Magic: The Graphening

Process Journal

Gus Wezerek

May 5, 2015

Table of contents

Week 1
A new direction

Original pitch

My original pitch for the project was a map. Users could search for their zip code and find the USDA Hardiness Zone where they lived, along with a list of plants that would thrive there. There were some additional features such as heat, rainfall and temperature extreme charts, but I was never too attached to the idea. When a week had gone by and I hadn't been able to re-project the map, let alone scrape the data I needed for the plant dataset, I decided it was time to write a second proposal.

Second pitch

The second idea was for an app that lets users compare and explore Magic: The Gathering (MTG) cards. I tend to shy away from exploratory interfaces, preferring a tight narrative that helps me understand why I should care. That being said, Magic cards have tons of dimensions: mana cost, power, toughness, color, rarity, set name, type, supertype, etc. I had hypotheses about patterns that I would find, based on my experience discovering Magic a few years ago, but I didn't know enough to start with the story.

User problem

The app I intend to build is very much a curio. That is, if you know nothing about MTG then you probably won't get very much out of it. Iif you're a diehard player then many of the graphs will just confirm mechanics and flavor that you already know. My audience are the players who are just starting to play the game or who played a few years back and want to see what had happened since. These are the kind of players who might be interested in seeing the most powerful cards in new sets or browsing the card art and flavor text from sets 10 years ago. But that behavior is hard to hook onto. It's like discovering an old friend on social media--you might go through their photos for a few minutes, but an app dedicated to reconnecting with that one friend every three months would be a hard sell. I hope that people "play around," remember why they loved MTG and share the link, but I don't think it's the kind of site that people will bookmark.

Competition

More to that point, there are a number of popular MTG-related sites that exist outside of the official Wizards of the Coast products. Here is a few tasks surrounding MTG that I identified when I was thinking about this app, along with the sites that aim to solve those tasks.

Learn about cards in a soon-to-be-released set
http://mythicspoiler.com/

Learn about Magic lore
http://archive.wizards.com/Magic/Multiverse/Planeswalkers.aspx?x=mtg/multiverse/Planeswalkers/tibalt

Learn about Magic tournaments/strategy
http://www.starcitygames.com/tags/Premium~Select/

Build a deck from scratch
http://tappedout.net/

Modify an existing deck archetype
http://magiccards.info/

Learn what cards have gone up in price recently
http://mtg.dawnglare.com/?p=viz

Learn about a card's price history
http://shop.tcgplayer.com/magic/dragons-of-tarkir/dragonlord-atarka

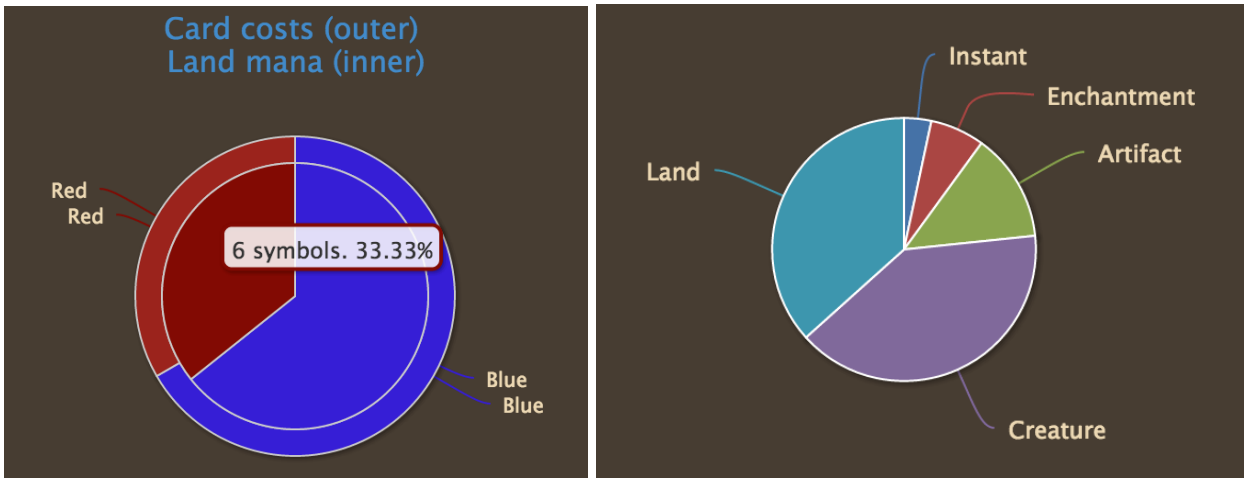
Learn prices of card collection
http://www.mtgcards.com/buylist/magic_singles-zendikar_block-zendikar/222
http://www.starcitygames.com/buylist/

Sell cards
http://www.mtgcards.com/buylist/magic_singles-zendikar_block-zendikar/222

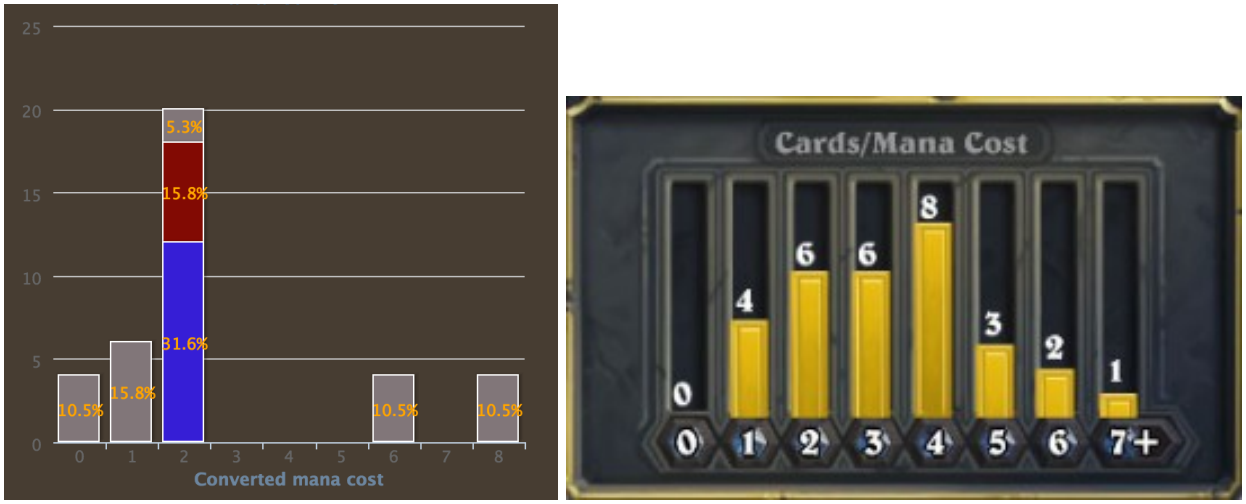
One of those tasks, pricing a card collection, is not fully served by its corresponding app (the UI is clunky and the task takes more time than it should). But my improved solution would require a whole lot of computer vision and not much data visualization, so I scrapped the idea.

Three of the tasks above use visualization to achieve their goals.

Tappedout.net lets users see the mana curve, color and type breakdown for a user's list of cards. TappedOut's focus is more on decks than sets. I suspect looking at decks is more interesting and useful, but a) deck info is hard to get (TappedOut has a large enough userbase to receive user submissions) and b) that task is already being solved. But the mana curve as a bar graph is an idea that's familiar to trading card game players, and the other two graphs reaffirm my assumption that color and type breakdowns will be interesting to users, even if a pie chart isn't the best way to convey that information.



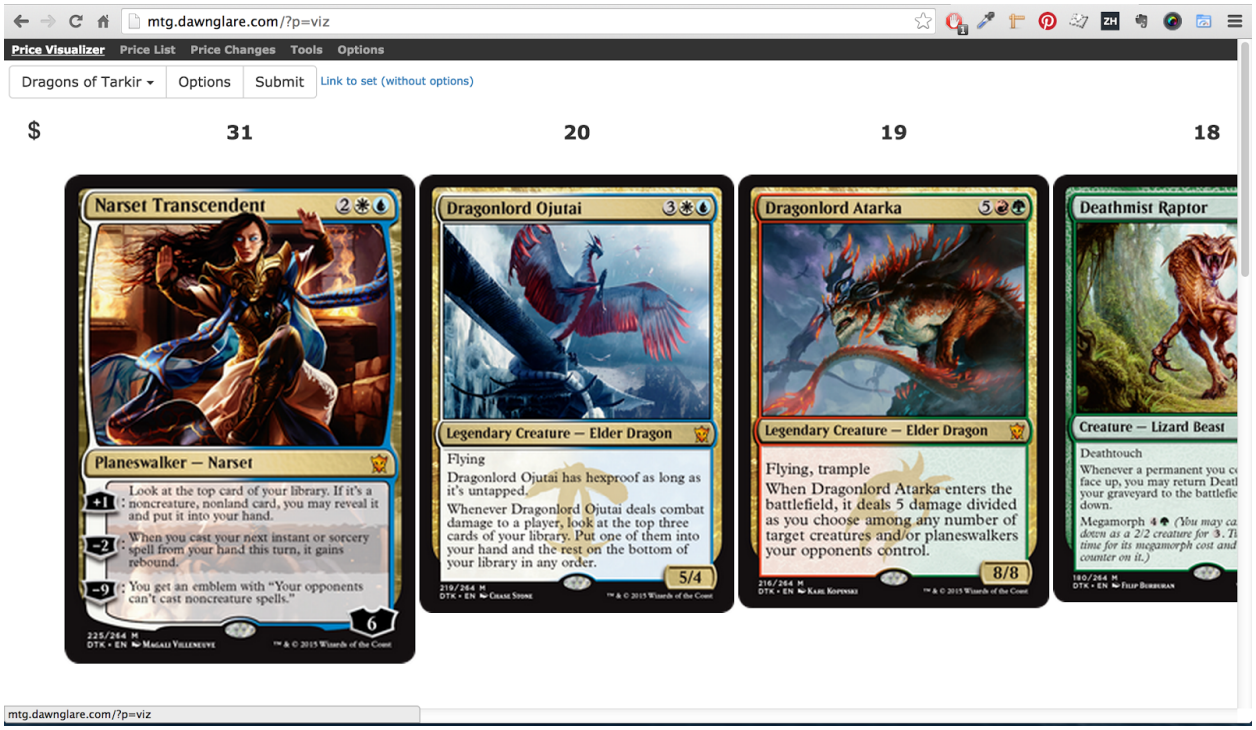
Left: Composition by color on tappedout.net; right: composition by type



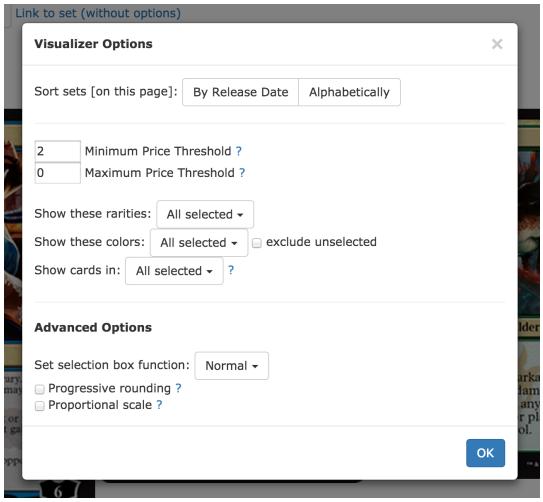
Left: Mana distribution ("curve") on tappedout.net; right: mana distribution chart in Hearthstone, a similar trading card game.

