Gas Genie – Interactive Fuel Economy and Carbon Emissions Data Visualization Tool

Tsung-Yen (John) Yu, Student, *Arizona State University*

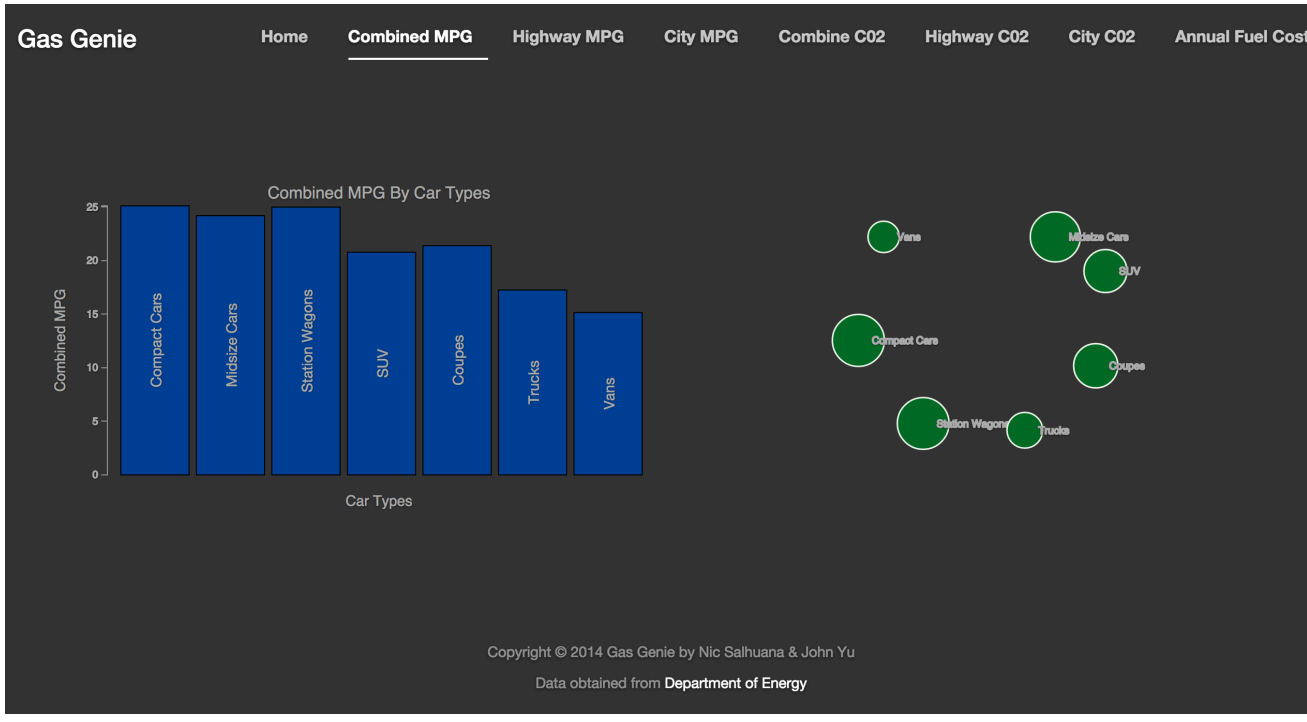

Fig. 1. Screenshot of the visualization

Abstract—Gas Genie is a tool that visualizes a mass amount of fuel economy data that we obtained from the Department of Energy. We created the tool with a goal to assists users to access and understand the data intuitively and provide an opportunity for them to interact with the data. The paper gives details on how the system was design and my personal contribution to the tool.

1 Introduction

With gas prices on the rise and consumers are more environmentally aware, the fuel economy and carbon emission of a car are important factors when consumers choose for their next car [1]. Gas Genie has take up the challenge to help consumers choose cars by presenting the data we obtain from Department of Energy in an interactive visualization. The visualization consists of both a stacked bar and a bubble graph that allows the users to drill down into specific cars and provides more details on demand. Though there are a lot of similar sites that presents the same data, such as car sales companies (i.e. dealers, Cars.com), data aggregation companies (i.e. GasBuddy.com), and research organizations (i.e. Department of Energy), our goal is to combine their advantages and improve their disadvantages. Most of the prior mentioned organization have the same data, however not all of the important data points are given to consumers or they are hard to navigate through.

