A CONSOLIDATED PROCESS BOOK FOR CS171

PROCESS BOOK

NOTES, IDEAS, COMMENTS, **PHOTOS.**

THIS PROCESS BOOK IS SPLIT INTO TWO PARTS: ONE A JOURNAL, AND THE OTHER AN EXPLANATION OF FUNCTIONALITY. WE BEGIN WITH THE JOURNAL.

Jason Shen | Angela Jiang | Kit Wu

A CONSOLIDATED PROCESS BOOK FOR CS171

Monday, March 30

ON THIS DAY, WE SEARCHED FOR PROJECT IDEAS AND SETTLED ON WORLD BANK.

- Brainstormed for project ideas
- List of data sources we considered:
  - http://www.socrata.com/top-open-data-datasets/
  - http://data.worldbank.org/
  - http://www.data.gov/
  - http://www.dol.gov/wb/stats/stats_data.htm
  - http://catalog.data.gov/dataset/2010-federal-stem-education-inventory-data-set
  - http://stats.oecd.org/
- Settled on HIV/AIDS data from the World Bank
  - http://databank.worldbank.org/data/databases.aspx
  - Health Nutrition and Population Statistics Database
    - Series: HIV/AIDS, Background, and Population Dynamics
- Considered .xlsx to .json converters
  - http://www.convertcsv.com/csv-to-json.htm
  - http://shancarter.github.io/mr-data-converter/

Wednesday, April 1

ON THIS DAY, WE FURTHER SEARCHED MORE EXTENSIVELY FOR WORLD BANK DATA. WE ALSO THOUGHT ABOUT HOW WE CAN VISUALIZE THAT DATA.

- Types of data:
  - Population data, i.e., who has HIV/AIDS
    - Rows 5, 6, 9, 17, 18, 19, 20
  - Preventative data
    - Knowledge
      - Male, Female
      - Age
    - Condom usage
      - Male, Female
      - Age
    - Rows 2, 3, 11, 12, 13, 14, 15, 16
  - Treatment
    - Rows 7, 8
  - Misc
    - Effects of AIDS (Deaths…)
  - GLOBAL DATA
    - Country
    - Year
- Data Linkage
  - Showing the correlation between knowledge and prevalence of HIV/AIDs among the sexually active age group. (15-24)
- Visual Design Ideas
  - Nodes with color intensity relating to the category
  - Map big picture
    - http://www.nourishinteractive.com/system/assets/general/images/nutrition-facts/childhood-obesity-rates-US-2011.gif
  - Panel with nodes to compare categories of a single selected country
- Final project as model

Jason Shen | Angela Jiang | Kit Wu

A CONSOLIDATED PROCESS BOOK FOR CS171

  - https://37a1fbf47ebfe09a1423d450d76daef4a9ba7558.googledrive.com/host/0B6YljmV-VTJZWlU4RU1hNUJOQzQ/index.html
- Use as map
  - http://jvectormap.com/

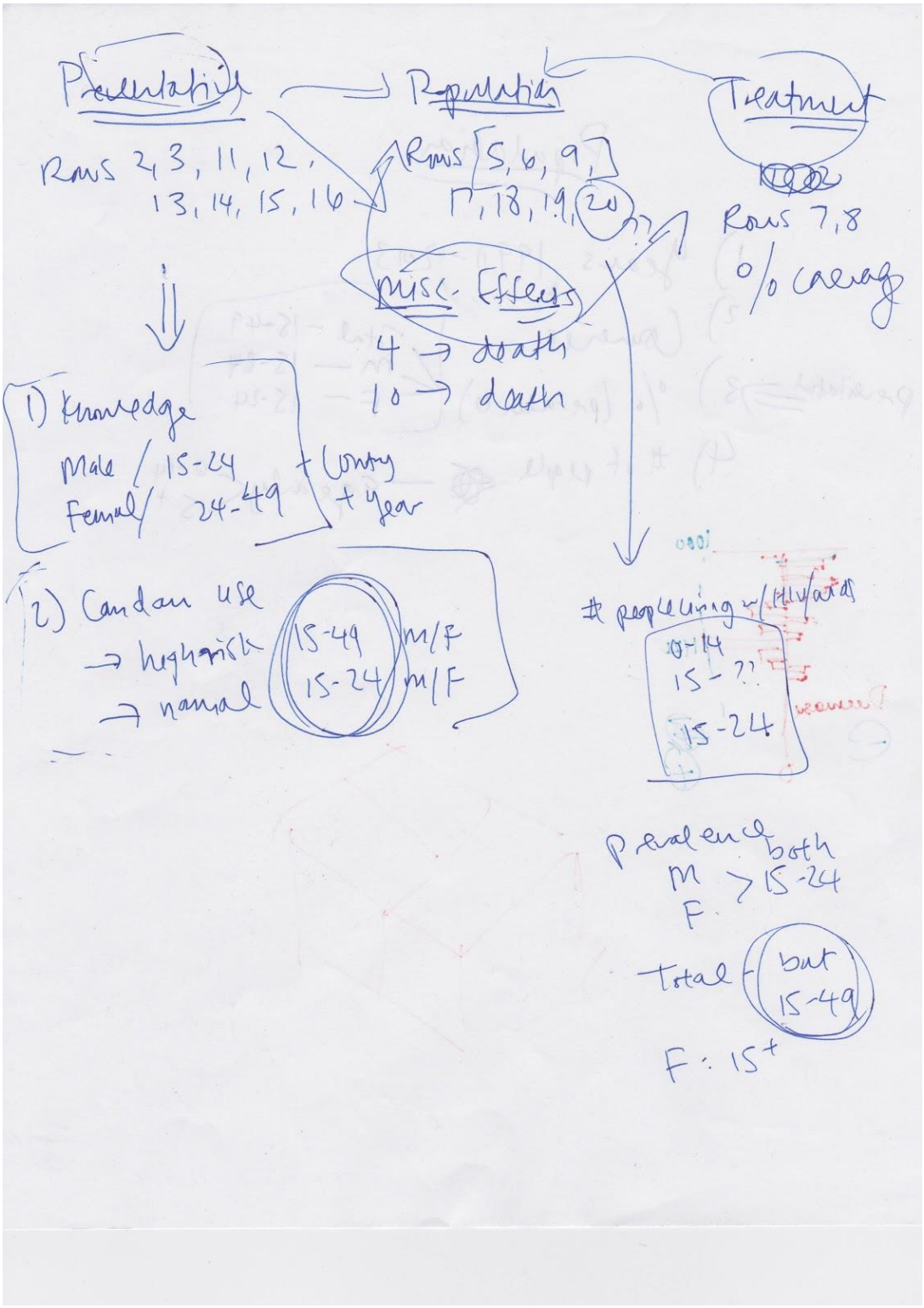
- Scheduling
  - Week of Monday, March 30: Project proposal
  - Week of Monday, April 6:
    - Look at map things and think about how to convert data to what the map would need
      - Consider other data points that we might want
        - What about population data?
          - GDP?
          - Male/female? (not pressing)
    - Data from .xlsx to .json
    - Have basic map showing up
  - Week of Monday, April 13:
    - FIgure out map stuff
      - Nodes
      - Slider
      - Color intensity
    - **Friday, April 17: Milestone 1**
      - Complete map view with year slider
      - Secondary goal: to have everything else on the page but they can just be static
      - For your Milestone you should have completed your data acquisition, or at least have a significant sample of your data. **You must have your data structure ready**. For example, if you plan to collect 1000 data records, but only have 200, that's fine. If you are missing one of two datasets you want to use you will lose points, since you have to have the whole structure.
      - **You must have a working visualization prototype**. You must not have all your views up and running, and it must not be completely interactive, but the direction and the content must be clear.
  - Week of Monday, April 20:
    - Do categories box with nodes
    - Start bar chart
  - Week of Monday, April 27:
    - Finish bar
    - Do line
  - Week of Monday, May 4:

