Resume/CV of Mike Kramlich

Mike Kramlich

2024 August 1

# Introduction

Mike Kramlich

programmer, software engineer, technical lead/architect

langs: Golang (Go), C, Python, Java, Unixy shell scripts

in the past (so avoid, but can lean on): JavaScript (JS), C++, Objective-C

OS: Linux (POSIX/UNIX), macOS/OSX

misc: SQL (sqlite, MySQL & Postgres); Docker; Kubernetes (some, once); AWS (mainly EC2 & S3); HTTP; TCP/IP; sockets; JSON; XML; Terminal-based workflow & CLI tool ecosystem; Apache; Nginx; memcached; Redis

# Software Type Experience

* web/Internet client & server-sides
* native desktop
* done native mobile app dev in past (can leverage it, but avoid)
* some embedded experience (eg. a kiosk), but avoid
* prefer “backend” focus now

# Specialities & Expertise

* performance & scalability
* latency optimization (instrumentation, monitoring, testing & tuning)
* threading & concurrency
* Internet & web client/server
* distributed systems
* solving hard problems (ie. fun puzzles!) (eg. Heisenbugs)
* prod incident on-call/pager & troubleshooting
* refactors & rewrites
* prototyping & MVPs
* pioneering or greenfield dev
* brainstorming

# Role Fit

*roles I can play and enjoy; not exhaustive, just examples*

* R&D
* systems programming
* perf lead
* tech lead of small teams

# Overall

* programming since age 10
* auto-didact and voracious reader from childhood on
* care deeply about my craft
* solid mental model of computing systems & associated fundamentals
* strong opinions, weakly held
* decades of professional experience
* FOSS contributor, tool-maker and lib author (eg. my *LatLearn* on GitHub)
* solved and shipped, over and over again
* good writer and oral communicator
* code samples in public repos on GitHub:  
  <https://github.com/mkramlich>

# Employer Highlights

* **self**  
  remote contracting & expert consulting, and (sometimes, also, have made/sold) indie computer games
* **The US State Department**  
  tech lead on natsec-related “counter-disinfo” project
* **Orbitz/Cheaptickets**  
  (on staff 4.5 years total) senior software engineer in Ops/Arch and core/cross-cutting tech (JVM, GC/leaks/crashes, logs, caches, sessions, thread pools (eg. fixing critical concurrency bugs), database conn layer, perf issue resolution, pager on-call if outages)

# Side Projects

*(active or recent)*

* **LatLearn**  
  latency perf instrumentation & reporting lib for Golang (as FOSS). Nanosecond scale measurements. Stat calcs for each named span, across set of all samples: min, max, last, mean, weight, cumul. Report file gen (in plain text so VCS friendly, enough structure for parsing.) Option to run common benchmarks of interest. Host & runtime config probe, added to reports. Overhead self-sampling, and option to auto-compensate in reported values. Variant span families: metric groups that vary only by task param values, environmental conditions or alternate flow paths or function exits. Concurrent & thread-safe by design. Plays well with Contexts, defer and panic. This lib provides features Golang does not out-of-the-box, and neither do generic/lang-agnostic system tools or profilers. Transparent and makes no “calls home” so maximally trustable when integrating into app code. Also: zero price.  
  <https://github.com/mkramlich/latlearn>
* **Tight Lightning**  
  writing a book about all things related to compute system performance, scalability, efficiency and cost (in effect: HPC). been working on it in my free time, slowly, on and off, for a few years. hope to finish 1st ed and publish sometime in 2024
* **Slartboz**  
  making a new real-time Rogue-like game (with a retro Terminal-based TUI) in a futuristic, sci-fi, post-apoc setting. style/theme for fans of Wasteland/Fallout, Idiocracy, HHGG (Douglas Adams), Stranger Things, X-COM and Dwarf Fortress. The engine is homegrown. And btw: I’ve been making game engines (both as an unpaid hobby in my free time, and, sometimes for clients or for the indie game market) for decades. My first public Rogue-like game (Dead By Zombie, in 2007) was commercial, turn-based and written in Python.  
  tech: Golang (Go), ncurses (goncurses), VLC (via Go lib), LatLearn  
  promo: <https://github.com/mkramlich/slartboz-pub>

