Mike Kramlich

Senior Software Engineer, Technical Lead, Architect

contact: groglogic@gmail.com

Summary & Highlights

programming since childhood. became profession. salaried/FT and remote/contractor experience
NOT one of the "coding bootcamp" newbs/fakers; doing since was not "cool", for passion and zero pay

- my code has run in your travel websites, a gas pump, a Bitcoin wallet, games and iOS apps
- bug fix contributor to open source projects (Electrum and Proxool)
- coded and successfully shipped many software projects for contract clients (most remote)
- 8 praise testimonials in public (managers, leads, coworkers, contract clients, game reviewer)

online presence: GitHub, LinkedIn, Twitter

- ideal/recent tech/skill mix: Java, Python, SQL, web, git, Linux, cloud, arch, bootstrap, leadership
- old/rusty/leverage: C, C++, native desktop GUI apps, iOS/Mac app dev (ObjC, XCode), Flash/AS3
- industry domain experience: travel, education, gaming, cryptocurrency

US citizen, resident, native, Colorado-based, excellent native English speaker, convenient US timezone

- *Orbitz:* former staff senior engineer (Java, Linux, web, distributed architecture, high traffic)
- *Dead By Zombie:* designed, coded, marketed world's first Python/commercial Rogue-like software
- designed/coded many computer games over years (30+; orig as hobby, then self-market & clients)
- *The Dread Space Pirate Richard:* published author of a comedy e-book (Amazon, SmashWords)
- *Diplomacy:* once won 1st place in a major regional tournament (in the 90s) in this historical board game of logic, strategy, diplomacy & leadership

hobbies/interests: reading, writing, game design, learning, music, LEGO, Advanced Squad Leader

Current Status & Recent Activity

freelance contracting & startup consulting

Maxitize: personal portfolio project to build a mock online shopping engine featuring techniques from data science, marketing & machine learning (A/B testing, correlations, Bayesian inference, and NN's)

tutoring a girl in math & writing; tutoring an adult in SQL; preparing to tutor HS student in calculus

This 1st page was brief intro and summary only. The remaining pages have more detailed CV and FAQ.

Goals

to solve business or societal problems via computer software, or innovation

to design, code, document, support, troubleshoot and optimize software systems

to surround myself with good, smart, mature, interesting people who I can both help and learn from

General Overview

Senior software engineer, technical architect & lead who is strong on programming and on engineering fundamentals, who loves what I do. Challenge me to bring out my best, and to get the most value.

Variety of technology solution and problem/industry domain experience. My current ideal employer or project tech mix would use some/most/all of: Linux, Python, Java, web apps, web services, SQL databases, git, cloud hosting and CLIs. Love excuses to learn and use new tech and tools.

Passion for and/or past experience with big distributed architectures, tooling, automation, client/server systems, backends, performance, scalability, technical leadership, small team leadership, code reviews, documenting, bootstrapping, prototyping, research, troubleshooting, UI design, textual UIs, GUIs, APIs, and free open source software (FOSS). Much game design and programming but not limited to it.

Programming since childhood. Learning continuously. Mixture of self-taught from books, original homebrew hobby software projects and professional on-the-job experience, both as a FT/salaried office employee and PT/remote contractor/consultant. Love to read, imagine, analyze, plan and design. Each day I try to learn something, solve a problem, make something or ship, however small or incremental.

Love making things. I believe I am good at fixing bugs, iterating, brainstorming, and sometimes hopefully innovating. But I clearly get things done and ship. The rest of my CV should back all this up.

Employment: Primary/Full-Time/Salaried/Daily-Office-Commute Work

2007 summer to present:
Principal Software Engineer at **Synrotek**

I'm working as an independent contractor doing software development and consulting for a variety of clients. The remaining sections of this doc include highlighted projects during this period.

2003 March to 2007 June:
Senior Software Engineer at **Orbitz/Cheaptickets/Trip/Travelport/Cendant** in Denver & Chicago

One of the world's biggest travel e-commerce website companies. I did a variety of things there

over the years, centered around programming, fixing and supporting Java web apps on Linux, all across the stack. I solved difficult legacy bugs in their Java code related to threading, memory management, caches, session handling, purchase event queues, database connections, and site performance. I also sometimes added or fixed user-facing product features. And did code reviews. Several times I wore the senior on-call pager for production support during evenings and weekends, representing software engineering. Over the years there I personally survived 4 company-wide layoff events, got a promotion and was twice tapped to work on core tech, architecture and production troubleshooting. I stuck with this company through many business ownership changes, management phases/methodologies, 4 different office buildings, 3 cities, 2 states, and about 3 different major rewrite-from-scratch codebase architectures. After 4 years I eventually quit, seeking a change and new opportunities. Loved the people there, kept in touch with many of them, and to this day think highly of my coworkers and our technical leadership.

Linux, Java, Python, Jython, bash, Spring, Ant, XML, XSL, JSON, HTTP, HTML, Javascript, Eclipse, JBoss, Bugzilla, JIRA, Confluence, LaTeX, JUnit, CVS, ClearCase, AccuRev, JMX, SNMP, GDS, Tangosol Coherence, DataPower XA35, Ethereal, Apache, Tomcat, LoadRunner, JMeter, TCP/IP, Graphite, Streambase, Proxool, SQL, threading, GC, JVM tuning, monitoring, logging, heavy/deep OO, TDD, DI, AOP

http://www.orbitz.com

http://www.cheaptickets.com

also see the testimonials farther down below, given by my longest manager/boss there (Scott) as well as a team lead engineer (Peter), as well as coworkers in network/ops (Marc) and QA (Reza).

