Process Book - Easier Openings

Contents

Overview and Motivation:

Basic Info

Project title: Easier Openings

Name: Ethan Stanley

Email: u1312964@utah.edu

uID: u1312964

Repository: https://github.com/ethanstanley3/dataviscourse-pr-mentalhealthfactors

Background and Motivation

I chose this project because I enjoy playing chess, but also because I am interested in game strategies that are suboptimal against an optimal opponent, yet more effective against imperfect opponents. In particular, many chess players study theoretical openings in order to improve. However, these openings are frequently based off of expert level play. As a result, less experienced chess players may have less success with these openings because they are not forgiving to mistakes. I think it would be cool to make a visualization that allows one to explore which openings are most successful across a range of skill levels.

Related Work:

I was inspired by a visualization shown in class demoing a dynamic pie chart. When clicking on a wedge of the pie, the chart would show data contained in the pie chart wedge. My visualization has a similar visual component. I think it is a cool way to dynamically visualize a tree data structure.

Questions:

The primary question I am trying to answer is: “what chess openings are most effective at lower skill levels?”. I would like a visualization that allows the user to explore what moves are popular or successful from a given position and be able to choose the skill level of the games in the dataset. That way, one could filter for intermediate games to see which openings and moves are effective at the intermediate level, as an example. A benefit is that people, including myself, could study and learn openings that consider the skill level of the players using them.

I am also curious to see if my visualization will reveal openings that are a) only effective at high levels b) only effective at lower levels or c) effective across all skill levels. This would be useful information to know when deciding which openings to study further. The visualization could answer questions such as “which openings will be effective right now?” and “which openings will continue to be effective as I improve at chess?”.

Currently, these are still the questions I am trying to answer with my project.

Data:

I collected my data from https://database.lichess.org/#standard_games.

This is an online database of all chess games played over the internet on lichess.com. There are currently around 3.7 billion games in the database. This dataset is in PGN format, which is a standard plaintext notation for chess games. The dataset is free for anyone to use.

I have done a fair amount of data processing. I wrote a python script (process_data.py) that reads the PGN file containing 3.7 billion games and turns it into a JSON tree that powers the visualization (each node in the tree represents a state of the chess board). It does this by reading the N first games that match the specified filters (such as player skill level) and then iterates through each move of the game, adding nodes to the JSON tree when a novel move is played and updating a node's win and usage rate when it already exists. Afterwards, I run a depth first search algorithm to prune all the nodes in the tree that have been played only a single time. That way, only aggregate data is stored. This also makes the data much easier to load into the visualization.

Currently, I have processed 100,000 games without filters and tested my visualization with this. For the final version, I will run the exact same script but alter the parameters to filter games for different skill levels so create multiple loadable files. The user will then be able to choose which file is loaded.

I chose to preprocess the data into JSON files rather than doing it dynamically because the raw data is far too large to upload to GitHub (it is over 250 GB). The JSON files, by contrast, are much smaller and can be loaded on demand without slowing down the visualization.

Exploratory Data Analysis:

I explored my data by generating a preliminary version of the JSON tree that powers my visualization. I logged it to the console and then expanded through the nodes to see if the amount of data I could feasibly process would generate a game tree of sufficient depth. I found that lines that are played by a high percentage of people can be explored very deeply (sometimes up to 50 moves). On the other hand, sequences of uncommon (bad) moves can run as low as 5 nodes deep. I think this is more than sufficient to explore openings because bad moves are almost certainly not worth exploring deeply anyway. Additionally, I will likely run my computer overnight to process more games (1-10 million instead of 100 thousand in the testing version) so that the game tree is richer. I have found that the size of the JSON tree is logarithmic against the number of games processed, so I don't anticipate this to slow down the visualization much.

Design Evolution:

I have included 5 sketches on the following pages: a brainstorm of elements to include, 3 prototypes exploring different aspects of the brainstorm, and a finalized version that incorporates the best components of the prototypes.

A general idea I explored in the brainstorm is how the user should interact with the visualization. First, I considered the most obvious mode of a chess board on which the player can move pieces. This would be the easiest input for a chess player to pick up, but it is hard to display meaningful data on the board. It gets cluttered too quickly. I explored this idea in prototype 2. Another option is to represent the possibilities in the chess game as a tree. The user could click on a node and the tree would dynamically expands to show subsequent possibilities. I could encode data (like move popularity or win rates) with the radius of the nodes, width of the edges, or color of the node. But this is problematic because it is hard to look at a path through a tree and relate it to a chess board. I explored this design choice more in prototype 1. Lastly, I considered a nested pie chart, where slices of the pie chart could be clicked and more slices would appear, designating possible moves. The area of the slices would encode the usage or win rate of the move. This idea was explored in prototype 3.

Prototype 1:

In this prototype, I explored using a tree to represent the moves of the chess game. This has the advantage of effectively showing all past moves, instead of just the current state of the board. The relevant data (move popularity or move win rate depending on what the user is interested in) would be encoded by the widths of the edges or the areas of the nodes. Width is one of the most effective visual encoders, so it would be effective. Area is not quite as effective, but it could be used to encode the metric that the user is less interested in. For example, area could encode win rate, while the width of the edge could encode usage rate. That way, the marks (nodes and edges) would efficiently encode the relevant channels. One problem with this design is that the screen could become cluttered by having too many nodes. On average, there are 20 possible moves from a given chess position. I included a table at the bottom of the visualization that can include more relevant data about the most recently selected node. This design uses sliders to filter the skill level of the chess games being visualized.

