Ping Pong Ball

Final Project Report

By

409261287
張甄庭

409261328
鍾馥羽

X86 Assembly Language
Fall 2021
Date Submitted: January 22, 2022

Abstract (required):

我們要做的是雙人遊戲，因為選修課有做過同性質的專題，所以我們決定做乒乓球遊戲。它須考量兩側 paddle、上下左右邊界、球的移動和判斷、鍵盤的讀取，除了遊戲介面，我們還加上了 Menu 來讓玩家知道如何操作。在這份報告裡，我們區分了各個項目的程式碼，不同的 procedure 放在不同的 asm 檔案裡，且報告中詳述了各個 label 間的邏輯和運行順序，還有過程中遇到的問題和解決辦法以及報告後新增的功能，最後也有附上 Demo 影片的連結可觀看最終成果!

Table of Contents (required):

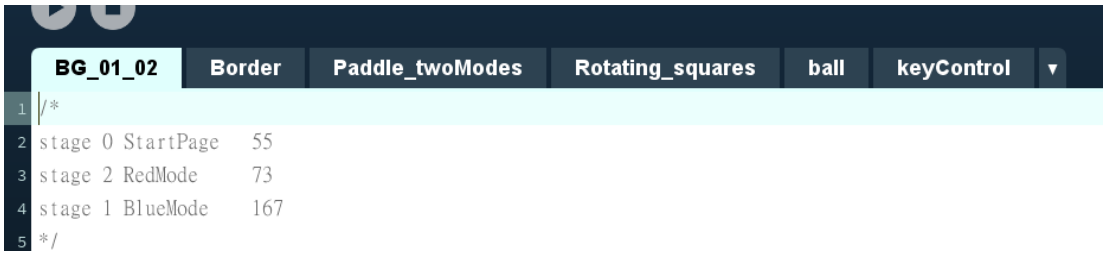
Introduction (required):

我們設計的是一個乒乓球的遊戲，是一個雙人遊戲，玩家一需要使用 W、S 鍵，來控制左邊的藍色 PADDLE 上下移動，玩家二則需要使用上、下鍵，來控制右邊的紅色 PADDLE 上下移動，當某一方未接到球，則另一方獲勝！需等待幾秒鐘遊戲就會繼續。

Project plan or Test Plan and Test Cases (required):

- **Project Plan 專題目標**
  1. 因為我們是第一次用組語寫遊戲，一開始並不是很孰悉，因此我們最初希望至少能做出一個可以玩的遊戲，**整體的遊戲介面是完整的**，然後我們前面有提到想設計雙人遊戲，這方面也希望可以達成
  2. **做出 GUI** 還有一些簡單的設計
  3. 做出 **Menu**，包括組員介紹還有遊戲規則等等
  4. 看到課本後的 Ascii code 我們希望能把球改成笑臉。
  5. 因為我們有時候會跟其他二乙班的同學問問題，也看過他們的組語作業，而他們其中一項作業被老師要求利用**分開的檔案寫 procedure**，詢問過如何使用後，我們一開始就打算**拆成好幾個 asm 檔案編寫**。
  6. 我們希望這個遊戲能有計分的功能，當某一方先達到 5 分就獲勝，但我們現在則是，有一方沒接到球，需要再等幾秒鐘(1.5 秒)，遊戲才會繼續

  ❖ **參考互動專題程式碼架構**



在 processing 裡面我們也是拆成好幾個 function 和 class 來編寫，優點是:

1. **方便分工**
2. **版面乾淨**
3. **容易抓錯**

因此組語也是按照同樣的方式分開 procedures 和共同編輯；除此之外我們(張甄庭&鍾馥羽)因為共同修了 Web 基本原理與技術的選修課程，也是在做 Web 期末專題時學會了用 VS2019 commit 到 GitHub 的功能，因此順利的**不見面也能同步專題進度**~

❖ **Test Plan & Test Cases**

所以的草稿圖、流程圖和測試過程，都在 Discussion & Analysis 裡面詳細提及。

Discussion or Analysis (required):

- **整體主程式(ppball.asm)**
  - ❖ **大概的運行(labels)**

  - ❖ **全域變數**

一開始我們先設定好部份的變數，例如顏色、長寬高、各座標，接著視程式的增長以及procedure的新增，一點一點新增我們的變數。

```
; 框架大小和顏色宣告
PLAY_T_EDGE_OFFSET equ 5d
PLAY_L_EDGE_OFFSET equ 20d
BOARD_R_LENGTH equ 80d
BOARD_C_LENGTH equ 1d
BOARD_BETWEEN equ 25d

;新增左右兩側看不到的黑色boards
PLAY_BLACK_X1 equ 14d
PLAY_BLACK_Y1 equ 4d
PLAY_BLACK_X2 equ 104d
PLAY_BLACK_Y2 equ 4d
```

```
MENU_PPBALL_COLOR equ yellow + (cyan*16)
MENU_TEXT_COLOR equ white + (black*16)
UP_DOWN_B_COLOR equ (lightGray * 16)
PADDLE_COLOR equ blue+(lightCyan * 16)
PADDLE_COLOR_TWO equ yellow+(lightRed*16)
BALL_COLOR equ black+(yellow * 16)

;球以每0.1秒速度移動
FRAME_RATE equ 100d             ;球跑的速度

;reset時間等待1.5秒
RESET_BALL_RATE equ 1500d       ;球reset
```

在經過.data 處理部分值後，傳入各個procedure。例如ppballMain

```
ppballMain:

    ;上下板子
    invoke DrawFrame, UP_DOWN_B_COLOR, PLAY_T_EDGE_OFFSET, PLAY_L_EDGE_OFFSET,
                      BOARD_R_LENGTH, BOARD_BETWEEN, BOARD_C_LENGTH, addr space

    ;玩家paddle初始和移動
    invoke ShowFirstTwoPaddles, PADDLE_COLOR, PADDLE_COLOR_TWO, PLAY_BLACK_X1, PLAY_BLACK_Y1, PLAY_BLACK_X2, PLAY_BLACK_Y2,
                                addr p1X, addr p1Y, addr p2X, addr p2Y, addr paddleHeight, addr spacePaddle
    invoke ReadKeyBoard, PADDLE_COLOR, PADDLE_COLOR_TWO, addr p1X, addr p1Y, addr p2X, addr p2Y,
                         paddleHeight, playTopEdge, playLowEdge, addr spacePaddle

    ;球的
    invoke Ball, BALL_COLOR, addr Ball_X, addr Ball_Y, addr xMove, addr yMove, addr buffer, PLAY_T_EDGE_OFFSET,
                 BOARD_BETWEEN, p1X, p1Y, p2X, p2Y, paddleHeight, RESET_BALL_RATE

    invoke SpeedOfBall, FRAME_RATE


   jmp ppballMain
```