2000 April to October:

Java Developer for **Gilbarco** aka **Marconi Commerce Systems** in Greensboro, NC

I did Java software development to help them create the outdoor customer-facing GUI of a next-generation gas pump model. I wrote the Java/C++ OPOS bridge and outermost implementation for the Keypad GUI. I also conceived, designed and coded a mock Keypad simulator and a test harness, and on my own initiative also created a few extra internal tools for the team including a JavaPOS diagnostics & compliance tool, and a JVM/JRE diagnostic GUI widget called SysInfo (imagine a JMX explorer client before/without JMX existing.) Our gas pump product model was featured in a year 2000 issue of *Scientific American* magazine (not sure which month, but it had a two-page spread with a big illustration of the pump — possibly the May 2000 issue, because SciAm index shows a piece title "Fill Er Up".

Java, Linux, JavaPOS, RMI, CORBA, OO, UML, Swing, Rational Rose, ClearCase, Kawa, VisualCafe, and the "Ice" embeddable Java browser widget

product series: http://www.gilbarco.com/us/products/fuel-dispensers/encore-700-s-gilbarco-gas-dispenser-pump

screenshot of SysInfo: http://synisma.neocities.org/SysInfo_screenshot.jpg

1999 July to October:

Software Testing Support for **AT&T @Home** in Englewood, Colorado

I installed, configured and supported their software during Y2K mitigation activities and compliance testing. patching of Solaris boxes. remote administration of Windows NT servers via SMS.

wrote SQL queries and Oracle PL/SQL scripts. some Oracle database configuration. supporting AT&T's Y2K testers with my knowledge of the Interplex GUI application, xBOI and CCS. (see my preceding gig for an explanation of these terms and context.)

1998 October to 1999 July:
C++/Java Developer for **TCI Cable / TCI.NET / AT&T @Home** in Englewood, Colorado

I did Java GUI application and C++ service backend development for the Interplex GUI call center application. distributed RPC middleware/bridge development and integration – the layer that sat between the Interplex GUI app and the CCS mainframe system provided by CSG Systems, via their API named xBOI. Looking back, I appeared at the time to be one of the few people who managed to get application client code running successfully on top of this CSG API. Therefore after my contract ended successfully and I left TCI I was soon thereafter contacted by several other people asking me to work on their systems which also needed to integrate with CSG's mainframe. TCI management also asked me to come back to help with urgent and necessary Y2K mitigation and testing support, in part due to my domain knowledge, and I obliged.

C++, Java, CORBA, DCOM, 3270, xBOI, Visual C++ 5, UML, Toad, Java AWT/Swing, Solaris

1997 September to 1998 September:
Senior Web Developer at **Digital Creators** in Boulder, Colorado

I contributed to the creation of web sites, web applications and multi-media CD-ROM's as a web developer and programmer. Designed & created a small Sybase SQL Anywhere database.

HTTP, HTML, JavaScript, DynaScript, CGI, Perl (for most server-side logic and batch jobs), Java (only small simple browser-side applets), UNIX shell scripting, a little Flash, a little Photoshop, ERWin, Sybase, MS IE, Netscape Navigator.

article about DC: http://bizwest.com/digital-creators-delivering-training-education-via-net/

Employment: Client/Contract/Remote Work

The clients/projects listed below were chosen as highlights because are generally the oldest, most inactive and/or the finite scoped, one-offs.

Postcast

I created a Mac app for a startup client to serve as their first working product proof-of-concept and help them gain additional investment. It was a GUI workflow tool which allowed you to load video clips from local disk, then specify subsets of the clip (bound by a user mouse specified rectangular viewport range, as well as time start/stop moments) with a custom text annotation. Its purpose was to assist video production teams tasked with reviewing raw footage or clip drafts, then suggesting edits or otherwise sharing context-tied feedback with other stakeholders.

Objective-C, Cocoa, QuickTime Pro, Mac video APIs, XCode, Mac

Dante's Dunk Tank

I fixed legacy bugs and added features to a "carnival style" web game for a contract client.

Flash, ActionScript 3.0, Python, Django, Linux, Audacity, Gimp
screenshot: http://synisma.neocities.org/dantes_dunk_tank_2A_Dante_v1.jpg

PencilBot ESL for **Double Encore** (in turn for **Edutainment Resources, Inc.**, owners of **PencilBot**)

I was the overall technical architect and team leader of a 3 person dev team which created and successfully shipped this iOS app series of educational games which taught English as a second language (ESL) to foreign students world-wide. There were 3 apps in the series total (Green, Blue, Red) and I structured the code and project config so we could build them all from the same master code tree and project files. I coordinated production workflow and timing with outside contractors (art, sound, video, text content) and did much of the ObjectiveC/iOS application programming as well, creating and finishing many of the features myself. Each app was essentially 5 smaller apps in one, all reached from a common top-level menu screen: an embedded topical video (with interactive captions that allowed the user to click on certain highlighted words in the caption text, which then paused the video and brought up an overlay widget, which showed its vocabulary definition — a very innovative feature, for its era, that we "solved" and brought to life, built on top of the built-in iOS media APIs), and 4 educational mini-games. I was the technical architect of their interactive video subsystem, and did 99% of the coding on it. I also reviewed all the other's code and gave constructive feedback and encouragement, and was responsible for maintaining quality and productivity. We worked both remotely and in F2F sprints I organized. We completed all features to spec and to the client's satisfaction, and then shipped all 3 apps successfully to the App Store, including all language/locale permutations. They all were approved by Apple and went live for sale by end of 2008.

