Project - Bee Invaders

Tutorial 2: Display The Bee At The Bottom Of The Screen

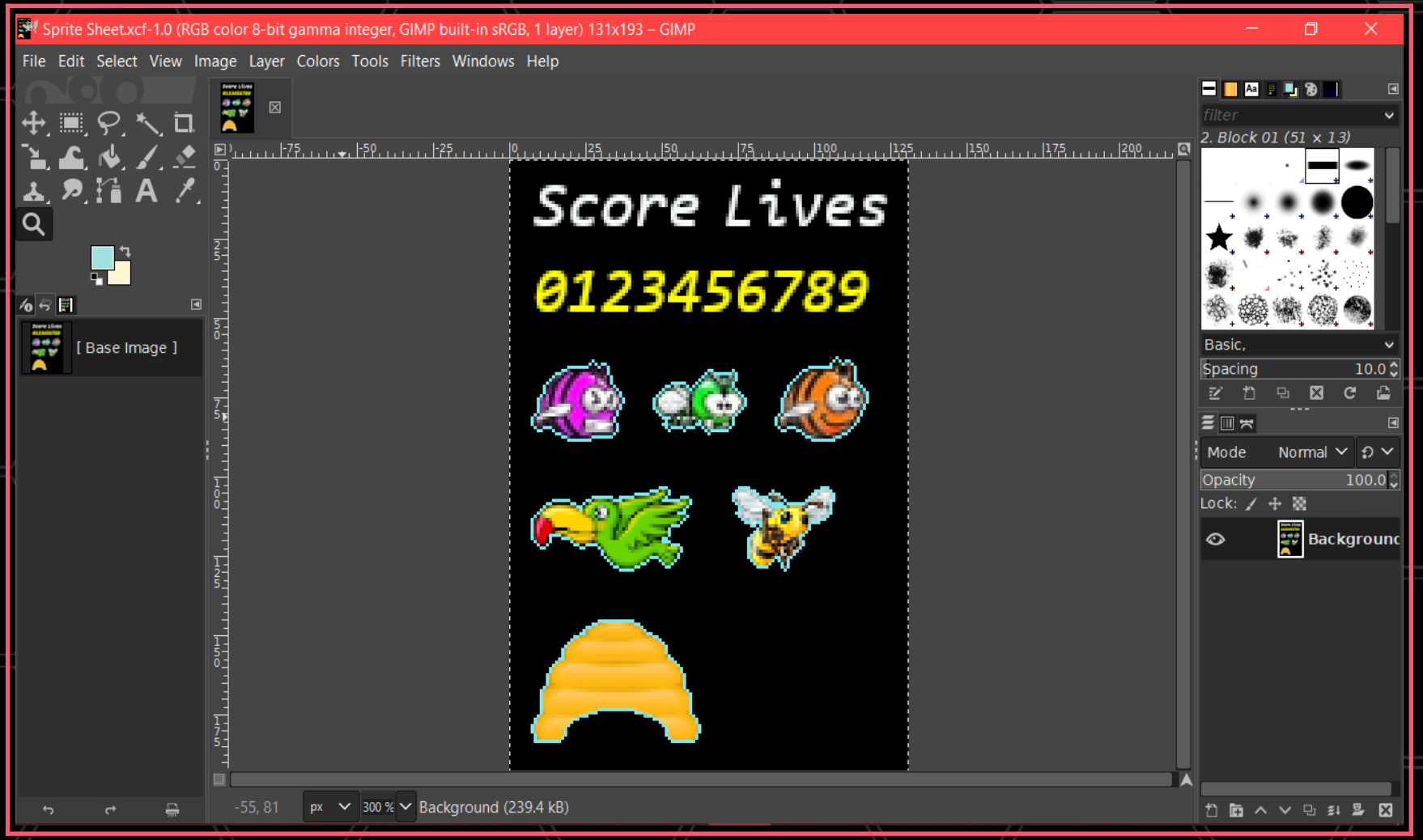
This Tutorial Is Specifically For The Digilent Basys 3 Board

Proposed Game

Instructions

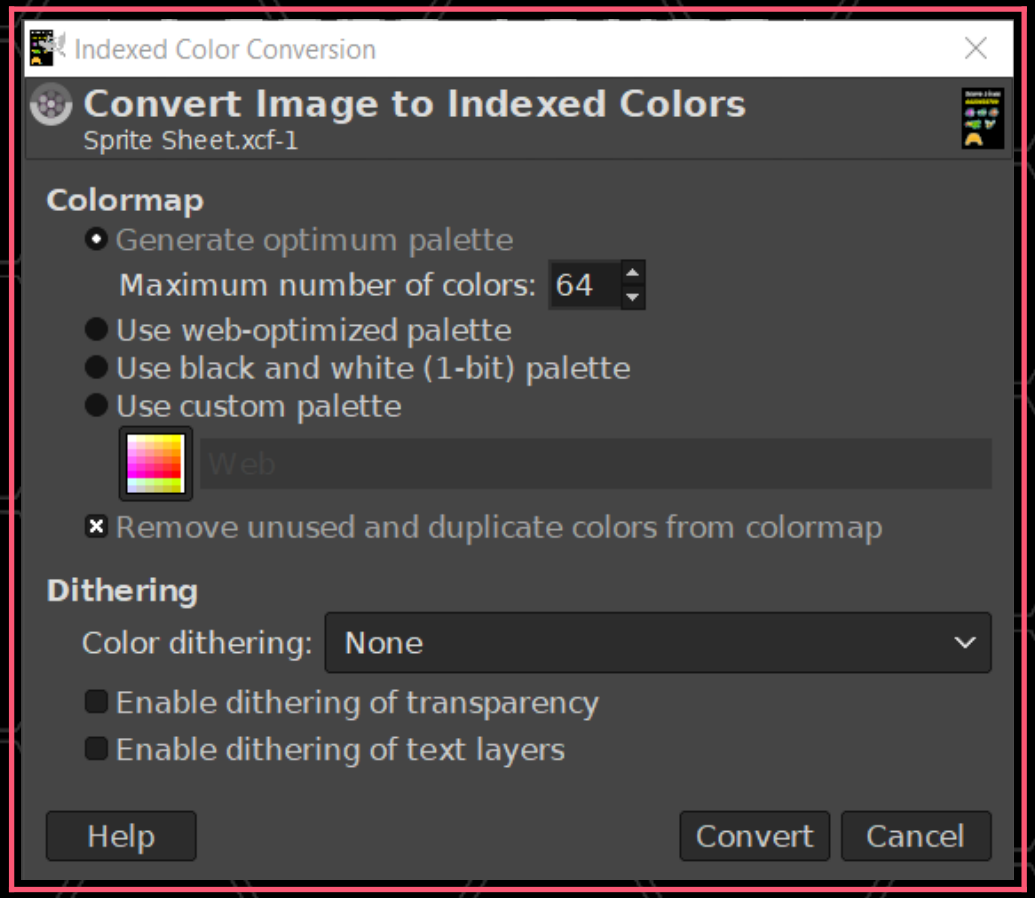
01 Download / install from the Gimp website their Free image manipulation software

