The Investment Spiral

A Visualization of the top 50 companies in the 2013 CrunchBase database

By Erica Gunn

**Overview**

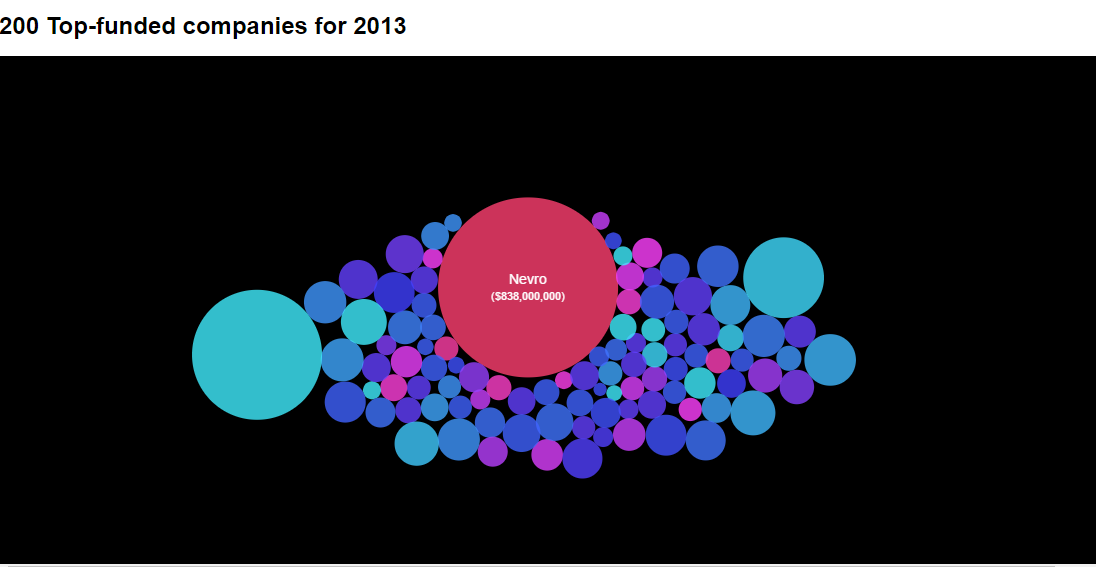
Our assignment for this project was to create an interactive visualization of the top companies and investors in the 2013 CrunchBase database. I wanted to show the connections between companies and investors so that a user could explore relationships and identify funding patterns within the data.

The final visualization is available online here: <http://egunn.github.io/ProcessingFinalProject/>