Jason Shen | Angela Jiang | Kit Wu

A CONSOLIDATED PROCESS BOOK FOR CS171

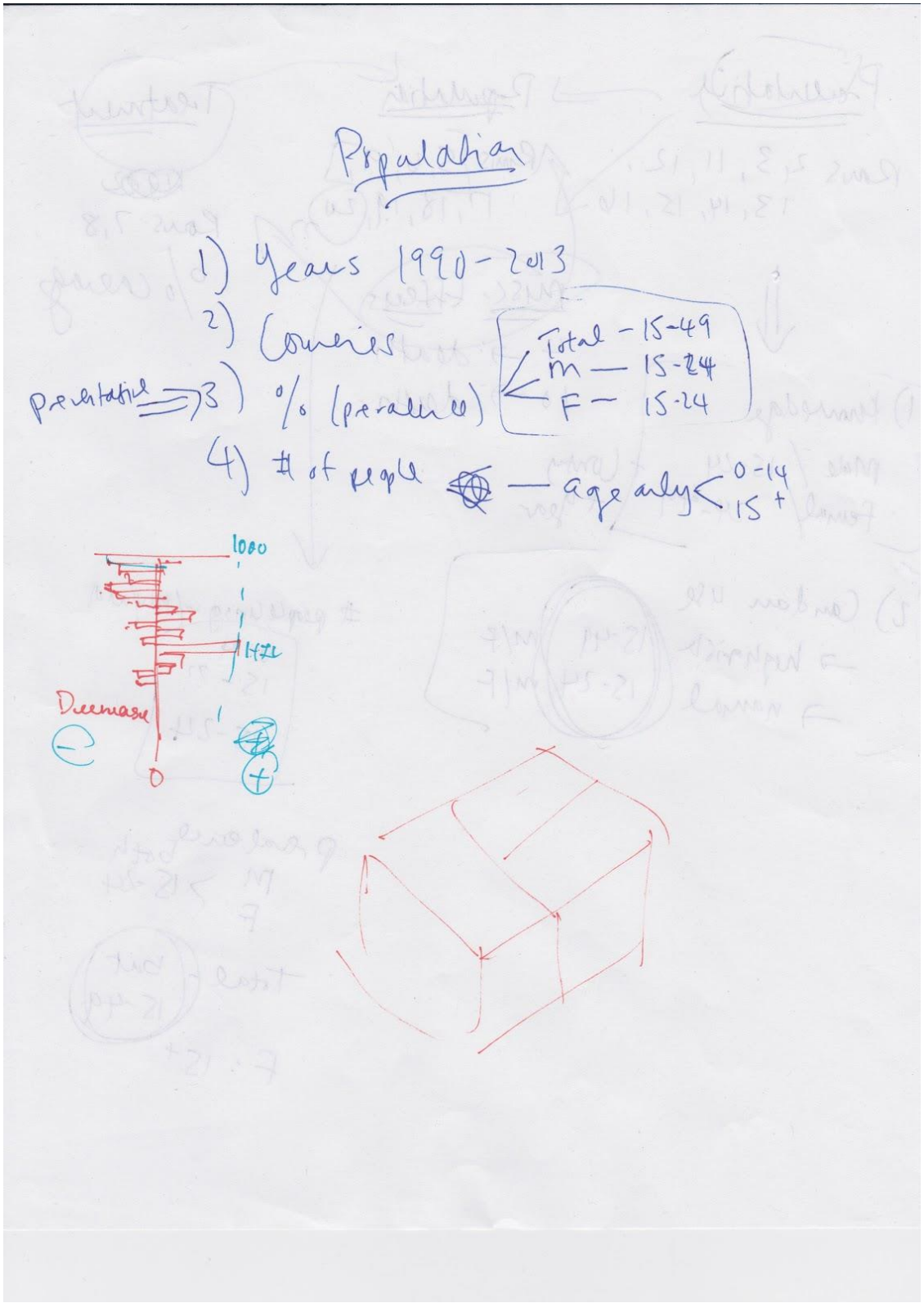
- General debugging, finishing touches
- Tuesday, May 5: Project Due