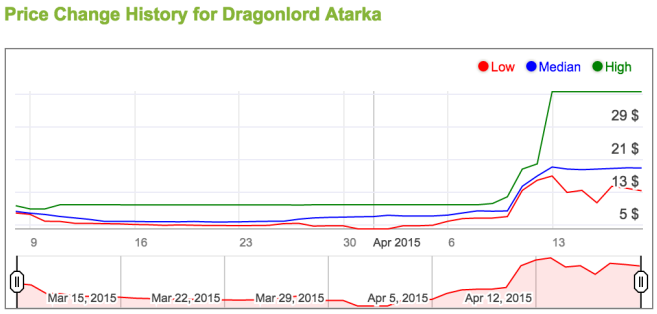
Tappedout.net lets users see the mana curve, color and type breakdown for a user's list of cards. TappedOut's focus is more on decks than sets. I suspect looking at decks is more interesting and useful, but a) deck info is hard to get (TappedOut has a large enough userbase to receive user submissions) and b) that task is already being solved. But the mana curve as a bar graph is an idea that's familiar to trading card game players, and the other two graphs reaffirm my assumption that color and type breakdowns will be interesting to users, even if a pie chart isn't the best way to convey that information.

MTG Dawnglare is a revivification of an older site that used card images and sized them to encode price data. The default scale isn't linear, as there is a desire to keep text legible for all cards, even when the most expensive card is $50 and the cheapest is $2. I'm not sure if there's a name for this semi-encoding to support design. Metacritic.com does it too with their bar charts that never drop so low as to obscure the movie poster that's embedded in each bar. But I digress. Based on a reddit thread hailing its creation, I suspect the site is popular and the balance of data and legibility is successful. Despite its inaccuracy, I think the merging of the entity and the encoding is elegant and works. And if you are interested in seeing the pure encoding, you can do so via an (easy-to-miss) filter modal.



The price visualizer at mtg.dawnglare.com. Defaulting to a filter showing the most recent set is a smart idea editorially and performance-wise.





Left: Filter modal on mtg.dawnglare.com; right: price time series on shop.tcgplayer.com. Idea for the future: A stacked area chart or streamgraph showing the total value of all MTG cards over the past five years. Would be able to see if the demand in dollars matches other popularity growth indicators such as attendance at qualifying tournaments.

Price is easily the most interesting dimension when it comes to MTG. It's a proxy for value that rarity, mana cost, power or toughness alone--or even in concert--can't match.

Dawnglare partners with TCGPlayer for it's price data, and even then only has access to snapshots. I was unable to find an API that provided full price history for free. I've put in a request with TCGPlayer for a free API token, but I haven't heard back. And given the scatterplot that I'm using to visualize the other dimensions, I'd need to be able to request the most recent price for every card at once, which would be a lot of server requests using TCGPlayer's API. So the price dimension for this project is off-limits. If I feel good about this project post-launch and TCGPlayer gives me a token, it's likely I'll refactor and redesign to feature/default to displaying that data.

Also of note: There is one very dedicated designer who puts together an infographic for each release (http://www.visualizingmagic.com/), but I don't think we're direct competitors given the interactive, historical nature of my app. Still, his charts were good inspiration for what dimensions I could visualize.

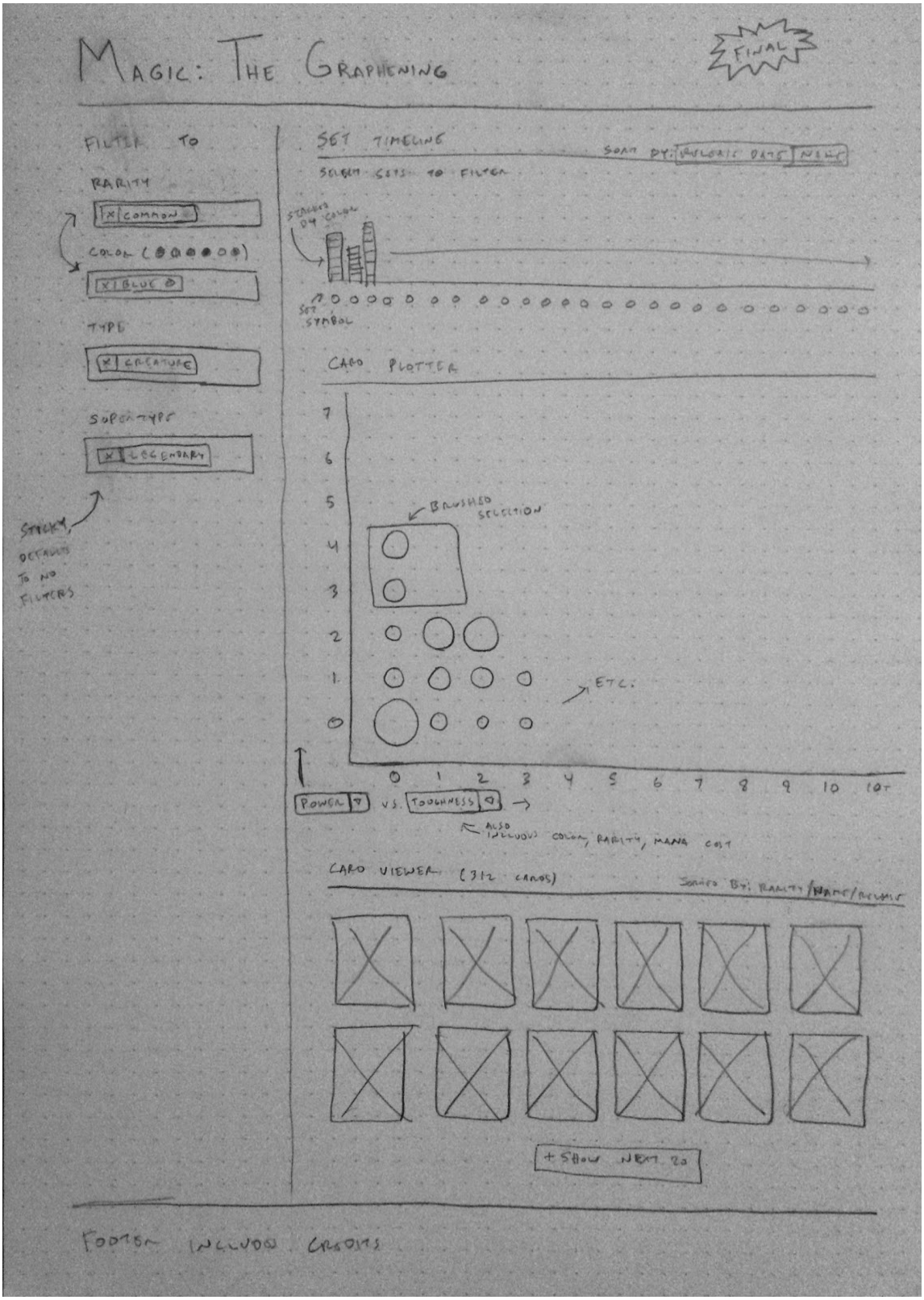
Data sources

As I mentioned above, price data is out of the question. I had better luck finding comprehensive data for the cards. A site called MTG JSON provides dumps of data by card and by set, while TK has an API for that data, along with a good documentation that lets developers append all sorts of query strings. I'm using the static data so I don't have to worry about rate limits, server errors, etc., but given the high degree of similarity between the two datasets I intend to eventually switch to the dynamic stream. I'll source images of the cards by calling the official Wizards of the Coast "Gatherer" site, cross referenced with the multiverse ID.

User story

So how will the app actually work? Here's what I imagine.

Tyler Palfrey is an 18-year-old who's planning to go to his first Friday Night Magic. He doesn't know what to expect, so he's browsing r/mtg, the subreddit for MTG. Near the top (actually, top link, 3k upvotes), is a link to "Everything You Need to Know About Dragons of Tarkir in Two Graphs." N.B. Dragons of Tarkir will be the newest MTG set when this app launches. *Wow, catchy title*, he thinks. *Only two graphs?* Tyler clicks and arrives at a page called Magic: The Graphening. *Another catchy title!* He sees three elements, alongside a sidebar of filters: a bar graph, a scatterplot and a grid of cards. The bar graph at the top shows the number of cards in every set of Magic. Seems like Dragons of Tarkir will have about 100 more cards than Fate Reforged, the last set. Below the bar chart is a scatterplot of the mana cost vs. color for all the cards in the set. He can see that green has a lot of high cost cards and white has a many one- or two-mana cards. Tyler sees there's a bubble in blue that has a mana cost of 9. He brushes over it and the card grid below filters down to one card, Clone Legion, a mythic rarity sorcery. "For each creature target player controls, put a token onto the battlefield that's a copy of that creature." *So you can double your own army or your enemy's--good for offense and defense,* Tyler thinks*. Maybe I'll play blue...*



My final sketch for the app, submitted (among others) in my revised proposal.

Week 2
Scaffolding the project

Front-end goals

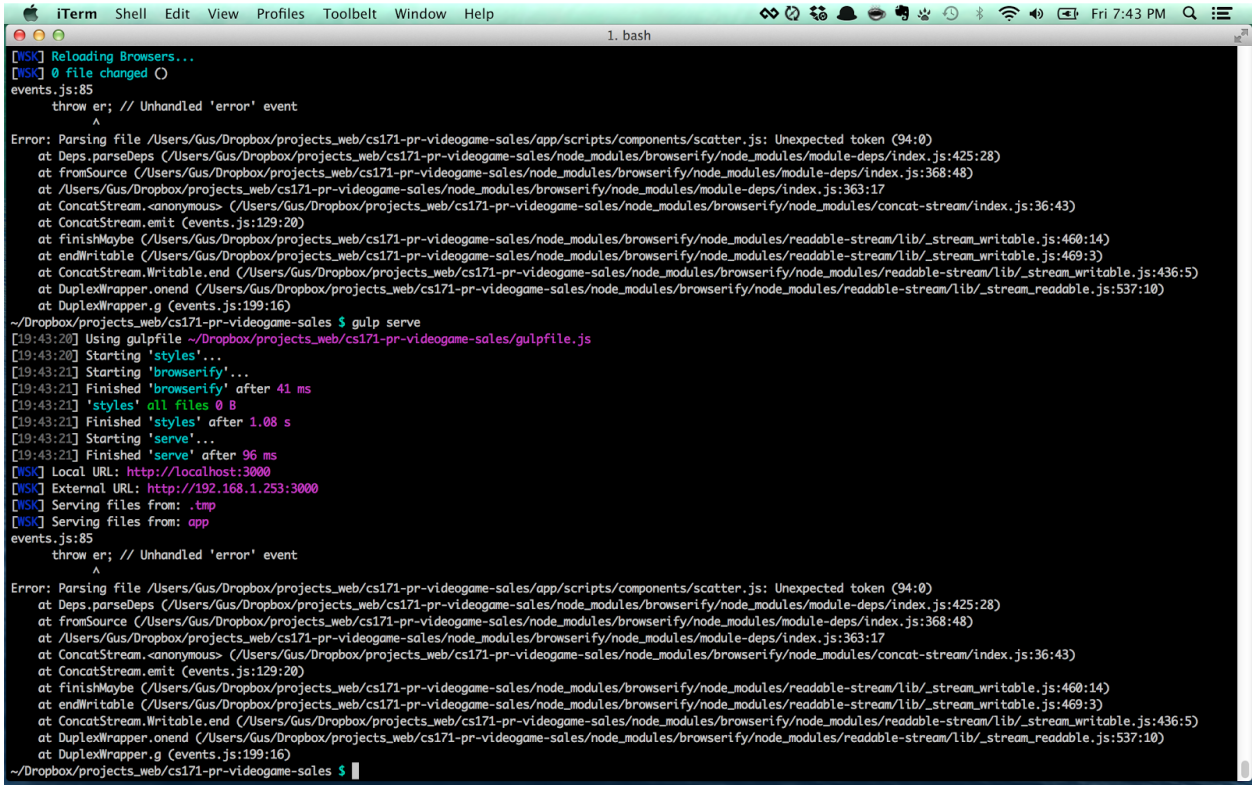
Like any frontend developer, I have a backlog of web technologies that I've been meaning to try or learn. At the top of my list was some sort of require system that would let me modularize my JS in a more elegant way than modifying object prototypes. I also wanted to use Flexbox for the page layout. I wanted the graphs to be adaptive, though I didn't commit to making the whole page responsive for the final deadline.

