NBA GAME TRACKER VIS

PROCESS BOOK

James Brissette & Chris DuHadway

1. Project Background and Description

As described in our initial project proposal, our team has involved (separately) in various sports-themed data analytics projects, but never have we had the opportunity or skillset required to visualize such dense and often complex statistics.

When given the choice of project topics, there was a natural draw to the sports realm and now having decided upon visualizing the NBA franchise, we excited to present our project: the NBA Game Tracker.

2. Primary Objective

- Create a visualization that allows for the exploration of positional player information and summary statistics across professional NBA matches in the 2015/16 season

To date there doesn't exist a quality visualization tool that tracks player location. When taken in concert with other derived metrics including player passing and positional heatmaps, the value exists in the ability to visually interpret and analyze player movement. This is not dissimilar to football coaches who watch game film and track movement and results. This visualization allows for interactivity with the data provided and semantic zooming between varying levels of abstraction.

3. High-Level Requirements

From the beginning we wanted our project to incorporate three positional visualizations of the game in time and allow for a tasteful semantic zoom between season, game, and player specific stats. We want the visualization to allow users to observe trends at multiple levels of abstraction.

PROCESS BOOK
TABLE OF CONTENTS

INTRODUCTION TO THE DATA

SOURCES

The data collection initially was a beast of a task to tackle. Fortunately for us the NBA formed a partnership with a third-party sports analytics firm called SportVU through which the began several years ago installing multiple high definition cameras in the rafters at all the major NBA arenas. These cameras in combination with a sophisticated suite of analytics software allowed for the NBA to begin publishing both individual player and aggregate team/franchise statistics for independent analysis.

Among the stats tracked and measured was player movement, meaning the physical (X,Y) coordinates of any given player at any given time on the court. This data, while not directly advertised was available through the NBA's complicated API for any individuals savvy enough to locate and extract it. However, as mentioned on their website, starting in the 2016-17 season, some technical malfunction (or perhaps intentional omission) prevented this data from being recorded or available[1]. In addition, the links to the existing datasets were removed and are no longer searchable.

1. Positional Data

Before the technical errors facing the NBA equipment manifested themselves, the concept of analyzing player positional information began to be of great interest and there was one GitHub user who downloaded all of the available data for the 2015-16 NBA season and published it in a public repository. The data was not well advertised and so locating it took more than a little while to accomplish. Ultimately we were able to download and verify its authenticity versus actual game footage from that year.

Currently the data is split into some 664 games and follows a very complicated (for the purposes of parsing) structure:

[1] Change in availability published on NBA site: https://stats.nba.com/help/whatsnew/

```
{
        “gameid”: “0021500xxx”,
        “gamedate”: “2016-xx-xx”,
        “events”: [{
                        “eventide”: “1”,
                        “visitor”: {
                              “name”: “Utah Jazz”,
                              “teamid”: 1xxxxxxxxx,
                              “abbreviation”: “UTA”,
                              “players”: [{
                                    “lastname”: “Hayward”,
                                    “firstname”: “Gordon”,
                                    “playerid”: xxxxx,
                                    “jersey”: xx,
                                    “position”: “F”
                                    } ...
                              }
                        “moments”: [[
                              1, ← representing period
                              720, ← representing time left on the game clock
                              24, ← representing time left on the shot clock
                              1xxxxxxxxx, ← some non-essential information
                              null,
                              [
                                    [teamId, playerId, x-coord, y-coord, radius], ←
                        for ball
                                          ... 10 more rows for info of players on
                        court
                              ]
                              ...
                              ]
```

The data continues on in this way for several hundred “moments” per event and several hundred “events” per game.

2. Player Data

Player data was pulled from the Player Data API[2]. This data comprises a variety of aggregated player statistics and data. The API has data for every game, as well as season data, in the below format. This, like most pieces of this project, has interesting and useful data that’s quite painful to parse.

```
["SEASON_YEAR","PLAYER_ID","PLAYER_NAME","TEAM_ID","TEAM_ABBREVIATION","TEAM_
NAME","GAME_ID","GAME_DATE","MATCHUP","WL","MIN","FGM","FGA","FG_PCT","FG3M",
"FG3A","FG3_PCT","FTM","FTA","FT_PCT","OREB","DREB","REB","AST","TOV","STL","
BLK","BLKA","PF","PFD","PTS","PLUS_MINUS","NBA_FANTASY_PTS","DD2","TD3","GP_R
ANK","W_RANK","L_RANK","W_PCT_RANK","MIN_RANK","FGM_RANK","FGA_RANK","FG_PCT_
RANK","FG3M_RANK","FG3A_RANK","FG3_PCT_RANK","FTM_RANK","FTA_RANK","FT_PCT_RA
NK","OREB_RANK","DREB_RANK","REB_RANK","AST_RANK","TOV_RANK","STL_RANK","BLK_
RANK","BLKA_RANK","PF_RANK","PFD_RANK","PTS_RANK","PLUS_MINUS_RANK","NBA_FANT
ASY_PTS_RANK","DD2_RANK","TD3_RANK"]
```

[2] http://nbasense.com/nba-api/Stats/Stats/Player/PlayerGameLogsStats#request-pa

3. Team Data

Team data was pulled from the Team Details API[3]. This contains any team-wide stats that we originally required and has additional information that may be useful should we require it while building our visualization moving forward.

PREPROCESSING

1. Player Movement

Player movement is derived from positional data. Given that we already have position at approximately 25hz, tracking movement is relatively easy.

2. Play-by-Play

The play-by-play data was a little trickier to acquire as the API regularly rejected requests and returned 404 Forbidden errors. Ultimately, we were able to pull it using a python script that simulated human interaction with the browser to download the information and preprocessed it into a format in which we could then merge it with the player movement data. This allowed us to correlate ball position to actual goals as opposed to the ball simply being under or near the rim.

3. Passing Data

A significant piece of our player-specific visualization is reliant on passing data, which would be implicitly encoded as possession information. Our original dataset claimed to contain possession information, but unfortunately this was not this case. In order to approximate this information, we preprocessed our original dataset and approximated possession data. Although this passing data is not perfect, visual analysis of our results and game footage lead us to believe that it's accurate enough to produce and analyze trends in passing.