2 System

2.1 Data

During initial research and idea gathering, we found Edmunds API, which provides mass amount of data points for individual vehicles and easy access to the data using their API. However, further research revealed that most of the interesting data, such as True Cost To Own, were not available for regular user. After meeting with Dr. Maciejewski, he suggested us to use Department of Energy's fuel economy data.

We end up settling with Department of Energy's data, however, we encountered two problems. One being the

file size is too large, because it contains many detailed information about fuel economy that most consumers does not need and have many similar data due to the similar models of car. The original data contains almost 2000 cars and with more than 150 data points for each cars. Since we were more interested in fuel economy and environmental impact, we decided to only keep some of the well-known car information that would affect fuel economy (i.e. engine size, number of cylinder, transmission, and etc.). We also only keep the better-known statistic measurement such as highway, city, and combined fuel economy; highway, city, and combined carbon dioxide emissions; annual fuel cost.

The second problem we had to overcome was the format of the data. The original data was in CSV format. The CSV format would work fine if we were to display the data without any interaction. However, we wanted to show some kind of parent/child relationship to allow easy navigation between cars. Having prior experience with JSON formatted data, we know JSON will be a great choice to represent the data and will be easily usable in D3.js. Converting CSV to JSON is a simple task that we accomplished by running an online CSV to JSON converter [2]. The converter changes each row in the CSV to a JSON Object, so the final output of the file is a JSON Array of JSON Objects with no relationship. In order to generate the parent/child relationship, we had to write an Objective-C program that automates the process of formatting the JSON file to have such relationship. The detail of the program is given in the My Contribution Section.

2.2 System Design

There are two main components to our system, the front-end and the back-end. Here we define the front-end and back-end differently than the common usage. According to a formal definition of front-end and back-end system, our system does not necessary have a backend, other than the server that is hosting the website. Our data is processed and displayed with user's browsers using JavaScript, more specifically the D3.js library, or Data-Driven Documents (D3), which is consider as front-end in the formal definition. We define the visual aspect of the system as front-end, such as how graphs are positioned and the overall look of the site. Even though D3 does help with displaying the graph, here we consider it as the back-end.

For the front-end, we used Twitter Bootstrap. Twitter Bootstrap allows us to upgrade our website from a dull, plain, and boring look to a professional, aesthetically pleasing, and interesting look with minimal coding required. The reason we took the extra step to make our site more visually appealing was because it provides a better user experience and provides a higher retention rate on our site. Also, one of the disadvantages that other sites had was either too cluttered with unimportant information or the site is very dull and does not engage users. As mentioned by Jef Raskin [3], "...superior interfaces are exceptional long-term investments that produce customer satisfaction, increase the perceived value of a product, minimize the cost of customer support, achieve a competitive advantage, and establish brand loyalty..."

As mentioned in class, D3 is a very powerful JavaScript library that brings boring data to life by visualizing the data or giving user a way to interact with the data. Therefore we chose D3 as our back-end to help us process the data and populate the graphs, however prior to choosing D3, our team did discuss the possibility of using Adobe Flash. After reading some articles, we realized that there were some drawbacks with using Adobe Flash. One drawback is presented in a paper written by Michael Bostock et al [4]. The paper shows that D3 can render graphs much faster and efficiently than Adobe Flash. Another reason we ended up choosing D3 was because not all modern browsers or devices supports Adobe Flash natively, thus by using Adobe Flash would mean that our user would have to have Flash installed on their device. Also, since JavaScript engine is built into the browser, therefore if there was an update to the JavaScript, JavaScript will be updated automatically when the browser update and JavaScript is widely supported; Adobe Flash on the other hand will require user to have a version that is high enough to be runnable. After some more research and testing with D3, we found that it has a very highly active and supported community online, which made learning this new library less difficult. D3 is also incredibly customizable, simple to bind data to visualization, and scalable with different sizes of datasets. One of the main selling points of D3 and one of our main choices of choosing D3 is that it allows us to easily modify HTML elements within the Document Object Model (DOM) according to our data. This feature is really important because we will be graphing two graphs, and D3 lets us easily bind the same data into two very different graphs and manipulate HTML to display our graphs.

