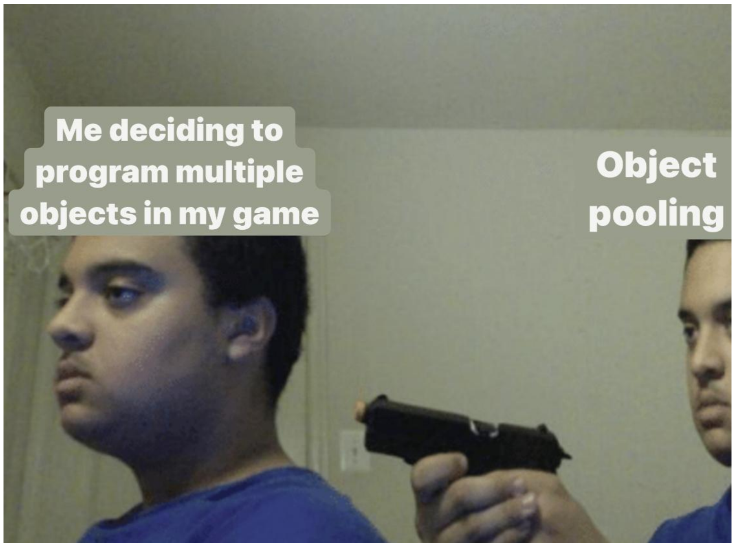
Object Pooling, Text in Mode 3



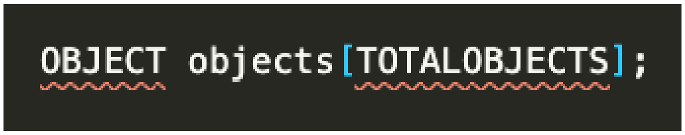
Me: I'm good in C Language.

Interviewer: Write HELLO WORLD using C.

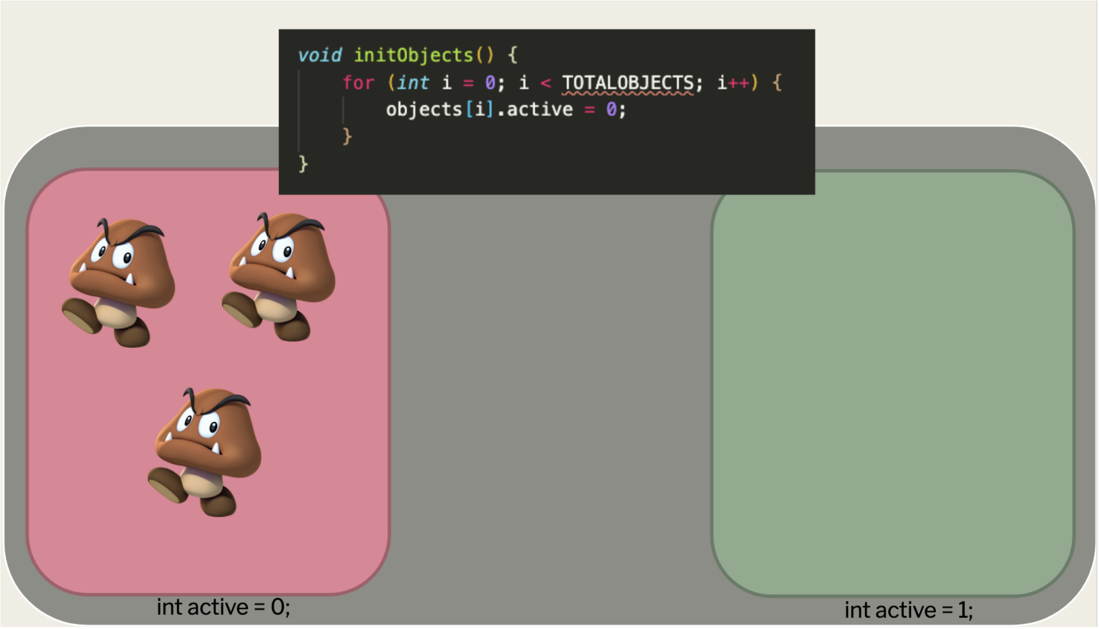
Me:

Object Pooling

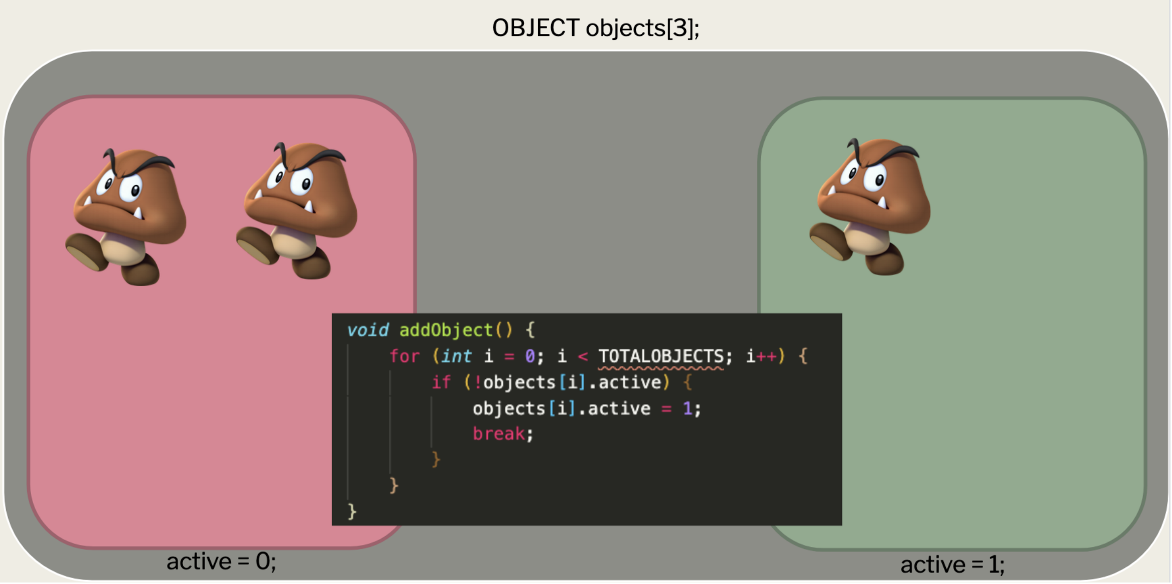
- Object pooling is a technique found in most video games that use multiple instances of the same object
- For example, bullets, or enemy entities
- It makes it much easier to create or remove entities as they are pre initialized in memory at the start of the game or scene
- In C, we use structs for this, and arrays to hold all the instances of the struct



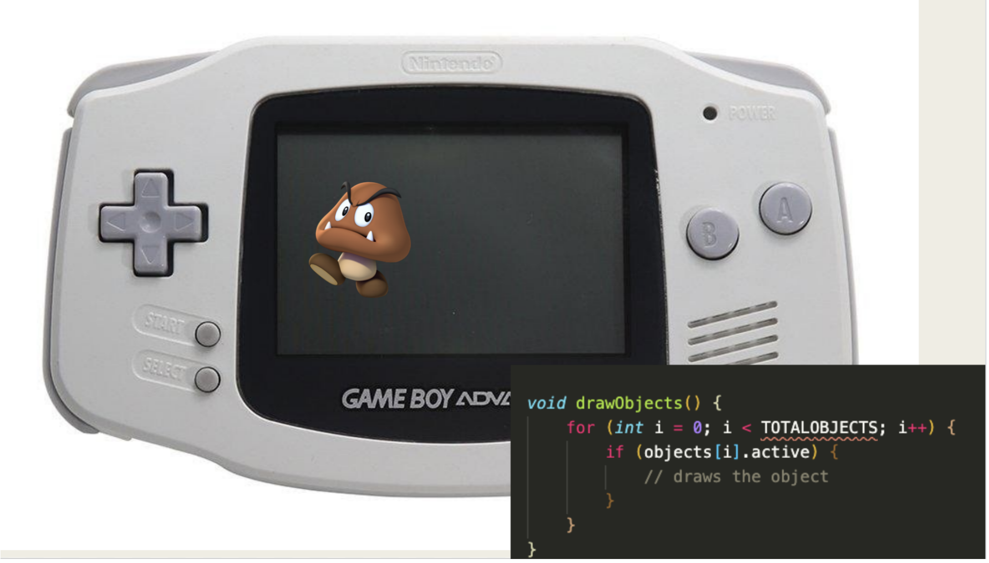
```
OBJECT objects[TOTALOBJECTS];
```



```
void initObjects() {
    for (int i = 0; i < TOTALOBJECTS; i++) {
        objects[i].active = 0;
    }
}
```



```
void addObject() {
    for (int i = 0; i < TOTALOBJECTS; i++) {
        if (!objects[i].active) {
            objects[i].active = 1;
            break;
        }
    }
}
```



```
void drawObjects() {
    for (int i = 0; i < TOTALOBJECTS; i++) {
        if (objects[i].active) {
            // draws the object
        }
    }
}
```

How do we do this in C?