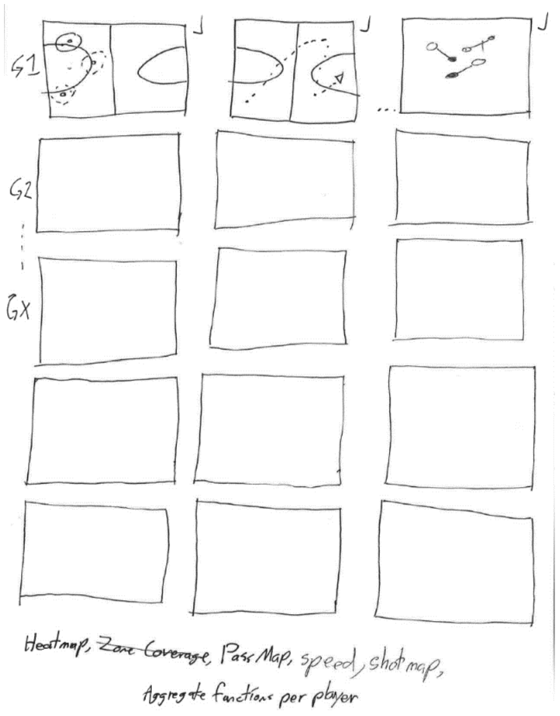
[3] http://nbasense.com/nba-api/Stats/Stats/Team/TeamDetails

PROCESS

DEVELOPING THE CONCEPT

As we sat down to decide what exactly we wanted this visualization to entail we leveraged the Five Design Sheet Methodology and first organized our thoughts per the outline. At the end of our activity we had finalized a skeleton of both the visualization layout as well as a list of vis elements that would be essential in each view. The goal was to create a structure that allowed for semantic zooming across several layers of abstraction. This would allow for users to explore the entire season on a high level and on a team by team basis as well as drill down into individual player statistics, movements and impact on a player by game level.

We devised three high level layouts that would users would be able to navigate between to achieve this desired analyzability. As we set out to begin constructing each outline, out thoughts were formalized into the following sketches:


G1
G2
GX
Heatmap, Zone Coverage, Pass Map, speed, shot map,
Aggregate functions per player
Figure 1: Aggregate Season Data by Team

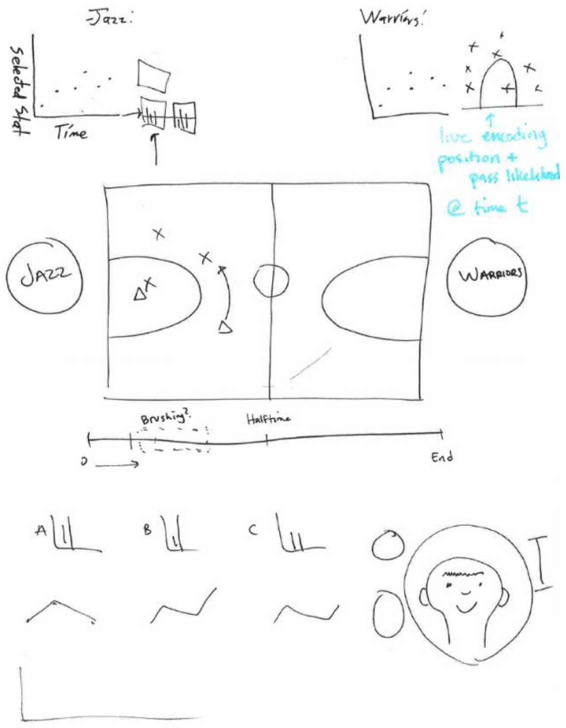


Figure 2: Team vs Team Data at Game Level

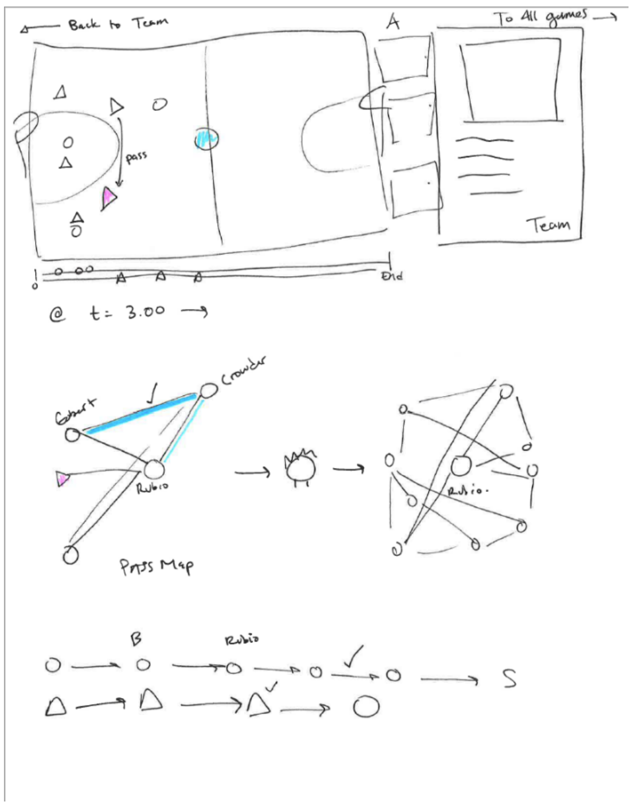


Figure 3: Player Level Data per Game

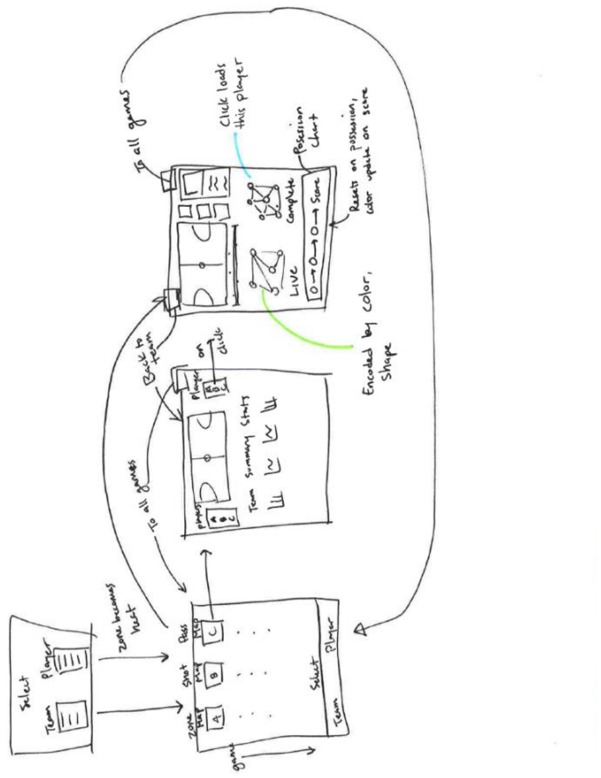


Figure 4: Interplay Between Levels of Abstraction

1. Feedback

During the peer review session our team met with and were critiqued by **Andrew Miller** and **Michael Watkins**. They were very impressed by the concept and excited to see how the final product would turn out. During our conversation they hit on a few points that on second look, and throughout the preprocessing phase of our project, has proven to have a lot of merit.