# Writing Online

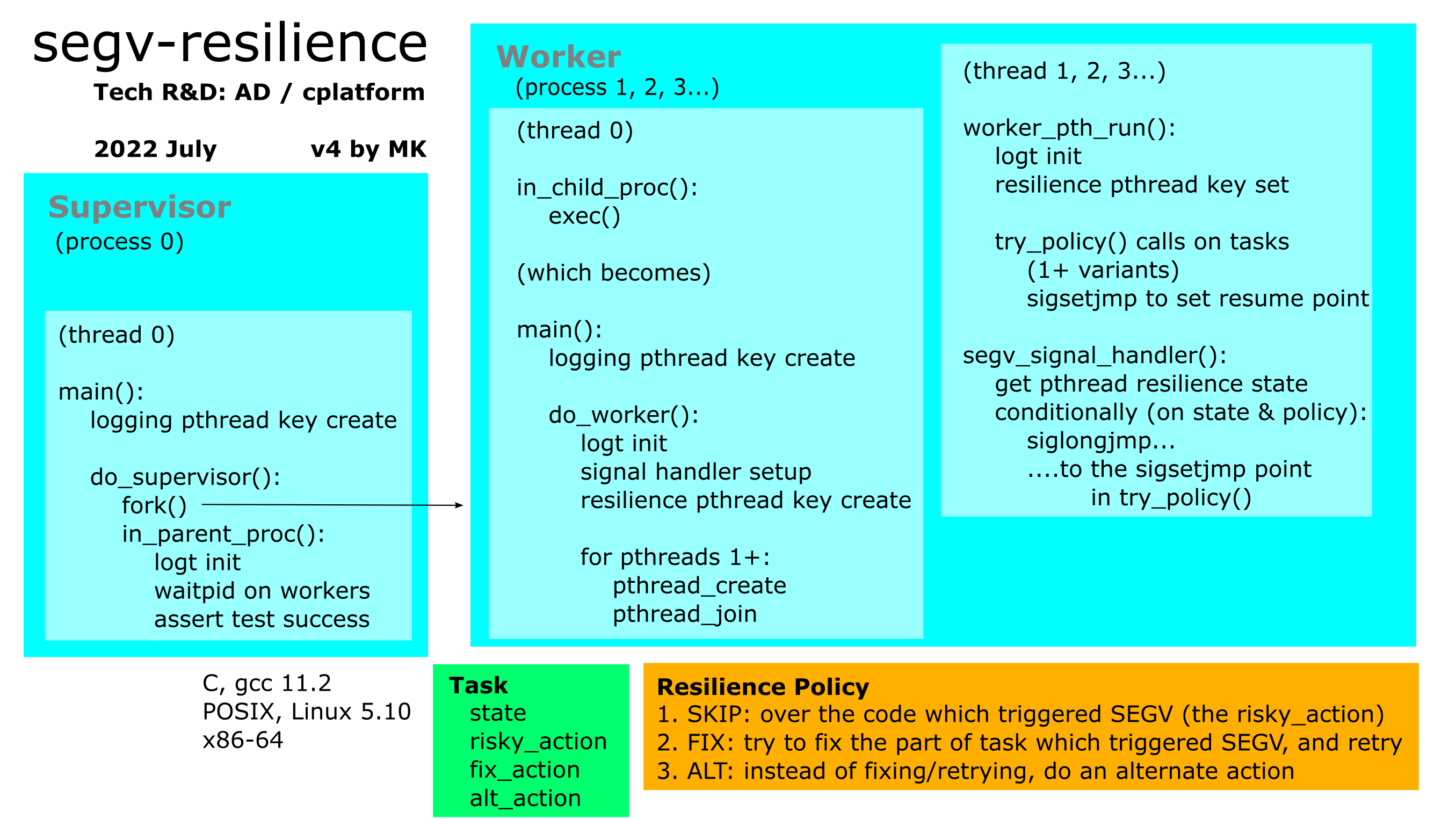
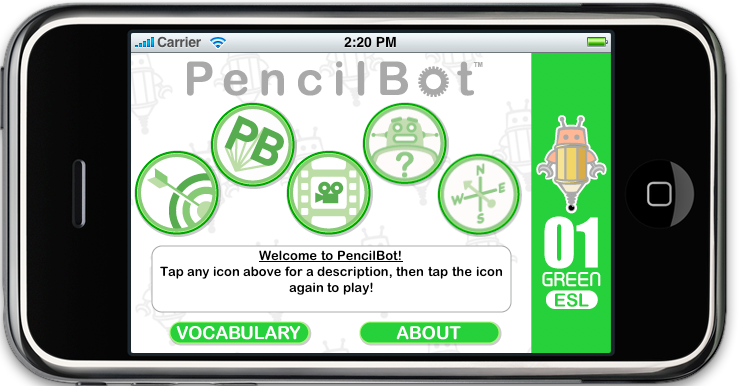
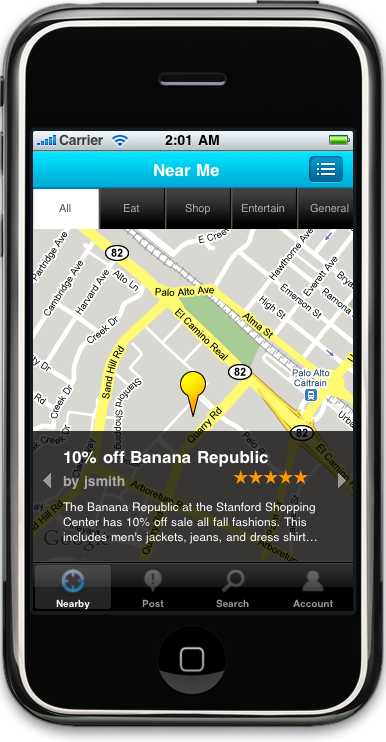
* *Fixing Java Thread & Database Connection Leaks For Fun & Profit*  
  <https://synystron.substack.com/p/fixing-java-thread-and-database-connection>
* *Software Performance & Scalability: A Cheatsheet*  
  <https://drive.google.com/file/d/1DsUG90gEkBtEsHJ7qJWR-VWLxHSTspcA/view>
* *CLIFMO: My Software Engineering Philosophy, Preferences and Practices*  
  <https://drive.google.com/file/d/1_zl1euz6sIhJ2VFonkaDS9grHNATBJfo/view?usp=sharing>
* *Video Calls Good and Bad*  
  <https://synystron.substack.com/p/video-calls-good-and-bad>
* *Time Wasters: What & Why To Avoid*  
  <https://synystron.substack.com/p/time-wasters-what-and-why-to-avoid>

# Talks & Presentations

* Computer Hardware & Software Performance (Talk 1, v4)  
  <https://youtu.be/nR7oPrmyCP4>

