書面報告 HIT THE SKULL

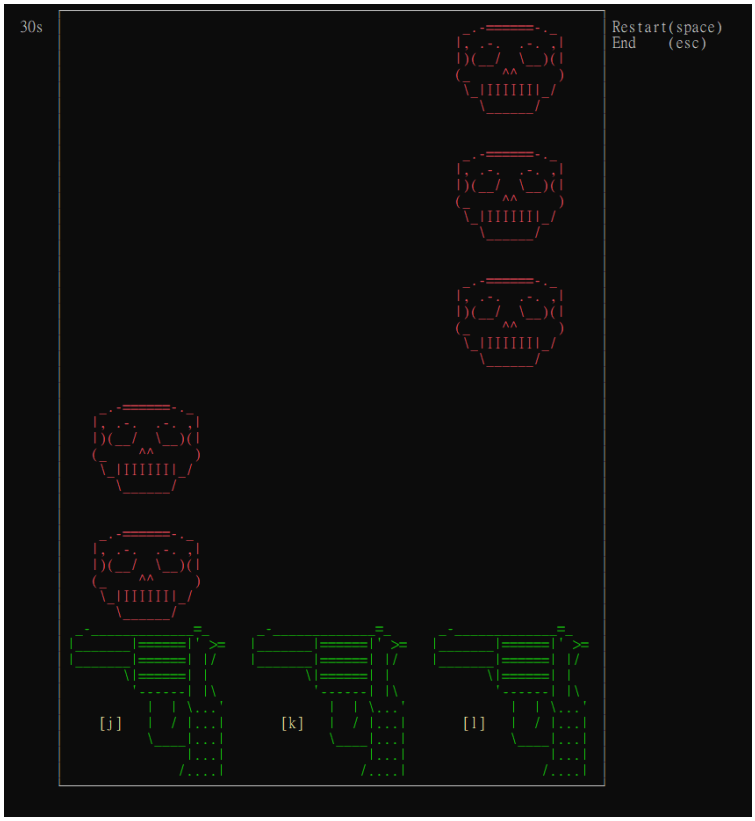
第 11 組

107201535 數學 4B 陳羽暉

107201023 數學 4B 蔡沐霖

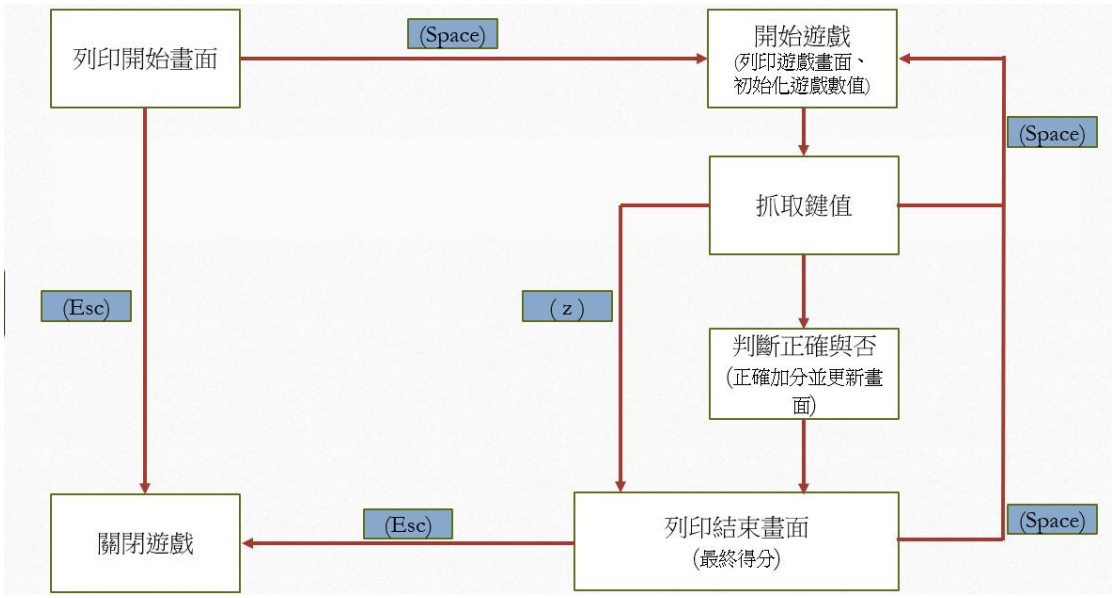
107201522 數學 4A 蘇柏瑜

遊戲規則：



按照骷髏頭的位置，選擇相對應的槍枝，分別是左(j)、中(k)、右(l)，選擇正確加分，錯誤不扣分。

程式架構：



程式說明：

main:

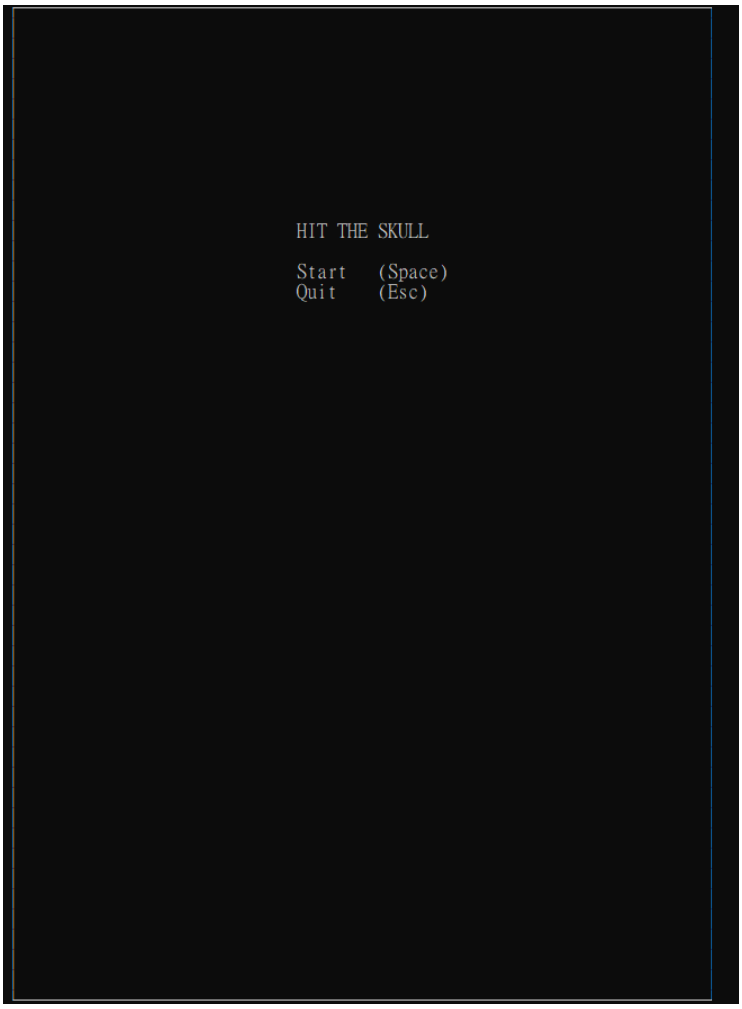
```
;開始畫面-------------------------------------------------------------------------------
START:
        ;印出畫面-------------------------------------------------------------------------
        invoke printScreen, ADDR startArray, screenIni
L5:
        call ReadChar
        .IF ax == 011Bh ;esc
                jmp gameEXIT                            ;按Esc,關閉遊戲
        .ENDIF
        .IF ax == 3920h ;space
                jmp GAME                                ;按Space,開始遊戲
        .ENDIF
        jmp L5
```

首先開始畫面是用 printScreen 這個函式把畫面印出來，開始畫面中需要印出邊界還有中間的事件選項(圖一)，然後依 ReadChar 所抓到的鍵值來判斷進入哪個事件。