The primary concerns and suggestions they had focused on the scope of the project and their hope was that in the event it proved to be too data intensive to visualize an entire season that we focus perhaps on one team, or in the extreme, one game. In their words, the hope was to “start small to begin with and expand from there”. As we’ve continued to comb through the data and evaluate the amount of resources required to load and process 600+ games, we’ve come to realize that this project will only come to life if, as was suggested, we start with one game, and build from there.

Additional considerations raised in our meetings reflected potential end user preferences that would allow for easier interpretation of the visualization. This included additional uses of color for categorical means that would allow users to easily identify the efficacy of passes made between players. This comment was a specific suggestion made in reference to the “pass-map” visualization we hoped to implement (a force directed node-link diagram with the selected player at the center and weighted/colored links identifying passes made to teammates).

A final comment they asserted was that they hoped we would preprocess the data in such a way that we could allow for the “live” plays and “live” player tracking metrics to be optional in the case when an end user simply wanted to see the final heat maps of player movement. The comment in their words was as follows, “maybe you can select a player and see just a static heat map for position or passes, or select a team and see a static heat map of their zone coverage.” We’ve taken that particular point seriously and hope to have these implementable and optional to users.

With this feedback, we were able to have an initial direction with which to begin constructing the visualization, ultimately deciding on the team vs team view first.

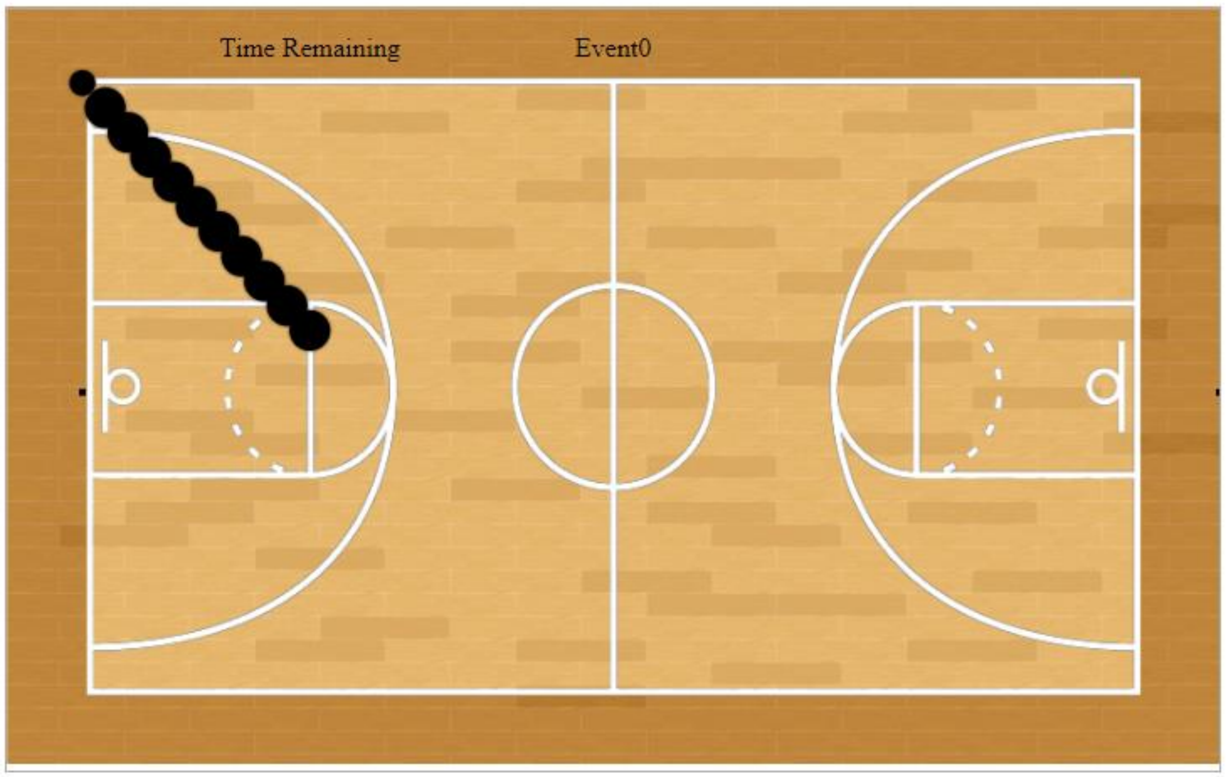
DESIGN EVOLUTION

Our original design was ambitious and unique among most designs proposed in this class. As such, we planned to be especially responsive to feedback. Universally, from conversations with TA’s, to student evaluations, the idea and design was praised, but the scope was questioned. The immediate worry was that visualizing the entire season of data would be difficult due to the quantity of data, and that our primary visualization idea wasn’t actually reliant on season-wide visualizations. We agreed with this feedback, and immediately moved our season-wide views to nice-to-have. This idea paid off in the long run. Instead of spending a semester parsing, saving, and handling 500 gigabytes of data, we were able to build the visualizations that we were interested in, and spend enough time on them to build them well.

