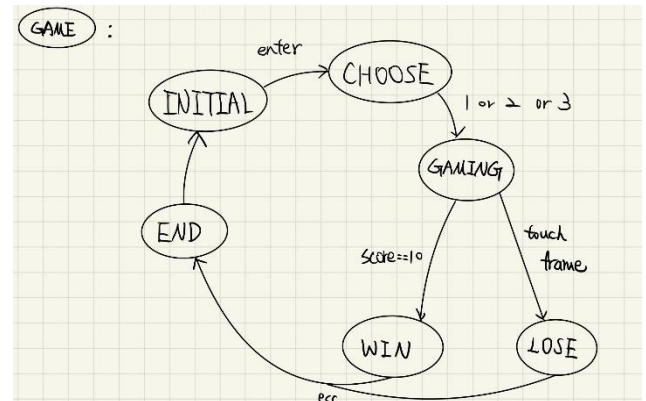
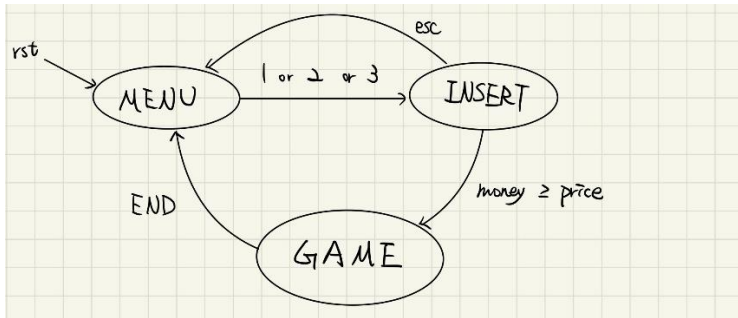


# Final Project Report

學號: 109000204、109000205

姓名: 王翔明、蕭皓隆

## ◎ State Diagram

















## ◎ 設計概念

一開始先進入主畫面，主畫面中可以選擇玩遊戲的背景音樂，共有三首歌，以鍵盤 1、2、3 選擇，選擇完後在有標註相對應數字的投幣孔(實體)投下代幣。接著進入準備畫面(顯示遊戲名稱)，(此時音樂已經開始放)，按下鍵盤上 enter 後再進入選擇貪食蛇身體顏色的畫面，共有三種顏色可以選擇，並以鍵盤上的 1、2、3 按鍵做選擇。按完按鍵後會開始遊戲。遊戲進行的畫面是由兩條亮綠色直線來表示貪食蛇活動空間範圍大小，在範圍內會隨機出現紅色點(蘋果)，在畫面左邊及右邊會顯示分數(當分數為奇數顯示在左邊，偶數顯示在左邊)。遊戲得分方式為貪食蛇的頭(紅色)吃掉蘋果，當吃掉一顆蘋果貪食蛇的身體長度會多加一節，而身體長度就是分數，得到 10 分即是過關成功(進入顯示 win 的畫面，音樂也會繼續播放)。當貪食蛇的身體範圍跑出限制的空間大小，或是當貪食蛇碰到自己的身體，即是遊戲失敗(進入 over 的畫面，音樂也會繼續播放)。在 win 及 over 的畫面時按鍵盤上的 esc 按鍵 就會再回到一開始選擇音樂的主畫面，若要再次進行遊戲則需再次投幣。



## ◎ 架構細節

我們將不同的部分用個別的 `module` 表示，共 14 個。

-  105C.v
-  apple\_gen.v
-  Final\_Project.v
-  game.v
-  Havana.v
-  keyboard\_ctrl.v
-  KeyboardDecoder.v
-  LightlyRow.v
-  OnePulse.v
-  pixel\_gen\_snake.v
-  PWM.v
-  snake\_ctrl.v
-  sonic.v
-  speaker\_control.v