DESIGN STUDIO



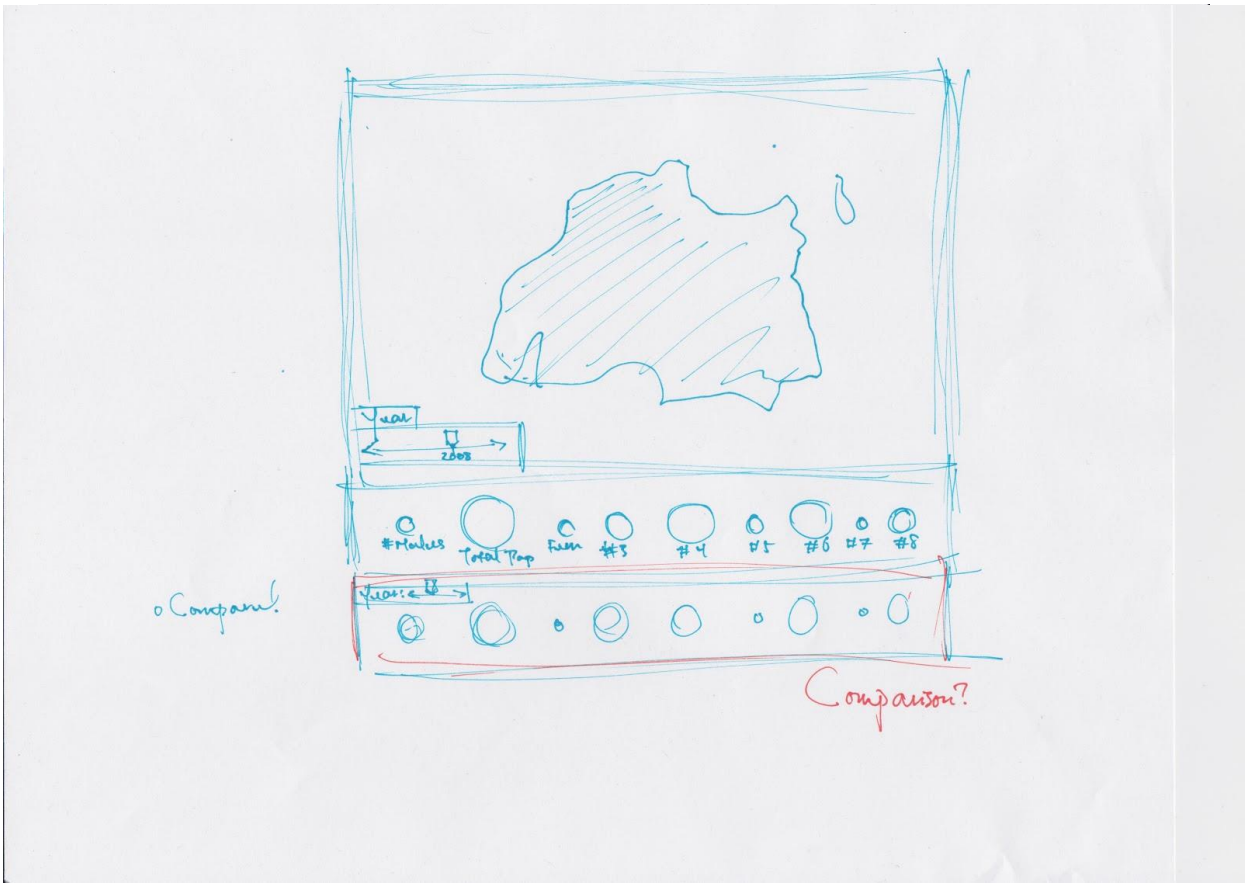
Jason Shen | Angela Jiang | Kit Wu

A CONSOLIDATED PROCESS BOOK FOR CS171



Jason Shen | Angela Jiang | Kit Wu

A CONSOLIDATED PROCESS BOOK FOR CS171



Sunday, April 12

PRIOR TO THIS DAY, WE REALIZED THAT WORLD BANK DATA WAS REALLY SPOTTY. SO WE BRAINSTORMED NEW IDEAS, SETTLED ON SPOTIFY, AND TALKED TO ALEX.

- Talked to Alex
  - New Spotify idea
    - Keep querying in one file
    - Do visualization in another file
  - Get started on data collection as soon as possible

Monday, April 13

ON THIS DAY, WE BRAINSTORMED HOW WE CAN MAKE A VISUALIZATION OUT OF SPOTIFY.

- Project Objectives
  - To (easily, visually) facilitate the exploration and discovery of new music by looking at artists related to artists we know we already like
- Visualization
  - General

Jason Shen | Angela Jiang | Kit Wu

        - Size of nodes/opacity shows strength of relation
            - Therefore, other encodings should show other things
    - Must-have features
        - Graph linking artists and related artists
        - Music must play for each artist clicked on
    - Probably-should-have features
        - Filters: genre (provided), popularity (provided), time period, location, age of artist....
        - Search bar (artist name, autocompletion)
    - Optional features
        - Show *n* degrees of separation between two artists
        - Second panel coordinated view
            - For instance, pie chart of first-degree genres, second-degree genres, popularity, etc.
- Schedule
    - Jason: not free until after Wednesday evening (eternally free after!!!!!!! for this week :( )
    - Kit: not free until Wednesday afternoon

Wednesday, April 15

ON THIS DAY, WE FURTHER REFINED OUR PLANNED VISUALIZATION OF SPOTIFY'S RELATED ARTIST NETWORK. WE ALSO SETTLED ON A FORCE-LAYOUT BASED NODE VISUALIZATION.

- Set a limit to how many loops the artist-getter runs
    - Corresponds to the number of first degree and second degree nodes
- https://github.com/mbostock/d3/wiki/Requests
- Cute idea: ask for starting artist; floating unconnected nodes in background
- NEED TO THINK ABOUT
    - We don't want to make another node for an artist we already have a node for
    - Two artists have same related artists; only want one node
    - Solved by checking for existing artist and connecting to that node
    - As we're pulling artists, we need to keep a master array of total artists
        - As in, every get request -> push to this master array
- # COMMENT EVERY LITTLE THING
- What we need to do
    - Logistically traverse through related artists
        - Right now, we can just start with Maroon 5
            - Later on, user can input starting point
        - Right now, starting with 20 related each round
            - Caveat: some will have less because already exists (that's fine)
    - Match node to artist
    - Display nodes
        - Minimum: show up, one color
    - Display all links

        - Will look super ugly
- Further down the road:
    - Implement “focus view”: change size of nodes when clicked, play music, colors, whatever
    - Allow user to input starting artist cutesy-ly
        - Allow user to reenter as desired
    - Have option for “reset” view: zoomed out view
    - Side panel of random views whatever coordination

Thursday, April 16

ON THIS DAY, WE KINDA FIGURED OUT HOW TO USE THE SPOTIFY API. WE ALSO DECIDED TO TAKE PHOTOS EVERY TIME WE MEET UP.

- When we make a call to “https://api.spotify.com/v1/artists/04gDigrS5kc9YWfZHwBETP/related-artists”, we get back an object
    - Obj only has one property, “artists”
    - Artists is an array of 20 objects, each of which holds the data for a related artist
    - artists[i] = ith related artist
        - artists[i].name = name of related artist
        - artists[i].id = id of related artist
        - artists[i].popularity
        - artists[i].genre = returns an array of strings, each of which is the name of a genre
        - artists[i].followers.total = total followers
- graph = {nodes: [], links: []}
    - Each node is an entire artist object
        - Unique by ID
        - Check ID before pushing onto nodes
    - Each link’s source and target are artist IDs
- STUPID implementation
    - Initialize degree counter to 0
    - For the FIRST STARTING ARTIST ONLY, we need to get request the artist object from starting_id (not related artist; just artist).
        - Push into nodes
    - Put starting_id into get request (related artist). Get back ^^^^ as specified above.
        - For each (on obj.artists)
            - Check if already in nodes
            - If not, push into nodes
                - Also put ID in frontier
            - Construct a link {source: PARENT, target: THECURRENTFOREACH.id}
        - Increment counter by 1

A CONSOLIDATED PROCESS BOOK FOR CS171

On the night of the 17th.

