ACCELERATOR CONTROL DATA VISUALIZATION WITH GOOGLE MAP

Wenge Fu, Seth Nemesure,
Brookhaven National Laboratory, Upton, NY 11793, USA

Abstract

Visualizing live accelerator control data with geological data on a local map can give main control room operators a convenient way to monitor the accelerator control system and detect whether there is a problem as well as the exact location of the problem. Google's Map API provides an easy way to visually present many kinds of control data into a Google Map with dynamic symbols and animations. This paper describes the details of how live control data visualizations can be implemented for displaying the running beam status, beam loss data, and RHIC complex building temperature data in the AGS/RHIC Control system. Most of the server side and client side code can be easily applied to many similar data visualization situations.

INTRODUCTION

Since its first release in June of 2005[1], the Google Map API has been a fast growing API both in terms of its popularity among web developers and its functionality. The Google Map API provides a convenient and powerful way to integrate Google Maps (a free web map service) and geographically tagged application data on to web pages. The Google Map service and Google Map API works within both an internet and mobile based environment. The API is based on javascript and, when combined with other Google APIs [2] such as Map Image API, Places API, Visualization API, and other third part server side and/or client libraries/APIs (javascript based or other languages based), the map API is a powerful tool for web application developers.

METHODS OF DATA VISUALIZATION

Considering Google Map as a "virtual canvas", any kind of graphic user interface object can be drawn onto the map. The Google Map API also has a dedicated overlay class named canvasLayer for supporting HTML5. The Google Map API provides many ways to allow developers to connect and display data onto a standard or customized Google Map and to add any GUI component onto the Google Map as an overlay to present the application data. It allows users to interact with the data in many different ways.

*. Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy.

# fu@bnl.gov

There are three key steps to accomplish this process:

1. Find the precise geographical location represented by Google Map geolocation data, This geographical location on the map is where you want your data to be displayed and visualized with proper GUI presentation. This step requires a data geographical survey.
2. Know what data you want to display at the desired location. In most cases, this can be a server side job. The data can be either pulled from the web server, or pushed from a back end web server (or application server) through HTTP network connection. Some key steps of data visualization such as data acquiring, data mining, data parsing, data mashup, and data filtering are mostly done in this step.
3. Display the data. The method with which the data is presented on the Map at the target location can be achieved as a static symbol such as an icon, as an animated GUI component(chart, image, video, etc), or as pop-up html-based box with more detailed data.

The Google Map Service uses geolocation data to specify the geographical location on a map. There are many ways to get the geolocation data for web based Google Map applications. The most common methods are:

- With web browsers which support W3C geolocation standard [3]. This can be used to get the users location data. This method only gives a rough estimate of a users location.
- With geolocation sensors (such as a GPS locater) available on a user device (such as mobile phone) to determine geolocation. The geolocation data obtained in this way will be much more accurate.
- Use predefined geolocation data, such as data stored in a database, as json data objects, xml files etc., or via publicly available data resources.

Here we focus on using the predefined geolocation data for the visualization of accelerator control system data and we focus on a small target area (e.g. 20 square mile geographic region) on the Google Map. This allows for the calculation of the geolocation with a reasonably high degree of precision.

The geolocation data used by Google map uses geographic position defined by a latitude [-90(S), 90(N) degrees] and longitude [-180(W), 180(E) degrees] coordinate system.

The Google Map API method for geolocation is expressed as google.maps.LatLng which represents a pair of latitude and longitude coordinates in decimal degrees.

