Adding Panoramas to Google Maps Using Ajax

Derek Bradley
Department of Computer Science
University of British Columbia

Abstract

This project is an implementation of an Ajax web application. AJAX is a new technology based on asynchronous communication between a client web browser and a back-end server, allowing web applications to request and receive data without ever reloading the page. A popular application that uses Ajax technology is Google Maps. The goal of this project is to develop an extension of Google Maps which includes viewing 360-degree ground-based panorama images. The project includes development of a client-side Ajax engine and user interface in JavaScript, and development of a back-end java servlet that serves panorama data and images.

1 Introduction

One of the hottest new technologies in web-based application development is Asynchronous JavaScript and XML (Ajax) [4, 7]. Ajax allows web applications to look and feel more like desktop applications because data from a server can be retrieved asynchronously, without reloading the web page. Updates appear almost immediately, which is an improvement over typical web applications which require users to wait for an entire page to reload, even for small changes. Ajax applications are driven by a client-side JavaScript engine that communicates using XML requests with a server in the background.

Currently, one of the biggest users of Ajax technology is Google. Two Ajax applications that Google provides are Google Suggest [6], and Google Maps [5]. Google Suggest is an interface to the Google search engine that dynamically suggests terms as the user types, almost instantly. Google Maps is an address-based global mapping application that allows users to zoom and drag maps around with the cursor without ever reloading the page. The goal of this project is to implement an extension to the Google Maps application, adding ground-based 360-degree panorama views of specific locations in the world. Inspired by previous work in panorama viewing and image-based navigation [2], this project investigates the added benefit of virtually seeing the world from the viewpoint of a specific map location. The ground-based views will add value to the Google Maps functionality, which currently only supports satellite views of the location. Panoramic images have been captured at various locations around the UBC campus. An Ajax application has been developed that incorporates Google Maps with an additional panorama server and a JavaScript front-end, providing the functionality of Google Maps with the addition of 360-degree ground-based panorama viewing.

The remainder of this paper is organized as follows. Section 2 provides background details, describing AJAX and Google Maps. In section 3 the implementation of this project is described. Section 4 shares some experiences that were encountered during development, and conclusions are given in section 5.

2 Background

This section describes Ajax in general and provides a brief overview of the Google Maps web application.

2.1 Asynchronous JavaScript and XML

Ajax is a new technology for web application development. The key idea is that communication between the client and the server happens asynchronously, so users do not have to wait for responses and the application appears to behave more interactively. This idea is demonstrated in Figure 1, where the Ajax model is compared to the typical synchronous web application model.

Ajax is actually several technologies combined together: a backend web server, data transfer using XML, asynchronous retrieval using XMLHttpRequest, dynamic HTML and a JavaScript client-side engine

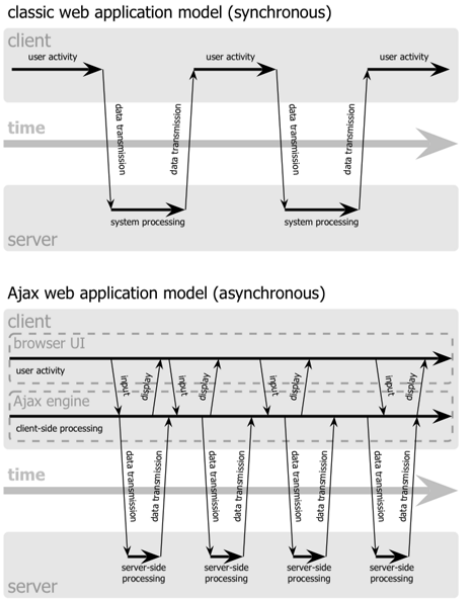

Figure 1. Comparing traditional synchronous web applications (top) with asynchronous Ajax applications (bottom). (Image taken from [4]).

that drives the application. The JavaScript engine (or "Ajax engine") is a new layer that is introduced between the user and the server. At the start of a session, the Ajax engine is loaded. The engine is responsible for rendering the user interface and also for communicating with the server. It is this engine that provides the asynchronous functionality of Ajax. When an XMLHttpRequest is sent to the server it is done so in a non-blocking manner. Control is returned to the user until a callback function is initiated indicating the server's response associated with a given request. The response is then processed by the engine without ever having to reload the webpage.

