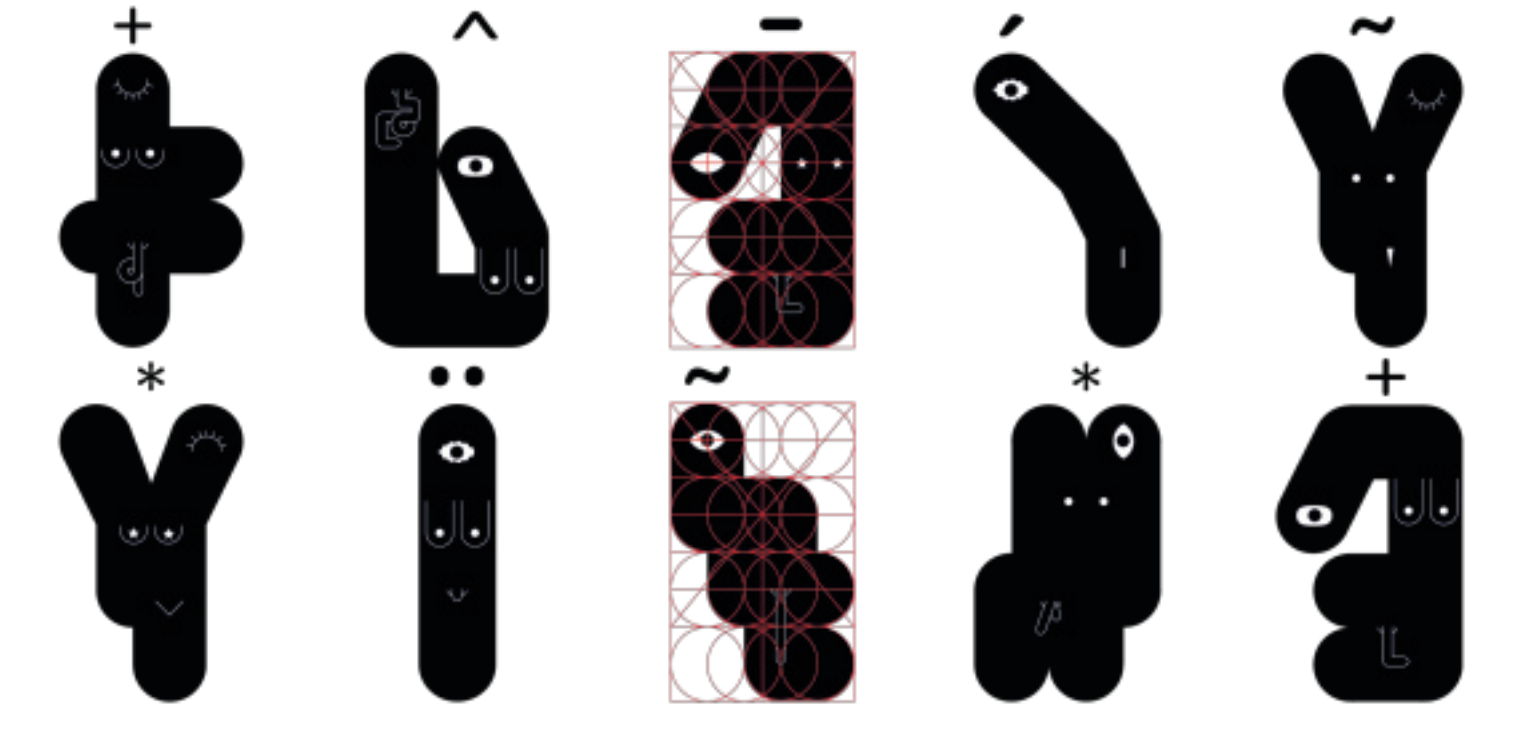


languages for introducing people to programming, due to its clear, readable syntax, which almost resembles the pseudocode used to explain abstract programming concepts. It also hides away (or puts sugar on) much of the complexity behind programming concepts, allowing us to focus on properly wording our orders to the computer.

Nodebox and Shoebot provide a sketchpad for writing small Python scripts. When running them, the program will create graphical output—an image. This instant visual feedback is a big plus for teaching code to creative-minded people, since it allows for swift tinkering and borrowing from the provided example scripts, and was crucial in easing design students into the coder way of thinking.

PRACTICAL EXAMPLE: CHARACTER GENERATOR

One major assignment in this class was to design an identicon generator. Identicons are icons commonly used in blogs, especially inside comment sections, which identify the commenter through a graphical representation of their IP address. This is done by combining different possible parts into one final image. MonsterID and Wavatar provide icons in the shape of quirky monsters, whereas Identicon generates abstract, geometric shapes. The goal of this assignment was to think up the design for an identicon, freely choosing the subject, and create a program that could generate different outputs randomly.

The size constraints of a blog icon are a big limitation, one that wasn't forced on the students in order for them to focus on the more relevant creative questions. Many of the students went through the character-design route, though others attempted more daring approaches, such as cake and bug identicons.

