Let's Play

PACMAN

START

RULES

01 เก็บดาวไปเรื่อยๆ

02 พยายามหลบหลีกไม่ให้โดนผี หากโดน ❤ จะลดลงไป 1 ดวง(มี 3 ดวง)

03 เมื่อหัวใจเหลือ 0 ก็จะ game over (แพ้)

EQUIPMENT

01 บอร์ด FPGA

02 จอแสดงผลที่รองรับ VGA Port

03 สายเชื่อมต่อ USB to micro USB

ROM

- การแสดงผลของเกมมีความกว้าง 640 pixel และความสูง 480 pixel
- รวมทั้งหมด 307,200 pixel
- ใช้ภาษา Python ในการ generate rom

ROM

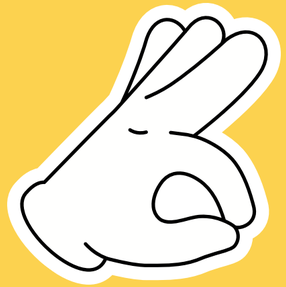
ในการเก็บ ROM ของทุกๆ ภาพ เราจะเก็บค่าสีในเลข 12 Bit ไว้ใน templete เดียวกันหมด ดังรูปภาพ

```
module rom
    (
        input wire clk,
        input wire [3:0] row,
        input wire [3:0] col,
        output reg [11:0] color_data
    );

    (* rom_style = "block" *)

    //signal declaration
    reg [3:0] row_reg;
    reg [3:0] col_reg;

    always @(posedge clk)
        begin
        row_reg <= row;
        col_reg <= col;
        end

    always @*
    case ({row_reg, col_reg})
        6'b000000: color_data = 12'b011011011110;

        ...

        default: color_data = 12'b000000000000;
    endcase
endmodule
```

CODING

Cool!

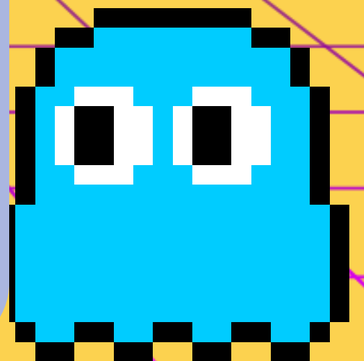
PYTHON : TO GENERATE A ROM MEMORY

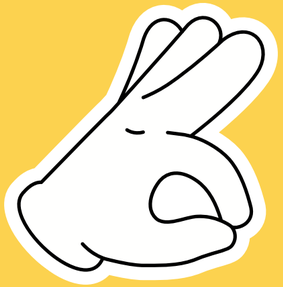


ทำหน้าที่ : แปลงภาพให้เป็นเลข 12 บิต และสร้างเทมเพลต ROM ในรูปแบบของไฟล์ VERILOG ด้วยการใช้ไลบรารี PIL (PYTHON IMAGING LIBRARY)



```python
from PIL import Image

def convert_to_12_bit_color(pixel):
    if isinstance(pixel, tuple):
        # RGB mode
        r, g, b = pixel
    else:
        # Grayscale mode
        r, g, b = pixel, pixel, pixel

    # Convert RGB values to 12-bit color
    r = (r >> 4) & 0b1111
    g = (g >> 4) & 0b1111
    b = (b >> 4) & 0b1111
    return (r << 8) | (g << 4) | b

def generate_rom_template(image_path):
    # Open the image
    img = Image.open(image_path)

    # Convert to RGB mode if not in RGB
    img = img.convert('RGB')

    # Resize the image to 16x16 pixels (width x height)
    img = img.resize((16, 16))

    # Get pixel data
    pixels = list(img.getdata())

    # Generate ROM template
    rom_template = ""
    for i, pixel in enumerate(pixels):
        color_data = convert_to_12_bit_color(pixel)
        rom_template += f"8'b{i:08b}: color_data = 12'b{color_data:012b};\n"

    return rom_template

# Example usage
image_path = "star.png"
rom_template = generate_rom_template(image_path)

# Save the ROM template to a file
with open("star_rom.v", "w") as f:
    f.write(rom_template)
```



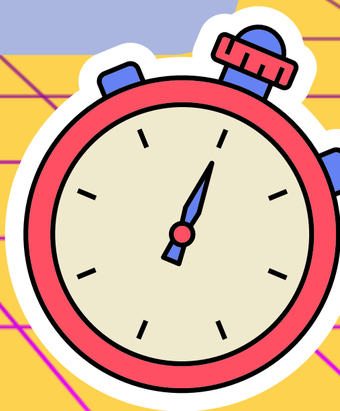


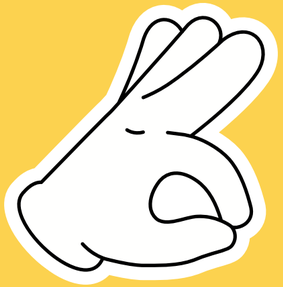
ฟังก์ชั่น : CONVERT_TO_12_BIT_COLOR

ทำหน้าที่ : แปลงค่าสีของแต่ละพิกเซลในรูปภาพเป็นเลข 12 บิต โดยดึงค่า สีแดง (R), สีเขียว (G), สีน้ำเงิน (B) จากพิกเซล และประเภท GRAYSCALE ใช้ค่าเดียวกัน และรวมค่าสีให้เป็นเลข 12 บิตเพื่อคืนค่า