Download my Gimp file "Sprite Sheet.xcf" and open it with Gimp

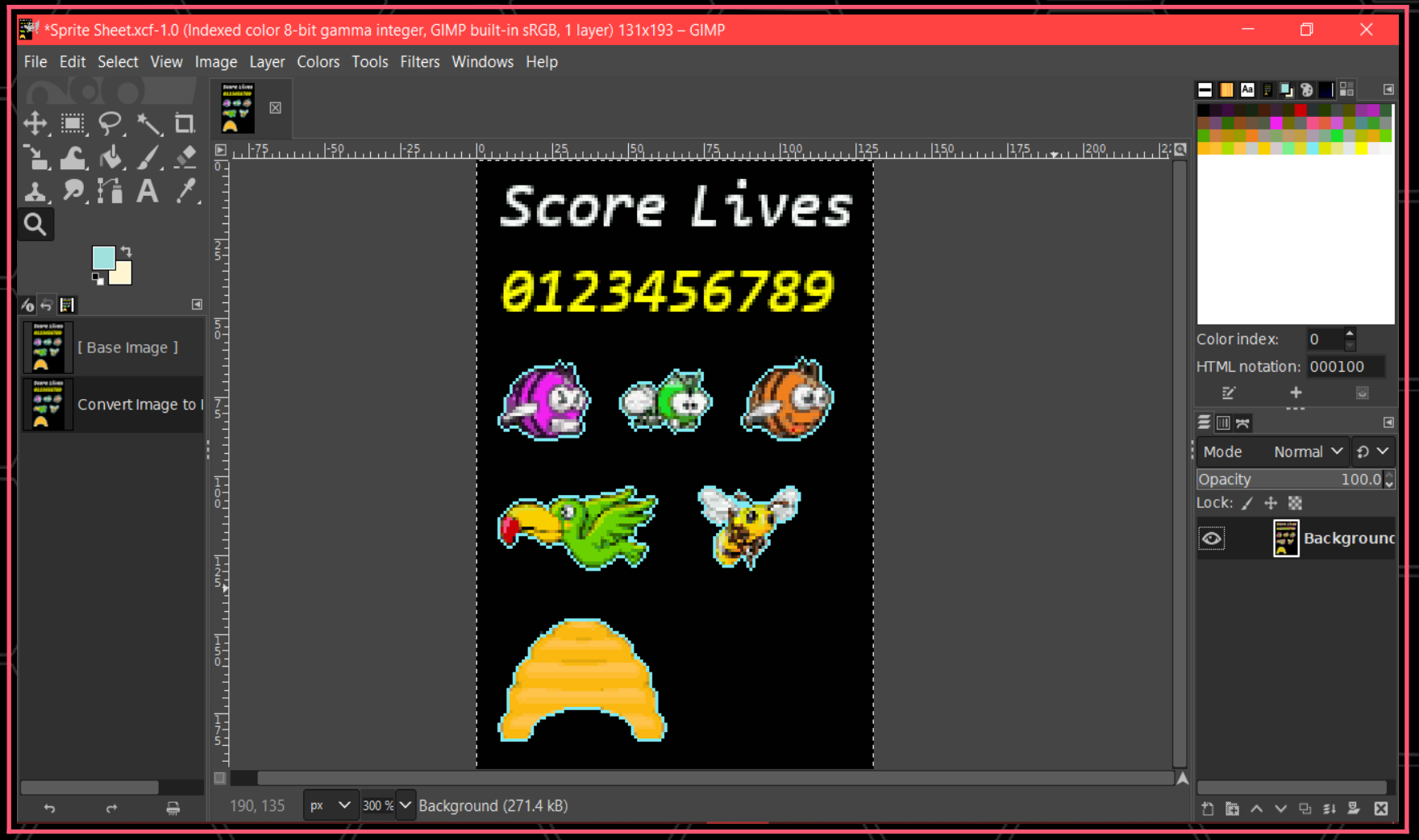


02

The sprite sheet needs to be converted to 64 colours by selecting: Image → Mode → Indexed, set the maximum number of colours to 64 and select Convert (as shown below)

