Picojustification

The attractive arrangement of words on a printed page can be a joy to behold. But these days, a lot of self-publishing looks downright terrible. Caused mostly, I guess, by the lack of attention to detail, a weak appreciation of layout fundamentals, the uncritical use of canned software, and simply not looking close enough.

It was a very sad day when Arizona Highways magazine dropped hanging punctuation because "it was impossible to do with a computer." Well, with any luck at all, you should be seeing lots of hanging punctuation right here. The only thing I can conclude is that Arizona Highways is not using a computer that is as good as my Apple IIe.

Hanging punctuation is trivial to provide. It is usually inexcusable not to do so.

At any rate, text justification is the process of arranging words on a page. Such that they create an intended visual effect that both communicates effectively and is artistically pleasant. While attractively filling an available space.

Let's look at the fundamentals of text justification. Then I'll show you a brand new picojustification technique that might simply and dramatically improve the quality of your final layout work. Yes, it works with a computer.

These are the common forms of text justification...

Although these are your big four, there are lots of other justification possibilities. A free form justification is any variation on a fill justify where the line lengths continually change. You might use this to inset figures or text pulls, or else to define an unusual paragraph shape.

Array justification gets used for connectors or pinouts. This one usually does multiple center justifies.

A menu justification puts down all of the dots to align selections with prices. Your goal is to mix proportionally spaced text and fixed pitch dots. The easiest way to do this is to put down all dots first. Then erase unneeded ones.

A tab justify is used for columns of information. This usually arranges callouts, or small blocks of text in tabular form. The blocks themselves are individually left, center, or right justified. The most common right justify use is for prices or columns of figures.

As a general rule, fill justification is the hardest to get under control. So, let's take a quick look at the...
On the Diablo 630 proportional daisywheels, a two-step process got used. Words were first spaced out to a visual limit, and then an extra 1/120th of a inch of fixed space was added between each character.

Because there are so many of them, modest character spacings can add up to a surprising amount of stretch. And dramatically improve your fill justification. As much as half of your space-gobbling can often get absorbed with tiny values of between-character spacing.

Today’s better-done computer text setting programs use a microjustify to stretch out the spaces at one rate and the character spacing by a second. The spacecharratio is often around 12 to 15 for longer lines, dropping down to 6 or so for tighter margins. A value of 6 means that word spaces widen 6 times faster than character spaces.

The widthshow operator in the PostScript language is particularly adept at fill microjustification.

Regardless of the system you use, your foremost rule is to give your fill justification machinery as little to do as possible.

The best way to hide white space is to not need any of it. Your usual crutch here is hyphenation, done on either a line-by-line basis or else by working with entire paragraphs at once. Hyphenation can be further improved by allowing second and third syllable breaks.

And, of course, by carefully making several post-layout passes to get rid of recurring problem lines.

I personally do not care for hyphenation. I avoid it when and wherever possible. Hyphens reduce your legibility and comprehension. They’ll also make keyword searches and latter day text resetting more difficult. Besides, it is a really fun puzzle to try and set tight text without it. Your bottom line is that most hyphenation is completely unnecessary.

Given some extra time and effort.

My own methods for tightly setting text are rewording and reorganization. The chances are you can still get your same message across by using slightly different words in a somewhat different order. I very strongly feel that tightly set text is as important as the message itself when it comes to smooth communication with your end reader.

Bells and Whistles

There’s lots of tricks you can pull to further improve fill justified paragraphs. Here’s a brief sampler–

Last line stretch– The last line of any paragraph usually gets left, rather than fill justified. Since fill justified lines have extra white in them, your last paragraph line usually ends up slightly darker than the rest. To beat this, add a little extra stretch to your last line.

Smaller caps– Longer numbers or any text you set in ALL CAPITALS just might end up looking "too big." A good workaround is to use a slightly smaller font for upper case. For instance, if your main text is 9.5 points, use 9 points for numbers and caps. As I have done here.

Widows and orphans– It is quite important not to have a single short word dangling on the last line of a paragraph. Or, worse yet, extended to the next column or page. Use rearrangement and rewording to minimize leftovers. Aim for mid-line paragraph endings.

Kerning– Some characters do not line up well. If you print the word AWARD, you’ll want to reduce the space between the slanty parts of the A and the W. Again, like I’ve shown here. Similarly, there are times when you may want to increase spacings that look too crowded. These tricks are called kening. While fancy kern pair tables are sometimes used, all you really need is the ability to throw in or remove an extra horizontal point or two. Vertical kerning can also be useful to center brackets.

