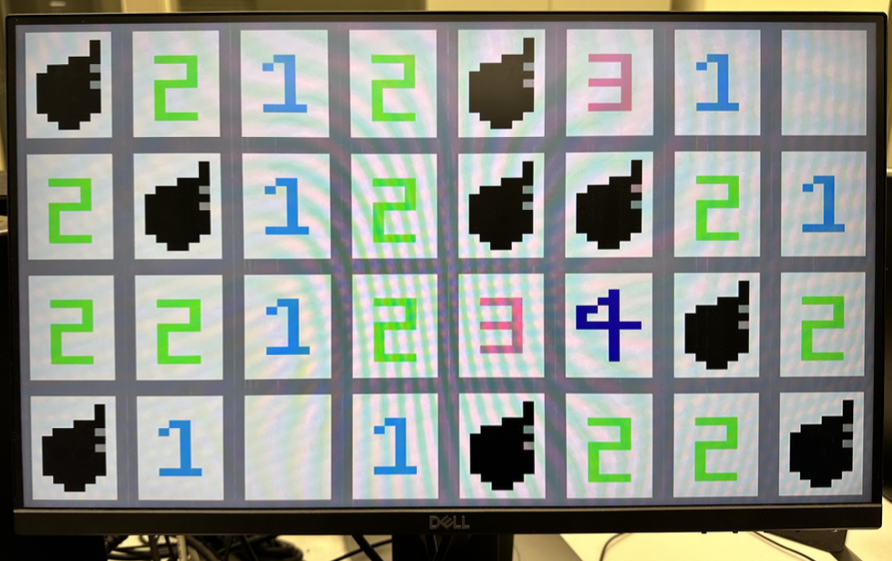
|  |  |
| --- | --- |
| **Final Project Report** | |
| 姓名：李晉方、楊承霖 | 組別：41 |

1. **­Design Concept**

****

踩地雷應該是大家小時候都有遊玩過的小遊戲，這次的專案中我們運用上課所教學過的switch、LED、button、7-segment、keyboard、VGA等等裝置。

一張含有 螢幕擷取畫面, 寫生, 設計, 圖表 的圖片

自動產生的描述

以上是這次project的簡易概念圖，與遊戲互動主要裝置是fpga板上的BTNU以及switch0、switch15以及拿來踩地雷用的鍵盤。

下圖是4\*8踩地雷所對應到的鍵盤按鍵。

一張含有 螢幕擷取畫面, 平行 的圖片

自動產生的描述

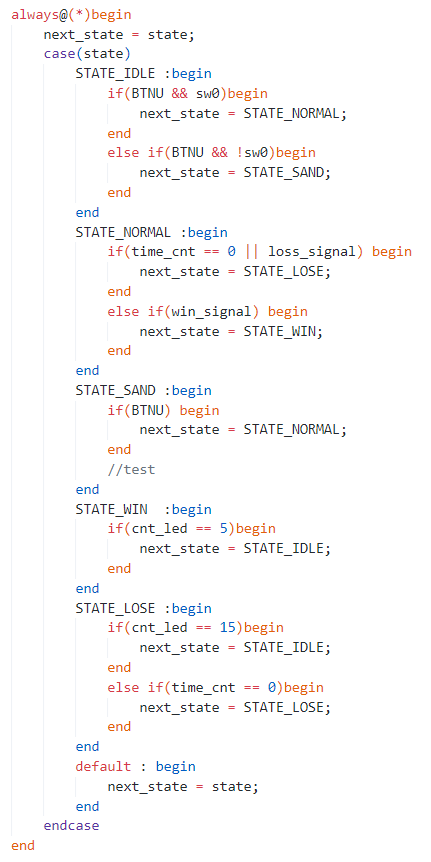
1. **Project Implementation**
2. block diagram

