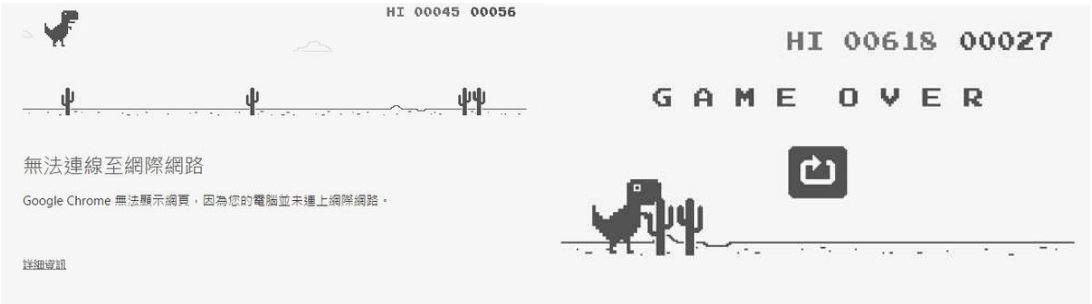
組合語言與系統程式期末專題

第 15 組，102403015 程祥恩、102403016 邱威穎、102403020 曾子軒

一、 主題

我們製作一款測驗反應的遊戲，主要靈感來自 Google Chrome 的離線恐龍遊戲，該恐龍遊戲的目標是控制恐龍，並利用空白鍵使恐龍跳躍來躲避障礙物並取得高分，一旦碰到障礙物及遊戲結束，如下列圖片所示。



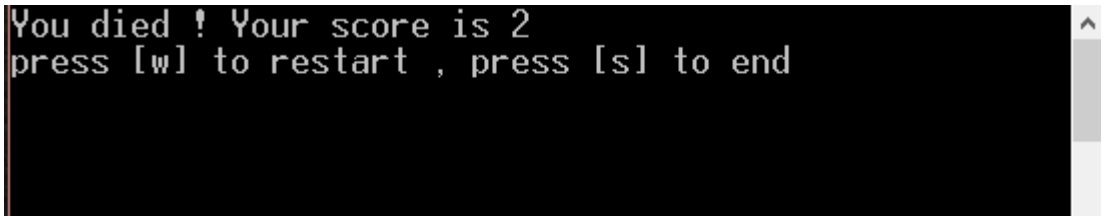
而我們這次以指標取代恐龍，以高低不定的牆壁來取代恐龍遊戲中的仙人掌，規則方面同樣是要透過跳躍來閃避障礙物，同樣是碰到牆壁及死去。不同的是，我們的遊戲畫面一次只會出現一個障礙物，每躲過一個障礙物就會加一分，，由於游標移動十分快速，要連續閃躲障礙物並不容易，所以這是一款極度考驗玩家之反應力的遊戲。

二、 流程

1. 進入遊戲後會看到歡迎文字，接著點選 w 可以開始遊戲

2. 開始遊戲後，玩家扮演的就是左邊的白色矩形，他會快速地向右移動，而玩家的目標是要透過跳躍躲開隨機出現的障礙物，紅色圈選的地方則為玩家的分數。(共躲過了幾個障礙物)

3. 碰到障礙物後，即結束遊戲



三、　程式碼：

1.　變數宣告

```
5   BoxWidth = 45
6   border = 44
7   sleepTime = 15
8
9   .data
10  consoleHandle    DWORD ?
11  xyInit COORD <0,19> ; 起始座標
12  xyBound COORD <BoxWidth,25> ; 一個頁面最大的邊界
13  xyPos COORD <?,?> ; 現在的游標位置
14
15  floor BYTE BoxWidth DUP(0C4h)
16  outputHandle DWORD 0
17  bytesWritten DWORD 0
18  count DWORD 0
19  floor_xyPosition COORD <0,20>
20  cellsWritten DWORD ?
21  attributes0 WORD BoxWidth DUP(0Bh)
22  wall BYTE 0DAh
23  xyPosWall COORD <20,20>
24  attributeWall WORD 0Bh
25  heightWall WORD ?
26  gameStartMsg BYTE "Welcome ! Press [w] to get start", 0
27  gameOverMsg1 BYTE "You died ! Your score is ", 0
28  gameOverMsg2 BYTE "press [w] to restart , press [s] to end", 0
29  isOver BYTE 0
30  score DWORD 0
```

這邊宣告了一些會用到的變數，包含指標的移動速度(sleepTime)、地圖的長度(BoxWidth)等，因為已經將其從程式中用變數代替，因此欲修改遊戲關卡屬性時可以直接設定這些變數的值，其他還宣告了用來記錄座標的 COORD 型態資料、要畫出地板與牆壁的內容、顏色以及在開始或結束時的提示字串。