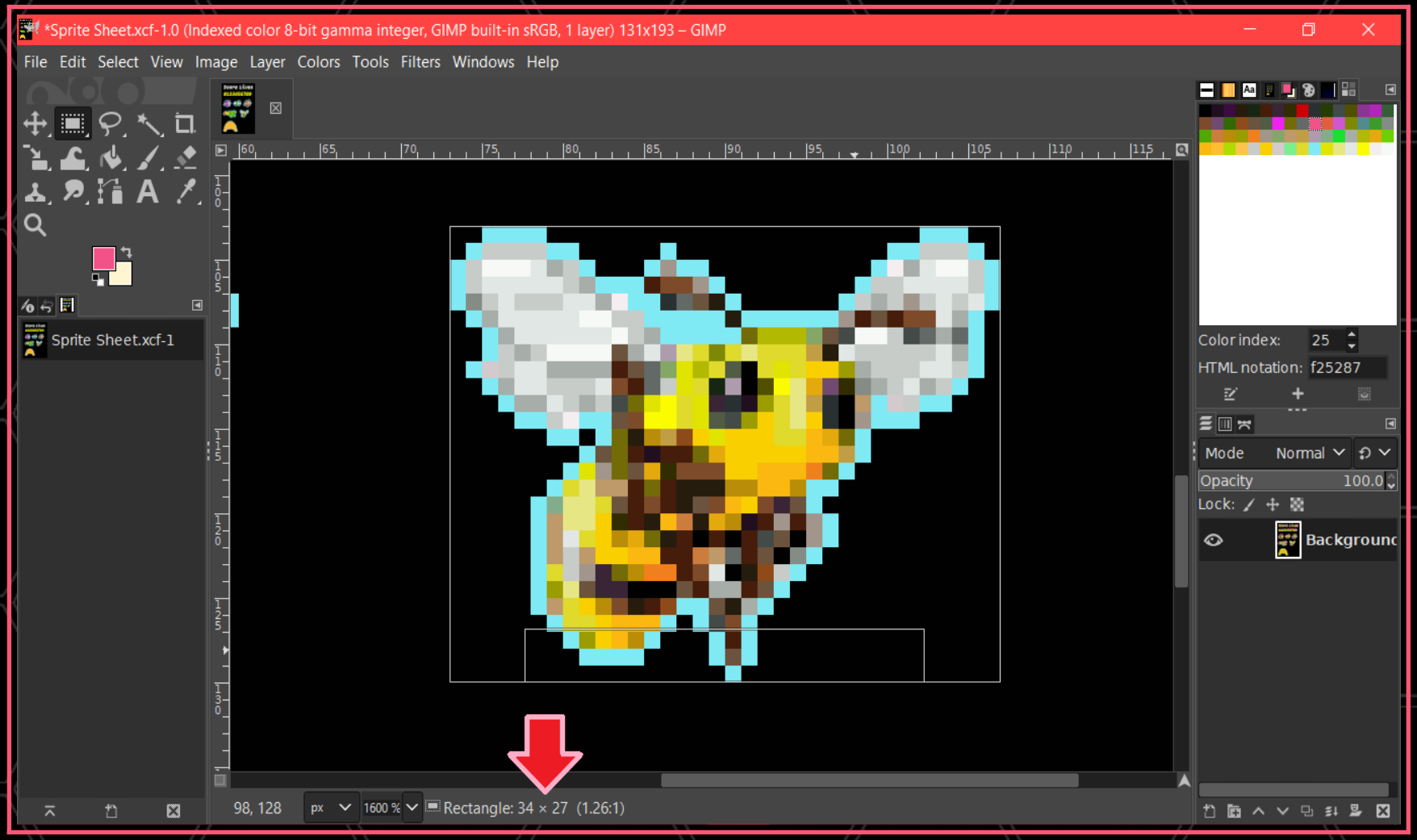


03 Select: Windows → Dockable Dialogs → Colormap, to show the colour palette created

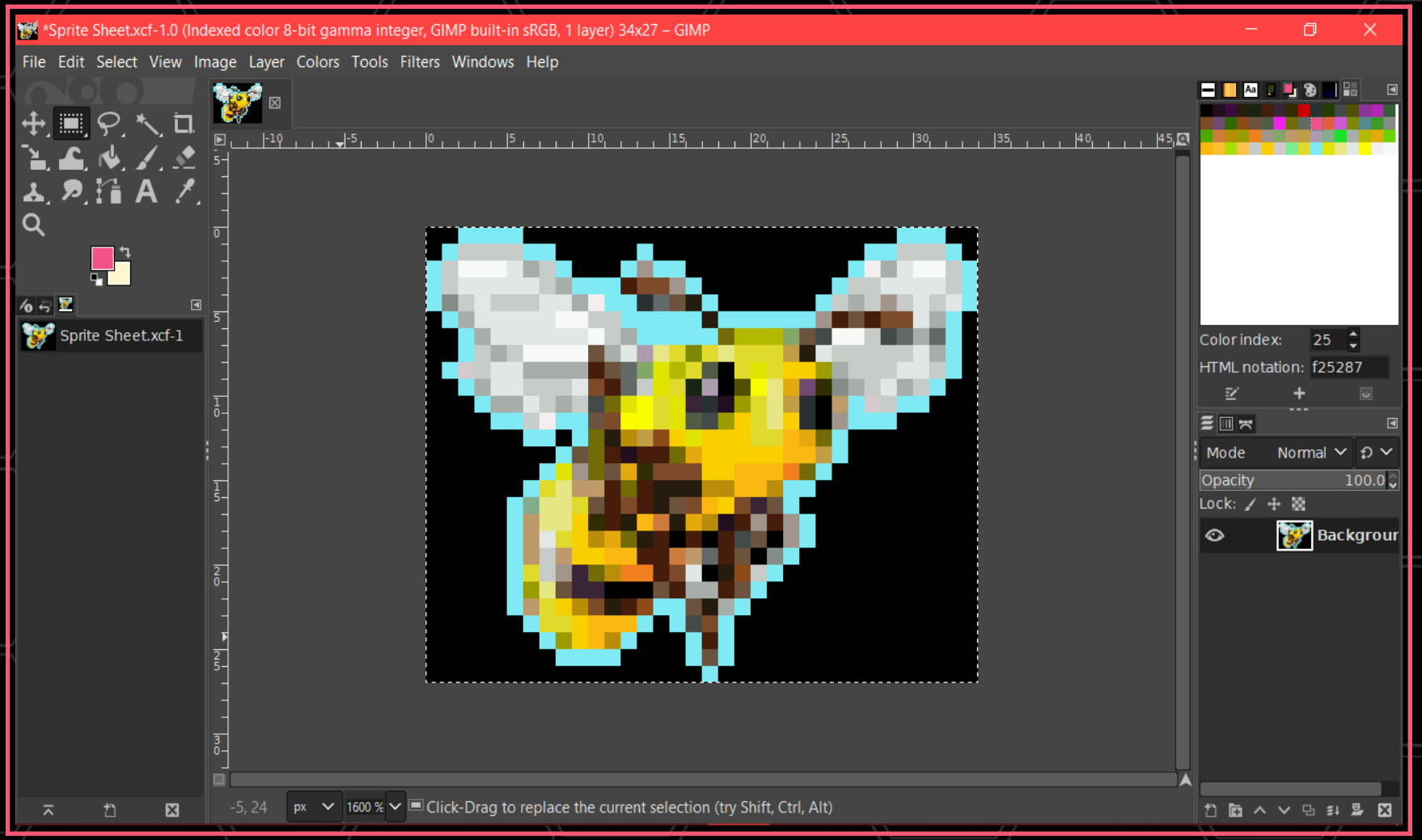


04

Zoom in on the Bee character and using the "rectangle select tool" select around the bee (this should be a rectangle 34 x 27 pixels)

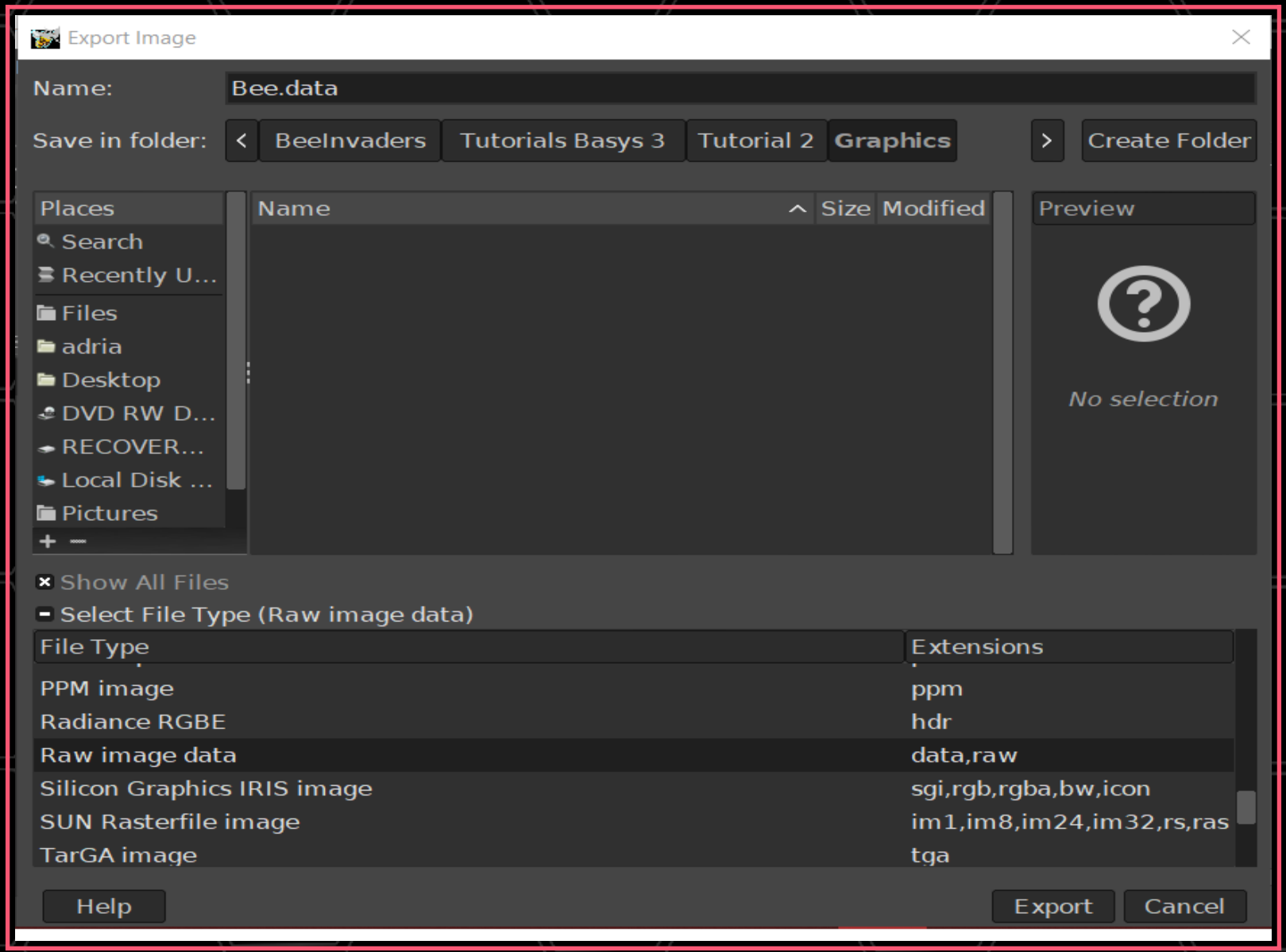


05 Now select: Image → Crop to Selection

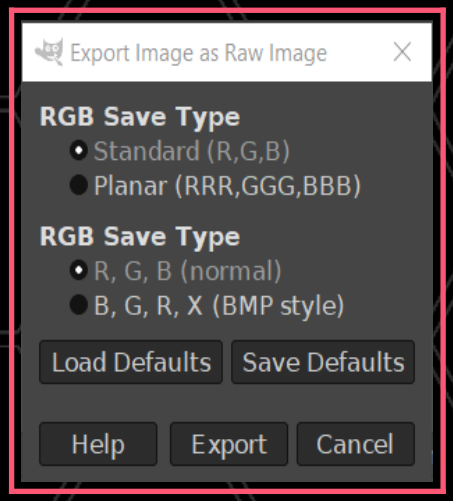


06

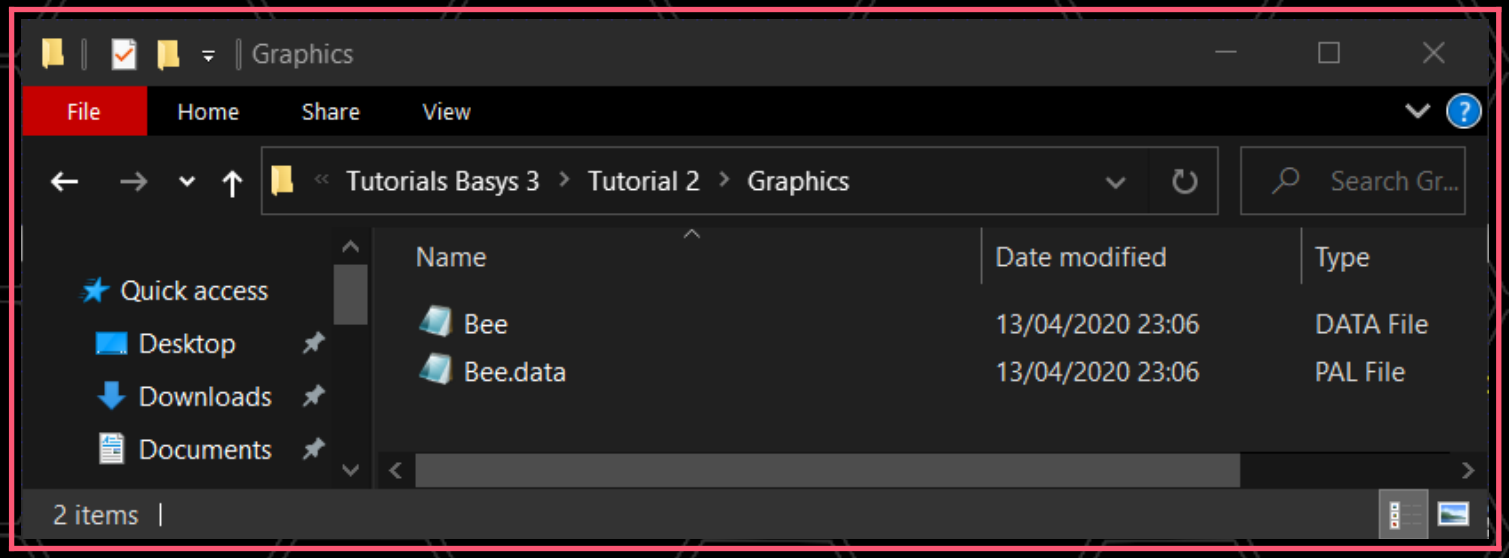
The image needs to be saved as a Raw Data File, do this using File → Export As → Raw image data.
Call the file "Bee.data", point "Save in folder" to somewhere on your computer and click "Export"