一張含有 文字, 圖表, 方案, 圖解 的圖片

自動產生的描述

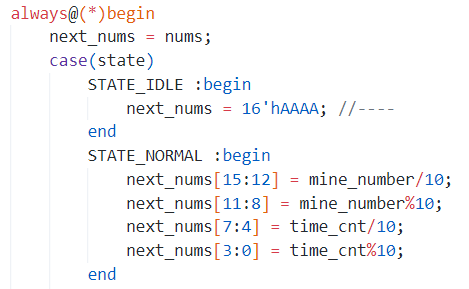
II. code

1. state transition



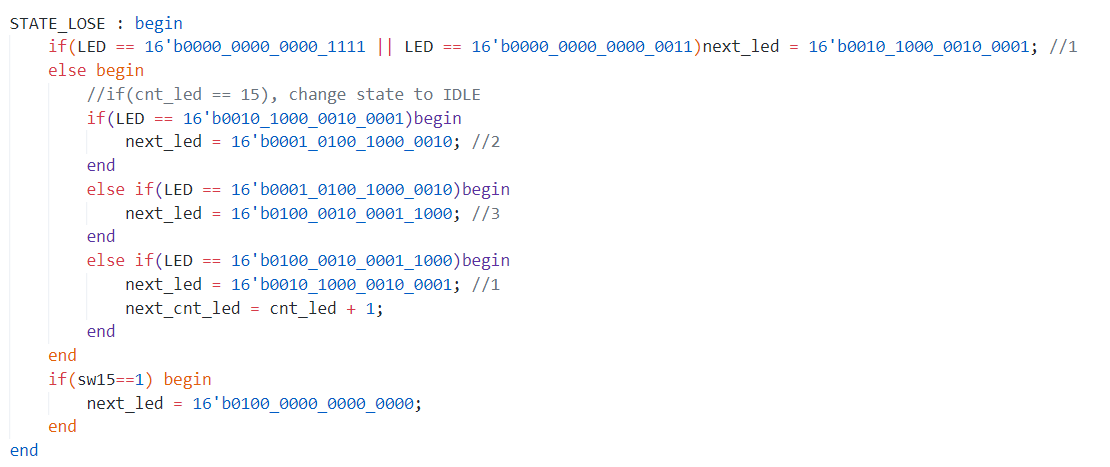
在遊戲中有五個STATE，分別為起始的STATE\_IDLE，按下BTNU可以決定要進入STATE\_NORMAL(普通模式)，STATE\_SAND (沙盒模式)，沙盒模式建置完地雷會回到普通模式遊玩，普通模式中，踩完所有非地雷的格子會進入STATE\_WIN，踩到地雷或超時(99秒)皆會進入STATE\_LOSS。STATE\_WIN跟STATE\_LOSS經過一段時間(cnt\_led計算)後會回到STATE\_IDLE。

1. seven segment and LED



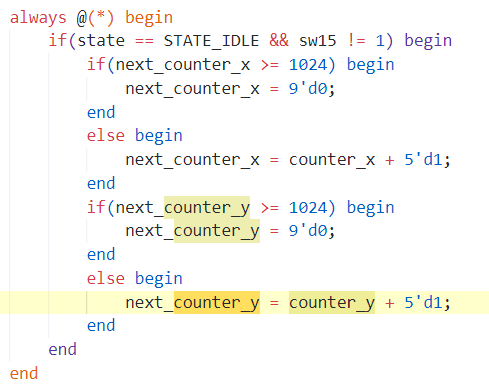
上面是七段顯示器的部分程式碼，在STATE\_NORMAL中前兩格顯示地雷數量，後兩格顯示倒數的時間。