Objective-C, CocoaTouch, iOS SDK, XCode, Python, QuickTime Pro, Gimp, Audacity, SoundConverter (Mac), JSON, bash, video formats and captioning, i18n/localization

PencilBot ESL site: http:///www.pencilbot.com/

PencilBot ESL screenshot: http://synisma.neocities.org/pencilbot_green_menu.png

also read the praise quote written by one of the programmers I led (Kenji), down in the Testimonials section below

FilaPet for **Luke Feldman**

I made a pet health management iOS app for a direct contract client couple who wanted it for their small biz venture. I did all of the technical design and programming. Luke provided the UI design, mockups and raw graphical elements, and his veterinarian wife Amanda brought the domain expertise and marketing. I completed it successfully to their satisfaction. We shipped to Apple. Went live in 2009.

iOS SDK/API, Objective-C, CocoaTouch, XCode, Gimp, bash, XML, HTTP, RSS

FilaPet site: http://filapet2.blogspot.com/

FilaPet screenshot: http://synisma.neocities.org/filapet_main.png

Luke Feldman: http://skaffs.com/ and http://lukefeldman.com

also read the positive testimonial farther down below by my client and boss on this (Luke)

The Adventures of Khaki Pants Pete for **Unilever (Klondike Bar)** via **Blockdot**

I made a retro 2D adventure game (in the spirit of Leisure Suit Larry — thus the name homage) about a guy, his house, and the adventures he gets up to one enchanted evening. I worked as a contractor for Blockdot (a major adver-gaming company in Texas) who in turn did it for their client,

Unilever (an international food conglomerate), with the purpose of promoting one of their ice cream products, the Klondike Bar (which was featured within the game itself.) I was the technical architect of this iOS app and did all its programming, including the technical design of a new script-driven game engine, and some helper tools (in Python and bash.) Someone else at Blockdot devised the game's "adventure game flow specification" script, in XML, so that they could feed the script to several different engines (iOS, Android, web, desktop). But I had to bring it to life for iOS by designing the engine to parse and then execute it, including the UI layout, the placement and movement of 2D bitmap sprites (with state-associated frames; and a fake Z-depth effect), animations, user interaction, sound effects and music. I completed and shipped all 4 chapters/versions to Apple successfully, and all went live, in 2009. We got lots of good early reviews and favorable star ratings with the biggest and most common criticism that users wished there was more content (more rooms, more scenes, more puzzles, more more more.) I was proud of the work I did, especially under the intense external time pressure from the third-party client, wanting to promote their Klondike product. I was also proud that I created and delivered a new, reusable engine codebase for Blockdot that I knew they could repurpose repeatedly in the future to make many other iOS games of the same style, all with greatly reduced risk, time cost and total dollar cost.

iOS SDK/API, Objective-C, CocoaTouch, XCode, Gimp, Audacity, bash, XML
KPP on GameSpot: http://www.gamespot.com/the-adventures-of-khaki-pants-pete/
KPP screenshot: http://synisma.neocities.org/klondike_mancave_start.png
also see the testimonial farther below, by my effective boss there: Matt Schmulen

Postabon (renamed since to **Signpost**)

I was the technical architect and sole programmer of this company's first iOS app, as a direct contract client. The app I made was a slick-looking, location-aware mobile front-end to their deal-finding service (think: coupons and promotions by restaurants and clothing stores in NYC). I completed it successfully and we shipped to the App Store, and it went live on sale. Postabon was featured on television on CBS and NBC, and in the New York Times. The actual iOS app I made was demonstrated live in a video segment. It was pretty exciting to see code that I wrote and shipped being used on screen on a major public TV network. I worked with and reported directly to their original co-founding CTO Shaneal Manek, who himself wrote all/most of their original backend in Lisp. He also built a public RESTful HTTP JSON service which I then made the iOS app communicate with, in order to submit and find deals to display. I estimate this was around 2009/2010. Sometime later their company gained more investment, including Google Ventures, shifted their feature/domain a little, and rebranded from Postabon to Signpost, and are still going strong today, in 2016. To this day I'd like to think my work helped them to reach that point.

iOS SDK/API, Objective-C, CocoaTouch, XCode, Gimp, bash, JSON, HTTP, geolocation, maps
Signpost (Postabon): http://www.signpost.com/
Postabon screenshot: http://synisma.neocities.org/postabon_nearby_in_map_mode.png

Startup Ventures

ZodLogic Games - *2007 to 2008*

conceived, owned, built and ran an online games website startup using a freemium subscription model and featuring at launch 5 original browser-based games (4 Flash games: *Adventureland*, *Flash War Command* (in Flash/AS3 and a diff game engine from my older C/Linux game of same name), *GGGG*, *Stratenism;* plus 1 browser-based HTML/Python web app variant of my *Dead By Zombie* game.) Abandoned and then closed the site eventually as a failed experiment as it never made enough money to justify keeping it alive (my advertising, hosting and dev costs exceeded revenue for too long.) Though I think it was a success from a technical and creative standpoint, and I learned a lot about business and games. I kept all the code, plus I later adapted that DBZ codebase into a downloadable standalone desktop Terminal/curses version, which ended up being more financially successful. I also got some traction with a downloadable Java strategy war game I created, during this same period, and under this same banner, named *Shattered Stars*, covered in more detail below.

