EXPLORING SPATIAL ANALYSIS CAPABILITIES IN GOOGLE MAPS MASHUP USING GOOGLE FUSION TABLES: A CASE STUDY IN LAND LEASE DATA RETRIEVAL

A THESIS PRESENTED TO
THE DEPARTMENT OF HUMANITIES AND SOCIAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

By
MICHAEL WEN

NORTHWEST MISSOURI STATE UNIVERSITY
MARYVILLE, MISSOURI
July, 2014

Exploring Spatial Analysis Capabilities In Google Maps Mashup Using Google Fusion Tables: A Case Study in Land Lease Data Retrieval

Michael Wen

Northwest Missouri State University

THESIS APPROVED

Thesis Advisor, Dr. Yanfen Le	Date
Dr. Patricia Drews	Date
Dr. Naijun Zhou	Date
Dean of Graduate School	Date

Exploring Spatial Analysis Capabilities in Google Maps Mashup Using Google Fusion Tables: A Case Study in Land Lease Data Retrieval

Abstract

This study examines whether a map mashup application created with Google Maps API and Google Fusion Tables API can be used to supplement traditional desktop GIS applications in certain geographical studies. Historically, researchers have used online mapping tools like Google Maps to aid in their geographical research but these online tools have typically played a supporting role; most of the spatial analysis is still done using desktop GIS software. In order to determine whether online tools can handle some of the analytical tasks, a map mashup application that can be used to aid in oil and gas lease management is developed. The application can be used to perform a number of tasks that were traditionally associated with desktop GIS, such as performing attribute and spatial analysis, drawing buffers and polygons, and retrieving data from an external data source.

This study uses Google Maps API and Google Fusion Tables API programmed with JavaScript, a popular interpreted computer programming language that is an integral part of most web browsers, to develop a map mashup application. The application has the ability to display polygons as a feature layer, to draw polygons to delineate the boundaries of study areas, to store the location and attributes of sample points selected from a map, to draw buffers around point features, and to perform spatial and attribute queries from a table of spatial records. An accuracy assessment performed on the external data retrieving capability shows an accuracy of 100 percent. The application met all of the functional requirements and successfully demonstrated the utility of map mashup applications in GIScience research. Among some of the major impediments for more widespread adoption of map mashup applications are legal restrictions Google placed on commercial use and several technical limitations enforced by Google to reduce server load.

Table of Contents

List of Figures

Chapter 1: Introduction

The Internet and mobile devices that connect to it have become ubiquitous features of contemporary life. It is therefore widely recognized that future developments in Geographical Information Systems (GIS) will center on Internet-centered GIS technology that uses the Internet as the primary means to access data, conduct spatial analysis, and provide location-based services (Peng and Zhang, 2004; Zhang and Tsou, 2009). This trend is driven in part by map mashups which were made possible by two developments. First, a number of popular online map providers like Google Maps, Yahoo! Maps, Microsoft Bing Maps, ESRI ArcGIS Online and MapQuest made an enormous amount of online maps and other spatial data available to the public at little or no cost. Second, the Application Programmable Interface (API) offered by these map service providers and other organizations online allows developers to retrieve and manipulate data over the web (Li and Gong, 2008).

On the web the term mashup means taking information published from multiple sources and integrating it into a new information stream (Gong, 2007). Mashups can be created by developers, professional programmers or amateurs using different technologies like eXtensible Markup Language (XML), web services, Real Simple Syndication (RSS), screen scrapping and Asynchronous JavaScript and XML (Ajax) (Li and Gong, 2008). Map mashups are a special type of mashup that combines at least one map data source with information from another source to create a new map (Gong, 2007). A typical map mashup uses a three level architecture: the first level contains data sources, the second level handles the business logic and the last level implements the user interface (Gong, 2007).   At the time of writing, ProgrammableWeb, a premier reference site for mashups, lists over 7,000 known mashup sites (ProgrammableWeb, 2013).

Despite their popularity among web developers relatively little was written to document the contribution or the potential role of map APIs and mashups in the development of Internet GIS applications (Chow, 2008). Most of the existing map mashups were created by using map APIs. A map API is a source code interface published by map service providers that grants web developers access to a program library and to request services in generating maps over the Internet (Chow, 2008). In a typical but relatively simple map mashup the map API is used to modify the content displayed on the map while data from other sources are retrieved from sites like Craigslist by using their APIs, by custom applications that extract pertinent information from web pages or with other technology (Miller, 2005). In addition to map APIs other types of APIs provide functions like querying, returning and updating data at an external source, geocoding based on addresses, and authenticating users (Chow, 2008; GeoCommunicator, 2013; Li and Gong, 2008).

Despite their versatility online maps and most existing map mashups do not really provide the same level of support of spatial analysis that we enjoy from a desktop GIS (Peng and Zhang, 2004; Chow, 2008; Batty, 2010). However Google Fusion Tables, a new tool released by Google in 2009, has the potential to overcome these deficiencies. It is an experimental application that lets users store, share, query, and display data tables. It offers the ability to manage data rows online and query the table for all rows that match spatial or data conditions through an API (Getting Started, 2013). It is a tool that has great potential to provide many online spatial analysis capabilities because it is designed to work in conjunction with Google Maps. There is no comparable product on the market that is as widely available and free to use. This tool is relatively new, having been around for only four years and so there have not been many formal studies done to apply its capabilities to GIScience. It is unique because in addition to offering database and spreadsheet like capabilities online, it also provides a number of spatial functions. With code written to manipulate these capabilities through the Google Maps API and Google Fusion Tables API, it should be possible to replicate in a Google Maps mashup some of the functions currently found only in desktop GIS applications.

The Public Land Survey System (PLSS) is a survey system used in most of the United States. Most of the lands in the public domain in the U. S. are subject to subdivision by this rectangular system of surveys, which is regulated by the U.S. Department of the Interior, Bureau of Land Management. It is used in the continental U.S. except for areas covered by the original 13 colonies and Texas (National Atlas of the United States, 2013b). The PLSS typically divides land into 6-mile-square townships, and townships are then subdivided into 36 one-mile- square sections. Sections can be further subdivided into quarter sections, quarter-quarter sections, or irregular government lots. A legal description of a section includes the State, Principal Meridian name, Township and Range designations with directions, and the section number. Courthouses in states that use PLSS use it to record land leases, and it is the survey system by which oil and gas companies manage their lease holdings (National Atlas of the United States, 2013b).

This study proposes a method of taking advantage of some of the newer capabilities of Google Fusion Tables to perform spatial and attribute queries through the use of the API for Google Maps and Google Fusion Tables. These capabilities are exclusive to Google Fusion Tables as of now. For example, none of the other online map service providers including Yahoo or MapQuest offer data manipulation functionality common to spreadsheet applications like search and sort that is easily accomplished in Google Fusion Tables. The goal of this study is to demonstrate that many of the spatial and attribute query capabilities that are previously available only in desktop GIS software can now be made available online for free through the use of these technologies. It is not the goal of this study to develop a commercially viable application, but to demonstrate the potential of these capabilities, so some of the real life concerns in software development like licensing and legal issues are not addressed. A study like this one can also expose limitations of these tools and associated technology in their current state.

1.1 Research Objective

The objective of this study is to determine whether a map mashup created with Google Maps API and Google Fusion Tables API can be used to produce accurate legal descriptions for land leases in PLSS. In particular, it seeks to find if Google Fusion Tables used in conjunction with Google Maps API can be used to perform some of the commonly used attribute and spatial queries and analysis on the web that currently require the data to be downloaded to desktop GIS. Additionally, this study also explores other common GIS capabilities, including the ability to draw buffers around point features, opening popup windows displaying attributes of clicked geographical location, retrieving data from external sources based on selected locations, and making attribute and distance queries on selected features.

1.2 Study Area

The Uinta Basin and Mountains are located in the northeast corner of the state of Utah (Figure 1) and are part of a larger physiographic area known as the Colorado Plateau Province. The Uinta Basin lies south of the Uinta Mountains. It is about a two-and-a-half hour drive east of Salt Lake City. The southern rim of the basin is formed by the Tavaputs Plateau of the Book Cliffs, and the western rim is formed by the Wasatch Mountains. The central portion of the basin has an elevation of 5,000 to 5,500 feet. Two major rivers intersect this area; the Green River flows southward out of the Uinta Mountains to the north, crossing the Uintah Basin while the Colorado River crosses the eastern portion of this section, cutting off an area of some 40 miles in diameter. This area was selected because it has seen a great deal of oil and gas related activities in recent years (Fuller, 2013).

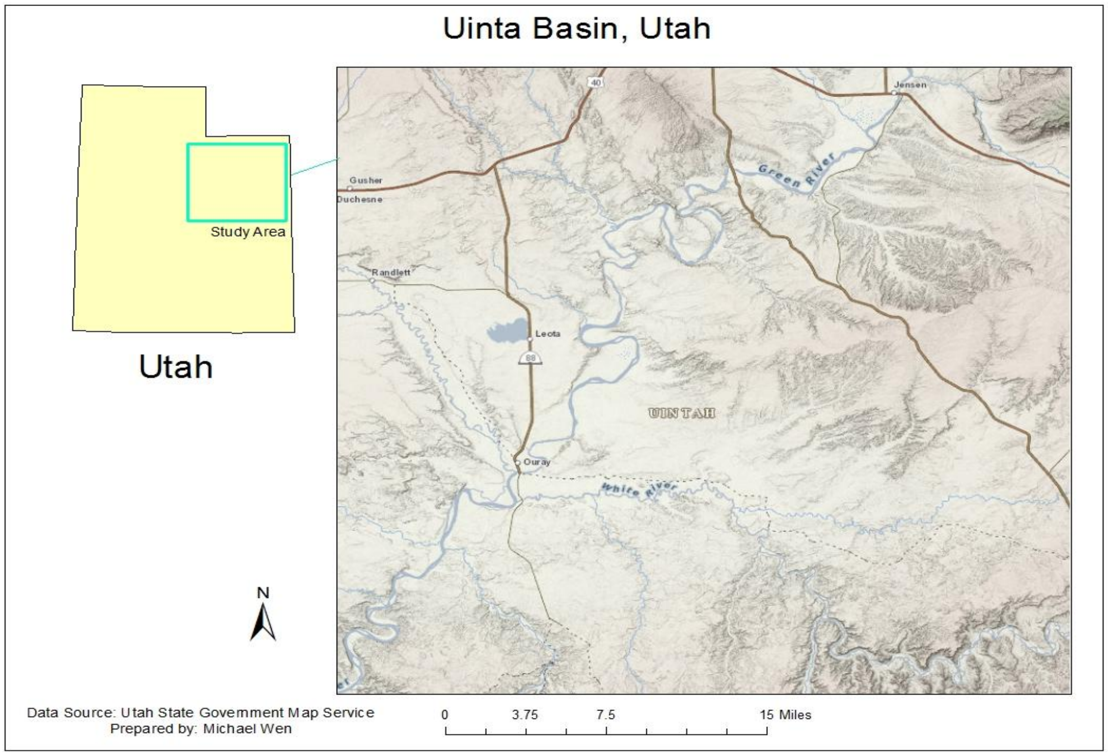


Figure 1 - Study Area

Chapter 2: Literature Review

Since the advent of map mashup applications in 2004, this technology was recognized as having the potential of playing a pivotal role in supporting new developments in geography such as Public Participation GIS (PPGIS) and locational aware applications (Batty et al., 2010; Newman et al., 2010). It was quickly recognized that applications built with this technology allow users with limited mapping knowledge to use GIS-like functions for a wide range of applications (Polczynski and Polczynski, 2013). However online maps and most existing map mashup applications do not yet contain the ability to perform common spatial operations like buffering, spatial querying, geoprocessing or performing map algebra. These features are not currently supported in the map APIs offered by many of the free major map service providers like Yahoo Maps and Google Maps. This deficiency has hampered the adoption of online maps in GIScience studies. For example, existing online maps lack the ability to produce random statistical samples, and currently such samples can only be created by first downloading the entire dataset to the desktop first (Wampler et al., 2013). Studies have been done that use various workarounds to take advantage of the open and freely available spatial content provided by map mashups while utilizing the advanced spatial analysis capabilities of traditional desktop GIS software. Many of the techniques focused on finding ways to export data from map mashups to local hard drives in order to perform spatial and attribute analysis with desktop GIS and spreadsheet applications. The most common approaches involved exporting data to files in Keyhole Markup Language (KML) or Comma-Separated Values (CSV) formats (Chow, 2008; Wampler et al., 2013). Some studies even adopted the approach of constructing maps by writing new KML from scratch which while low in cost is understandably a training intensive approach (Polczynski and Polczynski, 2013).