在STATE\_SAND會顯示自訂義放置的地雷有幾個了。



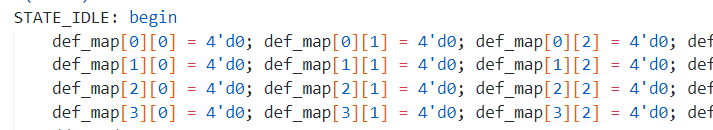
LED的部分就用if-else做出動畫的效果

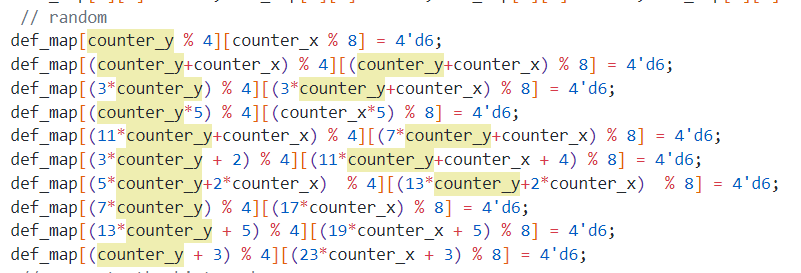
1. random coordinate generator



如上圖，首先使用一個counter計算counter\_x和counter\_y，只要state 是 IDLE時，這個counter就會時時更新。

注意sw15==1時是hint模式，這時候不會讓counter更新。

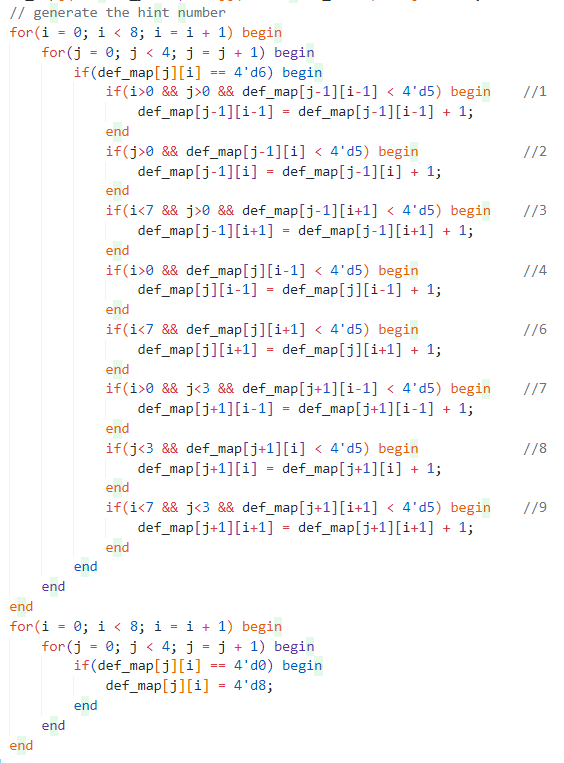




如上圖，在STATE\_IDLE時對counter\_x和counter\_y進行一些自訂的計算，因為counter的特性，每次都會計算出不一樣的結果，使其底圖(def\_map)放置地雷的位置有所不同，盡可能達到隨機的效果。

將10個不同的function覆蓋10個位置的地雷，由於不同function可能會產出相同的座標，故隨機生成的地雷至多有10個，可能比10個少。

1. number hint generator



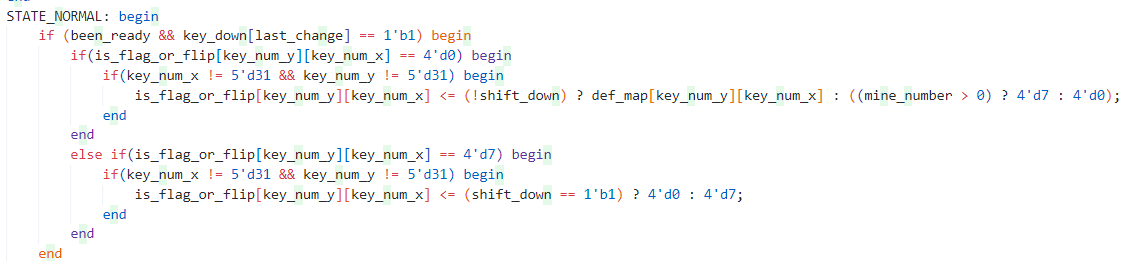
接續剛剛隨機生成地雷的部分，如果底圖有地雷的話，將周圍八個方格的值都加1。

這邊解釋def\_map值的意義，def\_map代表底圖。0是未挖過的圖，1、2、3、4分別代表地雷數量，5代表周圍有可能有5、6、7、8顆地雷，6為地雷，7為旗子，8表示周圍沒有任何地雷的底圖。