Python, Flash, ActionScript 3.0, Apache, Django, MySQL, Gimp, Audacity, GraphViz, PayPal web APIs (handoff, postback, notifs), Linux, WebFaction, DNS, Google AdWords

images for Adventureland: http://synisma.neocities.org/adventureland.html

screenshot of Stratenism: http://synisma.neocities.org/stratenism_sshot01.png

screenshot of Flash WC: http://synisma.neocities.org/zg_flash_warcommand_sshot02.png

ZG's web DBZ: http://synisma.neocities.org/zg_web_dbz_help_example_sshot_annotated.png

Open Source Project Contributions

Electrum *in 2013*

I became an accepted code contributor to the Electrum Bitcoin wallet software project by solving a few legacy bugs and delivering fixes up into the main repo; my Python code commits made it into the global public release.

code: https://github.com/spesmilo/electrum/commits?author=mkramlich

Proxool *in 2004*

After careful analysis of production logs, debug runs and source code study I once found and identified a bug in this popular Java database connection pool library. It was a race condition [NOTE: if you the reader are not a programmer I want to clarify that "race condition" is a technical term of the trade describing a particular kind of computing flaw that can happen in multi-threaded software] in the init phase which could cause database connections to become lost and therefore leak over time. It was confirmed to be the root cause behind an actual production issue we saw at Cheaptickets.com, that I was assigned to investigate, and my deployed fix solved it. I believe I notified this project's maintainer and emailed him my fix as a diff style patch, but do not know if he integrated into a later release.

Personal Software Projects

I've been programming and making things since a kid. And reading, learning and experimenting continuously. First I programmed only as a hobby, later as a profession, while still as a hobby on the

side in my free time. Because I like making things, solving problems and learning. And I love interesting, intricate systems. Understanding them. Interacting with them. Therefore most of the work in this section is on personal projects that I devised and worked on in my own free time, for no pay, with no external direction. A few I commercialized, a few I've shared the source code, and some have screenshots online. The bulk of these projects were when I was younger and have tapered off lately as I've put more of an emphasis on ensuring a guaranteed monetary ROI. The list below is ordered approximately starting from the most recently originated, at the very top, then back towards the oldest. To save space this list leaves out most of the more trivial, or oldest, projects, as well as most of my homegrown code libraries and tools. The vast majority of the projects below are inactive or retired. There is only one project currently active as of this document revision date (meaning I've recently been putting serious thought and new work sprints into it): *Maxitize*.

Maxitize

This project is purely a portfolio showpiece where the goal is to have an excuse to play with certain ideas, tech and toolkits related to data science, marketing analytics, machine learning and recently popular cloud services. To integrate them in a coherent way, and demonstrate my understanding and mastery of them. The topic domain chosen is that of a mock online store and advertiser (like Amazon but super simplified), and the ostensible end goal is to maximize profits for this imaginary business. I plan to publish code and write-ups once it is mature enough, including more details on the stack tech mix.

Currently active, time-boxed per week & month, only in my unpaid free time, started 2016 July.

Python, sqlite3, math

post #1: http://synisma.neocities.org/maxitize1.pdf

NemesisBit

conception, research, design and prototyping of a new system to help Bitcoin with attack/disaster/scale risk mitigation via an original testing/simulation sandbox and service. mostly an R&D spike and POC, to scratch an itch, and learn.

Python, Linux, Bitcoin (bitcoind, core/orig)

Aviron 7

conceived, designed and coded this 2D side POV GUI game about an alien outbreak at a sci-fi colony, featuring vector-drawn ships (with mutable state and animations), user interaction, and a pauseable "real-time" looped game engine. more of a little experiment than a full game.

Python, PyGame, Mac

details & screenshots: http://synisma.neocities.org/aviron7.html

code: https://github.com/mkramlich/Aviron7

Warconomy

conceived, designed and coded a simple turn-based strategy game with a text CLI.

Python, Twisted, client/server, sockets

code: https://github.com/mkramlich/warconomy

Ganymede

conceived, designed and coded an original sci-fi adventure game with spaceships with a real-time event-driven engine in a topdown-POV 2D GUI desktop app.

Python, PyGame, OO, Mac

screenshot: http://synisma.neocities.org/ganymede_pic5.png

EduGamon

conceived, designed, coded an original system for web-based, text-oriented educational software which teaches the student lessons via user interaction with a first-person story-like engine and a hand-crafted world state sim. stateful per student, load balanced, an event engine, a scoring system, a custom API for creating sims, defining lessons, and several app-specific admin tools.

