微處理機系統實習　　Lab7

班級：資訊三乙 學號：D1210799 姓名：王建葦

一、【實驗目的】：

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

本次實驗包含兩個主題：

- Lab7.1 — 單向移動球體系統
  - 本程式設計一個可控制的球體在 LCD 上由左向右移動的動畫。
  - 以 LCD 為主要顯示介面，搭配 Keypad 三鍵控制運動：
    - 按鍵 1 → 開始運動（球從左側起始）
    - 按鍵 2 → 暫停
    - 按鍵 3 → 繼續運動
  - 球體每次移動固定距離（8 pixels），到達最右邊界時觸發蜂鳴器響 0.1 秒，並重置位置。
  - 本設計目的為練習 LCD 畫面繪圖（draw_Circle）、Keypad 輸入偵測、延遲控制（SysTickDelay）與 蜂鳴器控制（GPIO active-low）的綜合應用。

- Lab7.2 — 彈跳球體與方塊碰撞遊戲
  - 此程式模擬一個可控制的「彈跳球體」，可與兩個隨機生成的方塊互動：
    - 玩家使用 Keypad 控制方向（左、右、斜上、斜下等）移動球體。
    - 球體與方塊碰撞時，方塊消失；兩方塊皆被擊中後球體回歸初始位置。
    - 按鍵 8 可重新產生方塊組合。
  - 程式中採用 AABB 碰撞檢測法（Axis-Aligned Bounding Box）以簡化圓形與方形之間的碰撞計算，同時實作 邊界反彈邏輯，讓球體在 LCD 範圍內持續彈跳運動。

二、【遭遇的問題】：

What problems you faced during design and implementation?

1. Lab7.1 中，當多次快速按鍵操作時，球體會顯示異常或跳躍位置。
2. 球體清除與重繪時，LCD 閃爍明顯，導致動畫不流暢。
3. 蜂鳴器聲音太短或未觸發，因 GPIO 初始化時序不穩定。
4. Lab7.2 初期方塊生成太接近，導致兩個方塊重疊或球體初始即發生碰撞。
5. 球體在邊界反彈時，有機率卡在邊緣無法繼續移動。
6. Keypad 連續按壓會重複觸發多次方向變化，導致方向錯亂。
7.

三、【解決方法】：

How did you solve the problems?

1. 在 Keypad 控制中增加「按鍵釋放偵測」邏輯，只在按鍵由無到有變化時觸發動作，避免長按重複輸入。
2. 將球體更新流程分為「清除 → 計算 → 重繪」三步驟，並設定適當延遲（約 50 ms），減少畫面閃爍。
3. 蜂鳴器初始化改用 GPIO_SetMode(PB, BIT11, GPIO_PMD_OUTPUT) 並先設高電位，確保主

程式執行前狀態正確。

4. 在方塊生成函式 GenerateTwoBlocks() 中引入最小間距 MIN_DISTANCE=20，防止重疊並分布於螢幕不同區域。
5. 於邊界偵測中限制座標範圍（new_x <= r, new_x >= 127-r），並即時反轉方向變數 dirX/dirY 形成反彈效果。
6. 每次更新後重繪仍可見的方塊，防止球體移動時「擦掉」其他方塊。
7. 將球體重置與畫面清除封裝為流程，確保遊戲在方塊消失後自動回復初始狀態。

在找尋這些問題的解決方法與問題點時，我有使用 ChatGPT 協助我找尋與解決問題。包含 實驗結報的內容修改與潤飾都有使用 ChatGPT 協助。

四、【未能解決的問題】：

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

1. 在 Lab7.1 中若系統延遲太短，LCD 刷新速度會高於顯示緩衝能力，仍偶有微閃現象。
2. Lab7.2 的蜂鳴器功能 Buzz() 尚未實作，未能在碰撞時提供聲響反饋。
3. 球體在連續對角反彈時偶爾發生「邊界跳動」現象，可能與邊界判斷的 ±1 像素誤差有關。
4. 方塊隨機生成基於簡化的 LCG，長時間運行仍可能出現相似位置。
5. 畫面未採雙緩衝（double buffering），導致圖像變化仍存在輕微閃爍。

