(portfolio events related to tech)

Unless otherwise stated: please assume that any code, iconography, illustrations, modeling, or design shown below was created personally and from scratch.

elmprov

Oct 2016 - Feb 2022

Controller scripting framework for live music improvisation via Bitwig

Started in 2016 and only just recently used for live improvised performance, elmprov is a secret passion of mine. It's a framework specifically for Bitwigs Controller Scripting API that extends Bitwig functionality to make it *a dream for live performance*.

In a nutshell, it simplifies controller mapping and yields new functionality around *starting/stopping recordings, queue stacking them, controlling parameters on selected tracks*, and whatever else I need.

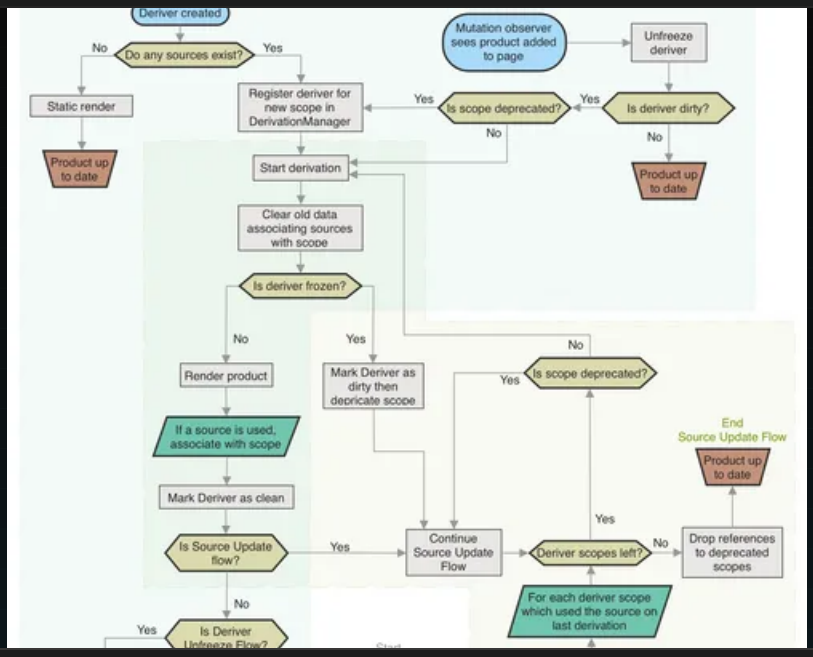
By far, *the hardest part of this project was the interaction design*. It's taken a lot of time and practice to figure out what really feels right in the groove.

Skills Used: TypeScript | APIs | Performance | UX | Audio | Bitwig

Helium-UI

Jun 2019 - Feb 2022

The tool used to make this portfolio. Similar to React or Vue, but faster, lighter, and more versatile. Still in private alpha.



Jeeze.. where to even begin? This project started with **"El-Tool"** in mid 2018. It was just a small ~200 line tool for taking a bunch of repetitive `document.createElement()` code and making it easier to read. In the end, *I liked El-Tool more than React for most use cases*, even if it didn't have states.

The next phase involved "observables," little state-like variables which had callbacks for whenever they were updated. I used these extensively in a Native-Script project, building a native mobile app. But observables got very tiresome at scale; they didn't chain

well and often created callback hells. I needed something better, and was blessed with a Eureka moment.

Helium is was arose from that. It was a pretty tiny project at first, and still is quite small (<400kb unminified with maps and definition files). But it's also definitely grown. At this point *it's the only tool I use for web development*, and it's magnificent! Having a couple of the right ideas together (I call it Source-Derivation) makes it so it's not only *more malleable* than any framework I'd used before, it's also *smaller* and *more powerful*... or at least that's what the benchmarks show.

Note: that compared to serving HTML, it is a bit slower: browsers multi-thread html like crazy, but multi-threading JS isn't so easy. This drawback effects all JS based tools.

Sometime around 2020 I broke the project into two halves: `helium-sdx` and `helium-ui`. The UI part extends sdx (Source Derivation) by adding a bunch of web specific stuff. And as of 2022 it's pretty much ready for release. *I've been using it in production for years on every project I do and rarely find things I want to add to it anymore*. But I don't feel like dedicating my life to it, and competing with Facebook seems like a nightmare so who knows if or when it might be released.

There's also a very useful sub-project for this I've been working on for a while called `helium-source-repo`. It's a tool that automatically manages pulling data objects from APIs, creating class versions of those objects, and serving them in a state-based source-derivation manner. It's hard to explain without showing, but it makes it really easy to get ahold of data, know if you should show a loading bar, edit the data, and push it back -- all without ever having to interact with any API directly or write a single callback.

Skills Used: Low-Level Web | TypeScript | Documentation | Helium UI | UX | Sketch App | Website

CTO at Dot Earth Networks

Sep 2020 - Jan 2022

Sole developer at social change start-up aiming to create collaboration between NGOs.

Heads up: due to funding issues, the project was not completed and there isn't much to show for the work.

The start of *this job was extremely serendipitous*. Boulder Dot Earth had been around for a long time, but had been lying dormant for the last few years. So Micha (CTO) got in contact with the originators to see if he could pick it up again, this time with newer tech.

In it's old form, it was a Wordpress site with event calendars for local non-profits, a directory of them, and some info on the generalized goals of the city. But *Micha wanted something a bit more tech heavy*, something with maps and a Single Page App (SPA) design. As it turned out, *I was working on a personal project with these exact properties when we met*.

My knowledge of the non-profit space was very limited, having come largely from a warless volunteer back-ground (art and firefighting). *Micha educated me on how dire the non-profit situation is*. Through his understanding of the space, and my ability to abstract and solve problems we came up with a solution.

The gist of **Dot Earth Networks** was to create *a mobile/web app that connected various loose ends together, creating a greater force for good*. The core goal was to get non-profits to *collaborate*, instead of being at war. But we also wanted to weave in *local voices, philanthropists, businesses, and government*. The exact details of how to build this

symbiotic network are a bit of a secret (as centralized political power strategies should be). But it was a good idea.

Micha secured a grant from the City of Boulder, and a collaboration with a local college to create the "Dot Earth Hive" where us and a small group of volunteers attempted to build something of *a social-good tech-startup*.

Unfortunately, while the network and tech skills were there, the money was not. I had to put in a three-month notice, which led to about four months of support trying to push it forwards. But as the only tech/business savvy participant, I was simultaneously the only person working on the actual product and heavily out-voted in regards to priorities. The appeal of the work drained tremendously.

In the end, I still love the idea and came out of the experience *enriched with a new understanding of the non-profit world*.

Skills Used: MapBox | Helium UI | TypeScript | Node JS | Servers | APIs | Web App | Stakeholder Mgmt. | Peer Leadership

Traffic Jam Nascar (48hr Game Jam)

Jun 2021

The theme of the "GMTK 2021 Game Jam" was "stuck together." Two friends and I made a game about aggressive break checking in traffic.

Created from total scratch (no framework) in less than 48 hours, the gist is: *accelerate and brake in a circle trying to cause collisions behind you*. There's a link below, if you want to see it in action.

The coding of the controls and visuals were pretty easy, with most of the difficulty coming from getting the acceleration and braking curves just right. *The hard part was the AI.* They do pretty decently, but balancing aggressiveness and risk tolerance made for a

fun and interesting challenge. I even got to watch someone do a play-through and enjoy it!

More Info

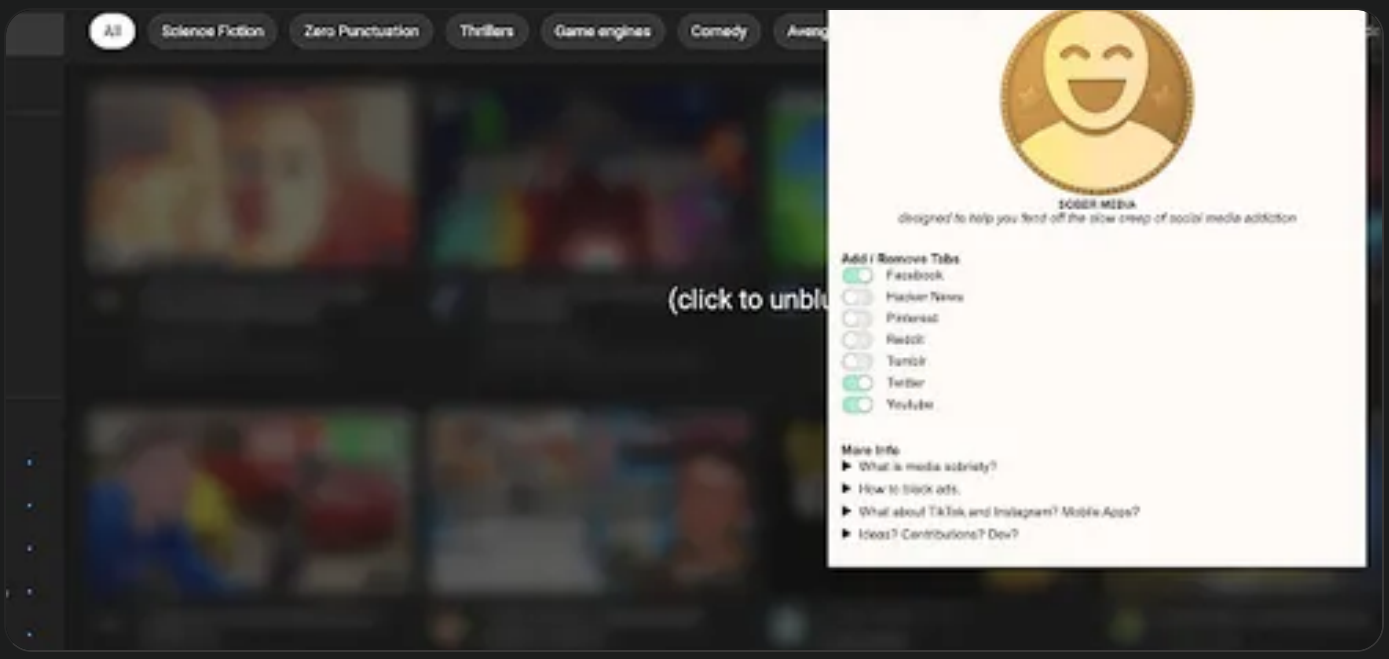
- Play the game
  https://sephreed.itch.io/traffic-jam-nascar

Skills Used: Games | Vanilla JS | HTML5 Canvas | TypeScript | UX | Sketch App | Collaboration | Peer Leadership | Task Delegation

Sober Media

Oct 2020 - Mar 2021

A chrome extension for selectively blurring coercive social media content.



