NEW YORK UNIVERSITY SHANGHAI

CSCI-SHU 370-001
Object-Oriented Programming

Visualization of New York Transit Systems Report

Watcher Wang cw1681@nyu.edu
Autumn Wu yw1370@nyu.edu

May 13 2015

1 Overview

Our Project is an visualization of the New York transit system built in Java. The data we use is downloaded directly from Metropolitan Transportation Authority. This raw data contained nine different txt files. They include very significant information of the New York transportation system, or more specifically, the subway system. They are the GPS locations of each Metro stop, stop times of different lines at each stop, the shape of each Metro line, and the schedule of individual line on different days of a week.

Our goal is to build a user-friendly visualization that exhibits significant features of the plain data mentioned above so as to help the public better understand the characteristics of the New York transportation system as well as the relationship between the geographical structure of the city and the functionality of the system.

For data processing, we carefully followed the instructions mentioned in the instruction PDF file professor Spathis offered us. It contained useful ways to preprocess and reorganize data. Detailed information can be referred to in section 2 and 3. However, for the background map part, Geotools did not satisfied our needs for a neat, organized, smoothly dynamic background map. We'll explain the detailed reasons later in the section 3.2. We then tried several other ways attempting to make the background map look more like a google map. We tried GeoServer, uDig, OpenLayer Sources, KML files, JXMapKit and JXMapViewer classes, etc. We would also get into the detailed information later in section 3.2.

In this report, we will give an overview of our code architecture and explain why we choose to do it in that specific way instead of another. We will also gradually show the whole process of this Java visualization project, including our struggles, efforts, functionality of our program and the parts that still require improvements in the future.

2. Code Architecture

Main: The JFrame class that initiate the program

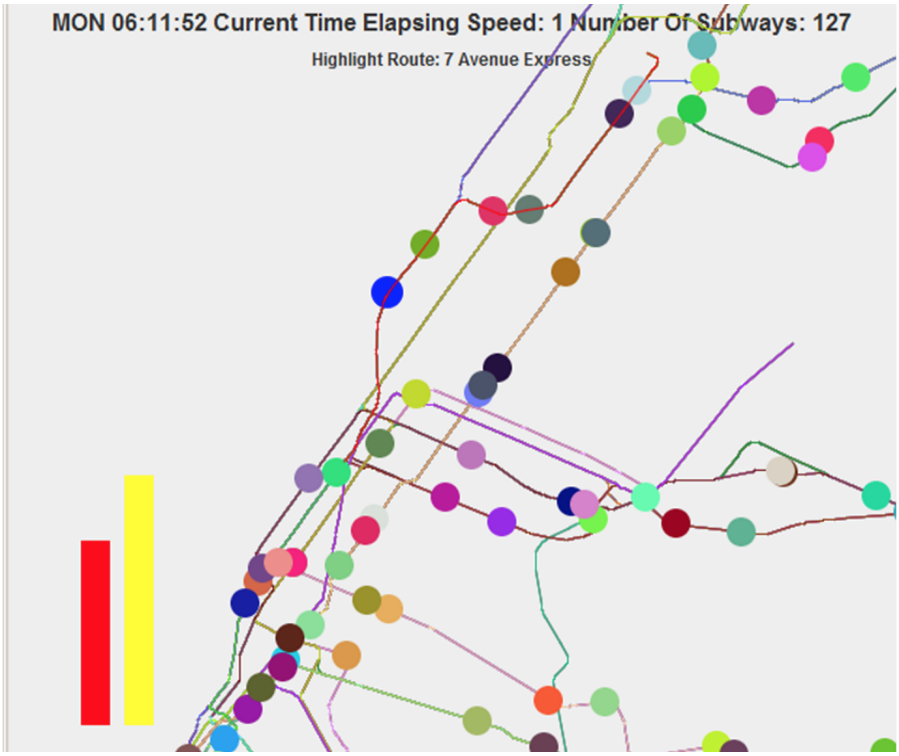
i. Board: A JPanel class that tackle visualization methods, the board updates colors and locations to display and paints on the canvas
    1. Trajectory: Each Trajectory object corresponds to a trip, along with its waypoints and timestamps. The major storage place for processed data. Board will refer to the information in all the Trajectory object to paint vehicles to their proper locations.
    2. Stop: Helper class to temporarily store information of the stops when processing raw data.
    3. Shape: Helper class to store the waypoints in shape.txt file for each route.
    4. Service: Helper class to store the services and calendars.
    5. SpeedColor: Helper class that will generate color for vehicles of different speeds in visualization, supporting Speed Mode.
    6. MassColor: Helper class that will generate color for vehicles that move in different traffic crowdedness, supporting Mass Mode.
ii. GTFSParser: The major class for processing raw data. Will store all the processed data in an ArrayList filled with Trajectory objects and return it to board.
    1. ReadCSV: Helper class to read CSV data in .txt files into String arrays.