Research

Last year I used a boilerplate repo via Yeoman called gulp-starter-kit for a side project. I had a good experience, so I investigated using it again and found that the project had a spiritual fork in the Google Web Starter Kit (developers.google.com/web/starter-kit). Given that I was familiar with the gulp tasks in the original project, I decided to give the Google version a try. I liked what I saw upon startup, with clear tasks defined for dev builds and distribution, along with asset minification, concatenation, SASS support and live reloading.

Setup

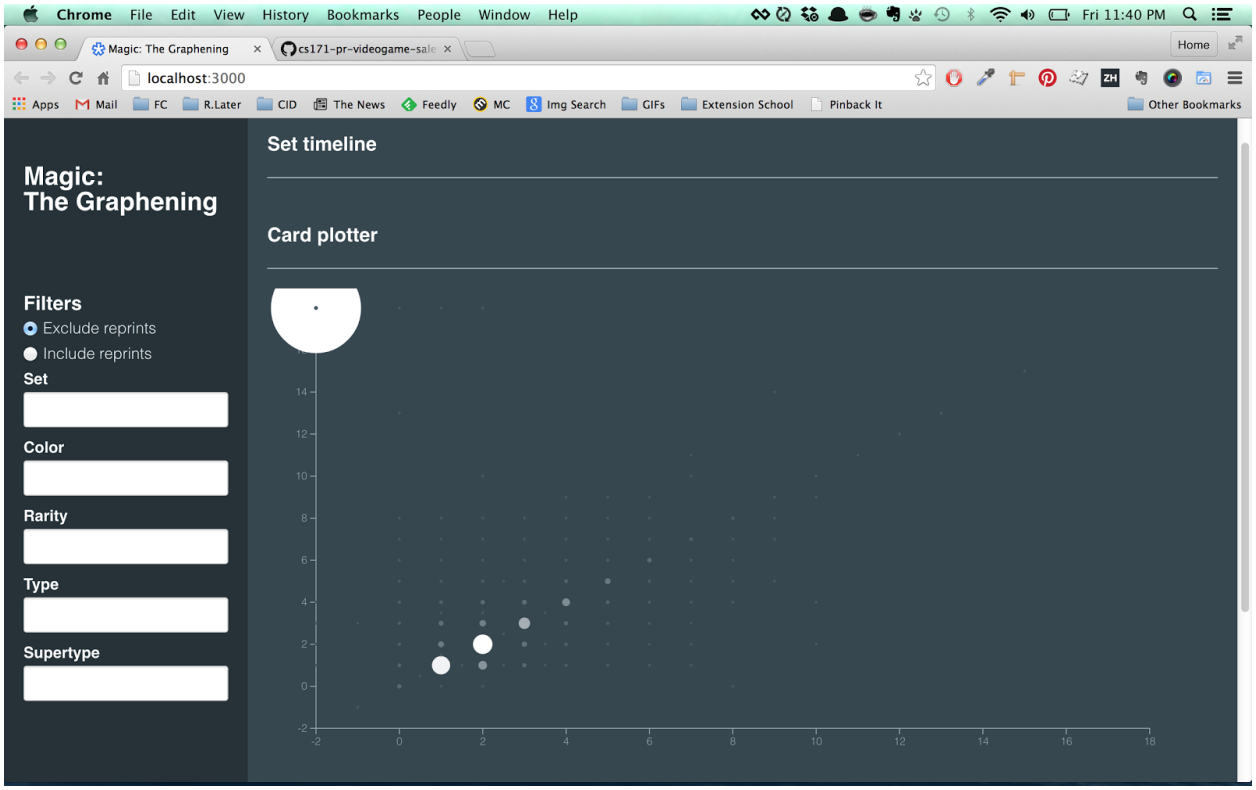
Getting the above to work was painless, although I did have to rip out a lot of the boilerplate styling. As I added dummy html and scaffolded out the DOM, I tried out Google's grid system (not a fan, tossed it) and some other SASS defaults and mixins they were using (again, tossed a lot of it). The biggest challenge was getting Browserify (my require() solution of choice) not only to work, but work in concert with the existing gulp tasks. That took all of two days and was quite the headache, but at the end I was bundling all of my JS (with a sourcemap to make debugging easier), linting only the code that I wrote, keeping the namespace free of globals and separating my utility functions and objects to their own file. To make sure I could get this to work with D3, and that the data was as clean as I assumed, I built out a scatterplot comparing mana cost to power.



At this point a lot of my process was debugging gulp tasks and npm in the terminal, so I took a screenshot.

Challenges

The scatterplot came with unexpected challenges. First, there were obviously a lot of non-creature cards in the set that didn't have a power dimension. Also, there were a number of outliers that really messed with the range of axes--one card had a power of 99 and there were a few with negative power ratings. The most interesting part of the graph was being dwarfed by the outliers. I'm going to solve that by defaulting the axes from >0 to 10 and grouping any cards with dynamic values or values <10 in a final bin on each axis. As for the non-creatures, I want to visualize the composition of the remainder so the user can see how many of the rest are lands, instants, sorceries, etc.



Scatterplot proof of concept.

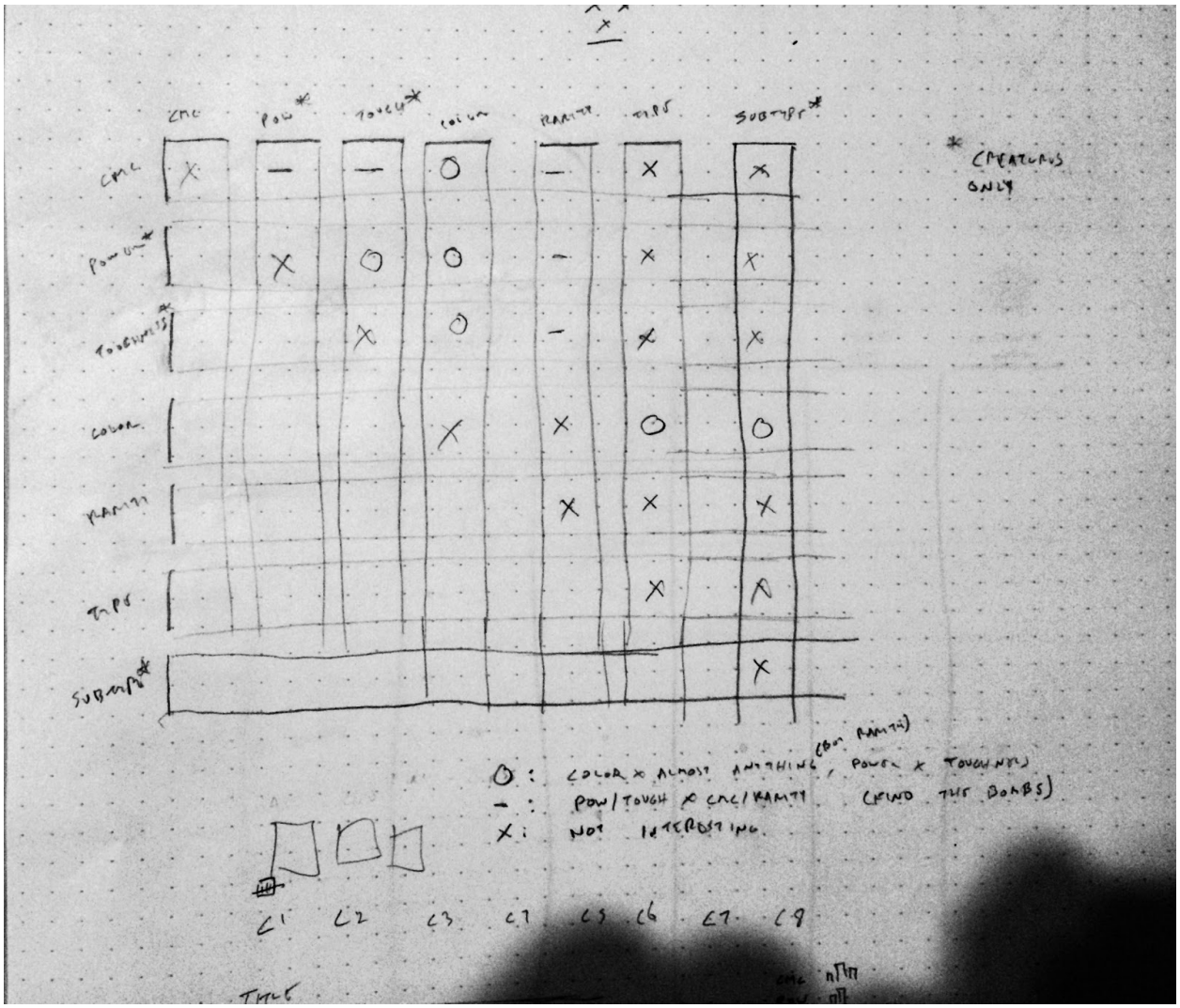
Writing the process book

As I started to write the process book, I got really hung up on the user story, which I probably should've included with original proposal (I find they keep the focus on the user in a way that a list of features doesn't). The default scatterplot I was considering at the time, power vs. toughness, just wasn't that surprising. Nor was the distribution of total cards by set. The ostensible goal of the first idea, to find the outliers, the power cards, wasn't nearly as compelling without the price dimension.

By limiting myself to one scatterplot, the dimension pairs that I thought might be interesting were hidden behind select menus. Why would someone "explore" a new app when the introduction was so uninspiring? My first thought was to add new graphs, to add interest and depth. Eventually I realized that I was just trying to bandage up a more fundamental flaw--my app was aimless, I didn't have a singular, overriding user goal that I could work toward. Without a target, there could be no hierarchy.

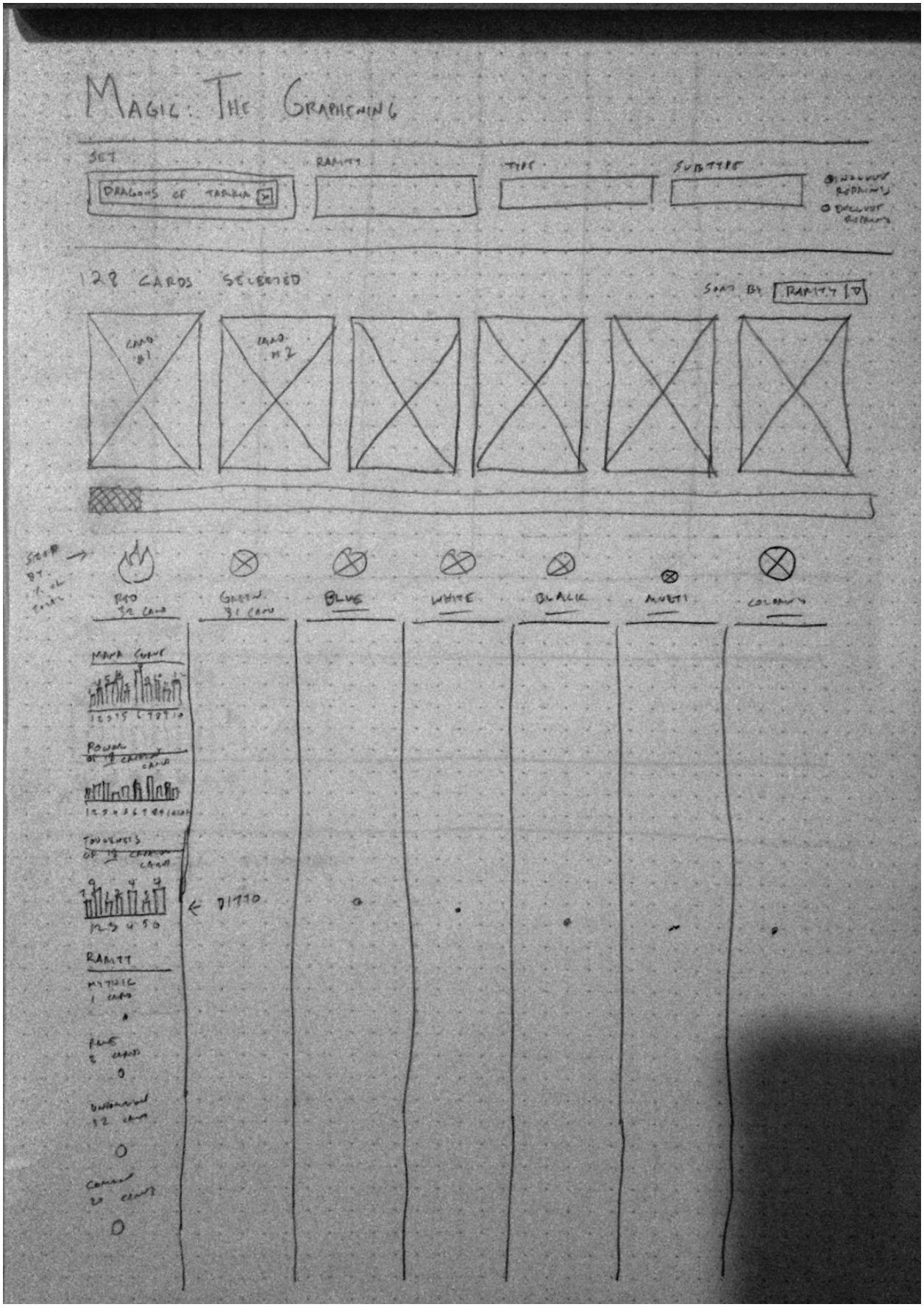
So I went back to the drawing board. Not entirely--it was too late to start from scratch. But I wanted to see if there was a way to take what I had learned from my initial exploration of the 7 primary dimensions and edit them down to a focused product. I

