VGA Graphics

E155

Why Graphics?

http://jonpeddie.com/press-releases/details/analysis-jon-peddie-research-views-on-the-computer-graphics-market-2011-to-/

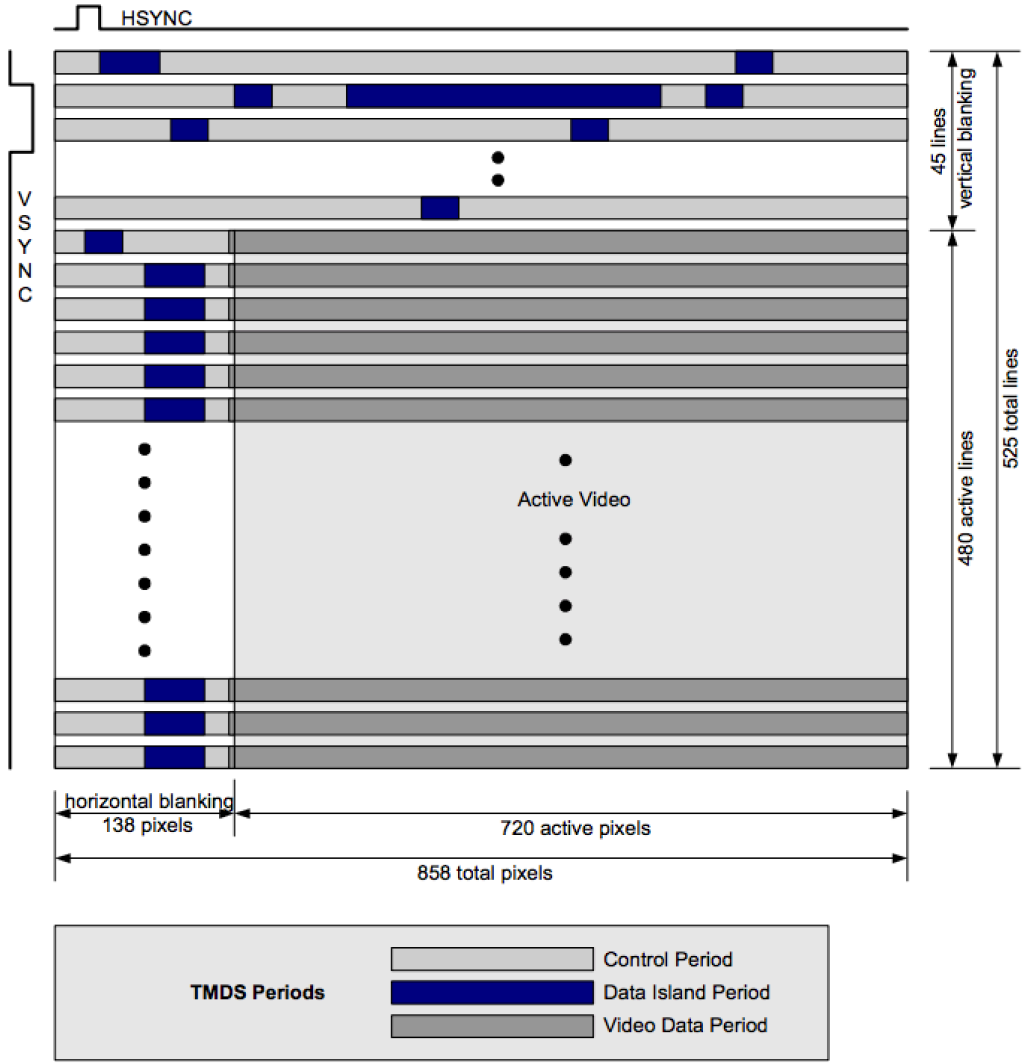
Graphics are Cool

Created with Lux Renderer

Why VGA?

- Relatively Simple
- Lab uses VGA
- Tons of displays use it
- Not fundamentally different than HDMI