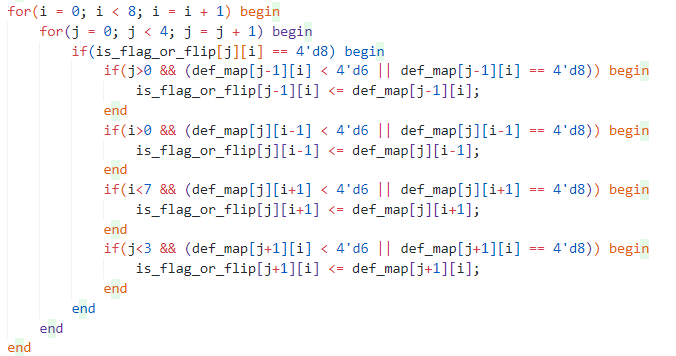
在STATWE\_IDLE中，一開始def\_map會定義為0，跑完隨機後代表有一些def\_map已被更新為6(地雷)。開始尋找，如果底圖有地雷的話，將周圍八個方格的值都加1，超過5顆地雷代表def\_map必為5或地雷(6)。全部事情做完後，將def\_map為0的 (未挖過的圖) 改為8。

1. keyboard and minesweeper implementation

實作的部分分成兩個圖層有顯示在螢幕的is\_flag\_or\_flip跟底圖def\_map，def\_map會在STATE\_IDLE都決定好。keyboard的操作改到的東西只會影響is\_flag\_or\_flip，在4\*8的鍵盤中，對應到各個地雷方格，同時按下按鍵跟shift為插旗。沒按shift為純粹翻開地雷。翻開地雷的話就是讓is\_flag\_or\_flip等於def\_map。



1. sweep the adjacent empty rigid in a time

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

Is\_flag\_or\_flip為8時，若翻開那塊為空白，且周圍十字範圍的區塊沒有地雷，這時候要進行連環爆，顯示出周圍的數字，一樣用迴圈去檢查，找到空白周圍不為地雷的全部打開。

1. lose or win judgement

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

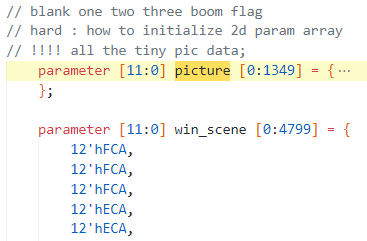
自動產生的描述

在STATE\_NORMAL判斷是否進loss or win state是用兩個signal來記錄，而那兩個signal在另一個module my\_pixel\_gen 裡。

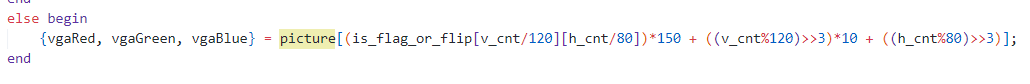
當前地圖中出現地雷代表輸掉遊戲 (4' d6為地雷)

判斷是否贏會多個步驟，先統計當前地圖沒被翻開的的數量 (4’d0為未翻開，4'd7為旗子 )，如果數量與地雷數量相等代表遊戲勝利。

1. VGA implementation

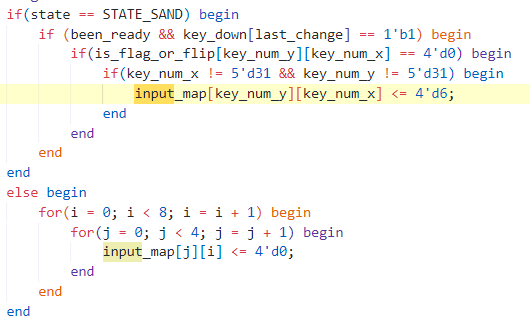


因為怕記憶體容量不夠，螢幕一開始設為60\*80，被分為32格，每格為15\*10，所以一格需要的pixel為150個，我們用陣列picture、win\_scene跟loss\_scene裝。