3. Features

3.1 Three visualization modes:

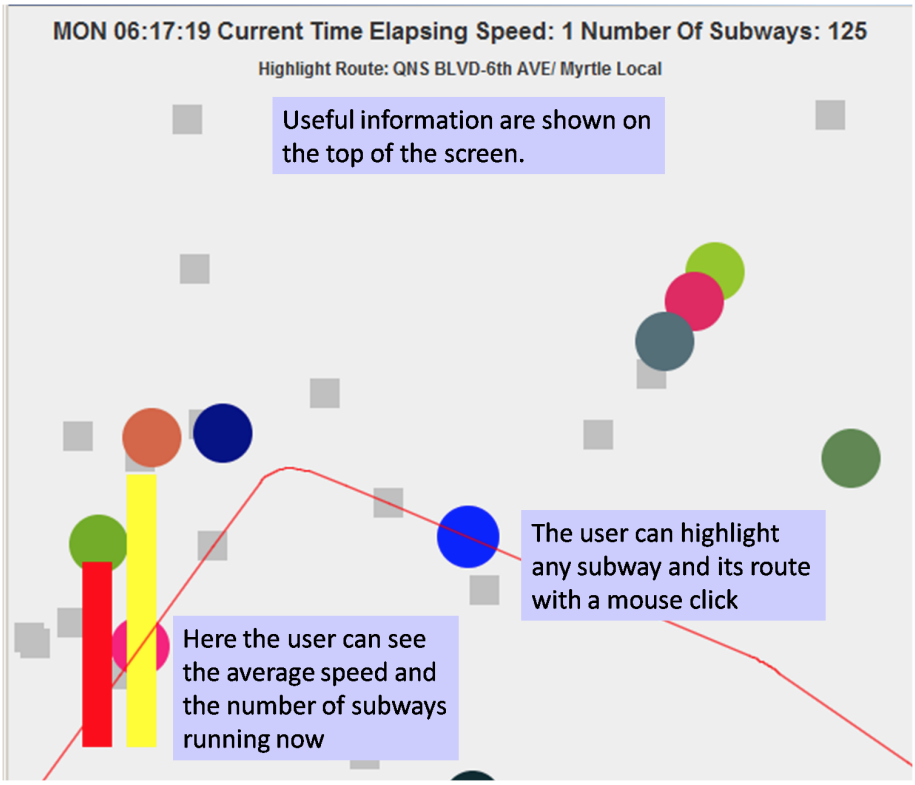
In our program we created the visualization of subway movements according to current time. The user is able to see all the subways in New York City displayed on the screen at their locations at the current time. There are three different visualization modes that the user can select by pressing "C" button on the keyboard.

Features in **Normal Mode**:



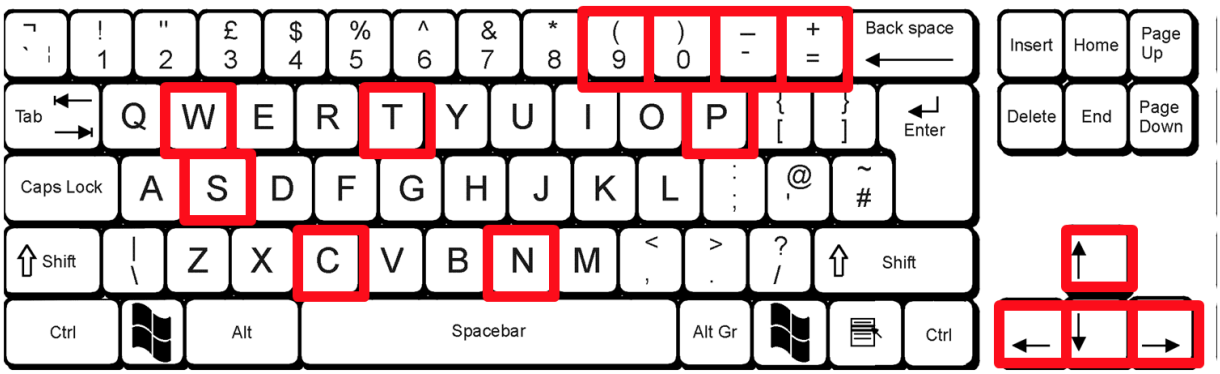
In Normal Mode the vehicles are displayed in random colors

1. The user is able to see all the subways in random colors. Every and each one of the subways has a distinguished color, making it easier to trace and follow.
2. The current weekday, time, time elapsing speed and the number of subways running at the same moment are shown at the top of the screen so the user can refer to the "current" time and the running speed of the time in the visualization system.



