World Academy of Science, Engineering and Technology
International Journal of Civil and Architectural Engineering
Vol:2, No:10, 2008

# Visualizing Transit Through a Web Based Geographic Information System

Ricardo Hoar



***Abstract*****—** Currently in many major cities, public transit schedules are disseminated through lists of routes, grids of stop times and static maps. This paper describes a web based geographic information system which disseminates the same schedule information through intuitive GIS techniques. Using data from Calgary, Canada, an map based interface has been created to allow users to see routes, stops and moving buses all at once. Zoom and pan controls as well as satellite imagery allows users to apply their personal knowledge about the local geography to achieve faster, and more pertinent transit results. Using asynchronous requests to web services, users are immersed in an application where buses and stops can be added and removed interactively, without the need to wait for responses to HTTP requests.

***Keywords*****—** Geographic Information Systems, Public Transit, Web Services, AJAX, Human Computer Interface

## I. Introduction

FUEL costs and environmental concerns present new obstacles to transportation. Our reliance on personal transport has spawned numerous simulations of traffic flow [8], light timings [7], pedestrian crossings[9] and other intelligent transportation applications [3]. Unfortunately, this focus has overlooked the mode of transport with the greatest potential: public transit. Public transit offers us a potential solution which does not require substantially new investment or infrastructure. Despite these potential benefits, many people have a negative opinion of public transportation, and will continue to drive their personal vehicles.

Transit's negative image stems from many independent variables including punctuality, frequency, overcrowding, security, schedule information, and others [6]. If access to transit information is hindered by poor user interfaces, large lag times, or difficult street name abbreviations, the negative image of public transportation will be reinforced. However, if we provide better, more intuitive access to information, we empower transit users, create a better overall experience, and effect a positive change in the public perception of transit [14]. Research of transit models, and route planning is extensive [2], [1], [17], but there remains limited research presenting the existing data to end users through the web [10].

For this research project, the public transit system of Calgary, Canada; a city with over 1 million citizens, is used. With 2 light rail transit lines, nearly 600 vehicles in service at a peak times[15], and over 5500 transit stops, the amount of data is significant, and through a classic web interface, absolutely overwhelming. Given the prevalence of successful Google Maps projects [12], a web based application has been developed to explore alternative techniques of schedule dissemination using the PHP scripting language[5], MySQL 5 database [11], client side Javascript and Google Maps API[4].

This paper describes the techniques and strategies developed for transit data visualization & web services that can be generalized to any transit system with similar schedules and scope. In particular, the database design, web services, user interface, and map visualizations will all be described, illustrated and analyzed to determine if these visualizations are effective.

The ongoing research project can be seen at *http://transit.mtroyal.ca*.

R. Hoar (*rhoar@mtroyal.ca*) is with Mount Royal College, Calgary, Canada supported by the 2008-2009 Petro-Canada Young Innovator Award.

## II. Public Interface

The public interface to a myriad of data presents a challenge for any interface designer. Current systems require users to input or select routes and stops in a multi step form submission process. By displaying schedule data through a GIS, we are able to leverage the viewport of a user as an explicit input of a location. This input can then be used to determine what information to display. Applied to transit, a location can serve as input to retrieve nearby routes, buses and stops.

However, more experienced users of transit may prefer to see a particular route, without the need to input a geographic location. Therefore a second interface was created to allow faster retrieval of data using an input box to select and add particular buses they are interested in. These two interfaces are called; the *location search* and *bus list* shown together in Fig. 1.

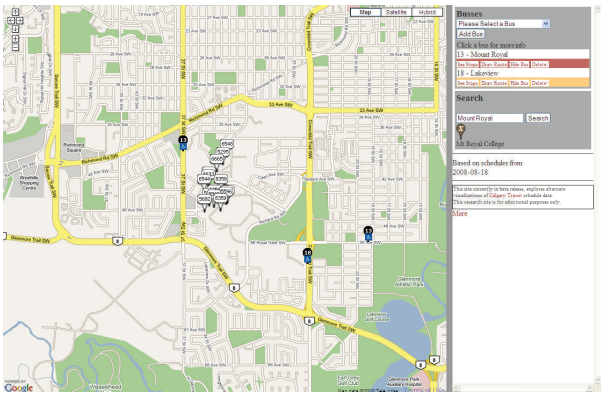



Fig. 1. Screenshot showing the overall interface

The *bus list* interface allows user selections to input exact values for queries to the web services, through add and delete options (Fig. 5). This direct interface allows experienced users to add and remove buses they are interested in, giving them

direct access to the information they require. Users who select buses through the location based interface will eventually notice those buses being added to the list. This list and it's associated options are always displayed for all bus routes being drawn allowing users to toggle the various visualizations depending on their particular needs.

