國 立 台 灣 科 技 大 學

微 算 機 概 論 實 習

指 導 教 授: 陸敬互 教 授

微算機概論實習報告

(不是原創)

期末報告

班 級　：　電機二甲

學 生　：　黃偉智、韓承志

學 號　：　B11107035、B11107037

建檔日期:　2023/12/8

一、 學習成果(程式功能說明，說明程式分幾個部分，各部分在做甚麼，搭配截圖code說明)

1. 設定變數，當中包含球拍大小、球的大小、速度、邊界等各種參數、要顯示的字串等

```
.model small

.stack

.data
    window_height   dw 0c8h     ;視窗高度
    window_width    dw 140h     ;視窗寬度
    window_temp     dw 04h      ; used to make the window 'smaller' and making the b

    ball_start_xpos dw 0a0h     ;球體起始x座標
    ball_start_ypos dw 64h      ;球體起始y座標
    ball_color      db 0        ;球體顏色

    ball_xpos    dw 0a0h        ;球體此時x座標
    ball_ypos    dw 64h         ;球體此時y座標
    ball_size    dw 04h

    ball_speed_x dw 04h
    ball_speed_y dw 04h

    paddle1_xpos dw 0ah
    paddle1_ypos dw 0ah

    paddle2_xpos dw 130h
    paddle2_ypos dw 0ah

    paddle_width    dw 04h
    paddle_height   dw 16h
    paddle_border   dw 06h
    paddle_speed    dw 06h

    player_points   db 0
    ai_points       db 0

    player_points_mes   db 'Score: 0$'
```

```
menu_welcome_mes    db '------- WELCOME TO PONG! -------$'
menu_rule_mes       db 'Reach 10 points to win!$'
menu_choose_mes     db 'Choose your difficulty:$'
menu_easy_mes       db 'Easy   - Press 1$'
menu_medium_mes     db 'Medium - Press 2$'
menu_hard_mes       db 'Hard   - Press 3$'
menu_exit_mes       db 'Press ESC to exit$'
menu_border_mes     db '--------------------------------$'

game_over_mes       db '--------- GAME OVER ---------$'
game_win_mes        db 'YOU WON!$'
game_lose_mes       db 'YOU LOSE!$'
game_restart_mes    db 'Press R to play again$'
game_ends_mes       db 'Press E to exit to main menu$'
game_border_mes     db '-----------------------------$'

game_easy_mes   db '  Easy$'
game_medium_mes db 'Medium$'
game_hard_mes   db '  Hard$'

sys_time        db 0
activeness      db 1
winner          db 0    ; 1: Player wins, 2: AI wins
menu            db 0    ; 0: Main menu, 1: Game
difficulty      db 0    ; 1: Easy, 2: Medium, 3: Hard
ai_precision    dw 0
```

2. 進入code區

這個遊戲有三個難度可以選擇 easy、medium、hard，call_screen 用來清除畫面，不同難度會影響拍子大小和球的速度

```
.code
main proc
    mov ax, @data
    mov ds, ax

    call clear_screen
    check_difficulty:           ;檢查難度
        cmp difficulty, 01h
        je easy_difficulty

        cmp difficulty, 02h
        je medium_difficulty

        cmp difficulty, 03h
        je hard_difficulty
        jmp start_game

    easy_difficulty:
        mov ball_speed_x, 03h
        mov ball_speed_y, 03h
        mov paddle_height, 28h
        mov ai_precision, 14h
        jmp start_game

    medium_difficulty:
        mov ball_speed_x, 04h
        mov ball_speed_y, 04h
        mov paddle_height, 20h
        mov ai_precision, 10h
        jmp start_game

    hard_difficulty:
        mov ball_speed_x, 05h
        mov ball_speed_y, 05h
        mov paddle_height, 18h
        mov ai_precision, 0ch
        jmp start_game
```