CONSTRUCTING THE VISUALIZATION

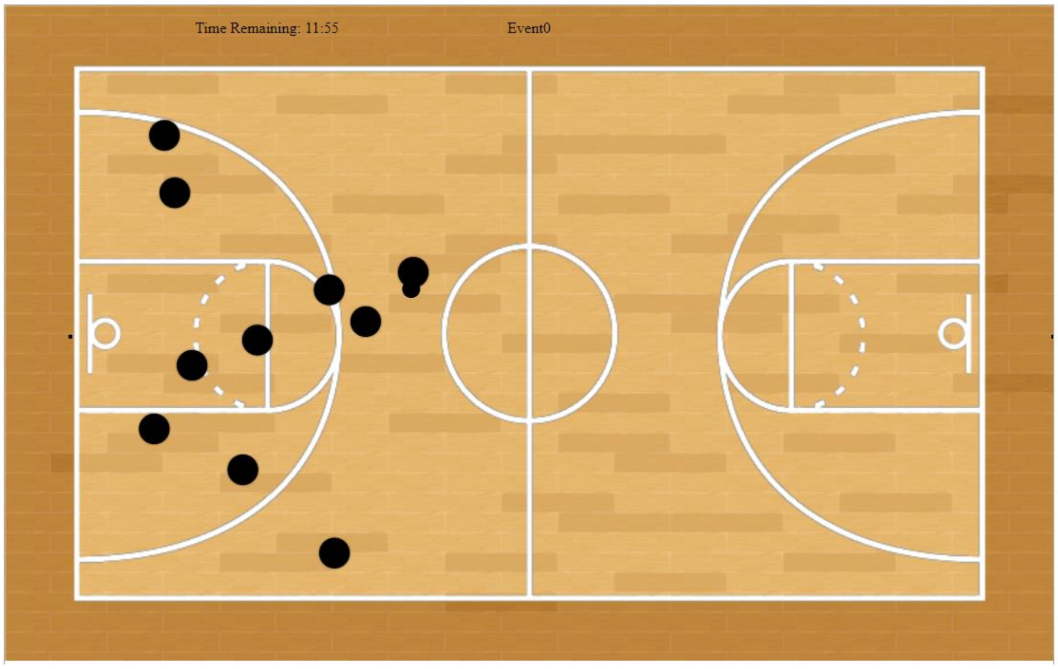
1. The Court

Given the uniqueness of our dataset, and the ready availability of other player and team metrics hosted by the NBA's stats arm, we knew the focal point and centerpiece of our visualization would be the live player tracker. As such the first step was to prove the concept by building a working visualization of a single game from the 2015-16 season. The example chosen was the Jazz at the Warriors and it proved to be more difficult than initially anticipated.



Initially, referencing player information and positions proved to be difficult using the native format of the data, and so each "moment" was boiled down solely to the positions so as to allow us to work with a simple array with which we could loop over. Drawing the players was trivial to accomplish, however, making them move was an entirely different struggle.

We tried first to move them using an infinite tick loop in combination with JavaScript's native requestAnimationFrame() command as was described in one of the lectures (http://dataviscourse.net/tutorials/lectures/lecture-javascript/), but the request failed to update player movements and ultimately resulted in a stack overflow as events continued to build up without execution. While this was the cause of a great deal of frustration, we stumbled upon an amazing example of animating object using d3 (https://bocoup.com/blog/smoothly-animate-thousands-of-points-with-html5-canvas-and-d3) and this was the key. We discovered d3.timer and were able to successively call an update on our court that at each execution redrew the players and suddenly, we had movement:

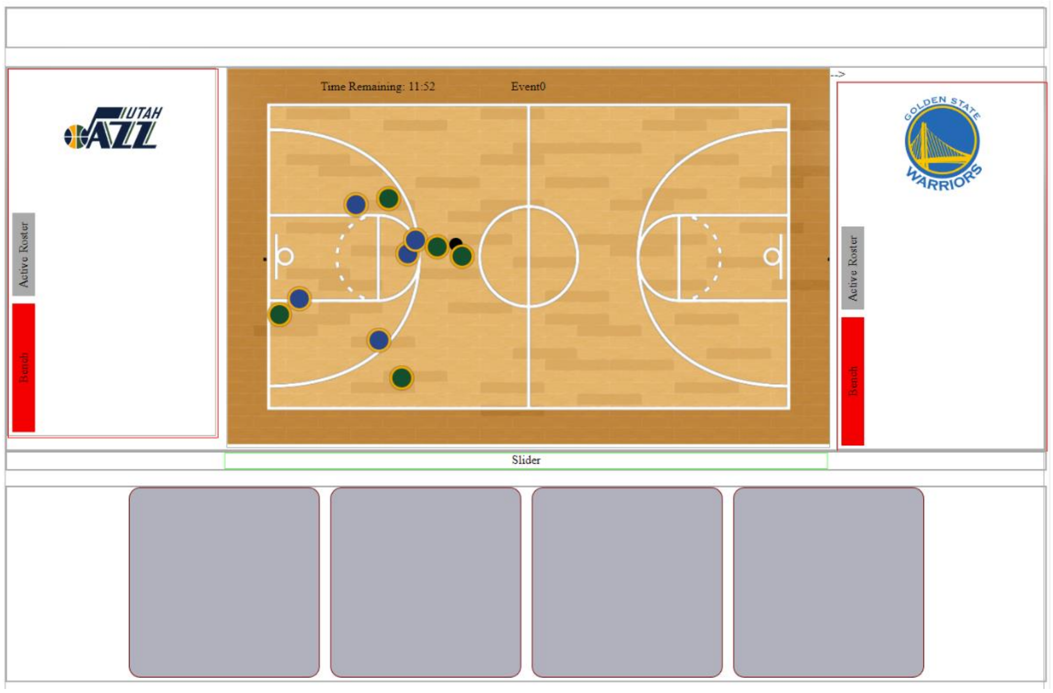


Our timing function currently is set to run for a fixed amount of time, and as we continue to develop the hope is that we will have the capacity to put controls in place that allow for more flexibility in terms of rewinding and skipping around:

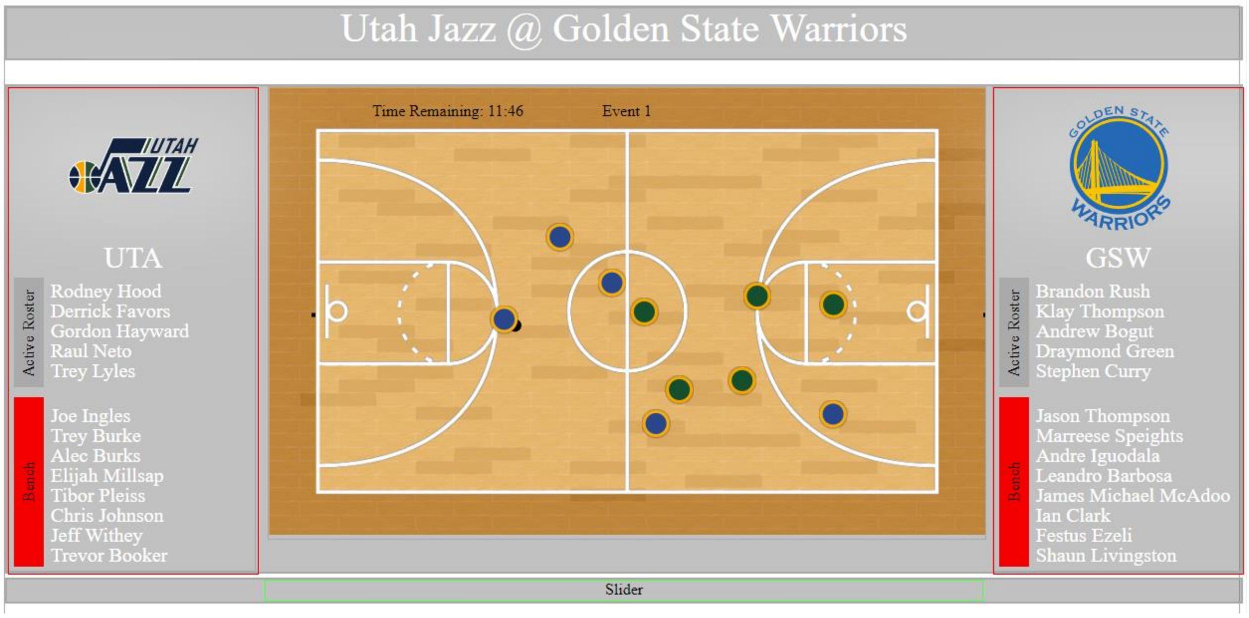
```
let timer = d3.timer((elapsed) => {
    timerCallback(elapsed)
});

function timerCallback(elapsed) {
    let t = Math.min(100, elapsed / 1000);
    if (draw) court.update();
    draw = !draw;
    if (t == 100) timer.stop();
}
```

2. The Teams

With the proof of concept established we set out to build a home for the visualization and to manage player identity so as to later allow linking. This was accomplished with a series of scripts, additional HTML elements, and a fair bit of CSS, but resulted in something like the following:



From here we needed a way to manage players as they were on the court, and we created separate scripts for each. One for the teams as they appear on either side of the court, and one to keep track of and update players, both as they rotated into the game and as they were selected by users. We developed a method of identifying and updating the active players on the court:

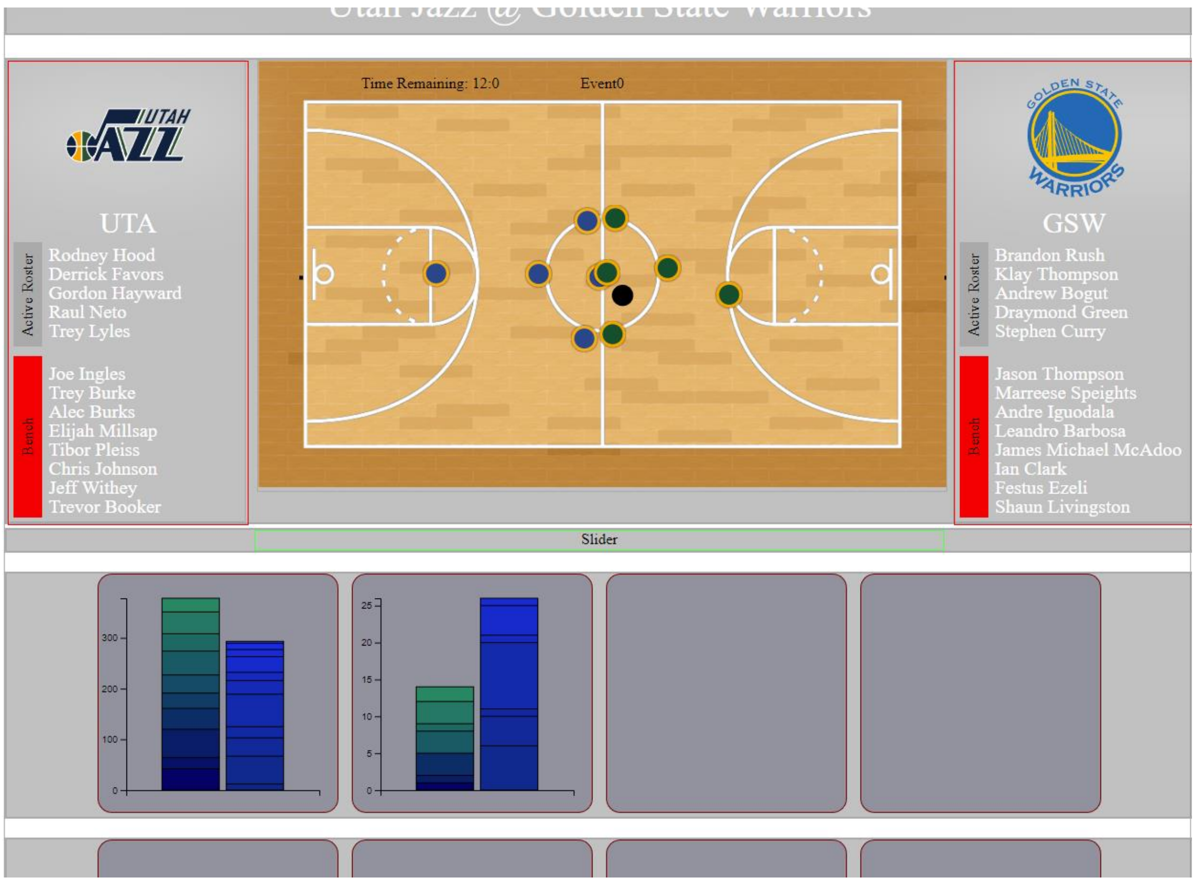


3. The Analytics

Our next objective was to instantiate linked visualizations, but we realized that we had yet to implement the various statistical chart on the bottom half of the screen, and so we ventured out to create a method that would automate the process and cut down on duplicate code.

Again, this required a trip to the NBA stat repository and a fair bit of preprocessing to augment each file into a format that our stackedBarChart class could parse. The idea was to create one class that took in standardized information and pumped out a chart where every aspect was handled. This made for great consistency across charts.