Currently defunct.

This project was inspired by a philosophy around *AI driven personalized feeds: they're about as fair as playing a game of "identity chess" against the grand master*. Plausibly, future generations shall look back on our laissez-faire attitudes around these algorithms the same way we currently look at the lax public health and sanitation rules of the dark ages: *"Such an obvious vector for diseases, what did they expect!?"*

Anyways, the gist of this extension is that it *blurs out anything and everything related to user coercion*, and if you want to unblur something just click. Or -- if you don't want something blurred in the first place -- go to the settings and disable that feature. Pretty simple, but it adds *a layer of user consent where one had not previously been*.

As of today, it doesn't appear to be working anymore, which is fine. It never gained popularity, and keeping it up-to-date with the various ui-element selection logic of different social media sites is an endless endeavor.

The extensions `background.js` was built using `helium-sdx` as the state manager with live updates for every user interaction (no page refresh required). The extension popup's UI was built with `helium-ui`. The background script communicated with an

injected front-end script to decide which elements should be selected and have "blurred" css classes added, as well as event listeners for un-bluring. *It was a relatively simple and short project*.

More Info

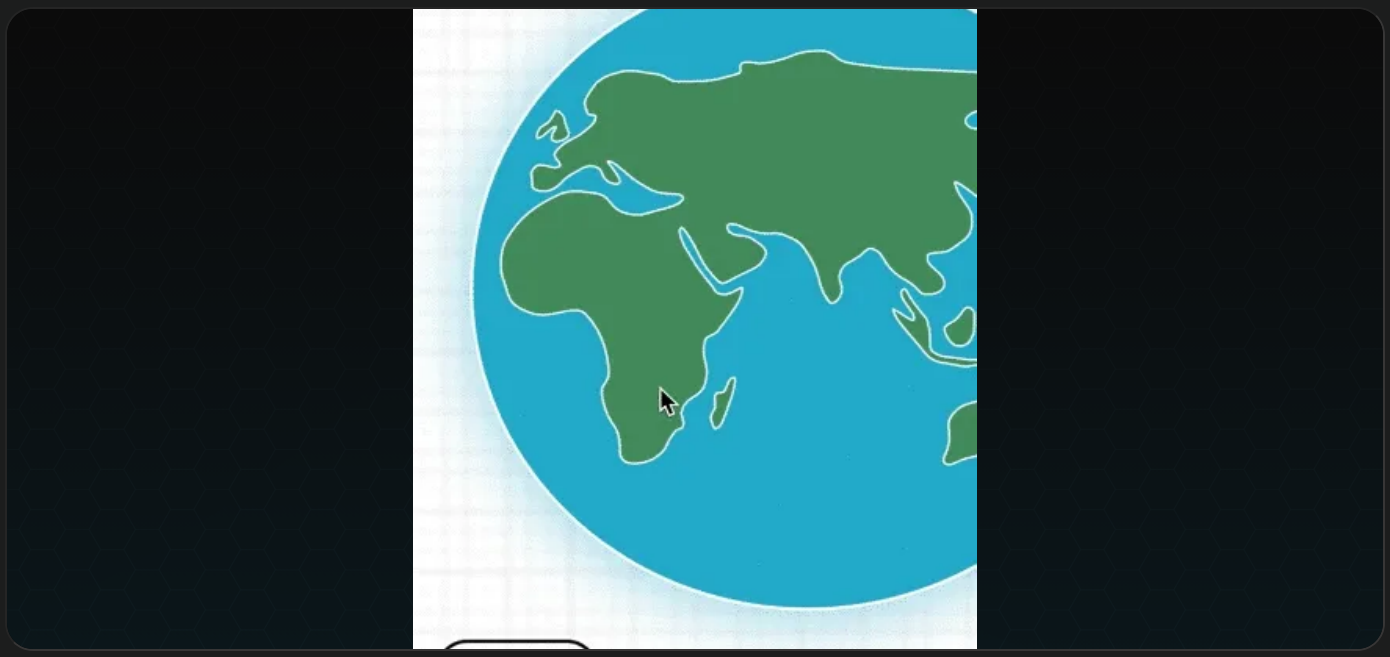
- ➤ See/download it in the Chrome Web Store
  https://chrome.google.com/webstore/detail/sober-media/hfjcloaonnidjodddmgehpinpdfndnbb

Skills Used: Web Extensions | TypeScript | Event Systems | Helium UI | Sass | UX | Sketch App

rCommunity

Jun - Aug 2020

A web platform for creating micro-economies through friends and trust-networks. Similar to FB Marketplace, but focused on ad-hoc businesses and resource sharing.



When pixel wealth came out, it felt like a call-to-action. So much of Misty and mine's income was going straight to Amazon; *what if we could buy, trade, and gift with locals instead?*

So I finally started working on rCommunity... again. *The first time I tried making this app was actually back in 2012*, but I didn't have the chops back then. The premise was pretty simple, *users create community trust networks and then advertise goods and services within them*. There's no exchange of money through the app itself (we had alternate financial models in mind), and no legal requirements for users to post within their groups.

The thing that really made this project special compared to normal online shops -- including Etsy or FB Marketplace Groups -- was the *actual community centered focus*. By putting community first, friends could *share resources and better circulate their micro-economies*. Also, I have a really dope categorization system which *uses a 20 questions approach to make perusing goods and services fun*.

The project was going really well, but I ended up getting distracted when a friend said he was starting a company and needed a CTO. I believed in the company mission, and

rCommunity only needed slight modifications to fit the bill! It was just too serendipitous to turn down.

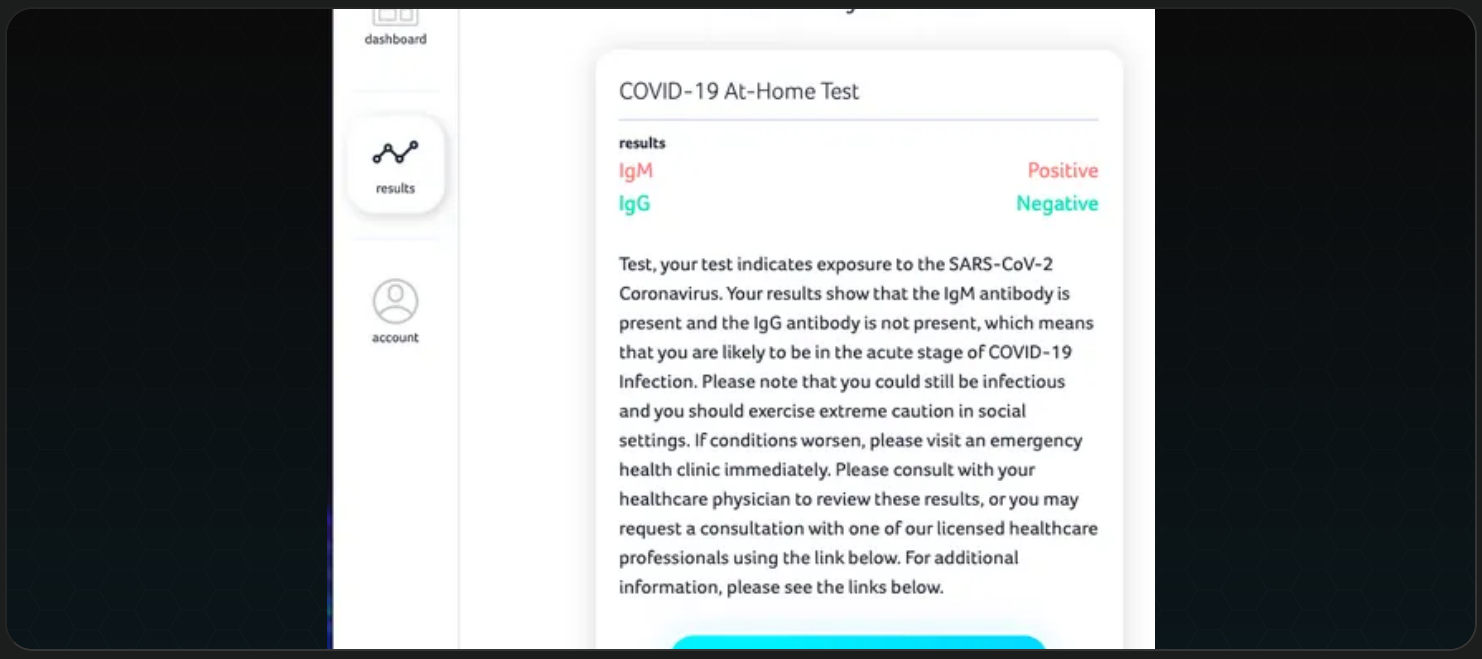
I do intend to start this project up again someday, but not without a fair chunk of change to really finance it.