ตัวอย่าง CODE

```python
from PIL import Image

def convert_to_12_bit_color(pixel):
    if isinstance(pixel, tuple):
        # RGB mode
        r, g, b = pixel
    else:
        # Grayscale mode
        r, g, b = pixel, pixel, pixel

    # Convert RGB values to 12-bit color
    r = (r >> 4) & 0b1111
    g = (g >> 4) & 0b1111
    b = (b >> 4) & 0b1111
    return (r << 8) | (g << 4) | b

```





ฟังก์ชั่น : GENERATE_ROM_TEMPLATE

ทำหน้าที่ : รับที่อยู่ไฟล์ภาพและแปลงเป็น RGB 16X16 พิกเซล เก็บเลข 12 บิต ใน ROM TEMPLETE (VERILOG)

ตัวอย่าง CODE

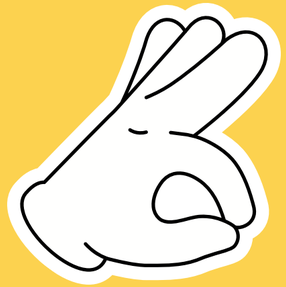
```
def generate_rom_template(image_path):
    # Open the image
    img = Image.open(image_path)

    # Convert to RGB mode if not in RGB
    img = img.convert('RGB')

    # Resize the image to 16x16 pixels (width x height)
    img = img.resize((16, 16))

    # Get pixel data
    pixels = list(img.getdata())

    # Generate ROM template
    rom_template = ""
    for i, pixel in enumerate(pixels):
        color_data = convert_to_12_bit_color(pixel)
        rom_template += f"8'b{i:08b}: color_data = 12'b{color_data:012b};\n"

    return rom_template
```

ตัวอย่าง : การใช้งาน

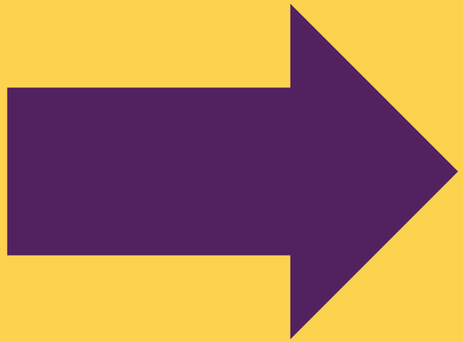
กำหนดที่อยู่ของไฟล์ภาพ ("STAR.PNG") และเรียกใช้ฟังก์ชัน GENERATE_ROM_TEMPLATE เพื่อสร้างเทมเพลต ROM จากภาพและบันทึกเทมเพลต ROM ลงในไฟล์ชื่อ "STAR_ROM.V

ตัวอย่าง CODE

```python
# Example usage
image_path = "star.png"
rom_template = generate_rom_template(image_path)

# Save the ROM template to a file
with open("star_rom.v", "w") as f:
    f.write(rom_template)
```



ตัวอย่าง : PACMAN (16X16 PIXELS)



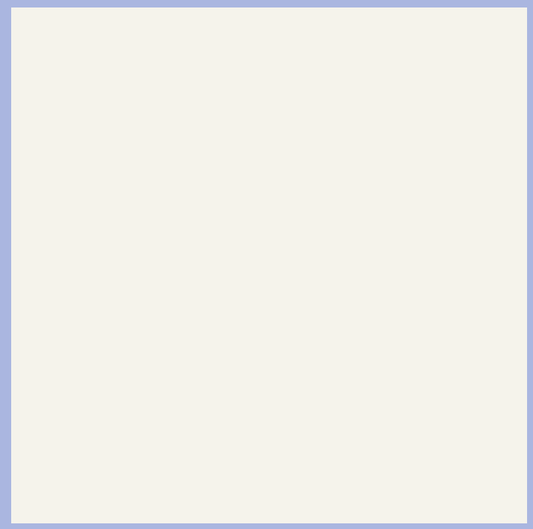
```
8'b00000000: color_data = 12'b111111111110;
8'b00000001: color_data = 12'b111111111110;
8'b00000010: color_data = 12'b111111111110;
8'b00000011: color_data = 12'b111111111110;
8'b00000100: color_data = 12'b110111011101;
8'b00000101: color_data = 12'b110011001011;
8'b00000110: color_data = 12'b001000100001;
8'b00000111: color_data = 12'b001000100000;
8'b00001000: color_data = 12'b001000100000;
8'b00001001: color_data = 12'b001000100000;
8'b00001010: color_data = 12'b001000100001;
8'b00001011: color_data = 12'b110011001011;
8'b00001100: color_data = 12'b110111011101;
8'b00001101: color_data = 12'b111111111110;
8'b00001110: color_data = 12'b111111111110;
8'b00001111: color_data = 12'b111111111110;
8'b00010000: color_data = 12'b111111111110;
```