A closer look: the user is able to zoom in, zoom out and move the current view

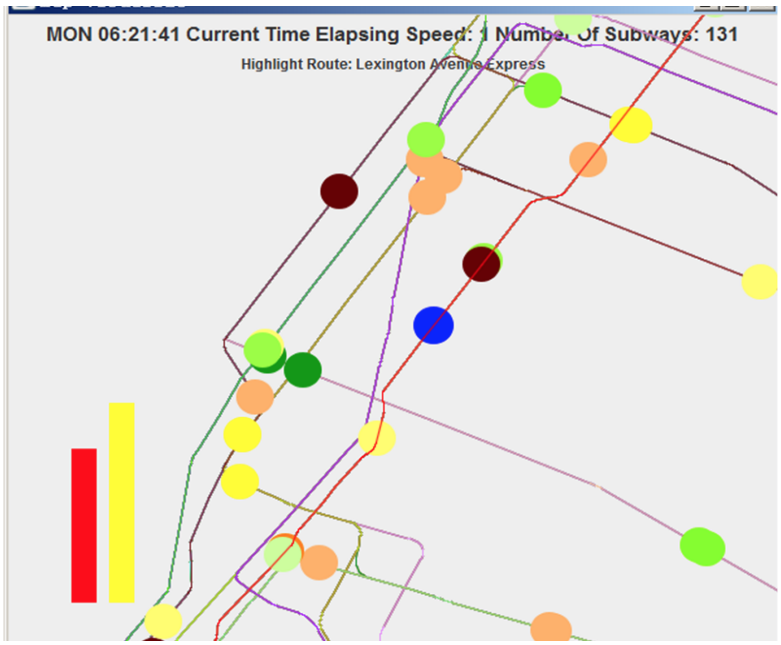
3. The user can move the map by pressing “LEFT”, “RIGHT”, “UP”, and “DOWN” keys and press “+”, “-” keys to zoom in and zoom out. The user can either look at the whole subway system from a city-scale view or check on the details near one station.
4. The user can press “)” and “(” keys to increase or decrease the elapsing speed of time in the visualization accordingly.Thus, user can watch everything flashing by and examine the motion pattern of moving vehicles or closely look a subway for details on a specific route.
5. Press “N” key to change the time to the next hour. User can conveniently see the visualization of different times of the day.
6. Press “W” key to jump to the next Weekday. For example from Monday to Tuesday.
7. The user can also press “P” key to switch on/off stop displays accordingly, Press “S” key to switch shape display; press “T” to switch trips display.



The user uses keyboard to control the viewing mode in the program

8. The number of subways running at the same time and the average relative speed of all the subways are graphically shown at the left of the screen, so that the user can intuitively receive information about the activeness of the subway system at a specific time and the efficiency of the system at the same time.
9. The user can click on any of the moving subways to highlight its route. The name of the route will be shown at the top of the screen.

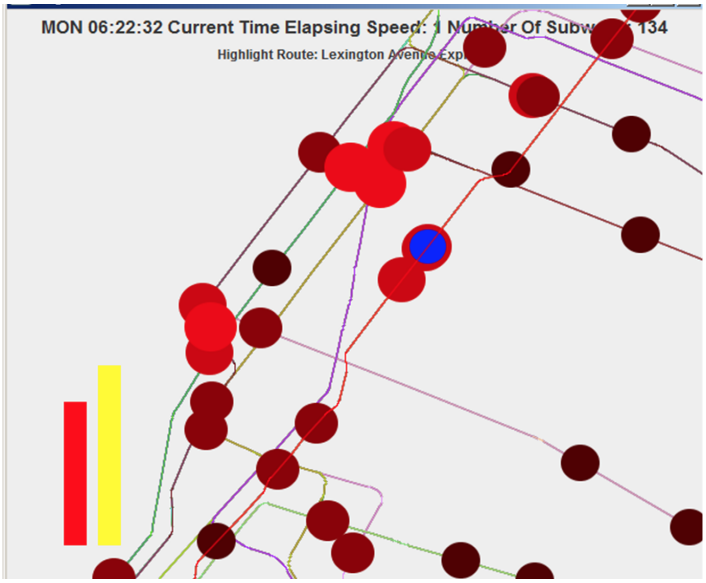
Features in **SPEED Visualization Mode**:



Vehicles are displayed in a color that shows its current speed.