3. 遊戲開始的迴圈，利用2ch讀取時間、sys_time存取上一次跑此迴圈的時間點，只要時間過去了就可以進行下一次的迴圈，dl是紀錄到毫秒的精準度，每次迴圈都會執行清除畫面、繪製球體、移動球體、繪製球拍、移動球拍、繪製玩家分數介面、快速的重複執行上述步驟以達到流暢的遊戲畫面

```
start_game:
    cmp menu, 0
    je display_start_menu

    cmp activeness, 0
    je draw_game_over

    mov ah, 2ch     ; get the system time
    int 21h

    cmp dl, sys_time    ; check if time has already passed
    je start_game       ; if time is still the same, then check again
                        ; if not, then move and draw the ball
    mov sys_time, dl    ; update time

    call clear_screen
    call draw_ball

    call move_ball
    call draw_paddles

    call move_paddles

    call draw_ui

    jmp start_game

    draw_game_over:
        call display_game_over
        jmp start_game

    display_start_menu:
        call display_main_menu
        jmp check_difficulty
```

4. 繪製球體:ball_color用來改變球的顏色，球體以小正方形表示

```
draw_ball proc
    mov cx, ball_xpos  ; set the x-position of ball
    mov dx, ball_ypos  ; set the y-position of ball

    draw_ball_xpos:
        mov ah, 0ch
        mov al, 01h     ; choose the pixel color for the ball
        add al, ball_color ;改變球的顏色
        mov bh, 00h
        int 10h

        inc cx          ; to the next pixel location
        mov ax, cx
        sub ax, ball_xpos
        cmp ax, ball_size   ; check if already reaches ball_size horizontally
        jng draw_ball_xpos     ; if not, then print pixel horizontally again

    draw_ball_ypos:
        mov cx, ball_xpos
        inc dx          ; go to the next line

        mov ax, dx
        sub ax, ball_ypos
        cmp ax, ball_size   ; check if alrady reaches ball_size vertically
        jng draw_ball_xpos     ; if not, then print pixel horizontally (to the next line)
        ret

draw_ball endp
```

5. 繪製球拍:paddle1是玩家操控的，paddle2是電腦

```
draw_ball endp

draw_paddles proc
    mov cx, paddle1_xpos
    mov dx, paddle1_ypos

    draw_paddle1_horizontal:
        mov ah, 0ch
        mov al, 0fh
        mov bh, 0
        int 10h

        inc cx
        mov ax, cx
        sub ax, paddle1_xpos
        cmp ax, paddle_width
        jng draw_paddle1_horizontal

    draw_paddle1_vertical:
        mov cx, paddle1_xpos
        inc dx

        mov ax, dx
        sub ax, paddle1_ypos
        cmp ax, paddle_height
        jng draw_paddle1_horizontal

        mov cx, paddle2_xpos
        mov dx, paddle2_ypos
```

```
    draw_paddle2_horizontal:
        mov ah, 0ch
        mov al, 0fh
        mov bh, 0
        int 10h

        inc cx
        mov ax, cx
        sub ax, paddle2_xpos
        cmp ax, paddle_width
        jng draw_paddle2_horizontal

    draw_paddle2_vertical:
        mov cx, paddle2_xpos
        inc dx

        mov ax, dx
        sub ax, paddle2_ypos
        cmp ax, paddle_height
        jng draw_paddle2_horizontal

    ret
draw_paddles endp
```

6. 清除畫面:利用不同模式間的切換以達到清除畫面的效果

```
clear_screen proc   ; 清除畫面  by restarting the video mode
    mov ah, 0
    mov al, 13h
    int 10h

    mov ah, 0bh
    mov bh, 0
    mov bl, 0      ; background color
    int 10h

    ret
clear_screen endp
```

7. 球體的移動:分為垂直跟水平、將球體座標加上速度值(如同位移量)

使重新繪製時有球移動的效果，如果讓球到達左邊的邊界(玩家沒成功回擊球)比賽就結束了，進入game_over 存取勝利者、及停止遊玩狀態

```
move_ball proc
    cmp player_points, 0ah  ; if the player reaches 10 points, they win
    jge game_over

    mov ax, ball_speed_x
    add ball_xpos, ax       ; move the ball horizontally  加上位移量

    mov ax, window_temp
    cmp ball_xpos, ax
    jl game_over       ; if ball_xpos < 0 then game over
    jmp move_ball_vertically

    game_over:

        mov ball_color, 0       ;重置顏色
        cmp player_points, 0ah
        jnl player_wins
        jmp ai_wins

        player_wins:
            mov winner, 01h   ;存取勝利者
            jmp continue_move_ball

        ai_wins:
            mov winner, 02h
            jmp continue_move_ball

        continue_move_ball:
            mov activeness, 0           ; stops the game
            mov player_points, 0
            ret
```