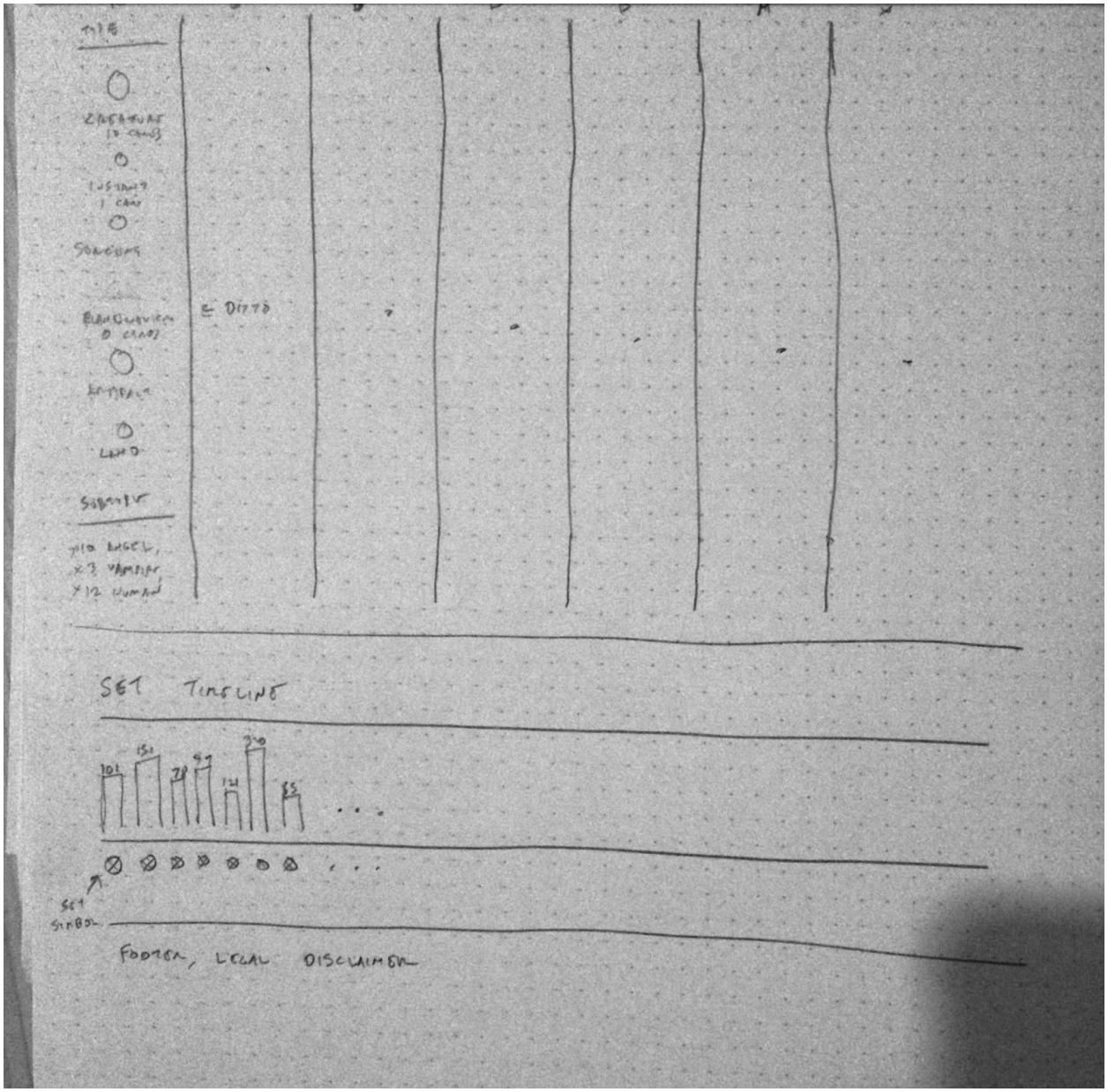
kept thinking about MTG users tasks and after a bit longer realized that I might have the keys to help answer one of them: “Which color should I play?” Whenever a new set is released, everyone wants to know how the colors compare and which they might end up choosing. It’s kind of like the sorting hat in Harry Potter. The metagame complicates this question eventually, as everyone gravitates toward two or three established decks that win at the top tournaments. But in the beginning, and always for the purist, the fun is in taking what the game designers provided and trying to figure it out for yourself. By taking a comparison across colors as my starting point, I could help answer a real question and honor the spirit of the game that initially drew me to it.



In order to hypothesize which dimension pairs were interesting, I created a pre-scatterplot matrix. Based on my experience playing MTG, I guessed that the most interesting comparisons would be between different colors. The two other graphs that I thought showed promise, power x toughness and mana cost x rarity, were just inferior

solutions to finding the power cards or “bombs” (compared to using price as an indicator).


MAGIC: THE GRAPHENING
SET
DRAGONS OF TARKIR
RARITY
TYPE
SUBTYPE
128 CARDS SELECTED
SORT BY RARITY
CARD #1
CARD #2
STEP BY 1 OF THIS
RED 32 CARD
GREEN 32 CARD
BLUE
WHITE
BLACK
MULTI
COLORLESS
MANA CURVE
POWER
TOUGHNESS
RARITY
MYTHIC 1 CARD
RARE 2 CARD
UNCOMMON 32 CARD
COMMON 32 CARD



My revised sketch of the application. I put the emphasis on the card images, and made a point to keep the graphs simple and comparable across color columns. If the writing is too small, the multiples I'll be creating for each color are distributions for mana cost, power, toughness, rarity, type and subtype. When a user selects a filter, the multiples for that dimension will disappear, as they'll be redundant. The set timeline remains at the bottom, but it's icing on the cake and not essential to the functionality of the app.

Next steps

While the design of the app is different than it was before this weekend, I'm not very behind in terms of development. I have the app scaffolded, all the data (and knowledge

that it looks good visualized) and a real design goal. By the end of next week I should have all of the multiples built out and the cards coming in. That'll give me the last week to implement filters, brushing and possibly the timeline at the end.

Weeks 3 and 4

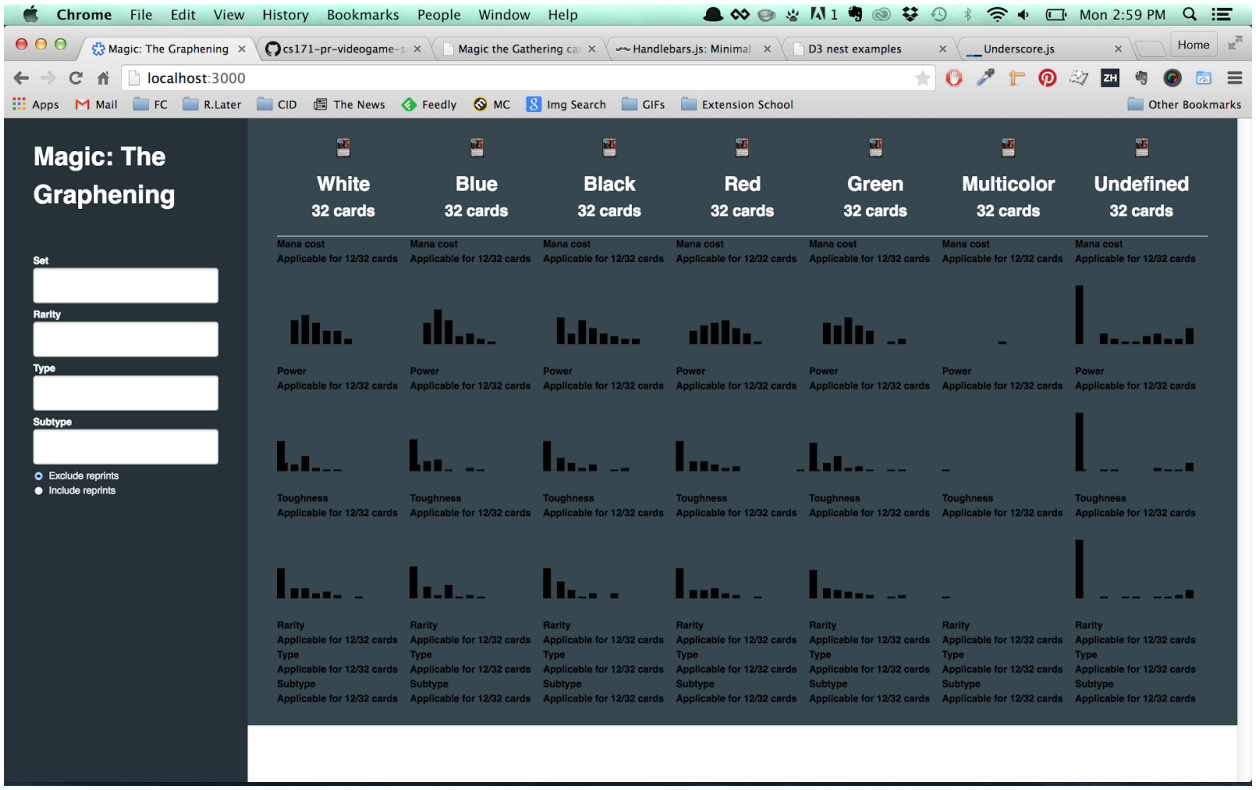
Layout challenges

Alright, so some of these sections will be a little brief because I'm writing this on May 4. Not how the the journal was supposed to work, I know, but I don't think I'll ever be one of those people who can keep a diary. That being said, a memoir is generally a better read, eh?

I started building the color columns with divs, keeping in mind the general advice to not use tables for layouts. Once I had them scaffolded and styled I realized that a table was completely semantic in this case and that really it was my lack of knowledge regarding table styling + flexbox that was keeping me away from that path. So I refactored everything to use flexbox and tables, learning a lot about table-specific styling along the way.

Bar graphs and bubble charts

Building the actual bars wasn't too tricky--I was able to reuse a lot of code from previous homework assignments. What was difficult was getting the rollups for each of the dimensions as well as dynamically calculating the domain extents for each dimension on filter. I learned a lot more about Underscore in this process, and finally got a grasp on why map() is useful. A recommendation for next year--allow students to use Underscore from the get-go and highlight the pluck() function in particular. It's such a common operation. I also finally started using debugger; much more than console.log(), which I've been wanting to do for awhile now.