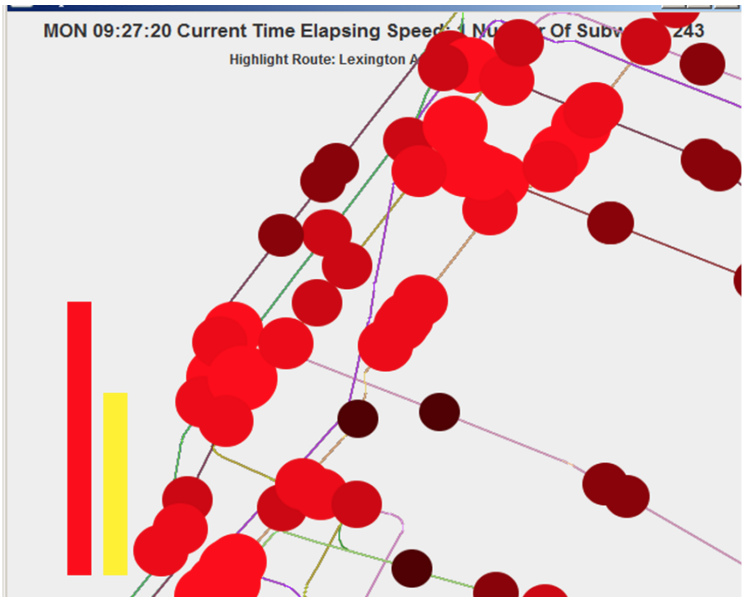
In addition to the features in Normal Mode, this mode displays each of the subways in a color that indicates its speed. The faster the subway goes, the greener its color will be. And slow subways are shown with a yellow, orange or even crimson color. This way the user will immediately find out useful informations about the subway system's overall performance across the city. The user will be able to see which subway lines run faster than others. It's also very intuitive to discover that in some parts of the city the subways generally tend to run slower than other parts of the city. Generally speaking, the Speed Visualization Mode provides more approaches to data analysis on New York City's subway system.

Features in **MASS Visualization Mode**:



In Mass mode the subways are influenced by nearby subways

In the Mass Mode we explore the possibility of the communication and interaction between nearby subways. When a subway has a lot of nearby neighbours, it would grow bigger and the color would become red and brighter, while a single subway only shows a dark color and has a smaller size. Since crowdedness is often related to frustration, we believe this visualization mode will help us understand better the frustration and intensity change in the public transport system.

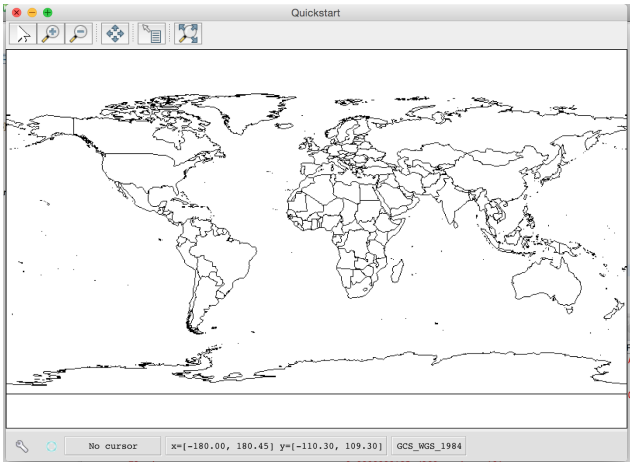


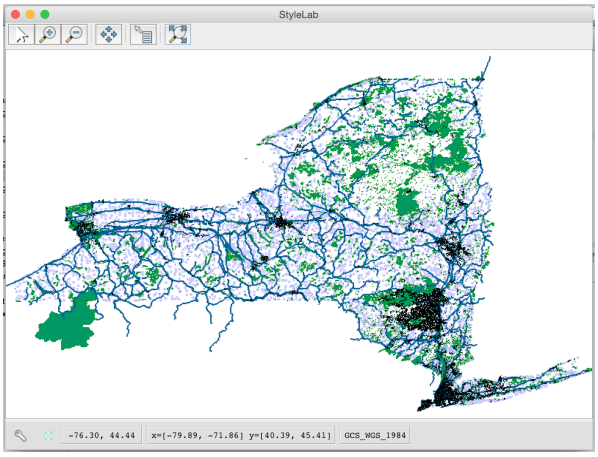
When there are a lot of subways near each other, they form bright red clusters.

From these two screenshots we can see that compared to situations at 6:00AM on Monday, NYC has a much more terrible crowdedness at 9:00AM. This mode can visualize the number and intensity of subways running in the city in a very powerful and intuitive way.