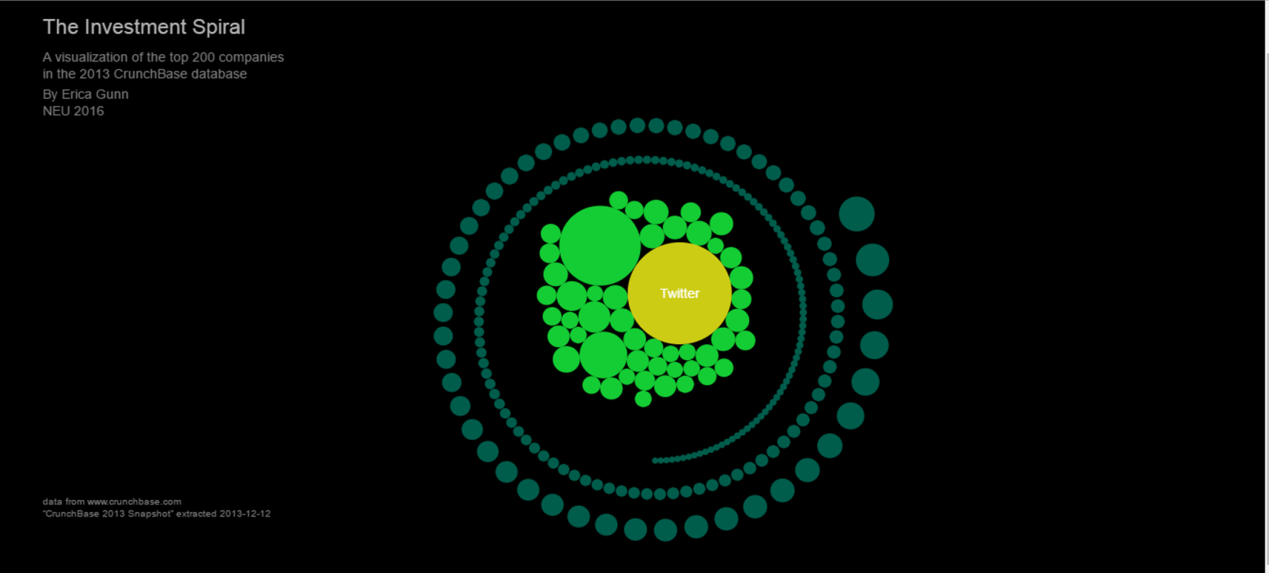
**Process**

We began by loading and filtering the company data into an array, and coded a basic force layout with an attractor that created a custom gravity field that attracted the particles to the center of the screen. To avoid having all of the particles end up in the same place, we also created a pairwise collision function to check whether particles overlapped and to move them apart.

By the midterm, we had created a packed bubbles force layout for the companies in the dataset. When the user highlighted a company with the mouse, we increased its radius and drew a label indicating the total amount invested in that company, as well as the company name. The color of the bubbles represented the categories given in the database for each country, and their size reflected the overall size of the investments.



From there, we began adding information about the investors as well. I started out by drawing the investors in a circle around the company bubbles, but there were so many investors that they overlapped and formed a solid ring, making it impossible to distinguish between them. Instead, I decided to place the investors on a spiral and scaled the size to match the value of their total investment.



A second view could be accessed by clicking on a company, which showed the selected company at the center of the screen and the investors that had invested in that company around the outside of the ring. The back button returned the user to the previous screen.



For my personalized visualization, I wanted to focus on the connections between companies and investors. Rather than splitting the view into two separate screens, I wanted to show the connections in a single visualization when the user hovered the mouse over an investor or a company. This way, the user can look for patterns in how investors invest, the number of investors per company, or other potentially interesting features of the dataset, without having to move back and forth between two separate views by clicking. I hoped that seamless integration of the highlight behavior into a single visualization would make it easier to explore the data.

Actually activating this behavior was something of a challenge. Just after the midterm, I completely rewrote my code to be based on Javascript prototypes rather than separate class functions. Companies and investors each had their own individual constructor functions with properties unique to their own specific behaviors, as well as a shared prototype Particle function that contained features common to both kinds of particle. This wasn’t required for the class, but I was curious to know more about how prototypes worked, and it seemed like a good opportunity to experiment. In addition to implementing the prototype functions, I had to go back through and update the entire project to accommodate the new structure, which included reshaping my object definitions and data structure to optimize searching for the variables I was most interested in.

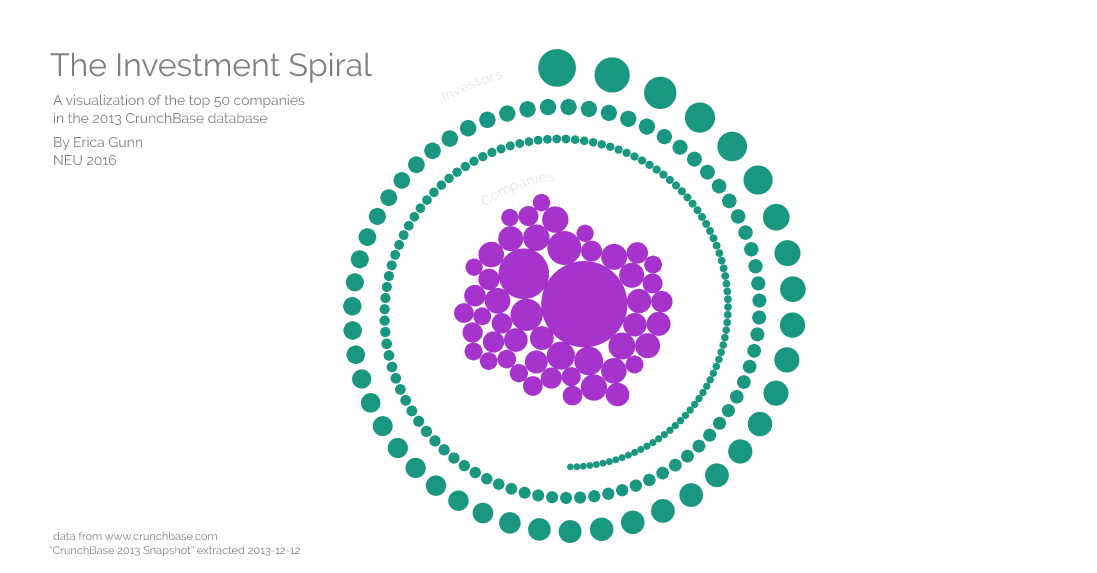
**Data management**

The question of how best to index the data was an ongoing concern, and something that changed several times throughout the semester. Because this project was a gradual evolution rather than a pre-planned structure, we came up with needs that we hadn’t anticipated and needed to adjust the data pre-processing to match. I re-configured my basic data structure 3 or 4 times during the course of the semester in order to optimize sorting and storage of the information, and minimize processing power required for each step.

