RESUME_redux.md

Michael Vaganov (`michael.vaganov@gmail.com`)

TL;DR - motivated programmer who loves teaching. See https://tinyurl.com/mvGitRes for details

Portfolio

- Projects: http://www.codegiraffe.com/portfolio
- Code Samples: https://github.com/mvaganov/
- LinkedIn: https://www.linkedin.com/in/mvaganov/ (recommendations and endorsements)

Notable Personal Software Projects

Suffrag Ex Machina

An experimental ensemble machine-learning technique that elects using ranked-choice voting.

Ethos

A prototype for a web-based assessment system designed to provoke personality development.

Impetus

An experimental suite of tools and prototypes to build a-game-about-Project-Management.

Skills

- **20+ years Programming**: hobbyist, game programmer, educator, consultant
- Programming Languages: C, C++, C#, Java, JavaScript, Python
- Software Domains: games, productivity & automation, UI/UX, multi-platform, Client/Server, VR, Web
- **15+ years Teaching Computer Science**: ages 7 to 40+. tutor, undergrad professor, high-school teacher

Employment

Key: (`F`) Full-time (40+ hrs/wk), (`p`) Part-time (~20 hrs/wk), (`v`) Volunteer (~10 hrs/wk)

Computer Science Teacher at Sacred Heart Prep

August 2015 to June 2019

Faculty member at an exclusive private school

- Taught computer science, with a curriculum designed to motivate (ask me about hacking)
- Wrote software widely used by school to manage day-to-day schedule notification

Code Coach at theCoderSchool

September 2014 to August 2015

Elite Computer Science education for youth (between age 7 and 17) in the Silicon Valley

- Custom-built simple and engaging tutorial content for Computer Science and electronic art
- Subjects: Unity3D, C and C++, Java, Blender, 2D and 3D math, Game Design, Project Management

Self Employed Programmer, Entrepreneur

December 2012 to Present

- Stealth startup project: Game & Project Management Software (unfinished)
- Contracted consulting work

Professor at DeVry University (Silicon Valley Campuses)

March 2006 to December 2014

Professor of Games and Simulation Programming (GSP), a Computer-Science-like Bachelors of Science degree program, with emphasis on game development

- Rated highly in students evaluations (consistently 3.5+ out of 4)
- Managed 30+ Senior Project teams (16 week project, 2 to 5 programmers /team)

Software Engineer at LimeLife

November 2006 to April 2008

Developer responsible for end-to-end network-aware mobile application development

- Senior-level engineer: product development, build-systems and automation, client/server

Software Engineer at Infospace Mobile Games

December 2004 to November 2006

Developer of mobile applications with emphasis on client/server interaction.

- Senior-level engineer: product development, framework, R&D, client/server

Porting Engineer at Atlas Mobile (later purchased by Infospace)

June 2004 to Dec 2004

Very productive first-6-months-of-professional-software-development.

- Client side QA developer, primarily tasked with porting and bugfixing

Education

Keller Graduate School of Management

September 2006 to 2010

Masters of Project Management

DeVry University

July 2001 to October 2004

BS of Computer Information Systems

Other

Volunteer Teaching

- Unityversity: nearly weekly classes teaching Unity and VR (Aug. 2016 to Present)
- Citizen Schools: public school outreach (Feb. to Apr. 2014, Oct. & Nov 2013)
- Coder Dojo Silicon Valley: conference-style tech meetups for kids (Sep. 2013 to 2017)
- Guest Lecturer at various universities in Uganda (Oct. & Nov. 2012)

Hobbies

- Hiking, Biking, Rock Climbing, Fencing
- Volunteer Teaching
- Software Side-projects, Game Jams, and Hackathons (samples at http://codegiraffe.com)

Personal Programming Axioms

- The price we must pay for being god-of-the-machine is Understanding.
- The best programmer writes the most Readable code. Speed is for the compiler.
- The first version is noisy, buggy, won't map-reduce, and later versions matter.
- The best code will survive long after a programmer leaves it.
- Single Point of Truth: One complexity, One bug, One change.
- Code explicit functionality rather than side effects, and /** document it */
- Comments are good, code that describes itself is better.
- Think about optimization now, but do the actual optimization later.
- Just Prototype. And don't expect another shot at it, so make it good!
- Refactor, Sooner rather than later; clean code grows into powerful code.
- Disciplined, results oriented software development is always in style.
- How most production code should be judged (in order):
  - Functionality: intended results are produced (with constraints in mind)
  - Survivability: useable again elsewhere (maintainable/readable/modular)
  - Robustness: stability with a wide range of input (no bugs)
  - Resource Use: resources used conservatively (Big-O, memory, threads, ...)
  - Everything Else: elegance/robust-unit-tests/optimal-efficiency/...
- The Unix way feels right (http://www.faqs.org/docs/artu/ch01s06.html)

Other Credo

- Persistence (iteration) is disproportionately important to success. (So, iterate. Faster.)
- Rules are for people who don't know any better; Rules are important, but understanding sets you free.
- Luck is where preparation meets random opportunity, which is happening constantly.
- To make the next best thing, the current best thing must be mundane.
- A spoonful of test dissolves a pound of design.
- Without clear goals we are wasting people's time, and we are made of time.
- Do not fear complexity, simplify it.
- more at: http://codegiraffe.com/quotes.txt