We tried this with two sample datasets as a proof of concept, namely passes and assists. We passed in the preprocessed data along with locations of where the data should be drawn and the results were satisfactory:



With this we are ready to process the remaining records and begin linking views to create a more complete team vs team visualization.

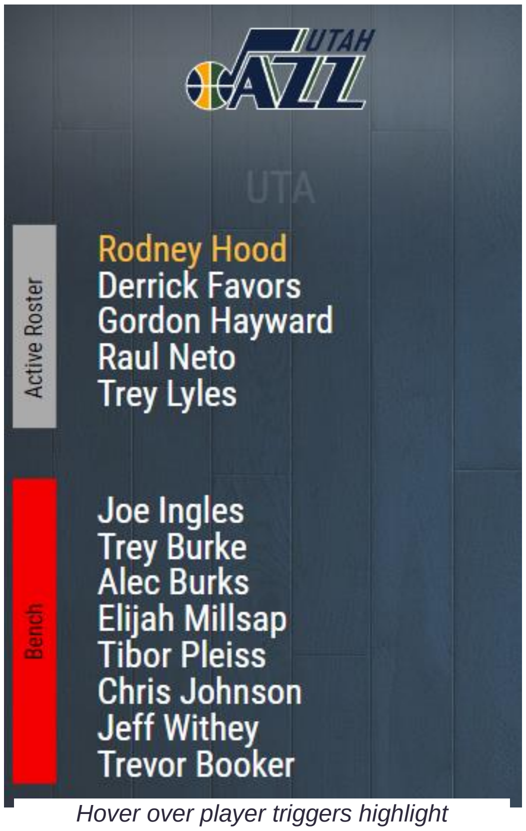
MILESTONE (NEXT STEPS)

Now that we've reached a point where we are able to pass in preprocessed data, we'll need first to finalize the data sets that are to be passed in and optimize the remaining ones to cut down on space usage and speed up loading times. We need next to link the remainder of the views and construct an additional page for player specific information and passing. We have not as yet converted out player movement data into heatmaps, but have a methodology worked out and are ready to begin implementing that as well. If we had to summarize our project in one word, that word would be "preprocessing".

LINKING THE VIEWS

Now having the basic backbone of our primary visualization set up, it quickly became apparent that our main view was missing several key features, and there was an obvious need to coordinate views to give the user the best possible experience. This began by working out the "who's who" of teams and players. Users being expected to interact most with the list of individual players, we used that as the hub for subsequent linking.

We refined the data input being passed to the Team class to include more than what was needed to leverage D3's data-driven creation of SVG text elements used to draw player names. By including full player details we were able to leverage the fact that each NBA player has a uniquely issued player ID which further allowed us to communicate across views often without passing instances of one object to another by encoding this information into class names. Each instance of an object that was acted on by an end user would simply select all elements with a class name associated with the ID of the player selected.



Hover over player triggers highlight

1. Teams to Court

The primary issue here was ultimately deciding on the convention we would use to associate views and how they would be called. The options included dedicated function calls to different views to execute nuanced snippets of code, as well as quick references to other objects to handle simple tasks such as highlighting and triggering selections. Ultimately, we opted to use a combination of methods, the first being a simple highlight based on hover using CSS:

```
vtmBench                                                        JS (Team Class)
    .text(d => d.firstname + ' ' + d.lastname)
    .attr('x', this.teamWidth / 6)
    .attr('y', (d,i) => 35 + 25 * i)
    .attr('text-anchor', 'start')
    .attr('class', d => 'p' + d.playerid)
    .on('mouseenter', d => {
        d3.selectAll('.p' + d.playerid).classed('selectedA',true)
    })
    .on('mouseleave', d => {
        d3.selectAll('.p' + d.playerid).classed('selectedA',false)
    })
```

```
players                                                        JS (Court Class)
            .attr('cx', (d,i) => ...)
            .attr('cy',  (d,i) => ...)
            .attr('r', d => ...)
            .attr('class', d => d[0] == -1 ? 'ball':
                                   (d[0] == teamA ? 'GSW p' + d[1] : 'UTA p' + d[1]))
            ...
            .on('mouseenter', d => {
                d3.selectAll('.p' + d[1]).classed('selectedA',true)
            })
            .on('mouseleave', d => {
                d3.selectAll('.p' + d[1]).classed('selectedA',false)
            })
```

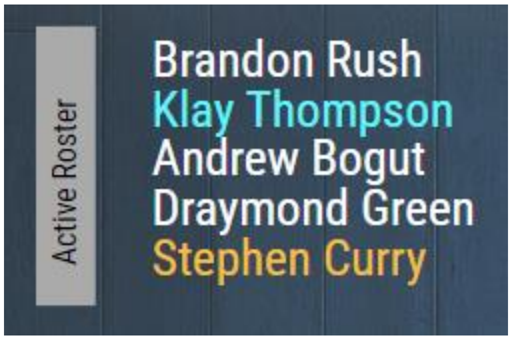
```
.selectedA {                                                               CSS
    fill: rgb(255, 192, 75) !important;
}
.selectedB {
    fill: rgb(48, 248, 255) !important;
}
```

Instantly this allowed us to identify players on the court in a convenient way.

2. Teams to Charts

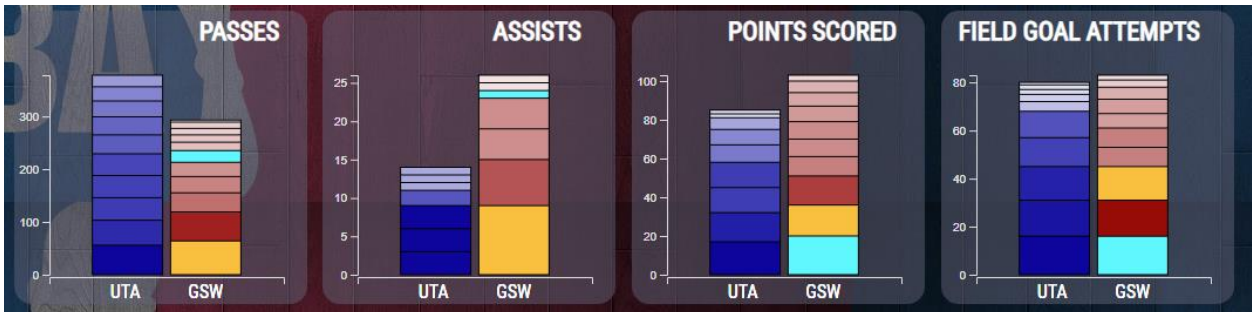
We understood that making our charts smaller would inevitably introduce increase difficulty interpreting the meaning and values. As such we knew before we could link them in a way that would be meaningful, we would need the charts themselves to be intuitive and easy to grasp. This process included a design overhaul as part of additional feedback we received from our TA and serves as the basis for our final layout. This is discussed in more detail in the next section.