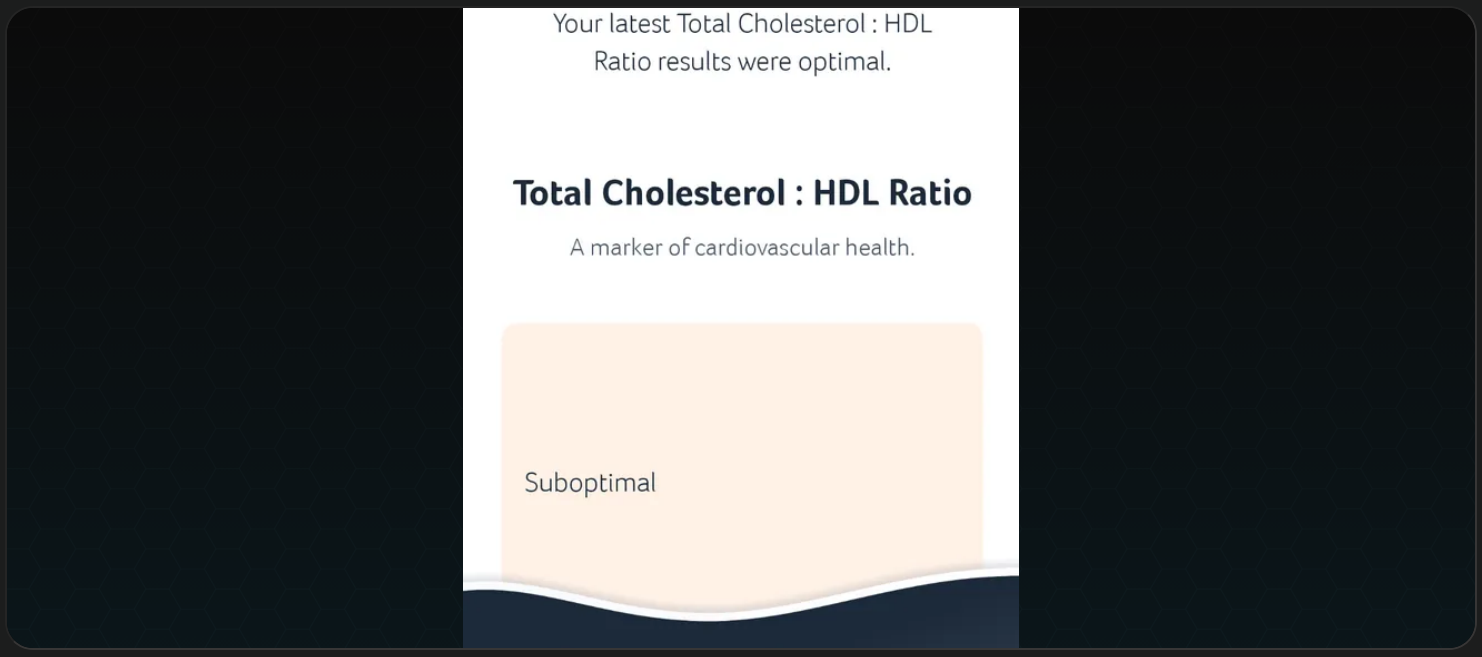
Skills Used: Web App | Helium UI | TypeScript | MapBox | APIs | Sass | Servers | UX | Sketch App

CTO at Choose Health

Mar 2019 - Jul 2020

Sole developer for a subscription based at-home blood testing start-up aiming to monitor health and give personalized recommendations.





This was the first job I ever landed through LinkedIn. Choose Health was a health-sector tech start-up birthed by **Mark** (CEO), a guy who'd long been working in the field of supplements and marketing. The impetus was pretty straight-forward: *he wanted to prove the positive health benefits of supplements*. And what better way than by tracking and measuring health?

When I was first interviewed, Mark and Sam (PO) had already spoken with physicians and ran a pilot. They wanted to prove that *blood could be used to measure health*, and

that regular measurements could add value to users lives. The results were mixed, but they saw promise and *needed a full-stack, do-everything dev to make it happen*. In the end, I developed all of the following:

- A native *mobile-app for IOS and Android* which displayed blood results and recommendations
- Multiple refactors of a product site (given to me as plain HTML, eventually made an SPA, then migrated into *Webflow* for easier PO edits)
- The entire *back-end, including HIPAA* security compliance
- Untold integrations for *payments (**Stripe**), emails (product and transactional)*, analytics, shipping and tracking, user feed-back and communications, ratings...
- *An entire separate micro-service* called "Diagnostic Infrastructure" which existed to connect doctors and in-need-of-release blood results. It was extremely asynchronous with extensive "user error" corrective measures. It had it's own front and back ends... it was a full project on its own.
- And, lastly, a *web-app* version of the mobile app

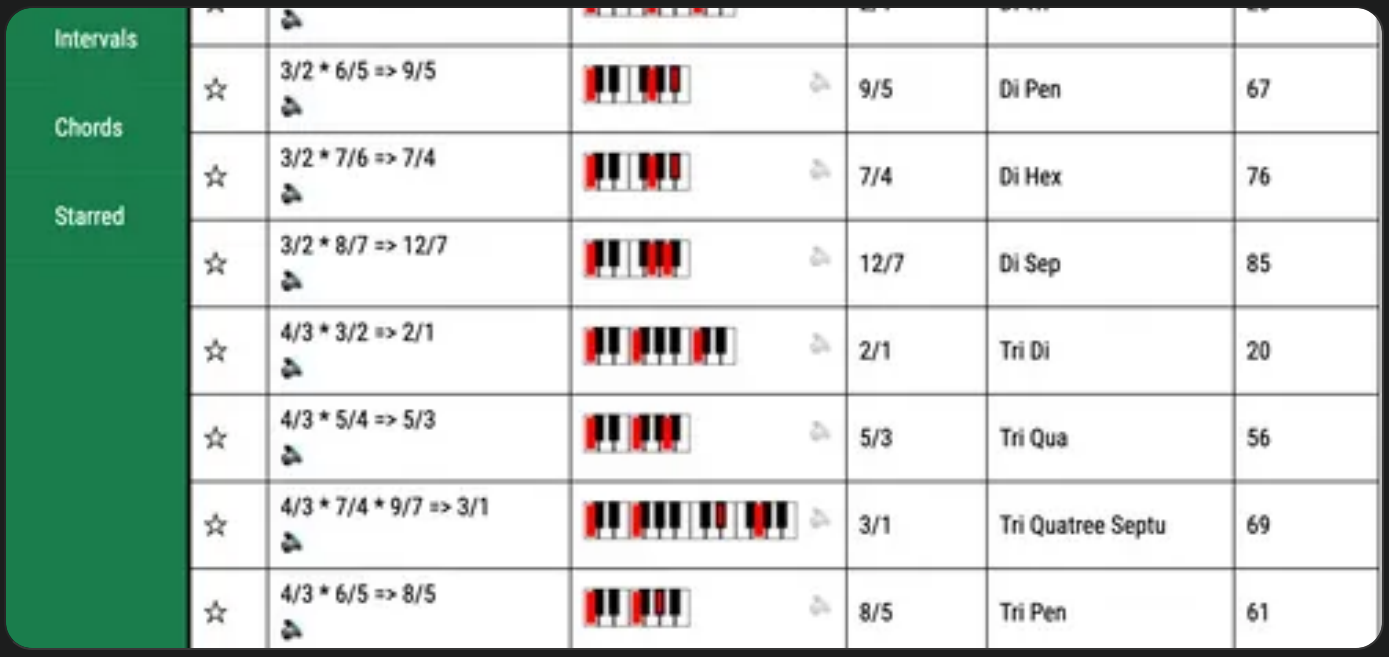
As I was approaching my 30th birthday, I took stock of my life and decided that -- while I was very happy with the quality of my work and freedom to pursue interesting solutions to problems -- I wanted to turn a new page. I put in my three month notice, *ended work on my 30th birthday then moved up into the Colorado Rocky Mountains*. I learned an absolute mountains worth while working there.

Skills Used: Mobile Apps | Micro-services | Integrations | Event Systems | Stripe | TypeScript | NativeScript | Node JS | SQL | Servers | APIs | Web App | Stakeholder Mgmt. | Roadmapping | Website

JustTunings.info

Feb - Mar 2020

A website for exploring micro-tonal music theory, with built in keyboard synth.



This project started out as a way to *visualize a micro-tonal music theory*. In particular a theory that one of the mathematic aspects of good musicality is having complex relationships that resolve well; like a major triad where `4/3 * 5/4 = 3/2`.

The first iteration of this program was just something that brute forces different combinations of a list of simple ratios. In order to have some bearing of where they were on a piano, `HTMLCanvas` was used to draw their "western approximation". Then, *in order to hear them, a synth was added*.

It was good enough to share, so I eventually made a public website for it (linked below) and added some information on the general "just intoned" music theory.

Since starting this project, **Bitwig** has added micro-pitch features which allow for experimentation with these alternative scales. Unfortunately -- as I'm not the first to find out -- *most of what people find "pleasant" in music is rooted in familiarity*. But it's still fun!