五、【程式碼】：

Lab 7.1:

```c
//
// Lab7-1: 單向移動球體系統
// 功能：球體從左向右移動，可透過按鍵控制開始、暫停、繼續
// EVB : Nu-LB-NUC140
// MCU : NUC140VE3CN
//
#include <stdio.h>
#include <math.h>
#include "NUC100Series.h"
#include "MCU_init.h"
#include "SYS_init.h"
#include "LCD.h"
#include "Scankey.h"
#include "Draw2D.h"

// LCD顯示器尺寸定義（寬度128像素，高度64像素）
#define LCD_W 128
#define LCD_H 64

// 球體相關參數定義
#define RADIUS 4       // 球體半徑（像素）
#define STEP_X 8       // 每次移動的X軸步進距離（像素），每次移動2個半徑的距離
#define TICK_US 500000 // 每次移動之間的延遲時間（微秒），約0.5秒

// --- 蜂鳴器控制相關定義（PB11腳位，低電位觸發） ---
#define BUZZ_PIN_MASK BIT11 // 蜂鳴器控制腳位遮罩（PB11）

/**
 * 蜂鳴器初始化函數
 * 功能：設定PB11為輸出模式並初始化為關閉狀態
 * 說明：PB11 = 1 表示蜂鳴器關閉（低電位觸發型蜂鳴器）
 */
static void BUZZ_Init(void)
{
    GPIO_SetMode(PB, BUZZ_PIN_MASK, GPIO_PMD_OUTPUT); // 設定PB11為輸出模式
    PB11 = 1;                                          // 設定為高電位，關閉蜂鳴器（因為是Active-Low設計）
}

/**
 * 蜂鳴器響聲控制函數
 * @param us 蜂鳴器響聲持續時間（微秒）
 * 功能：讓蜂鳴器響聲指定時間長度
 * 流程：開啟蜂鳴器 → 延遲指定時間 → 關閉蜂鳴器
 */
static void BUZZ_Beep(uint32_t us)
{
    PB11 = 0;             // 設定為低電位，開啟蜂鳴器
    CLK_SysTickDelay(us); // 延遲指定時間（微秒）
    PB11 = 1;             // 設定為高電位，關閉蜂鳴器
}
```

```
52      /**
53       * 繪製球體函數
54       * @param x 球心X座標
55       * @param y 球心Y座標
56       * @param color 球體顏色（FG_COLOR或BG_COLOR）
57       * 功能：在指定位置繪製一個圓形球體
58       * 說明：使用draw_Circle函數繪製，背景色為BG_COLOR以便清除
59       */
60      static void draw_ball(int x, int y, uint16_t color)
61      {
62          draw_Circle(x, y, RADIUS, color, BG_COLOR); // 繪製圓形，背景色用於清除
63      }
```

```c
/**
 * 主程式入口
 * 功能：實作球體單向移動系統，支援開始、暫停、繼續控制
 */
int main(void)
{
    // --- 系統狀態變數 ---
    int active = 0; // 球體運動啟用標誌（0=未啟用，1=已啟用）
    int paused = 0; // 球體運動暫停標誌（0=執行中，1=暫停）

    // --- 球體位置變數 ---
    int cx = RADIUS;                          // 當前球體圓心X座標，初始值為半徑（從左邊緣開始）
    const int cy = 32;                        // 球體圓心Y座標（固定在中間位置，LCD高度64/2=32）
    const int max_cx = (LCD_W - 1) - RADIUS; // 球體能移動的最大X座標（右邊界，避免超出螢幕）

    // --- 系統初始化 ---
    SYS_Init();   // 系統初始化（時鐘、GPIO等基本設定）
    init_LCD();   // LCD顯示器初始化
    clear_LCD();  // 清除LCD螢幕內容
    OpenKeyPad(); // 按鍵掃描功能初始化
    BUZZ_Init();  // 蜂鳴器初始化

    // --- 主程式迴圈 ---
    while (1)
    {
        int next_cx;         // 下一次移動後的預期X座標
        int key = ScanKey(); // 掃描按鍵狀態，返回值：0=無按鍵，1/2/3=對應按鍵

        // --- 按鍵處理邏輯 ---
        // 按鍵1：開始球體運動（僅在未啟用時有效）
        if (key == 1)
        {
            if (!active)
            {                                 // 只有在球體未啟用時才執行
                cx = RADIUS;                  // 重置球體位置到左邊緣
                draw_ball(cx, cy, FG_COLOR); // 在起始位置繪製球體（前景色）
                active = 1;                   // 標記為已啟用
                paused = 0;                   // 標記為非暫停狀態
            }
        }
        // 按鍵2：暫停球體運動（僅在已啟用時有效）
        else if (key == 2)
        {
            if (active)
                paused = 1; // 如果球體已啟用，則設定為暫停狀態
        }
        // 按鍵3：繼續球體運動（僅在已啟用時有效）
        else if (key == 3)
        {
            if (active)
                paused = 0; // 如果球體已啟用，則取消暫停狀態
        }
```

```
118             // --- 球體運動處理邏輯 ---
119             // 只有在球體已啟用且未暫停時才執行運動
120             if (active && !paused)
121             {
122                 CLK_SysTickDelay(TICK_US); // 延遲0.5秒，控制移動速度
123 
124                 // 第一步：清除當前位置的球體（用背景色繪製，相當於清除）
125                 draw_ball(cx, cy, BG_COLOR);
126 
127                 // 第二步：計算下一位置並判斷是否超出邊界
128                 next_cx = cx + STEP_X; // 計算下一次移動後的X座標（向右移動STEP_X像素）
129 
130                 // 判斷是否還在螢幕範圍內
131                 if (next_cx <= max_cx)
132                 {
133                     // 仍在範圍內：更新位置並繪製新球體
134                     cx = next_cx;                // 更新當前X座標
135                     draw_ball(cx, cy, FG_COLOR); // 在新位置繪製球體
136                 }
137                 else
138                 {
139                     // 已超出右邊界：執行結束流程
140                     cx = max_cx;                 // 將座標限制在最大邊界位置
141                     draw_ball(cx, cy, FG_COLOR); // 在邊界位置繪製球體
142 
143                     // 觸發蜂鳴器響聲，提示已到達終點
144                     BUZZ_Beep(100000); // 響聲0.1秒（100000微秒）
145 
146                     // 清除球體並重置狀態
147                     draw_ball(cx, cy, BG_COLOR); // 清除球體顯示
148                     active = 0;                  // 標記為未啟用
149                     paused = 0;                  // 清除暫停狀態
150                 }
151             }
152             // 當球體未啟用或處於暫停狀態時
153             else
154             {
155                 CLK_SysTickDelay(10000); // 短延遲10毫秒，降低CPU使用率
156             }
157         }
158     }
```