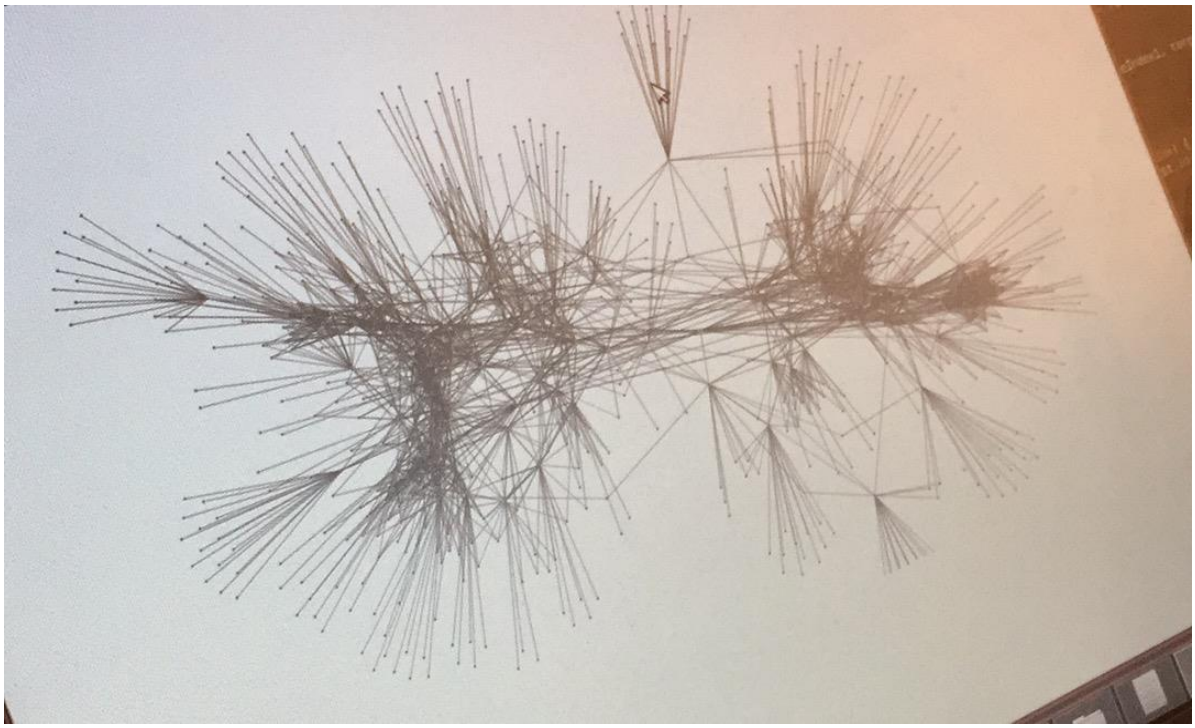
Friday, April 17

ON THIS DAY, WE (IN THIS CASE, JASON AND KIT) IMPLEMENTED NODES AND LINKS. WE WERE PRETTY DARN EXCITED.

- Sitting down to implement graph of nodes and links.
- First step: to understand Angela's code.
    - Console logged a lot of stuff
    - Eventually realized that frontier kinda had something to do with nodes/related artists in general
    - We still don't know a lot of stuff but we do know enough to create a really janky/rachet working version of what we want
- Referencing pset 2/section 3…
    - Created SVG object
    - We did a lot of copy/pasting
    - We played around with where to define the node svg objects, either:
        - 1) At the end of the get_related function
        - 2) After the get_related call in the start function
        - Choice 1 was the one that worked. Still don't know exactly why. Will ask Angela.
    - Initially, nodes were being drawn upon each iteration of the loop within the get_related function. We just created an if statement to only draw when degree == 3, i.e., when all the data has been loaded into the arrays. (We think that there's ***PROBABLY*** a better way of doing this, but hey we ain't complain' on a Friday night.)
    - Just for now, reduced the size of the nodes so we get a better picture.

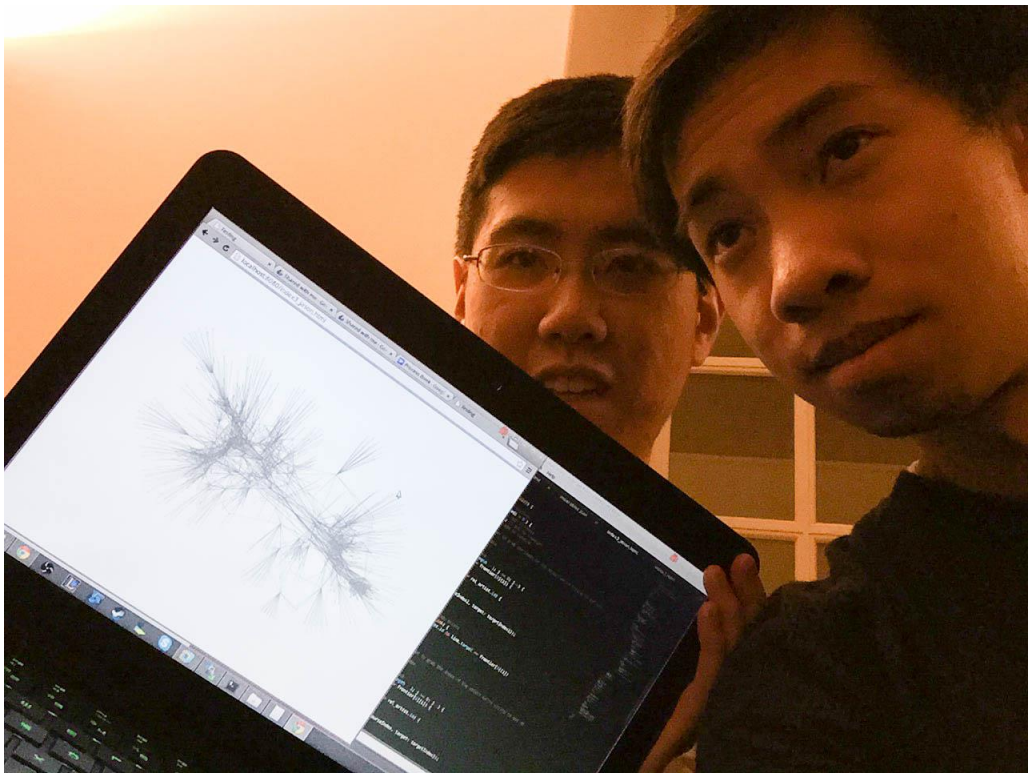
Jason Shen | Angela Jiang | Kit Wu

- More copy/pasting to define the link svg objects.
- Ran into the issue of links not appearing (part 1) because the source and targets were being defined as the "ID's" of the nodes and not the objects that they represent.
- To fix this, we implemented a for loop that looked up the index in graph.nodes which contains IDs that matched those in the sources and targets. Then we replaced the literal IDs in "source: " and "target: " with those indices/indexes (?) which allows us to define the sources and targets as objects (graph.nodes[x])
- However, links were still not appearing, even though in the element inspector, the x and y coordinates were appearing for some (but not all) links.
- Before tackling the issue of why it wasn't appearing for all links, we first wanted to figure out why the lines weren't being drawn at all.
    - Turns out this was a simple css fix.