105C.v 、 Havana.v 、 LightlyRow.v，為音樂檔。將音樂譜上的音符用對應的頻率表示，將左右喇叭分開寫。

## ▶ 105C.v

左右頻率分別都有八個小節，用 16 個 `beatNum` 去表示一個小節，共會有 512 個 `beatnum`。控制 `clk` 的速度，讓 `beatnum` 從 0 跑到 511 的速度符合歌曲的節奏。用 `case` 的方式從 0 到 511，將每個音串接起來，串成一首歌。

以下是第一小節的程式碼。而其他小節的差異處只是在音符不同所以給的頻率(變數)不同。

```
12'd0: toneR = `sil;      12'd1: toneR = `sil; // HG (half-beat)
12'd2: toneR = `sil;      12'd3: toneR = `sil;
12'd4: toneR = `sil;      12'd5: toneR = `sil;
12'd6: toneR = `sil;      12'd7: toneR = `sil;
12'd8: toneR = `sil;      12'd9: toneR = `sil; // HE (half-beat)
12'd10: toneR = `sil;     12'd11: toneR = `sil;
12'd12: toneR = `sil;     12'd13: toneR = `sil;
12'd14: toneR = `sil;     12'd15: toneR = `sil;
```

## ▶ apple\_gen.v

can 為傳入 module 的變數，當 can 為 1 時，即會產生蘋果(在螢幕上會顯示為紅色方格)。當值為 0 時將座標表示為 10'b11111\_11111，意即不會顯示在螢幕上。十個 reg 相連組成，產出一組 10 bits 的蘋果座標，並把 y 座標除以 23(場地高 24)，進一步防止蘋果超出遊戲範圍。can 從 snake\_ctrl 模組接入，當訊號為 posedge can 時，將訊號往前推移一格，產生下一組新座標 apple[9:0]，來達成亂數的效果。

```
always @(posedge can) begin
    if({dff_0,dff_1,dff_2,dff_3,dff_4,dff_5,dff_6,dff_7,dff_8,dff_9}==10'b00000_00000)
begin
    {dff_0,dff_1,dff_2,dff_3,dff_4,dff_5,dff_6,dff_7,dff_8,dff_9}<=10'b11111_11111;
    end
    else begin
        dff_0 <= in_0;
        dff_1 <= in_1;
        dff_2 <= in_2;
        dff_3 <= in_3;
        dff_4 <= in_4;
        dff_5 <= in_5;
        dff_6 <= in_6;
        dff_7 <= in_7;
        dff_8 <= in_8;
        dff_9 <= in_9;
    end
end
always @(can) begin
    if(can) begin
        apple={dff_0,dff_1,dff_2,dff_3,dff_4,{dff_5,dff_6,dff_7,dff_8,dff_9}%5'd23};
    end
    else begin
        apple = 10'b11111_11111;
    end
end
end
```



此為主要控制遊戲內 state 的 module，分別控制選擇音樂的主畫面進入準備畫面、遊戲畫面再到主畫面的條件。一開始設定為 **INITIAL**(選擇音樂的主畫面)再跳入 **CHOOSE**(選擇蛇顏色的畫面)，當按下鍵盤 1、2、3 其中一個按鍵時會跳入 **GAMING**(開始遊戲)，接著再依據得分來判斷下個 state 是 **WIN** 還是 **OVER**。兩個 state 在按下鍵盤上的 esc 後都會跳入 **END state** 再跑回 initial。會多設立一個 **END state** 是因為在這個 module 會傳出 Choose(所選擇的顏色)給 pixel\_gen\_snake、over、finish、state 跟 score(分數)。

以下是 case 的其中部分

```
case(state)
  INITIAL: begin
    if(enter==1 && State==GAME) begin
      choose_next = 2'd0;
      next_state = CHOOSE;
    end
    else begin
      choose_next = 2'd0;
      next_state = state;
    end
  end
  CHOOSE: begin
    if(one) begin
      choose_next = 2'd1;
      next_state = GAMING;
    end
    else if(two) begin
      choose_next = 2'd2;
      next_state = GAMING;
    end
    else if(three) begin
      choose_next = 2'd3;
      next_state = GAMING;
    end
    else begin
      choose_next = choose;
      next_state = state;
    end
  end
end
```