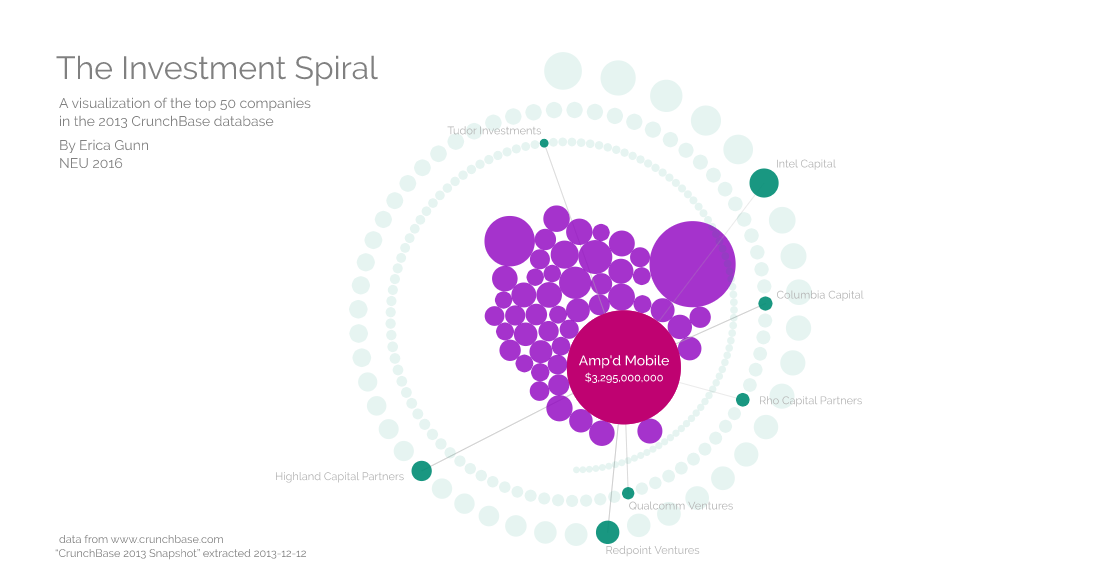
The data was imported from two separate .csv files, one that contained information about the companies and the amounts invested, and another that contained information about the investors and how much they had invested. We then wrote filtering and sorting algorithms to find the top 200 companies and all of the investors connected to them, and to aggregate total investment amounts for each group. This information was stored in a company array and an investor array, which were then used to generate the “particles” for display. To draw connections between the company and investor particles, I created a connections array that contained both a particle and an investor particle, as well as information about the investment amount for that particular connection. The basic visualization uses two separate arrays of company and investor objects to draw the basic bubble force diagram and spiral. The mouse highlight function runs when a user places the cursor over a company or investor, and matches the selected investor and company name to the appropriate entry in the connections array, and then updates the position and other relevant values of the company and investor particles stored in the connections array to draw the particles and the connections between them.