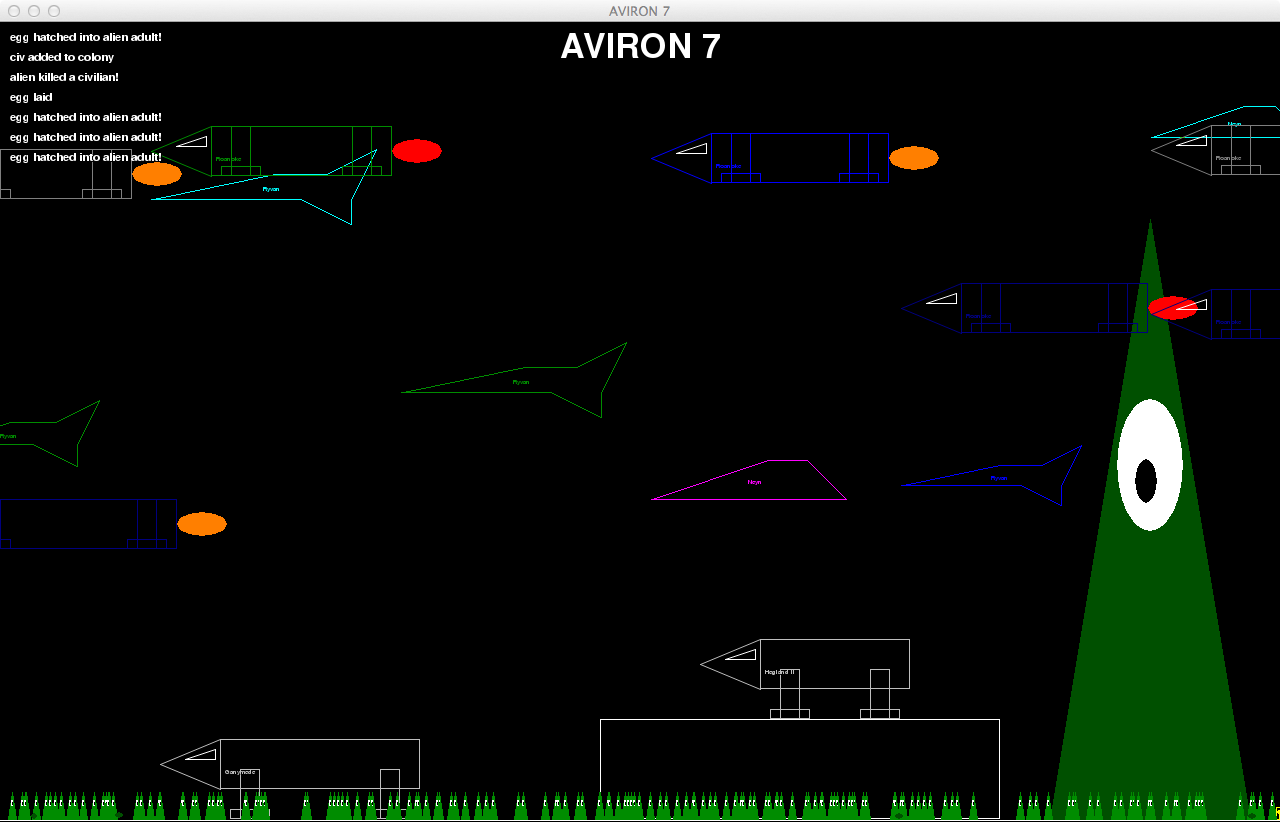
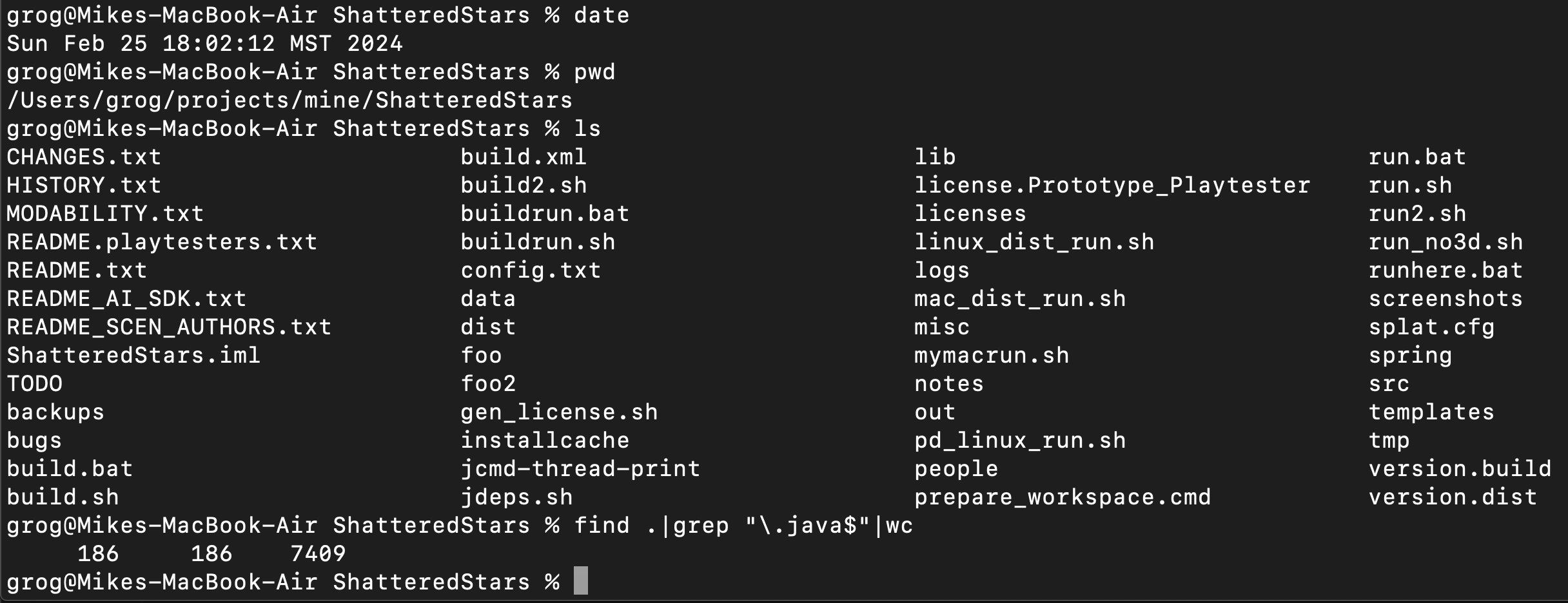