- Returning to the some-but-not-all problem, we just had to apply the same for-loop fix to earlier code.
- "We've officially lifted five pounds out of, like, a ton, off of Angela's shoulders" –Jason

A CONSOLIDATED PROCESS BOOK FOR CS171



"Lifting five pounds off of Angela's shoulders" - Jason

Sunday, April 19

ON THIS DAY, WE PLANNED OUT THE NEXT STEPS OF OUR IMPLEMENTATION: MUSIC, DATA STORAGE, LIMITS TO ARTISTS.

- Plan out meeting times for this coming week:
  - Monday 7pm-?
  - Tuesday 2:30-?
  - Tuesday 7-?
- To-do list for coming week (in the order that should be addressed)
  - Figure out question of links
    - Do we want both source:Maroon5 target:OneRepublic and source:OneRepublic and target:Maroon5?
      - Yes, for ease of the "focus" view
        - This way can just pick artist as source and show targets as related artists
      - Also worst case can just write fxn to find double links and only show one
  - Figure out if we want to store data or not
    - Because things run super slowly when limit > 3
    - Limit == 3 gives us 697 artists total; limit == 4 gives us 2403 artists total

Jason Shen | Angela Jiang | Kit Wu

        - How many do we want to show anyway? In theory this way (i.e., showing only limit max = 3, so have a maximum of roughly (20*20*20 = 8000 artists) people can only explore artists three away from the current artist
            - We can reload it every time they click on a new one but that would be real real slow
    - Implement focus view
        - First do this for Maroon 5
            - Size differentiation between first/second/third degree relations
        - Then do this such that clicking would allow transition from Maroon 5 to other related artist and making other artist the one in focus
    - Implement music woowowowow

Monday, April 20

ON THIS DAY, WE DECIDED TO RELOAD ON CLICK AND TALK ABOUT THE NITTY-GRITTY CSS STUFF THAT HAD TO BE DONE. ANGELA IMPLEMENTS CLICK AND HOVER, AND LAYS OUT THE BASIC STRUCTURE.

- Focus view: decided reload on click
    - Primary artist needs to be at center of screen, everything radiating outward
        - Deal with overlapping artists?
            - Answer: increase charge; decrease number of related artists
    - Need graph.nodes to store degree of each artist in relation to primary artist
        - Update as primary artist changes
        - Link opacity? Based on degree of graph.nodes
        - Class that reflects degree for CSS purposes
    - Click function
- Angela implementation:
    - Click and hover implemented
    - Colors changed
    - Structure is
        - <g class = “artist [name]”>
            - <circle id = “[name]” class = “node [degree]”>

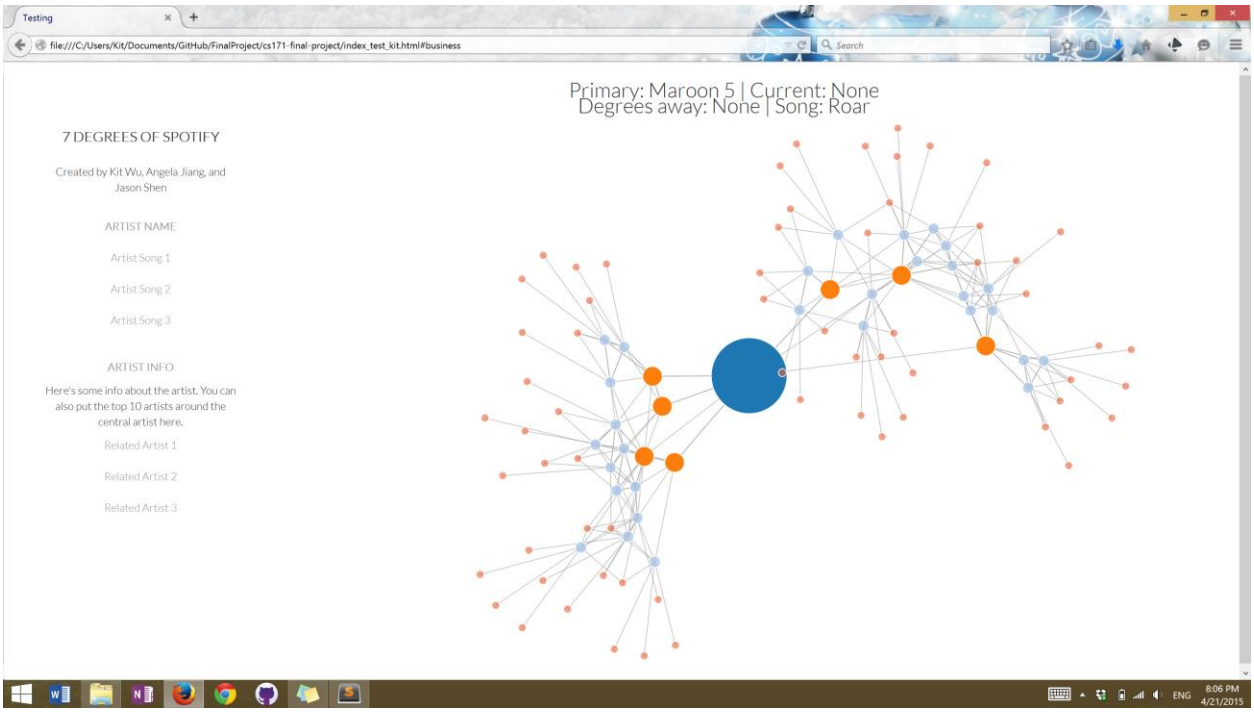
Tuesday, April 21

ON THIS DAY, WE HAD GRAPHS AND LINKS BUT WERE NOT SURE WHAT TO DO NEXT. SO WE CONSULT WITH ALEX AND DECIDE TO IMPLEMENT SIDEBARS, AND *MAYBE* TRANSITIONS.

- At a standstill in terms of focus visualization
    - How to do transition???? Ask TFs
- In the meantime, can implement:
    - Interactive initial page search bar
    - On page search bar

A CONSOLIDATED PROCESS BOOK FOR CS171

- Get an artist's top tracks
  - https://api.spotify.com/v1/artists/{id}/top-tracks
  - Returns *Object {tracks: Array[10]}*
    - Top 10 tracks (i = [0,9])
      - Math.floor((Math.random() * 10))
    - Random number generator
    - obj.tracks[i].preview_url
    - Look for error; try to play another track



Messing around with the sidebar.

Late Tuesday/Wednesday, 4/22

LATER ON THE SAME DAY, KIT IMPLEMENTS A SIDEBAR FRAME WITH PLACEHOLDER TEXT.

1. Sidebar framework implemented, with placeholder text for now but we'll probably have changing, interactive text in the future
2. Some discussion on what the sidebar could potentially have
3. A discussion with Alex in Quincy about how to move forward with the transition (it IS possible)
4. A function that displays names of nodes as text is implemented, although use is optional

Monday, April 27

ON THIS DAY, WE IMPLEMENT SOME BUTTONS, A SEARCH BAR, AND NOW SONGS SHOW UP ON THE LEFT SIDEBAR.