Prototype 2:

This prototype explores using a chess board as the main visual component. The possible moves would be encoded by arrows on the board. The relevant variables would be encoded by the width of the arrow. For a less important variable, I could use the opacity of the arrow. Opacity is a much worse channel than area or width, so I think this design less effectively encodes the data. Additionally, it suffers from being cluttered. Many arrows would overlap. This would make it hard to select which move to explore. To overcome the limited effectiveness of the encoding channels, this design would need a table containing relevant data at the bottom.

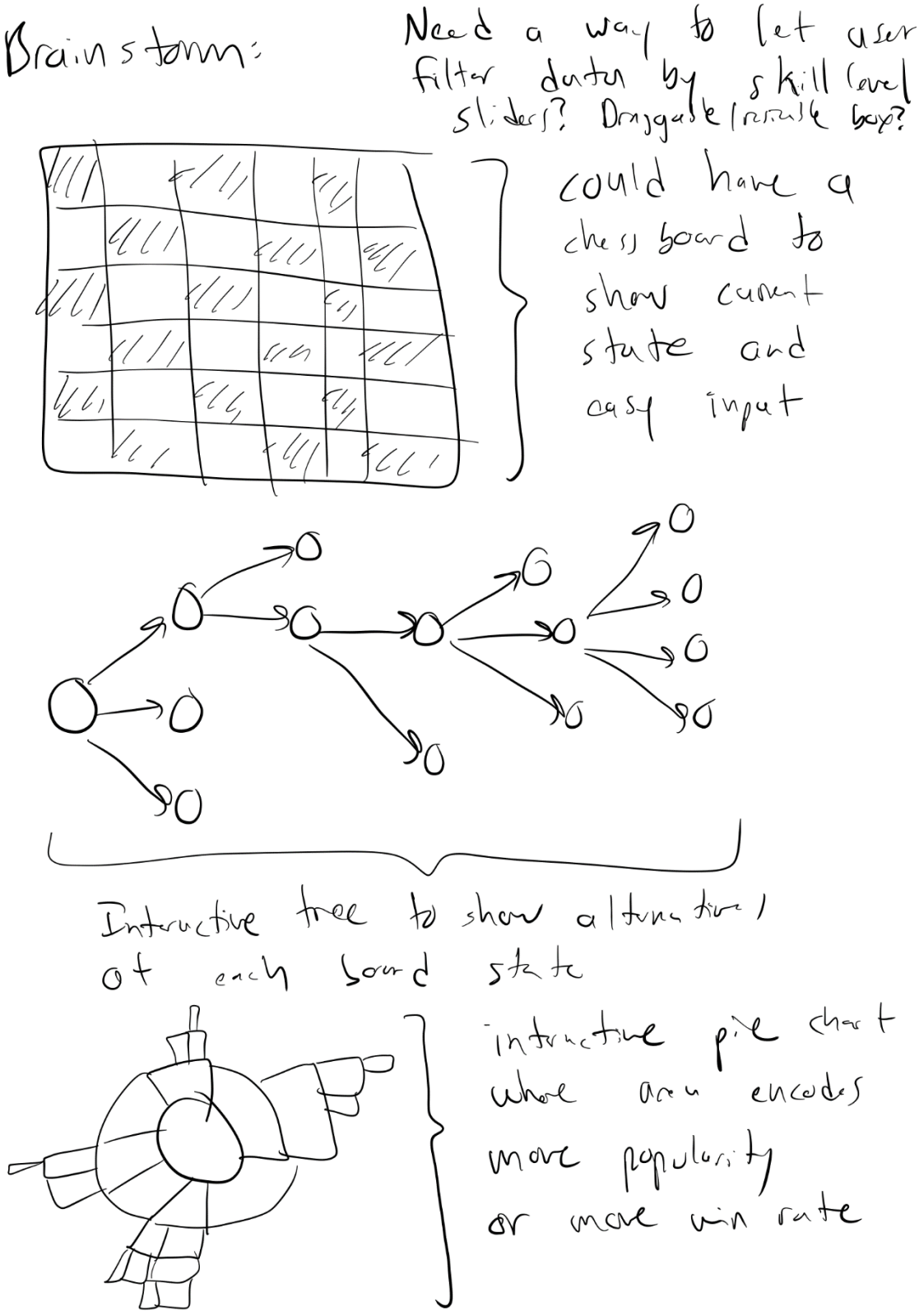
Prototype 3:

I explored a dynamic and layered pie chart in the design. Each slice of the pie chart represents a possible move, where the area of the slice encodes either the win rate or usage rate of the move (depending on the user's preference). When a slice is clicked, more slices appear above the clicked slice that encode possible moves. I have two channels to work with: slice area and slice color. These could be used to encode my two variables. I would use area to encode the more important variable. Although area is not as effective as width for encoding data, the slices of a pie chart will be easier to compare than the width of the edges of the tree found in prototype 1. This is because they share a common axis, unlike the edges of the tree.