Lab 7.2:

```c
//
// Lab7-2: 彈跳球體與目標方塊碰撞遊戲
// 功能：控制球體移動，碰撞目標方塊使其消失，所有方塊消失後重置
// EVB : Nu-LB-NUC140
// MCU : NUC140VE3CN
//
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "NUC100Series.h"
#include "MCU_init.h"
#include "SYS_init.h"
#include "LCD.h"
#include "Draw2D.h"
#include "Scankey.h"

// 像素狀態定義
#define PIXEL_ON 1  // 像素開啟（顯示）
#define PIXEL_OFF 0 // 像素關閉（隱藏）

// 球體初始位置定義
#define X0 64    // 球體初始X座標（螢幕中央，LCD寬度128/2=64）
#define Y0 60    // 球體初始Y座標（底部區域，從原本的32改為60）
#define RADIUS 3 // 球體半徑（像素）

// 目標方塊相關定義
#define BLOCK_SIZE 5    // 目標方塊尺寸（5x5像素）
#define MIN_DISTANCE 20 // 兩個方塊之間的最小水平距離（像素），避免方塊過於接近

/**
 * 蜂鳴器響聲控制函數（目前為預留功能，程式碼被註解）
 * @param number 響聲次數（目前未使用）
 * 功能：控制蜂鳴器發出響聲
 * 說明：此函數預留給未來擴展使用，目前函數體內程式碼已被註解
 */
void Buzz(int number)
{
    int i;
    // for (i = 0; i < number; i++) {
    // PB11 = 0; // PB11 = 0 時開啟蜂鳴器
    // CLK_SysTickDelay(100000);  // 延遲
    // PB11 = 1; // PB11 = 1 時關閉蜂鳴器
    // CLK_SysTickDelay(100000);  // 延遲
    //}
}
```

```c
/**
 * 產生兩個隨機位置目標方塊的函數
 * @param block1_x 方塊1的X座標指標（輸出參數）
 * @param block1_y 方塊1的Y座標指標（輸出參數）
 * @param block2_x 方塊2的X座標指標（輸出參數）
 * @param block2_y 方塊2的Y座標指標（輸出參數）
 * @param seed 隨機數種子指標（輸入/輸出參數）
 * 功能：使用線性同餘生成器產生兩個隨機位置的方塊，確保它們之間至少有MIN_DISTANCE的距離
 * 演算法：使用LCG（Linear Congruential Generator）偽隨機數產生器
 * 公式：seed = (seed * 1103515245 + 12345) & 0x7FFFFFFF
 */
void GenerateTwoBlocks(int16_t *block1_x, int16_t *block1_y, int16_t *block2_x, int16_t *block2_y, uint32_t *seed)
{
    int16_t distance; // 兩個方塊之間的水平距離

    // 產生第一個方塊的X座標
    // 使用LCG演算法更新種子值，確保每次產生的值都不同
    *seed = (*seed * 1103515245 + 12345) & 0x7FFFFFFF;
    // 限制X座標範圍在2~125之間（避免方塊超出螢幕或貼邊）
    // 公式：(seed % (最大值-最小值+1)) + 最小值
    *block1_x = (*seed % (125 - 2 + 1)) + 2;

    // 產生第一個方塊的Y座標
    // 再次使用LCG演算法更新種子值
    *seed = (*seed * 1103515245 + 12345) & 0x7FFFFFFF;
    // 限制Y座標範圍在2~29之間（上半部區域，避免與球體初始位置重疊）
    *block1_y = (*seed % (29 - 2 + 1)) + 2;

    // 產生第二個方塊，並確保與第一個方塊的距離足夠
    do
    {
        // 產生第二個方塊的X座標
        *seed = (*seed * 1103515245 + 12345) & 0x7FFFFFFF;
        *block2_x = (*seed % (125 - 2 + 1)) + 2;

        // 產生第二個方塊的Y座標
        *seed = (*seed * 1103515245 + 12345) & 0x7FFFFFFF;
        *block2_y = (*seed % (29 - 2 + 1)) + 2;

        // 計算兩個方塊之間的水平距離（絕對值）
        if (*block2_x > *block1_x)
        {
            distance = *block2_x - *block1_x; // block2在右邊，計算距離
        }
        else
        {
            distance = *block1_x - *block2_x; // block2在左邊，計算距離
        }

    } while (distance < MIN_DISTANCE); // 如果距離小於最小距離要求，重新產生直到滿足條件
}
```

```
99      /**
100      * 檢查圓形球體與方形方塊是否發生碰撞
101      * @param cx 圓心X座標
102      * @param cy 圓心Y座標
103      * @param cr 圓的半徑
104      * @param bx 方塊中心X座標
105      * @param by 方塊中心Y座標
106      * @param bsize 方塊尺寸
107      * @return 1=發生碰撞，0=未發生碰撞
108      * 功能：使用軸對齊邊界框（AABB - Axis-Aligned Bounding Box）碰撞檢測方法
109      * 說明：將圓形視為正方形邊界框進行簡化的碰撞檢測，提高計算效率
110      * 原理：檢測兩個矩形邊界框是否重疊，比精確的圓形-矩形碰撞檢測更快
111      */
112     int CheckOverlap(int16_t cx, int16_t cy, int16_t cr, int16_t bx, int16_t by, int16_t bsize)
113     {
114         int16_t circle_left, circle_right, circle_top, circle_bottom; // 圓形邊界框座標
115         int16_t block_left, block_right, block_top, block_bottom;     // 方塊邊界框座標
116         int16_t half_size;                                            // 方塊半邊長度
117 
118         half_size = bsize / 2; // 計算方塊半邊長度（用於計算邊界範圍）
119 
120         // 計算圓形的邊界框（將圓形視為正方形，以圓心為中心，邊長為2*半徑）
121         circle_left = cx - cr;   // 左邊界 = 圓心X - 半徑
122         circle_right = cx + cr;  // 右邊界 = 圓心X + 半徑
123         circle_top = cy - cr;    // 上邊界 = 圓心Y - 半徑
124         circle_bottom = cy + cr; // 下邊界 = 圓心Y + 半徑
125 
126         // 計算方塊的邊界框（以方塊中心為基準，向四周擴展半邊長度）
