Using Pixel Fonts in Flash MX

by M.D. Rowland

As a flash designer or developer, you're no doubt familiar with using embedded fonts with anti-alias rendering and device fonts with client-side non-anti-alias rendering. In addition to these two methods of dealing with type rendering, there is a third method that many professional designers are using to create sleek and sophisticated type with Flash.

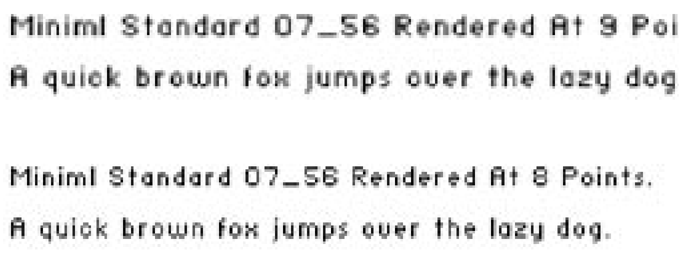
Pixel Fonts are fonts that are specifically designed to be used with Flash. Because Flash anti-aliases all embedded font data, type set at smaller sizes tends to be hard to read due to the blurriness of the anti-aliased edges. The alternative to blurry type is orthochromatic type using one of only a handful of common fonts like Verdana or Times New Roman. Flash designers want access to other fonts besides the universal web fonts, but they also want their type to be legible at smaller sizes. One solution to this problem is pixel-based fonts.

Pixel fonts are designed to be used at only one size. The fonts are drawn entirely of squares and have no curves. When they are set inside of Flash at the specific size they were at which they were designed, the anti-alias that Flash applies to the letter forms is invisible because the letters are made entirely of black square cells, or "pixels." With pixel-based fonts, Flash designers have access to a wider array of typefaces, and they can use these faces with confidence that their type will be legible at small sizes.

To use Pixel fonts, place the font files in your Fonts folder located in `System Folder/Fonts` for Mac OS 9 Users, or one of your `Library/Fonts` folders if you're a Mac OS X user. Next, launch Flash MX. There are a couple of important rules you must learn about pixel fonts:

- **You must always use the font at the size it was created**
  For many pixel-based fonts, this is 8 points. Using the font at smaller or larger sizes results in Flash having to re-render the letter forms and an anti-alias will be applied. (See **Figures 1** and **Figure 2**)

Figure 1: Flash Rendering of Pixel-Based Font, Comparison of Sizes, 100%



Miniml Standard 07_56 Rendered At 9 Poi

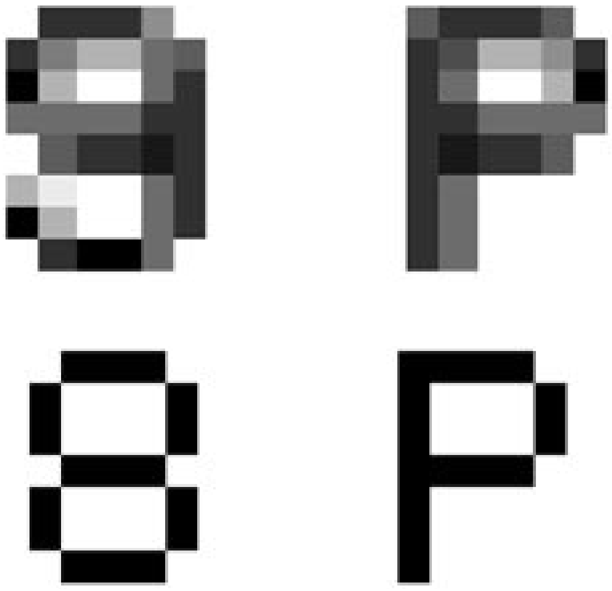
A quick brown fox jumps over the lazy dog

Miniml Standard 07_56 Rendered At 8 Points.

A quick brown fox jumps over the lazy dog.

When viewed at 100% (Figure 1) the rendering of Standard 07_56, a pixel-based font from foundry miniml.com is rendered with an anti-alias when set at 9 points, but is rendered entirely orthochromatic when properly set at 8 points.

Figure 2: Flash Rendering of Pixel-Based Font, Comparison of Sizes, 900%

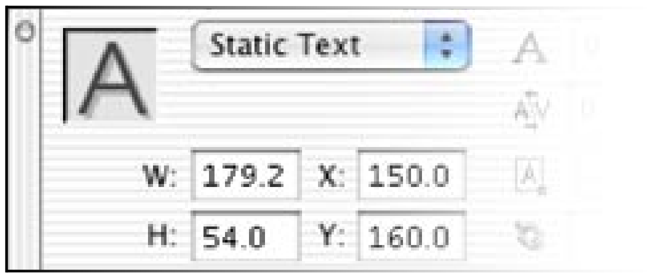


On closer inspection at 900%, the disparity between an anti-aliased letterform and an orthochromatic letterform rendered in Flash becomes much more apparent. It is crucial to set pixel-based fonts at their intended sizes to avoid an anti-alias.

