微處理機系統實習　　Lab5

班級：資訊三丁 學號：D1009212 姓名：邱柏宇

一、【實驗目的】：

What was your design? What were the concepts you have used for your design?

第一題

移動方塊是使用 draw_Bmp16x8 和 draw_Bmp8x16 來畫出方塊，

然後用 is_vertical 來記錄現在方塊是否是垂直的（是不是立起來）來決定要用哪個印出方式和碰到邊界的距離判定，另外為了避免在邊緣時改變 is_vertical 導致超出邊界，用 dirX 和 dirY 去計算 x 和 y 要怎麼移動，再繪製即可。

第二題

把場地（格子）、選擇時的閃爍、現在遊戲狀態（已經劃下去的 OX）分圖層，這樣一來誰覆蓋誰就可以比較容易控制。為了優化閃爍的問題，我將 LCD 和 draw2D 進行重構，創建 lcd_buffer_hex 和 lcd_buffer_hex_last 來記錄現在和上次顯示狀態，每次迴圈都會比較兩個 buffer 如果出現不相同情況會用 lcdSetAddr 和 lcdWriteData 來寫 lcd，如此一來就可以減少原生 LCD.h 和 Draw2D.h 沒必要的寫入 lcd 。判斷誰輸誰贏用窮舉的方式寫出 8 種可能（123, 456, 789, 147, 258, 369, 753, 159）直接比較。

二、【遭遇的問題】：

What problems you faced during design and implementation?

16 進制的 buffer 比較不直覺。

三、【解決方法】：

How did you solve the problems?

創建類似 java 的 set get 來存取 buffer，不直接對 buffer 操作。

四、【未能解決的問題】：

Was there any problem that you were unable to solve? Why was it unsolvable?

判斷輸贏的邏輯有點笨，如果時間允許或是題目更加複雜會使用 dfs 來解決。

五、【投影片中的問題】：

1. 程式中的<<是甚麼意思? 假設今天 k 是 3 時 0X01<<k 之後回傳的答案是多少?

<< 是左移運算子

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x01 | << | 3 | | | | | | | |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | | |
| | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = | 1 |
| | | 往左偏移 3 格 | | | | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | = | 8 |

還有下一頁

2. fgColor 跟 bgColor 是甚麼意思?分別用在哪邊?()

Lcd 的顯示原理是於兩片玻璃間注入液晶，然後再施以電極使其液晶分子產生旋轉角度變化，因而影響其透光性，達到明暗的變化來達到顯示的效果，所以事實上的「顯示」其實是用明暗的差別來做到，在 LCD.h 中有 #define FG_COLOR 0xFFFF 和 #define BG_COLOR 0x0000，但其實 LCD.c 中並沒有使用的，透過 drawPixel 可以看到 bgColor 這個參數其實並沒有用處，真正在判斷只有 fgColor,如果單純地改變 define 是不會達到相反的效果,得從 lcdWriteData 時，數值相反過來才有用。

```
void draw_Pixel(int16_t x, int16_t y, uint16_t fgColor, uint16_t bgColor)
{
    if (fgColor!=0)
        DisplayBuffer[x+y/8*LCD_Xmax] |= (0x01<<(y%8));
    else
        DisplayBuffer[x+y/8*LCD_Xmax] &= (0xFE<<(y%8));

    lcdSetAddr(y/8,(LCD_Xmax+1-x));
    lcdWriteData(DisplayBuffer[x+y/8*LCD_Xmax]);
}
```

這邊改寫 lcdWriteData 來驗證，加上 temp = 0xFF - temp，來做到。

```
// Wrtie data to LCD
void lcdWriteData(uint8_t temp) {
    temp = 0xFF - temp;
    SPI_SET_SS0_LOW(SPI3);
    SPI_WRITE_TX0(SPI3, 0x100 + temp);  // Write Data
    SPI_TRIGGER(SPI3);                  // Trigger SPI data transfer
    while (SPI_IS_BUSY(SPI3))
        ;  // Check SPI3 busy status
    SPI_SET_SS0_HIGH(SPI3);
}
```