HDMI



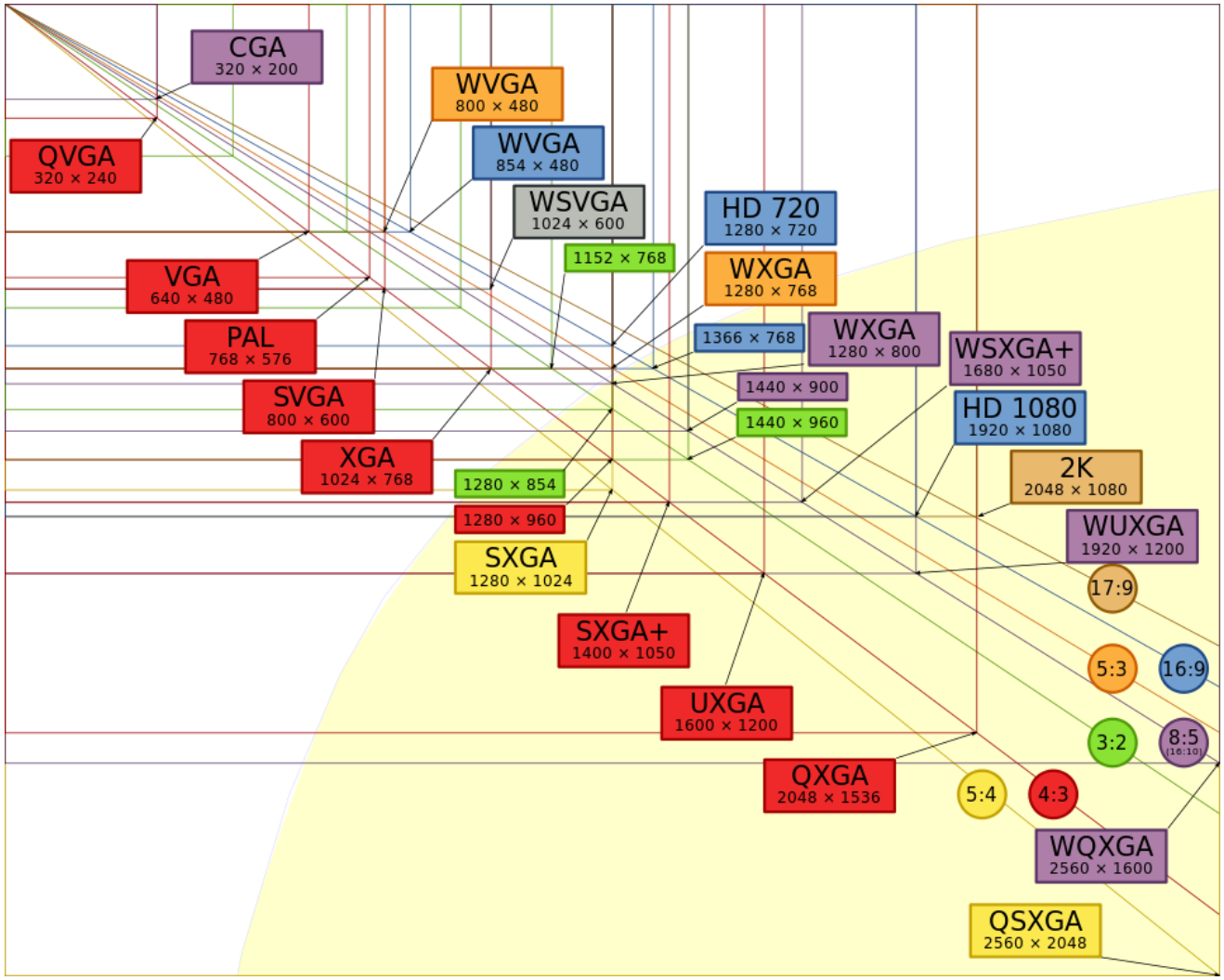
http://www.hdmi.org/pdf/HDMISpecInformationalVersion.pdf

What is VGA?

- Video Graphics Array
- IBM PS/2 computers in 1987
- Implemented as an ASIC
- 640x480 (traditionally)
- 256 kB video RAM
- 262,144 colors (6 bit/channel)

http://en.wikipedia.org/wiki/Vga

Displays



http://en.wikipedia.org/w/index.php?title=File:Vector_Video_Standards2.svg&page=1