More Info

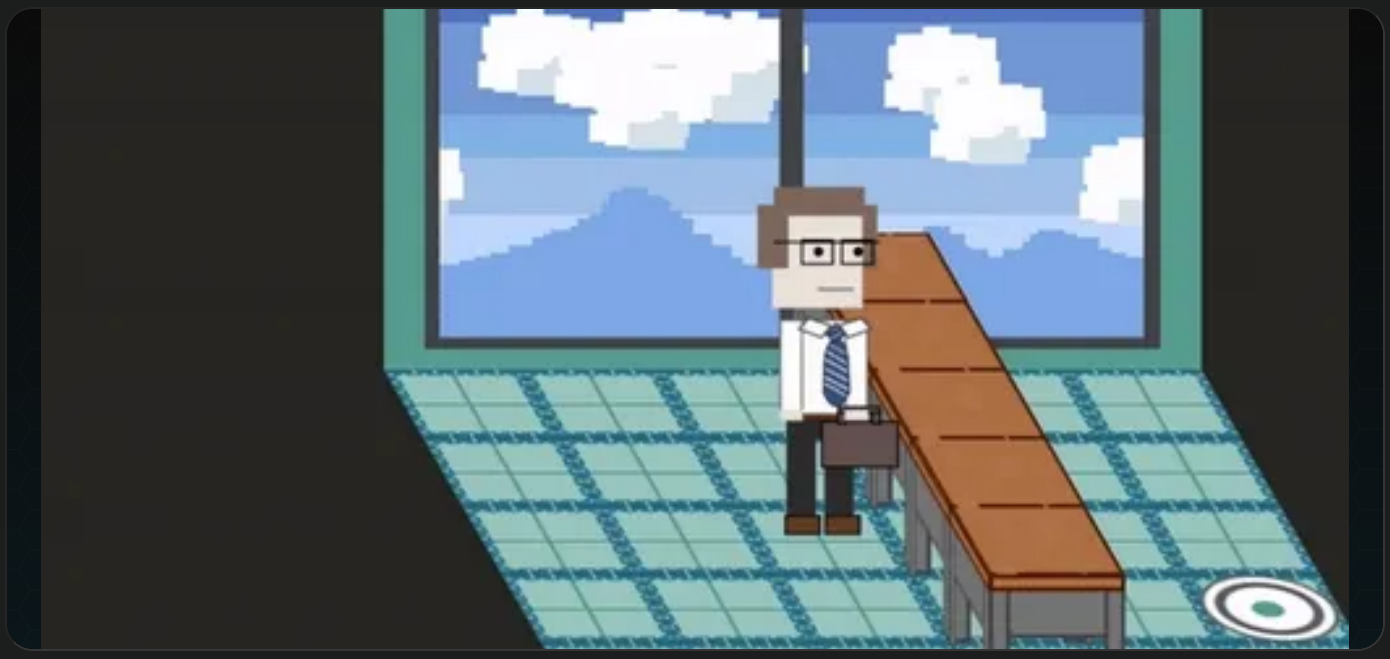
- JustTunings.info
  http://just-tunings.info/learn

Skills Used: HTML5 Canvas | Helium UI | Sass | TypeScript | Web Audio API | Website | Audio | Synths

Clock Out (48hr Game Jam)

Sep 2019

The theme of "GMTK 2020 Game Jam" was "Only One." This game let users "hack" multiple game-engine state variables a single time each to solve puzzles.



Made for the first ever GMTK Game Jam, this was also the first game I'd worked on since I was a child. The theme was "Only One," and *I really wanted to test out Helium UI's* ***source derivation*** *algorithm*. So -- using helium -- I coded a puzzle game entirely out of `HTMLElements`, where you "hacked" state variables to get to the goal.

This was a brash decision to have made, as *it yielded SO MANY BUGS*. But that was kind of the point: I made something *much more demanding than a website* and found the points of failure. At this point `helium-sdx` was just a baby, yet it was still possible to make a game with it -- a game that React simply wouldn't work for.

The game was submitted with no time to spare; it was uploaded as a compressed folder and nobody except my friends played it.

More Info

- ➤ Play the game
  http://sephreed.me/clock-out/

Skills Used: Games | Helium UI | TypeScript | Sass | Collaboration | Peer Leadership

Death Gun

May - Aug 2019

A tesla coil shooting gallery; I programmed the game and score board aspects.

Aerica is an amazing muralist; Steve an electrical engineer with a fondness for tesla coils. They're a bit of a power couple, and came up with this amazing project. But they wanted to take it a bit further by making the game more gamey: they needed a programmer.

The first part of the project involved setting up a *Raspberry Pi running Node JS and using one of the USBs as a serial port*. It took some finagling as it was new territory and there's always a lot of gotchas when working with micro-controllers.

From there the coding was breezy, taking only a few days. The *scoreboard was a website with a websocket* open to a localhost server. The same server controls the game and responds to user input. In regards to game design there was some experimentation: the system ran on asynchronous promises rather than a constant loop. For testing the game, a small mock up web version of them game was created.

The experience of the game was three levels in a branching path where each one could lead to either a greater or lesser challenge; a total of 6 levels. *Almost every player was able to complete the game, but only a few made it to secret-ish the hardest level*.

One of my favorite "pranky" aspects of the game was that there were only 6 digits for high scores, and if a player went above that it just rolled over to zero. *Many laughs were had*, and it set a new challenge: get as close to 999999 as possible without going over. The other "pranky" part of the scoreboard was that it displayed "most average scores" as well as high-scores. Players were rewarded with plenty of fanfare for getting a "most average" play-through.

More Info

- View Aerica Ravens Portfolio/Gallery
  https://www.aerica-raven.com/special-projects/deathgun

Skills Used: Node JS | Linux | Micro-controllers | Games | HTML5 Canvas | TypeScript | Interactive Art | UX | Collaboration

Tradition Churner

Dec 2018 - Mar 2019

A "Virtual Synth" VST which allowed me to finally hear a wave function I'd long dreamed of.

The idea for this synth first came to me around 2008. It was a decade later before I had the skills to make it.

The premise of this synth is that *it's possible to seamlessly shape a wave into the four standard forms* (sine, saw, square, triangle) using only two parameters. These parameters are the curviness and skew. You may be able to see what I mean in the animation above.

But just imagining the visual was not enough, it had to be heard. Alas, *this project was hard*. I leveraged the **JUCE** audio framework's solid tool-set as much as possible, but there was still tons of highly specific Digital Signal Processing (DSP) challenges. Notably,

creating the *perfect formulas for the curves* (linked below), and trying to implement an asynchronous multi-threaded "observables" system in C++ (*generics and mutexs* oh my!) were big wins. But the truly hard part was the routing system!

Any signal could be used to modulate any parameter on a per voice, per wave, or synth-wide basis. This introduced *a ton of performance complexity around how to minimize buffer usage*. Treating every signal to the "per-voice" solution would have wasted tons of resources. I mostly remember feeling more exhausted than I have with any other project.

This synth was never fully completed (art, right?), but it did get far enough to sound awesome! Since then **BitWig** has come out with their own modular synth system which is enough to generally recreate TraditionChurner. Though not as performant (sans caching), it's pretty cool to be living in the future.

More Info

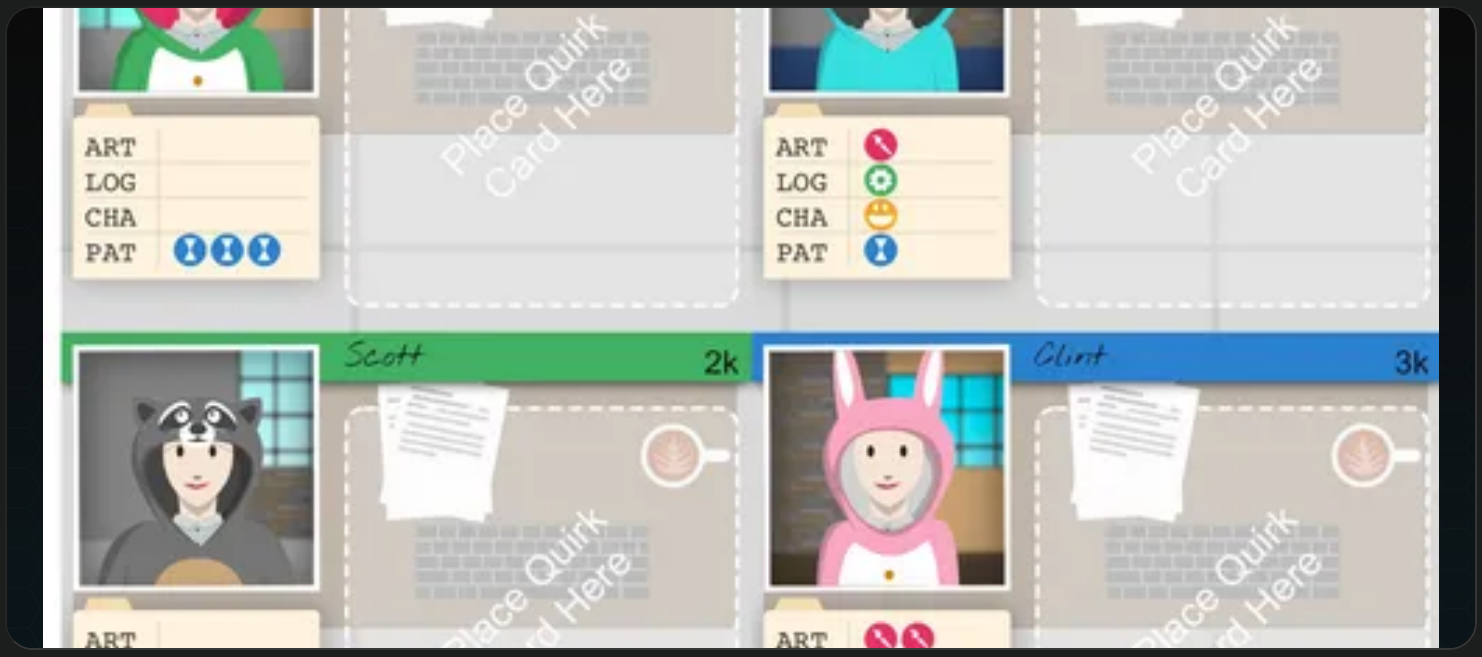
- Formula for tweening a sine curve to a square or line
  https://www.desmos.com/calculator/ac2i4rpsb4
- Output formula for curve knob
  https://www.desmos.com/calculator/zmhrbxvlju

