The potential of Maps APIs for internet GIS applications

1 author:

Edwin Chow

Texas State University

22 PUBLICATIONS   **211** CITATIONS

Research Article

The Potential of Maps APIs for Internet GIS Applications

T Edwin Chow
Department of Earth and Resource Science
University of Michigan

Abstract
Since the launching of Maps Application Programming Interfaces (APIs) in 2005, many web developers, including geographers and non-geographers, applauded the freely adaptable tools and used them to spawn numerous Internet applications. The success of the Maps APIs is largely attributable to its no-cost policy, the availability of global data coverage, dynamic navigation, query capability, and ease of implementation. Despite its versatility in dynamic exploration of geographic data online, the existing Maps APIs lack Geographic Information System (GIS) functionalities compared to other Internet Mapping Services. The goal of this research was to review the potential of the Maps APIs for Internet GIS applications. This research employed the Google Maps API and developed a web prototype that disseminates spatial information of urban sprawl in Mundy Township, Michigan. The results revealed that both vector and raster data could be effectively represented by using the Maps API. Moreover, the Geographic Markup Language (GML) approach illustrated great potential for developing Internet GIS solutions around open specifications. This research suggested several potential solutions to expand the spectrum of GIS operations of the Maps APIs by incorporating the XML-related technology and extending the JavaScript library.

1 Introduction

Since the emergence of the Internet in the 1990s, there has been a paradigmatic shift in all aspects of Geographic Information Systems (GIS). The conceptual model (and hence its technology) of GIS has undergone a trend of transformation – from an isolated architecture to an interoperable framework, from a standalone solution to a distributed approach, from individual proprietary data formats to open specification exchange of

Address for correspondence: T. Edwin Chow, University of Michigan-Flint, Department of Earth and Resource Science, Flint, MI 48502, USA. E-mail: chowte@umflint.edu

data, from a desktop platform to an Internet environment. The ontological changes and technological advancement have increased the awareness of GIS's potential among the general public, and also encouraged researchers to explore more powerful GIS techniques.

The recent development of web services, 3-dimensioanl (3D) visualization tools (e.g. Google Earth, World Wind) and Maps Application Programming Interfaces (APIs) have certainly contributed to the ever-increasing attention to the development and implementation of distributed GIS through the Internet. Among the recent advances in Internet technology, the literature has progressively acknowledged the importance of web services and 3D visualization tools in Geographic Information Science (GIScience). Smiatek (2005) described the implementation of web services as a neutral platform for climate models accessing GIS databases. Based on the web services technology, Tait (2005) introduced the concept of a geoportal, a gateway to discover and publish geographic content for developing distributed GIS applications. Butler (2006), Nourbakhsh et al. (2006), and Pearce et al. (2007) praised the valuable 3D visualization tools for scientists in conducting research in 3D space and developing global up-to-date Internet GIS applications. Lisle (2006) provided a list of great examples of using Google Earth for visualizing and exploring many geological landforms. However, unlike the web services and 3D visualization tools, the Maps APIs did not receive the same attention from scientists. Despite the popularity that Maps APIs gained among web developers, relatively little was written documenting the contribution or the potential role of Maps APIs in the development of Internet GIS applications.

The purpose of this research was to assess the potential of the Maps APIs for developing Internet GIS applications. This study suggested a conceptual model for exploring and extending the existing functionalities of Maps APIs in displaying and processing both raster and vector data. In particular, the Google Maps API was adopted to develop a web prototype that disseminates the geographic information of urban sprawl in Mundy Township, Michigan. The web prototype revealed that the spatial and attribute information of a GIS database can be effectively represented in the Geographic Markup Language (GML) by using the Google Maps API. The GML approach illustrated great potential for developing Internet GIS solutions around open specifications through the Maps APIs. This work provided useful insights in the future development of Maps APIs for Internet GIS applications.