07 When the below screen appears click "Export"

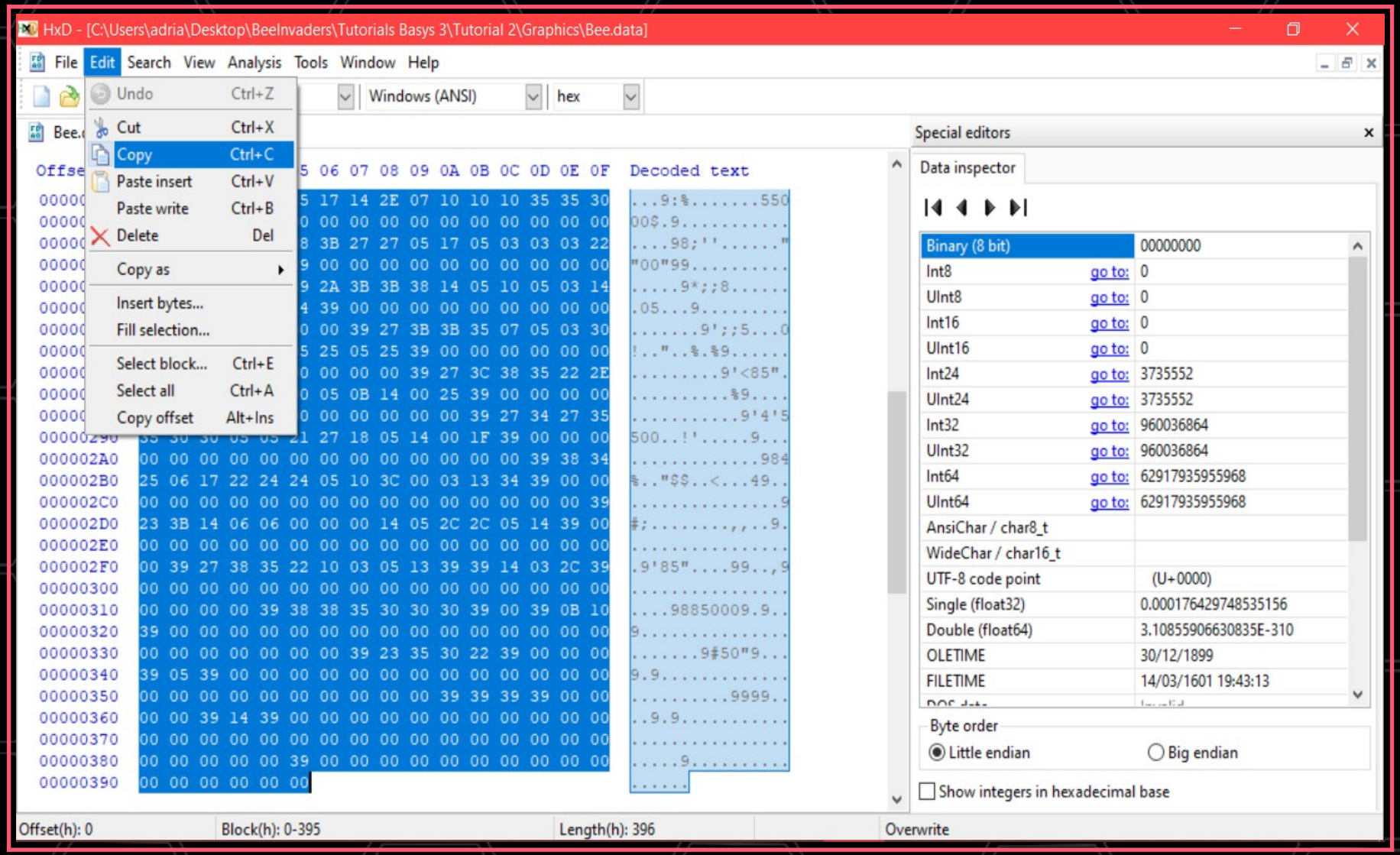


If you navigate to the folder where you exported the Bee character, you should now have 2 files in the folder (Bee.data and Bee.data.pal)

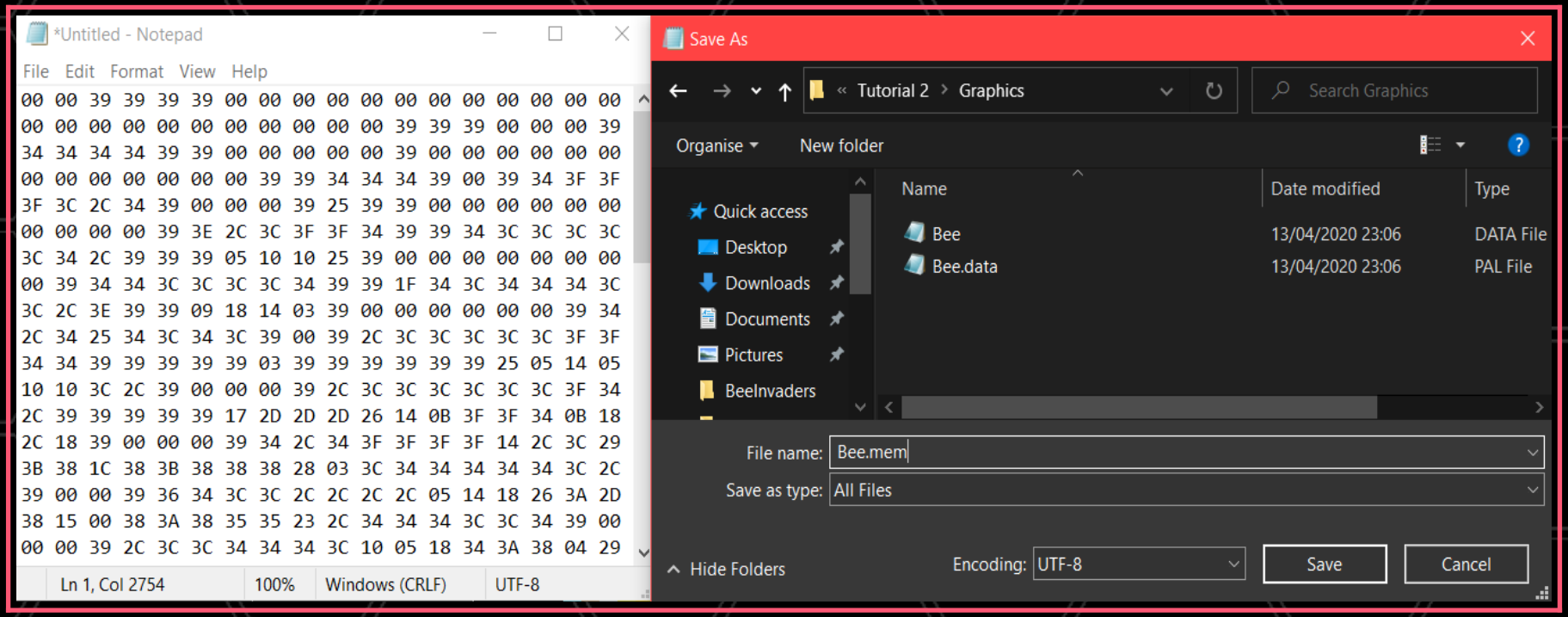


08

Next download, install and run the free program (or similar) called HxD Hex Editor and load the file "Bee.data", select all the data and copy it

