組語專題:

Google小恐龍(改)

組員:

資工2A-109502535-湯騏蔚

資工2A-109502536-沈富堅

分工狀況

1. 湯騏蔚(50%)
   * 開始、暫停、結束頁面(使用讀檔輸出)
   * 排行榜系統及畫面
   * 角色跳躍機制
   * debug
2. 沈富堅(50%)
   * 基本程式運行
   * 各種物體產生機制
   * 遊戲平衡性調整
   * PPT&書面報告製作
   * debug
3. 分工特色-使用github進行管理
   * https://github.com/jason01180118/ASMFinalProject
   * 有助於合作時運行的效率，在製作過程中可以時時更新狀態
   * 避免修改到隊友的程式碼導致不可預期的錯誤
4. 分工圓餅圖

遊戲玩法&功能介紹

1. 有開始、暫停、結束頁面(使用讀檔的方式)
   * 開始畫面有玩法說明並且可以查看排行榜
   * 結束畫面可重新開始或查看排行榜
2. 按下空白鍵進行跳躍(跳躍期間無法再次跳躍)
3. 不同方塊的意思
   * H(玩家操縱的主角)
   * X(障礙物，撞到就結束遊戲)
   * Z(彈簧，直接高速向上跳躍)
   * C(加速板，瞬間加速並且無視所有方塊)
   * O(金幣，可以讓分數增加)
4. 機率及延遲調整機制
   * 分數計算
   * 利用分數對速度進行調整(影響delay)
   * 利用分數對障礙物生成進行調整(機率控制)
5. 排行榜機制(**demo沒有的功能**)
   * 在遊戲結束的時候顯示分數兩秒然後請使用者輸入名稱並記錄進入排行榜
   * 記錄從始至今的前5名(用讀寫檔案的方式保留排行紀錄)
   * 可在開始畫面跟排行榜畫面切換

程式架構&困難之處

1. 函式
   * 由於我們希望整個程式的可讀性是非常高且明確易懂，所以我們把每一項功能都分開來寫，雖然原理差不多但我們不希望他們共用一個變數或陣列，這樣的話如果想要有不同的生成機制，那在更新上將會相對困難。

consoleChange PROTO                         ;螢幕清除並畫線

characterCheck PROTO                        ;判斷角色位置

groundCheck PROTO                           ;判斷地板位置

enemyCreate PROTO                           ;判斷敵人是否生成

enemyDraw PROTO                             ;判斷是否畫出敵人

enemyMove PROTO                             ;判斷前方是否有敵人並向前移動

gameOver PROTO                              ;判斷是否撞上敵人

springCreate PROTO                          ;判斷彈簧是否生成

springDraw PROTO                            ;判斷是否畫出彈簧

springMove PROTO                            ;判斷前方是否有彈簧並向前移動

springDetect PROTO                          ;判斷是否撞上彈簧

accelerateCreate PROTO                      ;判斷加速板是否生成

accelerateDraw PROTO                        ;判斷是否畫出加速板

accelerateMove PROTO                        ;判斷前方是否有加速板並向前移動

accelerateDetect PROTO                      ;判斷是否撞上加速板

coinCreate PROTO                            ;判斷金幣是否生成

coinDraw PROTO                              ;判斷是否畫出金幣

coinMove PROTO                              ;判斷前方是否有金幣並向前移動

coinDetect PROTO                            ;判斷是否撞上金幣

scoreConsole PROTO                          ;顯示分數

endingScreen PROTO                          ;結束頁面

beginScreen PROTO                           ;開始頁面

pauseScreen PROTO                           ;暫停頁面

initialization PROTO                        ;初始化

rankScreen PROTO                            ;排名畫面

rank PROTO                                  ;判斷排名

1. 變數名稱
   * 基本上變數名稱就代表了功用，所以這邊就沒有註解的部分，取名方式使用固定的方式，也就是lower camel case。

block BYTE ?

restart BYTE ?