3.2 Background Map

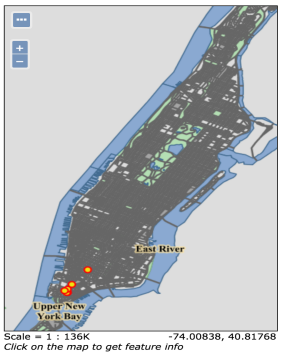
As we mentioned in the Overview section, we were not sure about using Geotools since the shapefile maps are not very suitable as a road map. We carefully followed the tutorial instructions, and finally got from the stage of black-white maps to multi-layer colorful maps. However, as you can see from the map on the right, the shapefile data we collected from geofabric for New York was distractive and contained too much details. This complexity of shapefile data made our map look disorganized and unrecognizable. However, if we add fewer layers to the map, we can hardly see a shape of New York.





Thus, an idea came up to our minds, Google map is a tool that is well-organized and contains useful information as a background road map which might be helpful to our visualization project, how about combining it with raw data for exhibition? Once this idea appeared, we started looking for possible ways to do it with Java. This started a long struggle.

We first tried GeoServer. We had no idea what it can provide before we searched online for solutions. GeoServer is an open source server for sharing geospatial data. We found OpenLayer map sources on it which satisfied our need for a beautiful, dynamic and useful map, thus, we started looking for ways to display OpenLayer maps in Java.



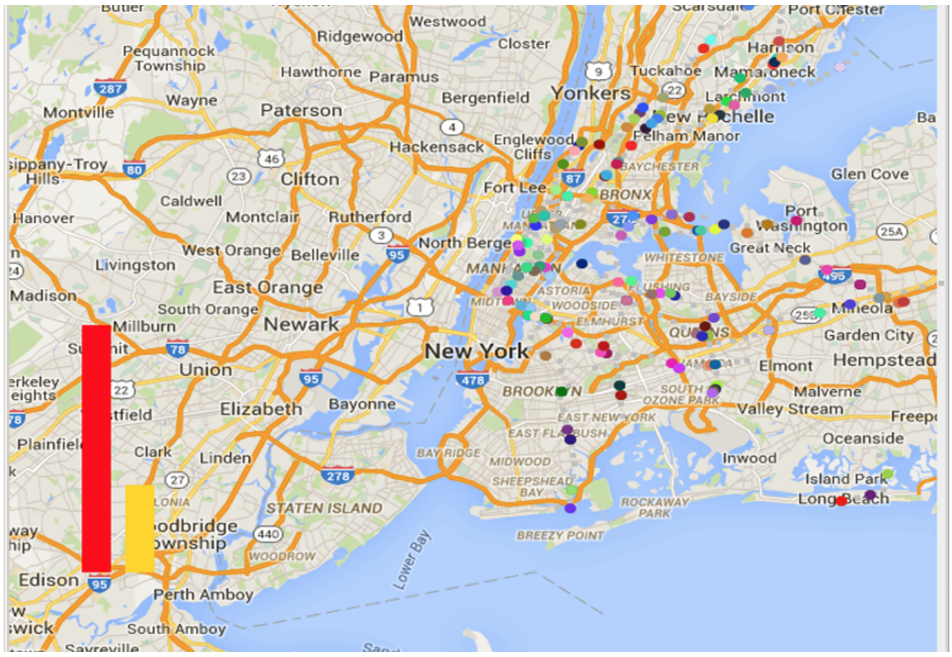
Unfortunately, we failed. OpenLayer is an open source to load, display and render maps from multiple sources on web pages for javascript. So we turned for

other solutions. We later found uDig. uDig is an open source desktop application framework, built with Eclipse Rich Client (RCP) technology aiming to provide a complete Java solution for desktop GIS data access, editing, and viewing.

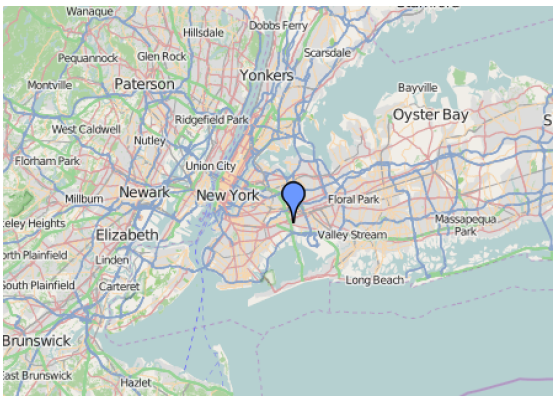
It was true that visualizing the background map became easier, however, it turned out that it's pretty hard for us to add our visualized transportation data on top of the background map. We had to give up this approach again. We started questioning is it possible to for us to balance our pursuit of a beautiful and dynamic map and the visualization of the textual transportation data.

