高中（職）組成果報告表單

題目名稱：懷舊遊戲的內部程式碼研究與開發

一、摘要：

本研究探討了經典遊戲 Pac-Man 內部的程式碼以及幽靈是否可以自行尋找目標，為了讓幽靈可以尋找到 Pac-Man 的定位，我們將地圖坐標化，讓幽靈追逐目標，並排除幾種遇到的障礙與困難，使程式順利執行。

二、探究題目與動機

日本的南夢宮公司製作了一款街機遊戲 - Pac-Man，並於 1980 年 5 月 22 日當天在日本發行。Pac-Man 在 1980 年代風靡全球，被認為是最經典的街機遊戲之一。在 2010 年 5 月 21 日協調世界時 15 時左右，為紀念遊戲誕生 30 周年，Google 標誌改為遊戲圖片，而點擊 Insert Coin 按鈕或等候十秒即可進行線上遊戲。我們希望能透過 processing 開發一個類似小精靈的遊戲，並將遊戲功能與介面優化。

三、探究目的與假設

本研究主要利用 processing 來開發程式碼。

1. 探討 Pac-Man 程式碼。
2. 探討 Pac-Man 的動畫製作。
3. 探討豆豆函數的製作。
4. 探討幽靈是否可以自行尋找目標。

四、探究方法與驗證步驟

一、Pac-Man 的程式碼

我們定義鍵盤上的 W、S、A、D 分別控制 Pac-Man 的上、下、左、右，所以當我們按下四個按鍵中的其中一個時，將會改變 Pac-Man 的移動方向，接著我們利用記憶鍵盤的按鍵來控制 Pac-Man 的方向，並且檢測它的定位，其程式碼如圖 1 所示。

```
236  if(x0>=580 && x0<=720 && y0>=715&&y0<=725){
237    if((key=='a')||(key=='A')||(keyCode==LEFT)){a=3;m=580;y0=720;}
238    if((key=='d')||(key=='D')||(keyCode==RIGHT)){a=4;m=720;y0=720;}
239      q=20;}
240
241
242  if(x0>=20 && x0<=720 && y0>=800&&y0<=810){
243    if((key=='a')||(key=='A')||(keyCode==LEFT)){a=3;m=20;y0=805;}
244    if((key=='d')||(key=='D')||(keyCode==RIGHT)){a=4;m=720;y0=805;}
245      q=21;}
246
247
248
249  if(x0>=15&&x0<=25 && y0>=20 && y0<=215) {
250    if((key=='w')||(key=='W')||(keyCode==UP)){a=1;m=20;x0=20;}
251    if((key=='s')||(key=='S')||(keyCode==DOWN)){a=2;m=215;x0=20;}
252    w=1;
253  }
254  if(x0>=15&&x0<=25 && y0>=550 && y0<=635) {
255    if((key=='w')||(key=='W')||(keyCode==UP)){a=1;m=550;x0=20;}
256    if((key=='s')||(key=='S')||(keyCode==DOWN)){a=2;m=635;x0=20;}
257     w=2; }
```

圖 1：判斷 Pac-Man 方向和定位的程式碼（來源：研究者自製）

二、Pac-Man 的動畫製作

每個 processing 程式都會計數自從它開始執行到現在經過了多少時間。它會以毫秒（千分之一秒）為計數單位，所以 1 秒之後，計數值為 1000；5 秒之後，計數值為 5000；1 分鐘後，計數值為 60,000。我們可以使用這個計數器在特定的時間觸發動畫。mills()函式可用於傳回此計數值，所以我們利用計時器以及同餘的方式，判斷 Pac-Man 的開口大小，如下圖 2 所示，其開口變化如下圖 3 所示。

```
399   if(a==3){
400     if((t/100)%4==1)
401       pac=loadImage("dot3.png");
402             if((t/100)%4==2)
403       pac=loadImage("dot.png");
404          if((t/100)%4==3)
405       pac=loadImage("dot3.png");
406          if((t/100)%4==0)
407       pac=loadImage("dot31.png");
408   if(x0>m)
409   x0-=sp0;
410   }
```