2.2 Google Maps

Google Maps is one of the most widely known applications of Ajax among current web applications. Google Maps provides an address-based global mapping service that allows users to search for a location anywhere in the world by address, intersection, or keyword. Once a map is loaded users can zoom, pan interactively, select points of interest, get driving directions, and view actual satellite images of the map location without requiring a reload. Map images are loaded on-demand as the user pans with the cursor, showing off the asynchronous Ajax implementation. Figure 2 shows screenshots of Google Maps, located at the Computer Science building of the UBC campus in Vancouver.

Google has made available an API[1] for integrating the Google Maps functionality into other web applications. This project makes use of the API to create an extended Google Maps application that includes ground-based 360-degree panoramic images.

3 Google Panoramas

This section describes the design and implementation details of the panorama extension to the Google Map application.

3.1 System Overview

The panorama web application is designed in typical Ajax style with a JavaScript client-side engine running in a web browser. The JavaScript client is described in section 3.3. In this application, there is not a single back-end server but two servers. The panorama application uses the Google Maps API, so some of the asynchronous communication is performed with the Google server (to provide the standard map functionality). The second server was designed as part of this project, and its role is to serve up the panorama data and image pieces. This component, developed as a java servlet, is described in section 3.4. An overview of the system is illustrated in Figure 3.

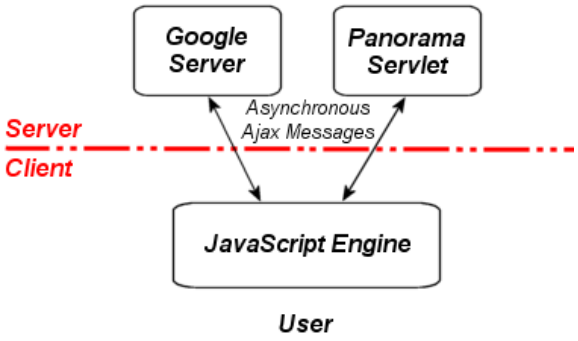

Figure 3. System overview.

3.2 Data Acquisition and Processing

Panorama images were captured from four different locations on the UBC campus. At each of the locations, a series of overlapping images were taken with a regular

[1] http://www.google.com/apis/maps/

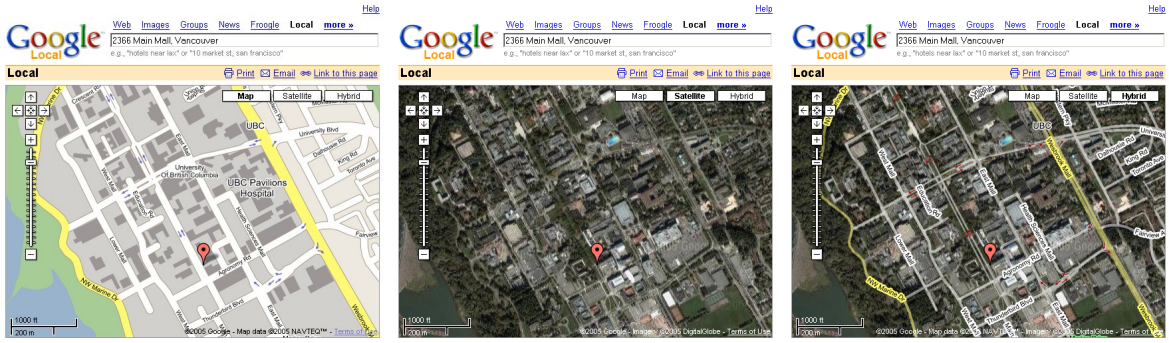

Figure 2. Google Map showing the CS building at UBC. (a) Regular, (b) Satellite view, (c) Hybrid view.

digital camera in order to capture the 360-degree field of view. Approximately 12 images were captured per location, and they were stitched together in a nearly seamless way using the Canon PhotoStitch commercial software. Figure 4 shows the final panoramic images of the four locations. For each location, the Global Positioning System (GPS) coordinates were also recorded using a standard GPS device.

