CS2261 Media Device Architecture

Announcements:

* Quiz next Wednesday (offsets will be on there as well as bitwise operations)
* Lab tomorrow in recitation

Notes:

* Lighting up a pixel
  + In GBA Mode 3, the screen is 240x160 with 38,400 pixels total
  + If you want to light up some arbitrary pixel, then you need to know it’s location
  + Offsets provide a solution to this
    - Allow you to calculate the distance between the base and target element
    - 2 bytes per pixel, so in memory, the distance from one pixel to the next is 2 bytes
      * shorts are 16 bits and GBA is byte (8 bits) addressable

|  |  |
| --- | --- |
| Bytes of memory | Pixel |
|  | 0 |
|  |
|  | 1 |
|  |
|  | 2 |
|  |
|  | … |
|  |

* + - * 0x6000000 (origin) + 2 = pixel at coordinates (0,1)
      * Another thing is that the pixels go like:

|  |
| --- |
|  |
|  |
|  |
| * + - * + … |

* + - Traversing to a certain pixel is:
      * 0x6000000 + MEM\_OFFSET
    - MEM\_OFFSET = PIX\_OFFSET\*2
      * PIX\_OFFSET(row, col, rowlen) = ((r)\*(rowlen)+(c))
    - PIX\_OFFSET: (row\*rowlength) will get you to the row you want to be in, and adding the (column) will get you to the right column. This is multiplied by 2 in the MEM\_OFFSET formula (because each pixel has 2 bytes to it, so to traverse to the address of the pixel you want to access, it’s multiplied by 2). Then this is added to 0x6000000.
  + The PC builders made all this easier with pointer arithmetic
    - Automatically moves you over the places in memory you want to move when you add offset to a memory location
    - PIX\_OFFSET(80, 120, 240) = offset from 0,0 to the center
    - \*( (unsigned short \*)(0x6000000) + PIX\_OFFSET(80, 120, 240) ) = GREEN
    - \*( videoBuffer + PIX\_OFFSET(80, 120, 240) ) = GREEN
* • C has special syntax for referencing locations –
  + pointer\_name[some\_number]
  + videoBuffer[57]
  + the location you land after passing 57 shorts down from videoBuffer
    - a[b] = \*(a+b)
    - Adds the pointer and the offset (a and b has to be one or the other) and then dereferences
    - Idea is that left side is cleaner
    - videoBuffer[PIX\_OFFSET(80,120,240)] = GREEN;
    - \*(videoBuffer + PIX\_OFFSET(80,120,240)) = GREEN

Prototypes

* Prototypes are similar to the interface in Java
* Essentially, you declare the skeleton of any function at the top of your C code above the main function and below all the # lines of code
  + It avoids errors and allows you more independence in using functions within functions and referencing things you haven’t fully declared
    - Necessary because C is read linearly, so if you are using a function2 in your funciton1 and you have yet to fully declare function2, then you’ll get an error

Bouncing Rectangle:

* Delay: when we were trying to delay how fast the rectangle was moving on the screen, we added in for-loops that went through a large number and incremented some variable inside the loop
  + compiler basically ignores that loop because whatever is in it isn’t being used significantly throughout the function
  + To force the compiler to not ignore it, use the modifier “volatile”