enemyProbability DWORD 10000

springProbability DWORD 10000

accelerateProbability DWORD 20000

coinProbability DWORD 50000

delayTime DWORD 50

begintext BYTE 10000 DUP(?)

pausetext BYTE 10000 DUP(?)

endingtext BYTE 10000 DUP(?)

enemyRow BYTE 120 DUP(0)

springRow BYTE 120 DUP(0)

enemyHeight WORD 120 DUP(0)

accelerateRow BYTE 120 DUP(0)

accelerateHeight WORD 120 DUP(0)

coinRow BYTE 120 DUP(0)

coinHeight WORD 120 DUP(0)

height DWORD 0

onGround WORD 20

ground WORD 21

enemy DWORD 0

spring DWORD 0

accelerate DWORD 0

aheight DWORD 0

kingKrim DWORD 0

coin DWORD 0

cheight DWORD 0

outputHandle DWORD 0

inputHandle DWORD 0

count DWORD 0

xyPosition COORD <0,12>

characterPosition COORD <10,20>

scoreTitleStringPosition COORD <102,0>

scorePosition COORD <113,0>

smallRect SMALL\_RECT <0,0,120,30>

consoleScreen COORD <120,30>

jumping BYTE 0

gameoverCheck BYTE 0

score DWORD 0

scoreTitleString BYTE "your score:" , 0

beginFile BYTE "START.txt",0

pauseFile BYTE "PAUSE.txt",0

endingFile BYTE "OVER.txt",0

rankAsking BYTE "What's your name(10 character at most):",0

WrongName BYTE "contain invalid character",0

NameTooLong BYTE "too long",0

rankScoreFile BYTE "rankScore.txt",0

rankNameFile BYTE "rankName.txt",0

backToStart BYTE "PRESS SPACE TO BACK TO MENU",0

endTheGame BYTE "PRESS OTHER KEY TO END THE GAME",0

fromEndScreen BYTE 0

1. 初始化
   * 將變數們回歸初始狀態以免發生錯誤

initialization PROC USES eax ebx ecx esi        ;初始化

    call Randomize

    mov enemyProbability,10000

    mov delayTime,50

    mov ecx,120

    mov esi,0

  INITIAL:

    mov [enemyRow+esi],0

    mov [springRow+esi],0

    mov [accelerateRow+esi],0

    mov [coinRow+esi],0

    mov [enemyHeight+esi],0

    mov [accelerateHeight+esi],0

    mov [coinHeight+esi],0

    inc esi

    LOOP INITIAL

    mov xyPosition.x,0

    mov xyPosition.y,12

    mov characterPosition.x,10

    mov characterPosition.y,20

    mov jumping,0

    mov gameoverCheck,0

    mov score,0

    mov kingKrim,0

    ret

    initialization ENDP

1. 創建畫面
   * 一開始本來想要用字串輸出成整個畫面，但後來發現會導致程式碼過於冗長，於是改用讀取檔案的方式，將畫面先在txt檔案裡設計好，再於遊戲運行時讀取檔案，同時也使得畫面排版設計更容易。
2. 讀寫檔案
   * 本以為讀寫檔案不會太困難，但實際上卻遇到了許多難處與bug。首先，因為讀取檔案的函式將檔案內容讀取進入buffer之後不會在最後面加上null，所以直接使用WriteString函式會導致輸出過多的字元亂碼，因此改用WriteConsole指定輸出字數。
   * 中途出現了一個bug發現若多次在暫停畫面與遊戲中切換，則暫停畫面可能出現亂碼，後來發現似乎是在LOCAL變數宣告的時候要求的記憶體太小，因此將buffer調整成4000而非原本的剛剛好3600字元。
   * 在製作排行榜系統時需要多次的開啟READ Handle與WRITE Handle，而且在讀取檔案的時候需要原有資料，在寫入檔案的時候則是要完全覆蓋原本資料，所以無法使用Irvine Library的讀寫檔函式，因此將所有的讀寫檔函式都改為使用Windows Library的CreateFile、ReadFile、WriteFile，同時因為參數不須放在暫存器中，也將程式碼稍微精簡了一些。
3. 排行榜運算
   * 一開始的想法是將使用者名稱與分數一同記錄進檔案，但後來發現並不是所有的編碼都有對應的字元，因此不能將整數型態的遊玩分數直接當作字串紀錄進檔案，改成使用雙層迴圈與條件判斷，把分數轉成字串型態再寫進檔案中，並且因為使用者名稱沒有固定長度，因此將名稱與分數分成不同檔案紀錄，而分數則是在前面補0直到滿10位數。
   * 在讀取榜上資料的時候，因為使用者名稱不像分數一樣可以用每10位為單位判斷是第幾個人，所以使用’|’字符當作每個名稱的分隔，並且在遊玩結束輸入名稱的時候加上判斷不能含有’|’字符，否則就要重新輸入。
   * 最終的排行榜函式分為幾個大部分，讀取使用者名稱，讀取舊有檔案內容，迴圈運算將舊有分數字串改成整數型態並與當前新分數比較，然後寫入進新scoreBuffer同時記錄當前新分數是第幾名，依照記錄好的當前新分數名次將新名稱插入進舊有的榜上名稱並記在新nameBuffer，然後分別將scoreBuffer與nameBuffer覆寫進檔案中。
4. 輸出畫面
   * 由consoleChange進行判斷，從第十行開始一格一格進入函式，分數則是藉由控制游標位置再call出來
   * 由幾個不同函式組成，分別畫出角色、障礙物、功能方塊、地板，利用座標以及功能方塊自己的陣列與高度等變數來進行位置調整

consoleChange PROC                          ;畫出遊戲畫面

    mov ecx,10

    push xyPosition                         ;紀錄起點

  DRAWLINE:                                 ;行數

    push ecx

    push xyPosition.X                       ;紀錄x位置

    mov ecx,CMDWIDTH

  DRAWROW:                                  ;列數

    push ecx

    mov block,' '

    INVOKE characterCheck                   ;判斷角色位置

    INVOKE groundCheck                      ;判斷地板位置

    INVOKE enemyDraw                        ;判斷畫出敵人

    INVOKE springDraw                       ;判斷畫出彈簧

    INVOKE accelerateDraw                   ;判斷畫出加速板

    INVOKE coinDraw                         ;判斷畫出金幣

    INVOKE WriteConsoleOutputCharacter,     ;輸出一格

       outputHandle,

       ADDR block,

       1,

       xyPosition,

       ADDR count

    pop ecx

    inc xyPosition.X

    LOOP DRAWROW                            ;增加x座標

    pop xyPosition.X

    pop ecx

    inc xyPosition.Y                        ;座標換到下一行位置

    LOOP DRAWLINE

    pop xyPosition

    INVOKE SetConsoleCursorPosition,         ;讓游標位置固定，顯示分數字串

        outputHandle,

        scoreTitleStringPosition

    mov edx,OFFSET scoreTitleString

    call WriteString

    ret

    consoleChange ENDP

characterCheck PROC USES eax ebx ecx        ;判斷角色位置

    mov ax,characterPosition.X

    shl eax,16

    mov ax,characterPosition.Y

    mov bx,xyPosition.X

    shl ebx,16

    mov bx,xyPosition.Y

    .IF eax==ebx                  ;利用eax ebx存取座標並比較,若相同則畫上0

      mov block,'H'

    .ENDIF

    ret

    characterCheck ENDP

groundCheck PROC USES eax ebx ecx           ;判斷地板位置

    mov ax,ground

    mov bx,xyPosition.Y

    .IF ax==bx                     ;利用ax bx存取座標並比較,若相同則畫上-

      mov block,'-'