In order to maintain generality, each of the panoramas was resized to 4200 x 400 pixels. A small utility program was then created using the Open Source Computer Vision library [3] to split each panorama into sub-images of size 200 x 200 pixels. These sub-images (or *panorama pieces*) are used in the Ajax web application to load the panorama piece by piece on demand.

3.3 JavaScript Client

The JavaScript client is the Ajax engine. It contains a user interface using dynamic HTML, a function for preparing XMLHttpRequest objects, and callback functions for handling the response events.

On one side of the user interface is the traditional Google map, with basic controls to pan, zoom, and switch views. This part of the interface is built when the page is loaded and the starting map location is the UBC campus by default. The Google Maps API is imported as a single JavaScript file, and the desired map is added to the page using a few simple calls. The API automatically handles Ajax communication with the Google server.

The second component of the interface is a button called "Display Panoramas". When this button is pressed the engine builds an Ajax request and sends it to the panorama servlet, asking for an XML file with the locations and names of all the panoramas. The associated callback function for this request parses the XML response and creates markers on the Google map at the specified GPS locations. The name of each panorama is also associated with the corresponding marker. When the user performs a mouse-over the marker, a small display window appears with the panorama name.

The third part of the user interface is an area of the screen where the panorama images are displayed. When a user clicks on one of the panorama markers on the map, an Ajax message is created to request the first four pieces of the specific panorama. In the callback function for this request, the XML data is parsed and the specific image source locations are loaded into image place-holders on the page. This creates a panorama view window with approximately a 35-degree field of view. Two buttons automatically appear next to the panorama view area, which allow the user to pan forward or backward within the panorama. Each pan request sends another Ajax message asking for the new pieces of the panorama. As a new set of image pieces come in, the engine displays them on the page (removing old pieces that are no longer in the field of view). Figure 5 shows a screenshot of the JavaScript client running in a web browser. The panorama image pieces are loaded asynchronously, so as the user waits for part of the panorama to be loaded the engine is still processing the user's actions. For example, the user can manipulate the Google map in any way while waiting for a panorama image to load, even if the command involves more Ajax communication with the Google server. This benefit is demonstrated with the addition of a "Continuous Pan" button on the user interface. When the user presses this button, the en-

Figure 4. 360-degree panoramic images captured from around UBC campus.

gine simulates a sequence of *right* pan operations, one every 500 milliseconds, until the "Stop Panning" button is pressed. This functionality shows the true asynchronous Ajax nature of the application, because the user is free to manipulate the Google map and even load new panoramas into the viewer, while it never stops panning correctly (and each pan operation involves communication with the servlet).

3.4 Java Servlet

The panorama data and image pieces are stored on a server. The JavaScript client communicates to this server using Ajax messages that are received by a java servlet program. The servlet runs on Apache Tomcat [1], which is an open source servlet container. The servlet operates with the thread-per-request model. When a request is received, the servlet dispatches the request to one of two request handlers, one for serving the basic data for all panoramas (ie: location, name, ID) and one for serving the individual panorama image pieces. The classes of the servlet application are listed in Table 1, along with a brief description of each.

The main panorama data is stored locally on the server in an XML file. This file can be modified at run time to add or remove panoramas from the global list. For the purposes of this project, the panorama servlet was always started on the local host. However, the servlet (and panorama data) should be able to reside on any server in the network or Internet that has a servlet container available.

Class	Description
LocationsRequestHandler	Request handler to serve panorama location, name and IDs.
PanoramaRequestHandler	Request handler to serve panorama image pieces.
PanoServlet	Main servlet that receives and dispatches requests.
Strings	Collection of defined strings.
XMLHandler	Helper class to parse XML data (SAX).
RequestHandler	Interface for the request handlers.

Table 1. Description of the classes in the java servlet application.

4 Experience

This section is to identify the two main problems that were encountered when developing the panorama web application for this project.