2 Background

A Maps API (e.g. Google Maps API, Yahoo! Map Developer API, Mapquest OpenAPI, or Map Control of Microsoft Virtual Earth, ESRI ArcWeb Services) is a source code interface that grants web developers access to a program library and to request services in generating a map over the Internet. The emergence of the Maps APIs was founded on powerful web map servers that provided extensive spatial data coverage around the globe. The spatial data that comprise the Internet map include the map data (e.g. road network, hydrographic features, political boundaries) and remotely sensed imagery (both satellite and aerial). In general, the high resolution imagery (with spatial resolutions of 5 m or less) and street-level map data may only be available in selected metropolitan areas. Thus, a Maps API enables the web developer to request spatial data for a selected geographic region through the Hypertext Transfer Protocol (HTTP) and embed the resulting map as an object in any external web site. The Maps API also allows the flexibility to

add custom map controls, such as a navigation slide bar for zooming in/out and a toggle button to switch between map/aerial and hybrid views, for dynamic navigation by the map users. From the developers' perspective, the access to such valuable spatial data and dynamic functionalities per request can be regarded as a form of distributed Geographic Information Services (GIServices). Table 1 compares the built-in features provided by

Table 1   Comparison of the common built-in Maps APIs[1]

	Google Maps API	Yahoo! Maps Developer API	MapQuest OpenAPI	Virtual Earth Map Control	ArcWeb Service JavaScript API[2]
Remotely Sensed Imagery					
Vertical Angle	Yes	Yes	No	Yes	Yes
Oblique Angle	No	No	No	Yes	No
Map Data					
State/City	International				
Highway	U.S., Canada and selected countries				
Street	U.S., Canada and selected countries				
U.S. Census	No	No	No	No	Yes
Vector Overlay					
Point	Yes	Yes	Yes	Yes	Yes
Polyline	Yes	Yes	No	Yes	Yes
Polygon	Yes	No	No	Yes	Yes
Raster Overlay	Yes	Yes	No	Yes	Yes
Custom GIS Data Overlay	No	No	No	No	Yes
U.S. Traffic Overlay	Yes	Yes	No	Yes	Yes
GeoRSS Overlay	Yes	Yes	No	Yes	Yes
3D Visualization	No	No	No	Yes	No
Address matching	Yes	Yes	Yes	Yes	Yes
Routing	Yes	No	Yes	Yes	Yes
Spatial Query of Custom GIS Data	No	No	No	No	Yes
Thematic Mapping	No	No	No	No	Yes
Scripting Language	JavaScript	JavaScript ActionScript	JavaScript	JavaScript	JavaScript
AJAX	Yes	Yes	No	Yes	Yes
SOAP Option	No	No	No	No	Yes
Plug-in Required	No	Flash Player (optional)	No	3D viewer (optional)	Flash Player
Registration Required	Yes	Yes	Yes	No	Yes

[1] The information presented in this table will vary with the frequent updates of newer versions.
[2] Only the free version is provided here in Table 1. Please refer to the text for a brief discussion of the licensed (i.e. paid) version.
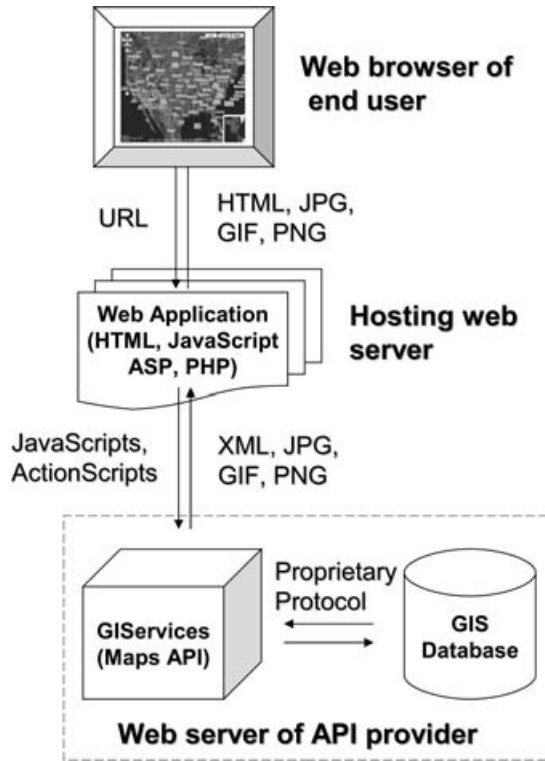
Figure 1   The system architecture of a simple web site that uses the built-in functionalities and data provided by the Maps APIs

the common Maps APIs, including Google, Yahoo, MapQuest, Mircosoft Virtual Earth and ESRI ArcWeb Services.