    .ENDIF

    ret

    groundCheck ENDP

enemyDraw PROC USES eax ebx ecx esi         ;判斷是否畫出敵人

    movzx esi,xyPosition.X                  ;如果當前X座標對應到敵人陣列中不是1就不畫

    .IF [enemyRow+esi]==1

      mov ax,ground                         ;如果當前Y座標不是地板上就不畫

      sub ax,[enemyHeight+esi]

      mov bx,xyPosition.Y

      .IF ax<=bx && bx<=onGround

        mov block,'X'

      .ENDIF

    .ENDIF

    ret

    enemyDraw ENDP

springDraw PROC USES eax ebx ecx esi         ;判斷是否畫出彈簧

    movzx esi,xyPosition.X       ;如果當前X座標對應到彈簧陣列中不是1就不畫

    .IF [springRow+esi]==1

      mov ax,onGround                       ;如果當前Y座標不是地板上就不畫

      mov bx,xyPosition.Y

      .IF ax==bx

        mov block,'Z'

      .ENDIF

    .ENDIF

    ret

    springDraw ENDP

accelerateDraw PROC USES eax ebx ecx esi         ;判斷是否畫出加速板

    movzx esi,xyPosition.X     ;如果當前X座標對應到加速板陣列中不是1就不畫

    .IF [accelerateRow+esi]==1

      mov ax,ground                     ;如果當前Y座標不是地板-高度就不畫

      sub ax,[accelerateHeight+esi]

      mov bx,xyPosition.Y

      .IF ax==bx

        mov block,'C'

      .ENDIF

    .ENDIF

    ret

    accelerateDraw ENDP

coinDraw PROC USES eax ebx ecx esi         ;判斷是否畫出金幣

    movzx esi,xyPosition.X                  ;如果當前X座標對應到金幣陣列中不是1就不畫

    .IF [coinRow+esi]==1

      mov ax,ground                    ;如果當前Y座標不是地板-高度就不畫

      sub ax,[coinHeight+esi]

      mov bx,xyPosition.Y

      .IF ax==bx

        mov block,'O'

      .ENDIF

    .ENDIF

    ret

    coinDraw ENDP

characterCheck PROC USES eax ebx ecx        ;判斷角色位置

    mov ax,characterPosition.X

    shl eax,16

    mov ax,characterPosition.Y

    mov bx,xyPosition.X

    shl ebx,16

    mov bx,xyPosition.Y

    .IF eax==ebx                  ;利用eax ebx存取座標並比較,若相同則畫上0

      mov block,'H'

    .ENDIF

    ret

    characterCheck ENDP

groundCheck PROC USES eax ebx ecx           ;判斷地板位置

    mov ax,ground

    mov bx,xyPosition.Y

    .IF ax==bx                     ;利用ax bx存取座標並比較,若相同則畫上-

      mov block,'-'

    .ENDIF

    ret

    groundCheck ENDP

enemyDraw PROC USES eax ebx ecx esi         ;判斷是否畫出敵人

    movzx esi,xyPosition.X                  ;如果當前X座標對應到敵人陣列中不是1就不畫

    .IF [enemyRow+esi]==1

      mov ax,ground                         ;如果當前Y座標不是地板上就不畫

      sub ax,[enemyHeight+esi]

      mov bx,xyPosition.Y

      .IF ax<=bx && bx<=onGround

        mov block,'X'

      .ENDIF

    .ENDIF

    ret

    enemyDraw ENDP

springDraw PROC USES eax ebx ecx esi         ;判斷是否畫出彈簧

    movzx esi,xyPosition.X       ;如果當前X座標對應到彈簧陣列中不是1就不畫

    .IF [springRow+esi]==1

      mov ax,onGround                       ;如果當前Y座標不是地板上就不畫

      mov bx,xyPosition.Y

      .IF ax==bx

        mov block,'Z'

      .ENDIF

    .ENDIF

    ret

    springDraw ENDP

accelerateDraw PROC USES eax ebx ecx esi         ;判斷是否畫出加速板

    movzx esi,xyPosition.X     ;如果當前X座標對應到加速板陣列中不是1就不畫

    .IF [accelerateRow+esi]==1

      mov ax,ground                     ;如果當前Y座標不是地板-高度就不畫

      sub ax,[accelerateHeight+esi]

      mov bx,xyPosition.Y

      .IF ax==bx

        mov block,'C'