Other studies were done that sought to find ways to perform spatial analysis on web applications without using desktop GIS. Peng and Zhang (2004) proposed a strategy to use Geographic Markup Language (GML) as a coding and data transporting mechanism to achieve data interoperability, Scalable Vector Graphics (SVG) to display GML data on the web, and Web Feature Service (WFS) as a data query mechanism to access and retrieve data at the feature level in real time on the web. While these technologies are not strictly speaking used for spatial analysis, they can be used for representational storage and transfer. This approach essentially uses these standards to duplicate the standard geoprocessing functionality of a desktop GIS. Chow (2008) took the GML-centric approach further by proposing a conceptual model that consists of three major steps: (1) convert the GIS database into GML or any web-compatible raster imagery; (2) query the spatial data by parsing the GML data or loading the web-compatible raster imagery; and (3) overlay the spatial data into corresponding map API classes for visualization. These approaches will no doubt play a much bigger role in Internet GIS a decade from now compared to the modest techniques used in this study but the amount of programming involved in developing these applications is much greater.

2.1 Google Maps and Google Maps API

Originally a C++ program designed by two Danish brothers Lars and Jens Rasmussen at the Sydney-based company Where 2 Technologies, Google Maps was acquired by Google, Inc. in 2004 and transformed into a web application (LeMay, 2005). The application is based on a close variant of the Mercator projection, but the coordinates of the features on Google Maps are GPS coordinates based on the WGS 84 datum (Map Types, 2014). It suffers from limitations of the Mercator projection, such as not being able to show the poles. Instead it cuts off coverage at 85.051125° north and south which is $\operatorname{atan}(\sinh(\pi))\times 180/\pi$, a requirement that the map is a square (Map Types, 2014).

Google launched Google Maps API in 2005 in order to allow web developers to integrate Google Maps into their websites (Taylor, 2005). It has since become the most heavily used web application development API on the Internet, used in the development of over 1,000,000 web sites (ProgrammableWeb, 2013). At the time of writing it is a free service and does not require advertisements, but the terms of use state that Google reserves the right to display ads in the future. It is also free for some commercial usage, provided that the site on which it is being used is publicly accessible, does not charge for access, and is not generating more than 25,000 map requests a day (Google Maps/Google Earth APIs Terms of Service, 2014). Sites that do not meet these requirements can purchase the Google Maps API for Business, which allows up to 100,000 map requests a day and also comes with technical support, a higher maximum resolution and an advanced analytics tool that enables you to see how visitors interact with your maps (Google Maps for Business, 2014).

The workflow involved in embedding Google Maps inside a web page involves the following six steps (Getting Started with JavaScript, 2014):

1. Declare the application as HTML5 using the <!DOCTYPE html> declaration.
2. Include the Maps API JavaScript using a script tag.
3. Create a div element named "map-canvas" to hold the Map.
4. Create a JavaScript object literal to hold a number of map properties.

5. Create a JavaScript "map" object, passing it the div element and the map properties.

6. Use an event listener to load the map after the page has loaded.

None of these steps require the use of any professional software development applications. A simple text editor like Microsoft Notepad is more than adequate for this purpose.

2.2 Google Fusion Tables API

The Google Fusion Tables API 1.0 is an API published by Google that allows developers to create and manage Fusion Tables resources such as tables, columns, rows, templates, and styles. It uses the Representational state transfer (REST) architectural style that uses the four HTTP methods GET, POST, PUT and DELETE to execute different operations. Unlike Simple Object Access Protocol (SOAP) based web services which try to model the exchange between client and server as calls to objects, REST tries to be faithful to the web domain by keeping the calls in the Uniform Resource Locator (URL) format, the one that is also used for web addresses (Getting Started, 2013).

Through the Google Fusion Tables API developers can use SQL statements to query and manipulate rows and work with tables, columns, styles, and templates (Getting Started, 2013). Security for the data in Fusion Tables is provided by an authentication scheme that requires requests sent to the Fusion Tables API to be accompanied by a unique identifier which can be either an API key string or an access token. The API allows the response for a successful call to be either JavaScript Object Notation (JSON) typed data, which is the default, or an untyped CSV or JSON (typed or untyped) data if using SQL to query rows. A typed data field returns its value in a specific predefined format depending on the content of the field. For example, latitudes and longitudes are returned as two numeric values with eight decimal places while KML data are returned in the format specified by the GeoJSon specification (Butler et al., 2008). Like the Google Maps API the bulk of the programming tasks involves parsing and sending the API requests, followed by the waiting for, receipt and then the processing of the response.

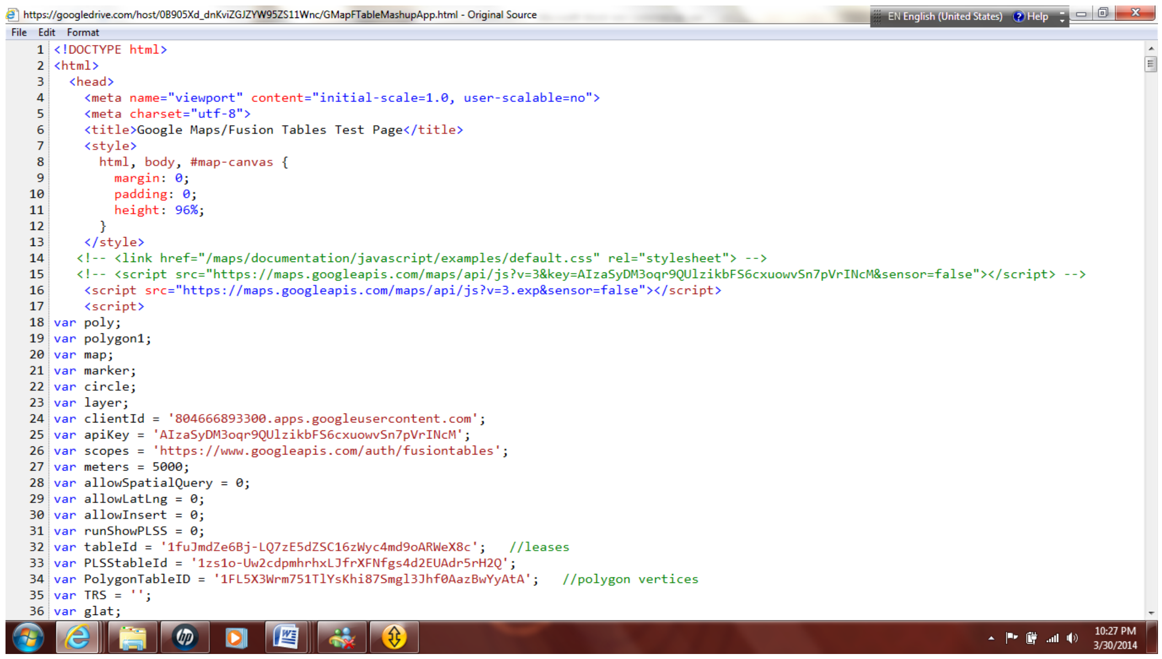
Visualizing the content of resources in Google Fusion Tables as a feature layer on a Google Map involves the use of the FusionTablesLayer object. A FusionTablesLayer object can be created in JavaScript by using the table's encrypted ID in a line of code **(**Fusion Table Layers (Experimental), 2014**)**. The code segment below is used to create a FusionTablesLayer on a Google Map application based on the content of an entire Fusion Table. In this example '1mZ53Z70NsChnBMm-qEYmSDOvLXgrreLTkQUvvg' is the Fusion Table’s encrypted ID.

```
var layer = new google.maps.FusionTablesLayer({
  query: {
    select: 'Geocodable address',
    from: '1mZ53Z70NsChnBMm-qEYmSDOvLXgrreLTkQUvvg'
  },
```

2.3 JavaScript and APIs

Both Google Maps and Google Fusion Tables have an API based on JavaScript, an interpreted computer programming language that come as a part of most web browsers, the implementation of which allows client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed (JavaScript Introduction, 2014). This language offers some flexibility in where the program code can reside, and the simplest approach is to place the code directly on the web page itself. This approach has the additional benefit of allowing everyone with access to the web page the ability to view the source code through the use of the “View

Source” function of the browser. As shown in Figure 2 the JavaScript code was entered directly into the body of the HTML page. No specialized software development environment was needed for this study.


https://googledrive.com/host/0B905Xd_dnKviZGJZYW95ZS11Wnc/GMapFTableMashupApp.html - Original Source
EN English (United States)
Help
File
Edit
Format
10:27 PM
3/30/2014