The conceptual architecture of a web application that uses the Maps API is quite simple (Figure 1). In general, the web application is hosted in a web server that will return Hypertext Markup Language (HTML) and web-compatible graphics (e.g. JPG, GIF, PNG) upon the request of a web browser. By using JavaScript to connect to the Maps APIs, the web application has access to the web servers of the API provider in requesting GIServices, such as zoom in/out. Based on the input parameters and values collected by the map interface of the web application, the web server of the API provider will return the spatial data (i.e. map) in the form of web-compatible graphics. Most API providers (with the exception of Virtual Earth) require the registration of a "map key" in accessing the Maps APIs and/or a limit on the number of page views, queries, and geocode requests per day for a single registered web directory. Currently, most Maps APIs remain free and do not include advertising. It is noted that ESRI ArcWeb Services offer both subscription-based commercial services as well as no-cost public services. The free ArcWeb Services have limitations in accessing some analytical and reporting capabilities, less map data available for overlaying and fewer compatible file formats in data management, while the commercial solutions have no restrictions but charge the web client on a pay-per-click basis.

Since the launching of Maps APIs in 2005, many web developers, including geographers and non-geographers, have applauded the freely adaptable tools that were used to spawn numerous Internet applications. The success of the Maps APIs is largely attributable to its no-cost policy, the availability of extensive data coverage, open specification, ease of implementation, dynamic navigation, and querying capability. Since the web map servers that provide the Maps API services (e.g. Google) are mainly based on JavaScript and eXtensible Markup Language (XML), it is possible to customize the map interface into existing web sites. Many well-known customized "map hacks" or "mashups" geocode point locations of features (e.g. Chicago crime data, photo sharing), or support overlays of polyline and polygon objects using the Maps APIs. Moreover, many developers have been successful in producing client-side or server-side scripts that extend the "out-of-the-box" functionalities in the Maps APIs and incorporate spatial databases in GIS data formats. Despite its versatility in dynamic exploration of geographic data online, the existing Maps APIs lack analytical and spatial functionalities compared to other Internet Mapping Services (e.g. ESRI's ArcIMS). For example, spatial operations like buffering, geoprocessing or map algebra are not supported in the current version of the Maps APIs. While the web technology of Maps APIs is still in its infancy, the lack of World Wide Web Consortium (W3C)-endorsed standards, technical support (such as a knowledge base), and literature reviews may present barriers to some.

Nevertheless, the Maps APIs have spawned numerous Internet applications that allow the users to visualize and query the spatial data. There has not been an official record of the total number of Maps API mashups; but it is evident that the number of new sites is growing every day (Google Maps 2007). At the same time, there have also been more web references for the Maps APIs, including official blogs, wiki-projects (i.e. a web encyclopedia edited by the interested public), online tutorials and mashup examples that support continual development and foster the exchange of information in customizing the Maps APIs. Moreover, the Maps API providers have been improving and enriching the built-in functionalities of their tools for web developers and users. It is expected that the technological gap, in terms of spatial functionalities, between the Maps APIs and the traditional Internet GIS mapping solutions will diminish over time. This article serves as a pilot study in documenting and investigating the potential of Maps APIs for developing Internet GIS applications.