LOGIC DESIGN : MODULE

vga_port : module ที่ใช้สำหรับการแสดงผลหน้าจอ (Display) ผ่าน vga port โดย output จะเป็นตำแหน่งแกน x และ y

Background_rom : เป็น module ที่เก็บ memory(ROM) สำหรับแสดงผล Background

ตัวอย่างภาพ Background :



(256x256 pixels)

LOGIC DESIGN : MODULE

Walls_rom : เป็น module ที่เก็บ memory(ROM) สำหรับใช้ใน “Wall.v”

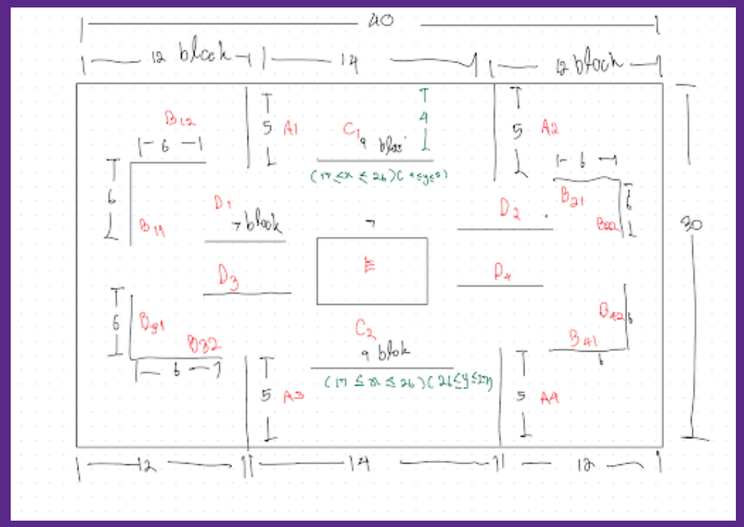
Walls : module สำหรับการแสดงผลกำแพง โดยใช้ Rom จาก “Walls_rom

ตัวอย่าง Code สำหรับ Walls

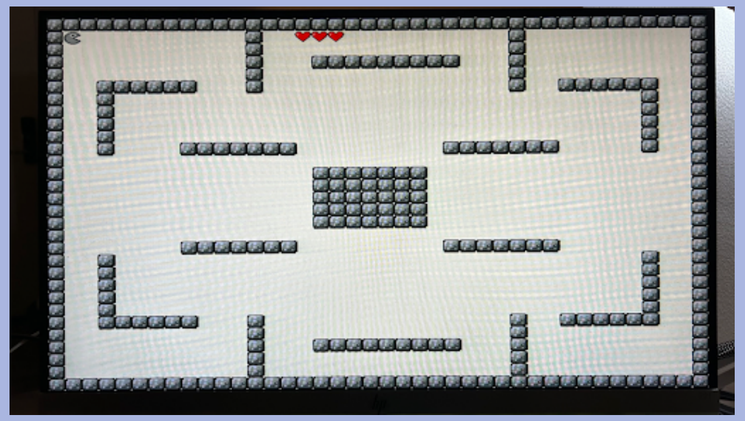
```
always @*
    begin
            //default
            rgb_out = 12'b000000000000;
            walls_on = 0;
            row = 0;
            col = 0;

            if(video_on)
                begin

                //top wall
                if (y >= 0 && y <= 15)
                    begin
                        row = y;
                        col = x;
                        if(walls_color_data != 12'b011011011110)
                            begin
                                rgb_out = walls_color_data;
                                walls_on = 1;
                            end
                    end
```

LOGIC DESIGN : MODULE

รูปภาพแสดงตำแหน่งในการสร้าง Wall บนหน้าจอ

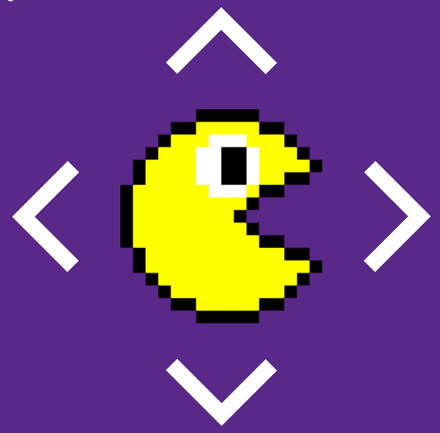


รูปภาพแสดงผลกำแพงทั้งหมดภายในเกม



LOGIC DESIGN : MODULE

Pacman_control : เป็น module ที่ใช้สำหรับแสดงผล pacman และควบคุมการเคลื่อนในแนวแกน x y ผ่านปุ่ม BtnL BtnR BtnU BtnD



ตัวอย่าง : ถ้ากดปุ่ม button left ให้ลบตำแหน่งของ pacman ไป 1 pixel

```
left:
    begin
    if(time_reg > 0)
        time_next = time_reg - 1;
    else if(time_reg == 0)
        begin
        if(s_x_reg >= 17 && !check_wall(s_x_reg - 1, s_y_reg))
            s_x_next = s_x_reg - 1;
        if(!btnL || btnR || btnU || btnD)
            begin
            motion_state_next = no_dir;
            start_next = 0;
            end
        if(start_reg > TIME_MIN)                          // 
            begin
            time_next = start_reg - TIME_STEP;
            start_next = start_reg - TIME_STEP;
            end
        else
            begin
                time_next = start_reg;
                start_next = start_reg;
            end
        end
    end
```