Cathode Ray Tube

Electron Gun – About the size of a roll of quarters

http://electronics.howstuffworks.com/question694.htm

CRT

1: Three electron guns

2: Electron beams

3: Focusing coils

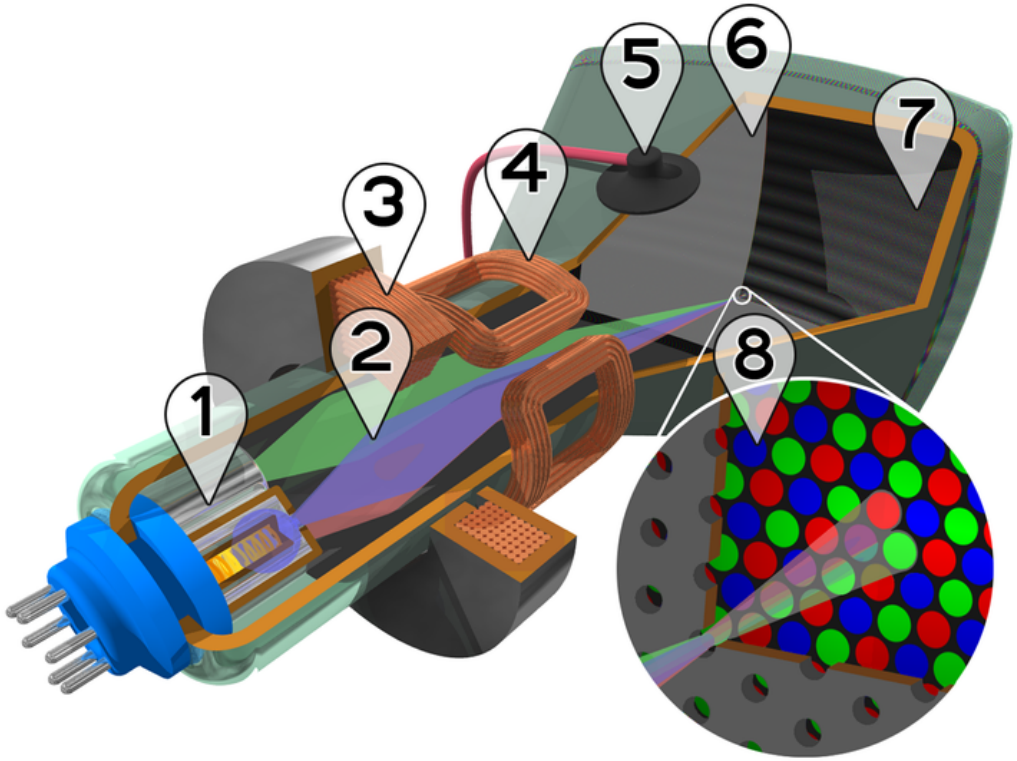
4: Deflection coils