Once the charts were set up and ready to go, we used the same logic to allow for highlighting as before, both forwards and backwards. However, this introduced a need that was not as present before, and this was the ability for the user to navigate across graphs while still being able to keep track of who it was they were interested in. From this we developed an additional CSS class – **`.selectedB`** – for use in keeping a player selected indefinitely.



CSS Class	Activated
`.selectedB`	`.on('click',…)`
`.selectedA`	`.on('mouseenter',…)`

This naturally led to the ability to quickly compare two players from the same or opposing teams and to identify trends in performance at a high level (e.g. it would appear that players that shoot the most score the most) as well as drill down into individual player performance (e.g. Thompson appears not to have been a top passer, or among the top players with assists. Does this indicate a ball hog or a go-to scorer?)

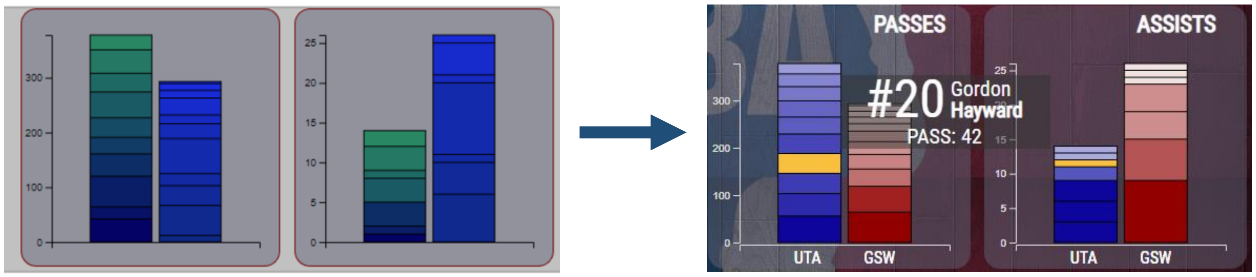


INCORPORATING ADDITIONAL ELEMENTS

1. Chart Tool Tips

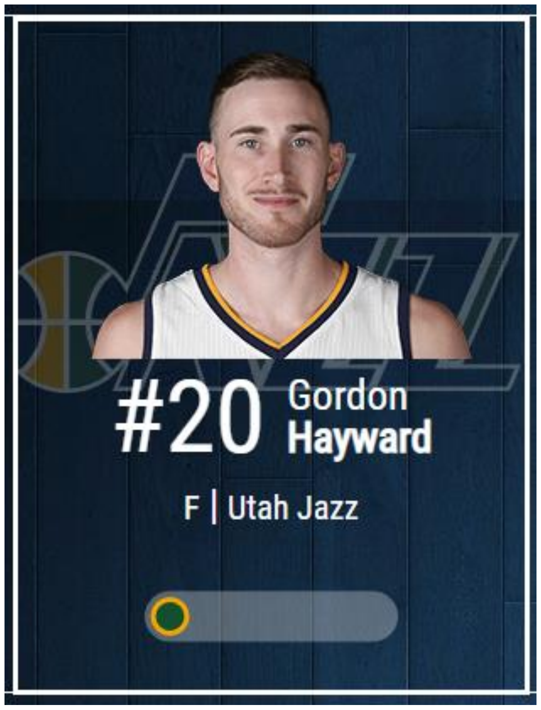
After meeting with our TA, we already had skeleton layouts in place for the physical implementation of our stacked bar charts, and so our thoughts turned to our design considerations. The data collected was meaningful, but not easily digestible and hardly comparable across players or teams. To alleviate this, we developed a consistent scheme in which values were sorted largest to smallest to allow quick comparisons of absolute magnitude, we continued to use position and introduced hue as a categorical encoding of team, and we encoded the actual data value by luminance.

Above each graph was appended a title, and to each plot we appended tool tips to allow for explicit investigation into a given statistic.



We opted to keep the tooltip semi -transparent to minimize occlusion.