Python, web.py, OO, HTTP, HTML, Javascript, sqlite3, nginx, bash, Linux, Linode, DNS

Shattered Stars aka **Galactic War**

conceived, designed and coded an original turn-based strategy computer war game in Java for Windows and Linux. Play mechanics were based loosely on Axis & Allies. It featured 2D bitmap graphics, animations, sound effects, music, hotseat multi-player, multiple/concurrent threads (background tasks like animations or loads), events, sprites, curated scenarios, semi-random world generation, simulated battles and maps based on Voronoi diagrams. I sent copies and demonstrated it to Greg Costikyan (designer of the Paranoia RPG) and Johnny Wilson (retired Editor-in-Chief of Computer Gaming World magazine.) They accepted it for publication by their indie game distributor startup, Manifesto Games. Johnny wrote and published in his online blog a preview based on his play and analysis of a pre-publication build. Greg and Johnny both gave me game design feedback and encouragement. Their startup never got enough total sales revenue traction (this was before Steam and the App Stores) so they eventually closed it (releasing the web domain.) My Shattered Stars codebase lives still, might republish it some day. I created many in-house Java lib APIs and tools for this game that I reused with other apps. Feel free to contact Greg or Johnny to confirm all of this. I was fortunate to have friends and work colleagues do playtests and give feedback on my prod candidate builds.

Java, AWT, Swing, OO, XML

details & screenshots: http://synisma.neocities.org/sstars.html

preview article from Johnny's blog: http://synisma.neocities.org/johnny_wilson_preview.txt

Manifesto Games: https://en.wikipedia.org/wiki/Manifesto_Games

Computer Gaming World: https://en.wikipedia.org/wiki/Computer_Gaming_World

Dead By Zombie

conceived, designed, coded, documented, released, marketed, promoted, sold, supported and overall orchestrated into existence an original commercial 14,000 LOC pure Python traditional ASCII/Terminal-style Rogue-like game, with a zombie apocalypse setting and a comedy style. I gave it a proper OO architecture, a simple creature-based AI, a FSM-structured modal UI, and a tick-driven subscription-based event engine. Also a homegrown license generation/detection/feature-unlocking subsystem, config subsystem, random world generation and a modular architecture designed to allow multiple distinct Rogue-like games to each reuse a common core (named WebHack) easily via inheritance and overrides (I helped myself to ensure this feature worked by maintaining a 2nd working

POC game override in parallel to DBZ: a crude 80's Castle Wolfenstein variant.) DBZ made its public in-person debut at the *Genghis Con 2009* game convention in Denver where I rented a vendor booth and personally promoted it there while in costume as a Mad Scientist. I contacted, contracted and paid the comic book artist Richard Pace for the rights to use 2 of his existing zombie illustrations to promote and include in my DBZ releases and marketing. Built and publicly released downloadable standalone desktop binary distros for Windows, Mac and Linux. (Later dropped Windows support.) It never made a ton of money, and so I eventually withdrew it from sale and instead created a GitHub repo as part of my portfolio of public code, figuring that might be better for my career. I did learn a lot about marketing and the power of the human element, and customer service, in making software more successful. Takeaway: I started with a blank slate, imagined something, made it, shipped it and sold it.

Python, curses, OO, FSM, AI, PayPal
marketing & photos: http://synisma.neocities.org/deadbyzombie.html
code: https://github.com/mkramlich/Dead_By_Zombie

Once Upon Orion (game) and **Pythulhu** (engine)

conceived, designed and coded an original game and game engine with a terminal text UI.
Python

Cursed Mansion (game) and **LittleAdventures** (engine)

conceived, designed and coded an original game and game engine with a web UI.
Python, web.py

Apocalypse Dawn

conceived, designed and coded an original computer adventure/strategy game set in a post-apocalyptic USA. Loosely inspired by an 80's 8-bit era game named *Road War 2000*, crossed with elements of the original *Wasteland*.

C, curses, gcc, Linux, Mac and Windows (via CygWin)
code: https://github.com/mkramlich/miscpub/tree/master/ApocalypseDawn
screenshot: http://synisma.neocities.org/apocdawn_sshot01.png

Grio

conceived, designed and coded an original Java AWT-based widget class which provides a Terminal/REPL-like GUI element within your app with features including: toggle-able visibility/liveness, toggle-able debug/superuser mode, registered sets of commands, built-in 'meta' commands like help listings & repeat-last, command handler API for apps, built-in command input format enforcement & parsing, scrollable output/history separate from the input field, optional slide-in/out animation effect, customizable fonts & colors. Scratched an itch because I wanted something like this in several of my Java GUI app projects (like *American Barons* and *Shattered Stars*.) Grio was a reinvention and replacement of my earlier attempt to create an in-house Java console-like widget: *SwingShell*. I dropped features that didn't matter or were unused, and added new ones.

Java, AWT, Swing, OO

American Barons

conceived, designed and coded an original turn-based strategy/role-playing game about modern US life as a desktop GUI app. It started life as *Legacy* then I added many more play mechanics, a richer simulated world and a more complex UI and turn flow.

Java for Mac and Windows, AWT, Swing

Empyreal

designed and coded a small game with a subset of *Empire*'s features.

Python, curses

code: https://github.com/mkramlich/empyreal

Tactihack2

unfinished spike which recreates a subset of my previous game Tacti-Hack's features, but this time with a GUI and client/server arch (with multiple client types included to prove it, including CLI).

