Rendering Overview

The Game Boy outputs graphics to a 160x144 pixel LCD, using a quite complex mechanisr to facilitate rendering.

> **TERMINOLOGY**
>
> Sprites/graphics terminology can vary a lot among different platforms, consoles, users and communities. You may be familiar with slightly different definitions. Keep also in mind that some definitions refer to lower (hardware) tools and some others to higher abstractions concepts.

Tiles

Similarly to other retro systems, pixels are not manipulated individually, as this would be expensive CPU-wise. Instead, pixels are grouped in 8x8 squares, called *tiles* (or sometime “patterns”), often considered as the base unit in Game Boy graphics.

A tile does not encode color information. Instead, a tile assigns a *color ID* to each of its pixels, ranging from 0 to 3. For this reason, Game Boy graphics are also called *2bpp* (2 bit per pixel). When a tile is used in the Background or Window, these color IDs are associate with a *palette*. When a tile is used in an OBJ, the IDs 1 to 3 are associated with a palette, b ID 0 means transparent.

Palettes

A palette consists of an array of colors, 4 in the Game Boy’s case. Palettes are stored differently in monochrome and color versions of the console.

Modifying palettes enables graphical effects such as quickly flashing some graphics (damage, invulnerability, thunderstorm, etc.), fading the screen, “palette swaps”, and mor

Layers

The Game Boy has three “layers”, from back to front: the Background, the Window, and th Objects. Some features and behaviors break this abstraction, but it works for the most pa

Background

The background is composed of a *tilemap*. A tilemap is a large grid of tiles. However, tiles aren’t directly written to tilemaps, they merely contain references to the tiles. This makes reusing tiles very cheap, both in CPU time and in required memory space, and it is the ma mechanism that helps work around the paltry 8 KiB of video RAM.

The background can be made to scroll as a whole, writing to two hardware registers. This makes scrolling very cheap.

Window

The window is sort of a second background layer on top of the background. It is fairly limited: it has no transparency, it’s always a rectangle and only the position of the top-left pixel can be controlled.

Possible usage include a fixed status bar in an otherwise scrolling game (e.g. *Super Mario Bros. 3*).

Objects

The background layer is useful for elements scrolling as a whole, but it’s impractical for objects that need to move separately, such as the player.

The *objects* layer is designed to fill this gap: *objects* are made of 1 or 2 stacked tiles (8x8 8x16 pixels) and can be displayed anywhere on the screen.

> **NOTE**
>
> Several objects can be combined (they can be called *metasprites*) to draw a larger graphical element, usually called “sprite”. Originally, the term “sprites” referred to fixed-sized objects composited together, by hardware, with a background. Use of the

> term has since become more general.

To summarise:

- **Tile**, an 8x8-pixel chunk of graphics.
- **Object**, an entry in object attribute memory, composed of 1 or 2 tiles. Independent o the background.