同時也要注意inc檔案裡各個procedure的PROTO指引，舉例:

```
66  ResetBall proto,
67      Ball_X: ptr dword,
68      Ball_Y: ptr dword,
69      newBall_X: dword,
70      newBall_Y: dword,
71      RESET_BALL_RATE: ptr dword
```

```
3   .code
4   ResetBall proc,
5       Ball_X: ptr dword,
6       Ball_Y: ptr dword,
7       newBall_X: dword,
8       newBall_Y: dword,
9       RESET_BALL_RATE: ptr dword
```

resetBall.asm裡

- **Menu頁面介紹 (_Menu.asm)**

為了讓我們遊戲更完整，我們覺得 Menu 頁面是不可少的一部分，下方左圖是我們當時設想的 Menu 呈現樣子，而下方右圖是實際 Demo 的截圖

這裡我們主要使用 call setTextColor、macros 裡的 mGotoxy 和 mWriteString 來達成。

舉例:

Menu.asm 第一個藍色 P 字母

```
39          ;P
40          mGotoxy 30,1
41          mWriteString OFFSET space2
42          mGotoxy 30,2
43          mWriteString OFFSET space2
44          mGotoxy 30,3
45          mWriteString OFFSET space2
46          mGotoxy 30,4
47          mWriteString OFFSET space2
48          mGotoxy 30,5
49          mWriteString OFFSET space2
50          mGotoxy 30,6
51          mWriteString OFFSET space2
52
```

印出系級

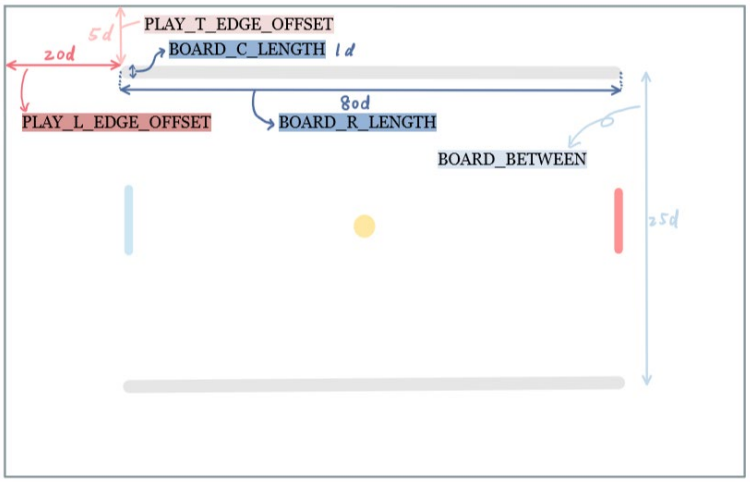
```
3   .data
4   ;Menu Page
5   MenuTitle byte "PPball !!",0
6   MenuClass byte "- 資工二甲 -" ,0
7   MenuOurName1 byte "張甄庭 409261287",0
8   MenuOurName2 byte "鍾馥羽 409261328",0
9   MenuIntroTitle byte "**********  -  遊戲介紹
10  MenuPlayerOne1 byte "玩家一需要使用W、S鍵",0
```

```
210
211     ;WE ARE?
212     mov eax, white + (gray*16)
213     call SetTextColor
214     mGotoxy 53,10
215     mWriteString OFFSET MenuClass
216
217     mov eax, colorText
218     call SetTextColor
```

- **DrawFrame.asm** (上下左右frames 繪製)

  我們一開始有想過直接用好幾個空白鍵上色直接畫出上下兩排的 boarder，不過因為當時我們還沒有確定我們整個遊戲版面各個東西的比例，為了不要在我們每次修改大小時就要在不同地方調整一次，我們 boarder 是用 loop 的方式來畫的。

  草圖:



需要的參數:
(在 ppball.asm 裡面的全域變數)

```
; 框架大小和顏色宣告
PLAY_T_EDGE_OFFSET equ 5d
PLAY_L_EDGE_OFFSET equ 20d
BOARD_R_LENGTH equ 80d
BOARD_C_LENGTH equ 1d
BOARD_BETWEEN equ 25d
```

還有 `UP_DOWN_B_COLOR equ (lightGray * 16)`

- ❖ **DrawFrame 傳入的參數們:**

ppball.inc (註解是對照全域變數)

```
DrawFrame proto,
    color: dword,                ;UP_DOWN_B_COLOR 設light gray
    playTopEdgeOffset: dword,    ;PLAY_T_EDGE_OFFSET
    playLeftEdgeOffset: dword,   ;PLAY_L_EDGE_OFFSET
    boardRowLength: dword,       ;BOARD_R_LENGTH
    boardsinBetween: dword,      ;BOARD_BETWEEN
    boardColumnLength: dword,    ;BOARD_C_LENGTH
    space: ptr byte
```

_darwframe.asm

```
.code
DrawFrame proc,
    color: dword,                ;灰
    playTopEdgeOffset: dword,
    playLeftEdgeOffset: dword,
    boardRowLength: dword,
    boardsinBetween: dword,
    boardColumnLength: dword,
    space: ptr byte
```

- ❖ 上下 **boarder loop 邏輯:**

LOOP 跑之前:

```
    mov eax, color      ;灰色
    call SetTextColor

    ;定下border 左上座標(20,5)
    mov ebx, playLeftEdgeOffset     ;20d
    mov eax, playTopEdgeOffset      ;5d
    mov dl, bl      ;column X座標
    mov dh, al      ;row Y座標
    call Gotoxy

    mov edx, space
    mov ecx, 2          ;UpDownBoards loop跑兩次因為有上下兩個
```

Up Down Boards: 要跑兩次因為有兩個灰色板子

```
33  UpDownBoards:
34      push ecx            ;ecx = 2
35      mov ecx, boardColumnLength      ;ecx = 1 (boardColumnLength)
36
```

Column of Board: 跑一次因為高度為 1

```
37      ColumnOfBoard:
38          push ecx    ;ecx = 1
39          mov ecx, boardRowLength         ;ecx = 80 (boardRowLength)
40
```

Draw The Board: 畫 80 次(EDX 裡的 space” “ 一直填填填填~~)

```
41          ;厚度為一的board畫80次
42          DrawTheBoard:
43              call WriteString
44          loop DrawTheBoard
```

Loop DrawTheBoard

```
45
46          pop ecx     ;ecx = 1
47      loop ColumnOfBoard      ;跑一次
```