ISBN 978-3-95450-139-7

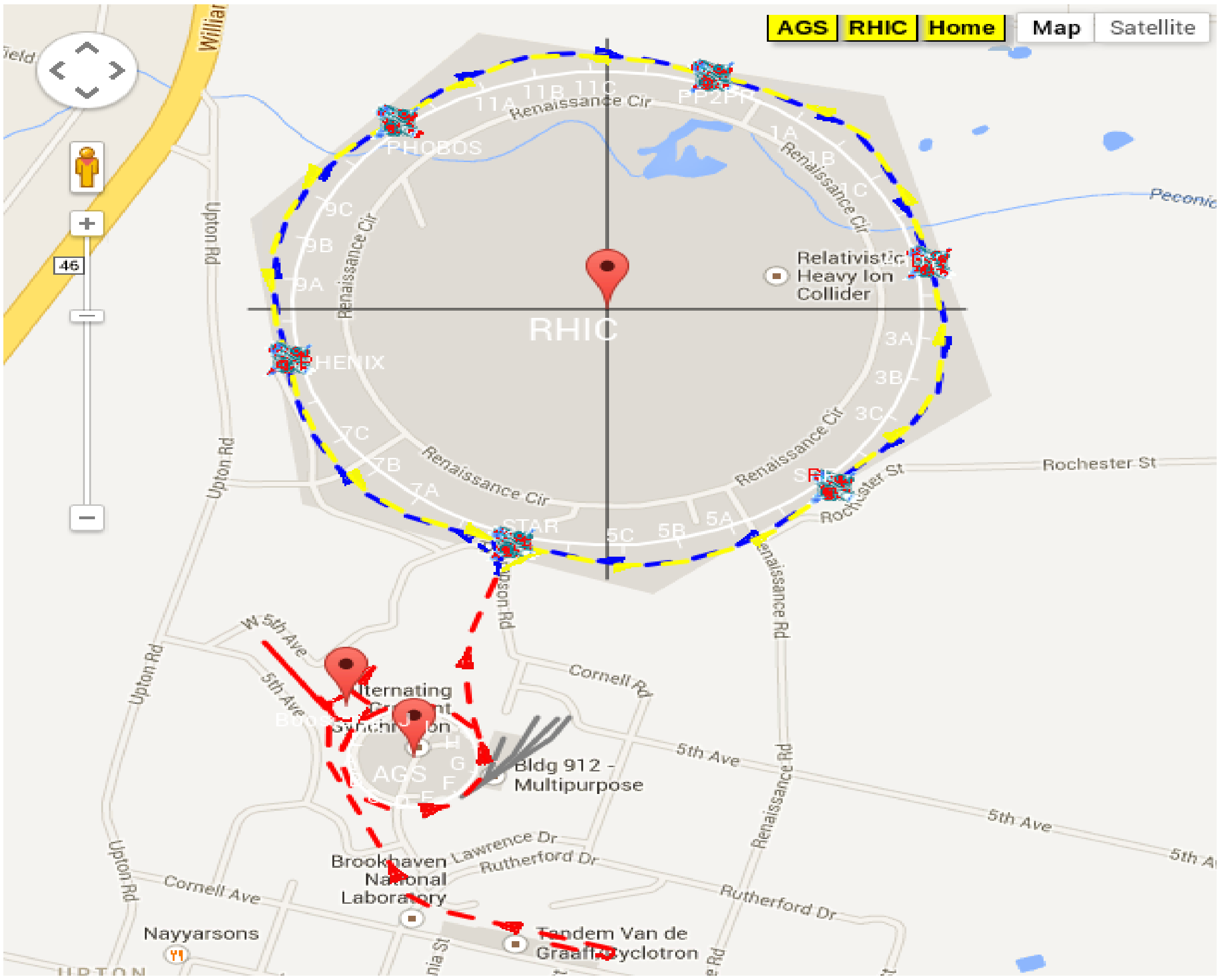


Figure 1: Live running status of RHIC/AGS system on Google Map.

For example, google.maps.LatLng(40.6984703, -73.9514422) in decimal format corresponds to latitude (40° 41' 54.492") and longitude (-73° 57' 5.1912") in degree/minute/second (DMS) format. The conversion between them is based on the formula:

Decimal Degrees =
Degrees + (minutes*60 + seconds)/3600

Because of the non-spheroidal (ellipsoidal) shape of the Earth, there is no accurate way to calculate the distance between two given geolocation points with very high precision. The Google Map API doesn't have a method to calculate intermediate geolocation data from given geographical location data by interpolation or similar mathematical methods. However, when working on a small and flat area, the effect of the non-spheroidal (ellipsoidal) shape of the Earth is insignificant and can be ignored allowing for a reasonably high degree of precision.