## ▶ pixel\_gen\_snake.v

是透過螢幕畫面上的 XY 座標控制每個 state 畫面所顯示的內容。

這裡的 state 是 **module game** 中所傳出來的，兩者會相同。在 **INITIATE state** 時會顯示數字 37 及英文 coin snake 再加上蛇的圖案。

在 **GAMING state** 會顯示的是標示場地大小的兩條直線、蘋果(紅色點)、得分數、貪食蛇。貪食蛇總共是由 9 個方格所組成在分數為多少時顯示相對應的長度，在判斷式(if)中有一個變數 len(長度)判斷顯示身長。得分數的部分，奇數顯示在左邊，偶數顯示在右邊。

當分數為 10 時會跳入 **WIN state**。下面的 else if 是表示當蛇頭碰到自己時(當蛇頭的座標和身體座標相同)或碰到場地邊界時會跳入 **OVER state**。

```
GAMING: begin
    if(score == 6'd10) begin
        choose_next = choose;
        next_state = WIN;
    end
    else if((snake0==snake4 && length>6'd4) || (snake0==snake5 && length>6'd5)
    || (snake0==snake6 && length>6'd6) || (snake0==snake7 && length>6'd7)
    || (snake0==snake8 && length>6'd8) || (snake0==snake9 && length>6'd9)
    || (snake0[4:0]>5'd23) || (snake0[9:5]==5'd31 && snake1[9:5]==5'd0)
    || (snake0[9:5]==5'd0 && snake1[9:5]==5'd31)) begin
        choose_next = choose;
        next_state = OVER;
    end
    else begin
        choose_next = choose;
        next_state = state;
    end
end
```

在 **WIN state** 會顯示邊框及英文字母 win 還有蛇的圖示。在 **OVER state** 會顯示邊框及英文字母 over。

在 **CHOOSE state** 會顯示邊框及數字 1、2、3、三顏色不同的蛇圖式。

以下是數字 3 的表示方法，當 XY 座標，在設定的範圍內時會顯示所預設的顏色，這邊是設定 12'fff(白色)。

```
else if((( 10'd0 + 10'd310 <=h_cnt && h_cnt< 10'd24 + 10'd310 ) && ( 10'd48 + 10'd300
<=v_cnt && v_cnt< 10'd60 + 10'd300 ) )||
    (( 10'd0 + 10'd310 <=h_cnt && h_cnt< 10'd24 + 10'd310 ) && ( 10'd72 + 10'd300
<=v_cnt && v_cnt< 10'd84 + 10'd300 ) )||
    (( 10'd0 + 10'd310 <=h_cnt && h_cnt< 10'd24 + 10'd310 ) && ( 10'd96 + 10'd300
<=v_cnt && v_cnt< 10'd108 + 10'd300 ) )||
    (( 10'd16 + 10'd310 <=h_cnt && h_cnt< 10'd24 + 10'd310 ) && ( 10'd48 + 10'd300
<=v_cnt && v_cnt< 10'd108 + 10'd300 ) )
    )//3
begin
    {vgaRed, vgaGreen, vgaBlue} = 12'hfff;//white
end
```

## ▶ snake\_ctrl.v

是控制貪食蛇在螢幕上所顯示的位置。Up Down Left Right 分別是鍵盤上的 WSAD 按鍵所控制。apple[9:0] 由 module apple\_gen 傳入蘋果生成座標，如果蛇頭和蘋果座標一樣，蛇就要變長(length+1)，蘋果要消失