5: Anode connection

6: Mask

7: Phosphor layer

8: Close up of 7

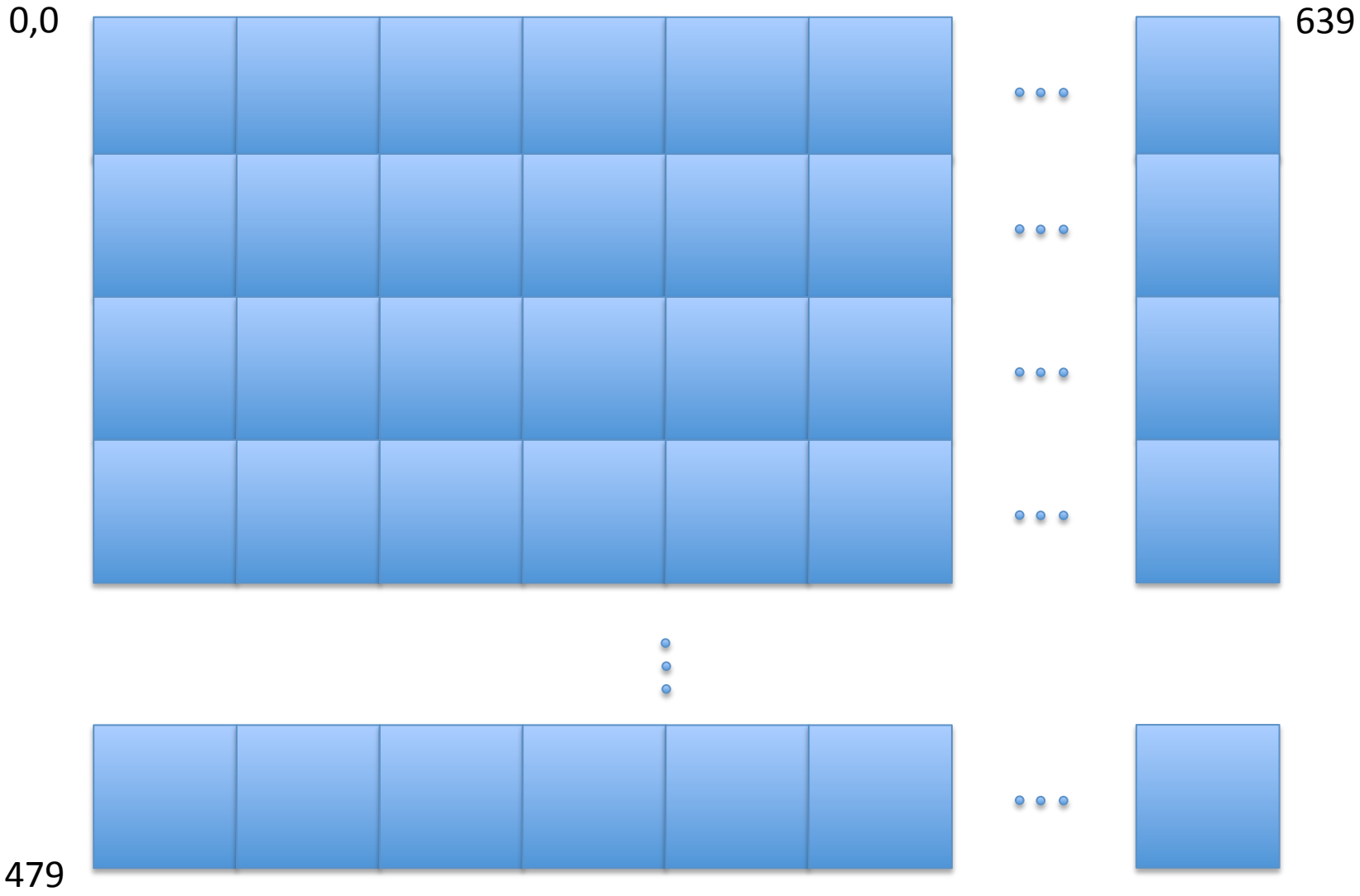


http://en.wikipedia.org/wiki/Cathode_ray_tube

Up Close

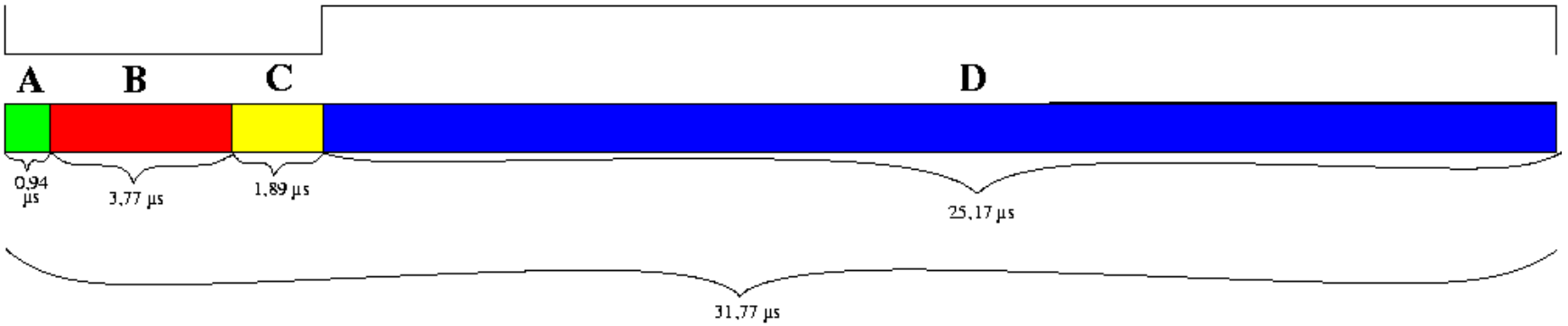
http://en.wikipedia.org/wiki/Computer_monitor