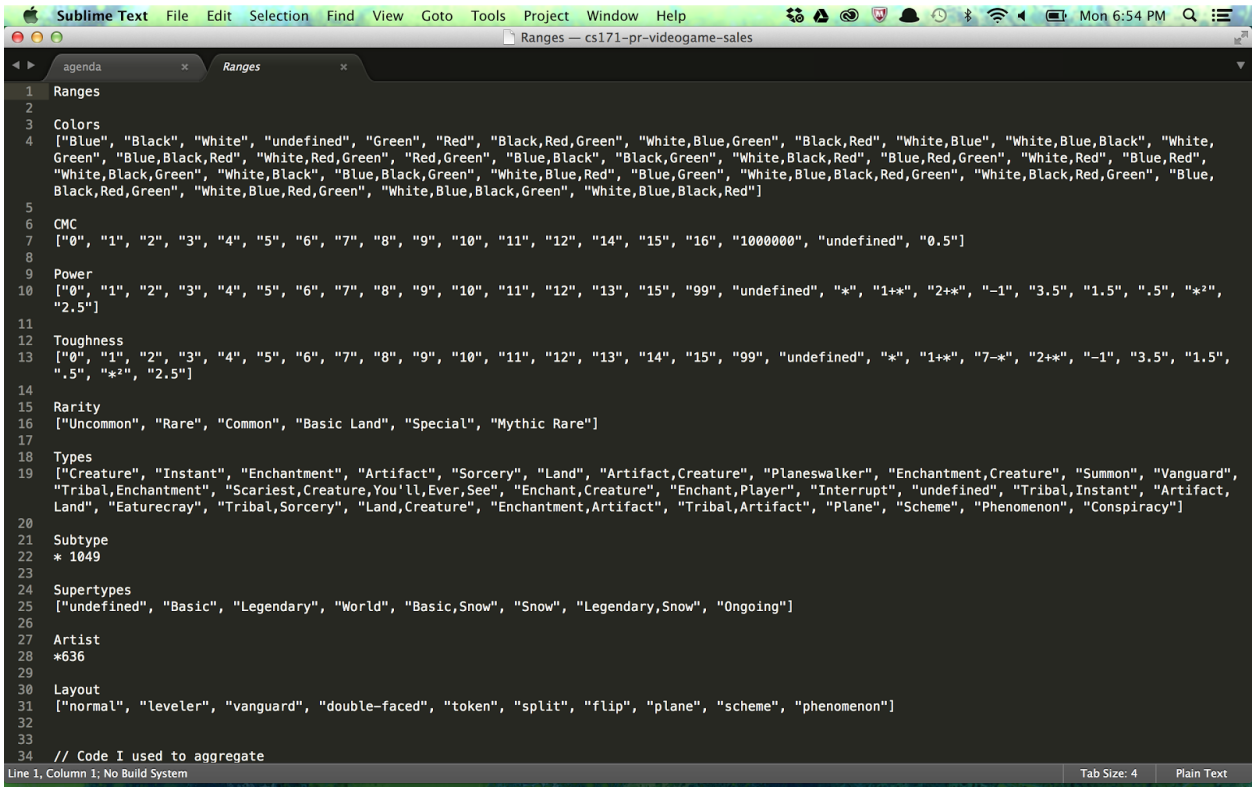
Bars proof of concept with a lot of dummy copy, but real data.

The bubbles were relatively straightforward as well, with the only difficulty being getting the height of the containing svg for each to scale appropriately (no matter how many times I did the math, I couldn't get the ordinal positioning to work perfectly). James, my TA, told me that I should explain here why I didn't use more complicated graphs than bars and bubbles, like showing deficits between dimensions with a dual-sided bar chart, for instance. The answer is simple: If there were a use case that I felt wasn't being addressed by the current visualizations, I would've. But I wasn't going to get fancy just for the sake of fanciness. The bars and bubbles were space-efficient, familiar (especially in the context of trading card game mana curves), easily comparable across columns and didn't require too many DOM nodes. Does this project have sunbursts and chord diagrams and a Sankey or two with brushing to boot? No. I focused on usability first and foremost, and I hope it shows in the final application.

Handling exceptions

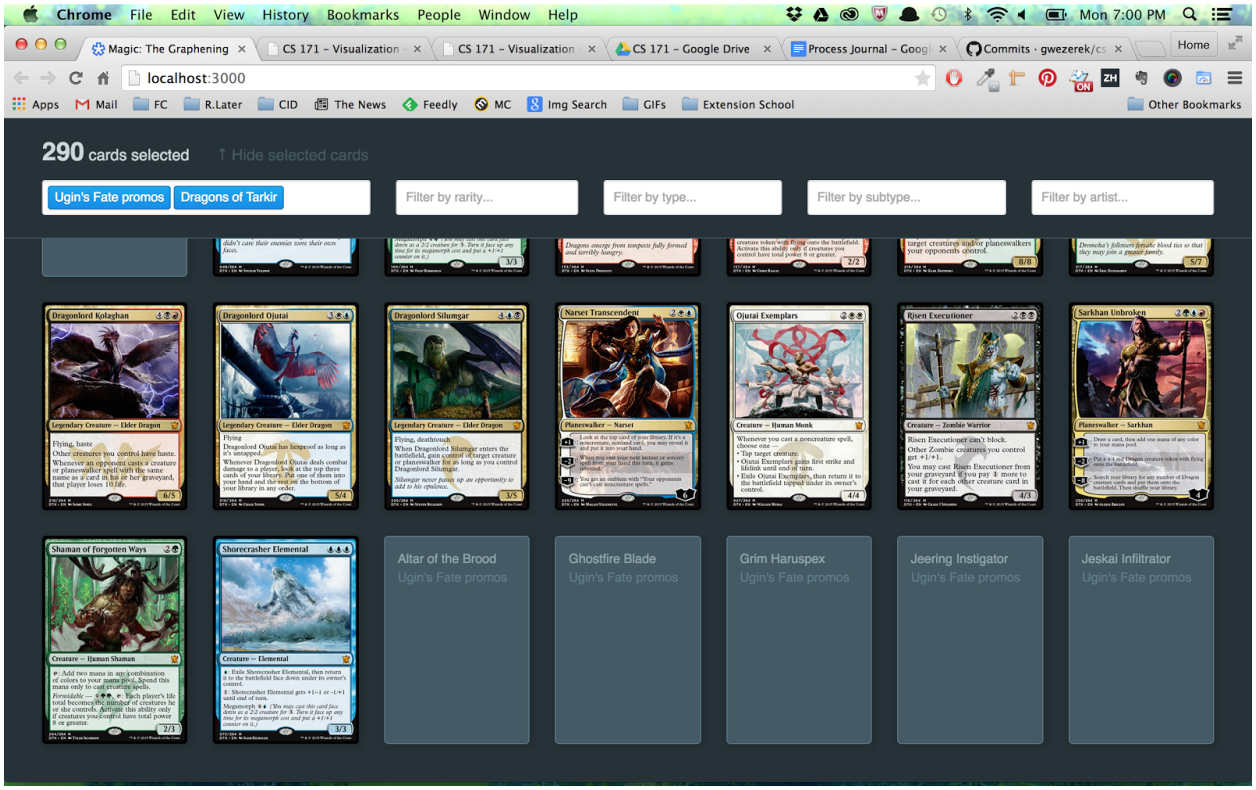
Once I had the basic charts down, I noticed errant data hanging out at the edges of my SVGs as well as a number of errors for invalid attributes in the console. Luckily it was easy to test whether I had caught everything as I could run the graphs for the set of all cards. Still, to wrap my head around which dimensions might be causing the problems

I wrote a function to filter, flatten and print the ranges of each dimension. That revealed a number of annoying values from Magic's joke sets, Unhinged and Unglued, that I was able to anticipate.



My ranges reference.

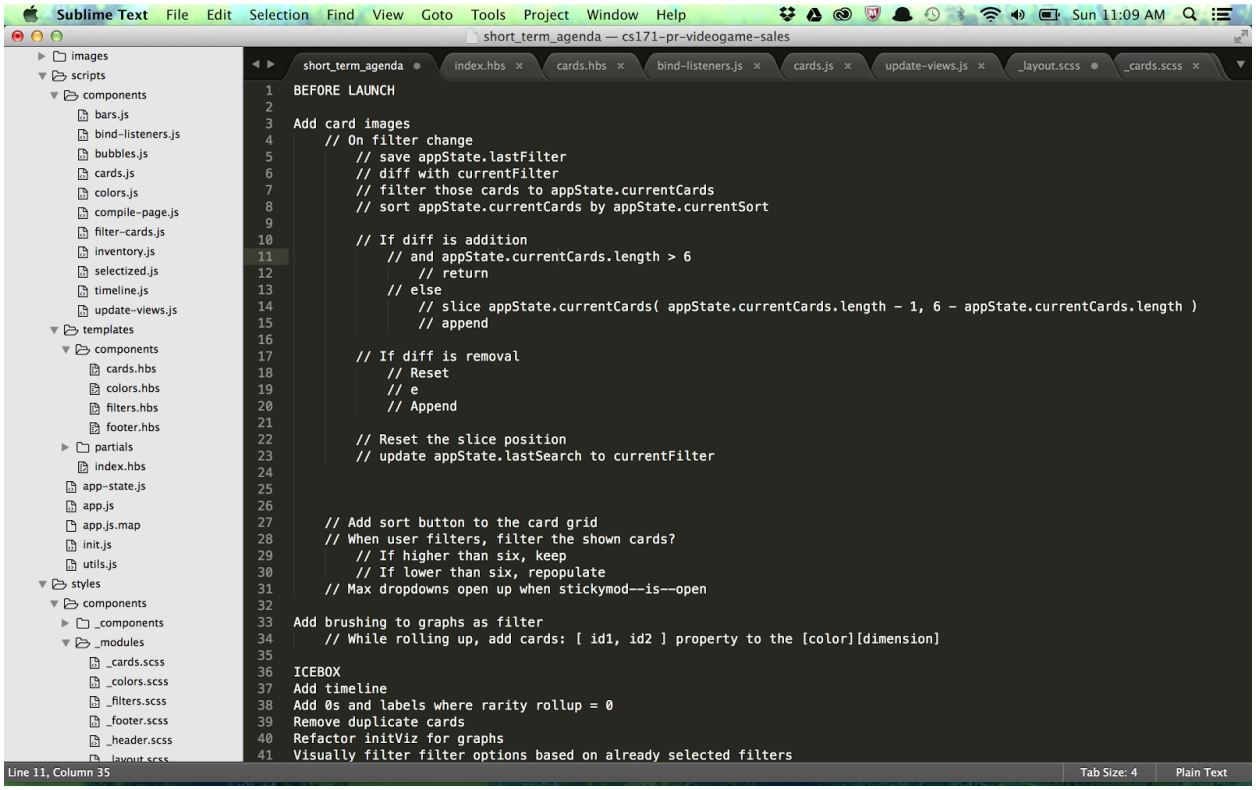
Card images were a special kind of pain. No single service, including the Gatherer site run by Wizards of the Coast, has images for every card. And though the Gatherer may be the most comprehensive, the scans are low-resolution. I ended using a control flow testing for all sorts of attributes that the Gatherer and magiccards.info used for their card images, as well as creating a proxy card style for when neither site had the image.



An example of image proxies in the final version.

Filtering

Again, Underscore was essential here. The details of the filtering aren't too interesting, aside from the fact that I needed to find the intersection of all select elements but the union of options within each select. Once I had the set of cards I wanted to use, I stored it in a required module called appState. Separating appState into its own module helped me remember at all times what variables I had available at any time.



Unused pseudocode for a smarter reload of the card grid on filter, to preserve cards within the new filter that have already been loaded. I'd like to go back and implement this after launch.