3 Methodology

In order to assess the potential of the Maps APIs for Internet GIS applications, this research proposed a framework for processing and visualizing GML data and custom raster images using the Maps APIs. The proposed conceptual model 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 Maps API classes for visualization. Figure 2 graphically portrays the overview of the conceptual model and the associated technology. On top of utilizing the spatial data stored on a local web server, the proposed framework was also connected, through the Maps APIs, with an external web map server that provides distributed GIServices and auxiliary spatial data. The following sections disclose each step in further detail.
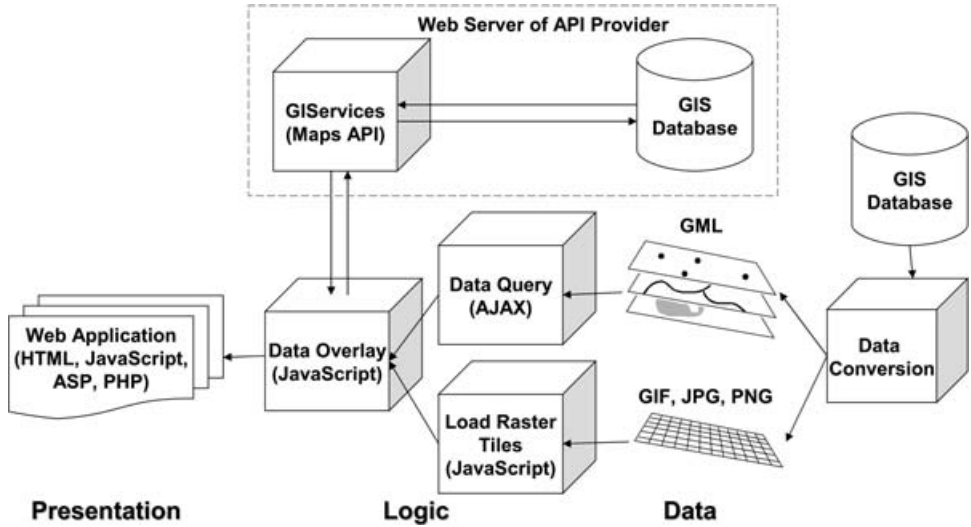
Figure 2 The conceptual model of an Internet GIS application that extends the built-in functionalities of Maps APIs and visualizes custom data

3.1 Data Conversion

The spatial data stored in a GIS repository was first converted to the Geography Markup Language (GML), a data description language for encoding and integrating geographic features, including both spatial and attribute information (Peng and Tsou 2003). In the GML 3.x encoding specification, the core schemata were expanded from the previous version in order to support geometric primitives (e.g. Point, LineString and Polygon), geometric complexes (i.e. closed collection of geometric primitives) and geometric aggregates (e.g. MultiPoint, MultiLineString, MultiPolygon, MultiGeometry). Thus, GML is capable of encoding sophisticated data models and geometries to represent real world objects.

As an extension of eXtensible Markup Language (XML), GML must also be well-formed and validated by an external reference of a GML application schema. Fortunately, there were existing freeware (e.g. GeoCon) and commercial software (e.g. TatukGIS Editor) that could readily convert most vector and raster database formats into GML. If the spatial data were in other open specification formats, such as Scalable Vector Graphics (SVG) or Keyhole Markup Language (KML), an eXtensible Stylesheet Language Transformation (XSLT) could be used to parse the XML-based data and convert them into the required GML standard (Antoniou and Tsoulos 2006). A sample of the GML file that stores the geometry and attributes of a river is represented in Figure 3.

Conceptually, it was possible to encode raster images into GML by using the *<gml:Grid>* or *<gml:RectifiedGrid>* elements (Cox et al. 2005). From the implementation point of view, however, visualizing the raster imagery in a GML format on the Internet was not the most efficient method of presentation. This was owing to the fact that the current version of existing Maps APIs does not support direct parsing and overlays of GML elements in raster format. Hence, the raster imagery was preprocessed in GIS software and converted to a web-compatible graphic file, such as GIF, JPG or PNG.

```
<gml:featureMember>
  <MundyRiver fid="MundyRiver_0">
    <FNODE_>1495</FNODE_>
    <TNODE_>1473</TNODE_>
    <LPOLY_>0</LPOLY_>
    <RPOLY_>0</RPOLY_>
    <LENGTH>0.03764089942</LENGTH>
    <RF3_>81</RF3_>
    <RF3_ID>81</RF3_ID>
    <CU>4080204</CU>
    <SEG>7</SEG>
    <MI>13.98</MI>
    <UP>-1</UP>
    <DOWN>0</DOWN>
    <LENGTH_M>3469.30004882813</LENGTH_M>
    <RF3RCHID>4080204    713.98</RF3RCHID>
    <TheGeometry>
      <gml:LineString>
        <gml:coordinates>
            -83.6894371999026,42.8726272579533
            -83.6895523074905,42.8726272568198
            -83.6897430416561,42.8725090018401
            -83.6903381341629,42.8720054621934
            -83.6909408570235,42.8717269885958
            -83.6908416753941,42.87152099517
            -83.6904014009229,42.8712246572895
        </gml:coordinates>
      </gml:LineString>
    </TheGeometry>
  </MundyRiver>
</gml:featureMember>
<gml:featureMember>
    ...................
</gml:featureMember>
```

Figure 3   A sample file of the GML encoding of a river

For global coverage, it was important to note that most Maps APIs adopt the geographic coordinate system of latitude and longitude (in decimal degrees). Hence, for applications that utilized spatial data that were in a projected map coordinate system (e.g. Universal Transverse Mercator), it was essential to preprocess the spatial data by transforming the coordinate systems and map projection into latitude and longitude before converting the vector/raster data.