Pixels



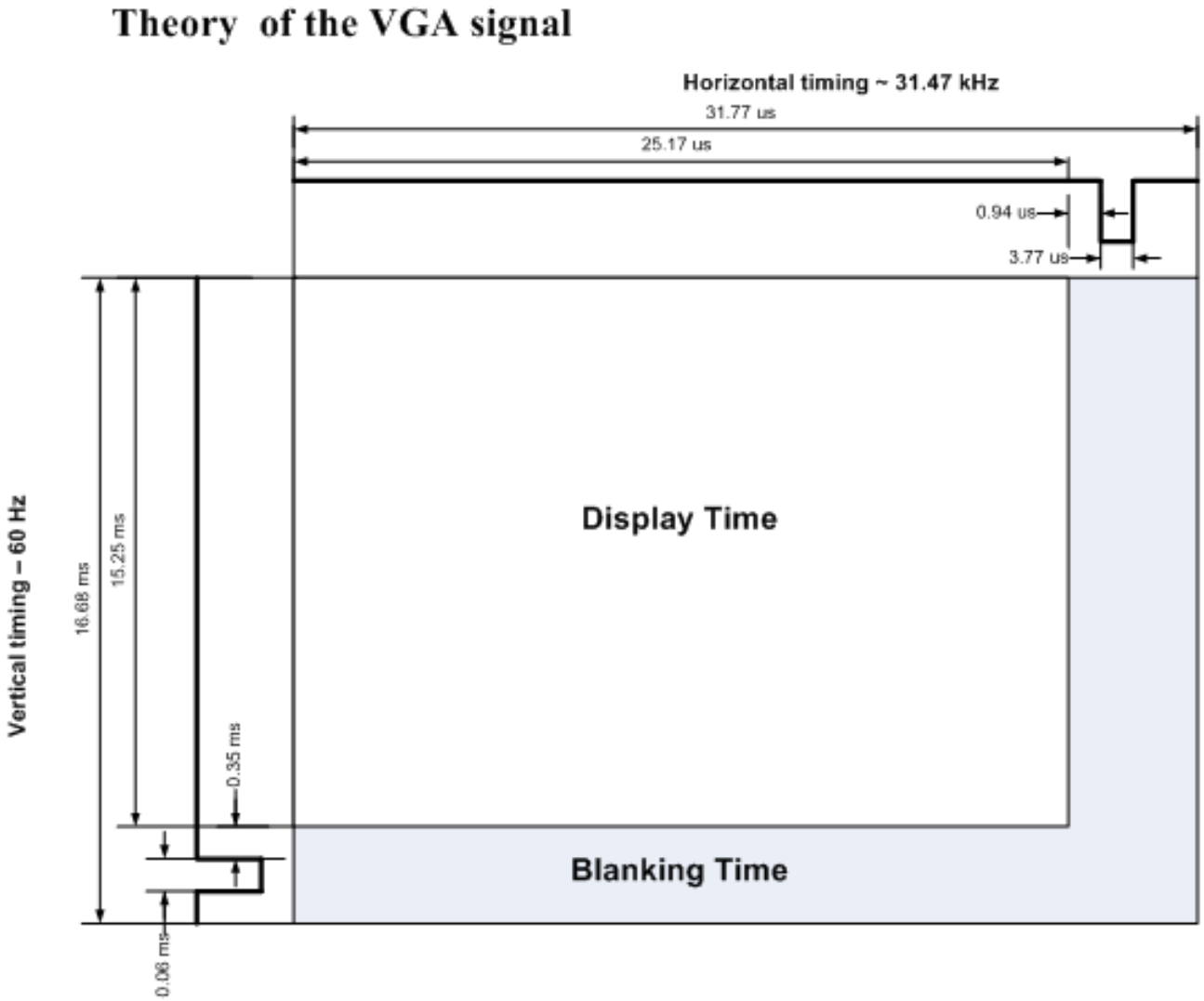
HSync Timing

- A: Front Porch 0.94 µs
- B: Sync Pulse 3.77 µs
- C: Back Porch 1.89 µs
- D: Active Video 25.17 µs



http://en.wikipedia.org/wiki/Vga

Timing



"VGA industry standard" 640x480 pixel mode

General characteristics

```
Line  frequency 31469 Hz
Field frequency 59.94 Hz
```

One line

```
  8 pixels front porch
 96 pixels horizontal sync
 40 pixels back porch
  8 pixels left border
640 pixels video
  8 pixels right border
---
800 pixels total per line
```