To supplement the expert users knowledge, the *location search* interface allows users to browse nearby stops and routes using a search box. Using Google's asynchronous geocoding request system, a request for a location can be translated into a location on the map which can query a web service to determine nearby stops and routes (Fig. 3). Since information will be displayed on the map, these type of interactions will be more natural to non expert users of the system.

Rather then segregate data, and require multiple interfaces and queries to display disjoint data sets, users are able to select multiple routes, buses and stops at once, providing them with data sets that help support their decision. Classic web systems require separate form submissions for each distinct bus route. By obfuscating and asynchronizing the requests made to web services, and allowing buses and stops to be added and removed in an easy way, the user is immersed in an application with fewer refreshes, and no loss of control over the map with each request made.

### A. Asynchronous Requests and Responses

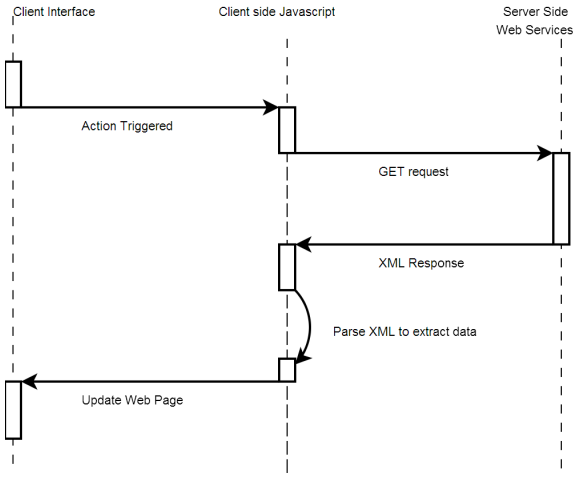



Fig. 2. Diagram depicting the typical web service interaction

To create web applications that behave like traditional desktop applications, AJAX technologies are utilized [13], [16]. Each time the user makes a request, a *HTTP GET* request is sent to a web service. This web service then returns an XML response to the query which is handled by the javascript callback function set to handle it. If the response is delayed due to bandwidth traffic or server load, the application continues to run, and allow user interaction until the response arrives. The response is parsed, and the specific data is presented to the user through the interfaces described below. A diagram of this asynchronous process is given in Fig 2.

### B. Location Search Interface

Geographic information systems are often used to search for a location given a street or business name. To facilitate such an interaction, a location search interface was developed using the geocoder from Google to present the results in a useful context.

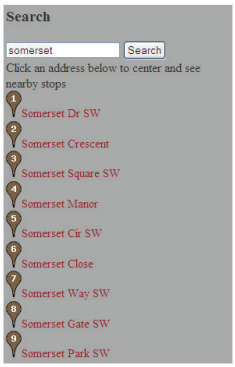
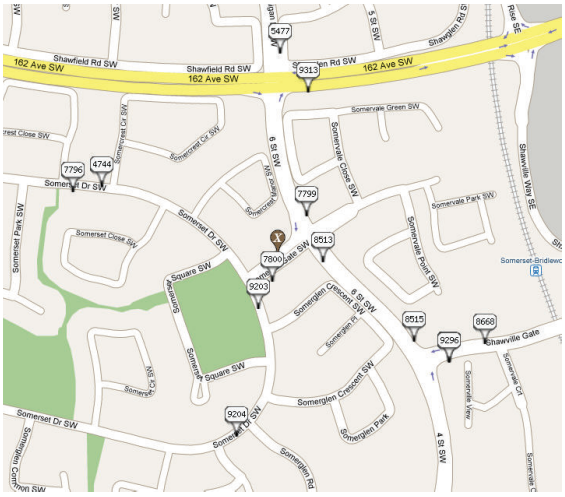
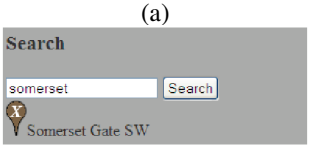



Fig. 3. Search Interface showing possible results

The user enters their query into the search box and clicks the search button. The request is sent to Google geocoding services which return an XML formatted result set containing possible matches. These possible matches are presented as a list to the user as seen in Fig. 3. The user can then select one of the matches to centre the map and show nearby bus stops (Fig. 4). This process merges the transit data with a particular geographic location of interest empowering the user who wants to see transit service in a particular area.




