A WEB-BASED TOOL FOR SPATIOTEMPORAL FILTERING AND CONTINUOUS ANIMATION

Alex Vakaloudis and Simeon Veloudis
TEI Serron, Terma Magnisias, Serres, Greece

Keywords: GIS GUI, Spatiotemporal navigation, Spatial-temporal filtering.

Abstract: We describe MoveMap, a front-end tool for spatiotemporal databases with moving objects. Built over the Google Maps technology, is independent of any underlying data model or query language. It accommodates continuous temporal navigation and aims to both precision and abstraction by employing the Google Maps DirectionsService utility. For filtering and controlling the display, it includes a set of spatiotemporal operators that can be dynamically triggered, as the navigation proceeds in time. Spatiotemporal querying is thus performed in two different layers; first at the server level which can be accomplished by any underlying framework and second on the client through this mechanism of associating query conditions to browser events.

1 INTRODUCTION

The web and technologies such as Google Maps are increasingly becoming a very interesting medium for the dissemination and processing of geographical information. In this paper we focus on maps with moving objects. This type of maps typically concern location-based services for applications such as surveillance (Hilton, 2006) and transportation and navigation (Wolfson and Bo Xu, 2010).

The importance of web-based maps is underlined by the continuous evolution of products by commercial giants, Bing Maps by Microsoft and Google Maps. The latter comes with a JavaScript API, currently in its 3rd version, which includes modelling of spatial data types such as points, lines and areas, spatial overlays, zooming and geocoding.

Nevertheless, the absence of any built-in provisions for spatiotemporal data types is remarkable and largely motivates the work presented in this paper. A mechanism that demonstrates moving objects on Google maps appears in (Williams, 2010). Although it is not database-driven, it verifies that this technology can form the foundation for an interface to a spatiotemporal database. Google Earth, does includes temporal support in KML via a time slider control which displays discrete transitions in the movement of points or the shape of lines/polygons.

The ArcGIS server provides a web interface with support for time-varying data. Change is discrete and hence it does not cover any continuous movement.

Apart from Google Maps, other efforts on web-based GIS have used applets (Voss and Andrienko and Andrienko and Gatalsky, 2001), Flash (Brannan, et al., 2008) or other plug-ins at the client side. Research on web-based moving objects for GIS includes the work of (du Mouza and Rigaux, 2002) who stresses the importance of continuous queries. Later efforts on Google Maps are Geotracker (Chen, et al., 2007) that visualises RSS events according to their timestamp and Temp-o-map (Kauppinen and Deichstetter and Hyvönen, 2007). However, these efforts deal with objects with discrete temporal characteristics (e.g. World Cup events) and thus do not cater for the animation of moving points.

From a system structure point of view, with the ever-increasing growth in processing power, the thin clients of previous web-based GIS systems are becoming "thicker" and equipped with more services like spatial navigation, data filtering and layering (Khan, 2010) (Horal, et. al., 2006). This provides better interaction with spatiotemporal data through local data manipulation (Hilton, 2006).

The objective of this work is twofold:

1. To control spatiotemporal navigation and display continuous movement of points with the optional use of DirectionsService utilities through

Vakaloudis A. and Veloudis S..
A WEB-BASED TOOL FOR SPATIOTEMPORAL FILTERING AND CONTINUOUS ANIMATION.
DOI: 10.5220/0003335703990403
In *Proceedings of the 7th International Conference on Web Information Systems and Technologies* (WEBIST-2011), pages 399-403
ISBN: 978-989-8425-51-5

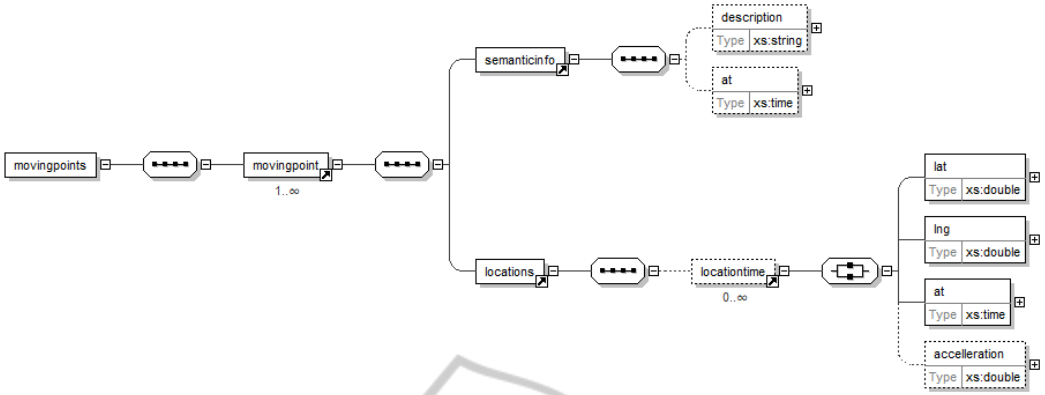

Figure 1: The MoveMap XSD schema for interacting with the application server.

the use of Google Maps as a front-end to a three-tier architecture. In the proposed scheme, the web-based client relays any queries to the underlying database through AJAX calls generically, i.e. independently of any specific spatiotemporal data implementation or query operators.