其中 d 代表方向(direction)，以下程式碼為當按下 WSAD 按鍵後改變 d\_next 的值，再透過 case 的方式改變蛇頭的 XY 座標。

```
always @(Up or Down or Right or Left) begin
    if(Right && snake0[9:5]!=snake1[9:5]-1'b1 && d!=LEFT) begin
        d_next = RIGHT;
    end
    else if(Left && snake0[9:5]!=snake1[9:5]+1'b1 && d!=RIGHT) begin
        d_next = LEFT;
    end
    else if(Down && snake0[4:0]!=snake1[4:0]-1'b1 && d!=UP) begin
        d_next = DOWN;
    end
    else if(Up && snake0[4:0]!=snake1[4:0]+1'b1 && d!=DOWN) begin
        d_next = UP;
    end
    else begin
        d_next = d;
    end
end
```

```
always @* begin
    case(d)
        UP: begin
            snake_head[4:0] = snake0[4:0] - 1;
            snake_head[9:5] = snake0[9:5];
        end
        DOWN: begin
            snake_head[4:0] = snake0[4:0] + 1;
            snake_head[9:5] = snake0[9:5];
        end
        RIGHT: begin
            snake_head[4:0] = snake0[4:0];
            snake_head[9:5] = snake0[9:5] + 1;
        end
        LEFT: begin
            snake_head[4:0] = snake0[4:0];
            snake_head[9:5] = snake0[9:5] - 1;
        end
        default: begin
            snake_head[4:0] = snake0[4:0];
            snake_head[9:5] = snake0[9:5];
        end
    end
```

```
    endcase  
end
```

以下為初始化貪食蛇一開始的位置。而貪食蛇的每一節座標皆為前面一節在上個 clk 的座標位置，因此我們只要改變蛇頭的位置，後面的每一節就會依照前一節來移動。

```
always @(posedge CLK or posedge rst) begin  
    if(rst) begin  
        snake0[9:5] <= 10'd10;  
        snake0[4:0] <= 10'd0;  
        snake1 <= 10'b1111111111;  
        snake2 <= 10'b1111111111;  
        snake3 <= 10'b1111111111;  
        snake4 <= 10'b1111111111;  
        snake5 <= 10'b1111111111;  
        snake6 <= 10'b1111111111;  
        snake7 <= 10'b1111111111;  
        snake8 <= 10'b1111111111;  
        snake9 <= 10'b1111111111;  
    end  
    else begin  
        snake0 <= snake_head;  
        snake1 <= snake0;  
        snake2 <= snake1;  
        snake3 <= snake2;  
        snake4 <= snake3;  
        snake5 <= snake4;  
        snake6 <= snake5;  
        snake7 <= snake6;  
        snake8 <= snake7;  
        snake9 <= snake8;  
    end  
end
```

當蛇身長(分數)3 時超過改變 frequency，並把 frequency 傳入 PWM\_gen 的 module 裡面改變頻率，讓蛇的移動速度變快。

```
always @* begin  
    if(3 <= length && length <= 9) begin  
        frequency = length + 5'd3;  
    end  
    else begin  
        frequency = 5'd3;  
    end  
end
```

## ▶ final\_project.v

這裡是所謂的 top module，主要是控制 7-segment 和 LED 燈並且串聯所有 module 然後控制一開始的選擇音樂畫面以及控制音樂價錢並決定投幣價錢。值得一提的是因為分別有三首歌，而每首歌的音頻以及 clock 不盡相同，所有要特別用變數去控制，才能達到理想的效果，並且要讓其在玩遊戲的 state 時才能播放。

```
assign freq_outL = 50000000 / freqL;
assign freq_outR = 50000000 / freqR;

assign freqL = (song==4'd1) ? 2*freqL_105 :
               (song==4'd2) ? freqL_havana :
               (song==4'd3) ? freqL_LR : `sil;

assign freqR = (song==4'd1) ? 2*freqR_105 :
               (song==4'd2) ? freqR_havana :
               (song==4'd3) ? freqR_LR : `sil;

wire en;
assign en = (state==GAME) ? 1 : 0;

wire clk_song;
assign clk_song = (song==4'd1) ? clk_21 :
                  (song==4'd2) ? clk_22 : clk_22;
```