Loop ColumnOfBoard

```
49      pop ecx         ;ecx = 2
50
51      push eax        ;5d
52      push edx        ;儲存edx
53
54      add eax, boardsinBetween            ; 兩個board間距25d(5+25=30)
55      mov dl, bl      ;X座標不變
56      mov dh, al
57      call Gotoxy
58      pop edx         ;恢復
59      pop eax
60
61      loop UpDownBoards
```

Loop UpDownBoards

❖ **遇到問題/心得**

之前做作業時還不太會使用 stack，如果沒有用 stack 我們可能會瘋狂的(?) 存變數再移來移去，除了要注意一下壓入和彈出的資料是什麼以外，畫 Boarder 時沒遇到什麼大問題。

但只是單單畫個上下板子，也跟在寫 processing(JAVA)時差很多，例如下方截圖:

```
class Border{
  int x,y1,y2,w,h;
  int s=100;
  Border(){
    x = 600;
    y1 = 0;
    y2 = 800;
    w = 1300;
    h = 10;
  }
```

```
  void showBlueB(){
    fill(0,0,255);
      rectMode(CENTER);
      rect(x,y1,w,h);
      rect(x,y2,w,h);
  }
}
```

同樣是傳入起始座標 XY 和長寬，但用組語時有種要帶著它一步一步畫出長方形邊框的感覺。

- **有關玩家Paddles** (初始ShowFirstTwoPaddle.asm 和移動ReadKeyBoard.asm)
  玩家paddles的部分牽連到了幾個procedure，除了主程式ppball 有設定它們最初的位置以外，paddle 最初畫出來的程式碼寫在ShowFirstTwoPaddles內；而做出移動感覺的程式碼在ReadKeyBoard裡面

  ❖ **ShowFirstTowPaddle傳入的參數們**

ppball.inc (註解是對照全域變數)

```
ShowFirstTwoPaddles proto,
    color: dword,
    color2: dword,
    blackX1: dword,                 ;PLAY_BLACK_X1
    blackY1: dword,                 ;PLAY_BLACK_Y1
    blackX2: dword,                 ;PLAY_BLACK_X2
    blackY2: dword,                 ;PLAY_BLACK_Y2
    p1X: PTR dword,                 ; 玩家一 x座標
    p1Y: PTR dword,                 ; 玩家一 y座標
    p2X: PTR dword,                 ; 玩家二 x座標
    p2Y: PTR dword,                 ; 玩家二 y座標
    paddleHeight: dword,            ;paddle 高度
    spaceWithStar: ptr byte         ;"*"
```

_ShowFirstTowPaddle.asm

```
.code
ShowFirstTwoPaddles proc,
     color: dword,
    color2: dword,
    blackX1: dword,                 ;PLAY_BLACK_X1
    blackY1: dword,                 ;PLAY_BLACK_Y1
    blackX2: dword,                 ;PLAY_BLACK_X2
    blackY2: dword,                 ;PLAY_BLACK_Y2
    p1X: PTR dword,                 ; x座標
    p1Y: PTR dword,                 ; y座標
    p2X: PTR dword,                 ; x座標
    p2Y: PTR dword,                 ; y座標
    paddleHeight: dword,            ;paddle 高度
    spaceWithStar: ptr byte         ;"*"
```

  ❖ **ReadKeyBoard傳入的參數們:**

ppball.inc (註解是對照全域變數)

```
ReadKeyBoard proto,
    color: dword,
    color2: dword,
    p1X: PTR dword,                 ; 玩家一 x座標
    p1Y: PTR dword,                 ; 玩家一 y座標
    p2X: PTR dword,                 ; 玩家二 x座標
    p2Y: PTR dword,                 ; 玩家二 y座標
    paddleHeight: dword,            ;5
    playTopEdge: dword,
    playLowEdge: dword,
    spaceWithStar: ptr byte         ;"*"
```

_ReadKeyBoard.asm

```
.code
ReadKeyBoard proc,
     color: dword,
    color2: dword,
    p1X: PTR dword,                 ; 玩家一 x座標
    p1Y: PTR dword,                 ; 玩家一 y座標
    p2X: PTR dword,                 ; 玩家二 x座標
    p2Y: PTR dword,                 ; 玩家二 y座標
    paddleHeight: dword,            ;5
    playTopEdge: dword,             ;最上座標
    playLowEdge: dword,             ;最下座標
    spaceWithStar: ptr byte         ;"*"
```

  ❖ **玩家Paddles初始** (ShowFirstTowPaddle.asm)
  我們兩個板子的位置並不是直接指定它們的初始位置，如同在DrawFrame 介紹裡提到的，因為一開始還不確定比例所以我們在分工前就有討論到板子的位置先用Frame草圖的那些參數來定好兩側paddle的位置，然後整體在ShowFirstTwoPaddle畫好後再進行微調。如同frame畫法，我們用Loop來畫板子。

以左側paddle為例:

左側玩家一

```
69        ; p1 paddle 初始
70        pushad
71        ;ptr 宣告的記得加[]
72        ;x座標
73        mov eax, [p1X]
74        mov dl, byte PTR [eax]
75
76        ;y座標
77        mov eax, [p1Y]
78        mov dh, byte PTR [eax]
79
80        ;paddle高度傳入ECX
81        mov eax, [paddleHeight]
82        mov ecx, [eax]              ;ecx = 5
```

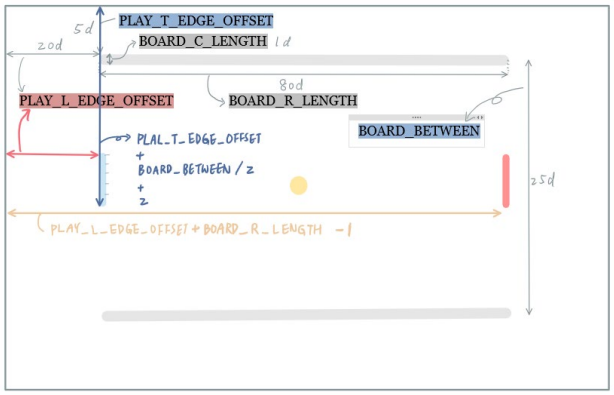
player1PaddleDraw: (跑五次)

```
83    player1PaddleDraw:
84        call Gotoxy                 ;新的Y座標
85
86        mov eax, color              ;淺藍底深藍字
87        call SetTextColor
88        push edx
89        mov edx, spaceWithStar      ;"*"
90        call WriteString
91        pop edx
92
93        dec dh                    ;每次Y座標-1，由下往上畫
94        loop player1PaddleDraw
95
```

