Access Parameters:

- Modeled/Inspired by TMS9918 and NES PPU
- 32bit data, 16bit address
- “Virtual” Address Space

<table>
<thead>
<tr>
<th>Valid Address Ranges</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000 -&gt; 0x03F</td>
<td>Attribute Table</td>
</tr>
<tr>
<td>0x1000 -&gt; 0x102F</td>
<td>Color Table</td>
</tr>
<tr>
<td>0x2000 -&gt; 0x24BA</td>
<td>Pattern Table</td>
</tr>
<tr>
<td>0x3000 -&gt; 0x3FFF</td>
<td>Sprite Table</td>
</tr>
</tbody>
</table>

```verilog
class logic [3:0] mem_write;
assign a_addr = mem_write[0] ? w_addr[5:0]: ar_addr[5:0];
assign c_addr = mem_write[1] ? w_addr[5:0]: cr_addr;

memory #(32, 64, 6) attr_table (.clk(clk), .we(mem_write[0]), .addr(a_addr[5:0]), .data_in(w_data), .data_out(sprite_attr));
memory #(32, 64, 6) color_table (.clk(clk), .we(mem_write[1]), .addr(c_addr[5:0]), .data_in(w_data), .data_out(color_out));
memory #(32, 2048, 11) pattern_table (.clk(clk), .we(mem_write[2]), .addr(p_addr[10:0]), .data_in(w_data), .data_out(pattern));
memory #(32, 4096, 12) sprite_table (.clk(clk), .we(mem_write[3]), .addr(s_addr[11:0]), .data_in(w_data), .data_out(sprite));
always_ff @(posedge clk) begin
    mem_write <= 3'b0;
    if (chipselect && write) begin
        case(address[15:12])
        4'b0000: mem_write[0] <= 1'b1;
        4'b0001: mem_write[1] <= 1'b1;
        4'b0010: mem_write[2] <= 1'b1;
        default: mem_write[3] <= 1'b1;
      endcase
      w_addr <= address;
      w_data <= writedata;
    end
end
```
<table>
<thead>
<tr>
<th>Category</th>
<th>Image</th>
<th>Size (pixels)</th>
<th>Variants</th>
<th>Total Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duck</td>
<td><img src="image" alt="Duck" /></td>
<td>32 x 32</td>
<td>4 (up, down, dead, flying away)</td>
<td>4 * 32 * 32 * 2 = 8,192 bits</td>
</tr>
<tr>
<td>Bullet</td>
<td><img src="image" alt="Bullet" /></td>
<td>16 x 16</td>
<td>1</td>
<td>1 * 16 * 16 * 2 = 512 bits</td>
</tr>
<tr>
<td>Background</td>
<td><img src="image" alt="Background" /></td>
<td>16 x 16</td>
<td>8</td>
<td>8 * 16 * 16 * 2 = 4096 bits</td>
</tr>
<tr>
<td>Score + Round (numbers)</td>
<td><img src="image" alt="Score" /></td>
<td>16 x 16</td>
<td>10</td>
<td>10 * 16 * 16 * 2 = 10240 bits</td>
</tr>
<tr>
<td>Crosshair</td>
<td><img src="image" alt="Crosshair" /></td>
<td>16 x 16</td>
<td>1</td>
<td>1 * 16 * 16 * 2 =</td>
</tr>
<tr>
<td>Color Pallet</td>
<td><img src="image" alt="Color Pallet" /></td>
<td>4 x 32 (bits not pixels)</td>
<td>8</td>
<td>4 * 32 = 128 bits</td>
</tr>
</tbody>
</table>

Each pixel location is 2 bits wide (to index into the color table!)

Total Budget Required: 23,680 bits = 2,960 bytes = 2.89 KB
Graphic Generation

- Automatically Generated Sprite Data Via Python Scripts!
Core Parameters

- \( x, y \) location of each sprite (20 bits)
- Sprite table for each sprite (8 bits)
- Color table for each sprite (4 bits)

<table>
<thead>
<tr>
<th>Param</th>
<th>Y Location</th>
<th>X Location</th>
<th>Sprite Table</th>
<th>Color Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Location</td>
<td>0 - 9</td>
<td>10-19</td>
<td>20-27</td>
<td>28-31</td>
</tr>
</tbody>
</table>
Core Parameters

- Duck 1
- Duck 2
- Score Digit 1
- Score Digit 2
- Round
- Bullet 1
- Bullet 2
- Bullet 3
- Crosshair
Cool Diagram of PPU’s datapath
Software

- **Sprites.c**
  - Used to load attributes, patterns, color palettes, and sprites into PPU at startup and during runtime.

- **Game.c**
  - Handles game mechanics such as gamestate, scoring, and duck movement.

- **Duck_hunt.c**
  - Main game loop.
  - Setups wiiuse library and handleings user input.
Game Logic

- Skeet Shooting Style Game with Multiple Rounds
- Bird Speed Increased as Game Progresses
- Birds Spawn From Fixed Location in Random Directions
- Player Has 3 Bullets to Shoot 2 Ducks Per Round
- If Player Shoots All Bullets or Misses All Ducks, The Next Round Begins
- After 8 Rounds The Game is Over
Inputs

- Wii Remote connected via Bluetooth
- WiiUse Library
  - Handles connecting to Wii Remote and handling inputs
  - duck_hunt.c uses wiiuse library to poll wii remote
- Custom Kernel built to support bluetooth drivers
- Press A to Start Game
- Move Crosshair and shoot
Future Work

- Better Start Game and End Game Screens
- More Varied Difficulty Levels Selectable by Players
- More Sprites!
  - Duck Shocked
  - Duck Mid Flap
  - A dog!
Thank you!