3.2 Data Query

After the conversion of spatial data into the proper format, the GML files and web-compatible graphics were loaded as individual data layers in the web browser. To enhance the processing speed and user interaction, this research adopted the web development technique of Asynchronous JavaScript and XML (AJAX) to parse and query the GML files. The concept of AJAX is to exchange only small bits of information with the server behind the scenes to avoid the downtime in reloading the entire web page every time the user sends a request. AJAX is not a new technology in itself but a more efficient way of programming by using a mixed technology of HTML, XML, Cascade StyleSheet (CSS) and JavaScript. In a nutshell, the user can request and get the data from the web server without reloading the page by employing the JavaScript *XMLHttpRequest* object. The Google Maps API tested in this research supports AJAX and provides a similar object called *GXmlHttp*.

Once the web browser gets the GML file, individual elements and tags could be queried by accessing the XML Document Object Model (DOM). An array was created for each data layer to store the spatial geometry and attribute information of all geographic

features from the GML data. Based on this GML approach, it was possible to perform an attribute query to pull only a small subset with specific attributes from the entire database by using XQuery and XPath. XPath is the expression that can be used to select a particular node or element in an XML document. XQuery is the equivalent technology that builds on XPath for XML (similar to Structured Query Language (SQL) for databases). For example, an XQuery can be used to return all the river elements that have a length longer than 3 km in a GML document.

3.3 Data Overlay

Most Maps APIs were equipped with some functionalities for overlaying custom geographic features in the map content. However, not all Maps APIs had the built-in classes developed for visualizing various data models and their geometries (e.g. point, line and polygon in vector) (Table 1). In some cases, the web developers might need to develop customized classes or employ a third-party library to overlay objects of various geometries and data models.

The Google Maps API tested in this research provides built-in support for points, polylines, and polygons (a new class supported as of version 2.69). Once the geographic features of individual layers were queried from the GML data and populated into an array, the spatial and attribute information stored could be used as parameters to create custom overlays of the map content by calling the corresponding classes.

Similarly, it was possible to overlay a raster layer onto the map by using the Maps APIs. For example, with reasonable programming effort (Williams 2006, Mapki 2007), one might add the Digital Orthophoto Quarter Quandrangle (DOQQ) from the U.S. Geological Survey or their own raster imagery onto the map provided by the distributed GIService. For this research, the TPhoto, a Google Maps API extension, was used to simplify the process of web development. With this extension, it was relatively easy to overlay any web-compatible imagery by linking the class with a source and defining the bounding spatial extent in latitude and longitude (Mangan 2007). The extension would also allow setting the opacity of the custom imagery.

3.4 Summary

For illustration purposes, the Google Maps API was adopted in this research. However, the conceptual model could be applied to other Maps APIs as well, assuming the selected Maps API supports the technology needed (e.g. AJAX) (Table 1). This research developed a web prototype that disseminates the spatial information of urban sprawl in Mundy Township, Michigan, as described below.