Python, PyGame, 0MQ

code: https://github.com/mkramlich/Tactihack2

Tacti-Hack

conceived, designed and coded an original hotseat multiplayer turn-based computer game about team-of-individuals/mission-driven/level-based combat with an engine-vs-scenario modular architecture, and play mechanics modeled after a mix of the classic game *X-COM: UFO Defense* with a *G.I. Joe*-like theme/setting and a Rogue-like UI.

C, curses, gcc, CygWin

StarSea

conceived, designed and coded an original turn-based strategy wargame with a hexagon-based map set in space with fleets of warships controlled by multiple player empires.

Java, AWT, Swing, 2D graphics, desktop GUI, algebra, trigonometry

screenshots: http://synisma.neocities.org/starsea.html

StarFront

conceived, designed & coded a simulation of real-time space-body travel & combat physics

Java, AWT, scaled zoomable 2D vector graphics & audio, desktop GUI, mouse, algebra, trig

screenshots: http://synisma.neocities.org/starfront.html

Ekonomy

conceived, designed and coded an original strategy computer game about buying and selling businesses. original version as a desktop app in Java with AWT. years later rewrote in Javascript as a pure browser-based app, and as an excuse to evaluate the new features of HTML5 and Canvas.

Java, AWT, Javascript, HTML5

SpaceWrath

conceived, designed, coded an original strategy computer game of ships & planets on a grid map

C, curses (and/or libgd, unsure), Linux, Windows (via CygWin)

manual: http://synisma.neocities.org/spacewrath_manual_rev2016.txt
screenshot: http://synisma.neocities.org/spacewrath_sshot.jpg

Legacy

conceived, designed and coded an original casual strategy game about life, mating, inheritance.
Java, AWT, Swing
screenshot: http://synisma.neocities.org/Legacy_manual_cover_pic.jpg

War Command

conceived, designed and coded an original real-time strategy computer game whose core game play and unit behavior mechanics were designed to emulate a classic old game I loved named *Command HQ* by Dan Bunten of Microprose. Featured a "real" Earth map with each player leading a modern nation, and pulsed/variable-speed unit (sprite) movement based on unit type, pathing, hidden placement/movement (fog of war), sight ranges, attack/ZOC ranges, city control/conquest, income, unit builds/buys, air/sea/ground units, air strikes, oil consumption/shortage effects. very simple, retro UI.

C, gcc, lint, gdb & valgrind (prob), OpenGL, GLUT, Linux, Windows, CygWin, algebra, trig
screenshots: http://synisma.neocities.org/warcommand.html

Imperium

conceived, designed and coded a turn-based strategy computer game whose core game play and rules were chosen to emulate a classic old game I loved named *Empire,* by Mark Baldwin and Walter Bright (yes, the D guy.) I added additional unit and terrain types, modes and effects, and tweaked unit stats to taste. I created all bitmap graphics by hand (for terrain, units, actions/animations) but used free or COTS sound effects. One of my most complete and "done" games.

C, gcc, lint, gdb, OpenGL, GLUT, Linux

Organism

conceived, designed and coded an original strategy desktop GUI computer game about rival colonies on a shared planet in a sci-fi setting. turn-based, top-down 2D perspective. I also designed and drew all the graphical elements for icons and textures.

C, C++, Windows
screenshots: http://synisma.neocities.org/organism.html

Valkyrie

conceived, designed and coded an original strategy desktop GUI computer game, involving turn-based battles between fleets of starships in 2D space. also created all the graphical elements.

C, Windows

Technical Writing & Thoughtful Proposals

Software Performance & Scalability: A Cheatsheet

http://synisma.neocities.org/perf_scale_cheatsheet.pdf

Relative Desirability of UI Types
http://synisma.neocities.org/comp_writing_interface_ideals.html

Dependency Installation: Ranking of Types
http://synisma.neocities.org/comp_writing_dep_install.html

Initial Advice & Tech Plan For A Hypothetical Web Startup
http://synisma.neocities.org/init_advice_and_tech_plan_for_hypo_startup.pdf

Multi-Threaded Network Programming and Linux: A Personal Demonstration/Discussion
http://synisma.neocities.org/Mike_Kramlich_on_mtnp_linux.pdf

Schools

University of Colorado at Boulder

My undergraduate course load emphasis was in Physics, Economics and History. However I eventually quit, before getting any degree, in order to reduce my expenses, increase income sooner and to learn more efficiently and flexibly. I've been a big reader and self-driven learner since childhood, with geeky hobbies and a love for hands-on projects and collaboration. And though I loved and respected education and teachers I was never a fan of school itself.

On a related note, near the end of my senior year of high school I was originally awarded a partial scholarship to join the undergraduate **honors CS program** at **UIUC** (University of Illinois at Urbana-Champaign.) I declined because CU-Boulder was still cheaper.

Awards & Geeky Claims To Fame

won a major **Diplomacy** tournament championship in Colorado (in the early 90's in Thornton.) It is a board game modeling a pre-WWI historical situation with players serving as national leaders, and requires a balance between chess-like logical thinking skills, strategic planning plus face-to-face interpersonal diplomacy, negotiation, charisma, speaking and group leadership skills.

with **Dead By Zombie** appeared to have released the world's *first* commercial classic (true, canonical, and in the original sense of the term) Rogue-like game, as well as the world's *first* one written in Python. At the time this was much harder and more novel than it would be today: back then there were no tutorials or Rogue-like frameworks or engines, and I designed and wrote mine 100% from scratch.