LOGIC DESIGN : MODULE

การคำนวณ row กับ col เพื่อไปใช้แสดงผล pacman

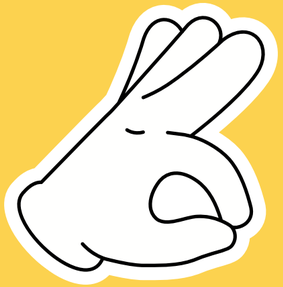
```
287  /*****************************************************************************/
288  /*                          Pacman display area logic                        */
289  /*****************************************************************************/
290  //Pacman display area boundaries
291  wire [3:0] row;
292  wire [3:0] col;
293
294  //current pixel coordinates - current sprite coordinates = pixel coordinates within pacman area
295  assign col = (dir_reg == LEFT && pacman_area) ? T_W - 1 - (x - s_x_reg) :
296               (dir_reg == RIGHT && pacman_area) ? x - s_x_reg :
297               (dir_reg == UP && pacman_area) ? x - s_x_reg :
298               (dir_reg == DOWN && pacman_area) ? x - s_x_reg : 0;
299
300  assign row = (dir_reg == LEFT && pacman_area) ? y - s_y_reg :
301              (dir_reg == RIGHT && pacman_area) ? y - s_y_reg :
302              (dir_reg == UP && pacman_area) ? y - s_y_reg :
303              (dir_reg == DOWN && pacman_area) ? y - s_y_reg : 0;
304
305
306  //Pacman rom
307  //vector for rom color_data output
308  wire [11:0] color_data_pacman, color_data_pacman_ghost;
309  Pacman_rom Pacman_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_pacman));
310  Pacman_Ghost_rom Pacman_ghost_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_pacman_ghost));
311
312
313  //vector to signal when vga_sync is in the pacman area
314  wire pacman_area;
315  assign pacman_area = (x >= s_x_reg && x <= s_x_reg + T_W - 1 && y >= s_y_reg && y <= s_y_reg + T_H - 1) ? 1 : 0;
```

LOGIC DESIGN : MODULE

Ghost_blue, Ghost_red : เป็น module ที่ใช้ในการแสดงผลผี และเคลื่อนไหว ขึ้นอยู่กับตำแหน่งของ pacman

ตัวอย่าง : การคำนวณ row กับ col เพื่อไปใช้แสดงผลผี

```
162  /*****************************************************************************/
163  /*                          Blue ghost display area logic                    */
164  /*****************************************************************************/
165  wire [3:0] row;
166  wire [3:0] col;
167
168  //current pixel coordinates - current sprite coordinates = pixe coordinates within display area
169  assign col = (dir_reg == LEFT && ghost_blue_area) ? T_W - 1 - (x - s_x_reg) :
170               (dir_reg == RIGHT && ghost_blue_area) ? x - s_x_reg :
171               (dir_reg == UP && ghost_blue_area) ? x - s_x_reg :
172               (dir_reg == DOWN && ghost_blue_area) ? x - s_x_reg : 0;
173
174  assign row = (dir_reg == LEFT && ghost_blue_area) ? y - s_y_reg :
175               (dir_reg == RIGHT && ghost_blue_area) ? y - s_y_reg :
176               (dir_reg == UP && ghost_blue_area) ? y - s_y_reg :
177               (dir_reg == DOWN && ghost_blue_area) ? y - s_y_reg : 0;
178
179
180  //Blue ghost rom
181  wire [11:0] color_data_Blue_ghost;
182  Blue_Ghost_rom Blue_Ghost_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_Blue_ghost));
183
184  //check if pixel is within display area
185  wire ghost_blue_area;
186  assign ghost_blue_area = (x >= s_x_reg && x <= s_x_reg + T_W - 1  && y >= s_y_reg && y <= s_y_reg + T_H - 1) ? 1 : 0;
```



LOGIC DESIGN : MODULE

ตัวอย่าง : การคำนวนเส้นทางการเดินต่อไปของผี โดยอ้างอิงจากตำแหน่งของ pacman

- ถ้าผีอยู่ทางขวาของ pacman => เคลื่อนที่ทางซ้าย
- ถ้าผีอยู่ทางซ้ายของ pacman => เคลื่อนที่ทางขวา
- ถ้าผีอยู่ด้านบนของ pacman => เคลื่อนที่ลง
- ถ้าผีอยู่ด้านล่างของ pacman => เคลื่อนที่ขึ้น

รูปภาพแสดงตัวอย่าง
การกำหนดการเคลื่อนของผี

```
92   always @*
93       begin
94       //default
95       dir_next = dir_reg;
96       if(pacman_x < s_x_reg)
97           dir_next = LEFT;
98       else if(pacman_x > s_x_reg)
99           dir_next = RIGHT;
100      else if(pacman_y < s_y_reg)
101          dir_next = UP;
102      else if(pacman_y > s_y_reg)
103          dir_next = DOWN;
104      end
```