      .ENDIF

    .ENDIF

    ret

    accelerateDraw ENDP

coinDraw PROC USES eax ebx ecx esi         ;判斷是否畫出金幣

    movzx esi,xyPosition.X                  ;如果當前X座標對應到金幣陣列中不是1就不畫

    .IF [coinRow+esi]==1

      mov ax,ground                    ;如果當前Y座標不是地板-高度就不畫

      sub ax,[coinHeight+esi]

      mov bx,xyPosition.Y

      .IF ax==bx

        mov block,'O'

      .ENDIF

    .ENDIF

    ret

    coinDraw ENDP

INVOKE SetConsoleCursorPosition,            ;讓游標位置固定，顯示分數

        outputHandle,

        scorePosition

    mov eax,score

    call WriteDec

1. 字元讀取
   * 一開始我們使用的是實習課使用的readchar進行實作，但這使得在未讀取字元時沒辦法繼續執行，與我們所需要的功能不符，經過一陣子的查詢才終於找到readkey的功能來控制角色跳躍和暫停遊戲的功能
   * 而開始、暫停與結束頁面則需要玩家按下案件後才進行下一步，所以使用的是readchar的方式
2. 角色跳躍
   * 用ReadKey接收鍵盤輸入，若為空白件則判斷跳躍
   * 我們使用了一個變數jumping來偵測跳躍的狀態，並依據狀態和角色位置onGround& characterPosition.Y來判斷是否在跳躍期間或是落下期間，以此做出隨著時間上下移動的手法，並且一次跳躍高度為7格。

call ReadKey

    mov bx,onGround

    .IF ax==3920h && characterPosition.Y==bx

      inc jumping                           ;開始跳躍過程

      dec characterPosition.Y

    .ENDIF

mov bx,onGround

    .IF characterPosition.Y<bx              ;若不在地上則下墜

      .IF jumping!=0                        ;判斷是否在跳躍過程

        .IF jumping<=7                      ;跳躍過程1到7每次向上1格

          inc jumping

          dec characterPosition.Y

        .ENDIF

        .IF jumping>7

          mov jumping,0                     ;跳躍過程結束歸零

        .ENDIF

      .ENDIF

      .IF jumping==0

        inc characterPosition.Y

      .ENDIF

    .ENDIF

1. 功能方塊
   * 共通：在每次進行迴圈過後，會將每個陣列的值往前挪動，以此來呈現角色移動的感覺。

INVOKE enemyMove                           ;判斷是否有舊的敵人並向前移動

INVOKE springMove                          ;判斷是否有舊的彈簧並向前移動

INVOKE accelerateMove                      ;判斷是否有舊的加速板並向前移動

INVOKE coinMove                            ;判斷是否有舊的金幣並向前移動

enemyMove PROC USES eax ecx esi             ;每一次重畫就判斷敵人移動

    mov esi,0

    mov ecx,119

  ENEMYLEFT:                                ;敵人陣列全部往前複製

    mov al,[enemyRow+esi+1]

    mov [enemyRow+esi],al

    mov ax,[enemyHeight+esi+1]

    mov [enemyHeight+esi],ax

    inc esi

    LOOP ENEMYLEFT

    mov esi,119                             ;敵人陣列最後一個補0

    mov [enemyRow+esi],0

    mov [enemyHeight+esi],0

    ret

    enemyMove ENDP

springMove PROC USES eax ecx esi             ;每一次重畫就判斷彈簧移動

    mov esi,0

    mov ecx,119

  SPRINGLEFT:                                ;彈簧陣列全部往前複製

    mov al,[springRow+esi+1]

