VIDEO 15.4 MAKING YOUR OWN IMAGES

C15 4 Creating new images

4:45 / 4:45 **1.0x**

PROFESSOR JONATHAN VALVANO: Next, let's show you how to make your own images.

But this is an optional step that you can do if you want.

So this is the starter project.

And I'm going to look inside the files.

And I'm going to edit one of the Space Invader art files.

So I'll find an enemy and we'll edit it.

So I'm going to open this enemy here.

And I'm going to open him in Paint, your favorite drawing program.

And any application that can edit BMP files can work.

Now, as you can see, it's pretty small.

So let's make him bigger.

So in this mode, we see that a white is going

to show up as a pixel on the screen.

And a black will show up as a blank.

So I'm going to change this guy to make him a very scary guy,

like moving his insides.

And then let's make him a little more

meaty here and have great big--

and have great big ears.

Because ears are scary.

And we'll give him more fluff around the outside.

Remember, I got to leave two pixels all around.

So we'll make a really scary.

So now I have a new image that I want to create.

Let's Save As.

And this step is important for you to make sure you save it as a 16-color

.BMP.

And so we're going to call him small enemy new.

Give him whatever name you want.

All right, so I've just created a BMP file, a new BMP file.

And so now what I'm going to do is, I'm going

to convert that BMP file into a-- I'm going

to convert that BMP file into text file.

So I'm going to use this executable, which

will take a BMP file and create a text file.

So I got to remember the name of the file I just created,

which was small enemy new.

And into this window, I'm going to write the name

of this file I just created small enemy new.

Hit enter.

And this created a text file.

So I'll find where it went, small enemy new text.

So I will open up the one I just created in a text editor.

You could see this is now C code.

I will copy it.

Now I'm going to go over to my application.

And I'm going to paste the bad boy right in there.

And now I have a new image that I can create.

That I can display.

So let's draw him on the screen, show you what he looks like.

So right here, I'm going to copy, paste, take

his name-- he's called small enemy new.

Where would you like to put him?

Let's put them in the middle.

Halfway across is 48.

Halfway down is 24.

Again, this will put him at the middle at location x equals 48, y equals 24.

So we will build, download, debug.

So again on the screen we see nothing, left over from last time.

And so we will step over, step over, step over.

This will clear the RAM screen.

This will draw the new enemy.

And then when I hit the display buffer, we

can see our new enemy is drawn on the

Help

In the game industry an entity that moves around the screen is called a **sprite**. You will find lots of sprites in the Lab15Files directory of the starter project. You can create additional sprites as needed using a drawing program like Paint. Most students will be able to complete the project using only the existing sprites in the starter package. Because of the way pixels are packed onto the screen, we will limit the placing of sprites to even addresses along the x-axis. Sprites can be placed at any position along the y-axis. Having a 2-pixel black border on the left and right of the image will simplify moving the sprite 2 pixels to the left and right without needing to erase it. Similarly having a 1-pixel black border on the top and bottom of the image will simplify moving the sprite 1 pixel up or down without needing to erase it. You can create your own sprites using Paint by saving the images as 16-color BMP images. Figure 15.6 is an example BMP image. Because of the black border, this image can be moved left/right 2 pixels, or up/down 1 pixel. Use the **BmpConvert.exe** program to convert the BMP image into a two-dimensional array that can be displayed on the LCD using the function **`Nokia5110_PrintBMP()`**. To build an interactive game, you will need to write programs for drawing and animating your sprites.

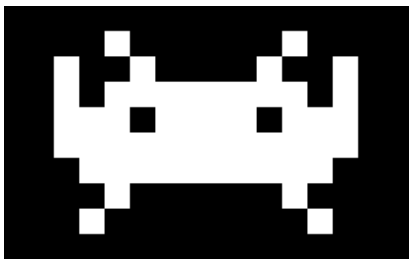

Figure 15.6. Example BMP file. Each is 16-color, 16 pixels wide by 10 pixels high.

