

Team No : 15	Team Name : TING & TING
Project Title : 小艾走不出去的迷宮	
Name : 留鈺婷	ID : 111062320
Name : 陳璟婷	ID : 111062112

一、設計概念

在這次 Project 中我們希望應用課堂上所學到的東西，設計出一個迷宮小遊戲。

(一) Background Story

三隻小艾散步時突然間被神秘的力量帶入一個異世界。眼前豎立起一座古老而神秘的迷宮，必須避開怪物的追蹤，跟循指示找到三把鑰匙，方能從唯一的出口離開。

每個小艾面臨不同的挑戰，他們必須借助玩家的智慧和冒險精神，才能成功逃離這個異世界。小艾們是否能在怪物的追捕中順利取得鑰匙，找到迷宮深處的出口？這全取決於玩家的指引！

現在，帶領小艾們踏上未知的冒險，逃離異世界！

(二) Description

1、**Title** 可以從 **Help** 查看遊玩方式，或是選擇關卡，只有通關前一關才會解鎖新的關卡。

2、玩家需要用鍵盤操控角色在迷宮中尋找鑰匙、電燈開關和出口，共有三個關卡。

(1) Stage1: Beginner's Luck

一般關卡，只需找到三把鑰匙和出口即可通關。

(2) Stage2: Live in the Light

除了玩家和電燈開關周圍，其他地方都是暗的。

玩家必須先找到電燈開關，再找到三把鑰匙和出口才能通關。

(3) Stage3: Be Free

有一位 boss 會根據玩家位置跟隨玩家，若被抓到會減少一條生命值。

玩家需躲避 boss 並找到鑰匙和出口才能通關。

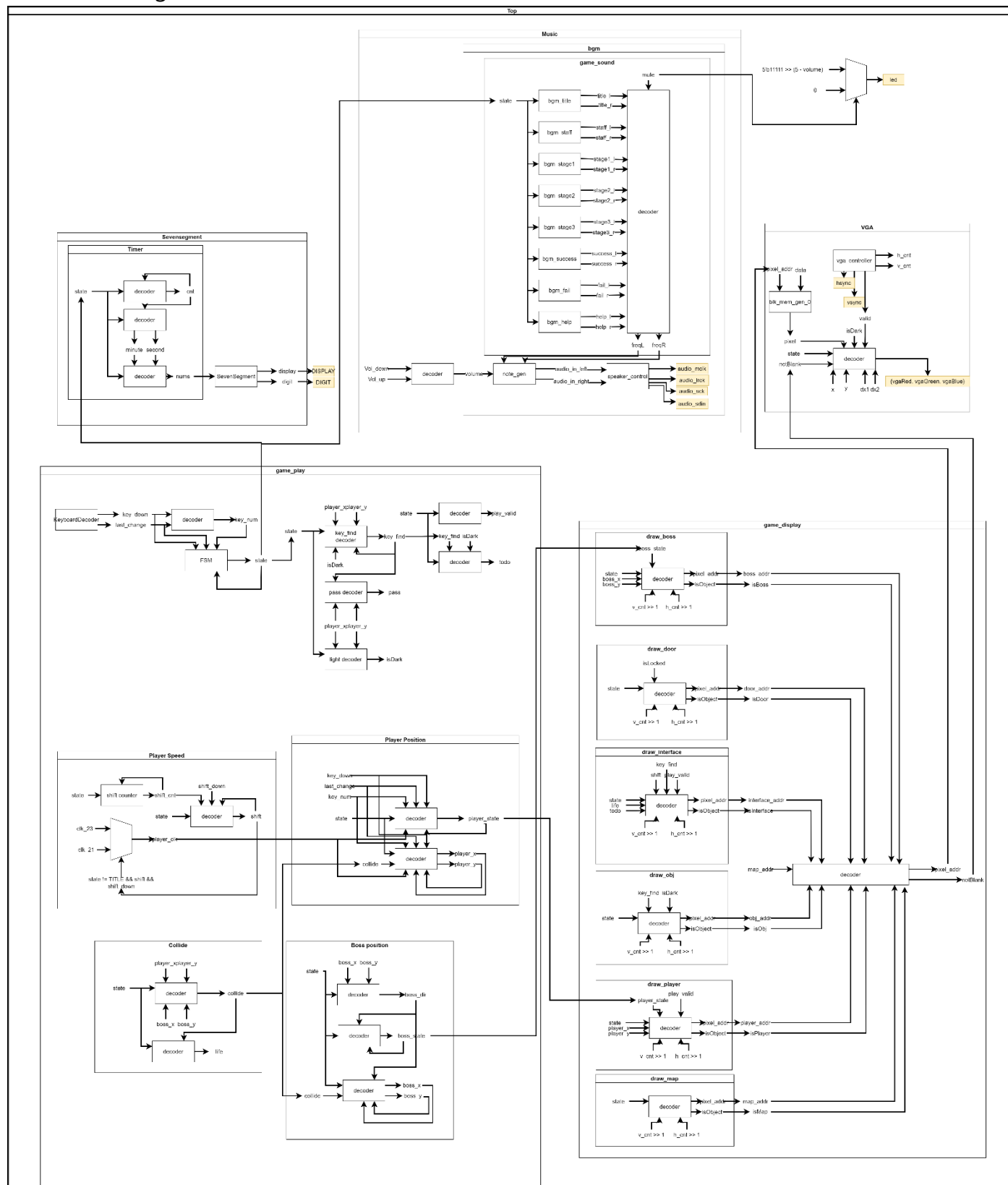
(4) 若生命值歸零會進入 **Game Over** 畫面，通關則會看到 **You Win** 畫面。

