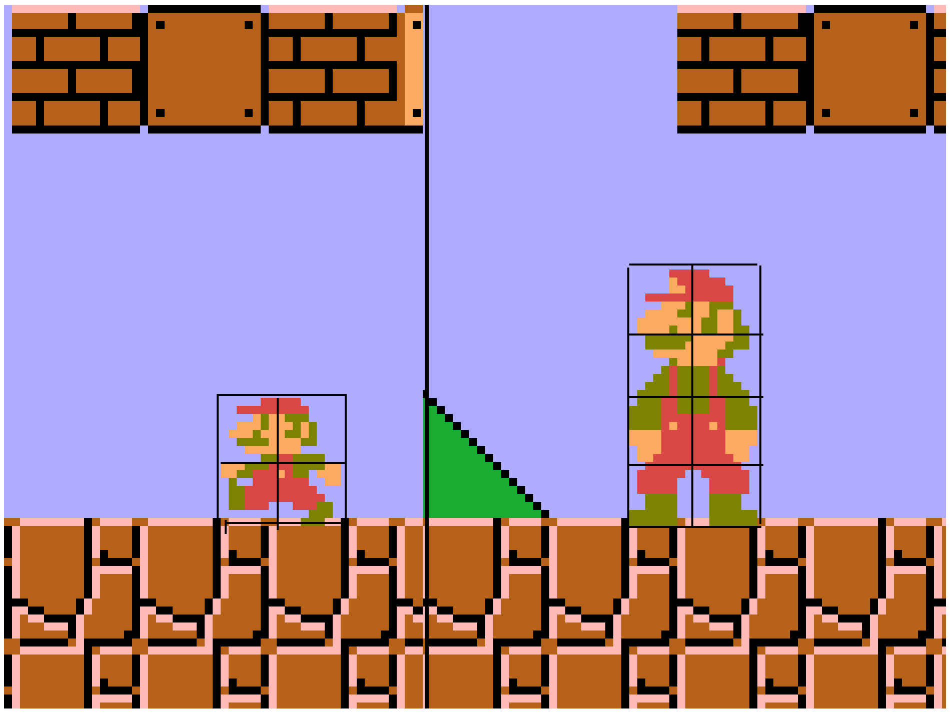
6. Sprites

A sprite is an 8×8 object that can move freely on the screen. (Technically, you can set it to 8×16 also). Nearly all characters are constructed out of sprites. There are several exceptions…some end game bosses are actually background tiles, and Little Mac in Punch-Out (also, the hero of Kung Fu) is actually made of background tiles. (The screen horizontal scroll is shifted mid-frame to give the appearance of movement).

I know what you're thinking, 8×8 is so small. That's why we connect many sprites together into a Metasprite. A 2×2 block of sprites = small Mario. A 2×4 block of sprites = big Mario.



There are a few limitations. The NES has only 64 sprites. Also, you can only put 8 sprites on the same scanline (left to right). Any more and the lower priority sprites won't show up on that scanline. Maybe you've seen games with flickering sprites, but actually the NES doesn't naturally flicker, that's just a way to keep them visible on the screen when there are too many. The programmer does this by shuffling sprite priorities each frame. A flickering sprite is better than an invisible sprite.

What's a sprite priority? Sprite 0 uses OAM addresses 0-3. Sprite 1 uses OAM addresses 4-7. Sprite 2 uses OAM addresses 8-B. Etc. Sprite 0 will always have priority over the other sprites. Sprite 1 will have the next highest priority, then Sprite 2, etc.

That means that if they occupy the same space, the lower numbered (ie. higher priority) sprites will show up on top. If there are more than 8 sprites on a scanline, the 8 lowest numbered (ie. higher priority) sprites [on that scanline] will show up and the others will disappear.