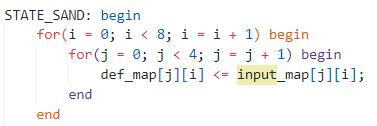
V\_cnt/120和h\_cnt/80將圖片化成4\*8，每組圖片在picture中會有150個pixel，所以根據is\_flag\_or\_flip的數字\*150就會對照到當組圖片。

1. Sand mode



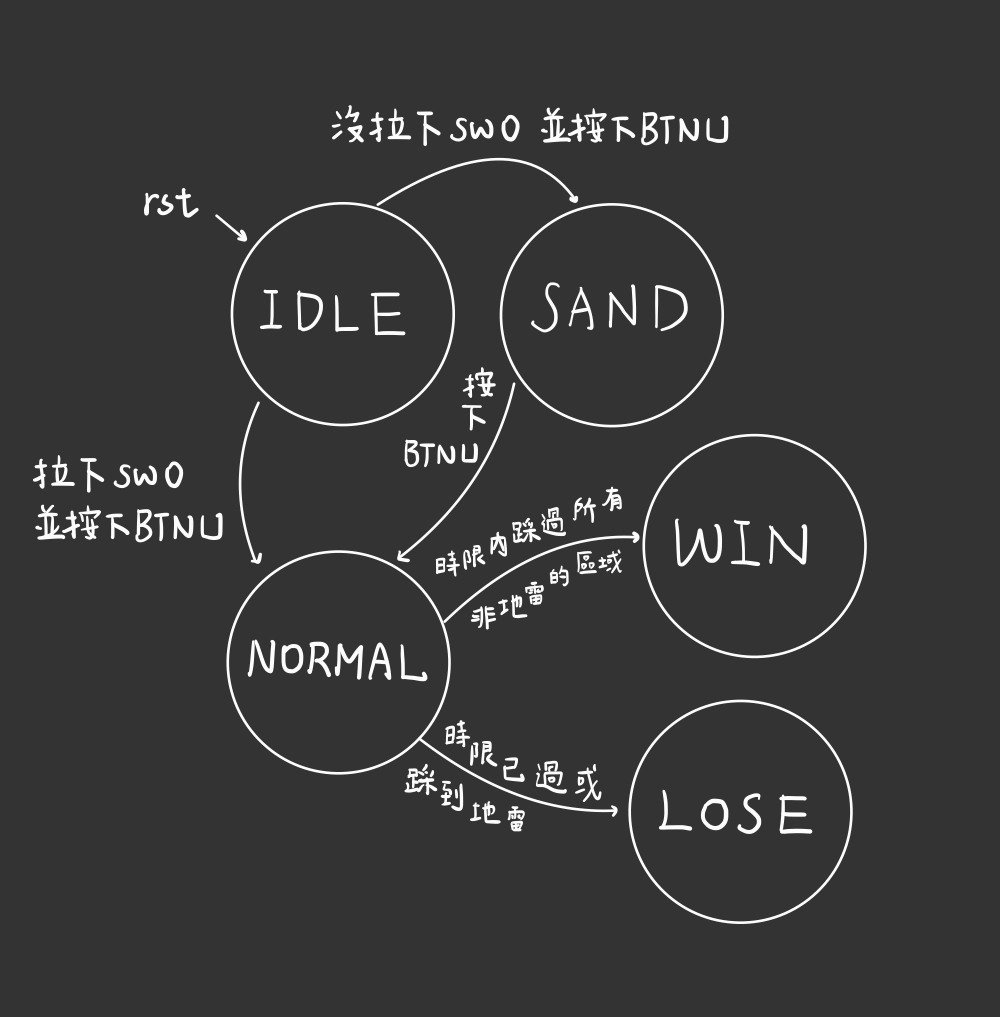
沙盒模式中，玩家可以布置自己的地雷並切回STATE\_NORMAL遊玩。

撰寫上就是開另一個map: input\_map將按下的鍵記錄下來



接著讓底圖def\_map的值為input\_map，這時切回STATE\_NORMAL理論上def\_map應要為input\_map，但卻一直為隨機地圖，明明隨機是在STATE\_IDLE寫的，這部分也成為我們沒完成的部分。