Jason Shen | Angela Jiang | Kit Wu

A CONSOLIDATED PROCESS BOOK FOR CS171

- Changed file names
    - index.html = most recent
    - index_nosidebars.html as name suggests
- Changed way of denoting node classes by number name instead of in numerals
    - Thus now change colors by CSS
- Now introducing stop button!

- Everything looks pretty ugly, but just trying to implement functions
- Left sidebar displays all 10 top tracks
    - The currently playing is highlighted
    - Stop button stops
    - If click on same thing twice, just rotates songs
- Next steps:
    - Jason figuring out what's wrong with the search bar??????
    - Make the songs on left clickable
    - Related artists

Sunday, May 3

PROGRESS HAS BEEN MADE; SEARCH FUNCTION IS ALMOST WORKING, AND WE'RE DEBUGGING. WAIT JUST KIDDING! SEARCH IS FUNCTIONAL AS OF 9 PM! AND A WHOLE LOAD OF STUFF IS NOW FUNCTIONAL AS WELL.

I *THINK* THIS WAS THE DAY WHERE WE IMPLEMENTED NODE CONTROLS?

There's so many changes you might as well read the notes for today.

(Jason writes in blue, Kit comments and writes in maroon)

- Search function is almost there
    - Right now, I am able to type in the name of any artist, hit go, and the page refreshes the visual accordingly.
    - The issue is, right now the search returns an array of artists with the closest match at index 0, and I hard coded it to refresh based on array[0].
    - What does this mean? It means that you will get a correct refresh every time only if you type the full artist name exactly. Ex: "bruno mars" and not "mar"

Jason Shen | Angela Jiang | Kit Wu

- I think this is fine however, because we will try to implement the auto suggest feature. Once this is done, the user will just select from the suggested names, which are full and exact.
- The current problem we need to address is how to get every artist on spotify into the pool of artists from which the auto suggest feature draws from.
- Potential implementation:
  - Right now, jquery search function that predefines the pool of artists from which to search from is run at the beginning, along with start.
  - We should define the search function to query the API for search results and create the pool of artists based on those results.
  - This function should run whenever the textbox is changed. So maybe add an onchange thing.
- UPDATE ABOUT 2 HOURS LATER: YES IT WORKS. NOICE.
- After countless hours, discovered that the dropdown for the search bar was not appearing because the stylesheet wasn't linked LOL. I thought I had linked it, but that was a different stylesheet…
- UPDATE AT 9:50 PM. SEARCH IS COMPLETE. There was a problem with the z-index of the right-sidebar-wrapper where it was really high and causing it to layer itself above everything else, among which was the dropdown menu, which was why the dropdown was appearing behind everything else.

Kit's comments:

- Wow we just made so many changes that I don't really know where to start.
- Implemented pretty much the entire user control panel on the right sidebar; now people can manipulate/control nodes at will, or at least within the limits that we gave them with the numeric buttons.
  - Decided not to do a slider; this is a little more neat. It also prevents the visualization from reloading every single time the slider is used.
- Jason implemented jQuery dropdown and I did a little fixing of the jQuery CSS to make the dropdown look nice (also changed the z-index to make the dropdown options appear on top of the sidebar, with the sidebar previously having a higher z-index than the dropdown menu)
  - Note in case I forget: the z-index is the number/value that determines what div or thing comes on top/in front.
- Generally made things look prettier; sidebar divs now extend all the way from top to bottom, svg is now confined to the screen (no scrolling needed), and various placeholder content inside the sidebars have now been removed.

GO TEAM

- Endnote: the differences between what we have right now and what we had a day ago is dramatic. The user has tons of node control and we might implement another control for friction/node repulsion as well. I'm not sure whether we'll be able to make another dramatic leap like this again before the deadline, but gone are the days when we argued about how many nodes a central/secondary node should have connecting to it; the user can decide for themselves, and listen to music at the same time.

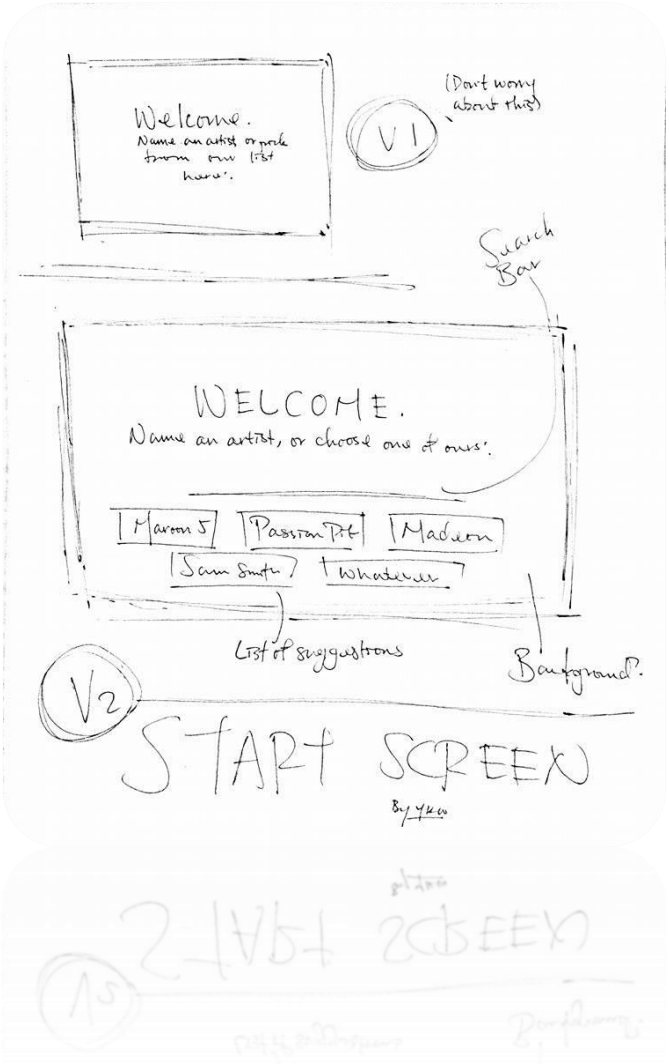
- Kit's personal rating: it's alright. Vis is fun to play around with, but doesn't have that extra "oomph" to make it on par with the other Spotify visualizations out there (talking about the developer showcase)