    mov [springRow+esi],al

    inc esi

    LOOP SPRINGLEFT

    mov esi,119                             ;彈簧陣列最後一個補0

    mov [springRow+esi],0

    ret

    springMove ENDP

accelerateMove PROC USES eax ecx esi             ;每一次清除版面重畫就判斷加速板移動

    mov esi,0

    mov ecx,119

  ACCELERATELEFT:                                ;加速板陣列全部往前複製

    mov al,[accelerateRow+esi+1]

    mov [accelerateRow+esi],al

    mov ax,[accelerateHeight+esi+1]

    mov [accelerateHeight+esi],ax

    inc esi

    LOOP ACCELERATELEFT

    mov esi,119                             ;加速板陣列最後一個補0

    mov [accelerateRow+esi],0

    mov [accelerateHeight+esi],0

    ret

    accelerateMove ENDP

coinMove PROC USES eax ecx esi             ;每一次清除版面重畫就判斷金幣移動

    mov esi,0

    mov ecx,119

  COINLEFT:                                ;金幣陣列全部往前複製

    mov al,[coinRow+esi+1]

    mov [coinRow+esi],al

    mov ax,[coinHeight+esi+1]

    mov [coinHeight+esi],ax

    inc esi

    LOOP COINLEFT

    mov esi,119                             ;金幣陣列最後一個補0

    mov [coinRow+esi],0

    mov [coinHeight+esi],0

    ret

    coinMove ENDP

* + 障礙物：我們使用一個陣列判斷是否有敵人在那格，並且記錄高度，利用與地板距離判斷印出對應高度的敵人，與主角座標相同時則會進入結束遊戲的函式中，並離開迴圈。

enemyCreate PROC USES eax ebx ecx esi               ;判斷敵人是否生成

    mov ebx,enemyProbability                    ;增加機率

    inc ebx

    mov enemyProbability,ebx

    mov eax,enemyProbability                    ;機率生成敵人

    .IF eax>enemy

      mov esi,119                             ;用陣列存位置

      mov [enemyRow+esi],1

    .ENDIF

    .IF eax>enemy

      mov esi,119                             ;用陣列存高度

      mov eax,height

      mov [enemyHeight+esi],ax

    .ENDIF

    ret

    enemyCreate ENDP

gameOver PROC USES eax ebx ecx esi             ;判斷遊戲結束

    movzx esi,characterPosition.X              ;如果當前X座標對應到敵人陣列中不是1就沒事

    .IF [enemyRow+esi]==1

      mov ax,ground                               ;如果當前Y座標不是地板上就沒事

      sub ax,[enemyHeight+esi]

      mov bx,characterPosition.Y

      .IF ax<=bx && bx<=onGround

        mov gameoverCheck,1

      .ENDIF

    .ENDIF

    ret

    gameOver ENDP

* + 彈簧：我們使用一個陣列判斷是否有彈簧在那格，並且生成在地上，與主角座標相同時則會進入開始彈跳的函式中，在5ms的延遲下連續向上7格(與跳躍高度相同)，期間內不改變其他物體位置，以此來模擬向上彈起的感覺。

springCreate PROC USES eax ebx ecx esi               ;判斷彈簧是否生成

    mov eax,springProbability

    mov esi,119

    .IF eax>spring && [enemyRow+esi]==0             ;機率生成彈簧

      mov [springRow+esi],1

    .ENDIF

    ret

    springCreate ENDP

springDetect PROC USES eax ebx ecx esi             ;判斷彈簧

    movzx esi,characterPosition.X              ;如果當前X座標對應到彈簧陣列中不是1就沒事

    .IF [springRow+esi]==1

      mov ax,onGround

      mov bx,characterPosition.Y

      .IF ax==bx

      mov ecx,7

  SPRINGOVER:

        mov eax,5                           ;延遲

        call Delay

        dec characterPosition.y

        LOOP SPRINGOVER

      .ENDIF

    .ENDIF

    ret

    springDetect ENDP

* + 加速板：我們使用一個陣列判斷是否有加速板在那格，並且記錄高度，利用與地板距離判斷印出單獨的加速板(可懸空)，與主角座標相同時則會進入加速的函式中，在1ms的延遲下連續向前10格，期間內會呼叫與物體生成和移動的函式但不處理碰撞，除了金幣，達到類似傳送概念的加速。