III. FSM



1. **Problem Encountered**

第一個問題 : 如何翻圖片

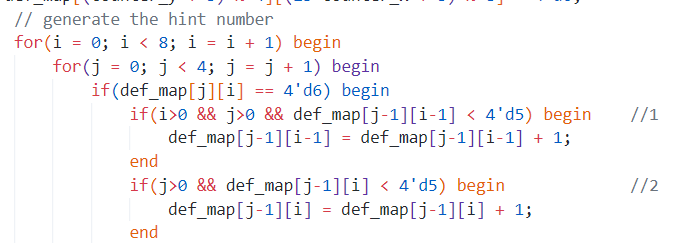
這邊是最剛開始做project時，對於實作的方向不確定，我們原本想先做兩張大張的底圖，也就是沒有隨機地圖，實作也比較簡單。後來考慮到完整性決定嘗試看看不熟悉的二維陣列，發現其實建圖沒想像中難，判斷的時候都加個for迴圈即可。

第二個問題：vga 邏輯

這邊由於牽扯到複雜的vga計算，踩地雷分成底圖def\_map與當前地圖is\_flap\_or\_flip (4\*8的地圖 )，picture是裝著大量圖片的rgb資料，有8個圖片的在裡面。在前期我們犯錯了幾次導致螢幕顯示不正確，後來對好v\_cnt與h\_cnt順利處理完地圖顯示。

第三個地圖 : 計算炸彈數量破圖

我們的做法是佈置好炸彈，將炸彈周圍八個非炸彈的位置都加 1，這邊沒什麼問題，但實際玩的時候會偶爾出現破圖的情況，也就是有格子會變成黑色或奇怪的形狀。(下圖為計算數字的部分程式碼)

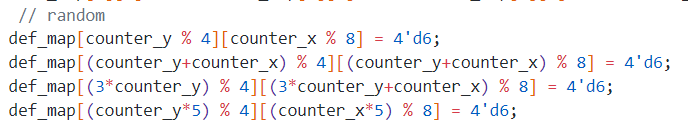


Def\_map為底圖，我們底圖是在STATE\_IDLE建立，一開始都會先設為0，接著跑隨機生成炸彈，而這些東西都是寫在always(\*)裡面，原本計算數字寫在STATE\_NORMAL，但在STATE\_NORMAL未進行歸零，def\_map可能會累加上去，最後溢位跑出奇怪的東西，最後我們把這些計算的地方也搬到STATE\_IDLE解決破圖的問題。

第四個問題 : 隨機生成地雷

我們的目標是生成至多10個地雷，且希望每次生成的地雷盡可能隨機，我們想到用很大的counter的值做10種不同計算，並且將結果%4及%8，也就是對應到4\*8的地圖中哪10個點為地雷，而由於點可能會重複，因此不一定會剛好10個地雷。

一開始的counter不小心寫錯，以為是其他地方的問題找很久沒找到，後來打掉重寫後順利解決。



未解之謎 : 沙盒模式

沙盒模式中，要可以自己布置地雷並自己遊玩，我們完成了自己布置的部分，但按下按鈕到遊玩時，底圖又變回NORMAL的隨機分布，經過長時間的debug及重寫仍未找出問題。

1. **專案實作完成程度**

<完成>

1. 踩地雷(keyboard＋VGA)
2. 自動生成地雷
3. 根據地雷位置生成附近的數字提示
4. 連環踩鄰近非爆炸位置設計
5. LED、7-segment等運用