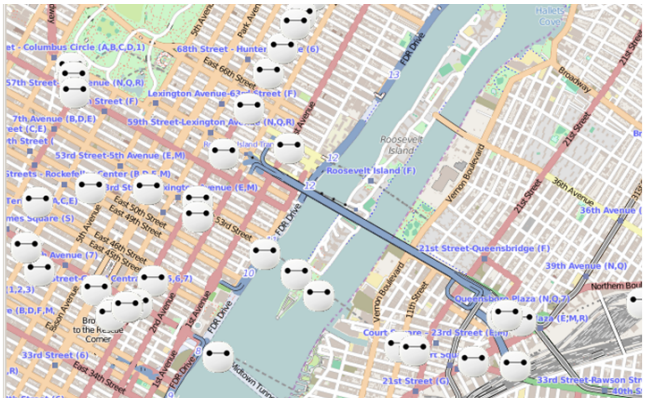
We chose to keep digging. Later we learnt a way to request a google map picture by using the address:
http://maps.google.com/maps/api/staticmap?center=(40.668897,-73.932942)&zoom=2&size=500x500&maptype=roadmap
Google map responds to our request according to the center value, zoom value...all the other features we typed in. Intuitively we considered to display the background map by simply displays it as a background image. And then when zoom-in or zoom-out, we request a different map from google according to our new need.

This idea sounds novel and simple, but there is one major problem. The google map we requested was just an image, it didn't contain any GPS coordinates information, thus, we had to calculate the ratio ourselves and then displayed it accordingly. It was an annoying job, because we didn't know the exact distance between any two stops. (Google map could only gave us an approximate number.) And yet we didn't know the precise distance of the same two coordinates on our screens (Similarly, rulers could only give approximate information). Thus, as you can see from the following figure, those stop locations have slight biases. However, that's the best we have get so far.



Finally, we discovered two class called JXMapKit and JXMapViewer. They were very useful tools to display maps in Java. Luckily we successfully overpainted our stop locations over the beautiful road map. Moreover, we successfully make BayMax(as subways) move along the routes. It is the inspiration for the Mass Mode. At that time we thought that we might show a map where a lot of Bay Maxes are representing subways and interact with each other.