Program 15.2 shows an example BMP file in C program format. There are 0x76 bytes of header data. All locations

0x12-0x15 is the width in little endian format. In this case (shown in blue) the width for this sprite is 16 pixels. At locations 0x16-0x19 is the height also in little endian format.

```
const unsigned char Enemy10Point1[] = {
0x42,0x4D,0xC6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x76,0x00,0x00,0x00,0x28,0x00,
0x00,0x00,
0x10,0x00,0x00,0x00,  // width is 16 pixels
0x0A,0x00,0x00,0x00,  // height is 16 pixels
0x01,0x00,0x04,0x00,0x00,0x00,
0x00,0x00,0x50,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x00,0x00,0x80,
0x00,0x00,0x00,0x80,0x80,0x00,0x80,0x00,0x00,0x00,0x80,0x00,0x80,0x00,0x80,0x80,
0x00,0x00,0x80,0x80,0x80,0x00,0xC0,0xC0,0xC0,0x00,0x00,0x00,0xFF,0x00,0x00,0xFF,
0x00,0x00,0x00,0xFF,0xFF,0x00,0xFF,0x00,0x00,0x00,0xFF,0x00,0xFF,0x00,0xFF,0xFF,
0x00,0x00,0xFF,0xFF,0xFF,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // bottom row
0x00,0x0F,0x00,0x00,0x00,0x00,0xF0,0x00,
0x00,0x00,0xF0,0x00,0x00,0x0F,0x00,0x00,
0x00,0x0F,0xFF,0xFF,0xFF,0xFF,0xF0,0x00,
0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,
0x00,0xFF,0xF0,0xFF,0xFF,0x0F,0xFF,0x00,
0x00,0xF0,0xFF,0xFF,0xFF,0xFF,0x0F,0x00,
0x00,0xF0,0x0F,0x00,0x00,0xF0,0x0F,0x00,
0x00,0x00,0xF0,0x00,0x00,0x0F,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // top row
0xFF};
```

Program 15.2. Example BMP file written as a C constant allocated in ROM.

In this case (shown in red) the height for this sprite is 10 pixels. The **Nokia5110_PrintBMP()** function restricts the width to an even number. This function also assumes the entire image will fit onto the screen, and not stick off the side, the top,

Help

or the bottom. There is other information in the header, but it is ignored. As mentioned earlier, you must save the BMP as a 16-color image. These sixteen colors will map to the on/off format of the LCD pixels. The 4-bit color 0xF is on and the color 0x0 is off or black. The function takes a threshold parameter to decide whether the colors 1 to 14 will be on or off. Starting in position 0x76, the image data is stored as 4-bit color pixels, with two pixels packed into each byte. Program 15.1 shows the purple data with 16 pixels per line in row-major. If the width of your image is not a multiple of 16 pixels, the BMP format will pad extra bytes into each row so the number of bytes per row is always divisible by 8. In this case, no padding is needed. The **`Nokia5110_PrintBMP()`** function will automatically ignore the padding.

Figure 15.7 shows the data portion of the BMP file as one digit hex with 0's replaced with dots. The 2-D image is stored in row-major format. Notice in Program 15.2 the image is stored up-side down. When plotting it on the screen the **`Nokia5110_PrintBMP()`** function will reverse it so it is seen right-side up.

```
................
...F........F...
....F......F....
...FFFFFFFFFF...
..FFFFFFFFFFFF..
..FFF.FFFF.FFF..
..F.FFFFFFFF.F..
..F..F....F..F..
....F......F....
................
```