Loop player1PaddleDraw

```
96        popad
97        ; 結束p1
98
```

由於我們paddle生成方式為由下往上拼，下面為ppball.asm裡左右側玩家的XY座標設定:




```
42    ;paddle 起始座標
43    ;左側player1
44    p1X dword (PLAY_L_EDGE_OFFSET)                ;設定p1 paddle 位置
45    p1Y dword (PLAY_T_EDGE_OFFSET + (BOARD_BETWEEN / 2) + 2)
46
47    ;右側player2
48    p2X dword (PLAY_L_EDGE_OFFSET + BOARD_R_LENGTH - 1)
49    p2Y dword (PLAY_T_EDGE_OFFSET + (BOARD_BETWEEN / 2) + 2)
50
51    paddleHeight dword 5d       ;paddle長
52
```

❖ **移動Paddle** (ReadKeyBoard.asm)

接下來就是paddle移動時的procedure了，這裡主要注意的點為1.鍵盤的讀取 2.paddle 上下移動時的圖形變化 3.阻止paddle超出範圍。

- **鍵盤讀取**

  在互動媒體專題時，我們就是利用WS按鍵和上下鍵來讓玩家們移動它們的paddle。

  以下是processing和組語的比較:

```
//左邊玩家一用WS控制上下移動
if (w == true){
  lt.movep_up();
}
if (s == true){
  lt.movep_down();
}
//右邊玩家用上下鍵控制上下移動
if (p_up == true){
  rt.movep_up();
}
if (p_down == true){
  rt.movep_down();
}
```

主程式接收

```
//按鍵壓下paddle移動
void keyPressed()
{
  if (keyCode == DOWN){
    p_down = true;
  }
  if (keyCode == UP){
    p_up = true;
  }
  if (key == 'W' || key == 'w'){
    w = true;
  }
  if (key == 'S' || key == 's' ){
    s = true;
  }
}
```

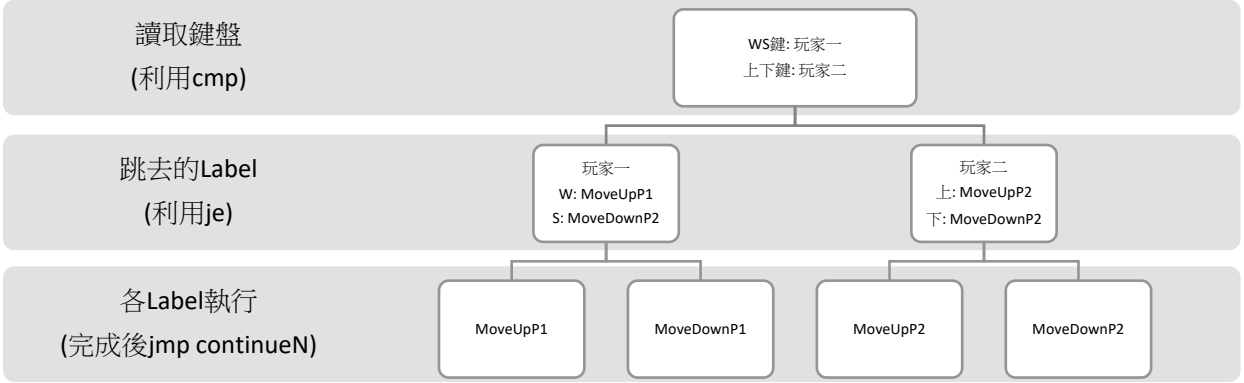
```
//按鍵放開paddle停止移動
void keyReleased()
{
  if (keyCode == DOWN){
    p_down = false;
  }
  if (keyCode== UP){
    p_up = false;
  }
  if (key == 'W' || key == 'w'){
    w = false;
  }
  if (key == 'S' || key == 's'){
    s = false;
  }
}
```

```
cmp al, "w"
je MoveUpP1                ;p1 paddle向上
cmp al, "s"
je MoveDownP1              ;p2 paddle向下

cmp ah, 48h                ;向上鍵
je MoveUpP2                ;p2 paddle向上
cmp ah, 50h                ;向下鍵
je MoveDownP2              ;p2 paddle向下

jmp continueN
```

組語WS鍵和上下鍵讀取，這裡我們原本遇到的問題主要是上下鍵讀取一開始不會用，翻到課本後面打了48h、50h也不能跑，之後查了資料才知道上下鍵讀取要用AH暫存器!而不是AL。

- **各Lable指令**

  各個Label主要有三個要進行的事情，一是限制移動範圍，二是刪除一格(與前進方向相反)，三是新增一格(朝著前進方向)



這裡比較**MoveUpP1**和**MoveDownP2**:

限制移動範圍

```
37
38  ;藍色板板有MoveUpP1, MoveDownP1兩個label
39  MoveUpP1:
40      ;當板子碰到上border，不給動(cont.)
41      ; eax = p1Y , [ebx] 存著當時的y座標!!!!
42      mov ebx, p1Y
43      mov eax, [ebx]
44      sub eax, [paddleHeight]             ; y - height
45      cmp eax, playTopEdge                ;如果小於等於
46      jbe continueN                       ;跳continue, 忽略上鍵要求
47
```

限制移動範圍

```
76  MoveDownP1:
77      ;當板子碰到下border，不給動(cont.)
78      ; eax = p1Y , [ebx] 也存著當時的y座標!!!!
79      mov ebx, p1Y
80      mov eax, [ebx]
81      inc eax                             ;p1Y + 1來和最底Y座標進行比較
82      cmp eax, playLowEdge                ;如果Y座標大於等於Edge
83      jae continueN                       ;跳去continueN, 忽略下鍵要求
84
```

兩者相差在Y座標判定

刪除一格

```
48      ;往上移動時，覆蓋最下面的"*",抓當時的XY座標
49      ;因為我們是由下往上畫，所以當時的座標會是最下方
50      mov eax, p1X
51      mov dl, byte PTR [eax]                  ; x 座標
52      mov eax, p1Y
53      mov dh, byte PTR [eax]                  ; y 座標
54      call Gotoxy
55      mov eax, 0                              ; 塗上黑色
56      call SetTextColor
57      mov edx, spaceWithStar
58      call WriteString                        ; 覆蓋
```

刪除一格

```
85      ;往下移動時，覆蓋最上面的"*",抓當時的XY座標
86      ;因為我們是由下往上畫，所以當時的座標會是最下方
87      mov eax, p1X
88      mov dl, byte PTR [eax]                  ; x 座標
89      mov eax, p1Y
90      mov dh, byte PTR [eax]                  ; y 座標
91      sub dh, byte PTR [paddleHeight]         ;減去paddle高度得最高點座標
92      add dh, byte PTR 1                      ;記得加一!!
93      call Gotoxy
94      mov eax, 0                              ; 塗上黑色
95      call SetTextColor
96      mov edx, spaceWithStar
97      call WriteString                        ; 覆蓋最上面的*
```