Adding brushing

There were a few fun bits of logic with the brushing. First was how to invert the ordinal rollup represented by the brushed bars into the individual cards. The solution, inspired by a conversation with my TA, James, was to return an object for each rollup instead of just a count. The object included an array of all the included card IDs. Given that I had to create 21 brushes, I needed a way to store and filter through all of them. Unfortunately, storing the d3.svg.brush() wasn't very straightforward. I ended up binding each brush to the chart's svg element and then using the SVG as an entry point when I needed to search through the brushes to see which was activated and brush.clear() the rest. To invert the rollup I ran each of the selected graph's bars through a function testing whether the xScale(key) + width of bar was within the brush.extent(). I also added handling for empty brush selections and made sure the brushing was efficient by storing a copy of the most recent filtered set and only searching for selected IDs within that array.

Adding templates

At a certain point I hated how everything was modular except for my html. I was using underscore templates for some dynamic DOM stuff, but I decided to go all the way and include Handlebars. It took awhile to get it into my build process with node-hbsfy, as well as to learn how to register partials in a require system, but it was totally worth it in the end.

Design details

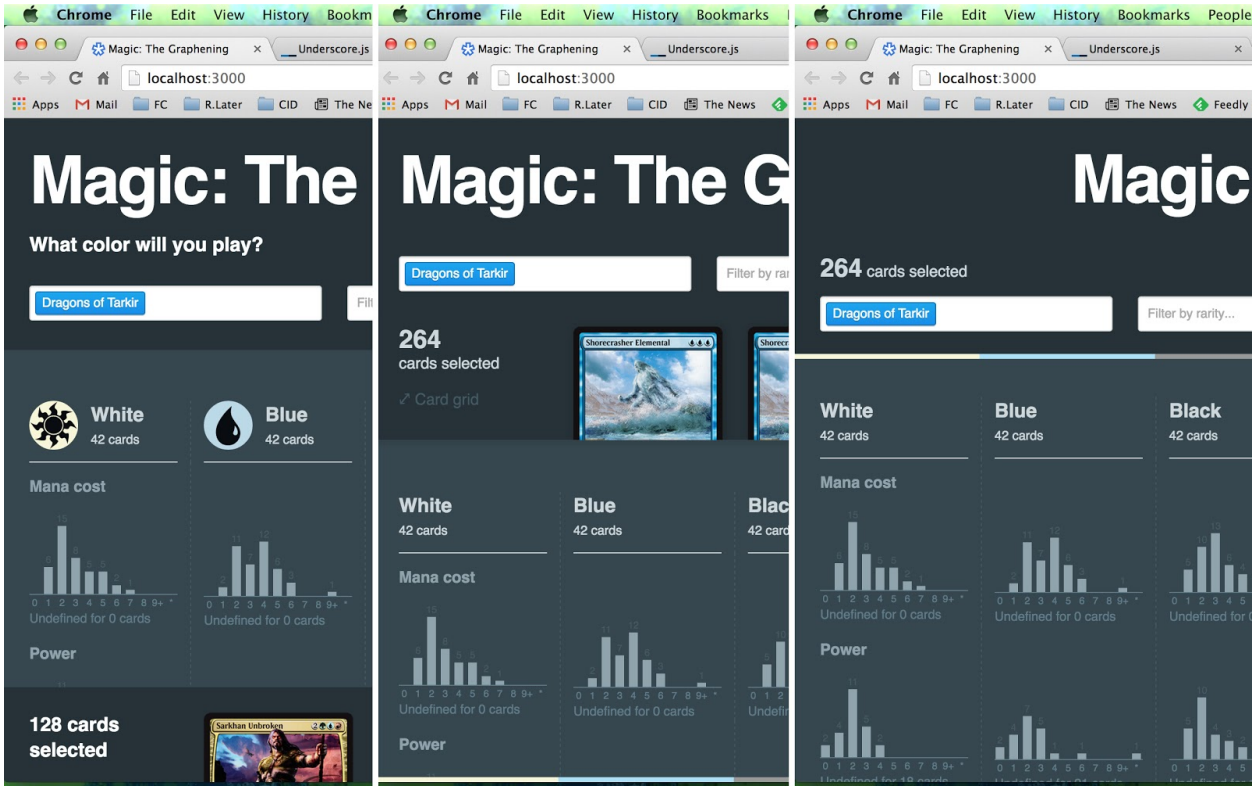
Small design decisions I want to point out:

- Drawer expand tease animation on hover for the expand button
- Sorting of card grid by descending rarity, so as to trigger fond memories of the most powerful cards if they're searching by set
- Increase in height of color gradient border on header becoming sticky so as to decrease visual clutter on load but bring attention to the utility of the labeling on scroll
- Subtle drop shadows on cards to add sense of physicality to the scans
- Consistent spacing and color palette thanks to SASS variables
- Amazing project title

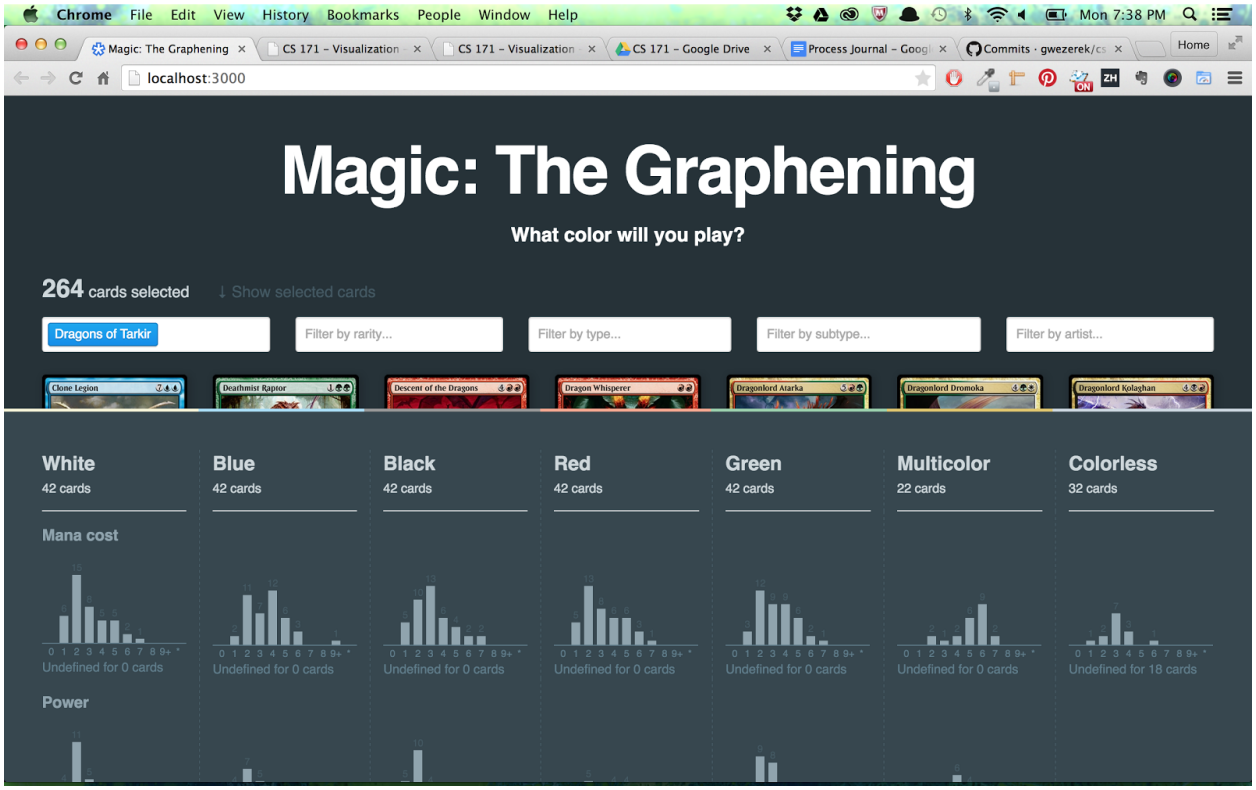
User testing and feedback

I hesitate to use the phrase user testing, because showing the project to a few friends does a disservice to what is a real and really difficult UX design task. But collect feedback I did. My boyfriend asked if there was a way to filter the cards by artist. That was one of the keys on each card object, and it only took me three minutes and three lines of code to implement. I was really proud that I had abstracted the functionality away enough to do that--that I had built a platform and not a total one-off.

Turns out the artist idea was a great one as a number of the people I showed after the fact mentioned that was their favorite feature, including one person who said, "I know someone who would use this tool exclusively for the art." Other feedback I got mentioned that the connection between brushing and the new list of selected cards wasn't strong enough, so I reneged on the design decision to hide the card grid and instead included a little peek at the cards. It means a bigger and busier sticky nav, but I think the usability gains make the tradeoff worthwhile.



A few of the many lockups I tried while balancing the header, colors, card grid and filters.

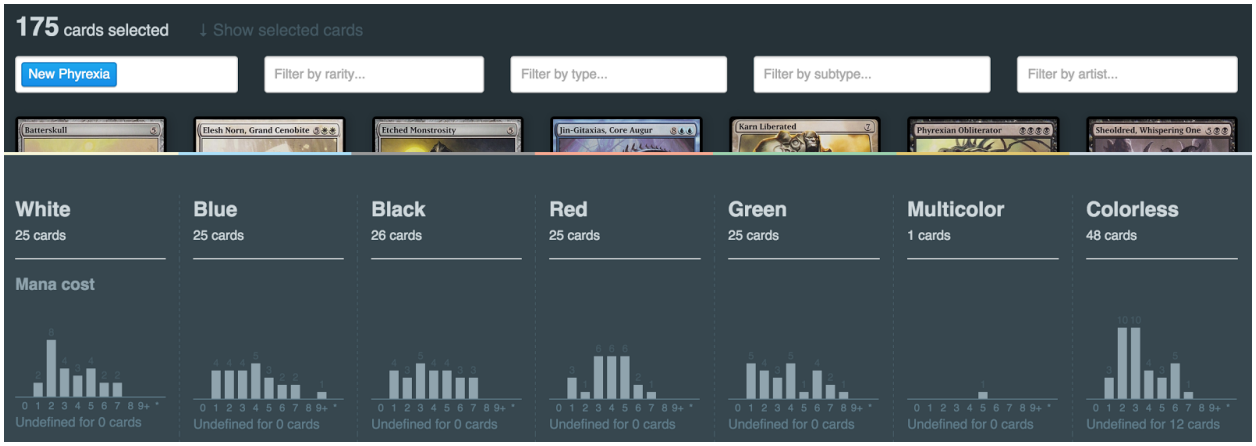


The final page.

