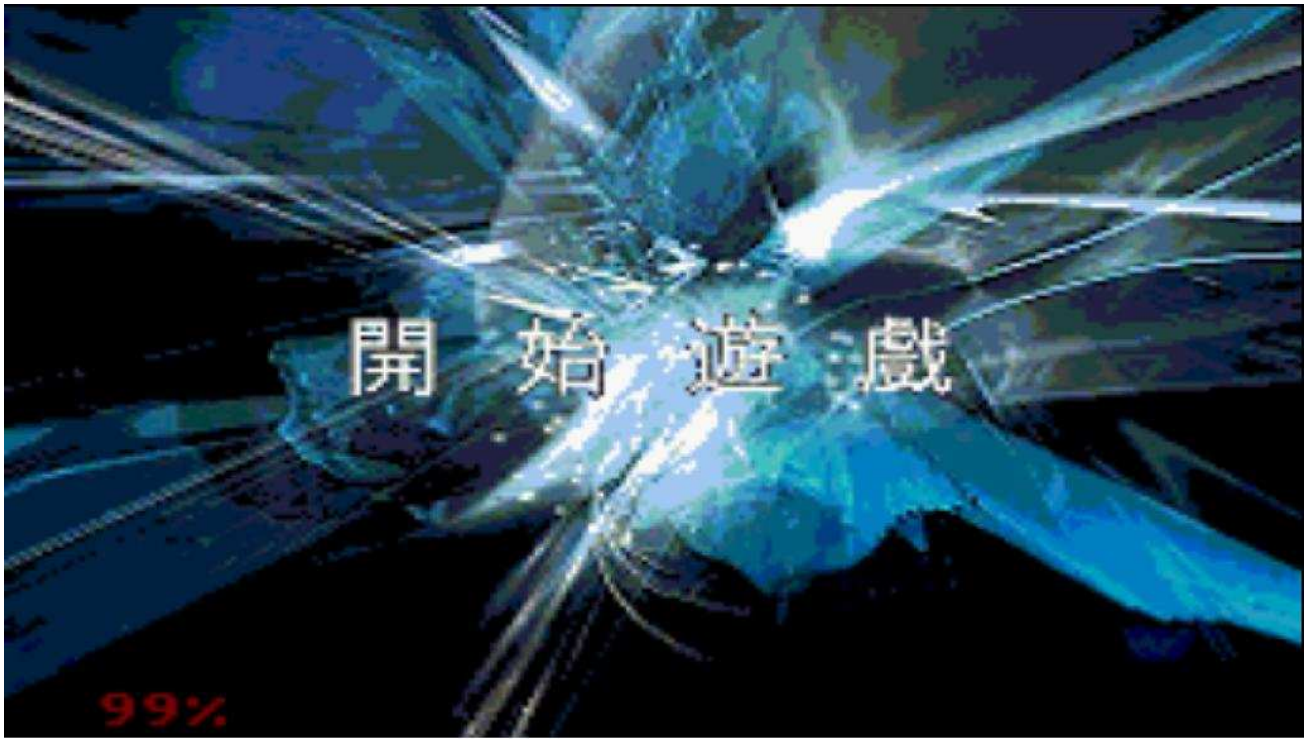
Computer Organization and Assembly Languages Final Project

Catch Bananas!

B96902030 郭宗倫
B96902032 徐國翔
B96902084 詹益齊

遊戲畫面

開始背景



結局圖

結局圖 2

一. 遊戲介紹

rabbit要盡量把所有掉下的banana接住，而banana會隨著吃得banana數越來越快，如果沒吃到掉到地上則會減少生命值,吃到apple則可以增加生命值;遊戲有五次機會可以沒接到banana,第六次沒接到則遊戲結束，如果接完全部banana則達成任務，並計算分數。

二. 為何要實作此遊戲？

以前玩很多RPG遊戲像是大富翁等等都會玩到類似這樣小遊戲,那時候就覺得這樣的遊戲基本道理應該是滿簡單的，以前也覺得遊戲製作還不簡單，指令弄一弄就好了，但是上完大一的程式設計後卻一個遊戲都寫不出來，實在讓我覺得自己不像是資工系的學生。所以這次組語的期末報告剛好有機會可以嘗試來實作遊戲，但太複雜的遊戲又超過我們的能力跟時間限制，所以就決定寫這個小遊戲。