8. 球體的水平移動，判斷是否有撞擊到頂端跟底端，若有的話要停住球拍不讓他繼續往上，並且將垂直方向的速度反向，如同碰撞後的反彈

```
move_ball_vertically:
    mov ax, ball_speed_y
    add ball_ypos, ax       ; move the ball vertically

    mov ax, window_temp
    cmp ball_ypos, ax
    jl reset_position_y     ; if ball_ypos < 0 then ball collides with bottom border

    mov ax, window_height
    sub ax, ball_size
    sub ax, window_temp
    cmp ball_ypos, ax
    jg reset_position_y     ; if ball_ypos > 0 then ball collides with top border
    jmp check_collision_paddle2

reset_position_x:
    call reset_ball
    ret

reset_position_y:
    neg ball_speed_y
    ret
```

9. 球拍碰撞(右側、電腦):判斷是否狀要滿足四個條件:
   1) 球的最大x軸座標 ”大於” 球拍最小的x座標
   2) 球的最小x軸座標 ”小於” 球拍最大的x座標
   3) 球的最大y軸座標 ”大於” 球拍最小的y座標
   4) 球的最小y軸座標 ”小於” 球拍最大的y座標

   若有一個未符合就跳去檢查左側球拍，若都符合就將速度反向(反彈)

```
; check if the ball is colliding with the right paddle
check_collision_paddle2:
    mov ax, ball_xpos       ; condition 1 of ball colliding w/ right paddle
    add ax, ball_size       ;碰撞條件:ball_max_x > paddle_min_x
    cmp ax, paddle2_xpos
    jng check_collision_paddle1  ;jmp if not greater

    mov ax, paddle2_xpos    ; condition 2 of ball colliding w/ right paddle
    add ax, paddle_width    ;碰撞條件:ball_min_x < paddle_max_x
    cmp ball_xpos, ax
    jnl check_collision_paddle1  ;jump if not less

    mov ax, ball_ypos       ; condition 3 of ball colliding w/ right paddle
    add ax, ball_size       ;碰撞條件:ball_max_y > paddle_min_y
    cmp ax, paddle2_ypos
    jng check_collision_paddle1

    mov ax, paddle2_ypos    ; condition 4 of ball colliding w/ right paddle
    add ax, paddle_height   ;碰撞條件:ball_min_y < paddle_max_y
    cmp ball_ypos, ax
    jnl check_collision_paddle1
;上述四個條件必須同時達成才算碰撞，因此有一個不符合就會直接去檢查另一邊
; if the program reaches this point, the ball is colliding with the right paddle
    neg ball_speed_x  ;反彈==速度反向
    ret
```

左側碰撞方式同上述所述

```
check_collision_paddle1:
    mov ax, ball_xpos       ; condition 1 of ball colliding w/ left paddle
    add ax, ball_size
    cmp ax, paddle1_xpos
    jng exit

    mov ax, paddle1_xpos    ; condition 2 of ball colliding w/ left paddle
    add ax, paddle_width
    cmp ball_xpos, ax
    jnl exit

    mov ax, ball_ypos       ; condition 3 of ball colliding w/ left paddle
    add ax, ball_size
    cmp ax, paddle1_ypos
    jng exit

    mov ax, paddle1_ypos    ; condition 4 of ball colliding w/ left paddle
    add ax, paddle_height
    cmp ball_ypos, ax
    jnl exit

    ; if the program reaches this point, the ball is colliding with the left paddle
    neg ball_speed_x  ;反彈
    jmp give_player_point ;玩家得分
    ret
```

10.若玩家拍子有成功碰撞就會幫玩家加分，並且改變求的顏色

```
    give_player_point:
        inc player_points ;加分
        add ball_color, 01h ;每得分一次就改變顏色
    ;    call reset_ball
        call update_player_points

    exit:
        ret

move_ball endp
```