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
    <meta charset="utf-8">
    <title>Google Maps/Fusion Tables Test Page</title>
    <style>
      html, body, #map-canvas {
        margin: 0;
        padding: 0;
        height: 96%;
      }
    </style>
   <!-- <link href="/maps/documentation/javascript/examples/default.css" rel="stylesheet"> -->
   <!-- <script src="https://maps.googleapis.com/maps/api/js?v=3&key=AIzaSyDM3oqr9QUlzikbFS6cxuowvSn7pVrINcM&sensor=false"></script> -->
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>
    <script>
var poly;
var polygon1;
var map;
var marker;
var circle;
var layer;
var clientId = '804666893300.apps.googleusercontent.com';
var apiKey = 'AIzaSyDM3oqr9QUlzikbFS6cxuowvSn7pVrINcM';
var scopes = 'https://www.googleapis.com/auth/fusiontables';
var meters = 5000;
var allowSpatialQuery = 0;
var allowLatLng = 0;
var allowInsert = 0;
var runShowPLSS = 0;
var tableId = '1fuJmdZe6Bj-LQ7zE5dZSC16zWyc4md9oARWeX8c';   //leases
var PLSStableId = '1zs1o-Uw2cdpmhrhxLJfrXFNfgs4d2EUAdr5rH2Q';
var PolygonTableID = '1FL5X3Wrm751TlYsKhi87Smgl3Jhf0AazBwYyAtA';   //polygon vertices
var TRS = '';
var glat;
```

Figure 2 - JavaScript code in HTML

2.4 Web Services

One of the many ways to provide communications between data providers and data consumers is through the use of web services. The World Wide Web Consortium (W3C) defines a web service as “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically the Web Service Definition Language (WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.” (Booth, 2004, #1.4). Aside from returning data web services can also be used for other tasks, such as conducting backend business transactions, performing analytical function, and authenticating users and IP addresses. Basically, many of functions currently done by desktop software can be performed by web services. One of the potential concerns involved in using web services is security. Because of the potential risks posed by malicious code, many of the top web browsers on the market do not permit JavaScript calls to be made to web services that reside on a network domain other than the one the JavaScript originates from. This security feature must be dealt with by the use of Cross-Origin Resource Sharing (CORS) mechanisms which allows developers to define ways to allow or disallow such the cross-origin requests (Geller, 2013).

The web service used by this study, the TownshipGeocoder service, is a JSON web service (TownshipGeocoder, 2014). It is provided for free by the Department of the Interior – Bureau of Land Management (BLM) and uses the government's PLSS data in NAD83 as the source. The application developed for this study sends a lat/long pair to this web service to retrieve a township/range/section value, but the web service is also capable of returning lat/longs of a specified township/range/section.

Chapter 3: Conceptual Framework and Methodology

The Google Fusion Tables API allows developers to use HTTP requests to programmatically perform common tasks available in the Google Fusion Tables application (Getting Started, 2013). Through this API a program can perform routine tasks like updating a table's data and the visualization of the data, adding and populating new columns for a table, reading a public table's metadata, or querying a table’s content. It also provides the ability to perform spatial queries to find features located within a certain distance or inside a bounding box, and this study explores this capability. The mashup application developed for this study uses the Google Fusion Tables API to perform data manipulation tasks like attribute and spatial queries, while at the same time it uses the Google Maps API to customize the visualization of resulting records in Google Maps.

3.1 Description of Data

The township/range/section legal description polygons used for overlay and for accuracy assessment are obtained from the GeoCommunicator website operated by the Bureau of Land Management (GeoCommunicator, 2013). They are downloaded as shapefiles and loaded into ArcMap for data validation. The data is first trimmed to remove records outside of the study area and then exported to KMZ using ArcMap. The KMZ file is converted to KML format and uploaded to a Fusion Table, which is then used to create the feature layer on the map mashup by creating a FusionTablesLayer on that table.

The list of existing oil gas leases in the study area used in the spatial and attribute section of the study was downloaded from the Bureau of Land Management’s Land &

Mineral Legacy Rehost 2000 System (BLM, 2014). It is downloaded in shapefile format. To load the data into the map mashup, the shapefile is converted to KMZ format in ArcMap and then converted to KML. The KML is then uploaded to a new Fusion Table. Since this dataset does not contain an expiration date column, one is created in Google Fusion Table and test values are then entered randomly into the table to enable searches by expiration date. The reason this is done is to allow the application to perform a search on the expiration date value because that is one of the most common attribute searches in land lease management based on the author’s experience in the oil and gas industry.

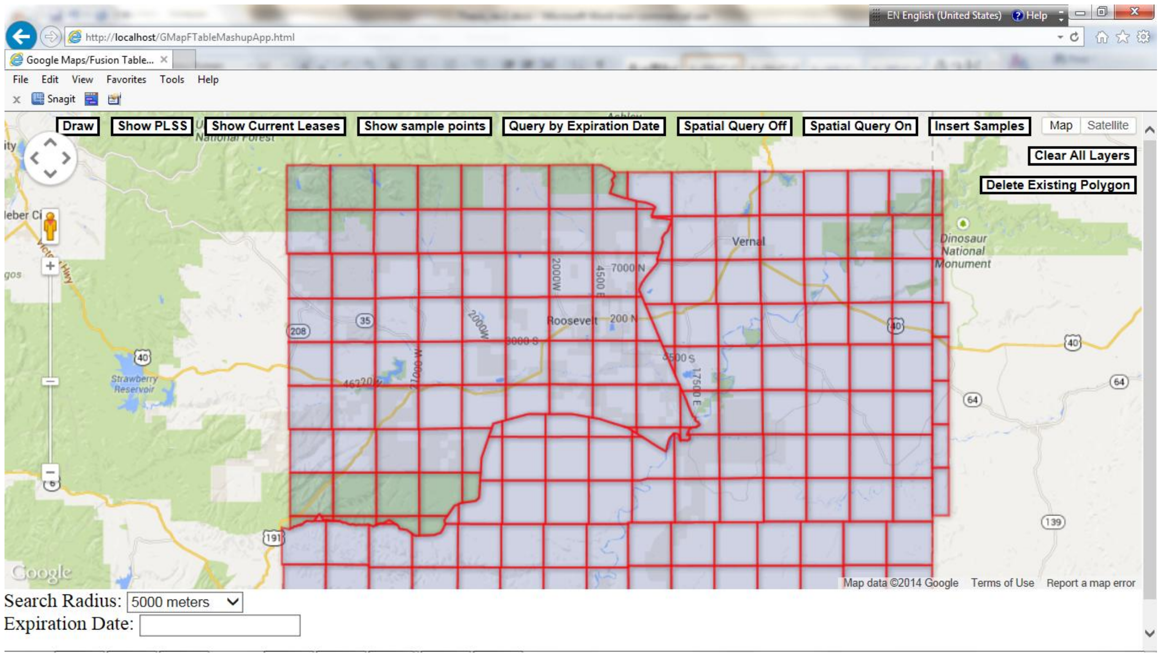
The web service used by the application to retrieve the township/range/section legal description is the TownshipGeocoder service hosted by the Geocommunicator website (TownshipGeocoder, 2014). This web service returns data in XML format, and the desired township/range/section information was parsed out of the XML by a specially written JavaScript function. All of the other spatial data and services needed for this study are hosted by Google, including the base map that appears on the application’s background and both of Google Fusion Tables and Google Maps and their associated APIs.

3.2 Methodology

This study seeks to demonstrate that a map mashup application made with Google Maps API and Google Fusion Tables API can be used to perform some of the tasks and operations that are more commonly done today with desktop GIS and spreadsheet applications. To that end a Google Maps mashup application was developed in JavaScript using Google Maps API and Google Fusion Tables API that satisfies the requirements specified in the following sections.

3.2.1. Display Feature Layer

Since one of the most basic and commonly used functions of desktop GIS is to display a feature layer on top of a base map the first requirement of the mashup application developed for this study is that it should have the ability to display a layer of PLSS township polygons as a feature layer over a base map. This is accomplished by downloading a shapefile containing township/range/section legal description polygons from BLM's website, remove data from outside of the area of interest, convert the shapefile to KML and then load the KML to a Fusion Table. Once the data is loaded into a Fusion Table it can be displayed on the map mashup application through JavaScript by creating a FusionTablesLayer object. After adjusting the opacity of the FusionTablesLayer, the desired effect of imposing the township/range/section polygons on top of the base Google maps is achieved as seen in Figure 3. Users can bring up the PLSS layer by clicking on the "Show PLSS" button at the top.

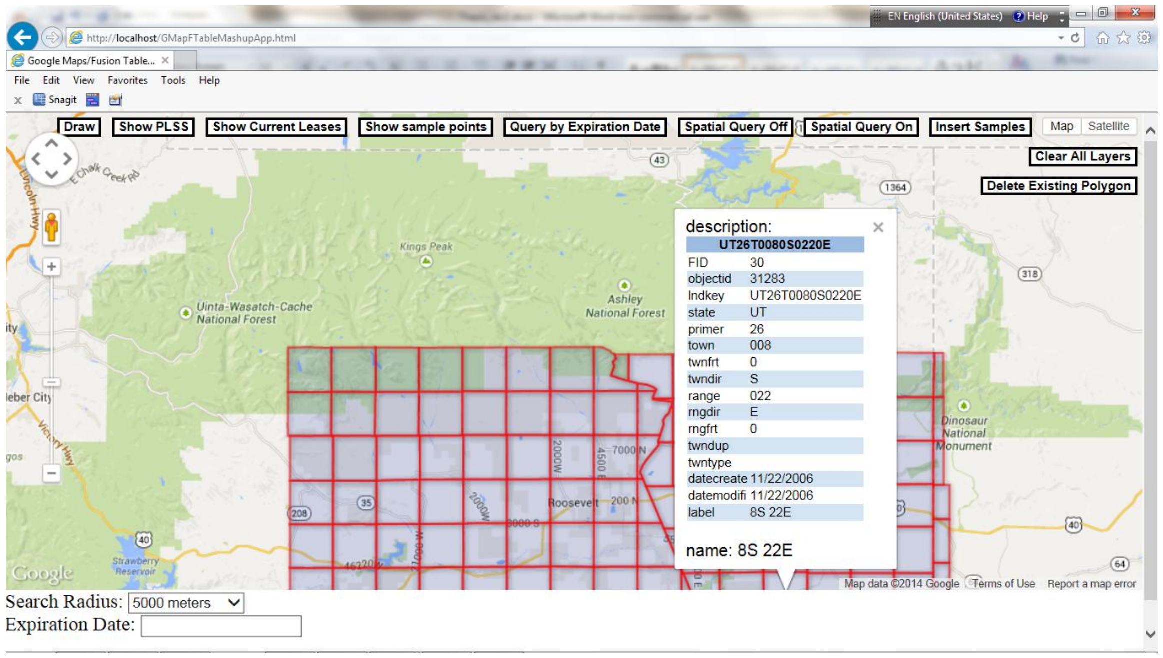


Search Radius: 5000 meters
Expiration Date:

Figure 3 - Legal description overlay

3.2.2. Identify Feature Function

It is relatively simple to display a polygon layer on a Google Maps mashup by creating a FusionTablesLayer, but unfortunately, it is not quite as easy to show labels on each of the polygon features (Issue 97, 2010). Steps like creating another layer just for labels based on an attributes in the Fusion Table may be necessary to produce the desired labeling effect, but this may require too much time to complete. Fortunately, features displayed by using FusionTableslayer contain limited inherent interactivity where the user can bring up a display window showing the values of all attribute columns in the Fusion Table of a selected record simply by clicking on it as shown in Figure 4. This feature works in the same way as the “Identify” function in ArcMap and is a convenient way for users to find all attributes associated with features on a map.



description:

UT26T0080S0220E	
FID	30
objectid	31283
lndkey	UT26T0080S0220E