The challenge of this assignment was not the coding itself. Most students were already rather comfortable with using randomness, drawing with code and importing external images. The focus was on creating consistent designs which could work with different compositions and still end up as a complete final result, not giving away the fact that it was generated by a program.

The illustrations running alongside this article are some of the results of this assignment.

Image Credits:
Page 18: Fábio Santos; Edgar Sprecher; Joana Estrela.
Page 21: Sofia Rocha e Silva; Telmo Parreira; Lídia Malho.

Nodebox: http://nodebox.net
Shoebot: http://shoebot.net
Pure Data: http://puredata.info
Drawbot: http://drawbot.com
Context Free: http://contextfreeart.org
Fluxus: http://pawfal.org/fluxus


5
4
3
2
1
7
6

Setting a book with Scribus

Pierre Marchand

I remember my own first time, the first serious one, was a bookletization of a famous, amongst afficionados, little parody by Pierre Louys under the title of *Manuel de civilité pour les petites filles à l'usage des maisons d'éducation*. With its typical late 19th century French style, it was natural to associate it to a didone font. I ended up using the Didot shipped with the Mac OS I owned at this time.

Here is the crux of this story: at this point I hadn't yet read the paper by René Ponot[1] convincingly establishing that it was not a good idea to use ligatures with the Didot typeface. I wanted to use them! But in this instance of the typeface, the ligatures and old numerals were outside the charmap, intended to be accessed only by means of OpenType substitution or glyph index. That, in itself, was an adventure.

This time, though, was also my first time going deeply into font technologies and Scribus code. Along my journey in these fields I came to read Theotiste Lefevre. His amazing *Guide pratique du compositeur d'imprimerie* helped me realize how much, even if still non-trivial, the making of a book has become within everyone's reach with desktop publishing and personal printers. For now, forget Louys and Didot and go for a book in a minute!

The recipe is as follows. First be modest and grab some text fallen into the public domain at gutenberg.org. If you attempt to write your own material, you will definitely not be able to do a book in a minute. Next, run a bit of Perl magic powder onto it like `perl -n -e 's/(\S)\r\n/\1 /ms; print $_;' original.txt > withoutlinebreak.txt` to let Scribus do its work at line breaking. (See below for more detailed instructions.)

Now you can create a new double-sided Scribus document with a bunch of pages and automatic text frames turned on. Import the text into the first text frame. Set the default paragraph style to something that looks like a book, serif typeface at 10 points, justified, etc. Et voilá!

Well, it isn't exactly ready to serve to your friends, but you get enough of the taste of an actual book to open the door and start to work. If you think not, click on the eye at the right bottom of the Scribus window to turn on Preview mode.

While writing these few lines, I'm doing the same as I did years ago and am still amazed by what Scribus brought to us — by what it allows us to do and the opportunity it gives to learn about desktop publishing. We have the opportunity to do publishing work as Scribus exposes its internal representation of graphic objects through an opened source code and file format.

1. Le Didot a-t-il besoin de ligatures ?
Cahiers Gutenberg no. 22 (1995), p. 17-41
http://cahiers.gutenberg.eu.org/cg-bin/article/CG_1995___22_17_0.pdf

EXECUTING COMMANDS IN THE COMMAND LINE

No matter what operating system you're using, you've got a command-line interface at your disposal. If you're of a certain age, you may remember fiddling around a little with MS-DOS. Even if you never did, don't worry about it. The command line is friendlier than you may think.

Now, because we're all designers here, chances are good you're using a Mac. Or, if you're like us, Linux. The tips and commands listed below work just fine for both. If you try them in MS-DOS (under Windows), your computer may explode. We're not quite sure, really.