One field

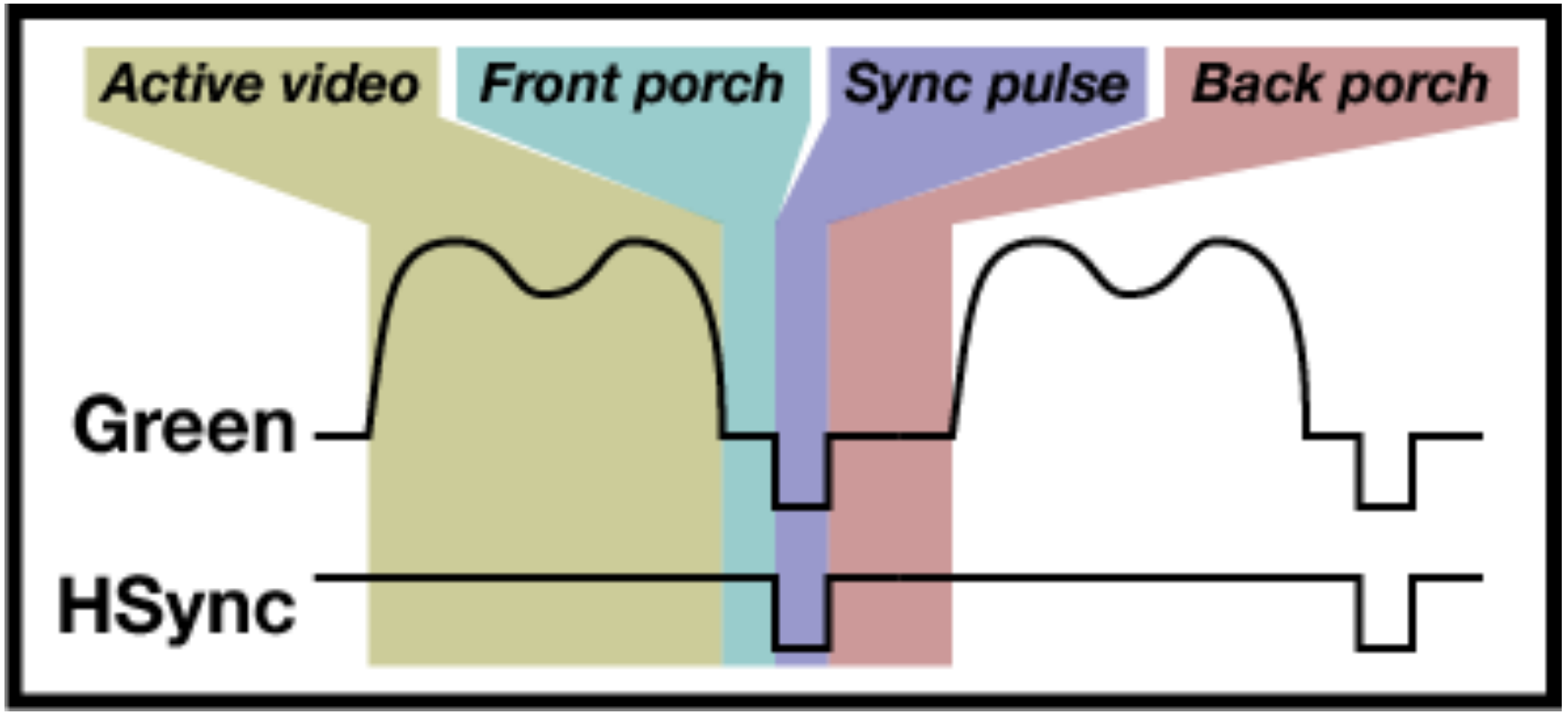
```
  2 lines front porch
  2 lines vertical sync
 25 lines back porch
  8 lines top border
480 lines video
  8 lines bottom border
---
525 lines total per field
```

Other details

```
Sync polarity: H negative, V negative
Scan type: non interlaced.
```