accelerateCreate PROC USES eax ebx ecx esi             ;判斷加速板是否生成

    mov eax,accelerateProbability

    mov esi,119

    .IF eax>accelerate && [enemyRow+esi]==0 && [springRow+esi]==0           ;機率生成加速板

      mov [accelerateRow+esi],1

    .ENDIF

    .IF eax>accelerate && [enemyRow+esi]==0 && [springRow+esi]==0

      mov esi,119                             ;用陣列存高度

      mov eax,aheight

      mov [accelerateHeight+esi],ax

    .ENDIF

    ret

    accelerateCreate ENDP

accelerateDetect PROC USES eax ebx ecx esi             ;判斷加速板

    movzx esi,characterPosition.X              ;如果當前X座標對應到加速板陣列中不是1就沒事

    .IF [accelerateRow+esi]==1

      mov ax,ground                               ;如果當前Y座標不是地板-高度就沒事

      sub ax,[accelerateHeight+esi]

      mov bx,characterPosition.Y

      .IF ax==bx

      mov kingKrim,10

  ACCERLERATEOVER:

      mov eax,1000000                            ;產生敵人變數

      call RandomRange

      mov enemy,eax

      mov eax,3                                 ;產生敵人高度變數

      call RandomRange

      inc eax

      mov height,eax

      mov eax,1000000                            ;產生彈簧變數

      call RandomRange

      mov spring,eax

      mov eax,1000000                            ;產生彈簧變數

      call RandomRange

      mov accelerate,eax

      mov eax,3                                 ;產生加速板高度變數

      call RandomRange

      inc eax

      mov aheight,eax

      mov eax,1000000                            ;產生硬幣變數

      call RandomRange

      mov coin,eax

      mov eax,3                                 ;產生硬幣高度變數

      call RandomRange

      inc eax

      mov cheight,eax

      INVOKE enemyMove                           ;判斷是否有舊的敵人並向前移動

      INVOKE springMove                          ;判斷是否有舊的彈簧並向前移動

      INVOKE accelerateMove                      ;判斷是否有舊的加速板並向前移動

      INVOKE coinMove                            ;判斷是否有舊的金幣並向前移動

      INVOKE coinDetect                          ;判斷是否撞上金幣

      INVOKE enemyCreate                         ;判斷敵人生成

      INVOKE springCreate                        ;判斷彈簧生成

      INVOKE accelerateCreate                    ;判斷加速板生成

      INVOKE coinCreate                          ;判斷金幣生成

      INVOKE consoleChange                       ;畫出畫面

      inc score

      mov eax,1                           ;延遲

      call Delay

      dec kingKrim

      cmp kingKrim,0

      jne ACCERLERATEOVER

      .ENDIF

    .ENDIF

    ret

    accelerateDetect ENDP

* + 金幣：我們使用一個陣列判斷是否有金幣在那格，並且記錄高度，利用與地板距離判斷印出單獨的金幣(可懸空)，與主角座標相同時則會進入拾起金幣的函式中，金幣陣列中的紀錄會消去並將分數增加10分。

coinCreate PROC USES eax ebx ecx esi               ;判斷金幣是否生成

    mov eax,coinProbability

    mov esi,119

    .IF eax>coin && [enemyRow+esi]==0 && [springRow+esi]==0 && [accelerateRow+esi]==0        ;機率生成金幣

      mov [coinRow+esi],1

    .ENDIF

    .IF eax>coin && [enemyRow+esi]==0 && [springRow+esi]==0 && [accelerateRow+esi]==0

      mov esi,119                             ;用陣列存高度

      mov eax,cheight

      mov [coinHeight+esi],ax

    .ENDIF

    ret

    coinCreate ENDP

coinDetect PROC USES eax ebx ecx esi             ;判斷金幣

    movzx esi,characterPosition.X              ;如果當前X座標對應到金幣陣列中不是1就沒事

    .IF [coinRow+esi]==1

      mov ax,ground                               ;如果當前Y座標不是地板-高度就沒事

      sub ax,[coinHeight+esi]