state	UT
primer	26
town	008
twnfrt	0
twndir	S
range	022
rngdir	E
rngfrt	0
twndup	
twntype	
datecreate	11/22/2006
datemodifi	11/22/2006
label	8S 22E

name: 8S 22E

Search Radius: 5000 meters

Expiration Date:

Figure 4 – “Identify” feature popup window

3.2.3. Drawing Polygons

It is fairly simple to import and export spatial data from desktop GIS to a Google Maps mashup application, and some of the past studies have taken advantage of this technique to move polygons drawn in ArcMap into an online map mashup application. Wampler et al. (2013) took this approach to display the boundaries of their areas of interest in Google Earth. They loaded polygons of the areas of interest drawn in ArcMap and moved them to Google Earth in order to take advantage of the latter's recent high resolution imagery so as to help them identify clusters of dwellings in their study, but we can easily imagine other reasons where areas of interest need to be uploaded to an online map. For example, an oil company land management team can use this tool to delineate areas where potential new leases can be acquired.

The current approach of creating newly drawn polygons in desktop GIS to indicate areas of interest and then exporting these polygons to an online map are numerous and time consuming. The steps involved are as follows:

1. Create a new feature class in ArcCatalog to store the new polygons by creating a polygon shapefile.
2. Create a new map in ArcMap and load the necessary base map that shows the study area, and then load the shapefile created in step 1.
3. Draw the polygons in ArcMap by starting an edit session and then use the "Construction Tools" tool. Figure 5 shows the polygons as drawn in ArcMap.
4. Export the new polygons to KML format by using the "Layer to KML" tool.
5. Unzip the KMZ file produced in step 4 to extract the KML file with an unzipping utility.

6. Load the KML file to the online map. This can be done by simply uploading the KML file to Google Drives, or by loading it to a Google Fusion Table and then use JavaScript to create a new FusionTablesLayer. It was decided that the first approach is more suitable for the purpose of this study. Figure 6 shows what the polygons look like once uploaded onto Google Maps.

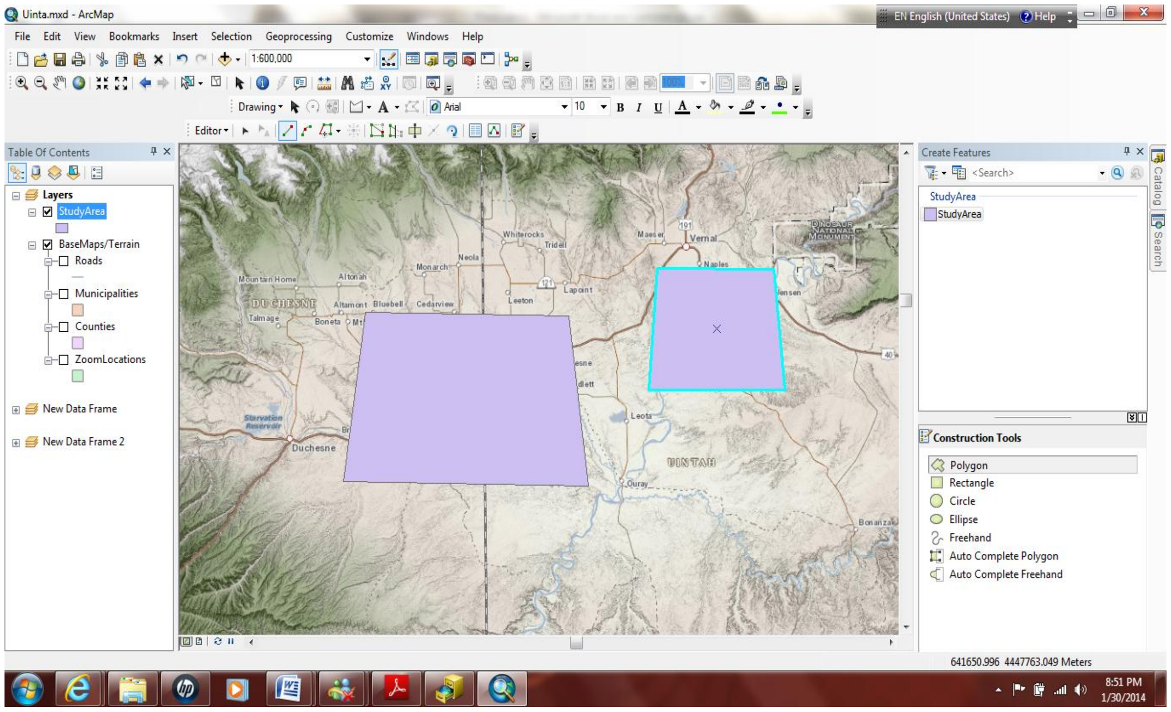


Figure 5 - Drawing polygons in ArcMap

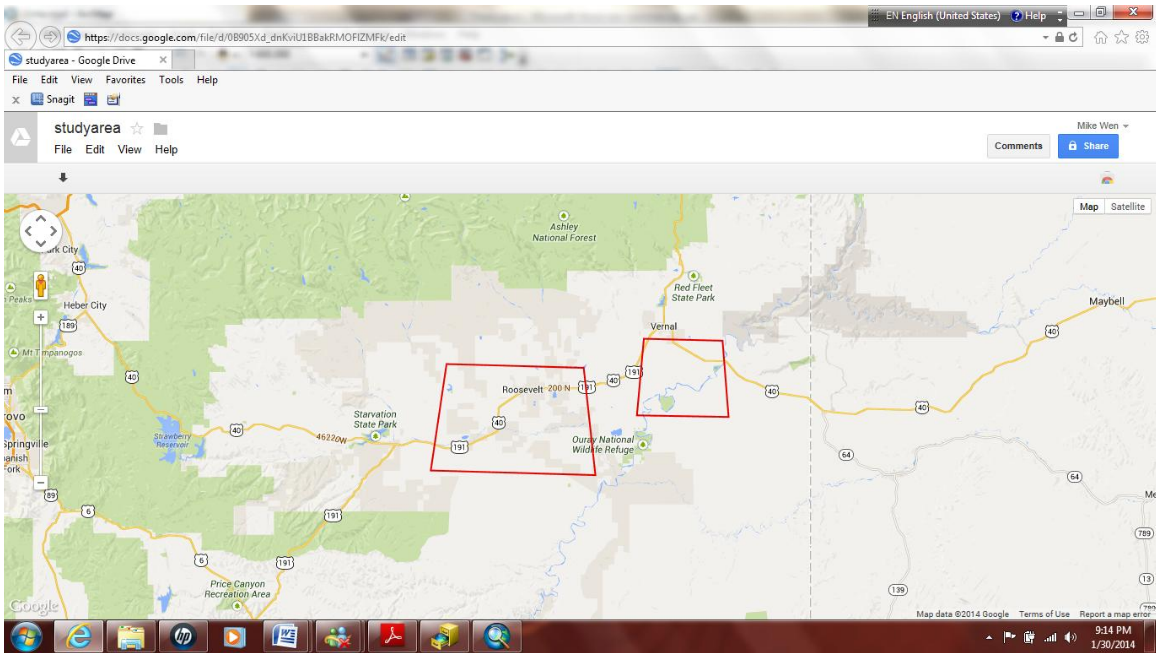


Figure 6 - Showing uploaded polygons in Google map

To demonstrate that the tasks described above can also be done by functions developed in Google Maps API, a drawing feature was developed that also provides the capability to draw “areas of interest” polygons. Users would be able to turn this drawing feature on by clicking on the “Draw” button and then draw the polygons by clicking on the map. The current version of the application allows the users to draw one polygon, which can be deleted by clicking on the “Delete Existing Polygon” button. The locations of the polygons’ vertices are stored in a Fusion Table where each record represents a vertex so that the polygon can be saved and re-displayed at a later time by other users with access to this map mashup application. To better display the area enclosed by the polygons, a fill color of salmon pink was drawn with some degree of transparency as seen in Figure 7. In order to compare and contrast the ease and speed of this approach to the ArcMap and upload approach, the amount of time and number of steps needed to accomplish this task using each method were measured.

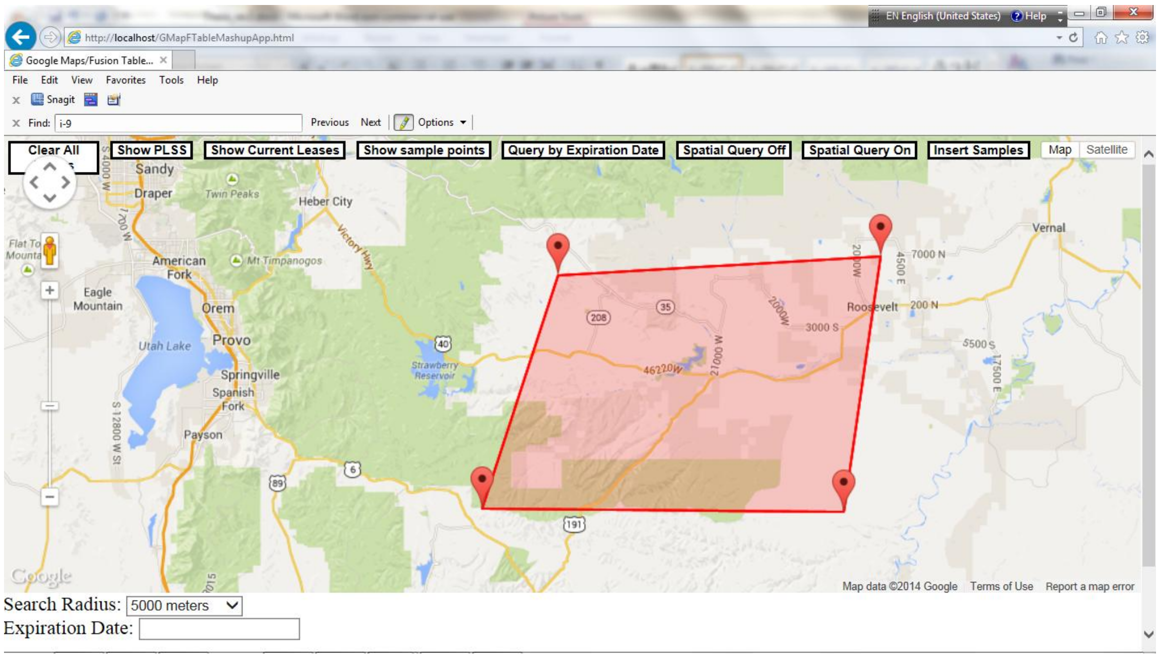


Figure 7 - Drawing polygon on the map mashup

Creating new polygons by using the drawing tool developed for this study is much simpler once the code was written. The new workflow contains only three steps:

1. Create a new Fusion Table or clear the content of an existing one to be used for storing the vertices of the polygons drawn.
2. Click on the “Draw” button to start the drawing process.
3. Draw the polygons.

3.2.4. Attribute/Spatial Queries and Buffers

The next step is to showcase the attribute and spatial querying capabilities of Google Fusion Tables. Currently, if a user wants to perform spatial and/or attribute queries on features residing on an online map, he or she would have to first download the queries into a format that can be loaded into either a desktop GIS or a spreadsheet application, and then perform the queries there. To replace this rather cumbersome approach, a function that provides the ability to select leases based on their location and expiration date was developed. The requirement of the attribute search function is that users would be allowed first to enter a date and then, with the click of the “Query by Expiration Date” button, have the application display all of the leases with expiration date values that are before the entered date. As explained before in section 3.1, expiration date is the most commonly searched field, and that is why it is used as an example in this application. The user also has the option to incorporate location into the search by clicking on the “Spatial Query On” button, select a search point, and then enter a number that restricts the search results to those leases that are within a circle centered on the start point with the entered number of meters as its radius. The spatial search can be turned off by clicking on the “Spatial Query Off” button. If successfully implemented, the ability to perform attribute and spatial queries directly online without first downloading the data to desktop GIS as is currently done can greatly enhance the utility of online maps in GIScience studies.

The drawing of buffer polygons on maps is a commonly used function of desktop GIS and is essential to any online application that supports some GIS functions. Another feature of the application that supplements the two querying functions is that a buffer with the same radius is drawn around the selected point to display the search area. The buffer circle displays the search area on the map and can help the user display the locations of the search results.

In order to build this function, it is necessary to load existing lease data on which to perform the spatial and attribute queries. To that end the current lease information hosted by the Bureau of Land Management’s Land & Mineral Legacy Rehost 2000 System was downloaded and then loaded to a Fusion Table via KML. As shown in Figure

8, the feature layer of the existing leases are more irregular in shape than the PLSS layer. Users can view existing leases by clicking on the “Show Current Leases” button.

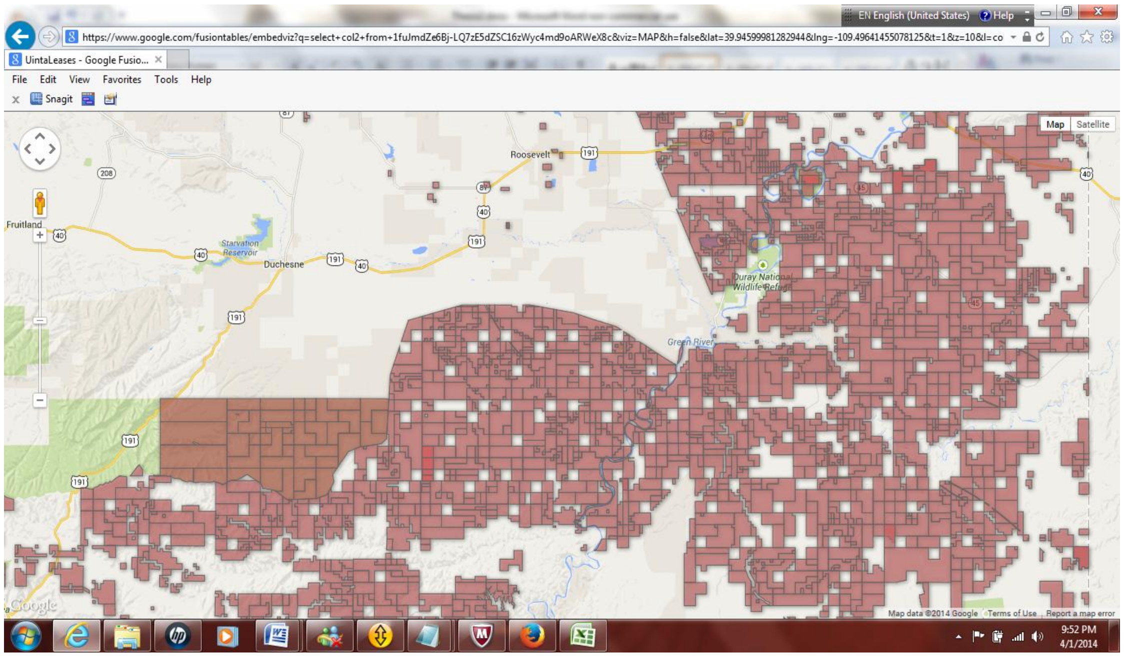

UintaLeases - Google Fusio...
File Edit View Favorites Tools Help
Map
Satellite
Roosevelt
Fruitland
Starvation Reservoir
Duchesne
Ouray National Wildlife Refuge
Green River
Map data ©2014 Google Terms of Use Report a map error
9:52 PM
4/1/2014

Figure 8 – Existing leases

One of the requirements for the application is to provide the ability to search for leases by expiration dates, but since the downloaded data does not contain an expiration date field, a new empty column named “expiration date” was added to the Fusion Table and then populated with sample expiration date values. In order to allow users the ability to specify search criteria such as expiration date and search radius, a dropdown box and a text box were added to the lower left corner of the application. The user also has the ability to switch on both query modes by clicking on buttons in the top right of the application. Figure 9 shows that most of the buttons are lined up in a row at the top of the screen, but some are stacked along the right side as there is no more room at the top.

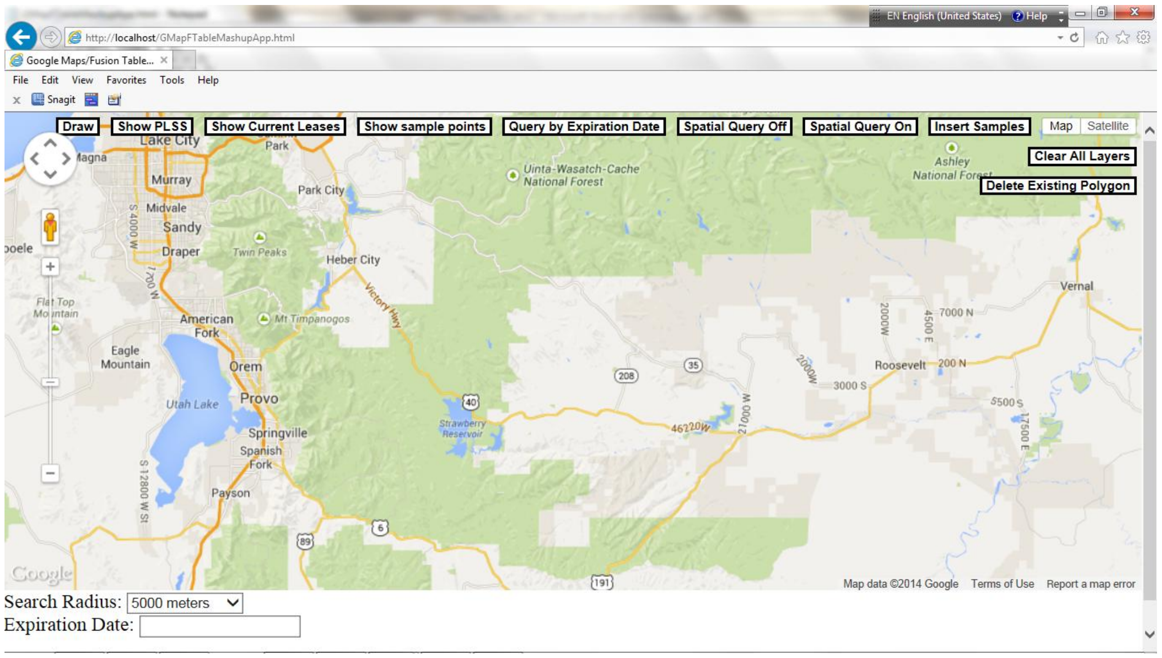
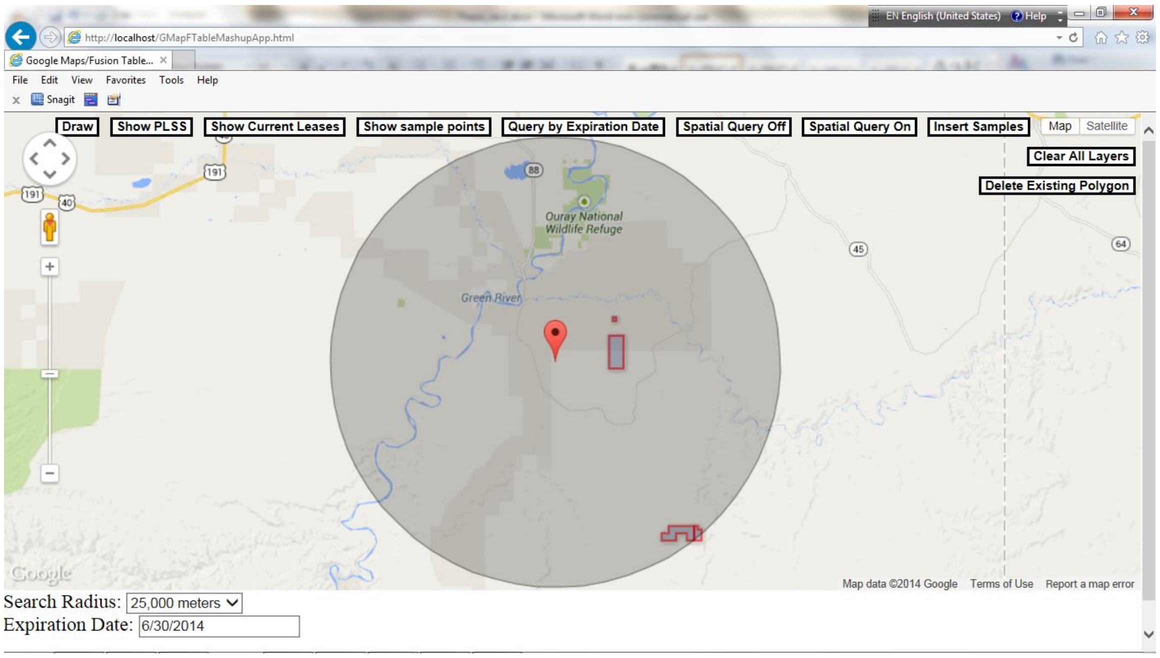


Figure 9 - Map mashup application buttons

As described above, the spatial query function displays the search area by drawing a circular buffer around the selected search point. The size of the buffer depends on the radius selected by the user in the dropdown box in the lowest left corner. All leases that are within the specified distance from the search point are highlighted on the map. If the attribute query functionality is also switched on, then only those leases that satisfy both attribute and spatial criteria are returned as seen in Figure 10.


http://localhost/GMapFTableMashupApp.html
Google Maps/Fusion Table...
File Edit View Favorites Tools Help
Draw
Show PLSS