Skills Used: C++ | APIs | Event Systems | Signal Processing | Audio | Sketch App | Synths

Synergy

Oct 2018 - Jan 2019

A tongue-in-cheek resource-management board game based on start-up culture.



reading this] "Hey boss. Sorry I won't be able to make it into work today, I've got that thing that's going around the office." Choose an employee to take 1 task from. Also take 1 task from any employees *directly adjacent* to them.

Regards,
Office

stand and not show it off. If you have any employees with the {{IMPULSIVE}} quality, they each get a blame token and you get -1SYN. Otherwise, the keg stand is dope +1SYN.

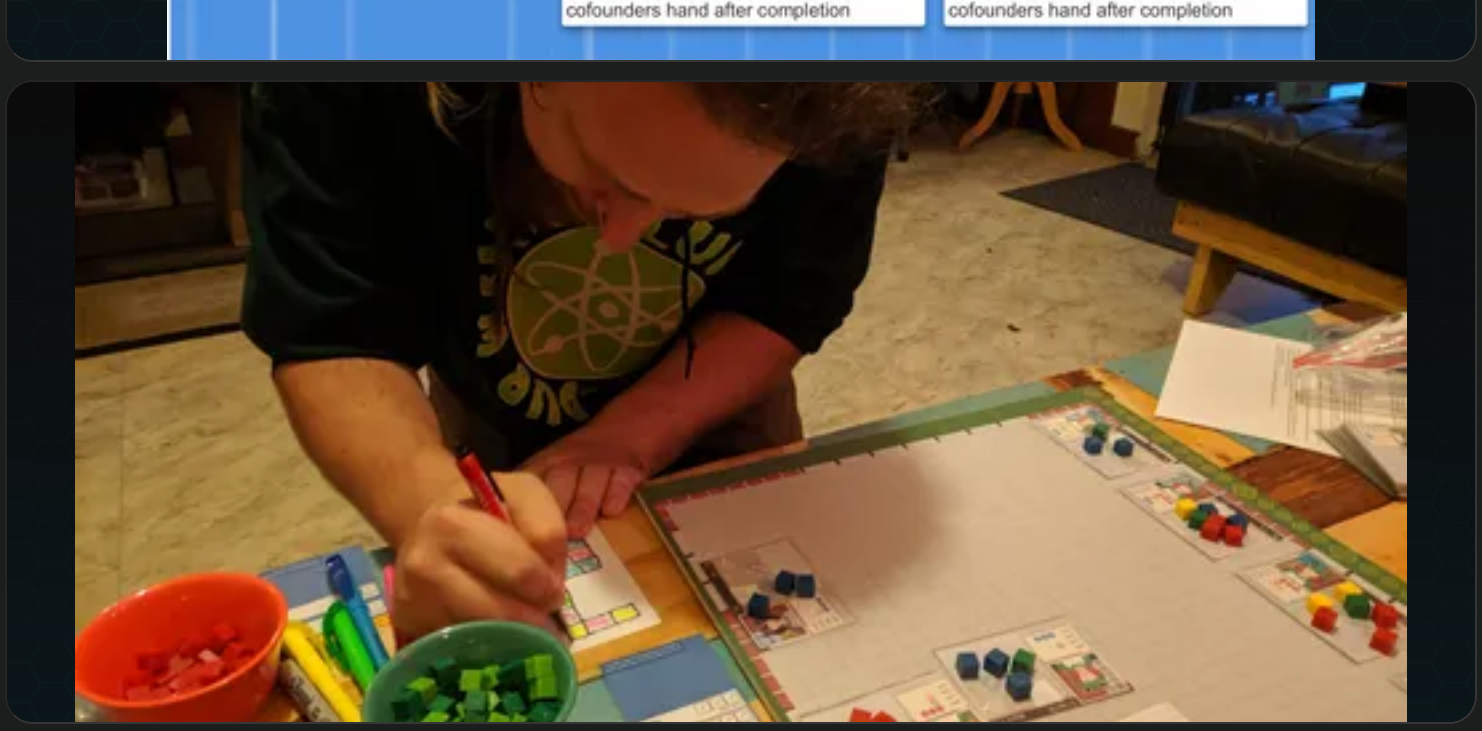
Regards,
Office

Donuts!!
from: Announcements

Someone brings donuts to the office. If you have more than one employee with {{IMPULSIVE}}, they eat all the donuts and each get a blame token. Otherwise, +1SYN.

Where the pens at?
from: Office

A signature is needed and no pens can be found. Who's job is this? If you have HR, the issue is caught in time. Otherwise, it goes on far too long for -1SYN.



Inspired by a copious amount of love for board gaming -- then mixed with the woes of job hunting and start-up culture -- Synergy was a fun and comedic respite for Misty and I. The process *started off with watching "Office Space" on loop*, trying to catch that humorous immortalization of tech life. But as time went on, the game become something more.

The basic gameplay has gone through various stages, with more to come. But the gist is: every player is a co-founder, you hire employees, upon hiring you pull quirk cards, events make the quirks come alive, you've got long term goals and short term tasks, and *Synergy represents that ever fleeting resource: employee happiness*. We'll see how much of this changes in the next iterations.

The characters were made in Sketch App, as well as the iconography and illustrations. But *the crux of this project was definitely rendering the cards*. Every single part of every type of card was defined in the form of a spreadsheet. These spreadsheets were then parsed and rendered in-browser to look like something out of Adobe Illustrator. And from there it's printable. This was a great design process as *having everything be generative from spreadsheets allowed for super fast iterations*.

We did a little bit of "blind" (no input) play-testing with our friends, cut short by me getting a new full time job. There were definitely some flaws; most notably "hierarchies"

kept emerging amongst players that led to one of them being the boss over the others. The other problems revealed were also pretty similar to real jobs. lol.

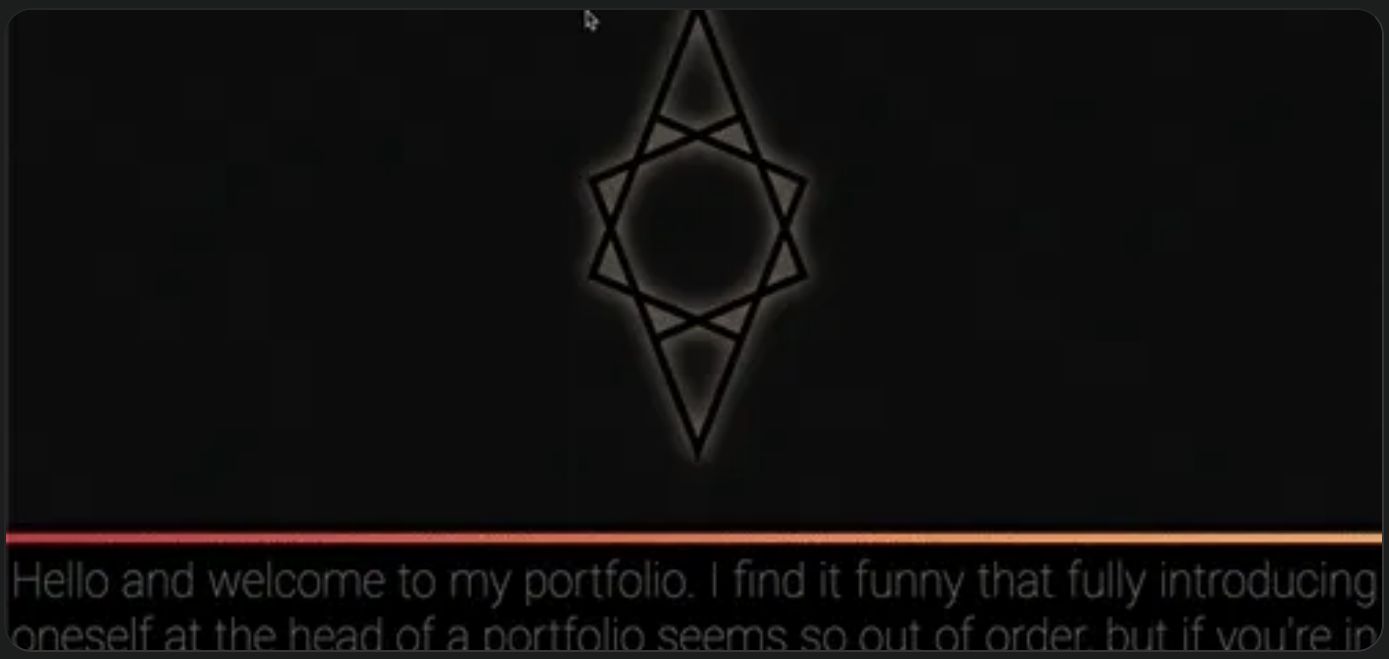
Can't wait to start this one up again later!