LOGIC DESIGN : MODULE

Check_collision : เป็น module ที่ใช้ในการเช็กการชนกันของ pacman และผีทั้งสอง โดยจะคิดจากถ้าตำแหน่งของ pacman เหลื่อมกัน 3 pixel

ตัวอย่าง : การเช็คการชนกันโดยการเทียบ pixel ที่ทับกัน

การเช็คการชนจะขึ้นอยู่กับทิศทาง pacman :

- ถ้าทิศทางของ pacman = ซ้าย : ให้เช็คโดยการนำ pixels แรกสุด - 13 (ขนาด pacman คือ 16 pixels) ว่าเท่ากับ ตำแหน่งของผีไหม
- ถ้าทิศทางของ pacman = ขวา : ให้เช็คโดยการนำ pixels แรกสุด + 13 (ขนาด pacman คือ 16 pixels) ว่าเท่ากับ ตำแหน่งของผีไหม

ตัวอย่าง code สำหรับการตรวจสอบการชนของ pacman และ ผีทั้ง 2 สี

```
22          if(direction == LEFT)
23              begin
24                  //if pacman and blue ghost are within each other's display area
25                  if(pacman_x - 13 <= blue_x && pacman_x >= blue_x - 13 &&
26                     pacman_y - 13 <= blue_y && pacman_y >= blue_y - 13)
27                      collided = 1;
28
29                  //if pacman and red ghost are within each other's display area
30                  if(pacman_x - 13 <= red_x && pacman_x >= red_x - 13 &&
31                     pacman_y - 13 <= red_y && pacman_y >= red_y - 13)
32                      collided = 1;
33              end
34
35          //if direction of pacman is right
36          else if(direction == RIGHT)
37              begin
38                  //if pacman and blue ghost are within each other's display area
39                  if(pacman_x + 13 >= blue_x && pacman_x <= blue_x + 13 &&
40                     pacman_y - 13 <= blue_y && pacman_y >= blue_y - 13)
41                      collided = 1;
42
43                  //if pacman and red ghost are within each other's display area
44                  if(pacman_x + 13 >= red_x && pacman_x <= red_x + 13 &&
45                     pacman_y - 13 <= red_y && pacman_y >= red_y - 13)
46                      collided = 1;
47              end
```

LOGIC DESIGN : MODULE

Hearts_rom : เป็น module ที่จะเก็บ memory สำหรับภาพของหัวใจ

Hearts_display : เป็น module ที่ใช้ในการแสดงผลหัวใจที่รับ memory มาจาก “Hearts_rom” ที่เหลืออยู่ โดยจะรับ input ว่าเหลืออยู่เท่าไหร่มาจาก module “Game_state_machine”

ตัวอย่าง code ในการแสดงผลหัวใจทั้ง 3 อัน

```
//if 1 heart(left)
if(x >= 240 && x < 256 && y >= 16 && y < 32)
    begin
        col = x-240;
        if(num_hearts > 0)                  //if  num_hearts > 0 (1,2,3) left heart is on
            row = y - 16;                   //set full heart

        else                                //else (0) left heart is off
            row = y;                        //set empty heart
        hearts_on = 1;
    end


//if 2 hearts(center)
if(x >= 256 && x < 272 && y>= 16 && y < 32)
    begin
        col = x-256;
        if(num_hearts > 1)                  //if  num_hearts > 1 (2,3) center heart is on
            row = y - 16;                   //set full heart

        else                                //else (0,1) center heart is off
            row = y;                        //set empty heart
        hearts_on = 1;
    end

//if 3 hearts(right)
if(x >= 272 && x < 288 && y >= 16 && y<32)
    begin
        col = x-272;
        if(num_hearts > 2)                  //if  num_hearts > 2 (3) right heart is on
            row = y - 16;                   //set full heart

        else                                //else (0,1,2) right heart is off
            row = y;                        //set empty heart
        hearts_on = 1;
    end
```

GAME_STATE_MACHINE

เป็น module ที่บอกสถานะของเกม และ คอยควบคุมจำนวนของหัวใจ
สถานะของเกมมีดังต่อไปนี้ :

Start : เป็น state ที่เอากำหนดค่าเริ่มต้นเพื่อรอผู้เล่นพร้อม (กดปุ่ม Start เพื่อเริ่มเกม)