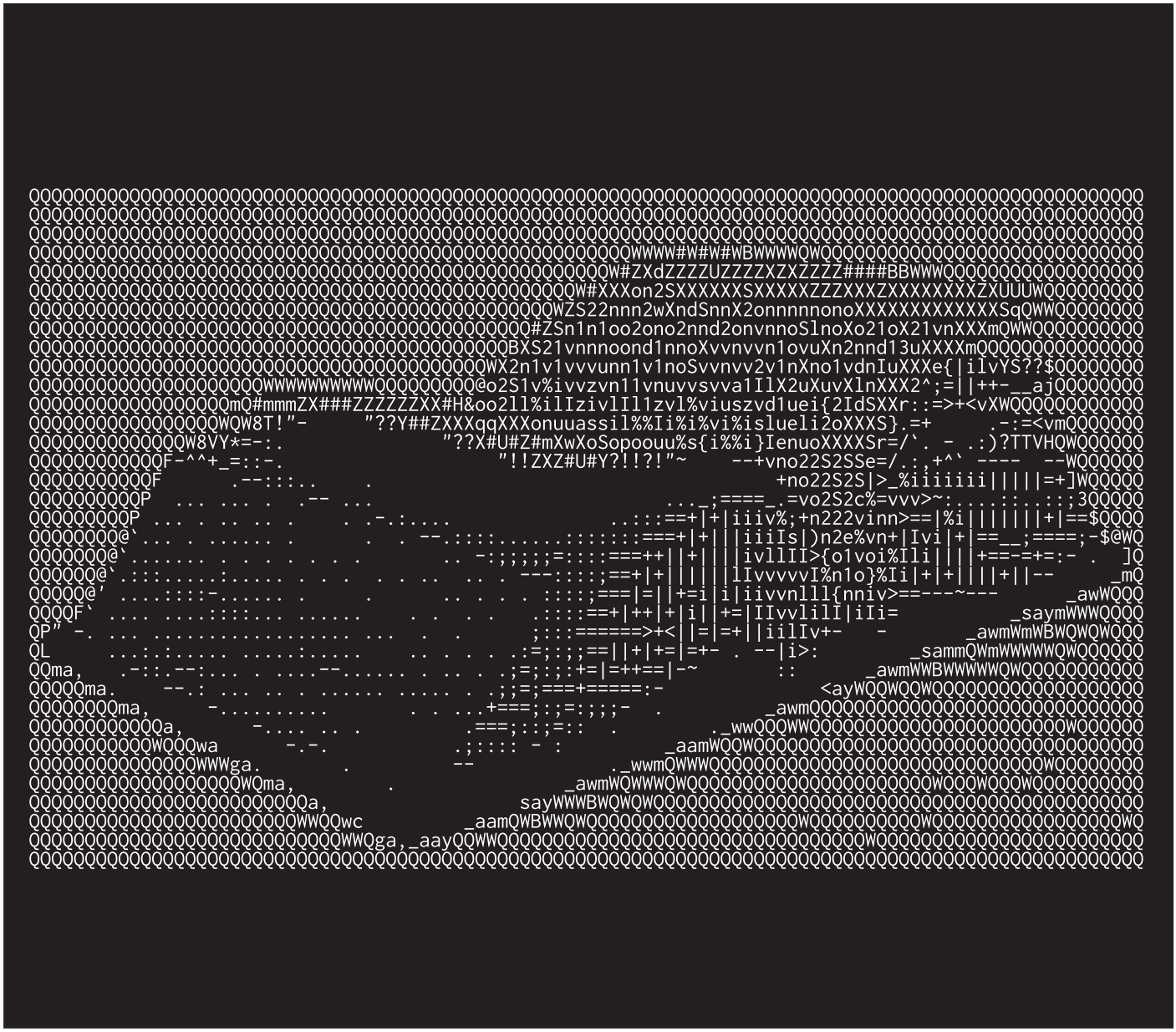
Open a terminal:
On a Mac: Crack open your Applications folder and go to Utilities. There's a program there called Terminal. Open it.

On Linux: Normally under your Accessories or System menu, you'll find something called Terminal. Open it.

Get to the right place:
If you've downloaded a book from Project Gutenberg, hopefully as a plain text file (something ending with .txt), great. If not, go back and do that. But make sure to take note of where you've saved it.

When you opened Terminal, it should have started you up in your home directory. To make it easier to find where you're going, open up your file browser (Finder on a Mac, or on most kind of Linux, just double click on the icon for your home directory). Navigate to where you put your file. Now, take a look at the path leading up to that. For example, if you left it in your Downloads directory, chances are good that it'll only be one directory past home.

Once you have an idea of where you've put your file, go back to the Terminal. To change directories (because that's what you're about to do, unless you've left the file in your home directory), you're going to use the cd command. It allows you to (yes!) change directories. Let's say you've left your file in the Downloads directory.



In your terminal, you'd type "cd Downloads" (without the quotation marks). That would take you to your Downloads directory. If, in the Downloads directory, you happened to have another directory, this one called books, for example, you'd then go "cd books" (note that it's case sensitive).

Looking around:
Now, we're in our fictional home/Downloads/books directory. Let's take a look at what's there. To get a list of the contents of a directory, just type "ls" while you're in the directory you want to look at. It'll turn up a list of all the files and directories contained within that directory. If you've gotten to the right place, ls should show you the book you've downloaded.

Running the script:
Now you can run the script mentioned in the article. Just copy it and paste it into your terminal. Or, if you're reading this in print, type it. Heck, type it in regardless, just for practice!

```
perl -n -e 's/(\S)\r\n/\1 /ms; print
$_;'  original.txt >
withoutlinebreak.txt
```

Of course, you'll want to change "original.txt" to reflect the actual file name of the book you downloaded. Then, hit Enter. If all goes well, the next time you do an ls, you'll find a new file, called "withoutlinebreak.txt" which will be the book you downloaded, without linebreaks and ready to be conveniently typeset in Scribus.

While this may seem like a lot of complicated steps, once you get used to it, you'll find that it's easy, convenient and fast. And it's just the beginning of what you can do with the command line.

—the editors