Skills Used: Games | Sass | TypeScript | RegEx | 2d Art | Sketch App | Collaboration

2018 Portfolio

Oct 2018

Previous portfolio, utilizing an early prototype of my web framework "Helium-UI"



The styling for my second portfolio was *based on a design session at Coder Inc where two others and myself were asked to redo the entire sites aesthetic*. We wanted high contrast, dark theme, with slender lines and hard edges. And indeed, it does come off pretty professional.

But there a few others things being experimented with here.

- Firstly: 'el-tool' (a precursor to Helium UI) was getting it's first real run,
- secondly: a grid-locked scaling system was created for UI elements, and
- thirdly: chamfered borders (not a 90 degree angle or a curve).

If you check out the old site, you'll see that there's a grid in the background which sort of glows wherever the cursor goes. And you'll also see that *all of the text and borders are aligned to the grid*. Pretty neat!

The chamfered borders were a major hassle, having to be first drawn by an `HTMLCanvas`, then exported to an image, then set to the border image of the element. Surprisingly, the hardware load was negligible for this.

I never intended to make a new portfolio, in fact this one started as merely an update...

More Info

➤ PDF version of the portfolio
http://sephreed.me/SephReed-Portfolio.pdf

Skills Used: TypeScript | Sass | Vanilla JS | HTML5 Canvas | Sketch App | Website

Front-End Developer at Coder Technologies

Jan - Sep 2018

At a tech start-up, assigned to develop a browser-embedded IDE and a CRUD site; ended up filling a bunch of other roles.

When I started at Coder there were less than a dozen employees, so I got to fill many roles besides front-end developer. Our mission was to create a collaborative cloud based software development suite, including in browser IDE and remote compilation/storage/hosting. Some of the projects I did include:

- Site wide theming, procedurally generated from Textmate/Bash themes (see Lumas)
- Many internal pages of outstanding quality (as said by the designers and QA)

- Major site components such as Button, Link, EditableText, ValidatorInput, PageTween, MinimizableDiv, and Modal
- A keyboard navigation system for the IDE
- Background/intro/outro music for company promo/tutorial vids
- Original concepting for landing page
- Some major influences in our workflow and timeline
- Head of Easter Eggs
- A few very nice blog posts

More Info

- Coder.com *
  http://coder.com
- Watch product video I made the music for
  https://www.youtube.com/watch?v=CMKRbc8DpVs
- Link to song from video above
  https://soundcloud.com/thumbz/forestbackground

Skills Used: TypeScript | Sass | React | Node JS | Vanilla JS | Documentation | Web App | Audio | Bitwig

Lumas

Jul 2018

A consistently readable UI pallette generator, meant to turn IDE themes into full site themes.

Lumas was a custom theming tool designed (while at Coder) to be able to *take arbitrary color palettes and turn them into readable, consistent UIs*. It does this by mixing and matching color gradients and luminosity mappings. The video below does a great job of showing these steps it takes when creating a theme:

1. First, organize the random colors of a set by luminosity (a function of human perception that takes some work to get right).
2. Then, make a gradient from black (0% luminosity) through the colors (in order of luminosity) to white (100% luminosity).
3. Finally, use luminosity mapped css variables such as foreground, background, faded text, and border color to specific luminosity positions within the gradient.

By keeping luminosity constant, contrast remains stable (regardless of changes to overall hue or saturation), and contrast is the most important part of any UI. Calculating and setting luminosity is much more difficult than Value in HSV or Lightness in HSL color spaces.

More Info

- Screencap of Lumas demo
  https://www.youtube.com/watch?v=47XC0WGpAQU

- * CIELAB color space
  https://en.wikipedia.org/wiki/CIELAB_color_space#CIELAB%E2%80%93CIEXYZ_conversions

Skills Used: Procedural Gen. | TypeScript | UX | 2d Art

El-tool

Jun 2018

Syntactic sugar for creating HTMLElements, turned out to be extremely performant.

```
    innards: [
      anchor("TopbarLink", { href: "here.com" }, "Here"),
      anchor("TopbarLink", { href: "there.com" }, "There"),
      anchor("TopbarLink", { href: "elsewhere.com" }, "Elsewhere"),
    ]
  }),
  div("MainContent", [
    div("Route", {
      ref: (ref) => routeState.observe((route) => {
        let content: HTMLElement;
        switch (route) {
          case "main": content = div("Main"); break;
          case "about": content = div("About"); break;
        }
        setInnards(ref, content);
      })
    }),
    span("ToTop", {
```

El-tool is the project which eventually flourished into `helium-ui`. I started developing for Coder IDE code-base. It *was basically just syntactic sugar to reduce the boilerplate for creating HTMLElements*... but then (on my birthday) I decided to do some benchmarks. I compared and contrasted various ways (React, Vue, native) of generating a few thousand divs with random numbers as text and found that the only way to render a DOM faster was with raw (and non-interactive) HTML. Basically, *I'd stumbled upon what was -- perhaps -- the fastest possible way to render an interactive DOM*. Hard to believe, but here's the two key factors:

1. It's faster to use `document.createElement("div")` to create an element than it is to use html strings (React)
    - this is true whether you create elements one at a time, or create an HTML string for thousands of divs
2. Any loop or function that gets called a lot (in V8 engine) will become "hot"
    - "hot" code is optimized for its common use, with look-ahead
    - also, as a side effect of "temporal locality" (used by RAM/CPU recently), "hot" code has reduced fetch time

Pretty neat! But, much *like default React, El-Tool did not have any useful state management system*. Because of that, I wouldn't consider it a true competitor to any

widely used tools. It wasn't *until it evolved into Helium* that it became something truly comparable.

More Info

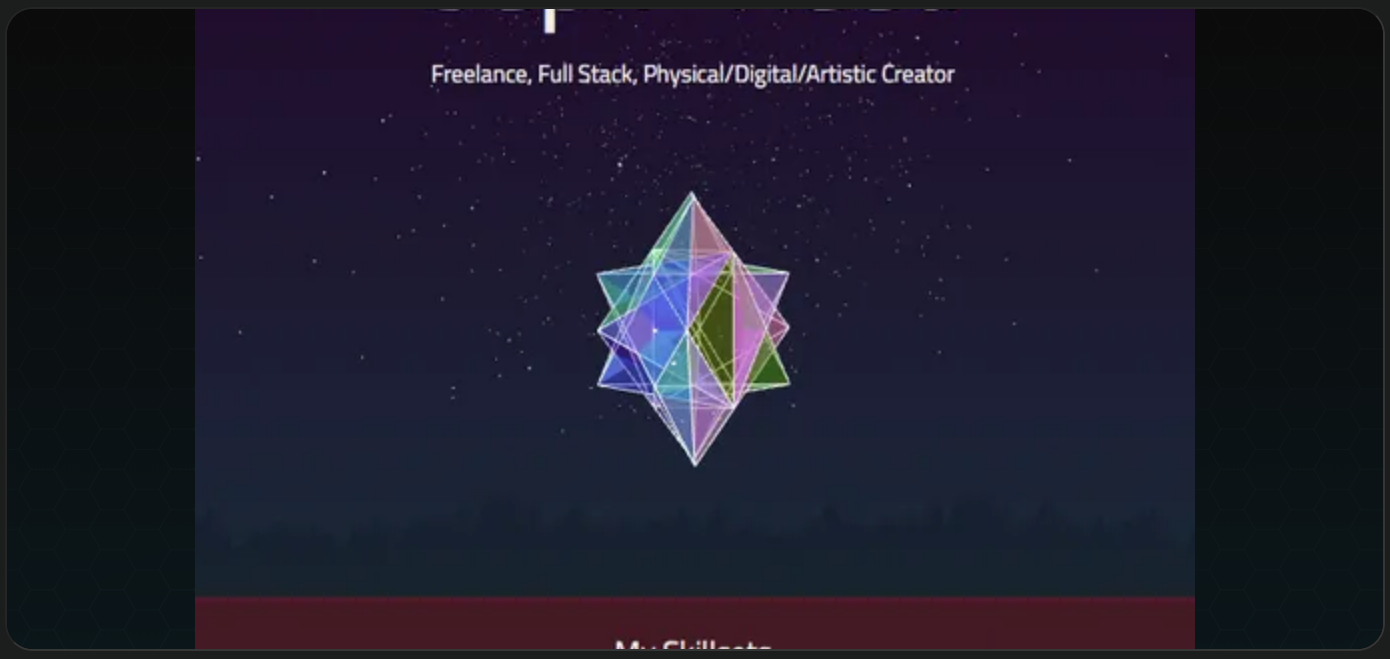
- Npm package page
  https://www.npmjs.com/package/el-tool
- * The fastest/slowest ways to append elements to DOM
  https://howchoo.com/g/mmuOnguznjg/learn-the-slow-and-fast-way-to-append-elements-to-the-dom
- ** Description of temporal/spacial locality
  https://stackoverflow.com/questions/7638932/what-is-locality-of-reference

Skills Used: Vanilla JS | Low-Level Web | UX

2015 Portfolio

Dec 2015 - Jan 2018

My first portfolio, using my first custom (Vue-like) web framework "PINE" and some 3d animations.



