Raoul Wols

curriculum vitæ

January 19th, 1989
Slobbengorsweg 129
3351LH Papendrecht
The Netherlands
+31 (0) 6 10917383
raoulwols@gmail.com
r@primef.actor
https://github.com/rwols
stackoverflow.com/users/990142
https://twitter.com/rwols
http://primef.actor

last update: 8th Sept, 2018

Education

from **sep 2001** to **june 2007**
Gymnasium
Johan de Witt Gymnasium (Dordrecht, The Netherlands)

from **sep 2007** to **june 2010**
Minor Jazz Piano
Royal Conservatory of The Hague (The Hague, The Netherlands)
Final grade: 7.5

from **sep 2007** to **july 2012**
Bachelor of Science in Mathematics
Leiden University (Leiden, The Netherlands)
Final grade: 7
thesis: Eindige Topologische Ruimten
advisor: Dr. R.S. de Jong

from sep 2012 **to july 2016**
Master of Science in Mathematics
Leiden University (Leiden, The Netherlands)
Final grade: 7.5
thesis: A McCord Functor for Alexandroff Categories
advisor: Dr. O.D. Biesel

Extracurricular Activities

from **sep 2007** to **sep 2008**
Treasurer of Music Committee
De Leidsche Flesch (Leiden, The Netherlands)
I was treasurer of a newly-founded committee that would encourage and incentivize students in the exact sciences faculty to get on stage and show their artistic talents.

from **sep 2008** to **sep 2009**
Chairman of Music Committee
De Leidsche Flesch (Leiden, The Netherlands)
The next year I became chairman of the same committee. The committee is still going strong as of this writing.

from **sep 2007** to **sep 2014**
Keyboardist
"The Circumstances" (The Netherlands)
I played keyboards in an indie-rock band. We played at lots of places like The Melkweg in Amsterdam, 013 in Tilburg. We made an album that's available on iTunes.

from **sep**
Baritone
Barbershop quartet "Auto o' Tune" (The Hague area)
I sang baritone in a quartet.

Experience

from **july 2011** to **jan 2012**
Intern
RoserConSys B.V. (Dordrecht, The Netherlands)
I researched resource constrained project scheduling problems. i.e. finding algorithms that will produce, in polynomial time, a "good" schedule. A resource constrained project scheduling problem is of class NP-hard if the goal is to find the optimal schedule. The languages were VB.NET and C++.

from **sep 2008** to **jan 2009**
Tutor
Leiden University Math Faculty (Leiden, The Netherlands)
I assisted a small group of students with their homework.

from **jan 2013** to **july 2014**
Teaching Assistant
Leiden University Math Faculty (Leiden, The Netherlands)
I assisted students in the Algebra 1 course.

from **july 2016** to **may 2017**
Founder
Primef.actor Entertainment (Papendrecht, The Netherlands)
I dedicated my time to create a video game called "Alien Invasion Game". The engine used was Unity, so everything is written in C#. Please refer to the "Projects" section to read more about this.

from **jan 2017** to **july 2017**

I have expanded my knowledge of C++17, I actively engage in "social coding" on GitHub via various forks and pull requests, I'm helping out the Sublime Text community, building a community around my plugin CMakeBuilder, and helping out the community around the TerminalView plugin. I became an expert at Git in this time (rebasing, pull requests, topic branching). I have also greatly expanded my knowledge of the Python programming language during this time, and I'm studying books about graphics programming. I did various code reviews for syntax highlighters over at github.com/sublimehq/Packages. I also made a giant contribution in the form of a rewritten Bash syntax highlighter and I expect this pull request to be merged after version 3.0 gets released of Sublime Text. The active pull request may be found here: github.com/sublimehq/Packages/pull/1057. I've also set up a website at http://primef.actor for which I'll be writing blog posts about various musing regarding C++ development.

from **sep 2017**

C++ Software Developer

QI Press Controls (Oosterhout, The Netherlands)

I work on various C++ and Python servers/clients. And a lot of image processing/transformations. As of may 2018 the team calls me "Scrum Master".

Projects

name — Plan-IT

language — *VB.NET*

description — Scheduling software with a GUI for which I wrote a proof-of-concept **genetic algorithm**. In computer science and operations research, a genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). The problem at hand was one where you have a set of "jobs", constraints between these jobs, and each job had a set of "resources" attached to it. The set of all jobs together with the contraints forms a directed graph. The goal is to minimize the time needed to finish all jobs, given that you have a finite number of resources per time unit available. A genetic algorithm turned out to work well.

name — Firing Neurons

type — *EP*

description — An Indie-Rock EP released on iTunes. I played keyboards.

name gintonic

language *C++*

description A hobby render engine. The current render method is deferred. It has a self-contained math library consisting of **vectors**, **matrices** and **quaternions**. A matrix is a linear map between vector spaces. A quaternion is a 4-tuple which one may add and multiply. Quaternions have a norm and a conjugation action. Every non-zero quaternion has an inverse, but the multiplication is not commutative. Hence we obtain a skew-field of quaternions. Quaternions can be thought of as encoding a rotation. The multiplication of two quaternions q_1 and q_2 results in a quaternion $q_1 \cdot q_2$ that first applies the rotation q_2 and then applies the rotation q_1. There is a natural bijection between quaternions of unit length and 3×3 rotation matrices. The engine features an **entity-component system** that underpins all **data-driven aspects** of the program. Various classes have accompanying unit tests, so you can trust that they work as advertised. The program relies heavily on the boost C++ libraries. The render backend consists of **modern OpenGL** using various vertex, geometry and fragment **shaders**. Find it at github.com/rwols/gintonic.