2. To optimise performance by including at the client-side an engine for spatiotemporal querying that will reduce the number of calls to the server. The client is thus no longer confined to a thin model with display-only capabilities.

To this end we propose MoveMap, a front-end tool for displaying the contents of spatiotemporal databases with moving objects.

2 ACCESS TO DATA MODEL

Over the past fifteen years a proliferation of spatio-temporal data models backed by query operators have been proposed (Guting and Schneider, 2005). Here we do not propose yet another such model; instead, we focus on the visualisation of spatiotemporal data and on controlling their display.

The MoveMap tool lies on the top-tier of a typical 3-tier architecture and communicates with the server/application tier which in turn is responsible for connecting to a database with spatiotemporal data. These data are converted to an XML feed which is subsequently sent to the client. The choice of XML is made for simplicity: we could similarly use JSON or web services to achieve equivalent results or even increase performance. An XSD schema (Fig.1) is thus defined to receive query results. The principal concept is the spatiotemporal point (*locationtime*) consisting of two spatial coordinates and a temporal timestamp. The last element of locationtime is the optional *acceleration*, introduced for the enhanced representation of movement. A *movingpoint* is defined as the composition of spatial and non-spatial evolution, in other words it is made up by a sequence of locationtime elements and a sequence of semanticinfo elements. The latter signifies any application-specific semantics associated to a moving point. Overall, the entire query result is represented by a set of movingpoint elements, the root element *movingpoints*,

3 ARCHITECTURE

Fig. 2 illustrates the architecture of *MoveMap* and its placement within a web-based GIS. We concentrate on the client tier. The display of a Google Map is augmented with two GUIs: one for *temporal controls* and one for the *definition/monitoring of spatiotemporal operators*.

Underlying are two engines implemented as JavaScript APIs: the *Movement Engine* responsible for updating the positions of points in the map and the *Spatiotemporal Query Engine* which controls the contents of the map and the interaction with the user through appropriate notifications.

Any data used are in the form of a JavaScript array of moving points (available to all modules) produced by a *Connector* module that communicates with an Application Server to receive query results.

3.1 Animation of Moving Objects

3.1.1 Temporal Controls

Spatial navigation and spatial zoom are already handled by Google Maps GScaleControl. For temporal navigation, similar to previous approaches, we provide a slider, a clock and temporal navigation

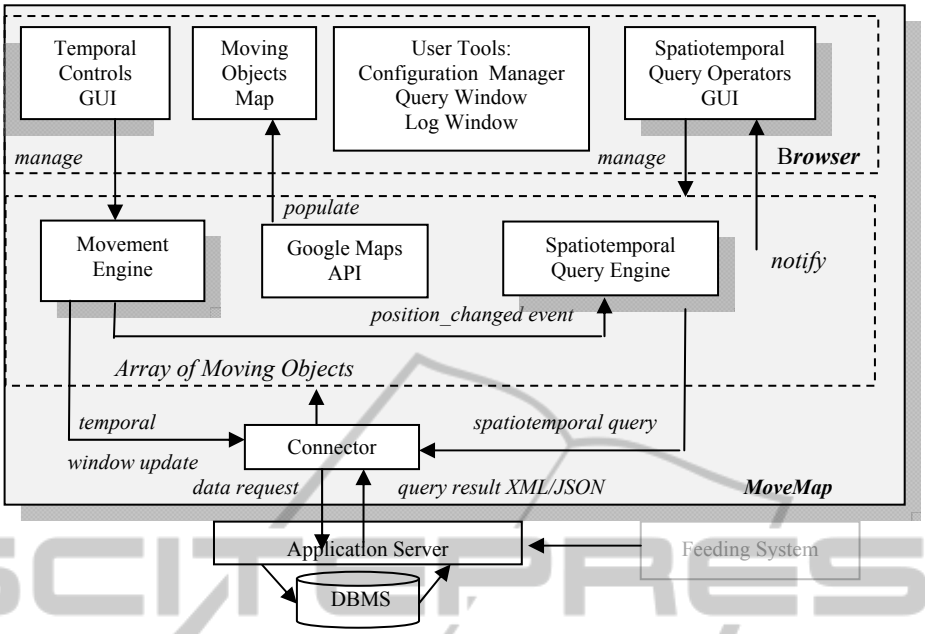

Figure 2: MoveMap Architecture.

controls. MoveMap supports multiple granularities and the choice is made in the configuration manager.