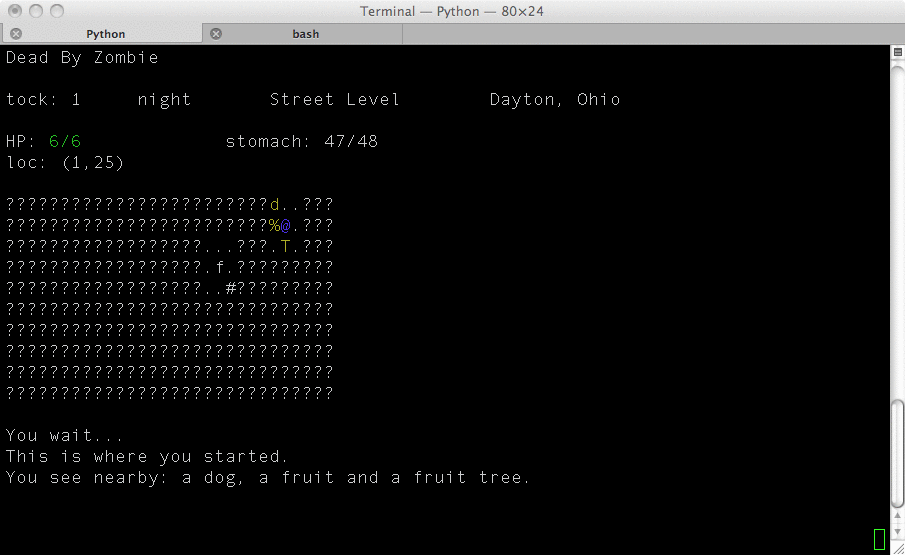
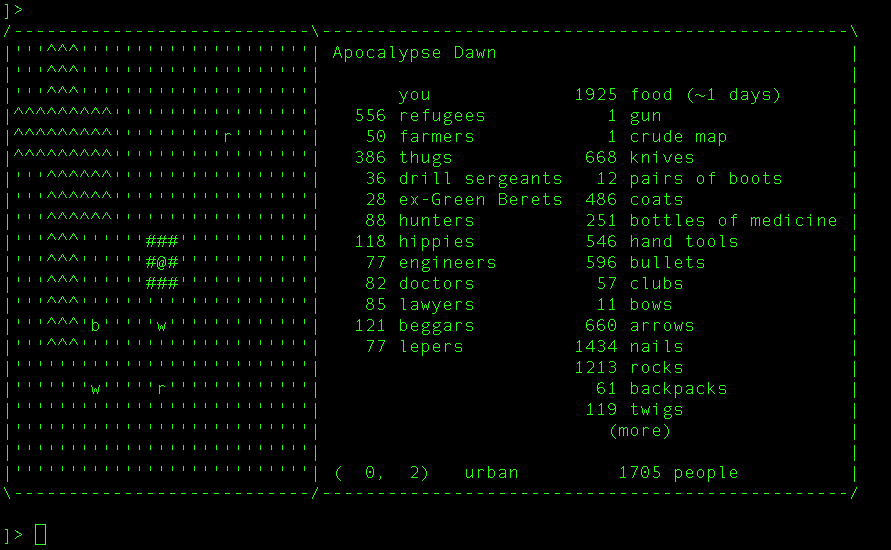