Monday, May 4

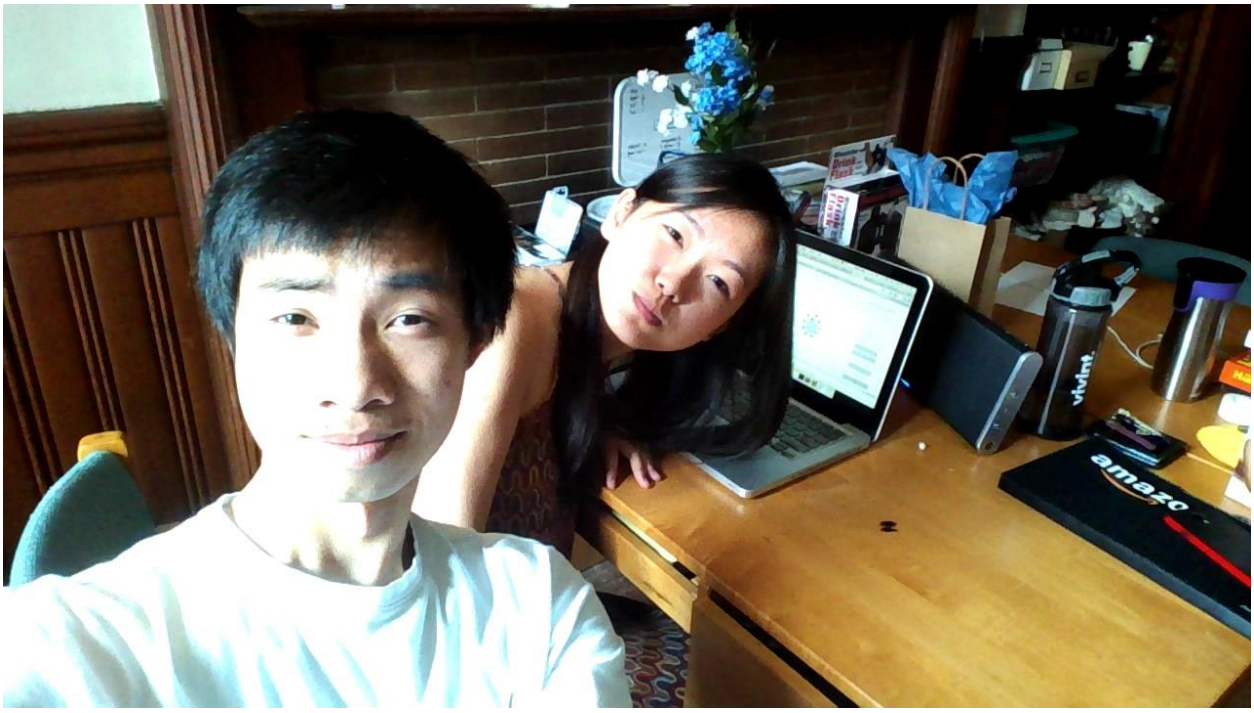
ON THIS DAY, WE DEBUGGED. AND ADDED A WHOLE BUNCH OF FEATURES. AND CHANGED CSS TO MAKE EVERYTHING LOOK GOOD. KIT ALSO WORKED WITH ANGELA TO REFINE NODE CONTROLS AND CHANGE SETTINGS ABOUT THE STANDARD FORCE LAYOUT.

- Function
  - **Jason:** If nothing typed in search, don't let it go; unsearched
    - Or else will throw error
      - Low priority: limit search return results
  - First screen
  - Additional visualization of google trends

- Prettiness/guidance
  - Layout margins and stuff
  - Click node buttons highlight
  - Size of search bar
  - Font
  - Draw lines on top of nodes
  - Color of nodes?
  - Positioning of text/nodes/clutter
- Organization
  - Separate vis separate files?
  - Separate css and etc
  - Prepare process book
  - Documentation and etc.
  - Video



A CONSOLIDATED PROCESS BOOK FOR CS171



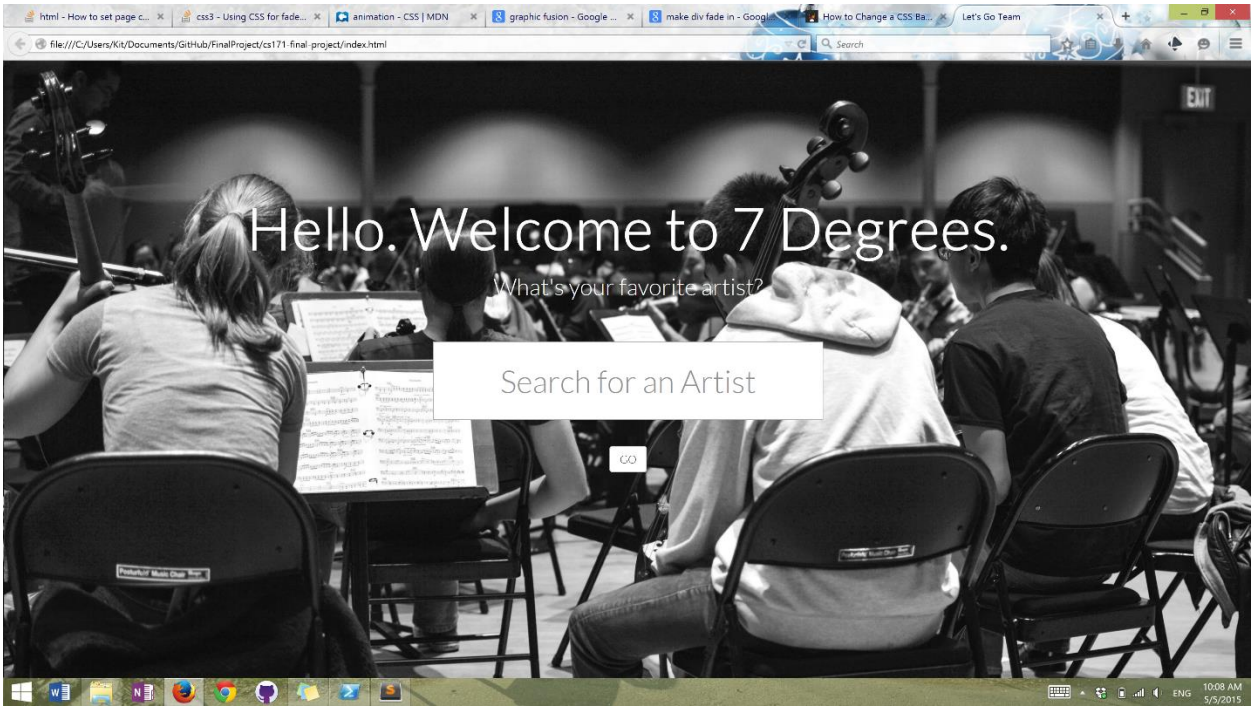
Listening to music, while implementing (more) music.

Tuesday May 5th

ON THIS DAY, JASON DECLARES HIMSELF A GOD AT WEB DEVELOPMENT AT ~3 AM IN THE MORNING, AND WE BASICALLY MAKE MORE PROGRESS THAN WE HAVE DONE IN ALL THE PREVIOUS WEEKS (SORTA) IN A DAY AS WE SPRINT TOWARDS THE FINISH LINE.