<未完成>

Only 沙盒模式

1. **難易度說明**
2. LED燈以及segment控制：★★☆☆☆

用來檢查state以及win或lose時的一些小特效。

由於地圖會隨機生成地雷，開始遊戲後七段顯示器會顯示這份地圖有幾個地雷，對應到可以插幾個棋子，插旗子數字會依序減少，並開始倒數計時99秒。

1. Keyboard控制：★★★☆☆

稍不同於之前的lab，這次我們用到keynum\_x以及keynum\_y去紀錄案件對應到的踩地雷位置，另外實作shift＋案件作為插旗功能也耗一番功夫。

1. VGA勝利與失敗畫面：★★☆☆☆

根據state去顯示畫面，失敗畫面有實作負片畫面與正片畫面的切換作為踩到地雷爆炸的特效。使用80\*60的解析度去顯示以節省空間。

1. VGA顯示方式：★★★★☆

使用9種15\*10的小圖（未踩過、1、2、3、4、芋、地雷、棋子、踩過），並將這些資料9\*15\*10=1350的顏色資料存放在一維陣列中，原本想要用二維陣列，但初始化超級不方便所以改用一維。用這種小圖重複顯示的方法可以很有效率地減少資源使用太多的問題。

1. 踩地雷：★★★★☆

使用二維陣列[3:0]def\_map[0:3][0:7]當作隨機生成或是沙盒佈置過後的底圖，並用[3:0]is\_flip\_or\_flag[0:3][0:7]去記錄32不同位置是否被踩過或是插旗子，但是module貌似不支援用二維陣列當一個port，所以這邊也卡滿多時間。

1. 地雷連續爆炸：★★★☆☆

原本想要用遞迴的方式去集體更新相鄰的is\_flip\_or\_flag，後來改成使用迭代取代，去找地雷的位置再更新附近非地雷非棋子區域的is\_flag\_or\_flip成def\_map。

1. 地雷隨機分配以及附近數字提示生成：★★★★★

預先在IDLE時將背景地圖def\_map的資料初始化成0，此時設計一個counter作為亂數生成丟進10個不同的function跑出10個不同的座標counter\_x、counter\_y，將def\_map對應座標的資料更新成炸彈，再使用迭代尋找炸彈的格子，將附近的格子資料從0累加上去，最多到4d5，對應到「芋」。這部分比較接近軟體，實作起來安全感比較不足。

1. **心得討論**

完成後真的是成就感滿滿，蹲在資電館數日滴下的苦水都顯得很十分有意義，看到自己完成出一個小時候常常在玩的遊戲，實質的體驗到這堂課給我們帶來的成長，原來我們是有能力將學習到的知識結合創意，在不斷試錯下完成我們的最終成品。儘管最後有些碰壁，不代表我們花費數十小時在沙盒模式是浪費時間，大膽地將自己的想法付諸於程式碼之中，並尋找最佳解，時時刻刻關注fpga板的資源會不會被耗盡，這次的project大大的訓練我們的創意力、專注力、細心程度，以及最重要的合作能力。

一份任務從零到有，最難熬的就是最一開始的起跑點，慶幸我們兩個合作屬實有效率，當一個人疲倦或是江郎才盡時，另一位組員總是能接棒，彼此都很信任彼此的能力。

感謝這次的project，讓我們能統整我們所學習到的學問並加以運用，而不是死板的學習應知識，看到不同組極具創意的作品，也讓大家良性競爭，激勵每一位學生所沒被發現的潛能，雖然這學期確實花最多精力在這堂課，但相對的也學習到最多，十分扎實又有趣味，屬實好課！

預祝教授以及助教新年快樂！

1. **分工**

楊承霖：LED(跑馬燈)、STATE、7segment(地雷數及倒數)、自動生成mine number hint

李晉方：random、地雷連環爆、keyboard(含插旗)、翻牌

共同製作：VGA scence(win scene + negflim)、map init、DEBUGGGGGGG、(sand mode)

1. **課程外比重**

負片映像，隨機生成地雷、多維陣列的應用、一些複雜的數據計算(連環爆、自動探測附近地雷數)

課程外比例：大約佔20%

自行完成比例：99%

1. **測試完整度**

90%~100％