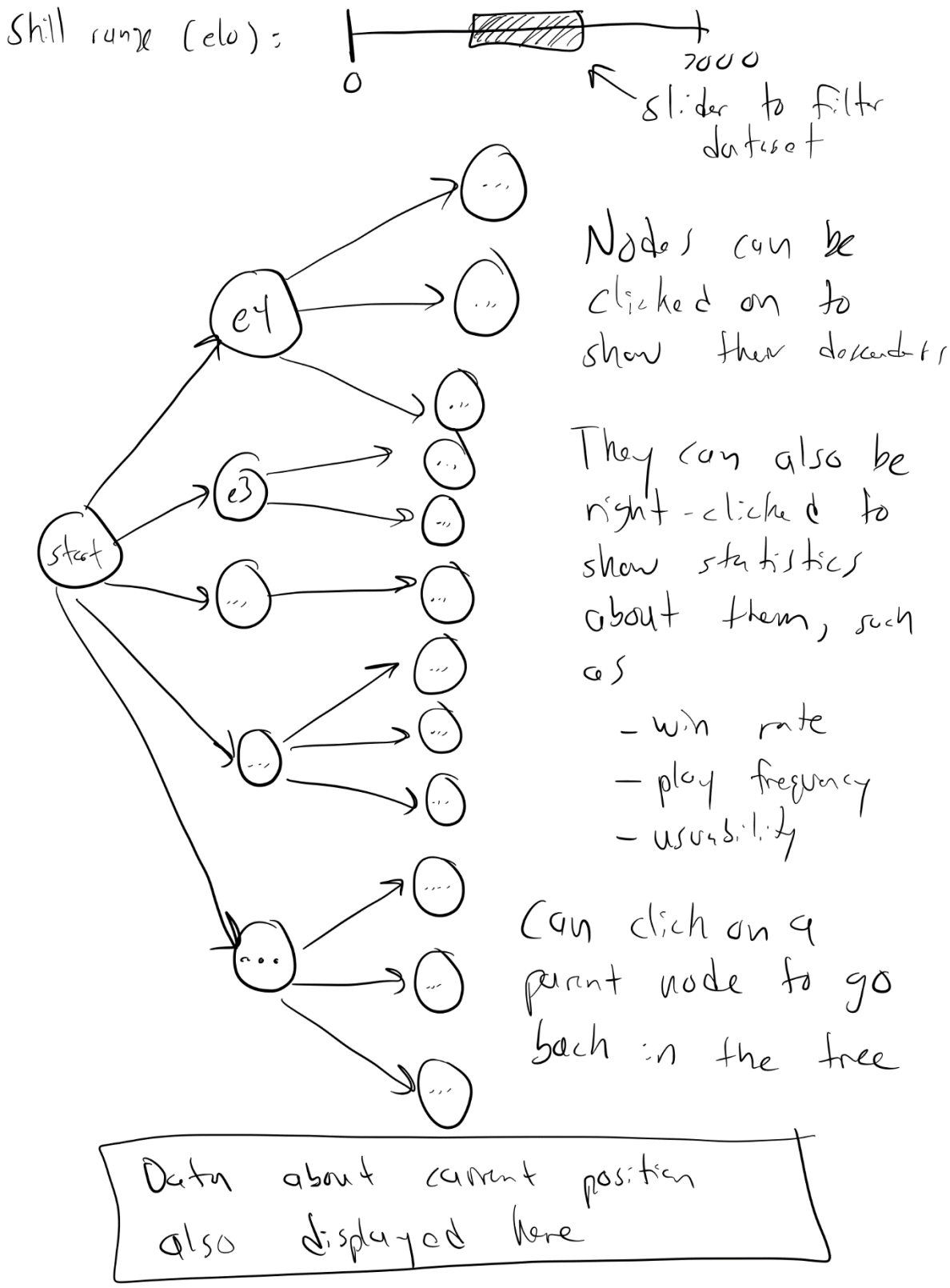
Finalization:

From these drafts, I kept the dynamic pie chart because I believe it is the best way to compare all the moves from a given state. I also kept the chess board as a tool for inputting moves and displaying the current state of the board because this is what any online chess player would be most comfortable manipulating. To filter the games in the dataset, the user can drag and resize a rectangular brush too. This is critical to fulfill the visualization's purpose of showing effective openings for any skill level, rather than just experts. I believe this combination most effectively encodes the data about a given move while still being easy to manipulate.

