

# Dance!


## 第 45 組


108504504 林佳儀

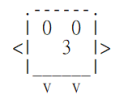
108504505 許方昱

1. 遊戲規則.....	2
2. 遊戲流程.....	2
3. 函式介紹.....	4
4. 引入函式庫 .....	7

## 1. 遊戲規則: 在指定範圍裡按下對應按鍵

 :按上下左右 +20 分

 :紅蘿蔔->空白鍵 +10 分

 :小怪獸->空白鍵(是陷阱不能按) -10 分

- 未於指定區域按下對應按鍵或於範圍內按錯按鍵-10 分

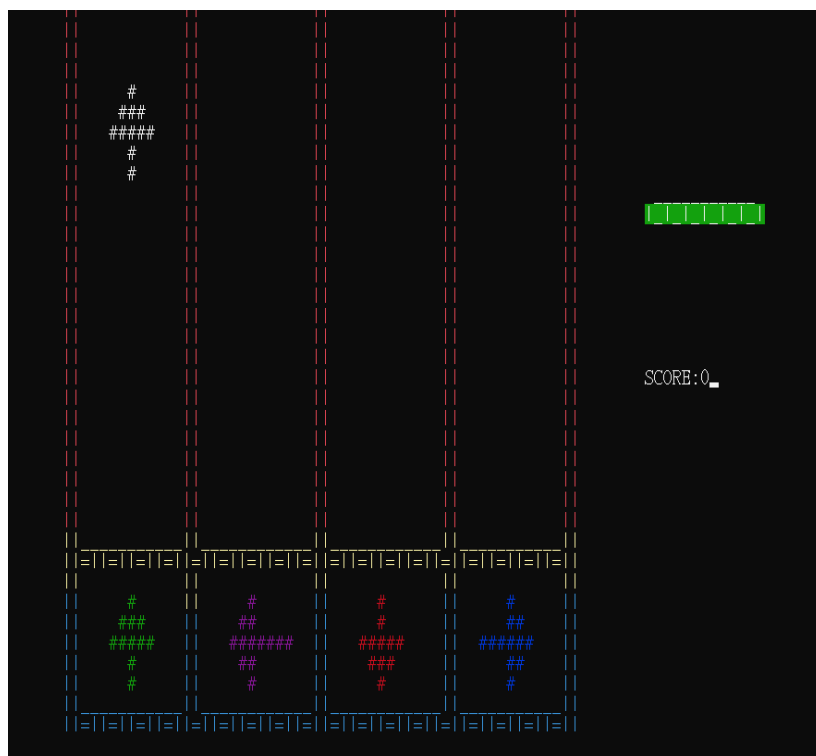
## 2. 遊戲流程

初始畫面 (由下面兩張圖來回切換做成動畫)

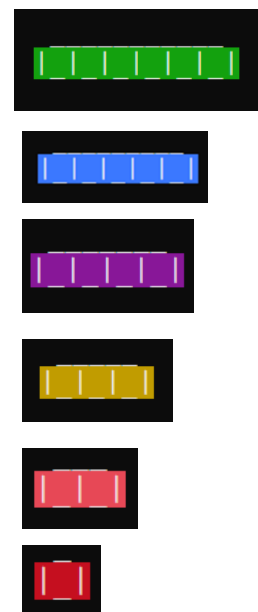
按下空白建即可開始遊戲



關卡顯示:

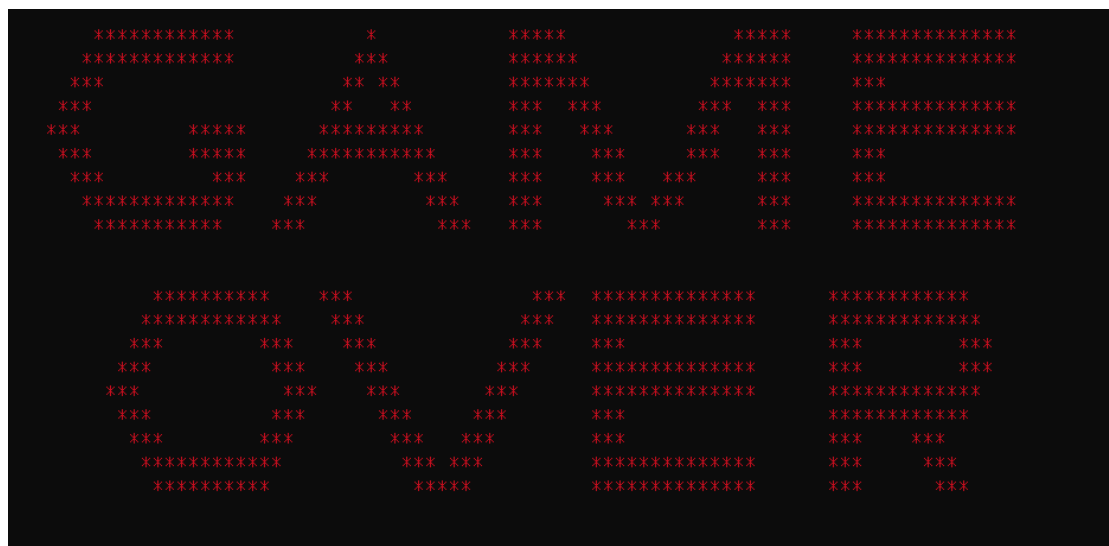


能量條變化:



結束畫面:

如果遊戲仍然未結束但是能量條已經清空。則顯示 gameover 的字串。



若遊戲結束，玩家仍有剩餘能量條。則顯示 you win 的字串。



### 3. 函式介紹

起始頁面：

#### Create\_str

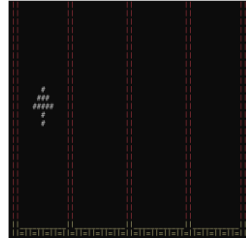
輸入座標 `x,y`、預期要印出的字串指標以及字串顏色，利用 `Gotoxy` 將游標移動到指定座標、`setTextColor` 改變要顯示字串的顏色，並利用 `Writestring` 將字串印出。

#### DanceStart

啟動遊戲標題動畫，利用 `readKey` 來讀取 `space`，判斷是否要進入遊戲畫面。

#### DanceStr\_Animation

根據 `Dance_Animation_Flag` 判斷要印出哪組標題字串，結束時將 `Dance_Animation_Flag` 設為補數。



箭頭移動：

## StartGame/StartGame1/StartGame2/StartGame3

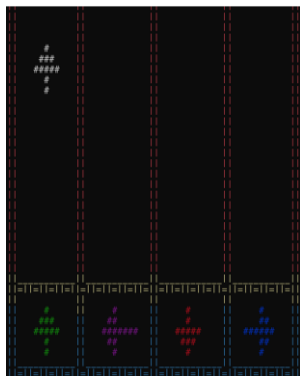
四個函式分別代表四個箭頭的軌道，透過迴圈在下一單位  $y$  座標為起始點創造新白色箭頭 (創造同時清除原本的白色箭頭讓外觀呈現見慢慢下移的樣子)，每一個下移的新箭頭都會呼叫測試碰撞，若偵測到答對的碰撞並在合理範圍內就會進行加分並重新呼叫地圖(使用 Random 產生新白色箭頭)；按錯情形發生時將會扣分，白色箭頭繼續執行迴圈往下移直到到地圖底端後，再使用 Random 產生新白色箭頭。

## StartGameMove/StartGameMove1/StartGameMove2/

## StartGameMove3

作為輔助 StartGame/StartGame1/StartGame2/StartGame3 之功能，為避免各個 StartGame 中迴圈過度冗長，將產生白色箭頭的步驟另外拉出來實踐。

## map\_build



繪製基本地圖(軌道、彩色箭頭)，利用課堂中教的 WriteConsoleOutputAttribute WriteConsoleOutputCharacte 和自定義函式 Creat\_str 分別諄對各個字元做調色設定來繪製彩色軌道及箭頭。