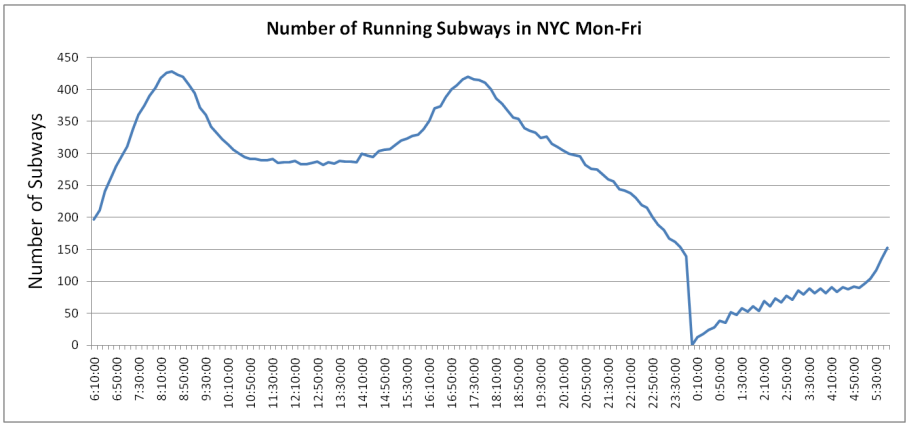


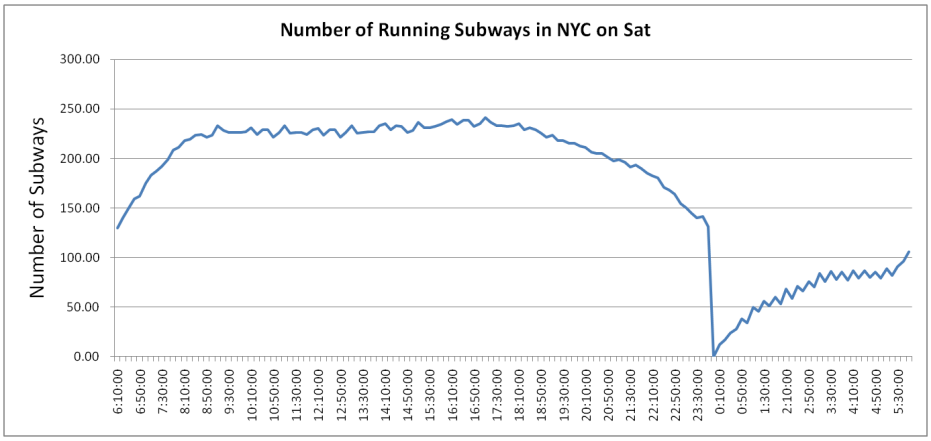
4 Interesting Results

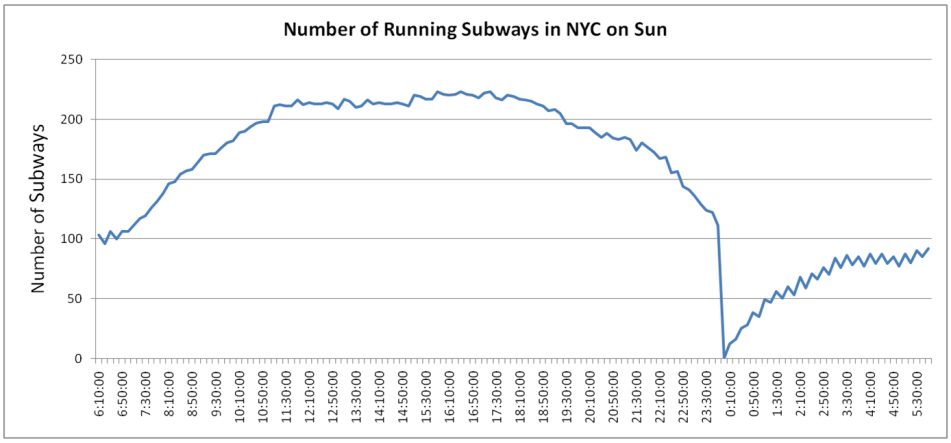
We are able to get some interesting results from the data we collected and processed. So we decided to go further in the idea of visualization by exploring these results and trying to find some interesting facts or conclusions from them.

As you can observe from the following 3 charts, from Monday to Friday, at around the time of 8am and 17:30pm, the number of running subways reaches its maximum. This is in consistent with the busy commute time on weekdays.

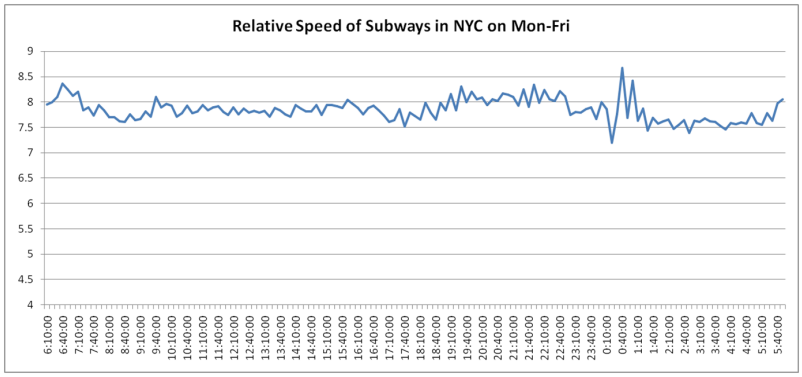
However, during weekends, the maximum number of subways decreases to less than 250. This indicates that people usually choose to stay at home or another means of transportation on Saturday and Sunday and the city's subway system might be designed to meet that change. Moreover, there's no outstanding peak time nor peak number for transportation during weekends, which implies that people may have a freer and balanced schedules.