11. 玩家球拍的移動:先判斷有沒有按鍵輸入，若沒有就去移動電腦的拍子，w代表向上，s代表向下(大小寫皆可)

```
move_paddles proc
    mov ah, 01h                 ; check if any key is being pressed
    int 16h                     ; check if ZF = 1
    jz check_paddle2_movement    ; jump if zero

    mov ah, 0
    int 16h

    cmp al, 77h     ; 'w'
    je move_paddle1_up
    cmp al, 57h     ; 'W'
    je move_paddle1_up

    cmp al, 73h     ; 's'
    je move_paddle1_down
    cmp al, 53h     ; 'S'
    je move_paddle1_down
    jmp check_paddle2_movement
```

12.玩家球拍的移動:利用paddle_speed(如同位移量)以達到移動效果，這邊會判斷球拍有無到達上下邊界，若有的話不能讓球拍超過

```
move_paddle1_up:
    mov ax, paddle_speed
    sub paddle1_ypos, ax

    mov ax, paddle_border
    cmp paddle1_ypos, ax  ;判斷有沒有到最上端
    jl fix_paddle1_up
    jmp check_paddle2_movement

    fix_paddle1_up:
        mov ax, paddle_border
        mov paddle1_ypos, ax    ;讓paddle停在最上面
        jmp check_paddle2_movement

move_paddle1_down:
    mov ax, paddle_speed
    add paddle1_ypos, ax

    mov ax, window_height
    sub ax, paddle_border
    sub ax, paddle_height
    cmp paddle1_ypos, ax
    jg fix_paddle1_down
    jmp check_paddle2_movement

    fix_paddle1_down:
        mov paddle1_ypos, ax
        jmp check_paddle2_movement
```

13.電腦球拍的移動: 電腦的球拍會自動去追球，無論如何都會打到球，因此玩家勝利條件是獲取10分(成功回擊10次)，這裡程式先判斷，球拍跟球的相對關係，再決定要往上還是往下，ai_precision設定為paddle2長度除以二

```
check_paddle2_movement: ;paddle2 要自動去追球
    mov ax, ball_ypos
    add ax, ball_size
    sub ax, ai_precision
    cmp ax, paddle2_ypos
    jl move_paddle2_up

    mov ax, paddle2_ypos
    add ax, paddle_height
    sub ax, ai_precision
    cmp ax, ball_ypos
    jl move_paddle2_down
    ret
```

14.電腦球拍的移動:跟玩家移動球拍的概念相同，要判斷有沒有到達上下邊界，並使其不超過邊界

```
        move_paddle2_up:
            mov ax, paddle_speed
            sub paddle2_ypos, ax

            mov ax, paddle_border
            cmp paddle2_ypos, ax
            jl fix_paddle2_up
            ret

            fix_paddle2_up:
                mov ax, paddle_border
                mov paddle2_ypos, ax
                ret

        move_paddle2_down:
            mov ax, paddle_speed
            add paddle2_ypos, ax

            mov ax, window_height
            sub ax, paddle_border
            sub ax, paddle_height
            cmp paddle2_ypos, ax
            jg fix_paddle2_down
            ret

            fix_paddle2_down:
                mov paddle2_ypos, ax
                ret

move_paddles endp
```

15.重製球體:將球體座標設定為起始位置，並將速度反向

```
reset_ball proc
    mov ax, ball_start_xpos
    mov ball_xpos, ax

    mov ax, ball_start_ypos
    mov ball_ypos, ax

    neg ball_speed_x
    neg ball_speed_y

    ret
reset_ball endp
```

16.繪製遊玩中的介面:先選定要輸出的位置，然後判斷玩家選擇的難度，輸出相對應的介面

```
draw_ui proc
    mov ah, 02h     ; cursor position
    mov bh, 00h     ; page number
    mov dh, 01h     ; set row
    mov dl, 04h     ; set column
    int 10h

    mov ah, 09h
    lea dx, player_points_mes
    int 21h

    mov ah, 02h     ; cursor position
    mov bh, 00h     ; page number
    mov dh, 01h     ; set row
    mov dl, 1dh     ; set column
    int 10h

    cmp difficulty, 01h
    je display_easy_mes
    cmp difficulty, 02h
    je display_medium_mes
    cmp difficulty, 03h
    je display_hard_mes
    ret
```