The user queries the database through a textual window. The decision to leave this utility in simple textual format is made because the use of a visual interface would mean reliance on a specific spatiotemporal query extension and loss of generality.

3.1.2 Supported Movement Modes

To achieve a more realistic presentation of the movement and produce more precise calculations it is important not to confine to simply showing objects at the recorded timestamps. For this reason *MoveMap* supports three modes of movement:

a) *Index*: A sequence of points with timestamps are given and objects are displayed when the respective timestamp is reached. Transition from one location to another is discrete. This type of movement is currently supported by Google Earth and has been extensively studied.
b) *Continuous*: A point is shown moving on a straight line between two successive pairs of timestamped coordinates in a manner specified by the acceleration property.
c) *Road based*: As in the continuous mode, but its route relies on Google Maps DirectionsService utility to identify the path followed instead of moving in a (non-realistic) straight line.

The last mode is the most interesting one and has not been used – to the best of our knowledge - by any web-based GIS. It offers simplicity since only two points are required to depict larger spatial variations and accuracy as an object is moving along roads hence correcting any erroneous margins in its recorded position. For objects moving along roads the distances calculated are more realistic than the continuous mode. Yet, since path marking is delegated to Google Maps this approach cannot assure the yielding of the desired route.

3.1.3 Movement Engine

The movement of objects is implemented as a JavaScript timer (*setTimeout()*). A moving point is represented by a Marker and at each clock tick an updated position is calculated through the acceleration property and movement type. Updating a position raises a Marker class *position_changed* event consumed by the query engine.

When the Road-based movement is preferred and the Google Maps DirectionService is called, the results are stored in a cache array for later use. This is essential not only since getting Google Map Directions is a time consuming process but there also exists currently a limit on daily use.

3.2 Spatiotemporal Querying on Client Level

3.2.1 Spatiotemporal Client Query Engine

The Query Engine facilitates querying of moving objects at the client-side. Having an application that does not rely entirely on a server for data manipulation and filtering, increases usability and is nowadays feasible because of increased computer performance and advanced browser scripting. Obviously this does not eliminate the need for getting data from the database. However, the user issues queries less times to a server and uses the client controls to revise the display.

The engine is implemented through DOM events that are fired when certain conditions are met. It takes as input the *position_changed* event specified for the Marker class in the Google Maps API. This is fired every time the *moving engine* changes the position of a moving point as it progresses with temporal navigation. A set of spatiotemporal event listeners process this event to check the satisfaction of the query conditions and notify the user accordingly.

Since spatiotemporal operators are evaluated at every tick of the clock, their satisfaction becomes observable only when the temporal navigation reaches the timestamp of activation. Thus navigation up to a certain timestamp is required to trigger the display of an operator's result.

3.2.2 Spatiotemporal Operators

For assisting the web-user in querying at the client-level we define a set of spatiotemporal listeners to correspond to spatiotemporal conditions. We concentrate on spatiotemporal criteria that have both spatial and temporal conditions to cover spatiotemporal behaviour. This set can obviously be augmented; its purpose is to provide a standard set of querying channels and to demonstrate the feasibility of processing queries at the client-side.

We cover the following spatiotemporal queries:

- snap shot queries (where was I at 3:45) and window queries (between 4:45 and 3:50)
- clustering queries (how many around the city)
- association queries (who was here before me)

3.2.3 New Spatiotemporal Query Operators

Defining a new query operator is a two step-process: First, the user specifies the operator by giving a name and choosing its type. According to this type, additional data are required, for instance for the proximity (circle) operator the radius and position of the circle. The operator may be applicable to a certain time window. Second, in the map display, the user selects the markers applicable to the operator by double clicking on them. Google Maps supports clickable markers and assigns them to the arguments of the operator.

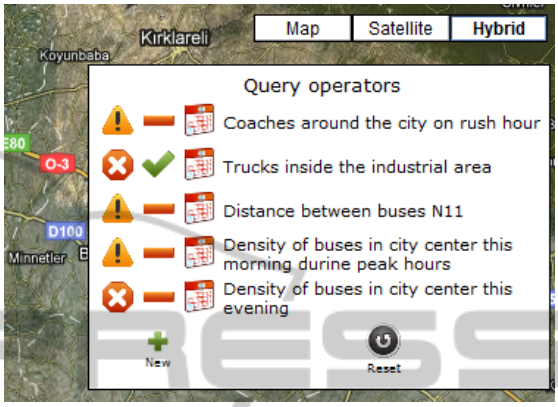

Figure 3: An example of a list of spatiotemporal operators.