## carrotRow

當遊戲預計要產生 carrot 的物件時，會 call carrotRow。carrotRow 會利用 randomRange 產生隨機的數字，每個數字對應不同的道路。

## carrot

作為輔助 `carrotStr` 之功能。將紅蘿蔔往下位移的步驟獨立出來，為了避免在 `carrotStr` 裡的迴圈過於冗長，造成暫存器空間不足無法順利執行的情況。

## carrotStr

透過迴圈在下一單位 `y` 座標為起始點創造新物件(創造同時清除原本的物件讓外觀呈現見慢慢下移的樣子)，每一個下移的新物件都會呼叫測試碰撞，若偵測到答對的碰撞並在合理範圍內就會進行加分並重新呼叫地圖(使用 `Random` 產生新物件)；按錯情形發生時將會扣分，物件會繼續執行迴圈往下移直到到地圖底端後，再使用 `Random` 產生新物件。

## monster

作為輔助 `monsterStr` 之功能。將小怪獸往下位移的步驟獨立出來，為了避免在 `monsterStr` 裡的迴圈過於冗長，造成暫存器空間不足無法順利執行的情況。

## monsterStr

透過迴圈在下一單位 `y` 座標為起始點創造新物件(創造同時清除原本的物件讓外觀呈現見慢慢下移的樣子)，每一個下移的新物件都會呼叫測試碰撞，若偵測到答對的碰撞並在合理範圍內就會進行加分並重新呼叫地圖(使用 `Random` 產生新物件)；按錯情形發生時將會扣分，物件會繼續執行迴圈往下移直到到地圖底端後，再使用 `Random` 產生新物件。同時，設定 `MonstreFlag = 1`，以方便後續的碰撞判斷。

## monsterRow

當遊戲預計要產生 `monster` 的物件時，會 `call monsterRow`。`monsterRow` 會利用 `randomRange` 產生隨機的數字，每個數字對應不同的道路。

## 碰撞判斷：

### Collision:

在 `collision` 中，會利用 `ReadKey` 來讀取玩家的鍵盤。並且透過 `whichArrow` 的變數比玩家所按的鍵是否與要求相同。若相同則將 `pressFlag` 設定為 `1`，方便後續跳回上一個迴圈時，來做清除畫面的判斷條件。還有 `lifeDetect` 設定為 `1`。同時利用 `monsterFlag` 來判斷按對鍵後，`score` 的加減條件。反之，若玩家按錯鍵，則 `score` 減 `10` 分(如果 `score` 已經是 `0` 的情況則不再扣分)。再透過 `WriteString` 將改變後的 `score` 顯示出來。若同時 `score = 0` 和 `lifeDetect = 1`，則對能量條進行改變。

結束畫面：

### GameoverStr

若  $lifeCount = 0$ ，call `GameoverStr`。 `GameoverStr` 會 invoke `Create_str` 將 “GAME OVER”的字串顯示出來。

### winStr

若遊戲預設的迴圈次數已經結束且  $lifeCount > 0$ ，call `winStr`。 `winstr` 會 invoke `Creat_str` 將 “YOU WIN” 的字串顯示出來。

能量條控制：



### CleanLife

當分數再次歸零時會需要更新能量條，利用類似 `Create_str` 的方式將預設空白字串將原本的能量條清空。

### Create\_life1

創造能量條方框上方底線，利用傳入不同長度的方框底線達成顯示相對命數的能量條上方方框。

### Create\_life2

創造能量條的本體，利用 `if` 判斷當前的命有幾條分別利用類似 `Crate_str` 的方式設置不同背景顏色和長度的能量條的樣式。

## 4. 引入函式庫

FROM Irvine32 Library

`Clrscr` ;清除螢幕

`Delay` ;暫停程式

`ReadKey` ;讀取鍵盤輸入

SetTextColor	;設定字體顏色
GoToxy	;定位游標位置
WriteString	;輸出字串
WriteDec	;輸出數字(以十進位表示法)
RandomRange	;產生隨機變數(在我們的預設範圍內)
WriteConsoleOutputAttribute	;設定單位屬性
WriteConsoleOutputCharacte	;設定單位字元