(a)




(b)

Fig. 4. The map displaying a location and nearby stops (a), and the associated search interface (b).

Each bus stop can display the upcoming stop times and buses, so the user can determine which buses service the

nearby area, merging the useful data from transit with a particular geographic location of interest. In future work, this location search will be the initial stage of selecting a start and destination for trip planning purposes.

### C. Bus List Interface

The major function of this interface is to allow the fast retrieval of a particular bus, by name or number. Each bus has visualization options, and a way to get stop times. As seen in figure 5 the interface allows users to click on javascript hyperlinks to toggle the display of routes, show and hide stops on that buses route, and to delete that Bus from the list of displayed buses. Each of the hyperlinks has descriptive text that changes when clicked, so as to enable toggling by clicking on the same link.

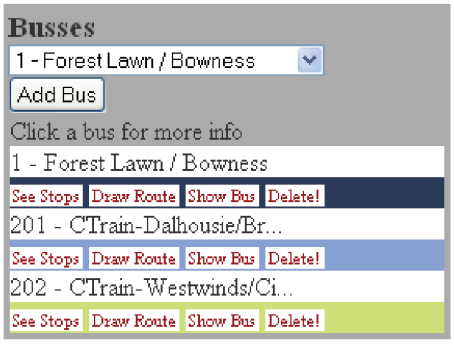



Fig. 5. Screenshot showing the interface for each bus.

*1) Visualizing Routes:* Since the order of stops in a bus route are known, and a particular bus route is requested, a route can be displayed by drawing a line on the map connecting the stops in order. The route can be displayed at all zoom levels since the polyline is redrawn with each map interaction, using the coordinates of the stops on the route. The user clicks the link *Draw Route* under the bus they wish to draw. The route will display, and the text will change to say *Hide Route* to allow toggling of the map with one link. Fig. 6 shows a bus route being displayed on the map. The route displayed does not necessarily depict the actual path of the bus, but rather a connection of straight lines connecting the stops in order. Although this discrepancy between the path travelled and the path drawn limits the accuracy of the route drawing, it provides implicit stop locations. Furthermore, large distances between stops are more readily observable then they would be if an exact trace of the bus path was used.

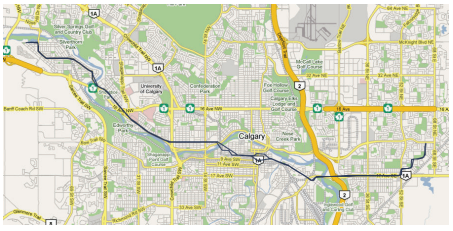


Fig. 6. Screenshot showing a route as a dark line overlayed on a map

Because multiple polylines can be displayed at once, multiple routes can be displayed. However, to ensure users can differentiate the routes, multiple colours must be used to distinguish routes apart.

*2) Visualizing Stops and Upcoming Stop Times:* If a user wants to select a particular stop, or browse stop locations, they click the *See Stops* link. This sends a request to the *Route* web service. Upon returning the list of stops associated with the bus is displayed. Users can move their mouse over the list of stops, with the map centering on each stop as they are selected (Fig. 7).

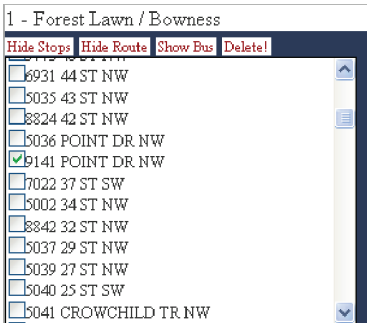



Fig. 7. Interface showing the list of stops for one bus

When a user checks the box next to a stop, that stop appears on the map as an icon with it's number. Icons are generated so that each stop is a unique image. Fig. 8 shows example icons.


Fig. 8. Generated icons for a stop, bus and a wheelchair accessible bus.

No HTTP request is required since the list of stops is cached from the earlier request. This, or any other stop icon shown on the map can now be clicked to trigger an asynchronous request to the *Stop Times Web Service*. The stop number and the time are sent, and the next few buses to stop at that stop are returned and displayed in the icon's pop up window (Fig. 9).

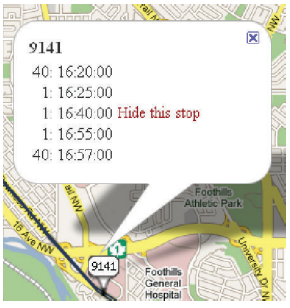



Fig. 9. Screenshot of a clicked stop icon displaying upcoming times.