Testimonials

by Scott Beals (Senior Manager, Cloud Engineering, Oracle) on 2016 March 6:

"Mike was a highly valued member of my software engineering team at CheapTickets. Given his deep technical knowledge and expertise coupled with a great skill to analyze and solve problems, I threw the most intractable problems at Mike. He left no stone unturned as he worked through and solved every problem thrown his way. Mike delivered outstanding code for the CheapTickets platform with an eye towards ensuring the best user experience possible."

[NOTE: Scott was my boss at Cheaptickets in Denver.]

by Peter Dutton (Principal Software Engineer, Trustwave) on 2009 March 16:

"Michael is an excellent engineer with experience in a wide variety of platforms and software languages. He is a self motivated individual who is capable of getting up to speed on new systems quickly and delivering quality code in a timely manner."

[NOTE: Pete was a lead senior software engineer at Cheaptickets in Denver.]

by Marc Latour (Architect at Delivery Hero; prior: Amazon, Orbitz, Cheaptickets) on 2016 April 20:

"I have been very fortunate to work with and learn from many smart people in my career and Mike is one of them. My favorite collaboration with Mike was deploying DataPower's XA35 module into the Cheaptickets environment… he was representing Dev and I Ops. A very smooth and successful project. Sped up our transformations considerably and we were able to repurpose or retire a majority of our front end servers."

[NOTE: Marc was a network/ops architect at Cheaptickets in Denver.]

by Reza Motallebi (Sr. Business Analyst at IHS) on 2016 May 12:

"I worked with Michael during our time at Cheaptickets and I was really impressed with his quality of work and skills. Michael is not only a great technologist, but also very easy to work [with]; he makes challenging projects much easier to tackle. His intelligence and vision was always a big advantage on any project we worked together. I would recommend Michael emphatically."

[NOTE: Reza was in QA at Cheaptickets in Denver.]

by Johnny L. Wilson (Manifesto Games; Computer Gaming World) in 2006 April 22:

"Also, in keeping with Mike Kramlich's design goal of keeping the game simple and focused on the combat, you don't design your ships a la Space Empires IV or develop your technology tees as per Master of Orion, Reach for the Stars, and Space Empires IV. You collect money, you build ships, you move, and you have combat. It's pretty simple in execution—more chess than Sid Meier's Civilization. It is more like the table-top marble game Albedo than Othello or Go. Movement is important, but it is restricted enough that grand flanking movements aren't likely. If you can maintain your front lines and make sure you have enough reserves to cover a potential breakthrough, you can keep pushing forward by taking a few systems each turn."

[NOTE: The above quote is an excerpt taken from an article in his blog where he wrote a play review of a desktop computer strategy game of mine: Shattered Stars aka Galactic War.]

by Kenji Suenobu (Senior Software Engineer at AgilData, RMS Corp) on 2016 May 13:

"I worked directly with Michael on a very large iOS project, and he both managed, and directed the project very well. The project was very demanding, and Michael coordinated and led the project in

a very professional manner. Michael is a developer who will get things done. I would be honored to work with him again."

[NOTE: Kenji worked on my contract dev team that made Pencilbot ESL.]

by Luke Feldman (artist & designer; Director of LFG) on 2016 May 3

"Mike created a high quality, stable iPhone app that met our specifications."

[NOTE: Luke hired me as the sole contract developer to make FilaPet.]

by Matt Schmulen (iOS Mobile Engineer at Elementum SCM) on 2016 May 6:

"I worked with Michael Kramlich when I was the director of software engineering at BlockDot. Michael was the lead delivery engineer for Blockdot's Khaki Pants Pete iOS app. Working with Michael was absolutely fantastic! He is an excellent engineer, has a great 'get it done' drive to ship code, and goes above and beyond in making smart decisions that save time and improve the app experience. A perfect example of this is Michael's initiative to take the original Khaki Pants Pete spec and see the opportunity to make the underlying iOS game engine more generic and flexible via a data-driven XML game file format that made adjusting the game play, settings, actions and visual experience very easy. This single innovation allowed us to ship on time with a great experience despite the multitude of late 'change requests' from the customer that so often occur in our industry. I can't wait to work with him again!"

FAQ (or not?)

Q: Why did you become a contractor and/or go freelance after leaving Orbitz?

A: At the time, initially? Change of pace. New and different opportunities. I think salaried/direct/office/commute jobs and freelance/contracting/remote gigs each have their attractions, and appreciate both. I like striving for best-of-both worlds kind of solutions. And I love long term relationships and being able to invest a lot of time and focus into one company, one group of people, one product area. While also being able to leverage my strengths, and grow.

Q: I see a lot of games in your CV. Do you only want to be working on games?

A: It's interesting to design and play games. A game that is also software is a bonus because I love programming as well. Therefore, best of both worlds. And no, not just games. I've worked on lots of things other than games.

Q: I can't believe you made so many games!