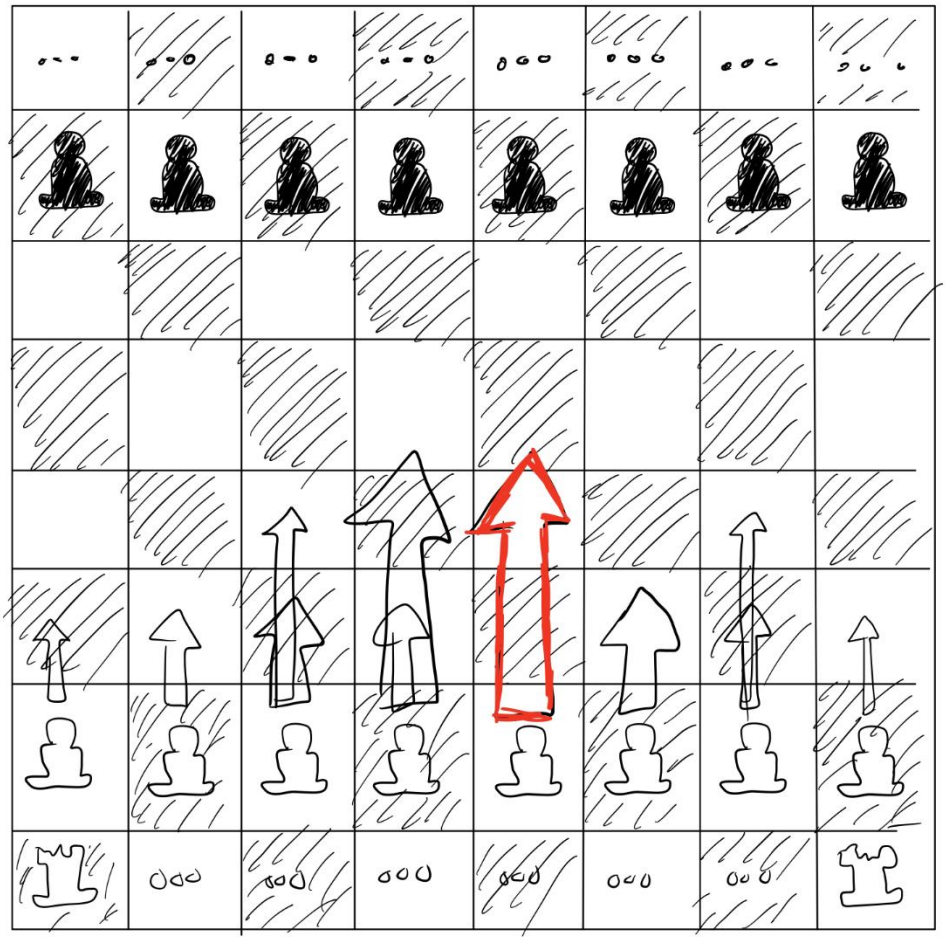
Brainstorm


Brainstorm:
Need a way to let user filter data by skill level sliders? Drop down/radio box?
could have a chess board to show current state and easy input
Interactive tree to show alternatives at each board state
interactive pie chart where area encodes more popularity or more win rate

Prototype 1


Skill range (elo):
0
2000
slider to filter dataset
e4
e3
start
Nodes can be clicked on to show their descendants
They can also be right-clicked to show statistics about them, such as
- win rate
- play frequency
- usability
Can click on a parent node to go back in the tree
Data about current position also displayed here