*3) Vehicle Plotting:* To indicate the location of a vehicle, an icon is used. To allow users to distinguish between buses, icons are generated for each bus number as seen in Fig. 8(b). Buses which are handicap accessible are indicated with a different

icon to give immediate information about the bus type ( Fig. 8(c)).

With the *Bus Position Web Service* able to return the current location of all buses, the map can display icons at each of the locations returned. By making this request at regular intervals, and refreshing the bus icons, the bus appears to move on the map in real time. This emergent effect is valuable to riders who can anticipate the arrival time of the next bus, through their own experience with the route and the current location of the bus.

It also provides a fast overview of the number of buses on a particular route. If the route is littered with multiple buses, one can infer the service is at a high level and less preparedness is required. However if only a single bus can be seen on the route, then the user knows more care must be taken in catching it. Together with actual stop times, and route visualization, the vehicle plotting provides a powerful and intuitive visualization of transit data.

## III. Data

Calgary Transit, like many public transportation systems, produces a wide range of information. The data is available to the public through the web site (*http://calgarytransit.com*) and an automated telephone system. Although the process a user must undergo in the current system is relevant to transit schedule usability, it is not generic enough to warrant in depth discussion in this paper. The uniform manner in which Calgary transit data is presented is well suited for automated retrieval. For this research, the data sets retrieved include: the bus names and numbers, the stops on each route, and stop times for each route for every day.

### A. Database Design

Data is conceptually separated into temporal and non temporal data. The list of buses, the route plans and the stop locations are all treated as static. The daily schedules are the temporal data which can be fetched anew every day. For an illustration of the tables and relationships see Fig. 10 where static tables are grouped on the left, and temporal tables on the right.

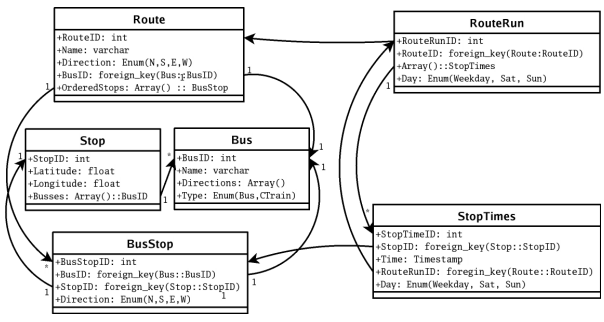



Fig. 10. ERD diagram depicting the relational database design

The *Bus* table stores the name, number and possible directions of each bus. The *Stop* table stores the latitude, longitude, name, and a list of buses that stop there. These intuitive tables are supplemented by a normalized version of *Stop* called *BusStop* which contains multiple entries for each stop, one duplicate for each distinct bus that stops there. The final static data is the *Route* table which maintains an ordered list of Stop numbers referencing the *BusStop* table, a direction and a reference to *Bus*. This data allows the drawing of routes on a map, by using the ordered list of stops from *Route* in conjunction with positional information from *Stop*.

Integrating the schedules requires two more tables. The *RouteRun* table uses one row for each such route execution, storing the start and end times and the day which provides temporal data to facilitate time based queries.

Every time a bus is scheduled to stop corresponds to a row in the *StopTimes* table. Each entry refers to the *Bus*, *RouteRun* and *BusStop* tables. This table is the largest is magnitude, with over 2 million rows for a week's worth of schedules. A summary of the database with one week of schedules is shown in Table I.

| Table | Rows | Size (in KB) |
|---|---|---|
| Bus | 284 | 15 |
| Route | 300 | 72 |
| BusStop | 10,544 | 455 |
| Stop | 5,547 | 575 |
| RouteRun | 36,052 | 15155 |
| StopTimes | 2,236,427 | 146944 |

TABLE I
A LISTING OF DATABASE TABLES ORDERED BY SIZE.

### B. Geocoding Stop Locations

The data acquired from Calgary transit does not provide coordinates for the stops, only names and numbers. Since geographic information systems require geographic attributes of displayable items, this data had to be acquired from a third party or entered manually. Since manual updates could be necessary in any event, a mapping interface which utilizes many of the GIS techniques for the main site was developed to plot bus stops. This administrative user interface allows each stop to be dragged onto it's perceived location, which in turn updates the underlying database through an asynchronous call to a web service. Fig. 11 shows this interface, with the stop list on the right, and the selected stop ready to be dragged. The different colours indicate the status of the stop: hand plotted, geocoded from Google, and unknown.