下方則是主要的 State 切換，並且遊戲會回傳 STATE(遊戲中的狀態)，若為 END 則要讓 top module 的 state 切換至未投幣的狀態，若想要再次進行遊戲，則需再次投幣。

```
case(state)
    MENU: begin
        if(song != 4'd0) begin
            next_state = INSERT;
        end
        else begin
            next_state = state;
        end
    end
    INSERT: begin
        if((money0 >= price0 && money1 >= price1) || (money1 > price1)) begin
            if(money0 != 4'd0 || money1 != 4'd0) begin
                next_state = GAME;
            end
            else begin
                next_state = state;
            end
        end
        else if(esc) begin
            next_state = MENU;
        end
    end
```



```
        else begin
            next_state = state;
        end
    end
end
GAME: begin
    if (STATE == END) begin
        next_state = MENU;
    end
    else begin
        next_state = state;
    end
end
default: next_state = MENU;
endcase
```

## ◎ 實作完成度/難易度/測試完成度

我們認為實作的完成度有 90%，而真正測試的完成度只有 80%，在我們的 proposal 有提到投幣的部分，這也算是蠻重要的一個環節，但我們沒有順利做出理想中的投幣裝置，過程中也發生蠻多小插曲，也花了不少時間 debug。

## ◎ 困難與解決方法

我們一開始打算先使用一個超聲波感測器，測試成功後再把相同方法套用至其他兩個超聲波感測器，但後來發現一開始測試使用的超聲波感測器是壞的，所以無法正常作動，我們因此寫了一個可以符合預期的判斷式，但其他兩個好的超聲波感測器卻反而無法達到我們想要的效果。

我們也試了許多方法嘗試解決投幣的問題，例如設計各種物理構造來彌補不足的地方，或是利用 clock 的延遲來抓訊號，但種種方法都無法達到預期的要求，也花了蠻多時間在這上面。

在我們決定更換投幣模組時也上網看了許多 DIY 投幣機的資料，但卻發現他們所使用的感測模組在學校附近的電子材料行都沒有賣，所以我們只能硬著頭皮繼續修改超聲波感測器。

經過這次失敗，我們認為最主要的原因還是因為模組的選擇錯誤，導致無法達成想要的效果，若下次還有機會，會多方嘗試各種模組。

## ◎ 分工

**109000205 蕭皓隆：**設計構想、貪食蛇設計、貪食蛇實作、鍵盤控制、音效模組控制、投幣裝置製作、超聲波訊號處理、撰寫報告

**109000204 王翔明：**設計構想、VGA 畫面繪製、撰寫音樂頻率、撰寫報告

## ◎ 心得討論

對於這次能順利做出貪食蛇小遊戲我們覺得還蠻開心的，雖然投幣的部分沒有順利做出來，彌補的方式也不夠明確，但還是覺得蠻有成就感的！這個遊戲結合了這學期所學的蠻多東西，例如 7-

segment、VGA 顯示、音效模組以及鍵盤控制等等，雖然這學期在此科目真的學的蠻辛苦也花蠻多時間做作業，但目前回頭看也覺得一切都值得了，畢竟這學期學到的東西在這次 Final Project 都有用上！

在投幣部分的環節雖然沒有達成目標，但我們也在 Debug 和修正的過程中學到蠻多東西的，也代表有從失敗當中得到收穫，若還有機會遇到這樣的問題，我們應該會重新選擇感測模組(紅外線、光遮模組等等...)來達到投幣的效果

實作 Project 的過程中，我們也體會到硬體的現實面，例如他不一定會照你原本的想法走，這需要程式碼以及物理構造等等的都有相對應配合上才有辦法達到預期的效果，確實是很值得學習的環節。

最後，還是要謝謝老師與助教這學期的教導、指導，這堂課可以很明顯感受到老師的幽默以及教學熱忱，這學期學到最多東西而且可以具現化的非邏輯設計莫屬了！老師與助教們這學期都辛苦了～