127         block_left = bx - half_size;   // 左邊界 = 方塊中心X - 半邊長
128         block_right = bx + half_size;  // 右邊界 = 方塊中心X + 半邊長
129         block_top = by - half_size;    // 上邊界 = 方塊中心Y - 半邊長
130         block_bottom = by + half_size; // 下邊界 = 方塊中心Y + 半邊長
131 
132         // AABB碰撞檢測：檢查兩個矩形邊界框是否重疊
133         // 兩個矩形重疊的條件：
134         // 1. 圓形右邊界 >= 方塊左邊界 且 圓形左邊界 <= 方塊右邊界（水平方向重疊）
135         // 2. 圓形下邊界 >= 方塊上邊界 且 圓形上邊界 <= 方塊下邊界（垂直方向重疊）
136         if (circle_right >= block_left &&
137             circle_left <= block_right &&
138             circle_bottom >= block_top &&
139             circle_top <= block_bottom)
140         {
141             return 1; // 發生碰撞
142         }
143 
144         return 0; // 未發生碰撞
145     }
```

```
147     /**
148      * 主程式入口
149      * 功能：實作彈跳球體與目標方塊碰撞遊戲
150      * 遊戲規則：
151      * 1. 使用按鍵控制球體移動方向
152      * 2. 球體碰撞到目標方塊時，方塊消失
153      * 3. 當所有方塊都消失時，球體重置到初始位置
154      * 4. 按鍵8可在方塊都消失後重新產生新的方塊
155      */
156     int32_t main(void)
157     {
158         // --- 球體運動參數 ---
159         int dirX, dirY;       // 球體移動方向（-1/0/1分別表示負方向/停止/正方向）
160         int movX, movY;       // 球體每次移動的距離（像素）
161         uint16_t r;           // 球體半徑
162         int16_t x, y;         // 球體當前位置座標
163         int16_t new_x, new_y; // 球體下一個位置的座標
164 
165         // --- 顯示顏色定義 ---
166         uint16_t fgColor, bgColor; // 前景色和背景色
167 
168         // --- 按鍵處理變數 ---
169         uint8_t keyin, last_key; // 當前按鍵值和上一個按鍵值（用於檢測按鍵釋放，避免重複觸發）
170 
171         // --- 遊戲狀態變數 ---
172         int is_moving; // 球體是否正在移動（0=停止，1=移動中）
173         int bounced;   // 是否發生邊界反彈（0=未反彈，1=已反彈）
174 
175         // --- 目標方塊相關變數 ---
176         int16_t block1_x, block1_y, block2_x, block2_y; // 兩個方塊的位置座標
177         int block1_visible, block2_visible;             // 兩個方塊的可見性標誌（0=已消失，1=可見）
178 
179         // --- 隨機數相關變數 ---
180         uint32_t seedCounter; // 隨機數種子計數器
181 
182         // --- 碰撞檢測變數 ---
183         int overlap; // 是否發生碰撞（0=未碰撞，1=已碰撞）
184 
185         // --- 迴圈計數變數 ---
186         int16_t i, j; // 用於繪製方塊的迴圈計數變數
187 
188         // --- 初始化遊戲狀態變數 ---
189         last_key = 0;       // 初始化上一個按鍵值為0（無按鍵）
190         is_moving = 0;      // 初始化為停止狀態（球體不會自動移動）
191         block1_visible = 1; // 方塊1初始為可見狀態
192         block2_visible = 1; // 方塊2初始為可見狀態
193         seedCounter = 0;    // 初始化隨機數種子計數器
194 
195         // --- 系統初始化 ---
196         SYS_Init();   // 系統初始化（時鐘、GPIO等基本設定）
197         init_LCD();   // LCD顯示器初始化
198         clear_LCD();  // 清除LCD螢幕內容
199         OpenKeyPad(); // 按鍵掃描功能初始化
```

```
201         // --- 蜂鳴器初始化 ---
202         GPIO_SetMode(PB, BIT11, GPIO_PMD_OUTPUT); // 設定PB11為輸出模式
203         PB11 = 1;                                 // 設定為高電位，關閉蜂鳴器（低電位觸發）
204
205         // --- 球體初始狀態設定 ---
206         x = X0;     // 設定球體初始X座標（螢幕中央，64）
207         y = Y0;     // 設定球體初始Y座標（底部區域，60）
208         r = RADIUS; // 設定球體半徑為3像素
209         movX = 0;   // 初始X方向移動距離為0
210         movY = 0;   // 初始Y方向移動距離為0
211         dirX = 0;   // 初始X方向為0（無方向）
212         dirY = 0;   // 初始Y方向為0（無方向）
213
214         // --- 顏色設定 ---
215         bgColor = BG_COLOR; // 設定背景色
216         fgColor = FG_COLOR; // 設定前景色
217
218         // --- 初始化隨機數種子（使用SysTick計數器獲取隨機初始值） ---
219         // SysTick->VAL是系統計數器的當前值，每次啟動時都不同，可作為隨機種子
220         seedCounter = SysTick->VAL;
221
222         // --- 在啟動時產生兩個隨機位置的目標方塊 ---
223         GenerateTwoBlocks(&block1_x, &block1_y, &block2_x, &block2_y, &seedCounter);
224
225         // --- 繪製第一個方塊（5x5像素方塊） ---
226         // 從方塊中心向四周繪製BLOCK_SIZE x BLOCK_SIZE的像素區域
227         for (i = block1_x - BLOCK_SIZE / 2; i <= block1_x + BLOCK_SIZE / 2; i++)
228         {
229             for (j = block1_y - BLOCK_SIZE / 2; j <= block1_y + BLOCK_SIZE / 2; j++)
230             {