按(Esc)，進入 gameEXIT，跳離遊戲，

按(Space)，進入 GAME，進行遊戲，

按其他任意鍵，則會跳回 L5 繼續判斷鍵值。



▲圖一

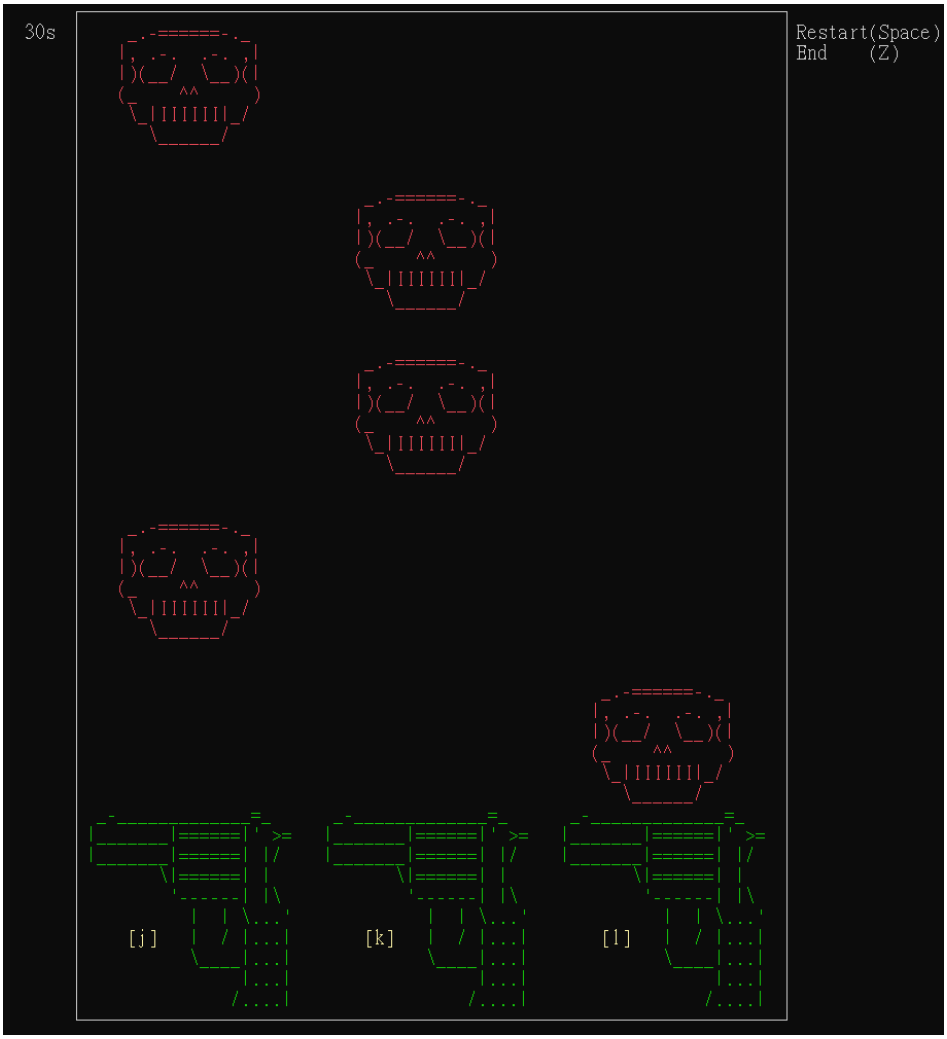
```
;遊戲畫面--------------------------------------------------------------------------------
GAME:
        ;初始化遊戲物件--------------------------------------------------------------------
        mov timeCountDown, 30
        mov loopCount, 0
        mov initialTime[0], 33h
        mov initialTime[1], 30h
        mov targetPosition.x, 0
        mov targetPosition.y, 23
        mov score,0
        mov killnumber,0
        mov scoreWeight,1

        ;印出畫面--------------------------------------------------------------------------
        invoke gameScreen, screenIni, screenIni
        invoke printPlayer, screenIni
        invoke initial_cube_set
        invoke printTarget
        invoke printTime

        ;進入迴圈--------------------------------------------------------------------------
LookForKey:
        ;判斷30秒是否結束------------------------------------------------------------------
        .IF timeCountDown == 0
                jmp gameEND
        .ENDIF
        ;判斷是否過一秒--------------------------------------------------------------------
        .IF loopCount == 20
                dec timeCountDown
                mov loopCount, 0
                .IF initialTime[1] == 30h
                        mov initialTime[1], 39h
                        dec initialTime[0]
                .ELSE
                        dec initialTime[1]
                .ENDIF
                invoke printTime
        .ENDIF
```

```
;判斷按鍵--------------------------------------------------------------------------
mov eax, 50
call Delay
inc loopCount
call ReadKey
jz LookForKey                                   ;沒抓到按鍵,進入下一迴圈
.IF al == 20h
        jmp GAME                                ;按Space,開始遊戲
.ENDIF
.IF al == 7Ah
        jmp gameEnd                             ;按z,結束遊戲
.ENDIF
.IF (al == 6Ah) && (cubePosition[0] == 00h)     ;按j且目標左,得分,更新目標位置
        inc killnumber
        invoke cube_renew
        mov targetPosition.x, 0
        mov targetPosition.y, 23
        invoke printTarget
.ENDIF
.IF (al == 6Bh) && (cubePosition[0] == 01h)     ;按k且目標中,得分,更新目標位置
        inc killnumber
        invoke cube_renew
        mov targetPosition.x, 0
        mov targetPosition.y, 23
        invoke printTarget
.ENDIF
.IF (al == 6Ch) && (cubePosition[0] == 02h)     ;按l且目標右,得分,更新目標位置
        inc killnumber
        invoke cube_renew
        mov targetPosition.x, 0
        mov targetPosition.y, 23
        invoke printTarget
.ENDIF

jmp LookForKey                                  ;進入下一迴圈
```