同樣是覆蓋"*"要注意補*的位置

新增一格並更改座標

```
60      ;向上移動時，除了要擦去下面，也要新增上方"*"
61      mov eax, p1X
62      mov dl, byte PTR [eax]                  ; x 座標
63      mov eax, p1Y
64      mov dh, byte PTR [eax]                  ; y 座標
65      sub dh, byte PTR [paddleHeight]         ;減去paddle高度得最高點座標
66      call Gotoxy
67      mov eax, color                          ;藍色板板~~
68      call SetTextColor
69      mov edx, spaceWithStar
70      call WriteString                        ; 畫上
71
72      ;[ebx]開頭存著當時的y座標
73      sub [ebx], dword PTR 1                  ;[ebx]-1 得新座標
74      jmp continueN
```

新增一格並更改座標

```
99      ;向下移動時，除了要擦去上面，也要新增下方"*"
100     mov eax, p1X
101     mov dl, byte PTR [eax]                  ; x 座標
102     mov eax, p1Y
103     mov dh, byte PTR [eax]                  ; y 座標
104     add dh, byte PTR 1                      ;加一高度得最低點座標
105     call Gotoxy
106     mov eax, color                          ; 藍色板板
107     call SetTextColor
108     mov edx, spaceWithStar
109     call WriteString                        ; 畫上
110
111     add [ebx], dword PTR 1                  ;[ebx]+1 得新座標
112     jmp continueN
```

- **遇到問題/心得**

  當時在寫paddle移動時，遇到了兩個問題。(因為做專題時有跟其他組的同學一起約了圖書館討論室，那組寫貪食蛇，參考了他們的貪食蛇移動程式碼)

  - **問題一：板子移動頭尾不同步**

    第一個是板子移動會像貪食蛇一樣，頭尾不同步，我們想要做出按一下按鍵就同時新增一格和刪去一格，後來發現是，因為我們Y座標同時存在EAX和EBX暫存器，我們原本想說如果在jmp continueN之前改變EBX暫存器的座標就好(因為執行時EAX也會改變)，但這樣就會變成先改變座標(EBX)後執行補上" * "座標(EAX控制)的動作。

    解決辦法是直接在補上米字號前，就直接讓EAX暫存器加一或是減一，跟EBX一起同步變化。

  - **問題二: 往下時一直多出一格!!**

    其實這是數學問題哈哈，但我們找了很久QAQ，假設paddle Y座標底為25，最上面的Y座標是25-paddle高度(5) = 20，但是! 這個不是正確座標，正確座標應該要加一!! 好險最後有發現哈哈哈。

- **有關Ball的移動和判定**

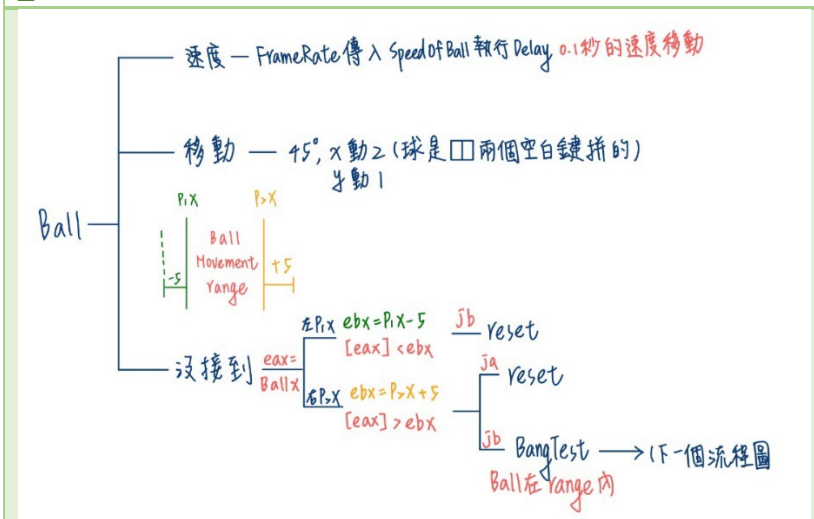
  球的部分總共有四個相關procedure，分別為Ball、 ResetBall、SpeedOfBall、newBallTime。球的程式碼也跟我們互動專題的架構和邏輯差異比較大，因為選修專題的processing可以用球是否撞到指定顏色來做反彈的回應，但組語沒有這種功能，所以這時就需要抓當時的各個座標位置和球的當下座標進行比對。

  - **球的相關程式碼整體架構**

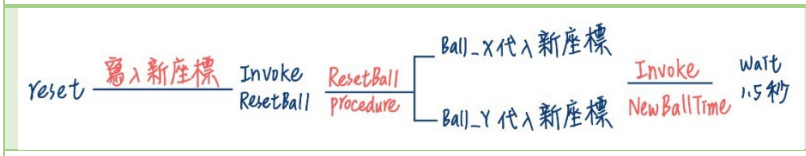
    因為球的程式碼邏輯較為複雜，我們在程式碼裡都有附上詳細的註解，下面的圖為我們的程式邏輯順序。

    - **Ball.asm 架構**

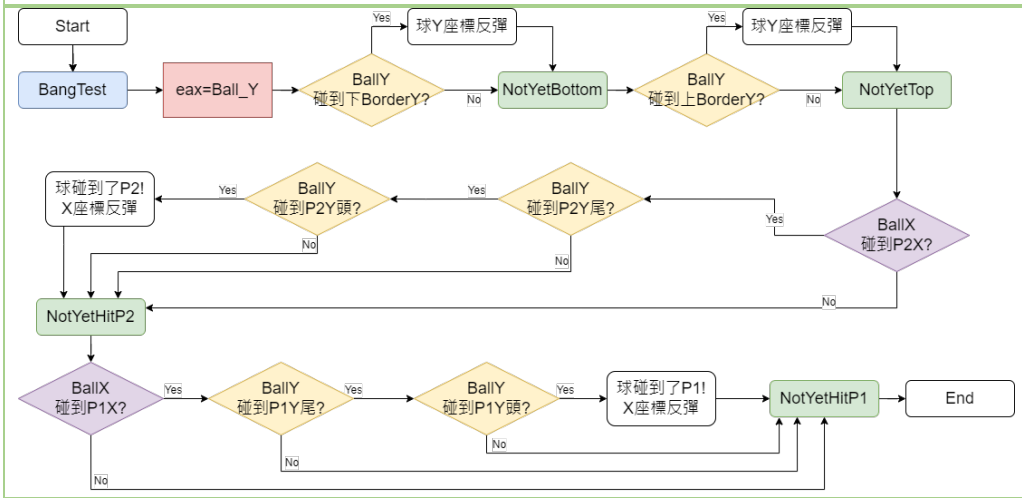
_ball.asm架構圖 - 1



_ball.asm架構圖 - 2 (Reset label裡)