How do we get sprites to show up on screen? In the start-up code, I set the y coordinates of all sprites to $f8 (just off the bottom of the screen). [I think anything between $fo and $ff would work]. Each sprite uses 4 bytes of memory..(some docs call this OAM)

1.Y coordinate

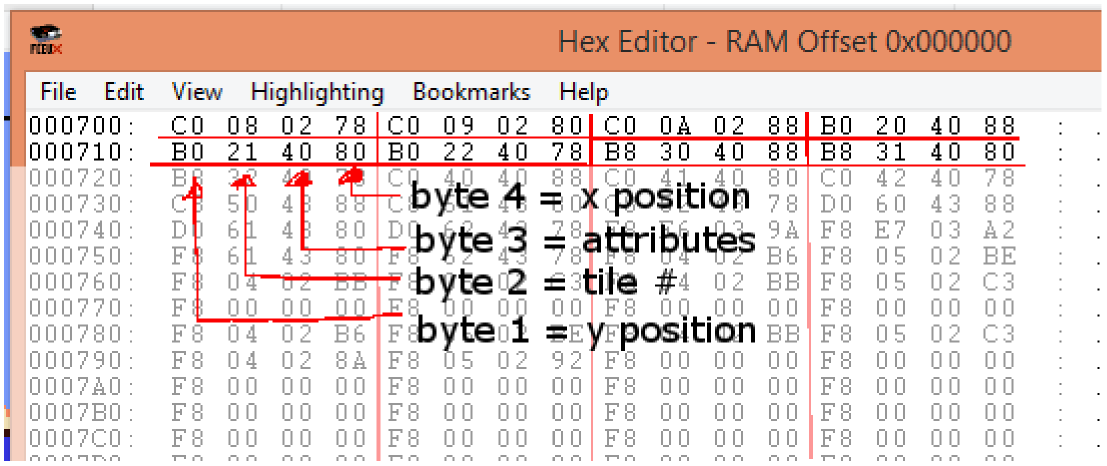
2.Tile

3.Attributes *

4. X coordinate

*attributes = flipping, which palette, whether it will appear above or below the background. See the wiki…

http://wiki.nesdev.com/w/index.php/PPU_OAM (http://wiki.nesdev.com/w/index.php/PPU_OAM)


Hex Editor - RAM Offset 0x000000
File Edit View Highlighting Bookmarks Help
byte 4 = x position
byte 3 = attributes
byte 2 = tile #
byte 1 = y position

```
000700:  C0 08 02 78 C0 09 02 80 C0 0A 02 88 B0 20 40 88  : .
000710:  B0 21 40 80 B0 22 40 78 B8 30 40 88 B8 31 40 80  : .
000720:  B8 [illegible] C0 40 40 88 C0 41 40 80 C0 42 40 78  : .
000730:  C8 50 4[illegible] 88 [illegible] 78 D0 60 43 88  : .
000740:  D0 61 4[illegible] 80 D0 [illegible] 78 F8 [illegible] 9A F8 E7 03 A2  : .
000750:  F8 61 43 80 F8 [illegible] 78 F8 04 02 B6 F8 05 02 BE  : .
000760:  F8 04 02 BB F8 [illegible] 02 BB F8 05 02 C3  : .
000770:  F8 00 00 00 F8 00 00 00 F8 00 00 00 F8 00 00 00  : .
000780:  F8 04 02 B6 F8 [illegible] BB F8 05 02 C3  : .
000790:  F8 04 02 8A F8 05 02 92 F8 00 00 00 F8 00 00 00  : .
0007A0:  F8 00 00 00 F8 00 00 00 F8 00 00 00 F8 00 00 00  : .
0007B0:  F8 00 00 00 F8 00 00 00 F8 00 00 00 F8 00 00 00  : .
0007C0:  F8 00 00 00 F8 00 00 00 F8 00 00 00 F8 00 00 00  : .
```

Note: this picture is using 0x700 as the Sprite page of RAM, whereas the example below is using 0x200 as the Sprite page of RAM.

So, to make a sprite show up, you assign it coordinates, a tile, and pick a palette. You could do this by writing the Sprite Address to $2003 and then writing the Sprite data to $2004. This is rarely done, as there is a much more efficient Memory Copy (DMA) option for sprites. So, we will use $200-2ff of the RAM to store the values, then wait for V-Blank, then write 0 to $2003, then write 2 to $4014 (DMA from RAM 200)…and all the sprite data will be transferred to the sprite memory (OAM).

In our example here, we will make a metasprite of 4 sprites. And, some code to auto move it.

