Option A, idea 3 - creative animation and transitions

When starting this project, I thought it would interesting to look at the different Android smartphone devices. I pulled the data from this Wikipedia article. I was able to export the table from Wikipedia, and performed several cleanup tasks in order for the data to be properly read in d3.js. I really liked the parallel coordinate visualization that showed information about cars. By playing around with the brush tool, you could uncover interesting patterns and trends that you wouldn't be able to see otherwise. I thought I could do something similar with the Android data I had. The authors of the paper cited a finding in a Tversky paper that stated that animations didn't seem to help "to illustrate or communicate the workings of complex systems" (Robertson George, p.4) Tversky's observations notwithstanding, my goal for this visualization was to create some sort of complementary animation or transition that would help visualize the data better.

Using the code provided by Jason Davies here, I manipulated the code to work with the Android data I had. The dimensions I thought would be most interesting to look at was the release date, display inches, RAM, and megapixels of the rear camera. I ran into a big obstacle when I tried to turn the release date information into it's own axis. After a couple of days of research, troubleshooting, and trial and error, I finally figured out how to display date data as its own axis. In terms of animation and transitions, the creator of this parallel coordinates provided a few transitions to help highlight the data that was being brushed. I thought I could help make visualization a little more striking and pleasant to look at by changing both the vertical axis dragging motion, creating a stroke-width 'pulse', and animating the initial drawing of the lines.

The way the dragging motion was currently set up, is that the creator didn't specify what type of *ease* he wanted, so d3.js defaulted the ease action to *cubic-in-out*. I thought this looked pretty boring, so I added in code to change the ease to *elastic*. Now, once the drag end action is fired, the vertical axis that was dragged snaps back into place with a little bounce.

The other animation I added was the stroke-width 'pulse'. I read up on how to chain several transitions together from this blog, and it taught me that using the **.each("end", function() { … }** code is the best way to make sure your transitions will fire sequentially without overlap. Using this chaining method, I change the stroke-width and color of the highlighted data lines to a thicker and darker line, and then change it back to the default width and color. The chaining method gives the transition a 'pulse' like animation to really accentuate the data that you've highlighted through brushing.

Lastly, I learned how to animate the drawing of the line data using this blog post. Taking advantage of *stroke-dasharray* and *stroke-dashoffset*, I was able to easily mimic the animation of a line being drawn onto the visualization.

I think the resulting visualization is interesting because it allows you to see trends in Android smartphone specifications. One cool observation is that it wasn't until 2012 when smartphones started breaking into the 5 inch display size. It's also interesting that the newest phones have yet to break the 3gb of RAM mark yet (except for one phone, which is the Samsung Galaxy Note 3).

After reviewing the animations and transitions I created for this visualization, from an objective standpoint, I don't think my changes really helped benefit the usability of this

visualization. I learned quite a bit about HOW to do transitions and animations, but when it comes down to it, I don't think the visualization is any better served with or without my changes.