17.繪製遊玩中的介面:依照遊戲模式輸出不同介面

```
    display_easy_mes:
        mov ah, 09h
        lea dx, game_easy_mes
        int 21h
        ret

    display_medium_mes:
        mov ah, 09h
        lea dx, game_medium_mes
        int 21h
        ret

    display_hard_mes:
        mov ah, 09h
        lea dx, game_hard_mes
        int 21h
        ret

    ret
draw_ui endp
```

18.紀錄玩家分分數，以輸出在螢幕上，07h用在位址上是指要將當前分數放在字串的第七個位置

```
update_player_points proc
    xor ax, ax
    mov al, player_points

    add al, 30h
    mov [player_points_mes][07h], al

    ret
update_player_points endp
```

19.遊戲結束的介面:一樣先設定輸出的位置，然後輸出標題跟獲勝者

```
display_game_over proc
    call clear_screen

    mov ah, 02h     ; display the game over title
    mov bh, 00h
    mov dh, 06h
    mov dl, 05h
    int 10h

    mov ah, 09h
    lea dx, game_over_mes
    int 21h

    mov ah, 02h     ; display the winner title
    mov bh, 00h
    mov dh, 09h
    mov dl, 05h
    int 10h

    cmp winner, 01h
    je winner_is_player
    jmp winner_is_ai
```

20.遊戲結束的介面:根據不同獲勝者輸出不同字串，並且顯示遊戲結束後的選單(重新遊戲)

```
winner_is_player:
    mov ah, 09h
    lea dx, game_win_mes
    int 21h
    jmp display_restart

winner_is_ai:
    mov ah, 09h
    lea dx, game_lose_mes
    int 21h
    jmp display_restart

display_restart:
    mov ah, 02h     ; display the play again title
    mov bh, 00h
    mov dh, 0ch
    mov dl, 05h
    int 10h

    mov ah, 09h
    lea dx, game_restart_mes
    int 21h
```

21.輸出選單(離開)、並且判斷玩家按下的按鍵，做出對應的功能

```
display_back_to_menu:
    mov ah, 02h     ; display the back to menu title
    mov bh, 00h
    mov dh, 0eh
    mov dl, 05h
    int 10h

    mov ah, 09h
    lea dx, game_ends_mes
    int 21h

mov ah, 02h     ; display the game over border
mov bh, 00h
mov dh, 12h
mov dl, 05h
int 10h

mov ah, 09h
lea dx, game_border_mes
int 21h

mov ah, 0       ; waiting for user input
int 16h

cmp al, 'R'
je restart_game
cmp al, 'r'
je restart_game

cmp al, 'E'
je back_to_main
cmp al, 'e'
je back_to_main
ret
```

22.重新遊玩(模式相同)或是回到遊戲開始的主選單

```
    restart_game:
        mov activeness, 01h
        call update_player_points
        call reset_ball
        ret

    back_to_main:
        mov menu, 0
        mov player_points, 0
        call reset_ball
        call update_player_points
        mov paddle1_ypos, 0ah
        mov paddle2_ypos, 0ah
        ret

display_game_over endp
```

