John Plaice on Omega (Ω) and Beyond (*an interview*)

Apostolos Syropoulos

66, 28th October Str.
GR-671 00 Xanthi
Greece
E-mail: asyropoulos at yahoo dot com

John Plaice is known for his pioneering work on Ω (Omega), the first project to expand the multilingual capabilities of TEX in the early and mid-1990s. That project is now over, but as John explains in this interview, its heritage is still alive in LuaTEX, but also in XML. John also talks about Cartesian Programming, his new project that one day may bring to life the *Cartesian document*.

Eutypon (Apostolos Syropoulos): *Thanks for accepting the invitation to talk to* Eutypon. *You are known in the TEX community for your Omega project, but before we talk about TEX and Omega, would you mind telling us a few things about you and your background in computer science?*

John Plaice: I was born in Montreal in 1962, as the Canadian province of Quebec was going through tremendous upheaval, with the French-speaking majority openly challenging the English-speaking élite that had been running the province since the Seven Years War (1756–1763), which led to New France being taken over by the British. This process, now called *La Révolution Tranquille* (The Quiet Revolution), pushed the Roman Catholic Church out of politics, led to mass schooling for the French, and ultimately to French becoming the official language of Quebec.

My father was an English teacher and my mother a translator (French to English). When my sister, four years my elder, started French classes with an English-speaker who could barely speak French, my parents decided to put her, my brother and myself in a French-speaking school, because they could see that in the future, it would be impossible to live in Quebec without speaking French. At the time, the publicly funded schools were all confessional, and the French schools were all run by the Catholic Church fundamentalists. As a result, we, non-believing Protestant heretics, were refused access. So we ended going to Cours Chateaubriand, a private Catholic school (the Church accepts heretics if they pay), where the teachers were imported from France so we would be taught with the right accent.

So, at the age of four, I went to French class, where Mademoiselle Le Brun, 130 years old and 1.30 m tall, told the few English-speakers that we could only speak French. I did not speak a word in class for two months, then I started to babble away in French. I think that this decision by my parents to send me to a French school was probably the most important single one that they made for my siblings and myself, and led ultimately to my doing a PhD in France and working later on a system called Omega.

We went to France for two years, then came back to Canada, where I was put in the English school system, because whenever I spoke English, it was with an outrageous French accent. I finished high school, did a year of CEGEP, then headed to the University of Waterloo in southern Ontario to study pure mathematics and computer science.

In my final year, I was skipping logic class with a fellow student, who told me he was rushing around to find references for a scholarship to study in Germany. I decided to do the same for France, and ended up with a scholarship from the French government to study there.

After a long trip and lots of squabbles with French bureaucracy, I met Joseph Sifakis, who became my master's supervisor. In 2008, he won the Turing Award, and I still recollect an experience working under him. He asked me one day to do something, and I told him that it did not work. He insisted, so I did as he asked, and met him again the next day. He told me that what I had produced was shoddy. "Of course!" I replied, "I told you so yesterday." His reply was that if something asked for does not work, then change things and produce something that does work. I still use this as a lesson for my own students.

After my master's, I continued in the same lab under Nicolas Halbwachs, who had just invented, with Paul Caspi, a programming language called LUSTRE (Synchronous Real-Time Lucid), using synchronous dataflow. I wrote the first semantics and the first compiler for this language. In the meantime, it turned out that the people in Aérospatiale who did avionics for the Airbus flight-control software were doing something similar, but without the linguistic elegance of LUSTRE. I left France soon after finishing my PhD. Under the hands of others, LUSTRE became a real success story. There is now a commercial suite using LUSTRE at its very core called Scade, sold by Esterel Technologies, which is used to program software for reactor control and avionics, including the Airbus A-380.

After my stay in France, I returned to Canada, where I spent two years in Victoria, British Columbia, where I invented, with William (Bill) Wadge

(the inventor of Lucid), possible-worlds versioning, a simple way for producing variants of software that assumes that all components of a system share the same variant space. I then moved to Ottawa, Canada's capital, where I worked for two years. I then (1992) moved to Quebec City, to Laval University, where I found that I needed to install all of the software that I was used to using in previous work environments. One of these pieces of software was TEX...

E: *So this is how you got involved with Omega?*

JP: Several of the postgraduate students at Laval came from overseas, so as I was installing TEX and making sure that the software worked for French and English, I decided to see if I could get things to also work for Arabic, Vietnamese and Chinese. I ordered all of the back issues of *TUGboat* and found that the name Yannis Haralambous showed up regularly among the authors of papers. I contacted him, and met up with him on a trip to Tunisia, with stopover in Europe. In two days, we discussed at length many ideas, and came up with the idea of the Omega project.