4 Prototype Implementation

In response to the changing landscape and land use of Genesee County, Michigan in recent years, a research project was initiated to improve land use planning and resource management (Hill-Rowley et al. 2006). Based on a series of conversations with the local communities to understand the dynamics and dimension of land use change (Orfield and Luce 2003), the research project conducted a build-out analysis that involved simulation of suburban growth of Mundy Township under various scenarios. The goal of the
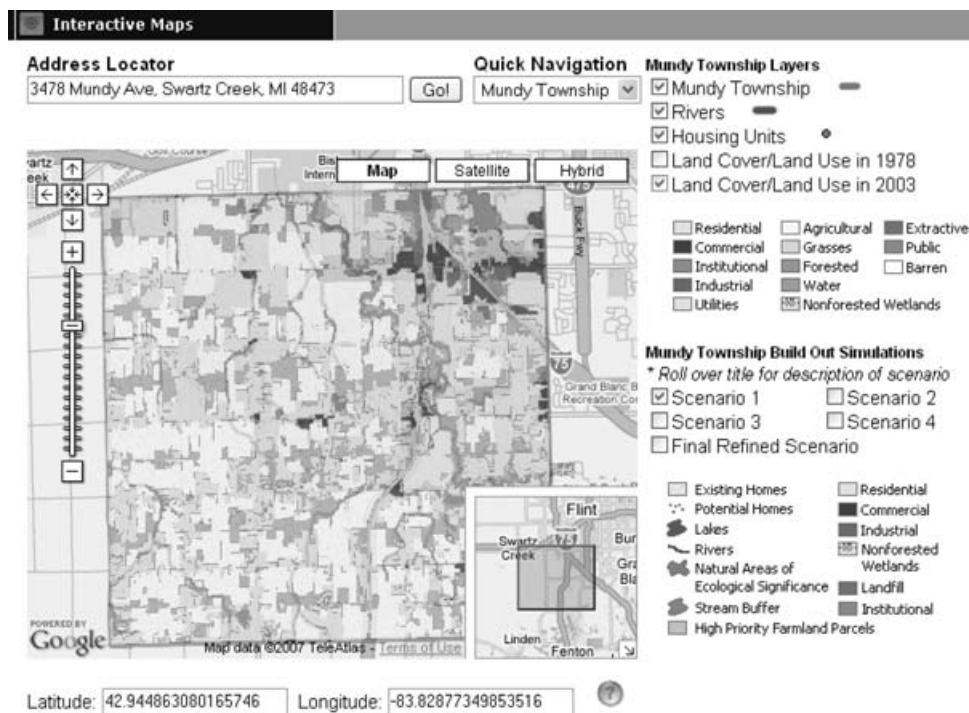
Figure 4   The map interface of the web prototype

project was to aid decision-making for future zone planning by gathering and providing the existing and simulated land use information to the stakeholders. Thus, the web prototype developed in this article served as one of the tools that provided an interactive map of the land use information to the stakeholders and the general public for communication and education purposes.

This case study was conducted to verify and examine the effectiveness and efficiency of the proposed framework in displaying and processing both raster and vector data using the Google Maps API. The web prototype adopted the proposed framework as outlined in the previous section. All files that included spatial data (e.g. GML), program logic (e.g. JavaScript) and presentation tiers (e.g. HTML, JavaScript, CSS) were stored in a web server. Figure 4 illustrates the web interface of the interactive map.

5   Discussion

By using the Maps API, the web prototype proved to be effective in visualizing and presenting spatial data on the Internet. It provided an intuitive and dynamic web interface for novice users to easily explore the distributed geographic information. The "openness" and detailed documentation of the Maps APIs made the task of deploying the mapping service with custom data relatively easy. Some Maps API providers, such as Google, have created a map search wizard that allows one to embed a JavaScript snippet that can be automatically generated by simply filling in the parameters of the desired map. While the novice developers might benefit from such a wizard-driven tool,

the experienced veterans could also adopt the open source code to create sophisticated mashups and even add-on extensions for the Maps API, like the TPhoto extension employed in this research. The warm welcome and general acceptance of Maps APIs among web developer communities has fueled huge demand for continued improvement of the Maps APIs by providing more classes and GIS-like functionalities in successive versions. For example, as requested by popular demand, the support of geocoding (i.e. address matching) and polygon classes were made possible in the revised version of the Google Maps API. It is expected that there will be broader support of technical infrastructure, such as mashup libraries, knowledge bases, user forums and blogs, step-by-step tutorials, or even user conferences in the immediate future.