231                 draw_Pixel(i, j, FG_COLOR, bgColor); // 繪製前景色像素（黑色方塊）
232             }
233         }
234
235         // --- 繪製第二個方塊（5x5像素方塊） ---
236         for (i = block2_x - BLOCK_SIZE / 2; i <= block2_x + BLOCK_SIZE / 2; i++)
237         {
238             for (j = block2_y - BLOCK_SIZE / 2; j <= block2_y + BLOCK_SIZE / 2; j++)
239             {
240                 draw_Pixel(i, j, FG_COLOR, bgColor); // 繪製前景色像素（黑色方塊）
241             }
242         }
243
244         // --- 繪製初始位置的球體 ---
245         draw_Circle(x, y, r, fgColor, bgColor);
```

```
247         // --- 主程式迴圈 ---
248         while (1)
249         {
250             seedCounter++; // 持續遞增種子計數器，增加隨機性（供未來使用）
251 
252             // --- 掃描按鍵狀態 ---
253             keyin = ScanKey();
254 
255             // --- 按鍵處理邏輯（僅在按鍵按下且與上次不同時執行，實現按鍵釋放檢測） ---
256             // 這可以避免按鍵長按時重複觸發，只有按鍵從無到有（按下）時才執行
257             if (keyin != 0 && keyin != last_key)
258             {
259                 // 根據按下的按鍵執行對應動作
260                 switch (keyin)
261                 {
262                 case 4: // 按鍵4：向左移動（水平移動）
263                     if (!is_moving)
264                     {                   // 只有在未移動時才能啟動新移動（避免重複觸發）
265                         dirX = -1;      // 設定X方向向左
266                         dirY = 0;       // Y方向為0（僅水平移動）
267                         movX = 3;       // 每次移動3像素（X方向）
268                         movY = 0;       // Y方向移動距離為0
269                         is_moving = 1; // 標記為移動狀態
270                     }
271                     break;
272 
273                 case 6: // 按鍵6：向右移動（水平移動）
274                     if (!is_moving)
275                     {
276                         dirX = 1;       // 設定X方向向右
277                         dirY = 0;       // Y方向為0（僅水平移動）
278                         movX = 3;       // 每次移動3像素（X方向）
279                         movY = 0;       // Y方向移動距離為0
280                         is_moving = 1; // 標記為移動狀態
281                     }
282                     break;
283 
284                 case 3: // 按鍵3：右上45度方向移動（對角線移動）
285                     if (!is_moving)
286                     {
287                         dirX = 1;       // 設定X方向向右
288                         dirY = -1;      // 設定Y方向向上（螢幕座標：上為負）
289                         movX = 3;       // 每次移動3像素（X方向）
290                         movY = 3;       // 每次移動3像素（Y方向）
291                         is_moving = 1; // 標記為移動狀態
292                     }
293                     break;
294 
295                 case 9: // 按鍵9：右下45度方向移動（對角線移動）
296                     if (!is_moving)
297                     {
298                         dirX = 1;       // 設定X方向向右
299                         dirY = 1;       // 設定Y方向向下（螢幕座標：下為正）
300                         movX = 3;       // 每次移動3像素（X方向）
301                         movY = 3;       // 每次移動3像素（Y方向）
302                         is_moving = 1; // 標記為移動狀態
303                     }
304                     break;
```

```
306                 case 1: // 按鍵1：左上45度方向移動（對角線移動）
307                     if (!is_moving)
308                     {
309                         dirX = -1;      // 設定X方向向左
310                         dirY = -1;      // 設定Y方向向上
311                         movX = 3;       // 每次移動3像素（X方向）
312                         movY = 3;       // 每次移動3像素（Y方向）
313                         is_moving = 1; // 標記為移動狀態
314                     }
315                     break;
316 
317                 case 7: // 按鍵7：左下45度方向移動（對角線移動）
318                     if (!is_moving)
319                     {
320                         dirX = -1;      // 設定X方向向左
321                         dirY = 1;       // 設定Y方向向下
322                         movX = 3;       // 每次移動3像素（X方向）