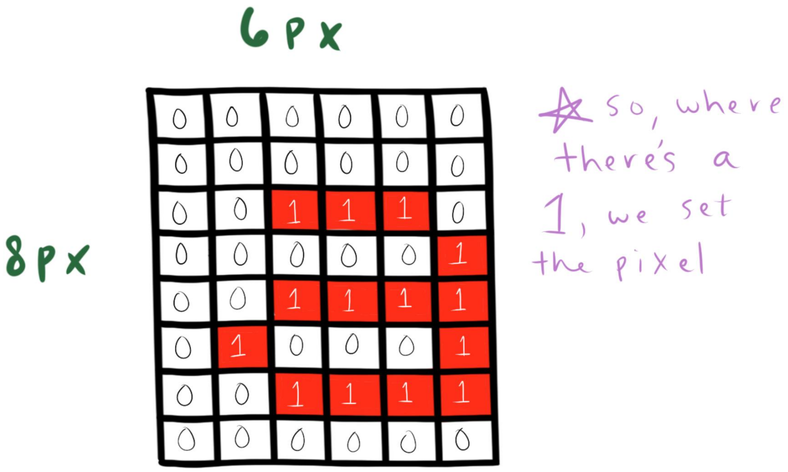
1. First off, we need a struct for the object we are pooling!
    a. This makes it easier to store all of our data for an entity type!
2. Then we add an active parameter to this struct!
3. We then need an initialize function, that initializes all of our objects as inactive, along with its other important starting variables
4. Most importantly, in our draw function, we check if an object is active when drawing, and only draw objects if the active parameter is true
    a. Make sure to use pointers to point at each object, as structs are inefficient to pass around without using pointers
5. Now we can easily set new objects as active as needed, and then when they are destroyed, set them as inactive so they are hidden on the next draw!
    a. Objects will need to be reset based on context for the next redraw as well! (consider a reset function for the object that can be easily called)

Text in Mode 3

- A font is just a bunch of pixels arranged in a specific order to look like different alphanumeric characters.

- We'll provide you with a font.c file in lab so you don't have to write your own font!

'a' in fontdata_6x8[12288]

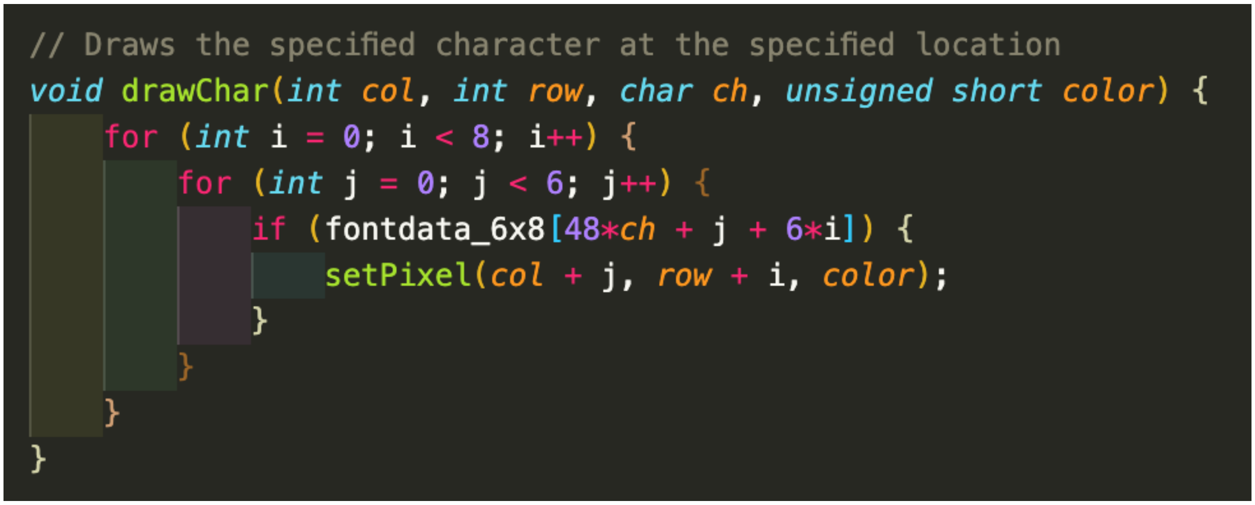


0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	1	1	0
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	0	1	1	1	1
0	0	0	0	0	0