When working on a small and flat desired area on a Google Map, a starting reference point is needed for the geolocation data. This can be the latitude and longitude coordines of data at the left-top corner and right-bottom corner of the target area. This reference data will be used to calculate all other geolocation data required by a web application within the target area. This starting reference data for a target area can be easily obtained with the Google Map API method getPosition() which can be triggered from a mouse click event. This is necessary step and is required during the data survey and basic data collection phase of a Google map API based web application. If the length and width of a target area are known and a reference point is defined (i.e. The top-left or bottom-right corner), other bound data can be calculated based on data conversion formula between meter and degree latitude or between meter and degree longitude. In this case, a proper formula with reasonable precision is required and many online tools for this kind of conversion are available[4].

Once the boundary data of the target area is available, the length and width of the target area can be obtained in units of “fractional part of the decimal degree”. This defines the virtual canvas mathematically for the target area. Any subsequent geolocation data can then be

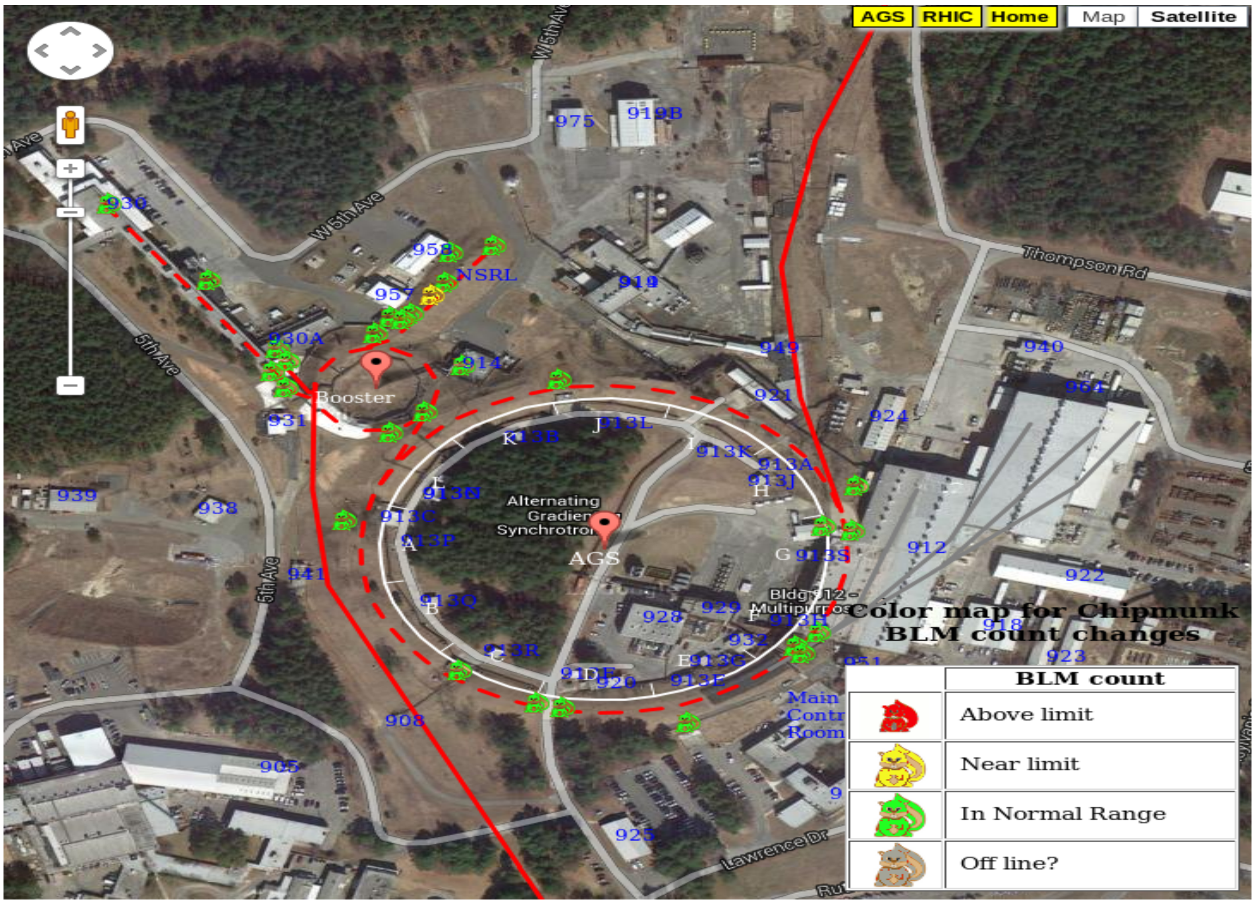


Figure 2: RHIC Chipmunk Beam Loss Monitors on Google Map.

obtained within this target area by using the mathematical geological calculation.

