當你滿分的時候或血被扣到零的時候會進入破關畫面或死亡畫面的條件。

在判定加減分的部份，我們設置了一個叫做 ScoreFlag 的 data，他的實際運作情形如下：

每一次回圈都去判斷 CurrentX&Current（即囧現在在螢幕上的座標）的下方有無一般樓梯或帶刺樓梯，若無，則 ScoreFlag 設為 0，接著進行正常的進行遊戲，若有則跳進判斷加減分的迴圈中。

迴圈中若 ScoreFlag 為 0，則可以進行加減分的動作，並在加減分完畢後將 ScoreFlag 設為 1，如此若停留在同一樓梯（普通或帶刺），則會因為加過一次分

後 ScoreFlag 已經變成 1 而進入不了加減分判定迴圈進而避免掉了重複加減分的情形，而一但離開了樓梯，ScoreFlag 又變回 0（因為下面沒有樓梯了），如此下次即可正常加減分。

而每次加減分之後，都會判斷是否分數已經被扣光或滿分，若扣光則進入 GameOver（就跟撞到天花板和地板一樣情況），若破關則進入破關畫面（詳見破關 Report）。

玩家的好幫手—進版畫面

實作：黃詠筑
Report：黃詠筑

研究動機

一個遊戲的根本就是一連串的遊戲介紹，從遊戲名稱、遊戲製作人、遊戲規則和一些特效等等，這些東西雖然不是遊戲的主角，但是他是帶領所有玩家進入遊戲世界的前線，所以重要性不亞於遊戲本身。我們的遊戲主題 JOHNNY GO?!中，雖然沒有像現今一般遊戲一樣有很華麗聳動的特效，然而我們設計的是一種屬於簡潔乾淨，最重要的是畫面切換的 timing 可以有把玩家漸漸帶入遊戲的感覺的頁面。於是，我們選擇了以課本 15.3 VIDEO Programming with INT 10h 介紹中的 function 13h 為幫我們達成目標的輔助，把字串都先宣告成一個個 BAR，每次要印出一個字串時，就先 set ES segment、字串大小和一些擺字串的起始位置，最後便去 INVOKE 一個寫好的 Show_Text function 去把我所要的字串打印出來。而 Show_Text function 中的動作便是我前述的 13h 功能，把要的字串 BAR 以參數型式傳入 FUNCTION，便可以履行到寫字串的動作。而畫面與畫面間的切換如果沒有”Press any key to start the GAME..”則是以 DELAY FUNCTION 遷入秒(1000)數來行使，如果讓每一個切換的畫面間都有一個適合的 TIMING，便可以達成遊戲成功起步的效果，感覺頗有成就感。至於”Press any key to start the GAME..”，

就會用到一些 KEYBOARD 的功能(15.2 Keyboard Input with INT 16h 中的 11h 和 10h)，讓畫面真的可以等到 ANY KEY 才切換。

疑難雜症與解決

製作過程，對個人而言最大的障礙是，我在放假的第二天就因為幾個月前訂好而退不了的機票問題很早出國，更無奈的是，MASM 在我家的電腦除了能編譯，EXE 檔卻跑不起來，編譯過後有 ERROR 等等的反而好處理，倒是編譯成功後才令人頭痛，因為看不見半成品的樣子，我想我的組員們真的很包容也幫助我很多，真的很謝謝他們，透過 MSN 形容給我聽執行的樣子、外觀、畫面切換的速度，排版等等・・。一開始我真的覺得 FUNCTION 很方便，以為只要把某字串丟給 edx 去 CALL WriteString 就會幫我寫字串，再 CALL 個 WaitMsg 就會自動顯示”Press any key to continue..”，而且連等待按

鍵都幫我執行。而一開始還會因為執行完後就會剩一大堆”Press any key to continue..”的狀況，我又去 CALL ClrScr 來幫我處理，於是我得到了一個髒髒、醜醜、灰灰、字小小、順序亂跳、不是我要的結果。於是我翻了課本，找到了很有效率的 13h，果然，幫我大幅改進很多，已經至少可以在畫面中隨意呈現我要的文字，只可惜的灰灰的畫面沒有改善，於是我又想說乾脆寫個調色盤來設定背景為黑，然而最終問題還是存在。幸好我的組員跟我說如果用 ClrScr function 可能就是會導致這種畫面灰灰醜醜的問題，於是他便提了一個以空白字串去遮蓋前一個字串的妙方・・・果然效率倍增，一下子就跑出我們想要的黑底白字簡潔乾淨風，太棒了！！只剩下一些修飾和 TIMING 的拿捏，便讓遊戲的進版畫面大功告成！！