However, with over 5500 stops in the Calgary transit system, plotting each one manually would be too onerous a task. Therefore, using the information contained in the stop name, the location of each stop was roughly determined through a geocoding request to Google. Stop names were first cleansed of extraneous data such as the bus direction, and words such as "over" and "under". The resulting string was then further parsed to replace street abbreviations with standard street types, which the geocoder could interpret. Once the cleaning process was completed the request was sent, and the result analyzed. If the result was within the city limits, it was recorded with a flag to indicate it was a result of a geocoding request. If a stop fell outside the city limits, or did not return

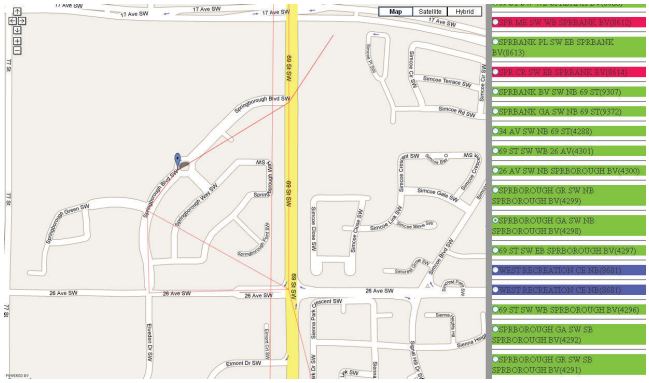



Fig. 11. Plotting interface, where stops can be updated by dragging and dropping

a coordinate, it was marked as undefined, and would later be coded by hand.

Using this geocoding technique over 90% of the stops were retireved. Manual refinement and verification of these stops, and the manual input of the remaining stops was then undertaken to identify and correct erronious stop locations. With stops now plotted, the visualization of bus stops, busses, and routes can occur through web services.

## IV. Web Services

Throughout the public user interface, users will make requests to change what information they want to see. By asynchronously answering their requests the interface does not refresh itself like a classic web page. Rather, the map and interface continue to be interactive and work correctly, with the request response resulting in new data being displayed as soon as possible. At the heart of this asynchronous request response system is a set of web services which respond to user queries. All web services will refer to the databases design presented above.

Each web service resides on it's own unique URL to differentiate and partition the services. All requests are made through HTTP GET requests facilitating asynchronous javascript. Each service replies using an XML encoded response containing the pertinent data.

### A. Route Service

Every bus in the system has one or more associated routes. There are 2 scenarios where a user will request this web service: a request to display the route as a polyline on the map, or to see the full list of ordered stops. Each point returned in the XML therefore has latitude and longitude as well as a display name and number. This data is easily retrieved from the *Route* table joined together with the *Stop* table. The parameters passed to this service is simply the route to return. A request to get route #1 would be:

*GET route.php?route=1.*

### B. Bus Position Service

If a user requests to see the state of buses at a particular time, or the current time, the interface sends their request to the bus position web service. A request will encode the time, and a list of - separated buses. For example, to see the buses 1 and 201 at the current time the web service would be called at:

*GET bus.php?bus=1-201&time=now.*

Using the passed time and buses, a query determines all the buses currently on the road using the start and end time stored in the *RouteRun* table. For each route on the road, the next and previous stop times are then retrieved from the *StopTimes* table. Using the two locations and times, we can calculate the current location of the bus by interpolation of the stop coordinates. These coordinates are returned in the XML response for client side processing.

### C. Nearby Stops Service

When a user wants to know which stops are near a particular location a request to the *Nearby Stops Service* is sent which includes the latitude and longitude associated with the area of focus. The XML result set includes the name and location of the nearest stops to the coordinates passed. This response can then be parsed and displayed to the user on the map. A query to get stops near Mount Royal College would be:

*GET /stops.php?lat=51.01672450&lng=-114.1296672.*

### D. Stop Times Service

Finally, a user may want to request stop times for a particular stop number, and a date. This web service will return upcoming stop times. Using the *StopTimes* table, the particular stop number requested will return all the times a bus stops at that stop. Specifying the next *n* stop times after the date stamp passed, refines that result set into useful data which is returned as XML to the requester. The query to get the next stop times at stop 6700 would be:

*GET stops.php?mystops=6700&time=now.*

## V. IP Based Customization

Many campuses and large corporations are assigned blocks of IP addresses, which easily identifies computers visiting from such locations. Using this knowledge, we can provide different default settings for visitors from specified Ip ranges, so that the map is centred around the location of the IP addresses, with local stops and buses being displayed by default.

