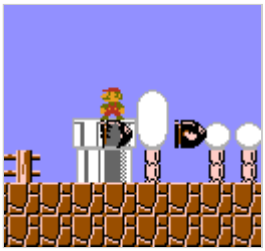
PPU sprite priority

From Nesdev wiki

In the NES PPU, each sprite has two values that affect **priority**, or what appears behind what: the index of the sprite within OAM (0 to 63), and the priority bit (attribute 2 bit 5, set to 0 for front or 1 for back).



Piranha plant sprite with lower index and back priority overlaps bullet sprite with higher index and front priority

Putting a back-priority sprite at a lower OAM index than a front-priority sprite can cover up the the front-priority sprite and let the background show through. *Super Mario Bros. 3* uses this for power-ups sprouting from blocks, by putting a non-transparent back-priority sprite "behind" the block at a low index and putting the power-up at a higher index. (You can see the corners of the back-priority sprite when Mario hits a note block in World 1-2, as the note block becomes more squared off.) The advantage of this approach is that the power-up can be hidden behind the block and still have front priority, meaning the area above the block doesn't have to be pure bg pixels like in Super Mario Bros.

The Nintendo DS PPU handles priority the "obvious" way,[1] (http://problemkaputt.de/gbatek.htm#dsvideoobjs) and some NES emulator developers initially think the NES PPU handles it the same way:

1. Front priority sprites in front
2. The background plane in the middle
3. Back priority sprites in back

What really happens in the NES PPU is conceptually more like this:

1. During sprite evaluation for each scanline (cycles 65 through 240), the PPU searches for the eight frontmost sprites on this line. Then during sprite pattern fetch (cycles 257 to 320), these eight sprites get drawn front (lower index) to back (higher index) into a buffer, taking only the first opaque pixel that matches each X coordinate. Priority does not affect ordering in this buffer but is saved with each pixel.
2. The background gets drawn to a separate buffer.
3. For each pixel in the background buffer, the corresponding sprite pixel replaces it only if the sprite pixel is opaque and front priority or if the background pixel is transparent.

The buffers don't actually exist as full-scanline buffers inside the PPU but instead as a set of counters and shift registers. The above logic is implemented a pixel at a time, as PPU rendering explains.

Detailed internals of the sprite priority quirk

During sprite evaluation the PPU copies the sprites that are in y range from the primary to the secondary OAM, from which eight internal sprite output units are then initialized. These sprite output units are wired such that the lowest-numbered unit that outputs a non-transparent pixel always wins, regardless of front/back sprite priority and regardless of what the background pixel at the corresponding location is.

Hence, when a back-priority sprite is hidden behind non-bg background pixels, it will still hide output from higher-numbered sprite output units wherever it has a non-transparent pixel.

Retrieved from "https://wiki.nesdev.com/w/index.php?title=PPU_sprite_priority&oldid=13081"

- This page was last modified on 2 November 2016, at 04:24.