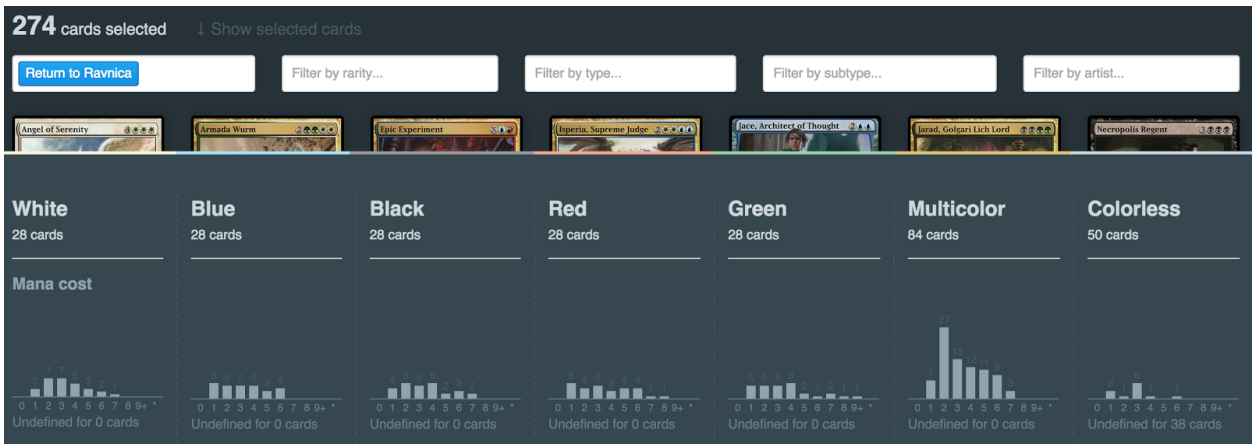
Evaluation and Next Steps

Evaluation

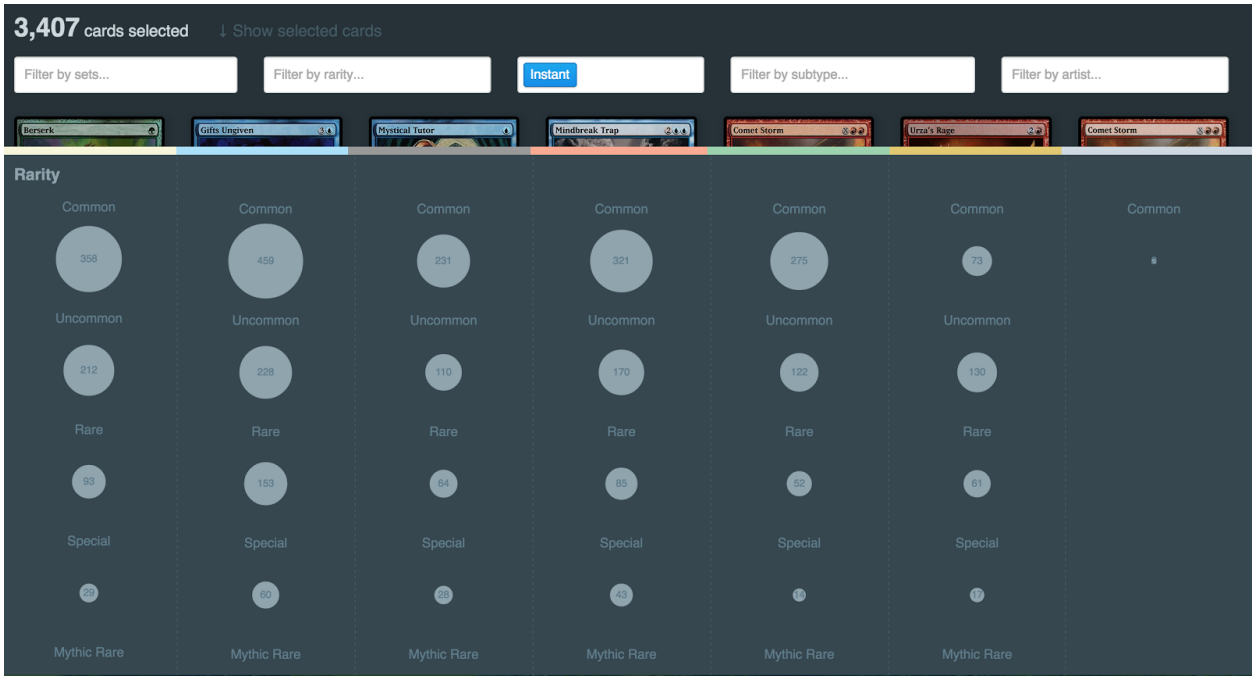
There aren't any earth-shattering discoveries in my project, just a lot of confirmation via charts of key elements of MTG. Eventually I hope to read through a lot of MTG lore and game design history and create a narrative view that pairs the history of Magic with the card sets and images to match. To get a sense for the kind of discoveries you can make with the app, I've included a few screenshots here.



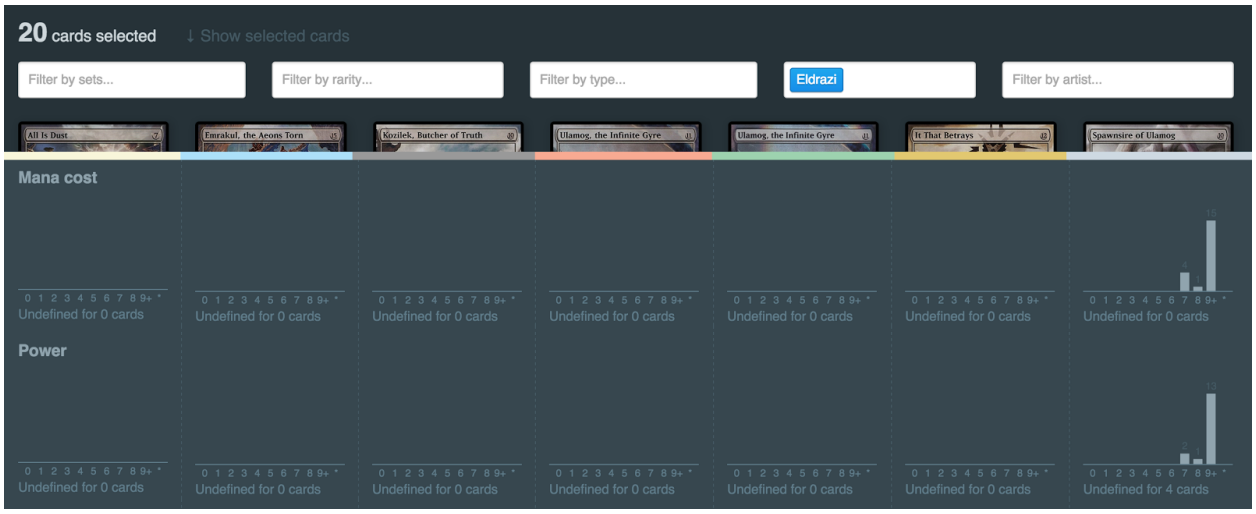
The New Phyrexia set had a lot of colorless cards.



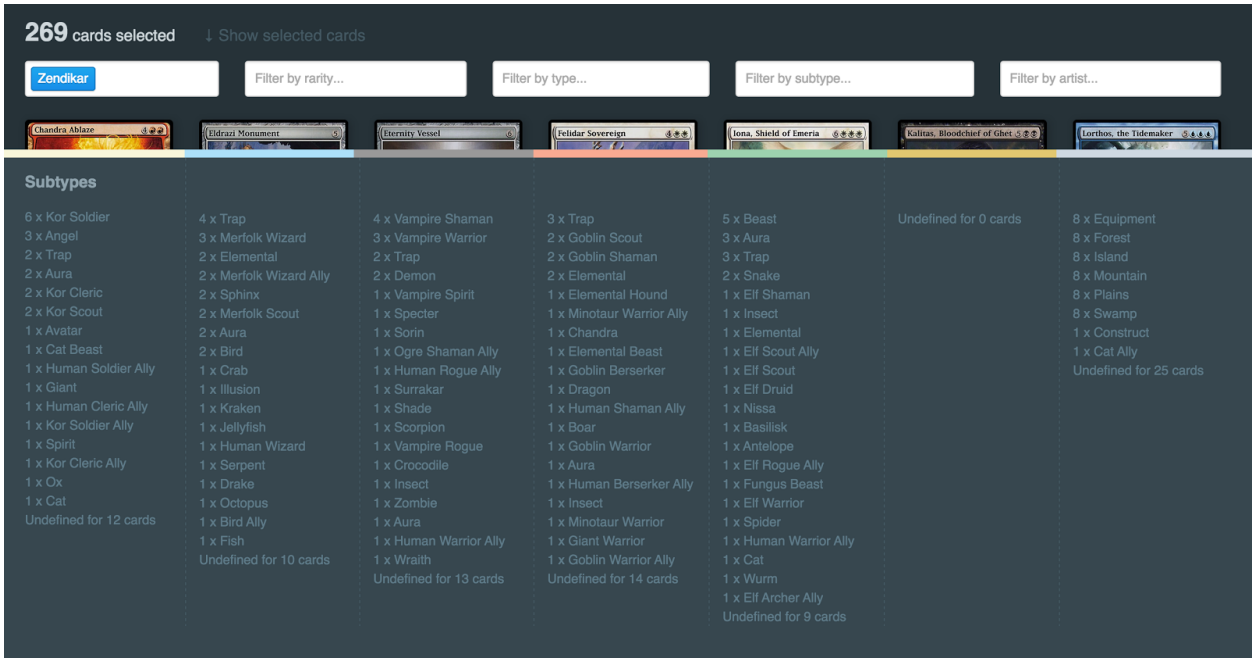
While Return to Ravnica was almost all multicolor.



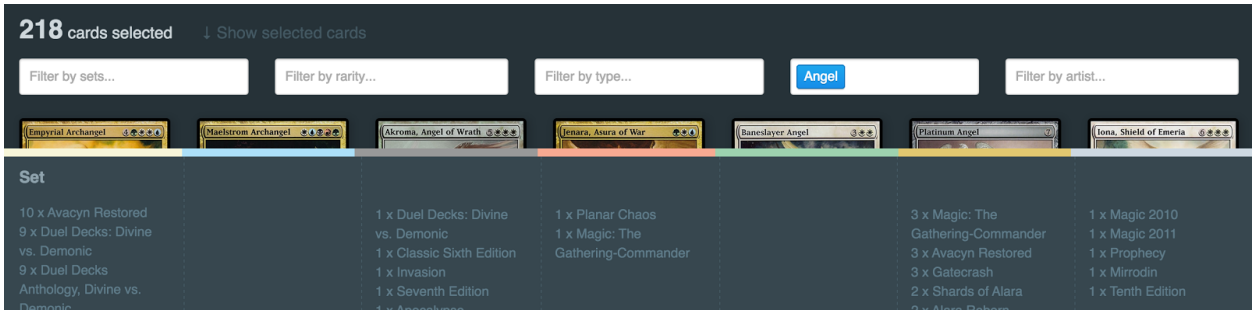
Instants are very common in blue, and there are basically no colorless instants,.



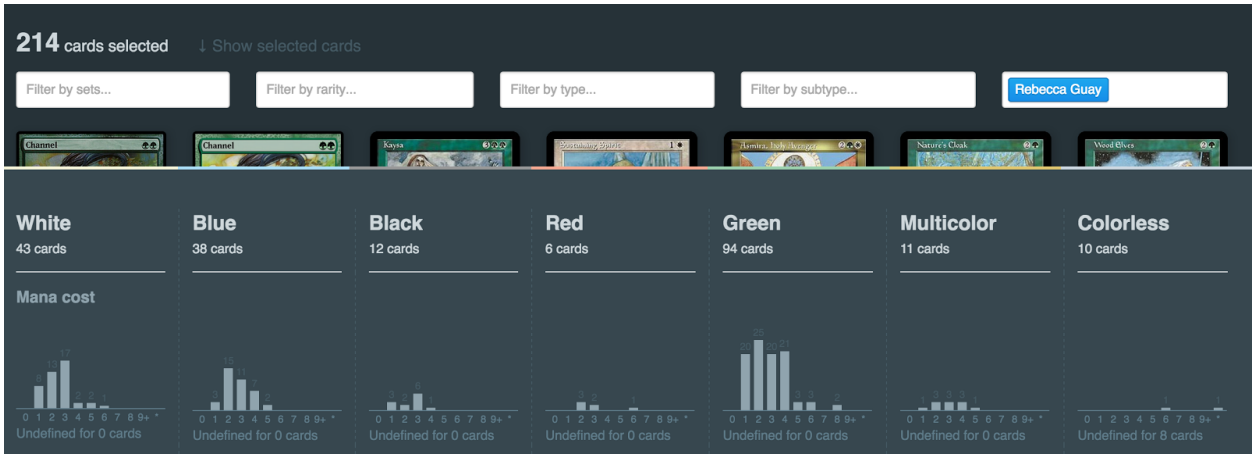
The Eldrazi are all extremely powerful cards with high casting costs to match.