Show Current Leases
Show sample points
Query by Expiration Date
Spatial Query Off
Spatial Query On
Insert Samples
Map
Satellite
Clear All Layers
Delete Existing Polygon
Ouray National Wildlife Refuge
Green River
Map data ©2014 Google Terms of Use Report a map error

Search Radius: 25,000 meters
Expiration Date: 6/30/2014

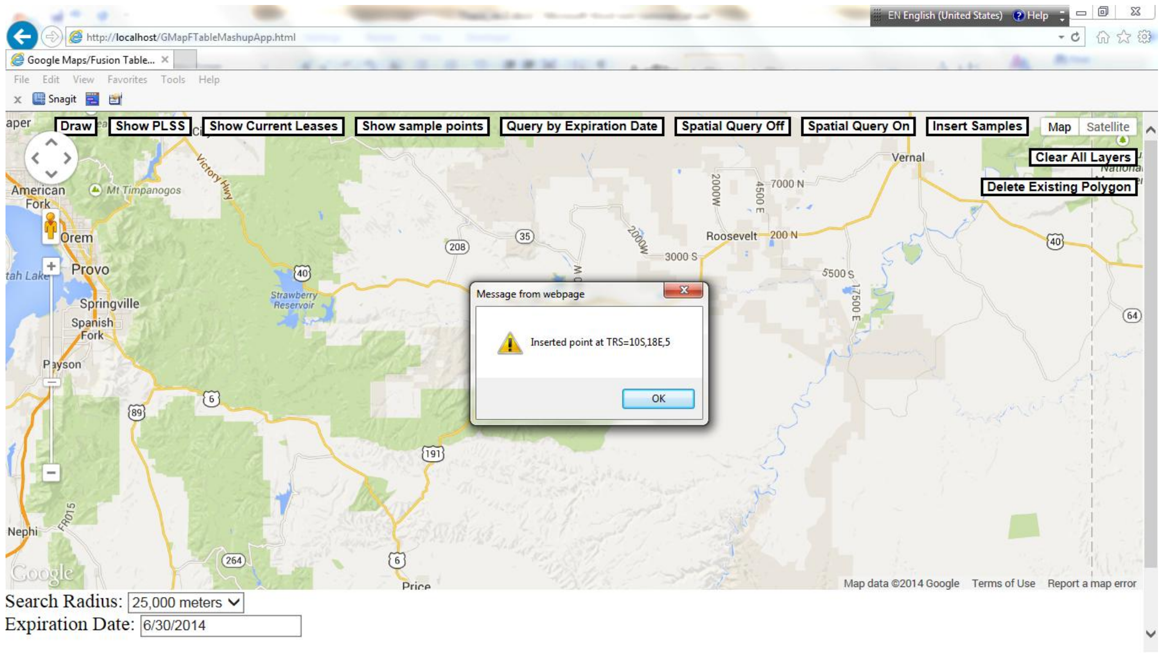
Figure 10 - Spatial and attribute query results

Through cursory examination of the application the two search functions appear to meet the requirements. The spatial search returns leases that contain multiple plots of land and returns polygons outside of the area defined in the buffer zone when part of the lease is inside the buffer, as shown in Figure 10. This can be confusing to some of the new users who are not very familiar with land lease management in oil and gas, but that can be overcome by labeling the leases better. Unfortunately, that is not one of Fusion Table API's strengths.

3.2.5. Using External Data Sources and Data Validation

Finally, to test the accuracy of using an external web service as a data source, a function was built into the mashup application that allows the user to click anywhere within the map to add a new sample point, and the application first displays the township/section/range number in a popup box and then stores it for accuracy assessment later. These sample points will be used for accuracy analysis. A button at the top called

“Insert Samples” is built to let the user start the sample collection process as shown in Figure 11. Unfortunately, this function is affected by the cross-domain restriction described in section 4.3, and since no CORS mechanism is implemented here, this feature is available only if the application is run locally. The user can click on the “Show Sample Points” button to view all of the sample points entered so far.



Message from webpage

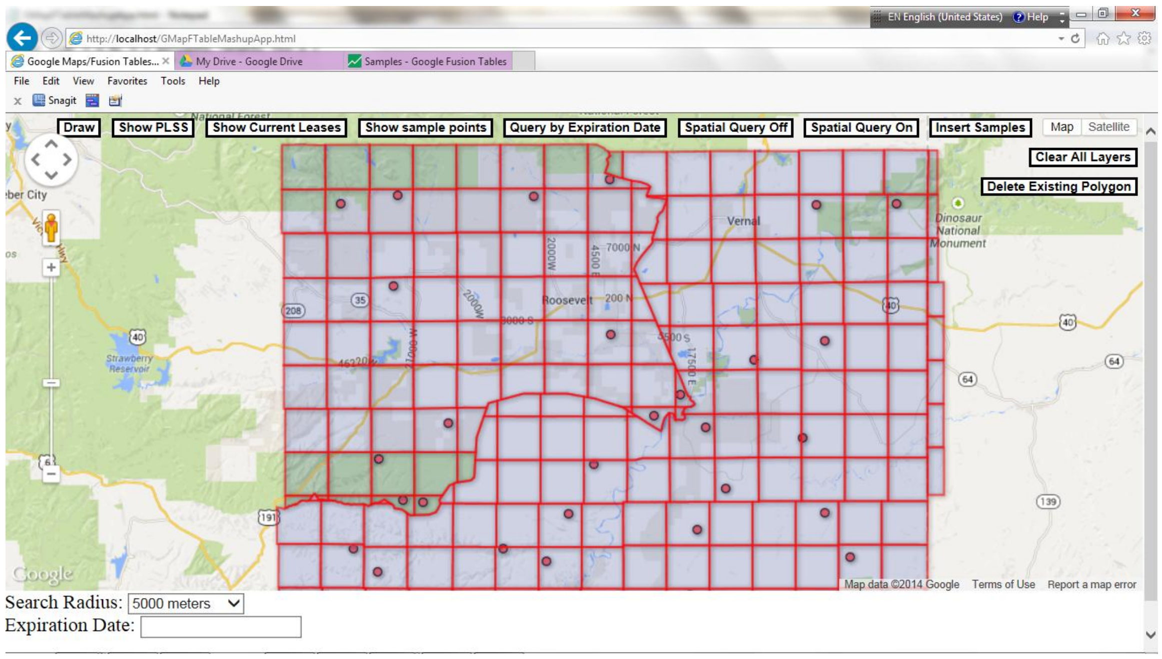
Inserted point at TRS=10S,18E,5

OK

Search Radius: 25,000 meters
Expiration Date: 6/30/2014

Figure 11 - Inserting sample points

To validate the returned values, the location and legal description of the sample points were downloaded in KML format from Fusion Tables and then loaded into ArcMap where they were matched to respective township range section polygons using “Join by Location” spatial queries. The table was then downloaded to an Excel spreadsheet where the two columns were compared by using the Excel “Exact()” function, which returns a True/False answer that can be converted to an integer, in this case a 0 for false and 1 for true, by the formula IF(EXACT(A2,B2),1,0). The average of that column can be calculated by the “Average()” function and can be used as the accuracy of the data as it has the same value as the number of correct values divided by the sample size. Since the data in question is not categorical, an error matrix is not suitable for evaluating the data accuracy in this study and was not used. Thirty sample points were collected in the entire study area. Efforts were made to select sample points near the borders and corners of survey polygons, as these are areas most likely to produce errors. Figure 12 shows some of the sample points selected for accuracy analysis.



Search Radius: 5000 meters
Expiration Date:

Figure 3 - Sample points selected for accuracy assessment

3.2.6. Deploying the Application

The map mashup application developed for this study was placed online and made accessible to others for study. After considering a number of options, it was decided that Google Drives would be used to host the application because it is free and easy to use. To deploy a web page using Google Drives, one simply needed to create a folder and then share it to the public. Anyone with an Internet connection can access the application by clicking on the link below:

https://7bfabc7566f4be707d65d1f22af6187758b45700.googledrive.com/host/0B-d5TwA_JeutZzhmdjFmZFZETXM/GMapFTableMashupApp.html. In order to provide authentication to access the content of the Fusion Tables, the application requires the users to login (Figure 13). A test account was created in Google for this purpose. The login for the account is gisthesisproject@google.com and the password is “nwmissouri”. The user has to log out of all other Google account in order for the authorization process to work.

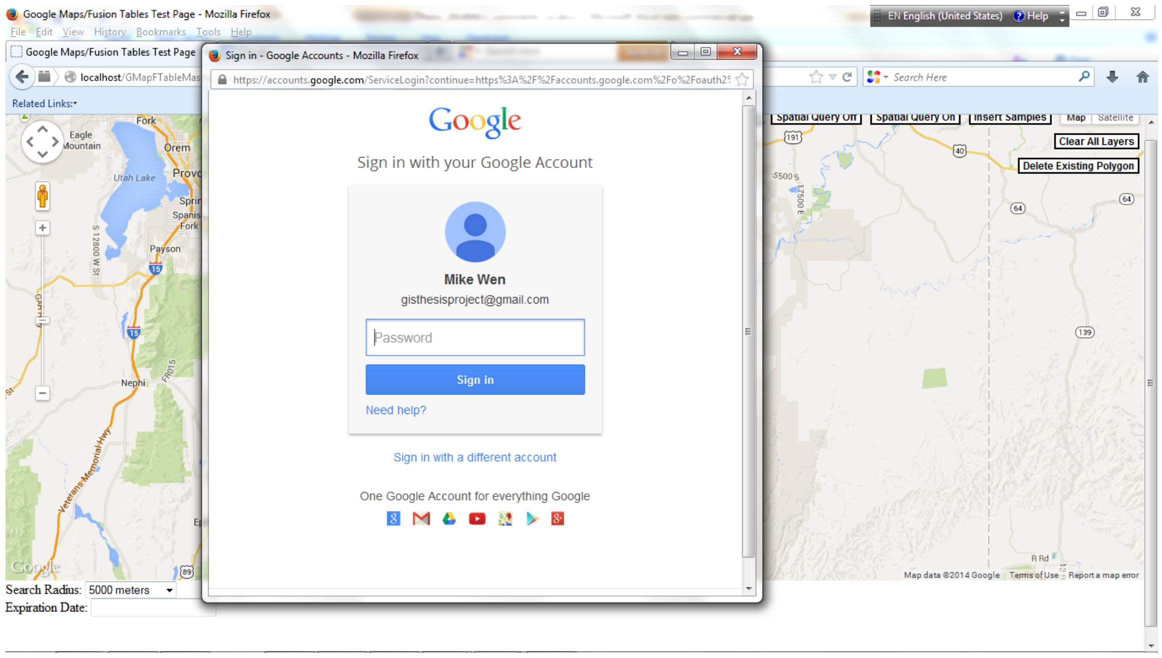
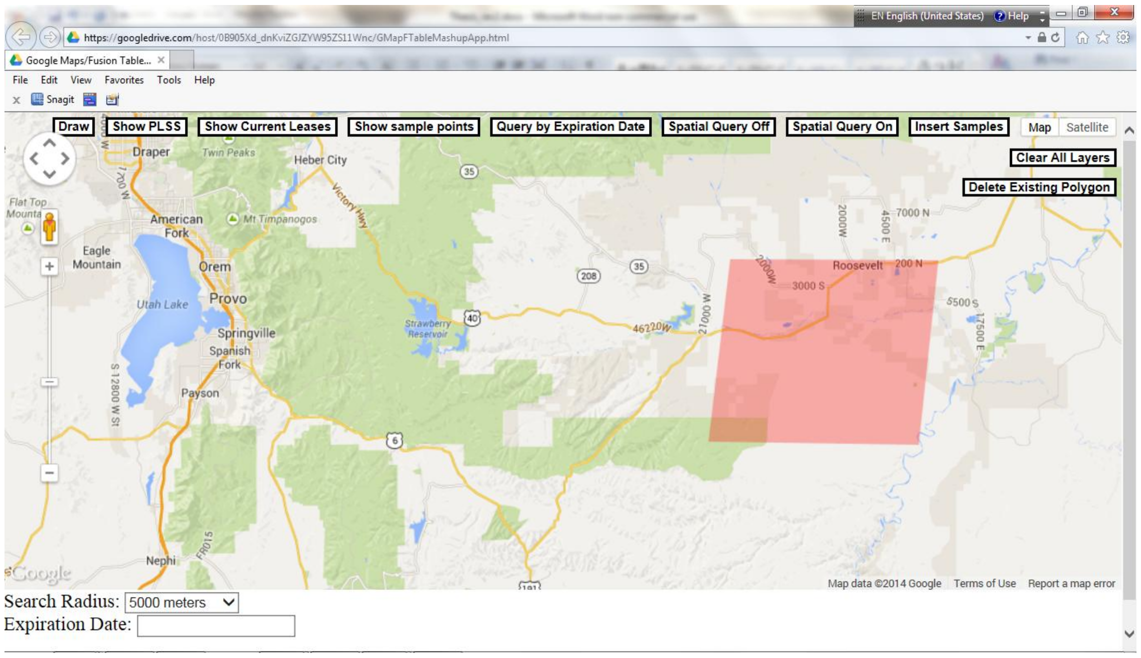

Google Maps/Fusion Tables Test Page - Mozilla Firefox
Sign in - Google Accounts - Mozilla Firefox
Google
Sign in with your Google Account
Mike Wen
gisthesisproject@gmail.com
Password
Sign in
Need help?
Sign in with a different account
One Google Account for everything Google
Search Radius: 5000 meters
Expiration Date:
Clear All Layers
Delete Existing Polygon
Spatial Query Off
Spatial Query On
Insert Samples

Figure 13 – Logging into the application

The first time a user logs into the application a window will popup to prompt the user for the login information provided above. Make sure to allow popup windows when the application is opened; these popup windows are an integral part of the authentication process. Because of this and the CORS issue discussed previously, this application is currently supported on Internet Explorer only.

The application opens a map centered in the study area as can be seen in Figure 14. If the user has previously drawn a polygon with the Draw tool, then that polygon will show automatically as the page is opened. If the screen gets too cluttered with features, the user can clear all feature layers and draw polygons by clicking on the "Clear All Layers" button. As mentioned before, all functions except for those called by "Insert Samples" should work at this link. The source code of the application can be viewed by clicking on https://drive.google.com/?tab=wo&authuser=0#folders/0B-d5TwA_JeutZzhmdjFmZFZETXM.



Search Radius: 5000 meters
Expiration Date:

Figure 14 – Viewing the application on browser

Chapter 4: Analysis Results and Discussion

Contrary to initial expectations, it was possible to fit all of the code for the entire map mashup application into one single html page. The application was written entirely in JavaScript without the use of external libraries except the one for Google Maps API v3. The whole application contains about 1100 lines of code, including the HTML. The level of programming skills required to complete this application could be described as intermediate level, because in addition to HTML and JavaScript some knowledge of XML and object oriented programming concepts were used.

The map mashup application developed for this study satisfied all of the requirements listed in Chapter 3. It provides functions to display feature layers, to draw new polygons, to identify feature layers, to perform attribute and spatial queries, to draw buffers covering the search area, and to use external data sources to obtain township/range/section information based on location.

However, for the most part it would be fair to say that the functions provided by the application were either not as convenient or as powerful as those available in desktop GIS. The literature suggests that there are ways of overcoming some of the limitations discovered over the course of the application's development but they invariably involve more advanced programming techniques. For example, both of the Google Maps API and the Fusion Tables API offer programmatic access to advanced styling features which this application for the most part does not use.

Since JavaScript is a client-side scripting language, the logic of the code was executed by the web browser. This feature facilitated the development process because the code can be run from a basic web server environment without requiring additional modules or add-ons to be installed. The entire development for this study was done from a personal desktop computer running the Windows 7 operating system without using any additional programming development tool. The barrier to entry for developing this type of program is pretty low from a software standpoint.

For this study it is beneficial to release as much of the code as possible to users for review, and so contrary to common programming practices, all of the styling information was placed inside the html page instead of being stored in an external Cascading Style Sheet (CSS) file, and all JavaScript code was entered into the html page instead of using an external JavaScript (.js) file on the server.

