CS2261 Media Device Architecture

Announcements: On our last HW. Afterwards, we’ll be moving on to our final project and the milestones

Sprites:

* Independent on screen, the hardware handles the movement of sprites without disturbing the background
  + Just need to change the memory location and rest is taken care of
* GBA supports a maximum of 128 sprites, ranging in size from 8x8 to 64x64
  + Shape of sprites is not restricted to squares
  + Example: the knight in Elements of Darkness demo was a rectangle

DEMO

* There are 6 things needed to put a sprite on the screen

1. Tile palette (SEPARATE FROM BG PALETTE)
2. Tile set
3. Tile map
4. Sprite palette
5. Sprite sheet
6. Sprite attributes

* We use USENTI for a majority of these
* Similar to tile generation
* DEMO: put all the sprites on one palette in USENTI and export them
  + Be sure to check tile, 4bpp (preferably, for more sprite memory), Pal, but don’t check Map
* Color Modes:
  + 4bpp: 16 colors in a palette
    - 32 bytes per 8x8 sprite
  + 8bpp: 256 colors
    - 64 bytes per 8x8 sprites

BGs and Sprites can use two different color modes

* In the bathtub and scrubs demo, the background was 256 colors (8bpp), and the scrubs were 16 colors (4bpp)

USENTI sheet has 32x32 tiles

* 32x32 = 1024 tiles
* There is a difference between 4bpp tiles and 8bpp tiles
  + 1024 tiles \* 64 pixels/tile = 65536 total pixels
  + 8bpp = 65536 bytes = 64 Kilobytes
  + 4bpp = 65536 bytes = 32 Kilobytes

Only have 32 KB of sprite memory (2 Character Blocks, each 16 KB)

* If you work with 4bpp tiles on the 32x32 tiles, then the memory perfectly fills up the two CBs

A CB is 16KB, the smallest sprite size is 8x8 = 32 bytes

* So a CB can hold 512 small sprites
  + (16384 bytes/32 bytes)
* 256 colors (8bpp) = ½ the amount
* With the largest sprite (64x64), at 256 colors = 4096 bytes

Finding Sprites:

* Sprites stored in the last two Character Blocks (4 and 5)
  + Can be broken into 1024 32-byte chunks
* 32 bytes (one chunk) is the size of one 4bpp 16-color tile
  + 1 tile can fit into 1 chunk
* When we want GBA to draw a sprite, we have to tell the GBA which chunk the sprite is in
  + Which of the 1024 32-byte chunks it’s in
    - Each chunk has its own index
* For 4bpp, to find the index of the sprites location, there’s a register that allows you to set indexing
  + The 2D image created from USENTI was made into an array
* Look at the USENTI grid for the index of the upper left tile, use the formula row\*32 + col to get the index

Object Attribute Memory (OAM)

* Object refers to sprites, even though sprites aren’t really objects
* 1KB in size
* 4 attributes for each sprite
  + Each sprite has a sequence of bits representing it
* Each attribute controls some aspect of the sprite, like size, shape, visibility etc
* Except the last attribute, it just takes up space