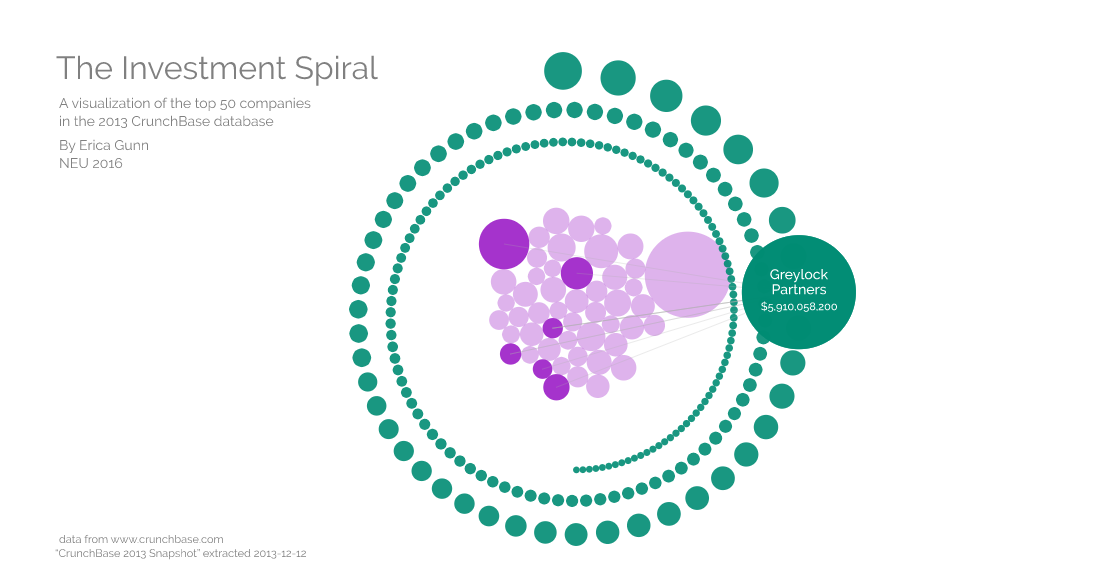
**Final version**

After updating the visual styling and tweaking other features, the current version looks like this:



When a user highlights a company, the software updates the display of the investors related to that company, and adds labels to identify them.



Similarly, highlighting an investor shows the companies funded by that investor.

I chose to change the company color on highlight because it’s hard to keep track of one particular company in the moving cluster. Because the investors have fixed positions and are organized by size, it’s easy to tell the difference between a highlighted and a non-highlighted investor, and I didn’t feel that it was necessary to change the color on highlight there.  I also decided not to resize the target company and investor particles when drawing connections, because the size is important when comparing them to others in the dataset. It would be easier to add labels to the companies if I changed their size, though, so I may need to do that in a future version.

**Future directions**

I am still tweaking the small details of the visualization to make it more functional and user-friendly. I added a function to detect overlaps in the investor labels when a company is highlighted. If two labels overlap, it will move one up and one down until it is possible to read the text. In some cases, this causes the labels to move further from their investor particles than I would like, so I need to see if I can improve that. I’d also like to include optimization in the x-direction, so that the program will move a label left or right to avoid collisions, in addition to up and down. It may also be necessary to explicitly code a minimization step to “stick” the label to its parent particle; in a couple of instances where the investors are grouped very close together, the labels have a tendency to wander off during the optimization, making it hard to tell which particle they relate to.

It might also be helpful to include labels for the company particles when an investor is highlighted, and to indicate relative investment amounts for each connection shown. Adding that much extra text without crowding the visualization is something of a challenge. I considering adding a click functionality here, so that clicking on a company or investor adds extra details in the labeling, and clicking off of the company (or mousing over a different one) returns to the original behavior. That’s less disruptive than changing screens, but also allows the viewer another level of control over the amount of detail that they want to see.

I won’t have time to implement all of these details in time for our final presentations, but those would be my primary areas of focus for a future version.

**Reflections**

I think that the final visualization works pretty well as a way to explore the CrunchBase data. I’m not sure how useful it would be for identifying patterns and trends; the fluidity of the packed bubbles arrangement makes it hard to keep track of specific companies while exploring. I think there might be other visualization techniques that would be better for rigorous comparisons, but this is fun to play with and is interesting to look at. Adding a visualization of the investment amounts for each connection would add another layer of information and complexity to the data, and might enhance its usefulness.

It was helpful to see how much the data structure matters to the final code, and to be reminded of the need to consider the final use case when doing the initial data prep. I didn’t have a good picture of what tools we’d be using or where we’d be going with the visualization when we started (or really, almost until the end of the semester), and so I spent a lot of time going back and reworking to make things smarter after the fact. Having a clear plan for the things I needed to access in the data from the beginning would have made life a lot easier later on.

I’m not sure that including prototypes increased the efficiency of my code in this case, because the company and investor particles had such different functionality in the end. Still, this project gave me a better understanding of when it is appropriate to use prototypes, and when it may not be necessary. (You told me that I didn’t really need them here at the beginning of this little adventure, but we agreed that it would be a good exercise to try it anyway, and I think it was worth the additional effort to do so.)

Overall, it was useful to see how to build a visualization from the bottom up, without relying on prepackaged libraries to do so. I have a much better understanding of the force layouts that we learned in d3 last semester, after having built one myself in this class. Seeing the complexity of code required to do even simple things also increased my appreciation of libraries like d3 that cover up a lot of the bookkeeping so that the programmer can focus on the higher-level design.