E: *Today OpenType fonts include kerning information, all sorts of ligatures and alternates and also some OpenType fonts include support for setting very demanding math text. Also, it seems that programs like MS Word, and to a lesser degree OpenOffice.org Writer, can do everything TEX and its derivatives can do. So what do you think: is there a future for TEX & Friends or is it an evolutionary dead end?*

JP: I am currently writing from India, which I first visited in 2002, with the support of the TEX Users Group India and the Mahatma Gandhi International Hindi University. I visited Thiruvananthapuram, Chennai, Hyderabad and New Delhi. I was able to meet software people, academics, and people in the bureaucracy of the Central government. And they all told me of the need for some free software that would properly support the Indian languages. Now, eight years later, I hear exactly the same words. The need is still there, for at least one fifth of the world's population, who still do not, for the most part, read or write science in their mother languages. The problem is that in this country, people who work in computer science or in related areas work and live in English, not in the vernaculars of the masses. And the success of free software is that the authors must use it...

"One of the useful aspects of TEX is its stability."

One might say, but the solutions provided by this company do the trick. And the answer is, even from a technical point of view, only partially. One of the useful aspects of TEX is its stability. The stability is not 100%, it is a lot more stable than most other pieces of software. A company like Springer, with its Lecture Notes series in Mathematics, Computer Science, Artificial Intelligence, etc., strongly advises against the use of tools like MS Word, because the results are not replicable. With TEX and similar pieces of software, I can run `make` on

every single article I have written, right back to my 1988 PhD thesis, and a few minutes later, I have virtually identical copies to the original articles; the look is even better on the screen than with the original, since the fonts are no longer bitmap.

The survival of a piece of software that implements TEX is, I think, ensured for a long time, because of the way it is used by mathematicians worldwide. Whether this future piece of software is called TEX is another.

More generally, I think that there is a pressing need for having some tool that has all of the power of TEX, all of the flexibility of fancier commercial tools, and a level of stability that goes way beyond TeX, taking into account all of the tools in the supply chain used to produce a given document. Which leads to my current research, what I call Cartesian Programming...

E: *Tell us about the Omega project, how you see it, and where it should go.*

JP: The original goals of the Omega project were to support all of the world's languages and scripts, in all of the different ways in which these might be typed in and printed out. And then we set out to do this, without drawing up proper specifications. As a result, I went off to work on what was important, and Yannis Haralambous went off to work on what was important... Without clearly specified goals or objectives, these became more and more ambitious, thereby making success a difficult task.

Given that Yannis had already drawn a number of quality fonts for a number of scripts, a natural division of labor took place. I worked on necessary modifications to the TEX code base, while he worked on fonts. This division was unfortunate, as it ultimately held back the project, since the developments of PostScript, PDF and of the various font technologies all should have interplayed with the layout engine. Not thinking about the likely necessary changes meant that I limited myself to making mundane changes, instead of more far-reaching ones, as Jonathan Kew later did with his XƎTEX project.

Initially, I made the changes needed so that the 8-bit data structures of TEX would allow 16 bits: number of input characters, number of glyphs in a font, number of available registers of each kind, and so on. However, the natural way of making these changes in Donald Knuth's code produced monstrously sized data structures. The core data structure in the TEX code base is a global table called the table of equivalents, which was savagely criticized by Ken Thompson, the inventor of Unix, soon after the code was published.

I had to develop a manner of transforming this table of equivalents into a data structure that implemented this large table in a sparse manner. This kind of task, a matter of a couple of hours in a decent language, was a real pain in the Pascal Web that Knuth used for his Literate Programming. Notwithstanding the praise for Pascal Web that Knuth heaped on himself, with the adulation of his followers, it is a useless tool. The real structure of a Pascal Web document is not the Pascal program that is generated, but, rather, the paragraphs that are used to make the aesthetically pleasing printout that is called *TEX: The Program*. To make a small change to the program may require

leafing through hundreds of pages just to make sure that nothing is missing. A similar experience was related by Taco Hoekwater, who rewrote the Pascal Web TeX code into C for the LuaTEX project: his conclusion was that this code was unmodifiable in its Pascal Web form.

Once these changes were made, then I started to work on more interesting changes, which always took far longer than they should have because of the use of Pascal Web. These changes included developing mechanisms for handling multiple character sets and encodings, Omega Translation Processes (regular expressions for rewriting the input stream for transliteration and other processing), and typesetting in multiple directions (vertical and horizontal, left-to-right and right-to-left). I also added features for automatically generating XML tags from TEX code, with hooks in the program and the input parsing to do this properly.

However, many of these neat ideas were stillborn, as they were not really tested. First of all, I did not need this tool for my own daily work, so I did not push the testing myself. As for Yannis, often he would ignore the elegant solutions that I developed, and hack together his own. As for the TEX community, most people were impressed by this grandiose project, but did not want to touch it, one of the reasons being that I dared, in public, criticize the Grand Wizard. The cult of the personality is as pernicious in science as it is in politics.