23.輸出遊戲開始時的主選單

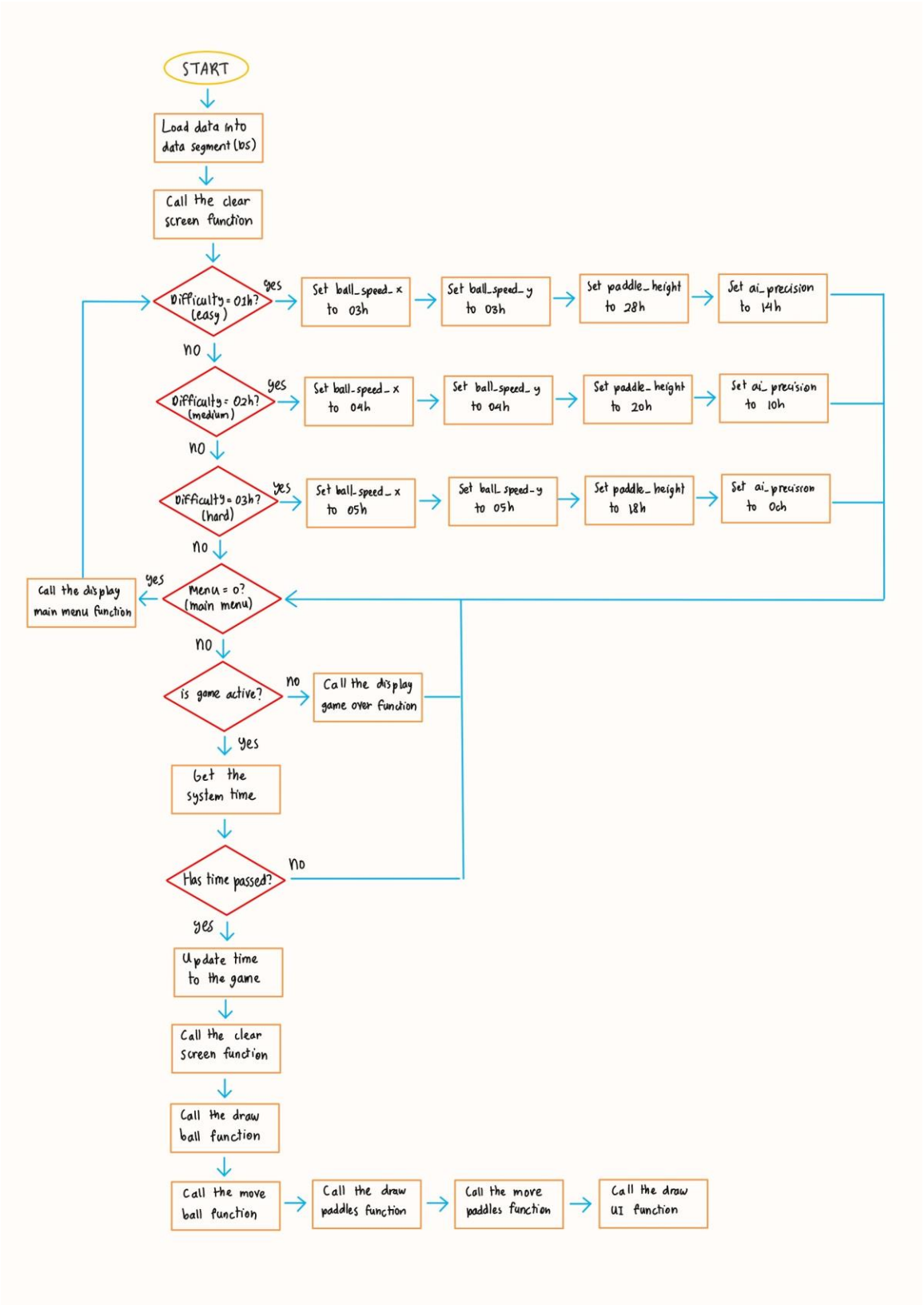
```
display_main_menu proc
    call clear_screen

    mov ah, 02h     ; display the welcome title
    mov bh, 00h
    mov dh, 04h
    mov dl, 04h
    int 10h

    mov ah, 09h
    lea dx, menu_welcome_mes
    int 21h

    mov ah, 02h     ; display the rule title
    mov bh, 00h
    mov dh, 07h
    mov dl, 04h
    int 10h

    mov ah, 09h
    lea dx, menu_rule_mes
    int 21h

    mov ah, 02h     ; display the choose difficulty title
    mov bh, 00h
    mov dh, 0ah
    mov dl, 04h
    int 10h

    mov ah, 09h
    lea dx, menu_choose_mes
    int 21h

    mov ah, 02h     ; display the easy title
    mov bh, 00h
    mov dh, 0ch
    mov dl, 04h
    int 10h

    mov ah, 09h
    lea dx, menu_easy_mes
    int 21h

    mov ah, 02h     ; display the medium title
    mov bh, 00h
    mov dh, 0eh
    mov dl, 04h
    int 10h

    mov ah, 09h
    lea dx, menu_medium_mes
    int 21h

    mov ah, 02h     ; display the hard title
    mov bh, 00h
    mov dh, 10h
    mov dl, 04h
    int 10h

    mov ah, 09h
    lea dx, menu_hard_mes
    int 21h
```