323                         movY = 3;       // 每次移動3像素（Y方向）
324                         is_moving = 1; // 標記為移動狀態
325                     }
326                     break;
327 
328                 case 5:             // 按鍵5：停止移動（S鍵）
329                     is_moving = 0; // 標記為停止狀態
330                     movX = 0;       // 清除X方向移動距離
331                     movY = 0;       // 清除Y方向移動距離
332                     break;
```

```
334                 case 8: // 按鍵8：重新產生方塊（R鍵 - Random）
335                     // 只有在兩個方塊都消失時才能重新產生
336                     if (!block1_visible && !block2_visible)
337                     {
338                         // 重新產生兩個新的隨機位置方塊
339                         GenerateTwoBlocks(&block1_x, &block1_y, &block2_x, &block2_y, &seedCounter);
340                         block1_visible = 1; // 標記方塊1為可見
341                         block2_visible = 1; // 標記方塊2為可見
342 
343                         // 清除整個LCD並重新繪製所有物件
344                         clear_LCD();
345 
346                         // 重新繪製第一個方塊
347                         for (i = block1_x - BLOCK_SIZE / 2; i <= block1_x + BLOCK_SIZE / 2; i++)
348                         {
349                             for (j = block1_y - BLOCK_SIZE / 2; j <= block1_y + BLOCK_SIZE / 2; j++)
350                             {
351                                 draw_Pixel(i, j, FG_COLOR, bgColor);
352                             }
353                         }
354 
355                         // 重新繪製第二個方塊
356                         for (i = block2_x - BLOCK_SIZE / 2; i <= block2_x + BLOCK_SIZE / 2; i++)
357                         {
358                             for (j = block2_y - BLOCK_SIZE / 2; j <= block2_y + BLOCK_SIZE / 2; j++)
359                             {
360                                 draw_Pixel(i, j, FG_COLOR, bgColor);
361                             }
362                         }
363 
364                         // 重新繪製球體（保持當前位置）
365                         fgColor = FG_COLOR;
366                         draw_Circle(x, y, r, fgColor, bgColor);
367                     }
368                     break;
369                 }
370             }
371 
372             last_key = keyin; // 記錄當前按鍵值，供下次比較使用（實現按鍵釋放檢測）
```

```
374             // --- 球體移動處理邏輯（僅在移動狀態時執行） ---
375             if (is_moving)
376             {
377                 // 計算球體的下一個位置
378                 new_x = x + dirX * movX; // 新X座標 = 當前X + 方向 * 移動距離
379                 new_y = y + dirY * movY; // 新Y座標 = 當前Y + 方向 * 移動距離
380 
381                 // --- 邊界碰撞檢測與反彈處理 ---
382                 // LCD尺寸為128x64，座標範圍：X(0-127), Y(0-63)
383                 bounced = 0; // 初始化反彈標誌為未反彈
384 
385                 // 檢查X軸邊界（左右邊界）
386                 if (new_x <= r)
387                 {
388                     // 碰撞左邊界：球體左邊緣觸碰到螢幕左邊緣
389                     if (dirX != 0)
390                     {
391                         dirX = 1; // 反彈向右（改變X方向）
392                     }
393                     new_x = r;   // 限制X座標在邊界位置（避免超出螢幕）
394                     bounced = 1; // 標記為已反彈
395                 }
396                 else if (new_x >= (127 - r))
397                 {
398                     // 碰撞右邊界：球體右邊緣觸碰到螢幕右邊緣（最大X為127）
399                     if (dirX != 0)
400                     {
401                         dirX = -1; // 反彈向左（改變X方向）
402                     }
403                     new_x = 127 - r; // 限制X座標在邊界位置
404                     bounced = 1;     // 標記為已反彈
405                 }
406 
407                 // 檢查Y軸邊界（上下邊界）
408                 if (new_y <= r)
409                 {
410                     // 碰撞上邊界：球體上邊緣觸碰到螢幕上邊緣
411                     if (dirY != 0)
412                     {