- ### Jason declares himself a god at web development
- Faced with the issue of figuring out how to go from start screen to index while keeping track of search input and loading index with the proper visual
    - At first, attempted to call a function when the GO button is clicked in start.html, where, it stores the search in a variable, loads index, then refreshes visual based on the variable, similar to how the search on index is implemented.
    - Quickly ran into the issue where upon redirecting and loading index, it kills all currently running scripts...so that was a no go.
    - After 2 hours…
    - Discovered the concept of hashes.
    - User searches, upon hitting go, it redirects to “index.html#...” where … is what the user searched. This allows you to keep track of what was searched using the browser.
    - Then you just perform the refresh using the hash as if it were a variable.
        - “window.location.href.split('#')[1]” returns the artist that was searched.
        - Example if the url is “index.html#Train”, then “window.location.href.split('#')[1]” returns “Train”.
    - Last problem was that the hash would remain in the url which could prove misleading

        - Example: If I searched Train, and then later on index redrew the visual to center around Maroon 5, the url would still say index.html#Train.
    - Solved this with:
        - history.replaceState({}, document.title, "/index.html");
        - This removes the hash without refreshing the page!!
- Problem with abstracting main.js
    - main.js, in addition to search.js, needed to be abstracted to get start.html working since main.js contains global variables that search.js uses.
    - I thought that since this big chunk of code was now abstracted, you could just delete this chunk from index.html and simply link main.js. This was not the case.
        - You end up with a blank visualization on index.
    - This is a problem because it results in ugly code. If someone in the future were to change this chunk of code in index.html, they might also have to change this same chunk in main.js.
- Renamed start to index and index to main



"I'm so ridiculously proud of this" – Kit

Tuesday, May 5

AFTERNOON CHECKPOINT OF THE SAME DAY

- To-do
    - FUNCTIONAL
        - Making sure users are not allowed to do crazy large combinations

Jason Shen | Angela Jiang | Kit Wu

- Add scale
- When paused, highlight that button

- PRETTY
  - Highlight currently playing song same as right sidebar (green!)
  - Add "Top Tracks"???? left sidebar top
  - Make bars look prettier/what to do with bars? (okay)
  - Colors of nodes (color scheme?)
  - Font maybe? who knows
  - Text overlapping with nodes meh
- ORGANIZATIONAL
  - delete unncessary code
  - comment out things
  - delete all console.logs
  - make sure everything is linked (css sheets, js sheets)
  - README - The README file must give an overview of what you are handing in: which parts are your code, which parts are libraries, and so on. The README must contain URLs to your project websites and screencast videos. The README must also explain any non-obvious features of your interface.
  - We will strictly enforce the two minute time limit for the video, so please make sure you are not running longer.
- DELIVERABLES
  - Final Project Submission Form
  - Once you have finished your final projects, please make sure to fill out the final project submission form (once per team) until midnight today: https://docs.google.com/forms/d/1ArRbFiwhZOaJf-0kFNb2uxhOlKqCO_4u3Z7Y6lXW_eo/viewform?usp=send_form
  - (And also make sure to fill out the peer assessment form: https://docs.google.com/forms/d/1fh-j8DD2000WUrYvParqEPNtp5fbryibDzMTu-x2kz4/viewform?usp=send_form)

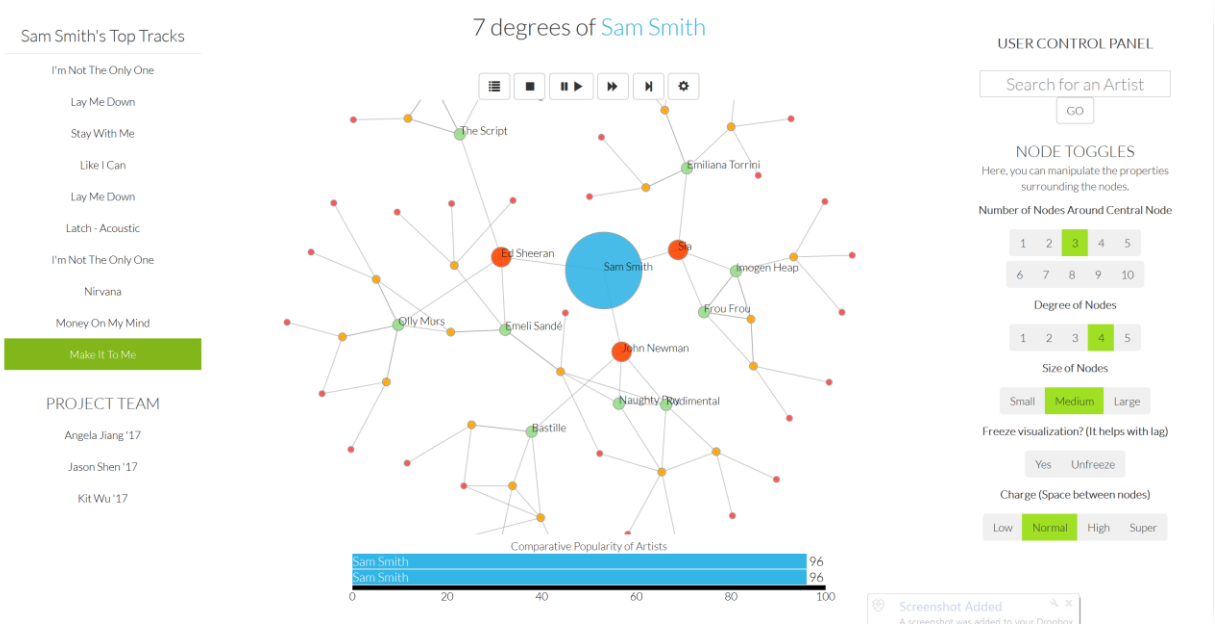
END OF JOURNAL

A CONSOLIDATED PROCESS BOOK FOR CS171

FUNCTIONALITY

A QUICK GUIDE TO HOW OUR FINAL VISUALIZATION WORKS

Our visualization is based on a force layout, with nodes representing a central artist, related artists to that artist, and related artists to *those* artists. So nodes of several degrees, in essence. The final webpage comes out like this;



On the left, you'll see a sidebar with top tracks of the selected artists. On the right, you'll see a "user control panel" for which you can adjust the number of nodes, degree of nodes appearing from the primary node, as well as other options. Some buttons on the top allow you to start and stop songs (as well as toggle the sidebars in and out), and two bars on the bottom allow you to compare the popularity of any node on the screen with the primary node, which is represented by the top bar.

The main focus of the visualization, of course, is the network of nodes in the middle. Clicking on any one of them will play a song from the artist's top tracks; double clicking on any one of them will allow you to "jump" to that artist, making the selected artist the new primary artist. The sidebar settings stay when you jump from artist to artist.

All in all, this is simply a much more intuitive way of displaying relationships between related artists; this also presents a much more interactive way of browsing from song to song. It accomplishes our original objective of giving people a much more visual way to see the artists that they usually know, and on top of that, it beats our unofficial milestone of being more interactive than some of the other Spotify artist visualizations out there on the Spotify developer's showcase.