CS2261 Media Device Architecture

HW5 due tonight, HW6 released tonight

Sprites:

* In Mode 0, we can’t set individual pixels
  + Therefore, no drawstring or drawRect, etc
* Tile-based solution to this
* Sprites: tile-based images
  + Sprites have the property of transparency
  + Tile-based images
  + Sprites can also go halfway offscreen without error

How they work:

* Instead of tiles and maps, we have tiles and attributes
  + Be sure not to check Map in USENTI
* Tiles are what your sprites looks like
* Also there are two separate palettes for backgrounds and sprites
* Each sprite has 4 attributes
  + Each attribute is 1 16-bit register

Sprite Tiles

* 2 kinds of sprite tiles, 8bpp or 4bpp
* Array of tiles in USENTI will be the sprite sheet
* If 4bpp, your sheet will be 32 x 32 tiles
  + More room than with 8bpp
* If 8bpp, it’ll be 16 x 32 tiles
  + Don’t do this, it wastes space and the TAs don’t know much about it
* All sprites share on sprite sheet, not each sprite with individual sprite sheet
  + Also share same palette
  + Must know the location of where each sprite is on the sheet to work

Sprite Attributes

* Located in Object Attribute Memory (OAM)
  + 1KB
  + Sprites are not objects, even though they are controlled by Object Attribute Memory
  + Room for 128 sprites
* Each attribute is a 16 bit register
* 4 attributes per sprite
  + Each attribute is 16 bits, 2 bytes,
  + 128 x 4 (attributes) x 2 bytes = 1024 bytes
    - 1 KB
    - Only enough room for 128 sprites
* 128 is not a max though, it’s more like a set amount. There must always be 128 sprites
  + By default, all are visible
* Must iterate through and set all of them to not visible

Attributes

* Each bit does something different, controlling the size, shape, visibility,
  + Some are more complicated
    - Like alpha blending and mosaic
    - Don’t need to worry about it but can use it
* In Attribute 1, there is drawing priority and palette row
  + Bits 10-11
  + Oddly enough, the higher priority ones are drawn first, but are covered over by the other ones
    - 0 = draw front
    - 3 = draw first, but covered by others
  + Bits 12-15
  + Palette row (used only for 4bpp)
  + Similar to rows of palettes in 4bpp for backgrounds
  + 4bpp can only have 16 colors a row, but each sprite can have its own 16 color-set as its row
  + Macro: ATTR2\_PALROW
* Attribute 3 is useless but still needs to be accounted for and space is still taken up by it
* Size and Shape determined by
  + Attribute 0: bits 14-15 (shape)
    - 0-square, 1-wide, 2-tall
  + Attribute 1: bits 14-15 (size)
    - 0-tiny, 1-small, 2-medium, 3-large

Sprites stored in the last 2 Character Blocks

* CBs 4 and 5
* Use DMANow to store them