      mov bx,characterPosition.Y

      .IF ax==bx

        add score,10

        mov [coinRow+esi],0

      .ENDIF

    .ENDIF

    ret

    coinDetect ENDP

1. 遊戲運作機制
   * 使用delay的延遲與無窮迴圈來重複進行判定，那一開始設定的延遲是50ms/格，這是經過多次測試最適合進行此遊戲的速度。
   * 分數的計算就是依照玩家所走的格數和金幣計算，也就是(迴圈執行的次數+金幣數量\*10)。
   * 為了增加遊戲難度，我們用(原本的延遲-分數/64)來增快移動速度，使得分數上升到1000以上之後就會不那麼容易存活。

.IF eax>=delayTime

      mov delayTime,10

      jmp DelayEDIT

    .ENDIF

    .IF eax<delayTime

      mov ebx,delayTime

      sub ebx,eax

    .ENDIF

  DelayEDIT:

    mov eax,ebx                           ;延遲

    call Delay

    inc score

    jmp L1

1. 機率產生及機制
   * 在一開始初始化先重置變數後，在每次移動一格後判定各種物體的生成，每個物體都有獨立的機率生成，但是當兩個物體同時生成則有優先度順序，分別是障礙物>彈簧>加速板>金幣。
   * 障礙物有隨著分數變動的機率，一開始是1/100，分數增加一分則增加1/100000的機率，也就是說，當分數到達一千分時會增加一倍的敵人數量(詳細見enemyCreate函式)
   * 其他物體的生成機率為，彈簧1/100，加速板1/50，金幣1/20

mov eax,1000000                            ;產生敵人變數

    call RandomRange

    mov enemy,eax

    mov eax,3                                 ;產生敵人高度變數

    call RandomRange

    inc eax

    mov height,eax

    mov eax,1000000                            ;產生彈簧變數

    call RandomRange

    mov spring,eax

    mov eax,1000000                            ;產生加速板變數

    call RandomRange

    mov accelerate,eax

    mov eax,3                                 ;產生加速板高度變數

    call RandomRange

    inc eax

    mov aheight,eax

    mov eax,1000000                            ;產生金幣變數

    call RandomRange

    mov coin,eax

    mov eax,3                                 ;產生金幣高度變數

    call RandomRange

    inc eax

    mov cheight,eax

心得感想

由於我們組別demo時間較早且只有兩個人完成，且對於一些內建函式的內容不甚了解的情況下，在撰寫方面常常遇到瓶頸，即便兩個人一起討論常常也花上不少時間才能夠解決，尤其是readkey那個部分，是上網查詢不少資料才實現出來，算是前進一大步，另一方面，實習課開始教學與demo相關的東西時間有點晚，導致一開始不知道要怎麼下手，不論是handle的取得，或是其他種類的功能，許多函式因為網路上的組合語言太過多元，所以很難查詢，而在許多迴圈函式運算的部分，也是經過不斷的思考與修正，甚至不乏有整個函式重寫的狀況，甚至在demo時並未完全完成，經過更多時間的修正下，我們終於改進了許多程式碼，並且做出了排行榜的部分，我認為這部分是非常獨特的功能，因為他涉及很困難的檔案讀寫，最後終於在繳交書面報告前完成，我們認為組合語言真的是非常神奇的領域，不僅很多原本在高階語言認為理所當然的一些功能都沒辦法簡單且直覺地完成，讓我們更加了解電腦實際上運作的一些原理，不管是所謂的暫存器或是記憶體的控制，一些獨特的語法更讓我們彷彿遇到一個又一個的新世界，這學期的組合語言時在讓我們非常有收穫，也感謝老師和助教上課時的提點讓我們能夠做出這個有趣的小遊戲。