Programming Project - Group 3

By Aran Nolan, Cal Martin, Daniel Crawford, John Milson and Shane Fay

"Displaying Taxi Information Is Easy With A MySQL Database And Modular Design"

Outline of design:

From the very outset of our project, we decided upon a modular design for the project. We decided to design an interface for the backend of our application to return queries in a uniform manner. This allowed for each team member to work on individual visualisations, and ignore how the queries were handled. As the backend was modified for better performance, each visualisation continued to work, as we had all agreed upon a uniform method of getting queries. We did this also for our visualisations, combining all the map displays into one class, with each visualisation extending off it. This allowed for a fluid method of switching between visualisations.

Work Distribution:

After the initial prototype from Cal, we decided to split up the project as follows:

- Cal initially worked on the backend of the application, designing a query interface and compressing the MySQL database down to 2.5GB from 5GB. He also created the initial HeatMap visualisation. Cal also designed the Current Query class, which allowed for all visualisations to share the same queries. It also used threading, to allow for queries to be fetched in the background while the application was running.
- Daniel focused on the TripAnimator visualisation, as well as improving the HeatMap visualisation to include a Heat Gradient. He added accurate to the minute sunrises and sunsets in taxi animator, as well as the choice of displaying the taxis by day in real time, up to a speed of 1000 x realtime.
- Shane set to work on his Location Visualisation, which tallied up the amount of taxis dropping off at particular locations marked on the map. He then also created the Stats Visualisation, which compared two queries with pie charts and data.
- John worked on the LinePieChart visualisation, which provided a very interesting look at the data, comparing the time of day with the amount of taxis at that point.
- Aran implemented the Query Comparison graph, which compared two queries on the map with different coloured towers. After this, he focused on getting the GUI working, creating an incredibly flexible query GUI, as well as a drop down menu for choosing between visualisations.

Features Implemented:

- Heat Map:
    - This was the initial prototype for our application. It takes a section of the map, which can be changed, and divides it into a 300x300 grid. It then takes a query as defined by the user, and displays a height map of where the taxi pickup points, as well as a heat colour based on the amount of taxis present. The heights of the towers are relative to the size of the query, while the heat colour is an absolute value which can be used to compare two queries.
    - Cal also implemented 3D mouse picking in this visualisations, to allow you to see the amount of taxis in each tower. This is done by creating a separate offscreen buffer, which you then draw all the towers to, each with a unique colour value. What this allows you to do is then get the colour value of the tower at the mouse(x,y) location on this offscreen buffer. Since each tower has it's own unique colour, you can then tell which tower the mouse in over in the 3D onscreen buffer.
- Taxi Animator
    - Taxi Animator shows all the trips in the set queries, showing the data from query 1 as yellow "taxis" and the data from query 2 as red taxis.
    - Taxis move from their start point to end point in the same time as the real trip took, although no pathfinding was implemented.
    - 2 modes are available - one mode shows the data "as it happened", across the two months. The other shows all the selected data compressed into 24 hours. This allows us to easily view directions of travel based on time of day.
    - Controls were added to change the time and speed of the visualisation. Supported speeds range between paused, realtime and approximately 1000 times realtime. Skipping forward and backward by an hour or a day is also available. Skipping backwards proved challenging as the appropriate taxis had to be "reset".
    - Various cosmetic features are implemented. Sunrises and sunsets change the background and ambient light, and are accurate to the minute to recorded data. Taxis in Manhattan are lit by point light sources, so they remain bright at night whereas taxis further afield are duller.
- Area Map Graph
    - This visualisation, created by Cal, is an interesting way at viewing the data. Similar to HeatMap, this visualisation draws circles at varying heights, radii, and colour depending on the amount of taxis at a particular location. With this visualisation, you can very quickly see the concentration of taxis at certain areas.
- Query Comparison
    - Heat Map was used as a basis for this visualisation, thus has similarities.

    - The purpose of this visualisation is to allow direct comparison of two different datasets: Query 1 shown in blue, query 2 shown in red. As with Heat Map, the height of the towers are proportional to the size of the data queries.
    - It offers the choice of two modes, ‘pair’ mode and ‘telescope’ mode. The ‘M’ key swaps between the two. Pair mode quite simply places two towers next to each other in each grid-cell. Telescope mode places the towers on top of each other, the shorter tower being the ‘base’ and the taller being the ‘peak’. The base is wider than the peak so that one does not entirely consume the other. If both towers are the same height a single tower represents them in a magenta color.
- Location Popularity:
    - Using the Location class already present in Unfolding Maps, specific points of interest were added to the map. They were displayed by default as grey circles flat on the map. Subsequently triangles were drawn above them that rotated to always face the user, along with text detailing the name of the location, and the number of visitors. Visitors were defined as any taxi with a drop off location within +-.01 longitude and latitude of each given location.
- Time/Passenger Chart

- Statistics Visualization
    - This simple visualization provided a static display of some information from a given query. number of passengers and vendor distribution were both displayed as pie charts and records (longest, shortest trip) as well as average were displayed underneath.
- Line Pie Chart
    - This visualisation takes a simple set of data, trip pickup time, and number of passengers and displays it in an interesting way.
    - It plots the points around a 24 hour clock face based on their pickup time, and their height above the clock face depends on the number of passengers.
    - The lines that join into the center make it easy to see where there are large clusters of taxi trips. For example, depending on the query set, it is usually clear that a large number of people are getting taxis in groups at around 4 am, this is most likely due to nightclubs closing.
    - On top of this, the lines into the center can be toggled on and off by pressing “0”, making the visualisation more like a 3D bar chart, making it slightly easier to read off the information.
    - Then animations were added for cosmetic purposes.
- GUI
    - Using the ControlP5 processing library, an intuitive to use GUI was implemented.
    - A dropdown-list in the top right corner of the window allows the user to select between the visualisations.

    - The main GUI feature is the Query selection panel. Sliders, buttons, checkboxes and radio buttons offer an easy way to create a detailed query. Upon pressing one of two ‘SET QUERY’ buttons, an SQL command is formulated according to the current selection. A new thread is set up to retrieve a query from the database. Upon completion, the current visualisation is informed and the data on screen is updated.
    - Post deadline, a hotkey help menu was added.

Problems Encountered:

- Memory Optimisation - There was a huge amount of data in the CSV files, and so much initial work was on decreasing the amount of memory needed to load the data into memory.
    - In order to dramatically reduce the size of the database, Cal created a data preprocessing class to strip out the Taxi Medallions and Hack numbers. We took each unique value, and replaced it with an integer, and stored the original medallion in a text file. This allowed us to shrink the file size by a factor of two! The medallion and hack files are then loaded into the application again during runtime, and the index of their respective arraylists matches the integers in the database. This was a massive memory saving, as there was only around 15,000 unique medallions, in the 30 million row database!
    - Pickup and Dropoff date and time were each initially stored as strings. Daniel created a DateTime class to allow converting each date and time into a single long representing seconds past midnight January 1st 2013. As we already had the trip time in seconds, drop off time could be calculated easily at need and didn’t need to be loaded into memory at all. This was a massive saving in memory as we converted two strings into a single long. It proved surprisingly challenging to calculate all the times correctly, so Daniel created an automated test to check every valid time in the year and convert it to seconds and back again.