圖 2：判斷 Pac-Man 開口大小的程式碼（來源：研究者自製）

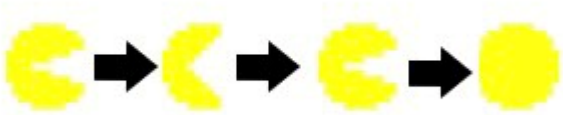

圖 3：Pac-Man 的開口變化（來源：研究者自製）

三、豆豆函數的製作

我們利用座標定位設定走道兩端的座標，並利用記憶陣列和迴圈來製作豆豆，如圖 4、圖 5。

```
923 void beenh(int xb1,int xb2,int y,int r){
924   fill(#FAD43A);
925 for(int i=xb1+20;i<=xb2+25;i+=20){
926 if( a1[(i-xb1-20)/20+r]==1){
927   rect(i,y+20,10,10);
928 }
929 if(i>=x0&&i<=x0+50&&y+20>=y0&&y+20<=y0+50){
930   a1[(i-xb1-20)/20+r]=0;
931   }
932 }
933 }
934
935 void beens(int x,int yb1,int yb2,int r){
936   fill(#FAD43A);
937 for(int i=yb1+40;i<=yb2;i+=20){
938 if( a2[(i-yb1-40)/20+r]==1){
939   rect(x+20,i,10,10);
940 }
941 if(x>=x0&&x<=x0+50&&i+20>=y0&&i+20<=y0+50){
942   a2[(i-yb1-40)/20+r]=0;
943  }
944
945 }
946 }
```

圖 4：豆豆函數的程式碼（來源：研究者自製）

```
55 beenh(20,720,805,178);
56 beenh(20,75,635,214);
57 beenh(665,720,635,218);
58 beenh(245,325,215,222);
59 beenh(410,495,215,227);
60 beenh(245,325,720,232);
61 beenh(410,495,720,237);
62
63 beens(20,20,130,0);
64 beens(160,20,130,4);
65 beens(325,20,130,8);
66 beens(410,20,130,12);
67 beens(580,20,130,16);
68 beens(720,20,130,20);
69 beens(20,130,215,24);
```

圖 5：呼叫豆豆函數（來源：研究者自製）

四、幽靈自行尋找 Pac-Man 的定位

我們為了讓幽靈能夠自行尋找 Pac-Man 的定位，我們先將地圖坐標化，並把每條橫向、縱向走道規劃為掃描區，下圖 6 是坐標化後的地圖。

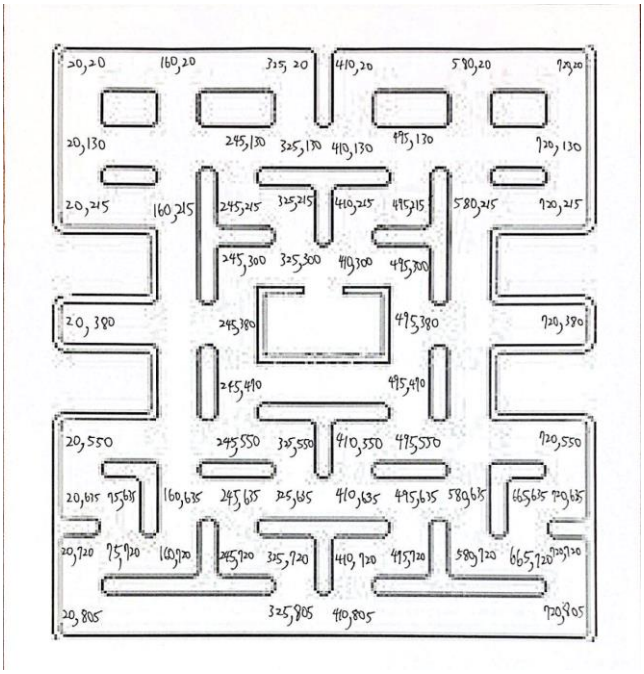

圖 6：坐標化的地圖（來源：研究者自製）

我們發現，如果程式判定 Pac-Man 和幽靈只有 x 坐標或 y 坐標相同，但兩者所在的掃描區卻不同，則幽靈將會停在原地不動，如下圖 7 所示。

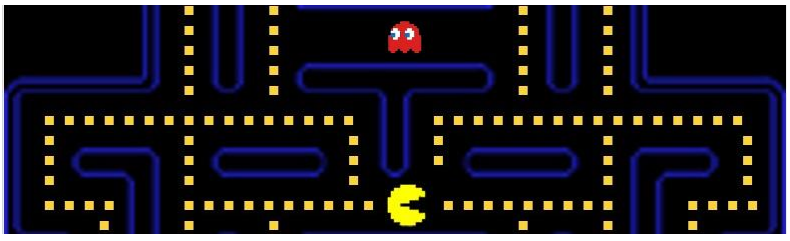

圖 7：幽靈的 x 坐標已跟小精靈相同，但 y 坐標尚未與小精靈相同（來源：研究者自製）

為了解決這個問題，我們設定若 Pac-Man 或幽靈 x 坐標相同 y 坐標不同所在的掃描區

不同；x 坐標不同 y 坐標相同且所在的掃描區不同，則讓程式測定橫向、縱向通道，判定幽靈較靠近掃描區的頭或尾，並且將尋找目標先鎖定成靠近頭或尾，如圖 8、圖 9。