4.1 Displaying the PLSS Survey Polygon Layer

The steps involved in displaying the PLSS township polygon as a feature layer are relatively straightforward. While all of the steps involved, from downloading the shapefile from the BLM's website, converting the file to KML, to loading the KML file to Fusion Tables, can be automated in a script, this snapshot approach is not ideal. However, for this study it is not an unreasonable solution because of the relatively static nature of PLSS survey data. It may not be as useful an approach if the spatial layer contains dynamic data, like temperature or rainfall. A more dynamic solution, like using the Web Map Service, would be better for these types of data. However, an effort to use the Web Map Service technology for this study was not successful as WMS is not easily combined with Google Maps (Jackson, 2011).

Another major drawback of the current approach is the difficulty in properly labeling the polygon features. While the detailed display window that pops up when the user clicks on a polygon is a good way for the user to identify a specific polygon's township/range/section legal description, it is not as convenient as showing the legal description directly on the polygons as labels. This is a popular feature request to Google, and it is hoped that an easier approach to labeling features will become a available in the near future (Issue 97, 2010).

4.2 Drawing Areas of Interest

The steps used to create and then export newly drawn polygons used to indicate areas of interest are numerous and time consuming when ArcGIS was involved. They are described in 3.2.3. and the entire process takes about fifteen minutes to complete, including the time spent waiting for applications to load, if step six took the faster route of loading the KML file to Google Drives. If the longer route of using JavaScript is used to create a new FusionTablesLayer, then the process takes as many as twenty two minutes.

The new workflow for drawing new polygons by using the drawing tool developed for the mashup application as described in 3.2.3. contains only three steps and takes only seconds to complete all of the tasks. It would also be possible to enhance the application by developing additional functions such as the ability to draw multiple polygons. Users can use the “Delete Existing Polygon” button to remove a previously drawn polygon and start over. Sometimes users will see the old polygon when they refresh the page due to cache refresh delay **(**Issue 396, 2011**)**.

4.3 Attribute and Spatial Queries

Most of the functions specified in the requirements listed in 3.2 were developed without encountering any significant issues. The only significant issues were encountered during the development of the function that returns the township/range/section legal description by using a web service, and then saves the location and the legal description to a Fusion Table. The security features of most of the common web browsers restrict the access of data sources from outside of the current network domain. In order to bypass this restriction, it was necessary to temporarily disable that security feature for the duration of that feature's use. Obviously, that does not present a sustainable solution for a commercial application, but fortunately there are other programming languages or tools that allow the use of cross-domain web services in map mashup applications without compromising security, such as the use of signed digital certificates, Asynchronous JavaScript and XML (Ajax), and JQuery (Geller, 2013). It was decided that implementing one of these CORS solutions to the application would take too much time and, in the opinion of the author, would not significantly enhance the core value of this study. Implementing a CORS solution was added to the list of topics for further research.

Another setback encountered during the development of the application was caused by Google's policy of setting a default two minute cache for its maps, which impacted the process of sample point collection because the map would not show the newly entered sample points (Issue 396, 2011). The function would be doing its job of inserting the location and the township/range/section legal descriptions returned by the web service of the selected sample points to the Fusion Table used to store them, but these newly added points would not show on the online map until minutes later. This behavior makes it hard to keep track of which areas needed more sample points. Unfortunately, Google has not yet provided an easy way to refresh the map cache through code. As a compromise code was added that adds bubble-like marker symbols that are used as place holders to let the user know the location of the previously added sample points as shown in Figure 15. These marker symbols are not stored and disappear once the browser closes, but they would have done their job before then.

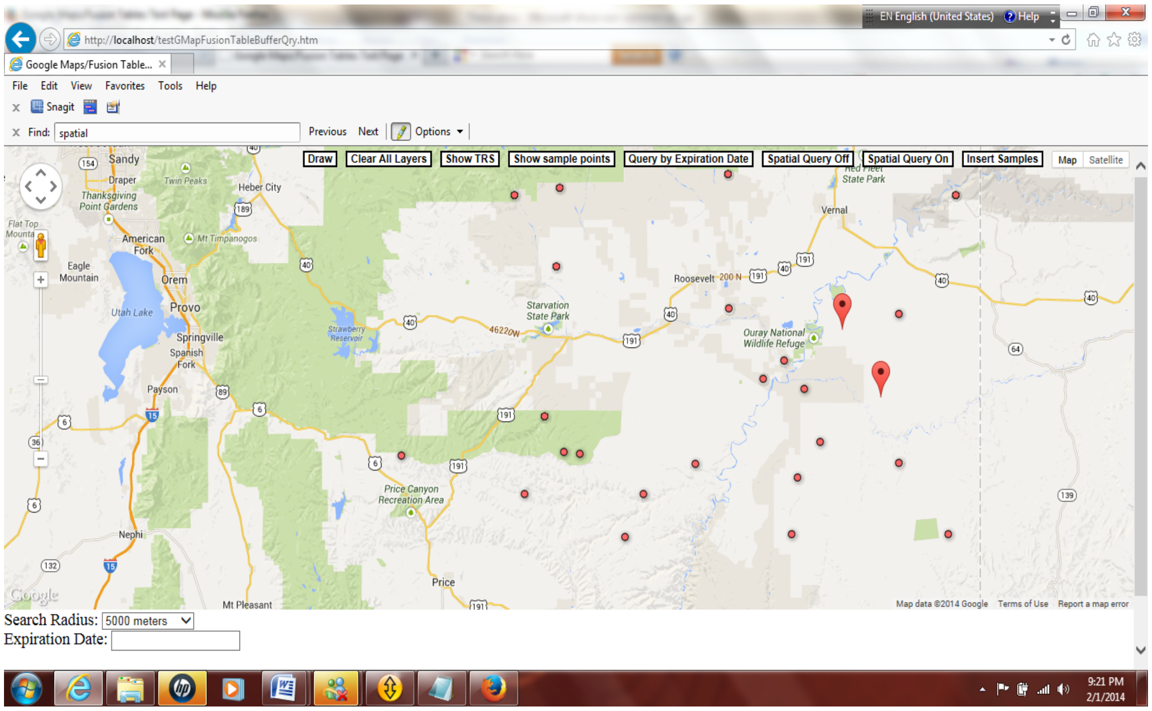


Figure 4 - Markers used to indicate recently added sample points

4.4 Drawing Buffers

One of the functional requirements of the application developed for this study is a function for drawing circular buffers around spatial search starting points. That task was fairly easy using Google Maps API. The following lines of code are all that was required for drawing a translucent circle centered on a point located at latitude y and longitude x with a radius of x meters.

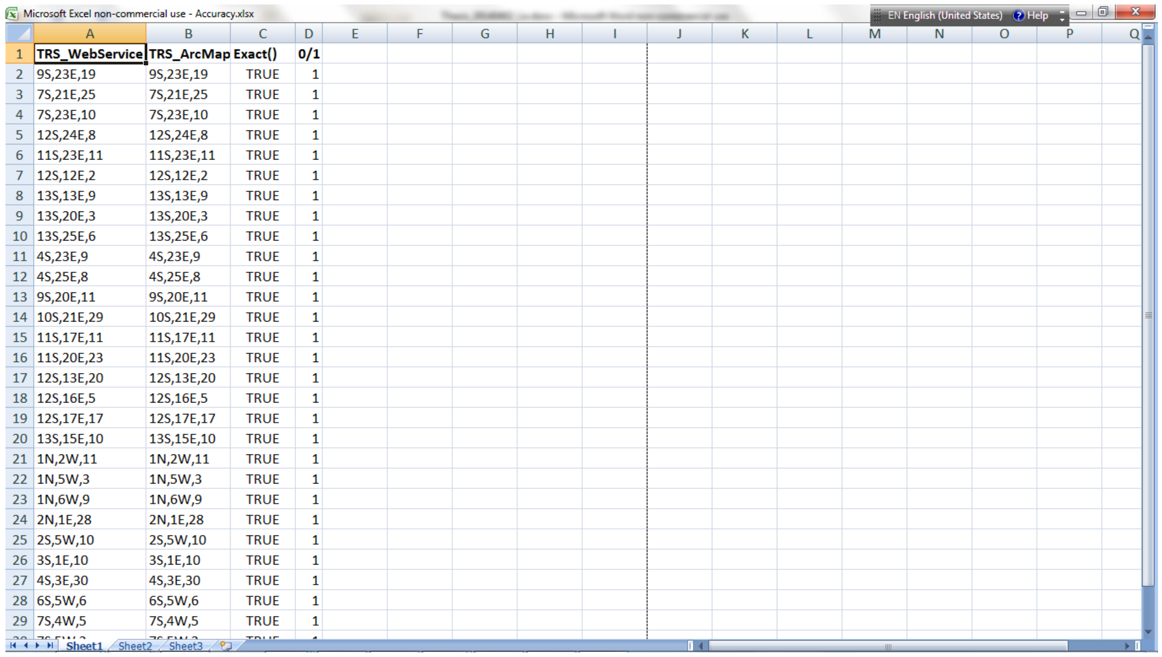
```
circle = new google.maps.Circle({
        center: new google.maps.LatLng(y, x),
        radius: x,
        map: map,
        fillOpacity: 0,
        strokeOpacity: 0.5,
        strokeWeight: 1
});
```

These lines of code create a “circle” object and then proceed to assign various properties to it. These properties can be reset in response to user generated events like mouse clicks to provide interactivity to the user. While drawing circles around points was easy, unfortunately there are no comparable built-in functions in Google Maps API version 3.0 for drawing buffers around polyline or polygon features, but there are third-party libraries like the JSTS Topology Suite that can be easily imported into the application for these purposes (JSTS Topology Suite, 2014).

4.5 Accuracy Assessment

For this study an accuracy assessment was performed for the function that returns the township/range/section legal description of clicked areas from a web service. Sample points were selected from the entire study area, instead of within a small subsection, since an online application such as this one would not likely be used for tasks that require high spatial accuracy like determining survey boundaries. After the function was written, sample points and their legal descriptions were collected from the study area, downloaded from Fusion Table as a KML file, and imported to ArcMap. The point features were then spatially joined to a township/range/section polygons feature layer that was downloaded earlier from BLM, and the legal descriptions of the matched BLM survey polygons were compared to the legal descriptions of the downloaded layer. The downloaded BLM layer functioned as the ground truth data in this case. For this study the only statistic suitable for accuracy assessment is accuracy, calculated by dividing the number of correctly matched points by the number of total sample points.

The result of the accuracy assessment is very encouraging. All of the township/range/section legal descriptions returned by the map mashup application were identical to those returned by spatial query using ArcMap despite efforts by the author to select sample points close to the edge of PLSS polygons where errors are more likely to occur. Figure 16 shows the result of the Excel spreadsheet used to compare the two sets of values. Values of "True" represent identical township/range/section legal descriptions. Such a high level of accuracy is unexpected. An accuracy of 100% indicates that at least an application that uses the TownshipGeocoder web service in conjunction with Google Maps can be ready for prime time for the purpose of land lease management.


Microsoft Excel non-commercial use - Accuracy.xlsx

	A	B	C	D
1	TRS_WebService	TRS_ArcMap	Exact()	0/1
2	9S,23E,19	9S,23E,19	TRUE	1
3	7S,21E,25	7S,21E,25	TRUE	1
4	7S,23E,10	7S,23E,10	TRUE	1
5	12S,24E,8	12S,24E,8	TRUE	1
6	11S,23E,11	11S,23E,11	TRUE	1