進入 GAME，會先初始化所有參數的數值，像是分數、時間、目標位置……等等，然後才依據這些參數印出遊戲畫面，如圖二。

進入 LookForKey，會先判斷剩餘秒數是否為 0，為 0 則跳到 gameEND，結算分數，不然就繼續執行。 判斷是否過一秒，過一秒則更新要印出的數字，判斷方式為看是否過 20 個迴圈，一迴圈為 50 個千分之一秒。 接下來用 ReadKey 抓取鍵值，但這邊必須配合 Delay，才能確保 ReadKey 能抓到值，Delay 是設為 50 個千分之一秒。 假如 Delay 完，ReadKey 還是沒抓到值，則跳回 LookForKey，迴圈數加一；假如 ReadKey 有抓到值，則依據抓取報的鍵值判斷事件。

按(Space)，跳回 game，重新開始遊戲，
按(z)，跳到 gameEND，結算分數，
按(j)且目標左，得分、更新目標位置，
按(k)且目標中，得分、更新目標位置，
按(l)且目標右，得分、更新目標位置。
然後跳回 LookForKey，進入下一迴圈。



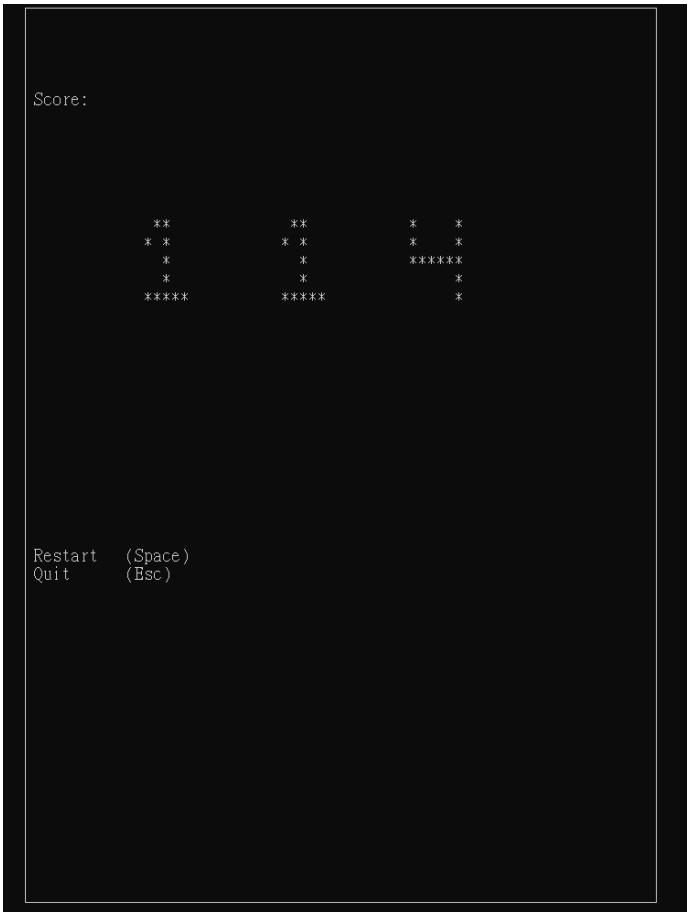
▲圖二

```
;結束畫面-------------------------------------------------------------------------------
gameEND:

        invoke computescore
        invoke endScreen, ADDR endArray, screenIni
        invoke printscore
L4:
        call ReadChar
        .IF ax == 011Bh ;esc
                jmp gameEXIT                            ;按Esc,關閉遊戲
        .ENDIF
        .IF ax == 3920h ;space                          ;按Space,開始遊戲
                jmp GAME
        .ENDIF
        jmp L4

;關閉遊戲-------------------------------------------------------------------------------
gameEXIT:
        call Clrscr
        exit


main ENDP
```