name Alien Invasion Game

language *C#*

description A 2D platformer with shoot mechanics. Can be found at rwols.itch.io. I have a lot of experience with **pair-programming** thanks to this project. I love working with others and coming up with solutions. The main problem of this project was getting the physics engine of Unity to work along with our vision of the game. In the end, we managed to use the components of the physics engine to our advantage. Some custom shaders were also written, and a custom loader for the maps was used. We forked Tiled2Unity for this.

name Clara

language *C++ and Python*

description Sublime Text 3 plugin for semantic C++ auto-completions. This has a compiled component. So the main language is C++, with some Python bindings. For each source file, there is an accompanying **translation unit** object that handles the **completion requests** while the user is typing. Since it may take some time before completions are presented, the whole procedure is implemented in terms of a **background worker thread** that waits for completion requests. This background thread is gracefully destroyed when the translation unit object is destroyed, and that happens when the user closes the buffer in the text editor. Find it at github.com/rwols/Clara.

name yaml-archive

language *C++*

description A **drop-in replacement** for `boost::archive::xml_oarchive` and `boost::archive::xml_iarchive` from the Boost.Serialization library. Find it at github.com/rwols/yaml-archive.

name CMakeBuilder

language *Python*

description Sublime Text 3 plugin for building CMake projects. The latest alpha version has a **server-client architecture** where a background process is started for each suitable cmake-project. This background process listens for requests and returns replies that the client, in this case the code editor, may consume. Find it at packagecontrol.io/packages/CMakeBuilder or github.com/rwols/CMakeBuilder.

name mRC3D

language *C++*

description **Camera** application for the QI Press Controls company. The software communicates with a custom-made board.

name Analyzer

language *C++*

description **Server** application for the QI Press Controls company. Main library used here is **OpenCV**.

name Press Controller

language *C++*

description **Server** application for the QI Press Controls company.

Natural Languages

Dutch Mother tongue.

English Fluent in conversation and comprehension.

German Basic knowledge.

French Basic knowledge.

Programming Languages

C++ Expert in language and standard library usage.

C# Expert in language, adequate in standard library usage.

Python Good at language, average at standard library usage.

Lua Good at language, average at standard library usage.

TeX Expert in (markup) language.

HTML, CSS	Average experience.
GLSL	Very good experience in language and library usage.
HLSL	Some experience; mainly via Unity.
CMake	Expert in language and library.
JavaScript	Some experience.
SQL	Some experience.

Programming Miscellaneous

Path Finding	I have a ton of experience regarding path finding and feel comfortable working with graphs, wether they are directed or undirected. In practice an A^*-search algorithm with a cleverly chosen heuristic function works very well. A^* ignores large patches of the search space and usually finds a solution in real-time. As a consequence the path may not be optimal. Contrast this with, e.g., Dijkstra's algorithm. This algorithm must process the whole search space (every node and edge of the graph) to get to an optimal solution.
Test-Driven-Development	I highly value unit tests for low-level classes. In general, classes with accompanying unit tests have a larger trust factor. I try to write unit tests while designing my classes, although I understand this is not always immmediately possible due to time-constraints or due to missing mock objects that the class may interact with. For instance, for a class that represents a database connection, you'd first need to create a mock database object. It's not always feasible to do this with regards to deadlines. On the other hand, going the extra mile and writing a unit test for it may have benefits for future projects.
Git	Git is my main tool for source-control. I have no problems in using it to my advantage for forking other people's projects and making a quick pull request to the original repository. I have used cherry-picking for commits from other branches and generally use topic branches for all of my features to implement. These days, I also like to make a pull request for the topic branch to my own master branch on GitHub, so that others may see what I'm doing even in a topic branch and even if it's not merged into master yet. This makes code reviews easy.
Code Reviews	I value code reviews, a lot. I find it interesting to read other people's code and I am at home in a code base in no-time.
Agile	I am aware of this method of software development. In general I would say that I prefer this method over the "Waterfall" method. Mainly because a working software prototype has more value than a design document, I believe. In a design phase, one may overlook details that may turn out to be the bane of the project. The devil is in the details, as the saying goes. When one is not restricted by a strict design document, there is the freedom to possibly work around these details that prevent the project from going further.
Waterfall	The Waterfall method is sequential in nature and is characterized by a top-down approach of first determining the system and software requirements, then doing the analysis in e.g. Matlab and then doing the design and coding, and so forth. This method has its merits but I am more in favour of an agile approach.

Functional Paradigms

Functional languages like Haskell have inspired the C++ community to adopt certain functional features, even in the standard library. I value these ideas, too, but it seems impossible to let go of the object-oriented nature of C++. However, one may cherry-pick certain aspects of a functional language and use that to one's advantage. For instance, I strive to make my classes as "const" as possible.

Design Patterns

I'm aware of a lot of design patterns. Here are some: object-oriented, functional, data-driven, factory, abstract factory, lazy initialization, singleton (but don't ever use this!), RAII, facade, flyweight, command pattern, iterator, the none object, the visitor, thread pool, locking and joining, consumer-producer, strategy pattern, loose coupling, curiously recurring template pattern, substitution-failure-is-not-an-error.

Tools Experience

I have worked with various tools. Here are the most important ones that I have experience with: Visual Studio, Xcode, Sublime Text (my favourite right now), clang, clang-cl, gcc, cl, msbuild, ninja, cmake, gdb, lldb, docker, npm

Open Source Contributions

I regularly contribute to the syntax highlighters of Sublime Text, which are open-sourced over at github.com/sublimehq/Packages. I sometimes contribute to clangd, the language-server based on the clang compiler. I maintain the CMakeBuilder plugin at github.com/rwols/CMakeBuilder.