3、通關後可以選擇要進入下一關或是回到 **Title**；失敗後可以選擇重新挑戰或回到 **Title**。

(三) I/O signal specification

Connect	Name	Description
W5	clk	Clock
BtnC	_rst	Reset
BtnU	_volUP	Volume Up
BtnD	_volDOWN	Volume Down
SW 14	_mute	Mute
LED 0 ~ 4	led	Volume Indicator
Pmod JB 1~4	audio_mclk 、 audio_lrck 、 audio_sck 、 audio_sdin	Pmod I2S
7-Segment	DISPLAY 、 DIGIT	Time Record
pin N19, J19, H19, G19	vgaRed	Screen Display
pin D17, G17, H17, J17	vgaGreen	
pin J18, K18, L18, N18	vgaBlue	
pin P19	hsync	
pin R19	vsync	
C17	PS2_CLK	Keyboard
B17	PS2_DATA	

1 、Block Diagram

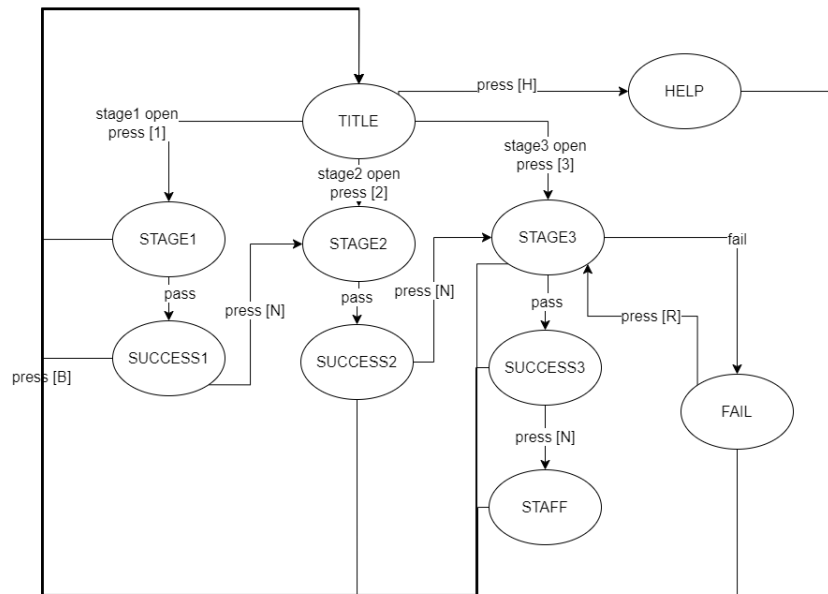


2、Explanation

Top module 中主要分為遊戲控制、螢幕顯示、音樂和時間顯示。

(1) 遊戲控制：game_play

game_play 利用 keyboard 轉換 state 和改變 player_position，shift 切換 player_clk 以變換 player 的速度；key_find decoder 中更新 key_find 數量，並以此判斷是否 pass；light decoder 判斷 stage2 中的 isDark 的值。在 stage3 中，偵測 player 是否與 boss 相撞，以改變 player 和 boss 的位置以及遊戲狀態。



螢幕轉換的機制

在 TITLE 中，如果按下[1]並且 STAGE1 是解鎖的，則畫面跳到 STAGE1；如果按下[2]並且 STAGE2 是解鎖的，則畫面跳到 STAGE2；如果按下[3]並且 STAGE3 是解鎖的，則畫面跳到 STAGE3；如果按下[H]，則畫面跳到 Help。