7	12S,12E,2	12S,12E,2	TRUE	1
8	13S,13E,9	13S,13E,9	TRUE	1
9	13S,20E,3	13S,20E,3	TRUE	1
10	13S,25E,6	13S,25E,6	TRUE	1
11	4S,23E,9	4S,23E,9	TRUE	1
12	4S,25E,8	4S,25E,8	TRUE	1
13	9S,20E,11	9S,20E,11	TRUE	1
14	10S,21E,29	10S,21E,29	TRUE	1
15	11S,17E,11	11S,17E,11	TRUE	1
16	11S,20E,23	11S,20E,23	TRUE	1
17	12S,13E,20	12S,13E,20	TRUE	1
18	12S,16E,5	12S,16E,5	TRUE	1
19	12S,17E,17	12S,17E,17	TRUE	1
20	13S,15E,10	13S,15E,10	TRUE	1
21	1N,2W,11	1N,2W,11	TRUE	1
22	1N,5W,3	1N,5W,3	TRUE	1
23	1N,6W,9	1N,6W,9	TRUE	1
24	2N,1E,28	2N,1E,28	TRUE	1
25	2S,5W,10	2S,5W,10	TRUE	1
26	3S,1E,10	3S,1E,10	TRUE	1
27	4S,3E,30	4S,3E,30	TRUE	1
28	6S,5W,6	6S,5W,6	TRUE	1
29	7S,4W,5	7S,4W,5	TRUE	1

Figure 5 - Comparison between web service retrieved legal descriptions and those obtained by spatial query using ArcMap

Chapter 5: Conclusion

5.1 Research Summary

This study shows that a map mashup application developed using Google Maps API and Google Fusion Tables API can perform some of the geographical analytical tasks that are now more commonly done in desktop GIS, such as the ability to display polygons as a feature layer, to draw polygons to delineate the boundaries of study areas, to store the location and attributes of sample points selected from a map, to draw buffers around point features, and to perform spatial and attribute queries from a table of spatial records. In addition, such an application can take advantage of functions and data made available by other online service providers through the use of web services to retrieve and process information from external sources. This study also conducted an accuracy assessment on the web service function used to retrieve the township/range/section legal description, and the result showed that the data returned was 100 percent accurate. The application also contains functions that are inherent to Google Map, such as Street View and the ability to switch to satellite imagery and back.

The amount of coding needed to obtain the results was not very extensive, and the development process did not involve the use of any proprietary software development kits or platforms. This study also shows that Google Maps API and Google Fusion Tables API programmed with JavaScript are adequate for the purpose of developing an application meeting the requirements of this study. It seems that the barrier to entry for developing such applications is not very high since no proprietary software development tools were needed, and any GIScience professional with some programming proficiency should be able to take on this challenge.

The technology landscape of online services made available by different providers changes at a very fast pace, and some of the functions developed for this research may either have already been, or will soon be, replicated in services offered by Esri or other GIS vendors. For example, Esri offers the ArcGIS Online service which provides a cloud-based solution that allows users to create, use and share spatial content as published web layers. While using services like ArcGIS Online has its advantages, like the ability to integrate features from other ArcGIS products like ArcGIS for Desktop, ArcGIS for Server, ArcGIS Web APIs, and ArcGIS Runtime SDKs, such benefits only extend to those already invested in ArcGIS software, while Google Fusion Tables is free for personal use. Even though ArcGIS Online contains an option to create a free public account for users, such accounts lack many of the features available to paid subscribers, such as the ability to conduct spatial searches (ArcGISOnline, 2014).

5.2 Limitations of Results

This study also uncovered a number of limitations on these technologies that may need to be overcome before they can become widely adopted. First, although both the Google Maps and Google Fusion Tables API provide an extensive list of functions, for the most part they are not as rich or convenient as traditional desktop GIS. For example, it is fairly difficult to label individual features in a feature layer using Google Maps API while that can be done fairly easily in ArcMap. Customizing the symbology of features requires programming using both APIs unlike in ArcMap where it is a built-in function. It is also fairly difficult to perform common data manipulation tasks like inserting, updating or deleting large numbers of records in Fusion Tables unlike in spreadsheet or relational database management software. The types of spatial queries provided by the current version of Fusion Tables API is also limited; the only type of spatial query that is available now is to retrieve a list of features that are located within the boundary of a geometric shape. ArcMap 10.1, on the other hand, offers fifteen types of spatial selection methods. The current version of Fusion Tables API also lacks the ability to allow multiple tables to be joined on a spatial column (Issue 88, 2010). Another deficiency is that both APIs currently lack intrinsic and easy to use functions to provide for the storage of new features drawn on the map, requiring code to be written for such basic functions.

Other types of limitations encountered in the course of this study involve the development tools used. JavaScript was selected as the programming language of choice for this study because of its relative simplicity and wide availability, not because it is the most powerful programming language in terms of capabilities. As a result of this choice some of the issues encountered during the development of the mashup application would not be an issue if another tool was selected. For example, problems were encountered while writing the function for returning the township/range/section legal description because it used a web service hosted on another server due to the restrictions most web browsers placed on cross-domain requests, since allowing such access opens the door for malwares to do harm. For the purpose of this study that security feature was briefly disabled on the browser, but that clearly is not a suitable approach for a commercial application. The literature, however, suggests that this limitation can be overcome by the use more advanced programming languages and techniques (Geller, 2013).

Another major limitation identified by this study is the significant delay in updating the content of a Fusion Table which was uncovered while developing the

"Delete Existing Polygon" function. Such a delay would limit Fusion Tables to storing information that does not need to be updated quickly or in real time.

Google also sets a number of restrictions on both Google Maps and Fusion Tables to manage their server load. For example, Google Maps are on a two minute cache refresh schedule, and so it was difficult to display recent changes to the data **(**Issue 396, 2011**)**. Fusion Tables also restrict the number of updates it allows each day to 100 for non-commercial users. Finally, since both technologies are provided by Google they are subject to its terms of usage which may be changed at a whim. It may also not be possible to develop certain commercial applications with these tools until Google moves Fusion Tables past the experimental phase (Geographic design constraints, 2014) are as follows.

Google has in place the following technical restrictions for Fusion Tables as stated in the Google Fusion Table's APIs Terms of Service (Using the API, 2014):

- "Only the first 100,000 rows of data in a table are mapped or included in query results.
- Queries with spatial predicates only return data from within this first 100,000 rows. Therefore, if you apply a filter to a very large table and the filter matches data in rows after the first 100K, these rows are not displayed.
- When importing or inserting data, the total size of the data sent in one API call cannot exceed 1MB.
- A cell of data in Fusion Tables supports a maximum of 1 million characters; it may sometimes be necessary to reduce the precision of coordinates or simplify polygon or line descriptions.
- The maximum number of vertices supported per table is 5 million.

- When looking at the map, you may notice: ◦The ten largest-area components of a multi-geometry are shown.
- When zoomed farther out, tables with more than 500 features will show dots (not lines or polygons). ”

5.3 Suggested Further Research

