Resume

Oleg Sivokon

<2014-11-28-Fri>

Contents

Contact Information

Name:	Oleg Sivokon
Birth date:	23, 06, 1978
Cell phone:	+972 (0) 544-563314
Email:	olegsivokon@gmail.com
Address:	Florentin 50, Tel Aviv, Israel

You can find an up-to-date electronic copy of this document at

https://github.com/wvxvw/resume

Introduction

I put free-form introduction before any historical data because I believe it can better describe me. I dedicated almost ten years of my programming career to the technology, from which I would like to depart now. Thus, I don't want to be judged on the merits of my past.

I'm looking to transition to Python. I've written some Python code before, but never an industrial-scale project. In order to teach myself the language, I developed a module for SCons build tool for compiling ActionScript code. I've also taken online courses in machine learning and artificial intelligence which used Python to illustrate the algorithms involved. Later, I wrote several utility programs for managing testing environment, used Python as a language for illustration of conceptual bits of a plan in a project specification and other minor tasks. Eventually I used Twisted, Tornado and Django frameworks, but it would be unfair to say I'm an expert. I'm a casual user at best, but I'm willing to improve.

At the time of this writing, it's been almost ten years since I wrote my first program. Most of this time I devoted to Adobe's technologies, Flash, Flex, and ActionScript. But it wasn't until about five years ago, when, after an introduction to free software ideology I started to distance myself from Flash. I made my first steps in Flash world largely due to the community support, learning through on-line resources, in cooperation with people like myself: passionate armatures.

I become a Linux user due to the community effort. Aside of programming, I dabble a little bit in philosophy, especially, I'm interested in moral philosophy, which for me is both personal and very important. So you can imagine that I'm a kind of an "ideological" Linux zealot. One that usually writes GNU/Linux, where others write Linux.

For a while I tried to make ActionScript programming on Linux more tolerable, but seeing how this is an exercise in futility and because I found better

friends in the Linux world, I gradually gave up the hope. I tried switching to Haxe, a free programming language that, beside other things, targets Flash platform, but quite soon I began to realize that it was a kind of a toy language, and I lost interest.

My first editor in Linux was Vim. But I never really learned it enough to feel comfortable with it. At the time I didn't know where to look for help or documentation; none of my friends even heard of such things and making progress on my own was hard.

Perhaps, if you ever played role-play games, this will not surprise you: many players and developers alike have a dream of an ultimate best game they will create. For me the dream was about the NPC (non-player characters) AI. The kind of "choose your own adventure" game, with no fixed rules, the plot unfolding unpredictably, boring hunting quests replaced by infinitely customizable game mechanics. This is where my passion for AI starts.

Purely by chance I read a Wikipedia article about SHRDLU. It had a link to the original sources written in Mac Lisp (I guess, or some other ancient Lisp dialect), which were entirely opaque to me. At the time I couldn't even imagine how much effort it will take to even understand the sources. I only realized that the program was a toy and a dead-end years hence, after I've abandoned my initial efforts.

Anyways, this is how I got into the world of Lisp and Emacs. Of course I made all the silly mistakes the Lisp novice would have made. I couldn't tell the difference between Emacs Lisp, Common Lisp and Scheme. This is how I ended up using Common Lisp. Even though today I understand that Scheme would've been a better choice, this choice stuck. Common Lisp was my first "serious" language. I could write some Java or C# before, eventually, I poked around PHP forum admin system. But Lisp was a whole different story! This was the language that was made the way it was because any self-respecting language would have to make all the same choices. It is, as if you compared Kantian imperative to the regulations in a medieval cloister. It was made by people who wanted to do things the (only) right way there is. Well, imagine my excitement.

There are only so many languages which can say the same about themselves. Haskell and some descendants of ML would be other examples. Forth and J are completely different languages, but they share the same spirit. Prolog is the same way too. Of course there are more, but I'll spare you the whole list.

At first I didn't like Emacs. It took me about half a year to try again. But the effort of learning it is the one which I will never regret. This has taught me a completely different way of looking at software, the interaction between software and humans, the goals which both may achieve. I felt once again behind the school desk when I read through the documentation and the source code. I used Eclipse quite a bit before it, obviously, FlashDevelop and Visual Studio. But this was so much different, and, well, better, so that today I feel like the computer is broken if it doesn't have Emacs in it.

Besides being an excellent editor, Emacs introduced me to a bunch of programming languages, it inspired me to return to school, to read programming literature, to change my everyday life and the way I use my computer. In the recent years this led to me becoming interested in natural language processing, machine learning, graph databases and history of programming. My motivation for learning math is largely due to my interest in mathematical logic, which is due to the interest in type theory and formal semantics. I know this will probably sound like I'm showing off (but isn't this the right place for it?), but I also believe that a good programmer must be familiar with these theories, and the more—the better.

CHAPTER 1

Employment history

1.1 PowToon Ltd.

- **Position:** *Programmer*
- **Term:** *January 2014 – Present*

Back to Flash... I have mixed feelings about it. It's like being good at being bad, except it's not the kind of bad, which is kind of cool.