413                         dirY = 1; // 反彈向下（改變Y方向）
414                     }
415                     new_y = r;   // 限制Y座標在邊界位置
416                     bounced = 1; // 標記為已反彈
417                 }
418                 else if (new_y >= (63 - r))
419                 {
420                     // 碰撞下邊界：球體下邊緣觸碰到螢幕下邊緣（最大Y為63）
421                     if (dirY != 0)
422                     {
423                         dirY = -1; // 反彈向上（改變Y方向）
424                     }
425                     new_y = 63 - r; // 限制Y座標在邊界位置
426                     bounced = 1;    // 標記為已反彈
427                 }
428 
429                 // --- 方塊碰撞檢測與處理（分為兩步驟） ---
430                 overlap = 0; // 初始化碰撞標誌為未碰撞
```

```
432                 // 步驟1：檢查當前位置是否與方塊重疊（防止球體停駐在方塊內）
433                 if (block1_visible)
434                 {
435                     if (CheckOverlap(x, y, r, block1_x, block1_y, BLOCK_SIZE))
436                     {
437                         // 當前位置已重疊！立即清除方塊1
438                         for (i = block1_x - BLOCK_SIZE / 2; i <= block1_x + BLOCK_SIZE / 2; i++)
439                         {
440                             for (j = block1_y - BLOCK_SIZE / 2; j <= block1_y + BLOCK_SIZE / 2; j++)
441                             {
442                                 draw_Pixel(i, j, bgColor, bgColor); // 用背景色繪製，清除方塊
443                             }
444                         }
445                         block1_visible = 0; // 標記方塊1為已消失
446                         overlap = 1;        // 標記為已碰撞
447                     }
448                 }
449 
450                 if (block2_visible)
451                 {
452                     if (CheckOverlap(x, y, r, block2_x, block2_y, BLOCK_SIZE))
453                     {
454                         // 當前位置已重疊！立即清除方塊2
455                         for (i = block2_x - BLOCK_SIZE / 2; i <= block2_x + BLOCK_SIZE / 2; i++)
456                         {
457                             for (j = block2_y - BLOCK_SIZE / 2; j <= block2_y + BLOCK_SIZE / 2; j++)
458                             {
459                                 draw_Pixel(i, j, bgColor, bgColor); // 用背景色繪製，清除方塊
460                             }
461                         }
462                         block2_visible = 0; // 標記方塊2為已消失
463                         overlap = 1;        // 標記為已碰撞
464                     }
465                 }
```

```
467                 // 步驟2：檢查下一位置是否與方塊重疊（預測碰撞）
468                 if (block1_visible)
469                 {
470                     if (CheckOverlap(new_x, new_y, r, block1_x, block1_y, BLOCK_SIZE))
471                     {
472                         // 將要發生碰撞！清除方塊1
473                         for (i = block1_x - BLOCK_SIZE / 2; i <= block1_x + BLOCK_SIZE / 2; i++)
474                         {
475                             for (j = block1_y - BLOCK_SIZE / 2; j <= block1_y + BLOCK_SIZE / 2; j++)
476                             {
477                                 draw_Pixel(i, j, bgColor, bgColor); // 用背景色繪製，清除方塊
478                             }
479                         }
480                         block1_visible = 0; // 標記方塊1為已消失
481                         overlap = 1;        // 標記為已碰撞
482                     }
483                 }
484 
485                 if (block2_visible)
486                 {
487                     if (CheckOverlap(new_x, new_y, r, block2_x, block2_y, BLOCK_SIZE))
488                     {
489                         // 將要發生碰撞！清除方塊2
490                         for (i = block2_x - BLOCK_SIZE / 2; i <= block2_x + BLOCK_SIZE / 2; i++)
491                         {
492                             for (j = block2_y - BLOCK_SIZE / 2; j <= block2_y + BLOCK_SIZE / 2; j++)
493                             {