在每個 STAGE 中，如果通關 (pass)，則跳至其對應的 SUCCESS 畫面；如果在 STAGE3 中失敗 (fail)，則跳至 FAIL 畫面。

在 SUCCESS1、SUCCESS2 畫面中，如果按下[N]，則跳至下一個 STAGE 畫面；在 SUCCESS3 畫面中，如果按下[N]，則跳至 STAFF 畫面。

在任意一個非 TITLE 的畫面中，只要按下[B]，會跳回 TITLE 畫面。

(2) 螢幕顯示：VGA、game_display

game_display 使用 **draw_[obj]** 查找 (h_cnt, v_cnt) 上是否有需要顯示的 object，並輸出 isBlank 和 pixel_addr。**blk_mem_gen_0** 則根據 pixel_addr 讀取圖片的顏色讓螢幕顯示。

draw_player	玩家控制的角色
draw_boss	Stage3 中會追逐玩家的角色
draw_map	迷宮的地圖 (牆壁)
draw_door	出口 (Locked & Unlocked)
draw_obj	鑰匙和燈泡
draw_interface	標題、遊戲指示等遊戲介面或其他互動物品

(3) 音樂：Music

bgm_[state] 根據音樂進度輸出 note 對應的頻率，**game_sound** 則依照 state 選擇不同的 bgm，最後再用 **speaker_control** 輸出音樂。另外由 vol_down、vol_UP、mute 調整音量並在 led 顯示。

(4) 時間顯示：SevenSegment

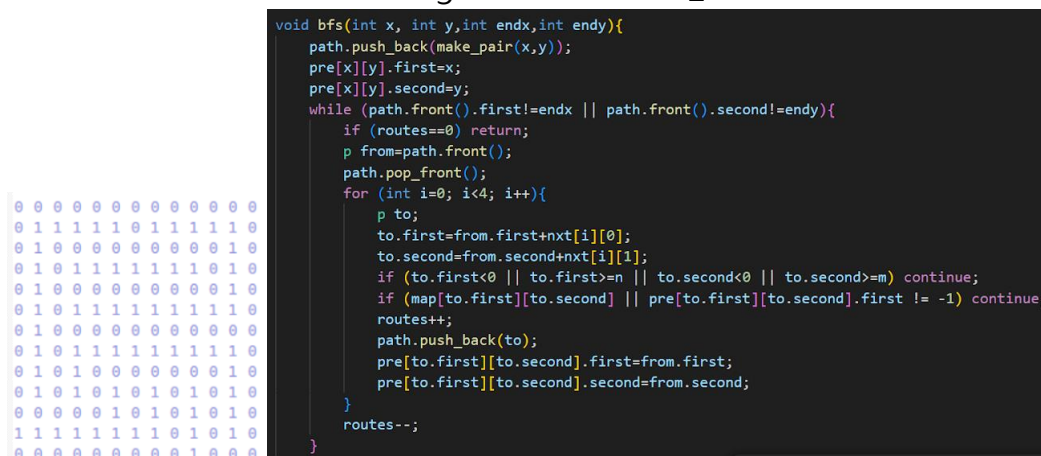
Timer 計算通關時間 (秒、分鐘)，再透過 **SevenSegment** 顯示。

(二) Game Implementation

除了之前 lab 做到的鍵盤與螢幕的互動等，這次 Project 中我們有一些比較重要的實作。

1、Boss movement

在 C++ 中，先利用一個 (13*13) 的地圖，用 BFS 演算法求任兩點間的最短路徑 path，並存其起始點的方向。再將此陣列帶回到 verilog 中，以 shortest_dir 儲存其值。



```
void bfs(int x, int y, int endx, int endy){
    path.push_back(make_pair(x,y));
    pre[x][y].first=x;
    pre[x][y].second=y;
    while (path.front().first!=endx || path.front().second!=endy){
        if (routes==0) return;
        p=from=path.front();
        path.pop_front();
        for (int i=0; i<4; i++){
            p=to;
            to.first=from.first+next[i][0];
            to.second=from.second+next[i][1];
            if (to.first<0 || to.first>=n || to.second<0 || to.second>=m) continue;
            if (map[to.first][to.second] || pre[to.first][to.second].first != -1) continue;
            routes++;
            path.push_back(to);
            pre[to.first][to.second].first=from.first;
            pre[to.first][to.second].second=from.second;
        }
        routes--;
    }
}
```

利用 `shortest_dir` 陣列存在任意 boss 位置和任意 player 位置時，boss 應該要往哪一個方向前進，其中，第 $13m + n$ 行代表 player 在 `map[m][n]` (13×13) 時，boss 在各點的追蹤方向，有上下左右四個方向，以兩個 bit 為一個單位 (00 : Up、01 : Down、10 : Right、11 : Left)，因此每一個陣列的元素會有 338 (169×2) 個 bit。

```
parameter [0:337] shortest_dir[0:168]={
```