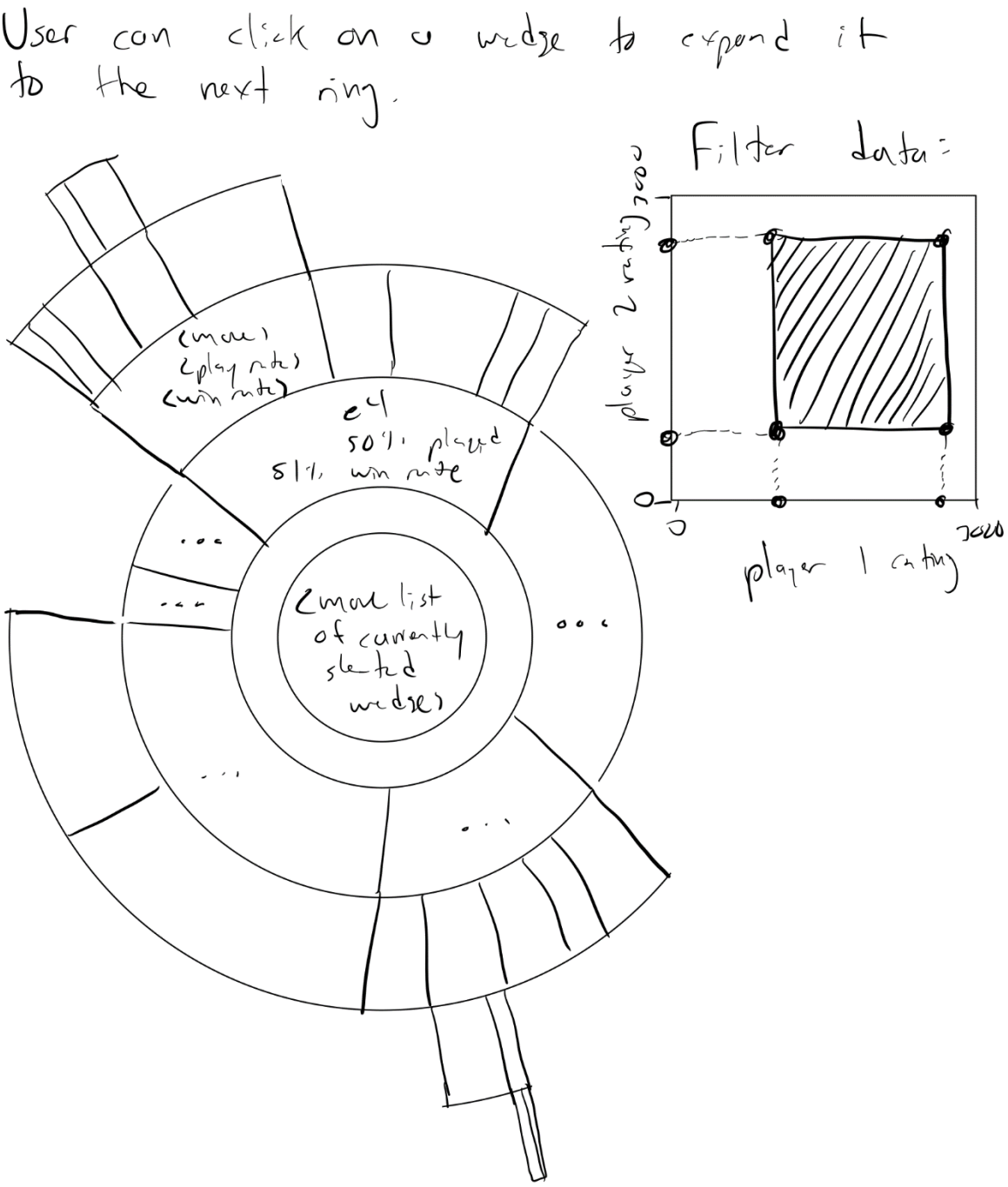
Prototype 2



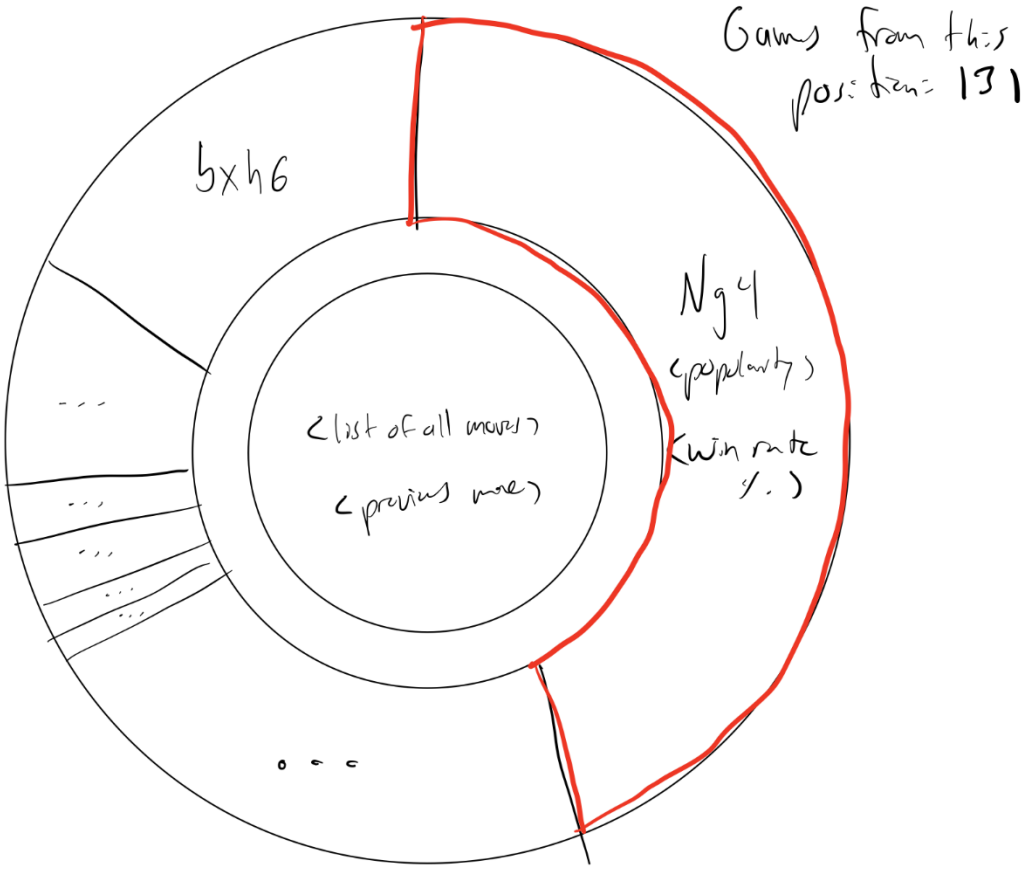
Popular moves

mod	# played	success rate	avg elo of user →
e4	60	X	X
e3	10	X	
d4	20	X	
d3	..	X	
...	...	X	
...	..	X	