你贏了嗎？—破關畫面

實作：林士涵
Report：林士涵

研究動機

依照一般的常理，遊戲破了關總是希望有點什麼，因此我們為了抓住玩家的這種心理，特別設計了破關畫面，讓玩家得以在破關時享受破關的喜悅。

研究過程

一開始就是構思要如何呈現這個畫面，最初的構想是能有一個特殊且長度足以涵蓋整個 X 軸的超大樓梯，硬是要把玩家壓到天花板而死，但後來考慮到玩家的感受，決定放棄這樣的構想，轉而朝向對於我們此次開發有重大幫助的恩師 CYY，期待在破關畫面能夠與 CYY 相關，最後則決定讓大大的 CYY 出現在螢幕上，並且讓他有機關並且有變色的功能。首先就是讓大大的 CYY 呈現在螢幕上，這個部份我們寫了一個叫做 DrawBlock 的 procedure，他的功能就是傳給他左上角 pixel 的 X 座標、Y 座標、長、寬、顏色如此它就會幫你畫出一個想要的方塊，如此一來只要將 CYY 分解成多個方塊，分別找出他們對應的五個值即可完成。接著在機關的部分就是讓囧剛好可以卡在中間那個 Y 的上方岔口內，使他無法動彈，進而進入變色狀態，而最後就是變色的設定，起初我們是以整個 CYY 如同第一次化的方法在換個顏色畫上去把舊的顏色蓋掉，然而這樣的做法卻無法讓我們的變色自燃，產生了不同步的情形(一塊一塊變，先後很明顯)，所以後來就改成從 CYY 這個圖的左上角開始一個點一個點讀值如果不是背景的黑色也不是囧的紫色時，就換成下一種顏色(當然不能是換成黑色和紫色)，就這樣把整個 CYY 掃完也就換完了顏色，如此一來，換色就順利多了，不但很自然還有由上往下換的效果，對於一個破關畫面來說應該是相當賞心悅目的了。

結語

感謝看完了我們的作品的您，因為這次的作業，讓我們付出了很多，然而，我們得到的更多……

這是我們第一次由一個團隊共同寫一個 code，不像很多常常比賽經驗豐富的同學，一切對我們而言都是相當陌生的，所以我們也遇到了很多的困難，其中我們學到最深刻的經驗就是”程式不能寫死”這件事，因為常常是一個人負責一部份，而每個人的思考邏輯都不盡相同，如果自己的 code 一點彈性都沒有，那很容易在要合併的時候就爛掉了，這點對於大一大部分時間都是孤軍奮戰的我們（無論是鋼彈或是使徒）而言，是很缺乏的，因為以前都只要不計手段達成目的就好了，現在要考慮的因素多太多了，有和其他組員的配合度、遊戲的流暢度、畫面的美觀、使用者的角度等等……很多新的觀念，真的讓我們吃盡了苦頭！

再來就是演算法的重要性。這份遊戲能夠變流暢的最大關鍵就是我們最後想出了讓所有樓梯一起隨機出現並移動的演算法，不然依照原本的爆搜，那小囧就真的可以體會到什麼叫”寸步難行”了。

還有最重要的一點，就是身為一個程式設計者，應該要隨時隨地用電腦、用邏輯的角度去思考問題和分析情況，我們常常寫出一些自己怎麼想都對，但是電腦看起來就跟我們完全是兩回是的 code，應該說是被 C 寵壞了吧？我們的 code 總是充滿著”想當然爾”、”不確定性”……等致命的因素，如總是忘記檢查 stack 現在是否應該是我們要的樣子，也常忘記某些暫存器在這個 function 中扮演的角色，進而對它做出一些錯誤的判斷……但是這些別人看起來很可笑的錯誤，當由我們親手寫出來的時候，可以說是完全陷入盲點而不自知的因為我們早就被自己”理所當然”的錯誤邏輯所蒙蔽了，但是相對的，再我們 de 出 bug 的那一瞬間，我們確實感到我們的實力又往上提升了！

總的而言，這次的組語 Final Project 對我們的幫助真的不小，而且又很有成就感，因為這是我們生平寫出的第一個加入自主創意的小遊戲，也是第一個完全以 Assembly Language 寫出來的程式，以前就算有寫過一些類似遊戲的”作業”，也都只是將教授要求的條件一一加入，程式本身是不具有 programmer 的意志的，但是這次作業卻讓我們做出了一個”活蹦亂跳”的小囧，看到我們自己辛苦設計的創意一一被實現，那份感動真的不可言喻，如果每份作業都那麼好玩的話，那寫作業這件事就真的一點都不覺得辛苦了！

Johnny Go !?

作者：邱珮甄 (chiuma)

陳奕安 (ilcvmy)

林士涵 (hotlin1224)

黃詠筑 (clairhyc)

策畫：台大資訊工程系 莊永裕教授 (cyy)

編輯/美術：邱珮甄

感謝名單：

莊永裕教授

Fall, 2007 ASM TAs

楊逸民 (asurada0207)