進入 gameEND，先印出結束畫面，包含邊界、分數以及事件選項(如圖三)，然後依 ReadChar 所抓到的鍵值來判斷進入哪個事件。

按(Esc)，進入 gameEXIT，跳離遊戲，

按(Space)，進入 GAME，進行遊戲，
按其他任意鍵，則會跳回 L5 繼續判斷鍵值。
進入 gameEXIT，清空畫面，終止程式。



▲圖三

Procedure:

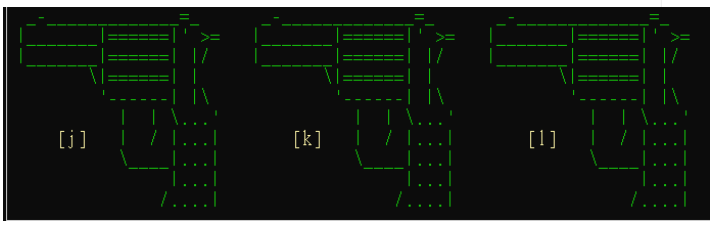
```
;-----------------------------------------------------
;printBox 列印遊戲邊界
;-----------------------------------------------------
printBox PROC USES eax ecx edi, screen:element
        mov screen.opHandle, 0
        mov screen.count, 0
        mov screen.position.x, 10
        mov screen.position.y, 0
        mov boxBodyR.x, 79
        mov boxBodyR.y, 1
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov screen.opHandle, eax
        call Clrscr
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                ADDR boxTop,
                70,
                screen.position,
                ADDR screen.count
        inc screen.position.y
        mov ecx, 48
L2:
        push ecx
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                ADDR boxBody,
                1,
                screen.position,
                ADDR screen.count
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                ADDR boxBody,
                1,
                boxBodyR,
                ADDR screen.count
        inc screen.position.y
        inc boxBodyR.y
        pop ecx
        loop L2
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                ADDR boxBottom,
                70,
                screen.position,
                ADDR screen.count
        inc screen.position.y
        ret
printBox ENDP
```

這裡就用到了 Irvine32.inc 函式庫中的 GetStdHandle、riteConsoleOutputCharacter 及我們設好的參數位置來印出邊界。

```
;-----------------------------------------------------
;printScreen 列印開始畫面
;-----------------------------------------------------
printScreen PROC USES eax ecx edi, array:PTR BYTE, screen:element
        invoke printBox, screenIni
        mov screen.opHandle, 0
        mov screen.count, 0
        mov screen.position.x, 31
        mov screen.position.y, 10
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov screen.opHandle, eax
        mov ecx, 5
        mov edi, array
L1:
        push ecx
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                edi,
                28,
                screen.position,
                ADDR screen.count
        pop ecx
        add edi, 28
        inc screen.position.y
        loop L1
        ret
printScreen ENDP

;-----------------------------------------------------
;endScreen 列印結束畫面
;-----------------------------------------------------
endScreen PROC USES eax ecx edi, array:PTR BYTE, screen:element
        invoke printBox, screenIni
        mov screen.opHandle, 0
        mov screen.count, 0
        mov screen.position.x, 12
        mov screen.position.y, 5
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov screen.opHandle, eax
        mov ecx, 27
        mov edi, array
L1:
        push ecx
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                edi,
                28,
                screen.position,
                ADDR screen.count
        pop ecx
        add edi, 28
        inc screen.position.y
        loop L1
        ret
endScreen ENDP

;-----------------------------------------------------
;gameScreen 列印遊戲畫面(右邊界)
;-----------------------------------------------------
gameScreen PROC USES eax ecx edi, screen1:element, screen2:element
        invoke printBox, screenIni
        mov screen2.opHandle, 0
        mov screen2.count, 0
        mov screen2.position.x, 81
        mov screen2.position.y, 1
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov screen1.opHandle, eax
        mov screen2.opHandle, eax
        mov edi, OFFSET right
        invoke WriteConsoleOutputCharacter,
                screen2.opHandle,
                edi,
                14,
                screen2.position,
                ADDR screen2.count
        inc screen2.position.y
        add edi, 14
        invoke WriteConsoleOutputCharacter,
                screen2.opHandle,
                edi,
                14,
                screen2.position,
                ADDR screen2.count
        ret
gameScreen ENDP
```