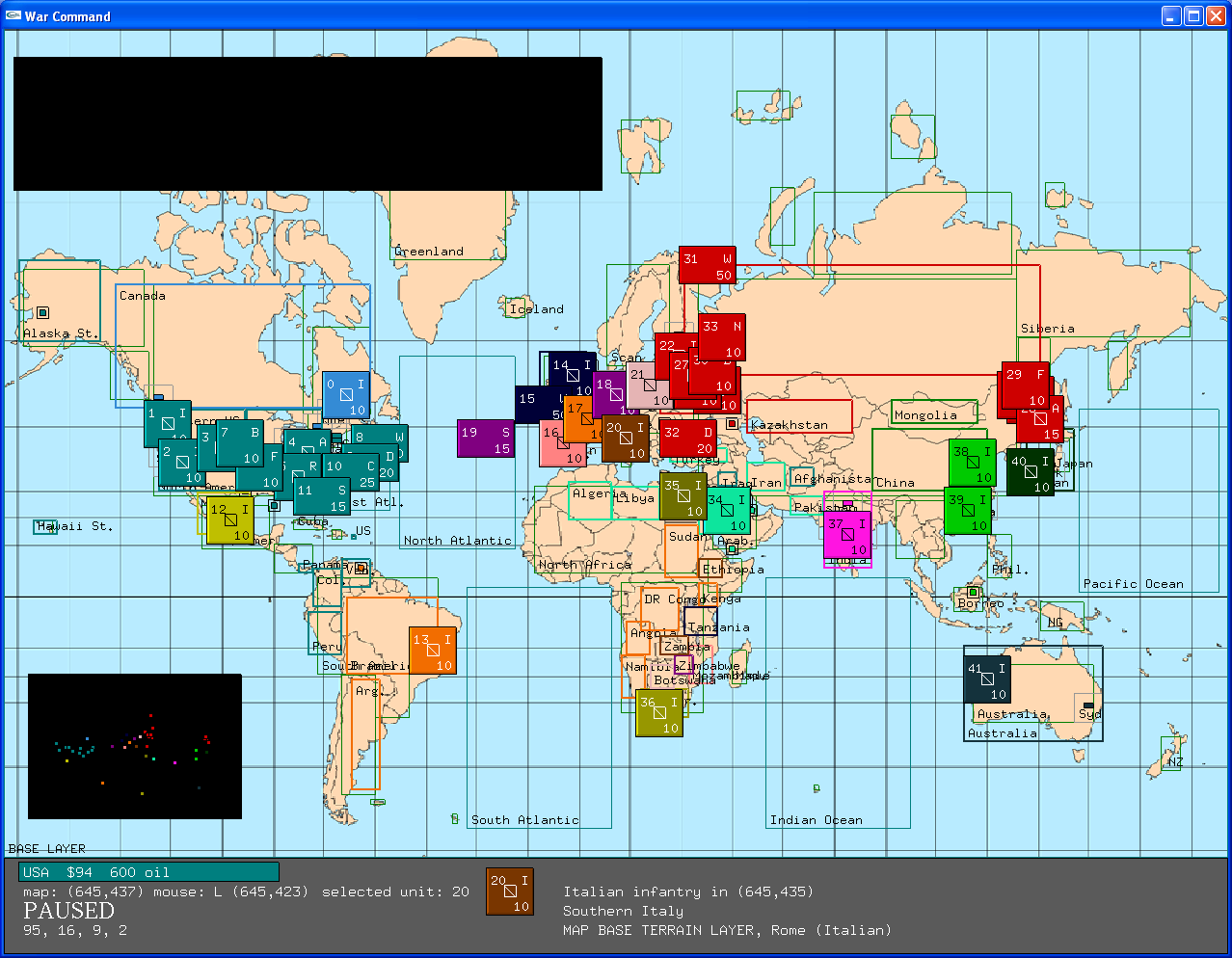
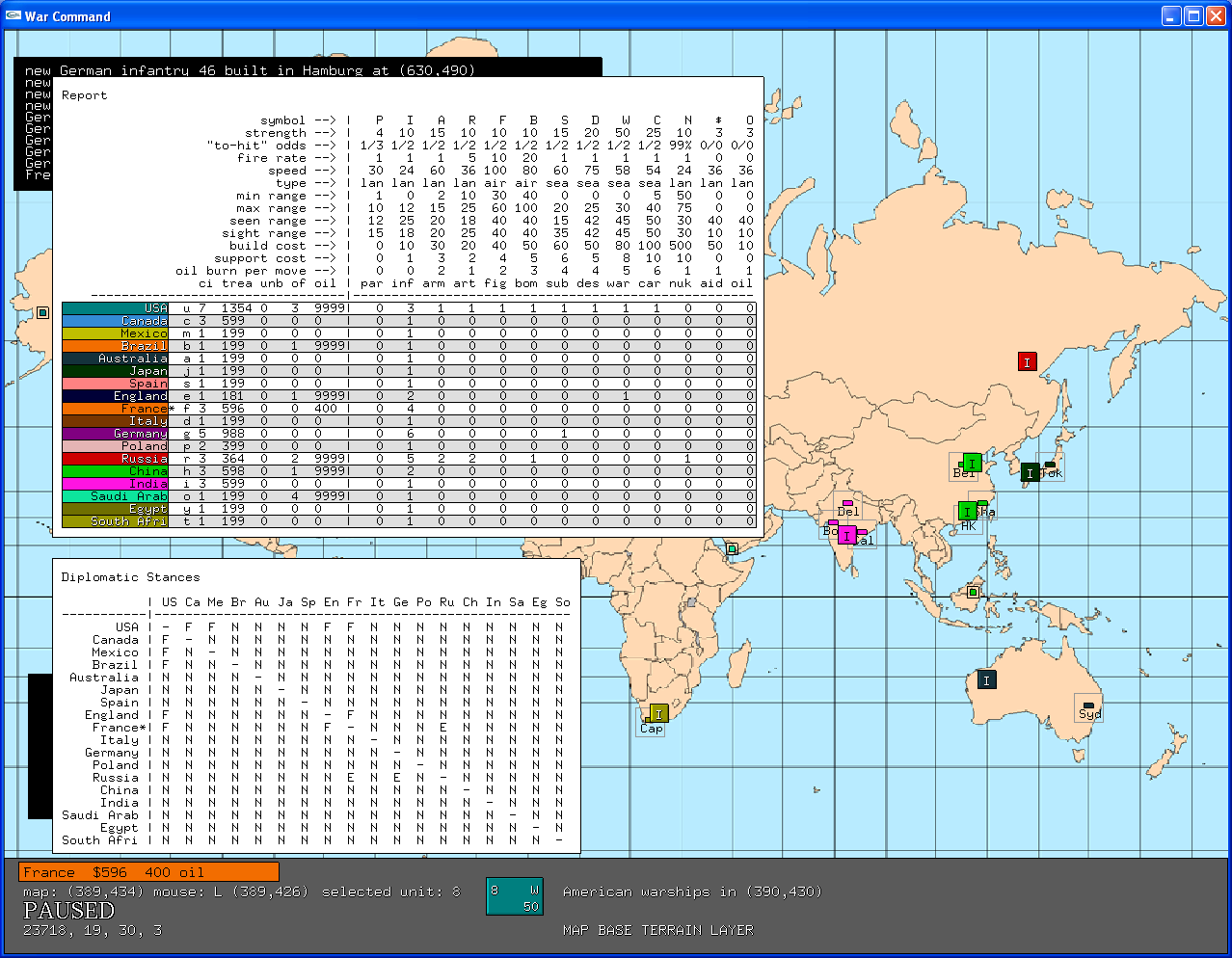