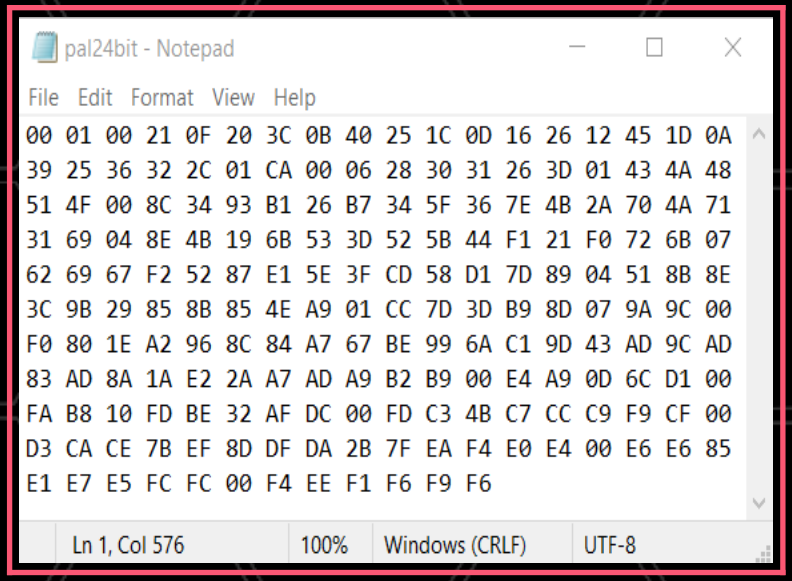


09 Paste the data into a Notepad file and save it as "Bee.mem" in the same folder you exported the Bee character to



10 Do the same with the "Bee.data.pal" file but save the Notepad file as "pal24bit.mem"

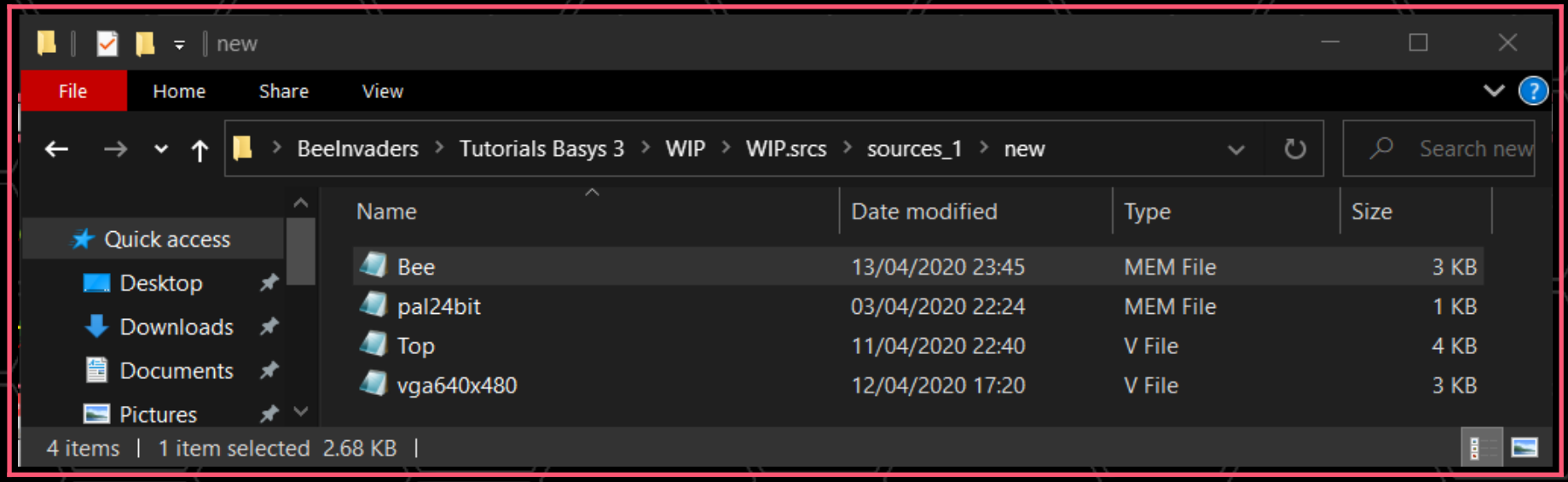
If you compare this file to the "pal24bit.mem" file in Tutorial 1, you will notice they are the same



The file contains data for 8 bit Red, 8 bit Green and 8 bit Blue colours (3 sets of 8 bit data by 64 colours = 24bit colour values)

11 Now copy and paste the "Bee.mem" file into the project folder;

Path: BeeInvaders\Tutorials Basys 3\WIP\WIP.srcs\sources_1\new



12

Open the project "WIP" in Vivado

Right click on "Design Sources" and left click on "Add Sources"

Select "Add or create design sources" and click "Next"

Select "Add Files" and navigate to the "Bee.mem" which you copied to "BeeInvaders\Tutorials Basys 3\WIP\WIP.srcs\sources_1\new" folder. Select the file and click "OK" and then "Finish"

Right click on "Design Sources" again and left click on "Add Sources"

Select "Add or create design sources" and click "Next"

Select "+" and click on "Create File" or click on the "Create File" button

Make sure "Verilog" is the "File Type:", enter "BeeRom" in the box entitled "File name:", ensure "Local to Project" is the "File location:" and click "OK"

Select "Finish" at the next screen, "OK" at the following screen and "Yes" at the last screen

13 Double click on "BeeRom (BeeRom.v)" in the Sources (design) panel to open the module

Remove all the code in the "BeeRom.v" box and copy & paste the code from either the "BeeRom.v" file you downloaded or from below, into the "BeeRom.v" code box

```
//----------------------------------------------------
// BeeRom Module - Single Port ROM : Digilent Basys 3
// BeeInvaders Tutorial 2 : Onboard clock 100MHz
// VGA Resolution 640x480 @ 60Hz : Pixel Clock 25MHz
//----------------------------------------------------
`timescale 1ns / 1ps

// Setup BeeRom Module
module BeeRom(
    input wire [9:0] i_addr, // (9:0) or 2^10 or 1024, need 34 x 27 = 918
    input wire i_clk2,
    output reg [7:0] o_data // (7:0) 8 bit pixel value from Bee.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] memory_array [0:917]; // 8 bit values for 918 pixels of Bee (34 x 27)

    initial begin
            $readmemh("Bee.mem", memory_array);
    end

    always @ (posedge i_clk2)
            o_data <= memory_array[i_addr];
endmodule
```

14 Right click on "Design Sources" and left click on "Add Sources"

Select "Add or create design sources" and click "Next"

Select "+" and click on "Create File" or click on the "Create File" button

Make sure "Verilog" is the "File Type:", enter "BeeSprite" in the box entitled "File name:", ensure "Local to Project" is the "File location:" and click "OK"

Select "Finish" at the next screen, "OK" at the following screen and "Yes" at the last screen

Double click on "BeeSprite (BeeSprite.v)" in the Sources (design) panel to open the module

Remove all the code in the "BeeSprite.v" box and copy & paste the code from either the "BeeSprite.v" file you downloaded or from below, into the "BeeSprite.v" code box

```
//------------------------------------------------
// BeeSprite Module : Digilent Basys 3
// BeeInvaders Tutorial 2 : Onboard clock 100MHz
// VGA Resolution 640x480 @ 60Hz : Pixel Clock 25MHz
//------------------------------------------------
`timescale 1ns / 1ps

// Setup BeeSprite Module
module BeeSprite(
    input wire i_clk,
    input wire i_rst,
    input wire [9:0] xx,
    input wire [9:0] yy,
```

```
    input wire aactive,
    output reg [1:0] BSpriteOn, // 1=on, 0=off
    output wire [7:0] dataout
    );

    // instantiate BeeRom code
    reg [9:0] address; // 2^10 or 1024, need 34 x 27 = 918
    BeeRom BeeVRom (.i_addr(address),.i_clk2(i_clk),.o_data(dataout));

    // setup character positions and sizes
    reg [9:0] BeeX = 297; // Bee X start position
    reg [8:0] BeeY = 433; // Bee Y start position
    localparam BeeWidth = 34; // Bee width in pixels
    localparam BeeHeight = 27; // Bee height in pixels

    // check if xx,yy are within the confines of the Bee character
    always @ (posedge i_clk)
    begin
        if (aactive)
            begin
                if (xx==BeeX-1 && yy==BeeY)
                    begin
                        address <= 0;
                        BSpriteOn <=1;
                    end
                if ((xx>BeeX-1) && (xx<BeeX+BeeWidth) && (yy>BeeY-1) && (yy<BeeY+BeeHeight))
                    begin
                        address <= (xx-BeeX) + ((yy-BeeY)*BeeWidth);
                        BSpriteOn <=1;
                    end
                else
                    BSpriteOn <=0;
            end
    end
endmodule
```

15 The "vga640x480" module has changed slightly, the pixel clock has moved from the "Top" module into this module

Double click on "vga640x480 (vga640x480.v)" in the Sources (design) panel to open the module.