ตัวอย่าง code ในการแสดงผลหัวใจทั้ง 3 อัน

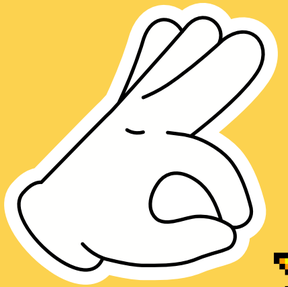
```
162  /*****************************************************************************/
163  /*                          Blue ghost display area logic                    */
164  /*****************************************************************************/
165  wire [3:0] row;
166  wire [3:0] col;
167
168  //current pixel coordinates - current sprite coordinates = pixe coordinates within display area
169  assign col = (dir_reg == LEFT && ghost_blue_area) ? T_W - 1 - (x - s_x_reg) :
170               (dir_reg == RIGHT && ghost_blue_area) ? x - s_x_reg :
171               (dir_reg == UP && ghost_blue_area) ? x - s_x_reg :
172               (dir_reg == DOWN && ghost_blue_area) ? x - s_x_reg : 0;
173
174  assign row = (dir_reg == LEFT && ghost_blue_area) ? y - s_y_reg :
175               (dir_reg == RIGHT && ghost_blue_area) ? y - s_y_reg :
176               (dir_reg == UP && ghost_blue_area) ? y - s_y_reg :
177               (dir_reg == DOWN && ghost_blue_area) ? y - s_y_reg : 0;
178
179
180  //Blue ghost rom
181  wire [11:0] color_data_Blue_ghost;
182  Blue_Ghost_rom Blue_Ghost_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_Blue_ghost));
183
184  //check if pixel is within display area
185  wire ghost_blue_area;
186  assign ghost_blue_area = (x >= s_x_reg && x <= s_x_reg + T_W - 1  && y >= s_y_reg && y <= s_y_reg + T_H - 1) ? 1 : 0;
```

GAME_STATE_MACHINE

Playing : เป็น state ที่บ่งบอกว่าเกมกำลังเล่นอยู่และ ถ้าเกิดมีการชนขึ้นมา ก็จะทำการหักหัวใจของผู้เล่นและให้ย้ายไป state ที่ “Hit” และถ้าหัวใจเหลือ 1 และเกิดการชน ก็จะให้ สถานะของเกม = gameover

ตัวอย่าง code ของสถานะ “start” ของเกม

```
playing:
    begin
        if(collision)
            begin
                if(hearts_reg == 1)     //if has 1 heart left
                    begin
                        hearts_next = hearts_reg - 1;     //decrement heart
                        game_state_next = gameover;
                        game_en_next = 0;
                    end
                else
                    begin
                        game_state_next = hit;
                        if (timer_reg == 0)  // Only set the timer if it's not already set
                            timer_next = 200000000;
                        hearts_next = hearts_reg - 1;
                    end
            end
    end
```

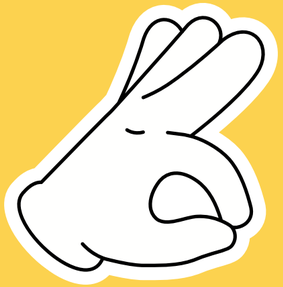


GAME_STATE_MACHINE

Hit : เป็น state ที่เอาทำการหน่วงเวลา(2 seconds) ตอนที่เกิดการชน ตัวอย่าง : หลังจากตรวจพบการชน ก็จะกำหนดให้ time = 200000000 (2 seconds) และถ้า time = 0 เมื่อไหร่ ถึงจะเกิดการชนอีกรอบได้ หมายความว่า ทุกครั้งที่เกิดการชน จะเป็นอมตะประมาณ 2 วินาที

ตัวอย่าง code ของสถานะ “hit” ของเกม

```
hit:
    begin
        if(timer_reg > 0)//pacman can not hit again until timer is 0 (wait 2 seconds)
            begin
                timer_next = timer_reg - 1;
            end
        else
            begin
                game_state_next = playing;
                timer_next = 0;
            end
    end
```



GAME_STATE_MACHINE

Gameover : เป็น state ที่บ่งบอกว่า gameover และถ้ามีการกดปุ่ม start ก็จะทำ Reset เกมและเริ่มใหม่

ตัวอย่าง code ของสถานะ “gameover” ของเกม

```
gameover:
    begin
        if(start_btn_posedge)
            begin
                game_state_next = start;
                game_reset = 1;
                hearts_next = 3;
            end
    end
```

LOGIC DESIGN : MODULE

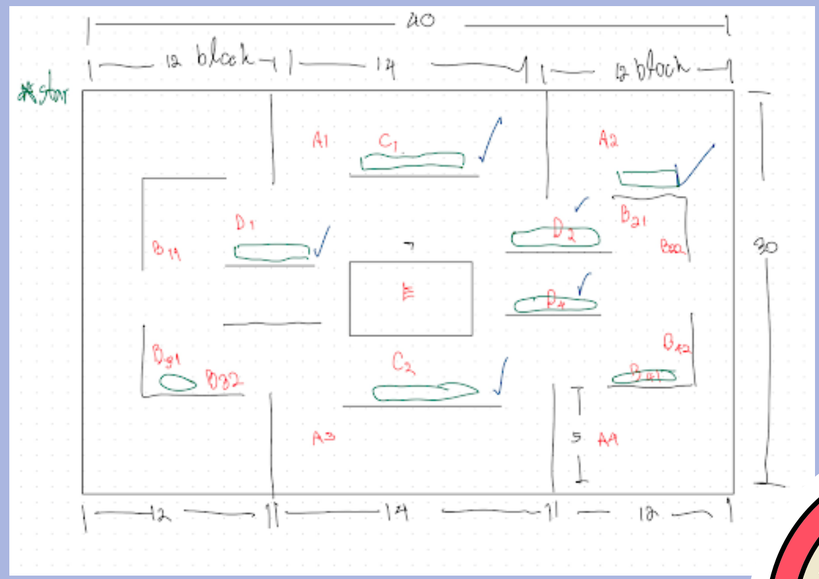
Star_rom : เป็น module ที่เอาไว้เป็น memory ของ star