(Note, sprites appear on screen 1 pixel lower than you would expect, so in order to line up to the background, you will have to subtract 1 from its y coordinate. If you look closely at the Mario graphic above, you will see his feet are 1 pixel into the floor.)

```
#pragma bss-name(push, "ZEROPAGE")

unsigned char NMI_flag;
unsigned char Frame_Count;
unsigned char index;
unsigned char index4;
unsigned char X1;
unsigned char Y1;
unsigned char move;
unsigned char move_count;




#pragma bss-name(push, "OAM")
unsigned char SPRITES[256];
// I've mapped OAM to ram addresses 200-2ff, in the cfg file




const unsigned char PALETTE[]={
0x19, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
0x19, 0x37, 0x24, 1,  0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0};



const unsigned char MetaSprite_Y[] = {0, 0, 8, 8}; // relative y coordinates



const unsigned char MetaSprite_Tile[] = {0, 1, 0x10, 0x11}; // tile numbers



const unsigned char MetaSprite_Attrib[] = {0, 0, 0, 0}; // attributes = flipping,
```

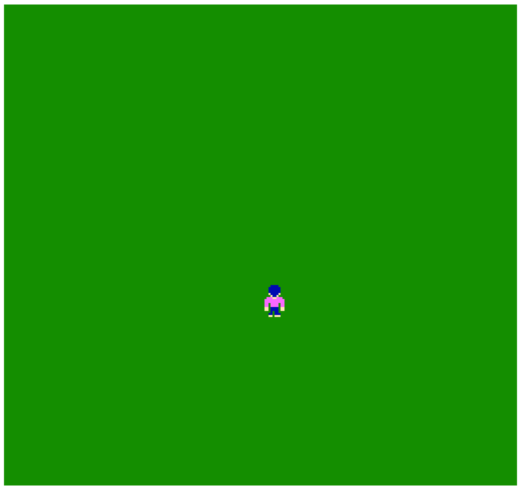
```
const unsigned char MetaSprite_X[] = {0, 8, 0, 8}; // relative x coordinates
// we are using 4 sprites, each one has a relative position from the top left spri
```

```
void every_frame(void) {

 OAM_ADDRESS = 0;
 OAM_DMA = 2; // push all the sprite data from the ram at 200-2ff to the sprite me
 PPU_CTRL = 0x90; // screen is on, NMIs on
 PPU_MASK = 0x1e;
 SCROLL = 0;
 SCROLL = 0;  // just double checking that the scroll is set to 0
}
```

```
// this automates changes to the sprites, like their position
void update_Sprites (void) {
 index4 = 0;
 for (index = 0; index < 4; ++index ){
  SPRITES[index4] = MetaSprite_Y[index] + Y1; // relative y + master y
  ++index4;
  SPRITES[index4] = MetaSprite_Tile[index]; // tile numbers
  ++index4;
  SPRITES[index4] = MetaSprite_Attrib[index]; // attributes, all zero here
  ++index4;
  SPRITES[index4] = MetaSprite_X[index] + X1; // relative x + master x
  ++index4;
 }
} // this is not very efficient, but for teaching purposes, this is clearer


void main (void) {
 All_Off(); // turn off screen
 X1 = 0x7f; // set the starting position of the sprites
 Y1 = 0x77; // near the middle of the screen
 Load_Palette();
 Reset_Scroll();
 All_On(); // turn on screen
 while (1){ // infinite loop
  while (NMI_flag == 0); // wait till NMI
  NMI_flag = 0;
  every_frame(); // should be done first every v-blank
  if (move == 0) ++X1;
  if (move == 1) ++Y1;
  if (move == 2) --X1;
  if (move == 3) --Y1;
  ++move_count;
  if (move_count == 20){ // does a move for 20 frames, then swithces moves
   move_count = 0;
   ++move;
  }
  if (move == 4) move=0; // keeps the moves between 0-3
  update_Sprites();
 }
}
```