2. Player Card



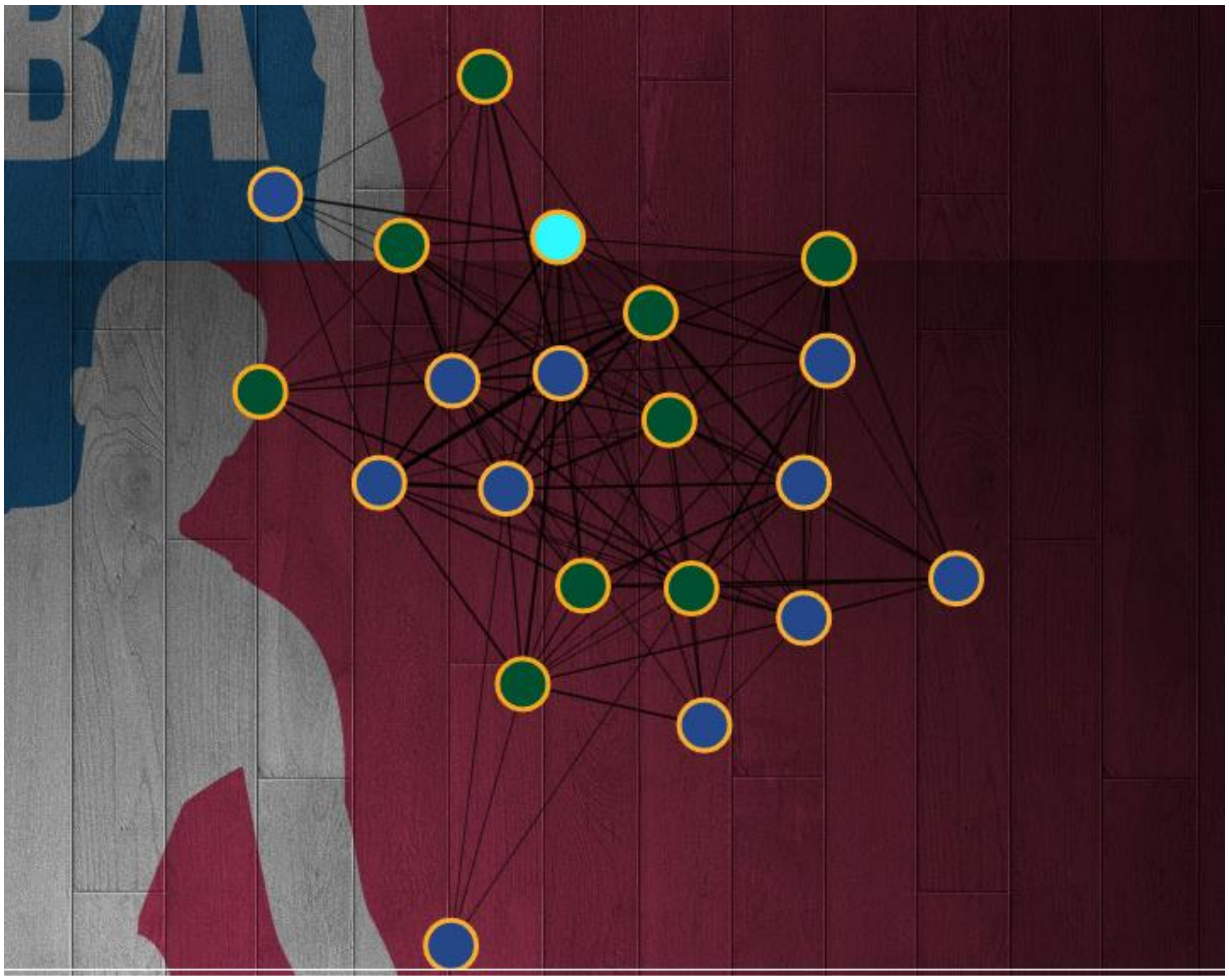
Realizing there was no way for a user to know anything about that players other than there names and stats, we built and incorporated a PlayerCard class that combines a players name, team, team logo, photo, position and jersey number into a compact and attractive view that is easily callable by other classes. This allowed us to let users click on player names and get a richer experience in terms of analyzing players. As an added bonus, this hideaway class allowed for us to incorporate sliders which would let users manipulate the window to reveal additional content previously hidden.

3. Force Directed Graph

Originally, tracking and displaying player passing habits was a central aspect of a piece of our view. To show passing data, we need to know which player has the ball. This data isn’t in the object that describes player position, so we chose to combine our positional data with additional play-by-play data for the same game. By lining up the game time, and monitoring which play-by-play event occurred, we were able to track who had possession of the ball, and therefore track passing data. Unfortunately, we initially did not realize that the play-by-play information that the NBA provides is very inaccurate. The timestamp for each event in the play-by-play can easily be off by 30 seconds from when they actually occurred, and the error varied greatly as well - it wasn’t a consistent shift. As a result, the force graph we created had very little useful information to display. We shifted strategies at this point, and did a manual intelligent parsing of the positional data, and derived which player has possession of the ball for each moment. This was computationally expensive, and performed much better than relying on NBA’s play-by-play data, but is still somewhat noisy. Our parsing correctly identifies all of the passes that occur, but occasionally adds a pass that did not occur. This is usually due to unusual game circumstances, where the ball enters the reach of an individual for an extended period of time, but that player did not in fact possess the ball. This type of scenario is ambiguous, since we only have positional information for the center of mass of each tracked object. Nonetheless, since this data was in fact more consistent then the play-by-play data, we used this to drive our force directed graph. The results are interesting, and with correct data, this type of visualization would be immensely powerful.

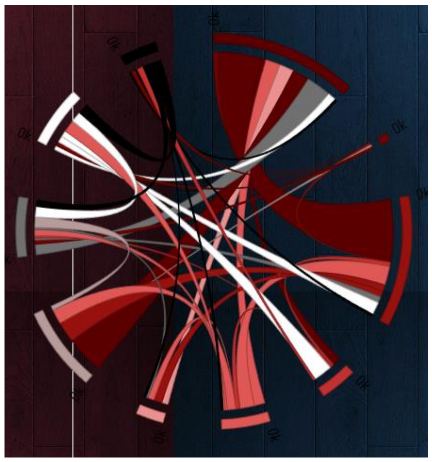
In order to implement the Force Directed Graph, we drew inspiration from the visualization lecture on on layouts and maps.4 The nodes in the graph are players, whose color corresponds to their team. The connections between nodes, and the strength and size of that connection, are dictated by the number of times the ball passed between their possession. Due to the relatively high variance of this number between different players, and since the maximum number of passes between two players is quite large, we log the number of passes before drawing the size of the connection. All of the nodes are draggable, and a custom title displays node information on hover. We had to implement a different tooltip specifically for FDG, because using a standard tooltip locks that node out of force calculation, leading to an error.

After we had a working implementation of the Force Directed Graph, it was clear that alone this display contained little interpretable data. That is why we created two separate chord charts, one for each team. The combination of both is much more effective then our original design.



4. Chord Graph

The two chord graphs were created after the majority of the implementation was done to illuminate data that the Force Directed Graph failed at. By displaying a single team's passing patterns, it more effectively highlighted the pertinent trends, while keeping a Force Directed Graph gave greater overall context when the user requires a broader perspective. Ultimately due to time constraints and the sheer brute force effort required to aggregate piecemeal data we were unable to fully implement these and have as such chosen to exclude them from the final visualization, however, here you see the concept:



5. Heatmap

This was another way we encoded spatial data from the game. Having a live display for the game is excellent, but you lose temporal information very quickly. In order to solve this, we implemented a 'heatmap' function, which would show the historical paths of up to 5 players. This allows the user to analyze player patterns and absolute positional trends, as well as compare those patterns and trends to other players.

EVALUATION

Our project was very interesting to build. We learned much about very different visualizations, how to handle large amounts of data, and how to visualize many different stats synchronously. We set out to build a tool with coaches, players, and sports analysts as the target audience, and we built a tool that is very useful in analyzing basketball games. Our clean stats visualization, along with a temporal abstract representation of the game, our historical tracking heatmap, and our passing visualizations all work together to provide an expressive way to explore data from an NBA game. Our project looks somewhat different from our original proposal, but the process of refining our idea into its core components, and then executing those components to build an excellent display was a very valuable experience.