計算 boss 和 player 對應到 13*13 map 的位置後，利用 shortest_dir 取得 boss 的追蹤方向，其中

`{ shortest_dir [ax+ay*13] [(bx+by*13)*2] · shortest_dir [ax+ay*13] [(bx+by*13)*2+1] }`
代表從 (bx, by) 到 (ax, ay) 的最短路徑的方向。

```
boss_x_maze = ((boss_x-60)/5-1)/3;
boss_y_maze = ((boss_y-30)/5-1)/3;
player_x_maze = ((player_x-60)/5-1)/3;
player_y_maze = ((player_y-30)/5-1)/3;
assign next_boss_dir = shortest_dir[(player_x_maze+player_y_maze*13)][(boss_x_maze+boss_y_maze*13)*2*2+shortest_dir[(player_x_maze+player_y_maze*13)][(boss_x_maze+boss_y_maze*13)*2+1];
```

2、Light

如果 state2 的燈還沒亮，用 dx 和 dy 計算 $dx^2 + dy^2 < r^2$ ，

讓人物和燈周邊圓圈範圍以外的圖片都不顯示。

```

if (state == 4 && x >= 60 && y >= 30) begin// STAGE2
    if (isDark) begin
        if (dx1 * dx1 + dy1 * dy1 < 400)
            {vgaRed, vgaGreen, vgaBlue} = pixel;
        else if (dx2 * dx2 + dy2 * dy2 < 200)
            {vgaRed, vgaGreen, vgaBlue} = pixel;
        end else {vgaRed, vgaGreen, vgaBlue} = pixel;
        dx1 = (x > player_x + 5) ? x - player_x - 5 : player_x + 5 - x;
        dy1 = (y > player_y + 5) ? y - player_y - 5 : player_y + 5 - y; // Player
        dx2 = (x > 70 + 5) ? x - 70 - 5 : 70 + 5 - x;
        dy2 = (y > 220 + 5) ? y - 220 - 5 : 220 + 5 - y; // Light
    end
end

```

3、Character

用 player_state 和 boss_state 判斷要顯示的動作圖片。

```
if(key_down[last_change]) begin
    case (key_num)
    4: player_state <= (player_state == UP2) ? UP3 : UP2;
    5: player_state <= (player_state == LEFT2) ? LEFT3 : LEFT2;
    6: player_state <= (player_state == DOWN2) ? DOWN3 : DOWN2;
    7: player_state <= (player_state == RIGHT2) ? RIGHT3 : RIGHT2;
    default: player_state <= player_state;
    endcase
end else player_state <= player_state / 3 * 3;
case (boss_dir)
0: boss_state <= (boss_state == UP2) ? UP3 : UP2;
3: boss_state <= (boss_state == LEFT2) ? LEFT3 : LEFT2;
1: boss_state <= (boss_state == DOWN2) ? DOWN3 : DOWN2;
2: boss_state <= (boss_state == RIGHT2) ? RIGHT3 : RIGHT2;
default: boss_state <= boss_state;
endcase
```

```
UP1 = 0, UP2 = 1, UP3 = 2;
RIGHT1 = 3, RIGHT2 = 4, RIGHT3 = 5;
LEFT1 = 6, LEFT2 = 7, LEFT3 = 8;
DOWN1 = 9, DOWN2 = 10, DOWN3 = 11;
```



4、Music

我們用 lab7 中的方法實作 bgm，但因為我們每一個畫面都有各自不一樣的 bgm，為了讓打譜不要浪費太多時間，我們有配合 C++ 設計方便寫音樂的程式。

```
string key[12] =
{"C", "Cs", "D", "Ds", "E", "F", "Fs", "G", "Gs", "A", "As", "B"};

bgm_title #(.LEN(2240)) bgmTitle(...);
bgm_staff #(.LEN(1600)) bgmStaff(...);
bgm_stage1 #(.LEN(1408)) bgmStage1(...);
bgm_stage2 #(.LEN(288)) bgmStage2(...);
bgm_stage3 #(.LEN(1024)) bgmStage3(...);
bgm_success #(.LEN(1536)) bgmSuccess(...);
bgm_fail #(.LEN(1152)) bgmFail(...);
bgm_help #(.LEN(1024)) bgmHelp(...);

int main () {
    ofstream myfile;
    myfile.open ("stage3_L.txt");
    int num = 4, measure = 16;
    for (int k = 0; k < measure; k++) {
        myfile << "// Measure " << k+1 << " //\n";
        cout << "Measure " << k+1 << ":\n";
        for (int i = 0; i < 2*num; i++) {
            string note;
            cout << "note: ";
            cin >> note;
            for (int j = 0; j < 4; j++) {
                myfile << "12'd" << 64*k+8*i+2*j << ": tone = `" << note << " ";
                myfile << "12'd" << 64*k+8*i+2*j+1 << ": tone = `" << note << ";\n";
            }myfile << "\n";
        }
    }
    myfile.close();
    return 0;
}
```