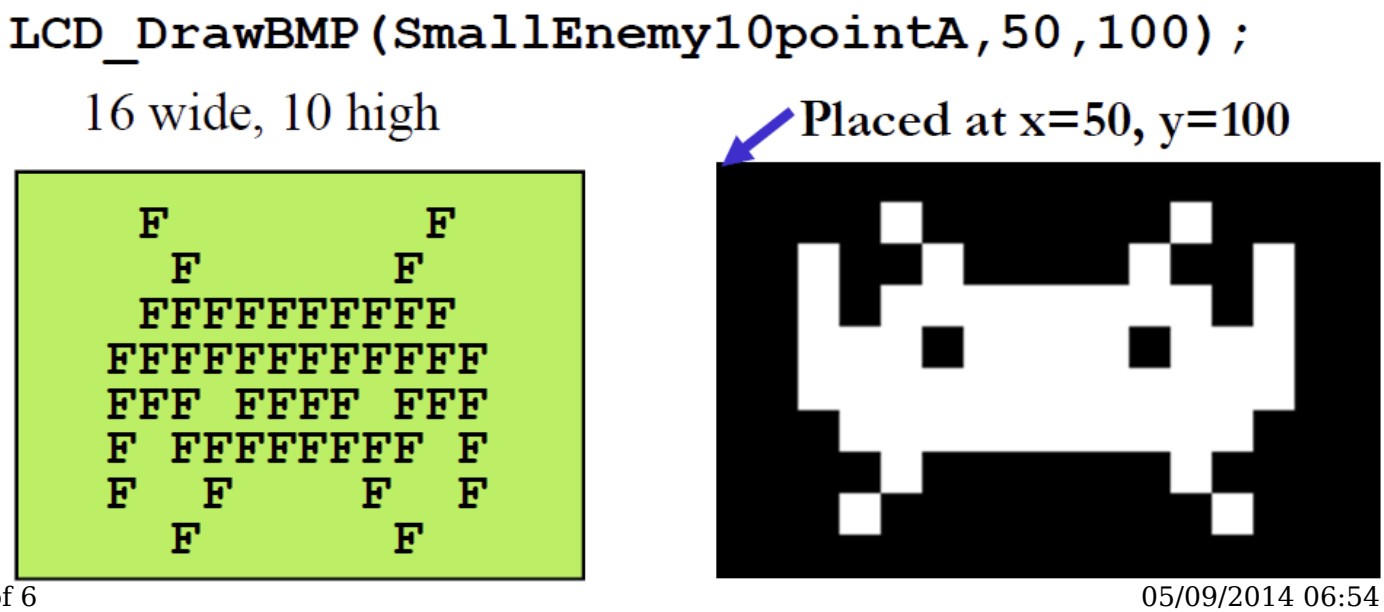

Figure 15.7. The raw data from BMP file to illustrate how the image is stored (0s replaced with dots). Notice the image in memory is upsided down. The Paint application automatically flipped the image when it saved the BMP file, and the LCD_DrawBMP function will flip it back when it puts it on the screen.

CHECKPOINT 15.6

Working with BMP files on our game will be very convenient, because there is a simple translation between drawing the image in Paint, and showing the image on the Nokia 5110. It will also be simple to upgrade to a color display, because we are using a standard graphics format. However there is cost of storing BMP files in C. The ememy sprite, **Enemy10Point1**, described in this section requires 200 bytes of ROM storage to hold. Not counting the width and height, how much actual Nokia data is in **Enemy10Point1**?

Hide Answer

The sprite is 16 by 10 pixels, which is 160 bits of data. 160 bits is 20 bytes, so we are wasting about 90%.

Help

About (https://www.edx.org/about-us) Jobs (https://www.edx.org/jobs)
Press (https://www.edx.org/press) FAQ (https://www.edx.org/student-faq)
Contact (https://www.edx.org/contact)

EdX is a non-profit created by founding partners Harvard and MIT whose mission is to bring the best of higher education to students of all ages anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.

(http://www.meetup.com/edX-Global-Community/)
(http://www.facebook.com/EdxOnline)
(https://twitter.com/edXOnline)
(https://plus.google.com/108235383044095082735/posts)
(http://youtube.com/user/edxonline)

© 2014 edX, some rights reserved.

Terms of Service and Honor Code -
Privacy Policy (https://www.edx.org/edx-privacy-policy)