24.繪製完選單等待玩家輸入按鍵選擇難度，exit_program用來結束程式

```
mov ah, 02h     ; display the esc title
mov bh, 00h
mov dh, 13h
mov dl, 04h
int 10h

mov ah, 09h
lea dx, menu_exit_mes
int 21h

mov ah, 02h     ; display the bottom border
mov bh, 00h
mov dh, 15h
mov dl, 04h
int 10h

mov ah, 09h
lea dx, menu_border_mes
int 21h

mov ah, 0       ; waiting for user input
int 16h

cmp al, '1'
je game_is_easy
cmp al, '2'
je game_is_medium
cmp al, '3'
je game_is_hard
cmp al, 1bh
je exit_program
ret
```

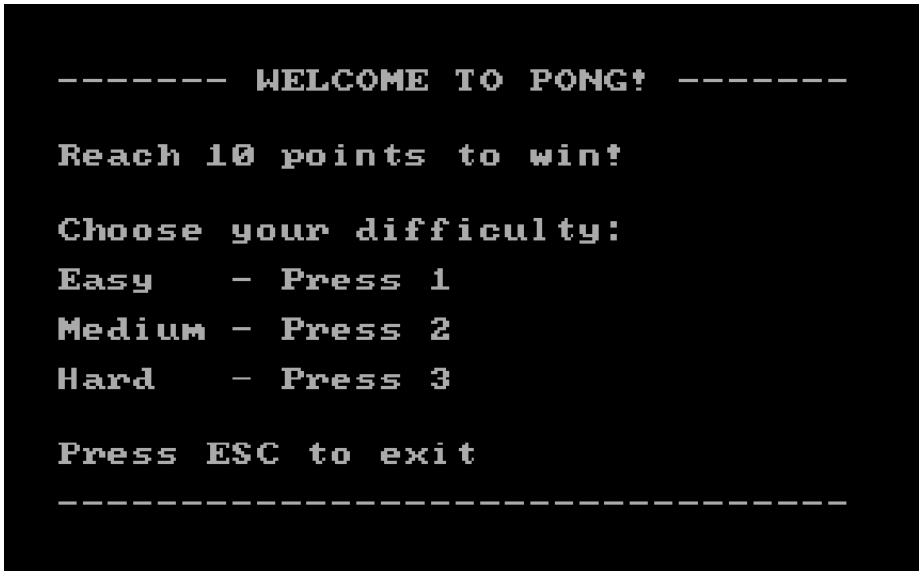
```
    game_is_easy:
        mov difficulty, 01h
        mov activeness, 01h
        mov menu, 01h
        ret

    game_is_medium:
        mov difficulty, 02h
        mov activeness, 01h
        mov menu, 01h
        ret

    game_is_hard:
        mov difficulty, 03h
        mov activeness, 01h
        mov menu, 01h
        ret

    exit_program:
        mov ax, 4c00h
        int 21h

display_main_menu endp

end main
```