Even though I'd been programming since age 10, and majored in it in college, at age 24 I'd still never worked a tech job. No regrets there, I was adventuring and being young! But *in 2014 I decided to maybe try out a bigger paycheck, and this was my first portfolio.*

It was built on my first web framework (called PINE), and had quite a bit of interactive qualities. In 2018 I made a new portfolio that was "more professional", but now (in 2022) I'm returning to my roots. This portfolio before you is very much inspired by my very first one.

More Info

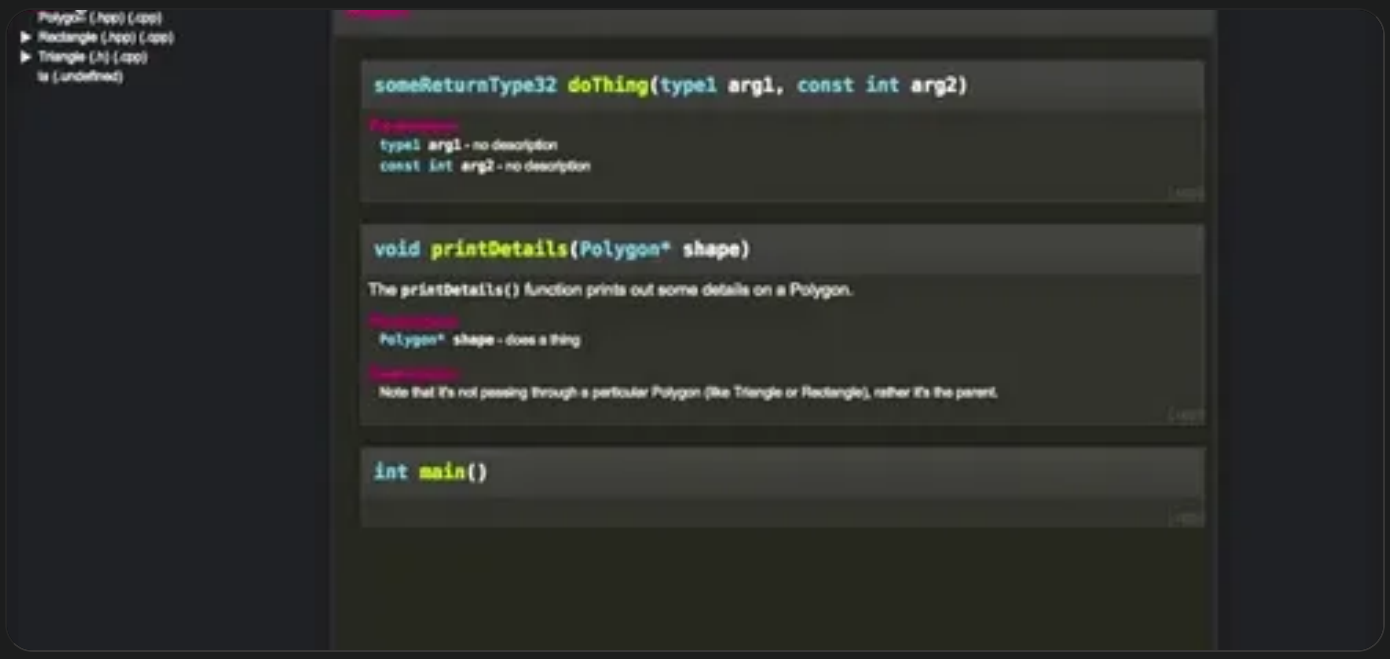
- Check out my github.io portfolio
  http://sephreed.github.io/portfolio/index.html

Skills Used: Vanilla JS | Three JS | HTML5 Canvas | Website

Docurizer

Oct 2017

A documentation generator built to dissect an open-source C++ project that had no docs.



In order *to make sense out of an undocumented open source C++ project*, I built these two tools to parse the project and *automatically generate docs based off any comments it could find laying around*. A year later, I discovered that when I created Scopifier I'd inadvertently reinvented grammars. As for Docurizer, it takes a symbolic tree generated by applying a grammar, reorganizes the information, then renders it as html.

More Info

- View Docurizers rendered output (try inspecting elements)
  http://sephreed.github.io/portfolio/Docurizer_Render/index.html?file=main.cpp.html&ov=inc
- View C++ grammar for Scopifier
  https://github.com/SephReed/Servers/blob/5e7984405e47587f234679b8bc0e114824e7da13/Docurizer/li
- View project source code
  https://github.com/SephReed/Servers/tree/master/Docurizer

Skills Used: Node JS | Syntax Parsing | C++ | RegEx | Documentation

FLAT.World

Jul - Sep 2017

My first attempted browser-based 2d game maker, inspired by the application I first learned to program in: RPG Maker 2000.

There's a soft spot in my heart for 2d games, and *RGP-Maker 2000 was what got me into programming when I was ten*. A decade and a half later, I decided to look back into it and -- to my dismay -- it wasn't what it used to be at all anymore. Plus it was only for windows.

The goal of this project was to recreate that childhood tool, in browser, and make it free for the world. I ended up setting this project aside as I simultaneously started my full time job at Coder, and began planning for the PINE Cononagon.

More Info

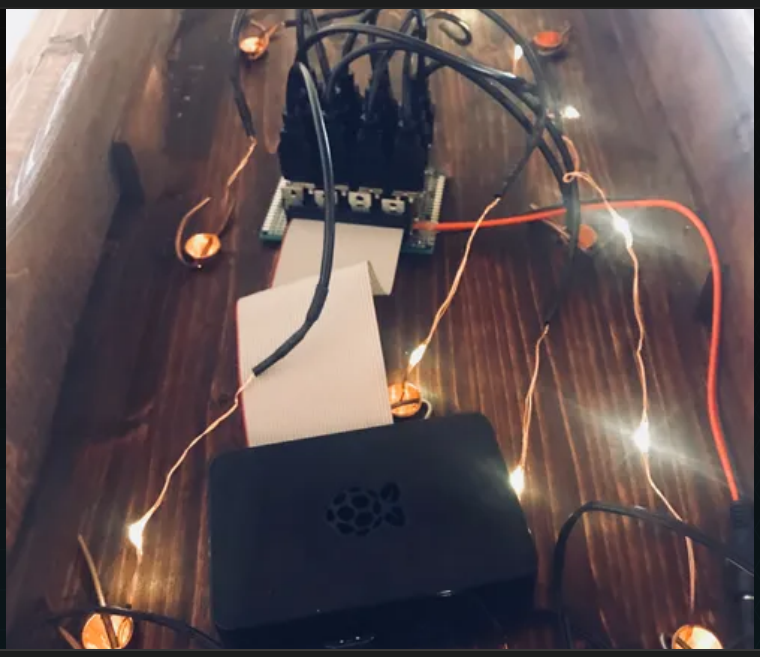
- Play with live demo (no server access)
  http://sephreed.github.io/FlatStory/index.html
- View source code
  https://github.com/SephReed/SephReed.github.io/tree/master/FlatStory

Skills Used: Web App | Node JS | Servers | HTML5 Canvas

Independent Contractor for Project "Dad Light"

Jan - Aug 2017

A custom built audio-reactive ceiling light fixture, with an app for control.



Sally Hall was just finishing up her classes at Austin Center for Design and had a wonderful idea of a gift to make her father. The basic premise was this: *Her father was an audiologist, so she wanted to make him a ceiling light which responded to sound* similar to the way the hairs in our ears do while playing music.

She spread this idea broadly, looking for someone who could make it happen (on a college student's budget), and found me. We met over coffee and discussed its feasibility, then I used sketchup to design the project alongside her. What we came up with was a seamless box containing *a raspberry pi, break-out board, and power supply that would*

hang from the ceiling. Extending out of this box was twelve hanging jar lamps with fairy lights inside, each lamp representing a tone from the western 12 tone scale.

In order to make the lights play music, I developed a webapp which was served from the pi, in which Sally could choose songs, run patterns, turn on mic input, and control things like brightness, on/off, and fade time. The *hardest parts of this project were getting the FFT to be performant, and designing the 12 female-USBs breakout board* which had to use MOSFETs to step up the pi's 3v3 GPIOs to the 5v power supply for the lights.

Skills Used: Web Workers | Linux | Micro-controllers | Node JS | Signal Processing | Mobile Apps | Web Audio API | Fabrication | UX

Robot Soul

Oct 2016 - Jan 2017

An in-browser modular-synth inspired DAW (Digital Audio Workstation).

Upset with limitations that no longer exist in modern DAWs (namely, microtonal arrangement), I decided to try my hand at making my own. Inspired by the DAW "Reason," *this project was doomed from the start*. No matter how cool the system for arranging and connecting sources was, `Web Audio API` is just not meant for this level of processing.

Skills Used: Web App | Node JS | HTML5 Canvas | Web Audio API | Synths

Web Developer at Stash Crypto Inc

Oct 2016

Sole developer for various web services related to a crypto cold-wallet; also did some 3d product modeling and renders.

I met the CEO of Stash Crypto at a crypto-currency meetup held at the Factom offices, and we hit it off pretty well. He happened to be in need of a web developer at the time. The company direction was often tumultuous, leading ultimately to the disposal of the company identity in an attempt to mimic Apple (a common story in start-ups). I made a 3d model of our original logo and our flagship product, as well as a website for an internet alliance they were attempting to start, prototypes for the digital identity search engine (with AES-EBC encrypted local storage of searches) and the product landing page (which

includes the product 3d renders). While working there, I started to really study security and cryptology, specifically through the cryptopals challenges.