3.2.4 Notification Levels

Applying the browser capabilities in conjunction with the special features of a moving objects map, *MoveMap* defines six leve ls of user notifications:

- Simple text notification in the log window.
- The temporal navigation ceases
- JavaScript alert
- Spatiotemporal zoom.
- Change of the status of the marker in the map: Show/hide, flash or highlight the marker.
- Display of Overlay (Circle or Rectangle).

3.3 Communicating with Server

The Connector module is responsible for issuing query commands to the application server, receiving the query results in XML format and producing arrays of moving objects. It is important to design a Connector module to manage the size of the incoming XML and the number of displayed points since a map with a large number of markers may crash or freeze.

The Connector also informs the server of the maximum concurrent number of moving points it can handle. If the number of query results exceeds this limit, they are sorted in time order and are partly returned. If the user's temporal navigation approaches or exceeds the upper timestamp of the partial query result, the Connector informs the application server so as to receive the next patch of

results. Communication is performed by AJAX calls without the need for user intervention.

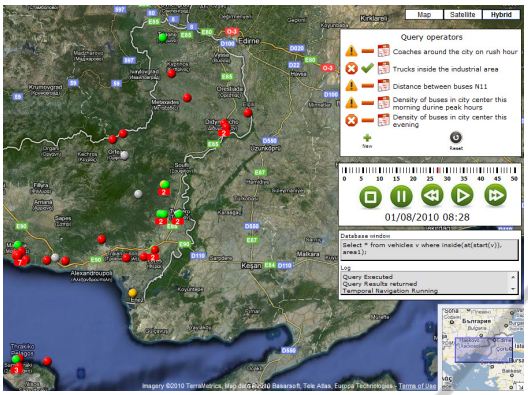

Figure 4: Monitoring the movement of coaches.

4 CONCLUSIONS AND FUTURE WORK

In this paper we presented a Web interface for applications with moving objects lying on Google Maps. It includes continuous DirectionsService-based animation and improves user experience with a set of spatiotemporal operators on the moving objects. These operators are activated on the fly i.e. as the display fluctuates dynamically in time and their inclusion constitutes an effort to make an already easily accessible interface of Google maps more self-sufficient in terms of queries to the application server.

Currently, this system is being used to monitor the operation of coaches in the Thrace region of Greece. Their positions are recorded by staff and imported to the database. The managers check punctuality, speed and distance between coaches as well as track stoppage times.

Future work involves extension of continuous animation to line and polygon data types and the exploitation of the XSD to visualise mutations on non-spatial attributes. Moreover, we investigate spatiotemporal clustering in terms of visualisation and client-based querying.

REFERENCES

Brannan S., Evens N,. Barnett C., Deyneka L., Ising A., Wheaton B. 2008. Web-Based Spatio-Temporal Display of NC DETECT Surveillance Data. *Advances in Disease Surveillance*, 5(1), 6.

Chen Y.F., Di Fabbrizio G., Gibbon D., Jana R., Jora S., Renger B, Wei B. GeoTracker: Geospatial and Temporal RSS Navigation. In *WWW '07 Proceedings of the 16th international conference on World Wide Web,* 2007

Guting R.H and Schneider M. (2005). *Moving Objects Databases*, Morgan Kaufmann

Hilton B.N. 2006. Open Source Software, Web Services, and Internet-based Geographic Information System Development. *CaGIS* 32(4)

Horak J., Unucka J., Stromsky J., Marsik V., Orlik A. 2006. TRANSCAT DSS architecture and modelling services. *Control and Cybernetics* 35(1)

Kauppinen T., Deichstetter C., Hyvönen E., Temp-O-Map: Ontology-based Search and Visualization of Spatio-Temporal Maps. In *ESWC 2007*, Innsbruck, Austria, 2007

Khan Z.A. 2010 Usability Evaluation of Web-based GIS Application. *Master Thesis*. *School of Computing Blekinge Institute of Technology, Sweden*

du Mouza C, Rigaux P, Web Architectures for Scalable Moving Object Servers. In *10th ACM-GIS'02*, Virginia, USA, 2002

Voss H., Andrienko N, Andrienko G, Gatalsky. 2001. Web-based Spatio-Temporal Presentation and Analysis of Thematic Maps. *The Journal of Cities and Regions, Journal of SCORUS, the Standing Committee on Regional and Urban Statistics and Research.* Nov., 51-61

Williams M. (n.d.), *Using the Google directions for an animated drive.* Retrieved September 1, 2010, from *http://econym.org.uk/gmap/example_cartrip.htm*

Wolfson O., Bo Xu B. Spatio-temporal Databases in Urban Transportation. 2010. *IEEE Data Eng. Bull,* 33(2)