2.3 Graph Design

We encoded the data into two different but interconnected graphs. The first graph is the stacked bar graph, this type of graph is very common and users are very accustomed to the meaning of the graph. The second graph is the bubble graph, this kind of graph is less common but we felt it would an interesting representation of our data and would be able to show the parent/child relationship very intuitively. As we discussed and designed our graphs, we kept Shneiderman's Information Seeking Mantra in mind: "Overview first, zoom and filter, then details-on-demand" [5]. Both of our graphs start by giving an overview of different types of vehicles, the user can then zoom and filter the vehicles by clicking on types and makes, finally on any level, if the user would like more information, they can click on the bar to have a pop-up dialog box show up with details about that specific bar.

Work done by Hardin et al. [6] suggests that bar graphs are best used when comparing information that is numerical and divided into categories. Stacked bar graphs with this type of dataset present itself with a clear trend of the highs and lows. In our stacked bar graph, we encoded the height of the bar to the corresponding value.

As mentioned earlier, each of the bars contains more information within them, the user can see more details on demand. Within the model level, some selection will show as stacked bar graph, this means for that specific model there are multiple trims and those trims affect the statistics, therefore has different value. For those selection that only show a regular bar graph, this could mean either that model only have one trim or the trims does not affect the statistic at all. In order for the stacked bar graph to be truly interactive, we added a functionality to allow going back to the previous level by clicking on the title of the stacked bar graph. In addition, due to the large amount of data points certain selection had, we translated the bar label vertically and added in a hover-to-display feature that shows the value of the bar to make understanding the graph simpler.

Work done by Chuang et al. [7] from Stanford University greatly influenced us to choose bubble graph as our second visualization. From their visualization, we found that a bubble graph would be a great choice because it provides an intuitive way to represent the data, especially with parent/child relationship that we are trying to accomplish. Similar to the stacked bar graph, we encoded the radius of the bubble to the corresponding value. As we know from class and from Wilkerson's work [8], size encoding for objects that are round are best to map the data with respect to the radius of the object instead of the surface area. Our approach was to have each click on the parent bubble to show its child surrounded by its parent and other bubbles that are the same level as that parent in a circle. However, due to the vast amount of data we have and limited screen size, we were only able to show this effect in the top level. For this reason, we added in hover-to-display to the bubble graph as well to allow users to know the exact value that the bubble is representing.

Another big part of our graph design is the color scheme of the graph. We wanted to find some visually safe color to pop out in our gray background. First, we had to decide what information to encode with color. We decided that in order for the stacks to be more visible and to show the descending value relationship in the stacked bar graph, we had to use different but sequential colors for each stack. Then in order to represent the parent/child relationship in the bubble graph, we decided that we had to also use different but sequential colors for bubbles that have a parent/child relationship. For the stacked bar graph, if there was a stack, we used a blue scale from dark blue to light blue to represent large to small value, respectively. As for the bubble graph, we chose the green scale with dark green to light green to represent parent and child, respectively. The last but very important information that we chose to encode was the connection between the two graphs. Since both of our graph will be simultaneously showing the same data of current selection but in different format, we wanted a way for the user to effortlessly find the corresponding data in both graph and act as a pre-attentive cue. Work done by Healey et al. suggests that changing in hue is actually one of the many visual variables that can achieve pre-attentive cue [9]. Therefore we decided to choose to change the color of the bar and bubble to orange when it is hovered as that it will both stand out in the gray background and in the blue and green scale.