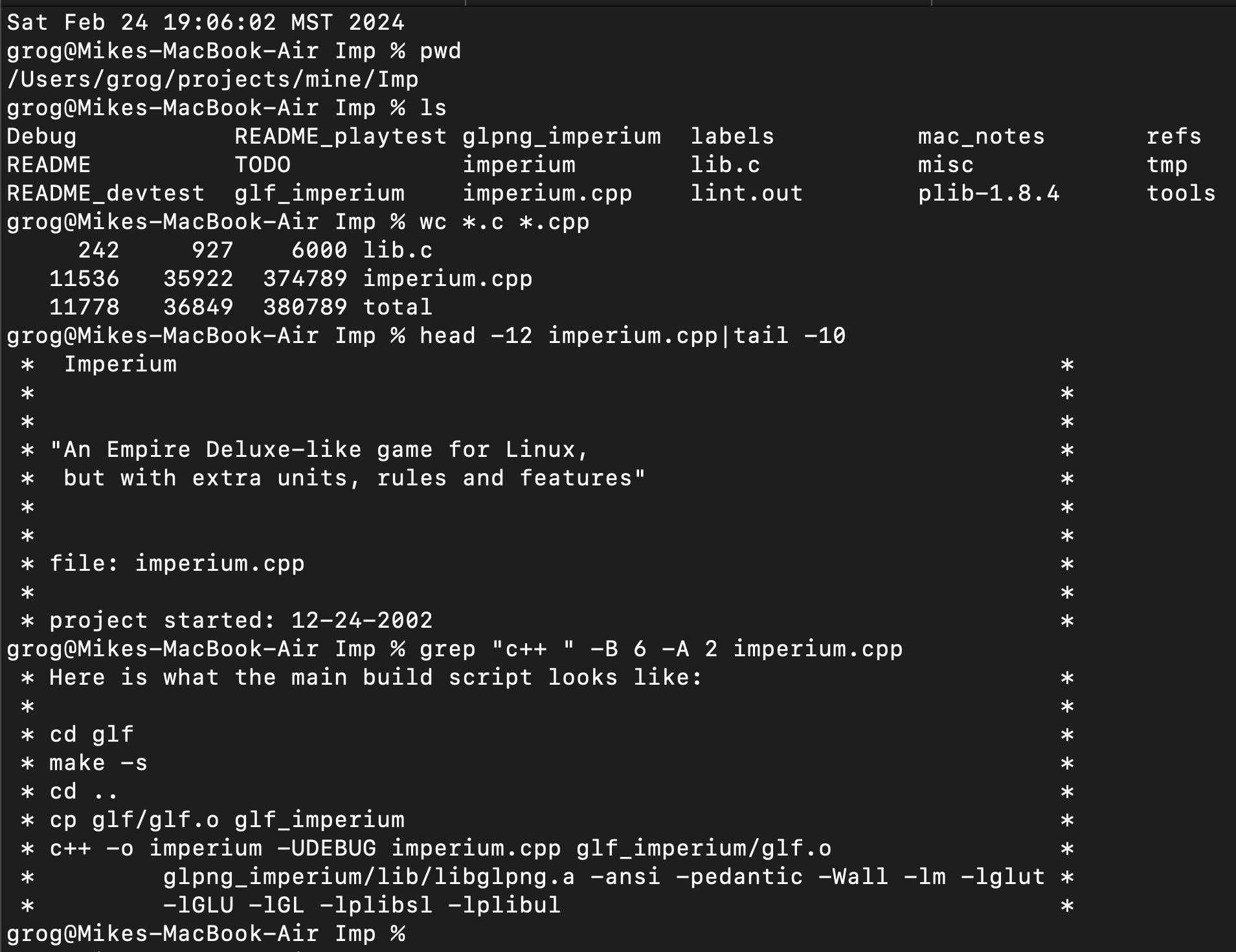
# Contract Work