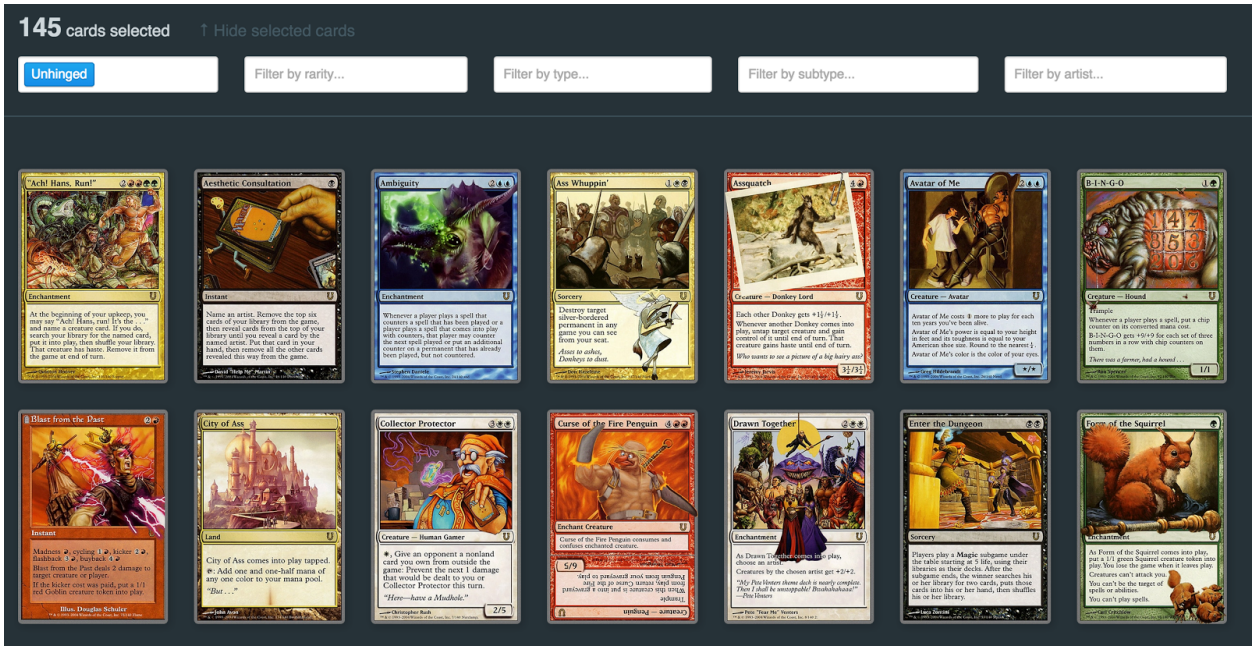
The Zendikar set featured a race known as the Kor, vampire shamans and a lot of traps.



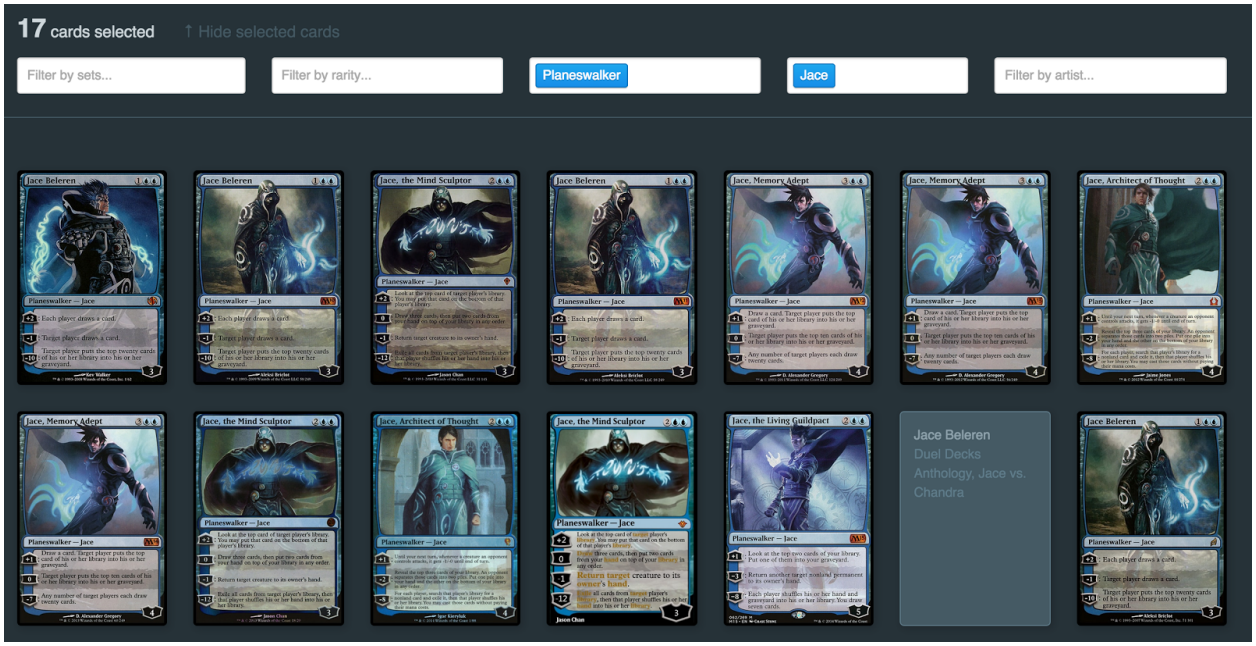
If you want to play with angels, your best bet is finding old packs of Avacyn Restored.



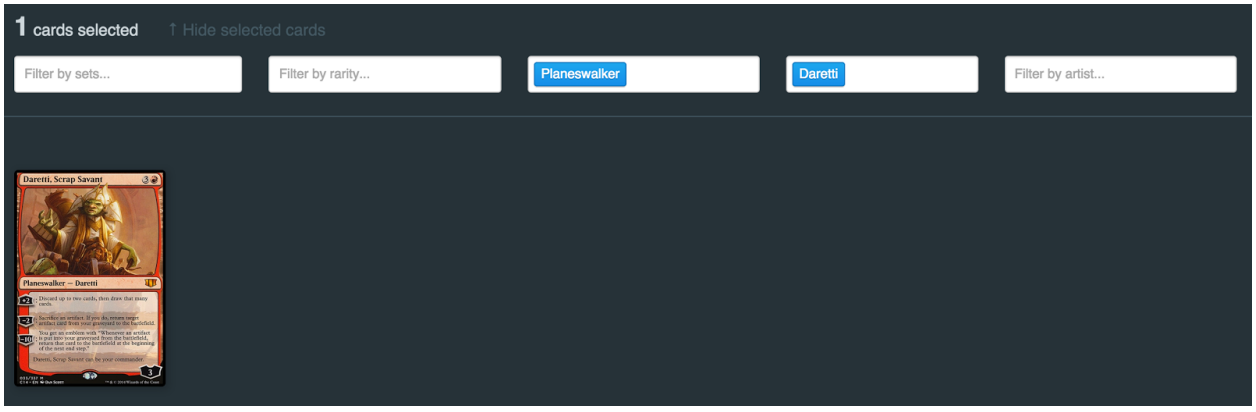
Artist Rebecca Guay has illustrated many cards, but almost exclusively in green, white and blue.



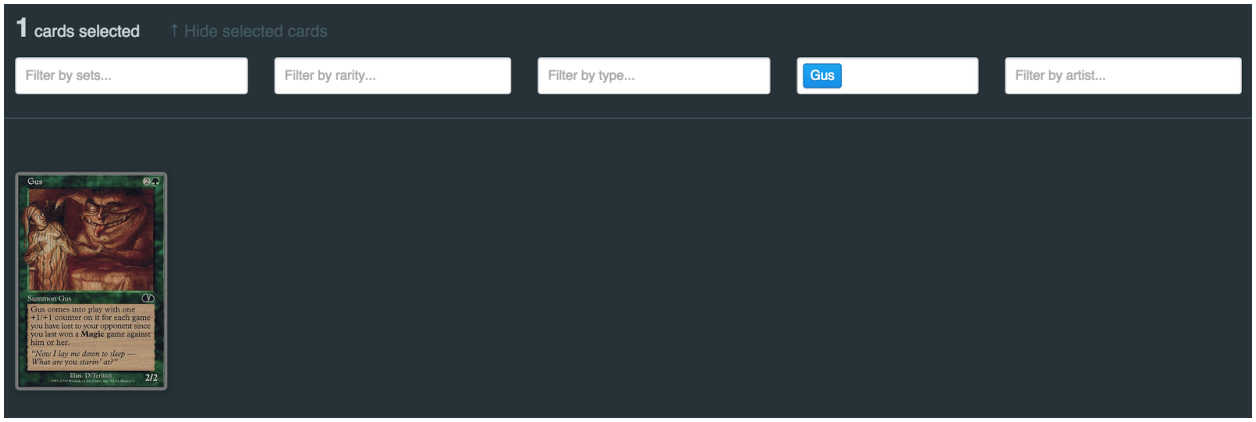
The Unhinged set was full of ridiculous cards with ridiculous art.



Some Planeswalkers have been reprinted many times, in many iterations.



While others never really took off...

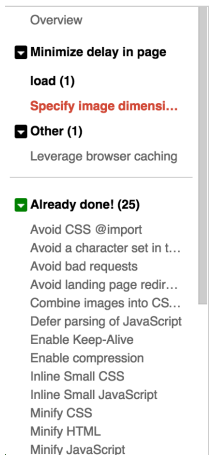


There's a card named after me!

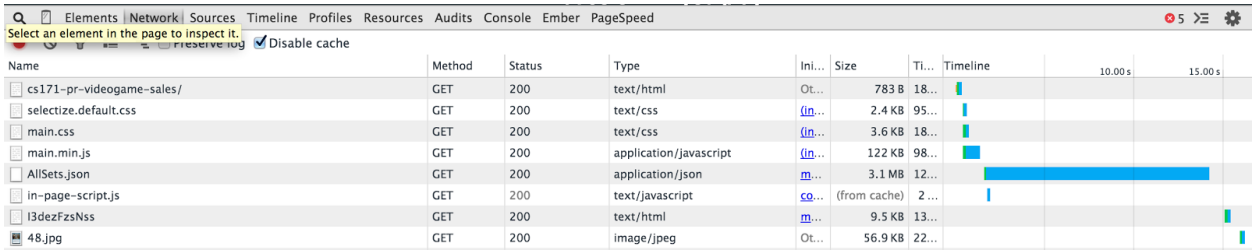
Next steps

I fully plan on continuing to build out and maintain this site after launch. I've been diligent about writing clean, modular, maintainable code and though that may not count for much towards a grade, I think it'll pay off big time in the long run. I plan on doing a full release and Reddit/Twitter blitz a few weeks after turning this in, once I'm able to add the following.

- Request only the most recent set on load and lazy load the full set after to shorten load times



Even when you're getting a solid A from Google Pagespeed Insights...



...there's only so much you can do before you have to acknowledge the 3.1MB (gzipped!) JSON file in the room.

- Precompile templates, including select boxes
- Scrape all images and serve them from my own database, as well as minify them
- Add a filter to remove reprints from the selection
- Add timeline bar graph or some other solution to showing what sets the selected cards belong to
- Add 0s and labels to the rarity bubbles where there are no cards
- Refactor graph js to accept an init flag and DRY out build logic
- Visually filter the filter options based on already selected filters
- Switch from static JSON dump to API
- Add icons, release date and optGrouping by block to set options
- Add "0 cards" copy to bar graphs when there are no cards
- Hide undefined labels if there aren't any undefined
- Add smarter card filtering for images
- Add functionality to sort card grid by name, rarity
- Add filtering functionality to bubbles and inventory to mirror already present functionality in the select elements
- Add Google analytics
- Add sharing buttons and counts for Twitter and Facebook
- Design a logo and branding for the site
- Snap brush selection to bars
- Restyle selectize inputs to match app
- Add favicon