2. 產生牆壁

```
createWall PROC uses eax ecx

    call Randomize
    mov eax,20
    call RandomRange
    add eax,25
    mov xyPosWall.x,ax
    mov eax,4
    call RandomRange
    add eax, 1
    mov heightWall, ax
    mov xyPosWall.y,20
    sub xyPosWall.y,1
    mov ecx,eax

L1:
    push ecx
    push eax
    INVOKE WriteConsoleOutputAttribute,
            consoleHandle,
            ADDR attributeWall,
            SIZEOF wall,
            xyPosWall,
            ADDR count

    INVOKE WriteConsoleOutputCharacter,
             consoleHandle,
             ADDR wall,
             SIZEOF wall,
             xyPosWall,
             ADDR count

    dec xyPosWall.Y
    pop eax
    pop ecx
    loop L1
    ret
createWall ENDP
```

先呼叫 Randomize 表示要使用隨機函數，接著呼叫 RandomRange 會抓取 eax 的值當範圍並隨機產生一數後放回 eax，因此先將 eax 設成 20 代表會隨機產生 0~19 的數，再將這個數變大一點當成牆壁的位置，因此牆壁都只會隨機出現在 25~44 也就是中間到右邊的位置。再用同樣的方法隨機產生牆壁的高度 1~4 層，最後便可用迴圈印出牆壁：WriteConsoleOutputAttribute 是印出顏色 WriteConsoleOutputCharacter 是印出字元。

3. 印出底線

```
createLine PROC uses eax ecx

    INVOKE WriteConsoleOutputAttribute,
      consoleHandle,
      ADDR attributes0,
      SIZEOF floor,
      floor_xyPosition,
      ADDR count

    INVOKE WriteConsoleOutputCharacter,
       consoleHandle,       ; console output handles
       ADDR floor,          ; pointer to the top box line
       SIZEOF floor,        ; size of box line
       floor_xyPosition,    ; coordinates of first char
       ADDR count           ; output count

    ret
createLine ENDP
```

同理，透過 WriteConsoleOutputCharacter 印出地板、顏色則是 WriteConsoleOutputAttribute。

4. 向上跳躍

```
JumpUp PROC

    sub xyPos.y,1 ;jump
        add xyPos.x,1
        mov ax, xyPos.x
        mov bx, xyPos.y
        mov dx, xyPosWall.y
        inc dx

        .IF ax == xyPosWall.x       ;判斷撞牆
            push ecx
            movzx ecx, heightWall
            isCollidedJump:
                .IF bx == dx
                    pop ecx
                    je END_FUNC2
                .ENDIF
                inc dx
                LOOP isCollidedJump
            pop ecx
        .ENDIF                      ;判斷撞牆

        .IF xyPos.x == border       ;跑完一圈
            sub xyPos.x, border
            call ClrScr
            call getScore
            call createLine
            call createWall         ;跑完一圈
        .ENDIF

        push ecx
        INVOKE Sleep, sleepTime
        INVOKE SetConsoleCursorPosition, consoleHandle, xyPos
        pop ecx
        ret

        END_FUNC2:                          ;撞到牆壁的處理
            push eax
            and eax , 00000000
            inc eax
            mov isOver , al
            pop eax
            ret

JumpUp ENDP
```

一旦使用者按跳躍，就會存取到這個方法，讓游標一步一步的往右上走，當然在這途中，程式也會不斷地判斷游標是否有撞到牆壁，若有撞擊，就會 return 回 main Function 並結束遊戲。若跳躍到一半碰到了螢幕邊界，程式也會進行判斷，讓游標回到適當的位置。當然，在刷新螢幕的同時，也會重新產生全新的地板、分數以及牆壁。

5. 向下降落

```
FallDown PROC
    add xyPos.y,1 ;fallDown
        add xyPos.x,1
        mov ax, xyPos.x
        mov bx, xyPos.y
        mov dx, xyPosWall.y
        inc dx

        .IF ax == xyPosWall.x       ;判斷撞牆
            push ecx
            movzx ecx, heightWall
            isCollidedFallDown:
                .IF bx == dx
                    pop ecx
                    je END_FUNC3
                .ENDIF
                inc dx
                LOOP isCollidedFallDown
            pop ecx
        .ENDIF                      ;判斷撞牆

        .IF xyPos.x == border       ;跑完一圈
            sub xyPos.x, border
            call ClrScr
            call getScore
            call createLine
            call createWall
        .ENDIF                      ;跑完一圈

        push ecx
        INVOKE Sleep, sleepTime
        INVOKE SetConsoleCursorPosition, consoleHandle, xyPos
        pop ecx
        ret

        END_FUNC3:                          ;撞到牆壁的處理
            push eax
            and eax , 00000000
            inc eax
            mov isOver , al
            pop eax
            ret


FallDown ENDP
```