Real typography– That ASCII character set only contains 95 printing characters. thus, there can be lots of pressure to substitute a hyphen for an em dash or an en dash. Or to use periods for ellipsis. Or quoteright for quotoleft or acute or grave. If you have the correct symbol, use it!

Hanging punctuation– The amount of “black” in typical punctuation is much lower than most characters. If you do nothing special, "notches" will be seen in each fill justified line that ends in any period or comma or whatever. The solution is called hanging punctuation. In which each line that ends in a lighter symbol gets stretched somewhat, say 40% of the final character’s width.

Half spacing– Paragraph comprehension might often get improved by adding extra vertical white space at the ends. Especially in instruction manuals and tutorials where you want the reader to digest one thought before going on to the next. But a half line spacing may look better here than a whole one. Separately, a traditional printer often used French spacing (or space-and-a-half) between sentences. But this has largely fallen into disuse.

Drop caps– Those Medieval monks always "illuminated" their manuscripts with a giant and fancy first character. Today, the technique is called a drop cap. And remains useful to guide the reader into the start of your message. Drop caps can be automated into the fill justify machinery, or you can simply indent and leave room for a separate art cut or character. One popular and simpler variation is the raised cap, which is nothing but a big first character.

Picojustification

These days, we are not in the least limited to using fixed character widths. So, we now have an incredibly powerful new picojustification tool for dramatically improving your fill justification.

Instead of putting all of your fill justified spaces between characters or words, you instead put a good portion of the excess space inside of all your characters by making selected characters slightly wider.

Surprisingly little character stretch is required to greatly improve any fill justification. At least any that is already reasonably well done. A pixel or two either way at 600 DPI on a 50 character line adds up to a 200 pixel correction range. This can easily sop up a third or more of your excess white for a 50% improvement in justification quality.

Naturally, I very strongly feel that you should be using genuine Adobe PostScript Level II anytime you want to dirty up otherwise clean sheets of paper. Especially if you are at all serious about doing first quality text justification. So, while you certainly could use picojustification with any scheme that lets you change font widths in rather small increments, PostScript works superbly well.

Here is a PostScript runtime utility you can try. It does a picojustify for you...
You can turn the picojustify on and off by using a true picojustify or a false picojustify command. Generally, you might want to use picojustification for your text body, but not for typical figures, artwork, headers or borders. That picothresh value lets you decide how much incremental stretching you are to get, while the picofract should set the percentage of total excess white space to be absorbed by your new picojustification.

Usually, you can place this module at the beginning of your code, right after your eps header. The rule is to insert the module where it will redefine run time awidthshow.

The code as shown should end up compatible with most layout packages. If you have problems, compile your text using that Adobe Acrobat distiller. Note that Acrobat can easily be taught to read most any file in most any format.

If you have problems, just do a print-to-disk to see what you’ve really got. Naturally, this code will not work if you are not fully using awidthshow in the first place. Or are not including all spaces in your strings. Give me a call if you have any problems adapting it to your current setup.

The minor speed penalty of three seconds or so per fancy page can be eliminated by redistilling. Distilled pages should print at the full mechanical printer speed. Your font cache will also fill up around four times faster than usual. But more often than not, you’ll only be using a few fonts for your text body and this should not end up as a serious problem. At least for most users.

You can, of course, speed things up bunches by doing a picojustify in the first place, rather than doing last minute repairs to older code. Your difficulty here depends on how open your older code was.

The concept of "analyze each line on the way out the door” can easily be extended to automatically add character spacing, hanging punctuation, or for many other special improvements. Thus, this is a very general tool.

As to the quality you can expect with picojustification, you are looking at it.

For more help
I’ve got a lot more info on quality text justification up on my Guru’s Lair. Especially on our Acrobat, PostScript, and Book-on-demand library pages.

I’ve also got a GONZO.PS routine that gives you some exceptionally high quality, ultra flexible text justification. This one is totally open and extremely easy to customize. But Gonzo is purposely not WYSIWYG for maximum power, control, and full device independence.

Consulting and InfoPack services are also offered. Give me a helpline call if you need any further help with text layouts, picojustified or otherwise.

Let’s hear from you. There’s some really exciting new possibilities here. ♦