```
`timescale 1ns / 1ps
/***************************************************************************
 * File Name: pixel_generator.v
 * Project: VGA Object Mapped
 * Designer: Marc Cabote
 * Email: marcdominic011@gmail.com
 * Rev. Date: 12 November, 2017
 *
 * Purpose: This module will generate the objects specified. These objects
 *          are a Wall, a Ball , and a Bar(Paddle). Each object will have
 *          a specified region.The Wall shall occupy the region from
 *          horizontal scan count 32 through 35.The Paddle shall occupy
 *          the region from horizontal scan count 600 through 603 and
 *          vertical scan count 204 to 276, at the beginning. The ball
 *          will go towards the rightmost part of the screen in the beginning
 *
 * Notes:   -  This module has no reset, the reset comes from vga sync
 *          -  vide_on enables objects to be displayed
 *          -  btn1 and btn0 makes the paddle go up and down
 *          -  The animation of the objects were done using Pong Chu's
 *             Verilog by examples as a reference.
 ***************************************************************************/
module pixel_generator(input  clk, rst, video_on,
                       input [1:0] btn,
                       input [9:0] pixel_x, pixel_y,
                       output reg[11:0] rgb);
   //object output signals
   wire wall, bar, ball;
   wire [11:0] wall_rgb, bar_rgb, ball_rgb;

   /*********************************
   * LOCAL PARAMETERS
   *********************************/

   /*--------------------------------
   * PADDLE
   --------------------------------*/
   //top-bottom boundaries
   wire [9:0] bar_y_T, bar_y_B;
   localparam bar_y_SIZE = 72;
   //accomodate changes in y-axis
   reg [9:0] bar_y_REG, bar_y_NEXT;
   //button sensitivity for paddle movement
   localparam bar_V = 4;//bar velocity

   /*--------------------------------
   * BALL
   --------------------------------*/
   localparam ball_SIZE = 8;
   //left-right boundaries
   wire [9:0] ball_x_L, ball_x_R;
   //top-bottom boundaries
   wire [9:0] ball_y_T, ball_y_B;
   //accomodate changes in y and x axis
   reg  [9:0] ball_x_REG, ball_y_REG;
   wire [9:0] ball_x_NEXT, ball_y_NEXT;
   //registers for velocity
```

```
    reg  [9:0] x_delta_REG, x_delta_NEXT,
               y_delta_REG, y_delta_NEXT;
    //ball velocity
    localparam ball_pos_V = 2;
    localparam ball_neg_V = -2;

    /*--------------------------------
    * HANDLE REGISTERS
    --------------------------------*/
    always @ (posedge clk, posedge rst)
       if (rst) begin
          bar_y_REG <= 0;
          ball_x_REG <= 0;
          ball_y_REG <= 0;
          x_delta_REG <= 10'h004;
          y_delta_REG <= 10'h004;
       end
       else begin
          bar_y_REG <= bar_y_NEXT;
          ball_x_REG <= ball_x_NEXT;
          ball_y_REG <= ball_y_NEXT;
          x_delta_REG <= x_delta_NEXT;
          y_delta_REG <= y_delta_NEXT;
       end

    /*********************************
    *generate 60 Hz clock tick
    *********************************/
    //denotes that after one screen refresh generate a tick
    wire refr_tick;
    assign refr_tick = ((pixel_y == 481)&&(pixel_x == 0));

    /*********************************
    * generate WALL
    *********************************/
    assign wall = (pixel_x >= 32) && ( pixel_x <= 35);
    assign wall_rgb = 12'hF00;//wall blue

    /*********************************
    * generate BAR
    *********************************/
    //boundaries
    assign bar_y_T = bar_y_REG;
    //subtract one to take movement into account
    assign bar_y_B = bar_y_T + bar_y_SIZE - 1;

    //pixel location of paddle
    assign bar = (pixel_x >= 600) && (pixel_x <= 603)
               &&(bar_y_T <= pixel_y) && (pixel_y <= bar_y_B);
    assign bar_rgb = 12'h0F0;//bar green

    //new bar position
    always @ (*) begin
       bar_y_NEXT = bar_y_REG; //no movement
       if (refr_tick)
          if (btn[1] & (bar_y_B < (max_y - 1 - bar_V)))
             bar_y_NEXT = bar_y_REG + bar_V; else //move down
```

```
            if (btn[0] & (bar_y_T > bar_V))
               bar_y_NEXT = bar_y_REG - bar_V;
      end

      /*********************************
      * generate BALL
      *********************************/
      //boundaries
      assign ball_x_L = ball_x_REG;
      assign ball_y_T = ball_y_REG;
      assign ball_x_R = ball_x_L + ball_SIZE - 1;
      assign ball_y_B = ball_y_T + ball_SIZE - 1;

      //pixel location of ball
      assign ball = (ball_x_L <= pixel_x) && (pixel_x <= ball_x_R)
                  &&(ball_y_T <= pixel_y) && (pixel_y <= ball_y_B);
      assign ball_rgb = 12'h00F;//ball red

      //new ball position
      assign ball_x_NEXT = (refr_tick)? ball_x_REG + x_delta_REG :
                                        ball_x_REG;
      assign ball_y_NEXT = (refr_tick)? ball_y_REG + y_delta_REG :
                                        ball_y_REG;

      //new ball velocity
      always @ (*) begin
         x_delta_NEXT = x_delta_REG;
         y_delta_NEXT = y_delta_REG;
         if (ball_y_T < 1) //ball reaches top
            y_delta_NEXT = ball_pos_V;else //positive velocity
         if (ball_y_B > (max_y - 1))//ball reaches bottom
            y_delta_NEXT = ball_neg_V;else //negative velocity
         if (ball_x_L <= wall_x_R)//reaches wall
            x_delta_NEXT = ball_pos_V;else //positive velocity
         if ((bar_x_L <= ball_x_R) && (ball_x_R <= bar_x_R) &&
             (bar_y_T <= ball_y_B) && (ball_y_T <= bar_y_B))
             //reach x of right bar and hit, ball bounce back
            x_delta_NEXT = ball_neg_V; //negative velocity
      end

      /*********************************
      * generate display
      *********************************/
      always @ (*) begin
         if (video_on)
            if (wall)
               rgb = wall_rgb; else
            if (bar)
               rgb = bar_rgb; else
            if (ball)
               rgb = ball_rgb;
            else
               rgb = 12'h000;//blank background
         else
            rgb = 12'h000;//blank
      end
   endmodule
```