While all this was going on, I was working on the development of possible-worlds semantics, building tools for the creation of dynamic, multidimensional Web pages that could be co-browsed by multiple viewers, each with their own preferences. I knew that I wanted to add this sort of functionality into Omega, but as time progressed, it became clear that it would be impossible to incorporate all this into the Pascal Web code base.

Furthermore, the theory for this work was not even fully developed. I dropped the Omega project to focus on what I now call Cartesian programming, whose foundations I have been able to define this year.

E: *So was it all a waste of time?*

JP: No, not at all. Omega has had influence in unexpected ways. I noticed that the XML way for determining the character encoding is the same one that I invented for Omega; this may be an accident, but I did know and correspond with some of the people involved in this stuff. In consulting for ArborText of Ann Arbor, Michigan, I helped them update their typesetting engine so that it could print CJK fonts and do right-to-left typesetting, so Fortune-1000 companies could print their documentation and reports in Chinese, Japanese, Korean, Hebrew, Arabic and so on.

"Omega has had influence in unexpected ways."

As for the font work done by Yannis Haralambous for Omega, some of the fonts are used by Donald Knuth for his remarkable *Art of Computer Programming* book series.

I attended the EuroTEX conference in 2009 in the Netherlands, and was astounded to listen to a talk given by Hartmut Henkel of the LuaTEX project present the multidirectional ideas of Aleph, which incorporates Omega. This work was done entirely by myself, but he did not even know it!

I am encouraged by the fact that Taco Hoekwater had the courage to take apart the Pascal Web code and stitch it back together in C. I think that parts of this new code base could be adapted to become part of a new Cartesian document model, which Blanca Mancilla, others and I are trying to develop.

E: *What is your opinion of XeTEX and LuaTEX? And do we really need LuaTEX when PerlTEX can do the same things?*

JP: I think both projects are neat ideas, and deserve praise for actually producing software that works and that people use.

XeTEX brought all the neat fonts available on one's computer to the world of TEX, without forcing a user to adapt those fonts to the TEX typesetting engine. However, there are still teething problems. For example, I downloaded a large manual explaining how to use XeTEX, and it turned out that I could not read it, as it used non-embedded fonts that were not available on my Linux box!

As XeTEX is used more and more, it would be nice to have some sort of good-quality portable typesetting engine that could produce good results that would guarantee replicability across multiple platforms. Then the wonderful features of XeTEX would be compatible with the replicability results of TEX. But this is probably a lot of work.

LuaTEX is interesting because the project began by taking apart the TEX code, rewriting it in C, then putting in hooks so that the different components of TEX could be manipulated in Lua. I personally like Lua because of its model of tuples, similar to Perl hash arrays, which are relevant to my model of Cartesian Programming.

I think that both LuaTEX and PerlTEX are working with the idea of extending the original TEX idea of *document as program*, as opposed to the XML idea of *document as tree*. For both, the flexibility to the writer is increased, as is the programmability. There is a cost, however: the searchable document disappears into cyberspace.

E: *Could you elaborate on this "new Cartesian document model"? What is it? What do you expect to achieve with that for the TEX community or for computer science in general?*

JP: I have been working on something I call "Cartesian Programming" since about 2008, and will be defending a French Habilitation thesis in Grenoble on the topic in December 2010. The idea in Cartesian Programming is that the coordinate system originally introduced by Descartes is relevant to all of computer science. Any object, program, function, etc., varies in a number of dimensions, including time and space. In Cartesian Programming, all objects

are assumed to vary with respect to the same set of dimensions; some dimensions may be relevant to some objects while other dimensions will be relevant to other objects. The same idea is used in physics: when writing differential equations, one only refers to the dimensions of relevance, all others are just ignored.

Cartesian documents take the same idea of multidimensionality and apply it to documents. A document can be considered to be a multidimensional container that is indexed by a context in a way that is compatible with Cartesian Programming. A Cartesian document can only be added to, not subtracted from. As a result, the idea of the permanent URL is extended to take into account versioning of a document, as well as processing of the document, such as typesetting. In each of these situations, the document grows, and can be viewed in a richer and richer set of contexts.

I am not sure how this affects the TEX community. This is an intriguing question, because I believe the Cartesian document allows us to continue to consider a document as program, whilst still allowing searchability and replicability, since one can always write a finer and finer context to retrieve more and more specific details of a document.

Cartesian Programming is well advanced. Currently we have a programming language called TransLucid, with a working interpreter; it is designed as a coordination language on top of C++; variants based on other languages are currently under study. Cartesian documents will follow, but the timeframe is still unclear.

E: *Dear John, we would like to thank you once more for your contributions in the development of the future generations of TEX and for the extremely interesting conversation we had. Good luck into your new Cartesian endeavours!*

JP: Thank you!