RHIC CONTROL DATA VISUALIZATION

With the strategy described above, a web application was developed to visualize three kinds of RHIC/AGS controls data with three different layers. In addition to the Google Map API, the application also uses YahooUI APIs on the client side for AJAX support and PHP scripts on server side.

Server side PHP scripts respond to client requests and deliver live controls data to clients. This part can be done by either client side data pulling – the method chosen by this implementation -- or by a server side data push. The decision on which method to choose depends on how the HTTP connection is setup.

In this Google Map web application, the basic layer, shared by all other application specific layers, displays all beam sources, beam lines, accelerators, and the collider in the RHIC complex as overlay layers. It also displays building data around the RHIC complex. The details of each object displayed on the map becomes available in either a popup information box or at an appropriate zoom level of the map. This gives viewers a basic and clear view of the layout of the entire RHIC complex and where each major control system component is located.

In the first layer, the running status of the RHIC complex is displayed with an animated beam line (the animation status is controlled by a live status measurement of the RHIC system). The data and animation components in this layer are an indication to viewers of the source of beam currently feeding RHIC. They provide details about which beam lines are active and whether there are collisions at the interaction regions.

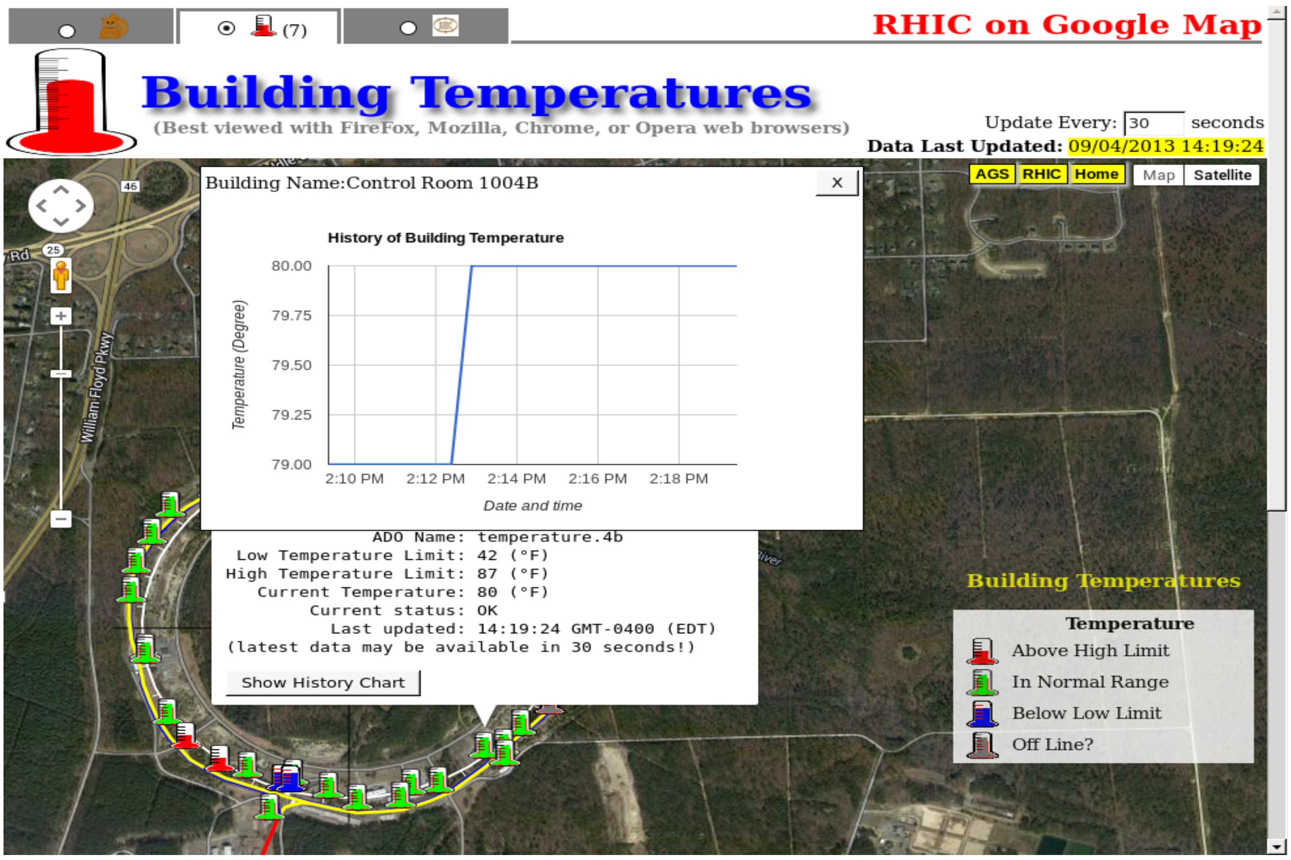
A detailed status for each component can also be displayed by a pop up information box which can be triggered by a Google Map mouse click event.