printScreen、endScreen 和 gameScreen 基本上架構都是一樣的，都是先用 printBox 畫出邊界後再印出各自的 array，不同的只有內容及位置。

```
;-----------------------------------------------------
;printPlayer 列印遊戲畫面(玩家)
;-----------------------------------------------------
printPlayer PROC USES eax ecx edi, screen:element
        mov screen.opHandle, 0
        mov screen.count, 0
        mov screen.position.x, 12
        mov screen.position.y, 39
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov screen.opHandle, eax
        mov ecx, 10
        mov edi,OFFSET player
        ;列印槍枝---------------------------------------
L1:
        push ecx
        mov ecx,3
L2:
        push ecx
        invoke WriteConsoleOutputAttribute,
                screen.opHandle,
                ADDR playerAttributes,
                20,
                screen.position,
                ADDR screen.count
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                edi,
                20,
                screen.position,
                ADDR screen.count
        pop ecx
        add screen.position.x, 23
        loop L2
        pop ecx
        add edi,20
        mov screen.position.x, 12
        inc screen.position.y
        loop L1
        mov screen.position.x, 16
        mov screen.position.y, 45
        mov discribe[1],6Ah
        mov ecx,3

        ;列印[j][k][l]---------------------------------
L3:
        push ecx
        invoke WriteConsoleOutputAttribute,
                screen.opHandle,
                ADDR discribeAttributes,
                3,
                screen.position,
                ADDR screen.count
        invoke WriteConsoleOutputCharacter,
                screen.opHandle,
                ADDR discribe,
                3,
                screen.position,
                ADDR screen.count
        pop ecx
        inc discribe[1]
        add screen.position.x, 23
        loop L3


        ret
printPlayer ENDP
```