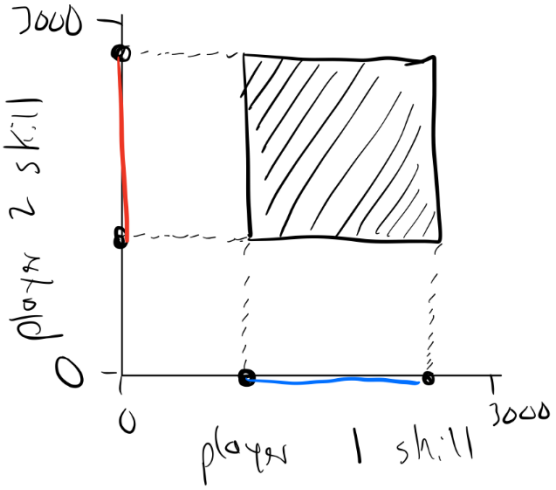
Prototype 3

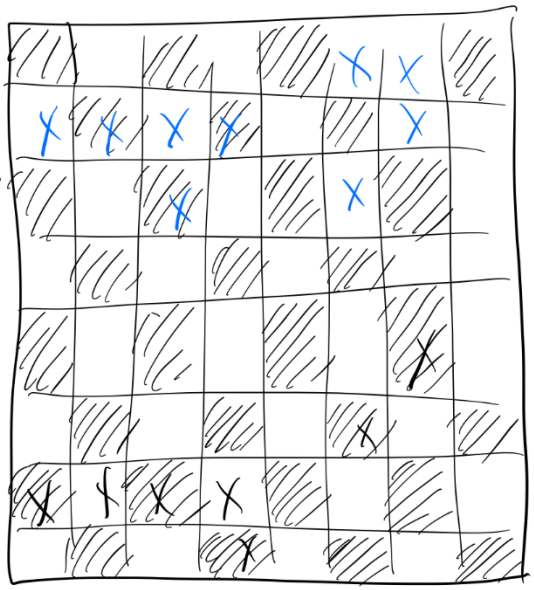

User can click on a wedge to expand it to the next ring.
Filter data:
player 2 rating
player 1 rating
0
3000
(name)
(play rate)
(win rate)
eg
50% played
51% win rate
Emoji list of currently selected wedges

Finalization

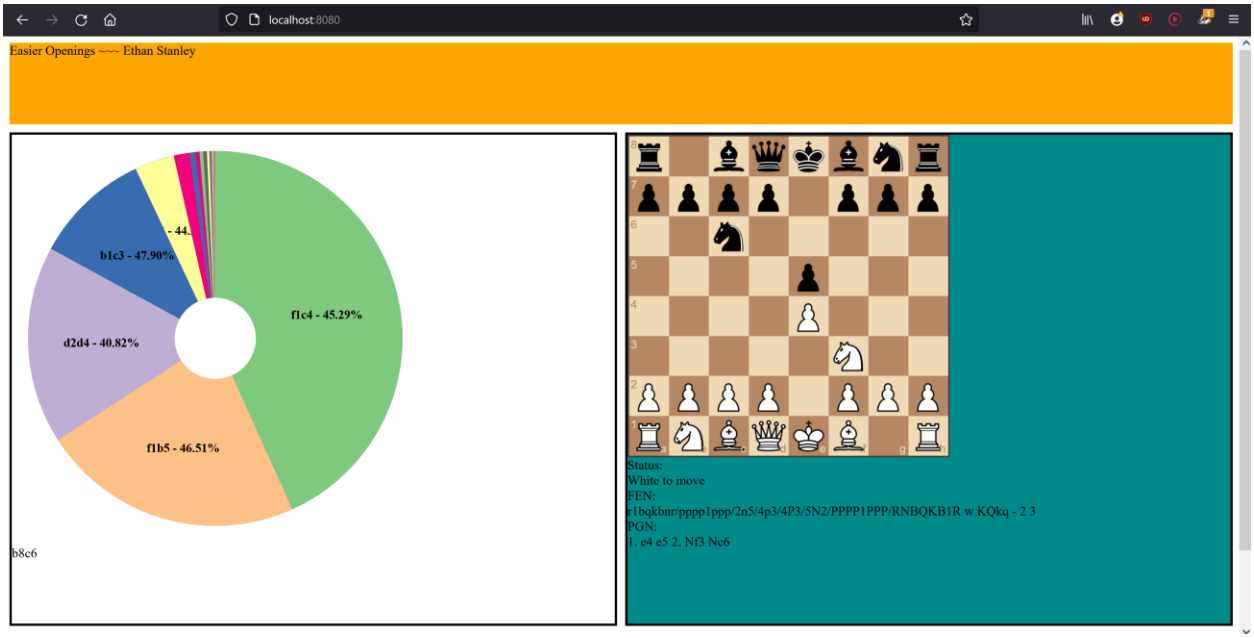


User can click on a wedge on it becomes the outer ring, or make a move on the chessboard.