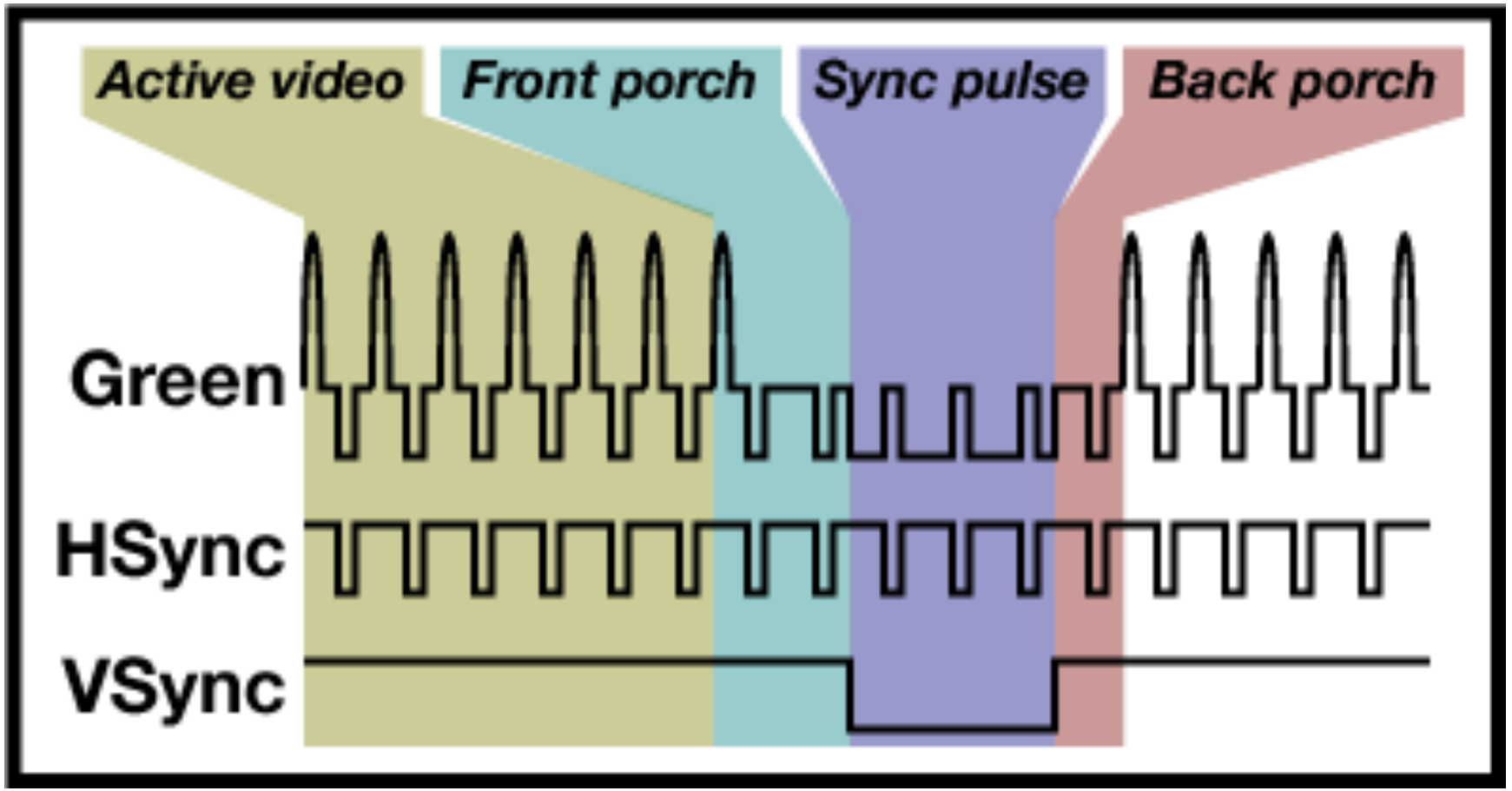
http://www.pyroelectro.com/tutorials/vhdl_vga/theory.html

Porches?



http://www-mtl.mit.edu/Courses/6.111/labkit/vga.shtml

HSync and VSync



http://www-mtl.mit.edu/Courses/6.111/labkit/vga.shtml

More Timing

Format	Pixel Clock (MHz)	Horizontal (in Pixels)				Vertical (in Lines)			
		Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
640x480, 60Hz	25.175	640	16	96	48	480	11	2	31
640x480, 72Hz	31.500	640	24	40	128	480	9	3	28
640x480, 75Hz	31.500	640	16	96	48	480	11	2	32
640x480, 85Hz	36.000	640	32	48	112	480	1	3	25
800x600, 56Hz	38.100	800	32	128	128	600	1	4	14
800x600, 60Hz	40.000	800	40	128	88	600	1	4	23
800x600, 72Hz	50.000	800	56	120	64	600	37	6	23
800x600, 75Hz	49.500	800	16	80	160	600	1	2	21
800x600, 85Hz	56.250	800	32	64	152	600	1	3	27
1024x768, 60Hz	65.000	1024	24	136	160	768	3	6	29
1024x768, 70Hz	75.000	1024	24	136	144	768	3	6	29
1024x768, 75Hz	78.750	1024	16	96	176	768	1	3	28
1024x768, 85Hz	94.500	1024	48	96	208	768	1	3	36

http://martin.hinner.info/vga/timing.html

More Timing

Format	Pixel Clock (MHz)	Horizontal (in Pixels)				Vertical (in Lines)			
		Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
640x480, 60Hz	25.175	640	16	96	48	480	11	2	31
640x480, 72Hz	31.500	640	24	40	128	480	9	3	28
640x480, 75Hz	31.500	640	16	96	48	480	11	2	32
640x480, 85Hz	36.000	640	32	48	112	480	1	3	25
800x600, 56Hz	38.100	800	32	128	128	600	1	4	14
800x600, 60Hz	40.000	800	40	128	88	600	1	4	23
800x600, 72Hz	50.000	800	56	120	64	600	37	6	23
800x600, 75Hz	49.500	800	16	80	160	600	1	2	21
800x600, 85Hz	56.250	800	32	64	152	600	1	3	27
1024x768, 60Hz	65.000	1024	24	136	160	768	3	6	29
1024x768, 70Hz	75.000	1024	24	136	144	768	3	6	29
1024x768, 75Hz	78.750	1024	16	96	176	768	1	3	28
1024x768, 85Hz	94.500	1024	48	96	208	768	1	3	36