Remove all the code in "vga640x480.v" box and copy & paste the code from either the "vga640x480.v" file you downloaded or from below, into the "vga640x480.v" code box

```
//----------------------------------------------------
// vga640x480 Module : Digilent Basys 3
// BeeInvaders Tutorial 2 : Onboard clock 100MHz
// VGA Resolution 640x480 @ 60Hz : Pixel Clock 25MHz
//----------------------------------------------------
`timescale 1ns / 1ps

// Setup vga640x480 Module
module vga640x480(
    input wire i_clk, // 100MHz onboard clock
    input wire i_rst, // reset
    output wire o_hsync, // horizontal sync
    output wire o_vsync, // vertical sync
    output wire o_active, // high during active pixel drawing
    output wire [9:0] o_x, // current pixel x position
    output wire [9:0] o_y // current pixel y position
    );

    // setup VGA timings
    //-------------------------------------
    // VGA 640x480 Horizontal Timing (line)
    localparam HSYNCSTART = 16; // horizontal sync start
    localparam HSYNCEND = 16 + 96; // horizontal sync end
    localparam HACTIVESTART = 16 + 96 + 48; // horizontal active start
    localparam HACTIVEEND = 16 + 96 + 48 + 640; // horizontal active end
    reg [9:0] H_SCAN; // line position

    // VGA 640x480 Vertical timing (frame)
```

```
    localparam VSYNCSTART = 10; // vertical sync start
    localparam VSYNCEND = 10 + 2; // vertical sync end
    localparam VACTIVESTART = 10 + 2 + 33; // vertical active start
    localparam VACTIVEEND = 10 + 2 + 33 + 480; // vertical active end
    reg [9:0] V_SCAN; // screen position

    // set sync signals to low (active) or high (inactive)
    assign o_hsync = ~((H_SCAN >= HSYNCSTART) & (H_SCAN < HSYNCEND));
    assign o_vsync = ~((V_SCAN >= VSYNCSTART) & (V_SCAN < VSYNCEND));

    // set x and y values
    assign o_x = (H_SCAN < HACTIVESTART) ? 0 : (H_SCAN - HACTIVESTART);
    assign o_y = (V_SCAN < VACTIVESTART) ? 0 : (V_SCAN - VACTIVESTART);

    // set active high during active area
    assign o_active = ~((H_SCAN < HACTIVESTART) | (V_SCAN < VACTIVESTART));

    // generate 25MHz pixel clock using a "Fractional Clock Divider"
    reg [15:0] counter1;
    reg pix_clk;
    always @(posedge i_clk)
        {pix_clk, counter1} <= counter1 + 16'h4000; // divide 100MHz by 4 = 25MHz : (2^16)/4 = 16384 decimal or
4000 hex

    // check for reset / create frame loop
    always @ (posedge i_clk)
    begin
        // check for reset button pressed
        if (i_rst) // jump to start of a frame and reset registers
        begin
            H_SCAN <= 0;
            V_SCAN <= 0;
        end

        // loop through a full screen
        if (pix_clk)
        begin
            if (H_SCAN == HACTIVEEND)  // if at the end of a line update registers
            begin
                H_SCAN <= 0;
                V_SCAN <= V_SCAN + 1;
            end
            else
                H_SCAN <= H_SCAN + 1; // else increment horizontal counter
```

```
            if (V_SCAN == VACTIVEEND)  // if at the end of a screen reset vertical counter
                V_SCAN <= 0;
        end
    end
endmodule
```

The "Top" module has also changed to accommodate the BeeSprite module

Double click on "Top (Top.v)" in the Sources (design) panel to open the module.

Remove all the code in "Top.v" box and copy & paste the code from either the "Top.v" file you downloaded or from below, into the "Top.v" code box

```
//-------------------------------------------------
// Top Module : Digilent Basys 3
// BeeInvaders Tutorial 2 : Onboard clock 100MHz
// VGA Resolution 640x480 @ 60Hz : Pixel Clock 25MHz
//-------------------------------------------------
`timescale 1ns / 1ps

module Top(
    input wire CLK, // Onboard clock 100MHz : INPUT Pin W5
    input wire RESET, // Reset button : INPUT Pin U18
    output wire HSYNC, // VGA horizontal sync : OUTPUT Pin P19
    output wire VSYNC, // VGA vertical sync : OUTPUT Pin R19
    output reg [3:0] RED, // 4-bit VGA Red : OUTPUT Pin G19, Pin H19, Pin J19, Pin N19
    output reg [3:0] GREEN, // 4-bit VGA Green : OUTPUT Pin J17, Pin H17, Pin G17, Pin D17
    output reg [3:0] BLUE // 4-bit VGA Blue : OUTPUT Pin N18, Pin L18, Pin K18, Pin J18/ 4-bit VGA Blue : OUTPUT
Pin N18, Pin L18, Pin K18, Pin J18
    );

    wire rst = RESET; // Setup Reset button

    // instantiate vga640x480 code
    wire [9:0] x; // pixel x position: 10-bit value: 0-1023 : only need 800
    wire [9:0] y; // pixel y position: 10-bit value: 0-1023 : only need 525
    wire active; // high during active pixel drawing
    vga640x480 display (.i_clk(CLK),.i_rst(rst),.o_hsync(HSYNC),
                        .o_vsync(VSYNC),.o_x(x),.o_y(y),.o_active(active));

    // instantiate BeeSprite code
    wire [1:0] BeeSpriteOn; // 1=on, 0=off
    wire [7:0] dout; // pixel value from Bee.mem
    BeeSprite BeeDisplay (.i_clk(CLK),.i_rst(rst),.xx(x),.yy(y),.aactive(active),
```

```
                        .BSpriteOn(BeeSpriteOn),.dataout(dout));

    // load colour palette
    reg [7:0] palette [0:191]; // 8 bit values from the 192 hex entries in the colour palette
    reg [7:0] COL = 0; // background colour palette value
    initial begin
        $readmemh("pal24bit.mem", palette); // load 192 hex values into "palette"
    end

    // draw on the active area of the screen
    always @ (posedge CLK)
    begin
        if (active)
            begin
                if (BeeSpriteOn==1)
                    begin
                        RED <= (palette[(dout*3)])>>4; // RED bits(7:4) from colour palette
                        GREEN <= (palette[(dout*3)+1])>>4; // GREEN bits(7:4) from colour palette
                        BLUE <= (palette[(dout*3)+2])>>4; // BLUE bits(7:4) from colour palette
                    end
                else
                    begin
                        RED <= (palette[(COL*3)])>>4; // RED bits(7:4) from colour palette
                        GREEN <= (palette[(COL*3)+1])>>4; // GREEN bits(7:4) from colour palette
                        BLUE <= (palette[(COL*3)+2])>>4; // BLUE bits(7:4) from colour palette
                    end
            end
        else
            begin
                RED <= 0; // set RED, GREEN & BLUE
                GREEN <= 0; // to "0" when x,y outside of
                BLUE <= 0; // the active display area
            end
    end
endmodule
```

17 "Run Synthesis" etc. and program the Basys 3 board

You should see a black coloured screen with the Bee character at the bottom of the screen, on your VGA monitor

The Memory Of The Basys 3 Board

When the board is programmed the Bitstream file created by Vivado is stored in SRAM-based memory cells within the FPGA. A bitstream (BIT) file is a binary data file that contains a bit image to be downloaded to an FPGA device

The Basys 3 board has;

1. Distributed RAM: 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)

2. Block RAM: 1,800 Kbits (225KB) of fast block RAM (BRAM)

Distributed RAMs can work without any Latency, while BRAMs have a minimum Latency of 1 clock cycle (the time it takes for data to come out of the BRAM)

In Tutorial 1 we created a register in the "Top" module called "palette" using;

```
reg [7:0] palette [0:191];
```
(See below section regarding errors found in Tutorial 1)

to represent an 8 bit (7:0) register memory array which can hold 192 entries (0:191). The register was then used to read the palette data from “pal24bit.mem” using;

```
$readmemh("pal24bit.mem", palette);
```

A Memory (MEM) file is a text file that describes contiguous blocks of data. All data values must be the same number of bits wide and must be the same width as expected by the memory model

Because the “pal24bit.mem” file is small and we have not specified which type of RAM to use, Vivado has implemented this using “Distributed RAM”

In this tutorial we have added a register called “memory_array” in the “BeeRom” module using;

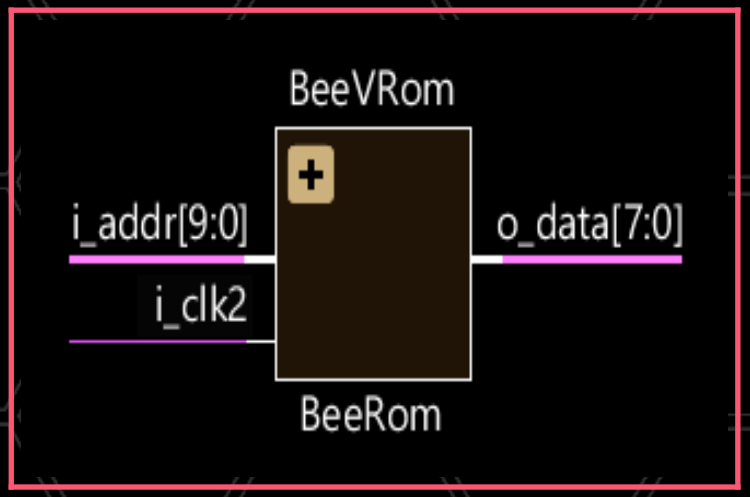
```
(*ROM_STYLE="block"*) reg [7:0] memory_array [0:917];
```