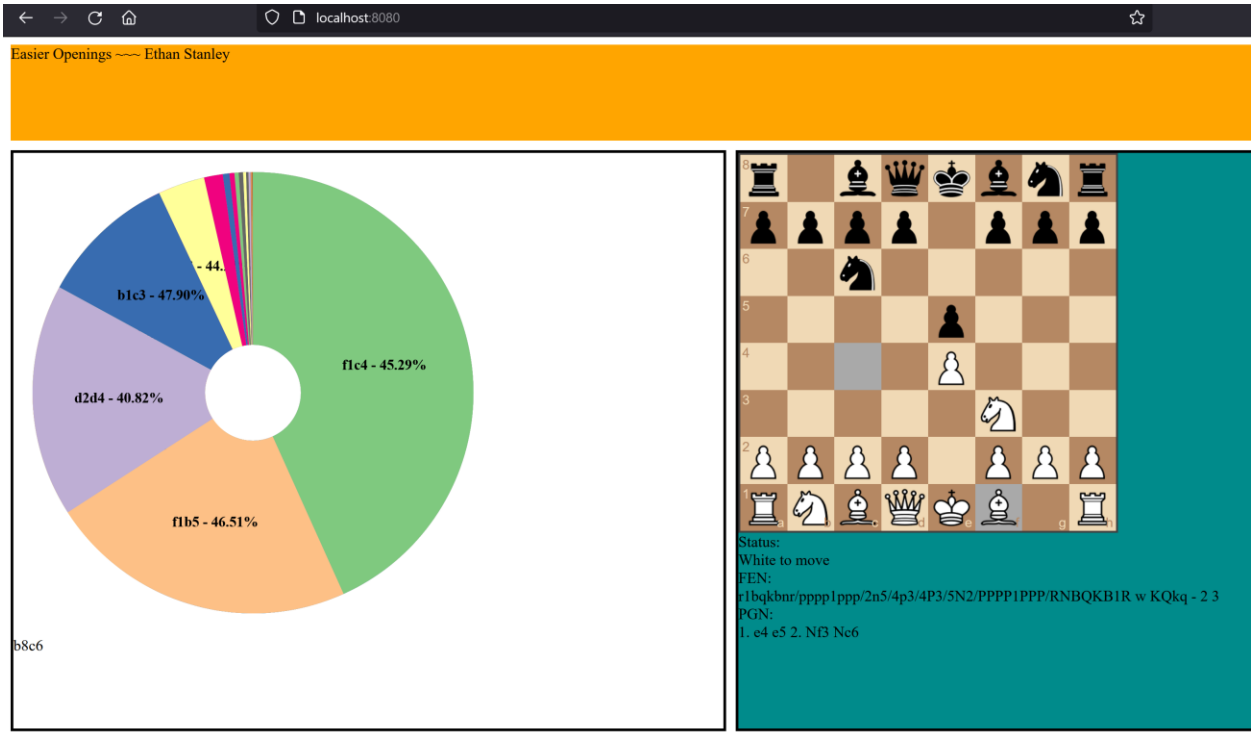
So far, my visualization resembles the final design plan:



It is unpolished but the major elements are there: a pie chart on the left and the chess board on the right. The chess board had text fields below it that encode the state of the game such as whose turn it is and what moves have been played. The pie contains a wedge for each move that has been played from this board state. It also lists the win rate of these moves.

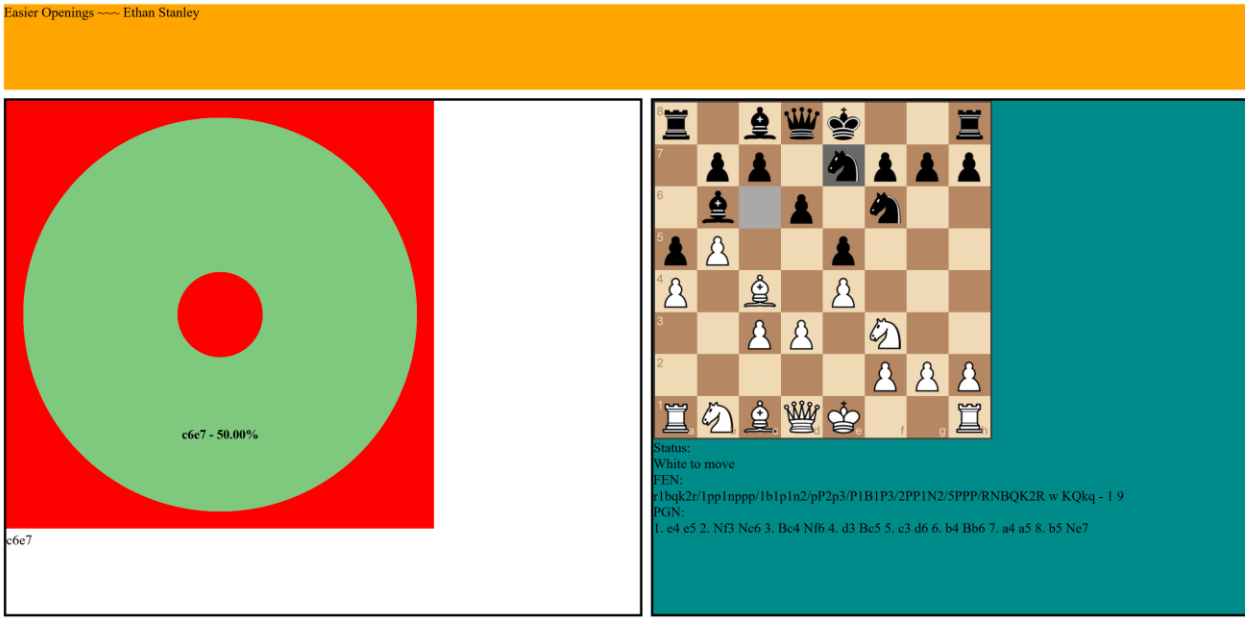
Ultimately, I will add the ability to filter by skill level with a slider in the header (orange) box at the top of the page.