三. 指令介紹

按鍵左，rabbit向左移動。
按鍵右，rabbit向右移動。
按鍵 Enter，遊戲開始。

Function Introduction:

void query_buttons()—使用者輸入，按方向鍵左(右)可使 rabbit 的方向參數變為左(右)並且位置參數改變。

void update_rabbit_gfx()—在 query_buttons()改變 rabbit 的方向參數之後，此 function 可使 rabbit 變更為方向參數的方向。

void update_rabbit_pos()—在 query_buttons()改變 rabbit 的位置參數之後，此 function 可使 rabbit 的位置改變。

void creat_banana()—隨機製造由出不同位置落下的 banana。

void drop_banana()—令 banana 落下。當 banana 尚未落地時,banana 持續落下。當 banana 觸地時，則移除 banana 並且扣除 rabbit 的生命值。當 banana 在落地前被 rabbit 碰觸到時，則移除 banana 並加一分分數。

void update_banana()--在 drop_banana()改變 banana 的位置參數之後，此 function

可使 banana 的位置改變。

void creat_apple()—隨機製造出由不同位置落下的 apple。

void drop_apple()—另 apple 落下。當 apple 尚未落地時，apple 持續落下。當蘋果觸地時，則移除 apple。當 apple 在落地時被 rabbit 碰到，則移除 apple 且生命加一。

void update_apple()—在 drop_apple()改變 apple 的位置參數之後，此 function 可使 apple 的位置改變。

void update_number_gfx()—將目前分數即時地顯示於螢幕的右上角。

參數解析

```
ANIM_RIGHT        1       //於 rabbit 左右圖式陣列中向左圖式的起點
ANIM_LEFT         0       //於 rabbit 左右圖式陣列中向右圖式的起點
NUM               50      //banana 總個數
u8 rabbit3[5];            //代表左上生命值的圖示
u8 title[4];              //"遊"戲"開"始"四字圖示
u8 k = 4;                 //目前顯示生命值圖示的 index
u8 rabbit;                //rabbit 物件
u8 rabbit_x = 110;        //rabbit 物件 x 作標
u8 rabbit_y =   128;      //rabbit 物件 y 作標
int i = 0;                //已產生 banana 物件的總數，結束回合時被設為-1
int timeToCreat;          //決定是否產生 banana 物件的隨機數
int score = 0;            //獲得分數
int HP = 100;             //剩餘生命值
u8 speed = 1;             //各個階段 banana 下降速度
u8 speed2 = 3;
u8 kl = 0;
int seed = 0;             //亂數種子，由 198:while 執行次數決定
int f = 200;              //決定 banana 產生時差
int api=0;                //已產生 apple 物件的總數
int apNUM = 4;            //apple 物件最大總數
int apTimeToCreat = 0;    //決定是否產生 apple 物件的隨機數
u8 banana[NUM];           //banana 物件陣列
u8 banana_x[NUM];         //banana 物件 x 作標
u8 banana_y[NUM];         //banana 物件 y 作標
```

```
u8 apple[NUM];              // banana 物件陣列
u8 apple_x[NUM];            // banana 物件 x 作標
u8 apple_y[NUM];            // banana 物件 y 作標
u8 number[2];
u8 counter[2]={0,0};
int vbl_count = 0;
int vbl_count2 = 0;
```

reference

HAM Tutorial http://www.aaronrogers.com/ham/

林仲丘, 王舜玄, GBA Programming in Ham Strike!, 6-29

心得

從這次的實作中，其實我覺得任何外面看似簡單的遊戲其實內容都不是那麼簡單的，尤其在GBA的限制之下，很多的圖形或是寫Code的方式都受到很大的限制，因為不同的時空背景，在他們那個年代就是這樣製作遊戲的。

其實這次因為時間很趕,大概兩三天內要完全理解一個之前並不是很熟悉的作業環境跟語言，有些東西並沒有真正理解就趕鴨子上架直接使用，其實滿可惜的，不過有了起頭之後，要理解其他類似的語言其實都變得十分容易，就像這學期學的三種語言，雖然指令內容會是使用方式上略有不同，但是他們的本質是相同的。而這些組合語言的理解讓我們更了解在高階語言的下一層他們是如何運行的，也對電腦有了更深一層的認識。

分工

詹益齊　寫 CODE、找圖

徐國翔　寫 CODE、DEBUG

郭宗倫　寫 CODE、DEBUG