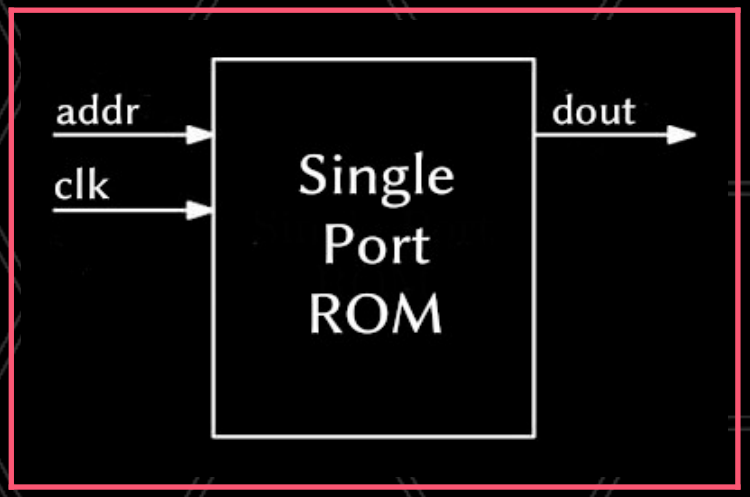
to represent an 8 bit (7:0) register memory array which can hold 918 entries (0:917). We have also instructed Vivado to specify which style of RAM to use (“block” RAM). The register was then used to read the pixel data from “Bee.mem” using;

```
$readmemh("Bee.mem", memory_array);
```

The "BeeRom" module used to read the Bee character data from the "Bee.mem" file is a;

Single Port ROM The below diagrams show a "Single Port Rom" with 2 inputs (a clock input and a 10 bit address input) and 1 output (an 8 bit data output)



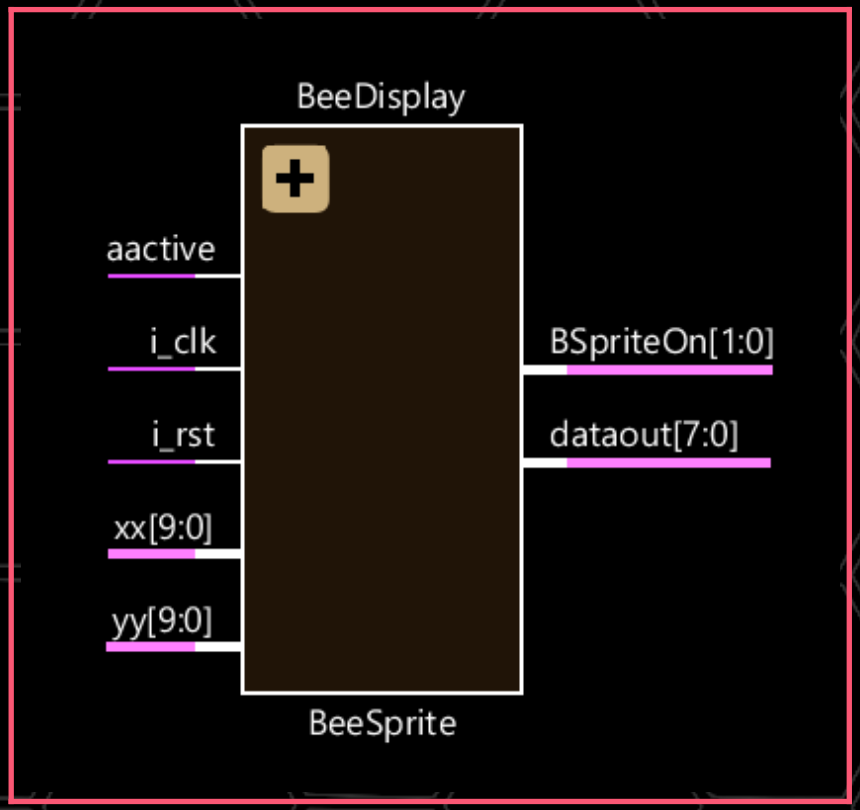


The code we are using to create a Single Port ROM for the Bee character consists of;

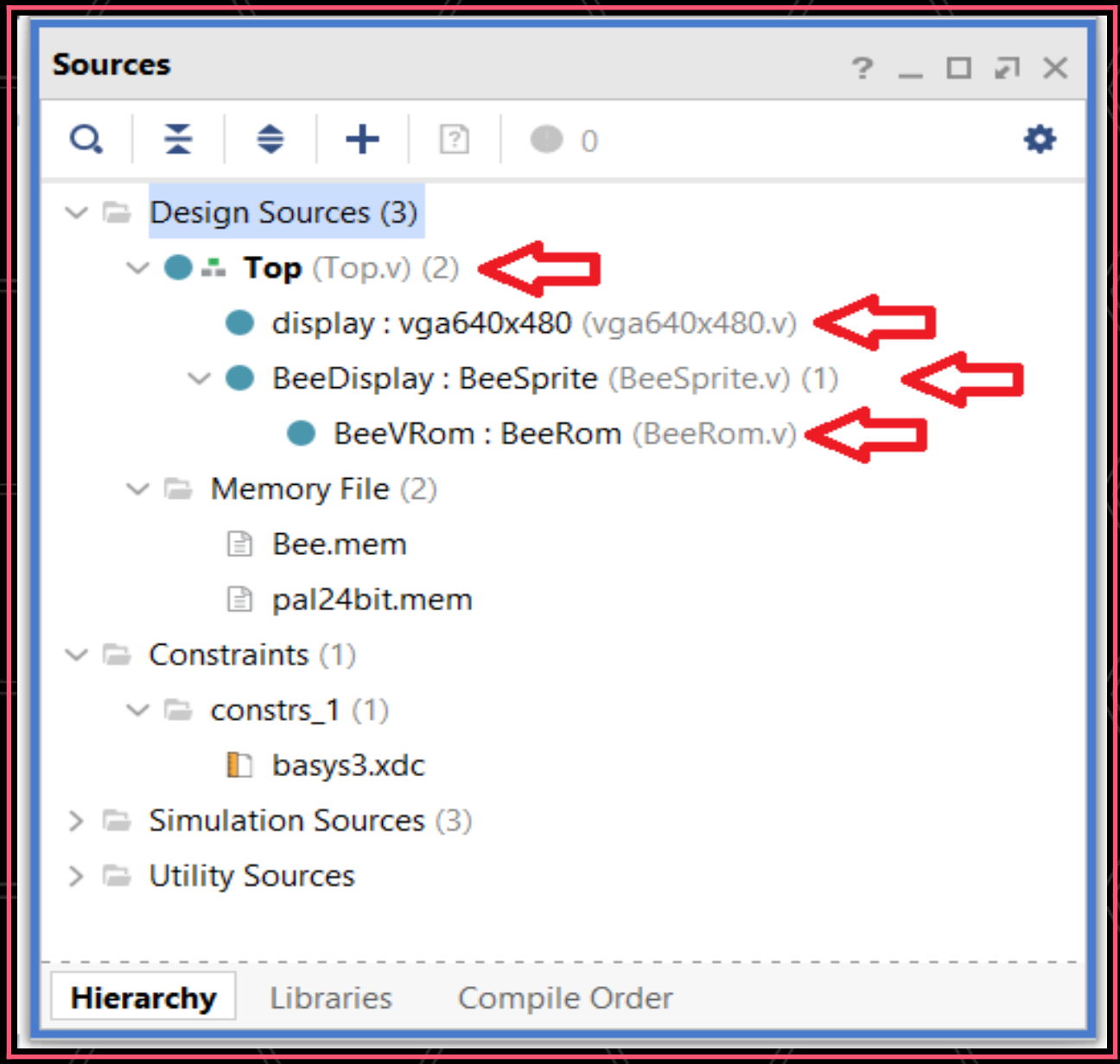
i_addr[9:0] The Bee character is 34 x 27 pixels = 918 pixels. [9:0] represents 10 bits, 2^10 or 1024, the closest number of bits above 918 pixels. "address" is used in the "BeeSprite" module ("i_addr" in the "BeeRom" module) to point to an address in the "memory_array" for the Bee character

o_data[7:0] The data found at the address of the "memory_array" is outputted in "o_data"