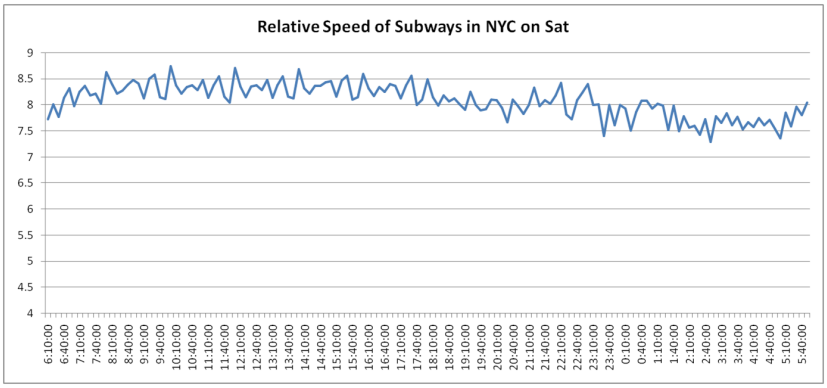


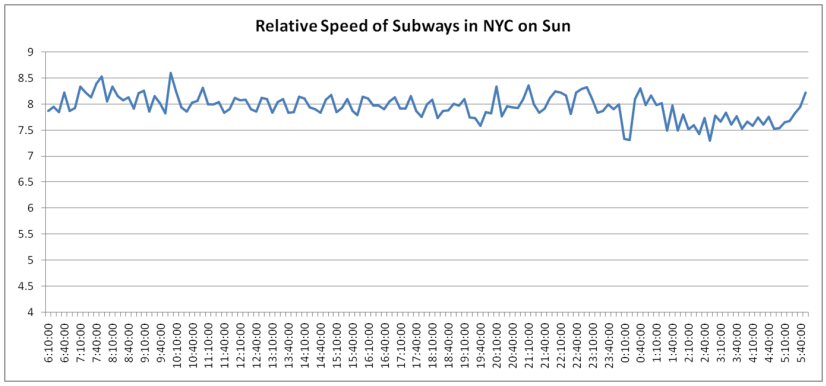




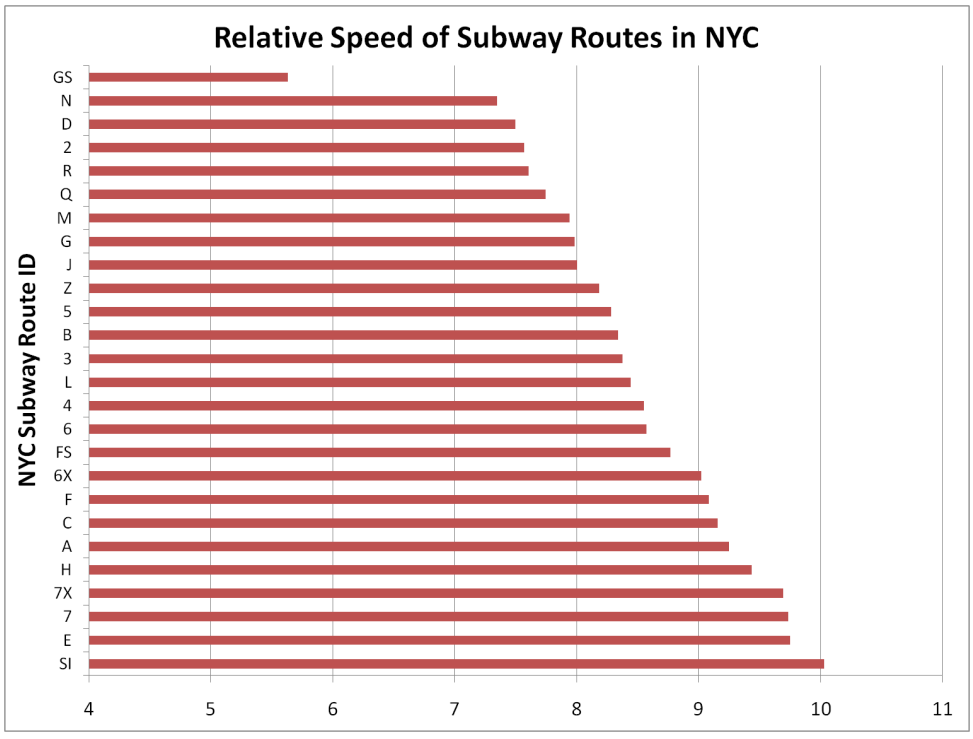
We thought that there might be a relation between the average speed of all running subways and the time of the day in New York City. However, it turned out that the average speed would neither be affected by the variations of days nor time. This might due to the fact that subways run beneath the city ground and thus are not strongly affected by the traffic situation. The average speed is quite constant.







In the last chart we put the speeds of different subways in an increasing order. Thus, we can easily observe that SI subway is the fastest one while GS is the slowest one.



5 Conclusion

We believe this visualization project will help the public to understand better the characteristics of the subway system in New York. We hope that by visualizing data, we can provide the public with an easier approach to these data that are widely available but not so often make sense to common people. Furthermore, we believe this project might be valuable in exploring the relationship between the geographical features of the city and the public transport system that locates and closely connects to it. By providing an easier way to this exploration, we might be able to facilitate the design of better and more efficient public transport systems.

Another powerful part of this project is that it only require slight changes for the project to apply to public transportation data of other cities, provided that they are in a standardized GTFS format and stored in CSV files.

6 Acknowledgement

We greatly appreciate the help and guidance provided by Professor Promethee Spathis, a computer science professor at NYU Shanghai throughout our project.