_ball.asm架構圖 - 3 (Bang Test label裡)



Ball.asm應該是註解最多的程式碼了，因為每個指令實在太過相似(一直在比較XY座標)，如果不畫流程圖輔助一不小心就會亂掉(這時也了解了為什麼要有pseudo code)。

- **BangTest和processing程式碼的比較**

Processing

```
void bounce()
{//+-16是因為要讓球邊碰到板子時就要反彈，球設定直徑32
  //如果球球左邊撞到非背景色(aka板子和板子以外的畫面)
  if (get(int(x)-16, int(y)) != bg){
    ball_x = true;//球球x座標動，往右繼續
  }
  if (get(int(x)+16, int(y)) != bg){
    ball_x = false;//球球x座標動，往左繼續
  }
  //撞到藍色(DMode)或是紅色(EMode)回彈
  if (get(int(x), int(y)-16) == redC || get(int(x), int(y)-16) == blueC){
    ball_y = false;//球球y座標動，往下繼續
  }
  if (get(int(x), int(y)+16) == redC || get(int(x), int(y)+16) == blueC){
    ball_y = true;//球球y座標動，往上繼續
    changespeed+=1.0/4;//撞到顏色板子速度加快(越來越快~)
  }
}
```

因為processing(類似於Java)我們是用抓當下球的座標有沒有碰到指定顏色來決定球的反彈走向，判斷較為單純，只需檢查上下左右座標再用boolean控制球的瞬間走向即可。

組語(以NotYetTop為例子)

```
NotYetTop:
    ;這裡比較是否撞到右邊paddle，X座標
    ;右板子往內3小格(視覺上差異)
    mov eax, p2X
    mov ebx, eax
    sub ebx,3
    ;球和座標比較
    mov eax, [Ball_X]
    cmp [eax], ebx
    jb NotYetHitP2          ;若小於，代表球X座標還沒飛到

    ;大於，可能碰到板子了
    ;接著判斷Y座標
    mov eax, p2Y
    mov ebx, eax
    mov eax, [Ball_Y]
    cmp [eax], ebx          ;若大於，代表球Y座標還沒飛到
    ja NotYetHitP2          ;也就是說此時球比板子還要下面

    ;小於，可能碰到板子了
    ;接著判斷有沒有在板子大小範圍內
    ;由下往上數5格內才算碰到
    mov ebx, p2Y            ;板子Y座標傳入ebx
    sub ebx, paddleHeight
    inc ebx                 ;板子最高點座標
    mov eax, [Ball_Y]       ;球Y座標EAX
    cmp [eax], ebx          ;比較，若再小於，代表球的位置比板子高(更上方)
    jb NotYetHitP2          ;沒碰到右板子，跳去[還沒碰到右板子]
    ;大於了，代表碰到右側，更改xMove
    mov eax, [xMove]
    neg dword ptr [eax]     ;變負數往反方向回彈
```

先檢查球當下X座標有沒有和右邊板子X座標相符。
若小於
跳至"還沒撞到右板子"
若大於

代表可能碰到右側板子
接著抓球當下的Y座標，因為板子為由下往上畫，所以是和板子最下方座標比較
若小於

代表可能碰到右側板子
接著抓球當下的Y座標和板子最高點Y座標進行比較
若大於，此時代表球的XY座標都有符合碰到板子的三個條件。

- **Ball.asm之Ball Movement**

  和Paddle的移動一樣，球移動到下一點時，它的上一個圖形所在位置要用黑色蓋掉。

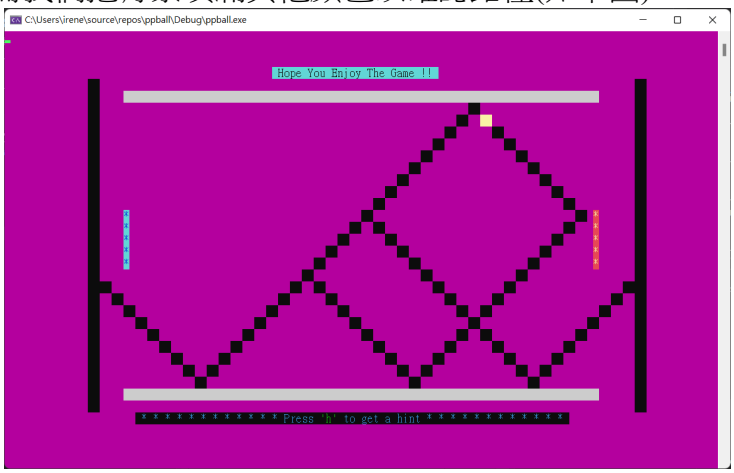
遮掉

```
BallMovement:
    ;遮掉當下的球
    mov eax, 0                  ;塗黑
    call SetTextColor
    ;要遮掉的座標位置
    mov eax, [Ball_X]
    mov dl, [eax]
    mov eax, [Ball_Y]
    mov dh, [eax]
    call Gotoxy
    mov edx, buffer
    call WriteString            ;畫出
```

繪製新的球

```
    ; 更新球的XY座標，畫新的球
    ; X座標
    mov eax, xMove          ;2
    mov ebx, [eax]          ;ebx = xMove
    mov eax, [Ball_X]       ;eax = Ball_X
    add ebx, [eax]          ;X當下座標加上2
    mov [eax], ebx          ;新的X座標
    mov dl, [eax]           ;Gotoxy
    ; Y座標
    mov eax, yMove          ;1
    mov ebx, [eax]          ;1
    mov eax, [Ball_Y]       ;Y當下座標
    add ebx, [eax]          ;Y當下座標加上1
    mov [eax], ebx          ;新的Y座標
    mov dh, [eax]           ;Gotoxy

    ;傳入新座標，印出新球球
    mov eax, color          ;黃色
    call SetTextColor
    call Gotoxy             ;新球球座標定出
    mov edx, buffer
    call WriteString        ;印出新球球
```

而我們把背景填滿其他顏色以確認路徑(如下圖):



- **球的速度和Reset Time**

  球的速度較為單純，我們使用Irvine library裡的call delay來達成我們想要的效果

ppball.asm

```
;球以每0.1秒速度移動
FRAME_RATE equ 100d             ;球跑的速度

;reset時間等待1.5秒
RESET_BALL_RATE equ 1500d       ;球reset
```