*(past, completed; order is arbitrary)*

* memory alloc latency & SEGV resilience R&D, in the ad industry. They had a very old but polished C codebase and were in a highly competitive space where any downtime, wasted milliseconds or blocked threads meant significantly lower revenue. Thus they were “chasing 9’s” and my challenge was to try giving their legacy engine a performance and uptime upgrade, by any means possible. I believe I succeeded. I delivered new C code, benchmarks, diagrams and a report.  
  tech: C, mmap, signals, fibers, POSIX, Linux, Solaris  
  
* created a novel real estate “AVM” market-clearing price prediction engine for a client using multi-dimensional OLS regression on rich labeled tabular data sets from multiple sources including regional broker listing agencies.  
  tech: Python, numpy, pandas, scipy, statsmodels, sqlalchemy, ElementTree, gnuplot, pandoc, SQL, sqlite3, bash, machine learning (ML), the then-dominant real estate listing service protocols/formats (can’t remember names anymore)
* created a Bayesian inference engine for a client in the online insurance industry.  
  tech: Java, Python, SQL, bash, Linux, machine learning (ML)
* fixed legacy bugs and added features to a “carnival style” web game.  
  tech: Flash, ActionScript 3.0, Python, Django, Audacity, Gimp, Linux
* fixed legacy bugs and significantly improved stability, latency and scalability of an iOS app in the geospatial imaging industry.  
  tech: iOS SDK, Objective-C, CocoaTouch, XCode
* advised a client with respect to a travel startup in their investment portfolio, serving as a de facto consulting/bootstrap CTO, and delivered a working prototype that demonstrated a way to solve one of their key technical obstacles.  
  tech: Python, Javascript, bash, Linux
* **The US Department of State (GEC)**  
  software research, product technical due diligence and architectural consulting as the Technical Lead of a team of outside, civilian expert contractors tasked with helping to quickly identify and objectively report on the best available commercial tools and relevant technology for the US government and its allies to counter or prevent online, foreign adversarial propaganda & disinformation (APD). Specifically focused on the kinds of APD operations believed by the US intelligence community to be carried out by traditionally hostile nation states such as China, Iran, and North Korea, but most especially by Russia. We evaluated many tools and technologies over the span of several months. We evaluated each candidate by dividing its qualities into several categories. I was responsible for the 6 most technical categories. The categories were: maintenance, interoperability, scalability, reliability, resistance to misuse, and security. For these categories I did the bulk of the research and had the dominant voice in deciding our final characterization, ratings, ranking, and recommendations. I was also the primary author of the text in their sections, in the final reports our team delivered. To be efficient and fair, our evaluation process used a filter funnel technique with multiple stages of increasing scrutiny, higher standards and greater time and energy investment, as well as delivering increasingly longer and more detailed & objective reports. In the later stages of the products I evaluated I was typically the main author of anywhere from one half to two thirds of the total page count of the report versions we delivered for the end client, the State Department. I also created and delivered an original questionnaire tailored to help us gather common relevant facts on the products we evaluated. It had 48 technical diligence questions spread across the categories that I was responsible for – I devised all of them. I championed it as a way to both improve team speed and productivity. As well as to ensure thoroughness, and to boost the credibility of the project, both in the eyes of all client stakeholders and in the leadership of the companies we evaluated.  
  [State’s GEC](https://www.state.gov/bureaus-offices/under-secretary-for-public-diplomacy-and-public-affairs/global-engagement-center/)  
  [C-SPAN video of my team’s work cited in US Senate testimony by then-head of GEC](https://www.c-span.org/video/?470038-1/propaganda-disinformation)
* 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.  
  tech: Objective-C, Cocoa, QuickTime Pro, Mac video APIs, XCode, Mac
* overall technical architect and team leader of a 3 person dev team which created and successfully shipped an 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.  
  tech: Objective-C, CocoaTouch, iOS SDK, XCode, Python, QuickTime Pro, Gimp, Audacity, SoundConverter (Mac), JSON, bash, video formats and captioning, i18n/localization  
  
* made a pet health management iOS app for a direct client who wanted it for their small biz venture. I did all of the technical design and programming. Client provided the UI design, mockups and raw graphical elements, and his partner brought the domain expertise and marketing. I completed it successfully to their satisfaction. We shipped to Apple, and it went live to real endusers.  
  tech: iOS SDK/API, Objective-C, CocoaTouch, XCode, Gimp, bash, XML, HTTP, RSS  
  
* technical architect and sole programmer of a company’s first iOS app, as a direct contractor. 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. 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. 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, 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. Sometime later their company gained more investment, including Google Ventures, shifted their feature/domain a little, and rebranded, and were still going strong as of 2016.  
  tech: iOS SDK/API, Objective-C, CocoaTouch, XCode, Gimp, bash, JSON, HTTP, geolocation, maps  
  
* made a retro 2D adventure game (in the spirit and UX mechanics of Leisure Suit Larry), for a client in the ad industry, about a guy, his house, and the adventures he gets up to one enchanted evening. Contractor for an adver-gaming company who in turn did it for a major worldwide food conglomerate. with the purpose of promoting one of their ice cream products. which was featured in 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 devised the game’s “adventure game flow specification” script syntax, 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 then it went live for sale to the public. 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.) for the the hefty $1 they paid (!!!) I was proud of the work I did, especially under the intense external time pressure from the third-party client, wanting to promote their product and coordinate with a multi-medium campaign. Also proud that I created and delivered a new, reusable engine codebase for them 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.  
  tech: iOS SDK/API, Objective-C, CocoaTouch, XCode, Gimp, Audacity, bash, XML  
  
* Java software development for a gas pump vendor to help create the customer-facing GUI of a then-innovative 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 May 2000 (volume 282, issue 5) because SciAm index lists a piece titled “Fill ’Er Up”.  
  tech: Java, Linux, JavaPOS, RMI, CORBA, OO, UML, Swing, Rational Rose, ClearCase, Kawa, VisualCafe, and the “Ice” embeddable Java browser widget  
  [SciAm, May 2000](https://www.scientificamerican.com/issue/sa/2000/05-01/)  
  
* Java GUI application and C++ service backend development for GUI call center application for a major telecomm billing provider. Distributed RPC middleware/bridge development and integration – the layer that sat between the GUI app and the legacy mainframe system provided, via a bridge API. Looking back, I appeared at the time to be one of the few people who managed to get application client code running successfully (and reliably) on top of their API. (Its docs at the time were soooo bad it almost felt “intended to fail” haha.) Therefore after I finished my contract successfully and left I was soon thereafter contacted by another company asking me to work on their systems which also needed to integrate on top of that same vendor’s bridge API.  
  tech: C++, Java, CORBA, DCOM, 3270, xBOI, Visual C++ 5, UML, Toad, Java AWT/Swing, Solaris

# Personal Projects

*(past, inactive, mothballed)*

* **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.  
  tech: Python, PyGame, Mac  
  code: <https://github.com/mkramlich/Aviron7>  
  
* **Warconomy**  
  conceived, designed and coded a simple turn-based strategy game with a text CLI.  
  tech: 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.  
  tech: Python, PyGame, OO, Mac
* **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.  
  tech: 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.  
  UPDATE: In the summer of 2020, around July 4th, I spent a day refurbishing my old “cobweb-gathered” code and fixed it to work again with the then latest version of Java. Which was Oracle’s 14. It had previously assumed Sun’s Java 1.4!  
  tech: Java, AWT, Swing, OO, XML  
  [Manifesto Games](https://en.m.wikipedia.org/wiki/Manifesto_Games)  
  [Johnny’s review](./misc/shstars_johnny_wilson_preview.txt)  
    