1.2 TransGaming Inc.

- **Position:** *Programmer*
- **Term:** *April 2013 – 2013 (less than a year)*

Freelancing, although there is very little freedom involved.

1.3 Hewlett-Packard

- **Position:** *Programmer*

- **Term:** *October 2011 – April 2012 (7 months)*

Flex + some Java

1.4 Rounds

- **Position:** *Programmer*
- **Term:** *December 2010 – August 2011 (9 months)*

Client side of the web tier of an on-line video chat application.

1.5 BrainPOP

- **Position:** *Advisor*
- **Term:** *2008 – March 2010 (2 years)*

1.6 AditAll

- **Position:** *Programmer*
- **Term:** *June 2007–March 2010 (2 years 10 months)*

Front-end UI / video encoding.

1.7 Gazit Studio

- **Position:** *Programmer*
- **Term:** *January 2006–June 2006 (6 months)*

Small to medium size Flash applets, site menus, image galleries as well as games.

1.8 TimeZero LLC

- **Position:** *Designer*
- **Term:** *June 2005 – December 2005 (7 months)*

Designed costumes (armors) for the game.

CHAPTER 2

Education

This is my third attempt at post-high-school education. I studied twice before, first time in the Printing Academy in Lvov. Next, I studied in Bezalel in Jerusalem. Today I am a first year student in Open University. The plan is to study mathematics, but I might defect in the future and try to compensate my bad grades with some programming courses.

CHAPTER 3

Languages and Tools

Languages, listed by families, with my remarks on how well do I know them and on what occasion did I use them.

- Lisps
  1. **Common Lisp** - My "mother tongue". This is the language I use whenever I need to research a problem, to do my homework, to do a quick sketch of a program.
  2. **Emacs Lisp** - I wrote some library-quality code, and I write in it occasionally, whenever my Emacs needs some improvement.
  3. **Scheme** - I'm interested in Guile project, but I'm mostly just watching it make progress. Perhaps if I had more time, I'd try to look into Guile based Emacs.
  4. **Clojure** - few times, when I had to write something quick to run on JVM with no specific language requirement, I tried to use Clojure. I'm not a functional programming puritan kind of person, so Clojure isn't really my cup of tea.
  5. **ECMAScript** - you are probably surprised to hear that JavaScript is a... well, Lisp. Frankly, it's a very bad Lisp. Regardless, it's been present on the last couple of Lisp symposiums. Brendan Eich claims it to be "inspired" by Scheme. So, we'll have to live with this taxonomy. I wrote quite a bit of JavaScript code at different times on different occasions. I never liked it, even though I, probably, know the language very well, there's not much to really know well there, and nothing to be proud of.

6. **ActionScript** - This is just another name for a particular edition of ECMAScript designed by Mozilla and Macromedia, (later Adobe took over) around 2004-2006. I'm talking of course about ActionScript 3. ActionScript 2 is simply ECMAScript 2 with a handful of proprietary extensions. I've spent most of my programming career writing in ActionScript. I might know the ins and outs of the language better than anyone else living and writing in this language today, but, as with JavaScript, I just shrug whenever someone mentions it.
7. **TypeScript** - A "JavaScript with types". A toy language designed by Microsoft. Basically a preprocessor macro, which adds ML-style types to JavaScript. Very ad hoc and simplistic, yet I have to admit I wrote a project in this language. A loader for arcade games for Smart TV.

- FORTRAN-esque

1. **C** - I can read and, eventually, patch others code. I wrote a wrapper for C library, which required marginal knowledge of C language. This whole family is certainly not my cup of coffee.
2. **Java** - On several occasions I had to write some Java code. Once it was Spring, another time it was JEE, but I only did maintenance work on old large projects, fixing a minor bug here and there.
3. **C#** - About five years ago I wrote couple of plugins for FlashDevelop. At another time, I tried to extend MSBuild with some C# code, but I quickly realized nobody (even in the MSVS world) uses MSBuild, so I gave it up. Again, this is not a kind of language that has a strong appeal for me.
4. **C++** - I took a course on CUDA (it's an NVidia technology for programming for their graphic processors), but I didn't finish it. The course required writing in C++. This was the only time in my life I wrote in this language. I've no interest in repeating the experiment.
5. **Python** - Is the only language in this family that I'm happy with. I wrote few things in Python, mostly scripts to be used by others in automation of their daily tasks. Builds, testing, producing reports etc. I even have an example project I completed as a test when I applied to Walla for the position of Python programmer in my GitHub repo.
6. **PHP** - I encountered PHP few times in my life, but there wasn't a time when I didn't regret encountering it.
7. I don't actually know FORTRAN.

