Kenrick Rilee
240-350-0419
kenrick.rilee@gmail.com
cricklet.github.io

WORK:

Full Stack Engineer at PlanGrid (*2015-16, 16 months*)

- *Python (Flask)*:
  Worked on automating the process of downgrading users with failed payments.
  Worked on the initial version of our public API.
  Worked on our feature-flagging system.
  Rewrote and added tests to fragile sections of our payments code.
  Focused on increasing test coverage and decreasing patching in our tests.
  Gave well-received talks to all of engineering focusing on managing state (OO vs FP) and organizing code for testability (i.e. hoisting dependencies in procedural code, dependency injection for OO code).

- *Javascript (React, Node)*:
  Worked on internal tools for support & sales.

- *Docker:*
  Containerized the services on my team using docker compose such that they could be easily run in arbitrary combinations and configurations.

Software Engineer at Codecademy (*2013-14, 9 months*)

- *Rails:*
  Learned and applied TDD as it applies to both feature development & bug-fixing.

- *Javascript & HTML:*
  Worked on the 'codebits' feature which allows users to create their own websites.
  Helped implement a rebrand of the entire Codecademy website.
  Owned the ideation, prototyping, and implementation of live-coding widgets.

Co-founder of Rex/Mapsaurus (*2012-13, 15 months*)

- TigerLabs University Accelerator (summer 2012), raised an angel round of $200k (late 2012).
  Built an Android app recommendation service that was used by 60,000 people.

- *Python (Flask)*:
  Built a distributed Google Play web crawler (w/ multiple IP addresses to avoid throttling).
  Built a recommendation engine which performed a PageRank inspired BFS on crawled app-to-app relationships.
  Built a search engine which leveraged app-to-app relationships to provide results unbiased by keyword hacking.
  Performed event analytics which tied user events to data about apps our users later installed.

- *Android*:
  Designed & built a smooth interface for exploring a network of app-to-app relationships simply by swiping.
  Built a batch uninstaller which allowed users to easily uninstall multiple apps at a time.
  Built a multi-pane tablet app, leveraging my open-source library *PanesLibrary*.

ACADEMICS:

Princeton University: (BSE, Computer Science)

- Coursework: advanced graphics, graphics, networks, computer vision, operating systems, systems, algorithms & data structures, computational physics, number theory.

Eleanor Roosevelt High School Greenbelt, MD

- Valedictorian: class rank 1st out of ~800 students.

PROJECTS: (much more at cricklet.github.io)

Javascript:

- *blue.js*: an implementation of collaborative editing & operational transform, using FlowType. https://github.com/cricklet/blue.js

- *Star Command*: a toy star-ship sim, experimenting with stateless code & algebraic types. github.com/cricklet/star-command

- *Rest In Peace*: a <canvas/> game built from scratch in 48 hours for Ludum Dare 30. github.com/cricklet/ld48-rip

Android or *Java*:

- *PanesLibrary*: open-source library for creating flexible phone/tablet apps. github.com/cricklet/Android-PanesLibrary (*300 stars on GitHub*)

- *AutoWallpaper*: updates your wallpaper with images from Reddit's API. github.com/cricklet/Auto-Wallpaper-for-reddit- (*10,000 downloads*)

- *Dead Arcade*: 2D platformer, built completely from scratch. (*20,000 downloads*)

- *2nd Place Princeton Facebook Hackathon (2011)*: built an RTS game from scratch in 22 hours.
- *2nd Place Hack Princeton (2012)*: built an Android app recommendation algorithm.

C or *C/C++*:

- *Hatched*: OpenGL renderer built from scratch, implementing VSMs, SSAO, and real-time hatching through an auto-reloading shader pipeline powered by RAII, lambda closures, shared_ptr, etc.
  github.com/cricklet/Hatched

- *Wolfenstein 3D AI*: a program that beats the first level of Wolfenstein 3D by analyzing rendered pixels and spoofing input events. I worked on localization via range-finding and particle filters.

- *Advanced Graphics*: path tracer, laplacian mesh editing, image analogies.
- *Graphics*: ray tracer, mesh manipulation, shaders, openGL, etc.
- *OS*: boot-loader, kernel, scheduling, virtual memory, file system.

Other: Python, React, Mocha/Chai, FlowType, Flask, Docker