三、實作完成度

(一) Undone

1、遊戲暫停

原本想設計遊戲暫停畫面，後來改成了 back 按鍵讓玩家可以直接回到 TITLE 代替。

2、隨機的鑰匙位置

想讓鑰匙的位置可以隨機生成，但是因為怕 LUT 不夠用，因此捨去。

3、音效

因為我們的背景音樂有主旋律和伴奏，已經將兩個聲道都占用了，所以就沒有做音效。

(二) Additional

1、Help 畫面

新增 Help 畫面介紹遊戲操作方式。

2、返回首頁

Stage 畫面中可以按 B 鍵回到 Title 畫面，讓玩家可以隨時離開遊戲回到 Title。

3、新增動畫

新增其他畫面的動畫，增加遊戲螢幕的豐富度。

四、困難與解決方法

(一) Boss 追蹤 player

為了讓 boss 能夠追蹤 player 的位置，我們一開始便想到用 BFS 取其追蹤的最短路徑之方向，但是經過嘗試之後我們發現 while 不能合成到 FPGA 中，會合成超久還 synthesis failed。

我們後來先利用 c++ 語言，將地圖上每個 a 點到 b 點最短距離要走的方向存進陣列。如果以一個 pixel 為單位直接跑 BFS 的話，計算和陣列空間仍然龐大，並且因為路寬超過一格，最短路徑解容易出現錯誤。

所以最後我們便將迷宮簡化為路和牆皆為一格單位 (13*13)，再用簡化後的路線找最短路徑。在 verilog 中，就可以直接透過目前 player 和 boss 對應到 13*13 地圖的位置，直接查找陣列決定 boss 下一步要走的方向。

(二) LUT 不夠

最後我們在各個畫面加上人物動畫時，發現 LUT 會超過限制。所以我們盡量將每個 module 中不必要的計算和變數去除掉，但作用微乎其微。最後我們決定將 Title 其中一個小動畫，也就是 Help 鍵旁邊原本會出現 boss 的動畫刪除，才終於讓 LUT 低於 100%，使其正常運作。因此也打退我們原本想新增更多功能的念頭。

五、心得討論

這是我們第一次製作沒有 template 的 project，雖然每個功能大概知道如何實作，但是整個程式的架構都還不清楚。一開始花了很多時間思考和整理硬體之間與 modules 之間的關係，再根據先前 lab 的經驗大致寫出 modules 的 template，好的開始是成功的一半，之後實作細節時就可以很順利的進行。

在過程中我們也遇到了一些困難，像是雖然 boss 追蹤 player 的方法可以用 C++ 解決，但實際用在遊戲裡有很多 boundary condition，難度不高卻需要細心花時間處理細節。後面做得越來越上手，我們新增了一些平常玩遊戲會有的功能，讓我們的 project 可以更接近一般遊戲體驗。不過在快結尾時發現 LUT 不夠用，即使有想新增的功能也不太能實作（我們認為主要是螢幕顯示的計算和記錄 boss 的陣列佔了不少），但自認有讓原本簡單的小遊戲多了更清楚明瞭的 UI（也更可愛）。

雖然每周都在煩惱各科的期末，HDL 的 project 倒是做得很開心。當初預想的進度規劃我們幾乎都有跟上，甚至有時候半夜做上頭了還會心血來潮新增小功能。所以我們就算遇到困難也都有足夠的時間靜下心處理，不會出現同時要處理一堆 bug 又要趕進度的焦慮狀況。

在這次 project 中，我們學習到如何從零開始製作一個遊戲，包括遊戲構想與程式架構規劃；當然也從每一次的 lab 中累積（debug）經驗和耐心，讓我們最後可以用養孩子的心態在設計這個 Project（辛苦又開心；；）。真的很感謝老師和助教們一學期以來提供的幫助，也希望透過這次的經驗，之後可以更有效率也更完善豐富！

六、分工

留鈺婷：

- Draw Pictures & Display
- Map Design
- Boss Direction of Shortest Path
- Speed Up

陳璟婷：

- Top Module
- Screen FSM
- Game Control
- Music