- ML-like

1. **OCaml** - I've used this language to practice when learning about the concepts of functional programming. All I wrote in it were the exercises from Project Euler, nevertheless, I believe, I have a tolerable level of understanding.
2. **Haskell** - I remember Douglas Crockford once said that today nobody talks about Java as a programming language, instead everyone uses it as an attack vector. Haskell is my attack vector. It would be unfair to call this language dumb or simplistic. After all, if you compare it to, say, JavaScript, it's a much, much better language. Yet, JavaScript never promised the kinds of things Haskell does, and I am particularly upset about Haskell's overpromise and underdelivery.
3. **Erlang** - Nice small language. I had a chance encounter with it about four years ago, when I had to write a chat bot in it. There are few awkward things about the language, but what really shines here is the abstraction of parallel execution.
4. **Scala** - This is practically an OCaml's clone. I have mixed feeling about this language. I never went past simple experiments with it, but I also can't really see a use case for it in my life. If I wanted OCaml, I'd rather use that. I've no use for the goodness of JRE.
5. **Haxe** - Haxe is actually a mix. It's half ECMAScript and half-ML. It feels like Nicolas Cannasse wanted very much to have ML in Flash, but was afraid that actual ML will scare the few supporters there were, so it kind of stuck half-way. Even though, this is probably your best option when it comes to writing for Flash Player. My last project in PowToon is half Haxe and half AS3, with the core of the project written in Haxe and the interface being AS3.

- Logic Programming

1. **Prolog** - I attempted it some time in the past and didn't understand it. I recently made a second attempt, equipped with The Art of Prolog, and I'm falling in love with this language.
2. **Mercury** - A more modern Prolog, compiled, with ML-like type system. An interesting language, which I tried once and would want to try more, if a possibility presents itself.
3. **SQL** - Yes, SQL isn't a logic programming language, but it is so closely related to Prolog, that I felt it would be appropriate to put it here. I'm not a pro. I had to use SQL with a handful of databases, but there wasn't much to write home about.

- Miscellaneous

1. **Forth** - This was my first non-mainstream language. It was totally mind-blowing. It made me rethink a lot of things about other languages. I'd be happy to get to know it better, if I had more time.

2. **PostScript** - Inspired by Forth, I looked at similar languages, and eventually, I even wrote a very reduced version of a PostScript interpreter in ActionScript (it's on GitHub too).
3. **J** - And when you'd think there can't be anything weirder... oh wait, J isn't at all weird. It looks like pigeon footprints to the non-initiated, but, in fact, this is a great language, with fantastic concept and very interesting perspective. If only I had more time.
4. **Cypher**, **Gremlin**, **SPARQL**, **RDF** - odd languages developed for querying graph databases. I've no good words for any of these.
5. **Shell** - Bash, actually. I'm not a pro, but you can't use a Linux computer without writing some shell script every now and then.
6. **GP 2** - This is a very unique language for graph processing, similar to Forth in that it's purely imperative and it's highly mathematical in that it's built around the concept of graph rewriting, specifically, double push-out. If only I had the time...

- Math and statistics

1. **Octave (Matlab)** - I never used Matlab, only Octave. I did it for the course in machine learning. I know enough of the language to do simple stuff for my math courses in the uni, but this is the kind of language where you need to be good at math and programming comes second.
2. **R (S)** - Surprisingly nice language. I don't know much about it yet, but few times I needed graphs I used its libraries, and that looks very impressive. I used R to work out the solutions for the assignments in few mathematical statistics courses that I took on-line.
3. **Calc** - This is the language of Emacs calculator. It is useful mostly for Org tables (aka spreadsheets). If you aren't familiar with Org, think of MS Excel VB macros.
4. **Maxima** - A math package written in Common Lisp (yet Maxima is a separate language). I know just enough to solve systems of linear equations.

- Markup and data

1. **XML** - At a time I compiled an E4X tutorial, which was quite popular on actionscript.org. I wrote an XML parser, probably even more then once. This also includes (maybe somewhat rudimentary XSLT, DTD and even RNG!).
2. **JSON** - As with Haskell, I'm trying to convince people that this does not achieve the goal of what a markup language can do.
3. **TEX** - I'm a newbie, but I'm improving. Obviously, this resume was set using LATEX and Emacs' awesome Org mode.

4. **Info** - I wrote several info pages, no kidding!
5. **Man** - When Adobe donated Flex to Apache I wrote Man pages for the compiler and other command-line tools from SDK, but they were never used...
6. **Protobuf** - Tried that too.
7. **AMF** - Most people don't know about it. This is a binary data format used by Flash. It's an open format and it fits ECMAScript languages very well. There are even libraries in several other non-ECMAScript languages to parse it. I once wrote a parser too, right, in Common Lisp.
8. **Dot** - The language for laying out graphs, saying this just in case.

By this time you may be wondering... yes, here they are!

- Languages that I don't know squat about
    1. **Pascal** - Yup, I've never written not even a single line of code in Pascal.
    2. **Basic** - Actually, I wrote some, but I'd rather not talk about it.
    3. **Perl** - I am horrified by the looks of it.
    4. **Ruby** - Nope, I know no Ruby either.
    5. **Smalltalk** - Neither Smalltalk.
    6. **Objective-C** - Oh, don't get me started.