http://martin.hinner.info/vga/timing.html

SystemVerilog VGA

```
module vga(input  logic       clk,
           output logic       vgaclk,      // 25.175 MHz VGA clock
           output logic       hsync, vsync,
           output logic       sync_b, blank_b  // to monitor & DAC
           output logic [7:0] r, g, b);    // color to video DAC

   logic [9:0] x, y;
   // setup PLL
   pll vgapll(.inclk0(clk), .c0(vgaclk));

   // generate monitor timing signals
   vgaController vgaCont(vgaclk,hsync,vsync,sync_b,blank_b,x,y);

   // user-defined module to determine pixel color
   videoGen videoGen(x, y, r, g, b);
endmodule
```

VGA Controller Verilog

```
module vgaController #(parameter
                            HACTIVE     = 10'd640,
                            HFP         = 10'd16,
                            HSYN        = 10'd96,
                            HBP         = 10'd48,
                            HMAX        = HACTIVE+HFP+HSYN+HBP,
                            VACTIVE     = 10'd480,
                            VFP         = 10'd11,
                            VSYN        = 10'd2,
                            VBP         = 10'd32,
                            VMAX        = VACTIVE+VFP+VSYN+VBP)
        (input  logic   vgaclk,
        output logic    hsync, vsync, sync_b, blank_b,
        output logic [9:0] x, y);
```

VGA Controller cont.

```
 // counters for horizontal and vertical positions
always @(posedge vgaclk) begin
    x++;
    if (x == HMAX) begin
        x = 0;
        y++;
        if (y == VMAX) y = 0;
    end
end

// compute sync signals (active low)
assign hsync = ~(hcnt >= HACTIVE+HFP & hcnt < HACTIVE+HFP+HSYN);
assign vsync = ~(vcnt >= VACTIVE+VFP & vcnt < VACTIVE+VFP+VSYN);
assign sync_b = hsync & vsync;

// force outputs to black when outside display area
assign blank_b = (hcnt < HACTIVE) & (vcnt < VACTIVE);
endmodule
```

Video Gen Verilog

```
module videoGen(input logic[9:0] x, y,
                output logic[7:0] r, g, b);
    logic pixel, inrect;

    // given y position, choose a character to display
    // then look up the pixel value from the ROM
    // and display it in red or blue. Also draw a green rectangle
    chargenrom chargenromb(y[8:3]+8'd65, x[2:0], y[2:0], pixel);

    rectgen rectgen(x,y,10'd120,10'd150,10'd200,10'd230,inrect);

    assign {r,b} = (y[3]==0)?{{8{pixel}},8'h00}:{8'h00,{8{pixel}}};

    assign g = inrect ? 8'hFF: 8'h00;

endmodule
```

Rectangle Verilog

```
module rectGen(input logic[9:0] x, y, left, top, right, bot,
               output logic inrect);

    assign inrect = (x >= left & x < right & y >= top & y < bot);

endmodule
```

ROM Verilog

```
module chargenrom(input logic[7:0] ch,
                  input logic[2:0] xoff, yoff,
                  output logic pixel);
    logic [5:0] charrom[2047:0];  // character generator ROM
    logic [7:0] line;             // a line read from the ROM

    // initialize ROM from a text file
    initial
        $readmemb("charrom.txt", charrom);

    // index into ROM to find the character
    assign line = charrom[yoff+{ch-65, 3'b000}]; //A is entry 0

    // reverse order of bits
    assign pixel = line[3'd7-xoff];

endmodule
```

ROM Contents

```
// A ASCII 65
011100
100010
100010
111110
100010
100010
100010
000000
// B ASCII 66
111100
100010
100010
111100
100010
100010
111100
000000
// ...
```

VGA Steps

1. Set up hsync timing (porches, sync, data)
2. Set up vsync timing (porches, sync, data)
3. Output data according to x,y
   - Could be procedural
   - Could be stored pixel data
   - Double buffer

SPI

- See slides from before