Another advantage lies in the open specification of the XML technology. As recommended by the Open Geospatial Consortium (OGC), the GML complies with the W3C standards and open specification to store and transport spatial data in a cross-platform Internet environment. The GML-based approach enhances interoperability among other web applications that can parse GML data, such as the Web Feature Service (WFS). Like XML, GML is text-based and can be readily interpreted by people or parsed by an XML-ready client. GML also inherits the advantages of XML in being flexible in creating custom tags and the separation of content and logic from the presentation (Peng and Tsou 2003). By complying with the XML standards, many XML-related technologies could be implemented to extend the built-in functionalities that Maps APIs currently offer. As illustrated in this research, XQuery and XPath can be used to incorporate attribute query of the geographic features. Conceptually, XLink and XPointer could potentially be used for joining or relating the GML data with an external database by creating "hyperlinks" in any XML documents. The joining or relating operations can define the relationship classes, such as one-to-one, one-to-many, many-to-one and many-to-many, between two entities. Users' requests and interaction could be handled by using Simple Object Access Protocol (SOAP), a platform- and language-independent web application that exchanges information over the HTTP based on XML communication. Therefore, the GML approach opens up a new venue for the development of future Internet GIS applications.

This research also exposed some limitations of the Maps APIs, despite their great potential, in developing Internet GIS applications. First of all, the built-in classes of existing Maps APIs lack much spatial and analytical functionality essential in the "traditional" Internet Mapping Services (e.g. ESRI's ArcIMS). Currently, the built-in classes offer limited spatial operations such as distance calculations between two points and routing. As demonstrated in this research, attribute query and joining might also be made possible with other XML-related technology. Other spatial operations that are commonly available in a desktop GIS solution, such as buffering, geoprocessing, spatial interpolation or map algebra, are not available within the current versions of Maps APIs. Such operations require extensive effort in web programming to develop classes that can be added onto the existing Maps APIs. However, this proposed framework suggests that the potential of Maps APIs is not limited to simple GIS functions such as geometry overlay and attribute query. Some Maps APIs offer interfaces to access the properties of basic geometry, vertices of points/polylines/polygons, as well as methods to compare the location, return the distance, and the mid-point of a line. Based on these properties and methods, it is possible to develop more GIS-like functions, such as the containment relationship between a point and a polygon. These spatial operations are essential components of data exploration, analysis and modeling in discovering the spatial relationship of the geographic phenomenon. Thus, to build more sophisticated

Internet GIS applications, future research may emphasize the development and implementation of more spatial operations by extending the functionalities of the Maps APIs.

Besides, the GML approach had implications on the performance of the web application. Since GML data are text-based, they are relatively larger in size than their binary counterparts. In representing general geographic features, the file size of the GML is dependent on the number of features, spatial scale (i.e. fineness of detail) of the geometry of individual features and their associated attributes. The transportation and parsing of a large file over the Internet may be slow and a burden if loaded on a busy web server. This may be particularly true of Internet GIS applications which often contain thousands (if not more) of geographic features with complex geometries. To resolve the performance issue of large file size, potential workarounds include setting the visible scale with "smart" query, using aggregate markers beyond a certain zoom level, or storing the attributes as another external file and querying the attributes of individual features using AJAX. The knowledge base of Maps APIs, such as Mapki for the Google Maps API, provides many helpful tips in optimizing the performance of loading large numbers of geographic features.

To present the raster imagery on the Internet, this research utilized the TPhoto extension, which was extended on the existing built-in function of Google Maps API by adding the imagery as a map tile. At the time of press, the latest version of Google Maps API offered a GGroundOverlay object in overlaying raster images. However, as documented in the previous section, the raster imagery was first converted into a web-compatible graphic format, such as JPG, GIF or PNG, in order to be rendered by the web browsers. Moreover, the existing Maps APIs do not allow accessing the raw imagery at the pixel level. The map tiles that had been added into the web prototype were not associated with any pixel value (or spectral reflectance). Hence, the raster images could only be used for visualization purposes. The users would not be able to query the pixel values of the raster imagery, or to apply other digital image processing techniques, such as band ratio or image classification. A potential solution was to encode the raster imagery into GML by using the *<gml:Grid>* or *<gml:RectifiedGrid>* elements (Cox et al. 2005). Antoniou and Tsoulos (2006) proposed a similar solution by encoding of raster imagery into XML and SVG, which could potentially allow the transportation and parsing of raster imagery at the pixel level. Essentially, this suggestion was similar to a raster-to-vector conversion by translating the raster imagery into a matrix of rectangular polygons that could be overlaid by existing Maps API classes. Another workaround solution was to pass the screen coordinate of the cursor movement into a specified coordinate system which could be used to query an external GML-encoded raster imagery for individual or a block of pixel values. These steps could be implemented by using AJAX to enhance the processing speed and user interaction.