3 Result

As mentioned earlier, the data we obtain from the Department of Energy is quite large, however, after adding the parent/child relationship and visualizing the data, you can see the overall picture easily. From the visualization page, as shown in Figure 1, without filtering anything the user can at a glance see that Vans and Trucks have a lower Combined MPG than Compact Cars and Midsize Cars. If the user wants to more about specific type of cars or different category, the user can demand more detail by clicking on the desired selection. One interesting fact we found using our visualization was there were couple models of cars that had a higher City MPG than the Highway MPG. Typically we think most cars should have the opposite, since in the city the traffic is more of stopping and accelerating, which consumes more fuel and on highway is more of a constant speed. Looking more closely, we noted that they were Hybrid Vehicles. Doing more research, we found that most Hybrid Vehicles employ certain technologies that are specifically for the city driving. One of them being regenerative braking, which charges the on-board battery when the brakes are applied, hence prolonging the battery's power [10]. Another technology is the turning off the engine after being idle for a certain amount of time, which will reduce gas usage therefore increase fuel economy.

Another advantage of our site is being able to easily compare different model side by side using our tooltip feature, which is pop-up dialog that display detail infor-

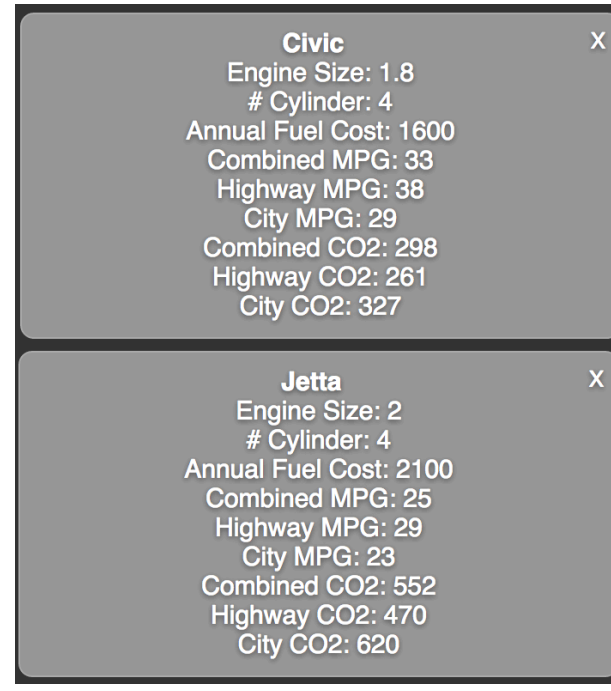

Fig. 2. Two tooltips comparing a Honda Civic and a Volkswagen Jetta.

mation about specific selection on the stacked bar graph. As shown in Figure 2, even though these two cars have different make, but we can still easily compare the two models by opening up the tool tip.

4 My contribution

As mentioned earlier, the original JSON file that we converted from CSV file does not fulfill our goal of providing a parent/child relationship. So in order to achieve this, we had to either format the objects and arrays in the JSON file in a nested parent/child relationship by hand or we could do it programmatically. Due to the large amount of data, we decided to write a program to format the file. Out of the two of us, I had more experience with JSON and have prior experience working with JSON in Objective-C, so I was in charge of formatting the JSON file. Even though we had already sort the cars into different types and makes and models was given, however I still had to figure out the whole structure by graphing out the relationship because we only have the child available, we have to generate the parents from the child. By graphing out the relationship, I was able to see that the structure can be establishing by finding all the cars with same type, then average their miles per gallon, engine size, annual fuel cost, and etc., then do the same for individual makes within each types. Since we were more focus on the Combined MPG, I sorted the child in descending order according to its Combined MPG to make it display correctly in our stacked bar graph.