494                                 draw_Pixel(i, j, bgColor, bgColor); // 用背景色繪製，清除方塊
495                             }
496                         }
497                         block2_visible = 0; // 標記方塊2為已消失
498                         overlap = 1;        // 標記為已碰撞
499                     }
500                 }
```

```
502                 // --- 步驟3：檢查是否所有方塊都已消失（遊戲結束條件） ---
503                 if (!block1_visible && !block2_visible)
504                 {
505                     // 所有方塊都已消失，重置遊戲狀態
506                     // 重要：在移動之前先重置，避免球體繼續移動
507 
508                     // 清除當前位置的球體
509                     fgColor = BG_COLOR; // 使用背景色清除
510                     draw_Circle(x, y, r, fgColor, bgColor);
511 
512                     // 重置球體到初始位置
513                     x = X0; // 重置X座標到中央
514                     y = Y0; // 重置Y座標到底部
515 
516                     // 停止移動並清除所有移動參數
517                     is_moving = 0; // 停止移動
518                     movX = 0;      // 清除X移動距離
519                     movY = 0;      // 清除Y移動距離
520                     dirX = 0;      // 清除X方向
521                     dirY = 0;      // 清除Y方向
522 
523                     // 在初始位置重新繪製球體
524                     fgColor = FG_COLOR; // 使用前景色繪製
525                     draw_Circle(x, y, r, fgColor, bgColor);
526 
527                     // 如果發生碰撞，觸發蜂鳴器（目前Buzz函數未實作）
528                     if (overlap)
529                     {
530                         Buzz(1);
531                     }
532 
533                     // 短延遲後繼續，避免重複處理
534                     CLK_SysTickDelay(10000);
535                     continue; // 跳過後續繪製步驟，直接進入下一次迴圈
536                 }
537 
538                 // --- 步驟4：正常移動處理 - 清除舊位置並繪製新位置 ---
539                 // 只有在所有檢查完成後才執行移動，確保不會在異常狀態下移動
540                 fgColor = BG_COLOR; // 使用背景色清除舊球體
541                 draw_Circle(x, y, r, fgColor, bgColor);
```

```
543                 // --- 步驟5：重新繪製仍然可見的方塊（避免球體移動時擦除方塊） ---
544                 // 這確保方塊在球體移動後仍然正確顯示
545                 if (block1_visible)
546                 {
547                     for (i = block1_x - BLOCK_SIZE / 2; i <= block1_x + BLOCK_SIZE / 2; i++)
548                     {
549                         for (j = block1_y - BLOCK_SIZE / 2; j <= block1_y + BLOCK_SIZE / 2; j++)
550                         {
551                             draw_Pixel(i, j, FG_COLOR, bgColor); // 重新繪製方塊1
552                         }
553                     }
554                 }
555 
556                 if (block2_visible)
557                 {
558                     for (i = block2_x - BLOCK_SIZE / 2; i <= block2_x + BLOCK_SIZE / 2; i++)
559                     {
560                         for (j = block2_y - BLOCK_SIZE / 2; j <= block2_y + BLOCK_SIZE / 2; j++)
561                         {
562                             draw_Pixel(i, j, FG_COLOR, bgColor); // 重新繪製方塊2
563                         }
564                     }
565                 }
566 
567                 // 更新球體位置到新位置
568                 x = new_x; // 更新X座標
569                 y = new_y; // 更新Y座標
570 
571                 // 在新位置繪製球體
572                 fgColor = FG_COLOR; // 使用前景色繪製
573                 draw_Circle(x, y, r, fgColor, bgColor);
574 
575                 // 如果發生反彈或碰撞方塊，觸發蜂鳴器（目前Buzz函數未實作）
576                 if (bounced || overlap)
577                 {
578                     Buzz(1);
579                 }
580 
581                 // 動畫延遲（控制移動速度）
582                 CLK_SysTickDelay(50000); // 延遲50毫秒，控制球體移動的視覺速度
583             }
584             // 當球體停止時
585             else
586             {
587                 CLK_SysTickDelay(10000); // 短延遲10毫秒，降低CPU使用率
588             }
589         }
590     }
```