The proposed framework presents an alternative to the OGC's Web Feature Service (WFS) standard, an open source Internet GIS specification. Similar to GML, which defines the encoding of geographic features, the WFS is another XML-based specification that defines the interface to request data access and manipulation to GML documents (Vrettanos 2005). Compared to the framework proposed in this article, WFS has the advantages of being open source and platform-independent. However, WFS is text-based (i.e. XML) and hence it is a difficult task to specify a complex spatial operation such as geoprocessing. On the other hand, most Maps APIs utilize a scripting language (e.g. JavaScript) with many mathematical and logical functions that make it a better candidate than WFS to implement the algorithms of more complex spatial operations. Moreover,

a scripting language offers a natural interfacing between HTML and GML to enhance the user interface and dynamic interaction. The adaptation of a scripting language also allows the embedment of looping structures, logical statements, and event procedures to build a more sophisticated spatial model for future Internet GIS applications.

6 Conclusions

This research proposed a framework to utilize the Maps APIs in visualizing and presenting geographic information, including both vector and raster data. The web prototype produced from this research proved to be effective in providing the users with a dynamic interface for data exploration. The openness of the Maps API source code and flexibility in adopting open-specification data standards illustrate some of the potential of the Maps APIs in developing new Internet GIS applications. Despite the great potential, there were limitations in this approach that require more research and experimentation. In utilizing the text-based GML data, the large file size may prolong the processing time of data transportation, parsing and rendering. Compared to other "traditional" Internet mapping solutions, the existing built-in functionalities of Maps APIs currently offer a small subset of the spatial operators common in GIS processing. In particular, the existing Maps APIs do not support direct access and rendering of raster data that further restrain many raster-based GIS operations. More research and development are needed to expand the spectrum of spatial operations of the Maps APIs by incorporating the XML-related technology and extending the JavaScript library. It is expected that the ever-growing attention and demand in Maps APIs promise great potential for further development and improvement of future versions in developing more sophisticated Internet GIS applications.

Acknowledgements

References

Antoniou B and Tsoulos L 2006 The potential of XML encoding in geomatics converting raster images to XML and SVG. *Computers and Geosciences* 32: 184–94
Butler D 2006 Virtual globes: The web-wide world. *Nature* 439: 776–8
Cox S, Daisey P, Lake R, Portele C, and Whiteside A 2005 Geography Markup Language (GML) Implementation Specification. WWW document, http://www.opengeospatial.org/standards/gml
Google Maps 2007 Google Maps Update. WWW document, http://www.googlemapsmania.blog-spot.com
Hill-Rowley H, Clark E and Sanker L 2006 Guiding growth in Mundy Township, Michigan. *Papers of the Applied Geography Conferences* 29: 263–71

Lisle R J 2006 Google Earth: A new geological resource. *Geology Today* 22(1): 29–32

Mangan T 2007 Tphoto. WWW document, http://gmaps.tommangan.us/tphoto.html

Mapki 2007 Add Your Own Custom Map. WWW document, http://mapki.com/wiki/Add_Your_Own_Custom_Map

Nourbakhsh I, Sargent R, Wright A, Cramer K, McClendon B, and Jones M 2006 Mapping disaster zones. *Nature* 439: 787–8

Orfield M and Luce T 2003 Genesee County Metropatterns: A Regional Agenda for Community and Stability in Genesee County, Michigan. WWW document, http://www.geneseelandnetwork.org/sprawl/narrative/metropattern.html

Pearce J M, Johnson S J and Grant G B 2007 3D-mapping optimization of embodied energy of transportation. *Resources, Conservation and Recycling* 51: 435–53

Peng Z R and Tsou M H 2003 *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Network*. Hoboken, NJ, John Wiley and Sons

Smiatek G 2005 SOAP-based web services in GIS/RDBMS environment. *Environmental Modelling and Software* 20: 775–82

Tait M G 2005 Implementing geoportals: Applications of distributed GIS. *Computers, Environment and Urban Systems* 29: 33–47

Vrettanos P A 2005 Web Feature Service Implementation Specification. WWW document, http://www.opengeospatial.org/standards/wfs

Williams M 2006 Google Maps API Tutorial. WWW document, http://www.econym.demon.co.uk/googlemaps/custommap.htm