The first problem was inconsistencies between different browsers. The application was tested in Mozilla Firefox 1.0.7 and Microsoft Internet Explorer 6.0. Some original prototype image manipulation code in JavaScript did not work in Firefox, yet worked fine in Internet Explorer. In addition, the document layout

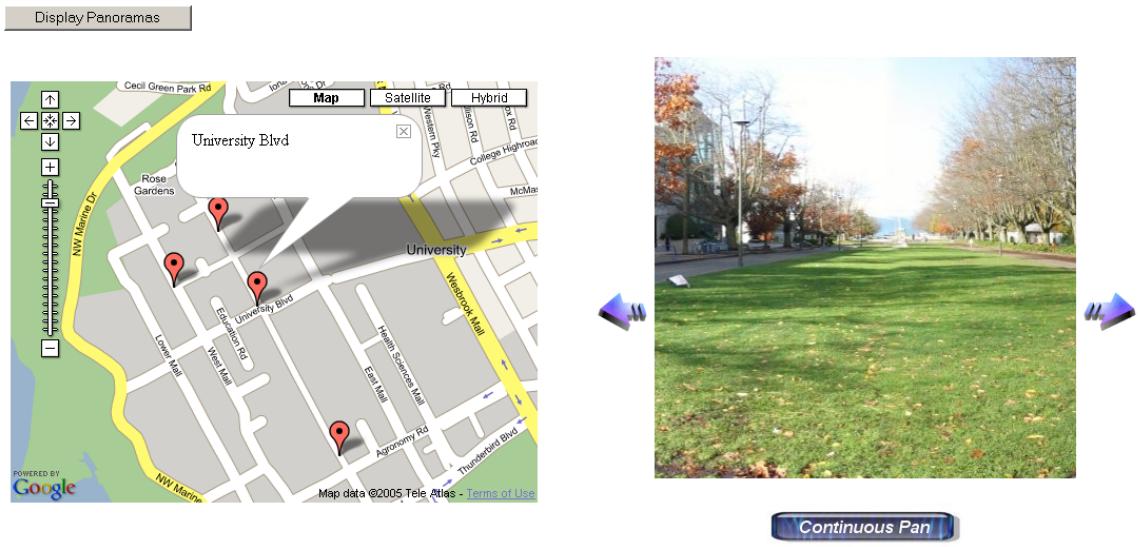

Figure 5. JavaScript client application.

is different between the browsers, and Firefox leaves a horizontal line of white space between the image pieces in the panorama view. However, the response to button clicks is faster in Firefox, and some rapid repeated button presses are lost in Internet Explorer.

The second difficulty in developing this system was the problem of debugging the application. This was the largest challenge encountered. In this web-based environment, experience showed that bugs typically result in a lack of response and no error messages. No debugging tools were employed, so locating sources of errors in the JavaScript code was achieved using old-fashioned output statements through dynamic HTML. Even more challenging was the task of locating bugs in the back-end servlet, because there was no output window to even write messages to. Debug statements from the servlet were compiled into XML responses and returned to the Ajax engine for display. The difficulty in debugging the application added more time to this project than originally anticipated, and unfortunately it also limited the number of features that were added to the final result.

5 Conclusion

This project is an implementation of an Ajax web application. The goal of this project is to extend the popular Google Maps functionality to include ground-based 360-degree panoramic images with an interactive user interface. Asynchronous Ajax communication is used in conjunction with the Google Maps API, a back-end java servlet, and a client-side JavaScript engine to provide a highly responsive web application. The system enables simultaneous map and panorama manipulation without ever reloading the web page. This project is yet another example of how Ajax technology can make web applications appear more like desktop applications, defining why Ajax is currently a hot topic in web design.

References

[1] Apache-Tomcat. http://tomcat.apache.org.

[2] D. Bradley, A. Brunton, M. Fiala, and G. Roth. Image-based navigation in real environments using panoramas. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2005.

[3] G. Bradski. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25(11):120, 122–125, nov 2000.

[4] J. J. Garrett. Ajax: A new approach to web applications (http://www.adaptivepath.com/publications/essays/archives/000385.php), 2005.

[5] GoogleMaps. http://maps.google.com.

[6] GoogleSuggest. http://www.google.com/webhp? complete=1&hl=en.

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