## VI. Conclusion

The current scheduled location of a particular bus is of interest to transit users. Using an asynchronous web based map application the location of a bus can be accurately and intuitively visualized providing information to the user which was previously obfuscated by overwhelmingly large data sets.

Seeing a bus route is of great importance to transit users. By allowing multiple routes to be drawn on an interactive web based map, users can see a bus route at multiple zoom levels, and situate themselves with map and satellite imagery. When multiple routes are viewed in parallel, intersections between routes can easily be seen, which is not possible with a disjoint presentation of individual routes.

These visualization techniques do not create new data, but rather allow the geographic attributes inherent in the data to be expressed intuitively. By increasing the perceived quality of the data and the ease with which information can be retrieved, the shortcomings of classic web based schedule dissemination techniques can be overcome, resulting in a better experience for the end user. By empowering transit users, we grant all users expert knowledge formerly reserved for experienced riders only and begin to address some of the perceived shortcomings of public transit.

## VII. Future work

With data retrieval, and several visualizations in place, future work will look towards further enhancing the user experience, and providing concise summaries of complex situations. Improvements will include user based customization where individuals can save their preferences which will then be applied to a trip planner, which can suggest transit trips based on start and end locations.

Integrating third party data such as weather, distance and speed between stops will increase the accuracy and provide a better basis for planning trips. Additional visualizations to illustrate the discrepancy between the scheduled and anticipated arrival time will provide even more useful information to users.

Finally, experiments which evaluate the effectiveness of the final interfaces will help us determine which visualizations and techniques are most and lest beneficial. Surveys and usability studies will assess the effectiveness of the web based GIS in comparison to traditional dissemination techniques.

## References

[1] Moshe Ben-Akiva Asad Khattak, Amalia Polydoropoulou. Modeling revealed and stated pretrip travel response to advanced traveler information systems. *Transportation Research Record: Journal of the Transportation Research Board*, 1537:46–54, 1996.

[2] S. Travis Waller Athanasios K. Ziliaskopoulos. An internet-based geographic information system that integrates data, models and users for transportation applications. *Transportation Research Part C: Emerging Technologies*, 8:427–444, February-December 2000.

[3] L. Figueiredo, I. Jesus, J.A.T. Machado, J.R. Ferreira, and J.L. Martins de Carvalho. Towards the development of intelligent transportation systems. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 1206–1211, 2001.

[4] Google. Google maps api reference. http://code.google.com/apis/maps/documentation/reference.html.

[5] The PHP Group. Php: Hypertext preprocessor. http://php.net.

[6] Inc. HarGroup Management Consultants. Calgary transit customer satisfaction survey 2007. http://calgarytransit.com/pdf/2007_.pdf, 2007.

[7] R. Hoar and J. Penner. The application of artificial intelligence to transportation system design. *Crossroads*, 9(3):5–9, 2003.

[8] R. Hoar, J. Penner, and C. Jacob. Evolutionary swarm traffic: if ant roads had traffic lights. *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, 2:1910–1915, 2002.

[9] Ricardo Hoar. Multi-agent modeling and analysis of pedestrian and vehicular traffic. Master's thesis, University of Calgary, 2004.

[10] S.D. Maclean and D.J. Dailey. Busview: a graphical transit information system. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 1073–1078, 2001.

[11] Sun Microsystems. Mysql 5.0 reference manual. http://dev.mysql.com/doc/refman/5.0/en/index.html.

[12] Christopher C. Miller. A beast in the field: The google maps mashup as gis/2. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 41:12, 2006.

[13] L.D. Paulson. Building rich web applications with ajax. *Computer*, 38(10):14–17, Oct. 2005.

[14] John C. Sutton. Gis applications in transit planning and operations: A review of current practice, effective applications and challenges in the usa. *Transportation Planning & Technology*, 28(4):p237 – 250, 20050801.

[15] Calgary Transit. About ct (satistics - fleet information). http://calgarytransit.com/html/fleet_information.html.

[16] J.S. Zepeda and S.V. Chapa. From desktop applications towards ajax web applications. *Electrical and Electronics Engineering, 2007. ICEEE 2007. 4th International Conference on*, pages 193–196, Sept. 2007.

[17] Qian Zhen, Lu Huapu, and Liu Chong. Research on urban transit planning and management system based on gis. *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 504–509, Sept. 2005.