CS2261 Media Device Architecture

Announcements:

* HW5 due Thursday
* Lab 7 due Thursday
* Quiz Monday

Notes:

Mode 0:

* Right now, with Modes 3 and 4, we can’t have transparency and we have trouble drawing objects.
  + Solution: Make everything into an image
    - New issue: too much space needed for this
  + Solution pt.2: have the many small pictures spliced together and repeated often to create on big image
  + To achieve this, we have to store tiles and their locations (maps)
    - Similar to storing color in a palette and referring to their location with indices

Mode 0:

* Tile display mode (in contrast to modes 3 and 4’s bitmap mode)
  + Video Memory (VM) doesn’t map directly to pixels here
  + VM still starts at 0x6000000
  + VM is broken into character blocks and screen blocks
    - Charblocks store tiles (4 total)
    - Screen blocks store maps
      * they map tiles to the background
* Three factors controlling what will show up: tiles, maps and offset

Tiles:

* Each is a 8x8 pixel bitmap image
* Each bit is mapped to a pixel
* 64 pixels a tile
  + If you want a 16x16 image, divide it into 4 8x8 tiles
* Each pixel of the tile is stored in an index of the game
  + Palette
* Two possible tile types: 8 bits per pixel and 4 bits per pixel
  + 8bpp – each pixel stored as a char and there are 256 possible colors for each pixel.
    - 64 bytes per tile
  + 4bpp – each pixel stored as half a char, and 16 possible colors for each pixel.
    - 32 bytes per tile
* What tiles look like in memory:
  + Tiles are the actual bitmap images and are stores are palette indices
  + Refer to slides for exact examples for 8bpp and 4bpp

Maps:

* A tile is an array of images, individual pixel indices of the palette
* Map is an array of screen entries (a tile and some information about it)
* Each screen entry is 16 bits
  + Bits 0-9 – tile index (2^10 = 1024 tiles possible)
  + Bit 10 – flip horizontally
  + Bit 11 – flip vertically
  + Bits 12-15 – which row of the palette we’re going to use,
    - “palette blank”
    - only matters for 4bpp
* Flipping tiles reduces the number of tiles actually created
* In 4bpp, screen entries store the row of the palette the tile uses for colors in the last 4 bits (12 – 15)
  + If two tiles are identical except the palette row they get colors from, then they are stored as one tile and the screen entry determines the palette row

Side note: know that there are 8 bits in a byte