Figure 1 is a snapshot of the web application containing the first layer.

This web application takes advantage of only some of the basic utilities available for visualizing the accelerator control data. Google Map also supports indoor maps (the next technology frontier) for displaying buildings and floor plans. This allows visualization of indoor accelerator control components.

In the second layer, the RHIC and AGS beam loss data monitored by Chipmunk devices at various locations are visually displayed with color coded animated chipmunk icons. The color and the animated status of an icon corresponds is related to the value reported by each monitor. When a beam loss monitor detects high beam loss, the web application will highlight the monitor in red and animate the chipmunk to draw attention from

ISBN 978-3-95450-139-7



```
           ADO Name: temperature.4b
  Low Temperature Limit: 42 (°F)
 High Temperature Limit: 87 (°F)
    Current Temperature: 80 (°F)
         Current status: OK
           Last updated: 14:19:24 GMT-0400 (EDT)
(latest data may be available in 30 seconds!)
```

Figure 3: Building temperatures in RHIC Complex.

operators. It is also possible to include a sound effect as well. Operators can immediately detect the degree of beam loss and take appropriate action. Figure 2 is a snapshot of this layer. In all cases, the information details may change with the map zoom level. The history of each data changes can also be displayed in a popup chart.

The third layer contains building temperature data monitored by devices at various locations around the RHIC complex. Thermometer icons are used for visualization of temperature. The color and the animation status of an icon corresponds to the live temperature data from each monitored building. When a building temperature is too high or too low the web application will highlight the monitor in red (high temperature) or blue (low temperature). Under these conditions the icon is animated to draw the attention of operators. Operators can immediately identify the location of the temperature alarm and take appropriate action. Figure 3 shows a snapshot of this layer.

In this web application, each layer works independently, but may also be linked together. Tabs at the top of the page (shown as in Figure 3) serves as a global status indicator of showed and hidden layers. This allows users the option to hide a layer while still being able to receive feedback about changes in other layers. For example, while watching the building temperature layer, the program may detect beam loss in a different layer. In this case the tabbed menu icon is animated to draw attention to the affected layer. The operator can select this tabbed layer to obtain details about the alarming loss monitor.

The Google Map API provides an easy way to visually represent many kinds of control system data onto a Google Map with dynamic symbols and rich GUI animations. An easy way to use this technology has been presented in this paper. The web API technology provides the possibility to create many different kinds of live web applications for accelerator control systems. Google's Map API presents an infrastructure for developing accelerator control interactive web applications which may be integrated with traditional standalone applications. The techniques used in this work can be easily applied to many other similar situations.

ACKNOWLEDGEMENT

This work is supported by US Department of Energy.

REFERENCES

[1] Google Site for Developers: https://developers.google.com/maps/

[2] Google API Documentation: https://developers.google.com/apis-explorer/

[3] W3C Geolocation API Specification: http://dev.w3.org/geo/api/spec-source.html

[4] Tools for Google Earth: http://www.earthpoint.us/Convert.aspx