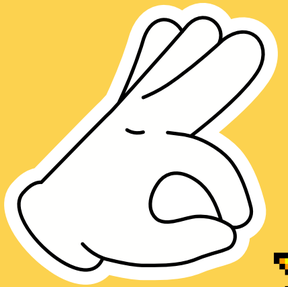
Star : เป็น module นำเอา memory จาก module “Star_rom” มาทำการ spawn star และคอยเช็คว่า pacman เก็บดาวแล้วหรือไม่

ตัวอย่าง : ตำแหน่งของดาวจะถูกกำหนดเอาไว้ ให้ spawn ได้ที่ตำแหน่งดังต่อไปนี้ :

- B32, B21, B41
- C1, C2
- D1, D2, D4

รูปภาพแสดงตำแหน่งที่ star สามารถ spawn ได้





STAR_STATE_MACHINE

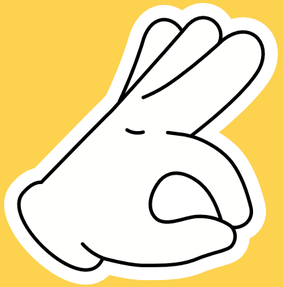
การ spawn star และ การตรวจสอบการชนของ pacman กับดาว

Spawn : สร้าง state สำหรับการ spawn ดาว ซึ่งดาวจะเกิดได้แค่ 1 ดวงต่อครั้ง และถ้า pacman มาเก็บ ก็จะได้ ทำการ respawn ใหม่

ตัวอย่าง : code สำหรับการ spawn ที่ Block “B21”

ตัวอย่าง code สำหรับการ spawn ดาวที่ตำแหน่ง B21

```
else if(area_select_reg == 1) //B21
    begin
        star_y_next = 64;
        star_x_next = B21_reg;
    end
```



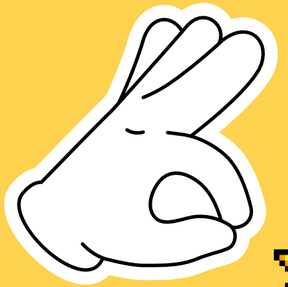
STAR_STATE_MACHINE

ตัวอย่าง code สำหรับการตรวจจับการชนของ pacman กับ ดาว เมื่อ pacman มีทิศทางซ้าย

```
//if pacman collide with facing LEFT
if(direction == LEFT && pacman_x - 13 <= star_x_reg && pacman_x >= star_x_reg - 13 && pacman_y - 13 <= star_y_reg && pacman_y >= star_y_reg - 13)
    begin
        star_state_next = respawn;
        new_score_next = 1;
    end
```

Waiting : เป็น state ที่เอาไว้ตรวจจับการชนของ pacman กับ ดาว โดยถ้า pixels ของ pacman เหลื่อม กับ pixel ของดาว 3 pixels ก็จะถือว่าเก็บดาวแล้ว

ตัวอย่าง : code สำหรับการเช็คการชน เมื่อ ทิศทางของ pacman = ซ้าย



GAME_STATE_MACHINE

ตัวอย่าง : การกำหนดค่าตำแหน่ง (row and col) สำหรับแสดงผลดาว

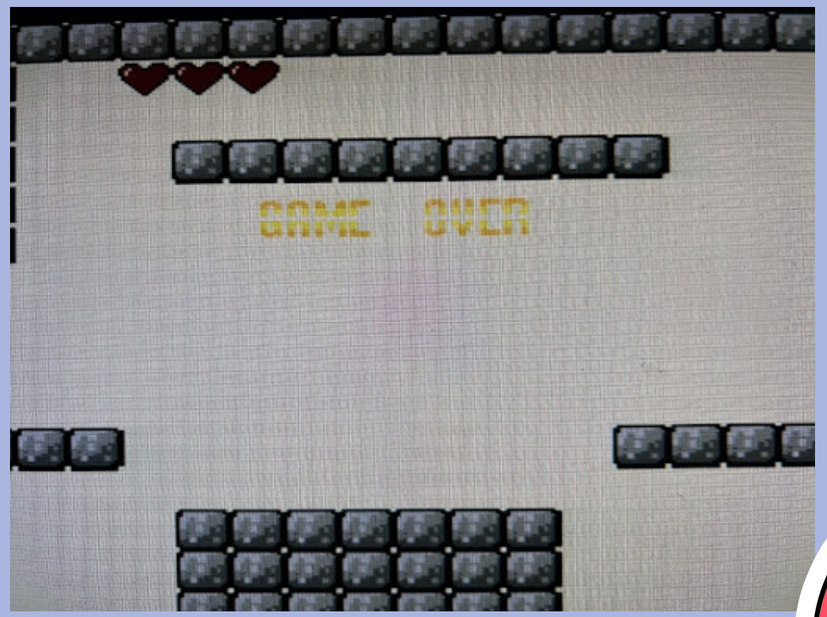
```
//sprite coordinate register to display stars
wire [3:0] col;
wire [3:0] row;

//current pixel coordinate - star coordinate = pixel coordinate of star
assign col = x - star_x_reg;
assign row = y - star_y_reg;
wire [11:0] color_data_star;

Star_rom Star_rom_unit (.clk(clk), .col(col), .row(row), .color_data(color_data_star));

//assign stars_on for output
assign stars_on = (x >= star_x_reg && x < star_x_reg + 16 && y >= star_y_reg && y < star_y_reg + 16 && color_data != 12'b111111111110)? 1 : 0;

//assign rgb_out for output
assign color_data = color_data_star;
```