The Single Port ROM forms part of the BeeSprite module



Explanation Of The Code



01 Top.v module

```
// instantiate BeeSprite code
   wire [1:0] BeeSpriteOn;                                          // 1=on, 0=off
   wire [7:0] dout;                                                 // pixel value from Bee.mem
   BeeSprite BeeDisplay (.i_clk(CLK),.i_rst(rst),.xx(x),.yy(y),.aactive(active),
                         .BSpriteOn(BeeSpriteOn),.dataout(dout));
```

This links the "Top" module to the "BeeSprite" module

BeeSpriteOn: If the x,y coordinates of the Bee character match the x,y coordinates generated by the "vga840x640" routine, BeeSpriteOn will be set to 1. If there is no match BeeSpriteOn will be set to 0

dout: Will contain the hex pixel value from the "Bee.mem" file

```
// generate 25MHz pixel clock using a "Fractional Clock Divider"
reg [15:0] counter1;
reg pix_clk;
always @(posedge CLK)
    // divide 100MHz by 4 = 25MHz : (2^16)/4 = 16384 decimal or 4000 hex
    {pix_clk, counter1} <= counter1 + 16'h4000;
```

To keep the "Top" module as simple as possible, the pixel clock code has been moved to the "vga640x480" module

```
// load colour palette
reg [7:0] palette [0:191];                  // 8 bit values from the 192 hex entries in the colour palette
reg [7:0] COL = 0;                          // background colour palette value
initial begin
    $readmemh("pal24bit.mem", palette);     // load 192 hex values into "palette"
end
```

The registers colourR, colourG and colourB have now been removed as they were not required

```
// draw on the active area of the screen
   always @ (posedge CLK)
   begin
      if (active)
         begin
            if (BeeSpriteOn==1)
               begin
                  RED <= (palette[(dout*3)])>>4;          // RED bits(7:4) from colour palette
                  GREEN <= (palette[(dout*3)+1])>>4;      // GREEN bits(7:4) from colour palette
                  BLUE <= (palette[(dout*3)+2])>>4;       // BLUE bits(7:4) from colour palette
               end
            else
               begin
                  RED <= (palette[(COL*3)])>>4;           // RED bits(7:4) from colour palette
                  GREEN <= (palette[(COL*3)+1])>>4;       // GREEN bits(7:4) from colour palette
                  BLUE <= (palette[(COL*3)+2])>>4;        // BLUE bits(7:4) from colour palette
               end
         end
      else
         begin
            RED <= 0;                                     // set RED, GREEN & BLUE
            GREEN <= 0;                                   // to "0" when x,y outside of
            BLUE <= 0;                                    // the active display area
         end
   end
endmodule
```

As you can see, this section has now been modified somewhat. If BeeSpriteOn is set to 1 the Bee character will be drawn on the screen

The colourR, colourG and colourB registers have all been replaced from;

```
colourR <= palette[(COL*3)];
...
RED <= colourR[7:4];
```

simplified to;

```
RED <= (palette[(COL*3)])>>4;
```

This does the same job and in order that "RED" equals the bits [7:4] the palette value has been logically shifted right 4 times ">>4"

02 vga640x480.v module

```
// generate 25MHz pixel clock using a "Fractional Clock Divider"
reg [15:0] counter1;
reg pix_clk;
always @(posedge i_clk)
   {pix_clk, counter1} <= counter1 + 16'h4000; // divide 100MHz by 4 = 25MHz : (2^16)/4 = 16384 decimal or 4000 hex
```

The pixel clock generator has been moved from the “Top” module to the “vga640x480” module

03 BeeSprite.v module

```
//------------------------------------------------
// BeeSprite Module : Digilent Basys 3
// BeeInvaders Tutorial 2 : Onboard clock 100MHz
// VGA Resolution 640x480 @ 60Hz : Pixel Clock 25MHz
//------------------------------------------------
`timescale 1ns / 1ps

// Setup BeeSprite Module
module BeeSprite(
    input wire i_clk,
    input wire i_rst,
    input wire [9:0] xx,
    input wire [9:0] yy,
    input wire aactive,
    output reg [1:0] BSpriteOn,                          // 1=on, 0=off
    output wire [7:0] dataout
    );

    // instantiate BeeRom code
    reg [9:0] address;                                   // 2^10 or 1024, need 34 x 27 = 918
    BeeRom BeeVRom (.i_addr(address),.i_clk2(i_clk),.o_data(dataout));
```

This module links to the "BeeRom" module (a single port ROM) to load the data for the Bee character

It is also linked to the "Top" module in order that the character can be drawn on the screen

```
    // setup character positions and sizes
    reg [9:0] BeeX = 297;                                   // Bee X start position
    reg [8:0] BeeY = 433;                                   // Bee Y start position
    localparam BeeWidth = 34;                               // Bee width in pixels
    localparam BeeHeight = 27;                              // Bee height in pixels

    // check if xx,yy are within the confines of the Bee character
    always @ (posedge i_clk)
    begin
        if (aactive)
            begin
                if (xx==BeeX-1 && yy==BeeY)
                    begin
                        address <= 0;
                        BSpriteOn <=1;
                    end
                if ((xx>BeeX-1) && (xx<BeeX+BeeWidth) && (yy>BeeY-1) && (yy<BeeY+BeeHeight))
                    begin
                        address <= (xx-BeeX) + ((yy-BeeY)*BeeWidth);
                        BSpriteOn <=1;
                    end
                else
                    BSpriteOn <=0;
            end
    end
endmodule
```

BeeX & BeeY hold the x,y coordinates for the Bee character and BeeWidth & BeeHeight hold the size of the character

When the "vga640x480" x,y coordinates are in the active area ("aactive") a check is made to see when the start of the Bee coordinates (minus 1 pixel on the x position) of the Bee have been reached

At this point address is set to 0 and the first data value is obtained from "BeeRom". As explained previously, the "BeeRom" (a single port ROM) has a latency of at least 1 clock cycle. This is the reason why we retrieve the "BeeRom" data 1 pixel early to compensate for the 1 clock cycle delay.

The BSpriteOn (BeeSpriteOn in the "Top" module) is also set to 1 in order that the "Top" module can start to draw the Bee on the screen

Thereafter and as long as the "vga640x480" x,y coordinates are within the boundaries of the Bee character, the data is retrieved and linked back to the "Top" module via the dataout wire

When the "vga640x480" x,y coordinates are outside of the Bee boundaries BSpriteOn is set to 0

04 BeeRom.v module, a Single Port ROM

```
//------------------------------------------------
// BeeRom Module - Single Port ROM : Digilent Basys 3
// BeeInvaders Tutorial 2 : Onboard clock 100MHz
// VGA Resolution 640x480 @ 60Hz : Pixel Clock 25MHz
//------------------------------------------------
`timescale 1ns / 1ps

// Setup BeeRom Module
module BeeRom(
    input wire [9:0] i_addr,                        // (9:0) or 2^10 or 1024, need 34 x 27 = 918
    input wire i_clk2,
    output reg [7:0] o_data                         // (7:0) 8 bit pixel value from Bee.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] memory_array [0:917];       // 8 bit values for 918 pixels of Bee (34 x 27)

    initial begin
        $readmemh("Bee.mem", memory_array);
    end

    always @ (posedge i_clk2)
        o_data <= memory_array[i_addr];
endmodule
```

This module creates a Single Port ROM which reads the data from the “Bee.mem” file

i_addr contains the address in the “Bee.mem” memory file provided by the “BeeSprite” module

o_data will contain the hex value obtained from the “Bee.mem” file which is sent back to “BeeSprite”

When the “Bee.mem” file is read, the data is stored in memory_array, an 8 bit register which holds the 918 pixel data values for the Bee character

Notice the line of code;

(*ROM_STYLE="block"*) reg [7:0] memory_array [0:917];

This tells Vivado to store the data in the style of Block RAM (BRAM)

Suggestions

1. Code improvements

Any improvements in the code used are most welcome. Please provide details of this for consideration in using in this tutorial

2. Errors or Mistakes

Any errors or mistakes spotted are most welcome, including incorrect explanations

3. Testbenches

I would like to include Testbenches in the tutorial. It would be most helpful to receive details / explanations of the following (including steps taken to arrive at the results);

a) Loading of the Bee character file hex values

Tutorial 3

The next tutorial will include;

1. Moving the Bee character with the buttons on the Basys 3 board
2. How to display the 5 rows of Aliens on the screen
3. As in previous tutorials, suggestions will be welcome / considered, including suggestions on the graphics used

Errors Found in Tutorial 1 / Amended In Tutorial 2

Top.v Module;

reg [7:0] palette [0:192]; amended to reg [7:0] palette [0:191];

Appendix

01 Instead of providing the data for each sprite / character, I have uploaded the Sprite Sheet which can be opened in Gimp;