  
* **Dead By Zombie**  
  conceived, designed, coded, documented, released, marketed, promoted, sold, supported and overall orchestrated into existence an original commercial 14 KLOC 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 design aspect 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.  
  tech: Python, curses, OOP, game character AI (no NNs or LLMs, haha), PayPal  
  promo artist: [Richard Pace](https://richardpace.artstation.com)  
    
  [GenghisCon 2008 booth photo](./sshots/dbz-gc2009-booth4.jpeg)  
  code: <https://github.com/mkramlich/Dead_By_Zombie>
* **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.  
  tech: C, curses, gcc, Linux, Mac and Windows (via CygWin)  
  code: <https://github.com/mkramlich/miscpub/tree/master/ApocalypseDawn>  
  
* **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.  
  tech: 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.  
  tech: Java for Mac and Windows, AWT, Swing
* **Empyreal**  
  designed and coded a small game with a subset of Empire’s features.  
  tech: 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).  
  tech: 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.  
  tech: 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.  
  tech: Java, AWT, Swing, 2D graphics, desktop GUI, algebra, trigonometry
* **StarFront**  
  conceived, designed & coded a simulation of real-time space-body travel & combat physics  
  tech: Java, AWT, scaled zoomable 2D vector graphics & audio, desktop GUI, mouse, algebra, trig
* **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.  
  tech: Java, AWT, Javascript, HTML5
* **SpaceWrath**  
  conceived, designed, coded an original strategy computer game of ships & planets on a grid map  
  tech: C, curses (and/or libgd, unsure), Linux, Windows (via CygWin)
* **Legacy**  
  conceived, designed and coded an original casual strategy game about life, mating, inheritance.  
  tech: Java, AWT, Swing
* **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.  
  tech: C, gcc, lint, gdb & valgrind (prob), OpenGL, GLUT, Linux, Windows, CygWin, algebra, trig  
    
    
  
* **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.  
  tech: 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.  
  tech: C/C++, Windows
* **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.  
  tech: C, Windows
* **Zodlogic Games**  
  I once built and ran a tiny indie computer game biz which sold access/license to games I made. There was a website with several Flash games, 1 web browser-based game (an early version of my Dead By Zombie) and I brought up to 2 downloadable native/desktop games to market too (a separate version of Dead By Zombie, and Shattered Stars.)  
  tech: Nginx, Django, Python, MySQL, PayPal, shell, Flash, ActionScript, JavaScript (JS), HTML, HTTP, DNS, Linux

# Testimonials

* “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.”
* “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.”
* “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.”
* “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.”
* “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.”
* “Mike created a high quality, stable iPhone app that met our specifications.”
* “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!”
* “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.” – by Johnny Wilson (ex editor of Computer Gaming World), in his preview of my game Shattered Stars, back when I ran ZodLogic Games and he was part of Manifesto Games

# Culture & Policies

* Unixen (eg. Linux, Mac) & UNIX philosophy; terminal-based workflows; vi/vim; text (plain prose or structured/parse-friendly syntax/DSL); CLIs/APIs/REPLs/TUIs; white-on-black (text color scheme, when working); KISS; MVP; iterating; AWC (always working code); ABS (always be shipping); RDD (README-driven dev); time boxing; evidence-driven; truth bias; CLIFMO (my term for a bundle of practices which work well together, in observed practice, though certainly not unique to me); bias to action; avoid “bike shedding”; “make hay while the sun shines”
* try to stay in a state of bliss; have fun; laugh; try to make others laugh
* try to always be learning, improving & helping teach or inspire others
* prefer people who use my time *efficiently*
* prefer people who let me work for/with them in a way that *maximizes* what I can deliver
* prefer people who *demonstrate* how much they value me (not necessarily in $!)
* … and I strive to do the same thing, in return
* NO UNPAID WORK
* want free samples of my code? look in my public repos. Golang, C, Java, Python
* TO BE VERY CLEAR (UPFRONT): I am not at a stage in my career where I have to prove I’m a programmer
* if you think I’m a liar or fraud or otherwise NOT a real programmer or engineer then PLEASE do me favor and DO NOT CONTACT ME. will save me hassle, and time I can reinvest on folks who do respect and value me more. why? life short, time precious
* READ my resume (this document), please, BEFORE our 1st call. before your 1st email to me, or recruitment decision; I will do something complimentary: I will read a job desc, your company site, perhaps a subset of your tech’s docs, and I often try to find an online bio of the person with whom I plan to speak
* I hate talking about myself, or ever having to self-promote; I wrote this doc in large part so I could “do it once” and get it out of the way; I find narcissism to be nauseating and will avoid taking on coworkers who exhibit it
* will not talk/deal with *third-party* recruiters or *outside* headhunters
* my camera will be *OFF* in video calls. *screensharing* is ok
* no *commutes* (though *travel* for work/biz reasons, when needed, is fine)
* NO JERKS. related: NO PARANOID bosses/employers, or, gaslighters

# Misc

* US citizen
* Colorado resident
* native US English speaker
* hobbies: gaming; wargaming; LEGO; watching football; reading sci-fi & history; writing comedy; long walks/hikes for cardio exercise & boosting endurance health

# Online Presence

* **email:** [groglogic@gmail.com](mailto:groglogic+ghport@gmail.com)
* **GitHub:** <https://github.com/mkramlich>
* **Substack:** <https://synystron.substack.com>
* **Mastodon:** <https://toot.io/synlogic>
* **HN:** <https://news.ycombinator.com/user?id=syngrog66>

UPDATED: 2024 August 1