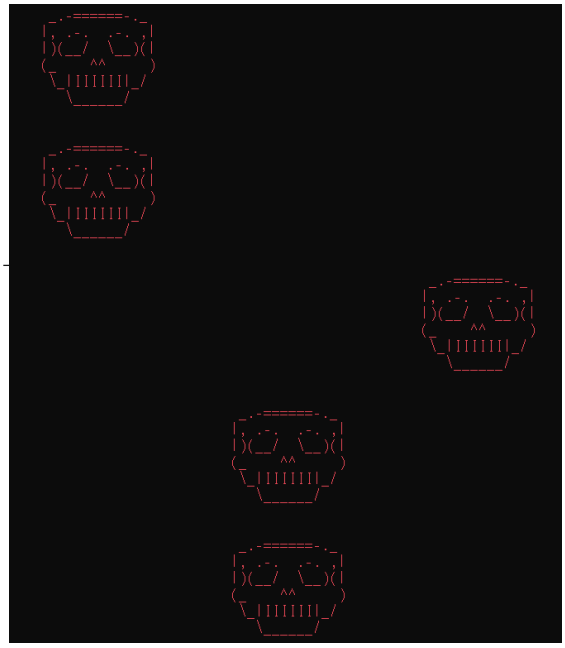


列印槍枝，並印出遊戲指示。

```
;-----------------------------------------------------
;cube 列印遊戲畫面(單一目標)
;-----------------------------------------------------
cube PROC USES eax ecx edi
        mov ecx, 6
        mov edi,OFFSET target
L:
        push ecx
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        invoke WriteConsoleOutputAttribute,
                eax,
                ADDR targetAttributes,
                14,
                targetPosition,
                ADDR screenIni.count
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        invoke WriteConsoleOutputCharacter,
                eax,
                edi,
                14,
                targetPosition,
                ADDR screenIni.count
        pop ecx
        inc targetPosition.y
        add edi,14
        loop L
        ret
cube ENDP
```

印出單一骷髏頭。

```
;-----------------------------------------------------
;printTime 列印遊戲畫面(所有目標)
;-----------------------------------------------------
printTarget PROC USES ecx edi
        mov ecx, 38
        mov coverPosition.x, 11
        mov coverPosition.y, 1
        ;清除畫面---------------------------------------
Cover:
        push ecx
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        invoke WriteConsoleOutputCharacter,
                eax,
                ADDR targetCover,
                68,
                coverPosition,
                ADDR screenIni.count
        inc coverPosition.y
        pop ecx
        loop Cover
        ;列印新目標-------------------------------
        mov targetPosition.y, 33
        mov edi, OFFSET cubePosition
        mov ecx, 5
L:
        .IF BYTE PTR [edi] == 0
                mov targetPosition.x, 15
        .ELSEIF BYTE PTR [edi] == 1
                mov targetPosition.x, 38
        .ELSE
                mov targetPosition.x, 61
        .ENDIF
        invoke cube
        inc edi
        sub targetPosition.y, 14
        loop L
        ret
printTarget ENDP
```



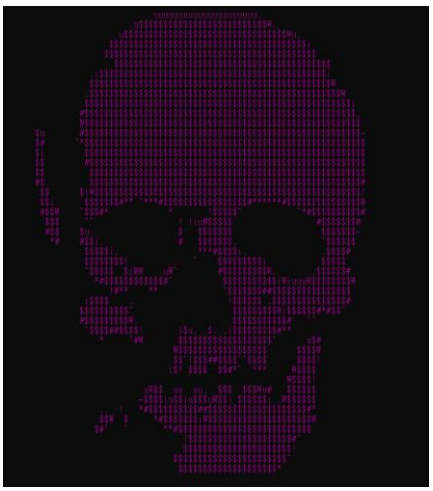
先將所有目前的骷髏頭覆蓋掉，再由我們所存的 targetPosition 來判斷骷髏頭應該印在左、中、右哪個位置，由最下面往上印。

```
;-----------------------------------------------------
;printTime 列印遊戲畫面(時間)
;-----------------------------------------------------
printTime PROC USES eax ecx
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        invoke WriteConsoleOutputCharacter,
                eax,
                ADDR initialTime,
                3,
                timer,
                ADDR screenIni.count
        ret
printTime ENDP
```

18s

28s

```
;-----------------------------------------------------
;printskull 列印骷髏頭
;-----------------------------------------------------
printskull PROC USES eax ecx edi
        mov ecx, 47
        mov edi,OFFSET skull
L:
        push ecx
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        invoke WriteConsoleOutputAttribute,
                eax,
                ADDR skullAttributes,
                67,
                skullPosition,
                ADDR screenIni.count
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        invoke WriteConsoleOutputCharacter,
                eax,
                edi,
                67,
                skullPosition,
                ADDR screenIni.count
        pop ecx
        inc skullPosition.y
        add edi,67
        loop L
        ret
printskull ENDP
```



```
;-----------------------------------------------------
;random_cube_set 隨機選擇左中右
;-----------------------------------------------------
random_cube_set PROC
        mov eax,3
        call RandomRange
        mov BYTE PTR [edi],al
        ret
random_cube_set ENDP
```

先用 RandomRangen 隨機選擇 0、1、2，以此來代表左中右三個位置，並將之迴船到目前 edi 所存的位置。

```
;-----------------------------------------------------
;initial_cube_set 初始化目標位置
;-----------------------------------------------------
initial_cube_set PROC USES eax ecx edi
        mov ecx,5
        mov edi,OFFSET cubePosition
cube_set:
        invoke random_cube_set
        inc edi
        loop cube_set
        ret
initial_cube_set ENDP
```