I was also in charge of visualizing the stacked bar graph and linking up the two graphs. I started from a small subset of our data to visualize the graph then work towards the whole dataset. Since we started with a well-defined structure of JSON file, implementing the stacked bar graph was fairly straightforward. However, I did hit some road bump during the process, such as bars does not show up or does not disappear after transitioning to different level, luckily I was able to resolve the problem following some tutorials online. However, linking up the two graphs was not as easy, the two graph works great individually but I had a hard time connecting the two. The main reason was because my partner and I separated the work of doing the graphs, he is in charge of the bubble graph and I was in charge of the stacked bar graph. In order not to interfere with each other's graph, we implemented the graph separately and took a different approach on binding the data. We didn't realize it would end up taking us more time at the end to connect the two graphs. One advantage of the bubble graph was it was built to have the parent/child relationship; therefore going between each level was built in. Though this is not the case for the stacked bar graph, going down the level was built in, but I had to manipulate the code to support going back to the previous level. The last hard part was to get both the stacked bar graph and bubble graph to go up and down the level together.

I also took up the task of adding the "detail on demand" feature, which was the pop-up dialog, tooltip, which provides detail information about a certain bar in the stacked bar graph with a single click.

7 Conclusion

Gas Genie is an interactive visualization tool that aims to provide users, especially potential car buyers, a clear view of cars fuel economy and carbon emission. We provide two very distinctive graphs to allow users to explore our data differently; we also provide more detailed information with tooltips.

I was able to learn a lot of new skills and brush up on some old skills through doing the project, such as D3 libaray, Twitter Bootstrap, and visualizing data. Visualizing data and giving life to a data was a new skill. Finding ways to best visualize it and present it in a way that most easily conprehendable. Though I had experience with JavaScript before, D3 was another new set of tools that took some learning. However, I believe learning this tool had greatly improved my JavaScript skills and provided me a go to tool for visualizations.

Group Member

Nicolas Salhuana.

Acknowledgment

The authors wish to thank Dr. Maciejewski, School of Computing, Informatics, & Decision System Engineering, and Department of Energy.

References

[1] "High gas prices motivate drivers to change direction. Nearly three-quarters of surveyed motorists would consider an alternative-fuel vehicle for their next car," ConsumerReports.org, May 2012, http://www.consumerreports.org/cro/2012/05/high-gas-prices-motivate-drivers-to-change-direction.

[2] Convert CSV to JSON; www.convertcsv.com/csv-to-json.htm.

[3] J. Raskin, "Looking for A Humane Interface: Will Computers Ever Become Easy to Use?," ACM, vol. 40, no. 2, Feb. 1997, pp. 98-101.

[4] M. Bostock, V. Ogievesky, and J. Heer, "D3: Data-Driven Documents," *IEEE TVCG*, vol. 17, no. 12, Dec. 2011, pp. 2301-2309.

[5] B. Schneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," Proc. IEEE Visual Language, IEEE Press, Sept. 1996, pp. 336-343.

[6] M. Hardin, D. Hom, R. Perez, and L. Williams, Which Chart or Graph is Right For You?, white paper, Tableau Software.

[7] J. Chuang, D. Ramage, C. Manning, and J. Heer, "Interpretation and Trust: Designing Model-Driven Visualizations for Text Analysis," *ACM SIG*, May 2012, pp. 443-452.

[8] L. Wilkinson, The Grammar of Graphics, Springer-Verlag, 2005.

[9] C. G. Healy, and J. T. Enns, "Attention and Visual Memory in Visuali-zation and Computer Graphics," IEEE TVCG, vol. 18, no. 7, July 2012, pp. 1170-1188.

[10] C. Lampton, "How Regenerative Braking Works?," HowStuffWorks, 23 Jan. 2009; http://auto.howstuffworks.com/auto-parts/brakes/brake-types/regenerative-braking.htm.

Tsung-Yen (John) Yu has been the TA for the Mobile Application Development course offered through Ira A Fulton. He has spent the past year helping develop apps and tutorials to help other students learn how to develop their own mobile applications. John is also well versed in web development in HTML, PHP, and JavaScript.