LOGIC DESIGN : MODULE

gameover_rom : ใช้เก็บ memory สำหรับข้อความ “GAMEOVER” เพื่อไปใช้ใน module “gameover_display”

gameover_display : นำ Rom จาก module “gameover_rom” มาใช้สำหรับแสดงผลข้อความ “GAMEOVER” บนหน้าจอเมื่อเกม over

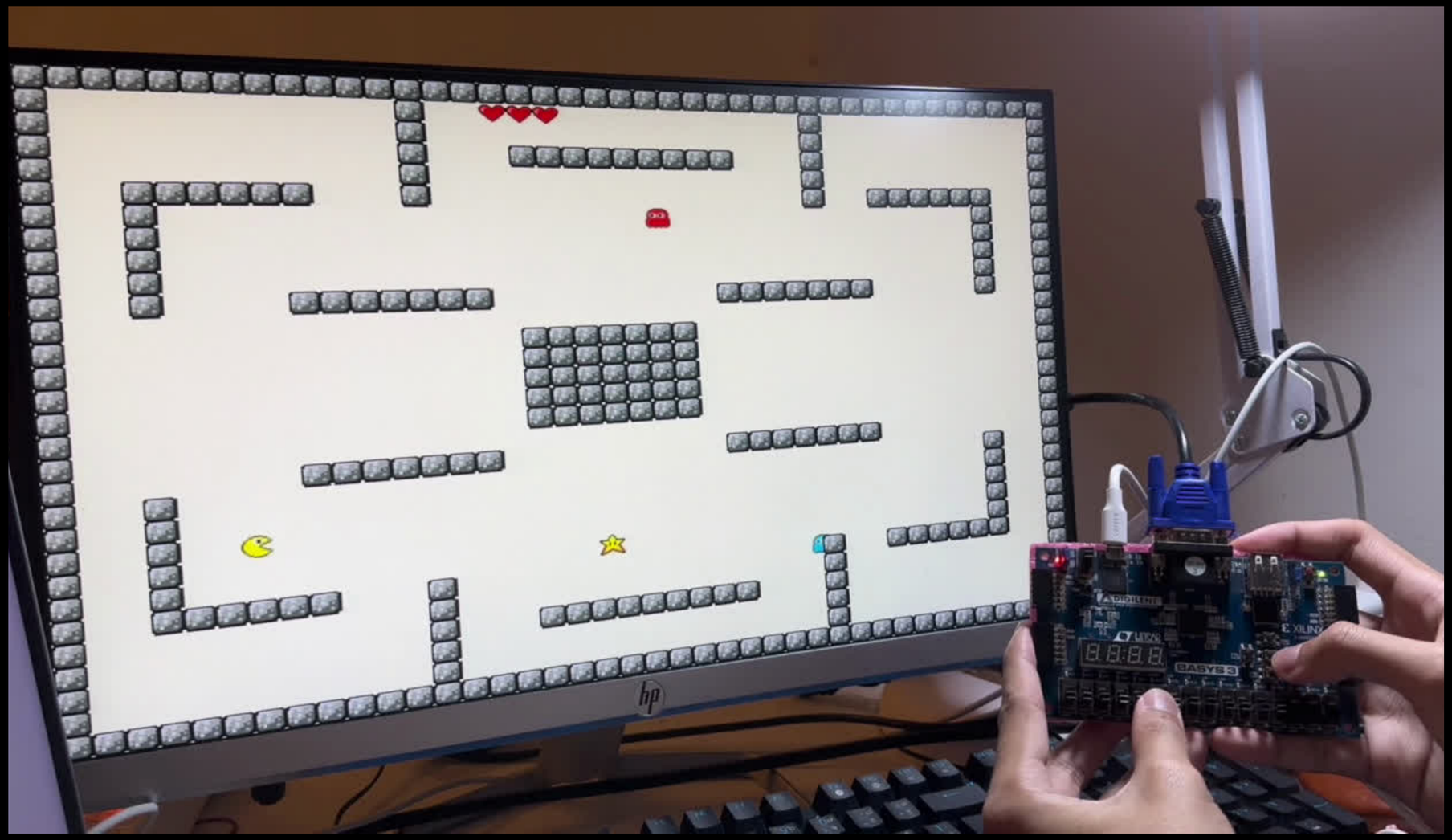
ตัวอย่างการแสดงผลข้อความ “GAME OVER”



DEMO

READY

DEMO



READY

THANK YOU