Implementation:

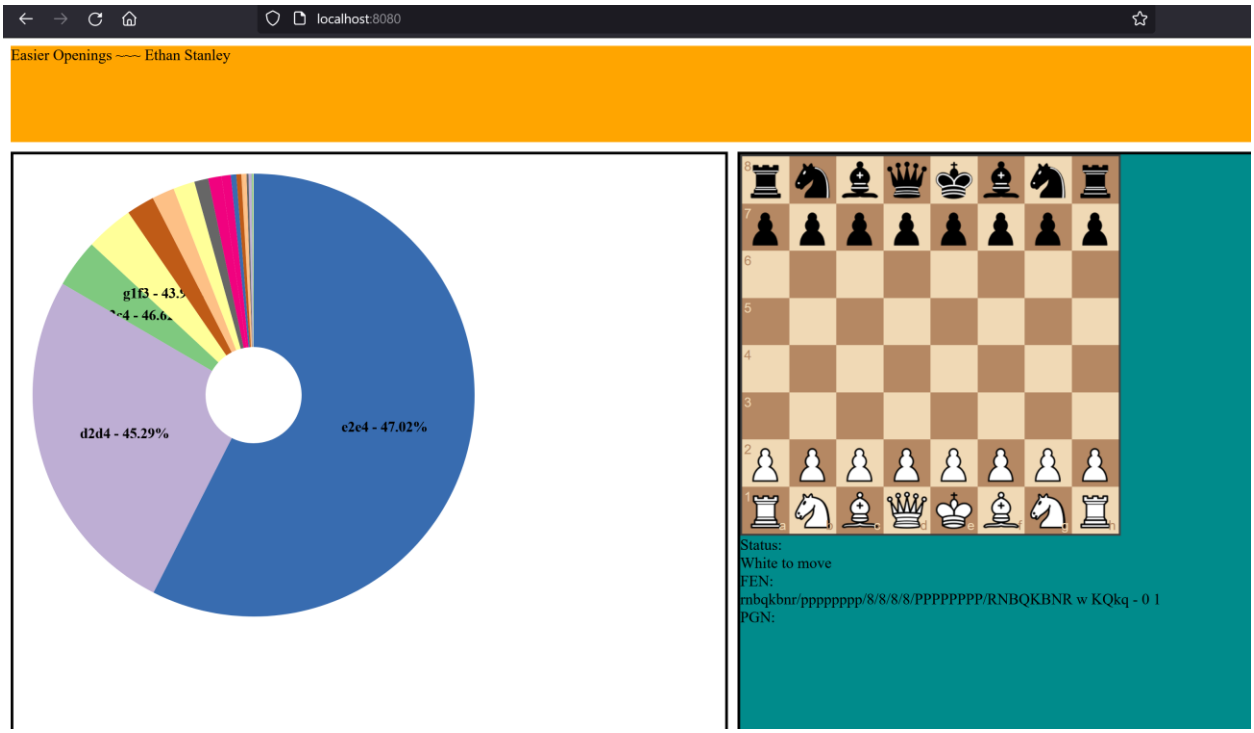


This image shows the major elements of my visualization: a donut chart showing every move in the database that was played from this board position, and a chess board showing the current position. These visual components are linked such that clicking on a wedge of a pie chart (which represents a move) plays that move on the chess board. Similarly, making a move on the chess board updates the pie chart to represent that board state. A feature that you can see in this screenshot is that hovering the mouse over a wedge of the pie chart highlights that move on the chess board (my mouse is hovered over the green wedge and the f1 and c4 squares are highlighted on the chess board). In the future, I will add a tooltip to the wedges as well.

The wedges of the pie chart are sized by how often a move is played from this position. The text in the pie chart denotes the move the wedge corresponds to and the win rate of this move (what percentage of time the person who played this move went on to win the chess game).



This image shows what happens when the maximum depth of the opening tree has been reached. The background of the pie chart turns red and the chess board is no longer interactive. In this case, 16 moves were played before depth was reached. This is listed in the text fields below the chess board.



This is what the visualization looks like when first loaded. When can see that e2e4 is the most commonly played move in this database (it has the largest wedge). Despite this, it only has a 47.02% win rate.

Evaluation:

So far, I have seen a general trend that win rate and usage rate correspond well to each other earlier in the tree but later on, the move with the highest usage rate rarely has the highest win rate. I think this is because people can study and remember which moves are good in the beginning of the game, but later on the game becomes more spontaneous. Thus, the best moves are not as well known.

A notable exception is that the most common starting move (e4) only has a 47.02% win rate. I found this interesting.

So far, I am pleased with how my visualization works. The major components work (the pie chart and the chess board) so all that is left to do is compute more filtered data, add transitions, and then polish the style (it's pretty ugly at the moment).