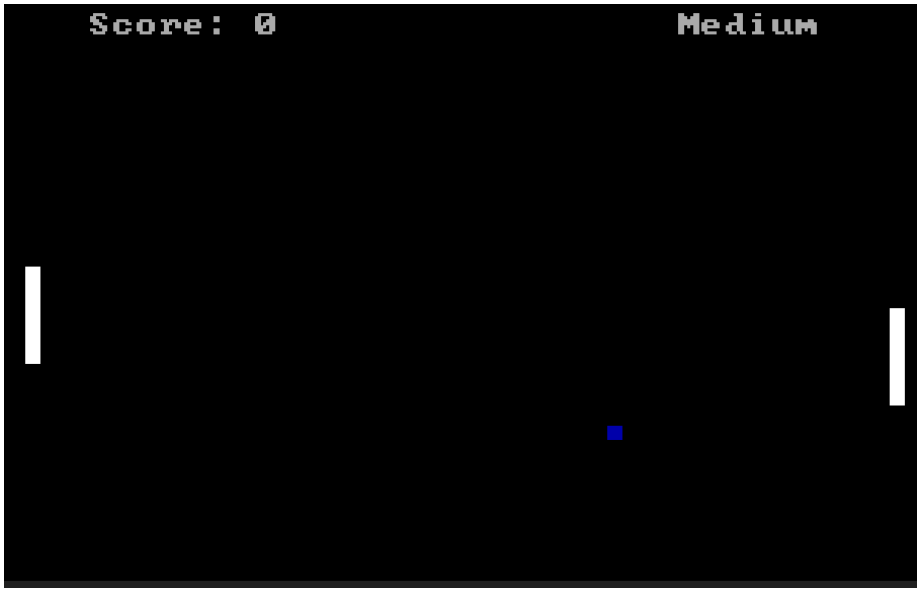
二、 流程圖

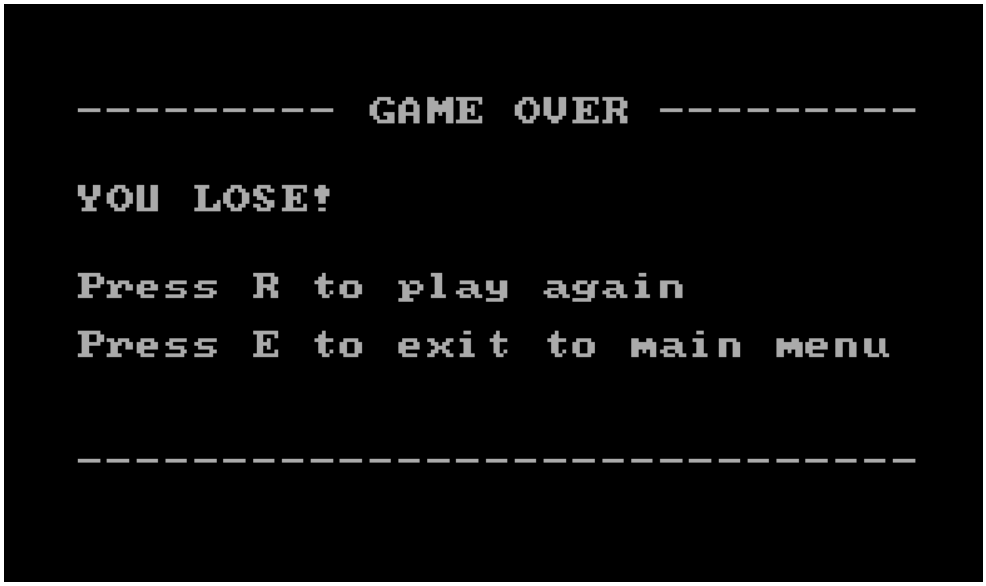

START
Load data into data segment (DS)
Call the clear screen function
Difficulty = 01h? (easy)
yes
Set ball_speed_x to 03h
Set ball_speed_y to 03h
Set paddle_height to 28h
Set ai_precision to 14h
no
Difficulty = 02h? (medium)
yes
Set ball_speed_x to 04h
Set ball_speed_y to 04h
Set paddle_height to 20h
Set ai_precision to 10h
no
Difficulty = 03h? (hard)
yes
Set ball_speed_x to 05h
Set ball_speed_y to 05h
Set paddle_height to 18h
Set ai_precision to 0Ch
no
Menu = 0? (main menu)
yes
Call the display main menu function
no
is game active?
no
Call the display game over function
yes
Get the system time
Has time passed?
no
yes
Update time to the game
Call the clear screen function
Call the draw ball function
Call the move ball function
Call the draw paddles function
Call the move paddles function
Call the draw UI function

三、 實習結果 (遊戲運作畫面)

以下貼上遊戲開始、遊戲過程跟遊戲結果的畫面，詳細遊玩過程會放置影片檔說明







四、 心得 (同一組兩位組員要分開寫，附在同一份報告裡)

- 黃偉智: 透過這個期末專題，我學到了很多。我之前對組合語言的繪圖還不太熟，可是在寫程式的過程中，我的能力慢慢進步了。當我第一次認識組合語言的時候，什麼都不了解，誰想到到時候可以使用組合語言做出來一個游戲。組合語言雖然很複雜，但是我覺得還挺好玩的。
- 韓承志: 這次的期末專題非常有趣，利用組合語言來設計遊戲，當中有許多的困難，版本問體、電腦問題等都非常的棘手，組合語言是非常底層的程式語言，要用組合語言編寫遊戲需要非常複雜的程式跟清晰的邏輯，讓我們對於程式的理解又更近一步了