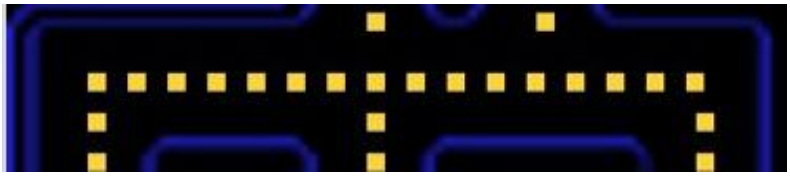

圖 8：橫向通道（來源：研究者自製）

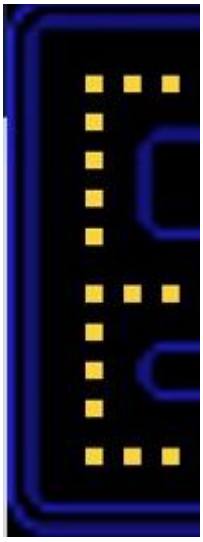

圖 9：縱向通道（來源：研究者自製）

接著，我們發現在某些特定轉角處，將會超出掃描區的範圍，如下圖 10 所示。

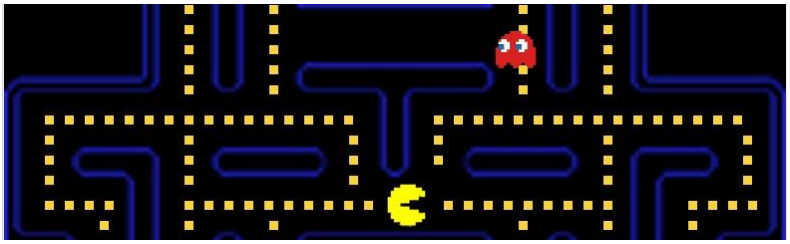

圖 10：幽靈超出掃描區（來源：研究者自製）

這個問題是由於在轉角處時，我們是使用先判定後變化的方法，但改成先變化後判定時，在這個方法執行時幽靈不會變化，所以改成在執行第一次時，先判定後變化，而之後都改成先變化後判定的方法，如此程式便可正常運作。

五、結論與生活應用

我們在本次的研究中發現了幾個尚未解決的問題。第一，因為我們的語法還不夠熟練，所以我們在有些地方只能用土法煉鋼的方式，這樣會造成我們定義許多不必要的變數；再來，因為我們的技術還不是很成熟，加上時間的不足，所以我們的幽靈只寫了一隻，未來如果我們有更多時間的時候，我們希望可以做出更多的幽靈，並且他們追逐的目標都不同；還有，因為時間上的不足，所以我們沒有比照原始的遊戲開發出大力丸，在未來我們希望可以開發出大力丸。

本研究探討了經典遊戲 Pac-Man 內部的程式碼以及幽靈是否可以自行尋找目標，為了讓

幽靈可以尋找到 Pac-Man 的定位，我們將地圖坐標化。並且讓幽靈不要因為跟 Pac-Man 的 x 坐標相同 y 坐標不同或是 y 坐標相同 x 坐標不同而停止行動，所以當上述行為發生時，我們將他的目標暫時轉移。以及使幽靈不要超出掃描區，我們將第一次執行與後面的執行不同。經過這次的研究，我們發現它可以推廣到比如說：自動駕駛的判斷障礙物系統、製作無線項圈，幫助飼主尋找寵物的位置以及幫助視力不好的人可以更方便尋找他的眼鏡。

參考資料

1. Casey Reas, Ben Fry（2020）。**Processing 入門**（第二版，蔣大偉譯）。臺北市：碁峰。
2. 唐元亮（2018）。Phaser -- HTML 遊戲框架。2021 年 01 月 15 日，取自 http://yltang.net/tutorial/phaser/6/。
3. Moby Games（年份不詳）。吃豆人。2021 年 01 月 15 日，取自 https://www.mobygames.com/game/pac-man。