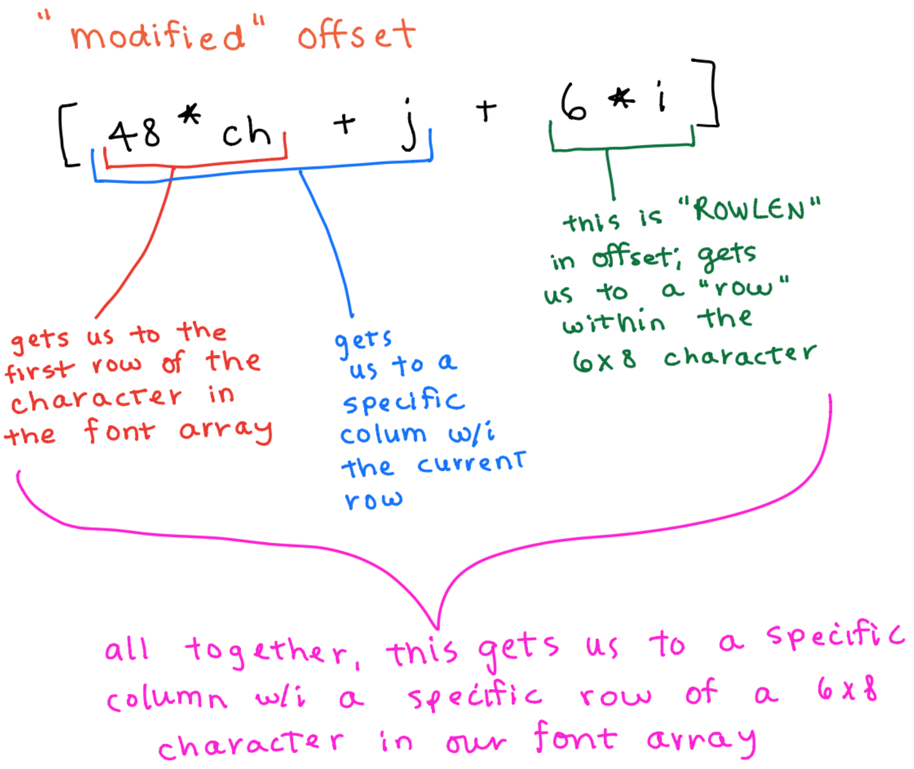
☆ so, where there's a 1, we set the pixel

Steps to draw a char:

1. Find the start of the character in the font array
    a. Do this by multiplying 48 * the ASCII value of the character you want to draw
2. For each "row" in the font array, ensure you're in the appropriate "column"
    a. The font array is actually a 1D array, so it doesn't understand the 2D concept of col and row. Thus, we can use a (modified) OFFSET like we do with the videoBuffer to get to the appropriate place in the array
3. For each "column" in a "row," setPixel if it's a 1



```
// Draws the specified character at the specified location
void drawChar(int col, int row, char ch, unsigned short color) {
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 6; j++) {
      if (fontdata_6x8[48*ch + j + 6*i]) {
        setPixel(col + j, row + i, color);
      }
    }
  }
}
```



"modified" offset
[48 * ch + j + 6 * i]
gets us to the first row of the character in the font array
gets us to a specific colum w/i the current row
this is "ROWLEN" in offset; gets us to a "row" within the 6x8 character
all together, this gets us to a specific column w/i a specific row of a 6x8 character in our font array

How does this work? CHAR's are inherently stored as integers!

C uses char type to store characters and letters. However, the char type is integer type because underneath, C stores integer numbers instead of characters, using ASCII. In C, char values are stored in **one byte in memory**.

All you need to know: our font.c file maps directly to character values based on its corresponding ASCII integer value!

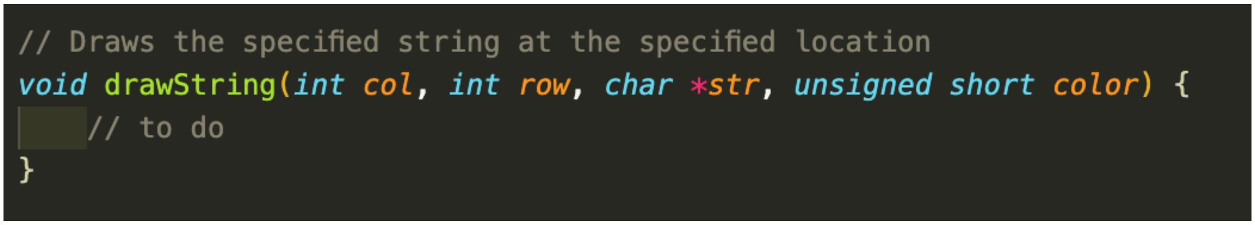
More reading: https://www.cs.swarthmore.edu/~newhall/unixhelp/C_chars.html
Tonc: https://www.coranac.com/tonc/text/text.htm

Drawing a String (in mode 3)!

1. Iterate through array of characters (i.e. your "string")
   a. Do this by using a while loop and check while you HAVEN'T reached '\0'
2. Call drawChar for each character
3. Increase the column you're passing into drawChar by 6 so that you don't draw your characters directly on top on each other

Note, step 2 and 3 should be done in your while loop!

Method signature: (you will complete this for your lab 4!)



```
// Draws the specified string at the specified location
void drawString(int col, int row, char *str, unsigned short color) {
    // to do
}
```