SpeedOfBall & newBallTime

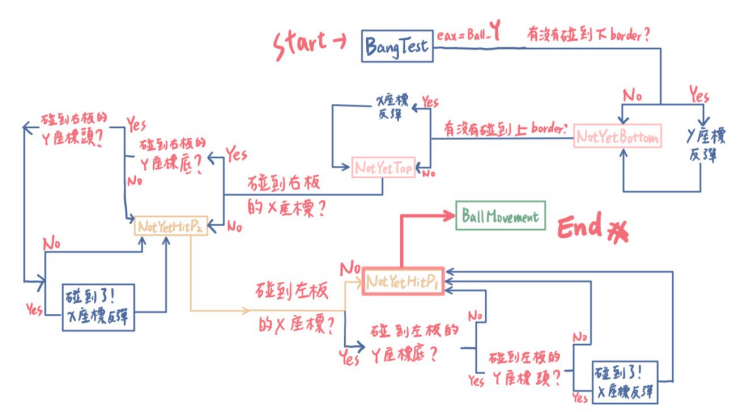
```
; SpeedOfBall
include ppball.inc

.code
SpeedOfBall PROC,
    timedu: dword

    pushad

    mov eax, timedu
    call Delay

    popad
    ret
SpeedOfBall endp
end

;text 5.2.1
```

```
; newBallTime
include ppball.inc

.code
NewBallTime PROC,
    revive: dword

    pushad

    mov eax, revive
    call Delay

    popad
    ret
NewBallTime endp
end

;附註 5.2.1(課本)
```

❖ **遇到問題/心得**

- **第一次畫流程圖!!**

  因為Ball的程式碼其實是我們裡面要考慮的點最多的，也是最複雜的一個，流程圖也是我們在討論的過程中一點一滴畫出來的，原本長這樣:



很醜，不過看得懂就好哈哈哈

- **球會在板子上抖動的Bug**

  一開始我們並沒有注意到有這個Bug(不過這個狀況有點難形容，除非在玩的過程中碰到)，後來重新審視了一下程式碼，發現一樣是數學問題哈哈，因為在判斷球是否碰到paddle且在paddle 5格的範圍內時，paddle座標底在減掉5時，要再加上1才會是正確的座標，雖然不太確定是不是這個問題但改了之後就沒再看到這個Bug了。

- **心得**

  Ball的部分就是無止盡的debug…只要腦袋一時不清楚，ja、jb條件就會不小心給錯，所以我們這部分程式碼是兩個人利用放學都在學校的時間，一個負責講出邏輯順序，另一個負責一邊打出來程式碼的。

- **口頭報告後的修改1** (新增隱藏的黑板子，在ShowFirstTwoPaddles.asm裡)
  除此之外我們一開始口頭報告時，當時的程式碼有球飛到邊界外不會消失的bug，我們在報告後用最直接的方式蓋掉球，就是在兩側新增黑色的板子～
  新增的變數:

ppball.asm

```
8   BOARD_BETWEEN equ 25d
9
10  ;新增左右兩側看不到的黑色boards
11  PLAY_BLACK_X1 equ 14d
12  PLAY_BLACK_Y1 equ 4d
13  PLAY_BLACK_X2 equ 104d
14  PLAY_BLACK_Y2 equ 4d
15
```

_ShowFirstTowPaddle.asm

```
6   ShowFirstTwoPaddles proc,
7       color: dword,
8      color2: dword,
9      blackX1: dword,                  ;PLAY_BLACK_X1
10     blackY1: dword,                  ;PLAY_BLACK_Y1
11     blackX2: dword,                  ;PLAY_BLACK_X2
12     blackY2: dword,                  ;PLAY_BLACK_Y2
13     p1X: PTR dword                   ; x座標
```

同樣的，我們也用Loop方式繪製，先印出有顏色的結果:

藍色的兩條border就是ball 在reset前停留的地方。

- ❖ **程式碼截圖**

左邊

```
21      ; 左黑board初始
22      pushad
23      ;(14,4)起始 長28
24      mov ebx, blackX1
25      mov dl, bl
26      mov eax, blackY1
27      mov dh, al
28      mov ecx, 28
```

右邊

```
45      ; 右黑board初始
46      pushad
47      ;(106,4)起始 長28
48      mov ebx, blackX2
49      mov dl, bl
50      mov eax, blackY2
51      mov dh, al
52      mov ecx, 28
```

BlackBoardLeft: 28次

```
29  BlackBoardLeft:
30      call Gotoxy
31
32      mov eax, (black*16)
33      call SetTextColor
34
35      push edx
36      mov edx, OFFSET space2      ;space2 "  "
37      call WriteString
38      pop edx
39      inc dh                      ;y座標加一，由上往下拼
40      loop BlackBoardLeft
41      popad
42      ; 結束BlackBoardLeft
43
```

Loop BlackBoardLeft

BlackBoardRight: 28次

```
53  BlackBoardRight:
54      call Gotoxy
55
56      mov eax, (black*16)
57      call SetTextColor
58
59      push edx
60      mov edx, OFFSET space2
61      call WriteString
62      pop edx
63      inc dh          ;往下畫，Y座標加一
64      loop BlackBoardRight
65      popad
66      ; 結束BlackBoardRight
```

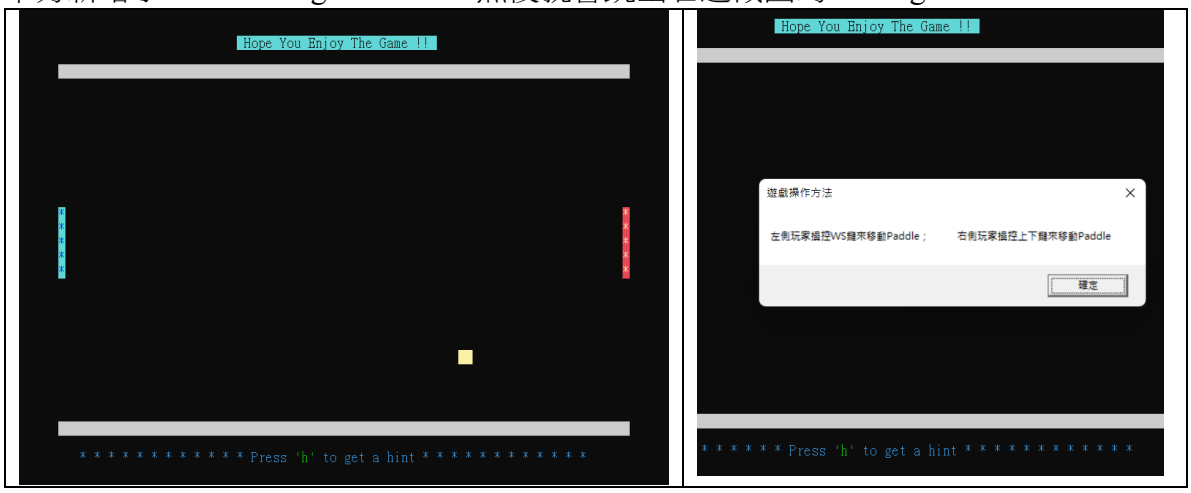
Loop BlackBoardRight:

然後就成功了~~~ 🥳 🥳 🥳

- **口頭報告後的修改2** (新增遊戲提示鍵MsgBox)

  因為在看大家的口頭報告時，看到有幾個組有用到Message Box 的功能，於是我們在我們遊戲畫面裡加了一個按下h鍵即可查看操作提示。

  下方新增了”Press h to get a hint”，然後就會跳出右邊截圖的Message Box。



- **口頭報告後的修改3** (球改成笑臉)

  很遺憾的，我們最後球還是沒有成功改成笑臉，我們也有下載第十一章作者的範例檔案來執行，不知道是不是設定上的問題，就算跑了課本的範例程式碼依然出不來，所以最後就繼續維持方形的球(?)了。

Conclusion (required):

- 結論/心得

  終於又完成一份期末專題了，這是我們兩個第二次一起完成的專題，好險在這學期的其他選修有先試過水溫，對於分工打code比較有概念，加上利用比較熟悉的遊戲來成為我們這次組語專題的主題，一開始對於整體遊戲架構就有一點幫助。

  在這次組語專題中我們遇到的最大問題是太晚開始做，因為時間接近期末，剛好又是各項考試以及專題的發表，最後一個月每天神經都很緊繃，我們在這次的組語專題中學到非常多，除了分工合作，流程圖、pseudo code都是在這次也學到的新東西！當然最主要還是對組合語言有更深一層的認識，一開始我們覺得組語的起步對我們來說有點困難，但其實在寫作業、問問題、和同學討論間我們也越來越進步，而專題的完成帶給我們非常大的成就感！

  另外我們印象蠻深刻的事情是，因為兩個人打程式的邏輯多少會有些不同，在開始寫組語專題時，我們雖然有先討論好整體架構，但剛開頭仍有些不順，加上要一邊寫專題一邊學新東西，不管是看影片還是查資料，要同時達到現學現賣非常花功夫和力氣，但當看到我們立定的目標一個一個完成時，每次小目標的達成都促使我們更有動力完成下一件目標，就這樣進入debug、抓錯、debug、修改的無限循環，最終成品出來時感到異常的滿足！

- 還可以改進的地方

  我們的 PPBall 專題比較可惜的部分是沒有完成計分功能，最主要還是下次專題還是早點開始做，因為靈感和想法通常都會在妳動手開始實行時一一冒出來，沒把遊戲做到更完整這部分有些可惜。

  另外就是我們口頭報告時影片上傳失敗的狀況，學到了一次教訓，當我們看到好幾位同學反應沒有 DEMO 影片時，評分的那幾天真的是天天提心吊膽啊!!

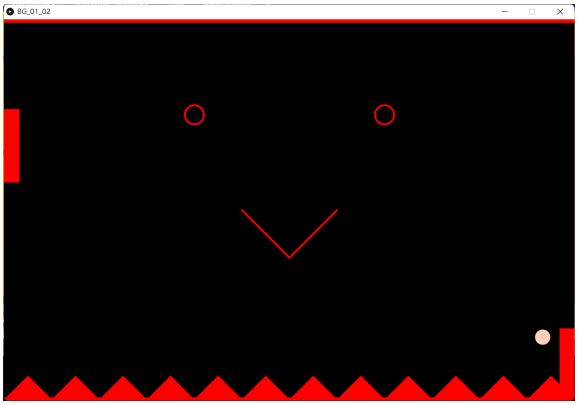
Team Work Arrangement:

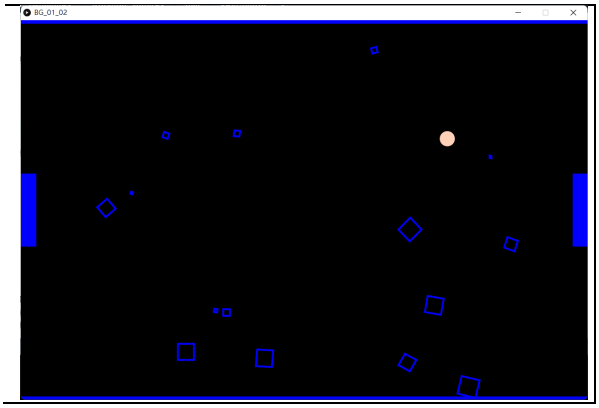
各個檔案	張甄庭	用顏色區分	鍾馥羽
1. Main檔			
ppball.asm	70		30
ppball.inc	65		35
2. Menu			
_Menu.asm	60		40
3. Frame & Paddles			
_drawframe.asm	70		30
_showFirstTwoPaddles.asm	40		60
_ReadKeyboard.asm	60		40
4. Ball			
_ball.asm	35		65
_resetball.asm	100		
_newBallTime.asm _SpeedOfBall.asm	100		
5. 口頭報告			
PPT	30		70
口頭報告	45		55
影片處理	20		80
6. 書面報告			
書面報告	65		35
7. 其他			
GUI介面設計	70		30
查資料	40		60
參與度	200		200

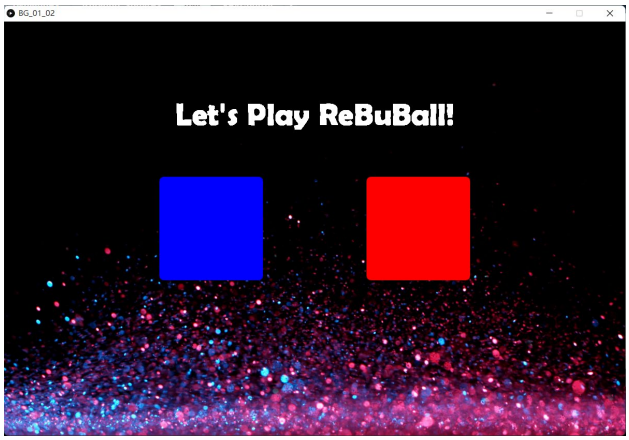
其他:

- **最新DEMO**
  - **DEMO影片連結**
    https://drive.google.com/file/d/1hh-G8bGanC7R2ivdkvmo5pf2ch-rDcCL/view?usp=sharing
  - 介面截圖

- Processing的專題來露臉一下






BG_01_02
Let's Play ReBuBall!