FlxBitmapFont

A Bitmap Font class for Flixel 2

Version 1.0 - 20th May 2010

by Richard Davey , Photon Storm

http://code.google.com/p/flxbitmapfont/
http://www.photonstorm.com

What is this?

`FlxBitmapFont` allows you to use graphical font sets in your Flixel games. Before True Type fonts became the norm back on the Atari ST / Commodore Amiga they would draw font sets by hand. These often elaborate designs worked their way into the demos and games of the era.

An example of a free bitmap font pack from SpicyPixel.net

Although production of such fonts has declined there are still hundreds of great sets out there, and with the resurgence of "retro" games people are still creating them. If you're trying to re-create a visual game look from the 90s then this is the way to go! Hopefully FlxBitmapFont will make it easy for you.

Set-up

`FlxBitmapFont` was created so that you can drop it into an already existing Flixel v2+ project.

Copy the `FlxBitmapFont.as` file into the `org/flixel` folder.

You are now ready to use it!

Example Code

Internally `FlxBitmapFont` extends an `FlxSprite`. This means that anything you can do to a sprite, you can do to a `FlxBitmapFont`. That includes performing collision detection against it, bouncing it, exploding it or anything else you care to think of. You can add them to FlxGroups for group based collision if you so wish.

Primarily you will probably want to use it for stats display such as a score or lives counter. But it's always handy to know you can throw it around your game just like a normal sprite, and even have characters jump off it.

Include in the download / svn are several fonts. For this example we'll use the "Blue Pink" font:

Embed the font as you would any other external asset:

```
[Embed(source = '../fonts/bluepink_font.png')] private var
bluepinkFont:Class;
```

Let's store the font in a local variable:

```
private var fb:FlxBitmapFont;
```

And then in our FlxStates create function we'll initialise it:

```
fb = new FlxBitmapFont(bluepinkFont, 32, 32, FlxBitmapFont.TEXT_SET2, 10);
```

These parameters are the most complex part of FlxBitmapFont, and are covered in detail in the next section.

To set the text that the font displays you simply use the text method:

```
fb.text = "bitmap fonts rock";
```

And add it to the Flixel display like you would any other sprite:

```
add(fb);
```

Two example source files are included. Demo 1 simply demonstrates putting lots of different fonts up at once, and Demo 2 shows you how you could move a font around with the arrow keys, and update its text value in real-time.

Constructor parameters

The parameters passed when you create an FlxBitmapFont are the most complex part of using it, but they are equally very powerful. Here are the parameters explained in full:

```
FlxBitmapFont(font:Class, width:uint, height:uint, chars:String,
charsPerRow:uint, xSpacing:uint = 0, ySpacing:uint = 0, xOffset:uint = 0,
yOffset:uint = 0)
```

Font	This is the name given to your Class in the embed statement
Width	The width of each character in the font set
Height	The height of each character in the font set
Chars	The characters in the order in which they appear in the font set. This is explained in detail below.
charsPerRow	How many characters are there per row in the font set?
xSpacing ySpacing	If the characters have been drawn with some horizontal padding between them, specify the amount here. Equally if the characters have extra spacing vertically between each row, use ySpacing to set the amount

xOffset yOffset	If the font set doesn't start at 0,0 in the graphic file, you can specify the X and Y offset that it should start grabbing from

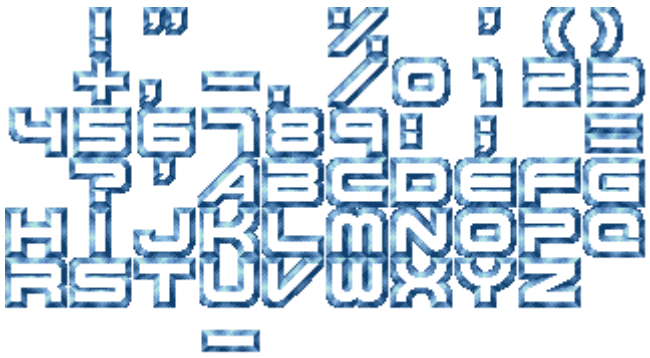
Character Order

You'll likely find that each font set is different in some way.

One of the biggest differences is the order in which the characters were drawn.

For example in the font above the characters go: A,B, C, D, E,F and so on until it hits the punctuation and special graphics at the end.

But in this font:



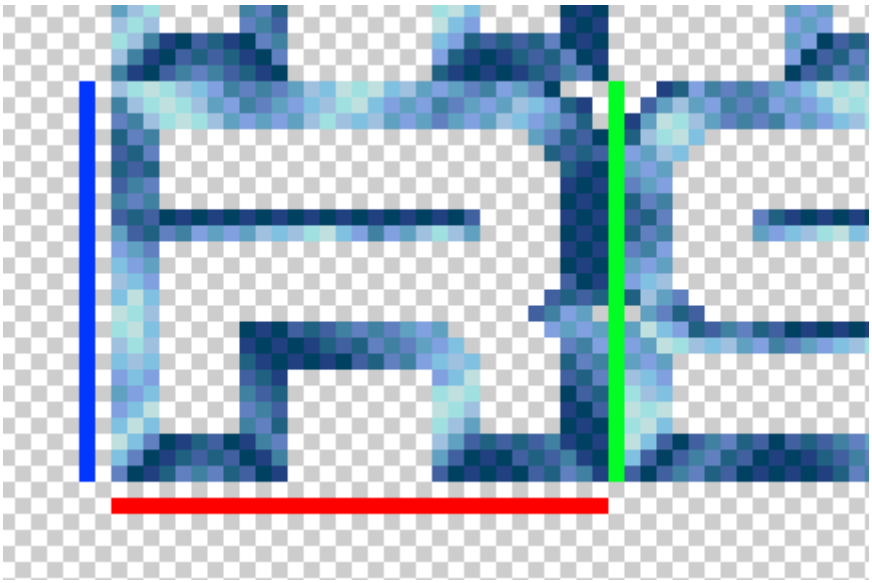
You can see that it starts with a space, followed by !"%'() etc. The variety between these character orders is vast. `FlxBitmapFont` includes several helper constants for you called TEXT_SET1 to TEXT_SET11. These are strings containing character sequences. For example TEXT_SET2 is a perfect match to the font above:

TEXT_SET2 = " !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ"

If none of the provided text sets match your font, you can type it in as a string. Just type it in the exact same order that it appears in the graphic. If the font contains a character you don't want to support just use a space. Equally if the font contains empty spaces, again use a space character.

You are also likely to find that most fonts only contain upper-case characters. By default `FlxBitmapFont` will convert text given to it (via .text or setText()) to upper-case to compensate for this. If you wish to access lower-case characters, or extended characters where lower-case characters form special graphics, then you need to tell `FlxBitmapFont` to not use autoUpperCase.

Working out the width/height and offset parameters is the trickiest part of using bitmap fonts that you didn't create yourself. The following guide should help. Here we have the classic Knighthawks font.



Inspecting the characters in Photoshop reveals that each character is 31x25 pixels in size (the red and blue lines). There also appears to be a 1 pixel gap between each character (highlighted in green), so we pass this as the xSpacing parameter value. There is no such vertical spacing.

The font has been drawn starting at exactly 0,0 so we don't need to include any custom offsets.

You will probably find that a lot of trial and error will be needed at first! But once you've got the parameters fixed use from that point on becomes much easier.

Multi Line Text

`FlxBitmapFont` isn't limited to producing just single lines of text, it can also handle multi-lines. The quickest way to do this is to use the setText() method. setText takes 6 parameters:

Content	A string containing the text you wish to display.
multiLines	A Boolean value indicating if you want to use multi lines or not. If set to true use \n in the Content for a new line. If false only the first line is drawn, no matter how many are passed.
characterSpacing	This parameter affects the display of the font. If you would like to add a 2px space between each character specify 2 here. Default is 0.
lineSpacing	When using multi lines you may wish to include some padding between lines. Use this value to set the amount. Default is 0.
lineAlignment	There are 3 parameters supported which are provided via the consts ALIGN_LEFT (default), ALIGN_RIGHT and ALIGN_CENTER. Each line of text will use this alignment.
allowLowerCase	A Boolean value which tells FlxBitmapFont to either support lower-case characters, or convert your String into upper-case for you. The default is false, which means that for a font set where all the characters have been drawn as capitals passing in text of "atari" would still render "ATARI" correctly.

You could set each of these values on their own. For example:

```
fb.setText("PRESS CURSORS\nAND NUMBER KEYS", true, 0, 4,
FlxBitmapFont.ALIGN_CENTER, true);
```

Is exactly the same as doing:

```
fb.multiLines = true;
fb.customSpacingY = 4;
fb.align = FlxBitmapFont.ALIGN_CENTER;
fb.autoUpperCase = false;
fb.text = "PRESS CURSORS\nAND NUMBER KEYS ";
```

You can see why setText() is a lot more compact.

However if you wish to use the longer method then be sure to call fb.text as the final command, as it will only use settings that have been given up until that point.

Performance Considerations

`FlxBitmapFont` is pretty fast, but there are some important things to remember:

Every time you change the text, the internal pixels of the sprite are modified. This updates the width/height values and other internal vars. While this process is fast, you shouldn't over-do it, or do it unnecessarily.

If you are displaying a score, *don't* have:

```
fb.text = FlxG.score.toString();
```

... in your **update** loop. Instead only change the text when the score actually needs to change.

Thanks!

I hope you enjoy using `FlxBitmapFont`. I certainly have, and it's proved highly useful for a couple of my games so far. Feel free to leave feedback in the Flixel forum, or on my blog at http://www.photonstorm.com

Richard Davey

v1.0 - 21st May 2010

Useful Links

Back in 1998 Daniel Guldkrans set-up the "Graphical fonts" site, which archives hundreds of fonts taken from demos, intros and games of the 16-bit era. Well worth a visit:

http://cgi.algonet.se/htbin/cgiwrap?user=guld1&script=fonts.pl

Spicy Pixel released a great royalty free bitmap font set:

http://www.spicypixel.net/2008/01/16/fontpack-royalty-free-bitmap-fonts/

Bitmap Font Generator

This program will let you generate a bitmap font set from a True Type font. You could then spice it up in Photoshop or similar to give it that "retro" look:

http://www.angelcode.com/products/bmfont/

A similar application by Codehead is available here:

http://www.codehead.co.uk/cbfg/

And finally Fancy Bitmap Font Generator by Iron Star Media is worth a spin:

http://www.ironstarmedia.co.uk/blog/fancy-bitmap-font-generator/