Another study that performs an accuracy assessment on the attribute and spatial querying capabilities of the Fusion Tables API is definitely needed before it can be fully established that the web API technology is ready for prime time. While the map mashup application developed for this study returned results that are consistent with expectations, it would better demonstrate the efficacy of this technology if a statistically significant accuracy analysis is conducted that compares the results of spatial queries done using a map mashup application to those done using conventional desktop GIS software.

Another promising avenue of future research would be to adopt some of the techniques described in the literature to circumvent the technical challenges encountered during the development of this application, such as by using Ajax or JQuery (Geller, 2013). For example, as mentioned before, JQuery can be used to provide access to data provided by web services hosted by a server outside of the current domain without compromising browser security. In addition, the styling features available in both Google Maps API and Fusion Tables API can be explored further as this study did not make extensive use of them. It would also be very useful to find another approach to publish the feature layers besides loading them into Fusion Tables and creating FusionTablesLayer. Techniques such as using WMS should be tried to make this process more automatic and the mapped data layer more dynamic.

Another area worth investigating is to see whether any of the other online mapping tools can offer similar functions with comparable programming effort. Alternative platforms like Yahoo! Maps API, Bing Maps Platform, MapQuest Development Platform, and OpenLayers may provide more functionality or be easier to use. Google Maps was chosen for this study not because it is necessarily the best online mapping tool but because it is the most popular.

References

ArcGISOnline, 2014. [online]. ESRI. Available from: http://www.esri.com/software/arcgis/arcgisonline/features/comparison-table [Accessed 15 June 2014].

Batty, M., 2010. Map mashups, Web 2.0 and the GIS revolution. *Annals of GIS,* 16(1), 1-13.

BLM, 2014. Bureau of Land Management's Land & Mineral Legacy Rehost 2000 System [online]. Available from: http://www.blm.gov/lr2000/ [Accessed 22 January 2014].

Booth, D. 2004. Web Services Architecture. [online]. W3C. Available from: http://www.w3.org/TR/ws-arch/ [Accessed 16 February 2014].

Butler, H., Daly M., Doyle, A., Gillies, S., Schaub, T. and Schmidt, C., 2008. [online]. The GeoJSON Format Specification. Available from: http://geojson.org/geojson-spec.html [Accessed 30 March, 2014].

Chow, T. E., 2008. The potential of maps API for internet GIS applications. *Transactions in GIS*, 12(2), 179–191.

Fuller, C., 2013. [online]. Utah Government Services. Available from: http://historytogo.utah.gov/utah_chapters/the_land/uintabasin.html [Accessed 21 April 2013].

Fusion Table Layers (Experimental), 2014. [online]. Available from: https://developers.google.com/maps/documentation/javascript/layers#FusionTables [Accessed 5 February 2014].

Geller, I. 2013. ASP.Net Web API, CORS Authentication, SSL, and self signing certificate. [online]. Available from: http://www.codeproject.com/Articles/669152/ASP-NET-Web-API-CORS-Authentication-SSL-and-self-s [Accessed 16 February 2014].

GeoCommunicator, 2013. [online]. Available from: http://www.geocommunicator.gov/GeoComm/services.htm#Data [Accessed 23 September 2013].

Geographic design constraints, 2014. [online]. Available from: https://developers.google.com/fusiontables/docs/v1/using [Accessed 21 February 2014].

Getting Started, 2013. [online]. Google Fusion Tables API. Available from: https://developers.google.com/fusiontables/docs/v1/getting_started [Accessed 14 April 2013].

Getting Started with JavaScript, 2014. [online] Google, Inc. https://google-developers.appspot.com/maps/documentation/javascript/tutorial [Accessed 6 February 2014]

Gong, J., 2007. *Mashing up web-based participatory GIS : potential for future e-government services*. Theses (MS). Ryerson University.

Google Maps/Google Earth APIs Terms of Service, 2014. [online]. Google, Inc. Available from: https://developers.google.com/maps/terms?hl=en [Accessed 15 February 2014].

Google Maps for Business, 2014. [online]. Google, Inc. Available from: http://www.google.com/enterprise/mapsearth/products/mapsapi.html [Accessed 14 February 2014].

Issue 88: Support join of a point table with a polygon table, 2010. [online]. Fusion Tables Issue Tracker. Available from: http://code.google.com/p/fusion-tables/issues/detail?id=88 [Accessed 20 February 2014].

Issue 97: Labels on Points or Polygons on the Maps, 2010. [online]. Fusion Tables Issue Tracker. Available from: http://code.google.com/p/fusion-tables/issues/detail?id=97 [Accessed 17 February 2014].

Issue 396: Proper Caching of embedded maps, 2011. [online]. Fusion Tables Issue Tracker. Available from: http://code.google.com/p/fusion-tables/issues/detail?id=396 [Accessed 20 February 2014].

Jackson, G., 2011. Overlaying wms data using google maps v3 api. [online]. Gavinj.net. Available from: http://www.gavinj.net/2011/02/overlaying-wms-data-using-google-maps.html [Accessed 29 March 2014].

JavaScript Introduction, 2014. [online]. W3Schools. Available from: http://www.w3schools.com/js/js_intro.asp [Accessed 11 February 2014].

JSTS Topology Suite, 2014. [online]. bjornharrtell. Available from: https://github.com/bjornharrtell/jsts [Accessed 18 February 2014].

LeMay, R, 2005. Google mapper: Take browsers to the limit. [online]. C/Net. Available from: http://news.cnet.com/Google-mapper-Take-browsers-to-the-limit/2100-1038_3-5808658.html [Accessed 10 January 2014].

Li, S. and Gong, J., 2008. Mashup: A new way of providing web mapping/GIS services. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 37(B4) 639-647.

Map Types - Google Maps JavaScript API v3 — Google Developers, 2014. [online]. Available from: https://developers.google.com/maps/documentation/javascript/maptypes. [Accessed 11 January 2014].

Miller, C. C., 2005. A beast in the field: The Google Maps mashup as GIS/2. *Cartographica*, 41(3), 187-199.

Muro, M., 2013. Custom Google Maps. [online]. Available from: http://webtide.wordpress.com/2008/08/27/custom-google-maps/ [Accessed 22 September 2013].

National Atlas of the United States, 2013a. National Atlas Data Download. [online]. Available from: http://nationalatlas.gov/atlasftp.html#plss00p [Accessed 22 April 2013].

National Atlas of the United States, 2013b. The Public Land Survey System (PLSS) [online]. Available from: http://www.nationalatlas.gov/articles/boundaries/a_plss.html [Accessed 19 April 2013].

Newman, G., Zimmerman, D., Crall, A., Laituri, M., Graham, J. and Stapel, L., 2010. User-friendly web mapping: lessons from a citizen science website. *International Journal of Geographical Information Science.* 24(12), 1851–1869.

Olea, P. P. and Mateo-Tomas, 2013. Assessing species habitat using Google Street View: a case study of cliff-nesting vultures. *Plos One*, 8(1), 1-8.

Peng, Z. and Zhang, C., 2004. The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS). *Journal of Geographical Systems*. 6(2), 95-116.

Polczynski, M. and Polczynski, M., 2013. A Microsoft Excel application for automatically building historical geography GIS maps. *Transactions in GIS*, 2013, 17(1), 148–157.

ProgrammableWeb, 2013. Keeping you up to date with APIs, mashups and the Web as platform [online]. Available from: http://www.programmableweb.com/ [Accessed 19 April 2013].

Taylor, B., 2005. The world is your JavaScript-enabled oyster. [online]. Google Official Blog. Available from: http://googleblog.blogspot.com/2005/06/world-is-your-javascript-enabled_29.html [Accessed 11 February 2014].

TownshipGeocoder, 2014. [online]. Geocommunicator. Available from: http://www.geocommunicator.gov/TownshipGeocoder/TownshipGeocoder.asmx [Accessed 20 January 2014].

Using the API, 2014. [online]. Google Fusion Tables API. Available from: https://developers.google.com/fusiontables/docs/v1/using#Geo [Accessed 31 March 2014].

Wampler, P., Rediske, R. R. and Molla, A. R., 2013. Using ArcMap, Google Earth, and Global Positioning Systems to select and locate random households in rural Haiti. *International Journal of Health Geographics*, 12(3), 1-7.

Zhang, T. and Tsou, M., 2009. Developing a grid-enabled spatial Web portal for Internet GIServices and geospatial cyberinfrastructure. *International Journal of Geographical Information Science*. 23(5), 605-630.