A: I started around age 10. Many worked on in my free time. Programming and problem solving were hobbies of mine for many years before they also became my profession. Another hobby was designing and playing games. When you're a kid you don't think, "I know... I'll make a new type of database management system!" You think, "I'd love to have a turn-based strategy game kinda like X meets Y but with these differences, these tweaks, and I'd have all the source code so I'd be able to upgrade it any time, and, and..." The rate at which I've worked on them, or made new ones, has decreased dramatically in recent years. Partly due to having less time and more responsibilities. Partly

because I put a higher priority on spending time on things which make money or more directly help my career. I also don't feel I have to "prove" anything any more, whether to myself or others. Well, not as much! I know what I can do, what I've done, what I'm good at (I hope) and that I ship.

Q: Why so many solo projects? Why not more built as part of a group?

A: Mainly, because I could. :-) There's no sinister explanation. But I have realized that one's time is limited. It is finite. Therefore anything you can use to do software development *more* rapidly is a *win*. A solo project lets you move extremely fast with very tight cycles of "observe, think, decide, do, repeat." They also let you exercise a broader range of skills than you might otherwise. And let you work at your best pace, at times that best fit your schedule, focus or energy levels. It's also better for being able to play the most senior role possible, in terms of responsibility and duties, which is also helpful for your portfolio. That said, teams are also great, and collaboration is also great, with their own strengths. The social aspect of programming is great. Learning and sharing with others is great. It is just different, and differences are good. Personally I've tended to do most of my own solo projects when I was younger and knew less, then more of my most collaborative work or team projects in recent years.

Q: Have you worked with unit tests? Do you know what TDD is? Are you against tests or TDD?

A: Yes I've written unit tests and suites of automated tests, and inherited legacy test suites and had to maintain them. I've also created and inherited lots of codebases that had no automated tests, and they worked perfectly fine, with zero or near-zero defect rates — therefore I know that's possible, especially given the right people, tools and practices. I know what TDD is, and I strongly appreciate it. I think TDD is incredibly smart and the perfect choice for some situations, where it's clearly a net win. And I think there are other situations where TDD would be a poor choice, or where its costs outweigh its benefits. Writing space rocket control system software — where things explode and people die if there's a bug — is a great example where not only are automated tests and TDD smart and a no-brainer, it should probably be considered field incompetence or legal malfeasance if they aren't used. Any project with large teams of mixed skill levels would probably benefit strongly from TDD or at least a comprehensive body of automated tests. For a small project, only one developer, it's not mission critical, perhaps it's a prototype or there's extreme pressure to get to market fast? No TDD and no automated tests needed — especially if the programmer is good enough and otherwise follows good practices. Ultimately, what's best depends on the project and the people involved. There's no single one right answer for everything. It can help or hurt. Do it if clearly helps in your situation. My top rule about tests and TDD is this: if my boss insists I use them then I do.

Q: Can you do OOP? What do you think of OOP?

A: Yes. I think OOP can be both good and bad. In some situations it fits well and therefore is good. For example, to model play world entities and actor taxonomies within a game. In other situations it doesn't fit well, or is overkill, or creates unnecessary complexity, or problems with understanding and managing state (eg. requiring shims like an ORM.) It depends on the situation. I appreciate it. I have designed and inherited many codebases which were heavily OO. I also understand and can design/code with plain old imperative/procedural, FP, declarative, DSLs, data-flow pipelines, EDAs, etc. Almost all my earliest programming was imperative and procedural, especially in C and Assembly. The worst I've seen OOP abused has been in large legacy enterprise Java codebases which

had too much abstraction, too many unnecessary "design patterns", too much indirection, astronaut architecture, XML, etc. I don't always do OOP anymore but when I do I like less ceremony and/or safer practices: duck typing, shallow inheritance, composition, high cohesion, low coupling, etc.

[NOTE: if you wish, see the Dead By Zombie codebase for some of my past OOP design work.]

Q: Do you know or have experience with "X"? (some particular tool, API, service or technology)

A: Short answer: if not, I can learn it, and relatively fast, as needed.

Longer answer: I may have already done so professionally, but not listed it. I may have done "Hello World" with it, at home. I'm almost certainly aware of it, read about it, and understand the fundamental patterns which make it work. I may have built and shipped with something extremely similar to it. There are many similarities between separate tech and tools, and those similarities can be leveraged. And the total surface area you can leverage grows with more years of learning, building and solving. (If you're familiar with LEGO, programming/computing is a lot like LEGO.) I'm pretty strong on the fundamentals of programming and software engineering. There are shared patterns and phenomena and techniques which are the "building block" elements that all of software engineering rests upon (eg. binary, bits, memory, processes, algorithms, etc.) They are reused everywhere. Having a dynamic model in one's mind of how a computer works, how an OS works, etc. goes a long way. I have learned enough on demand in the past that I am confident I can continue to do so in the future. It has helped to have started this mindset young. Helped that programming & imagination were big parts of my personal life long before became key to my career. Helped to be a reader and geeky facts sponge.

Hobbies

learning, reading, writing, game design, gaming, programming, walks, LEGO, Advanced Squad Leader

Online Presence

blog: http://groglogic.tumblr.com/

portfolio: http://synisma.neocities.org/

GitHub: https://github.com/mkramlich

LinkedIn: http://www.linkedin.com/pub/michael-kramlich/3/ab7/51a

this document last revised: 2016 November 17