# A genealogy of several OSP projects unfolds since 2008 around the questions we'll be excited to explore in another way in Seoul :

— During several Dingbat Liberation Fests, OSP insists on questioning from the side what a glyph is, what a letter is, and how we can re-appropriate these precious parts of the culture as our own, even if a legitimate smoothing effect by Unicode codification effort tend to loose the granularity in the heart of each triangle script-language-typography. Dingbat Dictées are recipes for teaching Unicode poetry to students of all ages.
http://ospublish.constantvzw.org/blog/?s=dingbat

— An osp installation system called Nancy, deploying the Dingbats Liberation Fest at the gallery My.Monkey in Nancy, (F) and then at Make-Art Festival in Poitiers, (F).
http://ospublish.constantvzw.org/nancy/

— By developing an experimental software setup for decentralized, collaborative typography, OSP member Pierre Marchand starts to ripple an agglutination of software elements following the many meanders of OSP projects. 'Nancy' is made to support collective font design, also if dispersed over time and geographic location. It is a patchwork of classic Free, Libre and Open Source softwares, stitched together into a networked power-tool. This system is semi-automated process of scanning of cardboard-cut dingbats and inclusion of these shape in a collaborative font. It turn the simplest design into a complicated process involving Fontforge, Subversion, Scribus Python scripter, PoDoFo, glue and good will…

— The Fonzie outgrowth has been developed after several pushes between, among others things, the need for a convenient way to reproduce hand-drawn fonts for translated version of comics, using Opentype features to automatically switch between different versions of each glyph that try to emulate variations of typical hand drawn lettering. Fonzie's progressive modifications follow various different type design needs and shapes multiple processes.
http://ospublish.constantvzw.org/blog/?s=fonzie

— A more straightforward version is used in workshops starting schools in Brussels and London, but also performances and book design.

— A hackerspace in Brussels built a DIY-bookscanner and another evolution of Fonzie nicknamed Funzie added other OCR features to convert physical to digital reading by producing hybrid fonts. The reading process of the machine can be watched, understood, read, rewritten, changed and endlessly executed. This explores the artistic potential of the different elements in the transmission from the physical book to the digital object: the code, the fonts, the training data, the book and the digitally reborn text.

http://video.constantvzw.org/VJ13/ -
http://www.vj13.constantvzw.org/archive/funzie_fonzie/

# Workshop proposal :

*Short version :*
By looking at the inner workings of digital text recognition software know as OCR (Optical Character Recognition), there seems to be a space in which we can put comprehension aside, and where we could be able to observe typography in its shapes, before making out it's specificities, or even it's meaning.

By declutching some of our digital habits, and taking time to understand and retrace processes that OCR uses to look at shapes, step by step, and over time working out which are characters, which are words, and what they correspond to, this software embodies a digital way of learning to read.

We believe there is a lot to take from this, a lot we, as type enthusiasts, can grasp on to. Could we propose a patient manual reverse engineering process of the digitally possible methods to work towards a deeper understanding of our typographic languages and our restitution of the embedable links between content and its forms?

*Long version :*
Fancy reading machines androids from science-fiction fantasies are embodied in our modern lower-profile real world as OCR software package. One of them, the free and open source Tesseract [1] is composed of two parts that we can study, thanks to it's license. There is the engine itself, and the training data for a language [2] partly based on what Tesseract called 'prototypes'. We could compare this 'before to type' (proto-type) to the culture a lecturer progressively gathers from his first lesson going from a novice to a fully grown expert.

By following the limit between the blank surfaces and the dark pixels of the shapes of letters, Tesseract compares its journey with other and previous ones, on images already followed in the past. It starts by learning patterns and specificities of languages, rhythms and irregularities. It goes on to recognize the body of a glyph, then it works out, bit by bit, if this glyph is a letter, form is a word, and eventually it makes out phrases. Like all of us, Tesseract learns typography in this same process, in a completely intertwined way, as sentences, script and eventually, language. [3]

Tesseract follows rules by which it can make decisions. In a basic example from latin script, if the software seems to be recognizing something resembling to iii (three times the letter 'i'), specific rules kick in to suggest that it is most lightly the letter 'm' and not a triple consonant. Grammar and language coming in at a later stage, as it did for us, still following this unusual idea of teaching software to read. [4] The very specificities of typography and how each shape is drawn and could or couldn't be deciphered from another one arrives just after, as in the previous example the potential small parts that protrudes from the i could form the arc of the m in a more convincing way if the font is a serif one than a sans serif one.

This process does become intertwined with the actual context: with time, the system becomes familiar, and extremely efficient with some specificities of a typeface. It's shape, it's overall form and size now mean something. It would have to relearn an entirely new toolkit to be able to read a different typeface. With this, could the relations binding shapes to their meanings be noticed?

At young, naive and early stages of deciphering writing systems, slowly working out the building blocks to a legible language, we wonder how synthetic constructions (like Hangul) compare to agglutinated ones (like Latin). More specifically, how do these methods influence OCR data?

On a more contemporary note, it would be hard to deny how much screens and screen text technologies have influenced typography these days. All languages carry different meanings, different cultures with their characters. These gri(d)tty displays are no favor to typographic heritage, but they have brought on so interesting conundrums. The rendering engine ttf autohint, by example, voluntary distorts vector shapes of glyphs to optimize screen rendering [6]. When the movement to follow the grid become displacement to fit, the boundaries between canvas based, stable and territorial, and flux based, flexible and moving, blurs itself.

In this workshop, we propose to carefully replay some of the processes the OCR system uses to reread typography from the departure point of any new learner, the one we all have known at first and mostly definitively forgotten by now... By patiently observing the various parameters at play when a letter is to be differentiated from another, the thin and variable line of separation between signification and shape, between letter and typography begins to reveal itself. Could the different parts of the letters that compose bare bones of other letters that are recreated in a kind of wild reverse engineered Metafont [5] paradigm, where all of the shapes of the glyphs are defined with geometrical equations?

We wonder how much we can learn from methods borrowed off OCR. By replaying its methods, but basing ourselves on some parameters only, not aiming for full comprehension, but basic knowledge of how our different sets of characters work retracing its first steps only? Would the outcome of this be enough to go on to understanding typographic subtilities, enabling a bridge between specificities in shape and specificities in language?

Finally, if we know organization in Hangul and Latin are different, and that they do work along with similar ideas, could we try to avoid the main caveats of forcing comparisons between each? Instead can we focus on the systems that the OCR-by-human must use to read both for rethinking deeper specificities between the two composition methods, between these two typography, between these languages?

1. http://en.wikipedia.org/wiki/Tesseract_(software)
2. https://code.google.com/p/tesseract-ocr/
3. http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3
4. http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3#The_last_file_%28unicharambigs%29
5. http://en.wikipedia.org/wiki/Metafont
6. http://www.freetype.org/ttfautohint/#samples