More Info

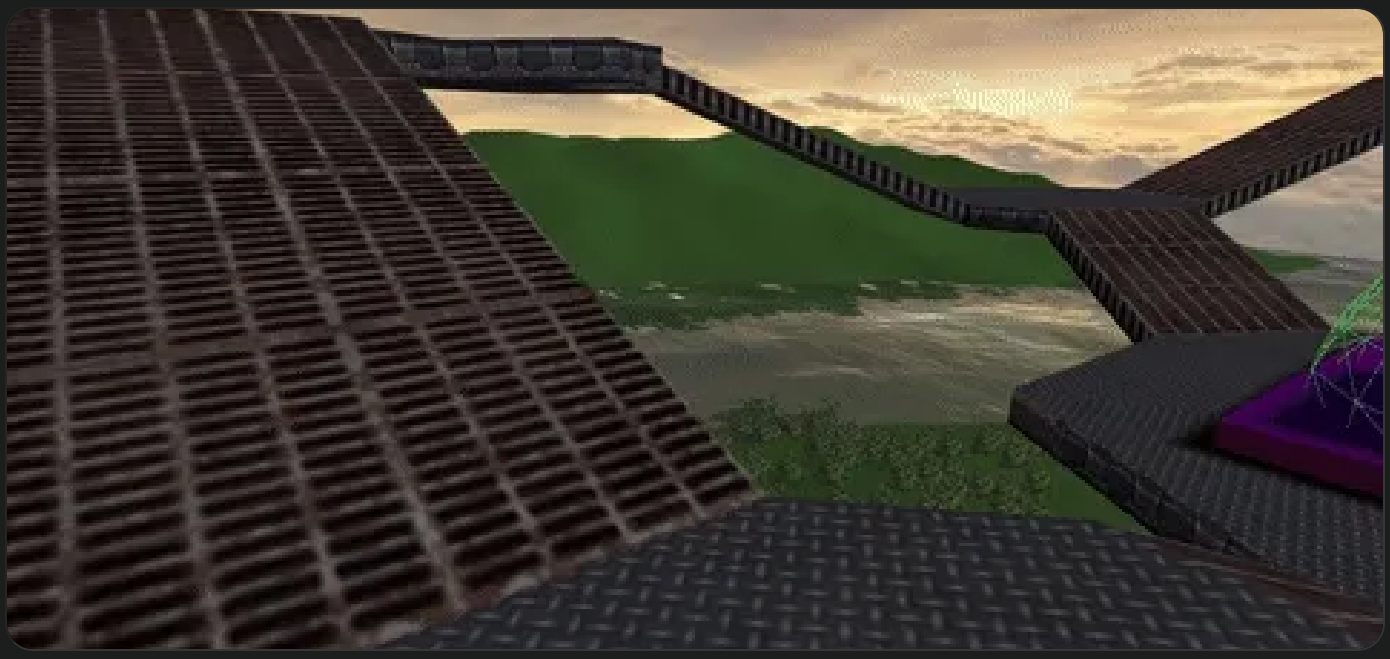
- StashCrypto.com
  https://stashcrypto.com
- View product page prototype
  http://sephreed.github.io/stash_node/index.html
- View digital identity search engine prototype
  http://sephreed.github.io/Digiden/index.html
- Cryptopals crypto challenges
  https://cryptopals.com/

Skills Used: Vanilla JS | 3d Modeling | Blender | Website

3d Walkscape

Oct 2016

My first time messing with webgl, specifically Three js. I was attempting to build an explorable scenic cabin in the mountains (as of 2022 I live in one!).



This was a fairly simple project, testing out Three.js to see what it was capable of. The answer is quite a lot for anything in a browser. The project has some very rough physics (the collisions are done with raycasting), perlin noise procedurally generated hills (no random seed), and a few interactive elements (the door to the house and the sphere). I'd like to get back into this project some day.

More Info

- Play with live demo
  http://sephreed.github.io/walkscape/index.html
- Watch screencap
  https://www.youtube.com/watch?v=Cu_6l9h4BIs&t=30s

Skills Used: Three JS | Procedural Gen. | Games | 3d Modeling | Blender | SketchUp

Baby Fingers

Mar - Jun 2016

An interactive music project which used tonal theory to transform the haphazard interactions of users into a ever changing soundscape.

A good friend was building an "Effigy" (large wooden structure to be burned), and wanted an interactive "baby music" element to it. The idea was that the fingers would have paddles as fingernails that would play sounds. He wanted a sort of xylophone/music box sound to it. I ended up taking it a step further.

The final design did play xylophone sounds, but the tones were algorithmically decided to take what would have been an obnoxious and repetitive ambience and turn it into a constantly evolving one. The basic gist of the program was as follows:

- Every 30 seconds, choose a new key from major, minor, or harmonic minor (rare)
- Every 10 seconds, choose a random chord from the key
- Every 5 seconds, choose 8 random notes from a stacking of 3 octaves worth of the current chord, then map those in order to each finger

Essentially, every 5 seconds the notes would move some small amount to either accentuate the same chord a bit differently, or move to a new chord or key entirely. There was also one last rule:

- If ever the baby isn't played with for over an hour, start a never ending loop of wailing

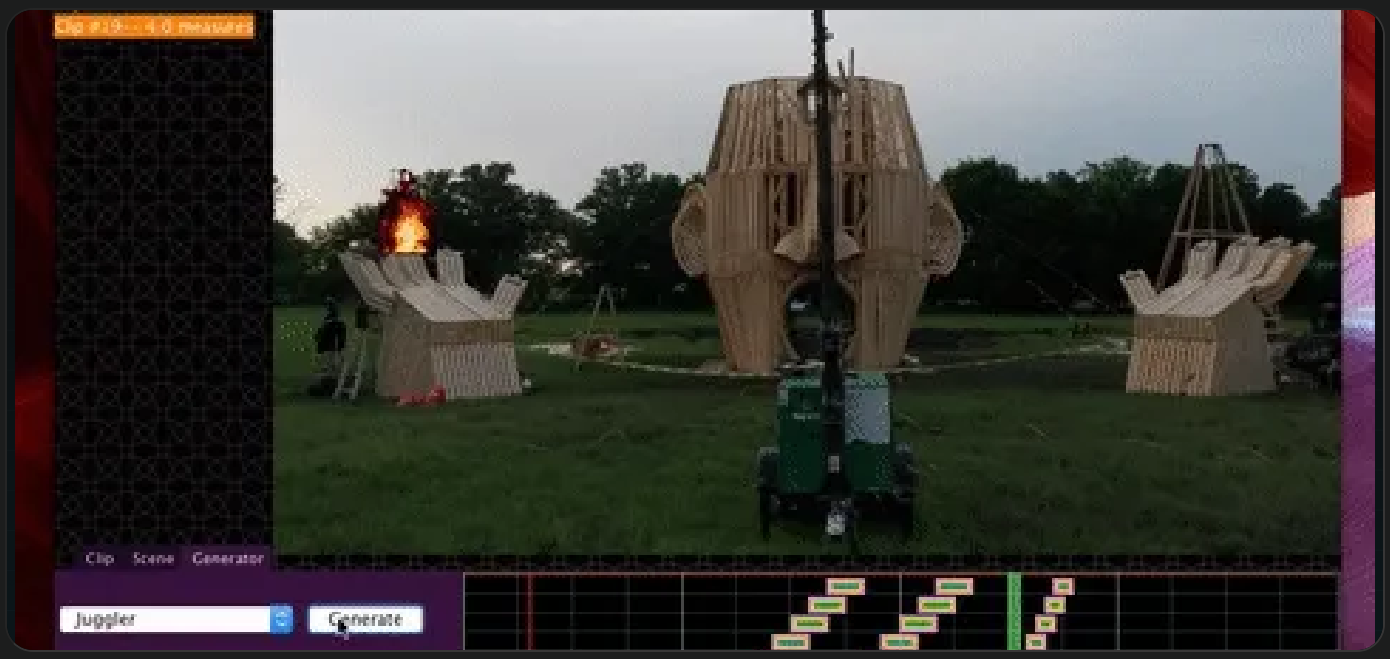
This last rule led to some magnificent prank moments, only ever happening twice over the 4 days; both times around 6am. Other than that, people were playing with it constantly, and it sounded great!

Skills Used: Python | Micro-controllers | Linux | Micro-controllers | Procedural Gen. | Audio | Interactive Art | UX | Jazz Theory

Effigy Flame Effects Sequencer

Mar - Aug 2015

Tool for producing and simulating flame effect patterns.



This was my introduction to an Austin community I came to love very dearly: the Flipsiders. The very first project I stumbled into was a four story wizard head, brimming with fun-house style creativity. And out front of it was a pair of hands, facing up towards the sky, with flame-throwers in each finger and palm.

They needed a tech crew to sequence these flame effects, so I took on the role of making the sequences. I was young and excited, so I made an entire Java program for it... with an embarrassing amount of custom UI elements. Many of them were pulled from early projects making sequencers, but still.

The way the program worked was very similar to any other sequencer (seriously, I could have just used a simple midi program), the only major feature being that it showed a simulation of what it might look like and had special tools for generating patterns. It even had its own proprietary file extension for saves, a .feg (eff-i-gy).

In the end, extreme weather conditions (25ft flooding) made the event into a bit of a nightmare for art projects, and the sequences never really saw the light of day.

More Info

- Video of the sequencer controlling leds
  https://www.youtube.com/watch?v=28CqNBAPzF0

- A full tour of the program
  https://www.youtube.com/watch?v=GQMoaTI1MQE

Skills Used: Java