- **You must leave "use device fonts" unchecked**
  In this situation, you want to ensure that Flash embeds font data because the odds of client-side machines having your particular pixel-based font installed on their machine is slim. We usually try to avoid anti-aliasing at smaller sizes but if used correctly, the anti-alias Flash applies to the letterforms will be invisible.

- **You must pay careful attention to pixel alignment**
  In order to ensure that a pixel-based font does not anti-alias when rendered to a shockwave movie, you must use one of three alignment techniques. Flash must use fractional pixel coordinates in order to facilitate frame interpolation and automation. This allows for much smoother rendering of animations. But it also means that objects that are not animating can be placed on fractional pixel coordinates, causing thier edges to anti-alias when not necessary. This is especially problematic when setting type. To make matters worse, when symbol nesting occurs, these fractional coordinates can mathematically compound. By using one

Figure 3: Pixel Alignment



In this figure you can see how whole pixel values have been supplied for the X: and Y: fields inside of the property inspector's text properties. Using whole pixels helps ensure proper rendering of pixel fonts. In the event that whole pixel alignment does not eliminate the anti-alias rendering, use the difference or identical offset methods discussed below.

of the following alignment techniques, you can overcome fractional pixel coordinate placement and ensure that your pixel-based font does not anti-alias.

Technique 1: Whole Pixel Alignment
Whole pixel alignment is by far the most successful of the three techinques for pixel alignment of pixel-based fonts. Over and above that, it is a good practice for every object that you place on the stage in your Flash projects. It helps ensure accurate rendering of vector objects when your project is exported to a shockwave movie.

With the text tool active, at the bottom left hand corner of the property inspector you can align your text box to whole pixels. This helps ensure that Flash will render your pixel-based fonts properly. See **Figure 3** for an example of aligning text to whole pixels.

Technique 2: Difference Pixel Alignment
When whole pixel alignment is not successful, use the differnce method. Take the decimal value of the width of your text box and subtract it from 10. Take this difference and use it as the decimal value of the X coordinate. For example, if your text box has a width of 75.6, make the x coordinate's decimal value 4. Repeat this procedure for the height/Y coordinate.

Technique 3: Identical Offset Pixel Alignment
In rare circumstances when whole or difference alignment are not successful, try the identical offset method. Take the decimal value of the width of the text box and use it as the decimal value for the X coordinate. For example, if your text box has a width of 75.6, make the X coordinate's decimal value 6. Repeat this procedure for the height/Y coordinate.

- **You can not apply bold or italic attributes**
  Turning on the bold and/or italic attributes changes the way Flash renders font data and will most likely result in an anti-alias. If you need to create bold and italic letter forms, choose a separate bold or italic pixel-based font.

- **Always read the instructions**
  There are no universal rules concerning pixel fonts. Depending on the creator, there may be special instructions for specific fonts to ensure proper rendering. Always visit the web site of the foundry that created your font(s) or read the accompanying documentation.

Finding pixel-based fonts is fairly easy. The pioneer in this field is an online foundry called miniml.com (http://www.miniml.com). Miniml used to offer several families of pixel-based typefaces for free, but after their fonts became popular they switched to a fee-based subscription service. However, there are a few fonts still available for free download at their web site.

Another major player in the field is FFF, FontsForFlash.com (http://www.fontsforflash.com). FFF has a variety of free and commercial fonts available from their web site.

There are dozens and dozens of other private foundries that can be located easily with internet search engines using the keywords "pixel fonts," or "flash fonts." Many have free, royalty-free fonts for download on their web sites. An afternoon of good web browsing will likely turn up dozens of good-looking pixel-based fonts for use in your Flash projects. One word of caution: you get what you pay for. With a few rare exceptions, the free fonts are mediocre in quality, and a few might not work at all. For a small investment, you can get a few quality pixel-based fonts from major foundries that will aid you in your designs for a long time to come.

Pixel fonts can be used to add sophistication, legibility, and professionalism to your Flash projects, but they're tough to get right. Only after you've mastered using both device fonts and embedded fonts should you try to tackle pixel-based fonts.

For a research prospective on anti-aliased fonts, see:

Chandler, Scott B. (2002). *Legibility and Comprehension of Onscreen Type.* Unpublished Dissertation.