在進入 GAME 時呼叫，用五次迴圈呼叫 random_cube_set，隨機決定一開始五層骷髏頭左中右的位置

```
;-----------------------------------------------------
;cube_renew 更新目標位置
;-----------------------------------------------------
cube_renew PROC USES eax ecx esi edi
        mov     ecx,4
        mov esi, OFFSET cubePosition
        inc esi
        mov edi, OFFSET cubePosition

        cld               ;clear direction flag
        rep movsb            ;do the move
        invoke random_cube_set
        ret
cube_renew ENDP
```

用 movsb 先將存放骷髏頭位置的 array 更新，使這個 array 保留後四項，並往前推一格，而第五項則是呼叫 random_cube_set 來決定新的骷髏頭位置。

```
;-----------------------------------------------------
;computescore   計算分數
;-----------------------------------------------------
computescore PROC USES eax

L:
        .IF killnumber < 33
                xor eax,eax
                mov al,scoreWeight
                mul killnumber
                add score,ax
                ret
        .ENDIF
        xor eax,eax
        mov al,scoreWeight
        mul killcount
        add score,ax
        sub killnumber,33
        inc scoreWeight
        jmp L
computescore ENDP
```

在遊戲中，每射死一個殭屍，killnumber 就會加一。

在這裡 killnumber 是否小於 33，若否，則將 33*scoreweight 加入分數，且 scoreweight 加一；則將剩餘 killnumber*scoreweight 加入分數，並回傳。

```
;-----------------------------------------------------
;scorechoose 選擇分數
;-----------------------------------------------------
scorechoose PROC USES eax edi

        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov outputHandle,eax
        mov ecx,5
        .IF bl == 0
                mov edi,OFFSET score0
L0:
                push ecx
                invoke WriteConsoleOutputCharacter,
                        outputHandle,
                        edi,
                        10,
                        scorePosition,
                        ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L0
        .ENDIF
        .IF bl == 1
                mov edi,OFFSET score1
L1:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L1
        .ENDIF
        .IF bl == 2
                mov edi,OFFSET score2
L2:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L2
        .ENDIF
        .IF bl == 3
                mov edi,OFFSET score3
L3:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L3
        .ENDIF
        .IF bl == 4
                mov edi,OFFSET score4
L4:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L4
        .ENDIF
        .IF bl == 5
                mov edi,OFFSET score5
```

```
L5:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L5
        .ENDIF
        .IF bl == 6
                mov edi,OFFSET score6
L6:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L6
        .ENDIF
        .IF bl == 7
                mov edi,OFFSET score7
L7:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L7
        .ENDIF
        .IF bl == 8
                mov edi.OFFSET score8
```

```
L8:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L8
        .ENDIF
        .IF bl == 9
                mov edi,OFFSET score9
L9:
                push ecx
                invoke WriteConsoleOutputCharacter,
                outputHandle,
                edi,
                10,
                scorePosition,
                ADDR screenIni.count
                pop ecx
                inc scorePosition.y
                add edi,10
                loop L9
        .ENDIF
        mov scorePosition.y,12
        ret
scorechoose ENDP
```

這是依據 bl 中的值來決定要印出的數字 array 並將其印出。

```
;-------------------------------------------------------
;printscore 列印分數
;-------------------------------------------------------
printscore PROC USES eax ebx

        mov scorePosition.x,51
        mov scorePosition.y,12
        xor eax,eax
        .IF score < 100
                jmp L
        .ENDIF
        mov ax,score
        div scoredivisor
        mov bl,ah
        invoke scorechoose
        sub scorePosition.x,15



        mov ah,0
        mov score,ax
L:
        mov ax,score
        div scoredivisor
        mov bl,ah
        invoke scorechoose
        sub scorePosition.x,15




        mov bl,al
        invoke scorechoose

        ret
printscore ENDP
```

先用 div，將 score 除以 scoredivisor(=10)，將 ah(也就是目前 score 的個位數)存入 bl，在呼叫 scorechoose 印出相對應的分數 array。