And, I also made a slightly better version, where the sprite changes tiles for each move, so he appears to be turning. Relevant changes were…

```
const unsigned char MetaSprite_Tile[] = { //more tiles
 2, 3, 0x12, 0x13, // right
 0, 1, 0x10, 0x11, // down
 6, 7, 0x16, 0x17, // left
 4, 5, 0x14, 0x15}; // up



void update_Sprites (void) {
 move4 = move << 2; // same as move * 4
 index4 = 0;
 for (index = 0; index < 4; ++index ){
  SPRITES[index4] = MetaSprite_Y[index] + Y1; // relative y + master y
  ++index4;
  SPRITES[index4] = MetaSprite_Tile[index + move4]; // tile numbers
  ++index4;
  SPRITES[index4] = MetaSprite_Attrib[index]; // attributes, all zero here
  ++index4;
  SPRITES[index4] = MetaSprite_X[index] + X1; // relative x + master x
  ++index4;
 }
}
```



And here is the link to the source code…

http://dl.dropboxusercontent.com/s/v2wl2aa5gbrjmad/lesson4.zip (http://dl.dropboxusercontent.com/s/v2wl2aa5gbrjmad/lesson4.zip)

November 22, 2015April 15, 2017 dougfraker

12 thoughts on "6. Sprites"

1. **coder225** says:
   June 2, 2016 at 9:28 am Edit
   hey my sprite wont work all i get on my screen is a black screen but my chr character is there when i look in ppu viewer (debug) only on one the left side please help here is my code:

   ```
   void ppu_load_palette(const unsigned char* palette)
   {
   int i;
   *PPUA = 0x3f;
   *PPUA = 0x00;

   for(i = 0; i < 32; i++)
   *PPUW = palette[i];
   }

   const unsigned char PALETTE[]={
   0x0f, 17, 26, 30};

   void Load_Palette(void) {
   PPU_ADDRESS = 0x3f;
   PPU_ADDRESS = 0x00;
   for( index = 0; index < sizeof(PALETTE); ++index ){
   PPU_DATA = PALETTE[index];
   }
   }
   ```

   //I edited your comment down to the relevant part I was referring
   // –Doug

   Reply

   dougfraker says:
   June 2, 2016 at 12:05 pm Edit
   You are only loading 4 colors, to 3f00, that's the BG. That might be your problem, no sprite colors have been defined. Try to write the entire 32 colors to the palette.

   Also, it doesn't look like your main function calls a Palette load function.

   Reply

2. **coder225** says:
   June 18, 2016 at 2:47 pm Edit
   also pls help it says Error: include file 'nes.h' is not found pls help

   Reply

   dougfraker says:
   June 22, 2016 at 5:44 pm Edit

neslib.h is a bunch of function prototypes, it comes with Shiru's library. Make sure the path to it is correct.

Reply

dougfraker says:
June 22, 2016 at 5:48 pm Edit
Oh, you mean nes.h, that's in the 'include' folder of cc65. It has nothing useful in it, you can omit that.

3.
Coder226 says:
June 23, 2016 at 6:14 pm Edit
How do you compile now as the latest cc65

Reply

dougfraker says:
July 3, 2016 at 2:50 am Edit
Good question. I'll have to look into that some time.

Reply

4.
Bawenang says:
September 26, 2016 at 6:52 pm Edit
Hi,

I want to know where do you get the sprites from? I mean the whole image of the sprites.

I know that you are using the update_Sprites to change the sprite's data. Quite possibly the sprite's image is changed form the second line (SPRITES[index4] = MetaSprite_Tile[index]; // tile numbers). Just my assumptions since I don't really know what's going on. But how do you change it? How do I change the sprite's image to something like a cat? I don't see any data that can construct an image of a character with black hair etc.

Thanks.

Reply

dougfraker says:
September 27, 2016 at 11:07 am Edit
Try editing the CHR file in a tile editor.

Reply

5.
Zed says:
June 2, 2017 at 3:03 pm Edit
Hello,
It's seems like you don't give too much indications about the reset.s and it looks like it's a big part of the programs.

Thanks

Reply

dougfraker says:
June 2, 2017 at 6:27 pm Edit

It’s the startup code. It turns off interrupts, and zeroes everything. It sets up the stack and some things that need to happen before you jump to main().

Reply

6. **Zed** says:
June 4, 2017 at 9:40 am Edit
Hello, I don’t find the part of the code which helps you to draw the little guy…

Reply