降落同理，和向上跳躍邏輯相同，只是把 y 軸從原本的遞減改為遞增，這邊同樣也會隨時判斷游標有沒有撞擊到牆壁，刷新螢幕的功能也有完整的實作在這裡。

6. 跳躍整合

```
183 playerJump PROC uses ecx eax ebx
184     mov ecx, 6
185     jump:                                   ;繼續往上跳
186         call JumpUp
187         mov al , isOver
188         .IF al == 1
189             jmp RET_FUNC
190         .ENDIF
191         LOOP jump                           ;繼續往上跳
192
193     mov ecx, 6
194     fall:                                   ;繼續下墜
195         call FallDown
196         mov al , isOver
197         .IF al == 1
198             jmp RET_FUNC
199         .ENDIF
200         LOOP fall                           ;繼續下墜
201     RET_FUNC:
202     ret
203
204 playerJump ENDP
205
206 getScore PROC uses eax
207     mov eax, score
208     call WriteDec
209     inc eax
210     mov score, eax
211     ret
212 getScore ENDP
```

當使用者在遊戲中按下 w(跳躍鍵)時，便會呼叫這個方法，首先會呼叫 JumpUp，完成從地板跳至最高點的動作，接著執行 188 行的判斷式，跳躍中途有碰到障礙物的話 isOver 就會被設成 1，判斷成立便直接 return 回去結束遊戲至結算畫面，若無則 isOver 仍為 0 會繼續跳躍的動作。

7. 主程式

```
214 main PROC
215
216 ; Get the Console standard output handle:
217     INVOKE GetStdHandle, STD_OUTPUT_HANDLE
218     mov consoleHandle,eax
219
220 ; 設定回到起始位置
221 GameActivated:
222     mov edx , OFFSET gameStartMsg
223     call ClrScr
224     call WriteString
225     call ReadChar
226     .IF ax == 1177h
227         jmp INITIAL
228     .ENDIF
229
230 INITIAL:
231     mov isOver , 0
232     mov score , 0
233     mov ax,xyInit.x
234     mov xyPos.x,ax
235     mov ax,xyInit.y
236     mov xyPos.y,ax
237     call ClrScr
238     call getScore
239     call createLine
240     call createWall
241
242 START:
243     ;push eax
244     INVOKE SetConsoleCursorPosition, consoleHandle, xyPos
245
246     call ReadKey
247
248     .IF ax == 1177h ;UP (w)
249         call playerJump
250
251         push eax
252         and eax,00000000
253         mov al,isOver
254         cmp al,1
255         je END_FUNC
256         pop eax
257
258     .ENDIF
259
```

主程式(續)

```
260     .IF ax == 011Bh ;ESC
261       jmp END_FUNC
262     .ENDIF
263
264     add xyPos.x,1
265     INVOKE Sleep, sleepTime
266
267     ; 檢查作完上下左右後有沒有超過限制邊界
268
269     mov ax,xyPosWall.x ; 註：比較不能用雙定址，故將其中一個轉成 register
270
271     .IF xyPos.x == ax
272         jmp END_FUNC
273     .ENDIF
274
275     mov ax,xyBound.x ; 註：比較不能用雙定址，故將其中一個轉成 register
276
277     .IF xyPos.x == border                       ;跑完一圈
278         sub xyPos.x, border
279         call ClrScr
280         call getScore
281         call createLine
282         call createWall
283     .ENDIF                                      ;跑完一圈
284
285     jmp START
286
287 END_FUNC:
288     call ClrScr
289     mov edx , OFFSET gameOverMsg1
290     call WriteString
291     mov eax , score
292     dec eax
293     call WriteDec
294     call Crlf
295     mov edx , OFFSET gameOverMsg2
296     call WriteString
297     INVOKE GetStdHandle, STD_OUTPUT_HANDLE
298     mov consoleHandle,eax
299     call ReadChar
300     .IF ax == 1177h
301         jmp INITIAL
302     .ENDIF
303     .IF ax == 1F73h
304         exit
305     .ENDIF
306
```

遊戲剛啟動時，會先執行 GameActivated LABEL，顯示歡迎字串並提醒使用者按下 w 鍵以開始遊戲，接著會初始化牆壁、地板、分數以及游標位置(位於地板的最左邊)，之後就會進入 START LABEL，一進去就會是一個無窮迴圈，每一圈都代表游標向前移動一步，而 sleepTime 就是控制移動速度的關鍵角色。在無窮迴圈的途中，我們也透過 ReadKey 隨時不間斷地監視使用者的鍵盤輸入，若輸入了 w，則呼叫 playerJump 進行跳躍，跳完之後會進入 251~256 行的判斷式，倘若在跳躍途中有撞到牆壁，拿來判斷是否有碰撞的變數便會被設定，這時程式會轉向 END_FUNC 至成績結算介面，此時按下 w 可以重新開始遊戲，按下 s 則會離開遊戲。若游標的位置超出範圍，則將游標的 X 座標減去螢幕邊界，讓游標回到最左邊，並更新分數、重新建立牆壁與地板。